IBM Information ON Demand 2010

Create Insight. Transform. Go Beyond.

TSB-2894A

What's coming from the optimizer in DB2 10 for z/OS?

Terry Purcell

Senior Technical Staff Member

IBM Silicon Valley Lab

Housekeeping

- We value your feedback - don't forget to complete your evaluation for each session you attend and hand it to the room monitors at the end of each session
- Overall Conference Evaluation will be provided at the General Session on Friday
- Visit the Expo Solutions Centre
- Please remember this is a 'non-smoking' venue!
- Please switch off your mobile phones
- Please remember to wear your badge at all times

IBM Disclaimer

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

Disclaimer/Trademarks

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.

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

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.

Trademarks The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the pages of the presentation:

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both: AIX, AS/400, DataJoiner, DataPropagator, DB2, DB2 Connect, DB2 Extenders, DB2 OLAP Server, DB2 Universal Database, Distributed Relational Database Architecture, DRDA, eServer, IBM, IMS, iSeries, MVS, Net.Data, OS/390, OS/400, PowerPC, pSeries, RS/6000, SQL/400, SQL/DS, Tivoli, VisualAge, VM/ESA, VSE/ESA, WebSphere, z/OS, zSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Other company, product, or service names may be trademarks or service marks of others.

Agenda

- Access path management
  - Dynamic Statement Cache Enhancements
  - Access Path Stability
  - Instance Based Statement Hints
- Query performance improvements
  - Safe Query Optimization
  - Aggressive View Merge
  - IN List Processing
  - SQL Pagination
  - Parallelism Enhancements

Dynamic Statement Cache

- Introduced in DB2 V5
- Re-uses SQL and access path
  - If identical SQL string
  - If same user,…
- Avoids full prepare (like a BIND)
- Good programming practice to use parameter marker (?)
  - ? are parameter markers
  - Ensures SQL is always the same
- Not all programs use ?
  - Ruby On Rails generates literals not ?
  - So SQL can not be re-used in Cache

Literal Replacement

- Dynamic SQL with literals can now be re-used in the cache
  - Literals replaced with & (similar to parameter markers but not the same)
- To enable either you:-
  - Put CONCENTRATE STATEMENTS WITH LITERALS in the ATTRSTRING in the PREPARE
  - Or set LITERALREPLACEMENT in the ODBC initialization file
  - Or set the keyword enableLiteralReplacement='YES' in the JCC Driver
- Lookup Sequence
  - Original SQL with literals is looked up in the cache
  - If not found, literals are replaced and new SQL is looked up in the cache
    - Additional match on literal usability
    - Can only match with SQL stored with same attribute, not parameter marker
  - If not found, new SQL is prepared and stored in the cache

Literal Replacement …

- Example:

  SELECT BALANCE FROM ACCOUNT WHERE ACCOUNT_NUMBER = 123456

  – This would be replaced by

  SELECT BALANCE FROM ACCOUNT WHERE ACCOUNT_NUMBER = &

- Performance Expectation
  - Using parameter marker provides still best performance
  - Biggest performance gain for small SQL with literals that have a cache hit now, but did not before
  - Determined access path is not optimized to provided literals,
    - Need to use REOPT for that purpose

