IBM

# DB2 UDB Server for OS/390 and z/OS Version 7

## Presentation Guide

- Description of new and enhanced functions
- Evaluation of new features for customer usability
- Guidance for migration planning

Maria Sueli Almeida
Paolo Bruni
Reto Haemmerli
Walter Huth
Michael Parbs
Paul Tainsh

Redbooks

ibm.com/redbooks

International Technical Support Organization

SG24-6121-00

# DB2 UDB Server for OS/390 and z/OS Version 7 Presentation Guide

March 2001

**Take Note!**

Before using this information and the product it supports, be sure to read the general information in Appendix D, “Special notices” on page 535.

**First Edition (March 2001)**

This edition applies to Version 7 of IBM DATABASE 2 Universal Database Server for OS/390 (DB2 UDB Server for OS/390 Version 7), Program Number 5675-DB2.

**Note**

This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE  Building 80-E2
650 Harry Road
San Jose, California 95120-6099

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

**© Copyright International Business Machines Corporation 2001.  All rights reserved.**
Note to U.S Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

# Contents

**Preface** . . . . . xi
The team that wrote this redbook . . . . . xii
Comments welcome . . . . . xiii

**Part 1. Contents and packaging** . . . . . 1

**Chapter 1. DB2 V7 contents and packaging** . . . . . 1
1.1 DB2 V7 at a glance . . . . . 2
1.1.1 Application enablement . . . . . 2
1.1.2 Utilities . . . . . 5
1.1.3 Network computing . . . . . 7
1.1.4 Performance and availability . . . . . 8
1.1.5 Data Sharing . . . . . 9
1.1.6 New features and tools . . . . . 10
1.1.7 Installation and migration . . . . . 11
1.2 DB2 V7 packaging . . . . . 11
1.2.1 Base Engine and no-charge features . . . . . 11
1.2.2 Optional no-charge features . . . . . 12
1.2.3 DB2 Management Clients Package . . . . . 14
1.3 Charge features . . . . . 15
1.4 DB2 Warehouse Manager . . . . . 16
1.4.1 Query Management Facility . . . . . 17
1.4.2 Net Search Extender . . . . . 17

**Part 2. Application enablement** . . . . . 19

**Chapter 2. SQL enhancements** . . . . . 21
2.1 UNION everywhere . . . . . 23
2.1.1 Subselects and all that . . . . . 24
2.1.2 Unions in views . . . . . 26
2.1.3 Unions in table-spec . . . . . 29
2.1.4 Unions in basic predicates . . . . . 30
2.1.5 Unions in quantified predicates . . . . . 31
2.1.6 Unions in the EXISTS predicate . . . . . 32
2.1.7 Unions in IN predicate . . . . . 33
2.1.8 Unions in INSERT and UPDATE . . . . . 34
2.1.9 Explaining the UNION . . . . . 36
2.2 Enhanced management of constraints . . . . . 38
2.2.1 Constraints in DB2 V6 . . . . . 39
2.2.2 Consistent constraint syntax . . . . . 42
2.2.3 Restriction on dropping indexes . . . . . 44
2.2.4 New constraint catalog tables . . . . . 46
2.2.5 SYSTABLES constraint column changes . . . . . 49
2.3 Scrollable cursors . . . . . 50
2.3.1 Cursor type comparison . . . . . 52
2.3.2 Declaring a scrollable cursor . . . . . 54
2.3.3 Opening a scrollable cursor . . . . . 55
2.3.4 Fetching rows . . . . . 57
2.3.5 Moving the cursor . . . . . 58
2.3.6 Insensitive and sensitive cursors . . . . . 61

© Copyright IBM Corp. 2001

2.3.7 Resolving functions during scrolling . . . . . 64
2.3.8 Update and delete holes . . . . . 66
2.3.9 Maintaining updates . . . . . 68
2.3.10 Insensitive scrolling and holes . . . . . 69
2.3.11 Locking for scrollable cursors . . . . . 70
2.3.12 Optimistic locking . . . . . 72
2.3.13 Stored procedures and scrollable cursors . . . . . 75
2.3.14 ODBC calls for scrollable cursors . . . . . 76
2.3.15 Distributed processing . . . . . 79
2.3.16 Scrollable cursor usage . . . . . 80
2.4 Row expression for IN subquery . . . . . 81
2.4.1 Row expressions and IN predicate . . . . . 82
2.4.2 Quantified predicates . . . . . 84
2.4.3 Row expression in basic predicates . . . . . 86
2.5 Limited fetch . . . . . 87
2.5.1 Fetching n rows . . . . . 88
2.5.2 Limiting rows for SELECT...INTO . . . . . 89
2.6 Self-referencing UPDATE/DELETE . . . . . 91
2.6.1 Executing the self-referencing UPDATE/DELETE . . . . . 91
2.6.2 Restrictions on usage . . . . . 92

**Chapter 3. Language support** . . . . . 93
3.1 Precompiler Services . . . . . 95
3.1.1 What are Precompiler Services? . . . . . 95
3.1.2 Program preparation today . . . . . 97
3.1.3 Using Precompiler Services . . . . . 98
3.1.4 How Precompiler Services works . . . . . 99
3.2 DB2 REXX language support . . . . . 101
3.3 SQL Procedures . . . . . 102
3.4 DB2 Java support . . . . . 104
3.4.1 JDBC and SQLJ . . . . . 106
3.4.2 What is JDBC 2.0? . . . . . 108
3.4.3 DB2 UDB for OS/390 and JDBC . . . . . 111
3.4.4 JDBC 2.0 DataSource support . . . . . 114
3.4.5 JDBC 2.0 connection pooling . . . . . 117
3.4.6 JDBC 2.0 distributed transactions . . . . . 119
3.4.7 Other JDBC enhancements . . . . . 122
3.5 Java stored procedures and Java UDFs . . . . . 124
3.5.1 Java terminology . . . . . 126
3.5.2 DB2 changes overview . . . . . 127
3.5.3 DB2 catalog changes . . . . . 128
3.5.4 Built-in stored procedures . . . . . 132
3.5.5 New authorizations . . . . . 135
3.5.6 CREATE PROCEDURE in DB2 V6 . . . . . 137
3.5.7 Runtime environment overview V5/V6 . . . . . 138
3.5.8 CREATE PROCEDURE in DB2 V7 . . . . . 140
3.5.9 The external-java-routine-name . . . . . 142
3.5.10 Runtime environment V7 . . . . . 144
3.5.11 Java stored procedures - preparation . . . . . 146
3.5.12 Preparation without the SPB . . . . . 147
3.5.13 Using the SPB . . . . . 148
3.5.14 DSNTJSPP input parameters . . . . . 150
3.5.15 Stored Procedure Builder flow . . . . . 151

3.5.16 The DSNTJSPP flow . . . . . 152
3.5.17 Stored procedure address space JCL . . . . . 154
3.5.18 Setup errors _ 1 . . . . . 155
3.5.19 Setup errors _ 2 . . . . . 156
3.5.20 Runtime errors . . . . . 156
3.5.21 Considerations . . . . . 157

**Chapter 4. DB2 Extenders** . . . . . 159
4.1 Introduction to DB2 Extenders . . . . . 160
4.1.1 What are DB2 Extenders? . . . . . 160
4.1.2 Extenders approach . . . . . 161
4.1.3 Architectural view of DB2 Extenders . . . . . 162
4.2 DB2 Text Extender . . . . . 163
4.2.1 What is DB2 Text Extender? . . . . . 163
4.2.2 Text Extender packaging . . . . . 165
4.2.3 Text Extender indexing . . . . . 167
4.2.4 DB2 Image Extender . . . . . 170
4.2.5 DB2 Audio Extender . . . . . 171
4.2.6 DB2 Video Extender . . . . . 172
4.3 DB2 XML Extender . . . . . 173
4.3.1 What is XML? . . . . . 175
4.3.2 Example: simple stock trade data . . . . . 178
4.3.3 XML and HTML . . . . . 179
4.3.4 XML and DB2 . . . . . 181
4.3.5 XML Extender functions . . . . . 183

**Part 3. Utilities** . . . . . 187

**Chapter 5. Utilities** . . . . . 189
5.1 Utilities with DB2 V7 . . . . . 190
5.2 New packaging of utilities . . . . . 192
5.3 Dynamic utility jobs . . . . . 194
5.3.1 Dynamic allocation of data sets . . . . . 196
5.3.2 Processing dynamic lists of DB2 objects . . . . . 210
5.3.3 TEMPLATE and LISTDEF combined . . . . . 223
5.3.4 Library data set and DB2I support . . . . . 226
5.3.5 Several other functions: OPTIONS . . . . . 229
5.3.6 Utility support for PREVIEW . . . . . 231
5.4 A new utility - UNLOAD . . . . . 232
5.4.1 Enhanced functionalities . . . . . 233
5.4.2 UNLOAD - syntax diagram, main part . . . . . 235
5.4.3 Unloading from table spaces . . . . . 236
5.4.4 Unloading from copy data sets . . . . . 239
5.4.5 Output data sets . . . . . 241
5.4.6 Output formatting . . . . . 243
5.4.7 LOBs and compressed data . . . . . 246
5.5 LOAD partition parallelism . . . . . 247
5.5.1 Parallel LOAD jobs per partition . . . . . 248
5.5.2 Partition parallel LOAD without PIB . . . . . 250
5.5.3 Partition parallel LOAD with PIB . . . . . 251
5.5.4 LOAD - syntax enhancement . . . . . 253
5.5.5 Other considerations . . . . . 254
5.6 Cross Loader . . . . . 255

5.7 Online Reorg enhancements . . . . . 257
5.7.1 Fast SWITCH. . . . . 258
5.7.2 Fast SWITCH - what else to know . . . . . 260
5.7.3 Fast SWITCH - termination and recovery . . . . . 262
5.7.4 BUILD2 parallelism . . . . . 263
5.7.5 DRAIN and RETRY . . . . . 265
5.8 Online LOAD RESUME . . . . . 266
5.8.1 Mixture between LOAD and INSERT . . . . . 267
5.8.2 More on Online LOAD RESUME . . . . . 269
5.9 Statistics history . . . . . 270
5.10 CopyToCopy . . . . . 271

**Part 4. Network computing** . . . . . 273

**Chapter 6. Network computing** . . . . . 275
6.1 Global transactions . . . . . 277
6.2 Security enhancements . . . . . 280
6.2.1 Kerberos . . . . . 282
6.2.2 Encrypted userid and password . . . . . 292
6.2.3 Encrypted change password . . . . . 294
6.2.4 CONNECT with userid and password . . . . . 295
6.3 UNICODE . . . . . 297
6.3.1 UNICODE fundamentals . . . . . 299
6.3.2 UCS-2 . . . . . 300
6.3.3 UCS-4 . . . . . 301
6.3.4 UTF-8 . . . . . 302
6.3.5 UTF-16 . . . . . 304
6.3.6 UNICODE examples . . . . . 305
6.3.7 DB2 and UNICODE . . . . . 306
6.3.8 DB2 support for UNICODE. . . . . 307
6.3.9 Storing UNICODE data . . . . . 308
6.3.10 Access to UNICODE data . . . . . 310
6.3.11 New options. . . . . 311
6.3.12 UNICODE and DB2 system data . . . . . 313
6.3.13 DECLARE HOST VARIABLE statement. . . . . 315
6.3.14 SQL support for UNICODE . . . . . 317
6.3.15 Routines and functions . . . . . 320
6.3.16 Utility support for UNICODE. . . . . 322
6.3.17 UNICODE considerations. . . . . 324
6.4 Network monitoring enhancements . . . . . 326

**Part 5. Performance and availability** . . . . . 327

**Chapter 7. Performance and availability** . . . . . 329
7.1 DB2 and z/OS . . . . . 331
7.2 Parallelism for IN-list index access . . . . . 333
7.3 Transform correlated subqueries . . . . . 334
7.4 Partition data sets parallel open. . . . . 336
7.5 Asynchronous INSERT preformatting . . . . . 337
7.6 Fewer sorts with ORDER BY . . . . . 339
7.7 MIN/MAX set function improvement. . . . . 340
7.8 Index Advisor. . . . . 341
7.9 Online subsystem parameters . . . . . 342

7.9.1 SET SYSPARM command . . . . . 343
7.9.2 Effects of SET SYSPARM command . . . . . 344
7.9.3 Parameter behavior with online change . . . . . 346
7.10 Log manager enhancements . . . . . 350
7.10.1 Suspend update activity . . . . . 351
7.10.2 Retry critical log read access errors . . . . . 356
7.10.3 Time interval checkpoint frequency . . . . . 358
7.10.4 Time driven checkpoints . . . . . 359
7.10.5 SET LOG command . . . . . 361
7.10.6 Long running UR warning enhancement . . . . . 363
7.11 Consistent restart enhancements . . . . . 365
7.11.1 Recover postponed UR . . . . . 367
7.11.2 Cancel thread no backout . . . . . 371
7.11.3 Consistent restart enhancements support . . . . . 373
7.11.4 Adding work files . . . . . 377

**Part 6. DB2 Data Sharing** . . . . . 379

**Chapter 8. DB2 Data Sharing** . . . . . 381
8.1 Coupling Facility Name Class Queues . . . . . 383
8.2 Group Attach enhancements . . . . . 385
8.2.1 How Group Attach works . . . . . 386
8.2.2 Group Attach problem . . . . . 388
8.2.3 Group Attach with NO GROUP option . . . . . 389
8.2.4 Group Attach STARTECB support . . . . . 390
8.2.5 Group Attach support for DL/I Batch . . . . . 391
8.3 IMMEDWRITE bind option . . . . . 392
8.3.1 IMMEDWRITE Bind option before V7 . . . . . 392
8.3.2 IMMEDWRITE BIND option in V7 . . . . . 394
8.4 DB2 Restart Light . . . . . 396
8.5 Persistent CF structure sizes . . . . . 400
8.6 Miscellaneous items . . . . . 403

**Part 7. DB2 features and tools** . . . . . 405

**Chapter 9. DB2 features** . . . . . 407
9.1 DB2 Management Clients Tools Package . . . . . 407
9.1.1 DB2 Control Center . . . . . 407
9.1.2 DB2 Installer . . . . . 408
9.1.3 DB2 Visual Explain . . . . . 410
9.1.4 DB2 Estimator . . . . . 410
9.1.5 Stored Procedure Builder . . . . . 412
9.2 Net.Data . . . . . 413
9.3 DB2 Warehouse Manager . . . . . 415
9.3.1 Information Catalog . . . . . 417
9.3.2 Information Catalog connections . . . . . 418
9.3.3 DB2 Warehouse Manager Agent . . . . . 419
9.3.4 The OS/390 Agent . . . . . 420
9.3.5 Data transfer . . . . . 421
9.3.6 User defined programs . . . . . 422
9.3.7 Submitting JCL . . . . . 423
9.3.8 Triggering steps from OS/390 . . . . . 424
9.3.9 Accessing Data Joiner . . . . . 425

9.3.10 Accessing IMS and VSAM . . . . . 426
9.3.11 Executing DB2 utilities . . . . . 427
9.3.12 Activate replication. . . . . 428
9.3.13 OS/390 Agent installation . . . . . 429
9.4 QMF . . . . . 432
9.5 DB2 Net Search Extender . . . . . 433
9.5.1 Key features . . . . . 433
9.5.2 Implementation tasks . . . . . 435

**Chapter 10. DB2 tools** . . . . . 437
10.1 DB2 tools at a glance . . . . . 439
10.2 IBM DB2 Administration Tool . . . . . 441
10.3 IBM DB2 Object Restore . . . . . 443
10.4 IBM DB2 High Performance Unload . . . . . 444
10.5 IBM DB2 Log Analysis Tool . . . . . 446
10.6 IBM DB2 Table Editor . . . . . 447
10.7 IBM DB2 Automation Tool . . . . . 450
10.8 IBM DB2 Archive Log Compression Tool . . . . . 451
10.9 IBM DB2 Object Comparison Tool . . . . . 452
10.10 IBM DB2 Performance Monitor . . . . . 454
10.11 IBM DB2 SQL Performance Analyzer. . . . . 456
10.12 IBM DB2 Query Monitor . . . . . 458
10.13 IBM DB2 DataPropagator . . . . . 459
10.14 IBM DB2 Row Archive Manager. . . . . 462
10.15 IBM DB2 Recovery Manager . . . . . 464
10.16 IBM DB2 Change Accumulation Tool . . . . . 466
10.17 IBM DB2 Bind Manager . . . . . 467
10.18 IBM DB2 Web Query . . . . . 468

**Part 8. Installation and migration** . . . . . 469

**Chapter 11. Installation** . . . . . 471
11.1 Direct migration from V5 or V6 to V7 . . . . . 473
11.2 Instrumentation enhancements . . . . . 475
11.3 Statistics history . . . . . 477
11.4 UNICODE . . . . . 479
11.5 Data sharing enhancements . . . . . 481
11.6 DBADM authority for create view . . . . . 483
11.7 Checkpoint parameter enhancements . . . . . 485
11.8 Maximum EDM data space size . . . . . 487
11.9 New job DSNTIJMP . . . . . 488
11.10 Star join performance enhancement . . . . . 489
11.11 Installation samples . . . . . 491

**Chapter 12. Migration and fallback** . . . . . 495
12.1 Migration improvements. . . . . 497
12.2 Migration considerations . . . . . 499
12.2.1 Non-data sharing . . . . . 500
12.2.2 Data sharing . . . . . 500
12.2.3 Required maintenance . . . . . 501
12.2.4 Immediate write . . . . . 502
12.2.5 Enhanced management of constraints . . . . . 502
12.2.6 Java support . . . . . 503

12.3 Release incompatibilities . . . . . 504
12.3.1 Windows Kerberos security support . . . . . 504
12.3.2 UNICODE . . . . . 504
12.3.3 Enhanced management of constraints . . . . . 504
12.3.4 Important notes . . . . . 505
12.4 Fallback considerations . . . . . 506
12.4.1 Unload utility . . . . . 507
12.4.2 Utility lists and dynamic allocation . . . . . 507
12.4.3 Consistent restart . . . . . 507
12.4.4 Online reorg without rename . . . . . 507
12.4.5 Data sharing enhancements . . . . . 508
12.4.6 Enhanced management of constraints . . . . . 508
12.5 Data sharing coexistence . . . . . 509
12.6 Evolution of the DB2 catalog . . . . . 512

**Part 9. Appendices** . . . . . 515

**Appendix A. Updatable DB2 subsystem parameters** . . . . . 517
A.1 List of changeable parameters . . . . . 517
A.2 Currently active zparms . . . . . 518

**Appendix B. SQL examples** . . . . . 521
B.1 Creating the credit card database . . . . . 521
B.2 Union in views: Create JANUARY2000 view . . . . . 527
B.3 Union in table-spec . . . . . 527
B.4 Union in basic predicates . . . . . 528
B.5 Union in qualified predicates . . . . . 528
B.6 Union in the EXISTS predicates . . . . . 529
B.7 Union in the IN predicates . . . . . 529
B.8 Union in INSERT and UPDATE . . . . . 529
B.9 Optimizing union everywhere queries . . . . . 531

**Appendix C. Using the additional material** . . . . . 533
C.1 Locating the additional material on the Internet . . . . . 533
C.2 Using the Web material . . . . . 533
C.2.1 System requirements for downloading the Web material . . . . . 533
C.2.2 How to use the Web material . . . . . 533

**Appendix D. Special notices** . . . . . 535

**Appendix E. Related publications** . . . . . 539
E.1 IBM Redbooks . . . . . 539
E.2 IBM Redbooks collections . . . . . 540
E.3 Other resources . . . . . 540
E.4 Referenced Web sites . . . . . 541

**How to get IBM Redbooks** . . . . . 543
IBM Redbooks fax order form . . . . . 544

**Glossary** . . . . . 545

**Abbreviations and acronyms** . . . . . 559

**Index** . . . . . 561

**IBM Redbooks review** . . . . . 567

# Preface

IBM DATABASE 2 Universal Database Server for OS/390 Version 7 and z/OS (DB2 UDB for OS/390 and z/OS Version 7, or just DB2 V7 throughout this IBM Redbook) is the eleventh release of DB2 for MVS and brings to this platform the data support, application development and query functionality enhancements for e-business while integrating warehouse management and building upon the traditional capabilities of availability and performance. The DB2 V7 environment is available for the OS/390 and z/OS platforms, either for new installations of DB2, or for migrations from *both* DB2 for OS/390 Version 6 and DB2 for OS/390 Version 5 subsystems.

This IBM Redbook, in the format of a presentation guide, describes the enhancements made available with DB2 V7. These enhancements include performance and availability delivered through newly separately packaged and enhanced utilities, dynamic changes to the value of many of the system parameters without stopping DB2, and the new Restart Light option for data sharing environments. Improvements to usability are provided with new and faster tools, the DB2 XML Extender support for the XML data type, scrollable cursors, support for UNICODE encoded data, support for COMMIT and ROLLBACK within a stored procedure, the option to eliminate the DB2 precompile step in program preparation, and the definition of view with the operators UNION or UNION ALL.

DB2 V7 also introduces two optional features: Warehouse Manager and Net search Extender. DB2 Warehouse Manager provides a set of tools for simplifying the design and deployment of a data warehouse within your S/390; Net Search Extender delivers high-speed full text search technology.

DB2 for OS/390 Version 6 has already introduced the support for data spaces, positioning itself for the exploitation of new technologies. Data spaces provide the foundation for DB2 to exploit the increased real storage, larger than the 2-GB limit, now available with the new 64-bit processors of the zSeries.

This redbook will help you understand why migrating to Version 7 of DB2 can be beneficial for your applications and your DB2 subsystems. It will provide sufficient information so you can start prioritizing the implementation of the new functions and evaluating their applicability in your DB2 environments.

The foils included in this redbook as additional material constitute a deliverable in Freelance format accessible through the Web from the IBM internal ITSO site under Materials Repository.

**Note:** Throughout this redbook, OS/390 is meant to signify both OS/390 and z/OS unless otherwise stated.

© Copyright IBM Corp. 2001

## The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization San Jose Center.

**Maria Sueli Almeida** is a Certified IT Specialist - Systems Enterprise Data, currently working as a Developer on DB2 Client/Server Solutions at the IBM Silicon Valley Lab. During the writing of this redbook she was a DB2 for OS/390 and Distributed Relational Database System (DRDS) Specialist with the International Technical Support Organization, San Jose Center. Before joining the ITSO in 1998, Maria Sueli worked at IBM Brazil assisting customers and IBM technical professionals on DB2, data sharing, database design, performance, and DRDA connectivity.

**Paolo Bruni** is a Certified Consultant IT Architect currently on assignment as Data Management Specialist for DB2 for OS/390 with the International Technical Support Organization, San Jose Center, where he conducts projects on all areas of DB2. Before joining the ITSO in 1998, Paolo worked in IBM Italy as account SE at Banking, Finance and Securities ISU. During his many years with IBM, in development and in the field, Paolo's work has been mostly related to database systems.

**Reto Haemmerli** is an Advisory IT Specialist on Database Systems with IBM Global Services and is based in Switzerland. He has 14 years of experience in information technology, mainly as system programmer in DB2 and IMS environments. His work with customers includes installation, maintenance and support, problem diagnosis and resolution, and consultancy services.

**Walter Huth** is currently a DB2 for OS/390 and DRDA Instructor and Course Developer with IBM Learning Services located in Germany. Previously he was database administrator for IBM internal applications. Before joining IBM Germany 14 years ago, Walter was a systems engineer with Taylorix-Tymshare, Germany, where he provided support for two years on designing and using a multi-dimensional database accessible in timesharing mode.

**Michael Parbs** is a DB2 Systems Programmer with IBM Australia and is located in Canberra. He has over 10 years experience with DB2, primarily on the OS/390 platform. Before joining IBM Global Services Australia in 1996 he worked for 11 years for the Health Insurance Commission, where he started using DB2. Michael's main area of interest is data sharing, but his skills include database administration and DB2 connectivity across several platforms.

**Paul Tainsh** is a Database Administrator with IBM Global Services in Sydney, Australia and works mainly at customer sites. He has 12 years experience with DB2 OS/390, having worked in application design and development, DB2 training and database administration, mainly within the financial sector. His main interests lie in how DB2 can be used to provide business solutions across multiple platforms and its interaction with net technologies.

Thanks to the following people for their invaluable contributions to this project:

Emma Jacobs
Yvonne Lyon
Claudia Traver
IBM International Technical Support Organization, San Jose Center

Peggy Abelite
Karelle Cornwell
Roy Cornford
Dan Courter
Chris Crone
Cathy Drummond
Keith Howell
Anne Jackson
Jeff Josten
Regina Liu
Susan Malaika
Roger Miller
Phyllis Marlino
Mary Paquet
San Phoenix
Jim Pickel
Jim Pizor
Jim Ruddy
Kalpana Shyam
Tom Toomire
Yumi Tsuji
Cathy Zagelow
IBM Silicon Valley Laboratory

Sarah Ellis
Mike Bracey
IBM UK, PISC, Hursley

Vasilis Karras
IBM International Technical Support Organization, Poughkeepsie Center

## Comments welcome

**Your comments are important to us!**

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 567 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

# Part 1. Contents and packaging

© Copyright IBM Corp. 2001

# Chapter 1. DB2 V7 contents and packaging

**DB2 UDB for OS/390 and z/OS Version 7**

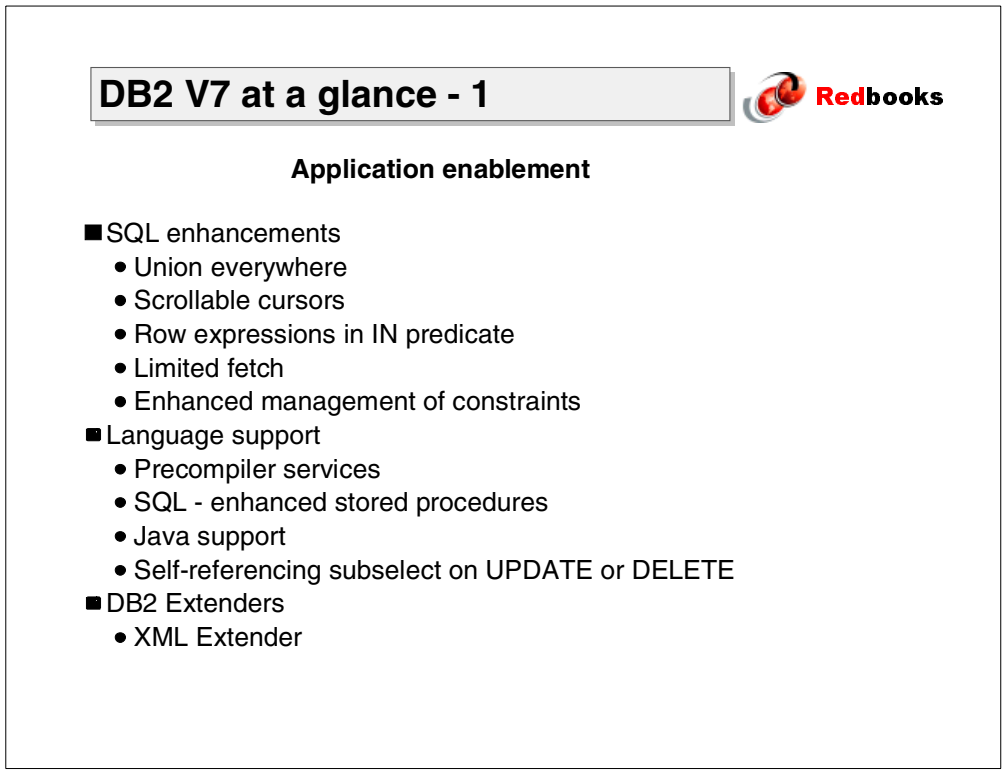
Redbooks

- Areas of enhancements
  - Application enablement
  - Utilities
  - Network computing
  - Performance and availability
  - Data sharing
  - Features and tools
  - Installation and migration
- Product packaging
  - New features
  - Utilities and tools

With DB2 V7, the DB2 Family delivers more scalability and availability for your e-business and business intelligence applications. Using the powerful environment provided by S/390 and OS/390, and the new zSeries and z/OS, you can leverage your existing applications while developing and expanding your electronic commerce for the future.

In this chapter we first briefly introduce the main enhancements introduced by DB2 V7, associating them to the areas listed in the foil, then we describe the features and tools that are currently shipped with the base product.

© Copyright IBM Corp. 2001




## 1.1 DB2 V7 at a glance

DB2 V7 delivers several enhancements for the usability, scalability, and availability of your e-business and business intelligence applications.

### 1.1.1 Application enablement

Greater flexibility and family compatibility comes from several SQL enhancements.

***Union everywhere***
This enhancement satisfies an old important requirement. It provides the ability to define a view based upon the UNION of subselects: users can reference the view as if it were a single table while keeping the amount of data manageable at the table level.

***Scrollable cursors***
Scrollable cursors give your application logic ease of movement through the result table using simple SQL and program logic. This frees your application from the need to cache the resultant data or to reinvoke the query in order to reposition within the resultant data.

Support for scrollable cursors enables applications to use a powerful new set of SQL to fetch data using a cursor at random and in forward and backward direction. The syntax can replace cumbersome logic techniques and coding techniques and improve performance. Scrollable cursors are especially useful for screen-based applications. You can specify that the data in the result table remain static or do the data updates dynamically. You can specify that the data in

the result table remain insensitive or sensitive to concurrent changes in the database. You can also update the database if you choose to be sensitive to changes. For example, an accounting application can require that data remain constant, while an airline reservation system application must display the latest flight availability information.

***Row expression in IN predicate***
This function allows the ability to define a subselect whose resultant data is multiple columns, along with the ability to use those multiple columns as a single unit in comparison operations of the outer level select. It is useful for multiple column key comparisons and needed for some key applications.

***Limited fetch***
It consists of FETCH FIRST 'n' ROWS SQL clause and fast implicit close. A new SQL clause and a fast close improve performance of applications in a distributed environment. You can use the FETCH FIRST 'n' ROWS clause to limit the number of rows that are prefetched and returned by the SELECT statement. You can specify the FETCH FIRST ROW ONLY clause on a SELECT INTO statement when the query can return more than one row in the answer set. This tells DB2 that you are only interested in the first row, and you want DB2 to ignore the other rows.

***Enhanced management of constraints***
You can specify a constraint name at the time you create primary or unique keys. DB2 introduces the restriction of dropping an index required to enforce a constraint.

***Precompiler services***
With DB2 V7 you can take advantage of precompiler services to perform the tasks currently executed by the DB2 precompiler. This API can be called by the COBOL compiler. By using this option, you can eliminate the DB2 precompile step in program preparation and take advantage of language capabilities that had been restricted by the precompiler. Use of the host language compiler enhances DB2 family compatibility, making it easier to import applications from other database management systems and from other operating environments.

***SQL — enhanced stored procedures***
Stored procedures, introduced with DB2 V5, have increased program flexibility and portability among relational databases. DB2 V7 accepts COMMIT and ROLLBACK statements issued from within a stored procedure. This enhancement will prove especially useful for applications in which the stored procedure has been invoked from a remote client.

***Java support***
DB2 V7 implements support for the JDBC 2.0 standard and, in addition, support for userid/password usage on SQL CONNECT via URL and the JDBC Driver execution under IMS.

DB2 V7 also allows you to implement Java stored procedures as both compiled Java using the OS/390 High Performance Java Compiler (HPJ) and interpreted Java executing in a Java Virtual Machine (JVM), as well as support for user-defined external (non SQL) functions written in Java.

### *Self-referencing subselect on UPDATE or DELETE*

Now, you can use a subselect to determine the values used in the SET clause of an UPDATE statement. Also, you can have a self-referencing subselect. In previous releases of DB2, in a searched UPDATE and DELETE statement, the WHERE clause cannot refer to the object being modified by the statement. V7 removes the restriction for the searched UPDATE and DELETE statements, but not for the positioned UPDATE and DELETE statements. The search condition in the WHERE clause can include a subquery in which the base object of both the subquery and the searched UPDATE or DELETE statement are the same. The following code sample is for an application which gives a 10% increase to each employee whose salary is below the average salary for the job code:

```
UPDATE EMP X SET SALARY = SALARY * 1.10
WHERE SALARY < (SELECT AVG(SALARY) FROM EMP Y
WHERE X.JOBCODE = Y.JOBCODE);
```

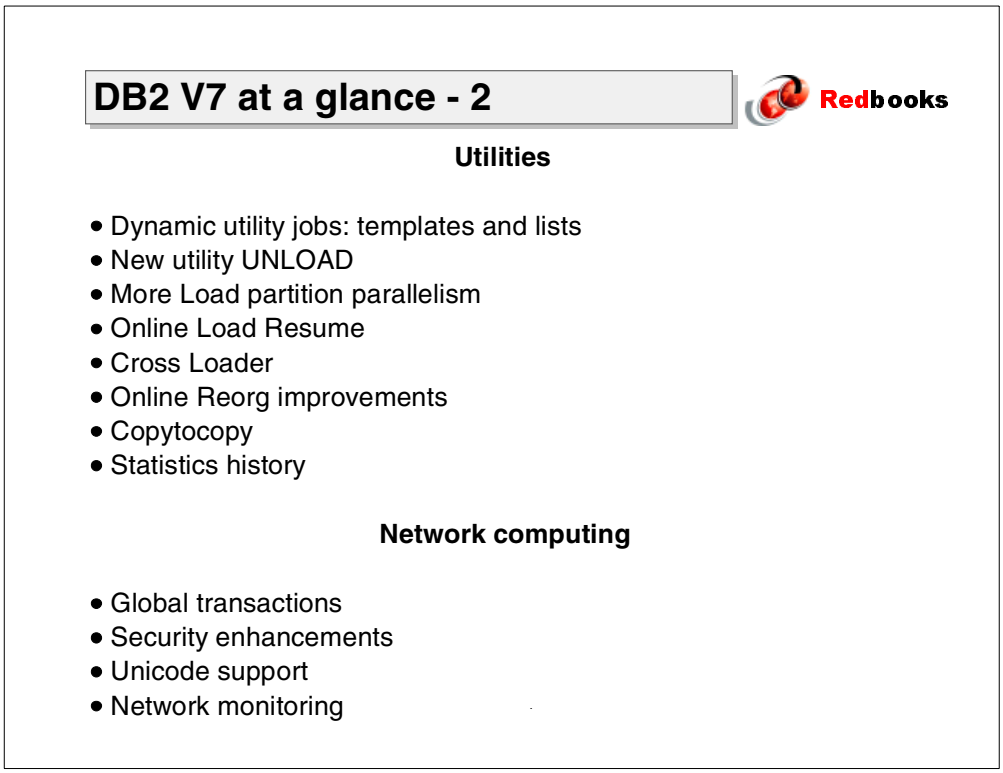
The base object for both the UPDATE statement and the WHERE clause is the EMP table. DB2 evaluates the complete subquery before performing the update.

### *Allow DBADM to create views and aliases for others*

DBADM authority is expanded to include creating aliases and views for others. This function is activated at installation time. See 11.6, “DBADM authority for create view” on page 483 for more details.

### *XML Extender*

DB2 V7 provides more flexibility for your enterprise applications and makes it easier to call applications. The family adds DB2 XML Extender support for the XML data type. This extender allows you to store an XML object either in an XML column for the entire document, or in several columns containing the fields from the document structure.




### 1.1.2 Utilities

***Dynamic utility jobs***
With DB2 V7, database administrators can submit utilities jobs more quickly and easily. Now you can:

- Dynamically create object lists from a pattern-matching expression
- Dynamically allocate the data sets required to process those objects

Using a LISTDEF facility, you can standardize object lists and the utility control statements that refer to them. Standardization reduces the need to customize and change utility job streams over time. The use of TEMPLATE utility control statements simplifies your JCL by eliminating most data set DD cards. Now you can provide data set templates and the DB2 product dynamically allocates the data sets that are required based on your allocation information. Database administrators require less time to maintain utilities jobs, and database administrators who are new to DB2 will learn to perform these tasks more quickly.

***UNLOAD***
With DB2 V7 you can take advantage of a new utility, UNLOAD, which provides faster data unloading than was available with the DSNTIAUL program. The UNLOAD utility combines the unload functions of REORG UNLOAD EXTERNAL with the ability to unload data from an image copy.

***More parallelism with LOAD with multiple inputs***
Using DB2 V7, you can easily load large amounts of data into partitioned table spaces for use in data warehouse or business intelligence applications. Parallel load with multiple inputs runs in a single step, rather than in different jobs. The utility loads each partition from a separate data set so one job can load partitions

in parallel. Parallel LOAD reduces the elapsed time for loading the data when compared to loading the same data with a single job in earlier releases. Using load parallelism is much easier than creating multiple LOAD jobs for individual parts.

The SORTKEYS keyword enables the parallel index build of indexes. Each load task takes input from a sequential data set and loads the data into a corresponding partition. The utility then extracts index keys and passes them in parallel to the sort task that is responsible for sorting the keys for that index. If there is too much data to perform the sort in memory, the sort product writes the keys to the sort work data sets. The sort tasks pass the sorted keys to their corresponding build task, each of which builds one index. If the utility encounters errors during the load, DB2 writes error and error mapping information to the error and map data sets.

***Online LOAD RESUME***
Earlier releases of DB2 restrict access to data during LOAD processing. DB2 V7 gives you the choice of allowing user read and write access to the data during LOAD processing, so you can load data concurrently with user transactions.

***Cross Loader***
The enhancement allows EXEC SQL statements results as input to the Load utility. Both local DB2s and remote DRDA compliant databases can be accessed.

***Online REORG enhancements***
Online REORG makes your data more available. Online REORG enhancements improve the availability of data in two ways: Online REORG no longer renames data sets, greatly reducing the time that data is unavailable during the SWITCH phase. You specify a new keyword, FASTSWITCH, which keeps the data set name unchanged and updates the catalog to reference the newly reorganized data set. The time savings can be quite significant when DB2 is reorganizing hundreds of table spaces and index objects. Also, additional parallel processing improves the elapsed time of the BUILD2 phase of `REORG SHRLEVEL(CHANGE)` or `SHRLEVEL(REFERENCE)`.

***CopyToCopy***
This feature provides the capability to produce additional image copies recorded in the DB2 catalog.

***Statistics history***
As the volume and diversity of your business activities grow, you require changes to the physical design of DB2 objects. V7 of DB2 collects statistic history to track your changes. With historical statistics available, DB2 can predict the future space requirements for table spaces and indexes more accurately and run utilities to improve performance. DB2 Visual Explain utilizes statistics history for comparison with new variations that you enter so you can improve your access paths. DB2 stores statistics in catalog history tables. To maintain optimum performance of processes that access the tables, use the MODIFY STATISTICS utility. The utility can delete records that were written to the catalog history tables before a specific date or that are recorded as a specific age.

### 1.1.3 Network computing

Network computing includes the following:

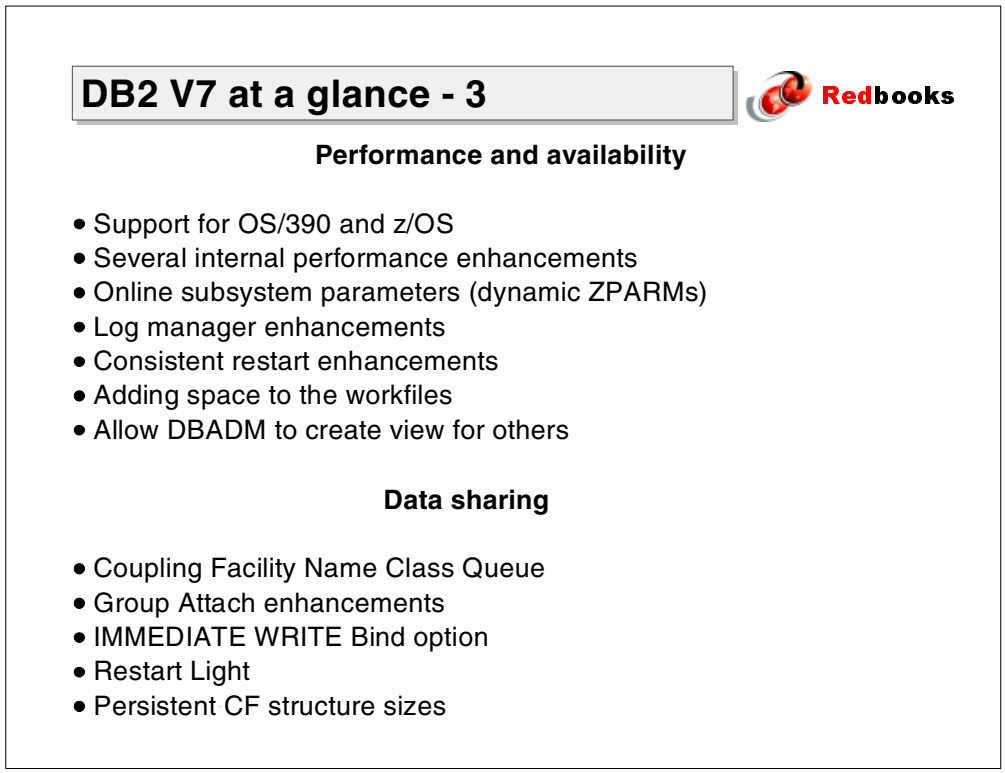
***Global transactions***
Privileged application can use multiple DB2 agents or threads to perform processing that requires coordinated commit processing across all the threads. DB2 V7 (and V6 RML), via a transaction processor, treats these separate DB2 threads as a single “global transaction” and commits all or none.

***Security enhancements***
You can more easily manage your workstation clients who seek access to data and services from heterogeneous environments with DB2 support for Windows Kerberos authentication, which:

- Eliminates the flow of unencrypted user IDs and passwords across the network.
- Enables single-logon capability for DRDA clients by using the Kerberos principle name as the global identity for the end user.
- Simplifies security administration by using the Kerberos principle name for connection processing and by automatically mapping the name to the local user ID.
- Uses the Resource Access Control Facility (RACF) product to perform much of the Kerberos configuration. RACF is a familiar environment to administrations of OS/390.
- Eliminates the need to manage authentication in two places, the RACF database, and a separate Kerberos registry.

***UNICODE support***
In the increasingly global world of business and e-commerce, there is a growing need for data arising from geographically disparate users to be stored in a central server. Previous releases of DB2 have offered support for numerous code sets of data in either ASCII or EBCDIC format. However, there was a limitation of only one code set per system. DB2 V7 supports UNICODE encoded data. This new code set is an encoding scheme that is able to represent the characters (code points) of many different geographies and languages.

***Network monitoring***
DB2 V7 introduces reporting server elapsed time at the workstation. Workstations accessing DB2 data can now request that DB2 return the elapsed time of the server, which is used to process a request in reply from the DB2 subsystem. The server elapsed time allows remote clients to quickly determine the amount of time it takes for DB2 to process a request. The server elapsed time does not include any network delay time, which allows workstation clients, in real-time, to determine performance bottlenecks among the client, the network, and DB2.




### 1.1.4 Performance and availability

***Online subsystem parameters***
One of the causes of a planned outage for DB2 is the need to alter one or more of the system parameters, known as ZPARMS. DB2 V7 enables you to change the value of many of these system parameters without stopping DB2.

***Log manager enhancements***
Log manager updates help you with DB2 operations by providing:

- Suspend update activity
- Retry critical log read access
- Time interval system checkpoint frequency
- Long running UR information

***Consistent restart enhancements***
DB2 V7 provides more control of the availability of the user objects associated with the failing or cancelled transaction without restarting DB2. Consistent restart enhancements remove some of the restrictions imposed by the current consistent restart function and add no backout of data and log feature to DB2's -CANCEL THREAD command.

***Adding space to the workfiles***
DB2 V7 allows you to CREATE and DROP workfile table space, without having to STOP the workfile database.

### 1.1.5 Data Sharing

Data Sharing customers can benefit from several new enhancements.

***Coupling Facility Name Class Queues***
DB2 V7 exploits the OS/390 and z/OS support for the Coupling Facility Name Class Queues. This enhancement reduces the performance impact of purging group buffer pool (GBP) entries for GBP-dependent page sets.

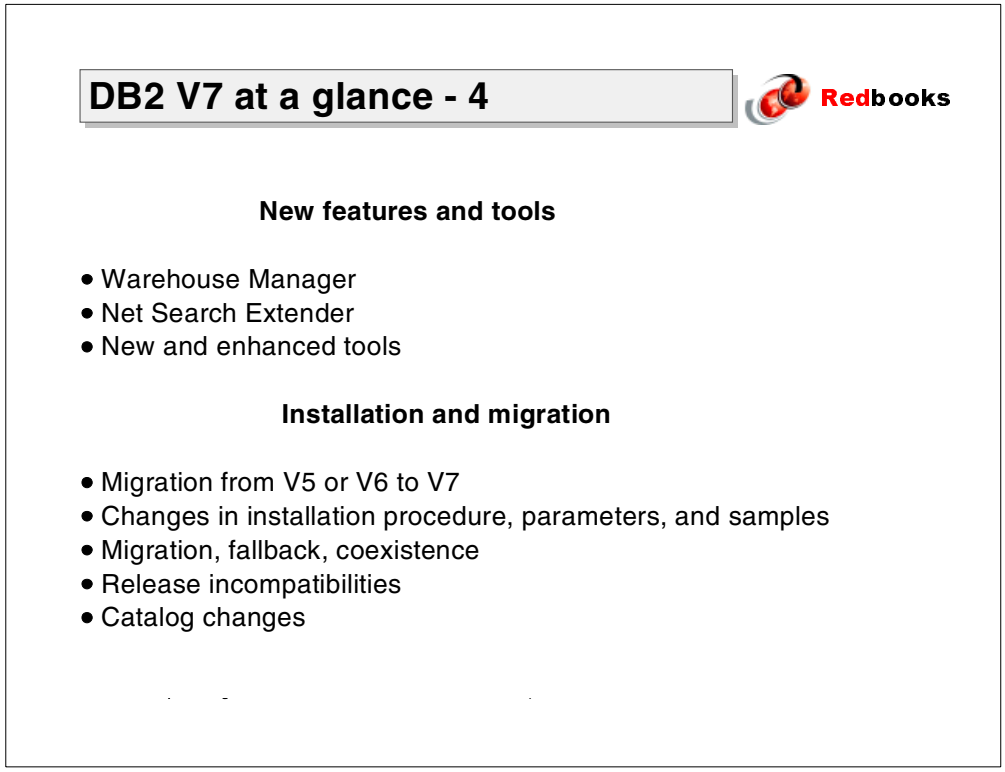
***Group Attach enhancements***
A number of enhancements are made to Group Attach processing in DBV7:

- An application can now connect to a specific DB2 member of a Data Sharing group
- You can now connect to a DB2 Data Sharing group by using the group attach name
- You can specify the group attach name for your DL/I Batch applications.

***Restart Light***
A new feature of the START DB2 command allows you to choose Restart Light for a DB2 member. Restart Light allows a DB2 Data Sharing member to restart with a minimal storage footprint, and then to terminate normally after DB2 frees retained locks. The reduced storage requirement can make a restart for recovery possible on a system that might not have enough resources to start and stop DB2 in normal mode. If you experience a system failure in a Parallel Sysplex, the automated restart in light mode removes retained locks with minimum disruption. Consider using DB2 Restart Light with restart automation software, such as OS/390 Automatic Restart Manager.

***IMMEDWRITE bind option***
A V6 enhancement offers you the choice to immediately write updated group-buffer-pool dependent buffers. In V7, the option is reflected in the DB2 catalog and externalized on the installation panels.

***Persistent structure size changes***
In earlier releases of DB2, any changes you make to structure sizes using the SETXCF START,ALTER command might be lost when you rebuild a structure and recycle DB2. Now you can allow changes in structure size to persist when you rebuild or reallocate a structure.




### 1.1.6 New features and tools

A new feature, DB2 Warehouse Manager, makes it easy to design and deploy a data warehouse on your S/390: you can extract operational data from your DB2 for OS/390 and import it into an S/390 warehouse without transferring your data to an intermediate platform. Prototyping the applications is quicker, you can query and analyze data, and help your users access data and understand information. The new DB2 Warehouse Manager feature gives you a full set of tools for building and using a data warehouse based on DB2 for OS/390.

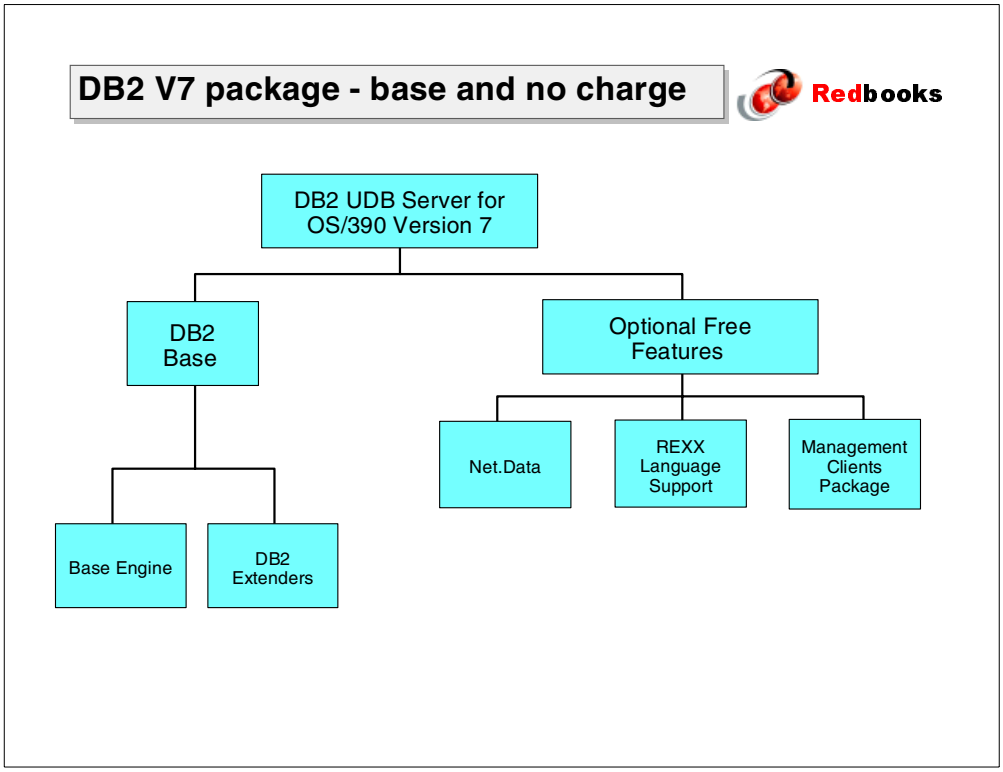
Net Search Extender adds the power of fast full-text retrieval to Net.Data, Java, or DB2 CLI applications. It also offers application programmers a variety of search functions.

With V7 DB2 delivers even more tools. They have been the subject of specific announcements in September 2000 and March 2001, together with several IMS tools. You have the opportunity of a trial period to discover and verify the benefits of these tools, some completely new to the server, others being new versions of tools already available with DB2 V6. Some of the new tools are:

- DB2 Bind Manager, to avoid unnecessary binds
- DB2 Log Analysis Tool, to assist in using the log information
- DB2 SQL Performance Analyzer, to evaluate the cost of a query before it runs
- DB2 Change Accumulation, to consolidate copies and logging offline
- DB2 Recovery Manager, to simplify recovery of data from DB2 and IMS.

### 1.1.7 Installation and migration

Migration with full fallback protection is available when you have either DB2 V5 or DB2 V6. You should ensure that you are fully operational on DB2 V5, or later, before migrating to DB2 V7.




## 1.2 DB2 V7 packaging

DB2 V7 incorporates several features which include tools for data warehouse management, Internet data connectivity, replication, database management and tuning, installation, and capacity planning.

These features and tools work directly with DB2 applications to help you use the full potential of your DB2 system. When ordering the DB2 base product, you can select the free and chargeable features to be included in the package.

You must check the product announcement and the program directories for current and correct information on the contents of DB2 V7 package.

### 1.2.1 Base Engine and no-charge features

The DB2 V7 Base Engine is program Number 5675-DB2 and currently consists of the packaging and shipping of:

- Object Code
- Externalized parameter macros
- JCL procedures
- TSO Clists

- Link Edit Control Statements, JCLIN
- Install verification
  - Sample Program Source Statements (Sample problem)
  - Sample Data Base Data
  - Sample JCL
- DB2 Directory / Catalog Data Base
- ISPF Components (Installation Panels, Messages, Skeleton Library, Command Table
- DBRM Library
- Online Help Reader and associated books (it can be instead of BookManager READ/MVS)
- IRLM
- Call Level Interface feature (which includes JDBC and SQLJ)
- DB2I ISPF panels (in the ordered language)
- Utilities

  With DB2 V7 most utilities have been grouped in three separate independent products:

  - Operational utilities
  - Diagnostic and Recovery utilities
  - Utilities Suite (which includes all the utilities of the above two products)

  All the utilities are shipped deactivated with the Base Engine. The corresponding product licences must be obtained to activate the specific utilities functions. However all utilities are always available for execution on DB2 Catalog and Directory.
- DB2 Extenders:
  - Text
  - Image
  - Audio
  - Video
  - XML
- Tivoli Readiness

### 1.2.2 Optional no-charge features

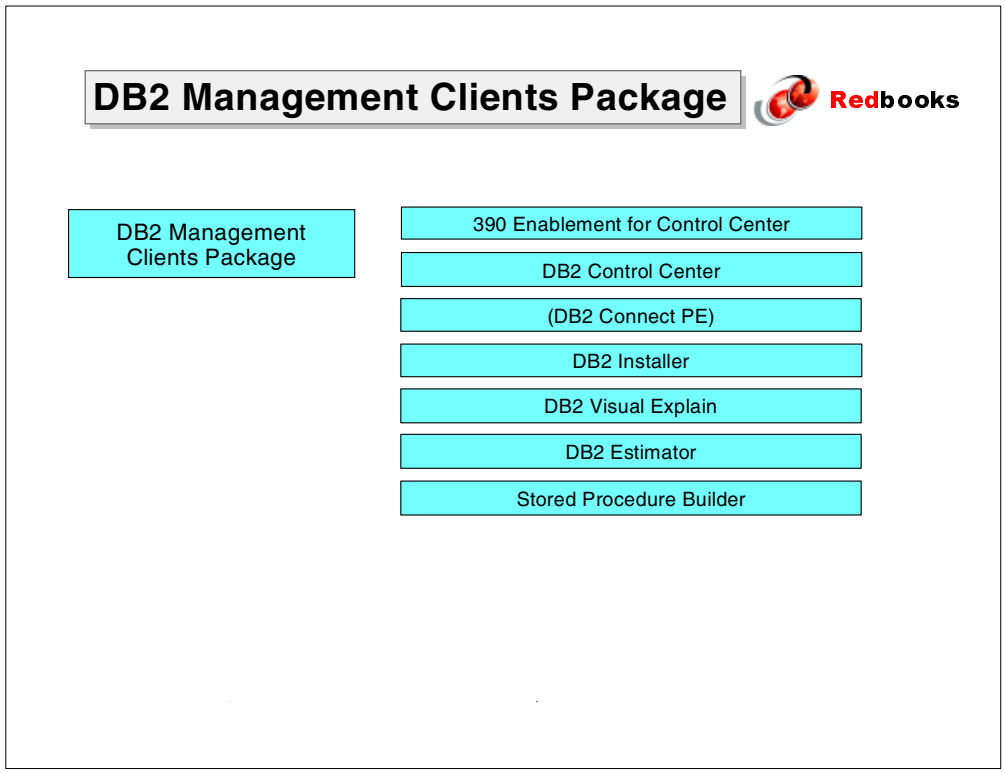
The optional no-charge features are the same as with DB2 V6.

#### 1.2.2.1 Net.Data

Net.Data, a no-charge feature of DB2 V7, takes advantage of the S/390 capabilities as a premier platform for electronic commerce and Internet technology. Net.Data is a full-featured and easy to learn scripting language allowing you to create powerful Web applications. Net.Data can access data from the most prevalent databases in the industry: DB2, Oracle, DRDA-enabled data sources, ODBC data sources, as well as flat file and web registry data. Net.Data Web applications provide continuous application availability, scalability, security, and high performance.

#### 1.2.2.2 REXX language support

REXX language and REXX language stored procedure support are shipped as a part of the DB2 V7 base code. You need to specify the feature and the media when ordering DB2. Documentation is still accessible from the Web. The DB2 installation job DSNTIJRX binds the REXX language support to DB2 and makes it available for use.



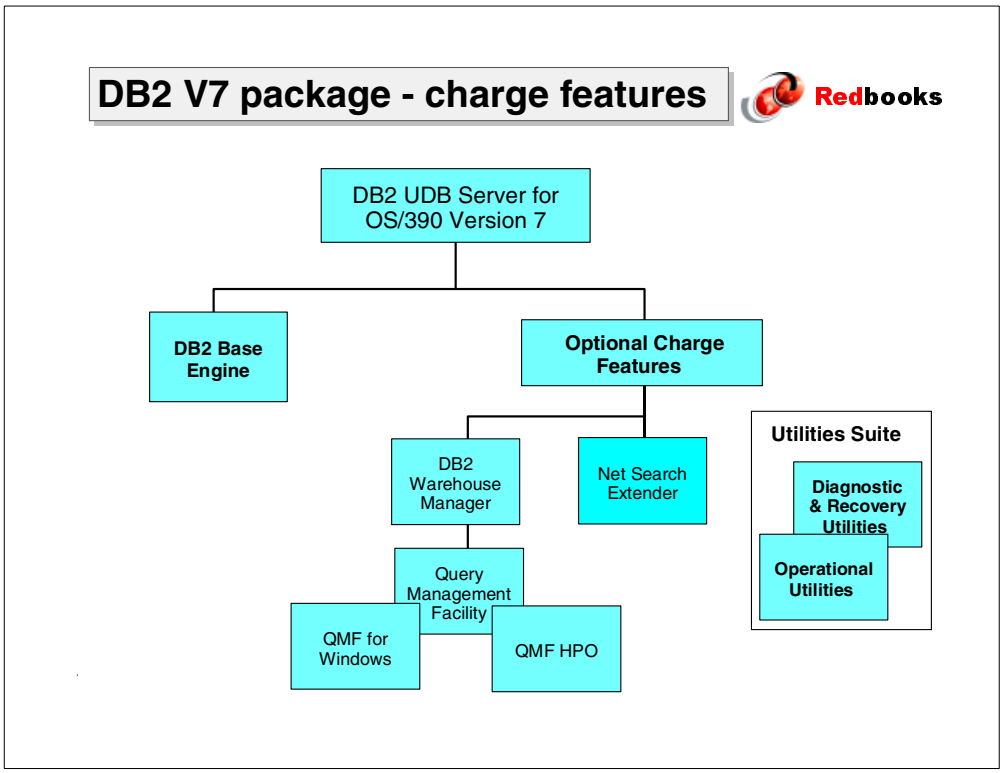

### 1.2.3 DB2 Management Clients Package

The DB2 Management Clients Package is the new name for DB2 V7 for the previous DB2 Management Tools Package. It is a collection of workstation-based client tools that you can use to work with and manage your DB2 for OS/390 and z/OS environments.

It is a separately orderable no-charge feature of DB2 V7 and it currently consists of the packaging and shipping of the following media:

- Tape, containing:
  - 390 Enablement to DB2 Management Clients Package
- Workstation Clients CD/ROM, containing:
  - DB2 Installer
  - Visual Explain
  - DB2 Estimator
  - DB2 Connect Personal Edition (special licence)
  - Control Center
  - Stored Procedure Builder

For functional details on the DB2 Management Clients Package products, please refer to the redbook *DB2 UDB for OS/390 Version 6 Management Tools Package,* SG24-5759.




## 1.3 Charge features

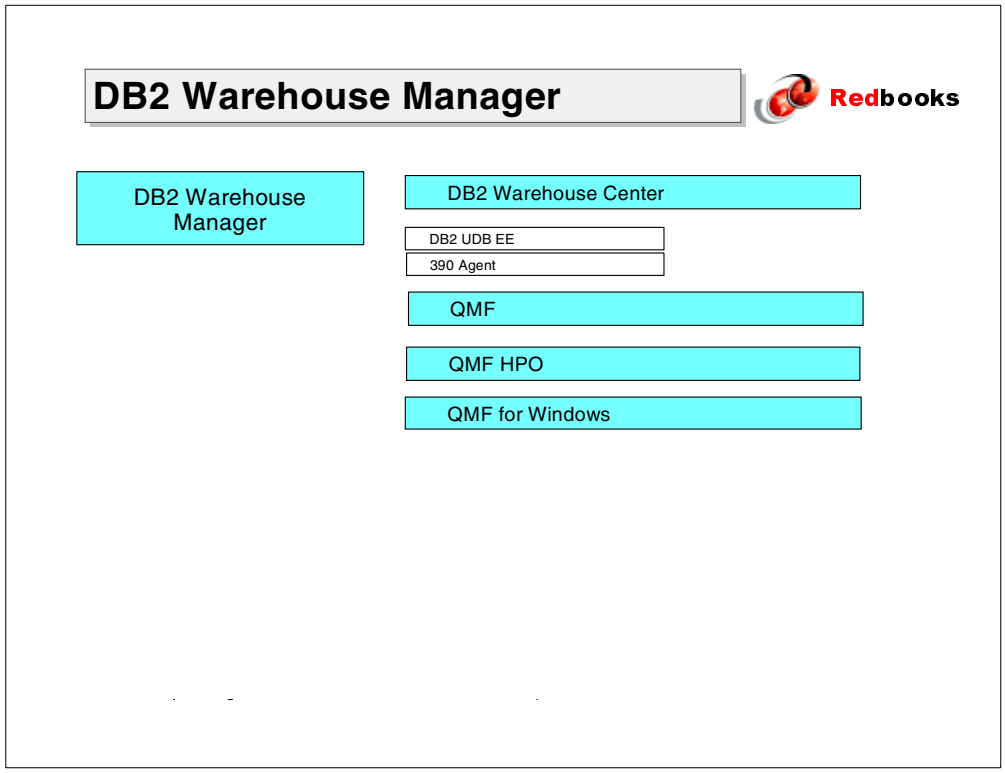
Two new charge features are shipped with DB2 V7:

- The DB2 Warehouse Manager
- The DB2 Net Search Extender

The Query Management Facility product is part of the DB2 Warehouse Manager feature, but it is also still available as a separate feature on its own.

With DB2 V7, the DB2 Utilities have been separated from the base product and they are now offered as separate products licensed under the IBM Program License Agreement (IPLA), and the optional associated agreements for Acquisition of Support. The DB2 Utilities are grouped in these categories:

- DB2 Operational Utilities, program number 5655-E63, which include Copy, Load, Rebuild, Recover, Reorg, Runstats, Stospace, and Unload.
- DB2 Diagnostic and Recovery Utilities, program number 5655-E62, which include Check Data, Check Index, Check LOB, Copy, CopyToCopy, Mergecopy, Modify Recovery, Modify Statistics, Rebuild, and Recover.
- DB2 Utilities Suite, program number 5697-E98, which combines the functions of both DB2 Operational Utilities and DB2 Diagnostic and Recovery Utilities in the most cost effective option.




## 1.4 DB2 Warehouse Manager

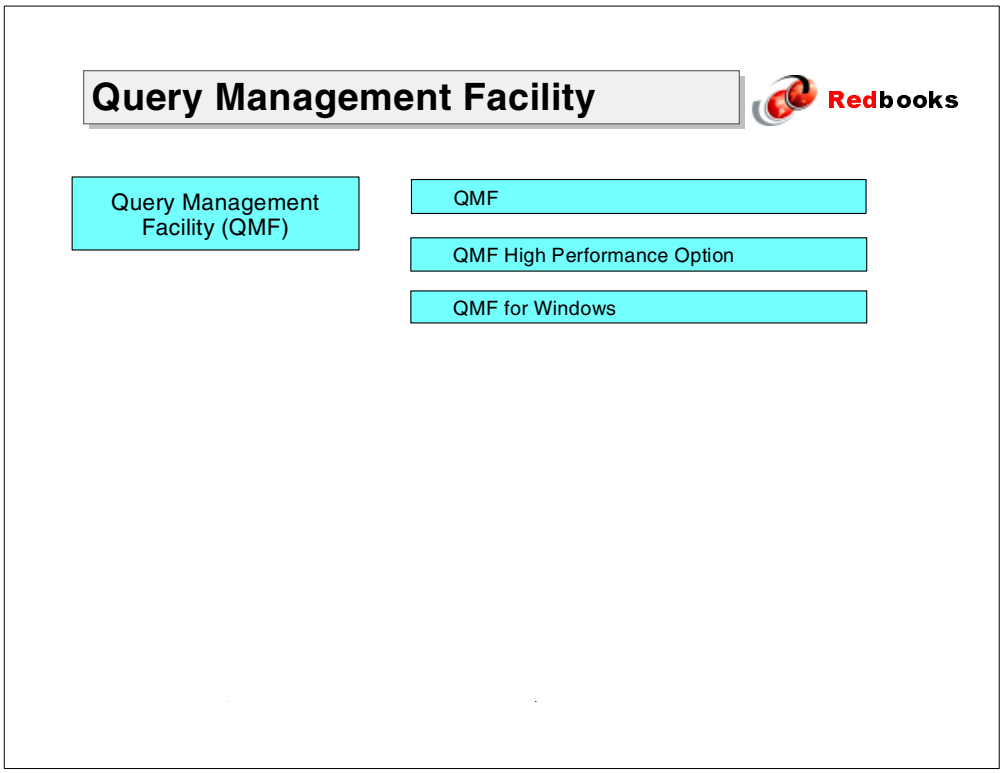
The DB2 Warehouse Manager feature brings together the tools to build, manage, govern, and access DB2 for OS/390-based data warehouses.

DB2 Warehouse Manager is a new, separately orderable, priced feature of DB2 V7.

The DB2 Warehouse Manager currently consists of packaging and shipping of:

- Tapes, containing:
  - DB2 Warehouse Center
  - 390 Warehouse Agent
  - DB2 UDB EEE
  - Query Management Facility for OS/390
  - Query Management Facility High Performance Option
- Workstation Client CD/ROM, containing:
  - Query Management Facility for Windows

  Softcopy Publications for DB2 Warehouse Manager are available on the CD-ROM delivered with the base DB2 order.




### 1.4.1 Query Management Facility

The Query Management Facility (QMF) is the tightly integrated, powerful, and reliable tool for query and reporting within IBM's DB2 family.

QMF for OS/390 is also a separately orderable, priced feature of DB2 V7.

QMF currently consists of the packaging and shipping of:

- Tape, containing:
  - Query Management Facility for OS/390
  - QMF High Performance Option
- Workstation Client CD/ROM, containing:
  - Query Management Facility (QMF) for Windows

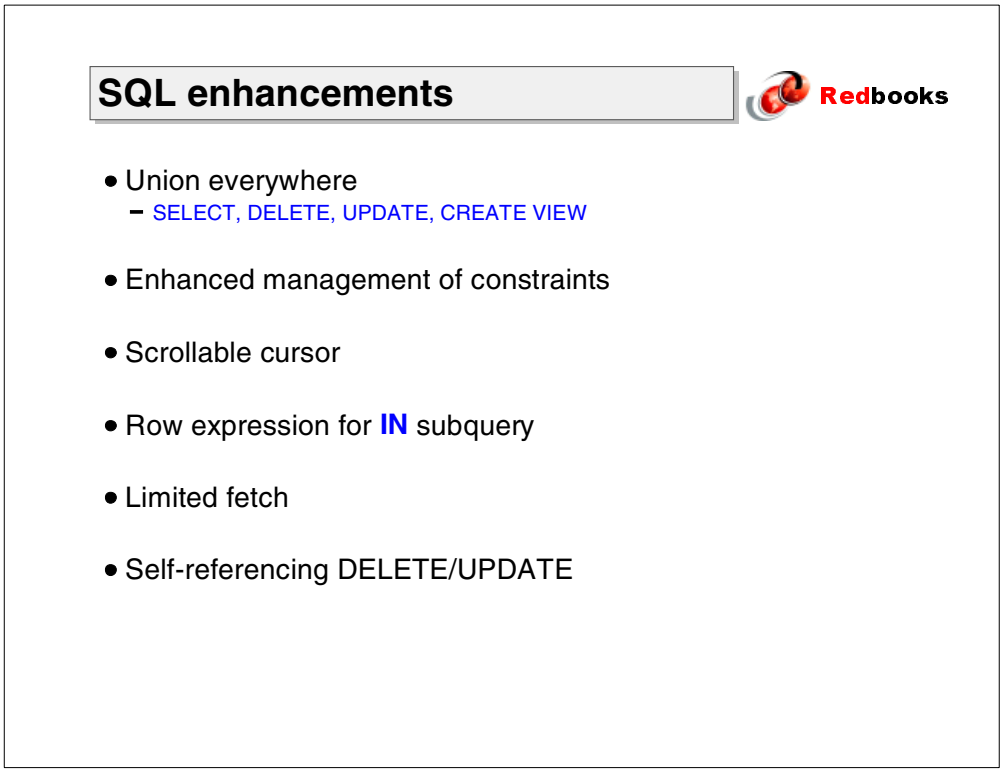
### 1.4.2 Net Search Extender

DB2 Net Search Extender contains a DB2 stored procedure that adds the power of fast full-text retrieval to Net.Data, Java, or DB2 CLI applications. It offers application programmers a variety of search functions, such as fuzzy search, stemming, Boolean operators, and section search.

# Part 2. Application enablement

© Copyright IBM Corp. 2001

# Chapter 2. SQL enhancements




In this chapter we describe the enhancements to SQL in DB2 V7:

- Union everywhere

  SQL has been extended to allow the UNION and UNION ALL to be coded in many more places. The SELECT, CREATE VIEW, DELETE and UPDATE statements have all been expanded.

- Enhanced management of constraints

  This enhancement changes in the way in which primary key, unique key, foreign key and check constraints are managed within DB2. It does this by standardizing the way in which constraints are defined and the syntax used to define them.

- Scrollable cursor

  Cursors can be coded to move to absolute points within a result set or to points both backup and forward of the current cursor position within the result set. This feature also allow the cursor to select whether updates from outside the cursor are visible or not to the cursor.

- Row expression for IN queries

  IN lists support multiple expressions on the left-hand side where a fullselect is specified on the right-hand side of the expression.

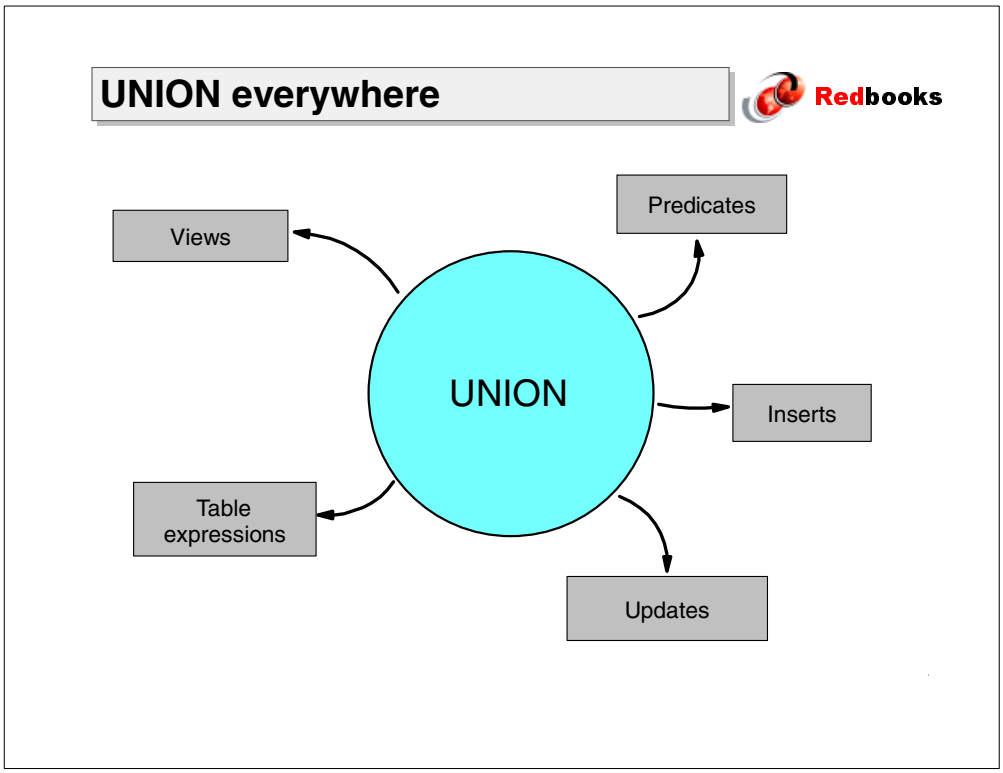
- Limited fetch

  The new clause FETCH FIRST n ROWS ONLY allows the coder to limit to n the number of rows returned by an individual SELECT statement.

© Copyright IBM Corp. 2001

- Self referencing UPDATE/DELETE statements

  Subselects within the UPDATE and DELETE statements can reference the tables that are being targeted for change by the statement.




## 2.1 UNION everywhere

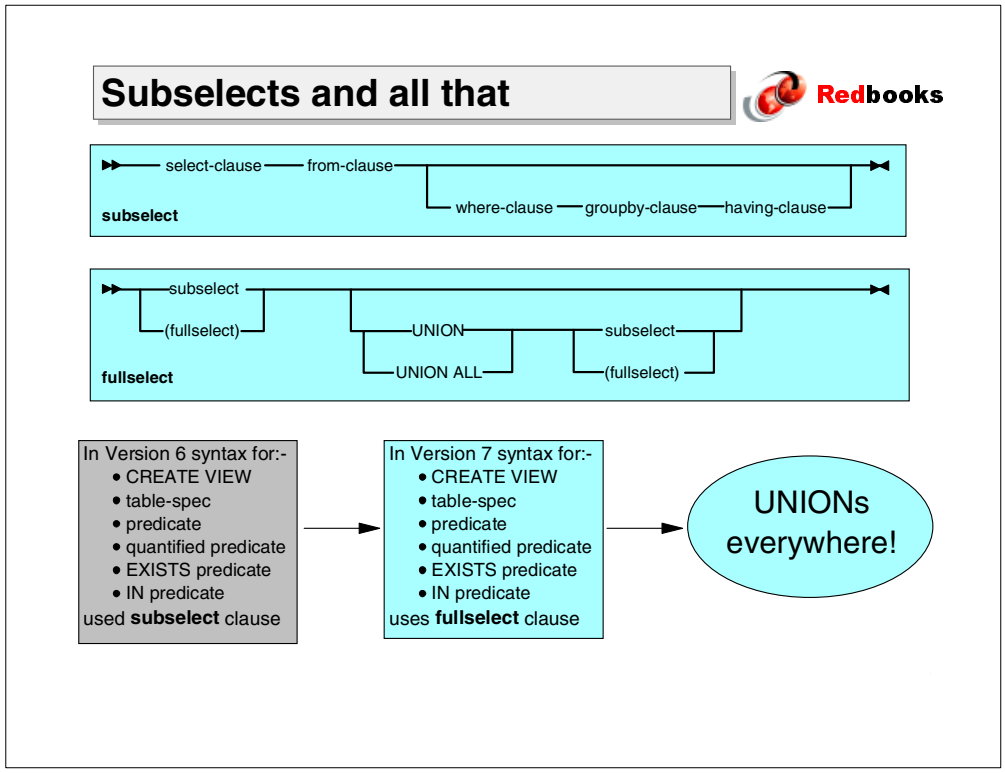
Prior to V7 of DB2, the use of UNIONs was limited to only those places which accepted a fullselect clause. This meant that unions were basically limited to the SELECT statement. Unions could not be used to create a VIEW, in table expressions, in predicates, or in INSERT and UPDATE statements.

V7 has expanded the use of unions to anywhere that a subselect clause was valid in previous versions of DB2.

This enhancement has come about to extend compatibility with other members of the DB2 UDB family and for compliance with the SQL99 standard. It also increases SQL usability by allowing tables that are split into smaller multiple tables to be viewed by the end users as a single table without the users having to understand the nuances of coding a UNION in a SELECT statement. Using this new feature can also help cut down and even eliminate the number of temporary tables required to merge data from disparate tables into a single table.

This section discusses the difference between a subselect clause and a fullselect clause and explains where unions can now be used.

Performance has also been recognized as being integral in the making of this change. Through query rewrite, DB2 will avoid materializing a view with unions as much as possible. To avoid this, it will use query rewrites. Other performance benefits have been included to exploit this enhancement. These are discussed in the performance section that follows.




### 2.1.1 Subselects and all that

The *subselect* syntax in DB2 has always represented the basic SELECT statement. It contains the SELECT, FROM, WHERE, GROUP BY, and HAVING clauses. For example, consider the SQL statement:

```
SELECT DATE, SUM(AMOUNT)
FROM GOLD
WHERE ACCOUNT LIKE 'ZZZ%'
GROUP BY DATE
HAVING COUNT(*) > 3;
```

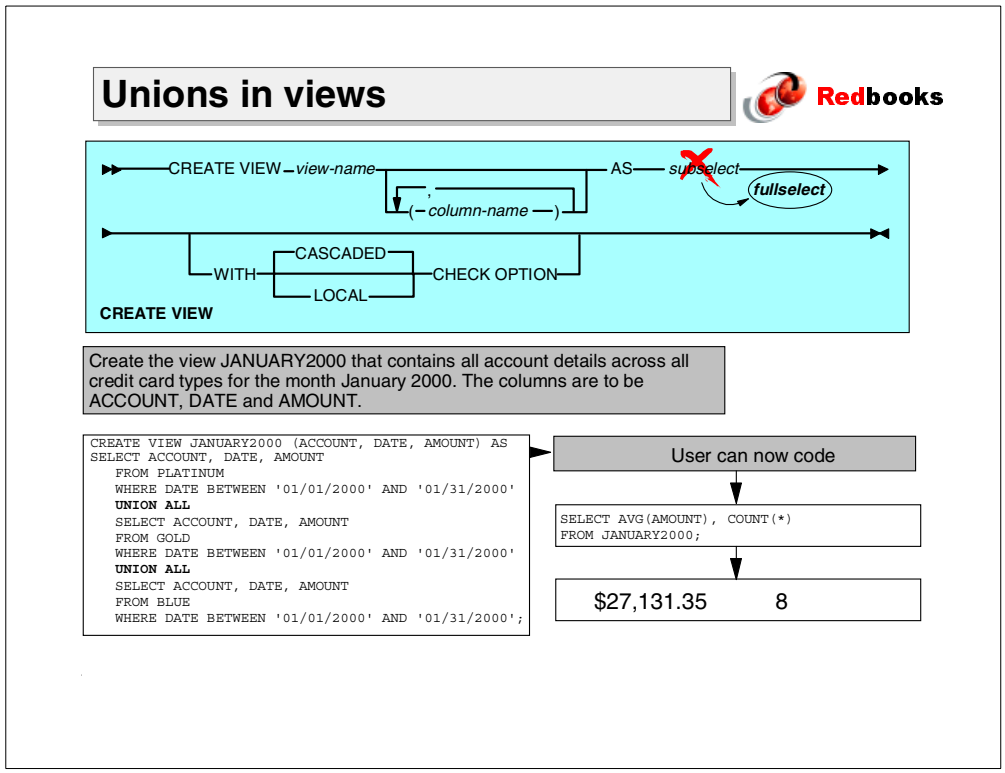
This statement represents all the clauses in the *subselect* syntax. Notice that this syntax does not contain any reference to the UNION keyword. That is because this is contained in the fullselect syntax, which references the *subselect* syntax. An example of an SQL statement with all the elements of a fullselect is:

```
SELECT DATE, SUM(AMOUNT)
FROM PLATINUM
WHERE YEAR(DATE) = 2000
GROUP BY DATE
HAVING COUNT(*) > 3

UNION ALL

SELECT DATE, SUM(AMOUNT)
FROM GOLD
WHERE YEAR(DATE) = 2000
GROUP BY DATE
HAVING COUNT(*) > 3;
```

As seen in the above example, the *fullselect* contains within it two *subselects* with a UNION ALL placed between them.

In all earlier versions of DB2, the use of a *fullselect* was not valid in subqueries, nested table expressions, within the CREATE VIEW statement or the UPDATE and DELETE statements.

V7 now allows a *fullselect* to be used wherever the *subselect* was only allowed in the previous versions of DB2. Therefore, where a *subselect* was only allowed in the UPDATE statement in V6, this statement can now include a *fullselect*, and therefore accepts the use of UNION and UNION ALL.




## 2.1.2 Unions in views

As the updated syntax diagram above shows, the use of the *subselect* clause in the CREATE VIEW syntax has been replaced with the *fullselect* clause. This means that either the UNION or UNION ALL keywords can be included as part of the CREATE VIEW syntax.

It is important to note that following the standard SQL rules for view, if the UNION or UNION ALL keyword is used in the creation of a view, then the view will be read-only. No updates will be allowed for the view.

For our example, we are using a credit card model, in which three cards are available to each customer: PLATINUM, GOLD and BLUE. For accounting purposes, the spending details of each card type are stored in a different table. A customer can only hold one card and if they change card, all details are moved to the current card table. Each account has a single row in the ACCOUNT table. The data in the tables is as follows:

**PLATINUM table**

| ACCOUNT | DATE | AMOUNT |
|---|---|---|
| ABC010 | 03/07/1998 | 5.25 |
| BWH450 | 01/10/2000 | 150.00 |
| ABC010 | 01/15/2000 | 1150.23 |
| BWH450 | 01/15/1999 | 67.00 |
| ABC010 | 02/29/2000 | -1150.23 |

**GOLD table**

| ACCOUNT | DATE | AMOUNT |
|---|---|---|
| ZXY930 | 12/13/1999 | 635.25 |
| MNP230 | 01/11/2000 | 150.00 |
| ZXY930 | 01/15/2000 | 233.57 |
| BMP291 | 02/15/1999 | 31.32 |
| ZXY930 | 01/30/2000 | -233.57 |

**BLUE table**

| ACCOUNT | DATE | AMOUNT |
|---|---|---|
| ULP231 | 03/07/1998 | 16.43 |
| XPM673 | 01/21/2000 | 10927.47 |
| XPM961 | 01/15/2000 | 15253.65 |
| XPM673 | 02/02/2000 | -31.32 |
| XPM961 | 01/30/2000 | -500.00 |

**ACCOUNT table**

| ACCOUNT | ACCOUNT_NAME | CREDIT_LIMIT | TYPE |
|---|---|---|---|
| ABC010 | BIG PETROLEUM | 100000.00 | C |
| BWH450 | HUTH & DAUGHTERS | 2000.00 | C |
| ZXY930 | MIGHTLY BEARS PLC | 50000.00 | C |
| MNP230 | MR P TRENCH | 50.00 | P |
| BMP291 | BASEL FERRARI | 25000.00 | C |
| XPM673 | SCREAM SAVER PTY LTD | 5000.00 | C |
| ULP231 | MS S FLYNN | 500.00 | P |
| XPM961 | MICHAL LINGERIE | 50000.00 | C |

In the first example, the users want to see all the account details across all cards for the month of January 2000. The view to be created is to be called JANUARY2000.

However, in past versions of DB2, there were only two options. A physical table that contained the merged data could be built into which all data from the separate tables was unloaded and loaded into the single table periodically. This would mean that there was the potential that the data was not accurate, as the base tables were not used as a basis for the user's queries.

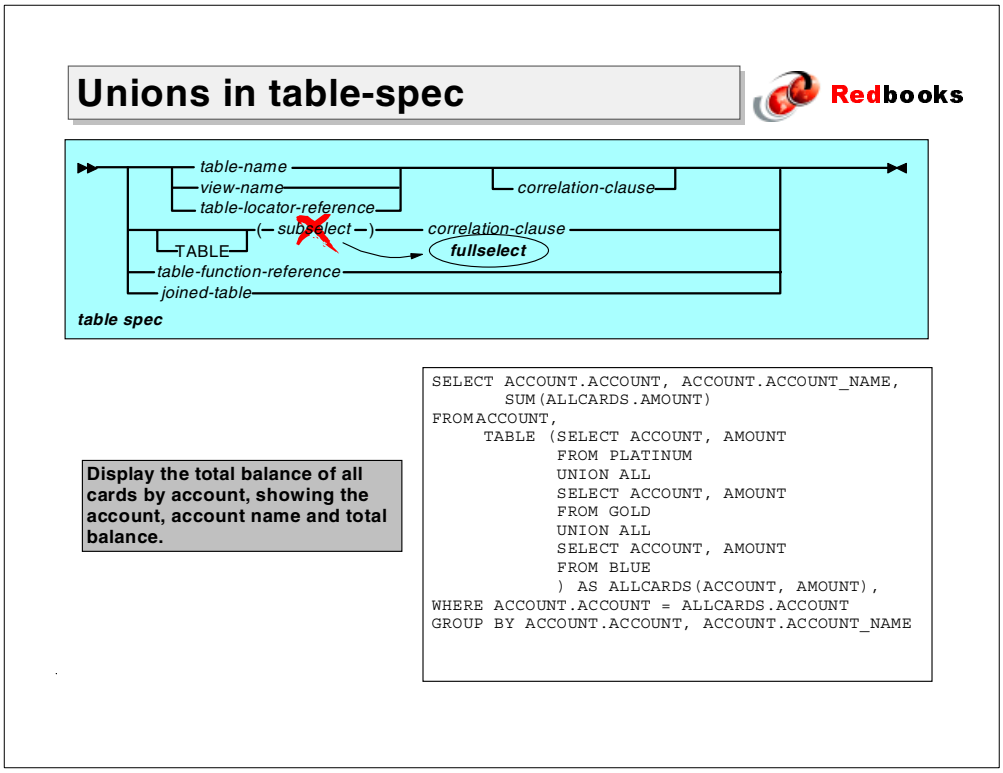
The second option was for the user to know about unions and code the statement themselves every time. The problem with this is that the union is a complex construct and would stretch average user's knowledge of DB2. Also, the user would have to code the statement again every time. Another issue would be that functions could not range across the whole data. The user would have problems in getting values such as averages, counts, and totals for all the tables. This could be done by creating a temporary table for output from the SELECT statement containing the union and then running another select with the function calls against the temporary table. This would be an even more complex problem for the average end user than writing a SELECT with a union in it.

V7 provides a simple answer for this problem. Data from the base tables can be merged dynamically by creating a view using unions as in the example above. Once coded, the user only has to refer to a single table to review data across the three tables and, even more significantly, can now use the full suit of functions, such as COUNT, AVERAGE, and SUM across all these tables data. For example, to find the average amount spent on all cards and the number of transactions across all cards made in the first quarter of 2000, this would be coded as:

```
SELECT SUM(AMOUNT), COUNT(*)
FROM JANUARY2000;
```

This returns the sum of $27,131.35 and a count of 8.

As can be seen, the addition of unions in a CREATE VIEW statement simplifies the merging of data from tables for end user queries and allows the use of the full suite of DB2 functions against the data without the need of temporary tables or complex SQL.



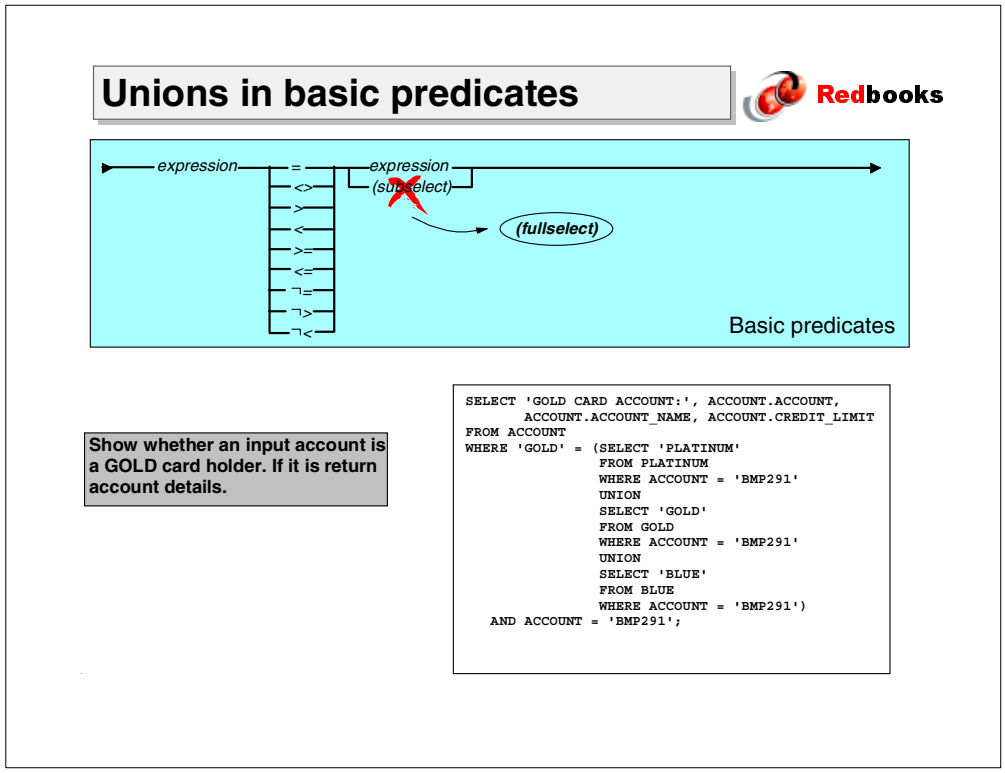

### 2.1.3 Unions in table-spec

This enhancement will be of benefit to the more sophisticated end user and SQL coder. Once again, this enhancement is applied by changing the syntax of the *table-spec* syntax. The *subselect* clause is replaced by the *fullselect* clause.

As the examples above show, this enhancement allows the use of functions across similar data which is stored in multiple tables. In this example, the data that has been merged is grouped. Older versions of DB2 would have required a temporary table to be created with a separate SQL statement to apply the functions. Now this can be achieved with a single SQL statement.

The above sample returns the following result:

```
ACCOUNT  ACCOUNT_NAME                              BALANCE
---------+---------+---------+---------+---------+--------
ABC010   BIG PETROLEUM                               5.25
BMP291   BASEL FERRARI                              31.32
BWH450   HUTH & DAUGHTERS                          217.00
MNP230   MR P TENCH                                150.00
ULP231   MS S FLYNN                                 16.43
XPM673   SCREAM SAVER PTY LTD                    10896.15
XPM961   MICHAL LINGERIE                         14753.65
ZXY930   MIGHTY BEARS PLC                          635.25
```




## 2.1.4 Unions in basic predicates

With union everywhere, the syntax of the predicate clauses have been updated wherever a *subselect* clause occurs. These clauses have been replaced with the *fullselect* clause. This means that unions can now be within the basic, quantified, EXISTS and IN predicate clauses. This gives the ability to merge data from separate tables to be used to compare to single column values within an SQL statement.

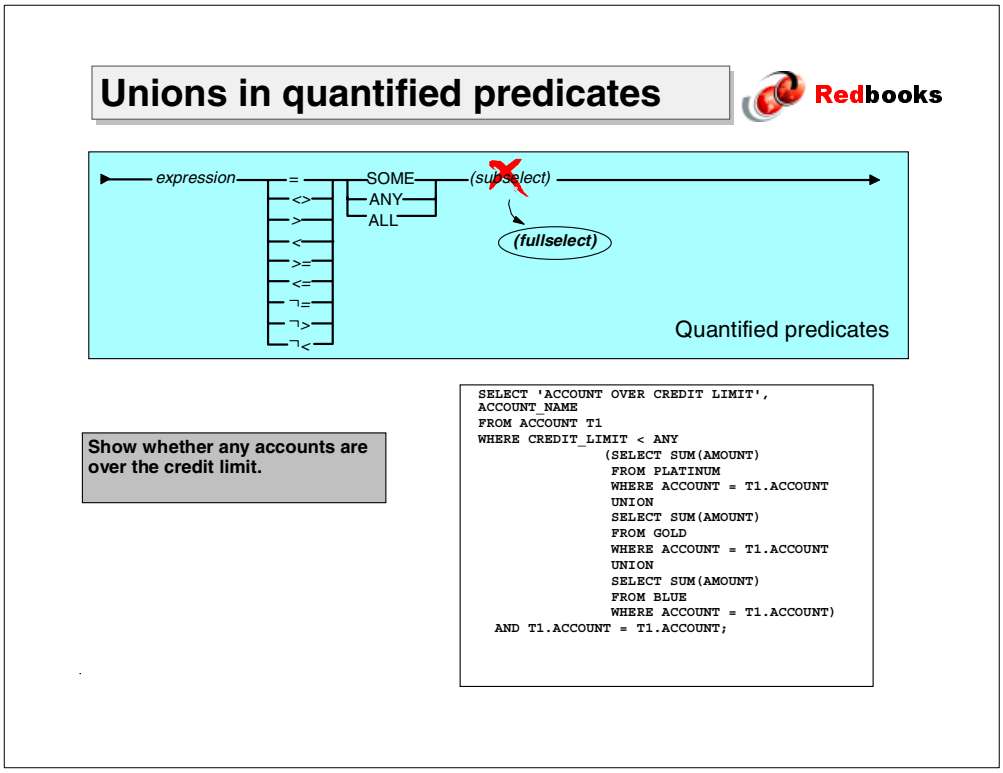
The SQL statement has been written to find out whether an input account is a GOLD card account. For example, if the account BMP291 is entered, then the result is:

```
                    ACCOUNT  ACCOUNT_NAME            CREDIT_LIMIT
---------+---------+---------+---------+---------+---------+----
GOLD CARD ACCOUNT:  BMP291   BASEL FERRARI                25000.00
```

If, however, a non-GOLD card account is entered, then no row is returned.

This example is reliant on account details appearing in only one card type table. If rows for this account appear in more than one table, then, as only one row can be returned as result of a subquery used in a basic predicate, the following error will be issued:

```
DSNT408I SQLCODE = -811, ERROR:  THE RESULT OF AN EMBEDDED SELECT STATEMENT OR
A SUBSELECT IN THE SET CLAUSE OF AN UPDATE STATEMENT IS A TABLE OF
MORE THAN ONE ROW, OR THE RESULT OF A SUBQUERY OF A BASIC PREDICATE IS
MORE THAN ONE VALUE
DSNT418I SQLSTATE   = 21000 SQLSTATE RETURN CODE
```



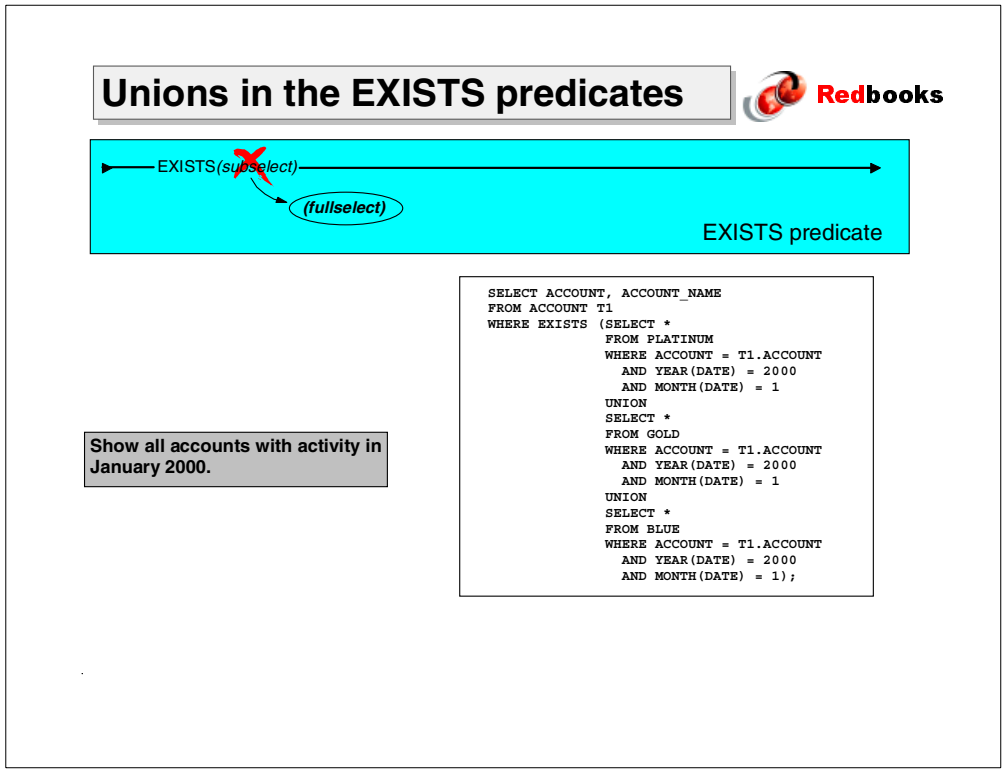

### 2.1.5 Unions in quantified predicates

This example shows where a value is being compared against a result set. Here we are trying to see whether any accounts are over the credit limit. A row with account details is produced if any such accounts are found.

In this instance, at least two tables will return a null value. The UNION keyword cuts out the duplicates, and therefore, only one null value is returned in the result set. If an account is in one of the three tables, then a value is returned. The ANY keyword is used as opposed to the ALL keyword, as the result set will always contain a value and a null for each account. If the credit limit in the ACCOUNT table is less than the returned value, then the account details are displayed.

The result of the example query is:

```
                              ACCOUNT_NAME
---------+---------+---------+---------+---------+---------+-------
ACCOUNT OVER CREDIT LIMIT  MR P TENCH
ACCOUNT OVER CREDIT LIMIT   SCREAM SAVER PTY LTD
```

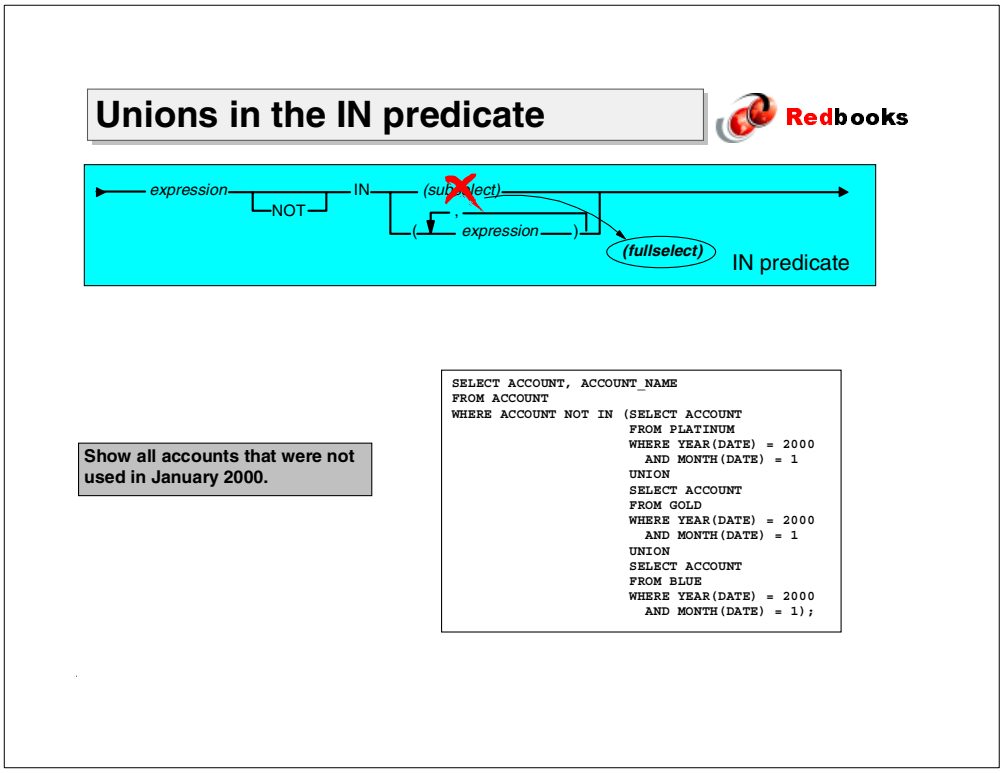



### 2.1.6 Unions in the EXISTS predicate

This example shows the use of the UNION in the EXISTS predicate. Here we are looking for all accounts that were active in January 2000. The UNION allows the query to access all data across all three tables in a single predicate. In the past, this could be achieved by OR'ing the EXISTS predicate with each table individually.

The result of the above query is:

```
ACCOUNT  ACCOUNT_NAME
---------+---------+---------+---------+---------+---------+-----
ABC010   BIG PETROLEUM
BWH450   HEFT & DAUGHTER
ZXY930   MIGHTY BEARS PLC
MNP230   MR P TENCH
XPM673   SCREAM SAVER PTY LTD
XPM961   MICHAL LINGERIE
```

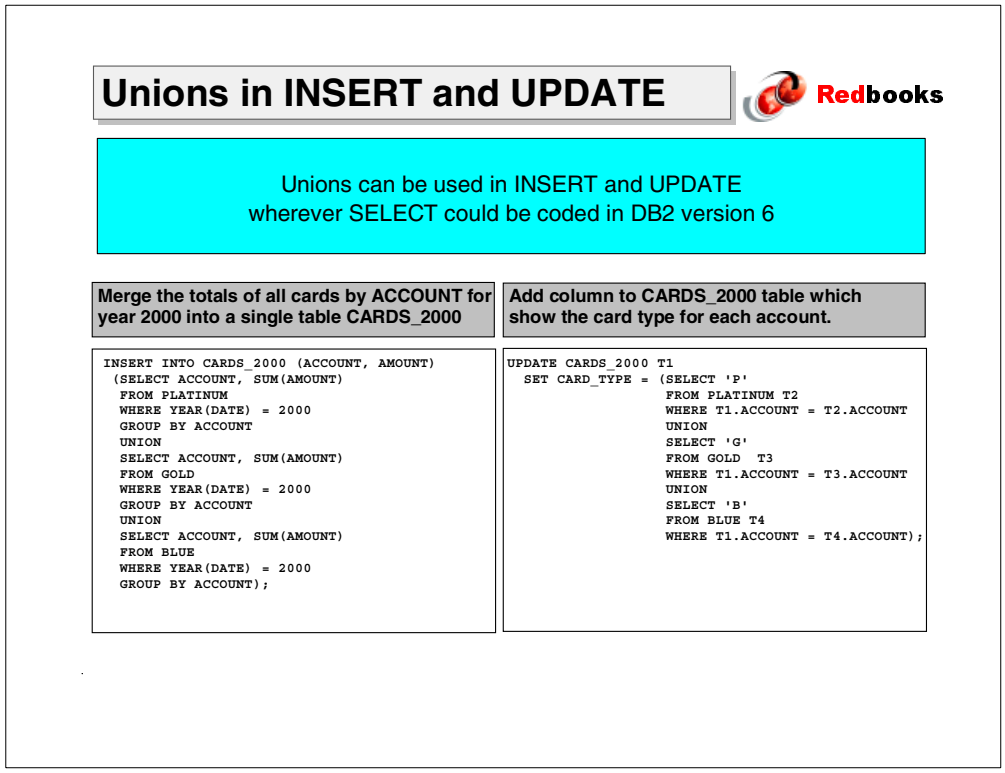



### 2.1.7 Unions in IN predicate

This example lists all accounts that were not used during January 2000. By using the UNION in the IN predicate, we are able to scan all three card tables using only a single predicate clause. Under DB2 V6, to code this statement, three AND'd IN clauses would have to be coded against each card table individually.

The results of this query are:

```
ACCOUNT  ACCOUNT_NAME
---------+---------+---
BMP291   BASEL FERRARI
ULP231   MS S FLYNN
```




## 2.1.8 Unions in INSERT and UPDATE

The above examples show how the UNION keyword can now be used in both the INSERT and UPDATE statements in DB2 V7. Basically, the rule that can be used is that unions are valid wherever a SELECT could be coded in the previous versions of DB2.

### 2.1.8.1 Union in the INSERT statement

The above example shows a simple way in which data from many separate tables can be selected, merged, and inserted into a single table using the UNION keyword in the INSERT statement.

A table CARDS_2000 is created. The INSERT takes only rows from the three card tables and places them into the newly created table. A SELECT run against the resulting table returns the results:

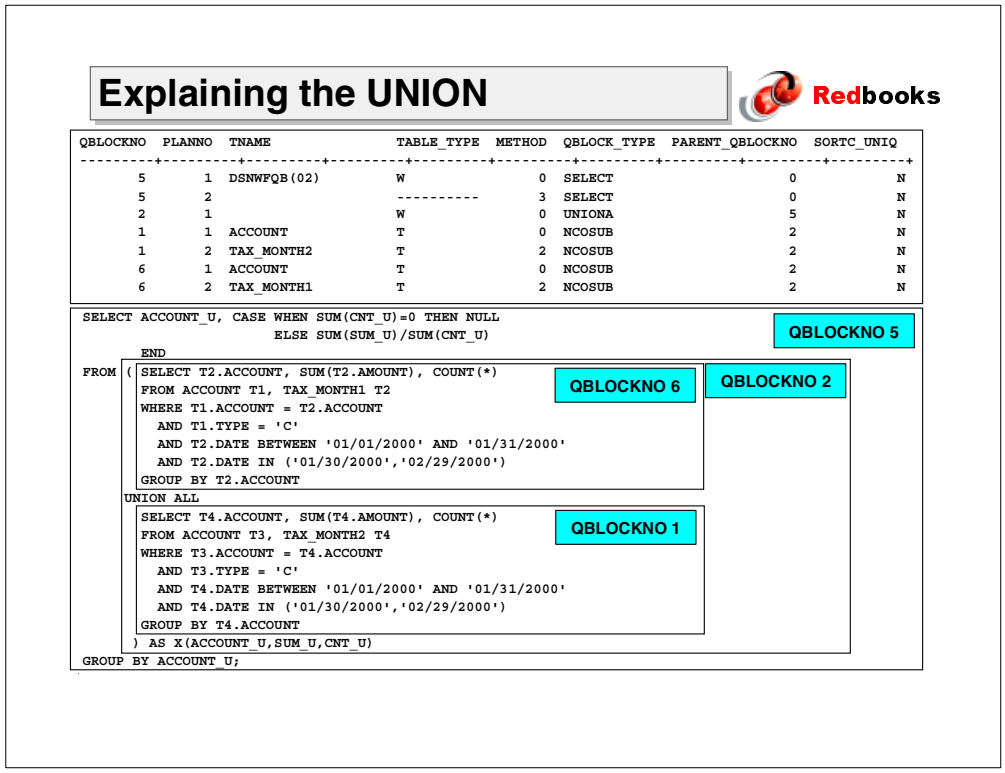
```
ACCOUNT     AMOUNT
---------+--------
ABC010          .00
BWH450       150.00
MNP230       150.00
ZXY930          .00
XPM673     10896.15
XPM961     14753.65
```

#### 2.1.8.2 Union in the UPDATE statement

In our example, we are adding a column to the CARDS_2000 table which shows the card type. The example shows how this column could be populated using the UNION keyword in the SET clause of the UPDATE statement. When populated, the CARDS_2000 table would look like this:

```
ACCOUNT     AMOUNT  CARD_TYPE
---------+---------+---------+
ABC010         .00  P
BWH450      150.00  P
MNP230      150.00  G
XPM673    10896.15  B
XPM961    14753.65  B
ZXY930         .00  G
```

The UNION or UNION ALL can now also be used in the WHERE clause of the UPDATE statement in much the same manner as for the WHERE clause of the SELECT statement.




### 2.1.9 Explaining the UNION

To support explaining of the UNION, two new columns have been added to the PLAN_TABLE. These are:

1. **PARENT_QBLOCKNO**
   Contains the number of the parent query block of this query block. In the example shown above query block number 6 has a parent query block number of 2 which is the query block number of the table expression.This means that the SELECT query is executed to fulfil the table expression.
2. **TABLE_TYPE**
   In general the possible values of this column are:

   **F** - function

   **Q** - temporary intermediate result table (used by STAR JOIN, not UNION)

   **T** - table

   **W** - work file

New values have been added to the existing QBLOCK_TYPE column, which shows the type of operation performed by the query block. The possible new values are:

**TABLEX** - table expression

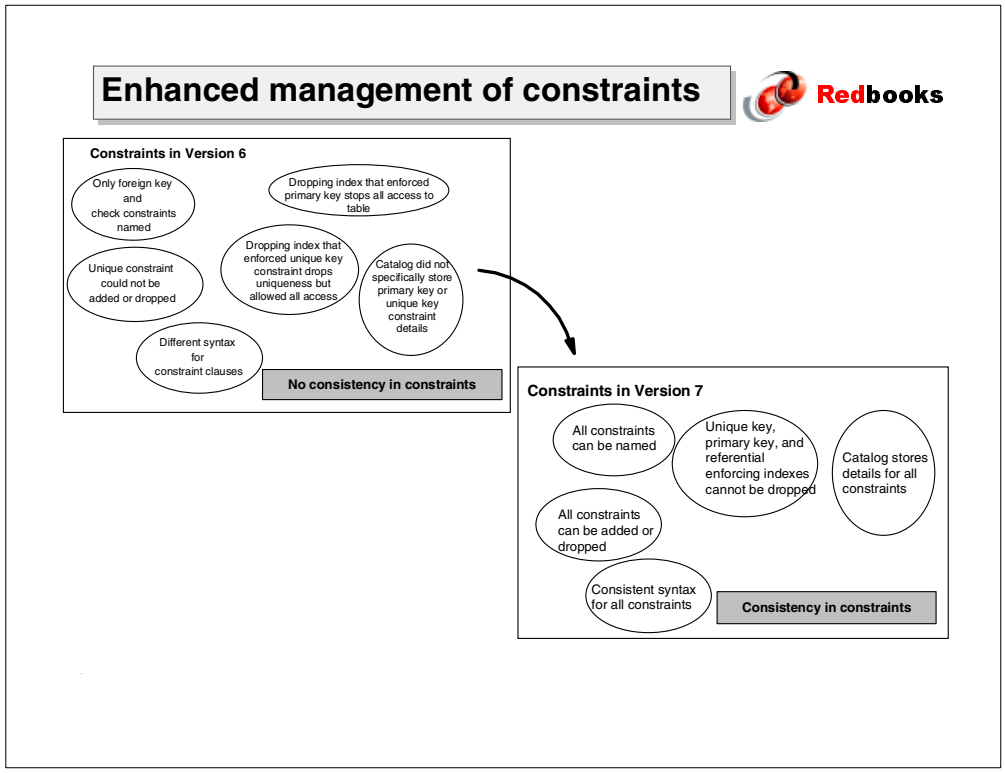
**UNION** - UNION

**UNIONA** - UNION ALL

The above EXPLAIN output shows the use of these new values for query block number 2, which has been rewritten to be a UNION ALL. What is most significant is the way in which the PARENT_QBLOCKNO column now gives some idea of how the statement is to be executed. In our example the lowest level query blocks are 1 and 6. These are executed first to create the table expression in query block number 2, which itself provides the basis for query block number 5.

In the query block number 5, the table type is set to ‘W’, which shows that materialization has taken place and that a work file has been created.

It should be noted that this is the only way to figure out how the rewriting of the query has taken place. DB2 does not provide specific information on the outcome of a query rewrite.




## 2.2 Enhanced management of constraints

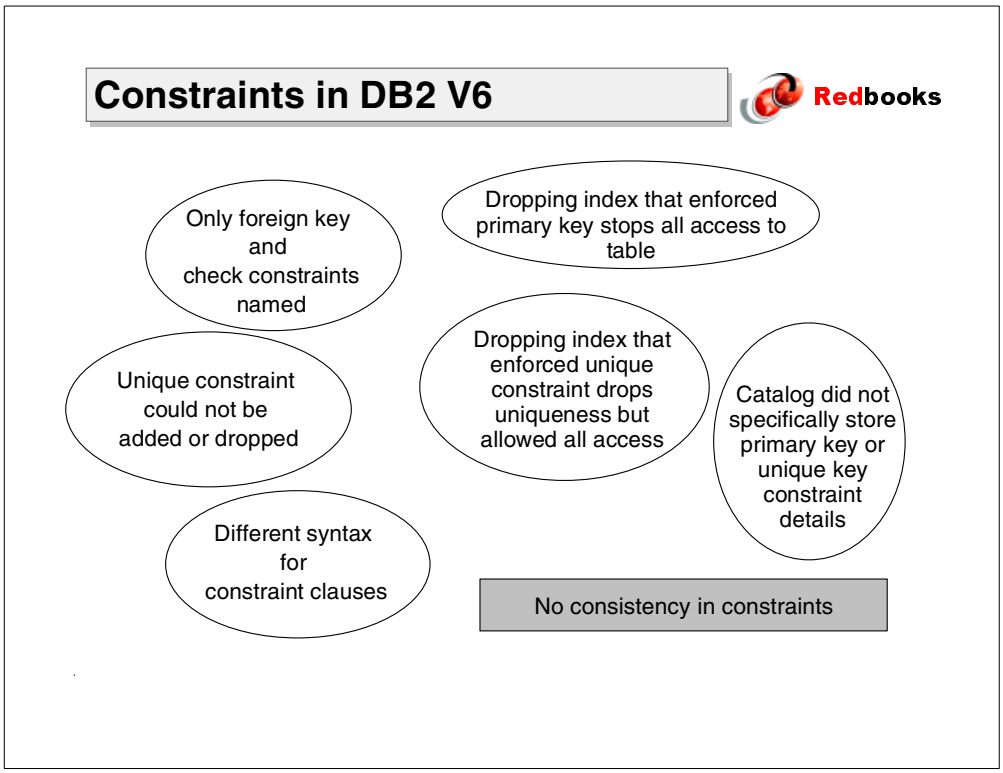
DB2 V7 introduces enhancements to the way in which primary key, unique key, foreign key and check constraints are created, maintained, and deleted.

To achieve this, the following enhancements have been made:

- All constraints can be named.
- Consistent syntax is provided for constraint clauses.
- Consistent rules are provided for dropping constraint enforcing indexes.
- Constraint details are stored explicitly in the DB2 catalogs.
- The TABLESTATUS column of SYSIBM.SYSTABLES flags incomplete table status for unique and primary key constraints.

These enhancements ensure that constraints can be managed in a consistent manner across the databases to ensure the integrity of the data stored in DB2 databases.

The changes also have clarified the manner in which unique key constraints are maintained in the DB2 catalog.




### 2.2.1 Constraints in DB2 V6

DB2 V6 had full functionality to ensure that the data in the tables under its management remained consistent. It used parent foreign relationship, check, primary key, and unique key constraints to manage the data in the database. However, in many ways, the manner in which these constraints were set up and the manner in which they operated was, to say the least, inconsistent.

First, only foreign key and check constraints could be named. Even in this regard, the way in which the constraint was named was different. For example, the following clause set up a foreign key relationship and named it as CONSTR1:

```
...FOREIGN KEY CONSTR1 (SUPPLIER_ID)
     REFERENCES SUPPLIER_TABLE (SUPPLIER_ID)...
```

The syntax for a check constraint was different. To define a check constraint with the name CONSTRAINT_C1, a clause like this would be used:

```
...CONSTRAINT CONSTRAINT_C1 CHECK (AGE BETWEEN 18 AND 105)...
```

All constraints could be defined in the CREATE TABLE statement; and the foreign key, check constraint, and primary key could be added and dropped using the ALTER TABLE command, like this:

```
ALTER TABLE CUSTOMER
  ADD PRIMARY KEY (CUSTOMER_ID);
```

or

```
ALTER CUSTOMER
  DROP PRIMARY KEY;
```

However, there was no clause in the ALTER TABLE statement that would allow a unique key constraint to be added or dropped.

The way in which indexes enforced the primary key and unique key constraints varied. If a table was created with a primary key specified and an attempt was made to access the table in any way, the following error was returned:

```
DSNT408I SQLCODE = -540, ERROR: THE DEFINITION OF TABLE CREATOR.TBNAME
         IS INCOMPLETE BECAUSE IT LACKS A PRIMARY INDEX OR A REQUIRED UNIQUE
         INDEX
DSNT418I SQLSTATE    = 57001 SQLSTATE RETURN CODE
```

If a table was created with a unique key specified only, the table could be read without returning an error. Any attempt to insert into this table would return the above message.

Until an index was created to support this key, a row was added to the SYSCOLUMN table with a COLNO of 0 which recorded the details of the key.

If an index was created to enforce the primary and unique keys, then full access was available to both. For a unique key constraint when the supporting index was created, the SYSCOLUMN COLNO 0 row was removed, and the DB2 catalog no longer maintained any references to the unique key constraint.

When an index, which enforced a primary key constraint, was dropped, the following warning would occur:

```
DSNT408I SQLCODE = 625, WARNING: THE DEFINITION OF TABLE CREATOR.TBNAME
         HAS BEEN CHANGED TO INCOMPLETE
DSNT418I SQLSTATE    = 01518 SQLSTATE RETURN CODE
```

Following this, all access to this table would return the following error:

```
DSNT408I SQLCODE = -904, ERROR: UNSUCCESSFUL EXECUTION CAUSED BY AN
         UNAVAILABLE RESOURCE. REASON 00C9009F, TYPE OF RESOURCE 00000D01, AND
         RESOURCE NAME nnn.n
DSNT418I SQLSTATE    = 57011 SQLSTATE RETURN CODE
```

The reason code 00C9009F meant that a table was incomplete, as it no longer had a unique index to enforce the primary key constraint. This situation would remain until the unique index was reinstalled or the constraint dropped.

If a drop of an index which enforced a unique key constraint took place, the following warning would occur:

```
DSNT408I SQLCODE = 626, WARNING: DROPPING THE INDEX TERMINATES ENFORCEMENT OF
         THE UNIQUENESS OF A KEY THAT WAS DEFINED WHEN THE TABLE WAS CREATED
DSNT418I SQLSTATE    = 01529 SQLSTATE RETURN CODE
```

However, if no foreign key constraint used the unique key constraint as its parent key, all access to the table would be still possible, without the unique key constraint restricting the data values entered.

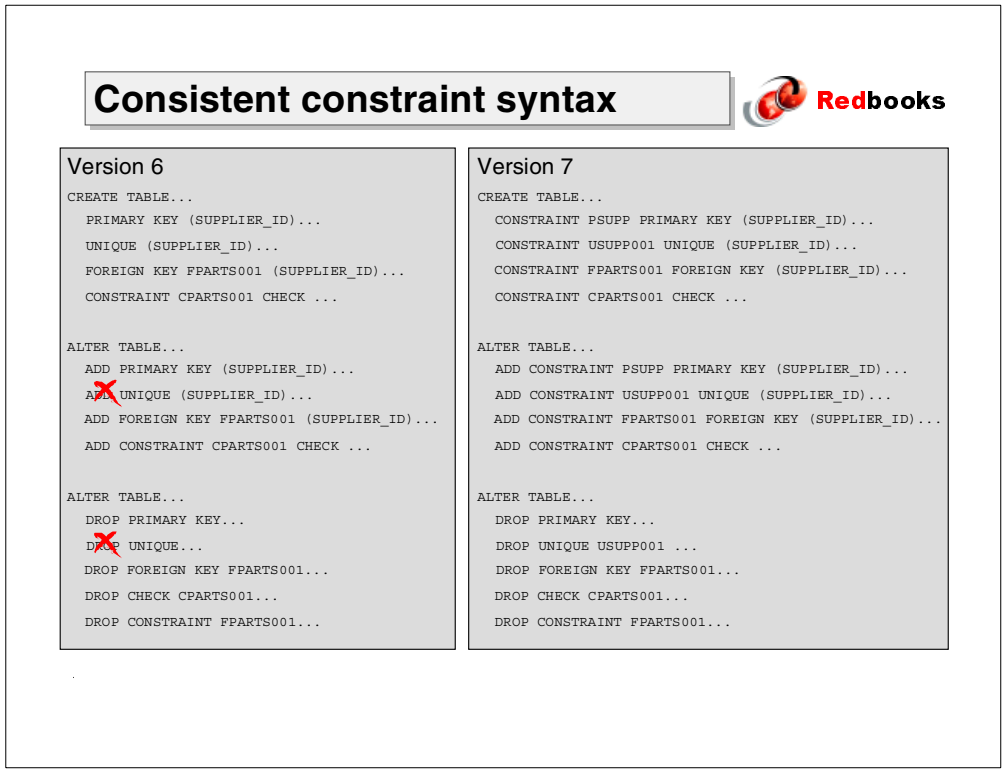
If a foreign key was set to use the unique key constraint as it parent key, then all access to this table was stopped, and the following error would be returned:

```
DSNT408I SQLCODE = -904, ERROR: UNSUCCESSFUL EXECUTION CAUSED BY AN
         UNAVAILABLE RESOURCE. REASON 00C9009F, TYPE OF RESOURCE 00000D01, AND
         RESOURCE NAME nnn.n
DSNT418I SQLSTATE     = 57011 SQLSTATE RETURN CODE
```

As can be seen from this, the behavior of enforcing indexes changed in different circumstances, and there was no consistency as to how or what would occur if an enforcing index was dropped.

Finally, the way in which details about the constraints were stored in the catalog table also varied. Both foreign key and check constraints had details stored in specific catalog tables: SYSIBM.SYSRELS and SYSIBM.SYSCHECKS, respectively. The details about the primary key columns were stored in SYSIBM.SYSCOLUMNS, but no further details were recorded. Unique key constraints were not stored in any manner that was accessible to the user once the supporting index had been created. If the unique index that enforced a unique key constraint was dropped, all information about the constraint disappeared.

V7 of DB2 addresses all of these inconsistencies and allows the user better means of creating, using, and maintaining constraints.




## 2.2.2 Consistent constraint syntax

V7 of DB2 introduces consistent syntax for the coding of all constraint clauses. This means that all constraints can now be created, added, or dropped in the same way through the CREATE TABLE and ALTER TABLE statements.

This enhancement achieves this through the addition of the CONSTRAINT keyword for all constraint types in the CREATE TABLE and ALTER TABLE statements. In previous versions, this keyword was only valid for the CHECK constraint clause. Now all constraints can be defined using a similar syntax.

### 2.2.2.1 Naming all constraints

The most important element of this enhancement is that all constraints can be explicitly named. This name can contain up to 8 characters for foreign key constraints and up to 18 characters (even though in the catalog there is space for 128) for all other constraints and must be unique for all constraints defined for that table.

The naming of constraints is optional, as is the use of the CONSTRAINT keyword. However, DB2 will always provides a name for all constraints, whereas in previous versions of DB2, this was only true for check and foreign key constraints. If a name is not specified, then DB2 gives the constraint a name by using the same name as the first column referred to in the constraint definition.

For example, if a constraint is defined for a primary key:

```
...
PRIMARY KEY (SUPPLIER_ID)
...
```

DB2 will name the constraint SUPPLIER_ID. If a further unique key constraint is defined by the clause:

```
...
UNIQUE (SUPPLIER_ID, SUPPLIER_NAME)
...
```

DB2 will name this constraint SUPPLIER_ID1. DB2 will always ensure that the name of a constraint which it defines will always follow its own rule, thus ensuring that the name is unique for all constraints defined for a table. This is the same naming convention as used where foreign key and check constraints were not explicitly named in previous versions.

As all constraints are now named with a distinct name it is now possible to drop all constraints using the DROP CONSTRAINT clause of the ALTER TABLE statement. If this clause is used, then the constraint name must be used.

#### 2.2.2.2 Managing unique key constraints

The standardization of the constraint syntax now allows unique key constraints to be handled in the same way as all other constraints. Unique keys can be defined when a table is created in the same fashion as other constraints. Of more significance, unique keys can be added to a table following its creation or dropped using the DROP CONSTRAINT clause of the ALTER TABLE statement. These constraints can also be dropped using the DROP UNIQUE clause of the same statement. This manipulation of unique keys was not possible in previous versions of DB2.

#### 2.2.2.3 Syntax from previous versions

DB2 accepts all the constraint clauses that could be coded in previous versions. This mainly applies to the FOREIGN KEY clause where the constraint name was coded following the FOREIGN KEY keywords. For example, the clauses:

```
...FOREIGN KEY FPART01 (SUPPLIER_ID) REFERENCES...
```

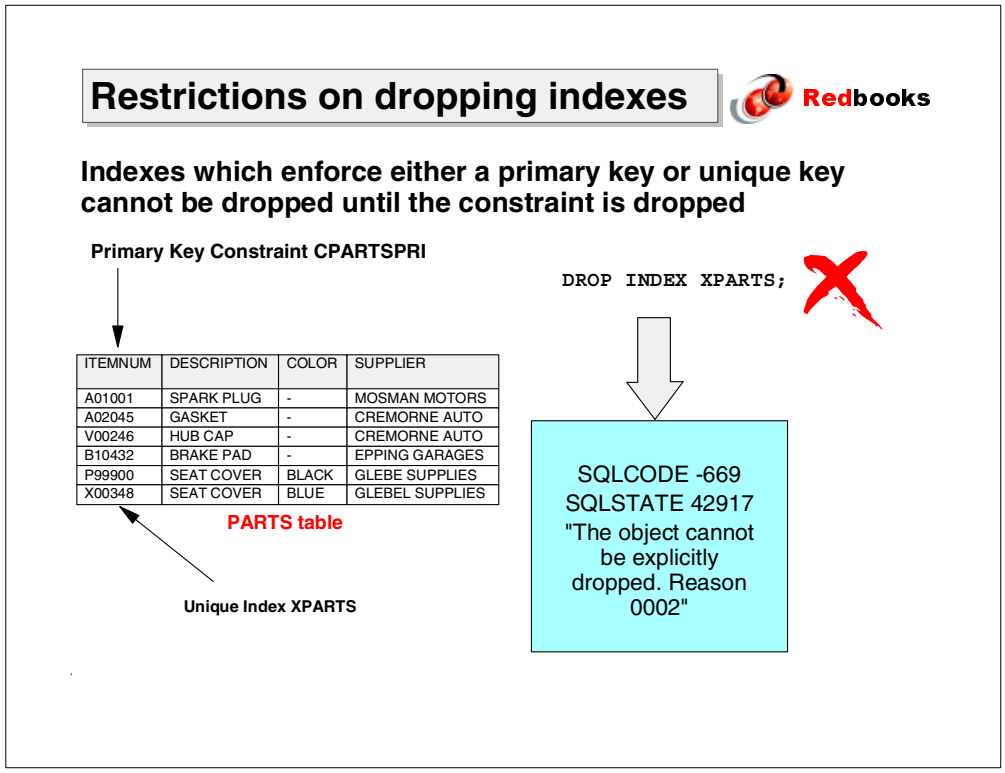
or

```
...CONSTRAINT FPART01 FOREIGN KEY (SUPPLIER_ID) REFERENCES...
```

are both valid in V7 of DB2.

#### 2.2.2.4 Constraint name inconsistency

One inconsistency still exists in the constraint names used in DB2 V7. All constraint names except the foreign key constraint can be up to 18 characters in length. The foreign key constraint name can only be up to eight characters.




## 2.2.3 Restriction on dropping indexes

Prior to DB2 V7, an index that supported a primary key constraint could be dropped and the table would be marked as incomplete. All access to the table would return an error and would fail.

For an index that enforced a unique key without a matching foreign key definition, the index could be dropped but the table would be operational. If a foreign key was set to point to the unique key constraint, the table with the unique key constraint was marked incomplete and all access failed.

With DB2 V7, once an index has been defined to enforce a constraint, it cannot be dropped until the constraint itself is removed. Any attempt to remove such an index will result in the error:

```
DSNT408I SQLCODE = -669, ERROR:  THE OBJECT CANNOT BE EXPLICITLY DROPPED,
         REASON 002
```

This protects the user from dropping an index which is fundamental to the enforcement of data integrity without knowing that the index is used to enforce such a rule.

### 2.2.3.1 Removing an index that enforces a constraint

To remove an index that supports a constraint the constraint itself must first be dropped using the ALTER TABLE command. The sample below shows how an enforcing index can be dropped.

To remove a constraint, the following steps would have to be followed:

**Remove any foreign key constraints using the key as a parent**

```
ALTER TABLE PARTS
  DROP FOREIGN KEY FPARTS01;
```

or

```
ALTER TABLE PARTS
  DROP CONSTRAINT FPARTS01;
```

**Remove parent key constraint**

```
ALTER TABLE SUPPLIER
  DROP PRIMARY PSUPP;
```

or

```
ALTER TABLE SUPPLIER
  DROP CONSTRAINT PSUPP;
```

or

```
ALTER TABLE SUPPLIER
  DROP UNIQUE USUPP01;
```

or

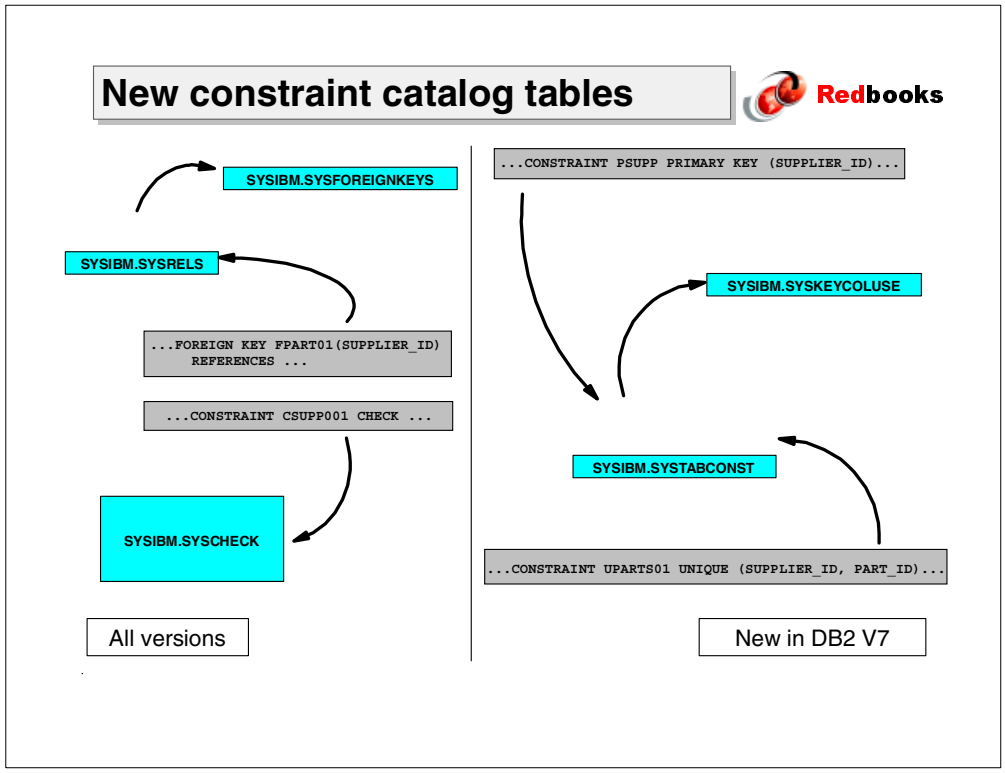
```
ALTER TABLE SUPPLIER
  DROP CONSTRAINT USUPP01;
```

You can now drop the index, as it is no longer supporting a constraint.

#### 2.2.3.2 Dropping enforcing indexes from previous versions of DB2

If a unique key constraint was created using a version prior to V7, it can be dropped without dropping the constraint first. However, dropping an index created in a previous version which enforces the primary key will still be restricted.

This could affect the behavior of third party products which are used to alter DB2 objects and which may operate on the understanding that supporting indexes could be dropped without removing the constraint itself. These should be tested for compatibility in this area.




### 2.2.4 New constraint catalog tables

In previous releases of DB2, only the details pertaining to foreign key and check constraints were held in specific catalog tables. These tables were SYSIBM.SYSRELS and SYSIBM.SYSCHECKS respectively.

For primary key constraints details were stored in the column KEYSEQ of the SYSIBM.SYSCOLUMNS table.

If a unique key constraint was defined in the CREATE TABLE statement, details about this constraint were stored in a system area with SYSIBM.SYSCOLUMNS table until an enforcing index was created. When an index was created to enforce the constraint, these details were removed, and a value of ‘C’ was stored in the UNIQUERULE column of SYSIBM.SYSINDEXES. If this index was not involved in RI and was then dropped, details of the constraint would be gone. If the index was involved in RI and dropped, details of the constraint would be returned to the system area of SYSIBM.SYSCOLUMNS.

To ensure that the primary key and unique constraints are visible, even if the index that enforces them has not been built, two new tables have been added to the DB2 catalog: SYSIBM.SYSTABCONST and SYSIBM.SYSKEYCOLUSE. They are found in the DSNDB06.SYSOBJ table space.

The format of these tables is described below.

#### 2.2.4.1 SYSIBM.SYSTABCONST

This table stores details about primary key and unique key constraints. One row will exist for each constraint of these types.

| Column name | Type | Description |
|---|---|---|
| CONSTNAME | VARCHAR(128)<br>NOT NULL | Name of constraint |
| TBCREATOR | CHAR(8)<br>NOT NULL | Authorization ID of the owner of the table on which the constraint is defined. |
| TBNAME | VARCHAR(18)<br>NOT NULL | Name of the table on which the constraint is defined. |
| CREATOR | CHAR(8)<br>NOT NULL | Authorization ID under which the constraint was created. |
| TYPE | CHAR(1) | Type of constraint:<br>**P** Primary key<br>**U** Unique key |
| IXOWNER | CHAR(8)<br>NOT NULL | Owner of the index enforcing the constraint or blank if index has not been created. |
| IXNAME | VARCHAR(18)<br>NOT NULL | Name of the index enforcing the constraint or blank if index has not been created. |
| CREATEDTS | TIMESTAMP<br>NOT NULL | Time when the statement to create the constraint was executed. |
| IBMREQD | CHAR(1)<br>NOT NULL<br>DEFAULT ‘N’ | Whether the row came from the basic machine readable (MRM) tape:<br>**N** No<br>**Y** Yes |

There are referential integrity rules between this table and SYSIBM.SYSTABLES, and between this table and SYSIBM.SYSKEYCOLUSE.

#### 2.2.4.2 SYSIBM.SYSKEYCOLUSE

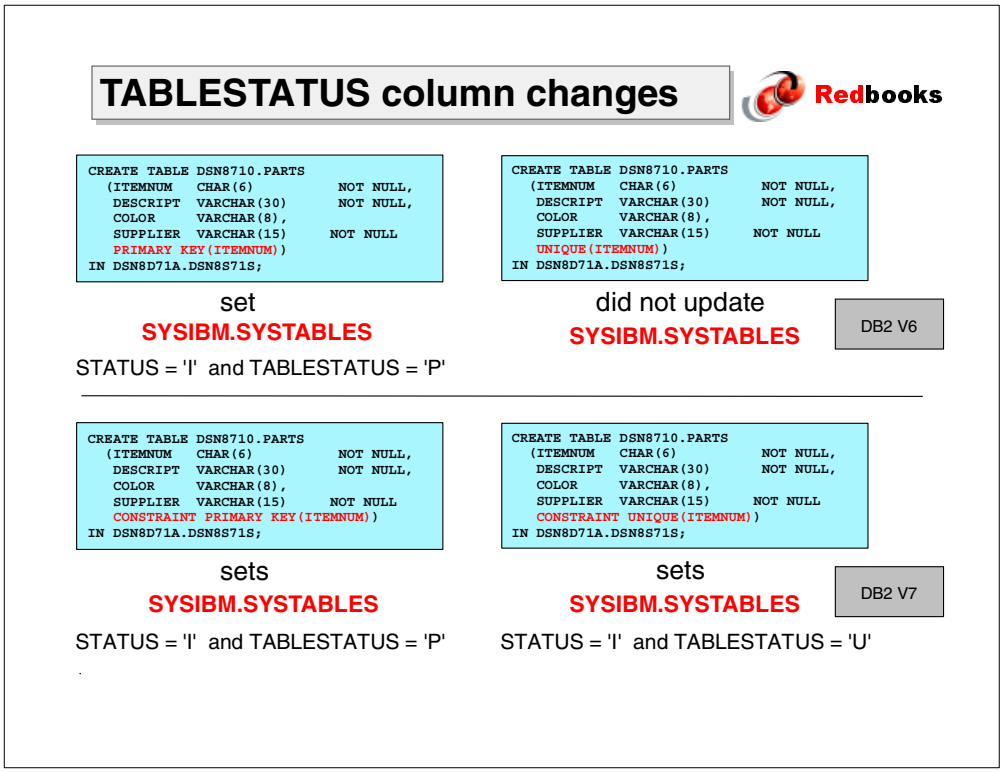
This table stores information about columns used in primary key and unique key constraints. A row will exist for each column used in keys of this type.

| Column name | Type | Description |
|---|---|---|
| CONSTNAME | VARCHAR(128)<br>NOT NULL | Name of constraint |
| TBCREATOR | CHAR(8)<br>NOT NULL | Authorization ID of the owner of the table on which the constraint is defined. |
| TBNAME | VARCHAR(18)<br>NOT NULL | Name of the table on which the constraint is defined. |
| COLNAME | VARCHAR(18)<br>NOT NULL | Name of column. |
| COLSEQ | SMALLINT<br>NOT NULL | Numeric position of the column in the key (the first position in the key is 1) |
| COLNO | SMALLINT<br>NOT NULL | Numeric position of the column in the table on which the constraint is defined. |
| IBMREQD | CHAR(1)<br>NOT NULL<br>DEFAULT 'N' | Whether the row came from the basic machine readable (MRM) tape:<br>**N** No<br>**Y** Yes |

There are referential integrity rules between this table and SYSIBM.SYSTABCONST.

Through the creation of these tables, details are maintained in the catalog as long as the constraint exists, even if the enforcing index does not exist. With the restriction on the dropping of indexes that enforce a constraint, this situation will only occur between the time of table creation and the time of the enforcing index creation.

With the creation of these tables, it is recommended that a unique key constraint be generated for all unique indexes. This ensures that the catalog maintains a comprehensive list of unique enforcing indexes on the databases, and removes the chance that a unique constraint can be removed by accident.




## 2.2.5 SYSTABLES constraint column changes

### 2.2.5.1 Prior to DB2 V7

DB2 stored flags within SYSIBM.SYSTABLES to show that a table was incomplete due to a primary key constraint not having a matching unique index. The column STATUS was set to ‘I’ to show that the table was incomplete, and TABLESTATUS was set to ‘P’ to show that a unique index that matched the primary key was required. When a unique index was created that matched the primary key, the STATUS column was set to ‘X’ to show that the table was now complete, and the TABLESTATUS column was set to blank.

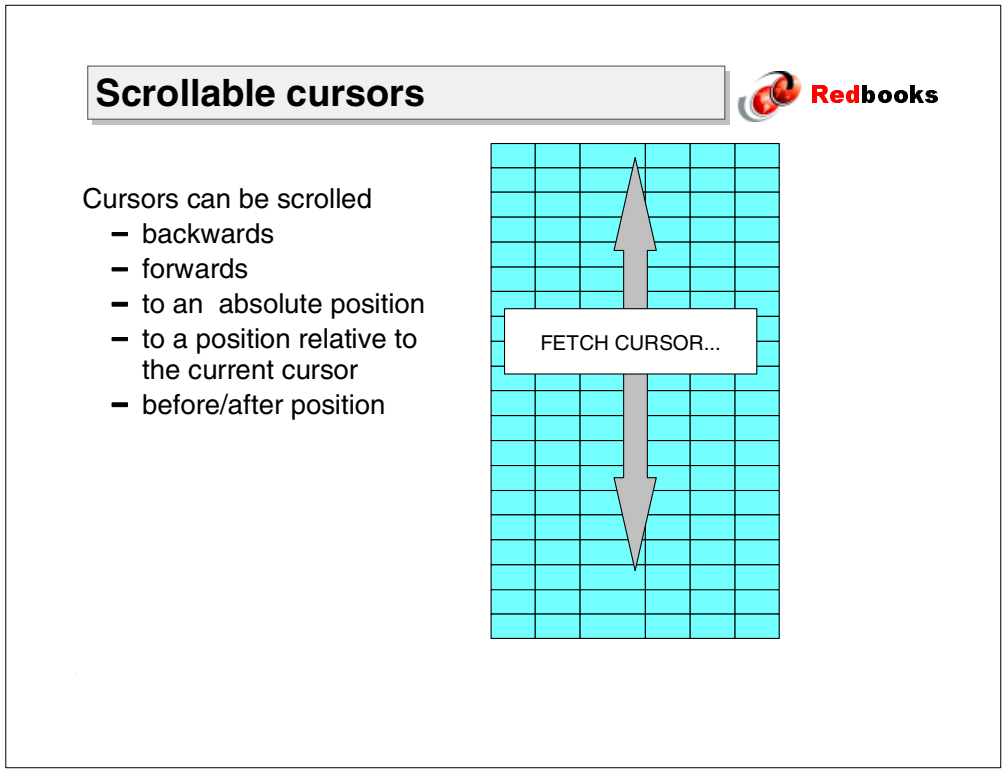
Setting up a unique key constraint in the CREATE table statement would not set any values of the STATUS and TABLESTATUS columns.

### 2.2.5.2 DB2 V7

If a table is created with a primary key in V7, then the STATUS and TABLESTATUS are set in exactly the same way as they were in the previous versions. The difference is that these columns now show if a table is incomplete due the creation of a unique key constraint in the CREATE TABLE statement.

When a unique key is set up at table creation time, the STATUS column is also set to ‘I’ to show that the table is incomplete. At the same time, the TABLESTATUS column is updated with a ‘U’ to show that an index must be built to support a unique constraint to complete the table.

If the setting up of a primary key index and unique constraint indexes are required to complete the table, the TABLESTATUS column will contain ‘PU’ to represent this.




## 2.3 Scrollable cursors

With the previous versions of DB2, cursors could only be scrolled in a forward direction. Cursors were opened and would be positioned before the first row of the result set. To move, a FETCH would be executed, and the cursor would move forward one row. The application program had to rely on different techniques to get around this limitation.

If we wanted to move back in a result set, as any programmer would attest was often the case, there were a number of options. One option was to CLOSE the current cursor and to start at the beginning, then repeat the FETCH until the desired row was reached. This was a slow response for large result sets, as many rows would be read unnecessarily on the way to the target row. Also, if other users had inserted data, this could effect the row number. Program logic had to be built to either ignore or process the change in row data.

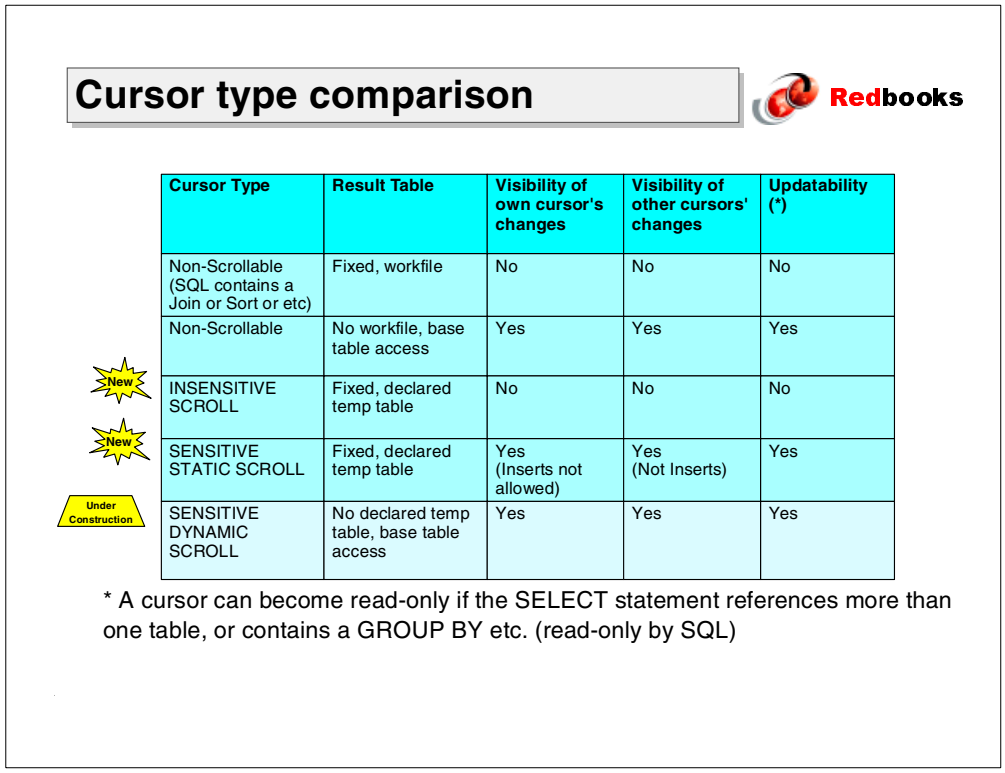
A second alternative was to build arrays in the program's memory areas. The result set would be opened and all the rows would be read into the array. The program would, then move backwards and forwards within this array.

This option needed to be carefully planned, as it could waste memory through low utilization of the space, or it could restrict the number of rows returned to the user to some arbitrary number. It was also hampered by the break with the actual data in the table and the data in the array. If other users were changing data, the program could miss this, as there was no fixed relationship between the array data and the table data once read.

The normal procedure was to select a changed row to verify that the data had not changed while the user was acting upon it and reject the update if the row had changed or update the new values. All this had to be coded in a program and not in DB2.

V7 introduces facilities to allow scrolling in any direction (except sideways) and also the ability to place the cursor at a specific row within the result set. New keywords have been added to both the DECLARE CURSOR and FETCH statements to set up a scrollable cursor.

DB2 also can, if desired, maintain the relationship between the rows in the result set and the data in the base table. That is, the scrollable cursor function allows the changes made outside the cursor opened, to be reflected. For example, if the currently fetched row has been updated while being processed by the user, and an update is attempted, a warning is returned by DB2 to reflect this. When another user has deleted the row currently fetched, DB2 will return an SQLCODE if an attempt is made to update the deleted row.




| Cursor Type | Result Table | Visibility of own cursor's changes | Visibility of other cursors' changes | Updatability (*) |
|---|---|---|---|---|
| Non-Scrollable (SQL contains a Join or Sort or etc) | Fixed, workfile | No | No | No |
| Non-Scrollable | No workfile, base table access | Yes | Yes | Yes |
| INSENSITIVE SCROLL | Fixed, declared temp table | No | No | No |
| SENSITIVE STATIC SCROLL | Fixed, declared temp table | Yes (Inserts not allowed) | Yes (Not Inserts) | Yes |
| SENSITIVE DYNAMIC SCROLL | No declared temp table, base table access | Yes | Yes | Yes |

* A cursor can become read-only if the SELECT statement references more than one table, or contains a GROUP BY etc. (read-only by SQL)

## 2.3.1 Cursor type comparison

***Non-scrollable cursor-characteristics***

- These are used by an application program to retrieve a set of rows or a result set from a stored procedure.
- The rows must be processed one at a time.
- The rows are fetched sequentially.
- Result sets may be stored in a workfile.

***Scrollable cursor-characteristics***

- These are used by an application program to retrieve a set of rows or a result set from a stored procedure.
- The rows can be fetched in random order.
- The rows can be fetched forward or backward.
- The rows can be fetched from the current position or from the top of the result table or result set.
- The result set will be fixed at OPEN CURSOR time.
- Result sets are stored in Declared Temporary Tables (V6 Line Item).
- Results sets go away at CLOSE CURSOR time.

***Insensitive scrollable cursor***

- Always static
- Fixed number of rows
- Declared temp table used

***Sensitive static scrollable cursor***

- Always static
- Fixed number of rows
- Declared temp table used
- Sensitive to changes, but not to inserts

***Sensitive dynamic scrollable cursor***

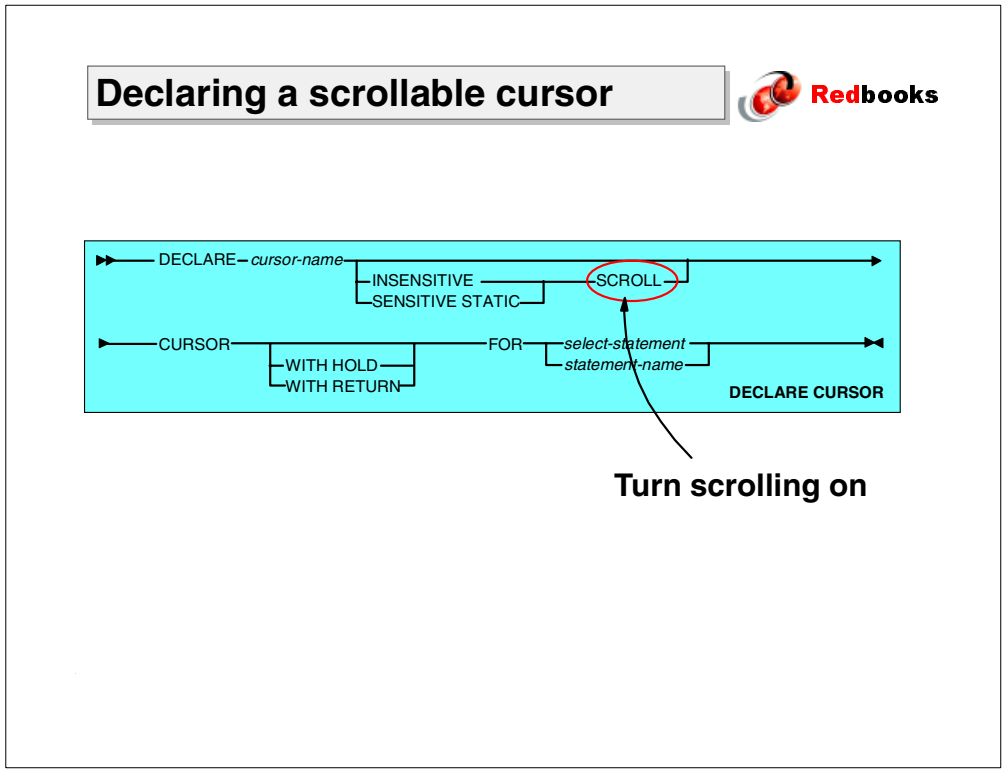
- Direct table access.
- Live data access.

***Updatability***

- The SELECT statement of DECLARE CURSOR can force the cursor to be read-only based on existing cursor rules.

***Non-scrollable and scrollable cursor-characteristics***

- The result set can be sensitive to its own changes.
- The result set can be sensitive to other's changes.
- The result set can be updatable.
- The result set can be fetched, updated, or deleted.



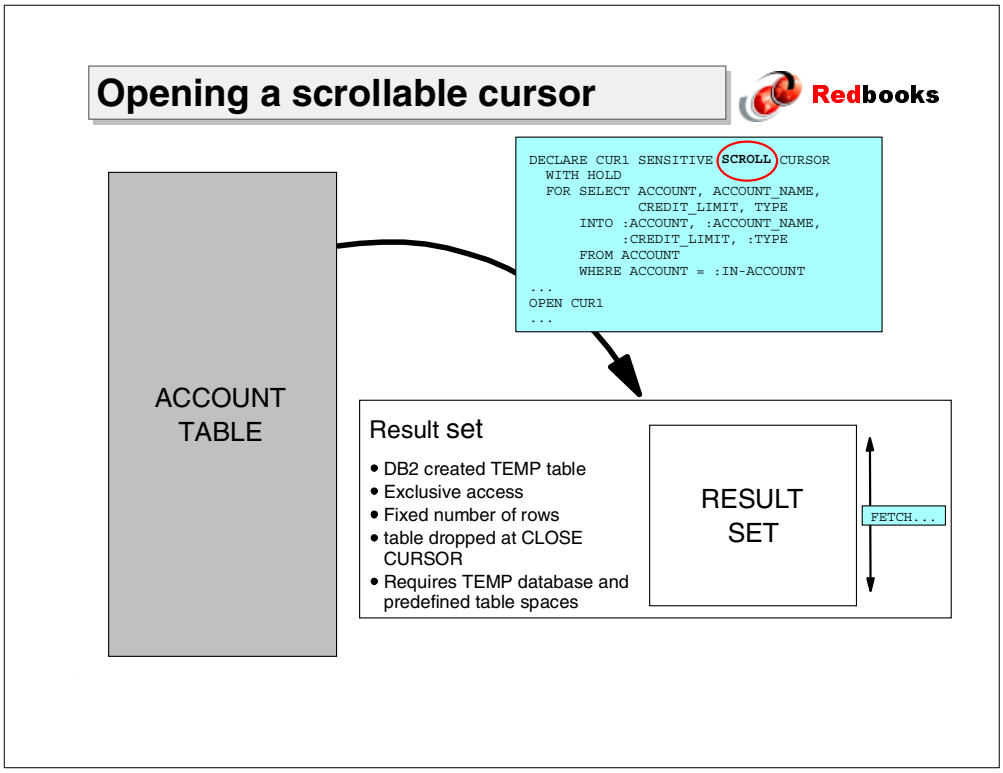

### 2.3.2 Declaring a scrollable cursor

To turn on a scrolling cursor, you use the SCROLL keyword in the DECLARE CURSOR statement. The FETCH statements that use this cursor can now be moved in any direction in the result set defined by the OPEN CURSOR SELECT statement.

The new keywords INSENSITIVE and SENSITIVE STATIC deal with the sensitivity of the cursor to changes made to the underlying table.

The STATIC keyword in the context of this clause does not refer to static and dynamic SQL, as scrollable cursors can be used in both these types of SQL. Here, STATIC refers to the size of the result table, once the OPEN CURSOR is completed, the number of rows remains constant.

For a more complete description, refer to 2.3.6, “Insensitive and sensitive cursors” on page 61.




### 2.3.3 Opening a scrollable cursor

To open a cursor that will use scrolling, the keyword SCROLL is used on the DECLARE CURSOR statement.

When the DECLARE CURSOR...SCROLL and OPEN CURSOR are executed, a temporary table in the DB2 TEMP database is created to hold the result set. The TEMP database has to be created by you. Following is an example of how to create it:

```
CREATE DATABASE DBTEMP01 AS TEMP STOGROUP DSN8G710;
CREATE TABLESPACE TSTEMP04 IN DBTEMP01 USING STOGROUP DSN8G710
  SEGSIZE 4;
CREATE TABLESPACE TSTEMP08 IN DBTEMP01 USING STOGROUP DSN8G710
  SEGSIZE 8;
CREATE TABLESPACE TSTEMP16 IN DBTEMP01 USING STOGROUP DSN8G710
  SEGSIZE 16;
CREATE TABLESPACE TSTEMP32 IN DBTEMP01 USING STOGROUP DSN8G710
  SEGSIZE 32 BUFFERPOOL BP32K;
```

All the rows which fit the selection criteria are, then written from the base table to the temporary table. The record identifier (RID) of the row is also retrieved and stored with the rows in the temporary table. If the cursor is declared as SENSITIVE STATIC, the RID's are used to maintain changes between the result set row and base table row.

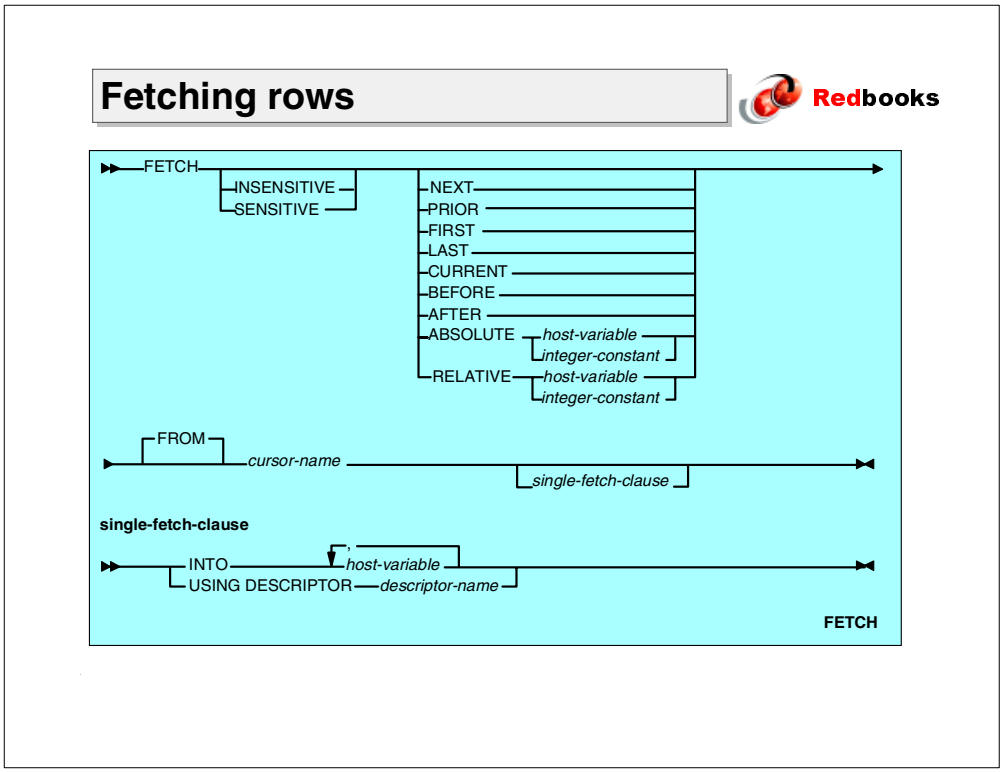
The temporary table that is created is only accessible by the agent that created it. Each user which is running an application that contains scrollable cursors will create its own temporary table at OPEN CURSOR time. The temporary table will be dropped on CLOSE CURSOR or at the completion of the program that invoked it.

It is important to note that, for the cursor which is declared as INSENSITIVE or SENSITIVE STATIC, the number of rows of the result set table will not change once the rows are retrieved from the base table and stored. This means that all subsequent inserts which are made by other users or by the current process into the base table and which would fit the selection criteria of the cursor's SELECT statement will not be visible to the cursor. Only updates and deletes to data within the result set may be seen by the cursor.

For statements that were coded before V7 of DB2, which only provided the ability to move forward from the current cursor position, there is no change to the way that they operate, and they will not create the temporary table to store the result set. OPEN CURSOR statements that do not use the new keyword SCROLL will not create the temporary table and will only be able to scroll in a forward direction.

To allow for the use of scrollable cursors, the DB2 TEMP database must be predefined. You should note that as all scrollable cursor result sets will be written to this database and its related table spaces, it is important to ensure that there is enough space in these objects to contain the resultant number of rows.

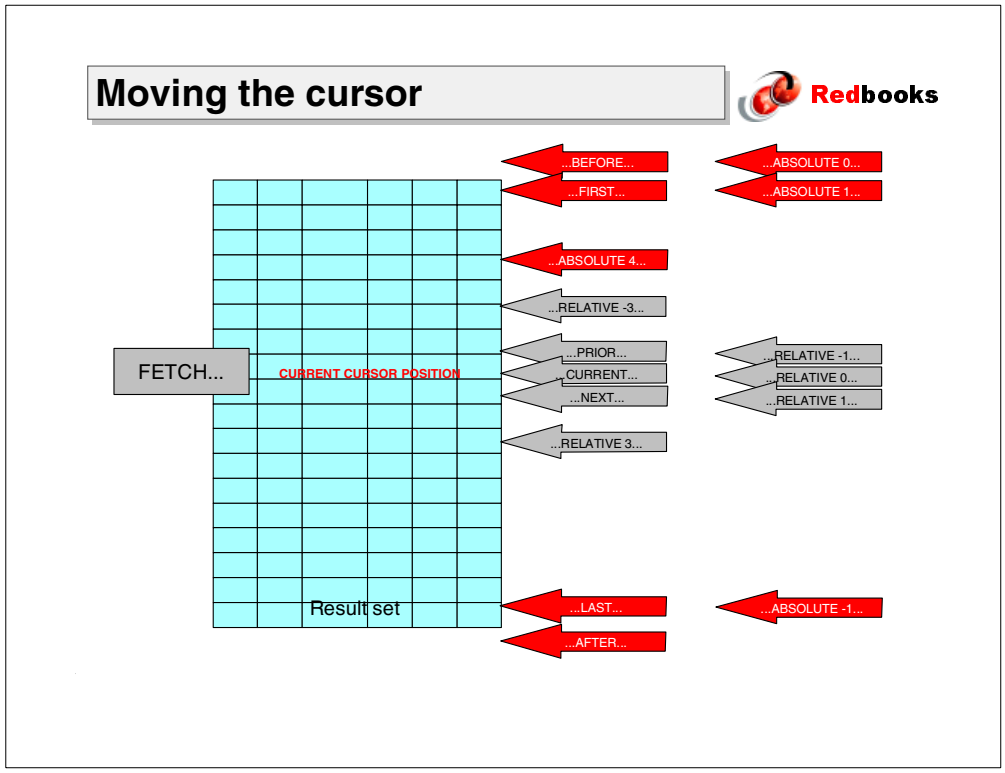
Once the result set has been retrieved, it is only visible to the current cursor process and only remains current until a CLOSE CURSOR is executed or the process itself completes. For programs, the result set is dropped on exit of the current program; for stored procedures, the cursors defined are allocated from the calling program, and the result set is dropped when the calling program concludes.




### 2.3.4 Fetching rows

The above diagram shows the syntax diagram for the FETCH statement. The syntax has been expanded to allow this statement to control cursor movement both forwards and backwards. It also has keywords to position the cursor in specific positions within the result set returned by the OPEN CURSOR statement.

It is important to emphasize that the BEFORE and AFTER clauses are positioning orientation, which means that no data is returned and SQL code zero (0) returns.




## 2.3.5 Moving the cursor

DB2 V7 introduces the ability to move the cursor both backwards and forwards within a result set. To achieve this, two types of commands have been provided which can be broken into two categories of cursor movement. The first category allows for positioning on specific rows within the result set, based on the first row being row number 1. These are called absolute moves. The second type allows for movement relative to the current cursor position, such as moving five rows back from the current cursor position. These are known as relative moves.

### 2.3.5.1 Absolute moves

If a program wants to retrieve the fifth row of a table, the FETCH statement would be coded as:

```
FETCH ... ABSOLUTE +5 FROM CUR1
```

or

```
MOVE 5 TO CURSOR-POSITION
...
FETCH ... ABSOLUTE :CURSOR-POSITION
```

Here, `:CURSOR-POSITION` is a host-variable of type INTEGER.

Another form of the absolute move is through the use of keywords which represent fixed positions within the result set. For example, to move to the first row of a result set, the following FETCH statement can be coded:

```
FETCH ... FIRST FROM CUR1
```

This statement can also be coded as:

```
FETCH ... ABSOLUTE +1 FROM CUR1
```

There are also two special absolute keywords which allow for the cursor to be positioned outside the result set. The keyword BEFORE is used to move the cursor before the first row of the result set and AFTER is used to move to the position after the last row in the result set. Host variables cannot be coded with these keywords as they can never return values.

A synonym for BEFORE is ABSOLUTE 0. However, ABSOLUTE 0 returns no data and SQLCODE +100.

#### 2.3.5.2 Relative moves

A relative move is one made with reference to the current cursor position. To code a statement which moves five rows back from the current cursor, the statement would be:

```
FETCH ... RELATIVE -5 FROM CUR1
```

or

```
MOVE -5 TO CURSOR-MOVE
...
FETCH ... RELATIVE :CURSOR-MOVE
FROM CUR1
```

Here, `CURSOR-MOVE` is a host-variable of INTEGER type.

As with absolute moves there are keywords that also make fixed moves relative to the current cursor position. For example, to move to the next row, the FETCH statement would be coded as:

```
FETCH ... NEXT FROM CUR1
```

If an attempt is made to make a relative jump which will be positioned either before the first row or after the last row of the result set, an SQLCODE of +100 will be returned. In this case the cursor will be positioned just before the first row, if the jump was backwards through the result set; or just after the last row, if the jump was forward within the result set.

#### 2.3.5.3 New FETCH keywords for moving the cursor

Below is a complete list of the new keywords which have been added to the FETCH statement syntax for moving the cursor.

| | |
|---|---|
| **NEXT** | Positions the cursor on the next row of the result table relative to the current cursor position and fetches the row - This is the default. |
| **PRIOR** | Positions the cursor on the previous row of the result table relative to the current position and fetches the row |
| **FIRST** | Positions the cursor on the first row of the result table and fetches the row |
| **LAST** | Positions the cursor on the last row of the result table and fetches the row. |

**CURRENT** Fetches the current row without changing position within the result table.

If CURRENT is specified and the cursor is not positioned at a valid row (for example, BEFORE the beginning of the result table) a warning SQLCODE +231, SQLSTATE 02000 is returned.

**BEFORE** Positions the cursor before the first row of the result table.

No output host variables can be coded with this keyword as no data can be returned

**AFTER** Positions the cursor after the last row of the result table

No output host variables can be coded with this keyword, as no data can be returned.

**ABSOLUTE** Used with either a *host-variable* or *integer-constant.* This keyword evaluates the *host-variable* or *integer-constant* and fetches the data at the row number specified.

If the value of the *host-variable* or *integer-constant* is 0, then the cursor is positioned at the position before the first row and the warning SQLCODE +100, SQLSTATE 02000 is returned.

If the value of the *host-variable* or *integer-constant* is greater than the count of rows in the result table, the cursor is positioned after the last row in the result table, and the warning SQLCODE +100, SQLSTATE 02000 is returned.

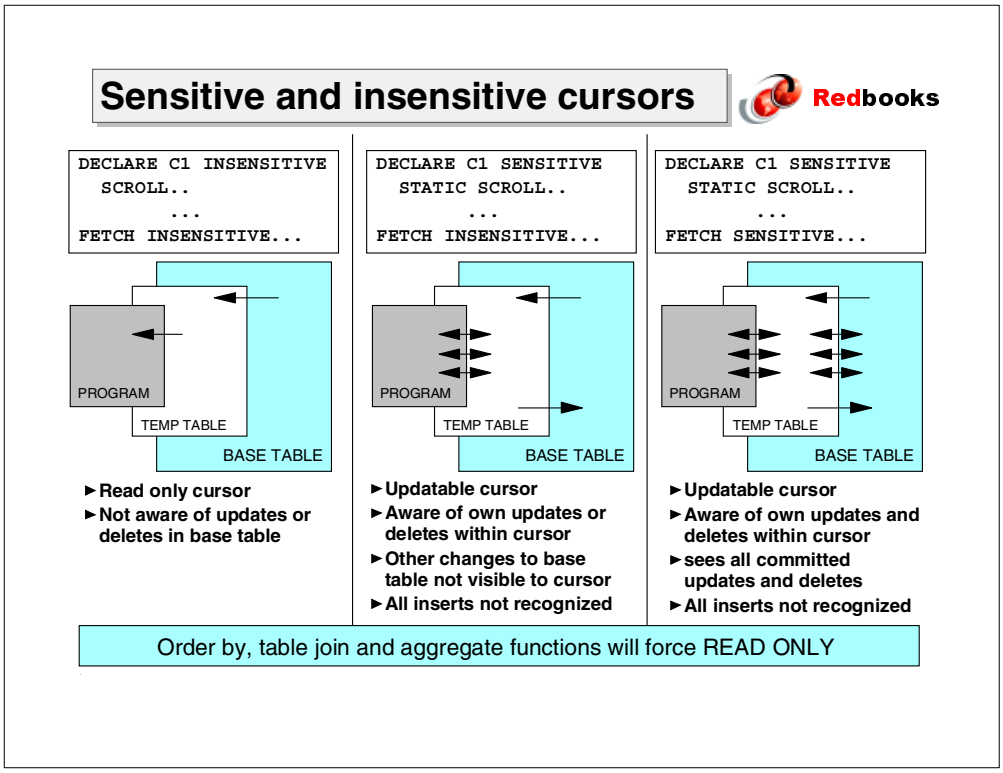
**RELATIVE** Used with either a *host-variable* or *integer-constant.* This keyword evaluates the *host-variable* or *integer-constant* and fetches the data in the row which is that value away from the current cursor position.

If the value in the *host-variable* or *integer-constant* is equal to 0, then the current cursor position is maintained and the data fetched.

If the value in the *host-variable* or *integer-constant* is less than 0, then the cursor is positioned the number of rows specified in the *host-variable* or *integer-constant* from the cursor position towards the beginning of the result table and the data fetched.

If the value in the *host-variable* or *integer-constant* is greater than 0, then the cursor is positioned the number of rows specified in the *host-variable* or *integer-constant* from the cursor position towards the end of the result table and the data fetched.

If a relative position is specified that is before the first row or after the last row, a warning SQLCODE +100, SQLSTATE 02000 is returned, and the cursor is positioned either before the first row or after the last row.




### 2.3.6 Insensitive and sensitive cursors

As we saw above, a significant problem with previous methods of cursor management was in the maintaining of the relationship between the data being updated by the cursor and the actual data in the base table.

V7 of DB2 introduces new keywords INSENSITIVE and SENSITIVE STATIC to control whether the data in the result set is maintained with the actual rows in the base table. DB2 will now ensure that only the current values of the base table are updated and will recognize where rows have been deleted from the result set. It can, if required, refresh the rows in the result set at fetch time to ensure that the data under the cursor is current.

Basically, INSENSITIVE means that the cursor is read-only and is not interested in changes made to the base data once the cursor is opened. With SENSITIVE, the cursor is interested in changes which may be made after the cursor is opened. The levels of this awareness are dictated by the combination of SENSITIVE STATIC in the DECLARE CURSOR statement and whether INSENSITIVE or SENSITIVE is defined in the FETCH statement.

INSENSITIVE cursors are strictly read-only. Updatable SELECT statements must use SENSITIVE STATIC cursors in order to be updatable scrollable cursors.

If an attempt is made to code the FOR UPDATE OF clause in a cursor defined as INSENSITIVE, then the bind will return the SQLCODE:

```
-228 FOR UPDATE CLAUSE SPECIFIED FOR READ-ONLY SCROLLABE CURSOR
USING cursor-name.
```

Fundamentally, the SENSITIVE STATIC cursor is updatable. As such, the FOR UPDATE OF can be coded for a SENSITIVE cursor.

If the SELECT statement connected to a cursor declared as SENSITIVE STATIC uses any keywords that forces the cursor to be read-only, the bind will reject the cursor declaration. In this case the bind will return the SQLCODE:

```
-243 SENSITIVE CURSOR cursor-name CANNOT BE DEFINED FOR THE
SPECIFIED SELECT STATEMENT.
```

Use of aggregate functions, such as MAX and AVG, table joins, and the ORDER BY clause will force a scrollable cursor into implicit read-only mode and therefore are not valid for a SENSITIVE cursor.

A SENSITIVE cursor can be made explicitly read-only by including FOR FETCH ONLY in the DECLARE CURSOR statement. Even if a SENSITIVE cursor is read-only, it will still be aware of all changes made to the base table data through updates and deletes.

When creating a scrolling cursor the INSENSITIVE or SENSITIVE STATIC, keywords must be used in the DECLARE CURSOR statement. This sets the default behavior of the cursor.

There is also the facility within the related FETCH statements to further specify the way in which the cursor will interact with data in the base table. This is done by specifying INSENSITIVE or SENSITIVE in the statement itself. If these keywords are not used in the FETCH statement, then the attributes of the DECLARE CURSOR statement are used. For example, suppose the DECLARE CURSOR is coded as:

```
DECLARE CUR1 SENSITIVE STATIC SCROLL CURSOR FOR
  SELECT ACCOUNT, ACCOUNT_NAME
  FROM PAOLOR2.ACCOUNT
  FOR UPDATE OF ACCOUNT_NAME;
```

and the FETCH CURSOR is defined as:

```
FETCH CUR1 INTO :hvaccount, :hvacct_name;
```

In this case, the cursor will use the SENSITIVE characteristics.

The combinations that are available and the characteristics of these attributes are:

- **CURSOR INSENSITIVE and FETCH INSENSITIVE**

  A DB2 temporary table is created and filled with rows which match the selection criteria.

  The resultant cursor is read-only. If a FOR UPDATE OF clause is coded in the DECLARE CURSOR SELECT statement, then the following SQL code is returned at bind time:

  ```
  -228 FOR UPDATE CLAUSE SPECIFIED FOR READ-ONLY SCROLLABE CURSOR USING
  cursor-name
  ```

If the cursor has been defined as INSENSITIVE and the FOR UPDATE OF CURSOR clause is coded in the FETCH statement, then the following SQL code is returned at bind time and the bind fails:

```
-510 THE TABLE DESIGNATED BY THE CURSOR OF THE UPDATE OR DELETE STATEMENT
CANNOT BE MODIFIED
```

Once the cursor has been opened, if updates or deletes are made to the rows returned by the SELECT statement, these changes are not visible to the cursor.

- **CURSOR INSENSITVE and FETCH SENSITIVE**

  This is not a valid combination. If a FETCH SENSITIVE statement is coded following for an INSENSITIVE cursor, then the following SQL code is returned at runtime:

  -224 SENSITIVITY sensitivity SPECIFIED ON THE FETCH IS NOT VALID FOR CURSOR *cursor-name*

- **CURSOR SENSITIVE STATIC and FETCH INSENSITIVE**

  A temporary table is created at open cursor time with all rows that match the select criteria.

  Updates and deletes can be made against the cursor using the WHERE CURRENT OF CURSOR clause. If an update is to be made against the cursor, the FOR UPDATE OF must be coded in the DECLARE CURSOR statement.

  However, if the FETCH INSENSITIVE is used, the rows returned will only reflect changes made by the current cursor. Changes made by agents outside the cursor will not be visible to data returned by the FETCH INSENSITIVE.

- **CURSOR SENSITIVE STATIC and FETCH SENSITIVE**

  A temporary table is created at open cursor time to hold all rows returned by the SELECT statement.

