DB2 10 for z/OS

What's New?

GC19-2985-13

DB2 10 for z/OS

What's New?

GC19-2985-13

Notes

Before using this information and the product it supports, be sure to read the general information under "Notices" at the end of this information.

October 13, 2017 edition

This edition applies to DB2 10 for z/OS (product number 5605-DB2), DB2 10 for z/OS Value Unit Edition (product number 5697-P31), and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Specific changes are indicated by a vertical bar to the left of a change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

Contents

About this information

This information provides an executive overview of new function in DB2® 10 for z/OS®. The topics in this information provide a framework for describing new function in DB2 for z/OS. New functions are categorized according to user benefits such as information on demand, availability, and performance.

In addition, this information summarizes changes that were introduced in this version for DB2 commands, DB2 utilities, SQL statements, the DB2 catalog, DB2 performance monitoring, and instrumentation facility component identifiers (IFCIDs).

Throughout this information, "DB2" means "DB2 10 for z/OS". References to other DB2 products use complete names or specific abbreviations.

Important: To find the most up to date content, always use IBM® Knowledge Center, which is continually updated as soon as changes are ready. PDF manuals are updated only when new editions are published, on an infrequent basis.

This information assumes that your DB2 subsystem is running in DB2 10 new-function mode.

Availability of new function in DB2 10
Generally, new SQL capabilities, including changes to existing functions, statements, and limits, become available only in new-function mode, unless explicitly stated otherwise. Exceptions to this general statement include optimization and virtual storage enhancements, which are also available in conversion mode unless stated otherwise. In DB2 Version 8 andDB2 9, most utility functions were available in conversion mode. However, for DB2 10, most utility functions become available in new-function mode.

Who should read this information

This information is written primarily for people who are evaluating and planning for DB2 for z/OS.

Terminology and citations

When referring to a DB2 product other than DB2 for z/OS, this information uses the product's full name to avoid ambiguity.

The following terms are used as indicated:

DB2 Represents either the DB2 licensed program or a particular DB2 subsystem.

Tivoli® OMEGAMON® XE for DB2 Performance Expert on z/OS
Refers to any of the following products:
- IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS
- IBM Tivoli OMEGAMON XE for DB2 Performance Monitor for z/OS
- IBM DB2 Performance Expert for Multiplatforms and Workgroups
- IBM DB2 Buffer Pool Analyzer for z/OS

C, C++, and C language
Represent the C or C++ programming language.

CICS® Represents CICS Transaction Server for z/OS.

IMS™ Represents the IMS Database Manager or IMS Transaction Manager.

MVS™ Represents the MVS element of the z/OS operating system, which is equivalent to the Base Control Program (BCP) component of the z/OS operating system.

RACF®
Represents the functions that are provided by the RACF component of the z/OS Security Server.

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 for z/OS documentation.

Send your comments by email to db2zinfo@us.ibm.com and include the name of the product, the version number of the product, and the number of the book. If you are commenting on specific text, please list the location of the text (for example, a chapter and section title or a help topic title).

Chapter 1. What's new in DB2 10?

DB2 10 for z/OS (also referred to as DB2 10 or DB2) delivers key innovations that reduce your total cost of ownership and increase productivity.

Improved operational efficiency for out-of-the-box savings

DB2 10 delivers great value by reducing CPU usage. Compared to previous releases of DB2 for z/OS, most customers can achieve out-of-the-box CPU savings of five to ten percent for traditional workloads and up to 20 percent for nontraditional workloads. DB2 10 reduces CPU usage by optimizing processor times and memory access, and by leveraging the latest processor improvements, larger amounts of memory, and z/OS enhancements. Improved scalability and constraint relief can add to the savings. Productivity improvements for database administrators and system administrators can drive even more savings.

For more information about the out-of-the-box performance savings that DB2 10 can provide, see Performance improvements from migrating to DB2 10.

Unsurpassed resiliency for business-critical information

Business resiliency is a key component of the value proposition of DB2 10 for z/OS and the z/OS operating system, supporting your efforts to keep your business running even during unexpected circumstances. Innovations in DB2 10 drive new value in resiliency through scalability improvements and fewer outages, whether those outages are planned or unplanned. In most cases, scalability improvements in DB2 10 deliver the ability to handle five to ten times more concurrent users in a single DB2 subsystem than in previous releases of DB2 for z/OS (as many as 20,000 concurrent active threads per subsystem). Improved availability is supported by schema evolution, or data definition on demand, and manageability enhancements for query performance.

For more information about scalability improvements, see Scalability.

Fast application and warehouse deployment for business growth

Staying competitive in today's global economy is tougher than ever, requiring agility, adaptability, and responsiveness. SQL and pureXML® enhancements in DB2 10 help extend usability, improve performance, and ease application portability to DB2 for z/OS. These improvements support your efforts to create a sustainable, competitive advantage.

For more information about SQL and pureXML enhancements, see SQL and XML.

Flexibility in migration paths

For this release, you can migrate a DB2 Version 8 subsystem or data sharing group in new-function mode to DB2 10. You also can fall back to DB2 Version 8 from DB2 10. In addition, DB2 DB2 Version 8 or DB2 9 can coexist with DB2 10 in a data sharing environment and in a distributed environment.

Supporting migration from either DB2 Version 8 or DB2 9 provides you with greater flexibility as you plan for DB2 for z/OS.

For more information about migration improvements, see Migration.

Performance improvements

In DB2 10, performance improvements focus on reducing CPU processing time without causing significant administration or application changes. Most performance improvements are implemented by simply migrating to DB2 10. You gain significant performance improvements from distributed data facility (DDF) optimization, buffer pool enhancements, parallelism enhancements, and more.

For more information about performance improvements, see Performance.

Continuous availability enhancements

DB2 10 introduces online schema enhancements that allow you to make changes to database objects (indexes and table spaces) while maximizing the availability of the altered objects. Through enhancements to ALTER statements, you can now change indexes and table spaces without having to unload the data, drop and re-create the objects, regenerate all of the security authorizations, re-create the views, and reload the data. Now, the changes are materialized when the altered objects are reorganized.

In addition, DB2 10 improves the usability and performance of online reorganization in several key ways. This release of DB2 for z/OS supports the reorganization of disjoint partition ranges of a partitioned table space, and improves SWITCH phase performance and diagnostics. Also, DB2 10 removes restrictions that are related to the online reorganization of base table spaces that use LOB columns.

For more information about availability enhancements, see Availability.

Increased concurrency and reduced catalog contention

In DB2 10 new-function mode, you can access currently committed data to dramatically minimize transaction suspension. Now, a read transaction can access the currently committed and consistent image of rows that are incompatibly locked by write transactions without being blocked. Using this type of concurrency control can greatly reduce timeout situations between readers and writers who are accessing the same data row. In addition, this new capability provides flexibility and increased performance to applications that require only available and committed data to be returned from DB2 tables.

Also, the DB2 catalog is restructured to reduce lock contention by removing all links in the catalog and directory. In addition, new functionality improves the lock avoidance techniques of DB2, and improves concurrency by holding acquired locks for less time and preventing writers from blocking the readers of data.

For more information about these enhancements, see Retrieving consistent data with improved concurrency and Reduction in catalog contention.

Virtual storage relief

In DB2 10, increased 64-bit storage utilization reduces below the bar virtual storage constraints and greatly improves scalability. For example, this release of DB2 for z/OS uses support for 64-bit run time, which provides virtual storage relief and can greatly improve the vertical scalability of your DB2 subsystem.

In addition, DB2 10 provides a 64-bit ODBC driver. The 64-bit ODBC driver runs in 64-bit addressing mode and reduces the virtual storage constraint, because the driver can accept 64-bit user data pointers and access user data above the 2 GB bar in the application address space.

Extended support for LOB and XML streaming, improvements to LOB and XML processing, and enhancements to the FETCH statement for LOB and XML data also contribute to reducing virtual storage consumption, and thereby, improving the stability of your DB2 subsystem.

For more information about scalability improvements, see Scalability.

Temporal tables and system-period data versioning

In this release of DB2 for z/OS, you have a lot of flexibility in how you can query data based on periods of time. DB2 10 introduces the *temporal table*, which is a table that records the period of time when a row is valid. DB2 supports two types of periods, which are the system period (SYSTEM_TIME) and the application period (BUSINESS_TIME). The system period consists of a pair of columns with system-maintained values that indicate the period of time when a row is valid. The application period consists of a pair of columns with application-maintained values that indicate the period of time when a row is valid. For use with the system period, DB2 10 introduces *system-period data versioning*, which specifies that old rows are archived into another table.

For more information about implementing temporal tables, see Creating temporal tables.

Security enhancements

This release of DB2 for z/OS provides critical enhancements to security and auditing. These enhancements strengthen DB2 security in the z/OS environment. For example, DB2 10 provides increased granularity for DB2 administrative authority, and offers a new a DB2 data security solution that enables you to manage access to a table at the level of a row, a column, or both. In addition, you can define and create different audit policies to address the various security needs of your business.

For more information about security enhancements, see Security and regulatory compliance.

pureXML improvements

This release of DB2 for z/OS substantially improves DB2 family consistency and productivity for pureXML. These improvements also optimize performance and include support for the binary XML format, XML schema validation as a built-in function, XML date and time data types and functions, XML type in native SQL routines, partial document update, and much more.

For more information about pureXML improvements, see XML.

Chapter 2. Performance

In DB2 10 of DB2 for z/OS, performance improvements focus on reducing CPU processing time without causing significant administration or application changes.

This release of DB2 for z/OS runs on only zEnterprise®, System z10®, System z9®, z990, and z890 or later processors, using z/OS Version 1 Release 10 or later operating systems. This configuration provides CPU reductions from the start. You will begin to see performance improvements in conversion mode, with more substantial CPU reductions for applications that can take advantage of the improvements in database design and application programming.

Performance improvements can be sorted into the following categories:

- Improvements that you receive by simply migrating to DB2 10
- Improvements that are implemented by DBA-level changes, without requiring application changes
- Improvements that require application changes
- Improvements from scalability enhancements

Performance improvements from migrating to DB2 10

By migrating to DB2 10, you gain significant performance improvements from distributed data facility (DDF) optimization, buffer pool enhancements, parallelism enhancements, and more. Some enhancements require that you rebind static applications.

Reduced CPU usage

DB2 10 delivers great value by reducing CPU usage.

Compared to previous releases of DB2 for z/OS, most customers can achieve out-of-the-box CPU savings of five to ten percent for traditional workloads and up to 20 percent for nontraditional workloads. DB2 reduces CPU usage by optimizing processor times and memory access, and by leveraging the latest processor improvements, larger amounts of memory, and z/OS enhancements. Improved scalability and constraint relief can add to the savings. Productivity improvements for database administrators and system administrators can drive even more savings.

Most performance improvements are implemented by migrating to DB2 10 and rebinding. For example, if you migrate and rebind plans or packages, the CPU savings can be twice as much than without rebinding. Also, you can use new keywords to rebind and keep the same access path.

Improved optimization techniques

DB2 10 introduces and improves several techniques for optimizing query performance.

Access path stability improvements

DB2 10 builds on the access path stability improvements that were offered in DB2 9. DB2 10 introduces the ability to specify the reuse of previous access paths for static SQL statements when you rebind packages. You can also detect access path changes for static SQL statements.

You can use the PLANMGMT bind option to save multiple copies of access paths. You can then use the SWITCH bind option to switch between different access paths for the same query. For both bind and rebind, you can use the APREUSE bind option to regenerate run time structures for static SQL statements without changing access paths. You can also use the APCOMPARE bind option to detect when access paths change when you bind or rebind packages for static SQL statements.

Related tasks:

- Managing and preventing access path change (DB2 Performance)

Related reference:

- APCOMPARE bind option (DB2 Commands)
- APREUSE bind option (DB2 Commands)
- PLANMGMT bind option (DB2 Commands)
- SWITCH bind option (DB2 Commands)

Safe query optimization

In DB2 10, the DB2 optimization process considers the uncertainty of predicate filtering when selecting an index.

In previous releases of DB2 for z/OS, query performance might be impacted if the DB2 optimization process overestimates the filtering of a predicate, which can occur if the literal value is not known. For example, when the predicate is a range predicate, when data is distributed non-uniformly, or when using host variables or parameter markers.

Now, when two indexes have a close cost estimate, the DB2 optimization process considers the uncertainty of matching and screening predicates when choosing the most efficient index. The DB2 optimization process might choose an index with a slightly higher cost estimate than another index, if that index will be more cost-effective.

In addition, this releases offers improvements to list prefetch processing. In previous releases of DB2 for z/OS, when DB2 exhausted RID pool resources or exceeded RID limits, DB2 reverted to table space scans. However, in DB2 10, when DB2 exhausts the RID pool resources, the RID list will write and continue RID pool processing by using work file resources.

In previous releases of DB2 for z/OS, list prefetch was disabled for tables that are defined with the VOLATILE option. However, in DB2 10, list prefetch is now allowed in most cases.

Related concepts:

- List prefetch (PREFETCH='L' or 'U') (DB2 Performance)

Improved index matching for OR and IN predicates

DB2 10 supports index matching on multiple IN-list predicates, list prefetch, and predicate transitive closure for IN-list predicates. For complex OR predicates that

are not transformed to an IN predicate, DB2 for z/OS supports single matching index access, improving the efficiency of retrieving a small set of rows from anywhere in a result set.

This release also introduces the range-list index scan, which is a method for simplifying the processing of OR predicates that can be mapped to a single index. This access type improves the performance of applications with data-dependent pagination.

When your SELECT statement contains an OR predicate, DB2 can use a range-list index scan to avoid scanning the index multiple times. A single index scan consumes fewer RID list resources than multiple index scans and reduces CPU overhead.

DB2 can use a range-list index scan when the SELECT statement meets the following requirements:

- Every OR predicate refers to the same table
- Every OR predicate has at least one matching predicate
- Every OR predicate is mapped to the same index

Related concepts:

Index access (ACCESSTYPE is 'I', 'IN', 'I1', 'N', 'MX', or 'DX') (DB2 Performance)

Range-list index scan (ACCESSTYPE='NR') (DB2 Performance)

Parallelism enhancements

DB2 10 eliminates many restrictions to parallelism and uses access methods that guarantee work is distributed more evenly across each parallel task.

The following parallelism enhancements are provided when the parallel mode is "C" (CP parallelism):

- Parallelism is now enabled when a work file resulted from view materialization, or table expression materialization.
- DB2 now runs a parallel query group for join when DB2 chooses multiple column hybrid join with sort composite.
- Parallelism is now enabled for multi-row fetch, if the cursor is declared as read-only.
- Parallelism is allowed when the leading table is sort output and the join between the leading table and the second table is multiple column hybrid join.

Related tasks:

Programming for parallel processing (DB2 Performance)

Early application of stage 2 predicates

In DB2 10, the DB2 optimization process can apply non-matching predicates to reduce the number of qualified rows, before the data pages are accessed.

DB2 can apply many stage 2 expressions and scalar functions as non-matching predicates, or stage 1 non-indexed predicates. These predicates cannot be index matching.

Related concepts:

Stage 1 and stage 2 predicates (DB2 Performance)

Collection of autonomic statistics

DB2 10 introduces autonomic collection of statistics on tables and indexes.

DB2 10 introduces a set of stored procedures that determine if statistics need to be collected (or recollected), and then autonomically performs the collection. This enhancements helps to ensure that DB2 has accurate statistics to use when optimizing queries. This enhancement also helps to ensure that statistics are not recollected unnecessarily.

For tables that change so frequently that it is not practical to recollect statistics after every change, DB2 can dynamically obtain index filtering statistics during query optimization to assist in access path selection.

Related concepts:

Autonomic statistics overview (DB2 Performance)

Support for high performance database access threads

Starting in DB2 10, the MODIFY DDF command option PKGREL determines whether DB2 honors the RELEASE bind option for database access threads.

Before DB2 10, the RELEASE bind option had no effect on database access threads. For database access threads, only RELEASE(COMMIT) processing was performed. Starting in DB2 10, when you issue the MODIFY DDF PKGREL(BNDOPT) or MODIFY DDF PKGREL command, you can modify this behavior. After you issue the command, the use of processor resources for package allocation and deallocation are minimized for packages that use database access threads and are bound with RELEASE(DEALLOCATE).

Related tasks:

Controlling allocation and deallocation processing for database access threads (DB2 Performance)

DDF optimization for OPEN, FETCH, and CLOSE requests

DB2 10 provides improvements in overall application performance for result sets from SELECT statements that contain the FETCH 1 ROW ONLY clause.

DB2 now optimizes the processing of this SELECT statement by combining the open cursor (OPEN), prefetch query data (FETCH), and close cursor (CLOSE) requests into a single request. This single request flows between the distributed relational data system (DRDS) and the relational data system (RDS) subcomponents, rather than three individual requests.

When a JDBC or CLI application calls the respective driver API to execute the query with the FETCH FIRST 1 ROW ONLY clause, the default driver settings declare the cursor as WITH HOLD, if you did not explicitly specify a cursor declaration. After the result row for the query is fetched, the DB2 server closes the cursor, regardless of whether the cursor is declared as WITH HOLD. Then, the DB2 server sends notification to the driver that cancels any additional FETCH or CLOSE requests.

I/O parallelism for index updates

DB2 10 provides the ability to perform insert operations in parallel on multiple indexes that are defined on the same table.

I/O parallelism for index insert operations improves workload performance by overlapping the synchronous I/O wait time for different indexes that are defined on the same table. With this functionality, DB2 can prefetch pages in parallel from different indexes that are defined on the same table and place them into a buffer pool, reducing the overall elapsed time for insert operations.

For example, the processor performs the insert operation on the first index, and if it encounters I/O wait time, continues to the next index and so on. After all of the I/O requests have been initiated for the indexes, the processor revisits the indexes that encountered I/O wait time. By this time, the I/O requests most likely have finished, and the insert operation can complete without waiting. With I/O parallelism, the processor is not waiting for I/O, but there is still only one processing task.

Related tasks:

Enabling index I/O parallelism for INSERT operations (DB2 Performance)

Improvements to index access

Compared to previous releases of DB2 for z/OS, DB2 10 offers significant reductions in the class 2 CPU time that is used for predicate evaluation, especially in complex queries that return a large number of rows.

Buffer pool enhancements

Buffer pool enhancements in DB2 10 enable you to take advantage of larger buffer pools. Also, buffer pool enhancements reduce latch contention issues that are common in environments with large buffer pools.

In previous versions of DB2 for z/OS, storage was allocated for the entire size of a buffer pool, even if no data was accessed in any table space or index using that buffer pool. Now, buffer pool storage is allocated on-demand as data is brought in.

If a query touches only a few data pages, only a small amount of buffer pool storage is allocated. For a query that performs index-only access, the buffer pool for the table space does not need to have any buffer pool storage allocated.

To reduce the cost of accessing a buffer pool, DB2 requests 1 MB pages for buffer pools that are defined with the PGFIX(YES) option, if your operating system is defined with large page frames.

In addition, DB2 provides a new buffer pool page stealing algorithm that you can use to specify objects using that buffer pool as *in-memory objects*. The PGSTEAL(NONE) option in an ALTER BUFFERPOOL command specifies that data for in-memory objects is preloaded into the buffer pool at the time the object is physically opened.

Related tasks:

Fixing a buffer pool in real storage (DB2 Performance)

Choosing a page-stealing algorithm (DB2 Performance)

Related reference:

-ALTER BUFFERPOOL (DB2) (DB2 Commands)

Related information:

Buffer storage allocation (DB2 10 for z/OS Performance Topics)

In-memory table spaces and indexes (DB2 10 for z/OS Performance Topics)

Support for z/OS enqueue management

DB2 10 uses IBM Workload Manager for z/OS enqueue management to more effectively manage lock holders and waiters.

Related concepts:

z/OS performance options for DB2 (DB2 Performance)

Related reference:

MVS Planning: Workload Management (MVS Planning: Workload Management)

Reduction in need for explicit REORG

DB2 10 provides several performance enhancements that reduce the need to reorganize indexes frequently, resulting in a reduction in CPU time and synchronous I/O waits.

Compared to DB2 9, in DB2 10 you might see increased activity in deferred writes because more buffers are in use by buffer pools to improve performance. For example, activity might increase for index root pages that are pinned in a buffer pool when the index page set or index partition is opened. Activity also might increase for space map pages to avoid an exhaustive scan of the buffer pool when going in and out of group buffer pool dependency (GBP-dependency). In these cases, you might want to increase the corresponding buffer pool size or increase the deferred write thresholds.

List prefetch of index leaf pages

Previously when an index was disorganized, performance could be affected, because more gaps existed between index leaf pages. However, this release of DB2 for z/OS provides the functionality to perform a list prefetch of index leaf pages based on non-leaf page information. This enhancement can greatly reduce synchronous I/O waits for long-running queries that are accessing disorganized indexes. This enhancement also helps utilities that need to access index leaf pages to extract the index key values by reducing the number of synchronous I/O waits and elapsed time. These utilities include REORG INDEX, CHECK INDEX and RUNSTATS.

Sequential detection and index lookaside for referential constraints

When you insert data into a dependent table, DB2 accesses the parent key to check for referential constraints. New functionality enables sequential detection and index lookaside for INSERT statements that are subject to referential constraints. This enhancement reduces the CPU time of insert workloads that involve referential constraints.

IFCID support for index page split

DB2 10 introduces support for IFCID 359. IFCID 359 records an index page split. This IFCID makes monitoring index page splits easier, which can impact application performance. This IFCID stores the following information:

- Database ID

- Page set ID
- Partition number of the splitting index
- Splitting page number
- Start timestamp of the split
- End timestamp of the split

Related tasks:

Reorganizing indexes (DB2 Administration Guide)

Universal table space enhancements

The default table space type for a partitioned table space has changed from the non-universal partitioned table space to the range-partitioned universal table space. Using universal table spaces serves to improve performance and availability, because universal table spaces support DB2 10 enhancements, such as hash access and inline LOB columns.

Introductory concepts

Universal (UTS) table spaces (Introduction to DB2 for z/OS)

Non-universal partitioned table spaces are still supported. You can create non-universal partitioned table spaces by specifying the SEGSIZE clause with a value of 0 on the CREATE TABLESPACE statement. Also, you can use a new subsystem parameter on panel DSNTIP7 to specify the default segment size to be used for a partitioned table space when the CREATE TABLESPACE statement does not include the SEGSIZE parameter.

When you create a partition-by-growth universal table space, the CREATE TABLESPACE statement lets you specify the number of partitions to be created. Also, you can add partitions by using the ALTER TABLE ADD PARTITION statement for a partition-by-growth universal table space.

Streaming LOBs and XML

This release of DB2 for z/OS offers additional performance improvements by extending the support for LOB and XML streaming to avoid full materialization of large LOBs or XML data.

This improvement reduces virtual storage consumption, and improves system stability. Class 2 CPU time is also reduced. Whether materialization is reduced, and by how much, depends on the type of operation and the quantity and size of the large LOBs or XML data. Typically, large LOBs are considered to be 2 MB or greater in size, and XML data would need to be 32 KB or greater in size to experience benefits.

In general, this improvement eliminates materialization in the following situations:

- When inserting a single LOB or XML value to a row
- When updating a single LOB or XML value to a row
- When the LOAD utility uses file reference variables
- When CCSID conversion is required by the LOAD utility
- When inserting XML data in a DRDA environment at the remote server

Performance enhancements for local Java and ODBC applications

DB2 10 offers performance improvements for Java™ and ODBC applications that run locally on a z/OS operating system.

You will notice performance improvements for queries that return more than one row and for queries that return LOBs and XML documents.

Backup and recovery enhancements

DB2 10 supports enhancements for the use of FlashCopy® technology at the data set level for both backup and recovery. FlashCopy can provide significant reduction in both elapsed time and CPU consumption.

You can create object-level FlashCopy image copies by using the COPY, REORG, LOAD, and REBUILD INDEX utilities. Then, these image copies can be used by the RECOVER utility. Running the COPY and LOAD utilities with the SHRLEVEL CHANGE option provides you with the ability to create consistent FlashCopy image copies with full data availability. This goal is accomplished by copying the object, and then backing out uncommitted changes. The COPYTOCOPY utility supports the creation of sequential image copies from FlashCopy image copies.

In addition, the RECOVER utility provides an option to backout changes from the current state of the data when recovering data to a prior point in time. Data can be available faster than if the most recent recovery base, prior to the point that was recovered to, is restored and forward log processing is used.

Related tasks:

Creating FlashCopy image copies (DB2 Administration Guide)

Enhanced monitoring of statement-level statistics for static and dynamic SQL statements

DB2 10 introduces the ability to collect statement level statistics for both dynamic and static SQL statements.

Monitor trace class 29 enables the collection of system-wide statement-level statistics for dynamic SQL statements in IFCID 0316, and for static SQL statements in IFCID 0401. You can create READS programs for the instrumentation facility interface (IFI) to capture and analyze these statistics.

Related concepts:

Performance trace (DB2 Performance)

Related tasks:

Monitoring the dynamic statement cache with READS calls (DB2 Performance)

Collecting statement-level statistics for SQL statements (DB2 Performance)

Monitoring static SQL statements with READS calls (DB2 Performance)

Performance improvements from DBA-level changes

This release provides a number of performance improvements that can be implemented by database administrators, without requiring any application changes.

Related concepts:

Query optimization with IBM Data Studio

Optimization of inline LOBs

In DB2 10, you can improve the performance of applications that access LOB data by specifying that a portion of the value for a LOB column be stored in the base table space.

An *inline LOB* is the portion of a LOB column that resides in the containing table's base table space rather than in the column's auxiliary table space. For LOBs that are less than or equal to the size of the specified inline length, DB2 stores the complete LOB data in the base table space. DB2 does not need to access the LOB table space or auxiliary indexes for processes that access the LOB data.

For LOBs that are greater in size than the specified inline length, the inline portion of the LOB resides in the base table space, and DB2 stores the remainder of the LOB in the LOB table space. In this case, any process that accesses the LOB data must access both the base table space and the LOB table space.

You can alter the inline length for a LOB column. The LOB data is moved appropriately between the base table space and the LOB table space during the next process that modifies a row. Additionally, the next REORG operation moves the LOB data for all rows.

You can create a LOB column as an inline LOB by explicitly specifying the new INLINE LENGTH clause on CREATE TABLE or CREATE TYPE statements. You also can alter an existing LOB column to become an inline LOB by using the ALTER TABLE statement. Alternately, the inline length attribute can be inherited from the new LOB_INLINE_LENGTH field on the DSNTIPD installation panel.

Also, you can create an expression-based index on the inline portion of a LOB column. This new expression-based index can help with the performance of locating LOB data.

Inline LOB columns can only exist in a universal table space.

Related concepts:

Expression-based indexes (Introduction to DB2 for z/OS)

Related tasks:

Improving performance for LOB data (DB2 Performance)

Improved fast access to individual rows

DB2 10 introduces a new access path for faster access to individual rows by using a fully qualified key rather than a traditional index. This access path is called *hash access*.

Hash access can reduce CPU time and potentially I/O wait time. When DB2 selects a hash access path, only one I/O is required to retrieve the row from the table,

which reduces CPU usage. A table space with hash organization might require additional storage to minimize the number of overflow records.

Hash access also can eliminate the need for table space scans and index scans for access to a single unique key.

Hash access is efficient for certain types of tables and queries, such as:

- Queries that use equal predicates to access a single row on a table
- Tables of a predictable and reasonably static size. To create an effective hash algorithm, DB2 needs a close estimate of the volume of data that is expected to be in a table before the table is populated.

Restriction: If a table is organized for hash access, index clustering is unavailable on that table. Clustering keys cannot be defined on tables that are organized for hash access.

Related concepts:

Database design with hash access (Introduction to DB2 for z/OS)

Hash access paths (Introduction to DB2 for z/OS)

DB2 hash spaces (Introduction to DB2 for z/OS)

Related tasks:

Organizing tables by hash for fast access to individual rows (DB2 Performance)

Managing space and page size for hash-organized tables (DB2 Performance)

Creating tables that use hash organization (DB2 Administration Guide)

Altering tables to enable hash access (DB2 Administration Guide)

Additional non-key columns in a unique index

DB2 10 provides the ability to alter and create unique indexes that contain additional non-key columns.

In previous versions of DB2, when a table is created with a unique constraint, a unique index must enforce the unique constraint. This unique index can be implicitly or explicitly created. However, only those columns that are used for the unique constraint can be specified.

In an effort to achieve better query performance on unique tables, database administrators sometimes create additional indexes that contain the same columns as the unique index, but the additional indexes are appended with additional columns to support index-only access to those columns. These additional indexes might achieve performance improvements, but at the cost of added overhead for maintaining and storing the additional indexes.

Now, DB2 10 expands the index functionality of new-function mode by adding the optional INCLUDE clause to the CREATE INDEX and ALTER INDEX statements. The use of INCLUDE columns is supported only on unique indexes, with the purpose of decreasing the index maintenance and the physical storage that is required for additional indexes.

Related tasks:

Adding columns to the set of index keys of a unique index (DB2 Administration Guide)

Support for the MEMBER CLUSTER option

Partition-by-growth universal table spaces and range-partitioned universal table spaces now support the MEMBER CLUSTER option. The MEMBER CLUSTER option is effective in reducing page latch contention during concurrent insert operations in a data sharing environment.

The MEMBER CLUSTER option specifies that data that is inserted by an insert operation is not clustered by the implicit clustering index (the first index), or the explicit clustering index. DB2 locates the data in the table space based on available space. When you alter the MEMBER CLUSTER structure for a table space, the change results in a pending definition change for the table space.

Related concepts:

Member affinity clustering (DB2 Data Sharing Planning and Administration)

Performance improvements requiring application changes

In this release, application changes are required to take advantage of some performance improvements.

Extended support for the SQL procedural language

In DB2 10, support for the SQL procedural language is extended to include the development of SQL scalar functions and SQL table functions. Support for SQL table functions is new in this release.

You can develop and debug user-defined SQL scalar functions that contain logic for DB2 for z/OS with ease, even if you do not have much programming experience. In addition, you can develop user-defined SQL table functions with a RETURN statement that returns the result table from a SELECT statement.

These improvements enable you to port functions from other DB2 products or competitive database products to DB2 for z/OS with minimal cost.

Related concepts:

Steps to creating and using a user-defined function (DB2 Application programming and SQL)

Dynamic statement cache enhancements

In DB2 10, when dynamic statement caching is active, you can specify that DB2 exclude the literal constants that are in dynamic SQL statements when searching for a statement text match in the dynamic statement cache. This enhancement results in more sharing and reuse of cached statements for your applications.

This enhancement to dynamic SQL statement sharing is provided by the new CONCENTRATE STATEMENTS WITH LITERALS clause that is within the ATTRIBUTES clause of the PREPARE SQL statement. When you specify this new clause in a PREPARE statement and DB2 determines that there is no match in the dynamic statement cache for the dynamic SQL statement, DB2 replaces certain literal constants with the ampersand character (&) in the dynamic SQL statement. Then, DB2 repeats the cache matching search by using this new version of the SQL statement. If DB2 does not find a statement text match in the cache, DB2 inserts a new statement into the cache. The SQL statement text that is inserted into the cache contains ampersand characters in place of the original literal constants.

When an application runs an instance of this dynamic SQL statement that has different literal constants, DB2 searches the cache and replaces the literal constants with ampersand characters. Now, DB2 finds matching statement text in the cache during the repeated cache search. If the additional cache matching criteria is satisfied, the new instance of the dynamic SQL statement that has different literal constants will share or reuse the cached version of the SQL statement that contains ampersand characters. In this example, your application avoids unnecessary preparation processes, and another statement is not inserted into the dynamic statement cache. These benefits can improve the overall performance of applications that use these statements and reduce storage growth in both the cache and DB2 storage pools.

This statement sharing enhancement does not apply to dynamic SQL statements that contain both parameter markers (?) and literal constants.

If a dynamic statement does not qualify to be inserted in the dynamic statement cache, a statement identifier is not generated for that statement. For example, dynamic DDL statements and the dynamic LOCK TABLE statement do not qualify for the dynamic statement cache. In DB2 10, to help with problem determination and diagnostics, a predefined statement identifier with a value of 1 is reserved for dynamic DDL statements and the dynamic LOCK TABLE statement.

DB2 10 also provides JDBC and ODBC properties for this enhancement so that you do not need to specify the new clause on the PREPARE statement. For IBM Data Server Driver for JDBC and SQLJ type 2 connectivity, the new connection property is **statementConcentrator**. For ODBC drivers, the new keyword is LITERALREPLACEMENT.

Related concepts:

Conditions for statement sharing (DB2 Performance)

Reoptimization for statements with replaced literal values (DB2 Performance)

Related tasks:

Improving dynamic SQL performance by enabling the dynamic statement cache (DB2 Performance)

Enabling dynamic SQL statement caching for ODBC function calls (DB2 Programming for ODBC)

Related reference:

PREPARE (DB2 SQL)

Common IBM Data Server Driver for JDBC and SQLJ properties for Db2® servers (DB2 Application Programming for Java)

Access to currently committed data

In DB2 10 new-function mode, you can access currently committed data to dramatically minimize transaction suspension.

Now, a read transaction can access the currently committed and consistent image of rows without being blocked, even if the rows are incompatibly locked by write transactions. This new capability provides flexibility and increased performance to applications that require only available and committed data to be returned from DB2 tables.

This new function is supported by a new bind option, CONCURRENTACCESSRESOLUTION, and two new PREPARE statement clauses, USE CURRENTLY COMMITTED and WAIT FOR OUTCOME, which you can specify in the *attribute-string*. In addition, this function is supported by the new SKIPUNCI subsystem parameter.

Access to currently committed data is available only on universal table spaces and applies only to row-level and page-level locks.

Related tasks:

Accessing currently committed data to avoid lock contention (DB2 Performance)

Related reference:

SKIP UNCOMM INSERTS field (SKIPUNCI subsystem parameter) (DB2 Installation and Migration)

CONCURRENTACCESSRESOLUTION bind option (DB2 Commands)

PREPARE (DB2 SQL)

Performance improvements from scalability enhancements

The scalability enhancements that you gain from migrating to DB2 10 also contribute to improved performance for DB2 for z/OS.

Reductions in log latch contention

DB2 10 greatly reduces the use of log latch.

For most cases, DB2 holds the log latch for less time when creating log records, because of the increased use of compare and swap logic. This enhancement does not require configuration or application changes.

This change provides reductions in transaction and CPU time when logging rates are high, and offers improvements to transaction throughput and DB2 vertical scalability. Because of this change, you might see increased logging rates compared to previous releases of DB2 for z/OS, if current workloads have significant log latch contention.

Chapter 3. Scalability

This release of DB2 for z/OS substantially increases the amount of virtual storage, while reducing storage monitoring. These improvements and others result in reduced cost, improved productivity, and easier management and growth of DB2.

Related concepts:

Reductions in log latch contention

Reduced catalog contention

In DB2 10, the DB2 catalog is restructured to reduce lock contention by removing all links in the catalog and directory.

Some utilities that operated on links are no longer needed in DB2 10 new-function mode. Whereas, other operations that you could not do on catalog tables that contained links are now possible in DB2 10 new-function mode.

Each table in the following table spaces is also moved to its own partition-by-growth table space in new-function mode:

- DSNDB06.SYSOBJ
- DSNDB06.SYSDBASE
- DSNDB06.SYSVIEWS
- DSNDB06.SYSPLAN
- DSNDB06.SYSPKAGE
- DSNDB06.SYSDBAUT
- DSNDB06.SYSGROUP
- DSNDB01.DBD01
- DSNDB01.SPT01
- DSNDB01.SYSUTILX
- SYSIBM.SYSDATABASE
- SYSIBM.SYSDBAUTH
- SYSIBM.SYSPLANAUTH

Many table spaces use row-level locking, and all tables are in reordered-row format.

Related concepts:

Objects that are subject to locks (DB2 Performance)

Elimination of UTSERIAL lock for DB2 utilities

DB2 10 eliminates the use of UTSERIAL lock by DB2 utilities. This enhancement prevents the majority of timeouts on the global UTSERIAL lock resource.

In new-function mode, DB2 utilities no longer acquire the UTSERIAL lock to serialize utility access to the DSNDB01.SYSUTILX table space. Access to this table space is governed by granular page locks, which significantly improves utility concurrency for compatible utilities. Serialization between incompatible utilities is unaffected by this change.

Increased size limitation for SPT01

In DB2 10, you can specify whether the SPT01 table space is to be compressed.

Use the subsystem parameter COMPRESS_SPT01 in macro DSN6SPRM to specify this behavior. Valid values are YES and NO. In a data sharing environment, all members should use the same setting for the COMPRESS_SPT01 parameter. The default value is NO, which means that the SPT01 table space is not compressed.

The SPT01 table space in the directory database (DSNDB01) is a partition-by-growth table space. As in previous releases of DB2 for z/OS, this table space still has a maximum partition size of 64 GB, and the maximum number of partitions is set to 1. However, the DSNDB01.SPT01 table space uses CLOB and BLOB columns, providing more space for growth.

Because of these enhancements, bind and rebind processing is less likely to fail due to lack of space in the DSNDB01.SPT01 table space.

Related reference:

COMPRESS SPT01 field (COMPRESS_SPT01 subsystem parameter) (DB2 Installation and Migration)

Utilization of 64-bit run time

DB2 10 moves much more memory above the 64-bit bar, which provides virtual storage relief and can greatly the improve the vertical scalability of your DB2 subsystem. To take advantage of the 64-bit run time support, packages need to be rebound on DB2 10.

Plans and packages that were created before DB2 10 and that contain static SQL statements that use parallelism run differently in DB2 10. DB2 incrementally rebinds those packages and plans after migration to DB2 10. Incremental rebinds can cause performance degradation, so you should manually rebind those plans and packages. In addition, to get the most benefit from virtual storage relief, the native SQL procedures that you created on DB2 9 need to be regenerated on DB2 10.

You can run a query in the DSNTIJPM job before you migrate to determine which packages can use parallelism, and are possible candidates for incremental rebinds. You should consider rebinding those packages after migration, as soon as your DB2 10 system is stable. After you migrate to DB2 10, you can also run a performance trace, class 3 or class 10 for IFCID 360, to identify the plans and packages that contain static SQL queries that use query parallelism, and therefore, need to be rebound.

Related tasks:

Run premigration queries (DSNTIJPM) (DB2 Installation and Migration)

Work file enhancements

Work file enhancements in DB2 10 improve scalability and reduce the CPU time for workloads that execute queries that require the use of small work files.

DB2 10 supports partition-by-growth table spaces in the WORKFILE database. Also, DB2 10 provides in-memory work file enhancements in the WORKFILE

database, adding more opportunity to use in-memory work files than in DB2 9. In the WORKFILE database, DB2 supports simple predicate evaluation for work files.

In addition, the records of work files that are created for joins and large sorts can span multiple pages. Work file records that span multiple pages can accommodate larger record lengths and larger sort key lengths for sort records. The maximum length of a work file record is 65,529 bytes.

Related reference:

DSNTIP9: Work file database panel (DB2 Installation and Migration)

Support for extended address volumes (EAV)

z/OS Version 1 Release 11 introduces extended address volumes (EAV) support for extended format sequential data sets and VSAM data sets.

With EAV support, the maximum number of cylinders on a volume is 262,668. This means that you can store more DB2 data on a single volume. However, the maximum amount of data that you can store in a single data set of a DB2 table space or index space is the same for EAV and non-extended address volumes, which is 64 GB.

Related concepts:

Storage servers and advanced features (DB2 Performance)

Support for deleting data sharing members

Starting in DB2 10, you can permanently remove a member from a data sharing group by deleting the member.

This enhancement provides many benefits. For example, removing members that you do not need can simplify disaster recovery, because you no longer need to maintain copies of the bootstrap data sets (BSDS) for those members. In addition, you can delete members that you added by mistake, and you can more easily consolidate members to take advantage of more powerful zSeries processors.

To delete a member from a data sharing group, all members of that data sharing group must be stopped. This group-wide outage is required, because all of the BSDSs in the data sharing group must be updated at the same time.

Related tasks:

Deleting data sharing members (DB2 Data Sharing Planning and Administration)

Chapter 4. Availability

DB2 10 continues to offer improvements in availability by allowing more online schema changes. As a result of these changes, database administrators can make changes and do maintenance online without bringing down a DB2 subsystem.

Online schema enhancements

This release allows you to make changes to database objects (indexes and table spaces) while maximizing the availability of the altered objects.

Through enhancements to ALTER statements, you can now change indexes and table spaces without having to unload the data, drop and re-create the objects, regenerate all of the security authorizations, re-create the views, and reload the data. Now, the changes are materialized when the altered objects are reorganized. These online schema enhancements apply to the following types of changes:

- Altering the page size of a table space
- Altering the data set size of a table space
- Altering the segment size of a table space
- Altering the table space type, as follows:
  - Changing a simple table space with only one table to a partition-by-growth universal table space
  - Changing a segmented table space with only one table to a partition-by-growth universal table space
  - Changing a non-universal partitioned table space to a range-partitioned universal table space
- Altering the MEMBER CLUSTER structure for a table space
- Altering the index page size

This concept of *pending definition changes* is supported by the introduction of the database exception table (DBET) state AREOR, the addition of new catalog tables, and enhancements to certain utility functions. In addition, the DROP PENDING CHANGES clause is added to the ALTER TABLESPACE statement to support dropping pending definition changes.