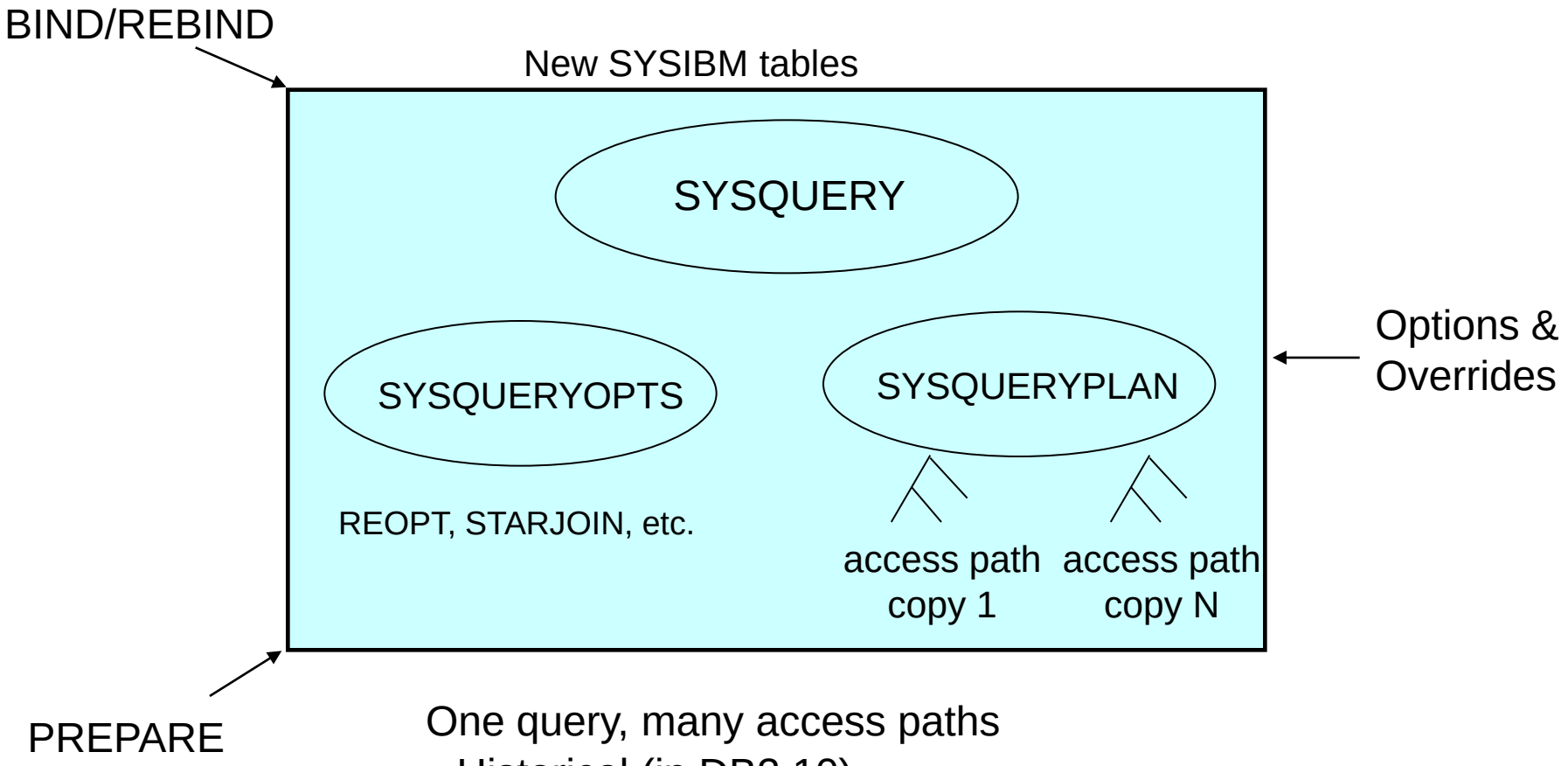
Access Path Stability

- Query optimization depends on many inputs.
  - Some of which may be missing/incomplete.
  - Small changes in the environment can change (degrade) an access path.
- Re-optimization is not always the preferred solution.
  - Static SQL typically insulated from erratic access path changes.
  - Dynamic SQL far more prone to access path degradation.
- When access paths change due to re-optimization (REBIND or PREPARE), often difficult to revert to a prior access path.
- In DB2 V9 (APAR PK52522)
  - Solution provided for fallback to prior path static SQL across REBIND.
- DB2 10 for z/OS
  - extends the V9 solution unifying the treatment of static and dynamic.

Management of static and dynamic access paths

- Ability to
  - Capture access paths for static and dynamic SQL queries in the access path repository
  - Save multiple copies of access paths
  - Switch between different copies of access paths of the same query
  - Manually control when captured dynamic SQL queries would be re-optimized
  - Regenerate runtime structures without changing access paths
    - Important for mass REBIND at DB2 migration
  - Perform before/after access path comparisons

Access Path Repository



One query, many access paths

- Historical (in DB2 10)
- Parametric (eventually)

Capturing Access Paths …

- Governed by two settings, PLANMGMT and PLANMGMTSCOPE, set via:
  - DSNZPARMs
  - BIND and REBIND options,
  - Profiling attributes
- PLANMGMT
  - OFF - No access path information is captured.
  - ON - Access path information is captured in the repository. However, no historical access paths are retained.
  - BASIC - Access path information is captured in the repository, plus one old access path is retained. Referred to as the PREVIOUS copy.
  - EXTENDED - Access path information captured in the repository, plus two old access paths are retained - PREVIOUS and ORIGINAL copies.
- PLANMGMTSCOPE
  - ALL - Includes static and dynamic SQL queries.
  - STATIC - Only include static SQL queries. This is ZPARM the default.
  - DYNAMIC - Only include dynamic SQL queries

Stabilizing Static & Dynamic SQL

- Capture access paths of dynamic SQL queries tagged as ‘critical’
  - I.e. Queries with PLANMGMT <> OFF
  - These queries are considered “stabilized”
- At PREPARE, check if an access path was previously captured
  - YES -- it is used
  - NO -- the query is compiled afresh. The resulting access path could get captured if query is a candidate for stabilization.
  - Once prepared, the statement could be cached
- Notes
  - No application changes required.
  - Stabilized access paths immune to changes in configuration, statistics, etc
  - Dynamic SQL queries behave like Static SQL
  - Fast lookup/matching algorithms minimize overhead.
  - Think on this as a “persistent dynamic SQL cache”

REBINDing Dynamic SQL

- Captured access paths may not stay optimal forever
  - Periodic tuning/maintenance may open up more optimal access paths
- New mechanism to REBIND captured dynamic SQL statements
  - Generate new access paths, either immediately or at next full PREPARE
  - Previous access paths can be retained as a backup
  - Similar to REBIND PACKAGE
- New mechanism to switch to a prior access path
  - Used when new access paths cause performance regressions
  - Similar to REBIND PACKAGE … SWITCH(PREVIOUS/ORIGINAL)
- Support varied scope
  - All queries
  - Queries that originated from a package
  - Queries that any user-definable criteria

Access Path "Fallback"

- For each stabilized package/query, DB2 retains a few copies of old query plans
  - CURRENT – The active copy
  - PREVIOUS – The copy in use before the last REBIND. Transient.
  - ORIGINAL – The oldest/first query plan. Never overwritten.
- When query plans cause performance regressions …
  - A DBA can revert to a prior query plan

