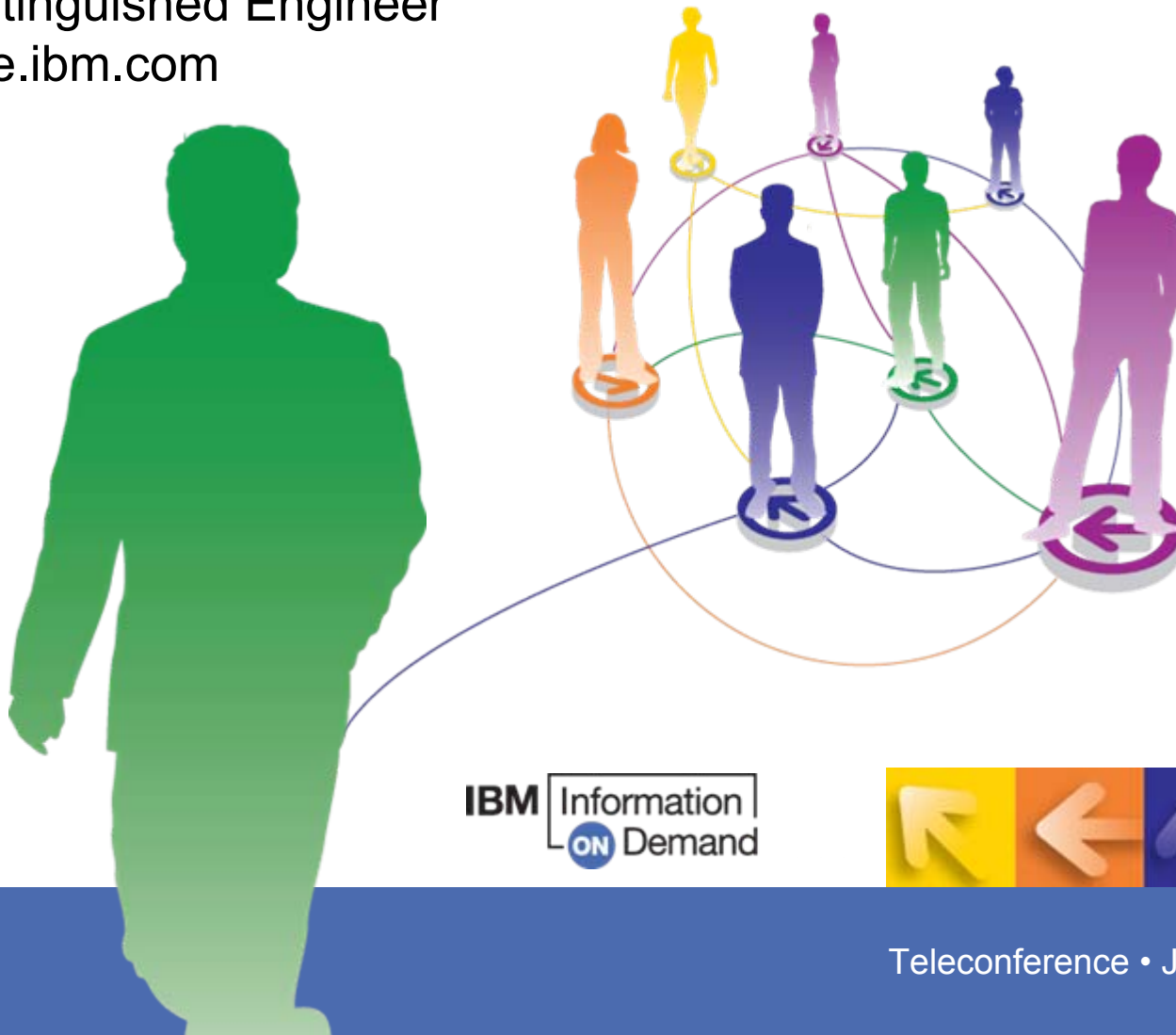


Hints and Tips to Get Most out of DB2 for z/OS

Namik Hrle
IBM Distinguished Engineer
hrle@de.ibm.com



IBM Information
ON Demand



IBM

Teleconference • January 26, 2010

Disclaimer

© Copyright IBM Corporation 2009. All rights reserved.

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

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

IBM, the IBM logo, ibm.com, DB2 and DB2 for z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

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

DB2 9 – A Rich, Features Filled Release

- SHRLEVEL(REFERENCE) for REORG of LOB tablespaces
 - Online RENAME COLUMN
 - Online RENAME INDEX
 - Online CHECK DATA and CHECK LOB
 - Online REBUILD INDEX
 - Online ALTER COLUMN DEFAULT
 - More online REORG by eliminating BUILD2 phase
 - Faster REORG by intra-REORG parallelism
 - Renaming SCHEMA, VCAT, OWNER, CREATOR
 - LOB Locks reduction
 - Skipping locked rows option
 - Tape support for BACKUP and RESTORE SYSTEM utilities
 - Recovery of individual tablespaces and indexes from volume-level backups
 - Enhanced STOGROUP definition
 - Conditional restart enhancements
 - Histogram Statistics collection and exploitation
 - WS II OmniFind based text search
 - DB2 Trace enhancements
 - WLM-assisted Buffer Pools management
 - Plan stability
 - ...
- Global query optimization
 - Generalizing sparse index and in-memory data caching method
 - Optimization Service Center
 - Autonomic reoptimization
 - Logging enhancements
 - LOBs network flow optimization
 - Faster operations for variable-length rows
 - NOT LOGGED tablespaces
 - Index on expressions
 - Universal Tablespaces
 - Partition-by-growth tablespaces
 - APPEND option at insert
 - Autonomic index page split
 - Different index page sizes
 - Support for optimistic locking
 - Faster and more automatic DB2 restart
 - RLF improvements for remote application servers such as SAP
 - Preserving consistency when recovering individual objects to a prior point in time
 - CLONE Table: fast replacement of one table with another
 - Index compression
 - Index key randomization
 - ...
- DECIMAL FLOAT
 - BIGINT
 - VARBINARY, BINARY
 - TRUNCATE TABLE statement
 - MERGE statement
 - FETCH CONTINUE
 - ORDER BY and FETCH FIRST n ROWS in sub-select and full-select
 - ORDER OF extension to ORDER BY
 - INTERSECT and EXCEPT Set Operations
 - Instead of triggers
 - Various scalar and built-in functions
 - Cultural sort
 - LOB File Reference support
 - XML support in DB2 engine
 - Enhancements to SQL Stored Procedures
 - SELECT FROM UPDATE/DELETE/MERGE
 - Enhanced CURRENT SCHEMA
 - IP V6 support
 - Unified Debugger
 - Trusted Context
 - Database ROLES
 - Automatic creation of database objects
 - Temporary space consolidation
 - 'What-If' Indexes
 - ...

