

# Cool DB2 10 for z/OS Features for Enterprise Applications

par Namik Hrle IBM



**GUIDE Share France**  
Une Association Indépendante d'Utilisateurs IBM

Réunion du Guide DB2 pour z/OS France  
Lundi 19 mars 2012  
Tour Euro Plaza, Paris-La Défense

# Legal Information

© Copyright IBM Corporation 2012. 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

The information on the new product is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information on the new product is for informational purposes only and may not be incorporated into any contract. The information on the new product is not a commitment, promise, or legal obligation to deliver any material, code or functionality. The development, release, and timing of any features or functionality described for our products remains at our sole discretion.

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](http://www.ibm.com/legal/copytrade.shtml)

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

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.

# DB2 10 - A Gold Mine for Enterprise Applications

- Full 64-bit support
- Reducing various latch contentions
- Internal performance enhancements
- Large buffer pools optimization
- Improving query parallelism
- **I/O parallelism for index updates**
- **Inline LOBs**
- **Non-key columns in index**
- Workfiles enhancements
- **UTS support for MEMBER CLUSTER**
- PBG tablespace enhancements
- No LOB/XML materialization within DB2
- **Hash access to data**
- Fine granularity DBA privileges
- Row and Column Access Control
- Bi-temporal support
- SQL PL in the engine
- Moving aggregate functions for OLAP
- Timestamp with Timezone
- Greater timestamp precision
- **Special 'null' indicator**
- Automatic SPs management
- **Support for IBM Smart Analytics Optimizer**
- **Enhanced monitoring support**
- DB2 catalog enhancements
- Automatic checkpoint, Pre-emptable backout
- **Rotating 'n to last' partitions**
- **Instance-based hints**
- Plan stability
- Safe query optimization
- **Dynamic Statements Cache enhancements**
- SQL pagination
- IN-list predicate performance
- UTSERIAL elimination
- Automatic Statistics
- Online schema evolution
- Currently committed data access
- Adding active log
- XML enhancements
- DEFINE NO for LOBs and XML
- **Compressing at insert**
- Reducing need for reorganization
- REORG enhancements
- Support for EAV
- FlashCopy enhancements
- ...

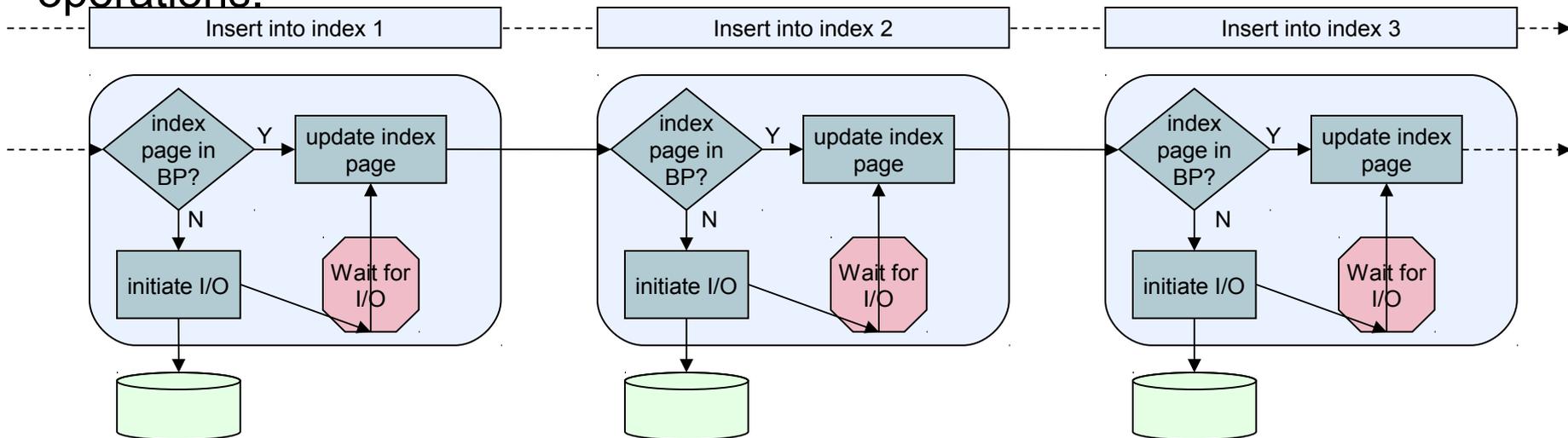
# Insert Performance Bottlenecks - Part 1

What is the largest, often unavoidable, contributor to insert elapsed time?

