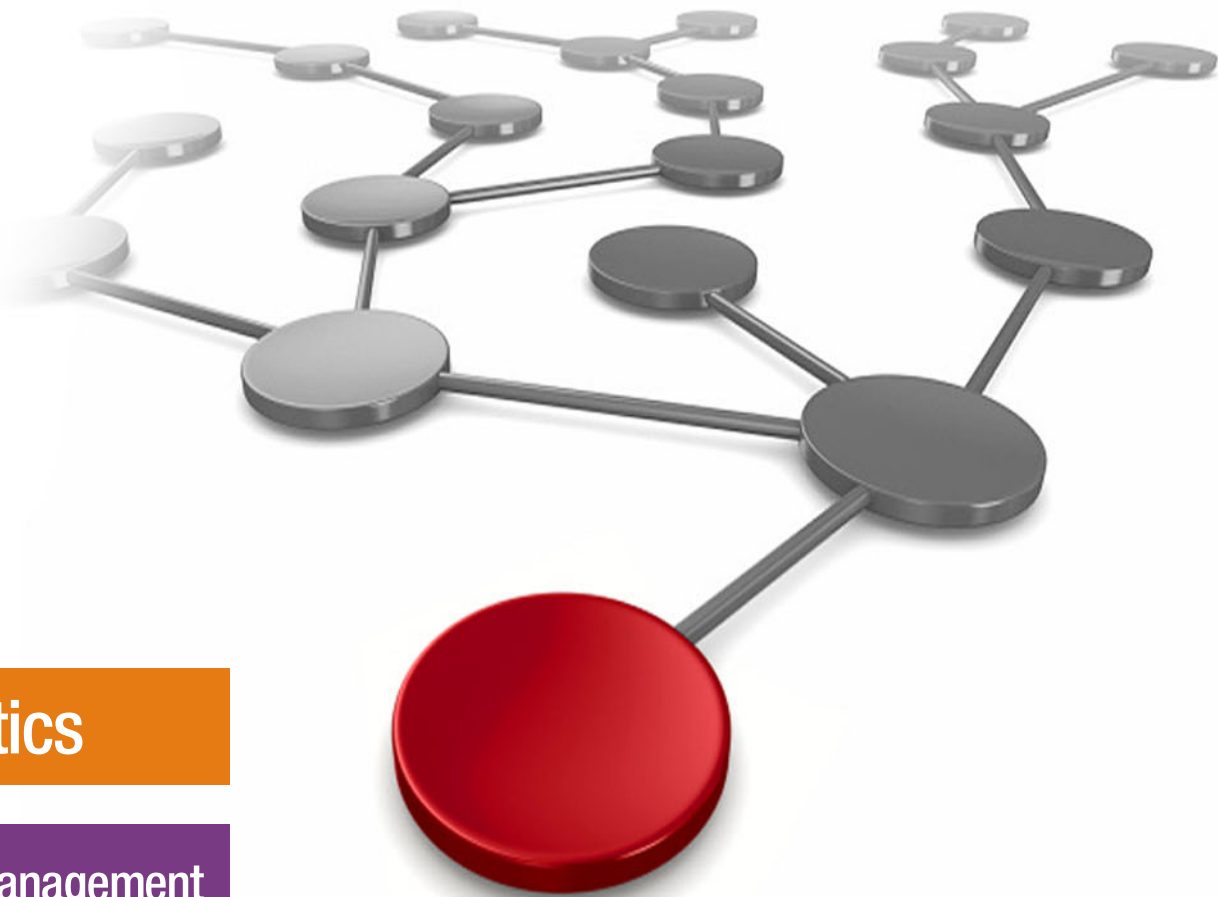


DB2 12 for z/OS Technical Overview and Highlights

Gareth Jones
John Campbell



 **Analytics**

Information Management



Introduction

This paper provides a high-level overview of the functions, features, and improvements that were introduced in IBM DB2® 12 for z/OS®. These businesses gain a competitive advantage, improve business efficiency and effectiveness, and discover business insights from their existing data.

The paper is intended for DB2 system administrators, database administrators, and application programmers, but might also be of interest to IT managers and executives. A more detailed description of DB2 12 is available in *IBM DB2 12 for z/OS Technical Overview*, SG24-8383.

The purpose of this paper is to provide the reader with an understanding of some of the key changes that were introduced in DB2 12 for z/OS, including the following topics:

- ▶ Performance for traditional workloads
- ▶ Performance enablers for modern applications
- ▶ Application Enablement
- ▶ Reliability, availability, and scalability
- ▶ Security
- ▶ Migration and prerequisites

Before looking at those areas in detail, it is important to put the benefits DB2 12 delivers in the context in terms of four themes for this release:

- ▶ Application enablement
- ▶ DBA productivity
- ▶ OLTP performance
- ▶ Query performance

Application enablement

Each release of DB2 has included several application enablement features, and DB2 12 is no exception. It addresses a number of key customer requirements:

- ▶ Expand the use of the existing features of DB2
- ▶ Make DB2 more available to modern applications by delivering mobile, hybrid cloud and DevOps enablement
- ▶ Enhance IDAA functionality by supporting a wider range of use cases
- ▶ Provide incremental improvements in the SQL and SQL/PL areas

DBA productivity

Even with the existing capability to grow partitioned table spaces to 128 TB, some customers have been constrained by table and partition scalability limits. Therefore, these limits are raised significantly at both the table space and partition level. In addition, large table management is simplified, with some of the biggest inhibitors to 24x7 continuous availability being removed.

OLTP performance

OLTP performance is still vitally important for DB2 customers, and DB2 12 continues the work done in DB2 10 and DB2 11 to deliver even more performance improvements. In-memory features deliver reduced CPU consumption in the range of 5 - 10% for targeted queries. Customers can expect to see a significant increase in throughput for concurrent transactions that insert rows at high speed into a non-clustered table. This improvement is up to double the throughput in the most favorable cases. Some of the system scaling bottlenecks that are associated with high n-way systems are removed. Data sharing groups can now scale vertically as well as horizontally, using large LPARs to drive more work through each member.

Query performance

Performance for OLAP, BI, and other workloads with complex SQL queries is another area where DB2 12 builds on the improvements that were delivered by DB2 11:

- ▶ A 20 - 30% CPU reduction for complex query workloads
- ▶ Improved efficiency delivered by reducing non-CPU resource consumption
- ▶ Improved performance for UNION ALL, with a 10-30% CPU reduction and a 60-80% elapsed reduction
- ▶ Simplified access path management, especially for dynamic SQL

Performance for traditional workloads: In-Memory Computing

DB2 12 places a strong emphasis on *In-Memory Computing*, combining large real memory with memory-optimized data structures to drive performance improvements. Unlike in prior releases, where some of the performance improvements were available without necessarily making use of more real memory, many of the DB2 12 enhancements require customers to provision more real memory for DB2 to use before they can realize significant performance gains. This is true of all the enhancements discussed here. As a general comment on real memory provisioning, plan to avoid all paging and make sure that you have sufficient free real memory to run safely.

In-Memory contiguous buffer pools

One of the features that falls into this category is the in-memory contiguous buffer pool. The objective of this buffer pool option is to cache entire table spaces or index spaces in the buffer pool. The larger the object, the larger the buffer pool, and the larger the buffer pool, the more real memory is required.

The in-memory contiguous buffer pool improves performance and reduces CPU consumption by providing direct page access in memory. DB2 development has measured up to an 8% CPU reduction for OLTP workloads using this feature. Direct page access greatly reduces the

CPU overhead of Get Page and Release Page operations, and is achieved by laying out objects contiguously in page order in the buffer pool, and then accessing the page directly in memory.

This is not the first step taken by DB2 to support in-memory buffer pools. DB2 10 introduced the PGSTEAL(NONE) buffer pool attribute for objects such as table spaces and indexes that can fit in their entirety into a buffer pool. The pages for each object are prefetched into the buffer pool when that object is first accessed, saving subsequent I/O, saving CPU, and providing elapsed time benefits. In DB2 10 and 11, after the initial read of data into a PGSTEAL(NONE) buffer pool, prefetch is disabled, saving CPU cycles. However, DB2 still maintains hash chains and LRU chains, the CPU cost of which is still measurable for getpage-intensive workloads that use large PGSTEAL(NONE) buffer pools.

DB2 12 changes PGSTEAL(NONE) behavior to avoid the LRU and hash chain management overheads. And to make this feature more resilient, DB2 12 introduces an overflow area that is automatically managed by DB2. An overflow area is reserved by DB2 from the buffer pool allocation, and takes up 10% of the buffer pool size. DB2 only uses the overflow area if the objects assigned to the buffer pool do not fit into the remaining 90% of the buffer pool size. It is allocated when the buffer pool is allocated, but is only backed by real storage when it is used. Any pages in the overflow area are automatically managed by DB2 on an LRU basis. Although no page stealing occurs within the main buffer pool area, it is possible in the overflow area. The lesson here is to ensure that any objects that are assigned to the buffer pool fit into 90% of the VPSIZE.

In-memory index optimization

As customer tables grow larger and larger, index sizes inevitably grow too. The larger the index in terms of the number of levels, the greater the cost of random index access becomes.

Before DB2 12, DB2 used index lookaside to try to avoid the full cost of index probing. However, this typically only benefitted skip sequential access through the index. Random SQL only benefitted from index lookaside occasionally, often requiring a getpage for each level of the index.

To make index lookups faster and cheaper, DB2 12 introduces the Index Fast Traverse Block (FTB) in-memory area to provide fast index lookups for random index access for unique indexes with a key size of 64 bytes or less. It is a memory optimized structure, and contains the non-leaf pages of the index. The FTB area resides outside of the buffer pool, in a memory area managed by DB2, and requires additional real memory. Unique indexes with include columns are also supported if the total length of all the columns that are specified in the index is 64 bytes or less.

A new ZPARM, INDEX_MEMORY_CONTROL, is used to control the size of this memory area, with a minimum size of either 500 MB or 20% of total allocated buffer pool storage, whichever is larger, and a maximum size of 200 GB. If you want to control the introduction of the FTB area at a system-wide level, you can use the ZPARM to disable fast index traversal. However, there is also a new catalog table, SYSIBM.SYSINDEXCONTROL, which is a mechanism for controlling fast index traversal at the index level. You can specify when an index should be considered for FTB exploitation by time window, or you can disable it altogether. By default, all qualifying indexes are eligible for FTB exploitation.

DB2 automatically determines over time which unique indexes would benefit from the FTB area. Fast index traversal is not attractive for indexes that suffer from frequent leaf page splits, due to inserts and updates for example, but when the index is used predominantly for read access by way of key lookups, a significant performance improvement is expected. SELECT, INSERT, UPDATE, and DELETE can all benefit from FTB area because they can avoid

traversing the index b-tree. The savings can be substantial depending on the number of levels in the index b-tree.

A new zIIP-eligible Index Memory Optimization Daemon monitors index usage, and allocates FTB storage to indexes that will benefit. Using the FTB, DB2 can do very fast traversals through the non-leaf page index levels without having to do page-oriented access.

Customers can monitor FTB area usage by using the new DISPLAY STATS command, which shows which indexes are using the FTB area. Customers by IBM service can also use two new IFCIDs (389 and 477) that allow the tracking of FTB area usage at a detailed level.

Figure 1 shows, based on measurements taken by DB2 development, that simple random index lookup is faster and cheaper in DB2 12. It also shows that the greater the number of index levels, the greater the expected CPU savings, varying from 6% for a two-level index to 23% for a five-level index. The expected CPU savings increase with the number of index levels. Index only access shows higher percentage CPU savings.

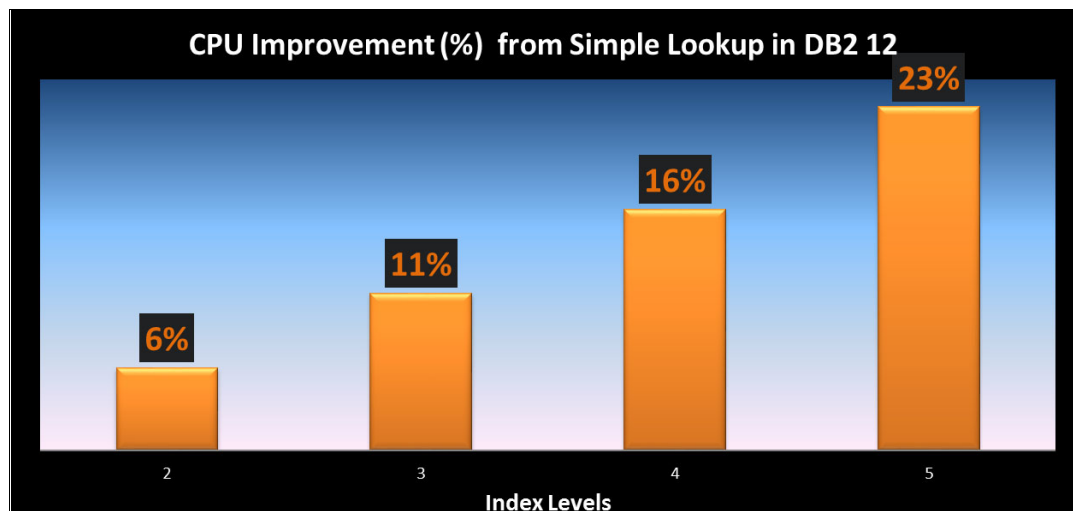


Figure 1 CPU improvement using fast index traversal

INSERT performance

INSERT workloads are very common across the DB2 for z/OS customer base, and are often performance critical. A typical use case is a journal or audit table with a high rate of concurrent insert, where rows are inserted at the end of the table space of partition, and where data clustering is not required. It is not always necessary for the rows to be inserted in the order of the clustering index. Some customers try to balance the need for fast insert processing with the need to be able to query the data rows in key order by processing the inserted rows again later to populate other tables with the rows now in clustering key order.

