



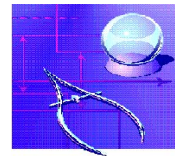
What's New in DB2 V8 for Database/System Administrators

*Dr. Jim Teng
IBM Distinguished Engineer*



**DB2 for z/OS Technical Conference
Sept 05-07, 2007 - Shanghai, China
Sept 10-12, 2007 - Beijing, China**

Disclaimers & Trademarks*



Information in this presentation about IBM's future plans reflect current thinking and is subject to change at IBM's business discretion. You should not rely on such information to make business plans. Any discussion of OEM products is based upon information which has been publicly available and is subject to change. The opinions expressed are those of the presenter at the time, not necessarily the current opinion and certainly not that of the company.

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States and/or other countries: AIX, AS/400, DATABASE 2, DB2*, Enterprise Storage Server, ESCON*, IBM, iSeries, Lotus, NOTES, OS/400, pSeries, RISC, WebSphere, xSeries, z/Architecture, z/OS, zSeries, System p, System I, System z

The following terms are trademarks or registered trademarks of the Microsoft Corporation in the United States and/or other countries: MICROSOFT, WINDOWS, ODBC

For more copyright & trademark information see ibm.com/legal/copytrade.phtml



Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in the operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. This information may contain 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 the names and addresses used by an actual business enterprise is entirely coincidental.

Agenda

- 64-bit evolution
- Index Improvements
- Table-controlled Partitioned Table
- Online Schema Evolution
- System Level Backup/Recovery



The main objective of this presentation are to:

- Share hard won valuable experience gained from field customer deployment of DB2 for z/OS Version 8
- Provide hints and tips for a successful migration
- Address some of the myths that have already grown up around DB2 for z/OS Version 8
- Provide up to date planning information
- Introduce and discuss some new enhancements to DB2 for z/OS Version 8 that have added since the initial announcement.

Scaling – V8



Image of Earth from Moon, Source: NASA (Public Domain)

Breaking through limitations: scaling

- Virtual Storage 2 GB 2^{31} to 2^{64}
- Partitions 254 to 4096
- Open data sets 32 K to 64 K
- Active logs 31 to 93
- Archive logs 1000 to 10,000
- Maximum table size 16 TB to 128 TB (partitioned, 32K page)



One of the keys to reengineering is breaking through the limits of the current architecture. Increasing some limits improves scalability. Increasing other limits improves productivity, portability & family consistency.

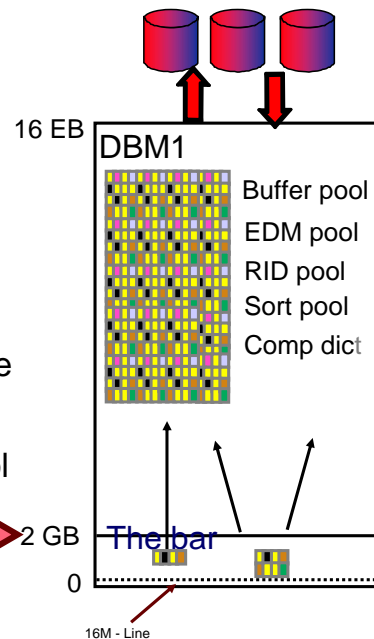
Increasing the amount of virtual storage we can address directly can help with the ability to scale and simplify management for virtual storage. It will require more real memory, but permit increased scalability and availability.

Increasing name sizes & SQL statement lengths makes porting from other DBMS much easier and improves DB2 family compatibility. Increasing the maximum number of partitions helps DB2 scale farther and makes management much easier when you can have one partition per day for 11 years.

Doubling the number of concurrent open data sets provides the needed ability to handle more data sets. This change needs z/OS 1.5. See PQ96189.

