Konferencja Optymalny znaczy najlepszy

czyli, co nam dają nowe wersje oprogramowania?

Anna Olczak

DB2 – Strzał w dziesiątkę

Nowszy Silnik Bazy Danych

Jak migrować do DB2 V9 ?

.... A może warto do V10 ?

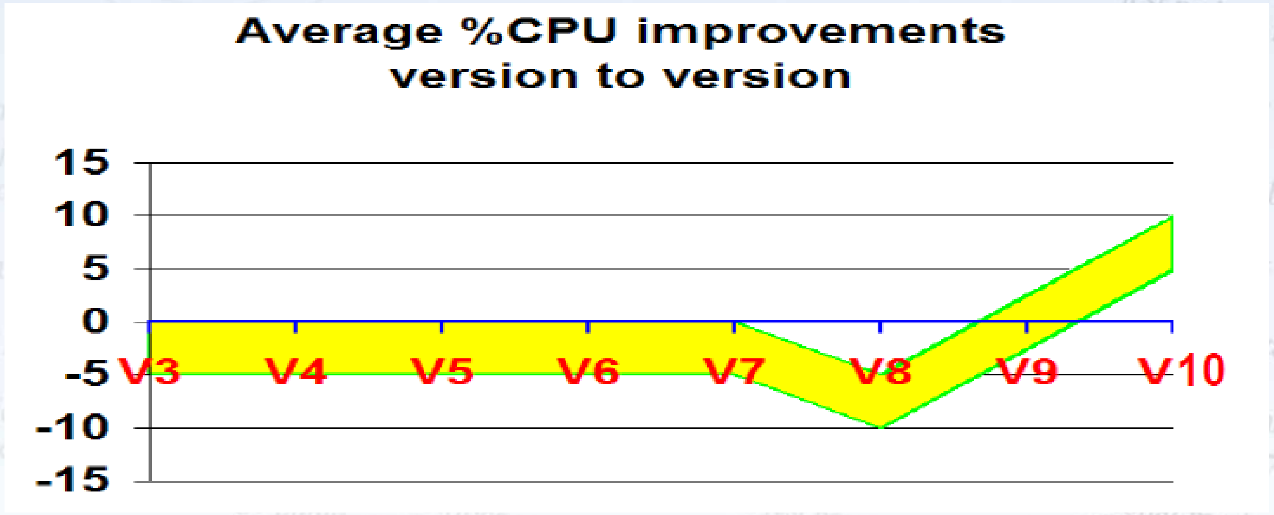
18-19 listopada 2010
Naruszewo, hotel Szkockie Ranczo

Legal Information

Wdrażajmy nową wersję !!!! Kiedy ? JAK?

	DB2 V8	DB2 9	DB2 10
GA	26.03.2004	16.03.2007	22.10.2010
EOS	30.03.2012		

Wykorzystanie CPU DB2 10 z/OS



Na ogół redukcja zużycia CPU 5%-10% po wykonaniu bindowania
Dla niektorych worklodów nawet 20% redukcji

Poprzedni skok dla DB2 2, przez następne lata inne priorytety
- wydajność zapytań
- skalowalność
- przepustowość

Skalowalność, dostępność, wydajność

V09

V10

- **5-6 większa ilość wątków** w jednym subsystemie (ponieważ 80%-90% pamięci wirtualnej przeniesiono powyżej 32GB)
- Większa konkurencyjność w dostępie do **katalogu** i programów **usługowych**
- 10 - krotnie więcej użytkowników subsystemu
- Większa funkcjonalność on-line w DDL
- Opóźniony ALTER

DB2 SQL

z z/OS V7
common
luw Linux, Unix & Windows V8.2

z Range partitioning

common Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, Call from trigger, statement isolation

luw Updateable UNION in Views, ORDER BY/FETCH FIRST in subselects & table expressions, GROUPING SETS, ROLLUP, CUBE, INSTEAD OF TRIGGER, EXCEPT, INTERSECT, 16 Built-in Functions, MERGE, Native SQL Procedure Language, SET CURRENT ISOLATION, BIGINT data type, file reference variables, SELECT FROM INSERT, UPDATE, or DELETE, multi-site join, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, MDC