- Static SQL
  - Reverts back to an older copy of the one or more package
  - REBIND PACKAGE (HRCOLL.*) … SWITCH(PREVIOUS)
- Dynamic SQL
  - Reverts back to an older copy of a dynamic SQL query
  - REBIND QUERY … SWITCH(PREVIOUS | ORIGINAL)

  - REBIND QUERY FILTER('REGRESSED') … SWITCH(ORIGINAL)
  - REBIND QUERY PACKAGE(HRCOLL.*) … SWITCH(PREVIOUS)

Access Path Reuse / Compare For Packages

- Ability to BIND/REBIND a package for reasons other than access path improvements
  - Due to service fixes that require the regeneration of runtime structures (REBIND) … ++HOLDs
  - Due to application changes (BIND)
- How do we retain the same underlying access paths, and hence, minimize impact?
  - **BIND/REBIND PACKAGE … APREUSE(YES)**
    - For REBIND, DB2 attempts to reuse prior access paths
    - For BIND, DB2 attempts the reuse prior access paths for any queries that haven't changed
  - **BIND/REBIND PACKAGE… APCOMPARE(WARN | ERROR)**
    - DB2 issues a warning/error for each statement that has an access path change

AP Reuse/Compare – Anticipated Scenario

- Mass rebind recommended after DB2 migration
  - 1st reason to regenerate runtime structures under new release
    - Regain loss of runtime optimizations such as SPROCs (approx 7% CPU)
  - Concern is access path regression
- Suggested scenario
  - REBIND APREUSE(YES) APCOMPARE(ERROR)
    - Allow DB2 to attempt to reuse the prior path (not guaranteed)
    - Fail the REBIND if access path changes
  - Anticipated that majority of REBINDs will succeed
    - Regaining lost CPU due to invalidated runtime structures
    - Mitigating risk of access path regression
  - After migration stabilizes
    - Explore new access path possibilities

Getting information on packages

- SYSIBM.SYSPACKCOPY
  - New catalog table
  - Hold SYSPACKAGE-style metadata for any previous or original package copies
  - No longer need to SWITCH to see information on inactive copies
- EXPLAIN PACKAGE
  - Extract PLAN_TABLE information for one or more packages and their copies
  - The package/copy must be created on DB2 9 or later
  - Useful if you didn’t BIND with EXPLAIN(YES)

Freeing SQL Access Paths

- Static SQL
  - Previous existing FREE PACKAGE command
  - PLANMGMTSCOPE supported in DB2 10
    - If PLANMGMTSCOPE not specified or PLANMGMTSCOPE(ALL), package will be freed including data in access path repository
    - If PLANMGMTSCOPE(INACTIVE) specified, only older copies freed from catalog, directory, and access path repository
- Dynamic SQL
  - FREE QUERY – new command to purge one/more queries from the access path repository
    - FREE QUERY QUERYID(6557)
    - FREE QUERY QUERYID(ALL)
    - FREE QUERY FILTER('UNUSED')

Robust hints system

- Current limitations in hint matching
  - QUERYNO is used to link queries to their hints – a bit fragile
  - For dynamic SQL, require a change to apps – can be impractical
- New mechanisms being considered:
  - Associate query text with its corresponding hint … more robust
  - Hints enforced for the entire DB2 subsystem, irrespective of static vs. dynamic, etc.
  - Hints integrated into the same access path repository
- PLAN_TABLE isn't going away
- Only the "hint lookup" mechanism is being improved.

Robust hints system (cont.)

Steps to use new hints:

- Populate a user table called DSN_USERQUERY_TABLE with query text of one or more queries.
- Populate PLAN_TABLE with the corresponding hints,
- Run new command, BIND QUERY, to integrate the hint into the repository.
- FREE QUERY can be used to remove the hint.

Statement-level BIND options

- DB2 supports a few ways to influence query processing behavior
  - ZPARMs to influence all queries on the subsystem
  - BIND options for queries in a entire package
- However, a statement-level granularity is sometimes needed
  - E.g., a package bound with REOPT(NONE) may have one statement that needs REOPT(ALWAYS)
- New mechanisms for statement-level bind options:
  - Similar to mechanism used for hints
  - DSN_USERQUERY_TABLE can also hold per-statement options

Safe Query Optimization

Optimal Plan Generation Challenges

- Potential causes of sub-optimal plans
  - Insufficient statistics
  - Unknown literal values used for host variables or parameter markers
  - Unpredictable runtime resource availability
    - Specifically RID pool usage
- A plan determined by purely cost-based optimization may lack the robustness if estimates do not reflect reality.
- The Safe Query Optimization goal is to:
  - Generate safe and robust access paths
  - Fallback to workfile usage (for RID processing) at RID limit failure

Safe Query Optimization

- Bind/Prepare:
  - Optimizer will evaluate the risk associated with each predicate
    - For example: WHERE BIRTHDATE < ?
      - Could qualify 0-100% of data depending on literal value used
  - As part of access path selection
    - Compare access paths with close cost and choose lowest risk plan
- Runtime:
  - If a RID limit is reached
    - Overflow RIDs to workfile and continue processing
    - Avoid fallback to tablespace scan as in V9.
  - Work-file usage may increase
    - Mitigate by increasing RID pool size (default increased in DB2 10).