64 bit evolution

- zSeries, z/OS, z/Architecture & large real storage
 - 64 bit real storage
 - f* V7 data space advantages
 - Version 8 64 bit virtual
 - f* Improve scalability, availability, ease
 - f* Move above the bar
 - f* Buffer Pools, EDM Pool, Sort Pool
 - f* Rid Pool, Comp Dict
 - f* IRLM locks, PC=YES
 - f* Hiperpool/dataspace no longer needed, no ECSA for locks



The biggest impact of the zSeries architecture on DB2 is the ability to use more memory more effectively. Prior to the zSeries, customers were limited to 2 GB real storage due to the 31-bit addressing of the S/390 architecture. The real storage limit of 2 GB is a leading performance inhibitor for many high end customers. Another performance inhibitor is the 2 GB virtual storage limit for the main DB2 (DBM1) address space. If you have zSeries & OS/390 V2R10 64-bit mode or z/OS, use V6 buffer pools in data spaces, but not otherwise. See V7 Performance Topics red book & the web. See What's New? for V8 use of 64 bit virtual storage. There are more steps as real & virtual memory sizes increase, moving more above the line and above the bar. See the Roadmap, GM13-0076-01 updated June 2002. ibm.com/servers/eserver/zseries/library/whitepapers/gm130076.html

64-bit Buffer Pools

- Max BP size is lifted to 1TB
 - f* Max size of single or summation of all
 - f* The actual maximum = the REAL storage available
 - f* Always allocated above 2GB
 - f* Castout buffers and Buffer Control Blocks are also allocated above 2GB
- Data space pools and Hiperpools are eliminated
 - f* Simplifying DB2 system management
- Migration
 - f* $V8_VPSIZE = V7_VPSIZE + HPSIZE$
 - f* Fallback uses $V7_VPSIZE$ and $HPSIZE$



Long Term Page Fix

- ALTER BUFFERPOOL
 - New option PGFIX(YES)
 - By buffer pool
 - Use where IO rate is high
 - Must have real storage
 - Up to 10% cpu savings
 - Allowed in all modes, e.g. compatibility mode



ALTER BUFFERPOOL has a new option that most customers should use for subsystems which read or write frequently. Recommendation: Alter your DB2 Version 8 buffer pools which have frequent page reads or writes to use PGFIX YES if you have sufficient real storage available for these buffer pools. Fixing the buffer page **once and keeping them fixed** in real storage avoids the processing time that DB2 needs to fix and free pages **each time there is an I/O**. In some cases, this processing time can be as much as 10% for I/O intensive workloads. To use this option, issue the following command:

```
ALTER BPOOL(bpname) VPSIZE(vpsize)  
PGFIX(YES)
```

where *bpname* is the name of the buffer pool and *vpsize* is the size of the virtual pool.

SQL Limitations



Image of Earth from Moon, Source: NASA (Public Domain)

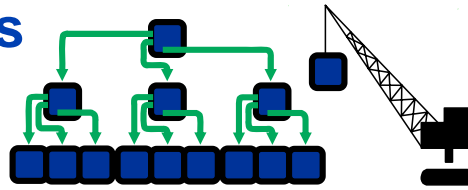
Breaking through limitations - SQL

- Table name sizes 18 to 128
- VIEW & ALIAS names 18 to 128
- Column name sizes 18 to 30
- SQL statement length 32K to 2MB
- Tables in a join 15 to 225
- Character Literals 255 to 32704
- Hex literal digits 255 to 32704
- Predicates 255 to 32704
- Index key 255 to 2000



One of the keys to reengineering is breaking through the limits of the current architecture. Increasing some limits improves scalability. Increasing other limits improves productivity, portability & family consistency. Increasing name sizes & SQL statement lengths makes porting from other DBMS much easier and improves DB2 family compatibility. Increasing the amount of virtual storage allows the longer names, larger SQL statements and increased sizes. DB2 V8 will often require a little more real memory (1% to 10% for large subsystems), but permit increased scalability and availability.