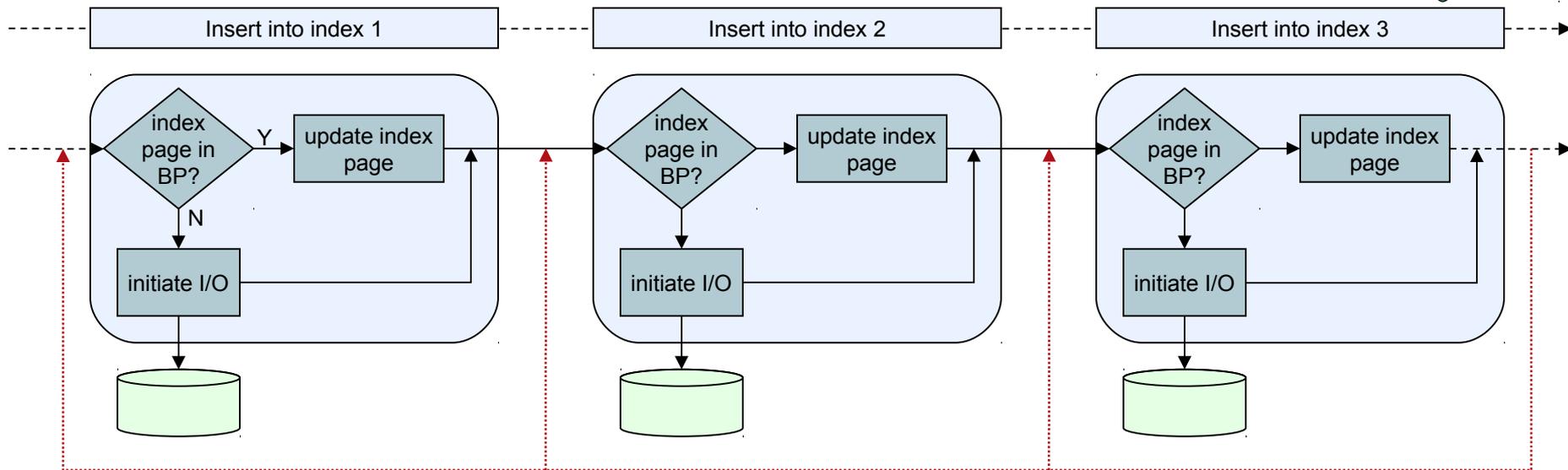
- Locating page to insert?
- Contention, e.g. on space map page, particularly in data sharing?
- Logging?
- Writing inserted pages?
- Read I/O?



Each index is inserted into consecutively, without any overlap of operations.



# Index Read I/O Parallelism at Insert



- There is still one processing task, but the index read I/Os are overlapped.
- It applies to LOAD SHRLEVEL CHANGE as well
- Conditions under which index read I/O parallelism is used:
  - DB2 10 compatibility mode or higher
  - Typically, three or more indexes defined on the table
  - Partitioned or Universal table space
  - zparm INDEX\_IO\_PARALLELISM set to its default value (YES)

# Insert Performance Bottlenecks - Part 2

DB2 9

By default, which of the following characteristics DB2 prefers when inserting rows?

- a) Speed of insert 
- b) Space usage efficiency 
- c) Speed of subsequent queries with range predicates 

The default preference can be changed by specifying:

- APPEND tables
  - ... which entirely ignores clustering and space reuse
- MEMBER CLUSTER tablespaces
  - ... which ignores clustering only,
  - ... but also provides the lowest space map page contention in data sharing,
  - ... and enables a kind of 'space efficient APPEND' behavior, also known as the *MC00 algorithm* triggered by table space settings:
    - MEMBER CLUSTER
    - FREEPAGE = 0
    - PCTFREE = 0

So ... what's the problem?

In DB2 9, MEMBER CLUSTER cannot be defined for UTS.

# MEMBER CLUSTER Enhancements

DB2 10

In DB2 10 MEMBER CLUSTER can be defined for UTS as well.

- For both, Partitioned by Growth and Partitioned by Range UTS
- Each space map covers 10 segments
- A new column MEMBER\_CLUSTER is added to the SYSTABLESPACE catalog table.
  - The values 'I' and 'K' in the TYPE column of SYSTABLESPACE are no longer used.

