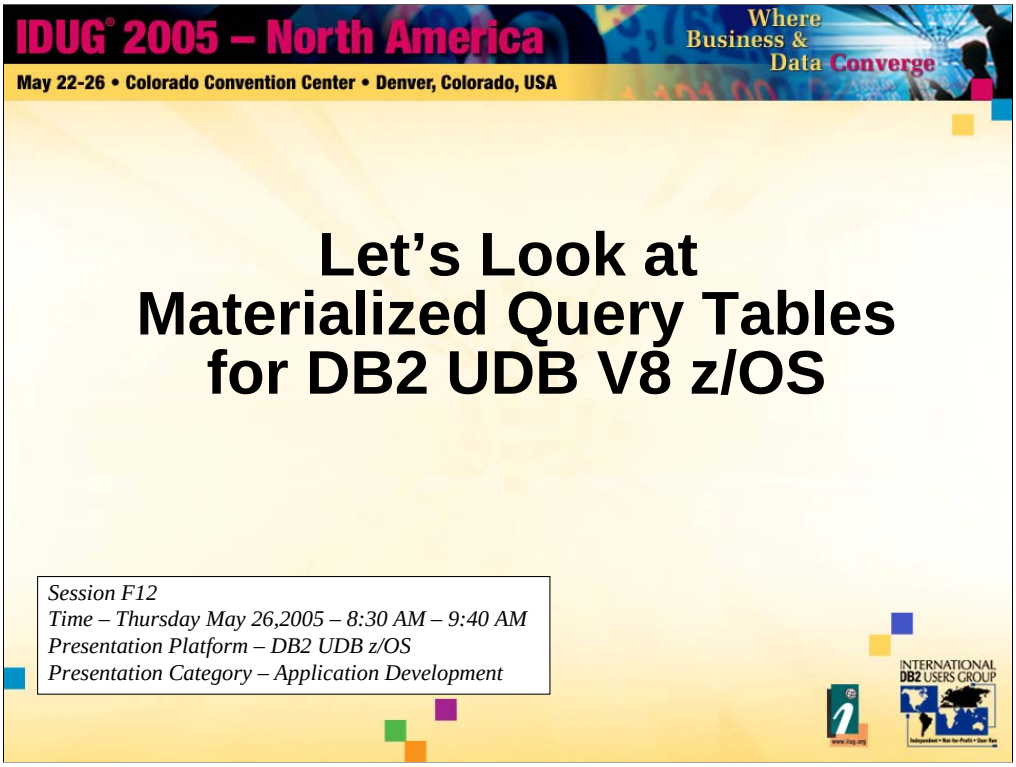


It is very common that as the amount of data grows in our DB2 tables and as our applications become more complex, queries must run against a large amount of data. Even though the amount of data has increased, there are still business requirements to deliver response times as quickly as possible. Materialized query tables are very effective in working with data warehousing applications. This presentation will discuss the basics of materialized query tables and give you an understanding of how you can best utilize this new functionality in DB2 V8.

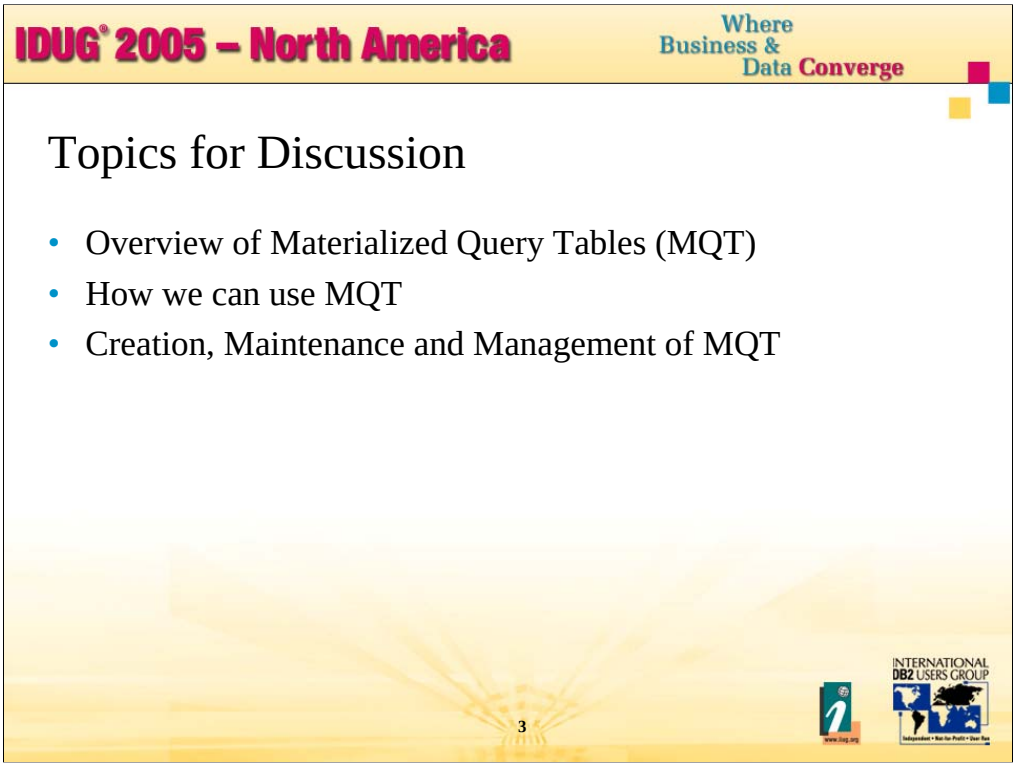
Who Are We?