TCO Reduction

Continuous Operations

Performance

Scalability

SQL

Portability

SELECT FROM UPDATE/DELETE/MERGE

V8

SELECT FROM INSERT

Retrieves columns values created by INSERT in a single SELECT statement including:

- Identity columns
- Sequence values
- User-defined defaults
- Expressions
- Columns modified by BEFORE INSERT trigger
- ROWIDs

Avoids possible expensive access path that separate SELECT might be using

V9

SELECT FROM INSERT UPDATE DELETE MERGE

One SQL call to DB2 modifies the table contents and returns the resultant changes to the application program.

E.g. we can now code destructive read from a table when a SELECT FROM DELETE statement is included. This feature is particularly useful when a table is used as a data queue.

ORDER BY and FETCH FIRST in Subselect

V8

ORDER BY and FETCH FIRST can be specified only as part of select-statement, i.e. one can write:

```
SELECT * FROM T1  
ORDER BY C1  
FETCH FIRST 1 ROW ONLY
```

but not the following:

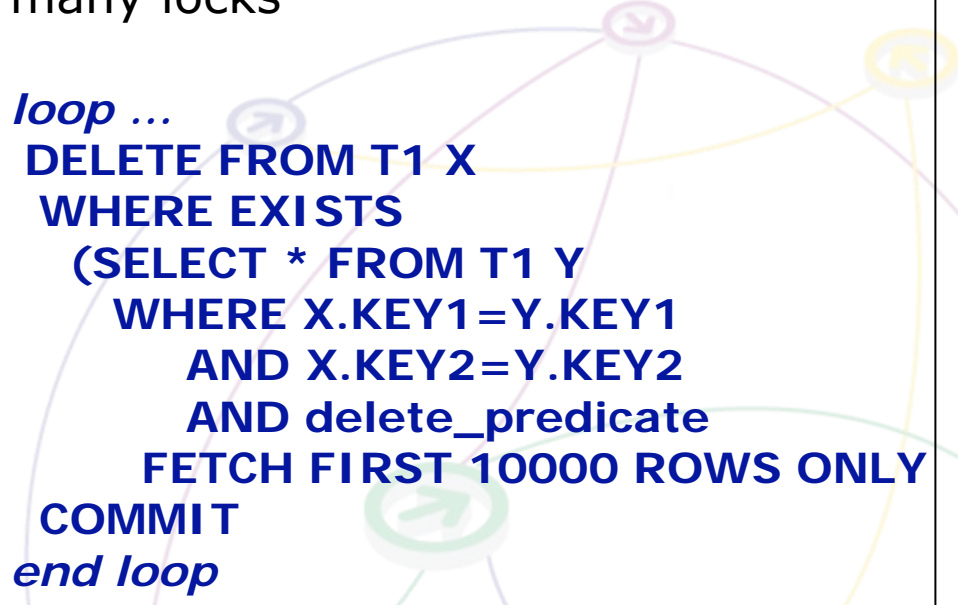
```
INSERT INTO T2  
(SELECT * FROM T1  
ORDER BY C1  
FETCH FIRST 1 ROW ONLY)
```

V9

The restriction has been removed – the clauses can be specified in either a subselect or fullselect.

Interesting example: a loop over the statement followed by a commit deletes rows without acquiring too many locks

```
loop ...  
DELETE FROM T1 X  
WHERE EXISTS  
(SELECT * FROM T1 Y  
WHERE X.KEY1=Y.KEY1  
AND X.KEY2=Y.KEY2  
AND delete_predicate  
FETCH FIRST 10000 ROWS ONLY)  
COMMIT  
end loop
```



Skipping Locked Rows

V8

Application does not scale well due to increased lock contention.

&

Application semantics requires committed and **available** rows only.

An example of such an application is a messaging system:

- only committed and available messages can be processed
- those locked at the time will be processed later.

V9

New SQL clause: **SKIP LOCKED DATA**

Applies to:

- select-statements, SELECT INTO, PREPARE, searched UPDATE, searched DELETE, UNLOAD utility
- Isolation levels CS or RS
- Row or page level locking

It allows a transaction to skip over rows that are incompatibly locked by other transactions, without being blocked.