Index Improvements



- Variable length index keys
- Index-only access for varchar data
- Maximum index key 2000 bytes
- Predicates indexable for unlike types
- Backward Index Scan
- Partitioning separate from clustering
- Data-partitioned secondary indexes (DPSI)
- Create index online during select, insert



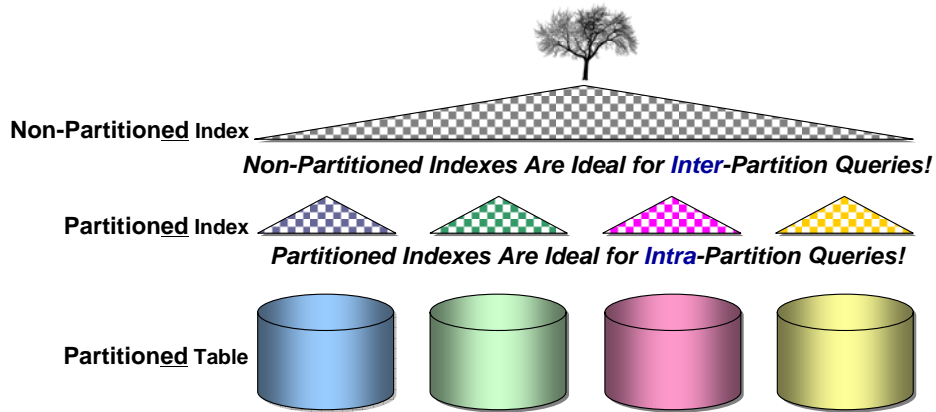
DB2 V8 provides many new opportunities for improving index processing, rebuilding the architecture for indexes.

We are able to use indexes more effectively, reducing the space in variable-length indexes, being able to have index-only access with variable-length data and being able to use the index when the predicates do not match.

In some cases, such as backward index scans or partitioning, we will be able to work as efficiently with one less index. Being able to eliminate an index will improve the insert, delete, LOAD, REORG and update processing.

We have more flexibility in indexes, with longer index keys, the ability to partition secondary indexes and the ability to have more effective clustering.

Non-Partitioned vs. Partitioned Indexes



It's All About the Index Trees!



When dealing with partitioned tables, the indexes can be either partitioned or non-partitioned. Partitioned indexes have a separate index tree for each partition. Non-partitioned indexes have a single index tree over all table partitions. With table-controlled partitioning, you can have as many partitioned indexes as you want.

Data Partitioned Secondary Indexes (DPSIs)

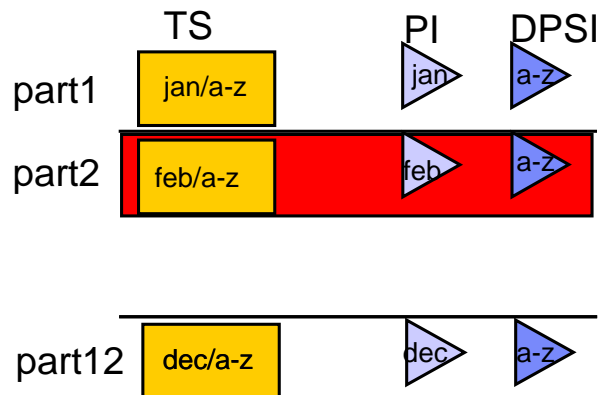
- DPSI = physically partitioned secondary index
 - f* Number of parts in DPSI = Number of parts in table
 - f* Keys in part 'n' of DPSI refer only to rows in part 'n' of table
- 3 kinds of indexes now:
 - f* Partitioning Index (PI).
 - As today, except optional in V8 and may or may not be partitioned
 - f* New Data Partitioned Secondary Index (DPSI).
 - f* Non Partitioned Secondary Index (NPSI) As today's NPI



For a DPSI, the same number of partitions are in the data and in the index. This is a new type of index that does not fit into the old categories. It is a secondary index, but it is partitioned. The partitioning is according to the table. The partitioning data might be columns that are not part of this index.

So the index partitions are partitioned just like the data. This provides many new options for combinations of partitioning and indexes.

Partitioned Table



- Partition data by month (PI is optional!)
- Clustering by id or name (DPSI clustering)
- Ideal for Online Reorg with fast switch, no BUILD2



This is an example of the new style table, with table-based partitioning, rather than index-based partitioning. Note that the data is partitioned by month. An index is not required for the partitioning. Clustering for the data is by the id or name within each partition of the DPSI. This is an ideal organization for online reorg of a single partition. The BUILD2 phase is not required. If the month is not provided, a name search using the DPSI may need to search in every partition.