DB2 SQL

z z/OS V8
common
luw Linux, Unix & Windows V8.2

z Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE for SQL, join across encoding schemes, IS NOT DISTINCT FROM, Session variables, range partitioning

common Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, **Star Join Sparse Index**, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT

luw Updateable UNION in Views, ORDER BY/FETCH FIRST in subselects & table expressions, GROUPING SETS, ROLLUP, CUBE, INSTEAD OF TRIGGER, EXCEPT, INTERSECT, 16 Built-in Functions, MERGE, Native SQL Procedure Language, SET CURRENT ISOLATION, BIGINT data type, file reference variables, SELECT FROM UPDATE or DELETE, multi-site join, MDC

DB2 SQL

z z/OS V9
common
luw Linux, Unix & Windows V9

z Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE for SQL, join across encoding schemes, IS NOT DISTINCT FROM, Session variables, **TRUNCATE, DECIMAL FLOAT, VARBINARY, optimistic locking, FETCH CONTINUE, ROLE, MERGE, SELECT from MERGE**

common Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, **UPDATE or DELETE, INSTEAD OF TRIGGER, Native SQL Procedure Language, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect and fullselect, caseless comparisons, INTERSECT, EXCEPT, not logged tables**, **range partitioning, compression**

luw Updateable UNION in Views, GROUPING SETS, ROLLUP, CUBE, 16 Built-in Functions, SET CURRENT ISOLATION, multi-site join, MERGE, MDC, **XQuery**

DB2 SQL 2010

z z/OS **10**
common
luw Linux, Unix & Windows **9.8**

z: Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE SQL, join across encoding schemes, IS NOT DISTINCT FROM, VARBINARY, FETCH CONTINUE, MERGE, SELECT from MERGE, **data versioning, access controls**

common: Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, UPDATE or DELETE, INSTEAD OF TRIGGER, Native SQL Procedure Language, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect & fullselect, caseless comparisons, INTERSECT, EXCEPT, not logged tables, OmniFind, spatial, range partitions, data compression, session variables, DECIMAL FLOAT, optimistic locking, ROLE, TRUNCATE, index & XML compression, created temps, **inline LOB, administrative privileges, implicit cast, date/time changes, currently committed, moving sum & average, index include columns, PureScale**

luw: Updateable UNION in Views, GROUPING SETS, ROLLUP, CUBE, more Built-in Functions, SET CURRENT ISOLATION, multi-site join, MERGE, MDC, XQuery, XML enhancements, array data type, global variables, even more vendor syntax, temp table compression

DB2 10 – Kluczowe zmiany - wydajność i dostępność

- 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
- ...

- 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 enhacements
- ...