An interesting usage scenario is serializing access to any kind of object:

- Create a table and insert a row for each object to be controlled
- Code: `SELECT ... FOR UPDATE OF ... SKIP LOCKED DATA`
- The object unavailability is identified by return code +100 (without any wait)

New Techniques to Retrieve LOBs: FETCH CONTINUE

V8

Existing techniques to retrieve entire LOBs with a large maximum length are not optimal

- Fetch into pre-allocated buffer
 - using max size buffer results in best performance, but inefficient storage utilization
 - Using smaller buffer results in truncation which is not accompanied by the actual LOB size
- Using LOB locators
 - storage efficient but requires multiple trips to DB2

V9

FETCH WITH CONTINUE when retrieving base row followed by **FETCH CURRENT CONTINUE** if LOB is truncated

New techniques to retrieve LOBs:

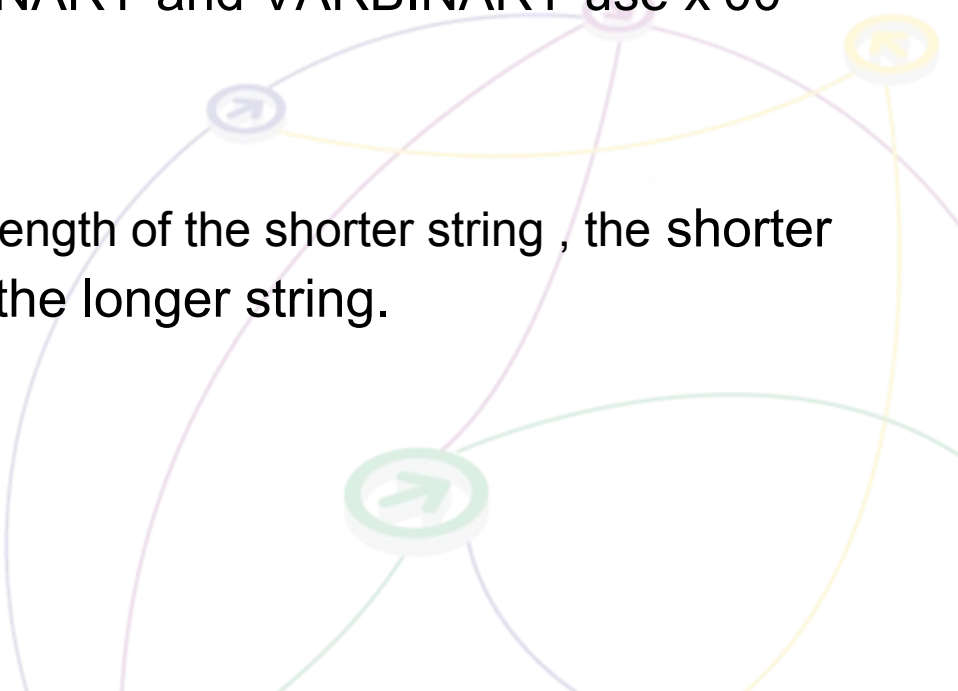
1. Fetch into moderately sized buffer, expected to fit most values. If it does not:
 - Allocate larger buffer using the actual LOB length returned on the FETCH
 - **FETCH CURRENT CONTINUE** to retrieve the rest
2. Fetch through a streaming, relatively small buffer (e.g. 32KB)
 - move and reassemble data pieces into a larger buffer or pipe it to an output file

These techniques apply to XML as well



New Data Types: BINARY and VARBINARY

- BINARY fixed-length binary string
 - 1 to 255 bytes
- VARBINARY variable-length binary string
 - 1 to 32704 bytes; maximum length determined by the maximum record size associated with the table
- Unlike FOR BIT DATA the new BINARY and VARBINARY use x'00' as a padding character
- Comparison rule:
 - If two strings are equal up to the length of the shorter string , the shorter string is considered less than the longer string.



APPEND

V8

All of the following applies:

- Critical, high insert rate workload needs better performance and all the conventional tuning steps have already been applied.
- Clustering is either not beneficial or more frequent reorganizations are acceptable
- *MC00* insert algorithm is still not fast enough or the prerequisites cannot be satisfied:
 - MEMBER CLUSTER
 - FREEPAGE=PCTFREE=0

V9

CREATE TABLE ... APPEND YES | NO

ALTER TABLE ... APPEND YES | NO

- The APPEND YES results in a fast insert at the end of the table or appropriate partition at the expense of data organization and rapid table space growth.
- After populating with the APPEND option in effect, clustering can be achieved by running the REORG utility providing a clustering index has been explicitly defined.
- Make sure PK81471 is applied
- Note that MC00 is still valid, but make sure that PK81470 is applied

Reordered Row Format

- Remember tuning recommendations for rows with variable-length columns?
 - Define fixed-length columns preceding variable-length columns
- DB2 9 implements this recommendation internally
 - New **Reordered Row Format** provides for direct access to variable length columns
 - Potential for significant CPU reduction
 - For all newly created tables, at REORG and LOAD REPLACE
 - With some exceptions

Prefix	Fixed Length Cols	Varchar Pointers	Varying Length Cols
--------	-------------------	------------------	---------------------

- Rarely leading to increased logging volume
- LOAD REPLACE and REORG convert rows into RRF
 - KEEPDICTIONARY ignored when the utilities run for the first time in DB2 9
- Related service
 - PK87348
 - REORG option (ROWFORMAT BRF | RRF) to convert a tablespace from RRF to BRF and vice versa
 - Support for BRF by UTS
 - Introducing public zparm RRF instead of hidden SPRMRRF
 - PK78958 – conversion to RRF suppressed for compressed pagesets
 - Namely, RRF can lead to lower compression ratio

Virtual Indexes a.k.a. 'What If' Indexes

V8

How to determine that a new index would benefit a given **dynamic SQL** query?

How to determine that dropping an index will not negatively affect a given query?