IBM Software Group

IBM Software Services for Information Management

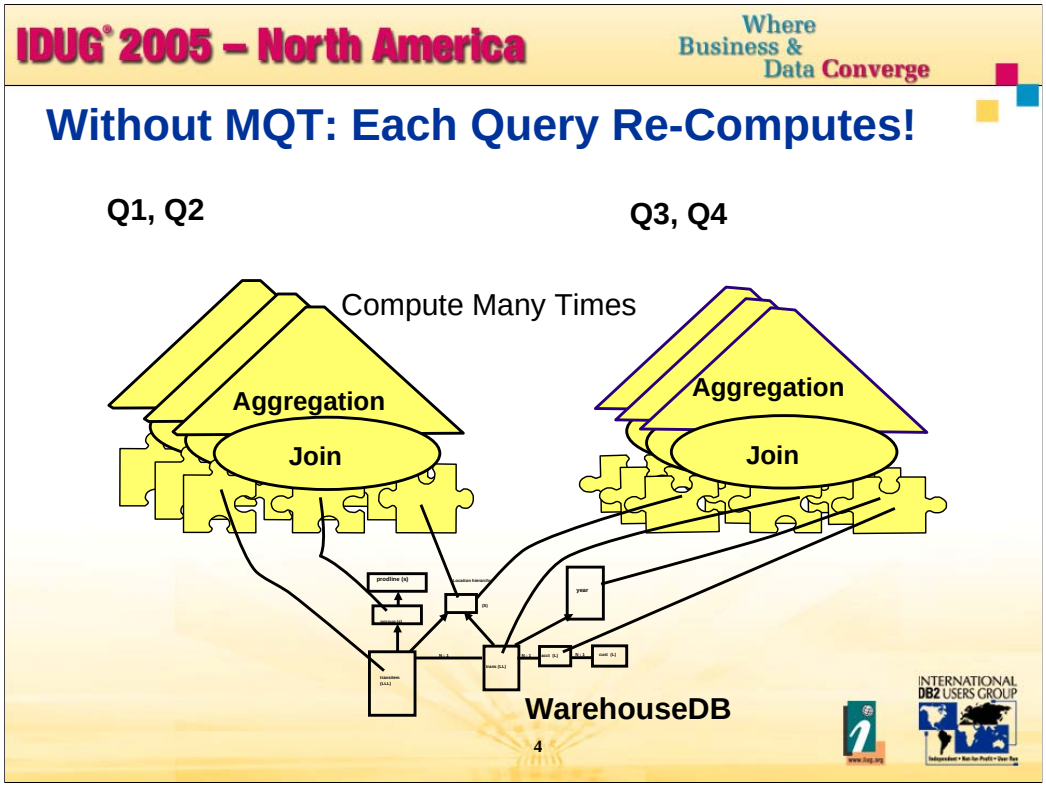
DB2 for z/OS Lab Services

The DB2 for z/OS and OS/390 Software Services Team delivers specialized services to accelerate your implementation and to fully leverage and exploit your investment in DB2 technology on these operating systems. Our consultants specialize in product installation and configuration and Version to Version migrations for DB2 for z/OS and OS/390 Server. They are highly skilled in system administration, including high availability, security, and health checks, and also provide expertise for utilities migration assessments and migration. We also have peer groups who specialize in DB2 on other platforms.