And the added bonus:

- MEMBER CLUSTER attribute can be ALTERed
- Pending ALTER
  - Tablespace placed in Advisory Reorg Pending status
  - REORG materializes the change

# Single Row Retrieval

DB2 9

What is the fastest way to retrieve a single row in DB2?

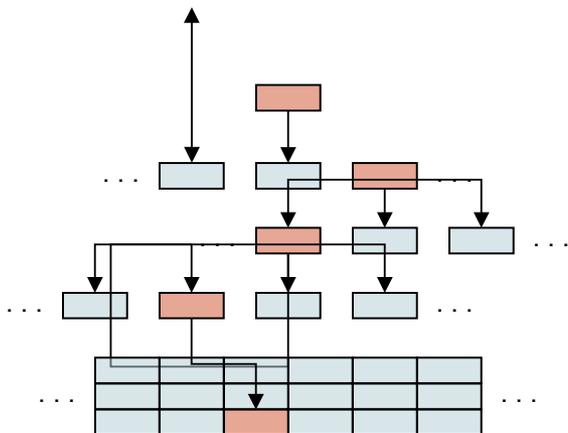
## Equal-unique index access path!

Selected by DB2 when predicate consists of equality conditions connected by AND operator, e.g.

SELECT \* FROM ... WHERE COL1=? AND COL2=? AND COL3=?  
and there is a unique index on (COL1, COL2, COL3).

For fastest performance of dynamic SELECT, additionally use FETCH FIRST 1 ROW ONLY

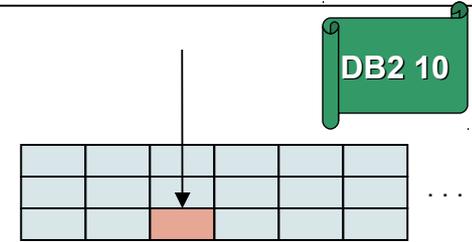
But, even the equal-unique access path might not be good enough:



- Large indexes result in increased number of getpages:
  - number of getpages =  $n + 1$ , where  $n$  is the index depth
- Likelihood of read I/Os increases
- Disorganized indexes exacerbate the problem

# Hash Access to Data

DB2 10 introduces a new, specialized, access path that results in a single getpage (most of the time)



- Applicable to a subset of cases where a unique index could be used
  - Single, unique row retrievals with equality or IN predicates
- Results in less getpages, lower CPU, less I/O
- Rows in a hash-organized table reside in fixed hash space and, optionally, in an overflow space
  - More than one getpage per retrieved row can happen if the row is relocated to the overflow space due to shortage of the fixed hash space
  - To minimize getpages the fixed hash space should be typically 1.2 to 2 times larger than a tablespace without hash organization
    - RTS is enhanced to include indicators assisting in detecting over-allocation or under-allocation of space (too many collisions)
  - Overall space usage might not increase as much if the corresponding index can be dropped (use RTS to check)

```
CREATE TABLE ... HASH KEY (key columns) HASH SPACE(number of bytes)
ALTER TABLE ... ADD HASH KEY (key columns) HASH SPACE(number of bytes)
ALTER TABLE ... DROP HASH ORGANIZATION
```

# Index-only Access Path

DB2 9

‘Overloading’ index with non-key columns, i.e. columns that are not necessarily used for locating data pages, is a common tuning technique

- Resulting index-only access path is very often a great trade-off to negative ramifications of enlarged redundancy
- However, in one specific case, the negative effects are particularly large

Unique index!

Unique indexes do not allow ‘overloading’ with non-key columns:

- It compromises the unique constraint they are enforcing
- Creating another index that includes all the key and non-key columns, but without the UNIQUE constraint comes with well-known drawbacks

# Non-key Index Columns

DB2 10

DB2 10 supports adding non-key columns, also known as 'include columns', to unique index without affecting the unique constraint.

- INCLUDE (column name, ...) clause added to CREATE/ALTER INDEX
- The include columns have different characteristics than the key columns
  - Can be added only to unique indexes
  - Do not participate in ordering of the key (they are just appended to the key)
  - Cannot be used to enforce referential integrity constraints
  - Cannot be converted to key columns (nor vice versa) without recreating the index
  - Cannot be used in:
    - Indexes on expression
    - System-defined catalog indexes
    - Auxiliary indexes
    - XML indexes
    - Partitioning indexes with explicitly specified limit key values

## Benefits:

- Improved performance of DB2 statements and utilities that result in index maintenance
- Disk space savings - by dropping otherwise redundant index

# Delayed Compression

DB2 9

Unlike index compression, data compression is dictionary based.