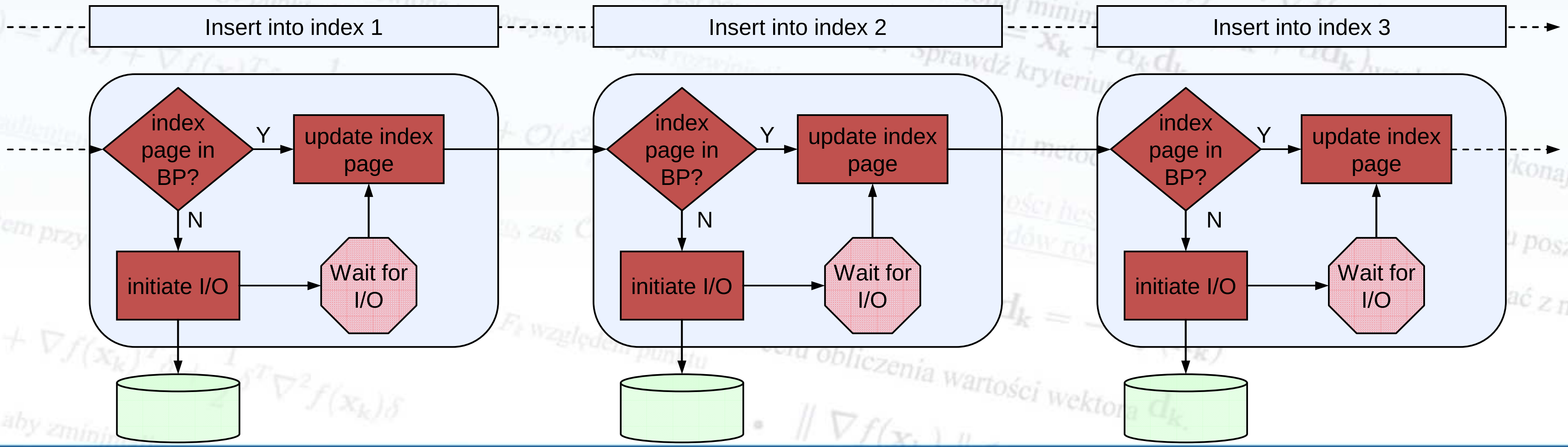
Insert Performance Bottlenecks - Part 1

DB2 9

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

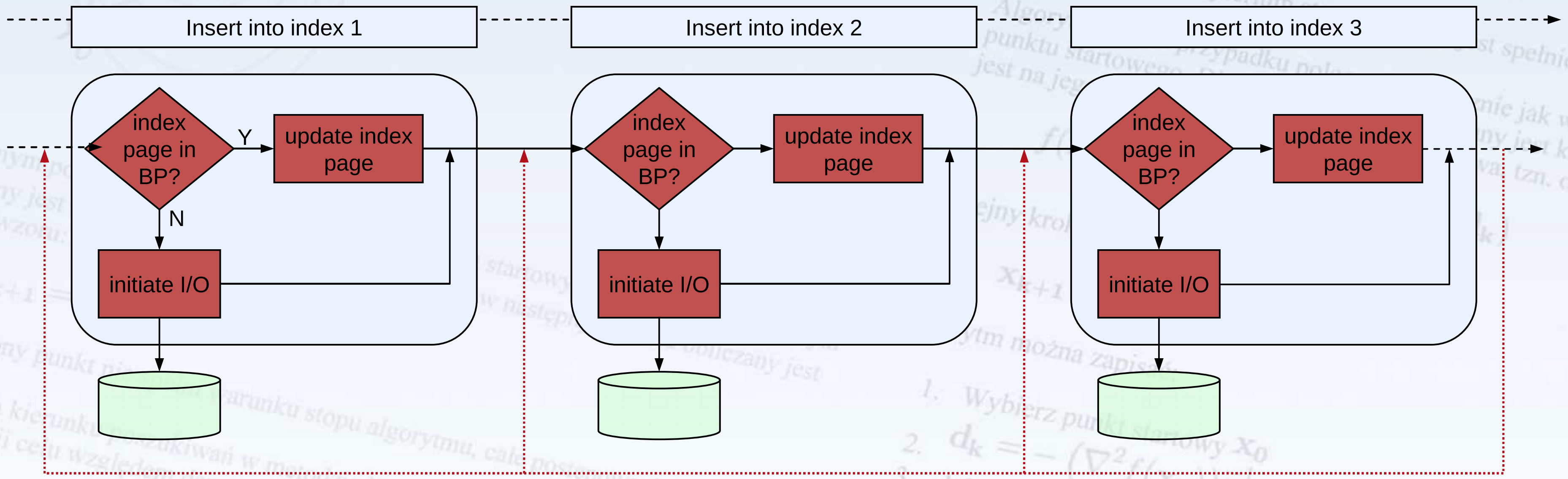
- Locating page to insert?
- Contention, e.g. on space map page, particularly in data sharing?
- Logging?
- Writing inserted pages?
- ☞ Read I/O?

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



Index Read I/O Parallelism at Insert

DB2 10



- 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 3
b) Space usage efficiency 2
c) Speed of subsequent queries with range predicates 1

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’ behaviour, 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?

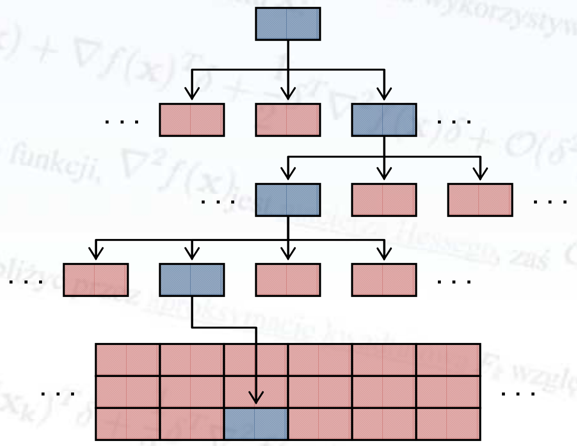
Equal-unique index access path!