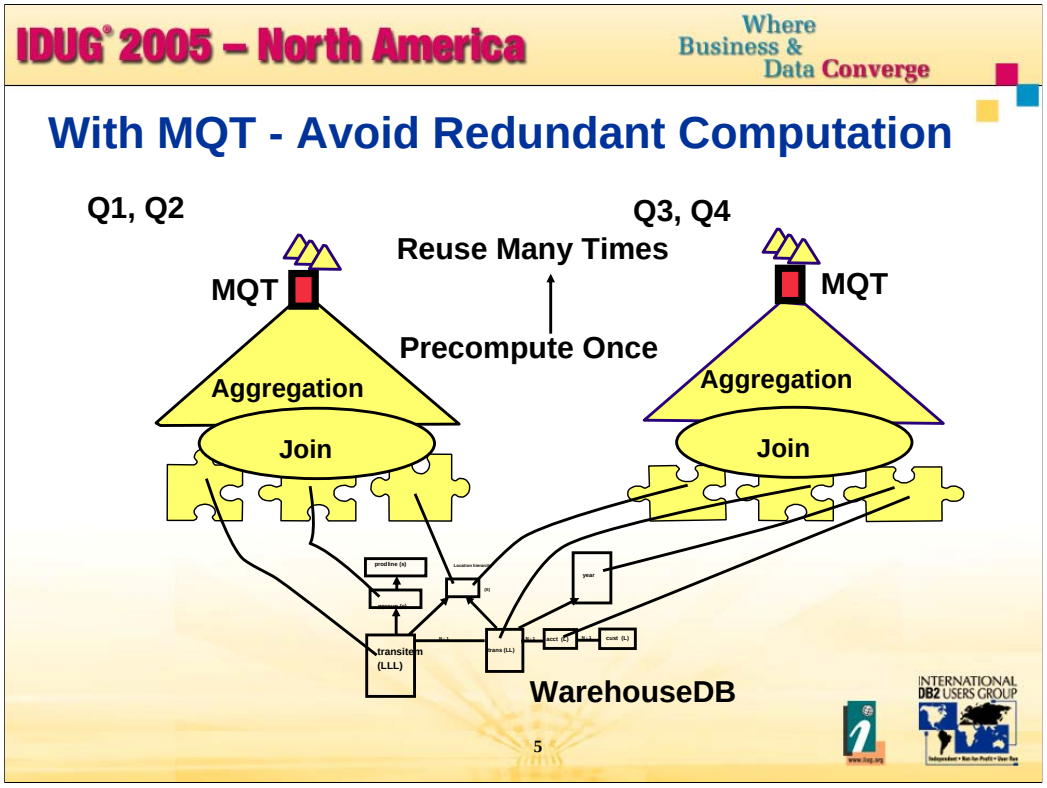


Materialized query tables are tables that contain information that is derived and summarized from other tables.

Automatic query rewrite is the process DB2 uses to access data in a materialized query table.



A materialized query table (MQT) can avoid redundant work of scanning, aggregation and joins. Multiple levels of summary tables have been used in warehouses and complex applications for years. One of the major issues is communicating the summaries to the users. In some cases, the users want to query the base data. With MQTs, the query users do not have to be aware of the MQT.



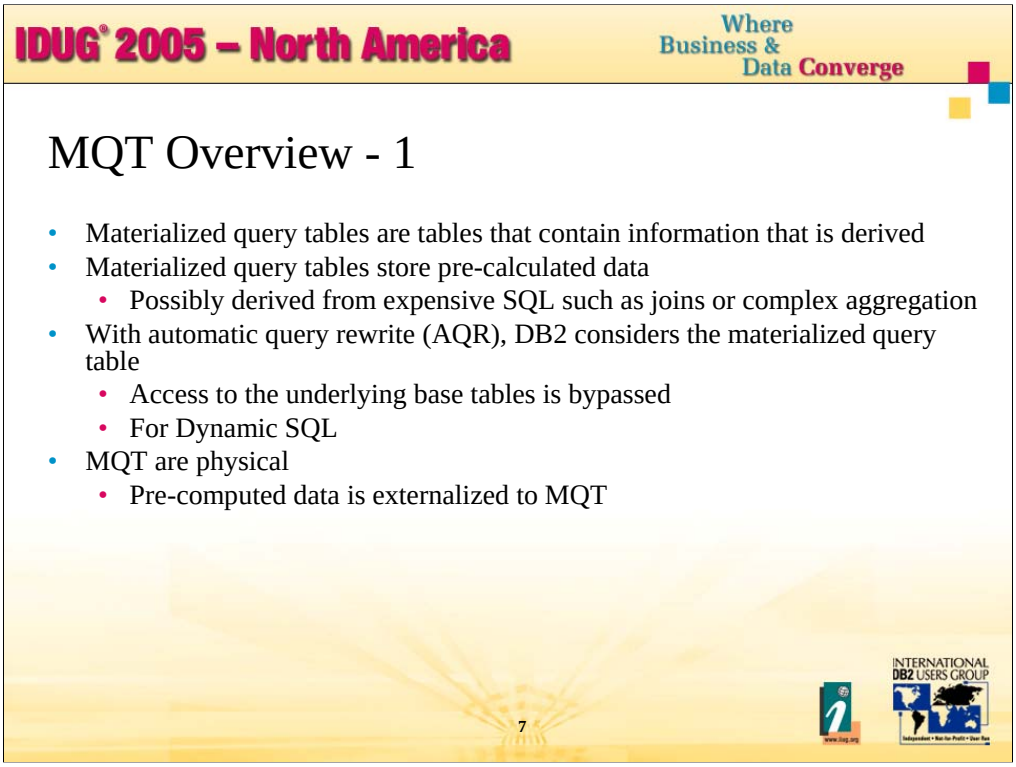
Even though the query is submitted for the base table, the optimizer can rewrite the query to use the MQT. Using the precalculated information can improve subsequent queries by as much as two or three orders of magnitude. Materialization or precalculation and parallelism resolve the long response times.

