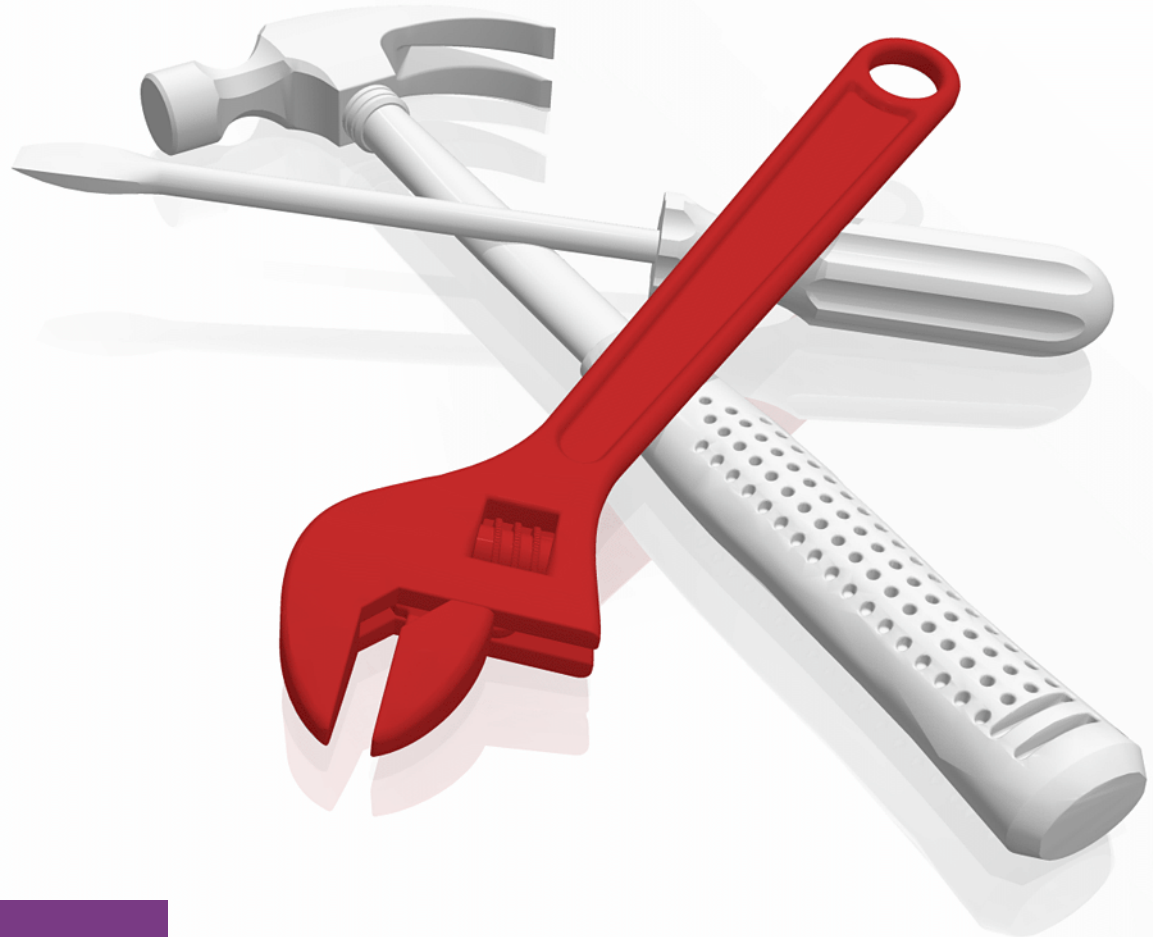


IBM Db2 13 for z/OS and More

IBM Db2 13 Project Team



Information Management



IBM Redbooks

IBM Db2 13 for z/OS and More

May 2022

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

First Edition (May 2022)

This edition applies to IBM Db2 13 for z/OS.

© Copyright International Business Machines Corporation 2022. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
Preface	xiii
Authors	xiii
Now you can become a published author, too!	xix
Comments welcome	xix
Stay connected to IBM Redbooks	xx
Chapter 1. IBM Db2 13 for z/OS and the ecosystem at a glance	1
1.1 Synergy with IBM Z	2
1.2 AI infusion	3
1.3 Resiliency, availability, and scalability	3
1.4 Performance	4
1.5 Security	5
1.6 Simplification and serviceability	6
1.7 Utilities	7
1.8 Db2 13 ecosystem performance, productivity, and DevOps enhancements	8
1.9 Summary of Db2 12 continuous delivery enhancements	11
Chapter 2. Installing and migrating	13
2.1 Highlights	14
2.1.1 Migrating to Db2 13 is consistent with continuous delivery	14
2.1.2 The catalog migration process and input changed	14
2.1.3 Application compatibility toleration is unchanged	15
2.2 Prerequisites for migrating to Db2 13	15
2.3 Function level V13R1M100	16
2.3.1 Coexistence between V12R1M510 members and V13R1M100 members	16
2.3.2 New function availability in function level V13R1M100	17
2.4 Enabling V13R1M500 new function levels with no catalog dependencies	17
2.4.1 Verifying all data-sharing members are at function level V13R1M100	17
2.4.2 Activating V13R1M500	17
2.4.3 Falling back to Db2 12 no longer possible after activating V13R1M500	18
2.4.4 APPLCOMPAT level V13R1M500	18
2.5 Enabling FL 501 new functions with catalog dependencies	18
2.5.1 Updating the catalog to V13R1M501	18
2.5.2 Activating V13R1M501	19
2.5.3 APPLCOMPAT level V13R1M501	19
2.6 Using the installation CLIST to migrate to Db2 13 FL 100, FL 500, or FL 501	19
2.6.1 The traditional migration approach	19
2.6.2 Using z/OSMF to migrate a Db2 subsystem	22
2.7 Using the installation CLIST to install Db2 13	24
2.8 Subsystem parameter changes	25
2.8.1 New subsystem parameters	25
2.8.2 Changes to existing subsystem parameters	26
2.8.3 Removed subsystem parameters	27
2.9 Verifying catalog- and function-level changes	28
2.9.1 Activating lower function levels in Db2 13	28
2.9.2 Activating higher function levels in Db2 13	29

2.9.3 Ensuring that the appropriate application compatibility level is set on packages . . .	30
2.10 Summary	30
Chapter 3. Availability and scalability	31
3.1 Online deletion of active logs from the bootstrap data set	32
3.1.1 Deleting active logs	32
3.1.2 Deleting active logs without a Db2 outage	32
3.2 Partition-by-growth to partition-by-range online conversion	35
3.2.1 Considerations for PBG to PBR conversion	35
3.2.2 Requirements for converting tables from PBG to PBR	35
3.2.3 Converting a table from a PBG to PBR partitioning scheme	36
3.3 Increased concurrency for ALTER TABLE DATA CAPTURE	37
3.3.1 ALTER TABLE DATA CAPTURE concurrency enhancement	38
3.3.2 Visibility of data capture alteration to other concurrent activities before commit . .	38
3.3.3 Considerations for data replication and other tools that read Db2 log records . . .	39
3.4 Extended common service area improvements	40
3.4.1 Calculating the ECSA storage for the Distributed Data Facility component	40
3.5 Reducing agent local pool below-the-bar storage consumption	41
3.6 Storage manager contraction	42
3.6.1 Smarter contraction of DBM1 below-the-bar storage	42
3.6.2 BTB contraction	42
3.6.3 ECSA contraction	43
3.6.4 Memory object contraction	43
3.7 Database access thread availability improvements	43
3.7.1 Gradual DBAT contraction	44
3.7.2 Overview of Db2 DDF DBAT usage	44
3.7.3 Db2 DBAT termination behavior	46
3.7.4 Db2 global DDF activity statistics reporting changes	48
3.8 Real-time statistics	49
3.9 Open data set scalability	52
3.10 Increased size of directory table spaces SPT01 and SYSLGRNX	52
3.10.1 Increasing SPT01 and SYSLGRNX table space size	53
3.11 Specifying the IRLM maximum storage for locks	53
3.11.1 Sample IRLM startup procedure	53
3.12 Dynamic alteration of CF lock structure storage	54
3.12.1 New IRLM messages	55
3.12.2 Function level requirements	55
3.13 Summary	55
Chapter 4. Performance	57
4.1 Fast index traversal enhancements	58
4.2 Index look-aside optimization	58
4.3 INSERT enhancements for partition-by-growth table spaces	59
4.3.1 Partition retry process	60
4.3.2 Improved cross-partition search in a descending partition number sequence . . .	61
4.4 SORTL enhancements	61
4.5 Reducing the CPU impact in a data-sharing environment when using PBR table spaces with relative page numbers	62
4.5.1 Considerations for function level and application compatibility level	62
4.6 More performance enhancements	63
4.6.1 REORG INDEX performance improvements	63
4.6.2 Group buffer pool residency time enhancements	63
4.7 Summary	64

Chapter 5. Application management and SQL changes	65
5.1 Application timeout and deadlock control	66
5.1.1 CURRENT LOCK TIMEOUT	66
5.1.2 DEADLOCK_RESOLUTION_PRIORITY	68
5.2 Profile table enhancements	69
5.2.1 Profile table support for local applications	70
5.3 Enhancing the database object change process	73
5.3.1 Profile table support for the RELEASE_PACKAGE keyword	73
5.3.2 Optimizing DDL processes by using the RELEASE_PACKAGE keyword	75
5.3.3 Selectively disabling high-performance DBATs by using the RELEASE_PACKAGE keyword	77
5.4 Migrating distributed applications	77
5.4.1 Determining the Db2 function level and application compatibility level	78
5.4.2 Controlling the application compatibility level	78
5.4.3 Creating driver packages to use new functions	78
5.4.4 Enabling driver packages to use new functions	79
5.5 Summary	80
Chapter 6. SQL Data Insights	81
6.1 Overview of SQL Data Insights	82
6.1.1 SQL Data Insights use cases	82
6.1.2 Synergy with IBM Z	83
6.2 Installing and configuring SQL Data Insights	83
6.2.1 Determining the system environment for SQL Data Insights	84
6.2.2 Configuring Db2 for SQL Data Insights	84
6.2.3 Configuring SQL Data Insights	85
6.3 Enabling an AI query	88
6.3.1 Adding AI objects	89
6.3.2 Selecting columns for inclusion	90
6.3.3 Specifying filter values	92
6.3.4 Processing the enablement	92
6.4 Running AI queries	93
6.4.1 Using the AI_SIMILARITY function	93
6.4.2 Using the AI_SEMANTIC_CLUSTER function	98
6.4.3 Using the AI_ANALOGY function	100
6.4.4 Interpreting AI query results	102
6.5 Summary	103
Chapter 7. Security	105
7.1 Reduced Security Manager contention for Db2 access	106
7.1.1 Plan authorization cache	106
7.1.2 Global authentication cache	107
7.2 Improved flexibility in managing plans, packages, and SQL routines	107
7.2.1 Owner type support for packages of SQL procedures and functions	107
7.2.2 Owner type support for packages and plans	108
7.3 Db2 support for continuous compliance	109
7.4 Read-only archived key Db2 support	110
7.5 Summary	110
Chapter 8. IBM Db2 for z/OS utilities	111
8.1 Utility history (function level 501)	112
8.1.1 Number of SYSUTILITIES rows that are inserted by a utility	113
8.1.2 EVENTID columns	114
8.1.3 SYSUTILITIES columns	114

8.1.4	Abends, restarts, and commands	115
8.1.5	Db2 catalog objects for utility history.	116
8.1.6	Cases where utility history is not collected	116
8.1.7	Queries on utility history	117
8.1.8	Cleaning up the utility history information	125
8.2	Enhanced inline statistics with page-level sampling (function level 500)	127
8.3	Enhanced space-level recovery (function level 100).	128
8.4	REORG INDEX NOSYSUT1 as the default option (function level 500)	129
8.5	REPAIR WRITELOG dictionary support (function level 100)	130
8.6	Summary.	130
Chapter 9. Instrumentation and serviceability		131
9.1	Index management diagnostics and serviceability enhancements	132
9.1.1	New IFCID 396 trace facility for recording abnormal index splitting	132
9.1.2	New fields for recording and aggregating general index split information.	133
9.2	Improved EDITPROC support for IFCID 306	133
9.3	Improved partition range support for IFCID 306	134
9.4	IFCID 369 aggregated accounting statistics is now part of STATISTICS CLASS(1)	135
9.5	IFCID 003 longest lock or latch waiter	135
9.6	Enhanced distributed thread monitoring	135
9.6.1	New IFCID 411 trace facility for recording DDF application statistics.	135
9.6.2	New IFCID 412 trace facility for recording DDF user statistics.	137
9.6.3	New fields for the IFCID 365 trace facility for recording DDF location statistics.	139
9.6.4	New fields added to the IFCID 1 trace facility for recording global DDF.	139
9.7	Group buffer pool residency time enhancements	140
9.8	Page reject of insert row diagnostic log record	141
9.9	The STATIME_MAIN subsystem parameter default is now 10 seconds.	141
9.10	Summary.	142
Chapter 10. Machine learning and artificial intelligence infusion		143
10.1	IBM Watson Machine Learning for z/OS.	144
10.1.1	Key features and capabilities	145
10.1.2	Latest WMLz updates.	148
10.1.3	Opportunities for using WMLz.	148
10.2	Db2 AI for z/OS.	150
10.2.1	SQL optimization.	150
10.2.2	System assessment	151
10.2.3	Distributed connection control.	151
10.2.4	What is new in Db2ZAI V1.5	152
10.3	Summary.	153
Chapter 11. Development and administration tools		155
11.1	IBM SQL Tuning Services.	156
11.1.1	SQL Capture API	156
11.1.2	Visual Explain API.	156
11.1.3	Statistics Advisor API	157
11.1.4	Query Environment Collector API.	157
11.1.5	Administrative APIs.	157
11.2	IBM Db2 for z/OS Developer Extension	157
11.2.1	Integrating with Db2 13.	157
11.2.2	Making it easier to deliver high-quality SQL	158
11.2.3	Improving query performance	160
11.2.4	Putting Db2 Developer Extension to use	161
11.3	IBM Db2 Administration Foundation for z/OS	162

11.3.1	Running SQL and Db2 commands	162
11.3.2	Working with catalog objects	163
11.3.3	Exploring SQL	164
11.3.4	Analyzing and tuning SQL statements	164
11.3.5	Use cases	166
11.4	Summary	174
Chapter 12. IBM Db2 for z/OS tools support for Db2 13		175
12.1	IBM Db2 Administration Tool for z/OS	176
12.2	IBM Db2 Automation Tool for z/OS	177
12.3	IBM Db2 Log Analysis Tool for z/OS	178
12.3.1	Application-level relationship discovery	178
12.4	IBM Db2 Recovery Expert for z/OS	178
12.5	IBM Db2 Query Monitor for z/OS	179
12.6	IBM Db2 Query Management Facility	180
12.7	IBM Db2 Sort for z/OS	180
12.8	IBM Data Virtualization Manager for z/OS	181
12.8.1	Key features	181
12.8.2	Db2 for z/OS usage	182
12.9	IBM OMEGAMON for IBM Db2 Performance Expert on z/OS	184
12.9.1	Monitoring SQL Data Insights accounting metrics	185
12.9.2	Monitoring GBP residency times (IBM z16)	189
12.9.3	Monitoring SET CURRENT LOCK TIMEOUT related Db2 changes	192
12.10	Summary	199
Chapter 13. IBM Db2 Analytics Accelerator for z/OS		201
13.1	Overview of Db2 Analytics Accelerator	202
13.1.1	Non-accelerator Db2 tables	203
13.1.2	Accelerator-shadow tables	203
13.1.3	Accelerator-archived tables and partitions	204
13.1.4	Accelerator-only tables	204
13.2	Query acceleration enhancements	205
13.2.1	Pass-through only expression support	206
13.2.2	SQL pagination support	206
13.2.3	Column mask support	207
13.2.4	Dynamic plan stability	208
13.3	Accelerator-only table enhancements	208
13.3.1	High availability support for AOTs	209
13.3.2	Distribution key support for AOTs	211
13.3.3	Explicit statistics collection on AOTs	212
13.3.4	Migration considerations from Db2 Analytics Accelerator 5 to 7	213
13.4	Integrated Synchronization enhancements	213
13.4.1	Online schema change (Add/Alter column) support	214
13.4.2	ALTER ROTATE support	215
13.4.3	DROP TABLE support	216
13.4.4	True HTAP support (the WAITFORDATA feature)	216
13.4.5	Db2 for z/OS utility support	218
13.5	Db2 Analytics Accelerator administration enhancements	219
13.5.1	Db2 Analytics Accelerator administration services	219
13.5.2	IBM Db2 Administration Foundation for z/OS	220
13.6	Summary	223
Chapter 14. IBM Db2 for z/OS Data Gate		225
14.1	Prerequisites and configuration overview	227

14.1.1	Software dependencies	227
14.1.2	Installation overview	227
14.1.3	Configuration overview	227
14.2	Administering Db2 Data Gate	228
14.2.1	Selecting a target database type	228
14.2.2	Creating a Db2 Data Gate instance	228
14.2.3	Pairing a Db2 Data Gate instance with a Db2 for z/OS subsystem	228
14.2.4	Managing tables	228
14.2.5	Monitoring	229
14.3	Using the most recent source data of an independent cloud replica of Db2 for z/OS by using WAITFORDATA	230
14.3.1	How WAITFORDATA works	231
14.3.2	WAITFORDATA behavior	232
14.4	Summary	232
Appendix A. IBM Db2 12 continuous delivery features and enhancements		233
Application development		234
SQL enhancements		234
Db2 REST services		237
Other application development capabilities		238
Database administration		239
Package management		239
Data Definition Language		240
Utilities		243
System administration and system management		243
Performance		243
Db2 system-related enhancements		244
Security		247
Synergy with z15 and IBM DS8000 product families		248
Activating V12R1M510: Considerations		249
Application impacts and continuous delivery		250
Progressing to function level V12R1M510		250
Planning to activate Db2 12 function level 510 (V12R1M510) in preparation for migration to Db2 13		251
Summary		251
Appendix B. Application development		253
SQL enhancements		254
Enhancement to temporal and archive transparency for WHEN clauses on triggers ..		254
CREATE OR REPLACE syntax for stored procedures		255
Application-level granularity for lock limits (NUMLKUS and NUMLKTS)		257
Db2 REST services		259
Initial Db2 native REST support		259
Creating, discovering, invoking, and deleting REST services		260
Native REST services and RESTful APIs with Db2 and z/OS Connect		261
SERVICE versioning		261
COPY options for promoting Db2 REST services		261
Remote issuance of the FREE SERVICE subcommand		262
Summary		262
Appendix C. Database administration		263
REBIND and package management		264
REBIND phase-in		264
Data Definition Language		267

Huffman compression	267
Preventing new deprecated objects	270
Migrating multi-table table spaces to PBG UTS	274
Db2 utilities	277
REORG	277
LOAD	280
RUNSTATS	283
CHECK DATA	284
COPY	284
RECOVER	285
MODIFY RECOVERY	285
Concurrency	286
Replication support	286
Stored procedures	287
Appendix D. System administration and system management	289
Performance	290
Fast index traversal	290
SORTL	293
Db2 system topics	294
Data sharing	294
Distributed Data Facility	297
Subsystem monitoring	298
DSNZPARM simplification	299
Security	300
Summary	303
Abbreviations and acronyms	305

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

CICS®	IBM Watson®	Redbooks®
Db2®	IBM Z®	Redbooks (logo)  ®
DS8000®	IBM z14®	SPSS®
FlashCopy®	Informix®	UrbanCode®
GDPS®	InfoSphere®	WebSphere®
IBM®	OMEGAMON®	z/OS®
IBM Cloud®	Parallel Sysplex®	z15™
IBM Cloud Pak®	Plex®	
IBM Research®	RACF®	

The following terms are trademarks of other companies:

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Zowe, are trademarks of the Linux Foundation.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

OpenShift, Red Hat, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM® Db2® 13 for z/OS® delivers a host of major new advancements for using, maintaining, and administering this enterprise-critical database platform. In parallel with the development of the IBM z16 hardware and the z/OS 2.5 operating system, Db2 13 leverages the latest transformative innovations on the IBM Z® platform to improve database performance, availability, scalability, reliability, and security. The integration of database artificial intelligence (AI) technology in Db2 13 increases too, which helps you simplify management, improve performance, and easily extract analytical insights from your data by using the SQL API.

This IBM Redbooks® publication describes the many innovations in Db2 13 and the companion products that comprise the Db2 for z/OS ecosystem. The contents of this book will help application developers, database administrators, and solution architects understand the newest capabilities of Db2 for z/OS and related tools and learn how to leverage them to get the greatest value from your database.

In addition, many Db2 12 enhancements are provided in the continuous delivery stream. This publication contains appendixes that describe the most important of the many new features that were delivered after the initial Db2 12 release.

Authors

This book was produced by The IBM Db2 V13 Project Team.

Allan Lebovitz is a Software Engineer who has been working at IBM for over 34 years. He has been a Db2 for z/OS developer for 22 years who is based at the IBM Silicon Valley Laboratory. Allan's focus in Db2 is primarily related to Sort and Sparse Index, but he also has experience in other areas of Db2, such as RDS Runtime and Data Manager Work File Manager.

Bart Steegmans is a Consulting Support Specialist from IBM Belgium, currently working for the IBM Silicon Valley Laboratory in San Jose to provide technical support for Db2 for z/OS performance problems. He has over 34 years of experience in Db2. Before joining IBM in 1997, Bart worked as a Db2 system administrator at a banking and insurance group. His areas of expertise include Db2 performance, database administration, and backup and recovery. Bart is a co-author of numerous IBM Redbooks publications about Db2 for z/OS.

Bill Bireley is a long-time Db2 for z/OS developer and quality assurance engineer that is based at the IBM Silicon Valley Laboratory. He has led development and test teams delivering features in both the Db2 relational engine and in related ecosystem components. Bill currently leads the System Test squad, and he is focused on IBM Db2 Analytics Accelerator stress testing.

Binghui Zhong is an IBM Software Engineer who is based at the IBM Silicon Valley Laboratory. He has 20 years of experience working with Db2 Relational Data System (RDS), and has been involved in multiple projects in the area of availability and scalability.

Craig Friske is a Software Engineering Manager with Rocket Software who is responsible for Db2 utilities development. He has 30 years of experience working with Db2 and leading projects like Online Reorg and PBR/RPN table spaces. He co-authored *Db2 for z/OS Utilities in Practice*, REDP-5503.

Debbie Yu is an IBM Software Engineer who is based at the IBM Canada Software Lab, Toronto. Her focus is on Db2 supplied stored procedures, Db2 administrative task scheduler, and Db2 installation and migration.

Dengfeng Gao is a Db2 for z/OS developer. She has 27 years of experience with general database management systems and 14 years of experience in Db2 development. Her focus is on the Db2 bind process with extended knowledge in RDS.

Dennis Lukas Mayer is a Product Manager for IBM Db2 Analytics Accelerator for z/OS and IBM Db2 for z/OS Data Gate who is base at the Boeblingen Lab, Germany. He joined IBM in 2011 and has worked with IBM Z and Db2 for z/OS customers since 2014 in various roles by supporting sales and deployment teams around the world. He has presented products roadmaps and strategy at various technical conferences and user group meetings.

Derek Tempongko is a Db2 for z/OS developer in the Distributed Data Facility (DDF) team. He has been working at IBM, specifically in the DDF team, for over 20 years. He also co-authored *IBM DB2 for z/OS: Configuring TLS/SSL for Secure Client/Server Communications*, REDP-4799.

Doug Dailey is an IBM Product Manager with over 10 years of experience in building software and solutions for customers. Doug worked in technical support for Informix Software (now an IBM company and known as IBM Informix®) and then transitioned to a Technical Account Manager role, and then became a CSM Manager for the IBM Accelerated Value Program. He specializes in data federation and data virtualization technologies.

Eric Radzinski is a technical writer and editor who is based at the IBM Silicon Valley Lab. He has spent his career documenting IBM Z database technology, including Db2 for z/OS, Db2 Tools, and IMS. He is a co-author of *The IBM Style Guide and Developing Quality Technical Information: A Handbook for Writers and Editors*. He is working on projects that are related to transforming IBM Z applications for today's modern application development environments.

Felipe Bortoletto is a Certified IBM IT Specialist in Information Management and an IBM Certified DBA and system programmer for Db2. He has 25 years of experience in IT and 21 years of experience with Db2 for z/OS. He joined IBM 16 years ago and is a member of the L2 Support Team for Db2 for z/OS with responsibility in the area of IBM Db2 Analytics Accelerator. He holds a degree in Computer Science from UNICAMP. Felipe co-authored *Securing and Auditing Data on Db2 for z/OS*, SG24-7720, *DB2 10 for z/OS Performance Topics*, SG24-7942, *DB2 11 for z/OS Technical Overview*, SG24-8180, and *Subsystem and Transaction Monitoring and Tuning with DB2 11 for z/OS*, SG24-8182.

Frances Villafuerte is a Software Engineer in the Db2 core engine area. She has over 22 years of experience working with Db2, where she develops and leads projects in the key functional areas of availability, scalability, and performance. She has wide technical knowledge and is a technical lead in the Data Manager component of Db2.

Frank Chan is a Software Engineer in the Db2 core engine area and technical lead of the Instrumentation facility. He has 15 years of experience working with Db2 and 20 years in the database field. He holds a degree in Computer Science from San Jose State University. His areas of expertise include parsing and compiling relational database programming languages, and profile monitoring and system performance report analysis.

Gayathiri Chandran is a Software Engineer in Db2 for z/OS development at the IBM Silicon Valley Lab. She is the technical lead for Db2 security and drives the design and development of security functions. She speaks at various events and conferences and blogs on various topics that are related to Db2 security.

Guanjun Cai is an information architect and developer at the IBM Silicon Valley Lab. He holds a Ph.D. in English from the University of Arizona. He has 24 years of experience in designing and developing technical documentation for IBM Data and AI, enterprise storage, and z/OS products. His areas of expertise include information architecture, development, and publishing, and user interface content design, markup languages, and automation. His recent publications include IBM Redbooks and Redpaper chapters and IDUG articles on infusing machine learning and AI into enterprise cognitive solutions.

Guenter Georg Schoellmann is an IT Specialist for IBM Db2 Analytics Accelerator for z/OS and IBM Db2 for z/OS Data Gate who is based at the Germany development lab as a member of the Center of Excellence team. As part of the development organization, his responsibilities include customer consultation, proof of concepts, deployment support, migrations, resolution of critical situations, and education for IBM Db2 Analytics Accelerator for z/OS. During his more than 37 years with IBM, Guenter has worked as a developer, team leader, customer advocate, and PMI certified project manager.

Hanif Kassam is a Software Engineer working as a Client Success Representative in Db2 for z/OS Development Lab. He holds a Bachelor of Science degree in Computer Science with a minor in Management Information Systems. He has been working with Db2 for z/OS since version 1.3 and has over 30 years of experience. His subject matter expertise is in Installation and Migration, for which he has developed internal courses. He currently handles USAGE cases in all components of Db2 for z/OS.

Harsha Dodda is a Software Engineer who is based at the IBM Silicon Valley Lab. He is a full-stack developer working on user interface and API development for the Db2 for z/OS administrator and developer experience transformation. He holds a degree in Computer Science from the University of Illinois at Chicago, and he has over 4 years of experience building software.

Hoang Luong is a Software Engineer in the Db2 for z/OS core engine area. He holds a Master's Software Engineering degree from San Jose State University. He has 7 years of experience working with database commands, Database Exception Table (DBET), and real-time statistics.

Jason Cu is a Software Engineer working on Db2 for z/OS at the IBM Silicon Valley Lab.

John R Lyle is a Software Engineer in Db2 for z/OS with more than 30 years of development experience. He worked in the RDS area with a focus on Db2 system catalog, migration, and fallback, and the Data Definition Language (DDL) and index manager areas. He has been the lead designer and implementer of the catalog and fallback changes since Db2 5. He has worked on numerous other development items over the years, including the Clone Table function and all the Db2 8 - 11 Utility related enabling new function mode (ENFM) catalog and directory changes. John was the design and development lead for the Db2 12 continuous development process and Online Migration capabilities. For Db2 13, John designed and helped implement the revolutionary changes to the release migration process. John is working in the Db2 Provisioning-Deployment-Migration team as the team leader. John speaks regularly at IBM conferences worldwide (such as IDUG and SHARE), user conferences, and regional user groups (RUGs) on migration and other topics.

Katherine Soohoo is the technical lead for IBM Db2 for z/OS Developer Extension on VS Code. She has been working on the development team for the Db2 Developer Extension since its first release in 2020. In her previous role, Katherine was a key member of the team integrating automated Db2 for z/OS application deployments by using Git, Jenkins, IBM UrbanCode® Deploy, and IBM Db2 Tools.

Keziah Knopp is a Db2 for z/OS specialist who is based at the IBM Z Washington Systems Center. In addition to working with Db2 for z/OS customers, she is the host of the IBM Z Customer Councils on the west coast, delivers the Db2 for z/OS: REST and Hybrid Cloud Workshop, and has presented sessions at IDUG and SHARE.

Laura Kunioka-Weis is a Software Engineer with Rocket Software. She is the technical lead for the Db2 backup and recovery utilities, and led the redirected recovery and utility history projects.

Long Tu is a Software Engineer who is based at the IBM Silicon Valley Laboratory. He holds a Master's degree in Software Engineering and has over 10 years of experience working with Db2 RDS. He is involved in and leads projects about Db2 availability.

Maria Sueli Almeida is an IBM Certified Thought Leader and member of Db2 for z/OS development who is based at the IBM Silicon Valley Lab in San Jose, California. She has extensive experience with Db2, including database administration and systems programming. She has worked on Db2 for z/OS advanced technology and solution integration. Currently, she is engaged with Db2 for z/OS support for user experience transformation, including DevOps adoption and cloud provisioning database-as-a-service (DBaaS) for DBAs, application developers, and sysprogs. Sueli also consults with Db2 customers worldwide.

Mark Rader is a Db2 for z/OS specialist who is based at the IBM Z Washington Systems Center. He has worked with many organizations that use Db2 for z/OS, starting with version 2.2. He has led many Db2 data sharing implementations and presented on many topics at conferences, including at IDUG, SHARE, IBM Technical University, and at Db2 RUG meetings. He has worked closely with Db2 development to deliver the Early Support Programs for Db2 10, Db2 11, and Db2 12. He is an author of IBM Redbooks publications about IBM Parallel Sysplex® application considerations, Db2 for z/OS data sharing, and Db2 for z/OS distributed function.

Mehmet Cüneyt Göksu is an Executive IT Specialist for IBM Z solutions in the context of Data and AI who is based at the IBM Germany development lab as a member of the Center of Excellence team. He holds a Bachelor degree and PhD in Computer Science, and an MBA. He has worked with IBM Db2 for z/OS and IBM Z for over 30 years. Cuneyt focuses on IBM Db2 Analytics Accelerator and Data Gate to help a diverse set of customers with proof-of-concepts, deployments, and migrations. Before joining IBM, Cuneyt was a member of the IBM Db2 Gold Consultant Team; was named an IBM Champion; became a member of the BOD in the IDUG community; and was a regular instructor for IBM Education Programs. He is a L3 Certified IT Specialist, and he actively attends certification board reviews and Db2 for z/OS Liaison@IDUG EMEA.

Mike Shadduck has been a Software Engineer working at IBM for over 30 years. He has been a Db2 for z/OS developer for 23 years, and he is based at the IBM Silicon Valley Laboratory. Mike's focus in Db2 has been in the Index Manager component, where he helps to deliver and support variable length index keys, index key versioning, index key compression, and index Fast Traversal Blocks (FTBs).

Paul Bartak has been working with IBM databases since 1985, including the first version of Db2 on MVS. As an IBM customer for 14 years, Paul worked in Application Development, Database Administration, and DBA. Currently as a Distinguished Engineer with Rocket Software, Paul focuses on the tools that are used to maximize z/OS platform value with emphasis in database administration, cloning, and DevOps processes. He supports multiple Db2 User Groups; has been a speaker at several IDUG, IBM Z Day, and Insight/Think conferences; is an IBM Redbooks and IBM Redpaper publications author; and IBM Champion.

Pui-Yi Mak is a Software Engineer at IBM. She has worked on numerous Db2 for z/OS projects, some of which are collaborations with other products within IBM. Her current focus is on Authorization and SQL Data Insights.

Randy Nakagawa is a Software Engineer who is based at the IBM Silicon Valley Laboratory. He has over 35 years of experience at IBM working on Db2 for z/OS and related technologies. He has extensive experience in several areas within the Db2 for z/OS product, and is focused on SQL statement preparation and dynamic statement caching.

Regina Liu is a Software Engineer who graduated from the University of California at Berkeley with a Bachelor's degree in Computer Science. She has over 24 years of experience in Db2 for z/OS development. As the technical lead for the DDL area, she leads and drives many availability and DBA-related enhancements across the RDS, core engine, and Utility areas in Db2. Her innovative approach for implementing DDL alterations as pending definition changes spear-headed online schema evolution, and she is continually driving improvements to DDL operations and concurrency.

Robert Lu is a Software Engineer currently working in the Agent Services area of Db2 for z/OS. He previously worked on the Buffer Manager component for Db2 for z/OS.

Sarbinder Kallar is a Software Engineer working in the DDF area of Db2 for z/OS. He is based at the IBM Silicon Valley Laboratory.

Sharon Roeder is a Software Engineer who is based at the IBM Silicon Valley Laboratory in the Db2 core engine area with a focus on log manager. She also has experience in the service controller, command processor, executive, optimizer, and Instrumentation Facility Component (IFC) components.

Shirley Zhou is a Software Engineer who is based at the IBM Silicon Valley Laboratory. She has been a developer for over 20 years in the Db2 for z/OS core engine area.

Sowmya Kameswaran is the technical lead for the administration and development experience transformation for the Db2 for z/OS engine, tools, and IBM Db2 Analytics Accelerator who is based at the IBM Silicon Valley Lab. She holds a Master's degree in Software Engineering from San Jose State University. She speaks at various events and conferences, and writes blogs on various topics that are related to the system and application architecture transformation.

Steven Ou is a Software Engineer who is based at the IBM Silicon Valley Laboratory with 6 years experience working on Db2 for z/OS. His current focus is on the IFC. Steven attended San Jose State University where he earned a Bachelor of Science degree in Computer Science.

Sydney Villafuerte is a Software Engineer who joined the IBM Db2 core engine area after graduating from the University of California at San Diego with a Bachelor of Science degree in Computer Science with a focus in Bioinformatics. Since then, she has worked in the XML, IFC, and Storage Management areas. Over the last year, she focused on Storage Manager contraction and contributed to the development effort that is described in this publication.

Tammie Dang is a Software Engineer working on Db2 for z/OS who is based at the IBM Silicon Valley Lab. She has over 30 years of experience in relational databases, and she is co-author of *IBM DB2 12 for z/OS Technical Overview*, SG24-8383. One of her current roles is leading the customer requirement assessment effort in the new application area.

Tim Hogan is a technical writer who is based at the IBM Silicon Valley Lab. He leads the Db2 AI for z/OS content development team.

Tom Toomire is a Software Engineer who is based at the IBM Silicon Valley Laboratory. He has over 33 years of experience at IBM working on Db2 for z/OS and related technologies. He has extensive experience in several areas within the Db2 for z/OS product, and is working in the DDF area as the lead developer for the Db2 DDF native RESTful services function.

Tracee Tao is a Software Engineer in Db2 for z/OS development who is based at the IBM Silicon Valley Lab. Her focus is primarily in the Authorization area of Db2.

Tran Lam is a Software Engineer who is based at the IBM Silicon Valley Laboratory. She has been working on Db2 for z/OS for 7 years after graduating from San Jose State University with Bachelor of Science degree in Software Engineering with a minor in Mathematics. Her areas of expertise include installation, ZPARM, samples, and Db2.

Ute Baumbach is a software developer who is based at the IBM Research® & Development Lab at Boeblingen, Germany. During her more than 30 years with IBM, Ute worked as a developer and team leader for various IBM software products, mostly related to Db2 for Linux, UNIX, and Windows (LUW) and Db2 z/OS topics and tools. Currently, she works as a member of the Analytics on IBM Z Center of Excellence team, which is part of the Db2 Analytics Accelerator development organization, and she focuses on Db2 Analytics Accelerator proofs of concepts, customer deployment support, and education. She has co-authored various IBM Redbooks publications, focusing on Db2 for LUW topics and Db2 Analytics Accelerator.

Vassil Dimov is a Technical Lead in the IBM Db2 for z/OS Data Gate development team who is based at the IBM Research & Development Lab at Boeblingen, Germany. He initially worked as a Software Engineer for IBM Db2 Analytics Accelerator 5 years ago with an emphasis on data replication technologies. Later, he became the development focal point for the L3 support team for all customer-related data replication issues. Two years ago, he moved to the Db2 for z/OS Data Gate team to provide a modern alternative to IBM Db2 Analytics Accelerator in the cloud.

Vicky Li is a Software Engineer who is based at the IBM Silicon Valley Laboratory. She has over 19 years of experience working with Db2 RDS, and she is involved in and leads DDL and database descriptor management (DBDM) projects for Db2 availability.

Xiao Wei Zhang is a Software Engineer who is based at the IBM China Laboratory. He has over 15 years of experience with database development and performance at IBM.

Yong Kwon is a Software Engineer working at IBM for over 17 years. He has been a Db2 for z/OS developer for all 17 years, and he is based at the IBM Silicon Valley Laboratory. Yong's focus in Db2 has been primarily related to Data Manager and Authorization.

Zijin Guo is a Software Engineer working on Db2 for z/OS who is based at the IBM Silicon Valley Lab. After joining IBM, he worked on RDS for 3 years and then focused on the Buffer Manager component in the Db2 core engine area.

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

Ada La, Akiko Hoshikawa, Ben Budiman, Bituin Vizconde, Bob Tokumaru, Chris Leung, David Nguyen, Fen-Ling Lin, Haakon Roberts, Jim Pickel, Julie Chen, Ka-Chun Ng, Kate Wheat, Ke Wei Wei, Kim Lyle, Koshy John, Laura Klappenbach, Leilei Li, Limin Yang, Maggie Lin, Manvendra Mishra, Mary Lin, MaryBeth White, Matthias Tschaffler, Meg Bernal, Mo Townsend, Nicholas Marion, Nina Bronnikova, Patrick Malone, Paul McWilliams, Ping Wang, Ralf Marks, Randy Nakagawa, Sueli Almeida, Teresa Leamon, Thomas Eng, Tom Majithia, Xiao Feng Meng, Xiaohong Fu, Xu Qin Zhao, Wei Li

Special thanks to the following people for their unique contributions to this publication and the overall project:

- ▶ Mo Townsend and Teresa Leamon for proposing and funding this project.
- ▶ Leilei Li and Bill Bireley for leading teams across organizations and geographies to ensure the timely delivery of this publication.
- ▶ Akiko Hoshikawa and Haakon Roberts for providing technical guidance and feedback to the project team.
- ▶ Eric Radzinski and Guanjun Cai for providing the technical edit of this publication.

This publication was produced by the IBM Redbooks publication team with valuable contributions from Martin Keen, IBM Master Inventor and IBM Redbooks Project Lead, and Wade Wallace, Senior Content Specialist and IBM Redbooks Team Lead.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:
<https://www.redbooks.ibm.com/residencies>

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

<https://www.redbooks.ibm.com/contact>

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, IBM Redbooks
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>



IBM Db2 13 for z/OS and the ecosystem at a glance

IBM Db2 13 for z/OS delivers significant advances in all the critical enterprise database success factors, including availability, scalability, performance, security, and ease of use. Additionally, the value of this new version is maximized by its synergy with tightly related tools and technology. This ecosystem, which includes the latest advances in IBM Z hardware, new and improved development tools, and add-on capabilities such as artificial intelligence (AI) infusion and IBM Db2 Analytics Accelerator query acceleration, complements all the traditional Db2 capabilities to provide the most complete, intelligent, and intuitive database environment yet. The release of Db2 13 for z/OS was intentionally coordinated with the release of IBM z16, the next generation of IBM Z hardware. This parallel approach to development underscores the deep synergy that exists between Db2 and the IBM Z platform.

Db2 13 was released approximately 5 years after the introduction of Db2 12 for z/OS. Db2 12 introduced a continuous delivery model that enables the controlled addition of functional enhancements through regularly delivered function level upgrades. Over the past 5 years, this model delivered over 100 small and medium enhancements across 10 individual function levels, including several that require simple Db2 catalog changes. More importantly, with the continuous delivery model, you can access new features sooner and selectively activate features as needed.

For more information about these Db2 12 function level enhancements, see the following appendixes:

- ▶ Appendix A, “IBM Db2 12 continuous delivery features and enhancements” on page 233
- ▶ Appendix B, “Application development” on page 253
- ▶ Appendix C, “Database administration” on page 263
- ▶ Appendix D, “System administration and system management” on page 289

However, the continuous delivery model was never intended to eliminate the need to create an entirely new version of Db2. The decision to deliver a new version was based on the scope of new enhancements, infrastructure changes, advances in related technologies, and business value. The convergence of all of these factors is here in the form of Db2 13 for z/OS.

This first chapter gives a brief overview of the most important functions that are provided by IBM Db2 13 for z/OS (also referred to throughout this book as Db2 13 for z/OS or Db2 13). Additionally, this book provides an overview of several important ecosystem components that deliver the insights, productivity, and performance boosts that further expand and improve the overall capabilities that are provided by Db2 13.

The remainder of this chapter introduces the primary themes, highlights, and ecosystem components that make up the Db2 13 experience. Subsequent chapters explain the major changes by functional area and describe the value of each one. For more information about Db2 13 and its related products, see the relevant product documentation.

This chapter describes the following topics:

- ▶ Synergy with IBM Z
- ▶ AI infusion
- ▶ Resiliency, availability, and scalability
- ▶ Performance
- ▶ Security
- ▶ Simplification and serviceability
- ▶ Utilities
- ▶ Db2 13 ecosystem performance, productivity, and DevOps enhancements

1.1 Synergy with IBM Z

The release of Db2 13 intentionally coincides with the release of the latest IBM Z hardware iteration: IBM z16. Db2 for z/OS has always been among the first exploiters of new IBM Z hardware and software features, and in many cases, is the inspiration for those features. The following recent Db2 enhancements are made possible by advances in IBM Z:

- ▶ Leveraging the new IBM Z processors for AI workloads. SQL Data Insights enhances AI capability by using AI and machine learning (ML) libraries (z/OS versions 2.4 and 2.5): IBM Z Deep Neural Network (zDNN) library, IBM Z AI Optimization (zAIO) library, and IBM Z AI Embedding library.
- ▶ More efficient sorts through expanded use of the IBM Integrated Accelerator for Z Sort, and its **SORT LISTS (SORTL)** machine instruction, which is available in IBM z15™ or later.
- ▶ Improved scalability by leveraging the z/OS 2.5 ability to manage more concurrent data sets and threads.
- ▶ Better monitoring of Group Buffer Pools (GBPs) in a data-sharing environment by using the new GBP residency time enhancements in the coupling facility (CF).

1.2 AI infusion

A major initiative in Db2 for z/OS is the enablement and exploitation of AI and ML. That effort is accelerated in Db2 13 with two primary AI initiatives:

- ▶ *SQL Data Insights* is a new optional feature that provides the power to easily draw insights and patterns from your existing enterprise data with minimal data science expertise. This exciting new capability provides a front-end visualization tool along with built-in analytics features in Db2 13 and z/OS. For more information, see Chapter 5, “Application management and SQL changes” on page 65 and Chapter 6, “SQL Data Insights” on page 81.
- ▶ *AI infusion in Db2 for z/OS* improves performance, reduces cost, and provides proactive system monitoring. The Intelligent Processing capability is most evident in the following two primary components of the Db2 ecosystem:
 - IBM Db2 AI for z/OS: This ecosystem product improves operational efficiency by using ML to help you optimize your SQL, better manage and monitor the health of your Db2 system, and automatically govern distributed connections.
 - The Db2 engine code: The engine code itself intelligently adapts its behavior, learning from execution history. Examples include the Index Lookaside enhancements (see Chapter 3, “Availability and scalability” on page 31) and smarter Distributed Data Facility (DDF) thread deallocation (see Chapter 5, “Application management and SQL changes” on page 65).

1.3 Resiliency, availability, and scalability

These three strengths historically are the foundation of Db2 for z/OS. Every new version increased the capacity, concurrency, and reliability of the core database capabilities. Db2 13 is no exception. The latest Db2 13 features reduce application outages and accommodate application and workload growth while simplifying the process for DBAs to make database object changes.

Db2 13 handles more concurrent user activity with efficient use of compute and memory resource. Here are the most impactful enhancements:

- ▶ Online conversion from a partition-by-growth (PBG) to a partition-by-range (PBR) partitioning scheme.
By using this enhancement, you can leverage PBR with minimal impact to an application.
- ▶ The ability to delete an active log data set from the bootstrap data set (BSDS) while Db2 is running.
As a DBA, you can encrypt active log data sets, increase their size, or change the key label periodically without bringing down Db2.
- ▶ Extended common service area (ECSA) constraint relief.
ECSA storage constraints previously restricted workload growth and caused outages due to insufficient storage. Db2 13 better handles peak workloads and supports new types of workloads by improving storage conditions and reducing unexpected system contention while running with many concurrent threads.

ECSA constraint relief provides the following significant improvements:

- The usage of ECSA below-the-bar (BTB) storage for instrumentation (IFC) is greatly reduced. Working storage for much of the Accounting and Statistics traces was moved from ECSA to 64-bit common.
- The DDF further reduced its ECSA storage footprint by moving the remaining ECSA use for TCP/IP connections to the common area above the bar.
- ▶ DBM1 31-bit storage relief.
Db2 13 can now store statement text above the bar so that Db2 supports more concurrent threads.
- ▶ Improvements in the Database Access Thread (DBAT) termination process.
With these improvements, Db2 supports more concurrent threads by making more below the bar storage available.
- ▶ Capacity for more concurrent open data sets.
Db2 13 leverages the z/OS 2.5 Dynamic Allocation feature for System Work Blocks (SWBs), which helps reduce BTB storage usage. With this reduction, there is more room for more concurrent data sets or for other Db2 activities, such as more concurrent threads.
- ▶ Real-time statistics (RTS) scalability.
Several tables that are used for RTS recording contain columns that are defined as INTEGER, which limits the maximum value. As data volume and workloads get larger, these columns must be expanded to handle the scale of current data volumes.
- ▶ Increased size of directory table spaces.
Db2 13 increases the maximum size of directory table spaces SPT01 and SYSLGRNX from 64 GB to 256 GB to avoid future concerns regarding space limitations, which creates capacity to address ever-growing workloads that require more applications and more data sets.

1.4 Performance

A cornerstone of every new Db2 version is the continued pursuit of improved performance for queries, transactional workloads, utilities, and connected applications. Faster turnaround, greater throughput, and reduced computing resource usage are improved in each new version and in continuous delivery enhancements. Chapter 4, “Performance” on page 57 describes the following major enhancements:

- ▶ Index Look-aside Optimization
Enjoy the query performance benefits of more indexes on tables while minimizing the index maintenance costs during **INSERT**, **UPDATE**, and **DELETE** operations. With Index Look-aside Optimization in Db2 13, root-to-leaf index traversals are further minimized for all **INSERT**, **UPDATE**, and **DELETE** operations regardless of the index cluster ratio.
- ▶ A larger maximum key size for Fast Index Traversal
This index traversal technique, also known as Fast Traversal Blocks (FTBs), now can be used by more tables that contain large index keys.
- ▶ Insert performance for PBG table spaces
Db2 13 improves the cross-partition search logic, along with the partition-locking strategy that is used on **INSERT** operations to improve the performance and the efficient use of space within existing partitions.

- ▶ Expansion of z15 **SO**R**T LIST** command (**SO**R**T**L) usage
Initially introduced as a Db2 12 functional enhancement, this feature came with several restrictions on when it can be used. Db2 13 lifts many those restrictions to enable a much wider usage of the feature.
- ▶ Smarter sort optimization
The sort optimization techniques that were recently introduced are extended to many more predicates, functions, and data types.

1.5 Security

Security and governance of data and processes in Db2 for z/OS are of utmost importance. As the application landscape and network connectivity have grown more open and complex, Db2 for z/OS kept ahead by implementing advances in encryption, rules processing, and definition of roles while allowing flexibility for using Db2 -based security or external security managers. However, increased security that comes at the cost of performance is not an acceptable tradeoff. Db2 continues to innovate with a focus on performance in its authorization checking.

In Db2 13, these trends continue with the following key enhancements:

- ▶ Continuous compliance.
As security regulations become more complex and stringent, organizations often struggle to interpret regulations, implement controls, and collect evidence of continuous compliance. Db2 13 is enhanced to provide the evidence that is needed for continuous compliance. This capability can work in concert with an external security compliance reporting tool, such as IBM Z Security and Compliance Center. Db2 for z/OS listens to the ENF 86 signal that is generated by the z/OS Compliance Agent services. Then, Db2 generates System Management Facilities (SMF) 1154 trace records for the recommended system security parameter settings.
- ▶ Improve productivity and ease of use for administrators deploying or authorizing packages.
Db2 13 provides the flexibility for DBAs to control the ownership of plans, packages, and SQL routine packages by using the DBA role without depending on the security administrator. New capability is added to identify the type of owner for plans and packages.
- ▶ Reduce contention and decrease downtime for concurrent IBM RACF® changes.
Db2 13 introduces a feature to cache a successful execution of plan authorization checks when using the Access Control Authorization (DSNX@XAC): Exit for access control.
Before Db2 13, this feature was available only for Db2 internal security checks. Removing this limitation improves performance for external security users by taking advantage of the caching in Db2. It also provides consistent behavior between Db2 native and external security controls.

1.6 Simplification and serviceability

Db2 13 helps reduce the total cost of ownership (TCO) through simplification, ease-of-use, and serviceability improvements. Although many of the enhancements that are already described have a focus on ease of use, a few areas are worthy of specific mention here.

Simplified database migration

The migration process from Db2 12 to Db2 13 improved in several ways to make your upgrade more controlled and less disruptive. These Db2 13 migration procedures borrow heavily from the Db2 12 continuous delivery process.

For example, the functional upgrade and catalog upgrade are controlled separately. Specific feature activation is governed by the function level that is set by you in the Db2 subsystem or at the application or package level. After a small initial catalog migration to Db2 13 (with no structural changes), you can defer further catalog migrations until you need a specific function that depends on that catalog level.

Additionally, because the initial catalog migration makes no structural changes, the contention with other concurrent activity is minimal, enabling true online version migration.

Migration to Db2 13 requires that your Db2 12 system is brought to function level 510 (FL510) and catalog level 509 (CL509), and it must have the fallback SPE APAR PH37108 applied. For more information about both the migration and the installation processes, see Chapter 2, “Installing and migrating” on page 13.

Application management

Managing application behavior for optimizing serialization and performance often requires individual editing or complex processes for using the same application in different contexts. Enhancements in Db2 13 help in several areas:

- ▶ More granular lock control is available at the application level by introducing a new lock timeout special register and deadlock weighting global variable. These items can be used to improve DDL break-in while reducing impact to applications.
- ▶ You can turn on and off some special registers and global variables without updating their local applications through an expansion of using application profiles.
- ▶ A new mechanism is introduced to the application profile tables that you can use to dynamically control the package **RELEASE** option. This mechanism increases your likelihood for successful DDL completion when Data Manipulation Language (DML) applications are running concurrently.
- ▶ An easy process is now available for allowing certain application packages to use new functions immediately while restricting others to an earlier level. Although introduced as a Db2 12 continuous delivery feature, changes in the Data Server Driver connectivity packages let you control new function level usage without performing package rebinds after each function level upgrade.

For more information about these application management changes, see Chapter 5, “Application management and SQL changes” on page 65.

Index Manager diagnostic and serviceability enhancements

With Db2 13, you can diagnose index-page-split performance issues more simply. During the index-split process, synchronous I/O is required in a data-sharing system when an index page set is GBP-dependent, which creates a significant performance impact.

A new Instrumentation Facility Component Identifier (IFCID) 396 record and three new catalog fields were added to record more detailed information about index splits. IFCID 396 is always enabled by default when Statistics Class 3 or Performance Class 6 are turned on. When an index split is considered an abnormal split process (for example, when the total elapsed time of an index split is greater than 1 second), a new IFCID 396 is generated, which includes detailed information about the index split. This information provides more analysis and should be helpful for both you and IBM to identify the root cause of **INSERT** performance issues.

EDITPROC support for IFCID 306

The Db2 instrumentation facility interface IFCID 306 reads log records for data replication products to process the changes to a table from SQL update operations such as **INSERT**, **UPDATE**, and **DELETE** statements. If the source table is encoded with an edit procedure (**EDITPROC**), IFCID 306 can find the **EDITPROC** to decode the log records of the encoded data. To provide you with more flexibility with IFCID 306 usage, Db2 13 allows you, in non-PROXY mode, to turn on or off the **EDITPROC** support function in IFCID 306.

Part range support for IFCID 306

Db2 13 helps users of replication products by providing more flexibility to supply partition filtering as a range of partitions to be processed. This enhancement provides better end-to-end performance, especially for replication products running multiple capture programs. Each capture process can work on different partitions of the same table space in parallel.

Improved diagnostics for Page Reject of Insert Row

Db2 13 adds more information to the Page Reject of Insert Row diagnostic log record, subtype (4400), now including one or more of the following sections:

- ▶ Data page rejection information
- ▶ Partition Retry Information
- ▶ Space map page search footprints

1.7 Utilities

The list of utilities enhancements in both Db2 13 and in the Db2 12 functional service stream is impressive. The Db2 12 enhancements are listed in detail in Appendix A, “IBM Db2 12 continuous delivery features and enhancements” on page 233.

Utilities history

Db2 can now record real-time and historical information about utility executions. This historical information can help you optimize your utility jobs and your overall utility strategy by providing the following capabilities:

- ▶ Check daily utility executions for failures and take immediate corrective actions.
- ▶ Ensure adherence to best practices or site standards for utilities.
- ▶ Analyze and compare utility information from one execution to another one.
- ▶ Analyze comparable utility executions.

Inline statistics page sampling (LOAD and REORG)

Db2 13 introduces page sampling for inline statistics during **REORG TABLESPACE** with **STATISTICS** or **LOAD REPLACE** with **STATISTICS**.

IBM performance measurements showed a significant, elapsed time and CPU time improvement.

REORG INDEX enforcement of NOSYSUT1 behavior

To promote better performance, the **REORG INDEX** utility no longer requires **NOSYSUT1** to be specified. When **REORG INDEX** is invoked with **SHRLEVEL REFERENCE** or **CHANGE**, the **NOSYSUT1** keyword specifies that **REORG INDEX** does not use work data sets to hold the unloaded index keys. In addition, multiple subtasks can be used to reorganize different partitions within a partitioned index in parallel, which results in shorter elapsed time and decreased CPU time.

RECOVER enhanced space level recovery

Db2 13 allows more flexibility to allow recovery when heterogeneous image copies are taken. As partition level image copies become a common approach rather than always capturing the entire table space in one data set, the **RECOVER** utility now allows for a mixing and matching of different granularities of image copies.

The **RECOVER** syntax for specifying recovery of the entire table space can now successfully use any space-level or partition-level image copies that are found when the **RECOVER** utility is running. This enhancement applies to recovery from sequential image copies, IBM FlashCopy® image copies that are created at the partition or piece level, or inline sequential or FlashCopy image copies taken by the **LOAD** or **REORG TABLESPACE** utilities.

Enabling independent software vendor applications for decompression dictionary log records

The **WRITELOG** option of the **REPAIR** utility is enhanced to allow independent software vendor (ISV) applications to write a decompression dictionary log record.

1.8 Db2 13 ecosystem performance, productivity, and DevOps enhancements

Beyond the core strengths of the Db2 for z/OS subsystem, the value of the platform is not complete without the extensive ecosystem that is provided by the many tools and features. Some of them are described in earlier sections, but there are several others that are described in this book that are critical to achieving the full capability and efficiency of Db2 13. This book also covers the latest features in these ecosystem components.

Db2 AI for z/OS

Db2ZAI improves operational efficiency. Db2ZAI 1.5 adds better visualization and drill down through performance insights and lets you govern distributed threads based on client user ID and application name. This product is described further in Chapter 10, “Machine learning and artificial intelligence infusion” on page 143.

Development and administrative tools

The GUI tools and application programming interfaces (APIs) that simplify and optimize application development and database administration continue to evolve and improve. Three features are described in this book that are either relatively new, or greatly improved, in synchronization with Db2 13. Chapter 11, “Development and administration tools” on page 155 describes the following offerings and recent enhancements in detail:

- ▶ IBM Db2 for z/OS Developer Extension

Db2 Developer Extension is a plug-in for working on Db2 for z/OS applications in the popular integrated development environment (IDE) Microsoft Visual Studio Code. This extension brings Db2 for z/OS awareness to the tool for SQL syntax, connection management, application editing, execution, and debugging.

You can work with all the programming languages that contain embedded SQL, such as COBOL, Java, Python, and Node.js. Version 1.4 of Developer Extension has full day-one support for Db2 13, recognizing all new SQL elements, syntax, and function signatures.

- ▶ IBM Db2 Administration Foundation for z/OS (Admin Foundation)

Admin Foundation is a robust GUI-based product for administering Db2 for z/OS and for analyzing and tuning SQL query performance. It can also be used to administer Db2 Analytics Accelerator. Its intuitive user interface enables you to navigate the Db2 catalog, run Db2 commands and SQL statements, generate DDL from the catalog for Db2 objects, easily find underperforming SQL statements and tune them, and many other common DBA tasks.

Admin Foundation is delivered as a no-charge component of IBM Unified Management Server for z/OS. The framework of Admin Foundation is provided by the open-source Zowe Virtual Desktop, which is a modern and intuitive alternative to traditional IBM Z interfaces.

- ▶ SQL Tuning Services

SQL Tuning Services provides a set of RESTFUL APIs that enables analysis and tuning of SQL applications. These APIs can perform the following tasks:

- Generate a visual representation of access plans for an SQL statement by using Visual Explain.
- Generate statistics data collection advice for database objects or for an SQL statement.
- Capture an SQL query environment, which is useful for re-creating an environment on a different Db2 subsystem or for diagnostic purposes.
- Perform administrative functions, such as managing **EXPLAIN** tables and jobs.

Although these APIs are available as a stand-alone offering, they are integrated within both Db2 Developer Extension and Admin Foundation.

Data Virtualization Manager for z/OS

Data Virtualization Manager for z/OS (DVM) for z/OS lets you read, write, and combine data from many sources on z/OS without requiring data movement and knowledge of the many complex access methods for each source. This capability makes the combined data easily consumable by web, analytics, and cloud applications. The DVM for z/OS server provides several resource optimization techniques, including the exploitation of IBM Z Integrated Information Processor (zIIP) processors to minimize cost.

Of course, Db2 for z/OS is an important virtualization target for DVM users. DVM can access Db2 data through Distributed Relational Database Architecture (DRDA) protocols for maximizing zIIP usage to ensure transactional integrity for read, update, and metadata activities. Alternatively, for read-only access to Db2 data, DVM can access the underlying 9 data sets directly for high-performance and bulk data access. Similarly, DVM optionally lets you access Db2 for z/OS unload files as a data source by using regular SQL. For more information about DVM for z/OS and how best to use it with Db2 13, see 12.8, “IBM Data Virtualization Manager for z/OS” on page 181.

Db2 for z/OS Tools Suite

The Db2 for z/OS Tools Suite provides over a dozen separate offerings that help simplify, automate, analyze, and optimize Db2 13 and the applications, processes, and health of the entire system. These tools have all been updated and tested to ensure support for Db2 13. Additionally, many tools are enhanced to leverage new Db2 13 features and recent Db2 12 enhancements. For more information, see Chapter 12, “IBM Db2 for z/OS tools support for Db2 13” on page 175.

Db2 Analytics Accelerator

Db2 Analytics Accelerator is an optional query acceleration feature for Db2 for z/OS that can speed the processing of complex queries by orders of magnitude over internal Db2 processing. The Db2 Analytics Accelerator solution has a few things in common with other vendors’ extract, transform, and load (ETL) solutions, but it is much more tightly integrated with Db2 for z/OS to manage all data utilities, replication, and query offload seamlessly. Recent Db2 Analytics Accelerator releases introduced a steady stream of improvements.

Here are a few recent high-value enhancements that extend and improve your acceleration capability:

- ▶ IBM Integrated Synchronization, which is an improved incremental update technology that is more tightly integrated with the Db2 subsystem with improved data coherency.
- ▶ Run accelerator queries that contain even more pass through-only analytical built-in functions that are recognized by the accelerator but not by Db2 for z/OS native SQL.
- ▶ Run accelerator queries that require pagination of result sets.
- ▶ Use the accelerator for data-sensitive uses cases that require masked columns.
- ▶ Ensure that certain queries are always routed or not routed to the accelerator by using dynamic plan stability.
- ▶ Extend accelerator-only tables (AOTs) to be defined on multiple accelerators for high availability (HA).
- ▶ For more information about Db2 Analytics Accelerator, see Chapter 13, “IBM Db2 Analytics Accelerator for z/OS” on page 201.

IBM Db2 for z/OS Data Gate

With IBM Db2 for z/OS Data Gate and IBM Cloud Pak® for Data, you can access Db2 for z/OS data without accessing Db2 for z/OS. This technology provides off-platform access to Db2 for z/OS data while reducing the cost and complexity that are found in many other data offloading alternatives. A key component of the Data Gate solution is using the Integrated Synchronization solution that maintains data consistency through replication technology and is tightly integrated with the Db2 for z/OS recovery log. For more information about IBM Db2 for z/OS Data Gate, see Chapter 14, “IBM Db2 for z/OS Data Gate” on page 225.

1.9 Summary of Db2 12 continuous delivery enhancements

Db2 12 for z/OS formally introduced the continuous delivery model in which new Db2 capabilities and enhancements are integrated into a single maintenance stream as the code becomes ready. At certain delineated function level points, an enabling PTF is released that activates the new enhancements that were integrated since the previous function level. Before Db2 12, there were occasional small enhancements that were delivered in the maintenance stream. Db2 12 continuous delivery formalized that process and provides a simple and controlled means for introducing new levels of functions and the option of leaving those capabilities disabled. The continuous delivery model continues in Db2 13.

Because the list of enhancements that is provided by Db2 12 continuous delivery is so extensive and some did not get the coverage that they might have at a release boundary, the appendixes of this book provide descriptions of the most significant of those enhancements. For more information, see the following appendixes:

- ▶ Appendix A, “IBM Db2 12 continuous delivery features and enhancements” on page 233
- ▶ Appendix B, “Application development” on page 253
- ▶ Appendix C, “Database administration” on page 263
- ▶ Appendix D, “System administration and system management” on page 289



Installing and migrating

Although the Db2 13 installation process remains largely unchanged from the previous version, changes to the migration process make moving to a new Db2 version easier than before.

The ability to perform an online migration has been at the top of Db2 administrators' wish list for the past releases, which means that it also has been a key focus point for the Db2 development team. Although great strides were made in this area over the past couple of releases, Db2 13 introduces some important enhancements that finally allow most, if not all, migrations to be accomplished online. The innovative differences that began in Db2 12 with function level V12R1M510 (FL 510) continue with trailblazing changes to the CATMAINT (DSNTIJTC) job and process in Db2 13, which can drastically improve your migration experience.

This chapter describes the following topics:

- ▶ Highlights
- ▶ Prerequisites for migrating to Db2 13
- ▶ Function level V13R1M100
- ▶ Enabling V13R1M500 new function levels with no catalog dependencies
- ▶ Enabling FL 501 new functions with catalog dependencies
- ▶ Using the installation CLIST to migrate to Db2 13 FL 100, FL 500, or FL 501
- ▶ Using the installation CLIST to install Db2 13
- ▶ Subsystem parameter changes
- ▶ Verifying catalog- and function-level changes

2.1 Highlights

The following sections summarize several key aspects about the Db2 13 migration process.

2.1.1 Migrating to Db2 13 is consistent with continuous delivery

Db2 13 continues to use the continuous delivery concepts that were introduced in Db2 12. However, one difference is that the first catalog level in Db2 13 is V13R1M100, and the initial catalog level in Db2 12 was V12R1M500.

After a successful migration to Db2 13, the initial function level is V13R1M100 (FL 100), and the catalog level is also V13R1M100 (CL 100).

After FL 100, Db2 13 subsystems and data-sharing groups progress through FL 500, catalog level V13R1M501 (CL 501), and finally to FL 501. These three function levels are available at general availability of Db2 13.

A new function that does not depend on new-release catalog changes becomes available when you activate FL 500. Then, after you update the catalog level to CL 501, you can activate FL 501 to start using the full complement of Db2 13 GA-level new functions.

Important: You cannot activate function level V13R1M500 in a data-sharing environment that has any active Db2 12 members, and when V13R1M500 is first activated, it is no longer possible to fall back to Db2 12 or have a release coexistence between Db2 12 and Db2 13.

Function-level and catalog-level changes are group-wide events in data sharing. As such, each member in a data-sharing group has the same function and catalog levels. When one member changes a function or catalog level, that change takes effect for all active members.

2.1.2 The catalog migration process and input changed

[Migrating your Db2 subsystem to Db2 13](#) describes 27 migration tasks compared to 28 in Db2 12, so at first glance migrating to Db2 13 might not appear to be any simpler than in previous versions. However, the content of the migration steps is far more important than the number of steps.

The most significant improvement to the Db2 13 migration process is the timing of the new release catalog changes. In earlier Db2 releases, new release catalog changes were made during the initial release migration process by catalog migration (CATMAINT) (job DSNTIJTC) processing.

Db2 13 still uses an initial release migration CATMAINT, but this CATMAINT job no longer changes the structure of the Db2 catalog, that is, it does not create tables, indexes, or columns in existing catalog tables.

You might wonder why a CATMAINT job is still needed if the structure of the catalog is not changing. The reason is so that you can control the timing of when the migration completes and when a subsystem is available for use. This initial CATMAINT process also sets internal information to indicate that the catalog level is now V13R1M100 (CL 100) and that the function level is V13R1M100 (FL 100).

These migration process changes are important and revolutionary for the following reasons:

- ▶ If the structure of the Db2 catalog does not change on a release boundary, Db2 no longer must add special code in the fallback release (in this case, Db2 12) that understands and handles both the old and new versions of a system object. Db2 12 is never exposed to any of the Db2 13 catalog changes. This improvement eliminates much of the complexity of the fallback small programming enhancement (SPE) in Db2 12 code.

- ▶ If the structure of the catalog is not changed during a release migration, plans and packages no longer need to be invalidated during the migration process. Therefore, Db2 eliminates the post-migration AUTOBIND issues that might otherwise have occurred during the migration process, such as the potential for resulting access path changes.

In release coexistence or fallback situations, Db2 likewise eliminates the possibility of an AUTOBIND occurring each time that an application is run on a different release.

- ▶ Because the structure of the catalog does not change, there is no longer contention between applications and utilities that might be running when CATMAINT is running. The Db2 13 CATMAINT job still obtains a database descriptor (DBD) lock for DSNDB06, but this process is quick and is unlikely to impact applications or user processes.

With the elimination of package and plan invalidations, SQL contention, and deadlock issues, the final roadblocks are removed for achieving true online migrations.

2.1.3 Application compatibility toleration is unchanged

No changes were introduced to the earlier release application compatibility (APPLCOMPAT) values that can be specified in Db2 13. Db2 13 adds only the ability to specify new Db2 13 APPLCOMPAT values.

Although Db2 13 still allows the usage of applications that are bound with APPLCOMPAT values of V10R1 and higher, it is a best practice to resolve incompatibilities and bind or rebind packages with more current APPLCOMPAT levels.

2.2 Prerequisites for migrating to Db2 13

Function level V12R1M510 must be activated in Db2 12 before a subsystem or data-sharing group is eligible for migrating to Db2 13. V12R1M510 is a special function level in Db2 12 that does not enable any new function. Instead, it ensures that a subsystem or data-sharing group is ready for migration to Db2 13.

The prerequisite catalog level for V12R1M510 is V12R1M509. Another requirement is that no packages that were used in the previous 18 months were last bound or rebound in any release earlier than Db2 11.

Before V12R1M510 activation, run the **SELECT** statement that is shown in Figure 2-1, which is provided in the DSNTIJPE SAMPLE job, to proactively determine whether the V12R1M510 activation will complete successfully.

```
SELECT STRIP(COLLID),
       STRIP(NAME),
       STRIP(VERSION),
       TYPE
FROM SYSIBM.SYSPACKAGE
WHERE LASTUSED >= DATE(DAYS(CURRENT DATE) - 548) AND
      RELBOUND NOT IN ('P', 'Q', 'R') AND
      VALID <> 'N' AND
      OPERATIVE <> 'N';
```

Figure 2-1 *SELECT* statement to determine whether the V12R1M510 activation will complete

If the DSNTIJPE SAMPLE job returns any rows for this sample query, it creates a data set that contains the **REBIND** statements that, when run, allow V12R1M510 to be activated.

As a best practice, run this sample query in advance of a scheduled function level change to V12R1M510 to ensure that a system catalog is in a good condition for the coming changes.

During activation of V12R1M510, the sample query runs. If it returns any rows, activation of V12R1M510 fails. A **REBIND** of all displayed packages allows V12R1M510 to be activated.

Db2 12 subsystems and members must have the PH37108 code with Db2 started before attempting to migrate to Db2 13. If the fallback SPE APAR (PH37108) is not on all active members of a data-sharing group or a non-data-sharing subsystem before a Db2 13 migration, the migration to Db2 13 fails.

2.3 Function level V13R1M100

Externally, the Db2 13 migration process changed little from the Db2 12 migration process.

However, the major Db2 13 migration process enhancement is that the structure of the Db2 catalog is not changed during the initial CATMAINT (DSNTIJTC) processing for catalog level V13R1M100 (CL 100). Running a CATMAINT job is still part of the migration process, but unlike in previous new Db2 versions, it does not add or change any existing catalog objects during processing. It functions as a switch that allows you to determine when migration processing completes.

2.3.1 Coexistence between V12R1M510 members and V13R1M100 members

Release coexistence between Db2 12 and Db2 13 is available only while function level V13R1M100 is the highest activated function level in Db2 13. If function level V13R1M500 or higher is activated in Db2 13, data-sharing group members can no longer be in release coexistence with Db2 12. Function levels V13R1M500 and higher cannot be activated if a data-sharing group has any active members still running Db2 12.

2.3.2 New function availability in function level V13R1M100

Some enhancements are available in Db2 13 starting with function level V13R1M100. For example, when you migrate to Db2 13, many virtual storage and optimization enhancements take effect, such as reducing extended common service area (ECSA) storage consumption, supporting IBM Z Security and Compliance Center with automated data collection, tracking Group Buffer Pool residency times, and analyzing data from a **SORTL** instruction to reduce workfile usage on subsequent SQL execution. However, most new application and SQL capabilities become available only after you activate function level V13R1M500 or V13R1M501 and use the corresponding APPLCOMPAT level.

2.4 Enabling V13R1M500 new function levels with no catalog dependencies

You can activate new function levels when all prerequisites are met, which usually means two things:

- ▶ All active members of a Db2 data-sharing group are operating on a level of code that understands the function level that is being activated.
- ▶ The catalog is at the required level for the function level that is being activated.

If no catalog changes are required for a function level to be activated, the code level requirement might be the only activation prerequisite.

Db2 13 provides the code for FL 100, FL 500, and FL 501, so you do not need to apply any program temporary fixes (PTFs) to activate any of these function levels.

There are no catalog changes for FL 500 in Db2 13.

2.4.1 Verifying all data-sharing members are at function level V13R1M100

Function levels and catalog levels are group-wide levels in data sharing. V13R1M500 and higher can be activated only if the data-sharing group has no active Db2 12 members.

You can use the **DISPLAY GROUP** command to determine the code level of a subsystem and, for a data-sharing group, the code levels of all members and their current function and catalog levels. If the **DISPLAY GROUP** command shows that the data-sharing group has any active Db2 12 members, the entire group is not eligible for FL 500 activation.

You can also use the **ACTIVATE** command with the **TEST** option to determine whether a subsystem or data-sharing group is eligible for the activation of a particular function level. The output of the **ACTIVATE** command provides information such as the output of the **DISPLAY GROUP** command.

2.4.2 Activating V13R1M500

There are no catalog changes for V13R1M500 (FL 500). You can activate FL 500 if there are no active Db2 12 members in a data-sharing environment. Catalog- and function-level changes are group-wide events in data sharing. These changes should not require application outages, and application impacts are unlikely.

You can issue the following command to activate FL 500:

```
ACTIVATE FUNCTION LEVEL (V13R1M500)
```

You can also use a sample job that the installation CLIST generates to run this command, which is described in 2.6, “Using the installation CLIST to migrate to Db2 13 FL 100, FL 500, or FL 501” on page 19 .

2.4.3 Falling back to Db2 12 no longer possible after activating V13R1M500

After V13R1M500 (FL 500) is activated, it is no longer possible to fall back to Db2 12 or to be in release coexistence in a data-sharing environment. Although you can activate function level V13R1M100 again to change the subsystem or data-sharing group to function level V13R1M100* (FL 100* or FL 100 star), activating function level V13R1M100* does not give a subsystem or data-sharing group the ability to return to Db2 12.

Returning to an earlier function level can be useful when you want to prevent the usage of a new function from a higher function level. A star function level indicates that the system or data-sharing group was previously at a higher function level.

You can query the SYSLEVELUPDATES catalog table to see the history of catalog changes and function level activations. You can also use the **DISPLAY GROUP** command to determine the highest activated function level.

2.4.4 APPLCOMPAT level V13R1M500

As in Db2 12, function level and APPLCOMPAT values work closely together to allow or disallow the usage of some Db2 13 functions. Most new SQL syntax and behaviors in V13R1M500 (FL 500) also require packages to use an equivalent or higher APPLCOMPAT level (APPLCOMPAT V13R1M500). As a result, you can rebind applications with a lower APPLCOMPAT value to prevent the usage of some new-release function.

2.5 Enabling FL 501 new functions with catalog dependencies

The first Db2 13 catalog changes are made for FL 501. The V13R1M501 CATMAINT job (DSNTIJTC) must be run while a subsystem or data-sharing group is in FL 500. FL 501 cannot be activated if the catalog is not yet at catalog level V13R1M501 (CL 501).

You can make these CL 501 catalog changes in advance of activating FL 501. There is no requirement that the CL 501 changes are made immediately before FL 501 activation.

2.5.1 Updating the catalog to V13R1M501

To update the Db2 catalog to V13R1M501 (CL 501), use the following **CATMAINT** statement in job DSNTIJTC:

```
CATMAINT UPDATE LEVEL V13R1M501
```

In continuous delivery, input to **CATMAINT** can be either the wanted function level or catalog level. **CATMAINT** processing determines the catalog level that is needed for the specified value. In function level V13R1M501, you can specify a level of V13R1M501 only because at Db2 13 GA, no other valid catalog or function level options are available.

2.5.2 Activating V13R1M501

You can activate V13R1M501 (FL 501) at any point after the catalog is changed to V13R1M501 (CL 501). FL 501 can be activated immediately after the CL 501 changes are made or at any other time. There is no requirement that FL 501 is activated immediately after the CL 501 changes are made. You have the flexibility to make catalog- and function-level changes at the same time or at different times.

Function level V13R1M501 can be activated by using the following **ACTIVATE** command:

```
ACTIVATE FUNCTION LEVEL (V13R1M501)
```

You can also use the installation CLIST to generate a sample job for this command, which is described later in 2.6, “Using the installation CLIST to migrate to Db2 13 FL 100, FL 500, or FL 501” on page 19.

2.5.3 APPLCOMPAT level V13R1M501

Most new SQL syntax and behaviors in V13R1M501 (FL 501) also require packages to use the equivalent APPLCOMPAT level (APPLCOMPAT V13R1M501). As a result, you can rebind applications with a lower APPLCOMPAT value to prevent the usage of some new-release functions.

2.6 Using the installation CLIST to migrate to Db2 13 FL 100, FL 500, or FL 501

The main Db2 13 migration process enhancement is that the structure of the Db2 catalog is not changed during the initial CATMAINT (DSNTIJTC) processing for catalog level V13R1M100 (CL 100). There is still a CATMAINT job, but it does not add catalog objects or change any existing catalog objects. It functions as a switch that allows you to control when migration processing completes.

The following sections document both the traditional approach to migration and using z/OSMF to migrate a Db2 subsystem.

2.6.1 The traditional migration approach

A target function level of V13R1M500 (FL 500) or higher is always accomplished by running the installation CLIST in **ACTIVATE** mode. Therefore, when you migrate a subsystem or the first member of a data-sharing group to a specific function level, the traditional migration process continues to require the actions in the following sections to migrate from Db2 12 to Db2 13 with a function level of FL 500 or higher:

- ▶ Migrating to Db2 13 function level V13R1M100
- ▶ Activating function levels V13R1M500 or V13R1M501

Migrating to Db2 13 function level V13R1M100

To migrate a Db2 subsystem, you invoke the installation CLIST. The first panel, DSNTIPA1, displays a list of operations to choose from. For the traditional migration approach, you specify NO for USE Z/OSMF WORKFLOW and MIGRATE for INSTALL TYPE, as shown in Figure 2-2.

```
DSNTIPA1 DB2 13 INSTALL, UPDATE, MIGRATE AND PROVISION - MAIN PANEL
===> _
Will the installation/migration be performed with z/OSMF?
  USE Z/OSMF WORKFLOW   ===> NO           Yes or No (Yes for Provision)

Check parameters and reenter to change:
  1 INSTALL TYPE       ===> MIGRATE      Install, Migrate, Activate, Update
                                          or Provision
  2 DATA SHARING      ===> NO           Yes or No (blank for Activate, Update)

For migration only: Enter the data set and member name with the settings from
the previous Installation/Migration:
  3 DATA SET(MEMBER) NAME ===> DB2V12.INSTALL.JCLLIB(DSNTIDBA)
For Db2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
  4 LIBRARY NAME PREFIX ===> DSND10
  5 LIBRARY NAME SUFFIX ===>
For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
  6 DATA SET NAME PREFIX ===> DSND10
  7 DATA SET NAME SUFFIX ===>
Enter to set or save panel values (by reading or writing the named members):
  8 INPUT MEMBER NAME   ===> DSNTIDXA   Default parameter values
  9 OUTPUT MEMBER NAME  ===> DSNTIDBA   Save new values entered on panels
PRESS: ENTER to continue RETURN to exit HELP for more information
```

Figure 2-2 DSNTIPA1 panel

Next, on panel DSNTIP00, as shown in Figure 2-3, the installation CLIST initializes the CURRENT FUNCTION LEVEL to V12R1M510 and TARGET FUNCTION LEVEL to V13R1M100. These two fields are not updatable, and they are used by the installation CLIST to generate the jobs that migrate a Db2 subsystem to function level V13R1M100.

```
DSNTIP00 MIGRATE DB2 - ACTIVATE A FUNCTION LEVEL FOR DB2
===> _
Enter the current and target Db2 function levels:
  1 CURRENT FUNCTION LEVEL===> V12R1M510 (Db2's current function level)
  2 TARGET FUNCTION LEVEL ===> V13R1M100 (Db2 function level to be activated)

Enter the default application compatibility level:
  3 APPL COMPAT LEVEL     ===> V12R1M500 (Db2 function level or V11R1 or V10R1)

Enter the default SQL level for the precompiler:
  4 PRECOMPILER SQL LEVEL ===> V12R1M500 (Db2 function level or V11R1 or V10R1)

PRESS: ENTER to continue RETURN to exit HELP for more information
```

Figure 2-3 DSNTIP00 panel

The Db2 13 migration process requires the CATMAINT (DSNTIJTC) job to be run with a level of V13R1M100. This Db2 13 CATMAINT job does not change the structure of the Db2 catalog, but the Db2 13 migration process is not considered complete until this CATMAINT job is run. After it completes successfully, the function level becomes V13R1M100 (FL 100), and the catalog level changes to V13R1M100 (CL 100). Function and catalog levels are group-wide values in data sharing. Therefore, each member is always at the same level for both.

Activating function levels V13R1M500 or V13R1M501

After migrating a Db2 subsystem to function level V13R1M100, you can activate a higher function level. To activate a higher function level, you invoke the installation CLIST. The first panel, DSNTIPA1, displays a list of operations to choose from. For traditional function level activation, you specify NO for USE Z/OSMF WORKFLOW and ACTIVATE for INSTALL TYPE, as shown in Figure 2-4.

```
DSNTIPA1 DB2 13 INSTALL, UPDATE, MIGRATE AND PROVISION - MAIN PANEL
===> _
Will the installation/migration be performed with z/OSMF?
USE Z/OSMF WORKFLOW   ===> NO           Yes or No (Yes for Provision)

Check parameters and reenter to change:
1 INSTALL TYPE         ===> ACTIVATE    Install, Migrate, Activate, Update
or Provision
2 DATA SHARING        ===>            Yes or No (blank for Activate, Update)

For migration only: Enter the data set and member name with the settings from
the previous Installation/Migration:
3 DATA SET(MEMBER) NAME ===>
For Db2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
4 LIBRARY NAME PREFIX  ===> DSND10
5 LIBRARY NAME SUFFIX  ===>
For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
6 DATA SET NAME PREFIX ===> DSND10
7 DATA SET NAME SUFFIX ===>
Enter to set or save panel values (by reading or writing the named members):
8 INPUT MEMBER NAME    ===> DSNTIDBA   Default parameter values
9 OUTPUT MEMBER NAME   ===> DSNTIABA   Save new values entered on panels
PRESS: ENTER to continue RETURN to exit HELP for more information
```

Figure 2-4 DSNTIPA1 panel

Next, on panel DSNTIP00, you specify the function level to be activated for TARGET FUNCTION LEVEL, as shown in Figure 2-5.

```
DSNTIP00 ACTIVATE DB2 - ACTIVATE A FUNCTION LEVEL FOR DB2
===> _
Enter the current and target Db2 function levels:
1 CURRENT FUNCTION LEVEL===> V13R1M100 (Db2's current function level)
2 TARGET FUNCTION LEVEL ===> V13R1M500 (Db2 function level to be activated)

Enter the default application compatibility level:
3 APPL COMPAT LEVEL     ===> V12R1M500 (Db2 function level or V11R1 or V10R1)

Enter the default SQL level for the precompiler:
4 PRECOMPILER SQL LEVEL ===> V12R1M500 (Db2 function level or V11R1 or V10R1)

PRESS: ENTER to continue RETURN to exit HELP for more information
```

Figure 2-5 DSNTIP00 panel

Depending on the TARGET FUNCTION LEVEL that you specify, the installation CLIST generates the jobs that are required to complete the target function level activation from V13R1M100.

TARGET FUNCTION LEVEL V13R1M500

For this function level, the following actions occur:

- ▶ Job DSNTIJTC updates catalog level V13R1M500. This job is optional.
- ▶ Job DSNTIJAF activates function level V13R1M500 (FL 500). There are no catalog changes for FL 500. FL 500 can be activated if there are no active Db2 12 members in a data-sharing environment. After function level FL 500 is activated, fall back to Db2 12 or release coexistence in a data-sharing environment is no longer possible.

TARGET FUNCTION LEVEL V13R1M501

For this function level, the following actions occur:

- ▶ Job DSNTIJA0 activates function level V13R1M500 (FL 500). There are no catalog changes for FL 500. FL 500 can be activated if there are no active Db2 12 members in a data-sharing environment. After FL 500 is activated, fall back to Db2 12 or release coexistence in a data-sharing environment is no longer possible.
- ▶ Job DSNTIJTC changes the structure of the Db2 catalog, and it changes the catalog level to V13R1M501 (CL 501). The V13R1M501 CATMAINT job must be run while a subsystem or data-sharing group is in function level V13R1M500 (FL 500). FL 501 cannot be activated if the catalog level is not V13R1M501.
- ▶ Job DSNTIJAF activates FL 501. FL 501 can be activated anytime after the catalog is changed to V13R1M501.
- ▶ Job DSNTIJX2 checks Db2 catalog indexes for new objects that were created in FL 501.

2.6.2 Using z/OSMF to migrate a Db2 subsystem

To migrate a Db2 subsystem by using z/OSMF, you invoke the installation CLIST. The first panel, DSNTIPA1, displays a list of operations to choose from. For migrations that use z/OSMF, you specify YES for USE Z/OSMF WORKFLOW and MIGRATE for INSTALL TYPE, as shown Figure 2-6.

```
DSNTIPA1 DB2 13 INSTALL, UPDATE, MIGRATE AND PROVISION - MAIN PANEL
===>
Will the installation/migration be performed with z/OSMF?
  USE Z/OSMF WORKFLOW   ===> YES      Yes or No (Yes for Provision)

Check parameters and reenter to change:
  1 INSTALL TYPE         ===> MIGRATE   Install, Migrate, Activate, Update
                                or Provision
  2 DATA SHARING       ===> NO        Yes or No (blank for Activate, Update)

For migration only: Enter the data set and member name with the settings from
the previous Installation/Migration:
  3 DATA SET(MEMBER) NAME ===> DB2V12.INSTALL.JCLLIB(DSNTIDBA)
For Db2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
  4 LIBRARY NAME PREFIX  ===> DSND10
  5 LIBRARY NAME SUFFIX  ===>
For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
  6 DATA SET NAME PREFIX ===> DSND10
  7 DATA SET NAME SUFFIX ===>
Enter to set or save panel values (by reading or writing the named members):
  8 INPUT MEMBER NAME    ===> DSNTIDXA Default parameter values
  9 OUTPUT MEMBER NAME   ===> DSNTIDBA Save new values entered on panels
PRESS: ENTER to continue RETURN to exit HELP for more information
```

Figure 2-6 DSNTIPA1 panel

Next, on panel DSNTIP00, you specify the function level to be activated for TARGET FUNCTION LEVEL, as shown in Figure 2-7 on page 23.


```

DSNTIP00          MIGRATE DB2 - ACTIVATE A FUNCTION LEVEL FOR DB2
==> _
Enter the current and target Db2 function levels:
  1 CURRENT FUNCTION LEVEL==> V12R1M510 (Db2's current function level)
  2 TARGET FUNCTION LEVEL ==> V13R1M501 (Db2 function level to be activated)

Enter the default application compatibility level:
s 3 APPL COMPAT LEVEL      ==> V12R1M500 (Db2 function level or V11R1 or V10R1)

Enter the default SQL level for the precompiler:
s 4 PRECOMPILER SQL LEVEL ==> V12R1M500 (Db2 function level or V11R1 or V10R1)

DSNT567I Enter values for activating Db2 function levels after initial
migration to V13R1M100

```

Figure 2-7 DSNTIP00 panel

The installation CLIST generates the two z/OSMF workflows, which contain the steps for completing the migration from Db2 12 to the Db2 13 function level that you specified in the TARGET FUNCTION LEVEL field. The step numbers that mentioned in these workflows do not correspond to the step numbers that are listed in [Migrating your Db2 subsystem to Db2 13](#).

Workflow 1: DSNTIWMS - Migrating to Db2 13 function level V13R1M100

Step step18a (DSNTIJTC) does not change the structure of the Db2 catalog, but the Db2 13 migration process is not considered complete until this job is run. After DSNTIJC completes successfully, the function level becomes V13R1M100, and the catalog level changes to level V13R1M100.

Workflow 2: DSNTIWMN - Migration from V13R1M100 to the specified TARGET FUNCTION LEVEL

The workflow steps depend on the TARGET FUNCTION LEVEL that you specified.

TARGET FUNCTION LEVEL V13R1M500

Step step28 (DSNTIJAF) activates function level V13R1M500 (FL 500). There are no catalog changes for FL 500. FL 500 can be activated if there are no active Db2 12 members in a data-sharing environment. After FL 500 is activated, fall back to Db2 12 or release coexistence in a data-sharing environment is no longer possible.

TARGET FUNCTION LEVEL V13R1M501

For this function level, the following actions occur:

- ▶ Step step27a (DSNTIJA0) activates function level V13R1M500 (FL 500). There are no catalog changes for FL 500. FL 500 can be activated if there are no active Db2 12 members in a data-sharing environment. After FL 500 is activated, fall back to Db2 12 or release coexistence in a data-sharing environment is no longer possible.
- ▶ Step step27b (DSNTIJTC) changes the structure of the Db2 catalog and the catalog level changes to V13R1M501 (CL 501). The V13R1M501 CATMAINT job must be run while a subsystem or data-sharing group is in function level V13R1M500 (FL 500). Function level V13R1M501 (FL 501) cannot be activated if the catalog level is not V13R1M501.

- ▶ Step step28 (DSNTIJAF) activates FL 501. FL 501 can be activated anytime after the catalog is changed to level V13R1M501.
- ▶ Step step28a (DSNTIJX2) checks Db2 catalog indexes for new objects that were created in FL 501.

2.7 Using the installation CLIST to install Db2 13

The Db2 13 for z/OS installation process is largely unchanged from the one for Db2 12 except that both the traditional installation and the installation that use z/OSMF automatically installs Db2 13 at function level V13R1M501 (instead of at V13R1M500).

To install a Db2 13 subsystem, you invoke the installation CLIST. The first panel, DSNTIPA1, shown in Figure 2-8, displays a list of operations to choose from. For a new Db2 13 installation, you specify INSTALL for INSTALL TYPE.

For a traditional Db2 installation, you specify NO for USE Z/OSMF WORKFLOW; for a Db2 installation that uses z/OSMF, you specify YES for USE Z/OSMF WORKFLOW as shown in Figure 2-8.

```

DSNTIPA1 DB2 13 INSTALL, UPDATE, MIGRATE AND PROVISION - MAIN PANEL
===>
Will the installation/migration be performed with z/OSMF?
USE Z/OSMF WORKFLOW   ==> NO           Yes or No (Yes for Provision)

Check parameters and reenter to change:
 1 INSTALL TYPE       ==> INSTALL      Install, Migrate, Activate, Update
                                     or Provision
 2 DATA SHARING      ==> NO           Yes or No (blank for Activate, Update)

For migration only: Enter the data set and member name with the settings from
the previous Installation/Migration:
 3 DATA SET(MEMBER) NAME ==>
For Db2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
 4 LIBRARY NAME PREFIX  ==> DSND10
 5 LIBRARY NAME SUFFIX  ==>
For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
 6 DATA SET NAME PREFIX ==> DSND10
 7 DATA SET NAME SUFFIX ==>
Enter to set or save panel values (by reading or writing the named members):
 8 INPUT MEMBER NAME    ==> DSNTIDXA  Default parameter values
 9 OUTPUT MEMBER NAME   ==> DSNTIDBA  Save new values entered on panels
PRESS: ENTER to continue RETURN to exit HELP for more information

```

Figure 2-8 DSNTIPA1 panel

Next, on panel DSNTIP00, the installation CLIST initializes the TARGET FUNCTION LEVEL field as V13R1M501, as shown in Figure 2-9 on page 25. This field is not updatable, and the installation CLIST always generates the jobs to install a Db2 subsystem at function level V13R1M501.

```

DSNTIP00          INSTALL DB2 - ACTIVATE A FUNCTION LEVEL FOR DB2
===> _
Enter the current and target Db2 function levels:
 1 CURRENT FUNCTION LEVEL===>          (Db2's current function level)
 2 TARGET FUNCTION LEVEL ==> V13R1M501 (Db2 function level to be activated)

Enter the default application compatibility level:
 3 APPL COMPAT LEVEL    ==> V13R1M501 (Db2 function level or V11R1 or V10R1)

Enter the default SQL level for the precompiler:
 4 PRECOMPILER SQL LEVEL ==> V13R1M501 (Db2 function level or V11R1 or V10R1)

PRESS:  ENTER to continue  RETURN to exit  HELP for more information

```

Figure 2-9 DSNTIP00 panel

For new Db2 13 installations, you must run a CATMAINT job (DSNTIJTC) during installation processing. This required CATMAINT job determines whether any CCSID-related updates must be made. The encoding of some special characters can vary between different Extended Binary Coded Decimal Interchange Code (EBCDIC) coded character set identifiers (CCSIDs). This required CATMAINT job identifies these encoding differences in the system catalog information and changes any characters that must be updated based on the source and target CCSIDs.

2.8 Subsystem parameter changes

Typically in a new Db2 release, some new subsystem parameters (DSNZPARMs) are added; some are changed; and some are removed or deprecated. The following sections describe the subsystem parameter changes in Db2 13:

- ▶ New subsystem parameters
- ▶ Changes to existing subsystem parameters
- ▶ Removed subsystem parameters

2.8.1 New subsystem parameters

- ▶ **DSN6SPRM.TABLE_COL_NAME_EXPANSION** controls whether column names can be longer than 30 bytes, up to a maximum length of 128 EBCDIC bytes:
 - Added: Function level V13R1M100
 - Valid values:
 - OFF: Column names longer than 30 EBCDIC bytes must not be specified.
 - 0N: Column names can be up to 128 EBCDIC bytes.
 - Default: OFF
- ▶ **DSN6SPRM.SPREG_LOCK_TIMEOUT_MAX** controls the value that can be specified in a **SET CURRENT LOCK TIMEOUT** statement:
 - Added: Function level V13R1M100
 - Valid values:
 - -1: Any supported value can be specified in a **SET CURRENT LOCK TIMEOUT** statement.

- 0-32767: The maximum value that can be specified in a **SET CURRENT LOCK TIMEOUT** statement.
- Default: -1
- ▶ **DSN6SPRM.UTILITY_HISTORY** specifies whether utility history information is collected:
 - Added: Function level V13R1M501
 - Valid values:
 - NONE: Utility history information is not collected. Information is not inserted into the SYSIBM.SYSUTILITIES catalog table.
 - UTILITY: Utility history information is collected. One row is inserted into the SYSIBM.SYSUTILITIES catalog table at the start of each utility execution. Information in the row is updated as the utility progresses. The final information is updated in the row when the utility execution finishes.
 - Default: NONE

2.8.2 Changes to existing subsystem parameters

Table 2-1 shows the new default values of Db2 subsystem parameters that are changed in Db2 13. The new defaults are effective from Db2 function level V13R1M100.

Table 2-1 Default values of Db2 subsystem parameters that are changed in Db2 13

Subsystem parameter	Old default value	New default value
DSN6FAC.DDF	NO	AUTO
DSN6FAC.MAXCONQN	OFF	ON
DSN6FAC.MAXCONQW	OFF	ON
DSN6LOGP.OUTBUFF	4000K	104857600
DSN6SPRM.EDM_SKELETON_POOL	51200	81920
DSN6SRPM.EDMDBDC	23400	40960
DSN6SPRM.FTB_NON_UNIQUE_INDEX	NO	YES
DSN6SPRM.MAXSORT_IN_MEMORY	1000	2000
DSN6SPRM.NUMLKTS	2000	5000
DSN6SPRM.NUMLKUS	10000	20000
DSN6SPRM.PAGESET_PAGENUM	ABSOLUTE	RELATIVE
DSN6SPRM.SRTPPOOL	10000	20000
DSN6SYSP.STATIME_MAIN	60	10

In addition to these changes to subsystem parameter default values, the behaviors of the following subsystem parameters also changed:

- ▶ **DSN6SPRM.AUTHEXIT_CACHEREFRESH**: If the subsystem parameter value is set to ALL and the z/OS release is 2.5 or later, Db2 refreshes the entries in the plan authorization cache when a resource access on the plan object profile is changed in Resource Access Control Facility (RACF) and when the access control authorization exit (DSNX@XAC) is active.
- ▶ **DSN6SPRM.DSMAX**: The maximum value is increased from 200000 to 400000.

- ▶ **DSN6SPRM.IRLMRWT:** The subsystem parameter is changed to online updatable to support more granular specification of the lock timeout value.
- ▶ **DSN6SPRM.REORG_INDEX_NOSYSUT1:** In Db2 function level V13R1M500 and higher, the subsystem parameter no longer influences whether **REORG INDEX SHRLEVEL REFERENCE** or **CHANGE** uses the **NOSYSUT1** behavior, which is always enabled as the default and only option.
- ▶ **DSN6SPRM.STATPGSAMP:** In Db2 function level V13R1M500 and higher, the subsystem parameter also applies to the collection of inline statistics when the **LOAD** or **REORG TABLESPACE** utilities run with the **STATISTICS** keyword.

2.8.3 Removed subsystem parameters

Important: For best results, check the subsystem parameters settings that are used in your Db2 12 environment, especially in data-sharing environments. If the current setting does not match the setting that is listed in Table 2-2, evaluate whether any other changes are needed to accept the new-behavior settings before migrating to Db2 13.

The subsystem parameters that are shown in Table 2-2 are removed in Db2 13, and their respective values can no longer be changed.