- In many cases predicting based on modeling is not reliable due to query complexity
- Indiscriminate adding of indexes creates permanent overhead for most operations (SQL and utilities)
- Creating a new index is obtrusive for concurrent operations
- Using a test system for experimenting lacks potentially crucial environmental factors that affect access path selection

V9

- Virtual i.e. hypothetical indexes can be specified and made visible to statement EXPLAIN STATEMENT FOR
- Table DSN_VIRTUAL_INDEXES is used to specify virtual indexes
 - Table columns include selected columns from SYSINDEXES and SYSKEYS
 - Users need to create the table manually, unless tooling such as Index Advisor does it automatically. Appropriate script is provided.
 - To create/drop an index, the table needs to be populated with a row that provide an appropriate description of index
- At EXPLAIN time, during query optimization, the virtual indexes compete with regular indexes on the tables in a cost-based fashion and the dropped indexes are not considered

TBCREATOR	Authorization ID of owner (or schema in V9) of table on which the index is being created/dropped
TBNAME	Name of the table on which the index is being created or dropped
IXCREATOR	Authorization ID (or schema in V9) of the owner of the index
IXNAME	Name of the index
ENABLE	Whether this index specification will be processed ('Y') or not ('N').
MODE	Whether the index is being created ('C') or dropped ('D')
UNIQUERULE	Whether the index is unique: D for No (duplicates are allowed); U for Yes
COLCOUNT	The number of columns in the key
CLUSTERING	Whether the index is clustered ('Y' or 'N')
NLEAF	Number of active leaf pages in the index. If unknown, the value must be -1.
NLEVELS	Number of levels in the index tree. If unknown, the value must be -1.
INDEXTYPE	The index type: '2' - NPSI; 'D' - DPSI
PGSIZE	Size, in bytes, of the leaf pages in the index: 4K, 8K, 16K, 32K
FIRSTKEYCARDF	Number of distinct values of the first key column. If unknown, the value must be -1.
FULLKEYCARDF	Number of distinct values of the key. If unknown, the value must be -1.
CLUSTERRATIOF	Clustering ratio. . If unknown, the value must be -1.
PADDED	Indicates whether keys within the index are padded for varying-length column data ('Y' or 'N')
COLNO1	Column # of the first column in the index key
ORDERING1	Ordering ('A' or 'D') of the first column in the index key
...	...
COLNO64	Column # of the last column in the index key. Needs to be populated only when # index keys = 64
ORDERING64	Ordering ('A' or 'D') of the last column in the index key.



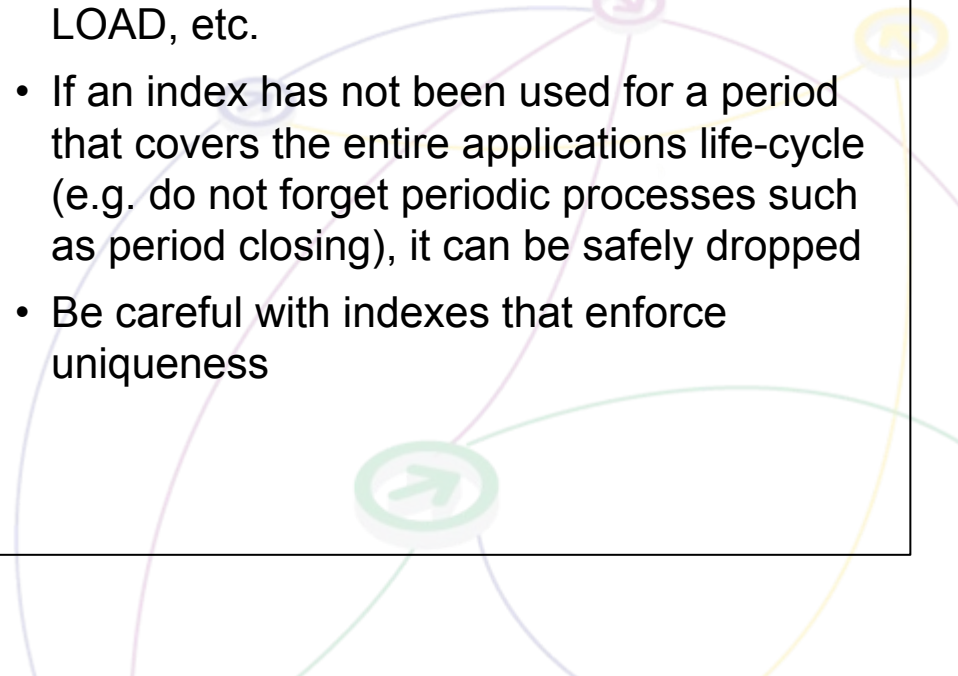
Identifying Unused Indexes

V8

- In order to improve the performance of different types of queries, some customers create many indexes just in case one of them can be useful.
- However, there is a significant cost in maintaining such indexes during Insert, Delete, Load, Reorg, ...
- Some of the indexes are no longer needed and they could be dropped, but how can we be sure that they have not been used for a very long time?

V9

- For each index, a new real-time statistics column (LASTUSED) is maintained in table SYSIBM.SYSINDEXSPACESTATS
- It is updated whenever the index is used in
 - SELECT/FETCH
 - searched UPDATE/DELETE
 - Referential Integrity checking
- The column is not updated for INSERT, LOAD, etc.
- If an index has not been used for a period that covers the entire applications life-cycle (e.g. do not forget periodic processes such as period closing), it can be safely dropped
- Be careful with indexes that enforce uniqueness



TRUNCATE TABLE

V8

Alternative way of deleting the entire table is needed for any of these reasons:

- DELETE without WHERE clause is not fast enough as the table includes delete triggers
- Using LOAD REPLACE with empty input data set (even when called via DSNUTILS) is not DBMS agnostic
- Storage occupied by deleted rows should be released faster