A database administrator can use an MQT much as she or he would use an index for optimization. Controls for usage, initial loading and refresh are part of the definition.

Why do we need MQT

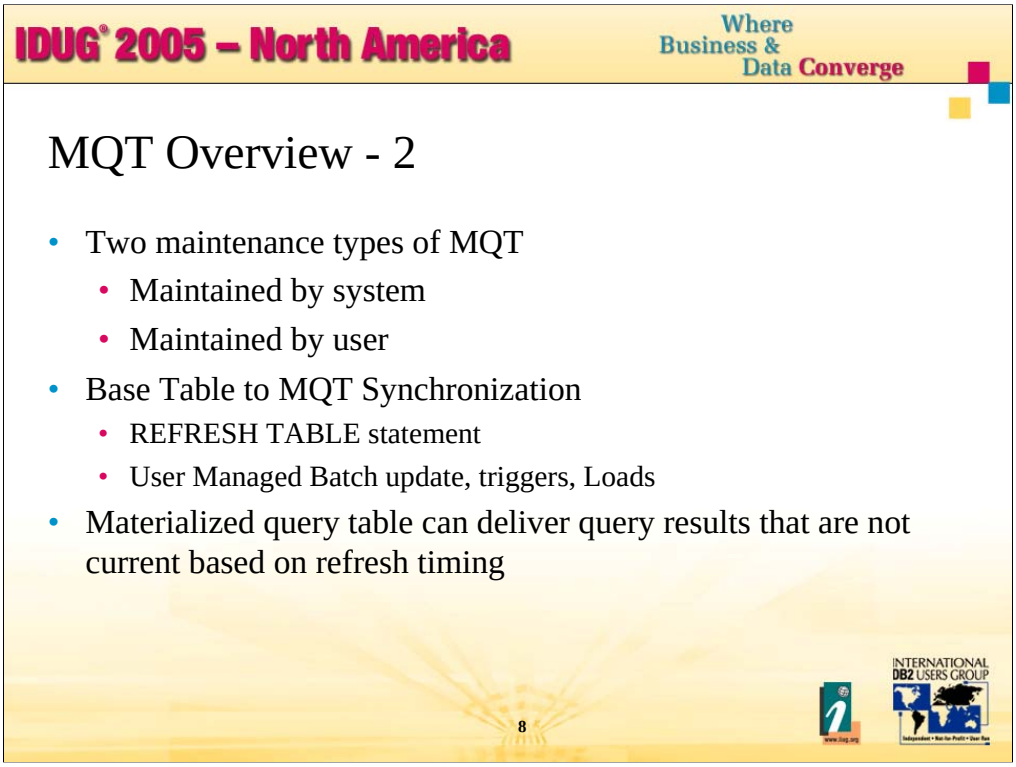
- Answer to many questions are against a large amount of data
- Summary queries are required
- Performance of this query is important
  - But 100 % accuracy may not be needed
  - Or may be needed….

