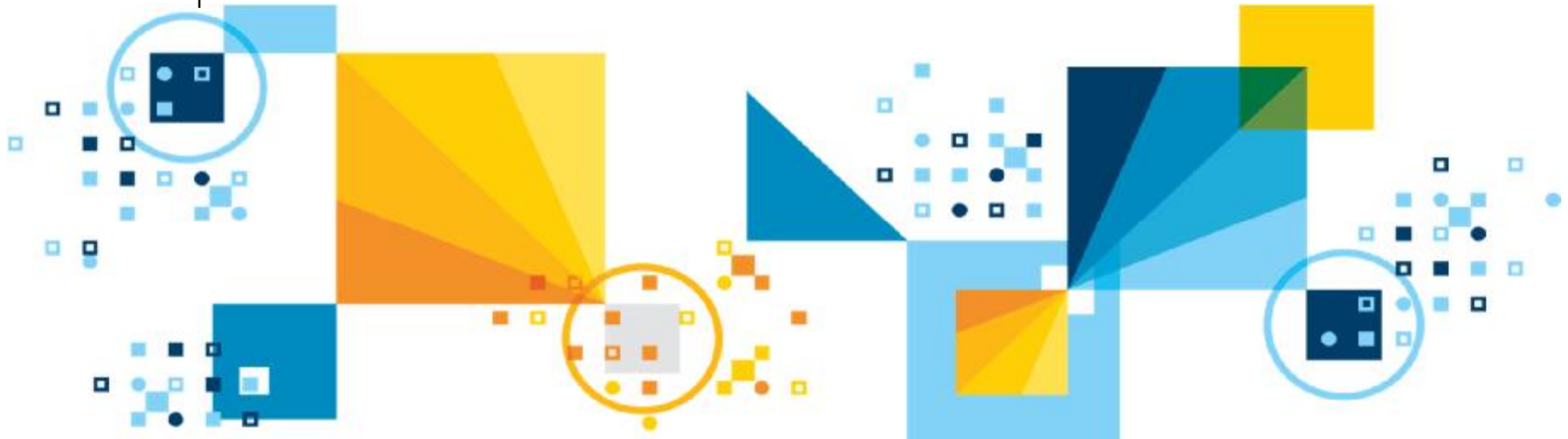


DB2 SQL Query Tuning with BLU Acceleration II

Module ID | 10304

Length | 1.5 hours + 2 hour hands on lab



For questions about this presentation contact askdata@ca.ibm.com

February 9, 2015

© 2015 IBM Corporation

PRESENTATION NOTES FOR PAGE 1

Disclaimer

© Copyright IBM Corporation 2015. All rights reserved.

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 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.

PRESENTATION NOTES FOR PAGE 2

Module Information

§ You should have completed or acquired the necessary knowledge for the following modules in order to complete this module:

- Module DB2 SQL Query Tuning with BLU Acceleration
- Module DB2 BLU Acceleration Fundamentals
- Module Monitoring Essentials in DB2 BLU Acceleration
- Module Deep Dive into DB2 BLU Acceleration Main Ideas

§ After completing this module, you should be able to know

- How to capture, analyze, and improve SQL through both better coding techniques and defining complementary objects like indexes and MQT's for your database Optim Query tuner
- Implement DB2 BLU Acceleration support for a new or existing DB2 database
- Configure a DB2 database that uses DB2 BLU Acceleration, column-organized tables, including sort memory and utility heap memory considerations
- Describe the default workload management used for DB2 BLU Acceleration processing and how you can tailor the WLM objects to efficiently use system resources
- Monitor a DB2 database or application that uses column-organized tables using SQL monitor functions
- Locate the column-organized processing portion of the access plans for column-organized tables in DB2 explain reports

PRESENTATION NOTES FOR PAGE 3

Agenda

§ Problematic SQL & Situation

§ Response time solution

§ SQL costs solution

- Monitoring
- Analyzing SQL
- Statement Concentrator

§ BLU Implementation and Use

§ Making Performance Improvements

- Database Objects
- Better Coding
- Design Advisor
- DB2 Optim Query Workload Tuner
- Other Considerations

PRESENTATION NOTES FOR PAGE 4

Consideration for implementing column-organized tables



PRESENTATION NOTES FOR PAGE 5

System requirements for column-organized tables

§ Linux (x86-x64, Intel and AMD processors) and AIX (POWER processors)

§ Database features not currently supported:

- pureScale cluster
- Database partitioning (DPF)
- Databases whose code set and collation are not UNICODE and
IDENTITY or IDENTITY_16BIT

PRESENTATION NOTES FOR PAGE 6

Notes:

System and database configurations

Column-organized tables are supported only on Linux (x86-x64, Intel and AMD processors) and AIX® (POWER® processors).

The following system and database configurations do not support column-organized tables in the DB2 Version 10.5 release.

Databases in a pureScale® environment

Partitioned databases

Databases that are created without automatic storage enabled

Table spaces that are not enabled for reclaimable storage

Databases that use the high availability disaster recovery (HADR) feature

Databases whose code set and collation are not UNICODE and IDENTITY or IDENTITY_16BIT

Databases that use any of the compatibility features that are controlled by the DB2_COMPATIBILITY_VECTOR registry variable. If a database has any of these compatibility features enabled, or the DB2_COMPATIBILITY_VECTOR registry variable is set to any nonzero value, a CREATE TABLE statement with the ORGANIZE BY COLUMN clause returns an error.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

General system configuration recommendations for column-organized table usage

	Small	Medium	Large
Raw data (CSV)	~1TB	~5TB	~10TB
Minimum:			
#cores	8	16	32
Memory	64GB	256GB	512GB
High-end performance:			
#cores	16	32	64
Memory	128 – 256GB	384 – 512GB	1 – 2TB

Assumption: all data is active and equally “hot”.

PRESENTATION NOTES FOR PAGE 7

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Sort memory configuration for column-organized tables

- § Ensure that the *sortheap* (sort heap) and *sheapthres_shr* (sort heap threshold for shared sorts) database configuration parameters ARE NOT set to AUTOMATIC
 - STMM management of sort memory is not currently supported for column-organized table processing
- § Consider increasing these values significantly for analytics workloads.
- § Suggested sort memory configuration
 - Set *sheapthres_shr* to the size of the buffer pool (across all buffer pools)
 - Set *sortheap* to some fraction (for example, 1/20) of *sheapthres_shr* to enable concurrent sort operations.

PRESENTATION NOTES FOR PAGE 8

Notes:

Ensure that the `sorheap` (sort heap) and `sheapthres_shr` (sort heap threshold for shared sorts) database configuration parameters are not set to `AUTOMATIC`. Consider increasing these values significantly for analytics workloads. A reasonable starting point is setting `sheapthres_shr` to the size of the buffer pool (across all buffer pools). Set `sorheap` to some fraction (for example, $1/20$) of `sheapthres_shr` to enable concurrent sort operations.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Current restrictions for column-organized tables in DB2 10.5

- § Schemas that include column-organized tables cannot be transported
- § Data federation with column-organized tables is not supported
- § For data replication with column-organized tables as either source or target:
 - A CREATE TABLE statement with the DATA CAPTURE CHANGES clause is not supported.
 - The use of column-organized tables as targets for data replication is not recommended.
- § Queries using the RR or RS isolation level are not supported with column-organized tables
- § Columns of type BLOB, DBCLOB, CLOB, or XML cannot be defined for column-organized tables

PRESENTATION NOTES FOR PAGE 9

Notes:

The following additional restrictions apply to column-organized tables in the DB2 Version 10.5 release:

Schemas that include column-organized tables cannot be transported.

Data federation with column-organized tables is not supported.

For data replication with column-organized tables as either source or target:

A CREATE TABLE statement with the DATA CAPTURE CHANGES clause is not supported.

The use of column-organized tables as targets for data replication is not recommended.

Automatic tuning of sort memory is not supported in environments using column-organized tables.

For label-based access control (LBAC), the following syntax is not supported: ALTER TABLE...SECURITY POLICY policynname

For row and column access control (RCAC), the following syntax is not supported:

ALTER TABLE...ACTIVATE ROW | COLUMN ACCESS CONTROL

CREATE PERMISSION / CREATE MASK

The following SQL statements do not support column-organized tables as target or source tables:

CREATE EVENT MONITOR

CREATE GLOBAL TEMPORARY TABLE

CREATE INDEX and ALTER INDEX

CREATE MASK

CREATE NICKNAME

CREATE PERMISSION

CREATE TRIGGER

DECLARE GLOBAL TEMPORARY TABLE

SET INTEGRITY

Queries using the RR or RS isolation level are not supported with column-organized tables.

XA transactions are not supported.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IBM InfoSphere Optim Query Workload Tuner for DB2 for LUW estimates benefits for column-organized tables

Advisor identifies candidate tables for conversion to columnar format.

Analyzes SQL workload and estimates execution cost on row- and column-organized tables.

Review Workload Advisor Recommendations

Database connection: TPCDSDANV10.2hotel67 (DB2 for Linux, UNIX, and Windows)

Estimated performance improvement: 83.44 %
Number of tables referenced in the workload: 11

Statements Summary Table organization Candidate Table Organization

Table	Creator	Current Organization	Recommended Organization	Conversion Warning	Cardinality	References to Table	Cumulative Total Cost
HOUSEHOLD_DEMOGRAPHICS	TPCDS	ROW	COLUMN	Indexes will be removed	2	2	846,761,862.00
DATE_DIM	TPCDS	ROW	COLUMN	Indexes will be removed	51	15	15,408,311,462.00
WEB_SALES	TPCDS	ROW	COLUMN	Indexes will be removed	15	1	13,587,838,304.00
STORE	TPCDS	ROW	COLUMN	Indexes will be removed	1	19	973,711,296.00
STORE_SALES	TPCDS	ROW	COLUMN	Indexes will be removed	19	3	14,261,145,494.00
CUSTOMER_ADDRESS	TPCDS	ROW	COLUMN	Indexes will be removed	3	1	1,022,691,606.00
STORE_RETURNS	TPCDS	ROW	COLUMN	Indexes will be removed	1	1	973,711,296.00
ITEM	TPCDS	ROW	COLUMN	Indexes will be removed	25	25	11,926,096,848.00
CUSTOMER	TPCDS	ROW	COLUMN	Indexes will be removed	4	4	2,635,452,752.00

SQL Statements Affected

Execution Count	Weight	Estimated Performance Gain(%)	Cost Before	Cost After	Statement Text
1	0.00	-202.20	22,203,011.00	67,097,300.00	SELECT Q5."SS_ADDR_SK" AS Q5C6, Q5."SS_CDEMO_SK" AS Q5C4, Q5."SS_CUSTOMER_SK" AS Q5C3, Q5."SS...
1	0.00	78.99	438,398,000.00	92,125,000.00	select iss.i_brand_id as brand_id, iss.i_class_id class_id, iss.i_category_id category_id from tpchs.store_sal...
1	0.00	81.84	824,558,800.00	149,764,000.00	select avg(ss_quantity), avg(ss_ext_sales_price), avg(ss_ext_wholesale_cost), sum(ss_ext_wholesale_cost) from ...
1	0.00	77.40	2,085,428,000.00	471,247,000.00	with cross_items as (select i_item_sk ss_item_sk from tpchs.item, (select iss.i_brand_id brand_id, iss.i_clas...
1	0.00	48.12	175,929,700.00	91,275,600.00	select cd_gender, cd_marital_status, cd_education_status, count(*) as gender_marital_educ_count, cd_purchase_estimate, count(*) as...

PRESENTATION NOTES FOR PAGE 10

Notes:

IBM® InfoSphere® Optim™ Query Workload Tuner Version 4.1 includes the Workload Table Organization Advisor, which examines all of the tables that are referenced by the statements in a query workload. Its recommendations lead to the best estimated performance improvement for the query workload as a whole. The advisor presents its analysis and rationales so that you can see the tables that are recommended for conversion from row to column organization.

Instructor notes:

Purpose —

Details —


Additional information —

Transition statement —

Sample recommendations for workload using Optim Query Workload Tuner

Review Workload Advisor Recommendations

This page shows the recommendations from the advisors that you ran.

Database connection:  TPCDSDANv10.2hotel67 (DB2 for Linux, UNIX, and Windows V10.5.0)

► Status/Description

Statements | Summary | **Table organization** | Candidate Table Organization

Estimated performance improvement: 83.44 %

Number of tables referenced in the workload: 11 Number of tables recommended for conversion: 11



Show DDL Script Test Candidate Table Organization   Filter by Tables to be converted ▼

Table	Creator	Current Organization	Recommended Organization	Conversion Warning
HOUSEHOLD_DEMOG...	TPCDS	ROW	COLUMN	Indexes will be remove
DATE_DIM	TPCDS	ROW	COLUMN	Indexes will be remove
WEB_SALES	TPCDS	ROW	COLUMN	Indexes will be remove
STORE	TPCDS	ROW	COLUMN	Indexes will be remove
STORE SALES	TPCDS	ROW	COLUMN	Indexes will be remove

PRESENTATION NOTES FOR PAGE 11

Notes:

Instructor notes:

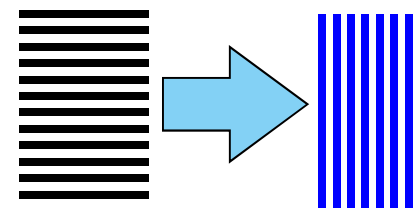
Purpose —

Details —

Additional information —

Transition statement —

db2convert – command line tool to ease converting row-organized tables to column-organized tables



- Converts one or all row-organized user tables in a specified database into column-organized tables.
- The row-organized tables remain online during command processing
- Calls ADMIN_MOVE_TABLE procedure
- For monitoring purposes, the command displays statistics about the conversion

```
db2convert
    -d <database-name>      (this is the only mandatory parameter)
    -stopBeforeSwap
    -continue                (resumes a previously stopped conversion)
    -z <schema-name>
    -t <table-name>
    -ts <tablespace for new table>
    -opt <ADMIN_MOVE_TABLE options> (e.g. COPY_USE_LOAD)
    ...
```

PRESENTATION NOTES FOR PAGE 12

The following table types cannot be converted with db2convert:

Range partitioned tables

Multidimensional clustering tables

Range clustered tables

Insert time clustering tables

Tables that have generated columns

Tables that have XML or LOB columns

Typed tables

Materialized query tables

Declared global temporary tables

Created global temporary tables

Additional notes for db2convert usage

- § These attributes are not used during conversion to column-organized tables:
 - Triggers
 - Secondary indexes
- § If not required, drop any dependent objects that cannot be transferred to column-organized tables before invoking the db2convert command.
- § These attributes are used as NOT ENFORCED during conversion to column-organized tables:
 - Foreign keys
 - Check constraints
- § Conversion process requires space for both the source and target tables.
- § No online process to convert column- back to row-organized tables, best practice is a backup before converting the tables to column organization.
- § If the database is recoverable and the default COPY_USE_LOAD option is used, performing the conversion in three separate steps is strongly recommended:
 1. Invoke the db2convert command, specifying the -stopBeforeSwap option.
 2. Perform a manual online backup of the target table space or table spaces.
 3. Invoke the db2convert command, specifying the -continue option.

PRESENTATION NOTES FOR PAGE 13

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Sample db2convert output shows progress and compression results

```

$ Table           RowsNum      RowsComm      Status      Progress (%)
$ -----
$ "TESTROW"."ACCT" 1000000      908100        COPY        90.81

$ Table           RowsNum      RowsComm      Status      Progress (%)
$ -----
$ "TESTROW"."ACCT" 1000000      1000000       COPY        100.00

$ Table           RowsNum      RowsComm      Status      Progress (%)
$ -----
$ "TESTROW"."ACCT" 0             0             REPLAY      100.00

$ Table           RowsNum      RowsComm      Status      Progress (%)
$ -----
$ "TESTROW"."ACCT" 0             0             SWAP        0.00

$ Table           RowsNum      RowsComm      Status      Progress (%)
$ -----
$ "TESTROW"."ACCT" 0             0             SWAP        100.00

$ Table           RowsNum      InitSize (MB)  FinalSize (MB)  CompRate (%)  State -----
$ "TESTROW"."ACCT" 1000000      126.25         26.12           79.31         Completed

$ Pre-Conversion Size (MB): 126.25
$ Post-Conversion Size (MB): 26.12
$ Compression Rate (Percent): 79.31

```

PRESENTATION NOTES FOR PAGE 14

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using ADMIN_MOVE_TABLE to convert row-organized tables to a column-organized table

§ The ADMIN_MOVE_TABLE procedure can be used to convert row-organized tables to a column-organized table

§ To indicate conversion to a column-organized table you can specify ORGANIZE BY COLUMN as an option of ADMIN_MOVE_TABLE
For example:

```
call  
admin_move_table('TEST', 'ACCT2', 'AS2', 'AS2', 'AS2',  
'ORGANIZE BY COLUMN', '', '', '', 'COPY_USE_LOAD', 'MOVE')
```

§ Specify COPY_USE_LOAD option to move data using a LOAD utility to generate the Column Dictionaries

§ Target table could be pre-defined as a column-organized table

§ Any Unique or Primary Key indexes on the source table will be added to the column-organized target table as enforced Unique or Primary Key constraints

PRESENTATION NOTES FOR PAGE 15

Notes:

```
call admin_move_table('TEST','ACCT2','AS2','AS2','AS2','ORGANIZE BY COLUMN',",",",",',','COPY_USE_LOAD','MOVE')
```

Result set 1

KEY	VALUE
AUTHID	INST20
CLEANUP_END	2013-06-26-12.13.36.463983
CLEANUP_START	2013-06-26-12.13.36.275748
COPY_END	2013-06-26-12.13.35.640270
COPY_OPTS	OVER_INDEX,LOAD,WITH_INDEXES,NON_CLUSTER
COPY_START	2013-06-26-12.13.21.951896
COPY_TOTAL_ROWS	1000000
INDEXNAME	ACCT2ACCT
INDEXSCHEMA	TEST
INDEX_CREATION_TOTAL_TIME	0
INIT_END	2013-06-26-12.13.21.537935
INIT_START	2013-06-26-12.13.19.632628
ORIGINAL_TBLSIZE	129280
REPLAY_END	2013-06-26-12.13.36.163455
REPLAY_START	2013-06-26-12.13.35.640947
REPLAY_TOTAL_ROWS	0
REPLAY_TOTAL_TIME	0
STATUS	COMPLETE
SWAP_END	2013-06-26-12.13.36.229232
SWAP_RETRIES	0
SWAP_START	2013-06-26-12.13.36.164119
UTILITY_INVOCATION_ID	010000000600000008000000000000000002013062612132153943900000000
VERSION	10.05.0000

23 record(s) selected.