Table 2-2 *Removed subsystem parameters*

Subsystem parameter	Setting that is used in Db2 13
AUTHCACH	4K
DDF_COMPATIBILITY	NULL
HONOR_KEEPPDICTIONARY	NO
DSVCI	YES
EXTRAREQ	100
EXTRSRV	100
IMMEDWRI	NO
IX_TB_PART_CONV_EXCLUDE	YES
MAXARCH	10000
MAXTYPE1	0
OPT1ROWBLOCKSORT	DISABLE
PARA_EFF	50
PLANMGMTSCOPE	STATIC
REALSTORAGE_MANAGEMENT ^a	AUTO
RESYNC	2
SUBQ_MIDX	ENABLE
TRACSTR	NO

- a. If your Db2 12 environment currently uses the **REALSTORAGE_MANAGEMENT** subsystem parameter with a setting other than **AUTO**, you should not change it before migration. Db2 13 uses enhanced automatic behavior, which avoids causing unnecessary z/OS Real Storage Manager (RSM) serialization from discard requests and is not available in Db2 12.

2.9 Verifying catalog- and function-level changes

Db2 message **DSNG014I** is issued to the MVS console whenever there is a catalog or function level change, as shown in Figure 2-10.

```
21172 12:57:59.16 DSNG014I -DB2A DB2 NEW CATALOG LEVEL (V13R1M100 )
                        DB2 CATALOG LEVEL (V13R1M100 )
                        CURRENT FUNCTION LEVEL (V13R1M100 )
21172 12:57:59.16 DSNG014I -DB2B DB2 NEW CATALOG LEVEL (V13R1M100 )
                        DB2 CATALOG LEVEL (V13R1M100 )
                        CURRENT FUNCTION LEVEL (V12R1M510 )
```

Figure 2-10 Db2 message **DSNG014I**

The **DSNG014I** messages indicate the exact time a function level or catalog change was made. This message is issued on each member of a data-sharing group for each catalog or function level change. You can also query the **SYSLEVELUPDATES** catalog table to determine when catalog and function levels changed. This table maintains a history of all such activities for a subsystem or data-sharing group.

2.9.1 Activating lower function levels in Db2 13

You can activate lower function levels the same way that you can in Db2 12. These earlier function levels are called *star-function* levels and are typically denoted with an asterisk (*). Star-function levels indicate that the subsystem or data-sharing group previously was at a higher function level. You can query the **SYSLEVELUPDATES** catalog table for details about when catalog updates were made and when function levels changed.

Activation of lower function levels can be used to prevent a new function from being used, for example to allow a new function to be implemented in a staged approach or to avoid some unexpected behavior. Some new functions use application compatibility (APPLCOMPAT) levels instead of function levels to determine usage, so issuing a **REBIND** command with an earlier APPLCOMPAT value might be required to prevent new function use.

You can use the **DISPLAY GROUP** command to determine what the various function and catalog levels are. The **DISPLAY GROUP** example in Figure 2-11 on page 29 shows that the current function level is **V13R1M100*** (FL100*):

```

-DIS GROUP
DSN7100I  -DB2A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNCAT ) CATALOG LEVEL(V13R1M100)
      CURRENT FUNCTION LEVEL(V13R1M100*)
      HIGHEST ACTIVATED FUNCTION LEVEL(V13R1M500)
      HIGHEST POSSIBLE FUNCTION LEVEL(V13R1M500)
      PROTOCOL LEVEL(2)
      GROUP ATTACH NAME(DSNG)
-----
DB2      SUB      DB2      SYSTEM      IRLM
MEMBER   ID      SYS      CMDPREF     STATUS     LVL      NAME      SUBSYS     IRLMPROC
-----
DB2A     1      DB2A    -DB2A      ACTIVE     131501   UTEC5     PR21      PRLM21
DB2B     2      DB2B    -DB2B      ACTIVE     131501   UTEC5     QR21      QRLM21
-----
SCA  STRUCTURE SIZE:      12288 KB, STATUS= AC,   SCA IN USE:      6 %
LOCK1 STRUCTURE SIZE:      12288 KB
NUMBER LOCK ENTRIES:      1048576
NUMBER LIST ENTRIES:      23151, LIST ENTRIES IN USE:      7
SPT01 INLINE LENGTH:      32138
*** END DISPLAY OF GROUP(DSNCAT )
DSN9022I  -DB2A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION

```

Figure 2-11 DISPLAY GROUP example

In this case, you can tell that the previous function level of the data-sharing group was FL 500 because the value in the HIGHEST ACTIVATED FUNCTION LEVEL field is V13R1M500. If a higher function level was previously activated, you can query the SYSLEVELUPDATES catalog table to determine what the previous function level was before the return to FL 100*.

2.9.2 Activating higher function levels in Db2 13

You activate a function level that is higher than the current function level by issuing the **ACTIVATE** command. Activation of a particular function level can be done only if all prerequisites are met. Prerequisites usually consist of catalog changes, but not every function level has a catalog change prerequisite. If you want to know whether a subsystem or data-sharing group is ready for a function level, use the **ACTIVATE** command with the **TEST** option.

In Figure 2-12, the current function level is V13R1M100*, and you want to know whether you can activate FL V13R1M501.

```

-ACTIVATE FUNCTION LEVEL (V13R1M501) TEST
DSNU757I  -DB2A DSNUGCCA
*** BEGIN ACTIVATE FUNCTION LEVEL (V13R1M501)
      GROUP NOT ELIGIBLE FOR FUNCTION LEVEL (V13R1M501)
      CATALOG NOT AT REQUIRED LEVEL (V13R1M501)
      CATALOG LEVEL(V13R1M100)
      CURRENT FUNCTION LEVEL(V13R1M100*)
      HIGHEST ACTIVATED FUNCTION LEVEL(V13R1M500)
      HIGHEST POSSIBLE FUNCTION LEVEL(V13R1M500)
-----
DB2      CURRENT      CAPABLE FUNCTION LEVELS
MEMBER   ID      CODE-LEVEL  LOWEST      HIGHEST     STATUS
-----
DB2A     1      V13R1M501  V13R1M100  V13R1M501  ELIGIBLE
DB2B     2      V13R1M501  V13R1M100  V13R1M501  ELIGIBLE
-----
DSN9022I  -DB2A DSNZACMD '-ACTIVATE FUNC' NORMAL COMPLETION

```

Figure 2-12 ACTIVATE command example

The information in the example output indicates that you cannot activate V13R1M501 because the catalog level is not at the required level (V13R1M501). The fact that the current function level is V13R1M100* does not affect the eligibility of the activation. If the catalog level was CL 501, the data-sharing group would be eligible to activate FL 501, as shown in Figure 2-13.

```

-ACTIVATE FUNCTION LEVEL (V13R1M501) TEST
DSNU757I  -DB2A DSNUGCCA
*** BEGIN ACTIVATE FUNCTION LEVEL (V13R1M501)
          GROUP ELIGIBLE FOR FUNCTION LEVEL (V13R1M501)
          CATALOG LEVEL(V13R1M501)
          CURRENT FUNCTION LEVEL(V13R1M100*)
          HIGHEST ACTIVATED FUNCTION LEVEL(V13R1M500)
          HIGHEST POSSIBLE FUNCTION LEVEL(V13R1M501)
-----
DB2      CURRENT      CAPABLE FUNCTION LEVELS
MEMBER   ID    CODE-LEVEL  LOWEST      HIGHEST     STATUS
-----
DB2A     1    V13R1M501  V13R1M100  V13R1M501  ELIGIBLE
DB2B     2    V13R1M501  V13R1M100  V13R1M501  ELIGIBLE
-----
DSN9022I  -DB2A DSNZACMD '-ACTIVATE FUNC' NORMAL COMPLETION

```

Figure 2-13 Data-sharing group would be eligible to activate FL 501

2.9.3 Ensuring that the appropriate application compatibility level is set on packages

Similar to usage in Db2 12, some new functions in Db2 13 require a particular application compatibility (APPLCOMPAT) level for use in Db2 13. You must ensure that an application is bound with an APPLCOMPAT value that allows usage of the new function.

2.10 Summary

The ability to perform an online migration has been at the top of Db2 administrators' wish lists since data sharing was introduced in Db2 4. The Db2 13 migration process was redesigned and greatly improved with online migration requirements in mind. These migration improvements ensure that Db2 13 migrations can be accomplished online in data-sharing environments and improve your migration experience.



Availability and scalability

To support the high availability (HA) features that were introduced in earlier versions of Db2, Db2 13 introduces the following major HA features through new and changed commands, utilities, and DDL statements:

- ▶ Online deletion of active logs from the bootstrap data set
- ▶ Partition-by-growth to partition-by-range online conversion
- ▶ Increased concurrency for ALTER TABLE DATA CAPTURE

To support your business growth, Db2 13 enables concurrent user activities at a level higher than before through improved efficiency and scalability by using compute, memory, and other system resources.

This chapter describes the following additional topics:

- ▶ Extended common service area improvements
- ▶ Reducing agent local pool below-the-bar storage consumption
- ▶ Storage manager contraction
- ▶ Database access thread availability improvements
- ▶ Real-time statistics
- ▶ Open data set scalability
- ▶ Increased size of directory table spaces SPT01 and SYSLGRNX
- ▶ Specifying the IRLM maximum storage for locks
- ▶ Dynamic alteration of CF lock structure storage

3.1 Online deletion of active logs from the bootstrap data set

To delete active logs from the bootstrap data set (BSDS) before Db2 13, you must stop the Db2 subsystem and then run the DSNJU003 utility. This disruption impacts the Db2 ability to support continuously available application-processing environments.

Starting with function level V13R1M500, you can delete active logs from the BSDS by using the new **-SET LOG REMOVELOG** command while Db2 is running. This new command complements the existing **-SET LOG NEWLOG** command for adding active logs while Db2 remains running.

By using both online removal and addition, you can replace existing active log data sets without shutting down Db2 and incurring any application impact. This new feature helps you avoid Db2 outages and increases high availability support, particularly in the following situations:

- ▶ When the number of active log data sets for a copy reaches the maximum limit of 93, existing log data sets must be deleted before replacing them with larger log data sets.
- ▶ To take advantage of Db2 12 support for active log data sets greater than 4 GB, existing active log data sets can be replaced with active log data sets that are greater than 4 GB.
- ▶ To leverage z/OS DFSMS data set encryption for log data sets, which was introduced in Db2 11, unencrypted log data sets must be replaced with new encrypted log data sets.
- ▶ Encrypted log data sets must be redefined periodically to replace encryption keys with new keys.
- ▶ Moving active log data sets to a different storage system might require them to be deleted and added.

3.1.1 Deleting active logs

You can delete an active log from the BSDS by using one of the following methods:

- ▶ Db2 stand-alone utility DSNJU003
Stop Db2, run **DSNJU003 DELETE** to remove the active log, and restart Db2.
- ▶ Db2 **-SET LOG REMOVELOG** command
When the function level is V13R1M500 or higher, issue the **-SET LOG REMOVELOG** command to delete the active log data set.

The **-SET LOG REMOVELOG** command behaves like DSNJU003 in that it removes only the log from the BSDS. The physical data set can be deleted as a separate step after the log is removed from the BSDS.

3.1.2 Deleting active logs without a Db2 outage

Instead of using the DSNJU003 change log inventory utility to delete an active log from the BSDS without a Db2 outage, you can use the **-SET LOG REMOVELOG** command.

The **-SET LOG REMOVELOG** command generates one of the following outcomes:

- ▶ A log that is not in use is successfully deleted from the BSDS.
- ▶ A log that is in use is marked REMOVAL PENDING.
- ▶ The command fails due to an error (for example, a log was not offloaded or is active).

The following conditions are true for a log that is marked for REMOVAL PENDING:

- ▶ It cannot be unmarked.
- ▶ It cannot be used for new log read or write requests. Instead, Db2 uses another active log copy, if it exists, or an archive log to satisfy log read requests.
- ▶ You can monitor the log's status, and when it is no longer in use, you can reissue the **-SET LOG REMOVELOG** command against it.
- ▶ If Db2 is restarted in a non-data-sharing environment, it is deleted from the BSDS during the restart.
- ▶ If Db2 is restarted in a data-sharing environment, its status persists after the restart.

You can monitor the status of a log that is marked for REMOVAL PENDING by using the following methods:

- ▶ To identify log data sets that are in the REMOVAL PENDING status, you can use the **-DISPLAY LOG** command with the new **DETAIL** option. Alternatively, you can use DSNJU004 to show information about the REMOVAL PENDING status for local active log data sets.
- ▶ For data-sharing environments, you can use the command **D GRS,RES=(*,<data-set-name>)** to determine whether peer members still have the active log data set allocated or in use.

Example 3-1 - Example 3-4 on page 34 illustrate some common scenarios of online deletion of active logs in Db2 13.

Example 3-1 An inactive log is successfully deleted from the BSDS

```
-DB2A SET LOG REMOVELOG(DSNC000.DB2A.LOGCOPY1.DS03) COPY(1)
DSNJ392I  -DB2A DSNJW106 COPY1 LOG DATA SET 161
DSNC000.DB2A.LOGCOPY1.DS03 REMOVED FROM THE ACTIVE LOG INVENTORY
  STARTRBA=000000000001916B3000  ENDRBA=00000000000192A62FFF
  STARTLRSN=00DAFF75B29545E29C00  ENDLRSN=00DAFF75C04587633600
  STATUS=REUSABLE
DSN9022I  -DB2A DSNJC001 '-SET LOG' NORMAL COMPLETION
```

Example 3-2 An active log is flagged REMOVAL PENDING

```
-DB2A SET LOG REMOVELOG(DSNC000.DB2A.LOGCOPY1.DS05) COPY(1)
DSNJ393I  -DB2A DSNJW106 COPY1 LOG DATA SET 302
DSNC000.DB2A.LOGCOPY1.DS05 MARKED AS REMOVAL PENDING IN THE ACTIVE
LOG INVENTORY
  STARTRBA=00000000000000000000  ENDRBA=00000000000000000000
  STARTLRSN=00000000000000000000  ENDLRSN=00000000000000000000
  STATUS=NEW, REMOVAL PENDING
DSN9022I  -DB2A DSNJC001 '-SET LOG' NORMAL COMPLETION
```

Example 3-3 A -SET LOG REMOVELOG command error

```
-DB2A SET LOG REMOVELOG(DSNC000.DB2A.LOGCOPY1.DS02) COPY(1)
DSNJ391I  -DB2A REMOVELOG OPERATION FAILED. THE SPECIFIED ACTIVE LOG
DATA SET DSNC000.DB2A.LOGCOPY1.DS02 CANNOT BE REMOVED. REASON: NEXT
CURRENT LOG
DSN9023I  -DB2A DSNJC001 '-SET LOG' ABNORMAL COMPLETION
```

Example 3-4 Monitoring log removal pending status with DSNJU004

```
DATA SET INFORMATION
-----
DSN=DSNC000.DB2A.LOGCOPY1.DS01
STATUS=TRUNCATED, REUSABLE
DSN=DSNC000.DB2A.LOGCOPY1.DS04
STATUS=TRUNCATED, REMOVAL PENDING
DSN=DSNC000.DB2A.LOGCOPY1.DS02
STATUS=NOTREUSABLE
```

Example 3-5 shows the usage of the new DETAIL option for **-DISPLAY LOG**.

Example 3-5 -DISPLAY LOG DETAIL

```
LOG DETAIL
DSNJ370I  -DB2A DSNJC00A LOG DISPLAY 853
CURRENT COPY1 LOG = DSNC000.DB2A.LOGCOPY1.DS02 IS 96% FULL
CURRENT COPY2 LOG = DSNC000.DB2A.LOGCOPY2.DS01.NEW IS 2% FULL
          H/W RBA = 0000000000017B365ADC
          H/O RBA = 0000000000017A0A2FFF
          FULL LOGS TO OFFLOAD = 0 OF 12
          OFFLOAD TASK IS (AVAILABLE)
          SOFTWARE ACCELERATION IS DISABLED
          ZHYPERLINK WRITE IS DISABLED
DSNJ384I  -DB2A
COPY1 LOGS: TOTAL=6 NOTREUSABLE=1 REUSABLE=3  STOPPED=0
              REMOVAL PENDING=2
LOG DATA SET NAME                REMOVAL PENDING READERS
DSNC000.DB2A.LOGCOPY1.DS03        21.122 10.00.42      2
DSNC000.DB2A.LOGCOPY1.DS01        21.123 10.32.56      1
COPY2 LOGS: TOTAL=6 NOTREUSABLE=1 REUSABLE=3  STOPPED=0
              REMOVAL PENDING=2
LOG DATA SET NAME                REMOVAL PENDING READERS
DSNC000.DB2A.LOGCOPY2.DS02        21.119 09.33.02      0
DSNC000.DB2A.LOGCOPY2.DS03        21.123 12.32.27      0
DSNJ371I  -DB2A Db2 RESTARTED 10:32:28 MAY  3, 2021 869
          RESTART RBA 0000000000017B35D000
          CHECKPOINT FREQUENCY 1000 LOGRECORDS
          LAST SYSTEM CHECKPOINT TAKEN 10:32:32 MAY  3, 2021
DSN9022I  -DB2A DSNJC001 '-DIS LOG' NORMAL COMPLETION
```

Note: The **-SET LOG REMOVELOG** command is not supported in IBM GDPS® active-active environments.

3.2 Partition-by-growth to partition-by-range online conversion

Db2 13 can convert a table's partitioning scheme from partition-by-growth (PBG) to partition-by-range (PBR) with minimal application impact. You can use an **ALTER TABLE** statement with the new **ALTER PARTITIONING TO PARTITION BY** clause to accomplish the conversion.

If the data sets of the table space for the altered table are already defined, the conversion results in a pending change that is materialized by a subsequent **REORG**. Otherwise, the conversion is immediate. The containing table space is converted to use relative page numbers (RPNs). The conversion process handles any existing indexes on the table; however, Db2 does not change any aspects or attributes of those indexes. Therefore, consider creating partitioned indexes on the table after completing the conversion.

3.2.1 Considerations for PBG to PBR conversion

PBG and PBR universal table spaces (UTS) are the strategic table space types for Db2 tables. PBG table spaces are the default UTS type, and all online conversions from simple or segmented table spaces to UTS result in PBG table spaces. PBG table spaces are well suited for small and medium-sized tables. However, when a table in a PBG table space grows in size, the following drawbacks can occur:

- ▶ Insert or query performance degradation
- ▶ Difficulty with regaining clustering of the data (requires a **REORG** of the entire table space)
- ▶ Difficulty with managing very large non-partitioned indexes
- ▶ Lack of partition parallelism support for utilities
- ▶ Limited support for partition-level utility operations

If you encounter these issues for tables in PBG table spaces, consider converting them to PBR.

3.2.2 Requirements for converting tables from PBG to PBR

Starting with application compatibility (APPLCOMPAT) level V13R1M500, you can convert a table from PBG to PBR if the following conditions are met:

- ▶ The high limit key for the last partition requires **MAXVALUE** for ascending key columns or **MINVALUE** for descending key columns to avoid data loss during the subsequent online **REORG** execution.
- ▶ If the table that will be converted is defined with **DATA CAPTURE CHANGES** and the data sets of the table space are defined, the number of partitions for the range-partitioning scheme must not be less than the maximum number of partitions for the PBG table space. This requirement eliminates a situation in which Instrumentation Facility Component Identifier (IFCID) 0306 cannot decompress log records that are associated with deleted partitions because the compression dictionaries no longer exist.
- ▶ To support PBR RPN, the data sets for the table space and indexes must be associated with a DFSMS data class that is specified with extended format and addressability.
- ▶ The table to be altered must meet the following conditions:
 - It must exist at the current server.
 - It must be in a complete state if the alteration is a pending change.
 - It cannot be a catalog or directory table.

- It cannot be a created global temporary table.
- It cannot have table OBID = 1.
- It cannot be involved in a clone relationship.
- It cannot be defined with **ORGANIZE BY HASH**.
- It cannot be an accelerated table.
- It cannot contain an XML column.
- It cannot contain a large object (LOB) column.
- ▶ The containing table space of the altered table must meet the following conditions:
 - It must be a PBG table space.
 - It cannot have unmaterialized pending changes.

For more information, see [Db2 SQL - ALTER TABLE](#).

3.2.3 Converting a table from a PBG to PBR partitioning scheme

To convert an existing table in a PBG table space to use range-based partitions, complete the following steps:

1. Determine whether the related table space and indexes are in an LPL, WEPR, or GRECP restricted state. If so, you must resolve the issues before proceeding.

You can issue the following **DISPLAY DATABASE** command with the **RESTRICT** keyword to display objects that are in these states:

```
-DISPLAY DATABASE(database-name) SPACENAM(*) RESTRICT(GRECP,LPL,WEPR)
```

2. Plan for any other required schema alterations.

After you complete step 5 on page 37 and until you materialize the change, all pending changes and certain immediate changes are restricted if you reference the table space or any objects within the table space. Also, if any schema definition changes are pending for the table space or any object within the table space, the partitioning scheme alteration is restricted.

For more information, see [Restrictions for pending data definition changes](#).

3. Determine a suitable partitioning scheme.

Consider the partitioning scheme to use for the table, including the columns that define the partitions and the limit key values for each partition. The following considerations affect the partitioning scheme that you use:

- The number of partitions that will be created (especially if the table that will be converted is defined with **DATA CAPTURE CHANGES** and subscribed to data replication).
- The data set size for each partition. The maximum size of the data set for the PBR partition is inherited from the original PBG partition, so you must choose limit keys for partitions to ensure that the data ranges can fit in the corresponding partitions.
- The distribution of data across the partitions.

You can use the **RUNSTATS** utility to collect useful statistics for planning the range-partitioning scheme.

4. Prepare the table for data replication.

If the table that will be converted is defined with **DATA CAPTURE CHANGES** and subscribed to data replication, the number of partitions for the PBR partitioning scheme must not be less than the maximum number of partitions of the PBG table space. Therefore, consider taking one of following actions before starting the conversion:

- Use an **ALTER TABLE** statement with the **DATA CAPTURE NONE** clause so that the number of PBR partitions is no longer enforced as equal or greater than the maximum number of partitions for the PBG.
- If the maximum number of partitions is excessively high, use an **ALTER TABLESPACE** statement with the **MAXPARTITIONS** clause to lower the maximum number of partitions. If the PBG table space has empty partitions, consider running the **REORG** utility for the PBG table space. Specify the **DROP_PART YES** keyword (or enable the **REORG_DROP_PBG_PARTS** subsystem parameter) to drop empty partitions and specify the **MAXPARTITIONS** clause with a suitable value.

5. Issue an **ALTER TABLE** statement.

Use an **ALTER TABLE** statement that specifies an **ALTER PARTITIONING TO PARTITION BY RANGE** clause. For example, the following statement converts the existing table TB01 to use range-based partitions and converts the containing table space to a PBR table space with relative page numbers:

```
ALTER TABLE TB01 ALTER PARTITIONING TO
PARTITION BY RANGE (ACCT_NUM)
(PARTITION 1 ENDING AT (199),
PARTITION 2 ENDING AT (299),
PARTITION 3 ENDING AT (399),
PARTITION 4 ENDING AT (MAXVALUE))
```

This statement is a pending definition change if the data sets of the table space are already created. If the data sets do not exist, the **ALTER TABLE** statement is an immediate definition change, and no further action is necessary.

6. If the alteration is a pending change, run the following utilities:

- Optionally, run the **REPORT RECOVERY** utility on the table space and keep the report for future reference if you must access data from image copies that are taken before the conversion is materialized.
- Run the **REORG TABLESPACE** utility with the **SHRLEVEL REFERENCE** or **CHANGE** clause against the entire PBG table space to materialize the pending change.

7. Optionally, consider creating partitioned indexes on the table to support query parallelism.

3.3 Increased concurrency for ALTER TABLE DATA CAPTURE

Many organizations use data replication tools to replicate data from a source table in Db2 to a target table. Data replication tools that rely on log-capture replication methods require log records to contain a full “before” image of a row for each update.

In Db2, the **DATA CAPTURE** attribute of a table dictates whether a full or partial before image is written out for updates to the table’s rows. By default, when a table is created, its **DATA CAPTURE** attribute is defined as **NONE**, which means that Db2 writes out only partial before images in log records for updates to the table’s rows.

To enable data replication, set the **DATA CAPTURE** attribute to **CHANGES** so that Db2 writes out full before images in log records (full log records) instead of partial ones (partial log records). Then, data replication tools can consume these full log records for replication processing. The “*after*” image of updated rows is always partial in the log record and does not depend on the **DATA CAPTURE** attribute.

If a table is not created with the **DATA CAPTURE CHANGES** attribute, you must alter the table by issuing an **ALTER TABLE** statement with the **DATA CAPTURE CHANGES** clause to enable the writing of full log records.

Before version 13, Db2 performs the following quiescing as part of the **DATA CAPTURE** alteration processing:

- ▶ Quiesces static packages that depend on the altered table.
- ▶ Quiesces cached dynamic statements that depend on the altered table.

In HA environments, making time available to run the alteration can be challenging. Repeated executions of **ALTER TABLE DATA CAPTURE** might time out and fail due to continuous concurrent activities on the table. As a result, you might not be able to enable or disable data replication in a timely manner, or you might need to incur an application outage to complete the alteration.

3.3.1 ALTER TABLE DATA CAPTURE concurrency enhancement

In Db2 13, **DATA CAPTURE** alteration is enhanced so that it no longer waits for dependent Data Manipulation Language (DML) statements to commit. The operation can be run successfully even when static or dynamic DML statements are running concurrently against the table.

The **ALTER TABLE DATA CAPTURE** enhancement provides the following benefits:

- ▶ Static packages that depend on the altered table are no longer quiesced.
- ▶ Cached dynamic statements that depend on the altered table are no longer quiesced.

The APPLCOMPAT level must be at V13R1M500 or later for the **ALTER TABLE** enhancement to be effective.

3.3.2 Visibility of data capture alteration to other concurrent activities before commit

An alteration of the **DATA CAPTURE** attribute results in an update to the catalog (SYSTABLES.DATACAPTURE field) and internal object descriptor (OBD) of a table, indicating the new attribute value.

Without the concurrency enhancement, OBD updates are visible only to other concurrent activities after the alteration is committed. For DML statements that are run in the same transaction (unit of work), all log records for the statements are either full or partial log records, and a mixture of full and partial log records for DML statements within the same transaction is not possible.

With the concurrency enhancement and the removal of quiescing on a **DATA CAPTURE** alteration, OBD updates are now visible to concurrently running DML statements on the same Db2 member even before the alteration is committed. As a result, when the OBD is updated (and before the **ALTER** commits), DML against the table on the same Db2 member starts writing out log records in the format that is indicated by the new **DATA CAPTURE** value in the OBD. Then, these concurrent DML statements in the same unit of work might now write out a mixture of partial and full log records because the timing of the change in log record format occurs earlier.

The timing of when catalog updates are visible to other threads is unchanged. This aspect depends on a reader's isolation level.

Another significant aspect that is associated with the concurrency enhancement is related to the rollback behavior of the **DATA CAPTURE** alteration. Rollback can occur if the **ALTER** statement encounters an error and fails, or if a **ROLLBACK** statement is issued for the **ALTER** transaction. If the **DATA CAPTURE** alteration is rolled back, the OBD update is also rolled back and immediately visible to concurrent DMLs against the table on the same Db2 member.

3.3.3 Considerations for data replication and other tools that read Db2 log records

Introducing a mixture of partial and full log records for DMLs in a concurrent transaction might impact data replication products or other tools that consume Db2 log records.

To prevent this potential impact, adhere to the following processes for enabling and disabling replication. Otherwise, assess whether the replication tools that you use are impacted and prepare for the changes.

To enable data replication:

1. Run **ALTER TABLE DATA CAPTURE CHANGES** on the source table.
2. Activate replication of the source table to target table.

The data replication tool checks Db2 catalog column `SYSTABLES.DATACAPTURE` to see whether data capture is enabled. If not, the activation fails; otherwise, the tool starts consuming the full log records of the source table through IFCID 0306 or some other method. Now, the **ALTER** is committed, and all log records that the data replication tool sees contain the full before image.

To disable data replication:

1. Deactivate the replication of the source table to target table.

The data replication tool stops consuming Db2 log records for the source table.

2. Run **ALTER TABLE DATA CAPTURE NONE** on the source table.

Now, the data replication tool is not reading the Db2 log records anymore, so any mixture of partial and full log records is no longer relevant, and the data replication tool is not impacted.

If this prescribed order of operations is followed, the changes that are introduced by this concurrency enhancement have no impact on data replication tools.

For tools that consume Db2 log records for purposes other than data replication, you must assess the potential impact.

3.4 Extended common service area improvements

Db2 load modules and control blocks are built to reside in an extended common service area (ECSA) to easily move between address spaces. To free the Db2 32-bit footprint in ECSA, Db2 13 moved some storage from the following pools in ECSA below-the-bar (BTB) to either 64-bit HVCOMMON above-the-bar (ATB) or private Db2 system address space BTB:

- ▶ POOL RMID=026 IFCSTATS
- ▶ POOL RMID=026 IFCMISC

As a result, you might see less ECSA consumption by Db2 13 on instrumentation-related processes, such as:

- ▶ **START TRACE**, **-STOP TRACE**, **-DISPLAY TRACE**, and **-MODIFY TRACE** commands, especially with the **AUDIT POLICY** option
- ▶ Writing Db2 statistic trace records, such as IFCIDs 1, 2, 124, 199, 202, 230, and 254
- ▶ Monitoring log records through IFCID 306
- ▶ IFI READS requests
- ▶ Selective dump

Additionally, as a result of agent storage improvement and Database Access Thread (DBAT) availability improvement in Db2 13, the amount of ECSA storage that is required for running distributed applications changed.

3.4.1 Calculating the ECSA storage for the Distributed Data Facility component

To determine the amount of required ECSA storage for Distributed Data Facility (DDF) processing, you must first determine the network protocol that your Db2 for z/OS server supports. Db2 for z/OS supports two types of network protocols: TCP/IP and SNA.

TCP/IP connections

If you operate with z/OS 2.2 or later, TCP/IP connections use zero bytes of ECSA storage.

If you operate with z/OS 2.1 or earlier, add 1 KB for each TCP/IP connection up to the maximum limit of your MAX REMOTE CONNECTED field (**CONDBAT** subsystem parameter). For example, if **CONDBAT** is set to 10000, set aside 10,000 KB (1 * 10000) of ECSA storage for TCP/IP connections.

SNA connections

Allow 1 KB per SNA connection up to the maximum limit of your MAX REMOTE CONNECTED field (**CONDBAT** subsystem parameter). For example, if **CONDBAT** is set to 10000, set aside 10,000 KB (1 * 10000) of ECSA storage for SNA connections.

After you calculate the amount of ECSA storage for DDF processing according to the type of network protocols that your Db2 server supports, for the ECSA storage requirements, add 1 KB for each server thread (DBAT) to obtain the total.

As a best practice, you can use up to two times of the value for the MAX REMOTE ACTIVE field (MAXDBAT subsystem parameter). Add 1 KB for each Db2 site in your network, regardless of the number of connections between the site and the Db2 subsystem where the site refers, to the following types of adjacent partners:

- ▶ Downstream servers that are listed in the communication database (CDB) of your Db2 subsystem (as a TCP/IP address or an SNA LU name).
- ▶ Upstream clients for which this Db2 subsystem acts as the DRDA server.

Example 3-6 shows an example of calculating the ECSA storage requirements for DDF processing.

Example 3-6 Stand-alone Db2 for z/OS server that uses only TCP/IP connections to service remote clients and access other remote sites

```
OS level = z/OS 2.4
Db2 SSID = DB2A
MAXDBAT = 500
CONDBAT = 10000
SYSIBM.IPNAMES
IPADDR = 198.168.1.100
IPADDR = 198.168.1.200
Total ECSA storage requirement =
2.5 MB + (1KB * 500 MAXDBAT) + (1KB * 2 downstream sites) = 3,135,448 bytes
```

3.5 Reducing agent local pool below-the-bar storage consumption

Storage constraints often limit Db2 subsystems to less than 10,000 concurrent threads today. A significant contributor to the constraints is the amount of agent local BTB storage that is consumed per thread. One of the largest remaining users of agent local BTB storage is the SQL statement text that is used for **PREPARE** and **EXECUTE IMMEDIATE**. Although most Db2 agent local storage is already acquired ATB, the requirement for more concurrent threads necessitates the moving of more agent local BTB storage to ATB.

Before Db2 13, dynamic SQL can be a heavy user of agent local BTB storage. The SQL statement text can be as large as 2 MB, and multiple dynamic SQL statements can be nested due to triggers, UDFs, and stored procedures. Db2 keeps a copy of the dynamic SQL input statement text and the attribute string in agent local BTB storage during the **PREPARE** and **EXECUTE IMMEDIATE** executions.

In Db2 13, the statement text and attribute string for **PREPARE** and **EXECUTE IMMEDIATE** are stored in agent local ATB storage instead. The benefit of this enhancement can be greater than anticipated because of the way that Db2 handles dynamic SQL in native SQL routines.

As an example, assume the following scenario:

- ▶ Table T1 has an after-insert trigger that calls the native SQL stored procedure SP.
- ▶ Stored procedure SP issues dynamic SQL:

```
UPDATE T2 SET C1=C1+1
```

When the application runs a simple statement like **INSERT INTO T1 VALUES(1)**, the trigger calls the stored procedure and runs an **UPDATE** to T2.

Assume that the application input host variable and the stored procedure SQL variable that hold the SQL statement text are both defined as CLOB(2M).

Before Db2 13, the initial application call to Db2 to prepare the **INSERT** statement acquires enough agent local pool BTB storage to fit the SQL text length plus other control structures. When **INSERT** runs, the trigger invokes the native SQL stored procedure. When this stored procedure prepares **UPDATE**, Db2 allocates agent local BTB storage based on the defined length of the input SQL variable, which is 2 MB.

In Db2 13, the **PREPARE** statement for both the application and the stored procedure allocates agent local ATB storage instead of BTB, and the amount of storage that is allocated for the SQL statement text is based on the length.

In addition, Db2 continues to optimize the interface between the DBM1 and DIST address spaces. Some control blocks that are used during SQL execution, which previously were in agent local pool BTB storage in DIST, are now allocated in shared ATB storage. These changes not only reduce the BTB storage consumption, but also improve performance by avoiding cross-memory copy operations of the data.

3.6 Storage manager contraction

In addition to moving more thread storage ATB, Db2 13 introduces smarter, controlled BTB and ECSA contraction to prevent fragmentation and decrease the likelihood that Db2 hits storage limitations, which might result in a system impact. An increase in the number of threads also impacts the releasing of storage above the 2G bar (ATB). In Db2 12, excessive discard requests can overwhelm the z/OS LPAR and might result in a significant increase in Db2 system service address space (SRB) CPU time and possibly RSMQ spin lock contention. In some cases, excessive discard requests can also cause a z/OS common storage shortage (ESQA). To alleviate the constant necessity of discard requests, Db2 13 changes how it manages storage above the 2G bar.

3.6.1 Smarter contraction of DBM1 below-the-bar storage

Db2 12 deprecated a subsystem parameter, **CONSTOR**, and removed the frequent contraction of threads and working pools for DBM1 BTB to improve performance.

Instead of requiring manual assessment and setting of **CONSTOR** as an on/off switch, Db2 13 introduces a smarter storage manager BTB and ECSA contraction scheme that detects the best time to contract precious storage based on workload consumption that exceeds normal levels. This capability allows you to create workloads or grow existing ones by supporting more concurrent threads.

With the removal of subsystem parameter **CONSTOR**, the IFCID 001 serviceability field **QSST_CONSTOR_NUM** in **DSNDQSST** is no longer relevant. This field is renamed to **QSST_CNTRCT_NUM** to represent the number of 31-bit agent local pools that are contracted by storage manager.

3.6.2 BTB contraction

Db2 13 adds logic to a Db2 daemon that periodically checks BTB storage consumption. If the Db2 BTB storage consumption exceeds the acceptable threshold of 64%, contraction of agent local pools is triggered.

After Db2 BTB storage consumption falls below 64%, contraction of agent pools is no longer triggered.

3.6.3 ECSA contraction

The Db2 daemon also monitors storage consumption in the ECSA. If ECSA storage consumption exceeds 85%, contraction is triggered by waking up the Db2 Storage Manager contraction engine, which results in the contraction of Db2 pools that are allocated in the ECSA.

After storage consumption in the ECSA falls below the 85% threshold, contraction is no longer triggered.

3.6.4 Memory object contraction

The Db2 12 subsystem parameter **REAL_STORAGEMANAGEMENT** determines how Db2 manages real frames that are consumed above the 2G bar. Thread storage is released at thread deallocation or at certain **COMMIT** intervals (every 120 commits). For more information, see [Db2 Real Storage Management and Recommendations](#).

Removing the subsystem parameter REALSTORAGE_MANAGEMENT

Db2 13 enhances storage management so that the real storage contraction is no longer driven by subsystem parameter **REALSTORAGE_MANAGEMENT**. This change reduces the impact in managing thread storage and avoids the z/OS level contentions that are observed when Db2 invokes excessive IARV64 REQUEST(DISCARDDATA) requests to return real storage frames to z/OS.

Trigger for memory object contraction

Db2 13 does not issue the IARV64 REQUEST(DISCARRDATA) request during thread deallocation or during certain **COMMIT** intervals. Instead, Db2 returns the memory that is used by threads to the memory object without immediate DISCARDDATA processing. A system-level timer drives the memory object contraction to release unused frames back to z/OS.

In addition, Db2 13 adds logic to a Db2 daemon to monitor storage consumption above the 2G bar. If the number of available free frames falls below the calculated threshold that is provided by z/OS, memory object contraction is triggered.

3.7 Database access thread availability improvements

To accommodate workload growth and application modernization, you might need to increase the level of Db2 DDF DBAT concurrency within Db2. Db2 13 delivers a gradual DBAT contraction enhancement that improves Db2 DBAT availability to support this continued growth of DDF workloads.

3.7.1 Gradual DBAT contraction

A system can experience a major performance impact during Real Storage Manager (RSM) processing to “discard” the storage frames that back up the Db2 ATB storage as pages within that storage are marked as freed. The more concurrent the requests to release ATB storage are, the greater the impact on system performance. When Db2 threads (DBATs included) terminate, the Db2 ATB storage pools that are related to those threads are freed. Because a DBAT uses numerous ATB thread storage pools, if many concurrent DBAT terminations occur, the RSM processing in which real memory frames are discarded can be detrimental to the performance of the system where Db2 is running. Although the Db2 storage manager addresses this storage pool contraction and discards issue in a separate Db2 13 enhancement, having DDF make changes to reduce the DBAT creation and termination thrashing in some workloads can still provide some benefit.

Common causes of many concurrent DBAT terminations include a short-term spike in new work and new connections that are created to Db2. One example of a new connection spike is when an application or application server starts and immediately creates and populates a local connection pool with many connections. A quick spike in new connections results in a corresponding spike in the number of DBATs that are needed to service the work. When the new connection spike is over, the additional DBATs that might have been created to service the spike in new connection requests can end with similar expiration times that are specified by the **POOL THREAD TIMEOUT (POOLINAC)** subsystem parameter. The Db2 DDF error monitor service, which is responsible for detecting and terminating DBATs that exceed their expiration time, currently runs by using a 2-minute interval cycle. A DBAT concurrent termination spike can result when many DBATs are detected as expired within a single 2-minute DDF error monitor cycle.

Most of the changes that are described in this section are applicable only when Db2 DDF inactive thread support is enabled by setting the **DDF THREADS (CMTSTAT)** subsystem parameter to **INACTIVE**.

3.7.2 Overview of Db2 DDF DBAT usage

The following overview explains the new DBAT termination behavior changes, the requests for which a DDF DBAT is used, and the different DBAT categories or types. A DBAT is required to handle the following requests from a distributed client:

- ▶ Accept a new connection.
- ▶ Process a transaction request.
- ▶ Terminate the connection.

When a DBAT is not processing a client request, it does not need to be attached to the client connection that is awaiting the next request from a client. This capability allows Db2 to service many more client connections with a smaller population of DBAT server threads. The focus of the gradual DBAT contraction enhancement is related to the disposition and treatment of DBATs that are no longer actively processing a client request.

The three general categories of Db2 DBATs that are affected by the changes in this enhancement are described in the following sections:

- ▶ Normal disconnected pooled DBATs
- ▶ KeepDynamicRefresh DBATs
- ▶ High-performance DBATs

Normal disconnected pooled DBATs

When a DBAT is not actively servicing a connection request, the DBAT is pooled and considered disconnected from a connection. If a pooled thread is not used within the amount of time that is indicated by the **POOLINAC** subsystem parameter, it is terminated and then possibly replaced. A DBAT is also terminated after processing a new connection request, a transaction request, or a terminate connection request, if it is used cleanly (no resources held past commit) to process up to 200 requests. The name of this capability is *inactive connection processing*, and it is activated when the **CMTSTAT** subsystem parameter is set to INACTIVE.

Setting the **POOLINAC** subsystem parameter to 0 prevents Db2 from terminating pooled DBATs, but it does not prevent Db2 from terminating a DBAT that was used for over 200 requests.

KeepDynamicRefresh DBATs

For some distributed application environments, most notably the ones that use Db2 packages that are bound with the **KEEPDYNAMIC** option, Db2 does extra processing beyond the basic inactive connection processing to maintain performance. With the **KEEPDYNAMIC(YES)** option, a dynamic SQL statement can be prepared, run, and rerun in the next unit of work without being reprepared.

For long-running applications that use **KEEPDYNAMIC** packages, Db2 periodically recycles the DBAT (and associated connection) to clean up persistent storage and package resources that are tied to the DBAT due to the **KEEPDYNAMIC** usage. The Db2 function that allows a DBAT with **KEEPDYNAMIC** packages to periodically recycle is called KeepDynamicRefresh (KDR) processing. KDR processing involves changes in both the client drivers and Db2. When a client connection indicates to Db2 that it both supports KDR processing and is performing either sysplex workload balancing or seamless failover, the Db2 server takes one of the following actions:

- ▶ At a clean transaction boundary, if the KDR DBAT has existed for over 1 hour, the entire DBAT and connection are terminated.
- ▶ At a clean transaction boundary, if the KDR DBAT has not been used for a new transaction request for over 20 minutes, the entire DBAT and connection are terminated.

When the client driver detects a KDR connection termination, the driver obtains a new connection transport either to the same member (seamless failover) or to a different member (Sysplex WLB), and all the current statements that are related to the application connection are marked as needing to be reprepared.

High-performance DBATs

Although KDR provides significant performance benefits for certain applications, it is not a suitable solution for most applications. Therefore, an alternative approach for improving performance was created that is called *high-performance (HIPERF) DBATs*. You can use the following methods to activate HIPERF DBAT processing:

- ▶ By issuing the **-MODIFY DDF PKGREL(BNDOPT|BNDPOOL)** command
- ▶ For packages that are used by an application connection, by binding at least one of the packages with the **RELEASE(DEALLOCATE)** option

Normally, any package that is used by a distributed application connection is always acquired as a **RELEASE COMMIT** package regardless of the package's bound **RELEASE** setting. However, after the DDF **PKGREL** option is set to **BNDOPT** (or **BNDPOOL**), any package that is used by a distributed application connection (not only the first package) is acquired based on its bound **RELEASE** setting. After the DBAT is used for processing a new connection request, a transaction request, or a terminate connection request, and a **RELEASE DEALLOCATE** package is used (not only the current package), the DBAT is considered a **HIPERF DBAT** and it is *not* disconnected from its client connection.

When the DBAT is used for up to 200 requests from its client connection, it is disconnected and terminated like a pooled DBAT. Also, if the DBAT is not used for the amount of time that is specified on the **POOLINAC** subsystem parameter since the last request it processed, the DBAT is disconnected and terminated. A subsystem parameter **POOLINAC** value of 0 is not acknowledged. Instead, when a value of 0 is detected on the **POOLINAC** subsystem parameter, **HIPERF DBATs** are terminated after 120 seconds of not being used.

3.7.3 Db2 DBAT termination behavior

DBAT creation and termination processing can significantly impact system resources. As the level of DBAT concurrency is increased to handle the increased processing of existing and new application workloads, DBAT creation and termination processing impact how Db2 can scale to handle the increased level of DBAT concurrency. Because Db2 cannot predict the arrival rate of application requests and any associated increase in DBAT concurrency, Db2 can focus only on how frequently DBATs are terminated or contracted, most notably when many of the three types of DBATs are terminated concurrently.

The changes that are introduced in Db2 13 to improve DBAT termination behavior have two main objectives:

- ▶ To reduce the overall frequency and number of DBAT terminations
- ▶ To reduce the number of concurrent DBAT terminations that are caused by a short-term DBAT usage spike

Reducing the overall frequency and number of DBAT terminations

The first behavior change reduces the overall frequency and number of DBAT terminations by allowing a DBAT to be used more times before it is terminated. The DDF DBAT reuse logic is changed so that a pooled, disconnected, or **HIPERF DBAT** is terminated after a maximum number of 500 "transactional" uses (before this change, the maximum was 200 transactional uses).

Reducing the number of concurrent DBAT terminations that are caused by DBAT usage spikes

The second behavior change reduces the number of concurrent DBAT terminations that can occur due to a short-term DBAT usage spike:

- ▶ The DBAT timer expiration logic is modified to include a pseudo-random amount of extra time. The extra time helps to stagger the termination of DBATs that were created to service a short-term usage spike.
- ▶ The Db2 DDF error monitor now runs more frequently than every 2 minutes. By increasing the error monitor run frequency, fewer DBATs are expected to be concurrently terminated per error monitor cycle.

DBAT termination behavior before Db2 13

The following list describes DBAT termination behavior before Db2 13:

- ▶ Normal disconnected pooled DBATs are terminated in the following situations:
 - When a “clean” DBAT (no resources held past commit) at a commit point is used 200 times.
 - When a pooled DBAT is not used for the amount of time that is indicated by the **POOLINAC** subsystem parameter.

Note: Db2 12 APAR PH36114 made a change that limits the maximum number of pooled and disconnected DBAT terminations to 50 DBATs for each error monitor cycle.

- Other situations, such as DDF **SUSPEND** or **RESUME**, can cause normal disconnected pooled DBATs to terminate.
- ▶ HIPERF DBATs are terminated in the following situations:
 - When a clean DBAT at a commit point with release deallocate packages is used 200 times by its client connection.
 - When its client connection terminates (**BNDOPT**).
 - When a DBAT does not receive a new transaction request within the amount of time that is specified by the **POOLINAC** subsystem parameter.

Note: HIPERF DBATs do not acknowledge a time of 0 in the **POOLINAC** subsystem parameter. A time of 120 seconds is used instead of 0.

- ▶ KDR DBATs are terminated in the following situations:
 - When a clean DBAT at a commit point is used for an hour or longer.
 - When a KDR DBAT does receive a new transaction request within 20 minutes.

Before Db2 13, the rationale for terminating DBATs after 200 transactional uses was based on product experiences where thread storage was acquired within the DBM1 address space below the bar. Because that storage area was limited and could easily be fragmented, terminating DBATs after 200 uses was considered an optimal solution to this issue. In Db2 13, most thread storage is ATB and is shared, which means that terminating DBATs after 200 uses might be detrimental to performance in some cases.

DBAT termination behavior in Db2 13

The following list describes the DBAT termination behavior in Db2 13, which is based on the objectives of reducing the frequency and number of DBAT terminations and reducing the number of concurrent DBAT terminations that are caused by a short-term DBAT usage spike:

- ▶ Normal disconnected pooled DBATs and HIPERF DBATs are candidates for termination if one or more of the following conditions are true:
 - When a clean DBAT at a commit point is used more than 500 times.
 - Db2 detects BTB or ECSA excessive storage usage.
 - When the **POOLINAC** subsystem parameter is not 0.

For a pooled, disconnected, or HIPERF DBAT, the end time for its period of inactivity is set to the sum of the current time of DBAT suspension, the current value of the **POOLINAC** subsystem parameter, and a random amount of extra time between 0 and the **POOLINAC** value.

- When **POOLINAC** is set to 0.

A pooled disconnected DBAT is not terminated. If you need Db2 to terminate these pooled disconnected DBATs, you can make an online change to the **POOLINAC** subsystem parameter to set it to a nonzero value.

HIPERF DBATs do not acknowledge a **POOLINAC** value of 0, and instead use a default **POOLINAC** value of 2 minutes. Therefore, when the **POOLINAC** value is 0, the end time for inactivity for HIPERF DBATs is set to the current time of DBAT suspension plus the sum of the 2 minutes and a random amount of extra time of 0 - 2 minutes.

- HIPERF DBATs continue to be terminated when their client connection terminates (when using **DDF PKGREL(BNDOPT)**).
- Processing that abended always results in DBAT termination.
- ▶ Every new KDR DBAT undergoes a randomization of their time to terminate. A KDR DBAT is a candidate for termination if one of the following conditions is true:
 - It exceeded its maximum use time, which is established as 1 hour plus a random amount of time of 0 - 1 hour.
 - It exceeded its maximum idle time waiting for the next transaction request, which is set to 20 minutes plus a random amount of time of 0 - 20 minutes.
- ▶ The DDF error monitor now runs more often than its default 2-minute cycle, and during the more frequent invocations focuses only on the following types of terminating DBATs:
 - Pooled disconnected DBATs that exceed their specified **POOLINAC** time are now checked every 30 seconds.

The Db2 V12 PH36114 change that limits the maximum number of pooled disconnected DBAT terminations to 50 DBATs for each error monitor cycle is also included in Db2 13.
 - HIPERF DBATs that exceed their specified **POOLINAC** time are checked every 30 seconds.
 - KDR DBATs that exceed their maximum idle time are checked every 30 seconds.

These changes take effect only when the **CMTSTAT** subsystem parameter is set to **INACTIVE**.

3.7.4 Db2 global DDF activity statistics reporting changes

To facilitate the understanding of when DBATs are terminated and also created, the Global DDF Activity statistics, which are mapped by **DSNDQDST**, were updated with the following new and changed counters:

0001 QDSTNDBA	The number of times that a DBAT was created.
0001 QDSTNAKD	The current number of DBATs that are active due to the usage of packages that are bound with KEEPDYNAMIC YES .
0001 QDSTMAKD	The maximum number of DBATs that are active due to the usage of packages that are bound with KEEPDYNAMIC YES .
0001 QDSTNDBT	The number of times that a DBAT was terminated since DDF started.

In Db2 13, the following changes were made:

- ▶ QDSTNDBA: The DBAT creation statistics counter now always count the number of DBATs that are created.
- ▶ QDSTNAKD: The current number of DBATs that are active due to the usage of packages that are bound with **KEEPDYNAMIC YES**.
- ▶ QDSTMAKD: The maximum number of DBATs that are active due to the usage of packages that are bound with **KEEPDYNAMIC YES**.
- ▶ QDSTNDBT: The number of times that a DBAT was terminated since DDF started.
- ▶ A new pair of statistics counters records the current and maximum number of KDR DBATs.
- ▶ A new statistics counter records the number of DBAT terminations.

3.8 Real-time statistics

To accommodate larger data volumes, Db2 13 expands many columns to larger data types in the real-time statistics (RTS) SYSIBM.SYSTABLESPACESTATS and SYSIBM.SYSINDEXSPACESTATS catalog tables and associated SYSIBM.SYSTABSPACESTATS_H and SYSIBM.SYSIXSPACESTATS_H history tables. The in-memory storage for RTS is also expanded.

Table 3-1 and Table 3-2 on page 50 show a summary of the expanded columns. To improve availability, the LOCKMAX of RTS table spaces SYSTSTSS and SYSTSIS and the associated history table spaces SYSTSTSH and SYSTSISH are changed from -1 (LOCKMAX SYSTEM is in effect) to 0 to avoid lock escalation during RTS externalization.

Table 3-1 Summary of expanded columns in SYSTABLESPACESTATS and SYSTABSPACESTATS_H

Column name	Data type	Description
EXTENTS	INTEGER	The number of extents in the table space. For multi-piece table spaces, this value is the number of extents for the last data set. For a data set that is striped across multiple volumes, the value is the number of logical extents. A null value indicates that the number of extents is unknown.
REORGINSERTS	BIGINT	The number of rows or LOBs that were inserted into the table space or partition or loaded into the table space or partition by using the LOAD utility without the REPLACE option since the last time the REORG or LOAD REPLACE utilities ran or since the object was created. A null value indicates that the number of inserted rows or LOBs is unknown.
REORGDELETES	BIGINT	The number of rows or LOBs that were deleted from the table space or partition since the last time the REORG or LOAD REPLACE utilities ran or since the object was created. A null value indicates that the number of deleted rows or LOBs is unknown.

Column name	Data type	Description
REORGUPDATES	BIGINT	The number of rows that were updated in the table space or partition since the last time the REORG or LOAD REPLACE utilities ran or since the object was created. A null value indicates that the number of updated rows is unknown.
STATSINSERTS	BIGINT	The number of rows or LOBs that were inserted into the table space or partition or loaded into the table space or partition by using the LOAD utility without the REPLACE option since the last time that the RUNSTATS utility ran or since the object was created. A null value indicates that the number of inserted rows or LOBs is unknown.
STATSDELETES	BIGINT	The number of rows or LOBs that were deleted from the table space or partition since the last time that the RUNSTATS utility ran or since the object was created. A null value indicates that the number of deleted rows or LOBs is unknown.
STATSUPDATES	BIGINT	The number of rows that were updated in the table space or partition since the last time that the RUNSTATS utility ran or since the object was created. A null value indicates that the number of updated rows is unknown.
COPYCHANGES	BIGINT	If the COPY utility ran with a SHRLEVEL value other than CHANGE , this value is the number of INSERT , UPDATE , and DELETE operations, or the number of rows loaded, since the last time that the COPY utility ran. If the COPY utility was run with SHRLEVEL CHANGE , this value is the total number of INSERT , UPDATE , and DELETE operations, or the number of rows loaded, while the last COPY utility ran and since the last time that the COPY utility ran. This value does not include operations that result in no change to the data, such as an update that sets the value of a column to its existing value. A null value indicates that the number of INSERT , UPDATE , and DELETE operations or the number of rows loaded is unknown.

Table 3-2 Summary of expanded columns in *SYSINDEXSPACESTATS* and *SYSIXSPACESTATS_H*

Column name	Data type	Description
SPACE	BIGINT	The amount of space, in kilobytes, that is allocated to the index space or partition. For multi-piece, linear page sets, this value is the amount of space in all data sets. A null value indicates that the amount of space is unknown.
EXTENTS	INTEGER	The number of extents in the index space or partition. For multi-piece index spaces, this value is the number of extents for the last data sets. For a data set that is striped across multiple volumes, the value is the number of logical extents. A null value indicates that the number of extents is unknown.

Column name	Data type	Description
REORGINSERTS	BIGINT	The number of index entries that were inserted into the index space or partition since the last time the REORG , REBUILD INDEX , or LOAD REPLACE utilities were run, or since the object was created. A null value indicates that the number of inserted index entries is unknown.
REORGDELETES	BIGINT	The number of index entries that were deleted from the index space or partition since the last time the REORG , REBUILD INDEX , or LOAD REPLACE utilities ran, or since the object was created. A null value indicates that the number of deleted index entries is unknown.
REORGAPPENDINSERT	BIGINT	The number of index entries with a key value that is greater than the maximum key value in the index or partition that were inserted into the index space or partition since the last time the REORG , REBUILD INDEX , or LOAD REPLACE utilities were run, or since the object was created. A null value indicates that the number of inserted index entries is unknown.
REORGPSEUDODELETES	BIGINT	The number of pseudo-deleted index entries in the index space or partition. A pseudo-delete is a row ID (RID) entry that is marked as deleted. A null value indicates that the number of pseudo-deleted index entries is unknown.
STATSINSERTS	BIGINT	The number of index entries that were inserted into the index space or partition since the last time that the RUNSTATS utility was run or since the object was created. A null value indicates that the number of inserted index entries is unknown.
STATSDELETES	BIGINT	The number of index entries that were deleted since the last RUNSTATS on the index space or partition was run or since the object was created. A null value means that the number of deleted index entries is unknown.
COPYCHANGES	BIGINT	If the COPY utility was run with a SHRLEVEL value other than CHANGE , this value is the number of INSERT , UPDATE , and DELETE operations since the last time that the COPY utility ran. If the COPY utility ran with SHRLEVEL CHANGE , this value is the total number of INSERT , UPDATE , and DELETE operations or the number of rows loaded while the last COPY utility ran, and since the last time that the COPY utility ran. A null value indicates that the number of INSERT , UPDATE , and DELETE operations is unknown.

3.9 Open data set scalability

The **DSMAX** subsystem parameter specifies the maximum number of data sets that can be open at one time. The **DSMAX** value of 200000 is the maximum that is allowed in Db2 11 and Db2 12. The limit of the concurrent data set open limit is a function of two factors: the required 31-bit DBM1 private memory per open data set and the amount of 31-bit private memory remaining in address spaces, such as Db2, after subtracting the required common memory.

Business consolidation, Db2 member consolidation, data growth, and conversion of segmented table spaces to PBGs increased the demand for more concurrent open data sets.

z/OS 2.5 provides an enhancement that moves a portion of the control blocks for open data sets ATB. Much of the improvement is transparent to Db2, and you can take advantage of the enhancement by migrating to z/OS 2.5. In z/OS 2.5, dynamic allocation processing supports scheduler work blocks (SWBs) for data sets in 64-bit storage, which helps reduce BTB storage usage for address spaces with many data sets. Moving SWBs above-the-bar requires exploitation from Db2. Although z/OS 2.5 provides relief, Db2 uses this feature to allow extra room for you to increase the number of concurrent data sets or other Db2 activities, such as more concurrent threads. Db2 performance might be improved when opening or closing many data sets concurrently.

To use this feature, make sure that you use z/OS 2.5 or later and complete one of the following actions:

- Update the ALLOCxx parmlib member to set the **SYSTEM SWBSTORAGE** value to ATB:

```
SYSTEM SWBSTORAGE(ATB)
```

The default value is SWA, which means that SWBs are in 24-bit or 31-bit storage. ATB indicates that SWBs may be in 64-bit storage if the application enables it.

- Issue the **SETALLOC SYSTEM,SWBSTORAGE=ATB** system command.

Updating the ALLOCxx parmlib is best practice because it remains effective across initial program loads (IPLs). If the **SETALLOC** command is used to enable **SYSTEM SWBSTORAGE**, you must restart Db2 for the change to take effect.

The following new DSNB280I message is issued during Db2 restart if the new dynamic allocation function that supports SWB blocks for data sets in 64-bit storage is enabled successfully:

```
DSNB280I DYNAMIC ALLOCATION SUPPORT FOR 64-BIT RESIDENT SWB BLOCKS IS ENABLED
```

Db2 13 increases the upper bound of **DSMAX** to 400000, but it is a best practice that you migrate to z/OS 2.5 and enable the new feature before increasing **DSMAX**.

3.10 Increased size of directory table spaces SPT01 and SYSLGRNX

The DSNDB01.SPT01 directory table space stores static bound packages. As the number of packages and package copies increases, DSNDB01.SPT01 can approach the 64 GB limit.

The DSNDB01.SYSLGRNX directory table space contains the RBA ranges when data sets are opened or closed for updates. Due to the increasing workloads and the increasing number of table spaces, the SYSLGRNX table space size can approach the 64 GB limit.

SPT01 and SYSLGRNX table space are both PBG table spaces with MAXPARTITIONS 1. Db2 13 can increase the DSSIZE limit of these table spaces to 256 GB.

3.10.1 Increasing SPT01 and SYSLGRNX table space size

For a new Db2 13 installation, SPT01 and SYSLGRNX table spaces are already set to 256 GB, which is reflected in the DSSIZE column of SYSTABLESPACE and SYSTABLEPART catalog tables.

When migrating from Db2 12, the first run of **REORG TABLESPACE** with **SHRLEVEL CHANGE** or **REFERENCE** on SPT01 and SYSLGRNX at function level V13R1M500 or higher converts the DSSIZE of the table spaces to 256 GB. The DSSIZE column in SYSTABLESPACE and SYSTABLEPART for the table spaces are updated to 256 GB. If the function level is reverted to V13R1M100*, a table space that is already converted to 256 GB remains unchanged. For a table space that is not converted yet, any run of **REORG** in V13R1M100* does not convert it to 256 GB.

Recovery to a point in time (PIT) before a **REORG** is supported. If the recovery of SPT01 or SYSLGRNX is to a PIT before the **REORG** that converts it to 256 GB, the table space DSSIZE is reverted to 64 GB. The next **REORG** on the table space converts it back to 256 GB if the function level is V13R1M500 or higher.

If any one of the catalog or directory objects is recovered to an earlier PIT, all the catalog and directory objects should be recovered to the same PIT.

3.11 Specifying the IRLM maximum storage for locks

The Db2 installation sets the MAX STORAGE FOR LOCKS field, which is passed to IRLM startup in the **EXEC PARM** statement instead of the **JOB** statement of the **MEMLIMIT** parameter. The **MEMLIMIT** specification in the IRLM startup procedure is also changed. **MEMLIMIT** must be passed as a **PARM** in the **EXEC PARM** instead of the **JOB** statement. If **MEMLIMIT PARM** is set in **EXEC PARM**, it overrides any other z/OS **MEMLIMIT** specification for an IRLM job. This enhancement prevents unexpected Db2 transaction failures due to insufficient IRLM private storage in situations where z/OS system settings for an IRLM job could override IRLM **MEMLIMIT**.

3.11.1 Sample IRLM startup procedure

Example 3-7 shows a sample IRLM startup procedure. The prolog is modified to describe valid values for **MEMLIMIT**, and **MEMLIMIT(&MLMT)** is added as a **PARM** to the **EXEC** statement.

Example 3-7 Sample IRLM startup procedure

```
//*****  
//*          IRLM JCL FOR PROCEDURE FOR THE STARTUP  
//*  
//*          THE MLMT PARAMETER SPECIFIES THE  
//*          THE AMOUNT OF STORAGE AVAILABLE TO  
//*          IRLM "ABOVE THE BAR", THAT IS THE  
//*          Z/OS MEMLIMIT SETTING FOR THE IRLM  
//*          ADDRESS SPACE. THE IRLM DEFAULT  
//*          IS 2160M; IRLM USES 2048M IF YOU  
//*          SPECIFY A VALUE OF LESS THAN 2048M
```

```

//*          FOR MLMT. YOU MAY SPECIFY A VALUE
//*          nnnnnu WHERE nnnnn IS 1-99999 AND u
//*          IS EITHER M, G, T, OR P AS FOLLOWS:
//*          o 2048M - 99999M
//*          o 2G   - 99999G
//*          o 1T   - 99999T
//*          o 1P   - 16384P
//*
//*****
//IRLMPROC PROC RGN=5000K,
//          LIB='DSN!!0.SDXRRESL',
//          IRLMNM=IRLM,
//          IRLMID=1,
//          SCOPE=LOCAL,
//          DEADLOK='1,1',
//          MAXCSA=0,
//          PC=YES,
//          MAXUSRS=7,
//          IRLMGRP=,
//          LOCKTAB=,
//          TRACE=NO,
//          PGPROT=YES,
//          LTE=0,
//          MLMT=2160M
//          EXEC PGM=DXRRLM00,DPRTY=(15,15),
//          PARM=(&IRLMNM,&IRLMID,&SCOPE,&DEADLOK,&MAXCSA,&PC,
//              &MAXUSRS,&IRLMGRP,&LOCKTAB,&TRACE,&PGPROT,&LTE,&MLMT),
//          REGION=&RGN

```

3.12 Dynamic alteration of CF lock structure storage

z/OS Sysplex provides services for monitoring structure utilization, and it can automatically alter the lock structure storage. The monitoring of coupling facility (CF) structure utilization is defined in the coupling facility resource management (CFRM) policy as a threshold percentage value, and it is enabled with a default of 80% when it is not equal to zero. This monitoring retrieves statistics on the CF lock structure every 60 seconds when the storage usage is less than the threshold and every 30 seconds when the storage usage is equal to or greater than the full threshold.

Under heavy workloads, the structure utilization monitoring by z/OS automatically expands the active CF lock structure storage, but that does not help when a spike in locking activities results in rejection of lock requests due to insufficient Record List Entries (RLEs) even before z/OS had a chance to start the CF lock structure alteration. Because IRLM has a higher level of granularity to determine the storage needed, it added a new internal monitoring mechanism that can dynamically expand the CF lock structure size and process the lock requests, instead of rejecting them.

3.12.1 New IRLM messages

New IRLM messages DXR189I and DXR190I are issued when IRLM is altering the CF lock structure size.

Message DXR189I

The syntax of this message is as follows:

```
DXR189I <irlmname> ALTERING LOCK STRUCTURE SIZE
```

When an IRLM member detects a full condition for storage in the CF lock structure, IRLM alters the lock structure storage size if the allocated size is less than the maximum size and the CFRM policy permits altering the lock structure. This message is displayed on the IRLM member that initiated the storage alteration.

IRLM initiates an internal function to alter the CF lock structure size while remaining online.

Message DXR190I

The syntax of this message is as follows:

```
DXR190I <irlmname> ALTER LOCK STRUCTURE COMPLETED
```

The **ALTER** function for CF lock structure storage that is initiated by IRLM completed. If the CF structure size alteration was successful, the increased record list space is available for use.

3.12.2 Function level requirements

The capability of dynamic alteration of CF lock structure storage is available in IRLM 2.3 with function level 50C, which is included with Db2 13.

3.13 Summary

Db2 13 provides many new and changed features that greatly enhance its availability and scalability. Some of these important features focus on PBR table spaces that use RPN because this table space attribute provides unmatched scalability and availability. You can take advantage of PBR RPN without incurring higher CPU usage or detrimental performance impact.

Db2 13 also provides essential optimization and expansion of database and system resources, including storage usage, database connections, real-time statistics, concurrent opened data sets, directory table spaces, and locking capacity.

Together, these enhancements, which are available at function level V13R1M500, aim to significantly increase the availability and scalability of your Db2 system.



Performance

As in previous versions, Db2 13 focuses on providing important performance enhancements and benefits, especially in the areas of CPU cost reduction, scalability, reliability, and consistency. These enhancements help to optimize the performance of your Db2 system and reduce the need for unplanned performance tuning.

At the time of writing, Db2 13 performance evaluations are not yet finalized. IBM expects that Db2 13 performance will be equivalent or better than Db2 12 on the same workloads.

This chapter describes the most notable performance enhancements in Db2 13, many of which are available by migrating to Db2 13 (function level V13R1M100).

This chapter describes the following topics:

- ▶ Fast index traversal enhancements
- ▶ Index look-aside optimization
- ▶ INSERT enhancements for partition-by-growth table spaces
- ▶ SORTL enhancements
- ▶ Reducing the CPU impact in a data-sharing environment when using PBR table spaces with relative page numbers
- ▶ More performance enhancements

4.1 Fast index traversal enhancements

Fast index traversal, also known as Fast Traversal Blocks (FTB), is one of the most important performance features that was introduced in Db2 12. It is an in-memory index performance optimization that enables Db2 to do fast index lookup by avoiding expensive index B-tree traversal. Fast index traversal is most beneficial when data is accessed in a random pattern. Any operation that requires traversal of an index, including the **SELECT**, **INSERT**, **DELETE**, and **UPDATE** statements, can benefit from fast index traversal.

When FTB was initially introduced, you had to define indexes as **UNIQUE** to qualify for FTB access. The feature supported unique indexes with **INCLUDE** columns, but the length of the index entry, including the key and any additional columns, was capped at a maximum size of 64 bytes.

Function level 508 (V12R1M508) removed some of the restrictions. Only the key size of the ordering columns had to be 64 bytes or less, and columns on the **INCLUDE** list no longer counted toward the size limit for the index key. However, fast index traversal was not used for the columns in the **INCLUDE** list.

APAR PH30978, when applied to FL 508, strengthened FTB support for non-unique indexes. After you reset the value of the **FTB_NON_UNIQUE_INDEX** subsystem parameter from the default **NO** to **YES**, all non-unique indexes became eligible for fast traversal processing. But, like for unique indexes, the key size for the columns of non-unique indexes had to be 56 bytes or less.

With FL500 (V13R1M500), Db2 13 continues to enhance FTB and extends the maximum key length for FTB eligible indexes as follows:

- ▶ For unique indexes, the key length is limited to 128 bytes.
- ▶ For unique indexes with **INCLUDE** columns, the unique part of the index must not exceed 128 bytes.
- ▶ For non-unique indexes, the key length is limited to 120 bytes.

In addition, the **FTB_NON_UNIQUE_INDEX** subsystem parameter is set to **YES** in Db2 13 by default, which means that non-unique indexes are automatically eligible for FTB processing if they meet other eligibility criteria, such as the maximum key lengths.

4.2 Index look-aside optimization

Database administrators create indexes on Db2 tables to improve SQL query (**SELECT**) performance. However, the cost of maintaining those indexes during the SQL **INSERT**, **DELETE**, and **UPDATE** operations, and the **GETPAGE** activities, increases as their size grows.

As described in 4.1, “Fast index traversal enhancements” on page 58, Db2 12 introduced an in-memory technique called fast index traversal (FTB). However, when FTB is not used for a particular index, Db2 must traverse the index tree from the root page to access the leaf pages for non-clustering indexes with a low cluster ratio for **INSERT** and **DELETE** operations or for all indexes for **UPDATE** operations. This process is CPU-intensive, which produces a huge performance impact.

To remove this performance barrier, Db2 introduced the *index look-aside* technique. Db2 tracks index value ranges and checks whether a requested index entry is on the same leaf page as the previous request. If the key that is being requested is on the same leaf page as the previous request, Db2 returns the entry without traversing the index B-tree or even performing a **GETPAGE** request for the leaf page, which results in significant CPU savings for index-intensive operations.

In Db2 12, index look-aside is used only by SQL **INSERT** and SQL **DELETE** operations on the clustering index or non-clustering index with a high cluster ratio that is based on catalog statistics.

Db2 13 enhances the usage of index look-aside by storing the last accessed index non-leaf and leaf page information for SQL **INSERT**, **DELETE**, and **UPDATE** statements when the thread does more than three **INSERT**, **DELETE**, and **UPDATE** operations in the same commit scope. Index look-aside is now enabled for all indexes during SQL **INSERT**, **DELETE**, and **UPDATE** operations, regardless of the cluster ratio. You do not need to make any application or index changes to leverage this enhancement.

This index look-aside enhancement reduces index maintenance cost for workloads that are **INSERT**, **DELETE**, and **UPDATE** intensive by reducing the number of index **GETPAGE** requests. The same benefits are also evident when FTB is used because it reduces the index **GETPAGE** cost for the **INSERT**, **DELETE**, and **UPDATE** operations.

In addition, workloads can benefit from index look-aside even when the catalog statistics on the index are not accurate because **RUNSTATS** has not run recently. This enhancement dynamically adjusts the usage of index look-aside to avoid any impact when Db2 detects that the **INSERT**, **DELETE**, and **UPDATE** pattern is random and index look-aside does not provide a benefit.

Db2 internal measurements show performance improvements with no CPU or elapsed time regression for the **INSERT**, **DELETE**, and **UPDATE** workloads that were evaluated in both data-sharing and non-data-sharing environments. You can expect more benefits when fast index traversal is not used for these indexes. A greater reduction in **GETPAGE** activity is expected for non-clustering indexes, especially when they have a high cluster ratio.

By reducing index maintenance cost during SQL **INSERT**, **DELETE**, and **UPDATE** operations, you can create and maintain more indexes on a table to provide faster access to the data and index-only access to data while avoiding sort processing for certain SQL statements. With these CPU improvements, your Db2 installation can handle new workloads while accommodating existing workload growth.

4.3 INSERT enhancements for partition-by-growth table spaces

Db2 13 improves the cross-partition search algorithm for **INSERT** transactions, which achieves higher success rate with minimum performance impact. Enhancements to the partition retry logic after a lock contention and to the bidirectional partition search logic, which are described in the following sections, apply to the partition-by-growth (PBG) table spaces that contain more than one partition.

4.3.1 Partition retry process

When inserting into a PBG table space, the **INSERT** transaction goes through the Db2 locking hierarchy protocol by first acquiring a partition-level lock and then lower-level locks, such as page or row locks. To achieve better performance in a high-concurrency environment, the transaction requests the partition lock conditionally when the PBG table space contains more than one partition. If the Internal Resource Lock Manager (IRLM) rejects the lock request for this partition, the **INSERT** operation skips the partition and the space search algorithm moves on to the next partition. The same operation is performed for the next available partition until the **INSERT** operation either completes successfully or fails after an exhaustive search through all existing partitions.

In this case, the **INSERT** operation fails due to being unable to acquire a conditional partition lock. The application receives an SQL return code -904 (resource not available). The reason code that is associated with this SQL code can be either be “partition full” (00C9009C) or “conditional lock failure” (00C90090). Either way, there is not enough information for you to diagnose the problem.

The contention duration on a partition lock that is encountered by an **INSERT** statement is usually short unless the **INSERT** application is running while another application holds a gross lock by design, for example, with a **LOCK TABLE** statement.

If the lock request is conditional, there is no lock wait time when IRLM detects contention for an incompatible lock; IRLM informs Db2 that the lock is not available. Because the **INSERT** operation does not retry after cycling through all available partitions, the **INSERT** statement is terminated.

Db2 13 improves the cross-partition search algorithm for **INSERT** transactions by retrying the partition after the initial conditional lock failure. Db2 maintains an in-memory list of up to 10 partitions that previously experienced a partition lock request failure. This list typically represents the higher partition numbers within the table space. However, even though up to 10 partitions are tracked, only five or fewer partitions are retried. The retry process can acquire the partition lock conditionally or unconditionally. When an unconditional partition lock is requested, the lock timeout interval is adjusted to approximately 1 second regardless of the setting of the **IRLMRWT** subsystem parameter (**ZPARM**). This interval helps prevent that long lock wait time that contributes to the degrading of system performance.

An application can still fail after the cross-partition retry process if the **INSERT** operation cannot successfully acquire the unconditional partition lock. The DSNT376I timeout message and the DSNT501I resource unavailable message, as shown in the following output, provide information about the lock contention, timeout interval, and resource consumption, which you can use to take corrective actions:

```
DSNT376I  -DB2A PLAN=DSNTEP3  WITH 691
          CORRELATION-ID=INSERTA4
          CONNECTION-ID=BATCH
          LUW-ID=DSNCAT.SYEC1DB2.DAE53CAF57D2=9
          THREAD-INFO=SYSADM: BATCH:SYSADM: INSERTA4: DYNAMIC: 3: *: *
          IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS PLAN=DSNTEP3

WITH
          CORRELATION-ID=TQI01005
          CONNECTION-ID=BATCH
          LUW-ID=DSNCAT.SYEC1DB2.DAE53CA276CB=8
          THREAD-INFO=SYSADM: BATCH:SYSADM: TQI01005: DYNAMIC: 1: *: *
          ON MEMBER DB2A
          REQUESTER USING TIMEOUT VALUE=1 FROM IRLMRWT
```

```
HOLDER USING TIMEOUT VALUE=60 FROM IRLMRWT
DSNT501I -DB2A DSNILMCL RESOURCE UNAVAILABLE 692
CORRELATION-ID=INSERTA4
CONNECTION-ID=BATCH
LUW-ID=DSNCAT.SYEC1DB2.DAE53CAF57D2=9
REASON 00C9008E
TYPE 00000210
NAME DB01 .TESTTS01.00000001
```

4.3.2 Improved cross-partition search in a descending partition number sequence

The cross-partition search for PBG table spaces is bidirectional. After an **INSERT** statement fails to find free space in the initial target partition, which is selected based on the clustering index, the partition to be searched next can be either an ascending or descending partition sequence number. The searching order is randomly decided at run time to avoid creating a “hot spot” in a certain partition.

In earlier Db2 versions, when a PBG table space has many empty partitions at the end, the descending cross-partition search algorithm often uses the last physical partitions and can leave many empty partitions unused in between. When Db2 reaches the first physical partition during a descending partition search, it wraps around and looks at the last physical partition next.

Db2 13 improves the descending partition search algorithm by reusing other trailing empty partitions more efficiently before the last physical partition is used. To do this task, Db2 now tracks the highest non-empty partition within the table space at run time.

During the execution of an **INSERT** statement, Db2 caches the highest non-empty partition that is accessed in the memory. Tracking starts when a data set is opened until it is closed.

When performing a descending cross-partition search and after reaching partition 1, the cached highest non-empty partition is used as the next partition to search. This information is tracked separately by each data-sharing member without any cross-communication or validation between the different members of a data-sharing group. The accuracy of the last non-empty partition that was accessed depends on the activities within each data-sharing member. The workload balance between each data-sharing member can be an important factor for reusing trailing empty partition more efficiently.

4.4 SORTL enhancements

The IBM z15 processor introduced a new feature that is called the Integrated Accelerator for IBM Z Sort to improve the performance of **SORT** processing. This feature includes a new **SORT LISTS (SORTL)** instruction.

With APAR PH31684, Db2 introduced the use of the **SORTL** instruction for sorting data during the execution of SQL statements by using the z15 sort accelerator. Initially, several rules and restrictions were added to the Db2 sort component to determine when and whether the **SORTL** instruction can be used.

Here is a list of some conditions that must be met for **SORTL** to be used by the Db2 sort component:

- ▶ A z15 processor must be used.
- ▶ There are no plans or packages before Db2 12.
- ▶ The key size must be greater than 0 but equal to or smaller than 136 bytes, and the data size must be equal to or smaller than 256 bytes.

Db2 13 maximizes the use of **SORTL** by analyzing data from previous executions, such as **SRTPOOL** size, key size, and data size. This enhancement helps reduce the workfile usage and storage consumption, which results in the reduction of CPU usage and elapsed time for running SQL statements.

4.5 Reducing the CPU impact in a data-sharing environment when using PBR table spaces with relative page numbers

A comparison between PBR table spaces with relative page numbers (RPNs) and absolute page numbers (APNs) might show higher CPU utilization when RPN is used. The increase in CPU usage is caused by a significant increase in false contentions for page P-locks. This issue primarily affects the table spaces that use row-level locking due to the usage of page P-locks for data pages.

To address this issue, Db2 must generate more unique hash values for lock resources, which are used in IRLM and z/OS cross-system extended services (XES) for locking. Db2 13 introduces internal changes to the resource hash values of page P-locks. The new hash algorithm provides a more balanced distribution of resource hash values, which reduces false locking contentions and CPU processing impacts. Db2 13 also modifies data page and index page header definitions. Large object (LOB) and XML pages are not affected by this change.

4.5.1 Considerations for function level and application compatibility level

The CPU processing impact reduction applies only to PBR table spaces that use RPN. To leverage this enhancement in Db2 13, you must activate function level 500 (V13R1M500). When a data-sharing group is at FL 500 or higher, new table spaces with RPN use the new lock hash values for data page P-lock resources. For existing table spaces with PBR RPN that were created with a lower function level or in Db2 12, you must run **REORG** or **LOAD REPLACE** to use the new lock hash values for PBR RPN when they are in an environment that is activated with FL 500 or higher. Because this function can take effect at a partition level, you do not need to run **REORG** on the entire table space.

If you activate V13R1M500, run **REORG**, and start using the new lock hash value for PBR RPN, you can activate a lower function level V13R1M100*. In this situation, Db2 continues to use the new hashing algorithm for page P-locks because V13R1M100* does not allow co-existence with or fall back to Db2 12. A new flag that is called **LOGRPNH2** in page set open and checkpoint log records indicates whether the table spaces were converted.

4.6 More performance enhancements

Other notable performance-related enhancements originate from the externalization of an online-changeable **REORG_INDEX_NOSYSUT1 REORG INDEX** subsystem parameter and the new ability of Group Buffer Pool (GBP) residency time, which are briefly highlighted in the following sections but fully described in other chapters.

4.6.1 REORG INDEX performance improvements

When **REORG INDEX** is invoked with **SHRLEVEL REFERENCE** or **CHANGE**, you can use the **NOSYSUT1** keyword to indicate that the utility operation will not use work data sets to hold the unloaded index keys.

This capability was added as a Db2 12 continuous delivery enhancement and then simplified with the externalization of an online-changeable subsystem parameter (**ZPARM**) that is called **REORG_INDEX_NOSYSUT1**. IBM performance measurements showed that this enhancement resulted in a significant reduction of elapsed time and CPU time. For partitioned indexes, the tests showed up to 86% reduction of elapsed time, and the CPU reduction was even better.

Db2 13 further simplifies the execution of the **REORG INDEX** utility. With the activation of function level 500 (V13R1M500), you no longer need to rely on the **REORG_INDEX_NOSYSUT1** subsystem parameter. The **NOSYSUT1** behavior that is described is used whenever possible for **REORG INDEX** with **SHRLEVEL REFERENCE** or **CHANGE**.

For more information, see Chapter 8, “IBM Db2 for z/OS utilities” on page 111.

4.6.2 Group buffer pool residency time enhancements

z15 hardware introduces the ability to track data and directory residency statistics for GBPs. The statistics provide a way to see how long data stays in the buffer pool before being removed. This information can help you adjust workload balance and GBP sizing to improve performance. You can leverage this feature in all Db2 13 subsystems that meet the following criteria:

- ▶ Your Db2 system runs on z/OS 2.4 (HBB77C0) or 2.5 (HBB77D0) with the necessary PTFs for cache residency time metrics (for more information about the PTFs, see APAR OA60650).
- ▶ Your CF runs on a CF Level 25 z16 machine or later.

Db2 13 adds two fields to relevant GBP statistics storage areas:

- ▶ The average time in milliseconds that a data area stays in a GBP before it is removed.
- ▶ The average time in milliseconds that a directory entry stays in a GBP before it is removed.

Both metrics are accessible through more fields in Instrumentation Facility Component Identifier (IFCID) 230 and 254 trace records and by using the **-DISPLAY GROUPBUFFERPOOL** command. For more information, see Chapter 9, “Instrumentation and serviceability” on page 131.

4.7 Summary

Db2 13 continues the never-ending pursuit of minimizing processing cost and reducing elapsed time for all types of workloads in Db2. Through a combination of internal enhancements, monitoring improvements, and IBM Z feature exploitation, Db2 13 helps you run new and existing workloads faster while minimizing cost and optimizing performance.



Application management and SQL changes

Application development and management are important topics for Db2 for z/OS due to the complex process and volume of applications that are exposed on the platform. Certain older applications are difficult to change, and all changes to applications typically require you to follow a strategic process from development to test before deploying to the production environment. In a cloud environment that hosts multi-tenancy, applications are typically different in characteristics. Each application can access different database objects, so each can have its own concurrency requirements and toleration.

You can now use Db2 13 to set application-granularity lock controls such as timeout interval and deadlock resolution to match the individual application's need. You can do this task without changing the application's source code, regardless of whether the application is local on the z/OS system or originates from a remote system.

Db2 13 introduces a mechanism to optimize the success of DDL break-in without needing to duplicate versions of the application packages and without impacting non-dependent applications.

Another challenge for Db2 administrators is managing new function permission for remote applications that connect through Db2 Data Server Drivers. As an administrator, you can configure two or more collections of packages for these drivers to allow new functions for one set of applications while blocking new functions from another set. The assignment to the appropriate collection can now be managed by using the DSN_PROFILE tables.

This chapter describes the following topics:

- ▶ Application timeout and deadlock control
- ▶ Profile table enhancements
- ▶ Enhancing the database object change process
- ▶ Migrating distributed applications

Db2 13 added support for SQL Data Insights with several new built-in functions. For more information about this feature, see Chapter 6, "SQL Data Insights" on page 81.

5.1 Application timeout and deadlock control

Throughout the history of relational databases, managing concurrency among threads that run SQL statements is one of the most important factors for protecting data integrity. Locking objects is still the primary mechanism for allowing concurrent thread execution while ensuring consistent, correct results when multiple threads work on those same objects.

The first version of Db2 for z/OS incorporated the Internal Resource Lock Manager (IRLM) to handle locking, and since that first Db2 version, all lock requests are susceptible to contention resolution, granting, waiting, and resuming. A key variable in lock-handling behavior is the timeout interval, which specifies the maximum amount of time a lock request can wait in contention before the request is denied. Nearly all logical lock requests from a Db2 system to IRLM used a common timeout interval that is set in the **IRLMRWT** subsystem parameter. This parameter, which is specified in seconds, is used as the system-wide timeout interval for lock requests regardless of where they come from. In Db2 12 and previous versions, this subsystem parameter was not online changeable, which means that you had to recycle the Db2 subsystem whenever you had to change this value.

However, not all applications have the exact same characteristics. Some applications can tolerate a lock wait time interval that is longer or shorter than the **IRLMRWT** value. Splitting an application off into another Db2 subsystem with a more suitable timeout interval according to application affinities increases management cost and decreases availability and resiliency.

To help manage applications that have different characteristics in the same Db2 subsystem or data-sharing group, Db2 13 provides the new **CURRENT LOCK TIMEOUT** special register.

You can set this new special register at an application level to control the number of seconds that the application waits for a lock before timing out. The Db2 for z/OS support is compatible with Db2 for Linux, UNIX, and Windows (LUW) support of this special register. To use this feature, Db2 must be at function level V13R1M500 or later and the application must be set to use application compatibility (APPLCOMPAT) (V13R1M500) or later. For more information, see 5.1.1, “CURRENT LOCK TIMEOUT” on page 66.

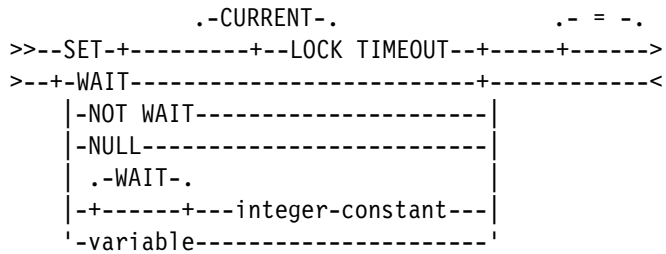
Similarly, when different lock requests from multiple threads form a deadlock cycle, IRLM resolves the deadlock by choosing a *victim* whose request is then denied. Previously, your application could do little to influence which thread is chosen as the deadlock victim. Batch data definition jobs frequently fail due to deadlocks. Attempting to address this issue through restart or retry logic and careful scheduling is increasingly difficult in continuous availability environments. Therefore, when an SQL Data Definition Language (DDL) statement runs, failing that statement instead of other statements might not be optimal.

Db2 13 introduces the new **DEADLOCK_RESOLUTION_PRIORITY** built-in global variable, which allows you to specify a deadlock resolution priority value to use in resolving deadlocks with other threads. This feature requires Db2 at function level V13R1M501 or later and your application to use APPLCOMPAT (V13R1M501) or later. The M501 function level is required because the new built-in global variable is defined in catalog level V13R1M501. For more information, see 5.1.2, “DEADLOCK_RESOLUTION_PRIORITY” on page 68.

5.1.1 CURRENT LOCK TIMEOUT

You can use the **SET CURRENT LOCK TIMEOUT** SQL statement in an application to assign an integer value to the **CURRENT LOCK TIMEOUT** special register that suits the needs of a specific application or even an individual SQL statement. This value overrides the **IRLMRWT** subsystem parameter. While the **IRLMRWT** value range is 0 - 3600 seconds, the **CURRENT LOCK TIMEOUT** range is -1 to 32767 seconds.

The following diagram shows the **SET CURRENT LOCK TIMEOUT** statement syntax:



The **SET CURRENT LOCK TIMEOUT** statement accepts the following values:

NULL	The IRLMRWT value is used to determine how long a lock request waits. This NULL value is the default value for the special register. A query of CURRENT LOCK TIMEOUT returns the IRLMRWT value.
0 or NOT WAIT	The lock request is conditional. If a lock cannot be obtained, an error is returned immediately. In this case, the DSNT376I message and Instrumentation Facility Component Identifier (IFCID) 196 are not written.
-1 or WAIT	A timeout cannot occur. The application waits indefinitely until the lock is released by the holder.
WAIT <i>n</i> or <i>n</i>	<i>n</i> is an integer or variable value 1 - 32767 that indicates the number of seconds to wait for a lock.

Important: Be careful when using a high value or -1 for the **CURRENT LOCK TIMEOUT** special register because the application thread might be holding many resources for a long time, which might impact other concurrent application threads that need those resources. To aid in diagnosing such situations, you can use IFCID 437 to monitor the value that is specified on the **SET CURRENT LOCK TIMEOUT** SQL statement. The Db2 accounting trace record includes a new counter for successful executions of this statement.

Additionally, you can use the new **SPREG_LOCK_TIMEOUT** subsystem parameter to control the maximum value that can be specified for the **SET CURRENT LOCK TIMEOUT** statement. The default value for **SPREG_LOCK_TIMEOUT_MAX** is -1, which allows all valid values to be set for the **CURRENT LOCK TIMEOUT**. After you migrate to Db2 13, the subsystem parameter load module must be rebuilt. If not, the **SPREG_LOCK_TIMEOUT_MAX** value might be set to 0 unintentionally, which can cause a **SET CURRENT LOCK TIMEOUT** statement to fail.

Because no corresponding bind option exists for this special register, **CURRENT LOCK TIMEOUT** is applicable to locks for both static and dynamic SQL statements in an application.

New fields in the trace record IFCID 196 and new text in the DSNT376I message are added to help identify when a lock request uses the **IRLMRWT** or **CURRENT LOCK TIMEOUT** value. For example:

```

DSNT376I PLAN=APPL1 WITH CORRELATION-ID=correlation-id1
CONNECTION-ID=connection-id1 LUW-ID=luw-id1 THREAD-INFO=thread-information1 IS
TIMED OUT.
ONE HOLDER OF THE RESOURCE IS PLAN=APPL2 WITH
CORRELATION-ID=correlation-id2
CONNECTION-ID=connection-id2 LUW-ID=luw-id2 THREAD-INFO=thread-information2 ON
MEMBER member-name.
REQUESTER USING TIMEOUT VALUE 10 FROM Special Register. HOLDER USING
TIMEOUT VALUE 30 FROM IRLMRWT.

```

When the CURRENT LOCK TIMEOUT special register is set, the actual amount of time it takes for the IRLM to deny a waiting lock request (total wait time) is determined by the value of the CURRENT LOCK TIMEOUT special register and the IRLM deadlock interval. This behavior is the same as though you used the **IRLMRWT** subsystem parameter to set the wait interval on a lock request.

You can use the following IRLM commands to obtain information or change the setting of the IRLM deadlock value:

- ▶ **F ir1mproc,STATUS** displays the IRLM deadlock interval.
- ▶ **F ir1mproc,SET,DEADLOCK=nnnn** (where *nnnn* is number of milliseconds) to change the IRLM deadlock interval.

Data-sharing communication to resolve lock contention does result in some processing impact. Therefore, specifying a smaller IRLM deadlock interval can result in IRLM timing out a lock request closer to the wanted wait interval, which is expressed either by **IRLMRWT** or **CURRENT LOCK TIMEOUT**. The program temporary fix (PTF) for the IRLM APAR PH43770 is recommended for a Db2 13 data-sharing group, which helps reduce the impact in data-sharing communication so that a waiting lock request can be denied closer to the **CURRENT LOCK TIMEOUT** value.

Lock requests from a DDL statement, such as **CREATE**, **ALTER**, or **DROP**, wait longer than the **IRLMRWT** value when the **DDLTOX** subsystem parameter is set to a value greater than 1. However, the **DDLTOX** setting affects all DDL statements. A database administrator (DBA) who performs an object schema change might consider setting the **CURRENT LOCK TIMEOUT** to a specific value that is suitable for the specific DDL operation being performed.

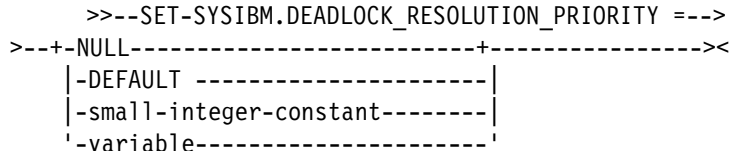
Additionally, the **CURRENT LOCK TIMEOUT** special register value is also acknowledged by certain Db2 internal synchronization processes that are similar to lock requests. Those processes include claims and drains, and when a DDL statement waits for cached dynamic SQL statements to be quiesced. Conversely, certain Db2 internal processes might use a different value than the **CURRENT LOCK TIMEOUT** value in effect. For example, a P-lock request does not acknowledge the **CURRENT LOCK TIMEOUT** value because P-locks are owned by the Db2 subsystem and not by the application. The **CURRENT LOCK TIMEOUT** special register affects logical locks only in an application.

5.1.2 DEADLOCK_RESOLUTION_PRIORITY

Before Db2 13, influencing how Db2 chooses a requester to cancel when multiple processes form a deadlock cycle with lock requests such that no process can proceed is not easy. Some techniques are available, but most have drawbacks or are hard to control. For example, a process that performs many database update activities with many log records is less likely to be denied than a process that makes fewer updates. Also, setting the **BMPTOUT** subsystem parameter for Information Management System (IMS) batch messaging processing (BMP) transactions and the **DLITOUT** subsystem parameter for IMS data language interface (DLI) transactions can give those transactions a heavier weighting factor in the deadlock resolution process. However, not all IMS BMP or DLI applications have the same characteristics. Setting a shared high priority for all these types of applications might not be the best decision, especially when they are run with important, non-IMS applications. An IBM Customer Information Control System (IBM CICS®) or IMS transaction can specify its own weighting factor on the attachment API Create Thread API, but this method is allowed only for those types of applications.

Db2 13 introduces the new built-in `SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY` system global variable, which can be used by any application through a simple SQL API. You can use the new `SET SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY` SQL statement to set a deadlock resolution priority (weighting value). This value is used when Db2 resolves deadlocks with other threads. The valid range is 0 – 255. The higher the value, the less likely that lock requests that are requested by an application are denied when the application is involved in a deadlock situation.

The following diagram shows the `SET SYSIBM.DEADLOCK_RESOLUTION_PRIORITY` statement syntax:



Like other global variables, the `WRITE` privilege is required on this global variable to run the `SET SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY` statement, and the `READ` privilege is required for an application to query the value of this global variable. Therefore, the DBA has a way to control which application uses this global variable and can monitor its setting because the usage of this global variable can affect other applications.

A DBA can choose a deadlock resolution priority value for a scheduled data definition process that helps ensure that the process is denied in an eventual deadlock scenario, thus optimizing the likelihood of success.

Other factors are involved in the deadlock resolution process. For example, a lock request for a no-logged table is unlikely to be denied. A higher `DEADLOCK_RESOLUTION_PRIORITY` value does not ensure that an application will never be denied. Therefore, the application setting of this global variable should still be ready to handle `SQLCODE -911` or `-913` errors.

The trace record `IFCID 172` reports the worth values of the holders and waiters that are involved in the deadlock cycle and whether the worth values are from the `DEADLOCK_RESOLUTION_PRIORITY` global variable.

In conclusion, Db2 13 provides two new special registers and a system built-in global variable to enable an application to control its own lock timeout and deadlock behaviors that suit the application. Furthermore, the DBA can set the `CURRENT LOCK TIMEOUT` and `DEADLOCK_RESOLUTION_PRIORITY` values for certain applications without having to incur the cost of changing the applications, as described in 5.2, “Profile table enhancements” on page 69.

5.2 Profile table enhancements

Support for profiles was added to an earlier version of Db2 to monitor and control various aspects of a Db2 subsystem in specific application contexts. A *system profile* is a set of criteria that identifies a specific context on a Db2 subsystem. Examples include threads, connections, or application plans and packages that contain SQL statements with certain attributes. The rows in the `DSN_PROFILE_TABLE` table define profiles and associated filtering criteria. Each row in the `DSN_PROFILE_TABLE` table has a unique `PROFILEID`. The actions that Db2 takes for processes that meet the filtering criteria for a profile in the `DSN_PROFILE_TABLE` table are defined by one or more rows in the `DSN_PROFILE_ATTRIBUTES` table that have the same `PROFILEID` value.

Then, Db2 added the ability to set special registers primarily by using the DSN_PROFILE_TABLE and DSN_PROFILE_ATTRIBUTES tables. In a further enhancement, Db2 added the ability to set built-in global variables through the profile tables. These functions are all part of the system monitoring feature of the profile table.

Although up until now these features were applicable to remote applications only, they are useful because remote applications do not need to be changed to take advantage of certain special registers or global variables. Making application changes can be a resource-intensive task that involves understanding the applications and thoroughly testing the changes before they are promoted to the production system. Using a profile to add the setting of a special register or global variable eliminates the need for multiple versions of an application, especially when these settings are temporary.

To further reduce the costs that are associated with the application change process and simplify the application management task, Db2 13 extends the profile table function to qualify both remote and local application processes and adds support for the CURRENT LOCK TIMEOUT special register and the DEADLOCK_RESOLUTION_PRIORITY global variable.

5.2.1 Profile table support for local applications

You can insert rows in the DSN_PROFILE_ATTRIBUTES table with the KEYWORD column equal to the string 'SPECIAL_REGISTER' or 'GLOBAL_VARIABLE' to indicate that Db2 should run the SQL SET statement that is specified in the DSN_PROFILE_ATTRIBUTES.ATTRIBUTE1 column. The application context where the SET statement runs is expressed or filtered by the corresponding row in the DSN_PROFILE_TABLE table with the matching PROFILEID column value.

For example, you can specify that when an application package with the client application name ABCD runs or when the primary authorization ID UTHRED runs any application package, the CURRENT LOCK TIMEOUT is set to 5. This behavior means that lock requests for SQL statements in these local or remote applications can wait for a maximum of 5 seconds in a case of contention.