V9

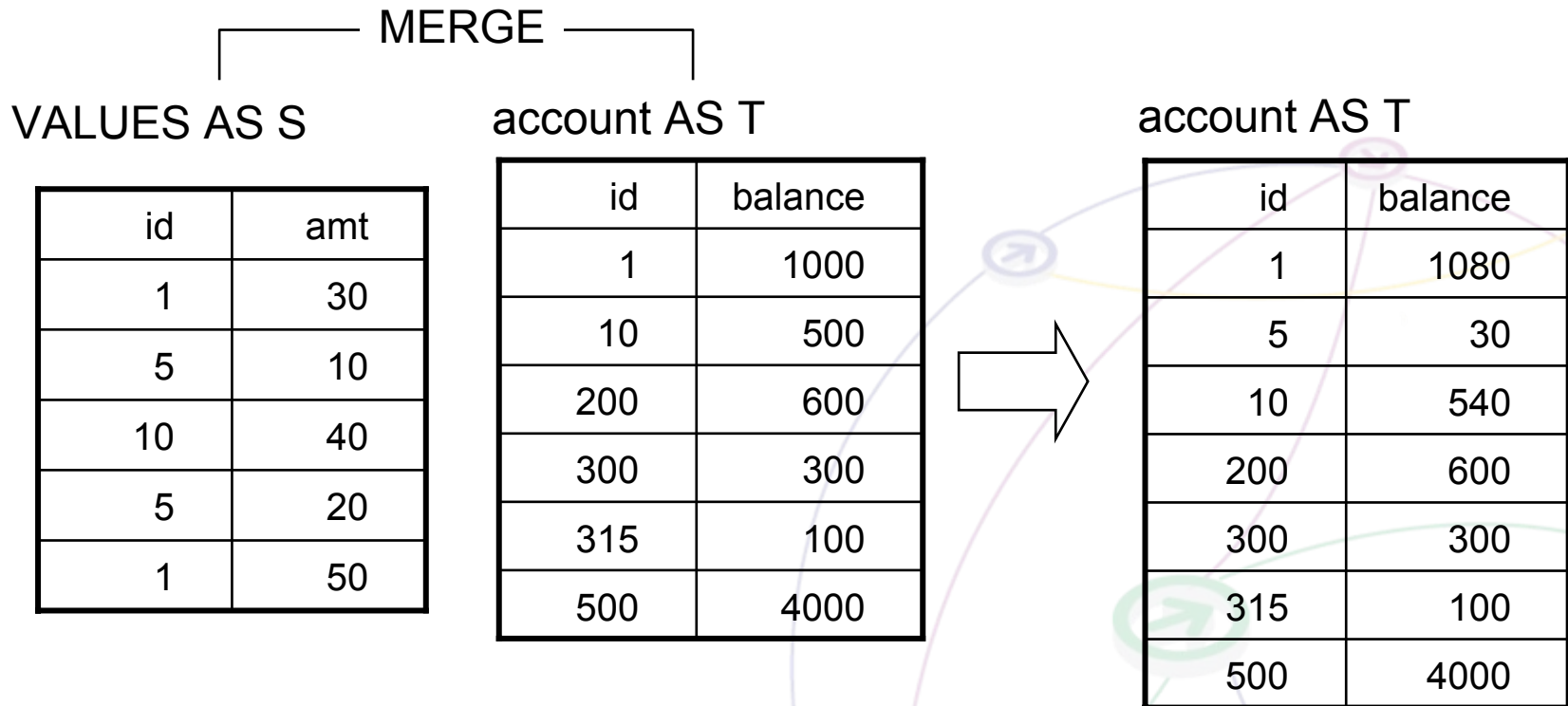
New DML statement:

TRUNCATE table
DROP | REUSE STORAGE
IGNORE | RESTRICT DELETE TRIGGERS
IMMEDIATE

Under the cover it's DELETE without WHERE clause, but without delete triggers processing overhead.
Therefore it is fast for tables in segmented and universal tablespaces for which there are no CDC, MLS and VALIDPROC enabled attributes.

MERGE Example

```
MERGE INTO account AS T
USING VALUES (:hv_id, :hv_amt) FOR 5 ROWS AS S (id, amt)
ON T.id = S.id
WHEN MATCHED      THEN UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```



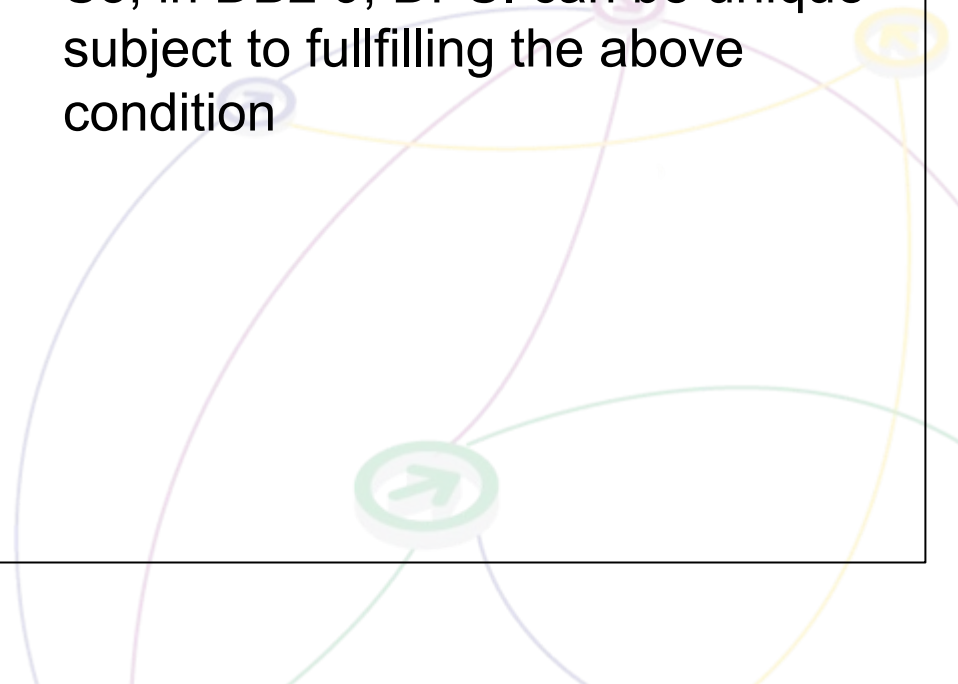
Unique DPSI

V8

- Data-Partitioned Secondary Index cannot be created as UNIQUE
- The reason:
Avoiding scanning all the partitions of DPSIs in case when uniqueness need to be enforced.
- An obstacle for wider usage of DPSIs

V9

- If the columns by which the table is partitioned are contained in the DPSI, only the corresponding DPSI partition need to be checked for uniqueness
- So, in DB2 9, DPSI can be unique subject to fullfilling the above condition



Universal Tablespaces

V8

A tablespace needs both partitioned and segmented organization:

- it's larger than 64GB
- inter-partition parallelism or independent processing is needed
- partition scope operations (ADD,ROTATE) apply
- rows are variable in length and optimal space utilization is preferred
- mass delete operations should be fast

V9

- A hybrid between partitioned and segmented organization