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:

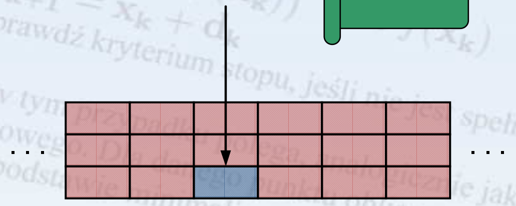


- Large indexes result in increased number of getpages:
  - number of getpages = *n* + 1, where *n* is the index depth
- Likelihood of read I/Os increases
- Disorganized indexes exacerbate the problem

Hash Access to Data

DB2 10

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:

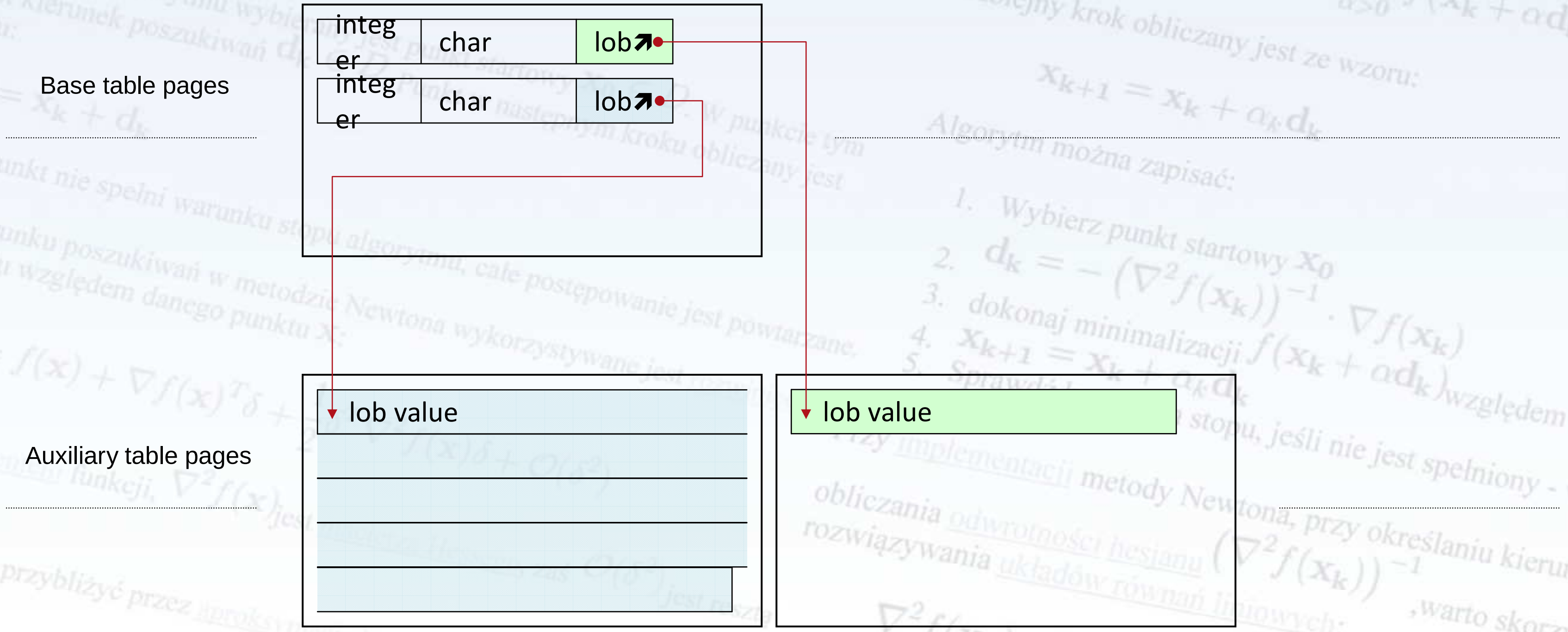
1. Create tablespace with COMPRESS NO
2. Populate table by INSERTs
3. Alter tablespace to COMPRESS YES
4. Reorganize tablespace

SLOBs - ‘Small’ Large Objects

DB2 9

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

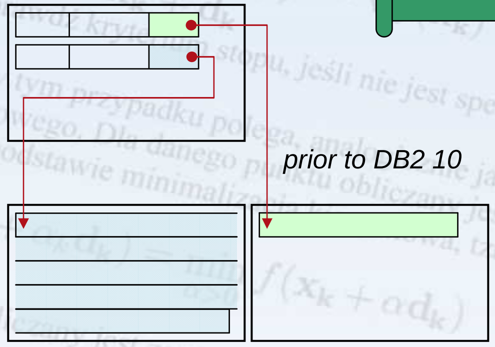
Performance and space utilization for LOBs with relatively smaller size.