Return Status = 0

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

DB2 Utility support for column-organized tables

§ REORG

- Only supports RECLAIM EXTENTS option, no standard offline or online table reorganization
- Automated by default, but the REORG can be run manually to free full extents emptied by deletes or updates

§ REORGCHK report is not useful for analysis of column-organized tables

§ RUNSTATS

- LOAD utility collects statistics by default, but standard RUNSTATS can be run manually
- Automatic statistics collection is not supported for column-organized tables

§ db2advis – does not make recommendations for column-organized tables

- Use Optim Query Workload Tuner

PRESENTATION NOTES FOR PAGE 16

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Referential Integrity and Unique constraints for column-organized tables

§ The CREATE INDEX statement can not be used to define traditional indexes on a column-organized table

§ Constraints

- ENFORCED check and foreign key (referential integrity) constraints are not supported on column-organized tables.
 - These constraints are supported as informational (NOT ENFORCED) constraints.
- You cannot specify the WITH CHECK OPTION when creating a view that is based on column-organized tables

§ Primary Key and Unique Constraints

- You can define ENFORCED or NOT ENFORCED Primary key and unique constraints for column-organized tables
 - DB2 will create system maintained indexes for enforced constraints
 - The performance of a select, update, or delete operation that affects only one row in a column-organized table can be improved if the table has unique indexes

PRESENTATION NOTES FOR PAGE 17

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using explain tools to evaluate access plans for column-organized tables



PRESENTATION NOTES FOR PAGE 18

Explain report for summary query using a column-organized table

```
SELECT  ACCT_GRP,  SUM(BALANCE)
FROM    test.ACCT
WHERE
  ACCT_GRP BETWEEN 100 AND 150
GROUP BY
  ACCT_GRP
```

Group BY Performed in
Column-organized
Portion of access plan



```

$      Rows
$      RETURN
$      (  1 )
$      Cost
$      I/O
$      |
$      57.5334
$      CTQ
$      (  2 )
$      378.646
$      41.7544
$      |
$      57.5334
$      GRPBY
$      (  3 )
$      378.641
$      41.7544
$      |
$      57997.4
$      TBSCAN
$      (  4 )
$      377.516
$      41.7544
$      |
$      1e+06
$      CO-TABLE: TEST
$      ACCT
$      Q1
$
```

PRESENTATION NOTES FOR PAGE 19

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Execution Plans for column-organized tables

- § Runtime operators are optimized for row- and column-organized tables
- § CTQ operator transfers data from column- to row-organized processing
- § Operators that are optimized for column-org tables below CTQ include:
 - Table scan
 - Hash-based join, optionally employing a semi-join
 - Hash-based group by. Potentially faster without the sort
 - Hash-based unique.
- § Aim is to “push down” most operators below CTQ
- § Some operations cannot be pushed down, such as:
 - SORT
 - SQL OLAP function (e.g., rank())
 - Inequality join (join predicate that is NOT equality)

PRESENTATION NOTES FOR PAGE 20

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Explain report detail for CTQ operator

\$	2) TQ : (Table Queue)	\$	Input Streams:
\$	Cumulative Total Cost:	\$	-----
\$	378.646	\$	3) From
\$	Cumulative CPU Cost:	\$	Operator #3
\$	8.04973e+08	\$	
\$	Cumulative I/O Cost:	\$	Estimated number of rows:
\$	41.7544	\$	57.5334
\$	Cumulative Re-Total Cost:	\$	
\$	129.893	\$	Number of columns: 2
\$	Cumulative Re-CPU Cost:	\$	
\$	8.04867e+08	\$	Subquery predicate ID: Not
\$	Cumulative Re-I/O Cost: 0	\$	Applicable
\$	Cumulative First Row Cost:	\$	
\$	8.17176	\$	Column Names:
\$	Estimated Bufferpool Buffers: 0	\$	----- +Q4.\$C1+Q4.ACCT_GRP ---
\$	Arguments:	\$	
\$	-----	\$	
\$	LISTENER: (Listener Table Queue type)	\$	Output Streams:
\$	FALSE	\$	-----
\$	TQDEGREE: (Degree of Intra-Partition	\$	4) To Operator
\$	parallelism)	\$	#1
\$	1	\$	
\$	TQMERGE : (Merging Table Queue flag)	\$	Estimated number of rows:
\$	FALSE	\$	57.5334
\$	TQORIGIN: (Table Queue Origin type)	\$	
\$	COLUMN-ORGANIZED DATA	\$	Number of columns: 2
\$	TQREAD : (Table Queue Read type)	\$	
\$	READ AHEAD	\$	Subquery predicate ID: Not
\$	UNIQUE : (Uniqueness required flag)	\$	Applicable
\$	FALSE	\$	

PRESENTATION NOTES FOR PAGE 21

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Explain report object data for column-organized table

```
§           Schema: TEST
§           Name:      ACCT
§           Type:      Column-organized Table
§                   Time of creation:      2013-06-17-11.53.33.299901
§                   Last statistics update: 2013-06-17-11.55.05.741512
§                   Number of columns:      6
§                   Number of rows:         1000000
§                   Width of rows:          40
§                   Number of buffer pool pages: 140
§                   Number of data partitions: 1
§                   Distinct row values:      No
§                   Tablespace name:         AS2
§                   Tablespace overhead:     6.725000
§                   Tablespace transfer rate: 0.320000
§                   Source for statistics:    Single Node
§                   Prefetch page count:     4
§                   Container extent page count: 4
§                   Table overflow record count: 0
§                   Table Active Blocks:     -1
§                   Average Row Compression Ratio: -1
§                   Percentage Rows Compressed: -1
§                   Average Compressed Row Size: -1
```

PRESENTATION NOTES FOR PAGE 22

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Explain report shows estimated costs vary depending on the number of columns accessed

```
$ Access Plan:
$ -----
$           Total Cost:
$           197.054
$           Query Degree:                1

$           Rows
$           RETURN
$           (    1)
$           Cost
$           I/O
$           |
$           .....
$           2336.77
$           CTQ
$           (    4)
$           196.229
$           38.5352
$           |
$           2336.77
$           TBSCAN
$           (    5)
$           196.216
$           38.5352
$           |
$           513576
$ CO-TABLE: TEST
$           HISTORY
$           Q1
```

```
SELECT
  HISTORY.BRANCH_ID,
  HISTORY.ACCTNAME
FROM
  HISTORY AS HISTORY
WHERE
  HISTORY.BRANCH_ID = 25
ORDER BY
  HISTORY.ACCTNAME ASC
```

← Increased cost
And I/O estimates →

```
$ Access Plan:
$ -----
$           Total Cost:
$           204.802
$           Query Degree:                1

$           Rows
$           RETURN
$           (    1)
$           Cost
$           I/O
$           |
$           .....
$           2336.77
$           CTQ
$           (    4)
$           203.945
$           62.6197
$           |
$           2336.77
$           TBSCAN
$           (    5)
$           203.927
$           62.6197
$           |
$           513576
$ CO-TABLE: TEST
$           HISTORY
$           Q1
```

```
SELECT
  HISTORY.BRANCH_ID,
  HISTORY.ACCTNAME,
  HISTORY.ACCT_ID,
  HISTORY.BALANCE
FROM
  HISTORY AS HISTORY
WHERE
  HISTORY.BRANCH_ID = 25
ORDER BY
  HISTORY.ACCT_ID ASC
```

PRESENTATION NOTES FOR PAGE 23

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

What to look for in access plans for column-organized tables

A good access plan would have: A suboptimal access plan could have:

- § One or few CTQ operators
- § Few operators above CTQ
- § Operators above CTQ work on few rows
- § Few rows flow through the CTQ

- § Many CTQs
- § Many operators above CTQ
- § Operators above CTQ work on many rows
- § Many rows flow through the CTQ

Disclaimer: While these are the most common indicators of good vs sub plans with column-organized tables, there can be exceptions.

PRESENTATION NOTES FOR PAGE 24

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Explain report for joining two column-organized tables

```
SELECT  HISTORY.BRANCH_ID, TELLER.TELLER_NAME,
        HISTORY.ACCTNAME, HISTORY.ACCT_ID, HISTORY.BALANCE
FROM HISTORY AS HISTORY,   TELLER AS TELLER
WHERE HISTORY.TELLER_ID = TELLER.TELLER_ID AND
      HISTORY.BRANCH_ID = 25
ORDER BY  HISTORY.BRANCH_ID ASC,  HISTORY.ACCT_ID ASC
```

Row oriented processing

```
$
$      Total Cost:
$      226.575
$
$      Query Degree:                1
$
$      Rows
$      RETURN
$      (    1)
$      Cost
$      I/O
$      |
$      2374.76
$      TBSCAN
$      (    2)
$      226.575
$      77.1883
$      |
$      2374.76
$      SORT
$      (    3)
$      226.479
$      77.1883
```

Sort Performed in Row-organized Portion of access plan

Column oriented processing

```
$      2374.76
$      CTQ
$      (    4)
$      225.686
$      77.1883
$      |
$      2374.76
$      HSJOIN
$      (    5)
$      225.661
$      77.1883
$      /-----+-----\
$      2336.77              1000
$      TBSCAN              TBSCAN
$      (    6)              (    7)
$      207.783              17.8558
$      74.662              2.52632
$      |                  |
$      513576              1000
$      CO-TABLE: TEST      CO-TABLE: TEST
$      HISTORY              TELLER
$      Q2                  Q1
```

PRESENTATION NOTES FOR PAGE 25

Notes:

Access Plan:

Total Cost: 226.575
Query Degree: 1

```

      Rows
      RETURN
      ( 1)
      Cost
      I/O
      |
      2374.76
      TBSCAN
      ( 2)
      226.575
      77.1883
      |
      2374.76
      SORT
      ( 3)
      226.479
      77.1883
      |
      2374.76
      CTQ
      ( 4)
      225.686
      77.1883
      |
      2374.76
      HSJOIN
      ( 5)
      225.661
      77.1883
      /-----\
      2336.77      1000
      TBSCAN      TBSCAN
      ( 6)      ( 7)
      207.783      17.8558
      74.662      2.52632
      |          |
      513576      1000
      CO-TABLE: TEST  CO-TABLE: TEST
      HISTORY      TELLER
      Q2          Q1
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Explain report for joining three column-organized tables

```

SELECT ACCT.ACCT_ID, .... HISTORY.TEMP
FROM   ACCT , TELLER , HISTORY
WHERE  ACCT.ACCT_ID = HISTORY.ACCT_ID AND
       ACCT.ACCT_GRP BETWEEN 100 AND 700 AND
       HISTORY.TELLER_ID = TELLER.TELLER_ID
ORDER BY HISTORY.PID ASC

```

```

$      Total Cost:
$      838.428
$
$      Query Degree:
$      1

```

*Row oriented
processing*

```

$      Rows
$      RETURN
$      ( 1)
$      Cost
$      I/O
$      |
$      310668
$      TBSCAN
$      ( 2)
$      838.428
$      139.857
$      |
$      310668
$      SORT
$      ( 3)
$      825.893
$      139.857
$      |

```

*Column oriented
processing*

```

$      310668
$      CTQ
$      ( 4)
$      626.331
$      139.857
$      |
$      310668
$      HSJOIN
$      ( 5)
$      620.916
$      139.857
$
$      /-----+-----\
$      305697                      1000
$      HSJOIN                      TBSCAN
$      ( 6)                      ( 9)
$      600.063                    18.9682
$      137.173                    2.68421
$
$      /-----+-----\
$      513576                    595232                    1000
$      TBSCAN                    TBSCAN    CO-TABLE: TEST
$      ( 7)                      ( 8)          TELLER
$      203.491                    388.68          Q2
$      84.2958                    52.8772
$      |
$      513576                    1e+06
$      CO-TABLE: TEST            CO-TABLE: TEST
$      HISTORY                    ACCT
$      Q1                          Q3

```

PRESENTATION NOTES FOR PAGE 26

Notes:

Access Plan:

Total Cost: 226.575
Query Degree: 1

```

      Rows
      RETURN
      ( 1)
      Cost
      I/O
      |
      2374.76
      TBSCAN
      ( 2)
      226.575
      77.1883
      |
      2374.76
      SORT
      ( 3)
      226.479
      77.1883
      |
      2374.76
      CTQ
      ( 4)
      225.686
      77.1883
      |
      2374.76
      HSJOIN
      ( 5)
      225.661
      77.1883
      /-----\
      2336.77      1000
      TBSCAN      TBSCAN
      ( 6)      ( 7)
      207.783      17.8558
      74.662      2.52632
      |      |
      513576      1000
      CO-TABLE: TEST  CO-TABLE: TEST
      HISTORY      TELLER
      Q2      Q1
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Explain report using db2expln for joining three column-organized tables

```

$ Estimated Cost = 849.247131
$ Estimated Cardinality = 323848.687500

$ ( 4) CDE Subquery
$   | Tables Referenced:
$   | | 1: TEST.TELLER ID = 4,6
$   | | 2: TEST.ACCT ID = 7,4
$   | | 3: TEST.HISTORY ID = 6,6
$   | #Output Columns = 10
$ ( 3) Insert Into Sorted Temp Table ID = t1
$   | #Columns = 10
$   | #Sort Key Columns = 1
$   | | Key 1: (Ascending)
$   | Sorthheap Allocation Parameters:
$   | | #Rows = 323849.000000
$   | | Row Width = 104
$   | Piped
$ ( 2) Access Temp Table ID = t1
$   | #Columns = 10
$   | Relation Scan
$   | | Prefetch: Eligible
$   | | Sargable Predicate(s)
$ ( 1) | | Return Data to Application
$   | | | #Columns = 10
$ ( 1) Return Data Completion

```

\$ Optimizer Plan:

```

$ Operator
$ (ID)
$
$ RETURN
$ ( 1)
$ |
$ TBSCAN
$ ( 2)
$ |
$ SORT
$ ( 3)
$ |
$ CTQ
$ ( 4)
$ |
$ *
$ +-----+-----+
$ Table: Table: Table:
$ TEST TEST TEST
$ HISTORY ACCT TELLER

```

PRESENTATION NOTES FOR PAGE 27

Notes:

Access Plan:

Total Cost: 226.575
Query Degree: 1

```

      Rows
      RETURN
      ( 1)
      Cost
      I/O
      |
      2374.76
      TBSCAN
      ( 2)
      226.575
      77.1883
      |
      2374.76
      SORT
      ( 3)
      226.479
      77.1883
      |
      2374.76
      CTQ
      ( 4)
      225.686
      77.1883
      |
      2374.76
      HSJOIN
      ( 5)
      225.661
      77.1883
      /-----\
      2336.77      1000
      TBSCAN      TBSCAN
      ( 6)      ( 7)
      207.783      17.8558
      74.662      2.52632
      |          |
      513576      1000
      CO-TABLE: TEST  CO-TABLE: TEST
      HISTORY      TELLER
      Q2          Q1
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Organized table

CO-TABLE: TEST	CO-TABLE: TEST
318667	318667
^HSJOIN	^HSJOIN
(4)	(4)
644.008	644.008
142.331	142.331
/-----+-----\	
318667	1000
CTQ	FETCH
(5)	(9)
605.559	28.6196
138.331	4
/---+---\	
318667	1000 1000
HSJOIN	IXSCAN TABLE: TEST
(6)	(10) TELLER
601.136	0.284191 Q2
138.331	0
/-----+-----\	
513576	620487 1000
TBSCAN	TBSCAN INDEX: SYSIBM
(7)	(8) SQL130617120709960
203.491	389.527 Q2
84.2958	54.0351
513576	1e+06

Row oriented processing

```

SELECT ACCT.ACCT_ID, ....
HISTORY.TEMP
FROM   ACCT , TELLER , HISTORY
WHERE  ACCT.ACCT_ID =
HISTORY.ACCT_ID AND
       ACCT.ACCT_GRP BETWEEN 100 AND
700 AND HISTORY.TELLER_ID =
TELLER.TELLER_ID
ORDER BY HISTORY.PID ASC
        