Index Exploitation with IN-list Predicates

Index Exploitation with IN-list Predicates

- For IN-list predicates, DB2 V10 adds
  - Transitive closure support for IN-list predicates
  - Matching on multiple IN-list predicates via single index access
  - List prefetch support
- IN-list predicates used extensively in some workloads
  - For example: SAP R/3

IN-list Predicate Transitive Closure (PTC)

```
SELECT *
FROM T1, T2
WHERE T1.C1 = T2.C1
  AND T1.C1 IN (?, ?, ?)

  AND T2.C1 IN (?, ?, ?) ←   Optimizer can generate
                             this predicate via PTC
```

- Without IN-list PTC (V9)
  - Optimizer in V9 will be unlikely to consider T2 is the first table accessed
- With IN-list PTC
  - Optimizer can choose to access T2 or T1 first.

IN-list Table - Table Type 'I' and Access Type 'IN'

- **The IN-list predicate will be represented as an in-memory table if:**
  - **List prefetch is also chosen, OR**
  - **More than one IN-list is chosen as matching.**

  - **The EXPLAIN output associated with the in-memory table will have:**
    - **New Table Type: TBTYPE – 'I'**
    - **New Access Type: ACTYPE – 'IN'**

```
SELECT *
    FROM T1
    WHERE T1.C1 IN (?, ?, ?);
```

QBNO	PLANNO	METHOD	TNAME	ACTYPE	MC	ACNAME	QBTYPE	TBTYPE	PREFETCH
1	1	0	DSNIN001(01)	***IN***	0		SELECT	***I***	
1	2	1	T1	I	1	T1_IX_C1	SELECT	T	***L***

IN-list Table - Table Type 'I' and Access Type 'IN'

QBNO	PLANNO	METHOD	TNAME	ACTYPE	MC	ACNAME	QBTYPE	TBTYPE
1	1	0	***DSNIN001(01)***	***IN***	0		SELECT	***I***
1	2	1	T1	I	1	T1_IX_C1	SELECT	T

- **The first row in the PLAN_TABLE is the access to the IN-list table.**
  - **DSNIN001(01) is the in-memory table name chosen by DB2 optimizer. "DSNIN" indicates that it relates to IN-list, "001" indicates the IN-list predicate number, and "(01)" indicates the query block number.**
  - **'IN' – new access type (ACTYPE)**
  - **'I' - new table type (TBTYPE)**

Outer Join View/Table Expression Merge

View/Table Expression Merge

- **More Merge scenarios for View/Table Expressions**
  - **Especially for View and Table Expressions involved in an outer join.**
- **Physical materialization (the alternative to Merge) is an overhead.**
  - **Can limit the join sequence considered.**
  - **Can limit the ability to apply predicates early in the processing sequence**
  - **The join predicate on a materialization work file can not be indexed.**
- **Generally a more aggressive Merge strategy for View/Table Expressions is preferable.**

Merge – CASE, VALUE, COALESCE expression on preserved side of an Outer Join.

- When there is a CASE, VALUE, or COALESCE expression on the preserved side of an outer join, DB2 will merge the view/table expression if the CASE, VALUE, COALESCE expression is not referenced. This avoids materialization.

```
SELECT A.C1, B.C1, A.C2, B.C2
FROM T1 ,(SELECT COALESCE(C1, 0) as C1 ,C2
            FROM T2 ) A              <--table expression 'A' will be Merged
          LEFT OUTER JOIN
                (SELECT COALESCE(C1, 0) as C1 ,C2
                 FROM T3 ) B       <-- B will be Materialized l
          ON A.C2 = B.C2
WHERE T1.C2 = A.C2;
```

Merge single table view / table expression on null-padded side which contains a subquery

- **For left/right outer joins, after query transformation, DB2 will allow subquery predicates in the on-clause.**
- **DB2 can Merge certain single table views / table expressions on null-padded side which contains a subquery.**
- **Performance may be improved due to materialization avoidance.**

Merge single table view / table expression on null-padded side which contains a subquery

```
SELECT *
FROM T1
   LEFT OUTER JOIN
      (SELECT *    <-- table expression contains subquery
       FROM T2
       WHERE T1.C1 = (SELECT MAX(T3.C1) FROM T3 ) <--subquery
       ) TE
ON T1.C1 = TE.C1;
```

```
SELECT *
FROM T1
      LEFT OUTER JOIN
      T2 AS TT                                  <-- table expression is merged
      ON TT.C1 = (SELECT MAX(TTT.C1)   <-- subquery ON-predicate
                  FROM T3 AS TTT)
      AND T1.C1 = TT.C1;
```

Other Merge table expressions opportunity

```
SELECT *
FROM T1,
        TABLE(
                SELECT * from T3 AS T2
                WHERE T1.C1= T2.C1
              ) AS X;        <-- table expression X is materialized in V9
```

```
SELECT T1.* , T2.C2          Not materialized in V10
FROM T1, T3 AS T2            Query rewritten
WHERE T1.C1 = T2.C2;
```

SQL Pagination

SQL Pagination targets 2 classes of OR queries:

- Cursor scrolling SQL
  - Retrieve next n rows
  - Common in COBOL/CICS and any screen scrolling application
- Complex OR predicates against the same index
  - Common in SAP

- In both cases:
  - The OR (disjunct) predicate refers to a single table only.
  - Each OR predicate can be mapped to the same index.
  - Each disjunct has at least one matching predicate.