Related tasks:

Altering table spaces (DB2 Administration Guide)

Online REORG enhancements

DB2 10 improves the usability and performance of online reorganization in several key ways.

This release of DB2 for z/OS supports the reorganization of disjoint partition ranges of a partitioned table space, and improves SWITCH phase performance and diagnostics. Also, DB2 10 removes restrictions that are related to the online reorganization of base table spaces that use LOB columns. However, if you altered the inline length for any LOB column, you cannot use online REORG until after that inline change has been materialized.

In new-function mode, the syntax for the REORG TABLESPACE statement has been enhanced. For partitioned table spaces, the PART specification is extended to allow for multiple parts or part ranges, and the SHRLEVEL REFERENCE and CHANGE specifications are extended to add a new keyword, **AUX YES/NO**. This new keyword allows for the inclusion of associated LOB table spaces in the reorganization of base data. This enhancement removes restrictions pertaining to the movement of base data rows between partitions by the REORG utility, and also reduces the need for explicit specification of LOB table spaces in LISTDEF statements for this utility.

Related concepts:

DB2 online utilities (DB2 Utilities)

Adding an active log data set to the active log inventory

In DB2 10, you can use the SET LOG command to add a new active log data set to the active log inventory without stopping DB2.

In previous versions, you had to stop DB2 to add a new active log data set to the active log inventory. Now, you can issue the SET LOG command and specify the NEWLOG and COPY keywords. If DB2 can open the newly defined log data set, the log is added to the active log inventory in the bootstrap data set (BSDS). The new active log data set is immediately available for use without recycling DB2.

When you add active log data sets, the limit is 93 data sets for each log copy.

Related tasks:

Adding an active log data set to the active log inventory with the SET LOG command (DB2 Administration Guide)

Retrieving consistent data with improved concurrency

New functionality improves the lock avoidance techniques of DB2. These lock avoidance techniques improve concurrency by holding acquired locks for less time and preventing writers from blocking the readers of data.

In DB2 10 new-function mode, you can access currently committed data to dramatically minimize transaction suspension. Now, a read transaction can access the currently committed and consistent image of rows that are incompatibly locked by write transactions without being blocked. Using this type of concurrency control can greatly reduce timeout situations between readers and writers who are accessing the same data page or row. In addition, this new capability provides flexibility and increased performance to applications that require only available and committed data to be returned from DB2 tables.

This functionality is supported by a new BIND option, CONCURRENTACCESSRESOLUTION, and two new PREPARE statement clauses, WAIT FOR OUTCOME and USE CURRENTLY COMMITTED, which you can specify in the *attribute-string*.

When you specify CONCURRENTACCESSRESOLUTION(WAITFOROUTCOME), if a lock contention occurs, the read transaction waits for the release of the lock by the write transaction. Specifying CONCURRENTACCESSRESOLUTION(USECURRENTLYCOMMITTED) allows

applications to access the currently committed data even if a lock contention is encountered. This option is applicable on scans that are accessing tables that are defined in a universal table space.

Related tasks:

Accessing currently committed data to avoid lock contention (DB2 Performance)

Point-in-time recovery enhancements

In DB2 10, you can use the RECOVER utility with the `BACKOUT YES` option to recover data to a previous point in time by backing out committed work. Using this option might decrease the amount of time that an object is unavailable during a recovery to a prior point in time, if the recovery point in time that you select is relatively recent.

The `BACKOUT YES` option specifies using the log to backout changes that were made since the log point that is specified in the RECOVER syntax by the `TOLOGPOINT` or `TORBA` options. Any uncommitted work at the specified log point is backed out to make the objects transactionally consistent.

With the `BACKOUT YES` option, the changes are backed out from the current state of the object. No image copy is restored.

Related reference:

RECOVER (DB2 Utilities)

Increased availability for CHECK utilities

DB2 10 provides increased availability and consistency for the CHECK DATA utility and the CHECK LOB utility.

In previous releases of DB2 for z/OS, the CHECK DATA and CHECK LOB utilities might set restrictive states on the table space after completion. Now, if these utilities find inconsistencies and the object was not restricted before, a restrictive state is not automatically set on the table space. The new CHECK_SETCHKP subsystem parameter specifies whether the CHECK DATA and CHECK LOB utilities are to place inconsistent pagesets in CHECK-pending status.

Related concepts:

DB2 online utilities (DB2 Utilities)

Support for rotating any logical partition

In DB2 10, you can use the ALTER TABLE statement to rotate any logical partition to become the last partition. Rotating partitions is supported for partitioned (non-universal) table spaces and range-partitioned table spaces, but not for partition-by-growth table spaces.

Related reference:

ALTER TABLE (DB2 SQL)

Online member-specific location aliases

You can define DDF location aliases that enable you to manage subsets of data sharing members dynamically without stopping and restarting DDF or DB2. Applications also can use such aliases and ports in non-data sharing environments, eliminating the need for future application changes when converting to a data sharing environment.

You can use the MODIFY DDF command with the ALIAS keyword to define, manage, and delete location aliases dynamically. You also can specify member-specific IP addresses for location aliases that are created in this manner. By doing so, you can control the IP addresses that are returned in the server list and used for future connections to the server through different alias names.

Location aliases of this type cannot be managed by the DSNJU003 utility, and the DSNJU004 utility does not provide any information about them. However, you can use the DISPLAY DDF command to obtain information about these aliases. You can still define as many as 8 static location aliases by using the DSNJU003 utility, but you cannot use the MODIFY DDF command to manage those location aliases, and you cannot define IP addresses for those aliases.

Related tasks:

Defining dynamic location aliases (DB2 Data Sharing Planning and Administration)

Online communications database enhancements

DB2 10 improves the availability of the communications database for TCP/IP connections.

In DB2 10, updates to the SYSIBM.LOCATIONS, SYSIBM.IPNAMES, and SYSIBM.IPLIST tables in the communications database take effect for new remote connection requests, without any need to stop and restart DDF. Existing connections are not affected.

Related reference:

Remote data sharing group requirements (DB2 Data Sharing Planning and Administration)

Optional TCP/IP domain names

The DB2 distributed data facility (DDF) no longer requires that you provide a TCP/IP domain name, as long as the IP address is provided by using the DB2 bootstrap data set (BSDS).

Related tasks:

Connecting systems with TCP/IP (DB2 Installation and Migration)

Chapter 5. Security and regulatory compliance

In DB2 10, improvements to security and regulatory compliance focus on data retention, and protecting sensitive data from privileged users and administrators. Improvements also help to separate security administration from database administration.

Support for row and column access control

Row and column access control is a DB2 data security solution that enables you to manage access to a table at the level of a row, a column, or both.

Implemented through row permissions and column masks, row and column access control is data-centric, policy-driven, and flexible. Unlike multilevel security, row and column access control is integrated into a database system and places the security logic where the data is. It effectively eliminates the need to filter security-sensitive data at the application level and ensures that the data is protected regardless of the applications and tools that are used to access it. All applications and tools that access the database are automatically subject to the same control.

Unlike views, row and column access control is based on a security policy that specifies the rules and conditions under which a user, group, or role can access the rows, the columns, or both of a table. No user has an implicit exemption from these rules. If the SEPARATE SECURITY system parameter on panel DSNTIPP1 is set to `YES` during installation, only a security administrator (SECADM) can activate the row and column access control for a table and manage access to the table.

Row and column level access control provides the following advantages:

- Integration within the database system
- Database level security
- SQL enforced security that does not require other products to monitor access
- Access that is managed by the DB2 security administrator
- Multiple access levels based on users, groups, or roles
- Row and column access control with filtering and data masking
- No requirement to filter sensitive data at the application level

Related information:

Managing access through row permissions and column masks (Managing Security)

Administrative privileges with finer granularity

DB2 10 implements new authorities and privileges that are designed to help businesses comply with government regulations and to simplify the management of authorities. The concepts of *separation of duties* and *least privilege* address these needs.

Separation of duties provides the ability for administrative authorities to be divided across individuals without overlapping responsibilities, so that one user does not possess unlimited authority, such as with SYSADM authority. For

example, the new system DBADM authority provides a user with the ability to manage any table, but without granting access to the data that is in the table. The new SECADM authority provides a user with the ability to manage access to a table in DB2, but that user cannot create, alter, or drop the table. Also, that user cannot access the data within the table. These new authorities are provided to minimize the need of SYSADM authority in day-to-day operations.

Least privilege allows an administrator to delegate part of his responsibilities to other users, without providing extraneous privileges. Under this model, database users might share administrative duties, while ensuring a high level of security.

In addition, to make sure that administrative authorities are not misused, DB2 10 provides new capability to audit the administrative authorities. An *audit policy* is a set of criteria that determines the categories to be audited. Audit policies help you configure and control the audit requirements of your security policies and to monitor data access by applications and individual users (authorization IDs or roles), including administrative authorities.

You can define and create different audit policies to address the different security needs of your business. New auditing features include the ability to dynamically enable the auditing of all static and dynamic statements against a table, and the ability to audit any use of a system or database authority.

Related concepts:

DB2 audit policy (Managing Security)

Related information:

Managing administrative authorities (Managing Security)

Support for new z/OS security features

DB2 10 supports new z/OS security features that help satisfy your security and compliance requirements.

Enabled for z/OS client login using digital certificates

DB2 10 is enabled to support z/OS client login through the use of digital certificates.

Previously, RACF users of traditional, remote client applications authenticated themselves to DB2 by providing their user IDs and passwords. Now, by using z/OS digital certificates, the Secure Socket Layer (SSL) protocol supports server and client authentication during the handshake phase. This enhancement enables a server to validate the certificates of a client at the server, preventing the client from obtaining a secure connection without an installation-approved certificate. The authentication of the remote client's digital certificate is performed for DB2 by Application Transparent Transport Layer Security (AT-TLS) that is provided with the z/OS Communications Server TCP/IP stack.

Support for z/OS client login by using digital certificates also provides the benefit of RACF certificate name filtering. A *certificate name filter* enables you to associate many client certificates with one user ID based on the unique user information in the certificate, such as the user's affiliation. You can create one or more certificate name filters to map a large number of client certificates to a limited number of user IDs, which helps you reduce administrative costs.

Support for z/OS RACF password phrase

This release of DB2 for z/OS supports RACF password phrases that were introduced in the z/OS Version 1 Release 10 Security Server feature. Password phrases have security advantages over traditional, 8-character passwords, because password phrases are more secure, while being easier to remember.

A password phrase is a character string that consists of mixed-case letters, numbers, and special characters, including blanks. When the new-password-phrase exit (ICHPWX11) is present and allows it, a password phrase can be 9 to 100 characters in length. When ICHPWX11 is not present, the password phrase must be 14 to 100 characters in length.

Support for z/OS identity propagation

Today, many transactions that run on a DB2 subsystem originate outside of z/OS from the internet and are initiated by users who authenticate their identities on web-based, or distributed, application servers. When a distributed application server establishes a connection to a DB2 subsystem, the connection might be associated with the identity of the distributed application user, as defined in a user registry where the transaction originated, or it might be associated with a shared RACF user ID that was assigned by the z/OS subsystem. To be effective, customers that audit user activities on DB2 subsystems need both the RACF user ID that is associated with the connection and the user identity that was presented when the user originally accessed the distributed application server.

Beginning with z/OS Version 1 Release 11, when you implement z/OS identity propagation, you use the RACF RACMAP command to map the user's distributed identity to a RACF user ID. This mapping function is accomplished by using distributed identity filters. This enhancement allows both user identities to be recorded in the SMF audit records that are written by RACF during the execution of supported transactions, providing more complete auditing for z/OS subsystems, such as DB2.

Also beginning with z/OS Version 1 Release 11, RACF accepts information about the identities of distributed users from authorized applications that issue the RACROUTE REQUEST=VERIFY request.

Related concepts:

Encrypting your data with Secure Socket Layer support (Managing Security)

Related tasks:

Implementing DB2 support for distributed identity filters (Managing Security)

Related reference:

New-password-phrase exit (ICHPWX11) (z/OS Security Server RACF System Programmer's Guide)

Security improvements from managing application data based on time

DB2 10 provides the capability to implement table-level specifications to control the management of application data based on time.

Now, application programmers can write queries that specify a search criteria based on the time that data existed. This function simplifies and reduces the cost

of developing DB2 applications that require system-period data versioning. Also, this improvement allows you to meet new compliance laws faster and more economically, because DB2 automatically manages the different versions of data.

Related concepts:

Support for temporal tables and system-period data versioning

Related tasks:

Creating temporal tables (DB2 Administration Guide)

Chapter 6. Application integration

In this release, many of the new distributed functions of DB2 are delivered through service-oriented architecture (SOA).

SOA is an application development paradigm as well as an application framework that facilitates code reuse, application system integrity, interoperability, and service management. To accomplish tasks, DB2 must work with IBM Data Studio, web services, and many other application development tools. Also, many languages and access techniques are required to support Java, .NET, C/C++, SQL procedures, Ruby on Rails, PHP, Perl, Python, and new and traditional languages.

This version of DB2 for z/OS delivers a number of key enhancements to application integration.

Enhanced monitoring support

This release improves the support for monitoring within DB2 for z/OS by providing additional performance and diagnostic monitoring capabilities.

DB2 10 enhances performance monitoring support and monitoring support for problem determination for both static and dynamic SQL. This new support uses the Instrumentation Facility Interface (IFI) to capture and externalize monitoring information for consumption by tooling.

In order to facilitate the collection and correlation of enhanced monitoring data, this release introduces a unique statement execution identifier (STMTID). The statement ID is defined at the DB2 for z/OS server, returned to the DRDA application requester, and captured in IFCID records for both static and dynamic SQL. Through DRDA, the statement ID is returned to the client drivers, along with a compilation source identifier and a compilation time.

To support problem determination, the statement ID is provided in several existing messages, including messages related to deadlocks and timeouts. In these messages, the STMTID is associated with thread information. You can use this thread information to correlate the statement execution on the server with the client application on whose behalf the server is executing the statement.

To support performance monitoring, some existing trace records that deal with statement-level information are modified to capture the new statement ID and new statement-level performance metrics. Also, this release introduces new trace records that provide access to performance monitoring statistics in real time, and allow tooling to retrieve monitoring data without requiring disk access.

In addition, you can use profiles to monitor server threads and connections. You can monitor server threads and connections to analyze the use of system resources by particular clients, applications, and users and prioritize resources accordingly.

Related tasks:

Monitoring performance (DB2 Performance)

Support for 64-bit ODBC driver

In DB2 10, a 64-bit ODBC driver is now available. This new driver allows 64-bit ODBC applications to take advantage of the expanded 16 million TB address space.

The 64-bit ODBC driver runs in 64-bit addressing mode and reduces the virtual storage constraint, because the driver can accept 64-bit user data pointers and access user data above the 2 GB bar in the application address space.

The 64-bit ODBC driver is XPLINK only. This new driver is shipped in addition to the 31-bit ODBC drivers (non-XPLINK and XPLINK) that are currently supported by DB2. The APIs for existing 31-bit applications have not changed and continue to work using the 31-bit ODBC drivers.

Related concepts:

The DB2 ODBC run time environment (DB2 Programming for ODBC)

DRDA support of Unicode encoding for system code pages

This release of DB2 for z/OS includes DRDA support of Unicode encoding (code page 1208) for system code pages, such as DRDA command parameters and reply message parameters.

This enhancement can provide improved response time and less processor usage for remote CLI and JDBC applications by removing the need for the drivers to convert DRDA instance variables between EBCDIC and Unicode.

Related concepts:

DRDA character type parameters in Unicode (DB2 Internationalization Guide)

Elimination of DDF private protocol

This release of DB2 for z/OS does not support DDF private protocol.

DB2 provides the private to DRDA protocol REXX tool (DSNTP2DP) for preparing a DB2 subsystem to use only DRDA access. If you want to migrate packages or plans to DB2 10, you should convert any package or plan to DRDA protocol in DB2 9 that was bound explicitly or implicitly with private protocol. Packages and plans that were bound with private protocol in DB2 9 and access remote locations do not execute successfully in DB2 10.

You can continue to use private protocol access for DB2 9 members of a data sharing group until all members have migrated to DB2 10 conversion mode.

Related reference:

The private to DRDA protocol REXX tool (DSNTP2DP) (DB2 Installation and Migration)

Addition of extended indicator variables

DB2 10 introduces extended indicator variables that specify that no value is provided for a target column for INSERT, UPDATE, and MERGE statements.

This functionality makes application development much easier, because you do not need to prepare separate INSERT statements for every combination of columns that

are being inserted. By enabling the extended indicator variables, you specify that the target column of the host variable will be set to a defined, DEFAULT value, or that the value of the host variable is UNASSIGNED. The UNASSIGNED value means that the target column of the host variable is to be treated as if the target column was not included in the statement. With these values, you do not need to resubmit the current value for a column, or know the default value for a column.

In addition to reducing the number of queries that you need to write, this functionality improves performance by eliminating the impact on the cache that would be caused by the extra statements. Also, the overhead of compiling the extra statements is removed.

Related concepts:

Indicator variables, arrays, and structures (DB2 Application programming and SQL)

New Universal Language Interface module (DSNULI)

DB2 10 provides the new Universal Language Interface module (DSNULI) that you can use to simplify application maintenance.

When an application that is link-edited with DSNULI runs, DSNULI determines the runtime environment, and dynamically loads and branches to the appropriate language interface module. DSNULI obviates the need to maintain multiple copies of an application for use with different language interface modules. In addition, DSNULI simplifies application programming, because application programmers do not need to call a specific language interface module.

You can link-edit the DSNULI module with applications that use the following attachment facilities:

- CAF (call attachment facility)
- CICS (Customer Information Control System)
- TSO (Time Sharing Option)
- RRS (Resource Recovery Services)

When an application calls DSNULI instead of directly loading a specific language interface module, a small performance degradation might result when a small number of number of SQL statements are executed.

Related concepts:

Universal language interface (DSNULI) (DB2 Application programming and SQL)

IBM Data Server Driver for JDBC and SQLJ type 2 connectivity enhancements

This release adds function to IBM Data Server Driver for JDBC and SQLJ type 2 connectivity to DB2 for z/OS that can improve query performance for local Java applications.

IBM Data Server Driver for JDBC and SQLJ type 2 connectivity to local DB2 for z/OS data servers uses more efficient methods for processing forward only, read-only queries. These methods include the use of:

- Limited block fetch protocol

- True progressive streaming for retrieval of LOB and XML data
- Query buffers that reside in 64-bit addressable storage for applications that run in a 64-bit JVM

Related concepts:

Supported drivers for JDBC and SQLJ (DB2 Application Programming for Java)

Chapter 7. XML

This release of DB2 for z/OS substantially improves DB2 family consistency and productivity for XML.

DB2 10 provides key improvements in the function, performance, and usability of pureXML, which was first delivered in DB2 9.

Addition of XML type modifier

With DB2 10, you can add an XML type modifier to an XML column.

Represented by one or more XML schemas, the XML type modifier enforces the validity of XML data. DB2 will automatically perform schema validation during insert, update, and load operations of XML data, according to the schemas that are in the type modifier.

Related concepts:

XML schema validation with an XML type modifier (DB2 Programming for XML)

XML schema validation

DB2 10 provides XML schema validation as a built-in function, and makes schema validation eligible for 100% redirection to IBM Z Integrated Information Processor (zIIP) and IBM Z Application Assist Processor (zAAP) specialty engines.

With this new built-in function, you do not need to specify an XML schema for validation. You can let DB2 find one based on the target namespace and the optional schema location in XML documents.

DB2 allows multiple versions of an XML schema to exist in the XML schema repository, so that existing XML documents do not need to be changed when the XML schema changes. DB2 can choose an older version of the XML schema for validation of an older document, and a newer version of the XML schema for validation of a newer document.

Related concepts:

XML schema validation (DB2 Programming for XML)

XML consistency checking with CHECK DATA

DB2 10 adds functionality to the CHECK DATA utility, so that you can use this utility to verify the consistency of XML documents that are stored in a separate XML table space.

The CHECK DATA utility verifies that all nodes in that XML document are structurally intact and that the node ID index is consistent with the content that is in the XML table space. In addition, this utility verifies that all of the XML documents of an XML column are valid against at least one XML schema that is specified in the XML type modifier.

Related reference:

CHECK DATA (DB2 Utilities)

Support for multiple versions of XML documents

In DB2 10, multiple versions of an XML document can coexist in an XML table. The existence of multiple versions of an XML document can lead to improved concurrency through lock avoidance.

In addition, multiple versions can save real storage by avoiding a copy of the old values in the document into memory in some cases.

Now, fetching a row in a work file with an XML column successfully retrieves the version of the XML document that corresponds to the time when the row was inserted into the work file. Also, readers do not need to lock XML documents, improving concurrency for update operations.

DB2 supports multiple versions of an XML document in an XML column if the base table space for the table that contains the XML column is a universal table space, and all other XML columns in the table support multiple versions.

Related concepts:

XML versions (DB2 Programming for XML)

Support for updating part of an XML document

DB2 10 supports the ability to update part of an XML document.

To update part of an XML document in an XML column, you can use the SQL UPDATE statement with the XMLMODIFY built-in scalar function.

The XMLMODIFY function specifies a *basic XQuery updating expression* that you can use to insert nodes, delete nodes, replace nodes, or replace the values of nodes in XML documents that are stored in XML columns.

The types of basic XQuery updating expressions are:

insert expression
Inserts copies of one or more nodes into a designated position in a node sequence.

replace expression
Replaces an existing node with a new sequence of zero or more nodes, or replaces a node's value while preserving the node's identity.

delete expression
Deletes zero or more nodes from a node sequence.

For a large document, only the records that are impacted by the update will have versions. Also, only the impacted part of the document will be revalidated against a schema, if there is an XML type modifier on the updated column.

Support for binary XML

This release of DB2 for z/OS supports the use of binary XML format between applications and the server.

The binary XML format is formally called Extensible Dynamic Binary XML DB2 Client/Server Binary XML Format. Support of binary XML provides performance improvements, because the XML data can be encoded more efficiently. Binary XML is more efficient for various reasons, such as:

- Binary XML uses a pre-tokenized format, and all values are a prefixed length. There is no need to look at every byte, or search for the end of element names for values.
- Binary XML uses string IDs for names. A string ID can represent some or all occurrences of the same name with an integer identifier.

Retrieval of data as binary XML is supported only in JDBC, SQLJ, or ODBC applications. Binary XML is also supported by the LOAD and UNLOAD utilities.

Related information:

IBM Binary XML Specification

Support for XML date and time

DB2 10 includes date and time support for XML data types and functions.

This enhancement provides time zone features, and arithmetic and comparison operators on date and time data types. In addition, XML indexes support date and time data types.

The following date and time data types are supported:

- xs:dateTime
- xs:time
- xs:date
- xs:duration
- xs:yearMonthDuration
- xs:datTimeDuration

The following XML functions and operators support working with date and time data:

- Comparison operators on duration, date, and time values
- Component extraction functions on duration, date, and time values
- Arithmetic operators on duration values
- Time zone adjustment function on date and time values
- Arithmetic operators on duration, date, and time values
- Context functions

Support for XML in routines

DB2 10 includes support for XML as the data type for parameters and a return data type in native SQL procedures, SQL scalar functions, and SQL table functions.

Also, native SQL procedures and SQL scalar functions support XML as the data type for SQL variables. With XML variables, you can parse a document once and use that document in many SQL statements for more sophisticated business logic.

Support for DEFINE(NO) for LOB and XML table spaces

DB2 10 supports deferring the physical creation of LOB and XML table spaces and their associated indexes to improve space management and application installation time.

With this enhancement, you can create LOB and XML table spaces and dependent index spaces with the DEFINE(NO) option. By specifying DEFINE(NO), the physical creation of underlying VSAM data sets is deferred until the first INSERT or LOAD operation. The undefined LOB or XML table spaces and dependent index spaces still have a DB2 catalog entry, but these objects are considered to be empty when accessed by SELECT or FETCH operations.

Support for XQuery

DB2 10 provides support for XQuery expressions.

XQuery provides a superset of XPath functionality. With XPath you can select a sequence of nodes from an existing XML document. XQuery adds constructors for creating XML structures within a query, and FLWOR expressions for iterating and for binding of variables to intermediate query results.

Support for XQuery increases consistency of pureXML support across the DB2 family by providing syntax that is similar to the syntax that is used by DB2 for Linux, UNIX, and Windows.

You can use XQuery expressions in the following contexts:
- XMLQUERY, XMLTABLE, or XMLMODIFY built-in functions
- XMLEXISTS predicate

Related concepts:

Expressions (DB2 Programming for XML)

Transform XML with an XSL style sheet

The user-defined function, XSLTRANSFORM, provides support for transforming an XML document with XSL Transformation processing.

The Java user-defined function, SYSFUN.XSLTRANSFORM, transforms well-formed XML documents that are stored in DB2 using the XL TXE-J processor in the IBM SDK for z/OS. Input parameters are as follows:

1. An expression that returns a single-rooted, well-formed XML document.
2. An XSL style sheet that conforms to the W3C XSLT Version 1.0 Recommendation.
3. An expression that returns an XML document that contains parameter values to the XSL style sheet.

The output of the user-defined function is a CLOB.

Related concepts:

Transforming an XML document with XSLTRANSFORM (DB2 Programming for XML)

Related reference:

XSLTRANSFORM (DB2 SQL)

Chapter 8. SQL

In DB2 10, SQL enhancements focus on adding new function and ensuring SQL consistency across the DB2 family of products.

Support for temporal tables and system-period data versioning

In DB2 10, you can create a *temporal table*, which is a table that records the period of time when a row is valid.

DB2 supports two types of periods, which are the application period (BUSINESS_TIME) and the system period (SYSTEM_TIME). An application period consists of a pair of columns with application-maintained values that indicate the period of time when a row is valid. A table with only an application period is called an *application-period temporal table*.

The system period consists of a pair of columns with system-maintained values that indicate the period of time when a row is valid. The system period is meaningful because you can define *system-period data versioning* on a table that has this period. System-period data versioning specifies that old rows are archived into another table. The table that contains the current active rows of a table is called the *system-period temporal table*. The table that contains the archived rows is called the *history table*. You can delete the rows from the history table when those rows are no longer needed, if you have the correct authorization.

When you define a base table to use system-period data versioning, or when you define system-period data versioning on an existing table, you must create a history table, specify a name for the history table, and create a table space to hold that table. You define system-period data versioning by issuing the ALTER TABLE ADD VERSIONING statement with the USE HISTORY TABLE clause.

A *bitemporal table* is a table that is both a system-period temporal table and an application-period temporal table. You can use a bitemporal table to keep application period information and system-based historical information. Therefore, you have a lot of flexibility in how you query data based on periods of time.

Related concepts:

Security improvements from managing application data based on time

Related tasks:

Creating temporal tables (DB2 Administration Guide)

Enhanced support for SQL scalar functions

DB2 10 provides enhanced support for SQL scalar functions. This enhancement results in two different types of SQL scalar functions: *inline SQL scalar functions* and *non-inline SQL scalar functions*.

Inline SQL scalar functions support the same capability as in prior releases. An inline SQL scalar function has a body with a single RETURN statement.

Non-inline SQL scalar functions can contain logic and are written by application developers by using SQL statements and SQL control statements. The

enhancements that are provided with non-inline SQL scalar functions make it easier to write applications and to port applications from the DB2 family or competitive database products. In addition, extensions allow you to debug a non-inline SQL scalar function by using the Unified Debugger.

Non-inline SQL scalar functions include the following support for versioning and source code management:

- Define multiple versions of an SQL scalar function, where one version is considered the "active" version
- Activate a particular version of an SQL scalar function
- Alter the routine options that are associated with a version of an SQL scalar function
- Define a new version of an SQL scalar function by specifying the same function signature as the current version, and different routine options and function body
- Replace the definition of an existing version by specifying the same function signature as the current version, and different routine options and function body
- Drop a version of an SQL scalar function
- Fall back to a previous version without requiring an explicit rebind or recompile

You can deploy non-inline SQL scalar functions to multiple servers to allow a wider community to use functions that have been thoroughly tested, without the risk of changing the logic in the routine body.

Related concepts:

SQL scalar functions (DB2 Application programming and SQL)

Support for SQL table functions

DB2 10 introduces support for SQL table functions. You can develop user-defined SQL table functions, and you can define SQL table functions to behave as parameterized views.

An SQL table function is a function that is written exclusively in SQL statements and returns a single result table. You can use an SQL table function to:

- Define a parameter for a transition table (for example, the TABLE LIKE ... AS LOCATOR syntax)
- Define a parameter as a built-in type or a distinct type
- Include a single SQL PL RETURN statement that returns a result table

The CREATE statement for an SQL table function is an executable statement that can be dynamically prepared only if DYNAMICRULES run behavior is implicitly or explicitly specified.

The ALTER statement for an SQL table function can be embedded in an application program or issued interactively. The ALTER statement is an executable statement that can be dynamically prepared only if DYNAMICRULES run behavior is implicitly or explicitly specified.

Related concepts:

SQL table functions (DB2 Application programming and SQL)

Enhanced support for native SQL procedures

DB2 10 provides enhanced support for native SQL procedures, which were introduced in DB2 9. This enhancement makes it easier to port applications from other members of the DB2 family or from competitive database products.

An enhanced native SQL procedure can define a parameter or SQL variable as a distinct type, define an SQL parameter or an SQL variable as an XML data type, and make limited use of scrollable cursors.

If you create native SQL procedures in DB2 10 before migrating to new-function mode, those native SQL procedures have the same functionality as objects that were created in a prior release.

Related tasks:

Creating a stored procedure (DB2 Application programming and SQL)

Extended support for implicit casting

DB2 10 delivers enhancements that extend support for implicit casting. These enhancements significantly simplify the porting of data to DB2 for z/OS from other database management systems, where implicit casting is supported.

Previous releases of DB2 for z/OS already support implicit casting in many cases. However, previous releases do not support implicit casting between a character or graphic string type and a numeric type, because character or graphic string types and numeric types are not compatible. In these cases, you must manually code applications to explicitly cast a character or graphic string value to a specific numeric type, before assigning it to a numeric target or comparing it with a numeric value. This process can be time consuming and complicated.

Now, DB2 10 supports the implicit conversion of values from a character or graphic string data type to a numeric data type, or from a numeric data type to a character or graphic string data type. For example, when two operands of an operation, such as assignment or comparison operation, have incompatible data types, DB2 implicitly converts one operand to a data type that is compatible with the other. This enhancement extends the compatibility rules so that character or graphic string data types and numeric data types become compatible. Therefore, a numeric value can be assigned directly to a target of a character or graphic string data type. In addition, a character or graphic string value that is cast to a numeric data type also can be assigned to a target of a numeric data type, or be compared with a numeric value.

LOB data types and binary string types still are not compatible with numeric data types. Casting between these data types is not supported.

Related concepts:

Casting between data types (DB2 SQL)

Greater timestamp precision for applications

DB2 10 supports greater timestamp precision for applications that are written in host languages, such as Java, .NET, and more.

For the TIMESTAMP data type, the number of digits of fractional seconds has been extended to support a range from 0 to 12. The default is 6 digits. This

enhancement enables TIMESTAMP data with varying timestamp precision to be stored or loaded into DB2 tables, and allows manipulation of that data.

In addition, the CURRENT TIMESTAMP special register provides new syntax so that you can specify the precision of the timestamp to return.

Related concepts:

Timestamp arithmetic (DB2 SQL)

Related reference:

TIMESTAMP (DB2 SQL)

CURRENT TIMESTAMP (DB2 SQL)

Support for TIMESTAMP WITH TIME ZONE

DB2 10 introduces a new SQL data type, TIMESTAMP WITH TIME ZONE, which further extends the TIMESTAMP data type to include time zone information.

For the TIMESTAMP WITH TIME ZONE data type, the default precision for fractional seconds is 6. The time zone is the time difference in hours and minutes between the local time and Coordinated Universal Time (UTC), formerly known as Greenwich Mean Time (GMT).

Related reference:

TIMESTAMP (DB2 SQL)

Support for moving sums and moving averages

DB2 10 delivers online analytical processing (OLAP) specifications for moving sums and moving averages.

These new OLAP specifications for moving aggregates (along with the OLAP specifications for RANK, DENSE_RANK, and ROW_NUMBER, which were introduced in DB2 9) are unlike traditional scalar and aggregating functions that are supported by DB2. Scalar functions compute a single value for the current row, based on zero or more input arguments (for example, the LOCATE or VARCHAR functions). Aggregating functions collapse a group of rows into a single row, and compute a single value for the group (for example, the SUM, AVG, and STDDEV functions). These new moving aggregate functions, along with ranking and numbering specifications, compute a single value for the current row based on some or all of the rows in a defined group.

These new OLAP specifications support cumulative sums and moving averages by using a window. In the window, you can specify partitioning, the ordering of rows within partitions, and an aggregation group. The aggregation group specifies which rows of a partition, relative to the current row, are included in the calculation.

You can include OLAP specifications in expressions, in a select-list, or in the ORDER BY clause of a select-statement. You cannot use the OLAP specifications for moving aggregates within an argument to an XMLQUERY function or an XMLEXISTS predicate, or as an argument of an aggregate function.

Related reference:

OLAP specification (DB2 SQL)

Chapter 9. Migration

As with each release of DB2 for z/OS, DB2 10 continues to provide substantial improvements that make migration easier.

Migration path from DB2 Version 8

For this release, you can migrate a DB2 Version 8 subsystem in new-function mode to DB2 10 without migrating to DB2 9.

You also can fall back to DB2 Version 8 from DB2 10. In addition, DB2 Version 8 or DB2 9 can coexist with DB2 10 in a data sharing environment and in a distributed environment.

Supporting migration from either DB2 Version 8 or DB2 9 provides you with greater flexibility as you plan for DB2 for z/OS.

Related concepts:

Introduction to migration from Version 8 (DB2 Installation and Migration)

Introduction to migration from DB2 9 (DB2 Installation and Migration)

Improvements to DB2 installation and samples

DB2 10 continues to provide improvements to installing and migrating DB2 subsystems and data sharing groups. This release of DB2 for z/OS provides enhancements to the DB2 installation CLIST, ISPF panels, and jobs, and provides new installation verification procedures (IVPs).

The DB2 installation CLIST, ISPF panels, and jobs are enhanced to minimize the overhead of installing DB2 10, or migrating from DB2 Version 8 or DB2 9 to DB2 10 by externalizing new system parameters and other settings for DB2 10.

The new IVPs that are delivered in this release build on existing IVPs, provide a reasonable health check of new functions in DB2 10, and demonstrate techniques for using these new functions.

Related tasks:

Tailoring DB2 jobs to your environment using the installation CLIST (DB2 Installation and Migration)

Verifying that installation or migration was successful with the sample applications (DB2 Installation and Migration)

Simplified installation and configuration of DB2-supplied routines

DB2 10 introduces improvements that simplify the installation, configuration, validation, and service of DB2-supplied stored procedures and user-defined functions. These DB2-supplied routines are delivered with the DB2 for z/OS base FMID.

In recent releases, the number of DB2-supplied routines (referring to both DB2-supplied stored procedures and user-defined functions) has grown dramatically. Installing, configuring, validating, and servicing the various objects

that support DB2-supplied routines is largely a user-managed effort, requiring a multitude of tasks and installation jobs. The process can be laborious.

Now, this release of DB2 for z/OS provides substantial improvements that relieve the manual effort of these processes. These improvements include:

- A new program for installing and servicing DB2-supplied routines according to user-specified configuration parameters
- A new program for validating that a DB2-supplied routine is correctly installed and configured
- New jobs for calling these programs
- A description of a core set of WLM environments, collectively suitable for running all DB2-supplied routines
- DB2-supplied address space procedures for these environments
- New installation panels for specifying routine configuration parameters
- Premigration queries and reports for identifying the current WLM environment, execute access list, and package owner (if any) for each routine

The process for establishing the security environment for DB2-supplied routines that require one is unchanged from previous releases of DB2 for z/OS.

Related tasks:

Installing DB2-supplied routines during installation (DB2 Installation and Migration)

Installing DB2-supplied routines during migration (DB2 Installation and Migration)

Related reference:

Installation information for procedures and functions that are supplied with DB2 (DB2 Installation and Migration)

DB2 catalog restructured

DB2 10 introduces substantial changes to the DB2 catalog. The DB2 catalog is restructured to reduce lock contention.

Also, all links in the catalog and directory are removed during ENFM processing. Some utilities that operated on links are no longer needed in DB2 10 new-function mode. Whereas, other operations that you could not do on catalog tables that contained links are now possible in DB2 10 new-function mode.

Additional changes to the DB2 catalog include:

- **Catalog and directory tables are moved to partition-by-growth table spaces:** Each table in the following table spaces is moved to its own partition-by-growth table space during ENFM processing:
  - DSNDB06.SYSOBJ
  - DSNDB06.SYSDBASE
  - DSNDB06.SYSVIEWS
  - DSNDB06.SYSPLAN
  - DSNDB06.SYSPKAGE
  - DSNDB06.SYSDBAUT
  - DSNDB06.SYSGROUP
  - DSNDB06.SYSALTER

- DSNDB06.SYSSEQ2
- DSNDB01.DBD01
- DSNDB01.SPT01

Each table space uses row-level locking, and all tables are in reordered-row format.

- **Some user-defined tables are now part of the catalog:** In previous versions, SYSIBM.SYSDUMMYA SYSIBM.SYSDUMMYE, and SYSIBM.SYSDUMMYU tables are user-defined and created as part of the DSNTIJSG job. After you run the DSNTIJTC job during migration to conversion mode, these tables are part of the DSNDB06 catalog database.
- **Some statements are stored in new LOB columns:** After migration to new-function mode, SQL statements that were previously stored in text columns of SYSVIEWS.TEXT, SYSTRIGGERS.TEXT, and SYSPACKSTMT.STMT are now stored in new LOB columns.

Related concepts:

Implications of migrating to DB2 10 new-function mode (DB2 Installation and Migration)

SYSDUMMYx tables (Introduction to DB2 for z/OS)

Related tasks:

Migration step 18: Tailor DB2 10 catalog: DSNTIJTC (DB2 Installation and Migration)

Related reference:

Catalog changes in DB2 10

Chapter 10. Leveraging your enterprise for information on demand

DB2 for z/OS delivers functions and support that increase compatibility within the DB2 family and let you more easily leverage your enterprise for information on demand.

Increases in family compatibility translate to increases in productivity for programmers who work in a standard application environment and increases in the portability of your applications. Enhancements that address these areas include pureXML support in the DB2 database and compatibility and family leadership through various new and improved SQL functions and statements.

In addition to enhancing programmer productivity and application portability, improvements in QMF™, and an array of DB2 Tools and new tools offerings, such as IBM Data Studio or IBM Data Server Manager and the DB2 Accessories Suite, let you access your data and manage your enterprise like never before.

Most of the new and improved functions deliver benefits for both business partners and customers.

Seamless integration of XML data and relational data

XML is an important innovation that enables business-to-business communication of data, regardless of differences in the systems that receive and work with the data.

With XML, you can easily send semi-structured data across the Web without losing the relationship between data within a document. Because the content of an XML document is independent from the formatting instructions, data can be rendered appropriately in a variety of output formats. Rich support of XML within a database management system provides advantages that include more efficient storage, query, and indexing capabilities.

Support for XML capabilities and functions span the entire DB2 family. DB2 Version 8 of DB2 for z/OS and Version 8 of DB2 for Linux, UNIX and Windows provide basic support for storing, retrieving, and querying XML documents. DB2 9 for Linux, UNIX and Windows continues the work by delivering rich support of XML. Now, this release of DB2 for z/OS expands on similar support by delivering seamless integration of XML data and relational data in the DB2 database. Functional changes are provided in the following areas:

- XML document storage
- XML document retrieval
- Application development
- Database administration support
- Performance benefits through indexing support

XML document storage

This release of DB2 for z/OS provides fully integrated storage of XML data in the DB2 database system, which lets your client applications access and manage the XML data by leveraging DB2 functionality. The XML column data type is provided

for storing XML data in DB2 tables. Most SQL statements support the XML data type. As a result, you can perform many common database operations with XML data, such as creating tables with XML columns, adding XML columns to existing tables, creating indexes over XML columns, creating triggers on tables with XML columns, and inserting, updating, or deleting XML documents. A decomposition stored procedure is also provided. With this stored procedure, you can extract data items from an XML document and store those data items in columns of relational tables.

XML document retrieval

You can use SQL to retrieve entire documents from XML columns in a way that is similar to retrieving data from any other type of column. When you need to retrieve portions of documents, you can specify XPath expressions, through SQL with XML extensions (SQL/XML).