```

Row oriented processing

Column oriented processing

```
SELECT ACCT.ACCT_ID, ....
HISTORY.TEMP
FROM    ACCT , TELLER , HISTORY
WHERE   ACCT.ACCT_ID =
HISTORY.ACCT_ID AND
        ACCT.ACCT_GRP BETWEEN 100 AND
700 AND   HISTORY.TELLER_ID =
TELLER.TELLER_ID
ORDER BY   HISTORY.PID ASC
```

PRESENTATION NOTES FOR PAGE 28

Notes:

With row based teller table

1) RETURN: (Return Result)

Cumulative Total Cost:	861.83
Cumulative CPU Cost:	2.5161e+09
Cumulative I/O Cost:	142.331
Cumulative Re-Total Cost:	12.8572
Cumulative Re-CPU Cost:	7.96683e+07
Cumulative Re-I/O Cost:	0
Cumulative First Row Cost:	848.973
Estimated Bufferpool Buffers:	1056

With column based tables:

1) RETURN: (Return Result)

Cumulative Total Cost:	838.428
Cumulative CPU Cost:	2.43083e+09
Cumulative I/O Cost:	139.857
Cumulative Re-Total Cost:	12.5345
Cumulative Re-CPU Cost:	7.76684e+07
Cumulative Re-I/O Cost:	0
Cumulative First Row Cost:	825.894
Estimated Bufferpool Buffers:	1029

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Optimization guidelines and profiles for column-organized tables

- § Optimization guidelines and profiles are supported on column-organized tables
- § In general, any table access or join request not involving an index scan is allowed
 - TBSCAN or ACCESS are allowed
 - IXSCAN, LPREFETCH, IXAND, IOA are not allowed
- § STARJOIN is not allowed
- § A column-organized table supports only HSJOIN requests
 - Any join request that references column-organized tables can be satisfied by retrieving the data from the column-organized tables and performing the join by using row-organized data processing
 - This would be used for NLJOIN or MSJOIN
 - If the requested join method is HSJOIN, a plan with HSJOIN being pushed down to column-organized data processing can also be used to satisfy the request.

PRESENTATION NOTES FOR PAGE 29

Notes:

Optimization guidelines and profiles are supported on column-organized tables. In general, any table access or join request not involving an index scan is allowed.

SQL0437W with reason code 13 is returned when a guideline that references column-organized tables cannot be satisfied. Also, EXP0035W is displayed in the Extended Diagnostic Information section of db2exfmt command output

Instructor notes:

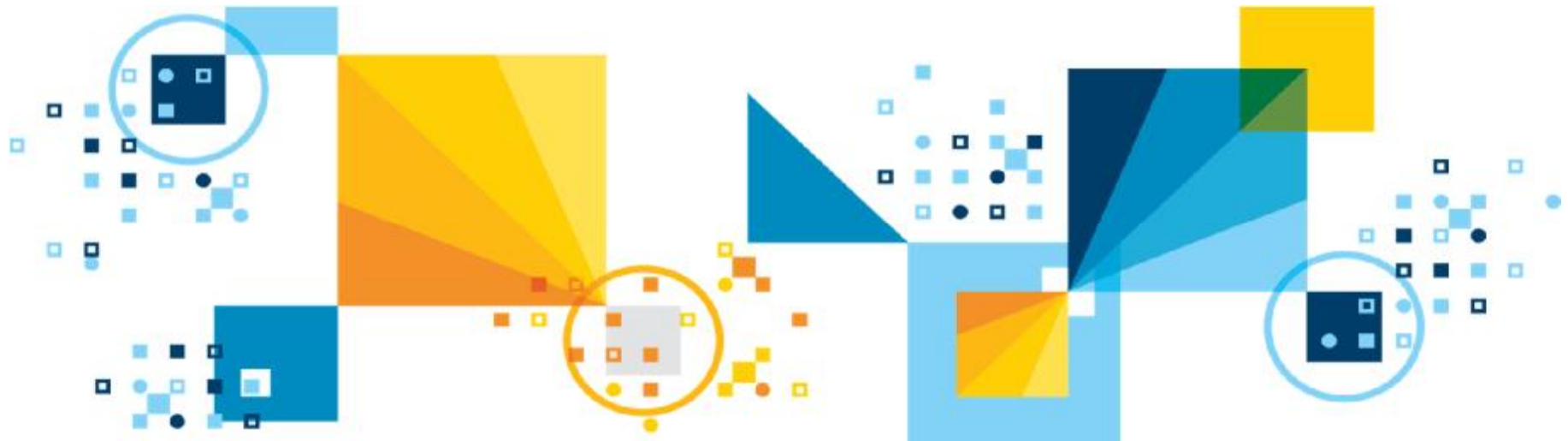
Purpose —

Details —

Additional information —

Transition statement —

DB2 Workload Management of databases with column-organized tables



PRESENTATION NOTES FOR PAGE 30

Default query concurrency management for AYALYTICS workload databases

- § Query processing using column-organized tables is designed to run fast by leveraging the highly parallelized in-memory processing of data
- § To ensure that heavier workloads on column-organized data do not overload the system when many queries are running simultaneously, DB2 can limit the number of *heavyweight* queries executing at the same time
- § The limit is implemented using a WLM concurrency threshold
 - Automatically enabled on new databases when DB2_WORKLOAD is set to ANALYTICS
 - Can be manually enabled on existing databases
- § Field experience with DB2 WLM has also demonstrated that in analytic environments controlling the admission of *heavyweight* queries into the system benefits stability and overall performance
- § When the limit on the number of heavyweight queries is reached, the remaining queries are queued
 - These must wait until other queries in this class complete before beginning their execution

PRESENTATION NOTES FOR PAGE 31

Notes:

Default query concurrency management

To ensure that heavier workloads on column-organized data do not overload the system when many queries are submitted simultaneously, there is a limit on the number of heavyweight queries that can execute on the database at the same time.

This limit can be implemented by using the default workload management concurrency threshold that is automatically enabled on new databases when the value of the DB2_WORKLOAD registry variable is set to ANALYTICS, or that can be manually enabled on existing databases.

The processing of queries against column-organized tables is designed to run fast by leveraging the highly parallelized in-memory processing of data. The trade-off for this high performance is that queries referencing column-organized tables also have a relatively large footprint in terms of memory and CPU when compared to similar queries processing row-organized table data. As such, the execution of these types of queries is optimal when relatively few of them are admitted to the system at a time. This enables them to individually leverage more processing power and memory on the system, and minimizes contention on the processor caches.

Field experience with DB2® workload management has also demonstrated that in analytic environments that support mixed workloads (where queries might vary widely in their degree of complexity and resource needs), controlling the admission of "heavyweight" queries into the system yields improvements in both system stability and overall performance, because resource overload on the system is avoided.

When the limit on the number of heavyweight queries is reached, the remaining queries are queued and must wait until other queries leave the system before beginning their execution. This can help to ensure system stability when a large number of complex ad hoc queries are running on systems that have not implemented a specific workload management strategy. Users who want to further optimize the execution of mixed workloads on their systems are encouraged to look at the full range of workload management capabilities offered in the DB2 database product

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Automatic Workload Management

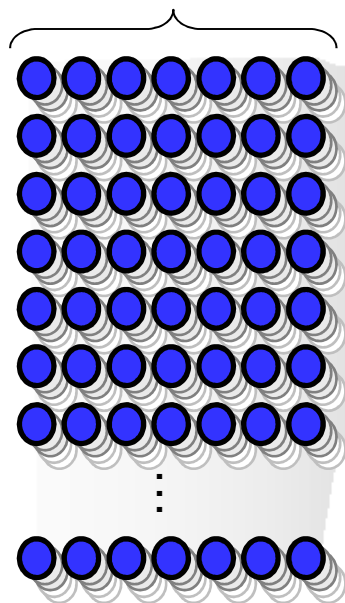
§ Built-in and automated query resource consumption control

§ Enabled automatically when DB2_WORKLOAD=ANALYTICS

§ Many queries can be submitted, but limited number get executed concurrently

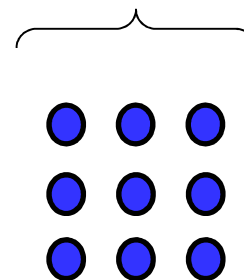
Applications and Users

Up to tens of thousands of SQL queries at once



DB2 DBMS kernel

Moderate number of queries consume resources



PRESENTATION NOTES FOR PAGE 32

DB2 Kepler has built-in and automated query resource consumption control

Every additional query that runs naturally consumes more memory, locks, CPU, and memory bandwidth. In other database products more queries means more contention.

DB2 Kepler automatically allows a high level of concurrent queries to be submitted, but limits the number that consume resources at any point in time

That means more memory and CPU for each query that's running. In the end, the entire workload benefits!

This is a bit like people leaving a crowded theatre. If 1000 people try to exit at the same in a mad rush it goes slowly. But if people line up courteously and exit 4 by 4, everyone gets through more efficiently.

Default workload management objects for concurrency control

- § Default query concurrency management is implemented using existing DB2 WLM infrastructure.
- § Several new default WLM objects will be created on both upgraded and newly created databases
 - A service subclass, **SYSDEFAULTMANAGEDSUBCLASS**
 - Under the existing **SYSDEFAULTUSERCLASS** superclass
 - Where heavyweight queries will run and can be controlled and monitored as a group
 - A **CONCURRENTDBCOORDACTIVITIES** threshold, **SYSDEFAULTCONCURRENT**
 - Which is applied to the **SYSDEFAULTMANAGEDSUBCLASS** subclass
 - Controls the number of concurrently executing queries
- § A work class set, **SYSDEFAULTUSERWCS**, and a new work class, **SYSMANAGEDQUERIES**, which identifies the class of heavyweight queries that are to be controlled
 - Work class encompasses queries that are classified as READ DML falling above a timeron threshold that reflects heavier queries
- § A work action set, **SYSDEFAULTUSERWAS**, and work action, **SYSMAPMANAGEDQUERIES**
 - Maps all queries that fall into the **SYSMANAGEDQUERIES** work class to the **SYSDEFAULTMANAGEDSUBCLASS** service subclass

PRESENTATION NOTES FOR PAGE 33

Notes:

Default workload management objects for concurrency control

Default query concurrency management is implemented by leveraging the existing DB2 workload management infrastructure. Several new default workload management objects will be created on both upgraded and newly created databases:

A service subclass, `SYSDEFAULTMANAGEDSUBCLASS`, under the existing `SYSDEFAULTUSERCLASS` superclass, where heavyweight queries will run and can be controlled and monitored as a group

A `CONCURRENTDBCOORDACTIVITIES` threshold, `SYSDEFAULTCONCURRENT`, which is applied to the `SYSDEFAULTMANAGEDSUBCLASS` subclass to control the number of concurrently executing queries that are running in that subclass

A work class set, `SYSDEFAULTUSERWCS`, and a new work class, `SYSMANAGEDQUERIES`, which identify the class of heavyweight queries that are to be controlled. The `SYSMANAGEDQUERIES` work class encompasses queries that are classified as READ DML (an existing work type for work classes) falling above a timeron threshold that reflects heavier queries.

A work action set, `SYSDEFAULTUSERWAS`, and work action, `SYSMAPMANAGEDQUERIES`, which map all queries that fall into the `SYSMANAGEDQUERIES` work class to the `SYSDEFAULTMANAGEDSUBCLASS` service subclass

Instructor notes:

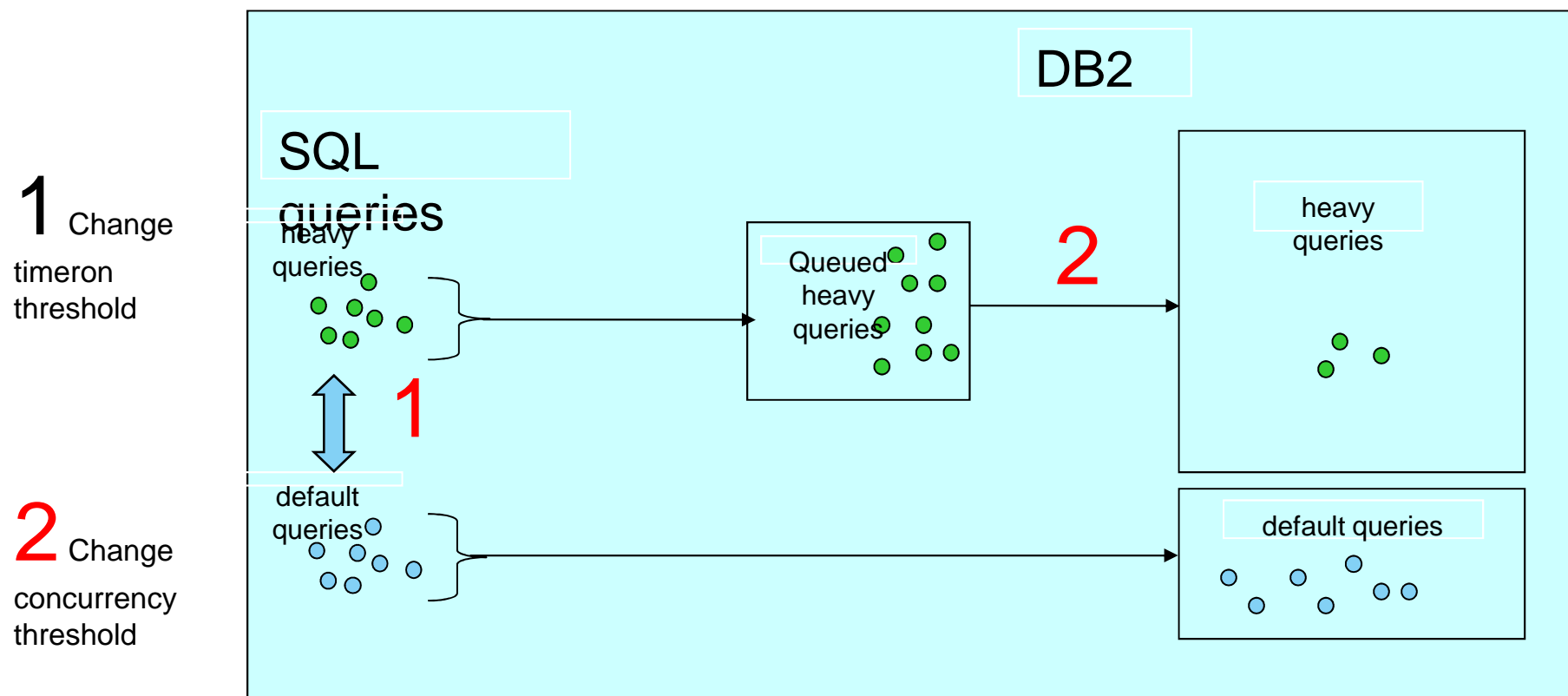
Purpose —

Details —

Additional information —

Transition statement —

Easily Tune default WLM controls for over-utilized or under-utilized state



Monitor ***wlm_queue_time_total***, using monitor functions like MON_GET_DATABASE, MON_GET_CONNECTION and MON_GET_SERVICE_SUBCLASS

PRESENTATION NOTES FOR PAGE 34

If the system appears to be running in an under-utilized state, take the following steps:

Examine the WLM_QUEUE_TIME_TOTAL metric, which is reported by various system-level table functions (or the statistics event monitor), to determine whether queries in the system are accumulating time waiting on concurrency thresholds.

If no such queue time is being accumulated, the system is simply running under peak capacity, and no tuning is necessary.

If queue time is being accumulated, work is being queued on the system. Monitor the amount of work running in SYSDEFAULTSUBCLASS and SYSDEFAULTMANAGEDSUBCLASS, and consider incrementally increasing the TIMERONCOST minimum on SYSMANAGEDQUERIES if it appears that too large a proportion of the workload is running within the managed class.

Assuming that the distribution of managed and unmanaged work appears reasonable, consider incrementally increasing the concurrency limit that is specified by SYSDEFAULTCONCURRENT until system resource usage reaches the target level.

If the system appears to be running in an over-utilized state, take the following steps:

Monitor the amount of work running in SYSDEFAULTSUBCLASS and SYSDEFAULTMANAGEDSUBCLASS, and consider incrementally decreasing the TIMERONCOST minimum on SYSMANAGEDQUERIES if it appears that too small a proportion of the workload is running within the managed class.

Assuming that the distribution of managed and unmanaged work appears reasonable, consider incrementally decreasing the concurrency limit that is specified by SYSDEFAULTCONCURRENT until system resource usage is back within the target range.

Adjusting the default WLM configuration

§ The mapping of heavyweight READ DML queries to SYSDEFAULTMANAGEDSUBCLASS can be enabled or disabled:

- To Enable the default work action mapping

```
ALTER WORK ACTION SET SYSDEFAULTUSERWAS ENABLE
```

- To Disable the default work action mapping so that all queries get mapped to SYSDEFAULTSUBCLASS

```
ALTER WORK ACTION SET SYSDEFAULTUSERWAS DISABLE
```

§ The timeron range for the mapping of heavyweight READ DML queries can be adjusted:

```
ALTER WORK CLASS SET SYSDEFAULTUSERWCS
```

```
ALTER WORK CLASS SYSMANAGEDQUERIES FOR TIMERONCOST  
FROM 100000 TO UNBOUNDED
```

§ The concurrency threshold on SYSDEFAULTMANAGEDSUBCLASS can be enabled or disabled:

- To Enable the concurrency threshold

```
ALTER THRESHOLD SYSDEFAULTCONCURRENT ENABLE
```

- To Disable the concurrency threshold

```
ALTER THRESHOLD SYSDEFAULTCONCURRENT DISABLE
```

§ The concurrency limit can be adjusted

```
ALTER THRESHOLD SYSDEFAULTCONCURRENT
```

```
WHEN CONCURRENTDBCOORDACTIVITIES > 100 STOP EXECUTION
```

PRESENTATION NOTES FOR PAGE 35

Notes:

Adjusting the default configuration

Note that only the following subset of workload management DDL is supported with the default workload management objects for concurrency control in the DB2 Version 10.5 release.

The mapping of heavyweight READ DML queries to SYSDEFAULTMANAGEDSUBCLASS can be enabled or disabled, as shown in the following example: # Enable the default work action mapping so that # heavyweight queries get mapped to # SYSDEFAULTMANAGEDSUBCLASS ALTER WORK ACTION SET SYSDEFAULTUSERWAS ENABLE # Disable the default work action mapping so that all # queries get mapped to SYSDEFAULTSUBCLASS ALTER WORK ACTION SET SYSDEFAULTUSERWAS DISABLE

The timeron range for the mapping of heavyweight READ DML queries can be adjusted, as shown in the following example: # Modify the default work class timeron range from A to B ALTER WORK CLASS SET SYSDEFAULTUSERWCS ALTER WORK CLASS SYSMANAGEDQUERIES FOR TIMERONCOST FROM <A> TO

The concurrency threshold on SYSDEFAULTMANAGEDSUBCLASS can be enabled or disabled, as shown in the following example: # Enable the concurrency threshold ALTER THRESHOLD SYSDEFAULTCONCURRENT ENABLE # Disable the concurrency threshold ALTER THRESHOLD SYSDEFAULTCONCURRENT DISABLE

The concurrency limit can be adjusted, as shown in the following example: # Change the concurrency limit to [N] ALTER THRESHOLD SYSDEFAULTCONCURRENT WHEN CONCURRENTDBCOORDACTIVITIES > [N] STOP EXECUTION

The following optional clauses are permitted when altering SYSDEFAULTCONCURRENT: alter-threshold-exceeded-actions AND QUEUEDACTIVITIES

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using SQL and db2pd to monitor processing column-organized tables



PRESENTATION NOTES FOR PAGE 36

Monitoring metrics for column-organized tables

§ column-organized tables utilize a new column-organized object from a storage perspective

§ Access to column-organized object pages are counted separate from other storage objects like data, index and xml

- Counters for total logical and physical column-organized data page reads

- POOL_COL_L_READS
- POOL_COL_P_READS
- POOL_COL_LBP_PAGES_FOUND

- Counter for column-organized data page writes - POOL_COL_WRITES

- Counters for asynchronous column-organized data page reads and writes and pages found:

- POOL_ASYNC_COL_READS
- POOL_ASYNC_COL_READ_REQS
- POOL_ASYNC_COL_WRITES
- POOL_ASYNC_COL_LBP_PAGES_FOUND

- Counters for column-organized data page reads per table:

- OBJECT_COL_L_READS
- OBJECT_COL_P_READS
- OBJECT_COL_LBP_PAGES_FOUND

PRESENTATION NOTES FOR PAGE 37

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Monitoring column-organized tables and synopsis tables using MON_GET_TABLE