Simple scrolling – Index matching and ORDER BY

- Scroll forward through the PHONEBOOK to obtain the next 20 rows from the current position – JONES, WENDY
  - Assumes index is available on (LASTNAME, FIRSTNAME)
- WHERE clause may appear as:

```
WHERE (LASTNAME='JONES' AND FIRSTNAME>'WENDY')
   OR (LASTNAME>'JONES')
ORDER BY LASTNAME, FIRSTNAME;
```

```
WHERE ((LASTNAME='JONES' AND FIRSTNAME>'WENDY')
   OR (LASTNAME>'JONES')
  AND (LASTNAME >= 'JONES'))
ORDER BY LASTNAME, FIRSTNAME;
```

← **Extra predicate required in V9 for matching index access**

Multiple OR ranges against same index

Assume an index on (LASTNAME, FIRSTNAME).

Given the following complex WHERE clause:

```
WHERE (LASTNAME='JONES' AND FIRSTNAME='WENDY')
OR (LASTNAME='SMITH' AND FIRSTNAME='JOHN');
```

SQL pagination will allow single matching index access for this type of OR condition – without list prefetch.

V9 requires multi-index access with list prefetch for matching index access.

Parallelism Enhancements

Removal Of Parallelism Restrictions #1

- **Support parallelism for multi-row fetch**
  - **In previous releases**
    - parallelism is disabled for the last parallel group in the top level query block
      - if there is no more table to join after the parallel group
      - and there is no GROUP BY clause or ORDER BY clause
  - Example:- SELECT * FROM CUSTOMER
    - There is no parallel group in the query and there are no table joins
    - There is no GROUP BY clause
    - There is no ORDER BY clause
    - So NO PARALLELISM will be used

- This restriction is only removed if the CURSOR is DECLARED as READ ONLY
  - Ambiguous Cursors will not have the restriction removed

Removal Of Parallelism Restrictions #2

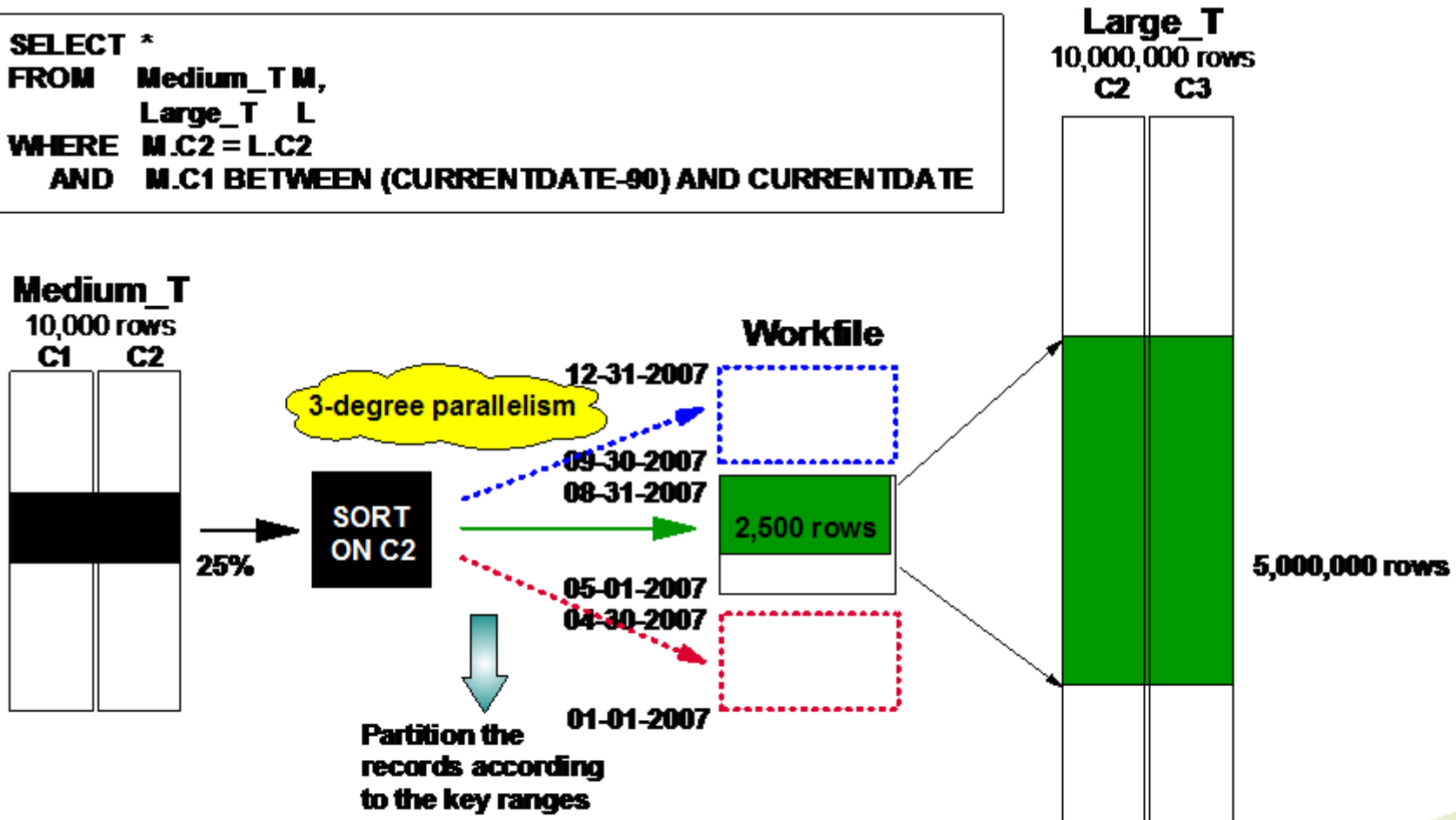
- Allow parallelism if a parallel group contains a work file
  - DB2 generates temporary a work file when view or table expression is materialized
  - This type of work file can not be shared among child task in previous releases of DB2, hence parallelism is disabled
  - DB2 10 will make the work file shareable
    - only applies to CP mode parallelism and no full outer join case