Application development

Application development support of XML in DB2 for z/OS enables applications to combine XML, relational data access, and storage. The following programming languages support the XML data type:

- Assembler
- C or C++ (embedded SQL or DB2 CLI)
- COBOL
- Java (JDBC or SQLJ)
- PL/I
- pureXML

Database administration support

Database administration support includes:

- An XML schema repository for all XML schemas that are required to validate and process XML documents that are stored in XML columns or decomposed into relational tables
- The ability to use DB2 for z/OS utilities to perform operations on XML objects in a way that is similar to handling LOB objects

Performance benefits through indexing support

This release of DB2 for z/OS offers performance benefits through indexing support, which is available for data that is stored in XML columns. Using indexes over XML data can improve the efficiency of queries that you issue against XML documents.

Tools that support your enterprise

IBM offers many tools that help you to perform administration tasks, to access and integrate your information, and to effectively manage utilities, performance, recovery, and applications.

Related information:

DB2 Tools for z/OS and DB2 10 for z/OS Compatibility

Accessing your enterprise data on demand with QMF

DB2 Query Management Facility™ (QMF) offers a cross-platform workstation and web-based solution that provides on demand access to data, reports, and interactive visual solutions through a rich desktop application or an ordinary web browser.

The QMF Version 8, Version 9, and Version 10 family of products works with DB2 10 for z/OS. For information about maintenance that you might need to apply to ensure the proper operation of QMF Version 8 or Version 9 with DB2 10, see DB2 Information Management Tools and DB2 for z/OS Compatibility.

The DB2 Query Management Facility (QMF) family of products includes:

- QMF for TSO/CICS
- QMF High Performance Option (QMF HPO)
- QMF Analytics for TSO
- QMF for Workstation
- QMF for WebSphere®

Related reference:

DB2 Query Management Facility (QMF) information

Managing your enterprise with DB2 Tools for z/OS

DB2 Tools for z/OS help reduce manual tasks, maintain high availability, and perform information replication and integration functions.

Together, the array of tools that IBM offers can help you meet performance standards and control IT costs. You can combine a selection of tools from any category to build a complete, flexible, and affordable solution that is tailored to meet your specific needs. A wide variety of tools are available and ready to support all major new DB2 for z/OS functions.

Database administration

Database administration tools meet common database service and support requirements to streamline the management of DB2. DB2 Database Administration tools include:

- IBM Data Server Manager
- IBM Data Studio
- IBM DB2 Administration Tool for z/OS
- IBM DB2 Administration Toolkit for z/OS the SAP Edition
- IBM DB2 Audit Management Expert
- IBM DB2 Object Comparison Tool for z/OS
- IBM DB2 Storage Management Utility for z/OS
- IBM Data Encryption for IMS and DB2 Databases for z/OS

Utilities management

Utilities management tools provide a full set of utilities that handle unload, load, and many other tasks. Numerous enhanced utilities help improve productivity, performance, and availability. Utilities management tools include:

- IBM DB2 Automation Tool for z/OS
- IBM DB2 Automation Toolkit for z/OS the SAP Edition
- IBM DB2 Cloning Tool for z/OS
- IBM DB2 High Performance Unload for z/OS

- IBM DB2 Utilities Enhancement Tool for z/OS
- IBM DB2 Utilities Suite for z/OS

  Important: In DB2 10 for z/OS, DB2 Utilities Suite for z/OS is available as an optional product. You must separately order and purchase a license to such utilities, and discussion of those utility functions in this publication is not intended to otherwise imply that you have a license to them. For more information, see DB2 utilities packaging (DB2 Utilities).

Performance management

Performance management tools help you keep DB2 running at peak levels even under heavy demand. Performance management tools include:
- IBM Data Server Manager
- IBM DB2 Buffer Pool Analyzer for z/OS
- IBM DB2 Performance Toolkit for DB2 on z/OS the SAP Edition
- IBM DB2 Query Monitor for z/OS
- DB2 Query Workload Tuner for z/OS
- IBM DB2 SQL Performance Analyzer for z/OS
- IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS
- IBM Tivoli OMEGAMON XE for DB2 Performance Monitor for z/OS

Recovery management

Recovery management tools manage a variety of image copy and backup and recovery tasks, including change accumulation, that improve the flexibility of DB2 and increase the value and availability of copied data for DB2. Recovery management tools include:
- IBM DB2 Archive Log Accelerator
- IBM DB2 Change Accumulation Tool for z/OS
- IBM DB2 Log Analysis Tool for z/OS
- IBM DB2 Object Restore for z/OS
- IBM DB2 Recovery Expert for z/OS
- IBM Application Recovery tool for IMS and DB2 Databases

Application management

Application management tools help manage DB2 applications and offer cost-effective global access to information and Web-enabled transactions. Application management tools include:
- IBM DB2 Bind Manager for z/OS
- IBM Db2 Data Archive Expert for z/OS
- IBM DB2 Path Checker for z/OS
- IBM DB2 Table Editor for z/OS
- IBM DB2 Test Database Generator for z/OS

Business analysis

Business analysis tools help business analysts leverage new business intelligence from their database data. Business analysis tools include:
- IBM DataQuant for z/OS
- DB2 Query Management Facility (QMF)
- IBM DB2 Web Query Tool for z/OS

Information integration

Information integration tools offer enhanced data replication and data event publishing capabilities as part of the full Information Integration Solutions portfolio. Information integration tools include:

- IBM WebSphere Classic Data Event Publisher for z/OS
- IBM WebSphere Classic Federation Server for z/OS
- IBM WebSphere Classic Replication Server for z/OS
- IBM WebSphere Data Event Publisher for z/OS
- IBM WebSphere Data Integration Classic Connector for z/OS
- IBM WebSphere DataStage® for z/OS
- IBM WebSphere QualityStage® for z/OS
- IBM WebSphere Replication Server for z/OS

Related reference:

IBM IBM Zsoftware

Related information:

DB2 Tools for z/OS and DB2 10 for z/OS Compatibility

Query optimization with IBM Data Studio

IBM Data Studio is available at no additional cost to help you optimize queries.

IBM Data Studio provides a complete development and testing environment for building database objects and queries, and for completing data object management tasks, such as:

- Developing database applications faster with the integrated query editor for SQL and XQuery
- Building and testing your stored procedures (Java and SQL) with the interactive routine debugger
- Providing heterogeneous data server support for IBM DB2 Data Server and Informix® Dynamic Server operating systems

IBM Data Studio supports extensive query tuning features for DB2 for z/OS, and replaces Optimization Service Center for DB2 for z/OS. With IBM Data Studio you can do the following tasks:

- Capture queries from all data sources that Optimization Service Center supports and from XML files that are exported from DB2 Query Monitor for z/OS
- View formatted queries
- View access plan graphs
- Capture information about the data server that queries run against, a feature that corresponds to Service SQL in Optimization Service Center
- Generate reports on the performance of queries
- Run a query statistics advisor to analyze the statistics that are available for the data that a query accesses, check for inaccurate, outdated, or conflicting statistics, and look for additional statistics that you might capture to improve how the data server processes the query

Related reference:

IBM Data Studio product overview (IBM Data Studio)

DB2 Query Monitor for z/OS

Chapter 11. Additional value for customers migrating from DB2 Version 8

You can migrate a DB2 Version 8 subsystem in new-function mode to DB2 10 without starting the subsystem in DB2 9.

When you migrate from DB2 Version 8 to DB2 10, you receive additional value from the new functions that were introduced in DB2 9. The following information provides the an overview of these new functions. This information is categorized according to user benefits, such as availability, performance, and regulatory compliance.

Key innovations introduced in DB2 9

If you are migrating to DB2 10 from DB2 Version 8, you can also benefit from exciting innovations that DB2 9 introduced in key areas to help you realize the business value of your company's information.

Read on for a brief overview of the innovations that are delivered in this release of DB2 for z/OS.

Decreased total cost of ownership

The total cost of ownership for maintaining an information management infrastructure is determined by many factors, including the costs of hardware and software and the number of staff required.

With this release of DB2 for z/OS, reductions in your total cost of ownership can be realized through savings in hardware and software, and productivity for people. The productivity comes through improved SQL, the addition of XML, and the reduction in database administration tasks. IBM Z or System z10 offerings such as zIIP and new Business Class and Enterprise Class Processors are economical and provide extra capacity and reductions in DB2 costs for some distributed, parallel query, and utility index workloads. To ease the challenging tasks that face database administrators, some tasks have been automated or eliminated. As a result, a database administrator can manage more terabytes of data and take on more business tasks.

More power to leverage enterprise for information on demand

This release of DB2 for z/OS delivers a number of capabilities that help you leverage your enterprise for information on demand through increased programmer productivity, increased compatibility in the DB2 family of products, and improvements for managing enterprise data.

Native support for pureXML increases productivity of programmers and provides unprecedented scalability and performance. With this release of DB2 for z/OS at the core of a service-oriented architecture, you can have seamless integration of XML and relational data. Enhancements to SQL increase family compatibility and leadership. Improvements and additions in the DB2 Query Management Facility (QMF) family of products improve access to your enterprise data. New and enhanced DB2 Tools improve management of your enterprise data. Most of these

improvements also provide benefits for key enterprise application partners (for example SAP, PeopleSoft, and Siebel) and their customers.

Increased availability through database definition on demand

DB2 Version 8 delivered online schema change capabilities that increased database availability with less disruption.

This release of DB2 for z/OS expands those capabilities to deliver data definition on demand. For example, improvements in online reorganization of table spaces for a few partitions is improved greatly. One of the important changes is the ability to replace one table with another table quickly. Another important change is the ability to rename a column or an index. Other capabilities let you modify, alter, create, and rebuild objects and code either online or by other means that are less disruptive to the availability of your data.

Better performance

Complex applications include both transactions and reporting. Performing both transactions and reporting well is imperative to running an enterprise efficiently. This release of DB2 for z/OS delivers a number of enhancements that boost transaction performance and help reduce your total cost of ownership through reduced CPU time, improved query performance, faster disk access, and improved logging and insert performance.

Reductions in CPU time are observed for LOAD and REORG utilities and varying-length data. In addition, queries that are written in the native SQL procedural language are eligible to run on zIIP. Also, several improvements in disk access can reduce the time for sequential disk access.

Key improvements for reporting include optimization enhancements that improve query and reporting performance and ease of use. Improved query performance SQL improvements include more queries, more consistency with the database management industry and within the DB2 family (which contributes to ease of use and the ability to port applications to DB2 for z/OS).

Regulatory compliance

Regulatory compliance, security and auditing are in the headlines and growing much more important. This release of DB2 for z/OS helps you respond to increasing needs to protect security, to assure integrity, and to comply with regulations such as the Sarbanes-Oxley Act, the Health Insurance Portability and Accountability Act (HIPAA), and Payment Card Industry (PCI) Security Standards. Improved access control with network trusted context and roles allows more precise control of security. Improved filtering makes auditing more usable. Secure Socket Layer (SSL) data encryption on networks is more secure.

Increased synergy with IBM Z

This release of DB2 for z/OS takes advantage of the latest improvements in IBM Z and System z10 hardware and software to provide better performance, improved value, more resilience, and better function.

For example, remote native SQL procedures are now enabled for processing on IBM Z Integrated Information Processor (zIIP) specialty engines. DB2 synergy with System z9 and System z10 continues with a range of I/O improvements in

channels, in disks, and in DB2. Tests with DB2 Version 8 that involve parallel access on one channel have shown faster response times for sequential access; similar improvements are anticipated for this release of DB2 for z/OS. For example, one early performance test for sequential performance achieved 183 MB per second. This release of DB2 for z/OS also takes advantage of new System z9 and System z10 hardware support with a new decimal floating-point data type that lets you use decimal floating-point numbers with greater precision. FlashCopy can be used for DB2 database backup and restore operations. Other improvements include added security and encryption and Unicode collation.

Availability enhancements introduced by DB2 9

If you are migrating to DB2 10 from DB2 Version 8, enhancements that were introduced in DB2 9 offer improved availability through a new DB2 command and extensions to existing SQL statements and online utilities.

Online REORG with no BUILD2 phase

In this release of DB2 for z/OS, the BUILD2 phase has been eliminated to improve availability of your data.

When you reorganize a table space with SHRLEVEL CHANGE in previous versions of DB2, the data that is undergoing reorganization is not available to applications during the relatively lengthy SWITCH and BUILD2 phases.

Faster replacement of one table with another

With this release of DB2 for z/OS, you can replace tables faster by using the new support for clone tables.

With clone table support, you can generate a copy of a current table, in the same table space, that has the same attributes, structure, and data as the original table. After you create a clone table, you can insert or load data into the clone table and exchange the clone table name with the current table name.

Extensions to the ALTER TABLE statement support the clone table function and let you:

- Create clone tables with the ADD CLONE option. Clone tables can be partitioned or non-partitioned, and you can make image copies of a clone table.
- Drop clone tables with the DROP CLONE option.
- Exchange current data with clone data with the EXCHANGE option, which provides a fast replacement of the original data, and is an alternative to the online LOAD REPLACE capability.

Web-based applications that need maximum availability can benefit from the option to implement a pair of tables that have cloned structures. Copies for application testing and auditing can be easily created. Clone table support also provides the unique ability to quickly change table names. As a result, applications can quickly and almost transparently switch between dual mirror tables. When the data in the clone table needs to become active to an application, you can use the EXCHANGE option of the ALTER TABLE statement to switch the table name, which provides fast replacement of the original data.

Universal table spaces

DB2 9 introduces a new type of table space: a *universal table space*. A universal table space is a table space that is both segmented and partitioned.

Two types of universal table spaces are available: the partition-by-growth table space and the range-partitioned table space.

A universal table space offers the following benefits:

- Better space management relative to varying-length rows: A segmented space map page provides more information about free space than a regular partitioned space map page.
- Improved mass delete performance: Mass delete in a segmented table space organization tends to be faster than table spaces that are organized differently. In addition, you can immediately reuse all or most of the segments of a table.

Partition-by-growth table space

Before DB2 9, partitioned tables required key ranges to determine the target partition for row placement. Partitioned tables provide more granular locking and parallel operations by spreading the data over more data sets. Now, you have the option to partition according to data growth, which enables segmented tables to be partitioned as they grow, without the need for key ranges. As a result, segmented tables benefit from increased table space limits and SQL and utility parallelism that were formerly available only to partitioned tables, and you can avoid needing to reorganize a table space to change the limit keys.

You can implement partition-by-growth table space organization in several ways:

- You can use the new MAXPARTITIONS clause on the CREATE TABLESPACE statement to specify the maximum number of partitions that the partition-by-growth table space can accommodate. The value that you specify in the MAXPARTITIONS clause is used to protect against run-away applications that perform an insert in an infinite loop.
- You can use the MAXPARTITIONS clause on the ALTER TABLESPACE statement to alter the maximum number of partitions to which an existing partition-by-growth table space can grow. This ALTER TABLESPACE operation acts as an immediate ALTER.

Range-partitioned table space

A *range-partitioned table space* is a type of universal table space that is based on partitioning ranges and that contains a single table.

The new range-partitioned table space does not replace the existing partitioned table space, and operations that are supported on a regular partitioned or segmented table space are supported on a range-partitioned table space. You can create a range-partitioned table space by specifying both SEGSIZE and NUMPARTS keywords on the CREATE TABLESPACE statement.

With a range-partitioned table space, you can also control the partition size, choose from a wide array of indexing options, and take advantage of partition-level operations and parallelism capabilities. Because the range-partitioned table space is also a segmented table space, you can run table scans at the segment level. As a result, you can immediately reuse all or most of the segments of a table after the table is dropped or a mass delete has been performed.

Better availability during REBUILD INDEX operations

The online REBUILD INDEX utility has been extended to support read and write access for a longer period of time during the utility operation.

As a result, applications have greater access to data while indexes on that data are being rebuilt. This complements the DB2 Version 8 support in which insert, update, and delete operations are supported on indexes that are non-unique while the index is in the process of rebuilding.

Improved availability with column and index renaming capabilities

Two new capabilities allow you to rename a column without dropping and rebuilding a table, and to rename an index through a catalog operation.

The ALTER TABLE statement has been expanded to include the RENAME COLUMN clause. Now, you can rename a column without needing to drop and rebuild a table, which is less time consuming, reduces processing costs, and provides a higher degree of availability. The ability to rename a column in this manner also allows for better reuse of columns because you do not need to add additional columns.

The RENAME statement has been expanded to include the INDEX keyword, which lets you rename an index through a catalog operation. The ability to rename indexes in this manner also lets you consolidate index names that are used in different releases of SAP.

Modify EARLY code without an IPL

With the new DB2 command, **REFRESH DB2,EARLY**, you can refresh EARLY code when DB2 is not active.

No IPL is required to complete the refresh operation. When you enter the command, the EARLY code modules are reloaded, and the EARLY control block is rebuilt. Previous copies of the EARLY modules are deleted the next time that you start DB2.

ALTER TABLESPACE and index logging improvements

In this release of DB2 for z/OS, the ability to alter a table space to suppress logging is extended. Now, you can suppress logging in base table spaces, XML table spaces, and the indexes that are associated with them.

Indexes inherit logging attributes from the base table with which they are associated. Suppression of logging is advantageous in a number of situations in which data is being duplicated. In those cases, if the data is lost, you can regenerate it from the original source rather than from an image copy and subsequent application of log records.

Related concepts:

Expanded support for not logging table spaces

Support for using SMS storage classes with DB2-defined data sets

With this release of DB2 for z/OS, you can use SMS storage classes with DB2-defined data sets by using the DB2 STOGROUP statement.

The CREATE STOGROUP and ALTER STOGROUP statements have been enhanced to include SMS data class, management class, and storage class as optional

parameters. Enabling DB2-defined data sets to use SMS storage classes increases the flexibility of handling these data sets while minimizing the time-consuming manual effort that is involved.

DB2 support for extended address volumes (EAV)

z/OS Version 1 Release 11 introduces extended address volumes (EAV) support for extended format sequential data sets and VSAM data sets.

With EAV support, the maximum number of cylinders on a volume is 262,668. This means that you can store more DB2 data on a single volume. However, the maximum amount of data that you can store in a single data set of a DB2 table space or index space is the same for EAV and non-extended address volumes, which is 64 GB.

Performance enhancements introduced by DB2 9

If you are migrating to DB2 10 from DB2 Version 8, enhancements that were introduced in DB2 9 provide improved performance through reductions in CPU processing time for a number of utilities, and faster disk access. Other changes include improved logging and insert operations, improved query optimization, and index improvements.

Reduction in CPU processing time for utilities

In this release of DB2 for z/OS, CPU processing time is reduced when you extract non-padded index keys from data rows by using certain utilities.

The following utilities extract non-padded index keys from data rows:
- COPY
- REORG
- LOAD
- REBUILD INDEX
- RECOVER
- RUNSTATS
- CHECK INDEX

SQL optimization improvements

This release of DB2 for z/OS offers a number of SQL optimization enhancements, including support for histogram statistics and improved optimization techniques.

Better data for DB2 optimization with histogram statistics

This release of DB2 for z/OS supports histogram statistics, which provide better data for the DB2 optimization.

Data distribution statistics are important for query optimization. DB2 chooses the best access path based on cost. The basic foundation of cost is predicate selectivity estimation, which relies heavily on data distribution statistics. Prior versions of DB2 for z/OS rely on frequency statistics that are collected (through the RUNSTATS utility) on single values, either from a single column or from multiple columns. Now, with histogram statistics, DB2 can improve access path selection by estimating predicate selectivity from value-distribution statistics that are collected over the entire range of values in a data set, unlike frequency statistics.

Improved optimization techniques

DB2 9 introduces improved several techniques for optimizing query performance.

Global query optimization

In prior versions of DB2 for z/OS, query performance problems can result when DB2 breaks a query into multiple parts (for example, a correlated subquery and an outer query) and optimizes each of those parts independently. While each of the individual parts might be optimized to run efficiently, the overall result can be inefficient when these parts are combined. DB2 9 supports global query optimization, which enables DB2 to optimize a query as a whole rather than as an independent part. When a query is optimized as a whole entity, DB2 can consider the effect of one query block on another, and can consider reordering query blocks to determine the most efficient query path.

Sparse index and in-memory data caching enhancements

In DB2 Version 8, materialized work files for star join queries can be stored in the global storage pool above the 2-GB bar (an in-memory data cache) as more cost-effective alternative to the sparse index technique that is used in prior versions. A sparse index is used as a fallback at run time if enough memory is not available in the global storage pool.

In DB2 9, the use of in-memory data caching is generalized to support more types of queries. In addition, data is cached in a local storage pool above the 2-GB bar, which can reduce potential storage contention because data caching storage management is associated with each thread. You can allocate up to 20 MB of memory for data caching. If the required memory exceeds the specified limit, a sparse index is used instead. This expanded access method provides potential benefits for tables that lack an appropriate index or enough statistics, and competes with other options for the most efficient access path.

Complex query optimization through page-range screening enhancements

The number of partitions that DB2 accesses to evaluate a query predicate can affect the performance of that query. A query that provides data retrieval through a data-partitioned secondary index (DPSI) might access some or all partitions of the DPSI. For a query that is based only on a DPSI key value or range, DB2 must examine all partitions. However, if the query also has predicates on the leading columns of the partitioning key, DB2 does not need to examine all of the partitions. Page-range screening, or limited partition scan, is a DB2 feature that removes inapplicable partitions from consideration for access.

Now, DB2 9 takes advantage of page-range screening by writing queries that create a non-matching predicate on the partitioning keys when literal values are used.

Autonomic reoptimization for dynamic queries

The access path for static and dynamic queries that contain host variables is chosen by the optimizer during bind time, before the values of the host variables are available. Because the values of the host variables are not considered, the access path that is chosen might not always be optimal. In versions prior to DB2 Version 8, this problem is partially solved by the REOPT(ALWAYS) bind option, which prepares the statement again at run time when the host input variables are available. As a result, the optimizer can reoptimize the access path based on known input variable values. However, frequently called SQL statements that take little time to execute are not good candidates for this method.

DB2 Version 8 introduced the bind option, REOPT(ONCE), which reoptimizes the access path of a dynamic query only once at run time, no matter how many times the same statement is executed. With this technique, DB2 chooses the access path based on the set of input variable values, and the access path is stored in the dynamic statement cache for use by subsequent executions of the query. While this option offers additional flexibility, the impact of host variables on the selection of optimal access paths continues to be observed.

DB2 9 offers greater flexibility with the REOPT(AUTO) option, which enables DB2 to autonomically determine if a new access path is needed for a statement in the dynamic statement cache. When you specify REOPT(AUTO), DB2 optimizes the access path for dynamic SQL statements at the first execute or open operation. Each time a statement is executed, DB2 determines whether a new access path is needed to improve the performance of the statement. If a new access path will improve the performance, DB2 generates a new access path and replaces the previous access path in the dynamic statement cache.

FETCH FIRST *n* ROWS ONLY and ORDER BY in subselects and fullselects

In prior versions, the ORDER BY and FETCH FIRST *n* ROWS ONLY clauses were supported only at the statement level as part of select-statement or a SELECT INTO statement. DB2 9 introduces additional flexibility by allowing both FETCH FIRST *n* ROWS ONLY and ORDER BY clauses when they are specified as part of a subselect or a fullselect statement. With the new support for FETCH FIRST *n* ROWS ONLY, you can select the top *n* rows in the result table of a table expression, a leg of a union, or a subquery (for a fullselect that is a component of a predicate). If you specify the FETCH FIRST *n* ROWS ONLY clause with the ORDER BY clause, the ordering is performed on the entire result table before the first *n* rows are returned.

Indexing improvements

Indexing improvements contribute to the overall improvements in query performance.

Specific improvements include index compression, index on expression, index key randomization, and larger index page sizes.

Index compression

This release of DB2 for z/OS provides the ability to compress an index without using a dictionary. Compressing an index reduces the physical storage space that an index requires. By eliminating the need for a dictionary, index data can be compressed as soon as the first index entries are added to an index, and the space that would normally be used by a dictionary becomes available. You can choose whether you want to use index compression by specifying COMPRESS YES or COMPRESS NO on the CREATE INDEX or the ALTER INDEX statements.

Expression-based index

Support for a new index type, expression-based index, lets you create an index on a general expression. Query performance can be enhanced if the optimizer chooses that index. When you use an expression-based index, the results of the expressions are evaluated during insertion time or during an index rebuild and are kept in the index. If the optimizer chooses to use that index, the predicate is evaluated against

the values that are stored in the index. As a result, run time performance overhead is eliminated.

Larger index page sizes

Prior versions of DB2 for z/OS limit the size of an index page to 4 KB. The size of an index page limits the number of index keys that the index page can accommodate and can cause contention in indexes that split frequently. DB2 9 lifts these restrictions by offering expanded index page sizes of 8 KB, 16 KB, and 32 KB. An index page size that is greater than 4 KB accommodates more index keys per page and can reduce the frequency of index page splits. You can use the INDEXBP option on both CREATE DATABASE and ALTER DATABASE statements to specify 4-KB, 8-KB, 16-KB, or 32-KB index buffer pools. You can also use the BUFFERPOOL keyword on the CREATE INDEX statement to specify 8-KB, 16-KB, and 32-KB buffer pools.

Index key randomization

Currently, online transaction processing (OLTP) workloads in a data sharing environment can experience locking contention on the last page of an index, especially when an application program uses indexes on columns that contain current timestamps or ever-increasing sequential values create insertion hot spots. As a result of this contention, an application must wait to acquire an index page. To offer relief from locking contention, DB2 9 provides support to randomize index key columns through the new RANDOM option on both CREATE INDEX and ALTER INDEX statements.

When you specify the RANDOM option, values are stored at random places in the index tree. As a result, the number of consecutive insertions on a page decreases and contention is also decreased. You can use a randomly ordered index to perform equality lookups on a specified column. In addition, key columns that are in random order can be used in non-matching index scans, and index-only access on random key columns is possible. Even though values are stored in random order, you can retrieve the original value of the random key column.

Related concepts:

Capability to create an expression-based index

- Expression-based indexes (Introduction to DB2 for z/OS)
- Expressions (DB2 SQL)

Relief for sequential key insert

- Types of indexes (Introduction to DB2 for z/OS)

Improved performance for varying-length rows

In DB2 9, the format in which a row that contains varying-length columns is stored in the table has changed. This new format facilitates locating columns within the row for data retrieval and predicate evaluation.

As a result, you no longer need to run a sequential scan, and the performance improves for access to data in tables that store rows with varying-length columns.

In previous versions of DB2 for z/OS, if you store a value that is shorter than the length of a column in a varying-length column, the data is not padded to the full length of the column. As a result, columns that follow varying-length columns are

at variable offsets in the row. When you need to locate and access such a column, you must scan the columns sequentially after the first varying-length column.

Relief for sequential key insert

This release of DB2 for z/OS includes support for asymmetric splitting of index pages and increased index page sizes (larger than 4 KB).

In prior releases, an index page splits so that approximately half of the index keys on the splitting page remain on one page, and the remaining index keys move to a new page. This 50:50 split can cause frequent page splits on an index that has sequential insert patterns. As a result, half of the splitting pages are empty. Non-partitioned indexes that are updated by LOAD utility jobs that run in parallel against multiple partitions are especially susceptible to these problems because sequential inserts into multiple ranges in the non-partitioned index can occur.

Allowing index pages to split asymmetrically can improve space utilization and reduce contention that results from frequent page splits in an index with sequential insert patterns in the middle of the index. An index page size that is greater than 4 KB can also relieve contention by accommodating more index keys per page, which reduces the frequency of page splits in indexes. You can use the INDEXBP option on both CREATE DATABASE and ALTER DATABASE statements to specify 4-KB, 8-KB, 16-KB, or 32-KB index buffer pools, and the BUFFERPOOL keyword on the CREATE INDEX statement to specify 8-KB, 16-KB, and 32-KB buffer pools.

Improved logging performance

This release of DB2 for z/OS provides several logging enhancements to improve performance.

These logging enhancements are as follows:

- In prior versions, data sharing workloads that perform a high volume of logging can experience significant delays because of the extra processing required to set unique LRSN (log record sequence number) values for each record. In DB2 9, the process for setting and updating LRSN values has been streamlined to improve performance.
- Prior versions of DB2 for z/OS use the basic direct access method (BDAM) to read archive logs from disk, which facilitates rapid access to the VSAM control intervals, but which does not support striped data sets and extended format data sets. DB2 9 replaces BDAM with the basic sequential access method (BSAM), which supports both striped and extended format data sets, and improves read and write performance.
- In prior versions of DB2 for z/OS, an active log can have up to 4 GB of tracks on a single disk volume, whereas the size limit for an archive log is only about 3 GB. As a result, installations that use 4-GB active logs are forced to put archive logs on tape. Version 1.7 of z/OS now supports large data sets and removes the 64-KB archive log limit. DB2 9 takes advantage of the new limit by introducing the DSNTYPE=LARGE attribute of the PRIMARY QUANTITY field of installation panel DSNTIPA. The new attribute supports an archive log that has up to 4 GB of tracks per disk volume.

Improved data insert performance

The APPEND option of the CREATE TABLE and ALTER TABLE statements offers increased performance for inserting data into the end of a table.

This option reduces the processing that is used to target the locations for new rows.

Like the LOAD RESUME utility, the rows are placed at the end of the table and are not clustered as they would be for a normal insert operation. The APPEND option provides a trade-off that favors inserts rather than retrievals and the need to reorganize.

Other DB2 9 functions that can improve data insert performance include several index improvements, logging performances improvements, and not logging table spaces.

Related concepts:

ALTER TABLESPACE and index logging improvements

Indexing improvements

Improved logging performance

Security and regulatory compliance enhancements that DB2 9 introduced

If you are migrating to DB2 10 from DB2 Version 8, enhancements that were introduced in DB2 9 help you respond to increasing needs to protect security, to assure integrity, and to comply with regulations.

These regulations include the Sarbanes-Oxley Act, the Health Insurance Portability and Accountability Act (HIPAA), and Payment Card Industry (PCI) Security Standards.

Improved access control with network trusted context and roles allows more precise control of security. Improved filtering makes auditing more usable. Secure Sockets Layer (SSL) data encryption on networks is more secure.

Roles and network trusted contexts

Support for trusted context helps to establish a trusted relationship between DB2 and an external entity, such as a database administrator or a middleware server.

With trusted context support, a series of trusted attributes are evaluated to determine whether a specific context can be trusted. After a trusted context is established, you can define a unique set of interactions between DB2 and the external entity, such as a middleware server, so that the existing database connection can be used by a different user without requiring authentication of the new connection user.

Support for trusted context also provides the ability, within a specific trusted context, for a DB2 authorization ID to acquire a special set of privileges that are not available outside that trusted context by defining roles. A role is a database entity that groups together one or more privileges and can be assigned to users. A role provides privileges, in addition to the current set of privileges, that are granted to the primary and secondary authorization identifiers. A role can own objects if the objects are created in a trusted context with the role defined as the owner. If a role is defined as an owner, then only the privileges that are granted to the role are considered for object ownership.

Improved auditing

Improved trace filtering makes the jobs of auditing and performance management easier.

Many more options can be used to minimize the amount of data collected, so that the overhead is reduced and the extraneous data does not need to be processed.

Support for Secure Socket Layer protocol

This release of DB2 for z/OS supports the Secure Socket Layer (SSL) protocol. The SSL protocol is implemented by the z/OS Communications Server IP Application Transparent Transport Layer Security (AT-TLS) function.

The z/OS Version 1 Release 7 Communications Server for TCP/IP introduces the AT-TLS function in the TCP/IP stack for applications that require secure TCP/IP connections. AT-TLS performs transport layer security on behalf of the application, such as DB2, by invoking the z/OS system SSL in the TCP layer of the TCP/IP stack. The z/OS system SSL provides support for TLS V1.0, SSL V3.0, and SSL V2.0 protocols.

More security options with INSTEAD OF triggers

INSTEAD OF triggers, which are defined on views, provide one way of ensuring security within the database.

INSTEAD OF triggers are used to process insert, update, and delete operations (through trigger logic) rather than relying on the INSERT, UPDATE, or DELETE statement to activate the trigger.

Related concepts:

INSTEAD OF triggers

Support for AES encryption

This release of DB2 for z/OS supports Advanced Encryption Standard (AES) encryption of user IDs and passwords for both a server and a requester.

AES encryption is supported only if the server and requester are at the DRDA security manager level 9 or later, and only for TCP/IP communications. Integrated Cryptographic Service Facility (ICSF) must be installed, configured, and active before servers can offer AES encryption and decryption services.

| See the DB2 for z/OS Program Directory for ICSF hardware and software
| requirements for AES encryption.

Related information:

DB2 for z/OS Program Directories

Compatibility and leadership with SQL

If you are migrating to DB2 10 from DB2 Version 8, enhancements that were introduced in DB2 9 delivers functional changes in SQL that increase productivity of programmers through family compatibility and leadership.

DB2 Version 8 of DB2 for z/OS took big steps toward improving consistency of SQL across the DB2 family by providing more functions that are common between DB2 for Linux, UNIX and Windows. This release of DB2 for z/OS and DB2 9 for Linux, UNIX and Windows moves even more SQL function from the unique set to

the common set. With the increase in common SQL, programmer productivity also increases. This release of DB2 for z/OS also introduces several new SQL functions that are firsts in the DB2 family.

SQL consistency improvements

Increased consistency with SQL across the DB2 family is good news for programmers who write cross-platform applications because porting and writing application programs becomes easier.

The following SQL functions, statements, and clauses are consistent across the DB2 family.

SELECT FROM UPDATE or SELECT FROM DELETE function

The INSERT-within-SELECT feature that was introduced in DB2 Version 8 of DB2 for z/OS has been expanded in DB2 9. DB2 9 supports the retrieval of columns from rows that are modified by a SELECT FROM DELETE or a SELECT FROM UPDATE statement.

Now, one SQL call to DB2 modifies the table contents and returns the resultant changes to the application program. In addition, an application can now be coded to perform a destructive read from a table when a SELECT FROM DELETE statement is included. This feature is particularly useful when a table is used as a data queue.

INSTEAD OF triggers

The introduction of INSTEAD OF triggers reduces the complexity in your application programs.

In previous versions of DB2 for z/OS, operations that required different rules for read and write access (for example, encryption and decryption) required users to build into their applications an awareness that the object for read access is a view, and that the object for write access is a base table.

INSTEAD OF triggers are defined on views only, and are used to process insert, update, and delete operations (through trigger logic) rather than relying on the INSERT, UPDATE, or DELETE statement to activate the trigger. With the INSTEAD OF trigger, an application doesn't need to include the complexity that specifies which operations are performed against views and which operations are performed against the base table because the activated trigger makes the operations seem as if they are performed against a view.

Examples of using INSTEAD OF triggers include updates through join queries or encoding and decoding data from the database within a view.

BIGINT data type and function

This release of DB2 for z/OS introduces a new SQL data type and function, BIGINT (big integer). BIGINT supports big integers and extends the set of currently supported exact numeric data types (SMALLINT and INTEGER).

A big integer is a binary integer that has a precision of 63 bits. The BIGINT data type can represent 63-bit integers and is compatible with all numeric data types. The BIGINT function returns a big integer representation of a number or a string representation of a number. You can store or load big integer values into DB2 tables, and manipulate the data in a variety of ways. In addition, you can use the CAST specification to increase the portability of your applications.

Note: Before migration to new-function mode, integer constants that are outside the range of large integers are treated as decimal constants. The range of large integers is -2147483647 to 2147483647. In DB2 10 new-function mode, integer constants that are outside the range of large integers but within the range of big integers are considered big integers. The range of big integers is -9223372036854775808 to 9223372036854775807. For more information about the implications of this change, see New BIGINT data type.

Related concepts:

Implications of migrating to DB2 10 new-function mode (DB2 Installation and Migration)

BINARY data type and function

This release of DB2 for z/OS provides expanded support for binary string data by introducing the BINARY data type and function.

The VARBINARY (varying-length binary string) data type and function are also introduced. The VARBINARY data type and function are not yet supported by other members of the DB2 family.

The BINARY data type represents a fixed-length binary string. When fixed-length binary string distinct types, columns, and variables are defined, the length attribute is specified, and all values have the same length. A binary string column is used for storing non-character data, such as encoded or compressed data, pictures, voice, and mixed media. A binary string column can also hold structured data for use by distinct types, user-defined functions, and stored procedures. The BINARY function returns a BINARY (fixed-length binary string) representation of a string of any type or a row ID type.

Related concepts:

VARBINARY data type

File reference variables

This release of DB2 for z/OS adds support for two new file reference variables, LOB and XML.

A *file reference variable* is a host variable that is defined in a host language (for example C or COBOL) to contain the file name that directs file input and output for a large object (LOB).

DB2 support for file reference variables allows a large LOB or an XML value to be inserted from a file or selected into a file rather than a host variable; the application does not need to acquire storage to contain the LOB or XML value. File reference variables also facilitate the movement of LOB or XML values from the database server to a client application or from a client application to a database server without going through the working storage of the client application. In addition, file reference variables bypass host language limitations on the maximum allowed size for LOB values that are located in working storage. The following host languages support file reference variables:

- C and C++
- COBOL
- PL/I
- Assembler
- REXX

INTERSECT keyword in a fullselect

The new INTERSECT keyword specifies the intersection set operator in a fullselect operation between two result tables.

Two types of INTERSECT operations are available: INTERSECT ALL and INTERSECT DISTINCT. If you specify INTERSECT ALL, the result consists of all rows that are in both the first result table and the second result table, and redundant duplicate rows are included. If you specify INTERSECT DISTINCT, the result consists of all rows that are in the first and second result tables, and the redundant duplicate rows are eliminated. In either case, each row of the result exists in the first and second result tables.

EXCEPT keyword in a subselect

This release of DB2 for z/OS contributes to SQL consistency by providing the new EXCEPT keyword. Use the EXCEPT keyword to specify the difference set operator in a fullselect operation between two result tables.

Two types of EXCEPT operations are available: EXCEPT ALL and EXCEPT DISTINCT. If you specify EXCEPT ALL, the result consists of all rows that are only in the first result table, and redundant duplicate rows are included. If you specify EXCEPT DISTINCT, the result consists of all rows that are only in the first result table, and the redundant rows are eliminated. In either case, each row in the result is a row from the first result table that does not have a matching row in the second result table.

Native support for SQL procedures

In this release of DB2 for z/OS, new support for native SQL procedures simplifies the definition and use of SQL procedures by eliminating the need for generating a C program.

In previous versions of DB2 for z/OS, a C program must be generated from transformed SQL statements and SQL control statements that are contained in the SQL procedure. The resulting C program is then run like an external stored procedure. Now, when you create a native SQL procedure in new-function mode, the procedural statements are converted to a representation that is stored in the database directory, as is true for other SQL statements. The parameter list and procedure options are stored in the database catalog tables as in prior releases. When a native SQL procedure is called, the representation is loaded from the directory, and the DB2 engine runs the procedure. A number of additional functions and extensions provide consistency with the SQL Standard and with the rest of the DB2 family. Examples of these improvements include support for:

- Changing name resolution within a procedure body
- Using delimited identifiers, including lowercase characters, for SQL condition names, SQL labels, SQL variables, and SQL parameters
- Nested compound statements, including a compound statement within the body of a condition handler
- Versioning and managing source code
- Deploying of native SQL procedures to multiple servers
- Debugging of native SQL procedures

Nested compound statements in native SQL procedures

Native SQL procedures support nested compound statements.

Compound statements introduce a block of SQL statements in an SQL procedure. Before DB2 9, the body of an SQL procedure could contain a single compound

statement (which could contain other SQL statements, but not another compound statement), or a single SQL procedure statement other than a compound statement. As a result, a condition handler could not contain a compound statement, either.

Now, with support for nested compound statements in native SQL procedures, you can:
- Use a compound statement within a condition handler
- Use nested compound statements to define different scopes for SQL variables, cursors, condition names, and condition handlers.

Expanded support for not logging table spaces

This release of DB2 for z/OS supports suspension of logging for base table spaces, XML table spaces, and the indexes that are associated with them.

DB2 Version 8 of DB2 for z/OS provides support for suspension of logging in global temporary tables, LOB table spaces, during the LOAD REPLACE operation, and during an insert of a declared temporary table. Now, you can specify the logging attributes, LOGGED or NOT LOGGED, at the table space level when you alter or create tables and table spaces, and when you create tables or auxiliary tables. You can also suppress logging during the online LOAD RESUME process.

Because the existing logging facilities in DB2 are already finely tuned, suspension of logging does not generally improve the performance of your system. However, the ability to suspend record logging is useful in a variety of situations in which data is being duplicated and loss of concurrency and recoverability is not a concern. In those cases, if the data is lost, you can re-create or regenerate it from the original source instead of using an image copy and applying log records. Examples of tables for which suspension of logging is advantageous include materialized query tables, summary tables, tables to which data is propagated, and temporary tables that are populated with the result set from a query as an intermediate step in an application. An additional benefit of suspended logging is improved scalability, particularly for operations that insert large volumes of data with the INSERT statement.

OLAP specifications for RANK, DENSE_RANK, and ROW_NUMBER

Online analytical processing (OLAP) specifications provide the ability to return ranking, row numbering, and existing aggregate function information. This information is returned as a scalar value in the result of a query.