Data Partitioned Secondary Indexes (DPSIs)

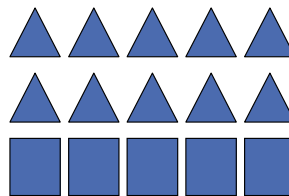
- Benefits include partition independence:

- f* More efficient utility processing
- f* Higher availability
- f* Streamline partition level operations
- f* Potential for lower data sharing overhead



- Potential impact to query performance

- f* Partition key is not specified
- f* Many partitions to search
- f* Not allowed for unique index



DPSI benefits are substantial in terms of the ability to reduce contention on indexes in SQL and utilities that process a range of partitions. The recovery time can be improved by an order of magnitude or more, eliminating the BUILD2 phase of online reorg.

This option does not fit some situations, such as when we must search many partitions or if the index must be unique.

DPSI -Design considerations

- DPSIs are a design choice and may not fit ALL situations
- Query predicates that solely reference columns in DPSIs must probe each partition - this will degrade query performance

- SELECT * FROM table WHERE DPSI_col = 'X'
- SELECT * FROM table WHERE DPSI_col BETWEEN 'X' AND 'Y'
ORDER BY DSPI_col

- If predicates also restricts the query to a single partition, query performance will benefit

- f* "Partition pruning"

- f* Bind time pruning

- P_COL = literal

- f* Runtime pruning with REOPT(VARS)

- P_COL = :hv , ? , special register

- Recommendation:** Code predicates explicitly in your applications to allow for partition pruning when a DPSI exists



Backward index scan enabled

- DB2 will now select an ascending index and use a backward scan to avoid the sort for the descending order
- DB2 will use the descending index to avoid the sort and scan the descending index backwards to provide the ascending order
- To be able to use an index for backward scan,
 - f* Index must be defined on the same columns as ORDER BY and
 - f* Ordering must be exactly opposite of what is requested in ORDER BY.
 - f* i.e., If index defined as DATE DESC, TIME ASC, can do:
 - Forward scan for ORDER BY DATE DESC, TIME ASC
 - Backward scan for ORDER BY DATE ASC, TIME DESC
 - f* But must sort for
 - ORDER BY DATE ASC, TIME ASC or ORDER BY DATE DESC, TIME DESC



Avoid Sort by using Backward Index Scan with ORDER BY

Same Index used

```
SELECT STATUS_DATE, STATUS
FROM ACCT_STAT
WHERE ACCT_NUM = :HV
ORDER BY STATUS_DATE DESC, STATUS_TIME DESC;
```

Backward index scan

For scrollable and non scrollable cursors

```
SELECT STATUS_DATE, STATUS
FROM ACCT_STAT
WHERE ACCT_NUM = :HV
ORDER BY STATUS_DATE ASC, STATUS_TIME ASC;
```

Forward index scan

DB2 optimizer will select an ascending index to provide a descending sort order by traversing the index backwards rather than do a sort

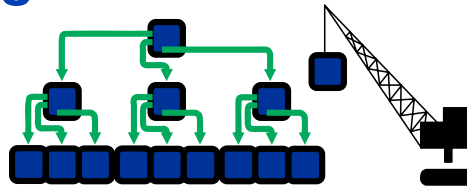
Index on ACCT_STAT is
ACCT_NUM, STATUS_DATE, STATUS_TIME



DB2 will now select an ascending index and use a backward scan to avoid the sort. The reverse case is also true. That is, DB2 will use the descending index to avoid the sort and scan the descending index backwards to provide the ascending order.

To be able to use backward scan, index must be defined on the same columns as ORDER BY and ordering must be exactly opposite of what is requested in ORDER BY. i.e., index defined as DATE DESC, TIME ASC enables: forward scan for ORDER BY DATE DESC, TIME ASC and backward scan ORDER BY DATE ASC, TIME DESC but must sort for: ORDER BY DATE ASC, TIME ASC or ORDER BY DATE DESC, TIME DESC.