Materialized Query tables are sometimes referred to Automatic Summary tables



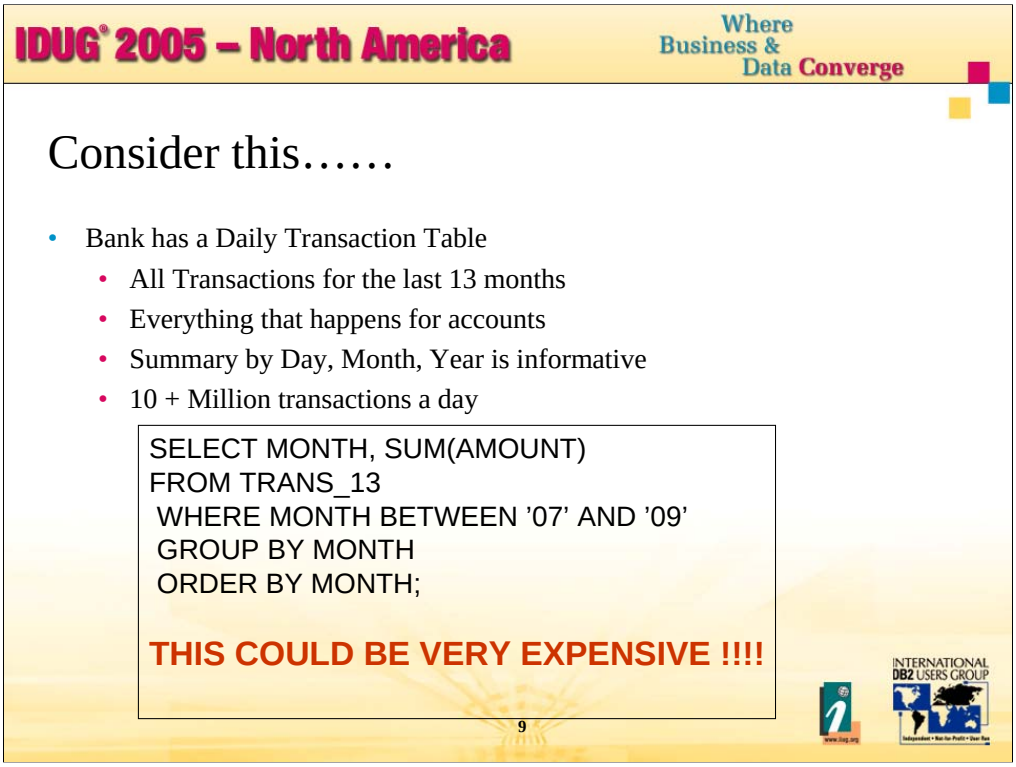
Materialized query tables are particularly effective in data warehousing applications.

materialized query tables can simplify query processing and greatly improve the performance of dynamic SQL queries.



Because databases have grown substantially over the years, queries must operate over huge amounts of data.

For example, in a data warehouse environment, decision-support queries might need to operate over 1 to 10 terabytes of data, performing multiple joins and other complex SQL



It is difficult to provide a performance improvement number as it is very dependent on the summarization work that would need to be done on the base tables. Lab tests are seeing potential 100 times return with the use of MQT.

But Consider This….

- With A MQT in place, results could be retrieved more efficiently

```
CREATE TABLE TRANS_13_SUMM AS
 (SELECT YEAR AS SYEAR,
 MONTH AS SMONTH,  DAY AS SDAY,  SUM(AMOUNT) AS SSUM
 FROM TRANS_13
 GROUP BY YEAR, MONTH, DAY)
DATA INITIALLY DEFERRED REFRESH DEFERRED
MAINTAINED BY SYSTEM DISABLE QUERY OPTIMIZATION;

REFRESH TABLE TRANS_13_SUMM;

ALTER TABLE TRANS_13_SUMM AS…….
DATA INITIALLY DEFERRED REFRESH DEFERRED
MAINTAINED BY SYSTEM ENABLE QUERY OPTIMIZATION;
```


Make note of the MQT name and refresh table step.

So SQL retrieves the data requested

- We "executed" this …..

```
SELECT MONTH, SUM(AMOUNT)
FROM TRANS_13
 WHERE MONTH BETWEEN '07' AND '09'
 GROUP BY MONTH
 ORDER BY MONTH;
```

- But this is the SQL that is really executed with AQR in place…

```
SELECT SMONTH, SSUM(AMOUNT)
FROM TRANS_13_SUMM
 WHERE SMONTH BETWEEN '07' AND '09'
 GROUP BY SMONTH
 ORDER BY SMONTH;
```


Returning data from the MQT could be decided by AQR or data could be returned via direct access to the MQT.

MQT Design - 1

- Understand the present issues
- Evaluate the effectiveness of MQT
- How does the process work today
  - Running against base table, query takes 45 minutes
- How would it work with MQT in place
  - It takes 60 minutes to complete the refresh of the MQT
  - Query against MQT takes 2 minutes

MQT exploitation could also allow for different design of base tables.

MQT Design - 2

- Grouping levels are key
- How to determine MQT Quantity
  - Specifically defined
  - Generically defined
  - Need to consider return on performance vs. cost of managing

Evaluation of the requirements of the requested information is key. Reviewing current SQL execution could also assist in determining ways to use MQT for existing applications.