The DSN_PROFILE_ATTRIBUTES.ATTRIBUTE2 column indicates which application type is applicable to which application row. The following values are valid:

Null	This value applies to remote applications only. SET statements are processed when the first package is loaded for execution (the first SQL statement in the first package is loaded).
1	This value applies to local applications only. SET statements are processed when each package is loaded (the first SQL statement in each package is loaded).
2	This value applies to both remote and local applications.

The Db2 DDF address space must be started to use the system monitoring function of the profile tables, which includes setting special registers and global variables. Even if you use the profile tables for local applications only, Db2 must be started with the DDF subsystem parameter set to AUTO or COMMAND. Db2 issues the message DSNT761I if DDF is not started, as shown in Figure 5-1 on page 71.


```

-DB2ASTA PROFILE
DSNT761I  -DB2A DSNT1RSP START PROFILE DETECTED
          DDF IS NOT LOADED. PROFILE ROWS FOR SYSTEM
          MONITORING WILL NOT BE ACTIVATED.
DSNT741I  -DB2A DSNT1SDV START PROFILE IS COMPLETED.
DSN9022I  -DB2A DSNT1STR 'START PROFILE' NORMAL COMPLETION

```

Figure 5-1 DSNT761I message

In general, after issuing the **-START PROFILE** command, you should query the DSN_PROFILE_HISTORY and DSN_PROFILE_ATTRIBUTES_HISTORY tables for any rejected rows. An error message “DDF NOT LOADED” might be issued in a DSN_PROFILE_ATTRIBUTES_HISTORY row if the Db2 DDF address space is not active.

The values 1 and 2 in the DSN_PROFILE_ATTRIBUTES.ATTRIBUTE2 column are allowed when the DSN_PROFILE_ATTRIBUTES.KEYWORDS column is SPECIAL_REGISTER or GLOBAL_VARIABLE.

You can insert the **SET CURRENT LOCK TIMEOUT** or **SET SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY** SQL statements into the DSN_PROFILE_ATTRIBUTES.ATTRIBUTE1 column. Most syntax variations of these SQL statements can be used in this column content except for the following values:

- ▶ A host variable
- ▶ A **WAIT** integer-constant (specify the integer-constant without the **WAIT** keyword instead)
- ▶ **MODE**
- ▶ **TO**

Function level V13R1M500 is required for the **SET CURRENT LOCK TIMEOUT** statement, and function level V13R1M501 is required for the **SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY** statement when these statements are specified in the DSN_PROFILE_ATTRIBUTES table. The row is rejected if the current function level requirement is not met when the **-START PROFILE** command is issued.

When setting special registers or global variables for local applications, you can identify the local applications by using only the following filtering criteria in the columns of the DSN_PROFILE_TABLE table:

- ▶ AUTHID, ROLE, or both
- ▶ COLLID, PKGNAME, or both
- ▶ One of CLIENT_APPLNAME, CLIENT_USERID, or CLIENT_WORKSTNNAME

Table 5-1 and Table 5-2 on page 72 shows how you might specify the new special register and global variable in profile tables.

Table 5-1 DSN_PROFILE_TABLE

PROFILEID	COLLID	ROLE	PROFILE_TIMESTAMP
1	COLLECTION1		2022-03-26-22.46.45.78344
2		DB2DEV	2022-03-26-22.46.47.78344

Table 5-2 DSN_PROFILE_ATTRIBUTES

PROFILEID	VALUE	ATTRIBUTE1	ATTRIBUTE2
1	SPECIAL_REGISTER	SET CURRENT LOCK TIMEOUT = 10	Null
2	SPECIAL_REGISTER	SET CURRENT LOCK TIMEOUT = 15	1
3	GLOBAL_VARIABLE	SET SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY = 5	2

In this example, after the profile is started by issuing the **-START PROFILE** command, the following actions occur:

- ▶ A remote package that is bound in collection ID COLLECTION1 uses the CURRENT LOCK TIMEOUT value of 10 seconds.
- ▶ A local package that is run in a trusted context with role name of DB2DEV uses the CURRENT LOCK TIMEOUT value of 15 seconds.
- ▶ Both remote and local packages running under the trusted context with the role name of DB2DEV use the DEADLOCK_RESOLUTION_PRIORITY of 5.

Db2 looks up the profile tables to set the special register, global variable, or both when each local package is loaded for execution. A package is loaded on the first SQL statement execution in that package. If the package is released at **COMMIT** or **ROLLBACK** due to the **RELEASE (COMMIT)** bind option behavior, the next SQL statement that runs after **COMMIT** or **ROLLBACK** loads the package again. A package for a stored procedure or user-defined function (UDF), even if invoked from a remote SQL statement, is considered a local package. At load package time, if an explicit SQL **SET** statement has not run for the special register or global variable from an application, Db2 runs the **SET** statement in the profile table because the explicit SQL **SET** statement in the application takes higher precedence than the **SET** statement in the profile table.

Special register values are not saved and restored at package switching time, which is when a package uses the language **CALL** statement to invoke another package (not the SQL **CALL** statement to run a Db2 stored procedure). Global variables are not saved and restored, even when a stored procedure or UDF is invoked. To avoid confusion about which special register and global variable values are used in an application that includes several packages, a best practice is to not mix the SQL **SET** statements in both the user application and the Db2 profile table.

In this example, PackageA and PackageB contain the statements that are shown in Figure 5-2.

```

PackageA:
SELECT * FROM T1
(language) CALL PackageB
                PackageB:
                    SET CURRENT LOCK TIMEOUT = 10
                    INSERT INTO T2
                    RETURN
UPDATE T1
    
```

Figure 5-2 Statements in packages A and B

If a **SET CURRENT LOCK TIMEOUT = 20** statement is in the profile table for package A, then the **UPDATE** statement in PackageA uses a **CURRENT LOCK TIMEOUT** of 10 seconds, even though the earlier **SELECT** statement in the same package uses the **CURRENT LOCK TIMEOUT** of 20 seconds. A **CURRENT LOCK TIMEOUT** of 10 seconds is used because the explicit SQL **SET** statement in PackageB overrode the value in the profile table. For diagnostic purposes, a query of the interested special register or global variable can be issued right before the SQL statement.

5.3 Enhancing the database object change process

Making database object schema changes is challenging in environments in which online transactions are continuously running. Although DDL statements such as **ALTER** or **DROP**, which change a database object, might not run frequently, when they are, they might impact applications that reference the same object, both in terms of performance and availability.

Applications that contain static Data Manipulation Language (DML) statements, such as **SELECT**, **INSERT**, **UPDATE**, and **DELETE**, depend on the referenced database objects. To protect data integrity, these applications must be serialized against the DDL process through a package lock. Any threads that run the application request the package lock in the share state, and any DDL processes request the package lock in the exclusive state. The amount of time that the application threads holds the package lock is controlled by the **RELEASE** bind option. The **RELEASE (COMMIT)** option allows the package to be released at the end of transaction when there are no open held cursors and the **KEEPDYNAMIC (NO)** bind option is in effect. The same package might need to be loaded again on subsequent transactions.

Conversely, packages that are bound with the **RELEASE (DEALLOCATE)** option remain with the running thread until the thread is deallocated. Threads that run with packages that use the **RELEASE (DEALLOCATE)** bind option tend to exist longer, including high-performance Database Access Threads (DBATs). As a result, although the applications perform better by using this option, they can hold the package locks for a long time, which can prevent a DDL process from obtaining the same lock. Therefore, when a database administrator must run a DDL statement, the high-performance DBATs must be stopped or deallocated.

Db2 13 further increases the likelihood of DDL success by providing the new **RELEASE_PACKAGE** keyword in the profile tables. This keyword provides you more control over managing packages that are bound with the **RELEASE (DEALLOCATE)** bind option before you attempt to make DDL changes.

5.3.1 Profile table support for the **RELEASE_PACKAGE** keyword

Db2 13 includes the following profile table enhancements to support the **RELEASE_PACKAGE** value:

- ▶ You can use the **DSN_PROFILE_TABLE** table to define profiles and associated filtering criteria to identify the interested threads, connections, or application packages.
- ▶ You can use the **DSN_PROFILE_ATTRIBUTE_TABLE** table to force Db2 to switch from using **RELEASE (DEALLOCATE)** behavior to using **RELEASE (COMMIT)** behavior when the profile starts.

For example, you can build a profile where either a local application package or any application that originates from the 191.3.5.1 IP address runs. If the package is bound with **RELEASE (DEALLOCATE)**, Db2 changes to **RELEASE (COMMIT)** behavior as soon as possible. This behavior is controlled by the new **RELEASE_PACKAGE** value in the **DSN_PROFILE_ATTRIBUTE_TABLE.KEYWORD** column.

When the **-START PROFILE** command is issued, Db2 loads the profile tables into memory for faster lookup by application threads. Db2 looks up the in-memory profile table at the following points:

- ▶ Each package load for execution in a local and remote application
- ▶ **COMMIT** and **ROLLBACK**

If a profile match exists, Db2 changes from the **RELEASE(DEALLOCATE)** behavior to the **RELEASE(COMMIT)** behavior at these points.

Note: Because the package is loaded for execution on the first SQL statement in the package, **COMMIT** statements should be issued frequently so that the profile tables can take effect.

The Db2 DDF address space must be started to use the system monitoring function of the profile tables, which includes setting special registers and global variables. Even if you use the profile tables for local applications only, you must start Db2 with the DDF subsystem parameter set to **AUTO** or **COMMAND**. Db2 issues the message DSNT761I if DDF is not started, as shown in Figure 5-3.

```
-DB2ASTA PROFILE
DSNT761I  -DB2A DSNT1RSP START PROFILE DETECTED
          DDF IS NOT LOADED. PROFILE ROWS FOR SYSTEM
          MONITORING WILL NOT BE ACTIVATED.
DSNT741I  -DB2A DSNT1SDV START PROFILE IS COMPLETED.
DSN9022I  -DB2A DSNT1STR 'START PROFILE' NORMAL COMPLETION
```

Figure 5-3 The DSNT761I message

In general, after issuing the **-START PROFILE** command, you should query the `DSN_PROFILE_HISTORY` and `DSN_PROFILE_ATTRIBUTES_HISTORY` tables to ensure that all inserted rows are accepted and that no rows were rejected. An error message DDF NOT LOADED might be issued in a `DSN_PROFILE_ATTRIBUTES_HISTORY` row if the Db2 DDF address space is not active.

You can insert the value 'RELEASE_PACKAGE' into the `DSN_PROFILE_ATTRIBUTES.KEYWORDS` column and the value 'COMMIT' into the `DSN_PROFILE_ATTRIBUTES.ATTRIBUTE1` column to force Db2 to switch from using the **RELEASE(DEALLOCATE)** behavior to using the **RELEASE(COMMIT)** behavior.

Function level V13R1M500 is required when the 'RELEASE_PACKAGE' value is specified in the `DSN_PROFILE_ATTRIBUTES` table. The row is rejected if the current function level requirement is not met when the **-START PROFILE** command is issued.

The `DSN_PROFILE_ATTRIBUTES.ATTRIBUTE2` column indicates which application type is applicable to the row. The following values are valid:

- | | |
|------|--|
| Null | This value is available to remote applications only. |
| 1 | This value is available to local applications only. |
| 2 | This value is available to both remote and local applications. |

When using the 'RELEASE_PACKAGE' value for a local application, you can identify the local application by the following filtering criteria in the DSN_PROFILE_TABLE table's columns:

- ▶ AUTHID, ROLE, or both
- ▶ COLLID, PKGNAME, or both
- ▶ One of CLIENT_APPLNAME, CLIENT_USERID, or CLIENT_WORKSTNNAME

Table 5-3 and Table 5-4 show how you might specify the new keyword in profile tables.

Table 5-3 DSN_PROFILE_TABLE

PROFILEID	COLLID	PROFILE_TIMESTAMP
99	COLLECTIONA	2021-02-28-14.55.56.78027 7

Table 5-4 DSN_PROFILE_ATTRIBUTES

PROFILEID	VALUE	ATTRIBUTE1	ATTRIBUTE2
99	RELEASE_PACKAGE	COMMIT	1

Assume that two packages are involved in this example, both of which are run in local plans only:

- ▶ COLLECTIONA.P1: Bound with the **RELEASE(COMMIT)** option.
- ▶ COLLECTIONA.P2: Bound with the **RELEASE(DEALLOCATE)** option.

Assume that thread THREAD1 is running package COLLECTIONA.P2, which was loaded with the **RELEASE(DEALLOCATE)** option. After the profile starts through the **-START PROFILE** command, the following actions occur:

- ▶ When new threads run package COLLECTIONA.P2, this package is loaded by the **RELEASE(COMMIT)** option.
- ▶ When the existing thread THREAD1, which loaded the package COLLECTIONA.P2 before starting the profiles, goes through **COMMIT** or **ROLLBACK**, this package can be released if there are no open-held cursors.

5.3.2 Optimizing DDL processes by using the RELEASE_PACKAGE keyword

A DBA can use the following process to gradually route packages that are bound with the **RELEASE(DEALLOCATE)** option to use **RELEASE(COMMIT)** behavior and retry the DDL statements to make object schema changes:

1. Query the SYSIBM.SYSPACKDEP catalog table to find the packages that depend on the object that must change. In the following example, the object is the COUNTY.VACCINATION table:

```
SELECT DCOLLID, DNAME FROM SYSIBM.SYSPACKDEP
WHERE BQUALIFIER='COUNTY' AND BNAME='VACCINATION'
```

- Assume that this query returns packages COL1.P1 and COL2.P2, and that the SYSPACKAGE.RELEASE column indicates they are bound with **RELEASE(DEALLOCATE)**.
- Assume that local threads THREAD1 and THREAD2 are running the packages COL1.P1 and COL2.P2.

2. Insert the rows that are shown in Table 5-5 and Table 5-6 into the DSN_PROFILE_TABLE and DSN_PROFILE_ATTRIBUTES tables.

Table 5-5 DSN_PROFILE_TABLE

PROFILEID	COLLID	PKGNAME
1	COL1	P1
2	COL2	P2

Table 5-6 DSN_PROFILE_ATTRIBUTES

PROFILEID	VALUE	ATTRIBUTE1	ATTRIBUTE2
1	RELEASE_PACKAGE	COMMIT	1
2	RELEASE_PACKAGE	COMMIT	2

3. Run the **-STA PROFILE** command, which loads the profile table rows into the in-memory profile table.

As a result of these steps, the following behavior changes occur:

- Existing threads THREAD1 and THREAD2 release the packages COL1.P1 and COL2.P2 when they reach a **COMMIT** or **ROLLBACK** even though they are bound with the **RELEASE(DEALLOCATE)** option, assuming that these packages do not have WITH HOLD cursors and the **KEEPDYNAMIC(YES)** bind option.
 - New threads load the packages COL1.P1 and COL2.P2 with the **RELEASE(COMMIT)** option even though they are bound with **RELEASE(DEALLOCATE)**.
4. Run a DDL statement against the COUNTY.VACCINATION table with the wanted CURRENT LOCK TIMEOUT and DEADLOCK_RESOLUTION_PRIORITY values.
 5. If the DDL statement is successful, disable or delete the rows in the DSN_PROFILE_TABLE and DSN_PROFILE_ATTRIBUTES tables and restart the profiles. Optionally, you can also issue the **-STOP PROFILE** command. Then, the packages COL1.P1 and COL2.P2 are loaded for execution with the **RELEASE(DEALLOCATE)** option.

Even though dependent packages are run with the **RELEASE(COMMIT)** option, they can still hold package locks and are quiesced only at the end of a transaction that concludes with no open held cursors. Therefore, you can retry the DDL statement until it completes successfully. With the profile started to temporarily route **RELEASE(DEALLOCATE)** behaviors, the chance that DDL can break in improves without having to bind copy a different set of packages for **RELEASE(COMMIT)**.

In a test environment, you can use the trace record IFCID 177 to monitor the **RELEASE(COMMIT)** and **RELEASE(DEALLOCATE)** behavior. This trace record is written when a package is successfully loaded for execution. If a package runs with the **RELEASE(DEALLOCATE)** behavior, even if several **COMMIT** or **ROLLBACK** statements run, only one IFCID 177 trace record is written.

5.3.3 Selectively disabling high-performance DBATs by using the **RELEASE_PACKAGE** keyword

For remote server connections, high-performance DBAT functions are enabled when at least one package with the **RELEASE(DEALLOCATE)** bind option is allocated and the **-MODIFY DDF PKGREL(BNDOPT | BNDPOOL)** command was issued previously. This approach can result in improved performance because the DBAT remains associated with the connection at transaction boundaries rather than being pooled. Using high-performance DBATs also avoids repeated package allocation and deallocation.

The same behaviors that make high-performance DBATs advantageous for performance can make package and DDL management more difficult because packages remain allocated and locks remain held for a longer duration. To allow DDL or a **BIND** operation to break in, high-performance DBATs are temporarily disabled by issuing the **-MODIFY DDF PKGREL(COMMIT)** command, which causes Db2 to follow the rules of the **RELEASE(COMMIT)** bind option, which are applied the next time that the package is loaded. However, the results of the **-MODIFY DDF PKGREL** command are applied at the Db2 subsystem level.

Another mechanism to disable high-performance DBATs is by using the **PKGREL_COMMIT** subsystem parameter with a value of **YES**. However, this approach requires all threads that are running the dependent packages to reach a **COMMIT** point during the DDL's lock waiting interval. Also, this approach does not prevent new threads from loading packages with the **RELEASE(DEALLOCATE)** option.

You can use the profile **RELEASE_PACKAGE** value to selectively disable high-performance DBATs when DDL or a **BIND** operation must break into objects that are held by only a few packages. This approach is an alternative to using the **-MODIFY DDF PKGREL(COMMIT)** command, which affects the entire Db2 subsystem.

You can define profiles with the appropriate filtering criteria by using the **RELEASE_PACKAGE** value to change packages that are bound with the **RELEASE(DEALLOCATE)** bind option. The **RELEASE_PACKAGE** profile is evaluated and applied during package load, commit, and rollback. At the next commit or rollback for the server thread, if all packages that are allocated by that thread no longer have the **RELEASE(DEALLOCATE)** behavior (either through the original **BIND** option or by a profile downgrade), then the connection goes inactive and that thread is pooled. The result is that high-performance DBAT behavior is selectively disabled for that thread.

5.4 Migrating distributed applications

All IBM Data Server Driver versions continue to be supported by Db2 13. To support continuous delivery of new features in Db2 13, the IBM Data Server set of packages are used to control the **APPLCOMPAT** level by an application. These packages enable you to upgrade continuously Db2 function levels without impacting your applications in production.

A common approach for controlling the introduction of new Db2 features but still allowing applications to use new features is to create two sets of Data Server Driver packages: One set that locks down the minimum set of features in use by applications, and another set that is used by applications that require new SQL or application-related functions that are introduced in function levels.

5.4.1 Determining the Db2 function level and application compatibility level

When a connection to Db2 is established, the product ID and function level are returned in the Db2-provided SQL communication area (SQLCA).

The APPLCOMPAT level is not known until the first package is loaded to process the initial SQL statement. During the processing of the SQL statement, Db2 ensures that the Data Server Driver level supports the APPLCOMPAT setting; otherwise, SQLCODE -30025 is returned.

5.4.2 Controlling the application compatibility level

If an application is not migrated to use Data Server Package 11.1 Fix Pack 2 or later, you can continue to use packages by using the NULLID collection with no impact to your applications as you migrate Db2 to newer function levels.

When an application requires an SQL feature that is delivered at a Db2 APPLCOMPAT level later than V12R1M500, first you must ensure that the Data Server Driver level that is used by the application supports the required APPLCOMPAT level.

For example, when the application requires the usage of the SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY built-in global variable, the application must have the APPLCOMPAT level V13R1M501. The following two items are needed:

- ▶ The Data Server Driver packages on Db2 for z/OS, which are bound with the APPLCOMPAT V13R1M501 option
- ▶ Data Server Driver 11.1 Modification 2 Fix Pack 2 or later

5.4.3 Creating driver packages to use new functions

To create a new set of the Data Server Driver packages with an APPLCOMPAT bind option in a new collection, you can use either the client **Db2Binder** command or the DSNTIJLC migration job that is provided by Db2.

The following example uses the **Db2Binder** command to create the IBM Data Server Driver for JDBC and SQLJ packages under the collection NULLID_NFM with the bind option **APPLCOMPAT V13R1M500**:

```
java com.ibm.db2.jcc.DB2Binder -url jdbc:db2://sys1.svl.ibm.com:5021/STLEC1 -user sysadm -password XXXXXXXX -bindoptions "APPLCOMPAT V13R1M500" -action REPLACE
```

Alternatively, you can use the DSNTIJLC job to copy the driver packages from the NULLID collection to the NULLID_NFM collection, and set **APPLCOMPAT V13R1M500**.

5.4.4 Enabling driver packages to use new functions

For an application that requires a higher APPLCOMPAT level than the level that is supported by the NULLID set of packages, you must change the collection of the Data Server Driver packages that are used by the application from NULLID to NULLID_NFM by using either of the following methods:

- ▶ Explicitly set the package set to the NULLID_NFM collection by using the client property `db2.jcc.currentPackageSet`.
- ▶ Populate the `SYSIBM.DSN_PROFILE_TABLE` and `SYSIBM.DSN_PROFILE_ATTRIBUTES` tables so that Db2 seamlessly changes the packages that are used by the application. The following example creates a profile for an application with a `CLIENT_APPLNAME` value of `ACCTG_APP501`. You can also use other filters that identify applications that require new functions.

```
INSERT INTO SYSIBM.DSN_PROFILE_TABLE (PROFILEID, CLIENT_APPLNAME,
PROFILE_ENABLED) VALUES (1002, 'ACCTG_APP501', 'Y');
INSERT INTO SYSIBM.DSN_PROFILE_ATTRIBUTES (PROFILEID, KEYWORDS, ATTRIBUTE1)
VALUES (1002, 'SPECIAL_REGISTER', 'SET CURRENT PACKAGE PATH=NULLID_NFM')
```

Issue the **-START PROFILE** command to load the profile table settings.

The `ACCTG_APP501` application can now successfully connect and use the Data Server Driver packages in collection `NULLID_NFM`. It can verify the current `APPLCOMPAT` setting and the collection that is being used by issuing the following statement in the application:

```
SELECT CURRENT APPLICATION COMPATIBILITY AS APPLCOMPAT,
GETVARIABLE('SYSIBM.PACKAGE_SCHEMA') AS COLLID FROM SYSIBM.SYSDUMMY1;
```

The query returns the following results:

```
APPLCOMPAT  COLLID
-----
V13R1M501   NULLID_NFM
```

Other applications can continue to connect by using the default collection `NULLID` set of packages.

Notes:

- ▶ You can use the `PKGNAME` column value as a filtering category for profile tables with the `SPECIAL_REGISTER` value. However, using only `PKGNAME` is not recommended as a filtering category for profile tables for IBM Data Server packages because special register values are set through the profile table only once at the beginning of a connection (along with the first non-**SET** SQL statement in the application). The package that is used for the first non-**SET** SQL statement can vary depending on the statement, sections in use, access type, and so on. Therefore, `PKGNAME` as a filtering category cannot provide consistent behavior.
- ▶ When you use Data Server Driver 11.1 Modification 2 Fix Pack 2 or later, the application does not need to set the `clientAppCompat` property regardless of the **APPLCOMPAT** bind option value of the driver packages.

5.5 Summary

In the area of application management in the context of SQL changes, Db2 13 can control lock timeout and deadlock resolution at the application level and avoid changing local or remote applications by using profile tables to set the CURRENT LOCK TIMEOUT special register and the SYSIBMADM. DEADLOCK_RESOLUTION_PRIORITY global variable. These enhancements are part of the effort to broaden the useful functions of profile tables to simplify how you manage your local applications, much like the capabilities for managing remote applications that were provided in earlier Db2 releases. Additionally, the process that is used by distributed applications that originate from clients to connect to Db2 for z/OS and to use new SQL enhancements was significantly simplified.



SQL Data Insights

SQL Data Insights (SQL DI) is an optionally installable feature that is included with Db2 13 for z/OS, which brings artificial intelligence (AI) capability to your Db2 engine.

This chapter describes the following topics:

- ▶ Overview of SQL Data Insights
- ▶ Installing and configuring SQL Data Insights
- ▶ Enabling an AI query
- ▶ Running AI queries

6.1 Overview of SQL Data Insights

SQL DI is an optionally installable feature of Db2 that brings AI capability directly to your Db2 data without requiring costly extract, transform, and load (ETL) processes to move the data off the IBM Z mainframe or hiring a team of data scientists to design machine learning models. You can choose a Db2 table or view to enable the AI query function and then use SQL to issue semantic queries against the data.

After you install the SQL DI GUI component that runs on z/OS, you can browse and select from a set of Db2 tables and views and enable them for AI query. Enabling AI query on a table or view trains an unsupervised neural network machine learning model that is stored in Db2.

After a Db2 table or view is enabled for AI query, you can run AI semantic queries against the table or view by using SQL. You can run the SQL through the built-in SQL editor, an application program, or an SQL tool such as SPUFI or QMF. AI semantic queries use the semantic similarity, clustering, and analogy functions that are built into the Db2 engine.

6.1.1 SQL Data Insights use cases

The semantic query capability that is provided by SQL DI can be applied to use cases across many industries. The AI semantic functions provide the kinds of queries that are described in Table 6-1.

Table 6-1 SQL Data Insights semantic query types

Db2 built-in AI semantic function	Semantic query type
AI_SIMILARITY	Similarity query
AI_SIMILARITY	Dissimilarity query
AI_SEMANTIC_CLUSTER	Inductive reasoning semantic clustering query
AI_ANALOGY	Inductive reasoning analogy query

You can use similarity queries to find groups of similar entities to decide on market segmentation or find a group of customers that behaves similarly to other groups, which have applications in the retail, finance, and insurance industries.

You can use dissimilarity queries to find outliers from the norm, which has applications in financial anomaly detection and fraud detection.

You can use semantic clustering queries to form a cluster of entities to test whether an extra entity belongs in the cluster. You can use semantic clustering in many contexts where similarity or dissimilarity queries are used as a broader test of similarity or dissimilarity to multiple entities.

You can use analogy queries to determine whether a relationship between a pair of entities applies to a second pair of entities, which has applications in retail, such as to determine whether a customer has a preference for a product and whether other customers have the same degree of preference for other products.

6.1.2 Synergy with IBM Z

SQL DI uses IBM Z acceleration for both machine learning model training and for the computation of the semantic functions for SQL queries.

IBM Z Deep Neural Network library (zDNN) is a component of z/OS that provides the interface to use IBM Z acceleration. Both the SQL DI machine learning model training and AI SQL functions use zDNN.

On IBM z14® or newer models, the enhanced mathematical library OpenBLAS can accelerate the SQL DI AI computations. The z/OS component IBM Z AI Optimization (zAIO) library provides the interface to determine whether OpenBLAS acceleration can be used.

SQL DI is supported on any hardware and software level that is supported by Db2 13. However, OpenBLAS is not available on IBM Z with processors earlier than the ones that are in z14. So, while z14, IBM z15, or IBM z16 are not requirements to run SQL DI, only systems with those processors can provide the AI acceleration.

zDNN, zAIO, and the IBM Z AI Data Embedding Library, which provides the services to train the SQL DI machine learning models, are available as PTFs for z/OS 2.4 and 2.5. You do not need to configure these libraries after they are installed. SQL DI training and SQL semantic functions automatically use them when they are available.

Portions of the machine learning model training process and the Db2 SQL queries that use the semantic functions are eligible to be run on the IBM Z Integrated Information Processor (zIIP).

6.2 Installing and configuring SQL Data Insights

The information in this section is not a substitution for the installation and configuration instructions that are described in IBM Documentation. Instead, it is intended to help you understand in broad terms what you need to know about SQL DI planning, installation, and configuration. For more information about the specific steps that are required to complete your installation and configuration, see at [Running AI queries with SQL Data Insights](#).

The following categories of software are required to run SQL DI:

- ▶ Db2 13 for z/OS
- ▶ z/OS 2.4 or 2.5 with PTFs for the zDNN library, zAIO library, Z AI Data Embedding Library, OpenBLAS, and IBM 64-Bit SDK for Java
- ▶ SQL DI that contains the GUI component and model training services

The z/OS requirements must be installed by your system programmer, but require no specific configuration. SQL DI must be installed by your system programmer and then configured. Configuration of SQL DI is described at a high level in 6.2.3, “Configuring SQL Data Insights” on page 85.

6.2.1 Determining the system environment for SQL Data Insights

SQL DI runs on z/OS. You can run it on the same z/OS logical partition (LPAR) on which Db2 is installed or on a separate LPAR. Here are some factors to consider when deciding where you want to run SQL DI:

- ▶ SQL DI has an embedded Apache Spark cluster that is used for training the machine learning model during the AI query enablement process. Depending on the size of the table being enabled, the machine learning model training can be a CPU-, memory-, and disk-intensive process. For more information, see [Preparing SQL Data Insights installation](#).
- ▶ SQL DI requires a UNIX System Services environment for its configuration, and it requires direct-access storage disks (DASDs) to be allocated in zFS file systems. You must be familiar with UNIX System Services to complete the configuration. For more information, see [Installing and configuring SQL Data Insights](#).
- ▶ SQL DI requires a block of contiguous Internet Protocol network ports to be allocated and, if applicable, reserved. You must work with your network team to allocate and configure the ports.
- ▶ Because SQL DI uses the secure sockets layer (SSL) to ensure security among web components, you must configure a keystore and key ring in z/OS through RACF or your equivalent security product. You must work with your security team to configure the keystore and key ring, and if they are needed, to create an SSL certificate for use by SQL DI.

6.2.2 Configuring Db2 for SQL Data Insights

The z/OS LPAR on which Db2 runs must have z/OS 2.4 or 2.5 with the prerequisite maintenance that installs the following AI libraries:

- ▶ The IBM Z AI Data Embedding library
- ▶ The zAIO library
- ▶ The zDNN library

SQL DI and the Apache Spark cluster communicate with Db2 through Type 4 Java Database Connectivity (JDBC). As a result, you must configure the Distributed Data Facility (DDF) in Db2 and bind the Java packages for JDBC with application compatibility (APPLCOMPAT) V13R1M500 or higher. This configuration enables you to run queries by using the semantic functions through the GUI.

SQL DI uses DRDA fast load (zLOAD) and the DSNUTILU stored procedure. You must also bind packages DSNUT121 and DSNUTILU in Db2.

Sample job DSNTIJAI

You must create a set of objects in Db2 that are required by SQL DI. The DSNxxxx.SDSNSAMP (DSNTIJAI) sample job is provided to create these objects and grant permissions to SQL DI users. With the proper administrative authority, you can customize and submit DSNTIJAI to perform the following actions:

- ▶ Create the SQL DI pseudo-catalog. The pseudo-catalog consists of a database DSNAIDB1, table spaces, tables, and indexes that are used to maintain the state of SQL DI objects and models.
- ▶ Create the stored procedures that are used by SQL DI. Your SQL DI users do not have to issue SQL directly against the SQL DI pseudo-catalog or the Db2 catalog. All such accesses occur through stored procedures that are created in DSNTIJAI, and the EXECUTE permission on the stored procedures is granted to users of the GUI.
- ▶ Create the database DSNAIDB2, which is initially empty, to hold tables that contain trained machine learning models.
- ▶ Grant all necessary permissions on all the objects that are created by DSNTIJAI to SQL DI users.

Before submitting the job, you must customize it to run in your environment by following instructions in the job itself. You are prompted to decide about choosing storage groups and default buffer pool assignments for the objects that are listed above. The model database can grow large if large tables are enabled for AI query.

For discussion and illustration in the rest of this chapter, assume that a Db2 administrator with ID SYSADM submits the DSNTIJAI job, and the administrator has SYSADM privileges in Db2. Also, user SQLDIUSR is a nonadministrative user but is granted all necessary permissions to use SQL DI after it is set up.

Sample job DSNTIJAV

The DSNxxxx.SDSNSAMP data set includes another sample job, DSNTIJAV. The job is not required for installation, but you can run it to create sample table DSNAIDB.CHURN for validating the SQL DI installation. The DSNAIDB.CHURN table, which is referred as the CHURN table, contains approximately 7,000 rows of data from a fictitious telecommunications company that is used later in this chapter to demonstrate a customer retention analysis scenario.

6.2.3 Configuring SQL Data Insights

After the installation of the prerequisites and SQL DI, you can configure SQL DI.

Before you begin, make sure that you review the planning considerations and write down the following information that you need for configuration:

- ▶ The zFS directory that contains your SQL DI instance. Make sure that sufficient space is allocated.
- ▶ Network ports that are allocated for SQL DI and the embedded Apache Spark cluster.
- ▶ Keystore and key ring information.

Also, collect the following JDBC connection properties information. You need this information to connect SQL DI to Db2.

- ▶ TCP/IP hostname or IP address where Db2 runs
- ▶ Db2 location name
- ▶ Db2 DDF TCP/IP port
- ▶ Your Db2 user ID and password

Creating an SQL Data Insights instance

By default, SQL DI is installed into a zFS directory. You must configure the installed instance by using the UNIX System Services shell.

To change directories to where SQL DI is installed, issue the following command:

```
cd /usr/lpp/IBM/db2sqldi/v1r1
```

To create an SQL DI instance, issue the following command:

```
./sqldi.sh create
```

This command invokes the interactive `sqldi.sh` script and prompts you for the information that you collected earlier. Enter the information, as shown in Figure 6-1.

```
$ ./sqldi.sh create
Enter a directory where SQL Data Insights configuration files and logs can be stored: /aidb/sqldihome
Bash version is 4.3
Installing SQL Data Insights ...
Enter the IP address or hostname for SQL Data Insights: utec275.vmec.svl.ibm.com
Enter the port number for SQL Data Insights or press <enter> to use 15001:

SQL Data Insights requires one of the following keystore types:
1) JCERACFKS (for managing RACF certificates and keys)
2) JCECCARACFKS (for managing RACF certificates and keys and exploiting hardware crypto)
Select your keystore type: 1
Enter the keyring name: MYKEYRING
Enter the keyring owner: SYSADM
Enter certificate label: MYKEY
Verifying the following keyring information ...
Configuring Spark cluster ...
Configuring new Spark runtime for SQL Data Insights ...
Enter the IP address or host name of your Spark master: utec275.vmec.svl.ibm.com
Enter the port number of your Spark master or press <enter> to use default port 7077:
Enter the port number of your Spark master REST API or press <enter> to use default port 6066:
Enter the port number of your Spark Web UI or press <enter> to use default port 8080:
Enter the port number of your Spark Worker or press <enter> to use system-assigned port:
Random port will be used.
Enter the port number of your Spark Worker Web UI or press <enter> to use default port 8081:
Enter the port number of your Spark Driver or press <enter> to use system-assigned port:
Random port will be used.
Enter port number of your Spark Block Manager or press <enter> to use system-assigned port:
Random port will be used.
Enter driver-specific port for the Spark Block Manager to listen on or press <enter> to use random port as the default:
Random port will be used.
Enter maximum number of retries when binding to a port or press <enter> to use default 16:
Starting Spark master...
Spark master started successfully
Starting Spark worker...
Spark worker started successfully
You have successfully configured and started Spark. Check the parameters used for Spark under /aidb/sqldihome/spark/conf.
Spark cluster is successfully configured.
Congratulations! You have successfully installed SQL Data Insights.
```

Figure 6-1 The `sqldi.sh` script

After this script completes successfully, SQL DI is configured, started, and ready for use. Note the URL that you will use to access the SQL DI user interface (UI).

Connecting to Db2

To create a connection to your Db2, open a web browser and go to the URL that is shown in the `sqldi.sh` script. You must log in to SQL DI by using your ID and password. Your ID must belong to the RACF group SQLDIGRP on the LPAR on which SQL DI is running.

When you log in for the first time, your only option is to add a connection to your Db2. Choose this option and enter the requested information, as shown in Figure 6-2 on page 87.

SQL Data Insights

Connections /

Add connection

Connection overview

Name

Description (optional)

Connection details

Host name or IP address

Port

Location ⓘ

JDBC properties (optional) ⓘ

Db2 special registers (optional) ⓘ

Certificates ⓘ

Port enabled for SSL connections

SSL Certificate (optional) ⓘ

Enter or copy and paste your certificate in PEM format. The PEM file must start with ----
 -BEGIN CERTIFICATE----- and end with -----
 END CERTIFICATE-----

Credentials

Username

Password

Figure 6-2 Add connection window

The relationship between SQL DI and Db2 is one-to-many. You can create more connections to other Db2 instances, either stand-alone systems or members of a data-sharing group. Each Db2 must be properly configured as described 6.2.2, “Configuring Db2 for SQL Data Insights” on page 84 and have all the required pseudo-catalog, model database, and stored procedures.

Configuring the SQL Data Insights settings

Review the system-wide settings for SQL DI by going to the upper right of the UI and clicking the gear icon to open the Settings page, which is shown in Figure 6-3.

Settings

Spark
Allocate system resources for executing Spark jobs

Driver core ①: 1
Driver memory (GB) ①: 2
Executor core ①: 1
Executor memory (GB) ①: 2

CPU threads
Specify threads for preprocessing data and training models

Base10 cluster threads ①: 2
Training threads ①: 12

Db2 load utility
Customize the Db2 LOAD utility control statement for loading trained models

Utility control statement ①

```
TEMPLATE SORTIN DSN &JO..&ST..SORTIN.T&TIME. UNIT SYSALLDA SPACE(100,100) CYL DISP(NEW,DELETE,DELETE)
TEMPLATE SORTOUT DSN &JO..&ST..SORTOUT.T&TIME. UNIT SYSALLDA SPACE(100,100) CYL DISP(NEW,DELETE,DELETE)
TEMPLATE MAP1 DSN &JO..&ST..MAP1.T&TIME. UNIT SYSALLDA SPACE(100,100) CYL DISP(NEW,DELETE,DELETE)
LOAD DATA INDDN SYSCLIEN WORKDDN(SORTIN,SORTOUT) MAPDDN(MAP1) REPLACE LOG(NO) REUSE NOCOPYPEND
UNICODE CCSID(1208)
```

Restore default

AI query
Specify the maximum number of rows to load for a query result set

Number of rows: 1000

Cancel Save

Figure 6-3 SQL Data Insights settings window

You can change the settings for tuning and resource allocation. Review the Db2 load utility control statement that will be used later when you enable AI query on a table. SQL DI runs the load utility to load the contents of the model table. Ensure that the data sets are sized and defined correctly for your system. Customize the control statements if needed.

6.3 Enabling an AI query

Enabling an AI query on a table or view makes that object ready for use with the AI query functions. The enabling process involves training a machine learning model and loading the model into Db2. In the SQL DI UI, select an object that you want to enable, choose the columns, and then click **Enable**.

The examples in this chapter demonstrate how to enable AI query on the CHURN sample table. You can follow the same process to enable your own tables and views for AI query.

6.3.1 Adding AI objects

Before a table or a view can be enabled for AI query, you must add it to the list of AI objects for a connection. Select the **List AI objects** option from the action menu next to your connection, as shown in Figure 6-4. If the list is empty, click **Add object** to add your first object.

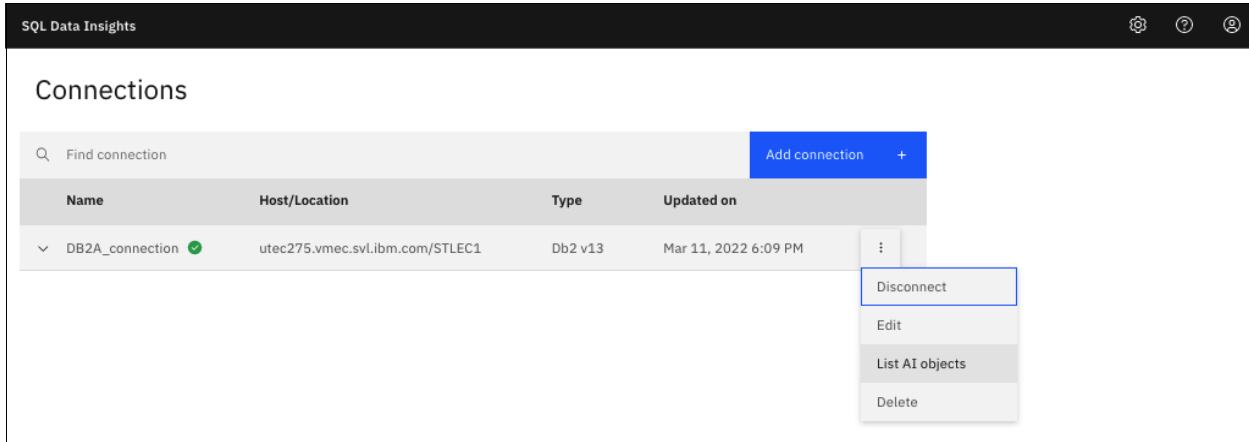


Figure 6-4 List AI objects menu

You can select the CHURN table by choosing its schema DSNAIDB from the drop-down list and then choosing CHURN by name. Check the box next to CHURN and click **Add object** to add DSNAIDB.CHURN to the list of AI objects, as shown in Figure 6-5.

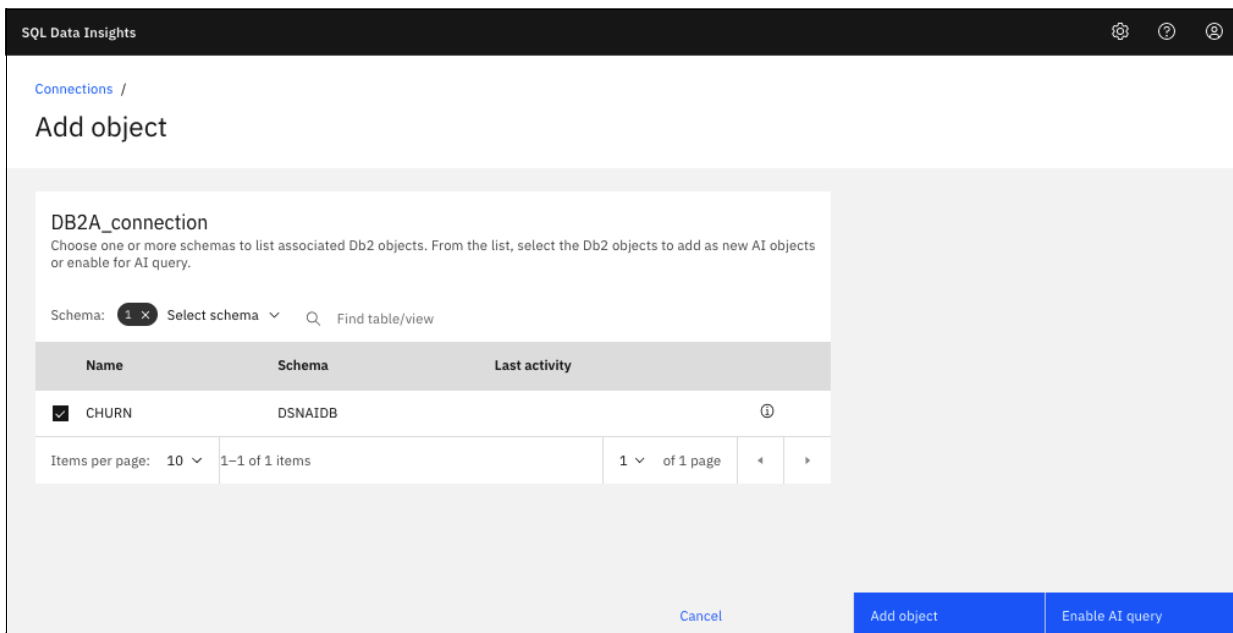


Figure 6-5 Add object window

An AI object that is identified but not yet enabled is in the Created state, as shown in Figure 6-6. Objects in this state can be analyzed to a limited extent and then enabled.

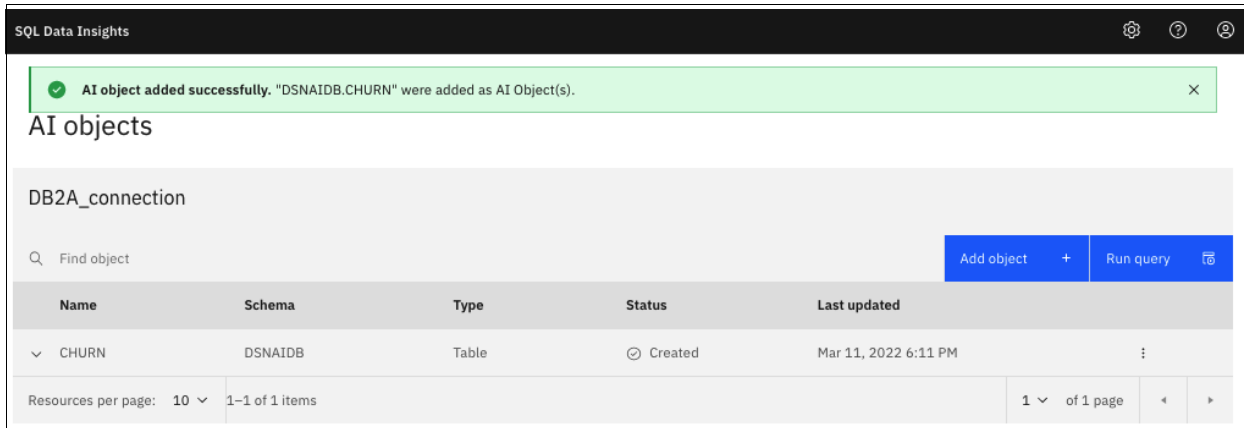


Figure 6-6 The CHURN object in the Created state

Your ID must have the SELECT privilege on the table that you are choosing. The privilege is not needed when you add the table to the list of AI objects, but it is required when you eventually enable the object for AI query. The privilege is needed to read the data for model training. The SELECT privilege for DSNAIDB.CHURN is granted to PUBLIC in the DSNTIJAI batch job, so it is not a concern for the sample table.

6.3.2 Selecting columns for inclusion

When you are ready to enable an object for AI query, select the object and choose the **Enable AI query** option from the action menu, as shown in Figure 6-7.

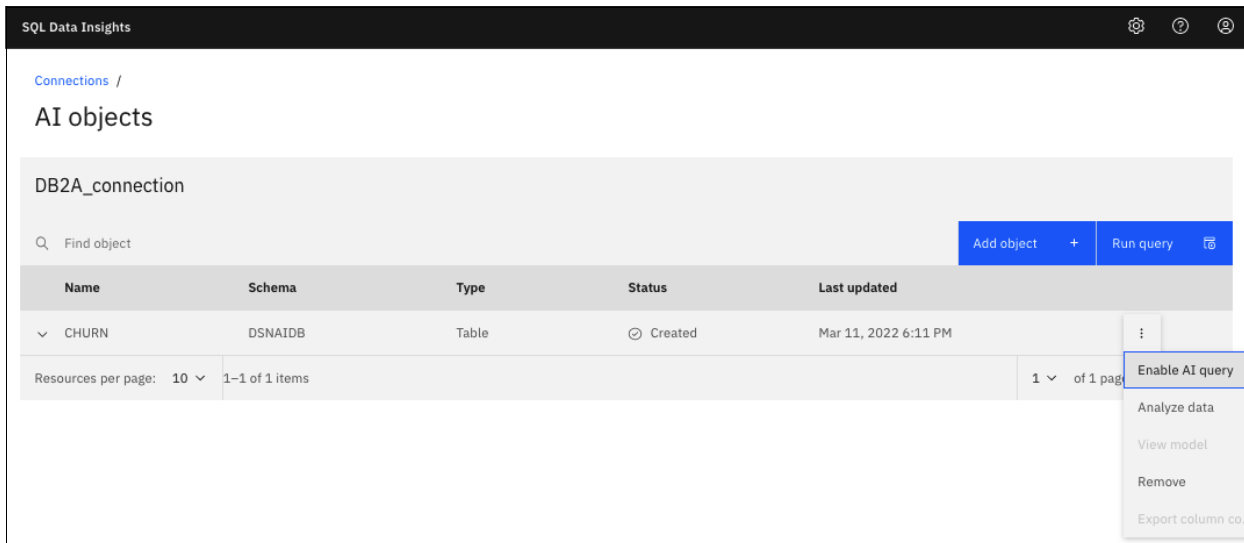


Figure 6-7 Enable AI query menu

From the list of columns, choose the ones that you want to include in the object model. When you run AI queries against the object, only the selected columns are used in the queries. However, if you know that some columns will never be used in a query and their values will not influence the kinds of semantic questions that will be asked of the data, omit those columns, which can result in a smaller model, less disk storage, and faster training time.

Because the sample CHURN table is not excessively large, you can select all the columns in the table.

You also must choose an SQL DI data type for each column. By default, columns with a numeric SQL type (such as integers, decimals, floats, or decfloats) are assigned the SQL DI numeric type, and nonnumeric columns are assigned the SQL DI categorical type.

Data of the SQL DI categorical type are discrete values that are treated separately if they are not exactly equal. A column with a numeric SQL type can be given the SQL DI categorical type if the values are intended to be treated individually. A decimal column that holds an account ID or a social security number is a good example of when you might want to choose an SQL DI categorical type for a column of a numeric SQL type.

Data of an SQL numeric type are continuous values where values that are “close” to one another are considered together. Only columns that have numeric SQL types can be chosen to have a numeric SQL DI type. A numeric column that holds the price of a good is an example of a numeric SQL DI type, where close values such as \$9.99 and \$10.00 might be considered together.

A third SQL DI data type is key. Assign the SQL DI key type to columns that represent the entire row in an AI query. If your table or view has a single-column unique key, it can be chosen as an SQL DI type key. For the CHURN data, assign the CustomerId column of the SQL DI data type and keep the default data type assignments for all others, as shown in Figure 6-8. Later, this chapter shows how you can use the CustomerId column to compare one customer to another one, that is, how to compare one entire row against another entire row.

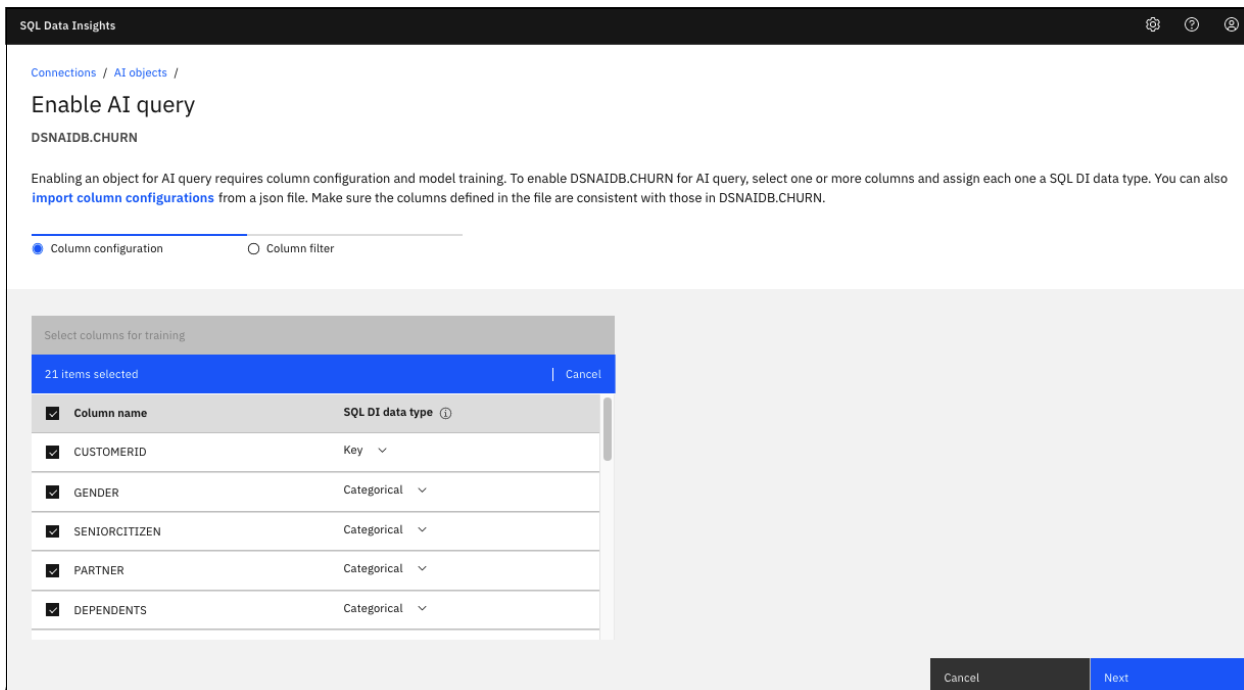


Figure 6-8 SQL DI data type

6.3.3 Specifying filter values

After you select the columns, you can identify values to be omitted from the columns and eventually the model, which are called *filter values*. For example, some tables have data that contains the value “N/A” indicating that there is no applicable value for this column, or a column with a numeric SQL type might contain -1, which is treated as an ignorable value. By designating such values as filter values, they do not become part of the model, and ultimately your AI queries will not consider “N/A” or -1 to be similar to other, nonfiltered values in your data.

You can specify filter values for a particular column or for the entire table. In the CHURN example, no filter values are specified, so you don’t need to specify anything on this window, as shown in Figure 6-9.

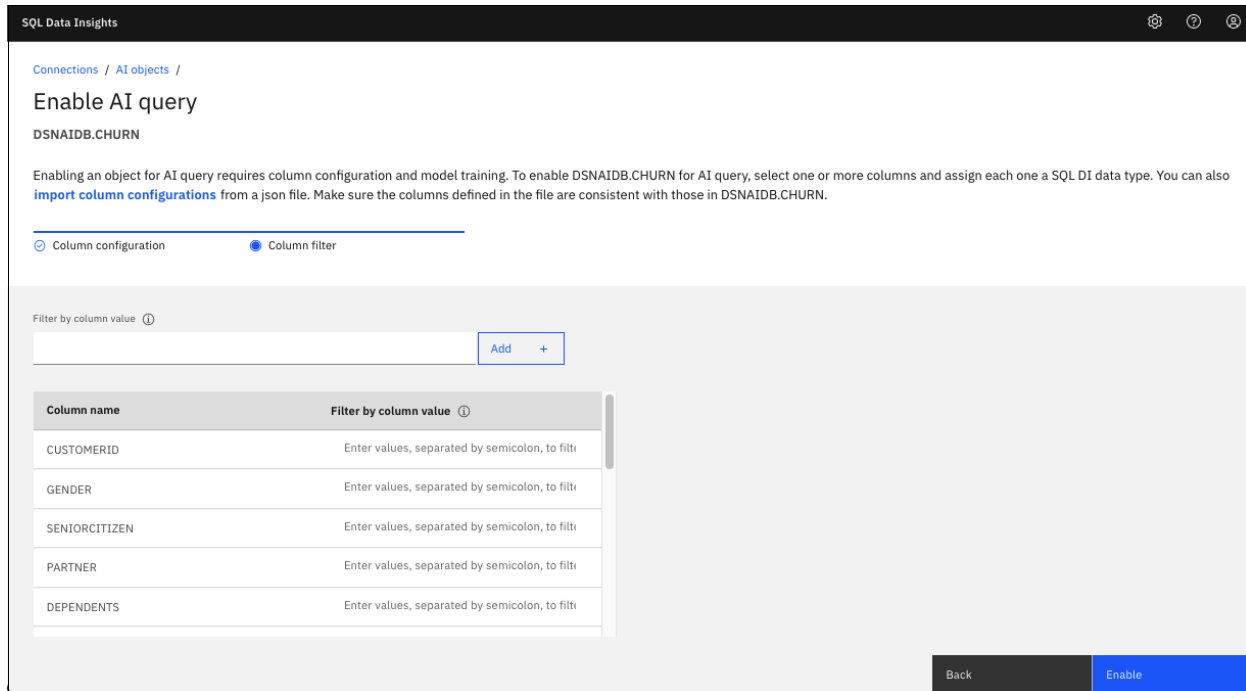


Figure 6-9 No filter values for CHURN

6.3.4 Processing the enablement

Clicking **Enable** begins the process of creating the model in Db2. You do not have to do anything more until the process completes. After you click **Enable**, your browser is brought back to the AI objects page where you can monitor the status, which is now Enabling.

In the Enabling phrase, SQL DI reads from DSNAIDB.CHURN into the embedded Apache Spark cluster. The connection to Db2 uses the authority of user SQLDIUSR that has the SELECT privilege on the table.

SQL DI analyzes the data from the table and decomposes it into a “vocabulary”, which is a list of all the unique words that appear in the table. The relationships between each pair of words in the vocabulary are built into the model that is a collection of numerical vectors that are associated with the words in the vocabulary.

To store the model in Db2, a table space, a table, and an index are created in the DSNAIDB2 database. When you ran the DSNTIJAI sample job to configure Db2, you granted authority to SQLDIUSR to create table spaces and tables in the DSNAIDB2 database to hold the trained model. The vocabulary words and their related vectors are stored in the model table. You do not need to do anything with this table directly, but later when you write and run AI queries, SQL DI reads from the model table and uses the data to compute the answers to your queries.

When the AI query enablement process completes successfully, the status becomes Enabled, and you are now ready to run AI queries against the table, as shown in Figure 6-10.

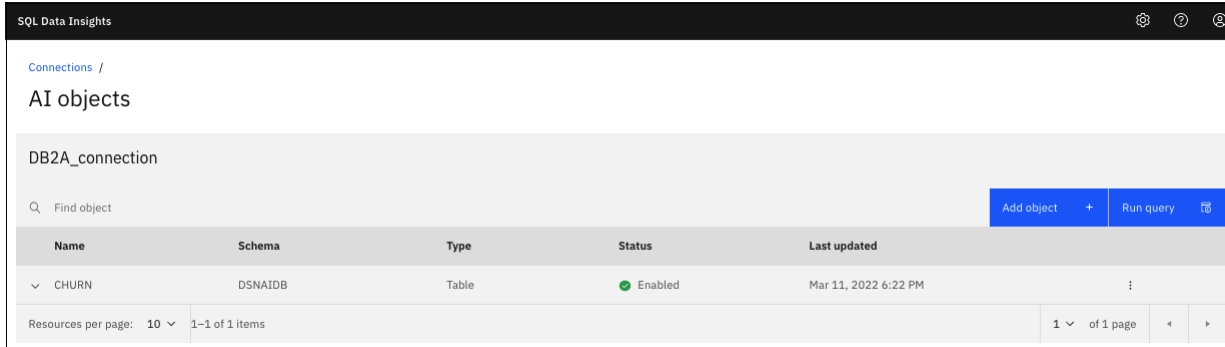


Figure 6-10 Enabled status

6.4 Running AI queries

For the SQL DI feature, Db2 13 introduces three new built-in SQL scalar functions: **AI_SIMILARITY**, **AI_SEMANTIC_CLUSTER**, and **AI_ANALOGY**. You can include these functions in your SQL statements to ask semantic questions about your data.

To enable and use these new functions, you must activate function level V13M1R500 or higher. You must also bind the packages for the program that issues the semantic query SQL statement (that is, the packages for SPUFI) with APPLCOMPAT V13M1R500 or higher.

You can use the three functions in SQL that allow scalar functions. For example, you can include them in a **SELECT** statement or as part of a **WHERE** clause predicate. You can also make the source of a **SET** clause of an **UPDATE** statement or one of the values in an **INSERT** statement. However, because the new functions are nondeterministic, they are not allowed in SQL contexts where nondeterministic functions are disallowed, such as in the definition of an index-on-expression or a materialized query table.

6.4.1 Using the AI_SIMILARITY function

AI_SIMILARITY is one of the three new AI functions for SQL DI. To understand what this new function is and how you can use it, let us look at some quick examples.

A conceptual example of AI_SIMILARITY

The **AI_SIMILARITY** function determines how similar two “entities” (words from the table or view) are within a “context” (a column of the table or view), and the result of the function is a similarity “score”.

Example 6-1 shows how the **AI_SIMILARITY** function is used.

Example 6-1 A simple example to illustrate AI_SIMILARITY

```
AI_SIMILARITY('APPLE', 'RASPBERRY' USING MODEL COLUMN FRUIT)
```

```
AI_SIMILARITY('BLACKBERRY', 'RASPBERRY' USING MODEL COLUMN FRUIT)
```

This example includes two instances of the **AI_SIMILARITY** function:

- ▶ The two entities that are compared are 'APPLE' and 'RASPBERRY'.
- ▶ the two entities that are compared are 'BLACKBERRY' and 'RASPBERRY'.

The context of the similarity comparison for both instances is the column that is named "FRUIT" that is specified in the **USING MODEL COLUMN FRUIT** clause. The words 'APPLE', 'RASPBERRY', and 'BLACKBERRY' are words in the FRUIT column, which implies that relationships among those words within that column are represented in the model. These words can appear in other columns in the same table, but in those other contexts (columns), the words might mean something different.

The result of **AI_SIMILARITY** is a score, which is a floating point number -1.0 - 1.0. A score of 1.0 means that the two entities are similar (or the same), and a score of -1.0 means that the two entities are dissimilar. The function that compares 'BLACKBERRY' and 'RASPBERRY' returns a similarity score of 0.86 because both are berries and close in size. The function that compares 'APPLE' and 'RASPBERRY' returns a similarity score of 0.45, which indicates that there is some similarity, but the two are not as similar as in the other example. Even though apples and raspberries are fruits, they are not both berries, and they are not similar in size.

An example of AI_SIMILARITY that uses the CHURN table

For a more realistic example that uses the CHURN table, consider a problem where Jane, a business analyst at a large telecommunications company, identifies a customer who closed their account and she wants to find out whether other customers share similar characteristics. If she can identify such customers, she might be able to take some preventive actions that forestall the customers from leaving the business.

Assume that the customer who left has ID '3668-QPYBK'. Example 6-2 shows the SQL that the analyst can issue to find other similar customers.

Example 6-2 Finding customers similar to a customer who closed their account

```
SELECT
  AI_SIMILARITY(X.customerID,'3668-QPYBK') AS SimilarityScore, X.*
FROM DSNAINDB.CHURN X
WHERE X.customerID <> '3668-QPYBK'
ORDER BY SimilarityScore DESC
FETCH FIRST 10 ROWS ONLY
```

In the result set that is shown in Figure 6-11 on page 95, you can see that most of the results are similar to but not the same as '3668-QPYBK'. The values for the categorical columns are mostly the same or similar. The values for the numeric columns TENURE, MONTHLYCHARGES, and TOTALCHARGES are all numerically close to the ones for the '3668-QPYBK' row.

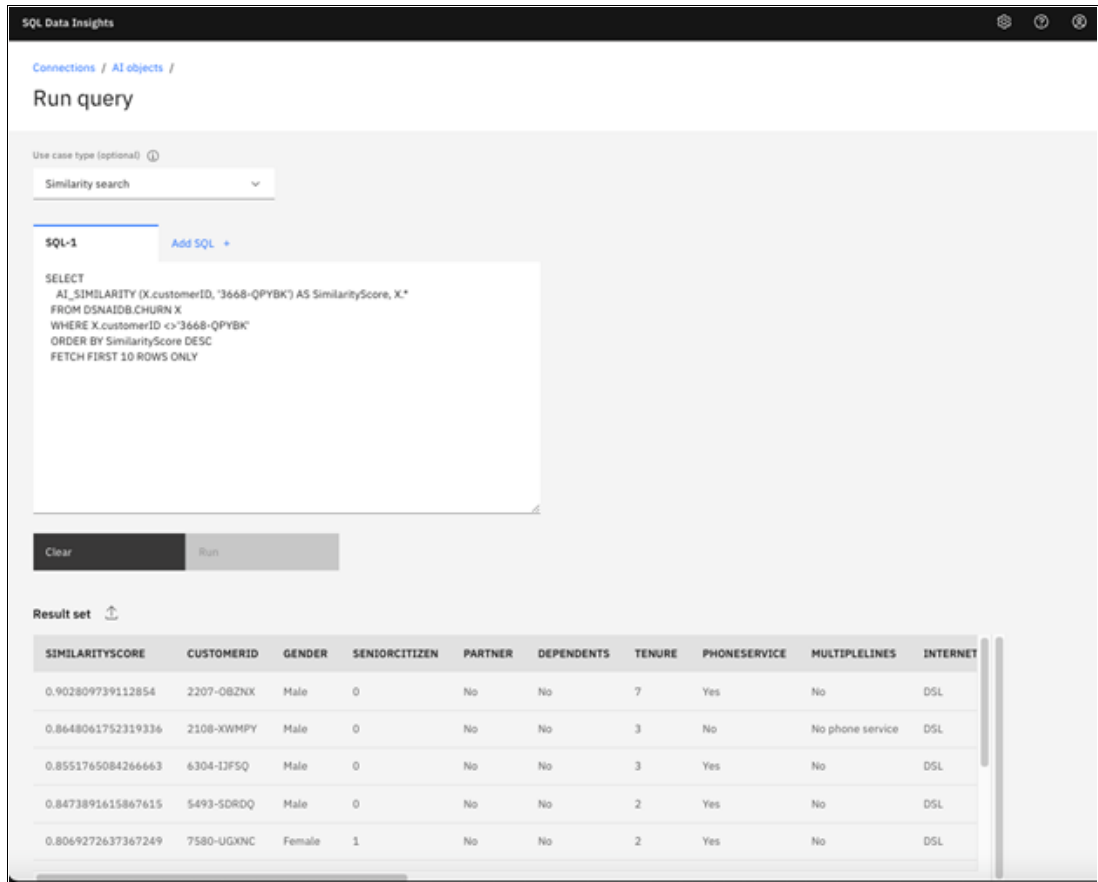


Figure 6-11 Run query results set

If you enabled AI query on the DSNAIDB.CHURN table in your own Db2 system and tried this query, your similarity scores and the exact ordering of the results might be slightly different from the results in this book. The reason for that is that the model training algorithm employs randomization. As a result, you can get slightly different sets of vectors each time the model is trained. When AI queries are run, the vectors are slightly different, and the results are slightly different too. That said, it is to be expected that even though the precise ordering of results can differ across model trainings, entities that are similar to one another continue to be similar to one another across trainings. Likewise, entities that are dissimilar continue to be dissimilar even though the exact similarity scores might change.

Example 6-2 on page 94 showed a use of AI_SIMILARITY that compared the customerID column, which was given a key SQL DI type when AI was enabled for DSNAIDB.CHURN, which means that when the similarity score is computed, the entire row for '3668-QPYBK' was compared to the values in the entire rows for the rest of the table. This behavior is useful because for this purpose it is important to consider the customerID to be representative of the entire customer record. "An example of AI_SIMILARITY that examines specific attributes" on page 97 examines the similarity of a nonkey column.

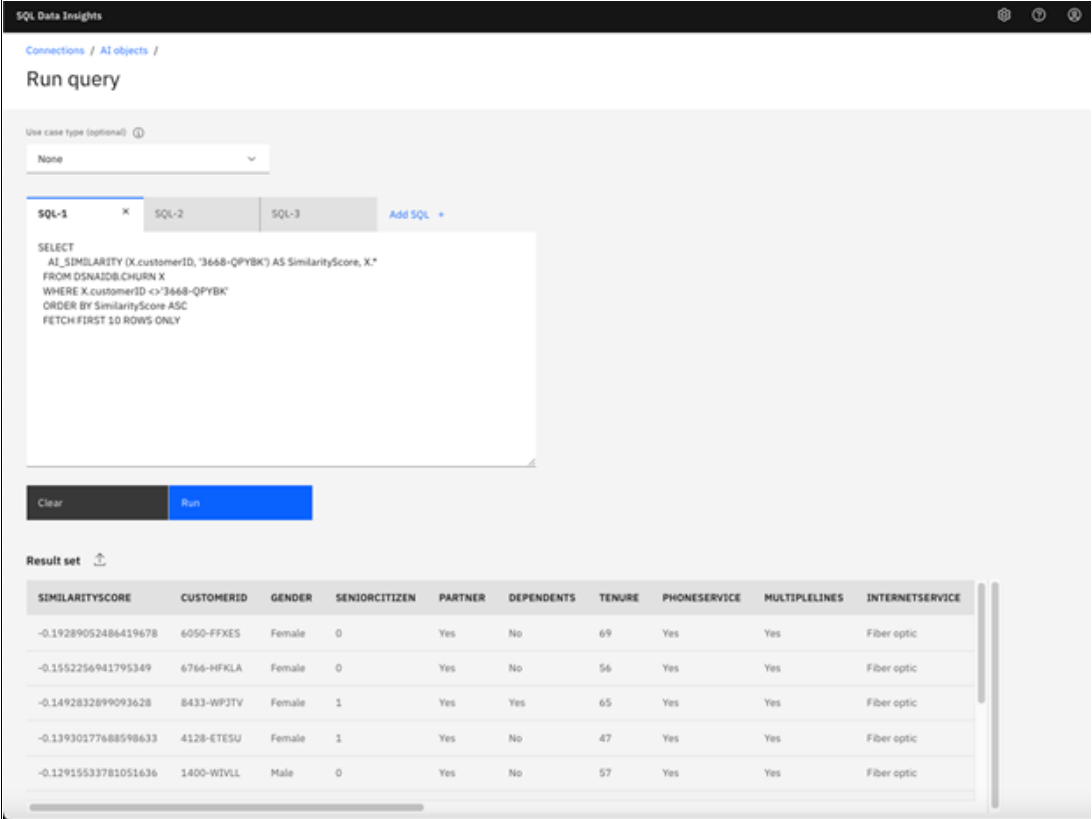
A dissimilarity query

You can use the `AI_SIMILARITY` function to run a dissimilarity query. Assume that analyst Jane knows that the customer with ID '3668-QPYBK' represents a customer that left the business. She tries to find customers who are least likely to leave and are therefore the most loyal customers by running the SQL that is shown in Example 6-3.

Example 6-3 Finding customers who are the most loyal by determining which are least similar to a customer who closed their account

```
SELECT
  AI_SIMILARITY (X.customerID, '3668-QPYBK') AS SimilarityScore, X.*
FROM_DSNAIDB.CHURN X
WHERE X.customerID <>'3668-QPYBK'
ORDER BY SimilarityScore ASC
FETCH FIRST 10 ROWS ONLY
```

The result set for this SQL query is shown in Figure 6-12.



The screenshot shows the SQL Data Insights interface. At the top, it says "SQL Data Insights" and "Connections / AI objects /". Below that is a "Run query" section with a "Use case type (optional)" dropdown set to "None". There are three tabs for SQL queries: "SQL-1", "SQL-2", and "SQL-3", with "SQL-1" selected. The SQL query is displayed in a text area: `SELECT AI_SIMILARITY (X.customerID, '3668-QPYBK') AS SimilarityScore, X.* FROM_DSNAIDB.CHURN X WHERE X.customerID <>'3668-QPYBK' ORDER BY SimilarityScore ASC FETCH FIRST 10 ROWS ONLY`. Below the query area are "Clear" and "Run" buttons. The "Result set" section shows a table with 10 columns: SIMILARITYSCORE, CUSTOMERID, GENDER, SENIORCITIZEN, PARTNER, DEPENDENTS, TENURE, PHONESERVICE, MULTIPLELINES, and INTERNETSERVICE. The results are sorted by SIMILARITYSCORE in ascending order.

SIMILARITYSCORE	CUSTOMERID	GENDER	SENIORCITIZEN	PARTNER	DEPENDENTS	TENURE	PHONESERVICE	MULTIPLELINES	INTERNETSERVICE
-0.19289052486419678	6050-FFXES	Female	0	Yes	No	69	Yes	Yes	Fiber optic
-0.1552256941795349	6766-HFKLA	Female	0	Yes	No	56	Yes	Yes	Fiber optic
-0.1492832899093628	8433-WP2TV	Female	1	Yes	Yes	65	Yes	Yes	Fiber optic
-0.13930177688598633	4128-ETESU	Female	1	Yes	No	47	Yes	Yes	Fiber optic
-0.12915533781051636	1400-WTVLL	Male	0	Yes	No	57	Yes	Yes	Fiber optic

Figure 6-12 Run query results set

The SQL for this query is nearly identical to the SQL in Example 6-2 on page 94. The only difference is in the ordering that is specified by the `ORDER BY SimilarityScore ASC` clause. Example 6-3 orders by ascending `SimilarityScore`, from smallest to largest, and Example 6-2 on page 94 orders by descending `SimilarityScore` from largest to smallest. The ordering in Example 6-2 on page 94 puts the most similar scores at the front of the result, and the ordering in Example 6-3 puts the most dissimilar scores, or those items with the lowest similarity scores, at the front of the result.

An example of AI_SIMILARITY that examines specific attributes

The analyst Jane understands the attributes that are important in determining whether customers leave the business. Perhaps she is trying to determine whether there is a correlation between the customers' decisions and their payment methods.

This query that is shown in Example 6-4 compares 'YES' in the context of the CHURN column to each value of PAYMENTMETHOD in the context of the PAYMENTMETHOD column. **DISTINCT** is added to the **SELECT** statement to avoid receiving multiple rows with the same values.

Example 6-4 Showing how similar each payment method is to CHURN='YES'

```
SELECT DISTINCT AI_SIMILARITY('YES' USING MODEL COLUMN CHURN,  
                             PAYMENTMETHOD),  
               PAYMENTMETHOD  
FROM DSNADB.CHURN  
ORDER BY 1 DESC
```

The results, which are shown in Figure 6-13, show that the payment method that is most similar to CHURN 'YES' is 'Electronic check'. The least similar payment method is 'Mailed check'. However, looking across the scores, the similarity score for 'Electronic check' is about 0.43, which is not very different from the similarity score of 0.38 for 'Mailed check'. The scores indicate that despite an ordered similarity ranking among the different payment methods from the top to the bottom, the differences are not significant, and payment methods probably do not affect whether a customer leaves.

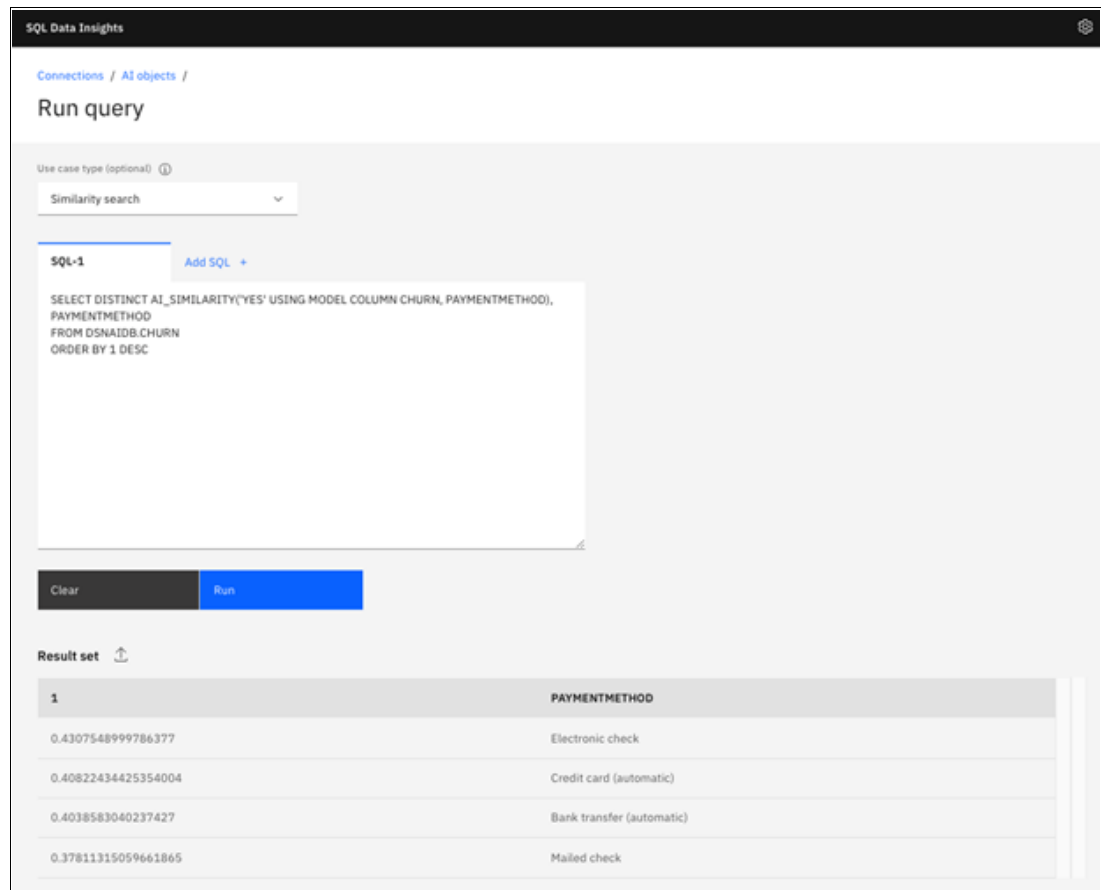


Figure 6-13 Run query results set

In Example 6-3 on page 96, both context columns, CHURN and PAYMENTMETHOD, are categorical, which means that neither of them has the SQL DI type key, and the specific attributes are being compared to each one, which is in contrast with Example 6-2 on page 94, which evaluates similarity by using columns with SQL DI type key and therefore compares entire rows to each other.

6.4.2 Using the AI_SEMANTIC_CLUSTER function

AI_SEMANTIC_CLUSTER is another of the three new AI functions in Db2 for SQL DI. The purpose of **AI_SEMANTIC_CLUSTER** is to form a cluster of up to three entities and produce a score that indicates how strongly an extra entity belongs to that cluster. The following examples illustrate what **AI_SEMANTIC_CLUSTER** is and how you can use it.

A conceptual example of AI_SEMANTIC_CLUSTER

For **AI_SEMANTIC_CLUSTER**, you can specify up to three clustering arguments to form a group of entities. Then, you specify an extra member argument to test for inclusion into that group. The function returns a clustering score, which is a floating point number -1.0 - 1.0. A score of 1.0 indicates a strong fit between the member argument and the cluster, and a score of -1.0 indicates a poor fit. All the arguments of the **AI_SEMANTIC_CLUSTER** function must share the same column, which means that they must refer to the same model column.

Example 6-5 and Example 6-6 illustrate how **AI_SEMANTIC_CLUSTER** works.

Example 6-5 A simple example to illustrate AI_SEMANTIC_CLUSTER

```
AI_SEMANTIC_CLUSTER('APPLE' USING MODEL COLUMN FRUIT,  
                    'RASPBERRY', 'BLACKBERRY', 'BLUEBERRY')
```

Example 6-6 Another simple example to illustrate AI_SEMANTIC_CLUSTER

```
AI_SEMANTIC_CLUSTER('STRAWBERRY' USING MODEL COLUMN FRUIT,  
                    'RASPBERRY', 'BLACKBERRY', 'BLUEBERRY')
```

The function in Example 6-5 creates a cluster of the last three arguments: **'RASPBERRY'**, **'BLACKBERRY'**, and **'BLUEBERRY'**. The first argument, **'APPLE'**, is a fruit, but it might not fit well in that cluster because it is not a berry, and it is a different size than other members of the cluster. The score that is computed for this function is 0.23.

The function in Example 6-6 creates the same cluster with its last three arguments: **'RASPBERRY'**, **'BLACKBERRY'**, and **'BLUEBERRY'**. The first argument **'STRAWBERRY'** is tested for inclusion in that group, and its score is 0.82, which is much higher than the score for **'APPLE'**. **'STRAWBERRY'** is much more strongly related to the group than **'APPLE'** because it is a berry and closer in size.

An example of AI_SEMANTIC_CLUSTER that uses the CHURN table

A common usage pattern emerges from starting with an initial seed, which is an example row that exhibits the behavior in question. An AI query that uses **AI_SIMILARITY** produces an ordered list of other values that are similar to the seed. The top three or a representative three values from the **AI_SIMILARITY** query are used to form clustering arguments to **AI_SEMANTIC_CLUSTER**, which can further refine the results.

Assume that analyst Jane starts with a seed that represents the customer with ID '3668-QPYBK'. She ran the query in Example 6-2 on page 94. From the results of Example 6-2 on page 94, she can take '3668-QPYBK' and the next two most similar customers who also churned, who have the IDs '2207-OBZNX' and '2108-XWMPY', to form a cluster. Then, she can run another AI query to determine which customers belong to this cluster, as shown in Example 6-7.

Example 6-7 Using AI_SEMANTIC_CLUSTER with a cluster that is formed from a customer who closed their account

```
SELECT AI_SEMANTIC_CLUSTER
      (X.CUSTOMERID,
       '3668-QPYBK', '2207-OBZNX', '2108-XWMPY')
      CLUSTERSCORE, X.*
FROM DSNAIDB.CHURN X
ORDER BY CLUSTERSCORE DESC
FETCH FIRST 10 ROWS ONLY
```

It should be no surprise that the first three results that fit best in this semantic cluster are the three customers that form the group, as shown in Figure 6-14. You can see that several of the subsequent results also have high clustering scores, which indicate a strong fit in this group.

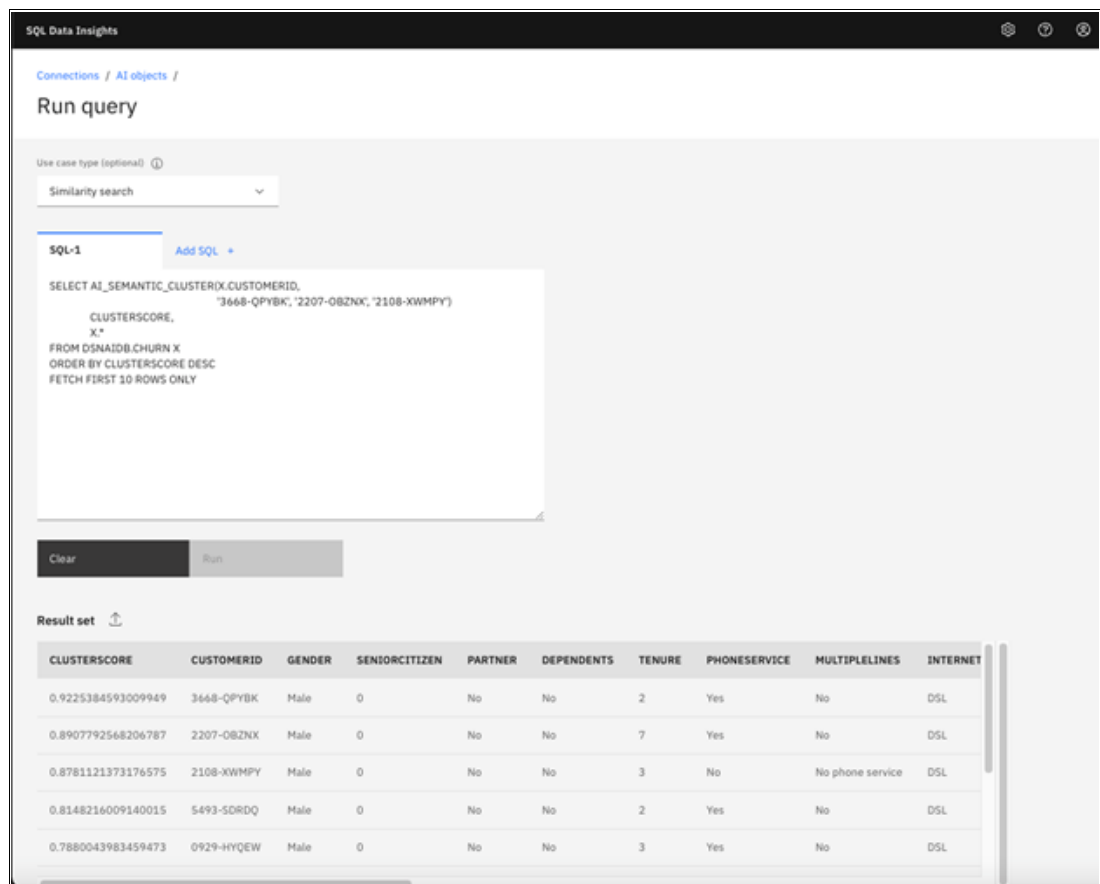


Figure 6-14 Run query results set

6.4.3 Using the AI_ANALOGY function

AI_ANALOGY is the last of the three new functions in Db2 for SQL DI. With this function, you can compare relationships across two pairs of entities. The relationships form an analogy. The result of the function is an analogy score, which is a floating point number. Unlike **AI_SEMANTIC_CLUSTER** and **AI_SIMILARITY**, the range of the result of **AI_ANALOGY** is not bounded by -1.0 and 1.0. Instead, the results are simply that a higher score is a better analogy than a lower score, and the results really make sense only when one is compared to the other.

A conceptual example of AI_ANALOGY

To understand how **AI_ANALOGY** works, consider the simple analogy that strawberry is to red as lemon is to yellow. To determine whether this analogy is a good one, think about the relationship between the first two entities “strawberry” and “red”. Naturally, you might think that a strawberry is colored red. But does the same relationship apply to the second pair of entities “lemon” and “yellow”? As shown in Example 6-8, a lemon is colored yellow. So, the relationship holds, and this example is one of a good analogy.

Example 6-8 A good conceptual example of AI_ANALOGY

```
AI_ANALOGY('STRAWBERRY' USING MODEL COLUMN FRUIT, 'RED',  
           'LEMON', 'YELLOW')
```

The first pair of arguments to the **AI_ANALOGY** function in Example 6-8 specifies the two entities to establish the relationship. The relationship is applied to the second pair of arguments to determine whether the analogy is good. In this case, the analogy is good, so it has a higher analogy score of 0.87.

The conceptual example in Example 6-9 represents a poor analogy. The first two arguments are the same as in Example 6-8, so the relationship between them is the same, which is the relationship that is based on color. When the color-based relationship is applied to the second pair of arguments, ‘BLUEBERRY’ and ‘ORANGE’, it does not hold. Blueberries are not colored orange, so it is a poor analogy, and it has a much lower analogy score of -0.24. By comparing the two analogy scores, you can see that the score that is produced by the **AI_ANALOGY** function in Example 6-8 indicates a better analogy than the score that is produced by the same function in Example 6-9.

Example 6-9 A poor conceptual example of AI_ANALOGY

```
AI_ANALOGY('STRAWBERRY' USING MODEL COLUMN FRUIT, 'RED',  
           'BLUEBERRY', 'ORANGE')
```

A more realistic example of AI_ANALOGY

Supposed that analyst Jane is trying to understand the relationships between contracts and the type of internet service that is subscribed. She observes that customers with month-to-month contracts predominantly subscribe to fiber optic internet service and asks the question about the kind of internet service that is subscribed by 2-year contract payers. She uses an **AI_ANALOGY** query to compare and determine the relationship between the month-to-month contract and fiber optic internet and apply that relationship to 2-year contracts and other internet service options, as shown in Example 6-10 on page 101.

Example 6-10 An AI_ANALOGY query that examines relationships between length of contract and internet service subscriptions

```
SELECT DISTINCT
    AI_ANALOGY('Month-to-month' USING MODEL COLUMN CONTRACT,
              'Fiber optic' USING MODEL COLUMN
              INTERNETSERVICE,
              'Two year',
              INTERNETSERVICE) AS ANALOGY_SCORE,
    X.INTERNETSERVICE
FROM DSNAIDB.CHURN X
WHERE X.INTERNETSERVICE <> 'Fiber optic'
ORDER BY ANALOGY_SCORE DESC
```

Jane can see from the query results that DSL is a good analogy, and no internet service is not as good of an analogy, as shown in Figure 6-15. The query does not specify the relationship, which happens to be that fiber optic internet service is the most common internet service that is subscribed by those customers with month-to-month contracts. The relationship is inferred through the model and tested against other pairs of entities.

The screenshot shows the SQL Data Insights interface. At the top, it says "SQL Data Insights" and "Connections / AI objects /". Below that is a "Run query" section. There is a dropdown menu for "Use case type (optional)" set to "None". Below the dropdown are three tabs labeled "SQL-1", "SQL-2", and "SQL-3", with "SQL-1" selected. To the right of the tabs is a button labeled "Add SQL +". The "SQL-1" tab is open, showing the SQL query from Example 6-10. Below the query editor are two buttons: "Clear" and "Run". Below the "Run" button is a "Result set" section. It contains a table with two columns: "ANALOGY_SCORE" and "INTERNETSERVICE". The table has two rows of data.

ANALOGY_SCORE	INTERNETSERVICE
0.8413964921922206	DSL
0.6485916530516833	No

Figure 6-15 Run query results set

6.4.4 Interpreting AI query results

SQL DI provides information about the data that can help you understand and interpret the results of your AI queries.

To see the data analysis, go back to the list of AI objects and choose the **Analyze data option** from the action menu, as shown in Figure 6-16.

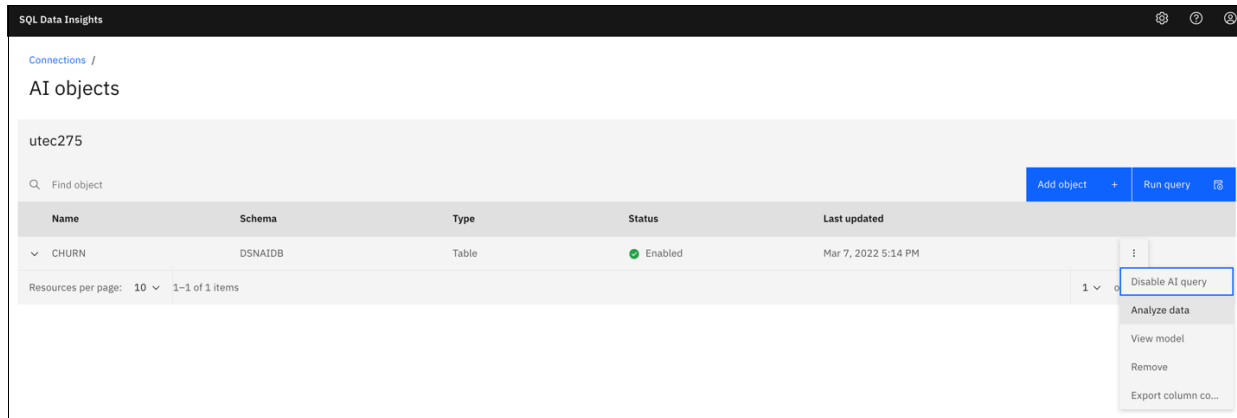


Figure 6-16 Analyze data option window

The Analyze data option window has three tabs:

- ▶ The **Object details** tab recaps your selection of SQL DI types for the model.
- ▶ The **Data statistics** tab shows statistical information about the data in the table, including value counts, frequencies, averages, maxima, minima, and standard deviations for numeric columns.
- ▶ The **Column influence** tab is where you can understand the contributions of columns to the results of an AI query. The tab shows a table of two values: the influence score and the discriminator score for each column.

The influence score is a calculation of data presence. A column with high prevalence of null or missing data has a low influence score, which means that such a column cannot influence the result of a query as much as a column that has relatively few or no null or missing values. The CHURN table does not have any null values, so the influence score is 1.0 for every column, which means that each of the columns can influence the results to the same degree. If you had specified filter values from the CHURN table, those filtered values are also treated as missing when the influence score is computed.

Because influence scores are not interesting for the CHURN table, select the **Discriminator** checkbox to display only the discriminator scores, as shown in Figure 6-17 on page 103.

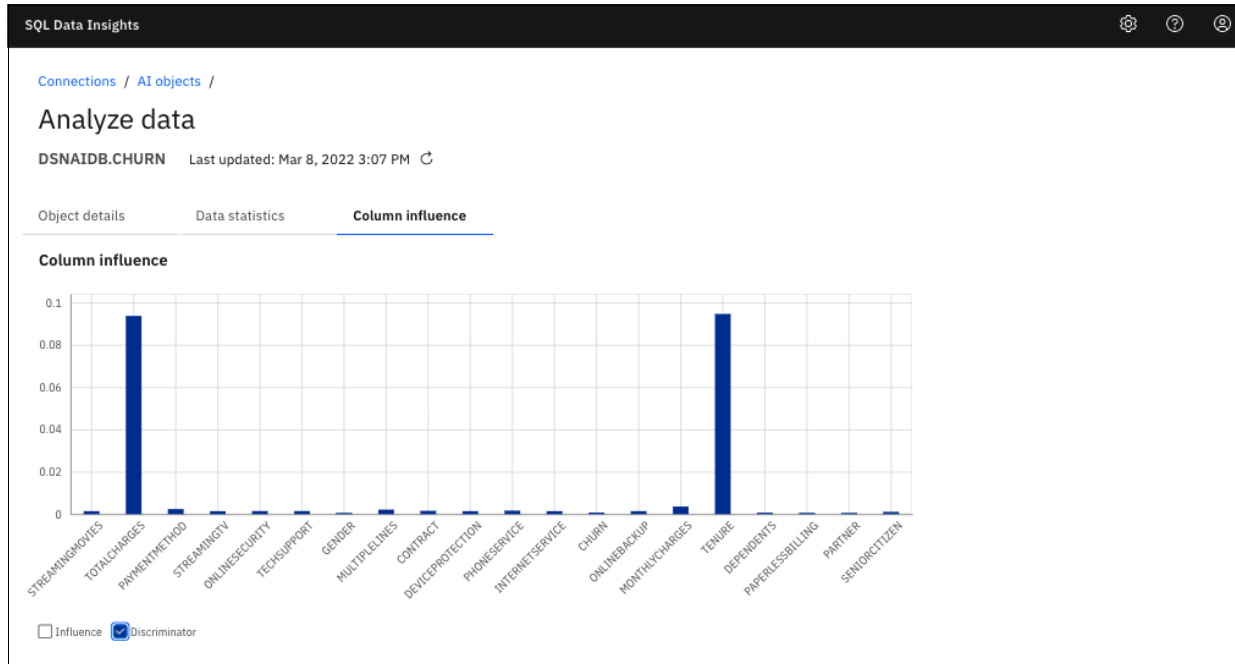


Figure 6-17 Discriminator checkbox

The discriminator score indicates how varied the values in each column are. Columns with very few distinct values, such as the gender column that contains only the values “Female” and “Male”, have a low discriminator score. A low discriminator score diminishes the ability to distinguish between co-occurrences of entities within rows and therefore reduces that column’s ability to contribute to the similarity calculation. Conversely, the Tenure and TotalCharges columns have very high discriminator scores because they have many distinct values and can contribute more to the similarity calculation.

6.5 Summary

SQL DI brings AI capability directly to your Db2 data in-place by extending Db2 SQL with new built-in functions that you apply to use cases across multiple industries, which are custom-built for IBM Z servers that can use multiple forms of hardware and software AI acceleration. After you install SQL DI and all its prerequisites, you can use the UI to create AI objects from Db2 tables or views, and then enable the objects for AI query, which trains an unsupervised machine learning model. You can then run AI queries against the objects either through the SQL editor or your own application programs.

Depending on the objective of your queries, you can select the **AI_SIMILARITY**, **AI_SEMANTIC_CLUSTER**, or **AI_ANALOGY** function. The **AI_SIMILARITY** function produces a similarity score describing how similar two entities are. The **AI_SEMANTIC_CLUSTER** function produces a score describing how closely an entity belongs within a group of up to three other entities. The **AI_ANALOGY** function examines a relationship between a pair of entities and determines the strength of the same relationship between a second pair of entities.

By using the power of these new SQL functions with SQL DI, you can now easily glean new insights from your Db2 data without having to move the data off-platform or acquiring a large set of machine learning and AI skills.



Security

The key security enhancements in Db2 13 for z/OS focus on reducing processing costs for RACF or other external security products; better performance for Db2 access checks, increased flexibility, and improved productivity for administrators deploying packages; and automated compliance data collection to support the IBM Z Security and Compliance Center solution.

This chapter describes the following topics:

- ▶ Reduced Security Manager contention for Db2 access
- ▶ Improved flexibility in managing plans, packages, and SQL routines
- ▶ Db2 support for continuous compliance
- ▶ Read-only archived key Db2 support

7.1 Reduced Security Manager contention for Db2 access

Db2 13 includes many improvements that reduce contention for Security Manager resources when accessing Db2.

7.1.1 Plan authorization cache

When you use the access control authorization exit for access control (DSNX@XAC), Db2 does not cache the successful authorization checks of the EXECUTE privilege on the application's plan by default. This behavior can cause performance issues during authorization checks when the plans run.

An enhancement that was introduced in Db2 13 improves performance for external security users by leveraging the caching in Db2. Because authorization checks that are done with Db2 native security already leverage this cache, this feature provides consistent behavior between Db2 native and external security controls.

Plan authorization cache also was enhanced to use a smarter algorithm for better management of authorization IDs and roles that can be cached per plan.

Plan authorization cache for external security

When you use the access control authorization exit (DSNX@XAC) for authorization checks, Db2 13 caches the successful EXECUTE of a plan when the access is allowed based on the profile in the RACF resource class for the plan (for example, MDSNPN or GDSNPN). The results are not cached if access is allowed due to administrative authority, such as DATAACCESS or SYSADM.

The plan authorization cache is enabled when the existing **AUTHEXIT_CACHEREFRESH** system parameter is set to ALL and the z/OS release is version 2.5 or later.

Before Db2 13, when the **AUTHEXIT_CACHEREFRESH** subsystem parameter is set to ALL and the access control authorization exit is active, Db2 listens to the type 62, type 71, and type 79 ENF signals from RACF for user profile or resource access changes and refreshes the Db2 cache entries as needed. Db2 13 is enhanced to also refresh the plan authorization cache entries for a particular plan when the user profile or resource access is changed in RACF and **SETROPTS RACLIST REFRESH** is issued for the RACF resource class for the plan, such as MDSNPN.