- Data cannot be compressed before the compression dictionary has been built
- The compression dictionary is built only by:
  - LOAD
  - REORG

What is the main deficiency of this restriction?

Excessive space usage if tables are initially populated by INSERTs!

What is the remedy for this challenge?

1. Stop inserting after 1000 or so inserted rows
2. Reorganize tablespace
3. Resume inserting

Remaining problem?

Operational complexity!

# Early Compression

DB2 10

DB2 10 enables early compression of inserted rows by 'just in time' building compression dictionary during:

- INSERT
- MERGE
- LOAD SHRLEVEL CHANGE RESUME YES

No changes to the applications are needed.

- Applies to all COMPRESS YES tablespaces and partitions
- DB2 transparently builds compression dictionary
  - The triggering operation and following operations do not wait
  - After the dictionary is built, the subsequent inserted rows are compressed
  - Compression dictionary built this way is spread over the whole tablespace

What to do, if for some reason, the old behavior is needed:

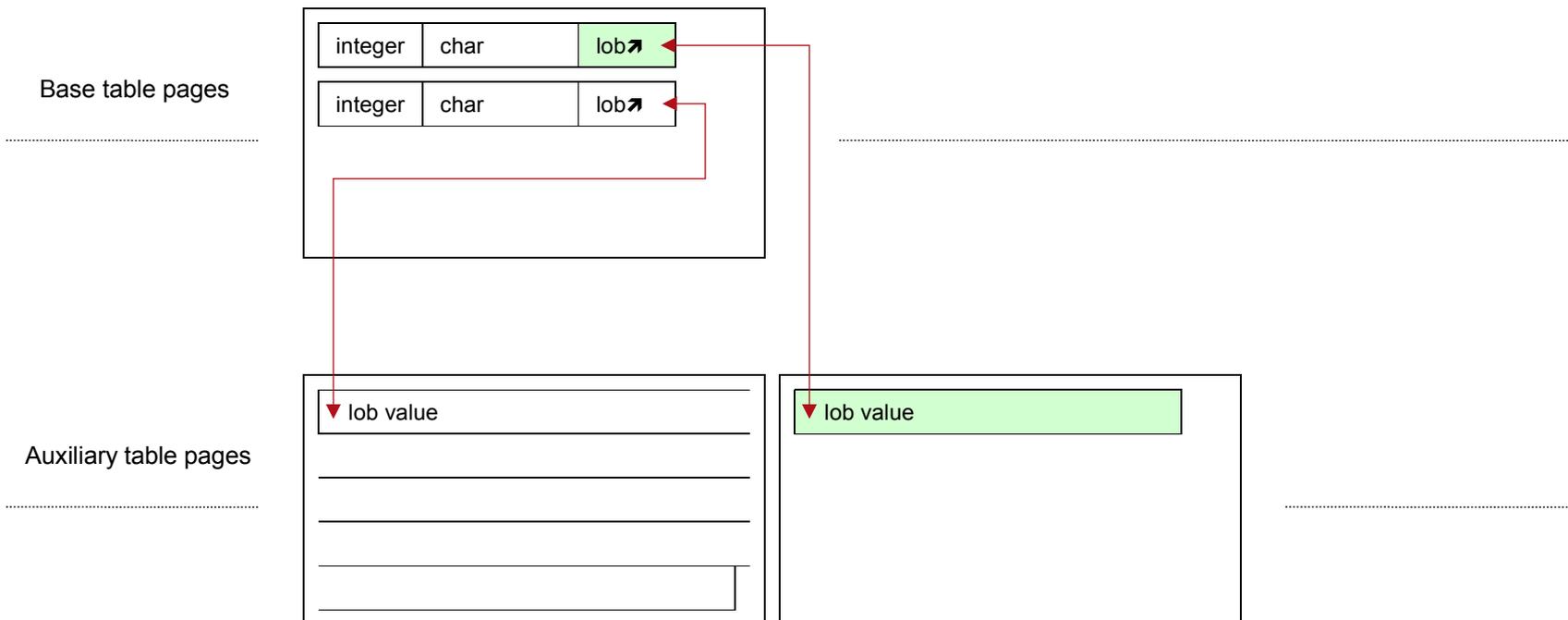
1. Create tablespace with COMPRESS NO
2. Populate table by INSERTs
3. Alter tablespace to COMPRESS YES
4. Reorganize tablespace

# SLOBs - 'Small' Large Objects

DB2 9

What is the largest inhibitor for more intensive use of LOBs?