```
CREATE TABLESPACE ...  
  SEGSIZE integer  
  NUMPARTS integer
```

- One table per UTS only
- Two types:
 1. Partitioned-by-growth
 - always UTS
 - described on the next page
 2. Partitioned-by-range
 - traditional partitioned tablespaces
 - optionally UTS
- Incompatible with MEMBER CLUSTER

Partition By Growth

V8

A table's growth is unpredictable (it could exceed 64GB) and there is no convenient key for range partitioning.

Partitioning by a ROWID column introduces additional tablespace administration overhead:

- estimating optimal number of partitions
- ADDing partitions if necessary
- less than optimal space utilization)

V9

CREATE TABLESPACE ... *explicit specification*
MAXPARTITIONS integer

CREATE TABLE ... *implicit specification*
PARTITIONED BY SIZE EVERY integer G

Associated SYSTABLESPACES columns

MAXPARTITIONS = max number of partitions
PARTITIONS = actual number of partitions
TYPE = G

- Only single-table tablespace
- Universal Tablespace organization: although the table space is partitioned, the data within each partition is organized according to segmented architecture
- Incompatible with MEMBER CLUSTER, ADD PARTITION, ROTATE PARTITION
- REORG considerations

Larger Index Page Sizes

V8

- In heavy insert applications frequent index pages splitting creates scaling bottleneck
- In a data sharing environment, the indicators are:
 - ✓ Large Accounting Class 3 synchronous log write suspension
 - Each GBP-dependant index page split requires 2 forced log writes
 - Note that these suspensions include also waits for phase 1 of 2-phase commit and P-lock negotiation
 - ✓ High number of latch class 6 contentions
- For non-data sharing, the indicator is large number of latch class 254 contentions (reported in IFCID 57)

V9

- Additional 8K, 16K and 32K index page sizes
- Advantages
 - + It can reduce index page splits by up to 2x, 4x or 8x respectively
 - + Traversing shallow index needs less getpages
 - + Faster index scans
- Disadvantages
 - Larger pages require larger buffer pools
 - If access predominantly random, buffer pool might need to be 2x, 4x or 8x larger
 - For data sharing it can result in more P-lock contentions
- Rule of thumb:
 - ✓ 4K index page and larger PCTFREE for indexes with random insert patterns
 - ✓ larger size index page for indexes with sequential insert patterns.

Comparing Tablespace vs. Index Compression

	Tablespace Compression	Index Compression	Remarks
Unit of compression	Row	Page	Index non-leaf pages are not compressed, but that's typically less than 5% of the entire index
Where is data compressed	on DASD, in buffer pools, in logs	on DASD only	
When does compression happen	at insert and fetch	at I/O	Hence, CPU overhead affected by BP hit ratio. Larger buffer pools strongly recommended for compressed indexes. Do not use page-fix for these buffer pools
Hardware assist	Yes	No	Index compression uses sophisticated software compression algorithms
Page size restrictions	No	Only for 8K, 16K and 32K-page indexes	Page size on DASD: → for tablespaces, equivalent to page size in the associated buffer pool → for indexes, always 4K
Compression dictionary used	Yes	No	
When does compression start	After the first Reorg or Load	At the first insert or update	
Compression CPU cost reported in	Accounting	Accounting and/or DBM1 SRB Statistics	No changes in accounting CPU time if index pages brought in by prefetch
Typical Compression Ratio	10 – 90%	25 – 75%	Maximum CR limited by index page size: 50% for 8K, 75% for 16K and 87.5% for 32K Use DSN1COMP to predict compression ratio

More Efficient Workfiles Usage

V8

- 4K-page workfiles are used whenever row is smaller than 4KB
- Since work file access is often sequential, using larger page size can be more efficient
- E.g. for 2050-byte records:
15 records on one 32K page
vs.
8 records on eight 4K pages

V9

- 32K-page workfiles are used much more aggressively
- 4K-page workfiles are now used only for small records
 - where the limit of 255 rows per page results in waste of space
 - e.g. over 90% wasted space on 32K page for 10-byte records
- Recommendations
 - ✓ Assign a larger 32K workfile buffer pool
 - ✓ Allocate more 32K workfile data sets
 - ✓ If 4K workfile buffer pool activity is significantly lower, then the corresponding buffer pool size and work file datasets can be reduced.
 - ✓ Monitor new statistics on how often more optimal 32K workfiles ran out and 4K workfiles had to be used instead, or vice versa

BACKUP and RESTORE SYSTEM Enhancements

V8

- Off-loading system-level backups to tape and restoring it from tape is a manual process that is not driven by the BACKUP and RESTORE SYSTEM utilities
- Despite availability of system-level backup many sites produce tablespace and index image copies for simpler recoveries of the individual objects
- Physical background copy phase of FlashCopy can take long and impact overall I/O performance
- RESTORE SYSTEM can be used for prior point in time recoveries only

V9

- Full integration of tape into
 - BACKUP SYSTEM as the target for saving system-level backups
 - RESTORE SYSTEM as the source for restoring previously saved system-level backups
- Automatic using of system-level backups as alternative sources for recovering individual tablespaces and indexes
- BACKUP SYSTEM supports Incremental FlashCopy functionality
- DB2 can be restarted in System Point-in-Time Recovery mode without truncating the logs to a prior log point. RESTORE SYSTEM can then be used to recover DB2 to the current end of log.
- MODIFY RECOVERY enhanced to support system-level backups as the primary means of backing up DB2 data

Online REBUILD INDEX

V8

- During rebuilding of an index the tablespace is unavailable for all data modifying operations
- This is especially restrictive for very large tables where the rebuild process typically takes considerable amount of time

V9