AUTHCACH system parameter

The plan authorization cache size is set to 4K, and the **AUTHCACH** subsystem parameter was removed to simplify cache management.

You can still specify the **CACHESIZE** bind option on the **BIND PLAN** subcommand to control the plan authorization cache size at the plan level.

Statistics record counters

The existing Db2 statistics trace counter QTAUCCH is incremented on a successful plan authorization check for both Db2 native and external security controls when the cache is used.

The following two new counters were added. These new counters are set regardless of whether you use Db2 native or external security controls.

- ▶ QTAUCNOT tracks the number of times that the plan EXECUTE privilege was checked and not found in the plan authorization cache.
- ▶ QTAUCOW1 tracks the number of times that Db2 had to swap out an authorization ID in the plan authorization cache due to lack of space.

Smarter algorithm for authorization IDs

The algorithm that is used to calculate the number of auth IDs per plan cache is enhanced to potentially store more IDs in the cache. This enhancement reduces the number of IDs that are swapped in and out of the plan cache.

7.1.2 Global authentication cache

Db2 can cache the user credentials in the global authentication cache when remote TCP/IP connections are established successfully. Before Db2 13, cache entries for connections that use credentials other than multi-factor authentication (MFA) are purged every 3 minutes.

The global authentication cache is enhanced to handle the cache entries based on the timestamp for all connections, when the `AUTHEXIT_CACHEREFRESH` subsystem parameter is set to ALL. The specific cache entry is invalidated when the user permission is changed in RACF.

7.2 Improved flexibility in managing plans, packages, and SQL routines

Before Db2 13, a Db2 for z/OS DBA who manages plans, packages, and SQL routines cannot control the owner of those plans, packages, and SQL routine packages when using both a role and an authorization ID. For example, if a DBA is binding a package under a trusted context with a role as the object owner, they cannot bind the package to be owned by an authorization ID. To achieve this goal, the DBA must ask the Db2 security administrator to disable the trusted context and grant the authority to the DBA. Then, the DBA can create the package with an authorization ID as the owner. After the package is created, the security administrator must enable the trusted context to activate role-based security for the DBA. The productivity and responsiveness of the DBA is negatively affected because of their dependency on the security administrator. This dependency hinders organizations from migrating to role-based security to achieve better compliance.

Db2 13 provides the flexibility for a DBA to control the ownership of plans, packages, and SQL routine packages by using the DBA role without depending on the security administrator. New capability was added to identify the type of owner for plans and packages.

7.2.1 Owner type support for packages of SQL procedures and functions

With application compatibility (APPLCOMPAT) level V13R1M500, the type of package owner can be specified for the following SQL statements:

- ▶ `CREATE/ALTER PROCEDURE` (native SQL)
- ▶ `CREATE/ALTER FUNCTION` (compiled SQL scalar)

New syntax for CREATE and ALTER statements

In **CREATE** and **ALTER** statements, new syntax is added to the option **PACKAGE OWNER** so that the owner type can be specified as **ROLE** or **USER**. The following diagram shows this new syntax:

```
>-----PACKAGE OWNER---authorization-name---+-----+-----<
                                                '-AS-+-ROLE-+-'
                                                '-USER-'
```

The **AS ROLE** parameter specifies that *<authorization-name>* is a role that exists on the server. The **AS USER** parameter specifies that *<authorization-name>* is an authorization ID. The default is **AS ROLE** if the process is running in a trusted context and the **ROLE AS OBJECT OWNER AND QUALIFIER** option is in effect. Otherwise, the default is the SQL authorization ID of the process.

Examples of using the owner type in CREATE and ALTER statements

If a DBA wants to create a native SQL procedure under a trusted context with a role as the object owner, they can specify **AS USER** in the **PACKAGE OWNER** option of the **CREATE** statement. For example:

```
CREATE PROCEDURE MYSCHEMA.MYPROCEDURE ()
  VERSION V1 LANGUAGE SQL VALIDATE BIND
  PACKAGE OWNER USER01 AS USER
  BEGIN
    UPDATE T1 SET C1 = C1 + 10;
    RETURN;
  END
```

The package owner is **USER01**, which is an authorization ID. Later, if the DBA must change the package owner to a role, they can specify the new owner by using **PACKAGE OWNER** with **AS ROLE** in the **ALTER** statement:

```
ALTER PROCEDURE MYSCHEMA.MYPROCEDURE
  PACKAGE OWNER DEVELOPER_ROLE AS ROLE
```

The package owner is changed to **DEVELOPER_ROLE**, which is a defined role.

Conversely, if a DBA wants to create a procedure without a trusted context, they can specify either a role or an authorization ID as the owner by using **AS ROLE** or **AS USER** in the **PACKAGE OWNER** option.

7.2.2 Owner type support for packages and plans

With function level V13R1M500, a new option, **OWNERTYPE**, can be specified in the following **BIND** and **REBIND** commands:

- ▶ **BIND/REBIND PACKAGE**
- ▶ **BIND/REBIND PLAN**
- ▶ **BIND SERVICE**

New syntax for BIND and REBIND commands

The new **OWNERTYPE** option is added to the **BIND** and **REBIND** commands. The **OWNERTYPE** option can be specified as **ROLE** or **USER**. The following diagram shows this new syntax:

```
>--+-----+>
  '-OWNER-(-authorization-id)---+-----+-----'
                                     '-OWNERTYPE -(-+-ROLE-+-)-'
                                     '-USER-'
```

When the command is issued in a trusted context with a role as an object owner, the default value of **OWNERTYPE** is **ROLE**. Otherwise, the default value of **OWNERTYPE** is **USER**.

Examples of using an owner type in BIND and REBIND commands

If a DBA wants to bind a package under a trusted context with a role as an object owner, and the package will be owned by an authorization ID, they can specify **OWNERTYPE(USER)**. For example:

```
BIND PACKAGE(MYCOLLID) MEMBER(MYPKG)OWNER(USER01) OWNERTYPE(USER)
```

The package owner is **USER01**, which is an authorization ID. Later, if the DBA must change the package owner, they can rebind the package by specifying the new owner. The following command changes the owner to a role that is named **DEVROLE**:

```
REBIND PACKAGE(MYCOLLID.MYPKG) OWNER(DEVROLE) OWNERTYPE(ROLE)
```

Similarly, if a DBA wants to bind a package without a trusted context or without the role as the object owner, they can specify either **USER** or **ROLE** as the **OWNERTYPE**.

7.3 Db2 support for continuous compliance

The IBM Z Security and Compliance Center automates the compliance process and takes the complexity out of evidence collection, requirement interpretation, and facts validation. Db2 is enhanced to support IBM Z Security and Compliance Center with automated data collection for Db2.

Db2 listens to the ENF 86 signal that is generated by the z/OS Compliance Agent services. Db2 generates System Management Facilities (SMF) 1154 trace records for the recommended system security parameter settings and Db2 Admin access configuration for Db2 for z/OS native controls.

The SMF 1154 records can be used independently of the IBM Z Security and Compliance Center solution.

For more information about SMF 1154 records, see the [Db2 13 library](#).

For more information about the ENF 86 signal, see the [Z/OS 2.5.0 library](#).

7.4 Read-only archived key Db2 support

The z/OS Integrated Cryptographic Service Facility (ICSF) added the decrypt-only archived key enhancement in z/OS 2.5. The archived key for decrypt operations can be used only to decrypt data.

Db2 13 adds support for the decrypt-only archived key when a key label is specified by using the Db2 interfaces for data set encryption. If the key label that is specified refers to a decrypt-only archived key, Db2 fails the key label specification if the key is used for creating or encrypting new data. If this key is used only for decrypting purposes, Db2 allows the use of the key. The intent is to allow archived keys to decrypt existing ciphertext, which enables re-encryption with a new key but not allow that same archived key to generate new ciphertext. This enhancement helps users with key rotations, which are an important aspect in maintaining data security.

You can specify a key label by using the **ENCRYPTION_KEYLABEL** subsystem parameter or by issuing the following DDL statements:

- ▶ **ALTER STOGROUP**
- ▶ **ALTER TABLE**
- ▶ **CREATE STOGROUP**
- ▶ **CREATE TABLE**

If the key label that is specified refers to a decrypt-only archived key, Db2 fails the key label specification and issues SQLCODE -20223 for the DDL and message DSNX242I for the subsystem parameter:

```
DSNT408I SQLCODE = -20223, ERROR: THE OPERATION FAILED. ENCRYPTION FACILITY NOT
AVAILABLE 00000000 00000D5F
DSNX242I -DB2A ENCRYPTION KEY LABEL CHANGE 655 UNSUCCESSFUL, KEY LABEL
SYSTEM.KEY01. REASON CODE 00E73006
```

Support for z/OS 2.5 decrypt-only archived keys is available in function level V13R1M100 and higher.

For more information about decrypt-only archived keys, see the [z/OS 2.5.0 library](#).

7.5 Summary

Db2 13 includes many significant enhancements that enable you to keep your data secure and compliant while increasing your ability to work with that data efficiently.

Reducing the contention for security manager resources mitigates any availability issues during remote connection authentication and performance issues during authorization checks when plans run. Changes to the **CREATE** and **ALTER** statements and to the **BIND** command eliminate certain types of restrictions that are associated with managing plans, packages, and SQL routines. The new SMF record helps streamline the compliance process. Lastly, support for decrypt-only archived keys is added for use in decrypt-only operations. These enhancements demonstrate a commitment to enabling you to work with your data easily and efficiently without sacrificing security.



IBM Db2 for z/OS utilities

Like for previous Db2 releases, IBM Db2 for z/OS utilities provide day-one support for all Db2 13 enhancements. Utilities are essential to the enablement of new core functions in Db2 13, such as the online conversion of tables from partition-by-growth (PBG) to partition-by-range (PBR) partitions, and the online change in DSSIZE to 256 GB for directory table spaces SPT01 and SYSLGRNX. Also, in keeping with the Db2 continuous delivery model, many new utility capabilities continue to be delivered in Db2 12 with PTFs. Appendix C, “Database administration” on page 263 describes the new utility functions that were added after the initial release of Db2 12.

This chapter describes several key utility enhancements in the IBM Db2 Utilities Suite for z/OS 13.1 and the core utilities.

This chapter describes the following topics:

- ▶ Utility history (function level 501)
- ▶ Enhanced inline statistics with page-level sampling (function level 500)
- ▶ Enhanced space-level recovery (function level 100)
- ▶ REORG INDEX NOSYSUT1 as the default option (function level 500)
- ▶ REPAIR WRITELOG dictionary support (function level 100)

For a complete list of all utility enhancements, see [Db2 Utilities in Db2 13](#).

8.1 Utility history (function level 501)

An important part of daily data management in Db2 subsystems is the running of utilities. Db2 system and database administrators use various utilities to back up, recover, reorganize, unload, load, and collect statistics on data. To prevent or minimize an adverse impact on critical applications and processes, it is imperative that the utilities run as efficiently as possible and finish successfully in a predictable timeframe.

Before Db2 13, it was challenging to gather historical information about utility executions, which also made utility management and tuning difficult and time-consuming. Checking for any utility failures in the past 24 hours could not be done quickly or easily. To analyze performance statistics or other information about utilities that ran on a Db2 subsystem, you had to gather sufficient information from each utility output or from Db2 traces, but it was nearly impossible to easily obtain a concise history of utility executions, which also would be useful for balancing and managing utility workloads.

Db2 13 introduces a new utility history function that collects essential and useful execution information across all the utilities. With function level 501 (V13R1M501), you can configure Db2 to gather the history of Db2 utilities in real time. When requested, Db2 collects and stores the historical information in the Db2 catalog. Then, you can use this information to better manage your utility strategy. For example, you can check daily utility executions for failures and take immediate corrective actions, or use the historical trends to balance the utility workload, such as moving jobs from one execution window to another one or moving objects from job to job.

The following Db2 utilities are enhanced to support the utility history function:

- ▶ **BACKUP SYSTEM**
- ▶ **CATMAINT**
- ▶ **CHECK DATA**
- ▶ **CHECK INDEX**
- ▶ **CHECK LOB**
- ▶ **COPY**
- ▶ **COPYTOCOPY**
- ▶ **LOAD**
- ▶ **MERGECOPY**
- ▶ **MODIFY RECOVERY**
- ▶ **MODIFY STATISTICS**
- ▶ **QUIESCE**
- ▶ **REBUILD INDEX**
- ▶ **RECOVER**
- ▶ **REORG**
- ▶ **REPAIR**
- ▶ **REPORT RECOVERY**
- ▶ **REPORT TABLESPACESET**
- ▶ **RUNSTATS**
- ▶ **STOSPACE**
- ▶ **UNLOAD**

A new subsystem parameter, **UTILITY_HISTORY**, was introduced to collect utility history, which can be updated online. By default, the parameter is set to **NONE**, and no utility history is collected. To activate utility history collection, set **UTILITY_HISTORY** to **UTILITY**.

For a data-sharing group, set the parameter in all Db2 members to the same setting to ensure the collection of a complete history when utilities are run on different members. The setting of the parameter is checked at the beginning of each utility statement and accepted until the completion of the utility execution.

Db2 stores the utility history in the new SYSIBM.SYSUTILITIES catalog table. A new row is inserted into SYSUTILITIES at the beginning of each utility execution. This row includes a unique event ID in the EVENTID column to identify the utility execution. If the utility execution inserts one or more rows into SYSIBM.SYSCOPY, the EVENTID value is also recorded in the new EVENTID column in SYSCOPY. The SYSUTILITIES row also contains details about the utility execution, such as the job name, utility name, number of objects, starting and ending timestamp, elapsed time, CPU time, zIIP time, final return code, and special conditions, such as restart and termination with the **TERM UTIL** command. Information is collected when the utility is invoked by the **DSNUPROC** JCL procedure, the **DSNUTILB** program, or the **DSNUTILU** or **DSNUTILV** stored procedure. Information in the SYSUTILITIES row is updated as the utility progresses.

To retrieve the utility history, you can query SYSUTILITIES, SYSCOPY, or both catalog tables. The output of the **REPORT RECOVERY** utility includes any relevant event IDs from SYSCOPY, and the output of the **DSNJU004** (print log map) utility includes any relevant event IDs for system-level backups created by **BACKUP SYSTEM** and recorded in the bootstrap data set (BSDS).

For more information about how utility history collection works, including restrictions and processing, see [Monitoring Utility History](#).

8.1.1 Number of SYSUTILITIES rows that are inserted by a utility

If object names are explicitly specified in a utility statement, the statement is treated as one utility execution and only one SYSUTILITIES row is inserted.

If a LISTDEF list is specified in a utility statement, the number of SYSUTILITIES rows that are inserted into the catalog table varies depending on the utility, the specified utility options, and the specified LISTDEF options. One row is inserted for each separate invocation of a utility. If a list of objects is split into separate utility runs by Db2, each run results in a separate row. The following examples show the number of SYSUTILITIES rows that are inserted by different utilities or utility runs:

- ▶ **COPY**, **RECOVER**, **COPYTOCOPY**, and **QUIESCE** are invoked once to process all the objects on a list and only one SYSUTILITIES row is inserted.
- ▶ **REORG TABLESPACE** on a list of partitions is invoked once for each group of related partitions (the ones that belong to the same table space). One SYSUTILITIES row is inserted for each group of partitions that is processed.
- ▶ **CHECK INDEX** or **REBUILD INDEX** on a list of index spaces is invoked once for each group of related index spaces (the ones that are associated with the same table space). One SYSUTILITIES row is inserted for each group of related index spaces that are processed.
- ▶ **MODIFY RECOVERY** on a list of table spaces is invoked once for each table space. One SYSUTILITIES row is inserted for each table space that is processed.

As shown in the following example, a new DSNU3031I message is issued when a SYSUTILITIES row is inserted:

```
DSNU3031I -DB2A 037 16:04:03.96 DSNUHUTL - UTILITY HISTORY COLLECTION IS ACTIVE.  
LEVEL: UTILITY, EVENTID: 3888
```

The message displays the event ID in the utility output when utility history collection is active.

8.1.2 EVENTID columns

The EVENTID columns in SYSUTILITIES and SYSCOPY have a data type of BIGINT. When the SYSUTILITIES row for a utility execution is inserted, the EVENTID column is assigned a unique value that uses Db2 sequence SYSIBM.DSNSEQ_EVENTID. The SYSCOPY rows that are inserted for the utility contain the same EVENTID column value as its corresponding row in SYSUTILITIES.

In a data-sharing group, the event IDs may not be assigned sequentially according to the order of utility executions because each Db2 member has a cache of assigned sequence values to use. Gaps might exist in the utility event IDs because unused sequence values in the cache are not kept when Db2 is restarted, which affects both data-sharing and non-data-sharing environments. You can use the STARTTS (start timestamp) column to sort the SYSUTILITIES rows to see the order of utility executions.

8.1.3 SYSUTILITIES columns

Most column names in the SYSUTILITIES table describe the column values, but some columns are not as intuitive and need more explanation or information:

- ▶ The NAME column contains the name of one of the utilities names that were listed earlier. Utilities that perform the same function are grouped. For example, **REORG TABLESPACE** and **REORG INDEX** reorganize data or index keys and are combined under **REORG**. Similarly, **RUNSTATS TABLESPACE** and **RUNSTATS INDEX** are grouped under **RUNSTATS**, both of which gather statistics.
- ▶ The INSERTEDBY column is set to 'Db2' when the SYSUTILITIES row is inserted by one of the Db2 utilities that were listed earlier.
- ▶ The STARTTS and ENDTS timestamp column values are in local time. The SYSUTILITIES row is inserted with the STARTTS value, which is preserved even if the utility receives anabend and then restarts.
- ▶ The STARTTS, ENDTS, ELAPSEDTIME, CPUTIME, and ZIIPTIME column values for a utility execution match the field values of the corresponding Instrumentation Facility Component Identifier (IFCID) 25 (utility end information) trace record. The ELAPSEDTIME, CPUTIME, and ZIIPTIME column values are in microseconds.
- ▶ The ZIIPTIME column is the time in microseconds when zIIP processors are used and a Db2 accounting class 1 trace is active. When applicable, it includes DFSORT or Db2 Sort zIIP processing time if reported to the utility by the sort program.
- ▶ The RETURNCODE and CONDITION values can be used together to help determine the status of a utility, as shown in Table 8-1 on page 115.

Table 8-1 Combinations of RETURNCODE and CONDITION column values

RETURNCODE column values	CONDITION column values	Explanation
NULL	Blank	Utility is active or stopped. Issue the DIS UTIL(utlid) command by using the UTILID column value from the SYSUTILITIES row and check for messages: <ul style="list-style-type: none"> ▶ DSNU105I: Utility is active. ▶ DSNU100I: Utility is stopped.
0 or 4	E	Utility ends successfully. For RETURNCODE=4, one or more warning messages are issued.
8	E	Utility ends with one or more errors.
8	T	Active utility accepts the TERM UTIL command.
NULL	T	The TERM UTIL command terminates a stopped utility.
NULL	F	The START DB SP ACCESS(FORCE) command terminates a stopped utility.

- ▶ The NUMOBJECTS column is the number of objects to process. In general, these objects are placed in a utility-in-progress state (UTUT, UTRO, or UTRW) by the utility. For partitioned objects, each partition is counted as one object. For non-partitioned objects, each object counts as one object except for data-set-level requests (**DSNUM** option) for backup and recovery utilities.
- ▶ The STARTLOGPOINT column value is the current relative byte address (RBA) for non-data-sharing or log record sequence number (LRSN) for data-sharing at the start of the utility, which can be used for diagnostic purposes.

Another 14 sets of columns that collect utility phase information are reserved for future use. For now, these columns have a default value of NULL: PHASEnNAME, PHASEnET, PHASEnCPUT, PHASEnZIPT, and PHASEnDATA, where *n* is the number that references one of the 14 columns.

8.1.4 Abends, restarts, and commands

When a utility abends, the SYSUTILITIES row is not updated. The ENDTS and RETURNCODE columns remain null and the CONDITION column value remains blank. The utility is in a stopped state and the SYSUTILITIES row is updated during termination after a restart or a **TERM UTIL** command.

When a stopped utility is restarted, the RESTART column value is set to 'Y' at the beginning of execution. Upon completion, the ELAPSEDTIME column value includes the time when the utility was stopped. The CPUTIME, ZIPTIME, SORTCPUTIME, and SORTZIPTIME column values reflect the resources that are used in the restart execution only. The restart of a utility accepts the **UTILITY_HISTORY** subsystem parameter setting from the first, original execution, which avoids incomplete information in SYSUTILITIES rows.

A **TERM UTIL** command that is issued on a stopped utility updates the **ENDTS**, **ELAPSED TIME**, and **CONDITION** columns. A **START DATABASE(...)** **SPACENAM(...)** **ACCESS(FORCE)** command, which terminates a stopped utility, also updates these columns. The **ENDTS** column reflects the time when the utility is terminated by the command. The **ELAPSED TIME** column value includes the time when the utility was stopped. The **CONDITION** column value is set to 'T' for **TERM UTIL** or 'F' for **START DATABASE** with **ACCESS(FORCE)**. The **CPUTIME**, **ZIIP TIME**, **SORTCPUTIME**, and **SORTZIIP TIME** column values are **NULL** by default.

A **TERM UTIL** command that is issued on an active utility notifies the utility to terminate. The **ENDTS**, **ELAPSED TIME**, and **CONDITION** columns are updated by the utility during the termination processing.

8.1.5 Db2 catalog objects for utility history

The new Db2 catalog objects for utility history are created by the **CATMAINT** utility. Table space **DSNDB06.SYSTSUTL** and indexes **SYSIBM.DSNULX01** and **SYSIBM.DSNULX02** on the **SYSUTILITIES** table are created with the **DEFINE NO** attribute. The first insert into the **SYSUTILITIES** table defines these objects.

You can run utilities on the utility history catalog objects if needed. Do not include other objects in the utility statement or list, and if possible, avoid running utilities on other objects concurrently. When a utility needs exclusive use of or allows limited access to the utility history catalog objects, it is best to avoid running other utilities because the insert or update of the **SYSUTILITIES** row for utility history processing might not be successful.

The **SYSTSUTL** table space must be copied and recovered by itself in separate utility statements than other table spaces. Otherwise, an error is issued. If the indexes **DSNULX01** and **DSNULX02** are altered with the **COPY YES** attribute, they can be included in the same utility statement along with any user-defined indexes on **SYSUTILITIES**.

The recovery order of catalog and directory objects is modified to include table space **SYSTSUTL** and the indexes over **SYSUTILITIES**. For more information, see [Recovering catalog and directory objects](#).

8.1.6 Cases where utility history is not collected

Utility history is not collected under the following conditions:

- ▶ Utility history collection does not occur if the **PREVIEW JCL** parameter or the **OPTIONS PREVIEW** control statement is specified for the utility job step. In **PREVIEW** mode, normal utility execution does not take place.
- ▶ Utility history collection is automatically deactivated if Db2 is started in the system recovery-pending state, remote site recovery, tracker site, or **ACCESS(MAINT)** mode when the recovery of all objects is deferred (**DEFER ALL**).
- ▶ Utility history information is not collected when utilities run on the catalog objects that contain utility history information. When these objects, such as the **SYSUTILITIES** table, its indexes (including user-defined indexes), and the **SYSTSUTL** table space, are named in a utility statement or in a list with other objects, utility history information is not collected for the utility execution.

- ▶ Utility history information is not collected for utilities that are run on user objects, directory objects, or other catalog objects if the utility history catalog objects are in a UTRO or UTUT utility-in-progress state, or a prohibitive state, such as COPY-pending, GRECP, logical page list (LPL), REBUILD-pending, RECOVER-pending, and REORG-pending. The utility history information cannot be inserted or updated if these objects are in a restricted or prohibitive state. The same situation applies if these objects were started for read-only access (including persistent read-only) or were stopped.
- ▶ Utility history information is not gathered during the recovery and rebuild of Db2 catalog and directory objects that are ahead of the utility history catalog objects in the recovery order. This condition avoids access of objects that are not recovered yet.
- ▶ When utility history is active and a resource-unavailable condition is encountered during the update of the SYSUTILITIES table or its indexes, a new informational message is issued:

```
DSNU3032I ERROR DURING UTILITY HISTORY COLLECTION, RETURN CODE X'return-code'
REASON CODE X'reason-code'
```

The utility bypasses this condition and continues processing but no longer collects or attempts to update history information. This condition does not impact the execution of the utility or its final return code.

8.1.7 Queries on utility history

You can run queries to retrieve the utility history information. To avoid impacting normal utility operations and other processes, run your queries with the **ISO(UR)** option. If needed, you can create user-defined indexes on SYSUTILITIES or SYSCOPY for query performance.

Sample SQL statements for retrieving utility history

The SQL **SELECT** statements are examples of retrieving utility history information. You can modify the SQL statements to run in your environment. Some of the query results show useful information immediately, but others need more history information to accumulate over time.

The sample SQL statements that are shown in Example 8-1 are for creating global variables and assigning values for retrieving utility history. You can tailor them to fit your needs or modify the queries to use specific values instead of these global variables.

Example 8-1 Sample SQL statements for creating global variables

```
SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
CREATE VARIABLE UH_DB VARCHAR(8); -- DATABASE NAME
CREATE VARIABLE UH_PS VARCHAR(8); -- TABLE SPACE NAME OR
-- INDEX SPACE NAME OR
-- '1' FOR ALL OBJECTS IN DATABASE

CREATE VARIABLE UH_NO_DAYS SMALLINT; -- NUMBER OF DAYS
CREATE VARIABLE UH_NAME VARCHAR(60); -- UTILITY NAME
CREATE VARIABLE UH_STARTTS TIMESTAMP; -- START TIMESTAMP
CREATE VARIABLE UH_ENDTS  TIMESTAMP; -- END TIMESTAMP
CREATE VARIABLE UH_DELETE_ROWS CHAR(1); -- '0' = DON'T DELETE SYSUTILITIES
-- '1' = DELETE SYSUTILITIES ROWS

CREATE VARIABLE UH_DELETE_PREVIEW CHAR(1); -- '0' = NO DELETE PREVIEW
-- '1' = DETAILED DELETE PREVIEW
```

If any of these global variables exist with different attributes, choose a different name and modify the SQL statements. Change the **SET CURRENT PATH** statement to the schema for the global variables.

The sample query that is shown in Example 8-2 retrieves history information about utilities that began execution between midnight and early morning. Utilities that are stopped or still active are also included.

Example 8-2 Sample query for displaying utilities that run between midnight and early morning

```
SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_STARTTS = '2022-04-04-00.00.00.000000'; -- START TIMESTAMP
SET UH_ENDTS = '2022-04-04-08.00.00.000000'; -- END TIMESTAMP
SELECT EVENTID,NAME,JOBNAME,UTILID,STARTTS,ENDTS,RETURNCODE,CONDITION
FROM SYSIBM.SYSUTILITIES
WHERE STARTTS >= UH_STARTTS
      AND STARTTS <= UH_ENDTS;
```

The sample query that is shown in Example 8-3 shows all active or stopped utilities. Issue the **DIS UTIL(utl id)** command by using the UTILID column value of each SYSUTILITIES row to check whether the utility is active or stopped.

Example 8-3 Sample query for displaying all active or stopped utilities

```
SELECT EVENTID,NAME,JOBNAME,UTILID,STARTTS,RETURNCODE,CONDITION
FROM SYSIBM.SYSUTILITIES
WHERE RETURNCODE IS NULL
      AND CONDITION=' ';
```

The sample query that is shown in Example 8-4 displays utilities with return code 8 or higher in the last 24 hours.

Example 8-4 Sample query for displaying utility executions with errors in the last 24 hours

```
SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_NO_DAYS = 1; -- NUMBER OF DAYS
SELECT EVENTID,NAME,JOBNAME,UTILID,STARTTS,RETURNCODE,CONDITION
FROM SYSIBM.SYSUTILITIES
WHERE STARTTS >= CURRENT_TIMESTAMP - UH_NO_DAYS DAYS
      AND RETURNCODE >= 8;
```

You can use the sample query that is shown in Example 8-5 to check on restarted utilities that are in an active or stopped state.

Example 8-5 Sample query for showing restarted utilities in an active or stopped state

```
SELECT
      EVENTID,NAME,JOBNAME,UTILID,STARTTS,RESTART,RETURNCODE,CONDITION
FROM SYSIBM.SYSUTILITIES
WHERE RESTART = 'Y'
      AND RETURNCODE IS NULL
      AND CONDITION=' ';
```

The sample queries that are shown in Example 8-6, Example 8-7, and Example 8-8 show the executions of **REORG**, **COPY**, and **LOAD** utilities respectively in the last 7 days sorted by different performance statistics.

Example 8-6 Sample query for displaying REORG TABLESPACE and REORG INDEX executions that are sorted by descending elapsed time in the last 7 days

```
SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_NAME = 'REORG'; -- UTILITY NAME
SET UH_NO_DAYS = 7; -- NUMBER OF DAYS
SELECT EVENTID,NAME,JOBNAME,UTILID,STARTTS,ELAPSEDTIME,RETURNCODE,CONDITION
FROM SYSIBM.SYSUTILITIES
WHERE NAME = UH_NAME
AND STARTTS >= CURRENT TIMESTAMP - UH_NO_DAYS DAYS
ORDER BY ELAPSEDTIME DESC;
```

Example 8-7 Sample query for COPY executions for job T6E11028 sorted by descending elapsed time in the last 7 days

```
SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_NAME = 'COPY'; -- UTILITY NAME
SET UH_NO_DAYS = 7; -- NUMBER OF DAYS
SELECT
EVENTID,NAME,JOBNAME,UTILID,STARTTS,ELAPSEDTIME,RETURNCODE,CONDITION
FROM SYSIBM.SYSUTILITIES
WHERE NAME = UH_NAME
AND JOBNAME='T6E11028'
AND STARTTS >= CURRENT TIMESTAMP - UH_NO_DAYS DAYS
ORDER BY ELAPSEDTIME DESC;
```

Example 8-8 Sample query for displaying LOAD executions that are sorted by descending CPU time in the last 7 days

```
SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_NAME = 'LOAD'; -- UTILITY NAME
SET UH_NO_DAYS = 7; -- NUMBER OF DAYS
SELECT
EVENTID,NAME,JOBNAME,UTILID,STARTTS,CPUTIME,RETURNCODE,CONDITION
FROM SYSIBM.SYSUTILITIES
WHERE NAME= UH_NAME
AND STARTTS >= CURRENT TIMESTAMP - UH_NO_DAYS DAYS
ORDER BY CPUTIME DESC;
```

The sample queries that are shown in Example 8-9 to Example 8-12 on page 120 show the counts of executions by utility name.

Example 8-9 Sample query for displaying the total count of executions by utility name

```
SELECT A.NAME AS UTILITY_TYPE, COUNT(*) AS UTILITY_COUNT,A.EVENTID
FROM SYSIBM.SYSUTILITIES A
GROUP BY A.NAME,A.EVENTID;
```

Example 8-10 Sample query for showing the total count of executions by utility name with a corresponding SYSCOPY row

```
SELECT A.NAME AS UTILITY_TYPE, COUNT(*) AS UTILITY_COUNT
FROM SYSIBM.SYSUTILITIES A
WHERE EXISTS (SELECT * FROM SYSCOPY B WHERE B.EVENTID = A.EVENTID)
GROUP BY A.NAME;
```

Example 8-11 Sample query for showing the count of executions by utility name in the past 7 days

```
SET UH_NO_DAYS = 7; -- NUMBER OF DAYS
SELECT A.NAME AS UTILITY_TYPE, COUNT(*) AS UTILITY_COUNT
FROM SYSIBM.SYSUTILITIES A
WHERE A.STARTTS >= CURRENT_TIMESTAMP - UH_NO_DAYS DAYS
GROUP BY A.NAME;
```

Example 8-12 Sample query for displaying the count of different utilities with SYSCOPY rows in the past 7 days

```
SET UH_NO_DAYS = 7; -- NUMBER OF DAYS
SELECT A.NAME AS UTILITY_TYPE, COUNT(*) AS UTILITY_COUNT
FROM SYSIBM.SYSUTILITIES A
WHERE A.STARTTS >= CURRENT_TIMESTAMP - UH_NO_DAYS DAYS
AND
EXISTS (SELECT * FROM SYSCOPY B WHERE B.EVENTID = A.EVENTID)
GROUP BY A.NAME;
```

Sample queries on SYSUTILITIES and SYSCOPY information

You can use the EVENTID column to join and retrieve the information in the SYSUTILITIES and SYSCOPY tables. If needed, query the SYSIBM.SYSTABLEPART, SYSIBM.SYSINDEXES, and SYSIBM.SYSINDEXPART catalog tables to find out whether an object is partitioned or non-partitioned. The sample queries in this section show you how to retrieve data in SYSUTILITIES and SYSCOPY.

The first example query (Example 8-13 on page 121) shows the most recent successful execution of **REORG TABLESPACE** for a specific table space or **REORG INDEX** for a specific index space (with the **COPY YES** attribute). Set global variables UH_DB to the database name and UH_PS to the name of the table space or the index space. For partitioned objects, the **REORG** information is displayed for each partition, even when space level processing occurred.

Some data in SYSCOPY and SYSUTILITIES might be deleted independently from each other by **MODIFY RECOVERY** and **SQL DELETE** respectively. It is possible that information might not exist for the most recent **REORG** execution. The information in a SYSCOPY row contains the object information and if it does not exist, the query displays 'NO CORRESPONDING SYSCOPY ROW FOUND'. If the SYSUTILITIES row does not exist, the query displays 'NO CORRESPONDING SYSUTILITIES ROW FOUND'. When DSNTEP3 is used to run this query and the 'NO CORRESPONDING ...' text appears, it might set a return code of 4, but it does not affect the query results.

You can set global variable UH_PS to '1' to show the most recent **REORG** for all table spaces and index spaces (with the **COPY YES** attribute) in the database.

Example 8-13 Query for the most recent successful REORG for table space DB000231.TP000231

```
SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_DB = 'DB000231'; -- DATABASE NAME
SET UH_PS = 'TP000231'; -- TABLE SPACE OR INDEX SPACE (COPY YES)
--SET UH_PS = '1'; -- ALL TABLE SPACES OR INDEX SPACES (COPY YES)
--THE OUTER MOST BLOCK OF THIS QUERY INCLUDES THE SYSUTILITIES
--ENDTS VALUE IF A SYSCOPY ROW WAS PREVIOUSLY FOUND. IF NO SYSUTILITIES
--ROW IS FOUND THEN THE NULL ENDTS IS CHANGED TO THE "NO CORRESPONDING"
--MESSAGE.
SELECT Y.PAGESET_TYPE, Y.DBNAME, Y.NAME, Y.PARTITION, Y.EVENTID
      ,Y.SC_TIMESTAMP
      ,(COALESCE(
        (SELECT CAST(ENDTS AS VARCHAR(30))
         FROM SYSIBM.SYSUTILITIES H
         WHERE H.EVENTID = Y.EVENTID
        ),
        'NO CORRESPONDING SYSUTILITIES ROW FOUND'
      )
      ) AS SU_ENDTS
FROM
(
--AFTER THE LATEST REORG PER DATABASE, PAGE SET, AND PARTITION
--IS FOUND, THE SYSCOPY EVENTID IS ADDED TO THE RESULT, WHICH MAY
--OR MAY NOT EXIST.
  SELECT Z.PAGESET_TYPE, Z.DBNAME, Z.NAME, Z.PARTITION
        ,(SELECT G.EVENTID
         FROM SYSIBM.SYSCOPY G
         WHERE G.DBNAME = Z.DBNAME
           AND G.TSNAME = Z.NAME
           AND ( G.DSNUM = Z.PARTITION
                OR G.DSNUM = 0)
           AND G.ICTYPE IN ('W', 'X')
           AND G.TIMESTAMP = Z.SC_MAXTIMESTAMP
        ) AS EVENTID
--IF NO CORRESPONDING SYSCOPY ROW IS FOUND OF AN ENTRY IN THE LATEST
--REORG LIST, THE NULL TIMESTAMP IS CHANGED TO A "NO CORRESPONDING..."
--MESSAGE.
        ,COALESCE(CAST(Z.SC_MAXTIMESTAMP AS VARCHAR(30)),
                  'NO CORRESPONDING SYSCOPY ROW FOUND'
        ) AS SC_TIMESTAMP
FROM
(
--ACTING ON THE PRIMARY LIST, THE ENTRIES ARE REDUCED TO THE LATEST
--REORG PER DATABASE, PAGE SET, AND PARTITION (MAX TIMESTAMP).
  SELECT D.PAGESET_TYPE
        ,D.DBNAME
        ,D.NAME
        ,D.PARTITION
        ,MAX(E.TIMESTAMP) AS SC_MAXTIMESTAMP
FROM
(
--THIS PRIMARY BLOCK CREATES A LIST OF PAGESETS BASED ON THE GLOBAL
--VARIABLE INPUT PROVIDED. IF A CORRESPONDING SYSCOPY ROW EXISTS
--AT THE PARTITION LEVEL, THE TIMESTAMP IS RETURNED, OTHERWISE IT
--WILL BE NULL.
```

```

SELECT 'TS' AS PAGESET_TYPE
      ,A.DBNAME AS DBNAME, A.TSNAME AS NAME
      ,A.PARTITION AS PARTITION
FROM SYSIBM.SYSTABLEPART A
WHERE A.DBNAME = UH_DB
      AND (UH_PS = '1' OR A.TSNAME = UH_PS)
UNION ALL
SELECT 'IS' AS PAGESET_TYPE
      ,B.DBNAME AS DBNAME, B.INDEXSPACE AS NAME
      ,C.PARTITION AS PARTITION
FROM SYSIBM.SYSINDEXES B
      ,SYSIBM.SYSINDEXPART C
WHERE B.DBNAME = UH_DB
      AND (UH_PS = '1' OR B.INDEXSPACE = UH_PS)
      AND C.IXCREATOR = B.CREATOR
      AND C.IXNAME = B.NAME
) D
LEFT OUTER JOIN
SYSIBM.SYSCOPY E
ON E.DBNAME = D.DBNAME
AND E.TSNAME = D.NAME
AND (E.DSNUM = 0 OR E.DSNUM = D.PARTITION)
AND ICTYPE IN ('W', 'X')
GROUP BY D.PAGESET_TYPE, D.DBNAME, D.NAME, D.PARTITION
) Z
) Y
ORDER BY 1 DESC, 2, 3, 4;

```

This sample query returns and displays results that are similar to Figure 8-1 and Figure 8-2 on page 123.

In Figure 8-1, the results include a row for each partition of table space DB000231.TP000231. Partitions 1 and 3 were processed by the **REORG TABLESPACE** execution with event ID 1001. Partitions 2 and 4 were processed by the **REORG TABLESPACE** execution with event ID 1004. A **REORG TABLESPACE** execution was not recorded in SYSCOPY and SYSUTILITIES for partitions 5 - 10, so 'NO CORRESPONDING ...' is displayed in the SC_TIMESTAMP (SYSCOPY TIMESTAMP) and SU_ENDTS (SYSUTILITIES ENDTS) result columns for these partitions.

----- VERSION DSNTPE3 P140 -----						
PAGESET_TYPE	DBNAME	NAME	PARTITION	EVENTID	SC_TIMESTAMP	SU_ENDTS
1 TS	DB000231	TP000231	1	1001	2022-04-21-09.54.23.656362	2022-04-21-09.54.24.299666
2 TS	DB000231	TP000231	2	1004	2022-04-21-09.55.05.849911	2022-04-21-09.56.06.538834
3 TS	DB000231	TP000231	3	1001	2022-04-21-09.54.23.656981	2022-04-21-09.54.24.299666
4 TS	DB000231	TP000231	4	1004	2022-04-21-09.56.05.850912	2022-04-21-09.56.06.538834
5 TS	DB000231	TP000231	5	?	NO CORRESPONDING SYSCOPY ROW FOUND	NO CORRESPONDING SYVSUTILITIES ROW FOUND
6 TS	DB000231	TP000231	6	?	NO CORRESPONDING SYSCOPY ROW FOUND	NO CORRESPONDING SYVSUTILITIES ROW FOUND
7 TS	DB000231	TP000231	7	?	NO CORRESPONDING SYSCOPY ROW FOUND	NO CORRESPONDING SYVSUTILITIES ROW FOUND
8 TS	DB000231	TP000231	8	?	NO CORRESPONDING SYSCOPY ROW FOUND	NO CORRESPONDING SYVSUTILITIES ROW FOUND
9 TS	DB000231	TP000231	9	?	NO CORRESPONDING SYSCOPY ROW FOUND	NO CORRESPONDING SYVSUTILITIES ROW FOUND
10 TS	DB000231	TP000231	10	?	NO CORRESPONDING SYSCOPY ROW FOUND	NO CORRESPONDING SYVSUTILITIES ROW FOUND

Figure 8-1 Query output for table space DB000231.TP000231

In Figure 8-2, the results show the most recent **REORG TABLESPACE** execution on each partition of table space DB000231.TP000231. In this case, all partitions were reorganized. The event IDs of 1001, 1004, 1007, and 1008 indicate that the partitions were processed by four different **REORG** executions.

-- VERSION DSNTPE3 P14D --					
PAGESET_TYPE	DBNAME	NAME	PARTITION	EVENTID	
1	DB000231	TP000231	1	1001	
2	DB000231	TP000231	2	1004	
3	DB000231	TP000231	3	1001	
4	DB000231	TP000231	4	1004	
5	DB000231	TP000231	5	1007	
6	DB000231	TP000231	6	1007	
7	DB000231	TP000231	7	1007	
8	DB000231	TP000231	8	1007	
9	DB000231	TP000231	9	1008	
10	DB000231	TP000231	10	1008	

-- VERSION DSNTPE3 P14D --		
SC_TIMESTAMP	SU_ENDTS	
1	2022-04-21-09.54.23.656362	2022-04-21-09.54.24.259606
2	2022-04-21-09.56.05.849911	2022-04-21-09.56.06.530034
3	2022-04-21-09.54.23.656981	2022-04-21-09.54.24.259606
4	2022-04-21-09.56.05.850912	2022-04-21-09.56.06.530034
5	2022-04-23-16.30.33.237481	2022-04-23-16.30.33.819598
6	2022-04-23-16.30.33.239329	2022-04-23-16.30.33.819598
7	2022-04-23-16.30.33.239395	2022-04-23-16.30.33.819598
8	2022-04-23-16.30.33.239456	2022-04-23-16.30.33.819598
9	2022-04-23-16.34.53.819420	2022-04-23-16.34.54.312315
10	2022-04-23-16.34.53.819832	2022-04-23-16.34.54.312315

Figure 8-2 Query output for table space DB000231.TP000231

The next example query (Example 8-14) shows the most recent successful execution of **REORG TABLESPACE** or **REORG INDEX** on objects in a database when the **SYSCOPY** information for the **REORG** exists. The results are like the results for Example 8-13 on page 121, but nothing is displayed when the **SYSCOPY** information does not exist.

Example 8-14 Query for the most recent successful REORG for table space DB000231.TP000231 when the corresponding SYSCOPY information exists

```

SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_DB = 'DB000231'; -- DATABASE NAME
SET UH_PS = 'TP000231'; -- TABLE SPACE OR INDEX SPACE (COPY YES)
--SET UH_PS = '1'; -- ALL TABLE SPACES OR INDEX SPACES (COPY YES)
--THE OUTER MOST PART OF THIS QUERY ACTS ON THE LIST OF THE LATEST
-- REORGS PER DATABASE, PAGE SET, AND PARTITION, ALONG WITH OPTIONAL
--SYSUTILITIES INFORMATION. IF NO SYSUTILITIES INFORMATION IS FOUND, A
--"NO CORRESPONDING..." MESSAGE IS DISPLAYED.
      SELECT I.PAGESET_TYPE, I.DBNAME, I.NAME, I.PARTITION, I.EVENTID
      ,I.SC_TIMESTAMP
      ,COALESCE(CAST(F.ENDTS AS VARCHAR(30)),
        'NO CORRESPONDING SYSUTILITIES ROW FOUND'
      ) AS SU_ENDTS
FROM
  (
--THIS SELECT LIST EXPRESSION ADDS THE SYSCOPY EVENTID INTO THE
--RESULT LIST OF THE LATEST REORG PER DATABASE, PAGE SET, AND
--PARTITION.
    SELECT G.PAGESET_TYPE, G.DBNAME, G.NAME, G.PARTITION
      ,(SELECT E1.EVENTID
        FROM SYSIBM.SYSCOPY E1
        WHERE E1.DBNAME = G.DBNAME
          AND E1.TSNAME = G.NAME
          AND (E1.DSNUM = G.PARTITION OR E1.DSNUM = 0)
          AND E1.TIMESTAMP = G.SC_TIMESTAMP
        ) AS EVENTID
      ,G.SC_TIMESTAMP
    FROM
  (
--THIS PRIMARY BLOCK CREATES A LIST OF PAGE SETS BASED ON THE GLOBAL
--VARIABLE INPUT PROVIDED. THIS QUERY ONLY RETURNS PAGE SETS
--WHERE A SYSCOPY ROW EXISTS, THEREFORE SYSCOPY INNER JOIN
--RETURNS THE TIMESTAMP COLUMN AND SELECTS THE LATEST REORG

```

```

--(MAX TIMESTAMP) PER DATABASE, PAGE SET, AND PARTITION.
SELECT D.PAGESET_TYPE, D.DBNAME, D.NAME, D.PARTITION
      ,MAX(E.TIMESTAMP) AS SC_TIMESTAMP
FROM
  (
    SELECT 'TS' AS PAGESET_TYPE
          ,A.DBNAME AS DBNAME, A.TSNAME AS NAME
          ,A.PARTITION AS PARTITION
    FROM SYSIBM.SYSTABLEPART A
    WHERE A.DBNAME = UH_DB
          AND (UH_PS = '1' OR A.TSNAME = UH_PS)
    UNION ALL
    SELECT 'IS' AS PAGESET_TYPE
          ,B.DBNAME AS DBNAME, B.INDEXSPACE AS NAME
          ,C.PARTITION AS PARTITION
    FROM SYSIBM.SYSINDEXES B
          ,SYSIBM.SYSINDEXPART C
    WHERE B.DBNAME = UH_DB
          AND (UH_PS = '1' OR B.INDEXSPACE = UH_PS)
          AND C.IXCREATOR = B.CREATOR
          AND C.IXNAME = B.NAME
    ) D
  ,SYSIBM.SYSCOPY E
  WHERE E.DBNAME = D.DBNAME
        AND E.TSNAME = D.NAME
        AND (E.DSNUM = 0 OR E.DSNUM = D.PARTITION)
        AND ICTYPE IN ('W', 'X')
  GROUP BY D.PAGESET_TYPE, D.DBNAME, D.NAME, D.PARTITION
  ) G
  ) I
--AFTER THE LATEST REORG PER DATABASE, PAGE SET, AND PARTITION
--IS FOUND, THE SYSCOPY EVENTID IS USED TO LOOK FOR A CORRESPONDING
--SYSUTILITIES ROW.
LEFT OUTER JOIN
SYSIBM.SYSUTILITIES F
ON F.EVENTID = I.EVENTID
  AND F.NAME = 'REORG'
  AND F.RETURNCODE IN (0, 4)
  AND F.CONDITION = 'E'
ORDER BY 1 DESC, 2, 3, 4;

```

This sample query returns and displays results that are similar to Figure 8-3, Figure 8-4 on page 125, and Figure 8-5 on page 125.

In Figure 8-3, the results include a row for four (out of the ten) partitions of table space DB000231.TP000231 with a **REORG TABLESPACE** execution that is recorded in SYSCOPY. Partitions 1 and 3 were processed by the **REORG** execution with event ID 1001. Partitions 2 and 4 were processed by the **REORG** execution with event ID 1004. No rows are displayed for partitions 5 - 10 because a **REORG** execution is not recorded in SYSCOPY.

PAGESET_TYPE	DBNAME	NAME	PARTITION	EVENTID
TS	DB000231	TP000231	1	1001
TS	DB000231	TP000231	2	1004
TS	DB000231	TP000231	3	1001
TS	DB000231	TP000231	4	1004

SC_TIMESTAMP	SU_ENDTS
2022-04-21-09.54.23.656362	2022-04-21-09.54.24.259606
2022-04-21-09.56.05.049911	2022-04-21-09.56.06.530834
2022-04-21-09.54.23.656361	2022-04-21-09.54.24.259605
2022-04-21-09.56.05.050912	2022-04-21-09.56.06.530834

Figure 8-3 Query output for table space DB000231.TP000231

In Figure 8-4, the results show the most recent **REORG TABLESPACE** on each of the 10 partitions of table space DB000231.TP000231. The event IDs of 1001, 1004, 1007, and 1008 indicate that the partitions were processed by four different **REORG** executions (recorded in SYSCOPY).

- VERSION DSNTSTEP3 P140 -					
PAGESET_TYPE	DBNAME	NAME	PARTITION	EVENTID	
TS	DB000231	TP000231	1	1001	
TS	DB000231	TP000231	2	1004	
TS	DB000231	TP000231	3	1001	
TS	DB000231	TP000231	4	1004	
TS	DB000231	TP000231	5	1007	
TS	DB000231	TP000231	6	1007	
TS	DB000231	TP000231	7	1007	
TS	DB000231	TP000231	8	1007	
TS	DB000231	TP000231	9	1008	
TS	DB000231	TP000231	10	1008	

- VERSION DSNTSTEP3 P140 -			
SC_TIMESTAMP	SU_ENDTS		
2022-04-21-09.54.23.656362	2022-04-21-09.54.24.259606		
2022-04-21-09.56.05.849911	2022-04-21-09.56.06.530834		
2022-04-21-09.54.23.656901	2022-04-21-09.54.24.259606		
2022-04-21-09.56.05.850912	2022-04-21-09.56.06.530834		
2022-04-23-16.30.33.237481	2022-04-23-16.30.33.819598		
2022-04-23-16.30.33.239329	2022-04-23-16.30.33.819598		
2022-04-23-16.30.33.239395	2022-04-23-16.30.33.819598		
2022-04-23-16.30.33.239466	2022-04-23-16.30.33.819598		
2022-04-23-16.34.53.819420	2022-04-23-16.34.54.312316		
2022-04-23-16.34.53.819632	2022-04-23-16.34.54.312316		

Figure 8-4 Query output for table space DB000231.TP000231

In this example, the global variable UH_PS was set to '1'. The results in Figure 8-5 show the most recent **REORG TABLESPACE** execution for each table space partition in database DB000231. The partitions without a row in the result table indicate that a **REORG TABLESPACE** was not recorded in SYSCOPY. Either the partition was never reorganized or the SYSCOPY information about the **REORG** was deleted. The results do not include information for index spaces because either **REORG INDEX** was not executed, the SYSCOPY information about the **REORG** was deleted, or the index spaces were not set with the **COPY YES** attribute.

- VERSION DSNTSTEP3 P140 -					
PAGESET_TYPE	DBNAME	NAME	PARTITION	EVENTID	
TS	DB000231	TG000231	1	1000	
TS	DB000231	TG000231	2	1005	
TS	DB000231	TG000231	3	1000	
TS	DB000231	TG000231	4	1003	
TS	DB000231	TG000231	5	1001	
TS	DB000231	TP000231	6	1004	
TS	DB000231	TP000231	7	1001	
TS	DB000231	TP000231	8	1004	
TS	DB000231	TU000231	9	1005	

- VERSION DSNTSTEP3 P140 -			
SC_TIMESTAMP	SU_ENDTS		
2022-04-19-15.56.23.287381	2022-04-19-15.56.23.894699		
2022-04-19-16.13.44.794485	2022-04-19-16.13.45.242710		
2022-04-19-15.56.23.289388	2022-04-19-15.56.23.894699		
2022-04-19-16.13.44.795792	2022-04-19-16.13.45.242710		
2022-04-19-15.56.26.032134	2022-04-19-15.56.26.896689		
2022-04-19-16.13.47.221166	2022-04-19-16.13.47.857437		
2022-04-19-15.56.26.039842	2022-04-19-15.56.26.896689		
2022-04-19-16.13.47.222288	2022-04-19-16.13.47.857437		
2022-04-19-16.13.49.113884	2022-04-19-16.13.49.402663		

TEP3130 SUCCESSFUL RETRIEVAL OF 9 ROW(S)
TEP315E RETCODE= 0

Figure 8-5 Query output for table spaces and index spaces in database DB000231

8.1.8 Cleaning up the utility history information

The SYSUTILITIES table can be updated, and with proper authorization, you can use the SQL **DELETE** statement to remove the rows that you do not need. The **MODIFY RECOVERY** utility does not support the deletion of SYSUTILITIES rows.

Before you start the cleanup, determine how long you want to keep useful utility history. You might want to keep historical data for certain utilities longer than for others. Assess the size of the SYSTSUTL table space regularly and clean out rows that you no longer need. You can then issue SQL **SELECT** and **DELETE** statements as shown in Example 8-15 on page 126 - Example 8-17 on page 127 to evaluate the rows in SYSUTILITIES and delete the ones that are older than a specified age.

You can modify the following sample SQL statements to meet your needs and remove only those SYSUTILITIES rows that you no longer need. If you use the sample SQL statements to clean up your utility history information, run the statements together in the specified order.

The first query (Example 8-15) generates a summary report of the total number of SYSUTILITIES rows, the number of rows that meet the deletion criteria (older than the specified number of days), and the percentage of rows that meet the deletion criteria.

Example 8-15 Query for a summary report of SYSUTILITIES rows and rows that meet the deletion criteria (older than 365 days)

```

SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_NO_DAYS = 365; -- OLDER THAN 365 DAYS
SELECT B.C1 AS #_QUALIFYING_SYSUTILITIES_ROWS
      ,B.C2 AS #_TOTAL_SYSUTILITIES_ROWS
      ,CASE
        WHEN B.C2 > 0
        THEN
          (CAST(ROUND(
            DECIMAL(B.C1) / DECIMAL(B.C2) * 100, 1) AS DECIMAL(4,1)))
          || '%'
        ELSE 'N/A'
        END
      AS "%_OF_QUALIFYING_ROWS"
FROM
(
  SELECT COUNT(*) AS C1
        ,(SELECT COUNT(*) FROM SYSIBM.SYSUTILITIES) AS C2
  FROM SYSIBM.SYSUTILITIES A
  WHERE
    STARTTS < CURRENT TIMESTAMP - UH_NO_DAYS DAYS
) B;

```

The next query (Example 8-16) produces a detailed report of the SYSUTILITIES rows that meet the deletion criteria (older than the specified number of days specified) when global variable UH_DELETE_PREVIEW is set to '1'. The report also includes the number of SYSCOPY rows that correspond to the SYSUTILITIES rows.

Example 8-16 Query for detailed report of SYSUTILITIES rows that meet the deletion criteria (older than 365 days) and the count of associated SYSCOPY rows

```

SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_DELETE_PREVIEW = '1'; -- PRODUCE DETAILED REPORT OF QUALIFIED ROWS
SET UH_NO_DAYS = 365; -- OLDER THAN 365 DAYS
SELECT A.EVENTID
      ,A.NAME
      ,A.JOBNAME
      ,A.UTILID
      ,A.USERID
      ,CAST(A.STARTTS AS VARCHAR(26)) AS STARTTS
      ,CAST(A.ENDTS AS VARCHAR(26)) AS ENDTS
      ,A.NUMOBJECTS
      ,(SELECT COUNT(*)
        FROM SYSIBM.SYSCOPY B
        WHERE B.EVENTID = A.EVENTID
       ) AS #_SYSCOPY_ROWS
FROM SYSIBM.SYSUTILITIES A
WHERE UH_DELETE_PREVIEW = '1'
      AND STARTTS < CURRENT TIMESTAMP - UH_NO_DAYS DAYS
UNION ALL
SELECT 0 AS EVENTID

```



```

        , 'SYSUTILITIES DETAIL REPORT BYPASSED, UH_DELETE_PREVIEW<>'1''
        AS NAME
        , '-' AS JOBNAME
        , '-' AS UTILID
        , '-' AS USERID
        , '-' AS STARTTS
        , '-' AS ENDTS
        , 0 AS NUMOBJECTS
        , 0 AS #_SYSCOPY_ROWS
    FROM SYSIBM.SYSDUMMY1
    WHERE UH_DELETE_PREVIEW <> '1';

```

To delete the SYSUTILITIES rows that match the deletion criteria (older than the specified number of days), run the following DELETE statement (Example 8-17). Use caution when setting the UH_NO_DAYS global variable because SYSUTILITIES rows older than the specified number of days will be deleted. The count of deleted SYSUTILITIES rows is reported.

Example 8-17 Deleting SYSUTILITIES rows that meet deletion criteria (older than 365 days)

```

SET CURRENT PATH = 'SC000231'; -- SCHEMA FOR GLOBAL VARIABLES
SET UH_DELETE_ROWS = '1'; -- DELETE SYSUTILITIES ROWS
SET UH_NO_DAYS = 365; -- OLDER THAN 365 DAYS
SELECT COUNT(*)
    FROM OLD TABLE
    (
        DELETE FROM SYSIBM.SYSUTILITIES
        WHERE UH_DELETE_ROWS = '1'
        AND STARTTS < CURRENT TIMESTAMP - UH_NO_DAYS DAYS
    ) ;

```

Rerun the first query to generate the summary report to verify that the expected rows were deleted.

8.2 Enhanced inline statistics with page-level sampling (function level 500)

Db2 10 introduced **TABLESAMPLE SYSTEM**, a **RUNSTATS** utility option for page-level sampling that can greatly reduce elapsed times in certain scenarios. Db2 12 enhances this support and enables you to use the **STATPGSAMP** subsystem parameter to make page sampling the default **RUNSTATS** behavior.

Db2 13 further extends the page sampling support. With function level 500 or V13R1M500, you can now use page sampling for gathering inline statistics when running **REORG TABLESPACE** or **LOAD REPLACE** utilities for table spaces. Specify the **STATISTICS** option for **REORG TABLESPACE** or **LOAD REPLACE** if you want to collect inline statistics for the utilities. Make sure that you also specify the **TABLESAMPLE SYSTEM** keyword.

With page sampling enabled, the target pages, which are already in memory, are chosen for inline statistics collection as **REORG** or **LOAD** loads the data during the **RELOAD** phase. This behavior optimizes the execution performance of the utilities and reduces both the elapsed time and the CPU time.

The improvements are greater if **TABLESAMPLE SYSTEM AUTO** (10% sampling rate) is set, which also collects column group frequency statistics (by the **COLGROUP FREQVAL** option) and histogram statistics (by the **HISTOGRAM** option). IBM performance tests on a PBG table with three partitions with a total of 180 million rows and a **DSSIZE** of 4 GB showed a reduction of up to 60% in elapsed time and up to 70% reduction in combined CPU and zIIP time. Similar improvements also are registered when running **LOAD** utilities for inline statistics with **TABLESAMPLE SYSTEM AUTO**.

For more information about **TABLESAMPLE SYSTEM** for inline statistics, see the IBM Documentation for [Syntax and options of the REORG TABLESPACE control statement](#) and the [Syntax and options of the LOAD control statement](#).

8.3 Enhanced space-level recovery (function level 100)

The strategy for data backup and recovery is changing. For example, instead of copying an entire table space in one data set, there is an increasing preference for using partition-level image copies to improve efficiency and reduce costs. The Db2 **RECOVER** utility is continuously enhanced to provide you the needed flexibility and capability as you adapt to the changes. With function level 100 (V13R1M100), Db2 13 enables you to recover at the table space or index space level with partition-level image copies. You can use the **RECOVER** utility to mix the recovery of base image copies that are taken with one, all, or a subset of table space or index partitions.

Before Db2 13, when **RECOVER** was invoked with **DSNUM ALL** for space-level recovery of table spaces, index spaces, or indexes, and the image copies were created at the partition or piece level, Db2 returned error message **DSNU512I (DATASET LEVEL RECOVERY IS REQUIRED)**, indicating that the recovery failed at the space level. Objects with these errors were left unrecovered, and the **RECOVER** operation ended with **RC8**. As a result, you had to retry the recovery of individual partitions or pieces, or specify a **LISTDEF** list with the **PARTLEVEL** option.

In Db2 13, when you specify **RECOVER** for recovering the entire table space or index space, the utility can use any space-level, partition-level, or piece-level image copies for the recovery. These copies include sequential image copies or FlashCopy image copies that are created at the partition or piece level. They also include inline sequential or FlashCopy image copies that are taken by the **LOAD** or **REORG TABLESPACE** utility. This feature is supported on universal table space (UTS), large object (LOB) table spaces, and partitioned indexes.

As shown in Example 8-18, Db2 issues a new message, **DSNU1576I**, when a space-level recovery uses partition-level or piece-level image copies. Each individual image copy that is used as a base for recovery is reported with a **DSNU515I** message.

Example 8-18 Messages in the RECOVER job output

```
DSNU050I    103 13:25:27.36 DSNUGUTC - RECOVER TABLESPACE USERDB.USERTS SCOPE ALL
DSNU1576I  -DB2A 103 13:25:27.38 DSNUCABU - RECOVERY OF TABLESPACE USERDB.USERTS
PROCEEDS AT THE PARTITION OR PIECE LEVEL
DSNU1569I  -DB2A 103 13:25:27.40 DSNUCAIN - RECOVERY LOGPOINT IS
X'00DB59F43197416DD000'
DSNU3345I    103 13:25:27.42 DSNUCBMT - MAXIMUM UTILITY PARALLELISM IS 3 BASED ON
NUMBER OF OBJECTS
DSNU397I    103 13:25:27.42 DSNUCBMT - NUMBER OF TASKS CONSTRAINED BY PARALLEL
KEYWORD TO 1
DSNU427I    103 13:25:27.42 DSNUCBMT - OBJECTS WILL BE PROCESSED IN PARALLEL,
NUMBER OF OBJECTS = 1
```

```

DSNU532I    103 13:25:27.42 DSNUCBMT - RECOVER TABLESPACE USERDB.USERTS DSNUM 2
START
DSNU515I    103 13:25:27.42 DSNUCBAL - THE IMAGE COPY DATA SET
SYSADM.IJ007RR8.USERTS.PO0002IC WITH DATE=20220413 AND TIME=132524
IS PARTICIPATING IN RECOVERY OF TABLESPACE USERDB.USERTS DSNUM 2
DSNU504I    103 13:25:27.82 DSNUCBRT - MERGE STATISTICS FOR TABLESPACE
USERDB.USERTS DSNUM 2 -
                NUMBER OF COPIES=1
                NUMBER OF PAGES MERGED=4
                ELAPSED TIME=00:00:00
DSNU532I    103 13:25:27.82 DSNUCBMT - RECOVER TABLESPACE USERDB.USERTS DSNUM 1
START
DSNU515I    103 13:25:27.82 DSNUCBAL - THE IMAGE COPY DATA SET
SYSADM.IJ007RK4.USERTS.PO0001IC WITH DATE=20220413 AND TIME=132524
                IS PARTICIPATING IN RECOVERY OF TABLESPACE USERDB.USERTS DSNUM 1
DSNU504I    103 13:25:28.24 DSNUCBRT - MERGE STATISTICS FOR TABLESPACE
USERDB.USERTS DSNUM 1 -
                NUMBER OF COPIES=1
                NUMBER OF PAGES MERGED=4
                ELAPSED TIME=00:00:00
DSNU532I    103 13:25:28.25 DSNUCBMT - RECOVER TABLESPACE USERDB.USERTS DSNUM 3
START
DSNU515I    103 13:25:28.25 DSNUCBAL - THE IMAGE COPY DATA SET
SYSADM.IJ007RWG.USERTS.PO0003IC WITH DATE=20220413 AND TIME=132524
                IS PARTICIPATING IN RECOVERY OF TABLESPACE USERDB.USERTS DSNUM 3
DSNU504I    103 13:25:28.68 DSNUCBRT - MERGE STATISTICS FOR TABLESPACE
USERDB.USERTS DSNUM 3 -
                NUMBER OF COPIES=1
                NUMBER OF PAGES MERGED=3
                ELAPSED TIME=00:00:00

```

8.4 REORG INDEX NOSYSUT1 as the default option (function level 500)

When you invoke the **REORG INDEX** utility with **SHRLEVEL REFERENCE** or **CHANGE**, the **NOSYSUT1** keyword or the **REORG_INDEX_NOSYSUT1** subsystem parameter disposes of the need for the utility to use the **SYSUT1** work data set to hold the unloaded index keys. Instead, the utility passes the unloaded keys in memory as input to the index build. Processing the index keys in memory instead of a work data set improves the overall performance of the utility. With this function, the performance is further improved when the utility leverages subtask parallelism to unload and build the index keys concurrently.

The **NOSYSUT1** keyword and the **REORG_INDEX_NOSYSUT1** subsystem parameter were introduced in Db2 12 with APAR PH25217. Starting in Db2 13 and with function level 500 (V13R1M500), the **NOSYSUT1** keyword is set as the default and only behavior for **REORG INDEX** when you specify the **SHRLEVEL REFERENCE** or **CHANGE** keywords. Also, the default value of the **REORG_INDEX_NOSYSUT1** subsystem parameter is changed to **YES**, and the user specification of this subsystem value is ignored.

IBM performance measurements for **NOSYSUT1** showed improvements in elapsed time and CPU time. For partitioned indexes, the tests indicated a reduction of up to 86% in elapsed time. Because the tasks are now all zIIP eligible, a 50 - 95% reduction was observed in non-zIIP CPU processing. For more information about the actual measurement numbers, see “Db2 utilities” on page 277.

8.5 REPAIR WRITELOG dictionary support (function level 100)

Replication products such as Change Data Capture (CDC), IBM InfoSphere® Data Replication Q-Replication (QREP), and SQL Replication (SQLREP), use IFCID 306 processing to read Db2 log data records for source tables. These products replicate the SQL **INSERT**, **UPDATE**, and **DELETE** operations to target tables. If the log records contain compressed data, the corresponding decompression dictionary must be used to expand the data.

The current dictionary is always available in the active source table space. However, if the dictionary is being rebuilt by the **REORG TABLESPACE** or **LOAD** utility, the old dictionary is written to the log where it can be used in IFCID 306 processing for SQL **INSERT**, **UPDATE**, and **DELETE** operations that occurred before the new dictionary was built.

ISV applications that create a compression dictionary might need to write decompression dictionary log records, so Db2 13 introduces dictionary support for **REPAIR WRITELOG**. With function level 100 (V13R1M100), **REPAIR WRITELOG** was extended to accept a new value for the **TYPE** and **SUBTYPE** values. The new value allows the decompression dictionary log record up to the maximum log record size that is supported by Db2. After the decompression dictionary is successfully written, **REPAIR** issues message DSNU3335I (Example 8-19) with the **RBA** or **LRSN** of the log record. Independent software vendor (ISV) applications can use this information to insert an appropriate **SYSIBM.SYSCOPY** record for the dictionary.

Example 8-19 DSNU3335I message

```
DSNU3335I -DB2A 103 13:20:57.19 DSNUCBWL - REPAIR WRITELOG SUCCESSFUL
LRSN/RBA: 00DB59F32FE67CDD3400
MEMBER: DB2A RBA: 000000000001BE7A75FD
```

8.6 Summary

With new and enhanced functions, Db2 for z/OS utilities continue to play a vital role in helping you manage your Db2 data. For more information about Db2 utilities, see the [Db2 utilities](#) and *Db2 for z/OS Utilities in Practice*, REDP-5503.



Instrumentation and serviceability

Db2 for z/OS 13 introduces important enhancements in instrumentation and serviceability that you can leverage. This chapter describes the new and changed instrumentation and serviceability functions.

This chapter describes the following topics:

- ▶ Index management diagnostics and serviceability enhancements
- ▶ Improved EDITPROC support for IFCID 306
- ▶ Improved partition range support for IFCID 306
- ▶ IFCID 369 aggregated accounting statistics is now part of STATISTICS CLASS(1)
- ▶ IFCID 003 longest lock or latch waiter
- ▶ Enhanced distributed thread monitoring
- ▶ Group buffer pool residency time enhancements
- ▶ Page reject of insert row diagnostic log record
- ▶ The STATIME_MAIN subsystem parameter default is now 10 seconds

9.1 Index management diagnostics and serviceability enhancements

Before Db2 13, when data was inserted into a base table, corresponding indexes were modified, which could cause index page splits. Index page splitting affects the performance of SQL insert operations, particularly when an index page set is Group Buffer Pool (GBP)-dependent and requires synchronous I/O for data sharing. Even though there is the existing Instrumentation Facility Component Identifier (IFCID) 359 trace facility (which records information specific to index split events), that information is often insufficient for identifying the exact cause of a performance issue.

Furthermore, IFCID 359 is disabled by default, which causes critical abnormalities during the index split process to go unnoticed, including the total elapsed time of the split that is greater than 1 second. Db2 13 address this concern by adding a IFCID 396 trace facility and three new catalog fields, all of which trace and capture more pertinent information about index page splits. The recorded information helps you identify and analyze the root cause of an **INSERT** performance issue.

9.1.1 New IFCID 396 trace facility for recording abnormal index splitting

The new IFCID 396 trace facility records detailed information that is specific to index page splits, and it is always enabled by default under Stats Class 3 and Performance Class 6. An IFCID 396 record is automatically written when the total time of an index splitting event exceeds the threshold of 1000 ms (1 second), which is considered “abnormal.”

IFCID 396 includes the following fields:

QW0396_Eye CHAR(8)	Eye catcher of IFCID 396.
QW0396DBID CHAR(2)	Database ID (DBID) of the currently splitting index.
QW0396PSID CHAR(2)	Page set ID of the currently splitting index.
QW0396MemID FIXED(16)	Data-sharing Db2 member ID where the split was performed. 0 means non-data sharing.
QW0396PartNum FIXED(16)	Partition number of the currently splitting index.
QW0396URID CHAR(10)	Unit Of Recovery ID (URID) of the unit of recovery that is performing the split.
<blank> CHAR(2)	Unused.
QW0396PageNum FIXED(32)	Page number of the index page that is currently being split.
QW0396TimeStamp CHAR(10)	Current timestamp of the IFCID 396 generation.
QW0396GBPD CHAR(1)	Indicator that the current index page set is marked as GBP-dependent. ‘Y’ = GBP dependent.
<blank> CHAR(1)	Unused.
QW0396ElapseTime FIXED(32)	Total elapsed time of this abnormal index split, expressed in milliseconds.

9.1.2 New fields for recording and aggregating general index split information

New catalog fields REORGTOTALSPLITS, REORGSPPLITIME, and REORGEXCSPLITS were introduced to record and aggregate general index split information after the last run of a **REORG TABLESPACE**, **REBUILD INDEX**, or **LOAD REPLACE** utility. The new catalog fields were added to the SYSIBM.SYSINDEXSPACESTATS and SYSIBM.SYSIXSPACESTATS_H tables.

Table 9-1 shows the new catalog columns.

Table 9-1 New catalog columns

Column name	Data type	Description
REORGTOTALSPLITS	Integer (4 bytes) nullable	The total number of index splits since the last REORG TABLESPACE , REBUILD INDEX , or LOAD REPLACE run.
REORGSPPLITIME	BigInt (8 bytes) nullable	The total or aggregated elapsed time of index splits since the last REORG TABLESPACE , REBUILD INDEX , or LOAD REPLACE run.
REORGEXCSPLITS	Integer (4 bytes) nullable	The total number of abnormal index splits (where the elapsed time exceeds 1 s) since the last REORG TABLESPACE , REBUILD INDEX , or LOAD REPLACE run.

9.2 Improved EDITPROC support for IFCID 306

The IFCID 306 trace facility can read log records for data replication products that process the changes to a table from SQL operations, such as **INSERT**, **UPDATE**, and **DELETE**. If the source table is encoded with an edit procedure (**EDITPROC**), IFCID 306 can locate the **EDITPROC** to decode the log records of the encoded data.

To provide more flexibility, Db2 13 enables you to run IFCID 306 in non-PROXY mode and to turn on or off the **EDITPROC** support function within IFCID 306.

The WQALLOPT field (1 byte) in the qualification area is mapped by DSNDWQAL. It is modified to allow the new values that are shown in Table 9-2.

Table 9-2 Processing options

WQALLOPT value in hex	Value description
X'04'	Requests Db2 to decode the log records by using the EDITPROC function.
X'05'	Requests Db2 to decompress the log records and to decode the log records by using the EDITPROC function.

WQALLOPT value in hex	Value description
X'06'	Requests Db2 to decode the log records by using the EDITPROC function and to convert the log records to the format for the table space version that was current when the data was written.
X'07'	Requests Db2 to decompress the log records, decode the log records by using the EDITPROC function, and convert the log records to the format for the table space version that was current when the data was written.

9.3 Improved partition range support for IFCID 306

High log reader throughput is essential for your replication operations, especially when you must customize and optimize your continuous availability solutions for minimal recovery point objective (RPO) and recovery time objective (RTO). Currently, if you want to retrieve logs from a range of partitions, it might take a long time for IFCID 306 to process your request.

Db2 13 adds more flexibility to IFCID 306 that enables you to use partition filtering as a range of partitions to speed up log read process and increase the throughput for replication operations. This enhancement benefits replication products that run multiple capture programs in which each capture processes different parts of the same table space object in parallel for better end-to-end performance.

A new **WQLSPTRG** flag is defined in the qualification area fields of WQLS. When you turn on this flag, the mapping of WQLSDBPS in WQLS is replaced with the new mapping of WQLSDBPP. The WQLSDBPP mapping contains two new fields that are called WQLSPPART_LOW and WQLSPPART_HIGH, which can be specified only once for each table space.

Table 9-3 lists and describes the qualification area fields of WQLSDBPP. This area is mapped by the assembly language mapping macro DSNDWQAL. This area maps the area that is used to filter IFCID 0306 records when WQLSTYPE is set to 'DBPP'.

Table 9-3 WQLSDBPP fields

Name	Hex offset	Data type	Value description
WQLSPDBID	0	Hex, 2 bytes	DBID of the database.
WQLSPPSID	2	Hex, 2 bytes	Pageset ID (PSID) of the partitioned table space.
WQLSPPART_LOW	4	Hex, 2 bytes	Low partition number of the table space partition range.
WQLSPPART_HIGH	6	Hex, 2 bytes	High partition number of the table space partition range.

For a partitioned table space, you can set WQLSPPART_LOW and WQLSPPART_HIGH to any value of low and high available partition numbers. If WQLSPPART_LOW is '0000'x and WQLSPPART_HIGH is '0000'x or 'FFFF'x, the entire partitioned table space qualifies for the IFCID 306 process.

For a non-partitioned table space, you must set WQLSPPART_LOW and WQLSPPART_HIGH to zero.

9.4 IFCID 369 aggregated accounting statistics is now part of STATISTICS CLASS(1)

IFCID 369 aggregated accounting statistics were introduced in Db2 10 to aggregate accounting statistics per connection type. Then, the statistics class was optional in a Db2 10 system. It was invoked by explicitly issuing a **START TRACE(STAT) CLASS(9)** command to start the collection and aggregation of accounting records.

In Db2 13, IFCID was added to **START TRACE(STAT) CLASS(1)** so that IFCID 369 is started automatically during Db2 startup. The result is that the collection and aggregation of accounting statistics also occurs automatically. To maintain compatibility, **START TRACE(STAT) CLASS(9)** remains unchanged.

9.5 IFCID 003 longest lock or latch waiter

IFCID 0003 thread level accounting now produces the beginning, ending, and elapsed time of the longest lock or latch that the reporting thread was waiting for. The new IFCID 0003 section is at the offset containing the field QWA01REO.

9.6 Enhanced distributed thread monitoring

Two new trace facilities, IFCID 411 and 412, were added, and existing trace facilities 1 and 365 were enhanced. These enhancements help identify applications (IFCID 411) and users (IFCID 412) that either constrain system resources or that do not follow the best practices by using distributed connections. You can use system profile tables to throttle any such applications or users. Applications can be changed to fully leverage the performance features that are available in Db2 13.

9.6.1 New IFCID 411 trace facility for recording DDF application statistics

IFCID 411 records detailed statistics about applications that access the Db2 subsystem by using the DRDA protocol.

The information can help identify applications that are using resources inefficiently in the following types of situations:

- ▶ An application is not releasing connections in a timely manner.
- ▶ An application is causing unexpected thread termination.
- ▶ An application is monitored by a profile but thresholds must be adjusted (too high or low).
- ▶ An application is monopolizing resources (such as opening multiple threads) and should be monitored.

The IFCID 411 fields are shown in Table 9-4.

Table 9-4 IFCID 411 fields

Column name	Data type	Description
QLAPAPPN	CHAR(16)	The name of the application that is running at the remote site.
QLAPPRID	CHAR(8)	The product ID of the remote location from which the remote application connects.
QLAPCOMR	BIGINT	The number of commit requests that are received from the requester (single-phase commit protocol) and committed requests that are received from the coordinator (two-phase commit protocol).
QLAPABRR	BIGINT	The number of abort requests that are received from the requester (single-phase commit protocol) and backout requests that are received from the coordinator (two-phase commit protocol).
QLAPRLV	CHAR(16)	Product level, if known.
QLAPNREST	BIGINT	The number of times that the application reported a connection or application condition from a REST service request.
QLAPNSSR	BIGINT	The number of times that the application reported a connection or application condition from setting a special register through a profile.
QLAPNSGV	BIGINT	The number of times that the application reported a connection or application condition from setting a global variable through a profile.
QLAPHCRSR	BIGINT	The number of times that the application used a cursor that was defined as WITH HOLD and was not closed. That condition prevented Db2 from pooling Database Access Threads (DBATs).
QLAPDGTT	BIGINT	The number of times that the application did not drop a declared temporary table. That condition prevented Db2 from pooling DBATs.
QLAPKPDYN	BIGINT	The number of times that the application used a KEEP DYNAMIC package. That condition prevented Db2 from pooling DBATs.
QLAPHIPRF	BIGINT	The number of times that the application used a high-performance DBAT. That condition prevented Db2 from pooling DBATs.
QLAPHLOBLOC	BIGINT	The number of times that the application had a held large object (LOB) locator. That condition prevented Db2 from pooling DBATs.
QLAPSPCMT	BIGINT	The number of times that a COMMIT was issued in a stored procedure. That condition prevented Db2 from pooling DBATs.
QLAPNTHDPQ	BIGINT	The number of times that a thread that was used by a connection from the application was queued because a profile exception threshold was exceeded.
QLAPNTHDPT	BIGINT	The number of times that a thread that was used by a connection from the application was terminated because a profile exception threshold was exceeded.
QLAPNTHDA	BIGINT	The number of times that a thread that was used by a connection from the application abended.
QLAPNTHDC	BIGINT	The number of times that a thread that was used by a connection from the application was canceled.
QLAPNTHD	INTEGER	The current number of active threads for the application.
QLAPHTHD	INTEGER	For a statistics trace, this number is the highest number of active threads during the current statistics interval; for a READS request, this number is the highest number of active threads since Distributed Data Facility (DDF) started.

Column name	Data type	Description
QLAPTHDTM	BIGINT	The number of threads that were queued because the MAXDBAT subsystem parameter value was exceeded.
QLAPTHDTI	BIGINT	The number of threads that were terminated because the IDHTOIN subsystem parameter value was exceeded.
QLAPTHDTC	BIGINT	The number of threads that were terminated because the CANCEL THREAD command was issued.
QLAPTHDTR	BIGINT	The number of threads that were terminated because a profile exception condition for idle threads was exceeded.
QLAPTHDTK	BIGINT	The number of threads that were terminated because the threads were running under KEEPDYNAMIC refresh rules, and the idle time exceeded KEEPDYNAMICREFRESH idle time limit (20 minutes).
QLAPTHDTF	BIGINT	The number of threads that were terminated because the threads were running under KEEPDYNAMIC refresh rules, and the time that the threads were in use exceeded the KEEPDYNAMICREFRESH in-use time limit.
QLAPTHDTN	BIGINT	The number of threads that were terminated due to network termination.

9.6.2 New IFCID 412 trace facility for recording DDF user statistics

IFCID 412 records detailed statistics about users who access the Db2 subsystem by using the Distributed Relational Database Architecture (DRDA) protocol. The information can help identify users who are using resources inefficiently in the following types of situations:

- ▶ A user is not releasing a connection in a timely manner.
- ▶ A user is causing unexpected thread termination.
- ▶ A user is monitored by a profile but thresholds must be adjusted (too high or low).
- ▶ A user is monopolizing resources (such as opening multiple threads) and should be monitored.

Table 9-5 shows the IFCID 412 fields.

Table 9-5 IFCID 412 fields

Column name	Data type	Description
QLAUUSRI	CHAR(16)	The name of the client user ID under which the connection from the remote application to the local site is established.
QLAUPRID	CHAR(8)	The product ID of the remote application from which the application connects.
QLAUCOMR	BIGINT	The number of commit requests that are received from the requester (single-phase commit protocol) and committed requests that are received from the coordinator (two-phase commit protocol).
QLAUABRR	BIGINT	The number of abort requests that are received from the requester (single-phase commit protocol) and the number of backout requests that are received from the coordinator (two-phase commit protocol).
QLAUPRLV	CHAR(16)	Product level, if known.
QLAUNREST	BIGINT	The number of times that an application that was run by the specified client user ID reported a connection or application condition from a REST service request.

Column name	Data type	Description
QLAUNSSR	BIGINT	The number of times that an application that was run by the specified client user ID reported a connection or application condition from setting a special register through a profile.
QLAUNSGV	BIGINT	The number of times that an application that was run by the specified client user ID reported a connection or application condition from setting a global variable through a profile.
QLAUHCRSR	BIGINT	The number of times that an application that was run by the specified client user ID used a cursor that was defined as WITH HOLD and was not closed. That condition prevented Db2 from pooling DBATs.
QLAUDGTT	BIGINT	The number of times that an application that was run by the specified client user ID did not drop a declared temporary table. That condition prevented Db2 from pooling DBATs.
QLAUKPDYN	BIGINT	The number of times that an application that was run by the specified client user ID used KEEP DYNAMIC packages. That condition prevented Db2 from pooling DBATs.
QLAUHIPRF	BIGINT	The number of times that an application that was run by the specified client user ID used a high-performance DBAT. That condition prevented Db2 from pooling DBATs.
QLAUHLOBLOC	BIGINT	The number of times that an application that was run by the specified client user ID used a held LOB locator. That condition prevented Db2 from pooling DBATs.
QLAUSPCMT	BIGINT	The number of times that a COMMIT was issued in a stored procedure that was called by the specified client user ID. That condition prevented Db2 from pooling DBATs.
QLAUNTHDPQ	BIGINT	The number of times that a thread that was used by a connection from the application that was run by the specified client user ID was queued because a profile exception threshold was exceeded.
QLAUNTHDPT	BIGINT	The number of times that a thread that was used by a connection from the application that was run by the specified client user ID terminated because a profile exception threshold was exceeded.
QLAUNTHDA	BIGINT	The number of times that a thread that was used by a connection from an application that was run by the specified client user ID abended.
QLAUNTHDC	BIGINT	The number of times that a thread that was used by a connection from an application that was run by the specified client user ID was canceled.
QLAUNTHD	INTEGER	The current number of active threads for the application that were run by the specified client user ID.
QLAUHTHD	INTEGER	For a statistics trace, this number is the highest number of active threads during the current statistics interval; for a READS request, this number is the highest number of active threads since DDF was started.
QLAUTHDTM	BIGINT	The number of threads that are associated with the specified client user ID that were queued because the MAXDBAT subsystem parameter value was exceeded.
QLAUTHDTI	BIGINT	The number of threads that are associated with the specified client user ID that were terminated because the IDTHTOIN subsystem parameter value was exceeded.
QLAUTHDTC	BIGINT	The number of threads that were associated with the specified client user ID that were terminated because the CANCEL THREAD command was issued.

Column name	Data type	Description
QLAUTHDTR	BIGINT	The number of threads that were associated with the specified client user ID that were terminated because a profile exception condition for idle threads was exceeded.
QLAUTHDTK	BIGINT	The number of threads that were associated with the specified client user ID that were terminated because the threads were running under KEEP_DYNAMIC refresh rules, and the idle time exceeded the KEEP_DYNAMIC_REFRESH idle time limit (20 minutes).
QLAUTHDTF	BIGINT	The number of threads that were associated with the specified client user ID that were terminated because the threads were running under KEEP_DYNAMIC refresh rules, and the time that they were in use exceeded the KEEP_DYNAMIC_REFRESH in-use time limit.
QLAUTHDTN	BIGINT	The number of threads that were associated with the specified client user ID that were terminated due to network termination.

9.6.3 New fields for the IFCID 365 trace facility for recording DDF location statistics

New fields were added for counting Database Access Threads from remote locations that were terminated. This new information can help identify locations that are using resources sub-optimally.

Table 9-6 shows these new fields.

Table 9-6 IFCID 365 fields

Column name	Data type	Description
QLSTNTPLH	BIGINT	The number of high-performance DBATs that were terminated after remaining in a pool longer than POOLINAC.
QLSTNTILS	BIGINT	The number of DBATs that were terminated when the TCP socket was closed due to connection loss.

9.6.4 New fields added to the IFCID 1 trace facility for recording global DDF

New fields were added for counting Database Access Threads that remained active or were terminated at the server. This new information can help analyze resource utilization at the server.

Table 9-7 shows these new fields.

Table 9-7 IFCID 1 fields

Column name	Data type	Description
QDSTNAKD	INTEGER	The current number of DBATs that are active due to usage of packages that are bound with KEEP_DYNAMIC yes.
QDSTMAKD	INTEGER	The maximum number of DBATs that are active due to usage of packages that are bound with KEEP_DYNAMIC yes.
QDSTNDBT	INTEGER	The number of DBATs that were terminated since DDF was started.

Column name	Data type	Description
QDSTNTPL	INTEGER	The number of DBATs that were terminated after remaining in a pool longer than POOLINAC.
QDSTNTRU	INTEGER	The number of DBATs that were terminated after being reused more times than the limit.

9.7 Group buffer pool residency time enhancements

IBM z16 introduces the ability to track data and directory residency statistics for GBPs. These statistics provide a way to see how long data stays in the buffer pool before being removed. This information can help adjust workload balance and GBP sizing to improve performance. This feature is available in all Db2 13 subsystems that meet the following criteria:

- ▶ Db2 running z/OS release 2.4 (HBB77C0) or 2.5 (HBB77D0) with necessary PTFs for cache residency time metrics (for more information about the PTFs, see APAR OA60650).
- ▶ Coupling facility (CF) running on a CF Level 25 z16 machine or higher

Two new fields were added to the relevant GBP statistics storage areas:

- ▶ The average time in milliseconds that a data area stays in a GBP before it is removed.
- ▶ The average time in milliseconds that a directory entry stays in a GBP before it is removed.

Both metrics are accessible through more fields in IFCID 230 and 254 trace records and by using the **-DISPLAY GROUPBUFFERPOOL** command.

Table 9-8 and Table 9-9 show the new fields that were added to IFCID 230 and IFCID 254.

Table 9-8 IFCID 230 fields

Column name	Data type	Description
QBGBART	CHAR(8)	“Data Area Residency Time” is the weighted average in microseconds of the elapsed time that a data area stays in a GBP before it is reclaimed.
QBGBERT	CHAR(8)	“Directory Entry Residency Time” is the weighted average in microseconds of the elapsed time that a directory entry stays in a GBP before it is reclaimed.

Table 9-9 IFCID 254 fields

Column name	Data type	Description
QW0254AR	CHAR(8)	“Data Area Residency Time” is the weighted average in microseconds of the elapsed time that a data area stays in a GBP before it is reclaimed.
QW0254ER	CHAR(8)	“Directory Entry Residency Time” is the weighted average in microseconds of the elapsed time that a directory entry stays in a GBP before it is reclaimed.

9.8 Page reject of insert row diagnostic log record

Db2 13 adds more information to the existing page reject of insert row diagnostic log record: record type '4400'X and subtype '0106'X. Each log record now contains one or more of the following sections:

- ▶ Data page rejection information.

As an insert transaction searches through data pages but fails to insert a row, the page is saved in memory. Only a subset of the visited pages is displayed in the first section of this log record.

- ▶ Partition retry information.

The partition retry process applies only to partition-by-growth (PBG) table spaces that contain more than one partition. It is the process of retrying partitions that failed to obtain a partition lock previously during cross-partition search. This section of the log record contains partition retry information. An eyecatcher of 'PE' indicates the start of this section on the log record.

- ▶ Space map page search footprints

This section contains a subset of the space map pages that were searched for this insert transaction. An eyecatcher of "SRCH" indicates the start of this section.

The improved diagnostic log record is written when one of the following conditions is met:

- ▶ When a space search of an insert transaction visits more than 120 data pages but fails to insert a row.
- ▶ When the insert transaction fails to find space for inserting a row and insert also performs cross-partition retry on a partition but fails with a conditional partition lock.
- ▶ The insert transaction causes lock escalation.

9.9 The `STATIME_MAIN` subsystem parameter default is now 10 seconds

Before Db2 13, the default setting of the `STATIME_MAIN` subsystem parameter was 60 seconds, which made it difficult for DBAs and Db2 system programmers to identify true workload peaks by using Db2 statistics for subsystem-level performance tuning and planning. Similarly, when a slowdown in the range of 5 - 15 seconds occurred, diagnosing the problem became difficult with a 60-second default interval.

To support the Db2 13 strategy for First Failure Data Capture (FFDC), a more granular default setting of 10 seconds was implemented for the `STATIME_MAIN` subsystem parameter. This new default value causes data to be collected more frequently instead of asking customers to re-create their problem. This serviceability enhancement can help to speed up problem diagnosis by having more data readily available.

9.10 Summary

Db2 13 for z/OS introduced many improvements in instrumentation and serviceability. The new IFCID 396 captures more information during abnormally long index split processing. Improved and more flexible support for IFCID 306 was introduced to help replication products process more efficiently and with better performance. IFCID 369 is now enabled by default when STASTICS CLASS(1) starts. Information about the longest lock or latch is now captured in IFCID 003. Distributed thread monitoring and the page reject of insert row diagnostic logging also was enhanced. A new default was implemented for the **STATIME_MAIN** to more frequently collect diagnostic information. Lastly, time tracking was added for GBP Residency. All these many changes to the instrumentation and serviceability functions in Db2 13 for z/OS should provide better insights and optimization that were not possible before.



Machine learning and artificial intelligence infusion

As an industry-leading database for mission-critical data, IBM Db2 for z/OS has transformed itself into an AI-powered enterprise data management system for the cognitive era. As a cognitive database, Db2 for z/OS infuses AI capabilities into its core engine and builds up a vibrant ecosystem with top-notch, end-to-end, and enterprise-ready machine learning and AI solutions.

IBM Watson® Machine Learning for z/OS (WMLz) and IBM Db2 AI for z/OS (Db2ZAI) are two of the solutions that integrate seamlessly with your Db2 for z/OS system. This chapter introduces WMLz and Db2ZAI, highlights their key features and capabilities, and describes the use cases or opportunities for using them in your Db2 for z/OS environment. It shows that together with Db2 for z/OS, WMLz and Db2ZAI can transform the way you harness the power of your data and gain actionable insights, which ultimately help you achieve greater speed to market and higher return on investment (ROI).

This chapter describes the following topics:

- ▶ IBM Watson Machine Learning for z/OS
- ▶ Db2 AI for z/OS

10.1 IBM Watson Machine Learning for z/OS

WMLz is an enterprise machine learning solution with components that run on IBM Z or Red Hat OpenShift Container Platform (RHOCP). You can run WMLz as a stand-alone solution or infuse it into your enterprise AI capability as a scalable platform.

WMLz consists of a required base component (the WMLz base) and an optional integrated development environment (IDE) component (the WMLz IDE). As Figure 10-1 shows, the WMLz base runs on z/OS and leverages IBM machine learning capabilities on IBM Z, including IBM Open Data Analytics for z/OS (IzODA). IzODA serves as the data processing cluster for WMLz and delivers advanced data analytics through z/OS Spark (Spark), z/OS Anaconda (Anaconda), and Mainframe Data Service (MDS).

Although Spark provides a best-of-class analytics engine for large-scale data processing and Anaconda supplies a wide range of popular Python packages for model training, MDS connects those data processing engines to your enterprise data sources on IBM Z, such as Db2 for z/OS, Information Management System (IMS), Virtual Storage Access Method (VSAM), and System Management Facilities (SMF). In addition to being one of the primary data sources, the WMLz base also uses Db2 for its repository service.

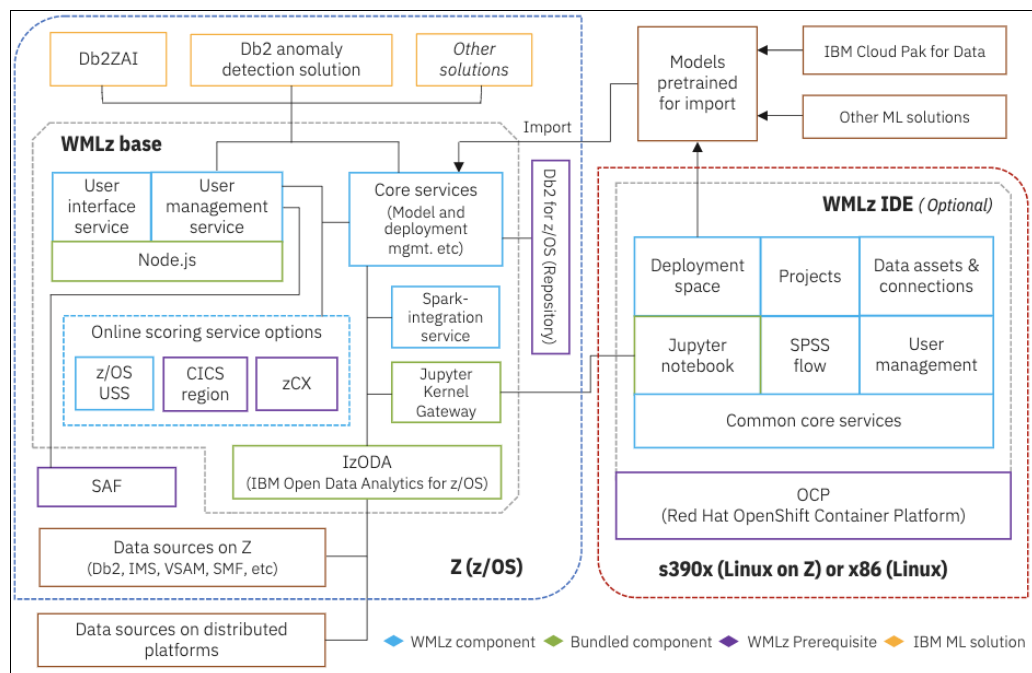


Figure 10-1 The WMLz architecture

The optional IDE runs on RHOCP for s390x and x86 64-bit servers. Built on Red Hat Enterprise Linux and Kubernetes, RHOCP provides a secure, scalable, and multi-tenant operating system and a range of integrated application run times and libraries. The IDE also leverages IBM machine learning capabilities in Watson Studio and IBM SPSS® Modeler.

Both WMLz base and IDE feature a web user interface (UI) and various APIs. The base also includes a configuration tool and an administration dashboard, as shown in Figure 10-2 on page 145. Together, the interfaces provide a powerful suite of easy-to-use tools for model development and deployment, user management, and system administration.

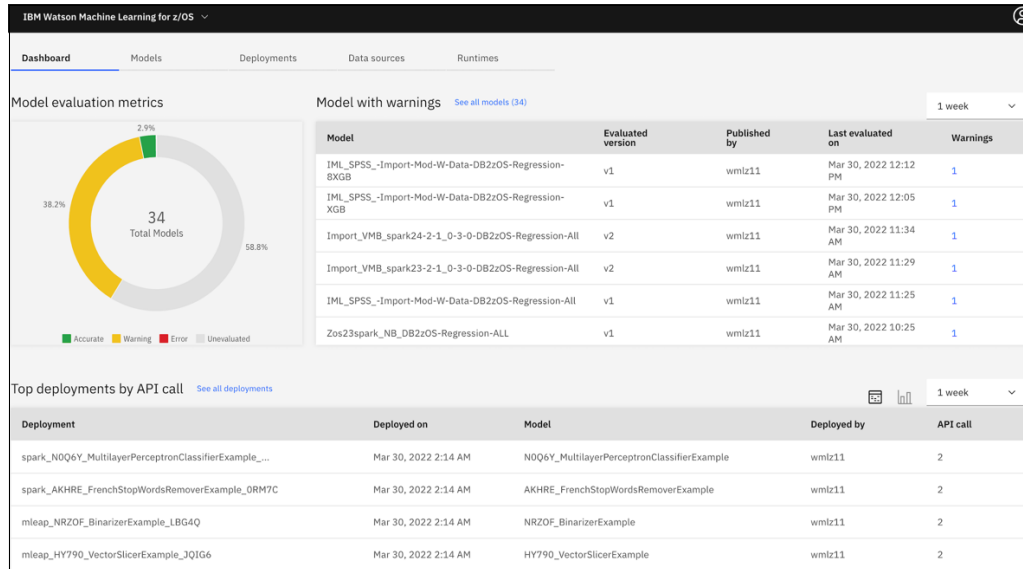


Figure 10-2 The WMLz base dashboard

The various tools, run times, and libraries simplify the machine learning operations for your key personas, including system administrators, machine learning administrators, data scientists, and application developers. For example, system administrators can use the administration dashboard to keep the services on the base running, and machine learning administrators can deploy machine learning models as services to IBM Customer Information Control System (CICS), Java, and other applications on IBM Z. The machine learning administrators can also import models that are trained in the IDE into the base for deployment and management. Application developers can use the deployments and services in their applications to make real-time predictions.

