

IBM System z Technology Summit



What's new for SQL
optimization in DB2 10 for
z/OS



Disclaimer

© Copyright IBM Corporation 2011. 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 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.

Agenda

- Access path management
 - Dynamic Statement Cache Enhancements
 - Access Path Stability
 - Instance Based Statement Hints/Options
- Query performance improvements
- Runstats usability and performance improvements

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 PREPARE ATTRIBUTES clause
 - 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:**

```
WHERE ACCOUNT_NUMBER = 123456
```

- This would be replaced by

```
WHERE ACCOUNT_NUMBER = &
```

- **Performance Expectation**

- Using parameter marker still provides best performance
- Biggest performance gain for repeated SQL with different literals
- NOTE: Access path is not optimized for literals
 - True for parameter markers/host variables today
 - Need to use REOPT for that purpose

Agenda

- Access path management
 - Dynamic Statement Cache Enhancements
 - Access Path Stability
 - Instance Based Statement Hints/Options
- Query performance improvements
- Runstats usability and performance improvements

Getting Information on Prior 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
 - Complaint from DB2 9

Retrieving Access Path with EXPLAIN(NO)



▪ EXPLAIN PACKAGE

- Extract PLAN_TABLE information for packages
- The package/copy must be created on DB2 9 or later
- Useful if you didn't BIND with EXPLAIN(YES)
 - Or PLAN_TABLE entries are lost

```
>>-EXPLAIN----PACKAGE----->
```

```
>>-----COLLECTION--collection-name--PACKAGE--package-name----->
```

```
>-----+-----+-----+-----+----->
      |               |               |
      +---VERSION-version-name---+   +---COPY--copy-id---+
      |               |               |
```

- COPY-ID can be 'CURRENT', 'PREVIOUS', 'ORIGINAL'

Storing duplicate prior copies

- **APRETAINDUP option of REBIND**
 - Default YES
 - Retain duplicate for BASIC or EXTENDED
 - Optional NO
 - Do not retain duplicate access path as PREVIOUS or ORIGINAL
 - PREVIOUS/ORIGINAL must be from DB2 9 or later

- **If a duplicate is NOT kept (APRETAINDUP(NO))**
 - SWITCH is not possible to non-existent copy
 - EXPLAIN PACKAGE not possible for non-existent copy

- **Space saving option**
 - Majority of SPT01 moved to LOB in V10 (not compressible)

What if ? for BIND

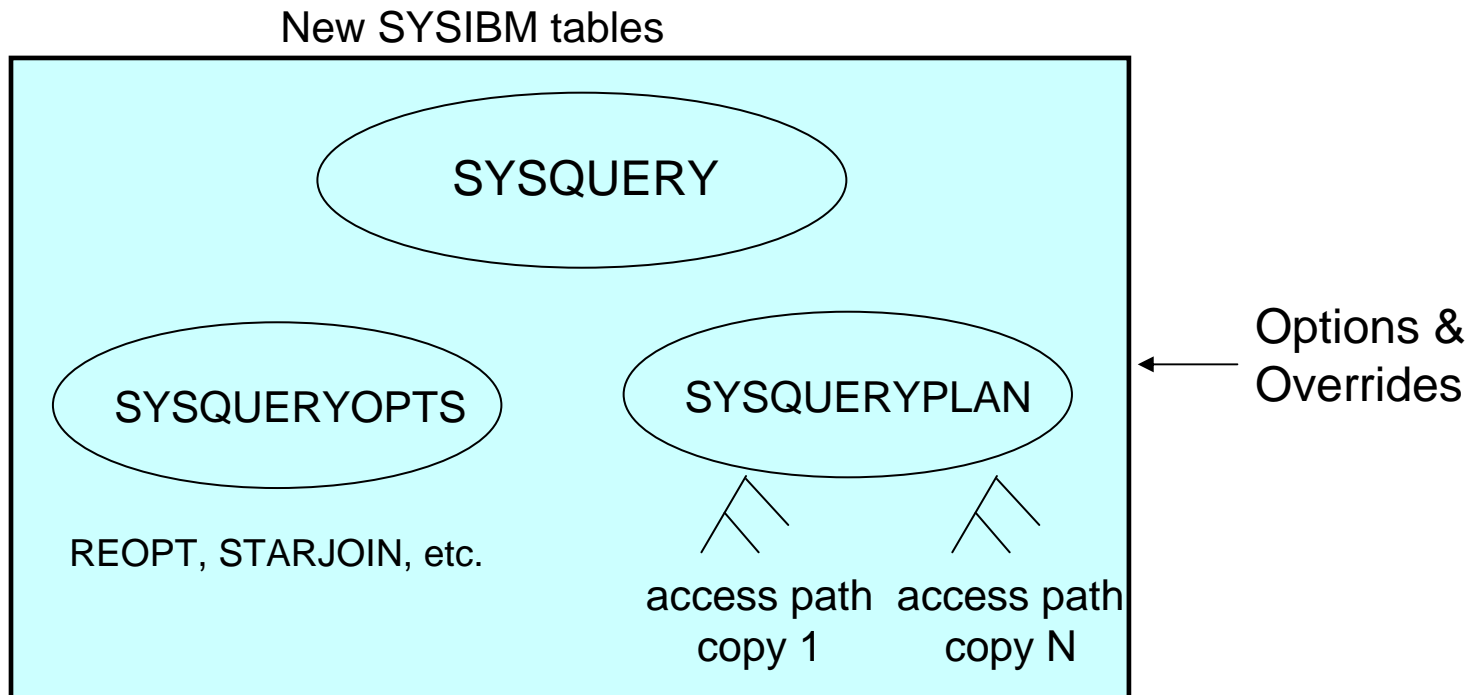
- **Bind package EXPLAIN(ONLY) & SQLERROR(CHECK)**