```
REBUILD INDEX ...  
  SHRLEVEL REFERENCE  
  DRAIN_WAIT integer  
  RETRY integer  
  RETRY_DELAY integer
```

```
REBUILD INDEX ...  
  SHRLEVEL CHANGE  
  DRAIN_WAIT integer  
  RETRY integer  
  RETRY_DELAY integer  
  MAXRO integer | DEFER  
  LONGLOG CONTINUE | TERM | DRAIN  
  DELAY 1200 | integer
```

Renaming SCHEMA, VCAT, OWNER and CREATOR

V8

- Some administrative operations (such as cloning a DB2 system) often includes the need to change any or all of the following:
VCAT, SCHEMA, OWNER, CREATOR
- These are error prone, time consuming and risky operations:
 - The changes are spread throughout DB2
 - Preserving coherency between the catalog, the dictionary and actual physical objects is absolutely crucial

V9

CATMAINT UPDATE ...

SCHEMA SWITCH (schema_name) TO (new_schema_name)

VCAT SWITCH (vcat_name) TO (new_vcat_name)

OWNER FROM (owner_name) TO ROLE

- Performs authorization/semantics checking and serialization
- Updates catalog and directory to reflect the new names
- Invalidates plan, packages and statement cache
- Names to be changed must not have view, function, MQT and trigger dependencies, cannot be 'SYSIBM'
- Use SCHEMA option to change owner, creator and schema names

Resource Limit Facility Improvements

V8

The qualifiers used by RLF to identify processes for which CPU time is governed are not specific enough for most 'middleware' servers such as SAP:

- Plan name is fixed to DISTSERV
- CLI and JDBC drivers use a common set of packages
- Authorization ID is typically the same for the entire workload

V9

- RLF governed processes can be now qualified by typical 'client-side' identifiers:
 - End-User ID
 - Transaction name
 - Workstation name
- Additionally, the process can be qualified by the IP address of the server that initiated request
- This applies to both the predictive and reactive governor
- The existing CLI and JDBC APIs are used to set the client-side identifiers.

Not Logged Tablespaces – Use It for Extreme Cases Only

V8

Performance degradation for workloads involving very large number of parallel inserts, updates and deletes due to:

- Log write latching
- Log data sets write bottlenecks

Very large volume of log data generated for these workloads

At the same time **recoverability of data is not required**, e.g. the data can be recreated from its original source rather than from backups and logs

V9

New tablespace attribute that controls whether Undo and Redo information for that tablespace and associated indexes are logged or not.

**CREATE or ALTER TABLESPACE ...
LOGGED|NOT LOGGED**

Typical process sequence:

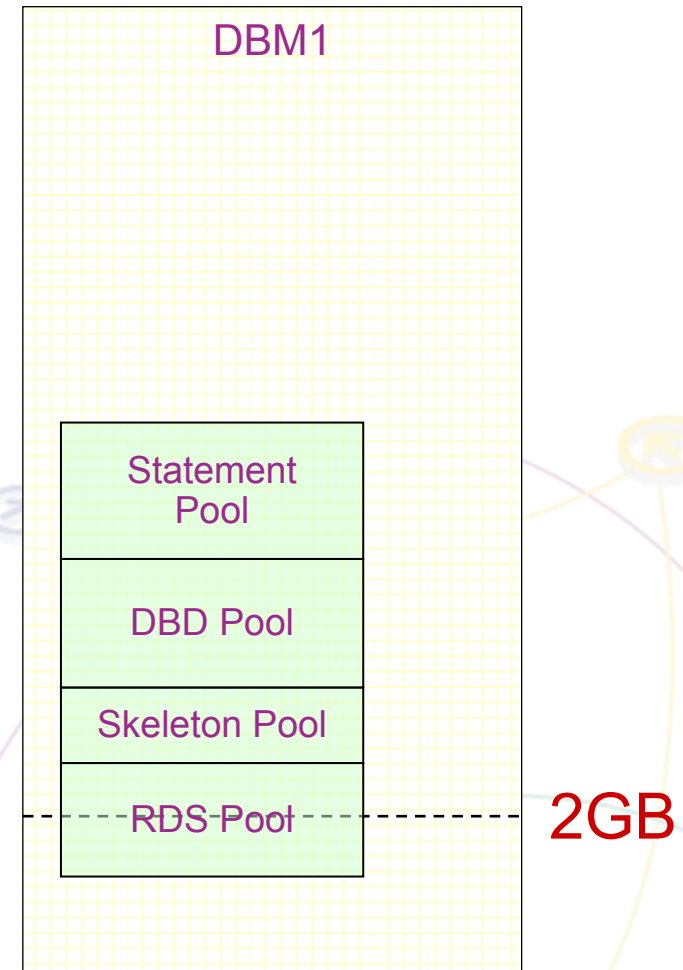
1. Take full image copy
2. Turn off logging
3. Make massive changes
4. Ensure that no other concurrent, non-repeatable changes happen
5. Turn on logging
6. Take full image copy

Get familiar with ramifications to rollbacks, restarts, lock contentions, data sharing, long running transactions

EDM Statement Pool