CREATE MQT EXAMPLE

```
CREATE TABLE MQT_TABLE AS

( SELECT T.PDATE, T.TRANSID,
SUM(QTY * PRICE) AS TOTVAL,
COUNT(QTY * PRICE) AS CNT
FROM SCNDSTAR.TRANSITEM TI, SCNDSTAR.TRANS T
WHERE TI.TRANSID = T.TRANSID
GROUP BY T.PDATE, T.TRANSID)

DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY SYSTEM
ENABLE QUERY OPTIMIZATION
IN MQTDB01.MQTTS01;
```


MQT is in SYSTABLES with a TYPE = ‘M’

MQT Definition - 1

- On CREATE or ALTER TABLE DDL
  - With FULLSELECT
  - DATA INITIALLY DEFERRED - REFRESHED DEFERRED
    - MQT Definition requirement
  - MAINTAINED MY SYSTEM
    - Updated via REFRESH TABLE ONLY
  - MAINTAINED BY USER
    - Can be updated directly

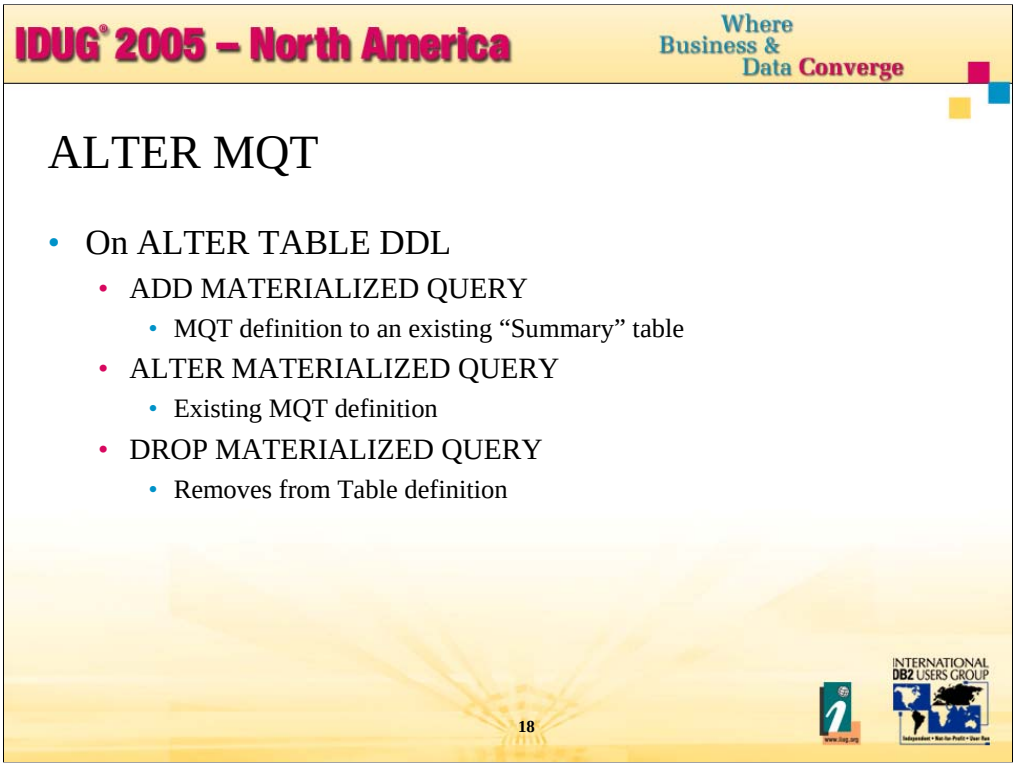
More on REFRESH TABLE later…..

MQT Definition - 2

- On CREATE or ALTER TABLE DDL
  - ENABLE QUERY OPTIMIZATION
    - Provides an option for where data is retrieved
  - DISABLE QUERY OPTIMIZATION
    - Access MQT directly
    - Or wait until REFRESH TABLE has been done
  - Consider initial definition
    - Timing of initial population of data