- Existing package copies are not overwritten
 - Performs explain or syntax/semantic error checks on SQL
- Requires BIND, BINDAGENT, or EXPLAIN privilege.
- Supported for BIND only
 - Not REBIND
 - Targeted to application changes
 - Eg. Development environment is DB2 LUW, production DB2 for z/OS

Agenda

- Access path management
 - Dynamic Statement Cache Enhancements
 - Access Path Stability
 - Instance Based Statement Hints/Options
- Query performance improvements
- Runstats usability and performance improvements

Access Path Repository – Hints/Statement level



Access Path Stability with 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:**
 - 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 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 mechanism

- Populate a user table `DSN_USERQUERY_TABLE` with query text
- 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

- **Statement-level granularity may be required rather than:**
 - Subsystem level ZPARMs
 - Package level BIND options
- **For example**
 - Only one statement in the package needs REOPT(ALWAYS) 
- **New mechanism for statement-level bind options:**
 - Similar to mechanism used for hints
 - DSN_USERQUERY_TABLE can also hold per-statement options

Agenda

- Access path management
- Query performance improvements
 - Predicate application
 - IN-list
 - Complex ORs
 - View/Table expr Merge
 - Safe Query Optimization
 - Parallelism Enhancements
 - Misc Query Perf Enhancements
- Runstats usability and performance improvements

Improvements to predicate application

- **Major enhancements to OR and IN predicates**



- Improved performance for AND/OR combinations and long IN-lists
 - General performance improvement to stage 1 predicate processing
- IN-list matching
 - Matching on multiple IN-lists
 - Transitive closure support for IN-list predicates
 - List prefetch support
 - Trim IN-lists from matching when preceding equals are highly filtering
- SQL pagination
 - Single index matching for complex OR conditions

- **Many stage 2 expressions to be executed at stage 1**

- Stage 2 expressions eligible for index screening



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 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: TBTYP – 'I'
 - New Access Type: ACTYP – 'IN'

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

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

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 (DB2 9)**

- Optimizer will be unlikely to consider T2 is the first table accessed

- **With IN-list PTC (DB2 10)**

- Optimizer can choose to access T2 or T1 first.