  Updates and deletes can be made using the WHERE CURRENT OF CURSOR clause. If an update is to be made against the cursor, the FOR UPDATE OF must be coded in the DECLARE CURSOR statement.

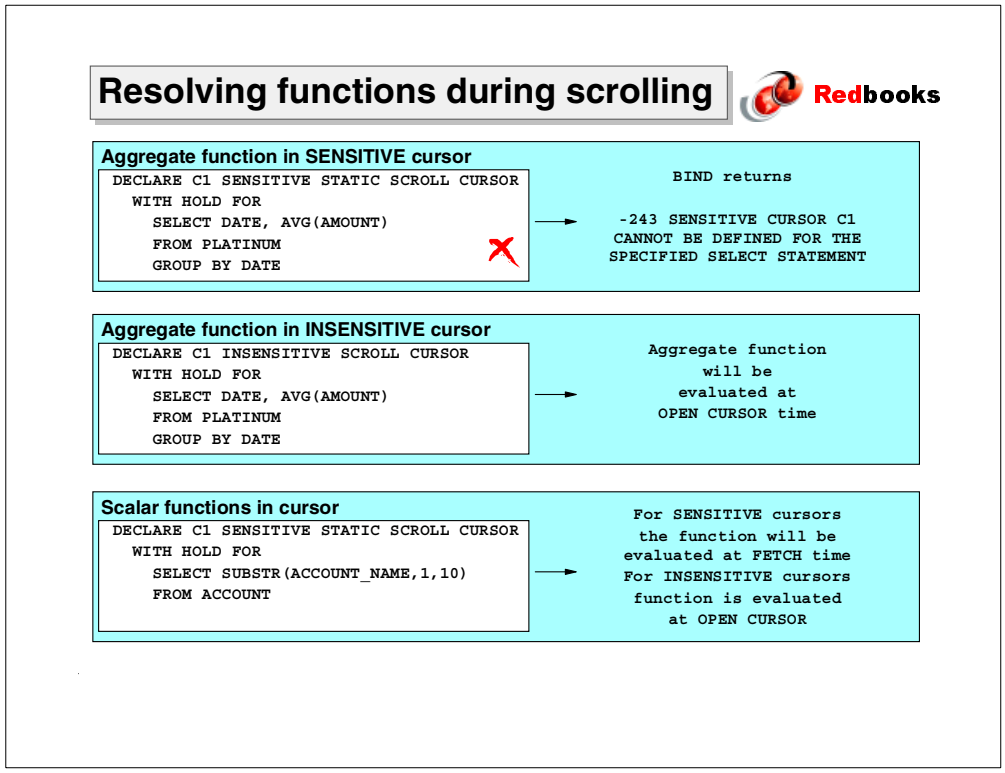
  The FETCH statement will return all updates or deletes made by the cursor and all committed updates and deletes made to the rows within the cursor's result set by all others.

#### 2.3.6.1 Setting the SQLWARN flags

For the scrollable cursor, new values will be returned at OPEN CURSOR time to signify the type of cursor open and whether the cursor is read-only or otherwise. These values are not set for non-scrollable cursors.

**SQLWARN4**
This flag will be set to 'I' if the cursor is INSENSITIVE or 'S' if it is SENSITIVE STATIC.

- **SQLWARN5**
  SQLWARN5 will be set to 1 if the cursor is read-only as a result of the contents of the SELECT statement being read-only or if the cursor is declared with the clause FOR FETCH ONLY; 2 if reads and deletes are allowed on the result set of the cursor but updates are not allowed; or 4 if the cursor result set is both updatable and deletable.




### 2.3.7 Resolving functions during scrolling

There are two types of system functions used in DB2. These are scalar functions which operate on row level effecting the values in a column of that row; or aggregate functions, which use the values from many rows to return a single value.

An example of a scalar function is the SUBSTR which returns a selected length string of characters from the specified column for each row returned by a select. Only if an individual row value changes will the value returned by this function change.

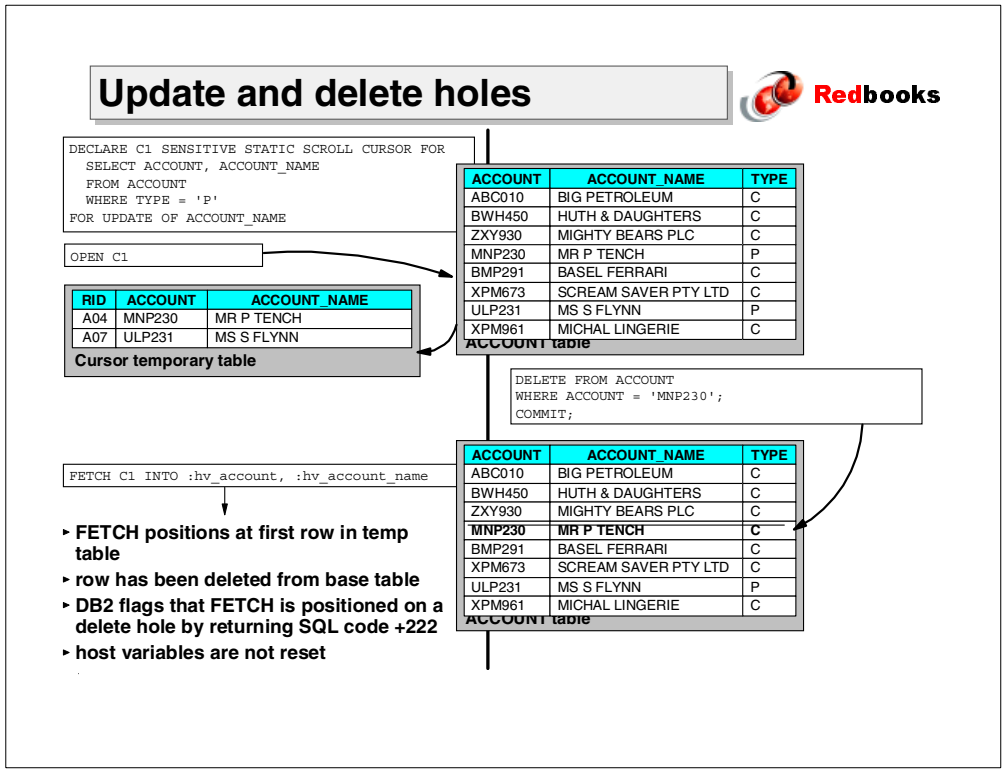
The AVG function is an aggregate function, as it will return the average value for a number of rows. If any of the values of any row in this set changes the value of the average will change.

The basic rule for using functions in a scrollable cursor is that if the aggregate function is part of the predicate, such as:

```
SELECT EMP, SALARY FROM EMP_TABLE
WHERE SALARY > AVG(SALARY + BONUS)
```

Then the value is frozen at the OPEN CURSOR.

However, with the scalar function, DB2 can maintain the relationship between the temporary result set and the rows in the base table, and therefore allows these functions to be used in both INSENSITIVE and SENSITIVE cursors. If used in an INSENSITIVE cursor, the function is evaluated once at OPEN CURSOR time. For SENSITIVE cursors and where a FETCH SENSITIVE is used, this function will be evaluated at FETCH time. For SENSITIVE cursors with an INSENSITVE FETCH the function is evaluated at FETCH time only against the result set for the cursor. The function will not be evaluated against the base table.




## 2.3.8 Update and delete holes

The above diagram shows what happens when sensitive cursors fetch a row which has be deleted from the base table by another agent. When a FETCH positions on such a row, the row itself is referred to as a *delete hole*. This can also occur if the cursor itself has deleted a row that was part of the result sets returned at OPEN CURSOR time.

A similar situation occurs when a row that was returned in the initial result set was updated in such a way as to make it no longer valid due to be included in the result set by the WHERE conditions of the SELECT statement. This is called an *update hole*. An example of the occurrence of an update is:

```
DECLARE C1 SENSITIVE STATIC SCROLL CURSOR FOR
  SELECT ACCOUNT, ACCOUNT_NAME
  FROM ACCOUNT
  WHERE TYPE = 'P'
FOR UPDATE OF ACCOUNT_NAME;
```

The OPEN CURSOR is executed and the DB2 temporary table is built with the same two rows as in the diagram above.

Another user executes the statement:

```
UPDATE ACCOUNT
SET TYPE = 'C'
WHERE ACCOUNT = 'MNP230';
COMMIT;
```

Here, it can be seen that the row for account not longer fulfils that requirements of the WHERE clause of the DECLARE CURSOR statement.

The process executes its first FETCH:

```
FETCH C1 INTO :hv_account, hv_account_name;
```

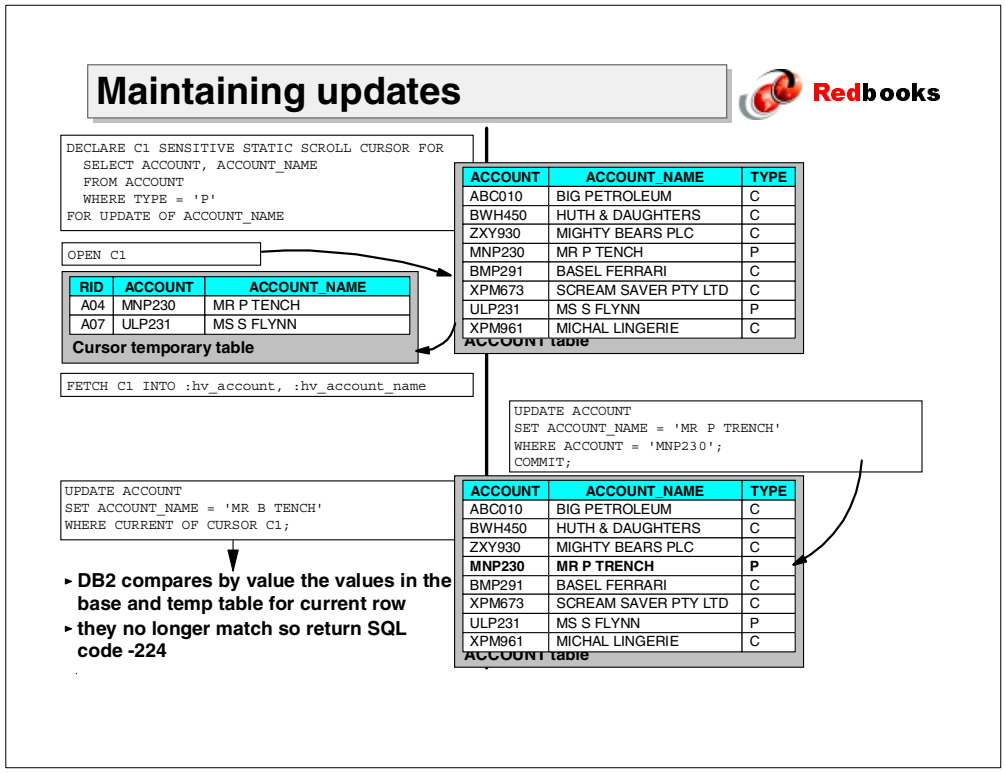
DB2 will verify that the row is valid by executing a SELECT with the WHERE values used in the initial open against the base table. If the row now falls outside the SELECT, DB2 returns the SQL code: +223: UPDATE HOLE DETECTED USING *cursor-name* to highlight the fact that the current cursor position is over an update hole.

At this stage, the host variables will be empty; however, it is important to recognize the hole, as DB2 does not reset the host variables if a hole is encountered.

If the FETCH is executed again, the cursor will be positioned on the next row, which in the example is for ACCOUNT ULP231. The host variables will now contain ‘ULP231’ and ‘MS S FLYNN’.

It is important to note that if an INSENSITIVE fetch is used, then only update and delete holes created under the current open cursor are recognized. Updates and deletes made by other processes will not be recognized by the INSENSITIVE fetch.

If the above SENSITIVE fetch was replaced with an INSENSITIVE fetch, the fetch would return a zero SQLCODE, as the update to the base row was made by another process. The column values would be set to those at the time of the OPEN CURSOR statement execution.




## 2.3.9 Maintaining updates

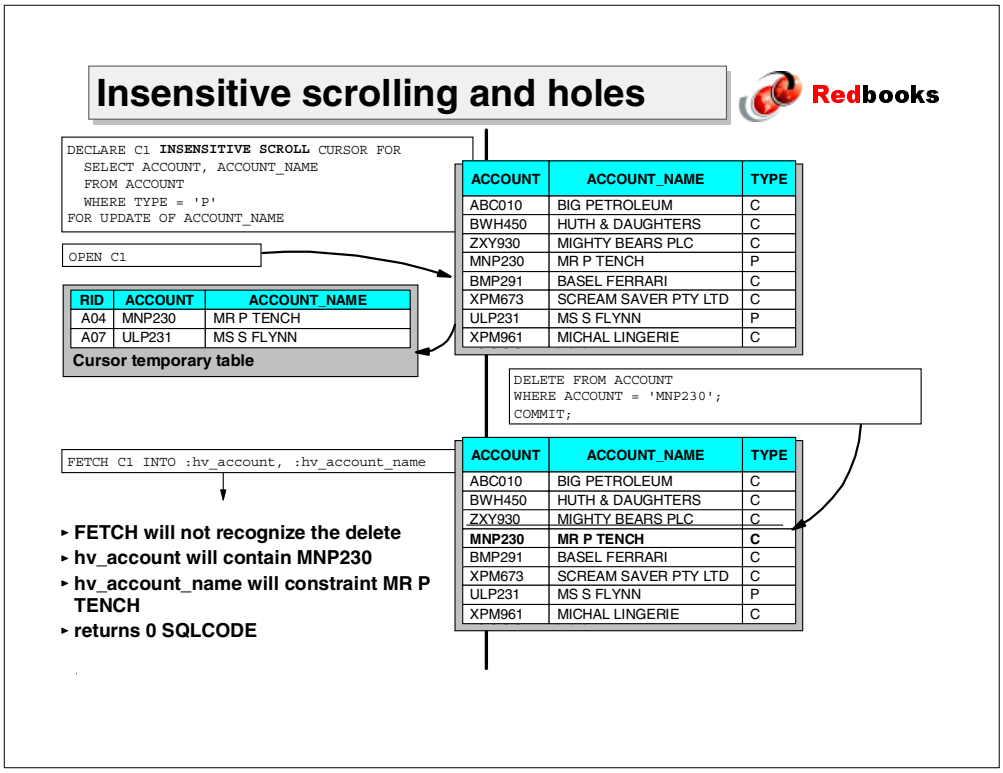
In previous versions of DB2, programs had to provide a means by which values in the current row fetched were still valid. Normally, this involved executing a SELECT using the primary key of the table to be updated and checking each column to ensure that the values matched.

DB2 now maintains a relationship between the rows returned by the scrolling cursor and those in the base table. If an attempt is made to UPDATE or DELETE the currently fetched row, DB2 goes to the base table, using the RID, and verifies that the columns match by value. If columns are found to have been updated, then DB2 returns the SQL code:

```
-224: THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING cursor-name.
```

When you receive this return code, you can choose to refetch the new data by using the FETCH CURRENT to retrieve the new values. The program can then choose to reapply the changes or not.

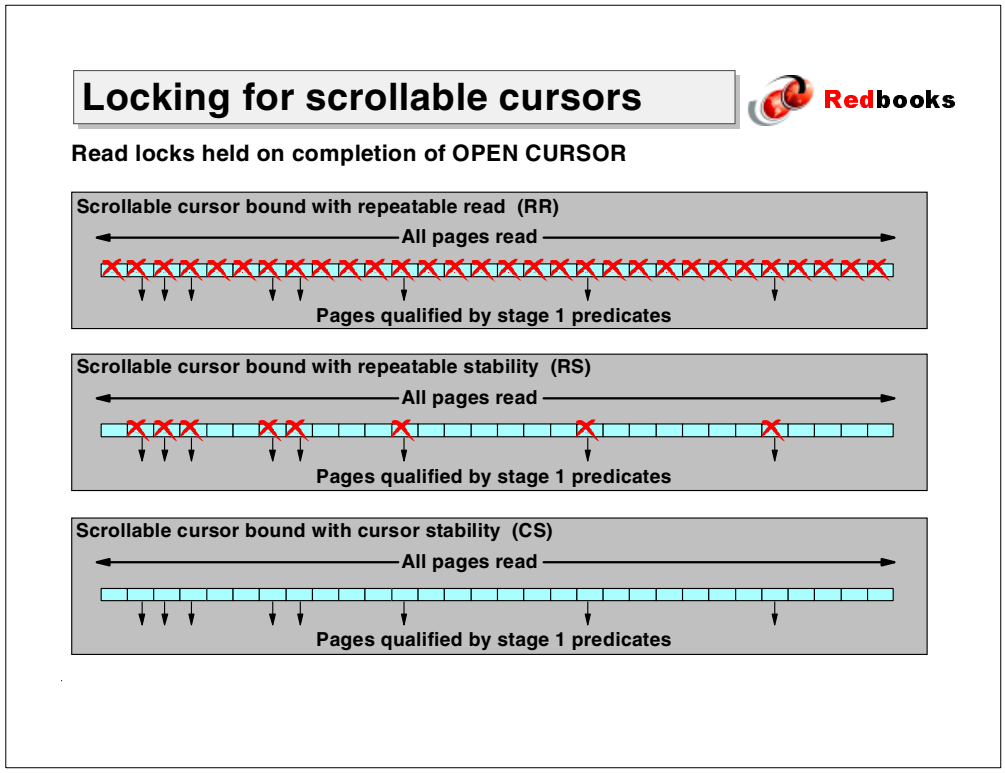
It should be noted that the new cursor will never see any rows that have been inserted to the base table and which fit the selection criteria of the DECLARE CURSOR statement.




## 2.3.10 Insensitive scrolling and holes

When a cursor is defined as INSENSITIVE, the cursor is read-only, and once rows are read, they never reference the rows in the base table. As such, if an update is made to a row in the base table that would create an update or delete hole in the cursor's result set, this cursor will not recognize it. A fetch against a row in the result set whose matching base table row has been deleted or updated will still return an SQLCODE of 0 and return the row values to the host variables as they were set at OPEN CURSOR time.

The use of the FETCH INSENSITIVE statement within a SENSITIVE cursor is more complex. Remember that updates and deletes can be made in a SENSITIVE cursor using the clause WHERE CURRENT OF in UPDATE and DELETE statements. However, when the INSENSITIVE keyword is used in the FETCH statement, we are saying that we do not want the cursor to reflect updates to the rows of the cursors made statements outside the cursor. In this case, the FETCH statement will show holes made by updates and deletes from within the current cursor, but will not show any holes created by any update and deletes outside the cursor.




## 2.3.11 Locking for scrollable cursors

Locking, under a scrollable cursor, behaves exactly the same as before for the different isolation levels and CURRENTDATA settings. During the operation of the OPEN CURSOR which is creating a temporary result set, the following locks will be taken.

### 2.3.11.1 Scrollable cursor bound with repeatable read (RR)

The cursor will keep a lock on all pages or rows read, whether the page being read has a row that will appear in the result set or not.

### 2.3.11.2 Scrollable cursor bound with read stability (RS)

This cursor will lock all pages which contain a row that qualifies with a stage 1 predicate.

### 2.3.11.3 Scrollable cursor bound with cursor stability (CS)

When reading rows during execution of the OPEN CURSOR statement with the CURRENTDATA setting of YES, a lock will be taken on the last page or row read.

If CURRENTDATA is set to NO, no locks will be taken except where the cursor has been declared with the FOR UPDATE OF clause. When this clause is specified, a lock is taken for the last page or row read. In this case, only committed data will be returned.

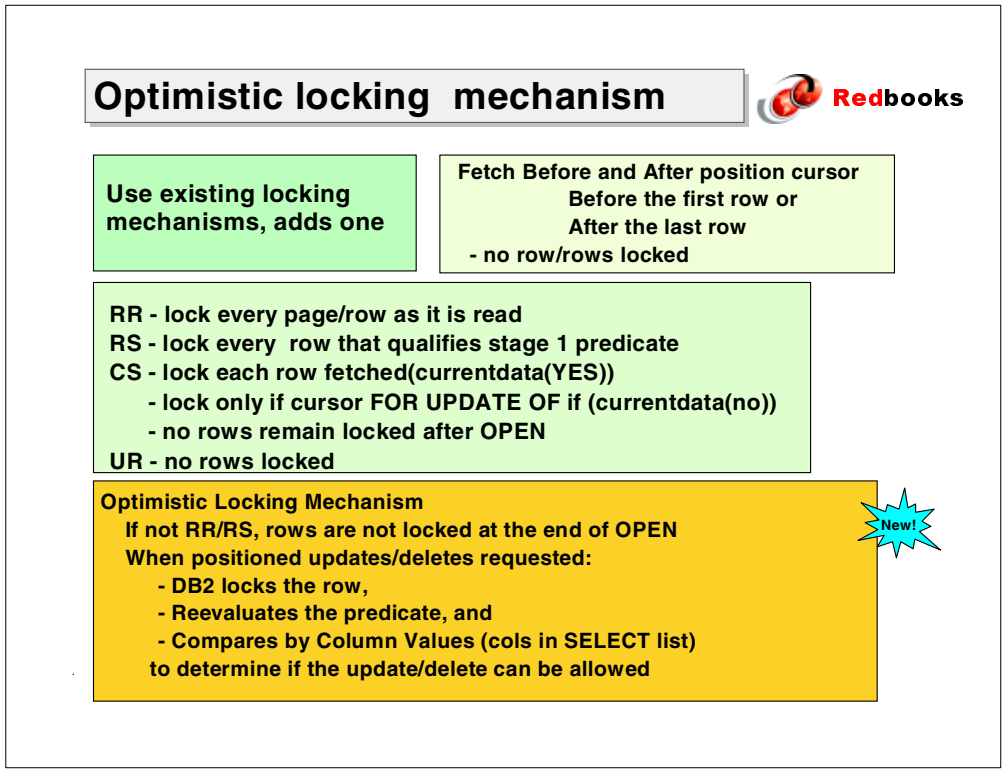
When cursor stability is specified, on completion of the OPEN CURSOR, all locks are released.

#### 2.3.11.4 Scrollable cursor bound with uncommitted read (UR)

A cursor that has been bound with uncommitted read (UR) will not take any locks, and will not check whether a row has been committed or not when selecting it for inclusion in the temporary result set.

#### 2.3.11.5 Duration of locks

Locks retained under repeatable read and read stability will be retained until commit time or where cursors are defined WITH HOLD, until the first commit after the cursor is closed. Positioned update or delete locks will be held for the same duration.




## 2.3.12 Optimistic locking

The existing locking mechanism applies, as usual, as defined by lock size and isolation level. The RR, RS, CS, UR parameters have the same implications as always. When positioning the cursor, no rows are locked after OPEN CURSOR.

### 2.3.12.1 Optimistic locking characteristics

Applications have used this technique forever; now optimistic locking is done in the DBMS, which means they may not have locks on the page or row:

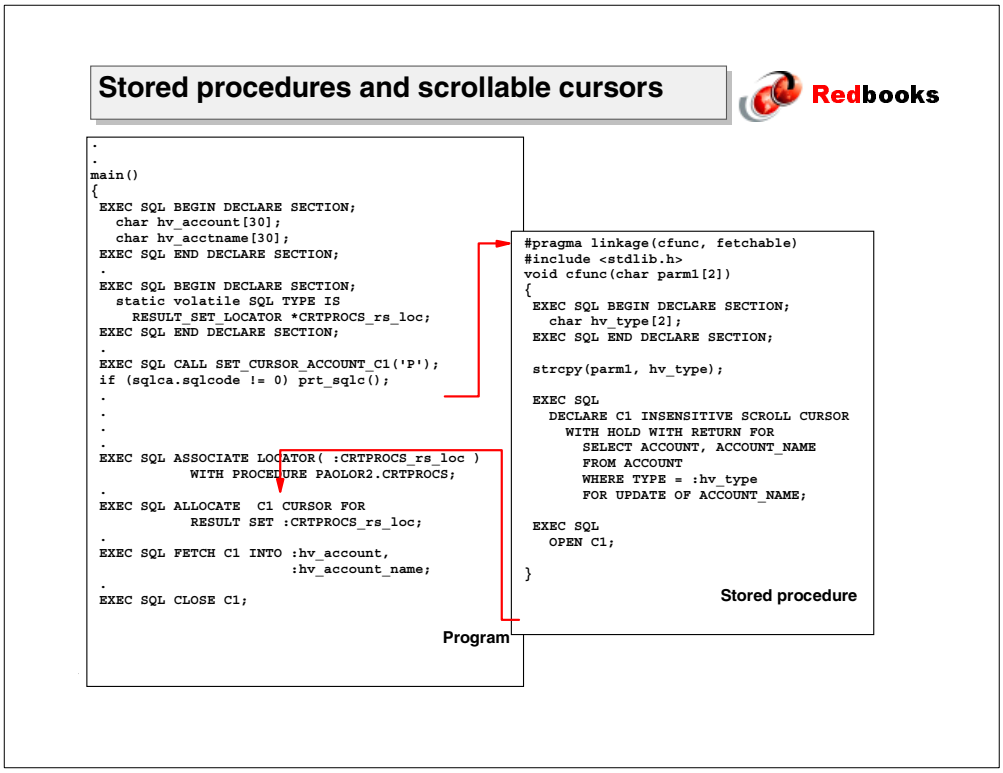
- Lock
- Evaluate, match
- Update/delete if row passes evaluation and column values match.

Table 1 summarizes the benefits of optimistic locking, such as the improved concurrency achieved by DB2's implementation of this feature:

*Table 1. Standard locking versus optimistic locking*

| **Regular forward only cursor scenario** | **Scrollable cursor scenario** | **Comments** |
|---|---|---|
| BIND with<br>ISOLATION LEVE(CS)<br>CURRENT DATA(YES) | BIND with<br>ISOLATION LEVE(CS)<br>CURRENT DATA(NO) | Regular cursor must effectively use CURRENT DATA(YES) if the cursor is declared FOR UDATED OF, whereas scrollable cursor does not need to use CURRENTDATA(YES).<br><br>There is no benefit with ISO(RR) or ISO(RS), because semantics request locking specifically. ISO(UR) is not considered because practically we need to compare using committed data. |
| DECLARE CURSOR C1<br><br>`SELECT NAME, DEPT`<br>`...`<br>`FOR UPDATE OF DEPT` | DECLARE CURSOR C2<br>SENSITIVE STATIC SCROLL<br>`SELECT NAME, DEPT`<br>`...`<br>`FOR UPDATE OF DEPT` | |
| OPEN C1 | OPEN C2<br><br>FETCH from base table<br>Populate result table<br>Release locks | No locks held by either cursor after end of OPEN. |
| FETCH C1<br>Acquires lock and holds it | FETCH SENSITIVE C2<br>Acquires lock and releases after FETCH | Here, the scrollable cursor releases lock after fetch and the regular does not. |
| Application processing<br>...<br>...<br>...<br>Low concurrency during this time since lock held | Application processing<br>...<br>...<br>...<br>High concurrency during this time since lock not held | Higher concurrency with optimistic locking because no lock held, optimistically released expecting no updates to the fetched row by others, and possibly no positioned update by this cursor. |
| UPDATE WHERE CURRENT OF C1<br><br>Update is performed right away | UPDATE WHERE CURRENT OF C2<br><br>Acquire lock<br>Revaluate row<br>Compare columns by values:<br>If<br>row evaluates and no values differ<br>Update base table<br>Update result table<br>Else<br>Return unsuccessful SQLCODE | Scrollable cursor achieves data integrity by reevaluating and comparing columns by value under a lock to ensure data integrity. |
| FETCH C1<br>Acquires lock and holds it | FETCH SENSITIVE C2<br>Acquires lock and releases after FETCH | Scrollable cursor releases lock after fetch. |

| Regular forward only cursor scenario | Scrollable cursor scenario | Comments |
|---|---|---|
| Application processing<br>...<br>...<br>...<br>Low concurrency during this time since lock held | Application processing<br>...<br>...<br>...<br>High concurrency during this time since lock not held | Higher concurrency with Optimistic locking because no lock held, optimistically released expecting no updates to the fetched row by others and possibly no positioned update by this cursor. |
| FETCH C1<br>Unlocks previous row<br>Acquires lock and holds it | FETCH SENSITIVE C2<br>Acquires lock and releases after FETCH | Scrollable cursor does not hold on to the previous row lock so allows for higher concurrency. |
| CLOSE C1 | CLOSE C2 | All locks released. |

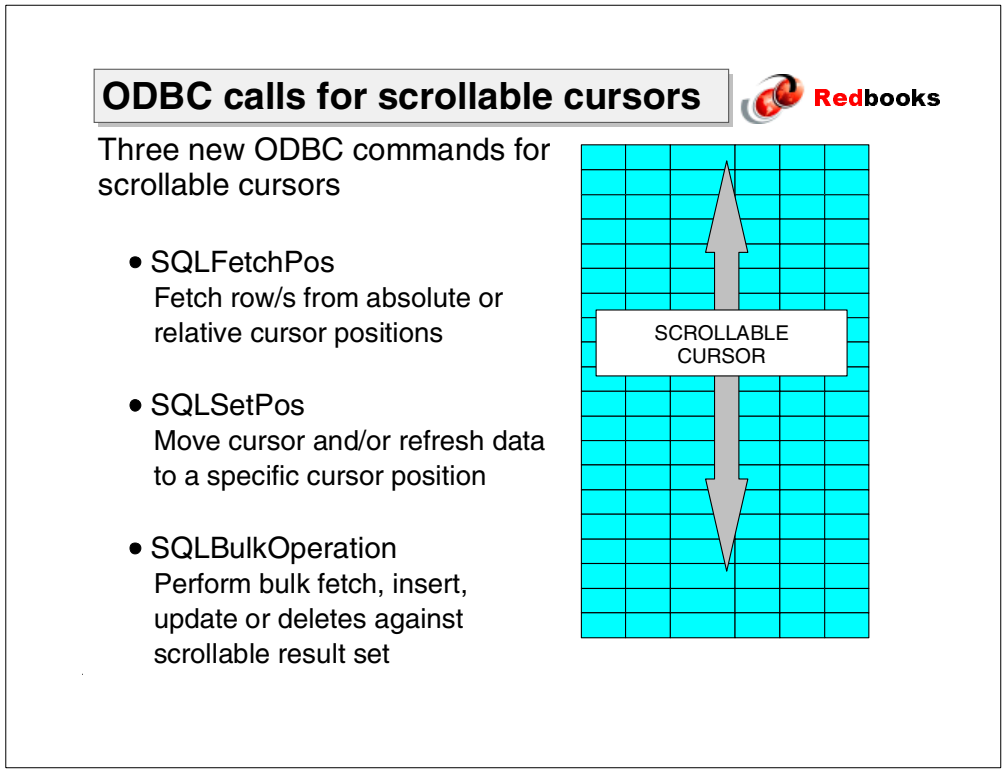



## 2.3.13 Stored procedures and scrollable cursors

Stored procedures can be used to define scrollable cursors. For this, these rules must be followed:

- On leaving the stored procedure, the cursor should be pointing at the position prior to the beginning of the table.
- On return, the calling program then allocates the cursor and executes the fetches.

All stored procedure defined cursors will be read-only, as is the case for non-scrollable cursors. However, these cursors can still make use of both INSENSITIVE and SENSITIVE style cursors.




## 2.3.14 ODBC calls for scrollable cursors

To allow for scrollable cursors, three new calls have been added to ODBC for OS/390. These calls are described below.

### 2.3.14.1 SQLFetchScroll

This command fetches the specified number of rows from the result set of a query and returns the values for each bound column. The command allows for the cursor to be moved to a relative or absolute position within the result set.

```
SQLRETURN SQLFetchScroll (SQLHSTMT       StatementHandle,
                          SQLUSMALLINT   FetchOrientation,
                          SQLINTEGER     FetchOffset);
```

The statement uses the FetchOrientation and FetchOffset values to decide where the cursor is be positioned within the result set. The command can return multiple rows from a single execution. The number of rows to be attempted to return is set in the SQL_ATTR_ROW_ARRAY_SIZE statement attribute.

The possible values of FetchOrientation parameter are:

| | |
|---|---|
| **SQL_FETCH_NEXT** | Returns the set number of rows following the current cursor position |
| **SQL_FETCH_PRIOR** | Returns the set number of rows starting at the row proceeding the current cursor position |
| **SQL_FETCH_RELATIVE** | Returns the set number of rows starting at FetchOffset from the current cursor position |

| | |
|---|---|
| **SQL_FETCH_ABSOLUTE** | Returns the set number of rows from position FetchOffset from the start of the result set |
| **SQL_FETCH_FIRST** | Returns the set number of rows starting with the first row in the result set |
| **SQL_FETCH_LAST** | Returns the set number of rows before the last row in the result set |
| **SQL_FETCH_BOOKMARK** | Returns the set number of rows FetchOffset rows from the bookmark specified by the SQL_ATTR_FETCH_BOOKMARK_PTR statement attribute |

On completion of this command, the cursor will be positioned on the first row returned by the call.

#### 2.3.14.2 SQLSetPos

This command can be used to position the cursor on a specific row within the result set. It can also be used to refresh the data at this position or to update the values within that row or to delete the entire row.

The format of the command is:

```
SQLRETURN SQLSetPos (SQLHSTMT       StatementHandle,
                     SQLUSMALLINT    RowNumber,
                     SQLUSMALLINT    Operation,
                     SQLSMALLINT     LockType);
```

The RowNumber value is set the absolute position within the result set with 1 representing the first row. If 0 is used the Operation parameter is used across the whole result set.

The Operation parameter can have the values:

| | |
|---|---|
| **SQL_POSITION** | DB2 will position the cursor at the RowNumber position. No refreshes take place. |
| **SQL_REFRESH** | DB2 positions the cursor on RowNumber row and refreshes the data in the row's buffer. This call will not re-qualify the WHERE clause so that the row remains even when it does not qualify due to updates to the row outside the cursor. |
| **SQL_UPDATE** | DB2 positions the cursor at the RowNumber and updates the row data with values from the row buffer. The SQL_ATTR_ROW_STATUS_PTR is set to SQL_ROW_UPDATED or SQL_ROW_UPDATED_WITH_INFO. |
| **SQL_DELETE** | The cursor is positioned at RowNumber and the underlying row is deleted. The SQL_ATTR_ROW_STATUS_PTR is set to SQL_ROW_DELETED. |

The only value that DB2 ODBC supports for the LockType parameter is SQL_LOCK_TYPE_CHAGE. This retains the locks taken by the DBRM prior to the SQLSetPos call.

#### 2.3.14.3 SQLBulkOperations

This command allows mass operations against the result set of a scrollable cursor query. New rows can be added or mass updates, deletes can be performed where rows in the result set are book marked.

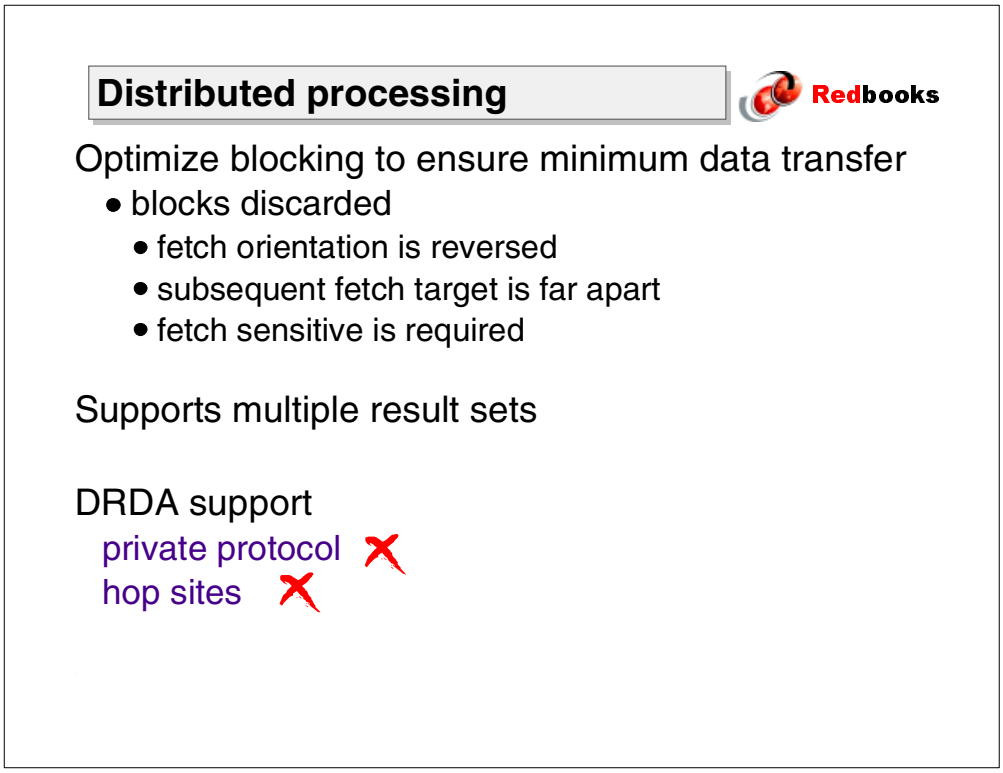
The syntax of the SQLBulkOperations command is:

```
SQLRETURN SQLBulkOperations (SQLHSTMT       StatementHandle,
                             SQLSMALLINT    Operation);
```

The available options for the Operation parameter are:

| | |
|---|---|
| **SQL_ADD** | Inserts rows from bound arrays. Multiple rows can be inserted by setting the SQL_ATTR_ROW_ARRAY_SIZE to the number of rows to be inserted. |
| **SQL_UPDATE_BY_BOOKMARK** | Performs mass update of book marked rows in query result set. |
| **SQL_DELETE_BY_BOOKMARK** | Performs mass delete of book marked rows from query result set. |
| **SQL_FETCH_BY_BOOKMARK** | Retrieves multiple book marked rows from query result set. |

This command should be used to insert rows into a scrollable cursor.

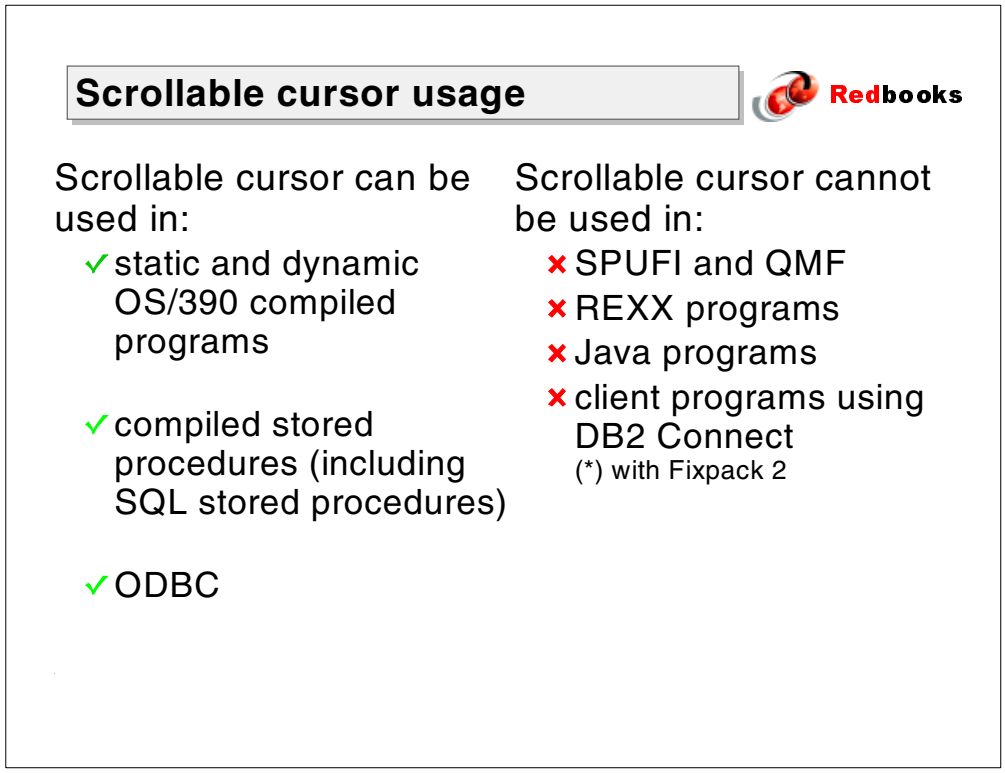



### 2.3.15 Distributed processing

The new SQLWARN4 setting is used to determine insensitive or sensitive.

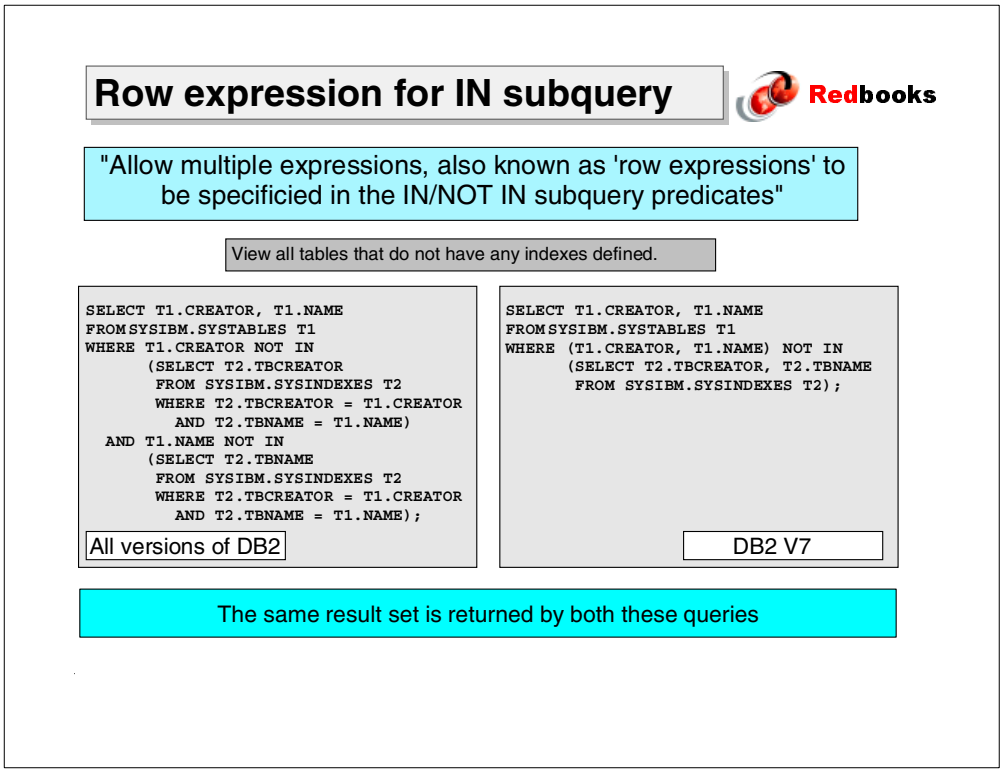
The new SQLWARN5 setting is used to determine read-only or updatable.

Performance is expected to be variable.




### 2.3.16 Scrollable cursor usage

The above diagram shows where scrollable cursors can be coded and used. This list may change prior to GA of DB2 V7, particularly in Java and with DB2 Connect. which requires Fixpack 2.




## 2.4 Row expression for IN subquery

Prior to DB2 V7, SQL only allowed for single column comparisons to be used in the subquery predicate. V7 extends the syntax of the subquery query predicate to allow the use of multiple columns to be specified in the IN and NOT IN subquery predicates.

In addition, the subquery predicate has been expanded further to support equal and not-equal operators. The new operators are:

= ANY (fullselect)

= SOME (fullselect)

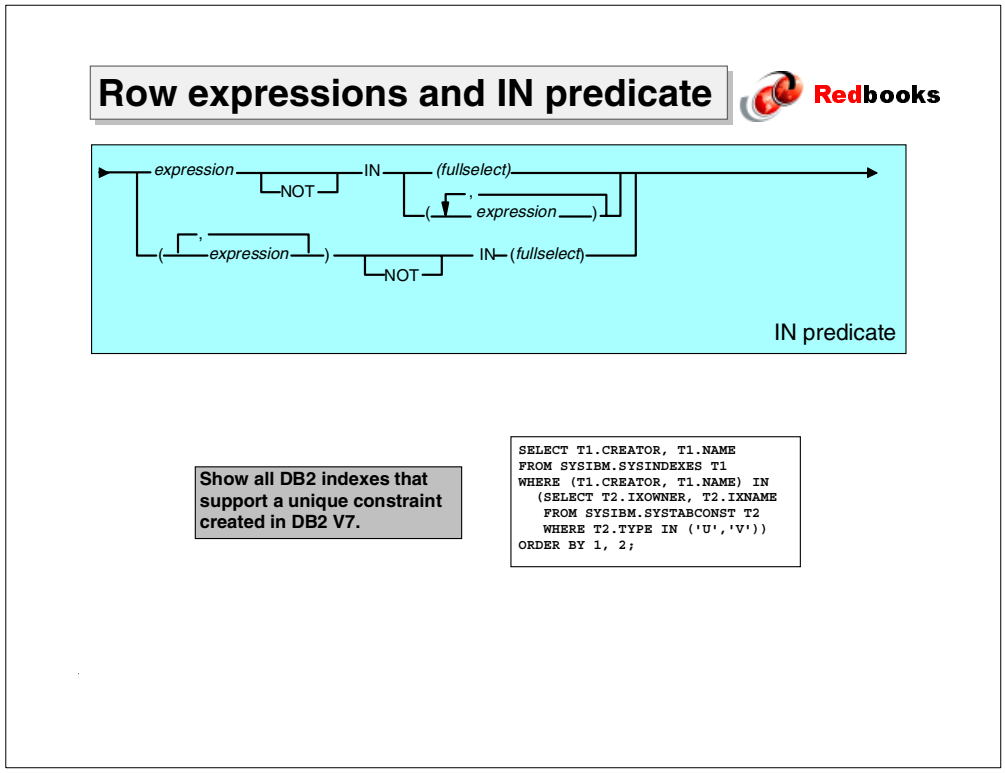
<> ALL (fullselect)

(exp, exp,...) = (exp,exp,....)

(exp, exp,...) <> (exp,exp,...)

Support for the “IN list” has not been extended in DB2 V7. This means that the following syntax is not valid:

```
SELECT ... FROM ...
WHERE (EXPRA1, EXPRA2,EXPRA3) IN ((EXPRB1, EXPRB2, EXPRB3),
                                  (EXPRC1, EXPRC2, EXPRC2));
```




### 2.4.1 Row expressions and IN predicate

The IN predicate now allows for multiple expressions to appear on the left-hand side when a fullselect is specified on the right-hand side. If multiple expressions are coded, then they must be enclosed in brackets. It is important that if a number of expressions are used on the left-hand side of the predicate, then the fullselect must return the same number of columns in its result set.

The predicate is evaluated by matching the first expression value against the first column value from the fullselect result set, the second expression value against the second column value until all expression and column values are compared. The values are AND'd together and only if all columns are matched will the predicate be true.

For example, if we use the statement from the diagram, and the values of the expressions T1.CREATOR and T1.NAME are 'PAOLOR2' and 'XCUST002' respectively, and the result set returned by the fullselect is:

| | |
|---|---|
| PAOLOR2 | XCUST002 |
| PAOLOR2 | XCUST002 |
| XCUST002 | PAOLOR2 |

Then the first statement holds true only if the second row returns a true result. Even though the values of the third row are the same as those of the expression, they are not true, as the first expression value does not match the first column value.

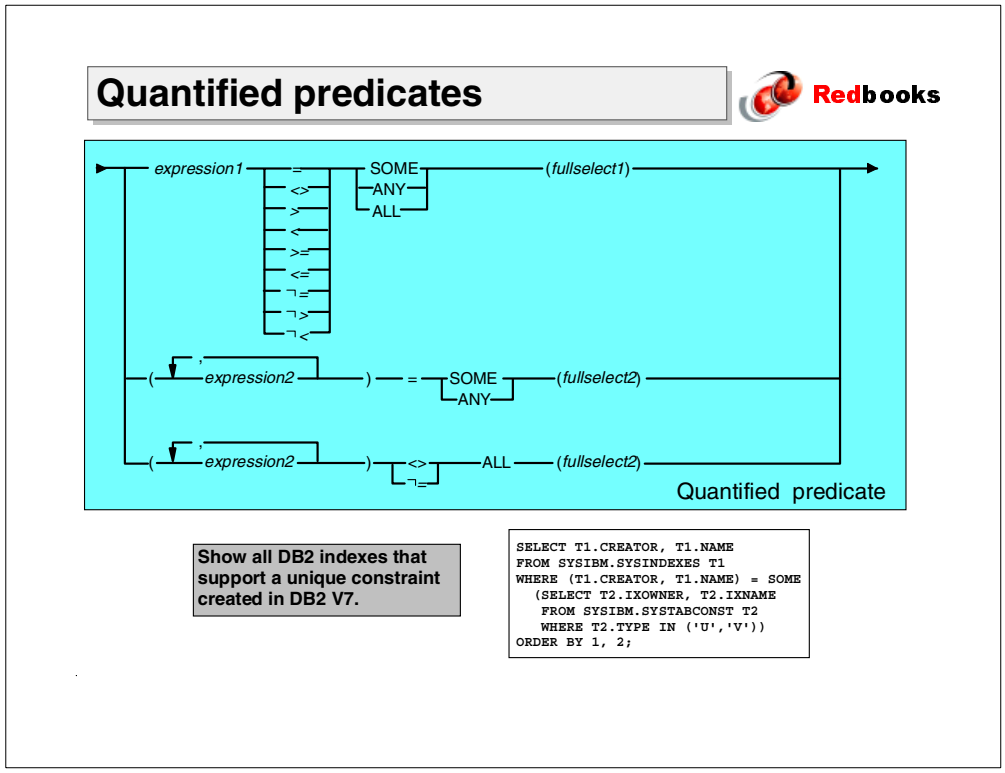
If the number of expressions and the number of columns returned do not match, then the SQLCODE -216 will be returned:
**THE NUMBER OF ELEMENTS ON EACH SIDE OF A PREDICATE OPERATOR DOES NOT MATCH. PREDICATE OPERATOR IS ISIN**.

Also, remember that multiple expressions are not valid in an IN-list. For example, this predicate is valid:

```
WHERE (T1.NAME, T1.CREATOR) IN ('SYSTABLES', 'SYSIBM')
```

However, this predicate is not valid:

```
WHERE (T1.NAME, T1.CREATOR) IN (('SYSIBM','SYSTABLES'),
                                ('SYSIBM','SYSINDEXES'))
```




## 2.4.2 Quantified predicates

A quantified predicate compares a value or values with a collection of values. When expression1 is used in the quantified predicate only a single column can be returned in the result set of the fullselect. If multiple columns are required, then expression2 has to be used.

As well as providing for multiple expressions and columns to be compared, the syntax of the quantified predicate has been expanded to allow the use of the keywords SOME, ANY, and ALL.

### 2.4.2.1 SOME and ANY keywords

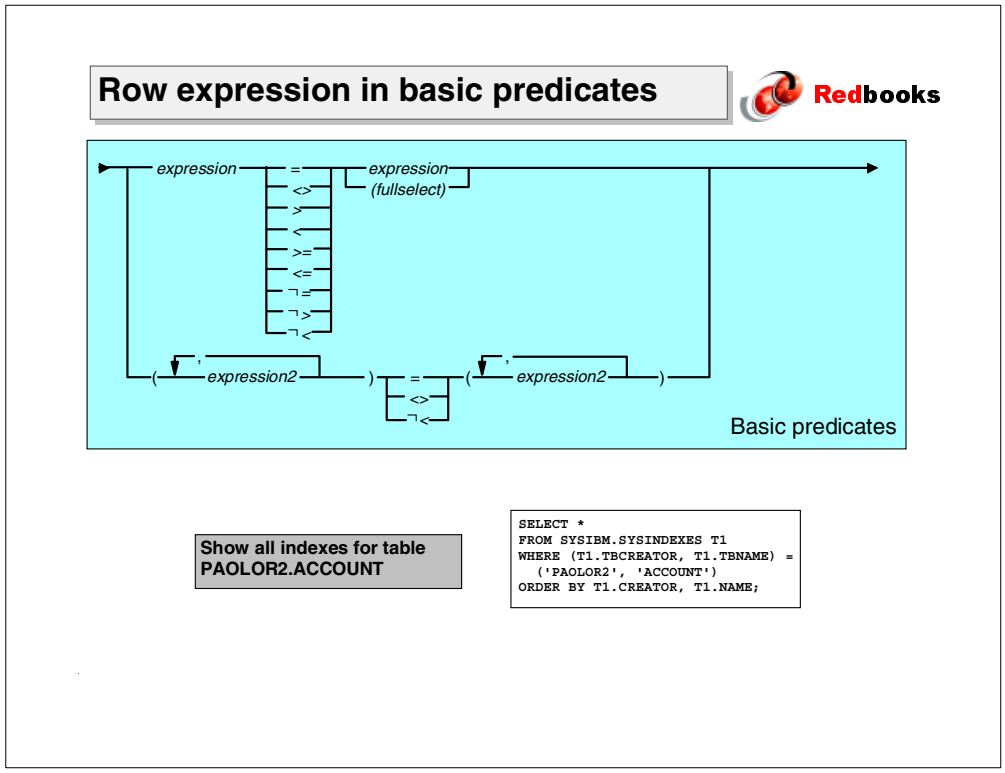
These keywords are used when you wish the predicate to be true if the expression matches any of the matching column values. Therefore, if the expression value for the current row is 1 and the fullselect returns the column values 1, 2 and 3, then the predicate is true for these expression/column values. However, if the column values are 2, 3 and 4, then the predicate is false.

There is a third value that can be returned when using the SOME or ANY keywords. The predicate will return unknown if the result of the expression and column values are untrue for all values and at least one of the comparison values is null. For example, if the expression value is 1 and the returned column values are 2, 3, and NULL, the predicate will return “unknown”.

#### 2.4.2.2 ALL keyword

The ALL keyword performs similarly to the SOME and ANY keywords except that, for the predicate to return a true value, the comparison between the expression and the returned column values must be true for all values. If the expression value is 1 and the column values returned are 1, 1, and 1, then the predicate is held to be true. If the final returned column value is 2, then the predicate is deemed to be false. This keyword returns unknown if all returned known values are true and at least one is null. The predicate will return unknown if the expression value is 1 and the returned column values are 1,1,1 and null.

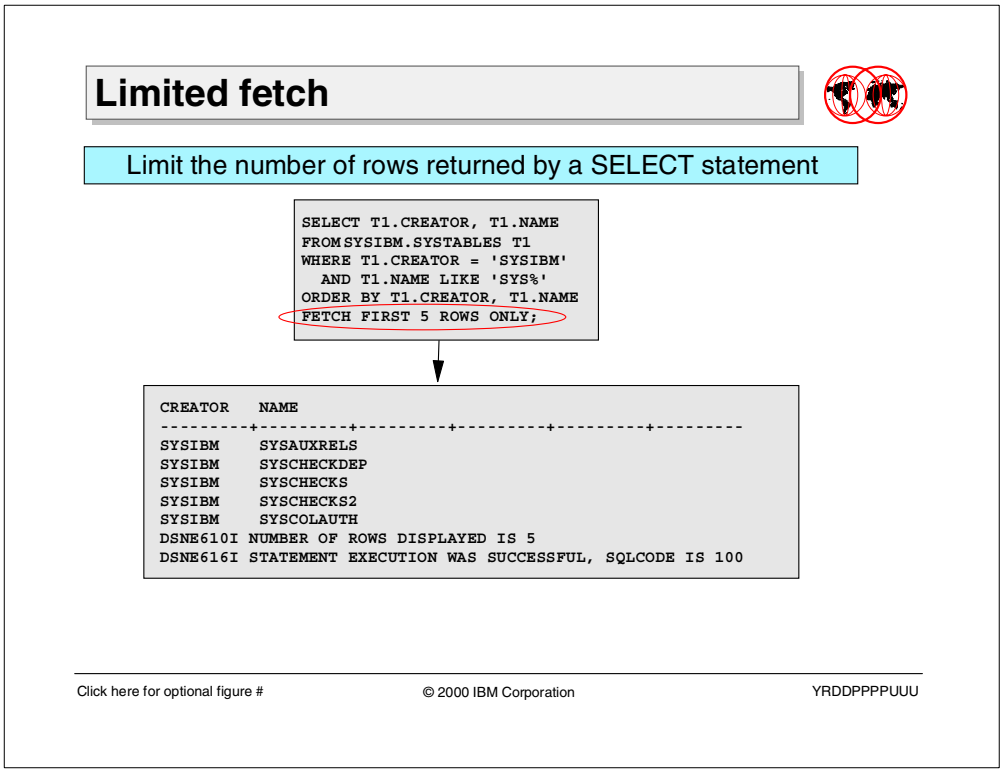
The use of multiple expressions on the left-hand side of the predicate with =SOME or =ANY is analogous to using the IN keyword. This can be seen in the example used in the diagram above. If <>ALL is used, then this is the same as using the NOT IN keywords.




### 2.4.3 Row expression in basic predicates

In the basic predicate, DB2 will only allow the fullselect to be coded for a single value result set. If more than one value is returned or multiple columns are coded in the fullselect, then errors occur.

Multiple expression can be used to compare multiple values in a single predicate. However, as in the example above, the number of expressions on the left-hand side of the operator must be the same as the number of expression on the right-hand of the expression.

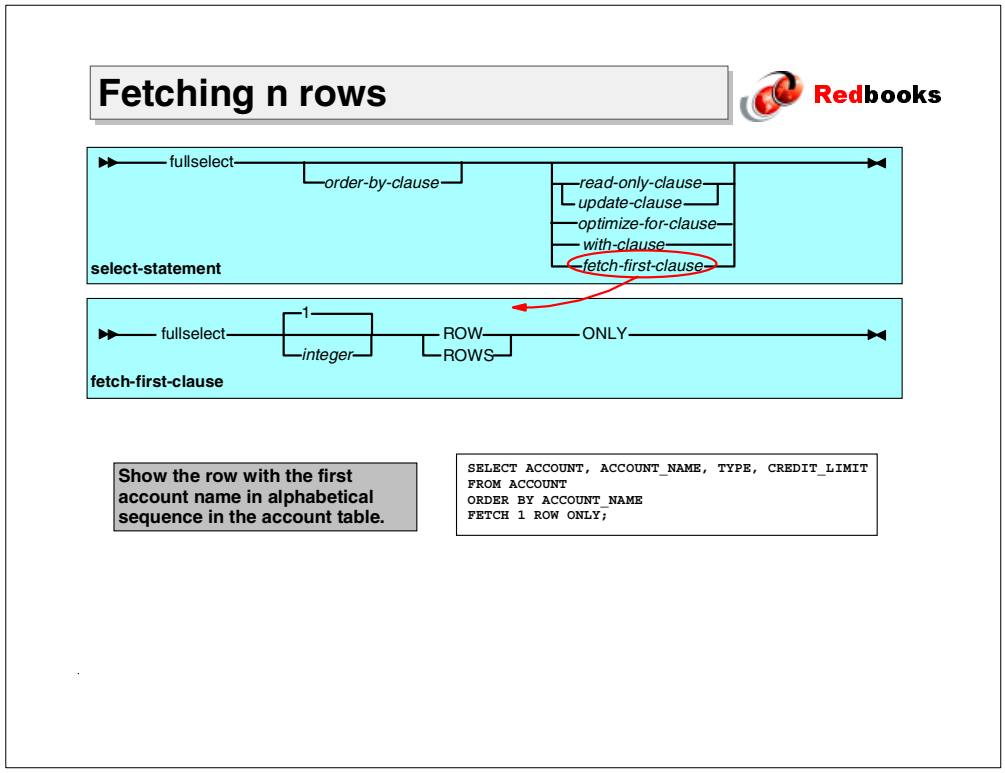



## 2.5 Limited fetch

DB2 V7 introduces the FETCH FIRST n ROWS ONLY, which allows a limit on the number of rows returned to the result set to be specified. This clause has been introduced to support ERP vendor products, many of which require only the first row to be returned. In previous versions of DB2, DB2 would prefetch blocks, the total number of rows of which would be ignored by single row processing. This had a negative impact on the performance of such statements. This enhancement is of particular value for distributed applications, but it is also applicable to local SQL.

If DB2 knows that only one row is required, then block prefetching will not take place and only the specified number of rows will be returned in the result set.

Performance is improved with DRDA applications as well, as the client application can specify limits to the amount of data returned and DRDA will close the cursor implicitly when the limit is met.

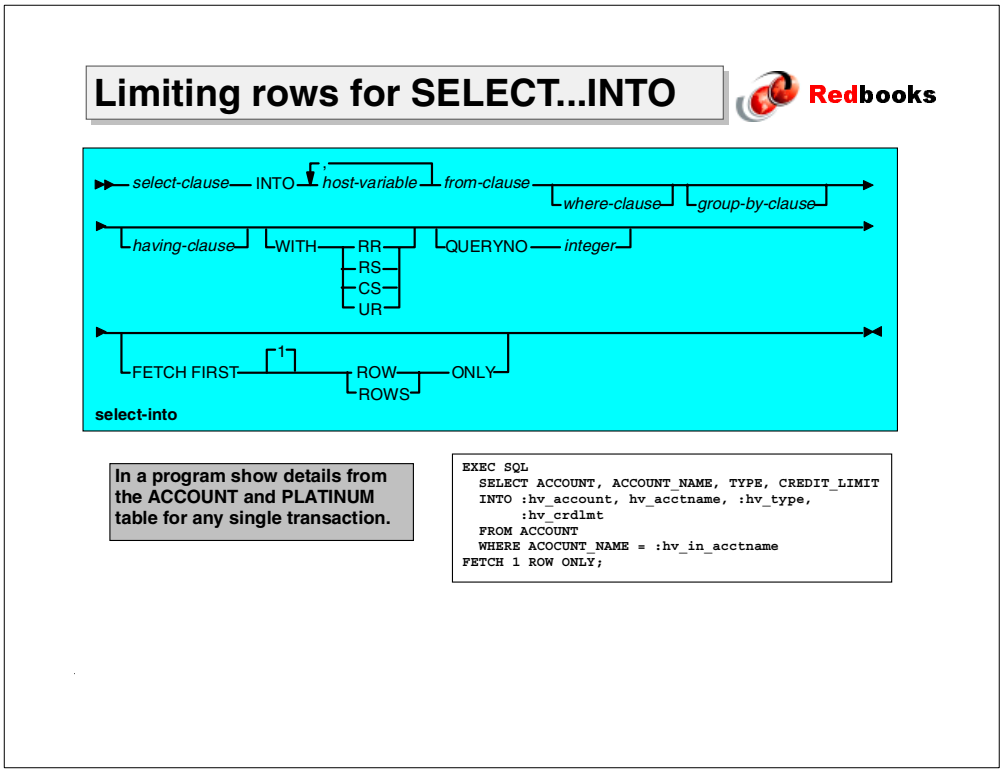



### 2.5.1 Fetching n rows

The above diagram shows the syntax for the new fetch-first-clause. The example shows how it can be used to select rows which have columns with maximum or minimum values by using the ORDER BY and the FETCH FIRST n ROWS clauses together.

In previous versions of DB2, you would have had to code a statement like this to achieve the same result:

```
SELECT T1.ACCOUNT, T1.ACCOUNT_NAME, T1.TYPE, T1.CREDIT_LIMIT
FROM ACCOUNT T1
WHERE T1.ACCOUNT_NAME = (SELECT MIN(T2.ACCOUNT_NAME)
                         FROM ACCOUNT T2);
```




### 2.5.2 Limiting rows for SELECT...INTO

In previous versions of DB2, using the SELECT...INTO required the program to ensure that only a single row was returned. This was normally done by using the primary key or any other unique key that existed on the table. This was further complicated if the SELECT statement contained a join to another table which could have any number of rows matching the join criteria.

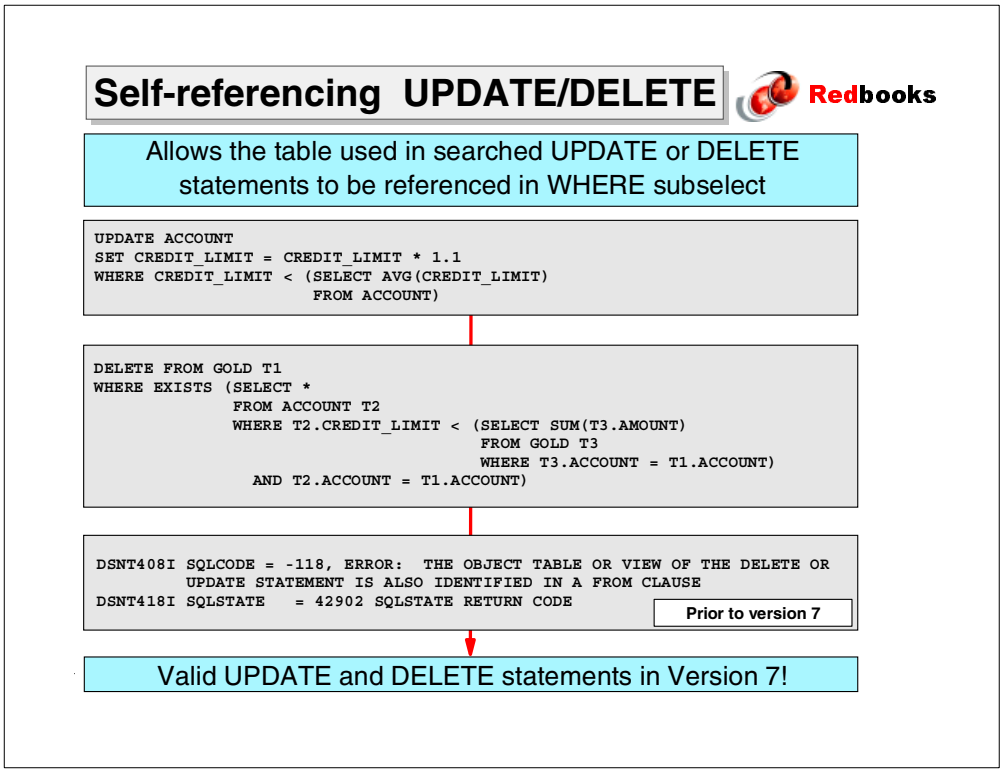
If the SELECT...INTO statement was coded and returned more than one row DB2 would return the SQLCODE -811 **THE RESULT OF AN EMBEDDED SELECT STATEMENT OR A SUBSELECT IN THE SET CLAUSE OF AN UPDATE STATEMENT IS A TABLE OF MORE THAN ONE ROW, OR THE RESULT OF A SUBQUERY OF A BASIC PREDICATE IS MORE THAN ONE VALUE** and the statement would be rejected.

If there was no way of ensuring that only a single row could be returned, then a cursor would have to be opened and the program itself would have to read in the first row that matched the selection and join criteria and throw away all other rows by closing the cursor.

With the addition of the FETCH FIRST n ROWS ONLY clause, this situation can be avoided by using the variant for the SELECT-INTO clause. The SELECT...INTO can now be coded with FETCH FIRST ROWS ONLY to specify that only one row will be returned to the program, even if multiple rows match the WHERE criteria.

In using this clause, it is important to remember that normally the SELECT...INTO is in error if multiple rows are returned, as the statement is being used to verify uniqueness. In this case, the return of the SQLCODE -811 is not only expected if multiple rows are returned but very important.

However, where uniqueness is not significant, this clause can be very powerful.




## 2.6 Self-referencing UPDATE/DELETE

This enhancement to SQL in DB2 V7 allows searched UPDATE and DELETE statements to use the target tables within the subselects in the WHERE clause.

For example, the above UPDATE statement targets the table ACCOUNT. The WHERE clause in this statement uses a subselect to get the average credit limit for all accounts in the ACCOUNT table.

In previous versions of DB2, this self-referencing would not be allowed. The following SQLCODE would be returned, and the statement would fail:

```
-118 THE OBJECT TABLE OR VIEW OF THE DELETE OR UPDATE STATEMENT IS
ALSO IDENTIFIED IN A FROM CLAUSE
```

DB2 now will accept the syntax of this statement, and the SQLCODE -118 will not be returned.

### 2.6.1 Executing the self-referencing UPDATE/DELETE

It is important to note that this enhancement requires that the subquery is evaluated completely before any updates or deletes take place.

For a non-correlated subquery, the subquery will be evaluated once and the result produced, the WHERE clauses will be tested and any valid rows will be updated or deleted.

Two steps may be required for correlated subqueries. The first step creates a work file and inserts the RID for a delete or RID and column value for an update. This is followed by a step which reads the work file and repositions on the record in the base table pointed to by the RID, and then either updates or deletes the row as required.

### 2.6.2 Restrictions on usage

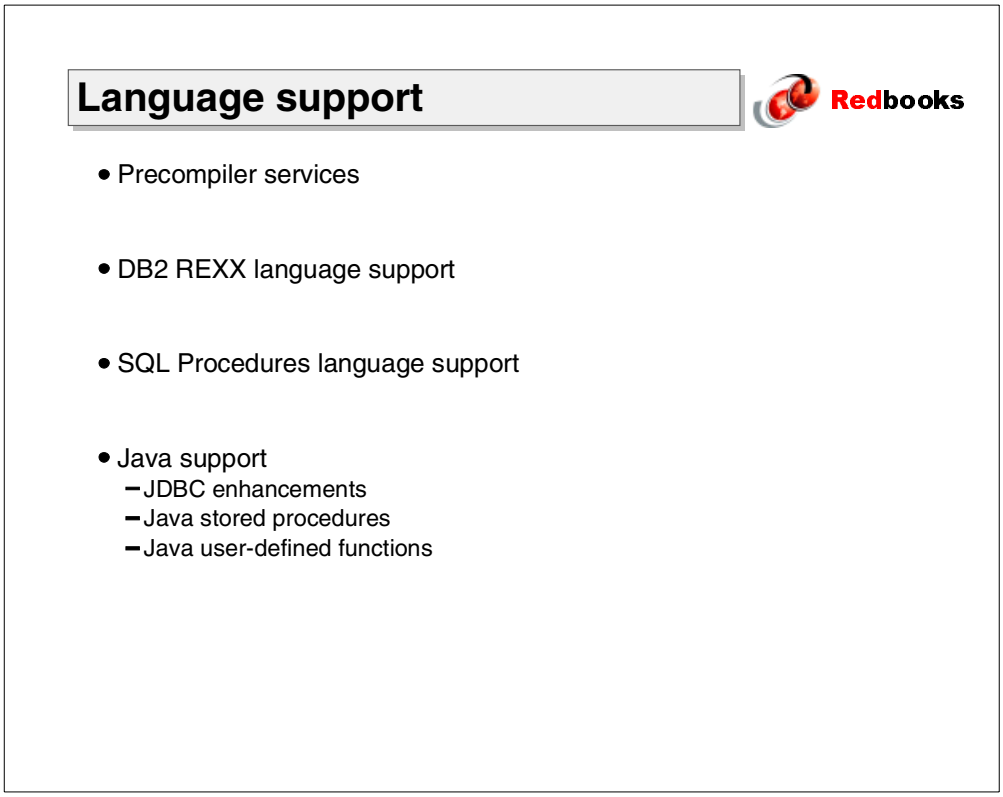
DB2 positioned updates and deletes will still return the SQLCODE -118 if a subquery in the WHERE clause references the table being updated or which contains rows to be deleted.

For example, the following positioned update is still invalid:

```
EXEC SQL DECLARE CURSOR C1 CURSOR FOR
           SELECT T1.ACCOUNT, T1.ACCOUNT_NAME, T1.CREDIT_LIMIT
           FROM ACCOUNT T1
           WHERE T1.CREDIT_LIMIT < (SELECT AVG(T2.CREDIT_LIMT)
                                    FROM ACCOUNT T2)
           FOR UPDATE OF T1.CREDIT_LIMIT;
.
.
EXEC SQL OPEN C1;
.
.
EXEC SQL FETCH C1 INTO :hv_account, :hv_acctname, :hv_crdlmt;
.
.
EXEC SQL UPDATE ACCOUNT
         SET CREDIT_LIMIT = CREDIT_LIMIT * 1.1
         WHERE CURRENT OF C1;
```

The SQLCODE -118 will be returned at bind time.

# Chapter 3. Language support




DB2 V7 continues to extend its support for e-business and application development technologies. Using DB2 V7, you can leverage your existing applications while developing and expanding your electronic commerce for the future.

***Precompiler Services***
Precompiler Services is an application programming interface (API) that is called by a host language compiler. In the new method of program preparation, the host language compiler performs both DB2 precompiler and host language compiler functions. The host language complier calls the Precompiler Services API to process the SQL statements. You eliminate the DB2 precompile step of program preparation.

By implementing Precompiler Services, DB2 V7 is able to remove a number of restrictions imposed by the DB2 precompiler. For example, you can use structured host variables (array elements) as host variables.

***DB2 REXX language support***
REXX language support is available with DB2 UDB for OS/390 Versions 5 and 6; REXX language stored procedures is also available in these DB2 releases by APAR. REXX language support and REXX language stored procedures are shipped with DB2 V7 base code. DB2 V7 also extends DB2 REXX support to allow the userid and password to be specified on the SQL CONNECT statement to DB2.

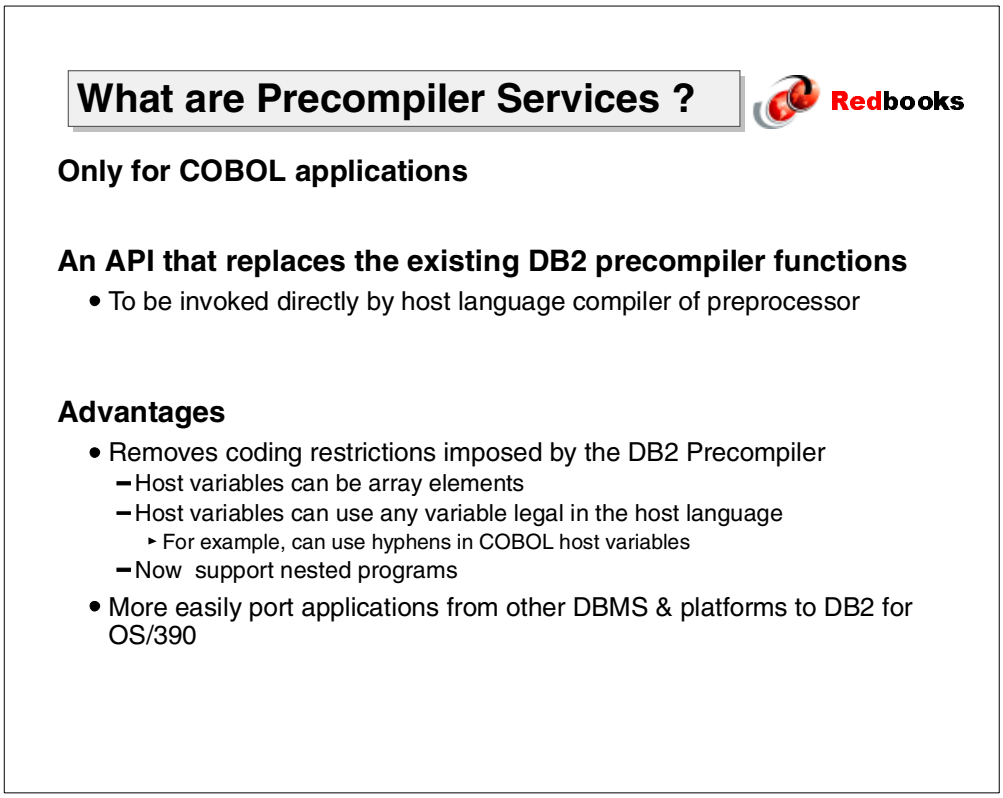
© Copyright IBM Corp. 2001

***SQL Procedures***
SQL Procedures are introduced in DB2 UDB for OS/390 Version 5 and Version 6 by APAR and download. SQL Procedures support is shipped with the DB2 V7 base code. In addition, the “user managed” tables SYSIBM.SYSPSM and SYSIBM.SYSPSMOPTS have moved to the DB2 catalog, as SYSIBM.SYSROUTINES_PSM and SYSIBM.SYSROUTINES_OPTS, respectively.

***DB2 Java support***
DB2 V7 introduces a number of significant enhancements to support Java as a programming language. Many of these enhancements can be considered as “technology enablement”, as they are designed to be exploited by new software releases, notably OS/390 WebSphere Version 4 and the OS/390 Java Development Kit Version 1.3.

***JDBC enhancements***
DB2 V7 implements support for the JDBC 2.0 standard, by implementing some of the functions that are defined in JDBC 2.0. These functions are required to support the JDK 1.3 on OS/390 and products such as WebSphere Version 4, planned to become available first quarter 2001.

***Java stored procedures***
DB2 V7 allows you to implement Java stored procedures as both compiled Java using the OS/390 High Performance Java Compiler (HPJ) and interpreted Java executing in a Java Virtual Machine (JVM). DB2 also provides support for user-defined functions written in Java. DB2 provides support for external functions only. The Java function cannot perform any SQL.




## 3.1 Precompiler Services

This section shows the enhancements related to the DB2 precompiler.

### 3.1.1 What are Precompiler Services?

Precompiler Services are implemented through an application programming interface (API) that is called by a host language compiler or preprocessor. The API performs the tasks that the DB2 precompiler currently performs. You eliminate the DB2 precompile step of program preparation.

By implementing Precompiler Services, DB2 V7 is able to remove some of the restrictions imposed by the DB2 precompiler.

***COBOL restrictions lifted***

- The restriction that the REPLACE statement has no effect on SQL statements, is lifted.
- The restriction that you cannot use hyphens in host variables is lifted.
- There are two restrictions when using VS COBOL II or COBOL/370 with the option NOCMPR2. Both of these restrictions are lifted:
  - All SQL statements and any host variables they reference must be within the first program when using nested programs or batch compilation.
  - DB2 COBOL programs must have a DATA DIVISION and a PROCEDURE DIVISION. Both divisions and the WORKING-STORAGE section must be present in programs that use the DB2 precompiler.

- Also, the restriction that you cannot write nested programs is lifted.

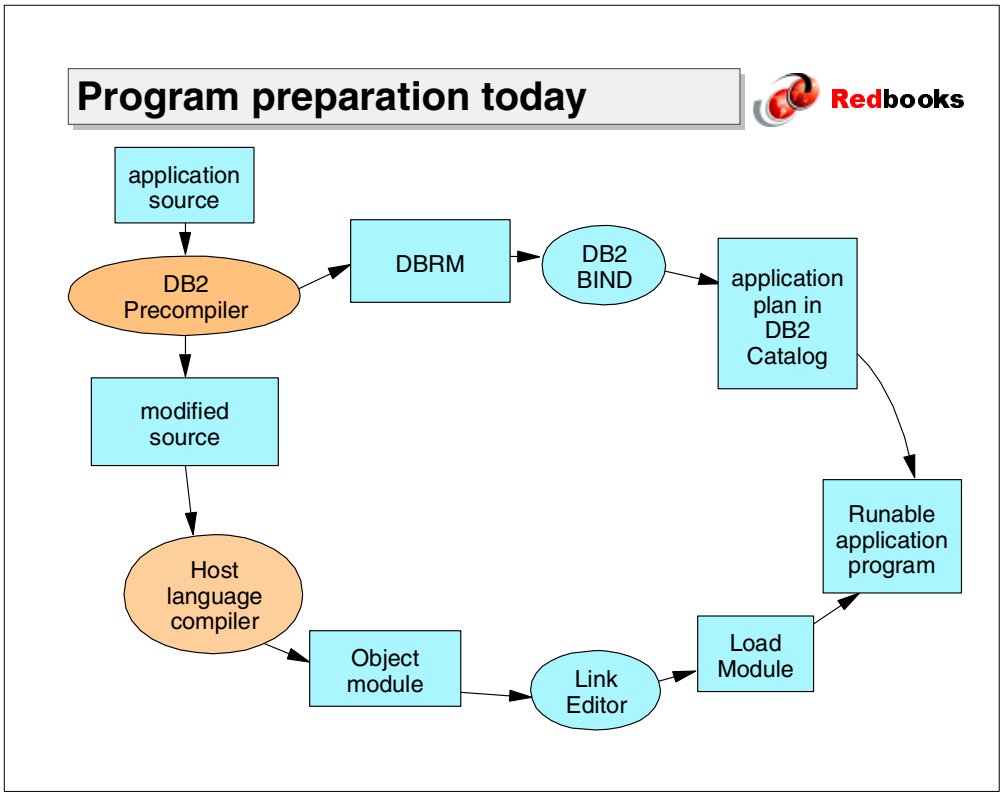
VS COBOL II supports structured programming methods primarily through the use of nested programs. A nested program is compiled in the same source module as the main line code, and has its own Data and Procedure divisions. Except for the Linkage section, nested program variables are local (visible only to that subroutine). Without Precompiler Services, the DB2 precompiler processes embedded SQL code in a manner which prevents its use within VS COBOL II nested programs. Any attempt to use embedded SQL within a nested COBOL II program will cause a syntax error at compile time. (The use of embedded SQL within nested programs is not restricted in "C" or PL/1.)

***Language independent restrictions lifted:***

- There will no longer be restrictions on using structured host variables (for example; array elements).
- With Precompiler Services, a programmer can use any variable legal in the host language as a host variable in a database application, provided that the data type is compatible with the data type of the corresponding column of the database.

The use of Precompiler Services also enhances the DB2 family compatibility. You can more easily port applications from other platforms and products (like Oracle, Informix and Sybase) into DB2 V7. For example, the host variable names which are valid and being used in the other DBMSs can also be used in DB2 for OS/390 and z/OS with no change.

The Precompiler Services function is intended to be used by host language compiler writers. Currently, the IBM COBOL, PL/I, and C/C++ compilers are scheduled to implement support for Precompiler Services in their preprocessors. When they have done so, you can perform the precompilation step as part of the host preprocessing or compilation step, rather than invoking the DB2 precompiler.



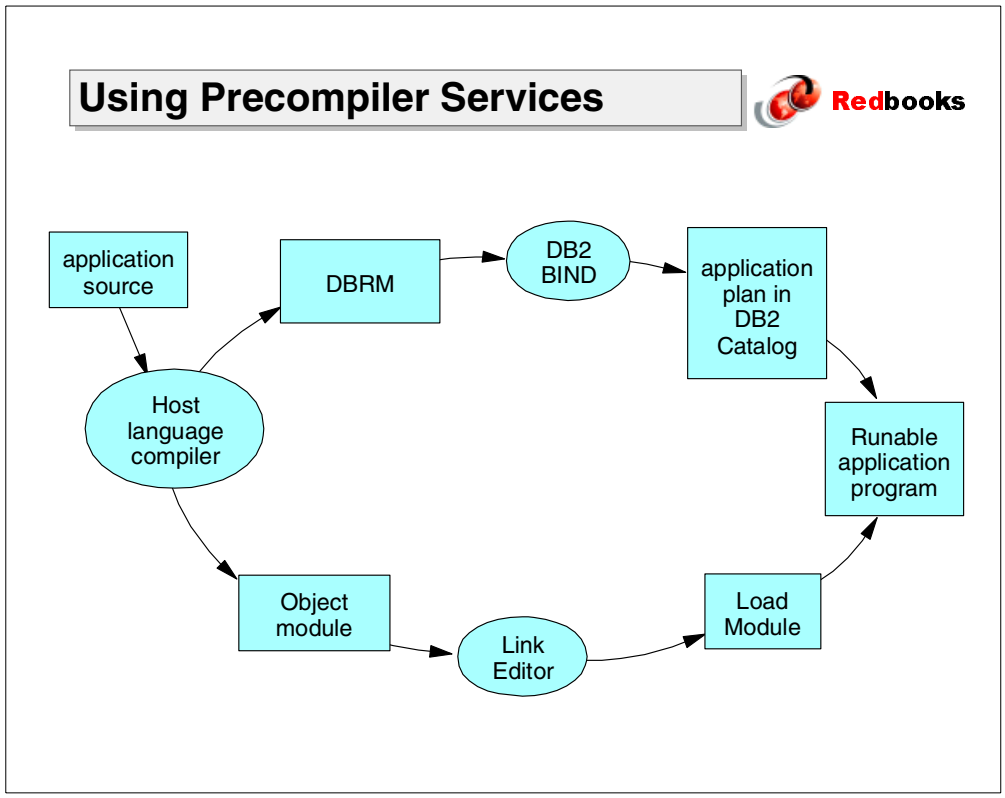

### 3.1.2 Program preparation today

This foil provides an overview of the tasks you must perform to prepare your application program for DB2.

You cannot compile your DB2 application programs containing SQL until you change the SQL statements into language recognized by your compiler or assembler. Hence, you must use the DB2 precompiler to:

- Replace the SQL statements in your source programs with compilable code.
- Create a database request module (DBRM), which communicates your SQL requests to DB2 during the bind process.

After you have precompiled your source program, you create a load module, possibly one or more packages, and an application plan. It does not matter which you do first. Creating a load module is similar to compiling and link-editing an application containing no SQL statements. Creating a package or an application plan, a process unique to DB2, involves binding one or more DBRMs into a plan.

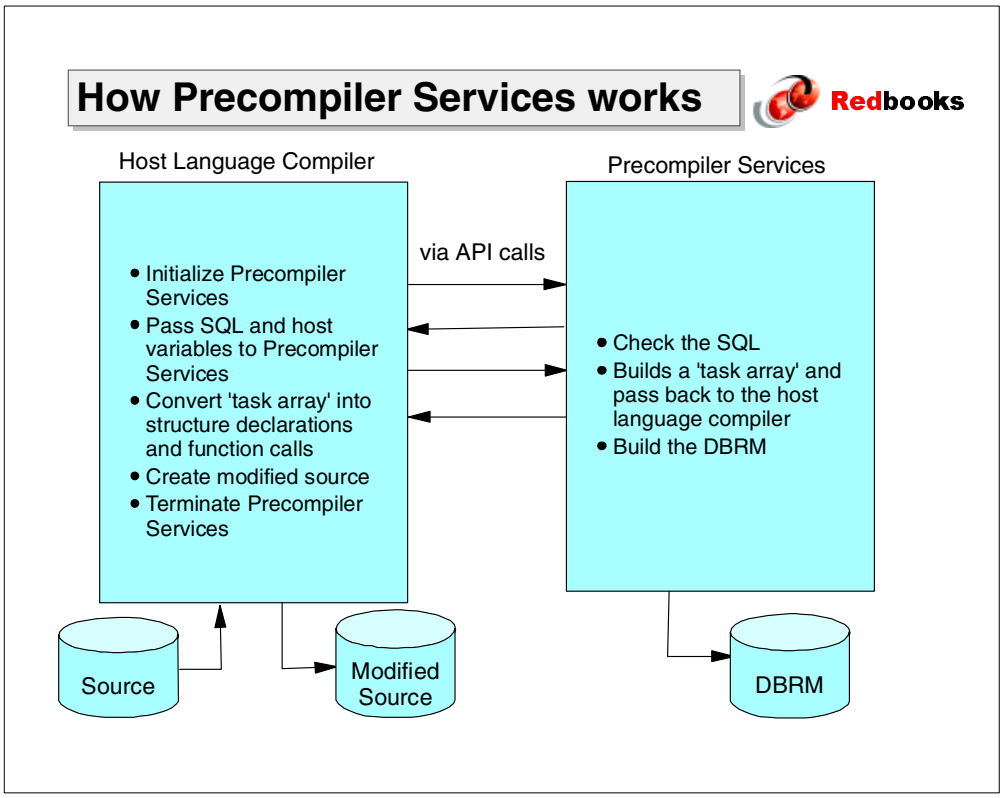



### 3.1.3 Using Precompiler Services

Here is the new method to prepare your DB2 application program for DB2, when you are using a host language compiler preprocessor that uses DB2's Precompiler Services:

1. Compile the source code, (or modified source code for PL/I or C/C++ applications), to produce modified source code and a DBRM. The compiler also replaces the SQL statements in your application program with compilable code.
2. Link-edit the object module.
3. Bind the DBRM(s) into package(s) and then a plan, or directly into a plan.

In the new method of program preparation, the host language compiler performs both DB2 precompiler and host language compiler functions. The SQL statements are passed by the host language compiler to the Precompiler Service for validation and generation of the DBRM. The host language compiler updates the source code with the required data structures and native host language compilable calls to DB2.




### 3.1.4 How Precompiler Services works

The modified host language compiler has the following responsibilities:

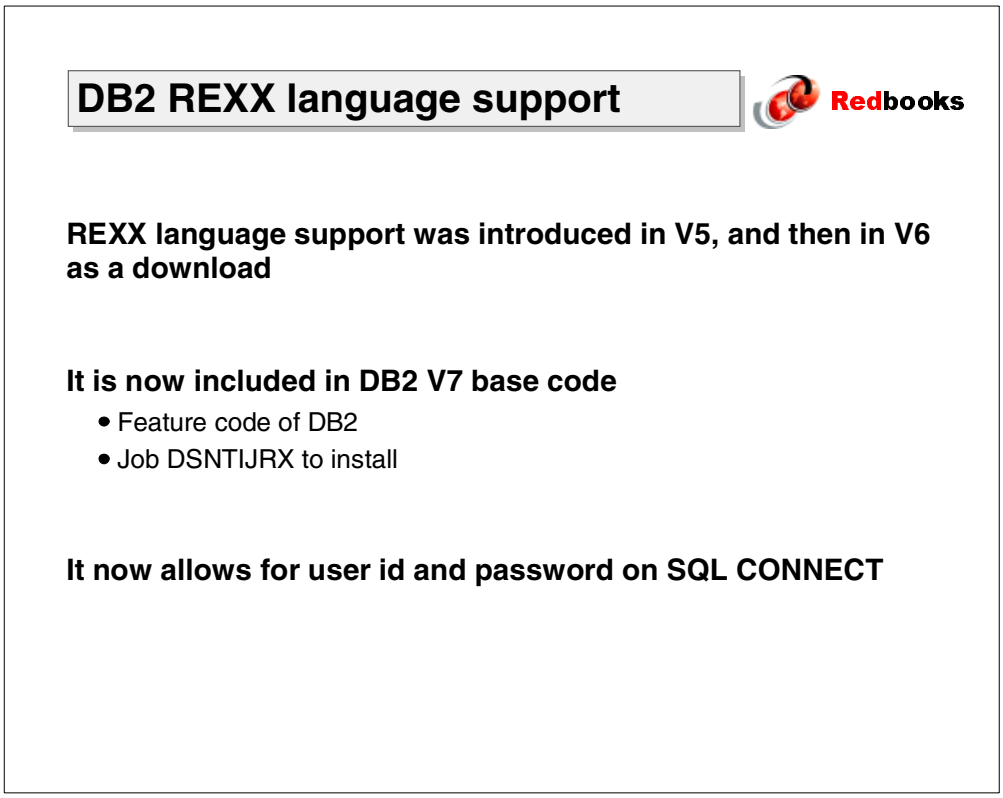
- Creates the necessary data structures.
- Translates the application source file into a modified source file.
- Processes host variable declarations.
- Processes SQL statements.
- Constructs host language compilable calls to DB2.

Throughout the compilation process, the host language compiler calls the Precompiler Services APIs many times, each time passing different SQL statements to process. The APIs are:

- **sqlainit -** Initializes the precompilation process.
- **sqlaalhv -** Records a host variable.
- **sqlacmpl -** Compiles an SQL statement and places it into a DBRM.
- **sqlafini -** Terminates the precompilation process.

The Precompiler Services function has the following responsibilities:

- Performs a full syntactic and semantic check of the SQL statements passed.
- While compiling an SQL statement, Precompiler Services determines how all host variables are used (that is, for input or for output, or as indicator variables).
- After processing all the SQL statements, Precompiler Services creates a list of tasks, called a task array, and passes the array back to the host language compiler. The host language compiler then converts tasks in this array to data structure declarations and function calls in the application program. It then inserts these declarations and calls into the modified source file. The host language compiler then compiles the application program so it can later be executed.
- Precompiler Services then creates a DBRM. The processed SQL statements are stored in a DBRM, to create the package at a later time.




## 3.2 DB2 REXX language support

REXX language support is available with DB2 V5 and V6. The REXX language stored procedures are also available by APAR:

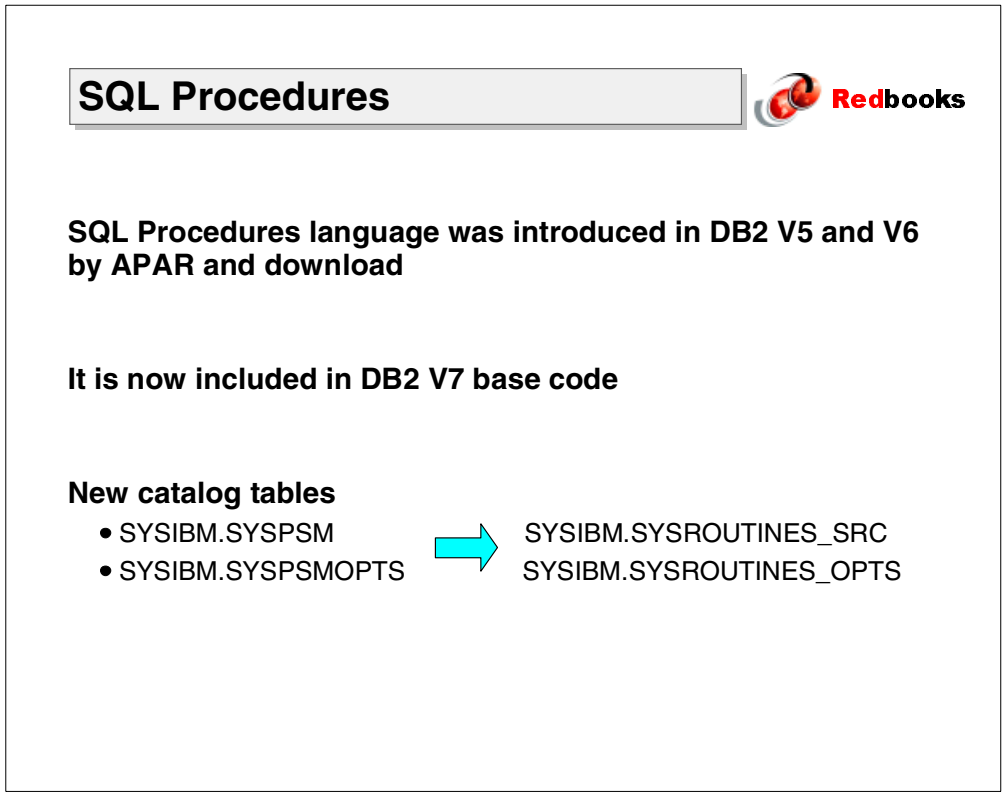
- DB2 UDB for OS/390 Version 5: APAR PQ29706
- DB2 UDB for OS/390 Version 6: APAR PQ30219

REXX language support for DB2 is also included in the DB2 UDB for OS/390 Version 6 April 2000 code refresh. Refer to the redbook *DB2 UDB Server for OS/390 Version 6 Technical Update*, SG24-6108, for further details.

REXX language and REXX language stored procedure support are shipped as a part of the DB2 V7 base code. You need to specify the feature and the media when ordering DB2. Documentation is still accessible from the Web.

The DB2 installation job DSNTIJRX binds the REXX language support to DB2 and makes it available for use.

DB2 V7 also extends DB2 REXX language support to allow the userid and password to be specified on the SQL CONNECT statement to DB2. Refer to 6.2.4, “CONNECT with userid and password” on page 295 for a discussion on the SQL CONNECT statement enhancements. With DB2 V7, REXX support is also extended to savepoints and scrollable cursors.




## 3.3 SQL Procedures

SQL Procedures are DB2 stored procedures written entirely in SQL Procedures language.

The SQL Procedures language is based on SQL extensions as defined by the SQL/PSM (Persistent Stored Modules) standard. SQL/PSM is an ISO/ANSI standard for SQL3.

For detailed information, refer to the redbook, *Developing Cross-Platform DB2 Stored procedures: SQL Procedures and the DB2 Stored Procedure Builder*, SG24-5485. Additional information on SQL/PSM can be found in the reference book, *Understanding SQL's Stored Procedures: A Complete Guide to SQL/PSM*, Jim Melton, Morgan Kaufmann Publishers, Inc., ISBN 1-55860-461-8.

SQL Procedure support is shipped in DB2 V5 and V6 by APAR and as downloads from the following Web site:

`www.ibm.com/software/data/db2/os390/sqlproc`

SQL Procedure support is shipped as a part of the DB2 V7 base code. It relies on DB2 REXX language support being installed. You need to specify the feature and the media when ordering DB2.

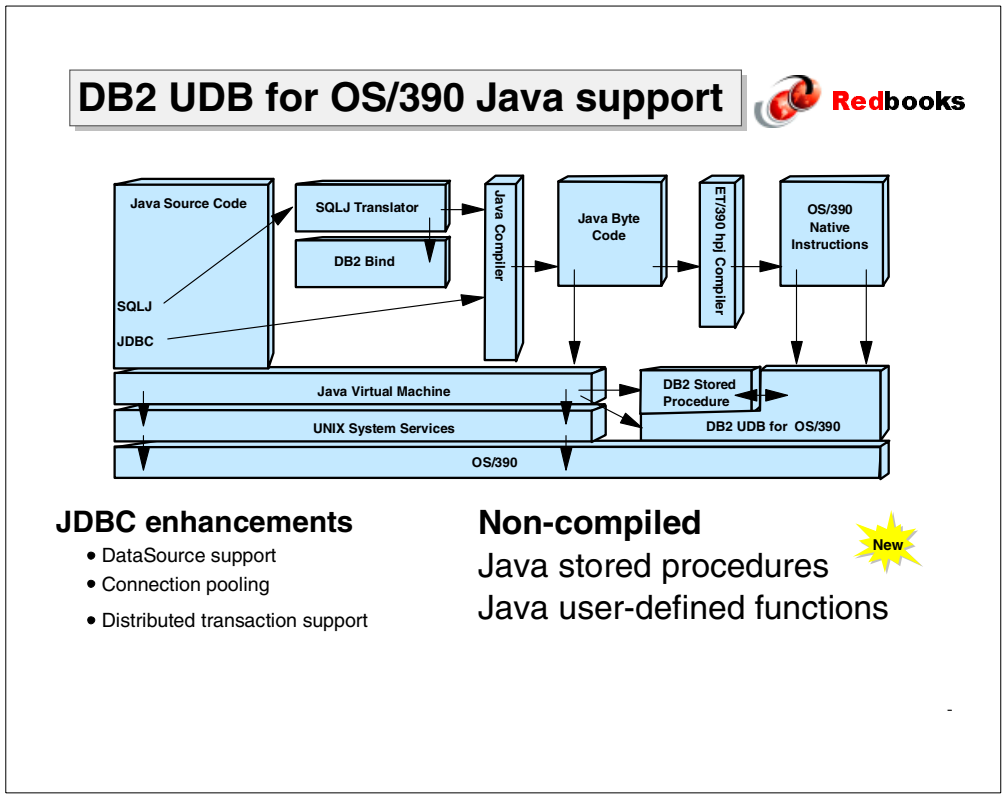
In addition, a new table space is introduced into the DB2 catalog to define the new tables SYSIBM.SYSROUTINES_SRC and SYSIBM.SYSROUTINES_OPTS.

The DB2 Stored Procedure Builder (SPB) tool uses these tables to store the SQL procedure source code and the SPB invocation options. This introduces better support for SQL stored procedures by providing extra functionality such as version control and better management of source code.

The migration job DSNTIJIN creates the dataset for a new table space DSNDB06.SYSGRTNS.

Step 3 of migration job DSNTIJTC creates the new catalog tables SYSIBM.SYSROUTINES_SRC and SYSIBM.SYSROUTINES_OPTS in DSNDB06.SYSGRTNS.

The migration job DSNTIJMP migrates any SQL procedure data from the ‘user-maintained’ tables in DB2 V5 and V6, SYSIBM.SYSPSM and SYSIBM.SYSPSMOPTS, to the new catalog tables in DB2 V7.




## 3.4 DB2 Java support

This section describes Java support enhancements.

***JDBC enhancements***

DB2 V7 implements support for the JDBC 2.0 standard, by implementing a number of the functions that are defined in JDBC 2.0. These functions are required to support the OS/390 Java Developers Kit (JDK) Version 1.3 and products such as WebSphere Version 4, planned to become available by first quarter 2001:

- JDBC 2.0 DataSource support
- JDBC 2.0 connection pooling
- JDBC 2.0 Distributed transaction support

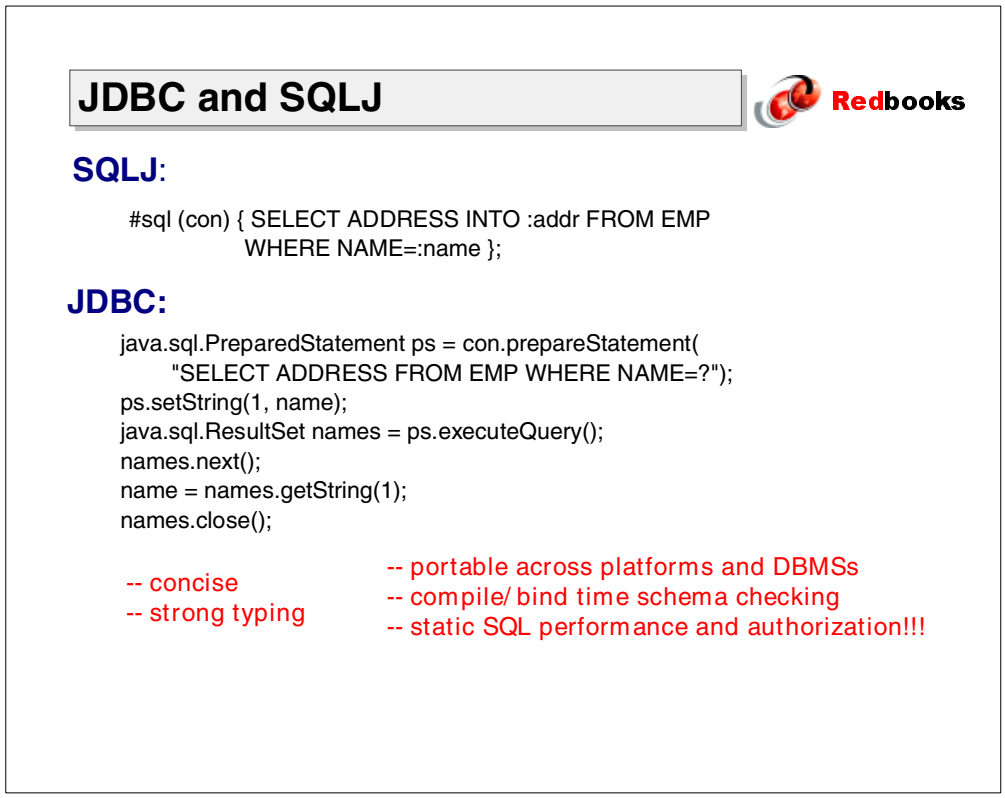
In addition, DB2 V7 implements the following enhancements to JDBC:

- Adds support for userid/password usage on SQL CONNECT via URL.
- Allows JDBC driver execution under IMS.

### *Java stored procedures*

DB2 V7 allows you to implement Java stored procedures as both compiled Java using the OS/390 High Performance Java Compiler (HPJ) and interpreted Java executing in a Java Virtual Machine (JVM).

DB2 V7 also provides support for user-defined functions written in Java. DB2 only provides support for external functions. The Java functions cannot execute any SQL.




### 3.4.1 JDBC and SQLJ

SQLJ supports embedded static SQL in Java applications and applets. Prior to SQLJ, SQL issued by Java programs was exclusively dynamic and supported through the JDBC application programming interface. The differences between SQLJ and JDBC are:

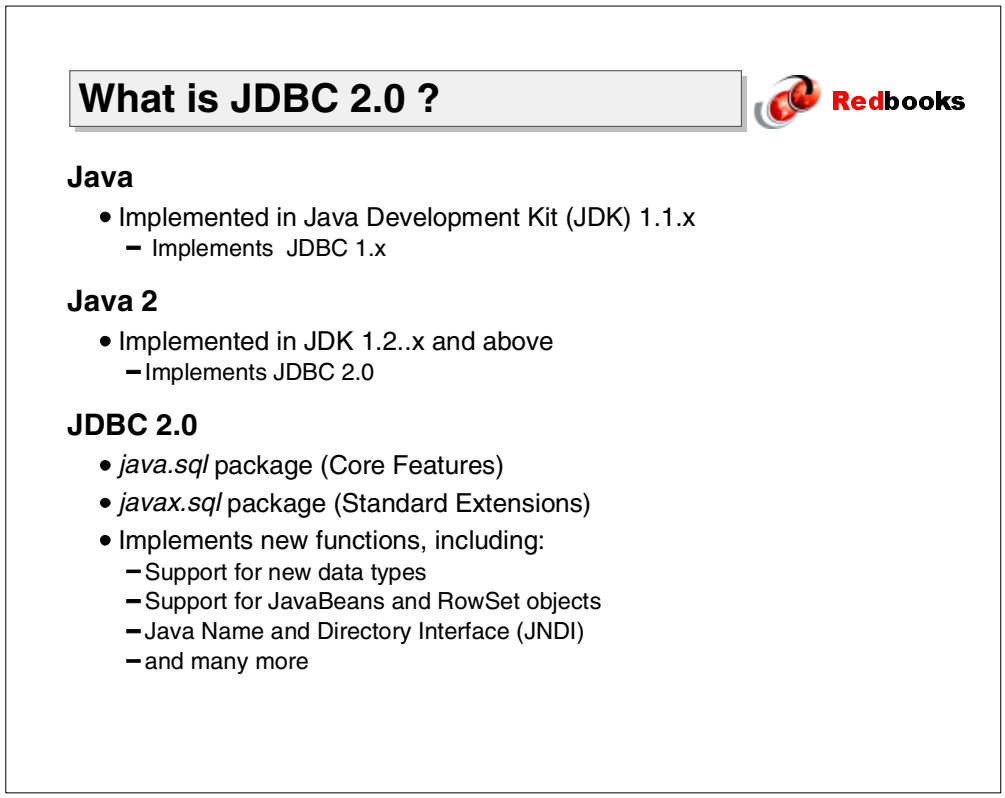
- SQLJ uses the static SQL model, and JDBC uses the dynamic SQL model.
- SQLJ source programs are smaller than equivalent JDBC programs, because certain code that the programmer must include in JDBC programs is generated automatically by SQLJ.
- SQLJ does data type checking during the program preparation process and enforces strong typing between table columns and Java host expressions. JDBC passes values to and from SQL tables without compile-time data type checking.
- In SQLJ programs, you can embed Java host expressions in SQL statements. JDBC requires a separate call statement for each bind variable and specifies the binding by position number.
- SQLJ provides the advantages of static SQL authorization checking. With SQLJ, the authorization ID under which SQL statements execute is the plan or package owner. DB2 checks table privileges at bind time. Because JDBC uses dynamic SQL, the authorization ID under which SQL statements execute is not known until run time, so no authorization checking of table privileges can occur until run time.
- SQLJ does not support dynamic SQL. You have to code dynamic SQL statements using JDBC calls.

- SQLJ is supported on the OS/390 platform when used in a Java application or when it is part of a servlet.

SQLJ supports the following types of SQL statements.

- Data Manipulation Language (DML) statements:
  - SELECT, both singleton and cursor-based
  - INSERT, searched or positioned
  - UPDATE, including UPDATE WHERE CURRENT OF
  - DELETE, including DELETE WHERE CURRENT OF
  - CALL, for calls to stored procedures
  - COMMIT and ROLLBACK
  - SET special register, and SET host variable
- Data Definition Language (DDL) statements:
  - CREATE
  - ALTER
  - DROP
- Data Control Language (DCL) statements:
  - GRANT
  - REVOKE

SQLJ does not support dynamic SQL operations such as PREPARE and EXECUTE. If your application needs to issue a dynamic SQL, you have to code dynamic SQL statements using JDBC calls. As JDBC already supports these operations, SQLJ simply interoperates with JDBC to enable support for dynamic SQL. In order to support this, SQLJ includes methods for CONNECT and DISCONNECT SQL statements.




### 3.4.2 What is JDBC 2.0?

Java Database Connectivity (JDBC) is the industry standard for database independent connectivity between Java applets/applications and a broad range of SQL relational databases. It allows a Java programmer to do three things:

1. Establish a connection to a database.
2. Issue SQL statements.
3. Process the results.

The initial release of the JDBC API standard is known as JDBC 1.0. The second release of the JDBC API is known as JDBC 2.0. When the JDBC 1.0 specification was created, it was based on SQL2, which was the Structured Query Language (SQL) standard at that time. The SQL3 standard is now emerging, and support for it is included in JDBC 2.0.

The Java Developers Kit (JDK) 1.1.x implements JDBC 1.x, and the JDK I.2.x and above implements JDBC 2.0. JDK 1.2.x is currently available on a number of platforms, while JDK 1.3.x will become available on OS/390 later this year.

The JDBC 2.0 API includes two packages:

1. The j*ava.sql* package, which is known as the JDBC 2.0 Core API. This includes the original JDBC API, referred to as the JDBC 1.0 API, plus the new core API that has been added.
2. The *javax.sql* package, which is the JDBC 2.0 Standard Extension API or Optional Package. This package is entirely new and is available as a part of the Java 2 Platform SDK, Enterprise Edition.

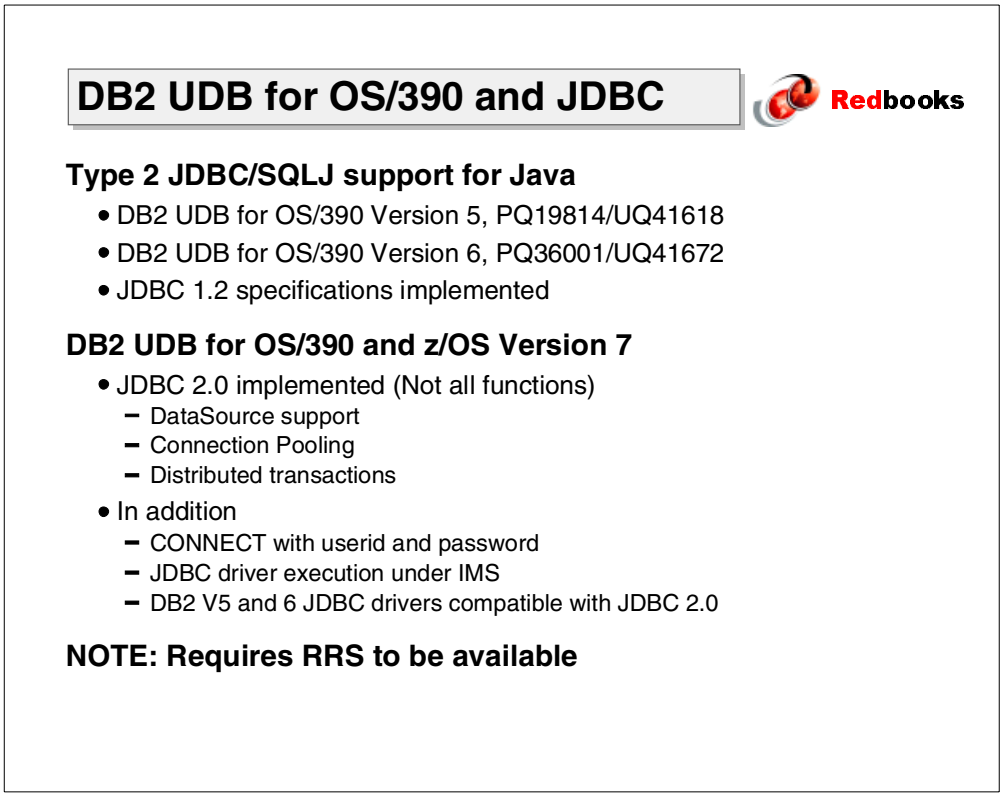
JDBC 2.0 has the following enhancements:

- JDBC Technology Core features (part of the Java 2 SDK, Standard Edition)
  - *Result set enhancements* - include:
    - *Scrollable result set* - Ability to move a result set's cursor to a specific row. This feature is used by GUI tools and for programmatic updates.
    - *Updatable result se*t - Ability to use Java programming language commands rather than SQL.
  - *New data types support* - This item increases support for storing persistent Java programming language objects (Java objects) and mapping new SQL data types such as binary large objects, and structured types. Performance improvements include the ability to manipulate large objects such as BLOB and CLOB without bringing them to the client from the database server.
  - *Batch updates* - The batch update feature allows an application to submit multiple update statements (insert/update/delete) in a single request to the database. This can provide a dramatic increase in performance when a large number of update statements need to be executed. This strategy can be much more efficient than sending update statements separately.
- JDBC Optional Package features (the javax.sql package)
  - *The Java Naming and Directory Interface (JNDI)* - This API can be used in addition to a JDBC driver to obtain a connection to a database. When an application uses the JNDI API, it specifies a logical name that identifies a particular database instance and JDBC driver for accessing that database. This has the advantage of making the application code independent of a particular JDBC driver and JDBC URL coding requirements. This also supports ease of deployment of Java code, (gives JDBC driver independence, makes JDBC applications easier to manage).
  - *Connection pooling* - The JDBC API contains *hooks* that allow connection pooling to be implemented on top of the JDBC driver layer. This allows for a single connection cache that spans the different JDBC drivers that may be in use. Since creating and destroying database connections is expensive, connection pooling is important for achieving good performance, especially for server applications.
  - *Distributed transactions* - Support for distributed transactions has been added as an extension to the JDBC API. This feature allows a JDBC driver to support the standard 2-phase commit protocol used by the Java Transaction Service (JTS) API.

- *JavaBeans (RowSet objects)* - As its name implies, a rowset encapsulates a set of rows. A rowset may or may not maintain an open database connection. Rowsets add support to the JDBC API for the JavaBeans component model. A rowset object is a bean. A rowset implementation may be serializable. Rowsets can be created at design time and used in conjunction with other JavaBeans components in a visual builder tool to construct an application. Rowsets are used to send data across a network to thin clients, such as Web browsers. Rowsets can be any tabular data source, even spreadsheets or flat files. This specification also makes results sets scrollable or updatable when the JDBC driver does not support scrollability and updatability. You can encapsulate a driver as a JavaBeans component for use in a GUI.

Refer to the following Web site for further information on JDBC:

`http://java.sun.com/products/jdbc/`




### 3.4.3 DB2 UDB for OS/390 and JDBC

DB2 for OS/390 introduced Type-2 JDBC/SQLJ support for Java, via APAR in:

- DB2 UDB for OS/390 Version 5, PQ19814/UQ41618
- DB2 UDB for OS/390 Version 6, PQ36001/UQ41672

The SQLJ specification consists of three parts:

- Database Languages – SQL – Part 0: Object Language Bindings (SQL/OLB) is also known as SQLJ Part 0. It was approved by ANSI in 1998, and it specifies the SQLJ language syntax and semantics for embedded SQL statements in a Java application.
- Database Languages – SQLJ – Part 1: SQL Routines using the Java Programming Language was approved by ANSI in 1999, and it specifies extensions that define: – Installation of Java classes in an SQL database – Invocation of static methods as stored procedures.
- Database Languages – SQLJ – Part 2: SQL Types using the Java Programming Language is under development. It specifies extensions for accessing Java classes as SQL user-defined types.

The DB2 for OS/390 implementation of SQLJ includes support for the following portions of the specification:

- Part 0
- The ability to invoke a Java static method as a stored procedure, which is in Part 1

For detailed information on Java, refer to the following books:

- *DB2 Java Stored Procedures: Learning by Example,*SG24-5945
- *DB2 UDB Server for OS/390 Version 6 Technical Update,* SG24-6108
- *DB2 UDB Application Programming Guide and Reference for Java,* SC26-9018

The JDBC/SQLJ support implemented in DB2 UDB for OS/390 Version 5 and Version 6 complies with JDBC 1.2 specification.

DB2 UDB for OS/390 and z/OS Version 7 implements some of the functions that are defined in JDBC 2.0. These functions are required to support the OS/390 JDK 1.3 and products such as WebSphere Version 4, planned to become available first quarter 2001:

- JDBC 2.0 DataSource support
- JDBC 2.0 connection pooling
- JDBC 2.0 Distributed transaction support

In addition, DB2 V7 implements the following enhancements to JDBC support:

- Add support for userid/password usage on SQL CONNECT via URL
- JDBC Driver execution under IMS

The JDBC driver shipped with DB2 UDB for OS/390 Version 5 and Version 6 will also be enhanced to support:

- JDBC Driver execution under IMS
- “Compatibility” with Java 2

DB2 V7 JDBC support uses RRSAF to connect to DB2 rather than CAF. RRS must therefore be set up before JDBC 2.0 can be used. JDBC 2.0 also has a pre-requisite of JDK 1.3 (a part of OS/390 V2 R8).

Support for VisualAge for Java Enterprise Edition for OS/390 Version 2.0 for general applications, and CICS applications in particular, was also delivered in DB2 UDB for OS/390 Version 5 and Version 6 by APARs PQ/36643/UQ43898 and PQ36644/UQ43899 respectively.This functionality is also shipped with DB2 V7 base code.

***JDBC driver restrictions***

The following methods, SQL types, and features are not supported by the JDBC driver shipped with DB2 V7:

Unsupported JDBC 1.2 methods:

- `ResultSetMetaData.getTableName()`
- `ResultSetMetaData.getColumnName()`

Unsupported JDBC 2.0 Core API methods:

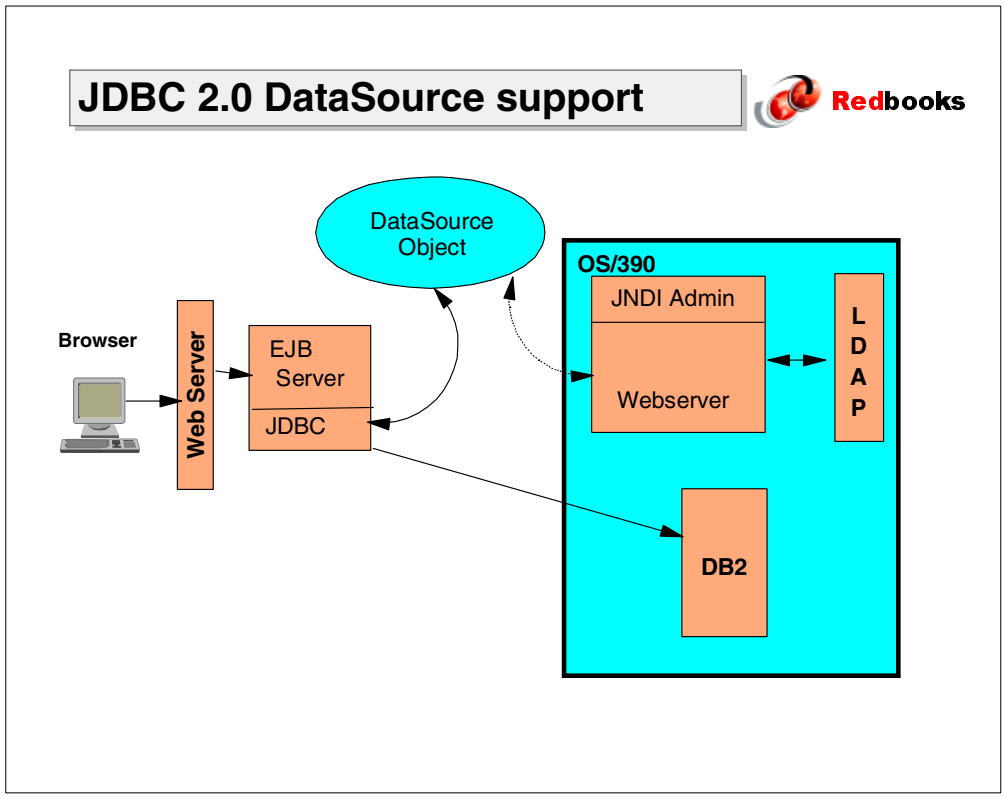
- `Reader.read()` fails with an `ArrayIndexOutOfBounds` exception when you read CLOB or DBCLOB data inserted by the method `PreparedStatement().setCharacterStream()`
- `ResultSetMetaData.getTableName()`
- `ResultSetMetaData.getColumnName()`
- `Resultset.updateXXX()`
- `Resultset.getTime(int, Calendar)`
- `Resultset.getTimestamp(int, Calendar)`

Unsupported JDBC 2.0 SQL types:

- ARRAY
- DISTINCT
- JAVA OBJECT
- REF
- STRUCT

Unsupported JDBC 2.0 Optional Package API features:

- `RowSet`
- Java Transaction API (JTA) is supported by the application driver (COM.ibm.db2.jdbc.app.DB2Driver), but is not supported by the applet driver (COM.ibm.db2.jdbc.net.DB2Driver)




### 3.4.4 JDBC 2.0 DataSource support

The `DataSource` interface provides an alternative to the DriverManager class for making a connection to a data source. Using a `DataSource` implementation is better for two important reasons: It makes code more portable, and it makes code easier to maintain.

A `DataSource` object represents a real world data source. Depending on how it is implemented, the data source can be anything from a relational database to a spreadsheet or a file in tabular format.

Information about the data source and how to locate it, such as its name, the server on which it resides, its port number, and so on, is stored in the form of properties on the `DataSource` object. This makes an application more portable because it does not need to hard code a driver name, which often includes the name of a particular vendor, the way an application using the DriverManager class does. It also makes maintaining the code easier because if, for example, the data source is moved to a different server, all that needs to be done is to update the relevant property. None of the code using that data source needs to be touched

A systems administrator or someone working in that capacity deploys a `DataSource` object, by setting the `DataSource` object's properties and then, optionally, registering it with a Java Naming and Directory Interface (JNDI) naming service, which is generally done with a tool. As part of the registration process, the systems administrator will associate the `DataSource` object with a logical name. This name can be almost anything, usually being a name that describes the data source and that is easy to remember. In the example that

follows, the logical name for the data source is `InventoryDB`. By convention, logical names for `DataSource` objects are in the subcontext `jdbc`, so the full logical name in this example is `jdbc/InventoryDB`.

Once a `DataSource` object has been deployed, application programmers can use it to make a connection to the data source it represents. The following code fragment shows how to retrieve the `DataSource` object associated with the logical name `jdbc/InventoryDB` and then uses it to get a connection.

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/InventoryDB");
Connection con = ds.getConnection("myPassword", "myUserName");
```

In a basic `DataSource` implementation, the Connection object that is returned by the `DataSource.getConnection` method is identical to a Connection object returned by the `DriverManager.getConnection` method.

For the application programmer, using a `DataSource` object is a matter of choice. However, programmers writing JDBC applications that include other JDBC 2.0 features like connection pooling and distributed transactions, must use a `DataSource` object to get their connections. These features are also implemented by the `DataSource` method.

JNDI is a generic interface for accessing objects stored in a naming and directory service. Examples of naming and directory services include DNS services, LDAP services, NIS, and even file systems. Their job is to associate names with information. DNS, for example, associates a computer name like `www.ibm.com` with an IP address like `198.133.16.99` The advantage of maintaining this association is twofold. First of all, `www.ibm.com` is clearly simpler to remember than `198.133.16.99`. Additionally, the physical location of the machine can be changed without impacting its name.

JNDI provides a single API for accessing any information stored in any naming and directory service.

One of the weak points of the JDBC 1.2 specification was the complexity of connecting to a database. The connection process is the only commonly used part of the JDBC specification that requires knowledge of the specific database environment in which you are working. In order to make a connection, you have to know the name of the JDBC driver you are using, the URL string that specifies connection information, and any connection parameters. You can use command line arguments to avoid hard coding this information into your Java classes, but the path of least resistance is to stick the JDBC URL and driver name right in your Java code. Furthermore, knowledge of connecting to one database does not translate to knowledge of connecting to another. The result is that the database connection is the most error prone part of database programming in Java.

The new JNDI support in the JDBC 2.0 Standard Extension enables this approach to JDBC programming.

The key to JNDI support in JDBC is the `javax.sql.DataSource` interface. A JDBC driver that supports the JDBC Standard Extension functionality provides an implementation of this interface. The implementation will have as its attributes any information necessary to make a connection to a database in the database engine it supports. For example, the `DataSource` implementation for the

SQL-JDBC has attributes that store the name of the machine on which the database runs, the port it is listening to, and the name of the database to which you wish to connect. This DataSource object is then stored in a JNDI directory service with all of these attributes. Your application code needs to know only the name under which this DataSource object is stored. The code to make a connection using the JDBC 2.0 Standard Extension support is now this simple:

```
javax.naming.Context ctx = new InitialContext();
javax.sql.DataSource ds = (DataSource)ctx.lookup("example");
java.sql.Connection conn = ds.getConnection();
```

Compare the simplicity of this paradigm to the old way:

```
Class.forName(driverName).newInstance();
java.sql.Connection conn = DriverManager.getConnection(jdbcURL, uid,
password);
```

The old way requires you to know the name of the JDBC driver you are using and to load it. Once it is loaded, your code has to know the URL to use for the connection. The new way, however, requires no knowledge about the database in question. All of that data is associated with a DataSource in the JNDI directory. If you change anything about your environment, such as the machine on which the database server is running, you only change the entry in the JNDI directory. You do not need to make any application code changes. The following example code shows in detail how to get a DataSource into a JNDI service and then get it back out.

The JDBC 2.0 API standard does not mandate that DataSource method must use the JNDI API. However the DataSource method must be capable of using JNDI services.

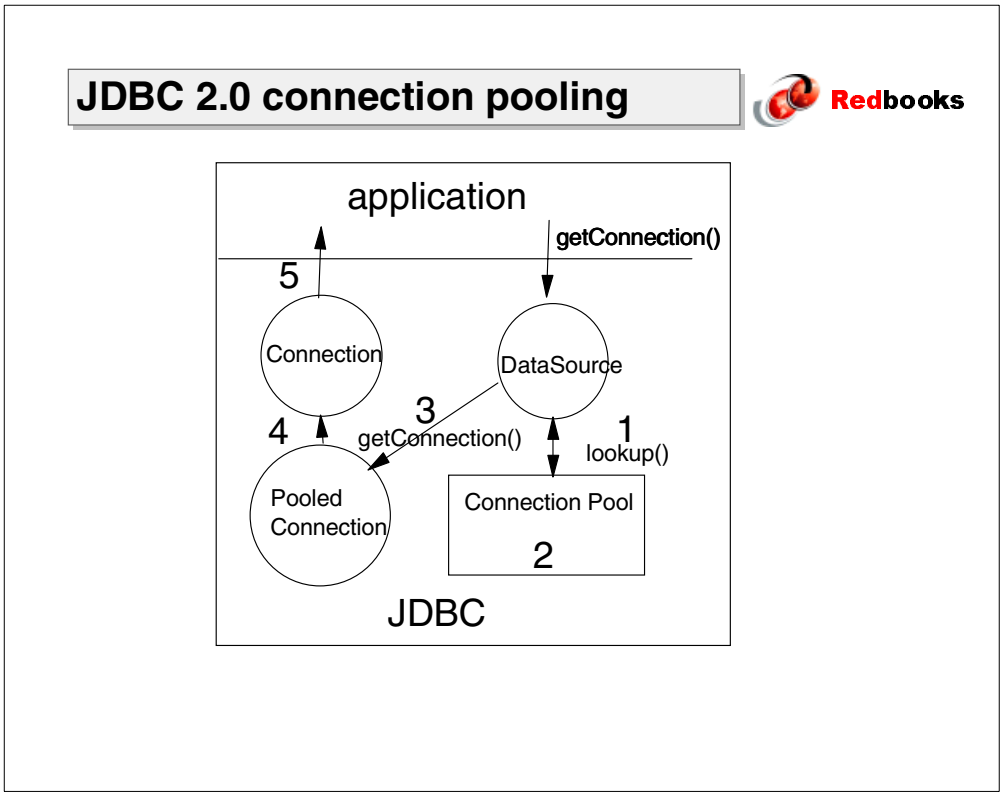
DB2 V7 JDBC support implements the DataSource method in two flavors, using JNDI and without using JNDI.

JNDI services is implemented on OS/390 in WebSphere Version 4. WebSphere uses LDAP as its data store. JNDI Administration services are implemented in WebSphere to register and maintain entries.

The DataSource method implemented with DB2 V7 can be configured to optionally use JNDI services provided by WebSphere, (hence the dotted arrow).

When you use the DataSource method which is configured to use WebSphere, you must first use the JNDI Administration tool shipped with WebSphere to define the DB2 subsystem as a data source.

When you use the DataSource method which is not configured to use JNDI through the Web server, there is more you must code in your Java application. You must first invoke the DataSource method to create the DataSource object before it can be referenced. When you want to connect to the data source you must then invoke the DataSource method again, This seems a little “self defeating”, as one of the reasons for DataSource methods is to remove knowledge of the data source from the Java code. Clearly you have to know this information to first create the DataSource object.




### 3.4.5 JDBC 2.0 connection pooling

Connection pooling is a mechanism whereby, when an application closes a connection, that connection is recycled rather than being destroyed. Because establishing a connection is an expensive operation, reusing connections can improve performance dramatically by cutting down on the number of new connections that need to be created.

The JDBC 2.0 standard specifies that connection pooling is implemented in the `DataSource` method. There are therefore no changes that are required to enable connection pooling.

Whether or not the connection returned by a call to the `DataSource.getConnection` method will be a pooled connection depends entirely on how the `DataSource` class being used has been implemented. If it has been implemented to work with a data source that supports connection pooling, a `DataSource` object will automatically return Connection objects that will be pooled and reused.

Just as there is no change in code to get a pooled connection, there is virtually no difference in the code for using a pooled connection. The only change is that the connection should be closed in a finally block, which is not a bad idea for closing any type of connection. That way, even if a method throws an exception, the connection will be closed and put back into the connection pool. The code for the finally block, which comes after the appropriate try and catch blocks, should look like this:

```
} finally {
  if (con != null) con.close():
}
```

This final block ensures that a valid connection will be recycled.

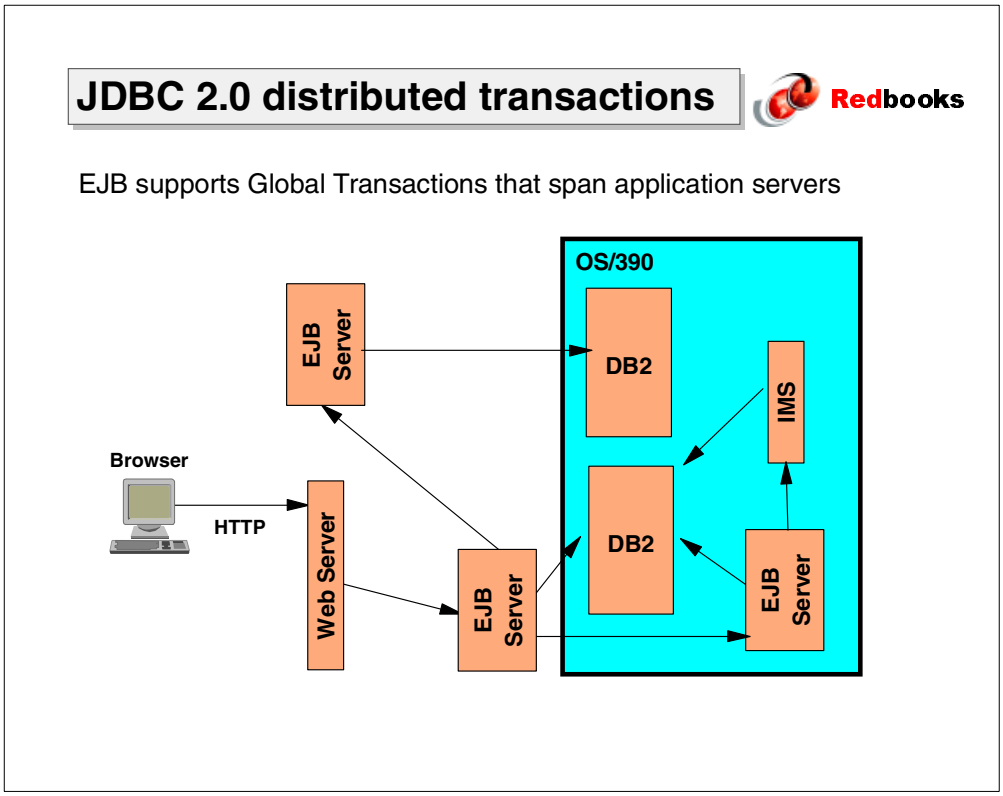
The foil shows the steps that are taken to satisfy a request for a database connection when connection pooling is being done.

1. When a `DataSource.getConnection()` in called by the application, the `DataSource` method performs a `lookup()` operation in the connection pool to see if there is a `PooledConnection` instance, (for example, a physical database thread), that can be reused.
2. If there is an available thread, the connection pool simply allocates the thread to the connection and returns the existing `PooledConnection` object to the `DataSource`. Otherwise a `ConnectionPoolDataSource` object is used to produce a new `PooledConnection` (not shown). In either case the connection pooling module returns a `PooledConnection` object that is ready to use. The `PooledConnection` object is implemented by the JDBC driver.
3. `PooledConnection.getConnection()` is then invoked to obtain a `Connection` object for the application to use.
4. The JDBC driver creates a `Connection` object. Remember this `Connection` object is really just a handle object that delegates most of its work to the underlying physical connection, or thread, represented by the `PooledConnection` object that produced it.
5. The `Connection` object is then returned to the application. The application uses the returned `Connection` object as though it is a normal JDBC Connection.

When the application is finished using the `Connection` object, it calls the `Connection.close()` method. The call to `Connection.close()` does not close the underlying physical connection represented by the associated `PooledConnection` object.

DB2 V7 implements JDBC 2.0 connection pooling through the `DataSource` method.

This is not to be confused with DB2 Database connection pooling, introduced with Type-2 inactive threads in DB2 UDB for OS/390 Version 6. Type-2 inactive threads can only be distributed threads, when JDBC threads are RRSAF threads connections.




### 3.4.6 JDBC 2.0 distributed transactions

The JDBC 2.0 Optional Package defines support for “Distributed Transactions”. DB2 UDB for OS/390 implements this support as “Global Transactions”.

To briefly introduce Global Transactions once more, consider this foil. We have a Web server that invokes an EJB application on an EJB server. This server in turn invokes EJBs at two other servers, and it also issues SQL directly to DB2 using JDBC or SQLJ. The other servers also interact with DB2 using JDBC or SQLJ. Lastly, one of the EJB servers invokes an IMS transaction that issues SQL using IMS Attach.

All these DB2 transactions can be part of the same global transaction. DB2 has been enhanced to recognize global transactions and share locks across branches of a global transaction. DB2, via a transaction processor (WebSphere through RRS in our example), also commits these DB2 threads as a single unit of work, “all or none”.

Refer to the redbook *DB2 UDB for OS/390 Version 6 Technical Update,* SG24-6108, for a detailed discussion of how global transactions are implemented in DB2 for OS/390. Refer also to page 278 for a discussion on global transactions in DB2 V7.

Obtaining a connection that can be used for global transactions is similar to that for getting a pooled connection. Again, the difference is in the way the `DataSource` class is implemented, not in the application code for obtaining the connection.

From the application programmer's point of view, there is practically no difference between a regular connection and a connection that can be used for global transactions. The only difference is that the transaction's boundaries, that is, when it begins and when it ends, are handled by a transaction manager behind the scenes. This means that the application should not do anything that could interfere with what the transaction manager is doing. So the application code cannot call the commit or rollback methods directly, and it cannot enable auto-commit mode, which calls the commit or rollback methods automatically when a statement is completed.

The `javax.transaction.UserTransaction` interface provides the application the ability to control transaction boundaries programmatically. This interface may be used by Java client programs or EJB beans. The `UserTransaction.begin` method starts a global transaction and associates the transaction with the calling thread. The transaction-to-thread association is managed transparently by the Transaction Manager.

Transaction context propagation between application programs is provided by the underlying transaction manager implementations on the client and server machines.

The EJB relies on the EJB Server to provide support for all of its transaction work as defined in the Enterprise JavaBeans Specification *(The underlying interaction between the EJB Server and the TM is transparent to the application)*.

JDBC in DB2 V7 supports global transactions, through the `DataSource` method. Global transactions are only supported through JDBC shipped with DB2 V6, using an OS/390 EJB server, which is WebSphere for OS/390 Version 4.

Transaction contexts are propagated between the EJB's by the EJB servers (which is WebSphere). JDBC, through the `DataSource` method, interfaces with RRS and passes the transaction context for each connection. RRS then indicates to DB2 whether the DB2 thread is globally coordinated by RRS or locally coordinated by DB2. DB2 assigns XID's to each thread with the same transaction context. This is how DB2 understands the different threads are part of the same global transaction.

The three methods that an application can exploit global transactions are:

1. A client uses *javax.transaction.UserTransaction* implementation provided by the Application Server to perform its own transaction demarcation. It finds this object by using JNDI services.
2. A client application calls an EJB with container managed transactions (set with the appropriate transaction attribute during deployment time).
3. A client application calls an EJB that manages its own transactions, using the *UserTransaction* object.

Here is an example of the first method:

// Get the system property value configured by administrator:

```
String utxPropVal = System.getProperty("jta.UserTransaction");
```

// Use JNDI to locate the *UserTransaction* object:

```
Context ctx = new InitialContext();
UserTransaction utx = (UserTransaction)ctx.lookup(utxPropVal);
```

// Start transaction work:

```
utx.begin();
```

// Use JNDI to locate a DB2 `DataSource` that provides *Connections* that can participate in a distributed transaction. The application has to know this JNDI name. The binding of the `DataSource` was done previously during "deployment time":

```
DataSource.ds = (DataSource)ctx.lookup("jdbc/someDB")'
Connection.con = ds.getConnection("myUseriD","myPasaword"0;
```

// Do work through `con.close()` statement:

// Note: Cannot call `commit, rollback or setAutoCommit` methods, since it is a part of a distributed transaction:

```
con.close();
```

// Must call close before the global commit:

```
utx.commit();
```

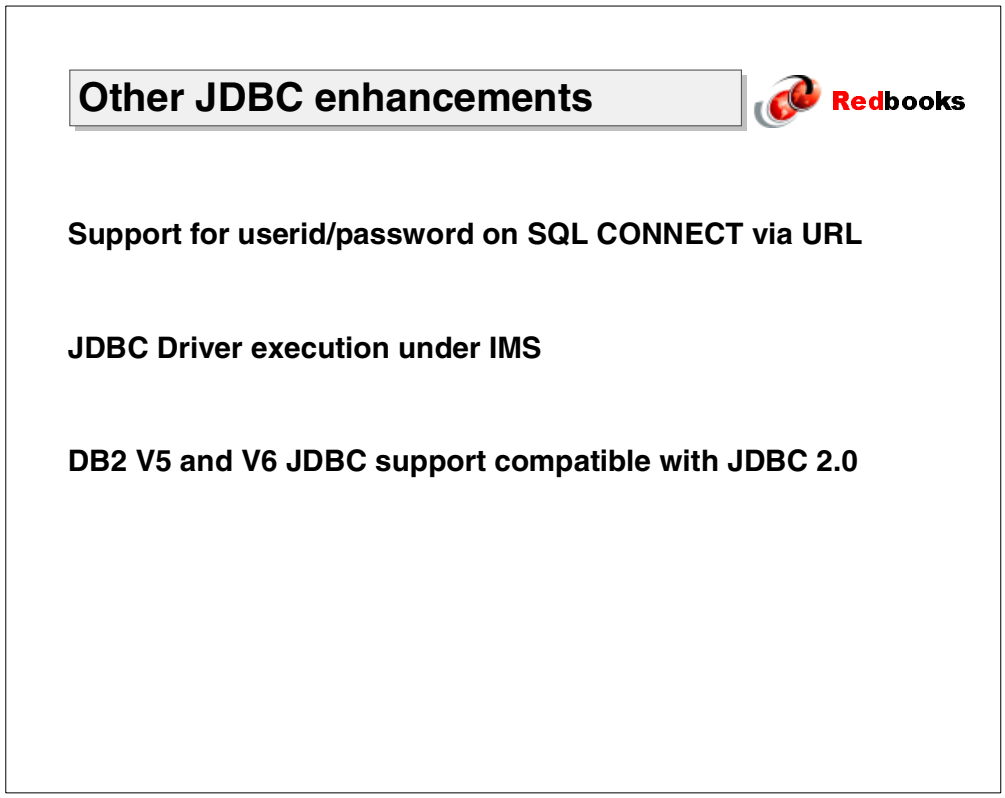
// Note: At this point the underlying physical database connection can now be reused for local or global transactions.

The last two methods are quite transparent to the client code, but may not be transparent to the EJB developer.

For the second method, the EJB has the transaction managed on its behalf by the container so all it has to do is obtain the `Connection` object from DB2 for OS/390's JDBC `DataSource` implementation, do some work, and close the *Connection*.

For the third method, code similar to the code above will be in a bean method, but rather than obtaining a `Context`, it will obtain a `SessionContext`.

The `UserTransaction` interface is used by Java client programs either through support from the application server or support from the transaction manager on the client host, to register global transactions. In OS/390, WebSphere V4 provides tools for an administrator to configure the `UserTransaction` object binding into JNDI. The `DataSource` method shipped with JDBC in DB2 for OS/390 also provides support for `UserTransaction` administration without JNDI.




## 3.4.7 Other JDBC enhancements

### 3.4.7.1 Support for userid/password usage on SQL CONNECT via URL

DB2 V7 enhanced the CONNECT statement, to allow you to specify a userid and password when connecting to a DB2 V7 DRDA server from an application running on OS/390.

Refer to 6.2.4, “CONNECT with userid and password” on page 295 for a discussion of this new enhancement.

JDBC has also been enhanced to accept and process the userid and password on the `getConnection` URL, when you connect to DB2 UDB for OS/390.

Prior to V7, DB2 ignores these parameters if they are specified on the URL.

### 3.4.7.2 JDBC Driver execution under IMS

The DB2 UDB for OS/390 JDBC driver will be enhanced so it can be used in an IMS program. This enhancement may not be available at general availability of DB2 V7, but should be available by APAR shortly after. The enhancement will also be made available for DB2 UDB for OS/390 Version 5 and Version 6 by an APAR.

The first implementation will see the JDBC code, compiled into the Java application load module by HPJ, connect to DB2 through the IMS Attach.

A subsequent implementation will see the JDBC code use the JVM and RRSAF, to connect directly to DB2 for OS/390.

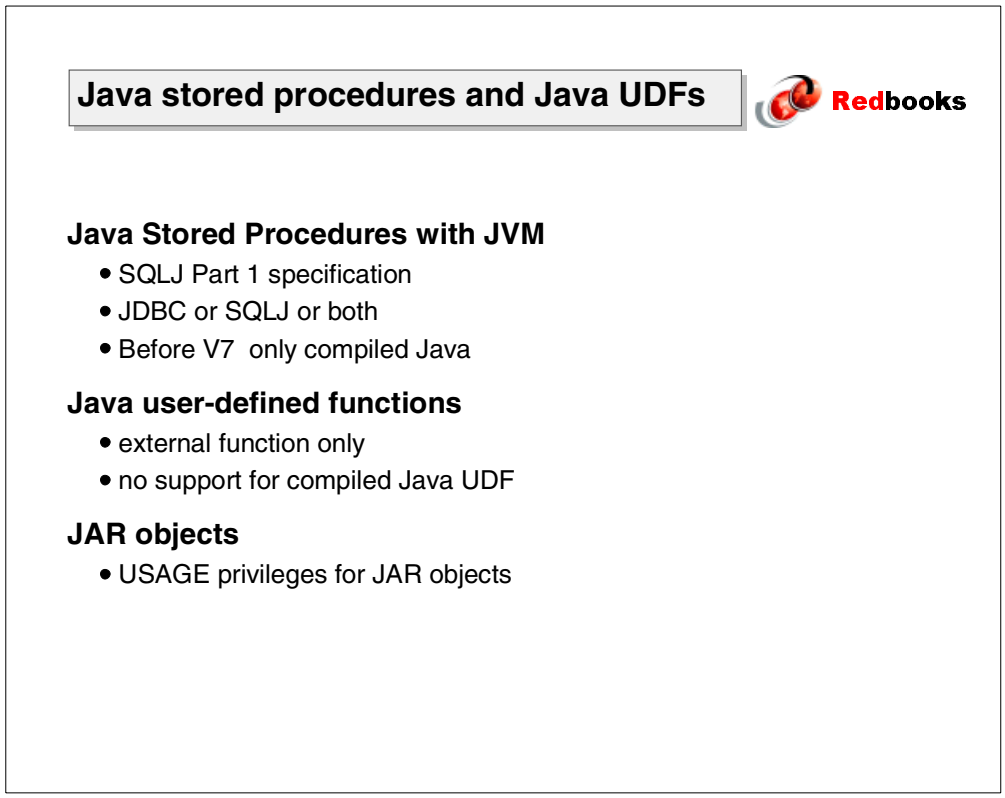
#### 3.4.7.3 “Compatibility” with Java 2

The Java Software Developers Kit (SDK) Versions 1.1.x provides Java developers with a set of tools to build enterprise applications. JDK 1.1.x implements a standard commonly known as Java, of which JDBC 1.x is a part.

JDK 1.2.x and above implements a standard commonly known as Java 2. Java 2 defines the standard for developing multi-tier enterprise applications using Java. JDBC 2.0 is included in the Java 2 standard.

Java 2 is shipped in the JDK for OS/390 Version 1.3.

The JDBC support shipped with DB2 UDB for OS/390 Version 5 and Version 6 will be enhanced to be compatible with Java 2, although no exploitation of Java 2 features.(These JDBC drivers currently only support JDK 1.2.)




## 3.5 Java stored procedures and Java UDFs

DB2 for OS/390 Stored Procedures can be written in many languages, including C, COBOL, Assembler, PL/1, REXX, SQL Procedure Language, and Java.

DB2 UDB for OS/390 Version 5 and Version 6 provides support for Compiled Java only. A Java Virtual Machine (JVM) is not required for Java stored procedures. They are compiled using the High Performance Java (HPJ) compiler, which is part of VisualAge for Java Enterprise Edition for OS/390. Compiled Java gives better performance than executing interpreted Java in a JVM.

For detailed instructions and advice, refer to the redbook *DB2 Java Stored Procedures: Learning by Example*, SG24-5945.

DB2 V7 extends Java support for stored procedures to allow Java stored procedures to also be executed in a JVM, (Interpreted Java). Refer to the American National Standard for Information Technology *SQLJ - Part 1 specification: SQL Routines using the Java Programming language*, which defines the installation of Java classes in an SQL database – Invocation of static methods as stored procedures.

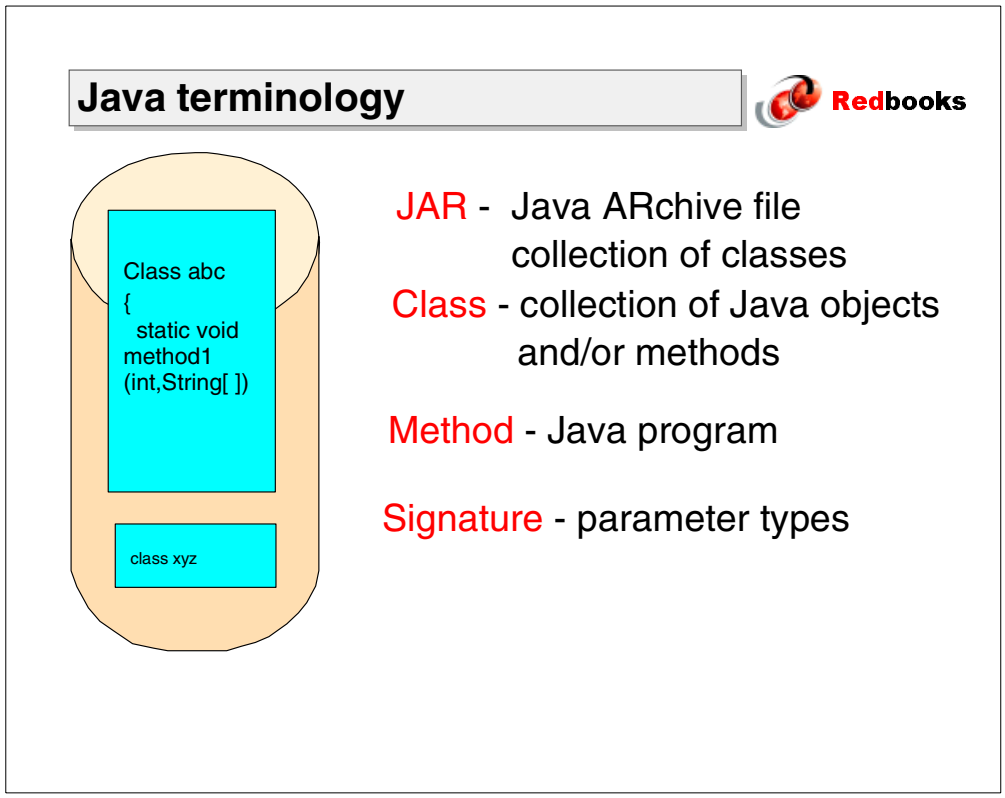
DB2 V7 also provides support for user-defined functions written in Java. DB2 provides support for external functions only.

DB2 UDB for UNIX, Windows, OS/2 supports Java (LANGUAGE JAVA) user-defined functions (external scalar and external table) and stored procedures. DB2 UDB for AS/400 only supports LANGUAGE JAVA for stored procedures.

DB2 V7 introduces a new object type of JAR and extends the GRANT and REVOKE commands to manage the USAGE of these new JAR objects. These JAR objects are stored in new DB2 catalog tables and are executed as interpreted stored procedures or functions.

Unlike compiled Java stored procedures which is executed in the WLM stored procedure address space, the Interpreted Java is invoked by the WLM stored procedure address space, but is executed in a Java Virtual Machine (JVM) under OS/390 UNIX System Services (USS).

This section will provide an overview of Java stored procedures and an outline of how to prepare them for both compiled Java and Interpreted Java.




### 3.5.1 Java terminology

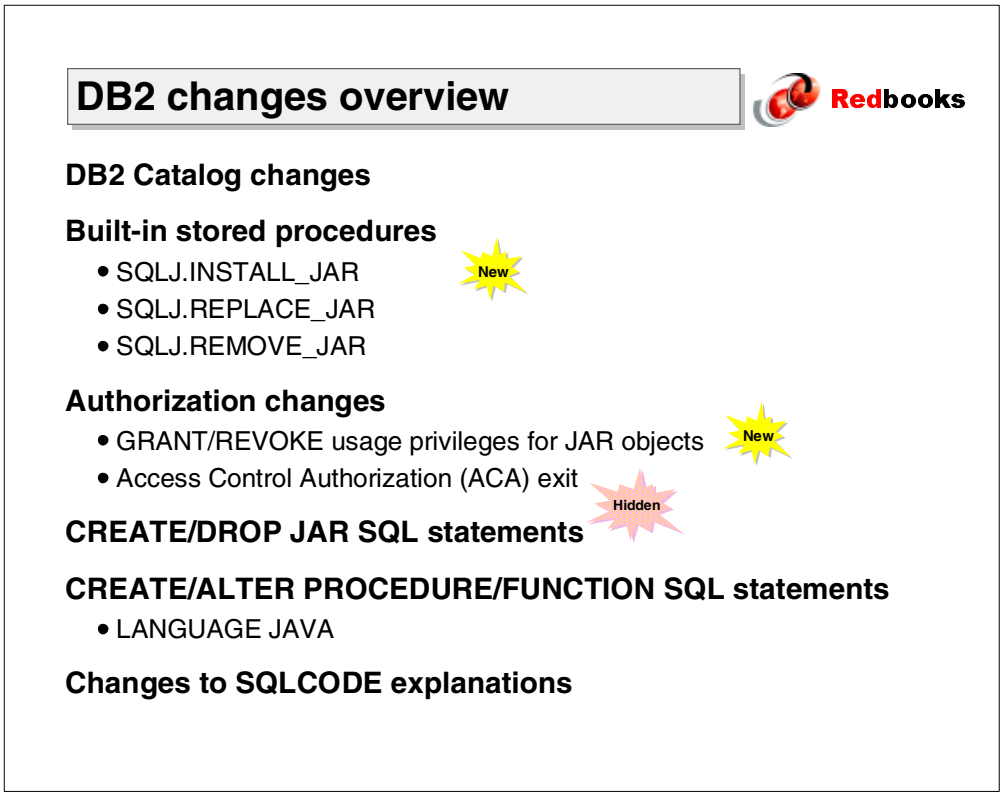
We will first review some basic terms used in Java programming.

***Java Class***
A Java Class contains one or more Java Methods and/or Objects and is identified by a class-name.

***JAR***
A Java ARchive file. This is a file that contains one or more Java classes in a compressed format.

***Java Method***
A Java program, identified by a method-name and exists in a Java class.

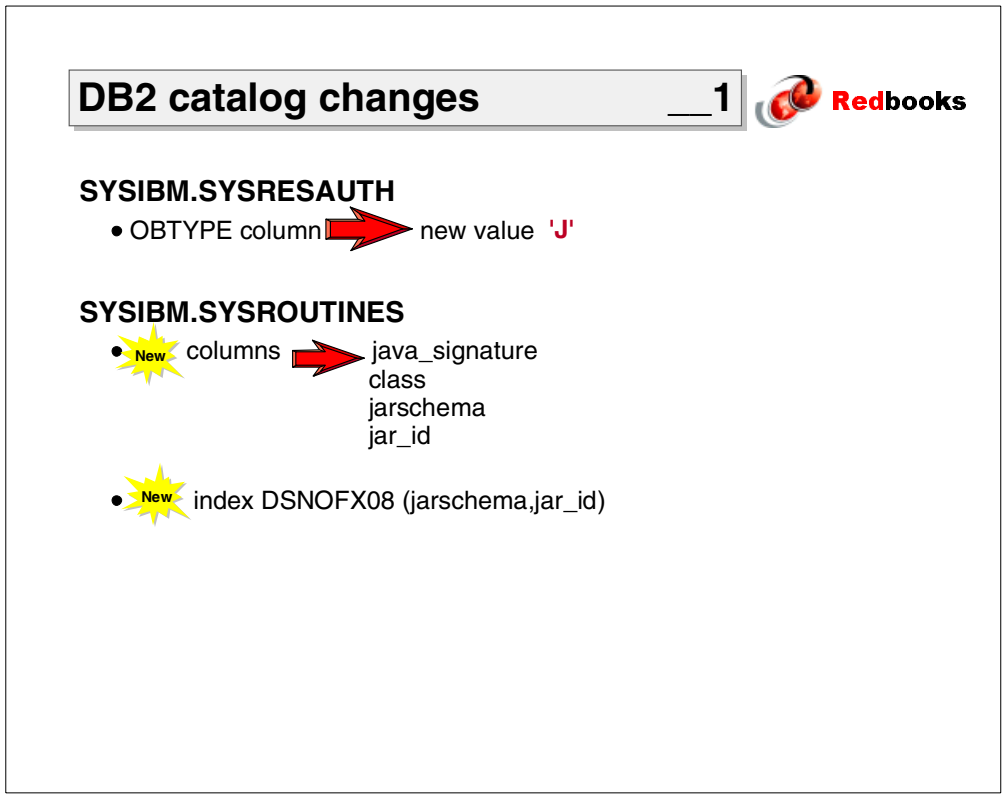
***Java Signature***
A list of parameters required by a Java method, or program.




### 3.5.2 DB2 changes overview

A number of enhancements have been made to DB2 to support interpreted Java stored procedures. These will be discussed in more detail in later foils:

- Three new catalog tables are created in a new tablespace, DSNDB06.SYSJAVA. These contain the Java class source for installed JAR files, the actual JAR file and options used to install the JAR file into DB2. In addition a number of new columns are added to the table SYSIBM.SYSROUTINES.
- Three new built-in stored procedures are shipped with DB2. They are used to install and manage JAR files into DB2.
- A new privilege class is created to manage JAR objects.
- Changes are made to the Access Control Authorization (ACA) exit interface, to accommodate the new Java objects.
- CREATE/ALTER PROCEDURE and CREATE/ALTER FUNCTION statements are enhanced to allow for a LANGUAGE of JAVA to be specified.
- A number of changes are also made to DB2 SQLCODE explanations.
- JAR is now a reserved word.

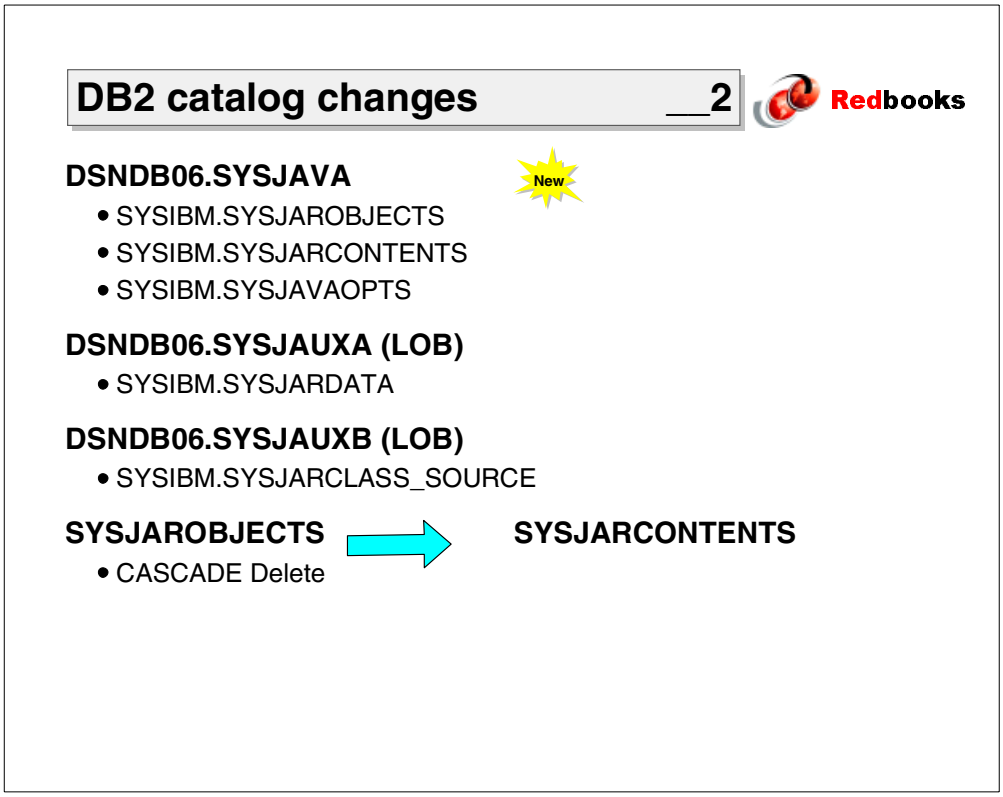



### 3.5.3 DB2 catalog changes

This foil summarizes the changes made to existing DB2 catalog tables:

- A new value of 'J' is added to OBTYPE column in SYSIBM.SYSRESAUTH. This indicates the row is used to store usage privileges associated with particular JAR objects.
- The catalog table SYSIBM.SYSROUTINES is extended to record the details of a Java classes which can be invoked as a interpreted Java stored procedures or functions. In addition, a new non-unique index is created on SYSIBM.SYSROUTINES, containing the columns JARSCHEMA and JAR_ID.

| Column Name | Data Type | Description |
|---|---|---|
| JAVA_SIGNATURE | VARCHAR(1024) | For an interpreted Java routine, the signature of the JAR file (not null with default) |
| CLASS | VARCHAR(128) | Name of the class in the JAR file (not null with default) |
| JARSCHEMA | CHAR(8) | Schema of the JAR file (not null with default) |
| JAR_ID | CHAR(18) | Name of the JAR file (not null with default) |




### 3.5.3.1 New DB2 catalog tables

DB2 V7 introduces a new table space into the DB2 catalog, DSNDB06.SYSJAVA, to contain three new catalog tables:

- SYSIBM.SYSJAROBJECTS records the contents of each JAR file for a Java stored procedure or function. The table has one unique index on JARSCHEMA, JAR_ID.

| Column Name | Data Type | Description |
|---|---|---|
| JARSCHEMA | CHAR(8) | Schema of the JAR file (not null) |
| JAR_ID | CHAR(18) | Name of the JAR file (not null) |
| OWNER | CHAR(8) | Authorization ID of the owner of the JAR file (not null with default) |
| JAR_DATA_ROWID | ROWID (Generated always) | ROWID that is used to support BLOB column JAR_DATA (not null) |
| JAR_DATA | BLOB(100M) | BLOB column for the BLOB that contains the contents of the JAR file (not null with default) |
| PATH | VARCHAR(1024) | The URL that represents the path for the source JAR file (not null with default) |

- SYSIBM.SYSJARCONTENTS records the classes in each JAR file for a Java stored procedure or function. The table has one non-unique index on JARSCHEMA, JAR_ID.

| Column Name | Data Type | Description |
|---|---|---|
| JARSCHEMA | CHAR(8) | Schema of the JAR file (not null with default) |
| JAR_ID | CHAR(18) | Name of the JAR file (not null with default) |
| CLASS | VARCHAR(128) | Name of the class in the JAR file (not null with default) |
| CLASS_SOURCE_ROW_ID | ROWID (Generated always) | ROWID that is used to support CLOB column CLASS_SOURCE |
| CLASS_SOURCE | CLOB(10M) | CLOB column for the CLOB that contains the contents of the class in the JAR file (not null with default) |

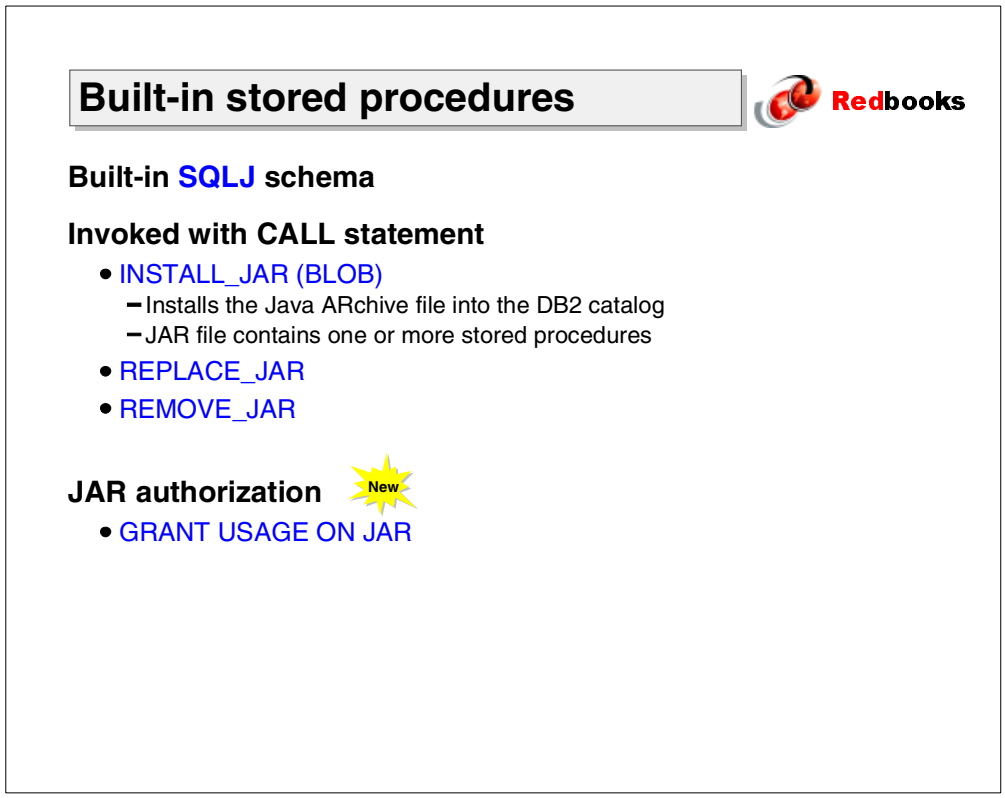
- SYSIBM.SYSJAVAOPTS records the build options for a Java stored procedure or function. The table has one non-unique index on JARSCHEMA, JAR_ID.

| Column Name | Data Type | Description |
|---|---|---|
| JARSCHEMA | CHAR(8) | Schema of the JAR file (not null) |
| JAR_ID | CHAR(18) | Name of the JAR file (not null) |
| BUILDSCHEMA | CHAR(8) | Schema name that is the qualifier for the procedure name that is specified in the BUILDNAME column (not null with default) |
| BUILDNAME | CHAR(18) | A procedure name that is associated with stored procedure DSNTJSPP (not null with default) |
| BUILDOWNER | CHAR(8) | Authorization ID that was used to create the Java routine (not null with default) |
| DBRMLIB | VARCHAR(128) | Name of the PDS that contains the DBRM for the routine (not null with default) |
| HPJCOMPILE_OPTS | VARCHAR(256) | HPJ compile options that are used when the routine is installed (not null with default) |
| BIND_OPTS | VARCHAR(1024) | Bind options that are used when the routine is installed (not null with default) |
| PROJECT_LIB | VARCHAR(128) | Name of the PDSE that contains the object code for the routine (not null with default) |

In addition, the table space DSNDB06.SYSJAUXA contains the table SYSIBM.SYSJARDATA which stores a BLOB where the contents of the JAR file for each Java stored procedure or function are stored. The table space DSNDB06.SYSJAUXB contains the table SYSIBM.SYSJARCLASS_SOURCE which contains a CLOB where the source code for a Java stored procedure or function is stored.

The tables SYSIBM.SYSJAROBJECTS and SYSIBM.SYSJARCONTENTS are defined in a referential integrity relationship. SYSIBM.SYSJAROBJECTS is a parent to SYSIBM.SYSJARCONTENTS through the columns JARSCHEMA and JAR_ID. The relationship is defined with a DELETE rule of CASCADE.

Not all these tables are used for interpreted Java stored procedures and functions. We shall see this later. The DB2 UDB Stored Procedure Builder (SPB) tool uses all these tables to store the Java source code and the invocation options, as well as the JAR file. This is to provide better support for compiled Java and interpreted Java stored procedures, providing extra functionality such as version control and better management of source code.




### 3.5.4 Built-in stored procedures

DB2 V7 ships three built-in stored procedures to install JAR files into DB2:

- SQLJ.INSTALL_JAR is used to install a JAR file into DB2
- SQLJ.REPLACE_JAR is used to replace a JAR file in DB2 with a new file.
- SQLJ.REMOVE_JAR is used to remove a previously installed JAR file from DB2.

These stored procedures are implemented as per the *SQLJ PART1: SQL routines using the Java programming language* ANSI specification.

DB2 V7 also ships a new built-in schema, SQLJ, which contains these built-in procedures.

SQLJ.INSTALL_JAR has the following parameters:

```
sqlj.install_jar (
      url IN VARCHAR(*),
      jar IN VARCHAR(*),
      deploy IN INTEGER)
```

SQLJ.REPLACE_JAR has the following parameters:

```
sqlj.replace_jar (
      url IN VARCHAR(*),
      jar IN VARCHAR(*),
      )
```

SQLJ.REMOVE_JAR has the following parameters:

```
sqlj.remove_jar (
      jar IN VARCHAR(*),
      undeploy IN INTEGER)
```

Assume you have assembled a Java class into JAR file with the local file name "~classes/myprog.jar"

