



Materialized Query Tables: The Key Ingredient

Tom McKinley
DB2 for IBM i Consultant



IBM Power Systems™

Hosted by:
Centerfield Technology
Rochester, MN USA
www.centerfieldtechnology.com

Agenda

- What are MQTs
- Populating and maintaining MQTs
- MQT Strategy
- Tools to help
- Misc considerations

What are MQTs

Fundamental Idea



Save the results of a query, to reuse....
Instead of rerunning the query again.

Save the results in a permanent SQLTable

Description

- By running the appropriate aggregate query one time and storing the results in an SQL table to be reused for subsequent queries
 - **it is possible to enhance query performance significantly.**
 - **It is possible to REDUCE resource impact of query processing**
- MQTs can improve response time for complex SQL queries, especially queries that involve some of the following:
 - Aggregated or summarized data that covers one or more subject areas
 - Joined and aggregated data covering a set of tables
 - Commonly accessed subset of rows

Creating an MQT

```
CREATE TABLE Sale_Geo_Reg_YR_MON_MQT
AS
(SELECT Geography,
        Region,
        Year,
        Month,
        SUM(Revenue) AS Total_Revenue,
        SUM(Quantity) AS Total_Quantity,
        COUNT(*) AS Rows_per_Group
FROM Sales_Transaction
GROUP BY Geography,
        Region,
        Year,
        Month)
DATA INITIALLY IMMEDIATE
REFRESH DEFERRED
ENABLE QUERY OPTIMIZATION
MAINTAINED BY USER;
```

Geography, Region, Year, Month, total_revenue, Total_Quantity and Rows_per_Group are COLUMNS in the new table SALE_GEO_REG_YR_MON_MQT

Base versus MQT

- Sales_transaction has 10 billion rows
 - 10 Geographies, 4 regions per geo, 3 years of data.
 - 200 bytes each = 2,000,000,000,000
- Sale_GEO_REG_YR_MON_MQT
 - $10 * 4 * 3 * 12 = 1440$ rows
 - ~ 40 bytes per row = 57,600

What are MQTs cont.