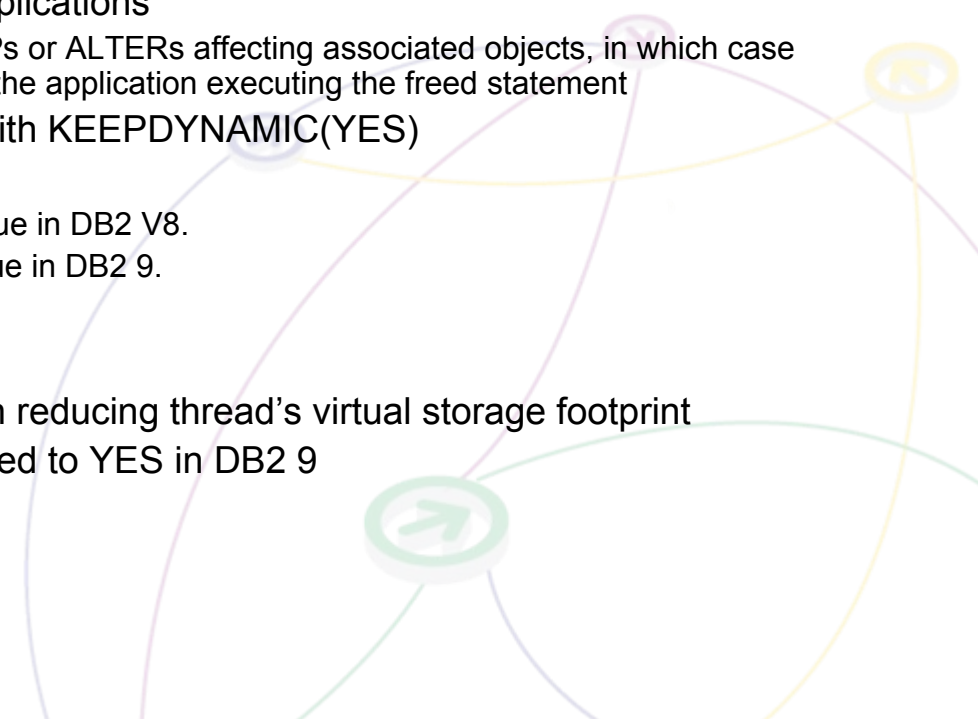
The dynamic statements are cached in the EDM Statement Pool which belongs to EDM storage.

- EDM Statement Pool
 - Skeleton dynamic statements (SKDS)
 - zparm EDMSTMTC
 - As of DB2 V8 above-the-bar
- EDM DBD Pool
 - Database Descriptors (DBDs)
 - zparm EDMDBDC
 - As of DB2 V8 above-the-bar
- EDM Skeleton Pool
 - Skeleton Cursor (SKCT) and Package Tables (SKPT)
 - zparm EDM_SKELETON_POOL
 - As of DB2 V9 above-the-bar
- EDM RDS Pool
 - Cursor (CT) and Package Tables (PT)
 - zparm EDMPOOL
 - As of DB2 V9 partially above-the-bar



Thread Virtual Storage

- A large portion (up to 50%) of virtual storage that holds locally cached statements (particularly INSERTs) moved above-the-bar in V9
- Changed defaults for two zparms that control thread virtual storage utilization
 - CACHEDYN_FREELOCAL
 - Introduced in DB2 V8, APAR PK21861
 - Enables freeing cached statement and releasing associated virtual storage upon statement's completion, i.e. does not wait for a commit to do that.
 - If the statement is re-referenced after being freed, DB2 performs implicit prepare, so the process is transparent to the applications
 - Unless there are concurrent DROPs or ALTERs affecting associated objects, in which case sqlcode -204 could be returned to the application executing the freed statement
 - Applies only to statements bound with KEEP_DYNAMIC(YES)
 - Possible values
 - 0 - Feature is disabled. Default value in DB2 V8.
 - 1 - Feature is enabled. Default value in DB2 9.
 - 2, 3, 4 are serviceability values
 - MINSTOR
 - If set to YES, DB2 actively works on reducing thread's virtual storage footprint
 - Default is NO in DB2 V8, but changed to YES in DB2 9



DB2 9 for z/OS RedBooks & RedPapers

- Powering SOA with IBM Data Servers SG24-7259
- LOBs with DB2 for z/OS: SG24-7270
- Securing DB2 & MLS z/OS SG24-6480-01
- DB2 9 Technical Overview SG24-7330
- Enhancing SAP - DB2 9 SG24-7239
- Best practices SAP BI - DB2 9 SG24-6489-01
- DB2 9 Performance Topics SG24-7473
- DB2 9 Optimization Service Center SG24-7421
- Index Compression with DB2 9 for z/OS paper
- DB2 9 Stored Procedures SG24-7604

IBM
**LOBs with DB2 for z/OS:
Stronger and Faster**

Define LOBs, see how they work, and
see how to store them
Manage LOBs in operational
environments



IBM
**Securing DB2 and
Implementing MLS on
z/OS**

Paula Brad
Mike Buckner
J. Dukeman
Steve Ober
by Garbutt

ks

IBM
**DB2 9 for z/OS
Technical Overview**

Information Management software
Read about the 1
functions of DB2
Understand your
migration process
Evaluate applica
major returns

IBM
**Enhancing SAP by
using
DB2 9 for z/OS**

IBM
**DB2 9 for z/OS
Performance Topics**

IBM.com
Reduced CPU time
Improved scalability and
availability
More z/FP eligibility



IBM
Steve Perle
Steve Beck
Mike Thayer
Mark Green
Michael Sills
John Ruppel
Lorenz Mayer
Neil Ruffalo
Willy Satt
Chickerman

ks

Paula Brad
Kevin Matthews
Gary D. Moore
Laila F. Williams
Giovanna Tosi

IBM.com/redbooks
Redbooks