Parallelism Enhancements - Effectiveness

- Previous Releases of DB2 use Key Range Partitioning
  - Key Ranges Decided at Bind Time
  - Based on Statistics (low2key, high2key, column cardinality)
  - Assumes uniform data distribution
  - If the Statistics are out of date or data is not uniformly distributed what happens to performance

Key range partition - Today

```
SELECT *
FROM   Medium_T M,
       Large_T  L
WHERE  M.C2 = L.C2
  AND  M.C1 BETWEEN (CURRENTDATE-90) AND CURRENTDATE
```



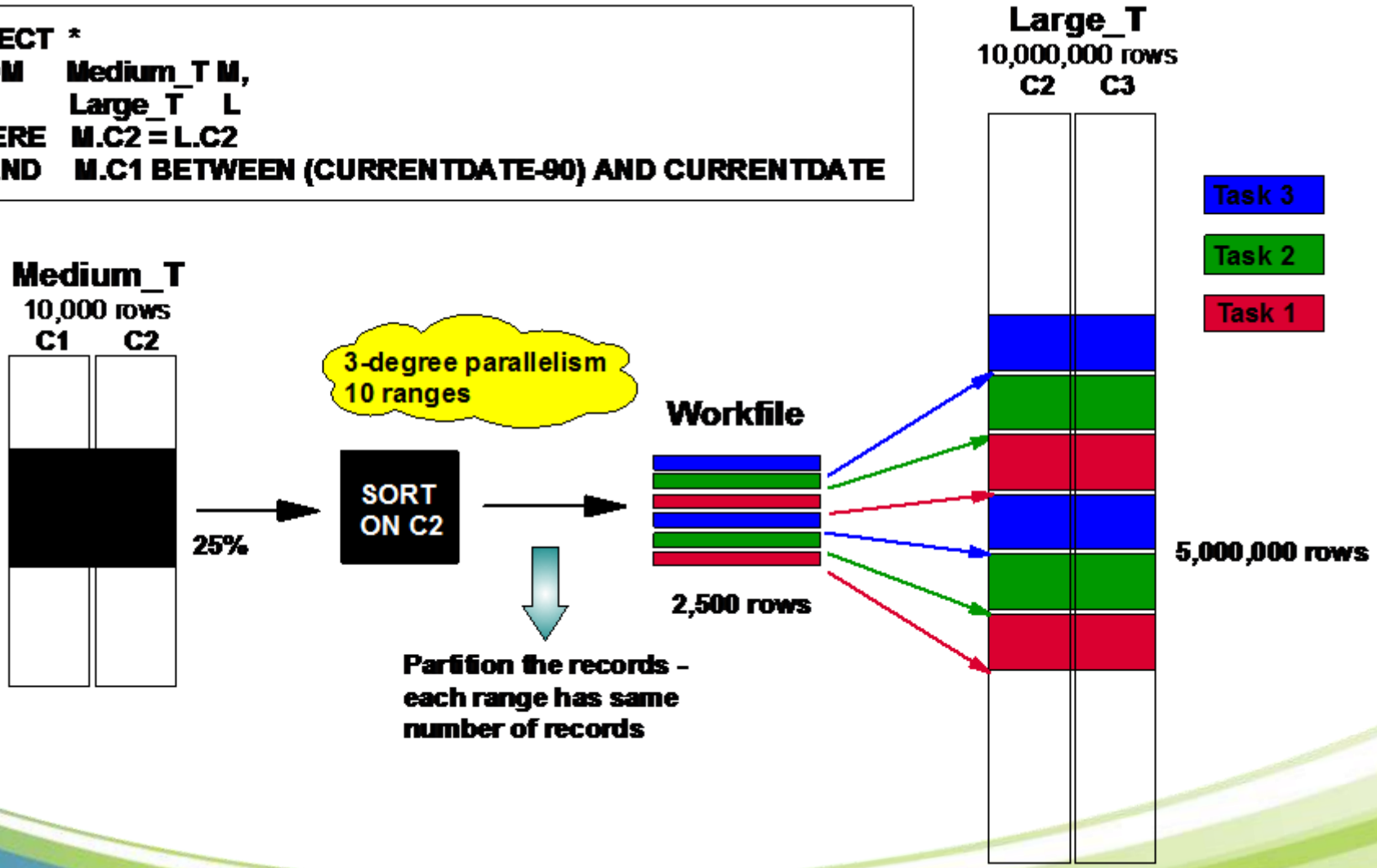
M.C1 is date column, assume currentdate is 8-31-2007, after the between predicate is applied, only rows with date between 06-03-2007 and 8-31-2007 survived, but optimizer chops up the key ranges within the whole year after the records are sorted :-(