However, performance has historically been a challenge for customers who need to insert very large volumes of data rows into the database at very high speed. This performance issue was addressed in prior releases of DB2 by forcing the insert of new rows at the end of the current partition for table spaces/tables with the MEMBER CLUSTER and APPEND attributes, without regard to data row clustering. Before DB2 12, the space search algorithm used for the table space or partition was often a bottleneck for insert performance. However, this bottleneck is addressed in DB2 12.

However, some customers need even more improvement in insert throughput performance, and DB2 12 takes a significant step forward with a new INSERT algorithm that streamlines the free space search operation. This feature is targeted specifically at UTS table spaces with the MEMBER CLUSTER attribute. The use cases that benefit from this change are broadened because DB2 does not consider the APPEND table attribute when determining which tables qualify for the new algorithm. The old insert algorithm is known as Algorithm 1, and the new, faster insert algorithm is known as Algorithm 2.

However, this feature is not available until a new function has been enabled in DB2 12: After new function activation (ANFA). The new insert algorithm depends on new log records introduced by the activation of new function.

Therefore, the new insert algorithm has these requirements:

- ▶ New function has been activated (ANFA)
- ▶ The UTS table space type is used
- ▶ The table space is defined with Member Cluster (MC)

To allow customers to control usage of the new algorithm, it can be turned off by using the new ZPARM DEFAULT_INSERT_ALGORITHM, or at the table space level by using the DDL attribute INSERT_ALGORITHM. You can specify '1' for the old algorithm, or '2' for the new algorithm. You only need to specify it at the table space level if you want to override the ZPARM setting. The default ZPARM setting after new function has been activated is '2.'

If you want to use this new feature, plan for extra real memory and larger size buffer pools for each qualifying table space partition and for each DB2 member. There is an additional real memory requirement per partition per member.

Figure 2 on page 6 shows faster insert rates into a group buffer pool-dependent partition by range (PBR) universal table space (UTS) defined with MEMBER CLUSTER and APPEND in two-way data sharing in DB2 12 AFNA. This is a special use case because the table has no indexes. But it does demonstrate the benefits of faster insert into a table space without any of the side effects introduced by indexes. The workload consists of a number of short, fast insert processes running in parallel, each inserting a small number of rows. There are three things to note with this chart:

- ▶ The number of inserts per second increase from under 800,000 per second to over 1,000,000 per second, which is a 25% improvement
- ▶ The class 2 elapsed time per transaction with Algorithm 1 at 0.012 seconds per transaction is reduced dramatically to 0.002 seconds per transaction
- ▶ The class 2 CPU time per transaction is reduced by about 20%

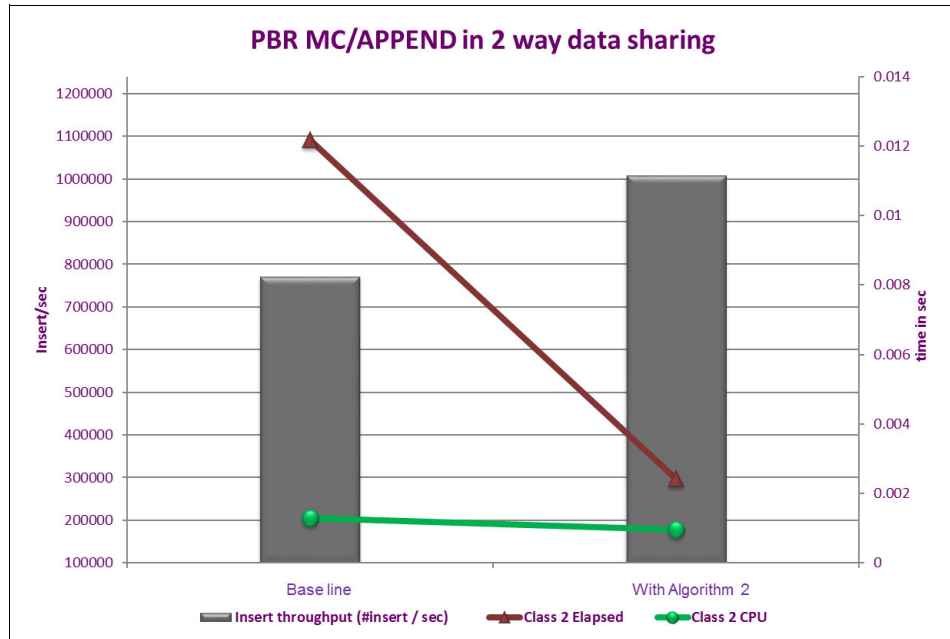


Figure 2 Response time and CPU time improvements with new insert algorithm

A more common scenario is outlined in Figure 3, which is based on a table from an application journal. As in the previous example, the experiments were performed in a two-way data sharing environment, with group buffer pool dependency. This is one of the use cases making up the high-insert performance workload test suite run by DB2 development on a continuous basis at the IBM Silicon Valley Laboratory (SVL).

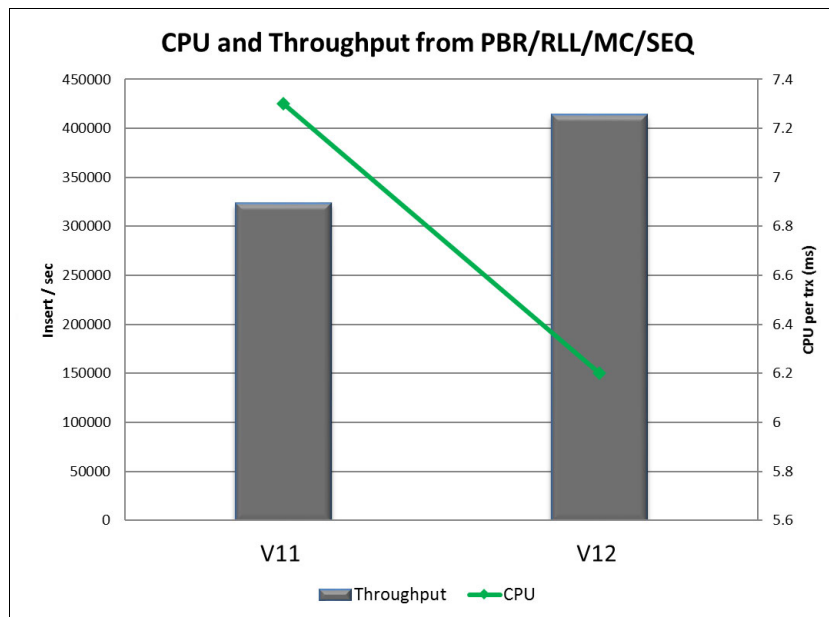


Figure 3 Improved throughput and CPU with the fast insert algorithm

Three tables are involved in this workload: One table with one index, the second with two indexes, and the third with three indexes. The journal table is defined as PBR UTS with row-level locking and Member Cluster. The index is based on a sequential key, and the rows arrive at the table in an order based on that sequential key.

The chart compares insert throughput (inserts per second) and CPU per transaction in milliseconds on DB2 11 with insert algorithm 1, and on DB2 12 with insert algorithm 2. The CPU per transaction drops significantly, from around 7.3 seconds per transaction to 6.2, approximately a 15% reduction. The insert throughput climbs dramatically, from just over 300,000 per second to just over 400,000 per second, which is around a 30% increase.

Keep in mind that the benefits you will see will vary. Workloads that are constrained by lock/latch contention on the space map pages and on the data pages in the table space or table space partition are more likely to benefit from the new insert algorithm.

Access path/access plan stability

Plan stability (the ability to stabilize access paths, also called access plans) was delivered for static SQL in DB2 9 and continued to evolve in DB2 10 and DB2 11. DB2 9 allowed customers to revert back to a previous access path after a BIND REPLACE or REBIND operation, typically because the new access path performed worse than the old one. DB2 10 complemented this feature with the ability to ask the DB2 Optimizer to reuse a previous access path but still build new runtime structures. DB2 11 provided usability enhancements and access path reuse capability at the statement level for both static and dynamic SQL. DB2 12 takes another significant stride forward with several new and improved features. The following are the main points:

- ▶ Dynamic SQL plan stability (the most important enhancement in this area)
- ▶ Incremental enhancements to improve static plan stability usability
- ▶ Preserving the local dynamic statement cache (DSC) at rollback
- ▶ Integration of RUNSTATS and the optimizer, which provides automatic update of statistics profiles
- ▶ Improved support for statistics profiles, with automatic update driven by CREATE/ALTER INDEX DDL, and usage of statistics profiles by inline statistics collection

Dynamic SQL plan stability

For many customers, unstable performance of repeating dynamic SQL statements has been an ongoing problem. Dynamic SQL is prepared (that is, the access path is recalculated) at every Global DSC miss and requires an expensive full prepare. A full prepare can require several tens of thousands to hundreds of thousands of CPU instructions. If statements are regularly purged from the Global DSC, they can be reoptimized many times a day. To add to the unwanted CPU cost of large numbers of prepares, environmental changes can result in access path changes when a dynamic SQL is reprepared, possibly leading to performance regression. This concern includes changes introduced by these events:

- ▶ RUNSTATS updates to the DB2 catalog, reflecting changes in data volumes and column value distribution
- ▶ Software maintenance affecting optimizer choices
- ▶ Migration to a new release of DB2
- ▶ ZPARM (system parameter) changes
- ▶ Database schema changes

Avoiding unwanted access path changes for dynamic SQL can be difficult to manage. Before DB2 12, static SQL had several advantages. The access path is established and locked in at BIND time, and, with a few exceptions, is not recalculated until an explicit BIND REPLACE or

REBIND operation. The static plan management features introduced in DB2 9 and DB2 10 already discussed, which provided advanced plan management functions.

DB2 12 makes static SQL advantages available to repeating dynamic SQL workloads. DB2 12 does this by providing the infrastructure to save, reuse, monitor, and manage dynamic SQL runtime structures. It stores dynamic SQL statements and their associated Global DSC structures in the DB2 catalog so that they can be reloaded into the Global DSC from the catalog on a Global DSC miss, and reused on subsequent executions of the statement. This feature provides access path stability, consistency, and predictability, not just in a single member but across several or all of the members of a data sharing group. This new capability to maintain access path stability is unaffected by events such as DB2 stop and restart, RUNSTATS, DB2 maintenance, and release migration.

To make dynamic SQL plan stability usable, DB2 12 provides a number of management and control mechanisms.

New ZPARM CACHEDYN_STABILIZATION controls whether dynamic SQL statements, or groups of statements, can be captured for stabilization and saved in the DB2 catalog, and whether they can be loaded into the Global DSC from the catalog when there is a miss in the Global DSC.

New commands can be used to perform a one-time capture of qualifying statements, or logical statement groups, cached in the Global DSC, or to keep monitoring the Global DSC to capture statements that qualify based on user-defined criteria. That is, capture can be performed with or without monitoring. Statements can qualify based on a statement ID, a current SQLID, the number of statement executions, or a combination of these. In addition, customers can set a global variable to drive stabilization of dynamic SQL for specific applications. As an aside, it is important to note that, when an application issues a prepare for a statement, for that statement to match with a prepared statement already in the cache, or stabilized in the catalog, the matching criteria include the SQL statement text, the AUTHID, and APPLCOMPAT (application compatibility).

It is possible to remove statements from stabilization control with new options on the FREE command. That is, it is possible to remove them from the DB2 catalog and from the Global DSC. This procedure can be done for an individual stabilized statement, or for a stabilized statement group. When this process is done, not only is the statement or statement group removed from the DB2 catalog, but it is also invalidated in the DSC.

To make it easier to identify statements to be removed from stabilization control, the LASTUSED stabilized dynamic SQL statistic can be used to identify statements that have not been run for a period of time, providing a list of candidate statements for removal from stabilization control.

Full invalidation support is retained. Database schema changes (SQL DDL) and authorization changes (SQL DCL) can invalidate stabilized dynamic SQL statements, in the same way that such changes can invalidate static SQL packages. To enable sophisticated access path management when stabilized statements have been invalidated, DB2 12 provides full EXPLAIN support. This support enables customers to explain the saved current copy and the saved invalid copy, allowing them to take action to favor one or the other of the access paths.

DB2 12 provides instrumentation so that customers can know which statements are using which stabilized runtime structures. The statement hash or query hash (a value derived from the SQL statement text itself) is externalized. Therefore, customers can use the hash as a stable identifier to monitor statements over time, even if they move in and out of the global statement cache or the catalog as stabilized dynamic SQL. In addition to statement tracking and explain support, customers can also obtain statistics about the cache and catalog hit ratios.