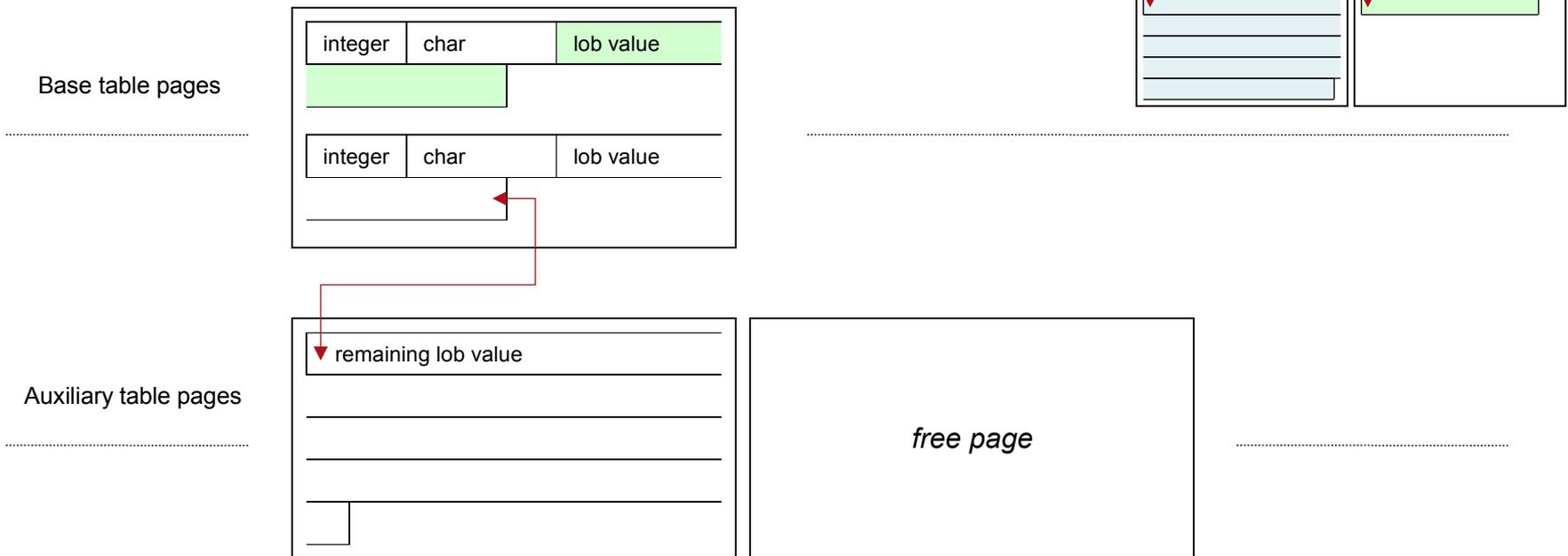
Performance and space utilization for LOBs with relatively smaller size.



- Operations on LOB columns always require access to additional pages which drives higher CPU, memory and I/O utilization
- LOB column, no matter how small the value might be, is stored in a separate tablespace

# LOBs Inlining

DB2 10 supports collocating the entire LOB column or a part of it with other columns within the base row.



- Improved elapsed and CPU time through fewer getpages and I/Os
- Improved space use (both disk and memory)
  - Completely inlined LOB values do not require pages in LOB tablespace (one per LOB!)
  - Inlined LOB values are subject to regular data compression
    - LOB tablespace cannot be compressed!
- Index key on a substring of the inlined part is allowed

# LOBs Inlining - How to Control It?

- LOB Inlining requires UTS
- LOB\_INLINE\_LENGTH zparm
  - Specifies the default inline length for any new LOB column
    - Valid values: 0 (default) to 32680 bytes
- INLINE LENGTH clause on CREATE DISTINCT TYPE
- INLINE LENGTH clause on CREATE TABLE
  - Overrides the value specified in zparm or distinct type definition
- INLINE LENGTH clause on ALTER TABLE
  - When adding new LOB column
  - When changing the inline length of the existing LOB column
  - REORG materializes change for existing rows
- Full support for DEFAULT values on LOB's inline part