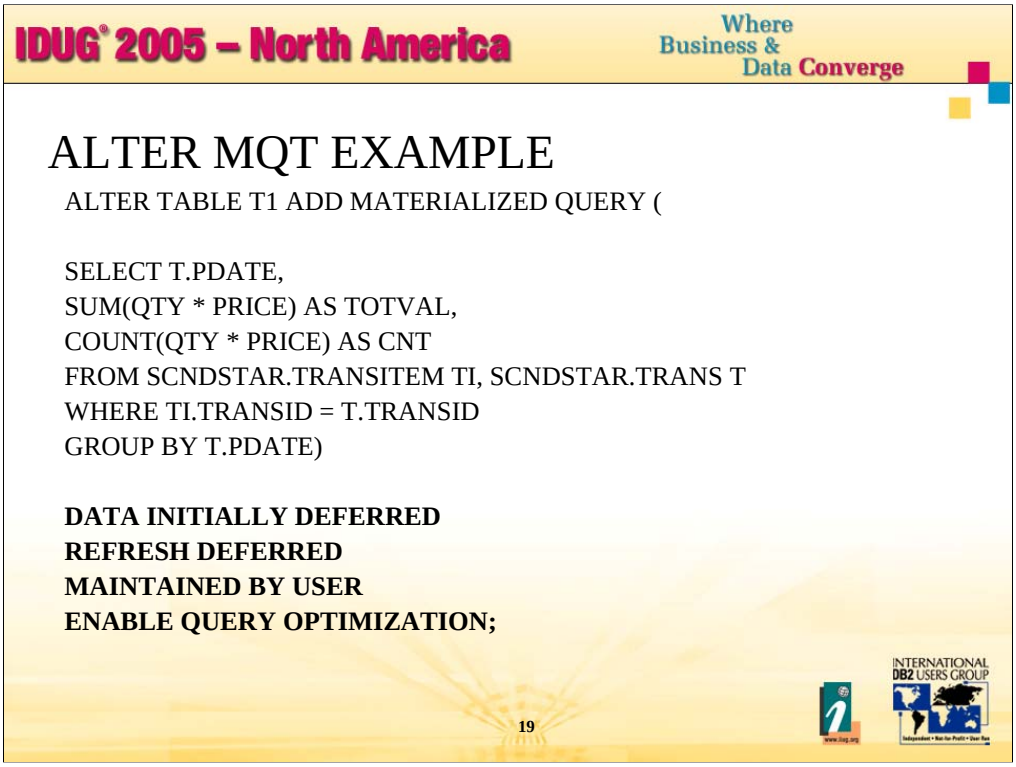
MQT Definition - 3

- Restrictions on FULLSELECT
  - No Special Registers
  - No other MQT
  - No Outer Joins
  - No External Action
  - Single query block after the merge



Consider the power of using ALTER to disable/enable USER or SYSTEM maintenance

And disable or enable query optimization



The example assumes that there is a MQ Table T1 that already exists. The data in table T1 was

generated using the fullselect shown above. That is, the data stored in T1 is the result of a

pre-computation, which performs some scalar functions, a join, GROUP BY, and so on, which

make up the SELECT statement. When altering an existing table into an MQT, it is the user's

responsibility to make sure that the data in the table matches the result of the query that

makes up the MQT.

REFRESH TABLE

- REFRESH TABLE MQTCALCTABLE identifies MQT
  - Full REFRESH
  - SQL Statement
- To “Update” MQT For MAINTAINED BY SYSTEM option
  - No Direct “Update”
  - Deletes all data and reinserts new data(with data from FULLSELECT)
    - Logging and performance issues
  - Updates the catalog for the refresh timestamp and cardinality of the MQT
  - Done as a single UOW
- Or done via LOAD/INSERT/UPDATE/DELETE
  - Defined as MAINTAINED BY USER
    - Could use REFRESH TABLE
  - Could be a triggered function

Only supported for DRDA, not private protocol

AQR Overview

- Automatic Query Rewrite
  - Recognizes when it can use the stored results of a materialized query table
  - Bypasses complicated, expensive query against underlying base table
- Considers estimated cost of the rewritten query with MQT with estimated cost of the original query

Automatic query rewrite is the process DB2 uses to access data in a materialized query table.

AQR Considerations

- For Dynamic SQL
  - Ready Only
- Optimization is at QBLOCK level
  - Complex SQL has many QBLOCKs
- Watch for timing of REFRESH TABLE
  - Consider DISABLE QUERY OPTIMIZATION
  - REFRESH TABLE x
  - ENABLE QUERY OPTIMIZATION
- Can MQT answer the question
  - Columns
  - WHERE clause
  - GROUP BY
- MQT won't be used for some queries
  - Optimizer calculates costs and chooses lowest cost

AQR Management

- For Application Level Control of AQR
- Important in a mixed workload DB2 subsystem
  - Two New Special Registers
    - SET CURRENT REFRESH AGE
      - REFSHAGE ZPARM – ZERO is default
      - ZERO or ANY only options
    - CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
      - MAINTYPE ZPARM – SYSTEM is default
      - ALL/NONE/SYSTEM/USER

Special register CURRENT REFRESH AGE

This special register controls whether an MQT can be considered in query rewrite as follows:

Value '0' means no MQTs are considered in query rewrite

Value 'ANY' means all MQTs are considered in rewrite

The default value of this special register can be specified in the CURRENT REFRESH AGE field on panel DSNTIP4 at installation time.

The data type for CURRENT REFRESH AGE is DEC(20,6).