Indexable Predicates



- **Predicates indexable for unlike types**
 - **Column is decimal; Host variable is float**
 - **Column char(3); Literal or host variable char(4)**
 - **Can be used with transitive closure**
 - **Some restrictions still for stage 1, indexable**



The most common mismatches for data types come with languages like Java, C++ and C and decimal data. Often the comparison is from a floating point host variable to a decimal column.

A second type of mismatch that is very common is to have a literal or host variable with a character column length greater than that of the column.

For both of these cases, the result was often poor performance because of the inability to use an index. While there are still some restrictions, performance is expected to improve substantially for many customers.

Stage 1 Indexable Unlike-types



- DB2 enhanced to allow index access when host variable & target column are not the same data type
- Deals with programming languages that don't support the full range of SQL data types
 - f* C/C++ has no DECIMAL data type
 - f* Java has no fixed length CHAR data type
 - f* etc.
- Significant performance improvement for many applications
- Simplifies application programmer & DBA tasks



It is sometimes necessary for customers to join tables on columns with different data types, or provide a search value with a data type that doesn't match the column's definition. For example, allow C double predicate to be indexable with a SQL DECIMAL(p,s). As z/OS data is accessed from PCs, UNIX, Linux and Java, we need to have a C data type of DOUBLE be indexable with a SQL data type of DECIMAL(p,s) or NUMERIC(p,s).

Currently, for many types of predicates, if the data types of the predicate operands do not match, then the predicate is residual. This can have a large negative effect on the performance of the query.

This change will improve the performance of queries that involve predicates with mismatched data types by allowing those predicates to be Stage 1, and possibly Indexable, (subject to certain restrictions).

Rebuild Pending Indexes – V8

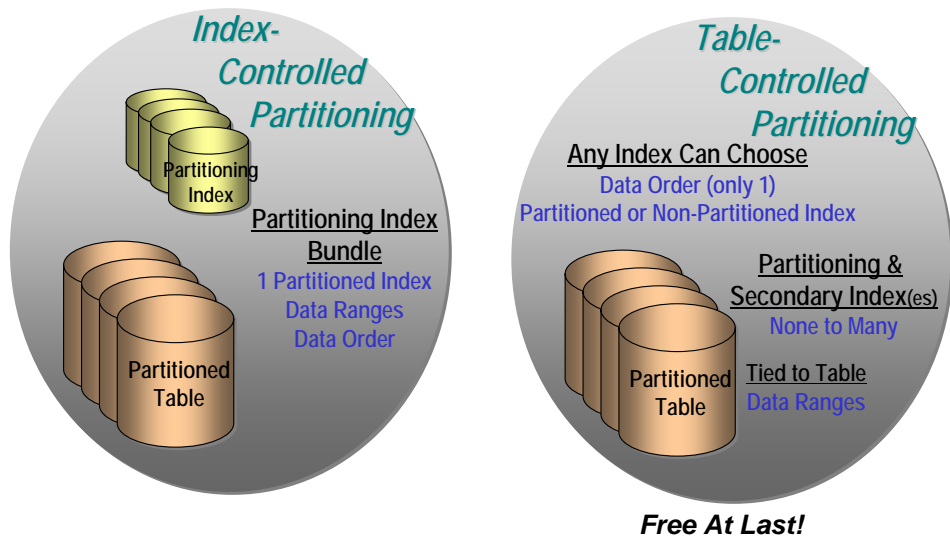


- RBDP Index SQL Allowed
 - SELECTs are allowed
 - DELETEs are allowed
 - Non-unique indexes
 - INSERTs allowed
 - UPDATEs allowed
 - Unique indexes
 - INSERTs not allowed
 - UPDATEs not allowed
- RBDP Index Usage
 - Flushes dynamic cache
 - Dynamic SQL – avoid RBDP index
 - Static SQL – will bind to RBDP index

With DB2 V8, Indexes in RBDP are avoided!



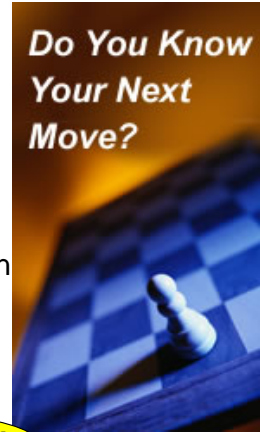
What's New With V8 Partitioned Tables?