Reducing Matchcols for IN-lists

```
SELECT *  
FROM T1  
WHERE C1 = ?  
      AND C2 IN (?, ?, ?, ?, ?) ← Optimizer may  
                                choose not to match
```

- **If the equals (=) predicates provide strong filtering**
 - Optimizer may choose not to match on the IN-list
 - Instead apply as index screening
 - To avoid overhead of additional index probing
 - Example above
 - MATCHCOLS reduced from 2 to 1
 - ACCESSTYPE changed from “N” to “I”
 - Optimizer already trims IN-lists if equals predicates are unique

Agenda

- Access path management
- Query performance improvements
 - Predicate application
 - IN-list
 - Complex ORs (SQL Pagination)
 - View/Table expr Merge
 - Safe Query Optimization
 - Parallelism Enhancements
 - Misc Query Perf Enhancements
- Runstats usability and performance improvements

SQL Pagination targets 2 classes of OR queries:

- **Cursor scrolling (pagination) SQL**
 - Retrieve next n rows
 - Common in COBOL/CICS and any screen scrolling application
 - Not to be confused with “scrollable cursors”
 - Hence term pagination to avoid confusion (???)
- **Complex OR predicates against the same columns**
 - 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 to obtain the next 20 rows
 - 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 ;
```

- DB2 10 supports
 - Single matching index access with sort avoided
- DB2 9 requires
 - Multi-index access, list prefetch and sort
 - OR, extra predicate (AND LASTNAME >= 'JONES') for matching single index access and sort avoidance



Complex OR predicates against same index

- Given WHERE clause
 - And index on one or both columns

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

- DB2 9 requires
 - Multi-index access with list prefetch
- DB2 10 supports
 - Matching single index access – no list prefetch
 - Or, Multi-index access with list prefetch

SQL Pagination – PLAN_TABLE representation

- **Order of PLAN_TABLE entries is by coding sequence**
 - Determination of execution sequence deferred to runtime
 - When all host variables/parameter markers are resolved
 - For this example, coding seq does not match execution seq

WHERE (LASTNAME > ' JONES ')

OR (LASTNAME = ' JONES ' AND FIRSTNAME > ' WENDY ')

ORDER BY LASTNAME , FIRSTNAME ;

QBlockno	Planno	Accessname	Access_Type	Matchcols	Mixopseq
1	1	IX1	NR	1	1
1	1	IX1	NR	2	2

New access type (NR = IN-List Range)

Coding seq

Agenda

- Access path management
- Query performance improvements
 - Predicate application
 - IN-list
 - Complex ORs
 - View/Table expr Merge
 - Safe Query Optimization
 - Parallelism Enhancements
 - Misc Query Perf Enhancements
- Runstats usability and performance improvements

View/Table Expression Merge

- Merge enhancements for View/Table Expressions
 - **With outer joins.**

- Physical materialization is an overhead.
 - **Can limit the join sequence considered.**
 - **Can limit the ability to apply predicates early**
 - **Cannot create indexes on materialized work files**

- Generally merge is preferred over materialization

Merge expression on preserved side of Outer Join

- **DB2 can merge view/table expr on preserved side of outer join**
 - CASE, VALUE, COALESCE, NULLIF, IFNULL
 - Exception if merged predicate is stage 2

```

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
ON A.C2 = B.C2
WHERE T1.C2 = A.C2;

```

Merge single table view/table expr with subquery

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



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

- View/Table expr with subquery on NULL-supplied side
 - Merge into ON clause
- On preserved side
 - Merge into WHERE clause

Correlated table expr merge

```

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;	

Agenda

- Access path management
- Query performance improvements
 - Predicate application
 - Safe Query Optimization
 - Parallelism Enhancements
 - Misc Query Perf Enhancements
- Runstats usability and performance improvements

Minimizing Optimizer Challenges

- **Potential causes of sub-optimal plans**

- Insufficient statistics
- Unknown literal values used for host variables or parameter markers

- **Optimizer will evaluate the risk for 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

Minimizing impact of RID failure

- **RID overflow can occur for**
 - Concurrent queries each consuming shared RID pool
 - Single query requesting > 25% of table or hitting RID pool limit

- **DB2 9 will fallback to tablespace scan***

- **DB2 10 will continue by writing new RIDs to workfile**
 - Work-file usage may increase
 - Mitigate by increasing RID pool size (default increased in DB2 10).
 - MAXTEMPS_RID zparm for maximum WF usage for each RID list



* Hybrid join can incrementally process. Dynamic Index ANDing will use WF for failover.