There are some key limitations in this initial implementation of dynamic SQL plan stability. Statements that are prepared with CONCENTRATE STATEMENTS WITH LITERALS, and statements accessing temporal objects are not supported for stabilization. Also, none of the plan management features such as REBIND SWITCH, or APREUSE/APCOMPARE are as yet available.

Static plan stability: Usability

DB2 9 introduced plan management to retain copies of the original, previous, and current versions of a package. Before DB2 12, only two options were available when freeing packages: Free all copies of a package, or free the inactive copies (original and previous). In DB2 12, FREE PACKAGE is enhanced to provide the capability to selectively free either the previous or original copies individually, both of the inactive copies, or all copies. This feature allows customers to easily remove, for example, a stale original copy without removing an up-to-date previous copy. It is also now possible to choose to free only invalid copies after schema changes or release migration has invalidated one or more of the copies.

As a usability enhancement, you can now also free invalid package copies when an application using the current copy of the packages is running because the exclusive package lock held by the running application no longer extends to the invalid copies.

DB2 10 introduced the capability to influence the optimizer to reuse the previous access path at REBIND time by using the APREUSE(ERROR) option. This feature proved to be very successful, with customers requesting further enhancements. Therefore, DB2 11 provided customers with more flexibility by delivering the APREUSE(WARN) option. DB2 12 takes this feature further by providing a new option for REBIND PACKAGE, called APREUSESOURCE. This option allows customers to explicitly specify whether the current, previous, or original copy should be used as the access path source for APREUSE.

As well as enhancing usability, this feature also addresses the problem where an invalid package copy accidentally becomes current. This problem could occur where, for example, in the following scenario:

- ▶ A DROP and CREATE of an index causes the current package copy to become invalid.
- ▶ When the package is automatically rebound by DB2, it picks up a bad access path.
- ▶ The previous and original versions are also invalid, but at least one of them contains the wanted access path.
- ▶ REBIND SWITCH to one of these results in an invalid package becoming current, with the risk of automatic rebind before REBIND with APREUSE can be run.

REBIND with APREUSESOURCE avoids the two-step process to restore a previous good access path and additionally avoids the risk of an invalid package becoming the current package.

Preserving the dynamic statement cache at rollback

At some customer sites, application programmers are using ROLLBACK instead of commit at the end of a read-only unit of work. This option causes the statement to be purged from the local DSC and drives an expensive full prepare. A second side-effect is to pollute the DB2 accounting and statistics metrics. At rollback, the access path is now preserved in the dynamic statement cache, rather than being invalidated, providing customers with a welcome performance enhancement by replacing the cost of a short prepare with prepare avoidance.

RUNSTATS enhancements for SQL performance

DB2 12 makes a number of improvements to RUNSTATS for improved access path selection.

DB2 12 enhances the CLUSTERRATIO formula to handle the effect that unclustered inserts have on the clustering index, and the effect of the space search algorithm on inserts into table spaces with larger segment sizes. This enhancement ensures that CLUSTERRATIO better reflects dynamic prefetch behavior.

Before DB2 12, RUNSTATS (and other utilities such as REORG and LOAD configured to collect inline statistics) invalidated dynamic SQL statements in the global dynamic statement cache that depended on the target object. For many customers, this process often lead to unwanted access path changes, so DB2 12 added an INVALIDATECACHE option to RUNSTATS, with a default of NO, to prevent RUNSTATS from invalidating the cache unless the parameter is set to YES. This is a change in the default behavior of RUNSTATS from previous releases, so if you want to ensure that statements are invalidated you should specify INVALIDATECACHE YES.

However, RUNSTATS UPDATE(NONE) REPORT(NO) will continue to invalidate the cache to ensure that existing customer jobs to invalidate the cache are not affected. For other utilities, statement invalidation will only occur if the object was in a restricted state before the utility began, such as rebuild pending or reorg pending states. Other utilities (for example, LOAD, REORG) do not provide an option to control dynamic statement cache invalidation.

Statistics profiles are now supported by inline stats, ensuring that they can be used whenever RUNSTATS can be collected by using the REORG and LOAD REPLACE utilities. Additionally, index changes by using SQL DDL update the profile to ensure that dropped indexes are removed, or new indexes are added.

DB2 12 provides an automated value for COUNT for FREQVAL, used for the collection of statistics on skewed column values. Knowing what number to specify for FREQVAL COUNT n has always been data dependent and therefore difficult. The number not only varies from table to table, but also for a single table over time. In DB2 12, specifying 'FREQVAL' without 'COUNT n' results in DB2 automatically determining the appropriate number of frequently occurring column values to collect. The objective is to allow DB2 to collect all the skewed values, up to the top 100 most skewed values, or until no skew can be detected for the remaining values.

More broadly, determining which statistics to collect has often been a challenge for customers, especially when large numbers of objects must be managed. To address this challenge, the DB2 11 optimizer externalized information about missing and conflicting statistics into catalog and explain tables. DB2 12 takes this a stage further so that the optimizer automatically updates the statistics profile for an object with RUNSTATS recommendations. Collection of statistics, specifying USE PROFILE, whether by using the RUNSTATS utility or REORG and LOAD REPLACE with inline statistics, automatically all statistics that are recommended by the optimizer.

This new feature is known as enhanced statistics profile management. As shown in Figure 4, when calculating the access path for a given query, DB2 updates the profile in the catalog to ensure that missing statistics are collected when using a statistics profile.

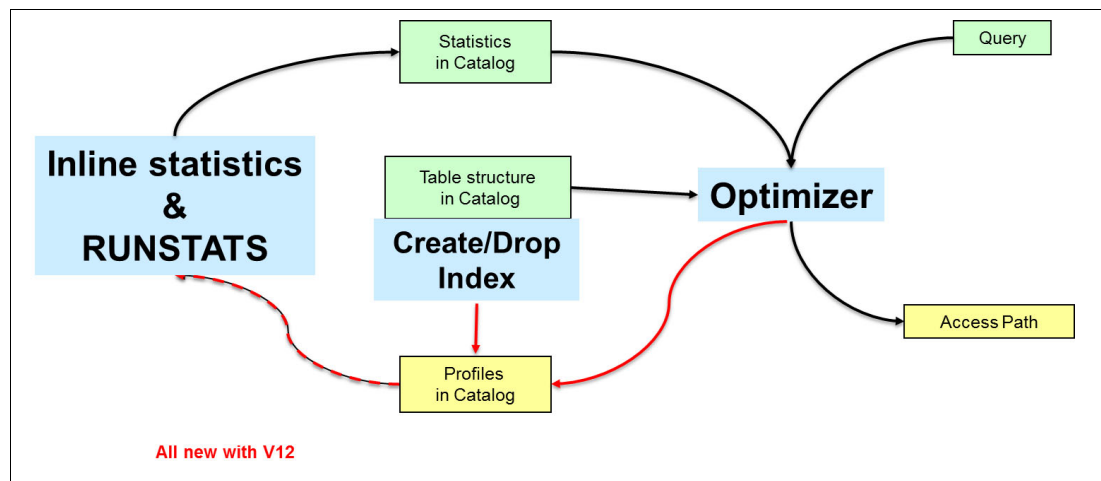


Figure 4 NEED A CAPTION

The DB2 optimizer updates the statistics profile to ensure that missing or conflicting statistics are corrected by RUNSTATS. This process is controlled by the new ZPARM STATFDBK_PROFILE. In addition, CREATE and DROP INDEX also update the profile.

The final point to mention about enhanced statistics profile management is that the DB2-supplied stored procedure DSNACCOX always recommends RUNSTATS whenever a profile has been updated. To ensure that you collect current statistics recommendations, specify USE PROFILE on RUNSTATS.

More granular global commit LSN and Read LSN

DB2 uses the Commit log sequence number (LSN) for lock avoidance checking to avoid taking locks unnecessarily and to reduce CPU consumption. Previously, in a data sharing environment, there was a single Commit LSN value across all group buffer dependent objects. The Global Commit LSN is the unit of recovery ID (URID) of the oldest uncommitted unit of recovery (UR) in the data sharing group. It is used to do lock avoidance checking for all group buffer pool-dependent objects in the data sharing group. This LSN means that a single long running program that does not commit, or that only commits infrequently, holds back the Commit LSN (CLSN), degrades lock avoidance, and causes poor space reuse when inserting LOBs.

To mitigate the impact of long running URs, DB2 12 maintains the Global CLSN of the 500 worst objects, and if an object outside those 500 objects is being accessed.

DB2 also provides a more granular current Read LSN value, which provides better space reuse from DELETES when inserting LOBS.

Avoid scheduling unnecessary prefetch

This feature is designed to avoid the CPU overhead of unnecessary scheduling of dynamic prefetch engines. The problem that it addresses is that, before DB2 12, when all the pages being accessed on behalf of the application are already in buffer pool memory, DB2 schedules a prefetch engine. This process wastes CPU, unnecessarily ties up a prefetch

engine, and can cause “out of prefetch engine” conditions, affecting other applications running on that DB2 member or subsystem.

DB2 development has tried to solve this problem in previous releases of DB2. It has been solved in DB2 12 by avoiding scheduling the prefetch engine until the first buffer miss (that is, the page is not found in the buffer pool). While the benefits of this feature are clearly workload-dependent, a CPU reduction of up to 6.8% for OLTP workloads, and up to 4.5% for query workloads has been observed.

Miscellaneous performance enhancements for traditional workloads

DB2 12 adds a buffer pool advisory mode that simulates the effects of larger size local buffer pools, in line with the emphasis on larger real memory exploitation. This feature provides a new set of instrumentation, simulating what would happen in terms of reducing I/O. These metrics can be seen in the DB2 statistics trace and in the output of the -DISPLAY BUFFERPOOL command. Buffer pool simulation has a small CPU overhead of about 1% - 2% per buffer pool and about a 1% real memory overhead, assuming a 4 KB page size. This change has been retrofitted to DB2 11, and early experiences show that this is a significant improvement on other buffer pool simulation technologies.

Claim/declaim processing has been streamlined. Our analysis of OLTP workloads shows that much processing is consumed by repetitive declaim and reclaim of application objects. This limitation is particularly true of persistent, long-running threads that are bound with RELEASE(DEALLOCATE), where there are many commit scopes during the lifetime of the thread. DB2 12 has been enhanced to avoid the reclaim overhead, but this feature means that the claim count is always elevated. However, the break-in thread capability introduced in DB2 11 has been enhanced to drive a dummy commit to take the claim count to zero to allow online REORG and other drainers to continue to break in.

Improvements have been made in storage (memory) pool management to simplify and remove scalability inhibitors. These improvements are in the area of the various Environmental Descriptor Manager (EDM) pools, and in the area of LOB and XML storage at the application and at the system level. “Hard” limits, in terms of maximum pool size, have been removed. The overall goal was to reduce latch contention to reduce CPU consumption and system “pinch” points that can become significant when the system is under stress.

There has been a number of other performance improvements:

- ▶ The CPU overhead of declaring DGTTs has been reduced.
- ▶ The log force write in data sharing when the values of identity column and sequences are cached has been removed.
- ▶ DB2 12 satisfies a long-standing requirement to implement Resource Limit Facility control for static SQL packages.
- ▶ The LASTUSED column in the SYSIBM.SYSPACKAGE table is now much more useful for determining when unused packages can be freed. Previously, a BIND REPLACE would set this column to January 1st 1901, but now this leaves the value unchanged.

System scaling enhancements

DB2 12 includes a number of system scaling enhancements, enabling customers to optimize use of system resources and improve throughput, with higher transaction volumes processing more data.

First, a number of large n-way scaling enhancements improve efficiency when running on LPARs with a high number (n-way) of general processors (CPs):

- ▶ Log latch contention reduction has been featured in several DB2 releases, and DB2 12 makes further significant improvements in this area. Testing shows up to 41% CPU reduction and 6% throughput improvement for high contention cases.
- ▶ In terms of buffer pool scaling improvements, use of LC23 has been reduced, and all use of the Perform Locked Operation (PLO) instruction has been avoided. Use of the PLO instruction was introduced in DB2 10, but we never got the performance improvements that we expected. As a result of the DB2 12 changes, DB2 development has observed a 5% to 30% CPU reduction when accessing hot pages.
- ▶ DB2 12 also focuses on IRLM latch contention reduction, and on reducing EDM DBD and skeleton pool contention, improving scalability.

Second, there are optimizations for new IBM z Systems® hardware, in terms of exploitation of the IBM z13® decompression enhancement, and in terms of internal structure changes for processor cache efficiency, enabling more processor prefetch.

Thirdly, the limit on total buffer pool size has been increased to 16 TB, making DB2 12 ready for future operating system and hardware enhancements. At the time of writing, these are theoretical limits. For example, if you are running on a z12, the maximum amount of memory configurable at the LPAR level is 3 TB, and 4 TB on a z13. The practical limit is around 1 TB per DB2 subsystem.

Assuming the use of 1 MB page frames, the practical limit on the maximum size of a buffer pool is around 800 GB because the large frame area (LFAREA) cannot be more than 80% of the real memory available on the LPAR. When specifying the LFAREA, it should be no larger than 75% of the real memory on the LPAR, minus the memory required to support the 4 KB frames needed by IBM CICS®, IBM IMS™, DB2, and other users of the system.

Finally, the maximum size of the active log data sets is now much larger than the previous 4 GB limit, at 768 GB, but the maximum number of log pairs remains 93.