You can include OLAP specifications in an expression, in a select-list, or in the ORDER BY clause of a select-statement. The result to which the OLAP specification is applied is the result table of the innermost subselect that includes the OLAP specification. This release of DB2 for z/OS delivers OLAP specifications for RANK, DENSE_RANK, and ROW_NUMBER.

RANK and DENSE_RANK

RANK and DENSE_RANK specify that the ordinal rank of a row within the specified window is computed. Rows that are not distinct with respect to the ordering within the specified window are assigned the same rank. You can define the results of ranking with gaps in the sequential rank numbering by using the RANK specification, or without gaps, by using the DENSE_RANK specification. Common examples of using RANK or DENSE_RANK specifications include:
- Ranking sales figures (for example, determining which stores in a chain have the lowest and highest sales)

- Ranking employees within a department or division according to various indicators
- Creating "top-*n*" queries (for example, retrieve the five employees that have the highest salaries)

ROW_NUMBER

ROW_NUMBER specifies that a sequential row number is computed for the row that is defined by the ordering, starting with 1 for the first row. If the ORDER BY clause is not specified in the window, the row numbers are assigned to the rows in an arbitrary order, as the rows are returned (but not according to any ORDER BY clause in the select-statement). You can use ROW_NUMBER to number the result rows of a query. Row numbers also enable easy formulation of queries for computing histogram statistics (quantile computations), and they enable formation of other OLAP specifications (for example, moving sums, moving averages, and so on).

Related reference:

OLAP specification (DB2 SQL)

COLLATION_KEY function

The new COLLATION_KEY function supports culturally correct and case-insensitive collation of Unicode data.

The COLLATION_KEY function processes a Unicode UTF-16 input string and a collation name, and returns a varying-length binary sort key. The result of the COLLATION_KEY operation can be compared to the result of another COLLATION_KEY operation on another string to determine their order within the specified collation name. The attributes of the collation name specify the collation characteristics (for example locale attribute, treatment of accent and case, an so on) for the sort key, and conform to the conventions of the z/OS-supported Unicode conversion services.

You can also use the COLLATION_KEY function to perform case-insensitive (or *caseless*) comparisons of string expressions by specifying attributes on the collation name that either ignore case or ignore a combination of case and other attributes such as spaces, punctuation, and symbols.

Capability to create an expression-based index

In this release of DB2 for z/OS, an extension to the CREATE INDEX statement lets you create an expression-based index.

An expression can be a column reference, a built-in function invocation, or a general expression with certain restrictions. Unlike a simple index, the index key of an expression-based index is composed by concatenating the result (also known as a key-target) of the expression that is specified in the ON clause. An expression-based index lets a query take advantage of index access and avoid a table space scan.

Related concepts:

Expression-based indexes (Introduction to DB2 for z/OS)

Expressions (DB2 SQL)

Index keys (Introduction to DB2 for z/OS)

Automatic creation of a database, a table space, and all system-required objects

The CREATE TABLE statement now supports automatic (implicit) creation of a database or a table space.

Implicit creation of a database, a table space, and all system-required objects: A database, a table space, and all system-required objects are created implicitly if you do not name a table space or a database in the IN clause of a CREATE statement. The attributes of the table space (for example, table space type, the underlying data sets, and use of data compression) are determined by table space installation parameters. If a table space is implicitly created, the following system-required objects are also created:

- The enforcing primary key index
- The enforcing unique key index
- The ROWID index (if the ROWID column is defined as GENERATED BY DEFAULT)
- A LOB table space, auxiliary table, and auxiliary index

Implicit creation of a table space: A segmented table space is created implicitly in DSNDB04 if you do not name a table space or a database in the IN clause of a CREATE statement.

IBM Spatial Support for DB2 for z/OS

IBM Spatial Support for DB2 for z/OS allows you to generate and analyze spatial information about geographic features and to store and manage the data on which this information is based.

IBM Spatial Support for DB2 for z/OS provides of a set of spatial data types, user defined functions, and stored procedures for spatial related queries. With Spatial Support, you can:

- Invoke spatial queries for local and remote clients to answer questions based on geographic relationships.
- Create spatial indexes on spatial columns, which can improve query performance when you use spatial predicate functions.
- Manage geographic coordinate systems, spatial indexes, and spatial column usages through stored procedure interfaces.

In addition, Spatial Support includes an ODBC program that you can use to enable and disable the spatial feature. This program can invoke each of the stored procedures through a set of command line arguments.

Leverage existing application programming skills

This release of DB2 for z/OS improves support for end-to-end application development. This support is improved by connecting new languages and environments to the scale and value of the existing infrastructure.

Increased consistency across the DB2 family lets you leverage the existing skills of your application programmers through support of a wider range of tools, environments, and languages. The IBM Rational® and WebSphere product lines provide part of the connection, with products like Rational Data Architect, Rational Application Developer, and WebSphere Information Integration. DB2 clients provide more support for new environments and new languages.

This release of DB2 for z/OS supports key database technologies, including SQL and native SQL procedures, XML, service-oriented architecture (SOA), and various web services.

A wide variety of development frameworks provide language support and application development and deployment tools for web-based applications that work with the DB2 family. These frameworks also support both traditional programming languages and newer, open source languages. For example, development communities exist for the following traditional programming languages:

- COBOL, PL/I, REXX, C, C++, Fortran, assembler, and APL2®
- Java support, including JDBC and SQLJ
- Microsoft language products, including .NET, Visual C# for .NET, and Visual Basic for .NET

Examples of Open Source development support include:

- Eclipse
- PHP and Zend Core for IBM
- Perl
- Python
- Ruby on Rails
- TOAD for DB2

Note: The IBM Data Server Driver for JDBC and SQLJ has been upgraded to Version 3.57, which is the same level that DB2 for Linux, UNIX, and Windows, and IBM Informix Dynamic Server provide.

Enhancements to large object support

This release of DB2 for z/OS offers many improvements for handling large objects (LOBs) through extensions to SQL statements and enhancements to utilities and performance.

SQL enhancements for large objects

This release of DB2 for z/OS provides enhancements to the FETCH statement for LOB and XML data. This release also includes support for LOB file reference variables.

Enhancements to the FETCH statement for LOB and XML data

In previous versions of DB2, applications that worked with LOBs had two primary methods for fetching LOB data: Fetching data into a pre-allocated buffer and using a LOB locator to retrieve a handle on the data. Fetching data into a preallocated buffer has the potential for causing virtual storage constraint problems, especially LOBs that have a large maximum length. In addition, an application that uses LOB locators that commit infrequently or do not explicitly free the locators can use considerable amounts of DB2 resources. The introduction of XML objects in DB2 9 presents additional complications because XML columns do not have locators, and when a table containing an XML column is created, no maximum length is defined for the XML object.

DB2 9 introduces the WITH CONTINUE clause on the FETCH statement to address the problems that are encountered when an application fetches LOB and XML data. You can use the WITH CONTINUE clause in an application to retrieve LOB and XML columns in multiple pieces without using a LOB locator, and continue a FETCH operation to retrieve the remaining data for LOB and XML

columns when truncation occurs. The application manages the buffers and reassembles the pieces of data. Two common uses for FETCH CONTINUE include:

- Dynamic allocation of appropriate storage size: You can use the initial FETCH statement to fetch data into a preallocated buffer of a moderate size. If the returned data item is too large to fit in that buffer, you can use the length information that is returned to allocate the right amount of storage, and use one FETCH CONTINUE statement to retrieve the remaining data.
- Streaming data through a single, fixed-size buffer: If more data remains after a FETCH operation, you can use as many FETCH CONTINUE statements as necessary to retrieve the data and use the same buffer area. In this case, the data in the buffer must be processed after each FETCH or FETCH CONTINUE operation.

File reference variables for large objects

A *file reference variable* is a host variable that is defined in a host language (for example C or COBOL) to contain the file name that directs file input and output for a large object (LOB). This release of DB2 for z/OS adds support for a LOB file reference variable.

With DB2 support for file reference variables, large LOB values can be inserted from a file or selected into a file rather than a host variable; the application does not need to acquire storage to contain the LOB value. File reference variables also facilitate the movement of LOB values from the database server to a client application or from a client application to a database server without going through the working storage of the client application. In addition, file reference variables bypass host language limitations on the maximum allowed size for LOB values that are located in working storage. The following host languages support file reference variables:

- C and C++
- COBOL
- PL/I
- Assembler
- REXX

Utilities enhancements for large objects

This release of DB2 for z/OS provides enhancements for handling large objects (LOBs) in the following utilities: LOAD, UNLOAD, REORG, CHECK LOB, and CHECK DATA.

Improved LOB handling for LOAD and UNLOAD utilities

Because of enhancements to the LOAD utility, an input field value can contain the name of the file that contains a LOB column value. The LOB column value is loaded from that file.

Enhancements to the UNLOAD utility let you store the value of a LOB column in a file and record the name of the file in the unloaded record in the base table.

Improved REORG of LOB table spaces

Before DB2 9, you cannot access LOB data during the REORG operation, and LOBs are moved within the existing LOB table space, which prevents physical space from being reclaimed from the LOB data set. Improvements in DB2 9 overcome those previous limitations. Now, the original LOB table space is drained of writers. All LOBs are then extracted from the original data set and inserted into a shadow

data set. When this operation is complete, all access to the LOB table space is stopped (the readers are drained) while the original data set is switched with the shadow data set. At this point, full access to the new data set is enabled, and an inline copy is taken to ensure recoverability of data.

CHECK LOB and CHECK DATA utilities

The options SHRLEVEL REFERENCE and SHRLEVEL CHANGE have been added to both CHECK LOB and CHECK DATA utilities. These options dramatically reduce the amount of time during which the data is in the read-only state when you run CHECK LOB and CHECK DATA utilities.

The CHECK DATA utility checks table spaces for violations of referential and table check constraints and reports information about violations that it detects. CHECK DATA also checks for consistency between a base table space and the corresponding LOB or XML table spaces. The new SHRLEVEL REFERENCE and SHRLEVEL CHANGE options provide the following functions:

- CHECK DATA with SHRLEVEL REFERENCE specifies that applications can read from but cannot write to the index, table space, or partition that is to be checked during CHECK DATA processing.
- CHECK DATA with SHRLEVEL CHANGE specifies that applications can read from and write to the index, table space, or partition that is to be checked during CHECK DATA processing.

The CHECK LOB utility identifies any structural defects in the LOB table space and any invalid LOB values. You run the CHECK LOB utility before the CHECK DATA utility if a table space contains at least one LOB column. The new SHRLEVEL REFERENCE and SHRLEVEL CHANGE options provide the following functions:

- CHECK LOB with SHRLEVEL REFERENCE specifies that applications can read from but cannot write to the table space that is to be checked during CHECK LOB processing.
- CHECK LOB with SHRLEVEL CHANGE specifies that applications can read from and write to the table space that is to be checked during CHECK LOB processing.

Performance enhancements for large objects

Several enhancements improve performance by eliminating locks for LOB operations and optimize retrieving data for small and medium LOBs.

Optimization of data retrieval for small and medium LOBs

In prior releases, processing large LOBs, particularly from a distributed perspective, has been optimized for retrieving larger amounts of data. Many applications effectively use locators to retrieve LOB data regardless of the size of the data that is retrieved. The mechanism involved incurs a separate network flow to determine the length of the data to be returned. Locators in this process remain active for the scope of the transaction, which can be longer than necessary for smaller amounts of data unless the locators are explicitly freed. As a result, valuable server resources are consumed.

DB2 9 introduces the ability for a server to dynamically determine the most efficient mode in which to return LOB or XML data. When the dynamic data format is enabled, the life span of the locator is the scope of the cursor (a cursor-based locator) instead of the scope of the transaction. A new mechanism is

also provided to allow the requester to retrieve sequential chunks of the LOB data while maintaining the position of the data (by means of the locator) at the server.

Increased performance through elimination of LOB locks

LOB locks are used to serialize LOB table space access and to determine whether the previously deallocated LOB space can be reallocated. In prior releases, DB2 acquires a lock on the LOB value while performing insert, update, delete, and select operations, and during LOB space allocation. This release of DB2 for z/OS eliminates the acquisition of locks on every LOB operation including LOB locks that are used for space allocation. The requirement for a LOB lock is removed for insert, delete, update, and select operations. In addition, a LOB lock is no longer required to serialize the consistency between the value of the LOB and the column of the base row for an uncommitted read operation. As a result, lock escalation for LOB locks at the table space level is also eliminated. This enhancement improves the overall elapsed time for LOB data retrieval.

SQL leadership: family firsts

This release of DB2 for z/OS provides leadership in SQL by offering a number of DB2 family firsts.

TRUNCATE TABLE statement

The TRUNCATE TABLE statement provides an efficient mechanism for deleting all data rows in a designated DB2 table. The table is deleted without activating delete triggers or altering the current table attributes in the DB2 catalog.

The statement also provides an IMMEDIATE option to permanently empty the designated DB2 table without issuing a commit, and it provides a REUSE STORAGE option to allow reuse of deallocated storage.

DECFLOAT built-in data type

DECFLOAT (decimal floating point) is a new built-in SQL data type, which has a maximum precision of 34 digits.

DECFLOAT data can be manipulated, stored, or loaded into DB2 tables.

A decimal floating-point value is an IEEE 754r (finite) number with a decimal point. The position of the decimal point is stored in each decimal floating-point value.

VARBINARY data type

This release of DB2 for z/OS provides support for the VARBINARY data type and function.

The VARBINARY data type represents a varying-length binary string. The VARBINARY function returns a varying-length binary string representation of a string of any type.

Related concepts:

BINARY data type and function

Enhancements to optimistic concurrency control and update detection

Enhancements to optimistic concurrency control provide a faster, more scalable locking alternative to database locking for concurrent data access. Also, a related enhancement provides a mechanism to detect recent (daily, weekly, or monthly) database updates.

Optimistic locking

Optimistic locking minimizes the time for which a given resource is unavailable for use by other transactions.

Because DB2 can determine when a row was changed, it can ensure data integrity while limiting the time that locks are held. With optimistic concurrency control, DB2 releases the row or page locks immediately after a read operation. To ensure data integrity, DB2 also releases the row lock after each FETCH operation and takes a new lock on a row only for a positioned update or delete.

A new row change timestamp column that you specify on the CREATE TABLE and ALTER TABLE statement lets you implement optimistic concurrency control. The column is defined with one of two options:

- NOT NULL GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
- NOT NULL GENERATED BY DEFAULT FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP.

DB2 maintains the contents of the row change timestamp column. When you want to use this change token as a condition when making an update, you can specify an appropriate condition for this column in your WHERE clause.

Daily, weekly, and monthly update detection

Database administrators benefit from knowing the volume of updates within specific time ranges so that they can plan for data replication, create auditing scenarios, and so on.

A new expression, ROW CHANGE, returns a token or a timestamp that represents the last change to a row. Now, an application has the following options:

- Determine when a row was last changed (or changed within a range of dates or number of days) by using the ROW CHANGE TIMESTAMP expression
- Return a token as a BIGINT (big integer) value that represents a relative point in the modification sequence of a row by using the ROW CHANGE TOKEN expression

MERGE and SELECT FROM MERGE statements

This release of DB2 for z/OS provides an easier and more efficient mechanism for merging data through the MERGE statement by using arrays of input.

In prior releases, merging data (from 100 transactions to a master table, for example) required many separate operations:

- 100 update operations to update existing rows in the master table
- Insert operation to insert zero to 100 rows from the transactions that do not currently exist in the master table

Now, the MERGE statement lets you update and insert many rows in a table from a single statement. You can embed the MERGE statement in an application

program or issue it interactively. The statement is executable and can be dynamically prepared. In addition, you can use the SELECT FROM MERGE statement to return all the updated rows and inserted rows, including column values that are generated by DB2.

Chapter 12. Planning for DB2 10 for z/OS

Every new version of DB2 for z/OS introduces some technical changes that you need to be aware of as you plan for your migration. When you migrate from DB2 9 to DB2 10, review the technical changes to ensure that your migration goes as smoothly as possible.

For example, DB2 10 includes changes to commands, utilities, SQL statements, the DB2 catalog, performance monitoring, and instrumentation facility component identifiers (IFCIDs).

This information is intended for all users of DB2, including application programmers, database administrators, and system programmers. It assumes that you are familiar with DB2 9.

This information assumes that your subsystem is running DB2 10 new-function mode. Generally, new functions that are described, including changes to existing functions, statements, and limits, are available only in new-function mode. Two exceptions to this general statement are new and changed utilities and optimization enhancements, which are also available in conversion mode unless stated otherwise.

Command changes in DB2 10

This release of DB2 for z/OS includes new and changed commands.

New commands in DB2 10

This release of DB2 for z/OS includes several new commands.

Table 1. New commands

Command	Description
BIND QUERY (DSN)	The DSN subcommand BIND QUERY reads the statement text, default schema, and a set of bind options from every row of DSN_USERQUERY_TABLE, and system-level access path hint information from correlated PLAN_TABLE rows. BIND QUERY then inserts the pertinent data into the SYSIBM.SYSQUERY, SYSIBM.SYSQUERYPLAN, and SYSIBM.SYSQUERYOPTS catalog tables.
FREE QUERY (DSN)	The DSN subcommand FREE QUERY removes one or more queries from the access path repository. Dynamic SQL queries, if specified, are also removed from the dynamic SQL cache.
MODIFY DDF (DB2)	The MODIFY DDF command modifies information regarding the status and configuration of DDF, as well as statistical information regarding connections or threads controlled by DDF.

Changed commands in DB2 10

This release of DB2 for z/OS includes changes to some DB2 commands.

The following table shows the existing commands that have new and changed options.

GUPI

Table 2. Changes to existing commands

Command	Description of enhancements and notes
-ACCESS DATABASE (DB2)	Additional authority: System DBADM
-ALTER BUFFERPOOL (DB2)	Additional option: PGSTEAL(NONE) specifies that no page stealing is to take place. DB2 pre-loads the data in the buffer pool when assigned objects are opened. The data stays resident in the buffer pool until the objects are closed. Page stealing is possible if the assigned objects do not fit in the buffer pool.
-ALTER UTILITY (DB2)	Additional authorities: • DATAACCESS • System DBADM
BIND PACKAGE (DSN)	Additional authority: System DBADM Additional options: • EXPLAIN(ONLY) • SQLERROR(CHECK) • EXTENDEDINDICATOR (NO | YES) • CONCURRENTACCESSRESOLUTION (USECURRENTLYCOMMITTED | WAITFOROUTCOME) • PLANMGMT (OFF | ON) • APREUSE(NO | YES) • APCOMPARE (NONE | WARN | ERROR)
BIND PLAN (DSN)	Additional option: CONCURRENTACCESSRESOLUTION (USECURRENTLYCOMMITTED | WAITFOROUTCOME)
-DECOMPOSE XML DOCUMENT	This command is deprecated in DB2 9, and removed from DB2 10.
DESCRIBE CALL	The SCALE information now includes the number of fractional digits or the number of fractional second digits for the timestamp.
-DISPLAY ARCHIVE (DB2)	Additional authority: System DBADM
-DISPLAY BUFFERPOOL (DB2)	Additional authority: System DBADM
-DISPLAY DATABASE	Additional authority: System DBADM Additional option: The ADVISORY option now includes the status choice *AREOR* to support online schema changes.
-DISPLAY DDF (DB2)	Additional authority: System DBADM
-DISPLAY FUNCTION SPECIFIC (DB2)	Additional authority: System DBADM Usage: You can display SQL functions in the output only if you invoke those functions in debug mode.
-DISPLAY GROUP (DB2)	Additional authority: System DBADM Usage: • DISPLAY GROUP DETAIL output now includes the current inline length of the LOB that contains bound package information. • DISPLAY GROUP output includes a status of DEACT in the STATUS column if a member is in the deactivated state.

Table 2. Changes to existing commands (continued)

Command	Description of enhancements and notes
-DISPLAY GROUPBUFFERPOOL (DB2)	Additional authority: System DBADM
-DISPLAY LOCATION (DB2)	Additional authority: System DBADM
-DISPLAY LOG (DB2)	Additional authority: System DBADM
-DISPLAY PROCEDURE (DB2)	Additional authority: System DBADM
-DISPLAY PROFILE (DB2)	Additional authorities: • DATAACCESS • System DBADM
-DISPLAY RLIMIT (DB2)	Additional authority: System DBADM
-DISPLAY THREAD (DB2)	Additional authority: System DBADM Usage: You can no longer use DB2 private protocol access to connect a database access thread to another database server location.
-DISPLAY TRACE (DB2)	Additional authority: System DBADM Additional option: The new AUDTPLCY option lets you specify up to eight audit policy names. Usage: DISPLAY TRACE output that is directed to the z/OS console is split across multiple messages if it exceeds 255 lines. Previously, if the output exceeded 255 lines, an abend occurred.
-DISPLAY UTILITY (DB2)	Additional authority: System DBADM
DSNH (TSO CLIST)	Removed PRIVATE choice from the DBPROTOCOL option. Private protocol is not supported in DB2 10.
FREE PACKAGE (DSN)	Additional authority: System DBADM
FREE PLAN (DSN)	Additional authority: System DBADM
-MODIFY TRACE (DB2)	Additional authorities: • SECADM • System DBADM
REBIND PACKAGE (DSN)	Additional authority: The privilege set for REBIND PACKAGE must include system DBADM authority. Additional options: • EXTENDEDINDICATOR(NO | YES) • CONCURRENTACCESSRESOLUTION (USECURRENTLYCOMMITTED | WAITFOROUTCOME) • APREUSE(NO | YES) • APCOMPARE (NONE | WARN | ERROR) Changed options: • PRIVATE has been removed from DBPROTOCOL option. Private protocol is not supported in DB2 10. • EXPLAIN option now includes ONLY keyword. • PLANMGMT option now includes ON keyword.
REBIND PLAN (DSN)	Additional authority: The privilege set for REBIND PLAN must include system DBADM authority. Additional option: CONCURRENTACCESSRESOLUTION (USECURRENTLYCOMMITTED | WAITFOROUTCOME) Changed option: PRIVATE has been removed from DBPROTOCOL option. Private protocol is not supported in DB2 10.

Table 2. Changes to existing commands (continued)

Command	Description of enhancements and notes
REBIND TRIGGER PACKAGE (DSN)	Additional options: • CONCURRENTACCESSRESOLUTION (USECURRENTLYCOMMITTED | WAITFOROUTCOME) • APREUSE(NO | YES) • APCOMPARE (NONE | WARN | ERROR) Changed options: • EXPLAIN option now includes ONLY keyword. • PLANMGMT option now includes ON keyword.
-RECOVER INDOUBT (DB2)	Additional authority: System DBADM
-RECOVER POSTPONED (DB2)	Additional authority: System DBADM
-SET LOG (DB2)	Additional options: • SINGLE specifies that only a single option, either LOGLOAD or CHKTIME is used to control checkpoint frequency. • BOTH specifies that both LOGLOAD and CHKTIME are used control checkpoint frequency. • CHKTIME(*integer*) option of LOGLOAD(*integer*) specifies the number of minutes between the start of successive checkpoints. • LOGLOAD(*integer*)option of CHKTIME(*integer*) specifies the number of log records that DB2 writes between the start of successive checkpoints. • NEWLOG(*data-set-name*) adds a newly defined active log data set to the active log inventory. • COPY(*log-copy*) specifies the log copy number for the new active log data set.
-SET SYSPARM (DB2)	Additional authority: SECADM
-START DATABASE (DB2)	Additional authority: System DBADM • Additional option: ACCESS(RREPL) Allows programs only read access to the specified databases, table spaces, indexes, or partitions, unless those programs were identified as replication programs.
-START DB2 (DB2)	• Changed options: ACCESS(MAINT) now allows access by user SECADM in addition to installation SYSADM and SYSOPR. • Additional option: DECP allows specification of a user-specified application defaults load module for the subsystem that is being started.
-START FUNCTION SPECIFIC (DB2)	Additional authority: System DBADM Usage: The START FUNCTION SPECIFIC command affects *all* versions of the SQL functions that you specify in the command.
-START PROCEDURE (DB2)	Additional authority: System DBADM Usage: The START PROCEDURE command affects *all* versions of the native SQL procedures that you specify in the command.
-START PROFILE (DB2)	Additional authorities: • DATAACCESS • System DBADM

Table 2. Changes to existing commands (continued)

Command	Description of enhancements and notes
-START TRACE (DB2)	Additional authorities: • SECADM • System DBADM Additional options: • STATEMENT places a statement header on the record. • AUDTPLCY(*policy-name*) gathers trace information for a list of up to eight audit policy names that you specify with this option. The maximum value for BUFSIZE value has increased to 67108864 (65536 KB).
-STOP DATABASE (DB2)	Additional authority: System DBADM
-STOP FUNCTION SPECIFIC (DB2)	Additional authority: System DBADM Usage: • STOP FUNCTION SPECIFIC is only applicable to external functions that run in the WLM application environment or non-inline SQL functions. STOP FUNCTION SPECIFIC cannot stop a built-in function, an inline SQL function, or a user-defined function that is sourced on another function. • The STOP FUNCTION SPECIFIC command affects *all* versions of the SQL functions that you specify in the command.
-STOP PROCEDURE (DB2)	Additional authority: System DBADM Usage: The STOP PROCEDURE command affects *all* versions of the native SQL procedures that you specify in the command.
-STOP PROFILE (DB2)	Additional authorities: • DATAACCESS • System DBADM
-STOP TRACE (DB2)	Additional authorities: • SECADM • System DBADM Additional option: The AUDTPLCY option stops the IFCIDs that correspond to the categories in the audit policies that you specify with this option.
-TERM UTILITY (DB2)	Additional authorities: • DATAACCESS • System DBADM

GUPI

Changes to utilities in DB2 10

This release of DB2 for z/OS contains no new utilities.

Utility option changes in DB2 10

This release of DB2 for z/OS does not include any new utilities. However, many existing utilities have new and changed options.

The following table lists and describes these new and changed options.

Table 3. New and changed utility options

Utility name	Description of enhancements and notes
CHECK DATA	**New options:** INCLUDE XML TABLESPACES, XMLSCHEMA, XMLCOLUMN, SCOPE XMLSCHEMAONLY INCLUDE XML TABLESPACES, XMLSCHEMA, XMLCOLUMN, SCOPE XMLSCHEMAONLY are added for XML support. CHECK DATA can check the integrity of XML documents and system-generated indexes that are associated with XML data.
COPY	**New options:** FLASHCOPY, FCCOPYDDN The FLASHCOPY option enables you to use FlashCopy technology to create an image copy of an object. The FCCOPYDDN option enables you to specify a template to use for the FlashCopy data set names.
DSNJU003	**New statements:** DELMBR, RSTMBR The DELMBR statement enables you to delete a member from a data sharing group. The RSTMBR statement enables you to restore a deactivated member to quiesced status in a data sharing group.
LISTDEF	**New options:** HISTORY, DEFINED **Changed option:** PARTLEVEL The HISTORY option supports the history tables that are associated with temporal tables. Specify HISTORY when you want the resulting object list to include history table spaces and index spaces. The DEFINED option enables you to specify whether the list is to include only table spaces or index spaces with defined data sets, only table spaces or index spaces with undefined data sets, or both. The PARTLEVEL option as been enhanced, so that you can now specify multiple disjoint partitions or partition ranges.

Table 3. New and changed utility options (continued)

Utility name	Description of enhancements and notes
LOAD	**New options:** • TIMESTAMP WITH TIME ZONE EXTERNAL • IMPLICIT_TZ • PERIODOVERRIDE • TRANSIDOVERRIDE • FLASHCOPY • FCCOPYDDN • BINARYXML • SPANNED **Changed option:** KEYCARD Both the TIMESTAMP WITH TIME ZONE data type option and the IMPLICIT_TZ option are provided to support the new SQL data type TIMESTAMP WITH TIME ZONE. Specify TIMESTAMP WITH TIME ZONE when you are loading of that data type. You can use the IMPLICIT_TZ option to specify a timezone when it is missing from the timestamp string that is being loaded, and the data type of corresponding column in the target table is TIMESTAMP WITH TIME ZONE. The PERIODOVERRIDE and TRANSIDOVERRIDE support temporal tables. Both of these options force data to be reloaded into temporal table columns that are defined as GENERATED ALWAYS. The PERIODOVERRIDE option applies to only system temporal table columns. The FLASHCOPY option enables you to use FlashCopy technology to create an inline copy of an object. The FCCOPYDDN option enables you to specify a template to use for the FlashCopy data set names. The BINARYXML option enables you to load an XML document in binary XML format. The SPANNED option enables you to load data from VBS data sets in spanned record format. This format can potentially improve performance when loading LOB and XML fields. The PARALLEL keyword specifies the maximum number of subtasks that are to be used in parallel to process the loading of a single input data set to reduce elapsed time. The KEYCARD option is deprecated. When you specify STATISTICS and INDEX, the KEYCARD functionality is used and cannot be disabled.
REBUILD INDEX	**New options:** FLASHCOPY, FCCOPYDDN **Changed option:** KEYCARD The FLASHCOPY option enables you to use FlashCopy technology to create an inline copy of an object. The FCCOPYDDN option enables you to specify a template to use for the FlashCopy data set names. The KEYCARD option is deprecated. When you specify STATISTICS, the KEYCARD functionality is used and cannot be disabled.

Table 3. New and changed utility options (continued)

Utility name	Description of enhancements and notes
RECOVER	**New options:** VERIFYSET, ENFORCE, BACKOUT The VERIFYSET keyword enables you to control whether a point-in-time recovery requires all base, LOB, XML, and history objects in a set. You can choose to break up point-in-time recoveries into multiple jobs and also avoid recovering objects that might not have changed. You can also use the ENFORCE keyword to specify whether CHKP and ACHKP pending states are set when only a subset of objects are recovered to a point in time. Use the BACKOUT option to specify whether a prior point in time recovery is to be done by rollback rather than by restoring the recovery base and using forward log processing. Such a rollback recovery could potentially decrease the amount of time that an object is unavailable during a recovery to a prior point in time if the specified recovery point in time is relatively recent.
REORG INDEX	**New options:** CURRENT_TIMESTAMP WITH TIME ZONE, FLASHCOPY, FCCOPYDDN, FORCE **Changed option:** KEYCARD The CURRENT_TIMESTAMP WITH TIME ZONE is provided to support the new SQL data type TIMESTAMP WITH TIME ZONE. You can specify this option in the labeled duration expression when you want to use a timestamp with a time zone. The FLASHCOPY option enables you to use FlashCopy technology to create an inline copy of an object. The FCCOPYDDN option enables you to specify a template to use for the FlashCopy data set names. The FORCE option enables the utility to cancel any blocking claimers when it needs to gain exclusive control over the target database objects. The KEYCARD option is deprecated. When you specify STATISTICS, the KEYCARD functionality is used and cannot be disabled.

Table 3. New and changed utility options (continued)

Utility name	Description of enhancements and notes
REORG TABLESPACE	**New options:** CURRENT_TIMESTAMP WITH TIME ZONE, AUTOESTSPACE, FLASHCOPY, FCCOPYDDN, AUX, FORCE, PARALLEL, SORTNPSI **Changed options:** PART, KEYCARD The CURRENT_TIMESTAMP WITH TIME ZONE is provided to support the new SQL data type TIMESTAMP WITH TIME ZONE. You can specify this option in the labeled duration expression when you want to use a timestamp with a time zone. The AUTOESTSPACE option supports hash access. Run REORG AUTOESTSPACE to improve access to rows in the overflow area. The FLASHCOPY option enables you to use FlashCopy technology to create an inline copy of an object. The FCCOPYDDN option enables you to specify a template to use for the FlashCopy data set names. The new AUX option enables you to specify whether REORG TABLESPACE is to also reorganize the LOB table spaces that are associated with the partitions of the base table space that is being organized. The FORCE option enables the utility to cancel any blocking claimers when it needs to gain exclusive control over the target database objects. The PARALLEL option lets you choose whether REORG TABLESPACE LIST processes partitions in parallel or serially. The PART option as been enhanced, so that you can now specify multiple disjoint partitions or partition ranges. The KEYCARD option is deprecated. When you specify STATISTICS and INDEX, the KEYCARD functionality is used and cannot be disabled. The SORTNPSI option enables you to specify when to sort all keys of a non-partitioned secondary index.
REPAIR	**New options:** NOAREORPEND, XMLTABLESPACE The NOAREORPEND option (in the SET clause of the REPAIR statement) supports the new advisory REORG-pending (AREOR) status. Specify NOAREORPEND to reset this status. The XMLTABLESPACE option enables you to specify an XML table space in which data is to be located for repair.

Table 3. New and changed utility options (continued)

Utility name	Description of enhancements and notes
RUNSTATS	**New options:** USE PROFILE, DELETE PROFILE, SET PROFILE, SET PROFILE FROM EXISTING STATS, UPDATE PROFILE, TABLESAMPLE SYSTEM **Changed options:** KEYCARD, SHRLEVEL The options USE PROFILE, DELETE PROFILE, SET PROFILE, and UPDATE PROFILE are added to support autonomic statistics. You can use these options to set up, use, and manage statistics profiles. If you specify SET PROFILE FROM EXISTING STATS, RUNSTATS automatically detects the statistics that were collected in the previous executions of RUNSTATS and builds a RUNSTATS profile to collect those same statistics. The TABLESAMPLE SYSTEM option helps improve performance when only a small subset of rows need to be examined in a large table space. The KEYCARD option is deprecated. The KEYCARD functionality is now built into the normal execution of the RUNSTATS INDEX utility and cannot be disabled. The default for SHRLEVEL is now CHANGE. In previous releases, the default was SHRLEVEL REFERENCE.
TEMPLATE	**New options:** TIME The TIME option specifies whether UTC or local time is used in expansion of date and time DSN variables. The default value is controlled by the value of the TEMPLATE_TIME subsystem parameter.
UNLOAD	**New options:** TIMESTAMP WITH TIME ZONE EXTERNAL, IMPLICIT_TZ, CURRENT_TIMESTAMP WITH TIME ZONE, BINARYXML, SPANNED The TIMESTAMP WITH TIME ZONE data type option, the IMPLICIT_TZ option, and the CURRENT_TIMESTAMP WITH TIME ZONE are provided to support the new SQL data type TIMESTAMP WITH TIME ZONE. Specify TIMESTAMP WITH TIME ZONE when you want the output field to have that data type. You can use the IMPLICIT_TZ option to specify a timezone when a value is being unloaded from a TIMESTAMP column with no time zone, and the target field is TIMESTAMP WITH TIME ZONE EXTERNAL. You can specify CURRENT_TIMESTAMP WITH TIME ZONE in the labeled duration expression when you want to use a timestamp with a time zone. The BINARYXML option enables you to unload XML data in binary XML format. The SPANNED option enables you to unload data to VBS data sets in spanned record format. This format can potentially improve performance when unloading LOB and XML fields.

Other utility changes in DB2 10

Other than new and changed utility options, this release of DB2 for z/OS includes additional changes to utilities.

FlashCopy support for image copies

In DB2 10, utilities that create image copies can do so by using the FlashCopy function that is provided by DFSMSdss. FlashCopy support requires FlashCopy Version 2 hardware. FlashCopy image copies are output to VSAM data sets.

You can use FlashCopy support for creating image copies only after the DB2 for z/OS subsystem is operating in DB2 10 new-function mode.

The following utilities can create image copies by using the FlashCopy function:

- COPY
- LOAD
- REBUILD INDEX
- REORG INDEX
- REORG TABLESPACE

The following utilities support FlashCopy image copies as input:

- COPYTOCOPY
- DSN1COMP
- DSN1COPY
- DSN1PRNT
- RECOVER

Utility enhancements for z/OS V1R12

In DB2 10, utilities that allocate sort work data sets have been enhanced to automatically use features that are provided by z/OS Version 1 Release 12 (V1R12).

The features that are provided by z/OS V1R12 reduce the limitations that are imposed by using storage below the 16 MB line. Because of these new V1R12 features, DB2 can open more data sets. Therefore, if multiple utilities are running in parallel, for those utilities that are invoked later, more storage below the 16 MB line remains available. This case can allow for an additional degree of subtask parallelism and thus performance might be improved.

The following utilities invoke DFSORT and might benefit from running on z/OS V1R12:

- CHECK DATA
- CHECK INDEX
- CHECK LOB
- LOAD
- REBUILD INDEX
- REORG INDEX
- REORG TABLESPACE
- RUNSTATS

Spatial index support

In DB2 10, the following utilities support spatial indexes:

- CHECK INDEX
- REBUILD INDEX

- REORG INDEX
- REORG TABLESPACE

Zero to eleven index keys can be generated for a single row of data.

How CHECK DATA and CHECK LOB affect check-pending states

In DB2 10, if a table space is not in CHECK-pending state before the CHECK DATA or CHECK LOB utility is run, the utility no longer places the table space in the CHECK-pending state when errors are found during processing.

As in previous releases of DB2 for z/OS, if the table space was already in a CHECK-pending state prior to running the CHECK DATA or CHECK LOB utility, and the utility finds errors during processing, the utility maintains the table space in the CHECK-pending state. If the utility finds no errors during processing, the CHECK-pending state is removed.

You can configure the CHECK DATA or CHECK LOB utility to place table spaces in a CHECK-pending state when errors are found by specifying YES on the new subsystem parameter CHECK_SETCHKP.

Utilities that skip new or obsolete catalog and directory objects

The following utilities skip new or obsolete catalog and directory objects during processing:

- COPY
- REBUILD INDEX
- RECOVER

Depending on the migration mode in which the DB2 for z/OS subsystem is running, these utilities skip new or obsolete catalog and directory objects during processing and issue message DSNU1530I for each skipped object. In conversion mode, these utilities skip catalog and directory objects that are new for DB2 10. In new function mode, these utilities skip catalog and directory objects that are obsolete in DB2 10. Specifying OPTIONS EVENT(ITEMERROR,SKIP) or OPTIONS EVENT(ITEMERROR,HALT) does not impact the skipping of new or obsolete objects.

The way that COPY handles these new and obsolete catalog and directory objects is slightly different if you specify a hardcoded DD statement to a tape device for the output image copy. In this case, the COPY utility opens and closes the data set to write a tape mark. This behavior ensures that subsequent image copies that are stacked to the same tape volume can be written.

Removed restriction for recovering from system-level backups

If a system-level backup is taken on z/OS Version 1 Release 11 or later and the DFSMS database copy pool is defined with the copy catalog option, the restriction against performing a recovery from a system level backup after any of the following utilities have run is removed:

- REORG TABLESPACE
- REORG INDEX
- REBUILD INDEX
- LOAD REPLACE

- RECOVER from image copy or concurrent copy

Recovery of catalog and directory objects

The order in which catalog and directory objects must be recovered has changed. See Recovering catalog and directory objects (DB2 Utilities).

How REORG TABLESPACE handles LOB table spaces

In DB2 10 new-function mode, SHRLEVEL NONE is the default value for REORG TABLESPACE. However, SHRLEVEL NONE is not supported when REORG is run against a LOB table space. Therefore if you specify REORG TABLESPACE SHRLEVEL NONE or REORG TABLESPACE without explicitly specifying a SHRLEVEL value against a LOB table space, the utility completes without reorganizing the LOB table space.

DB2 10 also removes restrictions that are related to the online reorganization of base table spaces that use LOB columns. However, if you altered the inline length for any LOB column, you still cannot use online REORG (REORG with SHRLEVEL CHANGE) until after that inline change has been materialized.

RUNSTATS behavior

The RUNSTATS utility has the following enhancements:

- The RUNSTATS utility now includes pseudo-empty pages when it determines the number of leaf pages. The number of leaf pages is maintained in column NLEAF of catalog table SYSINDEXES.
- The RUNSTATS utility is now zIIP-enabled.
- The RUNSTATS utility is enhanced to support statistics profiles for tables, which simplifies the RUNSTATS specification. In addition, autonomic features are introduced through a set of new catalog tables and stored procedures to further simplify and automate statistics collection.
- The RUNSTATS utility now supports page sampling for improved performance and reduced CPU consumption.
- The HIGH2KEY and LOW2KEY columns contain only the UTC portion of the TIMESTAMP WITH TIME ZONE value.

DSNJU004 (print log map) output

The output has been modified to include the following enhancements:

- The z/OS version number
- An indication of whether DFSMShsm has copied catalog information
- Information about deactivated data sharing members, and about destroyed data sharing members whose member IDs have not been reclaimed

REPORT output

The output has been modified to include the following enhancement:

- Information about deactivated data sharing members, and about destroyed data sharing members whose member IDs have not been reclaimed

SQL statement changes in DB2 10

This release of DB2 for z/OS provides new and changed SQL statements.

New SQL statements in DB2 10

This release of DB2 for z/OS includes new SQL statements.