10.1.1 Key features and capabilities

As an enterprise-grade machine learning solution on IBM Z, WMLz provides a range of key features and capabilities.

Secure IBM Z platform for running machine learning with data in place

WMLz provides state-of-the-art machine learning technology behind the firewall on IBM Z, the world's most secure computing platform, and sets the new standard for data gravity or data movement at a large scale. Most cognitive solutions require that data is moved off their on-premises or private cloud, exposing the data to potential breaches and introducing significant operational costs. IBM recognizes the imperative of data gravity and provides solutions that bring cognitive capabilities to the data. WMLz offers the unique ability for you to work directly with data securely in place on IBM Z. By keeping your mission-critical data at the source, WMLz delivers in-transaction predictive analytics with real-time data and minimizes the latency, cost, and complexity of data movement.

Enterprise-grade performance, availability, and reliability

Integrating closely with hardware and software that make up the IBM Z stack, WMLz leverages the strength of IBM Z technologies to provide the highest level of availability, reliability, and scalability for machine learning services. Like Db2 for z/OS, most applications running on IBM Z process and generate transactional data. High availability (HA) and reliability are a key requirement for enterprise machine learning on IBM Z. As Figure 10-3 shows, WMLz takes this requirement to heart in the architecture of its scoring service.

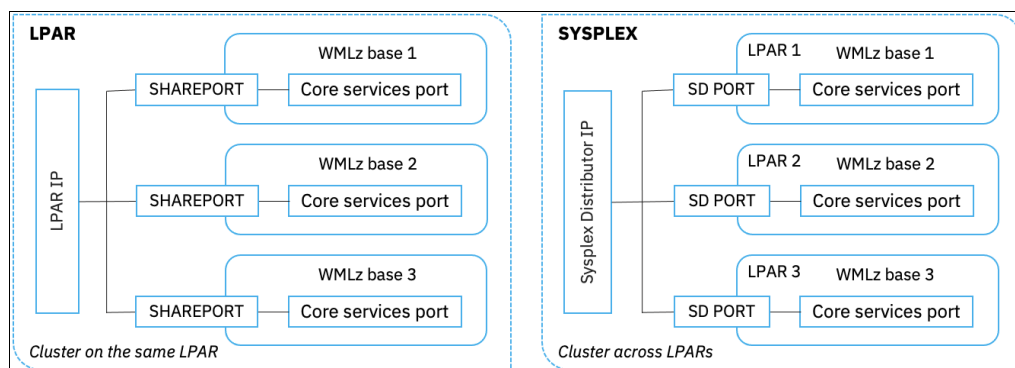


Figure 10-3 The WMLz base cluster configuration for high availability

Taking full advantage of the IBM Z sysplex, dynamic virtual IP address (VIPA), and Shareport technologies, WMLz ensures HA of a stand-alone scoring service or a scoring service cluster by running the services natively on a single logical partition (LPAR) or across multiple LPARs. WMLz also runs its online scoring service in CICS efficiently to meet service level agreements (SLAs) for transaction processing. Transactions that are running in CICS call the scoring service directly through CICS LINK, which eliminates the latency of network communication.

End-to-end machine learning platform for full lifecycle model management

Unlike most machine learning frameworks, WMLz recognizes that machine learning should not focus only on the creation of models. Instead, it must provide the complete workflow of data collection and preparation, feature engineering, model development and training, model deployment, monitoring, and retraining. WMLz also recognizes that the accuracy of a model is at the heart of a machine learning pipeline and that model accuracy can degrade over time with new data, which is why WMLz builds its workflow to form a continuous learning loop, as shown Figure 10-4.

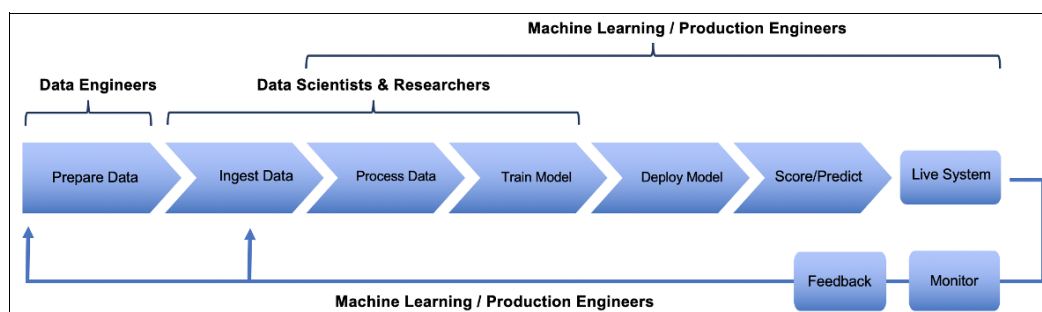


Figure 10-4 The WMLz model lifecycle management

A model is continuously monitored and retrained to improve its accuracy based on data from live transactions. The complete learning loop architecture of WMLz makes the full lifecycle management of a model possible. The model is optimized as the data is pushed back into the machine learning workflow in a feedback loop.

Real-time insights from your mission-critical data

The ability to run analytics and extract insights in real time is another key capability of WMLz. The best way to extract real-time insights is to run the analytics against transactions where and when they happen. WMLz runs on the same IBM Z system where transaction systems and transactional data live.

Real-time prediction with minimal impact to transaction processing are key to machine learning operations on IBM Z. WMLz addresses this requirement in the architecture and implementation of its online and batch-scoring services by leveraging IBM Z technologies. For example, with WMLz, you can configure the online scoring service in CICS, where most data transaction and processing happen on IBM Z. This flexibility ensures fast processing of scoring requests to meet SLAs for transaction processing. To ensure HA, WMLz leverages TCP/IP port sharing if the scoring service cluster is on a single LPAR or the sysplex distributor if the cluster spans across multiple LPARs.

Flexible choice of machine learning languages and frameworks

Different data scientists and machine learning engineers prefer different programming languages or machine learning frameworks. Like most IBM cognitive systems, WMLz supports many popular machine learning languages, including Python and Scala, and scoring engines, such as SparkML, Scikit-Learn, Predictive Model Markup Language (PMML), eXtreme Gradient Boosting (XGboost), and Open Neural Network Exchange (ONNX). This flexibility to choose the language and framework to work with makes WMLz easy to adopt.

Developer-friendly API interfaces for applications on IBM Z and RHOCP platforms

WMLz provides a large collection of developer-friendly REST, Scala, and Java APIs for applications, including COBOL, which run on IBM Z and RHOCP platforms. This capability drastically reduces the amount of effort that is required for application developers to apply predictive capabilities to transactions running on IBM Z. For example, CICS application developers can deploy WMLz online scoring service to the IBM WebSphere® Liberty (Liberty) server. With the CICS native **LINK** command support for COBOL applications, the input/output of the scoring request is passed through CICS channels, which is more efficient than REST calls on the same LPAR.

Interoperable models that can be trained and deployed anywhere

WMLz can process data on IBM Z, but does not confine its use on IBM Z. With the support of machine learning frameworks, you can use WMLz to train, save, publish, and deploy models as SparkML, Scikit-learn, and XGBoost types. WMLz also supports models in PMML and ONNX, two of the most popular model exchange formats. You can import a model that is trained on another platform and save and deploy the imported model as a PMML model type. WMLz also provides a unified runtime environment on IBM Z for deep learning models in ONNX format. You can import your long short-term memory (LSTM) and recurrent neural network (RNN) models in ONNX format, and the built-in scoring engine of WMLz processes and optimizes the models for in-transaction scoring on IBM Z.

With these industry-leading capabilities, WMLz helps you maximize the value of your mission-critical data in Db2 for z/OS and other sources. By keeping your data where it originates and your AI capabilities where your applications run, WMLz helps reduce the cost, security risk, and time to turn your raw data into valuable insights.

10.1.2 Latest WMLz updates

After becoming generally available, WMLz continuously rolls out new and enhanced features through timely releases, which together address the needs of data scientists, application architects, application developers, and machine learning engineers. The latest release leverages the advances in both IBM Z hardware and z/OS software technologies and introduces new and enhanced capabilities.

Advanced processing of deep learning ONNX models

With the latest release, the WMLz ONNX scoring engine runs natively on z/OS, which improves scoring performance and simplifies application development. You can embed the scoring engine in a CICS region so that application developers can make CICS **LINK** calls to the scoring service. In addition, WMLz extrapolates the micro-batching inferencing capability to the online ONNX scoring processing. Instead of inferencing one record at a time, the ONNX scoring engine can queue up scoring requests so that more records can be inferenced together, which improves inferencing performance and reduces resource consumption.

Application Transparent Transport Layer Security support with z/OS TLS/SSL protection and IBM Z cryptography

WMLz uses SSL to secure communications between its services and client applications. WMLz strengthens its communication security by leveraging the Application Transparent Transport Layer Security (AT-TLS) capability on z/OS. With the AT-TLS support, your WMLz services can communicate securely with client applications by taking full advantage of the z/OS policy-based TLS/SSL protection and the IBM Z hardware cryptography technology.

Integration with IBM Watson OpenScale

IBM Watson OpenScale is an enterprise-grade environment for AI applications. WMLz uses the Watson OpenScale service on IBM Cloud Pak for Data to monitor the quality of its models, analyze their fairness, and explain their results. Before the integration, you must enable your application with complex automatic payload logging for the WMLz scoring engine. The integration removes that complexity and adds a built-in payload logging capability.

10.1.3 Opportunities for using WMLz

Powered by unmatched capabilities and combined with your enterprise data, WMLz can help you anticipate market trends and customer needs, solve business problems as they arise, optimize your decision-making, minimize risks and costs, and grow top-line revenue by capitalizing on insights from real-time transaction data. As exemplified in the following use cases, you can apply WMLz in your segment of industry or your line of business. Using WMLz can help you harness the unprecedented business opportunities and capture the enormous benefits from the most advanced machine learning solution on IBM Z.

Reducing customer churn in banking

Customer churn in banking refers to the situation when customers end their relationship with a bank by closing their accounts and discontinuing all services. A high churn rate can cause severe financial losses. Losing existing customers is costly, and acquiring new ones is even more so. To retain customers and reduce costs, it is important for banks to predict and identify customers with high attrition risk and take proactive actions to mitigate this risk. If your historical churn data is stored in Db2 for z/OS or other sources on IBM Z, consider using WMLz to visualize the data, build a churn data model, score the model to predict potential churn rate, and identify the demographics of likely churners.

Providing a customer investment advisory

Investment advisors strive to give the right recommendations that help clients make the right investment decisions. With the changing dynamics in financial markets and the inherent risks in investment, making sound recommendations requires time-sensitive intelligence that is often hidden in raw data that is scattered in multiple locations. By using WMLz, you can analyze the industry affinity of your clients and market trends no matter where the data is stored, and quickly extract insights for the most suitable investment recommendations.

Analyzing credit risk and minimizing loan defaults

Loans are a major source of income for banks. Good loans bring tremendous profits, but bad loans often mean significant financial losses. Loan default or delinquency is one of the most common types of bad loans. Loan default occurs when a borrower fails to make payments on time. To minimize loans in default, banks must carefully analyze each application and accurately determine the creditworthiness of an applicant.

WMLz can help satisfy this requirement and optimize your loan decision-making. Instead of relying exclusively on explicit rules, WMLz can use the historical data of past loans and approvals as a base and a range of algorithms to uncover implicit patterns in a vast amount of application data and predict the risk of default before approving a new loan.

Eliminating fraud in government benefit programs

Fraud is endemic across industry sectors and government agencies. As businesses and governments move their operations online, fraud-related crimes become even more rampant and sophisticated, and fraud losses are increasingly crippling. Fighting fraud in government agencies is an uphill battle. The perpetrators use evolving technologies and change their tactics at will. Effective fraud detection requires the computational capability to quickly analyze a vast amount of data, accurately identify hidden patterns of fraudulent behaviors, and swiftly turn them into intelligence that enables organizations to make real-time decisions. WMLz can deliver that industry-leading capability.

Detecting system anomalies and resolving issues before they arise

Enterprises worldwide use IBM Z hardware and software, such as Db2 for z/OS, to support their mission-critical processes. These IBM Z servers and solutions can collect and save extensive operation and performance data. You can use this historical data to build machine learning models for monitoring system operations and predict issues before they arise. You can accomplish this task by leveraging the IT Operations Analytics (ITOA) function of either WMLz or Db2ZAI.

10.2 Db2 AI for z/OS

Db2ZAI leverages machine learning to determine the existence of patterns in your unique operating environment to improve the performance of your Db2 for z/OS applications.

Built on top of Watson Machine Learning for z/OS, Db2ZAI enables the Db2 for z/OS optimizer to leverage machine learning capabilities to reduce CPU consumption and IT costs.

Db2ZAI provides three main use cases for improving your Db2 for z/OS applications: SQL optimization, system assessment, and distributed connection control.

Figure 10-5 shows the **Overview** tab of the Db2ZAI dashboard.

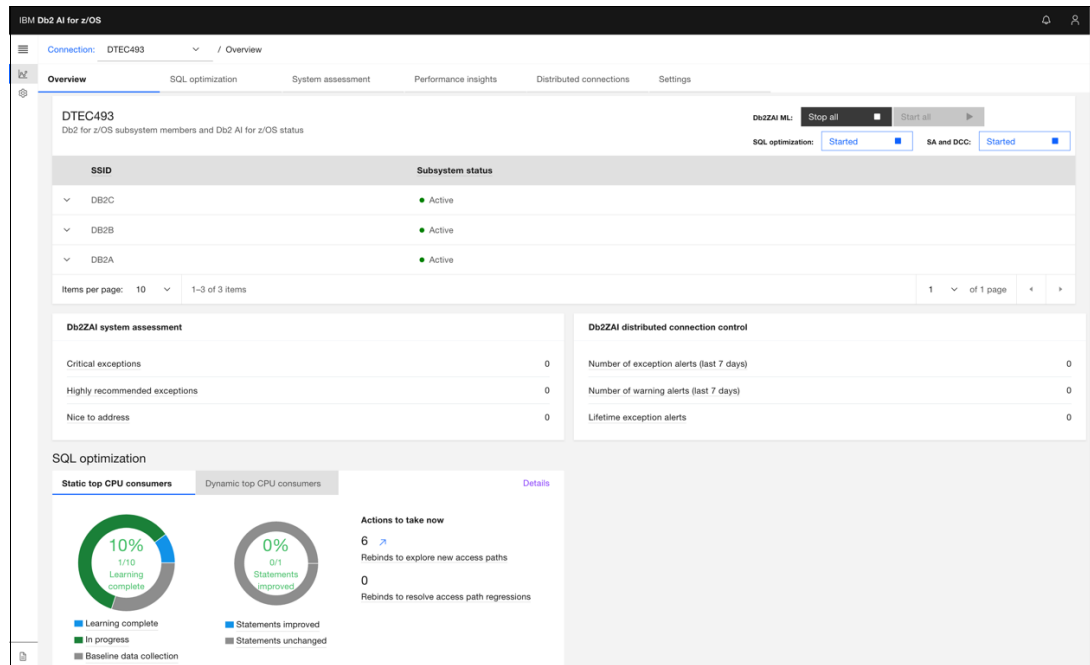


Figure 10-5 Db2ZAI overview dashboard

10.2.1 SQL optimization

The Db2ZAI SQL optimization feature brings machine learning to bear to improve the performance of your queries. It does this task by helping the Db2 optimizer to choose the most efficient access paths.

Db2ZAI predicts the likely behavior of your SQL (for example, host variable values and the number of rows that is fetched by the query) and uses this information to create better access paths.

How it works

Db2ZAI begins by learning the baseline access paths and performance of your SQL. It also learns the inputs to the Db2 Optimizer based on the runtime behavior of your queries.

By learning the values that are used by your host variables, Db2ZAI helps to achieve better filter factor estimation. With better filter factor and application behavior, the optimizer can choose better access paths. Another source of information for SQL optimization is screen-scrolling application behavior.

Benefits of Db2ZAI SQL optimization

The Db2ZAI SQL optimization feature provides the following main benefits:

- ▶ It assists the Db2 optimizer to choose more efficient access paths, which results in better query performance.
- ▶ It automates access path performance regression detection and resolution, which minimizes the impact of any access path regression that occurs.
- ▶ It automatically stabilizes simple dynamic statements, which helps to avoid the overhead from statement prepares.

10.2.2 System assessment

The Db2ZAI system assessment feature leverages AI to learn the normal performance characteristics of your system, and then it analyzes exceptions to provide recommendations so that you can tune for better performance. You can define various workload profiles to define the expected metrics within a particular work period.

How it works

The Db2ZAI system assessment feature learns from your system's historical data to understand the specific workload characteristics of the system.

It automatically collects the key system performance and workload metrics, and then it provides an exception analysis that is based on the learned baseline thresholds. Then, it generates a set of recommended actions to take (including current related zparm or buffer pool settings) and considerations for making updates.

Benefits of Db2ZAI system assessment

The Db2ZAI system assessment feature provides the following key benefits:

- ▶ It provides a list of exceptions and detailed, expert-recommended Db2 system-tuning actions.
- ▶ It provides detailed, specific recommendations to address performance issues.
- ▶ It provides an extensive number of interactive visualizations of system performance to help the identification of assessment results. The visualizations provide an easy way to see similarities and differences across members of your Db2 data-sharing group.

10.2.3 Distributed connection control

The Db2ZAI distributed connection control (DCC) feature helps you monitor connections and threads and provides a mechanism for managing them across your remote clients.

The DCC feature addresses many of the challenges that are faced by Db2 system administrators in managing their inbound network traffic.

For example, Db2 administrators lack knowledge about remote application behavior, and consequently lack protection from connection flooding and system level impacts.

In addition, Db2 administrators sometimes lack knowledge about how to set up the Db2 mechanisms that are used to control connections. Setting connection controls too low can cause an application outage, but setting connection controls too high can reduce their effectiveness.

How it works

The DCC feature uses the Instrumentation Facility Component Identifier (IFCID) 365, 402, 411, and 412 data that is collected by Db2. When you initiate it, DCC trains itself by using the IFCID 365 data to generate recommended connection profiles.

When training is complete, you review training data and the recommended Db2 profiles, and make any necessary edits. Finally, you activate the recommended profiles to enable protection.

When profile exceptions occur, alerts are raised in the Db2ZAI user interface and email alerts are sent to notify you about the situation. Then, you can review the historical statistical data to help you determine the correct course of action to resolve the exception.

Benefits of Db2ZAI distributed connection control

The Db2ZAI DCC feature provides the following key benefits:

- ▶ It automatically learns about connection and thread behavior.
- ▶ It identifies and warns you about problems before they cause an impact.
- ▶ It improves the visibility of your Db2 remote client connections.
- ▶ It uses learned behavior to identify why threads are getting blocked.
- ▶ It uses profile recommendations to prevent connection flooding from impacting other applications.

10.2.4 What is new in Db2ZAI V1.5

Db2ZAI V1.5 includes the following enhancements:

- ▶ Unique operational insights and deeper drill-down into Db2 system health:
 - It provides a quick overview of Db2 health with performance at-a-glance and deeper drill-down capability for root cause analysis on the timeline for each connection type.
 - It provides further drill-down into distributed connection and thread usage by IP address, client user ID, and application name.
- ▶ Enhanced, distributed thread management, including the ability to filter client information beyond IP addresses.

More profile recommendations based on client user IDs and application names are generated to improve the usability of distributed connection control.
- ▶ Aggregated accounting information provides more application-level insights for exception and correlation analysis in system assessments.
- ▶ Support for packages with multiple versions in SQL optimization.

10.3 Summary

WMLz and Db2ZAI are enterprise-grade machine learning and AI solutions that are designed to work with your Db2 for z/OS system. WMLz simplifies and automates the complex machine learning workflow so that enterprises like yours can quickly integrate machine learning into your operations. WMLz provides the development and runtime environment and the full lifecycle management capability that enable your business to identify hidden patterns from your mission-critical data, build models from those patterns, embed the models in your applications, and run scoring and predictions while ensuring the accuracy of the models over time.

Db2ZAI, which is powered by the WMLz platform, is a self-tuning and self-management solution that brings AI capabilities directly to your Db2 engine. The AI-infused SQL optimization feature collects data from your Db2 optimizer and the query execution history to determine the best-performing query access paths based on your workload characteristics. The system assessment feature is a perfect union between the expertise of subject matter experts (SMEs) and the insight of machine learning. Db2ZAI detects Db2 system performance exceptions or unusual system performance events, and then it provides a set of recommended actions for you to ameliorate the situation. Lastly, the distributed connection control feature helps you identify, monitor, and adjust connections and threads for inbound, distributed connections across your Db2 systems.

For more information about WMLz and Db2ZAI, see the following resources:

- ▶ [WMLz product details](#)
- ▶ [WMLz technical documentation](#)
- ▶ [Db2ZAI product details](#)
- ▶ [Db2ZAI technical documentation](#)



Development and administration tools

Developed in parallel with Db2 13, there are several new tools to simplify the Db2 13 ecosystem for application developers and database administrators at all skill levels. These tools help experienced Db2 users perform their tasks more quickly and efficiently and reduce the amount of in-depth Db2 for z/OS knowledge that new users need to become productive.

This chapter describes the following topics:

- ▶ IBM SQL Tuning Services
- ▶ IBM Db2 for z/OS Developer Extension
- ▶ IBM Db2 Administration Foundation for z/OS

11.1 IBM SQL Tuning Services

IBM SQL Tuning Services consists of a set of RESTful APIs that enable you to analyze and tune SQL applications that work with Db2 for z/OS.

Although they are available independently, the APIs are also integrated directly within Db2 Developer Extension to enable application developers to tune their code and within Db2 Administration Foundation for z/OS (Admin Foundation) to enable administrators to tune SQL that is identified through the catalog, statement cache, and other query monitoring integrations. Specifically, the APIs are provided for the following tasks:

- ▶ Retrieve SQL queries from several sources.
- ▶ Generate a visual representation of access plans for an SQL statement by using Visual Explain to enable you to more easily analyze the access path that was chosen for queries.
- ▶ Generate **RUNSTATS** recommendations to improve the information that is available to the optimizer so that the cost of running queries can be evaluated more accurately based on different access path choices.
- ▶ Capture the environment in which an SQL statement runs for diagnostic purposes.
- ▶ Perform administrative functions, such as managing **EXPLAIN** tables and jobs.

SQL Tuning Services is a component of the IBM Database Services Expansion Pack feature, which is included with Db2 Accessories Suite for z/OS.

11.1.1 SQL Capture API

The SQL Capture API retrieves queries from various sources, including the statement cache; plans; packages; stabilized dynamic queries that are stored in Db2 catalog tables; queries from tables and views that are stored in a user-defined repository; and queries that are stored in an uploaded file. The output of the SQL Capture API is used as input to other SQL Tuning Services APIs.

11.1.2 Visual Explain API

The Visual Explain API generates a visual representation of access plans for a query and returns the results in the form of a URL that you open in a browser. The information that this API produces is based on information that is stored in Db2 **EXPLAIN** tables.

You can use the Visual Explain API to gain the following insights about a query:

- ▶ Determine whether an index was used to access a table. If an index was not used, Visual Explain can help you determine which columns might benefit from being indexed.
- ▶ Easily examine table join sequence and methods and determine whether a sort is used.
- ▶ View the effects of performing various tuning techniques by comparing the before and after versions of the access plan graph for a query.
- ▶ Obtain information about each operation in the access plan, including the total estimated cost and number of rows that is retrieved (cardinality).
- ▶ View the statistics that were used at the time of optimization. Then, you can compare these statistics to the current catalog statistics to help you determine whether rebinding the package might improve performance.

11.1.3 Statistics Advisor API

The Statistics Advisor API generates a recommended set of **RUNSTATS** commands that you can issue to improve query performance. The recommendations that this API produces are based on Db2 optimizer-generated statistics recommendations that are stored in Db2 **EXPLAIN** tables.

11.1.4 Query Environment Collector API

The Query Environment Collector API captures diagnostic information about the environment in which you are running an SQL statement and stores that information in a downloadable file. This information is useful when you are working with IBM Support to resolve performance problems with an SQL statement or any other problem for which IBM Support needs detailed information about your environment.

11.1.5 Administrative APIs

In addition to the previously mentioned functional APIs, SQL Tuning Services also includes the following administrative APIs, which help with setting up and administering an SQL Tuning Services environment and running tuning actions:

- ▶ Authentication Service API
- ▶ Repository Database Setup API
- ▶ Connection Profile Management API
- ▶ User Privilege Management on Connection Profile API
- ▶ EXPLAIN Tables Management API
- ▶ Job Management API

11.2 IBM Db2 for z/OS Developer Extension

IBM Db2 for z/OS Developer Extension modernizes the process of writing, running, tuning, and debugging Db2 for z/OS applications by bringing Db2 for z/OS SQL support to Microsoft Visual Studio Code.

IBM Db2 Developer Extension provides language support for Db2 for z/OS SQL, including embedded statements in non-SQL file types, such as COBOL, Python, Java, and Node.js, so that application developers can use familiar programming languages and their preferred development environment to interface with Db2 for z/OS data.

Additionally, if you also use the Zowe Explorer extension for Visual Studio Code, you can use the Db2 Developer Extension capabilities on SQL that exists directly on z/OS.

11.2.1 Integrating with Db2 13

Db2 Developer Extension 1.4 was developed in parallel with Db2 13 to ensure day-one support for new SQL elements that were introduced in Db2 13. This support includes syntax highlighting, syntax checking, formatting, code completion, and signature help for Db2-supplied stored procedures and built-in functions.

Db2 Developer Extension also includes a database connection wizard that makes it easy to connect to Db2 subsystems and, optionally, to SQL Tuning Services servers. You can share the connection definition among multiple users to make it easier for your entire development team to connect to the same Db2 subsystems and SQL Tuning Services servers.

11.2.2 Making it easier to deliver high-quality SQL

Regardless of your skill level, if you write applications that include Db2 for z/OS SQL, you can benefit from the extensive collection of features for delivering well-formed SQL. Db2 Developer Extension integrates directly with and extends Visual Studio Code. Developers who are already using Visual Studio Code for other applications can jump right in and use the extension to write, test, tune, and debug SQL and native stored procedures directly within the editor they already know.

Writing SQL

Db2 Developer Extension offers a host of features that simplify the writing of SQL that is easy to parse and syntactically correct:

- ▶ *Syntax checking and highlighting* ensure that the SQL that you write is syntactically correct. Syntax errors are highlighted and, whenever possible, you are provided with potential fixes for the errors.
- ▶ *SQL formatting* makes it easy to parse large blocks of code and to understand the relationship between different blocks of SQL elements and clauses. SQL formatting is supported in all SQL file types, including .sql, .ddl, .spsql, and many others.
- ▶ *Code completion* enables you to type a few characters of the SQL element that you are looking for before you are presented with a selectable list of options, while signature help dynamically presents the parameters that are available as you code your SQL elements. As Figure 11-1 shows, these features are available for Db2-supplied stored procedures and built-in functions.

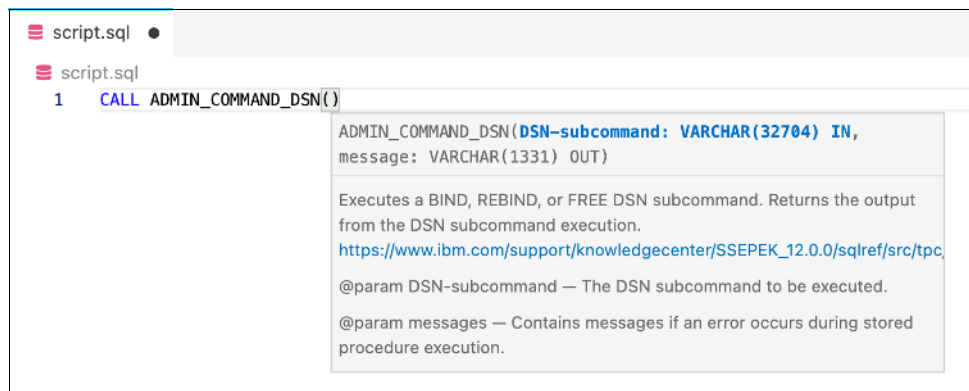


Figure 11-1 Signature help

- ▶ *Customizable code snippets* enable you to define commonly used blocks of code and easily insert them into your file.
- ▶ *Integrated SQL support* links you directly to Db2 for z/OS reference information for the SQL element that you are coding so that you do not have to search for syntax details.

Running SQL

Db2 Developer Extension provides a tremendous amount of flexibility for running your SQL. These features, which are summarized in the following list, enable application developers to deliver their SQL applications more quickly and efficiently:

- ▶ Select and run individual SQL statements from any type of file, both SQL and non-SQL, as shown in Figure 11-2.

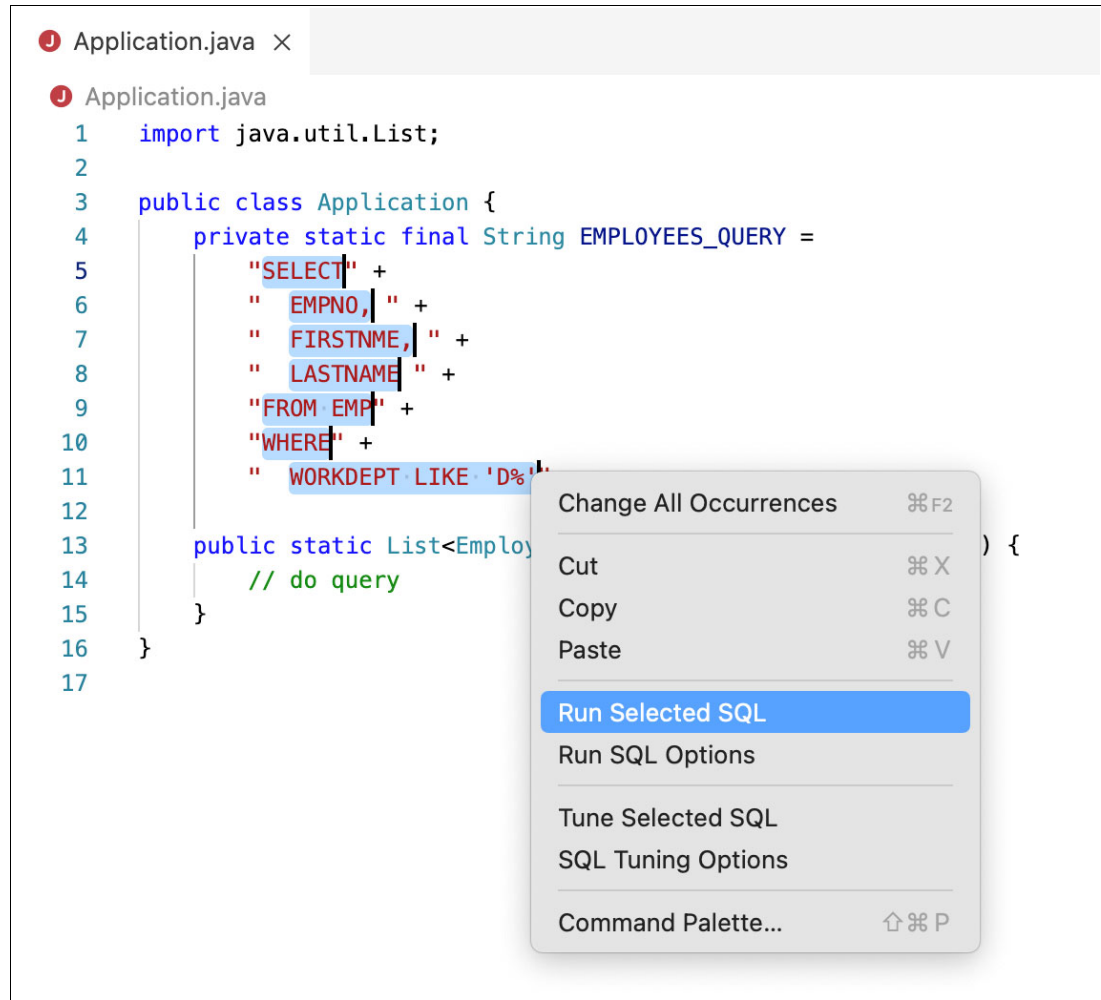


Figure 11-2 Running SQL

- ▶ Select multiple SQL elements on different lines and run those elements as a complete statement.
- ▶ Display consolidated results from running multiple SQL statements in a single view.
- ▶ Run SQL with or without parameter markers and host variables.
- ▶ Run SQL that includes parameters and variables from within a native stored procedure (.spsql file).
- ▶ Customize commit and rollback behavior that is based on whether an SQL statement fails or is successful.
- ▶ Store the results of every SQL execution for historical purposes.
- ▶ Sort the query history by the timestamp of the execution so that you can quickly identify and display failing SQL statements.

- ▶ Export the SQL execution result sets to a .csv file.
- ▶ Limit the number of rows in SQL result sets, which is helpful when you run a statement that returns many rows.

Creating, deploying, and debugging stored procedures

Stored procedures are a powerful tool for increasing the performance and efficiency of distributed applications. Db2 Developer Extension supports the development and debugging of Db2 for z/OS native stored procedures, which are stored procedures that are written entirely in SQL and are created by using the **CREATE PROCEDURE** or **CREATE OR REPLACE PROCEDURE** statements.

Db2 Developer Extension includes the following capabilities for working with native stored procedures:

- ▶ Easily create native stored procedures by using many of the ease-of-use features that are described earlier in this section.
- ▶ Define reusable deployment options that are separate from the code itself, which makes it easier to iteratively run your code and simplifies the ability to push your code into a source code manager.
- ▶ Set conditional hit-count breakpoints, which are helpful for debugging stored procedures that contain loops, as shown in Figure 11-3.

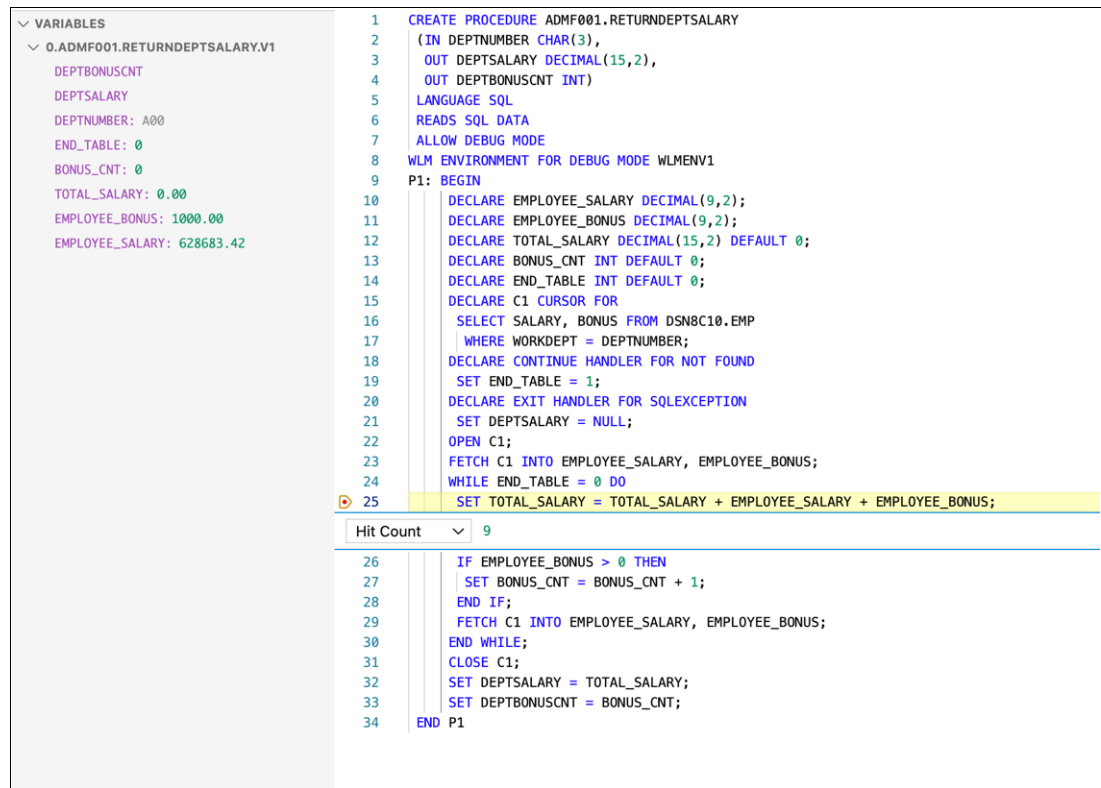


Figure 11-3 Debugging a native stored procedure

11.2.3 Improving query performance

Integrated within Db2 Developer Extension is the capability to run SQL Tuning Services. These tuning features are available with the addition of the Database Services Expansion Pack, which is a component of Db2 Accessories Suite.

The following tuning actions are available from Db2 Developer Extension when you add SQL Tuning Services:

- ▶ Run Visual Explain to generate a visual representation of access paths, as shown in Figure 11-4.

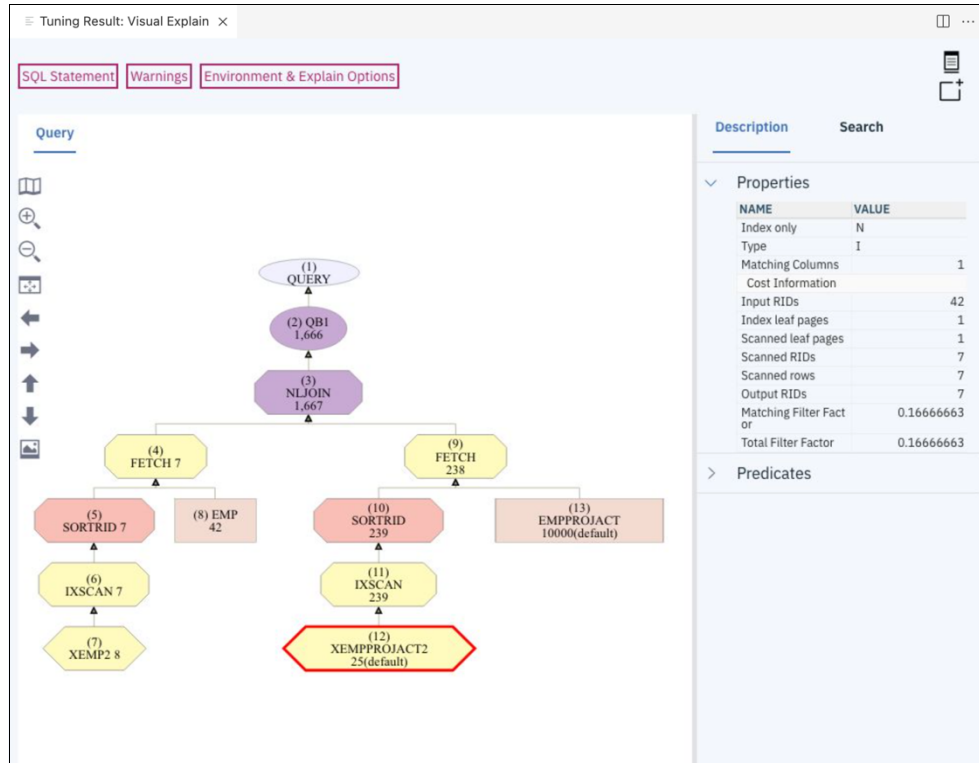


Figure 11-4 Visual Explain

- ▶ Generate advice for collecting statistics for the data objects that are involved in an SQL statement and update statistics profiles by using the Statistics Advisor.
- ▶ Generate and download all the artifacts that are needed to re-create access path issues, such as DDL and statistics, by using the Capture Query Environment feature.
- ▶ Create, migrate, and drop **EXPLAIN** tables.
- ▶ Configure tuning options.
- ▶ Save a history of your tuning activities.

11.2.4 Putting Db2 Developer Extension to use

Db2 Developer Extension is available for you to download today at no cost from the [Visual Studio Marketplace](#).

All the features and capabilities that are described in the previous sections and more are documented in detail in [IBM Developer Extension Documentation Repository on GitHub](#), along with tips and tricks for getting the most out of Db2 Developer Extension.

11.3 IBM Db2 Administration Foundation for z/OS

IBM Db2 Administration Foundation for z/OS (Admin Foundation) is a no-charge, separately installable solution for administering Db2 for z/OS. It provides foundational administrative capabilities in the form of a robust and intuitive web-based user interface that supports the following actions:

- ▶ Navigating the Db2 catalog
- ▶ Running, analyzing, and tuning SQL statements
- ▶ Running Db2 commands
- ▶ Generating DDL from the catalog for Db2 objects (including stored procedures), and editing and running the DDL
- ▶ Generating bind command syntax and performing bind, rebind, and free commands

Admin Foundation is available as a component of IBM Unified Management Server for z/OS. The framework of Admin Foundation is provided by the open-source Zowe Virtual Desktop, which is a modern and intuitive alternative to traditional IBM Z interfaces. It provides z/OS administrators and developers with a simple, open, and familiar tool platform.

11.3.1 Running SQL and Db2 commands

With Admin Foundation, you can run SQL statements and Db2 commands and save both the statements and commands that you run and their results. You can also edit and rerun the commands and statements directly from the history.

You can run multiple SQL statements simultaneously from the SQL editor, and the results of each run are displayed for easy review, as shown in Figure 11-5 on page 163.

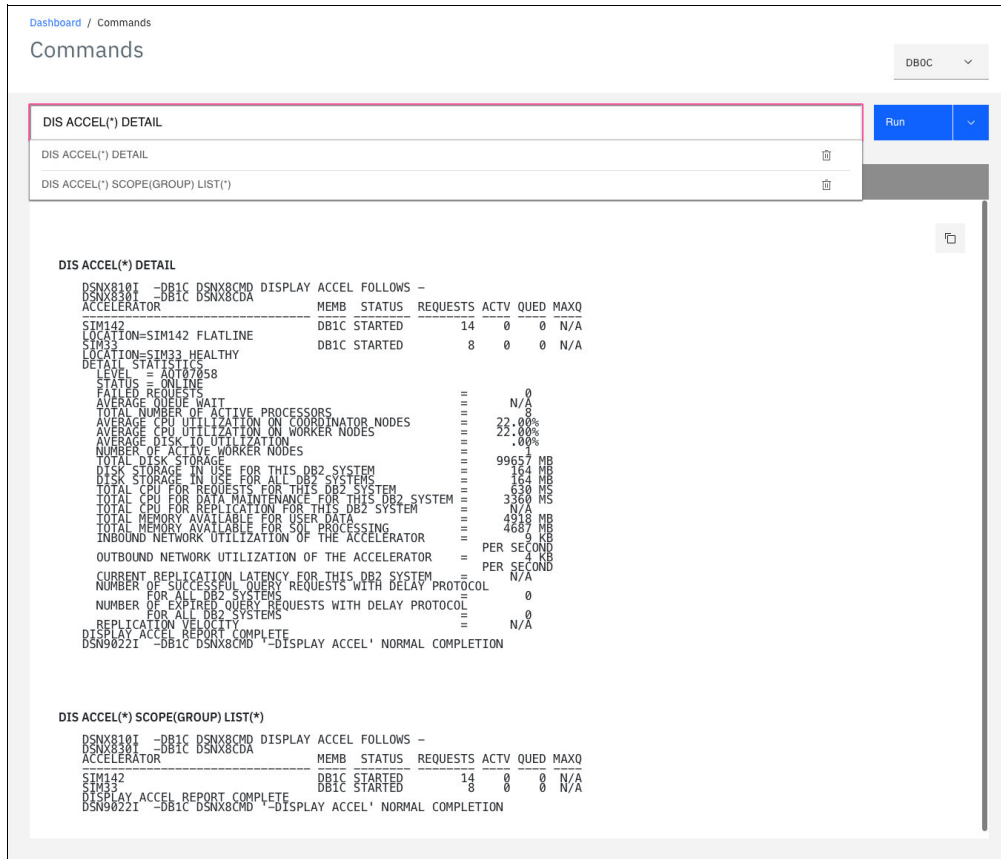


Figure 11-5 Running Db2 commands

11.3.2 Working with catalog objects

Use the Admin Foundation Explore objects dashboard to work with all types of database objects across all the subsystems in your sysplex:

- ▶ The robust and flexible search capabilities of this dashboard make it easy to define and refine search criteria to find the objects for which you are looking.
- ▶ Results are displayed in object detail dashboards, which provide information about the general characteristics of an object and details that vary based on the object type. For example, while browsing an object, you can use the interface to move up and down the object hierarchy and easily find and examine the details of related objects.
- ▶ From within search results, you can further refine your search criteria to focus on the information for which you are looking.
- ▶ For ease of reuse, you can save search criteria and assign them an intuitive name.
- ▶ You can view and generate DDL for Db2 objects.

11.3.3 Exploring SQL

Use the Admin Foundation Explore SQL dashboard to work with SQL statements from different sources:

- ▶ Capture SQL data from many sources, including the statement cache, plans and packages, an input file, or stabilized SQL statements.
- ▶ Filter search results based on performance criteria that you define. For example, statements that use more than a specified amount of CPU time, elapsed time, or a **GETPAGE** threshold.
- ▶ Tune underperforming SQL statements, which is described in detail in 11.3.4, “Analyzing and tuning SQL statements” on page 164.
- ▶ Determine whether an SQL statement is stabilized, and then either stabilize or free that statement depending on its current state.
- ▶ View the history of SQL statements and edit and rerun them.

11.3.4 Analyzing and tuning SQL statements

Admin Foundation integrates the SQL Tuning Services REST APIs to help you quickly identify, evaluate, and tune underperforming Db2 for z/OS SQL statements. Through the Admin Foundation GUI, you can perform the following tasks:

- ▶ Run Visual Explain, Statistics Advisor, and SQL Query Capture jobs.
- ▶ Capture and download a query’s environment for evaluation and troubleshooting purposes.
- ▶ Stabilize the access paths for cached dynamic SQL statements to gain a consistent and predictable level of performance, and then free stabilized SQL statements when necessary.
- ▶ Choose the source that contains the SQL that you want to work with including the statement cache, an input file, a package, a plan, or stabilized SQL.
- ▶ Specify the search criteria to help you identify the SQL statements that you want to analyze and sort them by those criteria.
- ▶ Create, manage, view, share, and delete tuning profiles, which serve as the connection between Admin Foundation and SQL Tuning Services. The profiles are required to use tuning capabilities from within Admin Foundation.

Tuning profiles

Tuning profiles reference a particular Db2 for z/OS subsystem that is registered in Admin Foundation and on which tuning actions are run. With Admin Foundation, you can create, manage, view, share, and delete tuning profiles. By using these tuning profiles, you can submit tuning jobs and then view the results of the submitted job and recommendations for actions that you can take on the submitted job.

Figure 11-6 on page 165 shows the Create tuning profile page.

Create tuning profile

Specify a tuning profile name and select a registered subsystem from the list.

Name ⓘ 0/254

Registered subsystems

DBOC

Location	DBOC
Hostname	9.30.132.107
Port	8030

Cancel Create

Figure 11-6 Creating a tuning profile

Supported sources

You can capture results from SQL statements from the following sources:

- ▶ Statement cache
- ▶ Input file
- ▶ Packages
- ▶ Plans
- ▶ Stabilized SQL

By using the capture results from the various sources, you can determine the SQL statements that are running efficiently; the statements that must be tuned or modified; the statements that are using too many resources; and the statements that can be improved.

Filtering and ordering

You can filter the SQL statements that you want to see, such as only the statements that consume too much CPU resource or the statements that take a long time to run, by selecting the appropriate column in your filtering criteria, as shown in Figure 11-7. Ordering enables you to display results in ascending or descending order. With these features, you can quickly identify the statements that are in most need of attention so that you can use your time efficiently.



Figure 11-7 Filtering and ordering options for columns of SQL statements

For example, some filtering criteria that you can specify for an SQL statement displays the load on the CPU from that statement; the length of time that the statement took to run; and the amount of resources that the statement is using, as shown in Figure 11-8. These statistics help you identify statements that must be tuned or modified and identify weaknesses in a system.

<input type="checkbox"/>	STMT_TEXT	STMT_ID	STAT_EXEC	STAT_GPAG	STAT_ELAP	STAT_CPU	
<input type="checkbox"/>	SELECT P.NAME, P.COLLOID, P.TIMESTAMP, I.NAME, I.CREAT...	47	47	670712	1.231028	1.220296	:
<input type="checkbox"/>	SELECT PL.COLLOID, PL.NAME, PA.COLLOID, PA.NAME, PA.TI...	33	89	333305	0.695676	0.689852	:
<input type="checkbox"/>	SELECT DSNAQT.ACCEL_GETVERSION() FROM SYSIBM.SYS...	232	41	0	0.461107	0.001279	:
<input type="checkbox"/>	SELECT P.NAME, P.CREATOR, P.BOUNDS, PA.COLLOID FRO...	31	89	29076	0.455493	0.425296	:
<input type="checkbox"/>	SELECT P.NAME, P.COLLOID, P.TIMESTAMP, I.NAME, I.CREAT...	86	42	256607	0.447525	0.443766	:
<input type="checkbox"/>	/*_IBM_TMS */ SELECT JOBID, JOBNAME, JOBTYP, JOBC...	72	20	14425	0.286155	0.20506	:

Figure 11-8 Statistics that indicate how an SQL statement is performing

11.3.5 Use cases

The following sections show how to use some of the significant capabilities of Admin Foundation.

Working with tables and their DDL

The Explore objects dashboard is the primary tool for finding and working with Db2 objects. You can use this dashboard to find a particular table; explore the characteristics of the table; and work with the table's DDL.

To use the Explore objects dashboard, complete the following steps:

1. Open the Explore objects dashboard, select **Table** from the **Object type** menu, and click **Apply** to display a sortable list of tables, as shown in Figure 11-9 on page 167.

Note: You can also specify the first characters of the object name, its schema, and a specific subsystem to search, but for this use case, all tables are being displayed.

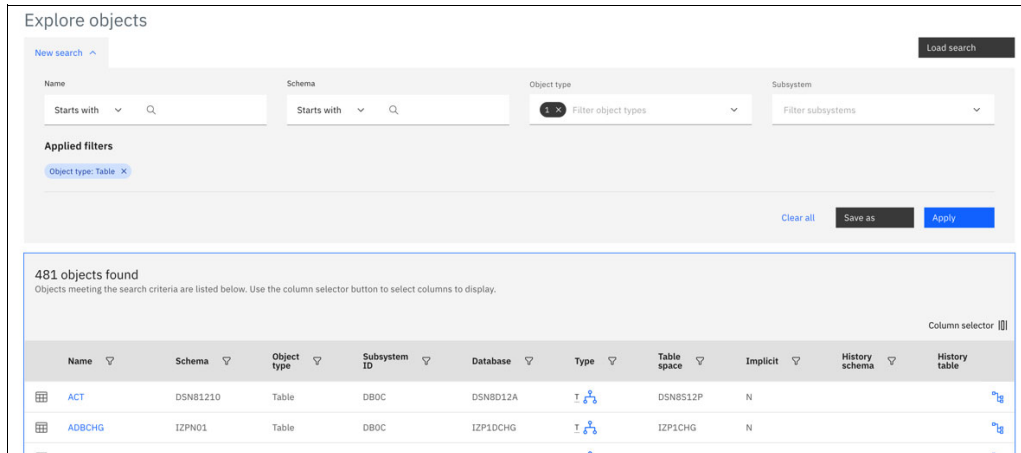


Figure 11-9 Exploring tables

2. Select the table that you want to explore to display details about that table. The details are organized by the following tabs:

- The **Overview** tab provides a high-level view of several key characteristics of the table, as shown in Figure 11-10. You can display more detailed information about the object hierarchy, referential integrity (RI), and the structure of the table by clicking the **Details** link in the respective views. You can also display these details by clicking the tabs at the top of the dashboard.

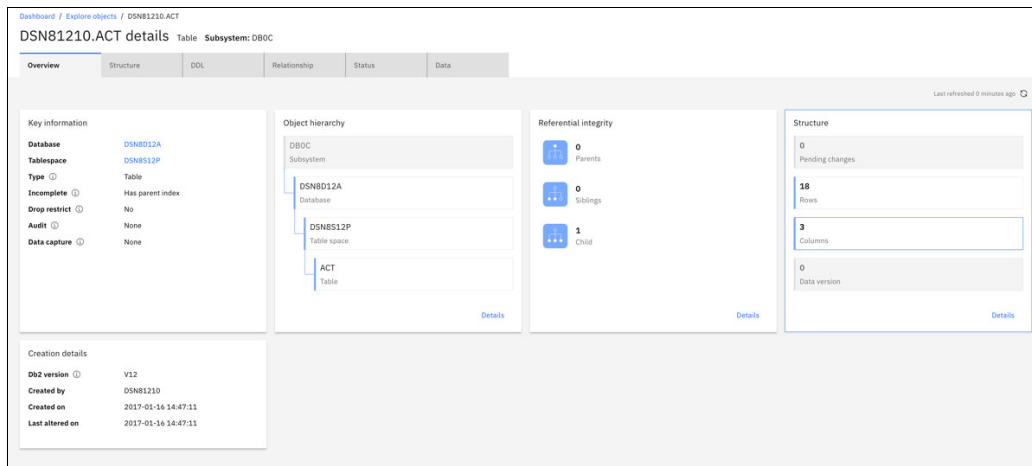


Figure 11-10 Object Explorer Overview tab for tables

- The **Structure** tab shows you all the columns in the table, their data types, length, and other characteristics of the columns. It also shows information about indexes that are defined for the table.
- The **DDL** tab displays the SQL that was used to create the table. From here, you can run the SQL directly, or you can copy it and edit it in the SQL editor or use it as a template to create a similar table elsewhere in your system.

- The **Relationship** tab shows a visual representation of the table’s hierarchy, from the subsystem down to the indexes, aliases, and views for the table, as shown in Figure 11-11.

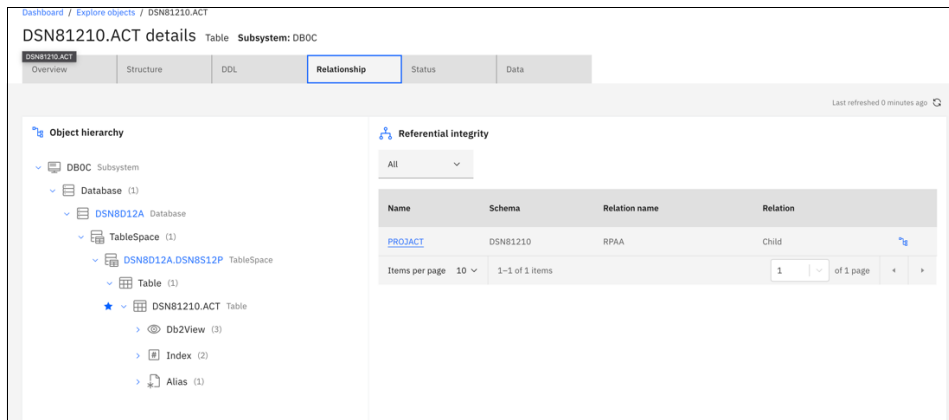


Figure 11-11 Visual representation of a table hierarchy in the Relationship tab

- The **Status** tab displays locks and claims on the table.
 - The **Data** tab shows the actual data that exists within the table.
3. Click the **DDL** tab to generate the DDL for the table, as shown in Figure 11-12.



Figure 11-12 DDL for a table

From here, you can run the DDL directly, or you can copy it and edit it in the SQL editor or use it as a template to create a similar table elsewhere in your system.

Working with native stored procedures and their DDL

You can also use the Explore objects dashboard to work with native stored procedures. To do so, complete the following steps:

1. Open the Explore objects dashboard, enter the first several characters of a stored procedure name in the Name field, select **Stored procedure** from the **Object type** menu, and click **Apply** to display a list of stored procedures whose names begin with the string that you entered, as shown Figure 11-13 on page 169.

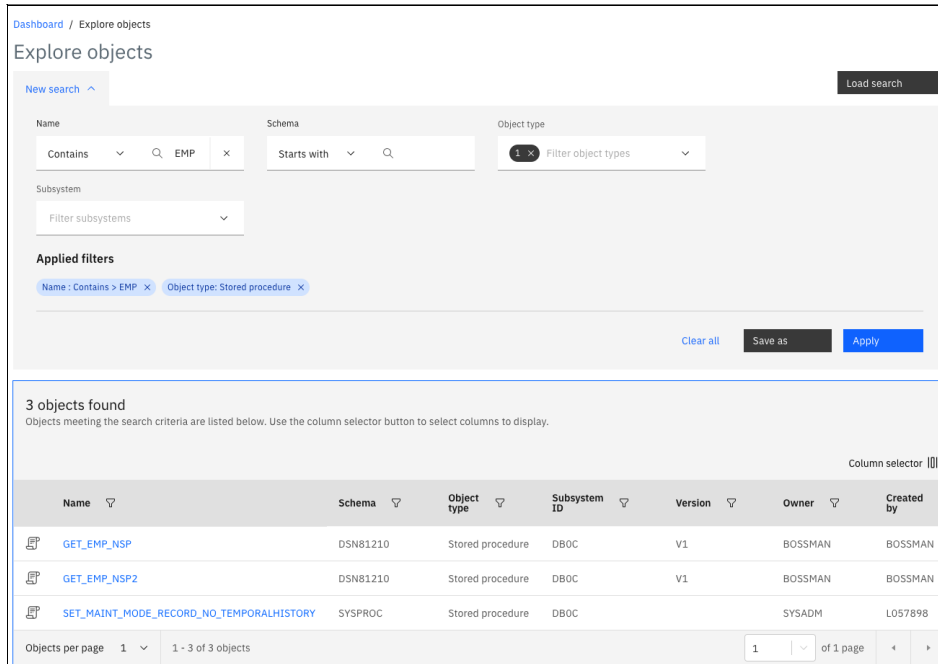


Figure 11-13 Exploring native stored procedures

2. Select the stored procedure that you want to work with to display a summary of its characteristics and structure.
3. Click the **DDL** tab to generate the DDL for the stored procedure.

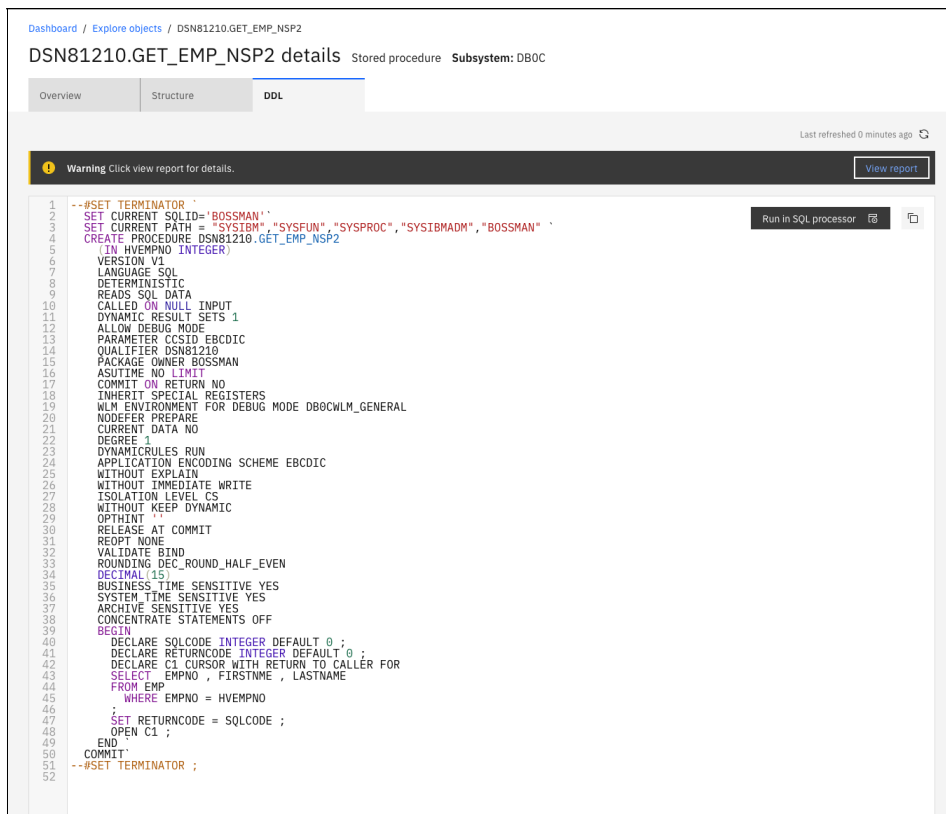


Figure 11-14 Generating DDL for a native stored procedure

From here, you can run the DDL directly, or you can copy and edit it in the SQL editor or use it as a starting point to create a similar stored procedure elsewhere in your system.

Exploring and tuning SQL statements

The most basic function that is provided by Admin Foundation is its ability to let you find and view objects and SQL statements. The following use case focuses on exploring SQL statements, but you can apply the same general process to Db2 objects. This use case also shows how to use Admin Foundation's tuning capabilities.

Complete the following steps:

1. Open the Explore SQL dashboard and select the source from which you want to capture SQL data. Admin Foundation supports capturing from the statement cache, plans, packages, and stabilized SQL.
2. Define filters and set ordering criteria.
3. As shown in Figure 11-7 on page 166, you filter by the STAT_ELAP column, which displays only those statements that meet the criteria that you define (for example, > 0.1). Filtering by STAT_ELAP makes it easy to find and focus on improving the longest-running queries. Applying the descending ordering criterion results in the longest-running query being displayed at the top of the results.
4. Click **Capture**. Statements are captured from the source that you specified, and the results are displayed based on the filtering criteria that you specified.
5. Sort the results by the STAT_ELAP column to display the longest-running queries at the top of the list (see Figure 11-8 on page 166).

By filtering and sorting results, you can take a methodical approach to improving SQL performance starting with the statements that need the most attention.

6. Select the longest-running query and click **Take Action** → **View SQL Query** to display the entire query. From here, you can edit and run the query, and you can tune it.
7. Click **Tune**. The Query tuning dialog box opens and shows details about that SQL statement in the SQL dashboard.
8. Specify the tuning profile that you want to use, the tuning actions that you want to run (Statistics advisor, Visual explain, or both), and relevant parameters. In this example, we select both **Statistics advisor** and **Visual explain**, as shown in Figure 11-15.

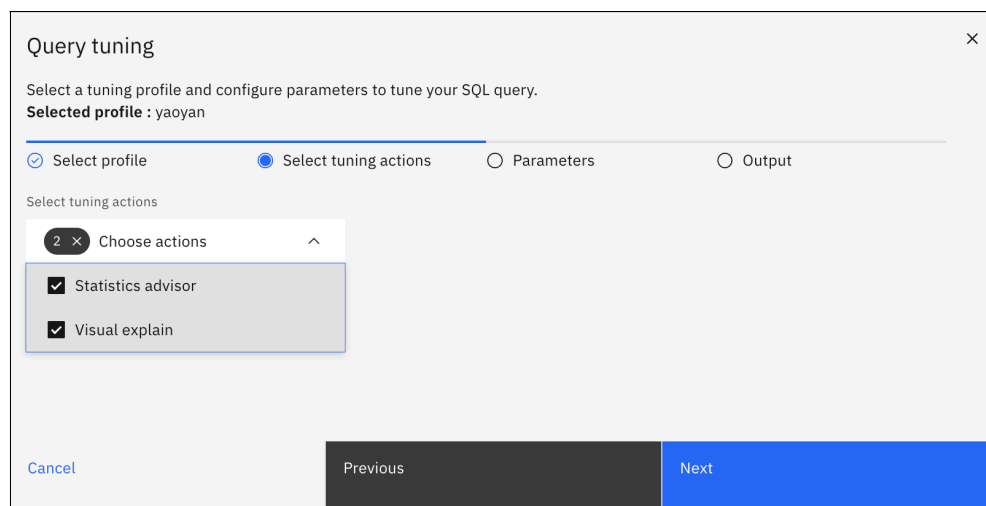


Figure 11-15 Selecting tuning actions

9. Click **Next** to run the tuning actions. The **Output** tab opens and shows job IDs, the job type, and the status (**Job running** or **Ready to view**).
10. Click **View results** for a job to display details about the SQL, as shown in Figure 11-16.

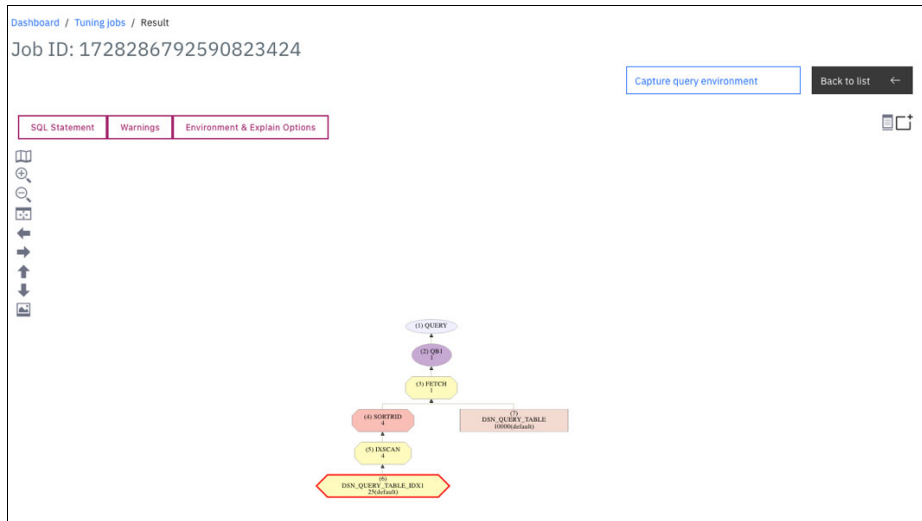


Figure 11-16 Results from a Visual Explain job

A typical use case involves running both Visual explain and Statistics advisor simultaneously, and then switching back and forth between the output of these two jobs to evaluate and understand any performance issues, as shown in Figure 11-17.

The screenshot shows a web interface for a tuning job. At the top, it displays 'Home / View Jobs / Tuning / Result' and 'Job ID: 1726773888847712256'. There are buttons for 'Capture query environment' and 'Back to list'. Below the job ID, there is a 'Recommendations' section with the text: 'Collect missing statistics. Re-collect statistics that are conflicting or that are potentially obsolete. Collect statistics to address data skew and data collection problems.' Below this is a 'RUNSTATS command' section with a large empty text area. At the bottom, there is a 'Details and reports' section with a link for 'RUNSTATS result'.

Figure 11-17 Results from a tuning job

Working with plans and packages

Another useful feature of Admin Foundation is its ability to capture all the SQL statements that are bound to a particular package or plan. To use this feature, complete the following steps:

1. Open the Explore objects dashboard, select **Package (Db2)**, **Plan (Db2)**, or both from the **Object type** menu, and click **Apply**. A sortable list of plans and packages is displayed.
2. Click the plan or package that you are interested in to display details about it, as shown in Figure 11-18.

The screenshot shows the 'ADBL.ADBO details' page in Admin Foundation. The page is divided into several sections:

- Key information:** Name: ADBO, Last used on: 0001-01-01, Collection: ADBL, Owner type: SYSADM, Version: V12.1.0.U155505, Valid: Rebind not required but objects..., Operative: Yes, Isolation: Cursor Stability, Explain: No, SQL error: Continue, Package source: Locally bound from a DBRM, SQL path: Default, Remote access method: DRDA, Function resolved TS: 2019-04-16 14:42:29.257027, Encoding CCSID: 37, Immediate write: Inheriting value from the plan, Concurrent access solution: Not specified, Extended indicator: Not specified, Release: Commit, Rounding: ROUND_HALF_EVEN, Distribute: N, Sensitive to business time: Sensitive, Sensitive to system time: Sensitive, Package type: Created By BIND Package com..., Copy ID: 0 (Current version).
- Structure:** 8 Number of table spaces, 8 Number of tables, 13 Number of indices, 0 Number of views.
- Bind information:** Last bind time: 2019-04-16 14:42:29.261662, Qualifier: SYSIBM, Degree: 1, Group member: DB1C, Dynamic rules: Not specified, Recoptimize: None, Defer prep: Data currency not required for a..., Defer prepare: Trigger package/Not specified, Keep dynamic at commit: No, Optimizer hint: No hint, Plan management: Off, Plan management scope: Scope static, Access path reuse: No, Access path retain duplicate: All copies retained, Application compatibility: V12R1M500, Sensitive to archive: Sensitive, Origin: BIND command, Apressure no function level: V12R1M500, Apressure no timestamp: 2019-04-16 14:42:29.261662, Concatenation statement enabled: No, Function level of row: V12R1M500, Consistency token: 1A84FF6417528163.
- Pre compile information:** Quote: Apostrophe, Comma: Period, Host language: PL/I, Character set: No, Mixed data: No, Dec31: No, Pre compiled timestamp: 2018-04-09 12:28:14.976581, PDS name: DB2TOOLS.ADB121.SADBDRM, Description stat: Yes.
- Enabled systems:** Enabled for all systems.
- Creation details:** Created by: VNDONNY, Created on: 2018-09-13 10:13:03, Release bound: V12.

Figure 11-18 Exploring a package

3. Click the **Bind** tab to display the **BIND** command for the package or plan that you selected in an editor, as shown in Figure 11-19 on page 173. From here, you can bind, rebind, and free packages and plans. You can also use the Diff view to review any changes that you make before you apply them.

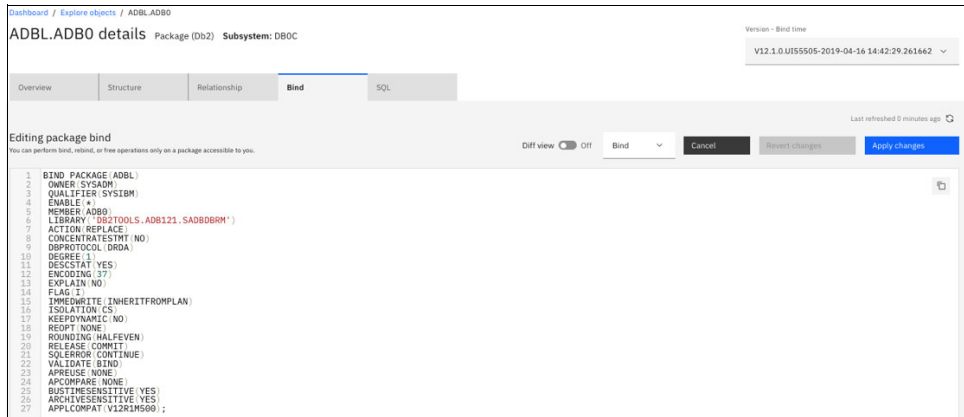


Figure 11-19 Bind package tab

4. Click the **SQL** tab to display all the SQL statements that are bound to that package, and then click the three vertical dots at the right side of each SQL statement to initiate tuning or to view the query.

This process can also be initiated by selecting plans or packages from the **Select source** menu in the Explore SQL dashboard.

Stabilizing and freeing SQL

When you stabilize a dynamic SQL statement, Db2 stores the SQLID, statement text, and runtime structures for the statement in certain Db2 catalog tables. When a cache miss occurs for a stabilized dynamic SQL statement, Db2 loads the cache structures from the catalog instead of processing a full prepare for the statement.

Admin Foundation makes it easy to stabilize SQL statements and to free statements that are already stabilized.

To stabilize or free an SQL statement, complete the following steps:

1. Open the Explore SQL dashboard, define any filtering and ordering criteria that you want, and select **Statement cache**, **Plan**, or **Package** from the **Select source** menu.
2. Click **Capture**. Statements are captured from the source that you specified, and the results are displayed based on the filtering criteria that you specified.
3. The Stabilized column indicates the stabilization state for each statement in the results.
4. Select a statement and click **Take Action**. Depending on the state of the statement, you can either stabilize it or free it, as shown in Figure 11-20.

1 item selected					Take Actions	Cancel
<input type="checkbox"/>	STMT_ID	STAT_CPU	Full statement text	STABILIZED		
<input checked="" type="checkbox"/>	1485260	0.000483	SELECT P.NAME, P.BOUNDS, P.CREATOR, R.NAME, R.SCHEMA, R.OWNER, R...	N	View SQL query	
<input type="checkbox"/>	1485714	0.000484	SELECT TS.PAGENUM FROM SYSIBM.SYSTABLESPACE TS WHERE TS.DBNA...	N	Tune	
<input type="checkbox"/>	1481013	0.000488	SELECT COUNT(*) FROM (SELECT DISTINCT CREATOR, TNAME FROM IBMT...	N	Capture Query Environment	
					Stabilize	
					Free stabilized	

Figure 11-20 Stabilizing an SQL statement

To display all currently stabilized SQL statements, complete the following steps:


1. Open the Explore SQL dashboard, define any filtering and ordering criteria that you want, and select **Stabilized SQL** from the **Select source** menu.
2. Click **Capture** to display all the stabilized SQL statements that meet the criteria that you defined.
3. Click the three vertical dots at the right side of an SQL statement and select **Free stabilized** to free it.

Administering Db2 Analytics Accelerator

With Admin Foundation, you can work with accelerators that are paired to Db2 for z/OS subsystems. For more information, see Chapter 13, “IBM Db2 Analytics Accelerator for z/OS” on page 201.

11.4 Summary

SQL Tuning Services, Db2 Developer Extension, and Admin Foundation represent the next generation of tools for administering Db2 for z/OS and developing Db2 for z/OS applications. They are designed and developed specifically to increase the efficiency and productivity of Db2 for z/OS users of all skill levels by using robust APIs, intuitive user interfaces, and familiar related tools. They are also designed to evolve alongside Db2 for z/OS to support the latest Db2 13 enhancements and beyond.



IBM Db2 for z/OS tools support for Db2 13

You can use the suite of IBM Db2 for z/OS tools to simplify the way that you interact with Db2 for z/OS. These tools help guide you throughout the process of developing, testing, monitoring, and maintaining your Db2 objects and applications. You can use IBM tools to modernize your applications and gain insights that can help you optimize performance and availability. In general, these tools help increase your productivity and reduce costs while maintaining unparalleled levels of privacy, security, and data integrity.

When you migrate to Db2 13, you can continue to use your favorite Db2 tools. You can use any of these tools with Db2 13 starting on day one of Db2 13 general availability (GA). More importantly, you can use these tools to leverage new features of Db2 13.

The following IBM tools have a new version or release to support Db2 13:

- ▶ Db2 Administration Tool for z/OS 13.1
- ▶ Db2 Query Management Facility 13.1
- ▶ IBM OMEGAMON® for Db2 Performance Expert on z/OS 5.5

For the following IBM tools, you can enable support for Db2 13 by applying a PTF to the latest version:

- ▶ Db2 Analytics Accelerator Loader for z/OS 2.1
- ▶ Db2 Automation Tool for z/OS 4.3
- ▶ Db2 Cloning Tool for z/OS 3.2
- ▶ Db2 Log Analysis Tool for z/OS 3.5
- ▶ Db2 Recovery Expert for z/OS 3.2
- ▶ Db2 SQL Performance Analyzer for z/OS 5.1
- ▶ Db2 Query Monitor for z/OS 3.3.
- ▶ Data Virtualization Manager for z/OS 1.1

For more information about which PTF that you need to apply for each tool, see [IBM Documentation](#).

This chapter describes the following topics:

- ▶ IBM Db2 Administration Tool for z/OS
- ▶ IBM Db2 Automation Tool for z/OS
- ▶ IBM Db2 Log Analysis Tool for z/OS
- ▶ IBM Db2 Recovery Expert for z/OS
- ▶ IBM Db2 Query Monitor for z/OS
- ▶ IBM Db2 Query Management Facility
- ▶ IBM Db2 Sort for z/OS
- ▶ IBM Data Virtualization Manager for z/OS
- ▶ IBM OMEGAMON for IBM Db2 Performance Expert on z/OS

12.1 IBM Db2 Administration Tool for z/OS

IBM Db2 Administration Tool for z/OS (Db2 Admin Tool) provides an ISPF panel interface that enables you to easily manage Db2 administrative tasks, such as navigating the Db2 catalog, creating and altering objects, and managing and tracking changes to objects.

Db2 Admin Tool released a new version (13.1) that includes comprehensive support for new Db2 13 features and usability improvements. You can run Db2 Admin Tool 13.1 on both Db2 13 subsystems and Db2 12 subsystems.

Db2 13 features are supported by the following enhancements in Db2 Admin Tool 13.1:

- ▶ The ability to easily convert partition-by-growth (PBG) table spaces to partition-by-range (PBR) table spaces by using the new Db2 13 **ALTER** syntax, which minimizes outages and maximizes concurrency.
- ▶ The ability to view Db2 utility history information for better insight into utility usage and optimization, including the ability to search and delete SYSUTILITIES rows and to navigate from SYSCOPY rows and **DISPLAY UTIL** output to related SYSUTILITIES rows.
- ▶ Support for deleting an active log data set simply by selecting the data set from a list.
- ▶ Support for specifying a package owner type (role or user), which increases the flexibility for package ownership.
- ▶ Page sampling support for inline statistics for the **REORG** and **LOAD** utilities, which has the potential to reduce both CPU time and elapsed time.
- ▶ A simple panel interface for managing Db2 profiles, including support for the profile enhancements in Db2 13, which make profiles increasingly important in helping automate certain aspects of monitoring and controlling a Db2 subsystem.
- ▶ Support for longer column names (up to 128 characters).
- ▶ Support for all changes to the real-time statistics (RTS) tables and Db2 catalog tables in function levels 500 and 501.
- ▶ Removal of the **NOSYSUT1** option from the **REORG INDEX** utility options panel and generated jobs to correspond to the **REORG INDEX** changes in Db2 13.

Note: To use any new Db2 13 features with Db2 Admin Tool 13.1, you must bind Db2 Admin Tool (the ADBMAIN program) with the appropriate Db2 13 application compatibility (APPLCOMPAT) value: V13V1M500 or later. If your Db2 subsystem is at V13R1M501, but ADBMAIN is bound with APPLCOMPAT V13R1M100, Db2 Admin Tool does not allow you to use Db2 13 new function.

Db2 Admin Tool 13.1 also includes the following usability enhancements:

- ▶ The ability to generate commands on a list of objects, which can save extra steps and time.
- ▶ The visibility of the catalog level and remote subsystem location (if applicable) on the main menu.
- ▶ The ability to navigate directly from a list of table spaces or indexes to the RTS tables for a selected object.
- ▶ The option to suppress the copyright statement when invoking Db2 Admin Tool, which allows external tools to seamlessly invoke the catalog navigation feature.

Note: For examples of some of these key enhancements, see the following blog posts in the IBM Community:

- ▶ [Db2 Administration Tool 13.1: Converting partition-by-growth \(PBG\) table spaces to partition-by-range \(PBR\) table spaces online](#)
- ▶ [Db2 Administration Tool 13.1: Generating commands for a list of objects](#)
- ▶ [Db2 Administration Tool 13.1: Managing utility history](#)
- ▶ [Db2 Administration Tool 13.1: Managing Db2 profile tables](#)

For a list of all the enhancements in Db2 Admin Tool 13.1, see [What's new in Db2 Admin Tool 13.1](#).

12.2 IBM Db2 Automation Tool for z/OS

IBM Db2 Automation Tool for z/OS automates the execution of Db2 utilities while reducing the effort and resources that are needed to perform routine and exception-based Db2 tasks. In addition, an extended automation function that is provided in a Db2 for z/OS solutions pack provides for comprehensive management and enhanced reporting for Db2 for z/OS utilities and automates the execution of tasks during a defined maintenance window.

Db2 Automation Tool 4.3 supports new features in Db2 13 by providing the following capabilities:

- ▶ You can view utilities history information in an easy-to-read format on an ISPF panel with the ability to delete history table entries.
- ▶ If you convert a table space from PBG to PBR, Db2 Automation Tool can detect the resulting pending status and trigger a **REORG** operation or other appropriate action.
- ▶ All new Db2 commands, parameters, new column types in RTS tables, and utility syntax changes are supported by Db2 Automation Tool.

For more information about these enhancements and other new features in Db2 Automation Tool, see the [IBM Db2 Automation Tool for z/OS 4.3 library](#).

12.3 IBM Db2 Log Analysis Tool for z/OS

IBM Db2 Log Analysis Tool for z/OS helps to ensure high availability (HA) and complete control over data integrity. It allows you to monitor data changes by automatically building reports of changes that are made to database tables.

Db2 Log Analysis Tool 3.5 supports Db2 13 by offering the following new features:

- ▶ Support for 128-character column names in Db2 tables
- ▶ Improved handling of inline image copies to implement processing through online conversion of a PBG table space to a PBR table space
- ▶ The ability to generate an Application-Relationships report based on intelligent analysis of the Db2 logs

12.3.1 Application-level relationship discovery

Many users have scripts and procedures to update their data, and these Db2 objects are often changed in bulk. However, there is no referential integrity that exists between those tables, which makes it difficult to track how Db2 objects are related from an application standpoint.

This new Db2 Log Analysis feature discovers the application-level relationship of Db2 objects and allows you to accomplish the following tasks:

- ▶ Optimize an environment to group objects and perform backup, recovery, and DevOps tasks more efficiently.
- ▶ Apply a relational database model to prevent update, insert, and delete operations' anomalies.
- ▶ Enhance transaction processing to ensure high performance, reliability, and consistency.

When you generate activity reports, a new option enables data collection on objects and units of work for relationship discovery. Application-level relationship discovery is available in the **Utilities** menu.

For more information about these enhancements and other new features in Db2 Log Analysis Tool, see the [IBM Db2 Analysis Tool for z/OS 3.5 library](#).

12.4 IBM Db2 Recovery Expert for z/OS

IBM Db2 Recovery Expert for z/OS is a storage-aware backup and recovery solution that integrates storage processor fast-replication facilities with Db2 backup and recovery operations. This solution allows efficient backups with reduced recovery time and simplifies disaster recovery (DR) procedures while using fewer CPU, I/O, and storage resources.

Db2 Recovery Expert 3.2 supports Db2 13 and introduces many usability enhancements, the most significant of which is Recover Health Check.

Recover Health Check can determine the recoverability of a set of objects and their resources within a certain timeframe. Then, the Intelligent Suggestions for Recovery Health component provides suggestions for improving recovery.

When you use the Recover Health Check feature, Db2 Recovery Expert reports timeframes when selected objects are unrecoverable and the reason why they are unrecoverable.

Recover Health Check surveys the selected objects and their resources, including image copies, logs, and system-level backups to create a holistic recovery view.

By using the report, you can identify any limitations of your current recovery policy. For example:

- ▶ Recover Health Check identifies that the log data set cannot be found, so the object is reported as unrecoverable for a certain timeframe.
- ▶ Recover Health Check identifies when there are two LOG NO events and no image copy or system-level backup in between, so the object is reported as unrecoverable for a certain timeframe.

Then, Intelligent Suggestions for Recovery Health provides suggestions for improving the recovery run time or the tool run time. These suggestions allow you to improve your recovery routine.

In the following scenarios, Db2 Recover Expert offers the following suggestions:

- ▶ If Recover Health Check identifies a deprecated table space within the selected objects, Intelligent Suggestions for Recovery Health suggests converting the table space.
- ▶ If Recover Health Check identifies an outdated SYSCOPY record, Intelligent Suggestions for Recovery Health recommends running **MODIFY RECOVERY**.

For more information about these enhancements and other new features in Db2 Recovery Expert, see the [IBM Db2 Recovery Expert for z/OS 3.2 library](#).

12.5 IBM Db2 Query Monitor for z/OS

IBM Db2 Query Monitor for z/OS offers current and historical views of query activity throughout Db2 subsystems. It efficiently customizes and tunes your SQL workload and Db2 objects, validates the effectiveness of your Db2 subsystems, and improves overall performance. It also offers you extensive choices so you can determine what monitoring information you can gather and when.

Db2 Query Monitor for z/OS 3.3.0 supports Db2 13 and introduces many other enhancements:

- ▶ The ability to perform exception and alert reporting by using adaptive thresholds
- ▶ Improved support for application development by enabling positive SQLCODEs to be specified in monitoring profiles
- ▶ Improved performance and serviceability:
 - Reduced CPU consumption by enabling detailed object collection by workload.
 - Smarter reorganization recommendations, which offer the ability to leverage SQL performance data that is collected about application objects to determine whether a reorganization is necessary.
 - Loop detection in applications running on a thread or Db2 subsystem.
 - Updated security protocols.

- ▶ Improved usability:
 - Quicker navigation to problematic queries.
 - Reorg Recommendation Monitor, which shows long name and offload column names for threshold metrics.
 - New **OPTKEY QUERYNO** added for static SQL.

For more information about these enhancements and other new features in Db2 Query Monitor, see the [IBM Db2 Query Monitor for z/OS 3.3 library](#).

12.6 IBM Db2 Query Management Facility

IBM Db2 Query Management Facility (QMF) is a security-rich, easy-to-use business analytics and visualization solution that is optimized for IBM Z data.

QMF 13.1 is a new release that offers hybrid cloud deployment, support for cloud-native data sources, improved task scheduling capability, and ease of use for configuration management. Key enhancements include the following ones:

- ▶ Simplification of workstation configuration rollout
- ▶ An improved user of awareness for QMF features with context-specific tips
- ▶ Hybrid cloud deployment on IBM Cloud® and Microsoft Azure
- ▶ Support for the cloud-native data sources Athena and Redshift
- ▶ Support for MongoDB
- ▶ Improved scheduling capability with time zone selection
- ▶ QMF for WebSphere deployment on IBM z/OS Container Extensions (IBM zCX)
- ▶ Enhanced Global Variable support for the QMF IBM Z client
- ▶ Enhanced QMF object organization with QMF for Time Sharing Option (TSO)

For more information about these enhancements and other new features in QMF 13.1, see the [IBM Db2 Query Management 13.1 library](#).

12.7 IBM Db2 Sort for z/OS

IBM Db2 Sort for z/OS (Db2 Sort) provides high-speed utility sort processing for data that is stored in Db2 for z/OS databases. Db2 Sort improves the sort performance of many of the Db2 utilities in the IBM Db2 Utilities Suite and of several other Db2 management tools.

With the application of APAR PH32374, Db2 Sort leverages IBM z15 Integrated Accelerator for Z Sort while running Db2 Utilities. Db2 **LOAD**, **RUNSTATS**, **REBUILD INDEX**, and **REORG TABLESPACE** can use the on-chip sort acceleration facilities with Db2 Sort, which can help reduce CPU usage and elapsed time for sorting various workloads.

This performance enhancement can greatly improve the throughput of your IBM Db2 Utilities without requiring any extra memory, and it can be used by all the Db2 Utilities that invoke the sort operation, not just **REORG TABLESPACE**. As a result, Db2 Sort has a larger range of IBM Z Sort candidates for Db2 Utilities than DFSORT. In addition to the performance advantages, Db2 Sort uses resources more intelligently.

For more information about these enhancements and other new features in Db2 Sort for z/OS, see the [IBM Db2 Sort for z/OS 2.1.0 library](#).

12.8 IBM Data Virtualization Manager for z/OS

IBM Data Virtualization Manager for z/OS (DVM) provides virtual, integrated views of data that is on IBM Z. It also provides users and applications with read/write access to IBM Z data in place without moving, replicating, or transforming data. It performs these tasks with minimal extra processing costs. By unlocking IBM Z data by using industry-standard APIs, DVM can save you time and money.

Developers can readily combine IBM Z data with other enterprise data sources to gain real-time insight, accelerate deployment of traditional mainframe and new web and mobile applications, modernize the enterprise, and take advantage of the API economy. DVM also supports data access and movement.

Its ability to provision virtual tables and virtual views on and over Db2 and non Db2 data sources on and off the IBM Z platform make DVM an important piece of the Db2 for z/OS data ecosystem.

12.8.1 Key features

DVM can modernize the way that you develop applications by simplifying access to an extensive number of traditional non-relational IBM Z data sources. It incorporates various interface approaches for accessing data that uses IBM Z Integrated Information Processor (zIIP) processors to redirect up to 99% of workloads. The amount of zIIP offload activity varies with different types of workloads depending on the data source, access method, and the effectiveness of SQL operations that are used.

The DVM server supports IBM supported hardware ranging from IBM z196 to the latest IBM models running IBM z/OS 1.13 or later. The technology supports traditional database applications, such as Db2 for z/OS, Information Management System (IMS), Integrated Database Management System (IDMS), ADABAS, and traditional mainframe file systems, such as sequential files, ZFS, Virtual Storage Access Method (VSAM) files, log-stream, and System Management Facilities (SMF).

DVM reduces overall mainframe processing usage and costs by redirecting processing that is otherwise meant for general-purpose processors (GPPs) to zIIPs. DVM provides comprehensive, consumable data that can be readily accessible by any application or business intelligence tools to address consumer demand and stay ahead of the competition.

DVM includes the following key features:

- ▶ Provides a layer of abstraction that shields developers from unique data implementation.
- ▶ Virtualizes IBM Z and non IBM Z data sources in place in real time.
- ▶ Supports modern APIs, such as Java Database Connectivity (JDBC), Microsoft Open Database Connectivity (ODBC), SOAP, and REST (requires z/OS Connect Enterprise Edition).
- ▶ Supports MapReduce for faster data access offloads up to 99% of general processing to lower-cost zIIP specialty engines.
- ▶ Supports Db2 and IMS Direct for efficient large data retrievals.
- ▶ Supports data encryption.

12.8.2 Db2 for z/OS usage

DVM provides unique benefits to organizations that are centered around Db2 for z/OS as their primary management system for business-critical data. DVM leverages Db2 native stored procedures and eliminates the requirement to write separate programs to access non Db2 data sources by using SQL. DVM can query non Db2 data sources and simplify using batch COBOL programs and EXEC SQL to access mainframe data sources that are provisioned by DVM.

Virtualizing Db2 for z/OS data

The DVM server can access Db2 for z/OS by using the Distributed Relational Database Architecture (DRDA) access method or the Resource Recovery Services attachment facility (RRSAF) access method.

The DRDA method allows a higher percentage of the Db2 workload to run in Service Request Block (SRB) mode and to be offloaded to a zIIP specialty engine. Running workloads in SRB mode lowers the total cost of ownership when compared to RRSF by reducing the dependency on z/OS GPP.

Traditional Db2 APIs

DVM allows for reading and writing of Db2 data with transactional integrity by virtualizing Db2 data object identifiers in the Db2 catalog. This capability ensures that any SQL that is issued against Db2 adheres to the transactional integrity that is enforced by the Db2 subsystem.

Db2 Direct access

Db2 Direct reads the underlying Db2 VSAM linear data sets directly without issuing an SQL statement against the Db2 for z/OS subsystem. This access method allows read-only access to the data and provides high performance and bulk data access. In this case, the work is 100% zIIP eligible compared to 60% eligible when Db2 uses DRDA requesters, as shown in Figure 12-1.

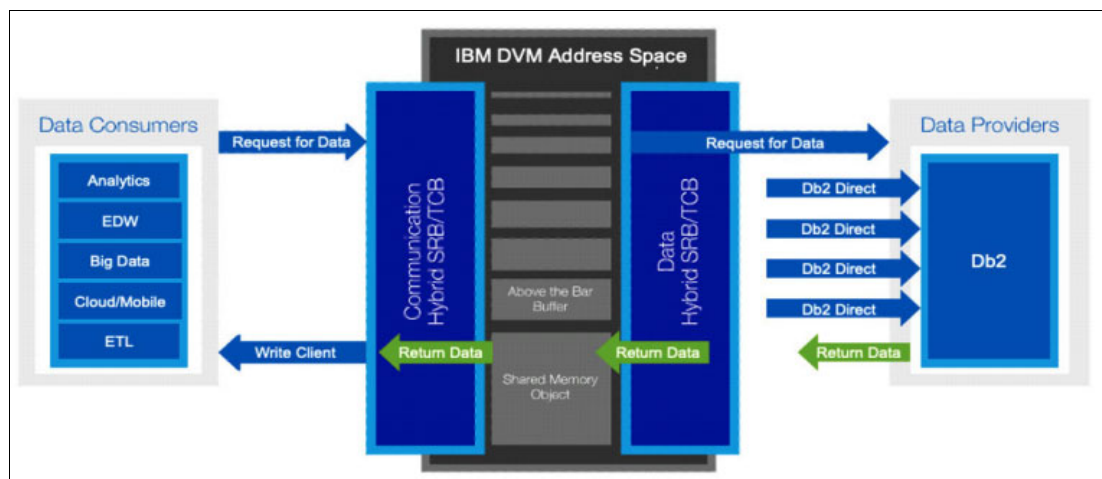


Figure 12-1 Db2 Direct provides direct access without using Db2 resources

Db2 for z/OS unload files

DVM supports Db2 unload files as a data source by activating the model rule AVZMDLDU. When this model is activated, you can directly access a Db2 unload set by using SQL with some limits.

Db2 for z/OS as a data hub

DVM leverages Db2 for z/OS to establish a single metadata model for local and remote data that is accessible through the DVM SQL engine by using common Db2 interfaces as though through a single API, as shown in Figure 12-2. This function means that commonly used APIs that use Db2 SQL can use Db2 for z/OS APIs to access other data sources, such as IMS, VSAM and identity access and management (IAM), Db2 for z/OS, ADABAS, Sequential, OPERLOG/SYSLOG, and DRDA sources like Db2, Oracle, SQL Server, and IBM Informix.

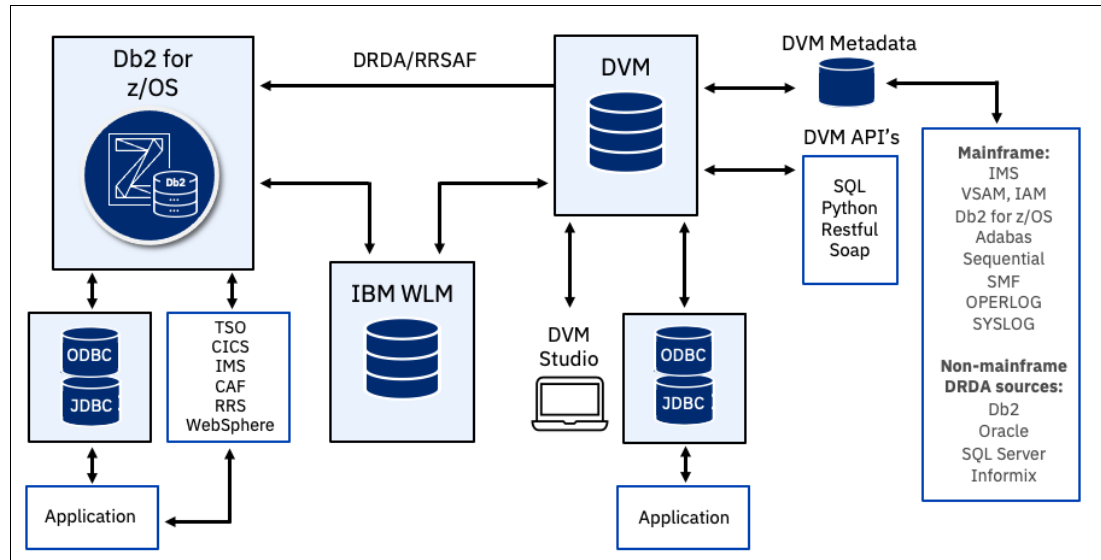


Figure 12-2 Db2 for z/OS as a data hub

By using user-defined table functions (UDTFs), both local and distributed DVM clients can use Db2 common interfaces through SQL and through external and sourced functions. DVM is enhanced to create UDTFs on virtualized data that is defined to the metadata catalog. When a UDTF is defined on a virtual object, it can be accessed by using Db2 SQL and supported attachment facilities.

Support for Integrated DRDA Facility

You can use Integrated DRDA Facility (IDF) for Db2 mainframe applications so that Db2 for z/OS systems can communicate with DVM to access IBM Z and distributed data sources by using DRDA.

Support for this feature is based on the usage of three-part-names within Db2 stored procedures and applications, Db2 RESTful services, MQF for TSO, and the SQL processor by using the file input (SPUFI) facility within the Db2 Interactive (Db2I) primary option menu of ISPF.

IDF support allows a DVM server to participate as an endpoint in multi-homed configurations. Whenever SQL is run in Db2 and a three-part table name designates an IDF endpoint, the SQL is sent to the DVM server for execution. Results are returned to Db2 for z/OS as though it were a local Db2 table.

Applying a COBOL copybook structure to an individual Db2 column

As part of the mainframe modernization movement, many organizations are migrating their legacy non-relational data into Db2 so it is under relational database control. Building a relational data model for this data can be complex and time-consuming, and it requires an extensive amount of data modeling to normalize the data.

Some organizations choose to create a single Db2 table for each data set with an identifier column for use as a search key, and a long varchar column in which to insert the complete sequential or VSAM record as is.

DVM can transform individual data fields within a single Db2 column into a single virtual table by creating a COBOL metadata layout that maps and links a relational structure to the fields in the Db2 multi-field column. SQL queries can be run against the individual data fields in the Db2 column by using the virtual table as though it were a fully normalized multi-column table.