DB2/DASD synergy

There are two enhancements in the area of DB2/DASD synergy, both of which have been retrofitted to DB2 10 and DB2 11.

The first is the capability to use the z/OS Hyperwrite feature, also known as the PPRC log write accelerator. The target is to reduce the latency of log writes in a PPRC environment, which is synchronous DASD mirroring over distance. This is of high value for OLTP applications that perform very frequent short writes to the DB2 log. DB2 development has measured up to a 30% reduction in log write latency.

The second of these enhancements is improved integration with the IBM DS8870 Easy Tier® multi-temperature management software. This is particularly beneficial when the DS8870 has a mixture of flash drives and traditional HDD drives. Easy Tier monitors the types of I/O requests being issued from the various systems, and moves data set extents between the flash drives and the HDD drives based on the frequency of access.

Until this new enhancement became available, the information about the types of I/O requests and the frequency of access would be lost when a REORG of a table space or index space was run. A new API into Easy Tier is used by the REORG utility, ensuring that the information is carried forward across REORGs, maintaining performance.

High-level performance expectations

This is an early view of the performance expectations for DB2 12. As the Early Support Program progresses and the product approaches general availability (GA), these figures will probably change. When DB2 12 becomes GA, the performance results will be published in an IBM Redbooks publication.

In the area of system and OLTP performance, the following are the expectations:

- ▶ A 2-3% CPU reduction without the Index In-Memory (Index fast traversal block (FTB) area) feature
- ▶ A 5-10% CPU reduction by using the Index In-Memory (FTB) area feature
- ▶ Further reduction is possible with contiguous buffer pools, persistent threads bound with `RELEASE(DEALLOCATE)`, or both

In the area of query performance, we expect a wide range of improvement:

- ▶ Typically in the area of 0-20% without a new access path
- ▶ Typically 10-40% with a new access path
- ▶ We have observed up to a 90% reduction in our testing for some queries

For concurrent insert against a table defined in a UTS with `MEMBER CLUSTER`, customers can expect a 5% - 10% CPU reduction. This level of improvement requires that the current bottleneck is in space search or in space map page or data page contention.

Instrumentation enhancements

For customers to measure and understand the performance characteristics of their applications and systems, DB2 already provides a rich set of instrumentation in the form of performance metrics. DB2 12 provides more information to customers by further enhancing the available instrumentation with a series of improvements:

- ▶ More granular wait times for IFCIDs 316 (which records performance metrics for dynamic SQL statements) and 401 (which does the same for static SQL statements), providing figures for accumulated wait time caused by global contention.
- ▶ The statement-level section of IFCIDs 53 and 58 PREPARE of a dynamic SQL statement has been enhanced to provide similar information to that already provided for INSERT, UPDATE, and DELETE.
- ▶ SQL performance tracing has been enhanced by adding the RDI Section Number to IFCIDs 53 and 58.
- ▶ The correlation header in IFCIDs now includes the batch job-step name.
- ▶ The number of REFRESH TABLE SQL statements has been added to the counts in data area DSNDQXST, referenced in several IFCID record types.
- ▶ Work file, and temporary file usage information has been added to the DB2 accounting trace.
- ▶ The precision of IFCID 199 (data set I/O statistics) has been enhanced to record this information in microseconds.

Performance enablers for modern applications

Query performance has become increasingly important to customers over time, as they seek cost-effective ways to discover valuable information hidden in the large amounts of business and operational data. Improved analytical query performance enables them to make business decisions faster at less cost.

DB2 12 has over twice the number of performance enhancements compared to DB2 11, which was itself known for impressive query performance improvements. Many of the enhancements are targeted at SQL constructs seen in both new analytics and complex transactional workloads.

DB2 development has tried to deliver up to a 25% CPU improvement for traditional query workloads through optimizations for DISTINCT, GROUP BY, reduced work-file usage, multiple index access, and list prefetch.

DB2 is intended to deliver up to a two times improvement for modern SQL applications, focusing on performance improvements for next generation SAP applications, for real-time analytics, and for complex OLTP workloads. These optimizations are related to outer join, UNION ALL, stage 2 join predicates, CASE expressions, VARBINARY data type indexability, DECFLOAT data type indexability, and others.

Parallel query child tasks are now 100% zIIP eligible in DB2 12. In prior releases, a complicated formula determined which parts of the parallel query were eligible for zIIP offload. In DB2 12, this process becomes much easier with all child tasks associated with the 'queries being zIIP eligible.

It is important to note that modern applications contain more complex SQL patterns, which are specifically targeted in DB2 12. They tend to use more sorting, joins, stage 2 predicates, and others. These complex patterns are less common in traditional OLTP and batch workloads.

Query workload CPU and elapsed time improvement from V11 (%)

Customers help DB2 development to profile various types of customer workloads. Development then uses these profiles to build important target workloads to stress test the performance enhancements in each release. These workloads are used to evaluate the performance improvements introduced by new DB2 releases. For DB2 12, the performance results were compared with those in DB2 11. The chart shown in Figure 5 demonstrates the variability of the DB2 12 performance improvements across different workload types.

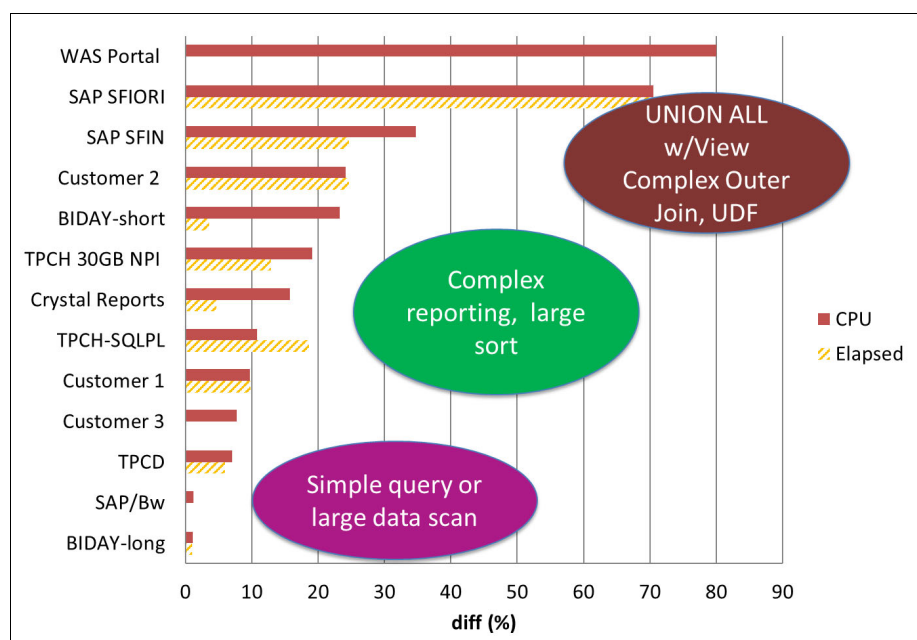


Figure 5 DB2 12 query workload CPU and elapsed time improvement over DB2 11

The percentage improvement in CPU and elapsed time for workloads involving simple queries or large data scans is relatively small compared to DB2 11. However, for query workloads involving complex reporting and large sorts, the CPU and elapsed time improvements are much more significant. Very significant savings are seen for query workloads involving UNION ALL with views, complex outer joins, UDFs, and so on.

These results are isolated measurements and might not be representative of actual performance results achieved by customers because of the large degree of variability in real customer workloads.

High-level performance focus

DB2 development has focused on two high-level performance targets:

- ▶ Query workloads typical of newer applications, including but not restricted to the analytical workloads so important to customers
- ▶ More general query performance bottlenecks

The query patterns used by newer workloads tend to use complex views or table user-defined functions (table UDFs), including functions and operators like UNION ALL, outer joins, and join predicates involving stage 2 predicates. They also tend to use CASE expressions, CAST functions, and scalar functions. These query patterns are targeted in DB2 12 for performance improvement.

The SQL UNION ALL construct is used extensively by DB2 in system temporal implementations, and in transparent archive. It is also widely used in new analytics workloads such as SAP Fiori and SAP Simple Financial, and in transactional workloads such as WebSphere Portal and WebSphere Business Process Server, among others. It turns out that UNION ALL and outer join share many of the same performance challenges and solutions. These newer workloads also make extensive use of outer joins and other complex query SQL operators and constructs such as CASE expressions, CAST functions, and scalar functions.

For general query performance bottlenecks that affect all complex workloads, IBM has focused on these goals:

- ▶ Reducing sort/work-file usage and contention for prefetch engines
- ▶ Reducing the cost and frequency of PREPARE
- ▶ In the area of I/O performance, avoiding the unnecessary scheduling of prefetch engines

To try to reduce the cost of PREPARE, DB2 development has targeted the case where a table has many indexes defined on it because this is one of the most expensive query types to PREPARE. To reduce the frequency of PREPARE, DB2 12 introduced Dynamic Plan Stability, the capability to capture dynamic SQL statement text and runtimes in the DSC and store them in the DB2 catalog, and to load them from the catalog into the DSC. PREPARE frequency is further reduced by avoiding Global DSC statement invalidation by utilities.

Possible future use of UNION ALL can include the pushing of sub queries into the IDAA. For reference, we did push sub queries out to the older ISAO, but when we went to the Netezza® based technology, we only pushed Full queries out to the IDAA). Pushing sub queries out to the IDAA is on the future requirements list, which would involve code changes and the use of UNION ALL.

Query performance focus

To focus in more detail on query performance, this section looks at four major topics:

- ▶ Improving the performance of UNION ALL and outer join
- ▶ Improving the performance of table UDFs
- ▶ Adaptive Index
- ▶ Other DB2 12 Query Performance Enhancements

Improving the performance UNION ALL and outer join

UNION ALL and outer joins share similar performance challenges. Improvements to these SQL constructs have largely been realized by avoiding materialization and by avoiding sort. Materialization can result in significant performance degradation and increased work-file resource usage if filtering is not applied before the materialization. Therefore, the first challenge is to address excessive work-file usage due to materialization, and the second is to address the lack of ability to apply predicates earlier.

To reduce materializations and to reduce the cost of materialization when it is required, DB2 12 introduces several changes:

1. DB2 12 reduces the number of situations where work-file usage is required for materialized outer join query blocks or UNION ALL legs. This change will reduce CPU and, possibly, I/O overhead.
2. DB2 12 trims columns in a materialized view or table expression that are not required by the outer query. DB2 also prunes unique LEFT OUTER JOIN views and table expressions if the columns in those views and table expressions are not included in the SELECT list. Pruning of unique LEFT OUTER JOIN tables was already delivered in DB2 10, and DB2 12 extends that to views and table expressions.
3. DB2 12 pushes predicates into UNION ALL legs or OUTER JOIN query blocks if it is cost effective for the optimizer to do so.
4. DB2 12 also pushes ORDER BY and FETCH FIRST into UNION ALL legs.
5. Finally, DB2 12 reorders the OUTER JOIN tables to avoid materializations if it is cost effective for the optimizer to do so.

Improving table UDF performance

Where possible, DB2 12 enhances table UDFs to have similar merge capabilities as views, avoiding materialization. It also enables the indexability of join and correlation predicates that are passed into the table UDF as a parameter, reducing the need for sort.

Adaptive index

Another major enhancement is the execution-time adaptive index, usually referred to as simply *adaptive index*. This is a solution for generic search-type queries. These queries present a challenge for any query optimizer because the degree of filtering of the predicates can change on every execution. For example, consider a predicate on LASTNAME using the LIKE operator where the predicate argument is stored in a host variable.

A predicate argument of 'J%' is less filtering than 'Jones%', which is less filtering than 'Holmondley%'. Now consider a query with many predicates. The possible argument values for each predicate range from highly filtering to unfiltering. This limitation makes it impossible for the optimizer to choose the single best access path for all executions because the best access path depends on how filtering each predicate is.

Adaptive index is a DB2 12 enhancement to multi-index and single index list prefetch-based access plans that introduce logic at execution time to determine the filtering of each index to ensure optimal execution sequence of indexes, or earlier reversion to a table space scan if no filtering index exists.

The solution has two parts.

First, DB2 12 allows RID-based plans (single index list prefetch or multiple index access) to quickly determine the filtering of the predicates from the index at run time without requiring the BIND option REOPT(ALWAYS), and to adjust the access path based on the degree of filtering of the predicates.

In the cases of list prefetch and multiple index ORing, this process provides an early opportunity to fall back to table space scan if a large percentage of the table must be read even when using index access.

For multiple index ANDing, DB2 can consider dynamically reordering the sequence in which the indexes are accessed from most to least filtering. DB2 can optionally eliminate non-filtering indexes stages (early-out) or fall back to a table space scan if there is no filtering.

Second, the optimizer uses uncertainty to determine the risk of a single index plan. A quick evaluation is performed based on the literals used. Any further evaluation of filtering is deferred until after one RID block is retrieved. This process ensures that very short running queries do not incur evaluation overhead.

To provide more detail, the optimizer considers a multi-index or list prefetch plan as in prior DB2 releases. However, in DB2 12 an additional consideration is the uncertainty that is associated with the filter factors, which is the estimated degree of filtering of each predicate. For example, a predicate such as WHERE COL1 < ? has a high degree of uncertainty because the value specified at execution might qualify all of the rows, or none of the rows. If a high degree of uncertainty exists for the indexes chosen, then the optimizer can add more indexes for a multi-index access access plan.