GUPI

Table 4. New SQL statements

SQL statement	Description
ALTER FUNCTION (SQL table)	The ALTER FUNCTION (SQL table) statement changes the description of a user-defined SQL table function at the current server.
ALTER MASK	The ALTER MASK statement alters a column mask that exists at the current server.
ALTER PERMISSION	The ALTER PERMISSION statement alters a row permission that exists at the current server.
ALTER TRIGGER	The ALTER TRIGGER statement changes the description of a trigger at the current server.
CREATE FUNCTION (SQL table)	The CREATE FUNCTION (SQL table) statement creates an SQL table function at the current server. The function returns a set of rows.
CREATE MASK	The CREATE MASK statement creates a column mask for column access control at the current server. A column mask specifies what value should be returned for a specified column.
CREATE PERMISSION	The CREATE PERMISSION statement creates a row permission for row access control at the current server.
SET CURRENT EXPLAIN MODE	The SET CURRENT EXPLAIN MODE statement assigns a value to the CURRENT EXPLAIN MODE special register.
SET SESSION TIME ZONE	The SET SESSION TIME ZONE statement assigns a value to the SESSION TIME ZONE special register.

GUPI

Changed SQL statements in DB2 10

In this release of DB2 for z/OS, many existing SQL statements have new and changed clauses.

The following table shows the changes to existing SQL statements.

GUPI

Table 5. Changes to existing SQL statements

SQL statement	Description of enhancements and notes
ALTER FUNCTION (SQL scalar)	The ALTER FUNCTION (SQL scalar) statement has been enhanced to allow multiple versions of the same SQL scalar function to be created, changed, made active, or dropped. In addition, the options that are available to SQL scalar functions has been increased and the SQL procedural language can now be used in the body of SQL scalar functions.

Table 5. Changes to existing SQL statements (continued)

SQL statement	Description of enhancements and notes
ALTER INDEX	**New clauses:** **ADD INCLUDE COLUMN**
ALTER PROCEDURE (SQL native)	The ALTER PROCEDURE (SQL native) statement has been enhanced to provide more a robust declaration of parameters.
ALTER TABLE	**New clauses:** **ADD PERIOD** **GENERATED AS ROW BEGIN** or **END** **GENERATED AS TRANSACTION START ID** **ADD VERSIONING** **ALTER VERSION** **DROP VERSIONING** **ADD ORGANIZE BY HASH** **DROP ORGANIZATION** **ALTER ORGANIZATION** **INLINE LENGTH***integer*
ALTER TABLESPACE	**New clauses:** **DROP PENDING CHANGES** **DSSIZE** *integer* **G** **MEMBER CLUSTER YES** or **NO** **SEGSIZE** *integer* Running ALTER TABLESPACE with the MAXROWS clause no longer requires that the table space be stopped. However, after a change to the MAXROWS specification of the table space, the table space is placed in an advisory REORG-pending status.
COMMENT	Comments can now be made on individual versions of functions.
CREATE FUNCTION (SQL scalar)	The CREATE FUNCTION (SQL scalar) statement has been enhanced to allow multiple versions of the same SQL scalar function to be created. In addition, the options that are available to SQL scalar functions has been increased and the SQL procedure language can now be used in the body of SQL scalar functions.
CREATE INDEX	**New clauses:** **BUSINESS TIME WITHOUT OVERLAPS** **INCLUDE** *column-name,...*
CREATE PROCEDURE (SQL native)	The CREATE PROCEDURE (SQL native) statement has been enhanced to provide more a robust declaration of parameters.

Table 5. Changes to existing SQL statements (continued)

SQL statement	Description of enhancements and notes
CREATE TABLE	**New clauses:** **PERIOD SYSTEM_TIME** or **PERIOD BUSINESS_TIME** **INLINE LENGTH** *integer* **GENERATED AS ROW BEGIN** or **END** **GENERATED AS TRANSACTION START ID** **BUSINESS_TIME WITHOUT OVERLAPS** **ORGANIZE BY HASH**
CREATE TRIGGER	The CREATE TRIGGER statement has been enhanced to provide options that are used during the creation of trigger packages. Also the SQL statements that are available to use in the body of an AFTER or INSTEAD OF trigger has been expanded.
CREATE TYPE (distinct)	**New clauses:** **INLINE LENGTH** *integer*
DELETE	**New clauses:** **FOR PORTION OF BUSINESS_TIME**
GRANT	The **PUBLIC AT ALL LOCATIONS** clause is no longer allowed.
PREPARE	**New clauses:** **WITHOUT EXTENDED INDICATORS** or **WITH EXTENDED INDICATORS** **CONCENTRATE STATEMENTS OFF** or **CONCENTRATE STATEMENTS WITH LITERALS**
UPDATE	**New clauses:** **FOR PORTION OF BUSINESS_TIME**

GUPI

New functions in DB2 10

This release of DB2 for z/OS includes new built-in functions that improve the power of the SQL language.

The following table shows the new built-in functions.

GUPI

Table 6. New functions

Function name	Description
BITAND, BITANDNOT, BITOR, BITXOR, and BITNOT	The bit manipulation functions operate on the *twos complement* representation of the integer value of the input arguments. The functions return the result as a corresponding base 10 integer value in a data type that is based on the data type of the input arguments.
DECFLOAT_FORMAT	The DECFLOAT_FORMAT function returns a DECFLOAT(34) value that is based on the interpretation of the input string using the specified format.
DECODE	The DECODE function compares the first argument to each of the following arguments. If the first argument is equal to the argument being compared, or both the first argument and the argument being compared are null, the value of a result expression is returned. If no comparison argument matches the first argument, the value of an alternate return expression is returned. Otherwise a null value is returned.
DSN_XMLVALIDATE	The DSN_XMLVALIDATE returns an XML value that is the result of applying XML schema validation to the first argument of the function.
MEDIAN	The MEDIAN function returns the median of a set of numbers. This function can run only on an accelerator server.
NVL	The NVL function returns the first argument that is not null.
TIMESTAMP_TZ	The TIMESTAMP_TZ function returns a TIMESTAMP WITH TIME ZONE value from the input arguments.
TO_NUMBER	The TO_NUMBER function returns a DECFLOAT(34) value that is based on the interpretation of the input string using the specified format.
TRIM	The TRIM function removes bytes from the beginning, from the end, or from both the beginning and end of a string expression.
VERIFY_GROUP_FOR_USER	The VERIFY_GROUP_FOR_USER function returns a value that indicates whether the primary authorization ID and the group authorization IDs that are associated with first argument are in the authorization names that are specified in the list of the second argument.
VERIFY_ROLE_FOR_USER	The VERIFY_ROLE_FOR_USER function returns a value that indicates whether and of the roles that are associated with the authorization ID specified in the first argument are included in the role names that are specified in the list of the second argument.
VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER	The VERIFY_TRUSTED_CONTEXT_FOR_USER function returns a value that indicates whether the authorization ID that is associated with first argument has acquired a role in a trusted connection and that acquired role is included in the role names that are specified in the list of the second argument.
XMLMODIFY	The XMLMODIFY function returns an XML value that was possibly modified by the evaluation of an XQuery update expression using an implicit context item and the XQuery variables that were specified as input arguments.

GUPI

Reserved words

Certain words cannot be used as ordinary identifiers in some contexts because those words might be interpreted as SQL keywords. For example, ALL cannot be a column name in a SELECT statement. Each word, however, can be used as a delimited identifier in contexts where it otherwise cannot be used as an ordinary identifier. For example, if the quotation mark (") is the escape character that begins and ends delimited identifiers, "ALL" can appear as a column name in a SELECT statement.

Certain keywords might be interpreted as ordinary identifiers in some contexts rather than as keywords. For example, in the statement `SELECT * FROM SYSIBM.SYSTABLES WHERE`, WHERE is interpreted as an ordinary identifier specified as a correlation name, rather than as the beginning of an incomplete WHERE clause.

New reserved words for this version of DB2 for z/OS are identified with notes in this topic. In addition, some topics in this information might indicate words that cannot be used in the specific context that is being described.

IBM SQL has additional reserved words that DB2 for z/OS does not enforce. Therefore, you should not use these additional reserved words as ordinary identifiers in names that have a continuing use. See *IBM DB2 SQL Reference for Cross-Platform Development* for a list of the words.

GUPI

ADD	AND	ASUTIME
AFTER	ANY	AT
ALL	AS	AUDIT
ALLOCATE		AUX
ALLOW		AUXILIARY
ALTER	ASENSITIVE	
	ASSOCIATE	
BEFORE		
BEGIN		
BETWEEN		
BUFFERPOOL		
BY		
CALL	COLLECTION	CONTINUE
CAPTURE	COLLID	CREATE
CASCADED	COLUMN	
CASE	COMMENT	CURRENT
CAST	COMMIT	CURRENT_DATE
CCSID	CONCAT	CURRENT_LC_CTYPE
CHAR	CONDITION	CURRENT_PATH
CHARACTER	CONNECT	CURRENT_SCHEMA
CHECK	CONNECTION	CURRENT_TIME
CLONE	CONSTRAINT	CURRENT_TIMESTAMP
CLOSE	CONTAINS	CURRVAL[1]
CLUSTER	CONTENT	CURSOR

DATA	DELETE	DO
DATABASE	DESCRIPTOR	DOCUMENT
DAY	DETERMINISTIC	DOUBLE
DAYS	DISABLE	DROP
DBINFO	DISALLOW	DSSIZE
DECLARE	DISTINCT	DYNAMIC
DEFAULT		
EDITPROC	ENDING	EXECUTE
ELSE	END-EXEC[2]	EXISTS
ELSEIF	ERASE	EXIT
ENCODING	ESCAPE	EXPLAIN
ENCRYPTION	EXCEPT	EXTERNAL
END	EXCEPTION	
FENCED	FOR	
FETCH	FREE	
FIELDPROC	FROM	
FINAL	FULL	
FIRST[1]	FUNCTION	
GENERATED	GRANT	
GET	GROUP	
GLOBAL		
GO		
GOTO		
HANDLER		
HAVING		
HOLD		
HOUR		
HOURS		
IF	INHERIT	INSERT
IMMEDIATE	INNER	INTERSECT
IN	INOUT	INTO
INCLUSIVE	INSENSITIVE	IS
INDEX		ISOBID
		ITERATE
JAR		
JOIN		
KEEP		
KEY		
LABEL	LIKE	LOCK
LANGUAGE		LOCKMAX
LAST[1]	LOCAL	LOCKSIZE
LC_CTYPE	LOCALE	LONG
LEAVE	LOCATOR	LOOP
LEFT	LOCATORS	
MAINTAINED	MINUTES	
MATERIALIZED	MODIFIES	
MICROSECOND	MONTH	
MICROSECONDS	MONTHS	
MINUTE		
NEXT[1]	NOT	
NEXTVAL	NULL	
NO	NULLS	
NONE	NUMPARTS	

OBID	OPTIMIZE	
OF	OR	
	ORDER	
OLD[1]	ORGANIZATION[1]	
ON	OUT	
OPEN	OUTER	
OPTIMIZATION		
PACKAGE	PATH	PRIOR[1]
PARAMETER	PIECESIZE	PRIQTY
PART	PERIOD[1]	PRIVILEGES
PADDED	PLAN	PROCEDURE
PARTITION	PRECISION	PROGRAM
PARTITIONED	PREPARE	PSID
PARTITIONING	PREVVAL	PUBLIC
QUERY		
QUERYNO		
READS	RESULT_SET_LOCATOR	ROUND_DOWN
REFERENCES	RETURN	ROUND_FLOOR
REFRESH	RETURNS	ROUND_HALF_DOWN
RESIGNAL	REVOKE	ROUND_HALF_EVEN
RELEASE	RIGHT	ROUND_HALF_UP
RENAME	ROLE	ROUND_UP
REPEAT	ROLLBACK	ROW
RESTRICT		ROWSET
RESULT	ROUND_CEILING	RUN
SAVEPOINT	SET	STOGROUP
SCHEMA	SIGNAL	STORES
SCRATCHPAD	SIMPLE	STYLE
SECOND		SUMMARY
SECONDS	SOME	SYNONYM
SECQTY	SOURCE	SYSDATE[1]SYSTEM
SECURITY	SPECIFIC	SYSTIMESTAMP[1]
SEQUENCE	STANDARD	
SELECT	STATIC	
SENSITIVE	STATEMENT	
SESSION_USER	STAY	
TABLE	TRUNCATE	
TABLESPACE	TYPE	
THEN		
TO		
TRIGGER		
UNDO	USER	
UNION	USING	
UNIQUE		
UNTIL		
UPDATE		
VALIDPROC	VCAT	
VALUE		
VALUES	VIEW	
VARIABLE	VOLATILE	
VARIANT	VOLUMES	
WHEN		
WHENEVER		
WHERE		
WHILE		
WITH		
WLM		

XMLEXISTS XMLNAMESPACES XMLCAST
YEAR YEARS
ZONE[1]

Note:

1. New reserved word for DB2 10.
2. COBOL only

GUPI

Catalog changes in DB2 10

This release of DB2 for z/OS includes changed catalog tables, new catalog tables, new indexes on catalog tables, and the catalog structure is changed.

Some of the structural changes are applied during migration to conversion mode and some are applied during ENFM processing. For more information about the structural changes to the catalog in DB2 10, see "DB2 catalog restructured" on page 44.

Related reference:

DB2 catalog tables (DB2 SQL)

New catalog tables in DB2 10

This release of DB2 for z/OS includes new catalog tables.

GUPI

Table 7. New catalog tables

Catalog table name	Description
SYSIBM.SYSAUDITPOLICIES	The SYSIBM.SYSAUDITPOLICIES table contains one row for each audit policy.
SYSIBM.SYSAUTOALERTS	The SYSIBM.SYSAUTOALERTS table contains one row for each recommendation from autonomic procedures.
SYSIBM.SYSAUTOALERTS_OUT	The SYSIBM.SYSAUTOALERTS_OUT table is an auxiliary table for the OUTPUT column of the SYSIBM.SYSAUTOALERTS table.
SYSIBM.SYSAUTORUNS_HIST	The SYSIBM.SYSAUTORUNS_HIST table contains one row for each time an autonomic procedures has been run.
SYSIBM.SYSAUTORUNS_HISTOU	The SYSIBM.SYSAUTORUNS_HISTOU table is an auxiliary table for the OUTPUT column of the SYSIBM.SYSAUTORUNS_HIST table.
SYSIBM.SYSAUTOTIMEWINDOWS	The SYSIBM.SYSAUTOTIMEWINDOWS table contains one row for each time period during which autonomic procedures can be run.
SYSIBM.SYSCONTROLS	The SYSIBM.SYSCONTROLS table contains one row for each row permission and column mask.

Table 7. New catalog tables (continued)

Catalog table name	Description
SYSIBM.SYSDUMMYA	The SYSIBM.SYSDUMMYA table contains one row. The table is used for SQL statements in which a table reference is required, but the contents of the table are not important. SYSIBM.SYSDUMMYA resides in table space SYSTSASC, which is a ASCII table space.
SYSIBM.SYSDUMMYE	The SYSIBM.SYSDUMMYE table contains one row. The table is used for SQL statements in which a table reference is required, but the contents of the table are not important. SYSIBM.SYSDUMMYE resides in table space SYSEBCDC, which is a EBCDIC table space.
SYSIBM.SYSDUMMYU	The SYSIBM.SYSDUMMYU table contains one row. The table is used for SQL statements in which a table reference is required, but the contents of the table are not important. SYSIBM.SYSDUMMYU resides in table space SYSTSUNI, which is a UNICODE table space.
SYSIBM.SYSINDEXES_RTSECT	The SYSIBM.SYSINDEXES_RTSECT table is an auxiliary table for the RTSECTION column of the SYSIBM.SYSINDEXES table and is required to hold LOB data.
SYSIBM.SYSINDEXES_TREE	The SYSIBM.SYSINDEXES_TREE table is an auxiliary table for the PARSETREE column of the SYSIBM.SYSINDEXES table and is required to hold LOB data.
SYSIBM.SYSPACKCOPY	The SYSIBM.SYSPACKCOPY table contains one row for the previous version of each package and one row for the original version of each package.
SYSIBM.SYSPACKSTMT_STMB	The SYSIBM.SYSPACKSTMT_STMB table is an auxiliary table for the STMTBLOB column of the SYSIBM.SYSPACKSTMT table and is required to hold LOB data.
SYSIBM.SYSPACKSTMT_STMT	The SYSIBM.SYSPACKSTMT_STMT table is an auxiliary table for the STATEMENT column of the SYSIBM.SYSPACKSTMT table and is required to hold LOB data.
SYSIBM.SYSPENDINGDDL	The SYSIBM.SYSPENDINGDDL table contains information about which objects have pending definition changes. The entries only exist during the window between when the pending option is executed and when the utility applies these pending changes to the object.
SYSIBM.SYSPENDINGOBJECTS	The SYSIBM.SYSPENDINGOBJECTS table contains the name of and OBID information about objects that are the pending creation. The data sets for these objects are created but the object definition have not been materialized to the catalog. The entries in this table only exist during the time between when the names of the new objects are generated and when the catalog definition of the new objects are materialized.
SYSIBM.SYSQUERY	The SYSIBM.SYSQUERY table contains one row for each query in a set of queries.
SYSIBM.SYSQUERY_AUX	The SYSIBM.SYSQUERY_AUX table is an auxiliary table for the STMTTEXT column of the SYSIBM.SYSQUERY table.
SYSIBM.SYSQUERYOPTS	The SYSIBM.SYSQUERYOPTS table contains optimization parameters for the queries that are in SYSIBM.SYSQUERY.
SYSIBM.SYSQUERYPLAN	The SYSIBM.SYSQUERYPLAN table contains the plan hint information for the queries in the SYSIBM.SYSQUERY table. It correlates to the SYSIBM.SYSQUERY table by the QUERYID column. For a query, there can be up to 3 copies of plan hints stored in the SYSIBM.SYSQUERYPLAN table, distinguished by the value of the COPYID column.
SYSIBM.SYSROUTINES_TREE	The SYSIBM.SYSROUTINES_TREE table is an auxiliary table for the PTREE column of the SYSIBM.SYSROUTINES table.

Table 7. New catalog tables (continued)

Catalog table name	Description
SYSIBM.SYSTABLES_PROFILES	The SYSIBM.SYSTABLES_PROFILES table contains one row for each profile that is associated with a table in SYSIBM.SYSTABLES.
SYSIBM.SYSTABLES_PROFILE_TEXT	The SYSIBM.SYSTABLES_PROFILE_TEXT table is an auxiliary table for the PROFILE_TEXT column of the SYSIBM.SYSTABLES_PROFILES table and is required to hold LOB data.
SYSIBM.SYSTRIGGERS_STMT	The SYSIBM.SYSTRIGGERS_STMT table is an auxiliary table for the STATEMENT column of the SYSIBM.SYSTRIGGERS table and is required to hold LOB data.
SYSIBM.SYSVIEWS_STMT	The SYSIBM.SYSVIEWS_STMT table is an auxiliary table for the STATEMENT column of the SYSIBM.SYSVIEWS table and is required to hold LOB data.
SYSIBM.SYSVIEWS_TREE	The SYSIBM.SYSVIEWS_TREE table is an auxiliary table for the PARSETREE column of the SYSIBM.SYSVIEWS table and is required to hold LOB data.

GUPI

Changed catalog tables in DB2 10

Many existing catalog tables are changed in this release of DB2 for z/OS.

The catalog is also restructured in DB2 10. Some of the structural changes are applied during migration to conversion mode. For more information about the structural changes to the catalog in DB2 10, see "DB2 catalog restructured" on page 44.

The following table shows a list of the new columns and the existing columns that are revised. Revisions to columns include new column descriptions, new values for a column, changed data types, changed column lengths, or both changed data types and lengths.

GUPI

Table 8. Summary of new and revised catalog table columns

Catalog table name	New column	Revised column
SYSIBM.SYSCHECK	PERIOD	
SYSIBM.SYSCOLUMNS	• HASHKEY_COLSEQ • CONTROL_ID • XML_TYPEMOD_ID • PERIOD • GENERATED_ ATTRIBUTE	• COLTYPE • LENGTH • SCALE • DEFAULT
SYSIBM.SYSCOPY		• ICBACKUP • STYPE • TTYPE
SYSIBM.SYSDATATYPES	INLINE_LENGTH	SCALE

Table 8. Summary of new and revised catalog table columns (continued)

Catalog table name	New column	Revised column
SYSIBM.SYSDBAUTH		AUTHHOWGOT
SYSIBM. SYSDEPENDENCIES	• BVERSION • DVERSION	• BNAME • BTYPE
SYSIBM.SYSINDEXES	• HASH • SPARSE • ROWID	• UNIQUERULE • INDEXTYPE • UNIQUE_COUNT
SYSIBM.SYSINDEXPART		• STORNAME • SPACE
SYSIBM. SYSINDEXSPACESTATS	• REORGINDEXACCESS • DRIVETYPE	NPAGES
SYSIBM. SYSKEYCOLUSE	PERIOD	
SYSIBM.SYSKEYS	PERIOD	ORDERING
SYSIBM. SYSKEYTARGETS		• LENGTH • SCALE
SYSIBM. SYSOBJROLEDEP		DTYPE
SYSIBM.SYSPACKAGE	• LASTUSED • CONCUR_ACC_RES • EXTENDED- INDICATOR • PLANMGMT • PLANMGMTSCOPE • APRETAINDUP • RECORD- TEMPORALHIST	• HOSTLANG • TYPE
SYSIBM.SYSPACKAUTH		AUTHHOWGOT
SYSIBM.SYSPACKDEP		• BNAME • BQUALIFIER • BTYPE • DTYPE
SYSIBM.SYSPACKSTMT	• ROWID • STATEMENT • STMT_ID	SEQNO
SYSIBM.SYSPARMS		SCALE
SYSIBM.SYSPLAN	• CONCUR_ACC_RES	
SYSIBM.SYSPLANAUTH		AUTHHOWGOT
SYSIBM.SYSRESAUTH		AUTHHOWGOT
SYSIBM. SYSROUTINEAUTH		AUTHHOWGOT

Table 8. Summary of new and revised catalog table columns (continued)

Catalog table name	New column	Revised column
SYSIBM.SYSROUTINES	• SECURE • INLINE • SYSTEM_DEFINED	VERSION
SYSIBM. SYSSCHEMAAUTH		AUTHHOWGOT
SYSIBM. SYSSEQUENCEAUTH		AUTHHOWGOT
SYSIBM.SYSSTMT		SEQNO
SYSIBM.SYSTABAUTH		AUTHHOWGOT
SYSIBM.SYSTABLEPART	• HASHSPACE • HASHDATAPAGES	• STORNAME • PERCACTIVE • CHECKFLAG
SYSIBM.SYSTABLES	• HASHKEY- COLUMNS • CONTROL • VERSIONING_ SCHEMA • VERSIONING_ TABLE	TYPE
SYSIBM.SYSTABLESPACE	• MEMBER_ CLUSTER • ORGANIZATION- TYPE • HASHSPACE • HASHDATA- PAGES	DSSIZE
SYSIBM. SYSTABLESPACESTATS	• REORGSCAN- ACCESS • REORGHASH- ACCESS • HASHLASTUSED • REORG- CLUSTERSENS • DRIVETYPE • LPFACILITY	
SYSIBM.SYSTRIGGERS	• SECURE • ALTEREDTS • ROWID • STATEMENT	

Table 8. Summary of new and revised catalog table columns (continued)

Catalog table name	New column	Revised column
SYSIBM.SYSUSERAUTH	• EXPLAINAUTH • SQLADMAUTH • SDBADMAUTH • DATAACCESSAUTH • ACCESSCTRLAUTH • CREATE-SECUREAUTH	AUTHHOWGOT
SYSIBM.SYSVIEWS	• ROWID • STATEMENT	

GUPI

New and changed indexes in DB2 10

This release of DB2 for z/OS includes new indexes and changes to existing indexes.

The catalog for this release of DB2 for z/OS has been significantly reorganized, with many catalog tables now residing in new table spaces. For information about the new and changed indexes for catalog tables, and information about catalog table space changes, see Table spaces and indexes.

Performance monitoring and tuning changes in DB2 10

This release of DB2 for z/OS includes changes that affect performance and changes to monitoring tools, such as EXPLAIN.

Performance changes in DB2 10

This release of DB2 for z/OS provides a number of performance enhancements.

Hash access to individual table rows

Hash access allows DB2 to directly access a single row in a table and avoid scanning the table or index for a matching equal predicate. This access path requires only one I/O operation to retrieve a row from a table. Hash access reduces CPU workload, but requires more disk space to function properly. Tables with hash access can require up to twice as much disk space as tables that use only indexes or table scans.

Improved LOB and XML processing

DB2 10 for z/OS reduces LOB and XML materialization into buffers significantly for local and distributed processing. This improvement reduces virtual storage consumption and therefore improves system stability. Class 2 CPU time is also reduced. Whether materialization is reduced and by how much it is reduced depends on the type of operation and the quantity and size of the LOB and XML data.

LOB data

You can use the INLINE_LOB_LENGTH subsystem parameter or the INLINE LENGTH clause with a CREATE TABLE or CREATE DISTINCT TYPE statement to specify that DB2 store a compatible portion of LOB data in the base table space with the other non-LOB data. You can expect processing for an inline CLOB to compare to that for a VARCHAR column in terms of CPU usage and elapsed time.

Locking

Utilities and utility-related commands do not acquire or hold the UTSERIAL lock. The UTSERIAL lock is replaced by more granular locks to reduce contention.

Related concepts:

LOB and XML materialization (DB2 Application programming and SQL)

Elimination of UTSERIAL lock for DB2 utilities

Related tasks:

Organizing tables by hash for fast access to individual rows (DB2 Performance)

Improving performance for LOB data (DB2 Performance)

EXPLAIN table changes in DB2 10

Before you can use EXPLAIN, you must create an EXPLAIN table to hold the captured information. Certain optimization tools depend on instances of EXPLAIN tables. You can create and maintain a set of EXPLAIN tables, or SQL optimization tools also create and maintain instances of EXPLAIN tables.

PSPI

Tables of the correct format for DB2 10 are created when you run the DSNTIJRT installation job. You can also view the SQL statements for creating EXPLAIN tables in the correct format for DB2 10 in member DSNTESC of the SDSNSAMP library.

Important: It is best to convert EXPLAIN tables to DB2 10 format during migration, or soon after migration. In DB2 10, the EXPLAIN function supports tables that have only the DB2 10, DB2 9, or Version 8 formats. However, DB2 9 format and Version 8 format EXPLAIN tables are deprecated. If you invoke EXPLAIN and DB2 9 or Version 8 tables are used, DB2 issues SQL code +20520. If tables of an unsupported format are found, DB2 issues SQL code -20008 and the EXPLAIN operation fails.

All EXPLAIN tables must be encoded in UNICODE format to be compatible with DB2 10 new-function mode.

PSPI

Related concepts:

Investigating SQL performance by using EXPLAIN (DB2 Performance)

Related tasks:

Converting EXPLAIN tables (before migration) (DB2 Installation and Migration)

- Creating EXPLAIN tables (DB2 Performance)
- Working with and retrieving EXPLAIN table data (DB2 Performance)
- Correlating information across EXPLAIN tables (DB2 Performance)

Related reference:

- Columns for correlating EXPLAIN tables (DB2 Performance)
- EXPLAIN tables (DB2 Performance)
- EXPLAIN (DB2 SQL)

Format of PLAN_TABLE in DB2 10

EXPLAIN tables, including PLAN_TABLE, of the correct format for DB2 10 are created when you run the DSNTIJSG installation job. You can also view sample CREATE TABLE statements for the DB2 10 in member DSNTESC of the SDSNSAMP library.

Plan table

PSPI

The following PLAN_TABLE formats, and only these, are supported in DB2 10. Previous formats with fewer than 58 columns are no longer supported.

DB2 10 format
All columns shown in the sample CREATE TABLE statement, up to and including the MERGN column (COLCOUNT=64).

DB2 9 format
All columns shown in the sample CREATE TABLE statement, to and including the PARENT_PLANNO column (COLCOUNT=59). This format is deprecated. For information about converting tables in this format to the current format, see Migration step 24: Convert EXPLAIN tables to the current format and encoding type (DB2 Installation and Migration).

DB2 Version 8 format
All columns shown in the sample CREATE TABLE statement, up to and including the STMTTOKEN column (COLCOUNT=58). This format is deprecated. For information about converting tables in this format to the current format, see Converting EXPLAIN tables for migration from DB2 Version 8 (DB2 Installation and Migration).

The following PLAN_TABLE columns are added in DB2 10:

BIND_EXPLAIN_ONLY
Whether the row was inserted because a command specified the EXPLAIN(ONLY) option. The data type is CHAR(1).

SECTNOI
The section number of a static SQL statement. The data type is INTEGER.

EXPLAIN_TIME
The time when the EXPLAIN information was captured. The data type is TIMESTAMP.

MERGC
Whether the composite table is consolidated before the join. The data type is CHAR(1).

MERGN
Indicates whether the new table is consolidated before the join. The data type is CHAR(1)

Data types of existing columns have not changed from DB2 9. However, if you are migrating to the DB2 10 format from an earlier format, you might have change the data types of additional columns.

For information about changes that were applied in DB2 9 see "Format of PLAN_TABLE in DB2 9" on page 174.

Formats of additional EXPLAIN tables

For the formats of the following additional EXPLAIN tables in DB2 10, see member DSNTESC in the SDSNSAMP library:

- DSN_STATEMNT_TABLE (DB2 Performance)
- DSN_FUNCTION_TABLE (DB2 Performance)
- DSN_STATEMENT_CACHE_TABLE (DB2 Performance)
- DSN_DETCOST_TABLE (DB2 Performance)
- DSN_FILTER_TABLE (DB2 Performance)
- DSN_PGRANGE_TABLE (DB2 Performance)
- DSN_PGROUP_TABLE (DB2 Performance)
- DSN_PREDICAT_TABLE (DB2 Performance)
- DSN_PTASK_TABLE (DB2 Performance)
- DSN_QUERY_TABLE (DB2 Performance)
- DSN_SORTKEY_TABLE (DB2 Performance)
- DSN_SORT_TABLE (DB2 Performance)
- DSN_VIEWREF_TABLE (DB2 Performance)
- DSN_VIRTUAL_INDEXES (DB2 Performance)
- DSN_COLDIST_TABLE (DB2 Performance)
- DSN_KEYTGTDIST_TABLE (DB2 Performance)

PSPI

Related concepts:

Investigating SQL performance by using EXPLAIN (DB2 Performance)

Related tasks:

Converting EXPLAIN tables (before migration) (DB2 Installation and Migration)

Creating EXPLAIN tables (DB2 Performance)

Working with and retrieving EXPLAIN table data (DB2 Performance)

Related reference:

PLAN_TABLE (DB2 Performance)

EXPLAIN (DB2 SQL)

New and changed EXPLAIN table columns in DB2 10

New columns are added to certain EXPLAIN tables in this release of DB2 for z/OS.

This topic describes only new and changed columns for the following EXPLAIN tables:

For the complete set of column descriptions for each table, see EXPLAIN tables. For the current format, and sample CREATE TABLE statements, see member DSNTESC of the SDSNSAMP library.

New columns for consistency among EXPLAIN tables

To improve the consistency of EXPLAIN tables, certain columns are added to each EXPLAIN tables if those columns did not previously exist in the table. In cases where the column already existed, the description might have changed to match the information in the following table.

Table 9. Descriptions of new columns for consistency among EXPLAIN tables.

Column name	Data type	Description	Change Type
SECTNOI	INTEGER NOT NULL DEFAULT -1	The section number of the statement. The value is taken from the same column in SYSPACKSTMT or SYSSTMT tables and can be used to join tables to reconstruct the access path for the statement. This column is applicable only for static statements. The default value of -1 indicates EXPLAIN information that was captured in DB2 10 for z/OS or earlier versions of DB2.	New unless previously existed

Table 9. Descriptions of new columns for consistency among EXPLAIN tables. (continued)

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL WITH DEFAULT	A number that identifies the statement that is being explained. The origin of the value depends on the context of the row: **For rows produced by EXPLAIN statements** The number specified in the QUERYNO clause, which is an optional part of the SELECT, INSERT, UPDATE, MERGE, and DELETE statement syntax. **For rows not produced by EXPLAIN statements** DB2 assigns a number that is based on the line number of the SQL statement in the source program. When the values of QUERYNO are based on the statement number in the source program, values that exceed 32767 are reported as 0. However, in a very long program, the value is not guaranteed to be unique. If QUERYNO is not unique, the value of EXPLAIN_TIME is unique. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, if the QUERYNO clause is specified, its value is used by DB2. Otherwise DB2 assigns a number based on the line number of the SQL statement in the compiled SQL function or native SQL procedure.	New unless previously existed
APPLNAME	VARCHAR(24) NOT NULL WITH DEFAULT[1]	The name of the application plan for the row. Applies only to embedded EXPLAIN statements that are executed from a plan or to statements that are explained when binding a plan. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column is not used, and is blank.	New unless previously existed

Table 9. Descriptions of new columns for consistency among EXPLAIN tables. (continued)

Column name	Data type	Description	Change Type
PROGNAME	VARCHAR(128) NOT NULL WITH DEFAULT	The name of the program or package containing the statement being explained. Applies only to embedded EXPLAIN statements and to statements explained as the result of binding a plan or package. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the specific name of the compiled SQL function or native SQL procedure.	New unless previously existed
COLLID	VARCHAR(128) NOT NULL WITH DEFAULT	The collection ID: **DSNDYNAMICSQLCACHE** The row originates from the dynamic statement cache **DSNEXPLAINMODEYES** The row originates from an application that specifies YES for the value of the CURRENT EXPLAIN MODE special register. **DSNEXPLAINMODEEXPLAIN** The row originates from an application that specifies EXPLAIN for the value of the CURRENT EXPLAIN MODE special register. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the schema name of the compiled SQL function or native SQL procedure.	New unless previously existed
VERSION	PLAN_TABLE; VARCHAR(122) NOT NULL WITH DEFAULT	The version identifier for the package. Applies only to an embedded EXPLAIN statement executed from a package or to a statement that is explained when binding a package. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the version identifier of the compiled SQL function or native SQL procedure.	New unless previously existed

Notes:

1. The data type of this column is VARCHAR(128) NOT NULL in the following tables:
   - DSN_COLDIST_TABLE
   - DSN_KEYTGTDIST_TABLE

PLAN_TABLE

The following table describes new and changed columns in PLAN_TABLE. For the current format of PLAN_TABLE and the complete set of column descriptions, see PLAN_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 10. Descriptions of new and changed columns in PLAN_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description

Table 10. Descriptions of new and changed columns in PLAN_TABLE (continued)

Column name	Data type	Description	Change Type
ACCESSTYPE	CHAR(2) NOT NULL	The method of accessing the new table: **DI** By an intersection of multiple DOCID lists to return the final DOCID list **DU** By a union of multiple DOCID lists to return the final DOCID list **DX** By an XML index scan on the index that is named in ACCESSNAME to return a DOCID list **E** By direct row access using a row change timestamp column. **H** By hash access. IF an overflow condition occurs, the hash overflow index that is identified by ACCESSCREATOR and ACCESSNAME is used. **HN** By has access using an IN predicate, or an IN predicate that DB2 generates. If a hash overflow condition occurs, the hash overflow index that is identified in ACCESSCREATOR and ACCESSNAME is used. **I** By an index (identified in ACCESSCREATOR and ACCESSNAME) **IN** By an index scan when the matching predicate contains an IN predicate and the IN-list is accessed through an in-memory table. **I1** By a one-fetch index scan **M** By a multiple index scan (followed by MX, MI, MU, or MH) **MH** By the hash overflow index named in ACCESSNAME **MI** By an intersection of multiple indexes **MU** By a union of multiple indexes **MX** By an index scan on the index named in ACCESSNAME. When the access method MX follows the access method DX, DI, or DU, the table is accessed by the DOCID index by using the DOCID list that is returned by DX, DI, or DU. **N** • By an index scan when the matching predicate contains the IN keyword • By an index scan when DB2 rewrites a query using the IN keyword • By hash access with the IN keyword • By hash access when DB2 rewrites a query using the IN keyword **NR** Range list access. **P** By a dynamic pair-wise index scan **R** By a table space scan **RW** By a work file scan of the result of a materialized user-defined table function **V** By buffers for an INSERT statement within a SELECT **blank** Not applicable to the current row	Changed description
VERSION	VARCHAR(122) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
COLLID	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description

Table 10. Descriptions of new and changed columns in PLAN_TABLE (continued)

Column name	Data type	Description	Change Type
TABLE_TYPE	CHAR(1)	The type of new table: **B** Buffers for SELECT from INSERT, SELECT from UPDATE, SELECT from MERGE, or SELECT from DELETE statement. **C** Common table expression **F** Table function **I** The new table is generated from an IN-LIST predicate. If the IN-LIST predicate is selected as the matching predicate, it will be accessed as an in-memory table. **M** Materialized query table **Q** Temporary intermediate result table (not materialized). For the name of a view or nested table expression, a value of Q indicates that the materialization was virtual and not actual. Materialization can be virtual when the view or nested table expression definition contains a UNION ALL that is not distributed. **R** Recursive common table expression **S** Subquery (correlated or non-correlated) **T** Table **W** Work file The value of the column is null if the query uses GROUP BY, ORDER BY, or DISTINCT, which requires an implicit sort.	Changed description
TIMESTAMP	CHAR(16)	This column is deprecated. Use EXPLAIN_TIME instead.	Changed description
HINT_USED	VARCHAR(128) NOT NULL WITH DEFAULT	If DB2 used one of your optimization hints, it puts the identifier for that hint (the value in OPTHINT) in this column. When a hint that is stored in the SYSIBM.SYSQUERYPLAN catalog table is used, the value of this column is SYSQUERYPLAN:*query-id*, where *query-id* is the value of the QUERYID column for the hint in the SYSQUERYPLAN catalog table.	Changed description
BIND_ EXPLAIN_ ONLY	CHAR (1) NOT NULL WITH DEFAULT 'N'	Identifies whether the row is inserted by the BIND command with EXPLAIN(ONLY) option.	New column
SECTNOI	INTEGER NOT NULL DEFAULT -1	See New columns for consistency among EXPLAIN tables.	New column

Table 10. Descriptions of new and changed columns in PLAN_TABLE (continued)

Column name	Data type	Description	Change Type
EXPLAIN_ TIME	TIMESTAMP NOT NULL WITH DEFAULT	The time when the EXPLAIN information was captured: **All cached statements** When the statement entered the cache, in the form of a full-precision timestamp value. **Non-cached static statements** When the statement was bound, in the form of a full precision timestamp value. **Non-cached dynamic statements** When EXPLAIN was executed, in the form of a value equivalent to a CHAR(16) representation of the time appended by 4 zeros.	New column
MERGC	CHAR(1)	Indicates whether the composite table is consolidated before the join. **Y** Yes **N** No	New column
MERGN	CHAR(1)	Indicates whether the new table is consolidated before the join. **Y** Yes **N** No	New column

DSN_COLDIST_TABLE (new table)

The following table describes the columns in the new DSN_COLDIST_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

The following table shows the descriptions of the columns in the DSN_COLDIST_TABLE table.

Table 11. Descriptions of columns in DSN_COLDIST_TABLE

Column name	Data Type	Description
QUERYNO	INTEGER NOT NULL	A number that identifies the statement that is being explained. The origin of the value depends on the context of the row: **For rows produced by EXPLAIN statements** The number specified in the QUERYNO clause, which is an optional part of the SELECT, INSERT, UPDATE, MERGE, and DELETE statement syntax. **For rows not produced by EXPLAIN statements** DB2 assigns a number that is based on the line number of the SQL statement in the source program. When the values of QUERYNO are based on the statement number in the source program, values that exceed 32767 are reported as 0. However, in certain rare cases, the value is not guaranteed to be unique. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, if the QUERYNO clause is specified, its value is used by DB2. Otherwise DB2 assigns a number based on the line number of the SQL statement in the compiled SQL function or native SQL procedure.
APPLNAME	VARCHAR(128) NOT NULL	The name of the application plan for the row. Applies only to embedded EXPLAIN statements that are executed from a plan or to statements that are explained when binding a plan. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column is not used, and is blank.
PROGNAME	VARCHAR(128) NOT NULL	The name of the program or package containing the statement being explained. Applies only to embedded EXPLAIN statements and to statements explained as the result of binding a plan or package. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the specific name of the compiled SQL function or native SQL procedure.

Table 11. Descriptions of columns in DSN_COLDIST_TABLE (continued)

Column name	Data Type	Description
COLLID	VARCHAR(128) NOT NULL	The collection ID: **'DSNDYNAMICSQLCACHE'** The row originates from the dynamic statement cache **'DSNEXPLAINMODEYES'** The row originates from an application that specifies YES for the value of the CURRENT EXPLAIN MODE special register. **'DSNEXPLAINMODEEXPLAIN'** The row originates from an application that specifies EXPLAIN for the value of the CURRENT EXPLAIN MODE special register. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the schema name of the compiled SQL function or native SQL procedure.
GROUP_MEMBER	VARCHAR(128) NOT NULL	The member name of the DB2 that executed EXPLAIN. The column is blank if the DB2 subsystem was not in a data sharing environment when EXPLAIN was executed.
SECTNOI	INTEGER NOT NULL	The section number of the statement. The value is taken from the same column in SYSPACKSTMT or SYSSTMT tables and can be used to join tables to reconstruct the access path for the statement. This column is applicable only for static statements. The default value of -1 indicates EXPLAIN information that was captured in DB2 9 or earlier.
VERSION	VARCHAR(122) NOT NULL	The version identifier for the package. Applies only to an embedded EXPLAIN statement executed from a package or to a statement that is explained when binding a package. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the version identifier of the compiled SQL function or native SQL procedure.