Support for IBM Workload Manager for z/OS

Workload (WLM) classification rules can map workloads coming into the system for access to the DVM Server subsystem type onto three different service class types:

- ▶ AVZ_SCHI ZIIPCLASS=AVZ: High priority for critical work
- ▶ AVZ_SCNM ZIIPCLASS=AVZ: Normal priority for administrative work
- ▶ AVZ_SCTX ZIIPCLASS=AVZ: Client work for client requests

Support for Db2 native RESTful applications

Db2 for z/OS empowers a wide community of developers with a simple and intuitive way to use data and services on IBM Z through RESTful APIs. You can combine the DVM implementation with Db2 native RESTful API support to develop REST interfaces that access both Db2 and non-Db2 mainframe data sources by using Db2 stored procedures that are redirected to DVM by using the IDF. Then, SQL CALL statements to these Db2 stored procedures can be incorporated into Db2 REST services.

Virtualized access from IBM Data Fabric and IBM Cloud Pak for Data

IBM Cloud Pak for Data provides a data virtualization service with integration to DVM. This service facilitates connectivity and access to data sources that are configured by DVM and supports business rules and policies that can be applied at an exposure point within IBM Cloud Pak for Data for downstream traditional and modern mainframe applications.

12.9 IBM OMEGAMON for IBM Db2 Performance Expert on z/OS

IBM OMEGAMON for Db2 Performance Expert on z/OS 5.5 (OMEGAMON for Db2 PE (OMPE)) enables users to assess the efficiency of Db2 and optimize its performance by combining sophisticated response time tracking, reporting, monitoring, and buffer pool analysis features, and expert database analysis functions.

OMEGAMON for Db2 PE introduces support for the following Db2 13 features:

- ▶ Db2 13 introduces the SQL Data Insights feature, which provides three key built-in cognitive functions (BIFs) that allow for injecting deep learning AI capability within Db2 SQL. From a monitoring and reporting perspective, SQL Data Insights exposes elapsed and CPU timers that are included in Class 2 elapsed and CPU times. It represents only those portions that are spent running the built-in scalar functions. With OMPE, you can monitor this new accounting information from both real-time monitoring interfaces and from the best-in-breed batch reporting (ACCOUNTING).

- ▶ Db2 13 adds new statistics counters for the Group Buffer Pool (GBP) statistics storage areas that include the average time that a data area is in a storage class before it is reclaimed, and the average time a directory entry is in a storage class before it is reclaimed. For this feature to work, you must be using IBM z16 hardware. OMPE allows you to monitor average residency times in real time and perform batch record traces. This enhancement enables you to better understand whether the sizing of the GBP structure is adequate.
- ▶ To improve concurrency control, Db2 13 introduces a **CURRENT LOCK TIMEOUT** special register that can be set from an application or a Db2 system profile. With OMPE, you receive real time and batch reporting capabilities for tracking the respective functions in the Db2 System Statistics and Accounting reporting areas. The extensions in the Deadlock and Timeout Db2 events are exposed in the event exception workspaces and batch reports. Additionally, the new Instrumentation Facility Component Identifier (IFCID) 427 tracking of the **SET CURRENT LOCK TIMEOUT** statement is supported in batch RECTRACE.
- ▶ Although monitoring data consistency and control concurrency by using the real-time monitoring is possible, this method is cumbersome to find the offending resource that holds a lock or latch for too long. For this reason, Db2 13 introduces a data section in Accounting IFCID 3 that tracks the longest lock or latch waiter, the longest Service Task Wait, and information about the longest Page Latch Wait. With OMPE BATCH reporting and the Enhanced 3270 (E3270) Thread History function, you can monitor this information from a historical perspective and improve application monitoring and Db2 serviceability.
- ▶ Db2 13 provides a new Database Access Thread (DBAT) termination behavior that allows for handling of large Distributed Data Facility (DDF)-based workloads better. As part of this enhancement, new metrics are exposed in the Global DDF Activity statistics records that help analyze and monitor the impact of this feature. OMPE helps to monitor these new metrics in both real time and batch by providing extended System Statistics workspaces and Report sections.
- ▶ Db2 13 now caches successful execute on plan authorization checks when Access Control Authorization Exit (ACAIE) is used for access control. Furthermore, Db2 adds a smarter algorithm for managing the authorization IDs in the plan authorization cache. These changes allow the caching of more authorization IDs per plan. Improvements that are related to the monitoring and reporting of the plan authorization cache provide evidence for a reduced RACF contention when checking for a plan EXECUTE privilege by using OMPE.
- ▶ With Db2 13, IFCID 396 is introduced to record more detailed information about an index page split. When an index split is considered to be an abnormal split process (such as when the total elapsed time of index split is > 1 second), the new IFCID 396 is generated that holds detailed information about this index split, including the DBID, PSID, member ID, URID, Page number, and other items. This information is helpful for both customers and developers to identify the root cause of **INSERT** performance issues. OMPE batch RECTRACE reporting allows you to format this new IFCID and helps with analyzing the potential **INSERT** performance issue.

12.9.1 Monitoring SQL Data Insights accounting metrics

To monitor SQL Data Insights, Db2 introduced new accounting instrumentation in the QWAC data section that is part of IFCID 3 and IFCID 148. Although IFCID 3 is used in batch reporting, IFCID 148 is used as part of the real-time monitoring function of OMPE.

BATCH reporting support

Extensions to the batch ACCOUNTING and RECTRACE report sets were made to track the respective SQL Data Insights metrics.

Using the SYSIN DD statement that is shown in Example 12-1 in the batch reporting job creates an ACCOUNTING REPORT and a RECTRACE report. ACCOUNTING TRACE support is also available (not shown).

Example 12-1 Sample SYSIN DD

```
ACCOUNTING
  REPORT
    LAYOUT(LONG)
RECTRACE
  TRACE LEVEL(LONG)
EXEC
```

Example 12-2 shows the SQL Data Insights metrics in the ACCOUNTING report (only CLASS 1 / 2 times block are shown here).

Example 12-2 Sample SQL Data Insights metrics in the ACCOUNTING report

AVERAGE	APPL(CL.1)	Db2 (CL.2)
-----	-----	-----
ELAPSED TIME	1.340014	1.337055
NONNESTED	1.340014	1.337055
STORED PROC	0.000000	0.000000
UDF	0.000000	0.000000
TRIGGER	0.000000	0.000000
CP CPU TIME	2.633505	2.632807
AGENT	0.698717	0.698019
NONNESTED	0.698717	0.698019
STORED PRC	0.000000	0.000000
UDF	0.000000	0.000000
TRIGGER	0.000000	0.000000
SQL DI	N/A	0.000000
PAR.TASKS	1.934789	1.934789
SQL DI	N/A	1.648177
SE CPU TIME	1.920307	1.920295
NONNESTED	0.000000	0.000000
STORED PROC	0.000000	0.000000
UDF	0.000000	0.000000
TRIGGER	0.000000	0.000000
SQL DI	N/A	0.000000
PAR.TASKS	1.920307	1.920295
SQL DI	N/A	1.569330
SUSPEND TIME	0.000000	0.667829
AGENT	N/A	0.638205
PAR.TASKS	N/A	0.029625
STORED PROC	0.000000	N/A
UDF	0.000000	N/A
NOT ACCOUNT.	N/A	N/C
Db2 ENT/EXIT	N/A	46.00
EN/EX-STPROC	N/A	0.00
EN/EX-UDF	N/A	0.00
EN/EX-SQL DI	N/A	1371968.00

DCAPT.DESCR.	N/A	N/A
LOG EXTRACT.	N/A	N/A

The report reflects an aggregation of the respective metrics that consider the respective OMPE IDs, such as PLANNAME or AUTHID. Special attention can be given to the parallel task processing metric for Specialty Engine (SE), which is computed by the reporting function. The fields show the CPU and elapsed time that were spent running the AI built-in functions and the number of invocations of AI functions.

In addition to the BATCH reporting function, the Performance Database is extended to include the new SQL Data Insights metrics. You can process these metrics by using the Spreadsheet Utility to generate CSV files from SMF or GTF data.

For the RECTRACE report, OMPE shows the details, as shown in Example 12-3.

Example 12-3 Sample RECTRACE report

CLASS 2 Db2 ELAPSED TIME	0.215507	Db2 ENTRY/EXIT
EVENTS	2	
TCB TIME	0.029649	NON-ZERO CLASS 2
YES		
STORED PROC ELAPSED TIME	0.000000	CLASS 2 DATA
COLLECTED	YES	
STORED PROCEDURE TCB TIME	0.000000	STORED PROC.
ENTRY/EXITS	0	
UDF ELAPSED TIME	0.000000	UDF SQL ENTRY/EXITS
EVENTS	0	
CP CPU TIME UDF	0.000000	SE CPU TIME
0.042903		
TRIG ELAP TIME UNDER ENCLAVE	0.000000	SE ELIGIBLE CP CPU
TIME	0.000000	
TRIG TCB TIME UNDER ENCLAVE	0.000000	QWACTRTT_ZIIP
0.000000		
TRIG ELAP TIME NOT UNDER ENCLAVE	0.000000	ELAPSED TIME
ELIGIBLE FOR ACCEL	0.000000	
TRIG TCB TIME NOT UNDER ENCLAVE	0.000000	CP CPU TIME
ELIGIBLE FOR ACCEL	0.000000	
		SE CPU TIME
ELIGIBLE FOR ACCEL	0.000000	
SQL DATA INSIGHTS ELAPSED TIME	0.174283	SQL DATA INSIGHTS
CPU TIME	0.025273	
SQL DATA INSIGHTS ENTRY/EXITS	19944	SQL DATA INSIGHTS
zIIP TIME	0.034893	

Real-time monitoring support

Real-time monitoring support for the SQL Data Insights metrics is added to the Performance Expert Client (PE Client) interfaces and the E3270 interface when you zoom into Thread Details from a Thread Summary workspace.

These metrics show the amount of CPU or SE time and SQL Data Insights Events that occur when the respective thread detail snapshot is taken. Continuing to refresh the details panel shows whether activity that is triggered by SQL Data Insights is performed by the respective thread.

Figure 12-3 shows how these metrics are displayed in the E3270 Thread Detail Accounting workspace (note the highlighted fields).

The screenshot displays the DB2 Thread Detail Accounting workspace. At the top, it shows the command 'KDPTHDA2' and the title 'DB2 Thread Detail Accounting'. The SMF ID is 'RS27' and the DB2 ID is 'QD1A'. Below this are several control buttons: 'Acct', 'Cls3', 'BP', 'GBP', 'SQLC', 'SQLT', 'EXP', 'Lock', 'Can', and '>'. The main content is divided into two sections: 'Thread Information' and 'Class 1/2 Times'.

Thread Information:

Plan.....	TS7885P	Correlation ID.....	TS7885
Authorization ID.....	TS7885	Connection.....	TSO
Job Name.....	TS7885	Commits.....	222
Aborts.....	0	MVS Status.....	SWAPPED-
Status.....	SWAPPED-		

Class 1/2 Times:

Elapsed class 1.....	6h 10m	In-DB2 Elapsed Class 2....	.651400s
Non Nested Class 1.....	6h 10m	In-DB2 Non-Nested Class 2/	.651400s
SE CPU Class 1.....	.085769s	In-DB2 SE CPU Class 2....	.085122s
CP CPU Time Class 1.....	.457698s	In-DB2 CP CPU Time Class 2	.290357s
Agent Class 1.....	.457698s	In-DB2 Agent CPU Time Clas	.290357s
Agent Non-Nested Class 1..	.457698s	In-DB2 Non-Nested CPU Time	.290357s
Stored Proc Class 1.....	0.00000s	In-DB2 SP Class 2/3.....	0.00000s
SP CPU Class 1.....	0.00000s	In-DB2 SP CPU Time Class 2	0.00000s
UDF Class 1.....	0.00000s	In-DB2 UDF Class 2/3.....	0.00000s
UDF CPU Class 1.....	0.00000s	In-DB2 UDF CPU Time Class	0.00000s
Parallel Tasks CPU Class 1	0.00000s	In-DB2 Parallel CPU Time C	0.00000s
Triggers Class 1.....	0.00000s	In-DB2 Waiting Time Class	.361043s
Trigger CPU Class 1.....	0.00000s	In-DB2 Suspend Time Class	.233384s
Elapsed Outside DB2 Class	6h 10m	In-DB2 Suspend Time Paralle	.106852s
CP CPU Outside DB2 Class 2	.167340s	In-DB2 Suspend Time Agent	.126532s
Non-Nested Outside DB2 Cla	.167340s	In-DB2 Not Accounted Class	.149389s
SDI Elapsed Time.....	0.00000s	SDI CPU Time.....	0.00000s
SDI CPU ZIIP Time.....	0.00000s	SDI Events.....	0

Figure 12-3 Sample Thread Detail Accounting workspace

Besides the real-time zoom-in capability from Thread Summary, the Thread History in E3270 is extended to include the SQL Data Insights metrics.

Support for the PE Client is similar. Note the Data Insight labels in Figure 12-4.

Class 1,2,3			
	In application (Class 1)	In DB2 (Class 2/3)	Outside DB2 (Class 2)
Elapsed time	1:23:47.366	0.056986	1:23:47.309
Non-nested	1:23:47.366	0.056986	
Stored procedure	0.000000	0.000000	
User defined function	0.000000	0.000000	
Data Insight		0.000000	
Trigger	0.000000	0.000000	
CP CPU time	0.015006	0.009675	0.005330
Agent	0.015006	0.009675	
Non-nested	0.015006	0.009675	0.005330
Stored procedure	0.000000	0.000000	
User defined function	0.000000	0.000000	
Data Insight		0.000000	
Trigger	0.000000	0.000000	
Parallel tasks	0.000000	0.000000	
SE CPU time		0.000000	
Data Insight		0.000000	
Waiting time		0.047310	

Figure 12-4 PE Client example

12.9.2 Monitoring GBP residency times (IBM z16)

OMPE allows you to monitor average residency times in real time and perform batch record traces. The new IBM z16 hardware allows for the capture of GBP residency time, which helps you better understand GBP performance characteristics.

The Db2 instrumentation changes are in IFCID 230 (Group Buffer Pool Attributes) and IFCID 254 (Coupling Facility Cache Structure Statistics).

BATCH reporting support

The RECTRACE report supports the data and entry residency times by providing the metrics that are shown in Example 12-4.

Example 12-4 RECTRACE report examples

CONNECT	INSTANCE	END_USER	WS_NAME	TRANSACTION
CORRNAME	CONNTYPE	RECORD TIME	DESTNO ACE IFC	DESCRIPTION
CORRNMBR		TCB CPU TIME	ID	DATA
N/P	DB3D6E00F125	N/P	N/P	N/P
N/P	'BLANK'	03:56:00.69592458	20909	1 254 CF CACHE NETWORKID: R
N/P		N/P		STRUCT STATS

GROUP BUFFER POOL NAME	0		EXPLICIT XI COUNTER	0
READ HIT		356	CHANGED PAGE WRITE HIT	786
READ MISS DIRECTORY HIT		87	CLEAN PAGE WRITE HIT	0
READ MISS ASSIGNMENT SUPPRESSED		0	WRITE MISS CACHE FULL	0
READ MISS NAME ASSIGNED		192	DIRECTORY ENTRY RECLAIM	0
READ MISS CACHE FULL		0	DATA ENTRY RECLAIM	387
SEC-GBP CHANGED PAGE WRITE HIT		0	SEC-GBP DIRECTORY ENTRY	0
SEC-GBP WRITE MISS CACHE FULL		0	SEC-GBP DATA ENTRY	0
DATA AREA RESID TIME		0	DIR ENTRY RESID TIME	0

N/P	DB3AF9A75148	N/P	N/P	N/P
N/P	'BLANK'	03:56:00.69600310	20910	2 230 GBP ATTRIBUTES NETWORKID: L
N/P		N/P		

GROUP BUFFERPOOL ID	:	0	ERROR FLAGS	:
ALLOCATED GBPOOL SIZE (4K)	:	2048	CURRENT DIR TO DATA RATIO	:
ACTUAL # OF DIR ENTRIES	:	4017	PENDING DIR TO DATA RATIO	:
ACTUAL # OF DATA ENTRIES	:	801	GBP CHECKPOINT INTERVAL (MIN)	:
DIRECTORY-ENTRY-RECLAIM	:	0	DATA-ENTRY-RECLAIM	:
TOTAL-CHANGED	:	0	XI-DIRECTORY-ENTRY-RECLAIM	:
MODE	:	SIMPLEX		
SEC-GBP ALLOC	:	N/A	SEC-GBP ALLOC DIRECTORIES	:
DATA AREA RESIDENCY TIME	:	0	DIRECTORY ENTRY RESID TIME	:
QGBBERC	:	0	QGBBERS	:

Real-time monitoring support

Support for the data area and data and directory entry residency times creates the Performance Expert Client interface and the E3270 host interface.

For the Performance Expert Client, support is added to the GBP details section (IFCID 254) in the Statistics Details pages (see the Data area Residency time and Directory entry Residency time fields), as shown in Figure 12-5.

Main		Global GBP: GBP0	
Global group buffer pool		GBP0	
Read			
Total	155,553	Data entry	120
Hit (%)	5.1	Data entry reclaim	137,006
- hit	7,928	Directory entry	1,076
- miss directory hit	7,541	Directory entry reclaim	1,615
- miss suppressed	41,298	XI Directory entry reclaim	1,672
- miss name assigned	98,786	Data area Residency time	0.000000
- miss cache full	0	Directory entry Residency time	0.000000
Castout	139,286	Secondary group buffer pool	
Total changed	120	- Changed page write hit	0
Changed page write hit	426,045	- Write miss cache full	0
Clean page write hit	0	- Directory entry	0
Write miss cache full	0	- Data entry	0
		- Total changed	0

Figure 12-5 Example Global GBP details in the Performance Expert Client

Furthermore, the GBP attributes changes are visible when you drill down into the Group Buffer Pools section of the System Parameter page, as shown in Figure 12-6.

Main		Group Buffer Pools: GBP0	
Group buffer pool name	GBP0	Current GBP cache setting	YES
Allocated buffer pool size (4K)	2,048	CF directory entry reclaimed	1,615
Current directory to data ratio	10	CF data entry reclaimed	137,125
Class castout threshold	5	Directory entry stolen / cross-invalidation signals sent	1,672
GBP castout threshold	30	Allocated data entries in changed state	25
Checkpoint interval (minutes)	4	Duplex or simplex indicator	SIMPLEX
Actual directory entry	6,131	Directory entries allocated for secondary GBP	0
Actual data entry	613	Data entries allocated for secondary GBP	0
Pending directory to data ratio	10	Castout class level threshold	0
Automatic recovery	YES	Data Area Residency Time	0.000000
		Directory Entry Residency Time	0.000000

Figure 12-6 Example GBPs in the Performance Expert Client

From an E3270 perspective, you can monitor the data from the System Statistics main workspace by navigating to the Buffer Pool workspace and then selecting the **Group Buffer Pool** tab. Zoom in to the respective GBPs and select the **Performance Counters** subtab to display the residency times, as shown in Figure 12-7 on page 191.


```

Command ==>
KDPGBPPR
Data Sharing Group Information
SMF ID : RS27
DB2 ID : QDB8

Sync Rd / GBP wrt Prefetch Castout P-Locks Secondary GBP Per

Performance Counters
GBP ID..... GBP0
Error Flag..... OK
Current Directory entry..... 10
Pending Directory entry..... 10
Castout class..... 5
GBP threshold..... 30
Autorec..... YES
GBP Cache..... YES
GBP Checkpoint interval..... 4
GBP Size..... 2048
Number of directory entries..... 6131
Number of data entries..... 613
Directory entry reclaim..... 1615
Data entry reclaim..... 137245
Directory entry stolen..... 1672
Data entries changed..... 0
GBP Indication..... SIMPLEX
Secondary GBP size..... 0
Number of dir entries in GBP 2..... 0
Number of dat entries in GBP 2..... 0
Class level..... 0
Data Area Residency Time..... 0.00000s
Directory entry Residency Time..... 0.00000s

```

Figure 12-7 Example of Performance Counters

For the GBP attributes display, also follow the System Statistics main workspace by navigating to the Buffer Pool workspace and then selecting the **Global Buffer Pool** tab. Zoom into the respective Global Buffer Pool to display the residency times, as shown in Figure 12-8.

```

Command ==>
KDPGBPDT
Data Sharing Group Information
SMF ID : RS27
DB2 ID : QDA7

Global Buffer Pool Detail for GBP0
Global Buffer Pool Name..... GBP0
Read Hit Counter..... 18.9K
Read Miss Directory Hit Counter..... 27.4K
Read Miss Assignment Suppressed Counter..... 70.6K
Read Miss Name Assigned..... 294.9K
Read Miss Cache Full Counter..... 0
Number of Castout Operations Performed..... 392.1K
Number of Data Entries in Changed State..... 20
Changed Page Write Hit Counter..... 892.4K
Clean Page Write-Hit Counter..... 0
Write Miss Cache Full Counter..... 0

Number of Allocated Data Entries (snapshot)..... 20
Data Entry Reclaim Counter..... 389.0K
Number of Allocated Directory Entries (Snapshot)..... 3.3K
Directory Entry Reclaim Counter..... 6.7K
XI Directory Entry Reclaim Counter..... 6.7K
Changed Page Write Hit Counter Secondary GBP..... 0
Write Miss Secondary Cache Full Counter..... 0
Dir Counter for Secondary Group Buffer Pool..... 0
Data Entry Counter of Secondary GBP..... 0
Total Changed Counter for Secondary GBP..... 0
Data Area Residency Time..... 0.00000s
Directory Entry Residency Time..... 0.00000s

```

Figure 12-8 Example of GBP Detail

12.9.3 Monitoring SET CURRENT LOCK TIMEOUT related Db2 changes

OMPE supports several instrumentation changes that were introduced with the Db2 13 SET CURRENT LOCK TIMEOUT feature. The changes are in data section QXST (IFCID 2 and 3 and 148) and in the Lock timeout and Deadlock IFCIDs (172 and 196).

In addition to these changes, a new IFCID 437 is written when a SET CURRENT LOCK TIMEOUT statement is issued from an application or a profile.

BATCH reporting support

The BATCH ACCOUNTING and STATISTICS REPORT, TRACE, and RECTRACE report sets are enhanced to show the new metrics. PDB and CSV support for ACCOUNTING, STATISTICS, and RECTRACE for IFCIDs 172 and 196 is available.

Example 12-5 shows the new metrics in the ACCOUNTING REPORT.

Example 12-5 New metrics in the ACCOUNTING REPORT report set

SQL DML	AVERAGE	TOTAL	SQL DCL	TOTAL
SELECT	0.00	0	LOCK TABLE	4
INSERT	0.00	0	GRANT	0
ROWS	0.00	0	REVOKE	0
IAG1	0.00	0	SET CURR.SQLID	0
IAG2	0.00	0	SET HOST VAR.	0
UPDATE	0.00	0	SET CUR.DEGREE	0
ROWS	0.00	0	SET RULES	0
MERGE	0.00	0	SET CURR.PATH	0
DELETE	0.00	0	SET CURR.PREC.	0
ROWS	0.00	0	SET TIMEOUT	3
			FROM APPL.	1
DESCRIBE	0.75	9	FROM PROF.	2
DESC.TBL	0.00	0	CONNECT TYPE 1	0
PREPARE	0.75	9	CONNECT TYPE 2	0
OPEN	0.00	0	SET CONNECTION	0
FETCH	0.00	0	RELEASE	0
ROWS	0.00	0	CALL	0
CLOSE	0.00	0	ASSOC LOCATORS	0
			ALLOC CURSOR	0
DML-ALL	1.50	18	HOLD LOCATOR	0
			FREE LOCATOR	0
			DCL-ALL	7

Example 12-6 shows the new metrics in the STATISTICS REPORT.

Example 12-6 New metrics in the STATISTICS REPORT report set

SQL DCL	QUANTITY	/SECOND	/THREAD	/COMMI
LOCK TABLE	12.00	0.05	0.03	0.0
GRANT	0.00	0.00	0.00	0.0
REVOKE	0.00	0.00	0.00	0.0
SET HOST VARIABLE	75.00	0.31	0.18	0.0
SET CURRENT SQLID	68.00	0.28	0.16	0.0
SET CURRENT DEGREE	0.00	0.00	0.00	0.0
SET CURRENT RULES	0.00	0.00	0.00	0.0

SET CURRENT PATH	0.00	0.00	0.00	0.00
SET CURRENT PRECISION	0.00	0.00	0.00	0.00
SET CURRENT LOCK TIMEOUT	8.00	0.03	0.02	0.01
FROM APPLICATION	5.00	0.02	0.01	0.00
FROM PROFILE	3.00	0.01	0.01	0.00
CONNECT TYPE 1	0.00	0.00	0.00	0.00
CONNECT TYPE 2	0.00	0.00	0.00	0.00
RELEASE	0.00	0.00	0.00	0.00
SET CONNECTION	0.00	0.00	0.00	0.00
ASSOCIATE LOCATORS	0.00	0.00	0.00	0.00
ALLOCATE CURSOR	0.00	0.00	0.00	0.00
HOLD LOCATOR	0.00	0.00	0.00	0.00
FREE LOCATOR	0.00	0.00	0.00	0.00
TOTAL	163.00	0.67	0.38	0.13

Example 12-7 shows an example of IFCID 437 formatting.

Example 12-7 Example of IFCID 437 formatting

Db2 VERSION: V13 PAGE DATE: 10/07/21

PRIMAUTH	CONNECT	INSTANCE	END_USER	WS_NAME	TRANSACT					
ORIGAUTH	CORRNAME	CONNTYPE	RECORD TIME	DESTNO ACE	IFC	DESCRIPTION	DATA			
PLANNAME	CORRNMBR		TCB CPU TIME			ID				
TS3473A	TSO	DA6D158FDAEC	TS3473A	TSO		TS3473A				
TS3473A	TS3473A	TSO	10:52:48.65548900	78	1 437	TIMEOUT	NETWORKID: ROCKNET1	LUNAME: DD1ADB2	LUWSEQ: 1	
DSNESPCS	'BLANK'		0.02568870			DEADLOCK CTRL				

SET CURRENT LOCK TIMEOUT										
SET DONE BY.....: PROFILE TABLE STATUS.....: SUCCESSFUL										
OLD TIMEOUT VALUE.....: N/P NULL INDICATOR FOR OLD VALUE.: YES										
NEW TIMEOUT VALUE.....: -1 NULL INDICATOR FOR NEW VALUE.: NO										

TS3473A	TSO	DA6D159CE8D2	TS3473A	TSO		TS3473A				
TS3473A	TS3473A	TSO	10:53:02.33921100	79	1 437	TIMEOUT	NETWORKID: ROCKNET1	LUNAME: DD1ADB2	LUWSEQ: 1	
DSNESPCS	'BLANK'		0.03507180			DEADLOCK CTRL				

SET CURRENT LOCK TIMEOUT										
SET DONE BY.....: APPLICATION STATUS.....: SUCCESSFUL										
OLD TIMEOUT VALUE.....: N/P NULL INDICATOR FOR OLD VALUE.: YES										
NEW TIMEOUT VALUE.....: 0 NULL INDICATOR FOR NEW VALUE.: NO										
PACKAGE CONSISTENCY TOKEN.: X'1A0D8BD811DAADD8'										
COLLECTION ID.....: COLLID1										
PACKAGE ID.....: DSNESM68										

Real-time monitoring support: SQL statement data

The QXST changes (SQL STATEMENT DATA) are displayed in the Performance Expert Client and the E3270 user interface. They are displayed for the System Statistics side and for the thread detail workspaces. The following two sections provide details about the E3270 support. The PE client displays are similar in nature.

E3270 workspace changes for System Statistics

These changes are available from both the single Db2 subsystem and the Db2Plex view. Go to the System Statistics main workspace and then select the **SQL Counts** tab (SQLC for IBM Db2 Plex® view, SQL for single Db2 view). On the SQL Counts workspace, select the **DCL** tab to view the new metrics, which are highlighted in Figure 12-9 and Figure 12-10.

Command ==> _____ DSG Name : QDS7
 KDPPSQL2 _____ Data Sharing Group Information _____

<< DML DCL DDL DDL2 RID PARAL NESTED PREP >>

Data Control Language (DCL)

Columns 4 to 5 of 7 Rows 1 to 21 of 21

◦MVS	◦DB2	◦Description	Value	+Delta
			0	
			0	
RS27	QDA7	Lock Table	380	0
RS27	QDA7	Grant	202	0
RS27	QDA7	Revoke	0	0
RS27	QDA7	Set Current SQLID	205	0
RS27	QDA7	Set Host Variable	481	0
RS27	QDA7	Connect(Type 1)	0	0
RS27	QDA7	Connect(Type 2)	4	0
RS27	QDA7	Release	0	0
RS27	QDA7	Set Connection	0	0
RS27	QDA7	Set Current Degree	85	0
RS27	QDA7	Set Current Rules	0	0
RS27	QDA7	Associate Locator	49	0
RS27	QDA7	Allocate Cursor	49	0
RS27	QDA7	Set Current Path	1	0
RS27	QDA7	Hold Locator	0	0
RS27	QDA7	Free Locator	0	0
RS27	QDA7	Set Current Precision	1	0
RS27	QDA7	Transfer Ownership	0	0
RS27	QDA7	Set current lock timeout(SQL)	0	0
RS27	QDA7	Set current lock timeout(P)	0	0
RS27	QDA7	Total DCL Count	8934	0

Figure 12-9 Example of Db2 Plex view drill-down

Command ==> _____ SMF ID : RS27
 KDPPSQL2 _____ SQL Counts - Data Control Language (DCL) DB2 ID : QD1A

<< DML DCL DDL DDL2 RID PARAL NESTED PREP >>

Interval in seconds..... 2

Columns 1 to 4 of 4 Rows 1 to 21 of 21

Description	Value	Delta	+Rate
Lock Table	543	0	.00
Grant	56	0	.00
Revoke	10	0	.00
Set Current SQLID	8176	0	.00
Set Host Variable	40588	0	.00
Connect(Type 1)	0	0	.00
Connect(Type 2)	1588	0	.00
Release	0	0	.00
Set Connection	0	0	.00
Set Current Degree	0	0	.00
Set Current Rules	154	0	.00
Associate Locator	622	0	.00
Allocate Cursor	622	0	.00
Set Current Path	49	0	.00
Hold Locator	1520	0	.00
Free Locator	1561	0	.00
Set Current Precision	0	0	.00
Transfer Ownership	0	0	.00
Set current lock timeout(SQL)	0	0	.00
Set current lock timeout(P)	0	0	.00
Total DCL Count	56157	0	.00

Figure 12-10 Example of single Db2 view drill-down

E3270 screen space for Thread Detail

To display the QXST changes in a Thread Detail view, from the single Db2 system view, navigate to the Thread Summary workspace and then drill down into a specific thread. Select the **SQL** tab and then the **DCL** tab on the SQL Counts workspace to view the details panel, as shown in Figure 12-11. You can access the same details panel by navigating from the Db2Plex thread summary view and zooming into the thread details.

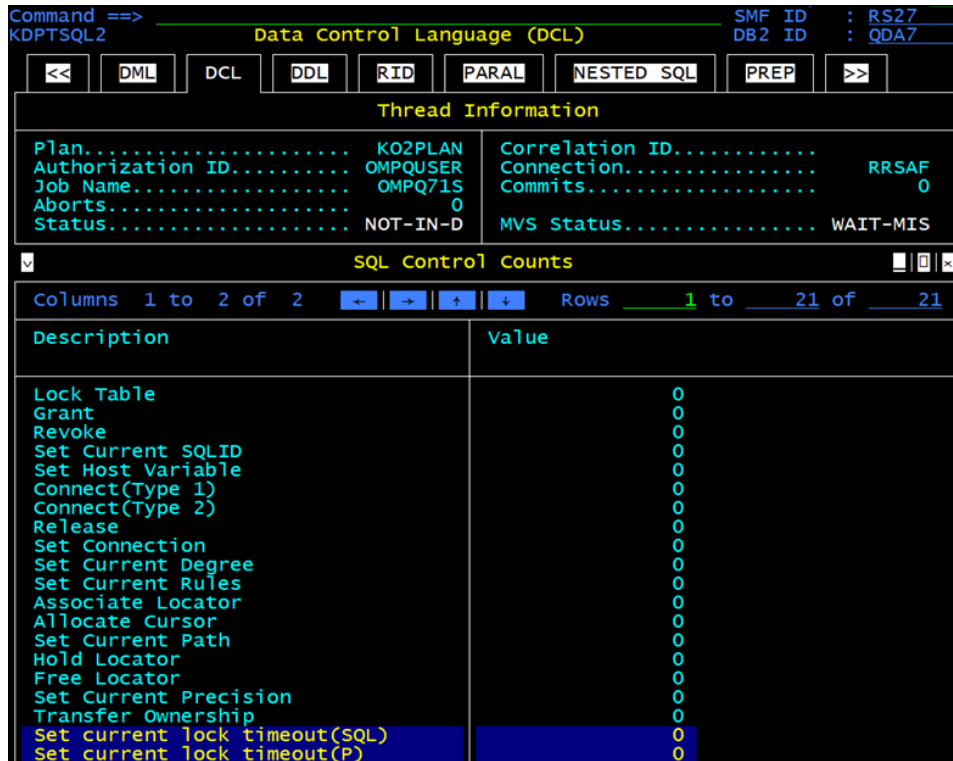


Figure 12-11 Example of SQL Counts “DCL”

Real-time monitoring support: Timeout and deadlock event details

You can monitor the changes for lock timeout and deadlock events by using either the OMPE Performance Expert Client Event Exception detail panels or the E3270 Events detail panels.

The following workflow uses the E3270 interface to illustrate the enhancements, but the Performance Expert Client changes are identical in nature.

To access and collect Db2 Events, the following tasks must be done:

- ▶ Db2 Event trace collection must be turned on.
- ▶ Timeout and deadlock data must be collected.

On the main single Db2 panel, navigate to the Events option (option E) and select a timeframe and event filter, as shown in Figure 12-12.

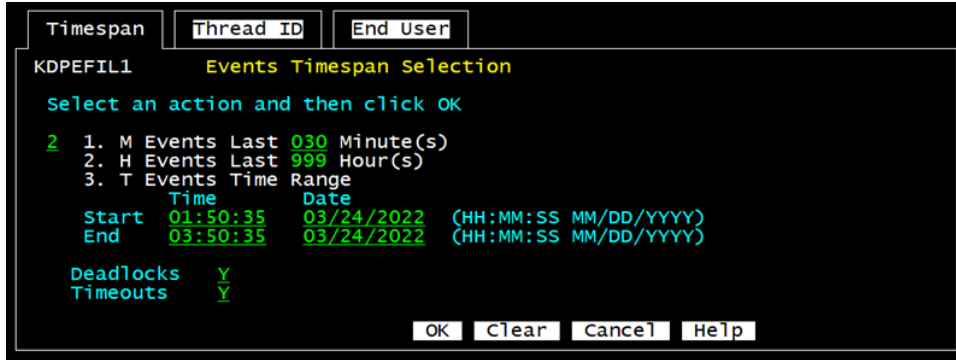


Figure 12-12 Example “Events Timespan Selection”

Click **OK**. The deadlock and timeout events are shown in the Events Summary workspace (Figure 12-13).

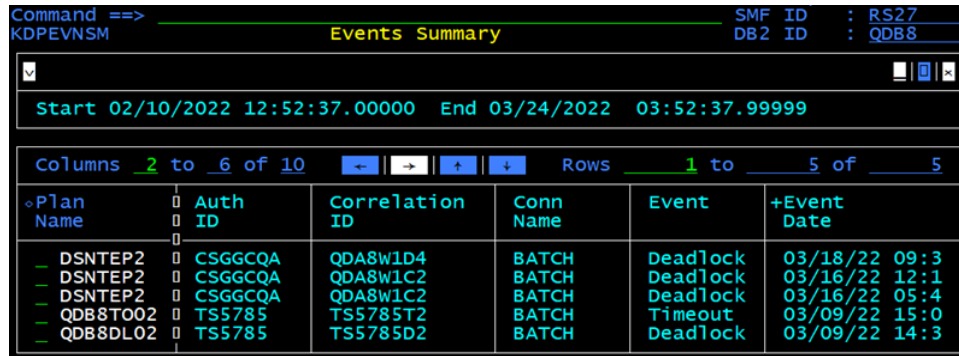


Figure 12-13 Example “Events Summary”

By zooming into the list of events for deadlock and timeout, you can see more metrics that support the enhanced timeout management function that was introduced by Db2 13:

- ▶ Timeout Detail
- ▶ Deadlock Details

Timeout Detail

The Interval Source field indicates what setting triggered the timeout condition (IRLM, special register, or subsystem parameter), as shown in Figure 12-14 on page 197.

```

Command ==>                                Plex ID : RS27
KDPENVTD                                     Sys ID  : QDB8
Events Timeout Detail

Primary AuthID..... TS5785
Instance Number..... DB2DE0E8
Plan Name..... QDB8T002
Correlation ID..... TS5785T2
Connection Name..... BATCH
End User..... TS5785
Workstation Name..... BATCH
Transaction Name..... TS5785T2
Requested Function..... Lock
Requested State..... Exclusiv
Requested Duration..... Commit
Requested Flag..... Uncondit
Owning Work Unit..... 047C0054
Resource Type..... Row Lock
Timeout Interval..... 30
Interval Source..... IRLMRWT

```

Figure 12-14 Example Events Timeout Detail

Zoom into the Blocker Details to display the Holder Timeout Interval and the Holder Interval Source fields (Figure 12-15).

```

Command ==>                                SMF ID : RS27
KDPENVTB                                     DB2 ID  : QDB8
Events Timeout Blocker Detail

Blocker Type..... Holder
Member Name..... QDB8
Primary AuthID..... TS5785
Plan Name..... QDB8T001
Correlation ID..... TS5785T1
Connection Name..... BATCH
End User..... TS5785
Workstation Name..... BATCH
Transaction Name..... TS5785T1
LUW Network ID..... ROCKNET1
LUW LU Name..... QDB8DB2
Instance Number..... DB2DE0E8
Lock State..... Exclusiv
Requested Duration..... Commit
Statement Type..... Static
Statement ID..... 802052
Holder Timeout Interval..... 30
Holder Interval Source..... IRLMRWT

```

Figure 12-15 Example Events Timeout Blocker Detail

Deadlock Details

Select the details of a deadlock event. The workspace that is shown in Figure 12-16 opens.

```

Command ==> KDPENVDD Events Deadlock Detail SMF ID : RS27
DB2 ID : QDB8

Date Deadlock Detected..... 03/09/22
Time Deadlock Detected..... 14:35:02
Primary AuthID..... TS5785
Plan Name..... QDB8DL02
Correlation ID..... TS5785D2
Conn Type..... BATCH
End User..... TS5785
Workstation Name..... BATCH
Transaction Name..... TS5785D2
Instance Number..... DB2DDA27
Resources Involved..... 2
Interval Counter..... 341694

Columns 2 to 5 of 5 Rows 1 to 2 of 2
Resource Type Blocker AuthID Blocker Plan Name Waiter AuthID Waiter Plan Name
Partition Lockin TS5785 QDB8DL02 TS5785 QDB8DL01
Partition Lockin TS5785 QDB8DL01 TS5785 QDB8DL02
  
```

Figure 12-16 Example Events Deadlock Detail

Drill down into the Resource details and locate the Waiter Source of the Worth value, which can be Global Variable or Other (see Figure 12-17). For Global Variable, Db2 uses the DEADLOCK_RESOLUTION_PRIORITY global variable to specify a deadlock resolution priority value that is used in resolving deadlocks with other threads.

```

Command ==> KDPENVDB Events Resource Details SMF ID : RS27
DB2 ID : QDB8

Event Date..... 03/09/22
Event Time..... 14:35:02
Resource Type..... Partitio

Resource Name
Database 393
PSID 5
Partition Number 01

Help for Resource Name
Waiter Plan Name..... QDB8DL01 Blocker Plan Name..... QDB8DL02
Waiter Corrid..... TS5785D1 Blocker Corrid..... TS5785D2
Waiter AuthID..... TS5785 Blocker AuthID..... TS5785
Waiter End User..... TS5785 Blocker End User..... TS5785
Waiter Workstation Name... BATCH Blocker Workstation Name.. BATCH
Waiter Transaction Name... TS5785D1 Blocker Transaction Name.. TS5785D2
Waiter Connection Name.... BATCH Blocker Connection Name... BATCH
Waiter Statement ID..... 802046 Blocker Statement ID..... 802049
Waiter Statement Type.... Static Blocker Statement Type... Static
Waiter Flags..... 10 Blocker Flags..... 04
Waiter LUW Network ID.... ROCKNET1 Blocker LUW Network ID... ROCKNET1
Waiter Instance Number... DB2DDA27 Blocker Instance Number... DB2DDA27
Waiter Thread Token..... C4C2F2C4 Blocker Thread Token..... C4C2F2C4
Waiter Duration..... Commit Blocker Duration..... Commit
Waiter Member Name..... QDB8 Blocker Member Name..... QDB8
Waiter Requested Function. Lock
Waiter Db2 Assigned Worth. 18
Waiter Source of the Worth Other
  
```

Figure 12-17 Example Events Resource Detail

These details represent a fraction of the Db2 13 day-one support that is provided by the latest version of IBM OMEGAMON for Db2 PE. For more information about these new capabilities, see the [IBM OMEGAMON for Db2 Performance Expert on z/OS 5.5 library](#).

12.10 Summary

Although Db2 13 itself provides a wealth of critical new features in areas such as performance, usability, availability, security, simplification, and application management, access to tools that seamlessly support these new features is important. The day-one support that is provided by the Db2 Tools that are described in this chapter helps ensure that you and your organization can take advantage of what this latest Db2 version offers.



IBM Db2 Analytics Accelerator for z/OS

IBM Db2 Analytics Accelerator is a high-performance component that is tightly integrated with IBM Db2 for z/OS. It delivers high-speed processing for complex Db2 for z/OS queries to support business-critical transactional, reporting, analytic, and artificial intelligence (AI) workloads.

This chapter provides an overview of key functions and involved table types. It describes the key function enhancements to Db2 Analytics Accelerator for z/OS 7.5 and the subsequent maintenance levels.

This chapter describes the following topics:

- ▶ Query acceleration enhancements
- ▶ Accelerator-only table enhancements
- ▶ Integrated Synchronization enhancements
- ▶ Db2 Analytics Accelerator administration enhancements

13.1 Overview of Db2 Analytics Accelerator

Db2 Analytics Accelerator transforms IBM Z or a mainframe into a hybrid transactional analytical processing (HTAP) environment. It reduces cost and complexity and enables analytics on transactional data as that data is generated to provide real-time insights to support real-time operational decisions. Db2 Analytics Accelerator and Db2 for z/OS form an integrated hybrid environment that can run transaction processing, complex analytical, and reporting workloads concurrently and efficiently so that business-critical data can be leveraged where it originates to integrate real-time insight with real-time operational decisions.

Db2 Analytics Accelerator for z/OS 7 originally offered the following deployment options:

- ▶ Accelerator on IBM Integrated Analytics System (IAS), which consists of a pre-configured hardware and software appliance for easy deployment, management, and high performance.
- ▶ Accelerator on IBM Z, which can be deployed in the following ways:
 - Within an IBM Secure Service Container (SSC) logical partition (LPAR) within IBM Z by using Integrated Facility for Linux (IFL) processors without needing to attach a separate appliance.
 - Within an IBM SSC LPAR, but leveraging IFL processors on IBM LinuxONE, an enterprise-grade Linux server.

The IAS appliance was withdrawn from marketing, so the preferred deployment option for new installations is IBM Z or IBM LinuxONE within an SSC LPAR.

The Db2 Analytics Accelerator on IBM Z provides a hyper-protected runtime environment with IBM Z levels of service. The deep integration with the IBM Z environment leverages system management services that integrate into existing IBM Parallel Sysplex and IBM Globally Dispersed Parallel Sysplex (IBM GDPS) based high availability and disaster recovery (HADR) implementations. This architecture provides resource sharing and the flexibility to adjust resources to support growth and changing requirements.

Db2 Analytics Accelerator provides the following key capabilities:

- ▶ Data synchronization:
 - Loading the data of Db2 tables into corresponding accelerator-shadow tables on the accelerator.
 - Replicating changes in Db2 tables into corresponding accelerator-shadow tables in near real-time through the IBM Integrated Synchronization built-in feature. IBM Integrated Synchronization is an advanced data synchronization technique that promotes data coherency between IBM Db2 for z/OS and the Db2 Analytics Accelerator.
 - Archiving Db2 table partitions of range partitioned tables to the accelerator into corresponding accelerator-archive tables through the High-Performance Storage Saver feature.
- ▶ Query acceleration on accelerator-shadow tables or accelerator-archive tables.
- ▶ Query acceleration on actual (latest committed) replicated data in accelerator-shadow tables through the **WAITFORDATA** feature (this feature provides true HTAP capability).
- ▶ In-database transformation on the accelerator by using AOTs.

- ▶ Query acceleration or multi-step reporting on AOTs.
- ▶ Federation: Query acceleration against accelerator-shadow tables or AOTs that do not belong to or originate from the Db2 subsystem that issues the query.

For more information about these features, see [Db2 Analytics Accelerator for z/OS 7.5](#).

The tables on the accelerator are core to the key capabilities. Db2 Analytics Accelerator uses several distinct types of tables. Understanding the role that each table type plays is critical to getting the most out of your Db2 Analytics Accelerator environment.

Figure 13-1 shows the characteristics of these table types. Details about each type are explained in the following sections.

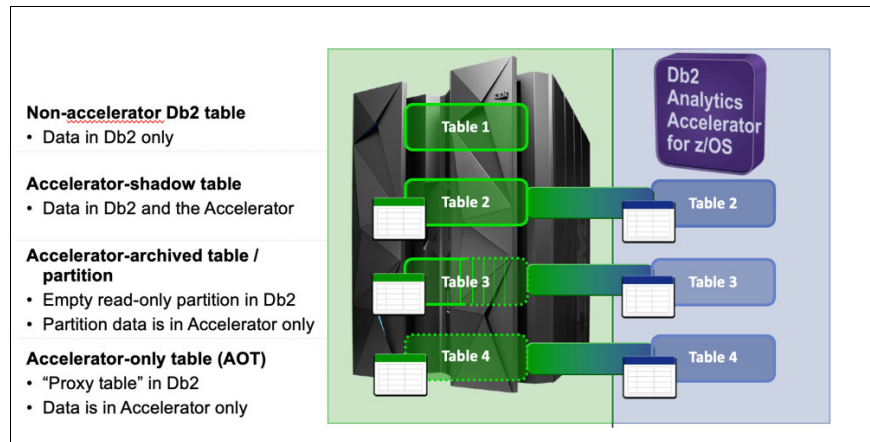


Figure 13-1 Accelerator table types

13.1.1 Non-accelerator Db2 tables

A non-accelerator Db2 table and its data exist in Db2 only. Queries that are run against that data are always run in Db2. For more information and a summary of the different Db2 table types, see [Types of tables](#).

13.1.2 Accelerator-shadow tables

An *accelerator-shadow table* is a table on the accelerator that is a copy of a Db2 table. The accelerator-shadow table contains all or a subset of the Db2 table columns, for example, columns with column types that are not supported by Db2 Analytics Accelerator are omitted.

After the accelerator-shadow table is defined on the accelerator, you can load data into the table by copying data from the original Db2 table. Alternatively, you can use IBM Integrated Synchronization, which is a replication method that is built into Db2 Analytics Accelerator, to replicate changed data continually from the Db2 table to the accelerator-shadow table in near-real time with low latency.

You can also use the Db2 Analytics Accelerator Loader to load external data, for example, from 9 files, into both Db2 for z/OS and the accelerator concurrently. Then, the same data exists in the Db2 table and the accelerator-shadow table. With the **HALOAD** utility, you can load Db2 data or external data into multiple accelerators concurrently while reducing CPU consumption and ensuring data consistency.

The **QUERY ACCELERATION** setting determines whether queries that use these tables are routed to the accelerator to run on the accelerator-shadow tables or stay on Db2 to run on the original Db2 tables. The value must be set to **ELIGIBLE**, **ENABLE**, **ENABLE WITH FAILBACK**, or **ALL**. You can set the value by using any of the following methods:

- ▶ The **QUERY_ACCELERATION** subsystem parameter
- ▶ The **CURRENT QUERY ACCELERATION** special register
- ▶ The **QUERYACCELERATION** bind option
- ▶ Connection properties for applications connecting by using Java Database Connectivity (JDBC) or Microsoft Open Database Connectivity (ODBC)
- ▶ Db2 profile tables for remote applications

Read-only queries are routed to the accelerator to run on accelerator-shadow tables. Data Manipulation Language (DML) statements cannot be run on accelerator-shadow tables. If you use **INSERT** from **SELECT** statements, the **SELECT** part can be routed to the accelerator and can run on accelerator-shadow tables if **ZPARM QUERY_ACCEL_OPTION** is set to option 2. Otherwise, the **SELECT** part is run in Db2. The **INSERT** part is always run in Db2.

13.1.3 Accelerator-archived tables and partitions

An *accelerator-archived table* is a table on the accelerator for which partitions of a range-partitioned Db2 table, or the entire range-partitioned table, were moved (archived) to the accelerator by using the High-Performance Storage Save function. Accelerator-archived tables are used primarily for historical data that is no longer actively used or maintained.

When the data is archived on the accelerator, the original Db2 partitions are emptied to save storage space on IBM Z. The table spaces of Db2 table partitions that are associated with an accelerator-archived table are set to a persistent read-only state so that the original Db2 table can no longer be changed. An image copy of the data is also created as part of the archive process to enable restoring the data in Db2. Archiving to multiple accelerators is supported.

To include accelerator-archived tables in your queries, set the **GET_ACCEL_ARCHIVE** special register, subsystem parameter, or **BIND** option to **YES**, in addition to setting the **QUERY ACCELERATION** setting to a value other than **NONE**.

13.1.4 Accelerator-only tables

An *accelerator-only table* (AOT) is a table that stores rows only in the accelerator, not in Db2. The table and column definition is contained in Db2 catalog tables. The **SYSIBM.SYSTABLES** Db2 catalog table reflects an AOT by setting **TYPE** to **D** (definition-only). The **SYSACCEL.SYSACCELERATEDTABLES** table contains information about which accelerator that the AOT is in.

To create an AOT, issue the **CREATE TABLE** statement in Db2 with the **IN ACCELERATOR <name>** clause, where *<name>* determines the name of the accelerator.

To drop an AOT, issue the **DROP TABLE** statement. To create or drop an AOT, the accelerator must be started, but the **QUERY ACCELERATION** setting can have any value.

The **QUERY ACCELERATION** setting must be set to **ENABLE**, **ELIGIBLE**, or **ALL** to enable execution of queries or DML statements on AOTs.

Any queries that reference AOTs must be run on the accelerator. These queries can also join other tables on the accelerator: either accelerator-shadow tables, accelerator-archived tables, or other AOTs.

However, a query that references an AOT cannot join a non-accelerator Db2 table. If a query that references an AOT is not eligible for query acceleration, or if the **QUERY ACCELERATION** setting is not set correctly, an SQL error -4742 is issued.

You can use DML statements to change the contents of an AOT. A DML statement, for example, an **INSERT** from a **SELECT** statement, can reference other tables on the accelerator: either accelerator-shadow tables, accelerator-archived tables, or other AOTs. However, this DML statement cannot reference a non-accelerator Db2 table.

You also can use the Db2 Analytics Accelerator Loader to load external data, for example, from a VSAM file, into an AOT.

AOTs are different from temporary tables. For example, the Db2 catalog does not store a description of a declared temporary table. Therefore, the description and the instance of the table are not persistent. Multiple application processes can refer to the same declared temporary table by name, but they do not share a description or instance of the table.

13.2 Query acceleration enhancements

The query acceleration enhancements that this section describes were introduced with Db2 12 APARs or function levels and are now available natively in Db2 13. With the enhancements in Db2 and Db2 Analytics Accelerator, you can accelerate your queries more than ever before. Consider using the enhanced functions in the following use cases:

- ▶ Enhancing Db2 for z/OS native SQL capabilities by using pass-through only expression support for analytical built-in functions that are available only on the accelerator
- ▶ Using the accelerator for applications that require pagination of result sets
- ▶ Using the accelerator for data-sensitive uses cases that require masked columns
- ▶ Ensuring that certain queries are always routed or not routed to the accelerator by using dynamic plan stability

About query acceleration: Db2 for z/OS considers routing a query to the accelerator if the **QUERY ACCELERATION** setting is set to any value other than NONE. In addition to the **QUERY ACCELERATION** setting, Db2 for z/OS also considers heuristics such as table size and estimated cost. Although most Db2 for z/OS queries are eligible for query acceleration, the following limitations cause a query to be ineligible for acceleration:

- ▶ A query that uses columns with unsupported data types such as XML or large object (LOB).
- ▶ A query that uses a user-defined function (UDF) other than inline SQL scalar UDF or compiled SQL scalar UDF.
- ▶ A query that joins tables with different encoding schemas.
- ▶ A query that uses an unsupported built-in function, for example, ACOS, ASIN, CLOB, or BLOB.
- ▶ A query that is not read-only.

13.2.1 Pass-through only expression support

Pass-through only expression support for built-in functions in Db2 for z/OS enhances Db2 native SQL capabilities if an accelerator is connected to Db2. Pass-through only expression support means that you can use built-in-functions in your SQL queries that are provided by the accelerator's database engine but not provided by Db2 for z/OS.

The accelerator's database engine (IBM Db2 Warehouse) offers a range of built-in analytic functions that are not supported natively by Db2 for z/OS., which include aggregate functions, online analytics processing (OLAP) functions, and scalar functions.

A subset of these types of functions can be used in SQL queries that are routed to the accelerator with built-in-function pass-through only expression support. Db2 for z/OS is "aware" of the accelerator when it parses the **SELECT** statement. If the **SELECT** statement references a built-in function that is available only on the accelerator, the Db2 for z/OS parser validates the signature and allows its invocation within the rewritten SQL.

This capability enhances the analytic SQL capabilities of Db2 for z/OS with an accelerator that is connected by allowing you to run more analytical workloads on your Db2 for z/OS data. For example:

- ▶ You can use regression functions for statistical analysis of the relationship among variables.
- ▶ You can use OLAP functions to return ranking, row numbering, and existing aggregate function information as a scalar value in a query result.

Db2 12 FL504 introduced pass-through support for the following built-in functions:

- ▶ OLAP and aggregate functions: **CUME_DIST**, **FIRST_VALUE**, **LAG**, **LAST_VALUE**, **LEAD**, **NTH_VALUE**, **NTILE**, **PERCENT_RANK**, and **RATIO_TO_REPORT**
- ▶ Scalar functions: **REGEXP_COUNT**, **REGEXP_INSTR**, **REGEXP_LIKE**, **REGEXP_REPLACE**, and **REGEXP_SUBSTR**

Db2 12 FL507 introduced pass-through support for the following built-in functions:

- ▶ Aggregate functions (all regression functions): **REGR_SLOPE**, **REGR_INTERCEPT**, **REGR_ICPT**, **REGR_R2**, **REGR_COUNT**, **REGR_AVGX**, **REGR_AVGY**, **REGR_SXX**, **REGR_SYY**, and **REGR_SXY**
- ▶ Scalar functions: **ADD_DAYS**, **DAYS_BETWEEN**, **NEXT_MONTH**, **BTRIM**, and **ROUND_TIMESTAMP** (if invoked with a data expression)

13.2.2 SQL pagination support

Pagination is commonly used by web applications. Some of these applications require pagination support to divide a larger result set into smaller discrete pages for display purposes. With Db2 12 APAR PH42015 and Db2 13, pagination is supported for queries that are routed to the accelerator.

The following two kinds of queries with pagination are eligible to run on the accelerator:

- ▶ Queries that include pagination (**OFFSET N ROWS** clause), for example:

```
SELECT * FROM T1
ORDER BY C1
OFFSET 10 ROWS;
```


- Queries that include a **FETCH FIRST N ROWS** clause and the N is a parameter marker or a host variable, for example:

```
SELECT * FROM T1
ORDER BY C1
FETCH FIRST ? ROWS;
```

13.2.3 Column mask support

A *column mask* is a Db2 for z/OS feature that describes a specific column access control rule for a column. In the form of an SQL **CASE** expression, the rule specifies the condition under which a user, group, or role can receive the masked values that are returned for a column.

Routing support for a query that references a column with a column mask is provided with Db2 12 for z/OS APAR PH33061 or Db2 13 and Db2 Analytics Accelerator for z/OS 7.5.6.

Tables that have masked columns are now fully added, loaded, or replicated to the accelerator.

Db2 Analytics Accelerator for z/OS 7.5.6 required. Before Db2 Analytics Accelerator for z/OS 7.5.6, the masked columns were omitted and only the non-masked columns were added, loaded, or replicated to the accelerator.

Queries that reference the masked columns are routed to the accelerator if the masked columns are not part of the outermost result **SELECT** list that is being returned to the application. In this case, the column mask does not need to be applied.

In the following example, two tables are defined (T1 and T2). Both tables have column C2 defined as a masked column, and columns C1 and C3 as non-masked columns.

```
TABLE T1(C1, C2, C3) column mask on C2
TABLE T2(C1, C2, C3) column mask on C2
```

The following query can be routed to the accelerator because the masked column C2 is used only to join the tables:

```
SELECT A.C1, B.C3
FROM T1 A,
     T2 B
WHERE A.C2 = B.C2;
```

The following query cannot be routed to the accelerator because the masked column C2 is returned as part of the outermost result set:

```
SELECT A.C1, B.C2
FROM T1 A,
     T2 B
WHERE A.C2 = B.C2;
```

The query fails with **SQLCODE -4742 rsn 15** if **CURRENT QUERY ACCELERATION ALL** is set; otherwise, it runs in Db2 for z/OS.

13.2.4 Dynamic plan stability

You can use the Db2 for z/OS dynamic plan stability feature to stabilize the Db2 optimizer's decision to route a query to the accelerator.

This feature can be useful if you use **CURRENT QUERY ACCELERATION ENABLE** but want to ensure that specific queries are always routed to the accelerator. **ENABLE** means that the Db2 optimizer considers cost estimates and other heuristics to decide whether to route a query to the accelerator. For example, cost estimates might change if the table statistics are not up to date. As a result, the Db2 optimizer could select a different access path at preparation time, which causes the query to run in Db2 for z/OS instead of on the accelerator.

Conversely, you can use the dynamic plan stability feature to stabilize the decision by the Db2 for z/OS optimizer to not route a query to the accelerator, but always run it on Db2 for z/OS. This feature protects the accelerator against unexpected Db2 for z/OS cost estimation changes (due to upgrading Db2 for z/OS maintenance), which might cause queries that had been run on Db2 before to be routed to the accelerator instead.

The following steps summarize the process of implementing the “never accelerate” behavior by using the dynamic plan stability feature:

1. Monitor the dynamic statement cache by using **EXPLAIN** and other tools.
2. Identify queries that have the wanted access paths and acceleration behaviors (for example, **ACCELERATED = NEVER**) and that must remain stable across Db2 for z/OS PTF maintenance.
3. Stabilize identified queries into **SYSDYNQRY** while they are in the dynamic statement cache.
4. Apply Db2 for z/OS PTF maintenance as usual.

After Db2 for z/OS is restarted, stabilized queries are loaded from **SYSDYNQRY** into the dynamic statement cache and are run from there. Loading from **SYSDYNQRY** does not result in a full prepare, so access paths and acceleration behaviors are preserved even if Db2 for z/OS PTFs changes the access paths for these queries.

13.3 Accelerator-only table enhancements

AOTs are used to store the intermediate or final results of data transformation or reporting processes. AOTs also can be used to accelerate data transformations that are implemented by using SQL statements.

AOTs provide the following key benefits and capabilities:

- ▶ They accelerate in-database data transformations and data movement processes.
- ▶ They can reduce the need of data movement processes to other platforms for data transformation purposes.
- ▶ They enable multi-step reporting on the accelerator.
- ▶ They can save disk space and CPU cost on IBM Z that is used for transformations and reporting steps.
- ▶ They allow data preparation steps for machine learning and other advanced analytics to run on the accelerator.
- ▶ They can be used to store a set of data in Db2 Analytics Accelerator only, not on Db2 for z/OS, without using the High-Performance Storage Saver function.

The following sections describe new functions for AOTs that enable them to be used in HA and workload-balancing scenarios with best performance. These functions were originally introduced with Db2 12 PTFs or with a Db2 Analytics Accelerator 7.5 maintenance level. They are part of Db2 13 natively without an extra PTF.

- ▶ HA support
- ▶ Distribution key support
- ▶ Explicit statistics collection
- ▶ Consideration for migrating AOTs from Db2 Analytics Accelerator 5 to 7.5

13.3.1 High availability support for AOTs

IBM Z users expect that all components that contribute to the IBM Z platform support the same operational characteristics regarding continuous availability and low cost of operation, which are significant factors in the total cost of ownership (TCO).

Before Db2 12 FL509, AOTs did not support a HA setup with an accelerator group that consisted of more than one accelerator. For example, if multiple accelerators were connected to the same Db2 subsystem for HA or workload balancing purposes, an AOT could be created on one accelerator only. This restriction meant that an AOT did not take part in HA or workload balancing scenarios. Db2 for z/OS always routed queries on an AOT to the same accelerator. If this accelerator was down for any reason, then the AOT could not be accessed unless a backup was taken and restored on an available accelerator.

Db2 12 FL509 APAR PH30574 and Db2 13 removed this restriction. You can now create AOTs on multiple accelerators that are connected to the same Db2 subsystem to support HA and workload balancing.

To use multiple AOTs with the same name, you define an accelerator group that consists of one or more accelerators, and then you create the AOT in the accelerator group.

Usage notes:

- ▶ **INSERT**, **UPDATE**, and **DELETE** operations on the AOTs must be run on single accelerator of the group by using the **CURRENT ACCELERATOR** special register (and then repeated on the others).
- ▶ Queries that reference the AOTs are routed to one of the accelerators within the group according to the existing workload balancing algorithm.
- ▶ Db2 Analytics Accelerator Loader supports High Availability Load (HALOAD) into all AOTs within the accelerator group.

Creating an AOT in multiple accelerators by using an accelerator group

The following example statements show how to define an accelerator group and how to create an AOT in that group.

To create an accelerator group (ACCLGRP) that contains two accelerators (ACCEL1 and ACCEL2), issue the following example **INSERT** statement into the SYSIBM.LOCATION table:

```
INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, DBALIAS)
VALUES ('ACCLGRP', 'DSNACCELERATORALIAS', 'ACCEL1 ACCEL2')
```

To create an AOT in both accelerators ACCEL1 and ACCEL2, specify the accelerator group (ACCLGRP) in the **IN ACCELERATOR** clause, for example:

```
CREATE MYAOT.T1 (<columnspec>) IN ACCELERATOR ACCLGRP
```

If Db2 for z/OS does not find ACCLGRP as a physical accelerator name during the **CREATE** process, it looks for it as a logical accelerator name.

This **CREATE** statement has the following results:

- ▶ An AOT with name MYAOT.T1 is created on both accelerators, ACCEL1 and ACCEL2.
- ▶ A single entry is created in SYSIBM.SYSTABLES.
- ▶ Two entries are created in SYSACCEL.SYSACCELERATEDTABLES, one for each accelerator that is represented by the accelerator group ACCLGRP.

Queries that reference MYAOT.T1 are routed to an accelerator that is defined in ACCLGRP according to existing workload balancing algorithms.

Db2 catalog entries for an AOT that is defined on multiple accelerators

Figure 13-2 illustrates the Db2 for z/OS catalog entries that are created as a result of running the example **CREATE** statement in the previous section.

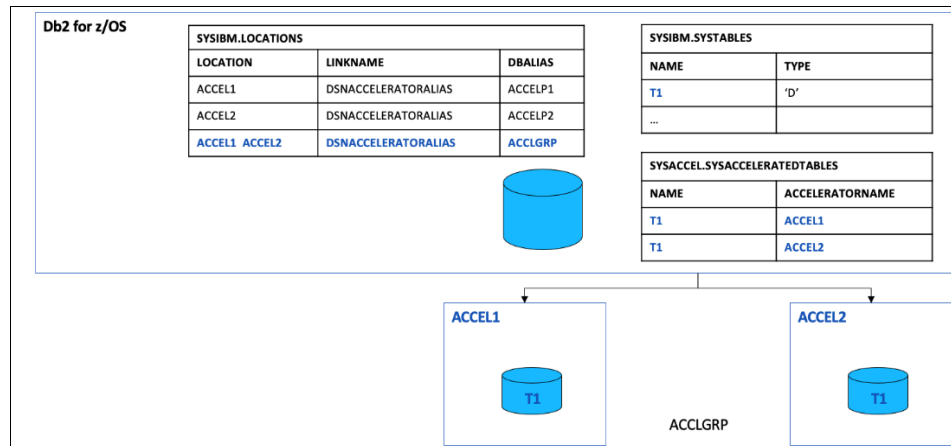


Figure 13-2 Db2 z/OS catalog entries for an AOT in multiple accelerators

The AOT with name T1 is created on the accelerators ACCEL1 and ACCEL2. This information persists in tables SYSACCEL.SYSACCELERATEDTABLES and SYSIBM.SYSTABLES.

ACCEL1 and ACCEL2 are defined in accelerator group ACCLGRP. This information persists in the SYSIBM.LOCATIONS table.

DML statements on AOTs that are defined on multiple accelerators

DML statements, such as **INSERT**, **UPDATE**, or **DELETE** statements, must be performed separately for each accelerator, as shown in the following example:

```
SET CURRENT ACCELERATOR = ACCEL1;
INSERT INTO MYAOT.T1 VALUES (1);
COMMIT;
SET CURRENT ACCELERATOR = ACCEL2;
INSERT INTO MYAOT.T1 VALUES (1);
COMMIT;
```

Alternatively, you can use the Db2 Analytics Accelerator Loader to load data into AOTs. By using its HALOAD function, you can load data into the AOTs on multiple accelerators in parallel.

13.3.2 Distribution key support for AOTs

A *distribution key* determines how tables rows are distributed among the available working nodes on the accelerator. Distribution keys are set by default on accelerator-shadow and accelerator-archived tables to allow colocated joins. If one or more enforced unique constraints are defined on a Db2 for z/OS table, the constraint with fewest number of columns is taken as distribution key when the table is added to the accelerator. If no unique constraint is available on the Db2 table, then the table rows are distributed randomly among the working nodes on the accelerator. Random distribution almost always forces repartitioning and redistribution on the join columns during query execution, which often results in query performance penalties and performance problems.

Distribution keys are now available for AOTs. Distribution keys can help to improve query performance for queries that reference AOTs. This feature requires Db2 12 for z/OS APAR PH30659, or Db2 13.

As for accelerator-shadow or accelerator-archive tables, you cannot set or maintain distribution keys for AOTs by using the Alter Keys function in Db2 Analytics Accelerator Studio or by running the `SYSPROC.ACCEL_ALTER_TABLES` stored procedure. Instead, you must specify distribution keys as part of the `CREATE TABLE` statement for an AOT, for example:

```
CREATE TABLE AOTWITHKEY (C1 INTEGER, C2 INTEGER, C3 INTEGER) IN ACCELERATOR ACCEL1
-- <ACCOPTS> <distributionKey> (C1,C2) </distributionKey> </ACCOPTS>
```

- ▶ `<ACCOPTS>` is an XML element that encloses more options for the accelerator.
- ▶ `<distributionKey>` is an XML element for specifying a distribution key.
- ▶ `(C1,C2)` are the table columns that make up the distribution key.

To create an AOT with a distribution key from IBM Db2 Data Studio by using a `CREATE TABLE` statement, you need Data Studio 4.1.4 APAR 2, which includes the Data Studio APAR=IT39577. Otherwise, Data Studio ignores the distribution key setting within the SQL comment.

For example, after you run the following statements from a Data Studio SQL Editor, the AOT and its distribution key are displayed in the accelerator tables view within Data Studio, as shown in Figure 13-3.

```
SET CURRENT QUERY ACCELERATION ENABLE;
--#SET SQLFORMAT SQLCOMNT
CREATE TABLE AOTWITHKEY (COL01 CHAR(8), COL02 CHAR(8), COL03      INTEGER) IN
ACCELERATOR ACCEL1
-- <ACCOPTS> <distributionkey> (COL01) </distributionkey>
</ACCOPTS>;
```

The screenshot shows the 'Tables (1 of 1 loaded / 1 of 1 enabled for acceleration)' view in Data Studio. It includes a toolbar with buttons for 'Add...', 'Alter Keys...', 'Remove', 'Load...', 'Acceleration', 'Storage Saver', and 'List Tasks'. Below the toolbar is a search bar with 'Name like: type filter text'. The main table displays the following data:

Name	Size	Acceleration	Last Load	Distribution Key
<ul style="list-style-type: none"> ▼ SYSADM AOTWITHKEY 	1 MB	1 of 1 Enabled	1 of 1 tables AOT ready	- COL01

Figure 13-3 Accelerator tables view in Data Studio displaying an AOT with distribution key

Note: The procedure for setting distribution keys cannot be used to set organizing keys. Organizing keys are not supported for AOTs currently.

13.3.3 Explicit statistics collection on AOTs

Existing data transformation or ETL processes might insert, update, or delete much data in AOTs. Before you reference those AOTs in subsequent queries, you should explicitly request the collection of statistics (**RUNSTATS**) to ensure the best performance of these queries.

You can collect statistics by calling the **SYSPROC.ACCEL_COLLECT_TABLE_STATISTICS** stored procedure on the specified set of tables. This stored procedure works synchronously and does not return results to the caller until all specified tables are processed. This behavior allows you to include the stored procedure in data transformation or ETL batch processes.

The following list summarizes the steps to request statistic collection as part of a batch process:

1. Insert rows into AOT A.
2. Collect statistics on AOT A by calling the **ACCEL_COLLECT_TABLE_STATISTICS** stored procedure.
3. Run a second **INSERT** statement that selects rows in AOT A and inserts these rows into AOT B.

Restriction: If a Db2 subsystem is paired to an accelerator for which CDC replication is enabled, **SYSPROC.ACCEL_COLLECT_TABLE_STATISTICS** does not run and exits with message AQT20139E. As a result, if your accelerator is paired with more than one Db2 subsystem, the stored procedure also does not work for tables from the other Db2 subsystems, regardless of whether these subsystems are CDC-enabled.

The following example shows the syntax of the **SYSPROC.ACCEL_COLLECT_TABLE_STATISTICS** stored procedure:

```
CALL SYSPROC.ACCEL_COLLECT_TABLE_STATISTICS
(accelerator_name,
 table_statistics_specification,
 message);
```

The following example shows a `table_statistics_specification` input XML string to collect statistics for the **NATION** and **REGION** tables:

```
<?xml version="1.0" encoding="UTF-8"?>
<dwa:tableSetForStatisticsCollection
  xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011"          version="1.0">
  <table name="NATION" schema="TPCH"/>
  <table name="REGION" schema="TPCH"/>
</dwa:tableSetForStatisticsCollection>
```

For more information, see [SYSPROC.ACCEL_COLLECT_TABLE_STATISTICS](#).

13.3.4 Migration considerations from Db2 Analytics Accelerator 5 to 7

AOTs that exist on Db2 Analytics Accelerator 5 must be re-created and populated on Db2 Analytics Accelerator V7. If a mixture of V5 and V7 accelerators are connected to one Db2 subsystem, then an AOT with a certain name can exist only on one of these accelerators. The HA feature for AOTs that is described in 13.3.1, “High availability support for AOTs” on page 209 is available only for multiple V7 accelerators, but not for a mixture of V5 and V7 accelerators.

Several methods are available for re-creating AOTs in V7:

- ▶ If AOTs are created as part of in-database transformation processes, then establish these processes on Db2 Analytic Accelerator 7 too.
- ▶ If AOTs are created and loaded by using Db2 Analytics Accelerator Loader, then use the same process for Db2 Analytics Accelerator 7.
- ▶ If AOTs cannot be created and populated on Db2 Analytics Accelerator 7 from the original data sources, you can back up AOTs from Db2 Analytics Accelerator 5 to z/OS or to Db2 and then restore them to a V7 accelerator by using one of the following methods:
 - Use Db2 Analytics Accelerator Loader 2.1 to back up AOTs to data sets on z/OS in Db2 internal format and then restore the data to AOTs on Accelerator 7.
 - If you do not use Db2 Analytics Accelerator Loader, then back up AOTs from Db2 Analytics Accelerator 5 to Db2 for z/OS by using one of the following methods:
 - Issue a **CREATE TABLE** statement in Db2 for z/OS, then issue an **INSERT into Db2 table SELECT FROM AOT** statement with the **ACCEL_QUERY_OPTIONS** subsystem parameter set to option 2.
 - Issue a **CREATE TABLE** statement in Db2 for z/OS and use the Db2 cross-loader function to populate the table from the AOT.

After the backup is complete, add and load the Db2 table to the V7 accelerator as an accelerator-shadow table. Then, create an AOT from the accelerator-shadow table to remove the accelerator-shadow table. Finally, drop the table in Db2.

13.4 Integrated Synchronization enhancements

Integrated Synchronization is a built-in product feature that you use to set up incremental updates, and it is part of Db2 Analytics Accelerator. The incremental update function of Db2 Analytics Accelerator updates accelerator-shadow tables continuously. Changes to the data in original Db2 for z/OS tables are read from the Db2 for z/OS log and replicated to the corresponding target tables with a high frequency and a brief delay. Query results from an accelerator are always extracted from recent, near real-time data.

The previous technology for replicating data between Db2 for z/OS and Db2 Analytics Accelerator was IBM Change Data Capture (CDC). CDC was a separate product that required individual installation and setup. CDC is deprecated and is no longer used with accelerator deployments.

IBM Integrated Synchronization provides the following significant advantages compared to CDC:

- ▶ Administration, packaging, upgrades, and support are simplified. These items are managed as part of the Db2 for z/OS maintenance stream.
- ▶ Updates are processed quickly.

- ▶ CPU consumption on IBM Z is reduced due to a streamlined, optimized design in which most of the processing is done on the accelerator. This situation provides reduced latency.
- ▶ It uses IBM Z Integrated Information Processor (zIIP) on Db2 for z/OS, which leads to reduced CPU costs on IBM Z and better overall performance data, such as throughput and synchronized rows per second.
- ▶ On z/OS, the workload to capture the table changes was reduced, and the remainder can be handled by zIIPs.
- ▶ With the introduction of an enterprise-grade HTAP, which is also known as the **WAITFORDATA** feature, Integrated Synchronization is enabled to support more analytical queries running against the latest committed data.

The following sections describe the features, enhancements, and capabilities of Integrated Synchronization in the context of Db2 for z/OS, which are added to Integrated Synchronization with Db2 12 APARs. However, they are all natively available with Db2 13.

For more information about Integrated Synchronization, see *IBM Integrated Synchronization: Incremental Updates Unleashed*, REDP-5616. Although this Redpaper refers to multiple Db2 for z/OS prerequisites, such as FL500 and certain PTFs, all these prerequisites are part of Db2 13 natively, so no additional PTFs are required.

13.4.1 Online schema change (Add/Alter column) support

Schema change (Add/Alter column) support for *replicated* tables was introduced with Db2 Analytics Accelerator 7.5.6.

With this enhancement, schema changes are supported in the sense that you do not need to remove, redefine, re-enable replication for, and reload an accelerator-shadow table after a schema change to an original Db2 for z/OS table. The schema change is detected by the IBM Integrated Synchronization function and, if necessary, a modification of the target accelerator-shadow table is initiated so that incremental updates can continue.

Add/Alter column operations on Db2 for z/OS tables that are enabled for replication are automatically synchronized to the accelerator. You no longer need to remove, re-add, and reload a table to the accelerator to make the schema change available.

The following schema changes are supported for replicated tables:

- ▶ ALTER TABLE <tablename> ADD COLUMN <colname> [type] NULL
- ▶ ALTER TABLE <tablename> ADD COLUMN <colname> [type] WITH DEFAULT [constant]
- ▶ ALTER TABLE <tablename> ALTER COLUMN <colname> SET DATA TYPE [extend length of current type] (VARCHAR and VARGRAPHIC types only)
- ▶ ALTER TABLE <tablename> ALTER COLUMN <colname> SET WITH DEFAULT [constant]

Schema change (Add/Alter column) support for *non-replicated* tables was initially introduced with Db2 Analytics Accelerator 7.5.7 as a technology preview.

With this enhancement, Add/Alter column operations on Db2 for z/OS tables that are not enabled for replication are now automatically synchronized to the accelerator if Integrated Synchronization is enabled for the Db2 system. You no longer need to remove, re-add, and reload a table to the accelerator to make the schema change available.

The supported schema changes for non-replicated tables are the same as for replicated tables.

The following items are required for schema change support:

- ▶ Integrated Synchronization must be enabled for the Db2 system.
- ▶ Db2 12 for z/OS APAR PH35389 - PTF UI76324 must be installed (see [APAR PH35389](#)). Db2 13 contains this support natively.
- ▶ If the schema change affects columns of type `TIMESTAMP`, Db2 12 for z/OS [APAR PH31772](#) is required. Db2 13 contains this support natively.
- ▶ The Db2 for z/OS user that was specified during enablement of Integrated Synchronization needs the following extra privileges:
 - `SELECT` privilege on `SYSIBM.SYSTABLES`, `SYSIBM.SYSCOLUMNS`
 - `SELECT` and `UPDATE` privileges on `SYSIBM.SYSACCELERATEDTABLES`

Regarding the advisory REORG pending (AREO) state and Integrated Synchronization:

- ▶ If replication is enabled for a table and the table is set to the AREO state, replication continues.
- ▶ If replication is disabled for a table and the table is set to the AREO state, perform the REORG first and then enable it for replication.

Certain cases are not supported by online schema change, such as with **ALTER MAXPARTITIONS**.

For example, if you have a simple table space with replication enabled for its table or tables and you issue **ALTER TABLESPACE** with **MAXPARTITIONS** for converting to universal table space (UTS), replication continues and the table space goes into the AREO state. However, after a reorganization, replication is disabled and requires a full reload again.

Because in this scenario you changed the table from a non-partitioned to a partitioned table space, you must reload the table to the accelerator.

13.4.2 ALTER ROTATE support

Before Db2 Analytics Accelerator 7.5.8, after an **ALTER ROTATE PARTITION** (see [Rotating partitions](#)) operation is done on the Db2 source table of a replicated table, the table is suspended from replication. The only alternative is to reload the affected tables to the accelerator. This requirement is a major disruption if you regularly rotate tables, especially if you were using CDC replication before using Integrated Synchronization because CDC replication tolerates this operation.

With **ALTER TABLE ROTATE PARTITION** support, rotate operations on a partitioned Db2 for z/OS table no longer require a reload of the corresponding replication-enabled accelerator shadow-table.

The table changes that are run as part of the **ALTER TABLE ROTATE PARTITION** operation are replicated to the accelerator by Integrated Synchronization processing.

The **ALTER ROTATE** support was initially introduced in Db2 Analytics Accelerator 7.5.8.

13.4.3 DROP TABLE support

After a **DROP TABLE** command runs on a Db2 for z/OS table that is enabled for replication, the corresponding accelerator-shadow table is removed from all accelerators on which it is defined.

Dropping a Db2 for z/OS table triggers a run of the **SYSPROC.ACCEL_REMOVE_TABLE** stored procedure, which removes the accelerator-shadow table from the connected accelerators.

When the stored procedure runs depends on the replication status:

- ▶ If the latency is low, the stored procedure is run almost immediately after the Db2 for z/OS table is dropped.
- ▶ If the latency is high, the stored procedure run is delayed until the log record with the **DROP TABLE** statement is read by the log reader for Integrated Synchronization.
- ▶ If replication is stopped, the stored procedure run is delayed until the replication process resumes.

You are notified of a removal by a corresponding DSNX8811 message in the z/OS syslog or by a similar message in the event viewer of your chosen administration client (Db2 Analytics Accelerator Studio or IBM Data Server Manager).

The **DROP TABLE** support was initially introduced in Db2 Analytics Accelerator 7.5.5.

13.4.4 True HTAP support (the WAITFORDATA feature)

HTAP support is an option that can be used for queries against accelerator-shadow tables that are enabled for replication. It effects a hold on the query execution until the most recent updates (changes committed in Db2 for z/OS) are applied to the tables that are referenced by the query. HTAP support is enabled by the **WAITFORDATA** feature, which can be set by a special register, subsystem parameter, or bind option.

True HTAP (**WAITFORDATA** feature) support provides the following benefits:

- ▶ Data coherency between Db2 for z/OS and Db2 Analytics Accelerator.
- ▶ Application transparency:
 - Latency of committed data is not an issue.
 - Queries are run when the required data is available on the accelerator.
- ▶ Queries on the accelerator use the most recent committed data. For example:
 - In fraud detection scenarios in which CPU-intensive queries run on consistent recent data.
 - In multiform consistency scenarios in which data is entered into a field on a panel and the application commits the data, the data that was entered is shown on subsequent panels and generated reports. This application of HTAP is useful for account data reports and summaries that are required immediately after a bank transaction occurs.

Figure 13-4 on page 217 shows how the **WAITFORDATA** feature works together with Integrated Synchronization.

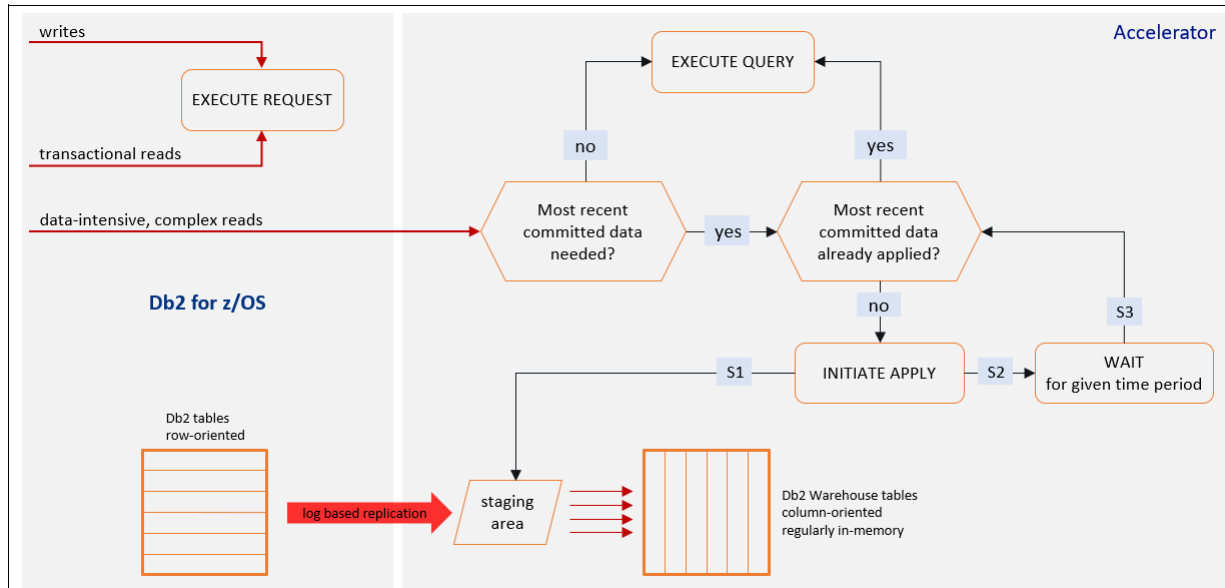


Figure 13-4 True HTAP processing with the WAITFORDATA feature

Row-based Db2 for z/OS table changes are transformed into Db2 Warehouse columnar tables through log-based replication. This step is the Integrated Synchronization processing.

Transactional WRITES and READS are run in Db2 for z/OS. But if a data-intensive complex READ query comes in and the Db2 optimizer routed this query to the accelerator, the accelerator checks to determine whether the most recent committed data is needed based on the **WAITFORDATA** setting.

If you do not care about the most recent committed data, then the query runs on the accelerator without further processing. But if the most recent committed data is needed, the accelerator checks whether the most recent committed data already was applied from the staging area into the columnar tables. In many cases, it already is applied, in which case the query runs.

If the data is not applied for the tables that are involved in the query, the accelerator initiates an apply (writes the data from the staging area into the tables) and waits for a period for this process to happen. When the apply completes, the query runs.

The challenge that is associated with achieving true HTAP is waiting for this period, which ultimately depends on how fast the data can be replicated and applied to the columnar tables.

Integrated Synchronization, which is lightweight, integrated, and fast, drastically reduces this period. Coupled with the **WAITFORDATA** feature, data redundancy and latency are transparent to the application, and queries on the accelerator run consistently on the most recent committed data. This process is how “true HTAP” processing works.

You can use the following mechanisms to specify a **WAITFORDATA** wait time in seconds for queries:

- ▶ By setting the **QUERY_ACCELERATION_WAITFORDATA** subsystem parameter
- ▶ By using the **ACCELERATIONWAITFORDATA BIND** option (for static SQL only)
- ▶ By setting the **CURRENT QUERY ACCELERATION WAITFORDATA** special register for dynamic SQL:
 - If you set **WAITFORDATA** to 0 seconds or do not set it at all, the accelerator runs a query immediately without waiting for committed data.
 - If you set **WAITFORDATA** to a value greater than 0 up to 3600 seconds, the accelerator waits up to the specified number of seconds for committed changes to be applied. If the changes are applied before the wait time is exceeded, the query runs immediately. If the wait time is exceeded before the committed changes are applied, the query either runs or fails depending on the delay expiration rule that you can specify by using the Configuration Console.
 - If you specify **CONTINUE** for the delay expiration rule and the wait time is exceeded, the query runs on the accelerator and returns **SQLCODE +904** to indicate that the query ran but not all committed changes were applied.
 - If you specify **FAIL** (default) for the delay expiration rule and the wait time is exceeded, the query fails with **SQLCODE -904** if the **QUERY ACCELERATION** setting has the value **ELIGIBLE**, **ENABLE**, or **ALL**. If the **QUERY ACCELERATION** setting has the value **ENABLE WITH FALLBACK**, the query runs on Db2 for z/OS instead of failing on the accelerator.

13.4.5 Db2 for z/OS utility support

Support for the **REORG**, **REORG DISCARD**, and **LOAD REPLACE** utilities was initially introduced with Db2 Analytics Accelerator 7.5.1, and it requires Db2 12 APAR PH15002 (UI65898). These features are available in Db2 13 natively.

If data is deleted from a synchronized Db2 for z/OS table or table partition by running **REORG DISCARD** or **LOAD REPLACE DD DUMMY**, the effect on the related accelerator-shadow table (either synchronization or table suspension) varies depending on the type of the Db2 for z/OS table:

- ▶ Synchronization: Only partition-by-range (PBR) or classic partitioned table spaces are supported. The deleted rows are also deleted from the related accelerator-shadow table, and replication remains enabled. A reload is not required.
- ▶ Table suspension: If a **REORG DISCARD** or **LOAD REPLACE DUMMY** is run on a partition-by-growth (PBG) table space, the table is suspended from Integrated Synchronization, query acceleration is disabled, and the table goes into error state. This feature is a security feature. A PBG table does not have fixed partition boundaries; therefore, a **REORG DISCARD** or **LOAD REPLACE DUMMY** operation on partitions of tables in this state does not make sense because the user does not know which rows will be affected. Most likely, not all rows of a partition or table will be deleted by this operation.

Partial discards are not supported by **REORG DISCARD**. Tables with partially emptied partitions will be set to the “Replication Error” status. Tables and partitions in this state must be reloaded. Starting with Db2 Analytics Accelerator 7.5.2 or later, query acceleration is disabled for tables in the Replication Error state.

13.5 Db2 Analytics Accelerator administration enhancements

The Db2 Analytics Accelerator for z/OS engine is a key product in the Db2 for z/OS ecosystem and provides a wealth of capabilities leading up to and including accelerating the running of resource-intensive queries. To make these capabilities accessible and easy to use, it is essential to provide an interface that makes the running experience simple and straightforward.

Most users are familiar with managing and administering Db2 Analytics Accelerator by using products such as the IBM Data Studio thick client or IBM Data Server Manager. Chapter 11, “Development and administration tools” on page 155 introduced the on-going work in the Db2 user experience transformation space. With Db2 Analytics Accelerator a key part of this ecosystem, the management and administration experience is being reimaged.

The goal is to extend the Db2 for z/OS transformation model to API-enable the business logic layer, which enables communication with the Db2 Analytics Accelerator server and decouples it from the user interface implementation. This approach allows for the flexibility of consuming the APIs to perform end-to-end workflows by using a user interface or to build workflows directly by stitching together the APIs into a CI/CD pipeline as needed.

13.5.1 Db2 Analytics Accelerator administration services

Management of Db2 Analytics accelerator involves administration of the objects and queries on the accelerator and the accelerator itself. You can perform many of these activities by using the existing user interfaces or by invoking the stored procedures directly. To have a fail-safe implementation (invocation) of certain capabilities, workflows might include calling one or more stored procedures or Db2 commands in succession. Also, some of the additional information that you might find valuable, such as recommendations for tables to load and recommendations for distribution or organizing keys, is not available with direct invocation of stored procedures.

The goal of Db2 Analytics Accelerator Administration Services is to API-enable unit of work operations with a fail-proof implementation. For example, to report the accurate status of an accelerator, you can call the REST API to fetch the status. The API checks the connectivity status and fetches the status of various accelerator components by calling more stored procedures and Db2 commands and compiling them into a standard JSON output for easy consumption.

API-enabling the functional capabilities provides more flexibility because the APIs can now be invoked from other user interface products to streamline administration tasks or they can be stitched together for a DevOps pipeline to automate management of the accelerator.

For more information about installing and configuring these services, see the [IBM Db2 Analytics Accelerator Administration Services](#) documentation. These services were introduced initially with Db2 Analytics Accelerator 7.5.8.

Db2 Analytics Accelerator Administration Services is an IBM WebSphere Liberty server-based application running on z/OS UNIX System Services on IBM Z. It provides REST APIs to manage the accelerator through Db2 for z/OS.

Figure 13-5 shows all the components in the solution and their interaction patterns. The acceleration administration user experience is browser-based and provided by the Zowe-based IBM Db2 Administration Foundation for z/OS (Admin Foundation).

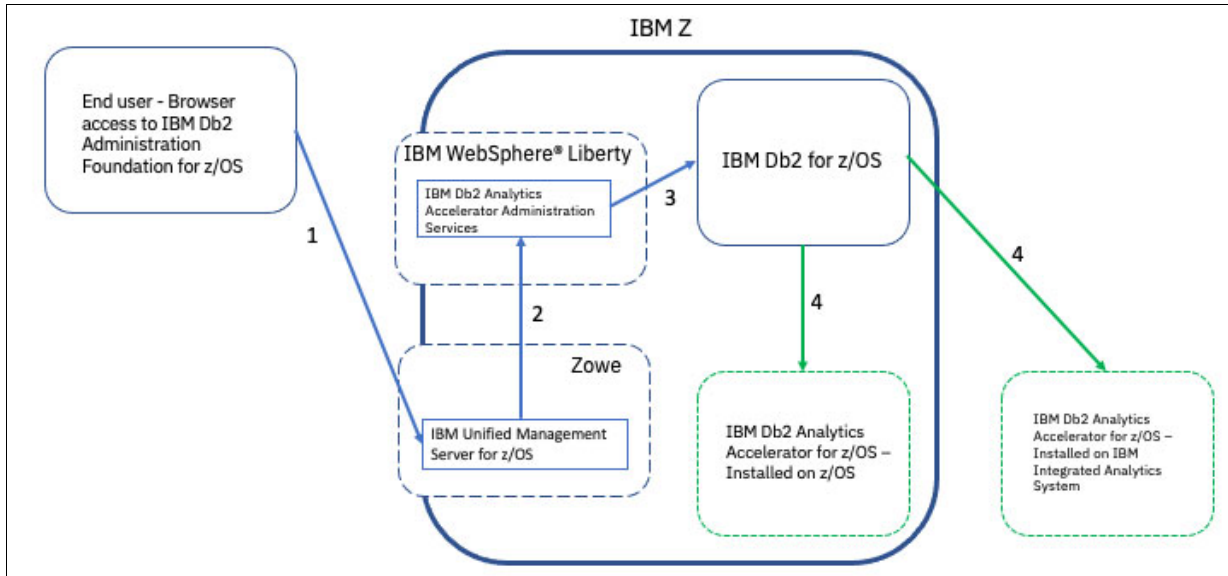


Figure 13-5 Db2 Analytics Accelerator Services ecosystem

13.5.2 IBM Db2 Administration Foundation for z/OS

13.5.1, “Db2 Analytics Accelerator administration services” on page 219 described the API enablement strategy of the accelerator services and how that enablement provides the flexibility to invoke these APIs with ease from other user interfaces. This section illustrates how these APIs are integrated into IBM Db2 Administration Foundation for z/OS (Admin Foundation) to simplify the job of administering Db2 Analytics Accelerator.

Chapter 11, “Development and administration tools” on page 155 introduced Admin Foundation as the strategic platform for the administration of Db2 for z/OS and its ecosystem. The next-generation user experience for Db2 Analytics Accelerator administration is built on Admin Foundation.

Admin Foundation makes it easy to discover and display all the accelerators that are paired to one or more registered Db2 subsystems. Selecting any one of the accelerators from the list displays the accelerator dashboard in the context of the Db2 subsystem with which it is paired, as shown in Figure 13-6 on page 221.

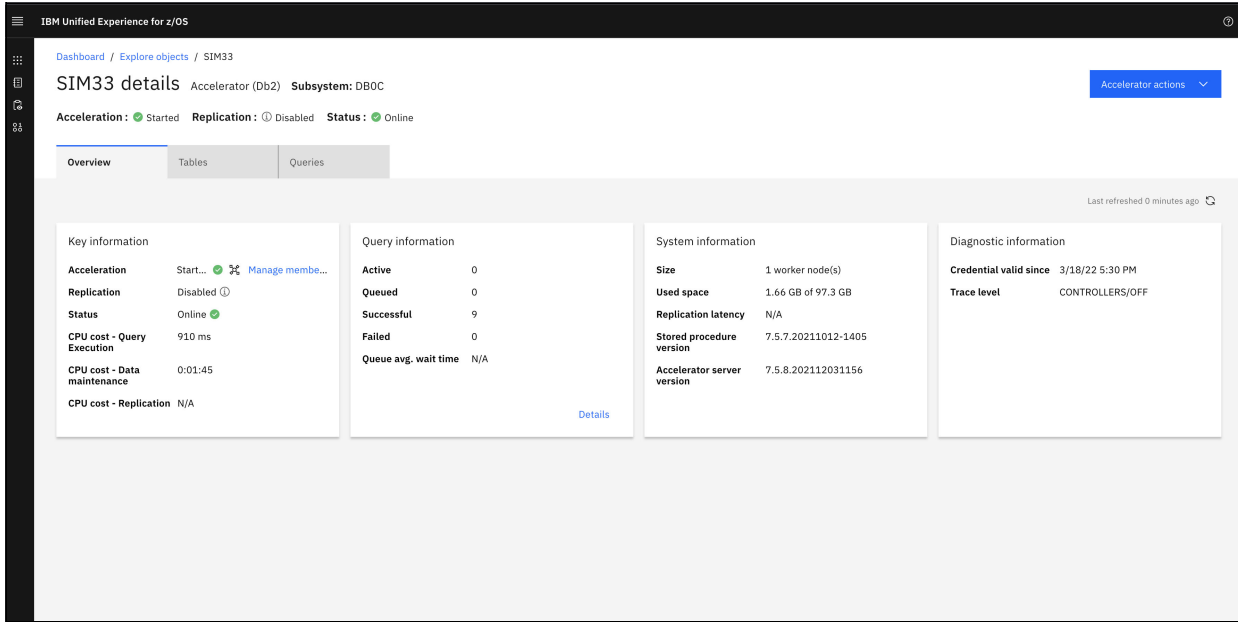


Figure 13-6 Db2 Analytics Accelerator overview dashboard on Db2 Administration Foundation for z/OS

To manage accelerated tables and data, click the **Tables** tab, as shown in Figure 13-7. You can add tables to the accelerators, load table data, enable and disable acceleration and replication, alter distribution and organizing keys, and remove tables from the accelerator.