Parallelism Enhancements - Effectiveness

- DB2 10 will use **Dynamic record range partitioning**
  - Materialize the intermediate result in a sequence of join processes
  - Results divided into ranges with equal number of records
  - Division doesn't have to be on the key boundary
    - Unless required for group by or distinct function
  - Record range partitioning is dynamic
    - no longer based on the key ranges decided at bind time
  - Now based on number of composite records and number of workload elements
    - Data skew, out of date statistics etc. will not have any effect on performance
    - DB2 will try to use in-memory work file for the materialization output if possible

Dynamic record range partition

```
SELECT *
FROM   Medium_T M,
       Large_T  L
WHERE  M.C2 = L.C2
  AND  M.C1 BETWEEN (CURRENTDATE-90) AND CURRENTDATE
```



Parallelism Enhancements - Effectiveness - Straw Model

- Previous releases of DB2 divide the number of keys or pages by the number representing the parallel degree
  - One task is allocated per degree of parallelism
  - The range is processed and the task ends
  - Tasks may take different times to process

- DB2 10 will use the Straw Model workload distribution method
  - More key or page ranges will be allocated than the number of parallel degrees
  - The same number of tasks as before are allocated (same as degree)
  - Once a task finishes it's smaller range it will process another range
  - Even if data is skewed this new process should make processing faster

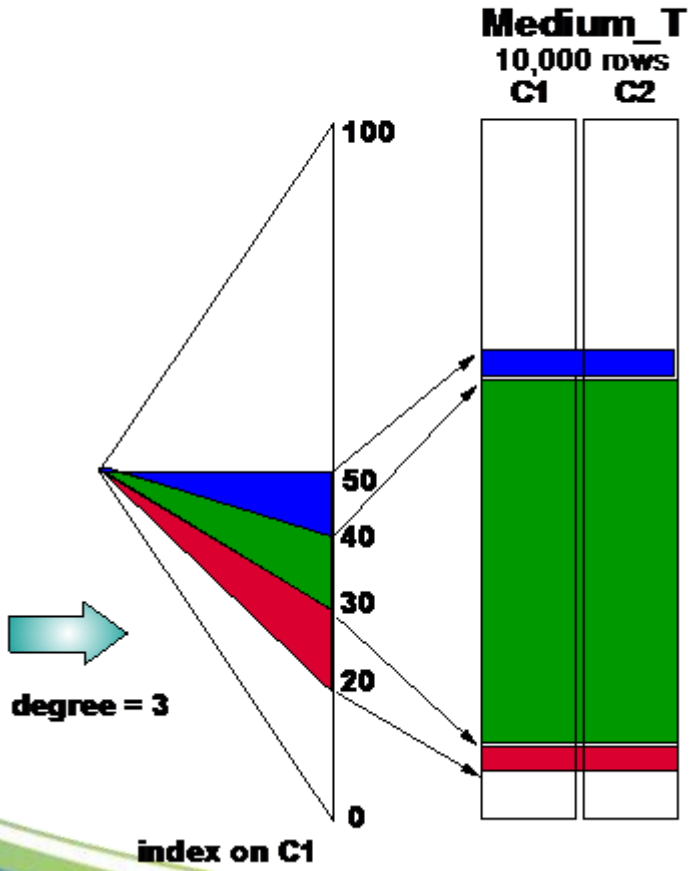
STRAW Model

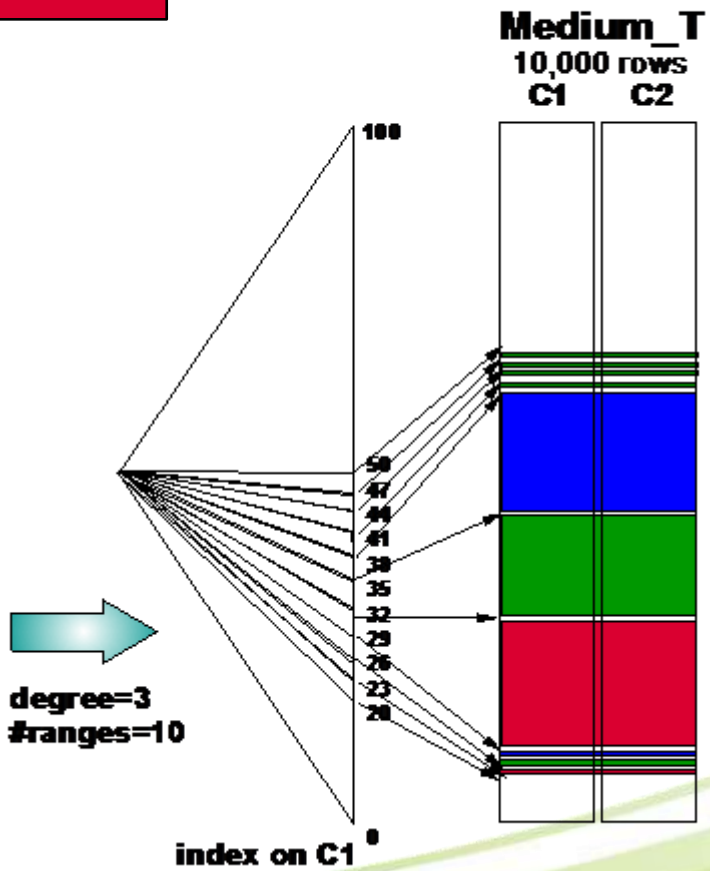
```
SELECT *
FROM   Medium_T M
WHERE  M.C1 BETWEEN 20 AND 50
```