Table 11. Descriptions of columns in DSN_COLDIST_TABLE (continued)

Column name	Data Type	Description
EXPLAIN_TIME	TIMESTAMP NOT NULL	The time when the EXPLAIN information was captured: **All cached statements** When the statement entered the cache, in the form of a full-precision timestamp value. **Non-cached static statements** When the statement was bound, in the form of a full precision timestamp value. **Non-cached dynamic statements** When EXPLAIN was executed, in the form of a value equivalent to a CHAR(16) representation of the time appended by 4 zeros.
SCHEMA	VARCHAR(128) NOT NULL	The schema of the table that contains the column.
TBNAME	VARCHAR(128) NOT NULL	The name of the table that contains the column.
NAME	VARCHAR(128) NOT NULL	Name of the column. If the value of NUMCOLUMNS is greater than 1, this name identifies the first column name of the set of columns associated with the statistics.
COLVALUE	VARCHAR(2000) NOT NULL FOR BIT DATA	Contains the data of a frequently occurring value in the column. Statistics are not collected for an index on a ROWID column. If the value has a non-character data type, the data might not be printable. This column might contain values that depend on the value of the type column: **TYPE='T'** One of the following values: • 'E3C2C1C3C1D9C4C6' for TBACARDF • 'E3C2C1D5C1C3E3C6' for TBANPAGF • 'E3C2C1D5D7C1C7C6' for TBANACTF **TYPE='L'** 'C3C1E3C6D3C4C3C6' for CATFLDCF **TYPE='P'** One of the following values: • 'D7C3C1D7D5D9E6C6' for PCAPNRWF • 'D7C3C1D7D5D7C7C6' for PCAPNPGF

Table 11. Descriptions of columns in DSN_COLDIST_TABLE (continued)

Column name	Data Type	Description
TYPE	CHAR(1) NOT NULL	The type of statistics: **C** Cardinality **F** Frequent value **H** Histogram **T** Real-time table cardinality **L** Real-time column cardinality (unique index only) **P** real-time partition cardinality
CARDF	FLOAT NOT NULL	For TYPE='C', the number of distinct values for the column group. For TYPE='H', the number of distinct values for the column group in a quantile indicated by the value of the QUANTILENO column. For TYPE='T', a value related to real-time statistics table values that are determined by the COLVALUE column. For TYPE= 'L', a value related to a real-time statistics column value that is determined by the COLVALUE column. The QUANTILENO column contains the column number. The NAME column contains the column name. For TYPE='P' a value related to real-time statistics partition value that is determined by the COLVALUE column. The QUANTILENO column contains the partition number.
COLGROUPCOLNO	VARCHAR(254) NOT NULL FOR BIT DATA	The identity of the set of columns associated with the statistics. If the statistics are only associated with a single column, the field contains a zero length. Otherwise, the field is an array of SMALLINT column numbers with a dimension equal to the value in the NUMCOLUMNS column. This is an updatable column.
NUMCOLUMNS	SMALLINT NOT NULL	Identifies the number of columns associated with the statistics.
FREQUENCYF	FLOAT NOT NULL	The percentage of rows in the table with the value that is specified in the COLVALUE column when the number is multiplied by 100. For example, a value of '1' indicates 100%. A value of '.153' indicates 15.3%.
QUANTILENO	SMALLINT NOT NULL	The ordinary sequence number of a quantile in the whole consecutive value range, from low to high. This column is not updatable. For TYPE= 'L', this column contains the column number. For TYPE='P', the column contains the partition number.
LOWVALUE	VARCHAR(2000) NOT NULL FOR BIT DATA	For TYPE='H', this is the lower bound for the quantile indicated by the value of the QUANTILENO column. Not used if the value of the TYPE column is not 'H'. This column is not updatable.

Table 11. Descriptions of columns in DSN_COLDIST_TABLE (continued)

Column name	Data Type	Description
HIGHVALUE	VARCHAR(2000) NOT NULL FOR BIT DATA	For TYPE='H', this is the higher bound for the quantile indicated by the value of the QUANTILENO column. This column is not used if the value of the TYPE column is not 'H'. This column is not updatable.

DSN_DETCOST_TABLE

The following table describes new and changed columns in DSN_DETCOST_TABLE. For the complete set of column descriptions, see DSN_DETCOST_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 12. Descriptions of new and changed columns in DSN_DETCOST_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	CARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
UNCERTAINTY	FLOAT(4) NOT NULL WITH DEFAULT	Describes the uncertainty factor of inner table index access. It is aggregated from uncertainty of inner table probing predicates. A larger value indicates a higher uncertainty. 0 indicates no uncertainty or uncertainty not considered.	New column
TABREF	VARCHAR(64) NOT NULL FOR BIT DATA	IBM internal use only.	Changed to FOR BIT DATA
UNCERTAINTY_1T	FLOAT(4) NOT NULL WITH DEFAULT	Describes the uncertainty factor of ONECOMPROWS column of the table. It is aggregated from all local predicates on the table. A larger value indicates a higher uncertainty. 0 indicates no uncertainty or uncertainty not considered.	New column
SECTNOI	FLOAT(4) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
VERSION	FLOAT(4) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
IMNP	FLOAT(4) NOT NULL WITH DEFAULT	IBM internal use only.	New column

Table 12. Descriptions of new and changed columns in DSN_DETCOST_TABLE (continued)

Column name	Data type	Description	Change Type
DMNP	FLOAT(4) NOT NULL WITH DEFAULT	IBM internal use only.	New column
IMJC	FLOAT(4) NOT NULL WITH DEFAULT	IBM internal use only.	New column
IMFC	FLOAT(4) NOT NULL WITH DEFAULT	IBM internal use only.	New column
IMJBC	FLOAT(4) NOT NULL WITH DEFAULT	IBM internal use only.	New column
IMJFC	FLOAT(4) NOT NULL WITH DEFAULT	IBM internal use only.	New column
CRED	INTEGER	IBM internal use only.	New column

DSN_FILTER_TABLE

The following table describes new and changed columns in DSN_FILTER_TABLE. For the complete set of column descriptions, see DSN_FILTER_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 13. Descriptions of new and changed columns in DSN_FILTER_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	Changed description
COLLID	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
SECTNOI	INTEGER	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122)	See New columns for consistency among EXPLAIN tables.	New column

Table 13. Descriptions of new and changed columns in DSN_FILTER_TABLE (continued)

Column name	Data type	Description		Change Type
PUSHDOWN	CHAR(1)	Whether the predicate is pushed down the Index Manager or Data Manager subcomponents for evaluation:		New column
		'I'	The Index Manager subcomponent evaluates the predicate.	
		'D'	The Data Manager subcomponent evaluates the predicate.	
		blank	The predicate is not pushed down for evaluation.	

DSN_FUNCTION_TABLE

New columns VERSION and SECURE are added to the DSN_FUNCTION_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 14. Descriptions of new and changed columns in DSN_FUNCTION_TABLE

Column name	Data type	Description	Change Type
FUNC_VERSION	VARCHAR(122) NOT NULL WITH DEFAULT	For a version of a non-inline SQL scalar function, this column contains the version identifier. For all other cases, this column contains a zero length string. A version of a non-inline SQL scalar function is defined in SYSIBM. SYSROUTINES with ORIGIN='Q', FUNCTIONTION_TYPE='S', INLINE='N', and VERSION column containing the version identifier.	New column
SECURE	CHAR(1) NOT NULL WITH DEFAULT	Indicates whether the user defined function is secure.	New column
SECTNOI	INTEGER	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122)	See New columns for consistency among EXPLAIN tables.	New column

DSN_KEYTGTDIST_TABLE (new table)

The following table describes the complete set of columns in the new DSN_KEYTGTDIST_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

The following table shows the descriptions of the columns in the DSN_KEYTGTDIST_TABLE table.

Table 15. Descriptions of columns in DSN_KEYTGTDIST_TABLE

Column name	Data Type	Description
QUERYNO	INTEGER NOT NULL	A number that identifies the statement that is being explained. The origin of the value depends on the context of the row: **For rows produced by EXPLAIN statements** The number specified in the QUERYNO clause, which is an optional part of the SELECT, INSERT, UPDATE, MERGE, and DELETE statement syntax. **For rows not produced by EXPLAIN statements** DB2 assigns a number that is based on the line number of the SQL statement in the source program. When the values of QUERYNO are based on the statement number in the source program, values that exceed 32767 are reported as 0. However, in certain rare cases, the value is not guaranteed to be unique. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, if the QUERYNO clause is specified, its value is used by DB2. Otherwise DB2 assigns a number based on the line number of the SQL statement in the compiled SQL function or native SQL procedure.
APPLNAME	VARCHAR(128) NOT NULL	The name of the application plan for the row. Applies only to embedded EXPLAIN statements that are executed from a plan or to statements that are explained when binding a plan. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column is not used, and is blank.
PROGNAME	VARCHAR(128) NOT NULL	The name of the program or package containing the statement being explained. Applies only to embedded EXPLAIN statements and to statements explained as the result of binding a plan or package. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the specific name of the compiled SQL function or native SQL procedure.

Table 15. Descriptions of columns in DSN_KEYTGTDIST_TABLE (continued)

Column name	Data Type	Description
COLLID	VARCHAR(128) NOT NULL	The collection ID: **'DSNDYNAMICSQLCACHE'** The row originates from the dynamic statement cache **'DSNEXPLAINMODEYES'** The row originates from an application that specifies YES for the value of the CURRENT EXPLAIN MODE special register. **'DSNEXPLAINMODEEXPLAIN'** The row originates from an application that specifies EXPLAIN for the value of the CURRENT EXPLAIN MODE special register. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the schema name of the compiled SQL function or native SQL procedure.
GROUP_MEMBER	VARCHAR(128) NOT NULL	The member name of the DB2 that executed EXPLAIN. The column is blank if the DB2 subsystem was not in a data sharing environment when EXPLAIN was executed.
SECTNOI	INTEGER NOT NULL	The section number of the statement. The value is taken from the same column in SYSPACKSTMT or SYSSTMT tables and can be used to join tables to reconstruct the access path for the statement. This column is applicable only for static statements. The default value of -1 indicates EXPLAIN information that was captured in DB2 9 or earlier.
VERSION	VARCHAR(122) NOT NULL	The version identifier for the package. Applies only to an embedded EXPLAIN statement executed from a package or to a statement that is explained when binding a package. A blank indicates that the column is not applicable. When the SQL statement is embedded in a compiled SQL function or native SQL procedure, this column indicates the version identifier of the compiled SQL function or native SQL procedure.

Table 15. Descriptions of columns in DSN_KEYTGTDIST_TABLE (continued)

Column name	Data Type	Description
EXPLAIN_TIME	TIMESTAMP NOT NULL	The time when the EXPLAIN information was captured: **All cached statements** When the statement entered the cache, in the form of a full-precision timestamp value. **Non-cached static statements** When the statement was bound, in the form of a full precision timestamp value. **Non-cached dynamic statements** When EXPLAIN was executed, in the form of a value equivalent to a CHAR(16) representation of the time appended by 4 zeros.
IXSCHEMA	VARCHAR(128) NOT NULL	The qualifier of the index.
IXNAME	VARCHAR(128) NOT NULL	The name of the index.
KEYSEQ	VARCHAR(128) NOT NULL	The numeric position of the key-target in the index.
KEYVALUE	VARCHAR(2000) NOT NULL FOR BIT DATA	Contains the data of a frequently occurring value. Statistics are not collected for an index on a ROWID column. If the value has a non-character data type, the data might not be printable. When the value of the TYPE column contains 'I', this column contains one of the following values: • 'C9C4E7C6E4D3D2C6' for IDXFULKF • 'C9C4E7D3C5C1C6C6' for IDXLEAFF • 'C9C4E7D5D3E5D3C6' for IDXNLVLF
TYPE	CHAR(1) NOT NULL	The type of statistics: **C** Cardinality **F** Frequent value **H** Histogram **I** Real-time index statistics
CARDF	FLOAT NOT NULL	For TYPE='C', the number of distinct values for the column group. For TYPE='H', the number of distinct values for the column group in a quantile indicated by the value of the QUANTILENO column. For TYPE='I', a value related to real-time index statistics values determined by the KEYVALUE column.
KEYGROUPKEYNO	VARCHAR(254) NOT NULL FOR BIT DATA	Contains a value that identifies the set of keys that are associated with the statistics. If the statistics are associated with more than a single key, it contains an array of SMALLINT key numbers with a dimension that is equal to the value in NUMKEYS. If the statistics are only associated with a single key, it contains 0.
NUMKEYS	SMALLINT NOT NULL	The number of keys that are associated with the statistics.

Table 15. Descriptions of columns in DSN_KEYTGTDIST_TABLE (continued)

Column name	Data Type	Description
FREQUENCYF	FLOAT NOT NULL	The percentage of rows in the table with the value that is specified in the COLVALUE column when the number is multiplied by 100. For example, a value of '1' indicates 100%. A value of '.153' indicates 15.3%.
QUANTILENO	SMALLINT NOT NULL	The ordinary sequence number of a quantile in the whole consecutive value range, from low to high. This column is not updatable
LOWVALUE	VARCHAR(2000) NOT NULL FOR BIT DATA	For TYPE='H', this is the lower bound for the quantile indicated by the value of the QUANTILENO column. Not used if the value of the TYPE column is not 'H'. This column is not updatable.
HIGHVALUE	VARCHAR(2000) NOT NULL FOR BIT DATA	For TYPE='H', this is the higher bound for the quantile indicated by the value of the QUANTILENO column. This column is not used if the value of the TYPE column is not 'H'. This column is not updatable.

DSN_PGRANGE_TABLE

The following table describes new and changed columns in DSN_PGRANGE_TABLE. For the complete set of column descriptions, see DSN_PGRANGE_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 16. Descriptions of new and changed columns in DSN_PGRANGE_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
SECTNOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	New column
COLLID	VARCHAR(128) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column

DSN_PGROUP_TABLE

The following table describes new and changed columns in DSN_PGROUP_TABLE. For the complete set of column descriptions, see

DSN_PGROUP_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 17. Descriptions of new and changed columns in DSN_PGROUP_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
COLLID	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
VERSION	VARCHAR(122) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
RANGEKIND	CHAR(1)	The range type: **'K'** Key range **'L'** IN-list elements partitioning **'P'** Page range **'R'** Record range partitioning	Changed description
APPLNAME	VARCHAR(24) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
SECTNOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
STRAW_MODEL	CHAR(1) NOT NULL WITH DEFAULT	IBM internal use only.	New column

DSN_PREDICAT_TABLE

The following table describes new and changed columns in DSN_PREDICAT_TABLE. For the complete set of column descriptions, see DSN_PREDICAT_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 18. Descriptions of new and changed columns in DSN_PREDICAT_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description

Table 18. Descriptions of new and changed columns in DSN_PREDICAT_TABLE (continued)

Column name	Data type	Description	Change Type
CLAUSE	CHAR(8)	The clause where the predicate exists: **'HAVING '** The HAVING clause **'ON '** The ON clause **'WHERE '** The WHERE clause **SELECT** The SELECT clause	Changed description
ORIGIN	CHAR(1)	Indicates the origin of the predicate. **Blank** Generated by DB2 **C** Column mask **R** Row permission **U** Specified by the user	New column
UNCERTAINTY	FLOAT	Describes the uncertainty factor of a predicate's estimated filter factor. A bigger value indicates a higher degree of uncertainty. Value zero indicates no uncertainty or uncertainty not considered.	New column
SECTNOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
COLLID	VARCHAR(128)	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122)	See New columns for consistency among EXPLAIN tables.	New column

DSN_PTASK_TABLE

The following table describes new and changed columns in DSN_PTASK_TABLE. For the complete set of column descriptions, see DSN_PTASK_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 19. Descriptions of new and changed columns in DSN_PTASK_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description

Table 19. Descriptions of new and changed columns in DSN_PTASK_TABLE (continued)

Column name	Data type	Description	Change Type
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
SECTOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
COLLID	VARCHAR(128) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column

DSN_QUERY_TABLE

The following table describes new and changed columns in DSN_QUERY_TABLE. For the complete set of column descriptions, see DSN_QUERY_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 20. Descriptions of new and changed columns in DSN_QUERY_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
SECTNOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	New column
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	New column
COLLID	VARCHAR(128) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column

DSN_SORTKEY_TABLE

The following table describes new and changed columns in DSN_SORTKEY_TABLE. For the complete set of column descriptions, see DSN_SORTKEY_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 21. Descriptions of new and changed columns in DSN_SORTKEY_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
COLLID	VARCHAR(128) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	Changed description
SECTNOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column

DSN_SORT_TABLE

The following table describes new and changed columns in DSN_SORT_TABLE. For the complete set of column descriptions, see DSN_SORT_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 22. Descriptions of new and changed columns in DSN_SORT_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
COLLID	VARCHAR(128) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	Changed description
SECTNOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column

DSN_STATEMENT_CACHE_TABLE

A new column GROUP_MEMBER is added, and new statistics columns are added and enlarged to 64-bit integers.

Table 23. Descriptions of new and changed columns in DSN_STATEMENT_CACHE_TABLE

Column name	Data type	Description	Change Type
GROUP_MEMBER	VARCHAR(24)	The name of the DB2 data-sharing member that inserted this row. This column is null if the subsystem is not configured for data-sharing.	New column
STAT_GPAGB	BIGINT NOT NULL WITH DEFAULT	The number of getpage operations that are performed for the statement.	New column
STAT_SYNRB	BIGINT NOT NULL WITH DEFAULT	The number of synchronous buffer reads that are performed for the statement.	New column
STAT_WRITB	BIGINT NOT NULL WITH DEFAULT	The number of buffer write operations that are performed for the statement.	New column
STAT_EROWB	BIGINT NOT NULL WITH DEFAULT	The number of rows that are examined for the statement.	New column
STAT_PROWB	BIGINT NOT NULL WITH DEFAULT	The number of rows that are processed for the statement.	New column
STAT_SORTB	BIGINT NOT NULL WITH DEFAULT	The number of sorts that are performed for the statement.	New column
LITERAL_REPL	CHAR(1)	Identifies statements where the literals values are replaced by the '&' symbol.	New column
STATUS_SUS_LATCH	FLOAT NOT NULL WITH DEFAULT	The accumulated wait time for latch requests for the statement.	New column
STATUS_SUS_PLATCH	FLOAT NOT NULL WITH DEFAULT	The accumulated wait time for page latch requests for the statement.	New column
STATUS_SUS_DRAIN	FLOAT NOT NULL WITH DEFAULT	The accumulated wait time for drain lock requests for the statement.	New column
STATUS_SUS_CLAIM	FLOAT NOT NULL WITH DEFAULT	The accumulated wait time for claim count requests for the statement.	New column
STATUS_SUS_LOG	FLOAT NOT NULL WITH DEFAULT	The accumulated wait time for log writer requests for the statement.	New column

DSN_STATEMNT_TABLE

The following table describes new and changed columns in DSN_STATEMNT_TABLE. For the complete set of column descriptions, see DSN_STATEMNT_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 24. Descriptions of new and changed columns in DSN_STATEMNT_TABLE

Column name	Data type	Description	Change Type
SECTNOI	INTEGER NOT NULL DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
VERSION	PLAN_TABLE; VARCHAR(122) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column

DSN_STRUCT_TABLE

The following table describes new and changed columns in DSN_STRUCT_TABLE. For the complete set of column descriptions, see DSN_STRUCT_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 25. Descriptions of new and changed columns in DSN_STRUCT_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
APPLNAME	VARCHAR(24) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
ORIGIN	CHAR(1) NOT NULL WITH DEFAULT	Indicates the origin of the query block: **Blank** Generated by DB2 **C** Column mask **R** Row permission **U** Specified by the user	New column
SECTNOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
COLLID	VARCHAR(128) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column
VERSION	VARCHAR(122) NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column

DSN_VIEWREF_TABLE

The following table describes new and changed columns in DSN_VIEWREF_TABLE. For the complete set of column descriptions, see DSN_VIEWREF_TABLE. The format is shown in the sample CREATE TABLE statement in member DSNTESC of the SDSNSAMP library.

Table 26. Descriptions of new and changed columns in DSN_VIEWREF_TABLE

Column name	Data type	Description	Change Type
QUERYNO	INTEGER NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
APPLNAME	VARCHAR(24)	See New columns for consistency among EXPLAIN tables.	Changed description
PROGNAME	VARCHAR(128) NOT NULL	See New columns for consistency among EXPLAIN tables.	Changed description
VERSION	VARCHAR(128)	See New columns for consistency among EXPLAIN tables.	Changed description
COLLID	VARCHAR(128)	See New columns for consistency among EXPLAIN tables.	Changed description
SECTNOI	INTEGER NOT NULL WITH DEFAULT	See New columns for consistency among EXPLAIN tables.	New column

Related concepts:

Investigating SQL performance by using EXPLAIN (DB2 Performance)

Related tasks:

Converting EXPLAIN tables (before migration) (DB2 Installation and Migration)

Creating EXPLAIN tables (DB2 Performance)

Working with and retrieving EXPLAIN table data (DB2 Performance)

Correlating information across EXPLAIN tables (DB2 Performance)

Related reference:

Columns for correlating EXPLAIN tables (DB2 Performance)

EXPLAIN tables (DB2 Performance)

EXPLAIN (DB2 SQL)

New and changed IFCIDs in DB2 10

This release of DB2 for z/OS contains a number of trace enhancements.

PSPI

This information briefly describes the new IFCIDs and the changes to the existing IFCIDs for each new function. For a detailed description of the fields in each IFCID record, refer to the mapping macros data set library DSN1010.SDSNMACS.

Changes to the following IFCIDs might require changes to your trace applications:

- 0002
- 0003
- 0015
- 0016

- 0017
- 0018
- 0027
- 0028
- 0096
- 0217
- 0225
- 0247
- 0248
- 0277
- 0278
- 0306

PSPI

New IFCIDs in DB2 10

DB2 10 introduces new instrumentation facility component identifiers (IFCIDs).

PSPI

Table 27. New IFCIDs

IFCID	Trace	Class	Mapping macro	Description
64-bit support				
0360	Performance	3, 10	DSNDQW04	Records information about queries that are incrementally bound because parallelism was chosen in packages that were created before DB2 10.
Row and column access control				
0270	Audit	10	DSNDQW04	Records information about a row permission or a column mask when it is created, dropped, or altered.
Index I/O parallelism				
0357	Performance	4	DSNDQW04	Records the start of an instance of the use of index I/O parallelism for indexes on a table.
0358	Performance	4	DSNDQW04	Records the end of an instance of the use of index I/O parallelism for indexes on a table.
New DB2 authorities				
0361	Audit	11	DSNDQW05	Records information for auditing administrative authorities.
0362	None	None	DSNDQW05	Records information about the start of an audit trace with AUDITPOLICY.
Parallelism enhancements				
0363	Performance	8	DSNDQW05	Records information about a new workload distribution method for parallel group execution.
Reduce need for frequent explicit REORGs				
0359	Performance	4	DSNDQW04	Records an index page split.
Enhanced monitoring support				

Table 27. New IFCIDs (continued)

IFCID	Trace	Class	Mapping macro	Description
0400	Monitor	29	DSNDQW05	A switch that enables collection of detailed information about static SQL statements that are in the EDM pool.
0401	Monitor	29	DSNDQW05	Records detailed information about static SQL statements that are being tracked in the EDM pool. **Important:** If you are migrating to DB2 10, to retrieve IFCID 0401 information for packages, you need to bind those packages in DB2 10 new-function mode.
0402	Statistics	4	DSNDQW05	Records information about a profile warning or exception condition that is related to system monitoring for threads and connections.
Other IFC enhancements				
0364	Global	3	DSNDQW05	Records address space creation and termination.
0365	Statistics	7	DSNDQW05	Records detailed statistics about the remote locations with which a DB2 subsystem communicates using the DRDA protocol.
0369	Statistics	9	DSNDQW05	Records wait time and CPU time, aggregated by connection type.
0373	None[1]	None	DSNDQW05	Records the location and name of the *dsnhdecp* (application defaults) module that was loaded by the attached subsystem at DB2 startup. This record is available only through an IFI READS call.
0378	Accounting, Monitor	3, 8	DSNDQW05	Records the beginning of an accelerator call event.
0379	Accounting, Monitor	3, 8	DSNDQW05	Records the end of an accelerator call event.
0380	Performance	24	DSNDQW05	Records stored procedure detail information.
0381	None[2]	None	DSNDQW05	Records user-defined function detail information.
0497	None[2]	None	DSNDQW05	Records statement ID information about statements that are not executing in a stored procedure or user-defined function environment.
0498	None[2]	None	DSNDQW05	Records statement ID information about statements that are executing in a user-defined function environment.
0499	Performance	24	DSNDQW05	Records statement ID information about statements that are executing in a stored procedure environment.

Note:

1. These trace records are available only through the IFI interface.
2. You can output these trace records by starting any trace class with the specified IFCID. For example:

```
START TRACE(PERFM) CLASS(32) IFCID(381)
```

PSPI

Changed IFCIDs in DB2 10

This release of DB2 for z/OS introduces changes to a number of trace records.

PSPI

Changes to selected trace records: The following table gives an overview of changes to specific IFCIDs. Changes to IFCID 0106, the system parameters record, are not included.

Table 28. Changed IFCIDs

IFCID	Description of changes
64-bit support	
0015, 0016, 0017, 0018, 0277, 0278	Four-byte fields that contain pointers to storage that is now above the 2 GB bar are no longer used. They are replaced by eight-byte fields.
0027, 0038, 0096	Four-byte fields that contain record counters are no longer used. They are replaced by eight-byte fields.
0225	Fields are added to record more storage statistics.
Access currently committed data	
0002	Fields are added to record the number of rows that were skipped or accessed by read transactions when currently committed read behavior was in effect for fetch operations.
Backup and recovery utility enhancements	
0024	A field is changed to record the SEQCOPY phase of the COPY utility.
Buffer pool enhancements	
0201, 0202	Fields values are added to indicate when the PGSTEAL attribute for a buffer pool is NONE.
Dynamic statement cache literal replacement	
0002, 0361	Fields are added to indicate when cached statements are reused or not reused due to literal replacement.
Eliminate private protocol	
157	This IFCID is removed.
Enhanced monitoring support	
0058	Fields are added to record the statement type, statement execution identifier and statement level performance metrics.
0063, 0350	Fields are added to return the statement type, statement execution identifier and original source CCSID of the SQL statement.
0065	A field is added to indicate whether implicit commit is requested when a cursor is closed.
0066	A field is added to indicate whether a close operation is an explicit or implicit.
0124	Fields are added to return the statement execution identifier and statement type.
0172, 0196, 0337	Fields are added to return the statement execution identifier and statement type when a deadlock or timeout condition is detected.
0316	• When a statement is removed from the dynamic statement cache, an IFCID 0316 record will now be written that describes the statement and statistics about the statement. • Statistics fields are increased from four bytes to eight bytes.
New DB2 authorities	

Table 28. Changed IFCIDs (continued)

IFCID	Description of changes
0140, 0141	Field values are added for the new authorities.
Row and column access control	
0062	Constants are added for new SQL statements and object types.
0140	Constants are added for privileges that are checked for row or column access control.
0145, 0350	A field is added to indicate whether the target table of the SQL statement uses row or column access control.
Temporal tables	
0021, 0044, 0150, 0172	Constants are added for the new index key lock.
Miscellaneous changes	
Trace records written to SMF	If subsystem parameter SMFCOMP is set to ON, everything after the SMF header is compressed. A bit in the SMF header indicates whether a record is compressed.
Correlation header	A new field is added to record the ACE (agent control element) of an accounting record. This field is used to correlate a given IFCID to an accounting interval.
0001	• Descriptions of QDST fields are updated and clarified. • New fields are added to record information about the number of database access threads that are active because they are in packages that are bound with RELEASE(DEALLOCATE). • New fields are added to measure minimum, maximum and average elapsed time spent waiting for a database access thread during the statistics period.
0001, 0002, 0202, 0217, 0225	When traces for these IFCIDs are enabled, the trace records are now written at fixed, one-minute intervals.
0002	• Pairs of fields that record storage use in whole megabytes and fractions of megabytes are replaced with single fields that record storage use in kilobytes. • Additional fields to track EDM pool storage usage are added. • Several counters in macros DSNDQBGL and DSNDQBST are increased from four bytes to eight bytes. • Field QISESPLR has been removed. • IFCID 0002 records can now contain up to 25 QBST repeating groups and 25 QBGL repeating groups. If additional QBST data exists, it is written in an additional IFCID 0002 record that contains *only* QBST repeating groups. If additional QBGL data exists, it is written in an additional IFCID 0002 record that contains *only* QBGL repeating groups. Therefore, the maximum number of IFCID 0002 records for a statistics interval is three. • Distributed data facility statistics records, which are mapped by DSNDQLST, are changed as follows: – Two-byte numeric fields are extended to four bytes. – Fields are reordered for better clarity. – Fields that refer exclusively to DB2 private protocol are removed. – Several pairs of fields contained similar information. For those fields, one field now contains the information from both fields, and the other field has been removed.
0002, 0003	• The number of DESCRIBE statements in field QXDESC now matches the number of DESCRIBE statements in the user application. • Several counters in macro DSNDQXST are increased from four bytes to eight bytes.
0002, 0003, 0148, 0316, 0401	Fields that track the use of RID lists for index access are added or changed to track RID list overflow to a work file.

Table 28. Changed IFCIDs (continued)

IFCID	Description of changes
0003	• Fields are added to describe: – The number of parallel query child agents in a trace roll-up record – Whether a trace roll-up record contains parallel query roll-up data – Information about waits for locks
0003, 0147, 0148	• The following types of data are eligible for DDF and RRSAF rollup: – Distributed accounting data – MVS and DDF (QMDA) accounting information – IFI accounting data
0003, 0147, 0148, 0239, 0346	• Fields are added to track: – The number of wait trace events that are processed for requests to an accelerator – The accumulated wait time for requests to an accelerator – The number of wait trace events that are processed for requests to an accelerator while a package is executing – The accumulated wait time for requests to an accelerator while a package is executing
0003, 0148	• A data section mapped by DSNDQWAR is added for rollup accounting correlation. • Distributed data facility accounting records, which are mapped by DSNDQLAC, are changed as follows: – Two-byte numeric fields are extended to four bytes. – Fields are reordered for better clarity. – Fields that refer exclusively to DB2 private protocol are removed. – Several pairs of fields contained similar information. For those fields, one field now contains the information from both fields, and the other field has been removed.
0021	The field that records lock types is changed to include the utility object lock, whose lock type is X'37'.
0053	You can now use IFCID 0053 to collect all the data that IFCID 0058 provides, without collecting IFCID 0059, 0060, 0061,0064, 0065, or 0066 data. To do that, you start a trace for IFCID 0053 and IFCID 0058, but not for IFCIDs 0059, 0060, 0061,0064, 0065, or 0066.
0058	Fields are added to record: • Accumulated wait time for a lock request • Accumulated wait time for a latch request • Accumulated wait time for a page latch • Accumulated wait time for a claim count • Accumulated wait for log writers • Type of SQL request
0148	Fields are added to record start and end times for the most recent SQL entry for the current nested activity.
0217, 0225	These records have been reorganized for better consumability.
0233	Fields are added to provide the mapping for data section 1 for IFCIDs 0380 and 0381.

Table 28. Changed IFCIDs (continued)

IFCID	Description of changes
0239	The following changes have been made: • Fields are added to record information about waits for locks. • Package detail data is now eligible for parallel query rollup and DDF and RRSAF rollup. • Field QPACSWITCH has been changed so that a return from a nested package no longer counts as a package switch.
0267	The field that records the type of operation that is performed on the underlying structure of a data sharing group is changed to include rebuild of the lock structure due to restart delay.
0306	The record header is increased from 60 bytes to 128 bytes. Records with the 128-byte header have the string 'V10 ' in the fifth through twelfth bytes.
0316	Fields are added to record: • Referenced table name • Transaction name • End-user ID • Workstation name • Accumulated wait time for a latch request • Accumulated wait time for a page latch • Accumulated wait time for a drain lock • Accumulated wait time for a drain during a wait for claims to be released • Accumulated wait for log writers • Timestamp when a statement was inserted into the EDM pool • Timestamp statement statistics were updated READS filters WQALEUID, WQALEUTX, and WQALEUWN now apply to IFCID 0316.
0342	Fields that record space use for tables and indexes in 4KB blocks are changed to record space use in kilobytes.
0343	Fields that record work file use in megabytes are changed to record work file use in kilobytes.

PSPI

Chapter 13. Planning when migrating from DB2 Version 8

In this release, you can migrate a DB2 Version 8 subsystem in new-function mode to DB2 10 without starting the subsystem in DB2 9. Consider the changes that were introduced in DB2 9 when you plan for DB2 for z/OS.

This information summarizes changes that were introduced in DB2 9 for DB2 commands, DB2 utilities, SQL statements, the DB2 catalog, DB2 performance monitoring, and instrumentation facility component identifiers (IFCIDs).

Command changes in DB2 9

This release of DB2 for z/OS includes new and changed commands.

New commands in DB2 9

This release of DB2 for z/OS includes several new commands.

GUPI

Table 29. New commands

Command	Description
ACCESS DATABASE	Forces a physical open of a table space, index space, or partition, or removes the GBP-dependent status for a table space, index space, or partition.
DISPLAY PROFILE	Displays if profiling is active or inactive.
REFRESH DB2, EARLY	Reloads the EARLY code modules, and rebuilds the EARLY control block.
START PROFILE	Loads or reloads the profile table into a data structure in memory.
STOP PROFILE	Stops or disables the profile function.

GUPI

Changes to commands in DB2 9

DB2 9 introduces changes to commands.

The following table shows that several existing commands have new and changed options.

GUPI

Table 30. Changes to existing commands

Command	Description of enhancements and notes
All	To improve auditing and serviceability, all commands are now written to the DB2 log, in the form of IFCID 0090 trace records. Logging of commands is always enabled.

Table 30. Changes to existing commands (continued)

Command	Description of enhancements and notes
-ALTER BUFFERPOOL (DB2)	The ALTER BUFFERPOOL command now changes the values of certain parameters. **New options:** AUTOSIZE(NO) AUTOSIZE(YES) The AUTOSIZE(NO) option is the default and specifies that the buffer pool does not use Workload Manager (WLM) services for automatic buffer pool sizing adjustment. The AUTOSIZE(YES) option specifies that the buffer pool uses WLM services, if available, to automatically adjust the size of the buffer pool based on dynamic monitoring of the workload goals and the available storage on the system.
-ALTER UTILITY (DB2)	The ALTER UTILITY command now changes the values of certain parameters of an execution of the REBUILD utility that uses SHRLEVEL CHANGE. **New and changed options:** REBUILD REORG DELAY (integer) The REBUILD option specifies that a REBUILD SHRLEVEL CHANGE utility is being altered. The REORG option specifies that a REORG SHRLEVEL REFERENCE or REORG SHRLEVEL change utility is being altered. The DELAY (integer) option specifies a lower bound for the interval between the time when the utility sends the LONGLOG message to the console and the time when the utility performs the action specified by the LONGLOG parameter. The integer is the delay in seconds.

Table 30. Changes to existing commands (continued)

Command	Description of enhancements and notes
BIND PLAN (DSN) BIND PACKAGE (DSN) REBIND PLAN (DSN) REBIND PACKAGE (DSN)	**New and changed options:** CURRENTDATA ISOLATION COLLID BIND PACKAGE DEPLOY DBPROTOCOL REOPT(AUTO) ROUNDING SET CURRENT PACKAGESET SET CURRENT PACKAGE PATH For the BIND PLAN and BIND PACKAGE subcommands, the default value for the CURRENTDATA bind option has changed from YES to NO. Also, for the BIND PLAN and the remote BIND PACKAGE subcommands, the default value for the ISOLATION bind option has changed from RR to CS. For the BIND PACKAGE subcommand, the current default (plan value) stays. The default change does not apply to implicitly built CTs (for example, DISTSERV CTs). The COLLID option of REBIND PLAN binds the DBRMs to packages, and binds the packages to the specified plan. COLLID applies only to plans to which DBRMs are bound directly. The BIND PACKAGE DEPLOY bind option deploys a native SQL procedure. The DBPROTOCOL option now has DBPROTOCOL(DRDA) as the default value. DB2 does not support DBPROTOCOL(PRIVATE) when either of the following conditions is true: • The subsystem parameter, PRIVATE_PROTOCOL, is set to NO or AUTH. • The DB2 system is running in Version 9 due to fallback from Version 10 conversion mode. The REOPT option with the keyword AUTO specified autonomically determines if a new access path needs to be generated to further optimize the performance for each execution. The ROUNDING option specifies the rounding mode at bind time. Rounding mode can be used to manipulate DECFLOAT data. You can use the BIND PACKAGE command with the SET CURRENT PACKAGESET option and SET CURRENT PACKAGE PATH option.
-DISPLAY DATABASE (DB2)	In DB2 10, you can use the DISPLAY DATABASE to display the following objects: • XML table spaces • Clone table information The DISPLAY DATABASE command can display information about the status of XML table spaces, which are displayed with a type of 'XS'. The DISPLAY DATABASE command can display information about base table objects and their clones. The information is automatically displayed if a clone table exists.
-DISPLAY DDF (DB2)	The DISPLAY DDF command's output now includes IPv6 addresses that are in colon-hexidecimal format. When IPv4 addresses are used, IPv4 dotted decimal format is the only accepted format.

Table 30. Changes to existing commands (continued)

Command	Description of enhancements and notes
-DISPLAY LOCATION (DB2)	The output for the DISPLAY LOCATION command now includes IPv6 addresses that are in colon-hexidecimal format. When IPv4 addresses are used, IPv4 dotted decimal format is the only accepted format.
-DISPLAY PROCEDURE (DB2)	In DB2 10, you can use the DISPLAY PROCEDURE to display native SQL procedures if you run in DEBUG mode. In Version 9, you need to run the procedure in DEBUG mode if the WLM environment column in the output contains the WLM ENVIRONMENT FOR DEBUG that you specified when you created the native SQL procedure. The DISPLAY PROCEDURE output shows the statistics of native SQL procedures as '0' if the native SQL procedures are under the effect of a STOP PROCEDURE command.
-DISPLAY THREAD (DB2)	**New and changed options:** TYPE (PROC) LIMIT The TYPE keyword now has a PROC option which displays information about threads that are executing stored procedures and user-defined functions. The LIMIT option accepts numeric input that specifies the number of lines of output. Output for this command now includes IPv6 addressing.

Table 30. Changes to existing commands (continued)

Command	Description of enhancements and notes
-DISPLAY TRACE (DB2)	**New options:** ROLE XPLAN XPKGLOC XPKGCOL XPKGPROG XAUTHID XLOC XUSERID XAPPNAME XWRKSTN XCONNID XCORRID XROLE The ROLE option specifies the connection roles. The XPLAN option provides exclude filtering for plan name. The XPKGLOC option provides exclude filtering for package location name. The XPKGCOL option provides exclude filtering for package collection ID. The XPKGPROG option provides exclude filtering for package program name. The XAUTHID option provides exclude filtering for authorization ID. The XLOC option provides exclude filtering for location. The XUSERID option provides exclude filtering for user ID. The XAPPNAME option provides IFC exclude filtering for application or transaction name. The XWRKSTN option provides IFC exclude filtering for workstation name. The XCONNID option provides exclude filtering for connection ID. The XCORRID option provides exclude filtering for correlation ID. The XROLE option provides exclude filtering for connection roles. The DISPLAY TRACE command now allows IP addresses to be specified in the LOCATION KEYWORD and the command accepts new colon-hexidecimal form for IP addresses.
-DISPLAY UTILITY (DB2)	**New output:** Output during UNLOAD phase of REORG with SHRLEVEL CHANGE or SHRLEVEL REFERENCE. Progress of the RECOVER utility. During the UNLOAD phase of REORG with SHRLEVEL CHANGE or SHRLEVEL REFERENCE, DSNU111I messages are issued for the following subtasks: unloading nonpartitioned indexes, building shadow nonpartitioned indexes, sort, build, and inline statistics. The phase for subtasks for unloading nonpartitioned indexes is UNLOADIX. During the LOGAPPLY phase, you can use the DISPLAY UTILITY command to check the progress status of the RECOVER utility.