```

$ SELECT VARCHAR(TABNAME,30) AS TABLE, VARCHAR(TABSCHEMA,12) AS SCHEMA,
$   ROWS_READ, OBJECT_DATA_L_READS AS DATA_L_READS,
$   OBJECT_COL_L_READS AS COLUMN_L_READS, OBJECT_COL_P_READS, OBJECT_COL_LBP_PAGES_FOUND
$ FROM TABLE(MON_GET_TABLE(NULL,NULL,-2)) AS T1
$ WHERE TABSCHEMA = 'TEST' OR (TABSCHEMA = 'SYSIBM' AND TABNAME LIKE 'SYN%')
$ ORDER BY TABSCHEMA,TABNAME

```

\$ TABLE	\$ SCHEMA	\$ ROWS_READ	\$ DATA_L_READS
\$ SYN130617110037170122_HISTORY	\$ SYSIBM	\$ 502	\$ 4
\$ SYN130617115333621920_ACCT	\$ SYSIBM	\$ 977	\$ 4
\$ SYN130618131822321797_TELLER	\$ SYSIBM	\$ 0	\$ 4
\$ ACCT	\$ TEST	\$ 606208	\$ 21
\$ HISTORY	\$ TEST	\$ 513576	\$ 35
\$ TELLER	\$ TEST	\$ 1000	\$ 23

\$ COLUMN_L_READS	\$ OBJECT_COL_P_READS	\$ OBJECT_COL_LBP_PAGES_FOUND
\$ 7	\$ 5	\$ 2
\$ 7	\$ 5	\$ 2
\$ 2	\$ 1	\$ 1
\$ 237	\$ 17	\$ 220
\$ 332	\$ 37	\$ 295
\$ 7	\$ 5	\$ 2

PRESENTATION NOTES FOR PAGE 38

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Monitoring Page Map Index statistics for column-organized tables using MON_GET_INDEX

```

§ SELECT VARCHAR(MON.TABNAME,12) AS TABLE, MEMBER,
§   VARCHAR(CAT.INDNAME,30) AS IX_NAME, MON.IID AS INDEX_ID, MON.NLEAF,
§   MON.OBJECT_INDEX_L_READS, MON.OBJECT_INDEX_P_READS,
§   MON.OBJECT_INDEX_LBP_PAGES_FOUND
§ FROM TABLE(MON_GET_INDEX('TEST',NULL,-2)) AS MON, SYSCAT.INDEXES AS CAT WHERE
§   MON.TABNAME = CAT.TABNAME AND MON.TABSCHEMA = CAT.TABSCHEMA
§   AND MON.IID = CAT.IID
§   AND MON.TABNAME IN ('ACCT','HISTORY','BRANCH','TELLER')
§   ORDER BY MON.TABNAME

```

§ TABLE	MEMBER	IX_NAME	INDEX_ID	NLEAF
§ ACCT	0	SQL130617115333860	1	1
§ HISTORY	0	SQL130617110037380	1	1
§ TELLER	0	SQL130618131822550	1	1

§ OBJECT_INDEX_L_READS	OBJECT_INDEX_P_READS	OBJECT_INDEX_LBP_PAGES_FOUND
§ 6	1	5
§ 9	1	8
§ 5	1	4

§ 3 record(s) selected.

PRESENTATION NOTES FOR PAGE 39

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Column-organized table join *sortheap* memory usage can be monitored using HASH join statistics

§ HASH join use of *sortheap* memory can be monitored using:

- TOTAL_HASH_JOINS
- TOTAL_HASH_LOOPS
- HASH_JOIN_OVERFLOW
- HASH_JOIN_SMALL_OVERFLOW
- POST_SHRTHRESHOLD_HASH_JOINS

§ These can be monitored at various levels

- Activity or package cache entry
- Connection, Unit of Work, Workload, Service Subclass, etc.
- Database level using MON_GET_DATABASE or MON_GET_DATABASE_DETAILS

PRESENTATION NOTES FOR PAGE 40

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Additional monitoring elements for column-organized table processing

§ The GROUP BY operator on column-organized tables uses hashing as the grouping method.

- Hashed GROUP BY operators are consumers of sort memory
- The following new monitor elements support the monitoring of sort memory consumption during hashed GROUP BY operations
 - TOTAL_HASH_GRPBYS
 - ACTIVE_HASH_GRPBYS
 - HASH_GRPBY_OVERFLOW
 - POST_THRESHOLD_HASH_GRPBYS

§ New time-spent monitor elements

- TOTAL_COL_TIME - represents total elapsed time over all column-organized processing subagents
- TOTAL_COL_PROC_TIME - represents the subset of TOTAL_COL_TIME in which the column-organized processing subagents were not idle on a measured wait time (for example: lock wait, IO)
- TOTAL_COL_EXECUTIONS - the total number of times that data in column-organized tables was accessed during statement execution.

PRESENTATION NOTES FOR PAGE 41

Notes:

Use new monitor elements for the hashed GROUP BY operator

The GROUP BY operator on column-organized tables uses hashing as the grouping method. Hashed GROUP BY operators are consumers of sort memory. The following new monitor elements support the monitoring of sort memory consumption during hashed GROUP BY operations. These elements are similar to existing monitor elements for other sort memory consumers.

TOTAL_HASH_GRPBYS

ACTIVE_HASH_GRPBYS

HASH_GRPBY_OVERFLOWES

POST_THRESHOLD_HASH_GRPBYS

Time-spent monitor elements provide information about how the DB2 database manager is spending time processing column-organized tables. The time-spent elements are broadly categorized into wait times and processing times.

The following monitor elements are added to the time-spent hierarchy:

TOTAL_COL_TIME

TOTAL_COL_PROC_TIME

TOTAL_COL_EXECUTIONS

The three TOTAL_* metrics count the total time that is spent in column-organized data processing across all column-organized processing subagents.

TOTAL_COL_TIME represents total elapsed time over all column-organized processing subagents. TOTAL_COL_PROC_TIME represents the subset of TOTAL_COL_TIME in which the column-organized processing subagents were not idle on a measured wait time (for example: lock wait, IO).

TOTAL_COL_EXECUTIONS represents the total number of times that data in column-organized tables was accessed during statement execution.

The parent metric of TOTAL_COL_TIME is TOTAL_SECTION_TIME. The parent metric of TOTAL_COL_PROC_TIME is TOTAL_SECTION_PROC_TIME. The parent metrics are the same in both the request and activity dimensions.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Monitor elements to monitor prefetch requests for data in column-organized tables

- § DB2 tracks buffer pool hit rates for each column so prefetch can be enabled or disabled on access to column data
- § New monitor elements to measure prefetcher efficiency for data in column-organized tables that are being submitted to prefetchers and the number of pages that prefetchers skipped reading because the pages were already in memory
- § Efficient prefetching of data in column-organized tables is important for mitigating the I/O costs of data scans
- § The following monitor elements enable the monitoring of prefetch requests for data in column-organized tables:
 - POOL_QUEUED_ASYNC_COL_REQS
 - POOL_QUEUED_ASYNC_COL_PAGES
 - POOL_FAILED_ASYNC_COL_REQS
 - SKIPPED_PREFETCH_COL_P_READS
 - SKIPPED_PREFETCH_UOW_COL_P_READS

PRESENTATION NOTES FOR PAGE 42

Notes:

The prefetch logic for queries that access column-organized tables is used to asynchronously fetch only those pages that each thread will read for each column that is accessed during query execution. If the pages for a particular column are consistently available in the buffer pool, prefetching for that column is disabled until the pages are being read synchronously, at which time prefetching for that column is enabled again.

Although the number of pages that a thread can prefetch simultaneously is limited by the prefetch size of the table space that is being accessed, several threads can also prefetch pages simultaneously.

New monitor elements to measure prefetcher efficiency can help you to track the volume of requests for data in column-organized tables that are being submitted to prefetchers, and the number of pages that prefetchers skipped reading because the pages were already in memory. Efficient prefetching of data in column-organized tables is important for mitigating the I/O costs of data scans.

The following monitor elements enable the monitoring of prefetch requests for data in column-organized tables:

POOL_QUEUED_ASYNC_COL_REQS

POOL_QUEUED_ASYNC_TEMP_COL_REQS

POOL_QUEUED_ASYNC_COL_PAGES

POOL_QUEUED_ASYNC_TEMP_COL_PAGES

POOL_FAILED_ASYNC_COL_REQS

POOL_FAILED_ASYNC_TEMP_COL_REQS

SKIPPED_PREFETCH_COL_P_READS

SKIPPED_PREFETCH_TEMP_COL_P_READS

SKIPPED_PREFETCH_UOW_COL_P_READS

SKIPPED_PREFETCH_UOW_TEMP_COL_P_READS

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Monitoring Database statistics with column-organized tables using MON_GET_DATABASE

```

$ SELECT ROWS_READ, ROWS_RETURNED,
$      TOTAL_SORTS, SORT_OVERFLOW,
$      TOTAL_HASH_JOINS, HASH_JOIN_OVERFLOW,
$      POOL_COL_L_READS, TOTAL_COL_TIME,
$      TOTAL_HASH_GRPBYS, HASH_GRPBY_OVERFLOW, SORT_SHRHEAP_TOP
$      FROM TABLE(MON_GET_DATABASE(-1))

```

```

$ ROWS_READ      ROWS_RETURNED      TOTAL_SORTS      SORT_OVERFLOW
$ -----
$              3532131              69427              47              0

```

```

$ TOTAL_HASH_JOINS  HASH_JOIN_OVERFLOW  POOL_COL_L_READS  TOTAL_COL_TIME
$ -----
$                14                0                2190                2798

```

```

$ TOTAL_HASH_GRPBYS  HASH_GRPBY_OVERFLOW  SORT_SHRHEAP_TOP
$ -----
$                 3                 0                12000

```

```

$ 1 record(s) selected.

```

PRESENTATION NOTES FOR PAGE 43

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Monitor column-organized table LOAD using db2pd command – utilities option

\$ Database Member 0 -- Active -- Up 0 days 00:25:21 -- Date 2013-05-20-08.40.33.104992

\$ Utilities:

\$ Address StartTime	ID DBName	Type NumPhases	CurPhase	State Description	Invoker	Priority
0x000000020557F540	3	LOAD		0	0	0
May 20 08:40:20	TESTBLU 4	3		[LOADID: 50.2013-05-20-08.40.20.042757.0		Mon
(4;6)]	[*LOCAL.inst20.130520122733] OFFLINE LOAD DEL AUTOMATIC INDEXING REPLACE COPY NO					
INST20 .ACCT						

\$ Progress:

\$ Address StartTime	ID Description	PhaseNum	CompletedWork	TotalWork
0x000000020557F868	3	1	0 bytes	0 bytes
Mon May 20 08:40:20	SETUPo			
0x000000020557FA20	3	2	1000000 rows	1000000 rows
Mon May 20 08:40:20	ANALYZE1			
0x000000020557FBA8	3	3	831694 rows	1000000 rows
Mon May 20 08:40:26	LOADm			
0x000000020557FD30	3	4	0 indexes	2 indexes
NotStarted	BUILD			

Analyze Phase shown for column-organized Table

PRESENTATION NOTES FOR PAGE 44

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

- § Having completed this unit, you should be able to:
- § Implement DB2 BLU Acceleration support for a new or existing DB2 database
- § Configure a DB2 database that uses DB2 BLU Acceleration, column-organized tables, including sort memory and utility heap memory considerations
- § Describe the default workload management used for DB2 BLU Acceleration processing and how you can tailor the WLM objects to efficiently use system resources
- § Monitor a DB2 database or application that uses column-organized tables using SQL monitor functions
- § Locate the column-organized processing portion of the access plans for column-organized tables in DB2 explain reports
- § Use db2convert or ADMIN_MOVE_TABLE to convert row-organized tables to column-organized tables

PRESENTATION NOTES FOR PAGE 45

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Agenda

§ Problematic SQL & Situation

§ Response time solution

§ SQL costs solution

- Snapshot Monitoring
- Event Monitoring
- SQL Monitoring Interfaces
- Analyzing SQL
 - Explain Tools
 - Visual Explain
- Statement Concentrator

§ Making Performance Improvements

- Database Objects
- Better Coding
- Design Advisor
- DB2 Optim Query Workload Tuner
- Other Considerations

PRESENTATION NOTES FOR PAGE 46

Indexes – Benefits and uses

- § Apply predicates to provide rapid look-up of the location of data in a database
- § Reduce the number of rows navigated
- § Try to avoid sorts for ORDER BY and GROUP BY clauses
- § Create an index on:
 - Join columns
 - Selective filter columns
 - Columns frequently used for ordering
- § To provide index-only access, which avoids the cost of accessing data pages
- § As the only automatic way to enforce uniqueness in a relational database