Then, at execution time, DB2 reevaluates the estimated filtering for each leg in the multi-index plan or single leg of a list prefetch plan after the literal values are known. It uses this information to determine whether the index legs should be reordered, the index legs should be discarded, or the plan should revert to a table space scan.

This enhancement applies to any multi-index or single index list prefetch plan, regardless of whether the optimizer determined there was high uncertainty to the predicate filtering or not. A rebind is required to take advantage of this feature for static SQL.

The following is a simple example of the targeted use case, representing a generic search query type:

```
SELECT * FROM TAB1 WHERE COL1 < ? AND COL2 < ? AND COL3 < ?;
```

The query retrieves all the columns from the table TAB1, and the search is based on the values in three columns: COL1, COL2, and COL3. These are range predicates because in each case the less-than operator causes DB2 to qualify rows where the column has a value less than a parameter marker. The table has three single-column indexes: IX1 on COL1, IX2 on COL2, and IX3 on COL3.

Filtering of this query is entirely dependent on the literal values for the parameter markers at execution time. This is a common pattern for search screens that generate many range predicates such as BETWEEN, LIKE, less than, and so on. It is also common that one index provides good filtering unless a highly skewed value is searched.

This is a good example of a scenario where the DB2 optimizer can recognize that each WHERE clause predicate has a high degree of uncertainty associated with its filtering estimate. If so, the indexes for the query are viable candidates for a multi-index access plan. Predicate patterns seen in search screens are good candidates for multi-index access and will benefit from the adaptive index enhancement, where the degree of filtering of the predicates can potentially change at each execution.

As an aside, this feature is not intended to encourage the use of many single-column indexes on tables. It is much better to have a smaller number of carefully chosen multi-column indexes.

Other DB2 12 query performance enhancements

DB2 12 introduces several other query performance enhancements:

- ▶ The first of these enhancements relates to join predicates with stage 2 expressions. Typically, these expressions involve arithmetic and scalar functions such as SUBSTR, DATE, INTEGER, and CHAR. The solution in DB2 12 is to calculate the result of the expression before the sort, and then to sort on the expression. This process allows merge scan join and sparse index to be used with many stage 2 join predicates.
- ▶ Previously, when using predicates on columns with VARBINARY and BINARY data types where the length of the operands did not match, DB2 would make these predicates stage 2, that is, non-indexable. DB2 12 implicitly uses CAST on these columns, making them stage 1 and indexable.
- ▶ The VARBINARY data type allows matching index access in DB2 12, and any scalar functions that returns VARBINARY, such as COLLATION_KEY, can now use Index on Expression.
- ▶ Expression evaluation for CASE and SUBSTR has been optimized to improve execution performance.
- ▶ Expressions (such as CASE expressions, UDFs, and scalar functions) that are duplicated in the SELECT list as a result of a view or table expression merge are now only run once. The result is then shared by all references to that expression. This process only applies if the duplication was due to a DB2 merge, and not if the SQL was originally coded with duplicate expressions.
- ▶ User Defined Functions that are defined as DETERMINISTIC will now have their execution results cached within the life of a single statement, such that repeat calls with the same input values can return the result from the cache rather than rerun the UDF.
- ▶ To enable more use of parallelism, 100% zIIP offload is available for parallel child tasks. In addition, optimizations have been made to improve the access path choices available for parallelism and the efficient distribution of work across the child tasks, reducing cost and resource consumption.
- ▶ A number of enhancements have been made to sort:
 - Sort performance is improved for GROUP BY and DISTINCT to improve duplicate removal as input to sort.
 - Reducing work-file usage for GROUP BY and DISTINCT. There are potentially many more nodes in the sort tree. The previous maximum was 32,000, and this has been raised significantly to 512,000. This change reduces the number of merge-passes in the sort.
 - Reducing the key length for GROUP BY and DISTINCT and for sparse index. As a result, sorts involving fixed-length columns might see a reduction in work-file usage where the sort key and the data share columns.

- Continuing the progress towards more use of in-memory sort for smaller sorts that begun in DB2 Version 9. This feature has been extended to intermediate sorts for all types of query blocks.

Application enablement

New application development technologies and paradigms are entering the marketplace at a rapid rate, providing businesses with the capability for continuous and high-speed deployment of new IT solutions with high functionality. These solutions are important to either stay competitive or to gain a competitive advantage. Many of them are concerned with processing large amounts of data quickly and inexpensively, performing complex analysis, and discovering information and meaning in the vast amounts of data held in data warehouses and operational data stores.

IBM customers want to be able to unlock the valuable information in the ever-growing amount of data stored in their DB2 for z/OS databases. DB2 12 continues the process begun in previous releases of adapting to modern application development technologies and patterns with a large set of enhancements aimed at enabling modern applications.

DB2 for z/OS: Adapting to modern application development paradigms

Many modern application developers work with REST or RESTful services and JSON data formats, both as a matter of preference and of company strategy, as seen in Figure 6. The DB2 Adaptor for z/OS Connect provides the means to do this type of work. Currently, a Beta program is in progress for this feature, and availability is planned for DB2 10 and 11 as well as DB2 12 when the Beta program finishes.

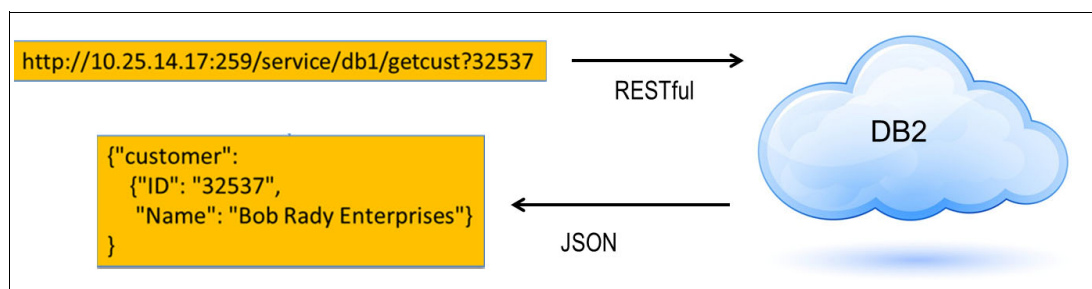


Figure 6 RESTful services and JSON data formats with DB2 for z/OS

Cloud/Mobile integration with DB2 and z/OS Connect

DB2 development wanted to get RESTful API requests and JSON data formats into the mainframe environment in a way that enables customers to take maximum advantage of the assets that exist there. These assets include IMS and CICS transactions, and DB2 stored procedures.

JSON is vital because it is the most widely used protocol / language-independent data format for developing mobile apps, and code for parsing and generating JSON data is readily available in a large variety of programming languages.

Representational State Transfer (REST) is an architectural style of APIs for networked hypermedia applications that was introduced in 2000. It is one of the most pervasive web application technologies. Its main use is to build web services. A service based on REST is called a RESTful service.

There are two objectives in delivering the DB2 for z/OS enablement of RESTful APIs and JSON data formats:

- Provide a common and consistent entry point for mobile access to one or many of the back-end systems (that is, IMS TM, CICS, and DB2).
- Simplify front-end functions by allowing them to pass RESTful API requests and to retrieve JSON formatted data, rather than being involved in, or even aware of, complex data transformation.

The solution is z/OS Connect, which is a unified solution for cloud, mobile and web integration. It offers service enablement, management, and discovery, and secure access to z/OS assets such as IMS TM or CICS transactions, or DB2 stored procedures. DB2 has provided an adaptor for z/OS Connect that runs as a new DB2 started task. Inside that started task is an IBM WebSphere® Liberty Profile Server that handles the RESTful API requests.

As can be seen from Figure 7, this configuration greatly simplifies connecting mobile devices to z/OS assets, allowing customers to benefit from a streamlined approach to mobile application development. IBM Bluemix, currently in open beta, is one of the technologies that allows web, cloud, and mobile apps to connect into resources accessible through the cloud, including z/OS assets.

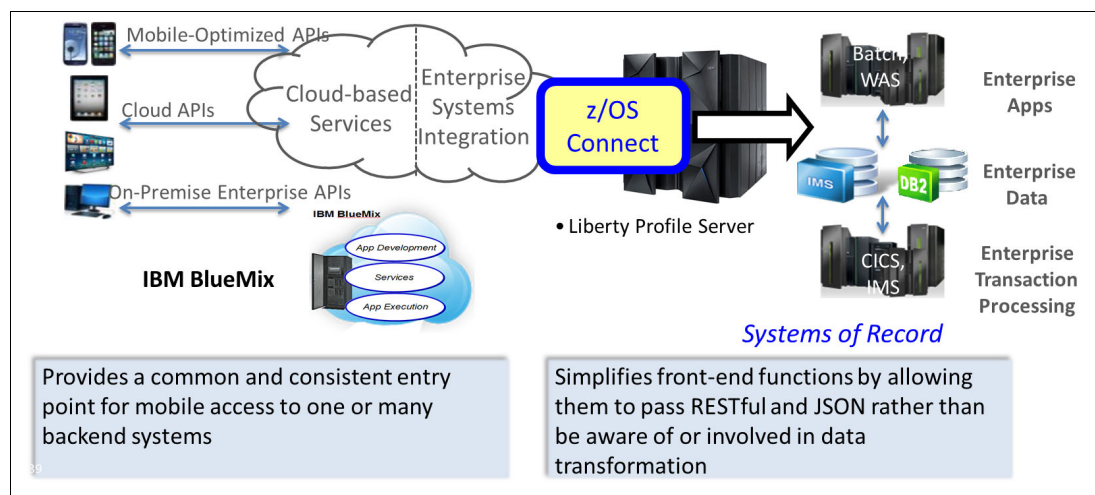


Figure 7 Cloud/Mobile integration with DB2 and z/OS Connect

Application-related enhancements

DB2 12 delivers an impressive set of application-related enhancements. There are many changes in the areas of the SQL language, XML performance, and improved JSON enablement. The following are some of the enhancements related to application management and ease-of-use:

- The first of these is DRDA Fast Load, which provides a callable API command for fast streaming of data into DB2 across the network. We discuss DRDA FAST Load in more detail in the following pages.

- ▶ System profiling is enhanced so that system profiles (which allow you to monitor and control connections and set default values for special registers for specific applications) are automatically started by DB2.
- ▶ It is also possible to set default values for global variables based on system profiles. For example, you can set a default value for the `MOVE_TO_ARCHIVE` global variable to support transparent archiving.
- ▶ There is a new option for the `MODIFY DDF` command, `PKGREL(BNDPOOL)`, that causes high-performance DBATs to be pooled at connection termination, instead being destroyed. This enhancement has been retro-fitted to DB2 11.
- ▶ IMS Attach support is added for `DSNULI`, the universal language interface.

This section looks at some of the larger items, including a long list of incremental SQL and XML improvements. However, remember that the order in which these items are presented is not intended to represent their relative importance. Different customers will find different enhancements of interest, based on their requirements.

DRDA Fast Load

In discussing DRDA Fast Load, it is important to understand the problem. DB2 already provides a stored procedure called `DSNUTILU` to provide applications with the capability to load data into DB2 from a network client. However, this procedure is difficult to use, and the application must transfer the data to a z/OS file.

The solution is a new DB2 Client API, with support for the command-line interface (CLI) and the command line processor (CLP), providing the capability to load data into DB2 remotely. This API enables easy, fast loading of data from a file that resides on the client. DB2 12 supports an internal format for SAP, as well as delimited and spanned format data, including LOB data.

As well as improving usability, DRDA Fast Load also provides the potential for better performance. The operation to transmit data across the network is overlapped with the operation to load data into DB2 by streaming data to the load process in continuous blocks. Measure results so far show this solution to be as fast as the DB2 `LOAD` utility or even faster.

This solution is applicable to a number of cloud use cases. IBM plans to use this feature in the future for fast write from Spark.

Enhanced MERGE support

Initial support for the `MERGE` statement was delivered with DB2 for z/OS Version 9. However, the functionality was limited to `UPDATE` and `INSERT`, and only one of each. In addition, the application programmer had to code host variable column arrays to provide multiple rows of input data.

In DB2 12, the `MERGE` statement is greatly enhanced and is aligned with the behavior defined in the SQL Standard and with the other members of the DB2 family.

The source data can now be a table-reference. Multiple `MATCHED` clauses are allowed, and extra predicates can be coded with the `MATCHED` and `NOT MATCHED` clauses. The `DELETE` operation is supported, and `IGNORE` and `SIGNAL` are supported.

SQL pagination

With the growth of web and mobile applications, application developers are looking for more efficient ways to develop applications that perform well. One of the challenges that application programmers face is writing efficient SQL for paging through results sets. To address this problem, DB2 10 introduced a new access path called *range list access*.

This feature, known as SQL pagination, has been very successful. However, DB2 12 takes it a stage further by providing easier and cleaner syntax to support SQL Pagination, and introduces the OFFSET n and LIMIT keywords. OFFSET n allows you to return rows starting with the 'nth' row of the result set, and LIMIT is alternative syntax to FETCH FIRST and FETCH NEXT.