Table 30. Changes to existing commands (continued)

Command	Description of enhancements and notes
MODIFY irlmproc,SET (z/OS IRLM)	**New and changed options:** DEADLOCK=*nnnn* PVT=*nnnn* DEADLOCK specifies the number, in milliseconds, of how often the local deadlock processing is scheduled. PVT specifies the upper limit of storage that is used for locks. You can specify this value in megabytes or gigabytes by specifying M (for megabytes) or G (for gigabytes) after the value, as follows, *nnnn*M or *nnnn*G.
-RESET INDOUBT DDF (DB2)	When you issue the RESET INDOUBT command, you must specify IP addresses and ports with a double period (..) instead of a colon (:). When you are specifying the IPADDR attribute, you must now also specify an IP address and port.
-START DATABASE (DB2)	In DB2 10, you can use the START DATABASE command on the following objects: • Databases • Table spaces • Index spaces • Physical partitions of partitioned table spaces or index spaces (including index spaces that contain data-partitioned secondary indexes) • Logical partitions of nonpartitioned secondary indexes
-START DB2 (DB2)	**New and changed options:** LIGHT(YES) LIGHT(NOINDOUBTS) LIGHT(YES) specifies that a restart light is to be performed. DB2 starts with reduced storage and terminates normally after freeing retained locks. DB2 now waits for indoubt units of recovery to resolve before it terminates. LIGHT(NOINDOUBTS) specifies that DB2, during a restart light, does not wait for indoubt units of recovery to resolve before terminating.
-START PROCEDURE (DB2)	In DB2 10, the START PROCEDURE command has new affects for native SQL procedures. The START PROCEDURE command affects current version of the native SQL procedures that you specify in the command.
START irlmproc (z/OS IRLM)	**New and changed options:** LTE=*nnnn* MAXCSA= PC= LTE specifies the number of lock table entries that are required in the coupling facility lock structure. MAXCSA is a required positional parameter but is currently unused. PC is a required positional parameter but is currently unused. MAXCSA and PC are currently unused because IRLM Version 2 Release 2 places locks only in private storage.

Table 30. Changes to existing commands (continued)

Command	Description of enhancements and notes
-START TRACE (DB2)	**New options:** ROLE XPLAN XPKGLOC XPKGCOL XPKGPROG XAUTHID XLOC XUSERID XAPPNAME XWRKSTN XCONNID XCORRID XROLE The ROLE option specifies the connection roles. The XPLAN option provides exclude filtering for plan name. The XPKGLOC option provides exclude filtering for package location name. The XPKGCOL option provides exclude filtering for package collection ID. The XPKGPROG option provides exclude filtering for package program name. The XAUTHID option provides exclude filtering for authorization ID. The XLOC option provides exclude filtering for location. The XUSERID option provides exclude filtering for user ID. The XAPPNAME option provides IFC exclude filtering for application or transaction name. The XWRKSTN option provides IFC exclude filtering for workstation name. The XCONNID option provides exclude filtering for connection ID. The XCORRID option provides exclude filtering for correlation ID. The XROLE option provides exclude filtering for connection roles. The START TRACE command now allows IP addresses to be specified in the LOCATION KEYWORD and the command accepts new colon-hexidecimal form for IP addresses.
-STOP PROCEDURE (DB2)	In DB2 10, the STOP PROCEDURE command has new affects for native SQL procedures. The STOP PROCEDURE command affects current version of the native SQL procedures that you specify in the command.

Table 30. Changes to existing commands (continued)

Command	Description of enhancements and notes
-STOP TRACE (DB2)	**New options:** ROLE XPLAN XPKGLOC XPKGCOL XPKGPROG XAUTHID XLOC XUSERID XAPPNAME XWRKSTN XCONNID XCORRID XROLE The ROLE option specifies the connection roles. The XPLAN option provides exclude filtering for plan name. The XPKGLOC option provides exclude filtering for package location name. The XPKGCOL option provides exclude filtering for package collection ID. The XPKGPROG option provides exclude filtering for package program name. The XAUTHID option provides exclude filtering for authorization ID. The XLOC option provides exclude filtering for location. The XUSERID option provides exclude filtering for user ID. The XAPPNAME option provides IFC exclude filtering for application or transaction name. The XWRKSTN option provides IFC exclude filtering for workstation name. The XCONNID option provides exclude filtering for connection ID. The XCORRID option provides exclude filtering for correlation ID. The XROLE option provides exclude filtering for connection roles. The STOP TRACE command now allows IP addresses to be specified in the LOCATION KEYWORD and the command accepts new colon-hexidecimal form for IP addresses.
-STOP DATABASE (DB2)	In DB2 10, you can use the STOP DATABASE command on the following objects: • Databases • Table spaces • Index spaces • Physical partitions of partitioned table spaces or index spaces (including index spaces that contain data–partitioned secondary indexes) • Logical partitions of nonpartitioned secondary indexes

GUPI

Changes to utilities in DB2 9

This release of DB2 for z/OS contains no new utilities.

Utility changes in DB2 9

This release of DB2 for z/OS includes changed utilities.

The following table lists and describes the new and changed options for many existing DB2 for z/OS utilities.

Table 31. New and changed utility options

Utility name	Description of enhancements and notes
BACKUP SYSTEM	**New option:** DUMP, DUMPONLY, DUMPCLASS, FORCE, TOKEN, ESTABLISH FCINCREMENTAL, and END FCINCREMENTAL DUMP, DUMPONLY, DUMPCLASS, FORCE, and TOKEN were added to support fast replication copies of the database copy pool or the log copy pool. ESTABLISH FCINCREMENTAL and END FCINCREMENTAL were added to specify a persistent or last incremental FlashCopy relationship is to be established.
CATMAINT	**New option:** SCHEMA SWITCH, OWNER FROM, and VCAT SWITCH SCHEMA SWITCH updates the owner, creator, or schema name, OWNER FROM changing the ownership of objects from a user to a role, and the VCAT SWITCH changes the catalog name used by the storage groups, user indexes, and table spaces.
CHECK DATA	**New option:** CLONE, LOBERROR, and XMLERROR **Changed option:** SHRLEVEL REFERENCE, SHRLEVEL CHANGE, PUNCHDDN, DRAIN_WAIT, RETRY, and RETRY_DELAY CLONE was added to support checking clone tables. LOBERROR and XMLERROR were added to specify what action to perform if a LOB or XML error is found. SHRLEVEL REFERENCE and SHRLEVEL CHANGE were added to support LOB table spaces. PUNCHDDN was added to specify a DD statement for a data set. DRAIN_WAIT, RETRY, and RETRY_DELAY were added to improve availability.
CHECK INDEX	**New option:** CLONE CLONE was added to check only the specified indexes that are on clone tables.

Table 31. New and changed utility options (continued)

Utility name	Description of enhancements and notes
CHECK LOB	**New option:** CLONE **Changed option:** SHRLEVEL REFERENCE, SHRLEVEL CHANGE, PUNCHDDN, DRAIN_WAIT, RETRY, and RETRY_DELAY CLONE was added to check the LOB table space for only the clone table, not the LOB data for the base table. SHRLEVEL REFERENCE and SHRLEVEL CHANGE were added to support LOB table spaces. PUNCHDDN was added to specify a DD statement for a data set. DRAIN_WAIT, RETRY, and RETRY_DELAY were added to improve availability.
COPY	**New option:** CLONE, SCOPE ALL, and SCOPE PENDING **Changed option:** CHANGELIMIT CLONE was added to copy only clone data in a specified table space or index space. SCOPE ALL copies all of the specified objects and SCOPE PENDING allows you to only copy objects in COPY-pending or informational COPY-pending status. CHANGELIMIT ANY was added to take a full image copy if any pages have changed since the last image copy.
COPYTOCOPY	**New option:** CLONE CLONE was added to support processing only image copy data sets that were taken against clone tables or indexes on clone tables.
DIAGNOSE	**New option:** CLONE CLONE was added to support displaying information for only the specified objects that are clone tables, table spaces that contain clone tables, indexes on clone tables, or index spaces that contain clone tables.
LISTDEF	**New option:** CLONED and XML CLONED indicates that the INCLUDE or EXCLUDE expression is to return only the names of clone tables, table spaces that contain clone tables, indexes on clone tables or index spaces that contain indexes on clone tables. XML was added to specify that only XML objects are to be included in this element of the list.

Table 31. New and changed utility options (continued)

Utility name	Description of enhancements and notes
LOAD	**New options:** BIGINT, BINARY, VARBINARY, DECFLOAT_ROUNDMODE, DECFLOAT, PRESORTED, FORMAT INTERNAL, and INDEXDEFER BIGINT, BINARY, and VARBINARY data types were added to specify a binary string in a column. DECFLOAT_ROUNDMODE was added to support various rounding modes. DECFLOAT was added to support the Decimal Floating Point data type. PRESORTED was added to improve performance of LOAD when the input data is already sorted in clustering key order. FORMAT INTERNAL was added to improve performance of LOAD when the input data is in DB2 internal format. DB2 internal format is the format that is produced when the UNLOAD utility is run with the FORMAT INTERNAL option. INDEXDEFER was added to improve performance of LOAD when the time that is required for the BUILD phase makes performance unacceptable. With INDEXDEFER, you can perform index builds after LOAD completes, by using the REBUILD INDEX utility.
MERGECOPY	**New option:** CLONE CLONE was added to support processing only image copy data sets that were taken against clone objects.
MODIFY RECOVERY	**New option:** CLONE, LAST (*integer*), and LOGLIMIT CLONE was added to support deleting SYSCOPY records and any related SYSLGRNX records for only clone objects. LAST (*integer*) was added to specify the number of recent records to be retained in SYSIBM.SYSCOPY. LOGLIMIT was added to determine the oldest archive log timestamp.
QUIESCE	**New option:** CLONE CLONE was added to support creating a quiesce point for only the specified table spaces that contain clone tables.
REBUILD INDEX	**New option:** CLONE and SHRLEVEL CHANGE CLONE was added to support rebuilding only the specified indexes that are on clone tables. SHRLEVEL CHANGE was added to allow application to read from and write to the index, table space, or partition that is to be checked.

Table 31. New and changed utility options (continued)

Utility name	Description of enhancements and notes
RECOVER	**New option:** CLONE, RESTOREBEFORE *X'byte-string'*, CURRENTCOPYONLY, FROMDUMP, and DUMPCLASS CLONE was added to support recovering only clone table data in the specified table spaces or the specified index spaces that contain indexes on clone tables. RESTOREBEFORE was added to recover an image copy with an RBA or LRSN value earlier than the specified *X'byte-string'* value for use in the RESTORE phase. CURRENTCOPYONLY was added to specify that the primary copy is used for the restore. FROMDUMP and DUMPCLASS were added to support fast replication copies of the database copy pool or the log copy pool.
REORG INDEX	**New option:** CLONE CLONE was added to support reorganizing only the specified index spaces that contain indexes on clone tables.
REORG TABLESPACE	**New options:** CLONE, PARALLEL, SORTNPSI **Changed options:** SHRLEVEL REFERENCE, PART CLONE was added to support reorganizing only clone tables from the specified table spaces. PARALLEL was added to let you choose whether REORG TABLESPACE LIST processes partitions in parallel or serially. SHRLEVEL REFERENCE was updated to support LOB table spaces. REORG was modified to reduce the amount of virtual storage that it uses to build compression dictionaries. PART was enhanced to support multiple parts or part ranges. The SORTNPSI option enables you to specify when to sort all keys of a non-partitioned secondary index.

Table 31. New and changed utility options (continued)

Utility name	Description of enhancements and notes
REPAIR	**New options:** CLONE, VERSIONS, INDEXSPACE, NOAREORENDSTAR, RBDPEND, PSRBDPEND **Changed options:** DBD REBUILD CLONE was added to support processing only the specified table spaces that contain clone tables. The VERSIONS option updates the version information for the named table space or index in the catalog and directory. Use this option when you are moving objects from one system to another or as a part of version number management. The INDEXSPACE option allows you to identify the index by specifying the qualified name of the index space, which you can obtain from the SYSIBM.SYSINDEXES table. You can use the NOAREOPENDSTAR option to reset the advisory REORG-pending (AREO*) status of the specified table space or index. You no longer need to start a database for access by utilities only before REPAIR DBD REBUILD could be performed. DB2 now performs this step for you. RBDPEND option for REPAIR SET INDEX was added to support the REBUILD-pending (RBDP) status. PSRBDPEND option for REPAIR SET INDEX was added to support he PAGE SET REBUILD-pending (PSRBDP) status.
REPORT	**New options:** SHOWDSNS SHOWDSNS was added to include VSAM data set names in the TABLESPACESET report.
RESTORE SYSTEM	**New options:** FROMDUMP, DUMPCLASS, TAPEUNITS, and RSA FROMDUMP and DUMPONLY were added to support fast replication copies of the database copy pool or the log copy pool. TAPEUNITS was added to limit the number of tape drives that the utility should dynamically allocate. RSA was added to specify a key-label in the utility control statement.
RUNSTATS	**New options:** HISTOGRAM and NUMQUANTILES **Changed options:** INDEX LIST HISTOGRAM and NUMQUANTILES were added to collect histogram statistics from columns. Support for the correlation stats-spec keywords was added, when specified along with the RUNSTATS INDEX LIST keywords.

Table 31. New and changed utility options (continued)

Utility name	Description of enhancements and notes
TEMPLATE	**New options:** LIMIT LIMIT added to support template switching.
UNLOAD	**New options:** CLONE, BIGINT, BINARY, VARBINARY, SKIP LOCKED DATA, DECFLOAT_ROUNDMODE, DECFLOAT, XML, and FORMAT INTERNAL **Changed options:** INTEGER EXTERNAL CLONE was added to support unloading data only from clone tables in the specified table spaces. BIGINT, BINARY, and VARBINARY data types were added to specify a binary string in a column. SKIP LOCKED DATA was added to skip rows on which incompatible locks are held by other transactions. DECFLOAT_ROUNDMODE was added to support various rounding modes. DECFLOAT was added to support the Decimal Floating Point data type. XML was added to specify that an XML column is being unloaded directly to the output record. FORMAT INTERNAL was added to allow unloading of data in DB2 internal format. Use of input data in DB2 internal format can improve LOAD performance. For INTEGER EXTERNAL, the greater than and less than values of the original data were changed to have a broader range. Also, the default value has been changed from 11 bytes to 20 bytes.
DSNJU003 (change log inventory)	**New options:** IPV4, IPV6, GRPIPV4, GRPIPV6, NOIPV4, NOIPV6, NGRPIPV4, NGRPIPV6 IPV4, IPV6, GRPIPV4, and GRPIPV6 were added to the DDF statement to identify an IP address. **Changed options:** SYSPITR FFFFFFFFFFFF value added to cause system point-in-time recovery to occur without log truncation.
DSNJU004 (print log map)	**New options:** IPV4, IPV6, GRPIPV4, GRPIPV6, IPV4, IPV6, GRPIPV4, and GRPIPV6 were added to the print log output.
DSN1COMP	**New option:** EXTNDICT EXTNDICT was added to allow the user to save an external copy of the compression dictionary that DSN1COMP builds.

Other utility changes in DB2 9

This release of DB2 for z/OS includes general changes to utilities.

Other changes to utilities are:

- You can use the following utilities to interact with the new partition-by-growth table spaces: COPY, LOAD, REBUILD INDEX, RECOVER, REORG TABLESPACE, and DSN1COPY.
- XML data can now be loaded or unloaded.
- You can use the RUNSTATS utility to scan an expression-based index.
- You cannot run concurrent **REORG TABLESPACE SHRLEVEL CHANGE PART** *integer* on the same table space. Instead of submitting multiple jobs, you can merge the jobs into one job and specify a range using **REORG TABLESPACE SHRLEVEL CHANGE PART** *integer1*:*integer2*, or specify **REORG TABLESPACE SHRLEVEL CHANGE SCOPE PENDING** if multiple partitions are in a REORG-pending state.
- The following utilities support spatial indexes:
  - CHECK INDEX
  - REBUILD INDEX
  - REORG INDEX
  - REORG TABLESPACE

  Zero to 11 index keys can be generated for a single row of data.
- All utility messages that are sent to the SYSPRINT destination include the day number from the Julian date, and a timestamp in the form HH:MM:SS:TT. For example:

```
DSNU586I  -DB2A 002 13:37:05.59 DSNUPSUM - REPORT RECOVERY TABLESPACE DB1.TS1
```

SQL statement changes in DB2 9

This release of DB2 for z/OS provides new and changed SQL statements.

New SQL statements in DB2 9

This release of DB2 for z/OS includes new SQL statements.

GUPI

Table 32. New SQL statements

SQL statement	Description
ALTER PROCEDURE (SQL - native)	Changes the description of or defines additional versions for a native SQL procedure
ALTER TRUSTED CONTEXT	Changes the description of a trusted context
CREATE PROCEDURE (SQL - native)	Defines a native SQL procedure
CREATE ROLE	Defines a role
CREATE TRUSTED CONTEXT	Defines a trusted context
EXCHANGE	Exchanges the data between a base table and the associated clone table
MERGE	Updates and/or inserts one or more rows of a table
RENAME	Renames an existing table or index
SET CURRENT DEBUG MODE	Assigns a value to the CURRENT DEBUG MODE special register

Table 32. New SQL statements (continued)

SQL statement	Description
SET CURRENT DECFLOAT ROUNDING MODE	Assigns a value to the CURRENT DECFLOAT ROUNDING MODE special register
SET CURRENT ROUTINE VERSION	Assigns a value to the CURRENT ROUTINE VERSION special register
TRUNCATE	Deletes all rows from a table

GUPI

Changed SQL statements in DB2 9

In this release of DB2 for z/OS, many existing SQL statements have new and changed clauses.

The following table shows the changes to existing SQL statements.

GUPI

Table 33. Changes to existing SQL statements

SQL statement	Description of enhancements and notes
ALTER FUNCTION (external)	**New clauses:** NO PACKAGE PATH or PACKAGE PATH *package-path*
ALTER INDEX	**New clauses:** REGENERATE
ALTER PROCEDURE (external)	**New clauses:** NO PACKAGE PATH or PACKAGE PATH *package-path*
ALTER STOGROUP	**New clauses:** DATACLAS *dc-name* MGMTCLAS *mc-name* STORCLAS *sc-name*
ALTER TABLE	**New clauses:** FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP HIDDEN ADD CLONE DROP CLONE APPEND NO or APPEND YES RENAME COLUMN *source-column-name* TO *target-column-name* **Changed clauses:** SESSION USER or USER

Table 33. Changes to existing SQL statements (continued)

SQL statement	Description of enhancements and notes
ALTER TABLESPACE	**New clauses:** LOGGED or NOT LOGGED MAXPARTITIONS *integer*
COMMENT	**New clauses:** ACTIVE VERSION or VERSION *routine-version-id* ROLE TRUSTED CONTEXT *context-name*
CREATE DATABASE	**New clauses:** AS WORKFILE ROLE TRUSTED CONTEXT *context-name*
CREATE FUNCTION (external scalar)	**New clauses:** NO PACKAGE PATH or PACKAGE PATH *package-path*
CREATE FUNCTION (external table)	**New clauses:** NO PACKAGE PATH or PACKAGE PATH *package-path*
CREATE INDEX	**New clauses:** RANDOM *key-expression* GENERATE KEY USING XMLPATTERN prolog pattern-expression AS SQL data-type
CREATE PROCEDURE (external)	**New clauses:** ALLOW DEBUG MODE, DISALLOW DEBUG MODE, or DISABLE DEBUG MODE NO PACKAGE PATH or PACKAGE PATH *package-path*
CREATE STOGROUP	**New clauses:** DATACLAS *dc-name* MGMTCLAS *mc-name* STORCLAS *sc-name*

Table 33. Changes to existing SQL statements (continued)

SQL statement	Description of enhancements and notes
CREATE TABLE	**New clauses:** FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP HIDDEN PARTITION BY SIZE APPEND NO or APPEND YES **Changed clauses:** SESSION_USER or USER If a database name or a table space name is not specified for the CREATE TABLE statement, and DB2 is in conversion mode, DB2 will create a segmented table space with a SEGSIZE of 4 and LOCKSIZE ROW. If DB2 is in new-function mode and a database name or a table space name is not specified, DB2 will implicitly create a database or table space.
CREATE TABLESPACE	**New clauses:** LOGGED or NOT LOGGED MAXPARTITIONS *integer* If the SEGSIZE, NUMPARTS, or MAXNUMPARTS clauses are not specified, a segmented table space is created with a default SEGSIZE of 4. If a database name is not specified, a segmented table space with the SEGSIZE of 4 is created in the default database, DSNDB04.
CREATE TRIGGER	**New clauses:** INSTEAD OF ON *view-name*
DECLARE GLOBAL TEMPORARY TABLE	**New clauses:** SESSION_USER or USER
DELETE	**New clauses:** *include-column* SKIP LOCKED DATA
DROP	**New clauses:** ROLE TRUSTED CONTEXT *context-name*
EXECUTE	**New clauses:** *source-row-data*
EXPLAIN	**New clauses:** • STMTS MONITORED • STMTS ALL • STMTS SCOPE

Table 33. Changes to existing SQL statements (continued)

SQL statement	Description of enhancements and notes
FETCH	**New clauses:** WITH CONTINUE CONTINUE
GET DIAGNOSTICS	**New clauses:** DB2_LINE_NUMBER
GRANT	**New clauses:** ROLE *role-name* DEBUGSESSION
INSERT	**New clauses:** *include-column*
PREPARE	**New clauses:** SKIP LOCKED DATA
RENAME	**New clauses:** INDEX
REVOKE	**New clauses:** ROLE *role-name* DEBUGSESSION
SELECT INTO	**New clauses:** SKIP LOCKED DATA
SET PATH	**New clauses:** SESSION_USER or USER
SET SCHEMA	**New clauses:** SESSION_USER or USER
SET *transition-variable* assignment	**New clauses:** DEFAULT
UPDATE	**New clauses:** *include-column* DEFAULT SKIP LOCKED DATA

GUPI

New functions in DB2 9

This release of DB2 for z/OS includes new built-in functions that improve the power of the SQL language.

The following table shows the new built-in functions.

GUPI

Table 34. New functions

Function name	Description
ADMIN_TASK_LIST	Returns a table with one row for each of the tasks that are defined in the task list of the administrative task scheduler.
ADMIN_TASK_STATUS	Returns a table with one row for each task in the task list of the administrative task scheduler that contains the status for the last time the task was run
ASCII_CHR	Returns the character that corresponds to the ASCII code value that is specified by the argument
ASCII_STR	Returns the ASCII version of the character or graphic string argument
COLLATION_KEY	Returns a string that represents the collation key of the argument in the specified collation
COMPARE_DECFLOAT	Returns a SMALLINT value that indicates whether two arguments are equal or unordered, or whether one argument is greater than the other
CORRELATION	Returns the coefficient of the correlation of a set of number pairs
COVARIANCE or COVARIANCE_SAMP	Returns the covariance (population) of a set of number pairs
DECFLOAT	Returns a DECFLOAT representation of its argument
DECFLOAT_SORTKEY	Returns a binary value that can be used when sorting DECFLOAT values
DECRYPT_BINARY	Returns the decrypted value of an encrypted argument
DIFFERENCE	Returns a value that represents the difference between the sound of two strings based on applying the SOUNDEX function to the strings
EBCDIC_CHR	Returns the character that corresponds to the EBCDIC code value that is specified by the argument
EBCDIC_STR	Returns an EBCDIC version of the string argument
EXTRACT	Returns a portion of a date or timestamp based on its arguments
GENERATE_UNIQUE	Returns a character string of bit data that is unique compared to any other execution of the function
GETVARIABLE	Returns a varying-length character string representation of the value of a session variable
LEFT	Returns a string that consists of the specified number of leftmost bytes or the specified string length units
LOCATE_IN_STRING	Returns the position at which the first occurrence of an argument starts within a specified string
LPAD	Returns a string that is padded on the left with blanks or a specified string
NORMALIZE_DECFLOAT	Returns a DECFLOAT value that is the result of normalizing the input argument
NORMALIZE_STRING	Returns a string value that is the result of normalizing the input Unicode value
OVERLAY	Returns a string that is composed of an argument inserted into another argument at the same position where some number of bytes have been deleted

Table 34. New functions (continued)

Function name	Description
QUANTIZE	Returns a DECFLOAT value that is equal in value (except for any rounding) and sign to one argument and which has an exponent set to be equal to the exponent of the second argument
RID	Returns the RID of a row
RIGHT	Returns a string that consists of the specified number of rightmost bytes or specified code units of a string
RPAD	Returns a string that is padded on the right with blanks or a specified string
SOUNDEX	Returns a value that represents the sound of the words in the argument
TIMESTAMPADD	Returns a timestamp derived from adding the specified interval to a timestamp
TIMESTAMP_ISO	Returns a timestamp derived from its arguments
TOTALORDER	Returns a SMALLINT value that indicates the comparison order of two arguments
UNICODE	Returns the Unicode (UTF-16) code value of the leftmost character of the argument as an integer
UNICODE_STR	Returns a string in Unicode (UTF-8 or UTF-16) that represents a Unicode encoding of the argument
VARCHAR_FORMAT	Returns a varying-length character string representation of a timestamp, with the string in a specified format
XMLATTRIBUTES	Returns an XML sequence that contains an XQuery attribute node for each non-null argument
XMLCOMMENT	Returns an XML value with a single comment node from a string expression
XMLDOCUMENT	Returns an XML value with a single document node and zero or more nodes as its children
XMLPARSE	Returns an XML value from parsing the argument as an XML document
XMLPI	Returns an XML value with a single processing instruction node
XMLQUERY	Returns an XML value from the evaluation of an XPath expression against a set of arguments
XMLSERIALIZE	Returns an SQL character string or a BLOB value from an XML value
XMLTEXT	Returns an XML value with a single text node that contains the value of the argument

GUPI

Reserved words

Certain words cannot be used as ordinary identifiers in some contexts because those words might be interpreted as SQL keywords. For example, ALL cannot be a column name in a SELECT statement. Each word, however, can be used as a delimited identifier in contexts where it otherwise cannot be used as an ordinary identifier. For example, if the quotation mark (") is the escape character that begins and ends delimited identifiers, “ALL” can appear as a column name in a SELECT statement.

Certain keywords might be interpreted as ordinary identifiers in some contexts rather than as keywords. For example, in the statement `SELECT * FROM`

SYSIBM.SYSTABLES WHERE, WHERE is interpreted as an ordinary identifier specified as a correlation name, rather than as the beginning of an incomplete WHERE clause.

New reserved words for this version of DB2 for z/OS are identified with notes in this topic. In addition, some topics in this information might indicate words that cannot be used in the specific context that is being described.

IBM SQL has additional reserved words that DB2 for z/OS does not enforce. Therefore, you should not use these additional reserved words as ordinary identifiers in names that have a continuing use. See *IBM DB2 SQL Reference for Cross-Platform Development* for a list of the words.

GUPI

ADD	AND	ASUTIME
AFTER	ANY	AT
ALL	AS	AUDIT
ALLOCATE		AUX
ALLOW		AUXILIARY
ALTER	ASENSITIVE	
	ASSOCIATE	
BEFORE		
BEGIN		
BETWEEN		
BUFFERPOOL		
BY		
CALL	COLLECTION	CONTINUE
CAPTURE	COLLID	CREATE
CASCADED	COLUMN	
CASE	COMMENT	CURRENT
CAST	COMMIT	CURRENT_DATE
CCSID	CONCAT	CURRENT_LC_CTYPE
CHAR	CONDITION	CURRENT_PATH
CHARACTER	CONNECT	CURRENT_SCHEMA
CHECK	CONNECTION	CURRENT_TIME
CLONE	CONSTRAINT	CURRENT_TIMESTAMP
CLOSE	CONTAINS	CURRVAL[1]
CLUSTER	CONTENT	CURSOR
DATA	DELETE	DO
DATABASE	DESCRIPTOR	DOCUMENT
DAY	DETERMINISTIC	DOUBLE
DAYS	DISABLE	DROP
DBINFO	DISALLOW	DSSIZE
DECLARE	DISTINCT	DYNAMIC
DEFAULT		
EDITPROC	ENDING	EXECUTE
ELSE	END-EXEC[2]	EXISTS
ELSEIF	ERASE	EXIT
ENCODING	ESCAPE	EXPLAIN
ENCRYPTION	EXCEPT	EXTERNAL
END	EXCEPTION	

FENCED	FOR	
FETCH	FREE	
FIELDPROC	FROM	
FINAL	FULL	
FIRST[1]	FUNCTION	
GENERATED	GRANT	
GET	GROUP	
GLOBAL		
GO		
GOTO		
HANDLER		
HAVING		
HOLD		
HOUR		
HOURS		
IF	INHERIT	INSERT
IMMEDIATE	INNER	INTERSECT
IN	INOUT	INTO
INCLUSIVE	INSENSITIVE	IS
INDEX		ISOBID
		ITERATE
JAR		
JOIN		
KEEP		
KEY		
LABEL	LIKE	LOCK
LANGUAGE		LOCKMAX
LAST[1]	LOCAL	LOCKSIZE
LC_CTYPE	LOCALE	LONG
LEAVE	LOCATOR	LOOP
LEFT	LOCATORS	
MAINTAINED	MINUTES	
MATERIALIZED	MODIFIES	
MICROSECOND	MONTH	
MICROSECONDS	MONTHS	
MINUTE		
NEXT[1]	NOT	
NEXTVAL	NULL	
NO	NULLS	
NONE	NUMPARTS	
OBID	OPTIMIZE	
OF	OR	
	ORDER	
OLD[1]	ORGANIZATION[1]	
ON	OUT	
OPEN	OUTER	
OPTIMIZATION		
PACKAGE	PATH	PRIOR[1]
PARAMETER	PIECESIZE	PRIQTY
PART	PERIOD[1]	PRIVILEGES
PADDED	PLAN	PROCEDURE
PARTITION	PRECISION	PROGRAM
PARTITIONED	PREPARE	PSID
PARTITIONING	PREVVAL	PUBLIC

QUERY QUERYNO		
READS REFERENCES REFRESH RESIGNAL RELEASE RENAME REPEAT RESTRICT RESULT	RESULT_SET_LOCATOR RETURN RETURNS REVOKE RIGHT ROLE ROLLBACK ROUND_CEILING	ROUND_DOWN ROUND_FLOOR ROUND_HALF_DOWN ROUND_HALF_EVEN ROUND_HALF_UP ROUND_UP ROW ROWSET RUN
SAVEPOINT SCHEMA SCRATCHPAD SECOND SECONDS SECQTY SECURITY SEQUENCE SELECT SENSITIVE SESSION_USER	SET SIGNAL SIMPLE SOME SOURCE SPECIFIC STANDARD STATIC STATEMENT STAY	STOGROUP STORES STYLE SUMMARY SYNONYM SYSDATE[1]SYSTEM SYSTIMESTAMP[1]
TABLE TABLESPACE THEN TO TRIGGER	TRUNCATE TYPE	
UNDO UNION UNIQUE UNTIL UPDATE	USER USING	
VALIDPROC VALUE VALUES VARIABLE VARIANT	VCAT VIEW VOLATILE VOLUMES	
WHEN WHENEVER WHERE WHILE WITH WLM		
XMLEXISTS XMLNAMESPACES XMLCAST		
YEAR YEARS		
ZONE[1]		

Note:

1. New reserved word for DB2 10.
2. COBOL only

GUPI

Other SQL language changes in DB2 9

In addition to the many new SQL statements and functions, this release of DB2 for z/OS includes other enhancements to existing SQL statements.

The following table lists these additional changes to the SQL language.

GUPI

Table 35. Other changes to the SQL language

Item	Description
Expressions	ROW CHANGE *expression* and OLAP specifications are provided as new expressions. A ROW CHANGE expression returns a token or a timestamp that represents the last change to a row. OLAP specifications (RANK, DENSE_RANK, and ROW_NUMBER) provide the ability to return ranking, and row numbering information as a scalar value in the result of a query. RANK, DENSE_RANK, and ROW_NUMBER are sometimes referred to as window functions.
Functions	The following functions now allow a source string that has a data type of CHAR FOR BIT DATA, VARCHAR FOR BIT DATA, or CLOB FOR BIT DATA: • CHAR • INSERT • LEFT • LOCATE • RIGHT • SUBSTRING • VARCHAR
Predicates	XMLEXISTS is provided as a new predicate. XMLEXISTS evaluates an XPath expression and returns a true or false value.
Special registers	This release of DB2 for z/OS introduces several new special registers. The CURRENT DEBUG MODE specifies the default value for the DEBUG MODE option of a CREATE PROCEDURE statement for a native SQL procedure or a Java procedure. The CURRENT DECFLOAT ROUNDING MODE specifies the default rounding mode that is used for DECFLOAT values. The CURRENT ROUTINE VERSION specifies the version identifier that is to be used when invoking a native SQL routine. The ENCRYPTION PASSWORD special register specifies the encryption password and the password hint (if one exists) that are used by the ENCRYPTION and DECRYPTION built-in functions. The SESSION_USER special register replaces the USER special register (USER can be specified as a synonym for SESSION_USER).

Table 35. Other changes to the SQL language (continued)

Item	Description
Session variables	Similar to special registers, session variables are another way to provide information to applications. This release of DB2 for z/OS supports many new DB2-defined session variables that store information that can be referenced by SQL statements. You can use the GETVARIABLE built-in function to retrieve the values of session variables. The new DB2-defined session variables are: • SYSIBM.APPLICATION_ENCODING_SCHEME • SYSIBM.COBOL_STRING_DELIMITER • SYSIBM.DATE_FORMAT • SYSIBM.DATE_LENGTH • SYSIBM.DECIMAL_ARITHMETIC • SYSIBM.DECIMAL_POINT • SYSIBM.DEFAULT_DECFLOAT_ROUND_MODE • SYSIBM.DEFAULT_SSID • SYSIBM.DEFAULT_LANGUAGE • SYSIBM.DEFAULT_LOCALE_LC_CTYPE • SYSIBM.DISTRIBUTED_SQL_STRING_DELIMITER • SYSIBM.DSNHDECP_NAME • SYSIBM.DYNAMIC_RULES • SYSIBM.ENCODING_SCHEME • SYSIBM.MIXED_DATA • SYSIBM.NEWFUN • SYSIBM.PAD_NUL_TERMINATED • SYSIBM.SQL_STRING_DELIMITER • SYSIBM.SSID • SYSIBM.STANDARD_SQL • SYSIBM.TIME_FORMAT • SYSIBM.TIME_LENGTH
Support to change existing columns from LONG VARCHAR and LONG VARGRAPHIC to VARCHAR and VARGRAPHIC	Although LONG VARCHAR and LONG VARGRAPHIC continue to be supported, you should define your columns as VARCHAR or VARGRAPHIC instead. To assist you in changing columns in existing tables from LONG VARCHAR and LONG VARGRAPHIC to VARCHAR and VARGRAPHIC, support is added to ALTER TABLE to change the data types of columns from LONG VARCHAR and LONG VARGRAPHIC to VARCHAR and VARGRAPHIC.

GUPI

Catalog changes in DB2 9

This release of DB2 for z/OS includes changed catalog tables, new catalog tables, and new indexes on catalog tables.

New catalog tables in DB2 9

This release of DB2 for z/OS includes new catalog tables.

GUPI

Table 36. New catalog tables

Catalog table name	Description
SYSIBM.SYSCONTEXT	Contains one row for each trusted context.
SYSIBM.SYSCONTEXTAUTHIDS	Contains one row for each authorization ID with which the trusted context can be used.
SYSIBM.SYSCTXTTRUSTATTRS	Contains one row for each list of attributes for a given trusted context.
SYSIBM.SYSDEPENDENCIES	Records the dependencies between objects.
SYSIBM.SYSENVIRONMENT	Records the environment variables when an object is created.
SYSIBM.SYSINDEXSPACESTATS	Contains real-time statistics for index spaces.
SYSIBM.SYSJAVAPATHS	Contains the complete JAR class resolution path, and records the dependencies that one JAR has on the JARs in its Java path.
SYSIBM.SYSKEYTARGETS	Contains one row for each key-target that is participating in an extended index definition.
SYSIBM.SYSKEYTARGETSTATS	Contains partition statistics for selected key-targets. For each key-target, a row exists for each partition in the table. Rows are inserted when RUNSTATS collects indexed key statistics or non-indexed key statistics for a partitioned table space. No row is inserted if the table space is nonpartitioned.
SYSIBM.SYSKEYTARGETS_HIST	Contains rows from the SYSKEYTARGETS table. Whenever rows are added or changed in SYSKEYTARGETS, the rows are also written to this table.
SYSIBM.SYSKEYTGTDIST	Contains one or more rows for the first key-target of an extended index key.
SYSIBM.SYSKEYTGTDISTSTATS	Contains zero or more rows per partition for the first key-target of a data-partitioned secondary index. Rows are inserted when RUNSTATS scans a data-partitioned secondary index. No row is inserted if the index is a secondary index.
SYSIBM.SYSKEYTGTDIST_HIST	Contains rows from the SYSKEYTGTDIST table. Whenever rows are added or changed in SYSKEYTGTDIST, the rows are also written to this table.
SYSIBM.SYSOBJROLEDEP	Lists the dependent objects for each role.
SYSIBM.SYSROLES	Contains one row for each role.
SYSIBM.SYSROUTINESTEXT	An auxiliary table for the TEXT column of SYSIBM.SYSROUTINES and is required to hold the LOB data.
SYSIBM.SYSTABLESPACESTATS	Contains real-time statistics for table spaces.
SYSIBM.SYSXMLRELS	Contains one row for each XML table that is created for an XML column.
SYSIBM.SYSXMLSTRINGS	Each row contains a single string and its unique ID that are used to condense XML data. The string can be an element name, attribute name, name space prefix, or a namespace URI.

Table 36. New catalog tables (continued)

Catalog table name	Description
SYSIBM.XSRCOMPONENT	Auxiliary table for the BLOB column COMPONENT in SYSIBM.SYSXSROBJECTCOMPONENTS. It is in LOB table space SYSXSRA3.
SYSIBM.XSROBJECTS	Contains one row for each registered XML schema. Rows in this table can only be changed using static SQL statements issued by the DB2-supplied XSR stored procedures.
SYSIBM.XSROBJECTCOMPONENTS	Contains one row for each component (document) in an XML schema. Rows in this table can only be changed using static SQL statements issued by the DB2-supplied XSR stored procedures.
SYSIBM.XSROBJECTGRAMMAR	An auxiliary table for the BLOB column GRAMMAR in SYSIBM.SYSXSROBJECTS. It is in LOB table space SYSXSRA1.
SYSIBM.XSROBJECTHIERARCHIES	Contains one row for each component (document) in an XML schema to record the XML schema document hierarchy relationship. Rows in this table can only be changed using static SQL statements issued by the DB2-supplied XSR stored procedures.
SYSIBM.XSROBJECTPROPERTY	An auxiliary table for the BLOB column PROPERTIES in SYSIBM.SYSXSROBJECTS. It is in LOB table space SYSXSRA2.
SYSIBM.XSRPROPERTY	An auxiliary table for the BLOB column COMPONENT in SYSIBM.SYSXSROBJECTCOMPONENTS. It is in LOB table space SYSXSRA3.

GUPI

Changed catalog tables in DB2 9

Many existing catalog tables are changed in this release of DB2 for z/OS.

The following table shows a list of the new columns and the existing columns that are revised. Revisions to columns include new column descriptions, new values for a column, changed data types, changed column lengths, or both changed data types and lengths.