## Difficulties at Influencing Optimizer Decisions

What do you do if the access path is suboptimal despite applying all the conventional tuning techniques:



DB2 9

- Providing appropriate indexes
- Collecting the latest statistics and refreshing the catalog
- Using appropriate reoptimization option
- Correctly setting relevant zparms (STARJOIN, MAX\_PAR\_DEGREE ...)
- Assess use of VOLATILE
- Read again relevant sections from the *Managing Performance* guide

### Update catalog statistics or Use optimization hints

- In any case, please report the incident to the IBM Service
- Updating catalog statistics (*lying* to the Optimizer) is risky and not recommended
  - It typically has broader (and often negative) effects than needed and intended
  - It's challenging and unpredictable
  - It creates administrative overhead
- Using optimization hints is preferred, but ...
  - Difficult to correlate the statement with its hints specification: QUERYNO can change
  - Particularly difficult for dynamic SQL

Additional challenge: zparms and bind options affecting the access path selection have too broad scope

# Statement-level Optimization Hints

DB2 10 introduces Access Path Repository, i.e. a set of tables in the SYSIBM schema that influence access path selection:

- SYSQUERY, SYSQUERYPLAN, SYSQUERYOPTS
- Using query text to match statement with its hint specification
  - No need to change application
  - Applies to any statement in the system with the same statement text
  - Scope can be either system or package
- How to specify statement-level hints
  - Insert a row into DSN\_USERQUERY\_TABLE user table containing QUERYNO, query text and optional data such as user, collection, package etc.
  - Populate PLAN\_TABLE with desired hints – use the same QUERYNO from
  - Issue BIND QUERY command
  - You can also delete specified hints by issuing FREE QUERY command
- The same mechanism can be used to limit the scope of selected optimizer relevant parameters to individual statements
  - zparms: STARJOIN, SJTABLES, MAX\_PAR\_DEGREE
  - Package bind options: REOPT, DEF\_CURR\_DEGREE

# Literals Reduce Dynamic Statements Cache Efficiency

DB2 9

Applications programmers use literals in the statement text :

- Intentionally
  - To force new access path selection for each set of different values
- Unintentionally
  - Bad coding practice
  - Forced by the development tooling

Unintentional use results in dynamic statements cache inefficiency

- A short (cost efficient) prepare is possible only if the new statement matches a cached statement character-for-character
  - Any difference in literal values results in a full (expensive) prepare
- New copies of statements that could have shared already cached statement 'thrash' dynamic statement cache

Is it possible to avoid the thrashing?

Yes, by using REOPT ALWAYS, but at the cost of not having statements caching at all!

# Concentrating Cached Statements

DB2 10

DB2 10 introduces an option to reuse a cached, previously prepared statement irrespective of literal values

- CONCENTRATE STATEMENTS WITH LITERALS
  - New option for ATTRIBUTES on PREPARE
  - CONCENTRATE STATEMENTS OFF (default) causes pre-V10 behavior
- Before a prepared statement is cached each literal is replaced by a single '&'
  - Additionally, DB2 removes the trailing blanks that follow the statement
  - '&'s are shown instead of literals in instrumentation (e.g. IFCID 317)
- Subsequent prepares of the same statement with different literals can result in short prepares
  - The exact match has a precedence over matching with '&'
  - The literals must be 'reusable' in the prepare context
  - Mixture of parameter markers '?' and literals results in the pre-V10 behavior
- Monitoring support
  - Values 'R', 'D' or ' '
    - for new column LITERAL\_REPL in DSN\_STATEMENT\_CACHE\_TABLE
    - for new field in IFCID 316
  - New statistics counters

## How To Drop Partition in DB2?



Strictly speaking, it cannot be done.

- A partition can be emptied by DELETE or LOAD REPLACE
- ... but it stays around forever

There is a special case when a partition can be effectively dropped.