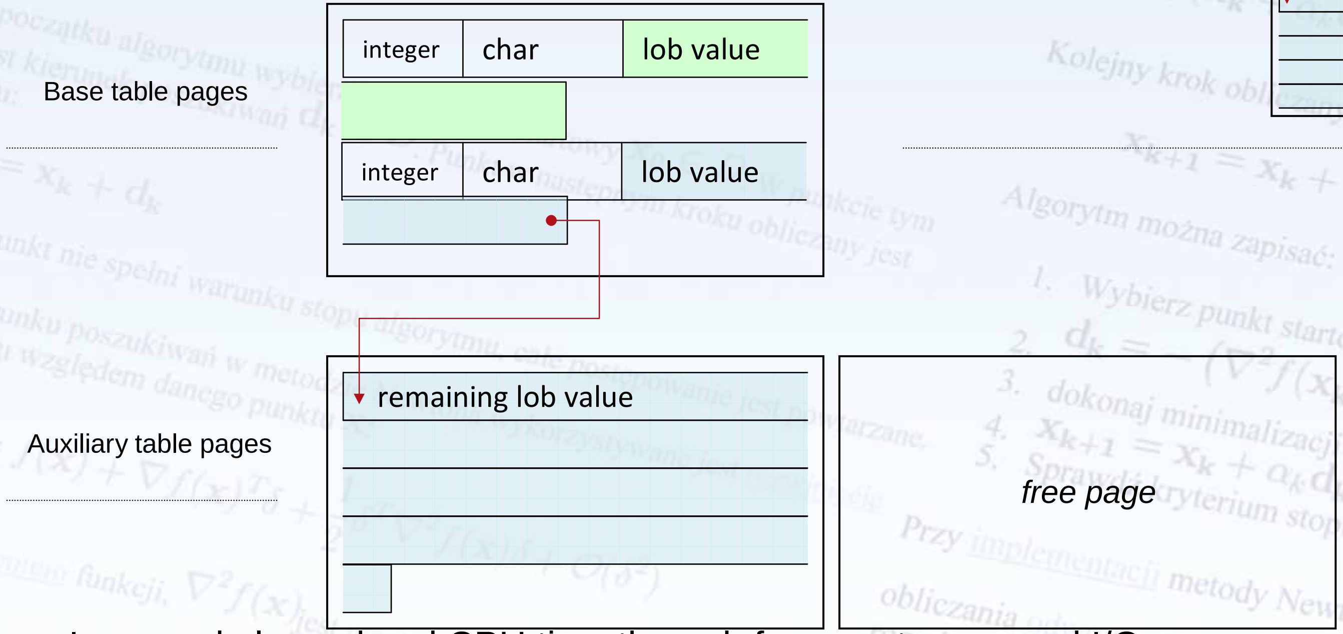
- Operations on LOB columns always require access to additional pages which drives higher CPU, memory and I/O utilization
- LOB column, no matter how small the value might be, is stored in a separate tablespace

LOBs Inlining

DB2 10



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?

DB2 10

- 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 LENGHT clause on CREATE DISTINCT TYPE
- INLINE LENGHT 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
    - REORG SHRLEVEL REFERENCE is required
- Full support for DEFAULT values on LOB's inline part

Lock Contentions on DB2 Catalog

DB2 9

The following processes can create incompatible locks on DB2 catalog, leading to lock suspensions, timeouts and deadlocks:

- Utilities (most often RUNSTATS)
- DDL
- Prepares (Bind)

What is the key reason for the increased lock contention?

Page level locking that is used for a number of DB2 catalog tablespaces

DB2 V7 introduced lock size ROW for selected catalog tablespaces, but the rest contain so called 'links' and could not have been enabled for row level locking:

- SYSDBASE
- SYSPLAN
- SYSVIEWS
- SYSDBAUT
- SYSGROUP

Row Level Locking for All Catalog Tablespaces

DB2 10

DB2 10 removes dependency on the 'links' and extends row level locking option to all the catalog tablespaces.

Independently of this DB2 10 introduces the following enhancements:

- Simplifies tablespaces growth management by converting a number of catalog and directory tablespaces to being partitioned by growth
- Automates underlying data set management by converting all the DB2 catalog and directory tablespaces into SMS-managed
- Uses LOB data type to consolidate SQL statement text that is split across multiple rows in a number of DB2 catalog tables
- Combines SYSUTIL and SYSUTILX into a single table

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
  - Always the case in SAP applications
- 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
  - This is why SAP in most cases uses parameter markers
- 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 behaviour
- Before a prepared statement is cached each literal is replaced by a single '&'
  - Additionally, DB2 removes the trailing blacks 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 behaviour
- 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?