```
SQLJ.INSTALL_JAR('file:~classes/myprog.jar","Myprog_jar",0)
```

The first parameter is a character string specifying the URL of the given JAR file. this parameter is never folded to uppercase.

The second parameter is a character string that is used as the name of the JAR file in DB2. The jar-name is used as a parameter of SQLJ.REMOVE_JAR and SQLJ.REPLACE_JAR procedures; as a qualifier of Java class names in CREATE PROCEDURE/FUNCTION statements; and as operands of grant and revoke statements.

The third parameter is a integer that specifies whether you do or do not (indicated by zero or non zero values) want the SQLJ.INSTALL_JAR procedure to execute the directions specified by the *deployment descriptor* in the JAR file. (These are essentially methods that can be executed as a install script or removal script to perform 'cleanup' actions, such as authorizations.)

DB2 V7 ships sample jobs to show you how to use the three new build-in stored procedures to install and manage JAR files in DB2. For more information, refer to the DB2 V7 standard manuals and sample libraries.

The new built-in stored procedures invoke new IBM internal SQL statements CREATE JAR and DROP JAR, to install and remove JAR file into the DB2 catalog.

Before invoking the SQLJ.INSTALL_JAR and SQLJ.REPLACE_JAR to install and change Java classes in DB2, you must perform these steps:

- Create the stored procedure into DB2 by issuing a CREATE PROCEDURE statement.
- Prepare the Java program:
  - Run *sqlj* translator.
  - Compile the Java program *javac*
  - Assemble the classes files into a single JAR file.

The CREATE PROCEDURE/FUNCTION SQL statement does the following:

- Registers a Java class as a stored procedure or function, by inserting a row into SYSIBM.SYSROUTINES.

SQLJ.INSTALL_JAR does the following:

- Execute a CREATE_JAR statement. This will define the JAR to DB2 and save the jar schema, jar id, owner (current sqlid) and jar file into the catalog table SYSIBM.SYSJAROBJECTS.
- Update JAR_DATA column in SYSIBM.SYSJAROBJECTS

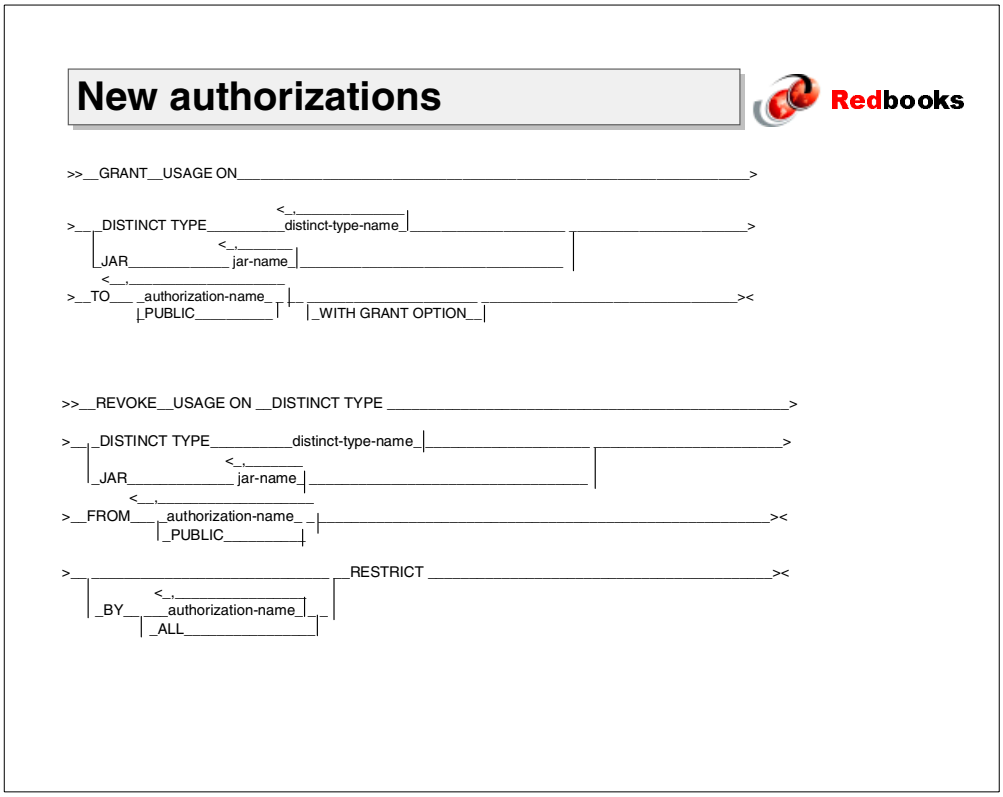
SQLJ.NSTALL_JAR does not extract all the class files that the JAR contains and install them into DB2. It only populates the SYSIBM.SYSJAROBJECTS table with the details of the actual JAR file. The table SYSIBM.SYSJARCONTENTS is not used.

SQLJ.REMOVE_JAR does the following:

- Execute a DROP_JAR statement. This will remove the JAR definition from the SYSIBM.SYSJAROBJECTS and clean up related JAR usage authorizations.

SQLJ.REPLACE_JAR does the following:

- Update the SYSIBM.SYSJAROBJECTS table with the new jar schema, jar id, owner (current sqlid) and jar file definitions.
- Update JAR_DATA column in SYSIBM.SYSJAROBJECTS.




### 3.5.5 New authorizations

***GRANT/REVOKE***

DB2 V7 introduces a new type of object named JAR. The GRANT and REVOKE commands have been enhanced to control the usage of JAR objects. This new privilege controls who can use CREATE PROCEDURE/FUNCTION SQL statements, to create a stored procedure or function using the JAR. The privilege also controls who can install and change the JAR in DB2.

JAR *jar-name* identifies the name, including the implicit or explicit schema name, of a unique JAR that exists at the current server. If you do not explicitly qualify the JAR name, it is implicitly qualified with a schema name according to the following rules:

- If the statement is imbedded in a program, the schema name is the authorization id in the qualifier bind option, or failing that, the owner of the package.
- If the statement is dynamically prepared the CURRENT SQLID special register is used.

Grants are recorded in the SYSIBM.SYSRESAUTH table with a new OBJTYPE if ‘J’.

Like collection authorities, the JAR does not need to exist in DB2 before you can grant use of the JAR to another authorization id.

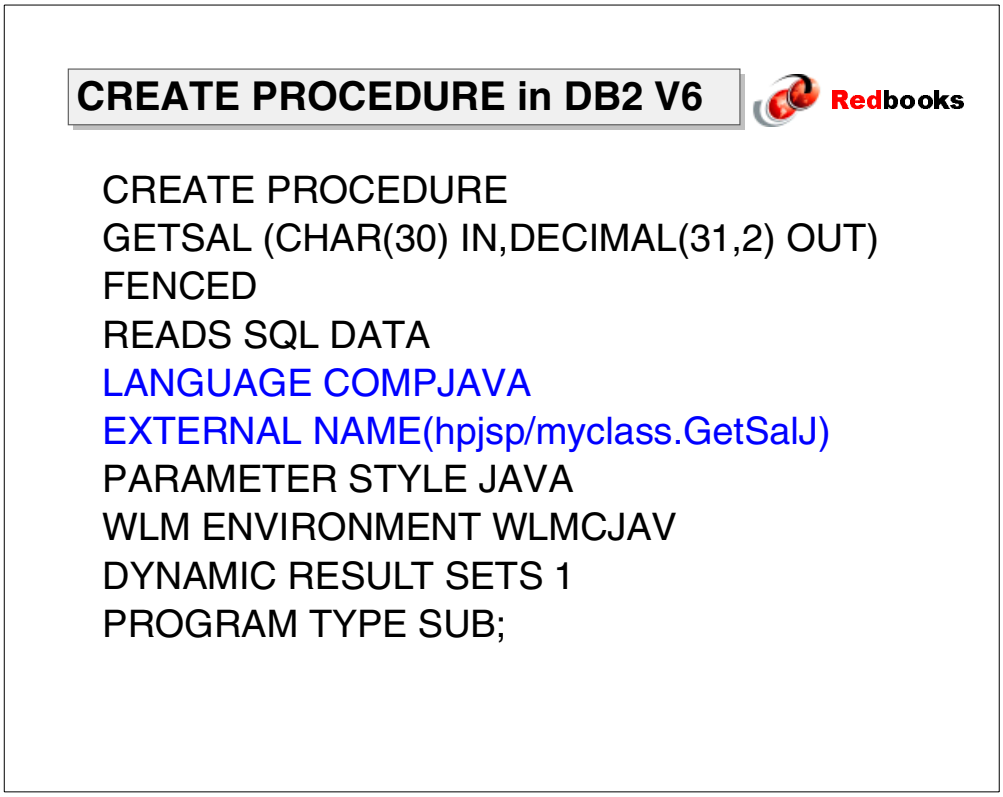
The RESTRICT clause on the REVOKE statement prevents the USAGE privilege from being revoked on a JAR if the revoker owns a stored procedure of function that references this JAR.

There exists one migration/fallback consideration with the new authorization privilege. If authorization id A, which has SYSADM authority, grants USAGE on a JAR in DB2 V7, another authorization id cannot revoke SYSADM authority from authorization id A in a previous release of DB2.

***Access Control Authorization (ACA) exit***

The Access Control Authorization (ACA) exit is used to control DB2 authorizations through an external security product, such as RACF.

The exit parameter list, DSNDXAPL, which is passed to DSNX@XAC, is enhanced to pass JAR usage authorizations. A new value of 'J' has been added to the field XAPLTYPE. The explanation for field XAPLOBJN has also been enhanced to include JAR objects.




### 3.5.6 CREATE PROCEDURE in DB2 V6

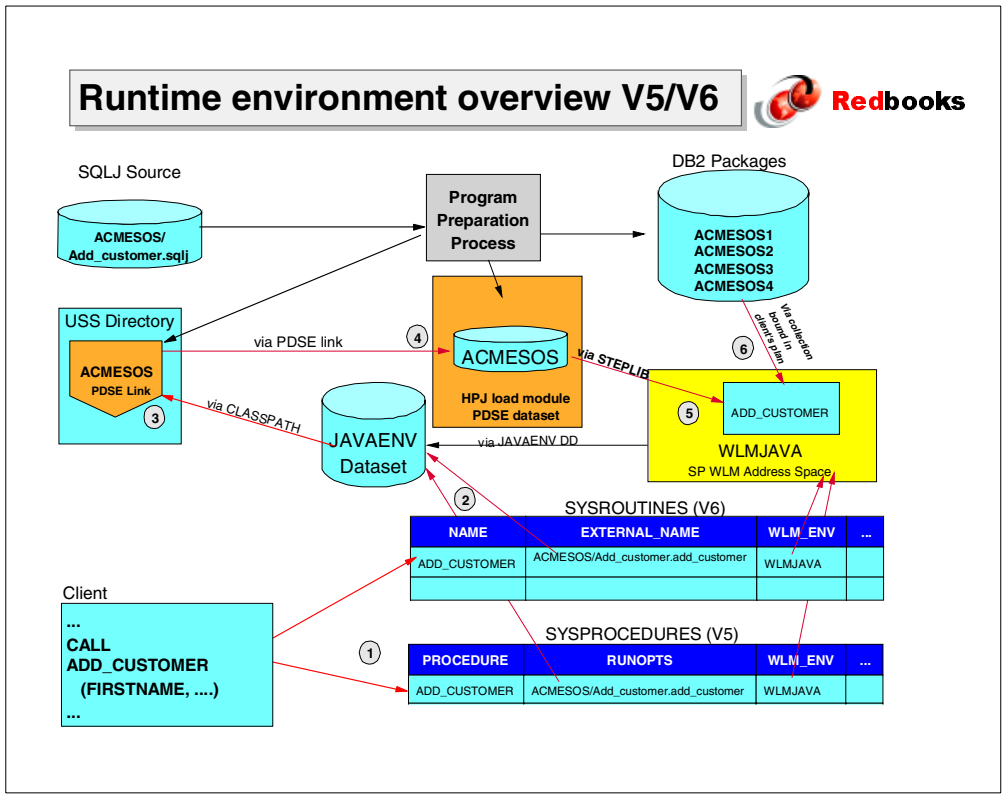
You define a Java stored procedure to DB2 using the CREATE PROCEDURE statement in the same way as for stored procedures written in other languages.

Note that the following parameters have different meanings for Java stored procedures: EXTERNAL specifies the program that runs when the procedure name is specified in a CALL statement. For Java stored procedures, the form is EXTERNAL NAME ‘class-name.method-name’ which is the name of the Java executable code that is created by the HPJ compiler. If the class is defined in a package, it is prefixed with the package name.

The following parameters must be specified:

- LANGUAGE COMPJAVA
- PARAMETER STYLE JAVA — required so that DB2 uses a parameter passing convention that conforms to the Java language and SQLJ specifications
- WLM ENVIRONMENT — Java has to run in a workload managed environment.
- PROGRAM TYPE SUB — Java stored procedures cannot run as MAIN routines

RUN OPTIONS will be ignored if you specify any. Because the Java Virtual Machine (JVM) is not destroyed between executions, language environment options cannot be specified for an individual stored procedure.




### 3.5.7 Runtime environment overview V5/V6

The program preparation process for a compiler Java program results in the following objects being produced:

- An OS/390 load module for the stored procedure, stored in a PDSE data set.
- A link to the load module, within a nominated library under OS/390 UNIX Systems Services (USS).
- A DB2 package, if the stored procedure accesses DB2 using SQLJ (JDBC procedures use a generic JDBC package).
- A serialized profile, if the stored procedure accesses DB2 using SQLJ (again, JDBC procedures use a generic JDBC profile).

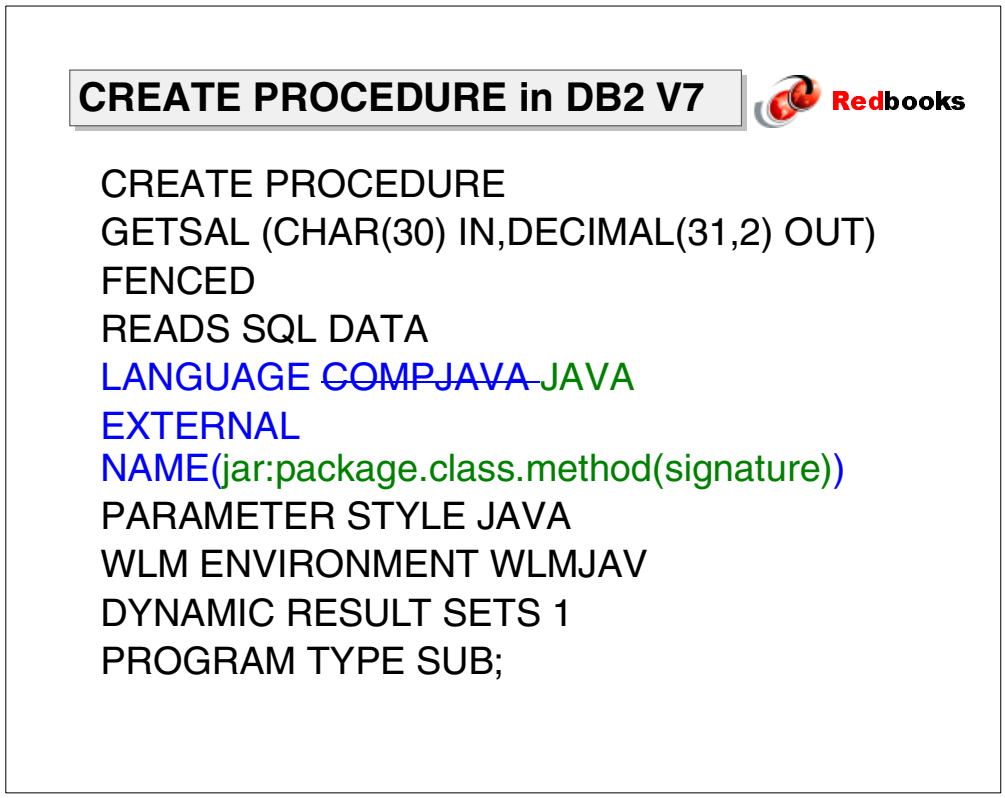
This foil shows how these objects are used when a client issues a call to a Java SQLJ stored procedure.

1. The client issues a call to the stored procedure `ADD_CUSTOMER`.
2. Depending upon the version of DB2 for OS/390 being used, either the `RUNOPTS` column of `SYSIBM.SYSPROCEDURES` or the `EXTERNAL_NAME` column of `SYSIBM.SYSROUTINES` is used to determine the Java package, class, and method associated with the call.
3. This is defined within the catalog according to standard Java syntax:

   `package/classname.methodname`

4. The JAVAENV data set specified in the procedure JCL for the selected WLM environment is used to obtain the value of the CLASSPATH parameter. The USS libraries specified in s CLASSPATH are searched to find a link named after the package name specified in the SYSROUTINES or SYSPROCEDURES definition (ACMESOS in our example).
5. The link is used to obtain the PDSE member name containing the HPJ compiled stored procedure load module for the Java package being used.
6. This load module is loaded into the relevant WLM stored procedure address space.
7. DB2 searches the package list associated with the client's plan to find the package for the stored procedure.

**Note**: For JDBC stored procedures, procedure-specific DB2 packages are not produced. DB2 searches the package list associated with the client's plan to find the generic JDBC packages, and these will be used instead.




### 3.5.8 CREATE PROCEDURE in DB2 V7

DB2 V7 enhances the CREATE PROCEDURE statement to allow you to create a Java stored procedure that will execute in a Java Virtual Machine (JVM).

The CREATE PROCEDURE/FUNCTION SQL statement allows you to specify a SQL name for a Java method. The Java method for interpreted Java, exists in a JAR file which is installed into DB2 by the SQLJ.INSTALL_JAR built-in stored procedure. A JAR file can contain any number of Java classes which can be invoked as DB2 stored procedures or functions. You can also reference the same Java class by more than one stored procedure name, by executing different CREATE PROCEDURE/FUNCTION SQL statements, referencing the same Java class.

Although there are no syntax changes, two parameters are enhanced. The LANGUAGE parameter now accepts JAVA, which indicates the stored procedure or function is written in Java and the Java Byte code will be executed in the OS/390 JVM, under USS. The EXTERNAL parameter which specifies the program that runs when the procedure name is specified in a CALL statement, is also extended. If LANGUAGE is JAVA then the EXTERNAL NAME clause defines a string of one or more *external-java-routine-name(s)*, enclosed in single quotes.

The CREATE PROCEDURE/FUNCTION SQL statements specify the SQL names and signatures for the Java methods specified in the external name. The format of the method names in the external names clause consists of the JAR name that was specified in the SQLJ.INSTALL_JAR stored procedure followed by the Java method name, fully qualified with the package name(s) if any, and class name.

The CREATE PROCEDURE/FUNCTION SQL statements do the following:

- Inserts a row into SYSIBM.SYSROUTINES.
- Inserts the schema-name from jar-name into JAVASCHEMA of SYSIBM.SYSROUTINES. If jar-name was not specified on the create statement then this column is left blank.
- Inserts jar-id into JAR_ID of SYSIBM.SYSROUTIONES. If jar-name was not specified on the create statement then this column is left blank
- Inserts method-name, which includes class-id, method-id and optional list of package ids, into EXTERNAL_ NAME of SYSIBM.SYSROUTINES. The does not include the jar-name or method-signature.
- Inserts the first 128 bytes of method-name, excluding periods and method-id, into CLASS of SYSIBM.SYSROUTINES. The package-id and class-id will be stored in both EXTERNAL_NAME and CLASS of SYSIBM.SYSROUTINES, (to help narrow the searches in the catalog)
- The method-signature is stored in JAVA_SIGNATURE column of SYSIBM.SYSROUTINES. This is left blank if method-signature was not specified on create statement, or contains () when an empty set of parentheses was specified.

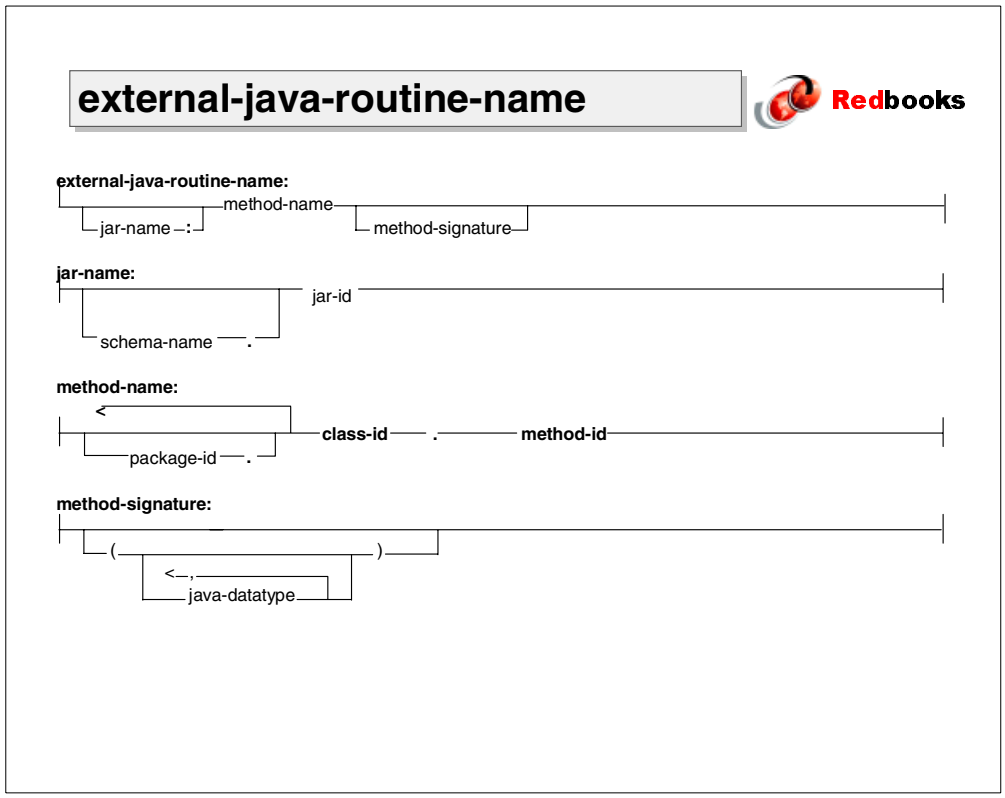
The external-java-routine-name does not have to exist when the CREATE PROCEDURE/FUNCTION SQL statement is executed, however it must exist and be accessible from the DB2 server when the procedure is called.

The following parameters must be specified for CREATE/ALTER PROCEDURE:

- LANGUAGE COMPJAVA/JAVA - COMPJAVA is kept for compatibility with the previous V6 support through HPJ but it also needs the EXTERNAL NAME to point to the HPJ executable code, JAVA indicates the new JVM execution.
- PARAMETER STYLE JAVA - required so that DB2 uses a parameter passing convention that conforms to the Java language and SQLJ specifications.
- WLM ENVIRONMENT - Java has to run in a workload managed environment.
- PROGRAM TYPE SUB - Java stored procedures cannot run as MAIN routines.
- RUN OPTIONS - must not be specified with language JAVA.
- DBINFO must not be specified with language Java.

The following parameters must be specified for CREATE/ALTER FUNCTION:

- LANGUAGE JAVA - Interpreted Java support, compiled Java is not supported.
- PARAMETER STYLE JAVA - required so that DB2 uses a parameter passing convention that conforms to the Java language and SQLJ specifications.
- WLM ENVIRONMENT - Java has to run in a workload managed environment.
- FINAL CALL - must not be specified when LANGUAGE JAVA for functions.
- SCRATCHPAD - must not be specified when LANGUAGE JAVA for functions.
- PROGRAM TYPE SUB — Java functions cannot run as MAIN routines.
- RUN OPTIONS - must not be specified with language JAVA.
- DBINFO - must not be specified with LANGUAGE JAVA.




### 3.5.9 The external-java-routine-name

The *external-java-routine-name* is specified on the EXTERNAL_NAME parameter of the CREATE PROCEDURE/FUNCTION SQL statement.

When LANGUAGE is specified as COMPJAVA, the EXTERNAL_NAME parameter must contain a string containing package-id, class-id and method-id, which defines the Java code to execute.

When LANGUAGE is specified as JAVA, the EXTERNAL_NAME parameter contains an external-java-routine-name. An external-java-routine-name contains:

***jar-name:***
Identifies the name given to the JAR when it was installed into DB2. The name contains *jar-id* which can optionally be qualified with a *schema*. Examples are `MyJar` or `Myschema.MyJar`. The unqualified jar-id is implicitly qualified in the following way:

- If the invoking SQL statement is imbedded in a program, the schema name is the authorization id in the qualifier bind option, or failing that, the owner of the package.
- If the invoking SQL statement is dynamically prepared the CURRENT SQLID special register is used.

***method-name:***
Identifies the name of the method. It is built from an optional list of one or more *package-id*s, that identifies the packages that the class identifier is a part of. Followed by a *class-id* which identifies a class identifier of the Java object. Followed by *method-id* which identifies a method with the Java class to be invoked.

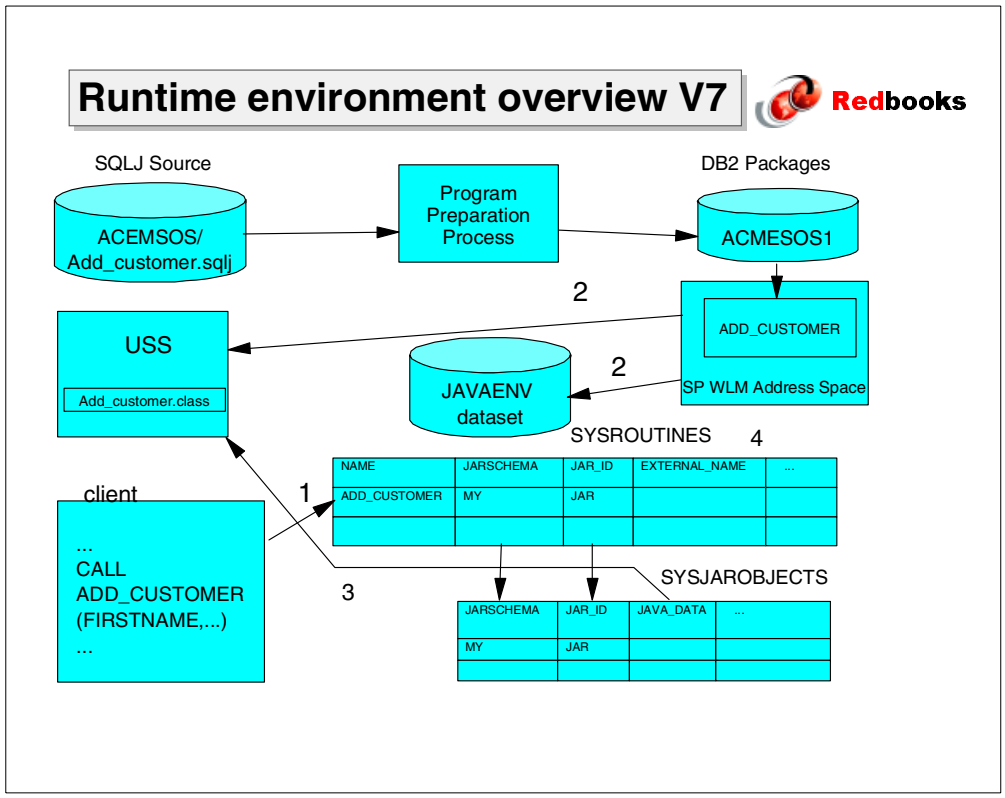
***method-signature:***
Optimally a method-signature, which identifies a list of zero or more Java data types for the parameter list.

The jar-name (jar-schema, jar-id) and method-signature items are optional parameters on the external-java-routine-name specification. If they are not specified on the CREATE PROCEDURE statement, the Java class files must exist in a USS directory that is in the `CLASSPATH` statement in the `JAVAENV` parameters of the WLM stored procedure environment. When the stored procedure is invoked, DB2 will execute the specified class/method out of the USS directory.

Signature validation is the process of checking the list of parameters specified in the method-signature, with the parameters that the specified Java class expects. This is not done by the CREATE PROCEDURE/FUNCTION SQL statement but the signature is validated at run time. (It would be desirable for the DDL to perform as much validation checking as it can during create time. However the CREATE PROCEDURE SQL statement must then invoke the JVM to extract the classes in the JAR file. This is not desirable.)

When the stored procedure is being invoked, DB2 searches for a Java method with the exact method-signature. The Java data types are used to determine which Java method to invoke.

A Java procedure can have no parameters. in this case coded am empty set of parentheses for method-signature, If a Java method-signature is not specified, DB2 will search for a Java method with a signature derived from the default JDBC types associated with the SQL types specified in the parameter list of the create procedure statement.




### 3.5.10 Runtime environment V7

The program preparation process for an interpreted Java program, results in the following objects being produced:

- A single JAR file containing all the Java classes and serialized profiles (if the stored procedure accesses DB2 using SQLJ, remembering that JDBC procedures use a generic JDBC profile). The JAR file is stored in the DB2 catalog.
- A DB2 package, if the stored procedure accesses DB2 using SQLJ (JDBC procedures use a generic JDBC package).

This foil shows how these objects are used when a client issues a call to a Java SQLJ stored procedure.

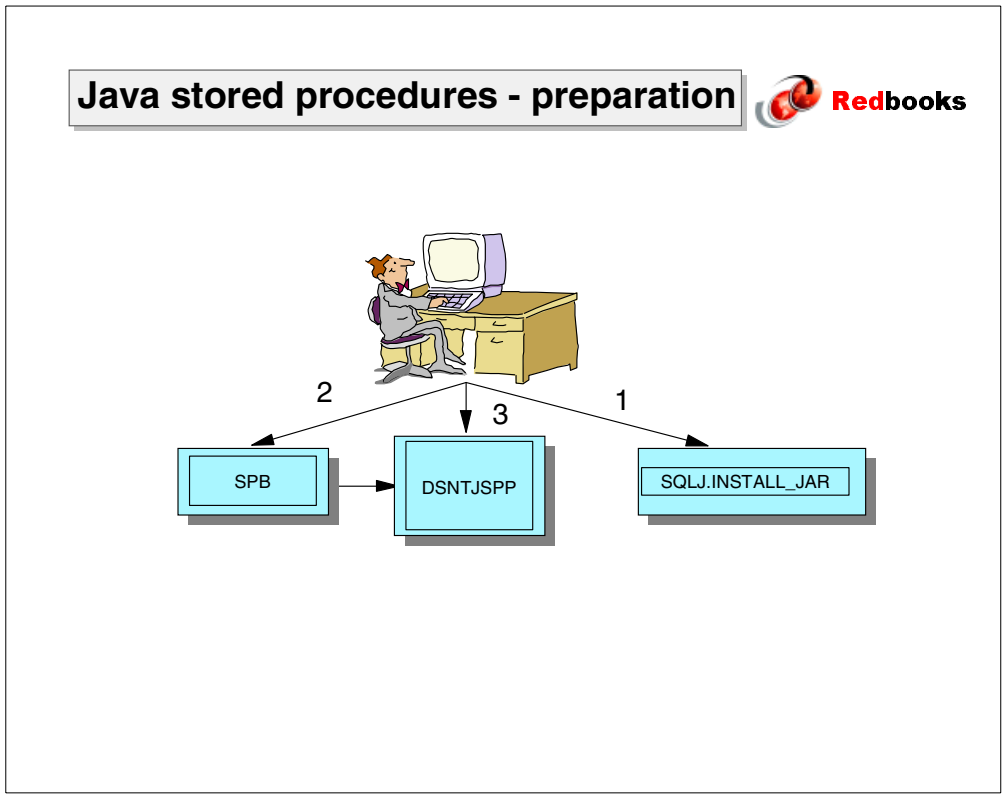
1. The client issues a call to the stored procedure `ADD_CUSTOMER`.
2. The `JAVAENV` data set specified in the procedure JCL for the selected WLM environment is used to obtain the value of the `CLASSPATH` parameter. The USS libraries specified in `CLASSPATH` are used to build the JVM environment where the Java class will execute.
3. The JAR file in the column `JAR_DATA` of `SYSIBM.SYSJAROBJECTS`, corresponding to the java_name in `SYSIBM.SYSROUTINES`, (columns `JARSCHEMA`, `JAR_ID`), is loaded into the JVM and the Java classes are extracted.

4. DB2 searches the Java classes looking for a Java class for a matching signature definition. The EXTERNAL_NAME column of SYSIBM.SYSROUTINES is used to determine the Java package, class, method and method signature in the JAR file which is associated with the call.
5. This is defined within the catalog according to standard Java syntax:

   ```
   package/classname.methodname.methodsignature
   ```

6. DB2 searches the package list associated with the client's plan to find the package for the stored procedure.

DB2 executes Interpreted Java stored procedures and functions using the OS/390 JDK 1.1.8 and above. Using the JDK 1.1.8, the JVM is created and destroyed each time the Java stored procedure is invoked. The OS/390 JDK 1.3 should overcome this problem.




### 3.5.11 Java stored procedures - preparation

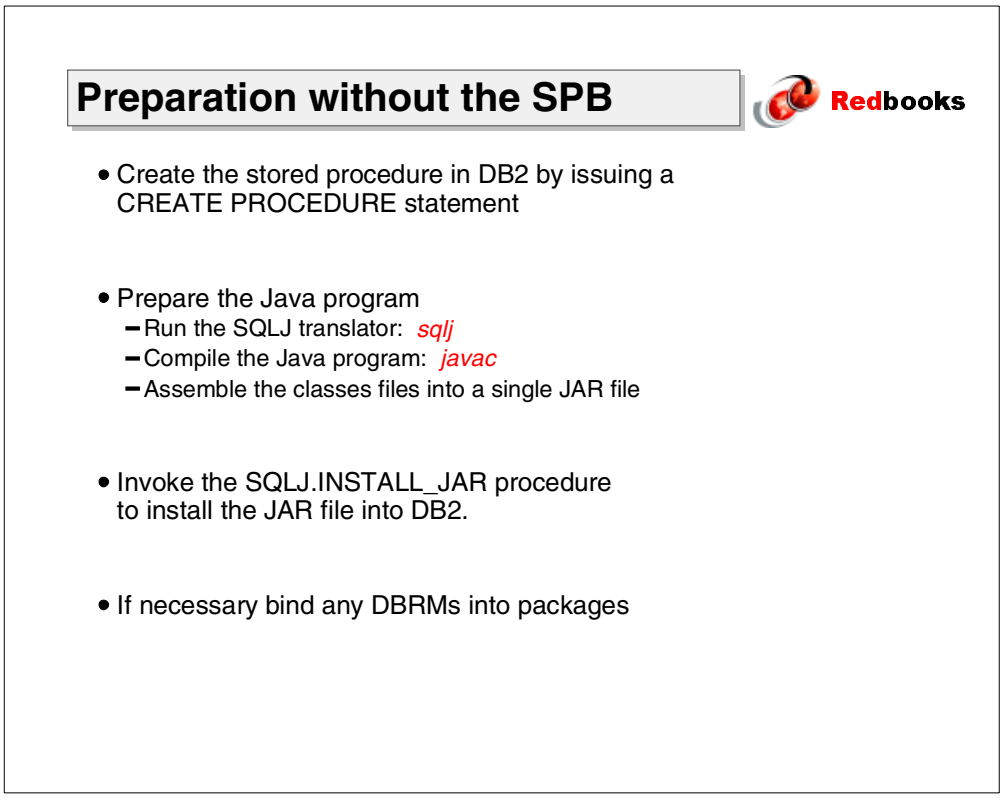
You can create a Java stored procedure in two different ways:

1. Use the JAR command to put the Java class files and any user files into a single JAR file. Then invoke the new DB2 built-in stored procedures, sending the JAR file as a BLOB. This defines the JAR name to DB2 and stores the binary JAR file in the DB2 catalog. Finally issue the CREATE PROCEDURE/FUNCTION SQL statement, to create the Java stored procedure or function.
2. Use the DB2 UDB Stored Procedure Builder (SPB) tool to write your Java, install the Java code into DB2 and prepare it for execution. The SPB also issues the CREATE PROCEDURE/FUNCTION SQL statements. To do all these functions the SPB invokes a stored procedure written in REXX, called DSNTJSPP, to interface to DB2 on OS/390.

Finally, you can also write an application yourself to interface with the stored procedure DSNTJSPP, to install the JAR file into DB2 and prepare the Java stored procedure and make it available for use.

We shall go into each method in a little more detail in the next few foils.

Note that the SPB can only be used to prepare Compiled Java stored procedures. Java functions for DB2 cannot be written and prepared using the SPB. The SPB only generates compiled Java at this time, however it is also intended to support interpreted Java at a later date.

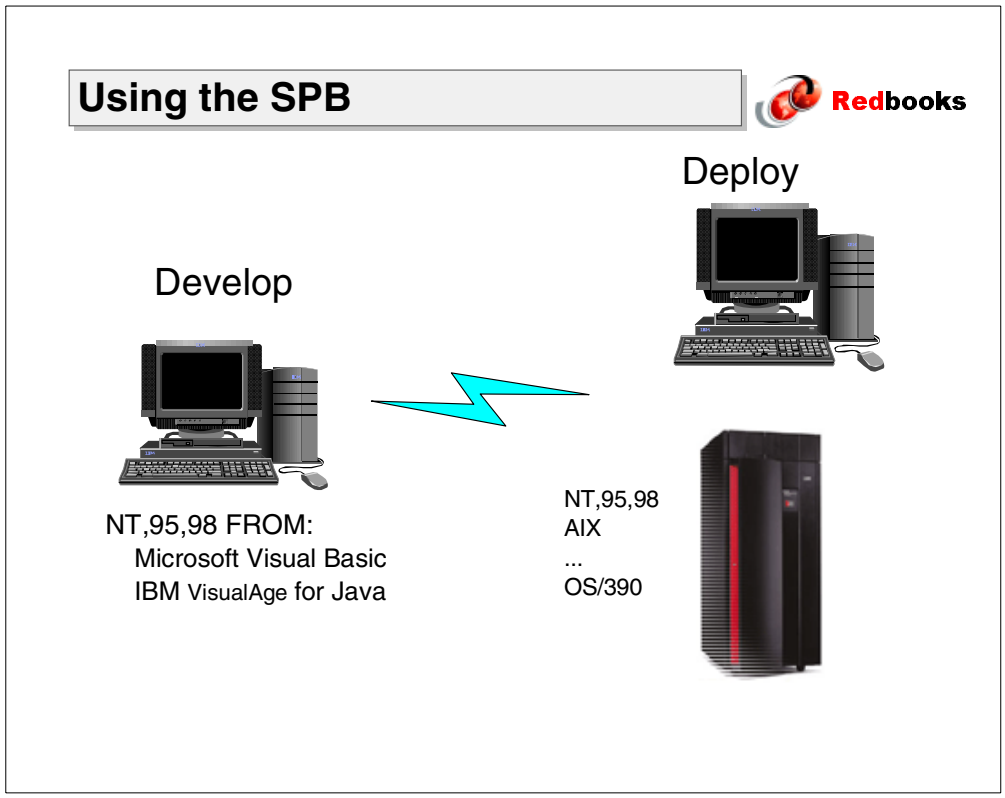



### 3.5.12 Preparation without the SPB

The processes you follow to install and make an interpreted Java class available as a stored procedure or function are:

- Create the stored procedure in DB2 by issuing a CREATE PROCEDURE statement.
- Prepare the Java program:
  a. Run *sqlj* translator.
  b. Compile the Java program *javac.*
  c. Assemble the classes files into a single JAR file.
- Invoke the SQLJ.INSTALL_JAR procedure to install the JAR file into DB2.
- If necessary bind any DBRMs into packages

IBM provide a number of sample jobs to show how to invoke the new built-in stored procedures which will register and load the JAR file into DB2.




### 3.5.13 Using the SPB

The DB2 UDB Stored Procedure Builder (SPB) tool is a graphical tool designed to help with the development of DB2 stored procedures. It provides all the functions required to create, build, test, and deploy new stored procedures. It also provides functions to work with existing stored procedures.

The SPB provides a single development environment that supports the entire DB2 family ranging from the workstation to OS/390. With the SPB you can focus on the logic of your stored procedure rather than on the process details of creating stored procedures on a DB2 server.

It is important to notice that the SPB is not a prerequisite to write stored procedures for DB2 servers. The support for stored procedures is built-in to the DB2 base code.

In summary, using the SPB you can perform a variety of tasks associated with stored procedures, such as:

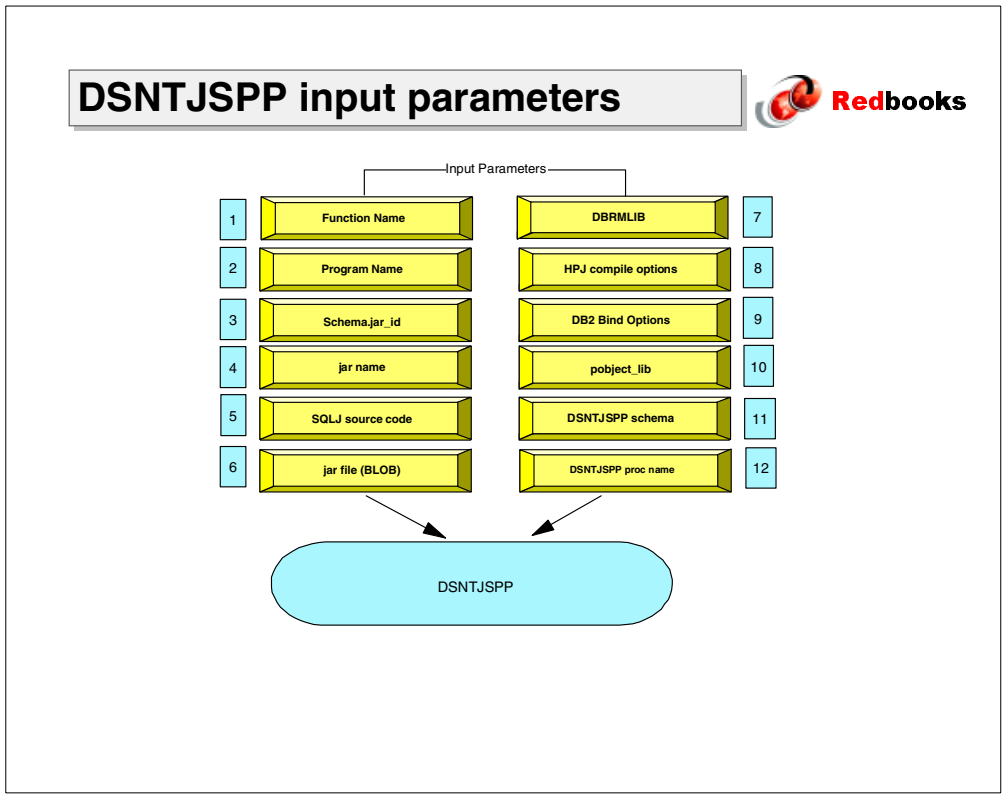
- Creating new stored procedures.
- Listing existing stored procedures.
- Modifying existing stored procedures (Java and SQL stored procedures).
- Running existing stored procedures.
- Copying and pasting stored procedures across connections.
- One-step building of stored procedures on target databases.

- Customizing the settings to enable remote debugging of installed stored procedures.

You can use the DB2 UDB Stored Procedure Builder (SPB) tool to write your Java, install the Java code into DB2 and prepare it for execution. The SPB issues the CREATE PROCEDURE/FUNCTION SQL statements for you and even binds any DBRMs that need to be bound. To do all these functions the SPB invokes a stored procedure written in REXX, called DSNTJSPP, to interface to DB2 on OS/390.

Today the Stored Procedure Builder (SPB) tool will install the JAR as compiled Java only, but it will be enhanced to also register interpreted Java.

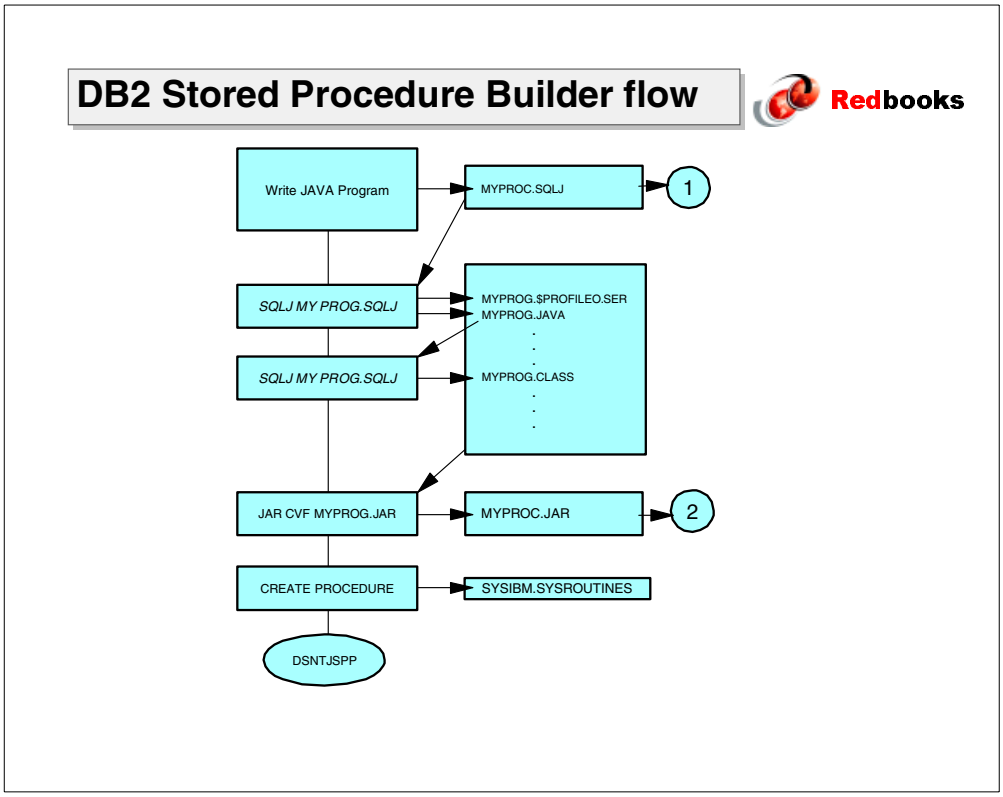
The SPB also stores the JAR file, the Java source code and the SPB invocation options in DB2 tables. This is to provide better support for compiled Java and interpreted Java stored procedures, providing extra functionality such as version control and better management of source code.




### 3.5.14 DSNTJSPP input parameters

DSNTJSPP is another new stored procedure that is invoked from the DB2 UDB Stored Procedure Builder (SPB) tool. It automatically performs all the steps you need to install Java classes into the DB2 catalog and also perform the steps necessary to make those classes an executable stored procedure. DSNTJSPP is designed to work primarily with the SPB tool. However, DSNTJSPP will also work with any caller capable of supplying the correct input parameters:

1. Function name (`INSTALL_JAR`, `REPLACE_JAR`, `REMOVE_JAR`) — varchar(20)
2. Program name (input to `db2profc`) — varchar(7)
3. The schema.jar_id — varchar(27)
4. The jar_name (HFS path name where jar should be installed) — varchar(128)
5. The sqlj source code — CLOB
6. The jar file — BLOB
7. DBRM library — varchar(128)
8. HPJ compile options
9. DB2 bind options — varchar(1024)
10. The pobject_lib (PDSE name, location where program object resulting from `INSTALL_JAR` should be placed) — varchar(1228)
11. The schema name of DSNTJSPP — varchar(8)
12. The procedure name of DSNTJSPP — varchar(18)

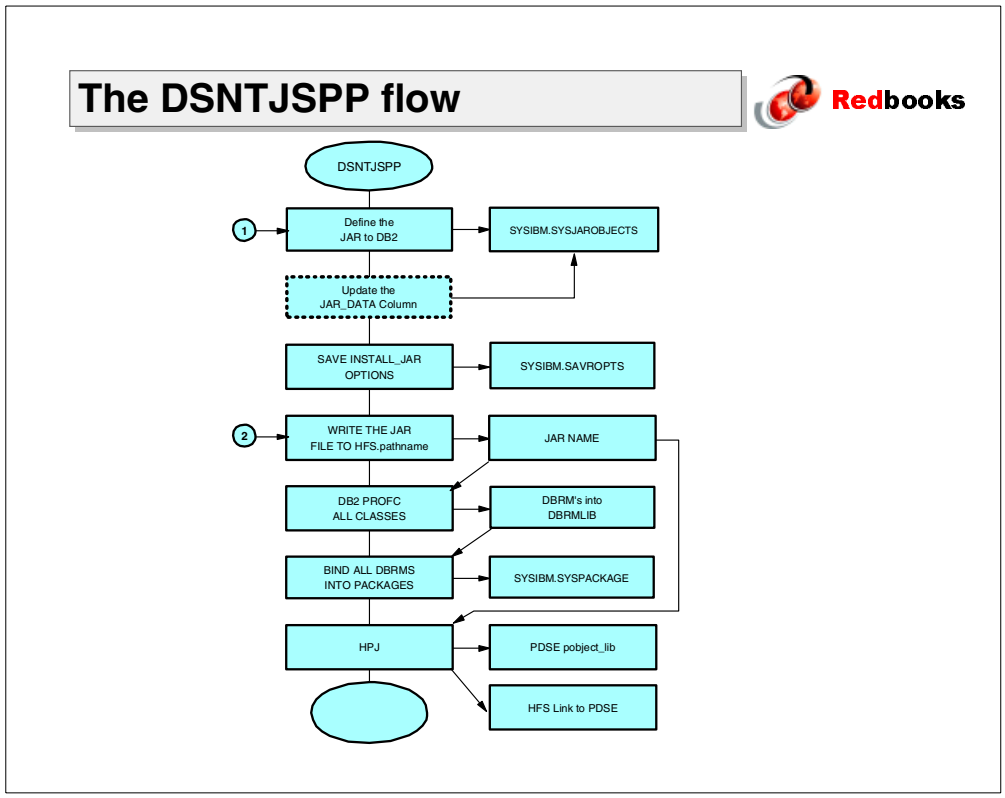



### 3.5.15 Stored Procedure Builder flow

Before invoking DSNTJSPP you must perform these tasks. All these tasks, except for writing the actual Java code, are all done for you by the SBP tool:

- Write the Java code.
- Prepare the Java code:
  - Run the *sqlj* translator.
  - Compile the Java code using *javac.*
  - Assemble the class files into a single JAR file.

The SPB then invokes DSNTJSPP to install the JAR file into DB2 and prepare the Java classes and make them available for execution.




### 3.5.16 The DSNTJSPP flow

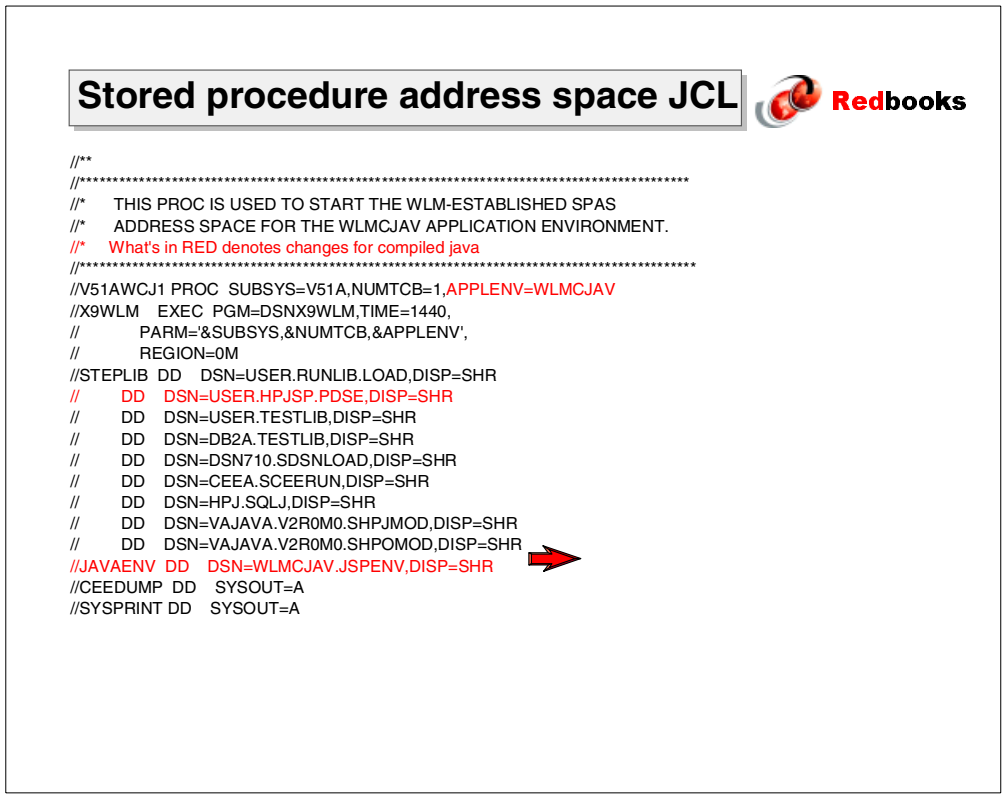
DSNTJSPP does the following to define a JAR file to DB2 and make the Java classes available for execution:

- Execute a CREATE_JAR statement. This will define the JAR to DB2 and save the jar schema, jar id, owner (current sqlid) and jar file into the catalog table SYSIBM.SYSJAROBJECTS.
- Update JAR_DATA column in SYSIBM.SYSJAROBJECTS.
- Save the SQLJ source into the table SYSIBM.SYSJARCONTENTS.
- Save all other SQLJ.INSTALL_JAR options into the table SYSIBM.SYSJAVAOPTS.
- Write the JAR file into the location specifies by jar-name.
- Precompile all the Java classes by running `db2profc` in the directory specified by jar name. The DBRMs are written to the partitioned dataset member specified in the input parameter DBRMLIB. (The environmental variable is set to the dbrmlib name.)
- Bind all the DBRMs produced by `db2profc` into packages.
- Use HPJ to compile all the classes into a PDSE program object and create a directory link in USS to the PDSE.

Note that the SPB must first issue a DROP PROCEDURE SQL statement to remove the stored procedure from DB2. The DSNTJSPP does the following to remove a JAR file and Java classes from DB2:

- Execute a DROP_JAR statement. This will remove the JAR definition from the SYSIBM.SYSJAROBJECTS and SYSIBM.SYSJARCONTENTS tables and also clean up related JAR usage authorizations.
- Delete the JAR file from the specified by jar name.
- Delete all the rows from SYSIBM.SYSJARAOPTS using jar schema.jar-id.
- DROP all packages with the name specified by input parameter program name.
- Delete the PDSE program object.

The output parameter of DSNTJSPP is an integer that indicates the success or failure of the SQLJ.INSTALL_JAR or SQLJ.REMOVE_JAR functions. A result set may also be passed back from DSNTJSPP. This result set contains a list of warning or error messages whenever DSNTJSPP produces a return code greater than or equal to 4.



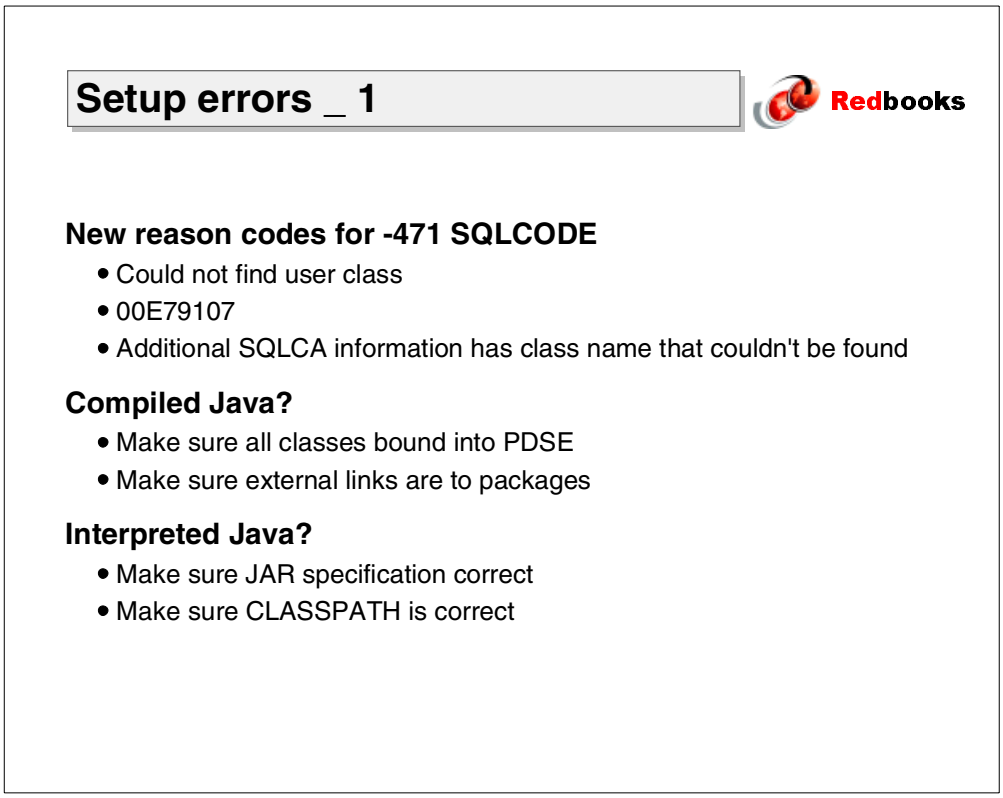

### 3.5.17 Stored procedure address space JCL

This foil highlights in red, changes you need to make to the WLM stored procedure JCL, in order to run Compiled Java stored procedures.

These DD statements are required to present the High Performance Java (HPJ) runtime libraries to the stored procedure address space. The dataset USER.HPJ.PDSE is where DB2 will find the executable Java load modules.

If you only intend to run interpreted Java stored procedures or functions, the only JCL change you need is to include the `JAVAENV DD` card. This contains the run options for the entire WLM stored procedure address space, not individual stored procedures. There must be a `CLASSPATH` entry which shows the directory for user external links, directory for compiled JDBC/SQLJ external links.

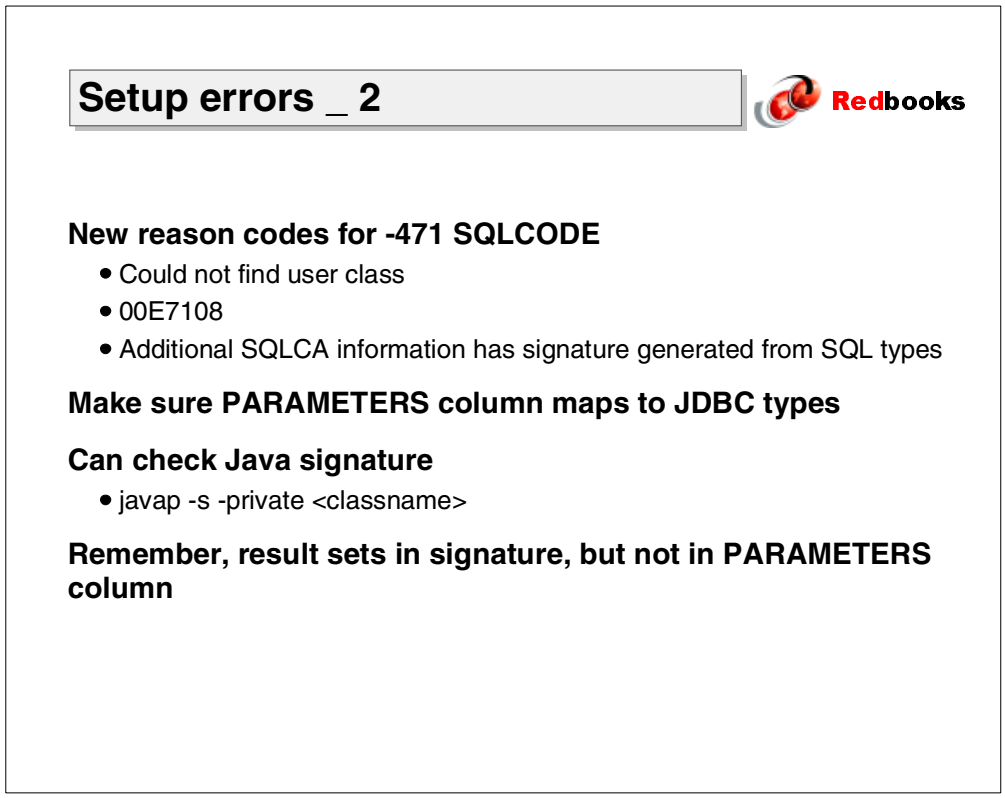
The `CLASSPATH` must contain all the Java routines, Java drivers such as JDBC drivers and directories containing all the PDSE links.




### 3.5.18 Setup errors _ 1

SQLCODE -471 is issued when the invocation of a stored procedure of function has failed. Two new reason codes can now be issued:

Reason code 00E79107 is issued when DB2 cannot find the Java class. The name of the Java class DB2 is looking for can be found in the SQLCA.



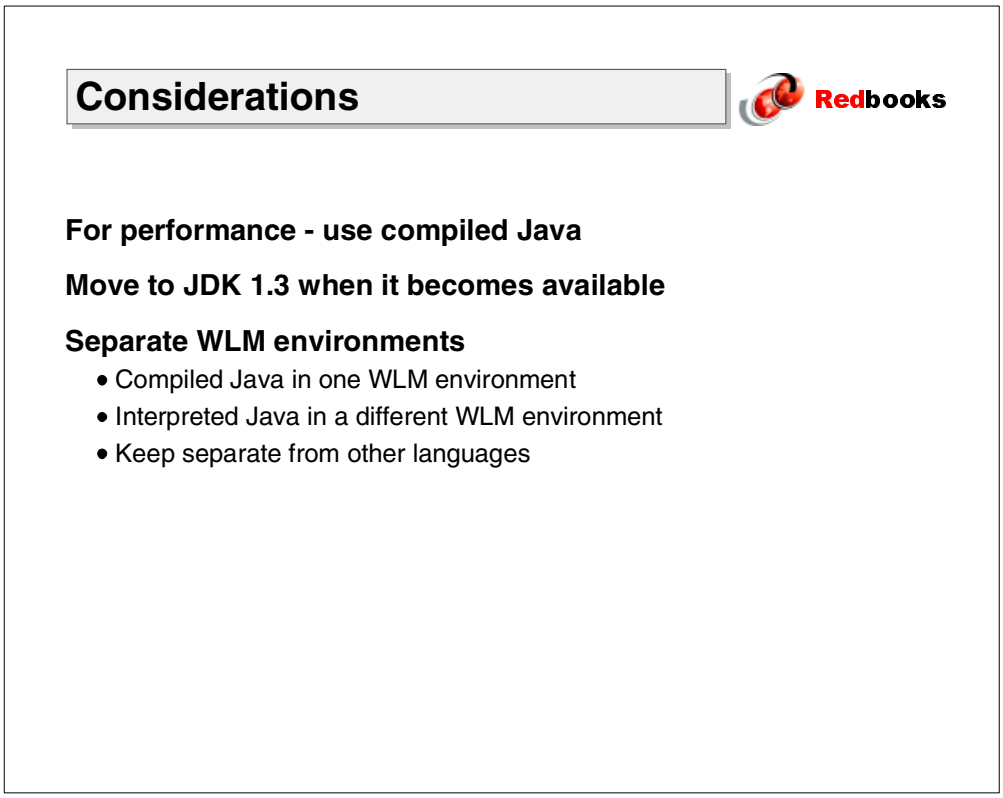

### 3.5.19 Setup errors _ 2

Reason code 00E79108 is also issued when DB2 cannot find the Java class. This reason code is issued when DB2 cannot find a Java class with a matching signature. The signature DB2 is looking for can be found in the SQLCA.

### 3.5.20 Runtime errors

ABENDS: LE/370 catches

- Java
  - Exceptions, not abends
    - Use try/catch logic
    - If uncaught, JVM terminates
    - New reason codes for -471 SQLCODE
- Other uncaught Java exceptions
  - -430 SQLCODE
  - Console message DSNX096I




### 3.5.21 Considerations

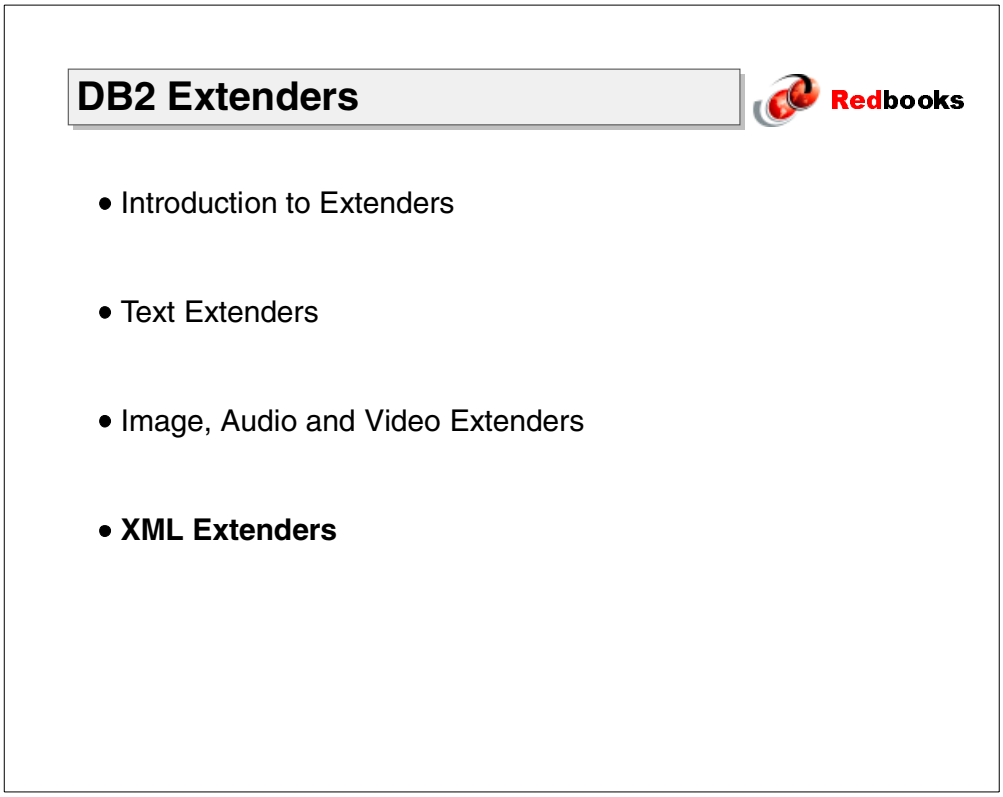
Java stored procedures implemented as compiled Java will perform better than if they are implemented as interpreted Java executing in a JVM. Compiled Java stored procedures execute in the WLM stored procedure address space while interpreted Java stored procedures execute in a JVM running in a USS environment. At the writing of this book, no performance figures are available to support this claim.

Using the JDK 1.1.8 for OS/390, the WLM stored procedure address space creates and destroys the JVM environment each time an interpreted Java routine is executed. The JDK 1.3 for OS/390 resolve this problem.

It has always been a recommendation to configure different stored procedure workloads into different WLM stored procedure environments. This includes separating stored procedures by language. The WLM environment does not have to be destroyed and a new runtime environment built when a different stored procedure needs to be executed. Java stored procedures also require large amounts of memory to execute efficiently.

This recommendation still holds for interpreted Java stored procedures and functions. For best performance, interpreted Java should also be separated from Compiled Java.

# Chapter 4. DB2 Extenders




Refer to the DB2 Extenders Web site for continuing up-to-date information on DB2 Extenders:

`http://www.ibm.com/software/data/db2/extenders`

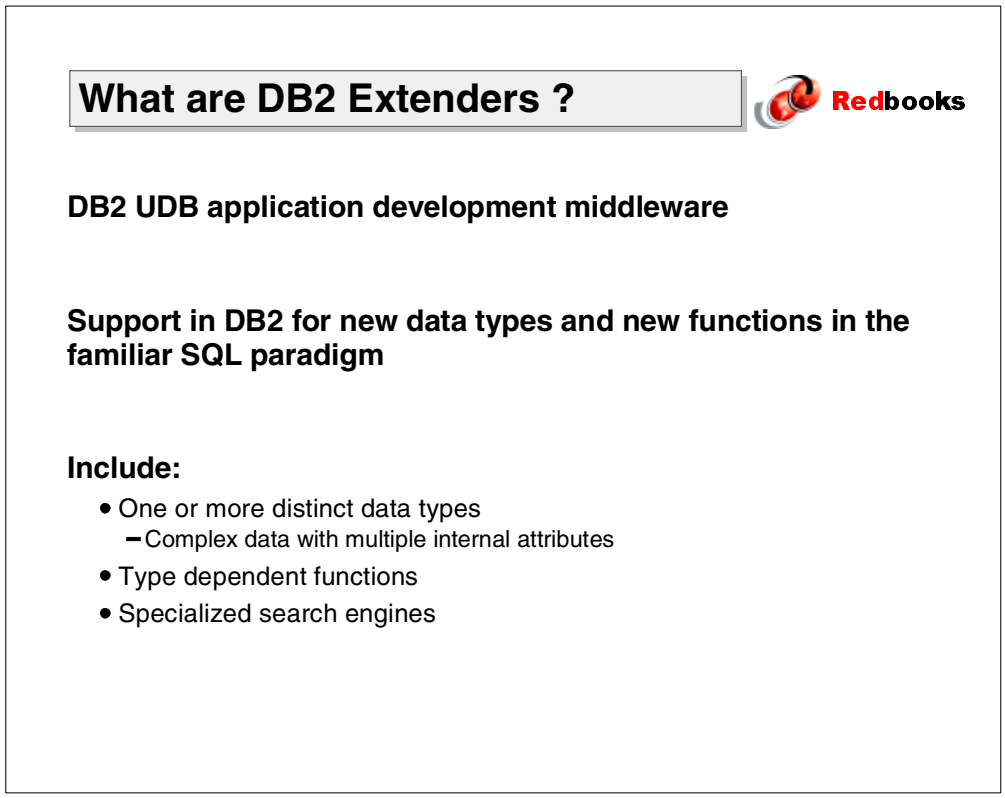
The DB2 Extenders now include XML. The following DB2 Extenders are shipped with DB2 V7:

- DB2 Text Extenders
- DB2 Image Audio Video Extenders (IAV)
- DB2 XML Extenders

Another Extender is the DB2 Spatial Extender. It allows you to generate and analyze spatial information about geographic features. It is available for the other members of the DB2 family but is not directly available for OS/390.

All DB2 Extenders make use of functions introduced with DB2 V6, namely the added built-in functions, the used defined functions and triggers, as well as LOBs.

© Copyright IBM Corp. 2001




## 4.1 Introduction to DB2 Extenders

In this section we briefly introduce the DB2 Extenders.

### 4.1.1 What are DB2 Extenders?

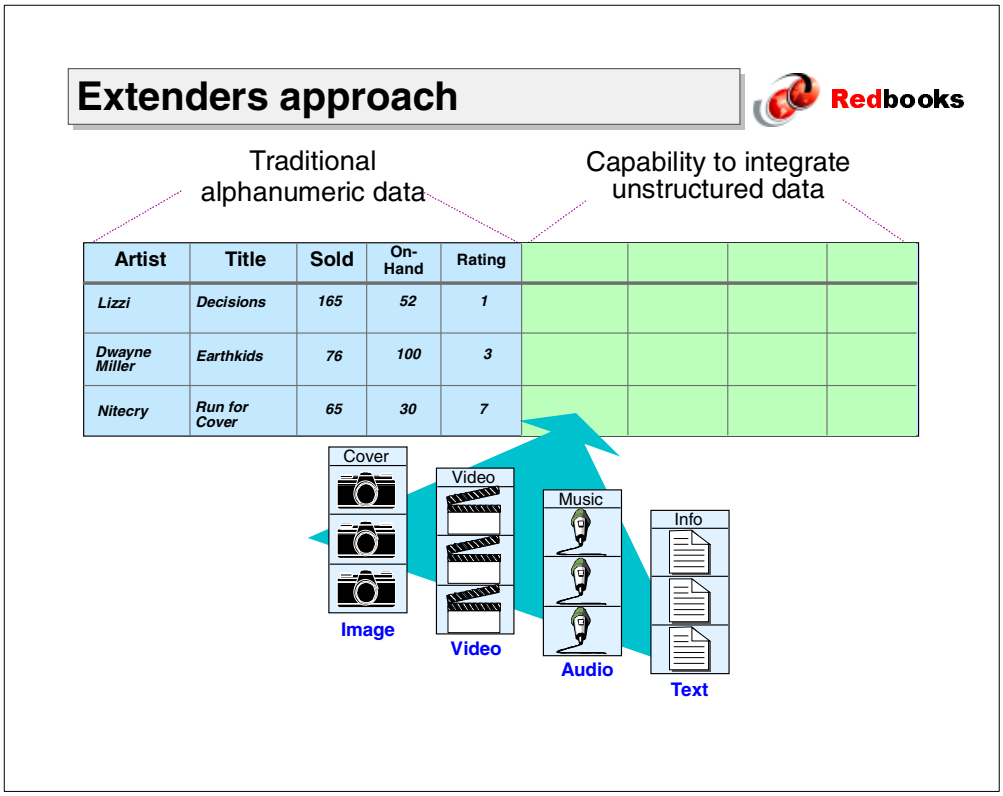
The family of extenders lets you search for a combination of text, image, video, and voice data types in one SQL query.

These extenders define new data types and functions using DB2 UDB for OS/390's built-in support for user-defined types and user-defined functions.

You can couple any combination of these data types, that is, image, audio, and video, with a text search query.

The extenders exploit DB2 UDB for OS/390's support for large objects and for triggers, introduced with DB2 V6, which provides for integrity checking across database tables ensuring the referential integrity of the multimedia data.

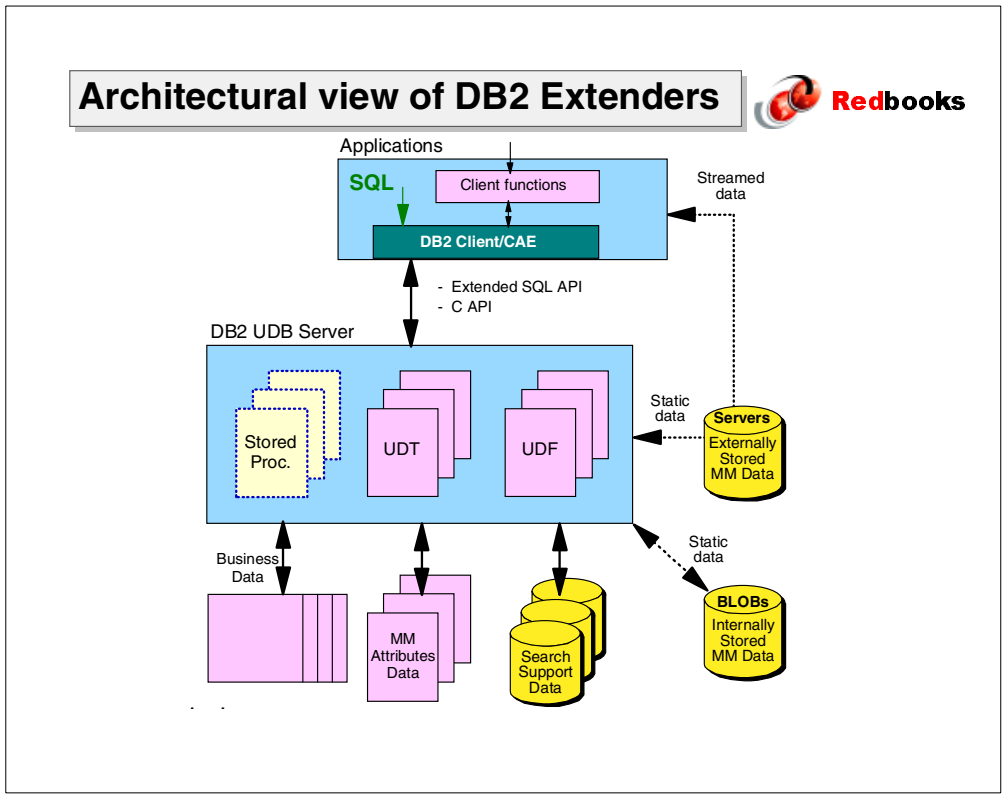
DB2 Extenders add the concepts and functions of objects to the relational engine, integrating them from the SQL language point of view, without compromising performance. They handle emerging new non-traditional data types in advanced applications, improving application development productivity and reducing development complexity.




### 4.1.2 Extenders approach

The approach used with DB2 Extenders is to integrate unstructured data with traditional alphanumeric data.

The structure for DB2 Extenders includes middleware between the RDMS engine and the applications and tools. The client has client functions such as administration and commands, while the server includes DB2 enriched with object relational facilities through the usage of UDFs and UDTs.



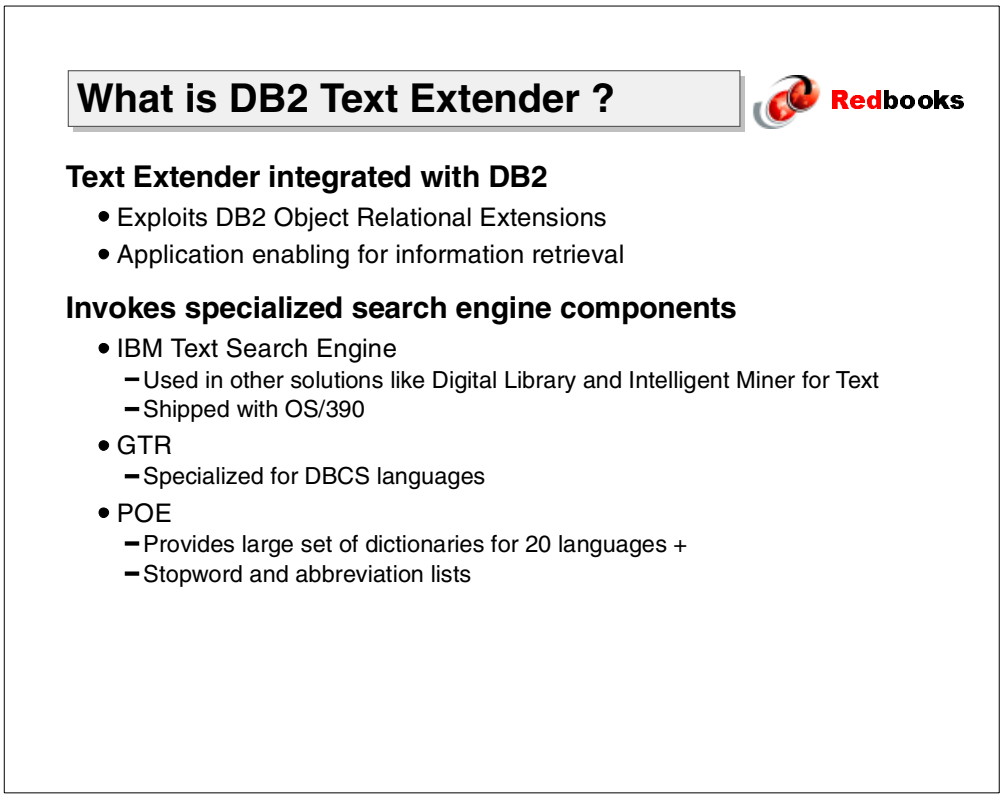

### 4.1.3 Architectural view of DB2 Extenders

The DB2 Extenders architecture allows a common method to extend the functionality and the area of application of DB2. DB2 Text Extenders are just one application example. SQL is still the programming interface. Some advantages are that it is standard, easy for SQL programmers to use, provides investment protection, and makes it extremely simple to program a query across several standard objects.

DB2 Extenders are built up by the following DB2 functions:

- LOB TYPES
  - Storing objects of any kind up to 2 GB into DB2
  - Data movement and externalization only if needed (LOB locators)
- USER-DEFINED FUNCTIONS
  - Build SQL Functions in any language
  - Controlled by DB2
  - Overloading functions managed by DB2
  - Integrated into DB2 Optimizer
- DISTINCT TYPES
  - Create new data types for type safety on DB2 base types
  - Type dollar (=integer) not compatible with type DM (=integer)
  - Automatic generated casting functions.

These functions provide the capability of building new objects and handling the semantics of the objects by the DB2 DBMS.




## 4.2 DB2 Text Extender

In this section we provide a brief introduction to DB2 Text Extender.

### 4.2.1 What is DB2 Text Extender?

Text Extender adds full-text retrieval to SQL queries by making use of features available in DB2 UDB for OS/390 that let you store unstructured text documents in databases.

Text Extender provides a fast, versatile, and intelligent method of searching through such text documents. Text Extender's strength lies in its ability to search through many thousands of large text documents at high speed, finding not only what you directly ask for, but also word variations and synonyms. Text Extender can access any kind of text document, including word-processing documents in their original native form, and offers a set of retrieval capabilities including word, phrase, wild card, and proximity searching using Boolean logic.

At the heart of Text Extender is IBM's high-performance linguistic search technology (used also in data mining). It allows your applications to access and retrieve text documents in a variety of ways. Your applications can:

- Search for documents that contain specific text, synonyms of a word or phrase, or sought-for words in proximity, such as in the same sentence or paragraph.
- Do wild card searches, using front, middle, and end masking, for word and character masking.

- Search for documents of various languages in various document formats.
- Make a “fuzzy” search for words having a similar spelling as the search term. This is useful for finding words even when they are misspelled.
- Make a free-text search in which the search argument is expressed in natural language.
- Search for words that sound like the search term.

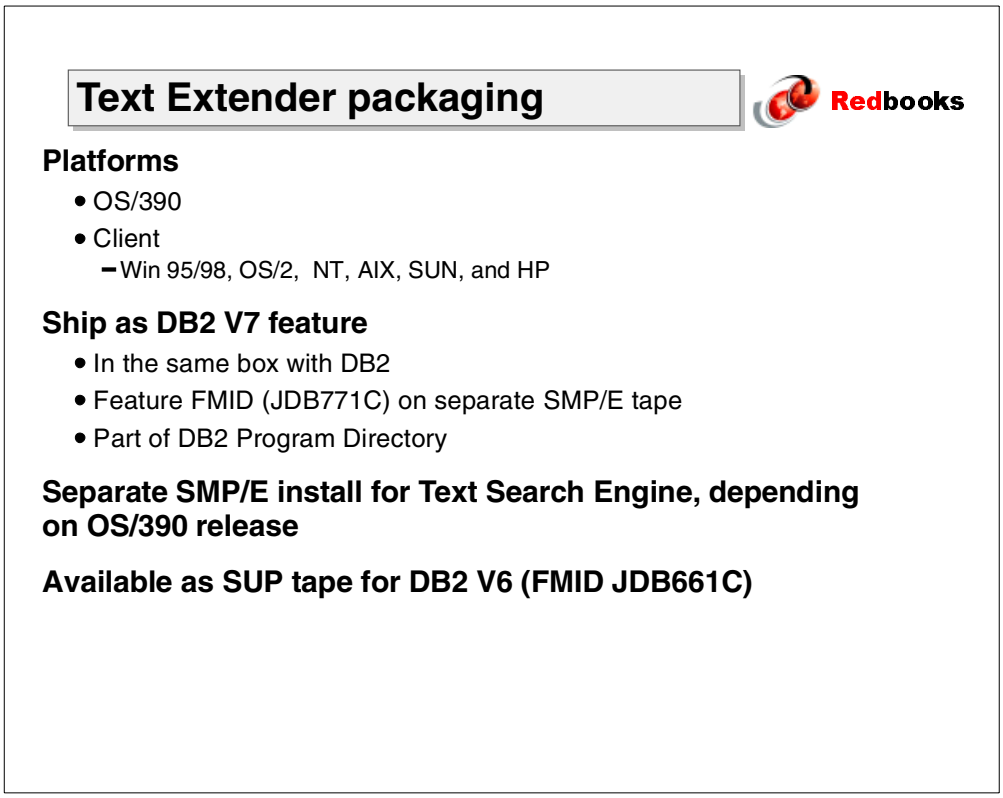
You can integrate your text search with business data queries. For example, you can code an SQL query in an application to search for text documents that are created by a specific author, within a range of dates, and that contain a particular word or phrase.

Using the Text Extender programming interface, you can also allow your application users to browse the documents.

By integrating full-text search into DB2 UDB for OS/390’s SELECT queries, you have a powerful retrieval function. The following SQL statement shows an example:

```
SELECT * FROM MyTextTable
WHERE version = ©2©
AND DB2TX.CONTAINS (
DB2BOOKS_HANDLE,
©"authorization"
IN SAME PARAGRAPH AS "table"
AND SYNONYM FORM OF "delete"©) = 1
```

In this example, `DB2TX.CONTAINS` is one of several Text Extender search functions. `DB2BOOKS_HANDLE` is the name of a handle column referring to column `DB2BOOKS` that contains the text documents to be searched. The remainder of the statement is an example of a search argument that looks for authorization, occurring in the same paragraph as `table`, and `delete`, or any of its synonyms.




### 4.2.2 Text Extender packaging

The main part of Text Extender is installed on the same machine as the DB2 server. Only one Text Extender server instance can be installed with one DB2 server instance. The installation includes:

- One or several Text Extender servers on any of the operating systems like UNIX, SUN-Solaris, and HP-UX workstations.
- AIX, SUN-Solaris, HP-UX, OS/2, Windows 98, or Windows 95, Windows NT clients with access to one or several remote Text Extender servers
- AIX clients containing a local server and having access to remote servers.

To run Text Extender from a client, you must first install a DB2 client and some Text Extender utilities. These utilities constitute the Text Extender “client” although it is not a client in the strict sense of the word. The client communicates with the server via the DB2 client connection.

Text Extender has the following main components:

- A command line interpreter: Commands are available that let you prepare text in columns for searching, and maintain text indexes.
- User-defined functions (UDFs): Functions are available that you can include in SQL queries for searching in text; and finding, for example, the number of times the search term occurs in the text. The UDFs on the client can be used as part of an SQL query. In fact, they are part of the server installation and are executed there. However, the UDFs can be used from any DB2 client without the need to install the Text Extender client.

As described in the DB2 Program Directory, DB2 UDB Text Extender for OS/390 requires:

- Workload Manager (WLM) environment as described in the DB2 Program Directory
- IBM Text Search Version as described in the DB2 Program Directory
- A group named SMADMIN must be defined to RACF. The group must have an OMVS segment and a defined group ID.The Text Extender instance owner must be a user ID assigned to the SMADMIN group, and must have DB2 SYSADM authority.

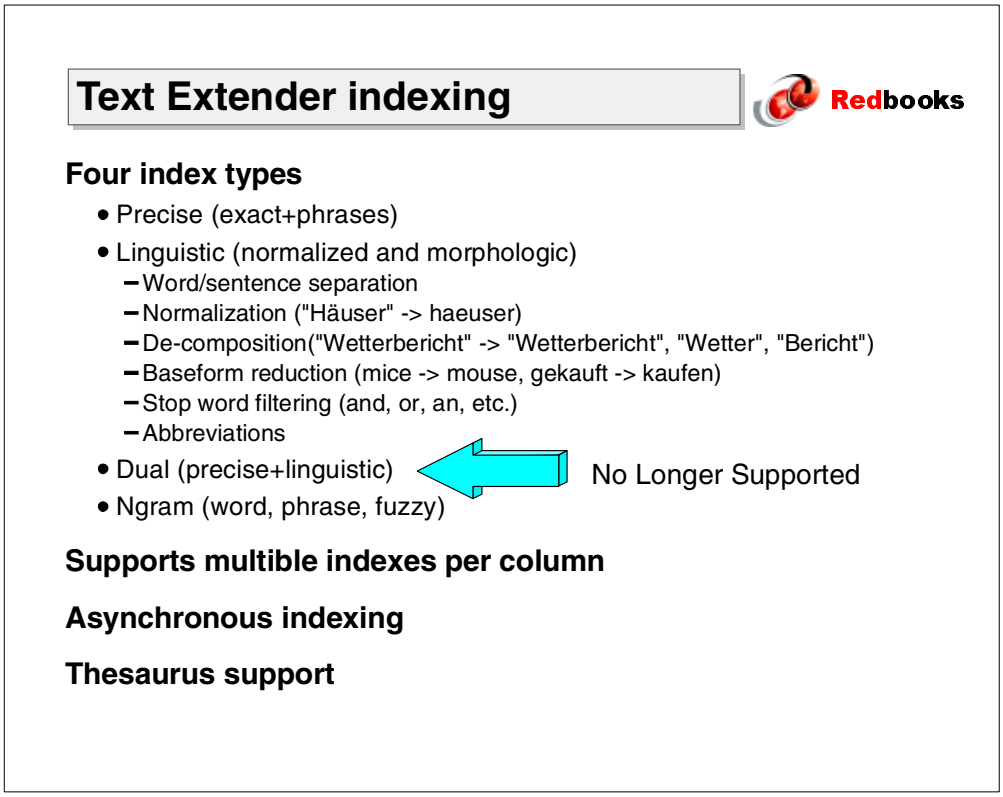
| **OS/390 Releases and Text Extender** | | | |
|---|---|---|---|
| OS/390 | DB2/TE V6 | TE V6 SUP | DB2/TE V7 |
| 2.4 | HIMN210 | HIMN210 | |
| 2.5 | HIMN210 | HIMN210 | |
| 2.6 | HIMN210 | HIMN210 | |
| 2.7 | HIMN210 | HIMN210 | HIMN210 |
| 2.8 | HIMN220 | HIMN220 | HIMN220 |
| 2.9 | HIMN230 | HIMN230 | HIMN230 |
| 2.10 | HIMN230 | HIMN230 | HIMN230 |

The table lists the OS/390 releases and the corresponding versions of DB2 Text Extender. The HIMN210 can be downloaded from the Web site:

`www.ibm.com/software/data/iminer/fortext`

The Text Search Engines (TSE) provide the following functions:

- HIMN210
  - Basic search engine functions
  - All index types
  - Thesaurus support
  - Flat structure support
  - UNIX file systems only
- HIMN220
  - GTR thesaurus
  - KeyPak (product) filter usage
  - Support of PDS and PS datasets
- HIMN230
  - Unicode (UTF8, UCS2) documents
  - XML support
  - Structured document support (such as nested)
  - Chinese support with POE




### 4.2.3 Text Extender indexing

You can assign one of these index types to a column containing text to be searched: linguistic, precise, and Ngram. You must decide which index type to create before you prepare any such columns for use by Text Extender.

***Linguistic index***

For a linguistic index, linguistic processing is applied while analyzing each document's text for indexing. This means that words are reduced to their base form before being stored in an index; the term "mice", for example, is stored in the index as `mouse`. For a query against a linguistic index, the same linguistic processing is applied to the search terms before searching in the text index. So, if you search for "mice", it is reduced to its base form `mouse` before the search begins. The advantage of this type of index is that any variation of a search term matches any other variation occurring in one of the indexed text documents. The search term `mouse` matches the document terms "mouse", "mice", "MICE" (capital letters), and so on. Similarly, the search term `Mice` matches the same document terms.

This index type requires the least amount of disk space. However, indexing and searching can take longer than for a precise index.

The types of linguistic processing available depend on the document's language. Here are the basic types:

- Word and sentence separation.
- Sentence-begin processing.
- Dehyphenation.
- Normalizing terms to a standard form in which there are no capital letters, and in which accented letters like "ü" are changed to a form without accents. For example, the German word "Tür" (door) is indexed as tuer.
- Reducing terms to their base form. For example, "bought" is indexed as buy, "mice" as mouse.
- Word decomposition, where compound words like the German "Wetterbericht" (weather report) are indexed not only as wetterbericht, but also as wetter and bericht.
- Stop-word filtering in which irrelevant terms are not indexed. "A report about all animals" is indexed as `report` and `animal`.
- Part-of-speech filtering, which is similar to stop-word filtering; only nouns, verbs, and adjectives are indexed. "I drive my car quickly" is indexed as `drive` and `car`. The words "I" and "my" are removed as stop words, but additionally the adverb "quickly" is removed by part-of-speech filtering.

***Precise index***

In a precise index, the terms in the text documents are indexed exactly as they occur in the document. For example, the search term `mouse` can find "mouse" but not "mice" and not "Mouse"; the search in a precise index is case-sensitive.

In a query, the same processing is applied to the query terms, which are then compared with the terms found in the index. This means that the terms found are exactly the same as the search term. You can use masking characters to broaden the search; for example, the search term `experiment*` can find "experimental", "experimented", and so on.

The advantage of this type of index is that the search is more precise, and indexing and retrieval is faster. Because each different form and spelling of every term is indexed, more disk space is needed than for a linguistic index.

The linguistic processes used to index text documents for a precise index are:

- Word and sentence separation
- Stop-word filtering.

***Ngram index***

An Ngram index analyzes text by parsing sets of characters. This analysis is not based on a dictionary.

If your text contains DBCS characters, you must use an Ngram index. No other index type supports DBCS characters.

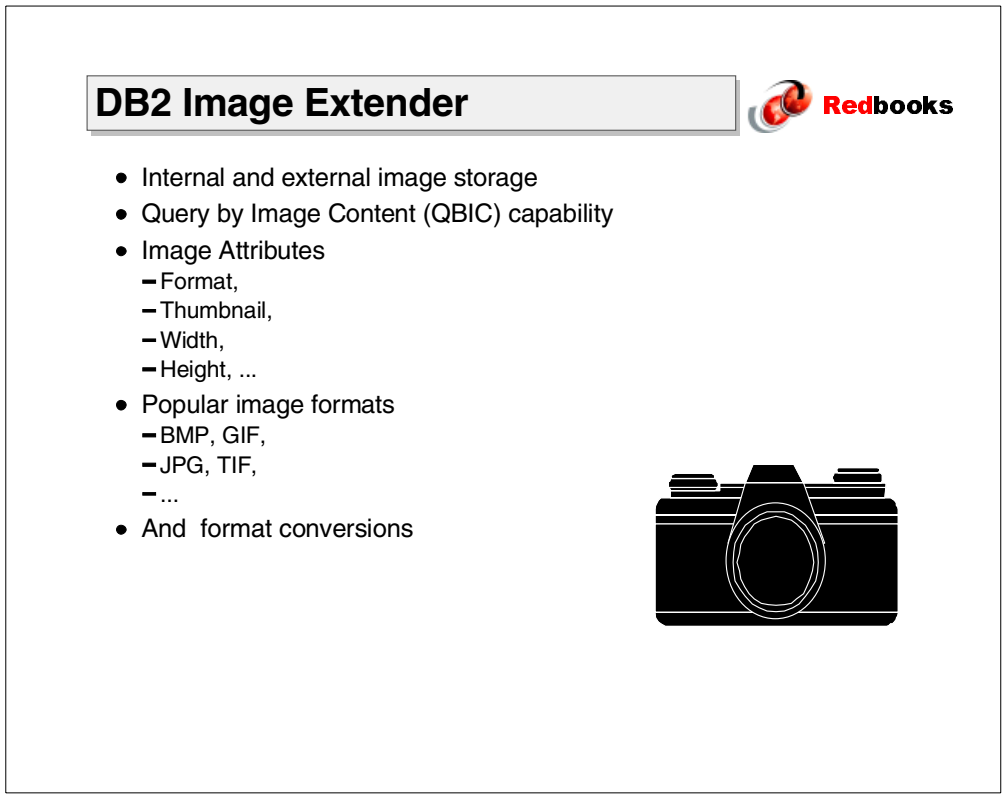
This index type supports "fuzzy" search, meaning that you can find character strings that are similar to the specified search term. For example, a search for Extender finds the mistyped word Extenders. You can also specify a required degree of similarity. Note that even if you use fuzzy search, the first three characters must match.

To make a case-sensitive search in an Ngram index, it is not enough to specify the PRECISE FORM OF keyword in the query. This is because an Ngram index normally does not distinguish between the case of the characters indexed. You can make an Ngram index case-sensitive, however, by specifying the CASE_ENABLED option when the index is created. Then, in your query, specify the PRECISE FORM OF keyword.

When the CASE_ENABLED option is used, the index needs more space, and searches can take longer.

The SBCS CCSIDs supported by Ngram indexes are 819, 850, and 1252. The DBCS CCSIDs supported by Ngram indexes are: 932, 942, 943, 948, 949, 950, 954, 964, 970, 1363, 1381, 1383, 1386, 4946, and 5039.

Although the Ngram index type was designed to be used for indexing DBCS documents, it can also be used for SBCS documents. However, it supports only TDS documents.



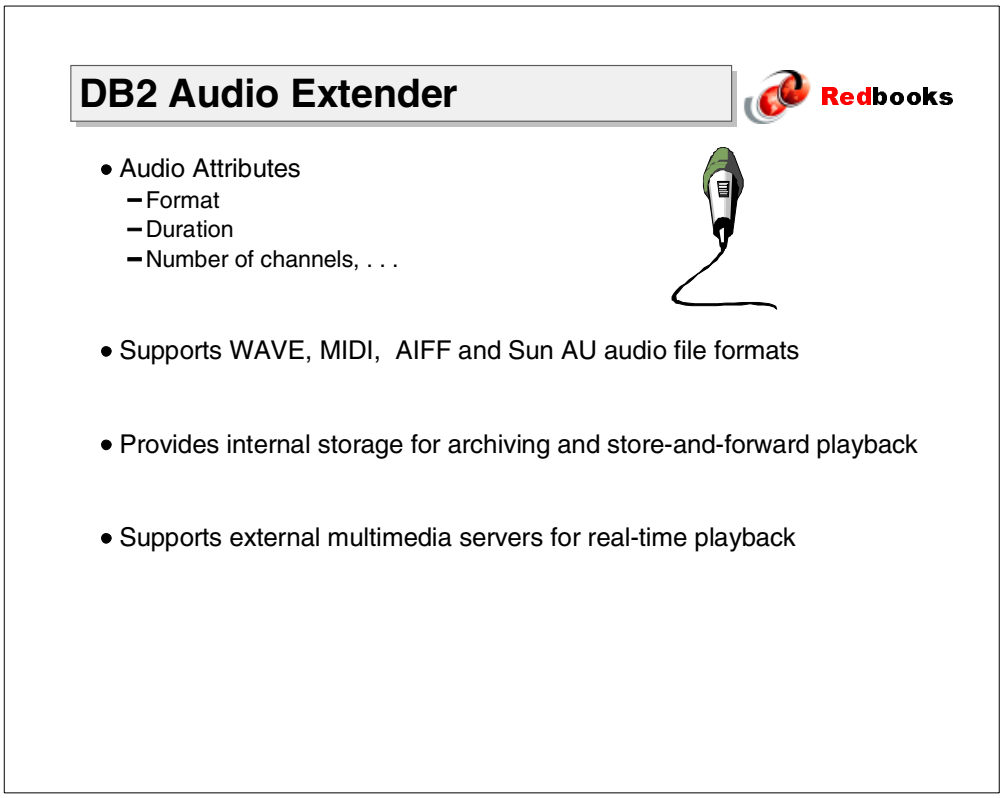

### 4.2.4 DB2 Image Extender

With the Image Extender, your applications can:

- Import and export images and their attributes into and out of a database
- Control access to images with the same level of protection as traditional business data
- Select and update images based on their format, width, and height
- Display miniature images and full images

You can integrate an image query with traditional business database queries. For example, you can program an SQL statement in an application to return miniature images of all pictures whose width and height are smaller than 512 x 512 pixels and whose price is less than $500, and also list the names of each picture's photographer.

Using the Image Extender, you can also allow your application users to browse the images.



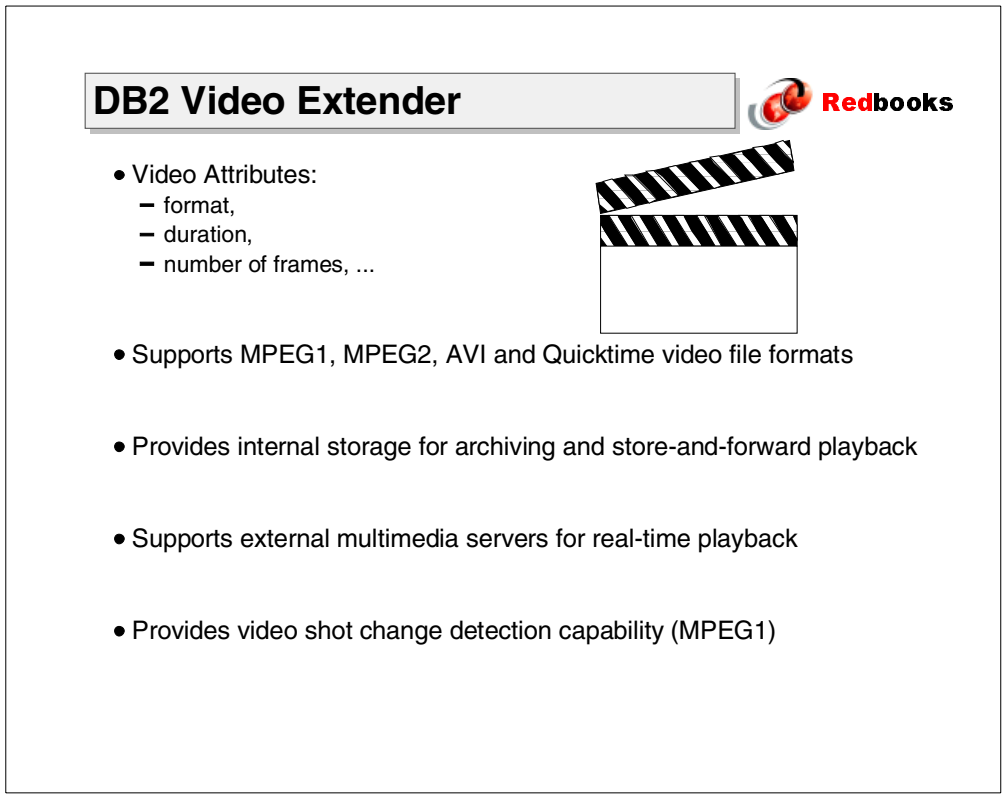

### 4.2.5 DB2 Audio Extender

With the Audio Extender, your applications can:

- Import and export audio clips and their attributes to and from a DB2 UDB for OS/390 database
- Select and update audio clips based on audio attributes, such as number of channels, length, and sampling rate
- Play audio clips

The Audio Extender supports a variety of audio file formats, such as WAVE and MIDI. Like the Video Extender, the Audio Extender works with different file-based audio servers.

Using the Audio Extender, your applications can integrate audio data and traditional business data in a query. For example, you can code an SQL statement in an application to retrieve miniature images of compact disk (CD) album covers, and the name of singers of all music segments on the CD whose length is less than 1 minute and that were produced in 1996. Using the Audio Extender, you can also allow your application users to play the music segments.




### 4.2.6 DB2 Video Extender

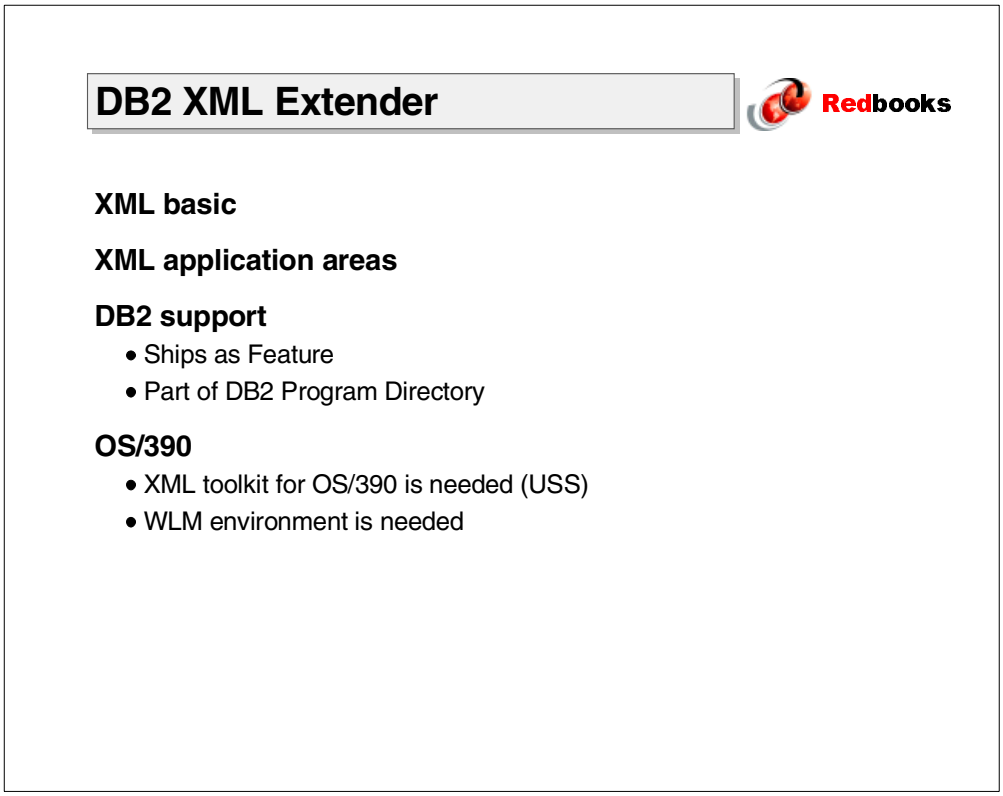
With the Video Extender, your applications can:

- Import and export video clips and their attributes to and from a DB2 UDB for OS/390 database
- Select and update video clips based on video attributes such as compression method, length, frame rate, and number of frames
- Retrieve specific shots in a video clip through shot detection
- Play video clips

You can integrate a video query with traditional business database queries. For example, you can code an SQL statement in an application to return miniature images and names of the advertising agencies of all commercials whose length is less than 30 seconds, whose frame rate is greater than 15 frames a second, and that contain remarks such as “Time Warp” in the commercial script. Using the Video Extender, you can also allow your application users to play the commercials.

There is no functionality difference between the IAV extenders shipped with DB2 V7, and the extenders that are available with DB2 V6.

They are a DB2 V7 Feature and are shipped with DB2. Feature FMID(JDB771B) is on separate SMP/E tape, and it is described as part of DB2 Program Directory. Prerequisites include a minimum level of OS/390 V2R4 and UNIX Services. The Extender clients are common workstations. IAV support is provided as for DB2.




## 4.3 DB2 XML Extender

The IBM DB2 Extenders family provides data and metadata management solutions to handle traditional and nontraditional data. The XML Extender helps you integrate the power of DB2 with the flexibility of XML.

DB2's XML Extender provides the ability to store and access XML documents, or generate XML documents from existing relational data and shred (decompose, storing untagged element or attribute content) XML documents into relational data. XML Extender provides new data types, functions, and stored procedures to manage your XML data in DB2.

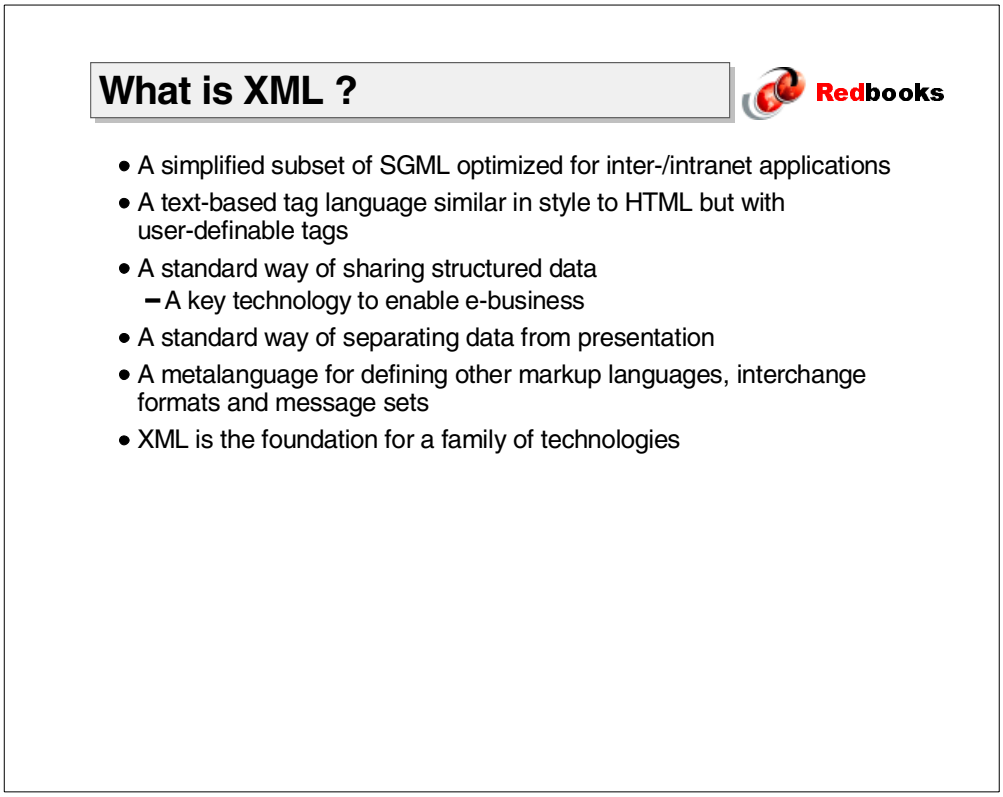
The XML Extender is also available on the following operating systems:

- Windows NT
- AIX
- Sun Solaris
- Linux

A new DB2 manual: *DB2 UDB for OS/390 and z/OS Version 7 XML Extender Administration and Programming*, SC26-9949, describes these new functions.

The XML Extender adds the power to search rich data types of XML element or attribute values, in addition to the structural text search that the DB2 Text Extender for OS/390 provides. An application server can send the XML documents over the Internet to other sites. XML is the standard for data interchange for the next generation of electronic business-to-business and business-integration solutions.

You can use interchange formats that are based on XML to leverage your critical business information in DB2 databases in business-to-business solutions. When you store, retrieve, and search XML documents in a DB2 database, you benefit from the unmatched performance, reliability and scalability of DB2 for OS/390 and z/OS. With the XML Extender, you can integrate Internet applications that are based on XML documents with your existing DB2 database.




### 4.3.1 What is XML?

In 1978, IBM developed the Standard Generalized Markup Language, or SGML, for developing documents. Initially, this was used in Web development, until it was determined to be too complex. In 1990, the HyperText Markup Language (HTML) was introduced, to simplify the process of building Web pages. While SGML was too complex, HTML was too simple to handle much of what people wanted to place on the Web. So XML was developed with the intention not to replace HTML, but rather to complement it.

The eXtensible Markup Language (XML) is a meta language. It is extensible in that the language itself is a metalanguage that allows you to create your own language depending on the needs of your enterprise. You use XML to capture not only the data for your particular application, but also the data structure. XML is not the only interchange format. However, XML has emerged as the accepted standard for data interchange. By adhering to this standard, applications can finally share data without needing to transform data using proprietary formats.

***XML-driven application areas***

XML provides significant benefits in three solution categories that are experiencing significant, if not explosive growth:

- Web publishing and content management, including Internet or intranet information portals for publishing complex documents and documents from different sources as well as applications that provide content syndication or subscription via the Web.
- E-commerce, both business-to-consumer and business-to-business.

- Application integration, particularly customer relationship management, Web-to-back office, and data warehousing applications.

**Web publishing**. XML initially sparked interest among groups publishing and managing Web content. It was generally viewed it as the successor to HTML. Already convinced of the value of structured information, the SGML community in particular had been looking for a way to leverage such information on the Web. Most initial XML products, including products from Inso Corp., Vignette Corp., ArborText Inc., Textuality, and Interleaf Inc., were designed for Web publishing and content management.

There are a number of advantages to using XML for Web publishing and content management applications. Once you structure data with XML tags, for example, you can easily combine data from different sources. And, once XML documents are delivered to the desktop, they can be viewed in different ways as determined by client configuration, user preference, or other criteria. For example, you could look at a product manual in "expert" mode, where only reference information is displayed, or in "novice" mode, where tutorial information is also displayed. XML tags also enable more meaningful searches, because searches can be restricted to specific parts of the document based on the content contained within different tags.

Some companies are using XML as part of their Internet or intranet information portals. For example, Dell uses a background XML application designed for content management and personalization on 17 different sites in Europe, the Middle East, and Africa. Before moving to XML content, Dell duplicated HTML pages for each country-specific site.

Other companies are providing content syndication and subscription via the Web. For example, Dow Jones Interactive Publishing collects data feeds in various formats from publishers of 6,000 periodicals and converts the data to XML before sending it to the intranets of about 100 business customers.

**E-commerce**. The real excitement over XML is not in Web publishing but in XML's potential as an enabler of the data interchange necessary for business-to-business e-commerce. Forrester Research has projected that business-to-business (B2B) e-commerce in the United States, will grow from $43 billion in 1998 to $1.3 trillion by 2003, for an annual growth rate of 99 percent. With this kind of money at stake, any technology that has the promise of making this kind of solution easier to implement, as XML does, is bound to have rapid adoption.

For example, businesses want to automate procurement of non-productions supplies (for example, office supplies) to lower costs and to take advantage of emerging Internet-based auction-style spot markets. Another example is a company, currently not using EDI, wanting to automate its production supply chain. These cross-organizational business processes are accomplished by passing electronic documents between organizations. These documents include purchase order, invoices, inventory queries, shipment tracking requests, etc.

Electronic Data Interchange (EDI) has been around for a number of years and can handle many of these processes. EDI has defined various types of documents (like purchase orders).

So why do we need XML? Data interchange formats with XML are very flexible. Without XML, two communicating applications must predetermine the format of the messages sent between them, the data elements that will be passed, and the order in which the data elements are arranged. However, when XML is the message format, the two applications can dynamically interpret the message format using an XML parser. And XML message formats are extensible: Using the same application that created the XML document, you could add an additional data element to support another application. The original application that used the document would be unaffected.

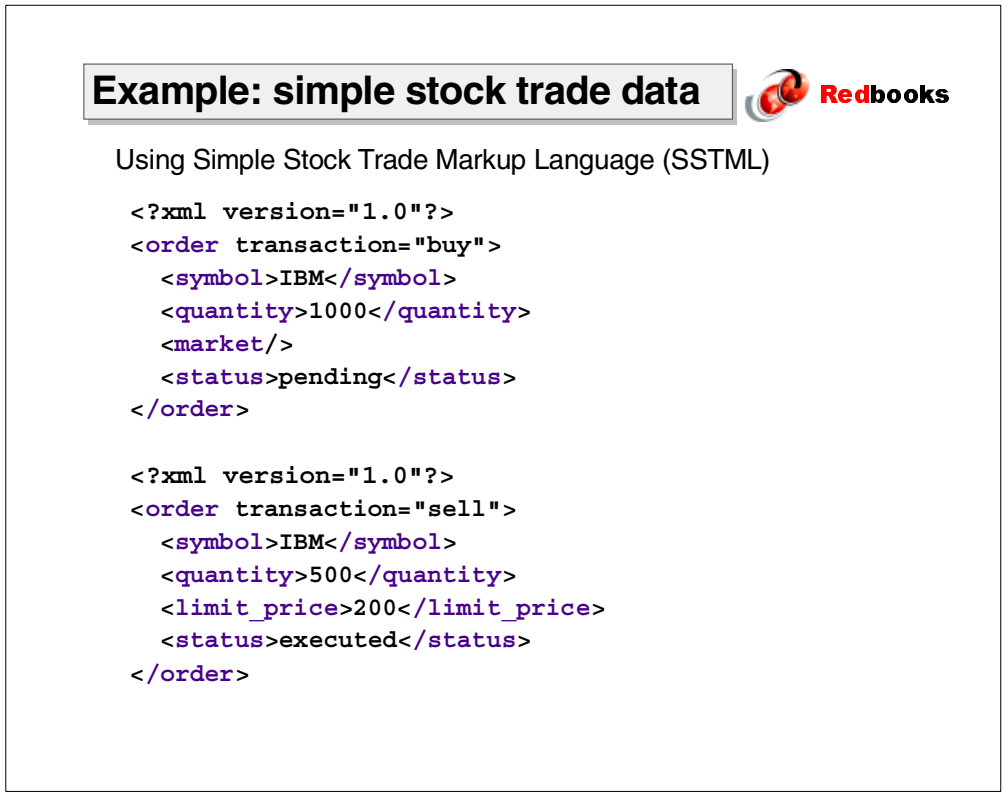
For example, XML is used to define a document that contains the results on an inventory query. This document could include an element called "Part", which includes "Part-number", "SKU", and "Quantity". The application that produces the XML document could add a new element, called "Price" to support a new application. The original applications would be unaffected, since they use an XML parser to look for "Part-number", "SKU", and "Quantity".

Dozens of industry-specific XML markup languages have been defined, including:

- Open Financial Exchange (OFX), a format for exchanging personal financial information between financial institutions and products such as Quicken and Microsoft Money, defined by Microsoft, Quicken, and Checkfree.
- RosettaNet Partner Interface Process' (PIPs), specifications for electronic commerce processes for the IT supply chain, defined by RosettaNet, a nonprofit consortium.
- Information and Content Exchange (ICE), a format for the automatic, controlled exchange and management of online content between business partners, including information such as access rights to data, expiration dates, and update frequency. This is defined by a group of companies lead by Vignette.

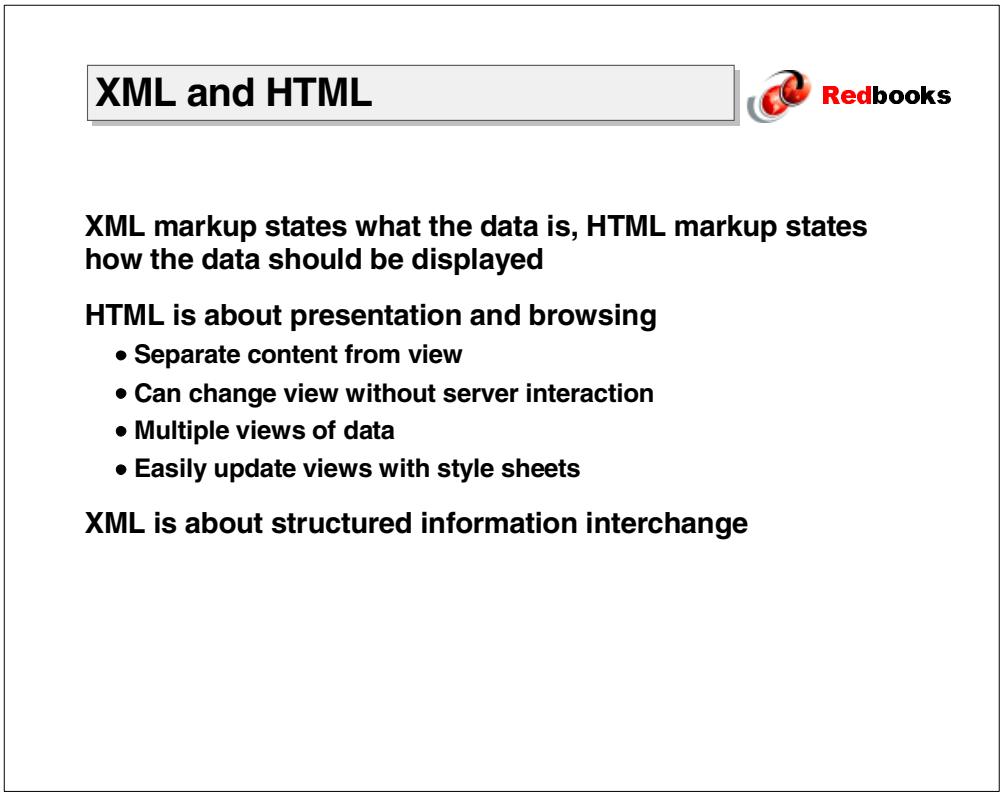
XML lowers the technical barriers to data interchange over the Internet because it is easier to understand and implement than standards such as ASN.1 and EDI. The base specification is only 30 pages long and is easily understood by those already familiar with HTML. And because XML is a text format rather than a binary one, anyone can read it. Designed with the Internet in mind, XML documents are compatible with Internet infrastructure elements, including HTTP protocols and fire walls. In contrast, EDI formatted documents are not compatible with Internet standards like HTTP and require custom value added networks (VANs).

**Application integration.** XML plays a significant role in the efforts many companies are undertaking to integrate e-commerce and CRM applications with their enterprise systems. The data to support real-time e-commerce is contained in legacy, back-office systems. Likewise, a comprehensive CRM solution requires access to data in a variety of disparate systems to achieve a complete picture of customer relationships. XML's usefulness as a data interchange format discussed in the previous section on e-commerce, also apply in this area. The primary difference is that instead of being defined by industry groups, the specific XML grammars are defined by individual companies for their internal use.




### 4.3.2 Example: simple stock trade data

SSTML is a specialized language for stock trading. That is, the language set (alphabet/allowed words, rules, symbols, quantities, and so on) is defined by means of a DTD.




### 4.3.3 XML and HTML

There are many applications in the computer industry, each with is own strengths and weaknesses. Users today have the opportunity to choose whichever application best suits the need for their particular tasks. However, because users tend to share data between their separate applications, they are continually faced with the problem of replicating, transforming, exporting, or saving their data as a different format that can be imported into another application. This can be a critical problem in business applications, because many of these transforming processes tend to drop some of the data, or they require at least that users go through the tedious process of ensuring that data is consistent.

Today, one of the ways to address this problem is for application developers to write ODBC applications to save the data into a database management system. From there, the data can be manipulated and presented in the form in which it is needed for another application. Database applications need to be written to convert the data into a form that an application requires, however applications change quickly and become out of date. Applications that convert data to HTML provide presentation solutions, but the data presented cannot be practically used for other purposes because HTML is not extensible, has no semantics, presents one only view of data.

XML has emerged to address this problem. It is extensible in that the language itself is a metalanguage that allows you to create your own language depending on the needs of your enterprise. You use XML to capture not only the data for your particular application, but also the data structure. XML is not the only interchange format available but it has emerged as the accepted standard for data interchange. By adhering to this standard, applications can finally share data without needing to transform data using proprietary formats. Many new applications will be able to take advantage of it.

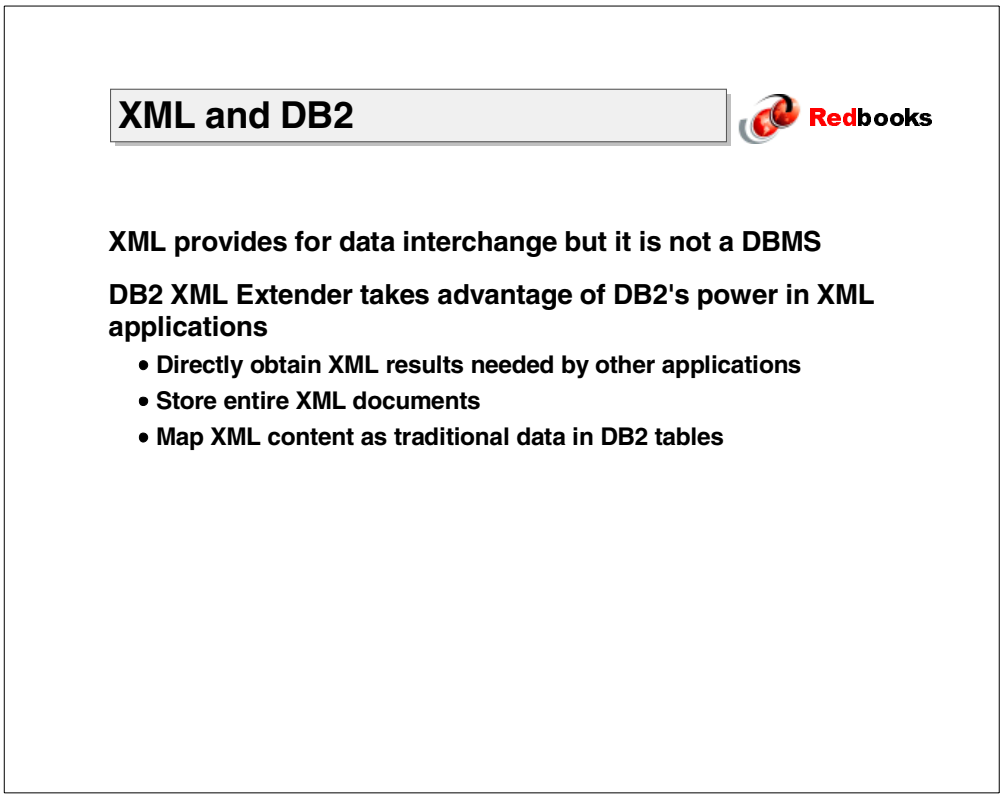
Suppose you are using a particular project management application and you want to share some of its data with your calendar application. With XML, this can be done with ease. In today's interconnected world, an application vendor will not be able to compete unless it provides XML interchange utilities built into its applications. So, in this example, your project management application could export tasks in XML, which could then be imported as is into your calendar application if the information conforms to an accepted Document Type Definition (DTD).

In the middle term, XML will not replace HTML. But as more and more XML documents are used, they will become the basis for a number of HTML documents, which are generated dynamically. This transformation will normally occur at a server, so that the HTML browser can still be used at the client side.

In the long run, this transformation will be by the browser at the client side (as rudimentary already possible by Microsoft Internet Explorer Version 5). Then XML will perhaps become more generally available at the client. But up to now, almost no tools exist which are as powerful as the HTML tools (such as Net.Objects Fusion and Cold Fusion).

XML becomes a concrete language for an application area (for example; exchange of data in trade) by another definition, the DTD (document type definition). Here restrictions apply concerning the vocabulary, the syntax, and the allowed values. You can compare the DTD with the declaration section in a program language, which limits/restricts the usage of variables in the program.

An XML document is called “well formed”, if it adheres to the rules of the XML syntax. It is called “valid”, respective to a DTD, if it additionally adheres to the rules of the DTD.




### 4.3.4 XML and DB2

Even though XML solves many problems by providing a standard format for data interchange, there are still other problems to overcome. When building an enterprise data application, you need to answer questions such as:

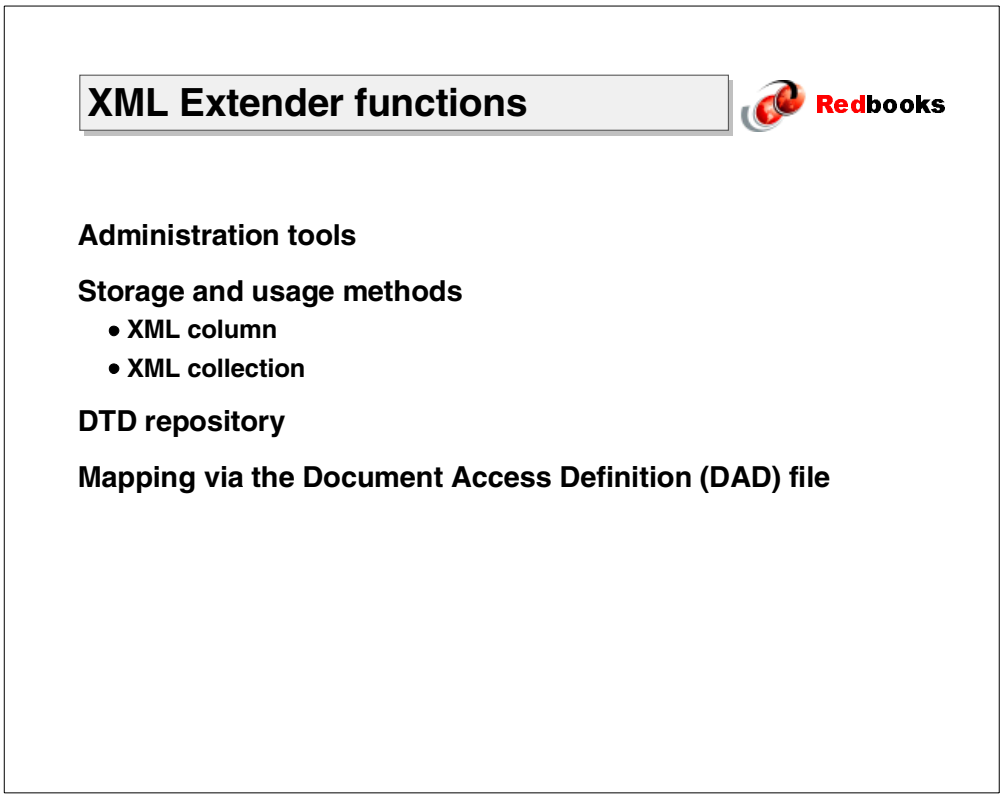
- How often do I want to replicate the data?
- What kind of information needs to be shared between applications?
- How can I quickly search for the information I need?
- How can I have a particular action, such as a new entry being added, trigger an automatic data interchange between all my applications?

These kinds of issues can be addressed only by a database management system. By incorporating the XML information and meta-information directly into the database, you can more directly and more quickly obtain the XML results that your other applications need for their particular purpose. This is where the DB2 XML Extender can assist you.

With the content of your structured XML documents in a DB2 database, you can combine structured XML information with your traditional relational data. Based on the application, you can choose whether to store entire XML documents in DB2 as a nontraditional user-defined data type, or you can map the XML content as traditional data in relational tables. For nontraditional XML data types, the XML Extender adds the power to search rich data types of XML element or attribute values, in addition to the structural text search that the DB2 Text Extender provides.

With the XML Extender, your application can:

- Store entire XML documents as column data in an application table or externally as a local file, while extracting desired XML element or attribute values into side tables for search. Using the XML column method, you can:
- Perform fast search on XML elements or attributes of SQL general data types that have been extracted into side tables and indexed
- Update the content of an XML element or the value of an XML attribute
- Extract XML elements or attributes dynamically using SQL queries
- Validate XML documents during insertion and update
- Perform structural-text search with the Text Extender
- Compose or decompose contents of XML documents with one or more relational tables, using the XML collection storage and access method




### 4.3.5 XML Extender functions

XML Extender provides the following features to help you manage and exploit XML data with DB2:

- Administration tools to help you manage the integration of XML data in relational tables
- Storage and usage methods for your XML data
- A DTD repository for you to store DTDs used to validate XML data
- A mapping scheme called the Document Access Definition (DAD) file for you to map XML documents to relational data

***Administration tools***

The XML Extender administration tools help you to enable your database and table columns for XML, and map XML data to DB2 relational structures. The XML Extender provides server administration tools for your use, depending on whether you want to develop an application to perform your administration tasks or whether you simply want to use a wizard. You can use the following tools to complete administration tasks for the XML extender:

- The XML Extender administration wizards provide a graphical user interface for administration tasks (DB2 Connect is a prerequisite).
- The DXXADM command from TSO or the `odb2` command line tool (using UNIX Systems Services command shell) provides an option for administration tasks.
- The XML Extender administration stored procedures provide application development options for administration tasks.

***Storage and access methods***
XML Extender provides two storage and access methods for integrating XML documents into DB2: XML column and XML collection. These methods have very different uses, but can be used in the same application.

- XML column:

  This method helps you store intact XML documents in DB2. XML column works well for archiving documents. The documents are inserted into columns that are enabled for XML and can be updated, retrieved, and searched. Element and attribute data can be mapped to DB2 tables (side tables), which in turn can be indexed for fast structural search.

- XML collection:

  This method helps you map XML document structures to DB2 tables so that you can either compose XML documents from existing DB2 data, or decompose (store untagged element or attribute content) XML documents into DB2 data. This method is good for data interchange applications, particularly when the contents of XML documents are frequently updated.

***DTD repository***
The XML Extender provides and XML Document Type Definition (DTD) repository, which is a set of declarations for XML elements and attributes. When a database is enabled for XML, a DTD reference table (DTD_REF) is created. Each row of this table represents a DTD with additional metadata information. Users can access this table to insert their own DTDs. The DTDs in the DTD_REF table are used to validate XML documents.

***Mapping with DAD***
You specify how structured XML documents are to be handled in a Document Access Definition (DAD). The DAD itself is an XML formatted document. It associates XML document structure to a DB2 database when using either XML columns or XML collections. The structure of the DAD is different when defining an XML column as opposed to an XML collection. DAD files are managed using the XML_USAGE table, created when you enable a database for XML.

***More on column and collection data management***
Here are some considerations on column and collection data management:

- **XML column**: Structured document storage and retrieval.

  Because XML contains all the necessary information to create a set of documents, there will be times when you want to store and maintain the document structure as it currently is.

  For example, if you are a news publishing company that has been serving articles over the Web you might want to maintain an archive of published articles. In such a scenario, the XML Extender lets you store complete or partial XML articles in a column of a DB2 table. This type of XML document storage is called an XML column.

  The XML Extender provides the following user-defined types (Adds) for use with XML columns: XMLVarchar, XMLCLOB, XMLFILE. These data types are used to identify the storage type of XML documents in the application table. The XML Extender supports legacy flat files; you are not required to store XML documents inside DB2.

The XML Extender provides powerful user-defined functions (UDFs) to store and retrieve XML documents in XML columns, as well as to extract XML element or attribute values. A UDF is a function that is defined to the database management system and can be referenced thereafter in SQL queries. The XML Extender provides the following types of UDFs:

- Storage: Stores intact XML documents in XML-enabled columns as XML data types
- Extract: Extracts XML documents, or the values specified in elements and attributes as base data types
- Update: Updates entire XML documents or specified element and attribute values

The extract functions allow you to perform powerful searches on general SQL data types. Additionally, you can use the DB2 UDB Text Extender with the XML Extender to perform structural and full text searches on text in XML documents. This powerful search capability can be used, for example, to improve the usability of a Web site that publishes large amounts of readable text, such as newspaper articles or Electronic Data Interchange (EDI) applications, which have frequently searchable elements or attributes.

- **XML collection**: Integrated data management.

  Traditional SQL data is either decomposed form incoming XML documents or used to compose outgoing XML documents. If your data is to be shared with other applications, you might want to be able to compose and decompose incoming and outgoing XML documents and manage the data as necessary to take advantage of the relational capabilities of DB2. This type of XML document storage is called XML collection.

  The XML collection is defined in a DAD file, which specifies how elements and attributes are mapped to one or more relational tables. You can define a collection name by enabling it, and then use it with stored procedures to compose or decompose XML documents.

  When you define a collection in the DAD file, you use one of two types of mapping schemes: SQL mapping or RDB_node mapping. SQL mapping uses SQL SELECT statements to define the DB2 tables and conditions of use for the collection. RDB_node mapping uses XPath-based RDB_node to define the tables, columns, and conditions.

  Stored procedures are provided to compose or decompose XML documents.

# Part 3. Utilities

© Copyright IBM Corp. 2001

# Chapter 5. Utilities

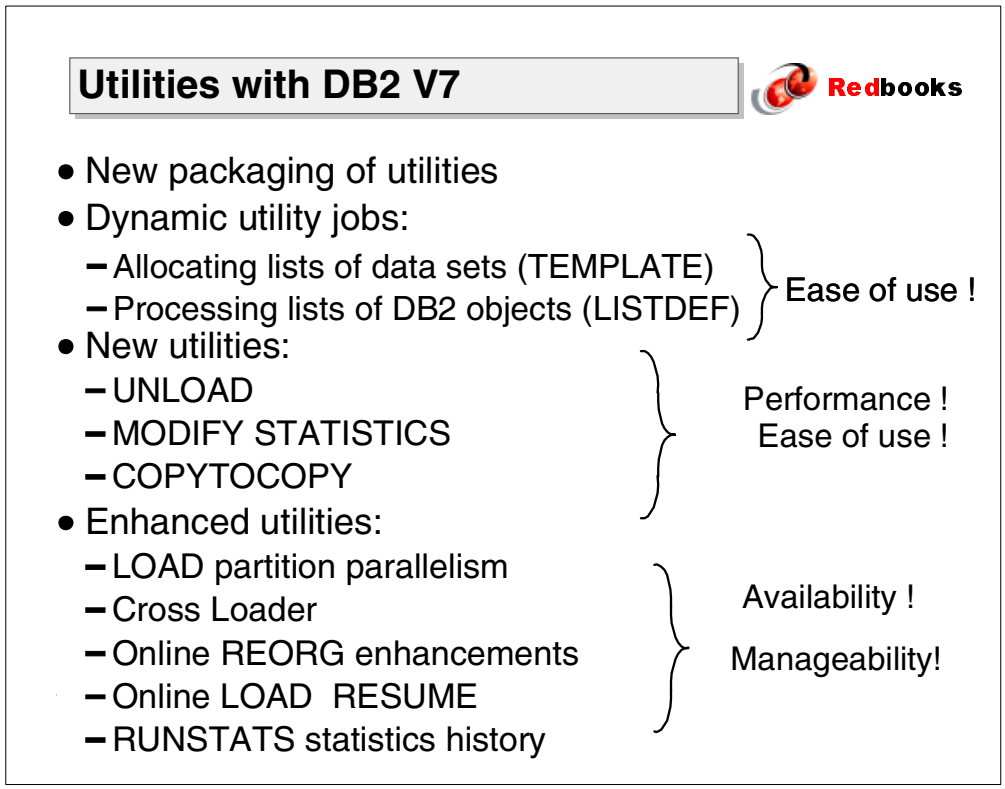
| Utility | DB2 V4 | DB2 V5 | DB2 V6 |
|---|---|---|---|
| Copy | Design change to improve performance up to 10 times.<br>CONCURRENT option for DFSMS. | Full copies inline with Load and Reorg.<br>CHANGELIM option to decide execution. | Copies of indexes.<br>Parallelism.<br>Check page.<br>PARALLEL recover. |
| Reorg | NPI improvements.<br>Catalog Reorg.<br>Path length reduction.<br>SORTDATA | Reorg and inline Copy (COPYDDN).<br>New SHRLEVEL NONE, REFERENCE, CHANGE option (Online Reorg).<br>Optional removal of work data sets for indexes (SORTKEYS).<br>NOSYSREC<br>PREFORMAT option. | Collect inline statistics.<br>Build indexes in parallel.<br>Discard and faster Unload.<br>Faster Online Reorg.<br>Threshold for execution.<br>SORTKEYS also used for parallel index build. |
| Runstats | Modified hashing technique for CPU reduction. | New SAMPLE option to specify % of rows to use for non-indexed column statistics.<br>New KEYCARD option for correlated key columns.<br>New FREQVAL option for frequent value statistics with non-uniform distribution. | Runstats executed inline with Reorg, Load, and Recover or Rebuild index.<br>Parallel table space and index. |
| Load | PI improvements.<br>Path length reduction. | Load and inline Copy (COPYDDN).<br>Optional removal of work data sets for indexes (SORTKEYS).<br>Reload phase performance.<br>PREFORMAT option. | Collect inline statistics.<br>Build indexes in parallel.<br>SORTKEYS also used for parallel index build. |
| Recovery | Usage of DFSMS CONCURRENT copies.<br>Recover index restartability.<br>Recover index SYSUT1 optional for performance choice. | Use inline copies from LOAD and Reorg.<br>Recover index unload phase performance (like SORTDATA). | Fast LOG apply.<br>Recover of indexes from copies (vs. Rebuild).<br>Recover of table space and indexes with single log scan.<br>PARALLEL recover. |
| Rebuild | N/A | N/A (APAR) | Inline Runstats.<br>SORTKEYS for indexes in parallel. |
| Quiesce | | | TABLESPACESET support. |
| ALL | Partition independence (NPI) from type 2 indexes.<br>BSAM I/O buffers.<br>Type 2 index performance. | BSAM striping for work data sets | Avoid delete and redefine of data sets (except for Copy). |

Since DB2 V1 IBM has enhanced the functionality, performance, availability, and ease of use of the initial set of utilities. Starting with V4 the progression of changes has increased. This table summarizes these changes.

Details on functions and related performance by DB2 version are reported in the redbooks *DB2 for MVS/ESA Version 4 Non-Data-Sharing Performance Topics,* SG24-4562, *DB2 for OS/390 Version 5 Performance Topics,* SG24-2213, and *DB2 UDB for OS/390 Version 6 Performance Topics,* SG24-5351. Other functions not included here but worth mentioning are the support for very large tables (DSSIZE) and pieces for non-partitioning indexes.

The trend of enhancements continues with DB2 V7.

© Copyright IBM Corp. 2001




## 5.1 Utilities with DB2 V7

DB2 Version 7 introduces several new functions and a different way of packaging these functions.

***New packaging of utilities***
With Version 7, many of these enhanced utility functions have been separated from the base product and are now offered as optional products of DB2. The three new utility products are:

- DB2 Operational Utilities
- DB2 Recovery and Diagnostic Utilities
- DB2 Utilities Suite

The Utilities Suite contains the whole set of DB2 utilities combining the two other products and provides the most cost effective option.

***Dynamic utility jobs***
DB2 V7 introduces two new utility control statements: TEMPLATE and LISTDEF. They provide for the dynamic allocation of data sets and for the dynamic processing of lists of DB2 objects with one utility invocation. These new statements can now be used by most DB2 utilities. Using these new statements will dramatically change your utility jobs and reduce the cost of maintaining them. They are also a prerequisite for partition parallelism within the same job.

***New utility - UNLOAD***
The UNLOAD utility unloads data from one or more source objects, table spaces or image copies, to one or more BSAM sequential data sets in external formats. It is

easier to use than the previously available REORG UNLOAD EXTERNAL, and it is faster than DSNTIAUL. UNLOAD also offers new capabilities when compared to the still available REORG UNLOAD EXTERNAL.

***New utility - MODIFY STATISTICS***
The MODIFY STATISTICS online utility deletes unwanted RUNSTATS statistics history records from the corresponding catalog tables which are now created by a new RUNSTATS function (see “RUNSTATS statistics history” on page 191). You can remove statistics history records that were written before a specific date, or you can remove records of a specific age.

***New utility COPYTOCOPY***
The COPYTO COPY online utility provides the function of making additional copies, local and/or remote, and recording them in the DB2 Catalog.

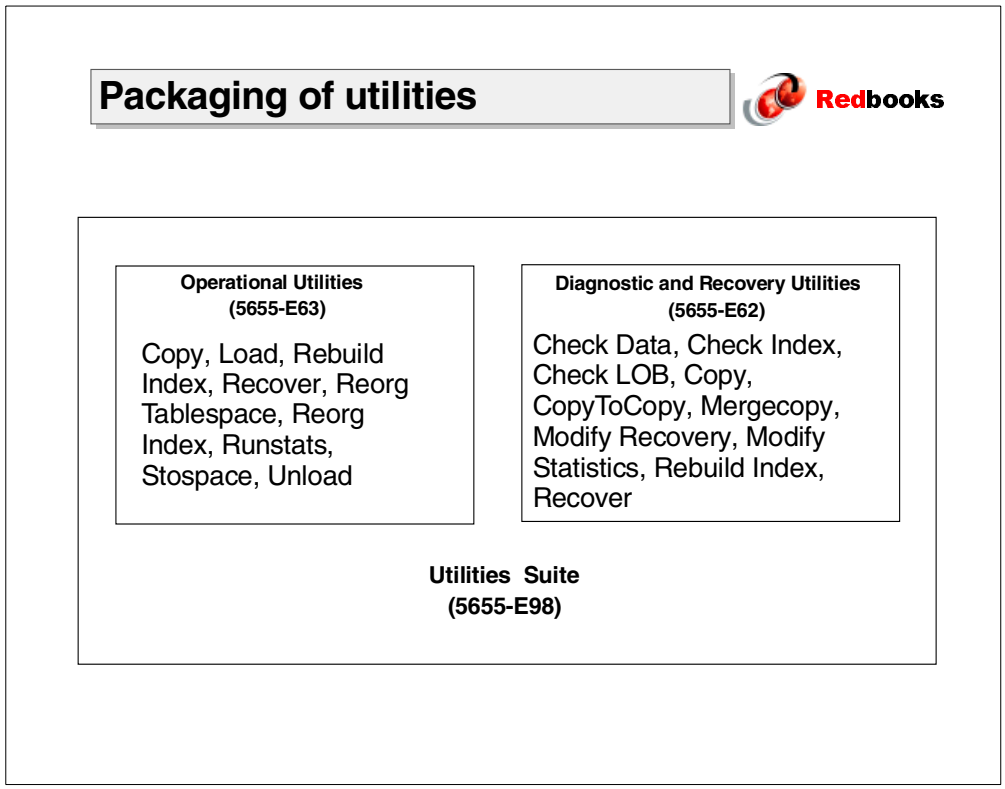
***LOAD partition parallelism***
Load now provides enhanced availability when loading data and indexes on different partitions in parallel within the same job from multiple inputs.

***Cross Loader***
A new extension to the Load utility which allows you to load output from a SELECT statement. The input to the SELECT can be anywhere within the scope of DRDA connectivity.

***Online REORG enhancements***
Online REORG no longer renames data sets, greatly reducing the time that data is unavailable during the SWITCH phase. You specify a new keyword, FASTSWITCH, which keeps the data set name unchanged and updates the catalog to reference the newly reorganized data set. Additional parallel processing improves the elapsed time of the NPIs BUILD2 phase of REORG SHRLEVEL(CHANGE) or SHRLEVEL(REFERENCE). New parameters allow more granularity in draining and waiting for resources to be available.

***Online LOAD RESUME***
Prior to V7, DB2 restricts access to data during LOAD processing. V7 offers the choice of allowing user read and write access to the data during LOAD RESUME processing, so that loading data is concurrent with user transactions. This new feature of the LOAD utility is almost like a new utility of its own: it works internally more like a mass insert rather than a LOAD. The major advantage is availability with integrity.

***RUNSTATS statistics history***
Changes to RUNSTATS provide the possibility to keep multiple versions of the statistics across time and allow tools and users more flexibility on monitoring the trends in data changes.




## 5.2 New packaging of utilities

A basic set of core utilities are included as part of DB2 since Version 1 was first delivered. These utilities initially provided a basic level of services to allow customers to manage data. Some customers have preferred to obtain such functions, however, from independent software vendors that have developed utility and tools offerings that offered additional performance, function, and features beyond that contained in the DB2 core utilities. With recent releases of DB2 for OS/390, in response to clear customer demand, IBM has invested in the improvement of the performance and functional characteristics of these utilities, as we have seen in the previous section.

With DB2 V7 the Utilities have been separated from the base product and they are now offered as separate products licensed under the IBM Program License Agreement (IPLA), and the optional associated agreements for Acquisition of Support. This combination of agreements provides to the users equivalent benefits to the previous traditional ICA license. The DB2 Utilities are grouped in:

- DB2 Operational Utilities

  This product, program number 5655-E63, includes Copy, Load (including Cross Loader), Rebuild Index, Recover, Reorg Tablespace, Reorg Index, Runstats (enhanced with history), Stospace, and Unload (new).

- DB2 Diagnostic and Recovery Utilities

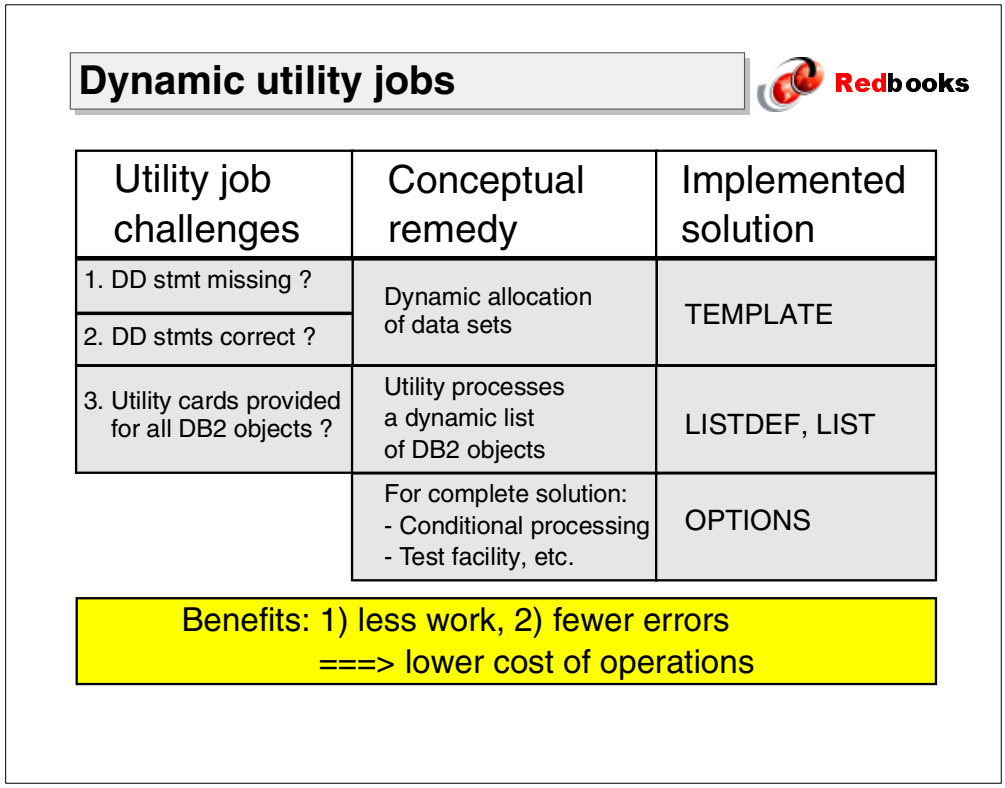
  This product, program number 5655-E62, includes Check Data, Check Index, Check LOB, Copy, CopyToCopy (new), Mergecopy, Modify Recovery, Modify Statistics (new), Rebuild Index, and Recover.

- DB2 Utilities Suite

  This product, program number 5697-E98, combines the functions of both DB2 Operational Utilities and DB2 Diagnostic and Recovery Utilities in the most cost effective option.

These products must be installed separately from DB2 V7 when accessing user data; they are all however available within DB2 when accessing the DB2 catalog, directory, or the sample database. You can install one or both of them and give them a test run during the provided trial period. Verify the benefits they bring to your database operations.

The following utilities, and all standalone utilities, are considered core utilities and are included and activated with DB2 V7: CATMAINT, DIAGNOSE, LISTDEF, OPTIONS, QUIESCE, REPAIR, REPORT, TEMPLATE, and DSNUTILS.




## 5.3 Dynamic utility jobs

Development and maintenance of utility jobs can be very time consuming and error prone. Users primarily face three challenges:

1. DD statements must be provided for all data sets
2. Within each DD statement, space, disposition, and other parameters must reflect the current situation
3. Utility statements must explicitly list all DB2 objects to be processed and these objects must be named accurately.

As new objects are constantly created, others are deleted, and since the sizes of most objects vary over time, it is particularly difficult to keep up with the changes.

DB2 V7 addresses these three challenges by introducing the following new utility control statements:

- **TEMPLATE template-name**

  With this utility control statement, you define a dynamic list of date set allocations. More precisely:

  - You create a skeleton or a pattern for the names of the data sets to allocate.
  - The list of these data sets to allocate is dynamic. That is, this list is generated each time the template is used by an executing utility. Therefore, a template automatically reflects the data set allocations currently needed.

- The allocation of the data sets is dynamic. That is, the data sets are not allocated at the beginning of the job step, but each invoked utility allocates the data sets at execution time.

  More specific information is given in 5.3.1.5, “Data set allocation - things to consider” on page 204 and in 5.3.3, “TEMPLATE and LISTDEF combined” on page 223.

- You make use of a template by specifying the name of the template within a utility control statement; for example, during an image copy, within the COPY control statement.

- **LISTDEF list-name**

  With this utility control statement you define a dynamic list of DB2 objects, namely table spaces, index spaces, or their partitions. More precisely:

  - You provide the rules or the algorithm used to generate the list of such DB2 objects.
  - This list is dynamic. That is, the actual list is generated each time the list is used by an executing utility. Therefore, a list automatically reflects the existent DB2 objects.
  - You make use of such a list by specifying the name of the list after the keyword **LIST** within a utility control statement, for example, within the COPY control statement.

- **OPTIONS**

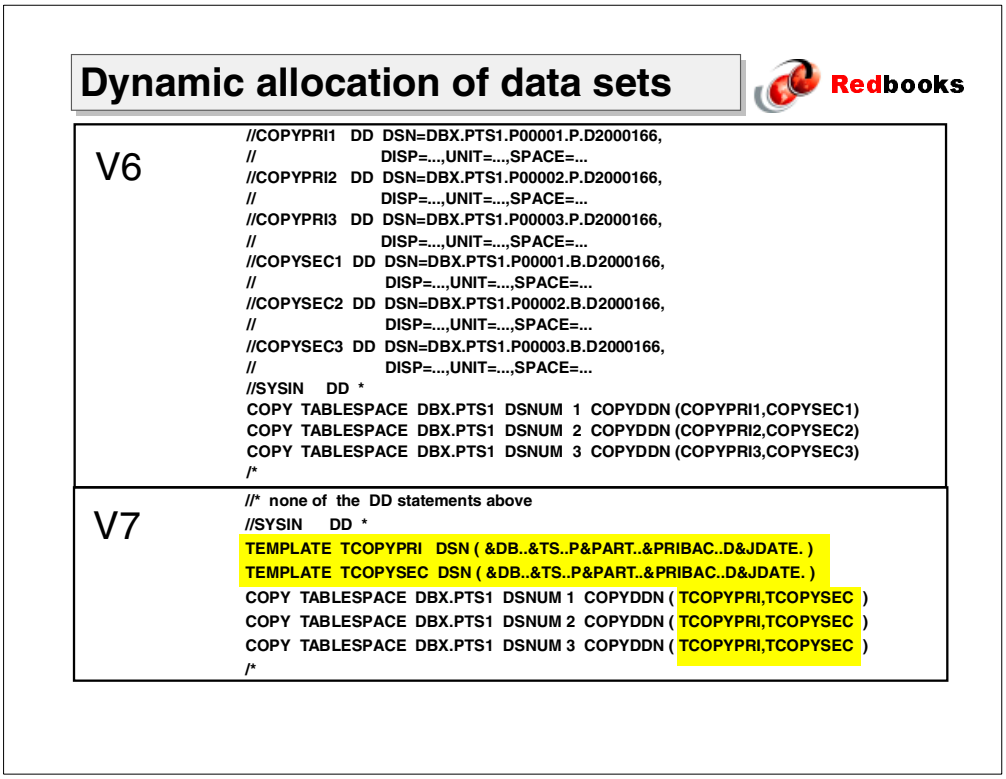
  This third new utility control statement serves various purposes, for example, conditional processing of objects, or as a test tool for your template and list definitions.

The benefits of these three new utility control statements are evident: The development and maintenance of the jobs is easier; and as changes are reflected automatically, they require less user activity and also reduce the possibility of errors. As a consequence, the total cost of operations can be reduced.

The actual usage of these statements will have a tremendous impact on your utility jobs, for example, a lot of individually developed REXX or CLIST procedures can be replaced now, using this “DB2 standardized” method instead.

In the following sections we present in more detail:

- Dynamic allocation of utility data sets via TEMPLATE
- Processing dynamic lists of DB2 objects via LISTDEF/LIST
- Combination of LISTDEF/LIST and TEMPLATE
- Library data set and DB2I support
- Several other goodies: OPTIONS




## 5.3.1 Dynamic allocation of data sets

In this section, we present the following subjects related to dynamic allocation of utility data sets using the TEMPLATE:

- Introductory example for TEMPLATE
- Substitution variables for data set names
- Dynamic space allocation via TEMPLATE
- Dispositions via TEMPLATE
- Data set allocation - things to consider
- TEMPLATE - syntax diagram
- Supporting utilities and DSNUTILS

### 5.3.1.1 Introductory example for TEMPLATE

Here we start with the first challenge: how to provide a complete set of data set allocations. Making use of the naming convention which is normally already in place at the customers' sites, we define and use templates. These dynamic lists of data set allocations facilitate the specification of the various data sets.

In our introductory example, we consider a table space, PTS1, with three partitions, residing in database DBX, and assume that we need to take image copies of these objects.

The foil first shows the traditional DB2 V6 job for generating primary and backup copies by partition.

With DB2 V7, by using templates, you can code an alternative job. This job has the same functionality but, instead of explicitly coding the DD statements:

1. You define two templates, named TCOPYPRI and TCOPYSEC, which outline the data set names (and optionally some allocation parameters as well) which DB2 should use.

   These definitions are done with the utility control statement TEMPLATE followed by the name of the template you want to create, for example, TCOPYPRI, and a pattern or skeleton for the data set name(s). This pattern is largely made up of variables (such as &TS.). How values are assigned to these variables is explained later.

   Actually, one template would be sufficient. But here two templates are used in order to demonstrate the replacement of DD names within the COPY statement by templates.

2. You code the utility control statement (in this example is COPY) almost as you did before, just use the template name(s) rather than the DD name(s) within this statement.

***What happens at utility execution time***

At first DB2 reads the TEMPLATE utility control statement with all the variables, for example &TS. Having read the COPY statement, DB2 knows the table space name, which is to be processed by this utility (in our example is PTS1). Therefore, DB2 can assign the value PTS1 to the &TS. variable. Similarly, all the other variables in the TEMPLATE utility control statement are filled with values.

This way the data set names, which are derived from the TEMPLATE and the COPY statements, are exactly the same as the ones specified in the V6 job.

Once the data set names are determined DB2 is able to dynamically allocate these data sets. DB2 also provides default DD names for these allocations. As you can see in the following job output, DB2 generates the DD names SYS00001, SYS00002, and so forth. These DD names are different from the ones in the V6 job.

This simple example already shows the advantages of using templates. Utility jobs can be developed and maintained more easily with:

- Fewer lines of code
- Number of data set allocations and the pertaining data set names are maintained automatically.

For example, you do not have to modify your jobs when data set names contain time references.

Note that the job output provides information on the allocation errors and helps you in identifying the cause. An example of job output follows.

### *Sample utility job output for TEMPLATE*

We list here a part of the utility output, run in V7, to show how DB2 translates the sample templates TCOPYPRI, TCOPYSEC to actual DD names and data set names:

```
- OUTPUT START FOR UTILITY, UTILID = TEMP
-  TEMPLATE TCOPYPRI DSN(&DB..&TS..P&PART..&PRIBAC..D&JDATE.)
- TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
-  TEMPLATE TCOPYSEC DSN(&DB..&TS..P&PART..&PRIBAC..D&JDATE.)
- TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
-  COPY TABLESPACE DBX.PTS1 DSNUM 1 COPYDDN(TCOPYPRI,TCOPYSEC)
- DATASET ALLOCATED.  TEMPLATE=TCOPYPRI
   DDNAME=SYS00001
   DSN=DBX.PTS1.P00001.P.D2000166
- DATASET ALLOCATED.  TEMPLATE=TCOPYSEC
   DDNAME=SYS00002
   DSN=DBX.PTS1.P00001.B.D2000166
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 1
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 33.00
 PERCENT OF CHANGED PAGES = 25.00
 ELAPSED TIME=00:00:00
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 1
-  COPY TABLESPACE DBX.PTS1 DSNUM 2 COPYDDN(TCOPYPRI,TCOPYSEC)
- DATASET ALLOCATED.  TEMPLATE=TCOPYPRI
   DDNAME=SYS00003
   DSN=DBX.PTS1.P00002.P.D2000166
- DATASET ALLOCATED.  TEMPLATE=TCOPYSEC
   DDNAME=SYS00004
   DSN=DBX.PTS1.P00002.B.D2000166
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 2
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 33.00
 PERCENT OF CHANGED PAGES = 25.00
 ELAPSED TIME=00:00:00
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 2
-  COPY TABLESPACE DBX.PTS1 DSNUM 3 COPYDDN(TCOPYPRI,TCOPYSEC)
- DATASET ALLOCATED.  TEMPLATE=TCOPYPRI
   DDNAME=SYS00005
   DSN=DBX.PTS1.P00003.P.D2000166
- DATASET ALLOCATED.  TEMPLATE=TCOPYSEC
   DDNAME=SYS00006
   DSN=DBX.PTS1.P00003.B.D2000166
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 3
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 32.00
 PERCENT OF CHANGED PAGES = 25.00
 ELAPSED TIME=00:00:00
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 3
- UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

### 5.3.1.2 Substitution variables for data set names

A broad set of DSN substitution variables are supported, some of which you have already seen on the previous foil, for providing dynamic data set names:

| JOB variables | UTILITY variables | OBJECT variables | DATE and TIME variables |
|---|---|---|---|
| JOBNAME | UTIL<br>(utility name truncated) | DB<br>(db-name) | DATE<br>(yyyyddd) |
| STEPNAME | ICTYPE<br>(“F” or “I”) | TS<br>(table space name) | YEAR<br>(yyyy) |
| USERID or US | LOCREM or LR<br>(“L” or “R”) | IS<br>(index space name) | MONTH<br>(mm) |
| SSID<br>(subsystem ID) | PRIBAK or PB<br>(“P” or “B”) | SN<br>(space name) | DAY<br>(dd) |
| | | PART<br>(five digits) | JDATE or JU<br>(Julian date: yyyyddd) |
| | | LIST<br>(name of the list) | JDAY<br>(ddd) |
| | | SEQ<br>(sequence number of object in the list) | TIME<br>(hhmmss) |
| | | | HOUR |
| | | | MINUTE |
| | | | SECOND or SC |

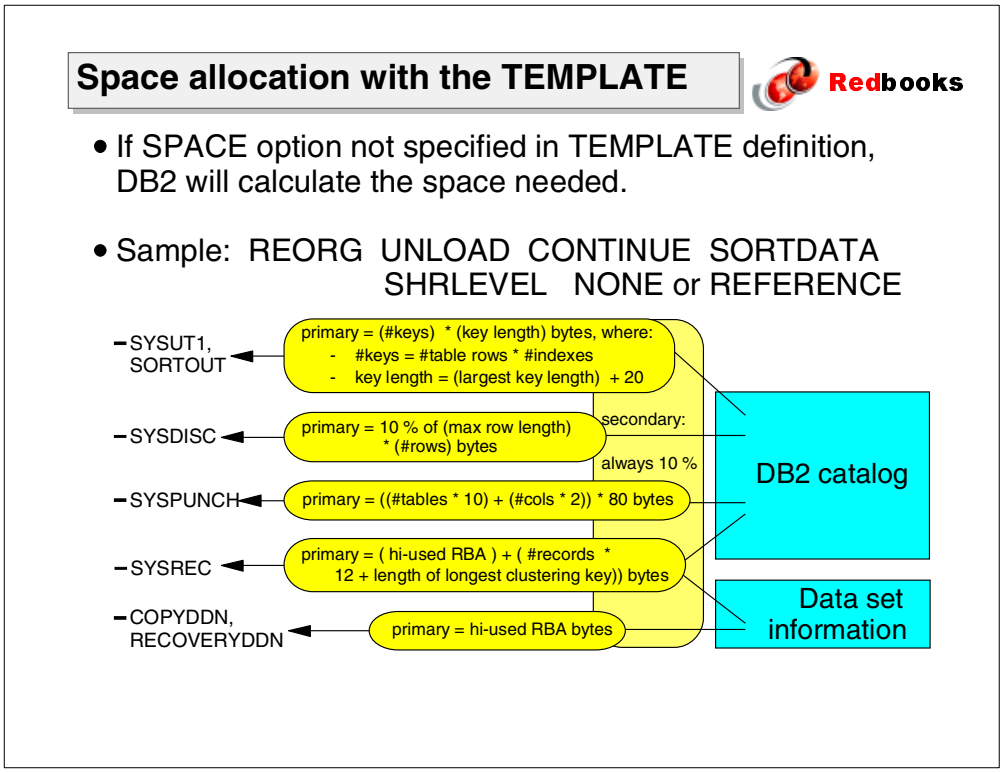
**Notes:**

- When you use these variables in your template definition, they must have a leading ampersand and a trailing period (for example, &TS.).
- Some substitution variables are invalid if used with the wrong utility. For example, ICTYPE with the LOAD utility.
- If the value of a variable starts with a digit, the variable cannot be used as a start of a qualifier of a data set name.
- If not specified otherwise, the first two characters of a variable may be used as an abbreviation of the same variable. Short forms are preferable when writing templates with longer, more complex data set names.
- Actually, TS, IS, and SN are synonyms; therefore, it is not an error when TS is used, for example, with an index space. This can be useful when you want to copy or recover a list including both, table spaces and copy-enabled index spaces.
- If PART is specified for a non-partitioned object, it evaluates to ’00000’.
- If PART is used in conjunction with LISTDEF (see below), then you must specify the PARTLEVEL keyword in the LISTDEF definition.

- If a utility processes a list, the variable LIST stores the name of that list. LIST is used together with COPY FILTERDDN templates. Copies of all objects in the entire list go to one data set, thus making the variables TS, IS, and SN meaningless.
- SSID (or SS) holds the subsystem ID (non-data sharing) or the Group Attach Name (data sharing). That is, there is no variable for a specific member of a data sharing group.
- Date/time values are captured in the UTILINIT phase of each utility and remain constant until the utility terminates.

Notes on using variables for the templates:

- You must assure that the templates follows the installation standards in order to minimize the impact on handling the names of the DB2 objects. For instance in reducing the need to changing SMS routines to handle only the ‘new’ objects introduced by online Reorg.
- You must also assure that, using templates, a data set is not allocated with a DSN that is being used by another MVS job. In this case conflict can occur at dynamic allocation time and the utility can wait on the resource. This can be avoided by including &JOB. and &STEP. in the TEMPLATE DSN variable.




### 5.3.1.3 Space allocation with the TEMPLATE

The second challenge we mentioned is the provision of correct parameters for the DB2 generated data set allocations. Most notably, the space parameters should reflect the current size of the table space or index space and the different space requirements for the different DASD data sets used by a particular utility.

As DB2 automatically allocates these data sets when using templates, DB2 must also generate these correct space parameters for each data set automatically, if you have not specified the space option in your template definition. To this end, DB2 uses formulas which are specific for each utility and each data set.

The input values used in these formulas mostly come from the DB2 catalog tables. The high-used RBA is read at open time and it is maintained by the buffer manager. It is the most current value, updated before it is even written to the ICF catalog.

All these formulas are documented in the standard DB2 manuals. The foil just presents, as an example, the specific formulas for the data sets for the REORG utility in case you use REORG UNLOAD CONTINUE SORTDATA KEEPDICTIONARY SHRLEVEL NONE or SHRLEVEL REFERENCE.

## Dispositions with the TEMPLATE

- If the DISPOSITION option is not specified in the TEMPLATE definition, DB2 will use default dispositions:
  - Defaults vary depending on utility and data set
  - Defaults differ for restarted utilities from new utility executions

- Example: REORG

| | | status | normal termination | abnormal termination |
|---|---|---|---|---|
| New utility | SYSREC | NEW | CATLG | CATLG |
| execution | SYSUT1 | NEW | DELETE | CATLG |
| | SORTOUT | NEW | DELETE | CATLG |
| | | | | |
| Restarted | SYSREC | MOD | CATLG | CATLG |
| utility | SYSUT1 | MOD | DELETE | CATLG |
| | SORTOUT | MOD | DELETE | CATLG |

### 5.3.1.4 Dispositions with the TEMPLATE

In a similar manner as for support of default space allocations, DB2 chooses default disposition parameters for the data sets if they are allocated with the templates and if you have not specified the disposition in your template definition.

The defaults are documented; the foil just presents some examples for the REORG utility.

***Restart support***

As an example we outline how these disposition defaults support restarts of the REORG utility. We assume that neither SORTDATA, NOSYSREC, SORTKEYS, SHRLEVEL REFERENCE, nor CHANGE are specified.

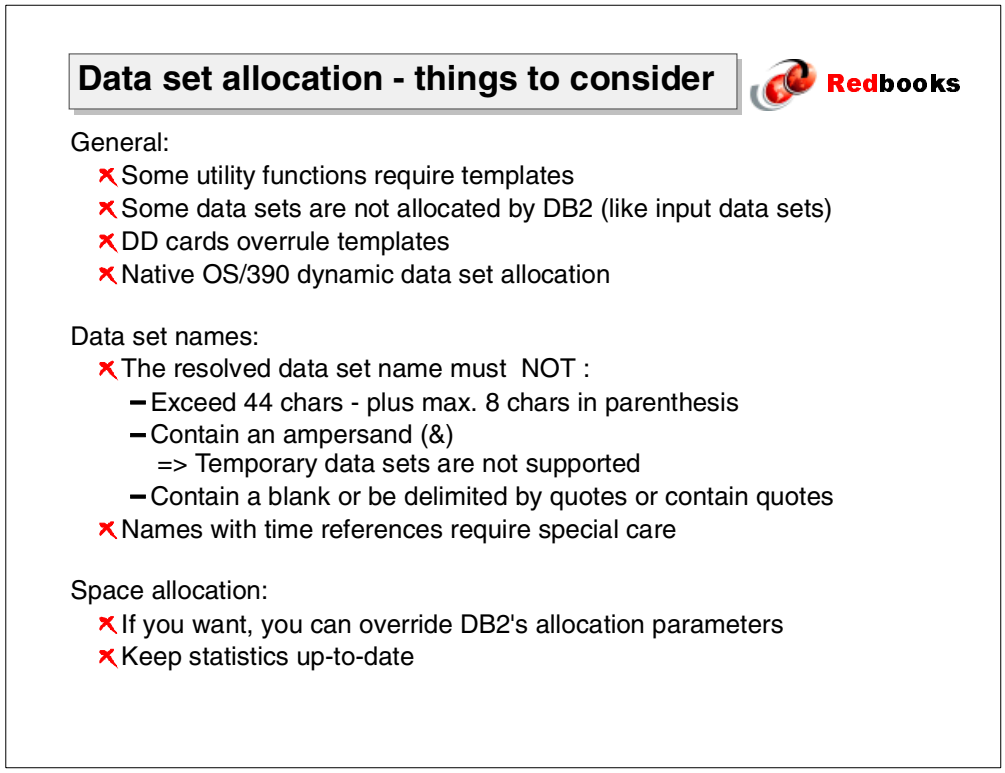
If the REORG fails, it is often due to space problems (such as B37) with either the SYSREC, SYSUT1, or SORTOUT data sets; this normally happens if statistics are not current. Therefore, the abnormal termination disposition for the corresponding data sets should be CATLG in order not to lose your data. This is especially true when the failure occurs after the deletion of the original table space and before the end of the RELOAD phase, when the data resides in the SYSREC data set only (if no COPY preceded the REORG). With the default dispositions on the foil, you have the best support when restarting a REORG. For details, see the following sample cases:

| B37 for: | SYSREC data set | SYSUT1 data set | SORTOUT data set |
|---|---|---|---|
| Abend in: | UNLOAD phase | RELOAD phase | SORT phase |
| Situation:<br><br><br><br>- positive: | - Utility stopped<br>- SYSREC data set filled incompletely | - Utility stopped<br>- SYSUT1 data set filled incompletely<br><br>- SYSREC data set filled completely and **cataloged** (see disposition) | - Utility stopped<br>- SORTOUT data set filled incompletely<br><br>- SYSREC, SYSUT1 data sets filled completely and **cataloged** (see disposition) |
| Your actions: | - Correct the error<br>- Delete SYSREC data set<br>- Terminate the utility<br>- **Start** REORG from scratch | - Correct the error<br>- Delete SYSUT1 data set<br><br>- **Restart** REORG (with RELOAD phase) | - Correct the error<br>- Delete SORTOUT data set<br><br>- **Restart** REORG (with SORT phase) |

Benefits:

- This table shows the advantage of the default disposition for abnormal termination, CATLG, as this value prevents losing the data sets which are already completely filled by the original REORG job. So the restart job can use these data sets.
- The foil shows that you do not have to change the disposition parameters from NEW to MOD or OLD for the SYSREC or SYSUT1 data sets anymore when restarting the REORG utility job after these sample B37 abends when you use templates and their default dispositions.

**Note:** If you decide differently from the defaults, objects may be left around and will need cleaning up afterwards.




### 5.3.1.5 Data set allocation - things to consider

In this section we add some considerations on data set allocation.

***Some new utility functions require templates***

The new UNLOAD utility supports the unloading of partitions. Templates are required if you additionally want:

- To have separate output data sets per partition, rather than a single output data set for the specified partition(s).
- To have the partitions unloaded in parallel.

In this case, the template must contain the &PART. (or &PA.) variable.

***Some data sets are not supported***

Dynamic allocation by means of templates is not supported for the following data sets:

- When there is no keyword in the utility control statement, so that the template name could replace the DD name for that data set. For example, DATAWKxx for REORG, SORTWKxx for LOAD or REORG
- When it is an input data set, for example, SYSREC for LOAD, input image copies for RECOVER or UNLOAD. In other words, templates generally support dynamic allocation for output data sets only.

***DD cards override templates***

If a template name is the same as any DD name in the same job step, the DD card takes precedence, and the template, even though defined, will not be used.

***Native OS/390 dynamic data set allocation***
DB2 does not behave differently from the native OS/390 dynamic data set allocation; for example, no additional enqueueing is performed by DB2. That is, the normal rules apply, the job will not wait if the allocation is not successful; data sets are allocated until the end of the job step. For dynamic allocation details, see *MVS/ESA SP V5 Programming: Authorized Assembler Services Guide,* GC28-1467-02.

What are the consequences when using templates? If two jobs happen to name the two output data sets the same, the second job will fail if it wants to allocate its output data set when the data set with the same name is still allocated by the first job. This is different from using DD cards in your jobs, as the second job will wait for the data set, in this case, due to MVS initiator enqueue services. Therefore, avoid using the same data set names in different jobs, for example, by using the &JOBNAME. variable - or ensure that your jobs run in sequence, if the second job should really append some data to the same output data set.

***Length of data set names***
Data set names cannot exceed 44 characters - plus a maximum of 8 characters in parentheses (for PDS members or GDG generation expression). The number 44 derives from the limit of maximal 5 qualifiers for a data set - plus the 4 separating periods.

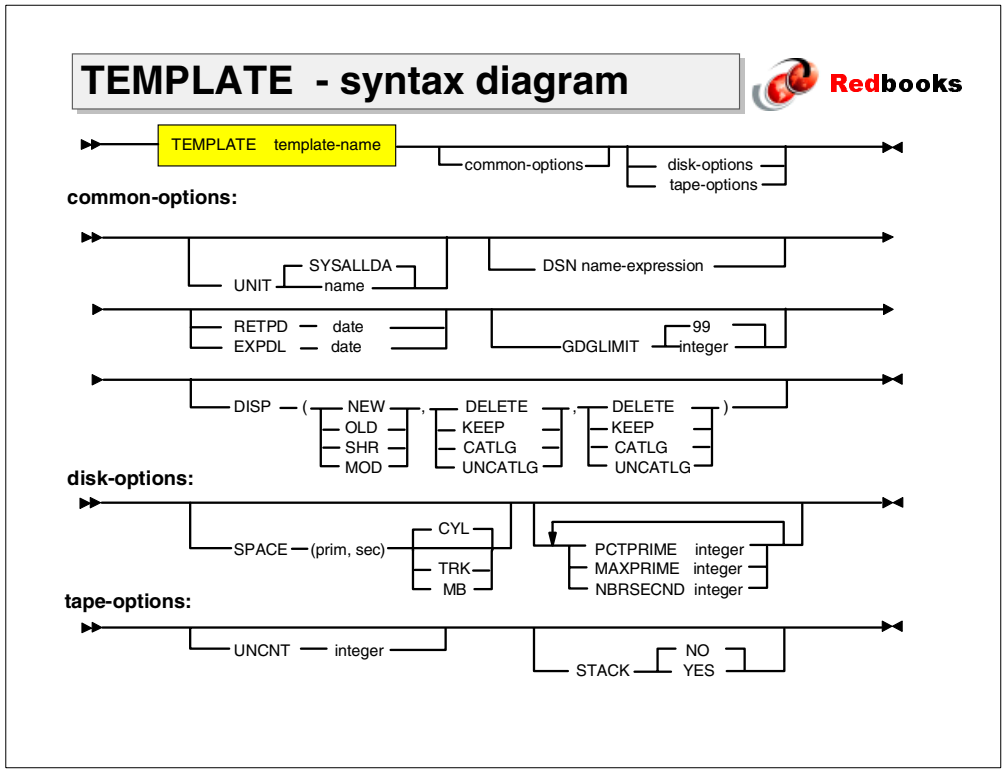
***Data set names with time references***
When using a substitution variable with a time reference, for example, &JDATE., in your template definition, the question arises what happens when restarting a *stopped* utility. Answer: As DB2 keeps track of this utility in SYSUTILX, DB2 can and does ensure that the change of the value of that variable will not affect data sets that were already opened by the utility prior to stopping. Only newly allocated data sets will be impacted by the changed variable value.

But suppose a utility *terminated* due to an error condition, for example, due to an authorization error, then the user corrects the error and starts the utility once again from scratch. In this case, as the utility terminated, DB2 has no data about the utility, the values of the time variables, or the data set names any more. So new data set names are used.

Please be aware that there might be left-over data sets from the unsuccessful first utility run, for example, a discard data set. The user should develop a clean-up job to either inspect these data sets and/or to get rid of them.

***Override DB2's space allocation***
In some cases, you might want to choose different space parameters than DB2, for example, if DB2's primary allocation might not fit onto one single volume. You can provide your own space specification in the TEMPLATE statement (see 5.3.1.6, "TEMPLATE syntax diagram" on page 206) overriding DB2's space calculation or you can still use DD statements for these data sets.

***Current statistics***
The need for keeping the statistics current in the catalog is quite obvious, as DB2's space calculation is based on catalog statistics and also takes compression into account. It is extremely important that these statistics are always current so that your jobs do not fail. This might influence your schedules for the RUNSTATS jobs.




### 5.3.1.6 TEMPLATE syntax diagram

Having seen the introductory example, we will take now a closer look at the overall TEMPLATE statement. There are three groups of template options: `common-options`, `disk-options`, and `tape-options`. For learning purposes, the syntax diagram on this foil is still simplified, for example, only the most important options are listed:

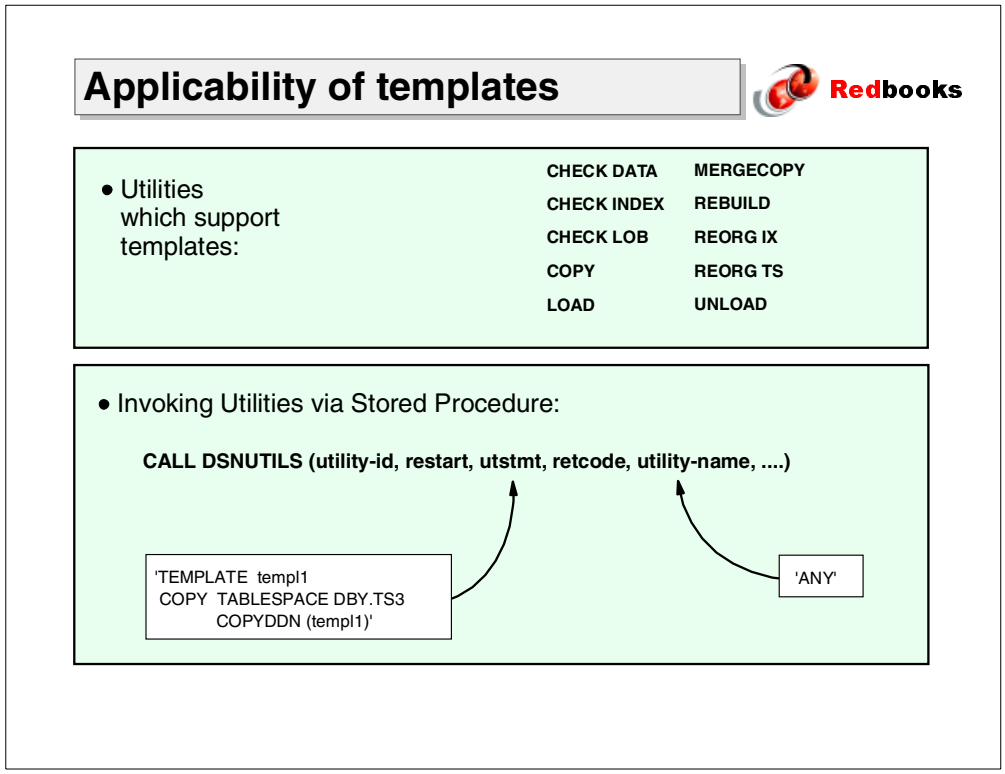
- `TEMPLATE template-name`. In order to be able to use this template in a subsequent utility control statement, you have to assign a name to the template you create. The following rules apply for that name:
  - Up to 8 alphanumeric characters
  - It cannot be UTPRINT, SORTLIB; it cannot start with SYS or SORTWK
- `UNIT name`. The MVS unit parameter, for example, SYSDA, or a specific volume id, or CART, or TAPE
- `DSN name-expression`. An explicit specification of one or a pattern for several data set names by means of the substitution variables, as shown earlier. Restrictions apply as stated earlier.
- `RETPD` or `EXPDL`. Retention period in days or expiration date
  (In contrast to the simplified syntax diagram, both options can be specified.)
- `GDGLIMIT integer`. The number of entries to be created in a GDG base if a GDG DSNAME is specified and the base does not already exist.
- `DISP`. Standard MVS disposition values are allowed. Default values vary depending on the utility and the data set being allocated. Default for restarted utilities also differ from those for new utility executions.