- If the partition to be dropped is the very first logical partition, ROTATE effectively drops that partition and creates a new one in its place.

```
ALTER TABLE ROTATE PARTITION  
FIRST TO LAST  
ENDING AT constant | MAXVALUE | MINVALUE
```

In all other cases you are faced with a growing number of partitions

- Only exceptionally the growth can be slowed down by redistributing data after changing key ranges
- Associated with significant operational complexity

## ROTATE n to LAST as Means to Drop the n<sup>th</sup> Partition

DB2 10 extends the ROTATE PARTITION scope

DB2 10

```
ALTER TABLE ROTATE PARTITION  
FIRST | integer TO LAST  
ENDING AT constant | MAXVALUE | MINVALUE
```

**integer** specifies the physical partition that will be:

- reset, i.e. emptied
- it's limit key set to value specified at ENDING AT
- FIRST continues to refer to the first logical partition

DB2 10 also improves availability by not requiring reorganization for a number of partition altering operations if the involved partition is empty.

## Lock Suspension Wait Indicators Overload



Most available and reliable indicators of performance inhibitors caused by concurrency problems are provided by DB2 Accounting trace Class 3 switch.

- It's a switch because it does not create new trace records: it rather controls what details are included in other traces such as IFCID 3, 148, 316, ...
- It provides frequency of occurring and accumulated time spent waiting for resuming processing after any of the following suspensions:
  - IRLM lock/latch and DB2 internal latch
  - Page latch
  - Drain lock
  - Claim release
  - Numerous data sharing specific serialization events



So, what's the problem?

The most common suspension type: waiting for lock request includes waits for entirely unrelated events: DB2 internal latches

- One can use Statistics trace to assess the number of DB2 internal latch suspensions
- But, the scope of reporting (system vs. thread or statement) does not match

## New Class 3 Waits Indicators: DB2 Internal Latches



DB2 10 separates waits for DB2 internal latches into dedicated Class 3 indicators.

- Accumulated time spent waiting for DB2 internal latch suspensions
- Number of times DB2 internal latch suspensions occurred
- The existing wait indicators are used for IRLM lock and latch suspensions only

Statistics trace is still needed to assess which type of internal latch suspensions occurred, e.g.

Latch Class	Description
LC06	Index tree P-lock latch contention caused most likely by splits of GBP-dependent index pages
LC07	Dependency Manager Hash Table
LC11	Generating Identity Column
LC14	BP LRU chain
LC19	Logging
LC24	BP LRU chain
LC25	EDM Pool hash chain
LC32	Storage Manager Pool Header

RoT: more than 10000 per sec is considered high.

## Monitoring of Static SQL Statements

Prior to DB2 10 there is no straightforward way to monitor individual statements that are statically bound

- DB2 provides rich performance details, but aggregated by package or plan
- Individual statements can only be traced by costly performance traces that are complicated to format and monitor
- On the contrary, for dynamically prepared statements that use statement cache, DB2 provides easily obtained detailed performance data

DB2 9

As of DB10, the per-statement performance details are extended to statically bound statements as well.

- Performance Class 29
  - IFCID 400 as analogue to 318 and IFCID 401 as analogue to 316
- Available for statements currently in EDM Pool (via READS) and statements that got purged from EDM Pool (READA or SMF/GTF)
- Request can be filtered by:
  - *exceeding a given threshold*, e.g. number of executions, number of getpages, ...
  - *belonging to a top list by a given performance indicator*, e.g. elapsed time, CPU, time, ...
  - *matching a given statement ID* (single statement retrieval)
  - *executed for a given end-user ID, transaction/application ID, workstation name*
  - *refer to a specific table*
  - *bound after specific time or re-executed after specific time*

DB2 10

## Special Register CURRENT EXPLAIN MODE

Prior to DB2 10 performance monitoring and tuning of dynamic SQL statements was not ideally suited for application programmers