- Or, any **other query** that could be answered using those results

Queries allowed to use that MQT

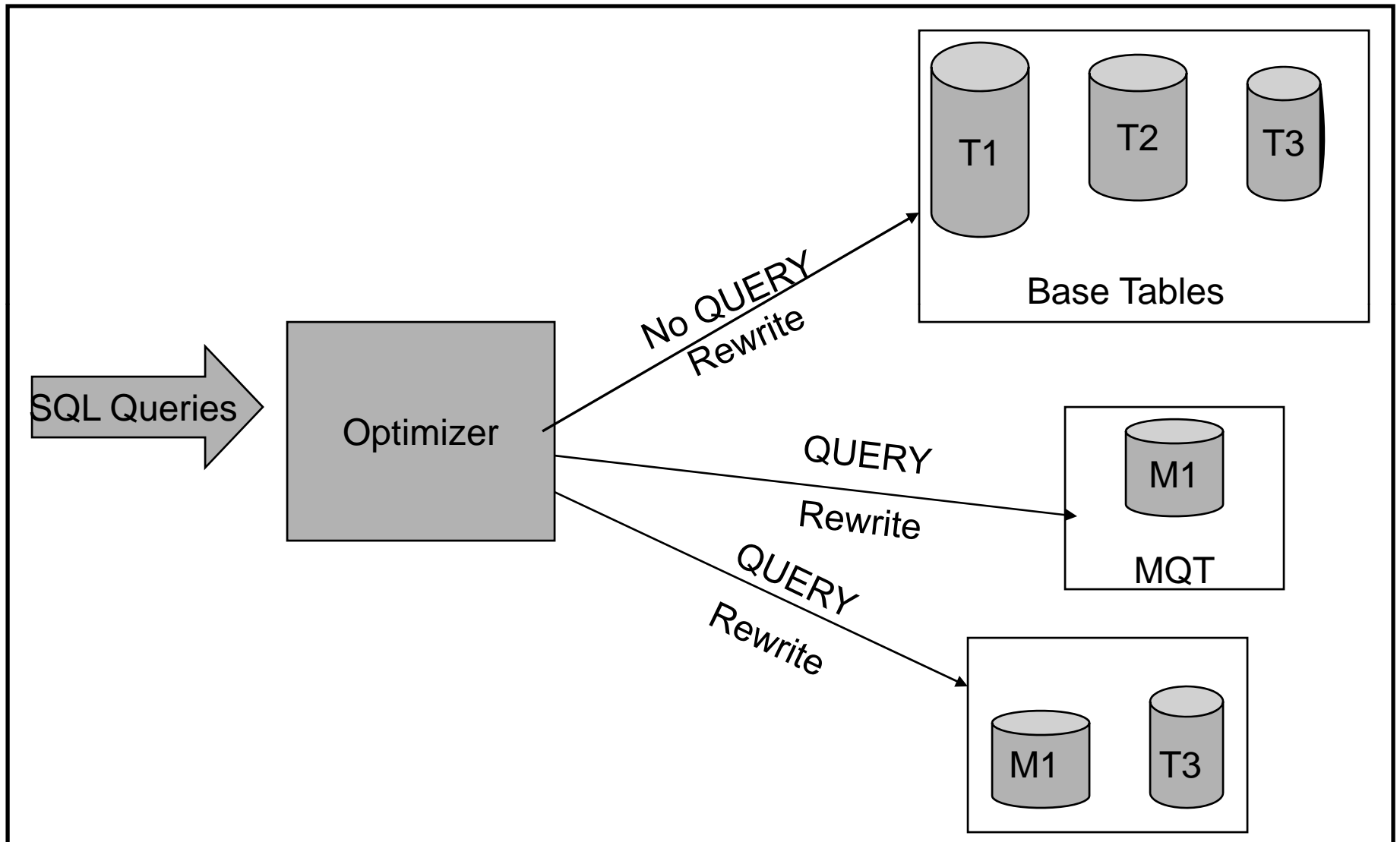
```
SELECT Geography, Region, Year,
       Month,
       SUM(Revenue) AS Total_Revenue,
       SUM(Quantity) AS Total_Quantity,
FROM Sales_Transaction
GROUP BY Geography,
       Region,
       Year,
       Month;
```

```
SELECT Geography, Region,
       AVG(Revenue) AS Avg_Revenue,
       AVG(Quantity) AS Avg_Quantity,
FROM SALES_Transaction
GROUP BY Geography
Region;
```

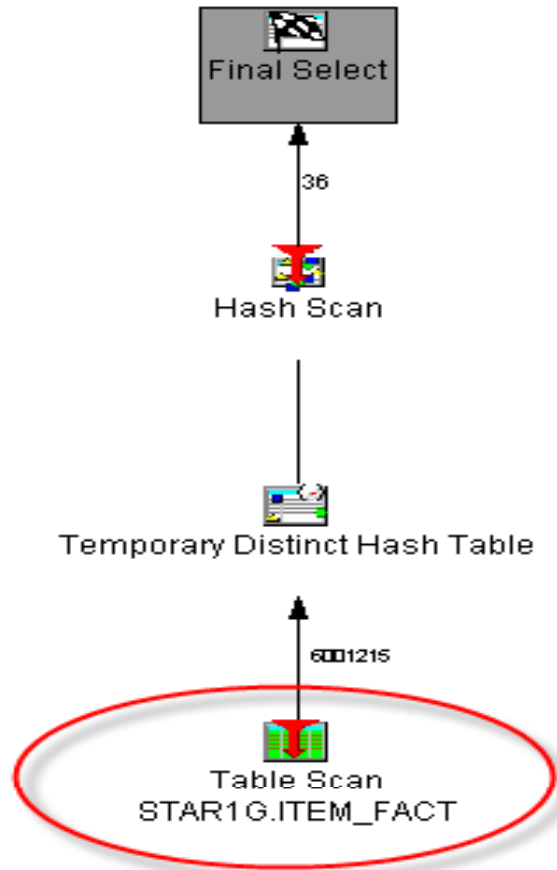
```
SELECT Geography, Year,
       SUM(Revenue) AS Total_Revenue,
       SUM(Quantity) AS Total_Quantity,
FROM Sales_Transaction
WHERE Year IN (2004, 2005)
GROUP BY Geography, Year;
```

```
SELECT Year, Month,
       SUM(Revenue) AS Total_Revenue,
       SUM(Quantity) AS Total_Quantity,
FROM Sales_Transaction
WHERE MONTH= 12
GROUP BY YEAR, Month
ORDER BY YEAR;
```