There are two kinds of SQL pagination, the first of which is numeric-based pagination. In the example below, all the columns are selected from the sample table, but the programmer has specified an offset into the result set (OFFSET 10 ROWS), and the number of rows to be retrieved (FETCH FIRST 10 ROWS ONLY).

```
SELECT * FROM table1 OFFSET 10 ROWS FETCH FIRST 10 ROWS ONLY
```

The second kind of SQL pagination is data-dependent pagination. Data-dependent pagination performs much better than numeric pagination, especially when the offset into the result set is large, because with numeric pagination, DB2 must discard many rows before returning rows to the application.

Before DB2 12, programmers had to use the existing syntax, but the new syntax is provided to make the SQL less cumbersome to code.

Existing syntax:

```
WHERE (LASTNAME = 'SMITH' AND FIRSTNAME >= 'JOHN') OR (LASTNAME > 'SMITH')
```

New equivalent syntax:

```
WHERE (LASTNAME, FIRSTNAME) > ('SMITH', 'JOHN')
```

Piecemeal DELETE

A new feature called Piecemeal DELETE allows interim commits for applications deleting many rows to mitigate the locking and logging issues that often arise when many rows are affected by a single DELETE statement. Consider, for example, the following simple statement, which could potentially delete many millions of rows, taking many locks without a commit, and performing a very large amount of logging in a short time:

```
DELETE FROM T1 WHERE C1 > 7
```

Piecemeal DELETE mitigates these problems by allowing the application to delete a subset of the target rows in one commit scope. The application can then iteratively cycle round the process of deleting a subset of the rows, issuing a commit, deleting a further subset of rows, and so on, until all the target rows have been deleted. This process is likely to reduce the amount of work DB2 must perform if the application abnormally terminates before issuing a commit, and reduce the number of UNDO-REDO log records DB2 must write during the roll-back process.

To enable this feature, DB2 12 supports a select as the target of a DELETE statement, where the fullselect allows for FETCH FIRST n ROWS ONLY to be specified, as in this example:

```
DELETE FROM (SELECT * FROM T1 WHERE C1 > 7 FETCH FIRST 5000 ROWS ONLY)
```

This is an alternative to the existing tactical solution deployed by some customers to call an AUTONOMOUS SQL PL stored procedure.

SQL PL

The SQL Procedural Language (SQL PL) was introduced as far back as DB2 for z/OS Version 7, and was significantly enhanced in DB2 9 with support for native SQL stored procedures that run inside the DB2 engine. SQL PL has been further improved in subsequent releases, and DB2 12 introduces another broad set of enhancements.

Richer capability is now available to the programmer in the body of an SQL PL trigger. SQL PL control statements such as IF-THEN-ELSE, LOOP, and REPEAT are now allowed. A wider variety of SQL statements, including dynamic SQL, variables, and handlers are also available. These improvements all provide easier porting of triggers from other database management systems (DBMSs).

Triggers also benefit from VERSION and DEBUGGER support. The VERSION support provides a much better way to change a trigger without having to DROP it and then CREATE it, and means that it is now possible to change a trigger online and maintain trigger activation order.

However, this richer capability means that there is some performance overhead compared to an equivalent traditional trigger.

SQL PL enhancements beyond triggers (for example, in native SQL PL stored procedures and in UDFs) now includes support for constants. The SQL PL source can be obfuscated. Dynamic SQL is allowed in SQL PL UDFs and stored procedures, and there is now DBMS_OUTPUT for UDF tracing. This last enhancement has been retro-fitted to DB2 11.

SQL enhancements

In addition to these major enhancements, DB2 12 delivers a long list of incremental SQL changes to further improve application enablement.

DB2 now supports simple VALUES by using dynamic SQL, providing the ability to access a SEQUENCE value (such as the NEXT or PREVIOUS value of a SEQUENCE). This enhancement has been retrofitted to both DB2 10 and DB2 11.

Some performance enhancements were added for the JDBC and ODBC Type 2 connections, which are non-network (local) connections from an application running on the same LPAR as DB2 for z/OS.

The ODBC driver has been enhanced in a number of ways:

- ▶ It now supports the TIMESTAMP WITH TIME ZONE data type.
- ▶ It supports multi-context with using ASSOCIATE and DISSOCIATE THREAD, which is the ability to establish multiple connections to a DB2 for z/OS data source, each associated with its own thread.
- ▶ It supports the ability to preserve the global dynamic statement cache after a rollback, avoiding the need for an expensive full PREPARE of the cached SQL statements.

DB2 now supports prepareAttribute literal replacement as a BIND option, which is the ability to specify whether literals should be replaced with literal replacement markers when dynamic SQL statements are prepared and when searching the dynamic statement cache for a matching statement.

Where possible, views and UDFs are kept intact when DDL is run against the underlying tables, including a rename of the table. DB2 12 increases the maximum number of tables that can be referenced in view, UDF, or SQL statement.

The following are some new built-in functions (BIFs):

- ▶ HEX2BIN
- ▶ MEDIAN, PERCENTILE_CONT, PERCENTILE_DISC
- ▶ COUNT(DISTINCT x)
- ▶ GENERATE_UNIQUE (new optional length parm)
- ▶ HASH functions (CRC32, MD5, SHA1, SHA256)

DB2 11 added support allowing binary Unicode columns in EBCDIC tables, and DB2 12 extends this by introducing support for byte-based Unicode columns in EBCDIC tables. A byte-based Unicode column in an EBCDIC table is recorded in the catalog as a character or graphic string column, depending on the CCSID. Many of the restrictions on the use of a binary Unicode column in an EBCDIC table that were implemented with DN1723, do not apply for a new byte-based Unicode column in an EBCDIC table.

There are IBM MQ UDF enhancements to allow the IBM MQ message header to be passed to the IBM MQ Listener stored procedure.

DB2 10 introduced built in support for temporal data, in the form of system temporal, business temporal, and bi-temporal features. These features were enhanced in DB2 11, and DB2 12 introduces further enhancements:

- ▶ Some auditing enhancements have been retro-fitted to DB2 11
- ▶ Inclusive-inclusive support for business time period specifications in SQL DML
- ▶ Logical transaction support for system time
- ▶ Temporal Referential Integrity (RI)

There is APPLCOMPAT support in DB2 12 so that the system migration is not dependent on making application changes to deal with release incompatibilities.

A number of enhancements to the Global Variable support were introduced in DB2 11:

- ▶ It is possible to define Global Variables with the array and LOB data types.
- ▶ Global Variables can be a target in a FETCH statement.
- ▶ Programmers can use Global Variables in EXECUTE and OPEN statements.

XML improvements

Native support for XML in the DB2 engine, IBM pureXML®, was introduced in DB2 Version 9, and has been enhanced in the subsequent releases. DB2 12 continues this strategy with a further set of improvements:

- ▶ DB2 development has tried to improve the performance of XML queries by choosing optimal access paths. Our measurements have seen up to a 76% reduction in DB2 accounting Class 1 time and 77% in Class 2 time.
- ▶ IBM has improved the performance of the XMLTable function that conducts the pivot-like operation to XML data with a name-value pair pattern. We have measured up to a 90% reduction in Class 2 CPU time using a customer's XML data and their queries operating on that data.
- ▶ IBM has improved both performance and developer productivity by allowing multiple updates inside a single invocation of an XMLModify function. We have measured up to a 90% reduction in elapsed time and up to a 97% reduction in CPU time when compared to the previously available functionality, which is performing the updates one at a time through the XMLModify function.
- ▶ A new XSLTRANSFORM function allows DB2 XML Extender users to change to pureXML.

JSON support

JSON support was introduced in DB2 11. Because this support is still fairly new to many customers, this section reviews the existing JSON support before describing the improvements in DB2 12.

The first goal of DB2 for z/OS development was to give the developers the ability to store data from web and mobile applications in its native form. This feature is very important because many web applications use JSON for storing and exchanging information. In addition, JSON is often the preferred data format for mobile application back-end systems.

The second goal was to allow customer application deployment teams to move applications from development to production in virtually no time, by allowing them to create and deploy flexible JSON schemas. This feature reduces the dependency of application developers and deployment teams on IT because there is no need to pre-determine schemas or to create or modify tables. Therefore, this process is ideal for agile, rapid development, and continuous integration.

DB2 provides two ways for working with JSON:

- ▶ By using the JSON API provided with the Java driver
- ▶ By using the SQL extensions introduced in DB2 11, with an enhancement in DB2 12

The DB2 11 JSON_VAL function is used to extract JSON values into SQL data types from Binary JSON (BSON), according to the following syntax:

```
>>-JSON_VAL(--json-value--,--search-string--,--result-type--)-----><
```

In DB2 12, the requirement that the first parameter must be a binary large object (BLOB) column is removed. This enhancement has been retro-fitted to DB2 11 in APAR PI39003. So now, the function supports more options as the first parameter:

- ▶ A View column
- ▶ A CASE expression
- ▶ A table expression with union all
- ▶ A trigger transition variable
- ▶ An SQL PL variable or parameter

Reliability, availability, and scalability

DB2 Development set themselves a number of reliability, availability, scalability, and security goals for DB2 Version 12. These goals are intended to meet requirements raised by our customers in these areas, among others:

- ▶ Easier usability and manageability
- ▶ Increased application availability
- ▶ Improved DBA productivity
- ▶ Better scalability to accommodate larger data volumes
- ▶ Easier compliance with governmental and business regulations, especially in the area of security

In brief, the following are the goals:

- ▶ Relief for table scalability limits because customers need to be able to store more data in a single table, both for business and regulatory compliance reasons.
- ▶ Simplified large table management, as IBM and our customers both anticipate that tables will get larger and larger in the future.
- ▶ Improved availability, by increasing the cases where maintenance can be performed on tables without affecting application availability because customers need to have seamlessly, continuously available services.

- ▶ Improved support for agile schemas (that is, more online schema changes). With applications changing more frequently during agile development to meet the demands of making new business function available at a higher and higher pace, customers need the ability to change the schema more often and non-disruptively.
- ▶ Security and compliance improvements to make it easier to comply with existing and new governmental and industry regulations.
- ▶ A streamlined migration process to reduce the size and impact of migrating to the new release of DB2.
- ▶ Improved utility performance to increase availability and enhance usability.

This section describes how DB2 12 meets those goals.

Lifting partition limits

Listing partition limits is a major item in DB2 12, and addresses requirements raised by many customers.

Currently, the maximum number of partitions that can be defined for a table depends on the DSSIZE (the VSAM data set size) and the page size (4 KB, 8 KB, 16 KB or 32 KB). There are some complicated rules based on DSSIZE and page size that the DBA has to work through to understand the limits on number of partitions and on total table size. For example, if the DSSIZE is 256 GB, and the page size is 4 KB, then the maximum number of partitions that can be defined is 64.

Running out of space in a partition causes an application outage. When altering DSSIZE to increase the partition size, the REORG utility must run against the entire table space. This procedure is cumbersome, especially for large tables. For some customers, time constraints and available work-file disk space make it difficult for the utility to run to successful completion.

The DSSIZE attribute is set at the table space level, and not at the partition level. All partitions inherit the same DSSIZE set at the table space level, so you cannot have differing partition sizes. REORG REBALANCE (to redistribute the data across the partitions by changing the partition limit keys) must be run against multiple partitions.

Another serious limitation for some customers is that the maximum table size is limited to 16 TB.

DB2 12 introduces a new UTS PBR (partition by range) table space structure called 'UTS PBR RPN' by introducing a new table space attribute. RPN stands for *Use relative page number*. This is not a new table space type, but is instead an optional table space attribute.

The solution involves using relative page numbers instead of absolute page numbers to locate pages and rows. The goals here are simplicity (avoiding the use of a complicated formula to calculate the maximum table size and number of partitions), usability, availability, and scalability. Most importantly, this solution removes the restrictive dependency between number of partitions and the partition size.

Table spaces with the relative page number attribute have 7-byte RIDs, with 2 bytes for the partition number and 5 bytes for the page number in the partition. The partition number is now only stored in the partition header page. Using the RPN enables the customer to store up to 256 trillion rows in a single table.

The REORG mapping table format has been changed to support the 7-byte RID, but this is optional until new function has been enabled.

The solution delivers a number of significant benefits:

- ▶ DB2 can now support partition sizes of up to 1 TB.
- ▶ The maximum table size is increased from 16 TB (assuming a 4 KB page size) to 4 PB (four petabytes or 4096 terabytes) and is designed for even larger sizes.
- ▶ There is still a limit of 4096 partitions, but increasing this is possible in the future.
- ▶ The size limit for non-partitioning indexes (NPIs) remains unchanged at 128 TB.
- ▶ Increasing the DSSIZE is supported at the partition-level. This is an immediate ALTER and no outage is involved. This represents a major step forward in terms of availability. However, an ALTER to decrease DSSIZE is a pending change at the table space level, which is the existing (pre-DB2 12) behavior.
- ▶ The DSSIZE attribute can be specified for indexes.

What do customers need to do to use UTS PBR RPN, both in terms of creating table spaces with relative page numbering and of converting existing table spaces to use relative page numbering? What are the considerations customers need to take into account in terms of implementation and in terms of the changes required in DB2 to support this exciting new feature?

A new system parameter, `PAGESET_PAGENUM`, controls whether relative page numbering or absolute page numbering is used when you create a new range partitioned table space. There is also a new table space attribute, `PAGENUM`, which can be used on both `CREATE` and `ALTER TABLESPACE` to specify whether the table space will use relative or absolute page numbering. The two available options are simply `RELATIVE` and `ABSOLUTE`. A conversion from absolute page numbering to relative page numbering is a pending alter and requires a tablespace-level online REORG to materialize the change.