With DB2 V8 table-controlled partitioning, the rules have changed. Decisions to partition an index, the partitioned by specs and clustering attributes have all been unbundled. That means more options and index design decisions for the DBA. The concept of the partitioning index has changed and the concept secondary indexes has emerged. There is increased flexibility in assigning, altering, rebalancing and reusing table partitions. There are a bunch of new DDL commands to alter index and partitioned table characteristics. The REORG utility empowers DBAs to better manage partitioned data while minimizing outages. It is a brave new world when it comes to table partitioning.

Converting to Table-Controlled Partitioning

- Methods of Conversion
 - Generally Less Disruptive Ways to Convert
 - Index Changes
 - Altering index clustering characteristics
 - Dropping the partitioning index
 - More Disruptive Ways to Convert
 - Index Changes
 - Creating an index that exploits new function
 - Table Changes
 - Altering table to exploit new Functions



Caution

You Don't Have to DROP/CREATE to Convert!



There is no command to directly convert to table-controlled partitioning. Dropping the partitioning or altering the clustering index will do the trick. But watch out, if you use some of the new table-controlled partitioning features, DB2 will automatically convert to table-controlled partitioning behind the scenes.

You don't have to drop and recreate objects. You don't have the disruptions of a type 2 index conversion. But you do need to reorganize soon after the ALTERs to avoid performance overhead.

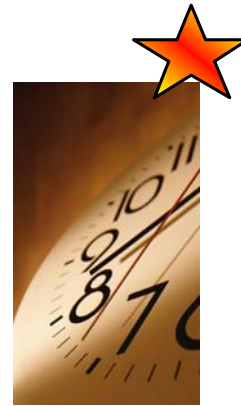
Online Schema Evolution 1

f Changing Partition Definitions

- Add Partition
- Rotate Partition
 - Move first part to last
 - E.g. keep rolling 36 months

f ALTER INDEX : ADD COLUMN, PADDED or NOT, CLUSTER or NOT

f CREATE INDEX while online



Three types of changes are very high on our priority list: changing partitions, changing table attributes and unbundling partitioning and clustering. This is the first category, partition changes.

Adding a new partition to an existing partitioned tables space is very important. Rotating the partitions, such as keeping a rolling 36 months of data is also key. Creating an index has been very disruptive, but V8 provides changes that will allow creation of an index while the work continues.

<ftp://ftp.software.ibm.com/software/db2storedprocedure/db2zos390/techdocs/Z21m.pdf>

<ftp://ftp.software.ibm.com/software/db2storedprocedure/db2zos390/techdocs/Z22m.pdf>

Alter index add column

- Ability to add a column to the end of an index
 - f* Creates a new version
- When column preexists in the table index is placed in RBDP
- If it's a **new column in the table**, add it to the table and index in the **same UOW**

E.g., ALTER TABLE CUST
ADD COLUMN NEW_COL;
ALTER INDEX CUST_IDX
ADD COLUMN NEW_COL ASC;
COMMIT;



A column can now be appended to an existing index.

If the column being added to the index already existed for a populated table, the index is placed in rebuild pending (RBDP), even if there is a default value for the column.

Many times when a new column is added to the table, there may also be a desire to add it to an existing index. Since no data exists, if this is done in the same unit of work, the index is not placed in RBDP.

See the example of two alters in the same

Online Schema Evolution 2

- Unbundling Partitioned Table Attributes
 - f* Partition without an index
 - Table controlled partitioning
 - May be able to have one less index
 - f* Data Partitioned Secondary Index
 - f* Cluster on any index
 - May be able to have more efficient clustering
 - E.g. partition by date, cluster by account
 - f* ALTER CLUSTER attribute



Partitioning and clustering are bundled in current DB2. Some of the time we are required to make a difficult choice. We also want to partition without an index and be able to cluster on any index. These changes will allow us to have one less index and less random IO in some cases.