GUPI

Table 37. Summary of new and revised catalog table columns

Catalog table name	New column	Revised column
IPLIST		IPADDR
IPNAMES		IPADDR USERNAMES SECURITY_OUT
LOCATIONS	TRUSTED SECURE	

Table 37. Summary of new and revised catalog table columns (continued)

Catalog table name	New column	Revised column
SYSAUXRELS	RELCREATED	TBOWNER AUXTBOWNER
SYSCHECKDEP		TBOWNER
SYSCHECKS	RELCREATED	TBOWNER
SYSCHECKS2	RELCREATED	TBOWNER
SYSCOLAUTH	GRANTORTYPE	GRANTEETYPE CREATOR
SYSCOLDIST	QUANTILENO LOWVALUE HIGHVALUE	TYPE CARDF FREQUENCYF TBOWNER
SYSCOLDIST_HIST	QUANTILENO LOWVALUE HIGHVALUE	TYPE CARDF FREQUENCYF TBOWNER
SYSCOLDISTSTATS	QUANTILENO LOWVALUE HIGHVALUE	TYPE CARDF FREQUENCYF TBOWNER
SYSCOLSTATS		TBOWNER HIGHKEY HIGH2KEY LOWKEY LOW2KEY
SYSCOLUMNS	RELCREATED	COLTYPE LENGTH HIGH2KEY LOW2KEY LENGTH2 DEFAULT DEFAULTVALUE TBOWNER
SYSCOLUMNS_HIST		TBCREATOR COLTYPE LENGTH LENGTH2 TBOWNER
SYSCONSTDEP	DTBOWNER OWNERTYPE	DTBCREATOR
SYSCOPY	LOGGED TTYPE INSTANCE RELCREATED	STYPE ICTYPE RELCREATED TIMESTAMP PIT_RBA
SYSDATABASE	IMPLICIT CREATORTYPE RELCREATED	
SYSDATATYPES	OWNERTYPE RELCREATED	CREATEDBY
SYSDBAUTH	GRANTEETYPE GRANTORTYPE	

Table 37. Summary of new and revised catalog table columns (continued)

Catalog table name	New column	Revised column
SYSDBRM	PLCREATORTYPE RELCREATED	
SYSINDEXES	KEYTARGET_COUNT UNIQUE_COUNT IX_EXTENSION_TYPE COMPRESS OWNER OWNERTYPE DATAREPEATFACTORF ENVID	PGSIZE TBCREATOR UNIQUERULE CLUSTERRATIOF CREATOR
SYSINDEXES_HIST	DATAREPEATFACTORF	TBCREATOR CREATOR
SYSINDEXPART		IXCREATOR LEAFDIST FAROFFPOSF NEAROFFPOSF
SYSINDEXPART_HIST		IXCREATOR
SYSINDEXSTATS	DATAREPEATFACTORF	OWNER
SYSINDEXSTATS_HIST	DATAREPEATFACTORF	OWNER
SYSJAROBJECTS	OWNERTYPE	
SYSKEYS		COLSEQ ORDERING
SYSPACKAGE	OWNERTYPE ROUNDING	TYPE REOPTVAR
SYSPACKAUTH	GRANTORTYPE	GRANTEETYPE
SYSPACKDEP	DOWNERTYPE	BTYPE DTYPE BQUALIFIER
SYSPACKSTMT		SEQNO
SYSPARMS	VERSION OWNERTYPE	ROWTYPE PARMNAME CCSID
SYSPLAN	CREATORTYPE ROUNDING	REOPTVAR
SYSPLANAUTH	GRANTEETYPE GRANTORTYPE	
SYSPLANDEP		BTYPE BCREATOR
SYSRELS	RELCREATED	CREATOR REFTBCREATOR IXOWNER
SYSRESAUTH	GRANTEETYPE GRANTORTYPE	
SYSROUTINEAUTH	GRANTORTYPE	GRANTEETYPE

Table 37. Summary of new and revised catalog table columns (continued)

Catalog table name	New column	Revised column
SYSROUTINES	VERSION CONTOKEN ACTIVE DEBUG_MODE TEXT_ENVID TEXT_ROWID TEXT OWNERTYPE PARAMETER_VARCHARFORM RELCREATED PACKAGEPATH	ORIGIN FENCED CREATEDBY WLMENVIRONMENT EXTERNAL_SECURITY
SYSSCHEMAAUTH	GRANTEETYPE GRANTORTYPE	
SYSSEQUENCEAUTH	GRANTORTYPE	GRANTEETYPE
SYSSEQUENCES	OWNERTYPE RELCREATED	SEQTYPE CREATEDBY
SYSSEQUENCESDEP	DOWNER DOWNERTYPE	DTYPE
SYSSTMT	PLCREATORTYPE	
SYSSTOGROUP	CREATORTYPE DATACLAS MGMTCLAS STORCLAS RELCREATED	
SYSSYNONYMS	CREATORTYPE RELCREATED	TBCREATOR
SYSTABAUTH	GRANTORTYPE	GRANTEETYPE SCREATOR TCREATOR
SYSTABCONST	CREATORTYPE RELCREATED	TBCREATOR IXOWNER
SYSTABLEPART	FORMAT REORG_LR_TS RELCREATED	IXCREATOR
SYSTABLES	APPEND OWNER OWNERTYPE RELCREATED	DBID OBID TYPE CREATOR TBCREATOR
SYSTABLES_HIST		CREATOR
SYSTABLESPACE	MAXPARTITIONS CREATORTYPE INSTANCE CLONE RELCREATED	LOG TYPE
SYSTABSTATS		OWNER
SYSTABSTATS_HIST		OWNER

Table 37. Summary of new and revised catalog table columns (continued)

Catalog table name	New column	Revised column
SYSTRIGGERS	OWNERTYPE ENVID RELCREATED	OWNER CREATEDBY TRIGTIME TBOWNER TBNAME
SYSUSERAUTH	GRANTEETYPE GRANTORTYPE DEBUGSESSIONAUTH	
SYSVIEWDEP	DOWNER OWNERTYPE	DCREATOR
SYSVIEWS	OWNER OWNERTYPE RELCREATED	CREATOR
SYSVOLUMES	RELCREATED	
USERNAMES		TYPE

GUPI

New and changed indexes in DB2 9

This release of DB2 for z/OS includes new indexes and changes to existing indexes.

The following table shows the new and changed indexes.

GUPI

Table 38. New and changed indexes

Table space DSNDB06. ...	Catalog table SYSIBM. ...	INDEX SYSIBM. ...	INDEX FIELDS
SYSCONTX	SYSCONTEXT	DSNCTX01	NAME
		DSNCTX02	SYSTEMAUTHID
		DSNCTX03	CONTEXID
		DSNCTX04	DEFAULTROLE
	SYSCONTEXTAUTHIDS	DSNCDX01	CONTEXTID.AUTHID
		DSNCDX02	ROLE
	SYSCTXTTRUSTATTRS	DSNCAX01	CONTEXTID.NAME.VALUE
SYSDBASE	SYSRELS	DSNDLX03	IXOWNER.IXNAME
	SYSTABAUTH	DSNATX01	GRANTOR.GRANTORTYPE
	SYSTABLEPART	DSNDPX04	IXCREATOR.IXNAME
SYSDBAUT	SYSDBAUTH	DSNADH01	GRANTEE.NAME.GRANTEETYPE
		DSNADX01	GRANTOR.NAME.GRANTORTYPE

Table 38. New and changed indexes (continued)

Table space DSNDB06. ...	Catalog table SYSIBM. ...	INDEX SYSIBM. ...	INDEX FIELDS
SYSGPAUT	SYSRESAUTH	DSNAGH01	GRANTEE.QUALIFIER.NAME.OBTYPE. GRANTEETYPE
		DSNAGX01	GRANTOR.QUALIFIER.NAME.OBTYPE. GRANTORTYPE
SYSHIST	SYSKEYTARGETS_HIST	DSNHKX01	IXSCHEMA.IXNAME.KEYSEQ.STATSTIME
	SYSKEYTGTDIST_HIST	DSNTDX02	IXSCHEMA.IXNAME.KEYSEQ.STATSTIME
SYSJAVA	SYSJAVAPATHS	DSNJIPX01	JARSCHEMA.JAR_ID.ORDINAL
		DSNJIPX02	PE_JARSCHEMA.PE_JAR_ID

Table 38. New and changed indexes (continued)

Table space DSNDB06. ...	Catalog table SYSIBM. ...	INDEX SYSIBM. ...	INDEX FIELDS
SYSOBJ	SYSDEPENDENCIES	DSNONX01	BSCHEMA.BNAME.BCOLNAME.BTYPE. DSCHEMA.DNAME.DCOLNAME.DTYPE
		DSNONX02	DSCHEMA.DNAME.DCOLNAME.DTYPE. BSCHEMA.BNAME.BCOLNAME.BTYPE
	SYSENVIRONMENT	DSNOEX01	ENVID
	SYSPARMS	DSNOPX04	SCHEMA.SPECIFICNAME.ROUTINETYPE.VERSION
	SYSROUTINEEAUTH	DSNOAX01	GRANTOR.SCHEMA.SPECIFICNAME. ROUTINETYPE.GRANTEETYPE.EXECUTEAUTH. GRANTORTYPE
	SYSROUTINES	DSNOFX01	NAME.PARM_COUNT. ROUTINETYPE.PARM_SIGNATURE. SCHEMA.PARM1.PARM2.PARM3. PARM4.PARM5.PARM6.PARM7. PARM8.PARM9.PARM10.PARM11. PARM12.PARM13.PARM14.PARM15. PARM16.PARM17.PARM18.PARM19. PARM20.PARM21.PARM22.PARM23. PARM24.PARM25.PARM26.PARM27. PARM28.PARM29.PARM30. VERSION
		DSNOFX02	SCHEMA.SPECIFICNAME. ROUTINETYPE. VERSION
		DSNOFX07	NAME.PARM_COUNT. ROUTINETYPE. SCHEMA. PARM_SIGNATURE. PARM1.PARM2.PARM3. PARM4.PARM5.PARM6.PARM7. PARM8.PARM9.PARM10.PARM11. PARM12.PARM13.PARM14.PARM15. PARM16.PARM17.PARM18.PARM19. PARM20.PARM21.PARM22.PARM23. PARM24.PARM25.PARM26.PARM27. PARM28.PARM29.PARM30. VERSION
	SYSSCHEMAAUTH	DSNSKX01	GRANTEE.SCHEMANAME.GRANTEETYPE
		DSNSKX02	GRANTOR.GRANTORTYPE
SYSPKAGE	SYSPACKAUTH	DSNKAX01	GRANTOR.LOCATION.COLLID.NAME. GRANTORTYPE
		DSNKAX02	GRANTEE.LOCATION.COLLID.NAME.BINDAUTH. COPYAUTH.EXECUTEAUTH.GRANTEETYPE
	SYSPACKSTMT	DSNKSX01	LOCATION.COLLID.NAME.CONTOKEN.STMTNOI. SECTNOI.SEQNO
SYSPLAN	SYSPLANAUTH	DSNAPH01	GRANTEE.NAME.EXECUTEAUTH.GRANTEETYPE
		DSNAPX01	GRANTOR.GRANTORTYPE

Table 38. New and changed indexes (continued)

Table space DSNDB06. ...	Catalog table SYSIBM. ...	INDEX SYSIBM. ...	INDEX FIELDS
SYSROLES	SYSOBJROLESEP	DSNRDX01	DSCHEMA.DNAME.DTYPE.ROLENAME
		DSNRDX02	ROLENAME
	SYSROLES	DSNRLX01	NAMES
SYSRTSTS	SYSTABLESPACESTATS	DSNRTX01	DBID.PSID.PARTITION.INSTANCE
	SYSINDEXSPACESTAT	DSNRTX02	DBID.ISOBID.PARTITION.INSTANCE
SYSSEQ2	SYSSEQUENCEAUTH	DSNWCX02	GRANTOR.SCHEMA.NAME.GRANTORTYPE
		DSNWCX03	GRANTEE.SCHEMA.NAME.GRANTEETYPE
SYSSTATS	SYSKEYTARGETSTATS	DSNTKX01	IXSCHEMA.IXNAME.KEYSEQ.PARTITION
	SYSKEYTGDIST	DSNTDX01	IXSCHEMA.IXNAME.KEYSEQ
	SYSKEYTGTDISTSTATS	DSNTSX01	IXSCHEMA.IXNAME.KEYSEQ.PARTITION
SYSTARG	SYSKEYTARGETS	DSNRKX01	IXSCHEMA.IXNAME.KEYSEQ
		DSNRKX02	DATATYPEID.KEYSPEC_INTERNAL
SYSUSER	SYSUSERAUTH	DSNAUH01	GRANTEE.GRANTEDTS.GRANTEETYPE
		DSNAUX01	GRANTOR.GRANTORTYPE
SYSXML	SYSXMLRELS	DSNXRX01	TBOWNER.TBNAME
		DSNXRX02	XMLTBOWNER.XMLTBNAME
	SYSXMLSTRINGS	DSNXSX01	STRINGID
		DSNXSX02	STRING
SYSXSR	XSROBJECTS	XSROBJ01	XSROBJECTID
		XSROBJ02	XSROBJECTSCHEMA.XSROBJECTNAME
		XSROBJ03	TARGETNAMESPACE.SCHEMALOCATION
		XSROBJ04	SCHEMALOCATION
	XSROBJECT-COMPONENTS	XSRCOMP01	XSRCOMPONENTID
		XSRCOMP02	TARGETNAMESPACE.SCHEMALOCATION
	XSROBJECT-HIERARCHIES	XSRHIER01	XSROBJECTID.TARGETNAMESPACE. SCHEMALOCATION
		XSRHIER02	XSROBJECTID.TARGETNAMESPACE
SYSXSRA1	XSROBJECTGRAMMAR	XSRXOG01	GRAMMAR
SYSXSRA2	XSROBJECTPROPERTY	XSRXOP01	PROPERTIES

Table 38. New and changed indexes (continued)

Table space DSNDB06. ...	Catalog table SYSIBM. ...	INDEX SYSIBM. ...	INDEX FIELDS
SYSXSRA3	XSRCOMPONENT	XSRXCC01	COMPONENT
SYSXSRA4	XSRPROPERTY	XSRXCP01	PROPERTIES

GUPI

Performance monitoring and tuning changes in DB2 9

This release of DB2 for z/OS includes changes that affect performance and changes to monitoring tools, such as EXPLAIN.

Performance changes in DB2 9

This release of DB2 for z/OS includes many improvements to performance. Changes include improvements to parallel processing, new page-range screening methods, a new DB2 system monitor, and changes to how DB2 selects access paths.

Parallelism changes

In previous releases of DB2, if the access path for a query used a data-partitioned secondary index that provided data ordering, DB2 could not use parallelism when it executed the query. In this release of DB2 for z/OS, parallelism is considered when DB2 executes this type of query.

For example, suppose that table T1 has two columns, C1 and C2, and that a data-partitioned secondary index is defined on C1. In previous releases of DB2, no parallelism was considered for the following query:

```
SELECT * FROM TABLE1 ORDER BY C1;
```

Now, DB2 considers parallelism for this query.

Page-range screening for tables with data-partitioned secondary indexes

You can now increase performance by writing queries so that page-range screening will reduce the number of partitions that DB2 accesses. You can take advantage of page-range screening by creating a predicate with literal values on a non-leading column of the partitioning key.

DB2 system monitor

The DB2 system monitor looks for CPU stalls that result in latch contention. When it detects a CPU stall, DB2 system monitor attempts to clear the latch contention by temporarily boosting WLM priority. In addition, the system monitor issues message DSNV508I, to report when DBM1 storage below the 2GB bar reaches critical storage thresholds, and a series of DSNV512I messages to identify the agents that consume the most storage.

Set the xxxxMSTR address space to SYSSTC dispatching priority in WLM in order for the monitor to work effectively.

RUNSTATS and randomized key columns

The values of HIGH2KEY and LOW2KEY columns in the SYSCOLSTATS and SYSCOLUMNS catalog tables are updated with decoded values if the column is a randomized key column.

Improved access path selection for duplicated and reverse-clustered indexes

For indexes that contain either many duplicate key values or key values that are highly clustered in reverse order, cost estimation that is based purely on CLUSTERRATIOF can lead to repetitive index scans. In the worst case, an entire page could be scanned one time for each row in the page. DB2 access path selection can avoid this performance problem by using a new cost estimation formula based on the DATAREPEATFACTORF statistic to choose indexes. Whether DB2 will use this statistic depends on the value of the STATCLUS subsystem parameter. To take advantage of the new formula, set the STATCLUS parameter to ENHANCED. Otherwise, set the value to STANDARD.

EXPLAIN table changes in DB2 9

This release of DB2 for z/OS provides new and changed columns in EXPLAIN tables.

PSPI

Before you can use EXPLAIN, you must create a table called PLAN_TABLE to hold the results of EXPLAIN. If you have an existing PLAN_TABLE from a subsystem that ran on a previous version of DB2 you can alter it to add the new columns.

You can also create additional EXPLAIN tables, such as DSN_STATEMNT_TABLE, DSN_FUNCTION_TABLE, and DSN_STATEMENT_CACHE_TABLE to capture different kinds of information about SQL statements.

You can find sample CREATE TABLE statements for the EXPLAIN tables in member DSNTESC of the SDSNSAMP library.

PSPI

Related concepts:

- Investigating SQL performance by using EXPLAIN (DB2 Performance)
- Interpreting data access by using EXPLAIN (DB2 Performance)

Related reference:

- PLAN_TABLE (DB2 Performance)
- EXPLAIN (DB2 SQL)

Format of PLAN_TABLE in DB2 9

In this release of DB2 for z/OS, PLAN_TABLE contains a new column, and the data types of certain columns have changed. The CCSID of the PLAN_TABLE has also changed to UNICODE. However, many earlier formats remain supported for DB2 10.

PSPI

The 59-column PLAN_TABLE format provides the most information. You can find the CREATE TABLE statement to create the PLAN_TABLE in member DSNTESC of the SDSNSAMP library. However, the following PLAN_TABLE formats with fewer columns are also supported:

DB2 9 format
All columns shown in the sample CREATE_TABLE statement, to and including PARENT_PLANNO (COLCOUNT=59). For information about converting tables in this format to the current format, see Migration step 24: Convert EXPLAIN tables to the current format and encoding type (DB2 Installation and Migration).

DB2 Version 8 format
All columns shown in the sample CREATE_TABLE statement, up to and including STMTTOKEN (COLCOUNT=58). For information about converting tables in this format to the current format, see Converting EXPLAIN tables for migration from DB2 Version 8 (DB2 Installation and Migration).

51-column format
Including the TABLE_TYPE column.

49-column format
Including the PRIMARY_ACCESSTYPE column.

46-column format
Including the BIND_TIME column.

43-column format
Including the IBM_SERVICE_DATA column.

34-column format
Including the PGROUP column

30-column format
Including the COLLID column.

28-column format
Including the MIXOPSEQ column.

25-column format
Including the REMARKS column.

Important: Support for the previous formats is deprecated in DB2 9, and this support is removed from future releases. Therefore, the recommendation is to upgrade your PLAN_TABLE to the DB2 9 format to prepare for future migrations. For more information about preparing to migrate your EXPLAIN tables for future releases, see Converting EXPLAIN tables for migration from DB2 Version 8 (DB2 Installation and Migration).

If you are migrating from the full 58-column DB2 Version 8 PLAN_TABLE format, changes are required only to the following columns (additional changes are required when migrating from earlier formats):

APPLNAME
The data type is changed to VARCHAR(24)

VERSION
The data type is changed to VARCHAR(128)

GROUP_MEMBER
The data type is changed to VARCHAR(24)

PARENT_PLANNO
This column is added. The value of the PARENT_PLANNO column corresponds to the plan number in the parent query block where a correlated subquery is invoked. The data type is SMALLINT.

The descriptions of the following PLAN_TABLE columns have also changed in this release:

- QUERYNO
- ACCESSTYPE
- MATCHCOLS
- ACCESSCREATOR
- ACCCESSNAME
- TSLOCKMODE
- MIXOPSEQ
- JOIN_TYPE
- QBLOCK_TYPE
- PRIMARY_ACCESSTYPE
- TABLE_TYPE

However, the names and data types of these columns remain unchanged

PSPI

Related concepts:

Investigating SQL performance by using EXPLAIN (DB2 Performance)

Interpreting data access by using EXPLAIN (DB2 Performance)

Related reference:

PLAN_TABLE (DB2 Performance)

EXPLAIN (DB2 SQL)

Format of DSN_STATEMNT_TABLE in DB2 9

In this release of DB2 for z/OS, DSN_STATEMNT_TABLE has one new column, TOTAL_COST. This new column provides the overall estimated cost of an SQL statement.

PSPI

You can find the CREATE TABLE statement to create the DSN_STATEMNT_TABLE in member DSNTESC of the SDSNSAMP library.

The following columns are new or changed in the DB2 9 format DSN_STATEMNT_TABLE:

The CCSID of all EXPLAIN tables is changed to UNICODE.

APPLNAME
The data type is changed to VARCHAR(24).

GROUP_MEMBER
The data type is changed to VARCHAR(24).

TOTAL_COST
This new column is added. The value indicates the overall estimated cost of the SQL statement, for reference purposes only. The data type is FLOAT.

The descriptions of the following columns have changed. However, the names and data types remain the unchanged:

- STMT_TYPE
- REASON

PSPI

Related concepts:

Investigating SQL performance by using EXPLAIN (DB2 Performance)

Related tasks:

Estimating the cost of SQL statements (DB2 Performance)

Related reference:

DSN_STATEMNT_TABLE (DB2 Performance)

EXPLAIN (DB2 SQL)

Format of DSN_FUNCTION_TABLE in DB2 9

In this release of DB2 for z/OS, DSN_FUNCTION_TABLE has two columns with changed data types.

PSPI

You can find the CREATE TABLE statement to create the DSN_FUNCTION_TABLE in member DSNTESC of the SDSNSAMP library.

The CCSID of all EXPLAIN tables is changed to UNICODE.

The following columns have changed data types:

APPLNAME
The data type is changed to VARCHAR(24)

GROUP_MEMBER
The data type is changed to VARCHAR(24)

PSPI

Related concepts:

Investigating SQL performance by using EXPLAIN (DB2 Performance)

Related tasks:

Checking how DB2 resolves functions by using DSN_FUNCTION_TABLE (DB2 Application programming and SQL)

Related reference:

DSN_FUNCTION_TABLE (DB2 Performance)

EXPLAIN (DB2 SQL)

New statement cache table in DB2 9

The statement cache table, DSN_STATEMENT_CACHE_TABLE, contains information about the SQL statements that are in the statement cache.

PSPI

This statement cache information is captured when you issue an EXPLAIN STATEMENT CACHE ALL statement.

PSPI

Related tasks:

Capturing performance information for dynamic SQL statements (DB2 Performance)

New and changed IFCIDs in DB2 9

This release of DB2 for z/OS contains a number of trace enhancements.

PSPI

Trace enhancements include:

- More ways to filter trace records

  DB2 adds these new filter types in the constraint block of a START TRACE, STOP TRACE, or DISPLAY TRACE command:
  - PKGPROG: Package
  - PKGLOC: Location
  - PKGCOL: Collection ID
  - USERID: User ID
  - APPNAME: Application name
  - WKRSTN: Workstation name
  - CONNID: Connection ID
  - CORRID: Correlation ID
  - ROLE: Role

  In addition, you can exclude trace records as well as include trace records by any filter type. For example, the following statement excludes trace records for plans named A, B, or C:

  ```
  -START TRACE(ACCTG) XPLAN(A,B,C)
  ```

- The record length of the IFCID flat file (DSNWMSGS) has been increased from 87 to 92, to accommodate larger trace field names.
- As of DB2 UDB for z/OS V8, the IFCID flat file has moved from the *prefix*.SDSNSAMP data set to the *prefix*.SDSNIVPD data set.

This information briefly describes the new IFCIDs and the changes to the existing IFCIDs for each new function. For a detailed description of the fields in each IFCID record, refer to the mapping macros data set library DSN910.SDSNMACS.

PSPI

New IFCIDs in DB2 9

This release of DB2 for z/OS includes new IFCIDs.

PSPI

Table 39. New IFCIDs

IFCID	Trace	Class	Mapping macro	Description
Trusted connections				
0269	AUDIT	10	DSNDQW04	Records information about establishment and reuse of a trusted connection.
Excessive temporary storage usage				
0343	PERFORMANCE	3	DSNDQW04	Records information about an agent when the MAXTEMPS subsystem parameter value is exceeded.
	STATISTICS	4		
READS package detail				
0346			DSNDQW04	Records package detail information.

PSPI

Changed IFCIDs in DB2 9

This release of DB2 for z/OS introduces changes to a number of trace records.

PSPI

Changes to selected trace records: The following table gives an overview of changes to specific IFCIDs. Changes to IFCID 0106, the system parameters record, are not included.

Table 40. Changed IFCIDs

IFCID	Description of changes
Clone table support	

Table 40. Changed IFCIDs (continued)

IFCID	Description of changes
0006, 0007, 0008, 0010, 0011, 0012, 0013, 0015, 0016, 0017, 0019, 0020, 0021, 0023, 0024, 0025, 0044, 0105, 0107, 0108, 0124, 0125, 0127, 0128, 0143, 0144, 0150, 0172, 0196, 0198, 0199, 0211, 0212, 0213, 0215, 0216, 0218, 0221, 0223, 0226, 0227, 0251, 0252, 0255, 0258, 0259, 0263, 0337, 0342	In the page set ID (PSID) field of each of these records, if the high-order bit is on, the instance number that is associated with the object is instance number 2. For PSID fields that are represented as FIXED(15) fields, if the instance number is 2, the PSID value in the trace record is a negative value.
Decimal floating-point data type	
0184, 0247, 0248	The decimal floating-point type is added to data type fields.
FETCH statement with CONTINUE	
0002, 0003, 0148	A counter is added to record the number of FETCH CURRENT CONTINUE statements that have occurred.
0059	A field is added to indicate whether the WITH CONTINUE or CURRENT CONTINUE clause is specified on a FETCH statement.
Global query optimization	
0022	Fields are added and changed to synchronize this trace record with the DB2 10 PLAN_TABLE.
Native SQL procedures	
0003	Plan-level accounting information for SQL procedures includes native SQL procedure data as well as external SQL procedure data.
0148, 0239	The field that specifies the package type is updated to include a package for a native SQL procedure.
Recover to a point in time with consistency	
0023, 0024	The new LOGCSR and LOGUNDO phases or RECOVER are added.
RENAME INDEX statement	
0062	RENAME INDEX is added to the start of execution record.
0140	RENAME INDEX is added to the authorization failure record.
REORG enhancements	
0023, 0024, 0025	• Trace records are written for new subtasks for unloading nonpartitioned indexes. • Due to the elimination of the BUILD2 phase, IFCID 0024 records with the BUILD2 phase are no longer written.
0140	RENAME INDEX is added to the authorization failure record.
Skip locked rows	
0018	A field is added to record the number of rows that were skipped because another transaction had a lock on those rows.
XML support	
0020, 0021, 0044, 0107, 0150, 0172, 0196, 0337.	Fields are added and modified for XML locks.
Miscellaneous changes	
0001, 0225	Fields are added to record stack storage statistics.
0025	Fields are added to record time that is spent processing utilities.

Table 40. Changed IFCIDs (continued)

IFCID	Description of changes
0034, 0035	• These trace records now include active log prefetch read wait time as well as read wait time. IFCID 0034 includes a new flag that indicates the type of wait time. • These trace records are now included in accounting class 3 and 8, and monitor class 3 and 8.
0104	IFCID 0104 records are now available through the IFI READS interface.
0142	IFCID 0142 records now trace ALTER operations, regardless of whether the AUDIT attribute of the table has changed.
0191	• A new section is added. • When a DRDA error occurs that results in reason code 00D3444E or 00D3444F, and a trace for IFCID 0191 is active, IFCID 0274 trace records are also generated.
0199	A field is added to indicate whether the data set on which statistics are reported is a shadow data set.
0201, 0202	Fields are added to record whether automatic buffer pool sizing is in effect.
0217, 0225	Fields are added to record the amount of virtual shared storage that is used by the *ssnm*DBM1 address space.
0225	These trace records are changed from SMF type 102 to SMF type 100 and subtype 4.
0239	A field is added to record the number of times that a package is allocated.

PSPI

Chapter 14. Deprecated function in DB2 10

Certain capabilities that DB2 10 for z/OS supports are *deprecated*, meaning that their use is discouraged. Although they are supported in DB2 10, support is likely to be removed eventually.

Avoid creating new dependencies that rely on deprecated function, and develop plans to remove any dependencies on such function.

Table 41. Deprecated functions in DB2 10

Deprecated function	Recommended alternative	Support removed
6-byte RBA and LRSN format for the BSDS	Starting in DB2 11, convert the BSDS to use the extended 10-byte RBA and LRSN formats. The BSDS conversion must be completed before migration to DB2 12.	DB2 12
ALCUNIT subsystem parameter	Set ALCUNIT to CYL. Later DB2 releases always use the behavior of this setting.	DB2 12
ASSIST subsystem parameter	Set ASSIST to NO. Sysplex query parallelism is removed in later releases. Later DB2 releases always use the behavior of this setting.	DB2 11

Table 41. Deprecated functions in DB2 10 (continued)

Deprecated function	Recommended alternative	Support removed
BIND PACKAGE command options ENABLE and DISABLE (REMOTE) REMOTE (*location-name,...,<luname>,...*)	Starting in DB2 10, you can no longer use the BIND PACKAGE options ENABLE and DISABLE (REMOTE) REMOTE (*location-name,...,<luname>,...*) to enable or disable specific remote connections. If you specify these options in new-function mode, a warning is issued. BIND PACKAGE uses the ENABLE(REMOTE) or DISABLE(REMOTE) option without the location list to enable or disable all remote connections at run time. If you specify these options in DB2 10 conversion mode (from both Version 8 and Version 9), no warning is issued. However, BIND PACKAGE uses the ENABLE(REMOTE) or DISABLE(REMOTE) option without the location list to enable or disable all remote connections at run time. In DB2 10 conversion mode (from both Version 8 and Version 9), the location list is stored in the DB2 catalog for use after fallback. When you run a package that was bound with a location list in a previous release, the location list is not used. The package is validated for all remote connections if ENABLE(REMOTE) is specified, or no remote connections if DISABLE(REMOTE) is specified.	DB2 11
Basic row format table spaces	Use reordered row format. Starting in DB2 12, any table space that uses basic row format is automatically converted to reordered row format when you run the LOAD REPLACE or REORG TABLESPACE utilities.	—
BIND PLAN command MEMBER option	Use BIND PACKAGE commands to bind DBRMs into packages explicitly.	—
CACHEDYN_FREELOCAL subsystem parameter	Set the value to 0. Later DB2 releases always use the behavior of this setting.	DB2 12
CACHE_DEP_TRACK_STOR_LIM subsystem parameter	Starting in DB2 12, storage is allocated from the SCB pool.	DB2 12
COORDNTR subsystem parameter	Set COORDNTR to NO. Sysplex query parallelism is removed in later releases.Later DB2 releases always use the behavior of this setting.	DB2 11

Table 41. Deprecated functions in DB2 10 (continued)

Deprecated function	Recommended alternative	Support removed
COPY utility CHANGELIMIT option	Use the DSNACCOX stored procedure to determine if the object needs to be copied.	—
DDF_COMPATIBILITY subsystem parameter values: • IGNORE_TZ • SP_PARMS_NJV • SP_PARMS_JV		DB2 11
ASSIST subsystem parameter	Set ASSIST to NO. Sysplex query parallelism is removed in later releases. Later DB2 releases always use the behavior of this setting.	DB2 11
CATALOG (in DSN6ARVP) subsystem parameter	Set CATALOG (in DSN6ARVP) to YES. Later DB2 releases always use the behavior of this setting.	DB2 12
CHECK_SETCHKP	Set CHECK_SETCHKP to NO. Later DB2 releases always use the behavior of this setting.	DB2 12
CONTSTOR subsystem parameter	Set CONTSTOR to NO. Later DB2 releases always use the behavior of this setting.	DB2 12
DB2SORT subsystem parameter	Set DB2SORT to ENABLE. Later DB2 releases always use the behavior of this setting.	DB2 12
DISABSCL subsystem parameter	Set DISABSCL to NO. Later DB2 releases always use the behavior of this setting.	DB2 11
DISALLOW_SEL_INTO_UNION subsystem parameter.	Modify applications to remove any use of UNION or UNION ALL as the outermost from-clause of a SELECT INTO statement. Then set DISALLOW_SET_INTO_UNION to YES.	—
DPSEGSZ subsystem parameter.	Set DPSEGSZ to 32. Later DB2 releases always use the behavior of this setting.	DB2 11
SYSPROC.DSNACCOR stored procedure	Use the SYSPROC.DSNACCOX stored procedure.	—
SYSPROC.DSNAEXP stored procedure	Grant the EXPLAIN privilege to users who must capture EXPLAIN information for SQL statements, but cannot have other privileges, such as privileges for executing the SQL statements.	DB2 11
DSNCHKR utility	Beginning in DB2 10 new-function mode. The DSN1CHKR utility is no longer necessary because catalog and directory table spaces do not contain hashes or links.	DB2 11
DSNRLMT and DSNRLST in older formats	Use the latest DB2 10 format resource limit tables.	—

Table 41. Deprecated functions in DB2 10 (continued)

Deprecated function	Recommended alternative	Support removed
EDMPOOL subsystem parameter	Starting in DB2 12, all EDM storage is above the bar.	DB2 12
EXPLAIN tables in DB2 9 or DB2 Version 8 formats	Use the latest DB2 10 format resource limit tables.	DB2 11
SYSPROC.DSNTBIND stored procedure	Use the SYSPROC.ADMIN_COMMAND_DSN stored procedure.	—
SYSPROC.DSNUNTILS stored procedure	Starting in DB2 11, use the SYSPROC.DSNUTILU stored procedure.	—
SYSPROC.DSNWZP	Use the SYSPROC.ADMIN_INFO_SYSPARM stored procedure.	—
INDEX_IO_PARALLELISM subsystem parameter	Set INDEX_IO_PARALLELISM to YES. Later DB2 releases always use the behavior of this setting.	DB2 12
LEMAX subsystem parameter	Starting in DB2 11, the LEMAX value is not used.	DB2 12
LOAD utility INDENTITYOVERRIDE option	Use the OVERRIDE(IDENTITY) option.	—
LOAD utility PERIODOVERRIDE option	Use the OVERRIDE(SYSTEMPERIOD) option.	—
LOAD utility TRANSIDOVERRIDE option	Use the OVERRIDE(TRANSID) option.	—
LOBVALA subsystem parameter	Use the default value. Starting in DB2 12, the database manager automatically determines the amount of storage for processing LOB values.	DB2 12
LOBVALS subsystem parameter	Use the default value. Starting in DB2 12, the database manager automatically determines the amount of storage for processing LOB values.	DB2 12
MATERIALIZE_NODET_SQLTUDF subsystem parameter	Set MATERIALIZE_NODET_SQLTUDF to YES. Later DB2 releases always use the behavior of this setting.	—
MINSTOR subsystem parameter	Set MINSTOR to NO. Later DB2 releases always use the behavior of this setting.	DB2 12
NEWFUN *dsnhdecp* parameter	Starting in DB2 11, use NEWFUN=V*nn* instead of YES or NO values. Starting in DB2 12, use SQLLEVEL.	DB2 12
NEWFUN SQL processing option	Starting in DB2 11, use NEWFUN(*nn*) instead of YES or NO values. Starting in DB2 12, use SQLLEVEL. NEWFUN is ignored if SQLLEVEL is specified.	—

Table 41. Deprecated functions in DB2 10 (continued)

Deprecated function	Recommended alternative	Support removed
ODBC 2.0 function	See Deprecated ODBC functions (DB2 Programming for ODBC).	—
OJPERFEH subsystem parameter	Set OJPERFEH to YES. Later DB2 releases always use the behavior of this setting.	DB2 11
OPTIOWGT subsystem parameter	Set OPTIOWGT to ENABLE. Later DB2 releases always use the behavior of this setting.	DB2 11
OPTIXIO subsystem parameter	Set OPTIXIO to ON. Later DB2 releases always use the behavior of this setting.	DB2 11
Password protection for active log and archive log data sets.		DB2 11
PGRNGSCR subsystem parameter	Set PGRNGSCR to YES. Later DB2 releases always use the behavior of this setting.	DB2 11
PTCDIO subsystem parameter	Set PTCDIO to OFF. Later DB2 releases always use the behavior of this setting.	DB2 11
REORG_IGNORE_FREESPACE subsystem parameter	Set REORG_IGNORE_FREESPACE to NO. Later DB2 releases always use the behavior of this setting.	DB2 12
REORG INDEX utility LEFDISTLIMIT and REPORT only options	Use the DSNACCOX stored procedure to determine whether the object needs to be reorganized	—
REORG INDEX utility UNLOAD ONLY option	Use the UNLOAD utility.	—
REORG INDEX utility UNLOAD PAUSE option	Use the DIAGNOSE utility to stop the process.	—
REORG TABLESPACE utility UNLOAD EXTERNAL option	Use the UNLOAD utility.	—
REORG TABLESPACE utility INDREFLIMIT and REPORTONLY options	Use the DSNACCOX stored procedure to determine whether the object needs to be reorganized.	—
REORG TABLESPACE utility OFFPOSLIMIT and REPORTONLY options	Use the DSNACCOX stored procedure to determine whether the object needs to be reorganized.	—
REORG TABLESPACE utility PARALLEL option	Starting in DB2 11, use the LISTPARTS option instead.	—
REORG TABLESPACE utility UNLOAD ONLY option	Use the UNLOAD utility.	—
REORG TABLESPACE utility UNLOAD PAUSE option	Use the UNLOAD utility FORMAT INTERNAL option.	—
REPAIR VERSIONS utility	Use the REPAIR CATALOG utility.	—
RETVLCFK subsystem parameter	Set RETVLCFK to NO. Later DB2 releases always use the behavior of this setting.	DB2 11

Table 41. Deprecated functions in DB2 10 (continued)

Deprecated function	Recommended alternative	Support removed
RRF subsystem parameter	Set RRF to YES. Later DB2 releases always use the behavior of this setting.	DB2 12
RETVLCFK subsystem parameter	Set RETVLCFK to NO and use non-padded indexes. Later DB2 releases always use the behavior of this setting.	DB2 11
SEQCACH subsystem parameter	Set SEQCACH to NO. Later DB2 releases always use the behavior of this setting.	DB2 11
SEQPRES subsystem parameter	Set SEQPRES to YES. Later DB2 releases always use the behavior of this setting.	DB2 11
SMSDCFL subsystem parameter	Use the SMS data class parameter with CREATE STOGROUP and ALTER STOGROUP statements instead.	DB2 11
SMSDCIX subsystem parameter	Use the SMS data class parameter with CREATE STOGROUP and ALTER STOGROUP statements instead.	DB2 11
DB2XML.SOAPHTTPC supplied user-defined function	Use the DB2XML.SOAPHTTPNC user-defined function.	—
DB2XML.SOAPHTTPV supplied user-defined function	Use the DB2XML.SOAPHTTPNV user-defined function.	—
SQWIDSC subsystem parameter	Set SQWIDSC to YES. Later DB2 releases always use the behavior of this setting.	DB2 12
Synonyms	Use aliases when writing new SQL statements or creating portable applications. Aliases behave the same for the DB2 family of products.	—
Sysplex query parallelism	DB2 11 and later use query CP parallelism for packages that are bound with sysplex parallelism enabled.	DB2 11
Table spaces of types other than universal table spaces, including simple, partitioned (non-universal), and segmented (non-universal) table spaces	Use universal table spaces. DB2 supports existing simple table spaces. However, you cannot create new simple table spaces.	—
TEMP database	DB2 uses the work file database instead.	—
UTSORTAL subsystem parameter	Set UTSORTAL to YES. Later DB2 releases always use the behavior of this setting.	DB2 12
XMLVALA subsystem parameter	Use the default value. In DB2 12, the database manager automatically determines the amount of storage for processing XML values.	DB2 12

Table 41. Deprecated functions in DB2 10 (continued)

Deprecated function	Recommended alternative	Support removed
XMLVALS subsystem parameter	Use the default value. In DB2 12, the database manager automatically determines the amount of storage for processing XML values.	DB2 12

Related concepts:

Changes that might affect your migration from DB2 9 (DB2 Installation and Migration)

Changes that might affect your migration from Version 8 (DB2 Installation and Migration)

Related tasks:

Preparing your system to install or migrate to DB2 10 (DB2 Installation and Migration)

Information resources for DB2 10 for z/OS and related products

Information about DB2 10 for z/OS and products that you might use in conjunction with DB2 10 is available online in IBM Knowledge Center or on library websites.

Obtaining DB2 for z/OS publications

Current DB2 10 for z/OS publications are available from the following websites:

http://www-01.ibm.com/support/docview.wss?uid=swg27019288

Links to IBM Knowledge Center and the PDF version of each publication are provided.

DB2 for z/OS publications are also available for download from the IBM Publications Center (http://www.ibm.com/shop/publications/order).

In addition, books for DB2 for z/OS are available on a CD-ROM that is included with your product shipment:

- DB2 10 for z/OS Licensed Library Collection, LK5T-7390, in English. The CD-ROM contains the collection of books for DB2 10 for z/OS in PDF format. Periodically, IBM refreshes the books on subsequent editions of this CD-ROM.

Installable information center

You can download or order an installable version of the Information Management Software for z/OS Solutions Information Center, which includes information about DB2 10 for z/OS, QMF, IMS, and many DB2 Tools for z/OS products. You can install this information center on a local system or on an intranet server. For more information, see http://www-01.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.db2z11.doc/src/alltoc/installabledzic.html.

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 give 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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

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

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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.

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

Programming interface information

This information is intended to help you to learn about and plan to use DB2 10 for z/OS. This information also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by DB2 10 for z/OS.

General-use Programming Interface and Associated Guidance Information

General-use Programming Interfaces allow the customer to write programs that obtain the services of DB2 10 for z/OS.

General-use Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:

GUPI

General-use Programming Interface and Associated Guidance Information...

GUPI

Product-sensitive Programming Interface and Associated Guidance Information

Product-sensitive Programming Interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive Programming Interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:

Product-sensitive Programming Interface and Associated Guidance Information...

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered marks of International Business Machines Corp., 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.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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

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

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions:

Applicability: These terms and conditions are in addition to any terms of use for the IBM website.

Personal use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights: Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Glossary

The glossary is available in IBM Knowledge Center.

See the Glossary topic for definitions of DB2 for z/OS terms.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Product Number: 5605-DB2
5697-P31

Printed in USA

GC19-2985-13

Spine information:

DB2 10 for z/OS

What's New?