Agenda

- Access path management
- Query performance improvements
 - Predicate application
 - Safe Query Optimization
 - Parallelism Enhancements
 - Misc Query Perf Enhancements
- Runstats usability and performance improvements

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

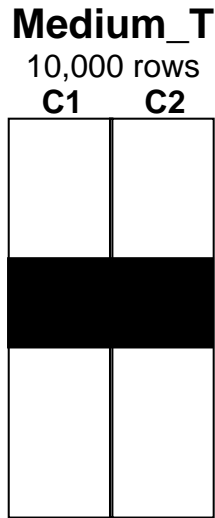
- **Previous Releases of DB2 use Key Range Partitioning**
 - Key Ranges Decided at Bind Time
 - Based on Statistics (low2key, high2key, column cardinality)
 - Assumes uniform data distribution
 - Histograms can help
 - But rarely collected
 - If Statistics are outdated or data is not uniformly distributed 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
    
```



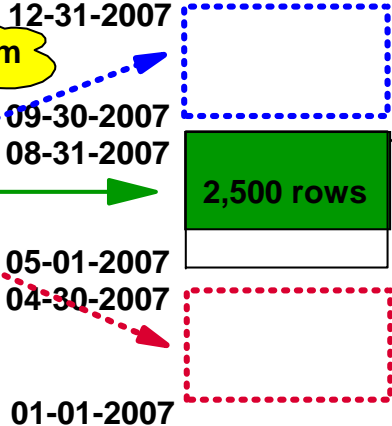
25%

3-degree parallelism

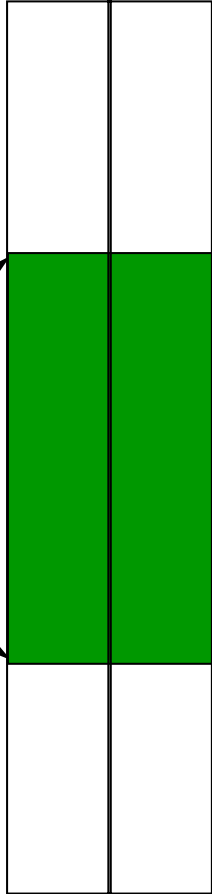
SORT ON C2



Partition the records according to the key ranges



Large_T
10,000,000 rows
C2 C3



5,000,000 rows

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 Effectiveness – Record range

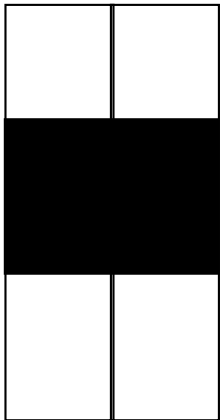
- **DB2 10 can use** Dynamic record range partitioning
 - Materialize the intermediate result in a sequence of join processes
 - Results divided into ranges with equal number of records
 - Division doesn't have to be on the key boundary
 - Unless required for group by or distinct function
 - Record range 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

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
    
```