- SPACE (primary, secondary). Means to overwrite the values DB2 would calculate. For CYL and TRK, DB2 will use 3390 capacities when transforming into bytes.
- If you are concerned whether the primary allocation could become too large, you have also got the following options:
  - PCTPRIME integer. Percentage of the calculated required space to be obtained as primary quantity (default 100)
  - MAXPRIME integer. Maximum allowable primary space allocation (in MB)
  - NBRSECND integer. Number of secondary space allocation chunks for the remaining space after the primary space is allocated (1 to default 10).
- UNCNT integer. Maximum number of allocated tape volumes - unit count (0-59).
- STACK NO | YES. Whether output data sets are to be stacked as successive files on the same logical tape volumes.

Other options are supported, but not listed on the partial syntax diagram:

- The common-options:
  - CATLG YES ! NO. For redefining the MVS catalog directive
  - MODELDCB dsname. Using the DCB information of a model data set
  - VOLCNT integer. Specification of largest number of volumes expected to be processed when copying a single data set.
  - BUFNO integer (default: 30). Number of BSAM buffers (0-99. default: 30)
  - For DF/SMS, if the distribution of the data sets onto volumes should not be controlled by the data set names, you can specify the parameters:
    DATACLAS name. Specification of the SMS data class
    MGMTCLAS name. Specification of the SMS management class
    STORCLAS name. Specification of the SMS storage class

    For all three SMS options, the value must be a valid class, the default value is NONE, and the data set will be cataloged if the option is specified.
  - VOLUMES (volser-list). Specification of the list of available volumes. There must be enough space on the first volume for the primary space allocation.
- The disk-options:
  - DASDMAX (cylinders, tape-template-name). For redirecting to tape template if use of original template exceeds DASDMAX.
- The tape-options:
  - JES3DD ddname. JCL DD name to be used at job initialization time for the tape unit. JES3 requires all tape allocations to be specified in the JCL.
  - TRTCH NONE, COMP, or NOCOMP. Specification of the Track Recording Technique for magnetic tape drives with Improved Data Recording Capability: NONE (the default) for eliminating the TRTCH specification from dynamic allocation, COMP for writing data in compacted format, NOCOMP for writing data in standard format.
  - TAPEMIN (cylinders, dasd-template-name). Opposite of DASDMAX.




### 5.3.1.7 Applicability of templates

Templates are used by the supporting utilities and DSNUTILS, the stored procedure utility interface.

***Most online utilities support templates***

All the online utilities listed on this foil support templates. In general, the remaining online utilities do not require DD statements: DIAGNOSE, MODIFY, QUIESCE, RECOVER, REPAIR, REPORT, RUNSTATS, STOSPACE.

Concerning the supporting utilities, the support varies by utility and data set requirement. As stated before, dynamic allocation by means of templates is not supported in the following cases:

- No keyword for DD names invoking templates in the utility control statement
- Input data sets

***Invoking utilities via stored procedure***

The usage of templates has been illustrated with utilities invoked by DSNUPROC; now we will look at utilities invoked via the stored procedure DSNUTILS.

In previous releases, DSNUTILS performed its own dynamic allocation of data sets before invoking DB2 utilities (program DSNUTILB). The `utility-name` parameter of DSNUTILS was used to specify what type of dynamic allocation to perform, because this depends on the utility.

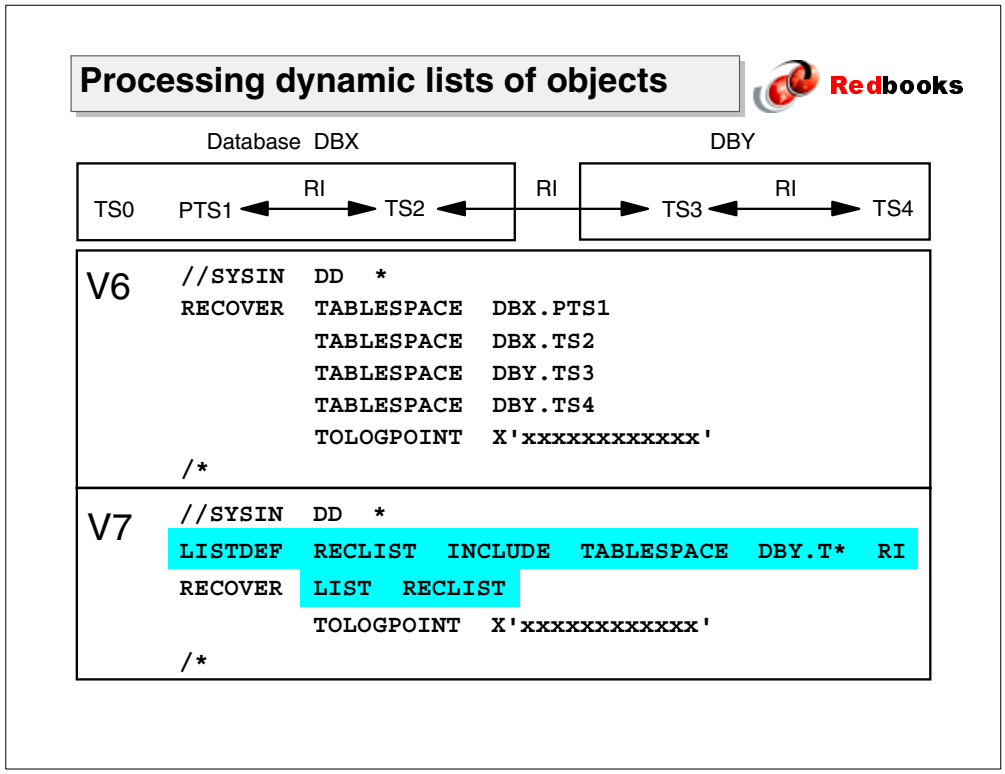
In order to allow TEMPLATE dynamic allocation, you need to specify two new parameters, as follows:

- The new value `ANY` instead of the `utility-name` parameter

  This will cause DSNUTILS not to perform dynamic allocation prior to utility invocation.

- The TEMPLATE control statement included in the utility statement pointed by the `utstmt` parameter

  The TEMPLATE control statements cause that form of dynamic allocation to take place. You do this by providing a TEMPLATE statement (and optionally a LISTDEF statement, see below) before your utility statement in the `utstmt` parameter, as in the example on the foil.




## 5.3.2 Processing dynamic lists of DB2 objects

In this section we talk about:

- Introductory example for LISTDEF/LIST
- LISTDEF - syntax diagram
- LISTDEF expansion
- LISTDEF/LIST - miscellaneous notes

### 5.3.2.1 Introductory example for LISTDEF/LIST

Having already replaced the DD statements and related disposition parameters by templates, we now address the third mentioned challenge when coding a utility job: how to generate a complete and accurate list of all DB2 objects to be processed by the utility (or by several utilities).

Let us suppose that we have already executed the following preparatory activities for the DB2 objects listed at the top of the foil:

- Creation of two databases with the five table spaces: TS0, PTS1, and TS2 in DB2X, and TS3 and TS4 in DB2Y.
- Creation of one table in each table space
- Addition of three referential constraints as outlined on the foil
- Insertion of some rows in each table
- QUIESCE TABLESPACESET TABLESPACE DBY.TS4

***Capability of utilities to process lists of objects***
In DB2 V6 you need to distinguish two cases:

- Some utilities can already process a list of objects. For example with RECOVER you must provide the explicit list in the RECOVER statement. In the example on the foil we have created five table spaces in two databases. We want to recover a table space set, consisting of the table spaces DBX.PTS1, DBX.TS2, DBY.TS3, and DBY.TS4, to a prior point in time. The foil shows how you would code this job in DB2 V6.
- Some utilities can only process one object (for example, LOAD, MERGECOPY, MODIFY) or a limited list of objects (for example, REBUILD). Then you must ensure that enough utility statements exist so that all DB2 objects will be processed. See the following REBUILD example.

***V7 code sample***
DB2 V7 introduces the new utility control statement **LISTDEF** for defining a dynamic list of DB2 objects, namely table spaces, index spaces, or their partitions. As mentioned before, the actual list is generated each time it is used by an executing utility. Therefore, a list automatically reflects which DB2 objects currently exist. Now all utilities can process lists of objects, if these lists are generated by a LISTDEF utility control statement.

In DB2 V7, you can then code an alternative job, which has the same functionality, but instead of specifying the four table spaces explicitly, you do the following:

1. First, you define a list, with name RECLIST, which contains exactly these four table spaces - as of this moment. This definition is done via the utility control statement **LISTDEF,** followed by the name of the list you want to create, here RECLIST, and some list generating options. Included in your list are:
   - The DB2 object you specify explicitly or those DB2 objects, whose names match the pattern you provide - in a similar way to the LIKE construct in SQL. As for templates, it is very helpful, if naming conventions are already in place.

     In the example on the foil, this pattern is TABLESPACE DBY.T*, hence the list initially includes the table spaces DBY.TS3 and DBY.TS4.
   - Those DB2 objects that are found when DB2 complies with the type of relationships you specify.

     In the example above, our specification is RI, so DB2 also includes those table spaces in the list which, by their respective tables, are “referentially connected” to table spaces already in the list. So, DBX.TS2 and DBX.PTS1 are added to the initial list, but not DBX.TS0.
2. Then, you use this named list for your utility job.

   To this end you must change your original utility statement: Specify the defined list after the new keyword **LIST** at the very place where you would specify a DB2 object (or multiple DB2 objects) after a keyword like TABLESPACE or INDEXSPACE.

   In our example, within the RECOVER utility control statement, we specify the list RECLIST after the keyword LIST - rather than all four table spaces, each after the keyword TABLESPACE.

***What happens at execution time***
First, DB2 reads the LISTDEF control statement, hence DB2 knows from the algorithm how to generate the list RECLIST. Having read the RECOVER statement, DB2 generates this list, based on the current definitions in the catalog. Then, DB2 processes the RECOVER statement as if the items of the generated list had been specified explicitly in the RECOVER statement.

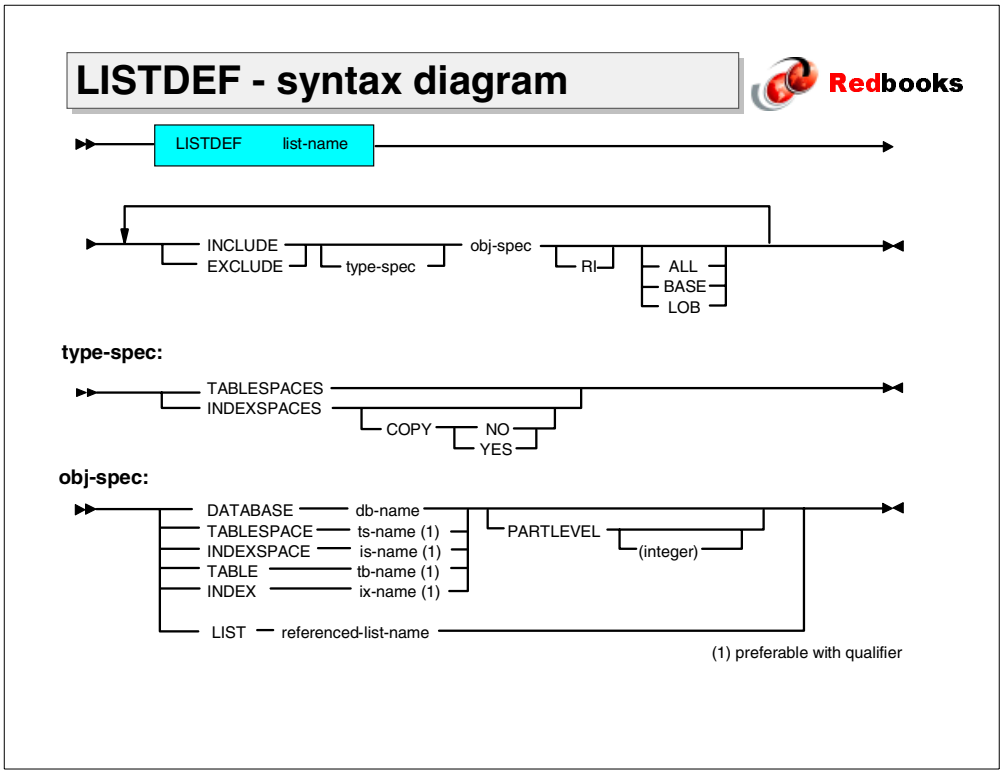
***Notes***
With this new LISTDEF/LIST feature, the example shown simulates a TABLESPACESET option for the RECOVER utility.

RECOVER could already process explicit, static lists in DB2 V6, now it can process dynamic lists.

***The benefit***
If the number of DB2 objects varies, you do not have to adapt your jobs when using the LISTDEF/LIST construct. This is especially helpful in a recovery situation.

***Sample utility job output for LISTDEF/LIST***
We attach part of the recover utility output, run in V7, showing how DB2 resolves the list RECLIST according to the provided LISTDEF definition.

```
OUTPUT START FOR UTILITY, UTILID = TEMP
 LISTDEF RECLIST INCLUDE TABLESPACE DBY.T* RI
LISTDEF STATEMENT PROCESSED SUCCESSFULLY
 RECOVER LIST RECLIST TOLOGPOINT X'000000A8EA76'
- RECOVER TABLESPACE DBY.TS3    START
- NO GOOD FULL IMAGE COPY DATA SET FOR RECOVERY OF TABLESPACE DBY.TS3
- RECOVER TABLESPACE DBY.TS4    START
- NO GOOD FULL IMAGE COPY DATA SET FOR RECOVERY OF TABLESPACE DBY.TS4
- RECOVER TABLESPACE DBX.PTS1    START
- NO GOOD FULL IMAGE COPY DATA SET FOR RECOVERY OF TABLESPACE DBX.PTS1
- RECOVER TABLESPACE DBX.TS2    START
- NO GOOD FULL IMAGE COPY DATA SET FOR RECOVERY OF TABLESPACE DBX.TS2
- INDEX PAOLOR3.IX3 IS IN REBUILD PENDING
- ALL INDEXES OF DBY.TS3  ARE IN REBUILD PENDING
- INDEX PAOLOR3.IX4 IS IN REBUILD PENDING
- ALL INDEXES OF DBY.TS4  ARE IN REBUILD PENDING
- INDEX PAOLOR3.IXP1 IS IN REBUILD PENDING
- ALL INDEXES OF DBX.PTS1  ARE IN REBUILD PENDING
- INDEX PAOLOR3.IX2 IS IN REBUILD PENDING
- ALL INDEXES OF DBX.TS2  ARE IN REBUILD PENDING
- RECOVER UTILITY LOG APPLY RANGE IS RBA 0000009FDDAF TO RBA 000000A00CA1
- RECOVER UTILITY LOG APPLY RANGE IS RBA 000000A03898 TO RBA 000000A8AC36
- FOLLOWING TABLESPACES RECOVERED TO A CONSISTENT POINT
 DBX.PTS1
 DBX.TS2
 DBY.TS3
 DBY.TS4
RECOVERY COMPLETE, ELAPSED TIME=00:00:09
UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=4
```




### 5.3.2.2 LISTDEF - syntax diagram

Before proceeding to more complex LISTDEF examples, we must know the syntax diagram. The description concentrates on the important parts:

A LISTDEF definition must consist of four basic parts:

1. `LISTDEF list-name`

   The name of the list (up to 18 characters) which is to be used in the utility control statement or in another LISTDEF statement.

2. `INCLUDE` or `EXCLUDE` keyword

   This indicates if the list being created includes objects or excludes them from the list built so far. More details follow.

3. `type-spec`

   The type of objects the list will contain, either table spaces or index spaces, as well as the specification of the output of your list definition.

   The syntax diagram seems to imply that `type-spec` is optional. However:

   - If `obj-spec` starts with DATABASE, the type-spec must be coded.
   - If `obj-spec` starts with TABLESPACE or TABLE, the default type-spec is TABLESPACES.
   - If `obj-spec` starts with INDEXSPACE or INDEX, the default type-spec is INDEXSPACES.
   - If `obj-spec` stats with LIST there are two cases:
     - If you have not specified the `type-spec` (neither `TABLESPACES` nor `INDEXSPACES`), all objects of the list `referenced-list-name` will be included.

- If you have specified the type-spec (either TABLESPACES or INDEXSPACES), only the objects of this type are extracted from the list referenced-list-name.

4. obj-spec

   This is your input for the search - the object(s) to be used in the initial catalog lookup. This initial search for objects can begin with a database, a table space, an index space, a table, an index, or even with another list.
   (But the resulting list will only contain table spaces, index spaces, or both.)

   Suppose that your obj-spec is: TABLESPACE tsname. This tsname can be an explicit, qualified table space name or a pattern-matching expression, for example, DB%.?TS*, where:

   **%** or ***** Denotes zero to seven arbitrary alphanumeric characters (% as in SQL LIKE)

   **?** Denotes one single arbitrary alphanumeric character; but:

   _ Underscore is a valid character in index and table names, but it is also a wildcard for a single arbitrary character in database, table space, and index space names.

   Qualifications:

   Not supported are patterns consisting of % (or * ) only. For example, the following specifications are invalid: DATABASE %,TABLESPACE *.*

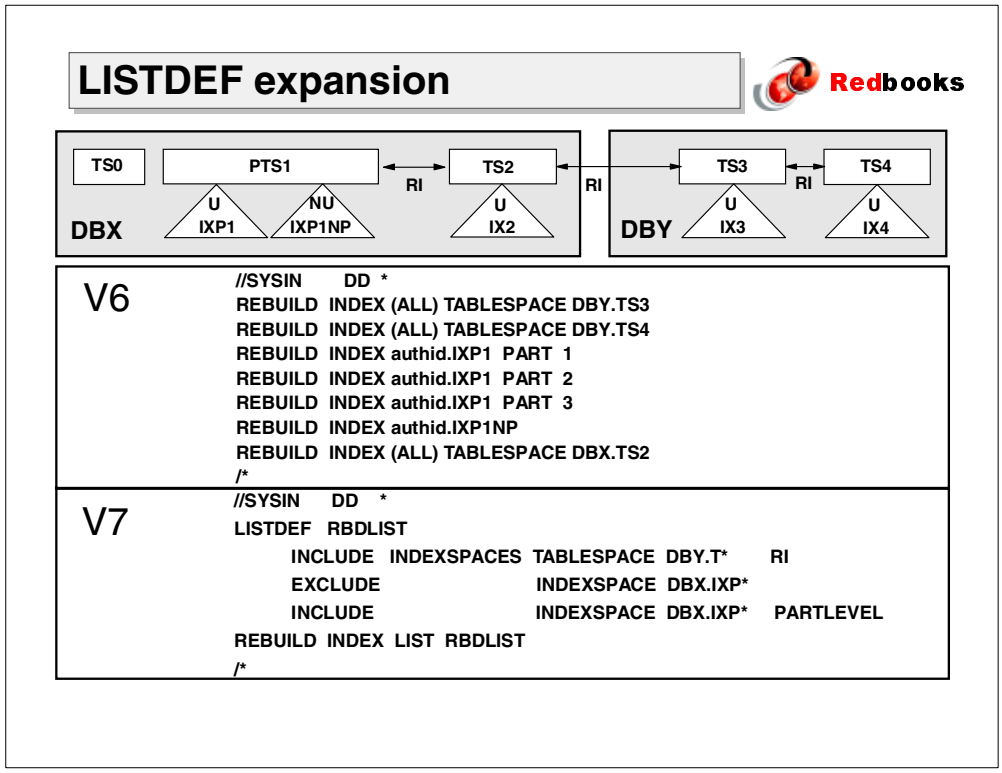
   PARTLEVEL or PARTLEVEL(integer). Specification of the partition granularity for partitioned table spaces or partitioned index spaces that are contained in the list. PARTLEVEL is ignored for non-partitioned objects. Suppose that INCLUDE was specified, then you have to distinguish: If PARTLEVEL is specified without (integer), the resulting list will contain one entry for each partition. If (integer) is specified, an entry for that partition integer will be added to the list. For EXCLUDE, analog rules apply.

   The PARTLEVEL keyword can be specified after the type-spec, too. This may seem more intuitive, as it describes the output list of an INCLUDE/EXCLUDE clause rather than the input pattern you provide for the initial lookup.

   Remark to prevent a possible misunderstanding: logical partitions are not processed by the PARTLEVEL option.

In addition to these four basic parts, optional keywords can be added, indicating what relationships to follow in order to add or remove related objects to the list. Two types of relationships can be used: Objects can be referentially related or auxiliary related. In order to let DB2 add or remove related objects to the list, you can specify:

- RI - related by referential constraint
- BASE - add non-LOB objects and remove LOB objects
- LOB - add LOB objects and remove non-LOB objects
- ALL - add both, LOB and BASE objects




### 5.3.2.3 LISTDEF expansion

Here, we will look at a LISTDEF expansion sample scenario.

***Why deal with the expansion algorithm?***

When DB2 expands your list definition, that is, generates a list of objects according to your LISTDEF statement, the sequence of steps DB2 uses influences the result. You must know this sequence in order to code more complicated list definitions correctly.

***The sample scenario - and the task to perform***

Five table spaces and their unique indexes reside in the two databases DBX and DBY. The partitioned table space PTS1 has an additional non-partitioning index IXP1NP. After the previous recovery of a table space set to a prior point in time, since we have not recovered the indexes to the same point in time, we have to rebuild all indexes on all tables in all tables spaces of this table space set. Normally, if no COPY enabled indexes are involved, the first INCLUDE clause of the LISTDEF statement on the foil is all you need to fulfill this task.

Therefore, the complete recovery job would be:

```
LISTDEF RECLIST INCLUDE TABLESPACE DBY.TS% RI
LISTDEF RBDLIST INCLUDE INDEXSPACES TABLESPACE DBY.T% RI
RECOVER LIST RECLIST TOLOGPOINT X'xxxxxxxxxxxx'
REBUILD INDEX LIST RBDLIST
```

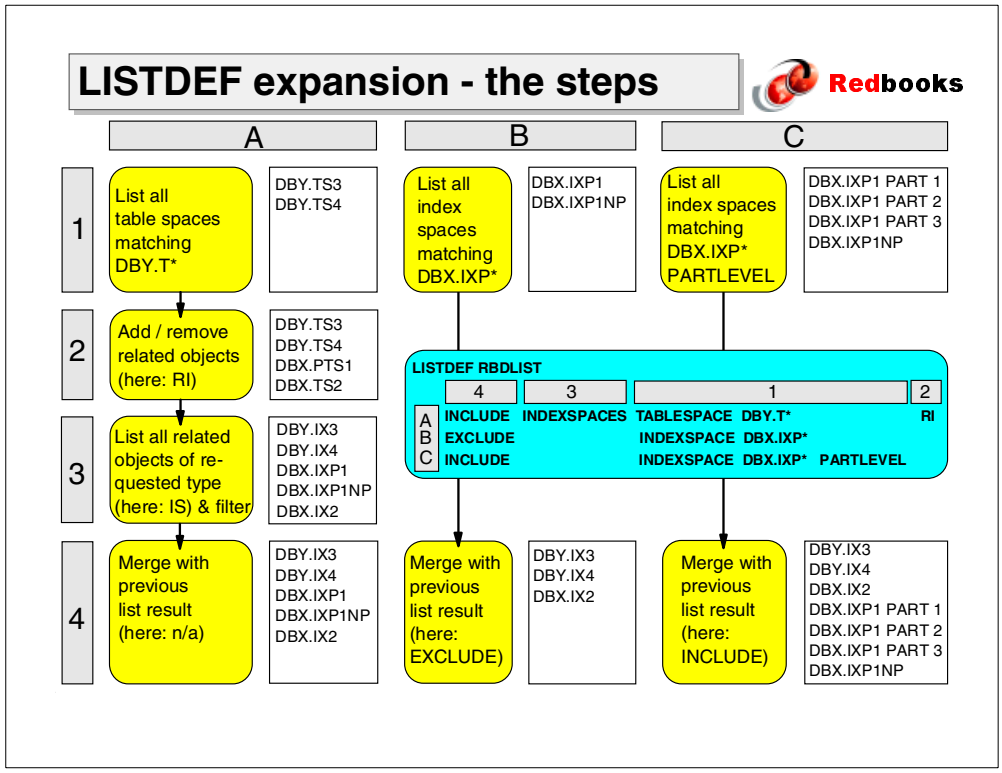
***A more complicated task - for learning purposes***
Let us additionally assume that you want to rebuild all your partitioned indexes per partition, if they reside in the database DBX. This is for the sake of demonstrating:

- How LISTDEF expansion works
- How you can deal with the LISTDEF restriction, that RI and PARTLEVEL cannot be combined in one INCLUDE / EXCLUDE clause

***The utility statements in V6***
As REBUILD supports only lists of index spaces belonging to the same table space, you need several REBUILD statements: It is your responsibility to specify a separate REBUILD statement for all table spaces of this table space set and for all partitions of all partitioned indexes in DBX in order to have parallel processing and decrease of the sort requirements.

***The V7 alternative***
If all partitioned index space names in DBX start with ‘IXP’, then the foil shows an alternative way, possible with V7. Again, the advantage is that you do not have to change the DB2 V7 job, if table spaces or indexes are dropped or added - as long as you adhere to established naming conventions.

On the next foil, we will see how this LISTDEF definition is resolved.




### 5.3.2.4 LISTDEF expansion - the steps

For reading convenience, the LISTDEF statement is annotated: It consists of three INCLUDE / EXCLUDE clauses, A, B, and C, which DB2 processes in this given sequence - and of course, this sequence influences the content of the final list!

Within each of these three clauses, the list definition is expanded in four steps:

1. An initial catalog lookup is performed to find and list the explicitly specified object or the objects which match the specified pattern.

   In our example, in the first INCLUDE clause, table spaces are searched which match the pattern DBY.T*. The two matching tables spaces, DBY.TS3 and DBY.TS4, are listed on the foil at the right side of step 1.

2. Related objects are added or removed depending on the presence of the RI, BASE, LOB, or ALL keywords. Two types of relationships are supported, referential and auxiliary relationships. More specifically, if you specify:

   - `RI`, the TABLESPACESET process is invoked and referentially related objects are added to the list.

     As a result, all - by their tables - referentially connected table spaces are included in the list.

     In our example, the foil shows these four table spaces.

   - `LOB` or `ALL`, auxiliary related LOB objects are added to the list.
   - `BASE` or `ALL`, auxiliary related base objects are added to the list.
   - `LOB`, base objects are excluded from the list.
   - `BASE`, LOB objects are excluded from the list.

3. This step consists of two sub-steps:

   a. All related objects of the requested type are searched and added to the list.

      This step can be skipped if the list built so far already consists of objects of the requested type, either table spaces or index spaces. Otherwise, we must distinguish two cases:

      - If you want a list of table spaces, that is, the type specification is `TABLESPACES`, but the initial catalog lookup has been done with another type of object, for example, with index spaces or indexes, then related table spaces are added to the list. Obviously, those table spaces are related, in which the base tables reside of the indexes or index spaces already in the list.
      - If you want a list of index spaces, that is, the type specification is `INDEXSPACES`, but the initial catalog lookup has been done with another type of object, for example, with table spaces or tables, then the related index spaces are added to the list.

        In our example, in the first INCLUDE clause, the four table spaces have five index spaces all together. These index spaces are added to the list.

   b. A filtering process takes place:

      If the user specifies `INCLUDE TABLESPACES`, all objects which are not table spaces are removed from the list. Analog rules apply for `INCLUDE INDEXSPACES`, `EXCLUDE TABLESPACES`, and `EXCLUDE INDEXSPACES`.

      The specification of COPY YES or NO is also taken into account in this step.

      In our example, in the first INCLUDE clause, INCLUDE INDEXSPACES has been specified. Therefore, all table spaces are removed from the list. The list now contains five index spaces only, as you can see on the foil.

4. If the keyword INCLUDE is specified, the objects of the resulting list are added to the list derived from the previous INCLUDE or EXCLUDE clauses, if they are not in the list already.

   In other words, an INCLUDE of an object already in the list is ignored; the list contains no duplicates.

   If the keyword EXCLUDE is specified, the objects of the resulting list are removed from the list derived from the previous INCLUDE or EXCLUDE clause. Obviously, an EXCLUDE of an object not in the list is ignored.

   In our example, for the first INCLUDE clause (A), there is no previous list to merge with. Therefore, this step is skipped.

Up to now, all four steps are explained for our sample first INCLUDE clause (A). Next, the EXCLUDE clause (B) and then the second INCLUDE clause (C) are processed. Some notes on these two clauses:

- Step 1 is similar to the already explained step 1 of the first INCLUDE clause (A)
- Step 2 is skipped as no relationship type was specified, neither RI, ALL, BASE, nor LOB
- Step 3 is skipped as `spec-type` INDEXSPACES is assumed if `obj-type` is INDEXSPACE, therefore, there is no need to look for related objects, nor to filter out objects of another type.

- In step 4 for the EXCLUDE clause (B), two objects are removed from the list generated by the first INCLUDE clause (A).
- In step 4 for the second INCLUDE clause (C), four objects are added to the list generated after the EXCLUDE clause (B).

***Disclaimer***

The purpose of the last foil is to conceptually present the resolution of a LISTDEF statement in order to help you to define your LISTDEF statements properly. The purpose is not to precisely document DB2's implementation of the sequence of the expansion steps which is in fact different in order to be generally applicable, not only for our example. DB2 may perform step 2 after step 3, for example, if you specify INCLUDE TABLESPACES INDEX authid.ix RI. Furthermore the consideration of the PARTLEVEL keyword may be postponed to the end, that is, after step 3.

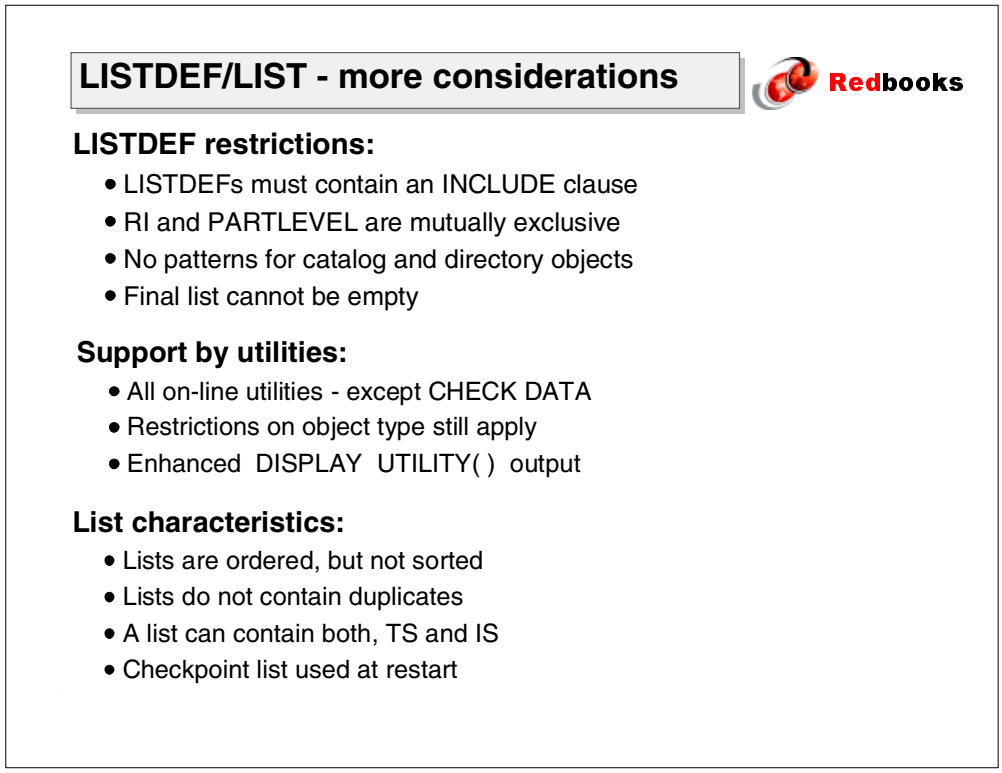
***Sample utility job output for expansion***

Part of a utility output ia attached here to show the expansion of the LISTDEF definition on the foil.

Note that in this job, the OPTIONS(PREVIEW) statements has already been used:

- The original INCLUDE and EXCLUDE clauses are not repeated totally during the expansion process of the individual INCLUDE / ECLUDE clauses - just the `INCLUDE / EXCLUDE` keyword and the `obj-spec` (without the `PARTLEVEL` keyword) is repeated.
- PREVIEW generates an expanded list with INCLUDE clauses of single objects. This list starts in the line with the comment "-- 00000007 OBJECTS" and comprises seven INCLUDE clauses. This part of the output is itself a valid LISTDEF definition, which you can extract and use. The advantage is that the time-consuming expansion process can be avoided; the disadvantage is that the list is not dynamic any more.

```
- OUTPUT START FOR UTILITY, UTILID = TEMP
-  OPTIONS PREVIEW
- PROCESSING CONTROL STATEMENTS IN PREVIEW MODE
- OPTIONS STATEMENT PROCESSED SUCCESSFULLY
-  LISTDEF RBDLIST INCLUDE INDEXSPACES TABLESPACE DBY.T* RI EXCLUDE
INDEXSPACE DBX.IXP* INCLUDE INDEXSPACE DBX.IXP* PARTLEVEL
- LISTDEF STATEMENT PROCESSED SUCCESSFULLY
  - EXPANDING LISTDEF RBDLIST
  -   PROCESSING INCLUDE CLAUSE TABLESPACE DBY.T*   <- A: INCLUDE
  -     CLAUSE IDENTIFIES 5 OBJECTS                 <- after step 3
  -   PROCESSING EXCLUDE CLAUSE INDEXSPACE DBX.IXP* <- B: EXCLUDE
  -     CLAUSE IDENTIFIES 2 OBJECTS                 <- after step 3
  -   PROCESSING INCLUDE CLAUSE INDEXSPACE DBX.IXP* <- C: INCLUDE
  -     CLAUSE IDENTIFIES 4 OBJECTS                 <- after step 3
  - LISTDEF RBDLIST CONTAINS 7 OBJECTS              <- final list
- LISTDEF RBDLIST EXPANDS TO THE FOLLOWING OBJECTS:
LISTDEF RBDLIST -- 00000007 OBJECTS
  INCLUDE INDEXSPACE DBY.IX3
  INCLUDE INDEXSPACE DBY.IX4
  INCLUDE INDEXSPACE DBX.IXP1NP
  INCLUDE INDEXSPACE DBX.IX2
  INCLUDE INDEXSPACE DBX.IXP1 PARTLEVEL(00001)
  INCLUDE INDEXSPACE DBX.IXP1 PARTLEVEL(00002)
  INCLUDE INDEXSPACE DBX.IXP1 PARTLEVEL(00003)
DSNUGUTC -  REBUILD INDEX LIST RBDLIST
DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DBY.IX3
DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DBY.IX4
DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DBX.IXP1NP
DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DBX.IXP1
DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DBX.IXP1
DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DBX.IXP1
DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DBX.IX2
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```




### 5.3.2.5 LISTDEF/LIST - more considerations

Here we discuss several user considerations.

***INCLUDE clause must be present***

Normally an INCLUDE clause comes first to indicate the objects to select. If the statement contains a nested list, such as:

```
LISTDEF list2 INCLUDE TABLESPACE LIST list1
```

If list1 contains any index spaces, all table spaces related to the index spaces will be included in list2. If list1 contains table spaces, they are included in list2.

***RI and PARTLEVEL are mutually exclusive***

RI and PARTLEVEL cannot be specified in the same INCLUDE / EXCLUDE clause, which is logical when you specify PARTLEVEL(n). Although it would be logically possible to combine them, if no integer value is specified after PARTLEVEL, the restriction nevertheless applies in this case, too.

RI and PARTLEVEL can be combined in the same list though, as shown in the previous example.

***No patterns for catalog and directory objects***

Catalog and directory objects are never included in a list by pattern matching, they must be specified individually by their fully qualified names.

***Final list cannot be empty***
Individual INCLUDE and/or EXCLUDE clauses may result in zero objects, but the final result list must contain at least one object or an error will be returned.

How many objects a list can comprise? Lists themselves are almost unlimited, but different utilities might impose different internal limits.

***Supporting utilities***
All online utilities, including the new UNLOAD utility support lists, except CHECK DATA (support has not been extended to exception tables).

***Utility restrictions on object type still apply***
For example, you cannot process a list of index spaces by
REORG TABLESPACE, neither a list of table spaces by REBUILD.

***Enhanced DISPLAY UTILITY() output***
DSNU100I for stopped utilities and DSNU105I for active utilities include two additional rows:

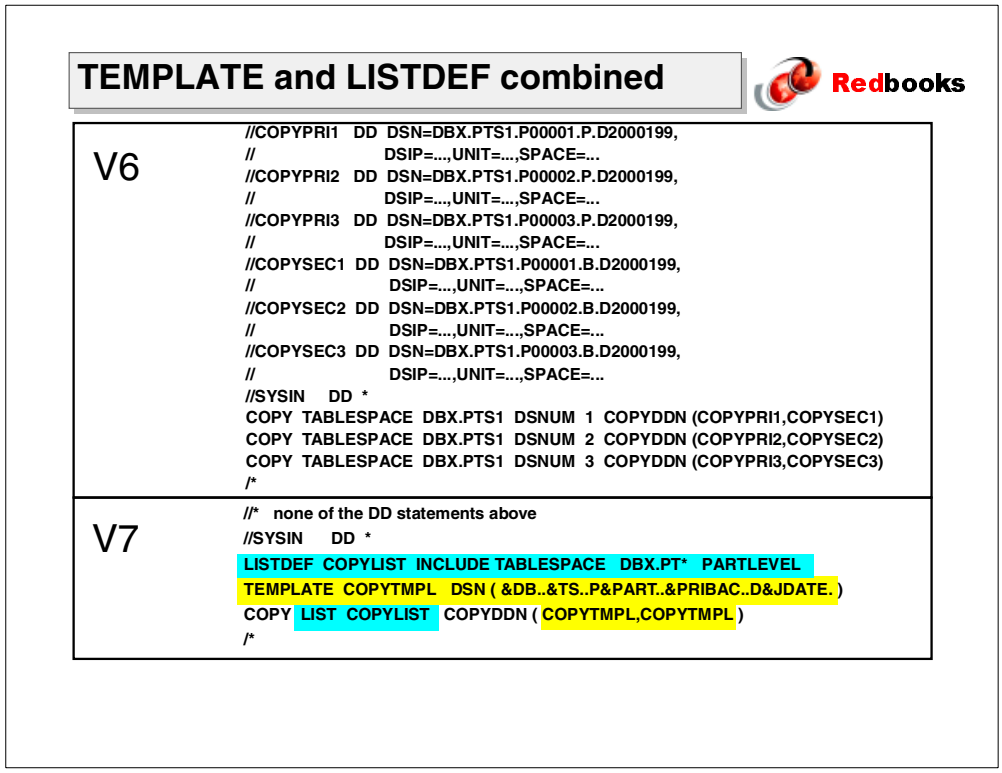
```
NUMBER OF OBJECTS IN LIST = xxx
LAST OBJECT STARTED = yyy
```

Following are some notes to help avoid possible misconceptions:

***Lists are ordered, but not sorted***
Of course, the items in the list are in a certain sequence, but the user cannot request a sorted list.

***A list can contain both table spaces and index spaces***
Both are included if it is composed of several INCLUDE clauses or of other lists. This way you can COPY and RECOVER both, table spaces and COPY enabled index spaces, with one list.

***Checkpoint list on restart***
When you restart a utility job, DB2 picks up the checkpoint list from SYSUTILX rather than reevaluating the list dynamically. This is done for every new execution.




### 5.3.3 TEMPLATE and LISTDEF combined

We now look at the combination of using both TEMPLATE and LISTDEF/LIST.

***Reason for combining LISTDEF/LIST and TEMPLATE***

If you want that a utility processes a dynamic list of DB2 objects, that is, a list that may increase or decrease over time, you can use LISTDEF and LIST. As many online utilities require data set allocations for each object they process, the number (and the parameters) of these allocations must be dynamic, too. That cannot be done with ordinary DD statements, but that is exactly what templates support: A dynamic number of data set allocations, fitting to the object or to the list of objects processed by a utility.

In summary, the usage of LISTDEF/LIST often requires the usage of TEMPLATE.

***Example for combined use of LISTDEF/LIST and TEMPLATE***

We want to generate primary and backup copies per partition for all partitioned table spaces in the database DBX whose names start with PT. This leads to the LISTDEF statement on the foil. Not knowing how many table spaces matching that list definition exist, and not knowing how many partitions these table spaces have, we can use the template on the foil for dynamically allocating the needed copy data sets. Please note, as the template includes the variable &PRIBAC., this template can be used in place of both DD names, the one for the primary copy (SYSCOPY) and the one for the backup copy.

***What happens at execution time***

At job execution time, DB2 first processes the LISTDEF statement, hence it knows the algorithm how to generate the list COPYLIST. Then the template definition is read and stored. When DB2 has read the COPY statement, the list COPYLIST is generated and DB2 tries to copy each item of this list. Now DB2 knows for each item the dbname, the tsname and the partition. Therefore, these values can now be assigned to the template variables in the template stored previously.

***Sample utility job output using LISTDEF/LIST and TEMPLATE***

Attached is part of the copy utility output, run in V7, showing how DB2 resolves the list COPYLIST and the template COPYTMPL:

```
- OUTPUT START FOR UTILITY, UTILID = PAOLOR3.TEMP
-  LISTDEF COPYLIST INCLUDE TABLESPACE DBX.PT* PARTLEVEL
- LISTDEF STATEMENT PROCESSED SUCCESSFULLY
-  TEMPLATE COPYTMPL DSN(&DB..&TS..P&PART..&PRIBAC..D&JDATE.)
- TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
-  COPY LIST COPYLIST COPYDDN(COPYTMPL,COPYTMPL)
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00001
   DSN=DBX.PTS1.P00001.P.D2000199
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00002
   DSN=DBX.PTS1.P00001.B.D2000199
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 1
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 33.66
 PERCENT OF CHANGED PAGES =  0.00
 ELAPSED TIME=00:00:00
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00003
   DSN=DBX.PTS1.P00002.P.D2000199
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00004
   DSN=DBX.PTS1.P00002.B.D2000199
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 2
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 33.66
 PERCENT OF CHANGED PAGES =  0.00
 ELAPSED TIME=00:00:00
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00005
   DSN=DBX.PTS1.P00003.P.D2000199
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00006
   DSN=DBX.PTS1.P00003.B.D2000199
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 3
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 32.33
 PERCENT OF CHANGED PAGES =  0.00
 ELAPSED TIME=00:00:00
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 1
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 2
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 3
- UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

***Note - time of dynamic allocation***
In DB2 V6, allocations are done at the beginning of a job step. Now, when using lists and templates, the output shows that dynamic allocations are only done when the utility execution proceeds to the very object (of the list) for which the allocations are needed.

***Advantages of the combination***
Here are some advantages:

- There are fewer utility control statements, and fewer DD statements to code.
- The combination of LISTDEF/LIST and TEMPLATE facilitates the development and maintenance of your utility jobs.

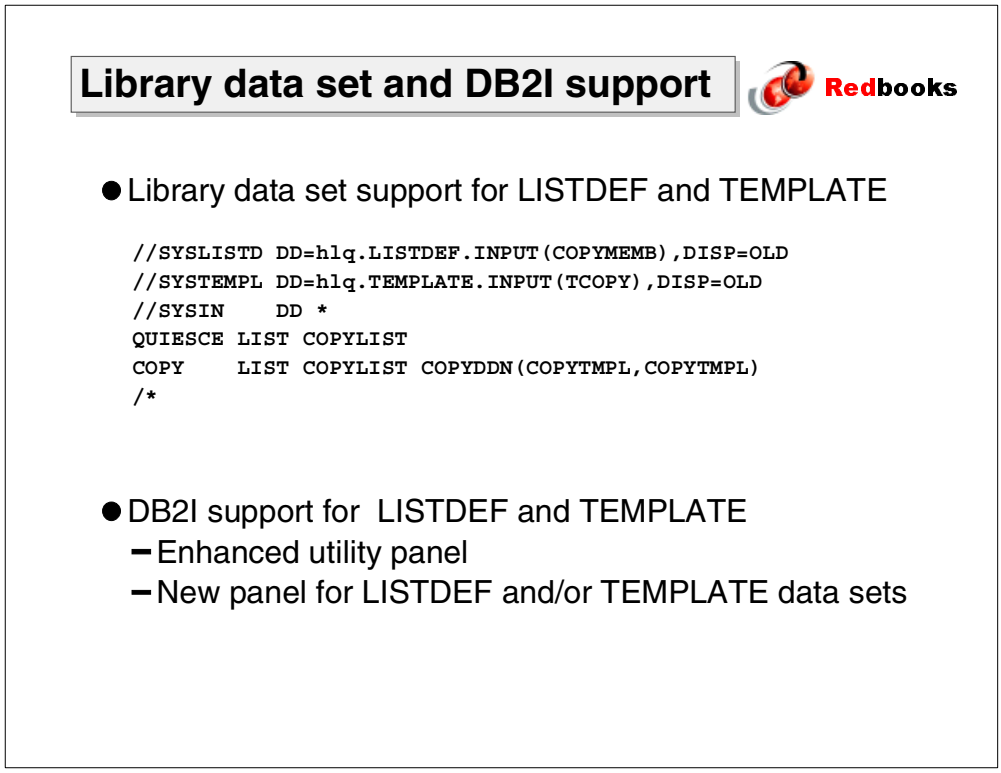
***Maximizing performance***
If several executions of CHECK, REBUILD, RUNSTATS utilities are invoked for multiple indexes for a list of objects performance can be improved. For instance, in the case of 2 table spaces with 3 indexes each, for the LISTDEF:

```
LISTDEF myindexes INCLUDE INDEXSPACES TABLESPACE db1.ts1

                  INCLUDE INDEXSPACES db2.ts2
```

DB2 generates CHECK statements for all indexes of the same table space as follows:

```
CHECK INDEX db1.ix1,db1.ix2,db1ix3

CHECK INDEX db21.ix1,db2.ix2,db2ix3
```

Thus the table space scans are reduced to two instead of a possible six.




### 5.3.4 Library data set and DB2I support

Instead of using SYSIN for providing your LISTDEF and TEMPLATE statements, you can store these statements in library data sets. When using these data sets in your utility jobs, the default DD names are respectively SYSLISTD and SYSTEMPL.

More precisely, when using lists of DB2 objects in your utility statements, for example, COPYLIST in QUIESCE and COPY, without a LISTDEF statement for COPYLIST in SYSIN, then these LISTDEF statements should reside in the data set allocated with the DD name SYSLISTD, for example, in the member COPYMEMB of the data set hlq.LISTDEF.INPUT with *recfm* FB and *recl* 80. In this example, the content of COPYMEMB is:

```
LISTDEF COPYLIST INCLUDE TABLESPACES PARTLEVEL TABLESPACE DBX.PTS*
```

In a similar way, when using templates in your utility statement(s), for example, COPYTMPL in the COPY statement, without a TEMPLATE statement for COPYTMPL in SYSIN, then these template statements should reside in the data set allocated with the DD name SYSTEMPL, for example, in the member TCOPY of the data set with hlq.TEMPLATE.INPUT with *recfm* FB and *recl* 80. In this example, the content of TCOPY is:

```
TEMPLATE COPYTMPL DSN(&DB..&TS..P&PART..&PB..D&JDATE..M&MINUTE.)
```

**Notes:**

- You can store multiple template definitions in one member of the designated partitioned data set. Alternatively, you can concatenate members in a job.
- The example shows that both the LISTDEF definition and the TEMPLATE definition are used twice.

**Benefit:** Using library data sets rather than SYSIN, the list and template definitions are more manageable.

***DB2I support for LISTDEF and TEMPLATE***

The sample job on the last foil can also be generated with DB2I, as DB2I was enhanced to support both, list definitions and template definitions, in library data sets. On the utility panel of DB2I you can specify, whether you want to use library data sets for list definitions, for template definitions, or for both:

```
 DB2 UTILITIES                    SSID: DB2A
===>

Select from the following:

 1 FUNCTION ===> EDITJCL           (SUBMIT job, EDITJCL, DISPLAY, TERMINATE)
 2 JOB ID   ===> userid.cop        (A unique job identifier string)
 3 UTILITY  ===> COPY              (CHECK DATA, CHECK INDEX, CHECK LOB,
                                    COPY, DIAGNOSE, LOAD, MERGE, MODIFY,
                                    QUIESCE, REBUILD, RECOVER, REORG INDEX,
                                    REORG LOB, REORG TABLESPACE, REPORT,
                                    REPAIR, RUNSTATS, STOSPACE, UNLOAD)
 4 STATEMENT DATA SET ===> SYSIN.INPUT(COPYSTMT)

Specify restart or preview option, otherwise enter NO.

 5 RESTART  ===> NO                (NO, CURRENT, PHASE or PREVIEW)

 6 LISTDEF? (YES!NO) ===> Y   TEMPLATE? (YES!NO) ===> Y

 * The data set names panel will be displayed when required by a utility.

PRESS:  ENTER to process     END to exit    HELP for more information
```

Once you have specified at least one Y(ES) on line 6, you will get a new DB2I panel, asking for the names of the data sets for list definitions, template definitions, or both:

```
 CONTROL STATEMENT DATA SET NAMES            SSID:
===>

Enter the data set name for the LISTDEF data set (SYSLISTD DD):
 1 LISTDEF DSN  ===> userid.LISTDEF.INPUT(COPYMEMB)
    (OPTIONAL)

Enter the data set name for the TEMPLATE data set (SYSTEMPL DD):
 2 TEMPLATE DSN ===> userid.TEMPLATE.INPUT(TCOPY)
    (OPTIONAL)
```

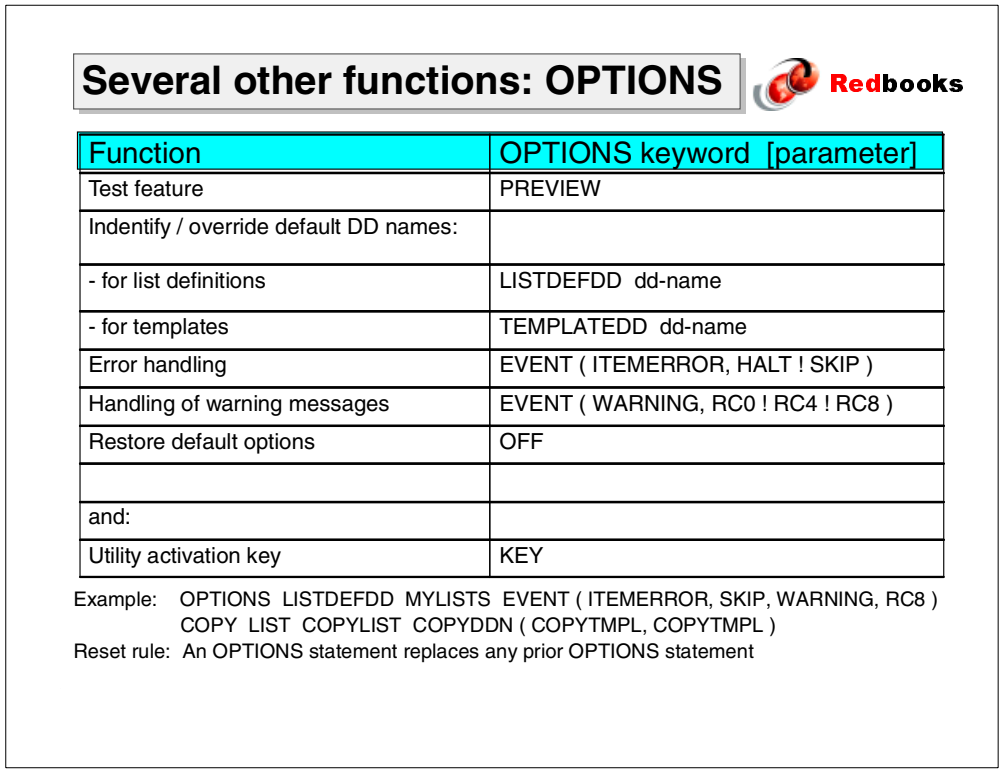
**Note:** If you specify a YES or Y in the field TEMPLATE on the DB2I utility panel, DB2I still presents the data set name panel, as you can still have DD statements in your job. But the input on this panel is optional.

***Sample utility job output for data set support***

Here, we report part of the job output from the QUIESCE and COPY utilities outlined on the foil - the job was generated via DB2I.

The data set names for list and template definitions do not appear in this output: They are the same as in the LISTDEF and TEMPLATE definitions that have been provided via SYSIN:

```
- OUTPUT START FOR UTILITY, UTILID = PAOLOR3.COP
-  COPY LIST COPYLIST COPYDDN(COPYTMPL,COPYTMPL)
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00001
   DSN=DBX.PTS1.P00001.P.D2000180.M43
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00002
   DSN=DBX.PTS1.P00001.B.D2000180.M43
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 1
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 33.66
 PERCENT OF CHANGED PAGES =  0.00
 ELAPSED TIME=00:00:00
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00003
   DSN=DBX.PTS1.P00002.P.D2000180.M43
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00004
   DSN=DBX.PTS1.P00002.B.D2000180.M43
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 2
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 33.66
 PERCENT OF CHANGED PAGES =  0.00
 ELAPSED TIME=00:00:00
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00005
   DSN=DBX.PTS1.P00003.P.D2000180.M43
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00006
   DSN=DBX.PTS1.P00003.B.D2000180.M43
- DATASET ALLOCATED.  TEMPLATE=COPYTMPL
   DDNAME=SYS00006
   DSN=DBX.PTS1.P00003.B.D2000180.M43
- COPY PROCESSED FOR TABLESPACE DBX.PTS1 DSNUM 3
 NUMBER OF PAGES=3
 AVERAGE PERCENT FREE SPACE PER PAGE = 32.33
 PERCENT OF CHANGED PAGES =  0.00
 ELAPSED TIME=00:00:00
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 1
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 2
- DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE DBX.PTS1 DSNUM 3
- UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```




### 5.3.5 Several other functions: OPTIONS

In order to offer a complete and manageable solution when dealing with dynamic utility jobs, another utility control statement, OPTIONS, is provided to complement the functions offered by LISTDEF/LIST and TEMPLATE. OPTIONS provide a general mechanism to enter options that affect utility processing across multiple utility executions in a job step. You can make use of these functions outlined on the foil by specifying the respective keyword in the OPTIONS statement. You code the OPTIONS statement within SYSIN in front of the utility control statement(s) which should act according to the option(s).

If you specify another OPTIONS utility control statement in SYSIN, it entirely replaces any prior OPTIONS statement.

***Test feature: PREVIEW***

Specifying `OPTIONS(PREVIEW)` will expand the list definition(s) and the template definition(s) and report them to you whenever a utility statement, for example COPY, follows. All utility control statements are parsed for syntax errors but normal utility execution will not take place. If the syntax is valid, all LISTDEF lists and TEMPLATE data set names which appear in SYSIN will be expanded and the results printed to the SYSPRINT data set. As stated before, you can extract this expanded list from the SYSPRINT data set and use it inside a utility control statement to avoid the repetition of the expansion process. Lists from the SYSLISTD DD data set and template data set names from the SYSTEMPL DD data set which are referenced by a utility invocation will also be expanded.

For a preview example, see the previous job output showing the LISTDEF expansion.

You can switch off the preview mode by an OPTIONS statement without the PREVIEW keyword, for instance, by `OPTIONS OFF`.

**Note:** The option `OPTIONS(PREVIEW)` is identical to the PREVIEW JCL parameter, which turns on the preview mode for the whole job step. With the new OPTIONS(PREVIEW) utility control statement you have a more detailed granularity: You can switch the preview mode on and off multiple times within a single job step. But when the preview mode has been initiated by the JCL parameter PREVIEW, the OPTIONS control statement cannot switch it off.

***Override default DD names***
When you define your lists or templates in data sets rather than in SYSIN, the default DD names for these data sets are SYSLISTD, respectively SYSTEMPL. If you want to use different DD names, you can specify the new DD names via:

```
OPTIONS LISTDEFDD(other-DD-name) TEMPLATEDD(other-DD-name)
```

***Error handling***
When you work with lists of DB2 objects, your utility might fail with a return code of 8 or higher when processing an item in this list. With the OPTIONS statement, you can specify how DB2 should react:

- To halt on such errors during list processing (default):

  ```
  OPTIONS EVENT(ITEMERROR, HALT)
  ```

- To skip any error and keep going:

  ```
  OPTIONS EVENT(ITEMERROR, SKIP)
  ```

**Note:** Abnormal terminations are not handled by these specifications.

***Handling of warning messages***
You can specify how a final job step return code of 4 should be treated. You can force the return code to become 0 or 8 - or to become 4 again:

```
OPTIONS EVENT(WARNING, RC0)
OPTIONS EVENT(WARNING, RC4)
OPTIONS EVENT(WARNING, RC8)
```

RC4 can be used to override a previous setting to 0 or 8 in the same job step.

***Restore default options***
You can specify that all default options are to be restored with:

```
OPTIONS OFF
```

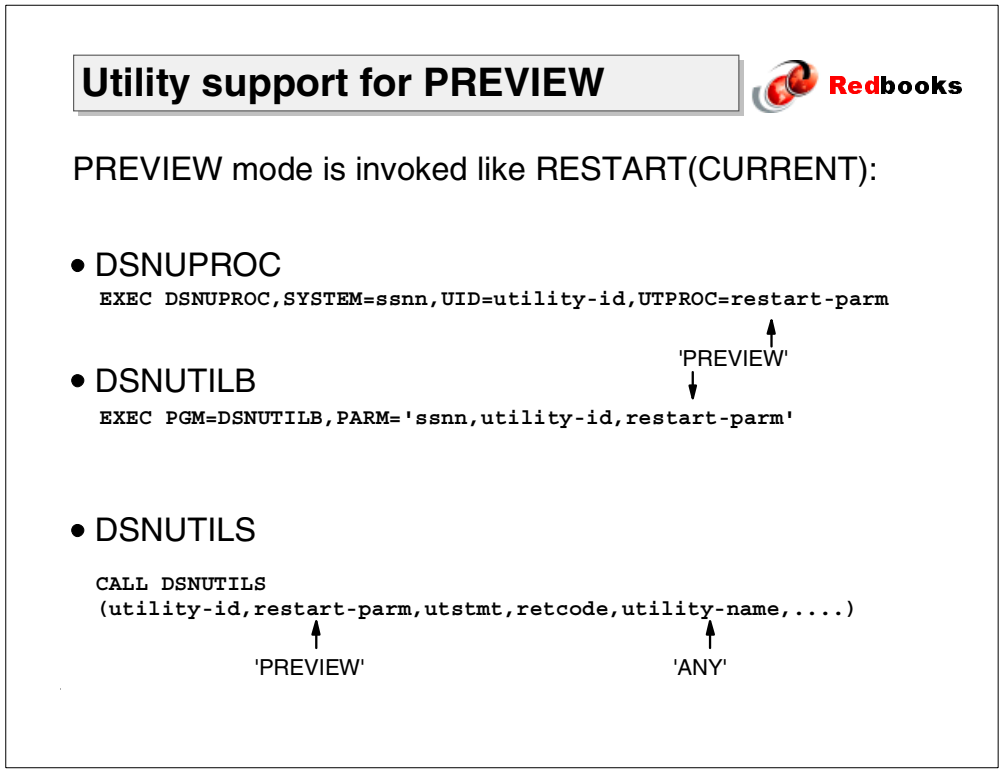
**Note:**

- No other keywords may be specified with the `OPTIONS OFF` setting.
- `OPTIONS OFF`

  is equivalent to:

  ```
  OPTIONS LISTDEFDD SYSLISTD TEMPLATEDD SYSTEMPL
          EVENT (ITEMERROR, HALT, WARNING, RC4)
  ```

  As usual for utility control statements, parentheses are required if a list, in contrast to a single item, can be specified.

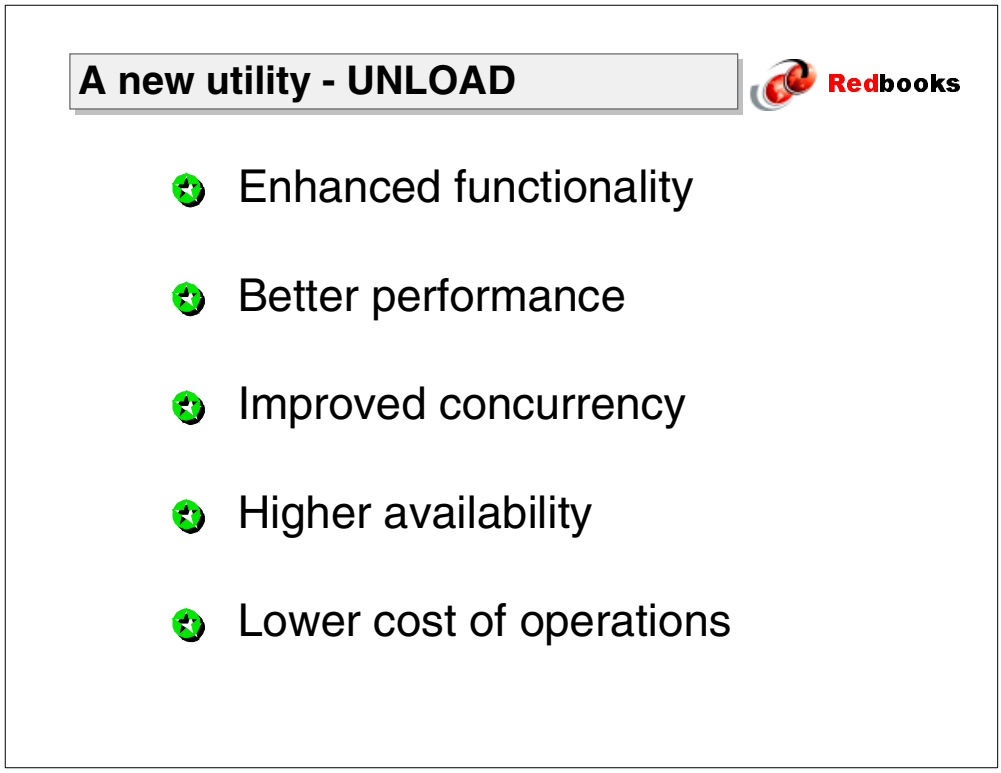



### 5.3.6 Utility support for PREVIEW

OPTIONS(PREVIEW) does not have to be specified in SYSIN. You can select the preview mode when you invoke the utilities via DSNUPROC, DSNUTILS, or directly via DSNUTILB.

***DSNUTILS***

As stated before, ANY suppresses all dynamic allocations done by DSNUTILS. Using PREVIEW, you can save this list before the actual utility is run, and if there is an error, the job output will show the list item where it failed. Since DSNUTILS does not provide for storing the reduced list, you can edit the saved list by removing the already processed items and start again with this reduced list.



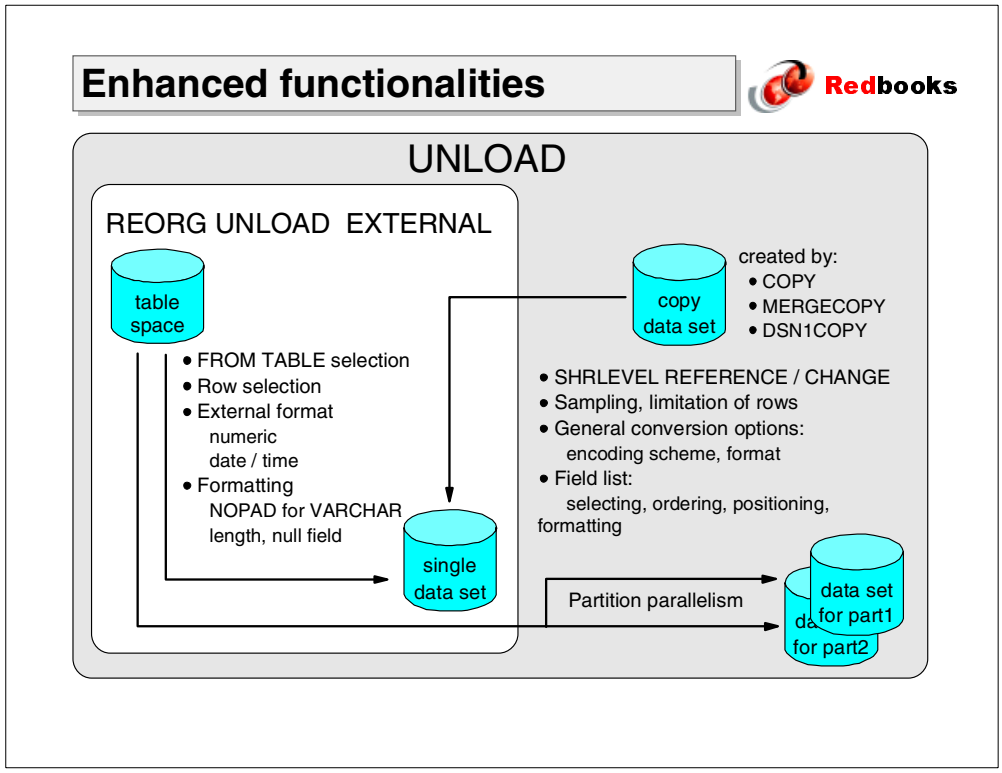

## 5.4 A new utility - UNLOAD

The new UNLOAD utility is essentially a better performing version of the existing REORG UNLOAD EXTERNAL with the same base output format, but more format options. This broad set of additional new capabilities offered by UNLOAD is presented in this section.

***Better performance***
In most cases, the UNLOAD utility is faster than both the DSNTIAUL sample program and REORG UNLOAD EXTERNAL. Performance can be dramatically improved, for example, by partition parallelism, not supported by DSNTIAUL nor REORG.

***Higher availability, improved concurrency***
With the SHRLEVEL CHANGE option, unloading table spaces does not stop your SQL applications, as it used to be the case with REORG UNLOAD EXTERNAL. And the possibility of unloading directly from copy data sets does not touch the data in the table space at all.

***Lower cost of operations***
UNLOAD is easier to use than REORG UNLOAD EXTERNAL; for example, the UNLOAD statement for unloading a subset of the tables of a table space is much more intuitive. Development and maintenance are easier - thus reducing the cost of operations.




### 5.4.1 Enhanced functionalities

This section concentrates on the features offered by UNLOAD which were and are not available with the REORG Unload External utility.

***INPUT: unloading from copy data set(s)***
REORG UNLOAD EXTERNAL and UNLOAD unload data from DB2 table spaces. With UNLOAD, you have an additional option: when the original data in your table space are already heavily accessed and you do not want to degrade the performance of your applications, you can unload your data from copies created by COPY, MERGECOPY, and DSN1COPY.

***OUTPUT: unloading into data sets by partition***
With REORG UNLOAD EXTERNAL and with UNLOAD, data is always unloaded to BSAM sequential data sets in external formats. In contrast to REORG UNLOAD EXTERNAL, with one UNLOAD invocation, you can unload partitions into separate data sets. You have to use templates to activate this capability. When you specify output separate data sets by partition, the unload of these partitions is executed in parallel, provided the system resources for parallelism are available.

***SHRLEVEL REFERENCE or CHANGE***
If you do not want to interrupt your applications, the new SHRLEVEL option improves concurrency when compared to REORG UNLOAD EXTERNAL.

***Sampling, limitation of rows***
By sampling, you can reduce the number of rows to be unloaded. These unloaded data can then be used, for example, to fill a test table. You can also specify a limit of rows to be unloaded, or both.

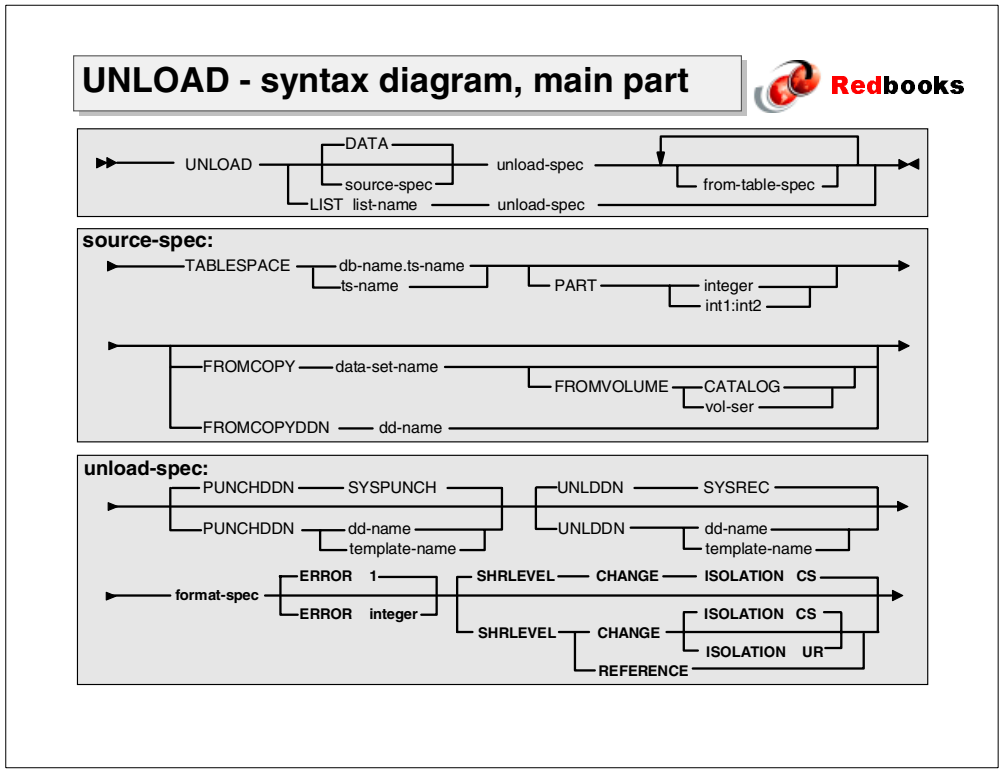
***General conversion options***
You can specify general conversion options. For example, you can

- Convert character data into a different encoding scheme: EBCDIC, ASCII, or UNICODE, and you can convert into a different CCSID
- Specify formatting options, such as NOSUBS, which suppresses character substitution when converting into a different character set with the CCSID conversion.

***Field list: selecting / ordering / positioning / formatting***
Using the REORG utility, you can select specific tables in a table space and specific records within such tables to be unloaded. The new UNLOAD utility offers even more granularity: you can

- Select specific columns of these tables to be unloaded
- Reorder these columns, compared to the order in the table
- Specify the position of these columns in the unload data set
- Convert the data types of these columns.

***Support for exits***
Delimited output and user exits are not supported. Unload does support EDITPROC and FIELDPROC definitions.

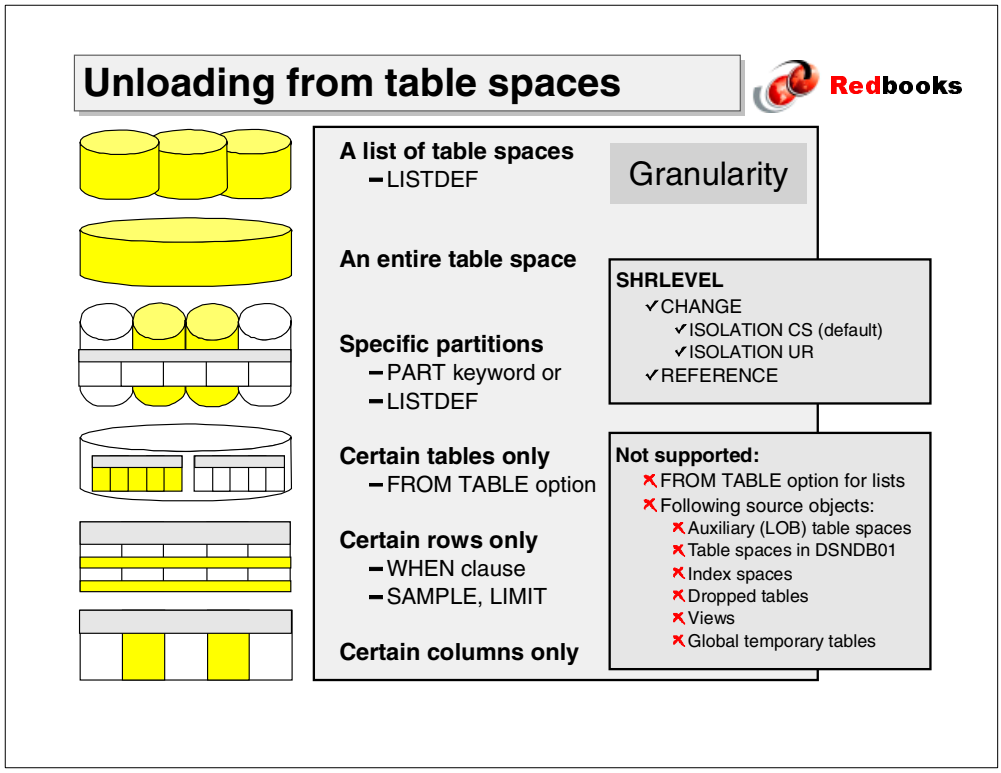



### 5.4.2 UNLOAD - syntax diagram, main part

This syntax diagram is provided for your reference during the following description of the UNLOAD utility. The `from-table-spec` and the `format-spec` portions of the diagram will be discussed later.

In this section, we present the main options in the following groups:

- The `source-spec`, consisting of two sub-groups:
  - Unload from table spaces, including the SHRLEVEL option
  - Unload from copy data sets
- The `unload-spec`, consisting of two sub-groups:
  - The output data sets
  - The formatting of the output




### 5.4.3 Unloading from table spaces

Here, we examine the different alternatives when unloading from table spaces.

***UNLOAD from a list of table spaces***

Unloading from a single table space was, and is still, possible with REORG UNLOAD EXTERNAL, too. The UNLOAD utility additionally supports lists of table spaces, hence you can unload multiple table spaces within a single run of the UNLOAD utility by using the new LISTDEF utility control statement and the LIST keyword in the UNLOAD statement. Details on LISTDEF/LIST and TEMPLATE are found in 5.3, “Dynamic utility jobs” on page 194.

***UNLOAD partitions***

With UNLOAD, partitions are selectable through the PART keyword as with REORG UNLOAD EXTERNAL. You can alternatively select the partitions to be unloaded by using an appropriate list, defined by the new LISTDEF utility control statement. Then UNLOAD offers additional capabilities, most notably parallelism, described under 5.4.5, “Output data sets” on page 241.

***UNLOAD specific tables***

With REORG UNLOAD EXTERNAL, normally all tables of a table space are unloaded. If you want to unload only specific tables, you must define a predicate, which is never true, for a (the) column(s) of the table(s) you want to exclude. When different people develop and maintain such jobs, problems may arise. With UNLOAD, the selection of tables from a table space is more intuitive:

- If you omit the FROM TABLE clause, all tables are unloaded.
- If you specify the FROM TABLE clause, only the tables associated with the given FROM TABLE clause are unloaded.

In other words, if you do not want to unload all tables of a table space, you have to specify all tables you want to unload in the FROM TABLE clause.

***UNLOAD certain rows only***

- Using the WHEN clause within the FROM TABLE clause, you can use a subset of the SQL WHERE capabilities to specify which rows you want to unload. In this regard, UNLOAD offers the same functionality as REORG UNLOAD EXTERNAL.

The additional features offered by UNLOAD are:

- Sampling

  Within the FROM TABLE clause, you can specify the percentage of rows to be unloaded.

  **Notes:**

  If selection conditions are specified by a WHEN clause within the same FROM TABLE clause, sampling is applied to the rows that are qualified by the WHEN selection conditions.

  The sampling is applied by table. If the rows from multiple tables are unloaded with sampling, the referential integrity among the unloaded tables may be lost.

  If you have a self-referencing table, the referential integrity within this unloaded table may be lost if sampling is used.

- Limitation

  You can specify the maximum number of rows to be unloaded from a table.

  **Notes:**

  If the number of unloaded rows reaches the specified limit, an informational message DSNU1201 is issued for the table and no more rows will be unloaded from the table. The process continues to unload qualified rows from other tables.

  Similar to sampling, the referential integrity within the unloaded table(s) may be lost.

***UNLOAD certain columns only***

In contrast to REORG UNLOAD EXTERNAL, UNLOAD allows you to even select the columns you want to unload by using the FROM TABLE clause.

***SHRLEVEL***

It specifies the type of access to the table space(s) or the partition(s) allowed to other processes while the data is being unloaded:

- `SHRLEVEL CHANGE ISOLATION CS`

  The UNLOAD utility assumes CURRENTDATA(NO).

- `SHRLEVEL CHANGE ISOLATION UR`

  Uncommitted rows, if they exist, will be unloaded.

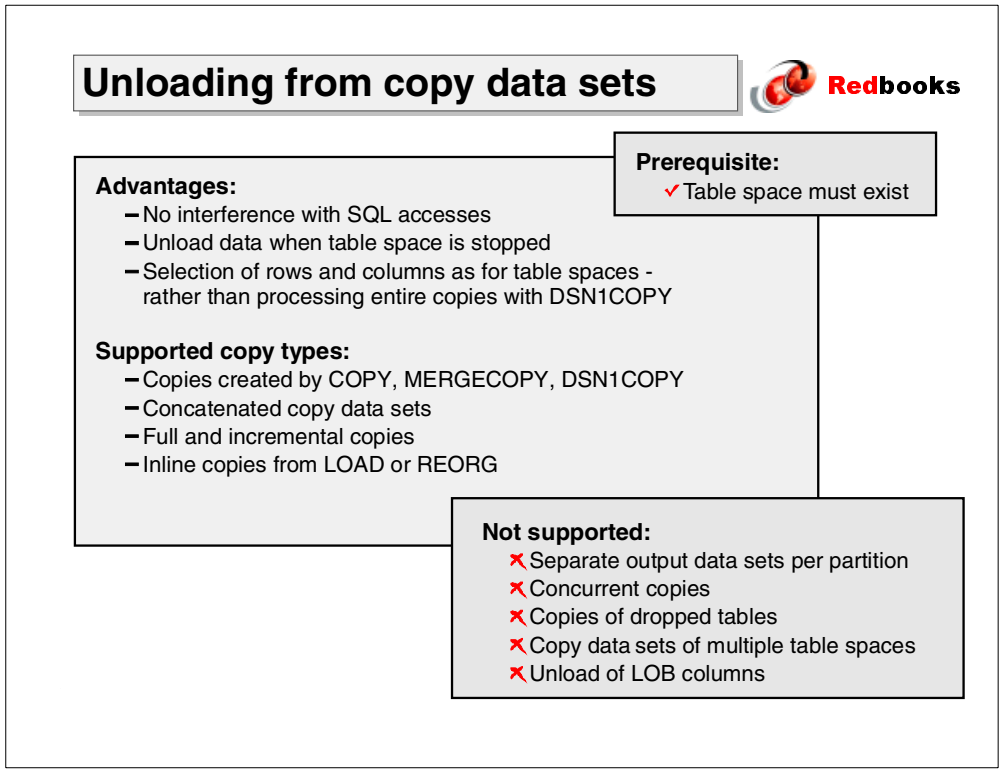
- `SHRLEVEL REFERENCE`

  The UNLOAD utility drains writers on the table space; when data is unloaded from multiple partitions, the drain lock will be obtained for all selected partitions in the UTILINIT phase.

***Restrictions***

If you unload a list of table spaces, you cannot specify the FROM TABLE clause. The unloading of a true subset of tables, rows, or columns is not possible any more.

Also, global temporary tables, both created and declared, are not supported.




### 5.4.4 Unloading from copy data sets

The advantages when unloading from copy data sets are:

- You do not touch the user data in the table space, therefore, you do not degrade the performance of the SQL applications.
- You can unload the data even if the table space is stopped.
- For regularly transferring data into another DB2 subsystem, some use DSN1COPY to refresh a table space in the target DB2 subsystem from a copy of the source DB2 subsystem. Using UNLOAD and LOAD instead, you can transfer exactly these rows and columns you need. Moreover, the latter way is more secure, as the DSN1COPY procedure has to be maintained with care, for example, if columns are added to the source table.

***Prerequisite: Existence of table space***

The UNLOAD utility, in one UNLOAD statement, supports copy data sets as a source for unloading data. The table space must be specified in the TABLESPACE option. This specified table space must still exist when the UNLOAD utility is run, that is, the table space has not been dropped since the copy was taken.

***Concatenated copy data sets***

Normally, if you unload from a single copy data set, the copy data set name can be specified in the FROMCOPY option.

But for partitioned table spaces, individual copy data sets may exist per partition. For unloading, you can concatenate these data sets under one DD name to form a single input data set image. This DD name must then be specified in the FROMCOPYDDN option.

Also, even a non-partitioned table space might be backed by several data sets, and separate copies may exists for these page sets. For unloading from these page sets, you can proceed as in the case of partitions and use the FROMCOPYDDN option.

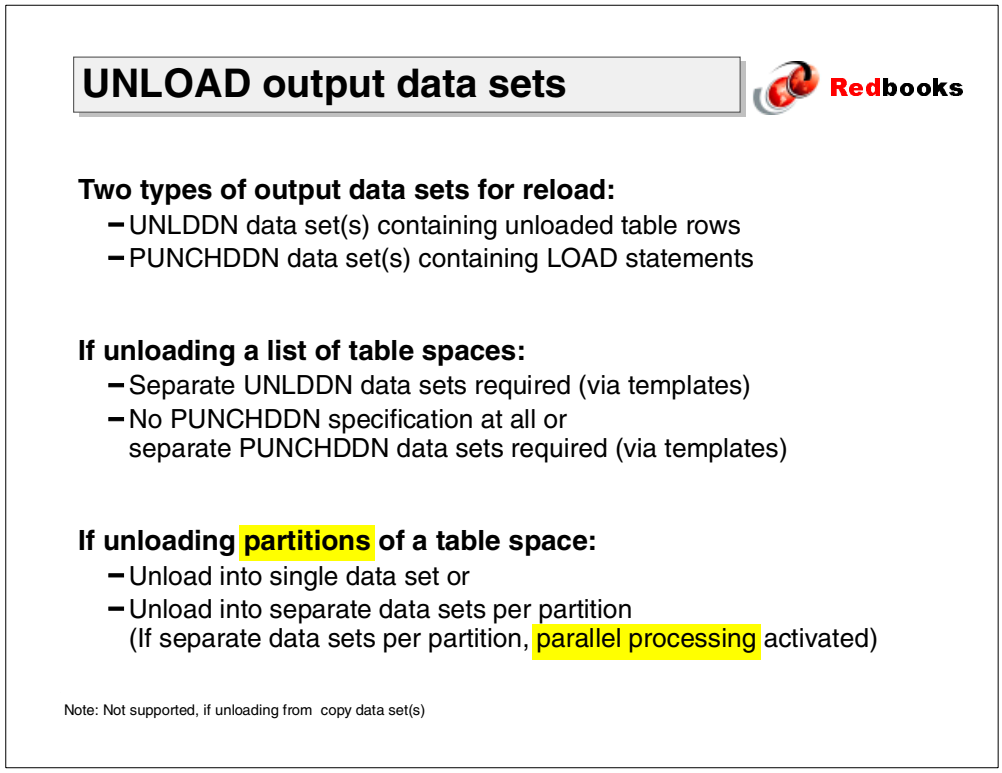
Unloading a single piece with the FROMCOPY option should be avoided: When a copy (either full or incremental) of a piece of a segmented table space consisting of multiple data sets is specified in the FROMCOPY option, and if a mass delete was applied to a table in the table space before the copy was created, deleted rows will be unloaded if the space map pages indicating the mass delete are not included in the data set corresponding to the specified copy.

**Recommendation:** You should concatenate the data sets of either partitions or pieces in the order of the data set number, and you should not intermix recent copy data sets with older copy data sets; otherwise the results may be unpredictable.

***Full and incremental copies***
It is possible to concatenate a full copy and incremental copies for a table space, a partition, or a piece using the FROMCOPYDDN option, but duplicate rows will also be unloaded in this case. If it is possible, consider to use MERGECOPY to generate an updated full copy as the input to the UNLOAD utility.

***Inline copies from LOAD or REORG***
If a copy was created by the in-line copy operation, the copy can contain duplicate pages. If they exist, the UNLOAD utility issues a warning message. All the qualified rows in duplicate pages will be unloaded into the output data set.

***Restrictions***
- If the FROMCOPY or the FROMCOPYDDN option is used, only one output data set can be specified, that is, unloading the data to multiple output data sets by partition is not supported for copy data sets.
- In one UNLOAD statement, all the source copy data sets you unload must pertain to the same single table space.
- The table space name must be specified in the TABLESPACE option.
- If a copy contains rows or dropped tables, these rows will be ignored, that is, you cannot unload dropped tables.
- The input data set where the copy resides cannot be a VSAM data set.




### 5.4.5 Output data sets

The UNLOAD utility will create two types of output data sets:

- **UNLDDN data set(s)** - One or more data sets that will contain the unloaded table rows. More precisely, the UNLOAD utility supports:
  - One single output data set for all data to be unloaded from a table space or from one or more copy data sets.
  - Multiple output data sets, that is, one physically distinct data set:
    - For each table space, if you are unloading a list of table spaces
    - For each partition when the data is unloaded from a DB2 partitioned table space.

  **Note:** Multiple output data sets are not supported if you unload from copy data sets, that is, if you use the FROMCOPY or the FROMCOPYDDN option.

  The maximum length of an output record is limited to 32 KB, including the record header field, and field related overhead like the NULL indicator bytes. By default, the first two bytes of the output records are reserved as the record header (RECORD OBID). The UNLOAD utility provides the HEADER option, either to replace it with a string of an arbitrary length or to eliminate it. Note that the HEADER option is a part of the FROM TABLE specification. Therefore,, if the LIST option is used, the default two byte header will always be included in the output records.
- **PUNCHDDN data set(s)** - One or more data sets that will contain the generated LOAD statements to be used when reloading the data.

If PUNCHDDN is not specified and a SYSPUNCH DD name does not exist, the LOAD statement(s) will not be generated.

***Reload support***
The output records written by the UNLOAD utility are compatible as the input to the LOAD utility (reloadable into the original or into different table(s) using the LOAD utility). The format of the output records (field formats and positions) can be identified by the generated LOAD utility statement written to the data set allocated under the SYSPUNCH DD name or under the DD name specified by PUNCHDDN.

The generated LOAD statement(s) will always include a WHEN specification associated with an INTO TABLE clause to identify the table into which the rows are to be reloaded, unless the HEADER NONE option is applied to the FROM TABLE clause in the UNLOAD control statement.

***Unloading a list of table spaces***
If you unload a list of table spaces, defined by a LISTDEF and referred to via the LIST keyword, the UNLOAD utility requires both, a separate UNLDDN data set and a separate PUNCHDDN data set, for each table space. If the list comprises multiple table spaces, this can only be accomplished by using templates for both, the UNLDDN data sets and the PUNCHDDN data sets. These templates must contain the table space as a variable (&TS., also possible: &IS., &SN.). Otherwise, unloaded data for some table spaces will be lost (overwritten) or data set allocation errors can occur.

As an exception from this rule, you can omit any PUNCHDDN specification, in which case LOAD statements are not generated at all.

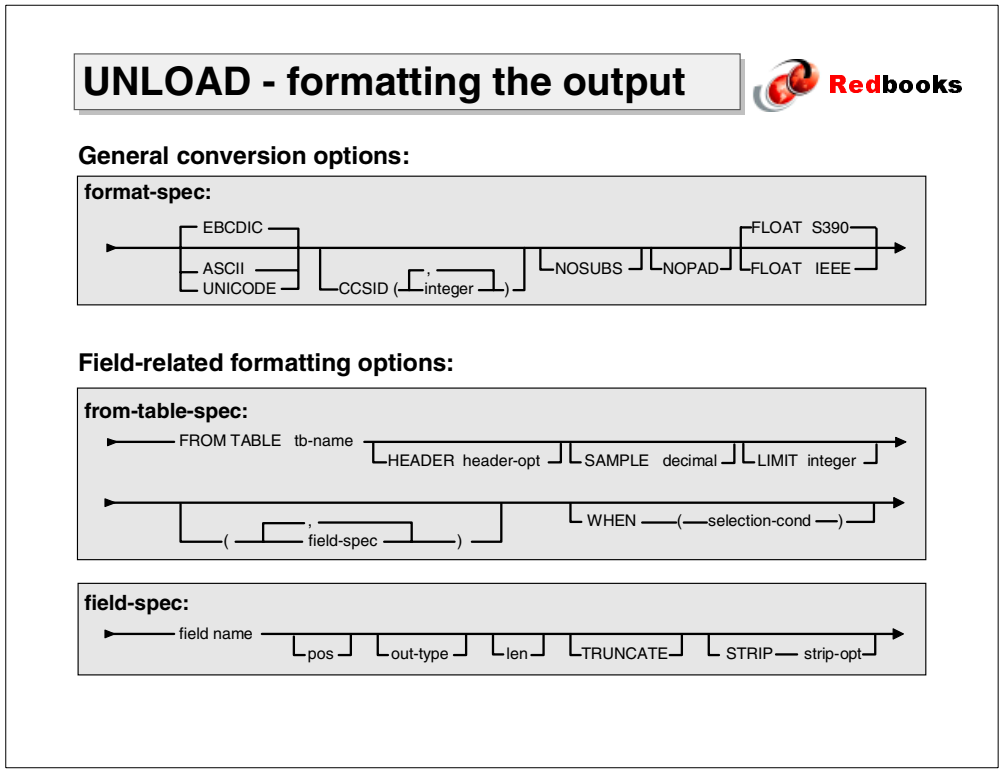
***Unloading partitions - partition parallelism***
Suppose you want to unload specific partitions only or a table space on partition level. These selected partitions can be unloaded into a single data set. Alternatively, you can unload into separate data sets per partition, if the following conditions are met:

- You use the UNLOAD utility rather than REORG UNLOAD EXTERNAL or DSNTIAUL.
- The UNLDDN and the PUNCHDDN output data sets are defined exclusively by templates containing the partition as a variable (&PART. or &PA.).

  Again, you can omit any PUNCHDDN specification, in which case LOAD statements are not generated at all.
- You are **not** unloading from copy data set(s).

Moreover, if the partitions are unloaded into individual data sets, the UNLOAD utility automatically activates multiple task sets and runs in *partition-parallel unload mode*. The maximum number of task sets will be determined by the number of CPU nodes of the processor unit on which the UNLOAD job runs.

**Note:** This is the major performance enhancement introduced by UNLOAD.




### 5.4.6 Output formatting

In this section we examine the general conversion options that are valid for the entire UNLOAD utility invocation and govern how the data being unloaded should be converted.

- **Numeric types:**

  Columns of the numeric types (SMALLINT, INTEGER, FLOAT, DOUBLE, REAL, and DECIMAL) will be converted from the DB2 internal format to the S/390 format. Please note that these are still internal representations, which you can change for each individual column via the EXTERNAL keyword.

  If a column of a floating point type is unloaded in the binary form, either the S/390 format (hexadecimal floating point, or HFP) or the IEEE format (binary floating point, or BFP) can be selected. The default is the S/390 format.

- **Character types - encoding schemes:**

  During unload, you can convert character data into a different encoding scheme, more specifically:

  - Character code conversion (EBCDIC, ASCII, UNICODE with or without CCSID code page specification) can be performed on the character type fields (SBCS, MIXED, or DBCS), including the numeric columns converted to the external (character) format and the character type LOBs. This option is not applied to data whose subtype is BIT.

- For converting into another CCSID, you can specify
  CCSID (integer [, integer [, integer] ] ]
  If one of the argument is specified as 0 or is omitted, the encoding scheme specified by EBCDIC, ASCII, or UNICODE is assumed for the corresponding data type (SBCS, MIXED, or DBCS). If the CCSID option is omitted,
  - If the source data is of character type, the original encoding scheme is preserved.
  - For character strings converted from numeric data, DATE, TIME, or TIMESTAMP, the default encoding scheme of the table is used. (See CCSID option of CREATE TABLE.)
- When a string is converted from one CCSID to another, including EBCDIC, ASCII, and UNICODE conversion, a substitution character is sometimes placed in the output string. For example, this substitution occurs when a character (referred to a codepoint) that exists in the character set of the source CCSID does not exist in the character set of the target CCSID. The NOSUB option prevents such a character substitution. More specifically, if a character substitution is attempted when unloading data with the NOSUBS keyword, it will be treated as a conversion error. A record with this error will not be unloaded.

- **Character types - VARCHAR**

  A varying length column can be unloaded to a fixed length output field (padded to the maximum length), or to a variable length output field (without padding) using the NOPAD option. Padding, the default, ensures that all unloaded records of one table have the same length. In either case, unless the fixed length data type is explicitly specified for the field, the data itself is treated as a varying length data, that is, a length field is appended to the data. This helps when reloading into a variable length field.

  LOAD is able to process records with variable length columns that are unloaded or discarded using the NOPAD option. LOAD can also take into account null indicators for these fields, if these indicators also are not in a fixed position in the input data set. An example for such a LOAD statement, generated by UNLOAD or REORG UNLOAD EXTERNAL, can be found in the *DB2 Utility Guide.*

  **Note:** If an output field is nullable and has a varying length, then the NULL indicator byte precedes the length field, and the length field in turn precedes the data field. No gaps will be placed between the NULL indicator, the length field, and the data field.

- **Character types in general:**

  Data can be unloaded into fields with a smaller length with a TRUNCATE option and/or a STRIP option - for a detailed description, see the *DB2 Utility Guide.*

- **DATE / TIME / TIMESTAMP types:**

  Columns of these types will always be converted into the external formats, that is into character strings, based on the installation dependent DATE, TIME, and TIMESTAMP formats.

  Note that the character code conversion (EBCDIC, ASCII, UNICODE and/or CCSID), if specified, is always applied to the character type data.

***Field-related options***

The following options apply to a specific column, respectively field, you unload. Instead of a detailed description of all options and possible combinations (given in the *DB2 Utility Guide*), only short hints are given to show which functionality is offered by UNLOAD:

- **Selection and reordering of columns**

  You can select the table columns you want to unload and the sequence of these columns by explicitly specifying a list of these columns in the `from-table-spec,` with the sequence of columns you want.

- **pos**

  For specifying at which position, respectively column, in the output data set you want to store a specific table column, you use a similar syntax as in the LOAD statement, starting with the keyword POSITION.

- **out-type**

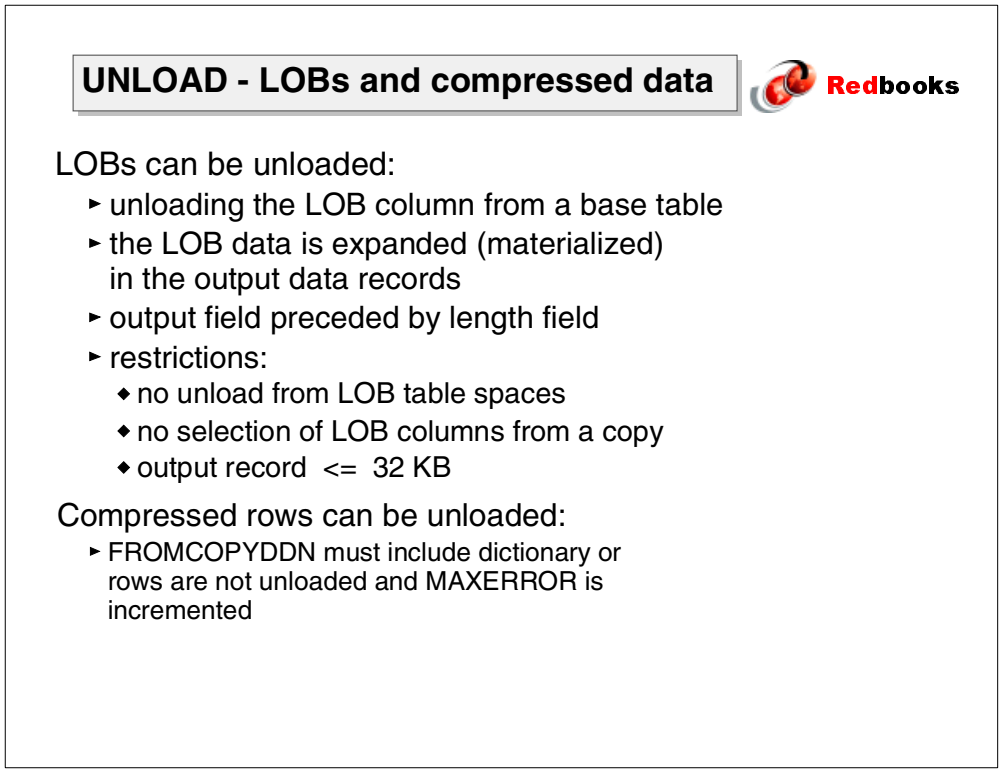
  This specifies the output data type for this column. Even a constant value can be specified. Furthermore, for some data types, additional keywords can be added, most importantly the EXTERNAL keyword (known from the LOAD utility), which is optional for numeric data types, but mandatory for the DATE, TIME, and TIMESTAMP data types.

- **len**

  This specifies the length of the output field.

- **TRUNCATE, STRIP**

  These specify how to shorten the output field.




### 5.4.7 LOBs and compressed data

In this section we include considerations on unloading LOBs and compressed data.

***Unloading LOBs***

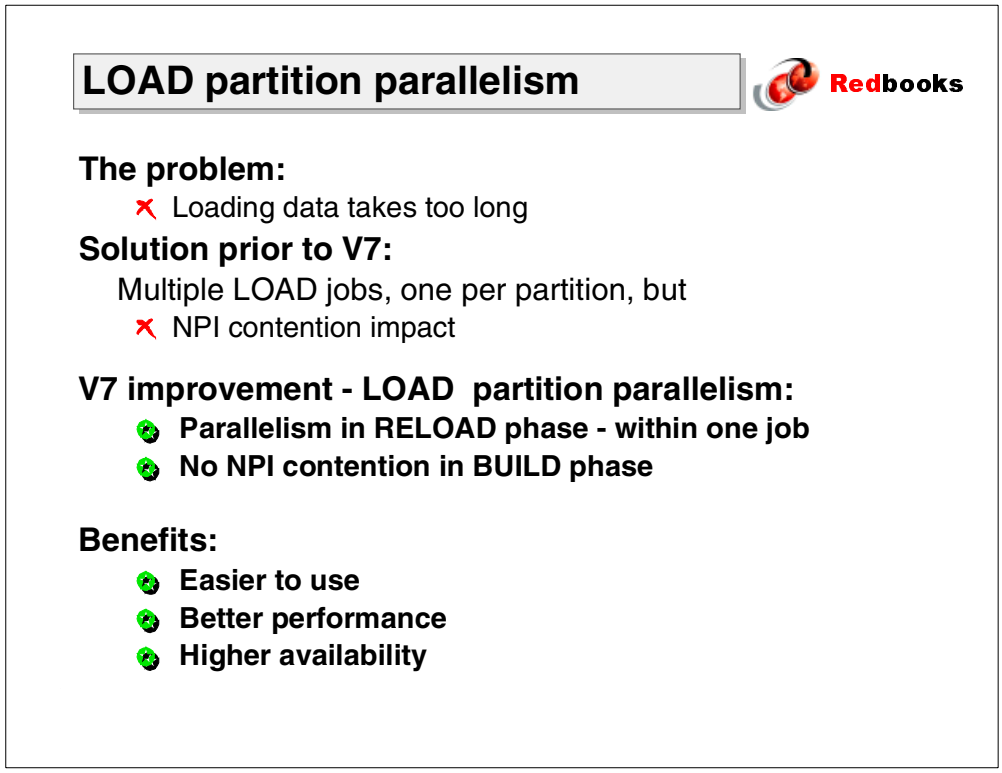
- If a table contains LOB columns and the LOB columns are selected to be unloaded (either explicitly or implicitly by omitting the entire list of field specifications), the LOB columns are replaced by the actual LOB data, that is, the LOBs are materialized in the output records.
- As stated earlier, UNLOAD supports output records with a total length of up to 32KB.
- No LOB support from copy data sets

  From a copy data set, selection of LOB columns is not supported, that is, unloading rows containing LOB columns from copies will be supported only when the LOB columns are **not** included in the field specification list.
- Length field

  As LOBs have a varying length, they are handled similar to VARCHAR. But whereas for VARCHAR and VARGRAPHIC, the preceding length field is a 2-byte binary integer, the length field for BLOB, CLOB, and DBCLOB, is a 4-byte binary integer.

***Unloading compressed data***

Compressed rows are unloaded as long as the compression dictionary is accessible from within the execution. Otherwise, an error is considered to have occurred for each row and the MAXERR option takes place.




## 5.5 LOAD partition parallelism

***The problem***

More and more customers have to load enormous amounts of data, for example, because of data warehouse or business intelligence applications. On the other hand, availability requirements cut down the batch window. For these customers, loading the data takes too long.

***Solution prior to V7***

If a partitioned table space must be loaded, dedicated LOAD jobs per partition are sometimes used in order to reduce the RELOAD time. But then different jobs try to access the non-partitioning indexes (NPI), resulting in considerable contention problems on these indexes.

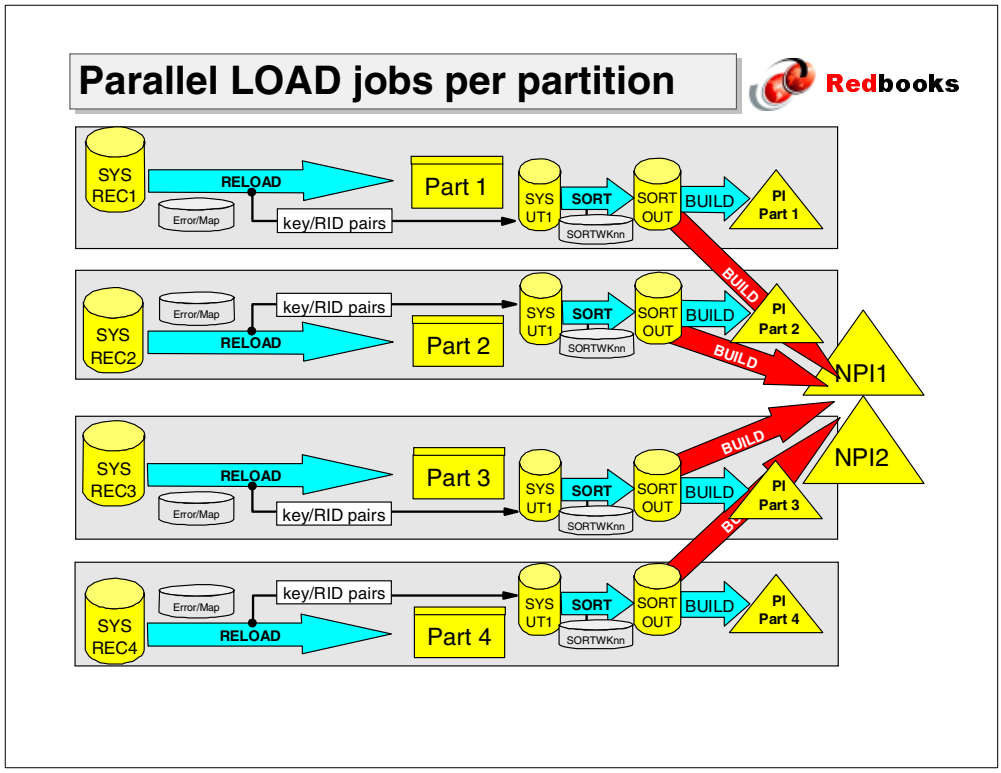
***V7 improvement - LOAD partition parallelism***

DB2 introduces this enhancement to the LOAD utility which addresses two issues at the same time:

- Partitions can be loaded in parallel within the same LOAD job
- NPI contention is reduced

***Benefits***

- Easier to use, as only one job has to be submitted
- The entire LOAD process does not take as long as before, as two phases are performed faster, the RELOAD phase because of parallel processing, the BUILD phase because there is no NPI contention.
- Because of these performance improvements, the time the partitioned table space is not available for SQL application is reduced.




### 5.5.1 Parallel LOAD jobs per partition

In order to set a common ground, let us first look at the situation at DB2 V5 or V6 level without the SORTKEYS parameter specified, reviewing the main parts of the LOAD process.

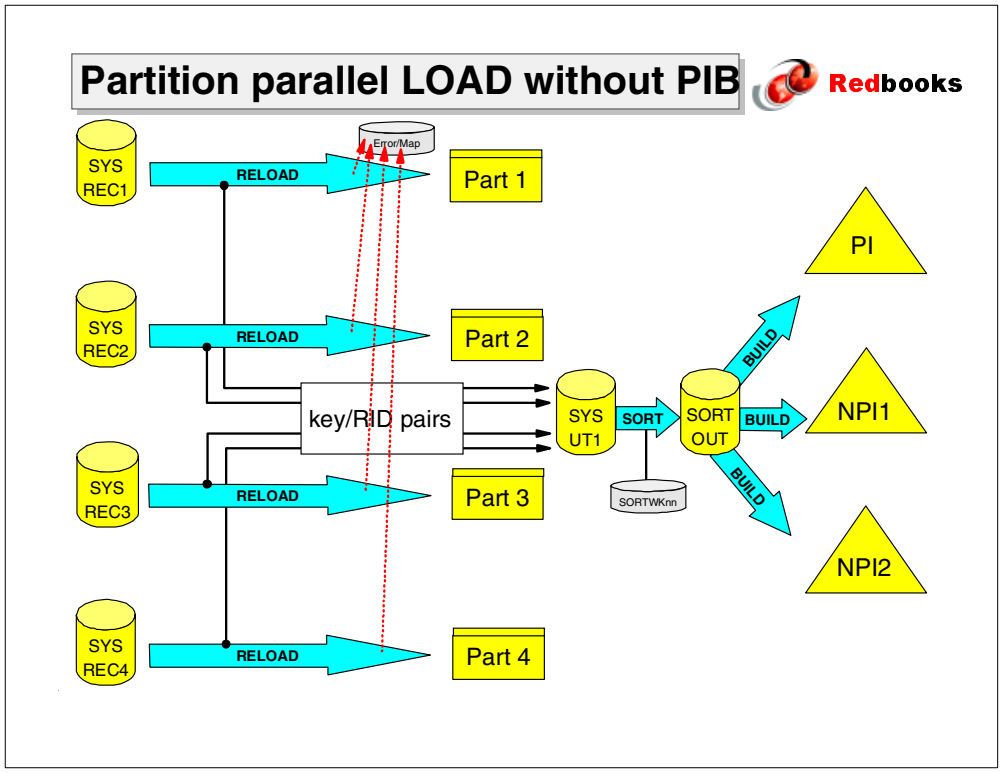
If a partitioned table space has to be loaded, and if the input records are stored in separate input data sets, one for each partition, then you can accelerate the LOAD task by submitting several independent LOAD jobs at the same time, one for each partition to load.

Looking at one of these jobs, we see that, during the RELOAD phase, the input records are loaded from the input data set into the respective partition: the keys for all indexes are extracted from the input records and, together with the RIDs of the just loaded records, stored in the SYSUT1 data set dedicated to this job. In the SORT phase, all these key/RID pairs for all indexes are sorted and written to the SORTOUT data set dedicated to this job. In the following BUILD phase, each job can build the respective index partition of the partitioning index easily.

But when all these jobs want to build the non-partitioning indexes, considerable contention occurs, as these jobs often want to insert into the same index page at the same time, thereby even inducing page splits. To ensure the physical page integrity, DB2 has to serialize these competing accesses, and this increases the elapsed time.

As a relief, some customers drop all NPIs, load the partitions by dedicated jobs, then recreate these indexes, sometimes with DEFER YES and following with REBUILD to load these indexes.

Note that any non-terminating errors result in records being written to the error and map data sets during the RELOAD phase. If any records encountered errors, the DISCARD phase, performed way after the phases shown on the foil, will process the error and map data set records, reread the original input records from the input data set, and write them to the corresponding discard data set.




## 5.5.2 Partition parallel LOAD without PIB

The new *LOAD partition parallelism* function of DB2 V7 enables the parallel loading of partitions within a LOAD job and avoids the NPI contention that we have seen in the previous foil. Note that, in this example, the *Parallel Index Build* (PIB) activated by the keyword SORTKEYS is not used; this option is considered in the next foil.

In this case a single job launches multiple RELOAD subtasks, optimally four, one for each partition, and in correspondence of each input data set. Each of these RELOAD subtasks reads the records from its input data set and loads into its partition. The keys for all indexes are extracted and, paired together with the RIDs of the just loaded records, are written to the common SYSUT1 data set. After sorting the normal BUILD phase is performed loading serially the indexes. This sequence allows the NPI contention to be avoided.

As for the one SYSUT1 data set, for this single job there exist only one error data set and one map data set.

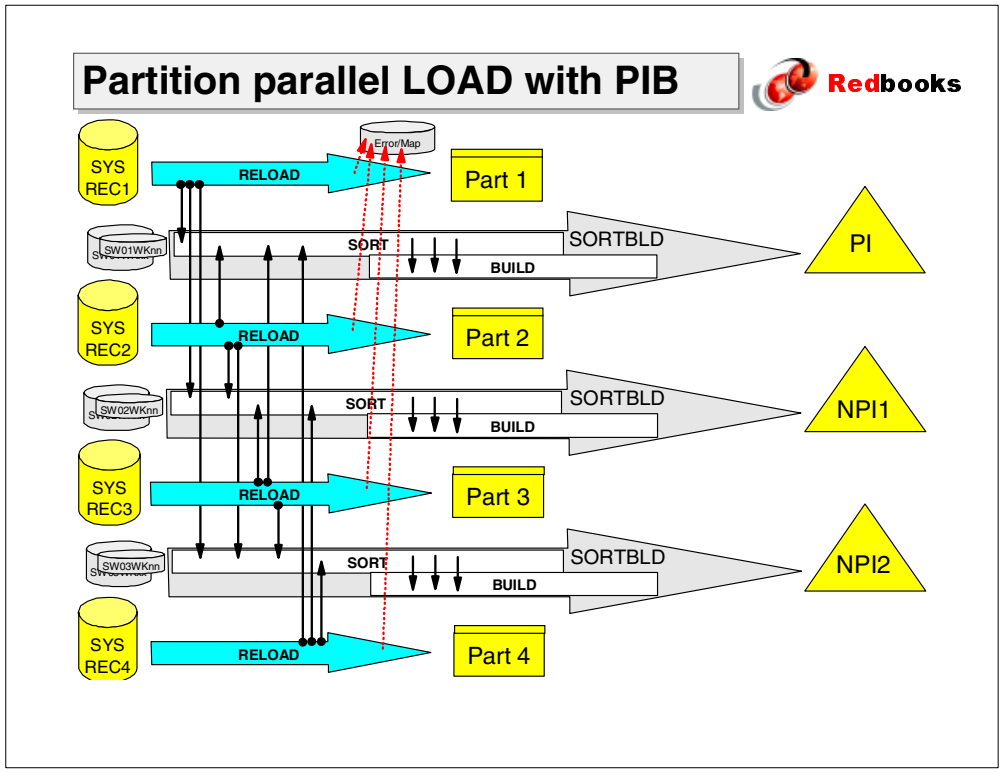
***Number of RELOAD tasks***

The number of RELOAD tasks is determined by:

- Number of CPUs
- Available virtual storage
- Available number of DB2 threads

The existing message DSNU397I explains the constraints on number of tasks. Furthermore, there is a new message:

```
DSNU364I PARTITIONS WILL BE LOADED IN PARALLEL, NUMBER OF TASKS = nnn
```




### 5.5.3 Partition parallel LOAD with PIB

The purpose here is to show that the new *LOAD partition parallelism* function works well together with the *Parallel Index Build* (PIB) function introduced with DB2 V6. The same considerations apply to REBUILD INDEX and REORG.

***PIB and SORTKEYS***

Prior to DB2 V7, the LOAD utility already supported parallelism. DB2 V5 introduced the SORTKEYS option to use Sort and eliminate multiple I/O accesses to intermediate workfiles when building the keys. DB2 V6 used the same option to activate the PIB as well: multiple pairs (as many as the indexes) of sort and build subtasks that can build the indexes in parallel.

Let us review the two accelerating effects of SORTKEYS:

- The SORT, the BUILD, and optionally the inline STATISTICS phases are performed partially in parallel. This way, the elapsed time of a LOAD job can be considerably reduced.
  - Phases:

    As these phases are no longer steps performed in sequence, the new SORTBLD phase has been introduced, carrying out the work of the former SORT, BUILD, and optionally STATISTICS phases. The SORTBLD phase contains now the mostly overlapping SORT, BUILD, and optionally STATISTICS sub-phases. Furthermore, the RELOAD phase and the SORTBLD phase can partially overlap.

- Subtasks:

  When SORTKEYS is activated, optimally one pair of subtasks will be started for each index to build. The two subtasks are the SORT subtask and the BUILD subtask. (Instead of having a pair of subtasks, there can be a triple of subtasks, being the SORT, the BUILD, and the inline STATISTICS subtasks.)

  It is not always possible to start as many pairs (or triples) of subtasks as many indexes exist. One way to influence the number of pairs (or triples) is presented on the foil: By providing sort work data sets under the DD name SWmmWKnn. With mm you specify the maximum number of pairs (or triples) of subtasks. Details on the standard DB2 manuals.

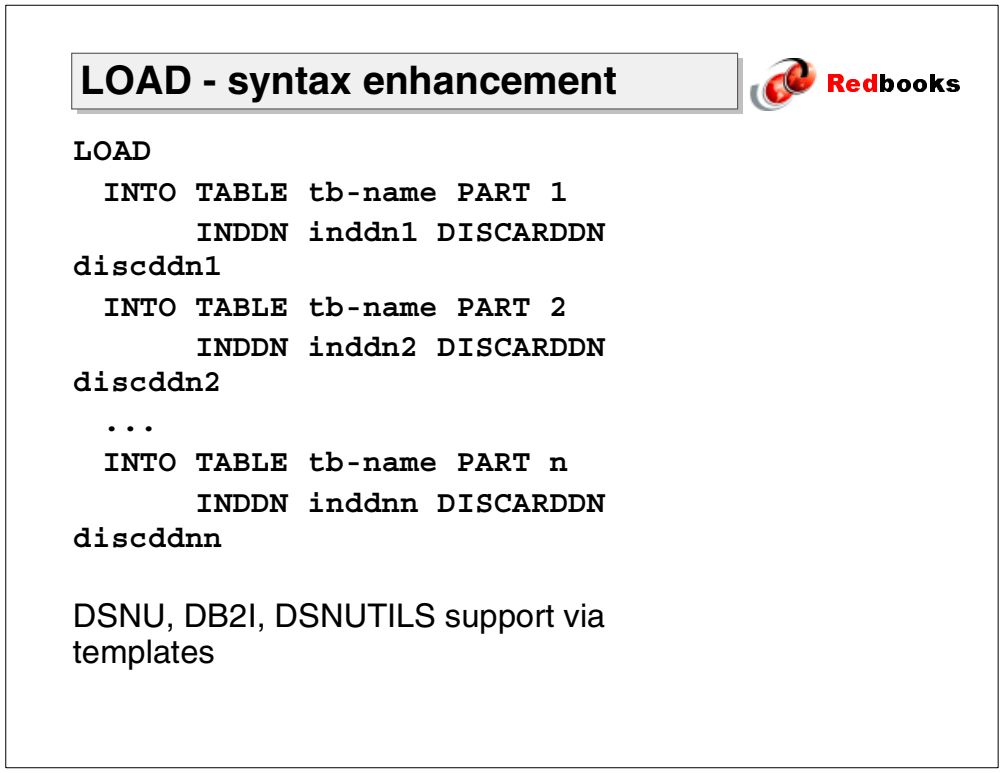
- Considerable I/O processing is eliminated by not using SYSUT1 and SORTOUT.

  The key/RID pairs are not written to the SYSUT1 data set (as shown in the previous section), but are directly piped to the SORT subtasks. These SORT subtasks perform the sort and directly pipe the sorted key/RID pairs to their BUILD subtasks instead of writing to the SORTOUT data set as before. These BUILD subtasks eventually build the indexes in parallel. As all indexes are individually built by one single BUILD subtask, NPI contention is eliminated.

***Partition Parallel LOAD together with Parallel Index Build***

Prior to DB2 V7, when PIB is activated via SORTKEYS, one single RELOAD task pipes the key/RID pairs to multiple SORT subtasks. Now, when *partition parallel Load* is also activated, DB2 supports that multiple RELOAD tasks, roughly dependent on the number of partitions, can pipe their key/RID pairs to the SORT/BUILD subtasks (one per index).

When SORTKEYS is specified, then some tasks are allocated for reloading, other tasks need to be allocated for sorting index keys and for building indexes in parallel. Thus the number of RELOAD tasks may be reduced in order to improve the overall performance of the entire LOAD job.




### 5.5.4 LOAD - syntax enhancement

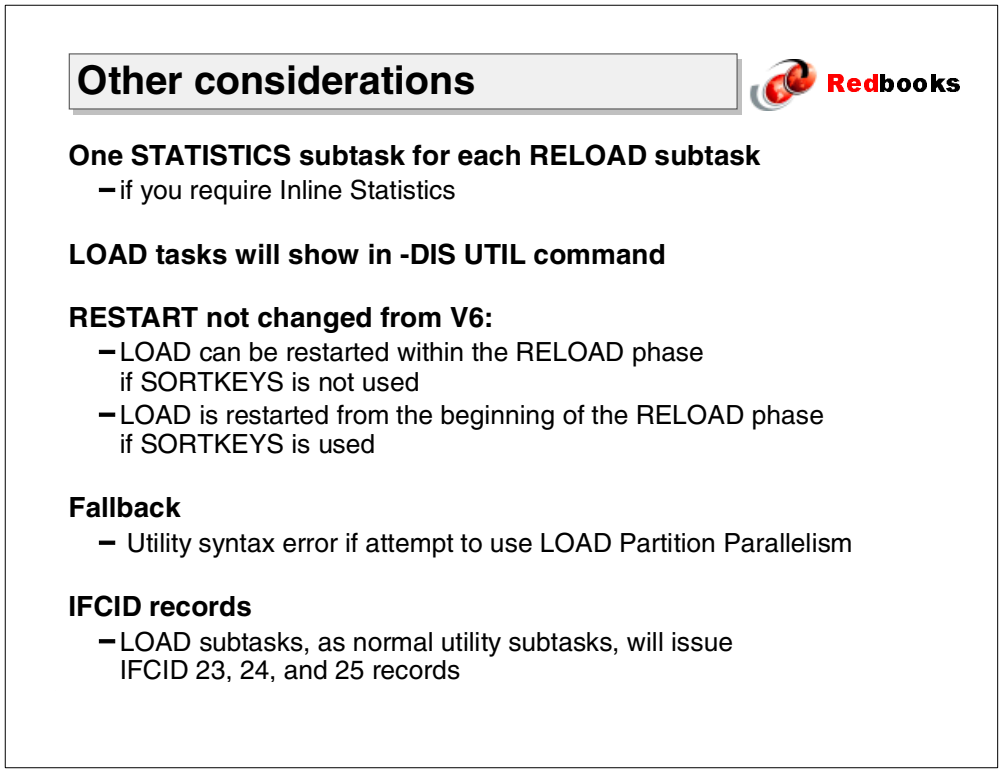
In order to allow each partition to be loaded from a separate data set, with discards (optionally) written to discard data sets for that partition, the LOAD syntax allows the specification of the INDDN and the DISCARDDN keywords as part of the INTO TABLE PART specification.

***Restrictions to the use of the INDDN and DISCARDDN keywords***

- INDDN and DISCARDDN are only allowed in the INTO TABLE specification if the PART keyword is specified. That means, they are not allowed for segmented or linear table spaces.
- If INDDN and DISCARDDN are specified in the INTO TABLE PART specification, neither may be specified at the table space level, that is, before the INTO TABLE specification.
- DISCARDDN may not be specified in the INTO TABLE PART specification unless INDDN is also specified there.
- If INDDN and DISCARDDN are specified in one INTO TABLE PART specification, and more than one INTO TABLE PART specification is supplied, they must be specified in all of the INTO TABLE PART specifications.

***DSNU, DB2I, DSNUTILS***

Multiple INDDN and DISCARDDN keywords are not explicitly supported by the DSNU CLIST, the DB2I utility panels, or the DSNUTILS stored procedure.



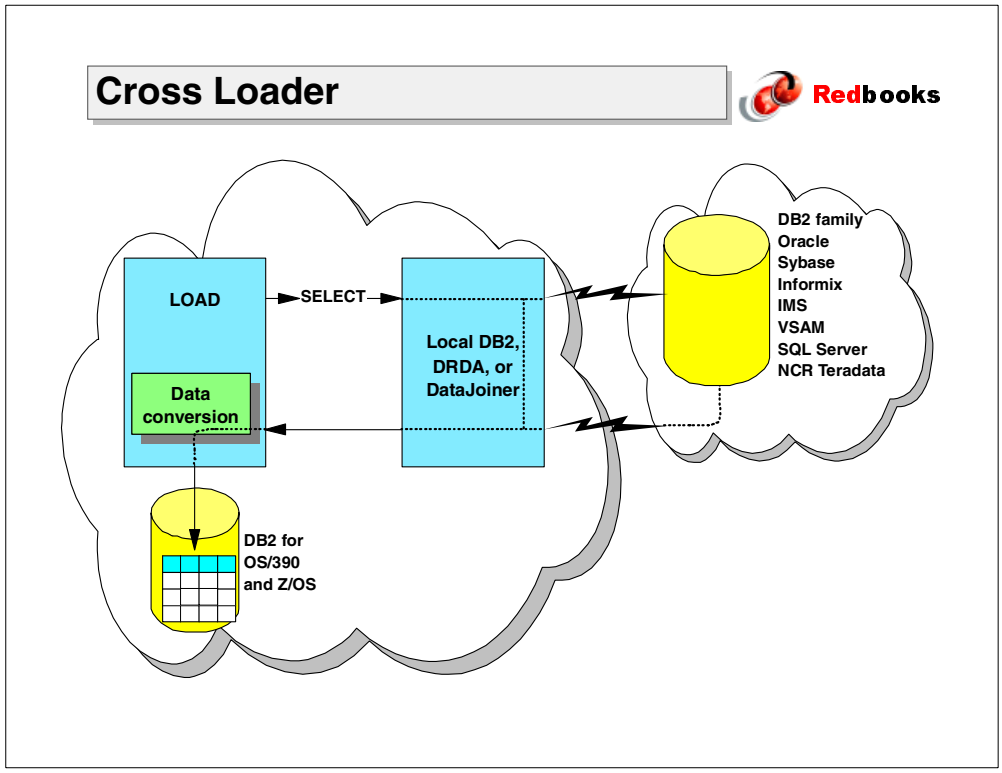

### 5.5.5 Other considerations

Here, we have grouped several considerations not covered in prior sections.

***Instrumentation***

When LOAD executes with partition parallelism, two or more subtasks will perform the loading of the data. Each subtasks will issue the following IFCID records:

1. One IFCID 23 at the start of the subtask
2. One IFCID 24 for each partition to be loaded, issued before the data is loaded
3. One IFCID 25 at the end of the subtask




## 5.6 Cross Loader

More and more customers have the need to move data across multiple databases, even across multiple platforms, in an easy way and with good performance. Cross Loader is the name given to a set of functions that have been added to the Load utility with these objectives. Think of a dynamic input file to the LOAD, implemented by executing SQL on your current data. Add DRDA and its flexible connectivity functions. Start moving data across.

The DB2 Family Cross Loader, a new function with DB2 V7, combines the efficiency of the IBM LOAD utility with the robustness of DRDA and the flexibility of SQL. It is an extension to the IBM LOAD utility which enables the output of any SQL SELECT statement to be directly loaded into a table on DB2 V7. Since the SQL SELECT statement can access any DRDA server, the data source may be any member of the DB2 Family, DataJoiner, or any other vendor who has implemented DRDA server capabilities. The Cross Loader is much simpler and easier than unloading the data, transferring the output file to the target site, and then running the LOAD utility. It can also avoid the file size limitation problems on some operating systems.

### *The EXEC SQL statement*

Basically you can directly load the output of a dynamic SQL statement into a table on DB2 V7. Within the new EXEC SQL utility statement you can declare a cursor or specify any SQL statement that can be dynamically prepared. The ENDEXEC indicates the end of the statement. The Load utility performs an EXECUTE IMMEDIATE on the SQL statement. Errors encountered during the checking of the statement or the execution will stop the utility and an error message will be issued. No host variables are allowed in the statement. Also no self referencing loads are allowed.

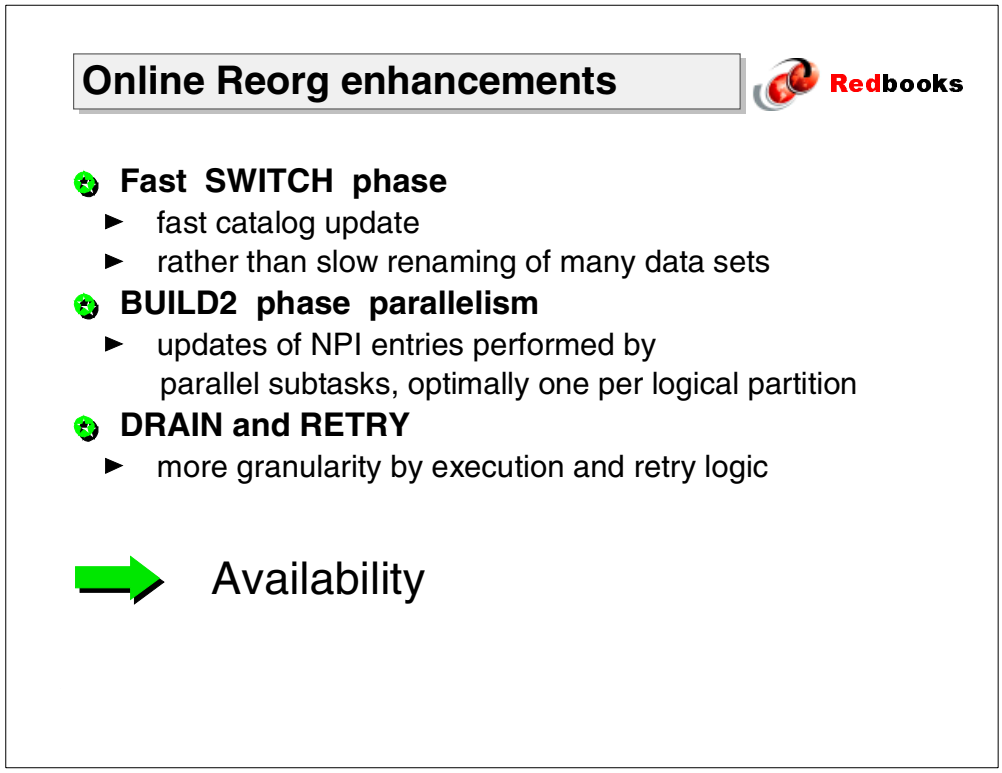
The following simple example first creates a table and then declares a cursor and executes a SELECT on the Employee table of the sample DB2 database. The results are then loaded into a table while updating the statistics on the catalog. Notice that DDL is allowed, there is an implicit COMMIT between the EXEC SQLs, and that the INCURSOR option of the LOAD statement must name the cursor C1 declared in the EXEC SQL statement.

```
EXEC SQL
CREATE TABLE MYEMP LIKE DSN8710.EMP
ENDEXEC
EXEC SQL
DECLARE C1 CURSOR FOR
SELECT * FROM DSN8710.EMP
WHERE SALARY > 10000
ENDEXEC
LOAD DATA
INCURSOR(C1)
REPLACE
INTO TABLE MYEMP
STATISTICS
```

When loading data from a remote location, you must first bind a package for the execution of the utility on the remote system, as in:

```
BIND PACKAGE (location_name.DSNUTILS) COPY(DSNUTILS.DSNUGSQL)-
    ACTION(REPLACE) OPTIONS(COMPOSITE)
```

Then, you must specify the three part name for the table with the `location_name`.




## 5.7 Online Reorg enhancements

There are three phases of an online REORG, during which the data is totally or partially not available, these are the LOG, the SWITCH, and the BUILD2 phases. In DB2 V7, the processing in two of these phases, SWITCH and BUILD2, is accelerated, hence these phases are shorter. Therefore, the availability of the data for the SQL applications is much higher, timeouts of SQL applications because of an online REORG are less likely to happen. The improvements are:

- Fast SWITCH phase

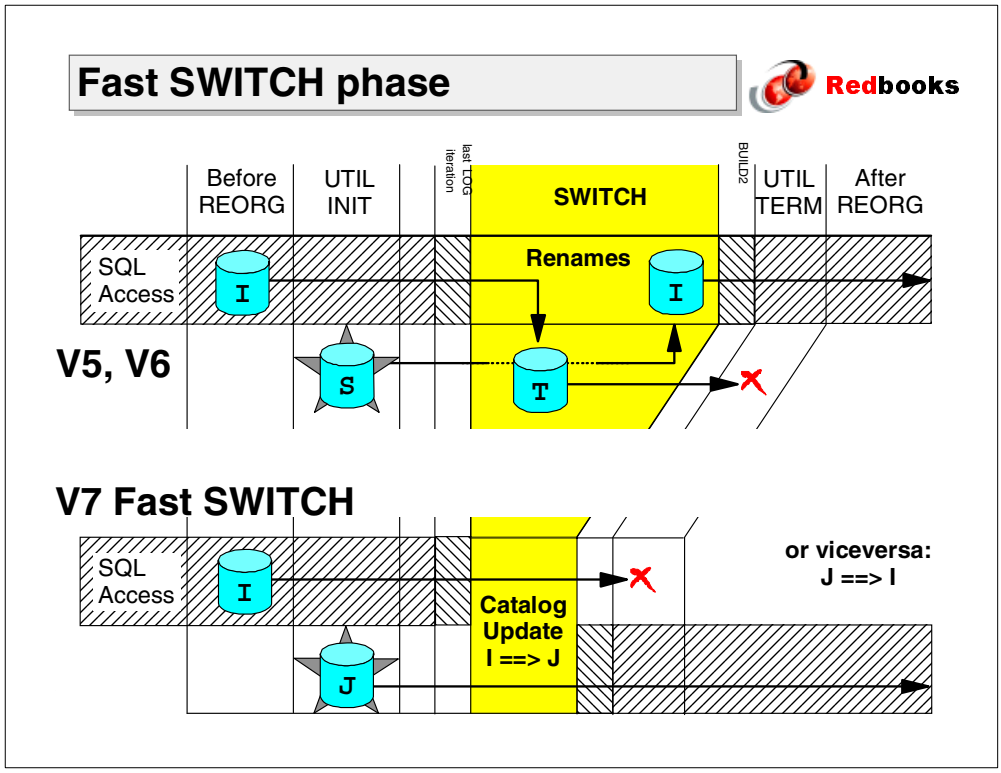
  In the SWITCH phase, all data sets involved in the online REORG are no longer renamed, as this can be very time consuming. Instead, the catalog and the directory are updated to point to the shadow data sets rather than to the original data sets.

- BUILD2 phase parallelism

  If only some partitions of a partitioned table space are subject to an online REORG, the shadow non-partitioning indexes (NPIs) do not contain all index entries, hence a replacement of the original NPIs by the shadow NPIs is not possible. Instead, the individual original index entries must be replaced by the shadow index entries. This processing is now done in parallel during the BUILD2 phase thereby greatly decreasing the elapsed time of this step.

- DRAIN and RETRY

  The new parameters DRAIN and RETRY have been added to the REORG REFERENCE or CHANGE utility statement to control the time that the utility will wait to establish a drain, and also to enable to retry waiting for a drain.




### 5.7.1 Fast SWITCH

Concentrating on the data sets, let us shortly see how online REORG, that is REORG SHRLEVEL REFERENCE or REORG SHRLEVEL CHANGE, works prior to DB2 V7.

In the UTILINIT phase, *shadow objects* of the table space (or its partitions) and for the index spaces (or their partitions) are created. Strictly speaking, this is not true, as these shadow objects are not reflected in the catalog. What it means is, that new *shadow data sets* are created, one for each data set of the original objects (or their partitions). The data set names of these shadow data sets differ from the original data set names insofar as their fifth qualifier, also referred to as *instance node,* is 'S0001' rather than 'I0001'.

In the SWITCH phase, DB2 renames the original data sets and the shadow data sets. More specifically, the instance node of the original data sets, 'I0001', is renamed to a temporary one, 'T0001'; afterwards, the fifth qualifier of the shadow data set, 'S0001', is renamed to 'I0001'.

During these renames, which take about two seconds each, SQL applications cannot access the table space. After the SWITCH phase, the applications can resume their processing on the new 'I0001' data sets.

In the UTILTERM phase, the data sets with 'T0001' are deleted, as they are not needed any more.

**Notes:**

- This description applies to DB2-managed table spaces.

- During the last log iteration and during the BUILD2 phase, SQL accesses are also limited.

***The problem: SWITCH phase too long, applications may timeout***

Some ERP applications, such as SAP and PeopleSoft, design DB2 table spaces with several hundred indexes. Others use partitioned table spaces with some hundred partitions. In both cases, several hundred data sets must be renamed in the SWITCH phase. For the renaming, DB2 invokes *Access Method Services* (AMS), AMS in turn invokes MVS supervisor calls (SVCs) which result in further cascading SVCs, for example, for checking whether the new name exists already, and whether the rename was successful.

Therefore, the elapsed time of the SWITCH phase can become too long. As the SQL applications cannot access the table space during that time, they may timeout because of the online REORG, which is exactly what you are trying to avoid.

***The solution: V7 Fast SWITCH***

A revised alternative to the processing is performed to speed up the SWITCH phase thus making the phase less intrusive to data access by others:

- Invocation of AMS to rename the data sets is eliminated.
- An optional processing, the Fast SWITCH, takes place:
  a. In the UTILINIT phase, DB2 creates shadow data sets. The fifth qualifier of these data sets is now ‘J0001’.
  b. In the SWITCH phase, DB2 updates the catalog and the object descriptor (OBD) from ’I’ to ’J’ to indicate that the shadow object has become the active or valid database object. During that time, SQL applications cannot access the table space.
  c. After the SWITCH phase, the SQL applications can resume their processing, now on the new ’J0001’ data sets.
  d. In the UTILTERM phase, DB2 deletes the obsolete original data sets with the instance node ’I0001’.

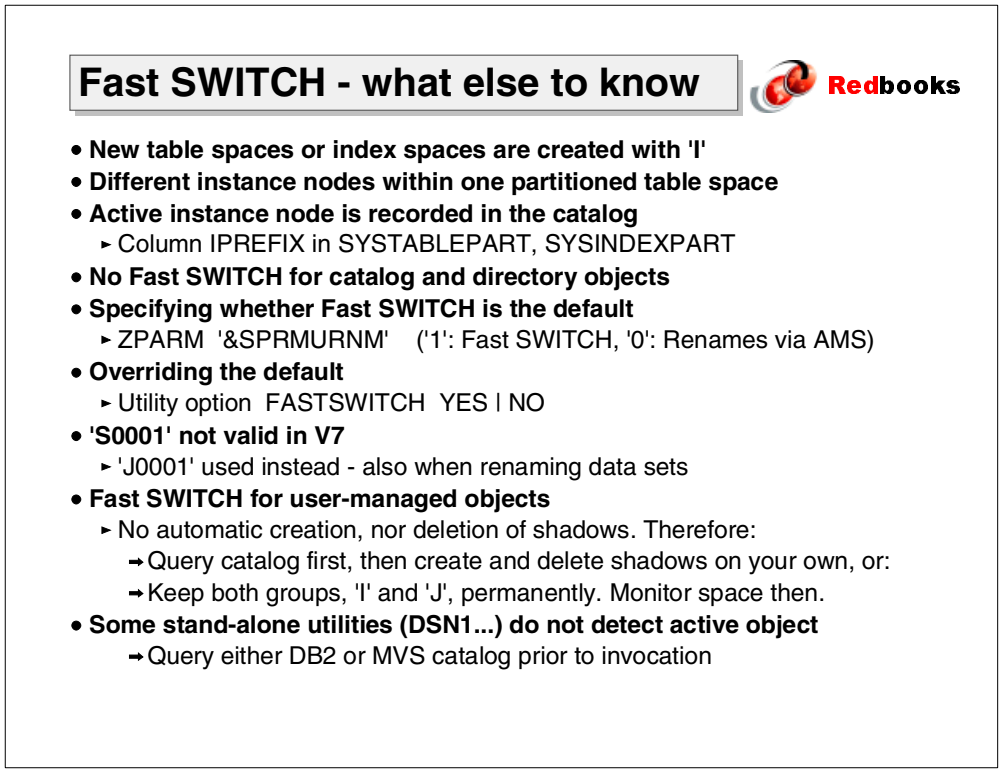
  **Notes:**

  - This description applies to DB2-managed table spaces. If the data sets are SMS controlled you need to change the ACS routines to accommodate the new DB2 naming standard and take advantage of the enhancement.
  - During the last log iteration and during the BUILD2 phase, the SQL access is limited, too; but we ignore this here.

As a result, the SWITCH phase is much shorter; therefore, the SQL applications are less likely to timeout.

***Behavior for second REORG, when J0001 already exists***

As the data set names of a table space now vary, DB2 queries the OBD of the object being reorganized in the UTILINIT phase in order to check, whether the current instance node is ’I0001’ or ’J0001’. If the current data base object has a ’J0001’ instance node, then DB2 will create a shadow object with an ’I0001’ instance node. Thus, during the SWITCH phase, the OBD and the DB2 catalog are updated registering the ’I0001’ object as the active data base object. The UTILTERM phase then deletes the database objects with the ’J0001’ instance node, the old originals.




## 5.7.2 Fast SWITCH - what else to know

***New table spaces or index spaces are created with 'I'***
As both instance nodes, 'I0001' and 'J0001', are valid in DB2 V7, the question arises, which instance node DB2 uses when a table space or an index space is created. The answer is 'I0001'.

***Different instance nodes within one partitioned table space***
After a *Fast SWITCH* online REORG on specific partitions only, more precisely, on a single partition or on a range of partitions,

- Some table space partitions have changed their instance node to, let say, 'J0001', others have kept their instance node 'I0001'.
- All non-partitioning indexes (NPI) will keep their instance nodes. (This will be understandable when you read 5.7.4, "BUILD2 parallelism" on page 263)

***Active instance node is recorded in the catalog***
The new column IPREFIX in SYSTABLEPART and SYSINDEXPART stores the active instance node, either 'I' for 'I0001' or 'J' for 'J0001'.

***No Fast SWITCH for catalog and directory objects***
Objects in DSNDB06 and DSNDB01 are not eligible for the *Fast SWITCH*, that is, they must always be reorganized utilizing the AMS rename. The reason is that DB2 has to look up and change catalog and directory for the *Fast SWITCH* method.

***Specifying whether Fast SWITCH is the default***
Customers can choose at installation level, whether or not they want *Fast SWITCH* as their default processing option for online REORGs. The default value for the new ZPARM parameter, '&SPRMURNM' in Macro DSN6SPRC, is '1' when DB2 V7 is installed. Thus the default is to exploit the Fast SWITCH feature. At installation time, you can set this ZPARM to '0', indicating that the AMS rename is the default processing for online REORGs.

***Overriding the default***
Customers can override the ZPARM default at the utility level, using a new keyword for the REORG SHRLEVEL REFERENCE / CHANGE utility.

- Specifying FASTSWITCH YES turns on the *Fast SWITCH* processing,
- Specifying FASTSWITCH NO lets the utility function as it had in V5 and V6.

***'S0001' not valid in V7***
DB2 V7 does not tolerate the instance node 'S0001' any more. As a consequence, even when reorganizing via AMS renames, the shadow data sets have the instance node 'J0001' instead of 'S0001'. The instance node of the temporary data sets can still be 'T0001'. Thus the rename acts as follows:

```
I0001 -> T0001
J0001 -> I0001
```

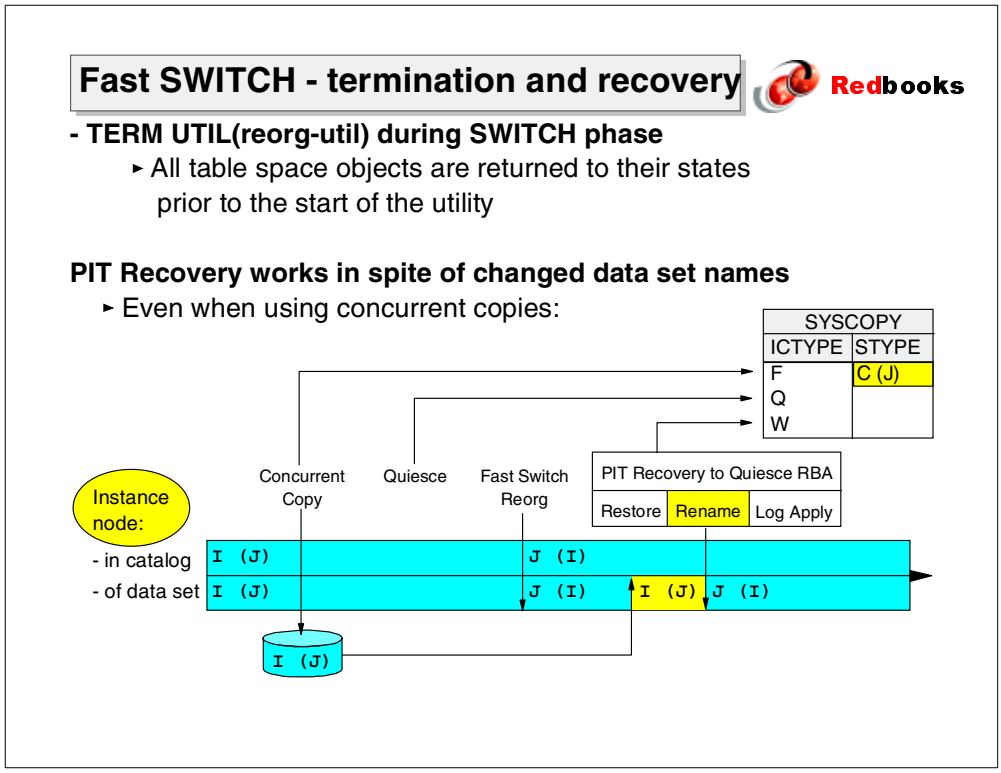
***Fast SWITCH for user-managed objects***
In the case of user-managed objects, DB2 will neither create the shadow data sets during the UTILINIT phase, nor delete the old originals in the UTILTERM phase.

Therefore, you have to query the IPREFIX column prior to executing a REORG with the Fast SWITCH method, in order to get to know the active instance node. Then you have to create the shadow data sets accordingly, that is, the data sets with instance node 'J0001', if the active node is 'I0001', or vice versa. Automation techniques could be developed to front end the REORG utility and prestage the shadow data set allocations.

Care should be taken when reorganizing an entire user-managed partitioned table space. A query for each of the data partitions, index partitions, and all non-partitioning indexes is needed to ensure that a shadow object with the correct instance node is allocated for each of the table space objects.

Alternatively, in environments where DASD space is available, both groups of data sets, the 'I0001' and the 'J0001' data sets, could be permanent allocations. But in this case, care monitoring would be required to ensure correct data set sizes.

***Some stand-alone utilities do not detect active object***
DSN1COMP, DSN1COPY, DSN1PRNT, and DSN1CHKR are not modified to dynamically allocate the correct data sets for a table space. You must query the DB2 catalog prior to invocation, or the MVS catalog, if you do not work with user-managed objects with both groups of data sets, 'I' and 'J', exist permanently.




## 5.7.3 Fast SWITCH - termination and recovery

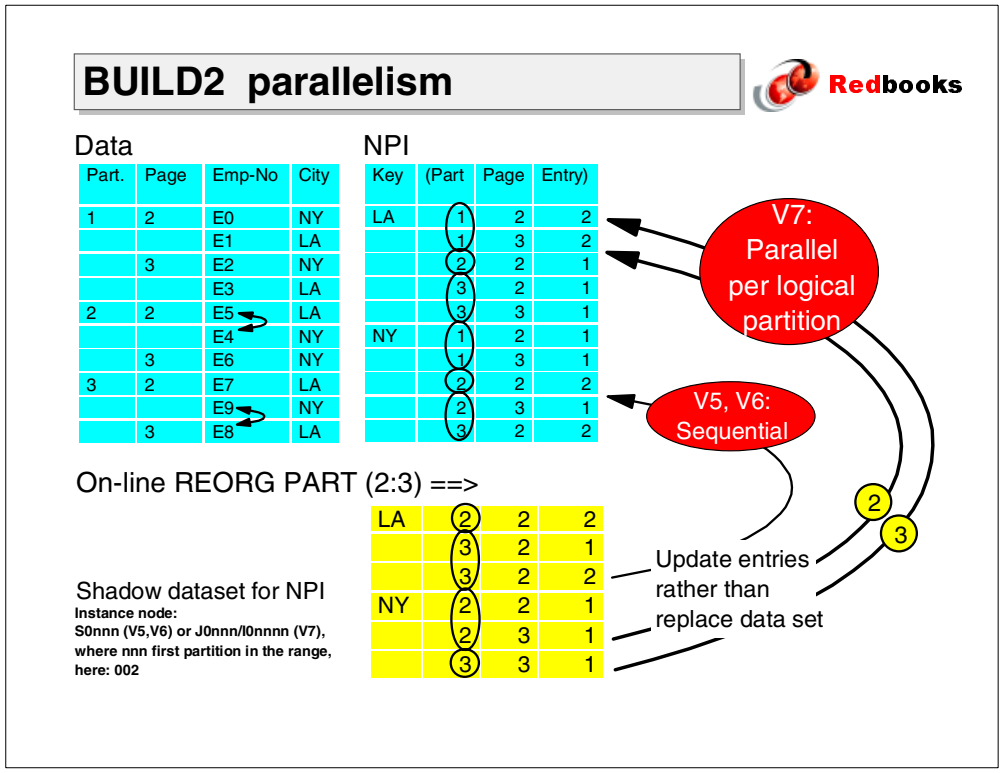
In case you wonder whether DB2 offers support in the following situations or whether you have to perform additional caution or steps when using Fast SWITCH, this foil should convince you that you can easily use the new feature.

If TERM UTIL is issued during the SWITCH phase, the objects will be returned to the status prior the execution of the Reorg.

When using concurrent copies and Fast SWITCH REORGs, the RECOVER utility does extra work to properly handle the following situation:

If you do a concurrent copy, DB2 will check the current instance node of the object and store this information in SYSCOPY: If the instance node is 'I', STYPE will be set to 'C' (as before), if the instance node is 'J', STYPE will be set to 'J'. (Here we use the abbreviations 'I' / 'J' rather than the correct instance nodes 'I0001' / 'J0001'.) A following Fast SWITCH REORG changes an initial instance node of 'I' to 'J' (or similar, an initial instance node of 'J' to 'I'). If you then recover to a QUIESCE point, the RESTORE phase is performed at first.

If you work with concurrent copies, the restored data set(s) has(have) the same name(s) it(they) had at copy execution time, for example, in our scenario on the foil, the name before the Fast SWITCH REORG took place. Therefore, the instance node of the data set(s) is again the same as the initial one, 'I' (or similar, 'J', if this was the initial one). Hence there is a mismatch between the actual instance node of the data set(s) and the recorded instance node in the catalog. The RECOVER utility therefore renames the data set(s) to correct this mismatch. Afterwards, the RECOVER utility can proceed with its normal processing, that is, with the LOGAPPLY phase.




## 5.7.4 BUILD2 parallelism

The parallelism during the BUILD2 phase introduces two benefits:

- Shorter elapsed time

  Parallelism should reduce the elapsed time of the BUIILD2 phase of an online REORG.

- Availability

  During the BUILD2 phase, SQL applications cannot access the NPIs. As a consequence, the data availability is improved, since the BUILD2 phase is shorter.

In the example on the foil, the partitions 2 and 3 are to be reorganized with all the indexes, as the records in these partitions are not in clustering order.

***BUILD2 phase - processing***

The BUILD2 phase applies when you do a REORG:

- With SHRLEVEL CHANGE or SHRLEVEL REFERENCE
- Only on a subset of the partitions of a partitioned table space, more specifically, on one partition only or on a (sub-) range of partitions only

In this case, the REORG utility must use shadow data sets, one for each non-partitioning index (NPI), in which it creates index entries for the logical partitions corresponding to the partitions being reorganized. In the example on the foil, these are only the entries of the partitions 2 and 3. Accordingly, the shadow data sets only contain a subset of the index entries for the non-partitioning indexes.

Therefore, the shadow data sets cannot just replace the original data set during the SWITCH phase. Instead, it follows an additional phase, the BUILD2 phase, in which the index entries (which belong to the logical partitions being reorganized) in the original data sets are replaced by the index entries in the shadow data sets.

**Note:** If you issue an online REORG for specific partitions only, without indexes, the instance nodes of the NPIs do not change - even if you use the Fast SWITCH method.

***BUILD2 Parallelism***
With DB2 V7, DB2 introduces BUILD2 Parallelism, that is, DB2 dispatches several subtasks, optimally one for each logical partition, to perform the updates of the entries in the original NPI data set(s) by the entries from the original NPI data set(s).

***Degree of parallelism***
The number of parallel subtasks is governed by:

- Number of CPUs:
- Number of available DB2 threads
- Number of logical partitions
- Utility ZPARM '&SRPRMMUP'

  **Notes:**

  - This value can be set up to 99.
  - In contrast to the COPY utility, in which you can override this value by the option `PARALLEL integer`, there is no keyword in the REORG utility syntax to govern the degree of the BUILD2 Parallelism.

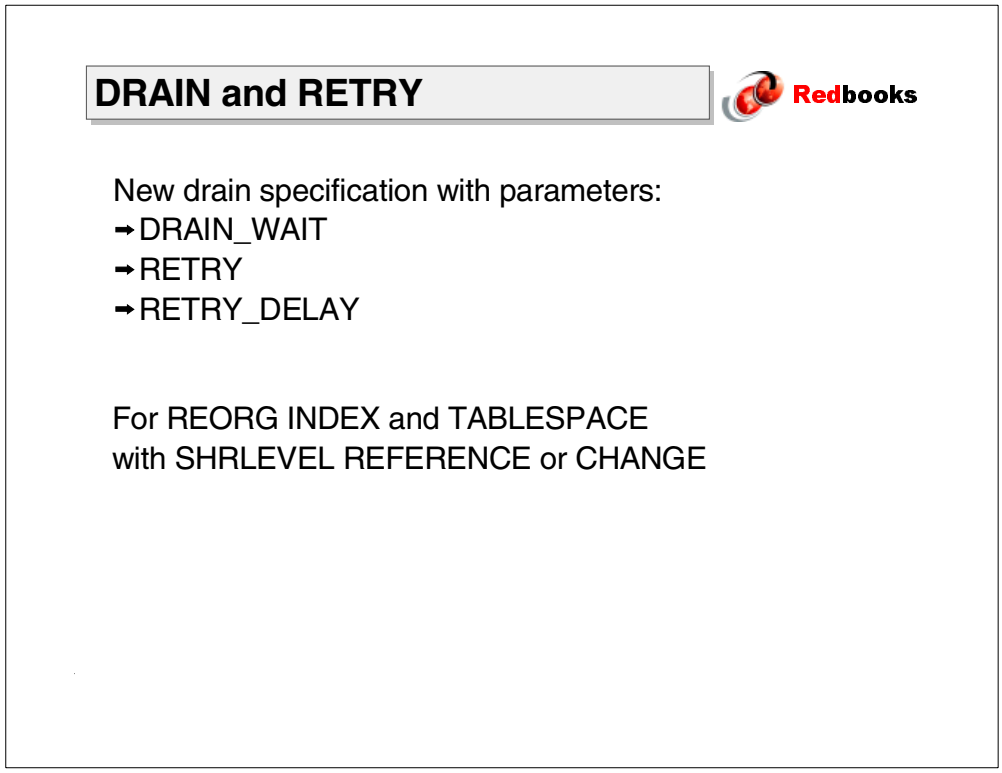
Optimally, if you have one subtask for each logical partition, the elapsed time of the BUILD2 phase could be the time it takes to process the logical partition with the most RIDs. This implies improvements for all cases with more than one NPI.

***Documentation of the parallelism***
Subphase information is provided for each subtask in the command output of:

```
-DIS UTIL

DSNU111I csect-name - SUBPHASE=subphase-name COUNT=n
Explanation: There will be one of these messages for each subtask.
This message gives the user an estimate of how much
processing the utility subtask has completed.
SUBPHASE subphase-name identifies the activity that the
subtask was performing at the time the -DISPLAY UTILITY command was issued.
COUNT n is the number of records processed by the utility subtask.
```

Another helpful message:

```
DSNU1114I csect-name LOGICAL PARTITIONS WILL BE LOADED IN PARALLEL,
NUMBER OF TASKS = nnnn
Explanation: This message is issued by the BUILD2 phase of the REORG utility.
When the REORG utility updates logical partitions in parallel, the number
of utility tasks used to update logical partitions is indicated by nnnn.
```




### 5.7.5 DRAIN and RETRY

When executing an Online REORG utility, both for REORG INDEX and REORG TABLESPACE, a new drain specification statement can be added, overriding the utility locking timeout value specified at subsystem level in the DSNTIPI installation panel with the values for Resource Timeout, IRLMRWT, and the multiplier Utility Timeout, UTIMOUT. This new specification offers granularity at REORG invocation level rather than at DB2 subsystem level. A shorter wait reduces the impact on applications and can be combined with retries to increase the chances of completing the REORG execution.

- DRAIN_WAIT n

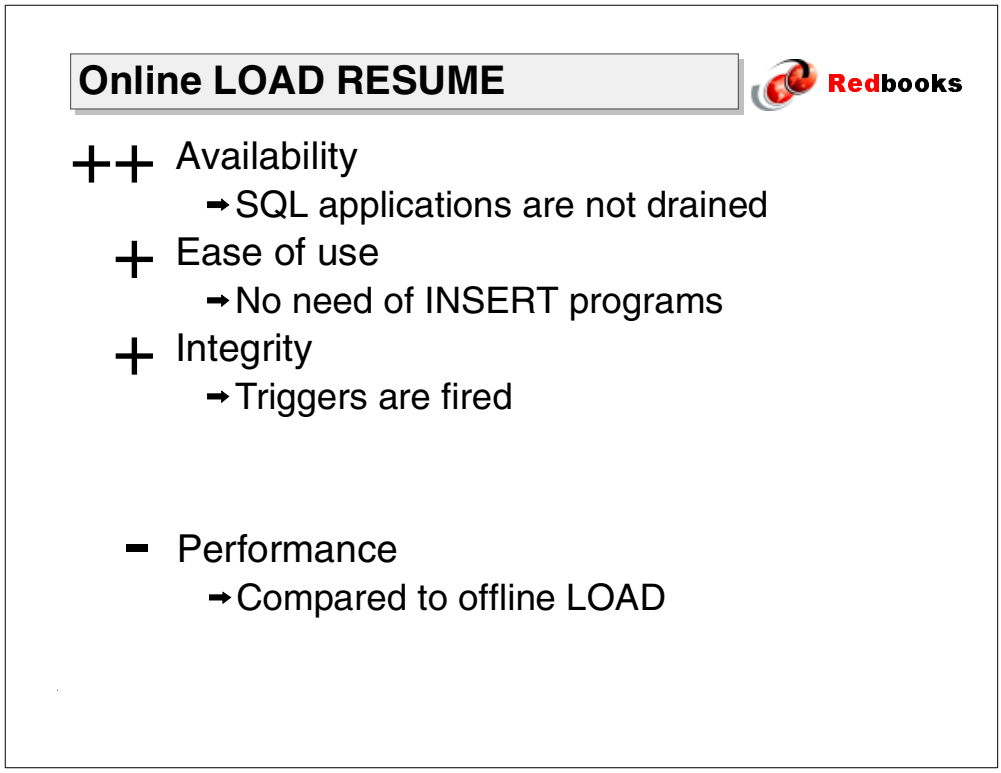
  It specifies the maximum number of seconds that the utility will wait when draining. The time specified in the integer n is the aggregated time for the table space and the associated indexes. The range of seconds allowed is between 0 and 1800. If the parameter is omitted or 0 is specified, the drain wait reflects the values specified for IRLMRWT and UTIMOUT.

- RETRY n

  It specifies the maximum number of times that the drain will be attempted, with the integer n assuming values from 0 up to 255. If omitted or 0 is specified, no retries are attempted.

- RETRY_DELAY n

  It works in conjunction with RETRY and specifies the minimum duration in seconds between retries. The range is 1 to 1800 seconds with a default of 300 seconds.




## 5.8 Online LOAD RESUME

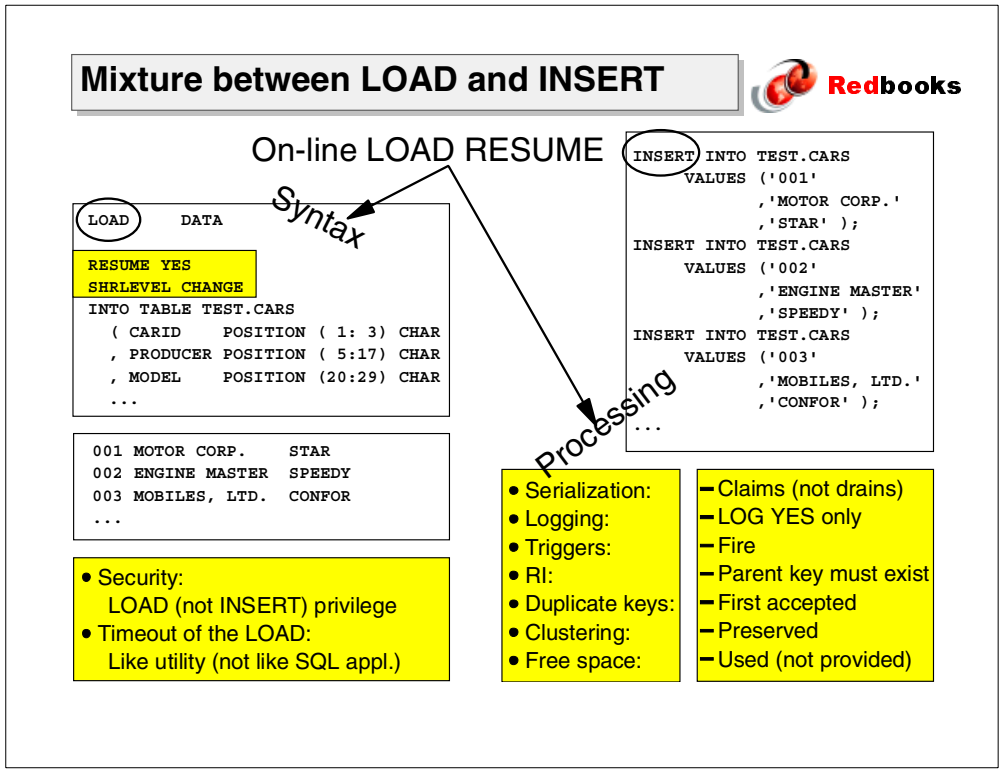
The classic LOAD drains all access to the table space, therefore, the data is not available to SQL applications. A possible solution is to write INSERT programs replacing the LOAD. In this way you can avoid the drain and keep the data available to the SQL applications. Maintaining hundreds of INSERT programs is expensive, therefore, DB2 V7 introduces this new option for LOAD. It is almost a utility of its own, which behaves externally like a LOAD, but it works internally like a mass INSERT.

In brief, with this new Online LOAD RESUME, you can load data with minimal impact on SQL applications and without writing and maintaining INSERT programs.

***Integrity***
Data integrity cannot be always assured by only using foreign keys. In some cases, triggers are additionally used to ensure the correctness of the data. The classic LOAD does not activate triggers, which then is a data integrity exposure. The new Online LOAD RESUME functionally operates like SQL INSERTs, therefore, triggers are activated.

***Performance***
As the new Online LOAD RESUME internally works like a SQL INSERTs, this kind of LOAD is slower than the classic LOAD.

But many customers are willing to trade off performance for availability, especially for data warehouse applications, where queries may run for several hours.




### 5.8.1 Mixture between LOAD and INSERT

The LOAD statement offers a new option, SHRLEVEL, with two possible parameters. SHRLEVEL NONE invokes the classic LOAD; SHRLEVEL CHANGE requires RESUME YES and LOG YES and invokes a completely different processing: Although the syntax is like the classical LOAD (and the input is provided as before), internally each input record is stored into the table space via a mechanism comparable to an individual INSERT.

Even though full INSERT statements are not generated, LOAD RESUME YES SHRLEVEL CHANGE will functionally operate like SQL INSERT statements as the ones on the foil. Whereas the classic LOAD drains the table space, thus inhibiting any SQL access, these INSERTs act like normal INSERTs by using claims when accessing an object. That is, they behave like any other SQL application and can run concurrently with other, even updating, SQL applications. Therefore, this new feature is called *online LOAD RESUME*.

***Consequences of INSERT processing***

These are some of the consequences of INSERT processing:

- Logging

  Only LOG YES is allowed. Therefore, no COPY is required afterwards.

- RI

  When you load a self-referencing table with *online LOAD RESUME*, the foreign key value in each input record:

  - Must already exist as primary key value in the table
  - Must be provided as primary key value within this record

  This is different from the classical LOAD, and it forces you to sort the input in such a way, that these requirements are met, rather than sorting the input in clustering index sequence, which you used to do.

- Duplicate keys

  When uniqueness of a column is required, INSERTs are accepted as long as they provide different values for this column. Following INSERTs with the same values are not accepted. This is different from the classic LOAD procedure, which discards all records having the same value for such a column.

  You may be forced to change your handling of the discarded records accordingly, when you change existing LOAD jobs to SHRLEVEL CHANGE.

***Clustering***

Whereas the classic LOAD RESUME stores the new records (in the sequence of the input) at the end of the already existing records; the new *online LOAD RESUME* tries to insert the records in available free pages as close to clustering order as possible; additional free pages are not created. As you probably insert a lot of rows, these are likely to be stored out of the clustering order (OFFPOS records).

So, a REORG may be needed after the classic LOAD, as the clustering may not be preserved, but also after the new *online LOAD RESUME*, as OFFPOS records may exist. A RUNSTATS with SHRLEVEL CHANGE UPDATE SPACE followed by a conditional REORG is recommended.

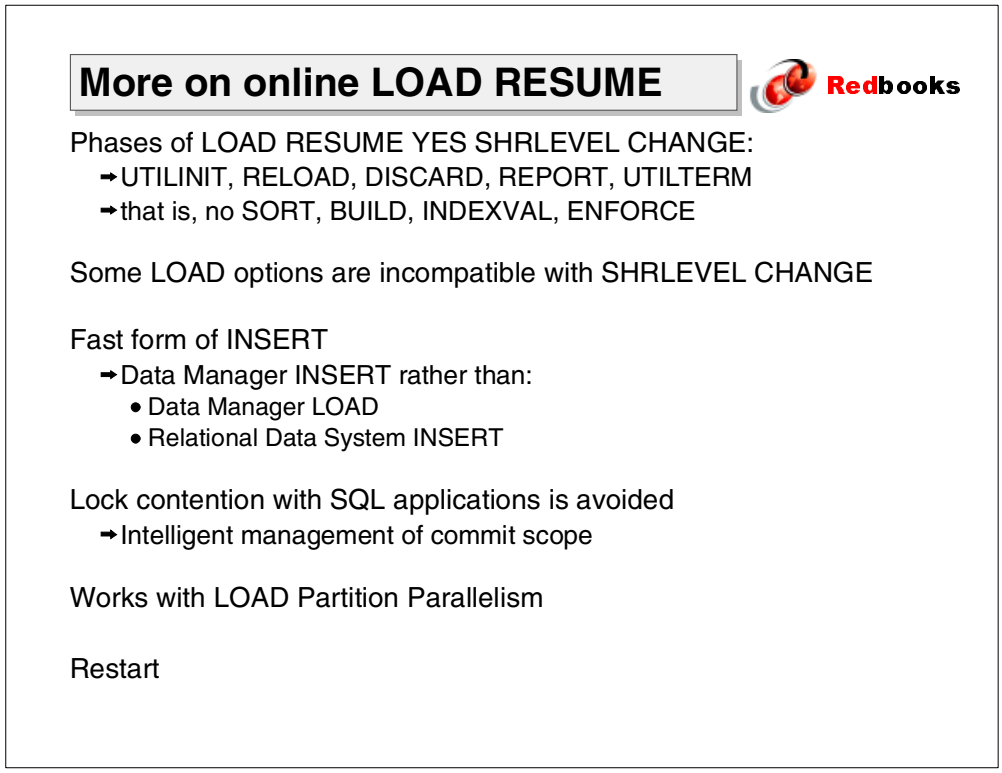
***Free space***

Furthermore the free space, obtained either by PCTFREE or by FREEPAGE, is used by these INSERTs of the Online LOAD RESUME - in contrast to the classical LOAD, which loads the pages thereby providing these types of free space.

As a consequence, a REORG may be needed after an Online LOAD RESUME.

***Messages***

The messages that can be issued during the utility execution are the same as for the LOAD utility, even though the data is accessed differently.




### 5.8.2 More on Online LOAD RESUME

Here are some more considerations regarding Online LOAD RESUME.

***Phases***
Some phases are obviously not included, as this kind of LOAD operates like SQL INSERTs. But the DISCARD and the REPORT phase are still performed, therefore, errors are handled similar to a classic LOAD:

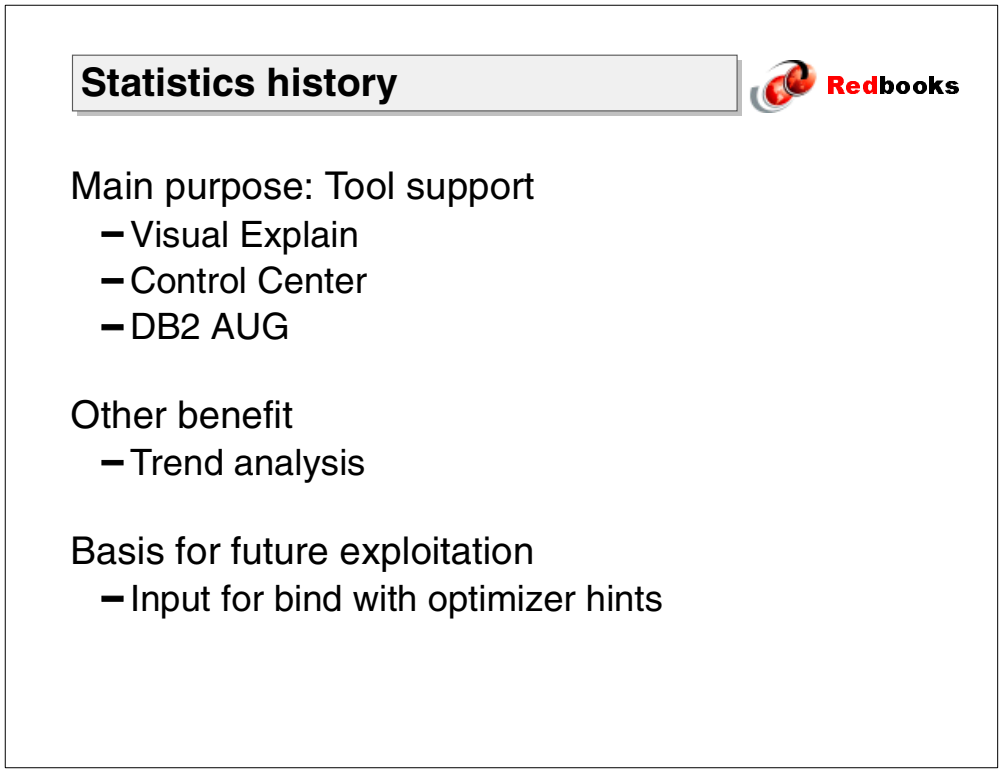
- Input records which fail the insert will be written to the discard data set
- Error information is stored in the error data set.

***Incompatible LOAD options***
SHRLEVEL CHANGE is incompatible with LOG NO, ENFORCE NO, KEEPDICTIONARY, SORTKEYS, STATISTICS, COPYDDN, RECOVERYDDN, PREFORMAT, REUSE, and PART REPLACE.

***Data Manager rather than RDS INSERT***
The difference is, that the Data Manager will not generate full flavored INSERT statements: The SQL overhead is omitted, that is, there is no SQL prepare, nor authorization checking for every INSERT statement.

***Lock contention is avoided***
DB2 manages the commit scope dynamically monitoring the current locking situation. (The details are proprietary.)

***Restart***
During RELOAD, internal commit points are set, therefore, RESTART(CURRENT) is possible as with the classic LOAD.




## 5.9 Statistics history

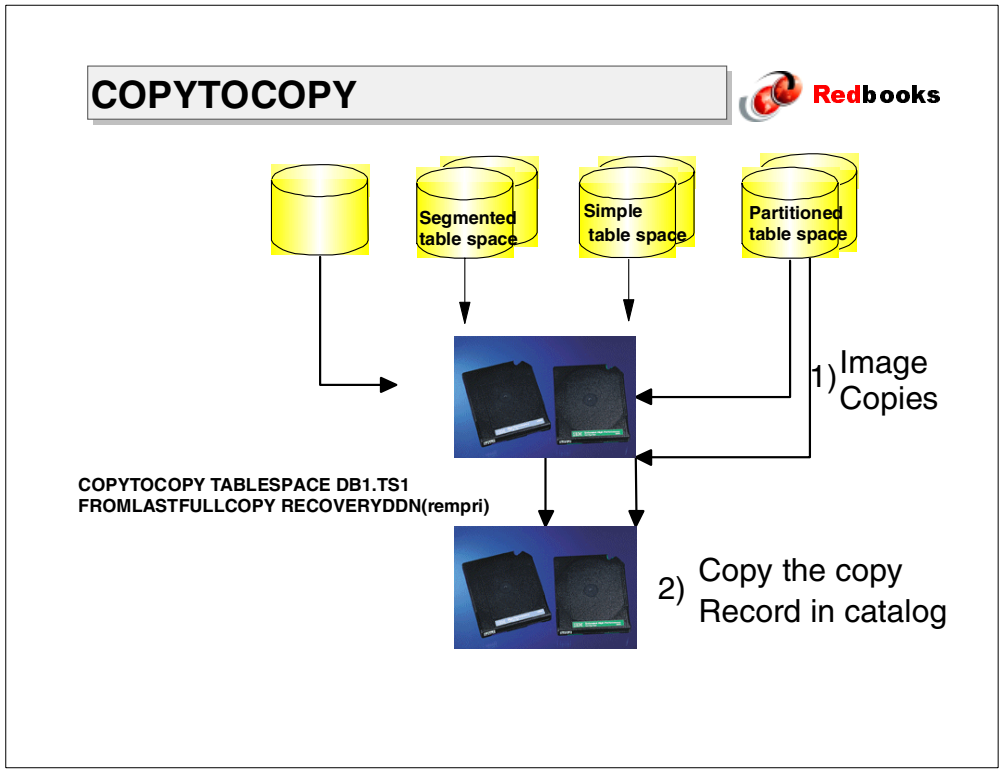
***Tool support***
Visual Explain explains statements that were possibly bound at a time when the statistics in the catalog were different. These explanations are difficult to understand if the statistics no longer match, and Visual Explain cannot work properly when only the current statistics are available. With statistics history, Visual Explain works better and it is easier to use.

Note that DB2 for the non-390 platforms stores all statistics information in the Bind file, therefore, Visual Explain can work, but these old statistics are hidden and not made available to other users. DB2 V7 takes another approach: with statistics history these data are generally available.

***Trend analysis***
With this history, you can query the development of some characteristics of your data, for example, PAGESAVE (compression still OK), LEAFDIST (too many page splits occurred?)

***Input for optimizer hints***
It is conceivable that you will be able to bind a package based on previous statistics.

This can be activated, like Runstats, during the execution of other utilities (such as Rebuild, Load, and Reorg)




## 5.10 CopyToCopy

DB2 V7 introduces a new utility: COPYTOCOPY.

It provides you with the opportunity to make additional full or incremental image copies, duly recorded in SYSIBM.SYSCOPY, from a full or incremental image copy that was taken by the COPY utility. This applies to table spaces or indexes. The maximum number of additional copies you are allowed to make is three out of the possible total of four copies; they are local primary, local backup, recovery site primary, and recovery site backup.

It is suitable for executing the extra copies asynchronously from the normal batch stream and it is mostly beneficial for remote copies on slow devices. COPYTOCOPY leaves the target object in read write access (UTRW), and that allows other utilities and SQL statements to run concurrently with the same target objects. This is not allowed for utilities that insert or delete records in SYSIBM.SYSCOPY: namely COPY, LOAD, MERGECOPY, MODIFY, RECOVER, QUIESCE and REORG utilities or utilities with SYSIBM.SYSCOPY as the target object.

The SYSIBM.SYSCOPY columns, ICDATE, ICTIME, START_RBA will be those of the original entries in the SYSIBM.SYSCOPY row when the COPY utility recorded them. While the columns DSNMAE, GROUP_MEMBER, JOBNAME, and AUTHID will be those of the COPYTOCOPY job execution.

An example of the COPYTOCOPY statement is:

```
COPYTOCOPY TABLESPACE DB1.TS1 FROMLASTFULLCOPY RECOVERYDDN(rempri)
```

The possible options are:

- FROMLASTCOPY
- FROMLASTFULLCOPY
- FROMLASTINCRCOPY
- FROMCOPY

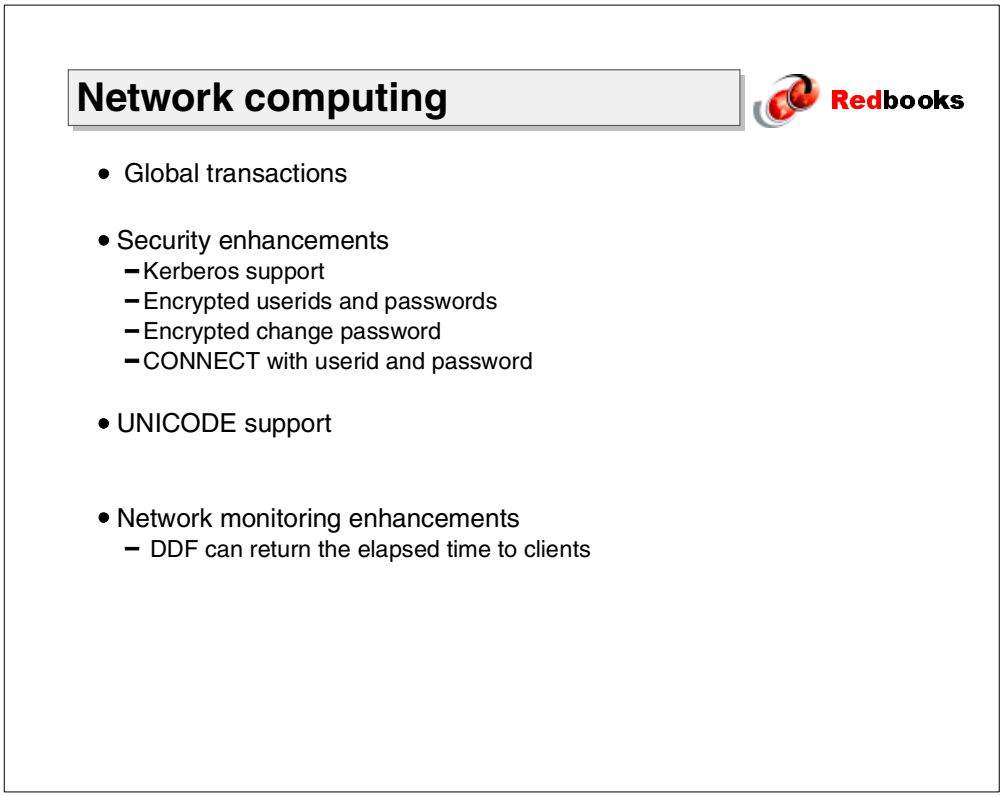
The COPYTOCOPY utility does not apply to the following catalog and directory objects:

- DSNDB01.SYSUTILX, and its indexes
- DSNDB01.DBD01, and its indexes
- DSNDB06.SYSCOPY, and its indexes.

# Part 4. Network computing

© Copyright IBM Corp. 2001

# Chapter 6. Network computing




DB2 V7 introduces a number of enhancements to improve the compatibility and usability of DB2 for OS/390 and z/OS in a network computing environment.

***Global transactions***

DB2 V7 base code provides the same support for global transactions as is shipped by APAR into DB2 UDB for OS/390 Version 6. Briefly, you can develop an application to use multiple DB2 agents or threads to perform processing that requires coordinated commit processing across all the threads. DB2, via a transaction processor, treats these separate DB2 threads as a single “global transaction” and commits all or none. Refer to the redbook *DB2 UDB for OS/390 Version 6 Technical Update,* SG24-6108, for a more detailed explanation of global transactions and how they work.

***Security enhancements***

A number of enhancements have been made in Security to support Network Computing:

- DB2 V7 provides server-only support for Kerberos authentication. This enhancement requires the OS/390 Kerberos security support, which is available in OS/390 Version 2 Release 10.
- The current DCE security support within DB2 UDB for OS/390 and z/OS Version 7 is removed.

© Copyright IBM Corp. 2001

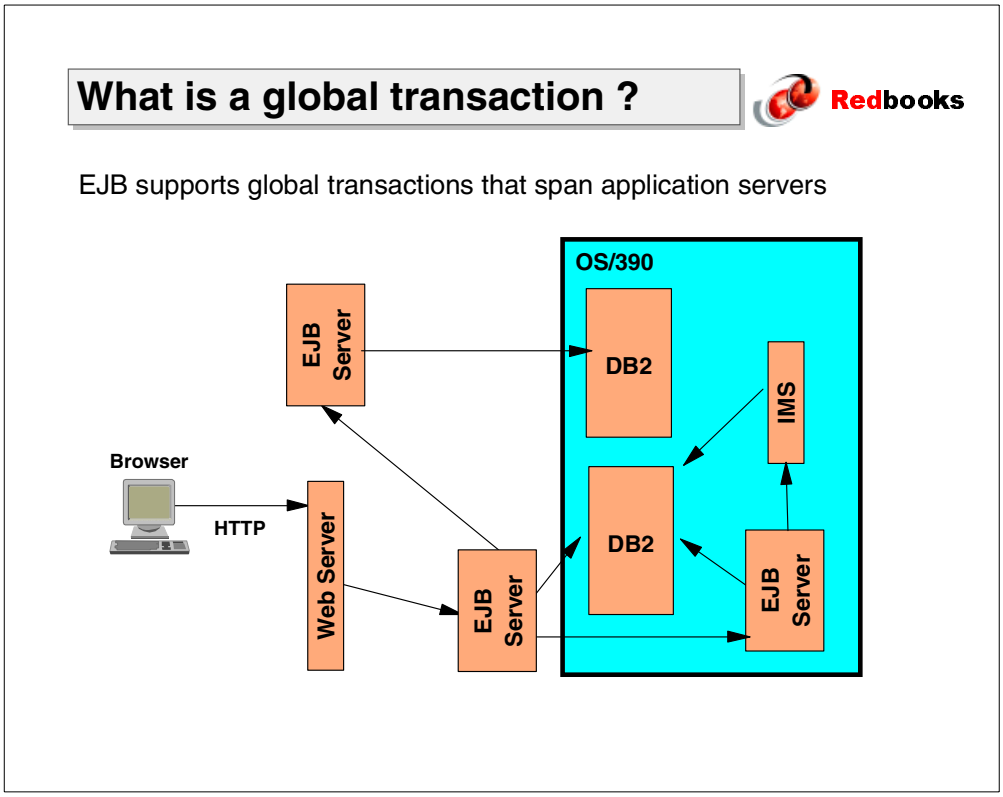
- DB2 V7 provides server support for both encrypted userids and passwords. Support for encrypted passwords was introduced by APAR in DB2 UDB for OS/390 Versions 5 and 6. The encrypted support has also been extended to support the change password function.
- You can now specify an optional userid and password when using the CONNECT statement to connect to DB2.

***UNICODE support***
DB2 V7 introduces full support for a third encoding scheme, UNICODE.

The UNICODE encoding standard is a single character encoding scheme that can include characters for almost all the languages in the world.

With this enhancement, DB2 V7 can truly support multinational and e-commerce applications, by allowing data from more that one country/language to be stored in the same DB2 subsystem.

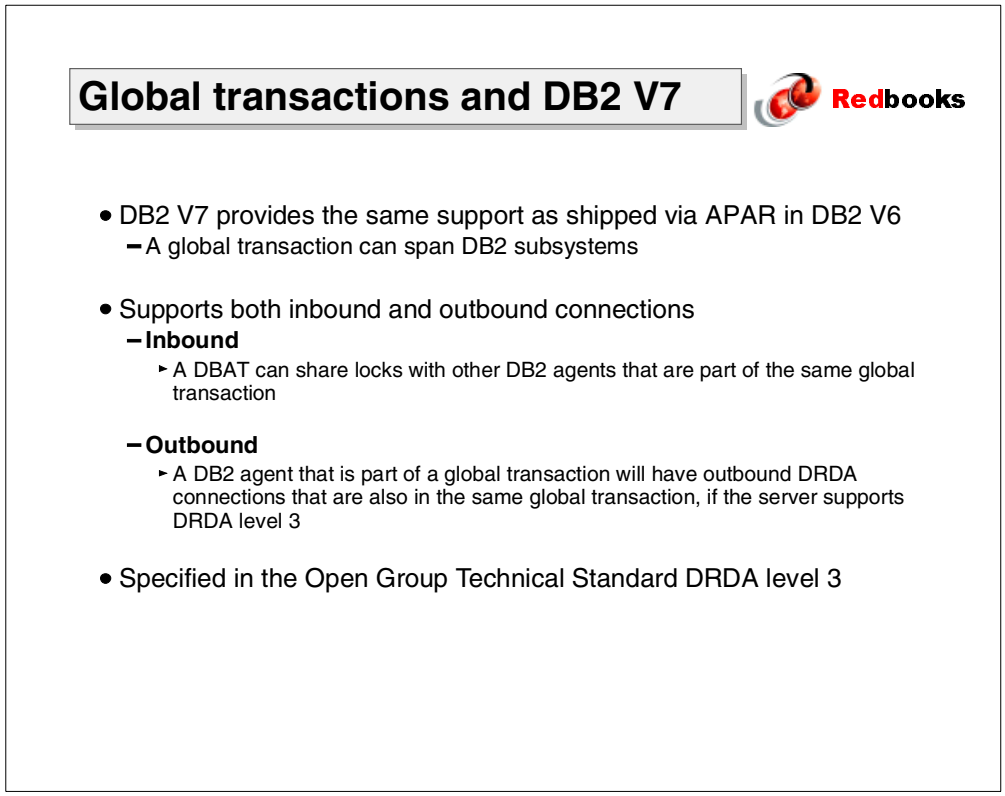
***Network monitoring enhancements***
Applications accessing a DB2 V7 server using DB2 Connect can now monitor the server's elapsed time using the System Monitor Facility. This enhancement makes it easier to monitor bottlenecks in client/server applications, by helping to easily identify where the bottleneck is.




## 6.1 Global transactions

One of the challenges that Enterprise Java Beans (EJBs) presents is the concept of *global transactions*. In this foil we have a Web server that invokes an EJB application on an EJB server. This server in turn invokes EJBs at two other servers and it also issues SQL directly to DB2 using JDBC or SQLJ. The other servers also interact with DB2 using JDBC or SQLJ. Lastly, one of the EJB servers invokes an IMS transaction that issues SQL using IMS Attach.

All these DB2 transactions can be part of the same global transaction. DB2 has been enhanced to recognize global transactions and allow the individual DB2 transactions to share locks across branches of a global transaction. DB2, via a transaction processor (WebSphere through RRS in our example), also commits these DB2 threads as single unit of work, (that is: “all or none”). Refer to the redbook *DB2 UDB for OS/390 Version 6 Technical Update,* SG24-6108, for a detailed discussion of what global transactions are and how they are implemented in DB2 for OS/390.




***Global transactions and DB2 V7***

APAR PQ28487/UQ34479 adds a global transaction support to DB2 UDB for OS/390 Version 6. APAR PQ32387/UQ37920 extends the support for global transactions into the distributed environment.

DB2 V7 base code provides the same support for global transactions as is shipped by APAR PQ32387 into DB2 V6. DB2 UDB for OS/390 sometime refers to the feature as *distributed global transaction* support, to highlight the fact that transactions in different DB2 subsystems, connected by DRDA, can be a part of the same global transaction.

Global transaction support requires a sync point coordinator or transaction manager that will coordinate commit operations among the various DB2 agents via a 2-phase commit protocol. A global transaction is identified by an identifier called an XID. Resources are always shared between DB2 agents that have the same XID.

Global transaction support has no effect on commit operations in that when multiple DRDA connections are part of the same global transaction, all connections must successfully complete prepare processing before the decision to commit can be made.

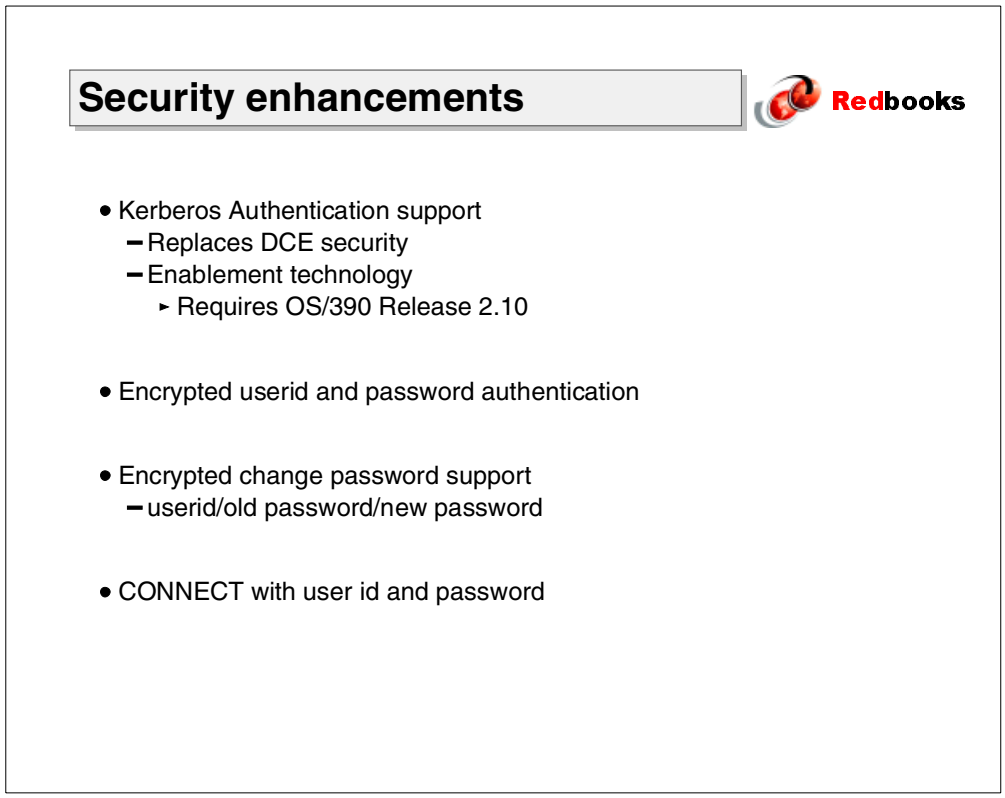
There is no support for a global transaction across members of a data sharing group. DB2 agents running on different members of the group can deadlock if they access the same data, even though they are part of the same global transaction.

The Open Group Technical Standard DRDA level 3 allows an XID to be transmitted along with the first SQL statement processed in a new connection or follows after a syncpoint operation completes. Any DBMS that supports the Open Group Technical Standard DRDA level 3 can distribute global transactions across DBMSs. (The Open Group Technical Standard DRDA level 3 is planned to be implemented by DB2 V7, DB2 Connect Version 7 and DB2 for UNIX, Windows, OS/2 Version 7).

**Note:** DRDA is now an open standard, administered by the Open Group, as a part of the Open Group Technical Standards. Refer the following Web site for more information on the Open Group and the Open Group Technical standard DRDA:

`http://www.opengroup.org`

OS/390 WebSphere Version 4, executing Java Beans and using JDBC shipped with DB2 V7, is able to use distributed global transactions. Here, OS/390 Resource Recovery Service (RRS) acts as the transaction coordinator. Refer to 3.4.6, “JDBC 2.0 distributed transactions” on page 119, for a discussion of JDBC support for DB2 global transactions.




## 6.2 Security enhancements

DB2 V7 introduces a number of security enhancements to support network computing

***Kerberos authentication support***
IBM and the Open Software Foundation (OSF) adopted the Distributed Computing Environment (DCE) authentication mechanism as a network security standard. DB2 for OS/390 implemented support for DCE in Version 5. DCE authentication is very similar to Kerberos, but was intended to be an improved extension to Kerberos. IBM has now adopted Kerberos security as a replacement standard for DCE.

Since DCE is no longer the standard and is not being used, the current DCE support within DB2 V7 will be removed. Refer to Chapter 12, “Migration and fallback” on page 495, for a discussion on migration/fallback considerations for Kerberos support.

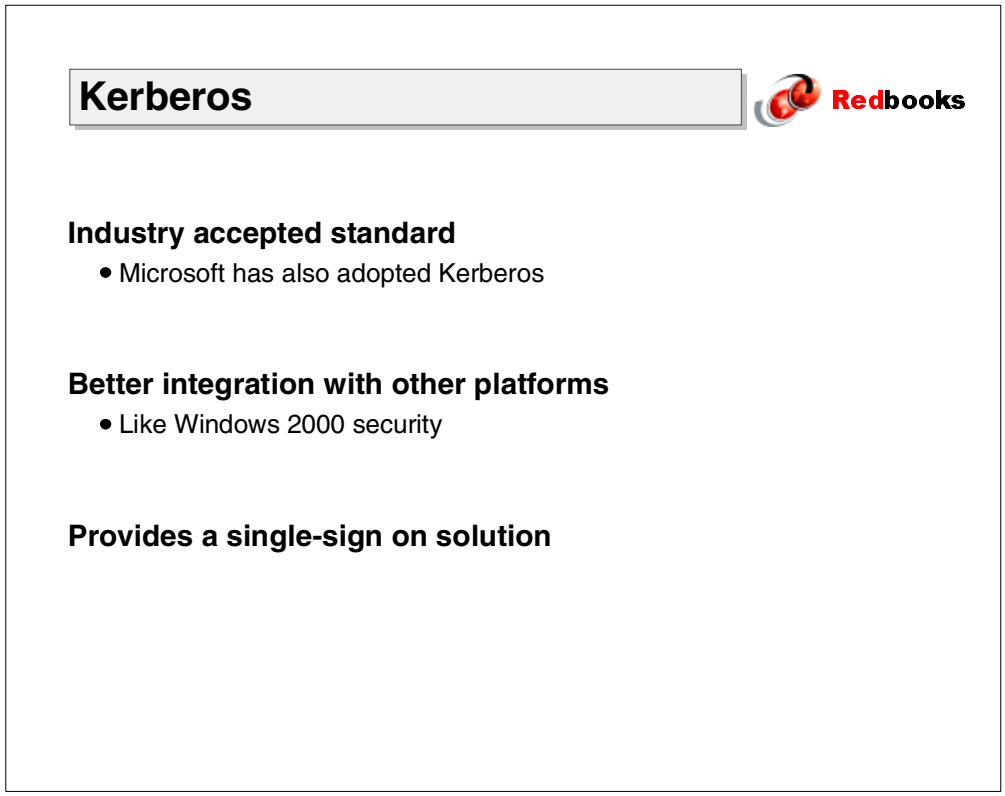
OS/390 support for Kerberos is available starting with OS/390 Version 2 Release 10. DB2 V7 will provide support for Kerberos authentication by utilizing this new OS/390 support. The OS/390 support is through the OS/390 SecureWay Security Server Network Authentication Privacy Service, and OS/390 SecureWay (RACF). The Network Authentication Privacy Service provides Kerberos support and relies on a security product such as RACF to provide registry support.

***Encrypted userid and password authentication***
DB2 V7 accepts encrypted userids in addition to encrypted passwords. (Support for encrypted passwords was implemented by APAR in DB2 UDB for OS/390 Version 5 and Version 6.)

***Encrypted change password support***
DB2 V7 provides server support for encrypted change passwords. Now the *userid/old password/new password* are all encrypted when sent to the host.

***CONNECT with userid and password***
You can now connect from an application running on OS/390, to a DB2 V7 DRDA server, with a userid and password. DB2 Connect already prompts you for a userid and password when you are connecting to a DB2 UDB for OS/390 server.

.




### 6.2.1 Kerberos

DB2 V7 implements Kerberos authentication as a replacement standard for DCE.

Kerberos is an industry accepted authentication standard, with a number of vendors implementing Kerberos solutions on a variety of platforms. For example, Microsoft has adopted Kerberos security for enterprise security in Windows 2000.

Kerberos support provides a better integration with Windows 2000 security and provides a single-sign on solution for new applications. For example, when you sign on to a Windows 2000 workstation you do not need to provide a host userid and password when using applications that access DB2 for OS/390 database servers.

## What is Kerberos ?

- Authentication mechanism for network security
- Developed by MIT
- Similar to DCE
  - Flows encrypted tickets instead of 'clear text' userids and passwords
- More information
  - http://web.mit.edu/kerberos/www/
  - http://web.mit.edu/kerberos/www/dialogue.html

### 6.2.1.1 What is Kerberos?

Kerberos is an authentication mechanism for network security, designed to provide users and applications with secure access to data, resources and services located anywhere on a heterogeneous network. Its purpose is to allow user authentication over a physically untrusted network.

Kerberos was developed by the Massachusetts Institute of Technology (MIT) and named from Greek mythology (from the story about a three-headed dog that guarded the gates of Hades).

Kerberos is a distributed authentication service that allows a process (a client) running on behalf of a principal (a user) to prove its identity to a verifier (an application server, or simply a server) without sending data across the network that might allow an attacker or the verifier to subsequently impersonate the principal.

Kerberos uses encrypted *tickets* instead of flowing userids and passwords “in the clear” over the network. Tickets are issued by a Kerberos Authentication Server (KAS). Both clients and servers must have keys registered with the Kerberos authentication server. In the case of the client, the key is derived from the clients user supplied password.

Kerberos can also optionally provide integrity and confidentiality for data sent between the client and server.

In the following sections we provide a short introduction to Kerberos, however, if you want to get a better understanding of Kerberos and the underlying DES (Data Encryption Standard) algorithm, you can reference these Web sites:

```
http://web.mit.edu/kerberos/www/
http://web.mit.edu/kerberos/www/dialogue.html
```

***Kerberos encryption***

Though conceptually, Kerberos authentication proves that a client is running on behalf of a particular user, a more precise statement is that the client has knowledge of an encryption key that is known by only the user and the authentication server. In Kerberos, the user's encryption key is derived from and should be thought of as a password. Similarly, each application server shares an encryption key with the authentication server, known as the server key.

Encryption in the present implementation of Kerberos uses the data encryption standard (DES). It is a property of DES that if ciphertext (encrypted data) is decrypted with the same key used to encrypt it, the plaintext (original data) appears. If different encryption keys are used for encryption and decryption, or if the ciphertext is modified, the result will be unintelligible, and the checksum in the Kerberos message will not match the data. This combination of encryption and the checksum provides integrity and confidentiality for encrypted Kerberos messages.

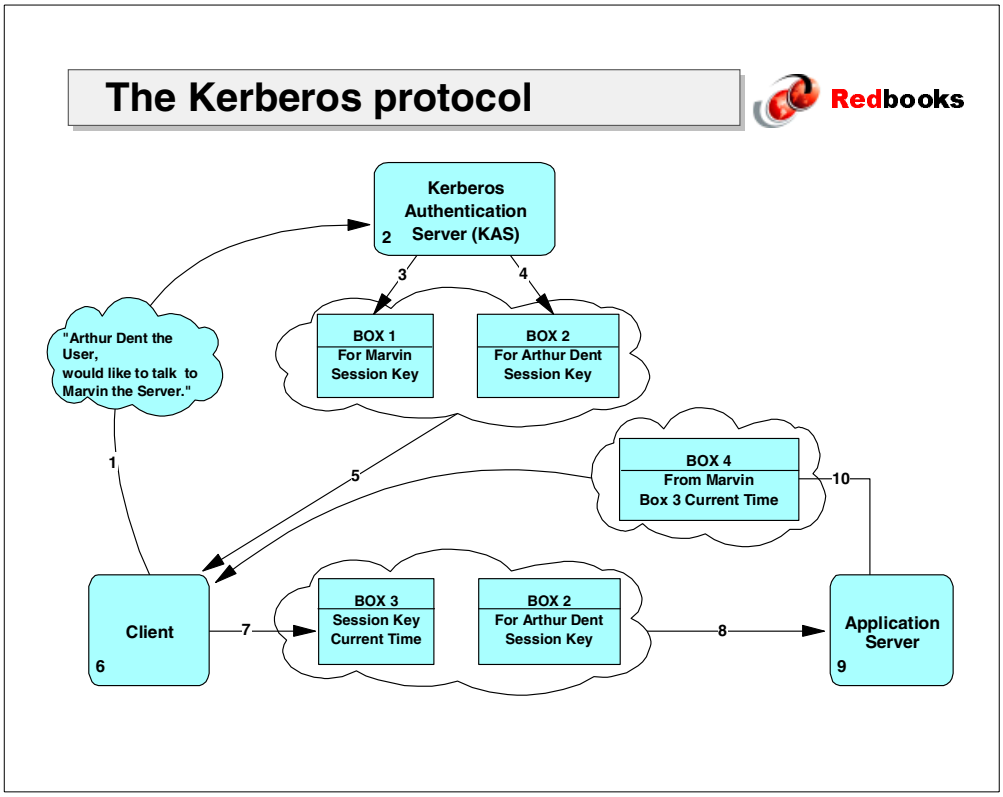
***The Kerberos ticket***

The client and server do not initially share an encryption key. Whenever a client authenticates itself to a new server it relies on the authentication server to generate a new encryption key and distribute it securely to both parties. This new encryption key is called a session key and the Kerberos ticket is used to distribute it to the server.

The Kerberos ticket is a certificate issued by a Kerberos authentication server (KAS), encrypted using the server key. Among other information, the ticket contains the random session key that will be used for authentication of the principal to the server, the name of the principal to whom the session key was issued, and an expiration time after which the session key is no longer valid. The ticket is not sent directly to the server, but is instead sent to the client who forwards it to the server as part of the application request. Because the ticket is encrypted in the server key, known only by the authentication server and intended server, it is not possible for the client to modify the ticket without detection.

***Kerberos and DB2 UDB for OS/390 and z/OS***

From a DB2 for OS/390 perspective, it really is not necessary to understand Kerberos design and protocol in great detail, since the OS/390 Kerberos support is provided by the OS/390 SecureWay (RACF). Basically, the Kerberos protocol is transparent to DB2 for OS/390.




### 6.2.1.2 The Kerberos protocol

It is through the exchange described below that a client proves his identity to the server, and optionally the server proves its identity to the client. There are two parts to the application request, a *ticket* and an *authenticator*. The *authenticator* includes, among other fields: the *current time*, a *checksum*, and an *optional encryption key*, all encrypted with the session key from the accompanying ticket.

Here is an overview of how the Kerberos protocol works.

**Note:** The following example is based on the article by Brian Tung, “The Moron's Guide to Kerberos”, which can be found at:

`http://www.isi.edu/gost/brian/security/kerberos.html`

Both the client and the application server have keys registered with the Kerberos Authentication Server (KAS). The client's key is derived from a password that has been chosen by the client’s user. The service key is a randomly selected key (since no user is required to type in a password).

For the purposes of this explanation, let us imagine that messages are written on paper (instead of being electronic), and are “encrypted” by being locked in a box by means of a key. In this scenario, clients are initialized by making a physical key and registering a copy of the key with the Kerberos Authentication Server (KAS).

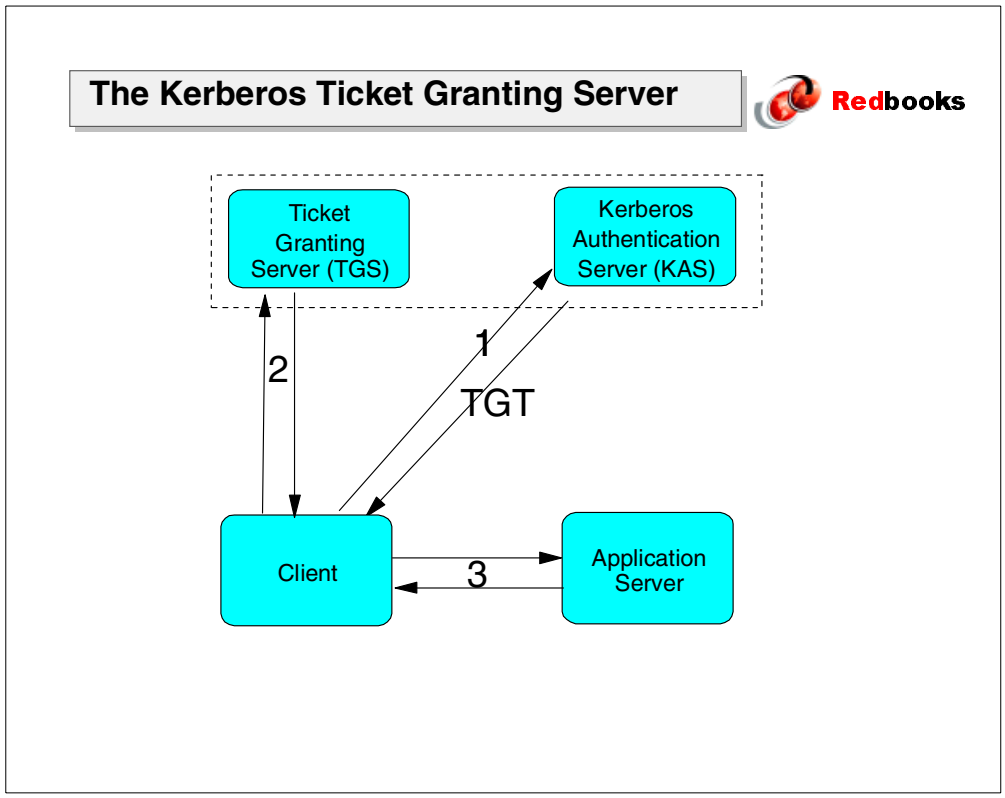
1. The client first sends a message to the KAS: “Arthur Dent the User, would like to talk to Marvin the Server.''
2. When the KAS receives this message, it makes up two copies of a brand new key. This is called the session key. It will be used in the direct exchange between the client and server.
3. The KAS puts one of the session keys in Box 1, along with a piece of paper with the name ``Marvin Server'' written on it. It locks this box with the user's key

   Why is this piece of paper here? Recall that this box is really just an encrypted message, and that the session key is really just a sequence of random bytes. If Box 1 only contained the session key, then the client wouldn't be able to tell whether the response came back from the KAS, or whether the decryption was successful. By putting in ``Marvin Server,'' the client will be able to verify both that the box comes from the KAS, and that the decryption was successful.
4. The KAS puts the other session key in a Box 2, along with a piece of paper with the name “Arthur Dent the User'' written on it. It locks this box with the server's key.
5. The KAS returns both boxes to the client.
6. The client unlocks Box 1 with his key, extracting the session key and the paper with ``Marvin Server'' written on it.
7. The client cannot open Box 2 (since it is locked with the server's key). Instead, he puts a piece of paper with the current time written on it in Box 3, and locks it with the session key.
8. The client then hands both Box 2 and Box 3 to the server.
9. The server opens the Box 2 with its own key, extracting the session key and the paper with “Arthur Dent the User'' written on it. It then opens Box 3 with the session key to extract the piece of paper with the current time on it. These items demonstrate the identity of the client.

   The timestamp is put in Box 3 to prevent someone else from copying Box 2 (remember, these are simply electronic messages) and using it to impersonate the client at a later time. Because clocks do not always work in perfect synchrony, a small amount of leeway (about five minutes is typical) is given between the timestamp and the current time. In addition, the server maintains a list of recently sent authenticators, to make sure that they are not resent immediately.
10. Sometimes, the client may want the server to be authenticated in return. To do so, the server takes the timestamp from the *authenticator* (Box 3), places it in Box 4, along with a piece of paper with “Marvin Server'' written on it, locks it with the session key, and returns it to the client. Clearly, it must include something with the timestamp; otherwise, it could simply return Box 3.

You may wonder how the server is able to open Box 3, if there is not anyone to type in a password. Well, the servers key is not derived from a password. Instead, it is randomly generated, then stored in a special file called a `service key file`. This file is assumed to be secure, so that no one can copy the file and impersonate the service to a legitimate user.

In Kerberos terminology, Box 2 is called the *ticket*, and Box 3 is called the *authenticator*. The *authenticator* typically contains more information than what is listed in this section. Some of this added information arises from the fact that this is an electronic message (for example, there is a checksum). There may also be an encryption key in the *authenticator* to provide for privacy in future communications between the client and the server.




### 6.2.1.3 The Kerberos Ticket Granting Server

There is a subtle problem with the previous exchange. It is used everytime a client wants to contact a server. Therefore, this exchange implies that the client must enter in a password (unlock Box 1 with the key) each time. The obvious way around this is to cache the key derived from the password. But caching the key is dangerous. With a copy of this key, an attacker could impersonate the client at any time (until the password is next changed).

Kerberos resolves this problem by introducing a new agent, called the Ticket Granting Server (TGS). The TGS is logically distinct from the KAS, although they may reside on the same physical machine.

The function of the TGS is as follows:

1. Before accessing any regular server, the client requests a ticket to contact the TGS, just as if it were any other server. This ticket is called the Ticket Granting Ticket (TGT).
2. After receiving the TGT, any time that the client wishes to contact a server, he requests a ticket, not from the KAS, but from the TGS. Furthermore, the reply is encrypted, not with the client's secret key, but with the session key that the KAS provided for use with the TGS. Inside that reply is the new session key for use with the regular service.
3. The rest of the exchange now continues as described above.

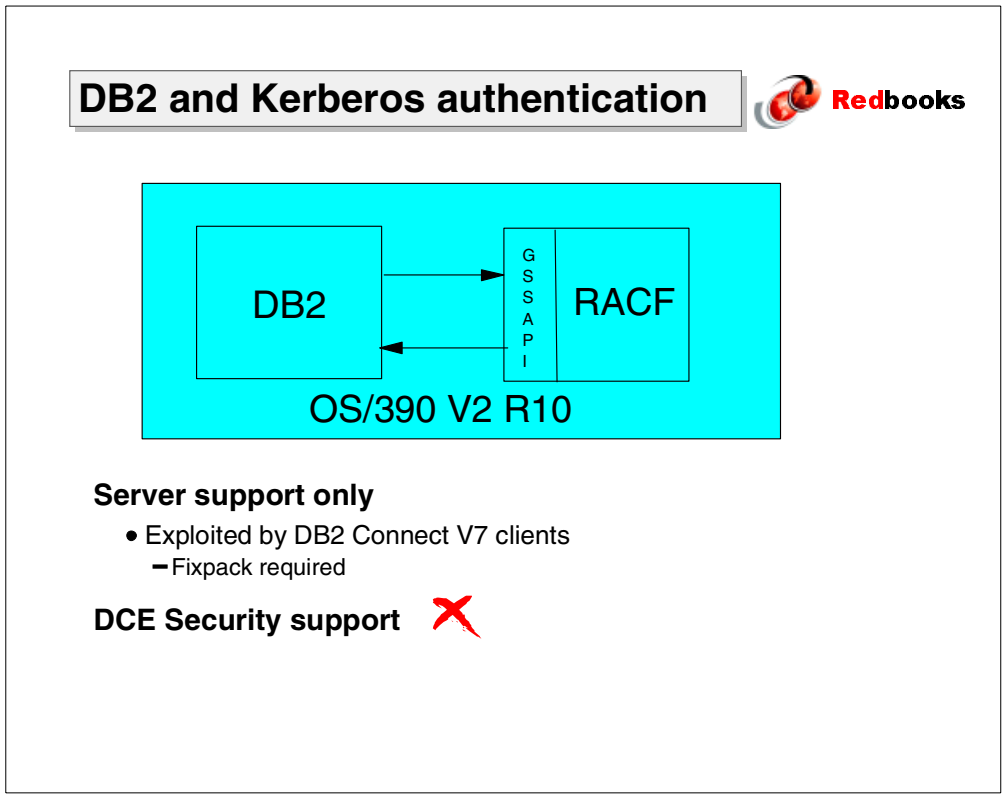
The advantage this provides is that while passwords usually remain valid for months at a time, the TGT is good only for a fairly short period, typically eight hours. Afterwards, the TGT is not usable by anyone, including the user or any attacker.

Summarizing, a client program logs on to Kerberos on behalf of a user. Under the covers, Kerberos acquires a Ticket-Granting Ticket (TGT) from the Authentication Service. The TGT is delivered as a data packet encrypted in a secret key derived from the user's password. Thus, only the valid user is able to enter a valid password and be able to decrypt the packet in order to the use of this TGT.

Whenever the client program requests services from the specific server, it must first send its TGT to the (Ticket Granting) Authentication Service to request a ticket to access that service. The TGT enables a client program to make such requests of the authentication service and allow the authentication service to verify the validity of such a request.

The ticket contains the user's identity and information that allows the ticket to be aged and expired tickets to be invalidated. The authentication service encrypts the ticket using a key known only to the desired server and the Kerberos Security Service. The key is known as the server key.

The encrypted ticket is transmitted to the server, who in turn presents it to the Kerberos authentication service to authenticate the identity of the client program and the validity of the ticket.




### 6.2.1.4 DB2 and Kerberos authentication

The key features of Kerberos security in OS/390 are:

- Authentication service

  The Kerberos authentication service provides trustworthy identification of principals involved in network operations. (A principal is defined to be an entity that can communicate securely with another entity.)

- A user (principal)

  A user (known to Kerberos as a principal) gains access to Kerberos by means of an account, which consists, in part, of the user's principal name and a secret key (derived from the user's password) that the user shares with the authentication service.

- Generic Security Services API

  The Generic Security Service API (GSSAPI) is a security API that distributed applications engaged in peer-to-peer communications can call to use the authentication services. The OS/390 SecureWay Security Server Network Authentication Privacy Service support is based on GSSAPI.

- RACF database as the registry for principal information

  The RACF database will be an integrated local OS/390 security registry and a Kerberos registry. The OS/390 SecureWay Kerberos Security Server registry information will be stored in RACF in the form of RACF Rule and General Resource Profiles. RACF commands will be used to administer the Kerberos registry information.

The OS/390 SecureWay Kerberos Security Server will then retrieve the information from RACF. DB2 will use two new security authorization facility (SAF) callable services:

- `R_ticketserv` — which enables OS/390 servers, like DB2, to parse and extract principle names from Kerberos tickets.
- `R_usermap` — which enables OS/390 servers, like DB2, to determine the RACF Userid associated with a Kerberos principle identity.

The RACF User profile will be extended to add a new KERB segment, which will be used to store information about local Kerberos principals. The RACF General Resource profile will also be extended to add a KERB segment

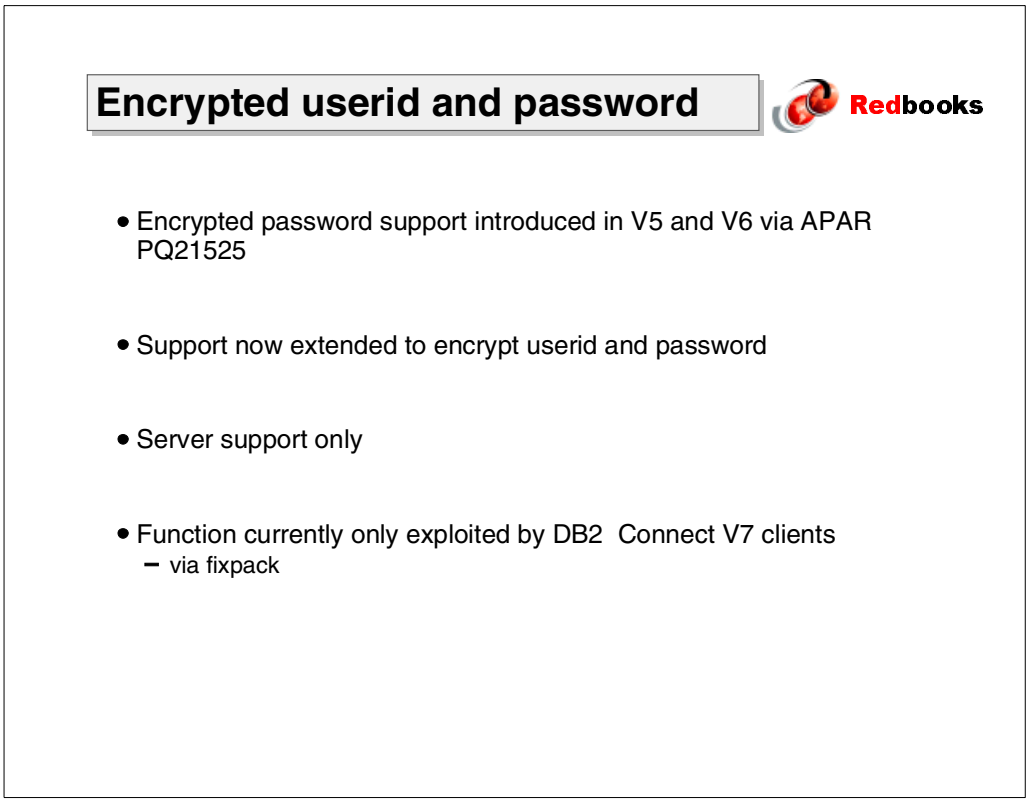
OS/390 support for Kerberos (the OS/390 SecureWay Security Server Network Authentication Privacy Service) will be available in OS/390 Version 2 Release 10. OS/390 will implement Level 5 of Kerberos. OS/390 support for Kerberos will be retrofitted to OS/390 Version 2 Release 8 and Release 9 by APAR/PTF. DB2 UDB for OS/390 Version 7 will recognize this support when it appears.

Since DCE is no longer the standard and is not being used, DCE support within DB2 UDB for OS/390 is removed in Version 7. Refer to Chapter 12, “Migration and fallback” on page 495 for a discussion of Migration/Fallback considerations for Kerberos support.

DB2 UDB for OS/390 Version 7 currently only supports Kerberos authentication as a Server.

Any Open Group Technical Standard DRDA Version 2 (implemented in IBM DRDA level 4) requester supporting the new Kerberos security mechanism can utilize the new DB2 for OS/390 Kerberos security support. DB2 Connect Version 7 supports the Kerberos security mechanism starting with Fixpack 2.

Finally, when support for DCE security was provided in DB2 UDB for OS/390 Version 5, the DISPLAY THREAD command was enhanced to indicate a status of ‘`RD`’ if a server thread was currently being authenticated using DCE services. The status of ‘`RD`’ is now replaced by ‘`RK`’, to indicate that the server thread is currently being authenticated by Kerberos services.




### 6.2.2 Encrypted userid and password

The APAR PQ21252 introduced support for encrypted passwords in DB2 for OS/390 Version 5 and Version 6. You can ask DB2 Connect to encrypt the password when it is sent to DB2 UDB for OS/390 for authentication.

The DB2 server support for the password encryption flows from the Open Group Technical Standard DRDA Version 2, (implemented in IBM DRDA level 4). Any compliant DRDA Version 2 requester can use password encryption. DB2 Connect V5.2 (Fixpack 7) and higher supports DRDA Version 2 password encryption.

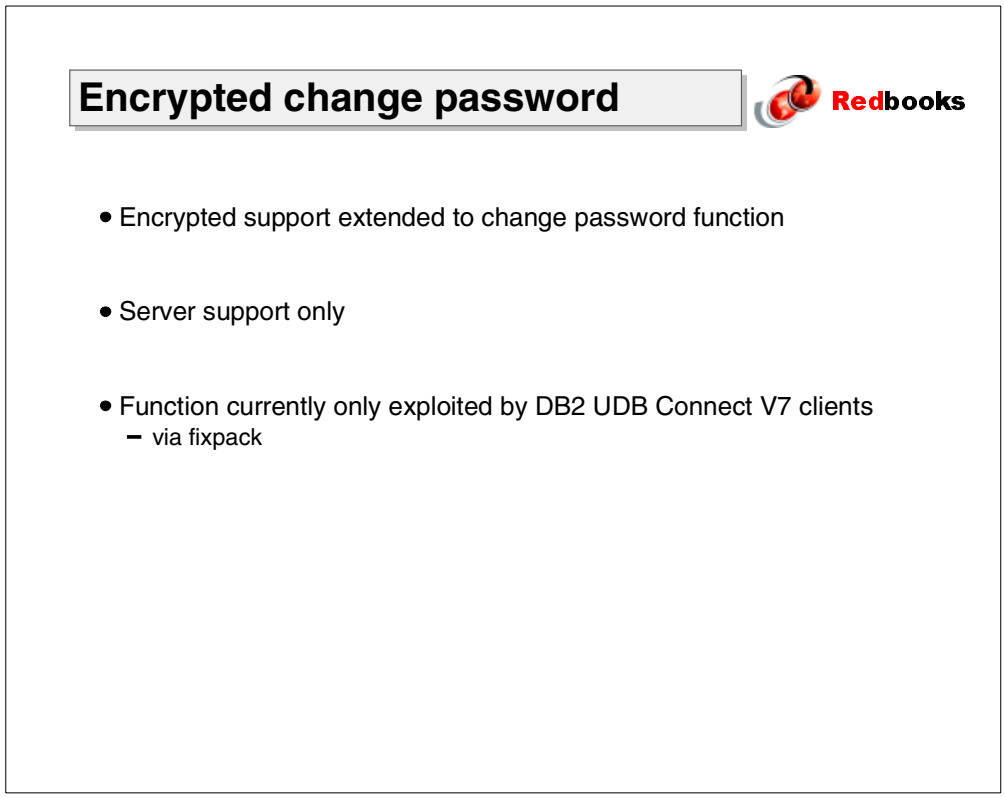
**Note:** If you need more details on the actual DRDA standards, visit the Web site:
`www.opengroup.org`

To enable DB2 Connect to flow encrypted passwords, DCS authentication must be set to DCS_ENCRYPT in the DCS directory entry. Refer to the Fixpack 7 for documentation.

DB2 UDB for OS/390 Version 7 extends this support to encryption of userids as well as passwords. Encrypted userids and passwords are only supported when DB2 UDB for OS/390 acts as a server. Remember, this enhancement is only applicable to distributed connections. DB2 UDB for OS/390 cannot act as a requester and send encrypted userids and passwords to a DRDA server.

When the workstation application issues an SQL CONNECT, the workstation negotiates this support with the DB2 server. If supported, a shared private key is generated by the client and server using Diffie-Hellman public key technology, and the userid and password are encrypted using 56-bit DES with the shared private key. The encrypted userid and password is non-replayable, and the shared private key is generated on every connect. If the server does not support password encryption, the application receives SQLCODE-30073 (“parameter not supported” error).

DB2 Connect V7 supports client userid encryption, via a fixpack yet to be determined.



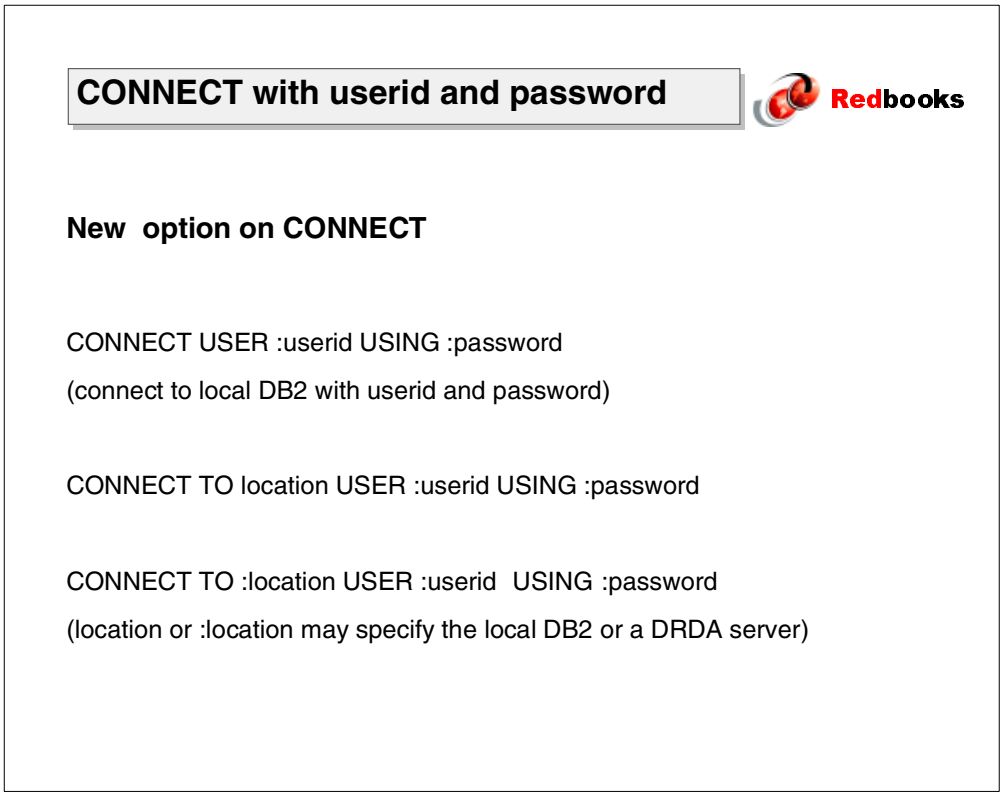

### 6.2.3 Encrypted change password

There is a sequence defined to allow remote client users to change their host OS/390 passwords remotely without having to logon to the host directly.

When encryption support was added, DB2 for OS/390 neglected to consider the change password process. As a consequence, DB2 for OS/390 required that the security tokens (userid, password, new password) must flow “in the clear”.

DB2 UDB for OS/390 Version 7 implements server support for encrypted userid and passwords. DB2 also extends this support to allow for encryption of change password tickets. Now the three security tokens, userid, old password, and new password, are all encrypted when sent to the host.

DB2 Connect V7 supports client encryption of change password, via a fixpack yet to be determined.




### 6.2.4 CONNECT with userid and password

You are now able to specify a userid and password when connecting to a server from an application running on OS/390. You can CONNECT to a remote server or a local server. The password is used to verify that you are authorized to connect to the DB2 subsystem.

DB2 UDB for UNIX, Windows, and OS/2 supports USER/USING as an option on the SQL CONNECT statement. Customers that wish to port applications to OS/390 or develop applications on workstation platforms, can now use this feature to port their applications to the OS/390 platform without having to reprogram, (for example, Java applications). In addition, products like WebSphere can make use of this function to reuse DB2 connections for different users and have DB2 for OS/390 perform the password checking that is not available via the DB2 SIGNON function.

Remote connections are established via DRDA connections. DRDA already provides support userid and password that may be up to 255 characters. DB2 for OS/390 supports these limits for the values specified via the CONNECT statement. For connections to DB2 for OS/390, both userid and password are limited to 8 characters.

The CONNECT statement has been enhanced to support the userid and password parameters, with the following rules:

- The userid and password can only be specified using host variables. This restriction is in place to reduce the security exposure caused by userids and passwords being entered as clear text. There is also is little value in being able to specify `userid` and/or `password` as literals.

- CONNECT USER :userid USING :password is equivalent to CONNECT TO :location USER :userid USING :password where :location is the location name of the local DB2 subsystem.
- You may either use TYPE 1 or TYPE 2 CONNECT.
- If you are using a TYPE 2 CONNECT, a current or dormant connection to the named server cannot exist unless the server is the local DB2 and the CONNECT statement is the first SQL statement executed after the DB2 thread was created. Otherwise you may use SET CONNECTION or CONNECT commands without the userid and password parameters to connect to the named server.
- The CONNECT statement can only be embedded in an application program. It is an executable statement that cannot be dynamically prepared.
- Authorization may not be specified when the connection type is IMS or CICS. An attempt to do so will cause a negative SQLCODE to be returned.

As the userid and password parameters on the CONNECT statement can only be host variables, the CONNECT enhancements are not completely compatible with the options supported by DB2 UDB for UNIX, Windows, OS/2. DB2 UDB for UNIX, Windows, OS/2 allows a userid and/or password to be supplied via a literal, and also supports the `NEW/CONFIRM` option which allows a new password value to be specified.

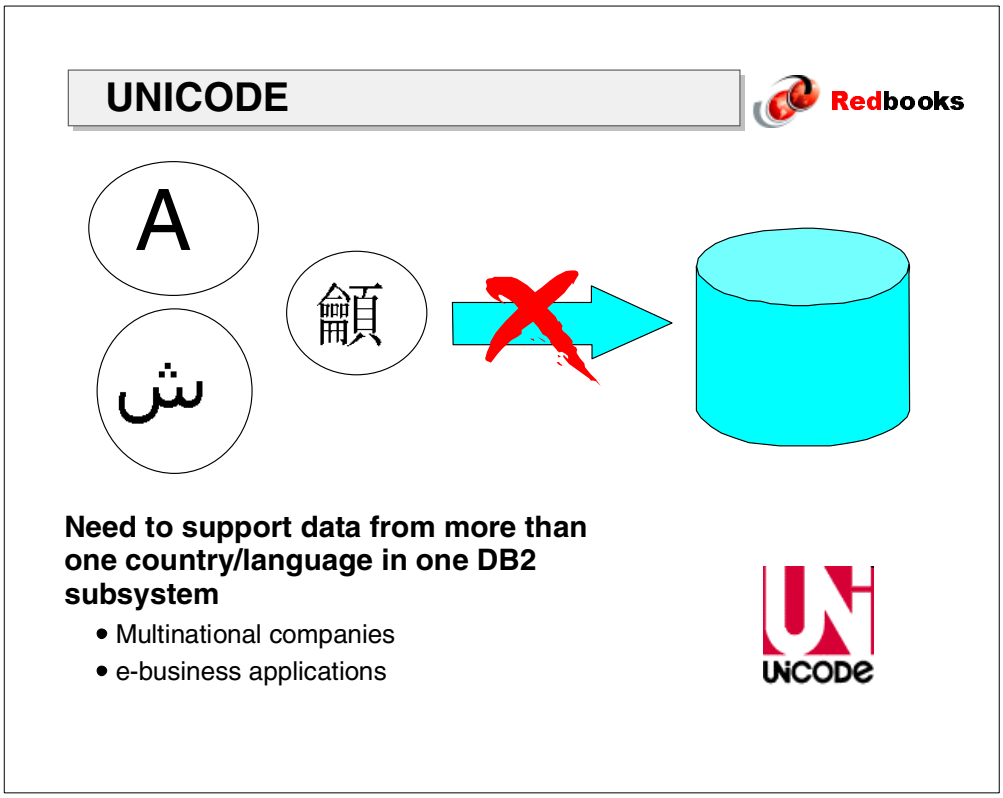
If the server is a local DB2:

- DB2 will invoke RACF via the RACROUTE macro to verify the password.
- If the password is verified and DB2 is a protected RACF resource (DSNR.* is active), DB2 then invokes RACF again via the RACROUTE macro to check whether the userid is allowed to use the DB2 subsystem.
- DB2 then invokes the connection exit routine if one has been defined. The connection now has a primary auth-id, possibly one or more secondary auth-ids and a SQLID.

If the server is not the local DB2, the server must support at least the Open Group Technical Standard DRDA Version 1 (implemented in IBM DRDA level 3). In this case, the userid and password are verified at the server. (DB2 UDB for OS/390 Version 5 introduced DRDA Version 1 support).

The following coding rules apply for the Communications Database at the local DB2 when connecting to a remote server:

- When using SNA, the ENCRYPTPSWDS column in SYSIBM.LUNAMES must not contain ‘Y’.
- The SECURITYOUT column in SYSIBM.LUNAMES must have either ‘A’ or ‘P’ specified. When ‘A’ is specified the userid and password will still be sent to the remote server.
- If the USER and USING parameters are specified on the CONNECT statement, no outbound translation will be done.




## 6.3 UNICODE

DB2 UDB for OS/390 is increasingly being used as a part of large client server systems. In these environments, character representations vary on clients and servers across different platforms and across many different geographies.

One area where this sort of environment exists is in the data centers of multinational companies. Another example is e-commerce. In both of these examples, a geographically diverse group of users interact with a central server, storing and retrieving data.

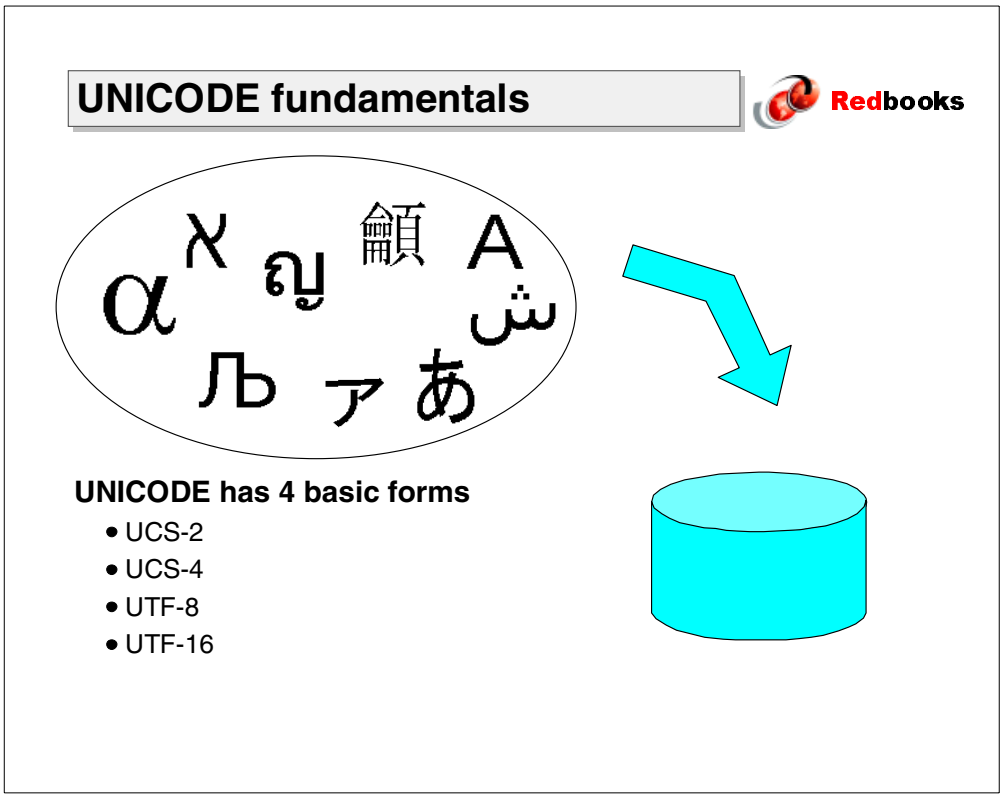
Today, there are hundreds of different encoding systems. No single encoding could contain enough characters: for example, the European Union alone requires several different encoding schemes to cover all its languages. Even for a single language like English, no single encoding was adequate for all the letters, punctuation, and technical symbols in common use.

These encoding systems also conflict with one another. That is, two encoding schemes can use the same number for two different characters, or use different numbers for the same character.

DB2 UDB for OS/390 Version 5, introduced support for storing data in ASCII. This support only solved part of the problem (padding and collating). It did not address the problem of users in many different geographies storing and accessing data in the same central DB2 server.

UNICODE is an encoding scheme that solves this problem. UNICODE is able to represent the characters of many different geographies and languages in the one encoding scheme.

In the following foils we will introduce UNICODE, then describe how UNICODE support is implemented in DB2 UDB for OS/390 Version 7.




### 6.3.1 UNICODE fundamentals

UNICODE provides a unique number for almost every character:

- No matter what the platform
- No matter what the program
- No matter what the language

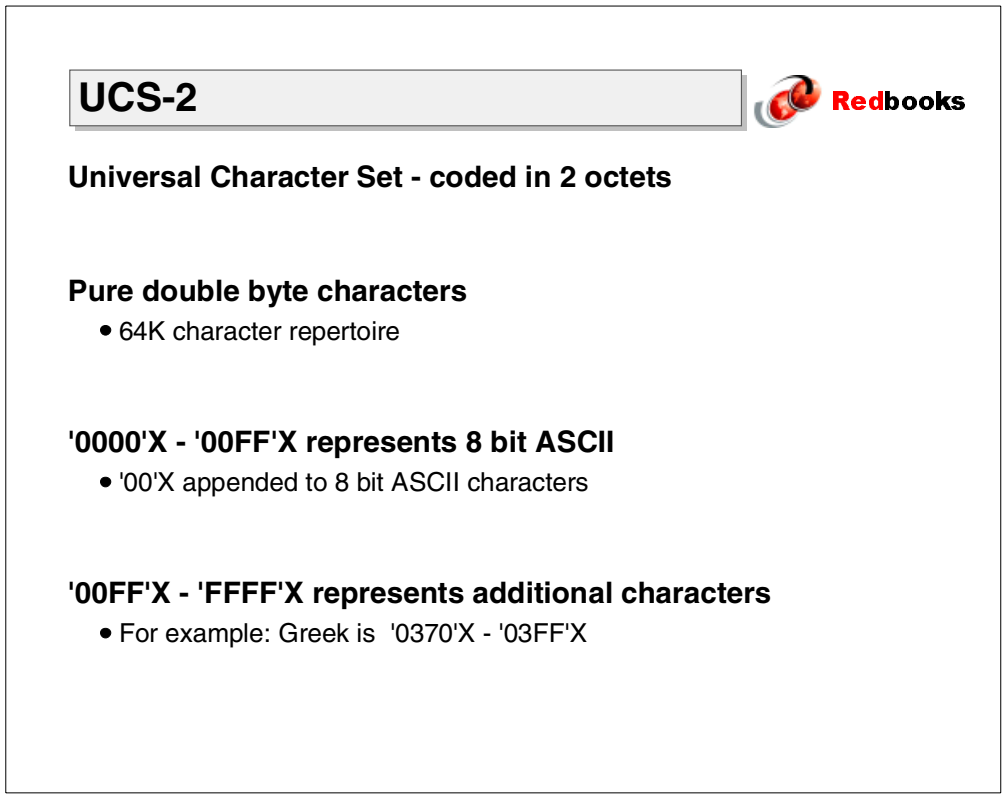
The UNICODE character encoding standard is a character-encoding scheme that includes characters from almost all the living languages of the world.

UNICODE is an implementation of the ISO-10646 standard and is supported and defined by the UNICODE consortium. See the Web site `http://www.unicode.org` for more information.

The UNICODE standard defines several popular implementations. The most popular are:

- UCS-2 - Universal Character Set coded in 2 octets.
- UCS-4 - Universal Character Set coded in 4 octets. This will become UTF-32.
- UTF-8 - UNICODE Transformation Format for 8 bit (ASCII safe UNICODE). Characters are encoded in 1 to 6 bytes.
- UTF-16 - UNICODE Transformation Format for 16 bits. The format is a superset of UCS-2 and contains an encoding form that allows more than 64K characters to be represented.

Each of these UNICODE implementations are expanded in the following sections.

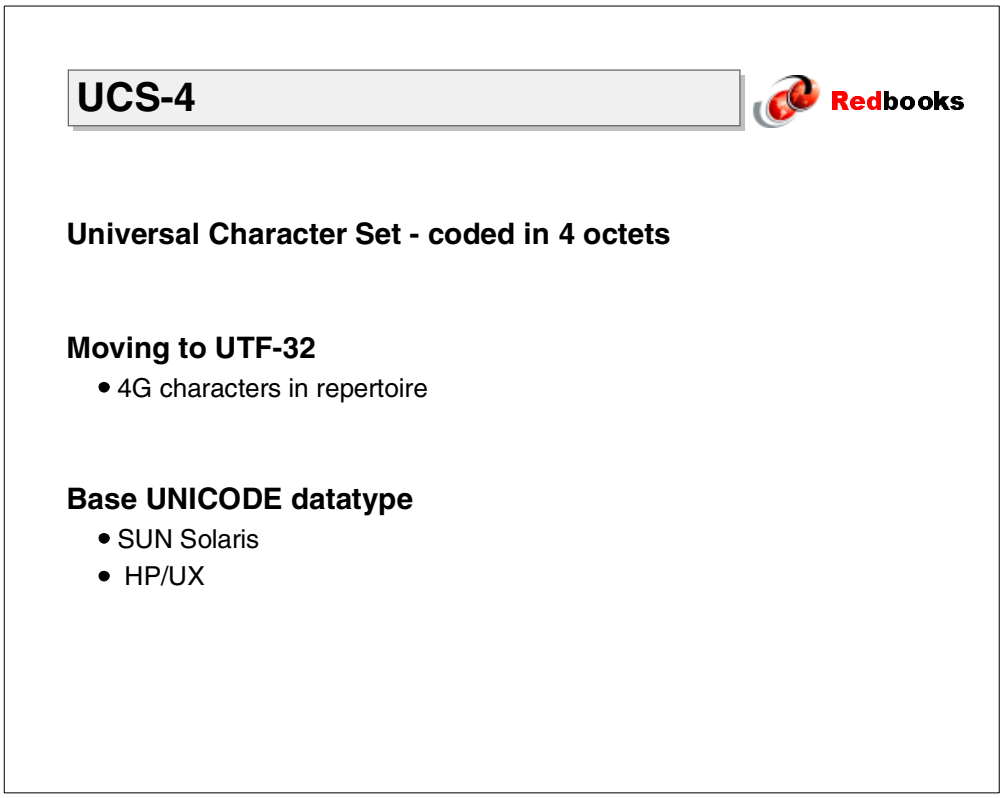



### 6.3.2 UCS-2

UCS-2 was published as a part of the original UNICODE standard.

UCS-2 is a fixed-width 16-bit encoding standard, with a range of 2*16 code points.

The UNICODE standard originally hoped this many coding points would be more than enough and hoped to stay with this range.

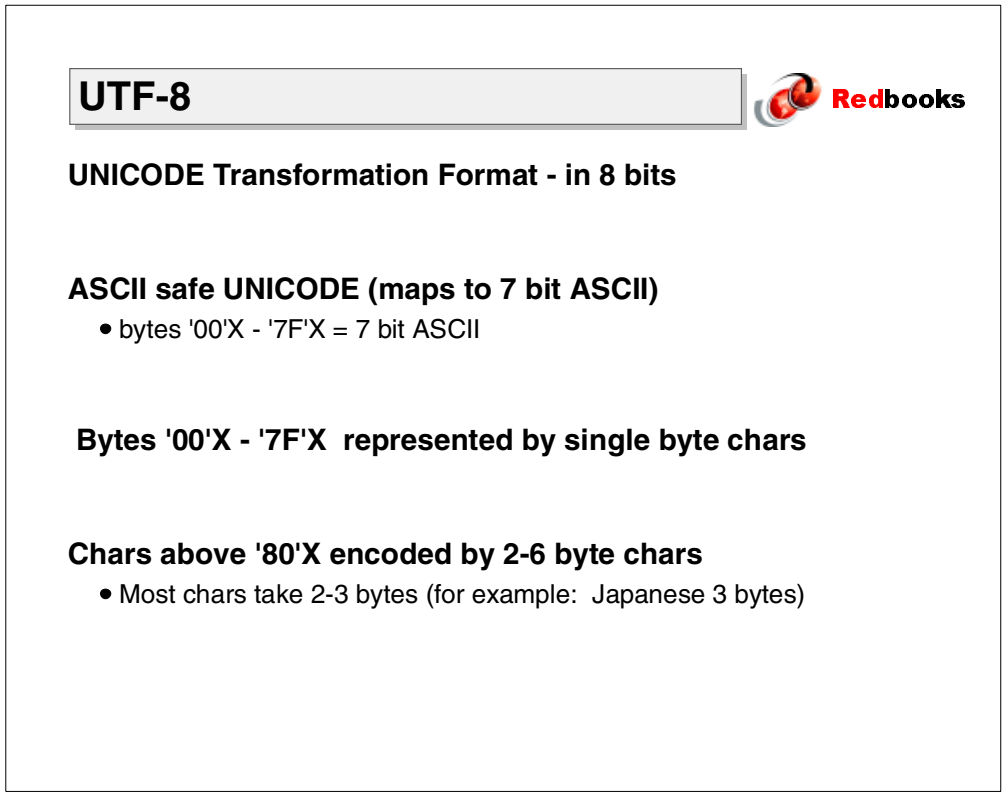



### 6.3.3 UCS-4

UCS-4 was also published as a part of the original UNICODE standard.

UCS-4 is a fixed-width 32-bit encoding standard, with a range of 2*31 code points. The 2*31 code points are grouped into 2*15 planes, each consisting of 2*16 code points. The planes are numbered from 0. Plane 0, the Basic Multilingual Plane (BMP), corresponds to UCS-2 above.

UTF-32 is simply another fixed-width 32-bit encoding standard, where each UNICODE code point (also know as scalar value) corresponds to a single 32-bit unit. UTF-32 is simply a subset of UCS-4 characters.




### 6.3.4 UTF-8

Shortly after the original UNICODE standard was published, a “UCS Transformation Format” (UTF) was defined. This was UTF-8.

You will observe that the TFs use bit numbering (8 and 16), while the CSs use octet numbering (2 and 4). This can be somewhat confusing.

UTF-8 uses a sequence of 8-bit values to encode UCS code points. Unlike UTF-16, UTF-8 can encode the entire UCS-4 space. UTF-8 looks like this:

- If the top bit is not set, it is a 1-octet sequence, representing an ASCII character.
- If the top bit is set and the next bit is unset, it is a multi-octet sequence, and we are looking at the Nth octet (where N>1).
- Otherwise, it's a multi-octet sequence, and we are looking at the first octet. The number of set bits before the first unset bit is equal to the number of octets in the sequence. So, we have:

  0xxxxxxx (payload of 7 bits)

  110xxxxx 10xxxxxx (payload of 11 bits)

  1110xxxx 10xxxxxx 10xxxxxx (payload of 16 bits)

  11110xxx 10xxxxxx 10xxxxxx 10xxxxxx (payload of 21 bits)

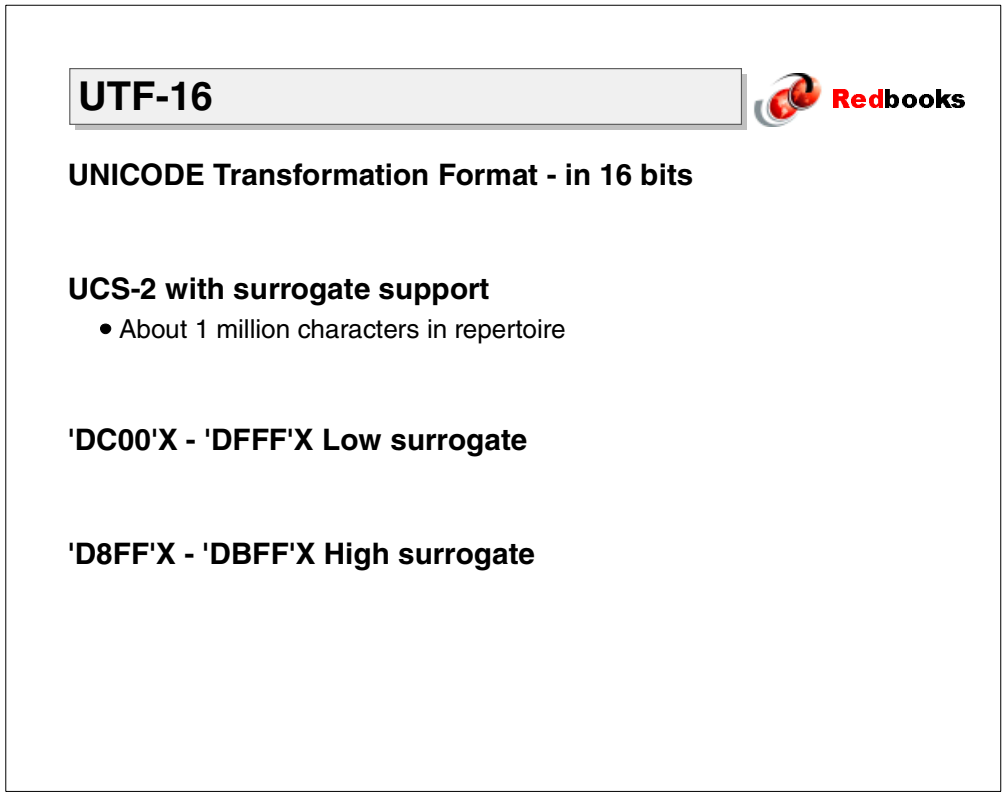
  111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx (payload of 26 bits)

The payload is derived by taking the UCS-4 encoding of the code point and dropping the bits into the octets, starting with the least significant bit. Unused bits are left unset.

An important rule is that you must use the least possible number of octets. So “A” is:

01000001 not: 11000001 10000001

Or, it can even be any longer sequence.



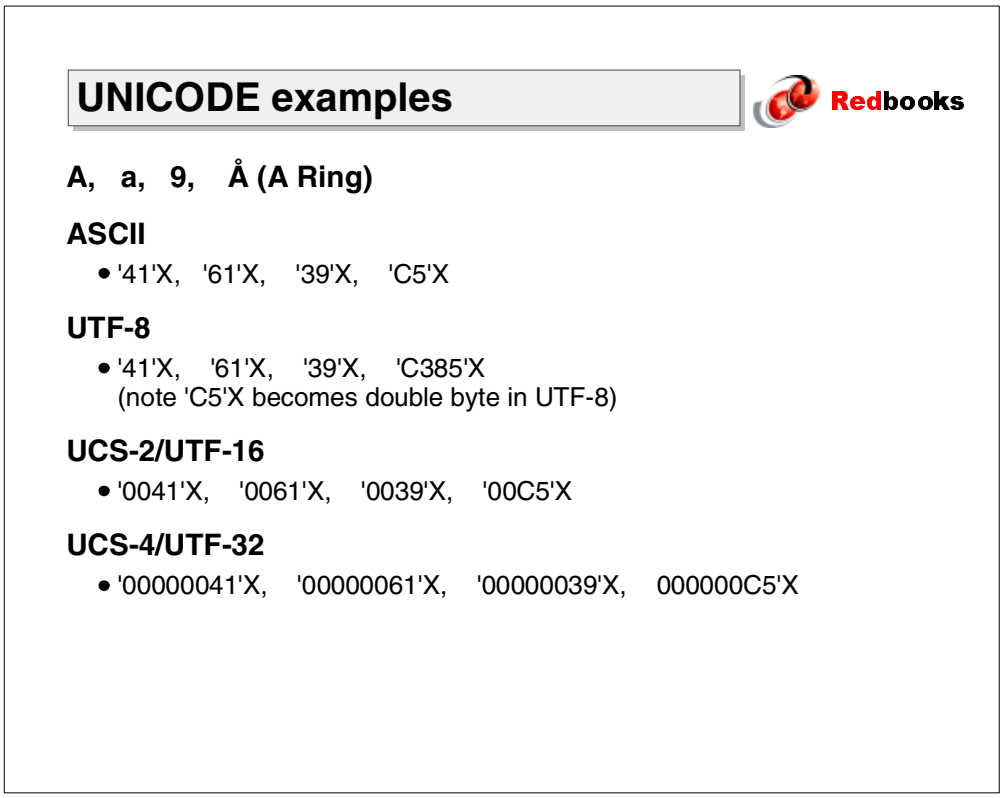

### 6.3.5 UTF-16

Not long after, UTF-16 was defined.

Of the 2*15 planes defined in UCS-4 above, only (2*4)+1=17 will be populated. The last two planes (15 and 16) are reserved for private use.

A “UCS Transformation Format” is defined which allows the encoding of this range of code points using a sequence of 16-bit values. This transformation format, called UTF-16, uses one 16-bit value from the range D800-DBFF, followed by one 16-bit value from the range DC00-DFFF, to encode any code point in the range 10000 to 10FFFF (i.e. in planes 1 to 16). It uses a single 16-bit value to encode any code point in the BMP. This is the same 16-bit value as used by UCS-2.

To explain further, each character is 16 bits (2 bytes) wide regardless of the language. While the resulting 65,536 code elements are sufficient for encoding most of the characters of the major languages of the world, the UNICODE standard also provides an extension mechanism that allows for encoding as many as a million more characters. This extension reserves a range of code values (D800 to D8FF, known as ‘surrogates’) for encoding some 32-bit characters as two successive code points.

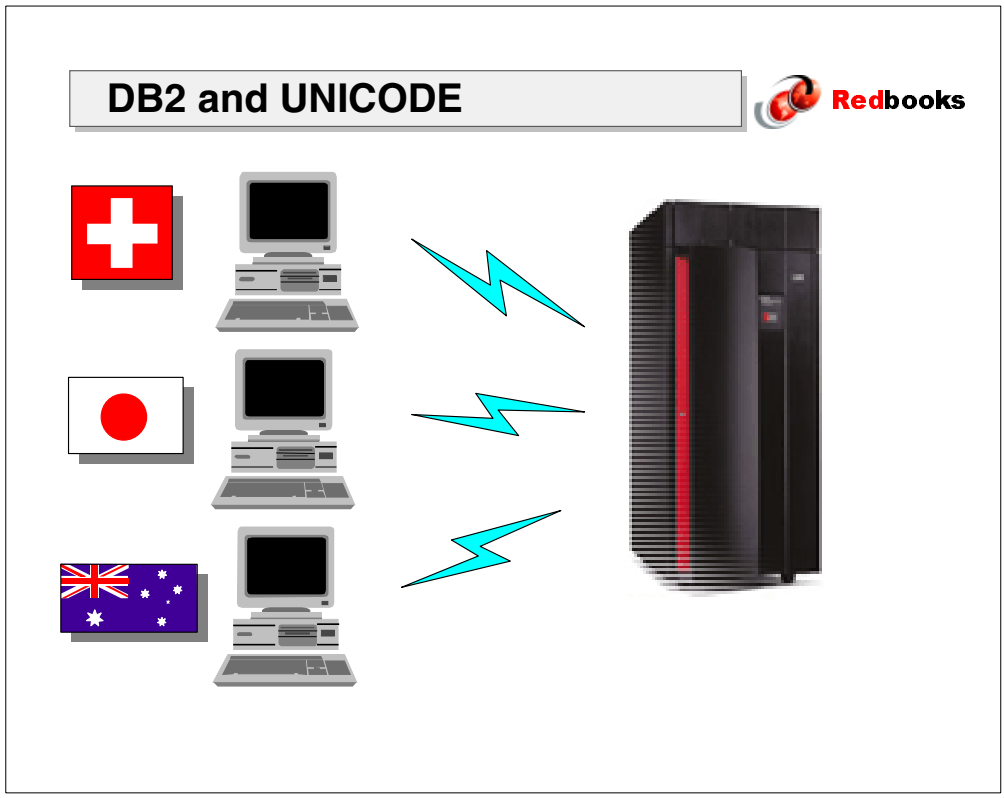



### 6.3.6 UNICODE examples

In this foil we show an example of how some characters are stored in UNICODE.

We are storing the characters ‘A’, ‘a’, ‘9’ and A-Ring (the character A with Ring accent).

Note that the character A-Ring requires 2 bites to be stored in UTF-8 format.




### 6.3.7 DB2 and UNICODE

DB2 UDB for OS/390 Version 7 introduces the concept of a UNICODE CCSIDs for both data storage and manipulation.

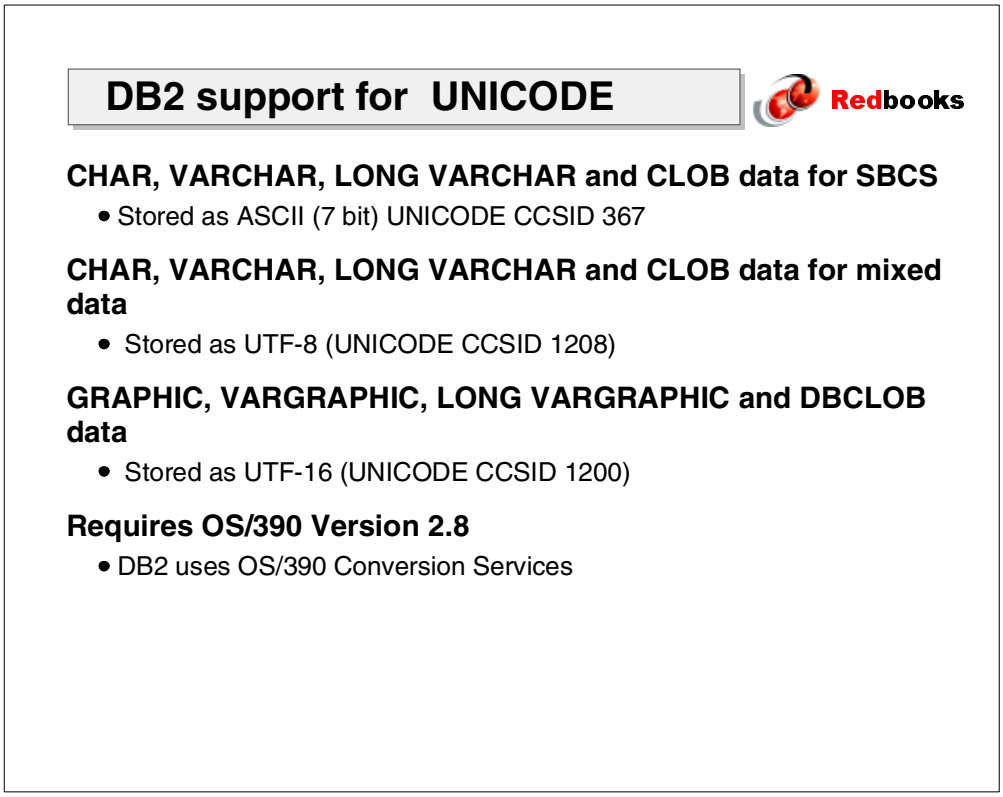
Prior to DB2 UDB for OS/390 Version 7, users are limited to a single encoding scheme, for example, the Latin-I subset of ASCII or EBCDIC. This is because DB2 UDB for OS/390 only allows one set of ASCII and one set of EBCDIC CCSIDs per system. ASCII and EBCDIC CCSIDs are set up to support either one specific geography, for example, 297 is French EBCDIC, or one generic geography, for example, 500 is Latin-I which applies for Western Europe.

There are no generic CCSIDs for the Far East, which means that there is no CCSID support for more than one Far Eastern country. For example, you cannot store Chinese and Korean characters in the same DB2 subsystem.

Remember the term ASCII as it is used here, is a generic term that refers to all ASCII codepages (CCSIDs) that DB2 UDB for OS/390 currently supports. The term EBCDIC as it is used here, is also a generic term that refers to all EBCDIC CCSIDs that DB2 UDB for OS/390 currently supports.

DB2's support for UNICODE is seen as an "application enabling technology", as many other software technologies already provide support for UNICODE, for example, Java, ODBC, Windows 2000, and XML.

You can now much more easily store and access data for many different languages in a single DB2 for OS/390 subsystem.



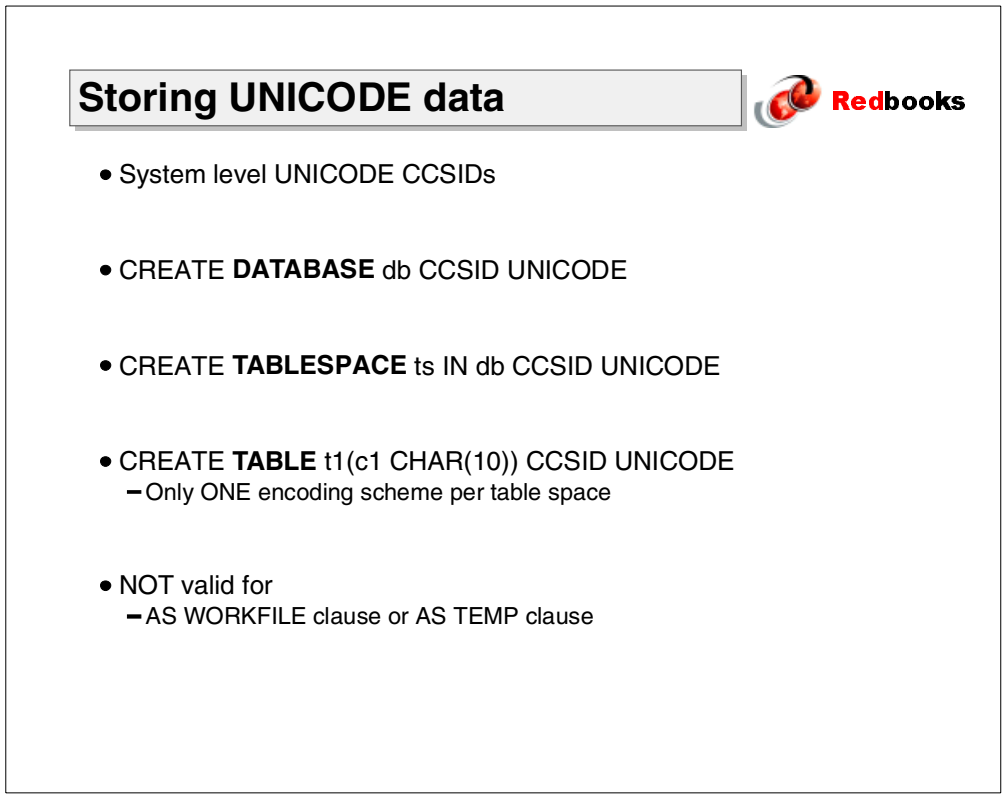

### 6.3.8 DB2 support for UNICODE

DB2 implements the UTF-8 and UTF-16 implementations of UNICODE:

- CHAR, VARCHAR, LONG VARCHAR and CLOB data for SBCS data is stored as ASCII (7 bit) UNICODE CCSID 367,
- CHAR, VARCHAR, LONG VARCHAR and CLOB data for mixed data is stored as UTF-8 (UNICODE CCSID 1208),
- GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC and DBCLOB data is stored as UTF-16 (UNICODE CCSID 1200)

If you are working with character string data in UTF-8, you should be aware that ASCII characters are encoded into one byte lengths; however, non-ASCII characters, for example, Japanese characters, are encoded into 2 or 3 byte lengths in a multiple-byte character code set (MBCS). Therefore, if you define an ‘n’ bytes length character column, you can store strings anywhere from ‘n/3’ to ‘n’ characters depending on the ratio of ASCII to non-ASCII character code elements.

DB2 does not use the table SYSIBM.SYSSTRINGS for conversion to and from UNICODE CCSIDs. Instead DB2 uses OS/390 Conversion Services, a feature shipped with OS/390 Version 2 Release 8, to manage all the conversions to and from UNICODE CCSIDs.




### 6.3.9 Storing UNICODE data

Data can now be stored in the database using ASCII, EBCDIC or UNICODE encoding schemes. Although an encoding scheme can be specified for CREATE DATABASE, CREATE TABLESPACE, CREATE TABLE, CREATE GLOBAL TEMPORARY TABLE or DECLARE GLOBAL TEMPORARY TABLE, all data stored in a table must use the same encoding scheme.

Except for Global Temporary Tables (created or declared), all tables within a table space must use the same encoding scheme, otherwise an SQLCODE -875 will be returned on the CREATE TABLE statement. The encoding scheme associated with a table space is determined when the table space is created.

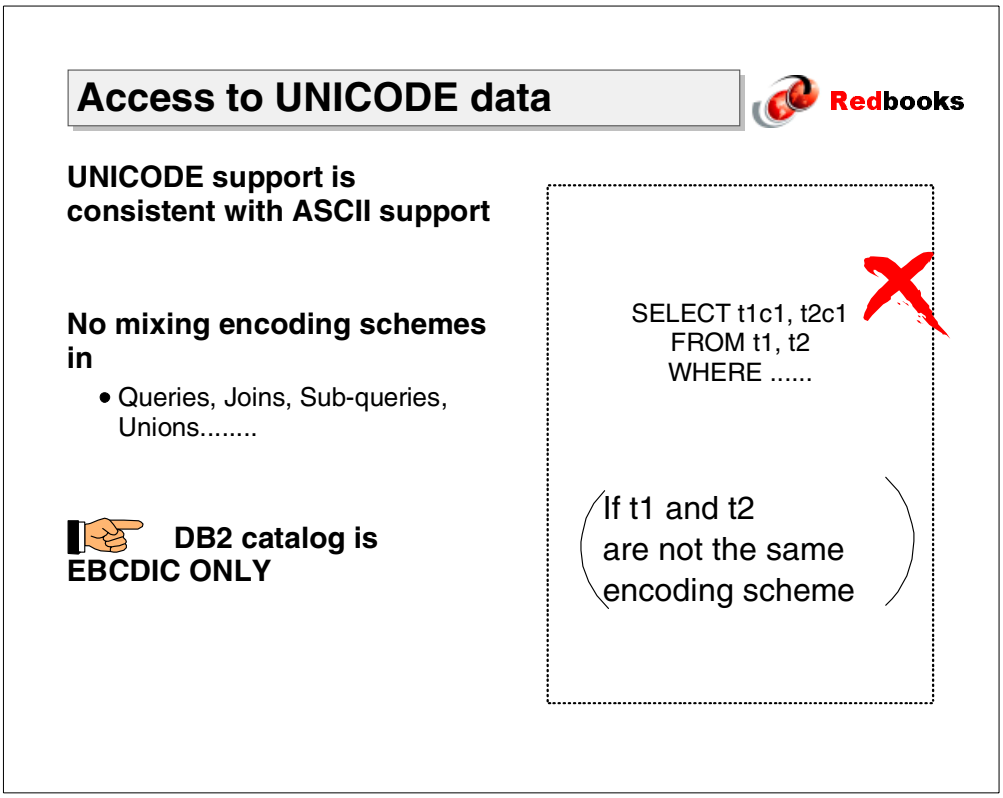
Indexes have the same encoding scheme as their tables. For a UNICODE table, all indexes are stored in UNICODE binary order.

There is no support for changing the ASCII/EBCDIC/UNICODE attribute on a database, table space or table once it has been defined. It is therefore strongly recommended that CCSIDs not be changed once they have been specified. Otherwise the results of SQL will be unpredictable.

The CCSID ASCII/EBCDIC/UNICODE clause is not supported for CREATE DATABASE for workfiles or Temp tables, or CREATE TABLESPACE for workfile temp tables. If the AS WORKFILE clause or AS TEMP clause is specified along with the CCSID ASCII/EBCDIC/UNICODE clause, the create will fail with SQLCODE -618.

The following SQL statements have also changed to support the specification of UNICODE as part of the data type declaration:

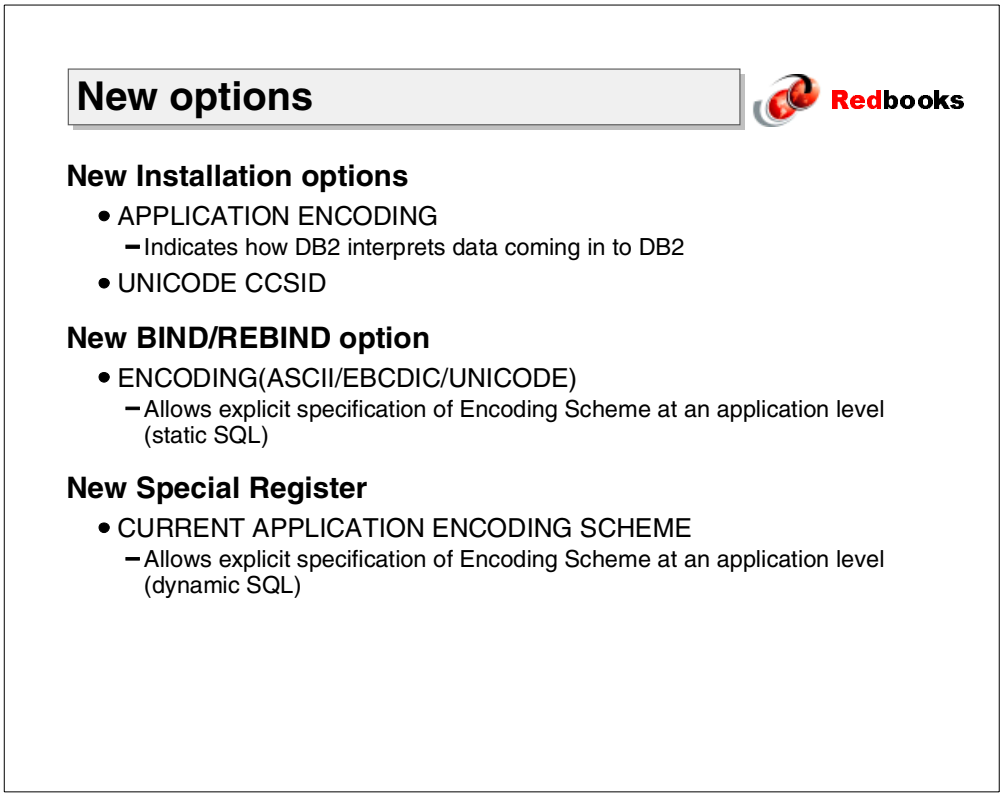
- DROP (function)
- CREATE DISTINCT TYPE
- CREATE FUNCTION
- CREATE PROCEDURE
- COMMENT
- ALTER FUNCTION
- GRANT
- REVOKE




### 6.3.10 Access to UNICODE data

A UNICODE table can be referenced in SELECT, INSERT, UPDATE and DELETE SQL statements as long as all tables referenced in the SQL statement refer to UNICODE tables. Referencing tables with more that one encoding scheme (ASCII, EBCDIC or UNICODE) in a single SQL statement is not supported. An SQL statement that violates this restriction will return a SQLCODE -873.

The DB2 Catalog database uses EBCDIC as its encoding scheme. This cannot be changed. Therefore, you cannot reference both a catalog table and a table encoded in ASCII or UNICODE in the same SQL statement.




## 6.3.11 New options

A number of new options have been included in DB2 UDB for OS/390 Version 7 to support the implementation of UNICODE.

### 6.3.11.1 Installation options

A new installation option, *APPLICATION ENCODING*, is added where the default application encoding scheme can be specified at installation time. This option can have the following values: ASCII, EBCDIC, or UNICODE.

The default APPLICATION ENCODING scheme affects how DB2 interprets data coming into DB2. For example, if you set your default application encoding scheme to 37, and your EBCDIC Coded character set is 500, then DB2 will convert all host variable data coming into the system from 37 to 500 before using it. This includes, but is not limited to, SQL statement text and host variables.

The default value, EBCDIC, will cause DB2 to retain the behavior of previous releases of DB2 (assume that all data is in the EBCDIC system CCSID).This default should not be changed if you need backward compatibility with existing applications.

A new field is also added to the installation panel DSNTIPF, *UNICODE CCSID*. This field is like the existing ASCII CCSID and EBCDIC CCSID fields, in that it specifies the detault UNICODE CCSID to use. DB2 defaults the UNICODE CCSID field to 1208 (the UTF-8 CSSID). DB2 picks the CSSIDs for the Double byte and Single byte CSSID values, (1200 for DBCS and 367 for SBCS). 1200 is the UTF-16 CSSID and 367 is a UNICODE 7-bit ASCII CSSID.

#### 6.3.11.2 BIND/REBIND option

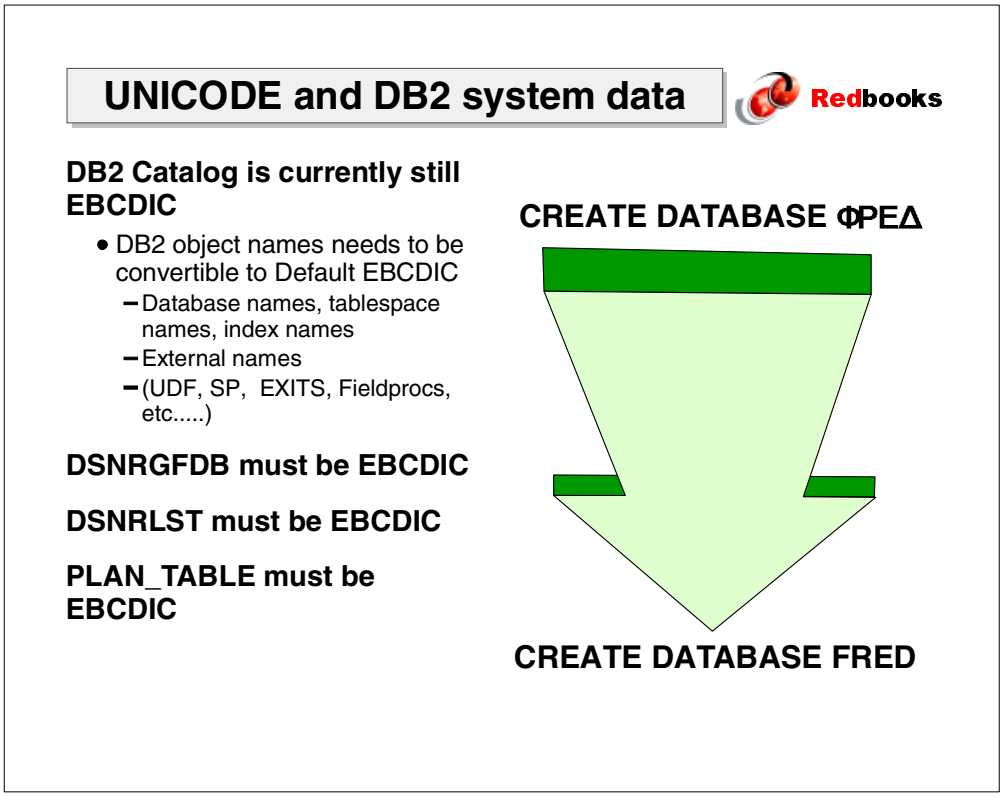
A new BIND/REBIND option is added,
*ENCODING(ASCII/EBCDIC/UNICODE/ccsid)*. This controls the application encoding scheme which is used for all the static statements in the plan/package. The default is the system default APPLICATION ENCODING scheme specified at installation time. The default package application encoding option is not inherited from the plan application encoding option

#### 6.3.11.3 Special register

A new special register is also added to DB2 UDB for OS/390 Version 7. *CURRENT APPLICATION ENCODING SCHEME* enables an application to specify the encoding scheme that is being used by the application for dynamic statements.

The value returned in the special register is a character representation of a CCSID. Although you can use the values 'ASCII', 'EBCDIC' or 'UNICODE' to SET the special register, the values set in the special register is the character representation of the numeric CCSID corresponding to the value used in the SET command. The value 'ASCII', 'EBCDIC' and 'UNICODE' will not be stored.

The new scalar function, *CCSID_ENCODING*, can be used to return a value of 'ASCII',' EBCDIC' or 'UNICODE' from the numeric CCSID value. Refer to 6.3.15, "Routines and functions" on page 320 for a description of this scalar function.




## 6.3.12 UNICODE and DB2 system data

The DB2 system tables remain as EBCDIC tables.

### 6.3.12.1 DB2 Catalog (DSNDB06)

The DB2 catalog database uses the system default EBCDIC encoding scheme. This cannot be changed. DB2 object names, like database names and table names, must therefore be convertible to the System EBCDIC CCSID without loss of data.

DB2 parsing of SQL and utility control statements is done in EBCDIC. DB2 converts the whole input string to the EBCDIC System default CCSID from ASCII or UNICODE before it passes the string to the parser.

Even though you can create, separately, a table with a Greek name and a table with a German name, an SQL statement like the following example is still not valid:

```
SELECT *
FROM <greek table name> t1, <german table name> t2
WHEREt1.c1 = t2.c2
AND ......
```

DB2 will convert both the <greek table name> and <german table name> to the same CCSID then fail to find the tables. (This is because only one CCSID is passed to DB2 so DB2 can only convert one table name to EBCDIC correctly remembering the DB2 Catalog tables are still EBCDIC.)

#### 6.3.12.2 Data Definition Control Support database (DDCS)

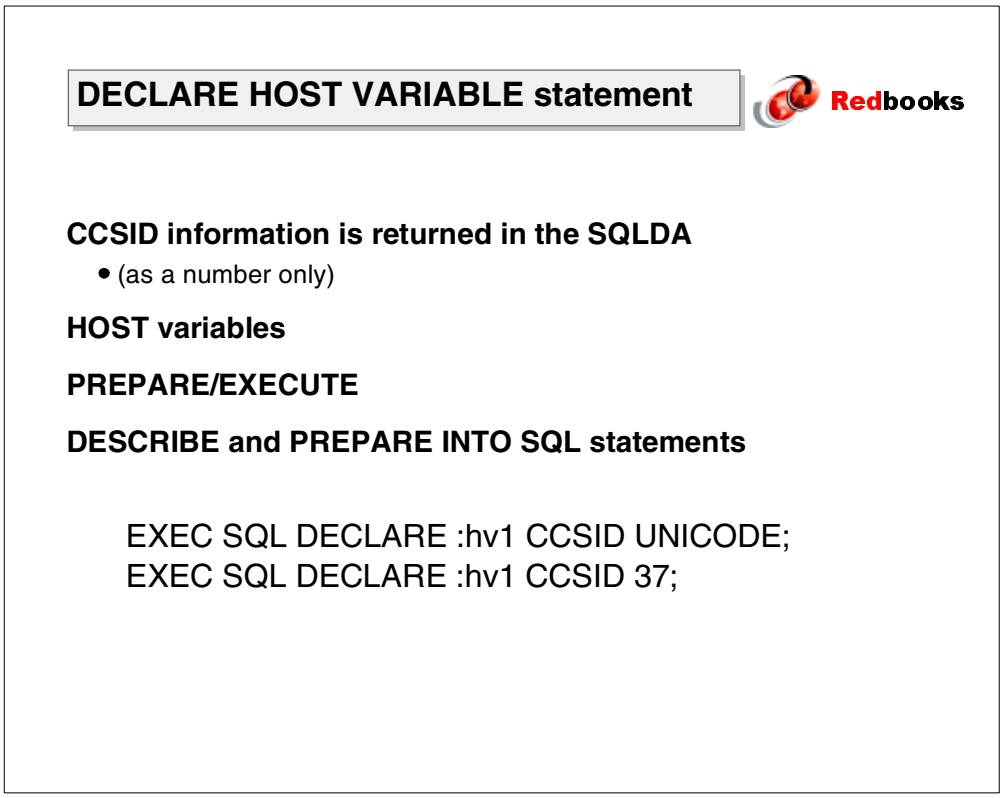
The tables in the database DSNRGFDB are used to control the access to run DDL in the DB2 subsystem. If during installation you specified different names for the objects id the DDCS database, they are restricted to EBCDIC. Any tables created in the DSNRGFDB database must be EBCDIC. Otherwise a SQLCODE -877 will result.

#### 6.3.12.3 Resource Limit Specification Tables (RLST)

The Resource Limit Specification Tables are used to control the resources that can be used by dynamic SQL. If during installation you specified different names for the objects id the RLST database, they are restricted to EBCDIC. Any tables created in the DSNRLST database must be EBCDIC. Otherwise a SQLCODE -877 will result.

#### 6.3.12.4 PLAN_TABLE

The PLAN_TABLE must be defined as EBCDIC. If a PLAN_TABLE is defined as ASCII or UNICODE then a SQLCODE -878 is issued when the table is used for an EXPLAIN SQL statement, or message DSNT408I is issued when the table is used for the EXPLAIN bind parameter.




## 6.3.13 DECLARE HOST VARIABLE statement

Applications have several options to return UNICODE data from DB2.

### 6.3.13.1 Host variables

If data encoded in UNICODE is returned to the application by a SELECT statement, the data is returned in the system default encoding scheme (it will be converted). In this case the data will also be returned in the binary order for UNICODE encoding scheme, except when:

- CCSIDs were specified in the SQLDA to indicate the data should be returned in the specified CCSIDs.

  To do this, you specify a USING clause with the SQLDA on the SQL statement. In this case, code the 6th byte of the SQLDAID as '+' and specify the UNICODE CSSID in the SQLNAME field associated with the UNICODE host variable. (Specifying CSSIDs in the SQLDA can be found in the manual *DB2 UDB for OS/390 SQL Reference.* SC26 9014.)

- The host variable is defined with the *DECLARE VARIABLE* statement.

  If the SQL statement uses a host variable that has been declared in a DECLARE VARIABLE statement, then the DB2 precompiler automatically codes the equivalent setting in the SQLDA with a CSSID. This allows statements where a USING clause is not allowed (for example, SELECT FROM :hostvar) to indicate that the data should be returned in a specific CCSID.

- The CURRENT APPLICATION ENCODING SCHEME special register (for dynamic SQL) is specified.

Refer to 6.3.11.3, “Special register” on page 312 for a discussion on the APPLICATION ENCODING special register.

- The ENCODING Bind option (for static SQL) is specified at bind time.

  Refer to 6.3.11.2, “BIND/REBIND option” on page 312, for a discussion on the new Bind options.

To be compatible with earlier releases of DB2, the default encoding scheme of host variables is the system default encoding scheme.

#### 6.3.13.2 PREPARE and EXECUTE IMMEDIATE statements

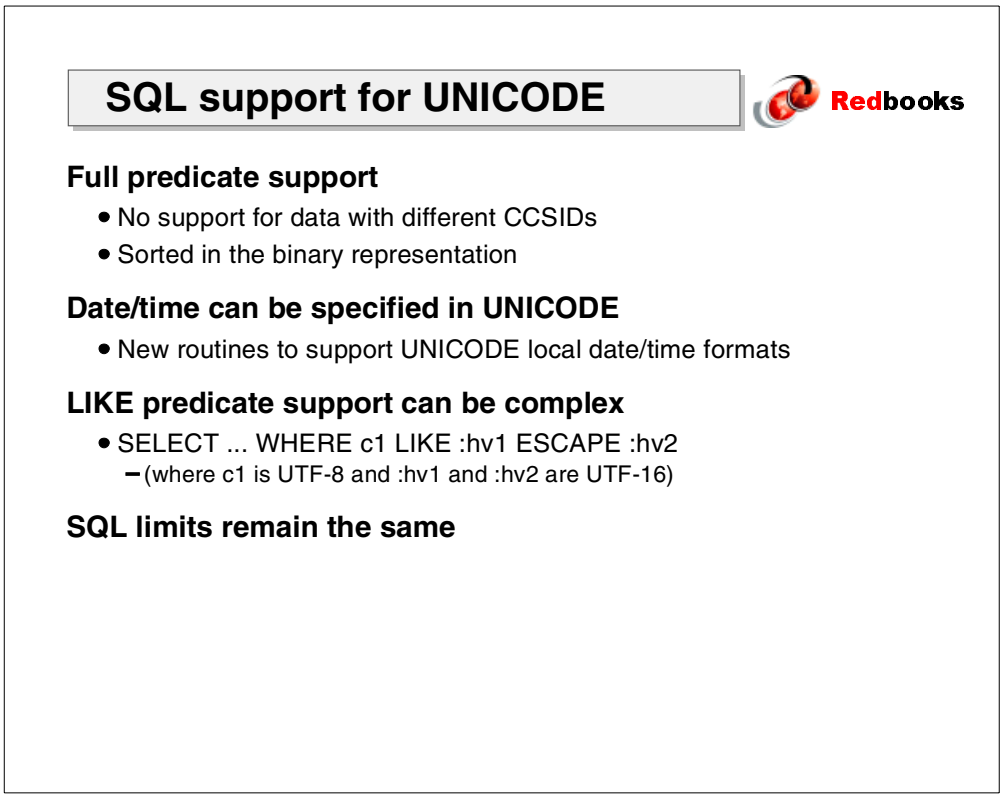
In the past there has been no way for an application to provide to DB2 information about the encoding scheme used for the string being prepared.

You can now use the new SQL statement, DECLARE VARIABLE, to tell DB2 about the CCSIDs of the host variable(s). The precompiler is also updated to provide CCSID information whenever this new host variable is referenced.

#### 6.3.13.3 DESCRIBE and PREPARE INTO SQL statements

CCSID information is returned in the SQLDA (in the SQLDATA field), when a DESCRIBE or PREPARE INTO statement is executed. However, the CCSID returned is merely a number, and there is no general way for an application to determine the encoding scheme (ASCII, EBCDIC or UNICODE) from the CCSID in the SQLDATA field of the SQLDA.

A new built in scalar function is included with DB2 UDB for OS/390 Version 7. CCSID_ENCODING will assist in determining if a CCSID is ASCII, EBCDIC or UNICODE. Refer to 6.3.15, “Routines and functions” on page 320 for a description of this scalar function.




## 6.3.14 SQL support for UNICODE

### 6.3.14.1 Full predicate support

The full range of SQL predicates is supported when processing UNICODE data. However, DB2 does not support data with different CCSIDs within SQL statements.

The coding sequence for string comparisons and sorting is determined by the encoding scheme of the data being compared or sorted. UNICODE data will be sorted in the binary representation of that data. (Note that ASCII and EBCDIC data is also sorted on the binary representation of the data.)

### 6.3.14.2 Date/time data

Date/time strings may be specified in UNICODE if the SQL statement has a USING clause identifying a SQLDA which specifies the UNICODE CSSID of the date/time string. Additionally you can store a date/time string in a date/time column in a UNICODE table in the database.

For the local format of date/time data, the existing date/time exit routines DSNXVDTX and DSNXVTMX, are invoked to format the date//time data in EBCDIC.

For date/time data encoded in ASCII, exit routine stubs DSNXVDTA for date and DSNXVTMA for time, are provided. These routines are invoked when local formatting is requested for date/time data encoded in ASCII.

Similarly, the exit routine stubs (DSNXVDTU for date and DSNXVTMU for time) are provided. These routines are invoked when local formatting is requested for date/time data encoded in UNICODE. Both DSNXVDTU and DSNXVTMU should be coded to accept UTF-8 format data. DB2 will pass these routines UTF-8 format data even if the input is presented to DB2 in a UTF-16 string.

#### 6.3.14.3 LIKE predicate

The LIKE predicate remains a stage 1 predicate and is indexable.

When the LIKE predicate is used with UNICODE data, the information in the LIKE clause of the SQL statement is converted to the appropriate format (UTF-8 or UTF-16). Support for LIKE predicate can therefore be a little complicated. Consider this example:

```
SELECT ... WHERE c1 LIKE :hv1 ESCAPE :hv2
(where c1 is UTF-8 and :hv1 and :hv2 are UTF-16)
```

DB2 must convert and compare host variables of different CCSID lengths.

The following UNICODE percent sign and underscore characters will be recognized:

| Character | UTF-8 | UTF-16 |
|---|---|---|
| half-width % | ‘25’X | ‘0025’X |
| full-width % | ‘EFB085’X | ‘FF05’X |
| half-width _ | ‘5F’X | ‘005F’X |
| full-width _ | ‘EFBCBF’X | ‘FF3F’X |

The full or half width ‘%’ character will match 0 or more UNICODE characters. The full or half ‘_’ character will match exactly 1 UNICODE character. When dealing with ASCII or EBCDIC data, the full width ‘_’ character will match 1 DBCS character. So the behavior of the full width ‘_’ character is slightly different for UNICODE data when compared to ASCII or EBCDIC data.

The escape character consists of a single SBCS (1 byte) or DBCS (2 bytes) character(s). An escape clause is allowed for UNICODE mixed (UTF-8) data but will continue to be restricted for ASCII and EBCDIC mixed data.

#### 6.3.14.4 Padding characters

If the data is UNICODE UTF-8 or SBCS the padding character is ‘20’X. For UNICODE UTF-16 the padding character is ‘0020’X.

If the data is ASCII SBCS or ASCII mixed the padding character is ‘20’X. For ASCII DBCS the padding character depends on the CCSID. For example, for Japan (CCSID 301) the padding character is ‘8140’X, while for Chinese it is ‘A1A1’X.

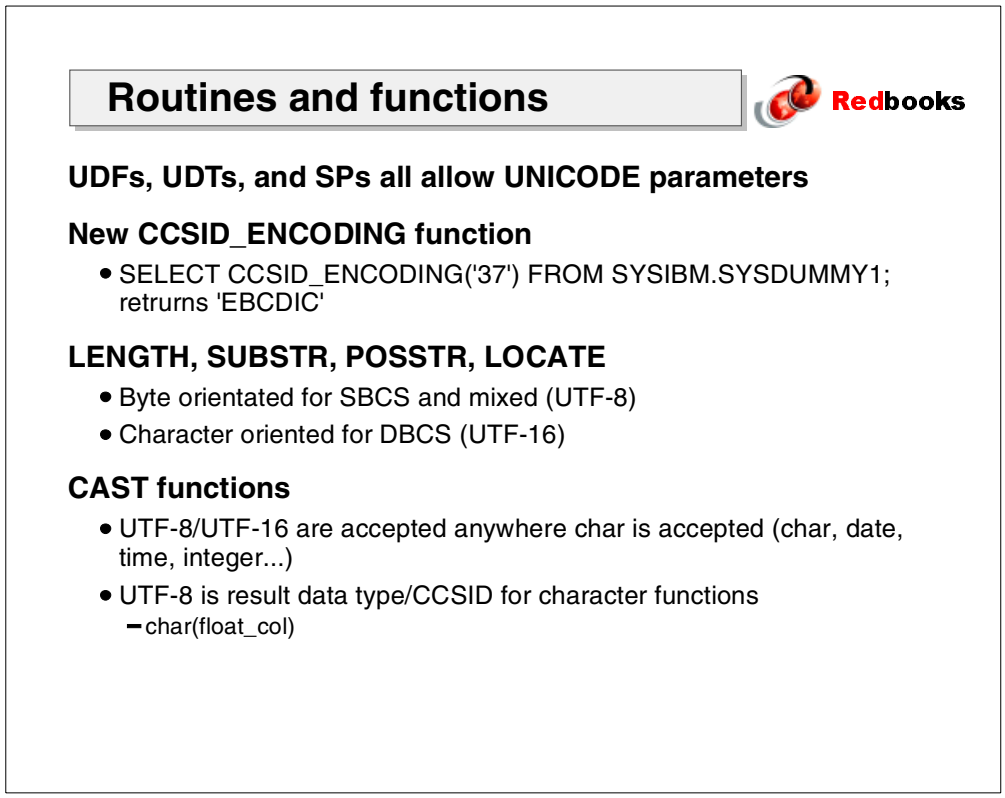
If the data is EBCDIC SBCS or EBCDIC mixed, the padding character is ‘40’X. For EBCDIC DBCS the padding character is ‘4040’X.

Remember that the coding sequence for string comparisons and sorting is determined by the encoding scheme of the data being compared or sorted. UNICODE data will be sorted in the binary representation of that data. (Note that ASCII and EBCDIC data is also sorted on the binary representation of the data.)

#### 6.3.14.5 SQL limits

Although UNICODE encoding can increase the number of bytes to code predicates, all the SQL limits remain unchanged:

- SQL predicates are still limited to 255 bytes.
- The maximum length for the pattern expression for LIKE predicates remain at 4000 bytes.
- The index key size remains at 255 bytes.
- Strings larger that 32K bytes must be defined as CLOBs.
- The maximum length of a character remains at 255 bytes.
- The varchar limit is still 32K bytes.




### 6.3.15 Routines and functions

The following DB2 features are enhanced to support UNICODE:

- Built-in scalar functions
- Built-in column functions
- Stored procedure support
- User defined functions
- User defined table functions

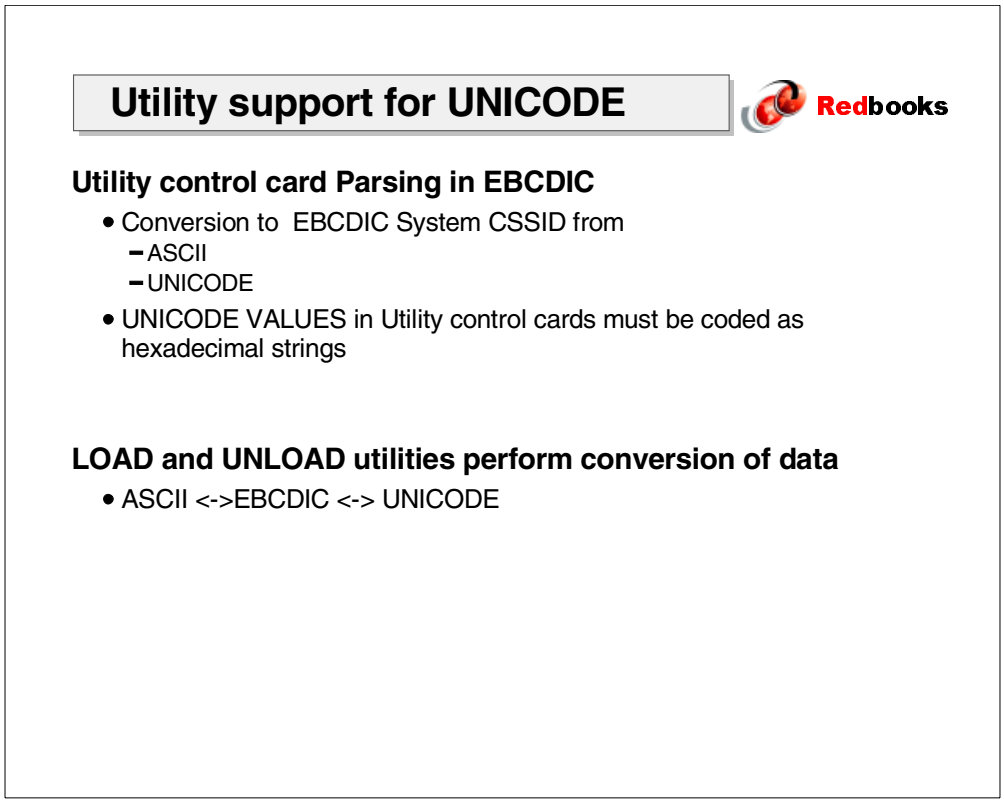
A new scalar function, *CCSID_ENCODING* is introduced with DB2 UDB for OS/390 Version 7.

The value returned in the new special register CURRENT APPLICATION ENCODING SCHEME is a character representation of a CCSID. Although you can use the values ‘ASCII’, ‘EBCDIC’ or ‘UNICODE’ for the SET special register function, the values set in the special register is the character representation of the numeric CCSID corresponding to the value used in the set command. DB2 also returns the character representation of the CCSID in the SQLDA.

The new scalar function, CCSID_ENCODING, can be used to return a value of ‘ASCII’,’ EBCDIC’ or ‘UNICODE’ from the numeric CCSID value.

The SQL functions LENGHT, SUBSTR, POSSTR, LOCATE operate at the byte level for SBCS and mixed (UTF-8). They are character oriented for DBCS (UTF-16).

UNICODE UTF-8/UTF-16 are accepted anywhere char is accepted (char, date, time, integer...). UTF-8 is the result cast data type/CCSID for character functions, (for example: char(float_col)).




### 6.3.16 Utility support for UNICODE

All utility control statements and messages are coded in EBCDIC. Any conversion of variables to the EBCDIC System default CCSID from ASCII or UNICODE is done before the DB2 catalog is accessed.

The LOAD utility input data may be coded in ASCII, EBCDIC or UNICODE. The ASCII/EBCDIC/UNICODE option in the LOAD utility control statements specify the format of the input data. Similarly, the CCSID option in the LOAD utility control statements specifies the CCSIDs of the data in the input file. Up to three CCSIDs may be specified, representing the SBCS, MIXED and DBCS CCSIDs. If any of the individual CCSIDs are omitted, the default CCSIDs for the encoding scheme is chosen.

The input data may be loaded into ASCII, EBCDIC or UNICODE tables. If the CCSID in the input data does not match the CCSID of the table space, the input data will be converted to the CCSID of the table space before being loaded.

For best performance the CCSIDs of the input data should match the CCSIDs of the table space and the CCSID option should not be specified.

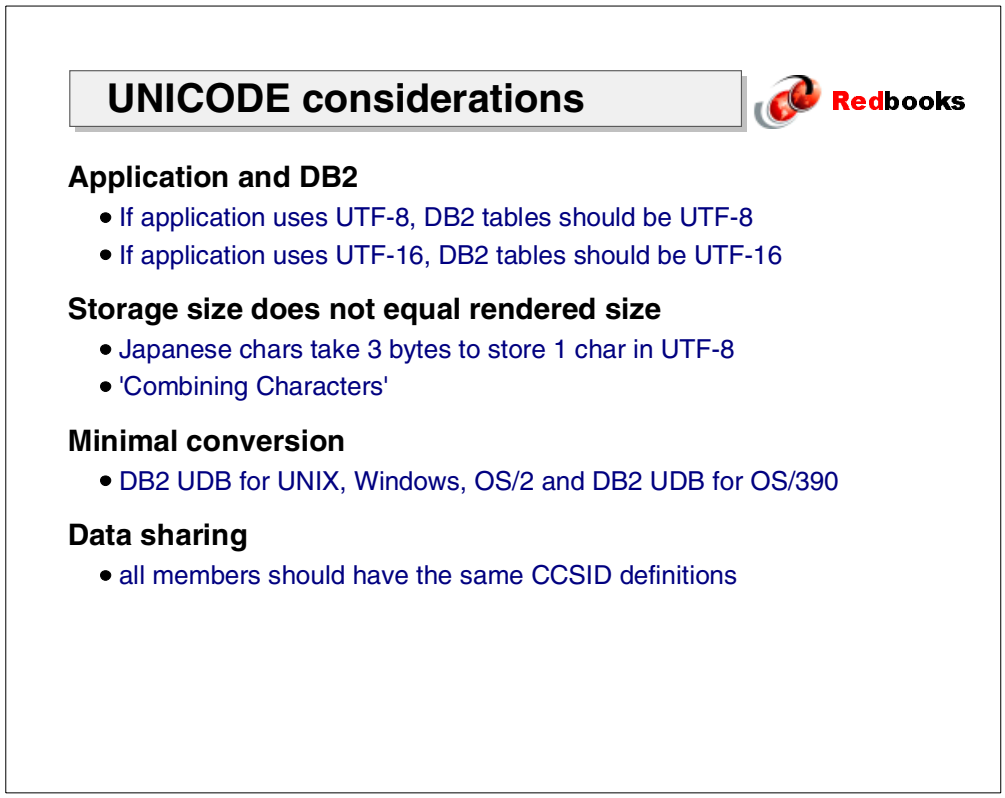
If DSN1COPY and DSN1PRNT are run on a UNICODE table space, the character string to be used to scan pages in the table space must be coded as a hexadecimal string in the VALUE option. Additionally, the readable portion of the output of these utilities will be interpreted as though the data was EBCDIC data.

In the LOAD REORG and UNLOAD utilities, character constants within the following options cannot be specified as UNICODE strings. No conversion of these values is done. To use these options with UNICODE data, the values must be specified as hexadecimal constants:

- CONTINUEI
- WHEN
- DEFAULTIF
- NULLIF

The REPAIR utility also does not allow character constants to be specified as UNICODE strings. No conversion of these values is done. To use these options with UNICODE data, the values must be specified as hexadecimal constants:

- LOCATE KEY
- VERIFY DATA
- REPLACE DATA




### 6.3.17 UNICODE considerations

There are a number of points you need to keep in mind when designing and implementing UNICODE applications.

***Beware the cost of conversion***
UTF-8 and UTF-16 are compatible almost everywhere, but you will pay a conversion cost. It is best to match the DB2 data definition to use the UNICODE model the application is using.

***UNICODE storage***
The storage size may not always equal the rendered size for some UNICODE characters. For example, Japanese characters take 3 bytes to store 1 character in UTF-8.

UNICODE has a concept called combining characters that allow something like A-Ring to be represented as A and Combining Character Ring. Combining Characters can add to the size needed for both UTF-8 and UTF-16 columns.

***Distributed data considerations***
At connect time, the requester and server systems exchange default system CCSID (SBCS, mixed and DBCS). When DB2 UDB for OS/390 is the requester, the default system CCSIDs exchanged are the EBCDIC default CCSIDs. These CCSIDs are used by the requester to send SQL statements to the server for PERPARE processing.

DB2 UDB for OS/390 Version 7 uses the UNICODE CCSID 367 for SBCS (7 bit ASCII), CCSID 1208 for mixed (UTF-8) and CCSID 1200 for DBCS (UTF-16).

DB2 UDB for UNIX, Windows, OS/2 supports UCS-2 (code page 1200) for GRAPHIC data, and UTF-8 (code page 1208) for CHAR data. When a database is created in UCS-2/UTF-8, CHAR, VARCHAR, LONG VARCHAR and CLOB data are stored in UTF-8, and GRAPHIC, VARGRAPHIC,LONG VARGRAPHIC and DBCLOB data are stored in UCS-2.

DB2 for AS/400 uses CSSID 13488 to support UNICODE. In general, conversion between DB2 for OS/390 and DB2 for UNIX, Windows, OS/2 will not occur when UNICODE data is exchanged, while conversion will occur between DB2 for OS/390 and DB2 for AS/400.

UNICODE CCSID 1200 is known as a 'superset' CCSID where new values are being assigned all the time. It also encapsulates other CCSIDs.

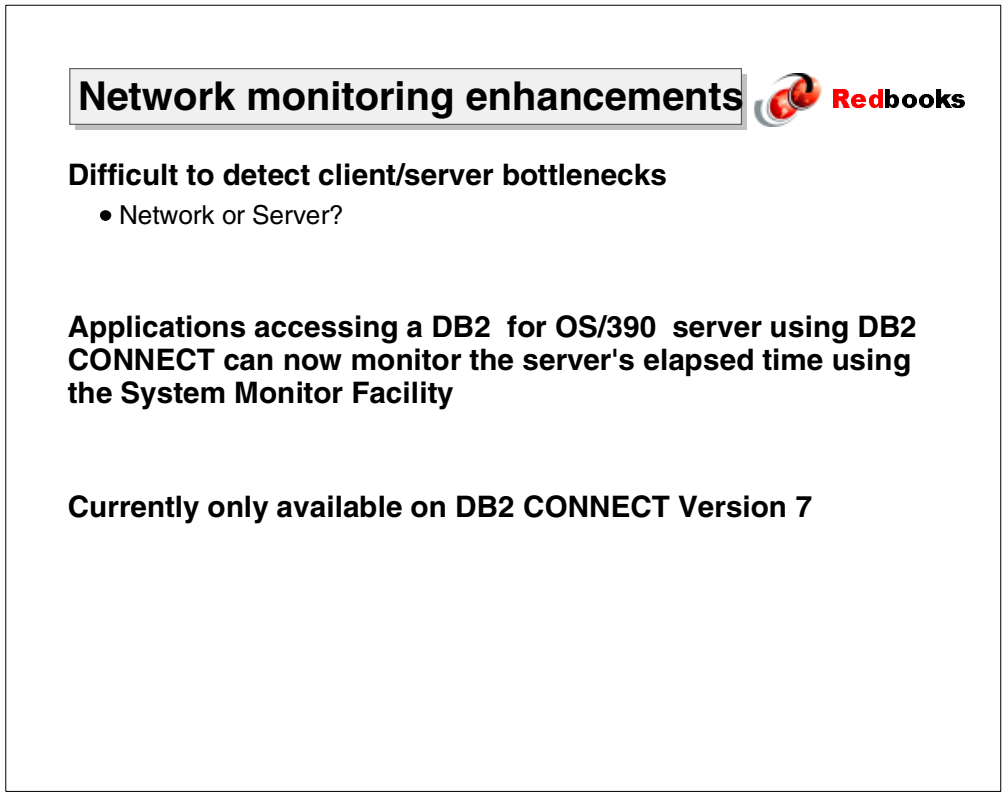
For example, UNICODE CCSID 13488 is defined in the UNICODE standard Version 2. UNICODE CCSID 17584 is defined in the UNICODE Version 3 standard. UNICODE CCSID 1200 is a superset of 13488 and 17584.

***Data sharing considerations***

We recommend that all members of a data sharing group adhere to these recommendations:

- Once one member of the data sharing group converts to a DB2 release that supports UNICODE, the rest of the members of the data sharing group should also be converted to the same release as soon as possible.
- If the CCSIDs are changed on one system, then the same changes should be made to all other members of the data sharing group.

If these recommendations are not followed, the results of SQL may be unpredictable.




## 6.4 Network monitoring enhancements

Currently, workstation environments cannot isolate the cause of poor response times for long running remote requests to conditions in the network or to the DB2 server, without performing a DB2 Connect trace and/or traces on the server. To improve this situation, DB2 UDB for OS/390 Version 7 provides the capability to selectively return the elapsed time spent in DB2 to process a remote request. This will allow clients to easily isolate poor response times to the network or to the DB2 server.

A timestamp is recorded when DDF first receives a remote request. A second timestamp is recorded after the DB2 server has parsed the request, processed any SQL statements and generated the reply. The difference in values is the elapsed time that is returned to the client.

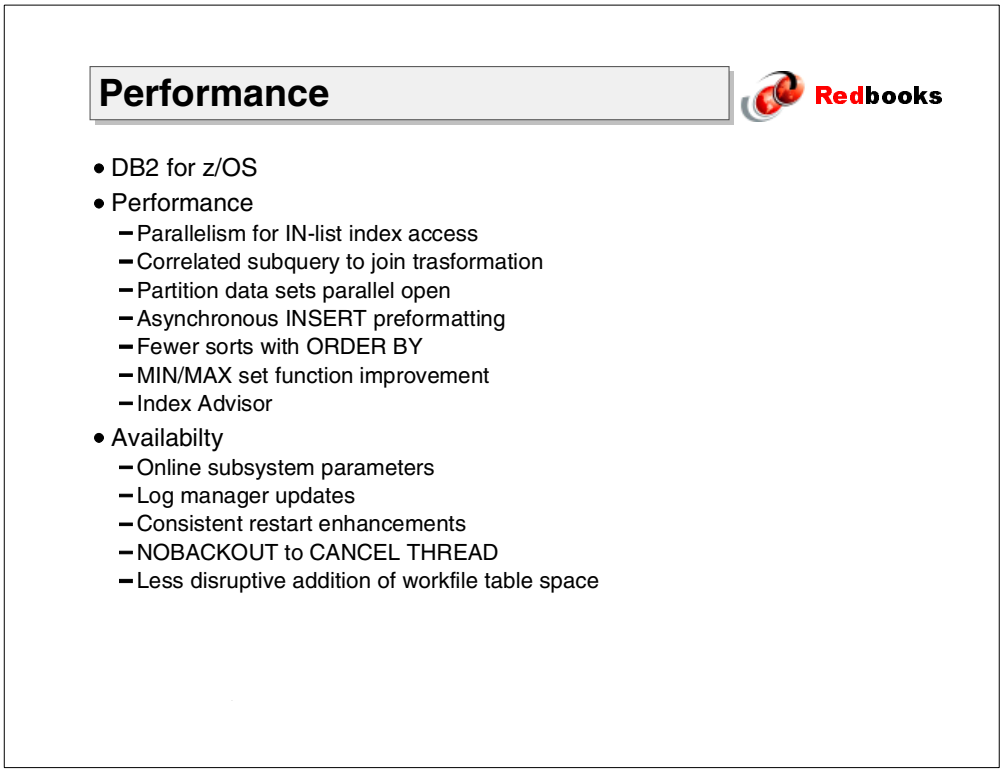
DB2 Connect Version 7 is adding to the System Monitor interface the ability to monitor the elapsed time spent by a DB2 server processing a request. DB2 Connect will only request this information when the System Monitor statement switch has been turned on. This information will then be returned as a new element through the regular System Monitoring APIs.

Currently, only DB2 Connect Version 7 workstation clients, via a fixpack, can request the DB2 to return the elapsed time to process a request and generate a reply. There is no interface on OS/390 to monitor the elapsed time at a server.

# Part 5. Performance and availability

© Copyright IBM Corp. 2001

# Chapter 7. Performance and availability




In this chapter we discuss the enhancements that improve the performance and availability of DB2. First we consider the performance enhancements. These enhancements are directly related to internal changes that improve performance.

- Parallelism for IN-list index access:

  This enhancement enables parallelism for those queries involving IN list index access.

- Correlated subquery to join transformation:

  DB2 V7 attempts to rewrite correlated subqueries in SELECT, UPDATE and DELETE statements, changing the subquery to joins if certain conditions are met.

- Partition data set parallel open:

  DB2 V7 allocates up to 20 tasks when opening the partitions of a table space.

- Asynchronous INSERT preformatting:

  DB2 V7 improves the performance of INSERTs by asynchronously preformatting allocated and not yet formatted but allocated pages

- Fewer sorts with ORDER BY:

  DB2 V7 can eliminate some of the elements in the ORDER BY if defined as constant in the WHERE clause.

© Copyright IBM Corp. 2001

- MIN/MAX set function improvement:

  DB2 V7 may eliminate the need for an extra index by being able to scroll forwards and backwards to point to the MIN or MAX values.

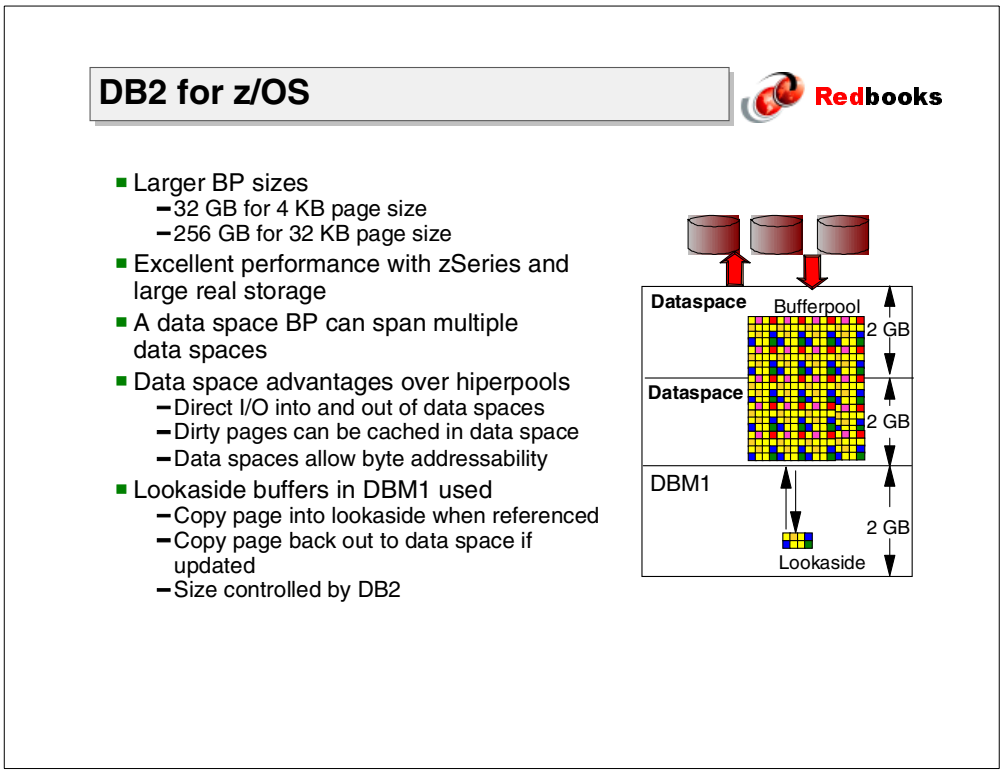
- Index Advisor:

  DB2 for UNIX, Windows, and OS2 hasan extension to the DB2 optimizer which returns advice on index definitions. DB2 V7 provides a way to obtain index candidates by migrating definitions.

Quantitative information on the performance of several of the enhancements described in this redbook will be included in another redbook, available after GA, dedicated to DB2 V7 performance.

In this chapter we also present the main enhancements that improve the availability of DB2. Some background information is provided by the redbook *DB2 UDB for OS/390 and Continuous Availability*, SG24-5486. These enhancements are for availability, some of them can indirectly improve also performance, and others offer a trade-off. The foil provides a summary of these enhancements and lists the topics that are presented.

- Online subsystem parameters
  - SET SYSPARM command
  - Generating and loading new parameters load module
  - Displaying current settings
- Log manager updates
  - Suspend update activity
  - Retry critical log read access errors
  - Time interval system checkpoint frequency
  - Long running UR enhancement
- Consistent restart enhancement
  - Recover postponed
- Add NOBACKOUT to CANCEL THREAD
- Less disruptive addition of workfile table space




## 7.1 DB2 and z/OS

Version 7 of DB2 for OS/390 includes the support for z/OS. z/OS is the new version of OS introduced to support and exploit the functions of the new line of large servers introduced with IBM e(logo)server zSeries. The zSeries builds on the prior generations of S/390 CMOS servers introducing a large increase in processing capacity when compared to the 9672 G6 Server. The zSeries provides also uniprocessor performance growth over the previous generation CMOS Turbo G6 model family and better performance with compressed data. The maximum number of Symmetrical Multi-Processors (SMPs) increase from 12 Central Processors (CPs) on the G6 Server to 16 CPs. For information relating to functions and performance of the z900 please refer to *IBM e(logo)server zSeries 900 Technical Guide*, SG24-5975, and its bibliography.

To ensure that the large single system image delivered by the zSeries can be exploited by our customers, IBM has introduced the 64-bit architecture. With this new architecture, the central storage-to-expanded storage Page Movement overhead associated with a large single system image is eliminated.

zSeries 900 has an enhanced I/O subsystem: The new I/O subsystem includes Dynamic CHPID Management (DCM) and channel CHPID assignment allow the full use of the bandwidth available for 256 channels in the z900.

Within the z900 the number of FIber CONnectivity (FICON) channels has been increased to 96, giving the z900 well over double the concurrent I/O capability of a fully configured IBM G6 Server. Fewer FICON channels are required to provide the same bandwidth as ESCON, and more devices per channel are supported. This reduces channel connections and thus reduces I/O management complexity.

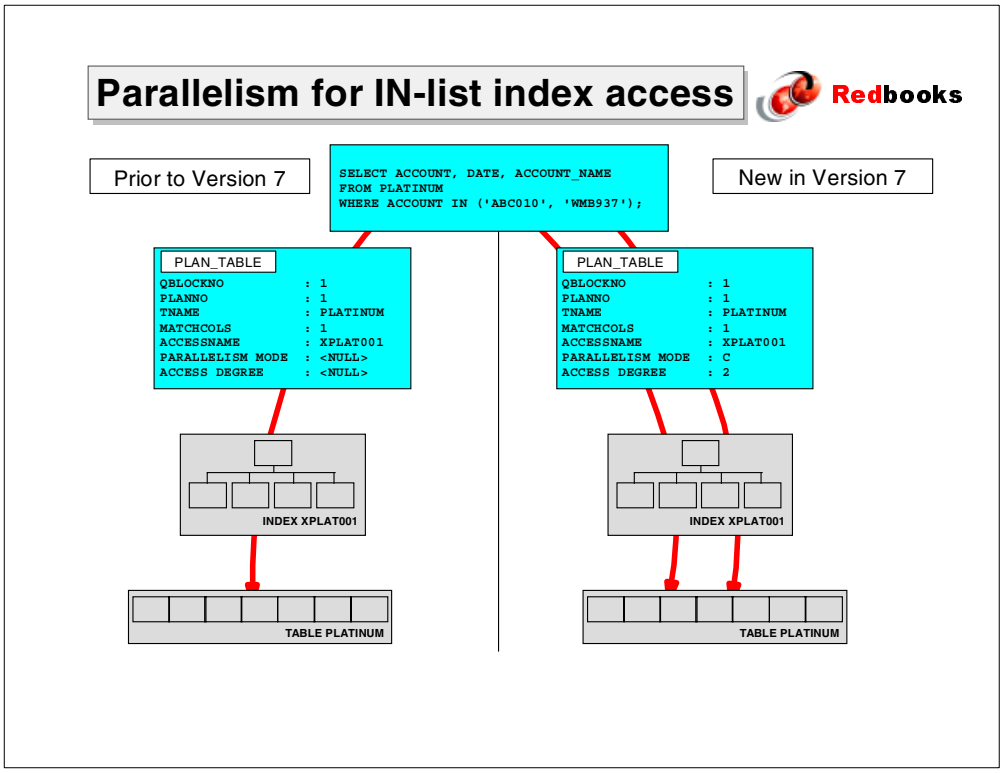
The z/Series 900 and z/OS V1R1 deliver immediate performance benefits to DB2. All supported releases of DB2 are compatible with z/OS. 64-bit support in z/OS V1R1 provides 64-bit arithmetic and 64-bit real addressing support, currently increasing the limit to 256 GB. Later releases of z/OS will support 64-bit virtual addressing which means support for 16 exabyte address spaces. OS/390 V2R10 also delivers 64-bit real addressing support.

All DB2 Versions will be able to receive immediate benefits from the increased capacity of central memory—up to 256 GB, from the current limit of 2 GB. More DB2 subsystems can be supported on a single OS image, without significant paging activity. The increased real memory provides performance and improved scaling for all customers.

Data spaces have some advantages over hiperpools: you can read and write to a dataspace with direct I/O. Data spaces are byte addressable, whereas hiperpools are block addressable. You can have larger bufferpools with data spaces than hiperpools: a data space can span up to 32 GB for a 4-KB page size buffer pool and 256 GB for a 32-KB page size.  Also, in high concurrency environments, data space pools tend to have significantly less latch contention than virtual pool and hiperpool combinations.

With Versions 6 and 7 of DB2, the key benefit of scaling with the larger memory is in the use of data spaces, first shipped with DB2 V6 architecture. Please refer to *DB2 UDB for OS/390 Version 6 Performance Topics,* SG24-5351 for considerations on the usage of data spaces. However, be aware that there is a performance penalty when the data spaces are not 100% backed by real storage. You need z/OS and OS/390 R10 large real storage support when running on zSeries servers to use large data space pools with good performance.

Data spaces allow for buffer pools and EDM pool global dynamic statement cache to reside outside of the DBM1 address space. More data can be kept in memory, which can help reduce I/O time. Customers who are reaching the 2GB address space limit should consider migrating to the DB2 and zSeries solution.



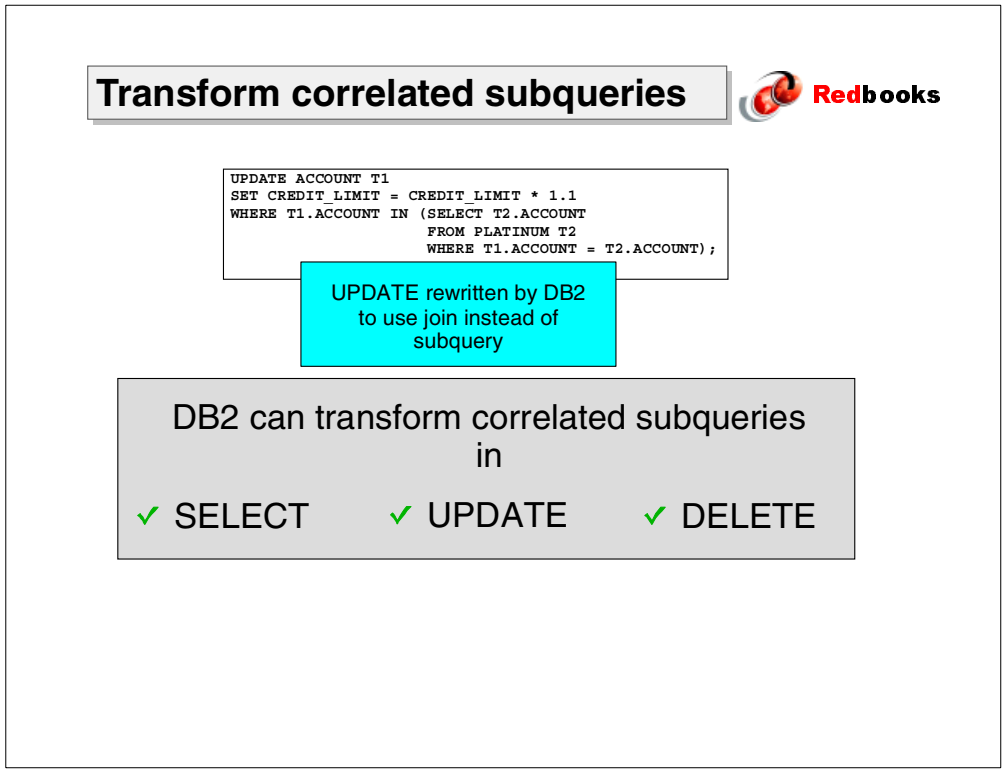

## 7.2 Parallelism for IN-list index access

DB2 V3 has started to introduce parallelism, and with each new version, DB2 has extended the usage of parallelism. DB2 V6 introduced the parallelism for IN-list access for the inner table of a parallel group (ACCESSTYPE=N). See *DB2 UDB for OS/390 Version 6 Performance Topics*, SG24-5351 for details. DB2 could only use parallelisms when a table was joined which had an IN-list coded in the predicate and DB2 had decided that the table was to be the inner table. The measurements show good elapsed time improvements in this specific case, with minimal CPU degradation due to the multi task handling.

DB2 V7 has removed this restriction and now allows parallelism on a IN-list predicate whenever it is supported by an index and parallelism is available. The restriction is removed not only for outer table, but also for access to a single table, as it is illustrated in the foil. This has the potential of improving performance significantly over the non-parallel performance, depending on the number of parallel processes available. In the above example we can see that for each element of the IN-list a parallel process will be used. The evidence of any parallelism can be seen in the EXPLAIN output of the query to the PLAN_TABLE. The above example will have two processes using query CP parallelism. This is the only externalization of this enhancement.

The parallel degree chosen is:

- For I/O intensive queries, the smaller value between the number of entries in the IN-list and the number of partitions
- For CPU intensive queries, the smaller value between the number of entries in the IN-list and the number of CPU engines.




## 7.3 Transform correlated subqueries

Prior to V7 DB2 had the facility to transform subqueries to a join between the result table of the subquery and the result table of the outer query. This was done to improve performance because correlated subqueries would always be processed as stage 2 predicates, whereas joins could be processed as stage 1.

V7 of DB2 has extended this process to include UPDATE and DELETE statements along with SELECT statements that have correlated subqueries using the comparison operators IN, =ANY, =SOME, and EXISTS. This gives DB2 the facility to process UPDATE and DELETE statements using the more efficient joins rather than being restricted to correlated subselects.

DB2 will transform the correlated subquery to joins if the following conditions are met:

- Translation to a join would not produce redundant rows. By way of example, the statement:

```
SELECT T1.ACCOUNT, T1.ACCOUNT_NAME
FROM ACCOUNT T1
WHERE T1.ACCOUNT IN (SELECT T2.ACCOUNT
                     FROM PLATINUM T2
                     WHERE T1.ACCOUNT = T2.ACCOUNT)
```

  Here, the PLATINUM table potentially contains multiple entries for each account, and cannot be transformed to a join, as it would produce a row for each entry in the PLATINUM table. The original query would produce a single row for each account that had any entry in the PLATINUM table.

The UPDATE statement in the foil was processed by EXPLAIN in a subsystem running V6, where transformation of updates did not occur, and also in V7.

The result of the V6 EXPLAIN was:

```
QBLOCKNO  PLANNO  TNAME               METHOD  QBLOCK_TYPE  SORTC_UNIQ
--------+--------+--------+---------+--------+---------+---------
       1       1  ACCOUNT                  0  UPDATE       N
       2       1  PLATINUM                 0  CORSUB       N
```

Each table is processed in a separate QBLOCKNO, the METHOD is 0 and the QBLOCK_TYPE is CORSUB showing that the PLATINUM table is being processed as a correlated subquery and therefore as a stage 2 predicate.

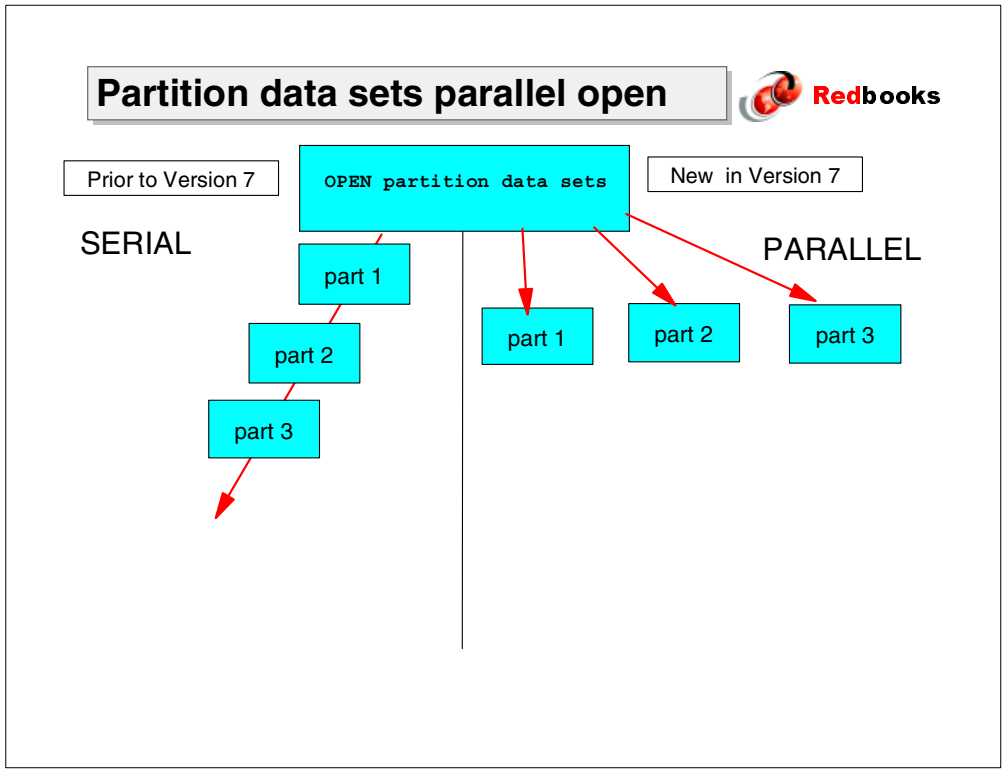
The result for V7 shows that a transformation has taken place.

```
QBLOCKNO  PLANNO  TNAME               METHOD  QBLOCK_TYPE  SORTC_UNIQ
--------+--------+--------+---------+--------+---------+---------
       1       1  ACCOUNT                  0  UPDATE       N
       1       2  PLATINUM                 2  UPDATE       N
```

In this version both tables are processed in the same QBLOCKNO and the METHOD shows that a merge scan join is to be used.

Notes:

- The join does not create a statement that accesses more than 15 tables.
- The FROM clause of the subquery only references a single table.
- The SELECT statement's outer query, or the UPDATE or DELETE statement accesses only one table.
- The left side and result of the subquery have the same format and length.
- The result of the subquery is not grouped by using the GROUP BY, HAVING clauses or a function call such as MAX.
- The subquery does not use a user defined function.
- The predicates of the subquery are not OR'd.
- The predicates in the subselect that correlate statement are stage 1 predicates.
- The subquery does not reference the target table of the UPDATE or DELETE.
- The SELECT statement does not use the FOR UPDATE OF clause.
- The UPDATE or DELETE statements do not use the WHERE CURRENT OF clause.
- The subquery does not contain a subquery.
- Parallelism is not enabled.
- If DB2 determines that very few rows qualify in the outer query, no transformation will take place because it will not pay off. Performance will improve most for large outer query results (which needed table space scan) and small subquery results. It has little to do with the size of the tables. It is even better that using parallelism because it requires less I/O and CPU.
- The PLAN_TABLE can be used to see if a transformation has occurred or not.




## 7.4 Partition data sets parallel open

For DB2 subsystems with a large number of data sets the open and close operations can be lengthy.

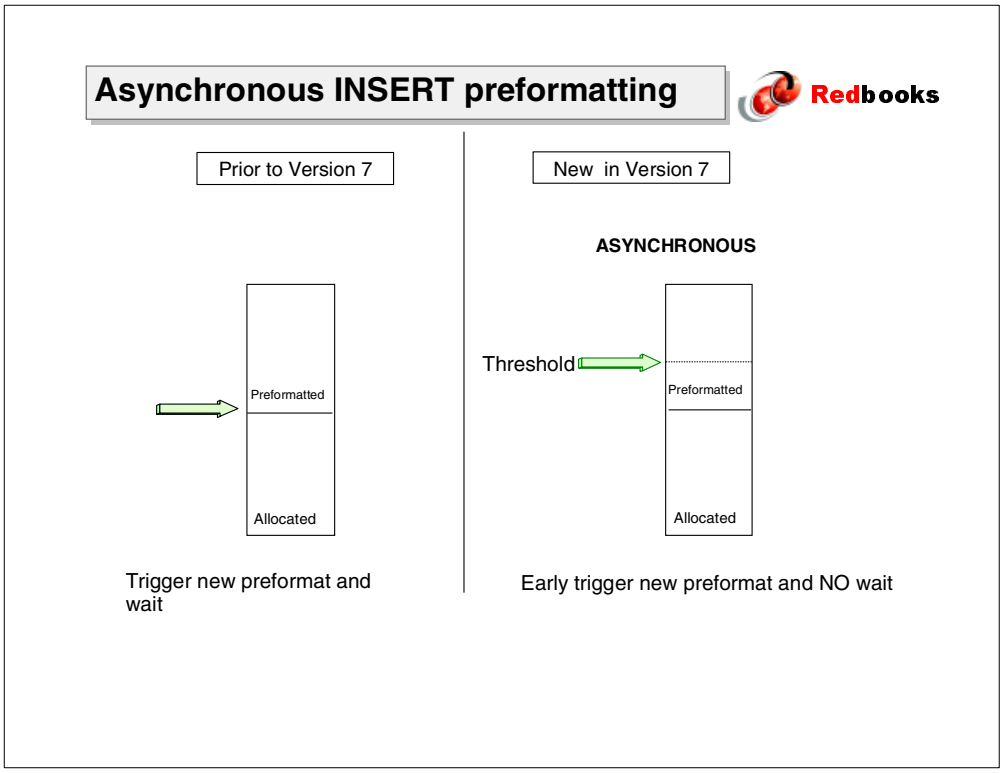
DB2 V5 used only one task to perform this operation. The tasks were increased to 10 through maintenance.

DB2 V6 increased the number of tasks for open and close of database data sets to 20.

However, this did not apply to the concurrent open and close of partitions belonging to the same table space and index space. With DB2 V7, up to 20 tasks can concurrently handle the open and close of all data sets of a partitioned table space or index.

For data sharing environments the Extended Catalog Sharing feature of DFSMS 1.5 or OS/390 V2R7 uses the Coupling Facilty to cache the ICF catalog sharing control data and provides elapsed time reduction for open and close operations.

.




## 7.5 Asynchronous INSERT preformatting

During insert processing, DB2, once it has exhausted the search algorithms for free space, currently synchronously preformats new pages at the end of the data before they can be used. This preformatting is generally performed in blocks of two cylinders (assuming the allocation was defined in cylinders) as pages are needed. Especially for insert-intensive applications, the delays caused by the preformatting process can affect the performance. To avoid the performance impact and also to achieve consistent execution times, the only option to use was the preformatting technique introduced with DB2 V5 as an option of the LOAD and REORG utilities. If the final size of the table space can be predicted, and the access to the data has a high insert-to-read ratio, especially during the fill-up period, this technique should be considered.

DB2 V7 improves the performance of INSERTs by asynchronously preformatting allocated and not yet formatted but allocated pages. When a new page is used for INSERT, and that page is located within an internally predefined range from the end of the formatted pages, DB2 preformats the next set of pages. This asynchronous preformatting ensures that the INSERT will not wait for a page to be formatted and it also reduces the need to use the preformatting option at LOAD time described in more details below.

However, once the preformatted space is used up and DB2 has to extend the table space allocation, normal data set extending and preformatting still occurs.

### *Preformatting during LOAD*

When DB2's preformatting delays impact the performance or execution time consistency of applications that do heavy insert processing, and if the table size can be predicted for a business processing cycle, you may want to consider using the PREFORMAT option of LOAD and REORG. If you preformat during LOAD or REORG, DB2 does not have to preformat new pages during execution. When the preformatted space is used and when DB2 has to extend the tablespace, normal data set extending and preformatting occurs.

You should specify PREFORMAT when you want to preformat the unused pages in a new table space, or reset a table space (or partition of a table space) and all index spaces associated with the table space after the data has been loaded and the indexes built. The PREFORMAT option can be specified as a main option of the LOAD utility or as a suboption of the PART option in the "into table spec" declaration. In the REORG utility, the PREFORMAT option is specified as an option within the "reorg options" declarations.

Following are some examples of the LOAD and REORG statements with the PREFORMAT option:

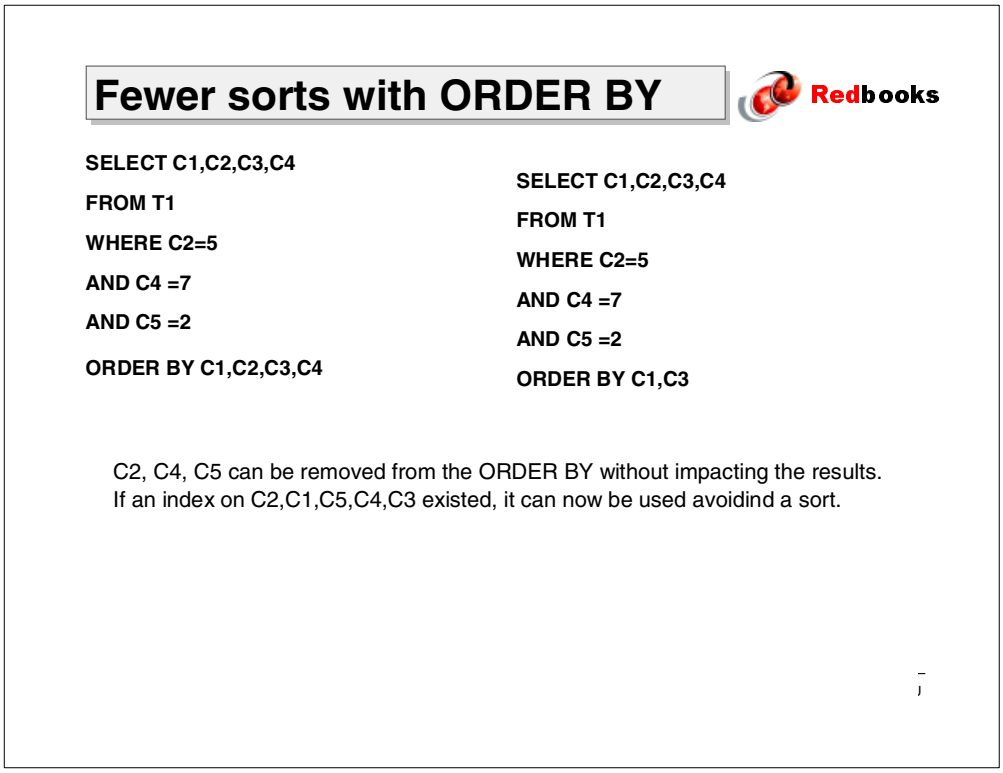
LOAD

– LOAD DATA PREFORMAT INTO TABLE tname
– LOAD DATA INTO TABLE tname PART 1 PREFORMAT

REORG

– REORG TABLESPACE tsname PREFORMAT
– REORG INDEX ixname PREFORMAT
– REORG TABLESPACE tsname PART 1 PREFORMAT
– REORG INDEX ixname PART 1 PREFORMAT

After the data has been loaded and the indexes built, the PREFORMAT option on the LOAD and REORG utility command statement directs the utilities to format all the unused pages, starting with the high-used relative byte address (RBA) plus 1 (first unused page) up to the high-allocated RBA within the allocated space to store data. After preformatting, the high-used RBA and the high-allocated RBA are equal. Once the preformatted space is used up and DB2 has to extend the table space, normal data set extending and formatting occurs.



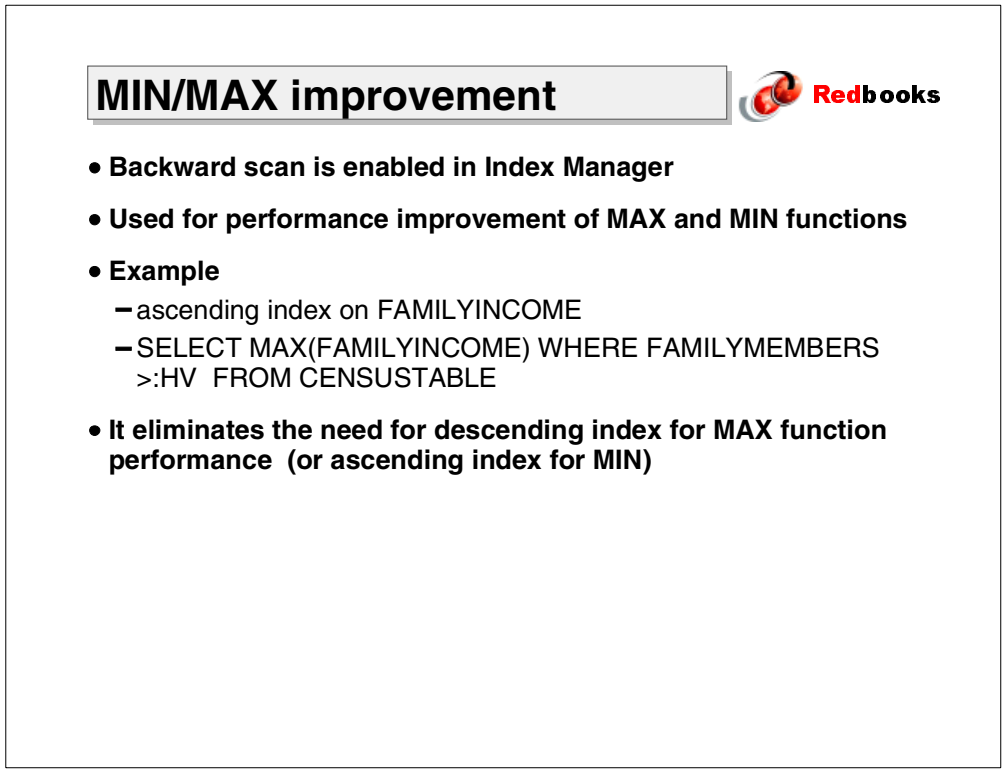

## 7.6 Fewer sorts with ORDER BY

ORDER BY execution is improved with DB2 V7.

DB2 can now perform fewer sort operations for queries that have an ORDER BY clause. Previously, when a query had a WHERE clause with a predicate in the form of COL=*constan*t, DB2 performed a sort when the column was included in the ORDER BY clause or an index key. Now, when a column has such a predicate, DB2 can consider it to be a constant in the result table. As a constant, it has no effect on ordering, and DB2 can then remove it from the ORDER BY list or index key, possibly avoiding a sort.

For example, the two queries in the foil provide the same results. The removal of C2 and C4 from the ORDER BY list does not change the order of the results.

If an index exists on (C2,C1,C5,C4,C3), DB2 logically removes C2, C4, and C5 from the index leaving the ordering columns C1 and C3. With the columns removed, it is easy to see that index supports the ordering that is required by the ORDER BY.

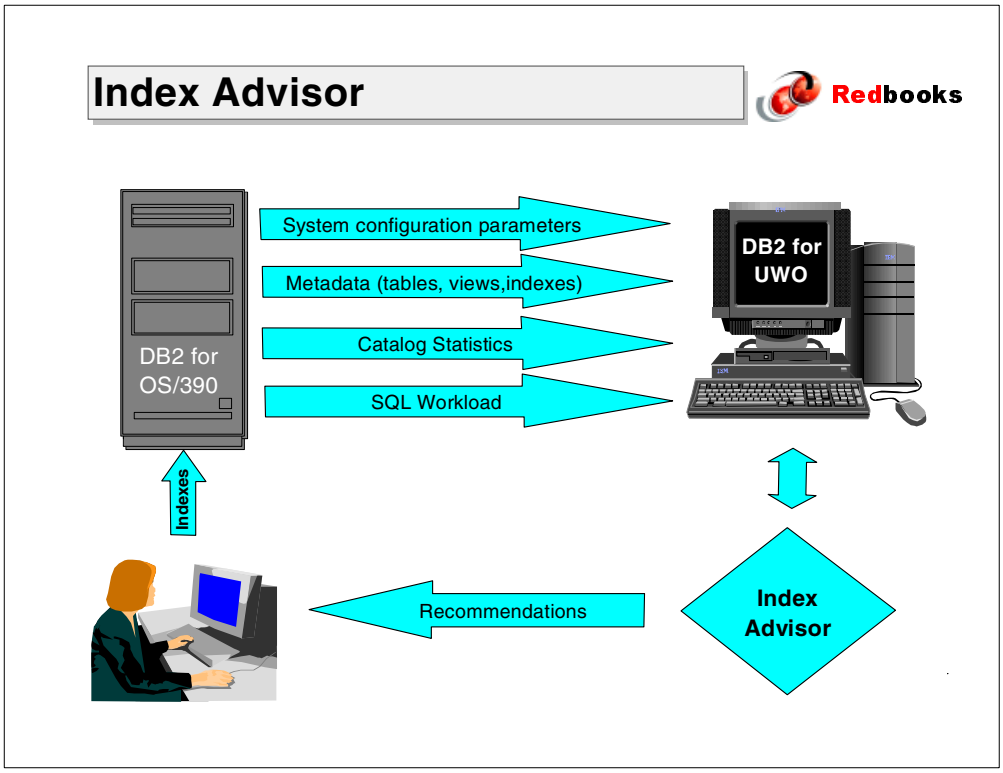



## 7.7 MIN/MAX set function improvement

The implementation of scrollable cursors has introduced the ability to use one index instead of two to be able to scroll forwards and backwards efficiently.

Performance can be improved for the Max or Min function, or an index may not need to be created. With prior releases, an ascending index can be used for fast access to the MIN, or a descending index can be used for fast access to the MAX.

With this change, either ascending or descending indexes can be used for both MIN and MAX.




## 7.8 Index Advisor

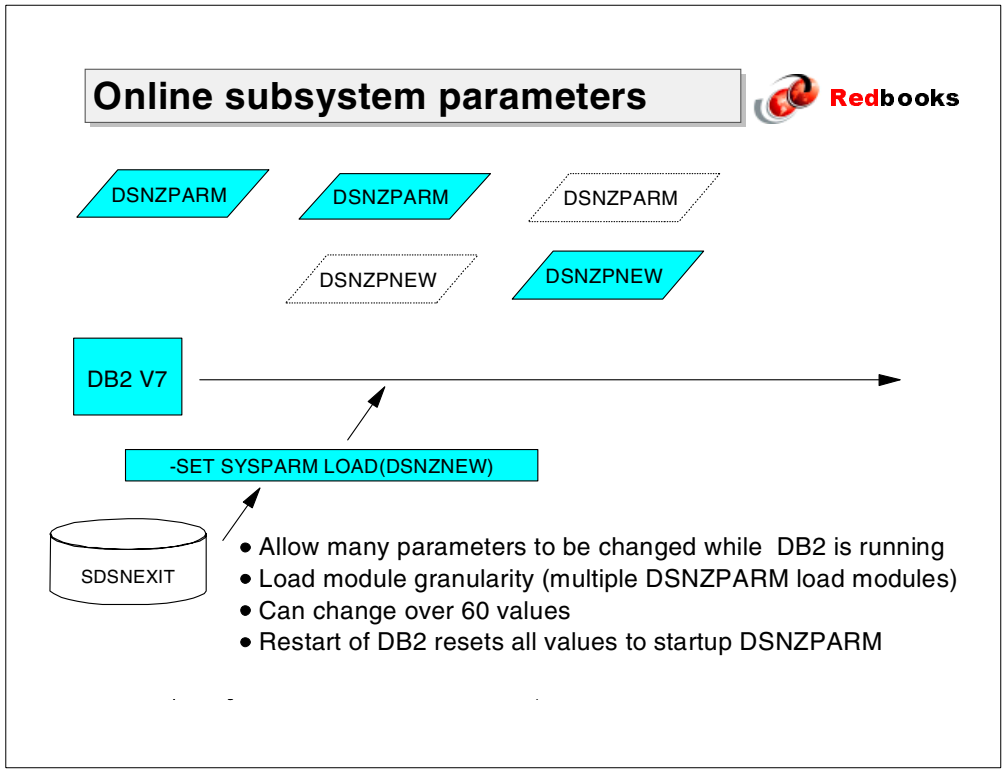
Index Advisor is an extension of the optimizer in DB2 for UNIX, Windows, and OS/2 (UWO in the picture) platforms. It provides index recommendations based on:

- All tables and views in the current database
- Existing indexes
- The SQL workload
- Disk space specified for indexes
- Time specified to complete analysis

The recommendations are accompanied by derived statistics on before and after costs, and the sample DDL to create the indexes.

The function is available from the DB2 Control Center through the Index SmartGuide under the Create Index tab.

Starting with the Version 7 of the DB2 Family of products, it is possible to model and analyze the DB2 for OS/390 subsystem on the other platform using the tools provided for capturing metadata, catalog statistics, and SQL workload, and transferring them to the Index Advisor's DB2. Since the process alters the configuration settings, the Index Advisor process should execute on a dedicated DB2 system.




## 7.9 Online subsystem parameters

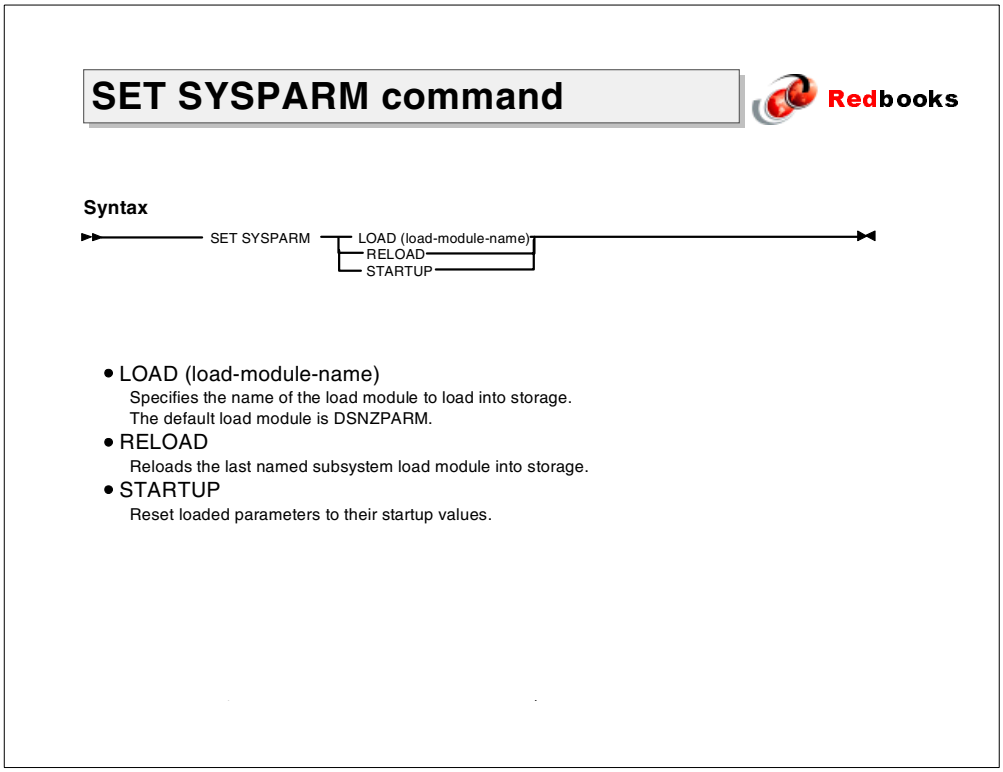
With a growing number of customers utilizing DB2 in a 24x7x52 mode, the need has been growing for online update of the major DB2 subsystem parameters. These parameters provide most of the basic configuration data used by DB2. Currently, the users can only load their parameters of choice once at DB2 startup by specifying a specific DSZPARM member in the command:

```
-START DB2 PARM(DSNZPARMx)
```

With DB2 V7, the new `-SET SYSPARM` command is introduced to dynamically reload the DSNZPxxx (subsystem parameters) load module.

All the parameters of the DSN6ARVP macro can be changed, and a large number from the DSN6SYSP and DSN6SPRM macros can be changed.

For a detailed but preliminary listing of the changeable parameters, please refer to Appendix A, “Updatable DB2 subsystem parameters” on page 517. Note that you must verify the current list of updatable parameters based on the maintenance level of your DB2 subsystem.




### 7.9.1 SET SYSPARM command

The DB2 command SET SYSPARM lets you change subsystem parameters online. This command can be issued from an MVS console, a DSN session under TSO, a DB2I panel (DB2 Commands), an IMS or CICS master terminal, or a program using the DB2 instrumentation facility interface (IFI). The data sharing scope is at member level.

***Authorization***

To execute this command, the privilege set of the process must include SYSOPR, SYSCTRL, or SYSADM authorities. DB2 commands that are issued from an MVS console are not associated with any secondary authorization IDs.

***Options description***

**LOAD** (load-module-name)

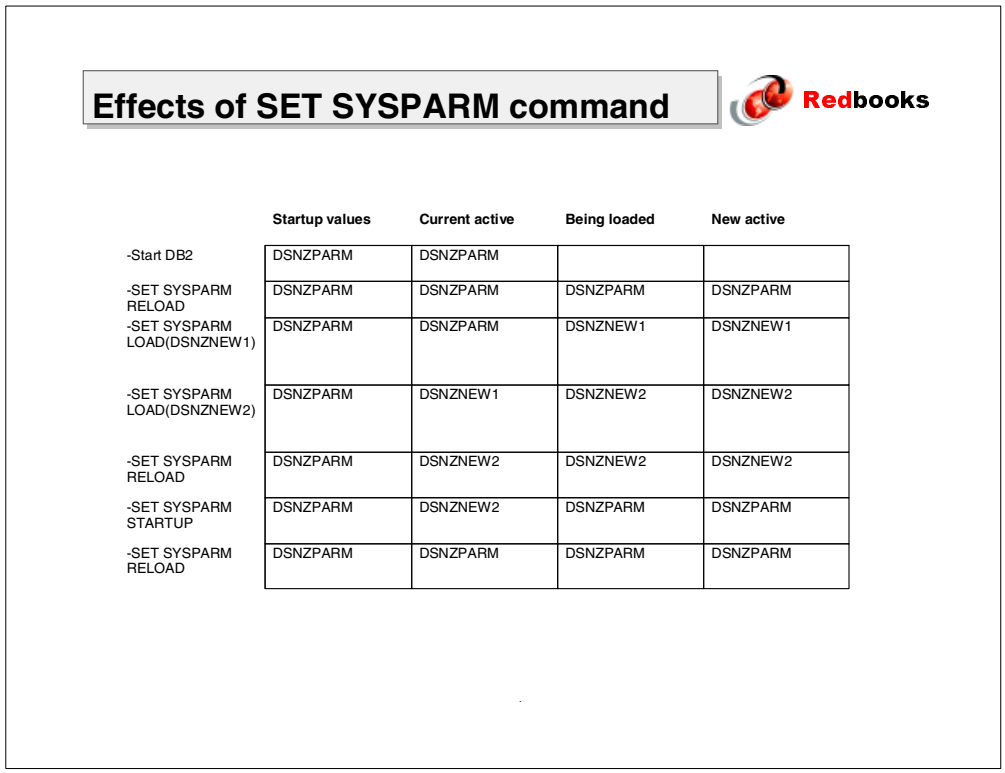
Specifies the name of the load module to load into storage. The default load module is DSNZPARM.

**RELOAD**

Reloads the last named subsystem parameter load module into storage.

**STARTUP**

Resets loaded parameters to their startup values.




## 7.9.2 Effects of SET SYSPARM command

DB2 will maintain up to three DSNZPxxx modules in storage at any one point in time:

- The load module when DB2 was started
- The load module currently active and in use (it could be the same as startup)
- The load module just reloaded by DB2 (for example, the new parameters)

The `-SET SYSPARM RELOAD` command always reloads a load module with the same name as the current active one. This could be used when you decide to have always the same parameter load module name when reassembling and relinking instead of having several parameter load modules with different names for different behaviors.

Please note that with a restart of DB2 all subsystem parameter values will be taken from the parameter load module specified during DB2 startup. This means that all online subsystem parameter changes will be “reset” to the values specified in the startup parameter load module.

The `-SET SYSPARM STARTUP` command resets the parameters to the values they had at startup time. Those values will be taken from the copy of the load module in storage. This means that even if you have re-assembled and re-linkedit the load module used during startup of DB2 with changed values, and issued a `-SET SYSPARM STARTUP` command, the updated parameters will not take affect until the next `-SET SYSPARM LOAD` or `RELOAD` command.

#### 7.9.2.1 Generating and loading new parameters load module

For generating a new DSNZPARM load module, you must execute the following steps:

1. Run through the installation process in update mode to produce a new DSNTIJUZ installation job.
2. Execute the first 2 steps of DSNTIJUZ to assemble and link edit the new DB2 parameter load module.
3. Issue the `-SET SYSPARM` command to activate the new load module.

The first 2 steps are business as usual (and good practice) when changing DB2 system parameters. Then, instead of stopping and starting DB2 you can activate the new parameters by issuing the `-SET SYSPARM` command which loads the new load module.

#### 7.9.2.2 Displaying current settings

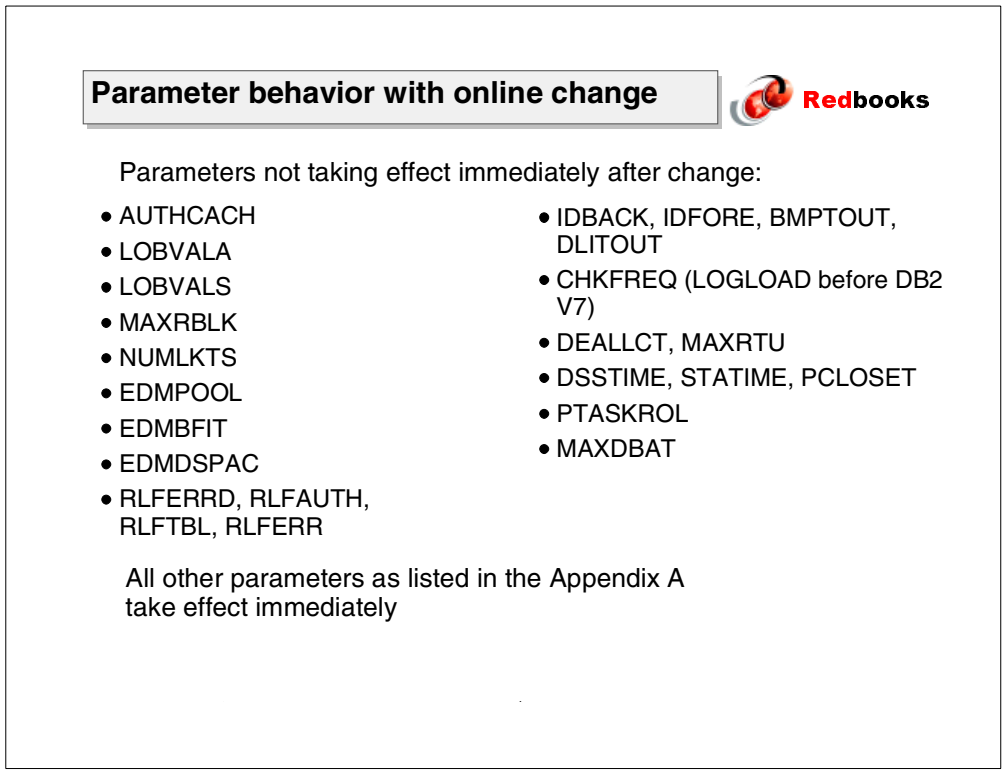
With the sample program DSN8ED7 you can generate a list of the current DB2 parameters settings. DSN8ED7 calls the stored procedure DSNWZP, a DB2 provided stored procedure used also by Control Center and Visual Explain, which returns the current settings of your DB2 subsystems parameters, and formats the results in a report.

DB2 sample application job DSNTEJ6Z prepares and executes the sample program DSN8ED7. Before running DSN8ED7 you must create the stored procedure DSNWZP (installation job DSNTIJSG).

***DSN8ED7 sample output:***

```
DSN8ED7: Sample DB2 for OS/390 Configuration Setting Report Generator

Macro    Parameter        Current
Name     Name             Setting
-------- ---------------- ----------------------------------------
DSN6SYSP AUDITST          00000000000000000000000000000000
DSN6SYSP CONDBAT          0000000064
DSN6SYSP CTHREAD          00070
...
                    Description/                          Install  Fld
                    Install Field Name                    Panel ID No.
                    ------------------------------------ -------- ----
                    AUDIT TRACE                           DSNTIPN     1
                    MAX REMOTE CONNECTED                  DSNTIPE     4
                    MAX USERS                             DSNTIPE     2
...
```

**Note:** This is a very brief sample output of DSN8ED7 to show what the report looks like. For a sample listing, to be verified against the current code level version, please refer to Appendix A, “Updatable DB2 subsystem parameters” on page 517.




### 7.9.3 Parameter behavior with online change

For most parameters, the online change will be transparent, with the change taking effect immediately. There are a few parameters for which this is not the case, because of the type of functions that they impact. The behavior exhibited by the system upon changes to these parameters is discussed below:

- AUTHCACH

  Changing the plan authorization cache only takes effect for new threads. Existing threads are not modified.

- LOBVALA

  Changing the user LOB value storage does not affect any currently running agents that have already acquired storage from data spaces for LOBs. It will only affect new agents.

- LOBVALS

  The system LOB value storage will be examined whenever an agent attempts to allocate storage from a data space pool for LOBs. If the parameter change decrements the value of LOBVALS such that the current amount of storage allocated for the system is greater than the new LOBVALS value, an RNA SQLCODE is not issued until the next attempt by an agent to acquire storage.

- MAXRBLK

  If the RID pool size is decremented and as a result, the current number of RID blocks allocated is greater than the newly specified value, the new MAXRBLK value will not take affect until a user attempts to allocate another RID block. When the value is decremented, an attempt will be made to contract the storage pool (which can only be contracted if there are empty segments).

- NUMLKTS

  The number of locks per tablespace will not change immediately for agents that have already been allocated when the parameter is changed.

- EDMPOOL

  The EDM pool storage parameter can be used to increase or decrease the size of the EDM pool. The initial allocation will work as it did prior to online change - getting a single block of storage the size of the request. Using dynamic parameters the user can increase the EDM pool from the initial size or reduce the EDM pool back to the initial size. An attempt to reduce the EDM pool size below the initial value will be rejected and indicated by message DSNG001I.

  The changes in the EDM pool size will be done in 5M increments rounding up to the next 5M boundary. For example, if the EDM pool is 40M and the request is to increase it to 58M it will be increased to 60M.

  If insufficient virtual storage is detected when expanding the EDM pool, DB2 will issue a warning message (DSNG003) and increase the pool as large as available space allows. A informational message (DSNG002I) will indicate the amount of the allocated size.

  Note that a contiguous EDM pool is the most efficient and using the SET SYSPARM command to increase the EDM pool size may not be contiguous.

  When contracting the EDM pool some storage may be in use. A message will be issued indicating how much storage was released and how much is pending. The pending storage will be released when it is no longer accessed. The changes in the EDM pool can be monitored using the EDM statistics or the 106 trace record.