```
CREATE UNIQUE INDEX EMP_IX ON EMPLOYEE(EMPNO) INCLUDE(FIRSTNAME, JOB)
```

PRESENTATION NOTES FOR PAGE 47

The INCLUDE clause, applicable only on unique indexes, specifies additional columns to be appended to the set of index key columns. Any columns included with this clause are not used to enforce uniqueness. These included columns can improve the performance of some queries through index only access. This option might:

- Eliminate the need to access data pages for more queries

- Eliminate redundant indexes

A clustered index is most useful for columns that have range predicates because it allows better sequential access of data in the table. This results in fewer page fetches, since like values are on the same data page.

Indexes – Overhead

- § They add extra CPU and I/O cost to UPDATE, INSERT, DELETE and LOAD operations
- § They add to “query prepare time” because they provide more choices for the optimizer
- § They can use a significant amount of disk storage

PRESENTATION NOTES FOR PAGE 48

Indexes can't be haphazardly defined because they come with a certain amount of overhead so a balance must be struck.

Indexes must be maintained, as data is added or updated to/on a table, it's indexes must be maintained. You will incur IO and CPU costs for every index you define so it is important to make sure that the benefits of adding an index out ways the overhead. The db2pd command can be used to monitor how much an index is used. Many software installations will have predefined indexes and it could very well be the case that these are not used depending on what you do with the application. Extraneous indexes should be dropped.

Another area where extra indexes can negatively affect performance is in the optimizer when potential access plans are prepared and assessed.

And finally indexes consume and use disk storage resources.

Indexes – Best Practices

- § Index every PK and most FKs in a database
- § Indexes on FKs also improve the performance of RI checking
- § Explicitly provide an index for the PK
- § Columns frequently referenced in WHERE clauses are good candidates for an index
- § An exception to this rule is when the predicate provides minimal filtering
 - An example is an inequality such as WHERE cost <> 4. Indexes are seldom useful for inequalities because of the limited filtering provided
- § Specify indexes on columns used for equality and range queries



PRESENTATION NOTES FOR PAGE 49

This is just a small list of best practices for indexes, with an emphasis on primary and foreign keys.

Index every PK and most FKs in a database. Most joins occur between PKs and FKs, so it is important to build indexes on all PKs and FKs whenever possible.

Explicitly provide an index for the PK. The DB2 database manager indexes the PK automatically with a system-generated name if one is not specified. The system-generated name for an automatically-generated index is difficult to administer.

Indexes on foreign keys can benefit performance in a number of ways including the checking of RI, this occurs for instance when an insert is performed on a child row and a parent row with an identical value must be verified before the insert is allowed.

DB2 will automatically create a unique index when a primary key is specified when creating a table, however it is recommended to explicitly create a unique index and define it as the primary key in order to avoid a system generated index name.

Certain queries like ones with range based and equality predicates can help you define the columns that are good candidates for indexes.

Additional BP's

Avoid redundant indexes.

Use numeric data types when possible to index data, really the key thing is to use data types for indexed columns that are "smaller" in terms of the storage required to define them.

Avoid defining and index with more than 5 columns. Though compound indexes can enhance overall index efficiency too many columns will have diminished returns due to the overhead of maintaining complex indexes.

Index Management Redefined in DB2 10



§ Jump scan

- Need fewer indexes

§ Smart index prefetching

- Faster index access & fewer index reorgs

§ Smart data prefetching

- Faster data scans & fewer data reorgs

§ Predicate evaluation avoidance

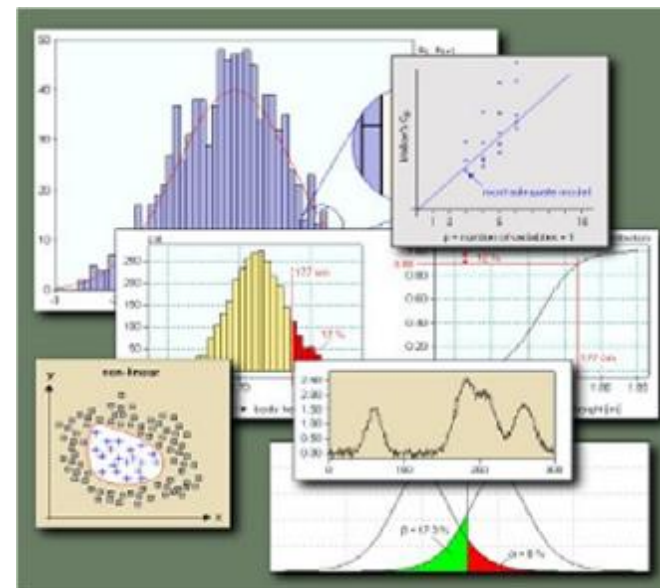
- Faster index scans

§ Higher performance

- Faster index performance

§ Lower costs

- Fewer indexes to maintain
- Dramatic reduction in index reorgs



PRESENTATION NOTES FOR PAGE 50

Jump Scan – skip sequential Improve performance of indexes with gaps in column definitions

Reduced number of indexes

Smart Index Pre-fetching Improve performance of index scans for poorly organized indexes

Reduced need for index reorgs

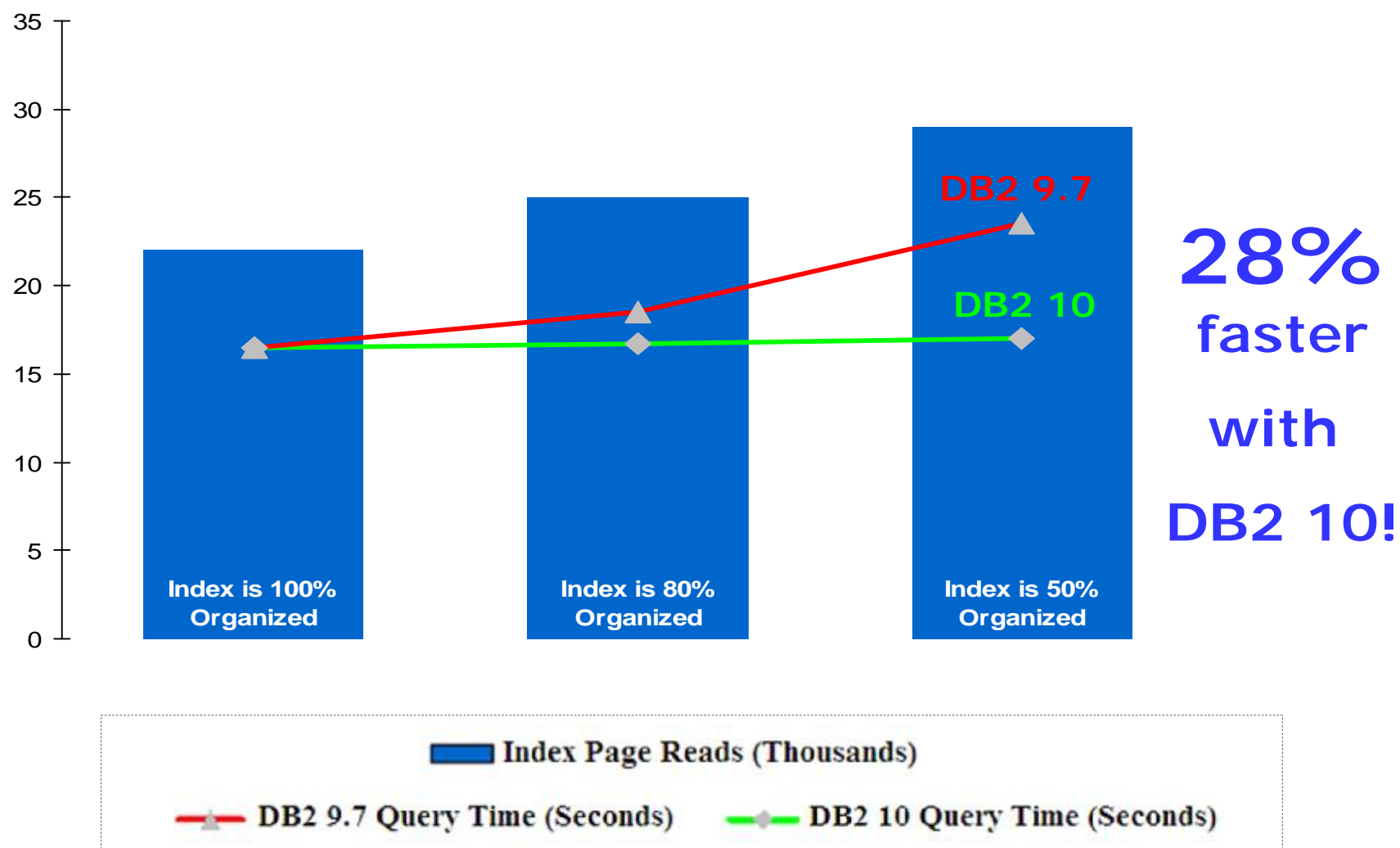
Smart Data Pre-fetching Improve performance of index scan for + data fetch for poorly clustered tables

Reduced need for table reorgs

Predicate evaluation avoidance for duplicate keys

Improve performance of index scans with duplicate keys

Smart Index Prefetching = Fewer Index Reorgs



PRESENTATION NOTES FOR PAGE 51

DB2 V10.5 Index improvements

§ **Expression based indexes improve query performance**

§ You can create an index that includes expressions.

§ Best suited when you want an efficient evaluation of queries that involve a column expression.

§ Values are transformed by the expressions that you specify.

§ For indexes created with the UNIQUE option, the uniqueness is enforced against the values that are stored in the index. The uniqueness is not enforced against the original column values.

§ **Clause EXCLUDE NULL KEYS from Index**

§ Resolves issues around sparsity of key values

§ Reduces size of index object

PRESENTATION NOTES FOR PAGE 52

Materialized Query Tables (MQTs)

§ Powerful way to improve response time for complex queries

§ Queries that might require one or more of the following operations:

- Aggregate data over one or more dimensions
- Join and aggregate data over a group of tables
- Data from a commonly accessed subset of data
- Repartitioned data from a table

§ Without MQTs, similar queries that do the same expensive operations may repeatedly compute the same results however:

- Consume extra disk space
- Must be updated to maintain their consistency
- Requires its own indexes for efficient access

§ The benefit of an MQT relative to its cost is therefore maximized if the MQT benefits many queries, particularly costly queries, or frequently executed queries

PRESENTATION NOTES FOR PAGE 53

Materialized query tables (MQTs) are a powerful way to improve response time for complex queries.

This is especially true for queries that might require one or more of the following operations:

Aggregate data over one or more dimensions

Joins and aggregate data over a group of tables

Data from a commonly accessed subset of data; that is, from a "hot" horizontal or vertical database partition

Re-partitioned data from a table, or part of a table, in a partitioned database environment

Without MQTs, similar queries that do the same expensive operations – such as joins and/or aggregation – may repeatedly compute the same results.

However, MQTs redundantly store data that is derivable from other data, so they consume extra disk space and must be updated to maintain their consistency with the source data whenever it changes, either periodically (deferred or full refresh) or as part of the same transaction (immediate refresh). Furthermore, an MQT, as any other stored table, requires its own indexes for efficient access.

The benefit of an MQT relative to its cost is therefore maximized if the MQT benefits many queries, particularly costly queries, or frequently executed queries.

You can override the optimizer decision by forcing it to choose specific MQTs with the MQTENFORCE element using optimization profiles.

Materialized Query Tables (MQTs)

§ An example of creating an MQT is shown below. The table definition is written just as it is when you use DDL for creating a normal table, but you do not define the columns and data types. You write an SQL query that describes the structure of the MQT, and then the MQT is filled with the data that the query returns:

```
CREATE TABLE MY_MQT AS (  
    SELECT SUM(T1.Sales) as Total_Sales,  
           SUM(T1.COGS) as Total_Costs,  
           SUM(T1.Expenses) as Total_Expenses,  
           T2.Prd_Family as Product_Family,  
           T3.Years Year, T3.Quarter as Quarter  
    FROM SALES_FACT T1, PRODUCT T2, TIME T3  
    WHERE T1.Product_ID = T2.Product_ID AND  
           T1.Time_ID = T3.Time_ID  
    GROUP BY T2.Prd_Family, T3.Year, T3.Quarter)  
DATA INITIALLY DEFERRED REFRESH DEFERRED  
ENABLE QUERY OPTIMIZATION MAINTAINED BY SYSTEM
```

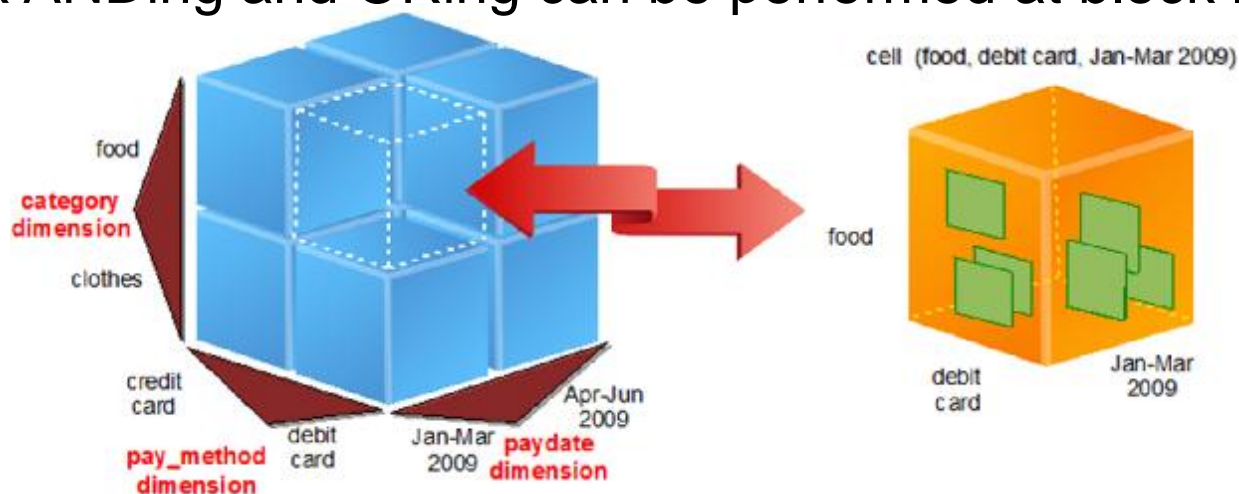
§ Based on the MQT DDL above, the DB2 Optimizer decides whether to use the base table or the MQT. When a query is run against the database, the DB2 Optimizer rewrites the query if the query matches the MQT definition and using the MQT would reduce the query costs. When the access plan has a better result with the MQT, then the query is run against the MQT to access the data

PRESENTATION NOTES FOR PAGE 54

To determine whether the DB2 Optimizer uses the created MQTs, you can use the DB2 Explain utility.

Multidimensional Clustering (MDC)

- § Enables a table to be physically clustered on several dimensions simultaneously
- § Primarily intended for data warehousing and large database environments
- § DB2 places records that have the same column values in physical locations that are close together
 - Block indexes are smaller than RID indexes
 - Faster lookup
 - Scan only required blocks
 - Index ANDing and ORing can be performed at block level



PRESENTATION NOTES FOR PAGE 55

MDC is a technique for clustering data along more than one dimension at the same time. However, you can also use MDC for single-dimensional clustering, just as you can use a clustering index. An advantage of an MDC table is that it is designed to always be clustered. A reorganization is never required to re-establish a high-cluster ratio.

Block indexes are smaller than RID indexes so its faster to lookup. Since everything is clustered, only the required blocks are scanned and everything is done at the block level , eliminating the need to include non-pertinent information.

MDC table optimization strategies can also exploit the performance advantages of intra-partition parallelism and inter-partition parallelism. Consider the following specific advantages of MDC tables:

Dimension block index lookups can identify the required portions of the table and quickly scan only the required blocks.

Because block indexes are smaller than record identifier (RID) indexes, lookups are faster.

Index ANDing and ORing can be performed at the block level and combined with RIDs.

Data is guaranteed to be clustered on extents, which makes retrieval faster.

Rows can be deleted faster when rollout can be used.

Multidimensional Clustering differences

§ Without MDC:

- Traditional indexes refer to records
- Traditional tables are managed by page
- Traditional tables can have only one clustering index!
 - Access via the clustering index reduces the number of pages that need to be read

§ With MDC:

- MDC indexes refer to blocks
- MDC tables are managed by block
- Each row in a block has the same values for the MDC dimensions
- MDC tables can be clustered on more than one key
- MDC tables can have MDC indexes and ordinary (RID-based) indexes



PRESENTATION NOTES FOR PAGE 56

It's clustering data at table level.

Multidimensional clustering (MDC) provides an elegant method for clustering data in tables along multiple dimensions in a flexible, continuous, and automatic way. MDC can significantly improve query performance.

In addition, MDC can significantly reduce the overhead of data maintenance, such as reorganization and index maintenance operations during insert, update, and delete operations. MDC is primarily intended for data warehousing and large database environments, but it can also be used in online transaction processing (OLTP) environments.

MDC enables a table to be physically clustered on more than one key, or dimension, simultaneously. With MDC, the benefits of single-dimensional clustering are therefore extended to multiple dimensions, or clustering keys. Query performance is improved where there is clustering of one or more specified dimensions of a table. Not only will these queries access only those pages having records with the correct dimension values, these qualifying pages will be grouped into blocks, or extents.

Second, although a table with a clustering index can become unclustered over time, in most cases an MDC table is able to maintain and guarantee its clustering over all dimensions automatically and continuously. This eliminates the need to frequently reorganize MDC tables to restore the physical order of the data. While record order within blocks is always maintained, the physical ordering of blocks (that is, from one block to another, in a block index scan) is not maintained on inserts (or even on the initial load, in some cases).

Third, in MDC the clustering indexes are block-based. These indexes are drastically smaller than regular record-based indexes, so take up much less disk space and are faster to scan.

Agenda

§ Problematic SQL & Situation

§ Response time solution

§ SQL costs solution

- Snapshot Monitoring
- Event Monitoring
- SQL Monitoring Interfaces
- Analyzing SQL
 - Explain Tools
 - Visual Explain
- Statement Concentrator

§ Making Performance Improvements

- Database Objects
- Better Coding
- Design Advisor
- DB2 Optim Query Workload Tuner
- Other Considerations

PRESENTATION NOTES FOR PAGE 57

Writing Efficient SELECT statements

§ Specify only columns that you need

§ Use predicates that limit to those rows that you need

§ Avoid numeric data type conversions

§ Avoid data type mismatches

§ Preferred data types

- CHAR instead of VARCHAR for short columns
- INTEGER instead of FLOAT, DECIMAL or DECFLOAT
- DECFLOAT instead of DECIMAL
- DATETIME instead of CHAR
- NUMERIC instead of CHAR

§ Avoid DISTINCT or ORDER BY if not required

§ Use IN list if same column used in multiple predicates

PRESENTATION NOTES FOR PAGE 58

Avoid numeric data type conversions whenever possible. When comparing values, try to use items that have the same data type. If conversions are necessary, inaccuracies due to limited precision, and performance costs due to runtime conversions might result.

Avoiding data type mismatches on join columns - In some cases, data type mismatches prevent the use of hash joins. Hash join has some extra restrictions on the join predicates beyond other join methods. In particular, the data types of the join columns must be exactly the same. For example, if one join column is FLOAT and the other is REAL, hash join is not supported. Additionally, if the join column data type is CHAR, GRAPHIC, DECIMAL, or DECFLOAT the lengths must be the same.

To decrease the probability that a sort operation will occur, omit clauses such as DISTINCT or ORDER BY if such operations are not required.

Consider using an IN list if the same column appears in multiple predicates. For large IN lists that are used with host variables, looping a subset of the host variables might improve performance.

Writing Efficient SELECT statements

- § Use OPTIMIZE FOR n ROWS clause
- § Use FETCH FIRST n ROWS ONLY clause
- § Use OPTIMIZE FOR n ROWS clause with the FETCH FIRST n ROWS ONLY clause
- § Use FOR READ ONLY or FOR FETCH ONLY clause
- § Use FOR UPDATE OF clause
- § Use FOR READ ONLY clause along with USE AND KEEP UPDATE LOCKS clause

PRESENTATION NOTES FOR PAGE 59

Use the OPTIMIZE FOR n ROWS clause to give priority to retrieving the first n rows from the full result set. The OPTIMIZE FOR clause declares the intent to retrieve only a subset of the result or to give priority to retrieving only the first few rows. The optimizer can then choose access plans that minimize the response time for retrieving the first few rows. In addition, the number of rows that are sent to the client as a single block are limited by the value of n. Thus the OPTIMIZE FOR clause affects how the server retrieves qualifying rows from the database, and how it returns those rows to the client. This clause affects both the choice of access plan and the number of rows that are blocked in the communication buffer.

The FETCH FIRST n ROWS ONLY clause sets the maximum number of rows that can be retrieved. Limiting the result table to the first several rows can improve performance. Only n rows are retrieved, regardless of the number of rows that the result set might otherwise contain.

If the OPTIMIZE FOR clause and the FETCH FIRST clause are both specified, the lower of the two n values affects the communications buffer size. The two values are considered independent of each other for optimization purposes. The OPTIMIZE FOR n ROWS clause indicates to the optimizer that the application intends to retrieve only n rows, but the query will return the complete result set. The FETCH FIRST n ROWS ONLY clause indicates that the query should return only n rows.

To take advantage of row blocking and improve performance, specify the FOR READ ONLY or FOR FETCH ONLY clause. Concurrency improves as well, because exclusive locks are never held on the rows that are retrieved. Additional query rewrites can also occur. Specifying these clauses, as well as the BLOCKING ALL bind option, can similarly improve the performance of queries running against nicknames in a federated database system. For result tables where updates and deletions are allowed, specifying the FOR READ ONLY clause can improve the performance of fetch operations if the database manager can retrieve blocks of data instead of using exclusive locks. Do not specify the FOR READ ONLY clause in queries that are used in positioned UPDATE or DELETE statements. In DB2 CLI applications, you can use the CLI connection attribute SQL_ATTR_ACCESS_MODE for the same purpose.

Updatable SELECT statements (using UPDATE/DELETE WHERE CURRENT OF statements) are non-blocking queries, so you should use them only when absolutely necessary.

An updatable SELECT ensures that the row has not changed between the time the SELECT is completed and the UPDATE/DELETE is issued. If this level of concurrency is not important to your application, an alternative is to use a DELETE or UPDATE with search criteria based on the values returned from a non-updateable SELECT (with FOR READ ONLY).

The FOR UPDATE clause limits the result set by including only those columns that can be updated by a subsequent positioned UPDATE statement. If you specify the FOR UPDATE clause without column names, all columns that can be updated in the table or view are included. If you specify column names, each name must be unqualified and must identify a column of the table or view. This enables the database manager to choose more appropriate locking levels initially and to avoid potential deadlocks. Note that FOR UPDATE cursors cannot take advantage of row blocking.

For cursors that will be used with searched updates, specify the FOR READ ONLY and the USE AND KEEP UPDATE LOCKS clauses to avoid deadlocks and still allow row blocking by forcing U locks on affected rows.

DB2 10 Up to 3x Faster Query Performance

§ Multi-core parallelism enhancements

§ Performance improvements in for:

- Queries over star schemas
- Queries with joins and sorts
- Queries with aggregation
- Hash joins

§ Higher performance

- Up to **35% faster** out-of-the-box performance
- Up to **3x faster** than DB2 9.7

§ Lower costs

- No need of more/new hw
- Postpone hardware upgrades



PRESENTATION NOTES FOR PAGE 60

How is out-of-the-box query performance being further enhanced in DB2 10?

Building on prior release performance improvements (such as, automatic performance improvements and RUNSTATS command improvements), DB2 10 performance improvements focus on reducing CPU processing time without causing significant administration or application changes. Most performance improvements are implemented by simply upgrading to DB2 10. You can gain significant performance improvements from improved query optimizer techniques and functionality including star schema query optimization, improved data and index prefetching, and improved use of statistical views. There are also additional RUNSTATS command improvements, improved performance for queries on tables with composite indexes, as well as improved multi-core parallelism.

DB2 10 makes it simpler for you to write and run efficient SQL queries, as well as, making your existing SQL queries run faster, often without any changes.

Agenda

§ Problematic SQL & Situation

§ Response time solution

§ SQL costs solution

- Snapshot Monitoring
- Event Monitoring
- SQL Monitoring Interfaces
- Analyzing SQL
 - Explain Tools
 - Visual Explain
- Statement Concentrator

§ Making Performance Improvements

- Database Objects
- Better Coding
- Design Advisor
- DB2 Optim Query Workload Tuner
- Other Considerations

PRESENTATION NOTES FOR PAGE 61

Using Design Advisor

§ Create the explain tables:

- The EXPLAIN.DDL file is located in the **\$INSTHOME/sql/lib/misc** directory
- Go to this directory and issue the `db2 -tvf EXPLAIN.DDL` command

§ Run the EXPLAIN.DDL DB2 command file:

```
db2 CONNECT TO database-name  
db2 -tvf EXPLAIN.DDL
```

§ Command Line Usage

```
db2advise -d sample -m MICP -i da.sql
```

§ Parameters:

- d database name
- m M-MQT I-Indexes C-MDC P-Partitioning

Workload type keyword: (choose one)

- s single SQL statement
- i SQL from input file
- w SQL from ADVISE_WORLOAD table by workload name
- g Get workload from dynamic SQL snapshot
- wlm SQL from the ACTIVITY Event Monitor tables

Other keywords:

- t specifies the maximum time, in minutes, to complete the operation

PRESENTATION NOTES FOR PAGE 62

You can invoke the DB2 Design Advisor on the command line using db2advis

-l disk-limit

Specifies the number of megabytes available for all recommended indexes and materialized views in the existing schema. Specify -1 to use the maximum possible size. The default value is 20% of the total database size.

Explain tables

The Explain tables capture access plans when the Explain facility is activated. The Explain tables must be created before Explain can be invoked. You can create them using one of the following methods:

Call the SYSPROC.SYSINSTALLOBJECTS procedure: db2 CONNECT TO database-name db2 CALL SYSPROC.SYSINSTALLOBJECTS('EXPLAIN', 'C', CAST (NULL AS VARCHAR(128)), CAST (NULL AS VARCHAR(128))) This call creates the explain tables under the SYSTOOLS schema. To create them under a different schema, specify a schema name as the last parameter in the call.

Run the EXPLAIN.DDL DB2® command file: db2 CONNECT TO database-name db2 -tf EXPLAIN.DDL This command file creates explain tables under the current schema. It is located at the DB2PATH\misc directory on Windows® operating systems, and the INSTHOME/sql/lib/misc directory on Linux® and UNIX® operating systems. DB2PATH is the location where you install your DB2 copy and INSTHOME is the instance home directory.

Using Design Advisor

§ Command Line Examples

–Example: All object types using an input file of SQL statements

```
db2advis -d sample -m MICP -i da.sql
```

–Example: For a single SQL statement

```
db2advis -d TPCD -s "Select * from part where partkey = 1"
```

```
db2advis -d TPCD -w wk1 -m M -c sim_space -b mqt_space -q newschema  
recommendations
```

```
db2advis -d TPCD -wlm db2activities -m MICP -o advise.out
```

–Example: Workload from then Activity Event Monitor tables

PRESENTATION NOTES FOR PAGE 63

Here are few examples

In the first example with the `-m` option the design advisor will provide recommendations for MQTs, Indexes, MDC tables as well as Partitioning of the tables (in a partitioned database.)

In the second example we see how DB2 Design Advisor can be invoked for a single SQL statement on the command line

In the third example we are specifying a workload using the name 'wk1'. This name is used in the `ADVISE_WORKLOAD` table where the SQL statements defining the workload are available. Using the `-c` option we specify a simulating tablespace in which to create the simulation catalog tables. The default is `USERSPACE1`.

With the `-b` option we specify a table space in which new MQTs will be created.

`-q schema-name`, specifies the qualifying name of unqualified names in the workload. It serves as the schema name to use for `CURRENT SCHEMA` when `db2advis` executes. The default schema name is the user ID of the person executing the command.

Agenda

§ Problematic SQL & Situation

§ Response time solution

§ SQL costs solution

- Snapshot Monitoring
- Event Monitoring
- SQL Monitoring Interfaces
- Analyzing SQL
 - Explain Tools
 - Visual Explain
- Statement Concentrator

§ Making Performance Improvements

- Database Objects
- Better Coding
- Design Advisor
- DB2 Optim Query Workload Tuner
- Other Considerations

PRESENTATION NOTES FOR PAGE 64

Optim Query Workload Tuner

§ Optimize Performance and reduce costs

§ Identify performance hot spots

- Capture the hot spot query or workload from numerous sources

§ Diagnose and analyze semantics and plan

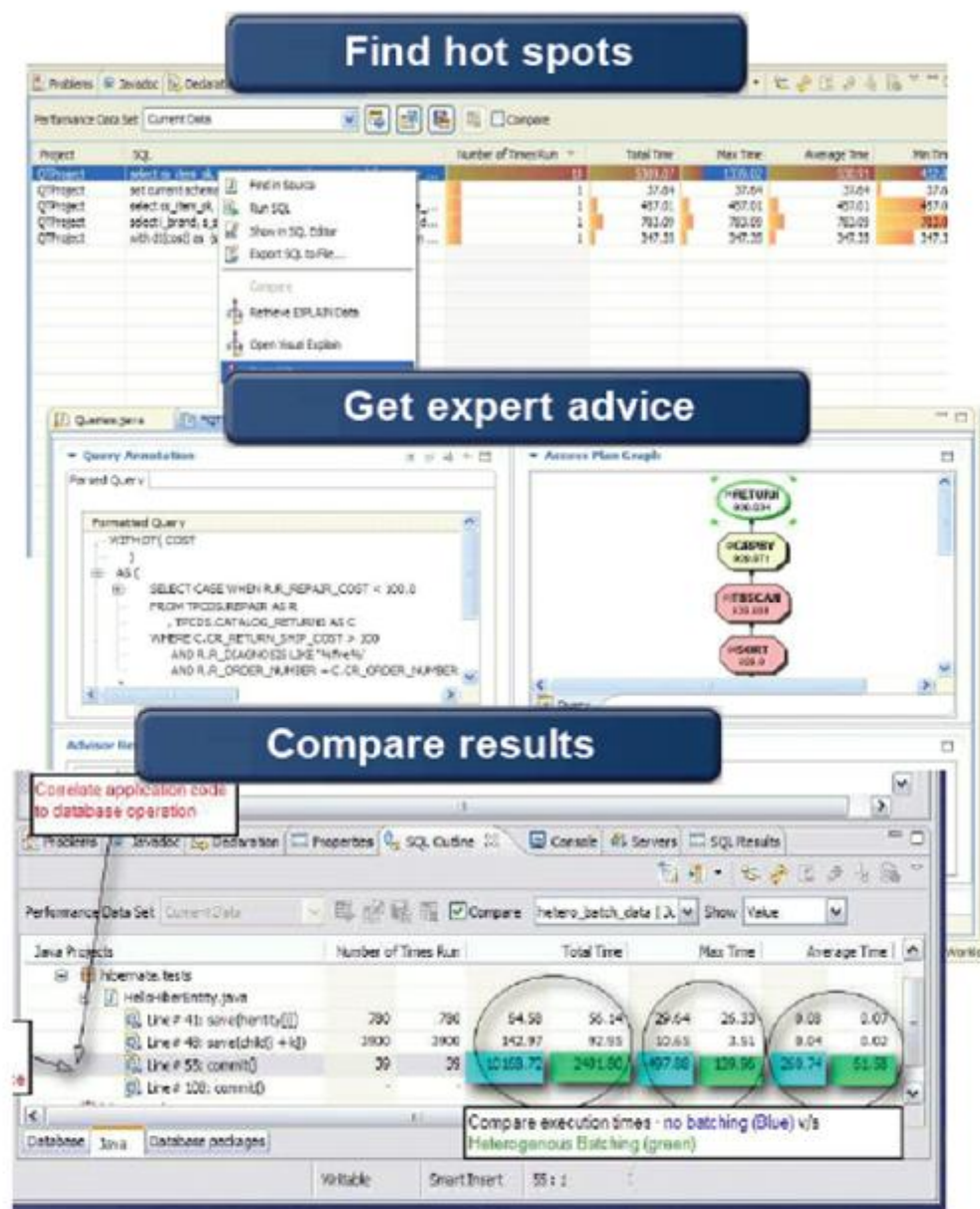
- Query formatting and annotation
- Access path visualization, annotation, comparison, and warnings
- Collect actual performance measures

§ Solve issues with expert tuning advice

- Improve query design
- Improve statistics quality
- Improve database design

§ Validate improvement and prevention

- Revise SQL
- Run application and RETUNE saved workloads
- Compare performance analysis and results

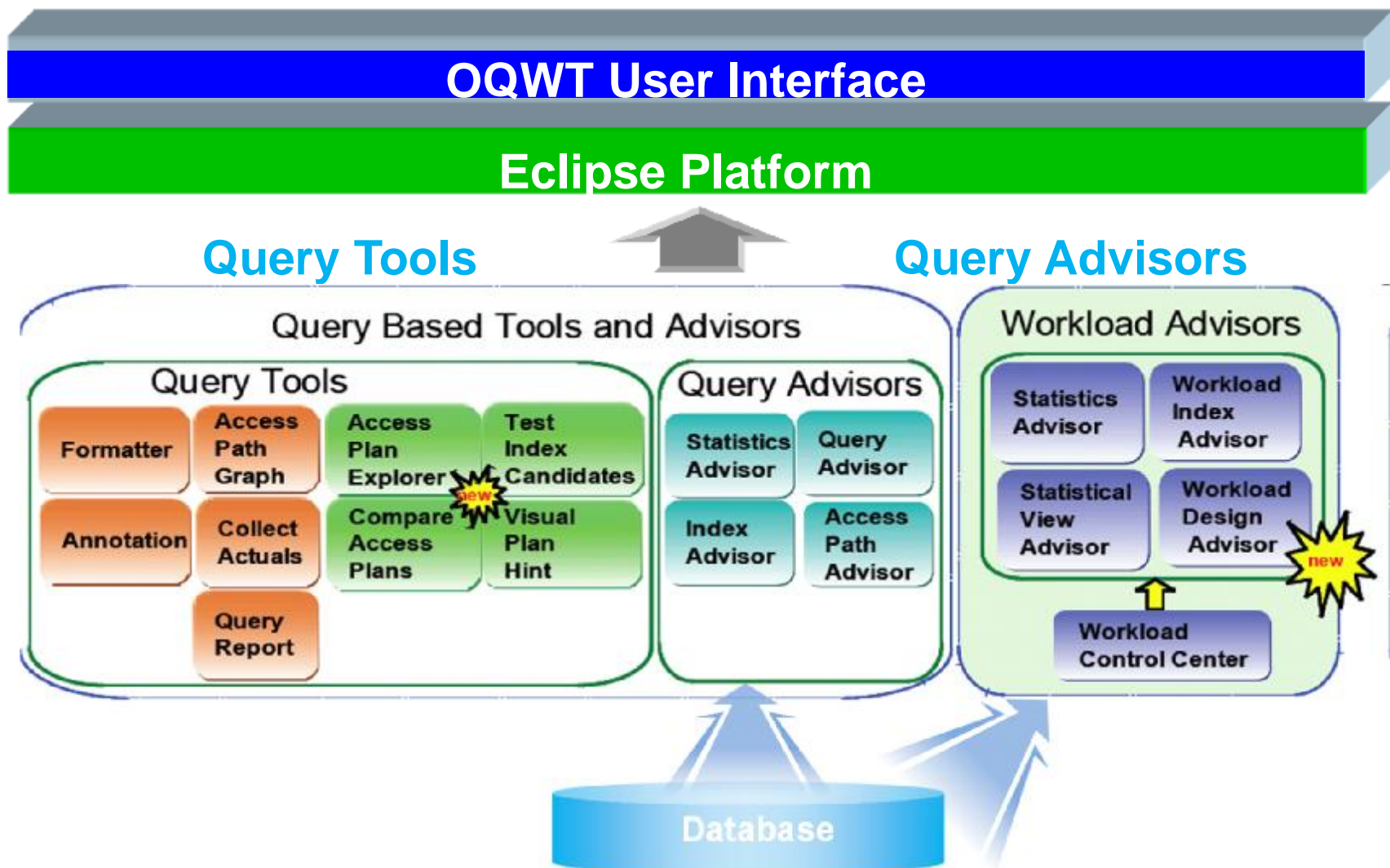


PRESENTATION NOTES FOR PAGE 65

Optim Query Tuner improves performance and cuts costs by providing expert advice on writing high quality queries and improving database design. Its easy-to-use advisors can help developers write high quality SQL code. Built-in integration with the Data Studio full client helps to find problematic queries in development. Database administrators can further tune performance by optimizing statistics and index creation.

Optim Query Workload Tuner

Functional Overview



PRESENTATION NOTES FOR PAGE 66

IBM Optim Query Tuner cuts cost and improves performance by providing expert advice on writing high quality queries and improving database design.

Its easy-to-use advisors enable developers to deliver optimized SQL right from the start.

Optimize SQL in the development process to improve performance while it is inexpensive and without waiting until production systems are impacted.

OQT can be launched right from the OPM Console, but it will open in Eclipse. If you are shell sharing with other Eclipse based tools, OQT will simply integrate with those tools and not open a separate client application.

OQWT - Query Tools

Format and Annotate your SQL

§ Formatter

- Provides a transformed SQL statement that is more readable to the naked eye
 - Original and transformed versions are both presented for easy reference and comparison
- Orders predicates in order of complexity
 - Or you can specify the order you like
- Has collapsible sections for easier navigation of complex queries

§ Annotator

- Provides catalog statistics and cost estimates, e.g.
 - “Qualified rows” estimates at the table level
 - Estimated filtering at the predicate level

Formatted Query