DB2 9

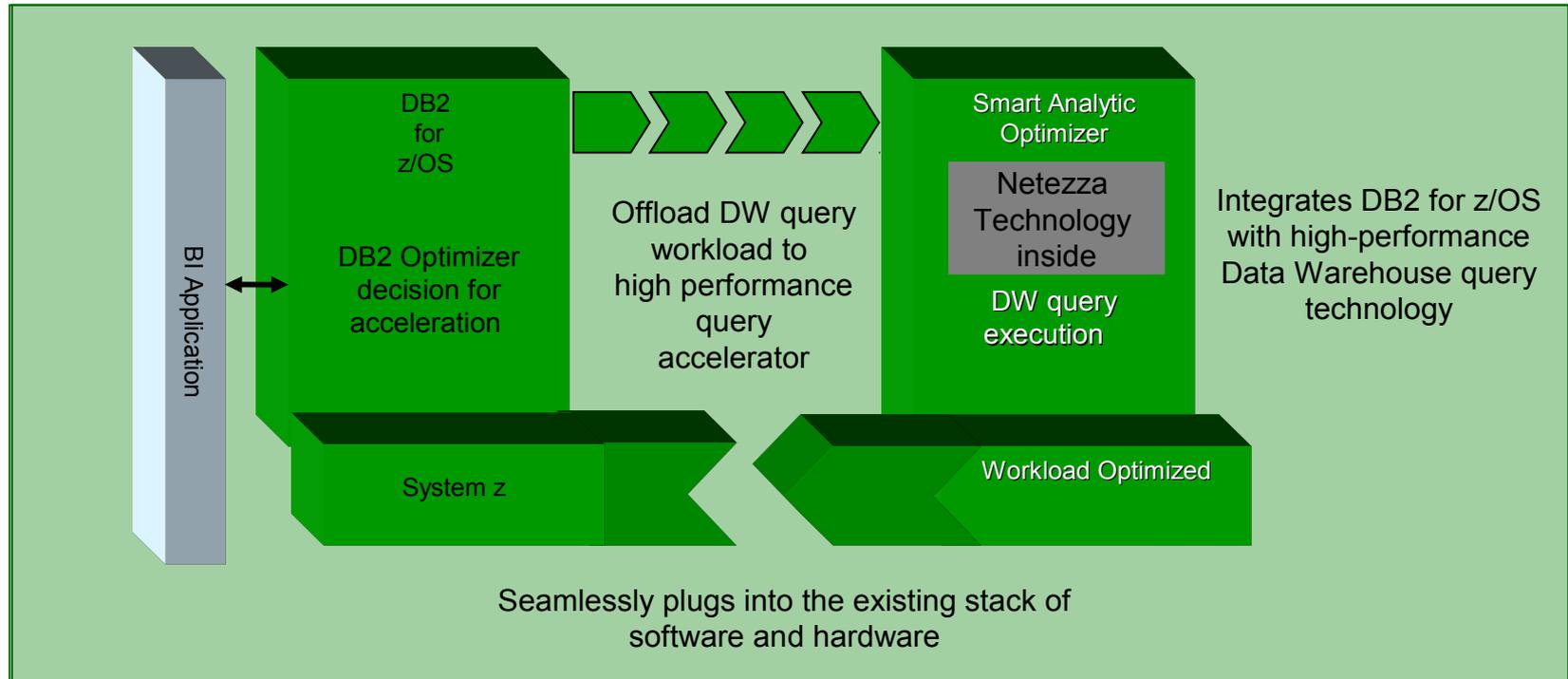
- There is no equivalent of EXPLAIN bind option that is provided for static SQL
  - Each monitored statement needs to be changed (adding STMTTOKEN)
  - Alternatively, time consuming and complex analysis of performance traces or DSN\_STATEMENT\_CACHE\_TABLE is needed
- Statement could be purged from the cache before having a chance to retrieve its performance characteristics

DB10 addresses these challenges

DB2 10

- Application programmers can turn on/off collection of access path details and performance data for dynamic SQL statements
  - SET CURRENT EXPLAIN MODE = NO | YES | EXPLAIN
  - JCC connection property: *currentExplainMode*
  - ODBC/CLI:
    - keyword *DB2Explain* in db2cli.ini file, or
    - Setting SQL\_ATTR\_DB2EXPLAIN with SQLSetConnectAttr() function
  - For YES and EXPLAIN, access path details are written in standard explain tables
    - This also applies to CACHEDYN=NO and REOPT(ALWAYS) cases
  - For YES, performance data are written in DSN\_STATEMENTS\_CACHE\_TABLE

# DB2 10 to Support IBM Smart Analytics Optimizer



## IBM Smart Analytics Optimizer key enhancements:

- Powerful query engine based on Netezza
- Significantly broadened query acceleration applicability
- 10-20 times increased data capacity
- Improved concurrent query execution
- Partition scope update option
- Unicode support

# Cool DB2 10 for z/OS Features for Enterprise Applications

par Namik Hrle IBM  
hrle@de.ibm.com