  ```
  DSNG002I EDM stype HAS AN INITIAL SIZE isize, REQUESTED SIZE rsize, AND AN
  ALLOCATED SIZE asize.
  Explanation: This message is issued in response to a request to increase or
  decrease the EDM 'stype storage.
  The INITIAL SIZE is the size prior to the request for a change.
  The REQUESTED SIZE is the new desired size.
  The ALLOCATED SIZE is the current size immediately available.
  - When increasing the EDM Pool or EDM DATASPACE the ALLOCATED SIZE is the
  storage available to satisfy the
  request.
  - When decreasing the EDM Pool or EDM DATASPACE the ALLOCATED SIZE is the
  INITIAL size reduced by the
  amount that could be released immediately.
  -- When the ALLOCATED size is larger than the REQUESTED size the difference
  is marked to be released when it is no longer referenced.
  System Action: Processing continues.
  ```

  Using the SET SYSPARM command to decrease the size of the EDM pool may involve a wait for system activity to quiesce, and therefore the results may not be instantaneous.

- EDMBFIT

  Large EDM better fit parameter is used to determine the algorithm to search the free chain for large EDM pools (greater than 40M). This change will only affect new free chain requests.

- EDMDSPAC

  The EDM pool data space parameter can be used to increase or decrease the size of the Data Space storage used for dynamic statements. If the initial value was zero when DB2 was started, this parameter cannot be changed. The data space storage can only be increased up to the maximum size specified at DB2 start time. Other than those restrictions this parameter behaves the same as the EDMPOOL parameter.

- RLFERRD, RLFAUTH, RLFTBL, RLFERR

  After a change to the resource limit facility parameters, for the dynamic statements SELECT, INSERT, UPDATE, and DELETE, the change takes effect after the resource limit facility is restarted using the -START RLIMIT command. The change will not be seen for dynamic DML statements issued before the RLF is refreshed with the -START RLIMIT command.

- IDBACK, IDFORE, BMPTOUT, DLITOUT

  Any change(s) to these parameters (max batch and max TSO connect, IMS BMP and DLI batch timeout) will not take effect until the next create thread request. Also, any threads currently executing will not be updated with the new value(s).

- Archive log parameters

  If an off load (archiving an active log data set) is active at the time of a related parameter change, the off load will continue with the original parameter values. The new archive log parameter values will not take effect until the next off load sequence.

- CHKFREQ (LOGLOAD before DB2 V7)

  The checkpoint frequency parameter can be modified by the -SET LOG COMMAND which will override the value in the subsystem parameter load module. If a new parameter is activated through the online change, it will override the last -SET LOG value. The current value can be displayed with the -DISPLAY LOG command.

- DEALLCT, MAXRTU

  The behavior for the archive deadlock period and the maximum allocated tape units parameters is similar to check point frequency (CHKFREQ). These values can be modified with the -SET ARCHIVE command. Changing the parameter online will override the last -SET ARCHIVE value. The current values can be displayed with the -DISPLAY ARCHIVE command.

- DSSTIME, STATIME, PCLOSET

  The data set statistics and the statistics interval time parameters will be read from the updated control block when the intervals are being set for the next timer pop for the statistics. So the existing interval for each must expire before the new value will be set.

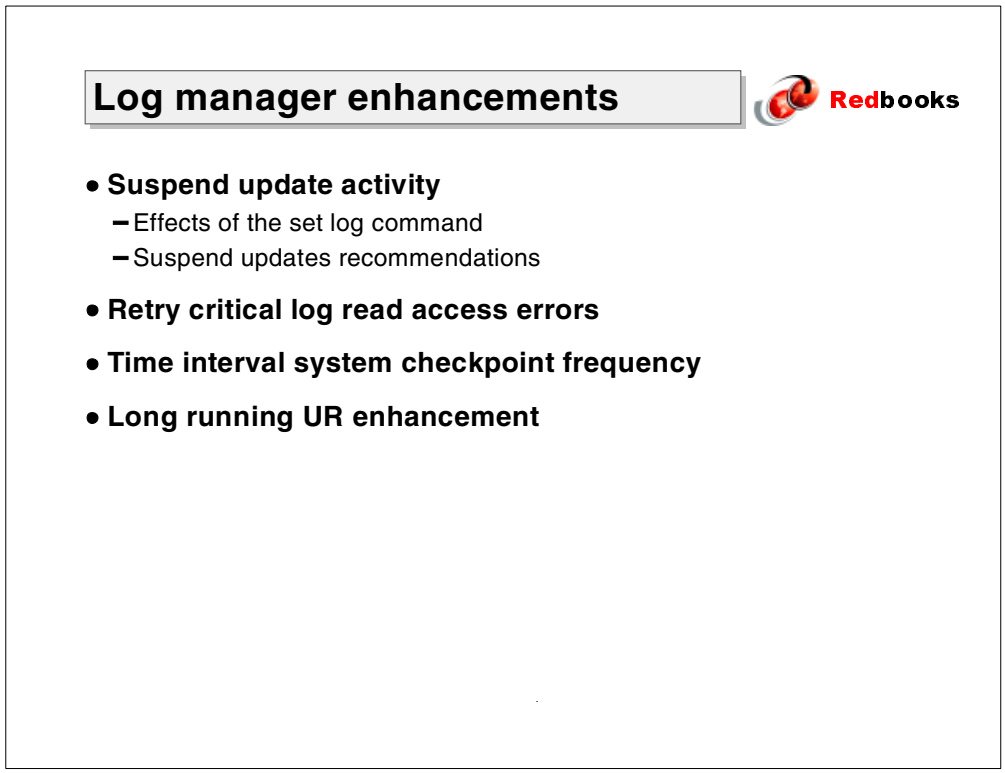
- PTASKROL

  This value, indicating whether to roll up query parallel task accounting trace records into the originating task accounting trace or not, will be read from the updated control block when the first child record is summed up for the child task accounting data rollup. The value will be saved in the parent accounting block so that the behavior for any parent and all its child tasks will be consistent.

- MAXDBAT

  When the maximum remote active value is increased, current agents that were suspended prior to the increase will not be resumed until existing active agents terminate or become inactive (see the "DDF THREADS" INACTIVE specification in the DSNTIPR installation panel).

For a detailed list and description of the DB2 V7 subsystem parameters names please refer to the *DB2 UDB for OS/390 and z/OS Version 7 Installation Guide,* GC26-9936.




## 7.10 Log manager enhancements

In the following sections we describe log manager updates that will help you with DB2 operations:

- Suspend update activity

  New commands allow to temporary freeze logging activity so that a consistent almost instantaneous copy of your data can be taken. This enhancement was made already available to Version 6 by APAR.

- Retry critical log read access

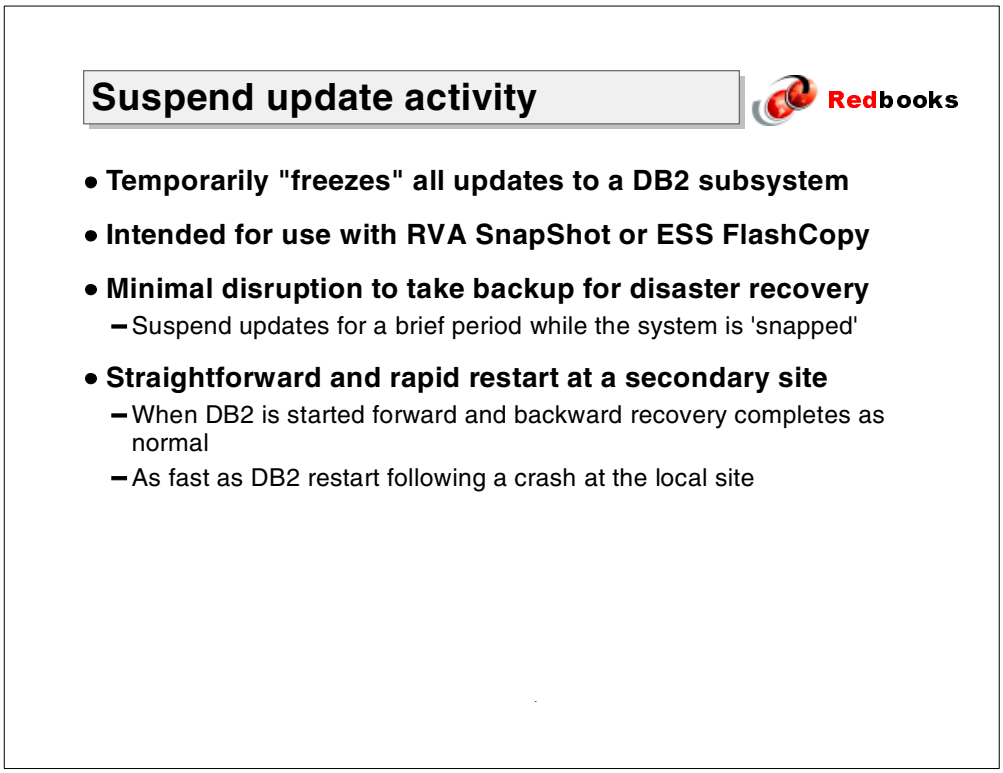
  It prevents DB2 termination for temporary log read access error during ‘must-complete’ operations

- Time interval system checkpoint frequency

  Option to specify a range of minutes between system checkpoints.

- Long running UR enhancement

  New warning message based on number of log records written by in-flight unit of recovery (UR)




### 7.10.1 Suspend update activity

This new feature, introduced by maintenance also in DB V6, enables you to suspend all updates to a DB2 subsystem. This allows you to take a snapshot of the entire system for local or remote site recovery with minimal impact on the availability of the system, and to restart with a consistent copy of your DB2 data.

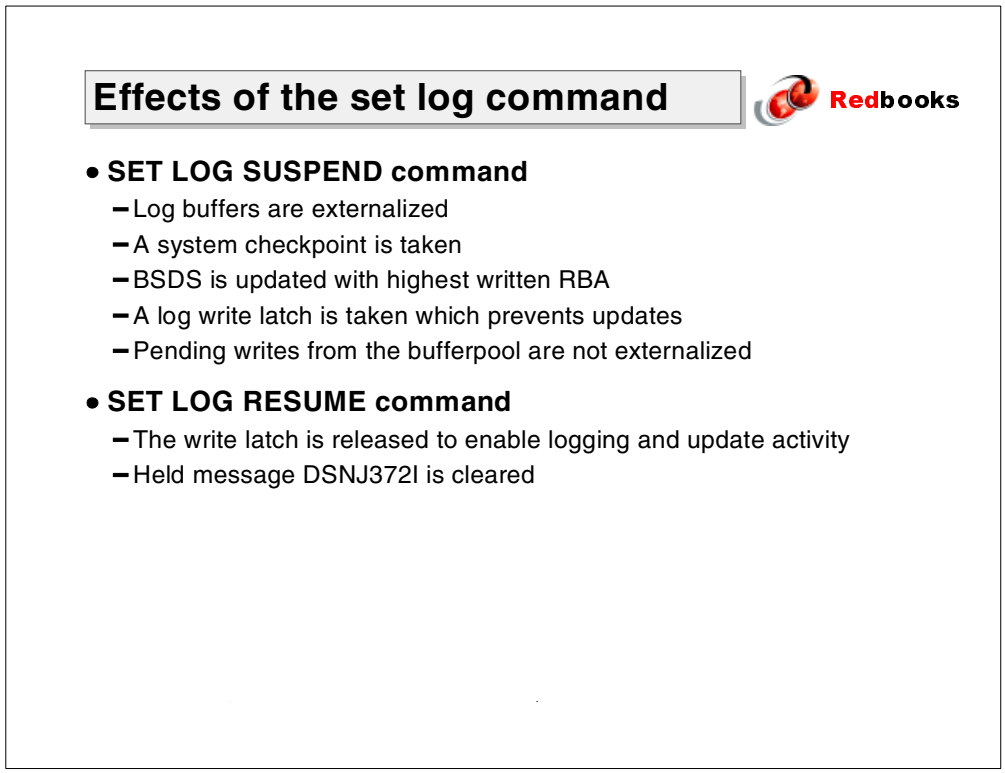
It is designed to be used in conjunction with external copy of DB2 data such as provided by RVA SnapShot or Enterprise Storage Server (ESS) FlashCopy technology.

Should a disaster occur, the snapshot (here used to mean either of the two techniques mentioned) can be used to recover the system to a point of consistency simply by starting DB2. Offsite recovery is as fast as a normal DB2 restart following a crash at the local site as it just requires a start of the system which will resolve inflight units of recovery.

The snapshot can also be used to provide a fast, consistent copy of your data for reasons different from recovery; one example is to periodically snap an entire system to enable point in time query with minimal operational impact.

This function is described in the redbook *DB2 UDB Server for OS/390 Version 6 Technical Update*, SG24-6108. More details are available from the Web site:

`http://www.ibm.com/storage/hardsoft/diskdrls/technology.htm`




### 7.10.1.1 Effects of the set log command

Here, we provide a description of the DB2 actions and messages when issuing the log suspension commands.

***Effects of SET LOG SUSPEND***

The `-set log suspend` command performs the following actions:

- All unwritten log buffers are externalized to the log.

  The `-set log suspend` command flushes only the log buffers. Pending writes from the bufferpools are not externalized.

  For a data sharing group, the command should be issued for each member.

- A system checkpoint is taken.
- The BSDS is updated with highest written RBA. This guarantees that PGLOGRBA (which records the RBA or LRSN of the last page update) in all the pagesets is no higher than the highest written RBA on the log when copied.
- A log write latch is taken which prevents further log records from being written until either the `-stop db2` or `-set log resume` commands are issued. If DB2 terminates abnormally, the latch is lost and update activity is permitted on restart.

- The following message will be issued to identify the subsystem name and the RBA at which log activity has been suspended. The message is held until update activity is resumed.

```
 =DB2A SET LOG SUSPEND
*DSNJ372I = DB2A DSNJC09A UPDATE ACTIVITY HAS BEEN 788
 SUSPENDED FOR DB2A AT RBA 000002AA3D4C
 DSN9022I  =DB2A DSNJC001 '-SET LOG' NORMAL COMPLETION
```

***Effect of SET LOG RESUME***

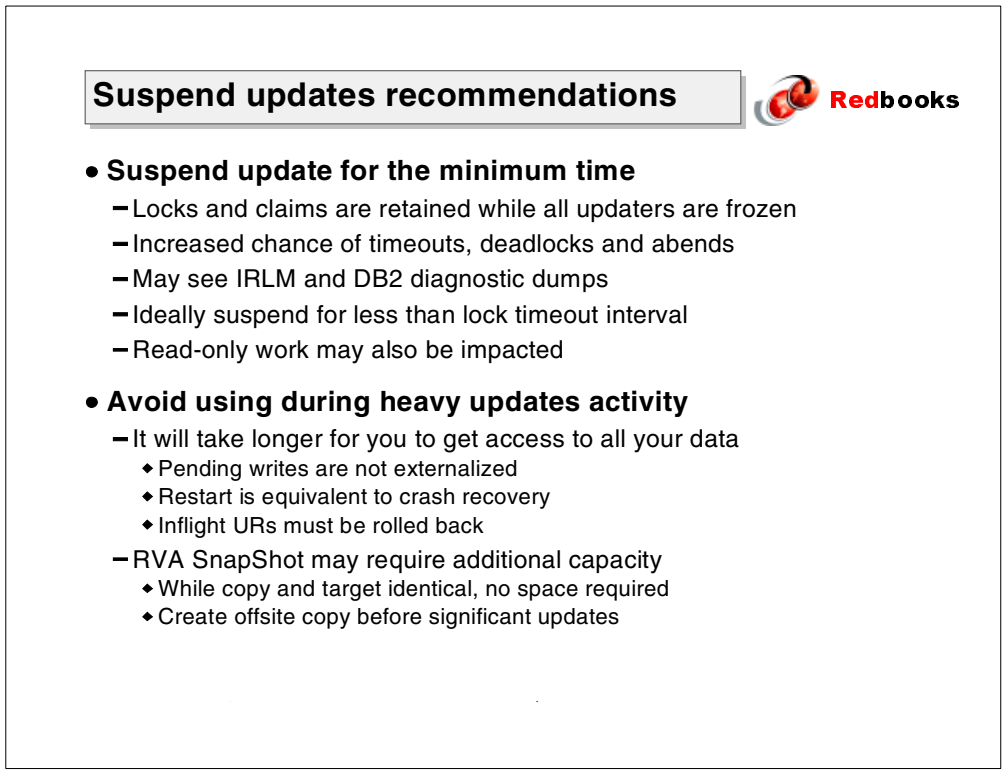
The following actions are performed when the `-set log resume` command is issued:

- The write latch is released to enable logging and update activity
- Held message DSNJ372I is cleared
- The following output is issued:

```
=DB2A SET LOG RESUME
DSNJ373I  =DB2A DSNJC09A UPDATE ACTIVITY HAS BEEN RESUMED FOR DB2A
DSN9022I  =DB2A DSNJC001 '-SET LOG' NORMAL COMPLETION
```

This function is described in the redbook *DB2 UDB Server for OS/390 Version 6 Technical Update*, SG24-6108. More details are available from the Web site:

`www.storage.ibm.com/hardsoft/diskdrls/technology.htm`




### 7.10.1.2 Suspend updates - recommendations

We recommend observing the following guidelines when using the `-set log suspend` to take a snapshot of the data for offsite recovery.

***Suspend updates for the minimum time***

While the log suspension is in effect, all applications which perform updates, all DB2 utilities and many commands will freeze. This includes utilities running with LOG NO which will halt because they update the DSNDB01.SYSUTILX table space.

All locks and claims held by hanging updating threads will continue to be held. If the period of suspension is greater than the lock timeout interval, you will see timeouts and deadlocks. The longer you suspend update activity and the more work inflight, the greater the likelihood and number of timeouts and deadlocks.

In addition, if there is a prolonged suspension, you may see DB2 and IRLM diagnostic dumps. This is more likely in a data sharing environment, where non-suspended members cannot get a response from a suspended member.

In general, read-only processing, both static and dynamic, will continue. However, there are some circumstances which mean that a system update is required to satisfy a read request. One possible cause is during lock avoidance, when the possibly uncommitted (PUNC) bit is set, but the page (or row) lock is successfully acquired. DB2 would then attempt to reset the PUNC bit. Another example is auto-rebinds, which cause updates to the catalog. Please bear in mind that, although updates during read only processing are rare, when they do occur, the suspension may cause other locks to be held longer than normal, causing contention within the system.

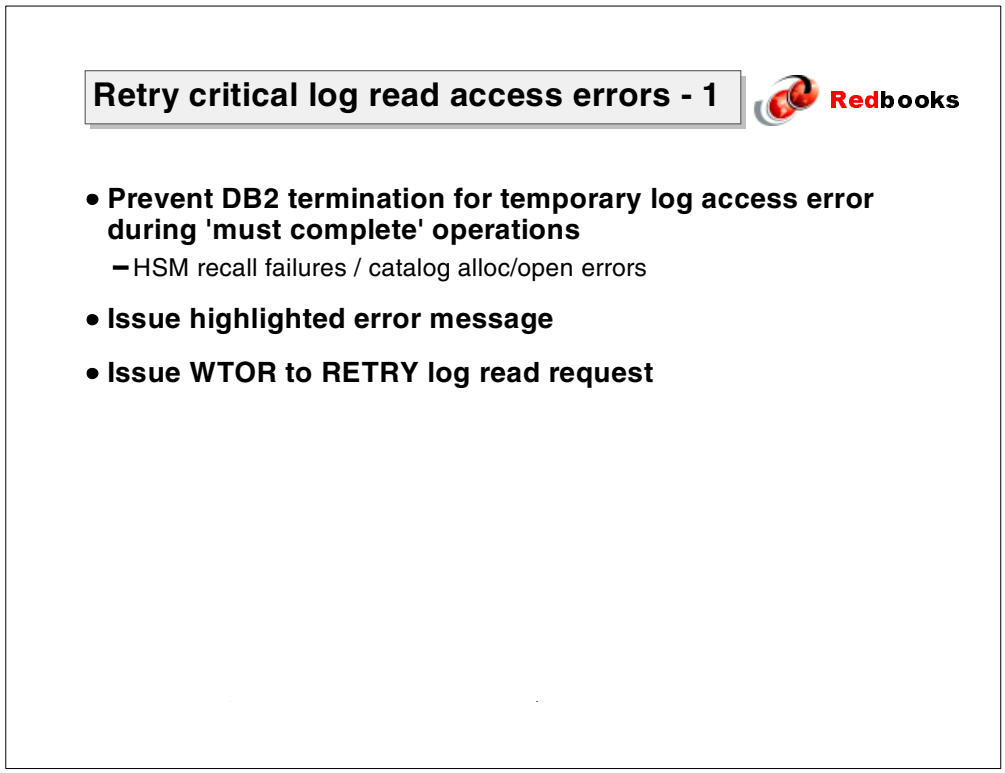
After issuing the `-set log resume` command, you will have to restart all abnormally terminated jobs and redo failed transactions. Since this impacts availability and requires intervention, we recommend suspending updates at a quiet time for the minimum period.

***Do not use during heavy update activity***
In addition to affecting more users and therefore increasing the likelihood of timeouts and contention, there are two adverse consequences of taking a snapshot at a time of heavy update activity:

***It will take you longer to get access to all your data***
Since pending writes from the bufferpools are not externalized, the table and index spaces on the offsite copy are not guaranteed to be in a consistent state. DB2 restart processing at the recovery site resolves all inconsistencies during the normal phases of forward and backward recovery. You can envisage the restart process following the restore at the remote site as being precisely equivalent to the restart processing at the local site after a system crash.

***RVA SnapShot will require additional capacity***
An RVA SnapShot copy takes no physical space within the RVA provided there is no difference between the original and the copy of the data. The more updates made before the copy is completed by backing up to tape or sent offsite with Peer-to-Peer Remote Copy, the more spare physical storage is needed onsite.

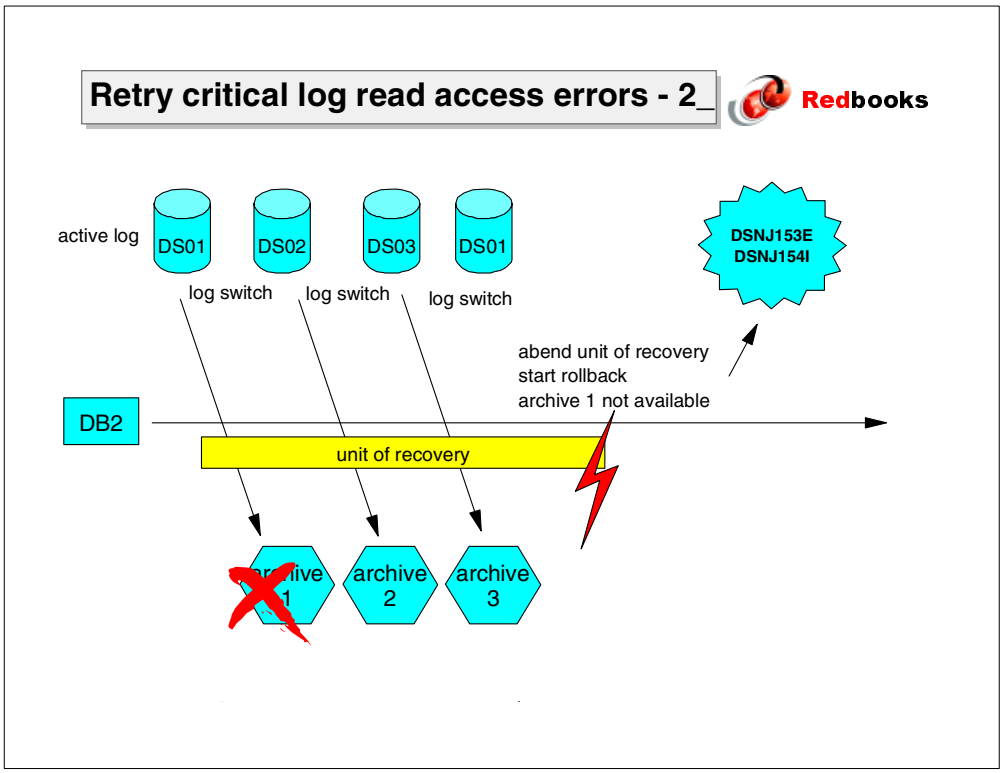
***Extra care required with 32 KB pages***
There is a small exposure when dealing with 32 KB pages if the write is suspended when writing the extents of a 32 KB page, or when crossing the volume boundaries. If the snapshot happens between the two I/Os, the page will be inconsistent. Issuing a SET LOG LOGLOAD(0) to force a checkpoint, say 10 minutes before suspending the updates, will further reduce the exposure because it will externalize all updates. However, it would be safer to run a DSN1COPY CHECK to exclude inconsistencies in the volumes backup, or recover the page from a DB2 data set based backup.




### 7.10.2 Retry critical log read access errors

While processing “must-complete” operations, a state during DB2 processing in which the entire operation must be completed to maintain data integrity, DB2 may encounter an error during an attempt to access the required log data sets. This is typically seen attempting to allocate or open archive log data sets during the rollback of a long running UR. These failures may be caused by a temporary problem with HSM recalls or tape subsystems, archives inadvertently uncataloged, or even archive mounts being cancelled by the operator. DB2 will attempt to retrieve the requested log records from all copies of the archive log, but will terminate the whole subsystem if failures occur accessing all copies during a ‘must-complete’ operation.

Suppose we have a UOR spanning several log data sets, some of them archived, and that, because of an abend, DB2 initiates a rollback up to an archived log, which is not available.

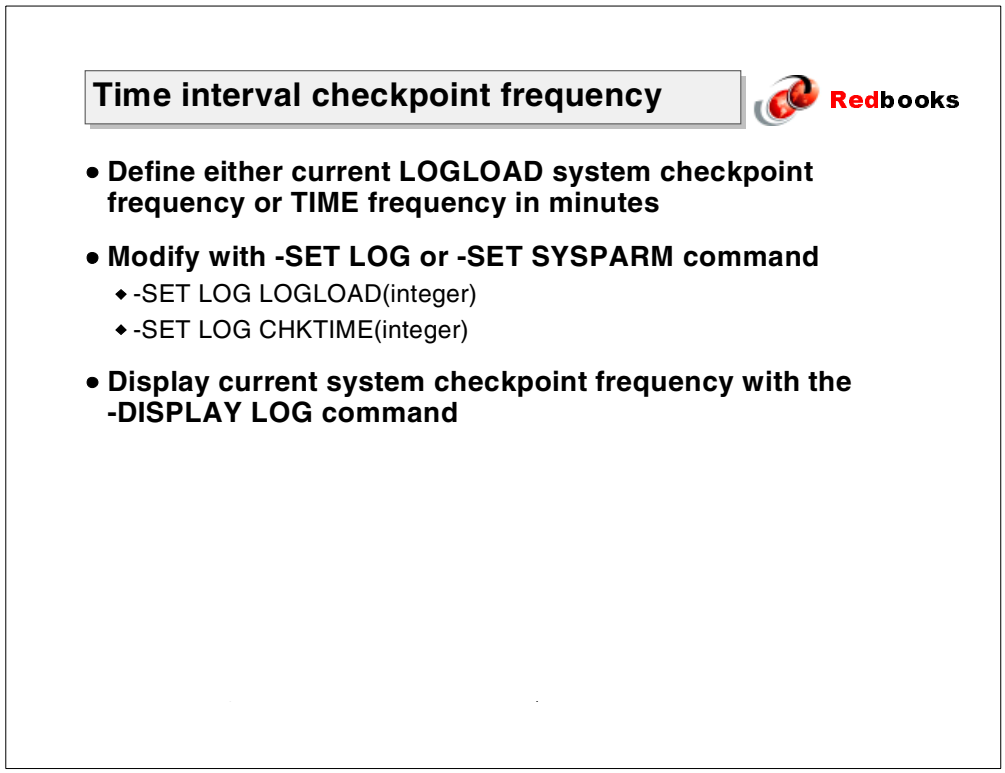



With DB2 V7, messages DSNJ153E and DSNJ154I (WTOR) will be issued to show and identify the critical log-read error failure. DB2 then will wait for the reply to message DSNJ154I before retrying the log-read access, or before abending.

```
 DSNJ104I     - DSNJR206 RECEIVED ERROR STATUS 00000004
                FROM DSNPCLOC FOR DSNAME=DSNC710.ARCHLOG1.A0000049
 DSNJ104I     - DSNJR206 RECEIVED ERROR STATUS 00000004
                FROM DSNPCLOC FOR DSNAME=DSNC710.ARCHLOG2.A0000049
*DSNJ153E     - DSNJR006 CRITICAL LOG READ ERROR
                CONNECTION-ID=TEST0001
                CORRELATIOND-ID=CTHDCORID001
                LUWID = V71A-SYEC1DB2.B343707629D=10
                REASON-CODE=00D10345
*26 DSNJ154I - DSNJHR126 REPLY Y TO RETRY LOG READ REQUEST,
                N TO ABEND
```

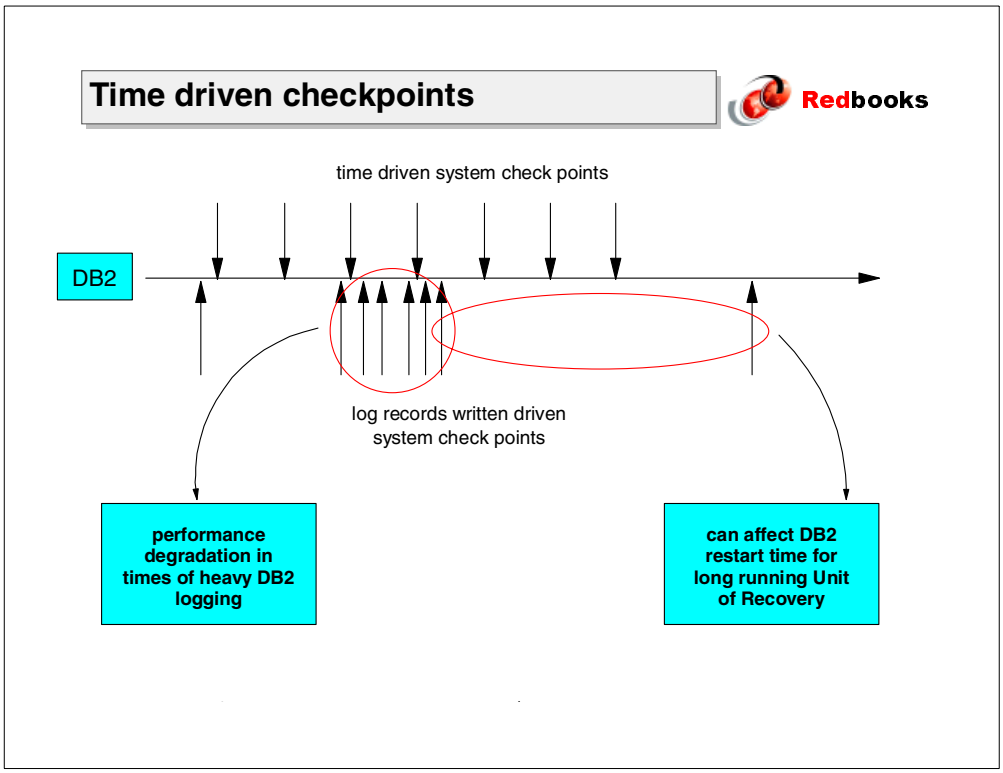
If you cannot correct the cause of the error by reviewing the description of the reason code and examining the system log for additional messages associated with the log-read error, you can quiesce the work on the DB2 subsystem before replying ‘N’ to the DSNJ154I message in preparation for DB2 termination.

With this enhancement, you will be aware of a critical log read error, and you may be able to fix it before the whole DB2 subsystem abends. All the other applications can go on doing their business as long as they do not depend on the “must-complete” operation which results in a better availability of your DB2 subsystem. If, for some reason, solving the log read error is not possible, the DB2 subsystem can be forced into terminating in a more controlled way by quiescing the work before replying ‘Y’ to the DSNJ154I message.




### 7.10.3 Time interval checkpoint frequency

With DB2 V7, the checkpoint frequency parameter is enhanced to allow to specify a range of minutes instead of a number of log records. Both options are available at install time and can be changed dynamically via commands. If you have widely variable logging rates, maximize system performance by specifying the checkpoint frequency in time to avoid the performance degradation of many system checkpoints being taken in a very short period of time due to very high logging rate. DB2 will start a new checkpoint at the interval you specify, either in minutes or number of log records.




### 7.10.4 Time driven checkpoints

Here, we compare records driven and time drive checkpoints. LOGLOAD and CHKTIME values can affect the amount of time needed to restart DB2 after abnormal termination. A large value for either option can result in lengthy restart times. A low value can result in excessive checkpointing.

For example, during prime shift, your DB2 shop might have a low logging rate, but require that DB2 restart quickly if it terminates abnormally. To meet this restart requirement, you can decrease the LOGLOAD value to force a higher checkpoint frequency. In addition, during off-shift hours the logging rate might increase as batch updates are processed, but the restart time for DB2 might not be as critical. In that case, you can increase the LOGLOAD value which lowers the checkpoint frequency.

You can also use the LOGLOAD option to initiate an immediate system checkpoint:

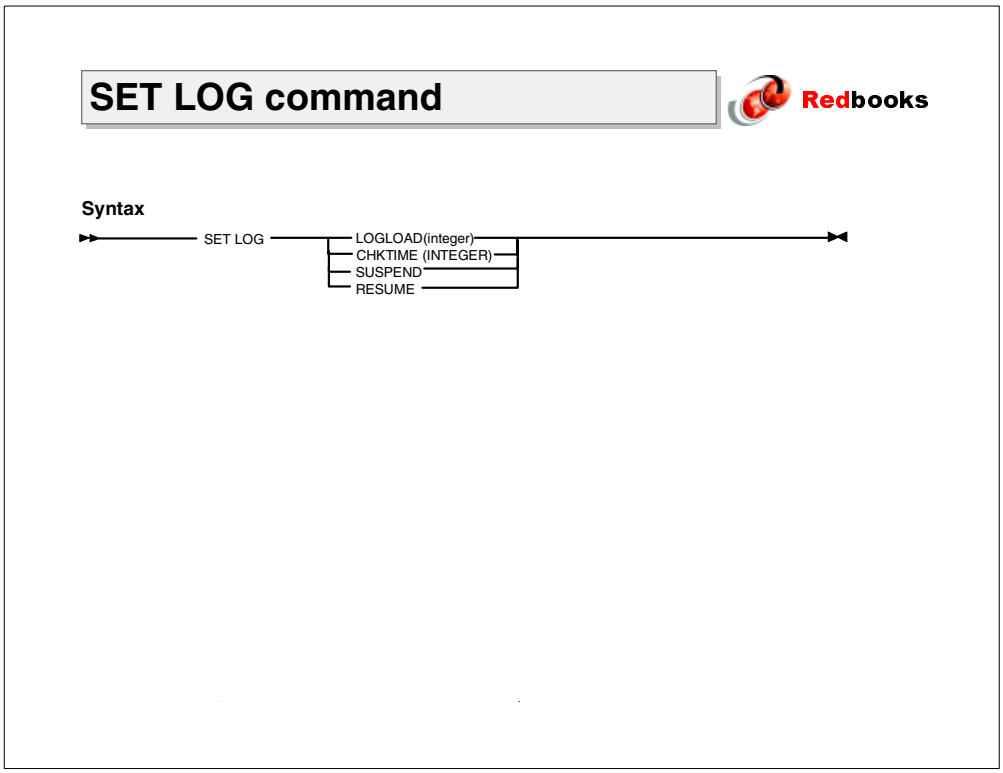
```
-SET LOGLOAD(0)
```

The LOGLOAD value that is altered by the SET LOG command persists only while DB2 is active. On restart, DB2 uses the LOGLOAD value in the DSNZPARM load module.

The current system checkpoint frequency can be displayed using the -DISPLAY LOG command:

```
=DB2A DIS LOG
DSNJ370I  =DB2A DSNJC00A LOG DISPLAY 313
CURRENT COPY1 LOG = DB2V710A.LOGCOPY1.DS02 IS 28% FULL
CURRENT COPY2 LOG = DB2V710A.LOGCOPY2.DS02 IS 28% FULL
H/W RBA = 000002B098A6, H/O RBA = 000000000000
FULL LOGS TO OFFLOAD = 2 OF 6, OFFLOAD TASK IS (BUSY,ALLC)
DSNJ371I  =DB2A DB2 RESTARTED 10:51:42 JUL 12, 2000 314
RESTART RBA 000002AA0000, CHECKPOINT FREQUENCY 45 MINUTES
LAST SYSTEM CHECKPOINT TAKEN 10:42:18 JUL 13, 2000
DSN9022I  =DB2A DSNJC001 '-DIS LOG' NORMAL COMPLETION
```

**Note:** The current system checkpoint frequency should always be investigated using the -DISPLAY LOG command instead of displaying the current subsystem parameter load module settings (DSNZPARM) using the DSN8ED7 sample program. Modifying the system checkpoint frequency using the -SET LOG command does not update the current subsystem parameter settings.




### 7.10.5 SET LOG command

The DB2 command SET LOG modifies the checkpoint frequency specified during installation. This command also overrides the value that was specified in a previous invocation of the SET LOG command. The changes that SET LOG makes are temporary; at restart, DB2 again uses the values that were set during installation. The new LOGLOAD value takes affect following the next system checkpoint.

***Environment***

This command can be issued from an MVS console, a DSN session under TSO, a DB2 panel (DB2 COMMANDS), and IMS or CICS master terminal, or a program using the instrumentation facility interface (IFI).

**Data sharing scope**: Member

***Authorization***

To execute this command, the privilege set of the process must include one of the following authorities:

- ARCHIVE privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands that are issued from an MVS console are not associated with any secondary authorization IDs.

***Options description***
Following are descriptions of the various options.

**LOGLOAD**(integer)

Specifies the number of log records that DB2 writes between the start of successive checkpoints. You can optionally specify a value of 0 to initiate a system checkpoint without modifying the current LOGLOAD value.

The value of *integer* can be 0, or within the range from 200 to 16000000.

**CHKTIME**(integer)

Specifies the number of minutes between the start of successive checkpoints. This option overrides log records specified by installation options or the LOGLOAD option that are base on checkpoint frequency.

The value of *integer* can be any integer value from 0 to 60. Specifying 0 starts a system checkpoint immediately without modifying the checkpoint frequency.

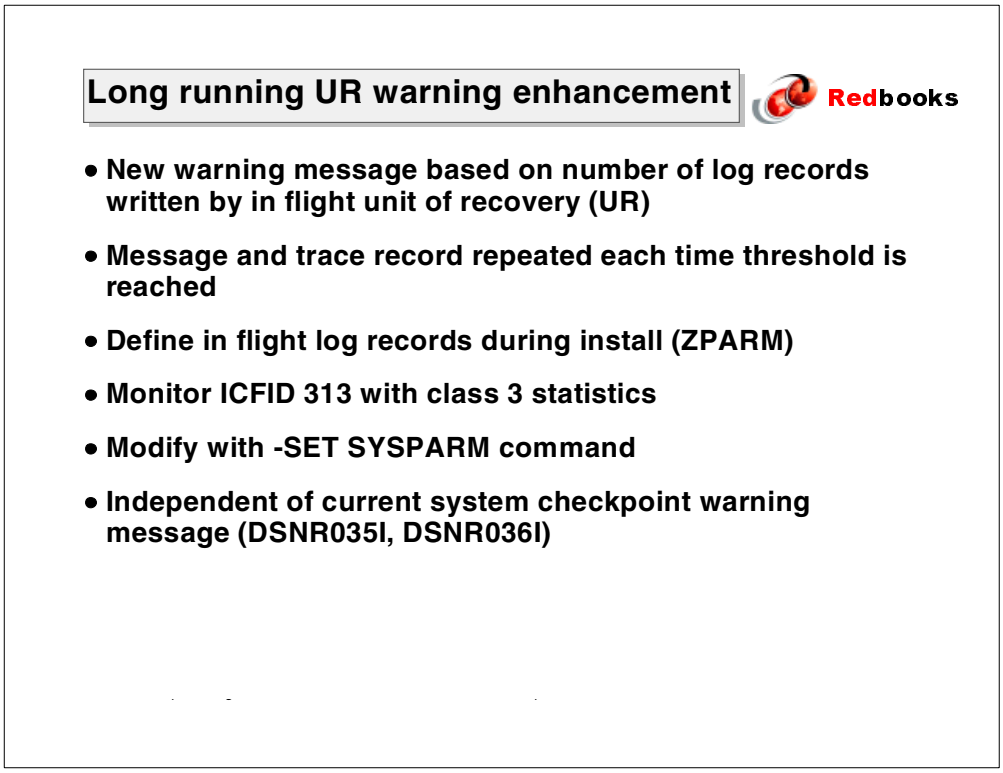
**SUSPEND**

This specifies to suspend logging and update activity for the current DB2 subsystem until SET LOG RESUME is issued. DB2 externalizes unwritten log buffers, takes a system checkpoint (in non-data sharing environments), updates the BSDS with the high-written RBA, then suspends the update activity. Message DSNJ372 is issued and remains on the console until update activity resumes. This option is not allowed when a system quiesce is active by either the ARCH VE LOG or STOP DB2 commands. Update activity remains suspended until SET LOG RESUME or STOP DB2 is issued.

**Recommendation:** Do not keep log activity suspended during periods of high activity or for long periods of time. Suspending update activity can cause timing-related events such as lock timeouts or DB2 and IRLM Diagnostic Dumps.

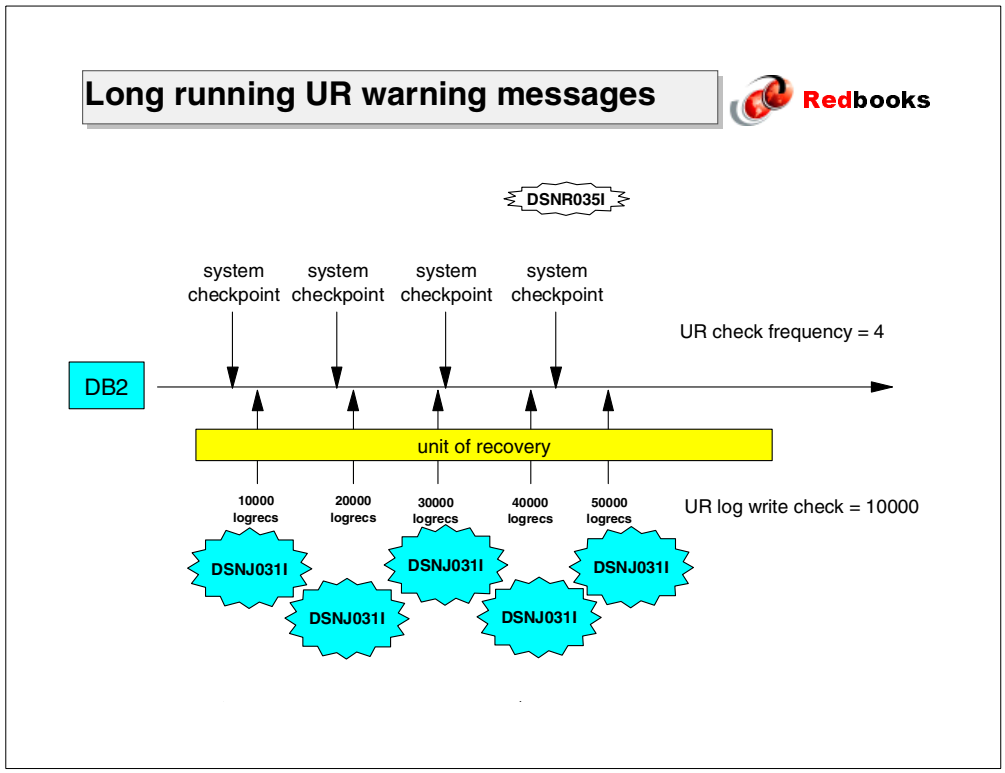
**RESUME**

Specifies to resume logging an update activity for the current DB2 subsystem and remove the message DSNJ372 from the console.




### 7.10.6 Long running UR warning enhancement

Prior to DB2 V7, the warning for long running unit of recovery (UR) was based on the number of checkpoint cycles to complete before DB2 issues a warning message for an uncommitted unit of recovery. But the number of checkpoints depends on several factors which may not include the long running job.

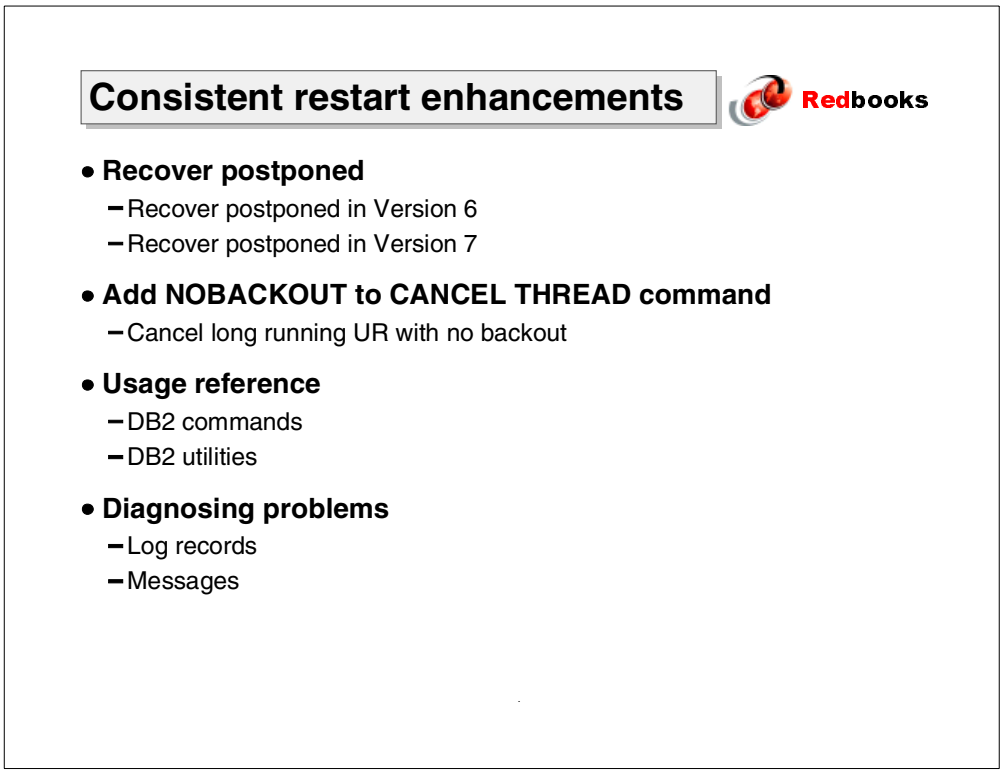



With DB2 V7, the warning mechanism is additionally based on the number of log records written by an uncommitted unit of recovery. The purpose of this enhancement is to provide notification of long running UR that may result in a lengthy DB2 restart or a lengthy recovery situation for critical tables.

```
DSNJ031I  =DB2A DSNJW001 WARNING - UNCOMMITTED UR 485
          HAS WRITTEN 11000 LOG RECORDS -
          CORRELATION NAME = PAOLOR5A
          CONNECTION ID  = BATCH
          LUWID = USIBMSC.SCPDB2A.B455F42B2279 = 22
          PLAN NAME = DSNTIA71
          AUTHID = PAOLOR5
          END USER ID = *
          TRANSACTION NAME = *
          WORKSTATION NAME = *
```

The warning message is repeated each additional time the threshold is reached. The value for written log records in the message is cumulative and indicates the number of log records written since the beginning of the UR. If statistics trace class 3 is active, an instrumentation facility component identifier (ICFID) 0313 is also written.

The UR log write check threshold is set in the DB2 parameter load module DSNZPARM (DSN6SYSP URLGWTH) at install time. The value may be modified using the `-SET SYSPARM` command.




## 7.11 Consistent restart enhancements

Restart pending (RESTP) and advisory restart pending (AREST) states of the objects are two new states introduced by DB2 V6. They are associated with the postponed abort transactions and indicate to the DB2 users that the data changes for these objects are not completely backed out. With DB2 V6, the RESTP and AREST states can only be removed when either the user or DB2 issues -RECOVER POSTPONED command. This command may take several minutes to several hours, depending on the content of the long running job. In certain cases it would be more efficient to recover those objects to a prior point in time (PIT) or running the Load utility with the replace option.

DB2 V7 consistent restart enhancements address two different objectives:

1. Removal of some of the restrictions imposed by the current consistent restart function:
   - Allow Recover utility to recover the objects associated with postponed abort UR.
   - Allow Load utility to replace the old data with the new data.
   - Allow the ability to cancel all postponed abort URs instead of recovering them.
2. Addition of no backout of data and log feature to DB2's -CANCEL THREAD command.

By providing support for these two objectives, DB2 user will be able to better control the availability of the user objects associated with the failing or cancelled transaction without restarting DB2.

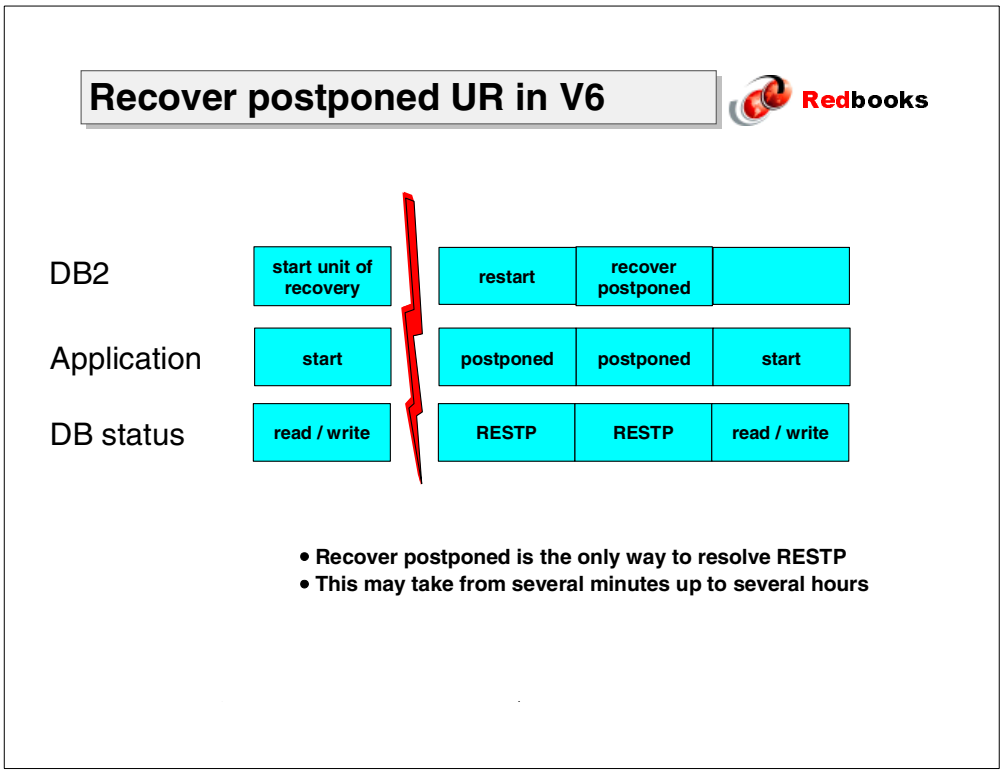
DB2 V7 introduces the following changes to support the new consistent restart enhancements:

- Recover postponed cancel command to cancel the recovery of postponed abort UR:
  - Cancel keyword is added to the existing recover postponed command
  - All postponed abort URs are cancelled
  - Message DSNV435I indicates that the cancellation of postponed URs has been scheduled.
  - Message DSNR042I indicates that the rollback for postponed abort URs has been cancelled.
- New database exception state

  Pageset/partition is marked refresh pending (REFP) in the database exception table (DBET) for each object whose backout is not complete when:
  - Postponed abort cancel command was issued or
  - Cancel thread nobackout command was issued.
- CANCEL THREAD NOBACKOUT command to cancel the long running thread without backing out data changes:
  - NOBACKOUT keyword is added to existing cancel thread and cancel ddf thread command.
  - Message DSNR042I indicates that the rollback for the indicated thread has been cancelled.

    A -CANCEL THREAD NOBACKOUT will also be accepted if the thread was previous cancelled without the nobackout keyword.
- Modified log records type LRHEABRT and LRHEUNDO:

  Both log records includes a new UR state called “CANCELLED/ABORT”. Both of these states are identified in DSN1LOGP summary report.




## 7.11.1 Recover postponed UR

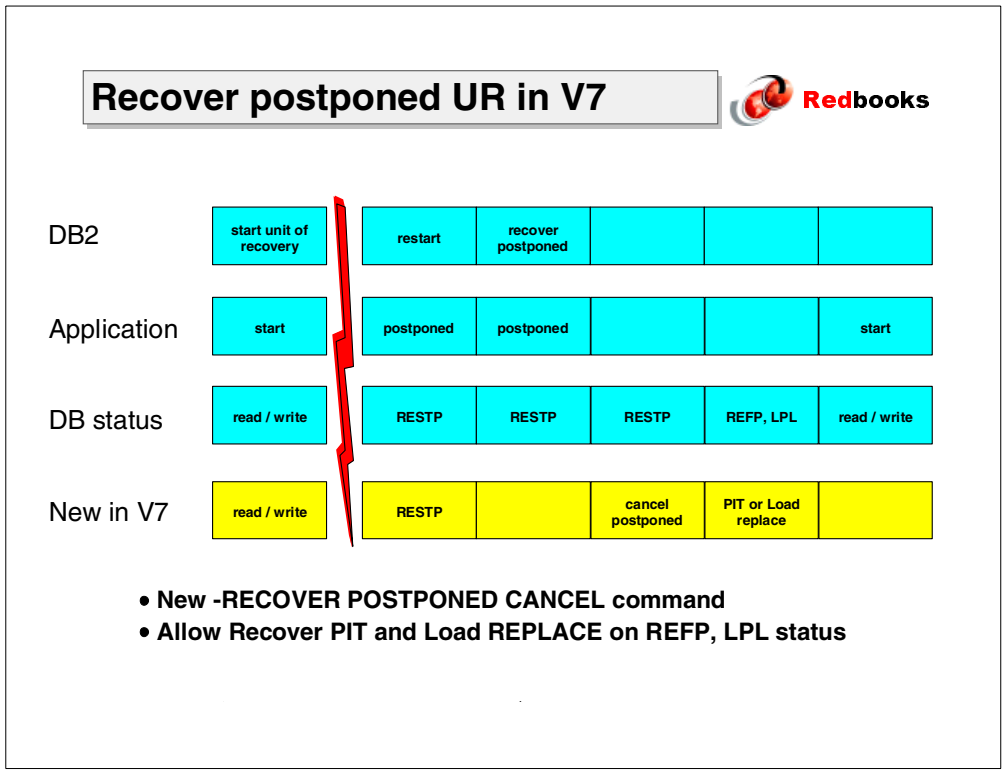
In this section we highlight the improvements in the consistent restart functions between DB2 V6 and V7.

### 7.11.1.1 Recover postponed UR in V6

With DB2 V6, the possibility to delay the backward log processing to process all inflight and inabort units of recovery (UR) can be postponed during restart by subsystem parameters (LBACKOUT YES/NO/AUTO and backout duration in DSNZPARM). Inflight and inabort URs that are not completely backed out during restart are converted to postponed abort status. Page sets or partitions with postponed backout work are put into restart pending (RESTP) or advisory restart pending (AREST) when using data sharing.

The backout processing for these URs left incomplete is delayed to make DB2 restart faster and allow access to the other objects. The only way to complete the backout processing for URs left incomplete during earlier restart (POSTPONED ABORT units of recovery) is the `-RECOVER POSTPONED` command, which can be implicit or explicit. Depending on the content of a long running application this command may take several minutes up to several hours to be executed.

DB2 V6 standard manuals and the redbook *DB2 UDB for OS/390 Version 6 Performance Topics*, SG24-5351 describe this function in more detail.




### 7.11.1.2 Recover postponed UR in V7

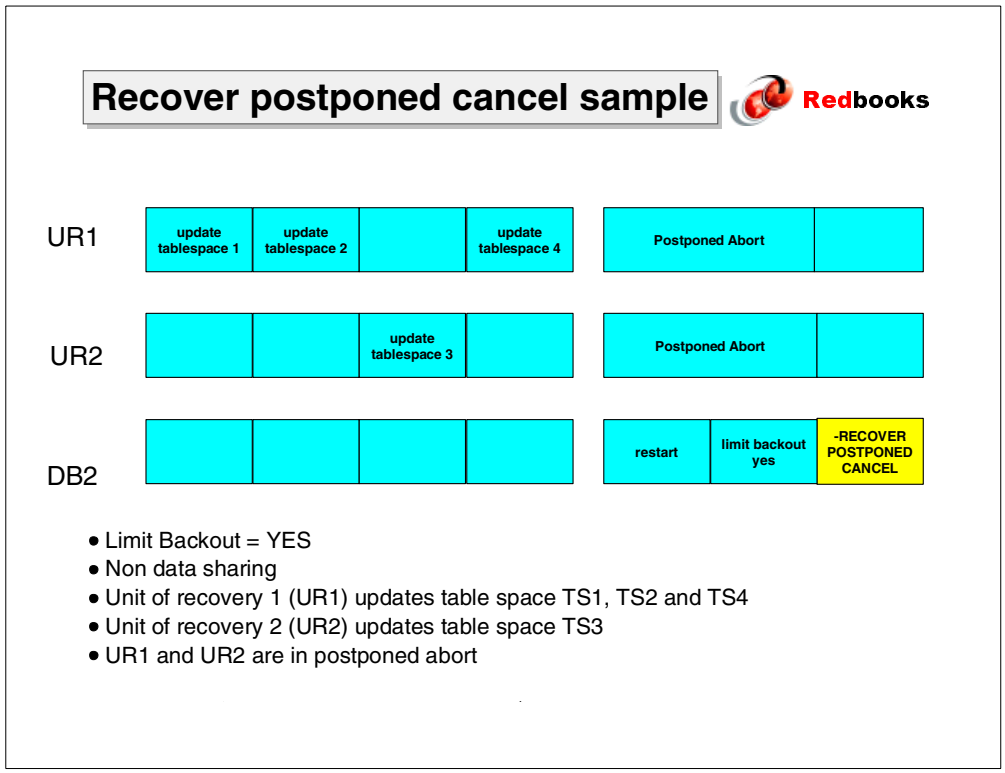
In certain cases, it could be more efficient to recover restart pending objects to a prior point in time using the Recover or the Load replace utilities to remove the RESTP or AREST state.

In DB2 V7, the `-RECOVER POSTPONED` command allows the ability to cancel all postponed abort URs instead of recovering them with the new optional keyword CANCEL. Also, Recover and Load utilities are allowed to operate on objects associated with postponed abort UR.

In order to support removing RESTP and AREST state, the user is required to issue first the `-RECOVER POSTPONED CANCEL` command. DB2 will accept this request either while postponed abort URs are actively being recovered or waiting to be recovered.

Please note that the `-RECOVER POSTPONED CANCEL` command will cancel recovery of all the postponed abort URs that exist in the DB2 that issued this command. At the end of the successful completion of this command, all objects associated with the postponed abort URs are marked in REFP and LPL states in the DataBase Exception Table (DBET). The DBET entry for the REFP object will also have a one-byte release dependency value to determine whether or not the object can be accessed in the current release. Recover (with TOCOPY, TORBA, or TOLOGPOINT) or Load (REPLACE) utilities can be run to recover these objects. No other utilities are allowed on objects marked refresh pending (REFP).

```
=DB2A RECOVER POSTPONED CANCEL
DSNB250E  =DB2A DSNICLPA A PAGE RANGE WAS ADDED TO 465
THE LOGICAL PAGE LIST
           DATABASE NAME=DRHATEST
           SPACE NAME=SRHATEST
           DATA SET NUMBER=1
           PAGE RANGE X'00000000' TO X'FFFFFFFF'
           START LRSN=X'00000679F335'
           END LRSN=X'FFFFFFFFFFFF'
           START RBA=X'00000679F335'
DSNI033I  =DB2A DSNICLPA PAGESET DRHATEST.SRHATEST 466
PART (n/a)
        IS MARKED REFP AND LPL ON BEHALF OF UR 00000679F2A5.
RECOVERY TO
        RBA 00000679F335 IS REQUIRED.
DSNV435I  =DB2A CANCELLATION OF POSTPONED ABORT URS HAS BEEN SCHEDULED
DSN9022I  =DB2A DSNVRP 'RECOVER POSTPONED' NORMAL COMPLETION
DSNR042I  =DB2A DSNRPBUP WARNING - UR ROLLBACK HAS 469
BEEN CANCELLED AND IS INCOMPLETE FOR
          CORRELATION NAME = PAOLOR5A
          CONNECTION ID  = BATCH
          AUTHID = PAOLOR5
          PLAN NAME = DSNTIA71
          URID = 00000679F2A5
```




### 7.11.1.3 Recover postponed cancel sample

The `-DISPLAY DATABASE(*) SPACENAM(*) RESTRICT` shows table space TS1, TS2, TS3 and TS4 are in RESTP state.

The `-RECOVER POSTPONED CANCEL` command cancel the rollback for postponed abort unit of recovery UR1 and UR2.

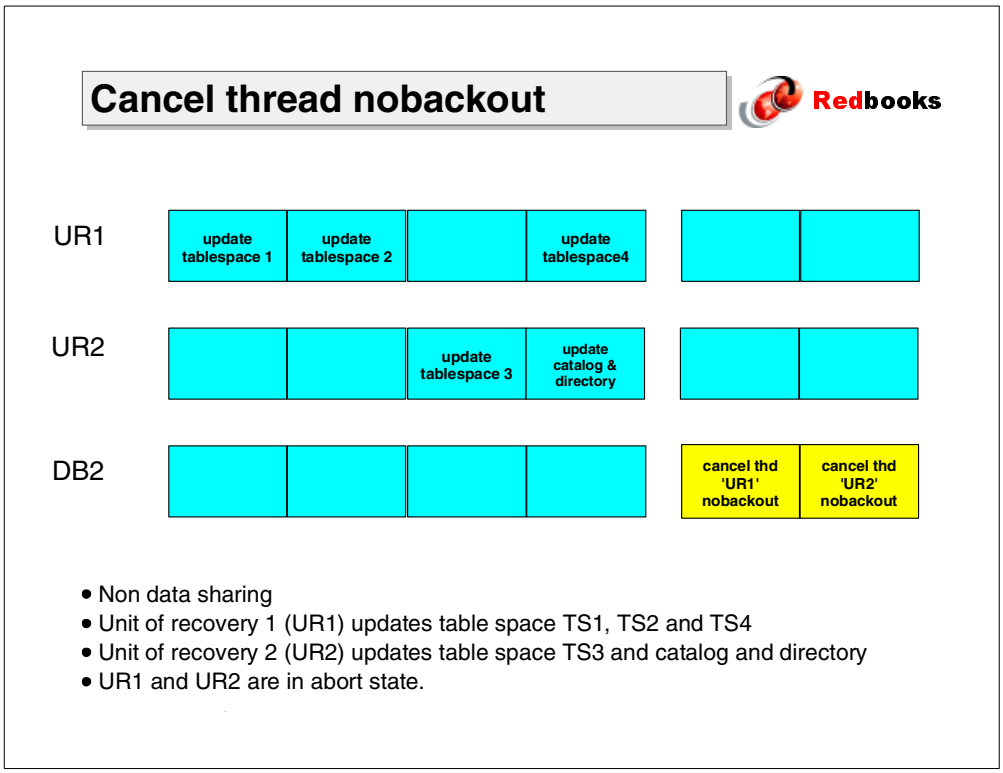
Message DSNI033I is issued for each tablespace to indicate that it is marked REFP, LPL.

Message DSNV435I is issued to indicate that the cancellation of recover postponed abort for UR1 and UR2 has been scheduled.

Message DSNR042I is issued for each unit of recovery (UR1 and UR2) to indicate that the cancellation of UR rollback processing has been cancelled.

The `-DISPLAY DATABASE(*) SPACENAM(*) RESTRICT` shows table space TS1, TS2, TS3 and TS4 are in REFP, LPL state.

Recover table space TS1, TS2, TS3 and TS4 using Recover PIT or Load REPLACE utility.




### 7.11.2 Cancel thread no backout

The second enhancement allows you to cancel a long-running thread without backing out data changes. A new keyword NOBACKOUT is added to the CANCEL THREAD command. With addition of this new keyword, DB2 may accept multiple requests for the CANCEL THREAD command.

For example, an application or an operator can issue the CANCEL THREAD command (this request can be made only once). Then later, when it determines that this cancel is running for too long, it can decide to issue the CANCEL THREAD NOBACKOUT command (multiple requests of this command are allowed) to stop reading log records and avoid writing and applying compensation log records. At the successful completion of this command, all the objects associated with the thread will be marked REFP and LPL. The CANCEL THREAD NOBACKOUT command can fail for the following reasons:

- If the catalog and/or directory changes are made by the thread and not completely backed out when the request for NOBACKOUT is made
- If the thread is part of the global transaction.

The `-DISPLAY DATABASE(*) SPACENAM(*) RESTRICT` shows table space TS1, TS2, TS3 and TS4 are in read/write (RW) state.

The `-CANCEL THREAD(token ur1) NOBACKOUT` and `-CANCEL THREAD(token ur2) NOBACKOUT` commands cancel the rollback for those units of recovery.

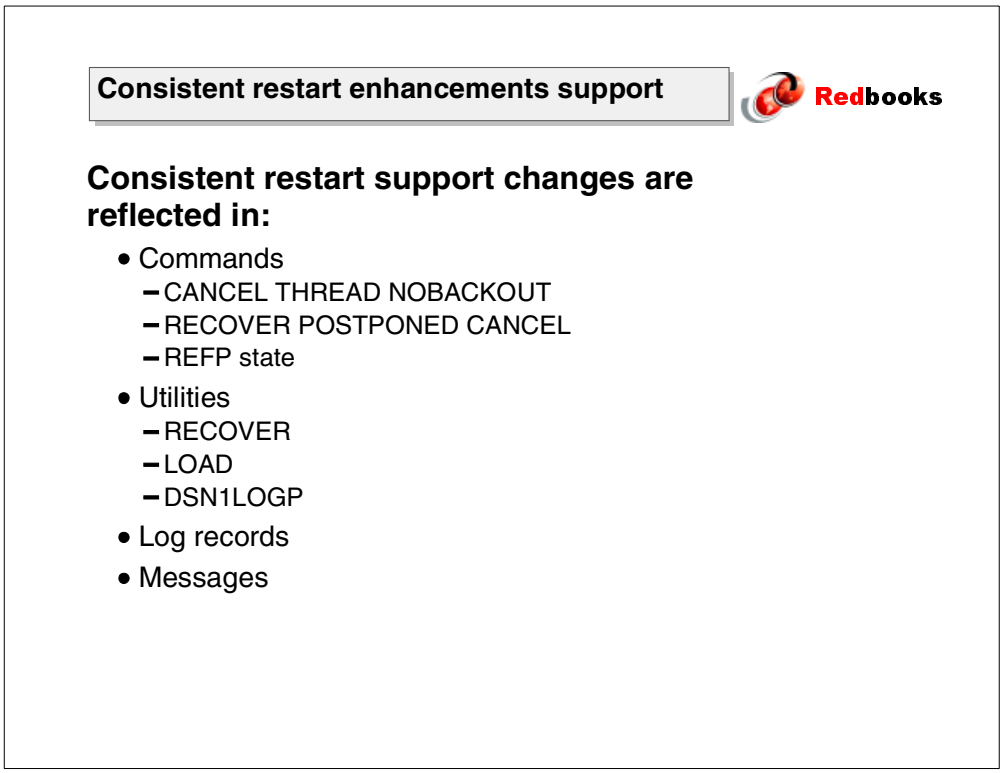
Message DSNI033I is issued for each updated table space (TS1, TS2, TS4) to indicate that each object is marked REFP, LPL.

Message DSNI032I with reason code 00C900CC for UR2 is issued to indicate that DB2 does not accept NOBACKOUT request during the rollback of catalog changes.

The `-DISPLAY DATABASE(*) SPACENAM(*) RESTRICT` shows table space TS1, TS2 and TS4 are in REFP, LPL state.

Recover table space TS1, TS2 and TS4 using Recover PIT or Load REPLACE utility.

Note: CANCEL NOBACKOUT can be issued after a normal CANCEL.




## 7.11.3 Consistent restart enhancements support

### 7.11.3.1 DB2 commands

The following commands are adapted to support the consistent restart enhancements.

***CANCEL THREAD***

To the -CANCEL THREAD command the optional keyword NOBACKOUT has been added. If NOBACKOUT is specified, then multiple cancel requests are permitted per thread.

***Warning***: Cancelling the thread with nobackout leaves the objects in an inconsistent state. Do not issue this command unless you already have a plan to resolve the data inconsistency that results from using this option.

When you specify this option, DB2 will not attempt to back out the data during transaction rollback processing. Multiple -CANCEL THREAD NOBACKOUT requests are allowed. However, if the thread is active and the first request is accepted, any subsequent requests will be ignored. You might choose to issue a subsequent request if the first request failed with the reason indicated in message DSNI032I. Each object modified by this thread that was not completely recovered (backed out) is marked Refresh pending (REFP) and LPL in the DBET. Resolve the REFP status of the object by running the recover utility to recover the object to a prior point in time or by running Load replace on the object.

***RECOVER POSTPONED***

The optional keyword CANCEL has been added to the -RECOVER POSTPONED command.

***Warning***: Cancelling the postponed abort units of recovery leaves the objects in an inconsistent state. Do not issue this command unless you already have a plan to resolve the data inconsistency that results from using this option.

When you specify this option, DB2 stops processing all postponed abort units of recovery immediately. Each object modified by the postponed units of recovery that was not completely recovered (backed out) will be marked Refresh pending (REFP) and LPL in the DBET. Resolve the REFP status of the object by running the Recover utility to recover the object to a prior point in time or by running Load replace on the object.

***START DATABASE ACCESS(FORCE)***
The -START DATABASE ACCESS(FORCE) command is allowed to remove the refresh pending (REFP) state, but the data is not consistent.

The consistent restart enhancement will also allow users to reset the object's RESTP, AREST and REFP states by using -START DATABASE ACCESS(FORCE) command. In order for -START DATABASE ACCESS(FORCE) command to work on these states, user should make sure that the object with one of the above listed states is not associated with the postponed abort or indoubt unit of recovery. The user can issue DISPLAY THREAD TYPE(POSTPONED) from each DB2 system to determine whether or not any postponed abort URs exist on the system. Similarly, user can issue DISPLAY THREAD TYPE(INDOUBT) from each DB2 system to determine whether or not any indoubt URs exist on the system.

When the user or the application issues a -START DB ACCESS(FORCE) command a new diagnostic log record will be written. This new log record is non-UR related and will not be processed during DB2 restart or during any data recovery process.It contains the DBID/OBID of object being forced plus the database name and page set name. It s a subtype name of "Start database force" log record.

***DISPLAY DATABASE***
Refresh pending will be displayed as REFP. It is a restrictive status, so any object in refresh pending will be displayed in response to a DISPLAY DB RESTRICT command.

REFP will be added to parameters that can be specified in the RESTRICT keyword (as for DISPLAY DB(*) SPACE(*) RESTRICT(REFP))

#### 7.11.3.2 DB2 Utilities
The following utilities are adapted to support the consistent restart enhancement.

***Recover***
The RECOVER utility will allow point in time (TOCOPY, TORBA and TOLOGPOINT) recovery for the objects marked *refresh pending* (REFP). The objects include all user defined table spaces and indexes but do not include catalog and directory table space and indexes.

***Load***
Only REPLACE option of the Load utility will be allowed on the user defined table spaces that are in REFP.

***DSN1LOGP***
The DISP (UR disposition) field in the summary report will be set to CANCELLED for those units of recovery whose recovery was cancelled. CANCELLED is a new unit of recovery status defined. A unit of recovery is placed in the cancelled state when the recovery of the unit of recovery is interrupted either by issuing the RECOVER POSTPONED CANCEL command or by issuing the CANCEL THREAD NOBACKOUT command.

***Note***: Please note that whenever DB2 decides to mark an object REFP, it also puts the same object in LPL. In other words, REFP status cannot be ON by itself. REFP status can NOT exist without LPL, but LPL state exists without REFP. At the successful completion of the RECOVER and LOAD REPLACE job, both (REFP and LPL) states will be reset.

#### 7.11.3.3 Log records
A new diagnostic log record is added that will log successful START DATABASE ACCESS(FORCE) commands. The DBID/OBID/PART will be logged plus the database and pageset name. The log record will be a diagnostic type log record with subtype name of "Start database force". This new subtype is defined as LGOPSTFR, whose value is 130 and is listed under the description section for macro DSNDLGOP, in the DSNDQJ00 macro.

#### 7.11.3.4 Messages
Several new messages have been introduced to support these new restart functions. They are reported here only as an example of the necessary changes in operations. You must verify the correct messages and related actions on your subsystem based on your level of maintenance, especially if you plan to automate operations.

- DSNV439I

**DSNV439I** *sect-name* **NOBACKOUT OPTION INVALID FOR THREAD** *token*

**Explanation:** This message is issued in response to the CANCEL THREAD command with the NOBACKOUT option. The NOBACKOUT option will not be honored because the cancelled thread is part of a global transaction.

- DSNI032I

**DSNI032I** *csect-name* **CANCEL THREAD NOBACKOUT COMMAND FAILED FOR THE THREAD =** *token* **REASON =** *reason*

**Explanation:** DB2 displays this message when it cannot grant a request to cancel a thread without backing out data changes.

The reason code shows why the request was rejected:

*token* Identifies a thread whose processing you requested to cancel. The token is a decimal number of 1 to 15 digits.

*reason* Indicates the reason why the command failed.

- DSNI033I

**DSNI033I** *csect-name* **PAGE SET** *dbnam.psnam* **PART** *part* **IS MARKED REFP and** *status* **ON BEHALF OF UR** *urid***. RECOVERY TO** *logpoint* **IS REQUIRED.**

**Explanation:** This message indicates that backout for unit of recovery *urid* has been cancelled and the specified page set or partition is marked Refresh pending (REFP) and LPL. No further backout processing will be attempted on the specified page set or partition.

For non-partitioned page sets, the partition given in the message is the string "n/a".

- DSNR042I

**DSNR042I** *csect-name* **WARNING - UR ROLLBACK HAS BEEN CANCELLED AND IS INCOMPLETE FOR**

**CORRELATION NAME =** *corrid*,

**CONNECTION ID =** *connid*,

**AUTHID =** *authid*,

**PLAN NAME =** *plan-name*,

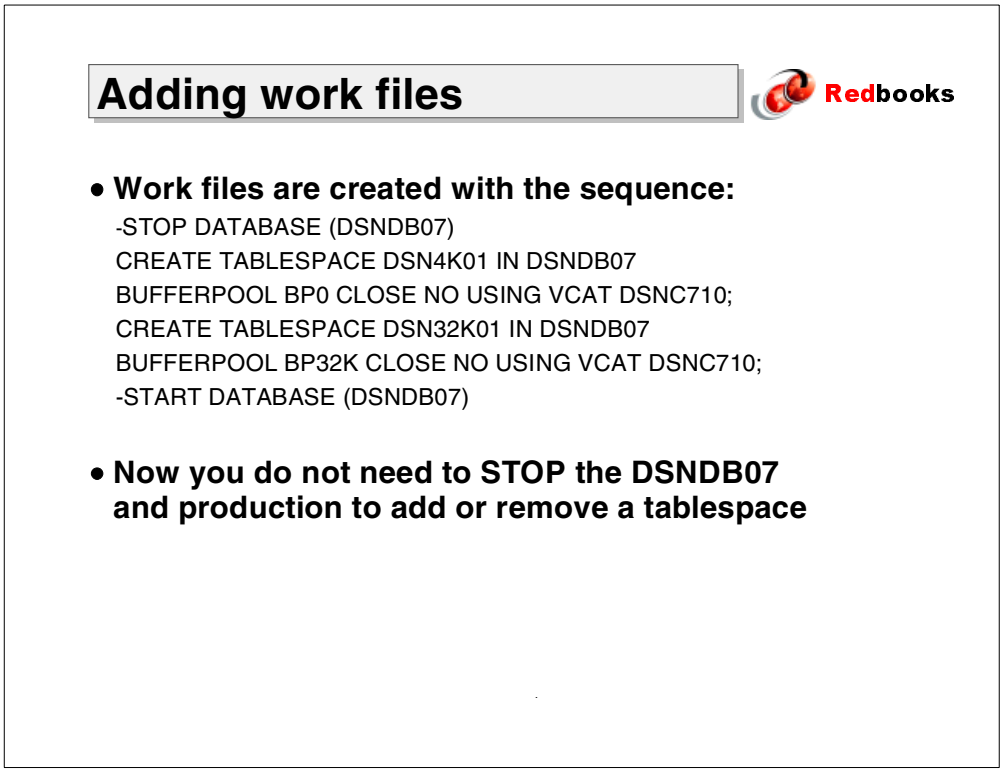
**URID =** *urid*

**Explanation:** This message is issued when rollback for the indicated thread has been cancelled by either the CANCEL THREAD NOBACKOUT command or the RECOVER POSTPONED CANCEL command.

- DSNU215I

**DSNU215I** csect-name **REFRESH PENDING ON** obj-type database.objectname **PROHIBITS PROCESSING**

**Explanation:** An attempt was made to execute a utility against a table space or index that is the refresh pending status.




### 7.11.4 Adding work files

DB2 V7 provide a less disruptive addition of workfile table space. It allows you to CREATE and DROP workfile table space without having to STOP the workfile database

All customers will benefit from this enhancement, particularly large sites where significant space needs to be allocated for large workloads, and/or large query sorting, and/or 24x7 applications.You will also be able to better manage your workfile space by changing the workfile allocations more frequently. This enhancement reduces performance problems and improves availability in a 24x7 environment because changing workfile space allocation is much less disruptive.

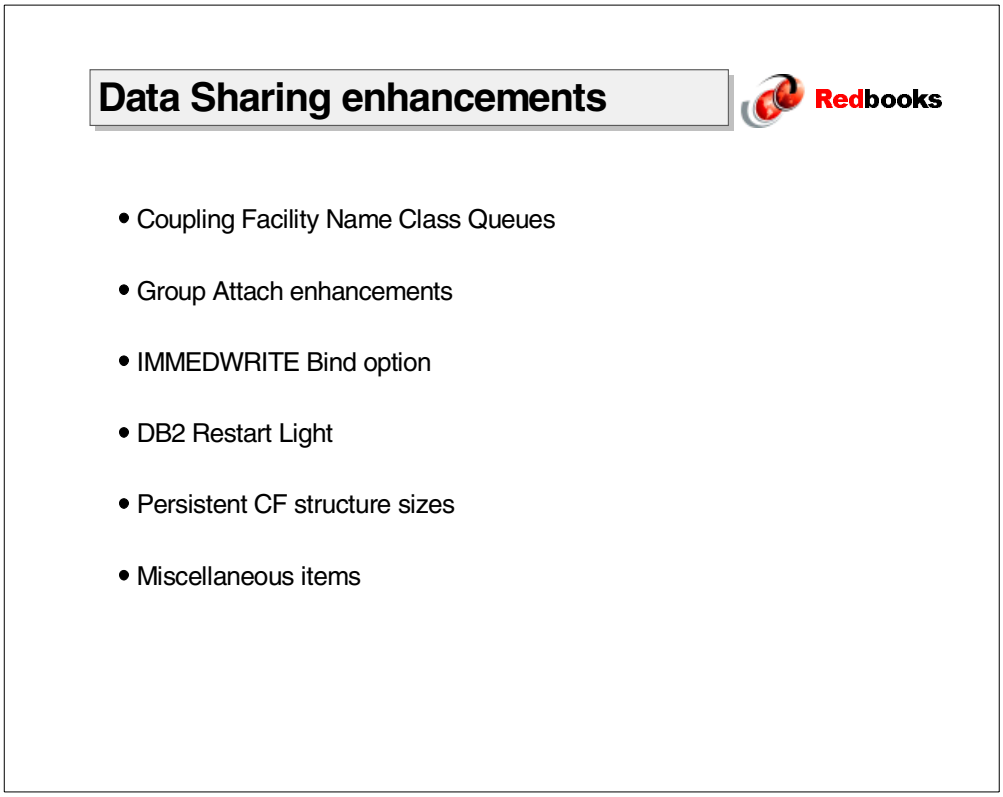
In a non-data sharing environment, an U lock is taken on the workfile DBD. Therefore while you are creating or dropping a workfile table space, other DB2 agents are able to continue to use other table spaces in the workfile database.

However in a data sharing environment, an U lock is taken on the workfile database DBD only if the DB2 member executing the DDL is the ‘owning’ DB2 member for that workfile database. Otherwise an X lock is taken on the DBD. Therefore DB2 will allow other DB2 agents concurrent use of other workfile table spaces in the database only if you execute the DDL on the ‘owning’ DB2 member. Otherwise DB2 will not allow concurrent access to the other workfile table spaces in the database being modified.

# Part 6. DB2 Data Sharing

© Copyright IBM Corp. 2001

# Chapter 8. DB2 Data Sharing




DB2 V7 introduces a number of enhancements to improve the availability, performance, and usability of DB2 Data Sharing.

***Coupling Facility Name Class Queues***
DB2 V7 exploits the OS/390 and z/OS support for the Coupling Facility Name Class Queues. This enhancement reduces the performance impact of purging group buffer pool (GBP) entries for GBP-dependent page sets.

***Group Attach enhancements***
A number of enhancements are made to Group Attach processing in DB2 V7:

- An application can now connect to a specific DB2 member of a Data Sharing group, where there are two or more members of the Data Sharing group active on the same OS/390 image and one member shares the Group Attach name. This is important for some applications who need to connect to specific members, rather than connecting to any active member on the OS/390 image. Monitoring type of applications are one example of this need.
- You can now connect to a DB2 Data Sharing group, by using the Group Attach name, and be notified when any member of the Data Sharing group becomes active on that OS/390 image.
- DB2 V7 allows you to specify the Group Attach name for your DL/I Batch applications.

© Copyright IBM Corp. 2001

***IMMEDWRITE bind option***

The IMMEDWRITE bind/rebind options, introduced by APAR in DB2 V5 and V6, are now fully implemented in DB2 V7. New columns are added to the DB2 catalog tables and DB2I is updated to include the IMMEDWRITE bind option. A new DSNZPARM parameter is also included in the Installation panels to allow you to specify a default IMMEDWRITE option for the DB2 subsystem.

***DB2 Restart Light***

DB2 V7 provides a new “Restart Light” option. This helps you to quickly restart a failed DB2 member on another OS/390 image with minimal impact on that OS/390 workload, in order to release any retained locks. During restart, the DB2 member will minimize its storage requirements by not allocating a number of structures (for example, no EDM pool, RID pool, or hiperpools). Once the DB2 member is successfully restarted in “light” mode and freed any retained locks that can be released, DB2 will terminate normally. DB2 will not accept any new work during a Restart Light.

***Persistent coupling facility structure sizes***

DB2 V7 now maintains the current size of DB2 structures persistently across DB2 executions and structure allocations. The saved size is used when:

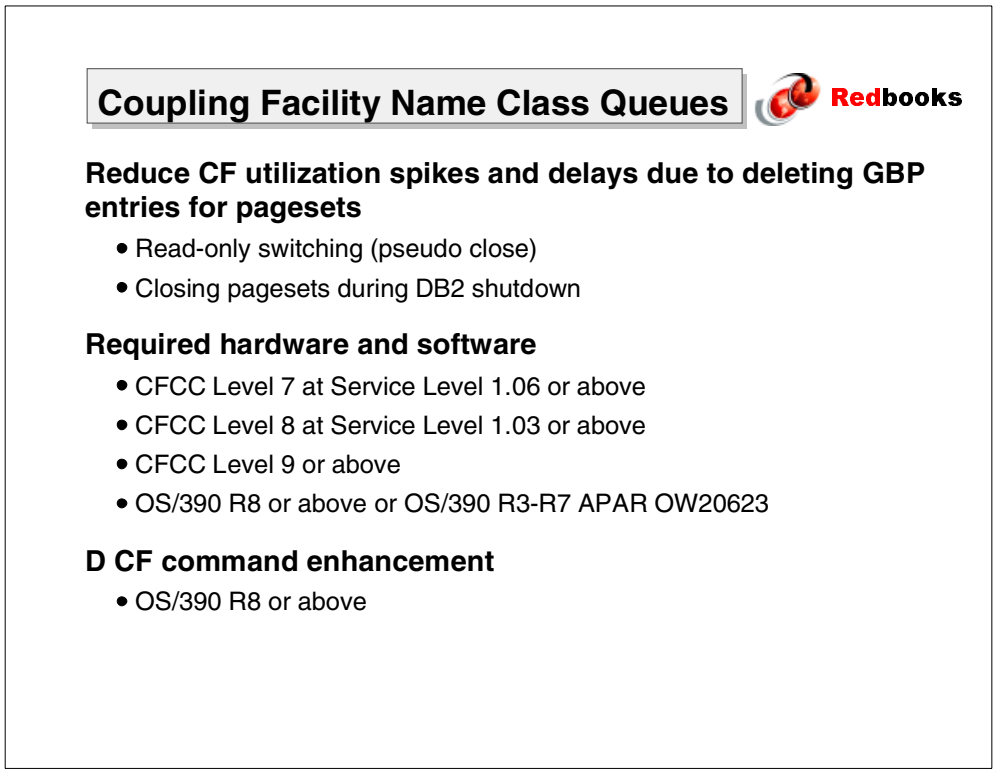
- Allocating a new coupling facility structure instance in response to a structure rebuild
- When allocating a secondary structure to support duplexing
- When allocating a group buffer pool after a SET XCF,ALTER command is used

***Miscellaneous items***

Finally, a number of usability enhancements are also introduced in DB2 V7:

- During normal shutdown processing, DB2 now notifies you of any incomplete units of recovery that will hold retained locks after the DB2 member has shut down. This message is in addition to the existing DSNR036I message that notifies you at each DB2 checkpoint of any unresolved indoubt URs.
- DB2 V7 reduces the MVS to DB2 communication that occurs during a coupling facility/structure failure. This enhancement will decrease recovery times in such failure situations.

.




## 8.1 Coupling Facility Name Class Queues

The Coupling Facility Control Code (CFCC) Level 7, introduced a new function called Named Class Queues. DB2 V7 requests to use Named Class Queues at group buffer pool (GBP) connect.

In a DB2 Data Sharing environment, DB2 can utilize the Named Class Queues feature to reduce the problem of coupling facility utilization spikes and delays caused by de-registering and deleting entries from the GBP, which occurs:

- When the last updater pseudo-closes the data set (read-only switching)
- When DB2 is shut down and that member is the last updater of the page set/partition.

Named Class Queues allows the CFCC to organize the GBP directory elements into “queues” based on DBID, PSID and partition number. Organization in this manner allows for locating and purging these elements more efficient as done during pseudo-close and DB2 shutdown. (DB2 no longer has to scan the whole directory structure looking for entries to purge.)

To avoid potential data corruptions in a Data Sharing environment when a GBP is allocated in a coupling facility at CF Level 7 or above, DB2 support for Named Class Queues requires a number of fixes to be applied to the Coupling Facility Control Code (CFCC). Due to an error in the CF code, a purge request for one page set may actually purge modified data belonging to another page set, resulting in broken data if the CFCC fix is not applied.

For CFCC Level 7, the fix is included in Service Level 1.06. For CFCC Level 8, the fix is included in Service Level 1.03. The fix is included in CFCC Level 9 base code.

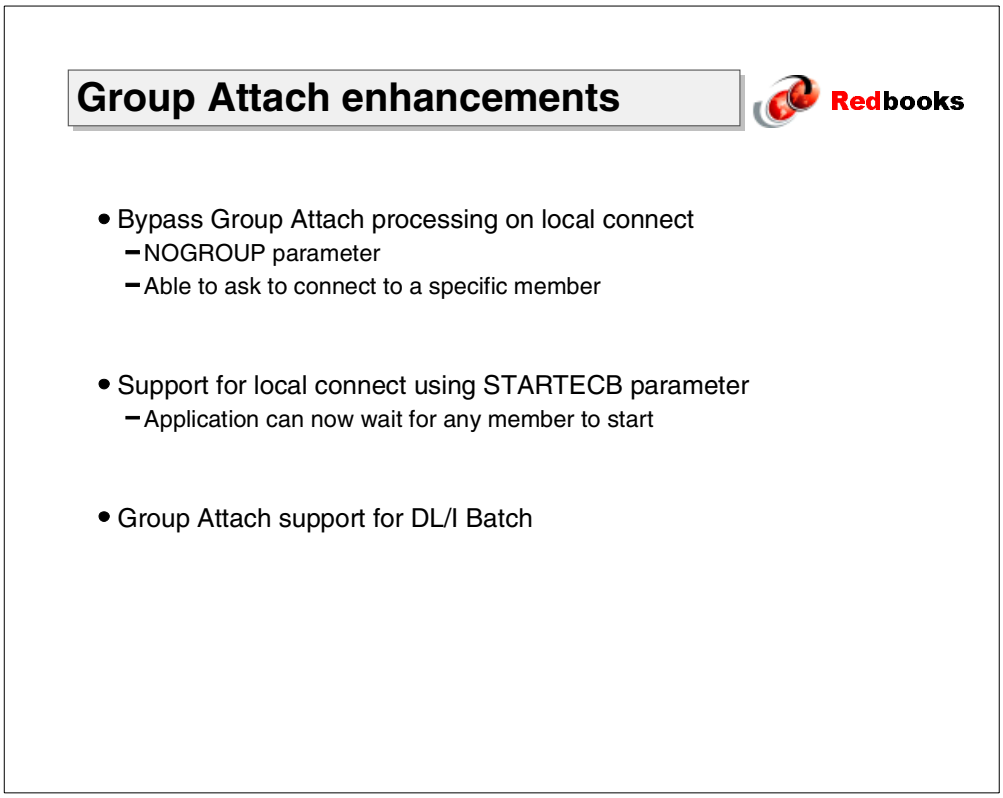
**Note:**

Ensure that the Coupling Facility Control Code is at least Level 7, Service Level 1.06, or Level 8, Service Level 1.03, or Level 9, before migrating any Data Sharing members to Version 7.

The CASTOUT(NO) option of the -STOP DB2 command, also introduced in DB2 V7, helps to reduce coupling facility utilization spikes and delays when a DB2 member is shut down. The DB2 member does not perform any castout processing when it is shut down.

In the past it has been difficult to determine the level of the CFCC running at any given time. Prior to OS/390 Version 2.8, the level of CFCC running was not externalized. The only way to safely determine what level of CFCC was running was to ask the system administrator. The OS/390 Version 2.8 enhances the `D CF` command output to externalize the level of CFCC:

```
D CF

- - - - -
CFLEVEL:           9
   CFCC RELEASE 09.00, SERVICE LEVEL 01.03
   BUILT ON 05/17/2000 AT 10:22:00
- - - - -
```



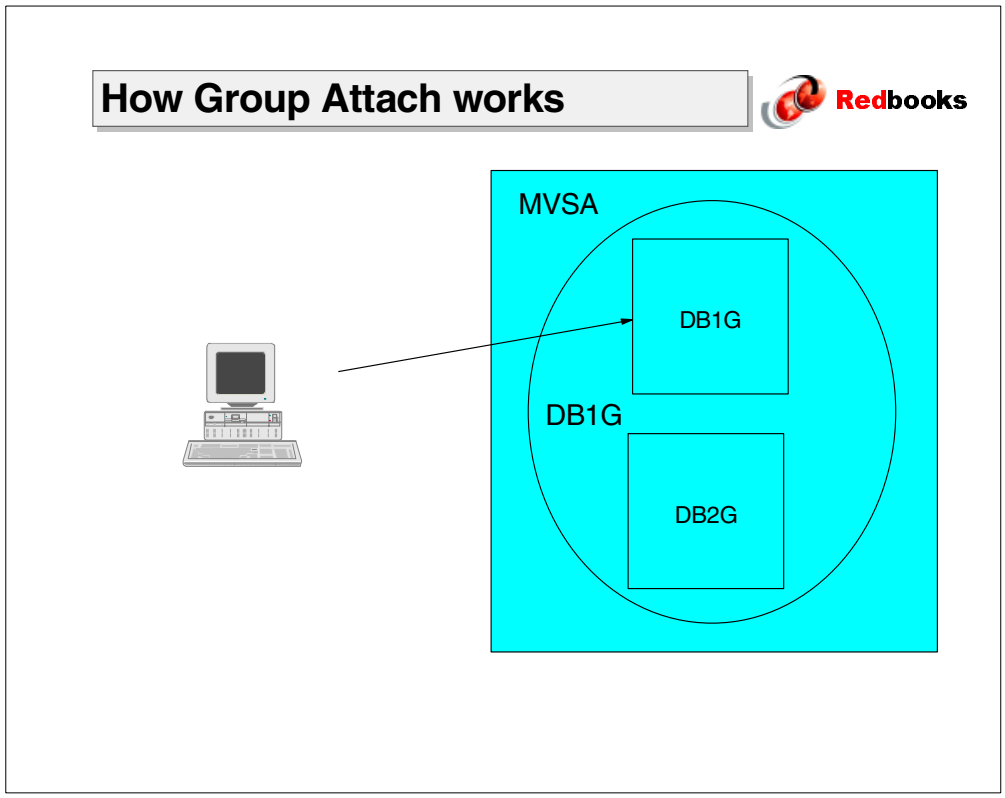

## 8.2 Group Attach enhancements

DB2 V7 introduces a number of enhancements to DB2 Group Attach processing.

***Bypass Group Attach processing on local connect***
An application can now connect to a specific member of a Data Sharing group, where there are two or more members of the Data Sharing group active on the same OS/390 image and the subsystem id of one of the members is the same as the Group Attach name. This is important for some applications that need to connect to specific members of the Data Sharing group, rather than generically connecting to any member of a given Data Sharing group (for example, monitoring applications.)

***Support for local connect using STARTECB parameter***
You can now specify the STARTECB parameter on a CONNECT call to DB2 using the Group Attach name, to wait for and be notified when any member of the Data Sharing group becomes active on this OS/390 image. Prior to this enhancement, the SRARTECB parameter is ignored if it was coded and you tried to connect to DB2 and Group Attach processing was invoked.

***Group Attach support for DL/I Batch***
DB2 V7 allows you to specify the Group Attach name for your DL/I Batch applications.




### 8.2.1 How Group Attach works

The DB2 Group Attach name is used to generically attach to any available member of the Data Sharing group. The Group Attach capability is essential in situations where a job or user logon can be dynamically routed to any system in the sysplex. Even in cases where work is not dynamically routed, Group Attach still provides ease-of-use single-system-image benefits by allowing you to code one set of JCL (or parameters) that can be used to connect to all the Data Sharing members, rather than having to change these parameters if work is moved from one system to another.

When you locally connect to DB2 on OS/390 and specify the Group Attach name on the connect call, DB2:

1. Assumes that the name you specified is a DB2 subsystem name and attaches to that subsystem if it is started.
2. If either of the following is true:
   - The name is not defined to the OS/390 image as a DB2 subsystem
   - A DB2 subsystem by that name is defined to the OS/390 image but not started and that subsystem's Group Attach name is the same as the subsystem name

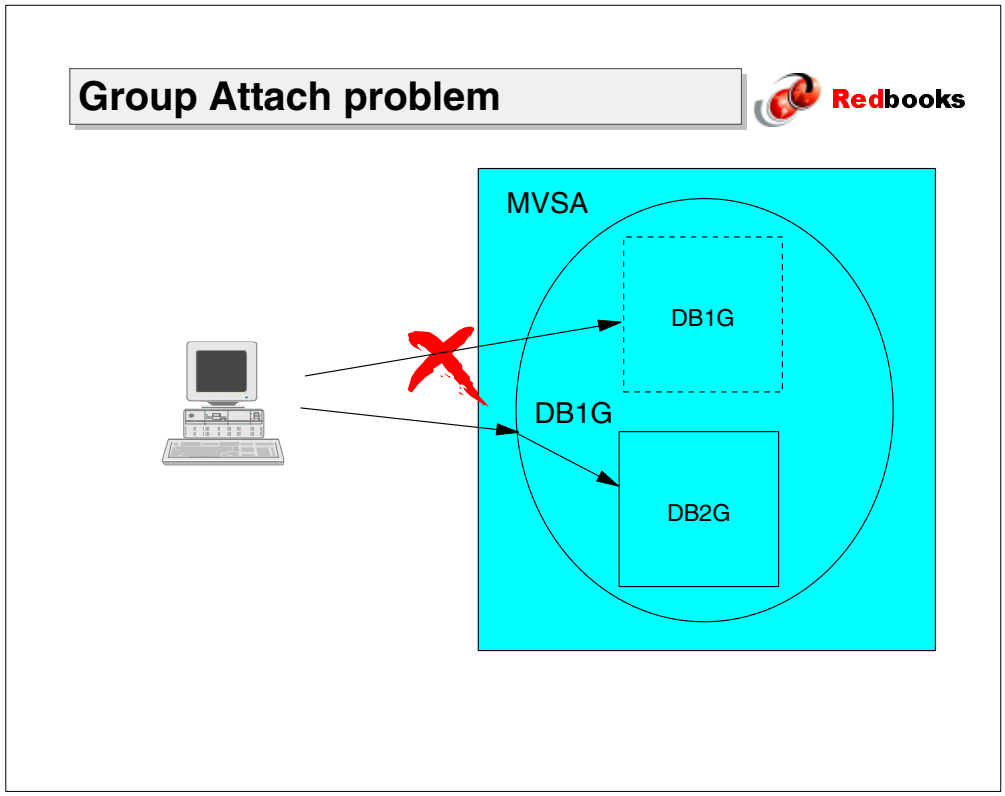
Then, DB2 checks to see if the name is a Group Attach name:

a. If the name is a Group Attach name, it constructs a list of DB2 subsystems that are defined to this OS/390 image, and tries to attach to each one in order, until it finds a DB2 subsystem that is started on this image, or until it reaches the end of the list. DB2 always attaches to the first “started” subsystem on the list. There is no load balancing.

b. If the name is not a Group Attach name, then a “not started” message is returned.

For example, assume two members DB1G and DB2G can run on the same OS/390 image. Assume also that the Group Attach name for the Data Sharing group is DB1G. Now let us assume there is an application using CAF that wants to attach to subsystem DB1G. The application issues a CAF CONNECT call passing DB1G as the `ssnm`. Because the DB2 member DB1G is active, the application connects directly to member DB1G.

However, if the Data Sharing group name is DB0G instead of DB1G and the application specified DB0G on the CONNECT call, Group Attach processing will connect the application to either DB1G or DC2G (whichever member was started first on this OS/390 image).

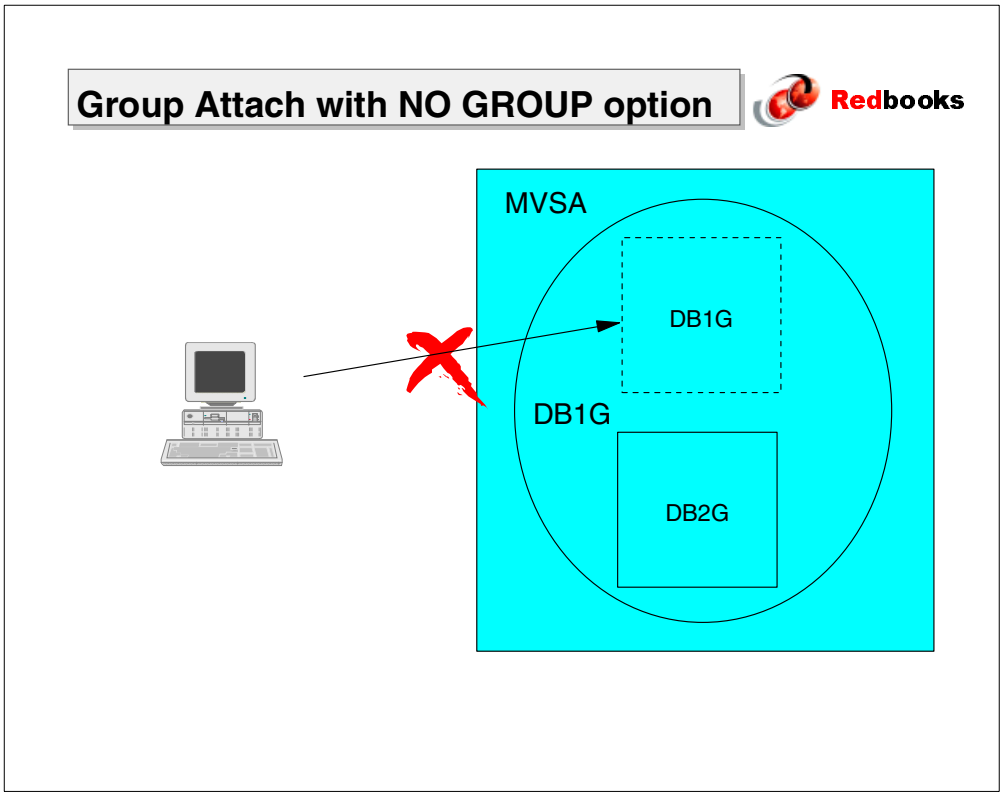



### 8.2.2 Group Attach problem

Now assume that the DB2 member DB1G is not active on this OS/390 image, and the application issues a CAF CONNECT call passing DB1G as the `ssnm`.

However, because the subsystem DB1G is not active and because DB1G is also the Group Attach name, the application connects to member DB2G. There is no way that the CAF application can connect to the subsystem DB1G and only DB1G.

When connecting from CAF, RRSAF, TSO attached, or running utilities, you cannot currently attach to a specific DB2 member, in the case where two or more members from the same Data Sharing group are running on the same OS/390 image, and the subsystem name of one of those members is the same as the Group Attach name.




### 8.2.3 Group Attach with NO GROUP option

DB2 V7 introduces a new parameter on the local connect call to DB2. The NO GROUP option indicates that the `ssnm` that is specified in the CONNECT call should always be treated as a subsystem name, even though the name may also match the Group Attach name.

In our example, the application issues a CAF CONNECT call with the NO GROUP parameter, passing DB1G as the `ssnm` parameter. The DB2 member DB1G is still not active. DB2 will bypass Group Attach processing and not search for a group name of DB1G.The application will not be connected to DB2G even though it is active on the same OS/390 image. Instead, a ‘connection not established’ error is returned to the application.

The NO GROUP parameter on CONNECT is only available to applications using CAF, RRSAF to connect to DB2. In addition, DB2 V7 introduces a GROUP(NO) parameter on the DSN command, for TSO attach to connect to DB2.

One example where a NO GROUP option is needed is where an application needs to have one instance of itself connected to each DB2 member of the Data Sharing group (for example, automation and monitoring functions). It is also useful for applications using RRSAF who need to reconnect to a specific DB2 member to resolve indoubt URs following a failure, during a cross-system restart. (This may also apply for CICS and IMS applications in the future when Group Attach is supported for these environments.)

### 8.2.4 Group Attach STARTECB support

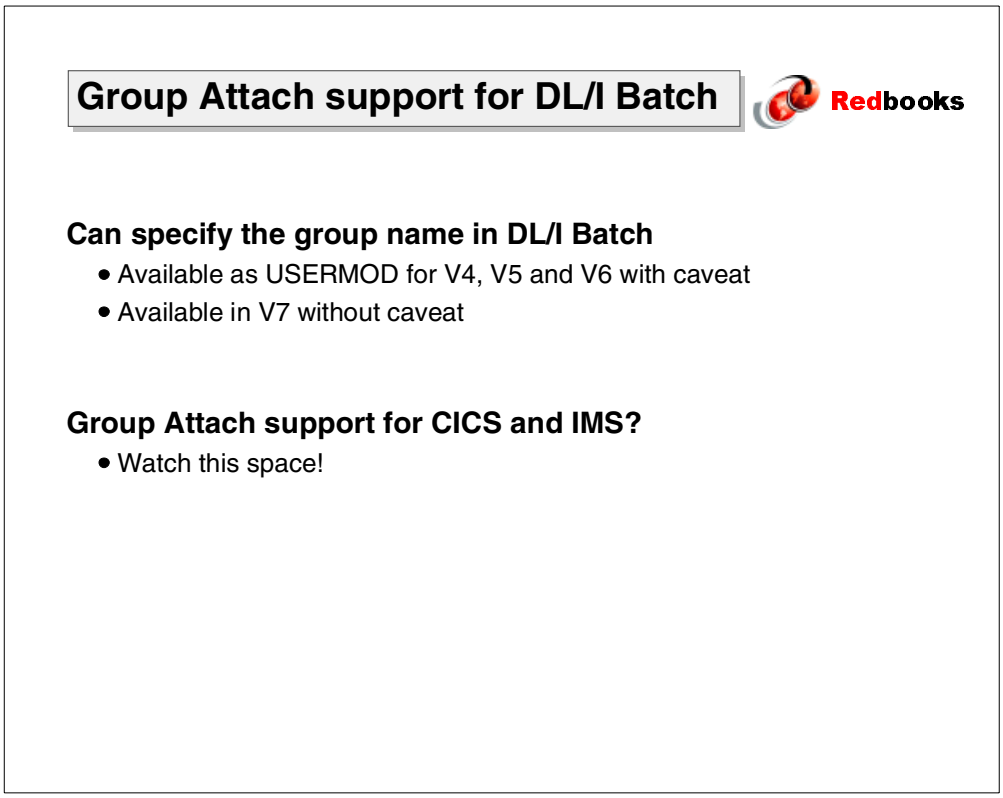
DB2 for OS/390 allows you to locally connect to DB2 and specify the STARTECB parameter on the connect call.

The STARTECB parameter of the CONNECT call to DB2 is used by applications who want to wait for the target DB2 subsystem to become available if it is currently not started. When the target DB2 comes up, it posts any startup ECBs that might exist to notify the waiting applications that DB2 is now available for work.

Prior to DB2 V7, DB2 ignored the STARTECB parameter on the CONNECT call, when the Group Attach name was coded.

DB2 V7 honors the STARTECB parameter when applications use it to connect to DB2 using the Group Attach name. The posting of the startup ECBs will be done by the first member of the Data Sharing group (as identified by the same Group Attach name) to start on the OS/390 image where the DB2 identify request came from.

For example, if the application tries to connect to DB0G and DB0G is the Group Attach name for DB2 subsystems DB1G and DB2G, then whichever of the subsystems starts first will post the startup ECBs.




### 8.2.5 Group Attach support for DL/I Batch

Prior versions of DB2 UDB for OS/390 do not allow DL/I Batch jobs to specify the Group Attach name when connecting to DB2.

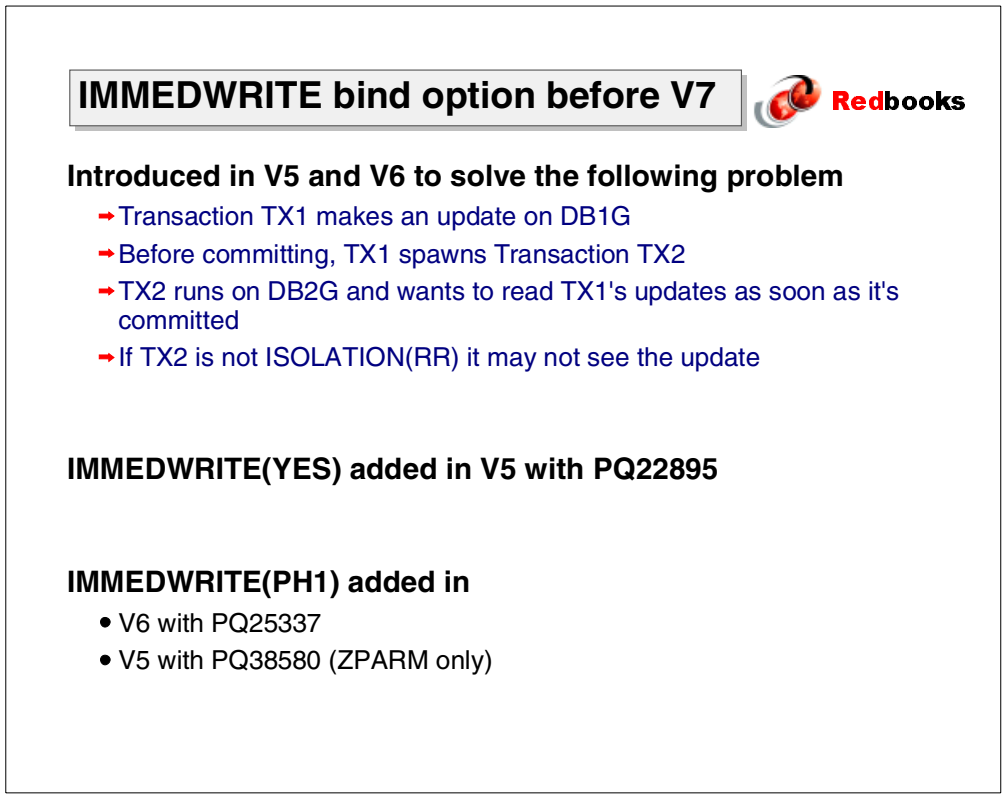
This restriction is becoming increasingly important to help in sysplex management and workload balancing across the whole sysplex.

There has been a continuing demand to have DL/I batch jobs submitted anywhere in the sysplex and connect to any DB2 member, rather than having to run on the same OS/390 image and connect to the same DB2 subsystem.

Limited support for using the Group Attach for DL/I batch is available in DB2 for MVS Version 4, DB2 UDB for OS/390 Versions 5 and 6 by arrangement with IBM Support, via APAR only. This support has restrictions. DB2 ignores the STARTECB parameter on the CONNECT call, when the Group Attach name is coded. DB2 therefore behaves differently, depending on if you specify the Data Sharing group name or DB2 subsystem name with the STARTECB parameter.

Now that STARTECB is supported for Group Attach, the incompatibility is removed and therefore Group Attach support can now be safely added for DL/I Batch without any incompatible behavior being introduced.

DL/I batch does not support 2 phase commit with DB2. DB2 can resolve any indoubt units of recovery caused by DL/I batch and/or DB2 failures without requiring the DL/I batch unit of work having to reconnect to the same DB2 subsystem after the failure.




## 8.3 IMMEDWRITE bind option

In this section we describe the IMMEDIATEWRITE Bind option: its introduction with DB2 V6 and its evolution with V7.

### 8.3.1 IMMEDWRITE Bind option before V7

Consider this scenario: A transaction performs a data base update from one DB2 member and then, before committing, spawns a second ‘dependent’ transaction. The second transaction executes on a different DB2 member and contains logic which depends on the updates that were made by the originating transaction.

However, if the dependent transaction is not running with ISOLATION(RR), then there is a chance, due to lock avoidance and the fact that the two members do not share a common virtual buffer pool, that the dependent transaction will not find the update that was made by the originator if the originator has still not yet committed.

Possible workarounds for this problem are the following:

- Always run the dependent transaction on the same member as the originating transaction
- Bind the dependent transaction with ISOLATION(RR)
- Have the originating transaction wait until after commit to spawn the dependent transaction.

APAR PQ22895 delivered in DB2 V5, a new bind/rebind option that can be used when none of the above actions are desirable. IMMEDWRITE(NO|YES) allows the user to specify that a given plan or package spawns dependent transactions that may run on other members, and DB2 should immediately write updated group buffer pool dependent buffers to the coupling facility, instead of waiting until commit or rollback. An “immediate write” means that the page is written to the group buffer pool as soon as the buffer update completes. This may have some performance impact.

However, IMMEDWRITE(YES) is not the optimum solution when:

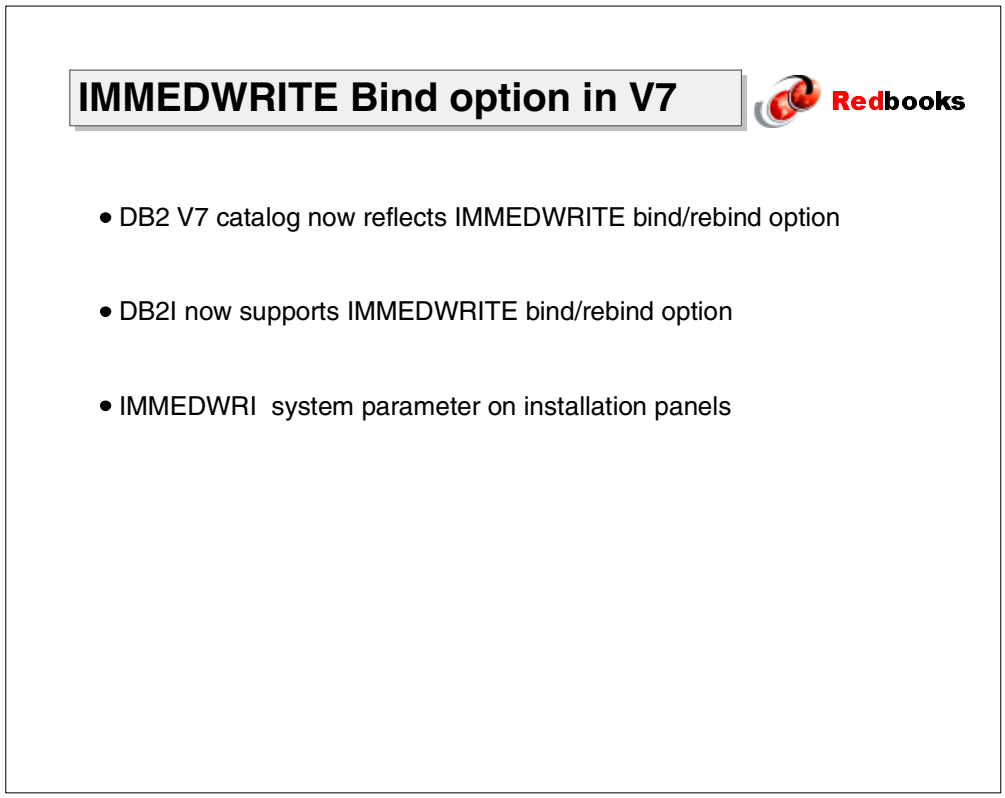
- The exact plans/packages which need IMMEDWRITE(YES) can not be identified and rebinding all plans/packages is not a feasible alternative, or
- There is a high concern for the performance impact of specifying IMMEDWRITE(YES).

APAR PQ25337 delivered in DB2 V6, the functionality introduced by PQ22895 in DB2 V5 with the addition of a third value for IMMEDWRITE and a new DSNZPARM parameter. These enhancements provide more flexibility regarding when the group buffer pool dependent pages are written and which plans/packages are affected.

IMMEDWRITE(PH1) allows the user to specify that a given plan or package should write updated group buffer pool dependent buffers to the coupling facility at or before Phase 1 of commit, instead of waiting until Phase 2 of commit or rollback. If the transaction subsequently rolls back, the pages will be updated again during the rollback process, and they will be written again at the end of abort.

A new DSNZPARM parameter added to DSN6GRP, IMMEDWRI(NO|PH1|YES), allows the user to specify at a DB2 member level whether immediate writes or Phase 1 writes should be done. IMMEDWRI effects all plans and packages that are run on this member, except those that are bound with this option.The default for REBIND PLAN/PACKAGE is the IMMEDWRITE value used the last time the plan/package was bound.

Refer to the *DB2 UDB for OS/390 Version 6 Technical Update,* SG24-6108, for a more detailed discussion of the IMMEDWRITE bind/rebind option.




## 8.3.2 IMMEDWRITE BIND option in V7

DB2 V7 further extends the support for IMMEDWRITE(NO|PH1|YES).

The DB2 catalog now supports the IMMEDWRITE BIND/REBIND option by adding a new column to both SYSIBM.SYSPLAN and SYSIBM.SYSPACKAGES. In DB2 Version 5 and Version 6 this information is ‘hidden’ in the catalog.

The bind/rebind panels of DB2I now support the IMMEDWRITE bind/rebind parameter.

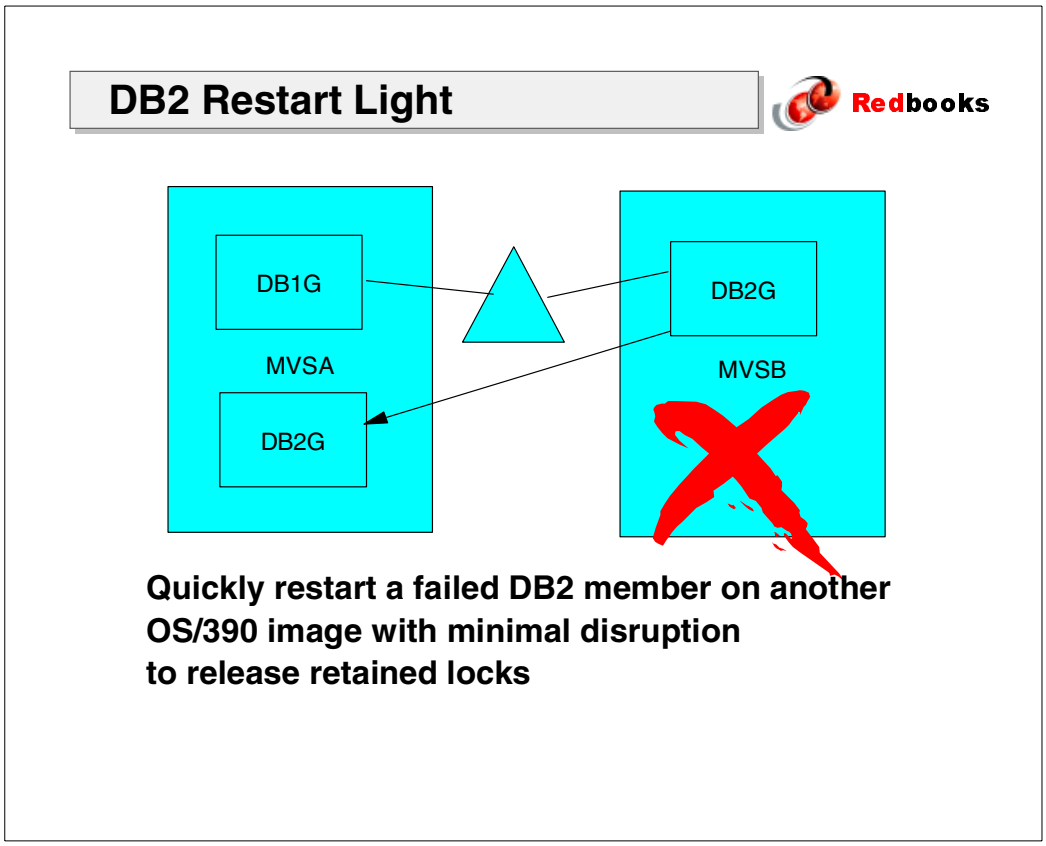
DB2 V7 also externalizes the IMMEDWRI DSNZPARM parameter to the installation panels.

### 8.3.2.1 IMMEDWRITE BIND option performance

IMMEDWRITE(YES) should be used with caution because of its potential impact to performance. The impact will be more significant for plans/packages that do many buffer updates to group buffer pool dependent pagesets/parts, and not as noticeable for plans/packages that do few buffer updates to group buffer pool dependent pagesets/parts.

IMMEDWRITE(PH1) should have little or no impact on overall performance. However, when IMMEDWRITE(PH1) is used, some of the CPU consumption will shift from being charged to the MSTR address space to being charged to the allied agent's TCB. This situation occurs because the group buffer pool writes are done at commit Phase 1, which is executed under the allied TCB instead of being done at commit Phase 2, which is executed under SRBs in the MSTR address space.

Another IMMEDWRITE(PH1) potential performance consideration is that for transactions that abort after commit Phase 1 has completed, there will typically be about double the amount of group buffer pool writes for IMMEDWRITE(PH1) as compared to normal. This situation occurs because each updated group buffer pool dependent page will be written out once during Phase 1 and again at the end of abort, instead of being written out only once at the end of abort, as would be typical for the IMMEDWRITE(NO) case.




## 8.4 DB2 Restart Light

Data Sharing customers have expressed concern that there is no good way to release DB2 retained locks in the case where an OS/390 image in a parallel sysplex has failed. The only options available are to:

- Re-IPL the failed OS/390 image and then restart the failed DB2 member after the IPL has completed, or
- Restart the failed DB2 member on another OS/390 image in the sysplex, to recover the retained locks. Then bring that member down once the retained locks have been removed. The DB2 member can then be restarted on its original OS/390 image after it becomes available

The problem with the first alternative is that the outage time is extended to wait for the re-IPL of the OS/390 system. This is unacceptable for some users.

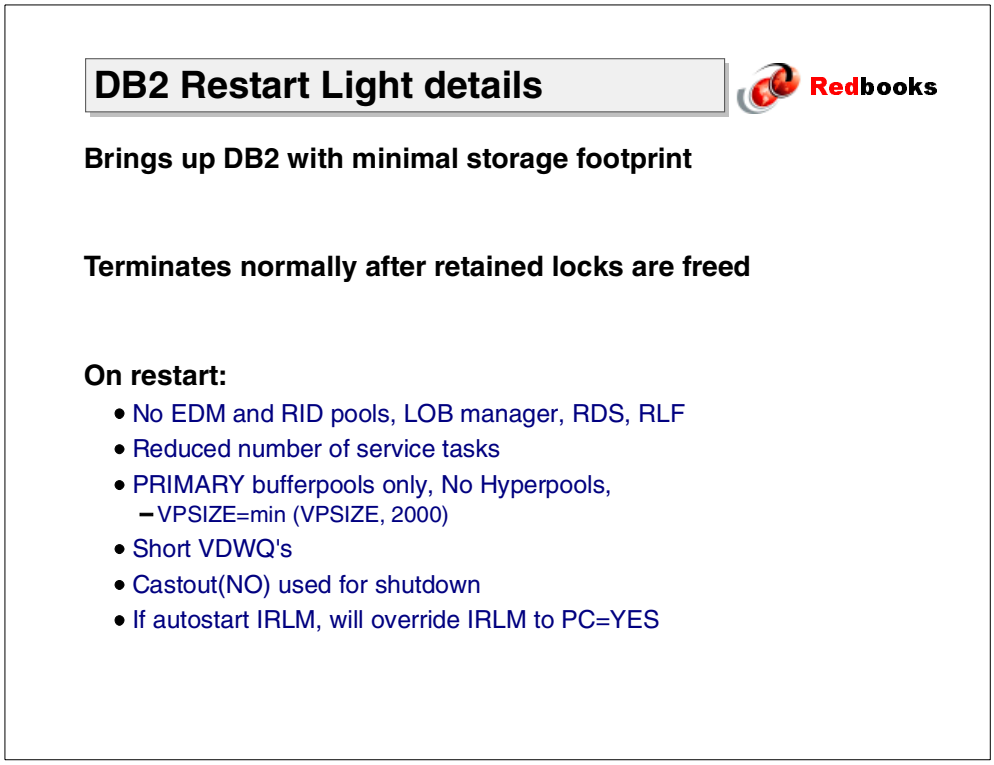
The main problem with the second alternative is that a significant amount of additional memory and ECSA needs to be configured on the OS/390 images that need to accommodate cross-system DB2 restarts. Even with the additional memory configured, the cross-system DB2 restart can cause significant paging and disruption to the workload that is already running on that OS/390 image, due to the large amounts of memory DB2 requires during restart.

A new parameter is added to the START DB2 command to indicate that DB2 is to be restarted in “light” mode, LIGHT(YES).

DB2 Restart Light provides a better alternative to recover retained locks in the OS/390 failure scenarios, Geographically Dispersed Parallel Sysplex (GDPS) fall-over scenarios and DB2 Data Sharing disaster recovery scenarios. DB2 Restart Light provides a way to restart DB2 with a minimal storage footprint. recover the retained locks, and then terminate normally. This provides a more effective alternative to quickly recover retained locks in cross-system restart scenarios

In a non-data-sharing environment, the RESTART(LIGHT) parameter is ignored, and restart processing continues a normal. The following message is generated:

```
DSNY015I  =DB1G DSNYSTRT LIGHT(YES) ON START COMMAND
WAS IGNORED, SYSTEM IS NOT ENABLED FOR DATA SHARING
```




***DB2 Restart Light details***

When restarted with the Restart Light parameter, DB2 does the following:

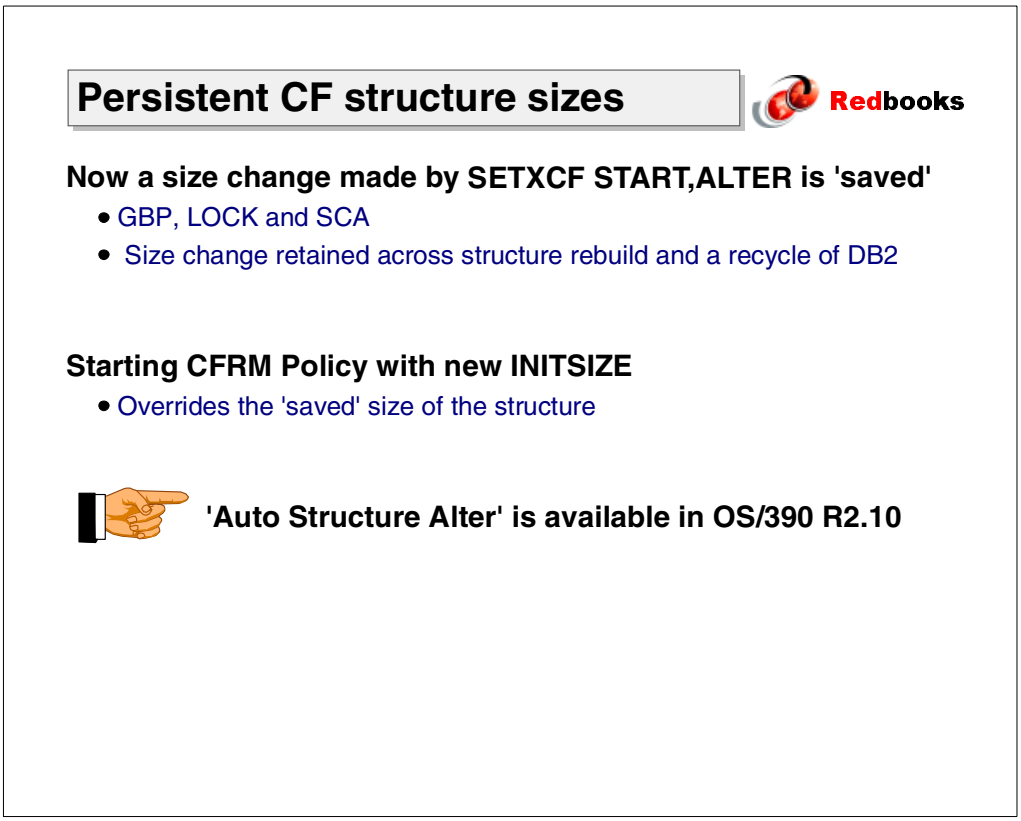
- Minimize the overall storage requirement for restart
- Recover the retained locks asap. All retained locks are removed except the following:
  - Locks that are held by indoubt units of recovery
  - Locks that are held by “postponed abort” units of recovery
  - ‘IX’ mode pageset P-Locks. These do not block access from other members; however, they do block drainers (for example, some utilities).
- After forward and back out recovery is complete, DB2 terminates normally without accepting any new work. DB2 does not register with ARM if it is performing a Restart Light, so that when DB2 terminates after the Restart Light, ARM will not automatically restart the DB2 member again.

DB2 takes the following actions to minimize storage:

- Smaller virtual pools are GETMAIN’d.
- No hiperpools are allocated.
- No EDMPOOL is GETMAIN’d.
- DDF is not started.
- IRLM is started with PC=YES to reduce ECSA requirements.
- Restart processing that is not essential for recovery of retained locks is skipped.

To utilize Restart Light in an ARM environment, an ARM policy must be put in place for DB2 that specifies the LIGHT(YES) keyword for a cross-system DB2 restart that is taking place as a result of a failed OS/390 image.

To take full advantage of all the benefits of Restart Light, the IRLM would need to be started with PC=YES. This will cause the IRLM to store locks in private storage rather than ECSA, reducing the impact on potentially critical ECSA storage. Usually the IRLM is autostarted by DB2, in which case DB2 will automatically restart the IRLM with PC=YES. If DB2 does not automatically restart the IRLM, then either the user would need to restart the IRLM with PC=YES manually, or in an ARM environment, an ARM policy is needed to restart the IRLM specifying PC=YES as a restart parameter, for a cross-system restart due to a system failure.




## 8.5 Persistent CF structure sizes

Changing the size of a coupling facility structure can be accomplished in one of two ways:

1. Alter the CFRM Policy with a new INITSIZE, compile and start the new policy, then rebuild the structure if it is currently allocated, or
2. Use the SETXCF START,ALTER command to dynamically increase the size. This approach assumes that the SIZE parameter in the CFRM Policy is specified and sufficient to support the new value.

If the second method is chosen, then the following problems exist:

- When rebuilding the coupling facility structure (for example, to move the structure from one coupling facility to another for coupling facility maintenance), the new structure reverts back to the original size, INITSIZE. Usually it is preferable to have the structure maintain its size across the rebuild.
- When establishing duplexing for the group buffer pool coupling facility structure, the secondary structure is allocated using the INITSIZE, instead of using the current size of the primary structure. The secondary structure will become a different size than the primary structure. Usually it is preferable to have the size of the secondary group buffer pool the same size as the primary group buffer pool.

DB2 V7 uses the currently allocated size of the SCA, Lock and GBP structures:

- When allocating a new coupling facility structure instance in response to a structure rebuild
- When allocating a secondary structure to support duplexing
- When allocating a new coupling facility structure, after the size was changed by a SETXCF START,ALTER command and the structure was subsequently deallocated.

DB2 now stores the currently allocates size of the Lock, SCA and GBP structures in the BSDS and these sizes will be used when DB2 needs to allocate the structures. DB2 will use the INITSIZE, if no SETXCF START,ALTER command was issued against the structure.

The Lock and SCA structures already have a form of size persistence, since the structures remain allocated while all members of the Data Sharing group are stopped. Group Buffer Pools (GBP) on the other hand are not.

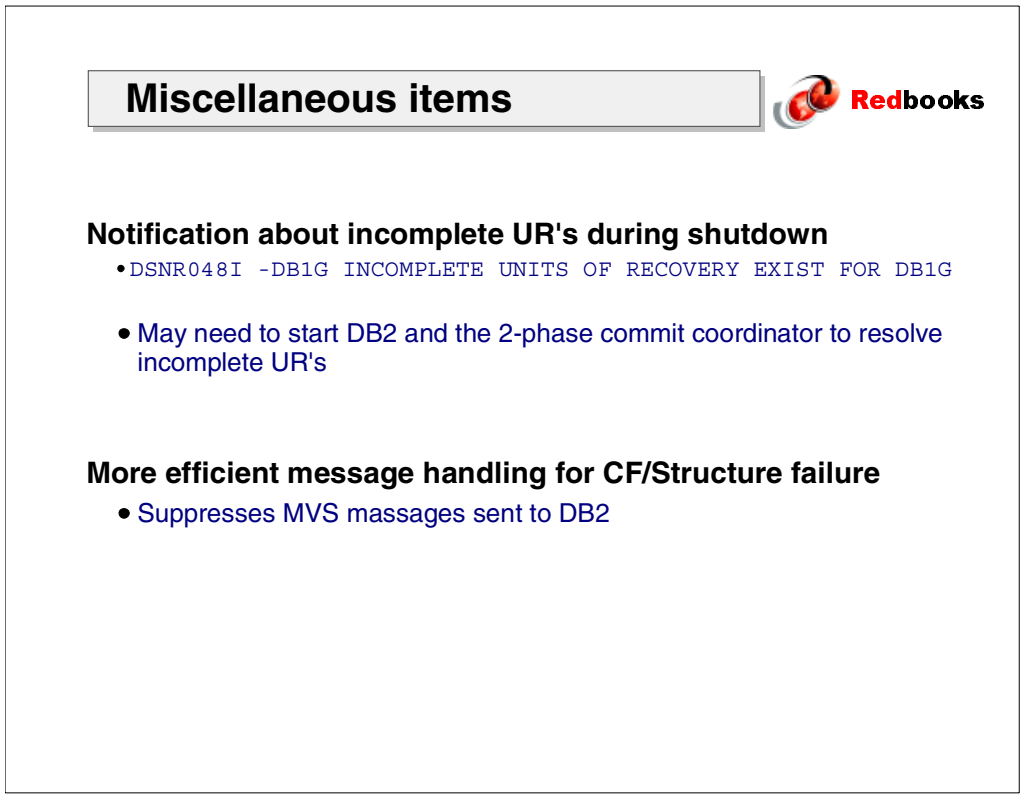
The SCA will be allocates using the current size when the structure is rebuilt as a result of a group restart. However, the Lock structure is handled a little differently because IRLM itself does not have any permanent storage (like DB2's BSDS) at its disposal to remember the last allocated structure size. So in cases where the entire Data Sharing group comes down, and all the coupling facility structures have been deallocated, when the group comes back up, the Lock will go back to its INITSIZE and SCA and GBPs will be initialized to the last remembered (in the BSDS) allocated size. This may be a consideration for disaster recovery scenarios, or system cloning scenarios but unlikely to be a factor in the normal operation of a DB2 Data Sharing group.

Please remember however, that the “saved” current size used for subsequent structure allocations, is overridden by a new INITSIZE parameter value in a new CFRM Policy.

DB2 manages the GBP directory ratio and lock/list ratios in the same way as in pervious Versions. The number of directory entries is dynamically increased when the GBP size increases or the ratio is increased. The number of directory entries is decreased only when the GBP is reallocated. Only the space allocated for “modify locks” is increased when the Lock structure size is increased. You can only increase the number of lock hash entries only by reallocating the lock structure.

DB2 V7 will also provide the flexibility to request a varying percentage of the lock structure to be allocated to the hash table. Currently this ratio is fixed by DB2 to 50%. This enhancement may not be available at GA, but should be availably shortly after GA.

OS/390 Version 2 Release 10, introduces a further enhancement, called “Auto Structure Alter”, for coupling facility structure management. OS/390 dynamically monitors the size and usage of structures allocated in the coupling facility. OS/390 also monitors the usage of directory entries within the group buffer pools and the usage of the lock and list structures within the DB2 Lock structure. OS/390 will dynamically change the size of all the structures and the number of GBP directory entries, depending on current usage. OS/390 cannot change the number of hash entries in the Lock structure. It will only change the size of the modify lock list. OS/390 will pass on all changes to the structure sizes and ratios to DB2, so these changes made by OS/390 will not be regressed when DB2 asks to rebuild the structures.




## 8.6 Miscellaneous items

DB2 V7 introduces a number of other enhancements to facilitate the usability and management of DB2 Data Sharing environments.

***Notification about incomplete units of recovery***
DB2 V7 produces the message DSNR046I during normal DB2 member shutdown, if there will be any retained locks remaining due to incomplete units of recovery. (Incomplete URs may exist due to the “recover postponed” feature, in V6) These retained locks will continue to block access to the affected DB2 data from other members.

If this message is issued, then you may choose to immediately restart the DB2 member in order to resolve the incomplete URs and remove the retained locks.

This warning is given in addition to the existing DSNR036I message that notifies DB2 you at each DB2 checkpoint of any unresolved indoubt URs.

***More efficient message handling for coupling facility/structure failures***
DB2 V7 reduces the MVS to DB2 communication that occurs during a coupling facility/structure failure. This enhancement should simplify recovery and could improve recovery time.

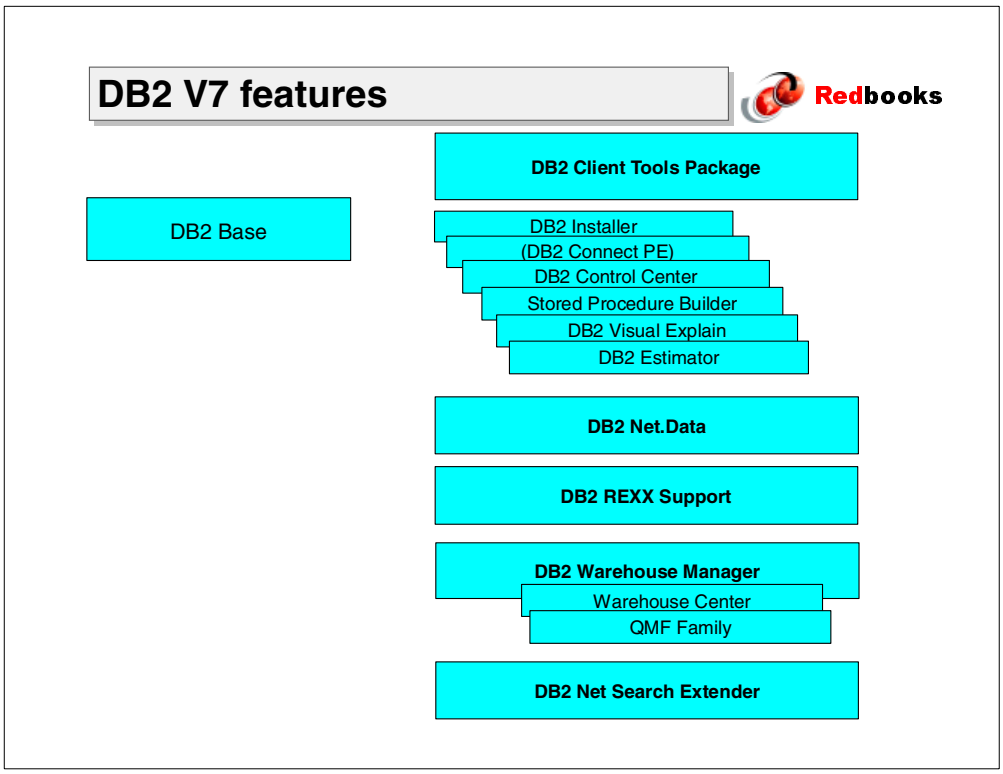
During coupling facility failure scenarios, the event exit for each DB2 member is invoked multiple times by MVS to help resolve the situation. A number of these events are “ignored” by DB2. OS/390 has shipped an enhancement to suppress some of these unnecessary events. DB2 V7 uses this new function to expedite group buffer pool failure recovery for both the coupling facility failure and coupling facility link failure scenarios. DB2 now connects to coupling facility structures by specifying the SUPPRESSEVENTS keyword to the IXLCONN invocation.

This enhancement should reduce message traffic in the sysplex and improve the switch time for group buffer pools and therefore improve the availability of a coupling facility or group buffer pool failure for a duplexed group buffer pool.

# Part 7. DB2 features and tools

© Copyright IBM Corp. 2001

# Chapter 9. DB2 features




## 9.1 DB2 Management Clients Tools Package

The DB2 for OS/390 and z/OS Version 7 Management Clients Tools Package feature (a rename for the previous DB2 for OS/390 Management Tools Package feature) is the collection of the following workstation-based tools:

- DB2 Control Center
- DB2 Installer
- DB2 Visual Explain
- DB2 Estimator
- DB2 Stored Procedure Builder

Assistance is provided through the IBM Support Centers on all of these five tools, as well as for the main product.

### 9.1.1 DB2 Control Center

The DB2 Control Center is a graphical interface for administering database objects for the Universal Database family of products for OS/2, Windows, and UNIX platforms. You can use the Control Center as your main point of administration to manage systems, DB2 instances, databases, and database objects, such as tables, views, and user groups. You can also use the Control Center to concurrently access DB2 subsystems. The Control Center only contains the replication administration for DB2 OS/400 and DB2 VM/VSE.

© Copyright IBM Corp. 2001

The DB2 Control Center can run either as a Java application or as an applet on your Web server, which your Web browser can access. The DB2 Control Center is part of the DB2 Software Developers Kit (SDK) on Windows, delivered with all editions of DB2 Universal Database and DB2 Connect products on Linux, OS/2, UNIX, and Windows. Because the Control Center requires DB2 Connect, the DB2 Management Tools Package provides a restricted-use copy of DB2 Connect Version 6 to satisfy this functional dependency.

With DB2 V7, the Control Center supports:

- Utility Wildcarding and Dynamic Allocation with the usage of data set templates and object lists
- Utilities (Copy, Concurrent Copy, Quiesce, Reorg) invocation on an object list
- Utility procedures to create, create like, delete, show statements, run
- Other generic CC functions: Locate, Filter, Show Related, Customize CC
- Defer defining of DB2 data sets
- DEFINE YES/NO option in CREATE TABLESPACE and CREATE INDEX
- Defer creation of VSAM data sets until the first write operation
- SQL Procedures by allowing Stored Procedure Builder to be launched
- User defined utility ID support
  - New OS/390 page in the Tools Settings notebook
  - User defines a utility ID template using a rich set of variables (such as USERID, UTILNAME, MONTH, DAY, HOUR)
  - Users have option to edit the utility ID generated before invoking a utility
- Ability to restart DB2 for OS/390 utilities
  - Restart action is accessible via Display Utility dialog
  - Restart from last committed point or the last committed phase
  - Can only restart utilities that are started from within CC
- Generation of DDL statements that can be used to recreate database objects and optionally the dependent objects
  - Requires DB2Admin to be installed on the OS/390 host
  - Invokes DB2Admin via a stored procedure ADB2RE
  - Generates and saves to the host DDL for databases, table spaces, tables, procedures, schemas, user defined types, user defined functions
- First step towards System Management by managing OS/390 data sets with new stored procedures

### 9.1.2 DB2 Installer

DB2 Installer provides a graphical user interface for customizing DB2 for OS/390 installation jobs on the workstation. It provides an alternative to the existing ISPF installation panels and CLISTs currently used on OS/390 systems.

DB2 Installer lets you install, migrate, and update your DB2 for OS/390 subsystem. The most current version also provides an extended support for the installation of DB2 Data Propagator and DB2 Performance Monitor. It is particularly suitable if you are customizing DB2 for OS/390 for the first time. If you are already an experienced installer, you can use DB2 Installer to increase your productivity.

The DB2 Installer application illustrates the overall installation process and keeps a graphical record of how each subsystem is defined. The graphical interface follows an easy-to-read map through the entire installation process including SMP/E, fallback and sample jobs. In addition, it provides a graphical record of completed and uncompleted tasks by subsystem.

You can customize the DB2 subsystem as much or as little as needed using DB2 Installer. You can install a basic subsystem quickly or modify every installation option.

DB2 Installer presents parameters that you must customize in the main windows, while parameters that can assume default values are available in secondary windows. Also, help options are available throughout DB2 Installer.

Once your DB2 subsystem settings have been customized, DB2 Installer gives you the option of using a TCP/IP connection to either transfer the edited jobs to the host or send the jobs directly to the OS/390 system execution queue. If you don't have TCP/IP, once you have customized your installations jobs, you will need to use a method outside of DB2 Installer to move jobs from the workstation to OS/390 for execution.

DB2 Installer makes it easy to change the subsystem parameters as well as keep track of the settings for several different subsystems. If the application is installed on a LAN, several users can share the access and tracking of DB2 subsystem settings. The flexibility of DB2 Installer allows you to use it in a way that best meets the needs of your site. You can also run some jobs directly on the host and some from DB2 Installer.

DB2 V7 brings the following new functions to Installer:

- Option to call a stored procedure to return configuration parameters directly from DB2. This permits a user doing migration to "ask" a DB2 for OS/390 subsystem to return its own configuration parameters. The user must still provide non-configuration parameter information such as the names of program product data sets.
- Option to generate a CLIST-compatible input member (DSNTIDxx) from the Installer parameter dictionary, and to store this member in a PDS on the host. This permits an installer user to switch to the CLIST more conveniently.
- "New!" icon to draw attention to install fields introduced or changed since the previous version of DB2. This eyecatcher helps the user identify areas of impact during migration.
- Scrollable windows that fit on laptops with 800 x 600 resolution.
- Skip-release migration from DB2 V5 or DB2 V6
- Continued support of Installation, Migration, SMP/E, Enable Data Sharing, Add Member, and Update, but now enhanced for V7
- DB2 Installer includes support for other features in the DB2 UDB:
  - DB2PM - SMP/E and Install
  - DpropR - SMP/E

### 9.1.3 DB2 Visual Explain

DB2 Visual Explain helps database administrators and application developers:

- See the access path for a given SQL statement
- Tune SQL statements for better performance
- View the current values for the DB2 subsystem

DB2 stores the results from EXPLAIN in the plan table, that describes your statement's access path. Interpreting these results can be difficult and time consuming. Visual Explain graphs the output, indicating the key objects and operations that comprise your statement's access of data. This graph and its associated features help you to better grasp the information given in the plan table.

The graph of the access path is displayed on an IBM OS/2 or Microsoft Windows NT workstation.

You can EXPLAIN SQL statements dynamically and immediately, and graph their access path. You can enter the statement, have Visual Explain read it from a file, or extract it from a bound plan or package.

The graphical representation of the access path allows you to instantly distinguish operations such as a sort, parallel access or the use of one or more indexes. You can view suggestions from the graph that describe how you might improve the performance of your SQL statement.

Visual Explain allows you to filter capabilities by access path of EXPLAINable SQL statements. For example, you can choose to only display statements that contain a sort or have an estimated cost greater than 500 milliseconds.

The report feature of Visual Explain, invoked through the Report Selection Wizard, allows you to view, save into a file or print the access path descriptions, statistics, SQL text and cost of any number of EXPLAINable SQL statements.

Also available through Visual Explain is the capability for you to browse the real time settings of the subsystem parameters (stored in DSNZPARM) and parameter settings needed for DB2 applications (stored in DSNHDECP).

DB2 Visual Explain makes Distributed Relational Database Architecture (DRDA) queries through a DB2 client on the workstation to get the information it needs. The subsystem parameter values are retrieved by calling a stored procedure on the host, which makes an IFI call to the appropriate DB2 trace record and returns them to the workstation.

### 9.1.4 DB2 Estimator

DB2 Estimator is a tool for your personal computer which runs on Windows. This tool is meant to help you predict the expected performance and cost of running DB2 applications and utilities under DB2 for OS/390. Here are some of its uses:

- DB2 Estimator can help you determine the cost of future applications or modifications to existing applications, measure workloads, and identify the cost variations associated with different hardware and software design approaches.
- You can also estimate the disk space required for tables and indexes.

- You can use DB2 Estimator to create definitions of DB2 tables, views, SQL statements, transactions and applications, DB2 utilities, and system configurations. These definitions are then used by DB2 Estimator to compute capacity usage and performance estimates.
- You can predict expected changes in the capacity and performance of your DB2 applications.
- You can also produce graphs and reports to help you compare these changes.
- DB2 Estimator lets you model and experiment with DB2 applications and utilities on your personal computer without the need to use DB2.
- With DB2 Estimator you can answer questions such as these:
  - What is the elapsed time for an SQL statement that fetches a specified number of rows?
  - How much processor resource is used during an N-way join?
  - What is the impact of adding and dropping an index from a table?
  - What is the impact of normalizing or de-normalizing a table?
  - Can my system support an anticipated increase in workloads?
  - If I double the amount of processor resource, what is the effect on transaction response time?
  - How much storage do I need for the new table and its indexes?
  - What is the effect on performance if my table doubles in size?
  - Should I partition my table?
  - What is the effect of compression?
  - What is the server cost of a distributed SQL?
  - What is the effect of data sharing?
  - Can my application complete within the batch window?
  - How long will my utility job take?
  - What effect does a particular trigger have on an SQL?
  - Can I use stored procedures effectively?

New functions in DB2 Estimator V7 are:

- DB2 V7 UNLOAD utility
- DB2 V7 Parallel LOAD Partitions
- DB2 V7 UNICODE
- Better bulk handling of table and SQL objects, for usability
- Partitions taken into account by Capacity Runs
- ORDER BY of columns not found in the SELECT clause
- DB2 V5, V6, and V7 projects

Also, more work is under way to fully support all V7 functions.

### 9.1.5 Stored Procedure Builder

Stored Procedure Builder is a graphical application that provides an easy-to-use development environment for creating, installing, and testing stored procedures, allowing you to focus on creating your stored procedure logic rather than the details of registering, building, and installing stored procedures on a DB2 server.

Additionally, with SPB, you can develop stored procedures on one operating system and build them on other server operating systems.

Using SPB, you can perform the following tasks:

- Create new stored procedures
- Build stored procedures on local and remote DB2 servers
- Modify and rebuild existing stored procedures
- Run stored procedures to test the execution of installed stored procedures
- Debug stored procedures

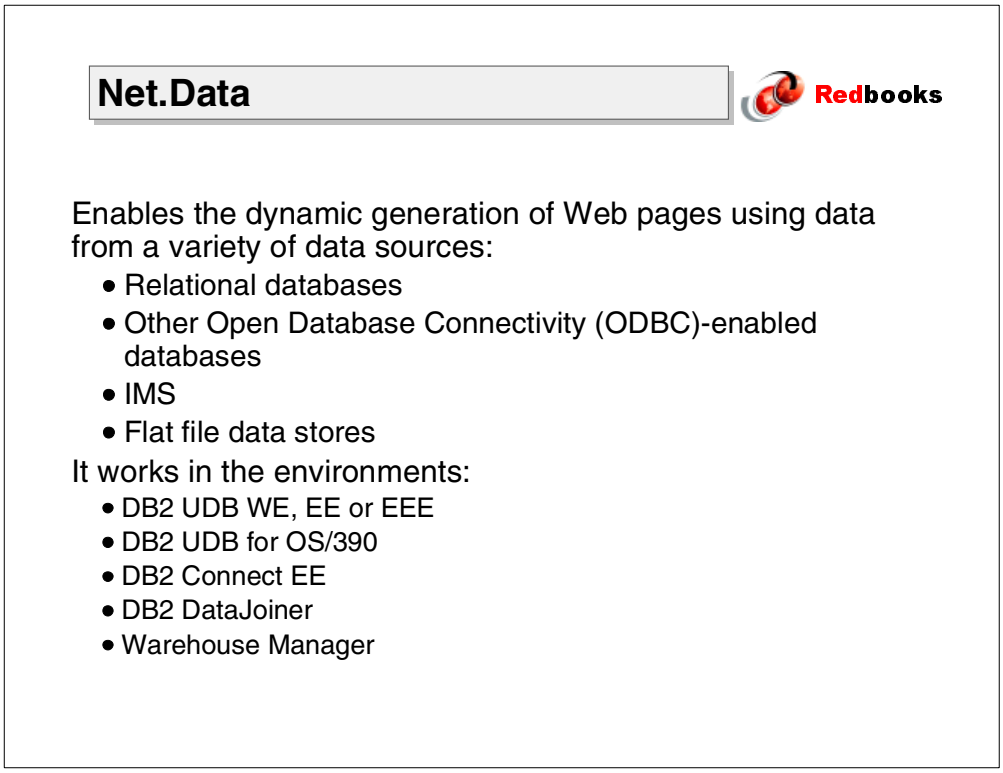
SPB manages your work by using projects to simplify the task of building applications. Each SPB project saves the following information:

- Your connections to specific databases.
- The filters you created to display subsets of the stored procedures on each database. When opening a new or existing SPB project, you can filter stored procedures so that you view stored procedures based on their name, schema, language, or collection ID (for OS/390 only).
- Stored procedure objects that have not been successfully built to a target database.

SPB provides a single development environment that supports the entire DB2 family ranging from the workstation to OS/390. You can launch SPB as a separate application from the IBM DB2 UDB program group, or you can launch SPB from any of the following development applications:

- Microsoft Visual Studio
- Microsoft Visual Basic
- IBM VisualAge for Java

SPB is implemented with Java and all database connections are managed by using a Java Database Connectivity API (JDBC). Using JDBC, you can establish a connection to a relational database, send SQL statements, and process the results. To write stored procedures with SPB, you only need to be able to connect to a local or remote DB2 database alias using a JDBC driver. You can connect to any local DB2 alias or any other database for which you can specify a host name, port, and database name. Several JDBC drivers are installed with SPB.




## 9.2 Net.Data

IBM Net.Data extends existing Web servers by enabling the dynamic generation of Web pages using data from a variety of data sources. The data sources can include relational and non-relational database management systems such as DB2, Oracle, Sybase, Open Database Connectivity (ODBC)-enabled databases, IMS and flat file data stores. Net.Data applications can be rapidly built using a macro language that includes conditional logic and built-in functions. Net.Data allows reuse of existing business logic by supporting calls to Java, C/C++, Perl, RPG (OS/400 only) and REXX.

Net.Data provides several features for high performance including persistent connections to the database and a cache manager for rapid serving of frequently accessed Web pages.

Net.Data is available on Windows NT, AIX, Sun Solaris, HP/UX, SCO UnixWare, OS/2, OS/390, and OS/400. It supports HTTP server API interfaces for Netscape Microsoft Internet Information Server, Lotus Domino Go Webserver, and IBM Internet Connection Server, as well as the CGI and FastCGI interfaces.

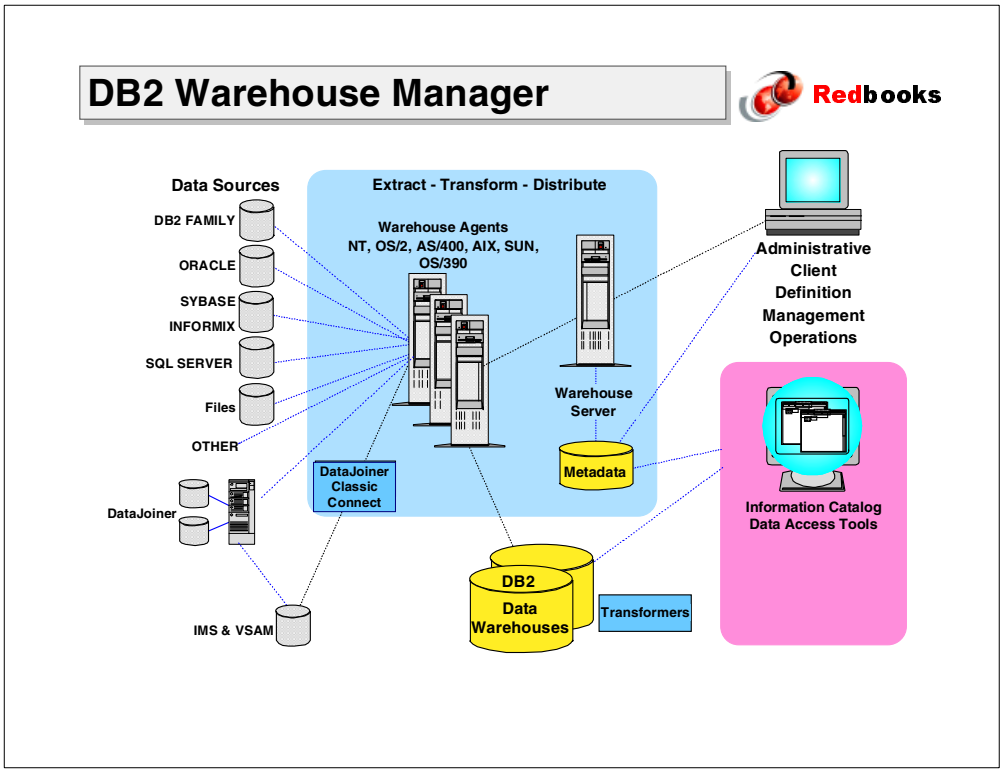
Net.Data is appropriate for customers that are making the transition to e-business using an "enterprise out" approach — thus enabling their existing IT infrastructure to the Web. Net.Data is a good choice for customers wanting to quickly build Web applications accessing data from a variety of data sources and utilizing existing business logic in a variety of programming languages. The development of Net.Data macros (scripts) is quick and easy; it does not require required learning a new programming language, like Java.

Net.Data can access a wide range of data sources. Net.Data provides native access to the data from DB2 on all platforms, as well as Oracle, Sybase, file data, and IMS. Also, Open Database Connectivity (ODBC) gives access to many other relational data sources. And, Net.Data optimizes access to advanced objects in the DB2 family such as DB2 Relational Extenders and DB2 stored procedures.

To develop Web applications, Net.Data provides a simple macro language, including conditional logic and variable substitution. Net.Data allows the customer to reuse existing business logic to create Web applications by supporting calls to Java, C/C++, Perl and REXX applications. Support for additional language environments can be added in a plugable fashion.

With DB2 V7 Net.Data has additional functions:

- Built-in XML exploitation
- Upload from browser (HTTP)
- Memory preallocation for large character data
- Ports to Linux and NUMA-Q.




## 9.3 DB2 Warehouse Manager

Data warehousing is an architecture for organizing information systems which has the purpose of supporting management's decision making process. The data is non-volatile, generally removed from production systems, and provides a single image of business reality for the organization. Briefly stated, you can build a data warehouse system with a set of programs that can:

- Extract data from the operational environment.
- Access a database that maintains data warehouse data.
- Provide data to the user.

The DB2 Warehouse Manager feature is based on proven technologies with new enhancements not available in previous releases. The DB2 Warehouse Manager feature delivers tightly integrated components that enable you to do the following tasks:

- Simplify prototyping, development, and deployment of your warehouse.
- Give control to your data center to govern queries, analyze costs, manage resources, and track usage.
- Help your users find, understand, and access information.
- Give you more flexibility in the tools and techniques you use to build, manage, and access the warehouse.
- Meet the most common reporting needs for enterprises of any size.

IBM's DB2 Warehouse Manager provides the components to make the warehouse usable not just build a warehouse. Visual Warehouse has been enhanced, merged into DB2 UDB V7.1, and integrated with the Control Center. Current VW V5.2 customers will be migrated automatically during the DB2 UDB V7.1 install.

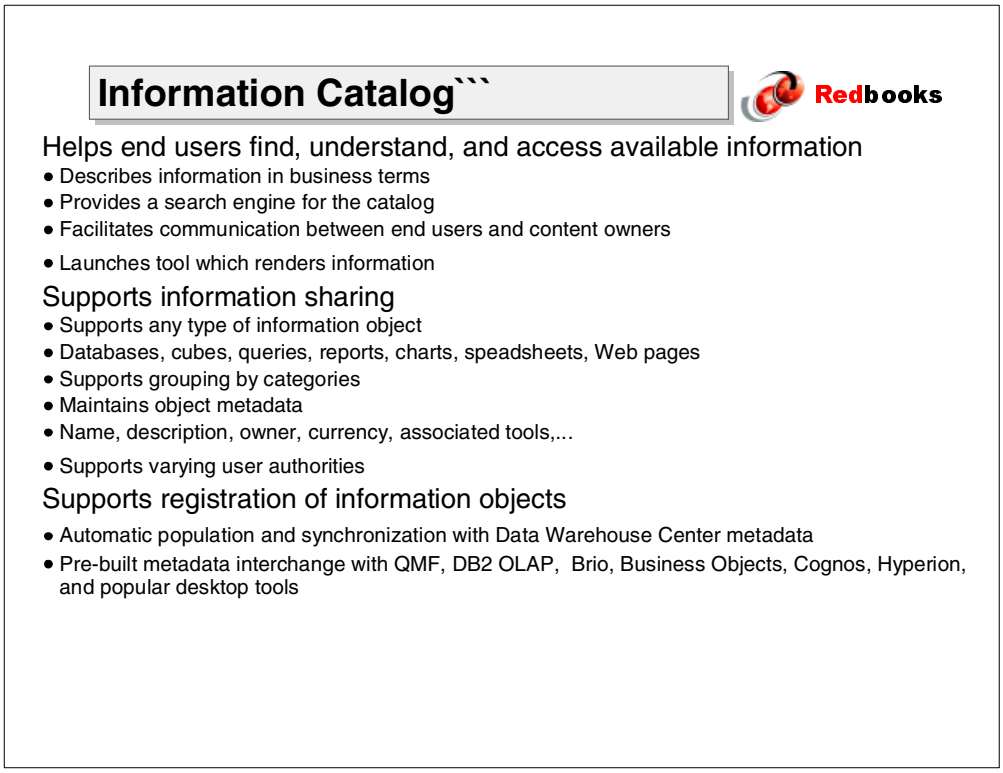
More details on migration are reported in *Migrating to DB2 UDB Version 7.1 in a Visual Warehouse Environment*, SG24-6107. The warehouse administrator GUI has been re-written and included in the base DB2 UDB V7.1 as the Data Warehouse Center. The Data Warehouse Center is accessed from a tools drop down menu in the DB2 Control Center. It consists of:

- An administrative client to define and manage the data and the warehouse operations
- A manager or kernel to manage the flow of data
- Agents residing on platforms to perform requests from the kernel

The possible data sources include:

- Any member of IBM DB2 family, including UDB for AS/400, OS/390, Windows NT, and AIX
- Oracle, Sybase, Informix, and SQL Server databases
- Flat files
- Data Joiner
- IMS and VSAM data through the Classic Connect interface

The data targets are typically DB2 family databases. You can use DB2 Warehouse Manager's process modeler to define a process. A process consists of individual steps and their cascade relationships to one another.




### 9.3.1 Information Catalog

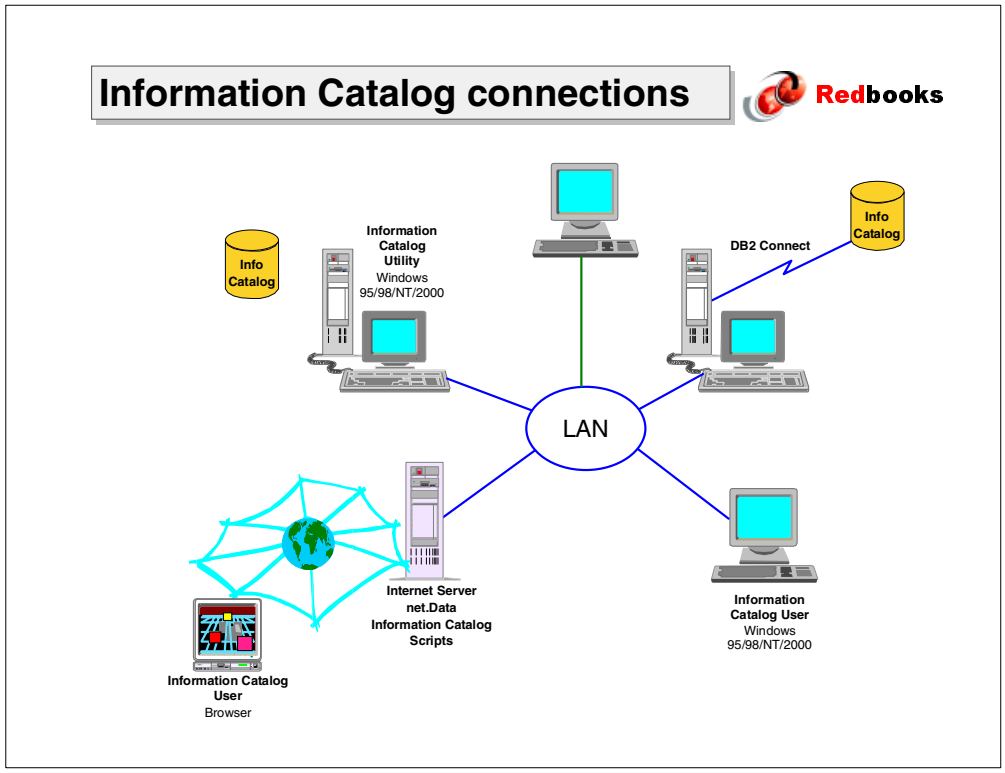
The Information Catalog is a common repository that provides metadata about the objects. It allows users to find, understand, share, and access available information on objects of virtually any nature. It is an evolution of the former Data Guide.

Using a tree structure, objects can be easily organized or grouped in the IC. Extensive search capabilities are offered. Models are provided with the IC, but can easily be extended for additional information.

The IC can be viewed either through a Windows front-end or a Web browser.

Users can find data lineage, meanings of columns, currency of data, contact information, and the next scheduled update of data, as well as obtaining any existing reports.

.




### 9.3.2 Information Catalog connections

Here are some features of the Information Catalog (IC):

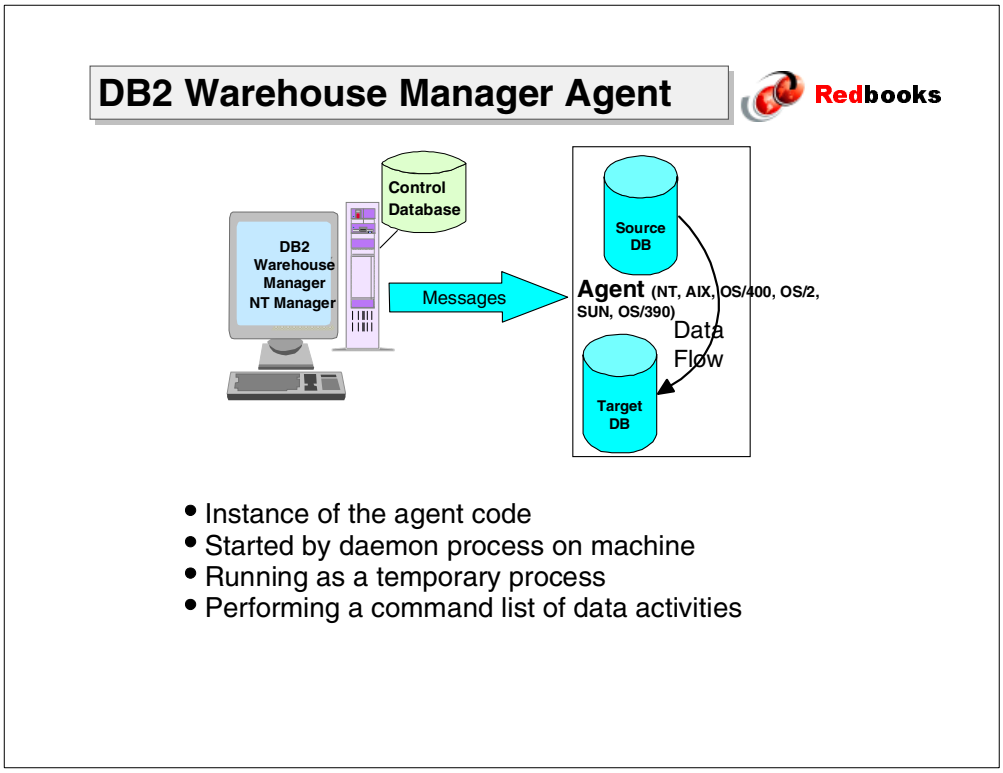
- The IC can be accessed via Windows or a Web browser.
- The IC can be distributed across workgroups or a central repository. Workgroups can be limited to information specific to their workgroup.
- The IC can use any DB2 family platform for its data storage.
- The Data Warehouse Center can automate the population of IC.

All functions of DataGuide are integrated; the Web interface is enhanced; and the DB2 OLAP extractor is enhanced to:

- Show Shared Members only once in the Information Catalog
- Show Aliases, if they exist, for physical names
- Place Calculations, if available, in the derived property.

Predefined program objects are added for QMF for Windows, Wired for OLAP, Seagate, Access, and PowerPoint.

The IC supports bidirectional metadata interchange with Business Objects, Brio, and Cognos. With bidirectional flow, these tools can register existing reports into the IC. Registering the reports allows customers to find tables and column definitions, as well as which reports exist for a table and the ability to launch the report directly from the IC.




### 9.3.3 DB2 Warehouse Manager Agent

An agent is an instance of the agent code which is started by a daemon process on the machine. It runs at the agent site as a temporary process performing data activities and executing the command list received from the kernel.

Key components of the agent are:

- Agent daemon

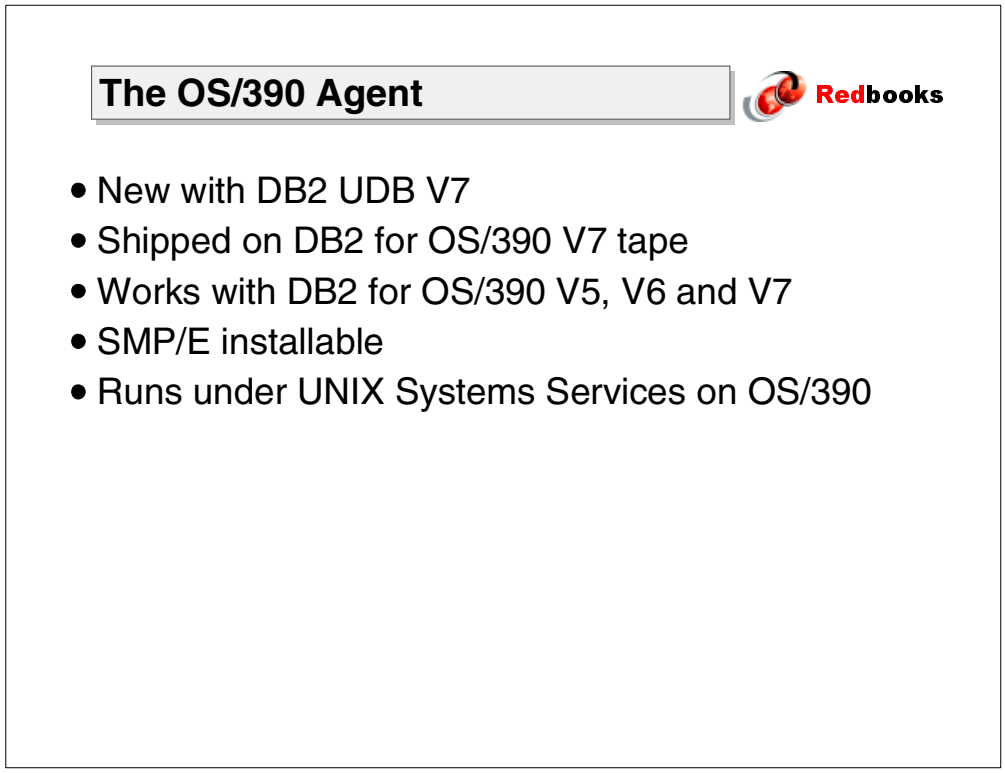
  There is one daemon per host which listens from the common port name Verde and port number 11001 and can be automatically started by the operating system at robot.

- Agent process

  There is one process per command cycle; it uses the DB2 CLI/ODBC interface to communicate with the database.

The default agent is installed wherever the server is installed with ODBC driver manager and drivers; it runs locally to the NT kernel. It does not require an agent daemon and is started directly by the kernel.

The default agent is always present, cannot be deleted, and it is free of charge: the current licensing allows for the default agent plus one other agent.

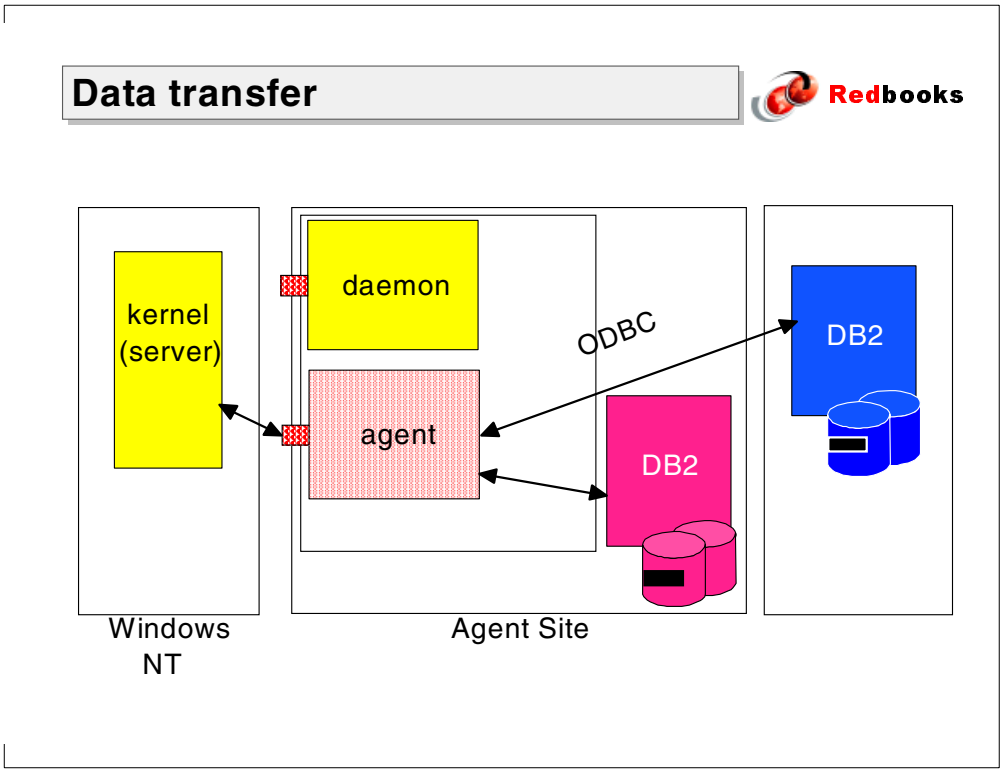



### 9.3.4 The OS/390 Agent

The OS/390 Agent is introduced with DB2 V7. It will be shipped as a feature with the DB2 for OS/390 and z/OS V7 tapes, but it is compatible with DB2 V5 and V6 as well. The OS/390 Agent was ported from the UNIX Agent, and it runs under OS/390 UNIX Systems Services. The OS/390 Agent provides the following functions:

- Copy from source DB2 to target DB2
- Sample contents of a table or file
- Execute user-defined programs
- Access non-DB2 databases through DataJoiner
- Access VSAM or IMS data
- Run DB2 utilities (Load, Reorg, Runstats)
- Replication

Utilities execution and access to VSAM and IMS data are the new functions in regard to what the UNIX Agent supports.




### 9.3.5 Data transfer

One of the main functions of a DB2 Warehouse Manager Agent is to transfer data from DB2 to DB2.

At startup, the OS/390 Agent daemon is listening at port 11,001 for a command from the kernel. When the DB2 Warehouse Manager kernel recognizes that a data transfer needs to occur, it sends a message to the daemon through the messaging component. The messaging component resides partly in the kernel and partly at the Agent site.

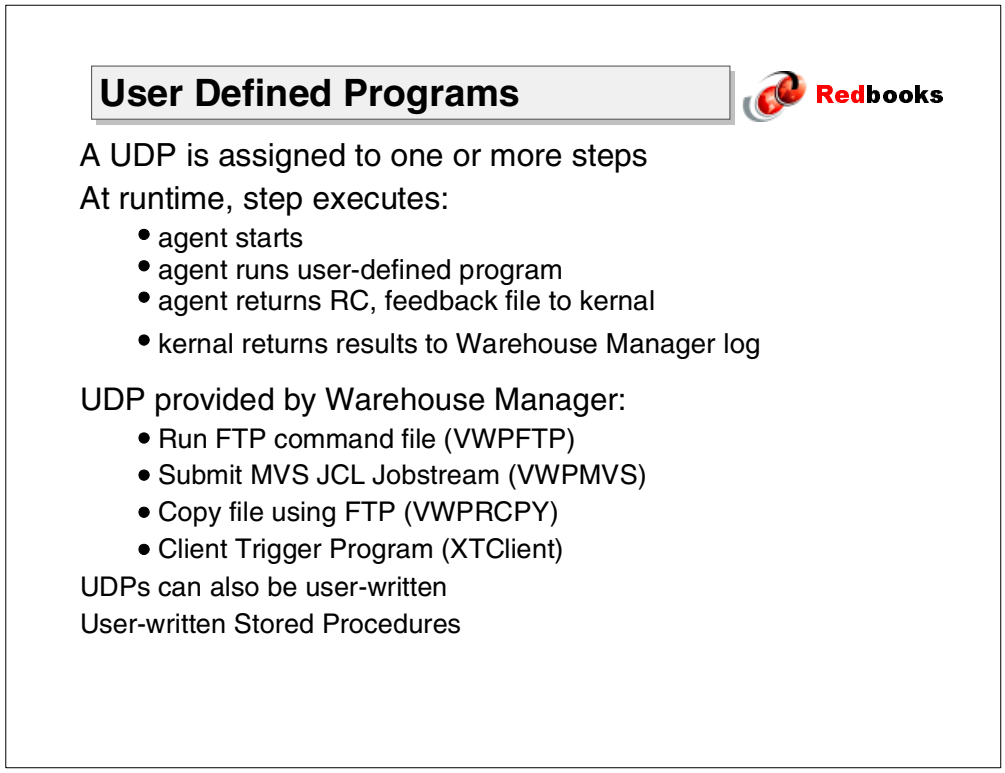
The daemon spawns off an Agent process (or TCB) to handle the request. The Agent gets its own port. The daemon can spawn more than one Agent at a time to handle multiple requests from multiple users.

The Agent tells the kernel its own port number. The kernel then sends the request to the Agent using the messaging component.

When the kernel tells the Agent to connect to the source, the Agent process (or TCB) connects to the source DB2 using an ODBC allocConnect command.

Upon request to connect to the target, the Agent connects to the target. Through use of fetches and inserts, data is transferred from DB2 to DB2. When done, the Agent process ceases to exist.

You can sample contents of flat files or DB2 tables using the OS/390 Agent. For flat files, the Agent takes a guess at the file format based on parameters in the properties of the file definition. It works on IMS or VSAM files via Classic Connect, UNIX Systems Services flat files, and OS/390 native flat files.



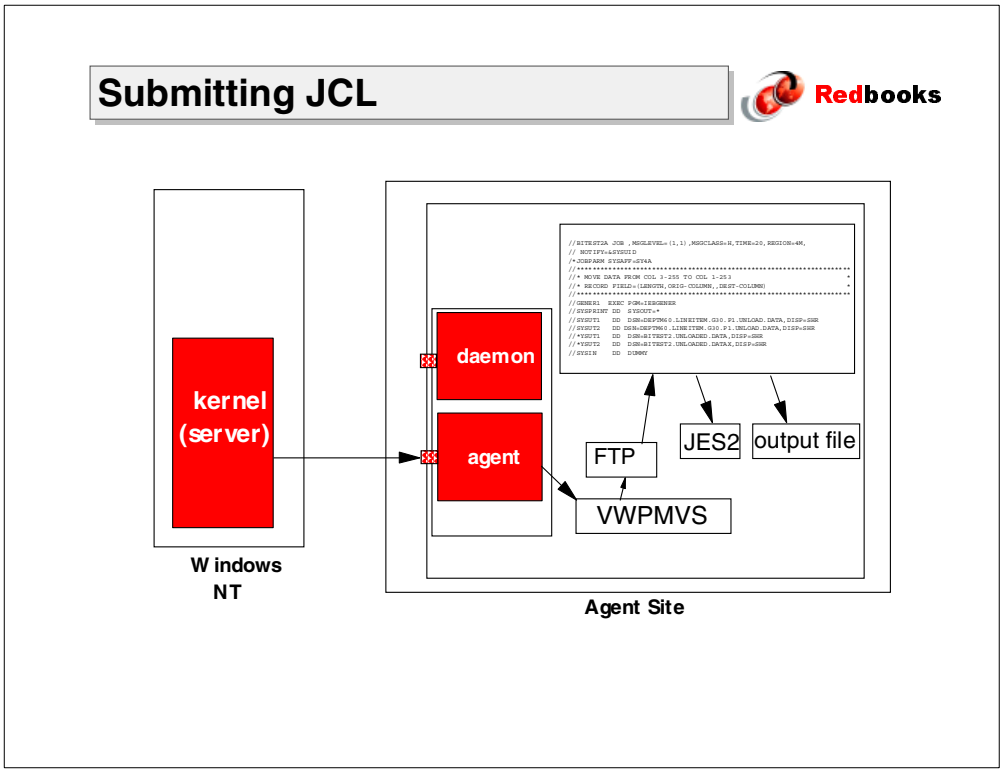

### 9.3.6 User defined programs

In a DB2 Warehouse Manager step which needs to do more than a simple data transfer, the customer can have Warehouse Manager execute a pre-defined program. Examples of possible uses for UDPs are batch utility jobs or jobs which apply updates to the data.A User-defined program is assigned to one or more steps

At runtime:

- Step executes
- Agent starts
- Agent runs user-defined program
- Agent returns RC, feedback file to agent
- Agent returns results to kernel

DB2 Warehouse Manager provides four user-defined programs for the customer's use. In addition, customers can define their own programs to data warehouse center. The OS/390 Agent supports any executables that run under UNIX Systems Services.

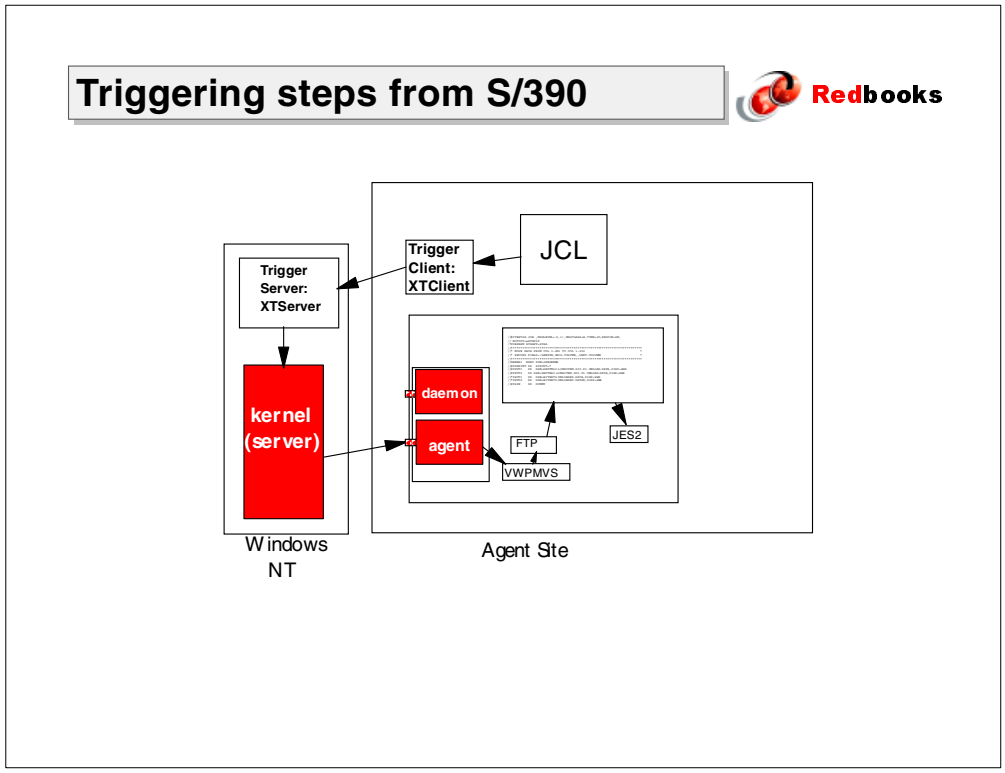



### 9.3.7 Submitting JCL

This diagrams how the Agent submits JCL using VWPMVS. The kernel sends a "run UDP" message to the Agent and passes with it the name of the OS/390 data set which contains the job to run. The Agent calls VWPMVS, which uses the SITE FILETYPE=JES command of FTP. FTP sends the job to the OS/390 JES reader and copies the output to an output file specified in DWM.

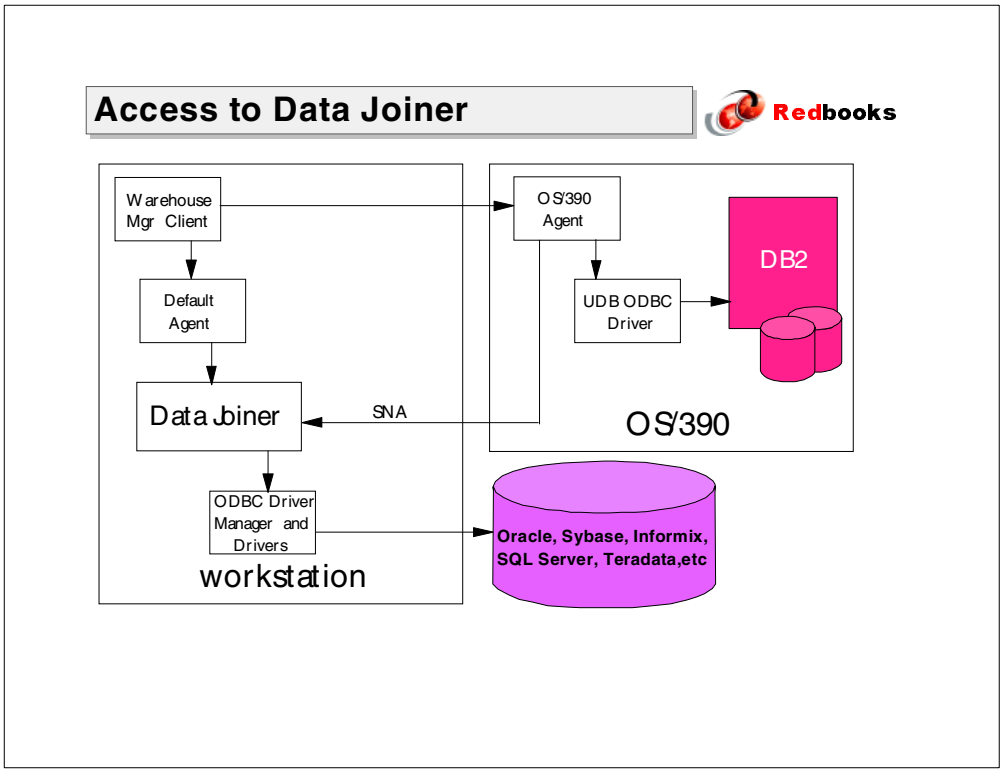
**Note:** Restriction on VWPMVS:

- Job name must be your userid plus one character
- Job must be routed to held output class
- VWS_LOGGING environment variable must be set
- Should have a .netrc file.




### 9.3.8 Triggering steps from OS/390

The trigger program solves the problem of customers who do not wish to have NT scheduling steps on an OS/390 platform. An OS/390 job scheduler or a customer can submit a job which triggers a step on DB2 Warehouse Manager. If the step is successful, the trigger step in the JCL returns a 0 return code. Although unlikely that customers would want to do this, the trigger program could even submit JCL using VWPMVS.

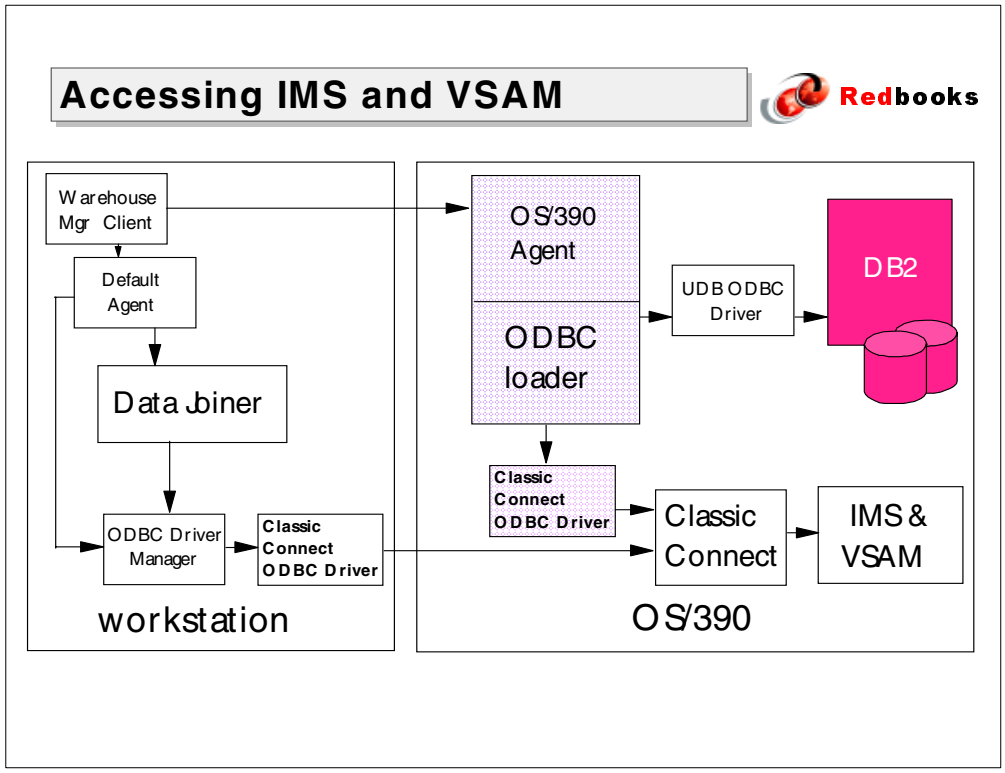



### 9.3.9 Accessing Data Joiner

To access non-UDB family databases, the OS/390 Agent uses Data Joiner, like the other Agents. Data Joiner has an interface that lets the Agent use a normal DRDA flow as if it were a UDB database. If an ODBC request is directed to a non-UDB source, Data Joiner invokes an additional layer of code to access foreign databases. For example, when accessing Microsoft SQL server, Data Joiner passes the request to the Windows ODBC driver manager, which then sends the request to the SQL server.

Data Joiner can access Oracle, Sybase, Informix, Microsoft SQL Server, Teradata, and anything else which has an ODBC driver which runs on NT, AIX or Sun. It can also access IMS and VSAM through Classic Connect, a Cross Access product separately installed.

Note that the OS/390 Agent can access DataJoiner as a source, but not as a target, since Data Joiner does not support 2-phase commit. Another restriction is that Data Joiner does not support TCP/IP connections to it from OS/390, so you must use a SNA connection to access it. TCP/IP connections can be used in a 2-hop configuration, but this may not be practical for some users.

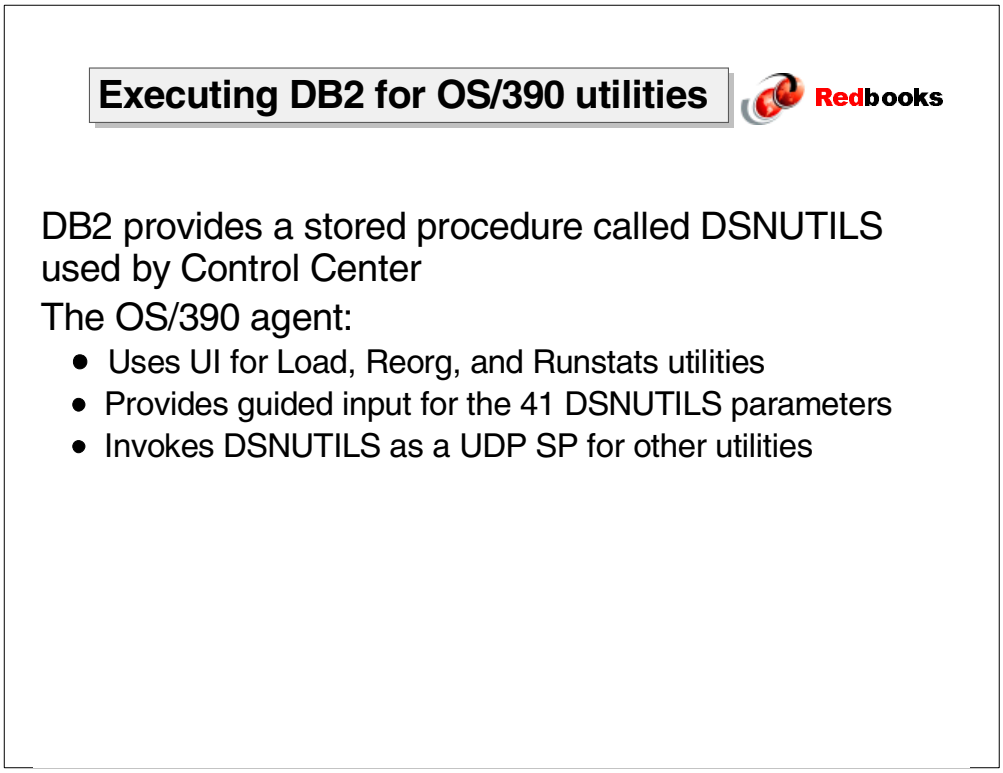



### 9.3.10 Accessing IMS and VSAM

The Windows and NT Agent access IMS and VSAM through the Cross Access Classic Connect product. Classic Connect allows customers to set up a DB2-like definition of IMS and VSAM data sets and then to access them through ODBC.

The Windows NT Agent goes to the Classic Connect ODBC driver on NT, and from there to Classic Connect on OS/390.

The OS/390 Agent could take a similar route. However, IMS and VSAM are already on OS/390. There is no ODBC driver manager that runs on OS/390, and the Classic Connect ODBC driver cannot be used for DB2 Universal Database access and vice versa. So the OS/390 Agent has an additional function which loads the correct ODBC driver based on whether a request is directed to Classic Connect or DB2. If it is a DB2 source, it loads the DB2 ODBC DLL; if it is a VSAM or IMS source, it loads the Classic Connect ODBC driver. It then processes the Agent's request.



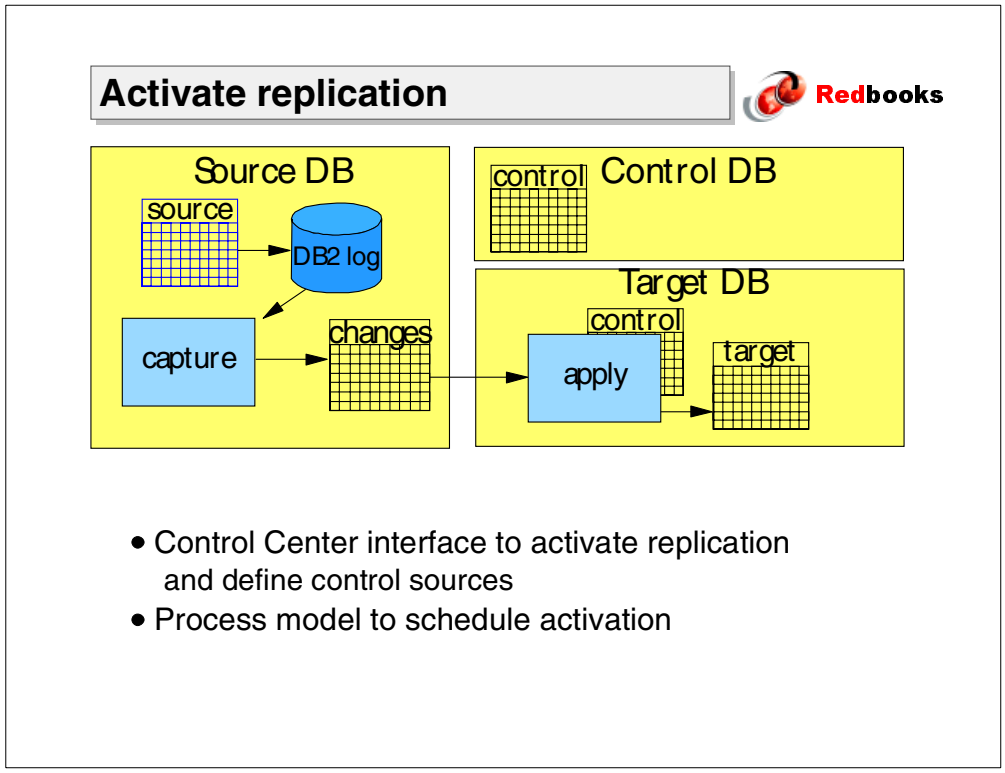

### 9.3.11 Executing DB2 utilities

DSNUTILS is a stored procedure that executes in DB2 for OS/390 in a WLM and RRS environment. Warehouse Manager provides an interface to DSNUTILs to allow inclusion of DB2 utilities in Warehouse Manager steps.

Since DSNUTILS is a stored procedure, you can use it to run any DB2 utilities that you have installed by using the user-defined stored procedure interface. There are also special interfaces to execute Load, Reorg, and Runstats.

To set up DSNUTILS:

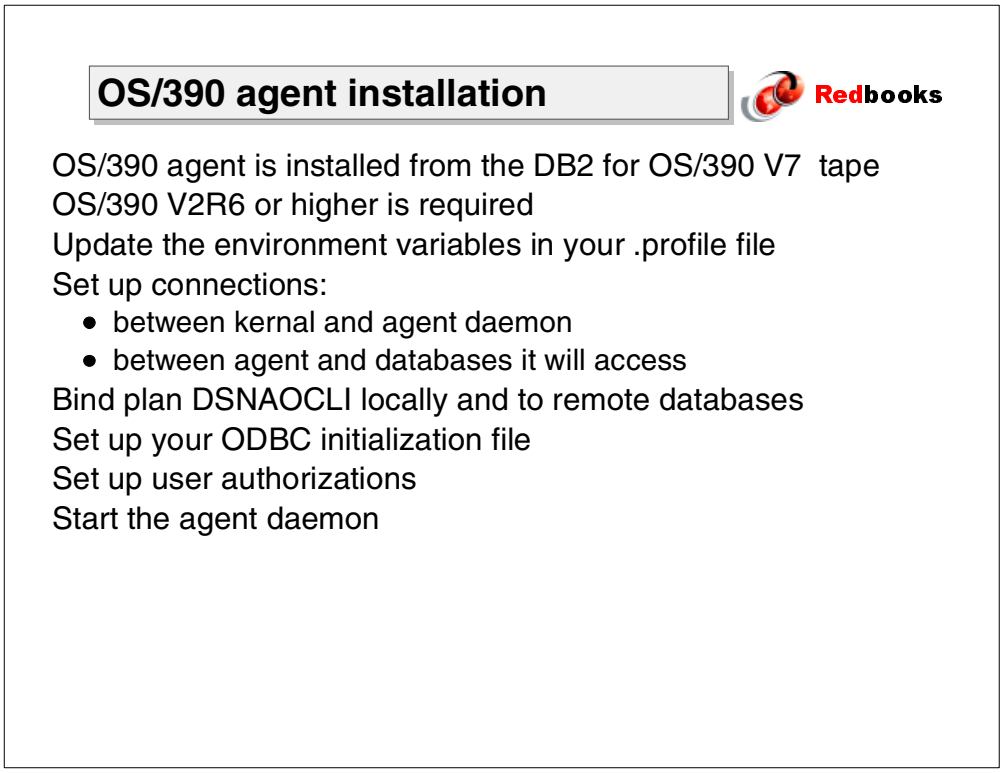
- Execute job DSNTIJSG when installing DB2 to define and bind DSNUTILS. Make sure the definition of DSNUTILS has parameter style general with nulls and linkage = N.
- Enable WLM-managed stored procedures.
- Set up your RRS and WLM environments.
- Run the sample batch DSNUTILS programs (not required, but recommended).
- Bind the DSNUTILS plan with your DSNCLI plan so that CLI can call the stored procedure: BIND PLAN(DSNAOCLI) PKLIST(*.DSNAOCLI.*, *.DSNUTILS.*).
- Set up a step using the Warehouse Manager UI and execute it. The population type should be APPEND; otherwise, Warehouse Manager will delete everything from the table before executing the utility.




### 9.3.12 Activate replication

You can use the OS/390 Agent to automate your DataPropagator apply steps. Replication involves a Source database, a control database, and a target database, all of which may be different or the same databases. A capture job reads the DB2 log to determine which of the rows in the source database have been added, updated, or changed, and it writes the changes out to a changed-data table. An Apply job is then run to apply the changes to a target database.

You can use Warehouse Manager to automate the execution of the apply job by creating a replication step. The Warehouse Manager allows you to define the type of apply to run and when to run it by customizing a JCL template.




### 9.3.13 OS/390 Agent installation

Environment variables for the Agent are listed in the standard documentation. They point the Agent to DB2 libraries, output directories, and so on. This is the contents of a sample .profile file that belongs to the home directory of the user who starts the Agent daemon:

```
export VWS_LOGGING=/u/VWSWIN/logs
export VWP_LOG=/u/VWSWIN/vwp.log
export DSNAOINI=/u/VWSWIN/dsnaoini.STPLEX4A_DWC6
export LIBPATH=$LIBPATH:/u/VWSWIN/
export PATH=/u/VWSWIN/:$PATH
export STEPLIB=DWC6.SDSNEXIT:DSN610.SDSNLOAD
```

To set up the kernel and daemon connection, add the following to your /etc/services or tcpip.etc.services file:

```
vwkernel 11000/tcp
vwd 11001/tcp
vwlogger 11002/tcp
```

To set up connections between the OS/390 Agent and databases, add any remote databases to your OS/390 communications database. Some sample CDB inserts:

```
INSERT INTO SYSIBM.LOCATIONS  (LOCATION, LINKNAME, PORT) VALUES
('NTDB','VWNT704','60002');
INSERT INTO SYSIBM.IPNAMES  (LINKNAME, SECURITY_OUT, USERNAMES, IPADDR)
  VALUES ('VWNT704', 'P', 'O', 'VWNT704.STL.IBM.COM');
INSERT INTO SYSIBM.USERNAMES  (TYPE, AUTHID, LINKNAME, NEWAUTHID, PASSWORD)
  VALUES ('O', 'MVSUID', 'VWNT704', 'NTUID', 'NTPW');
```

For more information, see the chapter on Connecting Distributed Database Systems in the *DB2 UDB for OS/390 Installation Guide*, GC26-9008-00.

Because the Agent uses CLI to communicate with DB2, you must bind your CLI plan to all remote databases your Agent plans to access. Some sample bind statements are:

```
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLICS) ISO(CS)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLICS) ISO(CS)
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLINC) ISO(NC)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLINC) ISO(NC)
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLIRR) ISO(RR)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIRR) ISO(RR)
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLIRS) ISO(RS)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIRS) ISO(RS)
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLIUR) ISO(UR)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIUR) ISO(UR)
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLIMS)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIC1)
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLIC1)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIC2)
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLIC2)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIQR)
BIND PACKAGE (DWC6CLI) MEMBER(DSNCLIF4)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIF4)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIV1)
BIND PACKAGE (NTDB.DWC6CLI) MEMBER(DSNCLIV2)
BIND PLAN(DWC6CLI) PKLIST(*.DWC6CLI.* )
```

For more information, see the *DB2 UDB for OS/390 ODBC Guide and Reference*, SC26-9005-00.

To execute the Agent daemon, you must be the owner of the following:

```
libtls0d.dll (dll)
iwhcomnt.x (side deckl)
vwd (executable)
```

To become an owner of libtls0d.dll, iwhcomnt.x, and vwd, you must run three `extattr` commands in the VWSWIN directory:

```
extattr +p vwd at the command line.
extattr +p iwhcomnt.dll at the command line.
extattr +p libtls0d.dll at the command line.
```

To run these commands, you must have permission to access the BPX.FILEATTR.PROGCTL facility class.

After you finish configuring your system for the OS/390 warehouse Agent, you need to start the warehouse Agent daemon, as follows:

- Telnet to USS on OS/390 through the OS/390 hostname and USS port.
- Navigate to the /DWC directory.
- Enter `vwd` on the command line.

To verify from a UNIX shell that the Warehouse Agent daemon is running:

- Enter `ps -e | grep vwd` on a UNIX shell command line.

  or

- Enter - `/D OMVS,a=all` on the OS/390 console

The Classic Connect nonrelational data mapper is a Microsoft Windows-based application that automates many of the tasks required to create logical table definitions for non relational data structures. The objective is to view a single file or portion of a file as one or more relational tables. The mapping must be accomplished while maintaining the structural integrity of the underlying database or file. This is purchased and installed separately from the Warehouse Agent.

To set up Warehouse access:

- Create a CrossAccess configuration file. A sample file can be found on /VWSWIN/cxa.ini. Update the DATASOURCE line. This line contains a data source name and a protocol address. The data source name must correspond to a Query Processor name defined on the CrossAccess Data Server, which is located in the QUERY PROCESSOR SERVICE INFO ENTRY in the data server's config file. The protocol address can be found in the same file on the TCP/IP SERVICE INFO ENTRY.

The USERID and USERPASSWORD in this file will be use when defining a Warehouse data source.

```
**************************************************************************
*  Cross Access  Sample Application Configuration File          *
**************************************************************************/
* national language for messages
NL = US English
* resource master file
NL CAT = /VWSWIN/v4r1m00/msg/engcat
FETCH BUFFER SIZE = 32000
DEFLOC = CXASAMP
USERID = uid
USERPASSWORD = pwd
DATASOURCE = DJX4DWC tcp/9.112.46.200/1035
MESSAGE POOL SIZE = 1000000
```

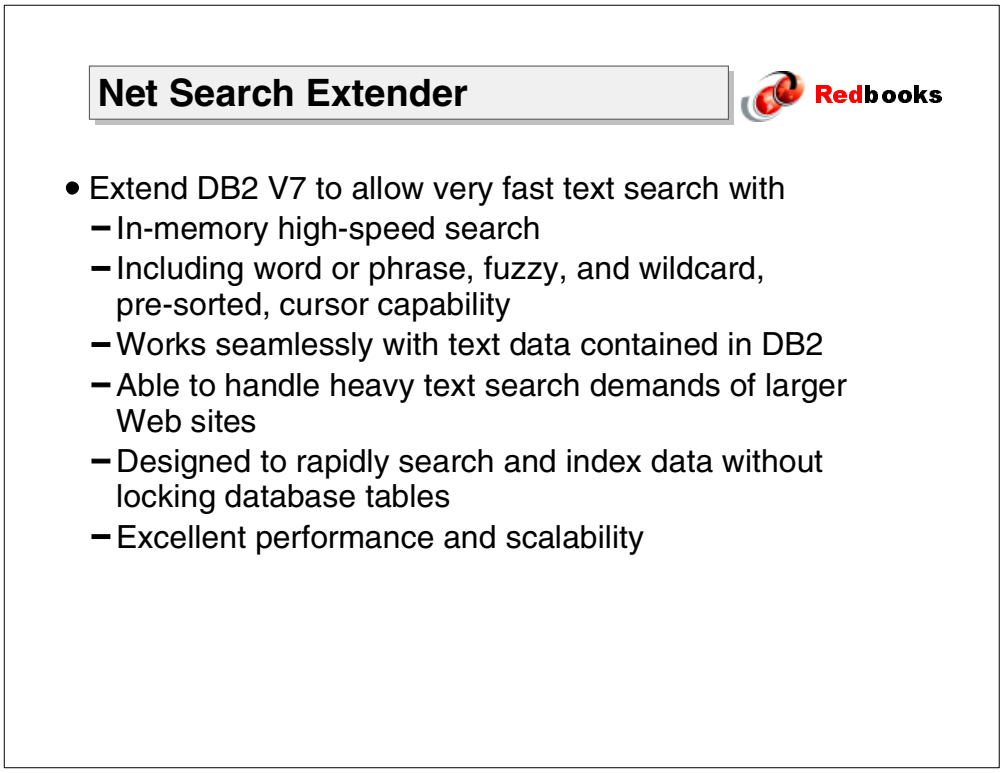
- You do not need to update your dsnaoini file, because DB2 for OS/390 does not have a driver manager. The driver manager for CrossAccess is built into the OS/390 Agent.
- Update your profile to export the CXA_CONFIG environment variable export XA_CONFIG=/VWSWIN/cxa.ini
- Update your LIBPATH environment variable to include /VWSWIN
- Verify the install with the test program cxasamp. From directory /VWSWIN enter cxasamp. The location/uid/pwd is the data source name/userid/userpassword defined in cxa.ini file.
- Now you can define a DWC Warehouse source as you would for any other data source.
- To use the OS/390 utilities you must specify the utilities under DB2 Programs -> DB2 for OS/390.

## 9.4 QMF

IBM Query Management Facility (QMF) is a tightly integrated, powerful, and reliable query and reporting tool set for IBM's DB2 relational database management system, and provides an environment that is easy for a novice to use but powerful enough for an application programmer. QMF offers extensive management and control over your enterprise query environment to protect valuable system resources.

With DB2 V7, QMF for OS/390 introduces enhancements in the following areas:

- DB2 access and connectivity — Distributed access to the entire DB2 Family of server products is now available, with the addition of support for:
  - DB2 for AS/400 server, Version 4.4
  - DB2 for VSE DRDA Remote Unit of Work Application Requester
- DB2 integration — DB2 features are now easily exploited, with the addition of:
  - Fully integrated support for the ROWID data type
  - Limited support for the LOB data types
  - A Date and time edit code that changes format to match the current database manager default
  - Cross platform DRDA package binding
- Usability — QMF ease of use is enhanced, with the introduction of:
  - More command defaults; working with QMF objects on the screen is easier with these commands:
    - Run, Save, Print, Edit, Export, Reset, Convert
    - Added flexibility for command options that accept quoted strings
    - Direct navigation to the QMF Home panel
    - Online help upgrades to stay informed and productive




## 9.5 DB2 Net Search Extender

DB2 Net Search Extender is very similar to DB2 Text Extender, but provides a higher search speed by offering one very fast index type and no use of triggers to dynamically update an index. It contains a stored procedure that adds the power of fast full-text retrieval to Net.Data, Java, or DB2 CLI applications. It offers application programmers a variety of search functions, such as fuzzy search, stemming, Boolean operators, and section search. Searching using DB2 Net Search Extender can be particularly advantageous in the Internet, when search performance on large indexes and scalability according to concurrent queries are important factors. DB2 Net Search Extender is similar to DB2 Text Extender, but provides a higher search speed. It runs on AIX, Solaris, Windows 2000 and Windows NT, and OS/390.

### 9.5.1 Key features

These are DB2 Net Search Extenders key features:

- Indexing
  - Allows multiple indexes on the same text column
  - Indexing proceeds without locking data
  - Creating indexes can be done on multiple processors
- Search results
  - Lets you specify how the search results are sorted
  - Lets you set a limit on the size of the search results
  - Allows positioning (cursor-setting) access on search results

- Search operations
  - Supports Boolean or wild card operations
  - Allows word, phrase, stemmed, or fuzzy search
  - Provides tag or section support, with or without Boolean operations

***Examples***

- Boolean or wild card

  Use wild card (masking) characters to find words that begin with night AND'd with dreams, dreamy, and so on:

  "night%" & "dream_"

- IN SAME SENTENCE AS

  A keyword that lets you search for a combination of terms occurring in the same sentence.

  Find two words in the same sentence (this assumes that the sentences are separated by periods):

  "computer" in same sentence as "book"

- STEMMED FORM OF

  A keyword that causes the word (or each word in the phrase) following STEMMED FORM OF to be reduced to its word stem before the search is carried out. This form of search is not case-sensitive.

  Search for inflectional endings of the word shock, such as shocked, shocking, and so on: stemmed form of "shock"

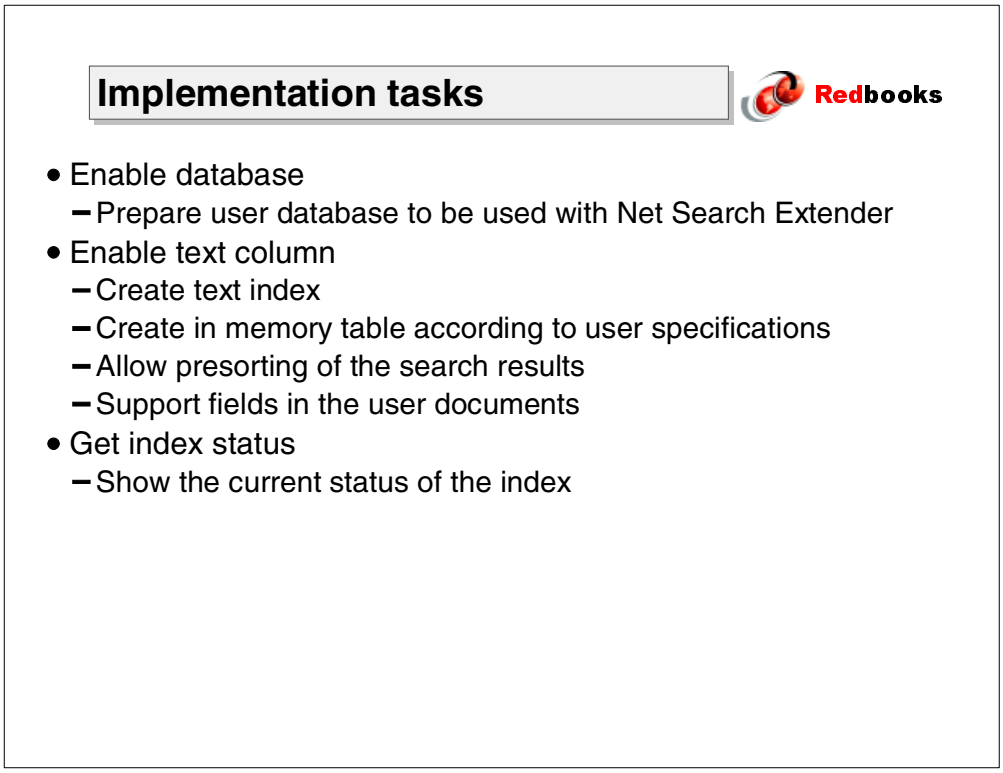
- FUZZY FORM OF match-level

  A keyword for making a "fuzzy" search, which is a search for terms that have a similar spelling to the search term. This is particularly useful when searching in documents that were created by an Optical Character Recognition (OCR) program. Such documents often include misspelled words. For example, the word "economy" could be recognized by an OCR program if spelled as "econony". The match-level is a value from 1 to 100, where 100 means least fuzzy (exact) and 1 means most fuzzy.

  Search for documents containing emergency and contain security department with 60% or more matching.

  "emergency" & fuzzy form of 60 "security department"

Searching using Net Search Extender can be particularly advantageous in the Internet when search performance on large indexes and scalability with concurrent queries are an important factors. Only read access to the user data is required




### 9.5.2 Implementation tasks

We list here the tasks for administration activities. Most tasks require input from the application programmers who develop the database applications that search for and retrieve data.

***DB2 actions***

- Design and create a database (table spaces and tables); load data
- Identify a table, text column, primary key, text tags/fields, optimize on columns, and all parameters related to creating a specific index

***Net Search Extender actions***

- Enable a database
- Enable a text column to create an index
- Update the index when the data changes in the database's text column
- Activate and deactivate an index
- Disable a text column to delete an index when an index is no longer needed
- Disable a database when all indexes are no longer needed

***To prepare a database for Net Search Extender***

1. Identify the text data and related columns
   - Once you have designed and created a database and loaded it with text data, identify the following information to prepare the database for Net Search Extender high-performance text search:
     - The table name and column name where specific text data resides. The column is of type CHAR, VARCHAR, or LONG VARCHAR.
     - The primary key of the table, and an index name of your choice.
   - Optional information includes:
     - Specific tags, when the text is a document that uses tags to distinguish text fields
     - Optimized columns whose data is to be loaded in memory
     - Order-by columns, if the search results are to be sorted
     - Location and directory for the created index if you do not want to store the index in the default directory
2. Verify and set the environment variables
3. Issue the command to enable the database
4. Issue the command to enable the text column - create a Net Search Extender index

To maintain the Net Search environment, you must update an index whenever the contents of its associated text column changes. Net Search Extender does not do this automatically; you must run a command to accomplish this task. When you update an index, the in-memory table associated with that index is recreated. This means that you cannot search the index while it is being updated.

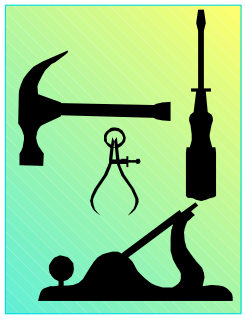
An index must be activated before you can search on it. This can be done automatically when you create the index. However, you may need to activate an index explicitly if the index has been deactivated explicitly by the deactivate index command, or when the system has been rebooted.

When an index is activated, considerable amounts of data may be loaded into memory. To prevent the system from running out of resources, you should explicitly deactivate indexes that are not being used.

# Chapter 10. DB2 tools

## IMS and DB2 tools

- New focus on database tools to complement DB2 for OS/390 and IMS
- New and repositioned products
- Four areas of interest on database
  - Administration
  - Performance management
  - Recovery and replication management\
  - Application management
- Web site:
  `ibm.com/software/data/db2imstools`



For OS/390 customers with IMS or DB2, the cost of software tools has grown to become a major factor in IT budgets. IBM is responding to these needs and has made a major development investment to provide IBM products that offer a viable alternative to third party products. The products are priced to meet your ever-changing environment, and are designed to perform at the level that today's systems require. The Data Management utilities and tools address the most common database requirements of DB2 and IMS users.

IBM has therefore announced a new focus on Database Tools to complement DB2 for OS/390 and IMS database engines. IBM seeks to be your preferred supplier of tools by providing competitive, high-performance, and functionally rich products.

IBM's strategy is to focus on data management tools by:

- Defining areas of importance to our customers
- Consolidating existing product offerings
- Providing a consistent set of Terms and Conditions
- Standardizing on industry consistent price methodology
- Being responsive to customer needs

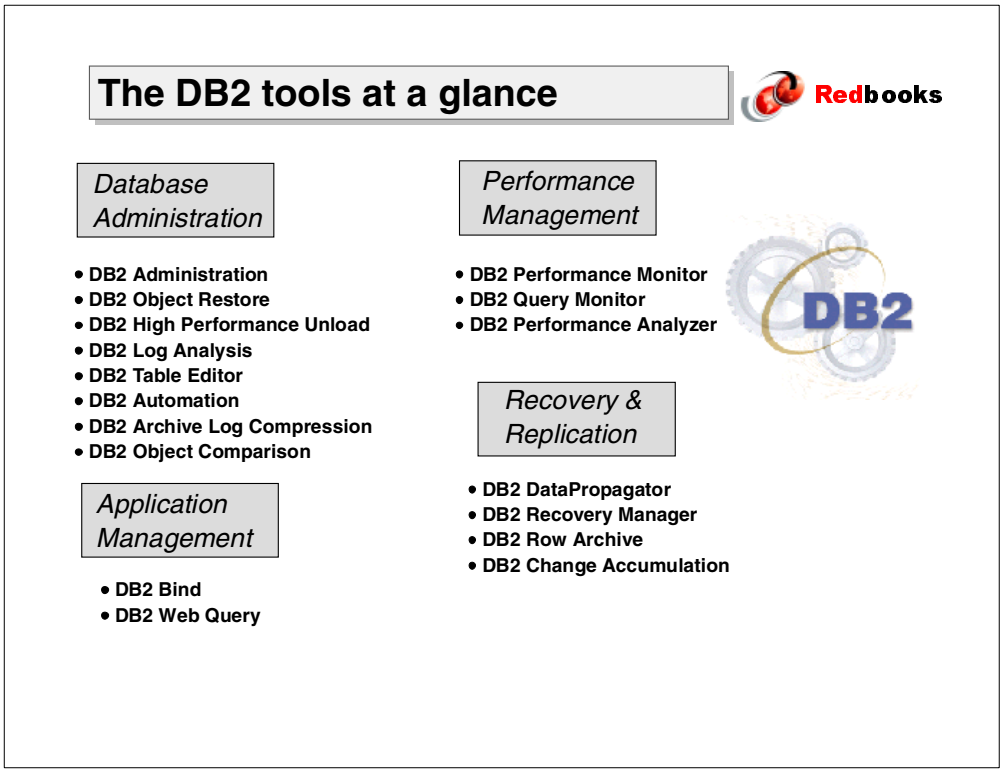
© Copyright IBM Corp. 2001

For OS/390 customers with IMS or DB2, the cost of software tools has grown to become a major factor in IT budgets. IBM is responding to these needs and is delivering new and repositioned products for DB2 and IMS. This is a very dynamic area, for current information on the available products you must check the recent announcements and the Web site:

http://ibm.com/software/data/db2imstools/

***Tools for DB2 UDB for OS/390 and z/OS***
IBM is now offering several tools for DB2 UDB Server for OS/390 and z/OS. Now DB2 V7 can be combined with many optional tools and priced features that make your databases even easier to administer, access, and manage.

***Tools for IMS***
IBM offers a selection of more than 30 tools. The IMS Support Tools is a set of database performance enhancements for your IMS environment which extend the functions already available in your IMS toolset. IMS is the most reliable transaction and database management system used by 90% of the world's major corporations.




## 10.1 DB2 tools at a glance

In this section we briefly describe the new DB2 tools and mention the recent enhancements for the tools already available with DB2 V6. This is a very dynamic area of products where enhancements and new tools are being announced frequently. Please refer to the official announcements for current information.

The tools can operate with DB2 V5, V6, and V7, under OS/390 or z/OS, and are grouped into four categories as follows:

- **Database administration tools**

  They help you maximize the availability of your systems and address the most common tasks required to service and support database operations. These include such tasks as unloading, reloading, reorganizing, copying, and catalog management. Many of these are operations where performance is critical in meeting your company's availability commitments. The tools in this area are:

  - DB2 Administration
  - DB2 Object Restore
  - DB2 High Performance Unload
  - DB2 Log Analysis Tool
  - DB2 Table Editor
  - DB2 Automation
  - DB2 Archive Log Compression
  - DB2 Object Comparison

- **Performance management tools**

  While the administration tools cover most of your maintenance and service operations, performance management tools are equally important in keeping your database environment operating at peak performance. The tools in this area are:

  - DB2 Performance Monitor
  - DB2 Query Monitor
  - DB2 Performance Analyzer

- **Recovery and replication management tools**

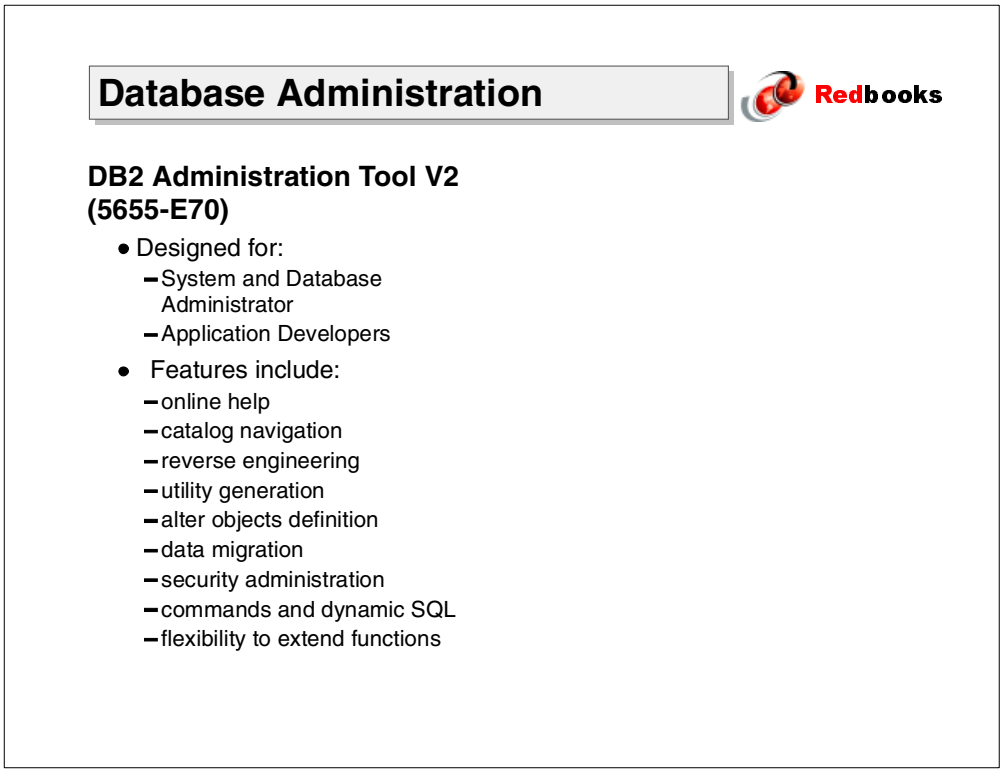
  Normal backup and recovery operations are handled by the image copy and recovery tools you choose. Many customers, however, have additional requirements or situations requiring special solutions. These might include: archiving data, point in time recovery, online recovery, and added image copy capabilities. In addition, asynchronous replication of data to where it is needed can enable your distributed applications. The tools in this area are:

  - DB2 DataPropagator
  - DB2 Recovery Manager
  - DB2 Row Archive

- **Application management tools**

  Tools helpful in the development, testing, operation, and connectivity of database applications, by providing the ability to build simple reporting applications, provide e-business connections, examine and change data, and control examine and change data, and control checkpoint activities. The tools in this area are:

  - DB2 Bind
  - DB2 Web Query




## 10.2 IBM DB2 Administration Tool

DB2 Administration Tool Version 2.1, program number 5655-E70, is the new release of DB2 Administration Tool, providing a comprehensive set of database management functions to help DB2 systems personnel manage your DB2 environment efficiently and effectively. It saves time in DB2 administration, simplifies routine day-to-day DB2 tasks, and increases knowledge and understanding of your DB2 system.

DB2 Administration has these features:

- Helps you derive the maximum performance and results from your DB2 system, dramatically reducing the effort required to support DB2.
- Provides a comprehensive set of functions that help DB2 systems personnel efficiently and effectively manage their DB2 environments.
- Offers complete DB2 catalog query and object management.
- Runs under Interactive System Productivity Facility (ISPF) and uses dynamic SQL to access DB2 catalog tables.

These are the main functions of DB2 Administration:

- Provides catalog visibility and navigation.
- Lets you explore unknown databases, get a quick overview of a database, and discover database problems quickly.
- Executes dynamic SQL statements.
- Issues DB2 commands against database and table spaces.

- Simplifies creation of DB2 utility jobs and run most DB2 utilities.
- Allows you to copy tables from one DB2 to another.
- Allows complex performance and space queries.
- Performs the EXPLAIN function.
- Performs various system administration functions, such as managing DDF, and updating limits.
- Lets you extend existing DB2 Administration applications or rapidly develop new applications using ISPF and DB2 interface.
- Permits reverse engineering:
  - Recreating DDL for objects in the DB2 catalog.
  - Generating DDL for underlying indexes, views, synonyms and aliases.
  - Generating authorization statements for the objects.
  - Adjusting space allocations.
  - Changing the name of the owner.
  - Changing the database name.

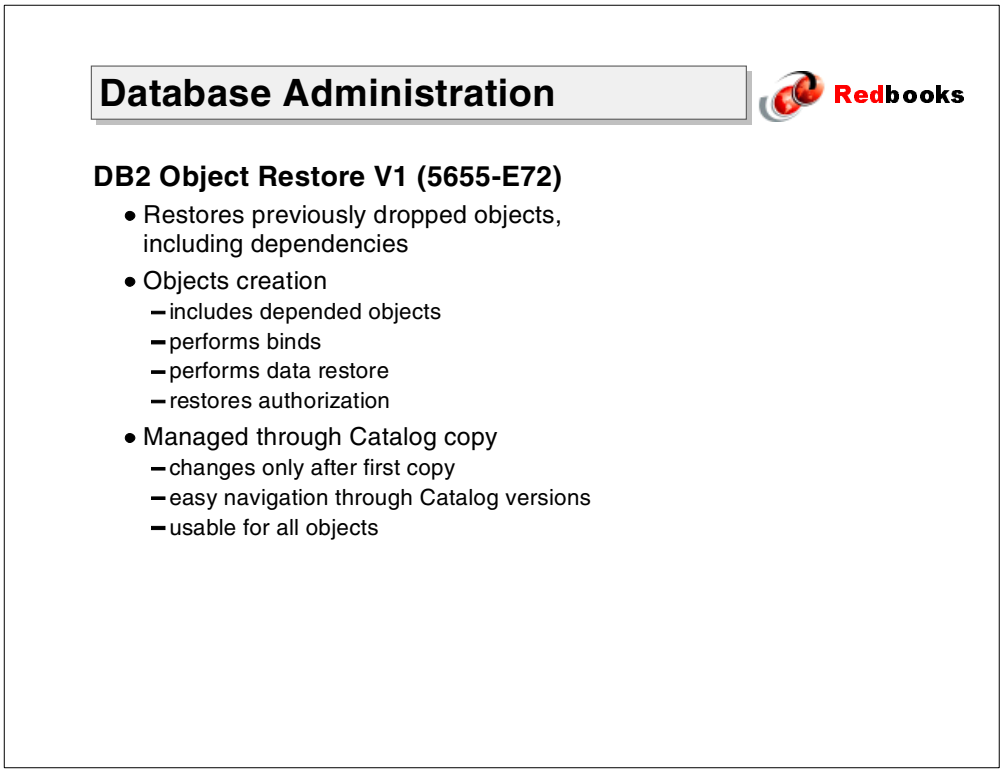
Version 2 of DB2 Administration has two new functions:

- *Alter* supports modification of tables and their attributes.
- *Migrate* provides facilities to copy data and objects to other DB subsystems.

Version 2 of DB2 Administration has two major enhancements:

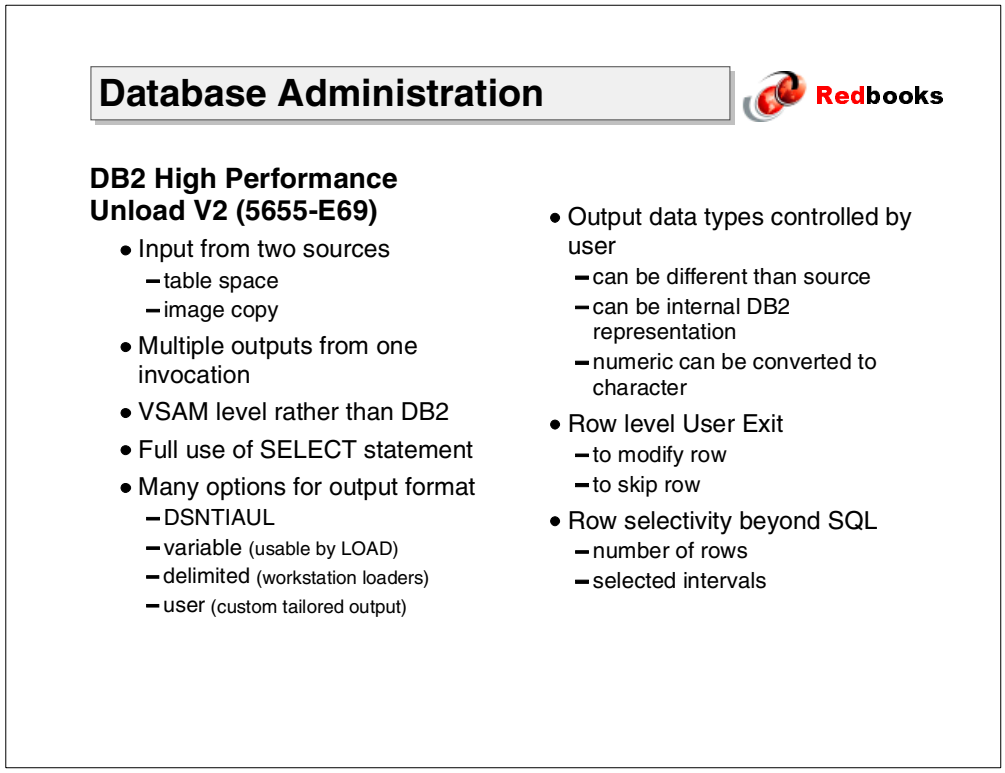
- Sort and search capabilities are improved.
- Installation-defined line commands are supported.

Recent maintenance has introduced further enhancements in the areas of restartability for Alter and Migrate, space management functions, and the possibility of saving and restoring parameters. DB2 Admin is also providing the option to launch other DB2 tools that have an ISPF interface.




## 10.3 IBM DB2 Object Restore

IBM DB2 Object Restore, program number 5655-E72, can automatically restore previously dropped objects and all related dependencies. DB2 Object Restore is superior to other tools that are available in the marketplace since it eliminates the need for a duplicate shadow copy of the catalog to recover objects. This means that DB2 Object Restore saves disk space. With DB2 Object Restore you can have confidence in cleaning up your DB2 system since you can now restore discarded DB2 objects.




## 10.4 IBM DB2 High Performance Unload

DB2 High Performance Unload, program number 5655-E69, can really make a difference to your unload management tasks. DB2 High Performance Unload offers sequential reading and accessing of your DB2 data at top speed. It can scan a table space and create output files in the format that you need. The major capabilities provided are:

- High speed unloads of DB2 data using native VSAM
- Ability to unload from image copies as well as active tables
- Multiple output files during a single unload with minimal additional cost
- Multiple output formats, including the opportunity to tailor your own
- Comprehensive and powerful SELECT statement
- More efficient unloads of DB2 data
- Help with batch windows constraints
- Ability to unload multiple tables each to a separate file or a single table many times

Sequential reading of DB2 tables requires long periods of time. This makes it difficult to schedule unloads of large tables in the ever-shrinking batch windows of DB2 installations. Large scan performance can become critical when several unloads have to read the same table space concurrently.

The performance symptoms can sometimes be caused by multiple reads of the same DB2 pages. Sometimes there are even channel conflicts.

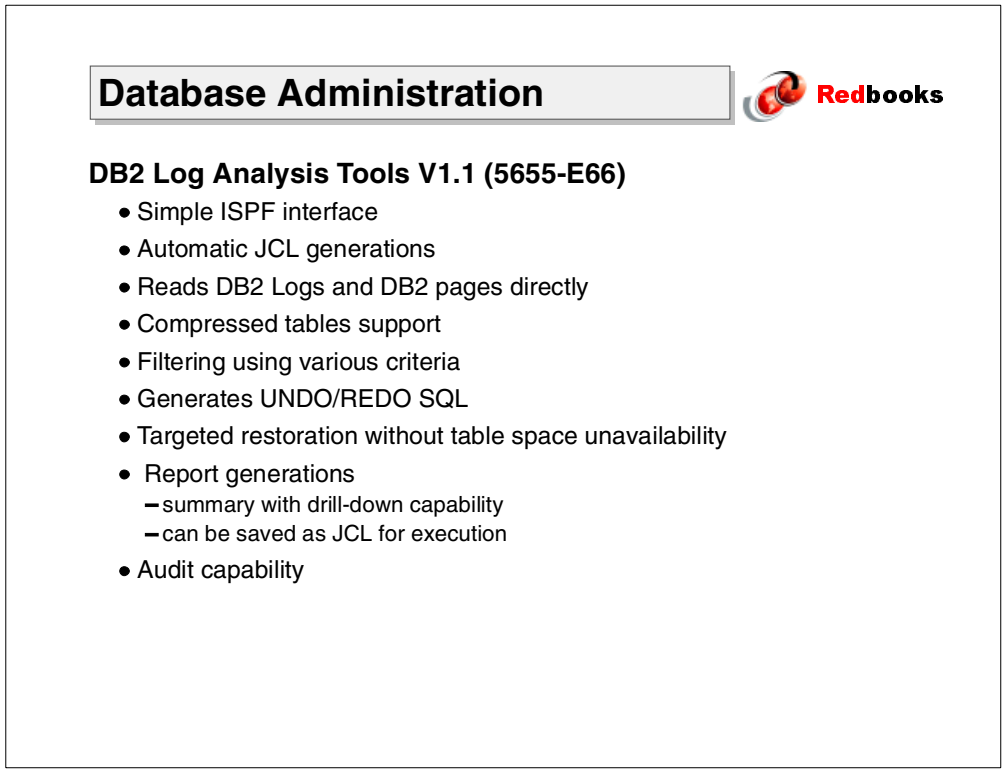
DB2 High Performance Unload can also make a difference to your unload management tasks.

DB2 High Performance Unload offers sequential reading and accessing of your DB2 data at top speed. It can scan a table space and create output files in the format that you need. All you have to do is select the criteria. Do you want the format to be DSNTIAUL compatible? Or do you need standard variable length records? Or is your choice a delimited file for export to another platform?

You can elect almost any type of conversion, giving your output the appearance that you want. You can code as many select statements as you want for any tables belonging to the same table space, so different output files can be created during the same unload process at almost no additional cost.

You can also unload multiple tables at once, each to its own file, as long as all the tables belong to the same table space. You can even unload the same table many times with different select statements, using only one table space scan.

Do not be concerned that all this activity will affect DB2 production. You can run your DB2 High Performance Unload against image copies (incremental or any full image copy) as well as active tables, so DB2 production databases are unaffected.

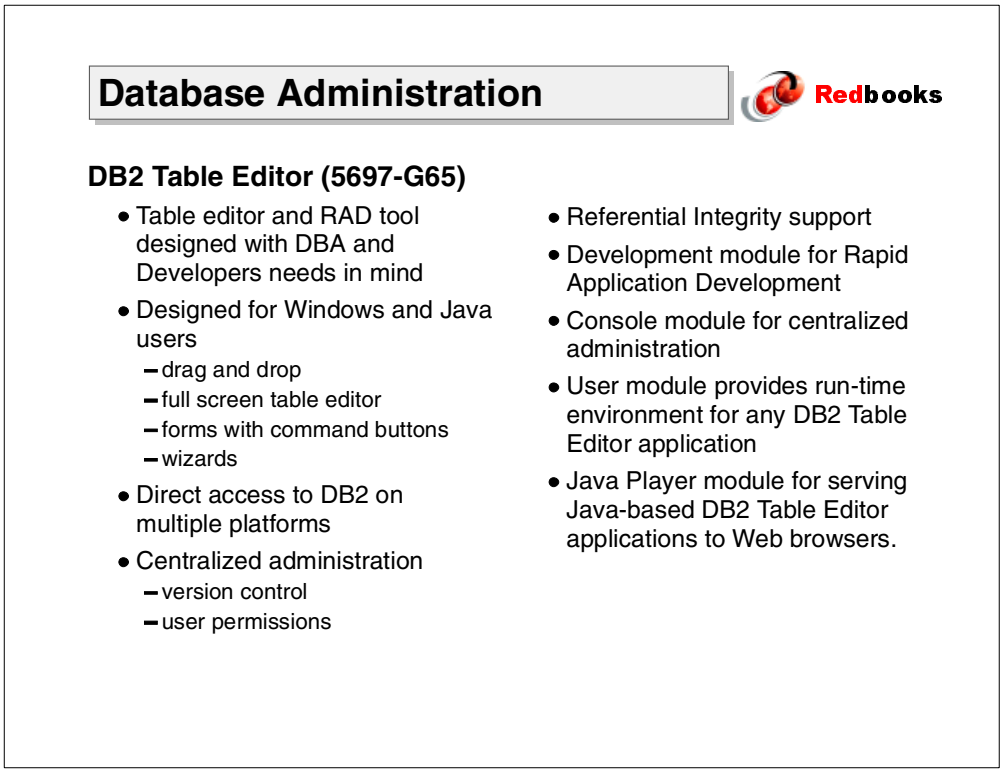



## 10.5 IBM DB2 Log Analysis Tool

IBM DB2 Log Analysis Tool, program number 5655_E66, provides the DB2 administrator with a powerful tool to ensure high availability and complete control over data integrity. It allows you to monitor data changes at a glance by providing the facilities to:

- Automatically build reports of changes made to database tables.
- Specify reports by various database resource criteria, such as date, user, or table.
- Quickly isolate accidental or undesired changes to your database tables.
- Avoid the processing overhead often associated with data change monitoring.
- Heighten confidence in data integrity while keeping DB2 for OS/390 systems at optimal efficiency.
- Maximize uptime for e-business availability.

Recent enhancements have added data sharing support, new summary reports, and additional filtering options.




## 10.6 IBM DB2 Table Editor

DB2 Table Editor, program number 5697-G65, is the new generation of DB2 Forms, program number 5697-G52. It is IBM's multipurpose table editing environment that offers database administrators, developers, and the entire enterprise, direct update and data creation operations on DB2 UDB for OS/390 and z/OS databases from within Java, ISPF or Windows-based interfaces. With DB2 Table Editor you can create Windows front-end applications.

DB2 Table Editor enables developers to quickly construct new applications, often in just minutes with a drag-and-drop interface. Add buttons, labels, text boxes and controls for containing data and drop-down lists. Behaviors, data sources and data validation rules are assigned to controls from easy-to-use dialogs that require no programming. Applications can be centrally stored for user access at the database server, and periodically updated and improved at will. It then provides users with access to finished applications.

DB2 Table Editor applications are stored centrally at the DB2 server and then launched from any Windows workstation. End users select applications from the catalog of custom forms. Each application presents specific data associated with it at development time. Typical applications include table editing, inventory or product catalog access, order entry, customer invoice retrieval, or query-by-example front ends. Access is available to local or remote locations, including users connecting from any location via the Internet to TCP/IP supported DB2 databases.

Administration functions enable administrators to set up user groups, permissions for those groups, and time-of-day and day-of-week schedules. Applications can be bound to their respective databases; user groups and permissions for users and groups can be configured. Governing settings are stored centrally at the database server and allow administrators to make entire applications (or just specific capabilities across all applications) unavailable to selected groups.

These are the main functions of DB2 Table Editor:

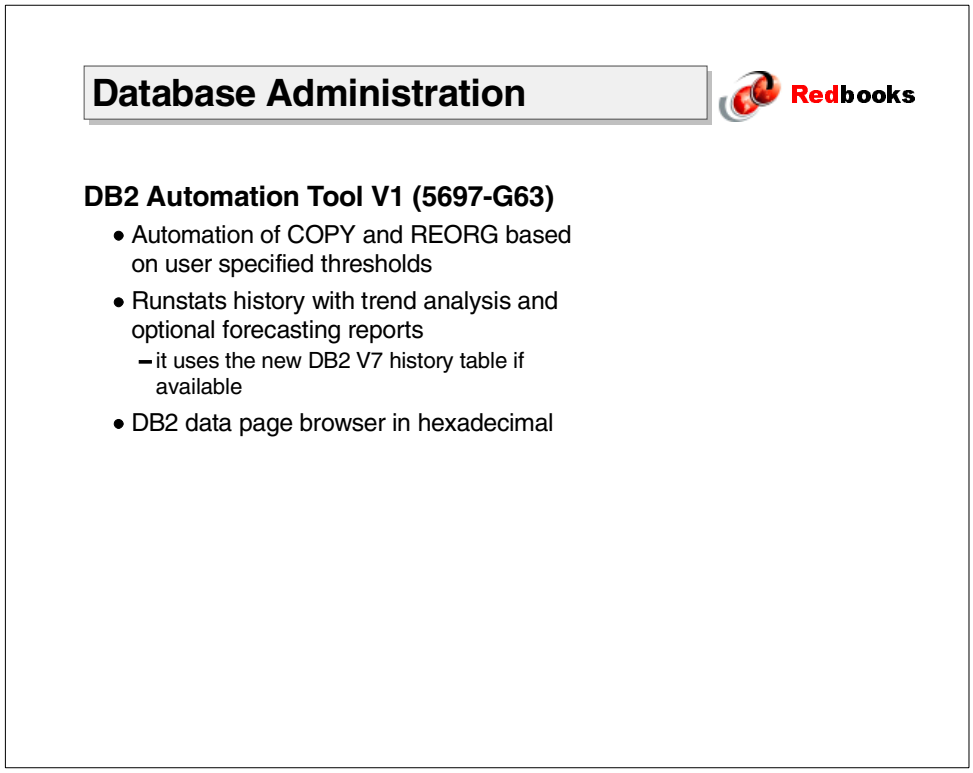
- Build applications and graphical user interfaces to any DB2 data warehouse or DB2 operational data.
- Create controls, data validation rules, and application behaviors within a drag-and-drop environment.
- Build in advanced database techniques and commands without programming or SQL knowledge.
- Satisfy universal requirements, such as transactions, table editing, QBE, and data entry.
- Slash development time while creating applications that set new standards for performance.
- Distribute finished applications freely like a browser to in-house or remote users.
- Set up users in minutes without database gateways, middleware, or ODBC drivers.
- Applications can connect directly to databases over the Internet (including, via dial-up to a local ISP).
- Restrict user/application permissions and track user activity at the server with centralized governing.
- Roll out with TCP/IP or SNA connectivity and full DB2 security support.
- Use IBM's DB2 Data Joiner to include multi-vendor data sources, such as IMS, VSAM, Oracle, Informix, Sybase, Microsoft SQL Server, and more.

Whether using table layouts in the full screen table editor, wizards, or forms with command buttons rapidly built in the DB2 Table Editor drag-and-drop development environment, both Windows and Java-based users now can have direct access to multiple DB2 database tables on multiple platforms, including OS/390, VSE and VM, and Windows workstation databases.

DB2 Table Editor includes enhanced data editing and referential integrity capabilities, full screen table editing interface, and new form components. It continues to provide a robust table editing and database *front end* building environment that offers:

- Reading and writing directly to IBM DB2 UDB database tables, through a choice of connectivity options
- Transparent cross-platform support for multiple IBM DB2 UDB database platforms, versions, and native DB2 security
- Rapid building of business and data validation rules into table editing forms, without programming or compiling

- Centralized management and administration, providing excellent concurrency and control over user permissions, database access, and versioning of table editing forms
- Server-based licensing for distribution to unlimited users
- The most rapid path in the industry to providing controlled, direct data operations upon DB2 tables from within Windows, Java-based workstations, or Web browsers.

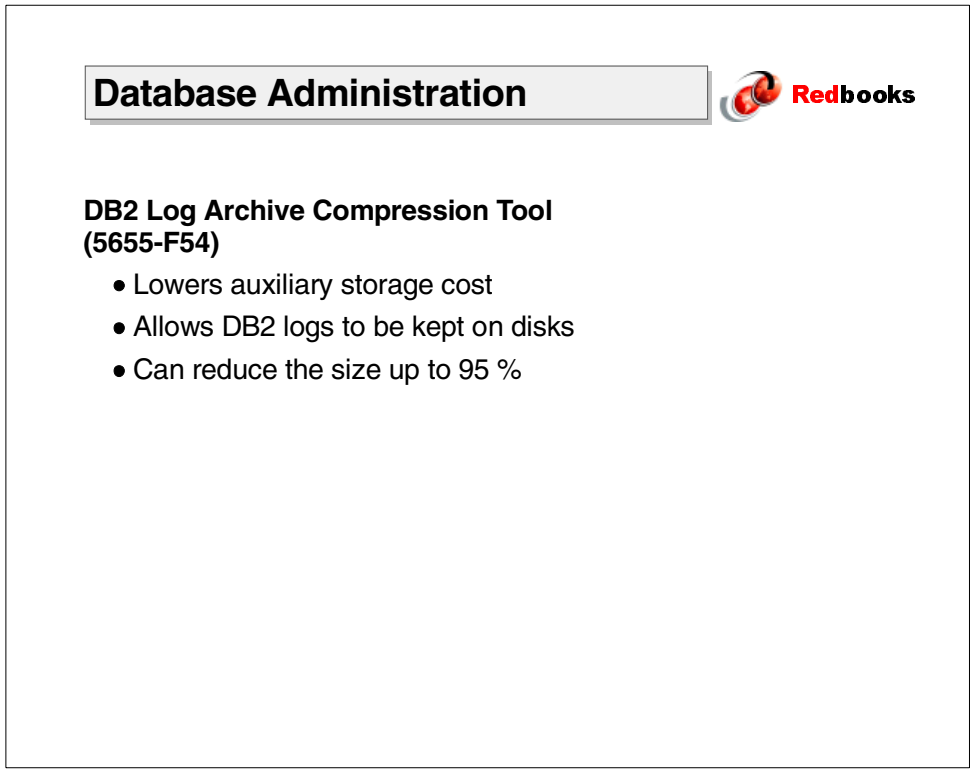



## 10.7 IBM DB2 Automation Tool

DB2 Automation Tool for S/390 and z/OS, program number 5697-G63, helps you realize the full potential of your DB2 system by continuously and automatically coordinating the execution of DB2 tools on an around-the-clock basis. The primary capabilities provided are:

- Automatic execution of DB2 tools against specified objects
  - Image Copy
  - REORG
  - RUNSTATS
- Manual, periodic, or rules-based execution of any number of tools, with a combination of parameters and options
- Easy creation of job specifications as *profiles* that may be Updated, Deleted, and Imported or Exported across subsystems and test or operational environments
- Development of job profiles through intuitive ISPF panels and special syntax without needing JCL skills
- Support that allows multiple database administrators to securely maintain their own sets of profiles

Now, without relying on repeated manual interventions, every DB2 administrator is able to add maximum value to the enterprise by extracting full performance constantly from even the most heavily-used database environment.




## 10.8 IBM DB2 Archive Log Compression Tool

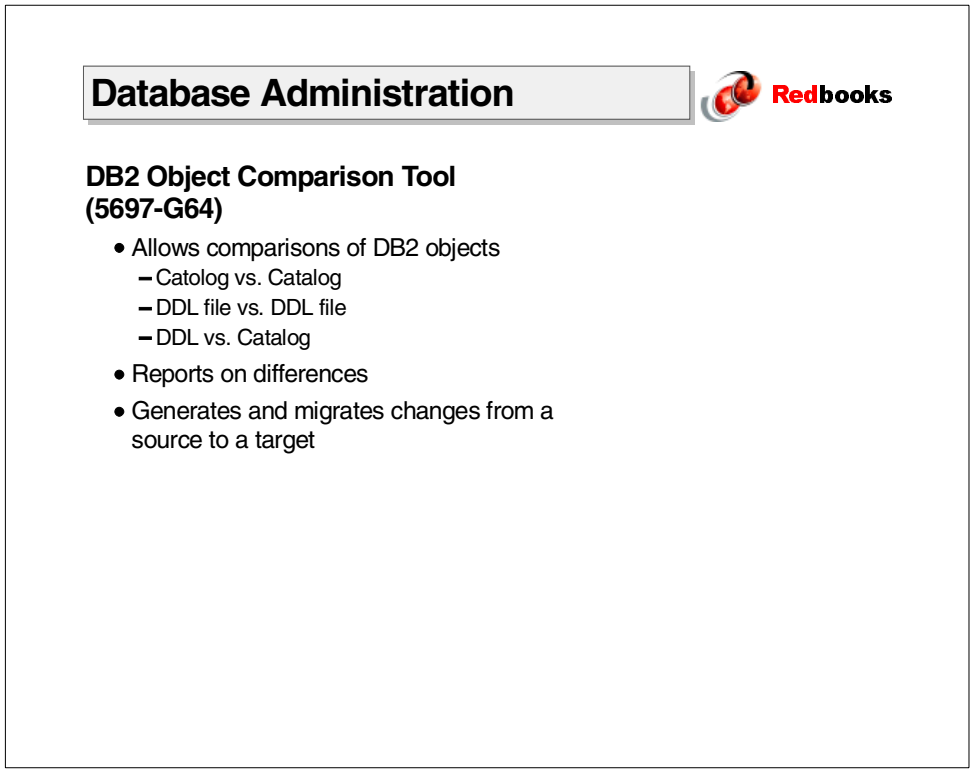
DB2 Archive Log Compression Tool for S/390 and z/OS, program number 5655-F54, makes highly compressed off-site copies of DB2 logs allowing administrators to reduce the volume of archived logs resulting in:

- Shorter I/O and recovery times
- Lower storage costs
- Making storage of DB2 logs on DASD affordable in many cases instead of tape

Archive Log Compression uses high performance compression technology designed especially for DB2 logs to maximize the:

- Reduction of your offload and retrieval time
- Value of your storage media

Administrators may elect to keep SQL UNDO entries in the log to be compressed and achieve compressions that may exceed 40 percent. Or they may choose to have DB2 Archive Log Compression remove SQL UNDO (which is not needed for disaster recovery) in order to obtain even higher rates of compression. The tool provides disaster recovery support by restoring directly from the compressed logs.




## 10.9 IBM DB2 Object Comparison Tool

DB2 Object Comparison for OS/390 and z/OS, program number 5697-G64, helps you keep your test and development system as a mirror image of the production system. New applications, application modifications, or mistakes can cause DB2 objects in one of these systems to have different attributes than on other systems. DB2 Object Comparison Tool allows you to compare objects, and dependent objects, from one source to another. Once a difference file is generated, the product can be used to generate the DB2 commands needed to bring the catalogs back into synchronization.

Masking and ignore files are supported to account for intentional differences and/or naming conventions that exist between the two sets of objects to be compared. For example, primary and secondary quantities usually are different between a test and production system. Likewise, the same object might have an owner name of TESTxxx on the test system and an owner name of PRODxxx on the production system. Use of the mask and ignore files allow you to compare only on real differences that might exist.

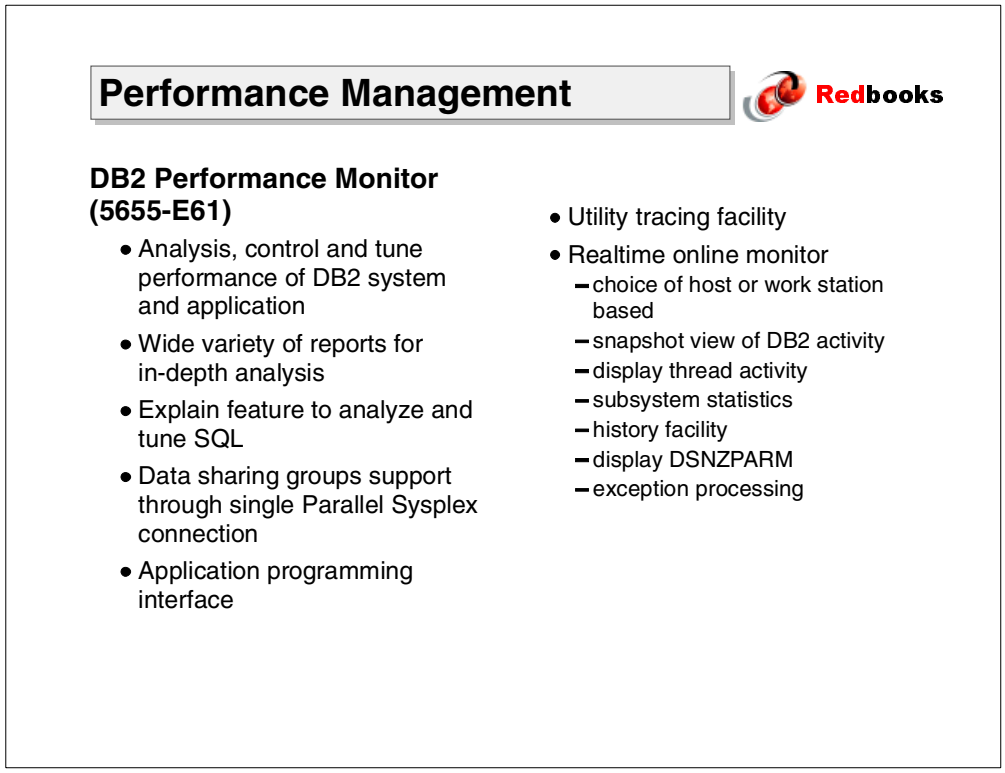
*Version files* are used for comparison. A version file is created from either a file of DDL or from objects in a DB2 catalog. The *known-to-be-correct* source version file is compared to a target version file that may be back level. Using version files as a base you may compare DDL, catalog object definitions, and other version files in any pair-wise combination desired.

The ability to do comparisons with previously generated version files gives you the opportunity to:

- Restore application objects to a previous version (backout)
- Compare a new version with several production versions (clones) of the objects

DB2 Object Comparison runs as an extension to DB2 Administration Tool Version 2.1 and consists of:

- An ISPF frontend for specification of the objects to be compared
- A DB2 catalog extract function that pulls definitions from the catalog into a version file to support the compare process A DDL extract function that reads DDL statements and converts them into a version file
- A batch compare function that compares two version files, produces a report that describes the differences found and generates the information needed to apply changes to the target
- A batch job generator that provides everything necessary to apply the changes to the target




## 10.10 IBM DB2 Performance Monitor

DB2 Performance Monitor for OS/390 Version 7, program number 5655-E61, is IBM's strategic tool for analyzing, controlling, and tuning the performance of DB2 for OS/390 systems as well as DB2 applications. DB2 PM Version 7 supports full performance monitoring and problem analysis for all functions of DB2, including the new DB2 enhancements introduced in V7. For example, DB2 PM supports all instrumentation, catalog, and PLAN_TABLE changes.

DB2 PM can be used in two main ways: batch editing and online monitoring. Online monitoring allows you to obtain a *snapsho*t view of DB2 activities, while the history facility allows you to view events both recently and in a more distant past. With the workstation-based monitor, which is replacing the traditional ISPF interface, you can monitor all your DB2 subsystems in parallel. It has interfaces to other IBM DB2 tools, such as Visual Explain for explaining SQL statements, and it can be launched by other tools such as the DB2 Control Center.

You can use the wide variety of reports already provided or you can customize them for even more in-depth performance analysis.

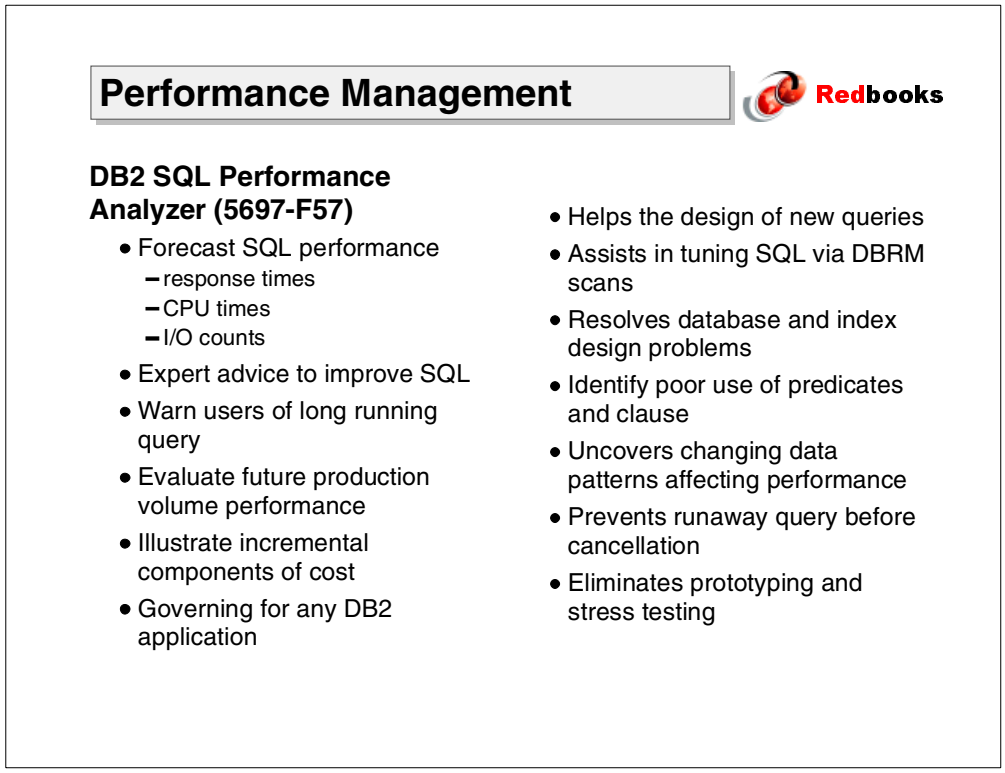
DB2 Trace data can be stored in the DB2 Performance Monitor performance database for further investigation and trend analysis.

Threshold-based and event-based exception event processing allows you to be notified of exceptional system situations immediately via online alerts, via user exits to system monitors, such as immediately via online alerts, via user exits to system monitors, such as Netview, or during report processing.

DB2 PM Version 7 also provides an application programming interface (API) to the Online Monitor data collector. Now you can retrieve performance information about the subsystem and the applications running on it and pass it to an application program. You can obtain raw data and derived performance information including snapshot information as well as recent history data. This includes exception alerts based on DB2 events and thresholds. DB2 PM Version 7 enhancements include:

- Data Sharing (Sysplex) Monitoring Online with group scope view
- Dynamic SQL statement cache monitoring
- Buffer pool data set statistics
- New Event exceptions:
  - Activity Log Dataset full
  - Activity Log Dataset full
  - Data set extent activity
  - Units of recovery in flight/in doubt

DB2 Performance Monitor continues to provide you with a powerful set of functions to do your daily work in analyzing, controlling, and tuning your DB2 environment.




## 10.11 IBM DB2 SQL Performance Analyzer

DB2 SQL Performance Analyzer for OS/390 Version 1, program number 5697-F57, delivers performance analysis for all phases of database application design and development. Developing applications under DB2 today requires the cooperative skills of the system architect, the database administrator, the programming staff, and members of the systems support team. In the daily quest to produce applications on time and within budget, too often corners are cut, such that performance is not adequately considered.

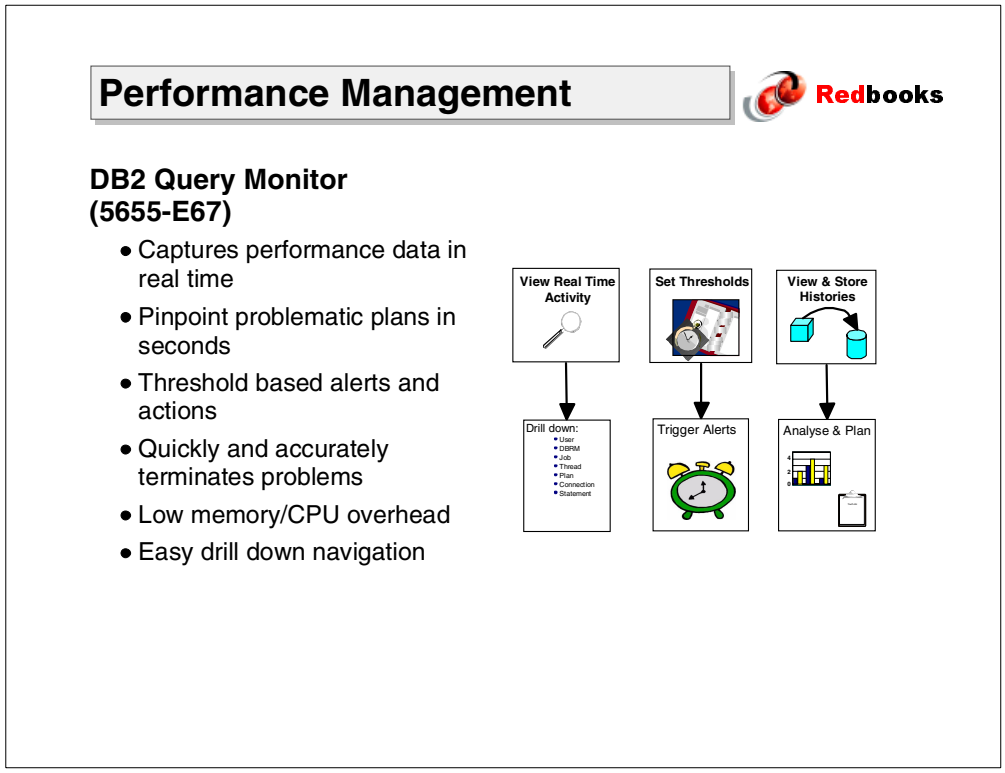
Once put into production, these applications fail to perform adequately, particularly at high transaction volumes. In addition, even applications that originally performed well may deteriorate over time due to changes in data distribution, data volume, system changes, and even changes to the DB2 product itself. Therefore, installations spend a great deal of their budget managing the performance of their applications.

All DB2 problem queries have one thing in common — they run too long. These queries cause the batch production window to shrink. Too often the online queries take what seems like forever to execute, causing customers and users to become frustrated. The most cost effective solution to the problem is prevention. IBM is introducing DB2 SQL Performance Analyzer to aid in preventing queries from running too long. With this tool you can find out how long queries will take:

- Before you run them
- Before resources are consumed
- Before the query exceeds your installations governor settings

Query cost can be determined regardless of which DB2 attach is used and regardless of whether static or dynamic SQL is used. Estimates are given in familiar units like CPU time, I/O count, and elapsed time and in even simpler terms, such as a single number representing overall cost. In addition, a monetary cost for each query is computed and delivered.

Recent enhancements have added the cost analysis for indexes othen than those chosen by the optimizer and the interface to DB2 Bind Manager.

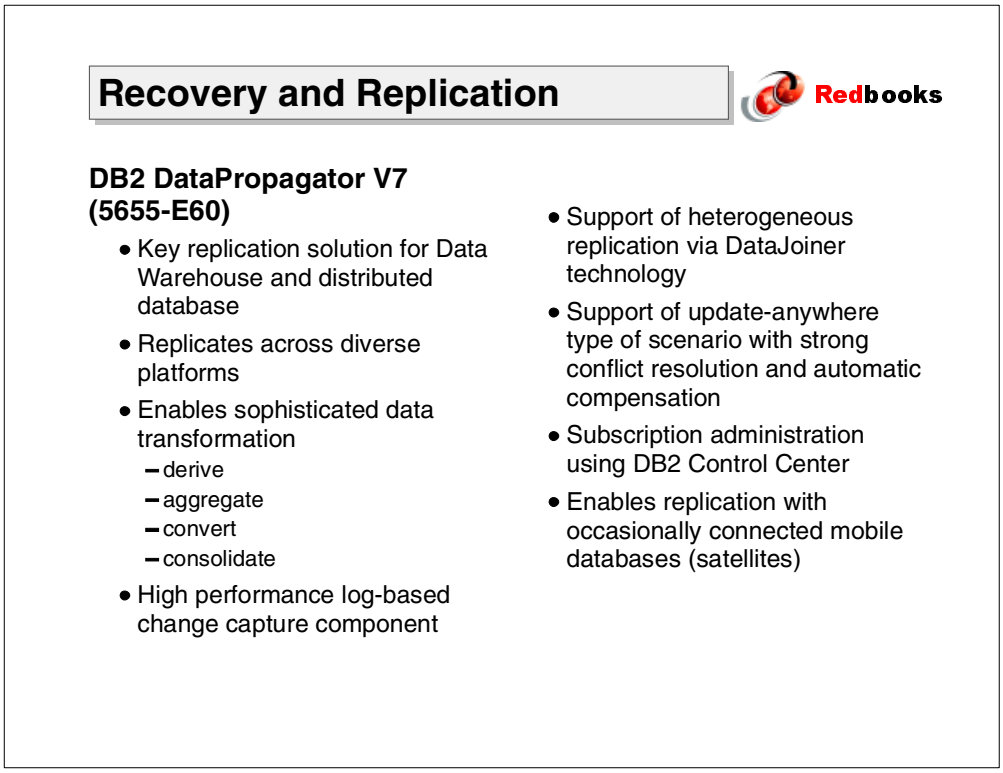



## 10.12 IBM DB2 Query Monitor

DB2 Query Monitor for OS/390 Version 1, program number 5655-E67, helps you to maximize DB2 availability. Query monitoring facilities can generate mountains of data, most of which may be insignificant for your specific purposes. The key to generating information that is useful to you is to customize what is gathered during the query monitoring process and to make sure that activity of interest grabs your attention.

IBM DB2 Query Monitor provides extensive choices in determining what data is gathered during activity monitoring, when it is gathered, about what database resources, and then what alerts and corrective action should be executed.

Monitoring agents, that can be started and stopped dynamically, use menu-driven criteria to watch up to 64 DB2 subsystems. As defined by administrators, data gathered is offloaded at intervals from memory to storage. IBM DB2 Query Monitor provides powerful, real-time views into the query processing events occurring in your enterprise OS/390 environment.




## 10.13 IBM DB2 DataPropagator

DB2 DataPropagator for OS/390 Version 7, program number 5655-E60, provides a key replication solution for data warehousing and other distributed database environments. Use it for highly efficient maintenance of consistent copies of relational data in the DB2 family of databases, automatically capturing and applying data changes.

It can help you leverage your data assets for decision making by enabling sophisticated data transformation. DB2 DataPropagator provides a powerful replication capability for the DB2 family of databases. The need to keep multiple copies of the same data in separate physical databases grows as you implement data warehouses and e-business. Data replication is an essential technology for putting timely enterprise data into the hands of your mobile worker.

DB2 DataPropagator, the core component of IBM's replication solution, unites your distributed relational databases into a cohesive and integrated database solution. It automatically captures your data changes in a source database and propagates those changes to any specified target database, keeping the two consistent. With DB2 DataPropagator, you can support these activities:

- **Re-engineering business processes**

  With DB2 DataPropagator, you can replicate transactional data to servers across your enterprise. You can also improve availability and responsiveness by moving data and applications to the point of each business transaction. Many replication products support the subsetting of data only according to information that is contained in the replicated data. Unlike these products, DB2 DataPropagator subsets data based on join predicates or subselects, allowing you to distribute data efficiently from normalized databases.

- **Going mobile**

  DB2 DataPropagator supports the unique needs of mobile users and occasionally connected systems and accommodates the infrequent, unpredictable, and expensive connections from these systems. Specifically, DB2 DataPropagator enables on-demand replication, automates connection and disconnection, and minimizes connection time. DB2 DataPropagator also lets you initiate all data transfers from the mobile units and infrequently connected users. Your mobile users can download data from a central server or upload data for consolidated processing.

- **Building powerful distributed applications**

  DB2 DataPropagator supports mobile computing, one component of a new breed of distributed applications. The update-anywhere replication capability of DB2 DataPropagator provides rigorous conflict detection and automatic compensation for offending transactions. This capability helps you maintain the integrity of your primary database and its many distributed replicas.

- **Improving your decision-making effectiveness**

  DB2 DataPropagator enables you to tailor data for maximum usability, letting you automate data enhancement as the tool copies data to the target table. For example, you can do the following tasks:

  - Derive data by using arithmetic, Boolean operators, or any valid SQL expression
  - Aggregate data to produce sums or averages by using SQL column functions
  - Convert data by translating encoded fields to descriptive fields
  - Consolidate data through joins or unions
  - Generate histories to support trend analysis

- **Managing replication with a graphical interface**

  An intuitive graphical user interface simplifies definition of replication scenarios including sources, targets, frequency and timing, replication-events, and pre-change and post-change processing. DB2 DataPropagator automatically creates and loads target tables. You can add, delete, or change replication requests while the rest of the system continues to run. See Figure 9 on page 24, which depicts how you can select settings for your replication sessions.

- **Minimizing impact on production systems and networks**

  DB2 DataPropagator uses a log-based change-capture technique that minimizes impact on transaction performance, therefore avoiding contention with source tables and inline transaction processing. DB2 DataPropagator has optimization features to support various networked environments. You can

specify distribution timing on a copy-by-copy basis. This action minimizes use of peak network periods or allows you to take advantage of economy network prices.

- **Integrating mixed database environments**

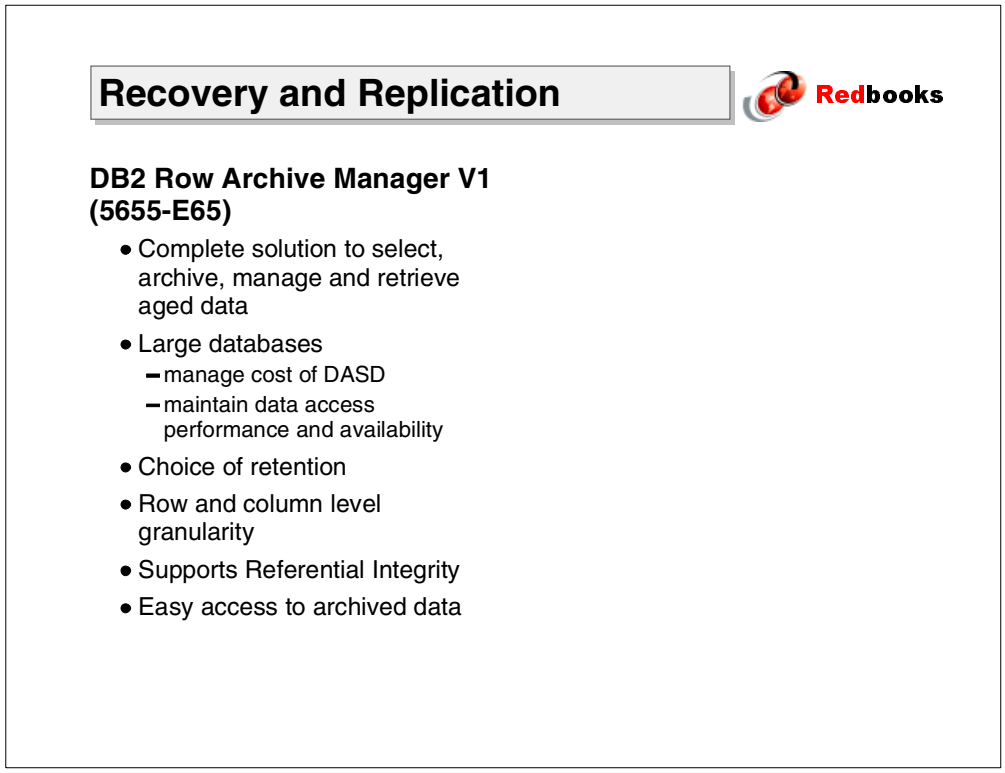
  DB2 DataPropagator expands the range of solutions available to you by supporting an open architecture. In particular:

  - DB2 DataPropagator uses standard SQL to leverage the database engine for data enhancement, network connectivity, and data security.
  - The data staging function supports interoperability among heterogeneous sources and targets, between relational and non relational formats, and among products from independent software vendors.
  - DB2 DataPropagator replication directly supports multifilament sources and targets through DataJoiner, IBM's multi database server product.

- **DB2 DataPropagator and IBM's data replication solution**

  DB2 DataPropagator establishes the base architecture for IBM's data replication solution that is based on individual components that work together. As changes occur in the source, the *Capture* component stores them in the staging table. The *Apply* component reads the staging area and applies those changes to targets, or copies data directly from the source in full-refresh mode. The *Administration* component provides a user interface for defining replication requests. DataRefresher and DataPropagator Non Relational (IMS) facilitate replication of non relational data from enterprise servers. The components can populate the data staging area for DB2 DataPropagator with IMS or VSAM data. DB2 DataPropagator then enhances, distributes, and applies data to the target tables to provide an end-to-end replication solution from multiple database sources.

New with this version is support of UNICODE and ASCII encoding schemes, which minimizes the need for data conversion in the replication environments.




## 10.14 IBM DB2 Row Archive Manager

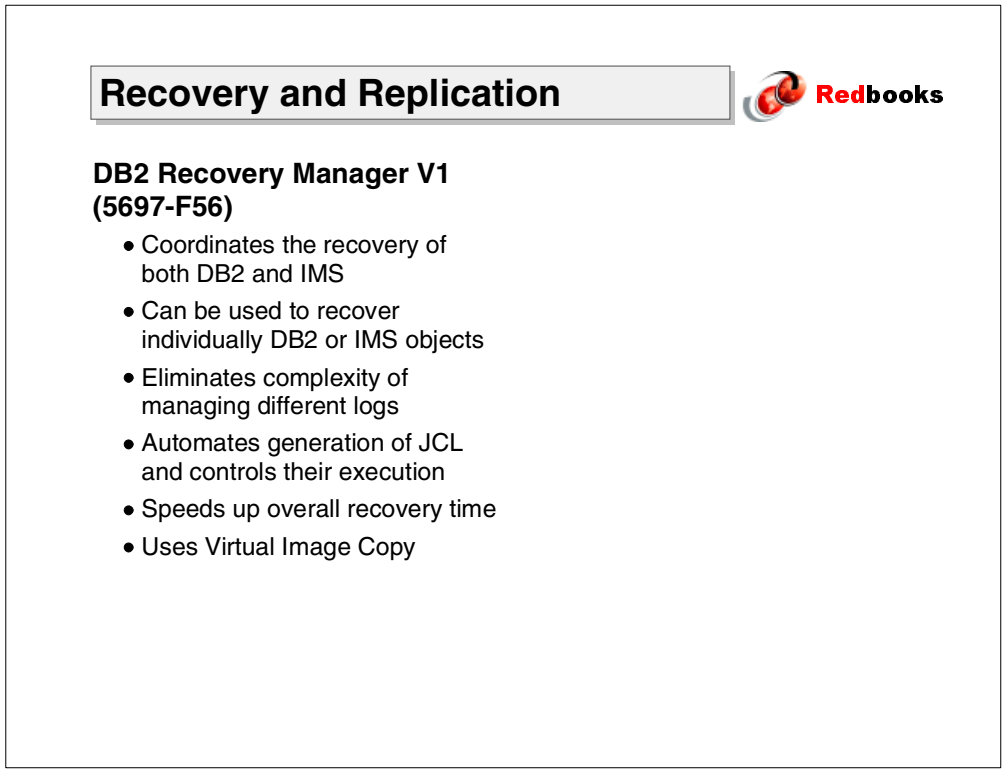
DB2 Row Archive Manager, program number 5655-E65, Version 1or OS/390 can save storage, improve performance, simplify maintenance, and reduce the overall costs of your DB2 environment. DB2 RAM provides a simple method to control the separation of aged data from your active DB2 data. You can use DB2 RAM to move seldom used data to a less costly storage medium.

DB2 Row Archiving Manager gives you a facility for separating aged data from active data, and archiving the aged data onto a less costly storage medium. The archived data can be selectively retrieved on demand. Archiving aged data results in less active storage requirements and less active data for DB2 to process. This can mean lower cost and better performance for your DB2 environment.

DB2 Row Archive Manager implements a set of rules specified by the administrator to determine what data is eligible to be archived. It allows data to be selected at a low level of granularity, for example, at a row level. This allows the administrator to precisely control which aged data is archived. Only selected rows are archived at any specified time and not all columns of a row need to be archived, rows from related tables can be archived as a unit. Individual tables or systems of related tables can be archived.

DB2 Row Archive Manager archives selected aged data into archive table spaces. It manages a catalog used to determine how to retrieve the aged data if it is needed by an application. Besides, this tool performs storage management functions to effectively manage the physical storage used by the archived table

spaces. It also includes additional utilities, such as the REMOVE utility, which can remove erroneous or obsolete archive specifications from the system.



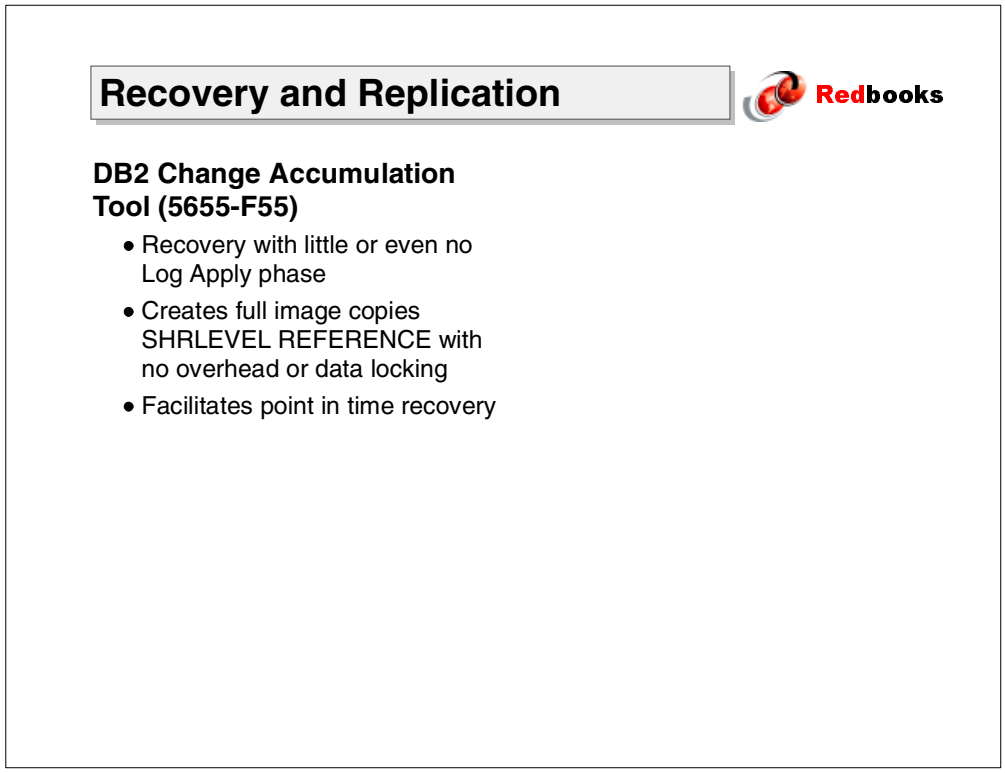

## 10.15 IBM DB2 Recovery Manager

DB2 Recovery Manager, program number 5697-F56, for OS/390 simplifies and coordinates the recovery of both DB2 and IMS data to a common point, cutting the time and cost of data recovery and availability. It eliminates the error-prone complexity of managing different logs, utilities and processes to do recovery from both databases.

Many businesses use both DB2 and IMS in their online transaction environment. An application can access data in both databases. When the application commits, IMS and DB2 coordinate the data changes so that all changes occur or none occur. However, if at some later time, you need to recover both IMS and DB2 to the same point, then you must deal with different logs, different utilities, and different processes to do the recovery. This leads to complex recovery scenarios that are time-consuming and error-prone. Each product must have its data recovered separately. The DB2 Recovery Manager is a new feature that simplifies this process.

The DB2 Recovery Manager works with IMS, DB2, or both. The DB2 Recovery Manager uses image copies for either product or both products. The tool processes the individual logs and works with incremental image copies for DB2 and the output from the change accumulation utility for IMS. Recovery Manager establishes synchronization points for the recovery. Recovery Manager calls such a synchronization point a virtual image copy. In the situation as described, after the application runs, you invoke the DB2 Recovery Manager to establish a virtual image copy. The DB2 Recovery Manager does not require image copies.

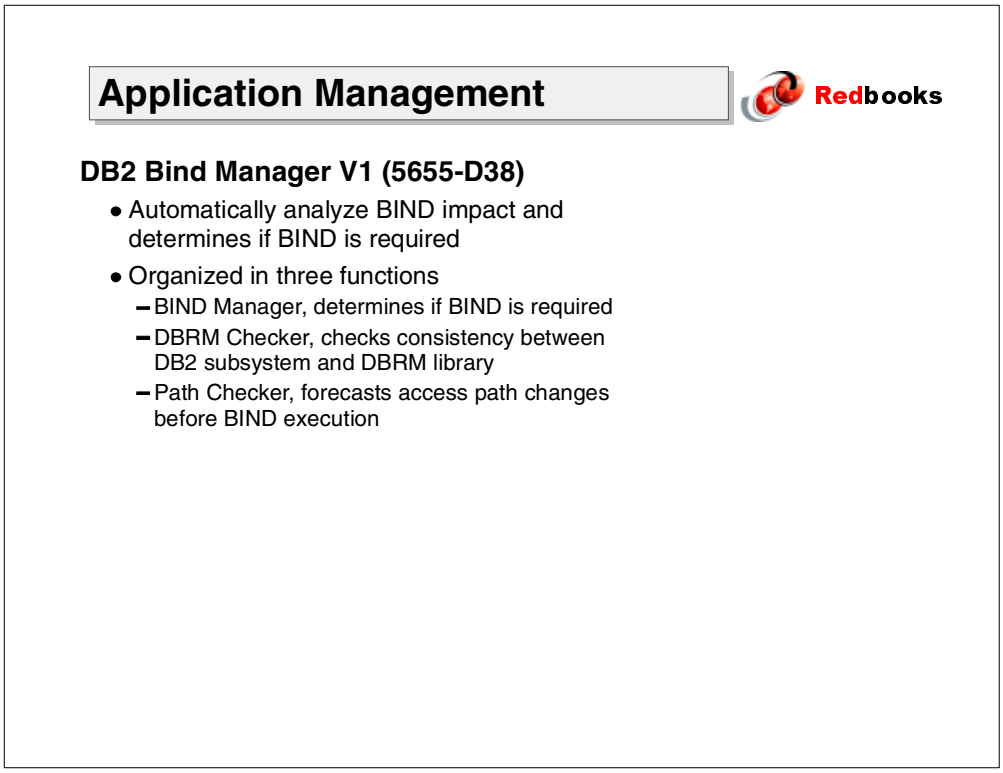
The DB2 Recovery Manager establishes a quiesce point and records that point on the respective logs. During recovery the user specifies the need to recover to this virtual image copy. The DB2 Recovery Manager applies the appropriate image copies and causes the database to apply the log to this point. If you prefer not to use virtual image copies for recovery, you can use the DB2 Recovery Manager to automate the recovery of resources for either DB2 or IMS. The DB2 Recovery Manager generates the JCL, locates the proper image copies, and controls execution of the jobs.




## 10.16 IBM DB2 Change Accumulation Tool

DB2 Change Accumulation for OS/390 and z/OS, program number 5655-F55, provides DB2 administrators with a powerful tool for restoring database objects in the most precise and least disruptive manner possible by:

- Making precise point-in-time recovery of database objects simple and reliable
- Allowing recovery routines to focus on single objects and previous states
- Producing SHRLEVEL REFERENCE image copies without the associated overhead and data locking
- Controlling the scope and specificity of image copy creation precisely via control cards
- Maintaining data integrity without recovery to RBA
- Reducing recovery session times significantly in many cases
- Providing low overhead and minimizing downtimes for high-volume, complex databases with large numbers of tables and dependencies

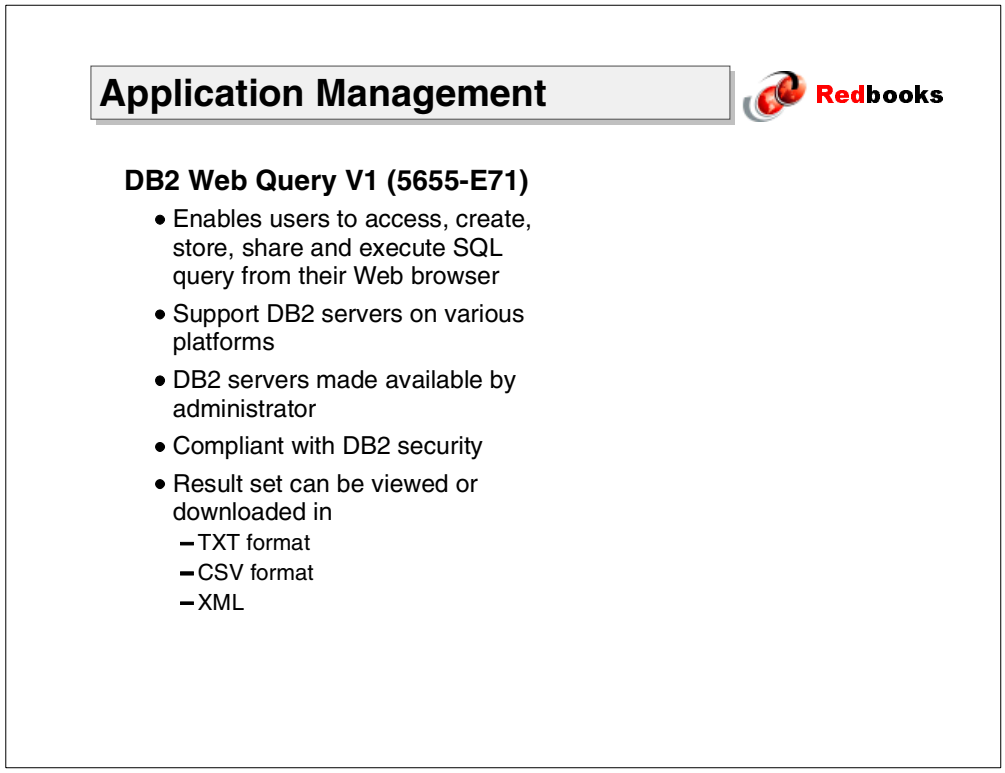



## 10.17 IBM DB2 Bind Manager

Use of DB2 Bind Manager, program number 5655-D38, can pay major dividends in change management, since it will automatically detect production application changes requiring a bind. This frees DBAs from analyzing bind impacts and allows them to concentrate only on the changes that affect the SQL structure.

Consistency checking may be done between an existing DBRMLIB and a DB2 subsystem using the DBRM Checker function of DB2 Bind Manager. DBRM Checker will identify DBRMs by plan that have consistency tokens that are inconsistent with those in the DB2 catalog tables. You can determine which of the DBRMs you need to bind. An application may have hundreds of packages (that is, DBRMs) making up one plan. If you only changed one, then you only need to bind that one package.

Using the Path Checker function of DB2 Bind Manager, you can quickly determine whether a bind of a DBRM will result in a changed access path. This is usually done when a new release (or version) of DB2 is installed, service is applied and/or you are migrating a large application from one system to another. You've spent many hours fine tuning the SQL for performance only to have the optimizer select a path you didn't expect. By using Path Checker, you can see the effects of doing a bind (or having done a bind). Path Checker does an EXPLAIN into your plan table of the *new* DBRM and gives you a report on those that changed. You can then use any EXPLAIN tool to look at the result and determine whether you need to take action or not.




## 10.18 IBM DB2 Web Query

Connecting to enterprise data most likely involves a large effort: complex configurations, obscure procedures, and in depth user training are necessary preparation steps in order to access, and finally be productive using, enterprise data.

The resulting time, effort, expense and, above all, waiting creates a gap between opportunity and action that cost your organization every time it happens, perhaps thousands of times a day.

DB2 Web Query tool, program number 5655-E71, changes this old paradigm and helps eliminate the pain previously associated with data access. With DB2 Web Query tool, end users and administrators have a single, powerful tool for bringing data access into the e-business age with speed, reliability and simplicity.

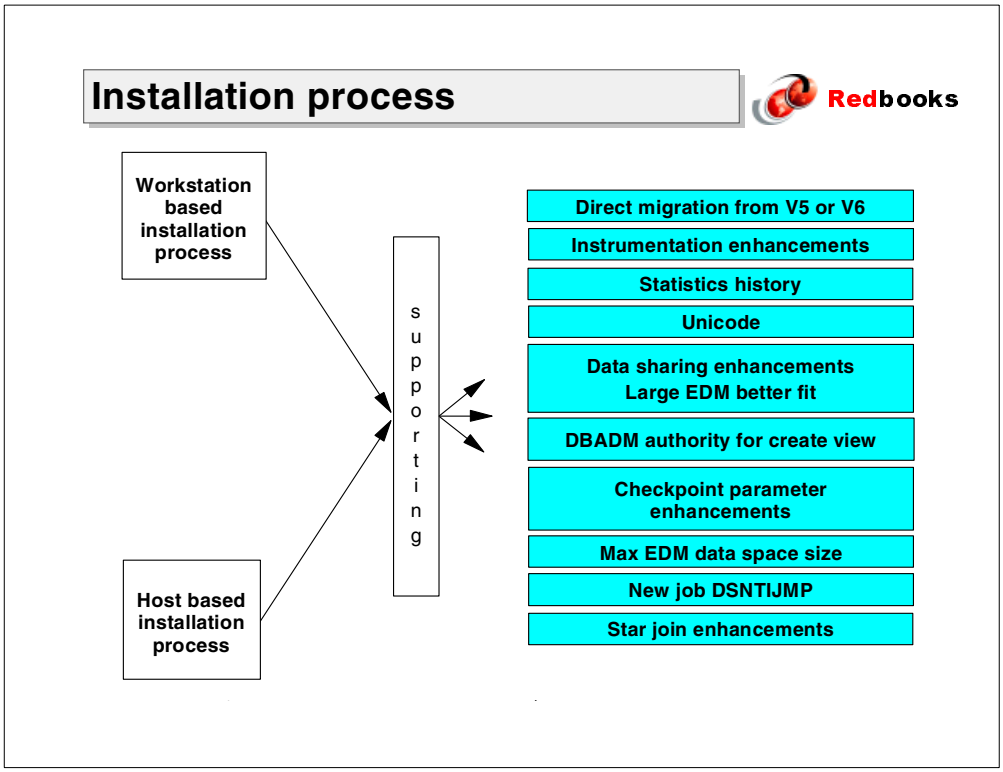
DB2 Web Query tool sets a new standard for business responsiveness because now anyone in your organization can take robust data access for granted, virtually anywhere and at anytime. Less latency; less waiting; and less waste means more opportunity for you and your organization.

With the trusted architecture of DB2, the DB2 Web Query tool enables pervasive connectivity over the Internet to every desktop from the novice user to the expert.

# Part 8. Installation and migration

© Copyright IBM Corp. 2001

# Chapter 11. Installation




The installation and migration processes of DB2 V7 have been adapted to take care of the new possibilities DB2 V7 offers. The workstation and the host based installation process reflect the changes.

***Workstation based installation process (DB2 installer)***
DB2 Installer is a workstation feature that is distributed with DB2 V7 under the Client Tools Package. It is a graphical user interface for customizing DB2 for install, migrate, update, fallback, data sharing or SMP/E. DB2 Installer establishes host communications via TCP/IP, and uses those connections to query the user's MVS system, run jobs, and return output to the workstation.

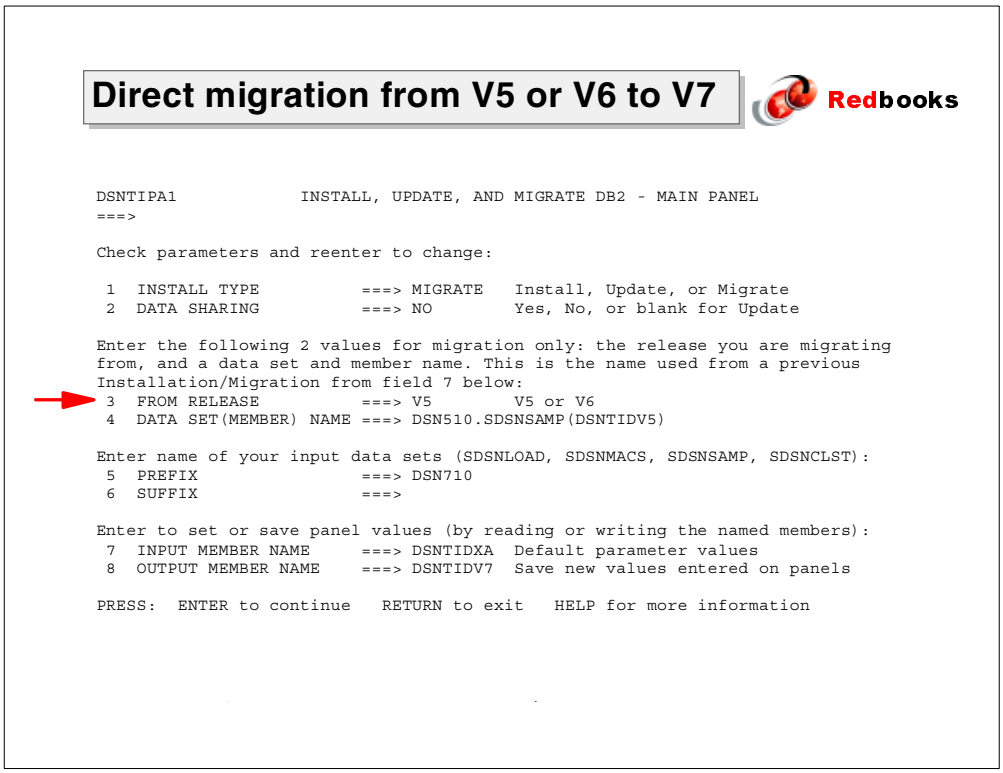
DB2 Installer is a usability tool for users who must customize and modify the subsystem parameters for DB2. It provides an alternative to the existing ISPF installation panels and CLISTs currently used on OS/390 systems.

DB2 Installer makes use of the stored procedure (DSNWZP) distributed with DB2, to optionally gather the user's current parameter settings from a specified DB2 subsystem. This can be used in place of using older DSNTIDxx member or the default DSNTIDXA settings. DB2 Installer keeps track of the definitions for multiple DB2 subsystems, includes SMP/E fallback examples, and has a new icon to highlight the changes for DB2 V7.

***Host based installation process***
The host based installation process of DB2 is the ISPF panel and CLISTs based process used on OS/390 systems to customize DB2 for install, migrate, update, or fallback.

© Copyright IBM Corp. 2001

The host based installation process will be executed from the TSO command line as in previous releases. The only parameter that may be passed to the CLIST is CONTROL to specify the tracing level.




## 11.1 Direct migration from V5 or V6 to V7

Migration to V7 of DB2 is supported either from V5 or from V6.

Migration, fallback, and data sharing coexistence with the ability to skip a release offers many possibilities. Some customers need the capabilities of V6 as soon as possible. Other customers may be running on V3 or V4 now. They can plan their V5 migration this year, then skip over V6 and go directly to V7.

For further migration considerations, please refer to 12.2, “Migration considerations” on page 499.

***Main panel DSNTIPA1***
On the main panel the following item(s) were added:

***FROM RELEASE***

| | |
|---|---|
| Acceptable values: | V5 or V6 |
| Default: | none |
| DSNZPxxx | none |

This parameter specifies from which release you are migrating from. The DB2 release indicator of the input member (item 4 ‘DATA SET(MEMBER) NAME’) is compared with the ‘FROM RELEASE’ value to assure correct release for migration input member.

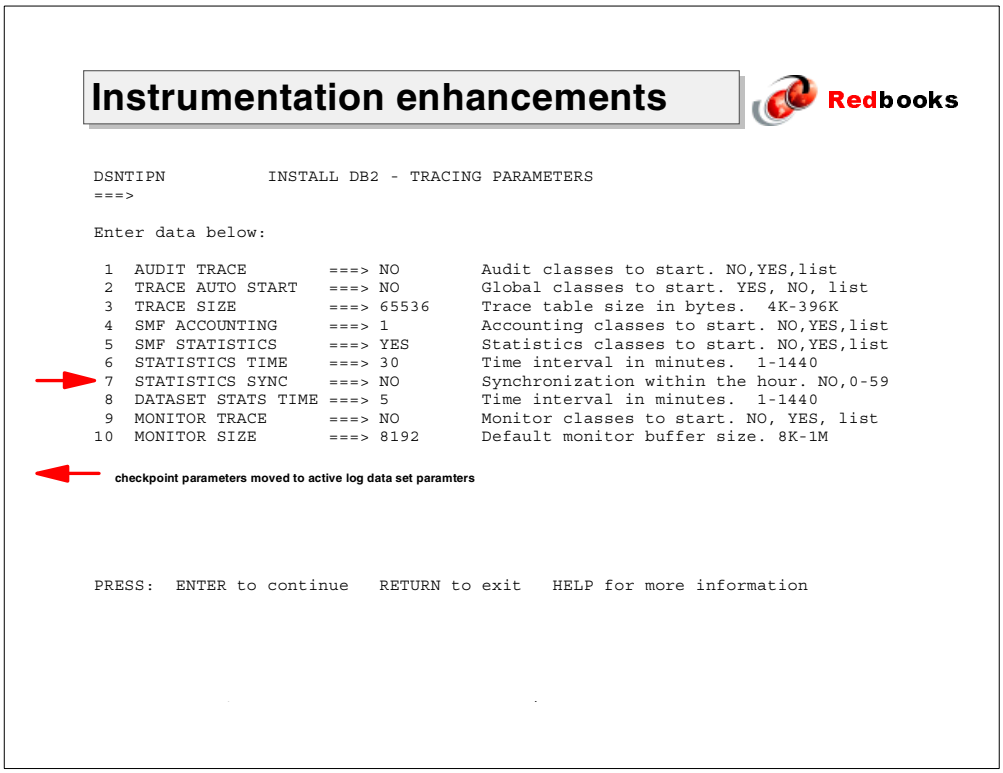
Message DSNT509I is issued to inform the user about a release mismatch:

```
DSNT509I WARNING - MIGRATION INPUT MEMBER LEVEL IS 510. LEVEL 610 IS REQUIRED.
RETURN TO PANEL DSNTIPA1 TO CHANGE MIGRATION INPUT MEMBER.
```

The job editing is done according to the migration release.

When migrating from V5, the following rules apply:

- Job DSNTIJMV does not rename DSNHCPPS and DSNHSQL procedures, as they are new to V6. The rename is done only for migration from V6 to V7.
- Job DSNTIJSG drops and recreates the index DSNARL01 and performs alterations on the resource limit specification table (RLST), since it was changed in V6.
- The fallback job DSNTIJFV does not rename procedures new to V6.
- The specifications of the total size of the 4 KB and 32 KB table spaces are converted from bytes to megabytes.
- The default database protocol, which is new in V6, is set to PRIVATE when migrating from V5 to V7. The default for V7 is DRDA.




## 11.2 Instrumentation enhancements

The STATISTICS SYNCH parameter specifies whether DB2 statistics recording is to be synchronized with some part of the hour. This allows for DB2 statistics to be synchronized with SMF and RMF reporting intervals, as well as across the DB2 members of a data sharing group. The installation can specify that the DB2 statistics recording interval be synchronized with the beginning of the hour (00 minutes past the hour) or any number of minutes past the hour up to 59. If NO is specified, then no synchronization is done. This parameter has no effect if STATISTICS TIME is greater than 60.

**Example:**

Assume the installation wants the DB2 statistics recording interval to have a length of 15 minutes and to be synchronized with 15 minutes past the hour, which means that DB2 statistics are recorded at 15, 30, 45, and 60 minutes past the hour. To establish this interval, specify the following: STATIME=15, SYNCVAL=15

***Tracing parameters panel DSNTIPN***

On the tracing parameters panel, the following item(s) were added or removed:

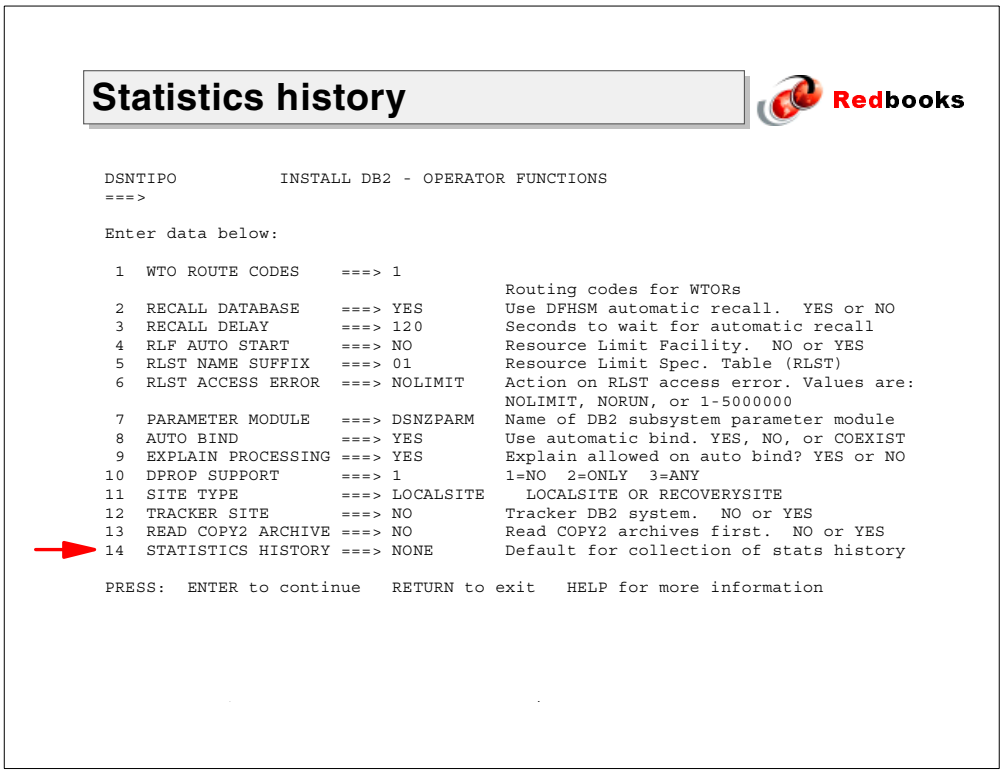
### *STATISTICS SYNC*

| | |
|---|---|
| Acceptable values: | NO, 0-59 |
| Default: | NO |
| DSNZPxxx: | DSN6SYSP SYNCVAL |

STATISTICS SYNC provides an option for synchronization of DB2 statistics recording across the DB2 members of a data sharing group.

### *Checkpoint Parameters*

The checkpoint parameters on the tracing parameter panel were cut out and moved to the active log data set parameters panel DSNTIPL.




## 11.3 Statistics history

As the volume of business activities grows in an organization, this will require changes to the physical design of the DB2 objects over a period of time. Statistics history will help to track these changes. The information stored in the catalog for the DB2 objects could be analyzed to determine when a change of the physical design is needed. If we have the data stored in the catalog on a historical basis, this would enable us to do trend analysis on the available data. The analysis would help to determine when it is appropriate to execute utilities for maintenance of the DB2 objects. Based on the available statistics, decisions such as when to run online reorg to improve DB2 performance can be made.

For further information, please refer to 5.9, “Statistics history” on page 270.

***Operator functions panel DSNTIPO***

On the Operator functions panel the following item(s) were added:

### *STATISTICS HISTORY*

| | |
|---|---|
| Acceptable values: | SPACE, ACCESSPATH, ALL, NONE |
| Default: | NONE |
| DSNZPxxx: | DSN6SPRM STATHIST |

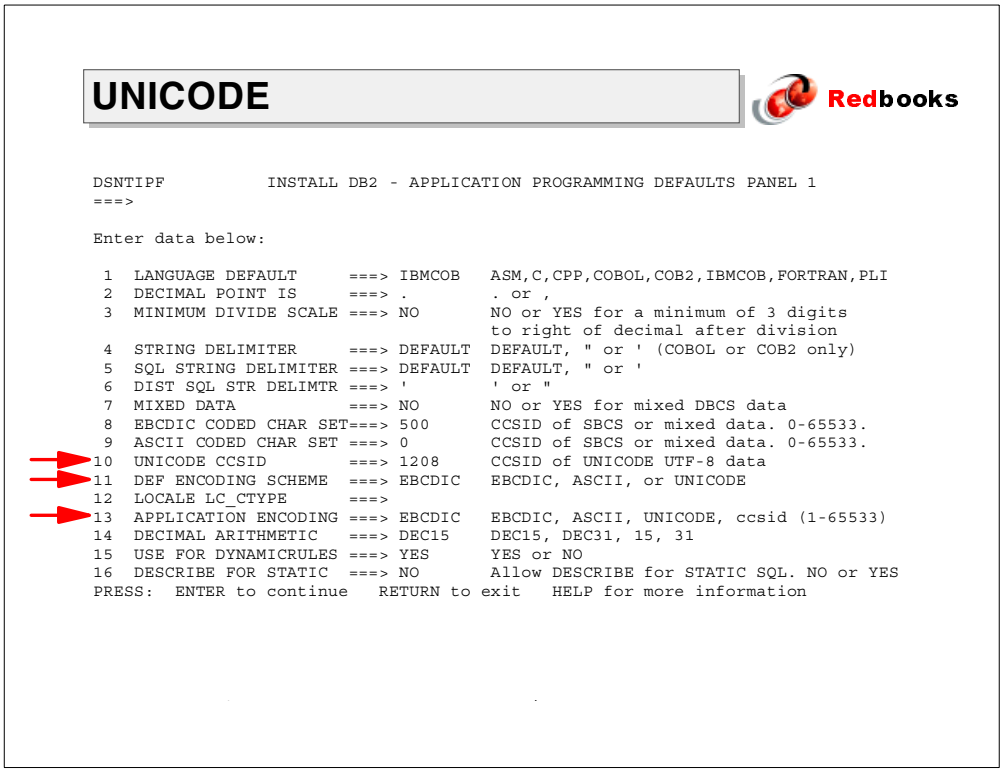
STATISTICS HISTORY provides an option for the collection of historical catalog statistics in the new history tables of the DB2 catalog.

SPACE specifies all inserts/updates made by DB2 to space related catalog statistics are recorded in catalog history tables.

ACCESSPATH specifies all inserts/updates made by DB2 to ACCESSPATH related catalog statistics are recorded in catalog history tables.

ALL specifies all inserts/updates made by DB2 in the catalog are recorded in catalog history tables.

NONE specifies changes made in the catalog by DB2 are not recorded in catalog history tables. This the default for the HISTORY subsystem parameter.




## 11.4 UNICODE

DB2 UDB for OS/390 is increasingly being configured as part of client/server systems. Character representations vary on clients and servers across different platforms and across many geographies.

In DB2 V5, storage of data encoded in ASCII was added to help address some of the problems that client/server solutions were having on OS/390. The ASCII support only solved part of the problem (padding and collation). The code added in V5 did not address the problem of users in many different geographies interacting with one DB2 server.

One area where this sort of environment exists is in the data centers of multinational corporations. Another example is e-commerce. In both of these situations, a geographically disparate group of users interact with a central server, storing or retrieving data.

Given the capabilities of DB2 UDB for OS/390 today, these users are really limited to the Latin-1 subset of ASCII or EBCDIC to represent the data used in their transactions. This is because DB2 UDB for OS/390 only allows one set of EBCDIC and one set of ASCII CCSIDs per system. ASCII and EBCDIC CCSIDs are set up to support either one specific geography (for example, 297 is French EBCDIC) or one generic geography (for example, 500 is Latin-1 which applies to Western Europe). There are no generic CCSIDs for the Far East (meaning no CCSID supports more than one Far Eastern country).

With DB2 V7, the concept of a UNICODE CCSID is introduced to DB2 UDB for OS/390 and z/OS. UNICODE is an encoding scheme that is able to represent the codepoints/characters of many different geographies and languages. To support all geographies, UNICODE requires more than one byte to represent a character.

For further information, please refer to 6.3, “UNICODE” on page 297.

***Application programming defaults panel 1 DSNTIPF***
On the application programming defaults panel the following item(s) were added:

***UNICODE CCSID***

| | |
|---|---|
| Acceptable values: | 1208 |
| Default: | 1208 |
| DSNHDECP | USCCSID (single-byte), UMCCSID (mixed), UGCCSID (graphic) |

This parameter specifies the CCSID of UNICODE data. This field will be pre-filled with 1208 (the UTF-8 CCSID) in the field. DB2 will pick the CCSIDs for the double byte and single byte CCSID values (1200 for DBCS and 367 for SBCS).

***DEF ENCODING SCHEME***

| | |
|---|---|
| Acceptable values: | EBCDIC, ASCII, UNICODE |
| Default: | EBCDIC |
| DSNHDECP | ENSCHEME |

Specify the format in which to store data in DB2. If you specify DEF ENCODING SCHEME=ASCII and MIXED DATA=YES, specify a mixed ASCII CCSID for ASCII CODED CHAR SET.

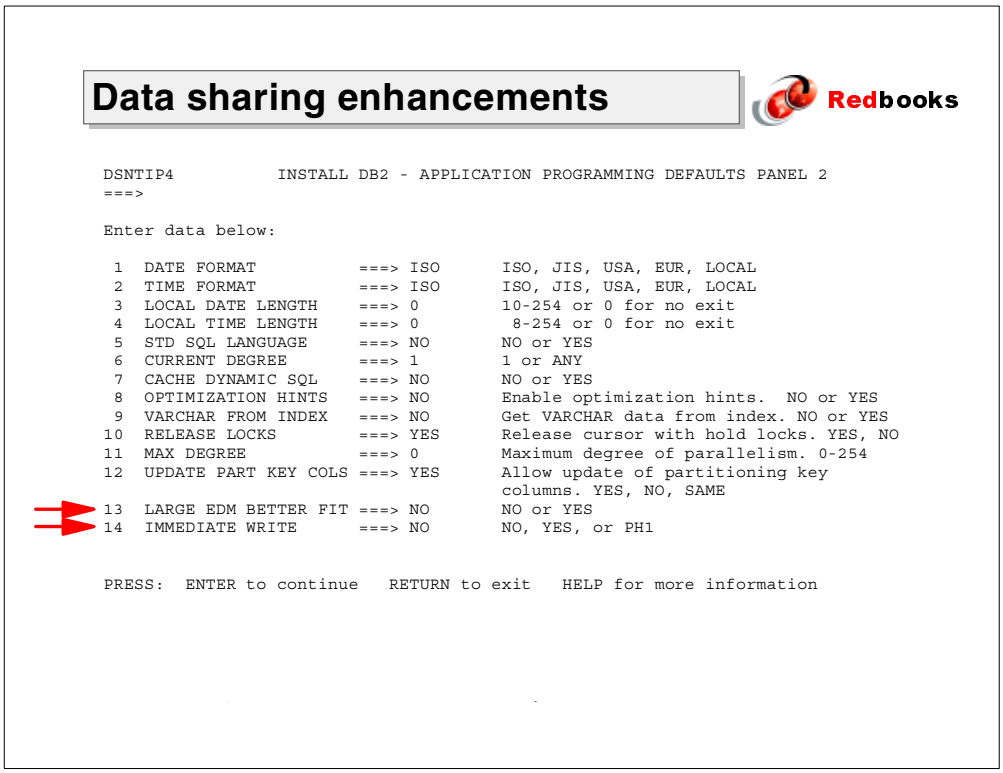
DEF ENCODING SCHEME now accepts a value of UNICODE as well as EBCDIC or ASCII.

***APPLICATION ENCODING***

| | |
|---|---|
| Acceptable values: | EBCDIC, ASCII, UNICODE, ccsid (1-65533) |
| Default: | EBCDIC |
| DSNHDECP | APPENSCH |

This parameter specifies the system default application encoding scheme, which affects how DB2 interprets data incoming into DB2. The default value of EBCDIC causes DB2 to retain the behavior of previous releases of DB2 and should not be changed if compatibility with previous releases of DB2 is desired.

**Note:** It is strongly recommended not to change the CCSIDs once they have been specified. Results of SQL may be unpredictable if this recommendation is not followed.




## 11.5 Data sharing enhancements

A new Bind option called IMMEDWRITE was delivered in V5 (APAR PQ22895) to help solve the problem of order-dependent transactions in a data sharing environment. It specifies if or when updates to GBP-dependent buffers are to be written to the CF. The IMMEDWRITE option values are:

- NO (default) - GBP-dependent pages are written to the CF at or before phase 2 of commit.
- PH1 - GBP-dependent pages are written to the CF at or before phase 1 of commit.
- YES - GBP-dependent pages are written immediately to the CF (as soon as the buffer update completes).

A new keyword subsystem parameter was added to DSN6GRP and DSNTIJUZ to allow the user to specify at a DB2 member level whether immediate writes or phase 1 writes should be done.

In DB2 V7, the immediate write parameter is now externalized to the application programming defaults panel DSNTIP4.

For further information, please refer to 8.3.2, “IMMEDWRITE BIND option in V7” on page 394.

### *Application programming defaults panel 2 DSNTIP4*

On the application programming defaults panel, the following item(s) were added:

#### *IMMEDIATE WRITE*

| | |
|---|---|
| Acceptable values: | YES, NO, PH1 |
| Default: | NO |
| DSNZPxxx | DSN6GRP IMMEDWRI |

This specifies if and when updates to GBP-dependent buffers are to be written to the CF. NO is the default and tells DB2 to not immediately write the change buffer to the CF (instead wait until commit). YES tells DB2 to immediately write the page to the CF after it is updated. PH1 tells DB2 to write the updated page during phase 1 commit.

#### *LARGE EDM BETTER FIT*

| | |
|---|---|
| Acceptable values: | NO or YES |
| Default: | NO |
| DSNZPxxx | DSN6SPRM EDMBFIT |

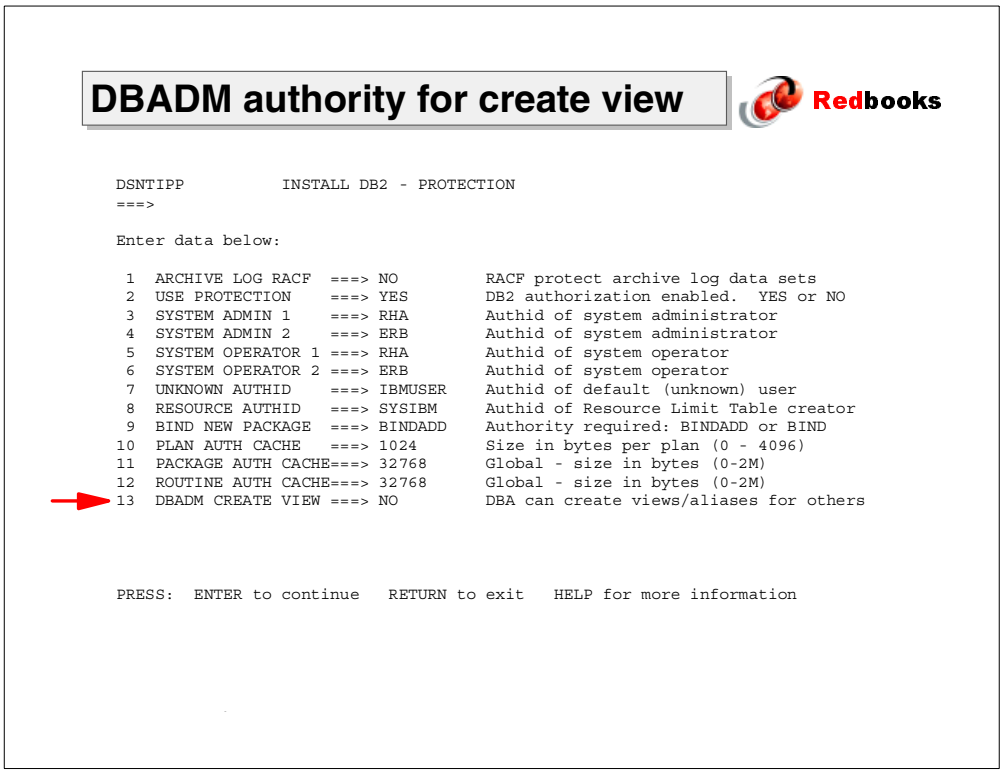
There is a trade-off between performance and storage utilization with the EDM pool. For smaller EDM pools, storage utilization (fragmentation) is normally more critical. For larger EDM pools, performance is normally more critical. This parameter is used to specify how the free space is utilized for large EDM pools (greater than 40M).

NO, the default, indicates that for large EDM pools DB2 should optimize for performance (use a first fit method in the free chain search).

YES, indicates that for large EDM pools (greater than 40M) DB2 should optimize for better storage utilization (use a better fit method in the free chain search).

For additional EDM pool information, please refer to 11.8, “Maximum EDM data space size” on page 487.

This parameter is not just a data sharing enhancement, but it is covered under that heading. It was added to V5 via APAR and is just externalized in V7.




## 11.6 DBADM authority for create view

These changes allow an authorization ID with DBADM authority to create a view or an alias for another authorization ID. This other authorization ID becomes the owner of the view or alias.

DBADM on any one of the underlying tables that a CREATE VIEW is based on, is sufficient for creation of a view for another ID. This view could be based on tables or a combination of tables and views. However, the view being created must be based on at least one table for this capability.

Although having DBADM on an underlying table of a CREATE VIEW is sufficient for creation of a view for another ID, all the other requirements for creating views have still to be met. For example: If the view involves user defined functions then the view owner will also need to have EXECUTE on the user defined function.

Here are some examples to clarify the change:

- If the view is based on tables and views in more than one database, then having DBADM on any one of the underlying databases is sufficient for creating a view for another ID provided the other view creation requirements are met.
- If the view is based only on views and no tables, then DBADM cannot create this view for another ID.
- If a view is based on tables and views in more than one database, and the view creator has DBADM on one of those databases, but the view owner does not have SELECT on one of the underlying tables/views in another database, then the view creation will fail, just like it does today.

The view owner will have SELECT on the created view without the GRANT option and be able to drop it.

Since not every environment needs this capability, a subsystem parameter will be provided. Subsystem parameter DBACRVW will control this DBADM ability to create views for others.

The programmer response for SQLCODE -164 is changed:

> **Programmer response**: Do not attempt to create views with other than your own ID as a qualifier. Only an authorization ID that holds SYSADM or DBADM authority, can create views for other authorization IDs. The DBADM should be on any of the databases that contain at least one of the tables on which this CREATE VIEW is based.

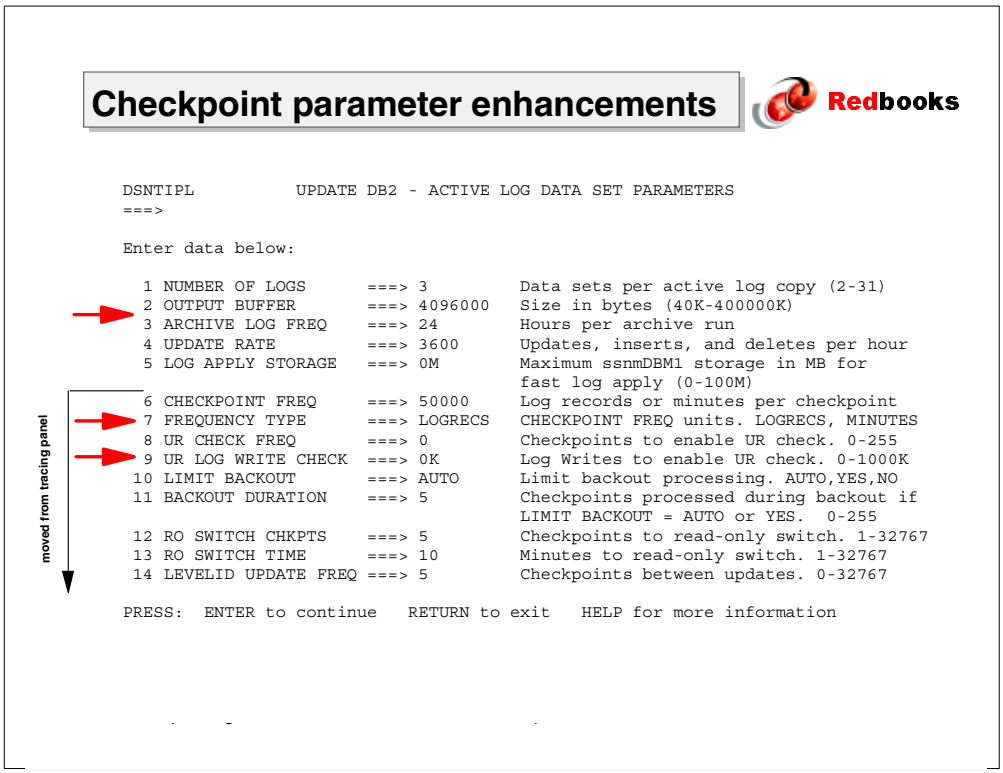
This change also affects the access control authorization (ACA) exit.

***Protection panel DSNTIPP***
On the application programming defaults panel, the following item(s) were added:

***DBADM CREATE VIEW***

| | |
|---|---|
| Acceptable values: | NO, YES |
| Default: | NO |
| DSNZPxxx | DSN6SPRM DBACRVW |

This specifies whether an authorization ID with DBADM authority can create a view for another authorization ID.




## 11.7 Checkpoint parameter enhancements

The checkpoint parameters are moved from the tracing panel to the active log data set parameters panel DSNTIPL.

### *Active log data set parameters panel DSNTIPL*

On the active log data set parameters panel, the following item(s) were added, changed, or removed:

### *WRITE THRESHOLD*

The write threshold parameter on panel DSNTIPL is no longer externalized. It becomes a hidden parameter in V7. The default remains 20.

### *CHECKPOINT FREQ*

| | |
|---|---|
| Acceptable values: | 200-16000000 (log records), 1-60 (minutes) |
| Default: | 50000 |
| DSNZPxxx | DSN6SYSP CHKFREQ |

The CHECKPOINT FREQ field now allows specification of a range of minutes as well as log records (range unchanged). The new FREQUENCY TYPE parameter determines if the checkpoint freq value is interpreted as number of log records or minutes. The value for number of log records can still be entered as bytes, kilobytes (by appending a K), or megabytes (M). Also, the keyword in DSNTIJUZ is changed from LOGLOAD to CHKFREQ to reflect the change.

### *FREQUENCY TYPE*

| | |
|---|---|
| Acceptable values: | LOGRECS or MINUTES |
| Default: | LOGRECS |
| DSNZPxxx | none |

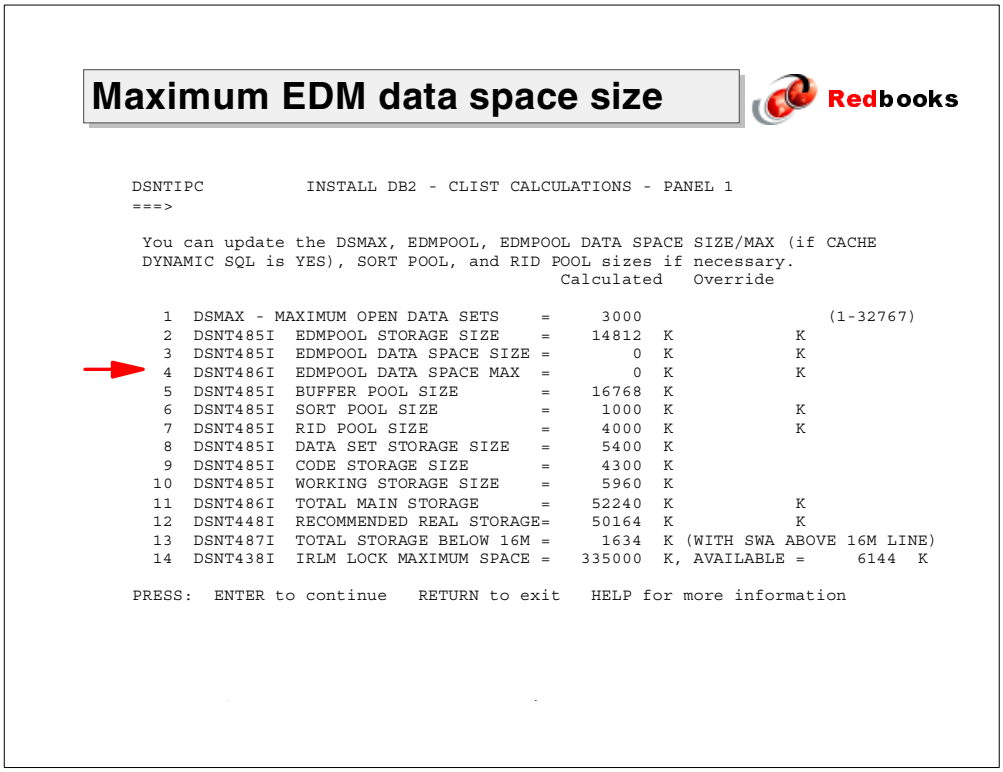
The FREQUENCY TYPE field indicates whether a time value of minutes or log records are to be used as units for CHECKPOINT FREQUENCY. The value is used to verify the CHECKPOINT FREQ parameter.

### *UR LOG WRITE CHECK*

| | |
|---|---|
| Acceptable values: | 0-1000K |
| Default: | 0K |
| DSNZPxxx | DSN6SYSP URLGWTH |

The UR LOG WRITE CHECK parameter specifies the number of log records written by an uncommitted unit-of-recovery (UR) before DB2 will issue a warning message to the console. The purpose of this option is to provide notification of a long-running UR that may result in a lengthy DB2 restart or a lengthy recovery situation for critical tables. The value is specified in 1K (1000 log record) increments. A value of 0 indicates that no UR log-write check is to be done.

Note: LIMIT BACKOUT and BACKOUT DURATION parameters have been introduced with DB2 V6 to support consistent Restart.




## 11.8 Maximum EDM data space size

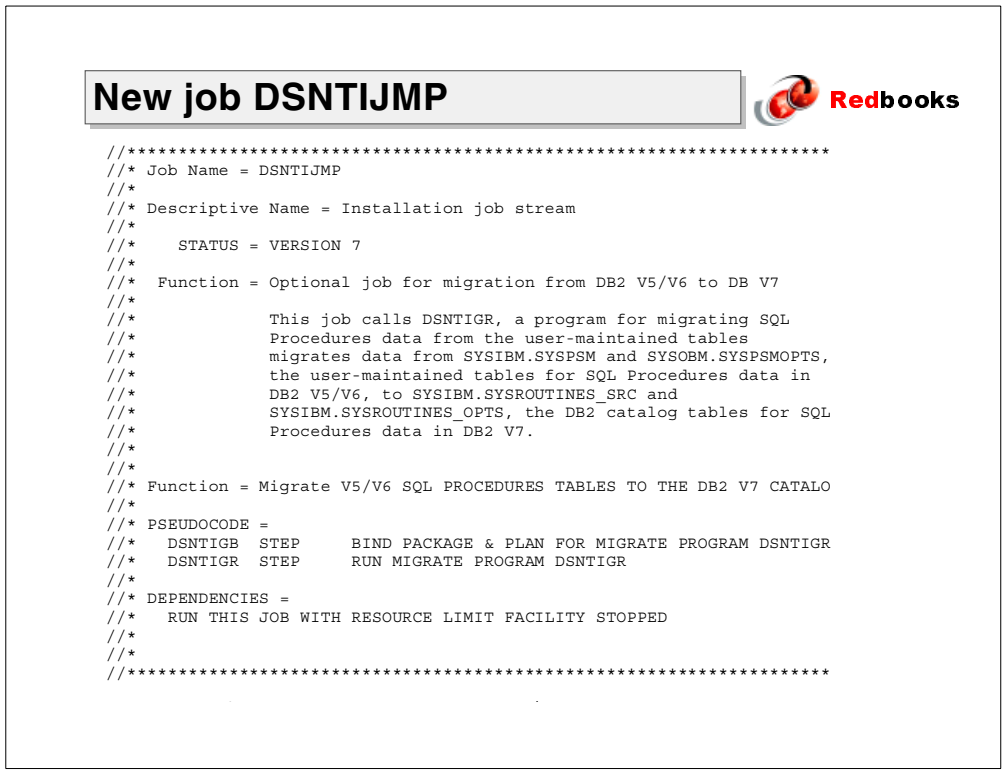
### *CLIST calculations panel DSNTIPC*

On the CLIST calculation panel the following item(s) were added:

### *EDMPOOL DATA SPACE MAX*

| | |
|---|---|
| Acceptable values: | 0-2097152 (when CACHE DYNAMIC SQL is YES) |
| Default: | 1048576 (1G, when CACHE DYNAMIC SQL is YES) |
| DSNZPxxx | DSN6SPRM EDMDSMAX |

This parameter specifies the maximum size in kilobytes that the data space used by EDM can expand. When ‘CACHE DYNAMIC SQL’ is YES on panel ‘DSNTIP4’ the default value will be 1048576 (1G). If NO, a zero will be used for the calculated value. The value is set at DB2 startup and it can not be modified by the SET SYSPARM command (to allow the size of the data space used by EDM to be increased and decreased the maximum size of the data space needs to be known at DB2 startup). Some installations may want to set a maximum size smaller than 2G because they do not have the real storage to back up 2G.




## 11.9 New job DSNTIJMP

For the migration of the SQL procedure data from user maintained tables to the new DB2 catalog tables, a new job DSNTIJMP is created during migration mode.

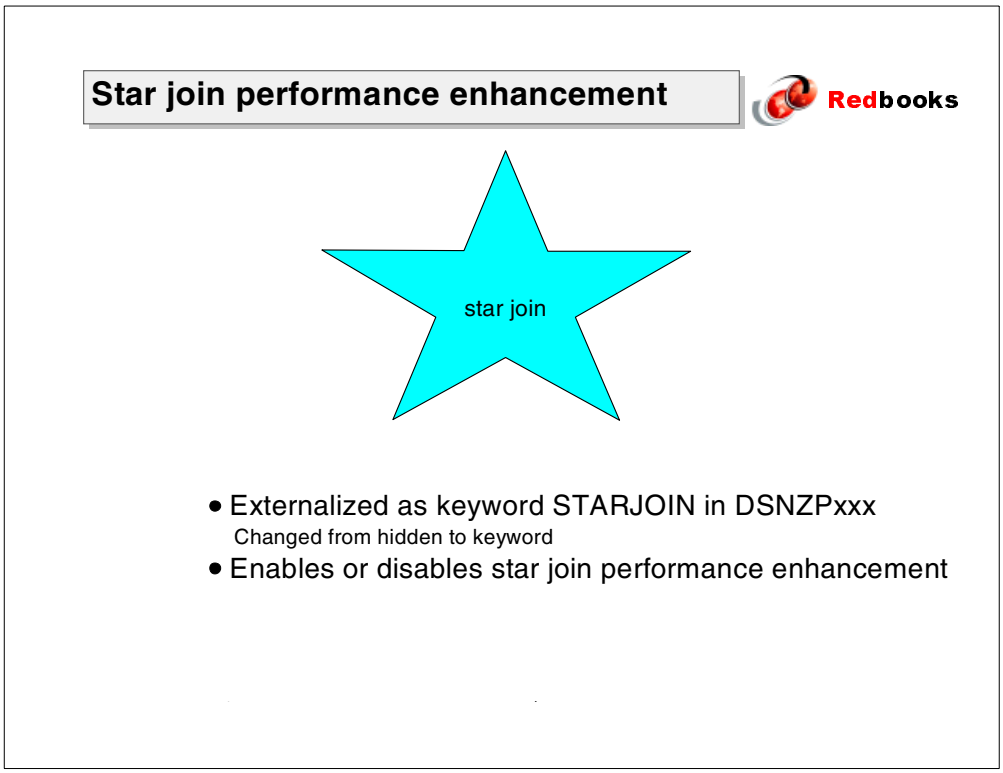
This job runs after installation job DSNTIJSG and calls program DSNTIGR, which does the following:

- Copies the user-maintained tables to the new catalog tables:

| Copy from | To |
|---|---|
| SYSIBM.SYSPSM | SYSIBM.SYSROUTINES_SRC |
| SYSIBM.SYSPSMOPTS | SYSIBM.SYSROUTINES_OPTS |

- Drops database DSNDPSM containing tables SYSIBM.SYSPSM and SYSIBM.SYSPSsMOPTS
- Creates views on the new catalog tables, for data sharing coexistence and fallback:

| Catalog table | View |
|---|---|
| SYSIBM.SYSROUTINES_SRC | SYSIBM.SYSPSM |
| SYSIBM.SYSROUTINES_OPTS | SYSIBM.SYSPSMOPTS |




## 11.10 Star join performance enhancement

A new way of processing multiple-table joins has been added as an option to DB2 V6. This is known as the star join performance enhancement because it is oriented to improve star join performance. A star join consists of several (dimension) tables being joined to a single central (fact) table using the table design pattern known as a star schema. The improvement also applies to joins of tables using the snowflake schema design which involves one or more extra levels of dimension tables around the first level of dimension tables.

In DB2 V7, this parameter is changed from hidden to an externalized keyword. The default value for star join is changed from enabled to disabled.

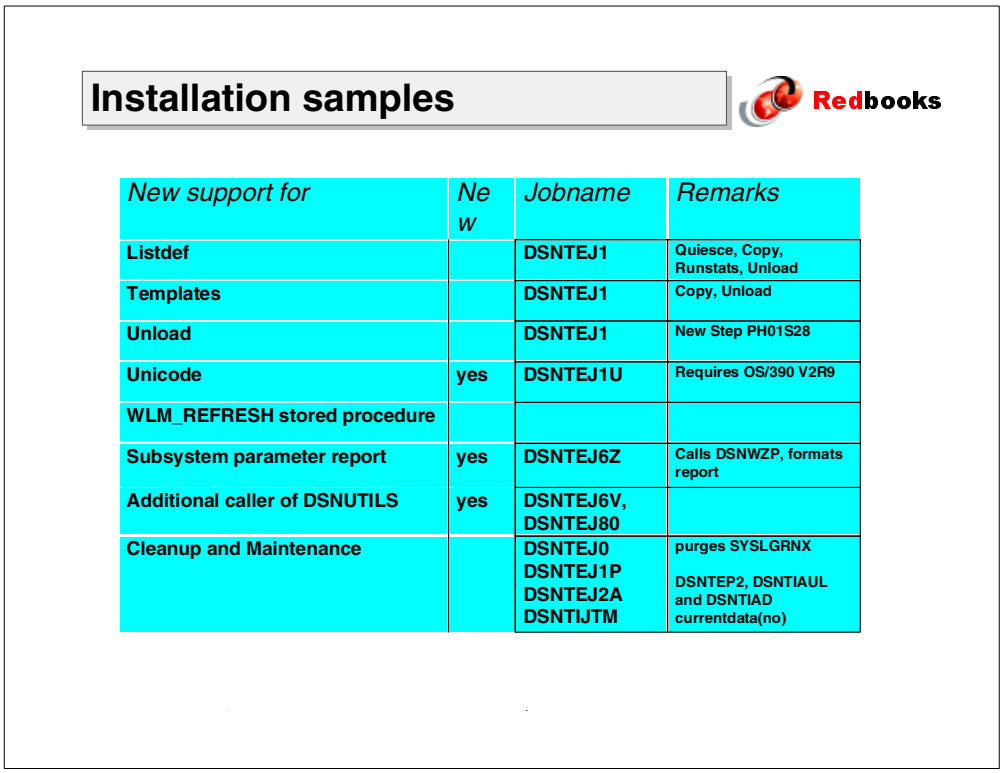
When you want to specify a value deviating from default, you must manually add the keyword STARJOIN to the invocation of the DSN6SPRM macro in the job DSNTIJUZ which assembles and link edits the DSNZPxxx subsystem parameter load module. Acceptable values are ENABLE, DISABLE or 1-32768.

Star join support for DB2 V6 is delivered by the fixes to APARs PQ28813 and PQ36206.

### *STAR JOIN*

| | |
|---|---|
| Acceptable values: | DISABLE, ENABLE, (1, 2-32768) |
| Default: | DISABLE |
| DSNZPxxx | DSN6SPRM STARJOIN |
| | |
| DISABLE | No star join |
| ENABLE | Enable star join - DB2 will optimize for star join |
| 1 | The fact table will be the largest table in star join query. No fact/dimension ratio checking is done. |
| 2-32768 | This is the star join fact table and the largest dimension table ratio. |

## 11.11 Installation samples




The sample applications used to verify either installation or migration have been adapted to verify and demonstrate new advantages of DB2 V7.

***Utilities lists and dynamic allocation (templates)***
Installation verification job DSNTEJ1 has been adapted to verify the new advantage of using dynamic allocation of data sets and processing of dynamic lists of DB2 objects, introduced as new utility control statements, LISTDEF and TEMPLATE.

LISTDEF control statements are invoked by the Quiesce, Copy, and Unload utilities; TEMPLATE by the Copy and Unload utilities.