```

SELECT A.EMPNO
, A.FIRSTNAME
, A.LASTNAME
, A.JOB
, A.SALARY
, A.BONUS
, A.COMM
, B.LOCATION
, C.PROJNAME
FROM DSN8910.DEPT AS B
, DSN8910.EMP AS A
, DSN8910.EPROJ AS C
WHERE ( A.EMPNO IN ( SELECT DSN8910.DEPT.MGRNO
FROM DSN8910.DEPT
WHERE DSN8910.DEPT.MGRNO IS NOT NULL
)
AND A.WORKDEPT = B.DEPTNO
AND B.DEPTNO = C.DEPTNO

```

Annotation

CARDF=14	QUALIFIED_ROWS=
CARDF=42	QUALIFIED_ROWS=
CARDF=(missing)	QUALIFIED_ROWS=
COLCARDF=42	MAX_FREQ=(
CARDF=14	QUALIFIED_ROWS=
COLCARDF=9	MAX_FREQ=42
COLCARDF=8/14	MAX_FREQ=
COLCARDF=14/(missing)	MAX_FREQ=

Format

Annotate

Easily see tables, predicates, joins, etc

Collapse sections

View table stats and more

PRESENTATION NOTES FOR PAGE 67

Optim Query Tuner solutions make it easy to visualize queries. Depending on the source of the query, you might find one large block of text. Query Tuner automatically formats the queries providing a good starting point for analysis. In the formatted query, each table reference, each column reference under the SELECT clause, and each predicate, is shown on its own line. Formatting alone can save hours of DBA time. Now it is easy to see how many and which tables are accessed in the query, what joins are performed, etc. You can expand and collapse sections of complex queries, such as query blocks and sub-queries, to see an overview of the query and drill into parts of the query in more detail. When you click any line in the formatted query, other lines of the query that contain column or table references from the same table are also highlighted. You can also customize the formatting/ordering the predicates according to various criteria such as local predicates or join predicates, table references, and highest filter factor.

Query Tuner also adds annotations of the relevant statistics for predicates and table references such as cardinality and qualified rows. It automatically traces back through views handling the translations behind nested views. Plus it adds cost estimates and additional information such as data skew or default values.

This way, DBAs can accelerate analysis and reduce downtime for urgent situations. They can

Easily spot human errors like a missing predicate or missing comma

Identify where filtering should occur based on filter factors

Determine where the optimizer is using filtering

Spot transformations which will occur

Examine the formatted query text, and click the Transformed tab. When you compare the transformed query to the original query, you can see that the optimizer creates a virtual table to process the transformed correlated sub-query and that the EMPLOYEE and DEPARTMENT tables are joined inside the sub-query before the outer query is processed. Notice the following ways that you can interact with the formatted query:

You can collapse the SELECT clause.

You can select one row in the query text to quickly identify related parts of the query. For example, when you click the row that contains, FROM DSN8910.DEPT, all parts of the query that are related to the DEPT table are also highlighted.

You can see information about skewed data for two of the predicates in the Additional Information column.

OQWT - Query Tools

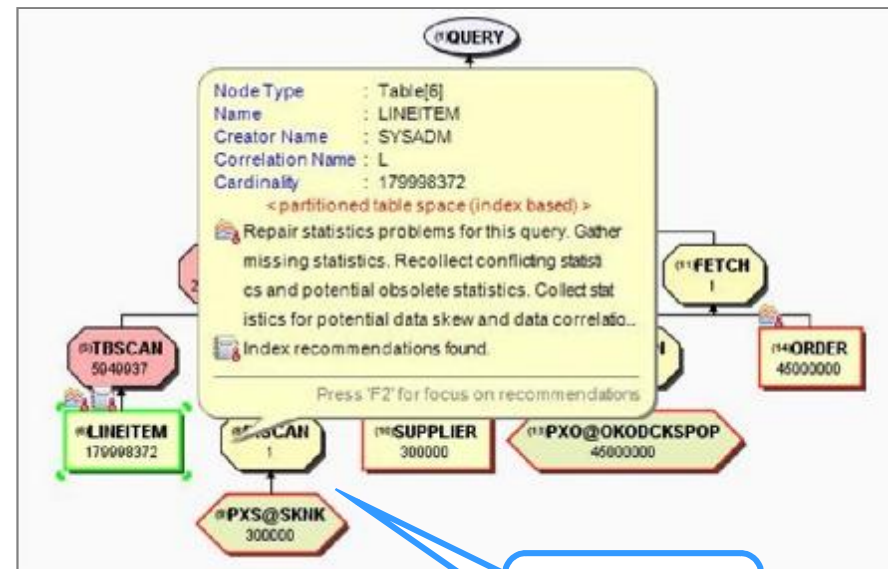
Access Path Graph and Query Report

§ Access Path Graph

- Visual explain with extra statistic and costing information - also has:
 - Built-in statistics advisor
 - Built-in index advisor
- Compare access plan graphs between two statements or two tuning analysis results side-by-side
 - Helpful when changing SQL and wanting to see both before and after visually

§ Query Report

- Creates table, index and predicate reports in:
 - HTML
 - Text



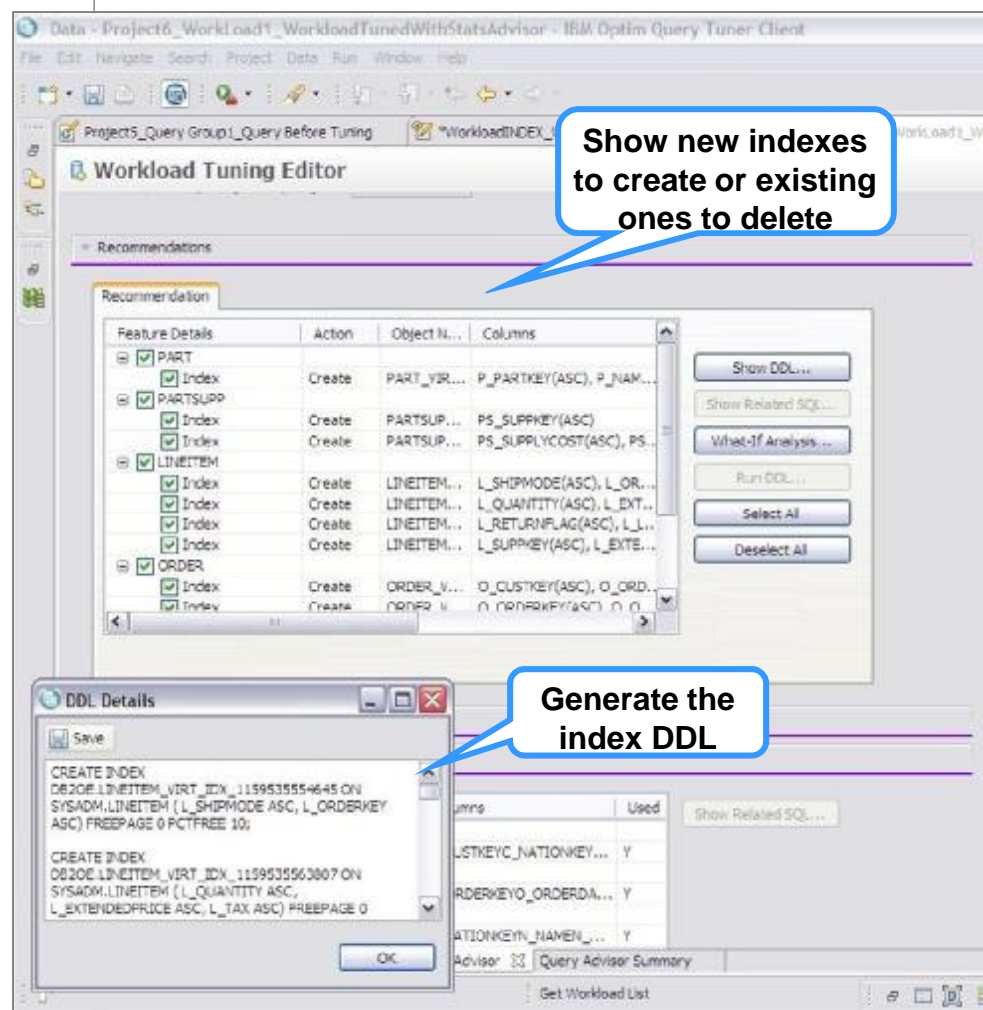
PRESENTATION NOTES FOR PAGE 68

A user can maintain multiple versions of a query (allowing them to make changes based on advice) within one project. When they invoke any of the advisors, multiple versions of the advice are maintained within the project.

OQWT - Query Advisors: Index Advisor

§ Index Advisor

- Provides expert advice to recommend new indexes for:
 - Foreign Keys in queries that do not have indexes
 - Identifying index filtering and screening
 - Supporting index-only access
 - Avoiding sorts
- Simplifies or consolidates current indexes
 - Can provide a single index recommendation for multiple indexes when possible
- Recommends to drop indexes not needed
- Uses virtual index capabilities built into DB2



PRESENTATION NOTES FOR PAGE 69

Index Advisor recommends indexes that might improve performance.

Uses the virtual index features of the engine.

A user can define their own virtual index.

Uses single SQL statement

Index Advisor has some advantages over the free DB2 Design Advisor:

DB2 LUW Design Advisor doesn't consolidate any duplicate or subset-superset index candidates for SQL, it will recommend what are recommended by the optimizer. For example:

Design advisor could recommend candidates:

IX1("L_ORDERKEY" ASC)

IX2("L_ORDERKEY" ASC, "L_EXTENDEDPRICE" ASC)

IX3("L_ORDERKEY" ASC, "L_EXTENDEDPRICE" ASC,
"L_QUANTITY" ASC)

In v2.2, OQT Index Advisor for LUW will consolidate them and only give recommendation of IX3.

OQWT - Query Advisors

Query Advisor

§ Query Advisor

- Provides advice on rewriting a query to make it more efficient
 - Based on best practice rules
 - Finds common mistakes
- Click on recommendation to highlight the SQL statement
 - Add missing join predicates for referential integrity
 - Use local predicates for joining tables or single table
 - Rewrite predicates as indexable predicates
 - Might use a matching index scan which is a more efficient access path
- Show or hide any recommendation individually or by scope

The screenshot displays the 'Query Recommendation Detail' window. The top section, 'SQL Text', shows a query with columns A.EMPNO, A.FIRSTNAME, A.LASTNAME, A.JOB, A.SALARY, A.BONUS, A.COMM, B.LOCATION, and C.PROJNAME. The FROM clause lists DSN8910.DEPT AS B, DSN8910.EMP AS A, and DSN8910.EMPJO AS C. The WHERE clause includes a subquery: WHERE (A.EMPNO IN (SELECT DSN8910.DEPT.MGRNO FROM DSN8910.DEPT WHERE DSN8910.DEPT.MGRNO IS NOT NULL)). A callout bubble points to the FROM clause with the text 'Highlight the relevant components'.

The bottom section, 'Selected Recommendations', contains a table with two columns: 'Description' and 'Recommendation'. The first recommendation describes a warning about a potentially costly Cartesian join between DSN8910.EMP and DSN8910.DEPT. A callout bubble points to this recommendation with the text 'Get recommendations and explanations'. The second recommendation shows a rewritten SQL statement for tables T1 and T2, adding join predicates T1.C2 = :charHV3 and T2.C2 = :charHV4.

PRESENTATION NOTES FOR PAGE 70

The Query Advisor evaluates how your query is written against a set of rules and best-practices to look for structures in the query that are likely causing the optimizer to choose a sub-optimal access path. You want a query that is going to"

Minimize the number of index pages and data rows that have to be read.

Minimize sort operations.

Utilize materialized query tables.

Often changing the structure of the query can help the optimizer find better access paths, e.g. adding a predicate that can be resolved by an index, adding predicates that include additional columns to increase filtering before data access, re-ordering predicates so that it is apparent that an MQT is usable. When the TARGETJOIN_LOCAL_PREDICATES parameter is set to OFF, all non-join predicates are processed after the join operation. When this parameter is set to ON, non-join predicates can be "pushed down" to the join phase of the query plan.

Specifically, the Query Advisor checks for:

Missing join predicates, but only if a foreign key is defined (see note above)

Stage 2 predicates that can improve performance if rewritten as Stage 1 or indexable

Stage 1 predicates that can improve performance if rewritten as indexable

Additional local predicates that are not automatically provided by DB2 that can provide Predicate Transitive Closure

Predicates that are pushed down to a nested table expression or materialized query tables without changing the result, and not already done automatically by DB2

Additional predicates added to a complex WHERE clause, containing ORs and ANDs and parentheses, this might improve performance without changing the result

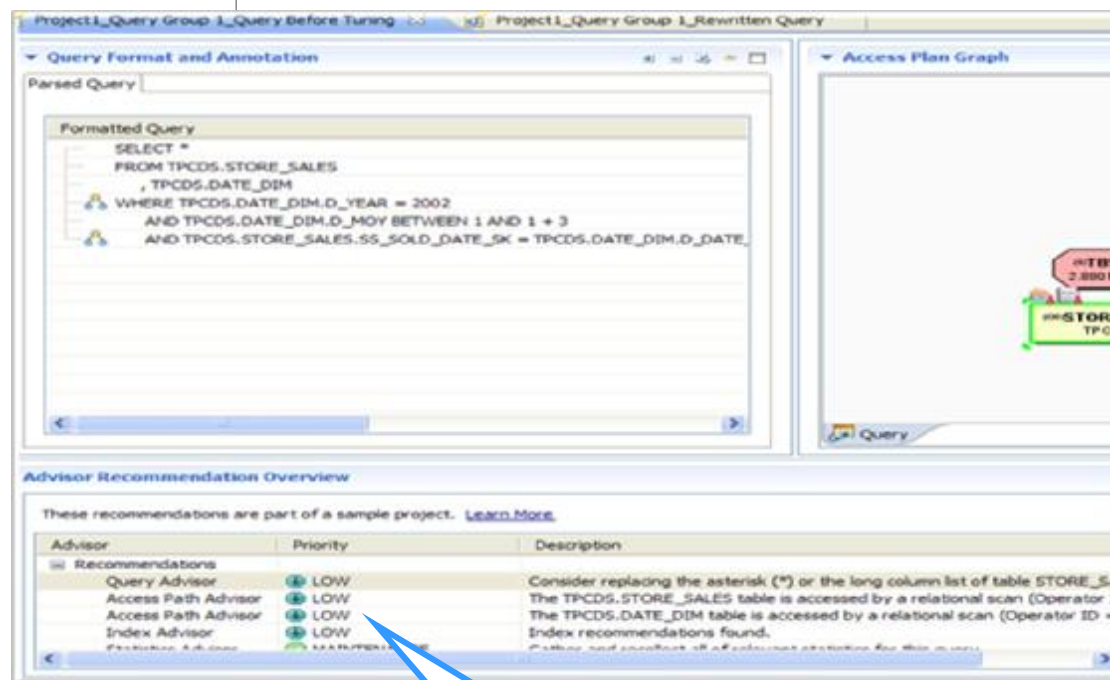
The use of SELECT * to replace a specific column list

OQWT - Query Advisors

Access Path Advisor

§ Access Path Advisor

- Identifies potential access path issues from the catalog
 - Recommendations based on best practice rules
- Presents a selection of access path possible problems giving:
 - A warning
 - An explanation
 - A description of issues found
 - The plan table information
- Ranks recommendations by severity: Low, Medium, High and Disaster
- Shows or hides any recommendation individually or by scope



View
recommendations
ranked in order of
severity

PRESENTATION NOTES FOR PAGE 71

This advisor identifies potential access path issues from the catalog. Recommendations are based on best practice rules.

The advisor interrogates the plan table, highlights potential access path performance issues and provides recommendations on how to eliminate such issues.

An explanation and description of issues are given to the user, and the recommendations are ranked by severity as Low, Medium, High and Disaster. When you select one of the warnings, a description and explanation for that warning are displayed, as well as the plan table information.

OQWT - Query Advisors

Statistics Advisor

§ Statistics Advisor

- Provides advice on:
 - Missing statistics
 - Problematic statistics
 - Out-of-date conflicts
 - Influential obsolete objects
- Saves implemented RUNSTATS to the server profile table for later use
- Interacts with the auto-stats feature of DB2 for LUW
- Gives you the following results:
 - Lower CPU consumption
 - Superior access plan selection
 - Improved maintenance window throughput
 - Accurate estimated cost of the query

Statistics Recommendation Detail

▼ Recommendations

RUNSTATS Control Statements