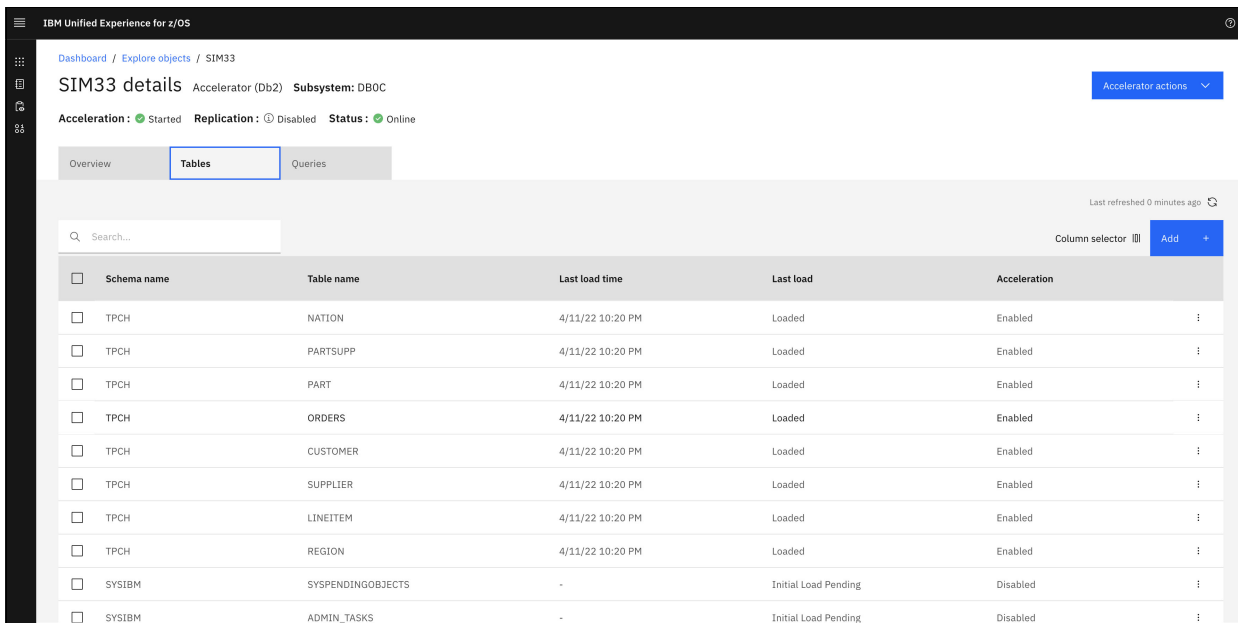


Figure 13-7 Db2 Analytics Accelerator tables dashboard on Db2 Administration Foundation for z/OS

To define filters to capture accelerated queries and display detailed information about each of the resulting queries, click the **Queries** tab, as shown in Figure 13-8. You can take a set of actions for each of the queries.

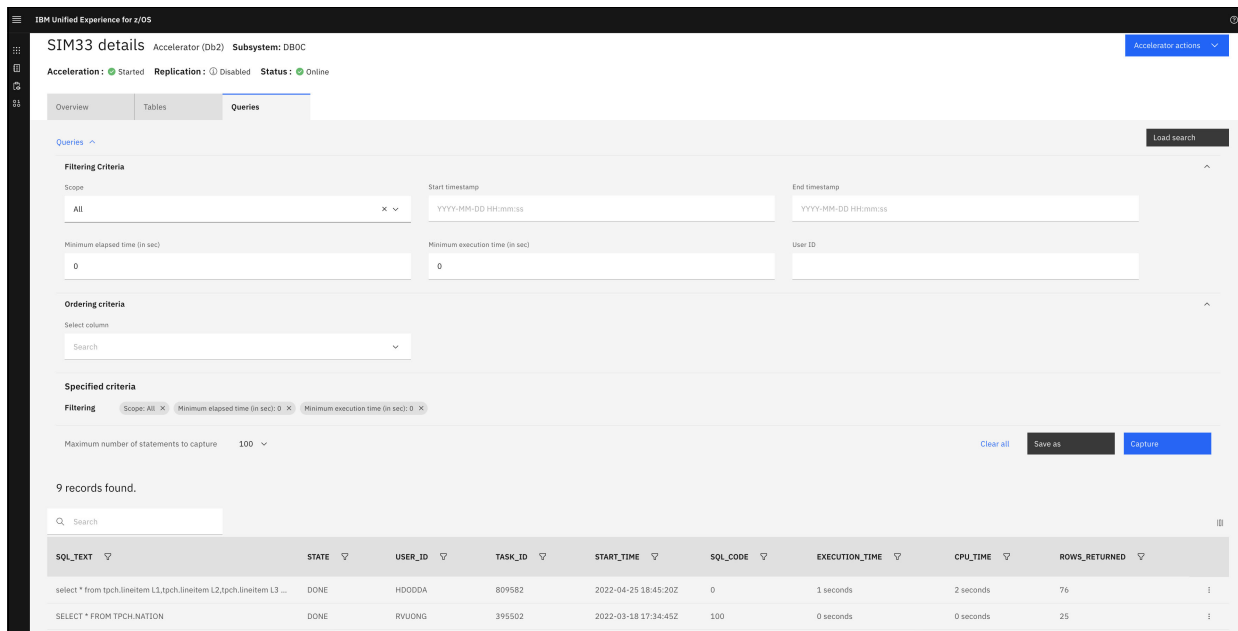


Figure 13-8 Db2 Analytics Accelerator queries dashboard on Db2 Administration Foundation for z/OS

The following Db2 Analytics Accelerator administration capabilities are available through Admin Foundation. User interface support for other capabilities, such as archiving and federation, will be made available in a future release.

- ▶ Query or browse accelerators by using Db2 Catalog Explorer.
- ▶ Manage accelerators by:
 - Starting and stopping acceleration
 - Starting and stopping replication
 - Configuring trace levels and saving traces
 - Removing accelerators from a Db2 subsystem
 - Monitoring system utilization
 - Viewing replication events
- ▶ Manage tables on the accelerator by:
 - Viewing all tables in the accelerator
 - Adding tables
 - Loading table data
 - Removing tables
 - Enabling and disabling acceleration
 - Enabling and disabling replication
- ▶ Manage accelerated queries by:
 - Viewing Full SQL text
 - Capturing query environment and performance data

- Showing Explain details
- Running Visual Explain

13.6 Summary

Db2 Analytics Accelerator is a high-performance component that is tightly integrated with Db2 for z/OS. It provides day-one support for Db2 13, which means that you can use Db2 Analytics Accelerator with Db2 13 and benefit from all provided features.

Because Db2 Analytics Accelerator is based a continuous delivery model, the serviceability, performance, and function features are enhanced regularly. In this chapter, various accelerator enhancements were introduced that were provided by a recent accelerator maintenance level, a Db2 12 PTF, or the combination of both. Db2 13 contains all Db2 enhancements for the accelerator natively.

You can explore the described enhancements to grow your existing accelerator use cases or discover new ones.



IBM Db2 for z/OS Data Gate

IBM Db2 for z/OS Data Gate (Db2 Data Gate) offers a modern hybrid cloud approach for IBM Z customers to deliver data that originates in Db2 for z/OS. By using Db2 Data Gate, your organization can readily support the increasing demand for analytics and the dramatic increase in modern, high-volume, and read-only applications. Data is synchronized between Db2 for z/OS data sources and target databases on IBM Cloud Pak for Data.

About IBM Cloud Pak for Data: IBM Cloud Pak for Data is a fully integrated data and artificial intelligence (AI) platform that modernizes how you collect, organize, and analyze data to infuse data-powered AI throughout your organization. It is a modular platform for running integrated data and AI services. IBM Cloud Pak for Data is composed of integrated microservices that run on a multi-node Red Hat OpenShift cluster, which manages resources elastically and runs with minimal downtime.

IBM Cloud Pak for Data runs on Red Hat OpenShift, which enables a wide range of target platforms. You can run IBM Cloud Pak for Data on the following types of cloud environments:

- ▶ An on-premises, private cloud cluster on Linux on IBM Z or x86 64-bit hardware
- ▶ Any public cloud infrastructure that supports Red Hat OpenShift, including IBM Cloud

Db2 Data Gate provides an end-to-end solution to ensure that data is available and synchronized from sources on IBM Z to targets that are optimized on IBM Cloud Pak for Data. Db2 Data Gate is powered by IBM Integrated Synchronization, which is the same technology that is used in the IBM Db2 Analytics Accelerator for z/OS. This technology includes a log data provider that captures log changes from Db2 for z/OS and sends consolidated, encrypted changes to a log data processor on the target side. The fully zIIP-enabled synchronization protocol is lightweight, high throughput, and low latency. It enables near real-time access to your data without degrading the performance of your core transaction engine.

Db2 Data Gate provides analytics and digital applications with low latency, read-only access to data that originates in Db2 for z/OS. Data is synchronized between Db2 for z/OS data sources and target databases, which can be based either on IBM Db2 in a row store format or on an IBM Db2 Warehouse in a column store. This model makes it suitable for supporting applications that require row-level access and analytical applications that benefit from a column-based data store. So instead of accessing data in the IBM Z data source directly, an application accesses a synchronized copy of the Db2 for z/OS data that is hosted by a separate system. This hierarchy is illustrated in Figure 14-1.

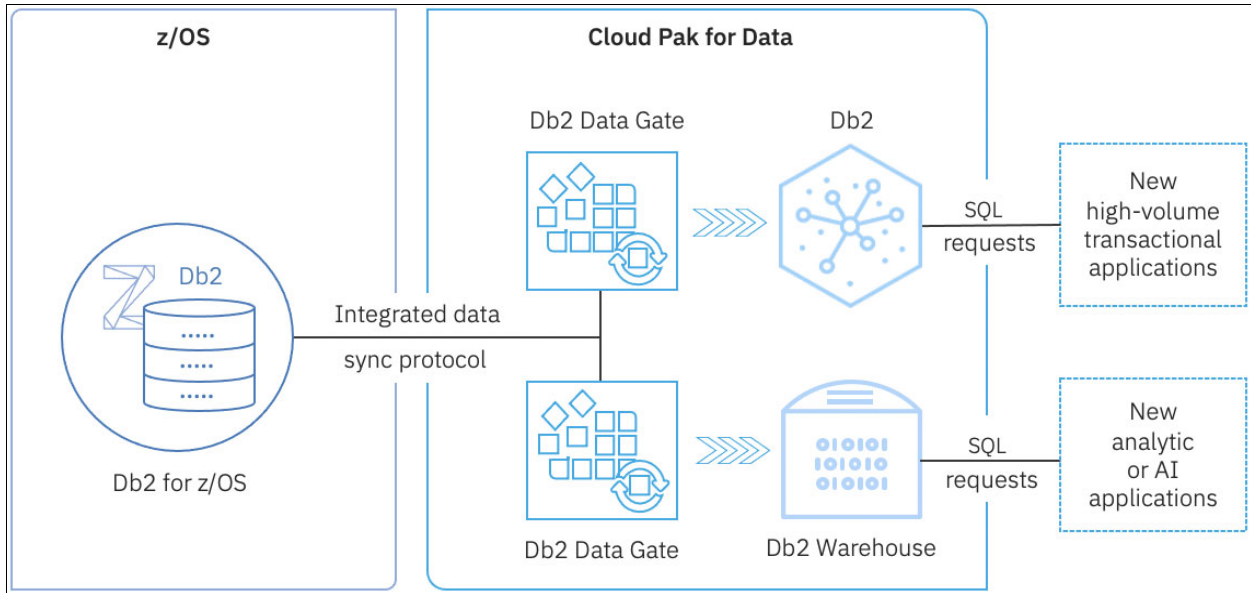


Figure 14-1 Db2 Data Gate architectural diagram

Compared to any custom-built solution, the Db2 Data Gate service reduces the cost and complexity of your application development. By using this service to synchronize and access your enterprise data on IBM Cloud Pak for Data, you can reduce your operational costs while accelerating your journey to the cloud and AI.

This chapter describes some functions of Db2 Data Gate and provides links to other key technologies within Db2 for z/OS.

This chapter describes the following topics:

- ▶ Prerequisites and configuration overview
- ▶ Administering Db2 Data Gate
- ▶ Using the most recent source data of an independent cloud replica of Db2 for z/OS by using WAITFORDATA

14.1 Prerequisites and configuration overview

This section describes the software requirements for Db2 Data Gate and provides an overview of the installation and configuration process.

14.1.1 Software dependencies

Db2 Data Gate requires the following software:

- ▶ IBM z/OS 2.2 (5650-ZOS) or later.
- ▶ IBM Db2 for z/OS with PTFs PH06628, PH19181, PH19886, PH20587, PH21187, PH21419, PH23895, PH28849, PH29443, PH30397, PH30783, PH35805, and PH37808 installed and running at Db2 12 FL505 or higher. These PTFs are required by the Integrated Synchronization function.

All these PTFs are part of Db2 13 natively and no additional PTFs are required.

- ▶ If you use the `WAITFORDATA` feature, install version 11.5.6-cn5 or later of Db2 or IBM Db2 Warehouse on IBM Cloud Pack for Data.

14.1.2 Installation overview

The process of installing Db2 Data Gate consists of the following two parts:

- ▶ Install Db2 Data Gate on IBM z/OS. For more information about how to install FMID HDGU110, see the [Program Directory for IBM Db2 for z/OS Data Gate](#). Then, install the program temporary fix for APAR PH45077.
- ▶ Install Db2 Data Gate on IBM Cloud Pak for Data. For more information, see [Installing the Db2 Data Gate service](#).

14.1.3 Configuration overview

Configuring Db2 Data Gate on IBM z/OS consists of the following steps:

1. Configure Db2 for z/OS:
 - a. Set the Db2 for z/OS subsystem parameters.
 - b. Create the Db2 Data Gate database and tables.
 - c. Create a workload manager (WLM) environment for the Db2 Data Gate stored procedures.
 - d. Define WLM performance goals for the Db2 Data Gate stored procedures.
 - e. Create the Db2 Data Gate stored procedures.
2. Configure network access:
 - a. Encrypt inbound network access.
 - b. Encrypt outbound network access from Db2 Data Gate on Cloud Pak for Data.

The network traffic between Db2 for z/OS and the Db2 Data Gate instance on IBM Cloud Pak for Data is encrypted bi-directionally. To encrypt outbound network traffic, you must specify an externally reachable hostname. The hostname is exposed by a Red Hat OpenShift route when you create a Db2 Data Gate instance.

For more information about these steps, see [Installing Db2 Data Gate on z/OS](#).

14.2 Administering Db2 Data Gate

As illustrated in Figure 14-1 on page 226, Db2 Data Gate provides an end-to-end solution to ensure that Db2 for z/OS data is available and synchronized on IBM Cloud Pak for Data. This section describes common administration tasks to maintain this end-to-end solution.

14.2.1 Selecting a target database type

You must associate either an IBM Db2 database or an IBM Db2 Warehouse database as the target database for your Db2 Data Gate instance. If your application is designed for transactional processing, you can use an IBM Db2 database. If your application is designed for analytic purposes, you must use an IBM Db2 Warehouse database.

Tip: For optimal performance, deploy the database on a dedicated Red Hat OpenShift worker node.

14.2.2 Creating a Db2 Data Gate instance

A Db2 Data Gate instance is a Kubernetes deployment that runs on IBM Cloud Pak for Data. The instance has several Kubernetes containers that provide functions for adding tables, removing tables, managing synchronization, and running `WAITFOR` queries.

When creating a Db2 Data Gate instance, you must specify the appropriate amount of resources (virtual CPU and memory) for your workload type and priority. For more information, see [Creating a Db2 Data Gate instance](#).

14.2.3 Pairing a Db2 Data Gate instance with a Db2 for z/OS subsystem

After you create a Db2 Data Gate instance, you must pair it with a single Db2 for z/OS subsystem. The process of pairing exchanges connection information between a Db2 Data Gate instance and a Db2 for z/OS subsystem. When the pairing completes, the Db2 Data Gate instance is ready to manage tables and data synchronization.

A single Db2 for z/OS subsystem can be paired with multiple Db2 Data Gate instances. You can edit the existing pair of a Db2 Data Gate instance to pair it with another Db2 for z/OS subsystem. However, a Db2 Data Gate instance can be paired with only one Db2 for z/OS subsystem at a time.

The process of pairing a Db2 Data Gate instance with a Db2 for z/OS subsystem is documented in [Creating a Db2 Data Gate instance](#).

14.2.4 Managing tables

Db2 Data Gate provides a web UI to add tables from a Db2 for z/OS source subsystem to Db2 Data Gate and enable synchronization for the added tables. After you add a table, Db2 Data Gate loads existing data and synchronizes the table's latest changes to the IBM Db2 Advanced Enterprise Server Edition (AESE) database or to the Db2 Warehouse database. Any future table changes on Db2 for z/OS are synchronized continuously. Your applications can access a Db2 or Db2 Warehouse database with near real-time data directly.

The key synchronization technology is IBM Integrated Synchronization, which is also used in IBM Db2 Analytics Accelerator. Db2 Data Gate builds on this technology and allows it to support IBM Db2 AESE databases, which use row store format.

You also can use the web UI to stop and start synchronization, remove tables, and load tables. For more information, see [Administering Db2 Data Gate](#).

14.2.5 Monitoring

You can monitor the Db2 Data Gate status and statistics on both Db2 for z/OS and IBM Cloud Pak for Data. The following mechanisms provide you with different types of information:

DISPLAY ACCEL command

When the system is configured, a heartbeat connection is established between the Db2 Data Gate instance and the connected Db2 subsystem. Every 30 seconds, the heartbeat connection provides the Db2 subsystem with status information about the connected Db2 Data Gate instance. You can view most of this information by using the **DISPLAY ACCEL** Db2 command.

SYSLOG messages

Because detailed information about specific events cannot be viewed by using the **DISPLAY ACCEL** command, this type of information is written to the z/OS system log (SYSLOG) in the form of DSNX881I messages. This information includes messages that are related to the synchronization function.

Db2 Data Gate dashboard

Db2 Data Gate provides a browser-based dashboard, which is shown in Figure 14-2, to display the status and statistics on the IBM Cloud Pak for Data side.

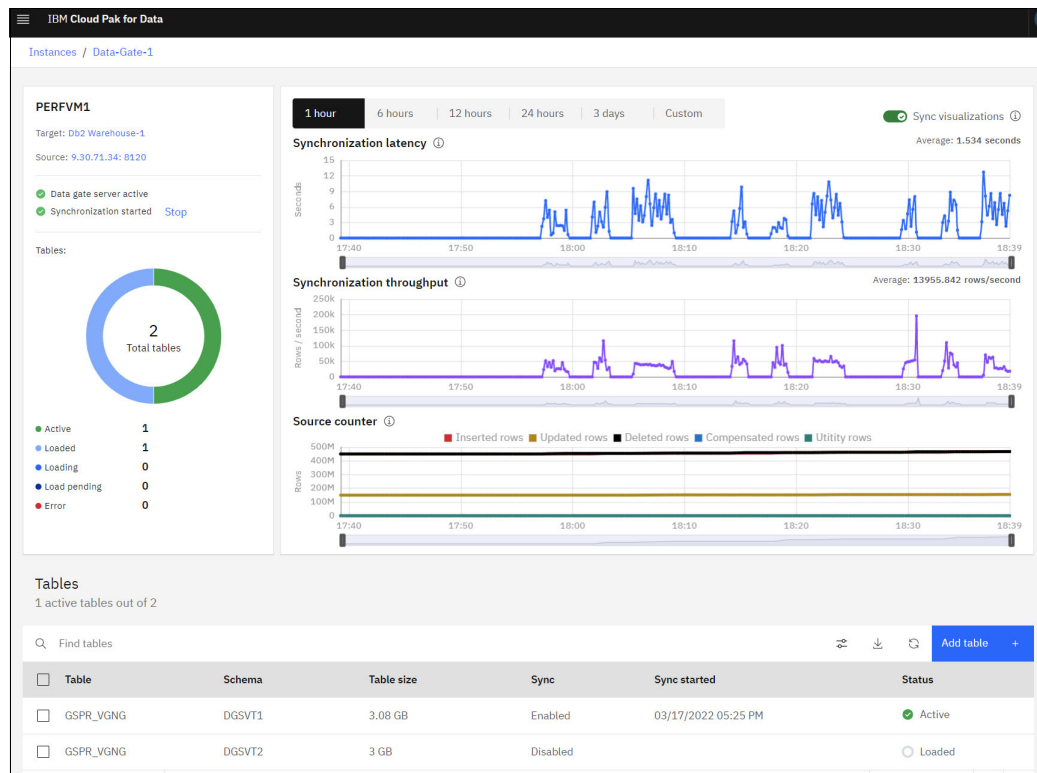


Figure 14-2 Db2 Data Gate dashboard

The Db2 Data Gate dashboard provides the following information:

- ▶ Db2 Data Gate server status
- ▶ Synchronization status
- ▶ Synchronization latency (seconds)
- ▶ Synchronization throughput (rows per second)
- ▶ Tables status pie chart

For more information about the data that is displayed in the dashboard and how to use it, see [Monitoring a Db2 Data Gate instance](#).

14.3 Using the most recent source data of an independent cloud replica of Db2 for z/OS by using WAITFORDATA

Db2 Data Gate provides direct access to the target database and asynchronous data synchronization based on IBM Integrated Synchronization, which is described in 13.4, “Integrated Synchronization enhancements” on page 213. A significant benefit of IBM Integrated Synchronization is that the resource utilization on the source database for replication is small. More importantly, the transaction execution time on the source is not impacted in any way. However, the asynchronous synchronization process results in an inevitable latency between the source and the target database.

Although Db2 Data Gate usually operates with low synchronization latency in the subsecond to low-seconds range, this level of performance might not satisfy the requirements for applications that must work with the most recent source data.

A good example of this requirement is a mobile banking application that both transfers data that is written in a source Db2 for z/OS database and that also displays the current account balance by querying the replicated target Db2 Warehouse. Due to the synchronization latency, if users of this mobile banking app refresh their balance immediately after they make a transfer, the transaction that is related to the transfer might not be reflected in their balance because the transfer has not yet been synchronized to the target database. This scenario represents a typical read-after-write inconsistency issue.

To mitigate such issues, Db2 Data Gate offers a feature that is called **WAITFORDATA**, which allows queries on synchronized tables that were submitted on the target database to be delayed until the most recent data from the source database is synchronized. The feature is comparable to the **WAITFORDATA** feature that is offered by Db2 Analytics Accelerator for z/OS, as described in 13.4.4, “True HTAP support (the WAITFORDATA feature)” on page 216.

Note: The main difference between the two implementations of **WAITFORDATA** is that for Db2 Data Gate the **WAITFORDATA** feature applies to queries that are directly run on the Db2 target database, and for Db2 Analytics Accelerator, the **WAITFORDATA** feature applies to queries that are routed from Db2 for z/OS to Db2 Analytics Accelerator and run there.

14.3.1 How WAITFORDATA works

Figure 14-3 illustrates how Db2 Data Gate uses **WAITFORDATA** to enable queries on the target database to run on the most recent data from the source database.

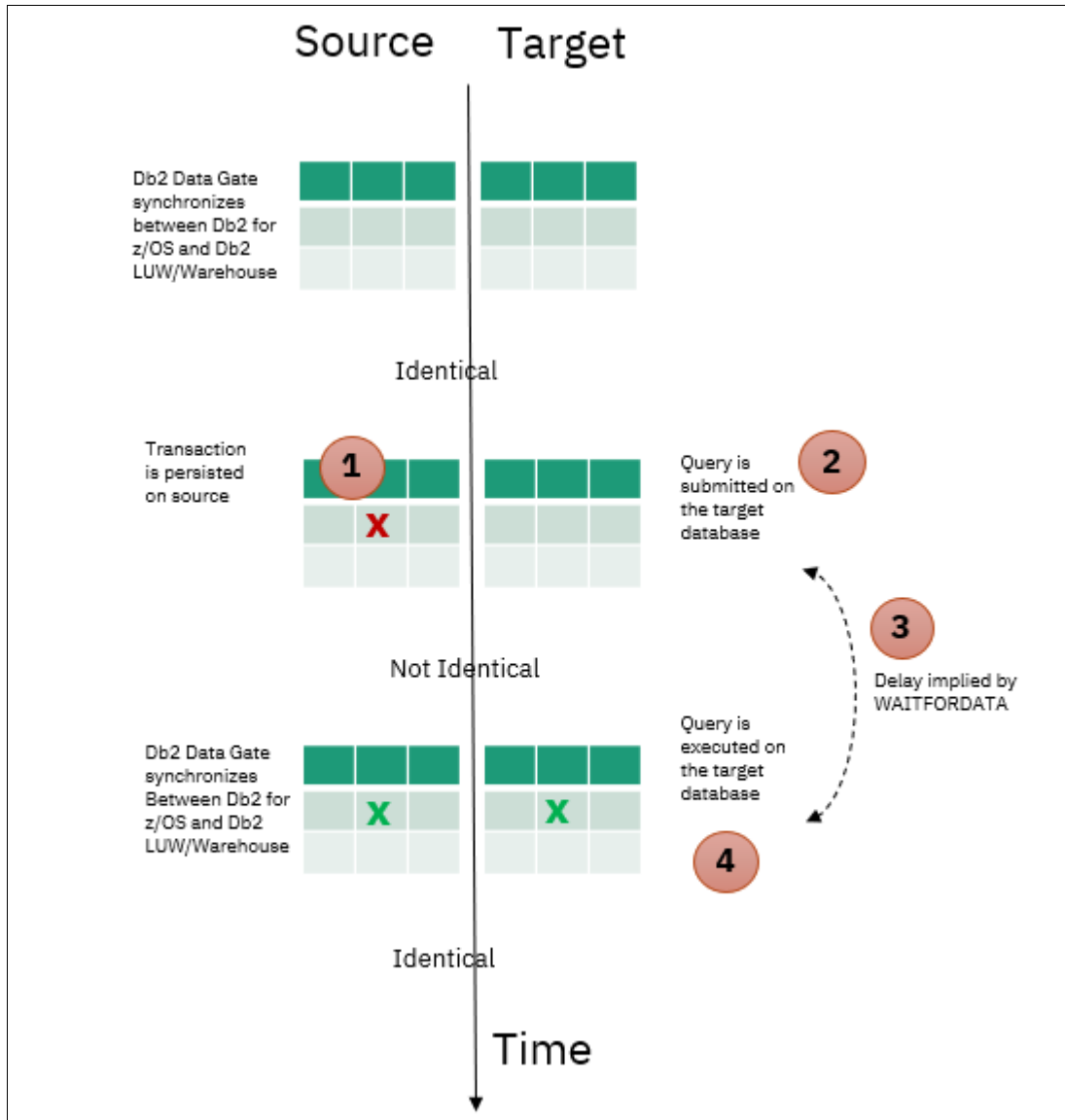


Figure 14-3 The role of **WAITFORDATA** for query execution

In Figure 14-3:

1. A transaction runs on the source database that persists data changes in a source-synchronized table.
2. An application requests to read data from the target-synchronized table and uses the **WAITFORDATA** feature. Now, the data from the transaction in step 1 has not yet been synchronized to the target database.
3. **WAITFORDATA** delays the transaction until the most recent source data is synchronized to the target database.
4. When the data is synchronized, the submitted query runs on the target database and accesses all the required data.

14.3.2 WAITFORDATA behavior

The **WAITFORDATA** feature controls are provided by the target database. An application that submits queries to the target database can specify a timeout period for delaying query execution by setting the **SET CURRENT QUERY WAITFORDATA** special register. The special register must be set at the beginning of the SQL transaction before the query statements that run.

The following example shows how to specify the special register:

```
SET CURRENT QUERY WAITFORDATA = 10.0;  
SELECT * FROM XYZ WHERE [...];
```

In this example, a query waits up to 10 seconds for the most recent source data before the query either runs or fails with a **TIMEOUT** exception. The **CURRENT QUERY WAITFORDATA** register value specifies a maximum delay of each query, that is, if the change set is replicated from the source to the target database before reaching the timeout, the query runs instantly without further delays.

WAITFORDATA delays are valid for synchronized tables only, but a query also can include other types of tables. **WAITFORDATA** behaves differently depending on the type or types of tables that are referenced in a query:

Only synchronized tables are referenced:

Query processing is delayed until all referenced tables are synchronized.

Synchronized and non-synchronized tables are referenced:

Non-synchronized tables are ignored. Query processing starts when synchronized tables are updated within the **WAITFORDATA** delay period.

Only non-synchronized tables are referenced:

The **WAITFORDATA** delay is ignored, and query processing starts immediately.

A query references no tables at all:

The delay is applied so that query processing starts after the newest source data for all replicated tables is synchronized. This behavior is useful, for example, for running stored procedures on the target database with the most recent source data.

14.4 Summary

Db2 Data Gate provides analytics and digital applications with low latency, read-only access to data that originates in Db2 for z/OS. Data is synchronized between Db2 for z/OS data sources and target Db2 databases on IBM Cloud Pak for Data by using IBM Integrated Synchronization technology. So instead of accessing data in the IBM Z data source directly, an application accesses a synchronized copy of the Db2 for z/OS data that is hosted by a separate system.

Using the **WAITFORDATA** feature ensures that applications access most recent committed data that is synchronized from Db2 for z/OS to the target database. Latency of the committed data does not impact consistency or coherency of the selected data because queries first run when the required data is available on the target database.



IBM Db2 12 continuous delivery features and enhancements

Db2 12 for z/OS provides many valuable features and enhancements that were continuously delivered in a single maintenance stream after the general availability (GA) of Db2 12. These features and enhancements can be grouped into three functional areas, which are summarized in the following sections of this appendix:

- ▶ Application development
- ▶ Database administration
- ▶ System administration and system management

This appendix also includes some considerations for your plan to activate Db2 12 function level 510 (V12R1M510), which is the prerequisite for migration to Db2 13, in “Activating V12R1M510: Considerations” on page 249.

If you are already running Db2 12 FL 510, for more information about migration to Db2 13, see Chapter 2, “Installing and migrating” on page 13.

Db2 12 does not add more function levels after FL 510, but adds capabilities in the maintenance stream. For more information about Db2 12 features that are delivered after GA, see [What's new in Db2](#). From there, you can find more information about specific function levels and other changes by reviewing the following resources:

- ▶ [Db2 12 function levels](#)
- ▶ [Recent enhancements to Db2 12](#)

Application development

Through continuous delivery, Db2 12 delivers numerous new and enhanced functions that support application development. In addition to new and changed SQL statements and the new REST service support, Db2 12 adds other features to enhance your application development experience, which are described in Appendix B, “Application development” on page 253.

SQL enhancements

The following enhancements to Db2 12 SQL represent either new SQL functions or changed SQL behaviors. The enhancements, including the ones in the following sections, were delivered in various function levels in the order of their availability.

- ▶ LISTAGG
- ▶ Casting of numeric values to GRAPHIC and VARGRAPHIC data types
- ▶ Temporal auditing query update
- ▶ Global variable for data replication override
- ▶ Syntax flexibility for special registers and NULL predicates
- ▶ WHEN clause on trigger visibility to rows in history or archive tables
- ▶ Column level security with new built-in functions
- ▶ Alternative names for built-in functions
- ▶ CREATE OR REPLACE syntax for stored procedures
- ▶ Application-level granularity for lock limits
- ▶ Referential integrity support for UPDATE or DELETE for parent tables in business-time temporal relationships

To use a specific new SQL function or a changed SQL behavior, you must activate a specific containing function level and run the enhanced SQL at the corresponding application compatibility (APPLCOMPAT) level.

LISTAGG

LISTAGG is a new, built-in aggregate function that produces a list of values that based on the inputs to the function. Introduced in FL 501 (V12R1M501), this function provides a much simpler SQL statement than what was previously required to produce such a value list.

For more information about LISTAGG, see [Function level 501 \(activation enabled by APAR PI70535 - May 2017\)](#).

Casting of numeric values to GRAPHIC and VARGRAPHIC data types

Db2 12 FL 502 (V12R1M502) supports direct casting of numeric values to GRAPHIC or VARGRAPHIC data types. Before FL 502 (V12R1M502), you had to cast numeric values to a character data type and then cast the character data type to a graphic data type. As shown in Example A-1 and Example A-2 on page 235, with FL 502, you can cast directly from a numeric data type to a double-byte GRAPHIC or VARGRAPHIC data type. The examples assume that the QUANTITY column is defined as INTEGER.

Example A-1 Casting numeric values to a graphic data type

```
SELECT CAST(QUANTITY AS GRAPHIC(10)) FROM ORDER_LINE_ITEM... ;
```

Example A-2 Casting numeric values to a variable graphic data type

```
SELECT VARGRAPHIC(QUANTITY) FROM ORDER_LINE_ITEM... ;
```

For more information, see [Explicit casting of numeric values to GRAPHIC or VARGRAPHIC](#).

Temporal auditing query update

Db2 12 FL 503 (V12R1M503) changes the behavior of certain temporal queries to correct the query results. This change applies to system-period temporal (also called row versioning) tables that are defined with **ON DELETE ADD EXTRA ROW**, where **GENERATED ALWAYS AS (DATA CHANGE OPERATION)** columns were added to both the system-period temporal table and the history table. Existing rows had NULL values for the added columns. Before FL 503 (V12R1M503), history table rows with NULLS were not returned in queries. For more information, see [Temporal auditing](#).

Global variable for data replication override

Row versioning, also called system temporal time, allows a history table to track changes to a row over time. If that table is replicated, you must override the automatic versioning at the remote replication target. This override is made easy in Db2 12 FL 503. By setting the built-in global variable **SYSIBMADM.REPLICATION_OVERRIDE** to 'Y' on the remote target, you achieve the complete and correct replication of data changes for tables that are enabled for row versioning. For more information, see [REPLICATION_OVERRIDE](#).

Syntax flexibility for special registers and NULL predicates

Db2 12 FL 504 (V12R1M504) provides syntax options that enhance the productivity of developers. For example, client information fields, such as accounting, application name, user ID, or workstation name, might be labeled differently on other platforms. This feature allows developers working with applications on multiple platforms to use the same syntax. It also eases the effort to change applications to access Db2 for z/OS data.

Table A-1 summarizes the new options.

Table A-1 Syntax flexibility for special registers and NULL predicates

Existing syntax	New syntax option
CURRENT CLIENT_ACCTNG	CLIENT ACCTNG
CURRENT CLIENT_APPLNAME	CLIENT APPLNAME
CURRENT CLIENT_USERID	CLIENT USERID
CURRENT CLIENT_WRKSTNNAME	CLIENT WRKSTNNAME
CURRENT SERVER	CURRENT_SERVER
CURRENT TIME ZONE CURRENT TIMEZONE	CURRENT_TIMEZONE
IS NULL	ISNULL
IS NOT NULL	NOTNULL

For more information, see [New SQL syntax alternatives for special registers and NULL predicates](#).

WHEN clause on trigger visibility to rows in history or archive tables

Db2 12 FL 505 (V12R1M505) allows you to reference system-period temporal tables or archive-enabled tables in the **WHEN** clause of triggers when you want to specify system time sensitivity or archive sensitivity. For more information, see “WHEN clauses that reference temporal or transparent archive tables” on page 255.

Column level security with new built-in functions

Db2 12 FL 505 (V12R1M505) improves your ability to secure sensitive data. You can encrypt and decrypt sensitive data at the column level with new built-in functions. These functions are based on AES256 algorithms and key labels. Your applications invoke the encryption and decryption built-in functions, and the data remains encrypted except when being used in your application. The encrypt and decrypt built-in functions are important because they provide a more efficient way to encrypt sensitive data than other alternatives. For more information about these and more security features, see “Security” on page 300.

Alternative names for built-in functions

Db2 12 FL 506 (V12R1M506) provides alternative names for built-in functions (BIFs), which enhances the productivity of developers. Some functions have different names on other database platforms. This feature allows developers who work with applications on multiple platforms to use the same function names. It also reduces the effort that was required to change applications to access Db2 for z/OS data.

Table A-2 summarizes the new options.

Table A-2 Alternative names for built-in functions

Existing name	New alternative name
CHARACTER_LENGTH	CHAR_LENGTH
COVARIANCE of COVAR	COVAR_POP
HASH_MD5(expression)	HASH(expression, 0) or HASH(expression)
HASH_SHA1(expression)	HASH(expression, 1)
HASH_SHA256(expression)	HASH(expression, 2)
POWER	POW
RAND	RANDOM
LEFT	STRLEFT
POSSTR	STRPOS
RIGHT	STRRIGHT
CLOB	TO_CLOB
TIMESTAMP_FORMAT	TO_TIMESTAMP

For more information, see [Alternative function names support](#).

CREATE OR REPLACE syntax for stored procedures

Improving support for agile development and ease of deployment, Db2 12 FL 507 (V12R1M507) adds support for the **CREATE OR REPLACE** syntax for stored procedures. You can change existing stored procedures without having to **DROP** and then **CREATE** them, as was required before FL 507. Another benefit is that you can replace stored procedures while preserving existing authorizations.

CREATE OR REPLACE PROCEDURE applies to native SQL procedures or external stored procedures. For native SQL procedures, you can specify **VERSION** as part of the change process. For more information, see “CREATE OR REPLACE syntax for stored procedures” on page 255.

Application-level granularity for lock limits

Subsystem parameters **NUMLKTS** and **NUMLKUS** control the number of locks for table spaces or users. If the number of locks for a table space exceeds **NUMLKTS**, lock escalation occurs for eligible table spaces. If an application exceeds **NUMLKUS**, the application receives a -904 SQLCODE. In some cases, these two subsystem parameter values do not provide sufficient flexibility for varying application requirements.

Db2 12 FL 507 (V12R1M507) provides the following two built-in global variables that allow you to specify locking limits for table spaces or users at the application level:

- ▶ **SYSIBMADM.MAX_LOCKS_PER_TABLESPACE**
- ▶ **SYSIBMADM.MAX_LOCKS_PER_USER**

You can also set these global variables for Distributed Data Facility (DDF) applications with the Db2 system profile tables. For more information about these global variables, see “Application-level granularity for lock limits” on page 237.

Referential integrity support for UPDATE or DELETE for parent tables in business-time temporal relationships

For business-time temporal tables that are parents in a referential integrity (RI) relationship, both **UPDATE** and **DELETE** were not allowed if **FOR PERIOD OF BUSINESS_TIME** was specified. Db2 12 FL 509 (V12R1M509) allows **UPDATE** or **DELETE** operations in such situations. Before FL 509, an **UPDATE** or **DELETE** in such situations resulted in SQLCODE -4736. For more information, see [Temporal RI allows UPDATE or DELETE on the parent table](#).

Db2 REST services

Db2 REST Service support allows an authorized user to access Db2 data with a REST HTTP client that sends JavaScript Object Notation (JSON) as input and receives JSON output, which allows you to securely share Db2 data through modern programming APIs that are adopted by mobile, web, and cloud applications. When you expose Db2 REST services, which comprise static SQL, mobile, web, or cloud developers do not need to understand or write SQL.

You can also invoke Db2 REST services by using IBM z/OS Connect APIs, which extend the value of Db2 REST services in combination with other z/OS based resources.

The Db2 12 REST services support delivered the following enhancements:

- ▶ The initial delivery of Db2 native REST support
- ▶ The options to create, discover, invoke, and delete REST services
- ▶ Db2 REST services versioning

- ▶ **COPY** options for managing and promoting Db2 REST services
- ▶ The ability to remotely issue the **FREE SERVICE** subcommand

For more information, see “Db2 REST services” on page 259.

For more information about how to enable Db2 REST support, see [Db2 REST services](#).

Other application development capabilities

Db2 12 also delivered the following new and enhanced capabilities for application development:

- ▶ SQL PL debugging
- ▶ Setting the **MOVE_TO_ARCHIVE** default for transparent archiving
- ▶ **OPEN/FETCH** warning message suppression
- ▶ **ISOLATION(UR)** loophole for **INSERT**, **UPDATE**, or **DELETE** closed

SQL PL debugging

Developers need another option instead of Data Studio to debug SQL Procedure Language routines. APAR PI44721 provides eight sample native SQL procedures that you can use to trace SQL PL routines. This enhancement is independent of function level. For more information, see [PI44721](#).

Another option is to use the Unified Debugger, which allows you to remotely debug stored procedures that run on Db2 for z/OS servers. For more information, see [Debugging stored procedures by using the Unified Debugger](#).

Setting the **MOVE_TO_ARCHIVE** default for transparent archiving

Db2 transparent archiving, introduced in Db2 11 for z/OS, relies on the **MOVE_TO_ARCHIVE** global variable. The default for this global variable was ‘N’, meaning developers or administrators must change the setting when appropriate. APAR PI68205 adds the subsystem parameter **MOVE_TO_ARCHIVE_DEFAULT** to remove the requirement to set the global variable for standard cases. For more information, see [PI68205](#).

OPEN/FETCH warning message suppression

PI79053 allows you to suppress warning messages that are related to **OPEN** or **FETCH** by adding the keyword **QUIET** to the tolerate warning (**TOLWARN**) parameter when using sample program DSNTIAUL. For more information, see [PI79053](#).

ISOLATION(UR) loophole for **INSERT**, **UPDATE**, or **DELETE** closed

APAR PH14791 provides subsystem parameter **ALLOW_UPD_DEL_INS_WITH_UR** with the default of **N0** to prevent embedded dynamic SQL from including **INSERT**, **UPDATE**, or **DELETE** in cases where **WITH UR** is specified in the **PREPARE ATTRIBUTES**. Executions of **INSERT**, **UPDATE**, or **DELETE** that are prepared this way can cause Db2 abends. For more information, see [PH14791](#).

Database administration

Enhancements in package management processes, SQL Data Definition Language (DDL), and Db2 utilities can increase the productivity of database administrators and reduce elapsed time and CPU cost for administering database objects. For more information about some of the **REBIND** and DDL changes, and enhancements to Db2 Utilities, see Appendix C, “Database administration” on page 263.

Package management

Db2 12 provides several enhancements that are related to managing packages. The most significant of these enhancements is **REBIND** phase-in, which is the ability to create a copy of a package while existing threads continue to use the previous copy of the package.

Db2 12 provides several other significant changes to package management, including enhancements to **FREE PACKAGE** and **APREUSE**, more options for database request module (DBRM) management, improved automatic rebind (autobind), and clarification of package versioning and ownership.

REBIND phase-in

This enhancement, introduced in Db2 12 FL 505, improves DBA productivity and application availability. You no longer need to wait for applications to stop before rebinding the packages that the applications use. You can react quickly to situations that call for package rebinds. For more information, see “REBIND phase-in” on page 264.

New FREE PACKAGE option

PH33295 adds the **PHASEOUT** option to the **FREE PACKAGE PLANMGMTSCOPE** subcommand, which allows you to remove unused copies of packages and reduce storage consumption in the Db2 catalog and directory. For example, you can use the following subcommands to free eligible phased-out copies:

```
FREE PACKAGE PLANMGMTSCOPE(PHASEOUT)
FREE PACKAGE PLANMGMTSCOPE(INACTIVE)
```

For more information about the **FREE PACKAGE** syntax, see [FREE PACKAGE \(DSN\)](#).

APREUSE

The following two enhancements improve your ability to use **APREUSE** to manage package performance:

- ▶ PH36728 reduces the likelihood of **APREUSE** errors or warnings in certain cases that involve query transformations. For more information, see [PH36728](#).
- ▶ PH36179 allows **APREUSE** to consider page range screening for the query access plan even when the earlier plan did not have page range screening. For more information, see [PH36179](#).

Autobind attempts APREUSE

During an autobind, Db2 tries to reuse the access path from the current package as though **APREUSECOURSE(CURRENT)** were specified. This enhancement reduces cases where an autobind, such as for an invalidated package or a package from an unsupported Db2 release, would result in a different access path and cause performance regression. If an access path cannot be reused, the autobind is successful and no messages are issued. For more information, see [PH15896](#).

DBRMs in Hierarchical File System for COBOL and PL/I

COBOL and PL/I compilers can write DBRMs to Hierarchical File System (HFS) files. This enhancement reflects greater use of z/OS UNIX System Services for application development activities.

Example A-3 provides the COBOL syntax and PL/I syntax for writing DBRMs to HFS files.

Example A-3 COBOL syntax and PL/I syntax for writing DBRMs to HFS files

```
cob2 myprogram.cbl -o myprogram -dbrmlib -qsql  
pli -o -qpp=sql -qdbrmllib -qrent myprogram.pli
```

For more information, see [PI88171](#).

Package versioning and package ownership

For packages that are associated with more than one version, duplicate records of implicit ownership might potentially exist in SYSIBM.SYSPACKAUTH. These duplicates do not cause any issues, but can appear to be a security issue. Db2 now considers the unique consistency token (**CONTOKEN**) for the package version when modifying and adding records in SYSPACKAUTH during the **BIND** or **REBIND** process. Db2 removes existing duplicate records during the **REBIND** process. For more information, see [PH32258](#).

Data Definition Language

The major changes to DDL support Huffman compression; encourage adoption of universal table spaces (UTS) by restricting creation of non-UTS objects; and add the **MOVE TABLE** option to the **ALTER TABLESPACE** statement, which facilitates converting multi-table table spaces to partitioned-by-growth (PBG) UTS objects.

Other DDL enhancements include implicit drop of explicitly created table spaces, implicit hidden ROWID columns, meta-data self-description, support of DECFLOAT columns in indexes, and the ability to rotate partitions for materialized query tables.

Huffman compression

Db2 12 function level 504 (FL 504) adds support for Huffman compression, also called entropy encoding. Huffman compression provides you with the opportunity to achieve even higher compression ratios with comparable or lower CPU costs than you can achieve with traditional table space compression. Both compression types are dictionary-based compression. Traditional compression uses fixed-length dictionary entries, but Huffman uses variable-length dictionary entries, with the shorter entries used for the most frequently occurring patterns.

Db2 12 function level 509 (FL 509) allows you to specify which type of encoding you prefer at the page set or partition level. FL 509 relies on catalog changes to track your compression encoding preferences. In addition, you can now use the **DSN1COMP** stand-alone utility to estimate the compression savings of objects that are compressed.

For more information about Huffman compression, see “Huffman compression” on page 267.

Depreciating Db2 objects

Db2 12 function level 504 (FL 504) allows you to control whether deprecated objects, including non-UTSs, hash-organized table spaces, and SYNONYMs, can be created in your environment. This capability reduces the chance that you create deprecated objects and encourages the shift to strategic UTSs. This control over creating deprecated objects depends on the APPLCOMPAT level setting for the package that you use to issue the DDL. If you use a package with an APPLCOMPAT level of V12R1M504 or later, you cannot create deprecated objects.

For more information, see “Preventing new deprecated objects” on page 270.

Implicit drop of explicitly created table spaces

Db2 12 function level 506 (FL 506) allows you to implicitly drop table spaces that you had created explicitly. When you drop a table in a UTS or an auxiliary table in a large object (LOB) table space, the corresponding table space is implicitly dropped, assuming that the package that you are running is at APPLCOMPAT level V12R1M506 or later. This capability applies if you drop a table in a UTS or drop an auxiliary table in a LOB table space. You can use this capability when you drop a system-period temporal table or an archive-enabled table to implicitly drop the related history or archive table and the associated table spaces, as shown in Example A-4.

Example A-4 DROP TABLE

```
CREATE TABLESPACE MYTS1 SEGSIZE 64 MAXPARTITIONS 2 IN MYDB1;  
CREATE TABLE MYTB1 (.....) IN MYDB1.MYTS1;  
DROP TABLE MYTB1;
```

If you are running a package with at least APPLCOMPAT level V12R1M506, the **DROP TABLE MYTB1** statement causes Db2 to drop table space MYTS1.

For an auxiliary table, the result is different than it is before FL 506. You might have processes that expect to reuse the LOB table space after you drop the auxiliary table, or you might have processes that include the explicit drop of the LOB table space. If either condition is true, you should adjust your processes. Example A-5 shows this scenario.

Example A-5 DROP auxiliary table

```
CREATE LOB TABLESPACE MYLOBTS1 IN MYDB1;  
CREATE AUXILIARY TABLE MYAUXTB1 IN MYDB1.BYLOBST1 STORES . . . ;  
DROP TABLE MYAUXTB1;
```

With the drop of MYAUXTB1, Db2 implicitly drops table space MYLOBTS1.

Example A-6 shows the system-period temporal table scenario; the same approach applies to the archive-enabled table scenario.

Example A-6 DROP system-period temporal or archive-enabled table

```
CREATE TABLESPACE MYTS1 SEGSIZE 64 MAXPARTITIONS . . . ;  
CREATE TABLESPACE MYTS2 SEGSIZE 64 MAXPARTITIONS . . . ;  
CREATE TABLE STT_ACT . . . IN MYDB1.MYTS1 ;  
CREATE TABLE STT_HIST . . . IN MYDB1.MYTS2 ;  
ALTER TABLE STT_ACT ADD VERSIONING . . . ;  
DROP TABLE STT_ACT ;
```

In this case, dropping the table STT_ACT leads to Db2 implicitly dropping both MYTS1 and MYTS2.

Migrating multi-table table spaces to partition-by-growth universal table spaces

Db2 12 function level 508 (FL 508) gives you the ability to convert from multi-table table spaces to strategic partition-by-growth (PBG) UTSs without an application outage. You can use the **MOVE TABLE** option of the **ALTER TABLESPACE** statement to move a table from the altered source table space to an empty target PBG UTS as a pending definition change. Then, a **REORG** with **SHRLEVEL CHANGE** or **REFERENCE** on the source table space materializes the pending move table operation.

For more information, see “Migrating multi-table table spaces to PBG UTS” on page 274.

Defining implicitly hidden ROWID columns

Db2 12 can define a hidden ROWID column to use as a partitioning key when there is no suitable alternative. For example, you can choose to migrate a PBG UTS to a PBR UTS to spread high-volume insert activity across the partitions, as shown in Example A-7.

Example A-7 Defining an implicitly hidden ROWID column

```
CREATE TABLE MYTB2 (ID INTEGER, TEXT CHAR(10),  
PKEY ROWID NOT NULL IMPLICITLY HIDDEN) IN . . .  
PARTITION BY (PKEY)  
(PARTITION 1 ENDING AT (X'4999'),  
PARTITION 2 ENDING AT (X'7999'),  
PARTITION 3 ENDING AT (X'A999'),  
PARTITION 4 ENDING AT (MAXVALUE));
```

Db2 generates a pseudo-random value for the first 8 bytes of the ROWID column. This approach has no application impact.

For more information, see [Row ID values](#).

Enhanced meta-data self-description capability

System pages, which make Db2 tables and table spaces self-describing, can be missing for tables that are still at version 0. After you apply APARs PI86880 and PI88940, Db2 automatically inserts system pages after **INSERT**, **UPDATE**, **REORG**, or **LOAD REPLACE**. Db2 also writes version 0 system pages for newly created tables after the definition is complete.

You can check for objects that are missing system pages by using **REPAIR CATALOG TEST**.

For more information, see [PI86880](#) and [PI88940](#).

DECFLOAT columns in indexes

You can define unique or non-unique indexes with DECFLOAT columns. You can also specify DECFLOAT columns in a unique constraint or in a primary key constraint. They can also be specified in a unique key or primary key constraint. You cannot specify DECFLOAT columns in referential constraints or as part of a partitioning key, and you cannot include DECFLOAT columns in INCLUDE indexes or in indexes on expression. Implicit casting to a DECFLOAT data type remains a stage 2 operation in Db2 12, regardless of index support.

For more information, see [PH21189](#).

ROTATE PARTITION for materialized query tables

You can now use the **ALTER TABLE . . . ROTATE PARTITION** statement to rotate partitions for materialized query tables (MQTs) or for tables with dependent MQTs.

For more information, see [PH00194](#).

Utilities

Db2 delivered many new utility capabilities after Db2 12 GA. These enhancements fall into one or more of the following categories:

- ▶ Utility support for Db2 enhancements that were added after GA
- ▶ Utility performance improvements
- ▶ Utility availability improvements, including concurrency with other utilities and with other applications
- ▶ Usability and simplification enhancements to improve your productivity

For more information about these Db2 12 utilities capabilities, see “Db2 utilities” on page 277.

System administration and system management

The initial release of Db2 12 together with its subsequent function levels and maintenance provide significant improvements in the areas of system administration and system management that can increase productivity, improve performance, and provide you with a higher level of security. This section provides an overview of these changes and enhancements, some of which are described more fully in Appendix D, “System administration and system management” on page 289:

- ▶ Performance
- ▶ Db2 system-related enhancements
- ▶ Security
- ▶ Synergy with z15 and IBM DS8000 product families

Performance

Enhancements in the performance area include monitoring and management changes, and opportunities for CPU and elapsed time reductions.

Resource limit facility

Db2 extends autonomics and improves your productivity when you use the resource limit facility (RLF). Db2 monitors activity on the active RLF tables and automatically refreshes the in-memory tables that Db2 uses to manage RLF limits.

In addition, the **START RLIMIT** command supports **SCOPE(GROUP)** for data-sharing configurations.

For more information, see [-START RLIMIT \(Db2\)](#).

The **DEFAULT_INSERT_ALGORITHM** default

Db2 12 changes the default setting for the **DEFAULT_INSERT_ALGORITHM** subsystem parameter from 2 to 1. This change reduces the likelihood that UTS objects will be defined with insert algorithm 2. You can set this parameter to 0 to disallow insert algorithm 2.

Accounting and statistics trace enhancement for record identifier list processing

To reflect record identifier (RID) changes that were introduced with FL 500, Db2 populates two RID list counters that reflect the maximum number of RID blocks that overflowed and the current number of RID blocks that overflowed. Db2 12 adds an exception counter to reflect the number of times that the final RID list processing was not used.

Db2 12 makes the following changes to support RID list processing:

- ▶ The description of QISTRLLM is changed to RDS LIMIT EXCEEDED.
- ▶ The following two fields are populated:
 - QISTWFRHIG (MAX RID BLOCKS OVERFLOWED)
 - QISTWFRCUR (CURRENT RID BLOCKS OVERFLOWED)
- ▶ The QXRFMIAF (RID LIST PROCESSING WAS NOT USED) field is added.

For more information, see [PH07513](#).

Fast index traversal (index fast traverse blocks (FTBs))

Db2 12 extends the benefits of fast index traversal after GA by adding support for non-unique indexes and providing you greater flexibility in monitoring and controlling fast index traversal in your environment.

For more information, see “Fast index traversal” on page 290.

SORTL

Db2 adds support for IBM Integrated Accelerator for IBM Z Sort, which is available on IBM z15 and later systems and uses the **SORT LIST (SORTL)** instruction to sort lists of data. Db2 can use **SORTL** for relational data system (RDS) sort, with the **REORG** utility, or on **DFSORT** invocation. For more information, see “SORTL” on page 293.

Db2 system-related enhancements

Db2 12 includes enhancements in replication, data sharing, DDF, system monitoring, and subsystem parameter (**DSNZPARM**) simplification.

Db2 adds the system-related improvements through the maintenance stream. These improvements do not depend on any particular Db2 function level.

- ▶ Replication
- ▶ Data sharing
- ▶ Distributed Data Facility
- ▶ Subsystem monitoring
- ▶ DSNZPARM simplification

Replication

To avoid possible inconsistencies between the source and target side of replication, Db2 adds the new **UTILS_BLOCK_FOR_CDC** subsystem parameter to block certain utilities for tables that are enabled for replication. **UTILS_BLOCK_FOR_CDC** defaults to **NO**, which is consistent with earlier behavior. Setting this parameter to **YES** avoids the potential inconsistencies that are caused by the following utility invocations:

- ▶ **CHECK DATA** with **DELETE YES LOG NO**
- ▶ **LOAD** with **SHRLEVEL NONE/REFERENCE**
- ▶ **RECOVER** to point-in-time (PIT)
- ▶ **REORG TABLESPACE** with **DISCARD**
- ▶ **REPAIR** with **LOCATE DELETE**

If you specify any of these supported Db2 utility options, you risk producing an inconsistency between the source and replication target, requiring a refresh of the target table. Setting **UTILS_BLOCK_FOR_CDC** to **YES** removes that risk.

Data sharing

Data sharing has the following enhancements:

- ▶ **UNLOAD**
- ▶ Retrying automatic logical page list and group buffer pool recover pending recovery
- ▶ Asynchronous cross-invalidation of group buffer pools
- ▶ Internal resource lock manager deadlock
- ▶ **ALTER GROUPBUFFERPOOL**
- ▶ Sysplex group authentication with Sysplex workload balancing
- ▶ Multi-factor authentication without Sysplex workload balancing

UNLOAD

Db2 12 changes the default of the **REGISTER** option for **UNLOAD...SHRLEVEL CHANGE ISO(UR)** to **YES** to ensure that the unloaded rows include the latest changes on other members in the Db2 data sharing group.

For more information, see “Data sharing” on page 294.

Retrying automatic logical page list and group buffer pool recover pending recovery

Db2 12 adds messages to support the retry of automatic logical page list (LPL) or group buffer pool recover pending (GRECP) recovery.

For more information, see “Data sharing” on page 294.

Asynchronous cross-invalidation of group buffer pools

Db2 adds support for asynchronous cross-invalidation (XI) requests to GBPs, which can reduce elapsed time for certain application processes.

For more information, see “Data sharing” on page 294.

Internal resource lock manager deadlock

The internal resource lock manager (IRLM) improves deadlock resolution in situations with many waiters. For more information, see “Data sharing” on page 294.

ALTER GROUPBUFFERPOOL

Db2 12 raises the limit of the **RATIO** option of the **-ALTER GROUPBUFFERPOOL** command from 255 to 1024 and changes the default for **RATIO** from 5 to 10. These changes reflect prevailing workload patterns of high page registration and the need for more directory entries.

For more information, see [-ALTER GROUPBUFFERPOOL \(Db2\)](#).

Sysplex group authentication with Sysplex workload balancing

Db2 12 allows IBM data server driver clients that use Sysplex workload balancing or seamless failover to retain authentication to the members of the Db2 data sharing group when using multi-factor authentication (MFA) or IBM RACF PassTickets.

For more information, see “Data sharing” on page 294.

Multi-factor authentication without Sysplex workload balancing

Db2 12 allows distributed clients that do not use Sysplex workload balancing or seamless failover to cache authentication to the members of the Db2 data sharing group when using MFA.

For more information, see “Data sharing” on page 294.

Distributed Data Facility

Db2 12 enhances your visibility into and control of DDF with changes to DDF messaging and with extensions to system profile monitoring for certain use cases.

DDF messaging

Db2 12 adds in-use thread information to several existing messages and adds two new messages that are related to in-use DBATs.

For more information, see “Distributed Data Facility” on page 297.

DDF and system profile monitoring

Db2 12 adds support for new counters for profile-related thread activity, the ability to request synchronous reads of IFCID 402 trace records, and the ability to control the queue depth for distributed threads.

Db2 12 adds new keyword columns to monitor connections and threads from unknown or unspecified IP addresses. This significant change allows you to control floods of requests and keep them from overwhelming your high-priority distributed workloads.

For more information, see “Distributed Data Facility” on page 297.

Subsystem monitoring

Db2 adds several enhancements to improve subsystem monitoring. For more information about the enhancements in this section, see “Db2 system topics” on page 294.

SYSPROC.ADMIN_INFO_IFCID

The **ADMIN_INFO_IFCID** stored procedure allows you to retrieve statistics information that is contained in Instrumentation Facility Component Identifiers (IFCIDs) 1, 2, or 225 with a **READS** call in a stored procedure.

STATIME_MAIN

The **STATIME_MAIN** subsystem parameter allows you to specify the interval in which Db2 writes key statistics records.

DSNV516I and DSNS005I for storage contraction

When Db2 issues message DSNV516I at the start of storage contraction, it now includes message DSNS005I with the reason for the contraction.

CICS attachment and client information special registers

Db2 12 adds support for fields that the CICS attachment facility can pass to Db2. Db2 uses the value of these fields as the default values for the CURRENT CLIENT_ACCTNG and CURRENT CLIENT_APPLNAME special registers.

DSNZPARM simplification

Db2 12 changes defaults for six subsystem parameters and removes 12 subsystem parameters to simplify the process of managing Db2.

For more information, see “Db2 system topics” on page 294.

STATGPSAMP

Db2 12 function level 505 sets the **STATPGSAMP** subsystem parameter so that the default for all **RUNSTATS** jobs is to use page sampling. If the **STATPGSAMP** default is not appropriate, you can set the subsystem parameter to **N0**, or you can override the individual jobs by using the **TABLESAMPLE SYSTEM** keyword specification. For more information, see “Db2 utilities” on page 277.

Security

Db2 12 provides new security capabilities for encryption and auditing and increases security for existing interfaces.

For more information about these encryption and auditing enhancements, see “Security” on page 300.

More secure Db2 CLP processing

When you invoke the Db2 command-line processor (CLP) to run an SQL file, you can now specify the **-u** option flag to specify a user ID and credentials, such as a password or RACF PassTicket. This option makes it easier than ever before to achieve an extra layer of security.

This option is available only in a z/OS UNIX System Services environment.

Secure socket layer-only connections

The Db2 TCP/IP communications environment can now restrict connections to use the secure socket layer (SSL) protocol only. The main subsystem **PORT** and **SECPOR**T can now be defined with identical values. **PORT** and **SECPOR**T can have the same values in both static location aliases and dynamic location aliases, for data sharing members, and in all DRDA connections, including REST. This enhancement ensures that all Db2 connections are secure connections.

SECADM and BINDAGENT

You can use the SECADM authority to grant the BINDAGENT privilege to the binding authid, which allows the grantee to bind and rebind plans and packages by specifying any owner. The SECADM granted BINDAGENT privilege will not allow the following actions:

- ▶ **REBIND PACKAGE** or **REBIND PLAN** without specifying the **OWNER** keyword
- ▶ **FREE PACKAGE** or **FREE PLAN**

- ▶ **COPY PACKAGE**
- ▶ **DROP PACKAGE**

You can grant the BINDAGENT privilege only to any binding authid if you have the SECADM authority; the BINDAGENT privilege cannot be granted to PUBLIC by using the SECADM authority.

Improved cross-memory address space separation

The new **DISALLOW_SSARAUTH** subsystem parameter determines whether user address spaces are blocked from setting a Db2 address space as a secondary address space.

If you set **DISALLOW_SSARAUTH** to NO, user address spaces are allowed to set a Db2 address space as a secondary address space. If you set **DISALLOW_SSARAUTH** to YES, user address spaces are blocked from setting a Db2 address space as a secondary address space. Setting the parameter to YES improves the security of your Db2 environment but can interfere with some tools. Check with your tool vendors before setting the parameter to YES.

RACF synchronization with Db2 security caches

For enhanced security, Db2 supports the type 71 ENF signal from RACF to revoke a user due to excessive invalid password attempts when **AUTHEXIT_CACHEREFRESH** is set to ALL. Db2 refreshes the cache entries that are associated with the revoked user as applicable.

Synergy with z15 and IBM DS8000 product families

Db2 12 environments and workloads can take advantage of several enhancements in the IBM Z and IBM DS8000® storage families.

IBM System Recovery Boost

System Recovery Boost can deploy extra general-purpose and zIIP processor capacity that is physically available on your system during recovery, initial program load (IPL), and shutdown situations. This capability allows you to minimize the outages that are experienced by your business applications.

For more information, see [System Recovery Boost for the IBM z15](#).

System Recovery Boost is also available on the IBM z16 with more functions, including middleware restart boost. You can elect for Db2 to use the middleware restart boost if you are running in a z16 environment.

Faster transport layer security with CryptoExpress 7S

You can improve security-related processing for your distributed workload by leveraging the CryptoExpress 7S (CEX7S) adapters on the IBM z15. Transport layer security (TLS) performs better and reduces application elapsed time if you deploy these adapters.

For more information about CEX7S, see the [CEX7S overview](#).

IBM Z data privacy: Excluding sensitive data in diagnostic information

You can use IBM Z Data Privacy For Diagnostics to produce diagnostic data that does not include sensitive fields. For example, you can send a Db2 dump to IBM Support without the dump including sensitive data in the Db2 buffer pools. This process increases security without slowing down the dump sending process itself.

For more information, see [IBM Z Data Privacy for Diagnostics](#).

IBM Db2 zHyperLink log write support

Db2 zHyperLink (zHL) is a hardware feature that uses a direct connection between the processor and the storage controller to avoid elapsed time that is associated with I/O operations and related interrupts. Db2 11 introduced support for using the zHL protocol for database reads.

Db2 12 can use the zHL protocol to complete the write of the Db2 log buffer without issuing an interrupt to the processor, which means the application continues running. This feature can decrease your application elapsed time. The zHL process is conceptually like coupling facility (CF) access.

To deploy zHL, you must have the appropriate storage controller (for example, IBM DS8000F), connectivity, and the Db2 **ZHYPERLINK** subsystem parameter set correctly. You can set the **ZHYPERLINK** parameter to the following values:

DISABLE	Db2 does not use zHyperLink for any I/O requests. This value is the default value.
ENABLE	Db2 requests the zHL protocol for all eligible requests.
DATABASE	Db2 requests the zHL protocol only for database synchronous I/Os.
ACTIVELOG	Db2 requests the zHL protocol only for active log write I/Os.

For more information about this subsystem parameter, see [Db2 zHyperLinks SCOPE field \(ZHYPERLINK subsystem parameter\)](#).

For more information about zHL, see [zHyperLink for the DS8880F overview](#).

You can use the IBM Z Batch Network Analyzer (zBNA) tool to process SMF 42-6 records to estimate the benefit of zHL in your environment. You can download the zBNA tool from [IBM Z Batch Network Analyzer \(zBNA\) Tool](#).

Activating V12R1M510: Considerations

You can activate function level V12R1M510 (FL 510) when your Db2 system is at any function level later than V12R1M100 after applying the new function APAR PH33727 (from April 2021). To determine whether your system is eligible for this activation, you can look at the **HIGHEST POSSIBLE FUNCTION LEVEL** value in the **-DISPLAY GROUP** command output. You can go straight to FL 510 with a single catalog migration followed by a single **ACTIVATE** command by completing the following steps:

1. Run the job **CATMAINT UPDATE CATALOG V12R1M510**.
2. Run the **-ACTIVATE FUNCTION LEVEL V12R1M510** command.

Running **CATMAINT** determines your current catalog level and makes whatever updates are necessary to achieve catalog level V12R1M509, which is the latest possible catalog level for Db2 12.

Issuing **ACTIVATE** is successful if there are no packages that are bound before Db2 11 that were run within the last 18 months.

If both steps are successful, Db2 is now at function level 510.

Though activating FL 510 is simple, you should take these steps with thought and planning.

Application impacts and continuous delivery

You should be able to progress through the stages of Db2 12 continuous delivery without impacting your applications and without scheduling application outages beyond the ones that are required for upgrade maintenance. For data sharing environments in which the applications tolerate or leverage data sharing, you can apply most maintenance without an application outage.

Catalog changes after Db2 12 FL 500 affect only a few catalog tables, and most applications should be able to run during the catalog update.

You can activate a function level while your applications run.

The only time that you must change your applications is to introduce new SQL syntax or behavior. In most cases, these types of changes require application code changes, which requires an outage.

Progressing to function level V12R1M510

When planning your migration, consider the following three categories of Db2 12 continuous delivery features:

- ▶ Features that were introduced in the maintenance stream that are non-function-level dependent
- ▶ Features that were introduced with function levels
- ▶ Features that were introduced with APPLCOMPAT levels

Each category presents unique planning characteristics.

Non-function-level-dependent features

Many of the features that are described in this appendix are active in your system already from applying regular maintenance. For example, data-sharing enhancements are independent of function levels. You can still avoid the effects of some of these features by not taking the actions that trigger the effect. For example, you can still control a new utility option that is not dependent on a function level by how you specify the option.

Db2 12 delivers over 100 features in this category. The potential impact of these features on your existing business workloads should be covered by standard testing as you promote maintenance through your environments to production.

Function-level-dependent features

You have control over when function-level-dependent features apply to your business applications based on when you issue the **ACTIVATE FUNCTION LEVEL** command. These features tend to have more significant impact on your environment but do not affect the SQL behavior of existing applications.

Most of these features do not take effect on activation of the corresponding function level. Rather, you must invoke the feature by taking a specific action. For example, **REBIND** phase-in is available but has no effect until you rebind a package. You can mitigate or avoid the impact of many of the function-level-dependent features. There are two ways to avoid the impact: by not invoking the triggering action, or by setting the related subsystem parameter to the value that represents earlier behavior.

With appropriate planning, you can activate FL 510 without a noticeable impact on existing business applications.

Application-compatibility-dependent features

You have even more control over features that depend on particular APPLCOMPAT levels. These features do not affect existing applications and packages unless some other action occurs. Even those features that have a potentially significant impact can be mitigated in most cases. For example, if you bound the packages that you use to issue DDL at APPLCOMPAT level V12R1M504, which affects creation of deprecated objects, and you must create such a deprecated object, you can set the CURRENT APPLICATION COMPATIBILITY special register to V12R1M503 or earlier to issue the specific DDL statement.

You do not need to change APPLCOMPAT levels for packages that run your business applications unless you want to deploy new features or behaviors. Db2 continues to support APPLCOMPAT levels of V10R1, V11R1, and all those levels in Db2 12.

For incompatible changes, you can use IFCID 376 to determine which application packages are incompatible. You can then schedule changes to address those incompatibilities independent of your activation of new function levels.

Planning to activate Db2 12 function level 510 (V12R1M510) in preparation for migration to Db2 13

Review the features that Db2 provides in each category: non-function-level dependent, function-level dependent, and application-compatibility dependent. Identify those features that you want to leverage. Identify which features might impact your existing applications so that you can plan for the introduction of those features into your environment.

Take advantage of the control Db2 continuous delivery provides as you plan to activate FL 510. You should find that these implementation steps have less impact on your normal business operations and do not require significant outages.

In addition to the information in [What's new in Db2](#), you might find it beneficial to read a technical paper that was written by IBM Gold Consultant Gareth Copplestone-Jones with input from other IBM Gold Consultants on the topic of Db2 12 continuous delivery. This paper covers the major features in each category and includes a limited survey of customer experiences, and can be found at [IBM Db2 12 for z/OS Function Level Activation and Management](#).

Summary

The Db2 12 continuous delivery process provides many features that you can deploy in your environment with minimal impact to your business applications. The last function level on Db2 12, V12R1M10, prepares your system for a smooth migration to Db2 13.



B

Application development

One of Db2 for z/OS strategic pillars is to extend modern application development capabilities within both Db2 and its ecosystem. In Db2 12, the extended capabilities include major enhancements to Db2 SQL and the introduction of Db2 REST services.

This appendix describes the following topics:

- ▶ SQL enhancements
- ▶ Db2 REST services

SQL enhancements

Each of the following SQL enhancements requires the activation of appropriate function levels and the binding of the package that runs the SQL with appropriate application compatibility (APPLCOMPAT) levels:

- ▶ Enhancement to temporal and archive transparency for **WHEN** clauses on triggers
- ▶ **CREATE OR REPLACE** syntax for stored procedures
- ▶ Application-level granularity for lock limits (NUMLKUS and NUMLKTS)

Other SQL enhancements, as listed below, are briefly described in “SQL enhancements” on page 234:

- ▶ **LISTAGG**
- ▶ Global variable for data replication override
- ▶ Syntax flexibility for special registers and **NULL** predicates
- ▶ Alternative names for built-in functions
- ▶ Referential integrity (RI) support for **UPDATE** and **DELETE** for parent tables in business-time temporal relationships

Enhancement to temporal and archive transparency for **WHEN** clauses on triggers

With Db2 12 function level 505 (FL 505), you can reference system-period temporal tables and archive-enabled tables in **WHEN** clauses of basic and advanced triggers regardless of the trigger definition for **SYSTEM_TIME SENSITIVE** or **ARCHIVE SENSITIVE** or the settings of the **SYSTIMESENSITIVE** or **ARCHIVESENSITIVE** bind options. Time machine (also called row versioning) and transparent archive data retrieval are fully supported in the **WHEN** clause. This enhancement requires that the package bind option for APPLCOMPAT is set at V12R1M505 or higher.

Triggers, temporal, and archive transparency

System-period temporal tables have corresponding history tables that contain the earlier state of the row when the row is changed in the base table. With this function, you can view the state of the row at various points in time as various application processes change the row. This capability is sometimes called *time machine support* or *row versioning support*.

Archive-enabled tables have corresponding archive tables into which rows that you delete from the base table can be inserted. With this function, you can make the archive-enabled table smaller and more efficient while still providing access to the archived rows.

When you create or alter an advanced trigger, you can specify **SYSTEM_TIME SENSITIVE YES** (default) for system-time temporal tables or **ARCHIVE SENSITIVE YES** (default) for archive-enabled tables. You can use **ALTER** for advanced triggers to change these values.

For basic triggers, you cannot specify system-period temporal or archive-enabled sensitivity at create trigger time. The default behavior is as though the option were specified **YES**. You can override the trigger definition for basic triggers during rebind with the bind option **SYSTIMESENSITIVE** or **ARCHIVESENSITIVE**, respectively.

WHEN clauses that reference temporal or transparent archive tables

Assume that you want to create a trigger with a **WHEN** clause that contains a query and the query references a system-period temporal table or archive-enabled table. Db2 processes the **WHEN** clause depending on the options that you specify when you create, alter, or rebind the trigger and when you bind or rebind the package. If you define the trigger with **SYSTEM_TIME SENSITIVE YES** or **ARCHIVE SENSITIVE YES** or specify the bind or rebind with **SYSTIMESENSITIVE YES** (for a system-period temporal table) or **ARCHIVESENSITIVE YES** (for an archive-enabled table), Db2 generates two sections for the **WHEN** clause at bind time. Only one of the sections contains a **UNION ALL** transformation to retrieve both current and historical or archived data. Then, Db2 chooses the section of the **WHEN** clause to run based on the **CURRENT TEMPORAL SYSTEM_TIME** special register (for a system-period temporal table) or the **SYSIBMADM.GET_ARCHIVE** global variable (for an archive-enabled table).

Before FL 505, you could not create, alter, or rebind a trigger package if system-time temporal tables or archive-enabled tables were referenced in a **WHEN** clause and if either the trigger definition or the bind options were specified YES for the corresponding system time or archive sensitivity. If you attempted to create, alter, bind, or rebind such a trigger before FL 505 or with the package APPLCOMPAT level set earlier than V12R1M505, you received SQL errors -270 and -20555, respectively. The time machine (or row versioning) and archive transparency features were restricted in trigger **WHEN** clauses before FL 505.

Figure B-1 illustrates the difference in what you can do with **WHEN** trigger clauses that reference system-period temporal tables or archive-enabled tables before and after activating FL 505 and binding the trigger package with APPLCOMPAT level V12R1M505.

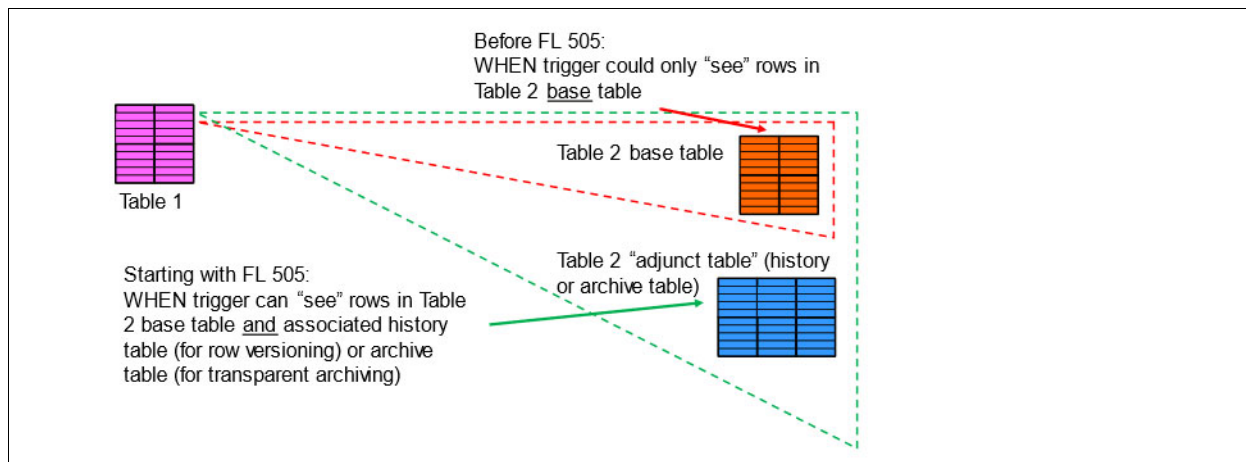


Figure B-1 WHEN trigger in FL 505

For more information, see [Temporal and archive transparency support for WHEN clause on triggers](#).

CREATE OR REPLACE syntax for stored procedures

Db2 can simplify the definition and maintenance of stored procedures by extending the existing **CREATE PROCEDURE** statement. In complex application environments, there might be a need to make schema changes to stored procedures. Instead of using an **ALTER** statement, it might be more efficient to modify the original **CREATE** statements and reissue them.

However, if a dependent procedure exists, it must be dropped before the execution of the modified **CREATE** statement. When doing so, you might encounter the following major issues:

- ▶ **Managing authorizations:** You must determine the authorizations that are granted on the dropped stored procedure to re-establish them for the re-created stored procedure.
- ▶ **Managing dependencies:** For a native SQL procedure, if there are other objects (procedures, functions, or triggers) that depend on the procedure that is dropped, the drop is restricted. You must first identify any dependent objects and drop them.

The time and effort to handle these issues on a repository that contains many procedures are unacceptable, particularly under an increasingly tight schedule.

To solve the issues, Db2 introduces a new **OR REPLACE** clause to enhance the following **CREATE PROCEDURE** statements:

- ▶ **CREATE PROCEDURE** (external)
- ▶ **CREATE PROCEDURE** (SQL - native)

To specify the **OR REPLACE** clause in an existing procedure, make sure that the new definition defines the same type of procedure (external or native SQL) as the existing one.

For native SQL procedures, you can replace an existing procedure or replace an existing version of the procedure or add a new version to the procedure. Assume that the following original statement created existing procedure MYPROC1:

```
CREATE PROCEDURE MYPROC1
  (IN P1 CHAR(5),
   OUT P2 DECIMAL(15,2) )
BEGIN
  SELECT AVG(SALARY) INTO P2
    FROM DSN8C10.EMP
    WHERE WORKDEPT = P1;
END
```

You can change the definition in MYPROC1 by reusing the original **CREATE PROCEDURE** statement and adding the **OR REPLACE** clause. The following example shows how you can add an **OR REPLACE** clause and issue the modified statement as a different query:

```
CREATE OR REPLACE PROCEDURE MYPROC1
  (IN P1 CHAR(5),
   OUT P2 DECIMAL (15,2) )
BEGIN
  SELECT AVG(SALARY + 1000) INTO P2
    FROM DSN8C10.EMP
    WHERE WORKDEPT = P1;
END
```

If the **VERSION** keyword is specified with a version identifier, you can add or replace a version of an existing native SQL procedure with the specified version identifier.

To add a new version to an existing native SQL procedure, specify the **OR REPLACE** clause and identify the new version ID after the **VERSION** keyword. The new version of the procedure is defined as though an **ALTER PROCEDURE** statement was issued with the **ADD VERSION** clause. If the procedure does not yet exist, it is created with the specified version ID for the first version of the procedure.

The statement in the following example adds a new version, V2, of the original procedure MYPROC1:

```
CREATE OR REPLACE PROCEDURE MYPROC1
  (IN P1 CHAR(5),
   OUT P2 DECIMAL (15,2))
VERSION V2          -- Identify the version to be added
BEGIN
  SELECT AVG(SALARY + 5000) INTO P2
    FROM DSN8C10.EMP
    WHERE WORKDEPT = P1;
END
```

To replace an existing version of a native SQL procedure, add the **OR REPLACE** clause and identify the ID of the version to be changed after the **VERSION** keyword. The version of the procedure is redefined as though an **ALTER PROCEDURE** statement was issued with the **REPLACE VERSION** clause. For example, the following statement replaces version V2 of procedure MYPROC1:

```
CREATE OR REPLACE PROCEDURE MYPROC1
  (IN P1 CHAR(5),
   OUT P2 DECIMAL (15,2))
VERSION V2          -- Identify the version to be replaced
BEGIN
  SELECT AVG(SALARY + 10000) INTO P2
    FROM DSN8C10.EMP
    WHERE WORKDEPT = P1;
END
```

For more information, see [CREATE PROCEDURE \(external\)](#) and [CREATE PROCEDURE \(SQL - native\)](#).

Application-level granularity for lock limits (NUMLKUS and NUMLKTS)

Db2 12 function level 507 (FL 507) provides greater granularity for locking limits. Locking limits default to the values that are specified in the subsystem parameters **NUMLKTS** for table spaces and **NUMLKUS** for users.

NUMLKTS sets the limit on the number of locks that an application can hold simultaneously in a table or table space. The optional **LOCKMAX** parameter on the **CREATE** or **ALTER TABLESPACE** statement sets the number of locks that an application process can hold simultaneously in the table space before lock escalation.

NUMLKUS specifies the maximum number of pages, row, or large object (LOB) locks that a single application can hold concurrently for all table spaces.

With FL 507, Db2 introduced two new built-in global variables to support application granularity for locking limits: **MAX_LOCKS_PER_TABLESPACE** and **MAX_LOCKS_PER_USER**. You can set these variables to provide specific thresholds for an application to have locking limit values that are different from the ones that are in the subsystem parameters **NUMLKTS** and **NUMLKUS**. Other applications can continue to use the subsystem parameters without being affected. The two global variables can help prevent table lock escalation that might have occurred with **NUMLKTS** previously, which mitigates resource timeouts and unavailability.

MAX_LOCKS_PER_TABLESPACE specifies the default maximum number of pages, row, or LOB locks that an application can simultaneously hold in a table space. **MAX_LOCKS_PER_USER** specifies the maximum number of pages, row, or LOB locks that a single application can concurrently hold for all table spaces. Both variables are type BIGINT and schema SYSIBMADM. You can set either or both global variables with system profile monitoring by including the SET statement in the ATTRIBUTE1 column of profile attributes table SYSIBM.DSN_PROFILE_ATTRIBUTES. The default values are derived from the settings in NUMLKTS and NUMLKUS, respectively.

You must bind or rebind packages that set **MAX_LOCKS_PER_TABLESPACE** or **MAX_LOCKS_PER_USER** with APPLCOMPAT of V12R1M507 or higher. For more information, see [Application granularity for locking limits \(NUMLKUS and NUMLKTS\)](#).

Assigning a value to either of the new built-in global variables to change the locking threshold for an application can have detrimental effects on other applications that run on the subsystem. Administrators may not allow applications to access the entire range of values for the SYSIBMADM.MAX_LOCKS_PER_TABLE_SPACE and SYSIBMADM.MAX_LOCKS_PER_USER built-in global variables. Instead of giving applications WRITE privileges on the built-in global variables, consider using SQL routines for fine-tuned control of the allowable value range. For example, consider using stored procedures like the following example to distribute a different range of values that is based on the SQL authorization ID:

```

CREATE PROCEDURE SET_TSLIMIT (IN LIMIT INT)
DISABLE DEBUG MODE
BEGIN
    DECLARE ADMF001_MAX INTEGER CONSTANT 8000;
    DECLARE ADMF002_MAX INTEGER CONSTANT 6000;
    DECLARE ADMF003_MAX INTEGER CONSTANT 2000;
    DECLARE ADMF00n_MIN INTEGER CONSTANT 1000;

    CASE CURRENT SQLID
        WHEN 'ADMFO01' THEN
            SET
SYSIBMADM.MAX_LOCKS_PER_TABLESPACE=MAX(MIN(LIMIT,ADMF001_MAX),ADMF00n_MIN);
        WHEN 'ADMFO02' THEN
            SET
SYSIBMADM.MAX_LOCKS_PER_TABLESPACE=MAX(MIN(LIMIT,ADMF002_MAX),ADMF00n_MIN);
        WHEN 'ADMFO03' THEN
            SET
SYSIBMADM.MAX_LOCKS_PER_TABLESPACE=MAX(MIN(LIMIT,ADMF003_MAX),ADMF00n_MIN);
    ELSE
        SIGNAL SQLSTATE '75002'
        SET MESSAGE_TEXT='UNSUPPORTED USER';
    END CASE;
END#
GRANT EXECUTE ON PROCEDURE SET_TSLIMIT TO ADMF001, ADMF002, ADMF003#

```

Db2 REST services

With Db2 Representational State Transfer (Db2 REST) services, an authorized user may access Db2 data with a REST HTTP client that sends JavaScript Object Notation (JSON) data as input and receives JSON data as output. This support enables mobile, web, and cloud applications to interact with existing Db2 data without issuing SQL.

Db2 REST services can also be invoked by using IBM z/OS Connect APIs, which extend the value of Db2 REST services in combination with other z/OS based resources.

For more information about enabling Db2 REST services support, see [Db2 REST services](#).

Initial Db2 native REST support

REST is an architectural style that enables service invocation through a straightforward, consistent interface. Db2 12 for z/OS introduced its support for REST services with PI70652.

Db2 REST services leverage existing Distributed Data Facility (DDF) capabilities, including authorization, authentication, client information management, service classification, system profiling, and service monitoring and display. Figure B-2 shows the workflow when a mobile application accesses Db2 data by invoking the Db2 native REST support.

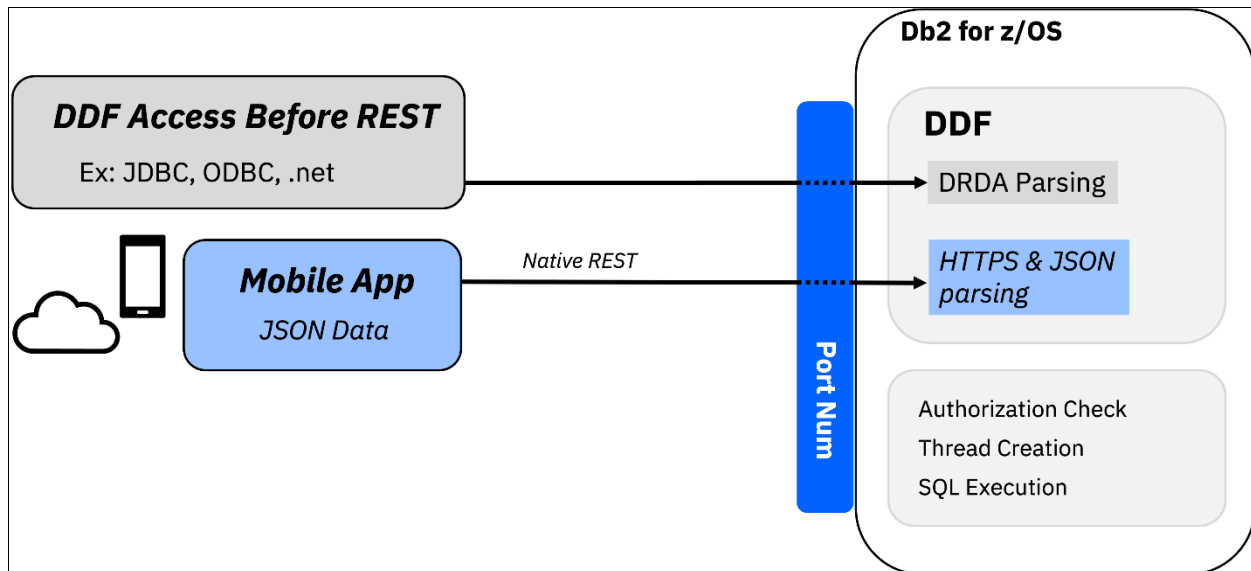


Figure B-2 Db2 native REST to DDF architecture

The initial Db2 native REST support included the functions for creating, discovering, invoking, and deleting REST services. Each REST service is a static package that consists of a single SQL statement, which may be a **CALL** to a stored procedure. Db2-supplied services are `DB2ServiceManager` and `DB2ServiceDiscover`, and user-defined services are added to `SYSIBM.SYSSERVICES` as they are created.

You can use one of the following SQL statements in a Db2 REST Service: **CALL**, **DELETE**, **INSERT**, **SELECT**, **TRUNCATE**, **UPDATE**, and **WITH**.

Note: You may use **MERGE** in a Db2 REST Service if the **MERGE** statement is part of the stored procedure that the REST service invokes. **MERGE** cannot be the single SQL statement in a Db2 REST Service.

Db2 REST services support all base data types, and BLOB, CLOB, DBCLOB, and XML data types.

Creating, discovering, invoking, and deleting REST services

You can create Db2 REST services by using a browser plug-in, a REST application, or the **BIND SERVICE** subcommand. REST browser plug-ins and REST applications generally have a similar user interface. You can issue the **BIND SERVICE** or **FREE SERVICE** subcommand from a program or session that you would normally use to interact with Db2.

Although you can use the interface that you prefer, the **BIND SERVICE** subcommand, REST application, or REST browser plug-in to create or delete (**FREE**) REST services in Db2, you must use a REST application or REST browser plug-in to discover or invoke REST services. Your applications access Db2 REST services through uniform resource identifiers (URIs) with an applicable HTTP method. Db2 native REST services invocation supports only the **POST** method for applications. The **POST** method applies to any Db2 REST Service whether the SQL statement that comprises the service is **CALL**, **SELECT**, **INSERT**, **UPDATE**, **DELETE**, **TRUNCATE**, or **WITH**.

With the appropriate authorization or privileges, you can discover all existing REST services by issuing an HTTP **POST** request. The requesting URI must define a “discover all services” request, as shown in the following example:

```
POST https://<host>:<port>/services/Db2ServiceDiscover
```

Alternately, you can also discover all REST services by using the HTTP **GET** method. For example, you can discover all existing REST services with the **GET** method in the following URI:

```
GET https://<host>:<port>/services
```

Using a browser plug-in or a REST application to create a service is like the process for discovering one. With the correct authorization and authority, you may issue a **POST** request through a REST client. The **POST** request includes a manager request in the URI and specifies all the details in the HTTP body of the request, as shown in the following example:

```
POST https://<host>:<port>/services/Db2ServiceManager
HTTP request body:
{ "requestType": "createService",
  "sqlStmt": "<sqlStatement>",
  "collectionID": "<serviceCollectionID>",
  "serviceName": "<serviceName>",
  "description": "<serviceDescription>",
  "bindOption": "<bindOption>"}
```

For more information about REST services, see [Db2 REST services](#).

Native REST services and RESTful APIs with Db2 and z/OS Connect

Db2 native REST services limit the HTTP methods that you can use. If you want your REST applications to use **GET**, **PUT**, **POST**, and **DELETE** for the corresponding services, you can use IBM z/OS Connect, which supports those HTTP methods. When defining an API that uses z/OS Connect, you can map the API to the appropriate Db2 REST Service so that your application developers can use the HTTP methods with which they are familiar.

z/OS Connect provides a single point of entry for z/OS resource access through RESTful APIs from the cloud or the web. Instead of disparate access points and protocols, z/OS Connect provides a common, secure entry point for all REST HTTP calls to reach the requested business assets and data on z/OS systems. It runs on a z/OS LPAR and maps the APIs with REST calls to copybooks and programs of z/OS assets.

The two main components of z/OS Connect are the runtime server and the tool platform. The runtime server hosts the APIs that you define to run, connects to the back-end system, and allows for multiple instances. The tool platform is an environment that is based on Eclipse where you can define APIs, create data mapping, and deploy the APIs to the runtime server.

With IBM z/OS Connect, a Db2 **POST** method with an SQL statement is mapped to the appropriate REST method, which allows for full RESTful implementation.

For more information about z/OS Connect, see [IBM z/OS Connect documentation](#).

SERVICE versioning

SERVICE versioning refers to the ability of Db2 REST services to develop and deploy a new version of a REST service while the existing versions are still being used. Db2 REST versioning is based on Db2 package versioning support. A new version of a REST service can be introduced without having to change existing authorizations.

You can enable SERVICE versioning by applying APAR PI98649 and running sample job DSNTIJR2. After you enable versioning, do not remove APAR PI98649. Otherwise, the entire REST services function is unavailable.

The SERVICE versioning option does not impact the existing REST services that do not specify a version. The version-less REST services have an empty string value as their version ID. The services that are created after you enable versioning always will be versioned.

You can specify the REST service version ID as part of a REST request. If no version ID is specified, the default service version is invoked.

You can use the **REBIND PACKAGE** subcommand to change the default version of the REST service.

COPY options for promoting Db2 REST services

Db2 REST services are resources that you must manage across your Db2 environment and locations. The **BIND SERVICE** subcommand provides the flexibility for managing REST services, including keywords **COPY**, **COPYVER**, and **OPTIONS**. These keywords of the **BIND SERVICE** subcommand are available for versioned REST services.

Remote issuance of the **FREE SERVICE** subcommand

To further ease management of Db2 REST services, you can use the **FREE SERVICE** subcommand to specify a remote location as the target of **FREE SERVICE**.

Summary

Db2 12 provides rich functions for SQL capabilities and REST services to improve your ability to deploy modern applications. For more information, see [What's new in Db2 12](#).



Database administration

The key Db2 12 continuous delivery features that relate to database administration processes are **REBIND** and package management enhancements, Data Definition Language (DDL) changes, and enhancements to Db2 utilities in the areas of availability, performance, simplification, and usability.

This appendix describes the following topics:

- ▶ REBIND and package management
- ▶ Data Definition Language
- ▶ Db2 utilities

REBIND and package management

The most significant Db2 12 for z/OS continuous delivery feature for **BIND** and **REBIND** is the ability to create a new copy of a package while existing threads continue to use the previous copy of the package. This enhancement improves DBA productivity and application availability. You no longer need to wait for applications to stop before rebinding the packages that the applications use. You can react quickly to situations that call for package rebinds.

Db2 12 provides several other significant changes to package management, including enhancements to **FREE PACKAGE** and **APREUSE**, more options for database request module (DBRM) management, improved automatic rebind (autobind), and clarification of package versioning and ownership. These changes are described briefly in Appendix A, “IBM Db2 12 continuous delivery features and enhancements” on page 233.

REBIND phase-in

With **REBIND** phase-in, you can rebind a package while that package is currently running. This feature increases your ability to react to situations where rebinding a package is necessary, such as to change bind attributes or to improve access paths. You do not need to find a time when the target package is unused or resort to complex workarounds to force the package rebind.

Starting with function level 505 (FL 505), if you issue a **REBIND** for a package that is currently running, Db2 creates a copy of the package. Existing threads continue to use the existing package copy, which becomes the phased-out copy. When the rebind operation commits the new copy, that copy becomes the current copy and is immediately available for new threads to use. A phased-out copy is removed after the last thread that uses the phased-out copy releases it.

As shown in Figure C-1, two threads are running by using the current package copy, which has copy ID 0. When you issue **REBIND** for that package, Db2 creates copy ID 4. If a new thread (Thread 3) runs the same package during the rebind process, it also uses copy ID 0. When the rebind process commits, copy ID 4 becomes the current copy and subsequent threads, such as Thread 4, use copy ID 4. Copy ID 0 is marked as phased-out. After Thread 1, Thread 2, and Thread 3 release the package, a subsequent rebind deletes copy ID 0.

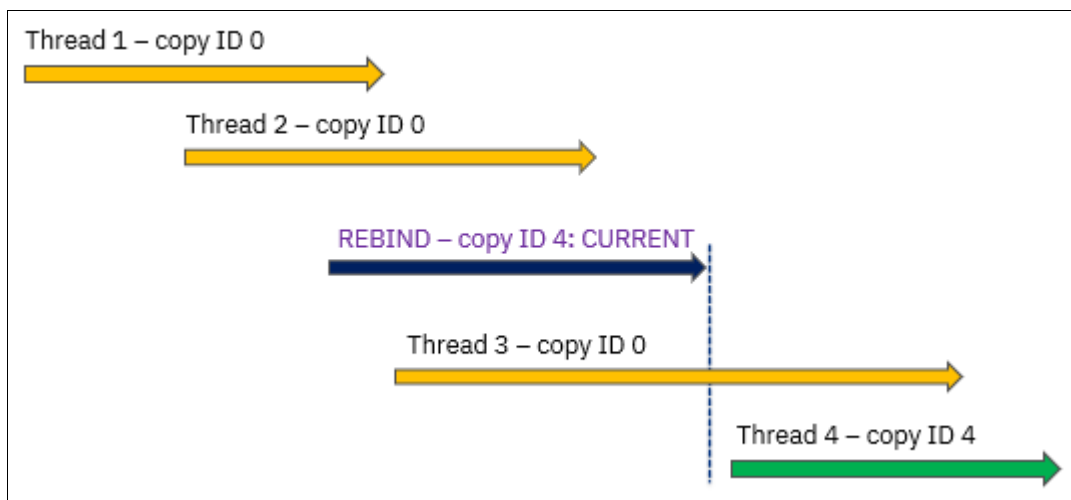


Figure C-1 REBIND phase-in example

Db2 keeps COPYID information in the SYSIBM.SYSPACKAGE and SYSIBM.SYSPACKCOPY tables, as shown in Table C-1.

Table C-1 Package copies

COPYID column	In SYSIBM...	Description
0, 4, 5, ...16	SYSPACKAGE	CURRENT. Used in wrap-around mode.
1	SYSPACKCOPY	PREVIOUS.
2	SYSPACKCOPY	ORIGINAL.
3	-	Reserved.
0, 4, 5, ...16	SYSPACKCOPY	Phase-out copies until deleted.

Db2 replicates the phased-out copy into PREVIOUS or ORIGINAL copies during the rebind process as necessary.

If you try to rebind a package and receive a DSNT500I message with reason code 00E30307, all the available copy IDs are in use. You must determine which application thread is blocking Db2 from freeing the phased-out copy. Start by looking at IFCID 393, which shows you which application thread must be recycled. To take advantage of IFCID 393, be sure to turn on it before issuing **REBIND**. You can also use a package accounting trace to show which package copy of a thread is running.

REBIND SWITCH phase-in

You can use **REBIND SWITCH** to copy the PREVIOUS or ORIGINAL copy to the new CURRENT copy by using the same process that is described for REBIND phase-in. Existing threads continue to run during the **REBIND SWITCH** process, and new threads use the new package copy after **REBIND SWITCH** completes.