<ftp://ftp.software.ibm.com/software/db2storedprocedure/db2zos390/techdocs/Z21m.pdf>

<ftp://ftp.software.ibm.com/software/db2storedprocedure/db2zos390/techdocs/Z22m.pdf>

Online Schema Evolution 3

▪ Table or column Changes

f Increase column within numeric data types

- smallint, integer, decimal, float
- No loss of precision allowed

f Change to expand character data type

f Change varchar to / from char



We are able to change the data type for columns. In V5 we could increase the size of varchar columns, but this change allows us to extend numeric and character columns and to change between char and varchar.

4096 Partitions

- Maximum number of parts raised from 254 to 4096
 - f* Table spaces and indexes
 - f* Table space use DSSIZE to go beyond 254 parts
- ALTER TS ADD PART adds partitions to the end
- Maximum table size remains 16 TB for 4 KB pages
- Data set naming convention
 - f* 'Axxx' - partitions 1-999 'Bxxx' - partitions 1000-1999
 - f* 'Cxxx' - partitions 2000-2999 'Dxxx' - partitions 3000-3999
 - f* 'Exxx' - partitions 4000-4096
- Maximum number of parts allowed depends upon page size & DSSIZE
 - 4K page size, DSSIZE=1GB => 4096 parts, 4 TB max table
 - 4K page size, DSSIZE=64GB => 256 parts, 16 TB max table



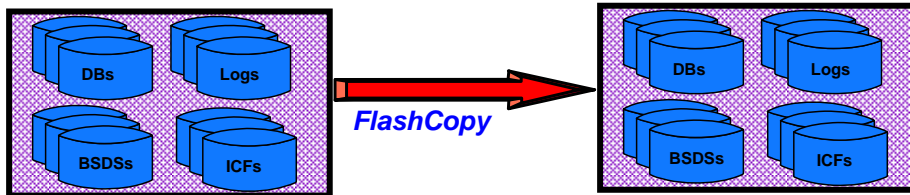
The maximum number of partitions goes from 254 to 4096, so that you can have one partition per day for more than 11 years. Do not define all of the partitions, since you can add new partitions at the end.

While the maximum size remains 16 terabytes for 4K pages, larger page sizes can exceed 16 terabytes for a single table. DSSIZE is recommended, although LARGE also allows more than 254 parts.

This change requires changes in the data set naming convention.

DB2 Managed FlashCopy

- Provide an easier and less disruptive way for fast volume-level backup and recovery
 - f* Use FlashCopy to backup DB2 data and logs
 - f* No longer need to suspend logs
 - f* Handle large number of tables and indexes
 - f* Backups are managed by DB2 and DFSMSHsm to support system level Point-In-Time recovery



DB2 Managed FlashCopy Solution in V8 ...

- *new utilities in DB2 for z/OS V8:*
 - f BACKUP SYSTEM*
 - f RESTORE SYSTEM*

- *Takes system-level copies of data and logs*
 - f Exploits SMS CopyPool in z/OS 1.5*
 - f DB2 data and logs must be SMS-managed*
 - f Write log activity is NOT suspended*
 - f Suspends data set creation, deletion, rename, and extend operations*



Automatic Storage Management

- No need to specify space parameters
- Sliding Scale to avoid reaching maximum extent limit
 - 255 extents
 - 7,257 extents (z/OS 1.7 or above)
- Start with small disk space
 - Start with small secondary extents
 - Larger secondary extents as table grows
- Improves DBA productivity
- Avoids application outage
- Reduces need to reorganize



Currently, database administrators need to specify primary and secondary space quantity. Then they need to monitor the data set sizes and extents to avoid an outage. Since the secondary extents are a fixed size, they must make the tradeoff between wasted space for many small tables and not having adequate space for rapidly growing tables. We can manage the extent size based upon the growth pattern of the table or index.