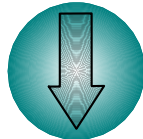
Medium_T
10,000 rows
C1 C2



25%

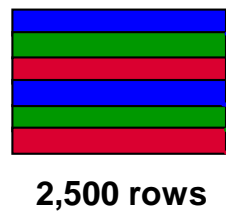
3-degree parallelism
10 ranges

**SORT
ON C2**

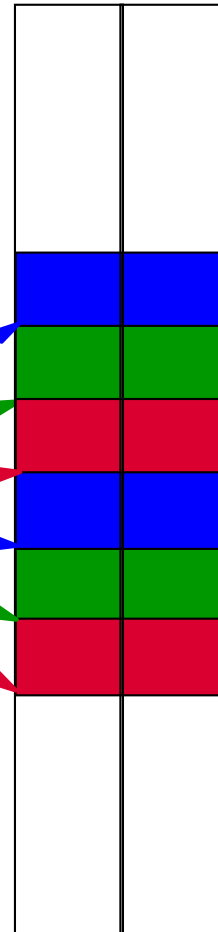


Partition the records -
each range has same
number of records

Workfile



Large_T
10,000,000 rows
C2 C3



Task 3
Task 2
Task 1

5,000,000 rows

Parallelism Effectiveness - Straw Model

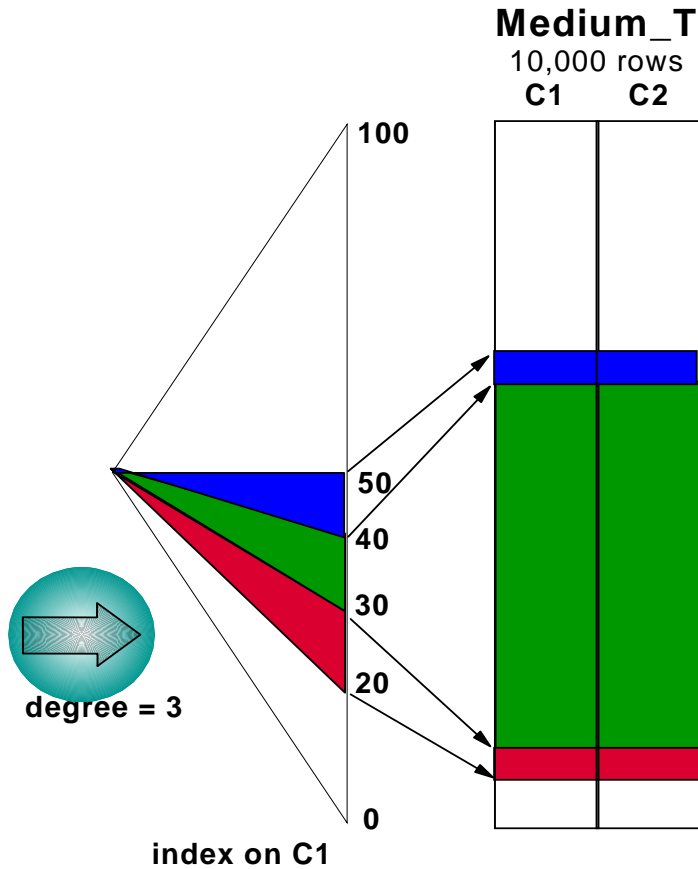
- **Previous releases of DB2 divide the number of keys or pages by the number representing the parallel degree**
 - One task is allocated per degree of parallelism
 - The range is processed and the task ends
 - Tasks may take different times to process

- **DB2 10 can 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

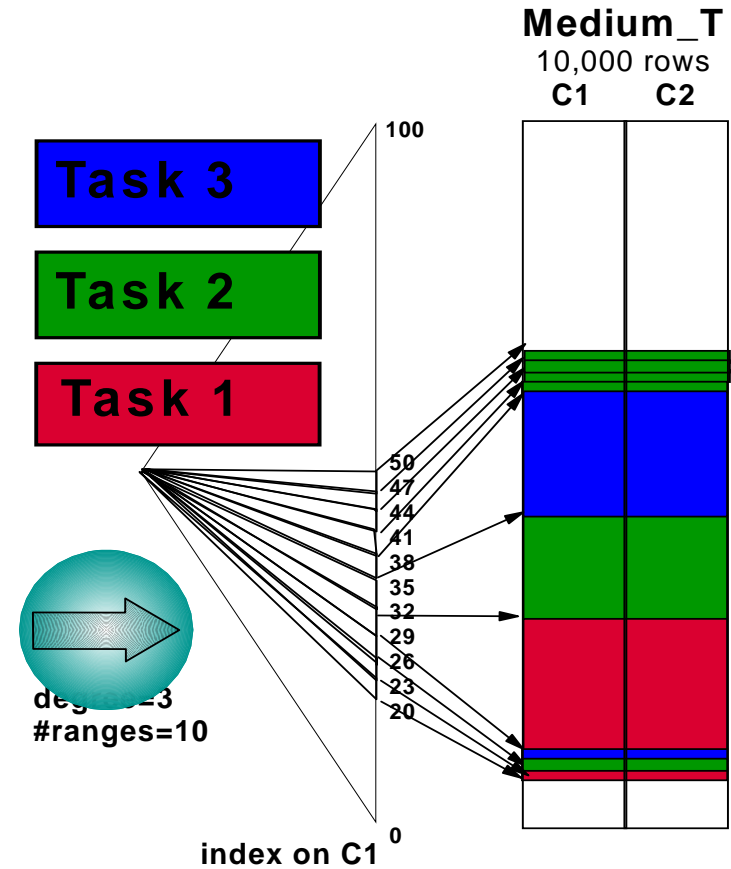
STRAW Model

```

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



Divided in key ranges before DB2 10




Divided in key ranges with Straw Model

Agenda

- Access path management
- Query performance improvements
 - Predicate application
 - Safe Query Optimization
 - Parallelism Enhancements
 - Misc query performance enhancements
- Runstats usability and performance improvements

Sort Performance Enhancements

- **FETCH FIRST n ROWS ONLY (FFnR) and Sort**
 - DB2 9 added in-memory replacement for FFnR to avoid sort
 - Provided $(n * (\text{sort key} + \text{data})) < 32\text{K}$
 - DB2 10 extends this to 128K

- **Avoid workfile usage for small sorts** 
 - DB2 9 avoided allocating WF for final sort only
 - If ≤ 255 rows and result $< 32\text{K}$ (sort key + data)
 - DB2 10 extends this to intermediate sorts also
 - Except for parallelism or SET function

- **Hash support for large sorts**
 - Potential for reduction in number of merge passes

Extending VOLATILE TABLE usage

- **VOLATILE TABLE support added in DB2 V8**

- Targeted to SAP Cluster Tables

- Avoids list prefetch

- Can be a problem for OR predicates or UPDATES at risk of loop

- **DB2 10 extends VOLATILE to general cases**



- Tables matching SAP cluster tables will maintain original limitations

- Table with 1 unique index

- Tables with > 1 index will follow NPGTHRSH rules

- Index access without limitation on list prefetch

Misc Performance enhancements

▪ Index **INCLUDE** columns



- Create an Index as UNIQUE, and add additional columns
- Ability to consolidate redundant indexes