```
RUNSTATS INDEX(SYSADM.PXS@SKNK FREQVAL NUMCOLS 1 COUNT 15)
SHRLEVEL CHANGE REPORT YES UPDATE ALL HISTORY NONE
```

Generate RUNSTATS commands

RUNSTATS TABLESPACE DB4LINE1.TSLINE1
 TABLE(SYSADM.LINEITEM) SAMPLE 40
 COLUMN(L_SHIPMODE)
 COLGROUP(L_SHIPDATE) HISTOGRAM NUMQUANTILES 25
 COLGROUP(L_RECEIPTDATE) FREQVAL COUNT 15 HISTOGRAM NUMQUANTILES 25
 COLGROUP(L_SHIPMODE) FREQVAL COUNT 15
 COLGROUP(L_RETURNFLAG) FREQVAL COUNT 15
 SORTDEV SYSDA SORTNUM 4
 INDEX(SYSADM.SXL@PKSKOKEPOSQN KEYCARD,

▼ Table, index, column, and column group details

Canvas

O_ORDERKEY
 Cardinality: 4.5E7
 Uniform statistics collection time: 0001-01-01 00:00:00.0
 Uniform statistics status: conflicting
 Frequency statistics collection time: 2001-10-04 13:01:28.076475
 Frequency statistics status: conflicting
 Histogram statistics collection time: null
 Histogram statistics status: missing
 Possibly point skewed: No
 Possibly range skewed: No

O_ORDERPRIORITY
 Cardinality: 5.0
 Uniform statistics collection time: 0001-01-01 00:00:00.0
 Uniform statistics status: OK
 Frequency statistics collection time: null
 Frequency statistics status: missing
 Histogram statistics collection time: null
 Histogram statistics status: missing
 Possibly point skewed: Yes
 Possibly range skewed: No

Show missing or conflicting statistics

▼ Conflicts detail

TABLE SYSADM.LINEITEM
 One of the frequency records (-1.0) of the L_ORDERKEY column group is out of range [0,1]
 Tolerance: 0.0010

The maximum frequency of the column group or column (L_ORDERKEY), (0,0), is less than the average frequency, or 1 divided by the greater than the average unless only least-frequently occurring values are being collected.
 Tolerance: 0.0010

Describe the conflicts in detail

PRESENTATION NOTES FOR PAGE 72

Recommends the right statistics that should be gathered to improve query performance and lower CPU consumption. Recommends statistics that you should update or collect to improve the performance of a particularly hot query, and generates control statements that you can use to collect and update needed statistics with the RUNSTATS utility. Allows you to invoke the RUNSTATS utility directly from your workstation. The generated RUNSTATS jobs capture statistics more detailed than those from a RUNSTATS ALL job.

The DB2 Optimizer is a cost-based optimizer which uses catalog statistics to determine the most efficient access plan for any given query.

Catalog statistics such as size of database, tables, indexes, column groups and statistical views

Data distribution information for specific columns of tables, indexes and statistical views if the columns are used to select rows or join tables

Obsolete or lack of statistics for relevant objects can lead the optimizer to select a plan that is not optimal thereby slowing down query execution

Advises whether gathering additional statistics could result in a better access path – recommendations:

Identify interesting objects – those which influence optimization

Missing statistics – optimizer assumes the default values to determine costs

Conflict statistics – access path statistics for a single table + its indexes and columns and column groups

Obsolete statistics:

Empty table may be considered obsolete

Always recommended for RUNSTATS REPAIR

Statistics gathered a long time ago

While auto-stats feature is available in DB2 LUW, Optim Query Tuner Stats Advisor can be used against databases that have auto-stats turned off.

When auto-stats is off, what statistics should DB2 collect? Auto-stats DB2 LUW relies on end user or DBA to create appropriate profile for each table to collect statistics on relevant table objects (columns, indexes etc) and the level of detail. In absence of user-defined profile, DEFAULT profile is used which potentially can have huge overhead.

The Optim Query Tuner Stats Advisor can be used to create appropriate Stats Profile leveraging the server side 'auto-stats' capability (which can be throttled).

Summary Report

§ HTML report of the single query tuning analysis

§ Collaborative effort among team members and across teams

IBM Query Tuner Report

This report contains a summary of the recommendations from the Query Tuner advisors and tools. Examine the recommendations and corresponding DDL scripts, if applicable, and take appropriate actions to tune your query. You can also examine the formatted query and access plan summary, and cross reference the recommendations generated by the advisors. Use the table, column, and index information to do further analysis and tuning. Navigate to the different sections using the action buttons and then return to the top of the report using [Back to top](#).

[Recommended Action](#)
[View Query](#)
[Access Plan](#)
[DB Catalog Info](#)
[Save Report...](#)

Overview

Recommendation generation timestamp: 2011-06-26 15:30:53

Database server configuration: jdbc:db2://localhost:50000/TPCDS1G (SQL09073)

Estimated plan cost: 502.280 units

Critical problems: 1 statistics recommendations, 0 index recommendations, 0 query recommendations, and 0 access path warnings.

Best practices: 0 statistics recommendations, 1 index recommendations, 2 query recommendations, and 1 access path warnings.

Advice Number	Advice Type	Tuning Recommendation Description
1	Statistics	CRITICAL: Consolidate statistics. Use RUNSTATS to recollect all the relevant statistics for this query for an accurate evaluation. Important: If statistics are missing, Query tuner estimates subsequent recommendations based on database default statistics. Click here to review the recommended RUNSTATS script.
2	Index	Index recommendation is found for the following tables: TPCDS.CATALOG_RETURNS. The total estimated disk space required is 0.839 MB and the estimated performance improvement is 47.410%. This index recommendation will help improve query performance for general indexing. Click here to review the recommended CREATE INDEX DDL script.
3	Query	Provide a predicate on column R_ORDER_NUMBER. Consider adding the following predicate to column R_ORDER_NUMBER in table REPAIR: R_ORDER_NUMBER IS NOT NULL to filter the table earlier and unlock more possible join sequences. Click here to see the affected query text.
4	Query	Provide a predicate on column R_ITEM_SK. Consider adding the following predicate to column R_ITEM_SK in table REPAIR: R_ITEM_SK IS NOT NULL to filter the table earlier and unlock more possible join sequences. Click here to see the affected query text.
5	Access path	Avoid reading all index keys on an index scan (Operator ID = 9). The table TPCDS.REPAIR is accessed by a non-matching index scan (Operator ID = 9). If a table is accessed by non-matching index scan, then all the index keys and their RIDs are read. When a large number of keys and RIDs are accessed, DB2 might be using an inefficient access path. Consider run the statistics advisor or run the index advisor to determine whether creating an index might improve the access path. Click here to see the access plan operator.

PRESENTATION NOTES FOR PAGE 73

Value of OQWT tuning features

- § Provides a wide range of automated features that target improving query workload execution performance (**includes DB2 10 fast query performance features**)
- § Allows concentrating on multiple problem statements at the same time by tuning these as a workload
- § Provides advisors that target the biggest need for tuning execution plans
- § Includes MANY features to allow for more detailed analysis and advanced tuning:
 - Analysis: Access plan graph, access plan explorer, collect actuals, compare plans, query annotation and formatting
 - Features: query revision advisor, access plan advisor, statistical view advisor, test candidate indexes, visual plan hint

PRESENTATION NOTES FOR PAGE 74

Overall benefit of OQWT to your enterprise

- § Targets and reduces performance hot spots across multiple database systems
- § Automates effort to make it easier and faster for DBAs, developers, designers, and others
- § Improves organizational productivity
 - Reduces time for users to respond to issues
 - Reduces the need for expertise in tuning
- § Tunes multiple workload statements together
- § Improves crucial statistics collection and database design efficiency
- § Improves customer satisfaction

PRESENTATION NOTES FOR PAGE 75

Agenda

§ Problematic SQL & Situation

§ Response time solution

§ SQL costs solution

- Snapshot Monitoring
- Event Monitoring
- SQL Monitoring Interfaces
- Analyzing SQL
 - Explain Tools
 - Visual Explain
- Statement Concentrator

§ Making Performance Improvements

- Database Objects
- Better Coding
- Design Advisor
- DB2 Optim Query Workload Tuner
- Other Considerations

PRESENTATION NOTES FOR PAGE 76

Other factors influencing query performance



NEW IN
DB2 10

- § Accurate database statistics – RUNSTATS
A number of improvements have been made to the RUNSTATS command to make statistics gathering faster in some cases. The command parameters have also been simplified
- § Defining [Informational] constraints
- § Use the REOPT bind option when host variable's values affect access plan
- § Using parameter markers to reduce statement compilation
- § Specifying Row Blocking for better cursor processing
 - Blocking All
 - Blocking No
 - Blocking Unambig
- § Data sampling for statistics
 - Row-level Bernoulli sampling
 - System page-level sampling

PRESENTATION NOTES FOR PAGE 77

Accurate database statistics are critical for query optimization.

Perform RUNSTATS regularly on any tables critical to query performance. You might also want to collect statistics on system catalog tables, if an application queries these tables directly and if there is significant catalog update activity such as DDL statements. Automatic statistics collection can be enabled to allow the DB2 data server to automatically perform RUNSTATS. Real time statistics collection can be enabled to allow the DB2 data server to provide even more timely statistics by collecting them immediately before queries are optimized.

NEW IN DB2 10: RUNSTATS and database statistics improvements:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.doc/c0058567.html>

Using constraints to improve query optimization

Consider defining unique, check, and referential integrity constraints. These constraints provide semantic information that allows the DB2® optimizer to rewrite queries to eliminate joins, push aggregation down through joins, push FETCH FIRST n ROWS DOWN through joins, remove unnecessary DISTINCT operations, and perform a number of other optimizations.

Informational constraints can also be used for both check constraints and referential integrity constraints when the application itself can guarantee the relationships. The same optimizations are possible. Constraints that are enforced by the database manager when rows are inserted, updated, or deleted can lead to high system overhead, especially when updating a large number of rows that have referential integrity constraints. If an application has already verified information before updating a row, it might be more efficient to use informational constraints, rather than regular constraints.

Using the REOPT bind option with input variables in complex queries

Input variables are essential for good statement preparation times in an online transaction processing (OLTP) environment, where statements tend to be simpler and query access plan selection is more straightforward.

Multiple executions of the same query with different input variable values can reuse the compiled access section in the dynamic statement cache, avoiding expensive SQL statement compilations whenever the input values change.

However, input variables can cause problems for complex query workloads, where query access plan selection is more complex and the optimizer needs more information to make good decisions. Moreover, statement compilation time is usually a small component of total execution time, and business intelligence (BI) queries, which do not tend to be repeated, do not benefit from the dynamic statement cache.

If input variables need to be used in a complex query workload, consider using the REOPT(ALWAYS) bind option. The REOPT bind option defers statement compilation from PREPARE to OPEN or EXECUTE time, when the input variable values are known. The values are passed to the SQL compiler so that the optimizer can use the values to compute a more accurate selectivity estimate. REOPT(ALWAYS) specifies that the statement should be recompiled for every execution. REOPT(ALWAYS) can also be used for complex queries that reference special registers, such as WHERE TRANS_DATE = CURRENT DATE - 30 DAYS, for example. If input variables lead to poor access plan selection for OLTP workloads, and REOPT(ALWAYS) results in excessive overhead due to statement compilation, consider using REOPT(ONCE) for selected queries. REOPT(ONCE) defers statement compilation until the first input variable value is bound. The SQL statement is compiled and optimized using this first input variable value. Subsequent executions of the statement with different values reuse the access section that was compiled on the basis of the first input value. This can be a good approach if the first input variable value is representative of subsequent values, and it provides a better query access plan than one that is based on default values when the input variable values are unknown.

Using parameter markers to reduce compilation time for dynamic queries

The DB2® data server can avoid recompiling a dynamic SQL statement that has been run previously by storing the access section and statement text in the dynamic statement cache.

A subsequent prepare request for this statement will attempt to find the access section in the dynamic statement cache, avoiding compilation. However, statements that differ only in the literals that are used in predicates will not match. For example, the following two statements are considered different in the dynamic statement cache:

SELECT AGE FROM EMPLOYEE WHERE EMP_ID = 26790
SELECT AGE FROM EMPLOYEE WHERE EMP_ID = 77543
Even relatively simple SQL statements can result in excessive system CPU usage due to statement compilation, if they are run very frequently. If your system experiences this type of performance problem, consider changing the application to use parameter markers to pass predicate values to the DB2 compiler, rather than explicitly including them in the SQL statement. However, the access plan might not be optimal for complex queries that use parameter markers in predicates. For more information, see "Using the REOPT bind option with input variables in complex queries".

Specifying row blocking to reduce overhead

Row blocking, which is supported for all statements and data types (including LOB data types), reduces database manager overhead for cursors by retrieving a block of rows in a single operation.

This block of rows represents a number of pages in memory. It is not a multidimensional (MDC) table block, which is physically mapped to an extent on disk.

Row blocking is specified by the following options on the BIND or PREP command:

BLOCKING ALL

Cursors that are declared with the FOR READ ONLY clause or that are not specified as FOR UPDATE will be blocked.

BLOCKING NO

Cursors will not be blocked.

BLOCKING UNAMBIG

Cursors that are declared with the FOR READ ONLY clause will be blocked. Cursors that are not declared with the FOR READ ONLY clause or the FOR UPDATE clause, that are not ambiguous, or that are read-only, will be blocked. Ambiguous cursors will not be blocked.

Before enabling the blocking of row data for LOB data types, it is important to understand the impact on system resources. More shared memory will be consumed on the server to store the references to LOB values in each block of data when LOB columns are returned. The number of such references will vary according to the value of the rrioblk configuration parameter.

Data sampling in queries

It is often impractical and sometimes unnecessary to access all of the data that is relevant to a query. In some cases, finding overall trends or patterns in a subset of the data will suffice. One way to do this is to run a query against a random sample from the database.

The DB2® product enables you to efficiently sample data for SQL and XQuery queries, potentially improving the performance of large queries by orders of magnitude, while maintaining a high degree of accuracy.

Sampling is commonly used for aggregate queries, such as AVG, COUNT, and SUM, where reasonably accurate values for the aggregates can be obtained from a sample of the data. Sampling can also be used to obtain a random subset of the rows in a table for auditing purposes or to speed up data mining and analysis.

Two methods of sampling are available: row-level sampling and page-level sampling.

Row-level Bernoulli sampling

Row-level Bernoulli sampling obtains a sample of P percent of the table rows by means of a SARGable predicate that includes each row in the sample with a probability of P/100 and excludes it with a probability of 1-P/100.

Row-level Bernoulli sampling always produces a valid, random sample regardless of the degree of data clustering. However, the performance of this type of sampling is very poor if no index is available, because every row must be retrieved and the sampling predicate must be applied to it. If there is no index, there are no I/O savings over executing the query without sampling. If an index is available, performance is improved, because the sampling predicate is applied to the RIDs inside of the index leaf pages. In the usual case, this requires one I/O per selected RID, and one I/O per index leaf page.

System page-level sampling

System page-level sampling is similar to row-level sampling, except that pages (not rows) are sampled. The probability of a page being included in the sample is P/100. If a page is included, all of the rows on that page are included.

The performance of system page-level sampling is excellent, because only one I/O is required for each page that is included in the sample. Compared with no sampling, page-level sampling improves performance by orders of magnitude. However, the accuracy of aggregate estimates tends to be worse under page-level sampling than row-level sampling. This difference is most pronounced when there are many rows per page, or when the columns that are referenced in the query exhibit a high degree of clustering within pages.

Specifying the sampling method

Use the TABLESAMPLE clause to execute a query against a random sample of data from a table. TABLESAMPLE BERNOULLI specifies that row-level Bernoulli sampling is to be performed. TABLESAMPLE SYSTEM specifies that system page-level sampling is to be performed, unless the optimizer determines that it is more efficient to perform row-level Bernoulli sampling instead.

If all tuning options fail

§ Use Optimization profiles

- Explicit Optimization guideline to DB2 optimizer
- XML document
- Define SQL statement
- Define optimization guideline
- No application or database configuration changes

§ Things to consider about Optimization profiles

- Requires effort to maintain
- For existing SQL statements only
- Optimizer still considers other possible access plans
- Optimizer ignores invalid or inapplicable guidelines



NEW IN
DB2 10

Optimization profile supports registry variables and inexact matching

PRESENTATION NOTES FOR PAGE 78

Optimization profiles and guidelines

An optimization profile is an XML document that can contain optimization guidelines for one or more SQL statements. The correspondence between each SQL statement and its associated optimization guidelines is established using the SQL text and other information that is needed to unambiguously identify an SQL statement.

The DB2® optimizer is one of the most sophisticated cost-based optimizers in the industry. However, in rare cases the optimizer might select a less than optimal execution plan. As a DBA familiar with the database, you can use utilities such as db2adviz, runstats, and db2expln, as well as the optimization class setting to help you tune the optimizer for better database performance. If you do not receive expected results after all tuning options have been exhausted, you can provide explicit optimization guidelines to the DB2 optimizer.

An explicit optimization guideline can be used to influence the optimizer. Optimization guidelines are specified using a simple XML specification. Each element within the OPTGUIDELINES element is interpreted as an optimization guideline by the DB2 optimizer.

Optimization profiles allow optimization guidelines to be provided to the optimizer without application or database configuration changes. You simply compose the simple XML document, insert it into the database, and specify the name of the optimization profile on the BIND or PRECOMPILE command. The optimizer automatically matches optimization guidelines to the appropriate statement.

Optimization guidelines do not need to be comprehensive, but should be targeted to a desired execution plan. The DB2 optimizer still considers other possible access plans using the existing cost-based methods. Optimization guidelines targeting specific table references cannot override general optimization settings. For example, an optimization guideline specifying the merge join between tables A and B is not valid at optimization class 0.

The optimizer ignores invalid or inapplicable optimization guidelines. If any optimization guidelines are ignored, an execution plan is created and SQL0437W with reason code 13 is returned. You can then use the EXPLAIN statement to get detailed diagnostic information regarding optimization guidelines processing.

NEW IN DB2 10: Optimization profile supports registry variables and inexact matching

The optimization profile can now be used to set certain registry variables and supports inexact matching. Inexact matching can be used for better matching when compiling query statements.

A subset of registry variables can be set in the optimization profile with the OPTION element in the REGISTRY element. The OPTION element has NAME and VALUE attributes where you specify the registry variable and its value. You can set many registry variables at the global level or, for specific statements, at the statement-level.

The optimization profile now supports inexact matching in addition to exact matching. Inexact matching ignores literals, host variables, and parameter markers when statements are being matched. To specify inexact matching in the optimization profile, set the EXACT attribute value of the STMTMATCH element to FALSE. You can specify the STMTMATCH element at both the global level or at the statement-level.

Summary

§ In this Module you learned about:

- Many performance improvements have been included in DB2 10 to improve the speed of many queries. These improvements are automatic; there are no configuration settings or changes to the SQL statements required!
- Use event monitors and Design Advisor for better response time results
- Use snapshot monitoring, event monitoring or SQL monitoring interfaces to identify SQL costs
- Visual Explain tools are a good way to review the query access plans
- Statement Concentrator as alternative to reduce SQL costs
- You can make performance improvements after database design and coding phase
- Optim Query Workload Tuner is a good tool when you need to make performance improvements

PRESENTATION NOTES FOR PAGE 79

The next steps...



PRESENTATION NOTES FOR PAGE 80

The Next Steps...

§ Complete the Hands on Lab for this module

- Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
- Find the module
- Download the workbook [10304_WB1_DB2_SQLQueryTuningwithBLUacceleration_II.pdf](#) and the virtual machine image [10300_VM1_DB2_PerformanceMonitoringAndTuning_10.5.part#.rar](#)
- Follow the instructions in the workbook to complete the lab

§ Complete the online quiz for this module

- Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
- Find the module and select the quiz

§ Provide feedback on the module

- Log onto SKI, go to “My Learning” page
- Find the module and select the “Leave Feedback” button to leave your comments



PRESENTATION NOTES FOR PAGE 81

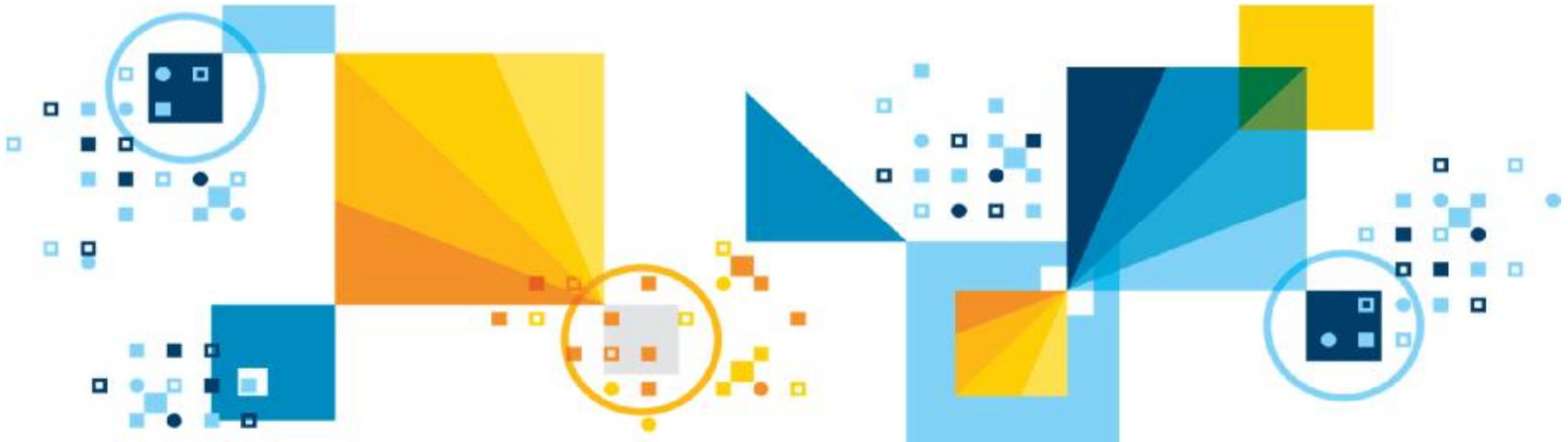
The Next Steps...

- § The next set of modules to consider :
 - Module DB2 Locking and Logging for Performance



PRESENTATION NOTES FOR PAGE 82

Questions?
askdata@ca.ibm.com



PRESENTATION NOTES FOR PAGE 83