```
LISTDEF DSN8LDEF
        INCLUDE TABLESPACES DATABASE DSN8D71A
        EXCLUDE TABLESPACE           DSN8D71A.DSN8S71R
        EXCLUDE TABLESPACE           DSN8D71A.DSN8S71S
TEMPLATE DSN8TPLT
         DSN(DSN710.SYSCOPY.&DB..&TS.)
         DISP (NEW,CATLG,DELETE)
         UNIT SYSDA VOLUMES(DSNV01)
         PCTPRIME 100 MAXPRIME 5 NBRSECND 10
    COPY LIST DSN8LDEF
         COPYDDN(DSN8TPLT)
```

For additional information about LISTDEF and TEMPLATES, please refer to 5.3, “Dynamic utility jobs” on page 194.

***Unload utility***
The new Unload utility introduced with DB2 V7 is called by a new step in installation verification job DSNTEJ1. It unloads by partition level and is coded using LISTDEF and TEMPLATE to provide additional use of those new utility enhancements.

```
 LISTDEF DSN8LDUL
         INCLUDE TABLESPACE DSN8D71A.DSN8S71E PARTLEVEL
         EXCLUDE TABLESPACE DSN8D71A.DSN8S71E PARTLEVEL(2)
TEMPLATE DSN8TPPU
         DSN(DSN710.&DB..&TS..SYSPUNCH)
         DISP(NEW,CATLG,DELETE)
         UNIT SYSDA VOLUMES(DSNV01)
         PCTPRIME 100 MAXPRIME 1 NBRSECND 1
TEMPLATE DSN8TPSY
         DSN(DSN710.&DB..&TS..P&PART.)
         DISP(NEW,CATLG,DELETE)
         UNIT SYSDA VOLUMES(DSNV01)
         PCTPRIME 100 MAXPRIME 5 NBRSECND 10
  UNLOAD LIST DSN8LDUL
         PUNCHDDN(DSN8TPPU)
         UNLDDN(DSN8TPSY)
         EBCDIC
         NOPAD
```

For additional information about the Unload utility, please refer to 5.4, “A new utility - UNLOAD” on page 232

***UNICODE***
The new job DSNTEJ1U creates a database, table space, and table with CCSID UNICODE. It loads data into the table from a data set containing a full range of characters in an EBCDIC Latin-1 code page, which results in a mix of single and double-byte characters in the UNICODE table. Then it runs several selects on the table to display the data in hex format (UNICODE ==> EBCDIC)

Note that this job requires OS/390 V2R9 or subsequent release for CCSID handling.

For additional information about UNICODE, please refer to 6.3, “UNICODE” on page 297

***WLM_REFRESH stored procedure***
DSNTWR uses MGCRE macro to pass refresh command to MVS, so it must reside in an APF-authorized library

DSNTWR uses the SAF RACROUTE macro to enable OEM security as well as RACF

The refresh is performed only if the current SQLID has READ access or higher to the SAF resource profile <ssid>.WLM_REFRESH.<wlm-environment-name> within SAF resource class DSNR

The sample job DSNTEJ6W includes a step showing how to create and permit access to the SAF profile

***Subsystem parameter report***
The new job DSNTEJ6Z prepare and invoke the sample program DSN8ED7 which is a sample caller of the stored procedure DSNWZP, a DB2 provided stored procedure that returns the current settings of your DB2 subsystem parameters. DSN8ED7 formats the results from DSNWZP in a report format and prints it. See appendix Appendix A, “Updatable DB2 subsystem parameters” on page 517 for a sample output of DSN8ED7.

Please note that before running DSN8ED7 the stored procedure DSNWZP must exist on your DB2 subsystem. Installation job DSNTIJSG creates and binds the DB2 provided stored procedure DSNWZP.

For additional information about online subsystem parameters, please refer to 7.9, “Online subsystem parameters” on page 342.

***Additional callers of DSNUTILS***
The new job DSNTEJ6V compiles, link-edits, binds, and runs a sample application that demonstrates using an object-oriented C++ program to invoke the DSNUTILS stored procedure to execute a utility. DSNTEJ6V requires a WLM procedure.

The application consists of two sample classes and a sample client:

- DSN8EE0 is an exception class for handling SQL errors.
- DSN8EE1 is a class with constructors and methods for creating and manipulating DSNUTILS as an object.
- DSN8EE2 is a client program that demonstrates using DSN8EE1 to run the DB2 CHECK INDEX utility.

The new job DSNTEJ80 prepares and executes DSN8OD1, a sample application program that demonstrates using ODBC to invoke DSNUTILS, the DB2 Utilities Stored Procedure.

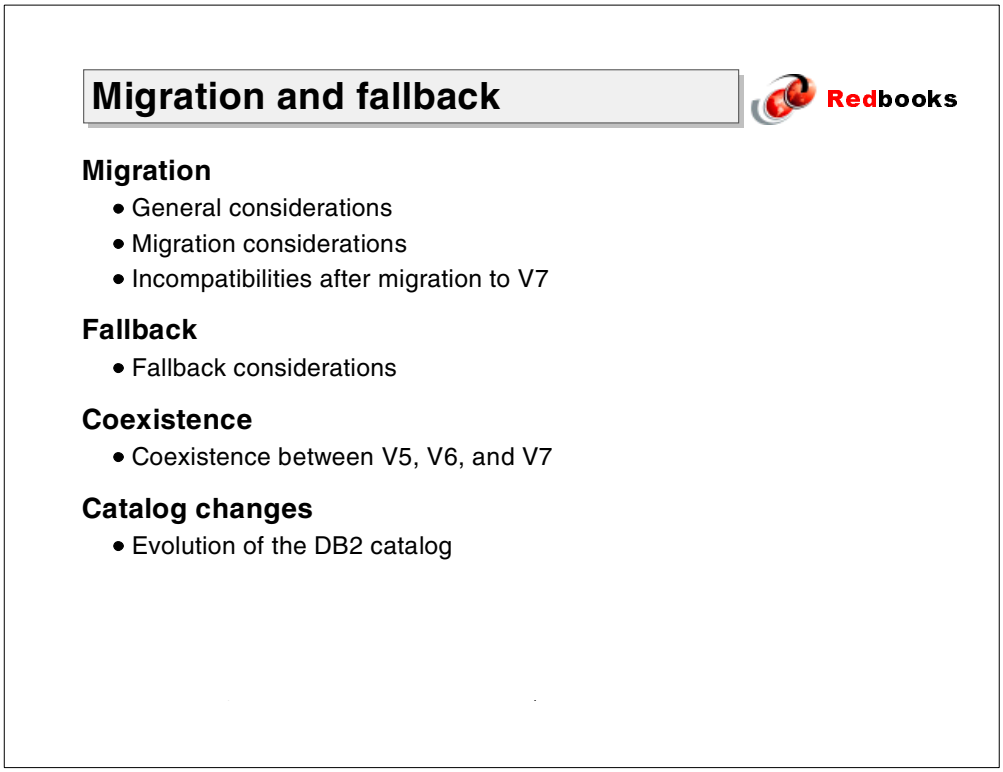
The following are required to run this job:

- DB2 UDB for OS/390 ODBC
- IBM C/C++ for OS/390
- DSNUTILS, the DB2 UDB for OS/390 Utilities stored procedure
- SDSNSAMP(DSNAOINI), the ODBC sample initialization file (INI), which assumes that this member has been customized for your system

***Cleanup and maintenance***
The sample Job DSNTEJ0, which frees all plans, drops all objects, and deletes data sets, now purges SYSLGRNX entries for the sample table spaces before dropping them by invoking the Modify utility.

The sample application programs DSNTEP2, DSNTIAUL, and DSNTIAD are now bound with CURRENTDATA(NO). This forces block fetch, allows lock avoidance, and reduces Coupling Facility access in data sharing.

# Chapter 12. Migration and fallback




Migration is the process of converting the DB2 catalog and directory from a previous release to an updated or current release. This catalog update makes new functions of the updated or current release available without the loss of any data from the previous release and without the need to convert user data.

Fallback is the process of returning to a supported previous release of DB2 after a successful migration of the catalog and directory to a new release.

DB2 V7 supports migration and fallback from and to either DB2 V5 or DB2 V6.

Coexistence of multiple releases of DB2 is of particular interest in a data sharing environment: the objective is that a data sharing group can be continuously available for any planned reconfiguration, including release migrations.

A sound migration plan is necessary when positioned on a previous version and considering a migration to current releases.

DB2 V3 became generally announced (GA) at the end of 1993, and so it is almost eight years old. It was withdrawn from marketing in February 2000, and the end of service is March 2001. Normal life expectancy for a release is about five years, but the Y2K issues extended the period. V4 has been withdrawn from marketing on December 1, 2000. The next step will be end of service at the end of 2001.

© Copyright IBM Corp. 2001

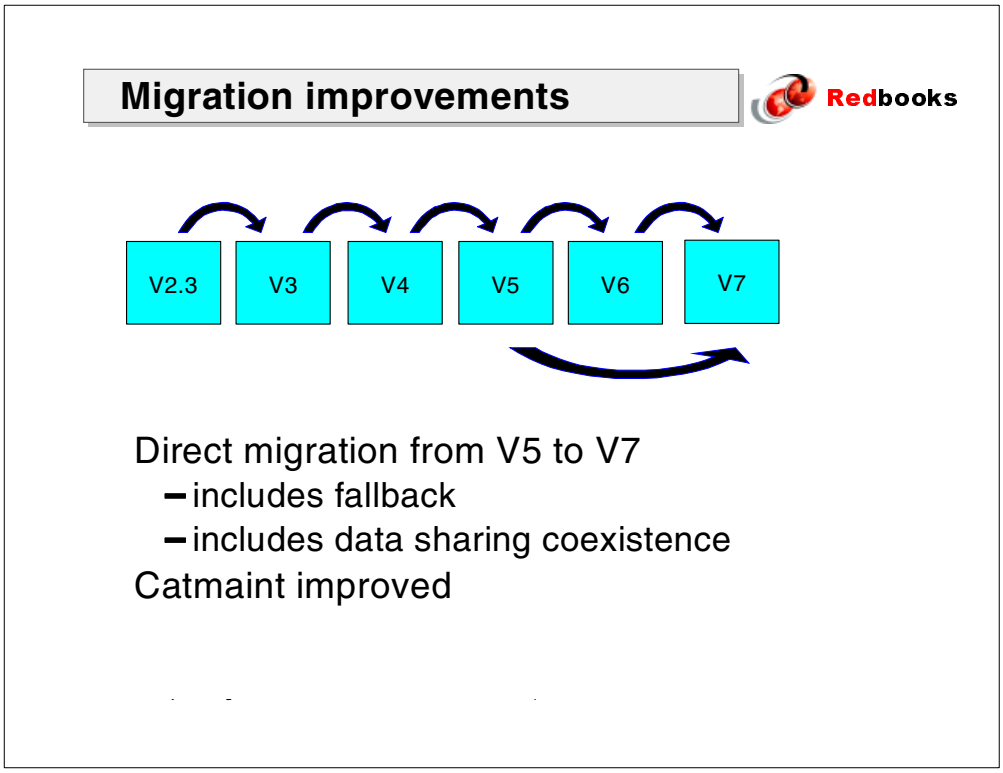
If you are on V4, you need to stay current and plan to migrate to V5 before December 2001. Then you can skip to V6 or V7 by 2002. If you are already on V5, you can start evaluating the best timing for your migration either to V6 or V7. Do not assume that skipping one release will save one migration period; there will be quite a bit more planning and more testing associated with a skip migration.

Running a wide variation of software dates means you are more likely to find a problem. Running very old software means you are less able to get a resolution. If you are beyond the end of service and need a fix, then you need to pay for the custom work, as this is not a standard offering.

The service status of each DB2 version is reported on the Web site:

`http://www.ibm.com/software/data/db2/os390/availsum.html`

It is important to highlight that this area of functions is directly related to maintenance and it is subject to change up to the actual shipment of the product, related maintenance, and their prerequisites.




## 12.1 Migration improvements

With DB2 V7, migration is supported from V5 or V6.

Migration, fallback, and data sharing coexistence with the ability to skip a release offer many possibilities. Most customers are already on V5 or V6 now. Those on V5 who need the capabilities of V6 as soon as possible will not wait. Other customers may wait, plan properly, and skip over V6, going directly to V7.

The **Catmaint utility** has been largely improved, and the execution will be faster. The V7 Catmaint is a three-step process:

1. Mandatory catalog processing:
   - Authorization check
   - Ensure catalog is at correct level
   - DDL processing
   - Additional processing and tailoring
   - Directory header page and BSDS/SCA updates
   - Single commit scope. it is all or nothing
   - No table space scan
2. Looking for unsupported objects in the DB2 catalog:
   - Type 1 indexes
   - Dataset passwords
   - Shared read-only data
   - Syscolumns zero records

The migration will not fail if any of these unsupported objects are found. A message will be issued for each unsupported object found.

There is a SYSDBASE table space scan in this step 2.

3. Stored procedure migration processing:

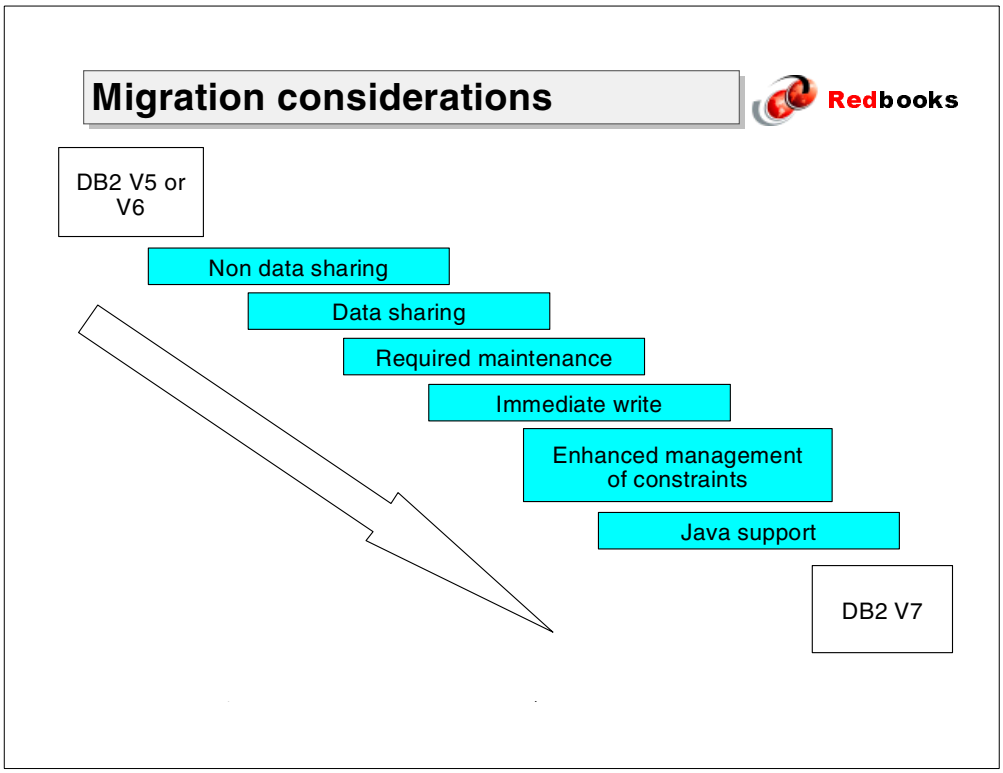
   Catalog table SYSIBM.SYSPROCEDURE is no longer used to define stored procedures to DB2. All rows in SYSIBM.SYSPROCEDURE are migrated to SYSIBM.SYSROUTINES and SYSIBM.SYSPARMS

   When migrating from V5, DB2 generates CREATE PROCEDURE statements which populate SYSIBM.SYSROUTINES and SYSIBM.SYSPARMS. Rows in SYSIBM.SYSPROCEDURES that contain non-blank values for columns AUTHID or LUNAME are not used to generate the CREATE PROCEDURE statements. DB2 also copies rows in SYSIBM.SYSPROCEDURES into SYSIBM.SYSPARMS and propagates information from the PARMLIST column of SYSIBM.SYSPROCEDURES

***Direct migration from DB2 V5 to DB2 V7***

Be aware that when you are migrating directly from V5 to V7, you have to take care of all the considerations related to migrating to DB2 V6, plus several new ones related to V7.

For a complete and detailed list of the DB2 V6 migrations considerations, please refer to the *DB2 UDB for OS/390 Version 6 Installation Guide,* GC26-9008-01. The redbooks *DB2 Server for OS/390 Version 5 Recent Enhancements - Reference Guide*, SG24-5421, *DB2 UDB Server for OS/390 Version 6 Technical Update*, SG24-6108, and *DB2 UDB for OS/390 Version 6 Performance Topics,* SG24-5351 can also be of assistance in evaluating the wide range of functions involved in the migration.




## 12.2 Migration considerations

This section is meant to provide a brief summary of topics to consider when migrating to DB2 V7. You must consult the updated official documentation available with the product or from the Web:

- *DB2 UDB for OS/390 and z/OS Version 7 Release Planning Guide*, SC26-9943
- *DB2 UDB for OS/390 and z/OS Version 7 Installation Guide*, GC26-9936
- Program directories
- Preventive service planning

Migration to DB2 V7 is supported from both V5 and V6. In this section we assume that you are migrating from V6. If you are migrating from V5 you must also refer to the DB2 V6 documention. Before starting to migrate to V7, make sure that the release being migrated from, either V5 or V6, is at the proper service level to allow for a fallback. Then review the information APARs and get the DB2 hipers (highly pervasive) installed on your target system. Do not activate major new functions until you are satisfied with your regression testing.

The migration from V6 to V7 should be simple when compared to the migration from V5 to V6. There are only minor incompatibilities that are documented in detail in the *DB2 UDB for OS/390 and z/OS Version 7 Installation Guide*, GC26-9936.

### 12.2.1 Non-data sharing

At DB2 startup time, the code level of the starting DB2 will be checked against the code level required by the current DB2 catalog. If the starting DB2 has a code level mismatch with the catalog, then an error message will be issued and DB2 will not start. A code level mismatch indicates that the starting DB2 is at a level of code that is down-level from what it needs to be at for the current catalog.

If the catalog has been migrated to V7, then the starting DB2 must be at V7, or at a release for which a migration to V7 is supported with the appropriate fallback SPE on.

Before attempting to migrate to V7, it is recommended that maintenance through the V7 fallback SPE be put on your DB2 subsystem prior to migration. This will most likely eliminate the need to apply maintenance before the fallback release is started in a fallback situation. If the fallback SPE is not on the DB2 subsystem before a migration to a later DB2 release, then the fallback SPE will need to be applied before DB2 is allowed to start in the fallback release.

### 12.2.2 Data sharing

At DB2 startup time, the code level of the starting DB2 will be checked against the code level required by the current DB2 catalog and against the code level of the other DB2s that are running. If the starting DB2 has a code level mismatch with the catalog or any of the other DB2s that are running, then a message will be issued, most likely DSNR041E or DSNX208E, and DB2 will not start. A code level mismatch indicates that the starting DB2 is at a level of code that is down-level from what it needs to be at for the current catalog or that one or more of already running DB2s are down-level from where they need to be.

Before attempting to migrate to V7, all started DB2 subsystems must have maintenance through the V7 fallback SPE on before any attempt is made to migrate to V7. If the appropriate code level and/or fallback SPE is not on all group members, then DB2 V7 will not start and you will not be able to attempt the migration. Message DSNR041E or DSNX208E will be issued in these cases.

We recommend that only one DB2 subsystem be started at V7 for migration processing. Once the migration to V7 completes, the other group members can then be brought up to V7 at any time.

During a migration to V7, the other group members may be active. Catmaint processing will get the locks necessary for the processing. The other active group members may experience delays and/or time-outs if they try to access the catalog objects that are being updated or locked by migration processing.

### 12.2.3 Required maintenance

Verify that you have installed all the required maintenance on your system before you start to migrate.

Start with the Information APARS:

```
II06683: INDEX TO DB2 INFORMATIONAL APARS
II12653: DB2 V6.1 MIGRATION/FALLBACK INFOAPAR TO/FROM DB2 V7.1 AND UPGRADING
R610
II12652: DB2 V5.1 MIGRATION/FALLBACK INFOAPAR TO/FROM DB2 V7.1 AND UPGRADING
R510
```

#### 12.2.3.1 Fallback SPE

It is recommended to install the following fallback and toleration maintenance. In case of data sharing, you must install this maintenance to all members of the data sharing group before migration to V7. This list of maintenance is only current as of the date of this writing:

- PQ36419 (UQ44220 for V5 and UQ44221 for V6), code to support future new functions
- PQ36757 (UQ45555 for V5 and UQ45556 for V6), code to support future new functions
- PQ37708 (UQ45789 for V5 and UQ45790 for V6), coexistence and fallback support for online reorg without rename
- PQ37762 (UQ45514 for V5 and UQ45515 for V6) code to support future new functions
- PQ38504 (UQ45884 for V5 and UQ45885 for V6), handles fallback for future new functions
- PQ38746 (UQ45721 for V5 and UQ45722 for V6), fallback and coexistence support for utilities
- PQ39199 (UQ45929 for V6), fallback and coexistence
- PQ40356 (UQ46006 for V5 and UQ46007 for V6), fallback and coexistence
- PQ40446 (UQ46008 for V5 and UQ46009 for V6), fallback and coexistence
- PQ40796, (UQ47687 for V5 and UQ47688 for V6), fallback and toleration
- PQ41011, (UQ46676 for V5 and UQ46677 for V6), prerequisite for future functions
- PQ34467, (UQ90024 for V5 and UQ90025 for V6), fallback SPE which prerequisites all fallback APARs and points to all of them
- PQ45557, (UQ51277 for V5 and UQ51278 for V6), reassembly and re-linkedit of DSNHDECP required after the SPE.

#### 12.2.3.2 Coupling Facility Control Code service level

If you use data sharing, your Coupling Facility Control Code (CFCC) needs to be at least on either of these service levels:

- CFCC Release 7 service level 1.06
- CFCC Release 8 service level 1.03

For further information about Coupling Facility Control Code level, please refer to 8.1, “Coupling Facility Name Class Queues” on page 383.

### 12.2.4 Immediate write

On initial migration to V7, the IMMEDWRITE column of SYSIBM.SYSPLAN and SYSIBM.SYSPACKAGE defaults to “blank” for all plans and packages. For migrated plans and packages, V7 correctly picks up the IMMEDWRITE specification from information that is recorded within the plan or package itself. The IMMEDWRITE catalog column will be populated with non-blank values as plans and packages are bound or rebound on V7 or above.

### 12.2.5 Enhanced management of constraints

DB2 V6 may have existing primary key and unique key constraints. Customers should not have any SYSCOLUMNS ZERO records (for unique key constraints without enforcing indexes) when migrating. The following query determines if you have any existing SYSCOLUMNS ZERO records to fix before attempting to migrate:

```
SELECT COLNO, TBCREATOR, TBNAME, REMARKS FROM SYSIBM.SYSCOLUMNS
 WHERE COLNO = 0;
```

For each table identified in the results from the select, you can:

- Drop the table if the table is not needed.
- Try to complete the definition of the identified table. Take a look at the REMARKS field for each table identified in the select result. The REMARKS field will list all the column numbers of all unique key constraints that do not have enforcing indexes. Each constraint column set is separated by a comma, and each column number within a constraint is separated by a space. A unique index needs to be created for each constraint that is listed in the SYSCOLUMNS ZERO record.

  For further information about enhanced management of constraints, please refer to 2.2, “Enhanced management of constraints” on page 38.
- Proceed with migration. After migration is completed, on DB2 V7, unload all data from the table, drop the table, and recreate the table with the desired constraints and indexes.

On migration to DB2 V7, a check will be done for the existence of SYSCOLUMNS ZERO records by doing a non-matching index scan on SYSCOLUMNS. If a SYSCOLUMNS ZERO record is found, warning message DSNU776 will be issued.

If customers still have SYSCOLUMNS ZERO records after migrating to V7, the associated tables will remain in incomplete status. V7 will not recognize whether an index being created is an enforcing index or not. To be able to fix the status of the table, follow the 3rd method above.

Please note that the following DDLs for a particular table should all be executed on the same release. Do not execute some of the DDLs on DB2 V7 and some of the DDLs on DB2 V6. Results will be unpredictable.

- CREATE TABLE PRIMARY KEY
- CREATE TABLE UNIQUE
- CREATE UNIQUE INDEX (to enforce primary key)
- CREATE UNIQUE INDEX (to enforce unique key)
- DROP INDEX (drop index enforcing primary key)
- DROP INDEX (drop index enforcing unique key)

- DROP INDEX (drop index enforcing referential constraint)
- CREATE TABLE FOREIGN KEY
- ALTER TABLE DROP PRIMARY KEY
- ALTER TABLE DROP UNIQUE
- ALTER TABLE DROP FOREIGN KEY

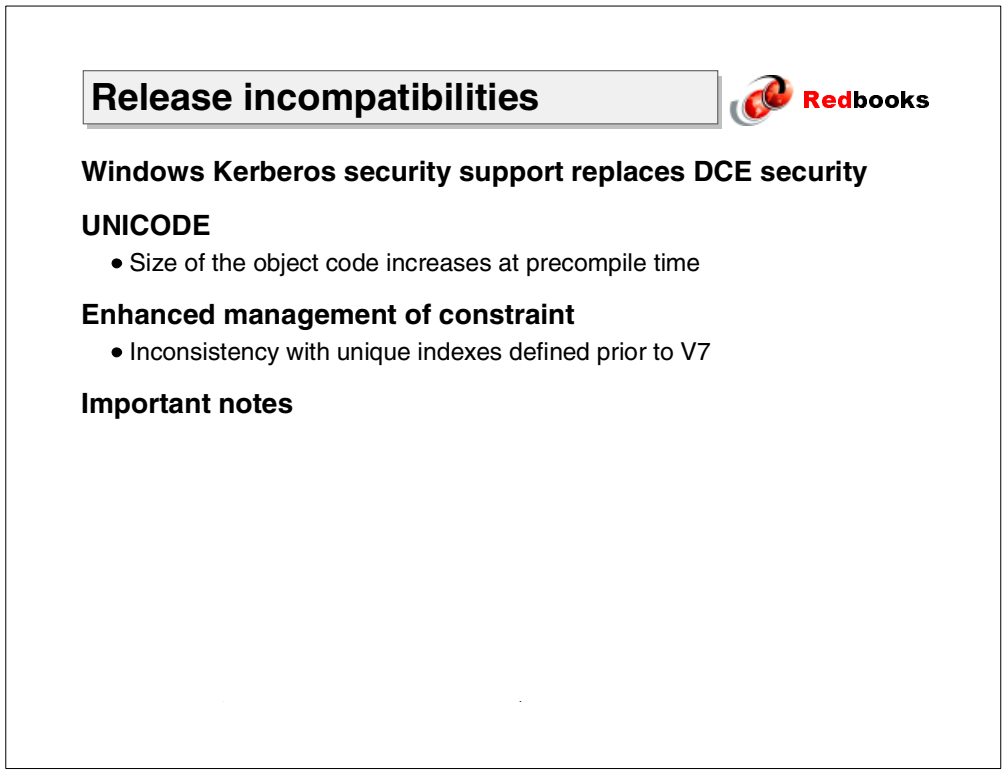
To take advantage of the new enhanced management of constraints, we would recommend that you add a unique constraint for all of your unique indexes by altering the related table with the alter table add constraint statement. This will assure that your unique indexes, which mostly are created in dependency and as part of the application logic, cannot be dropped accidentally.

### 12.2.6 Java support

DB2 V7 JDBC support uses RRSAF to connect to DB2, rather than CAF. Therefore Resource Recovery Services (RRS) must be set up before JDBC 2.0 can be used. JDBC 2.0 also has a prerequisite of JDK 1.3, which is a part of OS/390 2.8.

Stored procedure and user-defined functions with language JAVA cannot be defined or used in prior releases. With DB2 V7 JAVA is supported for stored procedures and user defined functions and JARs are introduced as new objects. With this JAR becomes a new reserved word.

For more information about Java support refer to 3.4, “DB2 Java support” on page 104 and 3.5, “Java stored procedures and Java UDFs” on page 124.




## 12.3 Release incompatibilities

In this section we discuss items that have incompatibilities after a migration to DB2 V7. Again, check the current documentation.

### 12.3.1 Windows Kerberos security support

Support for DCE security authentication is being removed in favor of Kerberos authentication. Users migrating from prior releases will no longer be able to utilize DCE security authentication.

### 12.3.2 UNICODE

Because of the availability of CCSIDs for host variables in V7, the host language statements generated by the precompiler for each PREPARE or EXECUTE IMMEDIATE statement may be larger. As a result, the size of the object code that results from compiling the output of the precompiler increases. This increase varies according to the number of PREPARE or EXECUTE IMMEDIATE statements.

### 12.3.3 Enhanced management of constraints

There is a new restriction disallowing the dropping of an index that is used to enforce a unique constraint (primary key or unique key) or referential constraint. If an attempt is made to drop an index enforcing one of these constraints, SQLCODE -669 will be issued.

Before an index that enforces a unique or referential constraint can be dropped, the constraint must first be dropped with an ALTER TABLE statement. Note that dropping a unique constraint cascades to dropping all dependent referential constraints.

An exception to the restriction above is made for any unique keys that were created before DB2 V7. Since there is no way to drop a unique key constraint created prior to DB2 V7, the enforcing index can be dropped without first dropping the unique key constraint. The unique key constraint is implicitly dropped when the enforcing index is dropped.

Before migration, to identify the indexes that will be restricted in V7, which include indexes that enforce primary key constraints and referential constraints, run the following query:

```
SELECT CREATOR, NAME, TBCREATOR, TBNAME, UNIQUERULE
 FROM SYSIBM.SYSINDEXES
 WHERE UNIQUERULE IN ('P','R');
```

After migration, to identify the indexes that are restricted, run the following query:

```
SELECT IXS.CREATOR, IXS.NAME, IXS.TBCREATOR, IXS.TBNAME, IXS.UNIQUERULE
FROM SYSIBM.SYSINDEXES AS IXS
 WHERE IXS.UNIQUERULE IN ('P','R')
UNION
SELECT IXS.CREATOR, IXS.NAME, IXS.TBCREATOR, IXS.TBNAME,IXS.UNIQUERULE
FROM SYSIBM.SYSINDEXES AS IXS,SYSIBM.SYSTABCONST AS BC
WHERE BC.TYPE = 'U'
AND IXS.CREATOR = BC.IXOWNER
AND IXS.NAME = BC.IXNAME;
```

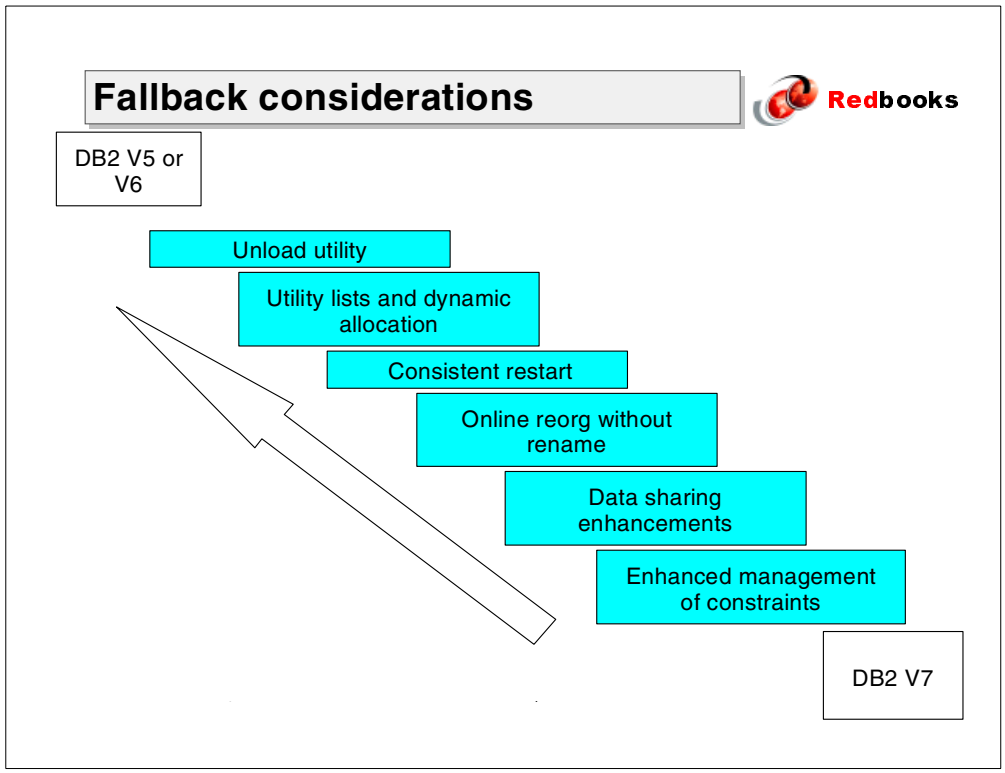
Once an index has been identified as being restricted, the constraint being enforced by the index must first be dropped before the index can be dropped.

### 12.3.4 Important notes

Catmaint step 1 process is all or nothing. All Catmaint processing will be rolled back with the exception of those indexes that were altered during catmaint processing:

- DSNAUH01
- DSNATX02
- DSNDXX02
- DSNVVX01
- DSNVTH01

If Catmaint processing fails and the indexes had already been altered these indexes must be rebuild.




## 12.4 Fallback considerations

The process of returning to a supported previous release of DB2 after a successful migration of the catalog and directory to DB2 V7 (falling back) is called as *fallback*. DB2 will support a fallback to either DB2 V5 or DB2 V6 after a successful migration to DB2 V7.

Prerequisite changes to the fallback release should be installed before migrating to DB2 V7 to ensure fallback to DB2 V5 or DB2 V6. See also 12.2.3, “Required maintenance” on page 501.

The SPE (PQ34467) must be on all members in a data sharing group before a migration to V7 is attempted.

In a non-data-sharing subsystem, the SPE must be on before fallback is allowed.

Fallback to DB2 V5 is only possible when the migration to V7 was done from V5.

Fallback to DB2 V6 is only possible when the migration to V7 was done from V6.

Falling back does not undo changes made to the catalog and directory during a migration to V7. The migrated catalog is used after fallback. Some objects in this catalog that have been affected by function in this release might become frozen objects after fallback. Frozen objects are unavailable, and they are marked with a release dependency indicator.

This section is meant to be only a quick summary of items to take care when falling back from V7.

### 12.4.1 Unload utility

DB2 V7 introduces the new Unload utility that can not be started/restarted from a prior release. An attempt to start/restart the Unload utility from a prior release causes a syntax error (as a non-existing utility name).

-TERM UTIL command can not be issued to a stopped Unload utility from a prior release (no code change will be involved). As an operational remark, any stopped Unload utility should be termed before fallback occurs; otherwise, SYSUTIL records for such Unload utility jobs will remain in the system.

### 12.4.2 Utility lists and dynamic allocation

Utility lists, dynamic allocations and consistent restart requires additional information to be check pointed in the SYSUTILX table. Therefore a paused utility which was executed on a V7 system can not be restarted from a prior level system. Utilities can only be restarted on the release they originated.

### 12.4.3 Consistent restart

Falling back to V6 with the objects remaining in refresh pending status (REFP), new with DB2 V7, will make those objects completely inaccessible in V6 or V5. Any attempt to access these objects will result in DB2 issuing resource unavailable message, DSNT501I, with the reason code of 00C900CE. The only way to access the object is to first issue the -START DB ACCESS(FORCE) command to remove all the exception states and the release dependency value from the DBET. Please note that the object will be in an inconsistent state. You must Recover to a point in time or run the Load Replace utility before accessing the object for any SQL activity. Thus, it is recommended that REFP state of the objects be resolved prior to falling back if possible.

### 12.4.4 Online reorg without rename

If fallback occurs after the execution of Online REORG, where objects have been renamed with the 'J0001' instance node, The V5/V6 migration/ fallback PTF will allow the access of these objects.

After fallback, subsequent executions of REORG SHRLEVEL CHANGE or REFERENCE, operating on objects named with the 'J0001' instance node will remain named with a 'J0001' instance node. The SWITCH phase will process in the same manner as in V6. The original object, in this case with the 'J0001' instance node, will be renamed to a 'T0001' instance node, then depending upon the object characteristics, this object will either be renamed to 'I0001' or deleted. The shadow object that is named with the 'I0001' instance node, will be renamed to a 'J0001' instance node. From that point on, as long as the customer remains on the fallback level of DB2, that object will remain named with the 'J0001' instance node. After the customer migrates forward to V7, subsequent REORG SHRLEVEL CHANGE or REFERENCE utilities will resume the V7 renaming scheme.

We recommend that prior to fallback, all utilities should either be TERM'ed or allowed to complete. A -DIS UTIL(*) command should be submitted to the system to identify all utilities in a stopped state. Those utilities that are in a stopped state should be -TERM'ed. In the event that a stopped V7 REORG SHRLEVEL CHANGE or REFERENCE can not be -TERM'ed on V7, the V6 -TERM command will be able to cleanup the utility, after fallback.

The -TERM UTIL command, when executed to cleanup after a failed REORG SHRLEVEL CHANGE or REFERENCE, has different processes depending upon the phase where the failed utility stopped. Prior to the SWITCH phase, the objects with the 'J0001' instance node are deleted. Under a fallback scenario, the original objects with a 'J0001' instance node will remain. If the stopped phase is the SWITCH phase, data set synchronization and cleanup will operate as it does for V6, and if the original object at the start of the utility was named with a 'J0001' instance node, the utility termination process will leave the original object, named with the 'J0001' instance node.

For details on the online reorg enhancements, please refer to 5.7, “Online Reorg enhancements” on page 257.

### 12.4.5 Data sharing enhancements

For plans and packages that are bound on V7, the IMMEDWRITE specification is correctly picked up by V5 or V6 members when the plan or package is executed (and subsequently auto rebound). When a V7 plan or package is rebound or auto rebound on a V5 or a V6 member, the downlevel member will set the IMMEDWRITE column to 'blank'.

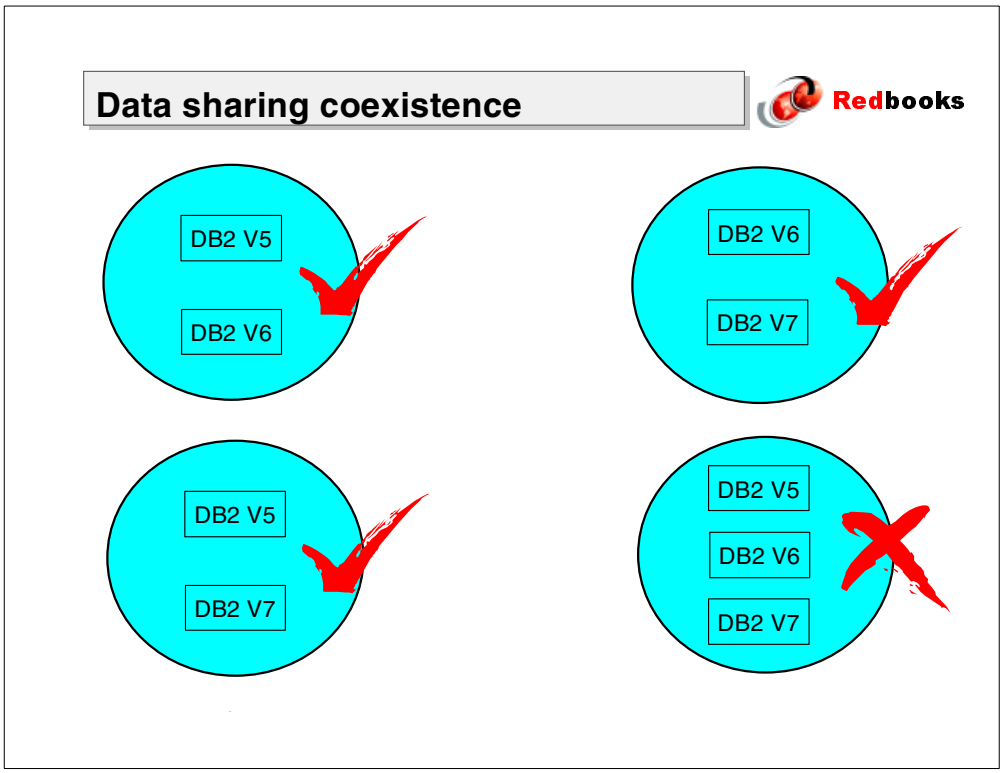
### 12.4.6 Enhanced management of constraints

If a table is created on DB2 V7 and it is in an incomplete state due to missing index on a unique constraint, do not attempt to complete the table definition on a prior release of DB2, V5 or V6. The definition must be completed on DB2 V7. The following DDLs are disallowed (with SQLCODE -904) on the prior releases for an incomplete table created on V7:

- ALTER TABLE
- CREATE INDEX
- DROP INDEX

Note that for a table with complete definition, DDL created on V7 follows V7 semantics, and DDL executed on the prior releases follows the semantics of the prior releases.

As with all new syntax, plans and packages that contain new syntax will be marked with a release dependency.




## 12.5 Data sharing coexistence

With data sharing groups, you can have V5 and V6, V5 and V7, or V6 and V7.

In a data sharing environment coexistence of multiple releases of DB2 is particularly of interest. The objective is that a data sharing group can be continuously available for any planned reconfiguration, including release migrations. So, if you have a data sharing group consisting of all V5 or V6 DB2 members, you can migrate your group to V7 by “rolling in” the release migration one member at a time (similar to the way you would roll in a PTF for an APAR fix), thus keeping at least a subset of the DB2 members available at all times.

During the “rolling in” process, you will have a period of time where there are either V5 and V7 members or V6 and V7 members coexisting within the same data sharing group. DB2 does not support the coexistence of more than two releases of DB2 at a time. During this period of coexistence, the new V7 function may or may not be available to the down-level members.

Before migrating from either V5 or V6 to V7, you must have the fallback SPE (APAR PQ34467) on before. This is enforced for data sharing:

- Information is kept in the BSDS/SCA that is checked at DB2 startup time to ensure that all group members have the SPE on.
- A new starting member’s code level will be checked to ensure that it too can coexist with the current catalog.
- In optional Catmaint cases (V5 and V6) DB2 will also ensure that all members have SPE on before allowing Catmaint processing to proceed.

You cannot have coexistence between more than 2 releases of DB2 at any time within the same data sharing group. For DB2 V5, V6. and V7. this means:

- You can have V5 and V7 together.
- Or you can have V6 and V7 together.
- **You cannot have V5, V6, and V7 together.**

***Persistent structure size changes***
In a coexistence environment, the size of CF structure allocations may be unpredictable. The reason is that the first connector to a new structure is the one that generates the allocation of the structure. If this member does not have the persistent structure size code on, the structure will always be allocated at the INITSIZE.

***Cross-system restart***
In a coexistence environment, if a member has ARM enabled for IRLM, and if that member does not have the IRLM maintenance to support Restart Light, then a cross-system ARM restarted IRLM (when DB2 is starting in light mode) may not have the advantage of lower CSA storage due to a forced PC=YES setting.

***Enhanced management of constraints***
In coexistence, you must create unique key constraints, create enforcing indexes, and drop enforcing indexes in the same release (all done in V5 or V6, or all done in V7). You should not cross releases. For example, if you create a unique key constraint on one release, and create the enforcing index on the other release, the index will not be recognized as an enforcing index, and the table will not be set to a complete status (hence the table will not be available for insertions or loads). If this occurs, you need to drop the index and create the index on the correct release.

The same considerations reported under fallback in section 12.3.3, “Enhanced management of constraints” on page 504 apply.

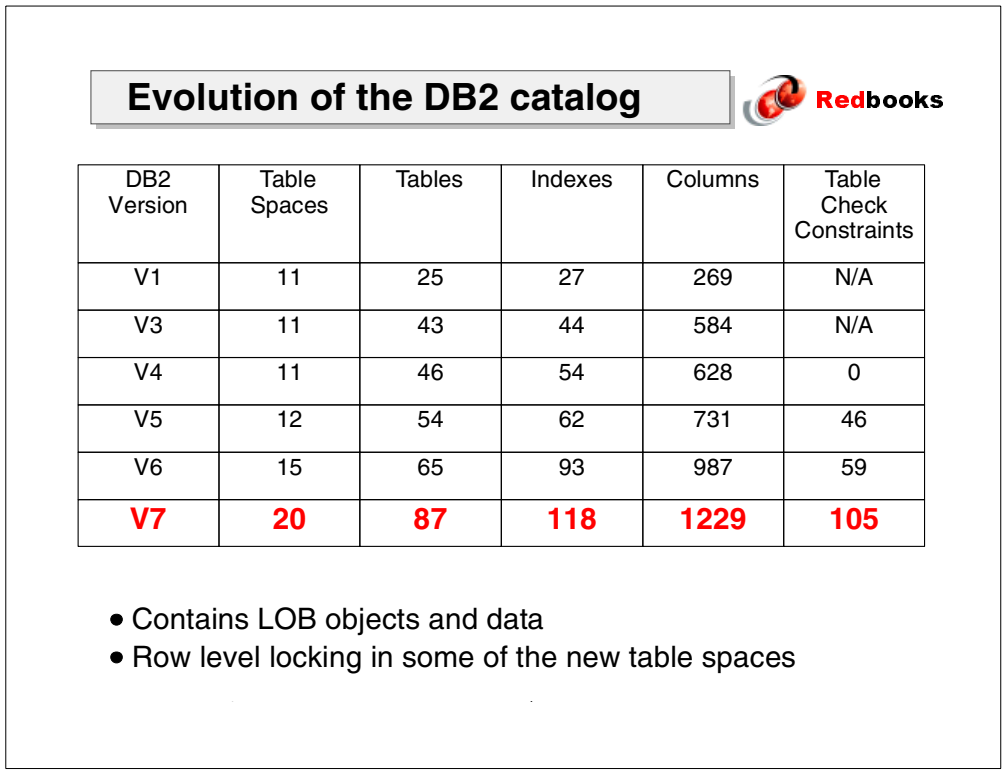
The following DDL statements for a particular table should all be executed on the same release. Do not execute some of the DDL statements on DB2 V7 and some of them on DB2 V6 or V5:

- CREATE TABLE PRIMARY KEY
- CREATE TABLE UNIQUE
- CREATE UNIQUE INDEX (to enforce primary key)
- CREATE UNIQUE INDEX (to enforce unique key)
- DROP INDEX (drop index enforcing primary key)
- DROP INDEX (drop index enforcing unique key)
- DROP INDEX (drop index enforcing referential constraint)
- CREATE TABLE FOREIGN KEY
- ALTER TABLE DROP PRIMARY KEY
- ALTER TABLE DROP UNIQUE
- ALTER TABLE DROP FOREIGN KEY

***Restrict the execution of packages and plans bound to V7***
Restrict the execution of packages and plans bound on V7 to members of the group that have already migrated.

This restriction serves two purposes:

- If new plans and packages use new functions, you avoid the application errors that can occur if the plan or package tries to execute a SQL statement that is not allowed in V6 or V5.
- You avoid automatic rebind that occurs when any plan or package that is bound on V7 is run on V6 or V5. It also avoids the automatic rebind that occur when a V7 bound plan or package that was rebound automatically on V6 or V5 is later run on V7.




## 12.6 Evolution of the DB2 catalog

The DB2 catalog continues to grow with every DB2 release.

With DB2 V7 the catalog contains LOB objects and data on the following tables:

- SYSIBM.SYSJARDATA (auxiliary table)
- SYSIBM.SYSJARCLASS_SOURCE (auxiliary table)
- SYSIBM.SYSJARCONTENTS
- SYSIBM.SYSJAROBJECTS

Row level locking is activated for the following table spaces:

- SYSIBM.SYSGRTNS
- SYSIBM.SYSJAVA
- SYSIBM.SYSSEQ2

For a complete and detailed list of the DB2 V7 catalog changes, please refer to the *DB2 UDB for OS/390 and z/OS Version 7 SQL Reference,* SC26-9944.

***New installation and migration jobs***

The following are new jobs that have impact on the catalog:

- DSNTIJMP

  Job DSNTIJMP is a new optional migration job. It migrates any SQL procedure definitions in the tables SYSIBM.SYSPSM and SYSIBM.SYSPSMOPTS to SYSIBM.SYSROUTINES_SRC and SYSIBM.SYSROUTINES_OPTS respectively.

- DSNTIJIN

  Installation job DSNTIJIN creates all the new data sets that are required for a new installation or migration of DB2 to V7. The data sets are primarily for new DB2 catalog table spaces and new DB2 catalog indexes. New table spaces include DSNDB06.SYSHIST for catalog statistics history, DSNDB06.SYSGRTNS for SQL stored procedures and DSNDB06.SYSJAVA for Java stored procedures.

- DSNTIJTC

  Migration job DSNTIJTC invokes CATMAINT and converts the DB2 catalog tables from one DB2 version format to another. For DB2 V7, the third step is optional. It converts any data stored in the tables SYSIBM.SYSPSM and SYSIBM.SYSPSMOPTS to SYSIBM.SYSROUTINES_PSM and SYSIBM.SYSROUTINES_OPTS respectively. These table are used by the DB2 UDB Stored Procedure Builder (SPB) tool when generating and implementing SQL stored procedures into DB2 for OS/390.

- DSNTIJRX

  Job DSNTIJRX is a new optional migration job. It binds the REXX language support interface into DB2 for OS/390 and makes it available for use.

# Part 9. Appendices

© Copyright IBM Corp. 2001

# Appendix A. Updatable DB2 subsystem parameters

This appendix includes a list of the DB2 parameters changeable online and an example of the output of the DSN8ED7 report on the current setting of the DB2 subsystem parameter. These lists can change at general availability of DB2 V7, and they could be affected by your maintenance level.

## A.1 List of changeable parameters

The table shows the current list by macro of the changeable DB2 subsystem parameters.

| Macro | Parameters |
|---|---|
| DSN6FAC | RLFERRD |
| DSN6LOGP | ARC2FRST, MAXRTU, DEALLCT |
| DSN6SYSP | IDBACK, IDFORE, CTHREAD, DSSTIME, DLDFREQ, CONDBAT, MAXDBAT, LOBVALA, LOBVALS, STATIME, DBPROTCL, PTASKROL, RLFAUTH, RLFTBL, RLFERR, PCLOSEN, PCLOSET, STORTIME, TBSBPOOL, IDXBPOOL, WLMENV, EXTSEC, STORMXAB, URCHKTH, URLGWTH |
| DSN6GRP | no parameters are changeable |
| DSN6SPRM | AUTHCACH, EDMPOOL, EDMDSPAC, EDMBFIT, MAXRBLK, MINRBLK, IRLMSWT, ABIND, DSMAX, NUMLKTS, CDSSRDEF, NUMLKUS, RECALLD, UTIMOUT, BMPTOUT, PARAMDEG, DLITOUT, RETLWAIT, SEQCACH, SEQPRES, DESCSTAT, RELCURHL, OPTHINTS, DBACRVW, STARJOIN, SUPERRS, STATROLL, MINSTOR |
| DSN6ARV | all parameters are changeable |

© Copyright IBM Corp. 2001

## A.2 Currently active zparms

**DSN8ED7: Sample DB2 for OS/390 Configuration Setting Report Generator**

| Macro Name | Parameter Name | Current Setting | Description/ Install Field Name | Install Panel ID | Fld No. |
|---|---|---|---|---|---|
| DSN6SYSP | AUDITST | 00000000000000000000000000000000 | AUDIT TRACE | DSNTIPN | 1 |
| DSN6SYSP | CONDBAT | 0000000064 | MAX REMOTE CONNECTED | DSNTIPE | 4 |
| DSN6SYSP | CTHREAD | 00070 | MAX USERS | DSNTIPE | 2 |
| DSN6SYSP | DLDFREQ | 00005 | LEVELID UPDATE FREQUENCY | DSNTIPL | 14 |
| DSN6SYSP | PCLOSEN | 00005 | RO SWITCH CHKPTS | DSNTIPL | 12 |
| DSN6SYSP | IDBACK | 00020 | MAX BATCH CONNECT | DSNTIPE | 6 |
| DSN6SYSP | IDFORE | 00040 | MAX TSO CONNECT | DSNTIPE | 5 |
| DSN6SYSP | CHKFREQ | 0000050000 | CHECKPOINT FREQ | DSNTIPL | 6 |
| DSN6SYSP | MON | 11000000 | MONITOR TRACE | DSNTIPN | 9 |
| DSN6SYSP | MONSIZE | 0000008192 | MONITOR SIZE | DSNTIPN | 10 |
| DSN6SYSP | SYNCVAL | NO | STATISTICS SYNC | DSNTIPN | 7 |
| DSN6SYSP | RLFAUTH | SYSIBM | RESOURCE AUTHID | DSNTIPP | 8 |
| DSN6SYSP | RLF | NO | RLF AUTO START | DSNTIPO | 4 |
| DSN6SYSP | RLFERR | NOLIMIT | RLST ACCESS ERROR | DSNTIPO | 6 |
| DSN6SYSP | RLFTBL | 01 | RLST NAME SUFFIX | DSNTIPO | 5 |
| DSN6SYSP | MAXDBAT | 00064 | MAX REMOTE ACTIVE | DSNTIPE | 3 |
| DSN6SYSP | DSSTIME | 00005 | DATASET STATS TIME | DSNTIPN | 8 |
| DSN6SYSP | EXTSEC | NO | EXTENDED SECURITY | DSNTIPR | 11 |
| DSN6SYSP | SMFACCT | 11100011000000000000000000000000 | SMF ACCOUNTING | DSNTIPN | 4 |
| DSN6SYSP | SMFSTAT | 10111000000000000000000000000000 | SMF STATISTICS | DSNTIPN | 5 |
| DSN6SYSP | ROUTCDE | 1000000000000000 | WTO ROUTE CODES | DSNTIPO | 1 |
| DSN6SYSP | STORMXAB | 00000 | MAX ABEND COUNT | DSNTIPX | 4 |
| DSN6SYSP | STORPROC | DB2SSPAS | DB2 PROC NAME | DSNTIPX | 2 |
| DSN6SYSP | STORTIME | 00180 | TIMEOUT VALUE | DSNTIPX | 5 |
| DSN6SYSP | STATIME | 00015 | STATISTICS TIME | DSNTIPN | 6 |
| DSN6SYSP | TRACLOC | 00016 | | | |
| DSN6SYSP | PCLOSET | 00010 | RO SWITCH TIME | DSNTIPL | 13 |
| DSN6SYSP | TRACSTR | 00000000000000000000000000000000 | TRACE AUTO START | DSNTIPN | 2 |
| DSN6SYSP | TRACTBL | 00016 | TRACE SIZE | DSNTIPN | 3 |
| DSN6SYSP | URCHKTH | 000 | UR CHECK FREQ | DSNTIPL | 8 |
| DSN6SYSP | WLMENV | | WLM ENVIRONMENT | DSNTIPX | 6 |
| DSN6SYSP | LOBVALA | 0000002048 | USER LOB VALUE STORAGE | DSNTIP7 | 1 |
| DSN6SYSP | LOBVALS | 0000002048 | SYSTEM LOB VALUE STORAGE | DSNTIP7 | 2 |
| DSN6SYSP | LOGAPSTG | 000 | LOG APPLY STORAGE | DSNTIPL | 5 |
| DSN6SYSP | DBPROTCL | DRDA | DATABASE PROTOCOL | DSNTIP5 | 6 |
| DSN6SYSP | PTASKROL | YES | | | |
| DSN6SYSP | EXTRAREQ | 00100 | EXTRA BLOCKS REQ | DSNTIP5 | 4 |
| DSN6SYSP | EXTRASRV | 00100 | EXTRA BLOCKS SRV | DSNTIP5 | 5 |
| DSN6SYSP | TBSBPOOL | BP0 | DEFAULT BUFFER POOL FOR USER DATA | DSNTIP1 | 1 |
| DSN6SYSP | IDXBPOOL | BP0 | DEFAULT BUFFER POOL FOR USER INDEXES | DSNTIP1 | 2 |
| DSN6SYSP | LBACKOUT | AUTO | LIMIT BACKOUT | DSNTIPL | 10 |
| DSN6SYSP | BACKODUR | 005 | BACKOUT DURATION | DSNTIPL | 11 |
| DSN6SYSP | URLGWTH | 0000000000 | UR LOG WRITE CHECK | DSNTIPL | 9 |
| DSN6LOGP | TWOACTV | 2 | NUMBER OF COPIES | DSNTIPH | 3 |
| DSN6LOGP | OFFLOAD | YES | | | |
| DSN6LOGP | TWOBSDS | 2 | | | |
| DSN6LOGP | TWOARCH | 2 | NUMBER OF COPIES | DSNTIPH | 6 |
| DSN6LOGP | MAXARCH | 0000001000 | RECORDING MAX | DSNTIPA | 10 |
| DSN6LOGP | DEALLCT | 00000:00000 | DEALLOC PERIOD | DSNTIPA | 9 |
| DSN6LOGP | MAXRTU | 00001 | READ TAPE UNITS | DSNTIPA | 8 |
| DSN6LOGP | OUTBUFF | 0000004000 | OUTPUT BUFFER | DSNTIPL | 2 |
| DSN6LOGP | WRTHRSH | 00020 | | | |
| DSN6LOGP | ARC2FRST | NO | READ COPY2 ARCHIVE | DSNTIPO | 13 |
| DSN6ARVP | BLKSIZE | 0000028672 | BLOCK SIZE | DSNTIPA | 7 |
| DSN6ARVP | CATALOG | YES | CATALOG DATA | DSNTIPA | 4 |
| DSN6ARVP | ALCUNIT | TRK | ALLOCATION UNITS | DSNTIPA | 1 |
| DSN6ARVP | PROTECT | NO | ARCHIVE LOG RACF | DSNTIPP | 1 |
| DSN6ARVP | ARCWTOR | NO | WRITE TO OPER | DSNTIPA | 11 |
| DSN6ARVP | COMPACT | NO | COMPACT DATA | DSNTIPA | 15 |
| DSN6ARVP | TSTAMP | NO | TIMESTAMP ARCHIVES | DSNTIPH | 9 |
| DSN6ARVP | QUIESCE | 00005 | QUIESCE PERIOD | DSNTIPA | 14 |
| DSN6ARVP | ARCRETN | 00005 | RETENTION PERIOD | DSNTIPA | 13 |
| DSN6ARVP | ARCPFX1 | DB2V710S.ARCHLOG1 | ARCH LOG 1 PREFIX | DSNTIPH | 7 |
| DSN6ARVP | ARCPFX2 | DB2V710S.ARCHLOG2 | ARCH LOG 2 PREFIX | DSNTIPH | 8 |
| DSN6ARVP | PRIQTY | 0000000750 | PRIMARY QUANTITY | DSNTIPA | 2 |
| DSN6ARVP | SECQTY | 0000000015 | SECONDARY QTY | DSNTIPA | 3 |
| DSN6ARVP | UNIT | SYSDA | DEVICE TYPE 1 | DSNTIPA | 5 |
| DSN6ARVP | UNIT2 | NONE | DEVICE TYPE 2 | DSNTIPA | 6 |
| DSN6ARVP | ARCWRTC | 1011000000000000 | WTOR ROUTE CODE | DSNTIPA | 12 |
| DSN6SPRM | ABIND | YES | AUTO BIND | DSNTIPO | 8 |
| DSN6SPRM | SYSADM2 | PAOLOT1 | SYSTEM ADMIN 2 | DSNTIPP | 4 |
| DSN6SPRM | AUTHCACH | 01024 | PLAN AUTH CACHE | DSNTIPP | 10 |
| DSN6SPRM | AUTH | YES | USE PROTECTION | DSNTIPP | 2 |
| DSN6SPRM | BMPTOUT | 00004 | IMS BMP TIMEOUT | DSNTIPI | 11 |
| DSN6SPRM | LEMAX | 00020 | MAXIMUM LE TOKENS | DSNTIP7 | 3 |
| DSN6SPRM | BINDNV | BINDADD | BIND NEW PACKAGE | DSNTIPP | 9 |
| DSN6SPRM | CDSSRDEF | 1 | CURRENT DEGREE | DSNTIP4 | 6 |
| DSN6SPRM | DBCHK | NO | | | |
| DSN6SPRM | DEFLTID | IBMUSER | UNKNOWN AUTHID | DSNTIPP | 7 |
| DSN6SPRM | CHGDC AND EDPROP | 1 | DPROP SUPPORT | DSNTIPO | 10 |
| DSN6SPRM | DECDIV3 | NO | MIN. DIVIDE SCALE | DSNTIPF | 3 |
| DSN6SPRM | DLITOUT | 00006 | DLI BATCH TIMEOUT | DSNTIPI | 12 |
| DSN6SPRM | DSMAX | 0000003000 | MAXIMUM OPEN DATA SETS | DSNTIPC | 1 |
| DSN6SPRM | EDMPOOL | 0015167488 | EDMPOOL STORAGE SIZE | DSNTIPC | 2 |
| DSN6SPRM | RECALLD | 00120 | RECALL DELAY | DSNTIPO | 3 |

```
DSN6SPRM RELCURHL        YES                                RELEASE LOCKS                      DSNTIP4    10
DSN6SPRM RECALL          YES                                RECALL DATA BASE                   DSNTIPO     2
DSN6SPRM IRLMAUT         YES                                AUTO START                         DSNTIPI     4
DSN6SPRM ABEXP           YES                                EXPLAIN PROCESSING                 DSNTIPO     9
DSN6SPRM IRLMPRC         IRLSPROC                           PROC NAME                          DSNTIPI     5
DSN6SPRM IRLMSID         IRLM                               SUBSYSTEM NAME                     DSNTIPI     2
DSN6SPRM IRLMSWT         0000000300                         TIME TO AUTO START                 DSNTIPI     6
DSN6SPRM NUMLKTS         0000001000                         LOCKS PER TABLE(SPACE)             DSNTIPJ     3
DSN6SPRM NUMLKUS         0000010000                         LOCKS PER USER                     DSNTIPJ     4
DSN6SPRM HOPAUTH         BOTH                               AUTH AT HOP SITE                   DSNTIP5     7
DSN6SPRM SEQCACH         SEQUENTIAL                         SEQUENTIAL CACHE                   DSNTIPE     7
DSN6SPRM RRULOCK         NO                                 U LOCK FOR RR OR RS                DSNTIPI     8
DSN6SPRM DESCSTAT        NO                                 DESCRIBE FOR STATIC                DSNTIPF    16
DSN6SPRM SEQPRES         NO                                 UTILITY CACHE OPTION               DSNTIPE     8
DSN6SPRM CACHEDYN        NO                                 CACHE DYNAMIC SQL                  DSNTIP4     7
DSN6SPRM RETLWAIT        00000                              RETAINED LOCK TIMEOUT              DSNTIPI    13
DSN6SPRM CACHERAC        0000032768                         ROUTINE AUTH CACHE                 DSNTIPP    12
DSN6SPRM EDMDSPAC        0000000000                         EDMPOOL DATA SPACE SIZE            DSNTIPC     3
DSN6SPRM CONTSTOR        NO                                 CONTRACT THREAD STG                DSNTIPE    10
DSN6SPRM MAXKEEPD        0000005000                         MAX KEPT DYN STMTS                 DSNTIPE     9
DSN6SPRM RETVLCFK        NO                                 VARCHAR FROM INDEX                 DSNTIP4     9
DSN6SPRM SYSOPR1         SYSOPR                             SYSTEM OPERATOR 1                  DSNTIPP     5
DSN6SPRM SYSOPR2         SYSOPR                             SYSTEM OPERATOR 2                  DSNTIPP     6
DSN6SPRM CACHEPAC        0000032768                         PACKAGE AUTH CACHE                 DSNTIPP    11
DSN6SPRM PARAMDEG        0000000000                         MAX DEGREE                         DSNTIP4    11
DSN6SPRM PARTKEYU        YES                                UPDATE PART KEY COLS               DSNTIP4    12
DSN6SPRM STATHIST        ACCESSPATH                         STATISTICS HISTORY                 DSNTIPO    14
DSN6SPRM RGFNMPRT        DSN_REGISTER_APPL                  APPL REGISTRATION TABLE            DSNTIPZ     8
DSN6SPRM RGFCOLID        DSNRGCOL                           REGISTRATION OWNER                 DSNTIPZ     6
DSN6SPRM RGFESCP                                            ART-ORT ESCAPE CHAR                DSNTIPZ     5
DSN6SPRM RGFINSTL        NO                                 INSTALL DDCTRL SUPT                DSNTIPZ     1
DSN6SPRM RGFDEDPL        NO                                 CONTROL ALL APPLICATIONS           DSNTIPZ     2
DSN6SPRM RGFFULLQ        YES                                REQUIRE FULL NAMES                 DSNTIPZ     3
DSN6SPRM RGFDEFLT        APPL                               UNREGISTERED DDL DEFAULT           DSNTIPZ     4
DSN6SPRM RGFDBNAM        DSNRGFDB                           REGISTRATION DATABASE              DSNTIPZ     7
DSN6SPRM RGFNMORT        DSN_REGISTER_OBJT                  OBJT REGISTRATION TABLE            DSNTIPZ     9
DSN6SPRM MAXRBLK         0000000250                         RID POOL SIZE                      DSNTIPC     7
DSN6SPRM MINRBLK         0000000001
DSN6SPRM SYSADM          KARRAS                             SYSTEM ADMIN 1                     DSNTIPP     3
DSN6SPRM SRTPOOL         0001024000                         SORT POOL SIZE                     DSNTIPC     6
DSN6SPRM IRLMRWT         0000000060                         RESOURCE TIMEOUT                   DSNTIPI     3
DSN6SPRM ALPOOLX         0000032256
DSN6SPRM SITETYP         LOCALSITE                          SITE TYPE                          DSNTIPO    11
DSN6SPRM UTIMOUT         00006                              UTILITY TIMEOUT                    DSNTIPI     7
DSN6SPRM XLKUPDLT        NO                                 X LOCK FOR SEARCHED U OR D         DSNTIPI     9
DSN6SPRM OPTHINTS        NO                                 OPTIMIZATION HINTS                 DSNTIP4     8
DSN6SPRM TRKRSITE        NO                                 TRACKER SITE                       DSNTIPO    12
DSN6SPRM EDMBFIT         NO                                 LARGE EDM BETTER FIT               DSNTIP4    13
DSN6SPRM EDMDSMAX        0000000000                         EDMPOOL DATA SPACE MAX             DSNTIPC     4
DSN6SPRM STARJOIN        DISABLE
DSN6SPRM DBACRVW         NO                                 DBADM CREATE AUTH                  DSNTIPP    13
DSN6SPRM SUPERRS         YES                                SUPPRESS SOFT ERRORS               DSNTIPM    10
DSN6SPRM STATROLL        NO                                 STATISTICS ROLLUP                  DSNTIPO    15
DSN6SPRM MINSTOR         NO                                 MANAGE THREAD STORAGE              DSNTIPE    11
DSN6SPRM EVALUNC         NO                                 EVALUATE UNCOMMITTED               DSNTIP4    15
DSN6SPRM CATALOG         DB2V710S                           CATALOG ALIAS                      DSNTIPA2    1
DSN6SPRM RESTART DEF     RESTART                            RESTART OR DEFER                   DSNTIPS     1
DSN6SPRM ALL-DBNAME      ALL                                DBSTARTX                           DSNTIPS  2-37
DSN6FAC  CMTSTAT         ACTIVE                             DDF THREADS                        DSNTIPR     7
DSN6FAC  TCPALVER        NO                                 TCP IP ALREADY VERIFIED            DSNTIP5     3
DSN6FAC  RESYNC          00002                              RESYNC INTERVAL                    DSNTIPR     6
DSN6FAC  RLFERRD         NOLIMIT                            RLST ACCESS ERROR                  DSNTIPR     5
DSN6FAC  DDF             AUTO                               DDF STARTUP OPTION                 DSNTIPR     1
DSN6FAC  IDTHTOIN        00000                              IDLE THREAD TIMEOUT                DSNTIPR    10
DSN6FAC  MAXTYPE1        0000000000                         MAX TYPE1 INACTIVE THREADS         DSNTIPR     8
DSN6FAC  TCPKPALV        ENABLE                             TCPIP KEEPALIVE                    DSNTIP5     8
DSN6FAC  POOLINAC        00120                              POOL THREAD TIMEOUT                DSNTIP5     9
DSN6GRP  ASSIST          NO                                 ASSISTANT                          DSNTIPK     6
DSN6GRP  COORDNTR        NO                                 COORDINATOR                        DSNTIPK     5
DSN6GRP  IMMEDWRI        NO                                 IMMEDIATE WRITE                    DSNTIP4    14
DSN6GRP  DSHARE          NO                                 DATA SHARING FUNCTION              DSNTIPA1    2
DSN6GRP  MEMBNAME        DSN1                               MEMBER NAME                        DSNTIPK     2
DSN6GRP  GRPNAME         DSNCAT                             GROUP NAME                         DSNTIPK     1
DSNHDECP DEFLANG         IBMCOB                             LANGUAGE DEFAULT                   DSNTIPF     1
DSNHDECP DECIMAL         ,                                  DECIMAL POINT IS                   DSNTIPF     2
DSNHDECP DELIM           D                                  STRING DELIMITER                   DSNTIPF     4
DSNHDECP SQLDELI         DEFAULT                            SQL STRING DELIMITER               DSNTIPF     5
DSNHDECP DSQLDELI        APOST                              DIST SQL STR DELIMITER             DSNTIPF     6
DSNHDECP MIXED           NO                                 MIXED DATA                         DSNTIPF     7
DSNHDECP SCCSID          00037                              EBCDIC CODED CHAR SET              DSNTIPF     8
DSNHDECP MCCSID          65534                              EBCDIC CODED CHAR SET              DSNTIPF     8
DSNHDECP GCCSID          65534                              EBCDIC CODED CHAR SET              DSNTIPF     8
DSNHDECP ASCCSID         00437                              ASCII CODED CHAR SET               DSNTIPF     9
DSNHDECP AMCCSID         65534                              ASCII CODED CHAR SET               DSNTIPF     9
DSNHDECP AGCCSID         65534                              ASCII CODED CHAR SET               DSNTIPF     9
DSNHDECP USCCSID         00367                              UNICODE CCSID                      DSNTIPF    10
DSNHDECP UMCCSID         01208                              UNICODE CCSID                      DSNTIPF    10
DSNHDECP UGCCSID         01200                              UNICODE CCSID                      DSNTIPF    10
DSNHDECP ENSCHEME        EBCDIC                             DEF ENCODING SCHEME                DSNTIPF    11
DSNHDECP APPENSCH        EBCDIC                             APPLICATION ENCODING               DSNTIPF    13
DSNHDECP DATE            ISO                                DATE FORMAT                        DSNTIP4     1
DSNHDECP TIME            ISO                                TIME FORMAT                        DSNTIP4     2
DSNHDECP DATELEN         000                                LOCAL DATE LENGTH                  DSNTIP4     3
DSNHDECP TIMELEN         000                                LOCAL TIME LENGTH                  DSNTIP4     4
DSNHDECP STDSQL          YES                                STD SQL LANGUAGE                   DSNTIP4     5
```

```
DSNHDECP CHARSET          ALPHANUM                  EBCDIC CODED CHAR SET              DSNTIPF     8
DSNHDECP SSID             DB2S                      SUBSYSTEM NAME                     DSNTIPM     1
DSNHDECP DECARTH          15                        DECIMAL ARITHMETIC                 DSNTIPF    14
DSNHDECP DYNRULS          YES                       USE FOR DYNAMICRULES               DSNTIPF    15
DSNHDECP COMPAT           OFF
DSNHDECP LC_CTYPE                                   LOCALE LC_CTYPE                    DSNTIPF    12
```

# Appendix B. SQL examples

This appendix contains the DDL, DML, and programming examples used for the functions reported in 2.1, “UNION everywhere” on page 23.

## B.1 Creating the credit card database

```
--*********************************************************************
--* CREATE EXAMPLET TABLES, INDEXES, VIEWS, AND CONSTRAINTS FOR THE
--* SQL ENHANCEMENS CHAPTER
--*
--* THE DATABASE REPRESENTS A CREDIT CARD SYSTEM. THE SYSTEM CONTAINS
--* AN ACCOUNT TABLE WITH A SINGLE ROW FOR EACH ACCOUNT. THERE ARE THREE
--* TABLES WHICH HOLD TRANSACTIONS FOR EACH ACCOUNT. THESE TABLES ARE
--* BROKEN INTO CARD TYPE AND ARE PLATINUM, GOLD AND BLUE.
--*
--* THREE ARE THREE FURTHER TABLES WHICH CONTAIN TAXATION DETAILS FOR
--* EACH ACCOUNT. THESE TABLES ARE TAX_MONTH1, TAX_MONTH2 AND
--* TAX_MONTH3. THERE IS A ROW WHICH REPESENTS 1% OF THE TRANSACTION
--* AMOUNT FOR EACH TRANSACTION FOR THAT MONTH ACROSS ALL CARD TYPES.
--*********************************************************************
--
  DROP DATABASE DBPAT003;
  COMMIT;
--
  CREATE DATABASE DBPAT003
    STOGROUP DSN8G710
    BUFFERPOOL BP0
    CCSID EBCDIC;

  CREATE TABLESPACE TSPAT003
    IN DBPAT003
    USING STOGROUP DSN8G710
              PRIQTY 48
              SECQTY 48
              ERASE NO
    LOCKSIZE PAGE LOCKMAX SYSTEM
    BUFFERPOOL BP0
    CLOSE NO
    CCSID EBCDIC;

  COMMIT ;

--*********************************************************************
--* CREATE ACCOUNT TABLE, INDEX AND POPULATE WITH DATA
--*********************************************************************
--
  CREATE TABLE ACCOUNT
     (ACCOUNT           CHAR(6)      NOT NULL,
      ACCOUNT_NAME      VARCHAR(30)  NOT NULL,
      CREDIT_LIMIT      DECIMAL(11,2) NOT NULL,
      TYPE              CHAR(1)       NOT NULL,
      CONSTRAINT CUSTP1 PRIMARY KEY(ACCOUNT))
  IN DBPAT003.TSPAT003
  CCSID EBCDIC;
  COMMIT;
```

© Copyright IBM Corp. 2001

```
--
--
  CREATE UNIQUE INDEX XACCT001
    ON ACCOUNT
    (ACCOUNT           ASC)
    USING STOGROUP DSN8G710
    PRIQTY 48
    ERASE NO
    BUFFERPOOL BP0
    CLOSE NO;
--
  INSERT INTO ACCOUNT (ACCOUNT, ACCOUNT_NAME, CREDIT_LIMIT, TYPE)
  VALUES ('ABC010', 'BIG PETROLEUM', 100000.00, 'C');
  INSERT INTO ACCOUNT (ACCOUNT, ACCOUNT_NAME, CREDIT_LIMIT, TYPE)
  VALUES ('BWH450', 'HUTH & DAUGHTERS', 2000.00, 'C');
  INSERT INTO ACCOUNT (ACCOUNT, ACCOUNT_NAME, CREDIT_LIMIT, TYPE)
  VALUES ('ZXY930', 'MIGHTY BEARS PLC', 50000.00, 'C');
  INSERT INTO ACCOUNT (ACCOUNT, ACCOUNT_NAME, CREDIT_LIMIT, TYPE)
  VALUES ('MNP230', 'MR P TENCH', 50.00, 'P');
  INSERT INTO ACCOUNT (ACCOUNT, ACCOUNT_NAME, CREDIT_LIMIT, TYPE)
  VALUES ('BMP291', 'BASEL FERRARI', 25000.00, 'C');
  INSERT INTO ACCOUNT (ACCOUNT, ACCOUNT_NAME, CREDIT_LIMIT, TYPE)
  VALUES ('XPM673', 'SCREAM SAVER PTY LTD', 5000.00, 'C');
  INSERT INTO ACCOUNT (ACCOUNT, ACCOUNT_NAME, CREDIT_LIMIT, TYPE)
  VALUES ('ULP231', 'MS S FLYNN', 500.00, 'P');
  INSERT INTO ACCOUNT (ACCOUNT, ACCOUNT_NAME, CREDIT_LIMIT, TYPE)
  VALUES ('XPM961', 'MICHAL LINGERIE', 100000.00, 'C');
--
--*********************************************************************
--* CREATE PLATINUM TABLE, INDEX AND POPULATE WITH DATA
--*********************************************************************
  CREATE TABLE PLATINUM
    (ACCOUNT           CHAR(6)      NOT NULL,
     DATE              DATE         NOT NULL,
     AMOUNT            DECIMAL(7,2) NOT NULL,
     CONSTRAINT FPLAT001 FOREIGN KEY (ACCOUNT) REFERENCES ACCOUNT)
  IN DBPAT003.TSPAT003
  CCSID EBCDIC;
  COMMIT;
--
  CREATE INDEX XPLAT001
    ON PLATINUM
    (ACCOUNT           ASC)
    USING STOGROUP DSN8G710
    PRIQTY 48
    ERASE NO
    BUFFERPOOL BP0
    CLOSE NO;
--
  INSERT INTO PLATINUM (ACCOUNT, DATE, AMOUNT)
  VALUES ('ABC010', '03/07/1998', 5.25);
  INSERT INTO PLATINUM (ACCOUNT, DATE, AMOUNT)
  VALUES ('BWH450', '01/10/2000', 150.00);
  INSERT INTO PLATINUM (ACCOUNT, DATE, AMOUNT)
  VALUES ('ABC010', '01/15/2000', 1150.23);
  INSERT INTO PLATINUM (ACCOUNT, DATE, AMOUNT)
  VALUES ('BWH450', '01/15/1999', 67.00);
  INSERT INTO PLATINUM (ACCOUNT, DATE, AMOUNT)
```

```
  VALUES ('ABC010', '02/29/2000', -1150.23);
--
--*********************************************************************
--* CREATE GOLD TABLE, INDEX AND POPULATE WITH DATA
--*********************************************************************
--
  CREATE TABLE GOLD
    (ACCOUNT           CHAR(6)      NOT NULL,
     DATE              DATE         NOT NULL,
     AMOUNT            DECIMAL(7,2) NOT NULL,
     CONSTRAINT FGOLD001 FOREIGN KEY (ACCOUNT) REFERENCES ACCOUNT)
  IN DBPAT003.TSPAT003
  CCSID EBCDIC;
  COMMIT;
--
  CREATE INDEX XGOLD001
    ON GOLD
    (ACCOUNT           ASC)
    USING STOGROUP DSN8G710
    PRIQTY 48
    ERASE NO
    BUFFERPOOL BP0
    CLOSE NO;
--
  INSERT INTO GOLD (ACCOUNT, DATE, AMOUNT)
  VALUES ('ZXY930', '12/13/1999', 635.25);
  INSERT INTO GOLD (ACCOUNT, DATE, AMOUNT)
  VALUES ('MNP230', '01/11/2000', 150.00);
  INSERT INTO GOLD (ACCOUNT, DATE, AMOUNT)
  VALUES ('ZXY930', '01/15/2000', 233.57);
  INSERT INTO GOLD (ACCOUNT, DATE, AMOUNT)
  VALUES ('BMP291', '02/15/1999', 31.32);
  INSERT INTO GOLD (ACCOUNT, DATE, AMOUNT)
  VALUES ('ZXY930', '01/30/2000', -233.57);
--
--*********************************************************************
--* CREATE BLUE TABLE, INDEX AND POPULATE WITH DATA
--*********************************************************************
--
  CREATE TABLE BLUE
    (ACCOUNT           CHAR(6)      NOT NULL,
     DATE              DATE         NOT NULL,
     AMOUNT            DECIMAL(7,2) NOT NULL,
     CONSTRAINT FBLUE001 FOREIGN KEY (ACCOUNT) REFERENCES ACCOUNT)
  IN DBPAT003.TSPAT003
  CCSID EBCDIC;
  COMMIT;
--
  CREATE INDEX XBLUE001
    ON BLUE
    (ACCOUNT           ASC)
    USING STOGROUP DSN8G710
    PRIQTY 48
    ERASE NO
    BUFFERPOOL BP0
    CLOSE NO;
--
  INSERT INTO BLUE (ACCOUNT, DATE, AMOUNT)
```

```
  VALUES ('ULP231', '03/07/1998', 16.43);
  INSERT INTO BLUE (ACCOUNT, DATE, AMOUNT)
  VALUES ('XPM673', '01/21/2000', 10927.47);
  INSERT INTO BLUE (ACCOUNT, DATE, AMOUNT)
  VALUES ('XPM961', '01/15/2000', 15253.65);
  INSERT INTO BLUE (ACCOUNT, DATE, AMOUNT)
  VALUES ('XPM673', '02/02/2000', -31.32);
  INSERT INTO BLUE (ACCOUNT, DATE, AMOUNT)
  VALUES ('XPM961', '01/30/2000', -500.00);
--
--*********************************************************************
--* CREATE TAX_MONTH1 TABLE, INDEX AND POPULATE WITH DATA
--*********************************************************************
--
--DROP TABLE TAX_MONTH1;
--COMMIT;
--
  CREATE TABLE TAX_MONTH1
    (ACCOUNT           CHAR(6)      NOT NULL,
     DATE              DATE         NOT NULL,
     AMOUNT            DECIMAL(11,2) NOT NULL,
     CONSTRAINT FTAXM101 FOREIGN KEY (ACCOUNT) REFERENCES ACCOUNT)
  IN DBPAT003.TSPAT003
  CCSID EBCDIC;
  COMMIT;
--
  CREATE INDEX XTAXM101
    ON TAX_MONTH1
    (ACCOUNT           ASC)
    USING STOGROUP DSN8G710
    PRIQTY 48
    ERASE NO
    BUFFERPOOL BP0
    CLOSE NO;
--
  CREATE VIEW V_TAX_MONTH1 (ACCOUNT, DATE, AMOUNT) AS
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM PLATINUM
    WHERE DATE BETWEEN '01/01/2000' AND '01/31/2000'
    UNION ALL
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM GOLD
    WHERE DATE BETWEEN '01/01/2000' AND '01/31/2000'
    UNION ALL
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM BLUE
    WHERE DATE BETWEEN '01/01/2000' AND '01/31/2000';
--
  INSERT INTO TAX_MONTH1 (ACCOUNT, DATE, AMOUNT)
```

```
    (SELECT ACCOUNT, DATE, SUM(AMOUNT) * .01
     FROM V_TAX_MONTH1
     GROUP BY ACCOUNT, DATE);
--
  SELECT * FROM TAX_MONTH1;
--
  DROP VIEW V_TAX_MONTH1;
--
--*********************************************************************
--* CREATE TAX_MONTH2 TABLE, INDEX AND POPULATE WITH DATA
--*********************************************************************
--
--DROP TABLE TAX_MONTH2;
--COMMIT;
--
  CREATE TABLE TAX_MONTH2
    (ACCOUNT           CHAR(6)      NOT NULL,
     DATE              DATE         NOT NULL,
     AMOUNT            DECIMAL(11,2) NOT NULL,
     CONSTRAINT FTAXM201 FOREIGN KEY(ACCOUNT) REFERENCES ACCOUNT)
  IN DBPAT003.TSPAT003
  CCSID EBCDIC;
  COMMIT;
--
  CREATE INDEX XTAXM201
    ON TAX_MONTH2
    (ACCOUNT           ASC)
    USING STOGROUP DSN8G710
    PRIQTY 48
    ERASE NO
    BUFFERPOOL BP0
    CLOSE NO;
--
  CREATE VIEW V_TAX_MONTH2 (ACCOUNT, DATE, AMOUNT) AS
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM PLATINUM
    WHERE DATE BETWEEN '02/01/2000' AND '02/29/2000'
    UNION ALL
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM GOLD
    WHERE DATE BETWEEN '02/01/2000' AND '02/29/2000'
    UNION ALL
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM BLUE
    WHERE DATE BETWEEN '02/01/2000' AND '02/29/2000';
--
  INSERT INTO TAX_MONTH2 (ACCOUNT, DATE, AMOUNT)
    (SELECT ACCOUNT, DATE, SUM(AMOUNT) * .01
     FROM V_TAX_MONTH2
```

```
     GROUP BY ACCOUNT, DATE);
--
  SELECT * FROM TAX_MONTH2;
--
  DROP VIEW V_TAX_MONTH2;
--
--*********************************************************************
--* CREATE TAX_MONTH3 TABLE, INDEX AND POPULATE WITH DATA
--*********************************************************************
--
--DROP TABLE TAX_MONTH3;
--COMMIT;
--
  CREATE TABLE TAX_MONTH3
    (ACCOUNT           CHAR(6)      NOT NULL,
     DATE              DATE         NOT NULL,
     AMOUNT            DECIMAL(11,2) NOT NULL,
     CONSTRAINT FTAXM301 FOREIGN KEY(ACCOUNT) REFERENCES ACCOUNT)
  IN DBPAT003.TSPAT003
  CCSID EBCDIC;
  COMMIT;
--
  CREATE INDEX XTAXM301
    ON TAX_MONTH3
    (ACCOUNT           ASC)
    USING STOGROUP DSN8G710
    PRIQTY 48
    ERASE NO
    BUFFERPOOL BP0
    CLOSE NO;
--
  CREATE VIEW V_TAX_MONTH3 (ACCOUNT, DATE, AMOUNT) AS
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM PLATINUM
    WHERE DATE BETWEEN '03/01/2000' AND '03/31/2000'
    UNION ALL
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM GOLD
    WHERE DATE BETWEEN '03/01/2000' AND '03/31/2000'
    UNION ALL
    SELECT ACCOUNT, DATE,
           CASE WHEN AMOUNT < 0 THEN AMOUNT * -1
                ELSE AMOUNT
           END
    FROM BLUE
    WHERE DATE BETWEEN '03/01/2000' AND '03/31/2000';
--
  INSERT INTO TAX_MONTH3 (ACCOUNT, DATE, AMOUNT)
    (SELECT ACCOUNT, DATE, SUM(AMOUNT) * .01
     FROM V_TAX_MONTH3
     GROUP BY ACCOUNT, DATE);
--
```

```
  SELECT * FROM TAX_MONTH3;
--
  DROP VIEW V_TAX_MONTH3;
--
  CREATE VIEW V_TAX_QTR1 (ACCOUNT, DATE, AMOUNT) AS
    SELECT ACCOUNT, DATE, AMOUNT
    FROM TAX_MONTH1
    WHERE DATE BETWEEN '01/01/2000' AND '01/31/2000'
    UNION ALL
    SELECT ACCOUNT, DATE, AMOUNT
    FROM TAX_MONTH2
    WHERE DATE BETWEEN '02/01/2000' AND '02/29/2000'
    UNION ALL
    SELECT ACCOUNT, DATE, AMOUNT
    FROM TAX_MONTH3
    WHERE DATE BETWEEN '03/01/2000' AND '03/31/2000';
--
```

## B.2 Union in views: Create JANUARY2000 view

```
--*********************************************************************
--* UNION IN VIEWS EXAMPLE: CREATE JANUARY2000 VIEW
--*********************************************************************
--
--DROP VIEW JANUARY2000;
--COMMIT;
--
  CREATE VIEW JANUARY2000 (ACCOUNT, DATE, AMOUNT) AS
    SELECT ACCOUNT, DATE, AMOUNT
    FROM PLATINUM
    WHERE DATE BETWEEN '01/01/2000' AND '01/31/2000'
    UNION ALL
    SELECT ACCOUNT, DATE, AMOUNT
    FROM GOLD
    WHERE DATE BETWEEN '01/01/2000' AND '01/31/2000'
    UNION ALL
    SELECT ACCOUNT, DATE, AMOUNT
    FROM BLUE
    WHERE DATE BETWEEN '01/01/2000' AND '01/31/2000';
--
  SELECT SUM(AMOUNT), COUNT(*)
  FROM JANUARY2000;
```

## B.3 Union in table-spec

```
--*********************************************************************
--* UNION IN TABLE-SPEC EXAMPLE
--*********************************************************************
--
  SELECT ACCOUNT.ACCOUNT, ACCOUNT.ACCOUNT_NAME,
         SUM(ALLCARDS.AMOUNT) AS BALANCE
  FROM ACCOUNT,
       TABLE (SELECT ACCOUNT, AMOUNT
              FROM PLATINUM
              UNION ALL
              SELECT ACCOUNT, AMOUNT
```

```
                FROM GOLD
                UNION ALL
                SELECT ACCOUNT, AMOUNT
                FROM BLUE
                ) AS ALLCARDS(ACCOUNT, AMOUNT)
  WHERE ACCOUNT.ACCOUNT = ALLCARDS.ACCOUNT
  GROUP BY ACCOUNT.ACCOUNT, ACCOUNT.ACCOUNT_NAME;
```

## B.4 Union in basic predicates

```
--*********************************************************************
--* UNION IN BASIC PREDICATES EXAMPLE
--*********************************************************************
--
  SELECT 'GOLD CARD ACCOUNT:', ACCOUNT.ACCOUNT, ACCOUNT.ACCOUNT_NAME,
         ACCOUNT.CREDIT_LIMIT
  FROM ACCOUNT
  WHERE 'GOLD' = (SELECT 'PLATINUM'
                  FROM PLATINUM
                  WHERE ACCOUNT = 'BMP291'
                  UNION
                  SELECT 'GOLD'
                  FROM GOLD
                  WHERE ACCOUNT = 'BMP291'
                  UNION
                  SELECT 'BLUE'
                  FROM BLUE
                  WHERE ACCOUNT = 'BMP291')
     AND ACCOUNT = 'BMP291';
```

## B.5 Union in qualified predicates

```
--*********************************************************************
--* UNION IN QUALIFIED PREDICATES EXAMPLE
--*********************************************************************
--
  SELECT 'ACCOUNT OVER CREDIT LIMIT', ACCOUNT_NAME
  FROM ACCOUNT T1
  WHERE CREDIT_LIMIT < ANY
                 (SELECT SUM(AMOUNT)
                  FROM PLATINUM
                  WHERE ACCOUNT = T1.ACCOUNT
                  UNION
                  SELECT SUM(AMOUNT)
                  FROM GOLD
                  WHERE ACCOUNT = T1.ACCOUNT
                  UNION
                  SELECT SUM(AMOUNT)
                  FROM BLUE
                  WHERE ACCOUNT = T1.ACCOUNT)
    AND T1.ACCOUNT = T1.ACCOUNT;
```

## B.6 Union in the EXISTS predicates

```
--*********************************************************************
--* UNION IN THE EXISTS PREDICATES EXAMPLE
--*********************************************************************
  SELECT ACCOUNT, ACCOUNT_NAME
  FROM ACCOUNT T1
  WHERE EXISTS (SELECT *
                FROM PLATINUM
                WHERE ACCOUNT = T1.ACCOUNT
                  AND YEAR(DATE) = 2000
                  AND MONTH(DATE) = 1
                UNION
                SELECT *
                FROM GOLD
                WHERE ACCOUNT = T1.ACCOUNT
                  AND YEAR(DATE) = 2000
                  AND MONTH(DATE) = 1
                UNION
                SELECT *
                FROM BLUE
                WHERE ACCOUNT = T1.ACCOUNT
                  AND YEAR(DATE) = 2000
                  AND MONTH(DATE) = 1);
```

## B.7 Union in the IN predicates

```
--*********************************************************************
--* UNIONS IN THE IN PREDICATES EXAMPLE
--*********************************************************************
--
  SELECT ACCOUNT, ACCOUNT_NAME
  FROM ACCOUNT
  WHERE ACCOUNT NOT IN (SELECT ACCOUNT
                        FROM PLATINUM
                        WHERE YEAR(DATE) = 2000
                          AND MONTH(DATE) = 1
                        UNION
                        SELECT ACCOUNT
                        FROM GOLD
                        WHERE YEAR(DATE) = 2000
                          AND MONTH(DATE) = 1
                        UNION
                        SELECT ACCOUNT
                        FROM BLUE
                        WHERE YEAR(DATE) = 2000
                          AND MONTH(DATE) = 1);
```

## B.8 Union in INSERT and UPDATE

```
--*********************************************************************
--* UNION IN INSERT AND UPDATE
--*********************************************************************
--
--DROP TABLE CARDS_2000;
--COMMIT;
```

```
--
  CREATE TABLE CARDS_2000
    (ACCOUNT           CHAR(6)      NOT NULL,
     AMOUNT            DECIMAL(7,2) NOT NULL,
     CONSTRAINT FPLAT001 FOREIGN KEY (ACCOUNT) REFERENCES ACCOUNT)
  IN DBPAT003.TSPAT003
  CCSID EBCDIC;
  COMMIT;
--
  CREATE INDEX XC200001
    ON CARDS_2000
    (ACCOUNT           ASC)
    USING STOGROUP DSN8G710
    PRIQTY 48
    ERASE NO
    BUFFERPOOL BP0
    CLOSE NO;
--
  SELECT * FROM CARDS_2000;
--
  INSERT INTO CARDS_2000 (ACCOUNT, AMOUNT)
    (SELECT ACCOUNT, SUM(AMOUNT)
     FROM PLATINUM
     WHERE YEAR(DATE) = 2000
     GROUP BY ACCOUNT
     UNION
     SELECT ACCOUNT, SUM(AMOUNT)
     FROM GOLD
     WHERE YEAR(DATE) = 2000
     GROUP BY ACCOUNT
     UNION
     SELECT ACCOUNT, SUM(AMOUNT)
     FROM BLUE
     WHERE YEAR(DATE) = 2000
     GROUP BY ACCOUNT);
--
  SELECT *
  FROM CARDS_2000;
--
  ALTER TABLE CARDS_2000
    ADD CARD_TYPE  CHAR(1) NOT NULL WITH DEFAULT;
--
  UPDATE CARDS_2000 T1
    SET CARD_TYPE = (SELECT 'P'
                     FROM PLATINUM T2
                     WHERE T1.ACCOUNT = T2.ACCOUNT
                     UNION
                     SELECT 'G'
                     FROM GOLD  T3
                     WHERE T1.ACCOUNT = T3.ACCOUNT
                     UNION
                     SELECT 'B'
                     FROM BLUE T4
                     WHERE T1.ACCOUNT = T4.ACCOUNT);
--
  SELECT * FROM CARDS_2000;
--
```

## B.9 Optimizing union everywhere queries

This section relies on a PLAN_TABLE being created with the new columns for DB2 V7. The CREATE for this can be found in the DB2 sample dataset.

```
--
--DELETE FROM PLAN_TABLE;
--
  EXPLAIN ALL SET QUERYNO = 1 FOR
    SELECT T2.ACCOUNT, AVG(T2.AMOUNT)
    FROM ACCOUNT T1,
         V_TAX_QTR1 T2
    WHERE T1.ACCOUNT = T2.ACCOUNT
      AND T1.TYPE = 'C'
      AND T2.DATE IN ('01/30/2000', '02/29/2000')
    GROUP BY T2.ACCOUNT;
--
  SELECT QBLOCKNO, PLANNO, TNAME, TABLE_TYPE,
         METHOD, QBLOCK_TYPE,
         PARENT_QBLOCKNO,
         SORTC_UNIQ, ACCESS_DEGREE, ACCESS_PGROUP_ID,
         JOIN_DEGREE, JOIN_PGROUP_ID
  FROM PLAN_TABLE
  WHERE QUERYNO = 1
  ORDER BY QBLOCKNO, PLANNO;
```

# Appendix C. Using the additional material

This redbook also contains additional material in CD-ROM or diskette format, and/or Web material. See the appropriate section below for instructions on using or downloading each type of material.

## C.1 Locating the additional material on the Internet

The Web material associated with this redbook is also available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://ibm.com/redbooks/SG246121

Alternatively, you can go to the IBM Redbooks Web site at:

http://ibm.com/redbooks/

Select the **Additional materials** and open the directory that corresponds with the redbook form number.

## C.2 Using the Web material

The additional Web material that accompanies this redbook includes the following:

| *File name* | *Description* |
|---|---|
| **P1_6121.zip** | Zipped Freelance Presentation Part 1 |
| **P2_6121.zip** | Zipped Freelance Presentation Part 2 |
| **P3_6121.zip** | Zipped Freelance Presentation Part 3 |
| **P4_6121.zip** | Zipped Freelance Presentation Part 4 |
| **P5_6121.zip** | Zipped Freelance Presentation Part 5 |
| **P6_6121.zip** | Zipped Freelance Presentation Part 6 |
| **P7_6121.zip** | Zipped Freelance Presentation Part 7 |
| **P8_6121.zip** | Zipped Freelance Presentation Part 8 |
| **ALLP_6121.zip** | Zipped Freelance Presentation all Parts |

Each file contains the Freelance foils included in the corresponding part of the redbook.

### C.2.1 System requirements for downloading the Web material

The following system configuration is recommended for downloading the additional Web material.

| | |
|---|---|
| **Hard disk space**: | 8 MB minimum |
| **Operating System**: | Windows 95 or NT or 2000 |
| **Processor**: | Intel 386 or higher |
| **Memory**: | 16 MB |

### C.2.2 How to use the Web material

Create a subdirectory (folder) on your workstation and copy the contents of the Web material into this folder.

© Copyright IBM Corp. 2001

# Appendix D. Special notices

This publication is intended to help managers and professionals understand and evaluate the applicability to their environment of the new functions introduced by DB2 UDB Server for OS/390 and z/OS Version 7 and the IBM DB2 Utilities Suite. It is in the format of a presentation guide and as such can be easily utilized to propagate the information. The information in this publication is not intended as the specification of any programming interfaces that are provided by DB2 UDB Server for OS/390 and z/OS Version 7 and DB2 Utilities Suite. See the PUBLICATIONS section of the IBM Programming Announcements for DB2 UDB Server for OS/390 and z/OS Version 7 and DB2 Utilities Suite for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

© Copyright IBM Corp. 2001

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM ®
AIX
AS/400
BookManager
CICS
COBOL/370
DATABASE 2
DataGuide
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 Universal Database
DFSMS/MVS
Distributed Relational Database Architecture
DRDA
Enterprise Storage Server
IIntelligent Miner
Language Environment
MQSeries
MVS/ESA
Net.Data
Netfinity
z/OS
OS/2
OS/390
OS/400
PAL
Parallel Sysplex
QMF
RACF
Redbooks
Redbooks Logo
RETAIN
RMF
RS/6000
S/390
SecureWay
System/390
VisualAge
Visual Warehouse
WebSphere
Wizard
XT
400
Lotus
Domino
eSuite
Notes
NetView
zSeries

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

# Appendix E. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## E.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 543.

- *DB2 UDB Server for OS/390 Version 6 Technical Update*, SG24-6108
- *DB2 Java Stored Procedures -- Learning by Example*, SG24-5945
- *DB2 UDB for OS/390 Version 6 Performance Topics,* SG24-5351
- *DB2 for OS/390 Version 5 Performance Topics,* SG24-2213
- *DB2 for MVS/ESA Version 4 Non-Data-Sharing Performance Topics,* SG24-4562
- *DB2 UDB for OS/390 Version 6 Management Tools Package,* SG24-5759
- *DB2 Server for OS/390 Version 5 Recent Enhancements - Reference Guide*, SG24-5421
- *DB2 for OS/390 Capacity Planning,* SG24-2244
- *Getting Started with DB2 Stored Procedures: Give Them a Call through the Network,* SG24-4693
- *Developing Cross-Platform DB2 Stored Procedures: SQL Procedures and the DB2 Stored procedure Builder*, SG24-5485
- *DB2 for OS/390 and Continuous Availability,* SG24-5486
- *Parallel Sysplex Configuration: Cookbook,* SG24-2076
- *DB2 Java Stored Procedures: Learning by Example,* SG24-5945
- *DB2 for OS/390 Application Design for High Performance,* SG24-2233
- *Using RVA and SnapShot for Business Intelligence Applications with OS/390 and DB2*, SG24-5333
- *IBM Enterprise Storage Server Performance Monitoring and Tuning Guide*, SG24-5656
- *Migrating to DB2 UDB Version 7.1 in a Visual Warehouse Environment*, SG24-6107
- J*ava Programming Guide for OS/390*, SG24-5619
- *IBM e(logo)server zSeries 900 Technical Guide*, SG24-5975

© Copyright IBM Corp. 2001

## E.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

| CD-ROM Title | Collection Kit Number |
|---|---|
| IBM System/390 Redbooks Collection | SK2T-2177 |
| IBM Networking Redbooks Collection | SK2T-6022 |
| IBM Transaction Processing and Data Management Redbooks Collection | SK2T-8038 |
| IBM Lotus Redbooks Collection | SK2T-8039 |
| Tivoli Redbooks Collection | SK2T-8044 |
| IBM AS/400 Redbooks Collection | SK2T-2849 |
| IBM Netfinity Hardware and Software Redbooks Collection | SK2T-8046 |
| IBM RS/6000 Redbooks Collection | SK2T-8043 |
| IBM Application Development Redbooks Collection | SK2T-8037 |
| IBM Enterprise Storage and Systems Management Solutions | SK3T-3694 |

## E.3 Other resources

These publications are also relevant as further information sources:

- *DB2 UDB for OS/390 Version 6 ODBC Guide and Reference*, SC26-9005
- *DB2 UDB for OS/390 Version 6 Installation Guide*, GC26-9008-01
- *DB2 UDB for OS/390 and z/OS Version 7 What's New,* GC26-9946
- *DB2 UDB for OS/390 and z/OS Version 7 Installation Guide,* GC26-9936
- *DB2 UDB for OS/390 and z/OS Version 7 Command Reference,* SC26-9934
- *DB2 UDB for OS/390 and z/OS Version 7 Messages and Codes,* GC26-9940
- *DB2 UDB for OS/390 and z/OS Version 7 Utility Guide and Reference,* SC26-9945
- *DB2 UDB for OS/390 and z/OS Version 7 Programming Guide and Reference for Java,* SC26-9932
- *DB2 UDB for OS/390 and z/OS Version 7 Administration Guide,* SC26-9931
- *DB2 UDB for OS/390 and z/OS Version 7 Application Programming and SQL Guide,* SC26-9933
- *DB2 UDB for OS/390 and z/OS Version 7 Release Planning Guide*, SC26-9943
- *DB2 UDB for OS/390 and z/OS Version 7 SQL Reference,* SC26-9944
- *DB2 UDB for OS/390 and z/OS Version 7 Text Extender Administration and Programming,* SC26-9948
- *DB2 UDB for OS/390 and z/OS Version 7 Data Sharing: Planning and Administration,* SC26-9935
- *DB2 UDB for OS/390 and z/OS Version 7 Image, Audio, and Video Extenders,* SC26-9947
- *DB2 UDB for OS/390 and z/OS Version 7 ODBC Guide and Reference,* SC26-9941
- *DB2 UDB for OS/390 and z/OS Version 7 XML Extender Administration and Reference,* SC26-9949

- *MVS/ESA SP V5 Programming: Authorized Assembler Services Guide,* GC28-1467-02

## E.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://ibm.com/software/data/db2imstools/` DB2 and IMS Tools
- `http://ibm.com/software/data/db2/os390/sqlproc` SQL Procedures
- `http://java.sun.com/products/jdbc` JDBC
- `http://ibm.com/software/data/iminer/fortext` Intelligent Miner for Text
- `http://ibm.com/software/data/db2/extenders` DB2 Extenders
- `http://ibm.com/storage/hardsoft/diskdrls/technology.htm` ESS
- `http://www.opengroup.org` DRDA standards

# How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

| | **e-mail address** |
|---|---|
| In United States or Canada | pubscan@us.ibm.com |
| Outside North America | Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl |

- **Telephone Orders**

| | |
|---|---|
| United States (toll free) | 1-800-879-2755 |
| Canada (toll free) | 1-800-IBM-4YOU |
| Outside North America | Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl |

- **Fax Orders**

| | |
|---|---|
| United States (toll free) | 1-800-445-9269 |
| Canada | 1-403-267-4455 |
| Outside North America | Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl |

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

**IBM Intranet for Employees**

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

© Copyright IBM Corp. 2001

## IBM Redbooks fax order form

**Please send me the following:**

| Title | Order Number | Quantity |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

| | | |
|---|---|---|
| First name | Last name | |
| Company | | |
| Address | | |
| City | Postal code | Country |
| Telephone number | Telefax number | VAT number |

☐ Invoice to customer number

☐ Credit card number

| | | |
|---|---|---|
| Credit card expiration date | Card issued to | Signature |

**We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.**

# Glossary

The following terms and abbreviations are defined as they are used in the DB2 library. If you do not find the term you are looking for, refer to the index or to IBM Dictionary of Computing. Also, it brings term and abbreviations related to Java and other products, mentioned in this book.

## A

**abstract class** A class that provides common information for subclasses, and therefore cannot be instantiated. Abstract classes provide at least one abstract method.

**abstract method** A method with a signature, but no implementation. You provide the implementation of the method in the subclass of the abstract class that contains the abstract method.

**abstract window toolkit (AWT)** The Abstract Window Toolkit API provides a layer between the application and the host's windowing system. It enables programmers to port Java applications from one window system to another. The AWT provides access to basic interface components such as events, color, fonts, and controls such as button, scroll bars, text fields, frames, windows, dialogs, panels, canvases, and check boxes.

**actual parameter list** Parameters specified in a call to a method. See also formal parameter list.

**address space**. A range of virtual storage pages identified by a number (ASID) and a collection of segment and page tables which map the virtual pages to real pages of the computer's memory.

**address space connection**. The result of connecting an allied address space to DB2. Each address space containing a task connected to DB2 has exactly one address space connection, even though more than one task control block (TCB) can be present. See allied address space and task control block.

**AFC** See *Application Foundation Classes*.

**allied address space**. An area of storage external to DB2 that is connected to DB2 and is therefore capable of requesting DB2 services.

**American National Standards Institute (ANSI)**. An organization consisting of producers, consumers, and general interest groups, that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States.

**ANSI**. American National Standards Institute.

**API** See *Application Program Interface.*

**applet** A Java program designed to run within a Web browser. Contrast with application.

**application** (1) A program or set of programs that perform a task; for example, a payroll application. (2) In Java programming, a self-contained, stand-alone Java program that includes a static main method. It does not require an applet viewer. Contrast with applet.

**Application Foundation Classes (AFCs)** Microsoft's version on the Java Foundation Classes (JFCs). AFCs deliver similar functions to JFCs but only work on Windows 32-bit platforms.

**application plan** The control structure produced during the bind process and used by DB2 to process SQL statements encountered during statement execution.

**application program interface (API)** A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or licensed program.

**application requester (AR)** See requester.

**AR** application requester. See requester.

**ASCII** (1) American Standard Code for Information Interchange.A standard assignment of 7-bit numeric codes to characters. See also *Unicode*. (2) An encoding scheme used to represent strings in many environments, typically on PCs and workstations. Contrast with EBCDIC.

**attachment facility** An interface between DB2 and TSO, IMS, CICS, or batch address spaces. An attachment facility allows application programs to access DB2.

**authorization ID** A string that can be verified for connection to DB2 and to which a set of privileges are allowed. It can represent an individual, an organizational group, or a function, but DB2 does not determine this representation.

**AWT** See *Abstract Window Toolkit*.

## B

**base table** (1) A table created by the SQL CREATE TABLE statement that is used to hold persistent data. Contrast with result table and temporary table. (2) A table containing a LOB column definition. The actual LOB column data is not stored along with the base table. The base table contains a row identifier for each row and an indicator column for each of its LOB columns. Contrast with auxiliary table.

**base type** In Java, a type establishes an interface to anything inherited from itself. See *type*, *derived type*.

**bean** A definition or instance of a JavaBeans component. See *JavaBeans*.

© Copyright IBM Corp. 2001

**BeanInfo** (1) A Java class that provides explicit information about the properties, events, and methods of a bean class. (2) In the VisualAge for Java Integrated Development Environment, a page in the class browser that provides bean information.

**binary large object (BLOB)** See BLOB.

**bind** The process by which the output from the DB2 precompiler is converted to a usable control structure called a package or an application plan. During the process, access paths to the data are selected and some authorization checking is performed.

**automatic bind**. (More correctly automatic rebind). A process by which SQL statements are bound automatically (without a user issuing a BIND command) when an application process begins execution and the bound application plan or package it requires is not valid.

**dynamic bind**. A process by which SQL statements are bound as they are entered.

**incremental bind**. A process by which SQL

statements are bound during the execution of an application process, because they could not be bound during the bind process, and VALIDATE(RUN) was specified.

**static bind**. A process by which SQL statements are bound after they have been precompiled. All static SQL statements are prepared for execution at the same time. Contrast with dynamic bind.

**BLOB** A sequence of bytes, where the size of the sequence ranges from 0 bytes to 2 GB - 1. Such a string does not have an associated CCSID. The size of binary large object values can be anywhere up to 2 GB - 1.

**browser** (1) In VisualAge for Java, a window that provides information on program elements. There are browsers for projects, packages, classes, methods, and interfaces. (2) An Internet-based too that lets users browse Web sites.

**built-in function** A function that is supplied by DB2. Contrast with user-defined function.

**business object** (1) An object that represents a business function. Business objects contain attributes that define the state of the object, and methods that define the behavior of the object. A business object also has relationships with other business objects. Business objects can be used in combination to perform a desired task. Typical examples of business objects are Customer, Invoice, or Account. (2) In the Enterprise Access Builder, a class that implements the IBusinessObject interface. Business objects are used to map interactions with an existing home.

**bytecode** Machine-independent code generated by the Java compiler and executed by the Java interpreter.

## C

**CAF** Call attachment facility.

**call attachment facility (CAF)** A DB2 attachment facility for application programs running in TSO or MVS batch. The CAF is an alternative to the DSN command processor and allows greater control over the execution environment.

**call level interface (CLI)** A callable application program interface (API) for database access, which is an alternative to using embedded SQL. In contrast to embedded SQL, DB2 CLI does not require the user to precompile or bind applications, but instead provides a standard set of functions to process SQL statements and related services at run time.

**cast function** A function used to convert instances of a (source) data type into instances of a different (target) data type. In general, a cast function has the name of the target data type. It has one single argument whose type is the source data type; its return type is the target data type.

**casting** Explicitly converting an object or primitive's data type.

**catalog** In DB2, a collection of tables that contains descriptions of objects such as tables, views, and indexes.

**catalog table** Any table in the DB2 catalog.

**C++ Access Builder** A VisualAge fro Java, Enterprise Edition tool that generates beans and C++ wrappers that let your Java programs access C++ DLLs.

**character large object (CLOB)** See CLOB.

**class** An encapsulated collection of data and methods to operate on the data. A class may be instantiated to produce an object that is an instance of the class.

**class hierarchy** The relationships between classes that share a single inheritance. All Java classes inherit from the Object class.

class method Methods that apply to the class as a whole rather than its instances (also called a *static method*).

**class path** When running a program in VisualAge for Java, a list of directories and JAR files that contain resource files or Java classes that a program can load dynamically at run time. A program's class path is set in its Properties notebook.

**CLASSPATH** In your deployment environment, the environment variable keyword that specifies the directories in which to look for class and resource files.

**class variable** Variables that apply to the class as a whole rather than its instances (also called a *static field*).

**CLI** See call level interface.

**client** (1)A networked computer in which the IDE is connected to a repository on a team server. (2) See requester.

**CLOB** A sequence of bytes representing single-byte characters or a mixture of single and double-byte characters where the size can be up to 2 GB - 1. Although the size of character large object values can be anywhere up to 2 GB - 1, in general, they are used whenever a character string might exceed the limits of the VARCHAR type.

**codebase** An attribute of the <APPLET> tag that provides the relative path name for the classes. Use this attribute when your class files reside in a different directory than your HTML files.

**column function** An SQL operation that derives its result from a collection of values across one or more rows. Contrast with scalar function.

**commit** The operation that ends a unit of work by releasing locks so that the database changes made by that unit of work can be perceived by other processes.

**Common Connector Framework** In the Enterprise Access Builder, interface and class definitions that provide a consistent means of interacting with enterprise resources (for example, CICS and Encina transactions) from any Java execution environment.

**Common Object Request Broker Architecture (CORBA)** Common Object Request Broker Architecture. A specification produced by the Object Management Group (OMG) that presents standards for various types of object request brokers (such as client-resident ORBs, server-based ORBs, system-based ORBs, and library-based ORBs). Implementation of CORBA standards enables object request brokers from different software vendors to interoperate.

**Common RFC Interface for Java** A set of Java interfaces and classes that defines a middleware-independent layer to access R/3 systems from Java. If applications are built on top of this interface, they can leverage different middleware at run time without recoding. The generated beans are based on this interface and provide the same flexibility.

**common server** Describes the set of DB2 products that run on various platforms and have the same source code. These platforms include OS/2, Windows, and UNIX.

**component model** An architecture and an API that allows developers to define reusable segments of code that can be combined to create a program. VisualAge for Java uses the JavaBeans component model.

**composite bean** A bean that can contain both visual and nonvisual components. A composite bean is composed of embedded beans.

**connection** In the VisualAge for Java Visual Composition Editor, a visual link between two components that represents the relationship between the components. Each connection has a source, a target, and other properties.

**connection handle** The data object that contains information associated with a connection managed by DB2 CLI. This includes general status information, transaction status, and diagnostic information.

**Console** In VisualAge for Java, the window that acts as the standard input (System.in) and standard output (System.out) device for programs running in the VisualAge for Java environment. constructor A method called to set up a new instance of a class.

**constant** A language element that specifies an unchanging value. Constants are classified as string constants or numeric constants. Contrast with variable.

**container** A component that can hold other components. In Java, examples of containers include applets, frames, and dialogs. In the Visual Composition Editor, containers can be graphically represented and generated.

**context** The application's logical connection to the data source and associated internal DB2 ODBC connection information that allows the application to direct its operations to a data source. A DB2 ODBC context represents a DB2 thread.

**cookie** (1) A small file stored on an individual's computer; this file allows a site to tag the browser with a unique identification. When a person visits a site, the site's server requests a unique ID from the person's browser. If this browser does not have an ID, the server delivers one. On the Wintel platform, the cookie is delivered to a file called'cookies.txt,' and on a Macintosh platform, it is delivered to 'MagicCookie.' Just as someone can track the origin of a phone call with Caller ID, companies can use cookies to track information about behavior. (2) Persistent data stored by the client in the Servlet Builder.

**CORBA** Common Objects Request Broker Architecture.

**Core API** Part of the minimal set of APIs that form the standard Java Platform. Core APIs are available on the Java Platform regardless of the underlying operating system. The Core API grows with each release of the JDK; the current core API is based on JDK 1.1. Also called *core classes*.

**cursor** A named control structure used by an application program to point to a row of interest within some set of rows, and to retrieve rows from the set, possibly making updates or deletions.

## D

**Data Access Bean** In the VisualAge for Java Visual Composition Editor, a bean that accesses and

manipulates the content of JDBC/ODBC-compliant relational databases.

**Data Access Builder** A VisualAge for Java Enterprise tool that generates beans to access and manipulate the content of JDBC/ODBC-compliant relational databases.

**database management system (DBMS)** A software system that controls the creation, organization, and modification of a database and access to the data stored within it.

**data source** A local or remote relational or non-relational data manager that is capable of supporting data access via an ODBC driver which supports the ODBC APIs. In the case of DB2 for OS/390, the data sources are always relational database managers.

**DBCLOB** A sequence of bytes representing double-byte characters where the size can be up to 2 gigabytes. Although the size of double-byte character large object values can be anywhere up to 2 gigabytes, in general, they are used whenever a double-byte character string might exceed the limits of the VARGRAPHIC type.

**DBMS** Database management system.

**DB2 thread** The DB2 structure that describes an application's connection, traces its progress, processes resource functions, and delimits its accessibility to DB2 resources. and services.

**debugger** A component that assists in analyzing and correcting coding errors.

**declaration** Statement that creates an identifier and its attributes, but does not reserve storage or provide an implementation.

**definition** Statement that reserves storage or provides an implementation.

**deprecation** An obsolete component that may be deleted from a future version of a product.

**derived type** In Java, a type that overrides the definitions of a base type to provide unique behavior. The derived type extends the base type.

**dipping** A metaphor, introduced by BeanExtender on alphaWorks, for modifying a component by hooking a special kind of Java bean onto it. Dipping lets you add new behavior or modify the Java bean's existing behavior without having to manipulate the Java bean's code. A dip is a special kind of Java bean that can be hooked on to another Java bean; it is the new feature you want to add to the component. Software examples of dips include printing and security. Dippable Java beans can have one or more dips connected to them. Almost any Java bean or class can be made dippable by extending it, a process called morphing.

**dip** A special kind of Java bean that can be hooked on to another Java bean; the new feature you want to add to the component. Software examples of dips include printing and security.

distributed processing Processing that takes place across two or more linked systems.

**distinct type** A user-defined data type that is internally represented as an existing type (its source type), but is considered to be a separate and incompatible type for semantic purposes.

**distributed relational database architecture (DRDA)** A connection protocol for distributed relational database processing that is used by IBM's relational database products. DRDA includes protocols for communication between an application and a remote relational database management system, and for communication between relational database management systems.

**DLL (dynamic link library**) A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a dynamic link library can be shared by several applications simultaneously. The DLLs. Enterprise Access Builders also generate platform-specific DLLs for the workstation and OS/390 platforms.

**double-byte character large object (DBCLOB)** See DBCLOB.

**double precision** A floating-point number that contains 64 bits. See also *single precision*.

**DRDA** Distributed relational database architecture.

**dynamic SQL** SQL statements that are prepared and executed within an application program while the program is executing. In dynamic SQL, the SQL source is contained in host language variables rather than being coded into the application program. The SQL statement can change several times during the application program's execution.

## E

**EBCDIC** Extended binary coded decimal interchange code. An encoding scheme used to represent character data in the MVS, VM, VSE, and OS/400Ñ environments. Contrast with ASCII.

**EAB** See *Enterprise Access Builder*.

**e-business** Either (a) the transaction of business over an electronic medium such as the Internet or (b) a business that uses Internet technologies and network computing in their internal business processes (via intranets), their business relationships (via extranets), and the buying and selling of goods, services, and information (via electronic commerce.)

**e-commerce** The subset of e-business that involves the exchange of money for goods or services purchased over an electronic medium such as the Internet.

**EmbeddedJava** An API and application environment for high-volume embedded devices, such as mobile phones, pagers, process control, instrumentation, office peripherals, network routers and network switches. EmbeddedJava applications run on real-time operating systems and are optimized for the constraints of small-memory footprints and diverse visual displays.

**embedded SQL** SQL statements coded within an application program. See static SQL.

**encapsulation** The grouping of both data and operations into neat, manageable units that can be developed, tested, and maintained independently of one another. Such grouping is a powerful technique for building better software. The object manages its own resources and limits their visibility.

**enclave** In Language Environment for MVS & VM, an independent collection of routines, one of which is designated as the main routine. An enclave is similar to a program or run unit.

**Enterprise Access Builder (EAB)** Feature of Visual Age for Java, Enterprise Edition, that creates connectors to enterprise server products such as CICS, Encina, IMS TOC, and MQSeries. Enterprise Edition See *VisualAge for Java, Enterprise Edition*.

**Enterprise Java** Includes Enterprise JavaBeans as well as open API specifications for: database connectivity, naming and directory services, CORBA/IIOP interoperability, pure Java distributed computing, messaging services, managing system and network resources, and transaction services.

**Enterprise JavaBeans** A cross-platform component architecture for the development and deployment of multi-tier, distributed, scalable, object-oriented Java applications.

**Enterprise ToolKit** A set of VisualAge for Java Enterprise tools that enable you to develop Java code that is targeted to specific platforms, such as AS/400, OS/390, OS/2, AIX, and Windows.

**Entry Edition** See *VisualAge for Java, Entry Edition.*

**equi-join** A join operation in which the join-condition has the form expression = expression.

**event** An action by a user, program, or system that may trigger specific behavior. In the JDK, events notify the relevant listener classes to take appropriate action.

**environment** A collection of names of logical and physical resources that are used to support the performance of a function.

**environment handle** In DB2 ODBC, the data object that contains global information regarding the state of the application. An environment handle must be allocated before a connection handle can be allocated. Only one environment handle can be allocated per application.

**exception** An exception is an object that has caused some sort of new condition, such as an error. In Java, *throwing* an exception means passing that object to an interested party; a signal indicates what kind of condition has taken place. *Catching* an exception means receiving the sent object. *Handling* this exception usually means taking care of the problem after receiving the object, although it might mean doing nothing (which would be bad programming practice).

**executable content** Code that runs from within an HTML file (such as an applet).

**extends** A subclass or interface extends a class or interface if it add fields or methods, or overrides its methods. See also *derived type.*

**external function** A function for which the body is written in a programming language that takes scalar argument values and produces a scalar result for each invocation. Contrast with sourced function and built-in function.

## F

**factory** A bean that dynamically creates instances of beans.

**field** A data object in a class; for example, a variable.

**first tier** The client; the hardware and software with which the end user interacts.

**framework** A set of object classes that provide a collection of related functions for a user or piece of software.

**free-form surface** In the VisualAge for Java Visual Composition Editor, the large, open area where you can work with visual and nonvisual beans. You add, remove, and connect beans on the free-form surface.

**File Transfer Protocol (FTP)** In the Internet suite of protocols, an application layer protocol that uses TCP and Telnet services to transfer bulk-data files between machines or hosts.

**foreign key** A key that is specified in the definition of a referential constraint. Because of the foreign key, the table is a dependent table. The key must have the same number of columns, with the same descriptions, as the primary key of the parent table.

**form data** A generated class representing the HTML form elements in a visual servlet.

**formal parameter list** Parameters specified in a method's definition. See also *actual parameter list.*

**FTP** See *File Transfer Protocol*.

**full outer join** The result of a join operation that includes the matched rows of both tables being joined and preserves the unmatched rows of both tables. See also join.

**function** A specific purpose of an entity or its characteristic action such as a column function or

scalar function. (See column function and scalar function.). Furthermore, functions can be user-defined, built-in, or generated by DB2. (See built-in function, cast function, user-defined function, external function, sourced function.)

## G

**garbage collection** Java's ability to clean up inaccessible unused memory areas ("garbage") on the fly. Garbage collection slows performance, but keeps the machine from running out of memory.

**Graphical User Interface (GUI)** A type of computer interface consisting of a visual metaphor of a real-world scene, often of a desktop. Within that scene are icons, representing actual objects, that the user can access and manipulate with a pointing device.

## H

**handle** In DB2 CLI, a variable that refers to a data structure and associated resources. See statement handle, connection handle, and environment handle.

**hierarchy** The order of inheritance in object-oriented languages. Each class in the hierarchy inherits attributes and behavior from its superclass, except for the top-level Object class.

**HotJava** A Java-enabled Web and intranet browser developed by Sun Microsystems, Inc. HotJava is written in Java. (Definition copyright 1996-1999 Sun Microsystems, Inc. All Rights Reserved. Used by permission.)

**Hypertext Markup Language (HTML)** A file format, based on SGML, for hypertext documents on the Internet. Allows for the embedding of images, sounds, video streams, form fields and simple text formatting. References to other objects are embedded using URLs, enabling readers to jump directly to the referenced document.

**Hypertext Transfer Protocol (HTTP)** The Internet protocol, based on TCP/IP, used to fetch hypertext objects from remote hosts.

## I

**IDE** See *Integrated Development Environment*.

**identifier** The name of an item in a program.

**IDL (Interface Definition Language)** In CORBA, a declarative language that is used to describe object interfaces, without regard to object implementation.

**IDL Development Environment** In VisualAge for Java, an integrated IDL and Java development environment. The IDL Development Environment allows you to work with IDL source code in the multipane IDLs page and generate Java code using an IDL-to-Java compiler.

**IDL group** A container used to hold IDL objects in the IDL Development Environment. It is similar to a file system directory.

**IIOP (Internet Inter-ORB Protocol)** A communications standard for distributed objects that reside in Web or enterprise computing environments.

**InfoBus** A technology for flexible, vendor-independent data exchange which is used by eSuite and can be used by other applications to exchange data with eSuite and other InfoBus-enabled applications. The 100% Pure Java release and the InfoBus specification are available for free download from: `http://java.sun.com/beans/infobus`

**inheritance** The ability to create a subclass that automatically inherits properties and methods from its superclass. See also *hierarchy*.

**initialization file** For DB2 ODBC applications, a file containing values that can be set to adjust the performance of the database manager.

**inner join** The result of a join operation that includes only the matched rows of both tables being joined. See also join.

**Inspector** In VisualAge for Java, a window in which you can evaluate code fragments in the context of an object, look at the entire contents of an object and its class, or access and modify the fields of an object.

**instance** The specific representation of a class, also called an object.

**instance method** A method that applies and operates on objects (usually called simply a method). Contrast with *class method*.

**instance variable** A variable that defines the attributes of an object. The class defines the instance variable's type and identifier, but the object sets and changes its values.

**Integrated Development Environment (IDE**) In VisualAge for Java, the set of windows that provide the user with access to development tools. The primary windows are the Workbench, Log, Console, Debugger, and Repository Explorer.

**interface** A list of methods that enables a class to implement the interface itself by using the implements keyword. The Interfaces page in the Workbench lists all interfaces in the workspace.

**Internet Protocol (IP)** In the Internet suite of protocols, a connectionless protocol that routes data through a network or interconnected networks. IP acts as an intermediary between the higher protocol layers and the physical network. However, this protocol does not provide error recovery and flow control and does not guarantee the reliability of the physical network.

**Internet Inter-ORB Protocol (IIOP) Access Builder** A tool that edits and generates CORBA-compliant

Java modules. See *Common Object Request Broker* Architecture (CORBA).

**interpreter** A tool that translates and executes code line-by-line.

**introspection** For a JavaBean to be reusable in development environments, there needs to be a way to query what the bean can do in terms of the methods it supports and the types of event it raises and listens for. Introspection allows a builder tool to analyze how a bean works.

**IP** See *Internet Protocol*.

## J

**JAE** See *Java Application Environment*.

**JAR file format** JAR (Java Archive) is a platform-independent file format that aggregates many files into one. Multiple Java applets and their requisite components (.class files, images, sounds and other resource files) can be bundled in a JAR file and subsequently downloaded to a browser in a single HTTP transaction.

**Java** An object-oriented programming language for portable, interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated. The Java environment consists of the JavaOS, the Virtual Machines for various platforms, the object-oriented Java programming language, and several class libraries.

**Java Application Environment (JAE)** The source code release of the Java (TM) Development Kit. (Definition copyright 1996-1999 Sun Microsystems, Inc. All Rights Reserved. Used by permission.)

**Java Development Kit (JDK)** The Java Development Kit is the set of Java technologies made available to licensed developers by Sun Microsystems. Each release of the JDK contains the following: the Java Compiler, Java Virtual Machine, Java Class Libraries, Java Applet Viewer, Java Debugger, and other tools.

**Java Foundation Classes (JFC)** Developed by Netscape, Sun, and IBM, JFCs are building blocks that are helpful in developing interfaces to Java applications. They allow Java applications to interact more completely with the existing operating systems. Also called Swing Set.

**Java IDL** Java IDL is a language-neutral way to specify an interface between an object and its client on a different platform. Provides interoperability and integration with CORBA, the industry standard for distributed computing, allowing developers to build Java applications that are integrated with heterogeneous business information assets.

**Java Management Application Programming Interface (JMAPI)** A specification proposed by Sun Microsystems that defines a core set of application programming interfaces for developing tightly integrated system, network, and service management applications. The application programming interfaces could be used in diverse computing environments that encompass many operating systems, architectures, and network protocols.

**Java Media and Communications APIs** Allows developers to integrate a wide range of media types into their Web pages, applets, and applications. Includes: Media, Sound, Animation, 2D, 3D, Telephony, Speech and Collaboration.

**Java Media Framework (JMF)** Java Media Framework API specifies a unified architecture, messaging protocol and programming interface for media players, capture and conferencing. JMF provides a set of building blocks useful by other areas of the Java Media API suite. For example, the JMF provides access to audio devices in a cross-platform, device-independent manner, which is required by both the Java Telephony and the Java Speech APIs. JMF will be published as three APIs: the Java Media Player, Java Media Capture, and Java Media Conference.

**Java Naming and Directory Interface (JNDI)** A set of APIs that assist with the interfacing to multiple naming and directory services. (Definition copyright 1996-1999 Sun Microsystems, Inc. All Rights Reserved. Used by permission.)

**Java Native Interface (JNI)** A native programming interface that allows Java code running inside a Java Virtual Machine (VM) to interoperate with applications and libraries written in other programming languages, such as C and C++.

**Java Platform** The Java Virtual Machine and the Java Core classes make up the Java Platform. The Java Platform provides a uniform programming interface to a 100%. Pure Java program regardless of the underlying operating system. (Definition copyright 1996-1999 Sun Microsystems, Inc. All Rights Reserved. Used by permission.)

**Java Record Editor** An editor that allows you to construct and refine dynamic record types.

**Java Record Framework** A Java framework that describes and converts record data.

**Java Remote Method Invocation (RMI)** Java Remote Method Invocation is method invocation between peers, or between client and server, when applications at both ends of the invocation are written in Java. Included in JDK 1.1.

**Java Runtime Environment (JRE)** A subset of the Java Development Kit for end-users and developers who want to redistribute the JRE. The JRE consists of the Java Virtual Machine, the Java Core Classes, and supporting files. (Definition copyright 1996-1999 Sun Microsystems, Inc. All Rights Reserved. Used by permission.)

**Java Security API** A framework for developers to include security functionality in their applets and applications. Includes: cryptography with digital signatures, encryption, and authentication. An intermediate subset of the Security API known as "Security and Signed Applets" is included in JDK 1.1.

**Java Server** An extensible framework that enables and eases the development of Java-powered Internet and intranet servers. The APIs provide uniform and consistent access to the server and administrative system resources required for developers to quickly develop their own Java servers.

**Java Virtual Machine (JVM)** A software implementation of a central processing unit (CPU) that runs compiled Java code (applets and applications).

**JavaBeans** Java's component architecture, developed by Sun, IBM, and others. The components, called Java beans, can be parts of Java programs, or they can exist as self-contained applications. Java beans can be assembled to create complex applications, and they can run within other component architectures (such as ActiveX and OpenDoc).

**JavaDoc** Sun's tool for generating HTML documentation on classes by extracting comments from the Java source code files.

**JDBC (Java Database Connectivity)** In the JDK, the specification that defines an API that enables programs to access databases that comply with this standard.

**JavaObjs** In Remote Method Invocation, the name of the user-defined default file that contains a list of server objects to be instantiated when the Remote Object Instance Manager is started.

**JavaOS** A basic, small-footprint operating system that supports Java. Java OS was originally designed to run in small electronic devices like phones and TV remotes, but it is also being targeted for use in network computers (NCs).

**JavaScript** A scripting language used within an HTML page. Superficially similar to Java but JavaScript scripts appear as text within the HTML page. Java applets, on the other hand, are programs written in the Java language and are called from within HTML pages or run as stand-alone applications.

**JFC** See *Java Foundation Classes.*

**JIT** See *Just-In-Time Compiler.*

**JMF** See *Java Media Framework.*

**JNDI** See *Java Naming and Directory Interface.*

**JNI** See *Java Native Interface.*

**JRE** See *Java Runtime Environment*.

**Just-In-Time compiler (JIT)** A platform-specific software compiler often contained within JVMs. JITs compile Java bytecodes on-the-fly into native machine instructions, thereby reducing the need for interpretation.

**JVM** See *Java Virtual Machine*.

## L

**large object (LOB)** See LOB.

**left outer join** The result of a join operation that includes the matched rows of both tables being joined, and preserves the unmatched rows of the first table. See also join.

**link-edit** To create a loadable computer program using a linkage editor.

**linker** A computer program for creating load modules from one or more object modules or load modules by resolving cross references among the modules and, if necessary, adjusting addresses. In Java, the linker creates an executable from compiled classes.

**listener** In the JDK, a class that receives and handles events.

**load module** A program unit that is suitable for loading into main storage for execution. The output of a linkage editor.

**LOB** A sequence of bytes representing bit data, single-byte characters, double-byte characters, or a mixture of single and double-byte characters. A LOB can be up to 2GB -1 byte in length. See also BLOB, CLOB, and DBCLOB.

**LOB locator** A mechanism that allows an application program to manipulate a large object value in the database system. A LOB locator is a fullword integer value that represents a single LOB value. An application program retrieves a LOB locator into a host variable; it can then apply SQL operations to the associated LOB value using the locator.

**local** Refers to any object maintained by the local DB2 subsystem. A local table, for example, is a table maintained by the local DB2 subsystem. Contrast with remote.

**local variable** A variable declared and used within a method or block.

**Log** In the VisualAge for Java IDE, the window that displays messages and warnings during development.

## M

**member** In the Java language, an item belonging to a class, such as a field or method.

**method** A fragment of Java code within a class that can be invoked and passed a set of parameters to perform a specific task.

**middleware** A layer of software that sits between a database client and a database server, making it easier for clients to connect to heterogeneous databases.

**middle tier** The hardware and software that resides between the client and the enterprise server resources and data. The software includes a Web server that receives requests from the client and invokes Java servlets to process these requests. The client communicates with the Web server via industry standard protocols such as HTTP and IIOP.

**morphing** The process of extending a Java bean to accept dips. Morphed Java beans are called dippable Java beans and can have one or more dips connected to them. Almost any Java bean or class can be made dippable. See *dipping*.

**multithreaded** A program where different parts can run at the same time without interfering with each other.

**multithreading** Multiple TCBs executing one copy of DB2 ODBC code concurrently (sharing a processor) or in parallel (on separate central processors).

**mutex** Pthread mutual exclusion; a lock. A Pthread mutex variable is used as a locking mechanism to allow serialization of critical sections of code by temporarily blocking the execution of all but one thread.

**MVS/ESA** Multiple Virtual Storage/Enterprise Systems Architecture.

## N

**native class** Machine-dependent C code that can be invoked from Java. For multi-platform work, the native routines for each platform need to be implemented.

**NCF** See *Network Computing Framework.*

**Network Computing Framework (NCF)** An architecture and programming model created to help customer and industry software development teams to design, deploy, and manage e-business solutions across the enterprise.

**Network News Transfer Protocol (NNTP)** In the Internet suite of protocols, a protocol for the distribution, inquiry, retrieval, and posting of news articles that are stored in a central database.

**nonvisual bean** A bean that is not visible to the end user in the graphical user interface, but is visually represented on the free-form surface of the Visual Composition Editor during development. Developers can manipulate nonvisual beans only as icons; that is, they cannot edit them in the Visual Composition Editor as they can edit visual beans. Examples of nonvisual beans include beans for business logic, communication access, and database queries.

**NNTP** See *Network News Transfer Protocol.*

**NUL** In C, a single character that denotes the end of the string.

**null** A special value that indicates the absence of information.

**NUL-terminated host variable** A varying-length host variable in which the end of the data is indicated by the presence of a NUL terminator.

**NUL terminator** In C, the value that indicates the end of a string. For character strings, the NUL terminator is X'00'.

## O

**object** The principal building block of object-oriented programs. Objects are software programming modules. Each object is a programming unit consisting of related data and methods.

**ODBC** See Open Database Connectivity.

**ODBC driver** A dynamically-linked library (DLL) that implements ODBC function calls and interacts with a data source.

**Open Database Connectivity (ODBC)** A Microsoft database application programming interface (API) for C that allows access to database management systems by using callable SQL. ODBC does not require the use of an SQL preprocessor. In addition, ODBC provides an architecture that lets users add modules called database drivers that link the application to their choice of database management systems at run time. This means that applications no longer need to be directly linked to the modules of all the database management systems that are supported.

**outer join** The result of a join operation that includes the matched rows of both tables being joined and preserves some or all of the unmatched rows of the tables being joined. See also join.

**ORB (Object Request Broker)** In object-oriented programming, software that serves as an intermediary by transparently enabling objects to exchange requests and responses.

**object-oriented design** A software design method that models the characteristics of abstract or real objects using classes and objects. Object-oriented design focuses on the data and on the interfaces to it. For instance, an "object-oriented" carpenter would be mostly concerned with the chair he was building, and secondarily with the tools used to make it; a "non-object-oriented" carpenter would think primarily of his tools. Object-oriented design is also the mechanism for defining how modules "plug and play." The object-oriented facilities of Java are essentially those of C++, with extensions from Objective C for more dynamic method resolution.

**overloading** The ability to have different methods with the same identifier, distinguished by their return type, and number and type of arguments.

**overriding** Implementing a method in a subclass that replaces a method in a superclass.

## P

**package** A program element that contains classes and interfaces.

**part** An existing, reusable software component. All parts created with the Visual Composition Editor conform to the JavaBeans component model, and are referred to as beans. See *visual bean* and *nonvisual bean*.

**persistence** In object models, a condition that allows instances of classes to be stored externally, for example in a relational database.

**Persistence Builder** In VisualAge for Java, a persistence framework for object models, which enables the mapping of objects to information stored in relational databases and also provides linkages to legacy data on other systems.

**plan** See application plan.

**plan name** The name of an application plan.

**POSIX** Portable Operating System Interface. The IEEE operating system interface standard which defines the Pthread standard of threading. See Pthread.

**precompilation** A processing of application programs containing SQL statements that takes place before compilation. SQL statements are replaced with statements that are recognized by the host language compiler. Output from this precompilation includes source code that can be submitted to the compiler and the database request module (DBRM) that is input to the bind process.

**prepare** The first phase of a two-phase commit process in which all participants are requested to prepare for commit.

**prepared SQL statement** A named object that is the executable form of an SQL statement that has been processed by the PREPARE statement.

**primary key** A unique, nonnull key that is part of the definition of a table. A table cannot be defined as a parent unless it has a unique key or primary key.

**process** A program executing in its own address space, containing one or more threads.

**Professional Edition** See *VisualAge for Java, Professional Edition.*

**program** In VisualAge for Java, a term that refers to both Java applets and applications.

**program element** In VisualAge for Java, a generic term for a project, package, class, interface, or method.

**project** In VisualAge for Java, the topmost kind of program element. A project contains Java packages.

**property** An initial setting or characteristic of a bean, for example, a name, font, text, or positional characteristic.

**Pthread** The POSIX threading standard model for splitting an application into subtasks. The Pthread standard includes functions for creating threads, terminating threads, synchronizing threads through locking, and other thread control facilities.

## R

**RDBMS** Relational database management system.

**relational database management system (RDBMS)**. A relational database manager that operates consistently across supported IBM systems.

**reentrant** Executable code that can reside in storage as one shared copy for all threads. Reentrant code is not self-modifying and provides separate storage areas for each thread. Reentrancy is a compiler and operating system concept, and reentrancy alone is not enough to guarantee logically consistent results when multithreading. See threadsafe.

**reference** An object's address. In Java, objects are passed by reference rather than by value or by pointers.

**remote** Refers to any object maintained by a remote DB2 subsystem; that is, by a DB2 subsystem other than the local one. A remote view, for instance, is a view maintained by a remote DB2 subsystem. Contrast with local.

**remote debugger** A debugging tool that debugs code on a remote platform.

**Remote Function Call (RFC)** SAP's open programmable interface. External applications and tools can call ABAB/4 functions from the SAP System. You can also call third party applications from the SAP System using RFC. RFC is a means for communication that allows implementation on all R/3 platforms.

**Remote Method Invocation (RMI)** RMI is a specific instance of the more general term RPC. RMI allows objects to be distributed over the network; that is, a Java program running on one computer can call the methods of an object running on another computer. RMI and java.net are the only 100% pure Java APIs for controlling Java objects in remote systems.

**Remote Object Instance Manager** In Remote Method Invocation, a program that creates and manages instances of server beans through their associated server-side server proxies.

**Remote Procedure Calls (RPC)** RPC is a generic term referring to any of a series of protocols used to execute procedure calls or method calls across a network. RPC allows a program running on one computer to call the services of a program running on another computer.

**repository** In VisualAge for Java, the permanent storage area containing all open and versioned editions of all program elements, regardless of whether they are currently in the workspace. The repository contains the source code for classes developed in (and provided with) VisualAge for Java, and the bytecode for classes imported from the file system. Every time you save a method in the IDE, it is automatically updated in the repository. See also *SCM repository* and *shared repository*.

**Repository Explorer** In VisualAge for Java, the window from which you can view and compare editions of program elements that are in the repository.

**requester** Also application requester (AR). The source of a request to a remote RDBMS, the system that requests the data.

**resource file** A non-code file that may be referred to from your Java program in VisualAge for Java. Examples include graphic and audio files.

**result set** The set of rows returned to a client application by a stored procedure.

**result set locator** A 4-byte value used by DB2 to uniquely identify a query result set returned by a stored procedure.

**result table** The set of rows specified by a SELECT statement.

**right outer join** The result of a join operation that includes the matched rows of both tables being joined and preserves the unmatched rows of the second join operand. See also join.

**RMI (Remote Method Invocation)** See *Remote Method Invocation.*

**RMI Access Builder** A VisualAge for Java Enterprise tool that generates proxy beans and associated classes and interfaces so you can distribute code for remote access, enabling Java-to-Java solutions.

**RMI compiler** The compiler that generates stub and skeleton files that facilitate RMI communication. This compiler can be automatically invoked by the RMI Access Builder, and can also be invoked from the Tools menu item.

**RMI registry** A server program that allows remote clients to get a reference to a server bean.

**rollback** The process of restoring data changed by SQL statements to the state at its last commit point. All locks are freed. Contrast with commit.

**RPC** See *Remote Procedure Calls*.

**runtime system** The software environment where compiled programs run. Each Java runtime system includes an implementation of the Java Virtual Machine.

## S

**sandbox** A restricted environment, provided by the Web browser, in which Java applets run. The sandbox offers them services and prevents them from doing anything naughty, such as doing file I/O or talking to strangers (servers other than the one from which the applet was loaded). The analogy of applets to children led to calling the environment in which they run the "sandbox."

**scalar function** An SQL operation that produces a single value from another value and is expressed as a function name followed by a list of arguments enclosed in parentheses. See also column function.

**SCM** See *Software Configuration Management.*

**SCM repository** In VisualAge for Java, a generic term for the data store of any external software configuration management (SCM) tool. Some SCM tools refer to this as an archive.

**scope** Determines where an identifier can be used. In Java, instance and class variables have a scope that extends to the entire class. All other identifiers are local to the method where they are declared.

**Scrapbook** In VisualAge for Java, the window from which you can write, edit, and test fragments of code without having to define an encompassing class or method.

**Secure Socket Layer (SSL)** SSL is a security protocol which allows communications between a browser and a server to be encrypted and secure. SSL prevents eavesdropping, tampering or message forgery on your Internet or intranet network.

**security** Features in Java that prevent applets downloaded off the Web from deliberately or inadvertently doing damage. One such feature is the digital signature, which ensures that an applet came unmodified from a reputable source.

**serialization** Turning an object into a stream, and back again.

**server** The computer that hosts the Web page that contains an applet. The .class files that make up the applet, and the HTML files that reference the applet reside on the server. When someone on the Internet connects to a Web page that contains an applet, the server delivers the .class files over the Internet to the client that made the request. The server is also known as the originating host.

**server bean** The bean that is distributed using RMI services and is deployed on a server.

**servlet** Server-side programs that execute on and add function to Web servers. Java servlets allow for the creation of complicated, high-performance, cross-platform Web applications. They are highly extensible and flexible, making it easy to expand from

client or single-server applications to multi-tier applications.

**SGML** See *Standardized Generalized Markup Language.*

**single precision** A floating-point number that contains 32 bits. See also double precision.

**SmartGuide** In IBM software products, an active form of help that guides you through common tasks.

**Software Configuration Management (SCM)** The tracking and control of software development. SCM tools typically offer version control and team programming features.

**sourced function** A function that is implemented by another built-in or user-defined function already known to the database manager. This function can be a scalar function or a column (aggregating) function; it returns a single value from a set of values (for example, MAX or AVG). Contrast with external function and built-in function.

**source type** An existing type that is used to internally represent a distinct type.

**SQL** Structured Query Language. A language used by database engines and servers for data acquisition and definition.

**SQL authorization ID (SQL ID)** The authorization ID that is used for checking dynamic SQL statements in some situations.

**SQL Communication Area (SQLCA)** A structure

used to provide an application program with information about the execution of its SQL statements.

**SQL Descriptor Area (SQLDA)** A structure that describes input variables, output variables, or the columns of a result table.

**SQLCA** SQL communication area.

**SQLDA** SQL descriptor area.

**SQL/DS** SQL/Data System. Also known as DB2 for VSE & VM.

**SSL** See secure socket layer.

**Standardized Generalized Markup Language** An ISO/ANSI/ECMA standard that specifies a way to annotate text documents with information about types of sections of a document.

**statement handle** In DB2 ODBC, the data object that contains information about an SQL statement that is managed by DB2 CLI. This includes information such as dynamic arguments, bindings for dynamic arguments and columns, cursor information, result values and status information. Each statement handle is associated with the connection handle.

**static field** See *class variable.*

**static method** See *class method.*

**static SQL** SQL statements, embedded within a program, that are prepared during the program preparation process (before the program is executed). After being prepared, the SQL statement does not change (although values of host variables specified by the statement might change).

**stored procedure** A user-written application program, that can be invoked through the use of the SQL CALL statement.

**stream** A communication path between a source of information and its destination.

**Structured Query Language (SQL)** A standardized language for defining and manipulating data in a relational database.

**subclass** A class that inherits all the methods and variables of another class (its *superclass*). Its superclass might be a subclass of another class in the hierarchy.

**subtype** A type that extends another type (its *supertype*).

**superclass** A class that defines the methods and variables inherited by another class (its *subclass*).

**supertype** A type that is extended by another type (its *subtype*).

**Swing Set** A group of lightweight, ready-to-use components developed by JavaSoft. The components range from simple buttons to full-featured text areas to tree views and tabbed folders.

**synchronized** This Java keyword specifies that only one thread can run inside a method at once.

## T

**table** A named data object consisting of a specific number of columns and some number of unordered rows. Synonymous with base table or temporary table.

**task control block (TCB)** A control block used to communicate information about tasks within an address space that are connected to DB2. An address space can support many task connections (as many as one per task), but only one address space connection. See address space connection.

**TCB** MVS task control block.

**TCP/IP** See *Transmission Control Protocol based on IP.*

**temporary table** A table created by the SQL CREATE GLOBAL TEMPORARY TABLE statement that is used to hold temporary data. Contrast with result table.

**thin client** Thin client usually refers to a system that runs on a resource-constrained machine or that runs a small operating system. Thin clients don't require local system administration, and they execute Java applications delivered over the network.

**third tier** The third tier, or back end, is the hardware and software that provides database and transactional services. These back-end services are accessed through connectors between the middle-tier Web server and the third-tier server. Though this conceptual model depicts the second and third tier as two separate machines, the NCF model supports a logical three-tier implementation in which the software on the middle and third tier are on the same box.

**thread** A separate flow of control within a program.

**threadsafe** Characteristic of code that allows multithreading both by providing private storage areas for each thread, and by properly serializing shared (global) storage areas.

**timestamp** A seven-part value that consists of a date and time expressed in years, months, days, hours, minutes, seconds, and microseconds.

**trace** A DB2 facility that provides the ability to monitor and collect DB2 monitoring, auditing, performance, accounting, statistics, and serviceability (global) data.

**transaction** (1) In a CICS program, an event that queries or modifies a database that resides on a CICS server. (2) In the Persistence Builder, a representation of a path of code execution. (3) The code activity necessary to manipulate a persistent object. For example, a bank application might have a transaction that updates a company account.

**transient** This Java keyword specifies that a field is not included in the serial representation of an object. See *serialization*.

**Transmission Control Protocol based on IP** (1) A network communication protocol used by computer systems to exchange information across telecommunication links. (2) An Internet protocol that provides for the reliable delivery of streams of data from one host to another.

**type** In VisualAge for Java, a generic term for a class or interface.

## U

**UDF** User-defined function

**UDT** User-defined data type

**Uniform Resource Locator (URL)** The unique address that tells a browser how to find a specific Web page or file.

**Unicode** A 16-bit international character set defined by ISO 10646. See also ASCII.

**user-defined data type (UDT)** See distinct type.

**user-defined function (UDF)** A function defined to DB2 using the CREATE FUNCTION statement that can be referenced thereafter in SQL statements. A user-defined function can be either an external function or a sourced function. Contrast with built-in function.

**URL** See *Uniform Resource Locator.*

## V

**variable** (1) An identifier that represents a data item whose value can be changed while the program is running. The values of a variable are restricted to a certain data type. (2)A data element that specifies a value that can be changed. A COBOL elementary data item is an example of a variable. Contrast with constant.

**virtual machine** A software or hardware implementation of a central processing unit (CPU) that manages the resources of a machine and can run compiled code. See *Java Virtual Machine*.

**visual bean** In the Visual Composition Editor, a bean that is visible to the end user in the graphical user interface.

**Visual Composition Editor** In VisualAge for Java, the tool you can use to create graphical user interfaces from prefabricated beans, and to define relationships (called *connections*) between beans. The Visual Composition Editor is a page in the class browser.

**visual servlet** A servlet that is designed to be built using the VisualAge for Java Visual Composition Editor.

**VisualAge for Java, Enterprise Edition** An edition of VisualAge for Java that is designed for building enterprise Java applications, and has all of the Professional Edition features plus support for developers working in large teams, developing high-performance or heterogeneous applications, or needing to connect Java programs to existing enterprise systems.

**VisualAge for Java, Entry Edition** An edition of VisualAge for Java suitable for learning and building small projects of 500 classes or less. It is available as a no-charge download from VisualAge for Java and VisualAge Developer Domain Web sites.

**VisualAge for Java, Professional Edition** A complete Java development environment, including easy access to JDBC-enabled databases for building Java applications.

## W

**WebSphere** WebSphere is the cornerstone of IBM's overall Web strategy, offering customers a comprehensive solution to build, deploy and manage e-business Web sites. The product line provides companies with an open, standards-based, Web server deployment platform and Web site development and management tools to help accelerate the process of moving to e-business.

**world readable files** A permission level on Web servers specifying that files can be read by any user.

**World Wide Web** A network of servers that contain programs and files. Many of the files contain hypertext links to other documents available through the network.

**Workbench** In VisualAge for Java, the main window from which you can manage the workspace, create and modify code, and open browsers and other tools.

**workspace** The work area that contains the Java code that you are developing and the class libraries on which your code depends. Program elements must be added to the workspace from the repository before they can be modified.

**wrapper** Code that provides an interface for one program to access the functionality of another program.

**WWW** See *World Wide Web*.

## X

**X/Open** An independent, worldwide open systems organization that is supported by most of the world's largest information systems suppliers, user organizations, and software companies. X/Open's goal is to increase the portability of applications by combining existing and emerging standards.

**100% Pure Java** Sun Microsystems initiative to certify that applications and applets are purely Java-written.

## Abbreviations and acronyms

| | |
|---|---|
| ***AIX*** | Advanced Interactive eXecutive from IBM |
| ***APAR*** | authorized program analysis report |
| ***ARM*** | automatic restart manager |
| ***ASCII*** | American National Standard Code for Information Interchange |
| ***BLOB*** | binary large objects |
| ***CCSID*** | coded character set identifier |
| ***CCA*** | client configuration assistant |
| ***CFCC*** | coupling facility control code |
| ***CTT*** | created temporary table |
| ***CEC*** | central electronics complex |
| ***CD*** | compact disk |
| ***CF*** | coupling facility |
| ***CFRM*** | coupling facility resource management |
| ***CLI*** | call level interface |
| ***CLP*** | command line processor |
| ***CPU*** | central processing unit |
| ***CSA*** | common storage area |
| ***DASD*** | direct access storage device |
| ***DB2 PM*** | DB2 performance monitor |
| ***DBAT*** | database access thread |
| ***DBD*** | database descriptor |
| ***DBID*** | database identifier |
| ***DBRM*** | database request module |
| ***DCL*** | data control language |
| ***DDCS*** | distributed database connection services |
| ***DDF*** | distributed data facility |
| ***DDL*** | data definition language |
| ***DLL*** | dynamic load library manipulation language |
| ***DML*** | data manipulation language |
| ***DNS*** | domain name server |
| ***DRDA*** | distributed relational database architecture |
| ***DTT*** | declared temporary tables |
| ***EA*** | extended addressability |
| ***EBCDIC*** | extended binary coded decimal interchange code |
| ***ECS*** | enhanced catalog sharing |
| ***ECSA*** | extended common storage area |
| ***EDM*** | environment descriptor management |
| ***ERP*** | enterprise resource planning |
| ***ESA*** | Enterprise Systems Architecture |
| ***FDT*** | functional track directory |
| ***FTP*** | File Transfer Program |
| ***GB*** | gigabyte (1,073,741,824 bytes) |
| ***GBP*** | group buffer pool |
| ***GRS*** | global resource serialization |
| ***GUI*** | graphical user interface |
| ***HPJ*** | high performance Java |
| ***IBM*** | International Business Machines Corporation |
| ***ICF*** | integrated catalog facility |
| ***ICF*** | integrated coupling facility |
| ***ICMF*** | internal coupling migration facility |
| ***IFCID*** | instrumentation facility component identifier |
| ***IFI*** | instrumentation facility interface |
| ***IRLM*** | internal resource lock manager |
| ***ISPF*** | interactive system productivity facility |
| ***ISV*** | independent software vendor |
| ***I/O*** | input/output |
| ***ITSO*** | International Technical Support Organization |
| ***IVP*** | installation verification process |
| ***JDBC*** | Java Database Connectivity |
| ***JFS*** | journaled file systems |
| ***JVM*** | Java Virtual Machine |
| ***KB*** | kilobyte (1,024 bytes) |
| ***LOB*** | large object |
| ***LPL*** | logical page list |

© Copyright IBM Corp. 2001

| | |
|---|---|
| ***LPAR*** | logically partitioned mode |
| ***LRECL*** | logical record length |
| ***LRSN*** | log record sequence number |
| ***LVM*** | logical volume manager |
| ***MB*** | megabyte (1,048,576 bytes) |
| ***OBD*** | object descriptor in DBD |
| ***ODBC*** | Open Data Base Connectivity |
| ***OS/390*** | Operating System/390 |
| ***PAV*** | parallel access volume |
| ***PDS*** | partitioned data set |
| ***PSID*** | pageset identifier |
| ***PSP*** | preventive service planning |
| ***PTF*** | program temporary fix |
| ***PUNC*** | possibly uncommitted |
| ***QMF*** | Query Management Facility |
| ***RACF*** | Resource Access Control Facility |
| ***RBA*** | relative byte address |
| ***RECFM*** | record format |
| ***RID*** | record identifier |
| ***RRS*** | resource recovery services |
| ***RRSAF*** | resource recovery services attach facility |
| ***RS*** | read stability |
| ***RR*** | repeatable read |
| ***SDK*** | software developers kit |
| ***SMIT*** | System Management Interface Tool |
| ***SP*** | stored procedure |
| ***SRB*** | system resource block |
| ***TCB*** | task control block |
| ***WLM*** | workload manager |
| ***z/OS*** | operating system for the z/Architecture |

# Index

## Numerics


00C900CC 372
5655_E66 446
5655-D38 467
5655-E60 459
5655-E61 454
5655-E62 15, 192
5655-E63 15, 192
5655-E65 462
5655-E67 458
5655-E69 444
5655-E70 441
5655-E71 468
5655-E72 443
5655-F54 451
5655-F55 466
5697-E98 15, 193
5697-F56 464
5697-F57 456
5697-G52 447
5697-G63 450
5697-G64 452
-904 508


## A


Adding space to the workfiles 8
alias 4
AREST 365, 368
asynchronous INSERT preformatting 337
Audio Extender 171
AUTHCACH 346


## B


BUILD2 257, 263


## C


CANCEL THREAD NOBACKOUT 366, 371
CCSID_ENCODING 320
Charge features 15
CHECK constraint clause 21, 42, 43, 46
checkpoint frequency 358
CHKFREQ 348
CHKTIME 359, 362
Classic Connect 425, 426
close 336
Connection Pooling 118
connection pooling 115
Consistent restart 365, 373
Consistent restart enhancements 8
CONSTRAINT
    CREATE TABLEALTER TABLE 42
    CREATE TABLECREATE TABLE 42
constraint
    naming 42
    syntax 42
COPY 233
COPYTOCOPY 191
CopyToCopy 6, 271
Coupling Facility Name Class Queues 9
CREATE JAR 133
Cross Loader 6, 255


## D


Data Joiner 425
Data Sharing
    Bypass Group Attach 385
    Named Class Queues 383
Data sharing
    CF structure sizes 400
    DB2 Restart Light 398
    DB2 restart light 396
    Group Attach STARTECB support 390
    Group Attach support for DL/I Batch 385, 391
    IMMEDIATEWRITE Bind option 392
    IMMEDWRI(NO|PH1|YES) 393
    IMMEDWRITE BIND option 394
    IMMEDWRITE Bind option 392
    IMMEDWRITE(NO 395
    IMMEDWRITE(PH1) 393, 395
    incomplete units of recovery 403
    local connect using STARTECB 385
    message handling 403
Data spaces 332
Data Warehouse Center 416
DataGuide 418
DataPropagator 428
DB2 Admin Tool 439
DB2 Administration Tool 441
DB2 Archive Log Compression Tool 451
DB2 Automation Tool 450
DB2 Bind Manager 467
DB2 Change Accumulation Tool 466
DB2 Connect 183
DB2 Control Center 341, 407
DB2 DataPropagator 459
DB2 Estimator 410
DB2 Extenders 159
DB2 Forms 447
DB2 functions 320
DB2 High Performance Unload 444
DB2 Installer 408
DB2 Log Analysis Tool 446
DB2 Management Tools Package 14, 407
DB2 Net Search Extender 433
DB2 Object Comparison Tool 452
DB2 Object Restore 443
DB2 OLAP 418
DB2 Performance Monitor 454
DB2 Query Monitor 458
DB2 Recovery Manager 464
DB2 Row Archive Manager 462
DB2 SQL Performance Analyzer 456


© Copyright IBM Corp. 2001

DB2 Table Editor 447
DB2 tools 10
DB2 V7
  at a glance 2
  contents and packaging 1
  data sharing 381
  Extenders 159
  features 407
  installation 471
  language support 93
  migration and fallback 495
  network computing 275
  packaging 11
  performance and availability 329
  SQL enhancements 21
  tools 437
  Utilities 189
DB2 V7 at a glance 2
DB2 Visual Explain 410
DB2 Warehouse Manager 10, 16, 415
  OS/390 Agent 420
DB2 Warehouse Manager Agent 419
DB2 Web Query 468
DB2I 253
DBADM 4
DBET 368
DBM1 332
DEALLCT 348
DEFER YES 248
DISCARD 249
DISCARDDN 253
Document Access Definition 184
Document Type Definition 184
DRAIN 257, 265
DRAIN_WAIT 265
DROP CONSTRAINT clause 43
DROP JA 133
DSN1CHKR 261
DSN1COMP 261
DSN1COPY 233, 261
DSN1PRNT 261
DSN6SPRC 261
DSN8ED7 345
DSNDB06.SYSGRTNS 103
DSNDB06.SYSJAUXA 131
DSNDB06.SYSJAVA 127, 129
DSNDB06.SYSOBJ 46
DSNHDECP 410
DSNI032I 372, 375
DSNI033I 370, 372, 375
DSNJ031I 364
DSNJ153E 357
DSNJ154I 357
DSNJ370I 360
DSNJ372 362
DSNR042I 366, 370, 376
DSNTEJ6Z 345
DSNTIAUL 232
DSNTIJIN 103
DSNTIJMP 103
DSNTIJSG 345
DSNTIJTC 103
DSNTIJUZ 345
DSNTJSPP 130, 146, 149, 151, 152, 153
  input parameters 150
DSNU 253
DSNU1114I 264
DSNU111I 264
DSNU215I 376
DSNU364I 250
DSNU397I 250
DSNUPROC 208
DSNUTILS 208, 231, 253, 427
DSNV435I 366
DSNV439I 375
DSNWZP 345
DSNZPARM 410
DSSIZE 189
DSSTIME 348
DSZPARM 342
dynamic allocation of utility data set 196
Dynamic utility jobs 5, 190, 194

## E

EDMBFIT 347
EDMDSPAC 348
EDMPOOL 347
Enhanced management of constraints 3
Enhanced stored procedures 3
external-java-routine-name 142

## F

Fast SWITCH 257, 258, 259
feedback 567
FETCH FIRST n ROWS 21, 87, 88, 89
fewer sorts 339
FlashCopy 351
FROMCOPY 239
FROMCOPYDDN 240

## G

Global transactions 7
global transactions 275, 277, 278, 279
Glossary 545
GRANT USAGE ON JAR 135
Group Attach enhancements 9

## H

HTML 175

## I

I0001 259
IBM Net.Data 413
ICF catalog 336
IDFORE 348
IFCID 23 254
IFCID 24 254

IFCID 25 254
IFI 410
Image Extender 170
IMMEDWRITE bind option 9
IMS 10, 426, 464
IMS tools 10
INDDN 253
Index Advisor 341
Index SmartGuide 341
Information Catalog 417
inline STATISTICS 251
IN-list 333
INSERT processing 267
Installation and migration 11
IPREFIX 260
IRLMRWT 265

## J

J0001 259, 260
JAR file 127
JAR object 125
JARSCHEMA 128, 129, 130, 131
Java
  terminology 126
Java support 3
Java Virtual Machine 94
JavaBeans 110
JDBC 412
  Connection Poolin 115, 118
  Connection Pooling 104, 109, 112, 117, 118
  DataSource support 104, 112, 114
  Distributed transactions 109
  RowSet objects 110
JNDI 109, 115, 116
JVM 94, 124, 137, 157

## L

LBACKOUT 367
LEAFDIST 270
Limited fetch 3
LIST 195, 221
LISTDEF 195
  DB2I support 227
  example 210
  expansion 215
  syntax diagram 213
LOAD
  syntax enhancement 253
LOAD partition parallelism 191, 247
LOBVALA 346
LOBVALS 346
Log manager 350
Log manager enhancements 8
log read 356
LOGLOAD 359, 362
LPL 368
LRHEABRT 366
LRHEUNDO 366

## M

MAXDBAT 349
MAXRBLK 346
MAXRTU 348
MIN/MAX 340
MODIFY STATISTICS 191
More parallelism with LOAD with multiple inputs 5

## N

Net Search Extender 10, 17
Net.Data 12
Network monitoring 7, 326
Ngram index 168
NUMLKTS 347

## O

ODBC 414
ODBC calls
  scrollable cursors 76
OFFPOS 268
Online LOAD RESUME 6, 191, 266
  performance 266
  Restart 269
Online REORG
  BUILD2 263
  BUILD2 phase 257
  DRAIN and RETRY 257, 265
  fast SWITCH 257, 258
Online REORG enhancements 6, 191
Online subsystem parameters 8, 342
open 336
OPTIONS 195, 229

## P

packaging 11
packaging of utilities 190, 192
PAGESAVE 270
Parallel Index Build 250
Parallel LOAD jobs per partition 248
Parallelism for IN-list index access 333
Partition data sets parallel open 336
PCLOSET 348
Persistent structure size changes 9
pieces 189
Precise index 168
precompiler services 3, 93, 95, 96, 98, 99, 100
PREFORMAT 338
PREVIEW 229
  utility support 231
programming examples 521
PTASKROL 348
PUNC 354
PUNCHDDN 241

## Q

QMF 432
Query Management Facility 17

## R


REBUILD 248
REBUILD INDEX 251
RECOVER POSTPONED 365, 368
RECOVER POSTPONED CANCEL 370
Redbooks 539
Referenced Web sites 541
REFP 366, 368
Related publications 539
RELOAD 249, 250
  number of tasks 250
REORG 251
REORG UNLOAD EXTERNAL 232
replication 428
Restart Light 9
RESTP 365, 368
RESUME 362
RESUME YES 267
RETRY 257, 265
RETRY_DELAY 265
REVOKE USAGE ON JAR 135
REXX 101, 102, 414
REXX language support 13
RLFERR 348
Row expression in IN predicate 3
RUNSTATS statistics history 191


## S


scrollable cursor 21
  absolute moves 58
  CLOSE CURSOR 56
  DECLARE 54
  distributed processing 79
  FETCH 57
  FETCH ABSOLUTE 60
  FETCH AFTER 60
  FETCH BEFORE 60
  FETCH CURRENT 60
  FETCH FIRST 59
  FETCH keywords 59
  FETCH LAST 59
  FETCH NEXT 59
  FETCH PRIOR 59
  FETCH RELATIVE 60
  OPEN CURSOR 55
  relative moves 59
  SQLWARN flags 63
  stored procedures 75
Scrollable cursors 2
scrollable cursors
  ODBC
    calls 76
    SQLBulkOperations 78
    SQLFetchScroll 76
    SQLSetPos 77
Security enhancements 7
Self-referencing
  DELETE 91
  Restrictions on usage 92
  UPDATE 91
Self-referencing subselect on UPDATE or DELETE 4
SET LOG 361
SET LOG RESUME 353
  actions and messages 353
SET LOG SUSPEND 352
  actions and messages 352
  messages 353
  recommendations 354
SET LOGLOAD 359
SET SYSPARM 343
SGML 175
SHRLEVEL 233
SnapShot 351, 355
SORT 251
SORTBLD 251
SORTKEYS 248, 250, 251
SQL_FETCH_BY_BOOKMARK 78
SQLBulkOperations
  SQL_ADD 78
  SQL_DELETE_BY_BOOKMARK 78
  SQL_FETCH_BY_BOOKMARK 78
  SQL_UPDATE_BY_BOOKMARK 78
SQLFetchScroll
  SQL_ATTR_ROW_ARRAY_SIZE 76
  SQL_FETCH_ABSOLUTE 77
  SQL_FETCH_BOOKMARK 77
  SQL_FETCH_FIRST 77
  SQL_FETCH_LAST 77
  SQL_FETCH_NEXT 76
  SQL_FETCH_PRIOR 76
  SQL_FETCH_RELATIVE 76
SQLJ.INSTALL_JAR 132, 133
SQLJ.REMOVE_JAR 132, 134
SQLJ.REPLACE_JAR 132, 134
SQLSetPos
  SQL_DELETE 77
  SQL_POSITION 77
  SQL_REFRESH 77
  SQL_UPDATE 77
START DATABASE ACCESS(FORCE) 374
STATIME 348
Statistics history 6, 270
STEMMED FORM 434
Stored Procedure Builder 412
stored procedures
  scrollable cursor 75
STRIP 245
substitution variables 199
SUSPEND 362
Suspend update activity 351
SYSIBM.SYSCHECK 41
SYSIBM.SYSCHECKS 46
SYSIBM.SYSCOLUMN 41
SYSIBM.SYSCOLUMNS 46
SYSIBM.SYSJARCLASS_SOURCE 131
SYSIBM.SYSJARCONTENTS 130, 131
SYSIBM.SYSJARDATA 131
SYSIBM.SYSJAROBJECTS 129, 131
SYSIBM.SYSJAVAOPTS 130

SYSIBM.SYSKEYCOLUSE 46, 47, 48
SYSIBM.SYSRELS 41, 46
SYSIBM.SYSRESAUTH 135
SYSIBM.SYSROUTINES 127, 128, 133, 141
SYSIBM.SYSROUTINES_OPTS 94, 103
SYSIBM.SYSROUTINES_PSM 94
SYSIBM.SYSROUTINES_SRC 103
SYSIBM.SYSTABCONST 46, 47, 48
SYSIBM.SYSTABLES 38, 47, 49
  STATUS 49
  TABLESTATUS 38, 49
SYSLISTD 226
SYSTEMPL 226

## T

TEMPLATE 194
  DB2I support 227
  dispositions 202
  example 196
  restart support 202
  space allocation 201
  syntax diagram 206
TEMPLATE and LISTDEF 223
templates applicability 208
Text Extender 163
Text Extender indexing 167
tools 10
Transform correlated subqueries 334
TRUNCATE 245

## U

UNICODE
  CCSID_ENCODING 320
  CURRENT APPLICATION ENCODING SCHEME 315, 320
  Date data type 317
  DB2 Routines 320
  DB2 Utility support 322
  DECLARE HOST VARIABLE 315
  DESCRIBE statement 316
  EXECUTE IMMEDIATE statement 316
  full predicate support 317
  Host variables 315
  LIKE predicate 318
  Padding characters 318
  PLAN_TABLE 314
  PREPARE INTO statement 316
  PREPARE statement 316
  Resource Limit Specification Tables 314
  SQL limits 319
  time data type 317
UNICODE support 7
UNION
  basic predicates 30
  CREATE VIEW 21, 23, 25, 28
  DELETE statement 21, 22, 25
  EXISTS predicate 32
  Explain 36
    PARENT_QBLOCKNO 36
    TABLE_TYPE 36
  GROUP BY clause 24
  HAVING clause 24
  IN predicate 33
  INSERT statement 23, 34, 40
  quantified predicates 31
  SELECT statement 23, 24, 28
  table-spec 29
  UPDATE statement 21, 22, 23, 25, 34, 35
  WHERE clause 24
UNION ALL 21, 25, 26, 35, 36, 37
  fullselect 25
  subselects 25
UNION everywhere
  programming examples 521
Union everywhere 2
UNLDDN 241
UNLOAD 5, 190, 232
  a list of table spaces 242
  certain columns only 237
  certain rows only 237
  conversion options 234
  from a list of table spaces 236
  from copy data sets 239
  limit of rows 233
  LOBs and compressed data 246
  output data sets 241
  output formatting 243
  partitions 236, 242
  sampling rows 233
  SHRLEVEL 237
  specific tables 236
  syntax diagram 235
updatable DB2 subsystem parameters 517
URLGWTH 364
USS port 430
UTILTERM 258
UTIMOUT 265

## V

Video Extender 172
view 4
Visual Explain 270
VSAM 426
VWPMVS 423

## W

warning message 364
WebSphere 94, 104, 112, 116, 120
work files 377

## X

XML and DB2 181
XML and HTML 179
XML collection 185
XML column 184
XML Extender 4, 173

## Z

z/OS 331, 332
z/Series 332

# IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

| | |
|---|---|
| **Document Number**<br>**Redbook Title** | SG24-6121-00<br>DB2 UDB Server for OS/390 and z/OS Version 7 Presentation Guide |
| **Review** | |
| **What other subjects would you like to see IBM Redbooks address?** | |
| **Please rate your overall satisfaction:** | O Very Good O Good O Average O Poor |
| **Please identify yourself as belonging to one of the following groups:** | O Customer<br>O Business Partner<br>O Solution Developer<br>O IBM, Lotus or Tivoli Employee<br>O None of the above |
| **Your email address:**<br>The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities. | O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction. |
| **Questions about IBM's privacy policy?** | The following link explains how we protect your personal information.<br>**ibm.com**/privacy/yourprivacy/ |

© Copyright IBM Corp. 2001

# DB2 UDB Server for OS/390 and z/OS Version 7 Presentation Guide

(1.0” spine)
0.875”<->1.5”
460 <-> 788 pages

# DB2 UDB Server for OS/390 and z/OS Version 7

## Presentation Guide

Redbooks

**Description of new and enhanced functions**

**Evaluation of new features for customer usability**

**Guidance for migration planning**

IBM DB2 UDB Server for OS/390 and z/OS Version 7 is the eleventh release of DB2 for MVS. It brings to this platform the data support, application development, and query functionality enhancements for e-business while building upon the traditional capabilities of reliability and performance. The DB2 V7 environment is available for the S/390 and zSeries platforms, either for new installations of DB2, or for migrations from both DB2 for OS/390 Version 6 and DB2 for OS/390 Version 5 subsystems.

This IBM Redbook, in the format of a presentation guide, describes the enhancements made available with DB2 V7. These enhancements include a new feature, DB2 Warehouse Manager, which simplifies the design and deployment of a data warehouse within your S/390, as well as performance and availability delivered through new and enhanced utilities, dynamic changes to the value of many of the system parameters without stopping DB2, and the new Restart Light option for data sharing environments. Improvements in usability are provided with new and faster tools, the DB2 XML Extender support for the XML data type, scrollable cursors, support for UNICODE encoded data, support for COMMIT and ROLLBACK within a stored procedure, the option to eliminate the DB2 precompile step in program preparation, and the definition of view with the operators UNION or UNION ALL.

This book will help you understand why migrating to Version 7 of DB2 can be beneficial for your applications and your DB2 subsystems. It will provide sufficient information so you can start prioritizing the implementation of the new functions and evaluating their applicability in your DB2 environments.

**INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

**BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**
**ibm.com**/redbooks

SG24-6121-00 ISBN 0738418250