Task 3

Task 2

Task 1





Divided in key ranges before DB2 10

Divided in key ranges with Straw Model

Runtime stats validation and Auto-Stats

Index Probing (Runtime stats validation)

- Estimate the number of rids within a given start/stop index key range by looking at a limited number of index non-leaf pages
- Index Probing will be done when both of the following conditions are met.
  - Query has matching index-access local predicate
  - Predicate contain literals, or REOPT(ALWAYS|ONCE|AUTO) is done
- In addition to the conditions above, at least one of the following must also be satisfied.
  - Predicate is estimated to qualify no rows
  - Stats indicate the table contains no rows
  - Table is defined with VOLATILE or passes the NPGTHRSH zparm threshold check
- New EXPLAIN table to externalize estimates obtained by index probing
  - DSN_COLDIST_TABLE
- Example testcases
  - VOLATILE tables with RUNSTATS failing to capture a representative sample
  - Ascending date values with stale statistics

AutoStats Problem Summary

- Collecting stats is a difficult and time consuming manual process
  - Need to look at the queries to figure out what stats are needed
  - Need to repeatedly look at the RTS tables to figure out when to recollect
- Inadequate stats collection leads to poor or inconsistent query performance
  - Sometimes the query runs well and sometimes it runs poorly
- Solution is to automate the process
  - More efficient
  - More accurate
  - More stable

Solution Overview

- Autonomic Statistics is implemented though a set of Stored Procedures
  - ADMIN_UTL_MONITOR
  - ADMIN_UTL_EXECUTE
  - ADMIN_UTL_MODIFY
  - STATS ADVISOR (Data Studio)
- SP's run automatically according to a predetermined schedule
- Working together, these SP's
  - Determine what stats to collect
  - Determine when stats need to be collected
  - Schedules and Performs the stats collection
  - Records activity for later review

Solution Overview

- Configuration / Communication via DB2 catalog tables
  - SYSAUTOTIMEWINDOWS
    - Defines when autonomic procedures can be run
  - SYSAUTORUNS_HIST
    - Keeps history of what procedures have executed autonomically
  - SYSAUTOALERTS
    - Populated when procedure detects that an action needs to be scheduled for execution (e.g. RUNSTATS needs to be scheduled)
  - SYSTABLES_PROFILES
    - Contains the RUNSTATS options for a particular table

Solution Overview

- RUNSTATS
  - New options to SET / UPDATE / USE a statistics profile
    - RUNSTATS … TABLE tbl COLUMN(C1)… **SET PROFILE**
    - RUNSTATS … TABLE tbl COLUMN(C5)… **UPDATE PROFILE**
    - RUNSTATS … TABLE tbl **USE PROFILE**
  - New option to do page-level sampling
    - RUNSTATS … TABLE tbl **TABLESAMPLE SYSTEM AUTO**

Runstats profile enhancement

SET PROFILE

```
RUNSTATS TABLESPACE
DB1.TS1 TABLE(S1.T1)
COLUMN(ALL) INDEX(ALL)
SET PROFILE
```

SYSIBM.SYSTABLES_PROFILES

TBNAME	PROFILE_TEXT
T2	COLUMN(C1, C2, C3)
T1	**COLUMN(ALL) INDEX(ALL)**
T3	COLGROUP(C1, C3, C2) HISTOGRAM
T4	COLGROUP(C1, C3) INDEX(ALL)
T5	COLUMN(C1) INDEX(ALL)

USE PROFILE

```
RUNSTATS TABLESPACE DB1.TS1
TABLE(S1.T1) USE PROFILE
```

SYSIBM.SYSTABLES_PROFILES

TBNAME	PROFILE_TEXT
T1	**COLUMN(ALL) INDEX(ALL)**
T3	COLGROUP(C1, C3, C2)
T2	COLUMN(C1, C2, C3)

```
RUNSTATS TABLESPACE
DB1.TS1 TABLE(S1.T1)
COLUMN(ALL) INDEX(ALL)
```

Summary

- Stats collection is an important process for query performance
- Automating the process should
  - Improve the quality of the statistics, leading to improved query performance and more stable query performance
  - Avoid collecting stats unnecessarily

More Information

- Join the World of DB2 for z/OS
- Follow us on Twitter
- Stay Connected DB2 for z/OS LinkedIn
- See you in Vienna or IOD Las Vegas

IDUG 2010 Europe
Hilton Vienna Hotel
8-12 November 2010.

Mandalay Bay
3950 S. Las Vegas Blvd.
Las Vegas, NV 89119

What's coming from the optimizer in DB2 10 for z/OS?

Terry Purcell
tpurcel@us.ibm.com