Special register CURRENT MAINTAINED TABLE TYPES

This special register specifies a VARCHAR(255) value. The name of the dynamic ZPARM is MAINTYPE. The value identifies the types of MQT that can be considered in query rewrite:

Value 'SYSTEM' means all system-maintained, query optimization enabled MQTs

Value 'USER' means all user-maintained, query optimization enabled MQTs

Value 'ALL' means all query optimization enabled MQTs

The initial value of this special register is determined by the value of field CURRENT MAINT TYPES on installation panel DSNTIP6. The default of this field is SYSTEM.

There is a relationship between these two special registers

To be considered - 1

- For user-maintained MQT, start with disable query optimization
  - DB2 might automatically rewrite queries against empty MQT
- Segmented TS consideration
  - Mass Delete on REFRESH TABLE
- RUNSTATS after REFRESH TABLE
- Use MQTs for higher CPU consuming queries
- How many MQT associated with a single base table
  - Could impact “bind” time

To be considered - 2

- A different design approach
  - Implement normalized tables
    - Data integrity during modification
  - Build MQTs to address denormalization
    - Improved performance
- MQTs aggregates results on the most used predicates and joins of large tables
  - DB2 optimizer /AQR can address re-grouping, predicate matching

New Referential Integrity Option

- Informational referential Integrity defines a constraint avoiding the overhead of enforcing RI by DB2
  - Allows DB2 to take advantage of the data association information provided by referential constraints
  - Addresses known relationships not being “enforced”
- Not enforced by Database Manager
- Considered by AQR
- These utilities consider new RI:
  - LISTDEF RI, QUIESCE, REPORT TABLESPACE

RI can be for lossless join, if extra tables are in MQT, these extra tables have to have lossless joins with the common tables. Without RIs, we cannot use an MQT if it contains extra tables for a query, which includes the common cases where a query has some dimensions missing but the MQT contains all dimensions.

CREATE TABLE – NEW RI Option

```
CREATE TABLE SCNDSTAR.TRANS
(TRANSID CHAR(10) NOT NULL PRIMARY KEY,
ACCTID CHAR(10) NOT NULL,
PDATE DATE NOT NULL,
STATUS VARCHAR(15),
LOCID CHAR(10) NOT NULL,
CONSTRAINT ACCTTRAN FOREIGN KEY (ACCTID)
REFERENCES SCNDSTAR.ACCT NOT ENFORCED,
CONSTRAINT LOC_ACCT FOREIGN KEY (LOCID)
REFERENCES SCNDSTAR.LOC NOT ENFORCED)
IN DBND0101.TLND0101;
```


Informational Referential Integrity to be used by AQR

Other Information

- Information about MQT in DB2 Catalog
  - SYSIBM.SYSVIEWS, SYSIBM.SYSTABLES
    - Seven other DB2 Catalog Tables
- No Triggers and Primary Keys
- Explain information
  - TABLE_TYPE of 'M'
- Security is from base table
- Multi-level Security - row-level granularity
  - Checked on reference to data
  - Not on Refresh

Type 'M" for MQTs is stored in SYSIBM.SYSTABLES, SYSIBM.SYSVIEWS, SYSIBM.SYSVIEWDEP, SYSIBM.SYSPLANDEP, SYSIBM.SYSPACKDEP, SYSIBM.SYSVTREE, and SYSIBM.SYSVLTREE.

There are two new columns in catalog table SYSIBM.SYSRELS; they are ENFORCED and CHECKEXISTINGDATA. If the value of ENFORCED is set to N, the entry belongs to an informational RI constraint. CHECKEXISTINGDATA basically contains the same information. If ENFORCED is set to N, CHECKEXISTINGDATA is also always set to N.

Both SYSIBM.SYSTABLES and SYSIBM.SYSROUTINES are expanded by column NUM_DEP_MQTS, which contains the information about how many MQTs are dependent on a table or a table UDF respectively.

In addition to that, table SYSIBM.SYSVIEWS has six new columns that contain information related to MQTs:

REFRESH - 'D' for deferred refresh mode; or blank, which means the row does not belong to an MQT.

- ENABLE - 'Y' or 'N' for QUERY OPTIMIZATION enablement, or blank for a view.
- MAINTENANCE - 'S' for system-maintained, 'U' for user-maintained or blank for view.
- REFRESH_TIME - only used by system-maintained MQTs. It indicates the timestamp of last REFRESH TABLE statement.
- ISOLATION - Isolation level when MQT is created or altered from a base table.
- SIGNATURE - Contains an internal description of the MQT.

SUMMARY

- Where does it fit ?
- More places than you think
- Evaluate the opportunities

Randy Ebersole
DB2 z/OS Lab Services
Information Management - IBM Software Group
ebersole@us.ibm.com

Session F12
Time – Thursday May 26,2005 – 8:30 AM – 9:40 AM
Presentation Platform – DB2 UDB z/OS
Presentation Category – Application Development