It is important to remember the points made earlier about altering the DSSIZE so that customers do not find themselves facing an unexpected challenge to application availability. Once converted to relative page numbering, an online alter to increase DSSIZE is immediate, non-disruptive, and requires no REORG. However, an online alter to decrease DSSIZE is a pending alter recorded in the DB2 catalog and requires a tablespace-level REORG to materialize the change.

To support this change, the log record format is changed to support 7-byte RIDs. A number of changes have been made to improve serviceability. For example, the partition number is now included in the log records, which makes it a lot easier to filter log records when analyzing them. To make the log records more useful when reading them, `DSN1LOGP` formats the partition number explicitly.

The new log record format is written immediately on starting DB2 12 for the first time, and there is fall-back toleration support in DB2 11 for the new 7-byte RIDs recorded in the log records. DB2 11 will continue to process log records as it always has, irrespective of whether you have converted your objects to use the 10-byte extended RBA/LRSN introduced in DB2 11.

This change in the log record format is not just for the UTS PBR RPN page sets, but is for all types of page sets. This change means that there will be an increase in the log record size, of approximately 20 bytes for table spaces, and 28 bytes for index spaces.

Online schema

Online schema evolution has been a constant theme of recent DB2 for z/OS releases. The changes help to meet customer requirements for improved application service availability, and DB2 12 continues the evolutionary trend with a number of important enhancements. This section covers the following online schema topics:

- ▶ Insert Partition (the ability to insert a new partition into the middle of a partitioned table space)
- ▶ The option to defer column-level ALTERs to avoid indexes going into the RBDP status
- ▶ Online deferred ALTER INDEX COMPRESS YES, avoiding an outage when converting an index to use compression

Insert partition

Insert partition is a new feature that is intended to solve a couple of problems face by many customers.

First, large range-partitioned tables often have hot spots (many data rows within one range of partitioning keys, and fewer rows in another range), and rebalancing across the entire set of partitions is a very onerous and expensive process.

Second, the partitioning scheme chosen when the table was originally designed might no longer be optimal.

DB2 12 provides a solution to these problems. However, this solution only applies to UTS PBR table spaces, and not to “classic” partitioned table spaces.

DBAs now have the ability to insert a new partition with specified limit-key, with the following syntax:

```
ALTER TABLE ADD PARTITION ENDING AT xxx
```

This is treated as a deferred ALTER and recorded in the DB2 catalog. The online materializing REORG splits an existing partition, and distributes the data rows between the new and old partitions. Only the affected partitions need to be reorganized. Be aware that there is no point-in-time (PIT) recovery capability, to a time before the materializing REORG.

Deferred column-level ALTERs

The objective of this feature is to address issues that are faced by some customers for some column-level alters. First, some column-level alters result in an impact on application availability, such as when the column-level alter causes one or indexes to be placed in the restrictive RBDP state. Second, immediate alters conflict with pending alters, and additional REORGs are required to materialize the pending alters before issuing the immediate alter to avoid SQL DDL failures.

The solution to both of these problems is to provide a new option to allow existing immediate alters to become pending alters. This option applies to UTS table spaces only, both PBR and PBG. When this option is specified, all pending alters are accumulated and materialized through a tablespace-level online REORG.

The option is activated or deactivated by using a new system parameter, DDL_MATERIALIZATION. Specifying ALWAYS_IMMEDIATE causes existing, pre-DB2 12 behavior to continue for column-level alters, such as altering the data type of a column continues to be an immediate alter. Specifying ALWAYS_PENDING causes those currently immediate alters to be converted to pending alters. This improvement has the great

advantage in that those types of ALTER COLUMN type avoid indexes being placed in the RBDP state.

Online deferred ALTER INDEX COMPRESS YES

Before DB2 12, if you run SQL DDL ALTER INDEX COMPRESS YES, this would cause an outage because the index would be placed in RBDP (rebuild pending) and was unavailable until the RBDP condition was resolved by rebuilding the index. DB2 12 introduces online deferred ALTER INDEX COMPRESS YES, which allows you to alter an index to use compression without causing an outage.

Security and general enhancements

DB2 12 includes a number of fairly general enhancements, including some important security enhancements, which you will find useful:

- ▶ Large object (LOB) (large object) compression has been enhanced. This feature compresses the out-of-line portion of the LOB, which is the portion of the LOB stored in an auxiliary table space. It does this in 1 MB chunks. LOB requires the zEDC hardware feature.
- ▶ Improved LOB handling for ISO(UR) (uncommitted read) queries to avoid SQLCODE +100. Before DB2 12, ISO(UR) queries could get SQLCODE +100 if the LOB they are reading is in process of being updated because of the way LOB locking worked.
- ▶ There are three new important features in the area of security:
 - A new SQL statement, TRANSFER OWNERSHIP, that allows you to transfer the ownership of database objects to a different AUTHID or database ROLE. When you use this new statement, the ownership of any implicitly created objects is also transferred automatically.
 - The capability to install a new DB2 subsystem, or upgrade an existing subsystem to a new release of DB2, without the need for INSTALL SYSADM (installation system administrator) authority. Instead, you can use INSTALL SYSOPER (the DB2 system operator privilege) to limit access to application data.
 - The new UNLOAD privilege to restrict use of the UNLOAD utility. When system parameter AUTH_COMPATIBILITY is set to SELECT_FOR_UNLOAD, DB2 checks whether the user has the SELECT privilege on the table. Otherwise, DB2 checks to see whether the user has the UNLOAD privilege on the table. There is also a new IFCID, 404, which records whether the SELECT privilege is being used for UNLOAD. This new IFCID has been retrofitted to DB2 11 through the service stream.
- ▶ DB2 now supports long, camel-cased DBRM names, which are used by some application management tools.
- ▶ IFCID 306 log records are now returned in the correct version. Previously, if a column was added to a table, or the data type of a column was changed, customers needed to run a REORG to get the log records in the correct format. With this enhancement, you do not need to run the REORG, avoiding any impact on the availability of the application service.

Utilities

The functionality and performance of the DB2 utilities is important for DB2 customers because they affect availability, recovery, and the total cost of ownership. Therefore, DB2 development has invested a great deal in the incremental improvement of the utilities with each DB2 release. This commitment is continued with DB2 12, delivering even greater value. The utility functions covered are REORG, LOAD, backup and recovery, and RUNSTATS.

REORG

A number of enhancements to the REORG utility are delivered in DB2 12, the first of which is improved IBM FlashCopy® management. Previously, if for some reason the FlashCopy image copy process fails, the REORG utility would terminate with return code 4, but with the object in the copy pending state, blocking INSERT, UPDATE, and DELETE. In DB2 12, a REORG utility with only a data set level FlashCopy inline image copy (that is, with no traditional sequential image copies) now causes REORG to fail with return code 8 if the FlashCopy cannot be taken. This change makes it easier to detect the exception, and to take action to eliminate or minimize any application service availability issues.

Partition-level UTS PBG REORGs are improved, giving the REORG utility the capability to create additional PBG partitions for overflow rows that can no longer fit back into the partition being reorganized.

DB2 12 avoids setting the copy-pending state on any LOB table spaces during the REORG of UTS PBG table spaces when new PBG partitions are created during the log apply phase of the REORG utility.

DB2 11 provided an option at the system parameter level (REORG_DROP_PBG_PARTS) to prune any empty PBG partitions resulting from a REORG. DB2 12 improves usability by providing REORG-level management of the delete of UTS PBG partitions. A new keyword, DROP_PART, is added to the REORG utility to control the pruning of empty PBG partitions when changing the system parameter is not feasible.

A COMPRESSRATIO catalog column has been added, which is maintained by the REORG and LOAD REPLACE utilities, independently of the KEEPDICTIONARY option and also by the RUNSTATS utility. This new statistic provides the DB2 utilities with the capability to calculate the sort-work data set sizes more accurately.

There is now up to an additional 17% offload to zIIP engines, with the reload phase of REORG now eligible for zIIP offload. This is targeted at reducing TCO.

As discussed earlier, there is a new mapping table format to support the 7-byte RIDs needed for UTS PBR RPN (relative page numbering). You can continue to use the DB2 11 format before activating new functions in DB2 12. However, you must have converted your mapping tables to the new format before you activate new function. Otherwise, DB2 automatically allocates a new mapping table in the correct format.

You can now run REORG against objects in a read-only (RO) state. To be more precise, REORG with any SHRLEVEL (CHANGE, REFERENCE, NONE) can be run against any table space or index space in the RO state. This enhancement was retrofitted back to DB2 11 in the service stream by using APAR PI46774.

There is a small but very useful enhancement to cause REORG to display claimer information on each drain failure, not just the last retry. This change makes it much easier to diagnose claim-drain contention and to take early action to either terminate the utility or take steps to allow it to complete successfully.

LOAD

As with REORG, customers will benefit from a number of enhancements to the LOAD utility.

The first improvement makes it much more efficient and cost-effective to use LOAD PART REPLACE with dummy input against an already empty PBR partition. NPI processing has been optimized to avoid scanning all non-partitioned indexes (NPIs) to find index keys for deletion for the target LOAD REPLACE partition if it is already empty. This feature reduces

elapsed time and CPU time significantly, especially if there are many keys for the other logical parts of the NPI. These savings are up to a 99% CPU and a 98% Elapsed Time reduction.

DB2 12 now provides parallel support for LOAD SHRLEVEL CHANGE UTS PBG table spaces. LOAD has been modified to remove the single input parallelism restriction for PBG table spaces for LOAD SHRLEVEL CHANGE. DB2 for z/OS SVL Laboratory measurements recorded up to 76% elapsed time savings, although there is some CPU overhead because of the parallelism. However, the single input parallelism restriction still applies LOAD SHRLEVEL NONE against a UTS PBG table space.

Many customers who use the LOAD utility and who want to reduce TCO welcomes the additional zIIP offload for the LOAD utility, up to 90%. This savings is possible because the RELOAD phase of the LOAD utility is now eligible for zIIP offload. This process includes data conversion and loading of the records into the page set.

Another availability enhancement for the LOAD utility is the option to specify BACKOUT YES for a LOAD RESUME SHRLEVEL NONE execution. Doing so avoids setting the recover-pending (RECP) state when the LOAD REPLACE utility fails. This new option causes LOAD to back-out any rows it has already loaded upon encountering an error without leaving the page set in RECP, such as a data conversion error, a duplicate key error, or violation of a referential integrity (RI) constraint.

DB2 has supported PREFORMAT for a base table space and its index spaces for a long time. DB2 12 introduces PREFORMAT support for auxiliary tables. The PREFORMAT option causes the remaining pages to be preformatted, up to the high-allocated RBA in the table space and index spaces that are associated with the specified table. The preformatting occurs after the data has been loaded and the indexes are built. This process has traditionally been done only on the base table space, but has been extended to support LOB table spaces and auxiliary indexes. However, XML objects are not preformatted. Preformatting improves the elapsed time and CPU time performance of transactions inserting LOB values into empty pages in the LOB table space.

The MAXASSIGNEDVAL column in SYSIBM.SYSSEQUENCES catalog table is now maintained for identity columns. Before DB2 12, this value was maintained if the value was generated by DB2, but not for user provided values on the LOAD utility. This enhancement allows LOAD to maintain the MAXASSIGNEDVAL for user-provided input and also reset the value if a LOAD REPLACE on the table space is run. This procedure eliminates the potential for potential duplicate values for the identity column for subsequent INSERTs.

Customers using UNLOAD utility with isolation uncommitted read (ISO(UR)) can now eliminate any data sharing overhead with the UNLOAD ISO(UR) REGISTER NO option. The new REGISTER YES/NO option is available for UNLOAD SHRLEVEL CHANGE ISO UR to bypass Coupling Facility page registration to avoid data sharing overhead when accessing the data with SHRLEVEL CHANGE. the option also avoids page registration spikes that could affect other applications.

LOAD REPLACE support for the COMPRESSRATIO catalog column is added to gather the average compression ratio at the record level instead of the page saved level. This value is used by the DB2 utilities to calculate the required utility sort-work data set sizes.