REBIND phase-in eligibility

REBIND phase-in is supported in the following cases:

- ▶ **APREUSE(NONE) PLANMGMT(EXTENDED)**.
- ▶ **APREUSE(WARN) PLANMGMT(EXTENDED) APREUSESOURCE(CURRENT)**.
- ▶ **APREUSE(ERROR) PLANMGMT(EXTENDED) APREUSESOURCE(CURRENT)**.
- ▶ The package is not a generated package for a trigger or SQL routine, such as a procedure or user-defined function (UDF).

No dependency on the application compatibility level

REBIND phase-in is an “always on” feature beginning with the activation of FL 505. There is no dependency on the setting of the package application compatibility (APPLCOMPAT) level.

FREE PACKAGE

Unused phased-out copies might stay in SYSPACKCOPY for some time. You can use the following **FREE PACKAGE** commands to free eligible phased-out copies, which reduce the size of SYSPACKCOPY:

- ▶ **FREE PACKAGE PLANMGMTSCOPE(PHASEOUT)**
- ▶ **FREE PACKAGE PLANMGMTSCOPE(INACTIVE)**

For more information about the **FREE PACKAGE** syntax, see [FREE PACKAGE \(DSN\)](#).

EXPLAIN

When you issue the **EXPLAIN PACKAGE COPY** statement, you can include the following copy-id specifications:

- ▶ **CURRENT** performs an **EXPLAIN** that is based on the copy-id in the SYSPACKAGE.COPYID column.
- ▶ **PREVIOUS** performs an **EXPLAIN** that is based on copy-id 1 in the SYSPACKCOPY table.
- ▶ **ORIGINAL** performs an **EXPLAIN** that is based on copy-id 2 in the SYSPACKCOPY table.

When you issue the **EXPLAIN PACKAGE** statement without the **COPY** keyword, the output includes **EXPLAIN** for the **CURRENT**, **PREVIOUS**, and **ORIGINAL** copies.

REBIND phase-in best practices

Adhering to the following best practices can help ensure that you are using **REBIND** phase-in effectively:

- ▶ Ensure that long-running threads that are bound with **RELEASE (DEALLOCATE)** can take advantage of **REBIND** phase-in:
 - Evaluate your use of high-performance Database Access Threads (HIPERF DBATs). An active HIPERF DBAT is recycled only after 200 units of work or 120 seconds without activity (or the setting of the **POOLINAC** subsystem parameter), which can result in a long delay before the new package copy is allocated.

Note: These values changed in Db2 13.

- Avoid pooling of HIPERF DBATs because pooled DBATs continue to run the package copies that were originally allocated to them. Use the following command to allow HIPERF DBATs that are not pooled at connection termination:
-MODIFY DDF, PKGREL(BINDOPT)
- Set the reuse limit for IBM CICS protected threads to the default (REUSELIMIT = 1000) to ensure that CICS transaction programs quickly start to run the new package copy.
- Instead of Information Management System (IMS) Fast Path, use IMS pseudo wait-for-input (pseudo WIFI) regions to increase the chances of running the most current package copy. IMS Fast Path WIFI region threads can remain allocated for days or weeks.
- ▶ APAR PH28693 improves **REBIND** phase-in and should be applied to production environments before using this feature. The improvements that are provided by this APAR reduce the likelihood of timeouts when newly arriving threads request the phased-out package copy before the rebind operation completes.
- ▶ Use the **FREE PACKAGE** command to remove unused package copies from the SYSPACKCOPY.table.
- ▶ Thoroughly test the behavior of **REBIND** phase-in before activating FL 505 in your production environment.

Data Definition Language

The major changes to DDL support Huffman compression, encourage adoption of universal table space (UTSs) by restricting the creation of non-UTS objects, and add the **MOVE TABLE** option to the **ALTER TABLESPACE** statement, which facilitates converting multi-table table spaces to partition-by-growth (PBG) UTS objects:

- ▶ Huffman compression
- ▶ Preventing new deprecated objects
- ▶ Migrating multi-table table spaces to PBG UTS

Other DDL enhancements include implicit drop of explicitly created table spaces, implicit hidden ROWID columns, meta-data self-description, support of DECFLOAT columns in indexes, and the ability to rotate partitions for materialized query tables. For more information about these enhancements, see “Data Definition Language” on page 240.

Huffman compression

Db2 12 introduces support for Huffman compression in function level 504 (FL 504) and extends that support in FL 509. Huffman compression gives you another option to consider when compressing your data.

Db2 for z/OS compression

Db2 for z/OS first supported compression by using an edit procedure (EDITPROC). This type of compression relied on software compression algorithms and was relatively expensive in terms of processing impact.

Starting in Db2 3, Db2 supported dictionary-based table space compression. The compression dictionary has fixed-length entries, so this type of compression is called fixed-length compression. Db2 table space compression uses a special hardware co-processor, CMPSC, which was originally a separate hardware feature but now is integrated into the processor in recent IBM Z models. Many installations use table space compression and CMPSC, which saves significant disk storage for a relatively small cost in processing. Pages of compressed table spaces remain compressed in the buffer pool, potentially increasing buffer pool hit ratios and reducing the cost and elapsed time of I/O. For some low-write, high-read workloads, compression can reduce processing cost compared to accessing uncompressed table spaces.

Db2 supports index compression by using software algorithms for indexes that are defined in page sizes larger than 4 K. If you compress indexes, the compressed page is always 4 K in size. Index pages are always decompressed when they are in their buffer pool. The benefit of index compression is disk savings.

Db2 12 introduced support for large object (LOB) compression in function level 500 (FL 500). Db2 LOB compression is not dictionary-based, and it requires a separately priced software feature in addition to hardware support for IBM Z Enterprise Data Compression (zEDC). The hardware support for zEDC in z14 processors requires a separately priced feature card. In IBM z15 and later, zEDC support is included on the processor; the software feature code is still required.

For more information about Db2 compression, see [Compressing your data](#).

Huffman compression overview

Huffman compression uses a variable-length dictionary entry to provide further efficiencies for compressed table spaces. When a Huffman compression dictionary is created, the data is compressed like it is for fixed-length compression, and then the shortest codes are assigned to the most frequently occurring values. In many cases, this approach results in higher compression ratios with comparable processor costs.

Huffman compression in FL 504

Db2 12 FL 504 introduced initial support for Huffman compression for UTSs. The choice of deploying Huffman compression is based on the `TS_COMPRESSION_TYPE` subsystem parameter, which can be set to `HUFFMAN` or to `FIXED LENGTH` (the default). If `TS_COMPRESSION_TYPE` is set to `HUFFMAN`, the next time a dictionary is created or built, it is a Huffman dictionary. In FL 504, the level of granularity for specifying the compression algorithm is the subsystem or member level.

Huffman compression in FL 509

Db2 12 FL 509 allows you to choose the compression algorithm at the table space or partition level. You can use the `CREATE` or `ALTER TABLESPACE` statement with the `COMPRESS YES` clause to specify the compression algorithm. You can also use the `CREATE TABLE` statement with `COMPRESS YES` for implicitly created table spaces.

Figure C-2 provides an excerpt of the `COMPRESS` clause that you can specify at the table space or partition level for UTSs.

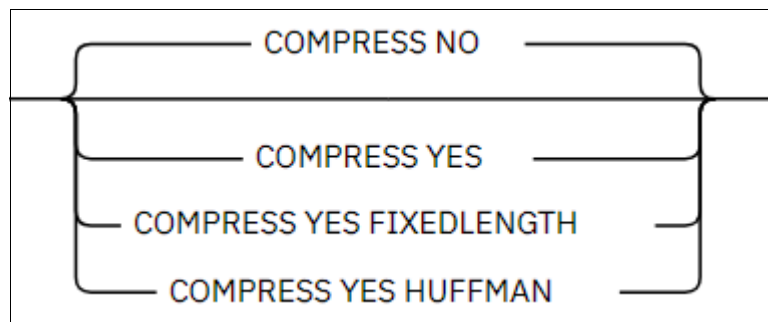


Figure C-2 Excerpt of `CREATE` or `ALTER` showing the `COMPRESS` clause

Building a compression dictionary

Table space compression, whether fixed-length or Huffman, requires the table space definition to specify `COMPRESS YES`. You can populate `COMPRESS YES` table spaces with data and build the compression dictionary in any of the following ways:

- ▶ Run the `LOAD` utility with `REPLACE`, `RESUME NO` and without `KEEPDICTIONARY`.
- ▶ Run the `LOAD` utility with `RESUME YES SHRLEVEL CHANGE` and without `KEEPDICTIONARY`.
- ▶ Run the `REORG` utility without `KEEPDICTIONARY`.
- ▶ Issue `INSERT` statements.
- ▶ Issue `MERGE` statements.

To use any of these methods to build a Huffman compression dictionary, the following requirements must be met; otherwise, a fixed-length dictionary is built:

- ▶ The table space must be a UTS.
- ▶ The table space is not the directory table space (SPT01).
- ▶ The table space is not defined with `ORGANIZE BY HASH`.

- ▶ The Db2 subsystem or member that creates the dictionary runs on an IBM z14 or later system.
- ▶ The current function level is FL 504 or higher.
- ▶ For the specification of the compression algorithm through a **CREATE** or **ALTER** statement, the current function level must be FL 509 or higher, and the APPLCOMPAT level in effect for the **CREATE** or **ALTER** statement must be V12R1M509 or higher.

Catalog changes for Huffman compression

Catalog level V12R1M509 adds a column and changes two columns in the Db2 catalog:

- ▶ **SYSIBM.SYSTABLEPART**:
 - **COMPRESS_USED** (new) with values F, H, ' ', or NULL
 - **COMPRESS** (changed) with values Y, F, H, or ' '
- ▶ **SYSIBM.SYSTABLESPACE**
 - COMPRESS** (changed) with values Y, F, H, ' ', or NULL

This catalog level is required to activate FL 509 and gives you visibility into the compression algorithm at the table space or partition level.

These V12R1M509 catalog changes are supported by compression-on-insert processes and by these utilities:

- ▶ **REORG TABLESPACE**
- ▶ **LOAD SHRLEVEL NONE** or **SHRLEVEL REFERENCE**
- ▶ **RUNSTATS**
- ▶ **RECOVER** to a point-in-time (PIT)
- ▶ **REPAIR CATALOG**

DSN1COMP

The **DSN1COMP** stand-alone utility estimates space savings that can potentially be achieved by data compression in table spaces, including LOB table spaces and indexes. You can estimate the space savings for fixed-length compression, Huffman compression, or both.

You can run **DSN1COMP** on the following types of data sets:

- ▶ Db2 full image copy data sets
- ▶ Virtual Storage Access Method (VSAM) data sets that contain Db2 table spaces
- ▶ Sequential data sets that contain Db2 tables spaces (for example, **DSN1COPY** output)

These data sets can contain compressed or uncompressed data. For example, you can use **DSN1COMP** on a table space partition that already is compressed with fixed-length compression to determine whether you might experience greater space savings by using Huffman compression.

Preventing new deprecated objects

Many items were deprecated since Db2 10 for z/OS, but the creation and use of these deprecated objects continue to be supported to allow time for converting to newer alternatives with enhanced functions, such as partition-by-growth (PBG) UTS or partition-by-range (PBR) UTS. However, to deter the use of deprecated index-controlled range partitioning, you could have used the **PREVENT_NEW_IXCTRL_PART** subsystem parameter, but a general method did not exist for preventing the creation and proliferation of new deprecated objects.

After activating function level V12R1M504, you can control and prevent the creation of the following deprecated objects in your Db2 environment:

- ▶ Synonyms
- ▶ Non-UTS table spaces, including segmented non-UTS (“classic” segmented) and partitioned non-UTS (“classic” partitioned)
- ▶ Hash-organized tables and the associated hash table spaces

Table C-2 summarizes these changes.

Table C-2 Deprecated objects when APPLCOMPAT level is later than V12R1M504

This DDL:	Results in:
CREATE TABLESPACE ... SEGSIZE <i>n</i>	A PBG UTS is created (instead of segmented non-UTS).
CREATE TABLESPACE ... NUMPARTS <i>p</i> (DPSEGSZ subsystem parameter = 0)	A PBR UTS is created (instead of partitioned non-UTS).
CREATE TABLESPACE ... NUMPARTS <i>p</i> (DPSEGSZ subsystem parameter = nonzero)	A PBR UTS is created (no change).
CREATE TABLESPACE ... SEGSIZE 0 NUMPARTS <i>p</i>	An error occurs.
CREATE TABLESPACE ... SEGSIZE <i>n</i> NUMPARTS <i>p</i>	PBR UTS is created (no change).
CREATE TABLESPACE ... SEGSIZE <i>n</i> MAXPARTITIONS <i>p</i>	PBG UTS is created (no change).
CREATE SYNONYM	An error occurs. Use ALIAS instead.
CREATE / ALTER TABLESPACE ... ORGANIZE BY HASH	An error occurs.
CREATE TABLE referencing an empty simple, segmented non-UTS or partitioned non-UTS table space	An error occurs. Create a table in PBG or PBR UTS instead.

These changes apply only when SQL DDL statements run at APPLCOMPAT level (APPLCOMPAT) V12R1M504 or later. You can still create the deprecated objects at any Db2 function level where the DDL statements run with APPLCOMPAT V12R1M503 or earlier.

For more information about how you can control APPLCOMPAT, see [APPLCOMPAT bind option](#) or [CURRENT APPLICATION COMPATIBILITY](#).

Also, if you have not yet activated function level V12R1M504, all your packages are still bound at APPLCOMPAT V12R1M503 or earlier, so you do not need to rebind any packages at a higher APPLCOMPAT until you are ready for the applications to start using the features of a later function level. So, activating function level V12R1M504 by itself has no immediate impact on your ability to create deprecated objects.

More specifically, the following changes in functions take effect in APPLCOMPAT level V12R1M504 or later:

- ▶ Synonyms
- ▶ Simple table spaces
- ▶ Segmented non-UTS table spaces
- ▶ Partitioned (non-UTS) table spaces
- ▶ Hash-organized tables
- ▶ DSNACCOR stored procedure
- ▶ Work file table space creation and separation
- ▶ Deprecated objects can still be created and are supported

Synonyms

Synonyms are still supported but can no longer be created. A **CREATE SYNONYM** statement results in an error. Consider using aliases instead.

Simple table spaces

Simple table spaces are still supported, but new tables can no longer be created in a simple table space. A **CREATE TABLE** statement that specifies a simple table space results in an error. Consider creating tables in their own individual UTSs instead.

Segmented non-UTS table spaces

Segmented (non-UTS) table spaces are still supported but are no longer created by default with a **CREATE TABLESPACE** statement that does not explicitly indicate what table space type is wanted, and new tables can no longer be created in a segmented (non-UTS) table space.

For reference, before FL 504, the syntax for creating or defaulting to a segmented table space was:

- ▶ **CREATE TABLESPACE ... SEGSIZE *n*** (without **MAXPARTITIONS** or **NUMPARTS** specifications)
- ▶ **CREATE TABLESPACE ...** (without **MAXPARTITIONS**, **NUMPARTS**, or **SEGSIZE** specifications)

This syntax, in which a **CREATE TABLESPACE** statement does not specify the **MAXPARTITIONS** or **NUMPARTS** clauses, results in the creation of a PBG table space by default instead of a segmented (non-UTS) table space starting at the target APPLCOMPAT level. The value of **MAXPARTITIONS** is the same as the default **MAXPARTITIONS** value for implicitly created PBG table spaces. The value of **SEGSIZE** is determined by the explicit **SEGSIZE** clause, or Db2 uses the **DPSEGSZ** subsystem parameter value. (For more information about the **DPSEGSZ** subsystem parameter, see [DEFAULT PARTITION SEGSIZE field \(DPSEGSZ subsystem parameter\)](#).) Any other table space attributes that are not specified are set to the normal default values that are indicated in the **CREATE TABLESPACE** statement description.

Starting in APPLCOMPAT level V12R1M504, syntax that can specify the creation of a segmented (non-UTS) table space is not available. The syntax that would have resulted in a segmented (non-UTS) table space before APPLCOMPAT level V12R1M504 now results in a PBG table space.

Because a segmented (non-UTS) table space can no longer be created through the **CREATE TABLESPACE** statement, the **LOCKSIZE TABLE** specification is no longer valid because **LOCKSIZE TABLE** can be specified only for a segmented (non-UTS) table space. **LOCKSIZE TABLE** was removed from the **CREATE TABLESPACE** syntax diagram and description.

However, for compatibility with earlier versions where **CREATE TABLESPACE** does not have an explicit UTS specification (no **MAXPARTITIONS** or **NUMPARTS** is specified), Db2 treats **LOCKSIZE TABLE** as a synonym for **LOCKSIZE TABLESPACE** instead of issuing an error. For example, the following statement (without a **MAXPARTITIONS** or **NUMPARTS** specification) results in a PBG UTS with **LOCKSIZE TABLESPACE** in the target APPLCOMPAT level:

```
CREATE TABLESPACE ... LOCKSIZE TABLE
```

In APPLCOMPAT level V12R1M503 or earlier, Db2 issues an error for the following statements, and continues to issue this error even in APPLCOMPAT levels V12R1M504 or later:

- ▶ **CREATE TABLESPACE ... LOCKSIZE TABLE MAXPARTITIONS *n***
- ▶ **CREATE TABLESPACE ... LOCKSIZE TABLE NUMPARTS *n***
- ▶ **ALTER TABLESPACE ... LOCKSIZE TABLE** (where the table space is a UTS)

ALTER TABLESPACE with the **LOCKSIZE TABLE** specification continues to be supported if the table space being altered is a segmented (non-UTS) table space.

Additionally, a **CREATE TABLE** statement to create a table in a segmented (non-UTS) table space results in an error.

Partitioned (non-UTS) table spaces

Partitioned (non-UTS) table spaces are still supported but can no longer be created, and a new table can no longer be created in a partitioned (non-UTS) table space.

For reference, here is the syntax for creating or defaulting to a partitioned (non-UTS) table space:

- ▶ **CREATE TABLESPACE ... SEGSIZE 0 ... NUMPARTS *n*** (without the **MAXPARTITIONS** specification)
- ▶ **CREATE TABLESPACE ... NUMPARTS *n*** (without the **MAXPARTITIONS** specification, and with **DPSEGSZ = 0**)

A **CREATE TABLESPACE** statement that specifies **SEGSIZE 0** and **NUMPARTS** results in an error.

A **CREATE TABLESPACE** statement that specifies **NUMPARTS** but does not specify the **MAXPARTITIONS** and **SEGSIZE** clauses results in the creation of a PBR UTS table space by default, even if the **DPSEGSZ** subsystem parameter is set to 0.

The value of **SEGSIZE** is determined by the explicit **SEGSIZE** clause or, if a **SEGSIZE** clause is not specified, by the value of the **DPSEGSZ** subsystem parameter value. Any other table space attributes that are not specified are set to the normal default values that are indicated in the **CREATE TABLESPACE** statement description. For more information about the **DPSEGSZ** subsystem parameter, see [DEFAULT PARTITION SEGSIZE field \(DPSEGSZ subsystem parameter\)](#).

Additionally, a **CREATE TABLE** statement to create a table in a partitioned (non-UTS) table space results in an error. Consider creating PBR UTSs instead.

Hash-organized tables

Hash-organized tables are still supported but can no longer be created. A **CREATE TABLE** or **ALTER TABLE** statement that specifies **ORGANIZE BY HASH** results in an error.

Hash-organized tables provide improved performance for a specific set of use cases. However, Db2 10 introduced index-processing enhancements and newer capabilities such as fast index traversal in Db2 12 to provide more benefits. With these improvements, the niche where hash-organized tables outperform standard UTS tables is extremely small. With fast index traversal, Db2 monitors the indexes and decides whether to use it autonomically.

DSNACCOR stored procedure

The Db2 12 installation and migration processes were modified to discontinue creating and configuring **DSNACCOR** without regard to the function level or APPLCOMPAT level. Existing instances are not removed, but new ones are no longer created. Consider using **DSNACCOX** instead.

Work file table space creation and separation

In APPLCOMPAT level V12R1M503 or lower, you can create either segmented (non-UTS) or PBG UTSs in the work file database.

Sort activity and declared global temporary tables can use either type of table space in the work file database, but determination of which table space to use for which activity depends on the setting of the **WFDBSEP** subsystem parameter.

Starting in APPLCOMPAT level V12R1M504, segmented (non-UTS) table spaces can no longer be created, including in the work file database. Only PBG UTSs can be created in the work file database. A **CREATE TABLESPACE** statement that does not have an explicit **MAXPARTITIONS** specification no longer results in a segmented (non-UTS) table space. Instead, a PBG UTS is created with the following default values:

- ▶ **MAXPARTITIONS** 256
- ▶ **DSSIZE** 4GB
- ▶ **SEGSIZE** 16

For more information about the default values of other options, see the full syntax for [CREATE TABLESPACE](#).

Deprecated objects can still be created and are supported

Considerable time and effort are required to convert deprecated objects. Many vendor tools, and even some from IBM, still create and use deprecated objects. The Db2 catalog, directory, and other system objects still use deprecated objects. Clearly, more work must be done before the code that supports deprecated objects can be fully removed.

For these reasons, deprecated objects remain supported in Db2 12 and Db2 13. You can still create the objects by controlling the APPLCOMPAT level, and Db2 13 supports all Db2 12 APPLCOMPAT levels. For more information, see [Creating non-UTS table spaces \(deprecated\)](#).

Also, **ALTER** is fully supported at every APPLCOMPAT level for all existing objects, except when the alteration can result in the creation of a deprecated object.

Nevertheless, it is a best practice that you adopt the use of strategic object types such as UTS and that you develop plans for migrating your existing deprecated objects to one of these UTS types. These latest UTS types offer many significant benefits (online schema evolution support, flexible size partitions, and so on), and the deprecated objects will not be supported forever. For more information about the advantages of UTSs, see [Table space types and characteristics in Db2 for z/OS](#).

Migrating multi-table table spaces to PBG UTS

UTSs are the strategic table space type and have new functions that are not available in multi-table table spaces, which are deprecated. To take advantage of the new functions and move away from deprecated table spaces, you must convert multi-table table spaces to UTSs.

Previously, this process involved unloading data, dropping tables, re-creating tables in UTSs, re-creating associated objects, re-granting privileges, and reloading data. This conversion process resulted in an application outage and was not feasible for production environments. You can now convert these deprecated table spaces to partition-by-growth (PBG) UTSs by using the new **MOVE TABLE** option on the **ALTER TABLESPACE** statement.

MOVE TABLE

Use the new **MOVE TABLE** option of **ALTER TABLESPACE** to move tables from a source multi-table table space to target PBG UTSs. The **MOVE TABLE** operation becomes a pending definition change that must be materialized by running the **REORG** utility on the source table space. Only one table can be moved per **ALTER TABLESPACE MOVE TABLE** statement, but multiple pending **MOVE TABLE** operations can be materialized in a single **REORG**.

The new **MOVE TABLE** clause is effective when running with APPLCOMPAT level V12R1M508 or later.

Before you can move a table, the target table space must exist as a PBG UTS in the same database as the source table and must be defined with the following attributes:

- ▶ **DEFINE NO**
- ▶ **MAXPARTITIONS** value of 1
- ▶ **LOGGED** or **NOT LOGGED** matching the source table space
- ▶ **CCSID** values that match the source table space

Two options are available for using this capability:

- ▶ Option 1: Use **ALTER TABLESPACE MOVE TABLE** to move all or a subset of the tables from the source table space to PBG UTS, as shown in Figure C-3.
 - Then, run the **REORG** utility on the source table space to materialize all pending changes including moving the tables.
 - After the **MOVE TABLE** operations are materialized, if no tables remain in the source table space, drop the source table space by issuing the **DROP** statement.

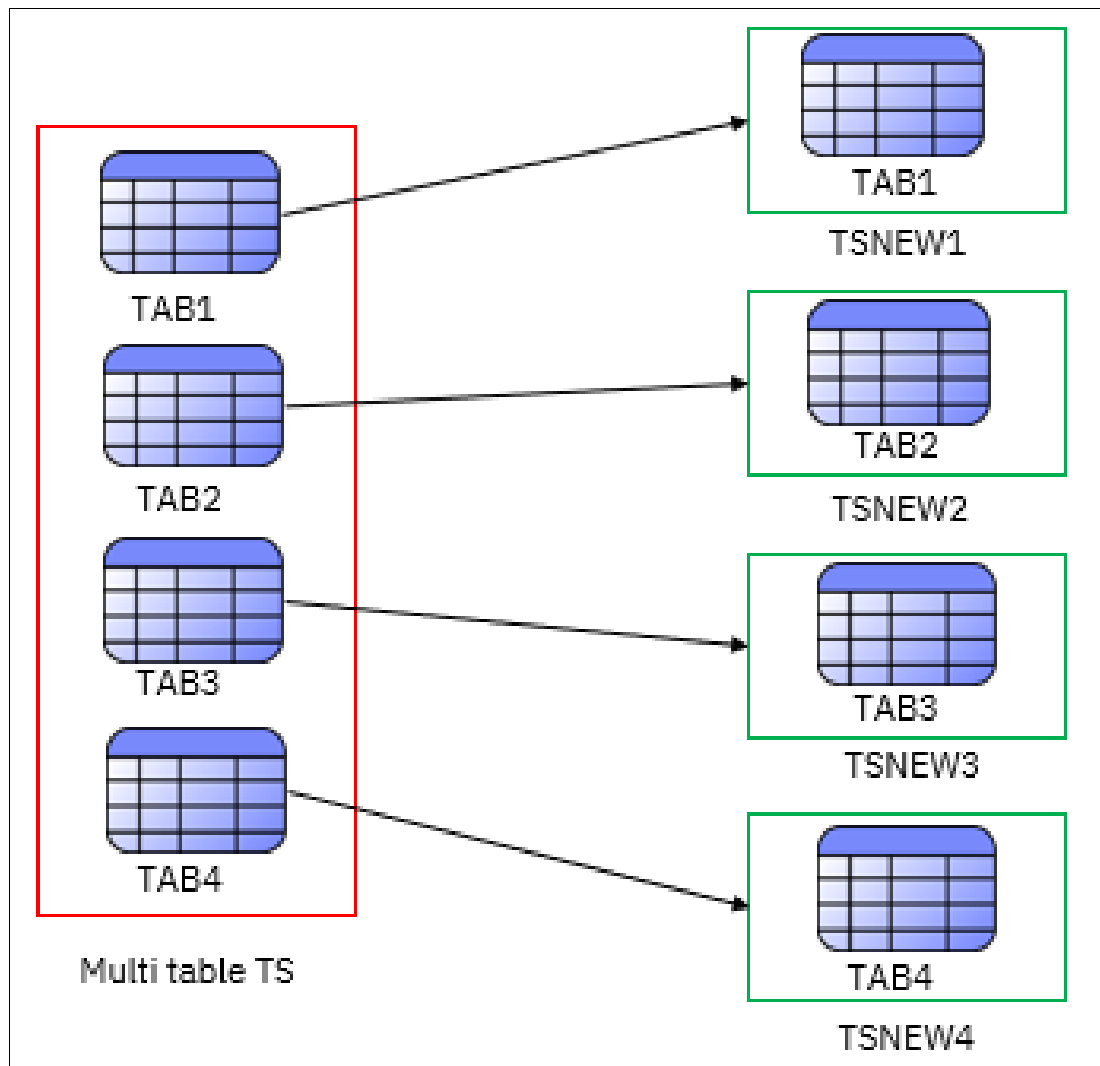


Figure C-3 Option 1: Source table space conversion to target PBG UTS by using MOVE TABLE

- Option 2: Move all but one table from the source table space to PBG UTSs, then convert the source table space into a PBG UTS by issuing `ALTER TABLESPACE` with `MAXPARTITIONS 1`, as shown in Figure C-4. This conversion of the source table space is a pending definition change that can be materialized with any pending definition changes for the source table space, including the pending `MOVE TABLE` operations.

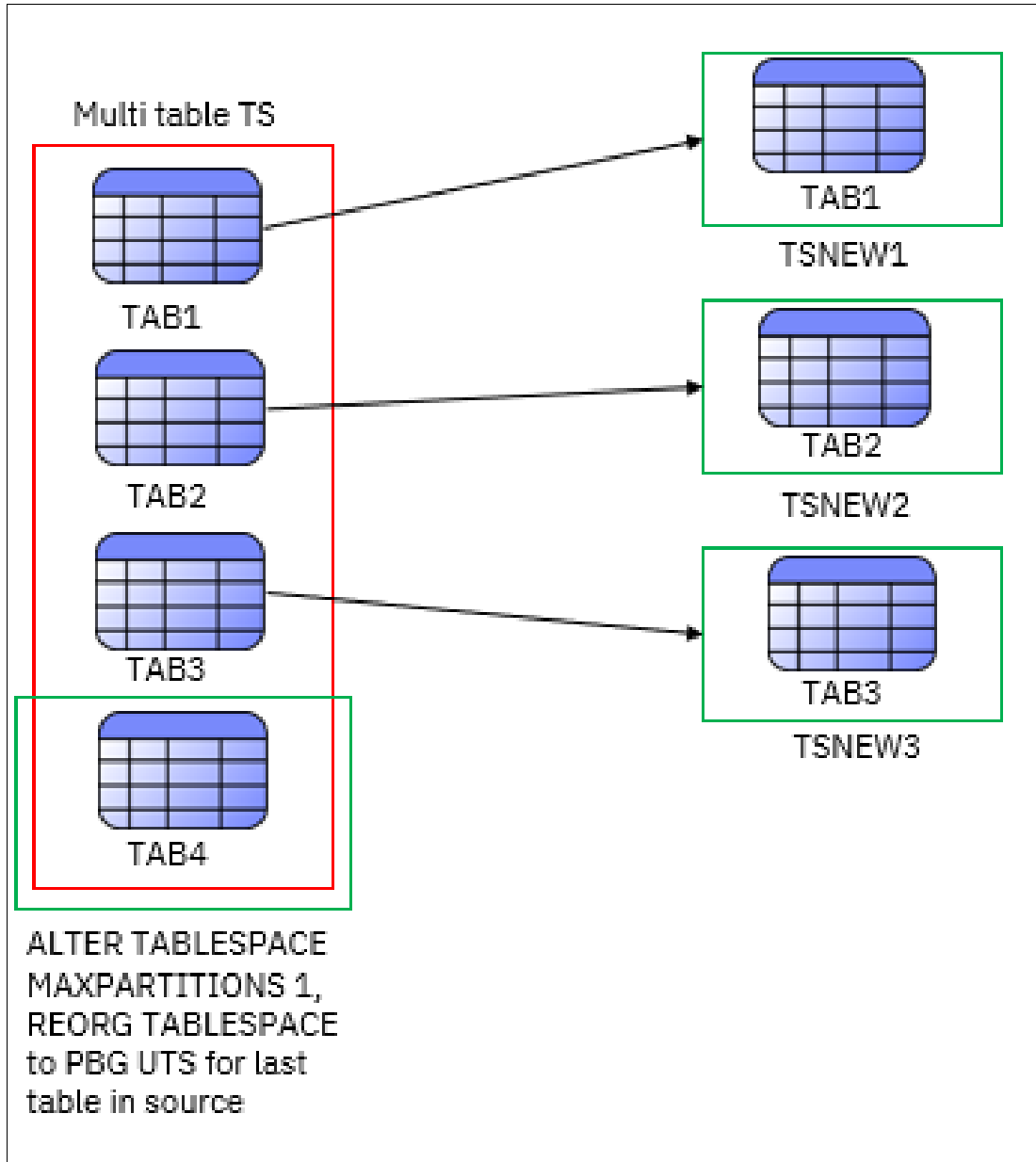


Figure C-4 Option 2: Source table space conversion to target PBG UTS by using `MOVE TABLE`

For more information about preparing for and issuing `MOVE TABLE` operations, see [Moving tables from multi-table table spaces to partition-by-growth table spaces](#).

For more information about creating image copies of the source table space and accessing historical table data from these image copies, see [Accessing historical data from moved tables by using image copies](#).

Db2 utilities

Db2 utilities added many features after the initial release of Db2 12 across many utility functions. Some reduce elapsed time or processing cost or increase zIIP usage. Numerous availability changes improve application access and utility concurrency. Others improve your productivity as you manage utility execution within your overall schedule.

Db2 utilities features that were added after the initial release of Db2 12 are supported at function level 100 (FL 100) unless otherwise indicated:

- ▶ **REORG**
- ▶ **LOAD**
- ▶ **UNLOAD**
- ▶ **RUNSTATS**
- ▶ **CHECK DATA**
- ▶ **COPY**
- ▶ **RECOVER**
- ▶ **MODIFY RECOVERY**
- ▶ Concurrency
- ▶ Replication support
- ▶ Stored procedures

REORG

Enhancements to the **REORG** utility can be grouped into these categories:

- ▶ Availability
- ▶ Performance
- ▶ Simplification and usability

Availability

Availability enhancements result in a greater likelihood of utilities completing and less impact to application workloads.

REORG of SYSCOPY

PI96693 UI57559 RSU1812: When **REORG SHRLEVEL CHANGE** runs for the SYSIBM.SYSCOPY table in table space DSND06.SYSTSCPY, other utilities that are running on different objects at the same time and need to access or update SYSCOPY are no longer impacted. This enhancement allows **REORG** to be run without having to consider any special scheduling.

REORG NOCHECKPEND

PH13527 UI65509 RSU2003: The new **NOCHECKPEND** keyword allows for improved availability when **REORG** discards rows from a parent table of a referential integrity relationship. When the keyword is specified, **REORG** avoids setting **CHECK-pending (CHKP)** status on dependent table spaces. This option applies only when **REORG DISCARD** is specified. The **NOCHECKPEND** specification does not reset a **CHECK-pending** status that is already in place before running the **REORG**.

REORG FORCE hard cancel

PH12240 UI68539 RSU2006: When **REORG FORCE ALL** is specified, the utility tries to cancel any blocking read or write claimers on the table space so that it can drain the table space to complete the **SWITCH** phase. Before this enhancement, the **FORCE** option did not always work in cases where a claimer's thread was running code outside the Db2 address space; that deficiency was eliminated so that the cancel always takes effect.

REORG drain failure messages

PI95911 UI59673 RSU1903: If a **REORG** cannot successfully acquire a drain at the beginning of the **SWITCH** phase, **REORG** issues the **DIS DB LOCKS** command instead of **DIS DB CLAIMERS**. This command allows better visibility for determining which applications or processes are blocking **REORG** from draining. The utility now shows locks that block **REORG** and that do not have accompanying claims (for example, **RETAIN** locks).

Performance

REORG performance enhancements can reduce elapsed time, CPU time, or both, and they can result in higher proportions of specialty engine (zIIP) usage.

REORG INDEX performance

PH25217 UI75477 RSU2109: When **REORG INDEX** is invoked with **SHRLEVEL REFERENCE** or **CHANGE**, the **NOSYSUT1** keyword specifies that **REORG INDEX** no longer uses work data sets to hold the unloaded index keys. The utility passes the unloaded keys in memory as input to the index build for this process, so performance is improved. For example:

```
REORG INDEX index-name NOSYSUT1
```

In addition, multiple subtasks can be used to reorganize different partitions within a partitioned index in parallel. You can specify the **PARALLEL n** keyword to constrain the number of subtasks that are used.

The IBM performance measurements show a significant elapsed-time and CPU-time improvement. Partitioned indexes show up to an 86% improvement in elapsed time, and CPU reduction can be even higher. Because the tasks are now all zIIP eligible, the reduction in non zIIP CPU was measured to be 50 - 95%.

For ease of enablement, a new online changeable subsystem parameter was introduced to enable the new **REORG INDEX NOSYSUT1** behavior without specifying the new **NOSYSUT1** keyword. The **REORG_INDEX_NOSYSUT1** subsystem parameter defaults to **N0**, but you can set it to **YES** to enable all **REORG INDEX** executions to use this new performance improvement.

These performance improvements are summarized in Table C-3 and Table C-4 on page 279.

Table C-3 Elapsed time by index type

Index type	Elapsed time (ET) base	ET new	% delta
Partitioning index (PI)	39.44	6.68	-83.06
Non-partitioned secondary index (NPI)	26.93	13.16	-51.13
Data-partitioned secondary index (DPSI)	34.34	4.62	-86.55

Index type	Elapsed time (ET) base	ET new	% delta
NPI with multiple columns	32.54	16.3	-49.91
NPI with CHAR and VARCHAR columns	140.24	62.86	-55.18

Table C-4 CPU and zIIP

CPU old	CPU new	% delta	zIIP old	zIIP new	% delta
11.49	3.73	-67.54	5.7	25.36	-6.29
10.91	0.01	-99.91	5.0	96.97	36.94
10.93	3.27	-70.08	5.1	54.6	-10.68
11.61	0.01	-99.91	5.5	37.59	37.25
26.74	0.01	-99.96	12.26	15.07	22.9

REORG with single PI parallel build

PH28092 UI71385 RSU2012: **REORG TABLESPACE** running with **SHRLEVEL REFERENCE** or **CHANGE** and **SORTDATA**, with data unload reload partition parallelism, skips the sorting of keys for the clustering PI and builds the PI partitions in parallel during the **RELOAD** phase. This optimization is enabled when the PI is the only index that is defined on the table.

REORG TABLESPACE DFSORT Z SORT acceleration

PH28183 UI71668 RSU2012: **REORG TABLESPACE** supports **DFSORT** with the IBM Integrated Accelerator for IBM Z Sort feature, which is available in the latest z15 mainframe hardware. This feature leverages the acceleration for variable-length records during **REORG TABLESPACE** when the data records of the target table space are sorted. CPU and elapsed-time improvements are achieved with more in-memory sorting.

Reduced REORG SYSUTILX logging

PI92536 UI56383 RSU1809: The **SYSUTILX** table space saves a substantial amount of information that can be used to support restart processing. This information is written into the Db2 log. However, for **REORG TABLESPACE SHRLEVEL CHANGE**, restarting of the utility is not allowed until the **SWITCH** phase, so logging is now greatly reduced in earlier phases.

Simplification and usability

Enhancements to simplification and usability can be grouped into these categories:

- ▶ REORG inline copy partition bundling
- ▶ Tolerating deleting rows discarded by REORG DISCARD
- ▶ DSNU2930I REORG row wrong partition message

REORG inline copy partition bundling

PI75518 UI75979 RSU2108: **REORG TABLESPACE** traditionally allows the creation of a single inline copy data set for any number of partitions being reorganized. Two new keywords are supported for the **REORG TABLESPACE** utility: **ICLIMIT_DASD** and **ICLIMIT_TAPE**. These new keywords allow you to control how many partition-level image copy data sets **REORG** can or should allocate based on the table space being reorganized and the restrictions in the user environment.

Due to hardware restrictions (for example, the number of available tape drives) or virtual memory requirements on a large partitioned table space that has many partitions, it might not always be feasible for **REORG** to allocate a separate image copy data set for each data partition that is being reorganized. At the same time, having a single image copy data set for all target partitions can result in problems with management and slower recovery times in some scenarios.

You can now more easily manage these trade-offs by using the new keywords to dictate the upper boundary on the number of image copy data sets to allocate. **REORG** uses this input to decide how to bundle partitions into the specified number of image copies used.

Reorganization for PBR table spaces with relative page numbering (RPN) is also improved by removing the previous restriction that required separate image copies for each partition that is being reorganized. Performing a multi-partition **REORG TABLESPACE** operation against a PBR RPN without a **TEMPLATE** that has **&PART** or **&PA** specified in the **COPYDDN** or **RECOVERYDDN** statements no longer fails with RC8 and message DSNU2922I - PARTITION LEVEL INLINE DATASETS ARE NOT SPECIFIED.

Tolerating deleting rows discarded by REORG DISCARD

PI98259 UI60755 RSU1906: Removing stale data from a table space is commonly done during **REORG** by using the **DISCARD** option. However, if these “stale” data rows happen to unexpectedly be deleted by an application while **REORG** is discarding them, **REORG** now tolerates the anomaly and completes successfully.

DSNU2930I REORG row wrong partition message

PI88906 UI62291 RSU1909: Partitioned table spaces that have limit keys for each partition can experience data integrity problems if data is in the wrong partition. When this situation occurs, the **REORG** might fail if it detects the error when rebuilding a partitioned index. However, in some cases it can go undetected. **REORG** now detects data rows in the wrong partition and issues a warning message so that the error situation can be more quickly diagnosed and remedied.

LOAD

Enhancements to the **LOAD** utility can be grouped into these categories:

- ▶ Performance: **LOAD PRESORT** support
- ▶ Simplification and usability
- ▶ Availability

Performance: LOAD PRESORT support

PH19067 UI67869 RSU2006: The **PRESORT** specification allows sorting of input data records in clustering order before loading them into the table. No external presorting or subsequent **REORG** is needed to ensure that the data is in clustering sequence.

You can specify the **PRESORT** option when loading from a single input data set into a single table, or when loading from multiple input data sets, one per partition. It also can be specified when loading multiple tables, each with its own clustering sequence. **PRESORT** does not affect existing rows in the table space, and full clustering cannot be guaranteed when loading into a non-empty page set.

Simplification and usability

Enhancements to simplification and usability can be grouped into eight categories:

- ▶ LOAD image copy improvements
- ▶ LOAD consistent image copy
- ▶ LOAD override row timestamp
- ▶ New option UPDMAXASSNEDVAL YES/NO
- ▶ LOAD DEFAULTIF Extensions
- ▶ Julian date packed decimal format
- ▶ LOAD multiple SYSREC support
- ▶ LOAD REPLACE to DEFER DEFINE AUX data sets

LOAD image copy improvements

PH40709 UI78864 PUT2201: This enhancement effectively splits the copy-spec from the resume-spec to allow **LOAD** to have a single **COPYDDN** (or **RECOVERYDDN**) specification for the entire table instead of multiple partition-level copy-spec clauses when specifying individual target partitions in the **LOAD** statement. Instead of requiring an inline image copy either for every specified partition or all partitions in the table space, this feature allows the bundling of one or more partitions (or all table space partitions) within a single inline copy.

This capability can be useful in cases where you want to drive **LOAD** at the partition level, but are constrained by a limited number of separate image copies that can be created during a single **LOAD** due to the limited availability of tape drives. Also, this enhancement allows serialization on the partitions that are included in the **LOAD** rather than the entire table space.

LOAD consistent image copy

PH39300 UI78743 PUT2112: **LOAD SHRLEVEL REFERENCE** is enhanced so that the image copy no longer contains rows that were not loaded into the table because they had errors. If duplicate key errors were encountered or if referential integrity violations occurred, the rows might be discarded instead of loaded. In this case, at the end of **LOAD** execution, the data rows that are contained in the image copy now always match the rows that were loaded into the table space.

LOAD override row timestamp

PH25572 UI70476 RSU2012: A new **OVERRIDE** option of **TIMESTAMP** allows data to be loaded into a row change timestamp column that is defined as **GENERATED ALWAYS**. This specification allows you to load a table with data that was previously unloaded from another table so that the original timestamps can be preserved.

New option UPDMAXASSNEDVAL YES/NO

PH28476 UI71293 RSU2012: When you load tables that have identity columns, by default the **LOAD** utility updates the value of **MAXASSIGNEDVAL** in the **SYSIBM.SYSSEQUENCE** catalog table as needed. If the value of the identity column is generated by Db2, **LOAD** updates the **MAXASSIGNEDVAL** column with the last assigned value.

A new keyword, **UPDMAXASSIGNEDVAL**, allows you to override updating the **MAXASSIGNEDVAL** by specifying **NO**. The default is **UPDMAXASSNEDVAL YES**.

LOAD DEFAULTIF Extensions

PH30610 UI73338 RSU2106: In **LOAD** utility statements, you can load a default value into a column by specifying the **DEFAULTIF** clause. Originally, the only predicate that was supported was the “equal to” (=) comparison. However, the “not equal to” (<>) comparison is now supported, along with the new **CONV_ERROR** keyword. The **CONV_ERROR** specification allows a default value to be loaded if a conversion error occurs.

The following variations for **DEFAULTIF** on a column specification are now supported:

- ▶ **COLx DEFAULTIF(any_column = 'ABC')**
- ▶ **COLx DEFAULTIF(any_column <> 'ABC ')**
- ▶ **COLx DEFAULTIF(CONVERR)**

Julian date packed decimal format

PH22944 UI70034 RSU2009: **LOAD** now supports a new column data type, **DATE_P**, which accepts a 3-byte packed decimal format 'YYDDDs'x as input.

LOAD multiple SYSREC support

PI96136 UI62234 RSU1909: The **LOAD** utility now allows multiple input data sets to be specified in the **INDDN** option. This enhancement is supported at the statement level when loading an entire table, and in the **INTO TABLE PART** clauses when individual partitions are loaded.

LOAD REPLACE to DEFER DEFINE AUX data sets

PH19072 UI70477 RSU2012: A new keyword option, **DEFINEAUX**, enables the **LOAD** utility to define all LOB or XML objects for a table, even if they are not populated when loading the data.

Availability

Enhancements to availability can be grouped into these categories:

- ▶ **LOAD** Read Only table spaces support
- ▶ **LOAD SHRLEVEL REFERENCE FORCE** option
- ▶ **UNLOAD REGISTER YES** default

LOAD Read Only table spaces support

PH26131 UI71238 RSU2012: A new subsystem parameter, **LOAD_RO_OBJECTS**, supports loading data into target objects that are in a read-only state. When the **LOAD_RO_OBJECTS** subsystem parameter is set to **YES**, **LOAD** is allowed to run against objects in read-only Database Exception Table (DBET) status. The default value of **LOAD_RO_OBJECTS** is **NO**.

Target objects include the ones that were started with **-START DATABASE() SPACENAM() ACCESS(RO)**, and include the base table space, base table space indexes, LOB table spaces, LOB indexes, XML table spaces, and XML indexes. **LOAD_RO_OBJECTS** also applies when the underlying database for an object is in read-only state.

LOAD SHRLEVEL REFERENCE FORCE option

PH24369 UI70432 RSU2012: A new keyword option, **FORCE**, enables the **LOAD** utility to attempt to cancel any blocking read and write claimers to drain a table space and complete the **SWITCH** phase. Options of **NONE**, **READERS**, or **ALL** are available to specify no action, canceling of read claimers, or canceling of both read and write claimers.

UNLOAD

There is only one enhancement to **UNLOAD**.

UNLOAD REGISTER YES default

PI99075 UI57005 RSU1812: The **REGISTER** keyword indicates whether pages that contain the unloaded rows should be registered with the coupling facility. The default is **YES**, which registers pages. This behavior allows rows that are changed concurrently on other members to be picked up when unloading. You can specify **REGISTER NO** to override the default to give better performance while potentially not including changes that were made concurrently on other members.

RUNSTATS

Enhancements to the **RUNSTATS** utility can be grouped into these categories:

- ▶ Performance
- ▶ Simplification and usability

Performance

Enhancements to performance can be grouped into these categories:

- ▶ COLGROUP sort avoidance
- ▶ RUNSTATS STATCLGSRT keyword
- ▶ RUNSTATS default page sampling FL505

COLGROUP sort avoidance

PI76730 UI51754 RSU1803: When collecting statistics, you can use the **COLGROUP** keyword to collect statistics to improve the access path selection; however, doing so might add a significant cost for collecting statistics because it uses an external sort to order the values. As an alternative, you can use an in-memory hash technique for single-column **COLGROUPS** with a **COUNT <= 100**. This sort avoidance technique is used when the new **STATCLGSRT** subsystem parameter is specified. Specifying **STATCLGSRT n** indicates how much memory is to be used for implementing the new hash technique.

RUNSTATS STATCLGSRT keyword

PH03678 UI60747 RSU1906: APAR PI74408 enabled **COLGROUP** sort avoidance by using the subsystem parameter **STATSCLGSRT**. However, instead of sorting, you can use a more granular approach for this hash technique to avoid sorting by specifying the **STATSCLGSRT** keyword as part of **RUNSTATS** or the inline statistics specification with **LOAD** or **REORG**.

RUNSTATS default page sampling FL505

PH09191 UI63807 RSU1909: **RUNSTATS** can look at every row when gathering distribution statistics, sample rows, or sample pages. The page sampling technique is the preferred method because it offers the best performance to determine the correct access path and is considered the best practice for large amounts of data.

In function level 505 (FL505), the **STATPGSAMP** subsystem parameter is set so that the default for all **RUNSTATS** jobs is to use page sampling. If the **STATPGSAMP** default is not acceptable, you can set the subsystem parameter to **N0**, or you can override the individual jobs by using the **TABLESAMPLE SYSTEM** keyword specification.

Simplification and usability

The following sections describe the simplification and usability enhancements.

USE PROFILE OPTIONS in output

PH11423 UI63894 RSU1912: The recommended method for gathering statistics is to use the **PROFILE** keyword for controlling which statistics are gathered for a table space or index. The keywords that are used within a given profile are adaptive because the optimizer can make changes for improved statistics gathering. For a better understanding of the options that are used to gather statistics, you can evaluate the details of the profile, which are now echoed in the job output.

USE PROFILE to delete non-profile statistics (FL 507)

PH16345 UI68772 RSU2009: When a profile is used to gather statistics, all statistics in the catalog that are not related to the profile and were not recently gathered, updated, and inserted, are deleted.

CHECK DATA

There is only one **CHECK DATA** enhancement.

CHECK SHRLEVEL CHANGE reset CHKP

PH25593 UI70821 RSU2012: **CHECK DATA SHRLEVEL CHANGE** and **CHECK LOB SHRLEVEL CHANGE** now reset an existing **CHECK**-pending, auxiliary **CHECK**-pending, or auxiliary warning restrictive state if no inconsistencies are found.

COPY

Enhancements to **COPY** can be grouped into these categories:

- ▶ Performance
- ▶ Availability

Performance

Enhancements to performance can be grouped into these categories:

- ▶ **COPY FLASHCOPY CONSISTENT**
- ▶ **TEMPLATE Large Block Interface support (FL 500)**

COPY FLASHCOPY CONSISTENT

PI93390 UI56284 RSU1807: Objects are now processed at the table space level instead of handling them at the partition level.

TEMPLATE Large Block Interface support (FL 500)

PH30093 UI72836 RSU2103: Support for the Large Block Interface (LBI) provides better performance by decreasing CPU and elapsed time for utilities. The **BLKSSZLKIM** option allows the specification of much larger blocks for basic sequential access method (BSAM) files. LBI is useful mainly when creating new data sets on tapes.

Availability

There is only one availability enhancement.

FLASHCOPY_XRCP subsystem parameter to support remote pair FlashCopy in Extended Remote Copy environments

PH01728 UI59783 RSU1903: Asynchronous mirroring of Db2 production disks by using Extended Remote Copy (XRC) supports remote pair FlashCopy (RPFC). The primary Db2 production volumes at the local site and the secondary volumes at the remote site are paired, and the FlashCopy operation is done remotely by sending the command to be run. This operation eliminates the need to replicate the data across the connection. Setting **FLASHCOPY_XRCP** to YES allows utilities to use FlashCopy on a target data set that is on primary XRC volumes.

RECOVER

Enhancements to **RECOVER** can be grouped into these categories:

- ▶ Redirected recovery for table spaces (FL 500)
- ▶ Redirected recovery for index spaces and indexes (FL 500)

Redirected recovery for table spaces (FL 500)

PH27043 UI72057 RSU2103: Testing the recovery of production data without impacting the data and applications can be time-consuming and complex. Redirected recovery provides a safe, easy method to run test recoveries to accurately determine how long the recoveries will take, check for recovery issues, review or undertake forensic analysis of production data, and generate data at different points in time (PITs) with transactional consistency, all without affecting production table spaces and with no impact to production applications.

With redirected recovery, the target table space is recovered by using the recovery assets (image copy and logs) of the associated source table space.

Redirected recovery for index spaces and indexes (FL 500)

PH35266 UI76045 RSU2112: Index spaces and indexes (with the **COPY YES** attribute) can be processed by redirected recovery along with table spaces for estimating recovery time, testing recovery, and reviewing and generating data. The target index is recovered by using the recovery assets (image copy and logs) of the associated source index. You can also use redirected recovery to evaluate and compare index recovery and index rebuild operations.

MODIFY RECOVERY

There is only one **MODIFY RECOVERY** enhancement.

Improved FlashCopy image copy management (FL 500)

PH04023 UI62236 RSU1909: A popular backup and recovery strategy uses a technique of creating instantaneous backups on disk (for example FlashCopy image copies) that are later copied to sequential image copies on tape by using **COPYTOCOPY**. Before this enhancement, these duplicate image copies were managed together, which occupied disk space until both image copies were deleted together by **MODIFY RECOVERY**.

This enhancement allows the original FlashCopy image copies to be deleted separately and sooner by using **MODIFY RECOVERY FLASHCOPY ONLY** while keeping the sequential image copies on tape. Disk space can now be freed and reused for subsequent FlashCopy image copies.

Concurrency

There is only one concurrency enhancement.

LOAD and REORG SHRLEVEL CHANGE concurrency

PH27915 UI71560 RSU2012: **REORG SHRLEVEL CHANGE** can now run concurrently with an **UNLOAD SHRLEVEL CHANGE** utility. However, concurrency is still not allowed when **REORG** requires exclusive control of the target objects in the last **LOG** iteration and **SWITCH** phase, during which the concurrent utilities still use the standard drain and claim serialization control.

Replication support

Enhancements to replication support can be grouped into these categories:

- ▶ Writing a diagnostic log record
- ▶ Prohibiting utilities on replicated tables

Writing a diagnostic log record

PH11871 UI65271 RSU1912: A new **REPAIR** option, **WRITELOG**, allows writing a diagnostic log record that was used by replication to force a refresh. This enhancement improves the functions of replication processes.

Prohibiting utilities on replicated tables

PH14363 UI67200 RSU2006: Currently, certain utility executions that are enacted on a table with the **DATA CAPTURE CHANGES** attribute or on a table space that contains such a table causes the data in the replication target tables to be inconsistent until a refresh is done. You can use the new **UTILS_BLOCK_FOR_CDC** subsystem parameter to prohibit certain utilities from running on replicated objects. When a utility is blocked, message DSNU1871 is issued by the restricted utilities at run time. This restriction applies to the following utilities:

- ▶ **CHECK DATA** with **DELETE YES LOG NO**
- ▶ **LOAD WITH SHRLEVEL NONE** or **REFERENCE**
- ▶ **RECOVER** with **TOLOGPOINT**, **TORBA**, **TOCOPY**, **TOLASTCOPY**, or **TOLASTFULLCOPY**
- ▶ **REORG TABLESPACE** with **DISCARD**
- ▶ **REPAIR** with **LOCATE DELETE**

Stored procedures

There is only one stored procedures enhancement.

DSNACCOX

PH25108 UI72637 RSU2103: On a large subsystem with many objects in a restricted or advisory state, the number of messages that are returned by a **DISPLAY** command can exceed the space that is available, in which case the resulting messages can be truncated. In cases where you might be interested in checking objects in a specific database, provide a **CRITERIA** parameter with **DBNAME = *database-name***. However, because the resulting messages of the **DISPLAY DATABASE** are truncated, the specific objects in the database in which you are interested in might have been truncated before they can be evaluated by **DSNACCOX**.

DSNACCOX was enhanced to check the **CRITERIA** parameter. If it contains **DBNAME = *database-name*** or contains a **DBNAME = *database-name*** specification, **DSNACCOX** extracts the database name, uses it to build the **DISPLAY DATABASE** command, and returns the objects of interest.

This APAR also fixes a problem where an **SQLCODE = -305** for no **NULL** indicator is returned from a data sharing check by using the SQL function **GETVARIABLE**.

DSNACCOX defaults

PH32763 UI73842 RSU2106: The following default values for **DSNACCOX** were changed based on recommendations from a Db2 360 study:

- ▶ **RRIPseudoDeletePct** from -1 (off in non-data sharing) to 5
- ▶ **RRIEmptyLimit** from 10 to = 5



System administration and system management

Db2 12 enhancements in the system administration and system management areas include performance, system topics, security, and synergy with IBM z15 and IBM DS8000 storage families.

For more information about extra features that provide synergy between IBM Db2, z15 systems, and the DS8000 family, see Appendix A, “IBM Db2 12 continuous delivery features and enhancements” on page 233.

This appendix describes the following topics:

- ▶ Performance
- ▶ Db2 system topics
- ▶ Security

Performance

Enhancements in area of performance include monitoring and management changes and opportunities for CPU and elapsed time reductions. You can achieve CPU and elapsed time reductions by using the changes to fast index traversal and by using new **`SORT LIST (SORTL)`** capabilities:

- ▶ Fast index traversal
- ▶ **`SORTL`**

Monitoring and management changes include a new command option for the resource limit facility (RLF), a new default setting for the **`DEFAULT_INSERT_ALGORITHM`** subsystem parameter, and the new capability of populating reporting fields and adding a reporting field for index rowid (RID) list processing.

Fast index traversal

Fast index traversal, also known as index fast traverse blocks (FTBs), can improve the performance of random index access. Db2 continuously monitors index activity and applies fast index traversal to eligible indexes that exceed internally defined thresholds.

Db2 12 extended the benefits of fast index traversal after general availability (GA) by adding support for non-unique indexes so that you have greater flexibility to monitor and control fast index traversal in your environment.

Fast index traversal and real storage

FTB real storage consumption is in addition to buffer pool real storage requirements, so it must be included in real storage budgeting. Db2 does not allocate FTBs if your system is experiencing paging.

For more information about fast index traversal, see [Fast index traversal](#).

Fast index traversal at Db2 12 general availability

FTBs directly target the performance of common online transaction processing (OLTP) workloads, specifically random index access during direct key lookups. Previous index optimizations provided getpage reduction benefits to sequential index access. For random index access without FTBs, multiple getpage operations might be required to traverse the index from the root page to the leaf page in the index tree. By placing the root page and non-leaf pages in an FTB structure, Db2 can provide significant reductions in getpage operations.

FTBs use a separate memory area to cache non-leaf pages so that fewer getpage operations are needed for index traversal. Initial FTBs support only unique indexes with a key size of 64 bytes or less. Unique indexes with **`INCLUDE`** columns also are eligible if the unique part of the key does not exceed 64 bytes. Db2 deploys FTBs for indexes automatically and dynamically based on current workload patterns.

Figure D-1 on page 291 shows the process of accessing a leaf page before using FTBs.

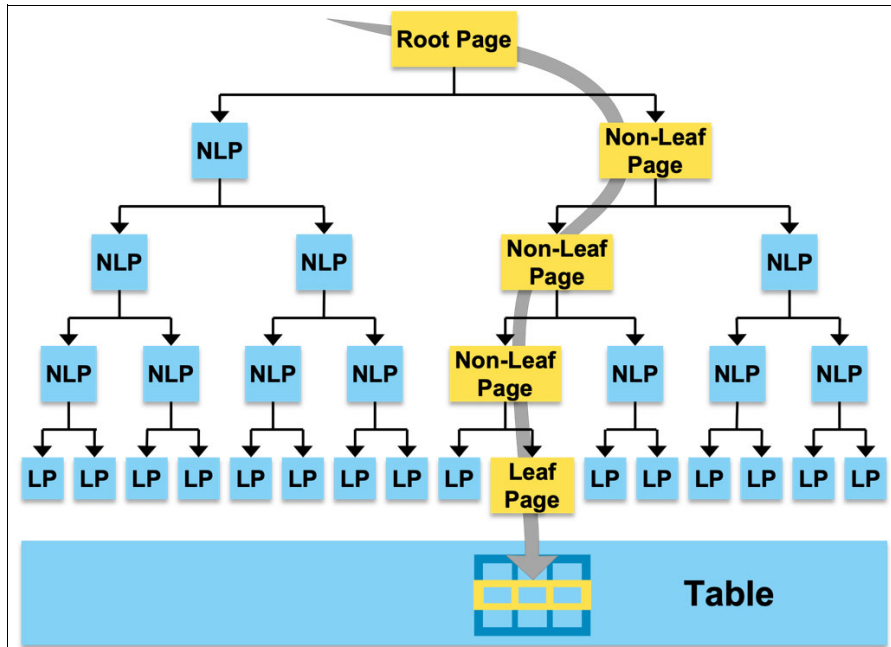


Figure D-1 Accessing a leaf page without using FTBs

Db2 performs a getpage for each of the index pages and for the data page. In this example of a 5-level index, five getpage operations are issued for the index pages, and one more to access the data page, for a total of six getpage operations.

Figure D-2 shows the getpage reduction by using FTBs. The access starts with the FTB structure for the index. The root and non-leaf pages are represented in the FTB. In this case, Db2 performs getpage operations for only the green pages (the index leaf page and the data page).

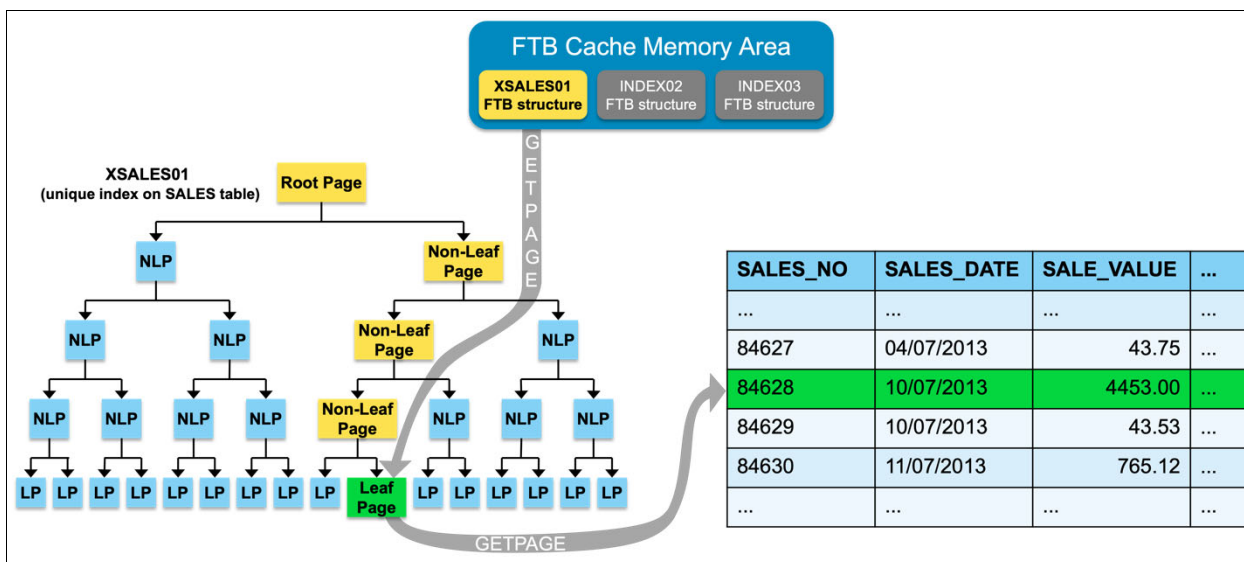


Figure D-2 Accessing a leaf page by using FTBs

The getpage reduction that is shown in Figure D-2 results in significant reduction in CPU cost for random index access. For the same five-level index scenario, only two getpage operations are required.

Fast index traversal control

By default, Db2 builds FTBs for eligible indexes, which can consume up to 20% of the storage value of your overall buffer pool allocation. You can control whether Db2 builds FTBs and how much storage Db2 uses by using the **INDEX_MEMORY_CONTROL** subsystem parameter. Initial support for **INDEX_MEMORY_CONTROL** provided three choices:

AUTO	Db2 determines the amount of storage to allocate and for which indexes to build FTBs. This value is the default.
DISABLE	Db2 does not build FTBs.
<i>n</i>	The range for <i>n</i> is 10 - 200000 MB (10 MB to 200 GB).

In a data-sharing environment, it is recommended that all members use the same setting.

Fast index traversal is intended to be controlled by Db2 automatically. If you require more granular control over which indexes have FTBs built for specific exceptions, you can populate rows in the SYSIBM.SYSINDEXCONTROL catalog table. By using this table, you can direct Db2 to include or exclude indexes from consideration for FTBs overall or based on the time of day or on the day of the week or month.

Fast index traversal enhancements after Db2 12 general availability

The following sections summarize enhancements that were made to FTBs after the initial release of Db2 12.

New INDEX_MEMORY_CONTROL option

Db2 adds the new (**SELECTED, *n***) and (**SELECTED, AUTO**) options for the **INDEX_MEMORY_CONTROL** subsystem parameter. You can use either option with the SYSIBM.SYSINDEXCONTROL table if you want to limit the index spaces or index partitions that Db2 can consider as candidates for FTB allocation. When you specify **SELECTED**, Db2 considers those indexes or index partitions with a value of 'A' in column "SELECTED" in SYSINDEXCONTROL. The values of *n* and **AUTO** remain the same: *n* can be 10 MB - 200 GB, and **AUTO** means that Db2 determines the storage allocation for the FTBs.

Non-unique index support

Db2 12 function level 508 (FL 508) extends fast index traversal support to include non-unique indexes with a key of 56 bytes or less. You can set the **FTB_NON_UNIQUE_INDEX** subsystem parameter to YES if you want Db2 to consider non-unique indexes as candidates for FTBs. The default is NO.

The -DISPLAY STATS enhancement

Db2 12 introduces the new **DISPLAY STATS (INDEXTRAVERSECOUNT)** command, or **DIS STATS (ITC)**. You can use this display command to see the index traverse count for indexes or index partitions in your environment.

Example D-1 and Example D-2 on page 293 show some examples of using this command.

Example D-1 Displaying index traversal counts for a specific index partition

```
DISPLAY STATS(INDEXTRAVERSECOUNT) DBNAME(DB1) SPACENAM(IX1) PART(1)
DSNT830I -
DBID PSID DBNAME IX-SPACE LVL PART TRAV. COUNT
-----
0017 0005 DB1 IX1 003 0001 00000000999
***** DISPLAY OF STATS ENDED *****
```

Example D-2 Displaying index spaces with the most index traversals

```
DISPLAY STATS(ITC) DBNAME(DB1) LIMIT(*)
DSNT830I -
DBID PSID DBNAME IX-SPACE LVL PART TRAV. COUNT
--- ---
0017 0005 DB1 IX1 003 0001 00000050099
0017 0005 DB1 IX2 003 0010 00000050001
0017 0005 DB1 IX2 003 0008 00000030021
0017 0005 DB1 IX2 003 0016 00000029999
0017 0005 DB1 IX3 003 0001 00000000999
0017 0005 DB1 IX1 003 0003 00000000800
0017 0005 DB1 IX5 003 0001 00000000666
0017 0005 DB1 IX1 003 0022 00000000100
0017 0005 DB1 IX7 003 0001 00000000100
0017 0005 DB1 IX1 003 0002 00000000100
***** DISPLAY OF STATS ENDED *****
```

You can use the output of these commands to evaluate potential candidates for index fast traversal.

For more information about the syntax options for this command, see [-DISPLAY STATS \(Db2\)](#).

SORTL

Db2 12 supports IBM Integrated Accelerator for IBM Z Sort, which is available on the z15 and later systems. It uses the **SORTL** instruction to sort lists of data. During the input phase, **SORTL** sorts multiple lists of unsorted data into one or more lists of sorted data, and then merges these multiple lists of sorted input data into a single list of sorted output data.

SORTL does the sort and merge during the input phase based on the maximum number of records that it can handle at one time. If all the data cannot be sorted in the input phase, **SORTL** creates work files to hold the resultant sorted lists. Then, **SORT** merges these work files together during the merge phase.

You might observe reduced CPU and elapsed time by using **SORTL** for sort-intensive operations. The enhancement is limited to a sort key size of 136 bytes and a sort record size of 256 bytes. You also observe more memory consumption during sort phases.

Db2 can use **SORTL** for Relational Data System (RDS) sort, the **REORG** utility, or on a **DFSORT** invocation. RDS uses **SORTL** for ORDER BY and GROUP BY operations automatically after APAR PH31684 is applied. To enable **SORTL** for **REORG**, you must apply APAR PH28183 and set the **UTILS_USE_ZSORT** subsystem parameter to YES (NO is the default). For **DFSORT**, use **OPTION ZSORT** in the **DFSORT** control statement or **ICEMAC/PRM**.

As part of this support, Db2 adds statistics counters to track the total number of Db2 sorts and the number of Db2 sorts that use **SORTL**. There are changes to Instrumentation Facility Component Identifier (IFCID) 96 to add identifiers (STLO for ORDER BY operations and STLG for GROUP BY operations) and new fields (key size and data size).

You can use the IBM Z Batch Network Analyzer (zBNA) tool to estimate potential **SORTL** benefits for your workload. You can download zBNA at [IBM Z Batch Network Analyzer \(zBNA\) Tool](#).

Db2 system topics

Db2 provides the following system-related improvements in the Db2 maintenance stream. These improvements do not depend on any particular Db2 function level.

- ▶ Data sharing
- ▶ Distributed Data Facility
- ▶ Subsystem monitoring
- ▶ DSNZPARM simplification

Data sharing

The following topics are described in this section:

- ▶ RUNSTATS and UNLOAD utilities
- ▶ Retrying automatic logical page list and GBP recovery pending recovery
- ▶ Asynchronous cross-invalidation of Group Buffer Pools
- ▶ IRLM deadlock
- ▶ -ALTER GROUPBUFFERPOOL (Db2)
- ▶ Sysplex support for multi-factor authentication and IBM RACF PassTickets

RUNSTATS and UNLOAD utilities

Because **RUNSTATS** or **UNLOAD** can cause Group Buffer Pool (GBP) dependency or page registration in some circumstances, Db2 12 adds support for the **REGISTER** option for a **SHRLEVEL CHANGE** invocation of these utilities.

The default value for the **REGISTER** options is YES in both cases:

- ▶ **RUNSTATS...SHRLEVEL CHANGE REGISTER (YES, NO)**
- ▶ **UNLOAD...SHRLEVEL CHANGE ISO(UR) REGISTER (YES, NO)**

Setting **REGISTER** to NO might reduce GBP dependency, depending on your other workloads, and removes the requirement of the utility to register the pages that the utility reads. Each of these effects might reduce utility elapsed time and reduce CPU and coupling facility (CF) processor consumption. By not registering interest in those pages, the utility might not be aware of changes being made by another Db2 member in the data sharing group. You should consider the impact of non-current data for these utilities before specifying **REGISTER NO**.

Retrying automatic logical page list and GBP recovery pending recovery

When pages are added to the logical page list (LPL) because they cannot be written to the GBP or when objects are placed in GBP recovery pending (GRECP) status, Db2 automatically attempts recovery actions. Sometimes the initial recovery attempts are not successful. Db2 12 retries automatic recovery up to three times if pages or objects are still in the LPL or GRECP status after automatic LPL or GRECP recovery. After any retries are complete, Db2 issues DSNB317I for successful recovery actions or DSNB328I if some recovery actions were not completely successful, as shown in Example D-3 on page 295 and Example D-4 on page 295.

Example D-3 Possible messages after automatic GRECP recovery

```
DSNB317INO OBJECTS IN GRECP STATUS
DSNB328I OBJECT NAME object-name OBJECT ID X'object-id' REMAINS IN GRECP STATUS.
USER ACTION IS NEEDED TO RECOVER THE OBJECT
```

Example D-4 Possible messages after automatic LPL recovery

```
DSNB317INO OBJECTS IN LPL STATUS
DSNB328I OBJECT NAME object-name OBJECT ID X'object-id' REMAINS IN LPL STATUS. USER
ACTION IS NEEDED TO RECOVER THE OBJECT
```

You can use these messages to understand the results of Db2 automatic recovery. You might want to automate responses to DSNB328I messages to reduce the impact that these types of object restrictions can have on your applications.

Asynchronous cross-invalidation of Group Buffer Pools

Db2 12 adds support for asynchronous cross-invalidation (XI) of GBPs, which can reduce elapsed time for certain application processes. Asynchronous XI support is available with z/OS 2.2 or later with z/OS APAR OA54688 applied and with coupling facility control code (CFCC) level (CFLEVEL) of 23 or greater.

Cross-invalidation is the process where Db2 writes newly changed pages to the GBP, and cross-system extended services (XES) sends messages to the LPARs where other Db2 members have a copy of one or more of those pages. Those other Db2 member copies are now out of date and must be invalidated. XI messages normally are synchronous with the process to write the changed pages to the GBP. In some cases, especially where CF LPARs are at some distance from the originating GBP, this synchronous process contributes to application delays.

If the XI messages can be sent asynchronously, the GBP writes do not wait for the XI messages to complete, and the application can proceed more quickly. If asynchronous XI support is available, Db2 evaluates XI messages and compares the numbers of pages in a page set that must be cross-invalidated concurrently with an internally defined threshold. If the number of pages is higher than the threshold, Db2 can request that the XI signals be sent asynchronously. Asynchronous XI can occur for deferred writes or force-at-commit writes to the GBP, but it does not occur if the data page size is 32 KB.

When asynchronous XI occurs, application processes do not need to wait until the XI completes. The primary benefit is for configurations where Db2 data sharing members are far from each other, although data sharing members that are close together can also benefit. You can observe the performance improvement in accounting class 2 elapsed times.

IRLM deadlock

The internal resource lock manager (IRLM) reduces the likelihood of workload stall conditions in situations where there are many waiters and deadlock detection is complex. APARs PH08431 and PH08708 combine to provide relief by timing waiters out more quickly and suppressing resultant timeout messages (DSNT376I). Db2 continues to produce the IFCID 196 timeout records for this situation.

For more information, see [PH08431](#) and [PH08708](#).

-ALTER GROUPBUFFERPOOL (Db2)

Db2 raises the limit of the **RATIO** option of the **-ALTER GROUPBUFFERPOOL** command from 255 to 1024, and changes the default for **RATIO** from 5 to 10. These changes reflect prevailing workload patterns of high page registration and the need for more directory entries.

For more information, see [-ALTER GROUPBUFFERPOOL \(Db2\)](#).

Sysplex support for multi-factor authentication and IBM RACF PassTickets

Db2 provides Sysplex support for distributed clients that access Db2 data sharing members to use either multi-factor authentication (MFA) or PassTickets. APARs PI94236 and PH21433 add support for Sysplex workload balancing and seamless fail-over scenarios. APAR PH21341 adds support for clients without Sysplex workload balancing.

You can enable your distributed clients to cache MFA or RACF PassTicket caching in a Db2 data-sharing environment. The clients can use MFA caching whether they use Sysplex workload balancing or seamless failover. The clients can use PassTicket caching only if Sysplex workload balancing is enabled. In both cases, the following conditions must be true:

- ▶ The ICSF load library, SCSFMOD0, is in the LINKLIST of the LPAR where Db2 is running.
- ▶ The **AUTHEXIT_CACHEREFRESH** subsystem parameter is set to ALL for all the members of the Db2 data sharing group.

Beyond these two requirements, the actions that you must take depend on whether the distributed clients are using Sysplex workload balancing.

You can implement MFA if you installed the IBM Z Multi-Factor Authentication product. For more information, see [IBM Z Multi-Factor Authentication](#).

If you use RACF PassTickets, see [Enabling Db2 to receive RACF PassTickets](#).

Sysplex group authentication with Sysplex workload balancing

Sysplex group authentication eliminates connection authentication failures that occur when a Db2 client enables Sysplex workload balancing or seamless failover by using either of the following methods:

- ▶ The **enableWLB** keyword in the **<wlb>** section of the IBM data server driver configuration file (**db2dsdriver.cfg**) for the non Java client
- ▶ The **enableSysplexWLB** JDBC database property for a Java client

For more information, see [Enabling caching of MFA-based authentication credentials for client with Sysplex workload balancing](#).

Multi-factor authentication without Sysplex workload balancing

For your distributed clients that do not use Sysplex workload balancing or seamless failover, you can specify the amount of time that MFA authentication credentials can be cached. The **MFA_AUTHCACHE_UNUSED_TIME** subsystem parameter determines the number of seconds that an authorization cache entry can remain unused. Choose a value 120 - 7200. For best results, use the same value for each member of the Db2 data sharing group.

For more information, see [Enabling caching of MFA and RACF PassTickets credentials for clients with Sysplex workload balancing](#).

Distributed Data Facility

The following sections summarize changes and enhancements to the Distributed Data Facility (DDF).

DDF messaging

APAR PH30222 adds important information to several messages and adds two messages:

- ▶ Db2 issues messages DSNL519I and DSNL523I at DDF startup or when the DDF service is restored. Before this APAR, the messages included only the main TCP/IP port for this DDF. These messages now include the main port, the resynchronization (resync) port, and the secure port (if defined).
- ▶ Db2 includes message DSNL093I as part of the output of the **-DISPLAY DDF DETAIL** command. Before this APAR, this message included only the number of pooled Database Access Threads (DBATs) and the number of inactive connections. This message now includes the number of active, in-use DBATs, which are DBATs that are currently processing client requests.

The extra fields in these updated messages help you to manage your DDF workload:

- ▶ DSNL077I is a new message that is issued when the current number of active, in-use DBATs exceeds 85% of the current value of the **MAXDBAT** subsystem parameter. **MAXDBAT** represents the maximum number of concurrently active DBATs, including in-use and disconnected DBATs.
- ▶ DSNL078I is a new message that displays when the current number of active, in-use DBATs no longer exceeds 75% of the value of **MAXDBAT**.

The combination of DSNL077I and DSNL078I provides greater insight into DDF activity.

DDF and system profile monitoring

Db2 12 improves your ability to monitor and manage distributed workloads with significant enhancements to system profile monitoring. With system profile monitoring, you have granular control over connections and threads when you populate the **DSN_PROFILE_TABLE** and **DSN_PROFILE_ATTRIBUTES** tables with the appropriate filters, keywords, and attributes.

With APAR PH12041, Db2 adds support for new counters for profile-related thread activity, the ability to request synchronous reads of IFCID 402 trace records, and the ability to control the queue depth for distributed threads.

Db2 produces IFCID 402 trace records for profile warnings or exceptions. These records now include the following information at the profile level:

- ▶ Current active thread counters
- ▶ Current suspended thread counters
- ▶ High water mark of thread counter since DDF started
- ▶ Current connection counter
- ▶ High water mark of connections counter since DDF started

A monitor program that uses the IFI READS function can now request synchronous data from IFCID 402 trace records. You can now monitor your DDF workload and act more quickly.

You can control the queue depth for distributed threads that exceed the value in the ATTRIBUTE2 column of the DSN_PROFILE_ATTRIBUTES table for the MONITOR THREADS keyword by specifying a value in ATTRIBUTE3. The value in the ATTRIBUTE3 column must be a whole number between zero and the value in the ATTRIBUTE2 column.

- ▶ If the ATTRIBUTE3 column value is zero, then Db2 does not queue requests that are waiting for a thread to complete. Instead, Db2 terminates the connection with a -30041 SQLCODE. You can use the value of zero to force the request to another member of a Db2 data sharing group.
- ▶ If the ATTRIBUTE3 column value is greater than zero, it represents the number of requests that Db2 queues waiting for a thread to complete. Use a nonzero value to allow some threads to queue and set a lower limit than the value in ATTRIBUTE2.
- ▶ If ATTRIBUTE3 is null, then Db2 queues the number of requests up to the value in ATTRIBUTE2.

With APAR PH30780, Db2 adds new keyword columns to monitor connections and threads from unknown or unspecified IP addresses. With this significant change, you can control floods of requests and keep them from overwhelming your high-priority distributed workload.

Db2 adds the following two keywords to the DSN_PROFILE_ATTRIBUTES table:

- ▶ MONITOR ALL CONNECTIONS
- ▶ MONITOR ALL THREADS

These keywords apply to profiles for the default location filtering criteria, where the LOCATION column of the DSN_PROFILE_TABLE contains '*', ':::0', or '0.0.0.0'. You can use these keywords to enforce warning and exception thresholds for the cumulative number of connections or threads from across all locations that match the default location filtering criteria. In other words, the values that you specify for MONITOR ALL CONNECTIONS or MONITOR ALL THREADS limit the total of number of connections or threads from locations that you do not specify in other profile rows.

For more information about Db2 system profile monitoring, see [Profiles for monitoring and controlling Db2 for z/OS subsystems](#).

Subsystem monitoring

Db2 adds several enhancements to improve subsystem monitoring:

- ▶ SYSPROC.ADMIN_INFO_IFCID stored procedure
- ▶ STATIME_MAIN subsystem parameter
- ▶ Reason for storage contraction in message DSNS005I
- ▶ IBM CICS attachment and client information special registers

SYSPROC.ADMIN_INFO_IFCID stored procedure

With the ADMIN_INFO_IFCID stored procedure, you can retrieve statistics information that is contained in IFCIDs 1, 2, or 225 with a **READS** call in a stored procedure. With this capability, you have more flexibility in monitoring key indicators of your Db2 subsystem or data sharing member.

For more information, see [ADMIN_INFO_IFCID stored procedure](#).

STATIME_MAIN subsystem parameter

With the **STATIME_MAIN** subsystem parameter, you can specify the granularity with which Db2 writes key statistics records. You can specify the recording interval in 5-second increments, from 5 to 60 seconds. The default is 60 seconds. Smaller recording intervals provide greater insight into workload fluctuations.

The following IFCIDs are included in these records:

- ▶ 0001 (system statistics)
- ▶ 0002 (Db2 statistics)
- ▶ 0202 (buffer pool attributes)
- ▶ 0225 (storage statistics)
- ▶ 0230 (GBP attributes)
- ▶ 0254 (CF cache statistics)
- ▶ 0369 (wait and CPU time aggregated by connection type)
- ▶ 0411 (remote application statistics)
- ▶ 0412 (remote user statistics)

Reason for storage contraction in message DSNS005I

Db2 issues message DSNV516I at the beginning of a storage contraction. Db2 now issues an extra message, DSNS005I, that includes the reason for the contraction.

IBM CICS attachment and client information special registers

CICS APAR PH30252 allows CICS tasks to pass extra fields to Db2 that identify the origins of CICS requests. Db2 adds support for these fields by using the value of these fields as the default values for the **CURRENT CLIENT_ACCTNG** and **CURRENT CLIENT_APPLNAME** special registers.

You can use these fields to filter specific data from Db2 accounting trace records (IFCID 0003).

For more information, see [CURRENT CLIENT_ACCTNG](#) and [CURRENT CLIENT_APPLNAME](#).

DSNZPARM simplification

Db2 12 simplifies subsystem parameters (DSNZPARM) by changing defaults or removing parameters.

Db2 12 changes the default values for the parameters that are shown in Table D-1 with APAR PH21370.

Table D-1 Changed defaults for subsystem parameters in Db2 12

Parameter name	Prior default or range	New default or range
CHKFREQ	5 minutes (range: 1 - 60)	3 minutes (range 1 - 5)
CHKMINS	1 - 1439 minutes	Less than 5 minutes
PCLOSET	10 minutes	45 minutes
MAXRBLK	400000 KB	1000000 KB

Parameter name	Prior default or range	New default or range
NPGTHRS	0	1
INLISTP	50	1000

The new default values represent best practices or reflect common usage.

Db2 12 removes the parameters that are shown in Table D-2. Db2 behaves according to implicit values, sets the highest value, or no longer uses the process.

Table D-2 Removed subsystem parameters in Db2 12

Parameter name	Implicit behavior	APAR
COMPRESS_SPT01	YES	PH24358
SPT01_INLINE_LENGTH	32138	PH24358
OBJECT_CREATE_FORMAT	EXTENDED	PH26317
UTILITY_OBJECT_CONVERSION	NOBASIC	PH26317
CACHEPAC	Highest value	PH28280
CACHERAC	Highest value	PH28280
IRLMAUT	YES	PH28280
IRLMSWT	200	PH28280
EDPROP	NO	PH28280
CHGDC	NO	PH28280
PCLOSEN	Only PCLOSET	PH28280
MGEXTSZ	YES	PH28280

For more information, see the appropriate APAR.

If you do not currently use the value that reflects the implicit behavior, you should verify that the change is compatible with your subsystem and applications by going to [Adjust subsystem parameter setting for parameters removed in Db2 12](#).

Security

Db2 12 provides new security capabilities for encryption and auditing and increases security by using existing interfaces.

- ▶ Encryption
- ▶ Auditing: tamper-proof audit policy

For more information about security enhancements for command-line processing (CLP), secure socket layer-only connections, coordination of SECADM and BINDAGENT privileges, improved cross-memory address space separation, and RACF synchronization with Db2 security caches, see “Security” on page 247.

Encryption

Db2 12 provides the following encryption-related enhancements:

- ▶ Encrypting authorization IDs and passwords in the SYSIBM.USERNAMES table for establishing remote connections
- ▶ Data set encryption
- ▶ Application-aware column encryption

DSNLEUSR stored procedure

You can use the DSNLEUSR sample Db2 stored procedure to encrypt authorization IDs and passwords in the SYSIBM.USERNAMES table by using Integrated Cryptographic Services Facility (ICSF) functions. Before APAR PI74797, the ICSF functions relied on a hardware cryptographic adapter to perform the encryption. With APAR PI74797, Db2 calls ICSF functions that use software encryption, which removes the dependency on a cryptographic adapter. In addition, DSNLEUSR now uses an AES algorithm for more secure encryption.

Data set encryption

Db2 11 introduced data set encryption support as part of pervasive encryption. Db2 12 function level 502 (FL 502) adds ways to specify key labels for encryption:

- ▶ The **ENCRYPTION_KEYLABEL** subsystem parameter for system objects (Db2 catalog, directory, and archive log data sets)
- ▶ SQL **CREATE** or **ALTER** statements with the **KEY LABEL** option for user objects

In each case, the action to allocate the data set, such as during a **REORG**, causes the data set to be encrypted.

Starting in FL 502, you can display key label information by using the **-DIS GROUP** command or the **REPORT** utility. If you encrypt the active or archive log data sets, you can display the key label with **-DIS LOG** or **-DIS ARCHIVE**.

The ADMIN_DS_LIST stored procedure can display data set encryption status and key label information. This enhancement, which is delivered in APAR PH12920, is available in the Db2 12 base and does not require a particular function level.

For more information about data set encryption, see [Db2 for z/OS Security with Data Set Encryption](#).

Column-level encryption built-in functions

Db2 12 FL 505 adds encrypt and decrypt built-in functions (BIFs). These functions provide column-based, application-aware encryption for string and numeric data types. These BIFs are based on ICSF and key label support.

An application invokes the encrypt and decrypt BIFs to protect sensitive data at the column level. The application uses the encrypt BIF to specify the expression to be encrypted, the key label, which identifies the key ICSF uses to encrypt the expression, and the encryption algorithm, as shown in Figure D-3.

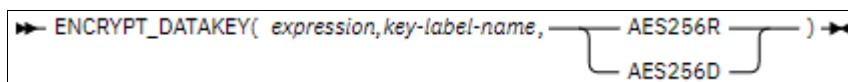


Figure D-3 ENCRYPT_DATAKEY syntax

The algorithms each use AES256 encryption. The choices of AES256R or AES256D specify whether a random (AES256R) or a fixed (AES256D) initialization vector is used:

- ▶ A random initialization vector produces a unique encrypted value every time it operates on a given input expression. If you use AES256R, you must decrypt the encrypted value before using it in a comparison.
- ▶ A fixed initialization vector produces the same value every time it operates on a given input expression. If you use AES256D, you can perform comparisons by using the encrypted value.

The data type of the encrypted value is either VARBINARY or BLOB depending upon the data type of the input expression. BIGINT, INTEGER, DECIMAL, CHAR, VARCHAR, GRAPHIC, and VARGRAPHIC input expressions produce VARBINARY data types. CLOB and DBCLOB input expressions produce BLOB data types.

When the encrypted value is decrypted, your application uses the decrypt BIF that corresponds to the original input data type. For example, if the encrypt BIF encrypted an expression with data type of VARCHAR, the application uses the BIF `DECRYPT_DATAKEY_VARCHAR` to retrieve the unencrypted value.

For more information about the `ENCRYPT_DATAKEY` BIF, see [ENCRYPT_DATAKEY](#).

For more information about decrypting by using BIFs, see [DECRYPT_DATAKEY_type](#).

For more information about Db2 encryption, see [Protecting data through encryption and RACF](#).

Auditing: tamper-proof audit policy

Function level 509 in Db2 12 introduces support for tamper-proof audit policies. The new tamper-proof feature provides the security administrator with the capability of creating an audit policy that cannot be tampered with, which enables auditing without the loss of audit information and compliance with privacy laws.

Creating a tamper-proof audit policy

You can create a tamper-proof audit policy by inserting an audit policy record into the `SYSIBM.SYSAUDITPOLICIES` catalog table with a `DB2START` value of 'T'. For more information, see [Creating and activating audit policies](#).

A newly inserted tamper-proof audit policy record with a `DB2START` column value of 'T' is started automatically during Db2 startup. If the audit trace must start immediately before Db2 restart, you must issue a `START TRACE` command.

For example, suppose Sara, a Db2 security administrator, wants to enable a tamper-proof audit policy, `TAMPERPRFPOLICY01`, before Db2 restarts. The following steps occur:

1. Sam, a z/OS security administrator, activates the RACF DSNR class and issues the RACF command `RACLIST` to the DSNR class if the command has not already been run.

Optionally, Sam defines a default profile, `DSNAUDIT.*`, in the RACF DSNR class that prevents any tamper-proof audit policy records from being modified or stopped. The RACF DSNR class must be refreshed afterward.

2. Sara creates a tamper-proof audit policy by inserting a record into the `SYSIBM.SYSAUDITPOLICIES` table with the `DB2START` column set to a new value 'T':

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
  (AUDITPOLICYNAME, CHECKING, VALIDATE, SYSADMIN, DB2START)
VALUES ('TAMPERPRFPOLICY01', 'A', 'A', 'I', 'T');
```


3. Sara starts the tamper-proof policy manually by using the **START TRACE** command:

```
STA TRACE(AUDIT) AUDTPLCY(TAMPERPRFPOLICY01)
```

Modifying or stopping a tamper-proof audit policy

To modify or stop a tamper-proof audit policy, you must be authorized to access the audit policy profile in an IBM z/OS security product that is external to Db2, such as RACF. A z/OS system security administrator must perform a special task in the external security product to grant you access to update, delete, or stop the audit policy. In addition to access to the profile, you must also have the privileges that are required for the statements and commands that you use to modify or stop the tamper-proof audit policy.

For example, suppose John, a Db2 security administrator, must update the tamper-proof audit policy TAMPERPRFPOLICY01 that is already started. John must first ask Sam, a z/OS security administrator to complete the following steps:

1. Create a profile in the RACF DSNR class for the tamper-proof audit policy and permit John access to the profile:

```
RDEFINE DSNR DSNAUDIT.TAMPERPRFPOLICY01 UACC(NONE) OWNER(DB2OWNER)
PE DSNAUDIT.TAMPERPRFPOLICY01 ID(JOHN) ACCESS(READ) CLASS(DSNR)
```

2. Refresh the RACF DSNR class.

John must then stop and restart the tamper-proof audit policy to pick up the policy updates. After John finishes with his update, the following steps must be done:

1. John issues the **STOP TRACE** and **START TRACE** commands to manually pick up the update:

```
STO TRACE(AUDIT) AUDTPLCY(TAMPERPRFPOLICY01)
STA TRACE(AUDIT) AUDTPLCY(TAMPERPRFPOLICY01)
```

2. Sam removes John's privileges for accessing the audit policy profile:

```
PE DSNAUDIT.TAMPERPRFPOLICY01 ID(JOHN) DELETE CLASS(DSNR)
```

3. Sam refreshes the RACF DSNR class to pick up the changes.

You cannot use the **SCOPE (GROUP)** option in a **START TRACE** or **STOP TRACE** command that starts or stops a tamper-proof audit policy. To start or stop a tamper-proof audit policy for all members of a data sharing group, issue the command on each data sharing member.

For more information, see [Updating tamper-proof audit policies](#).

Summary

Db2 12 for z/OS continuous delivery provides significant enhancements for system administration and system management. Improvements for performance, data sharing, DDF, monitoring, parameter simplification, encryption, and auditing combine to increase your ability to meet your organization's requirements for performance, scalability, and security.

Abbreviations and acronyms

ACAE	Access Control Authorization Exit	DLI	data language interface
AESE	Advanced Enterprise Server Edition	DML	Data Manipulation Language
AI	artificial intelligence	DPSI	data-partitioned secondary index
AOT	accelerator-only table	DR	disaster recovery
API	application programming interface	DRDA	Distributed Relational Database Architecture
APN	absolute page number	E3270	Enhanced 3270
APPLCOMPAT	application compatibility	EBCDIC	Extended Binary Coded Decimal Interchange Code
AREO	advisory REORG pending	ECSA	extended common service area
ATB	above-the-bar	ENFM	enabling new function mode
BIF	built-in function	ETL	extract, transform, and load
BMP	batch messaging processing	FFDC	First Failure Data Capture
BSAM	basic sequential access method	FL510	function level 510
BSDS	bootstrap data set	FTB	Fast Traversal Block
BTB	below-the-bar	GA	general availability
CCSID	coded character set identifier	GBP	Group Buffer Pool
CDB	communication database	GPP	general-purpose processor
CDC	Change Data Capture	HA	high availability
CEX7S	CryptoExpress 7S	HADR	high availability and disaster recovery
CF	coupling facility	HALOAD	High Availability Load
CFCC	coupling facility control code	HFS	Hierarchical File System
CFRM	coupling facility resource management	HIPERF	high-performance
CHKP	CHECK-pending	HTAP	hybrid transactional analytical processing
CICS	Customer Information Control System	IAM	identity access and management
CL509	catalog level 509	IAS	IBM Integrated Analytics System
CLP	command-line processing	IBM	International Business Machines Corporation
DASDs	direct-access storage disks	ICSF	Integrated Cryptographic Service Facility
Db2I	Db2 Interactive	IDE	integrated development environment
Db2ZAI	Db2 AI for z/OS	IDF	Integrated DRDA Facility
DBA	database administrator	IDMS	Integrated Database Management System
DBaaS	database-as-a-service	IDUG	International Db2 Users Group
DBAT	Database Access Thread	IFC	Instrumentation Facility Component
DBD	database descriptor	IFCID	Instrumentation Facility Component Identifier
DBDM	database descriptor management	IFL	Integrated Facility for Linux
DBET	Database Exception Table		
DBRM	database request module		
DCC	distributed connection control		
DDF	Distributed Data Facility		
DDL	Data Definition Language		

IMS	Information Management System	ROI	return on investment
IPLs	initial program loads	RPFC	remote pair FlashCopy
IRLM	Internal Resource Lock Manager	RPN	relative page number
ISV	independent software vendor	RPO	recovery point objective
ITOA	IT Operations Analytics	RRSAF	Resource Recovery Services attachment facility
IzODA	IBM Open Data Analytics for z/OS	RSM	Real Storage Manager
JDBC	Java Database Connectivity	RTO	recovery time objective
JSON	JavaScript Object Notation	RTS	real-time statistics
KDR	KeepDynamicRefresh	RUG	regional user group
LBI	Large Block Interface	SE	Specialty Engine
LOB	large object	SLA	service level agreement
LPAR	logical partition	SME	subject matter expert
LPL	logical page list	SMF	System Management Facilities
LRSN	log record sequence number	SORTL	SORT LISTS
LSTM	long short-term memory	SPE	small programming enhancement
LUW	Linux, UNIX, and Windows	SQLCA	SQL communication area
MDS	Mainframe Data Service	SQLREP	SQL Replication
MFA	multi-factor authentication	SRB	Service Request Block
ML	machine learning	SSC	Secure Service Container
NPI	Non-partitioned secondary index	SSL	secure sockets layer
OBD	object descriptor	STAT	START TRACE
ODBC	Open Database Connectivity	SWB	System Work Block or scheduler work block
OLAP	online analytics processing	SYSLOG	system log
OLTP	online transaction processing	TCO	total cost of ownership
ONNX	Open Neural Network Exchange	TSO	Time Sharing Option
PBG	partition-by-growth	UDF	user-defined function
PBR	partition-by-range	UDTF	user-defined table function
PI	partitioned index	UI	user interface
PIT	point-in-time	URI	uniform resource identifier
PMML	Predictive Model Markup Language	UTS	universal table space
PSID	Pageset ID	VIPA	virtual IP address
PTF	program temporary fix	VSAM	Virtual Storage Access Method
QMF	Query Management Facility	WIFI	wait-for-input
QREP	Q-Replication	WLM	workload manager
RACF	Resource Access Control Facility	WMLz	IBM Watson Machine Learning for z/OS
RBA	relative byte address	XES	cross-system extended services
RDS	Relational Data System	XI	cross-invalidation
RHOCP	Red Hat OpenShift Container Platform	XRC	Extended Remote Copy
RI	referential integrity	zAIO	IBM Z AI Optimization
RID	row ID	zBNA	IBM Z Batch Network Analyzer
RLE	Record List Entry	zDNN	IBM Z Deep Neural Network
RLF	resource limit facility	zHL	IBM Db2 zHyperLink
RNN	recurrent neural network		

zIIP

IBM Z Integrated Information
Processor

Redbooks

IBM Db2 13 for z/OS and More

SG24-8527-00

ISBN 0738460575



(0.5" spine)

0.475" x 0.873"

250 <-> 459 pages



SG24-8527-00

ISBN 0738460575

Printed in U.S.A.

Get connected