Larger CI size for larger page sizes

- New CI size equals page size by default
 - e.g. 16K CI for 16K page, DSNZPARM to set
- Enables I/O striping for 8K, 16K, 32K page
- Higher data rate for 8K, 16K, and 32K page
 - 16K page measurement with 16K instead of 4K CI
 - ✓ +36% for non EF (Extended Format) data sets
 - ✓ +70% for EF data sets
 - ✓ VSAM EF getting nearly equivalent to non EF in data rate performance



A new option in DB2 V8 allows use of larger CI sizes with 8K, 16K and 32K pages, rather than using 4 K pages on disk. There are very substantial performance improvements in disk performance for this change. Note that 32K CI size results in 16K block size, so space is not wasted on disk. The improvement in performance is better for Extended Format data sets, so that having a larger DSSIZE is more reasonable for performance.

Data Sharing Enhancements

- **Batching of GBP writes and castouts**
 - f* Write/castout multiple pages in a single CF operation
 - f* Improved data sharing performance, especially for batch updates
 - f* Requires z/OS V1R4, CFLEVEL=12
- **Reduced global contention for table space L-locks**
 - f* Reduced XES-level contention across members
 - f* Improved data sharing performance, especially for OLTP
 - f* RELEASE(DEALLOCATE) may not be needed



Batching of GBP writes and castouts

Write/castout multiple pages in a single CF operation

Improved data sharing performance, especially for batch updates

Requires z/OS R4, CFLEVEL=12

Reduced global contention for table space L-locks

IX/IX and IX/IS TS locks no longer hit XES-level contention across members

Improved data sharing performance, especially for OLTP

Recommendation for RELEASE(DEALLOCATE) can be softened

New locking protocol enacted only with New Function Mode

Data Sharing Enhancements ...

- Changed pages written to GBP at Phase1 instead of Phase2
 - f* Transactions invoking other transactions at syncpoint for same data
 - f* Unusual "record not found" from another member
 - f* Easier to manage
 - f* Equivalent performance
- More efficient index split processing for data sharing



Changed pages written to GBP at Phase1 instead of Phase2

Some Tx Managers spawn other transactions at syncpoint

Spawned tx can encounter "record not found" if it tries to read originating tx's update from another member (rare, but a few customers have reported it)

Moving writes up to Phase1 by default removes need to monitor for this and to set IMMEDIATE PH1 Zparm or Bind option if needed

Equivalent performance for Ph1 vs. Ph2 writes
More efficient index split processing for data sharing

Utility Improvements



- f* **Online CHECK Index**
- f* **On-line REORG Enhancements**
 - **DISCARD**
 - **Avoid BUILD2 with DPSI**
 - **REORG DB2 catalog SHRLEVEL REFERENCE**
 - **REBALANCE partitions**
- f* **LOAD & UNLOAD delimited input & output**
- f* **SCOPE PENDING**
- f* **RUNSTATS non-uniform statistics on non-index columns**
- f* **System-level point-in-time backup and recovery**
- f* **LOAD & UNLOAD LOB**
- f* **Improved defaults for performance**



Many utility enhancements are part of the base changes in this version, supporting long names, Unicode, 64 bit addressing, DPSIs, system point-in-time backup and recovery and schema evolution. These utility enhancements improve our value for the money.

Schema evolution uses utility support to rotate the first partition to the last partition. The new REBALANCE function can balance the sizes of a partition range or of all partitions.

The REORG DISCARD can be performed with SHRLEVEL CHANGE. DPSIs can be reorganized without a BUILD2 phase. The DB2 catalog tables can all be reorganized in SHRLEVEL REFERENCE or read only mode.

Delimited files can be used as input to LOAD or output from UNLOAD.

SCOPE PENDING provides improved usability. SCOPE PENDING indicates that only partitions in a REORP or AREO* state for a specified table space or partition range are to be reorganized.

<ftp://ftp.software.ibm.com/software/db2storedprocedure/db2zos390/techdocs/Z06m.pdf>

Summary

- **Scalability Enhancements including 64 Bit Storage**

- **Index enhancements provide greater efficiencies:**
 - **DPSI index to improve partition independence**
 - **Index only access on varying length data**
 - **Decreased storage requirements (in most cases) for varying-length keys**
 - **Increased key size, now up to 2000 bytes**

- **Ability to ALTER column data types and index columns rather than DROP and CREATE**

- **DB2 and DFSMS managed system level backup and recovery**