Backup and recovery

The backup and recovery capabilities of DB2 are second to none, but with each release of DB2, those capabilities are extended and improved to maximize recoverability, availability, and resilience. DB2 12 further builds on those capabilities with a number of enhancements:

- ▶ PIT (point-in-time) recovery support for UTS PBG table spaces is enhanced to allow recovery of PBG UTS to a previous PIT before a REORG that materialized the following physical pending alters:
 - Page size (involving a change of buffer pool)
 - DSSIZE
 - SEGSIZE
 - MEMBER CLUSTER
- ▶ A new option called FLASHCOPY_PPRCP for the RESTORE and RECOVER utilities allows customers to specify or override the Preserve Mirror option for PPRC (synchronous disk mirroring) on the utility statement. Before DB2 12, the RESTORE SYSTEM utility used the DFSMSshm default, and the RECOVER utility used the system parameter FLASHCOPY_PPRC.
- ▶ In the area of PIT recovery, DB2 12 introduces an option to skip PIT recovery for non-updated page sets, which are objects that were not updated after the PIT recovery point. In other words, RECOVER does not run the RESTORE phase, and there is no need for the LOGAPPLY phase. This is the new default behavior for the RECOVER utility when recovering to a previous point in time. The reasoning behind this is that the data in those objects at the target PIT and at the current PIT is identical, so there is no need to incur the cost of recovering them. This avoids unnecessary CPU and I/O resource consumption and reduces elapsed time. This default behavior can be overridden by specifying SCOPE(ALL) on the RECOVER utility.
- ▶ The BACKUP SYSTEM and RESTORE SYSTEM utilities now support multiple copy pools. This feature allows the customer to use additional copy pools to keep a daily “golden copy” or to keep a backup for critical events. This is a very practical proposition for customers who do not use incremental FlashCopy - for those customers who do use incremental FlashCopy, then there are some additional operational considerations.
- ▶ To aid in utility diagnosis and improve serviceability, DFSMSshm messages are now included in the utility job output for the BACKUP SYSTEM and RESTORE SYSTEM utilities. With the DFSMSshm and DFSMSdss messages included in the DB2 utility job output, customers no longer must look in the separate DFSMSshm job logs for important diagnostic messages. This feature is available starting with z/OS V2.2.
- ▶ A new option enables the COPY utility to specify FASTREPLICATION(REQUIRED) when using DFSMSshm services to take a FlashCopy image copy. To take a FlashCopy image copy, the object being copied and the target FlashCopy image copy data set must be in the same DASD box. DFSMSdss allows three options when taking a FlashCopy of a data set: NONE, PREFERRED, and REQUIRED.

If you specify PREFERRED and a FlashCopy cannot be taken, then a standard DSS copy is taken instead. If you have a hard requirement to take a FlashCopy image copy, then the COPY utility must specify FASTREPLICATION(REQUIRED) to make sure that if it cannot take a FlashCopy image copy, the utility will fail.

This change is implemented using a new system parameter, COPY_FASTREPLICATION REQIPREFINONE (required, preferred, or none), which causes the COPY utility to specify the appropriate option when it starts DFSMSshm services. Previously, COPY defaulted to FASTREPLICATION PREFERRED.

RUNSTATS

DB2 12 includes several RUNSTATS enhancements:

- ▶ The usability of RUNSTATS profiles, which were introduced in DB2 10, has been improved. DB2 12 supports RUNSTATS profile when collecting inline statistics as part of the REORG and LOAD utilities.
- ▶ Collecting COLGROUP inline statistics as part of the LOAD utility was introduced in DB2 11, but was not supported when LOAD option PARALLEL was specified. This restriction is removed in DB2 12 and avoids the need for a stand-alone RUNSTATS to collect COLGROUP statistics after LOAD PARALLEL.
- ▶ COLGROUP statistics performance is improved, with a reduction of up to 25% CPU time and up to 15% elapsed time. This is achieved when a COLGROUP column specification matches with a specified INDEX. In this case, the duplicate COLGROUP can be safely ignored because the statistics are collected when the index statistics are collected.
- ▶ DB2 12 introduces a new INVALIDATECACHE option to control whether RUNSTATS invalidates statements in the DSC. Previously, the RUNSTATS utility would invalidate any statements in the DSC that referenced the target object. Be aware that the default behavior has changed: RUNSTATS will not invalidate prepared statements in the DSC.

Other utility enhancements

The DSNUTILU stored procedure, which can be used to run utilities from an application program, is supplemented by a new stored procedure, DSNUTILV. This procedure supports CLOB input to utility statement input more than 32 MB long. In fact, because the DSNUTILV UTSTMT utility statement parameter is now a CLOB instead of a VARCHAR, DB2 now supports a utility statement of up to 2 GB long. For existing applications modified to call DSNUTILV but still passing a VARCHAR, DB2 casts the VARCHAR to CLOB.

The DSNACCOX stored procedure is a sample stored procedure delivered with the DB2 product that can provide recommendations on housekeeping based on real-time statistics (RTS). Based on customer feedback and requirements, some changes have been made to the stored procedure to avoid unnecessarily recommending REORG. This adjustment is done by changing the default setting to OFF when recommending REORG based on the number of inserts and pseudo deletes since the last time REORG was run for the target object. The criteria for recommending a REORG based on REORGLASTTIME, LOADRLASTTIME, or REBUILDLASTTIME being NULL are removed.

To avoid scheduling conflicts, there is improved concurrency for the MODIFY, COPYTOCOPY, and MERGECOPY utilities. This change allows these utilities to run concurrently alongside exclusive utilities like LOAD and REORG running on the same target objects.

Data sharing improvements

Large DB2 customers always anticipate data sharing improvements with a new DB2 release, and DB2 12 is no exception.

Improved support is provided for global transactions. A global transaction, sometimes referred to as an XA transaction, is a unit of work that involves work across multiple DB2 subsystems, and can have many branches. Prior DB2 releases provided restricted XA transaction support. DB2 12 removes the need for a client gateway with Sysplex workload balancing enabled. In DB2 12, a global transaction is tied to single DB2 member, from the start to the end of the transaction. Seamless failover support is provided for global transactions, and is on by default.

DDF session data can now be shared across the DB2 data sharing group. Previously, a client configured with Sysplex workload balancing had to maintain session information to support transaction pooling, client reroute, or both. In DB2 12, Sysplex workload balancing-enabled clients pass a session token to DB2. DB2 then maintains session data for that application across the data sharing group. That session information includes the values for Global Variables and Special Registers.

DB2 12 also delivers a number of data sharing performance improvements. Some of these have been already described, such as more granular commit LSN, which provides improved lock avoidance checking and a reduced number of CF lock requests. In-memory indexes can reduce GETPAGE requests and requests to the CF Group buffer pools. The improved insert space search algorithm for unclustered insert can avoid P-lock contention and streamline insert processing. And RUNSTATS and UNLOAD run with ISO(UR) avoid CF page registration.

There are two short but important items related to recovery:

- ▶ A new data sharing peer recovery option has been developed for IBM GDPS® automation that is applicable to customers who do not already have automation in place.
- ▶ Failed automatic LPL and GRECP recovery requests are automatically retried.

DB2 12 will use asynchronous CF Lock duplexing. This method reduces the overhead for system-managed duplexing of the CF LOCK1 and SCA structures. The updates of the secondary structures updates are performed asynchronously with respect to the updates of the primary structures.

To ensure data integrity, the DB2 lock manager periodically syncs up with z/OS to make sure that all modify locks have been “hardened” in the secondary lock structure before the corresponding undo/redo record for the update is written to the DB2 active log on DASD. This process increases the practical distance for multi-site Sysplex and data sharing operations when using system-managed duplexing of the CF LOCK1 and SCA structures.

This feature is not yet enabled, but is planned for delivery in the fourth quarter of 2016. It will require a new z/OS 2.2 APAR and a new leaving of the CF microcode.

Figure 8 shows how this feature works. At step 1, XES the system lock manager, receives a global lock request from the local DB2 and IRLM. At step 2, the request is sent to the coupling facility where the primary lock structure is located (the primary CF). At step 3, the system runs the request and sends a response back to the sending z/OS system. Then, at step 4, the local XES system lock manager sends a response back to DB2 and IRLM.

At step 5, the primary CF sends a request to the CF containing the secondary lock structure (the secondary CF). At step 6, the secondary CF runs all the requests coming from the primary CF, in strict time order. Periodically, the DB2 log manager queries the secondary CF to make sure that modify lock requests have been “hardened” into the secondary lock structure before the corresponding UNDO-REDO records are written to the DB2 active log.

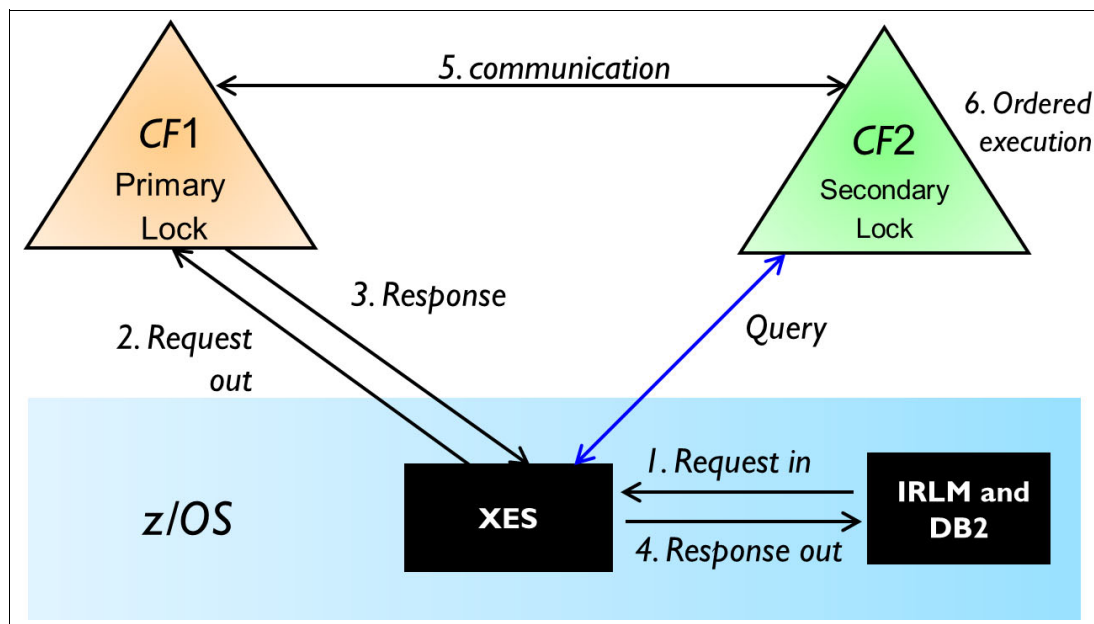


Figure 8 Asynchronous CF lock structure duplexing

Upgrade and prerequisites

Upgrade to DB2 12 is the last major topic in this paper. It covers upgrade prerequisites, both hardware and operating system, the upgrade process itself, including changes to the catalog, and improvements to online upgrade.

Upgrade prerequisites: Hardware and operating system

DB2 12 requires a z196 or later processor, running z/OS V2.1, or later. DB2 12 will probably require increased real storage for a workload compared to DB2 11.

In terms of software requirements, you need to be running z/OS V2.1 Base Services (5650-ZOS) or later, DFSMS V2.1 or later, IBM Language Environment® Base Services, and z/OS V2.1 Security Server (IBM RACF®) or later. You also need to run IRLM Version 2 Release 3 (delivered with DB2 12). For more information, see the [DB2 12 announcement letter](#).

Upgrade and the catalog

For DB2 Version 12 there is a single phase upgrade process, which is different from the preceding versions. There is no enabling-new-function mode (ENFM) phase, which has been eliminated. New function is activated through new command:

`-ACTIVATE NEW FUNCTION`

The rules regarding APPLCOMPAT and fall-back continue to apply.

Before upgrading to DB2 12, you must have converted the BSDS to the format supporting the extended 10-byte log RBA/LRSN.

No packages bound before DB2 10 are supported in DB2 12. This is because we have removed support for the 31-bit package runtime. Doing this reduces the number of code paths and delivers a performance improvement.

Basic Row Format (BRF) is deprecated. Existing data sets in BRF format are still supported, but the system parameter and the associated REORG options are removed.

DB2 12 supports temporal RTS tables. These are defined in the DB2 catalog, but enablement is optional.

Online upgrade improvements

There are some welcome online migration improvements. DB2 automatically pauses statistics externalization during migration to reduce the possibility of lock contention on the DB2 catalog. In addition, the duration of locks on the catalog and directory is reduced. Previously, these locks would have affected online upgrade, particularly in the area of catalog REORGs.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Gareth Jones has worked in IT since 1985. Until 2000, he was an IBM client with experience as a systems programmer and DBA. He now works in DB2 for z/OS development as a member of the SWAT Team, which is led by John Campbell. He has worked with many customers around the world to help them succeed in their use of DB2. Gareth has written several technical papers and presented at many conferences and group meetings. Gareth is also known as Gareth Copplestone-Jones.

John Campbell has over 30 years' experience working with DB2 for z/OS reports to the Director for z/OS Development at the IBM Silicon Valley Lab. He has extensive experience with DB2 in terms of systems, database, and applications design. He specializes in design for high performance and data sharing. He is one of IBM's foremost authorities for implementing high end database / transaction processing applications. John has extensive experience working with and advising large enterprises including many of the top Fortune 100 companies. John is passionate about sharing his proven industry knowledge and expertise, which was one of the drivers of writing this whitepaper. You can follow John on Twitter @GURUDB2.

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

Ernest A. Keenan
International Technical Support Organization

John Campbell
IBM Distinguished Engineer, IBM Silicon Valley Lab

Now you can become a published author, too!

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

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

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

Notices

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

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

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

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

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

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

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

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

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

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

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

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


COPYRIGHT LICENSE:

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

Trademarks

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

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

CICS®	IMS™	WebSphere®
DB2®	Language Environment®	z Systems®
Easy Tier®	pureXML®	z/OS®
FlashCopy®	RACF®	z13®
GDPS®	Redbooks®	
IBM®	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

Netezza, and TwinFin are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

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

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



REDP-5444-00

ISBN 073845611X

Printed in U.S.A.

Get connected