```
INDEX1 UNIQUE (C1)
INDEX2 (C1,C2)
```



Consolidate to

```
INDEX1 UNIQUE (C1) INCLUDE (C2)
```

▪ Hash Data Access

- Directly locate a row in a table without having to use an index
 - DB2 10 implementation focus is OLTP
 - Parallelism not supported
- Great for Equality and IN predicates
 - Secondary indexes can be defined for Range Scans
 - Tablespace scan supported also

Agenda

- Access path management
- Query performance improvements
- Runstats usability and performance improvements
 - Optimizer Exploitation of Real-Time Stats
 - Auto-Stats
 - RUNSTATS Simplification/Performance

Optimizer Validation with Realtime Stats



- **Index Probing & RTS lookup**
 - Estimate # of rids within a given start/stop index key range at bind/prepare

- **Exploited when these two conditions are met.**
 - Query has matching index-access local predicate
 - Predicate contain literals, or REOPT(ALWAYS|ONCE|AUTO)

- **And 1 of the following is also true**
 - Predicate is estimated to qualify no rows
 - Stats indicate the table contains no rows
 - Table is defined as VOLATILE or qualifies for NPGTHRSH

- **New EXPLAIN table to externalize runtime estimates**
 - User managed DSN_COLDIST_TABLE

RUNSTATS 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**

- **Solution is to automate the process**
 - More efficient
 - More accurate
 - More stable

Autonomic Statistics Solution Overview

- **Autonomic Statistics is implemented through a set of Stored Procedures**
 - *Stored procedures are provided to enable administration tools and packaged applications to automate statistics collection.*
 - ADMIN_UTL_MONITOR
 - ADMIN_UTL_EXECUTE
 - ADMIN_UTL_MODIFY
 - Working together, these SP's
 - Determine what stats to collect
 - Determine when stats need to be collected
 - Schedule and Perform the stats collection
 - Records activity for later review
 - *See Chapter 11 "Designing DB2 statistics for performance" in the DB2 10 for z/OS Performance Monitoring and Tuning Guide for details on how to configure autonomic monitoring directly within DB2.*

RUNSTATS Simplification/Performance Overview

▪ RUNSTATS options to SET/UPDATE/USE a stats profile

– Integrate specialized statistics into generic RUNSTATS job

- RUNSTATS ... TABLE tbl COLUMN(C1)... **SET PROFILE**
 - Alternatively use **SET PROFILE FROM EXISTING STATS**
- RUNSTATS ... TABLE tbl COLUMN(C5)... **UPDATE PROFILE**
- RUNSTATS ... TABLE tbl **USE PROFILE**



▪ New option for page-level sampling

– But what percentage of sampling to use?

- RUNSTATS ... TABLE tbl **TABLESAMPLE SYSTEM AUTO**