DB2 9

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 nth 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.

Long Running Readers Prevent REORG to Complete

DB2 9

Even threads that do nothing else but reading without locking (isolation level UR) prevent REORG completion.

- They use locking serialization mechanism called ‘claims’
- Claims need to be removed before final stages of REORG can proceed

Tips for handling the situation:

- Use appropriate REORG drain specification options
  – DRAIN_WAIT, RETRY, RETRY_DELAY
- Identify long running, non-committing readers
  – Set zparm LRDRTHLD to a non-zero value (default is zero, which means no reporting)
  – DB2 will generate IFCID 313 for each thread which holds a claim without committing for longer than specified LRDRTHLD value
  – Capture and format IFCIDs 313 (or use suitable monitoring tool)
- Identify threads for a given object for which REORG cannot proceed and cancel them (if appropriate)

```
DISPLAY DATABASE(dbname) SPACENAM(tsname) CLAIMERS or LOCKS
```

REORG's Weapon Against Long Running Readers

DB2 10

```
REORG TABLESPACE | INDEX ...
          FORCE NONE | READERS | ALL
```

FORCE specifies which threads, if any, will be cancelled by REORG at the last attempt (last RETRY) to take over control of an object.

NONE	No claimers are cancelled. Existing behaviour. Default.
READERS	Read claimers cancelled at DRAIN ALL
ALL	All claimers cancelled at DRAIN ALL or WRITERS

For REORG to succeed, the cancelled threads must free the objects (release claims) before DRAIN_WAIT time expires.

Long running readers detection changes:

- System console message
- Default for zparm LRDRTHLD is 5 min

Refreshing Catalog Statistics: What, Which, When?

DB2 9

DBAs spend lots of time managing catalog statistics.

- What objects have stale statistics?
- Which statistical data needs to be collected for a given object?
- When should the jobs for collecting and refreshing statistics be run?

The common reason for answering these questions in optimal way is:

Collecting catalog statistics is resource-intensive and not transparent to other concurrent database activity.

- Often there is a very large number of objects, not all have stale statistics
  - Real Time Statistics is the answer to this challenge
- Different objects might need different level of statistics details
  - Associated with time-consuming analysis of workload for each of the objects
  - Tools can help
- The jobs to collect and refresh statistics should ideally be as transparent to the concurrent transaction and batch processing as possible
  - CPU contention
  - Catalog locking contention

Building Blocks for Automating RUNSTATS

DB2 10

- The new Auto-Stats feature provides the following:
- Profile support for RUNSTATS utility:
  - RUNSTATS TABLESPACE.... SET / USE / UPDATE / DELETE PROFILE
- New stored procedures
  - ADMIN_UTL_MONITOR checks need for new/better statistics
  - ADMIN_UTL_EXECUTE collects new statistics using ADMIN_UTIL_SCHEDULE
- Control tables:
  - SYSIBM.SYSTABLES_PROFILES contains the RUNSTATS options at table level
  - SYSIBM.SYSAUTOTIMEWINDOWS defines when autonomic procedures can be run
  - SYSIBM.SYSAUTOALERTS is populated when ADMIN_UTL_MONITOR detects that an action needs to be scheduled for execution
- DB2 Admin Scheduler can be used to automatically run the stored procedures

Online Schema Change - The Journey Continues

DB2 10

DB2 V8 and 9 focused on online changes of table attributes, while DB2 10 on online changes of tablespace and index structural attributes.

The following ALTERations are now possible:

- Page size and buffer pool assignment
- DSSIZE, SEGSIZE, MEMBER CLUSTER
- Tablespace type:
  - Simple (single table) into UTS PBG
  - Segmented (single table) into UTS PBG
  - Classic partitioned into UTS PBG
  - UTS PBR into UTS PPG
  - UTS PBG into hash table (but overflow index in RBDP state)

New concept, *Pending Definition Change*, is introduced to maximize availability of altered objects.

- ALTER statement is put on a 'to-do list' and returns sqlcode +610
- Object placed in AREOR state: REORG Advised
- Change of mind allowed: ALTER ... DROP PENDING CHANGES
- Next REORG materializes changes