QUERY REWRITE



Query Rewrite Example

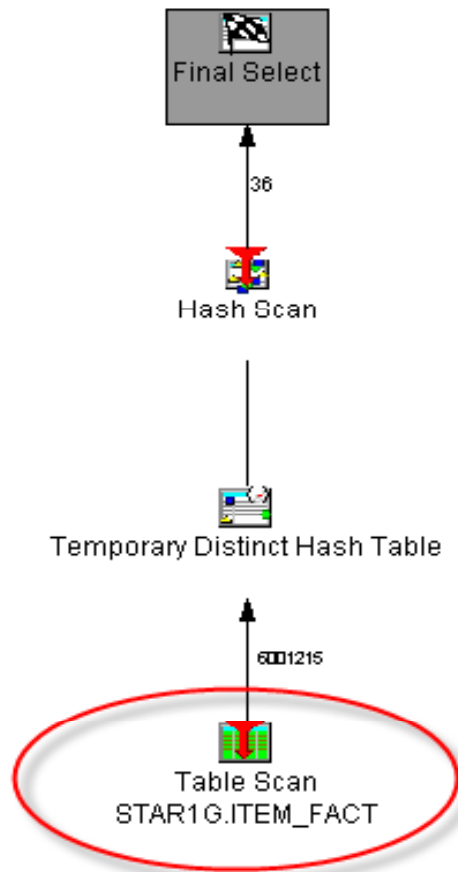


```
SELECT year, quarter, month,  
       SUM(revenue_w_tax) AS srevenue_w_tax,  
       SUM(revenue_wo_tax) AS srevenue_wo_tax,  
       SUM(profit_w_tax) AS sprofit_w_tax,  
       SUM(profit_wo_tax) AS sprofit_wo_tax,  
       SUM(quantity) AS squantity,  
       COUNT(*) as number_items_per_group  
FROM   ITEM_FACT  
GROUP BY year, quarter, month;
```

Without MQT...
Scan and aggregate
6,000,000 rows

Query Rewrite Example

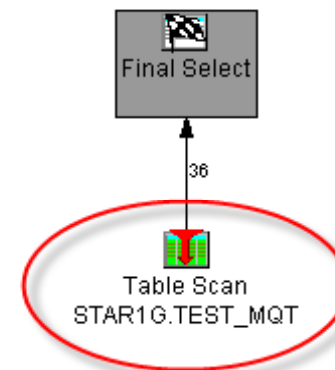
Before...



Without MQT...
Scan and aggregate
6,000,000 rows

```
SELECT year, quarter, month,
       SUM(revenue_w_tax) AS srevenue_w_tax,
       SUM(revenue_wo_tax) AS srevenue_wo_tax,
       SUM(profit_w_tax) AS sprofit_w_tax,
       SUM(profit_wo_tax) AS sprofit_wo_tax,
       SUM(quantity) AS squantity,
       COUNT(*) as number_items_per_group
FROM   TEST_MQT
GROUP BY year, quarter, month;
```

Base table(s)
replaced by
MQT



With MQT...
Scan 36 rows

MQTs and Join Queries

- You need to create MQTs so they can be used by Join Queries
 - Include the join in the MQT definition
 - OR, include the Join column in the MQT so that you can Join to it

Join Example

MQT

```
CREATE TABLE STAR1M.ITEM_REV_PROF_DAY
AS (
  SELECT SHIPDATE , SUM ( REVENUE_W_TAX ) AS SREVENUE_W_TAX ,
  SUM ( REVENUE_WO_TAX ) AS SREVENUE_WO_TAX ,
  SUM ( PROFIT_W_TAX ) AS SPROFIT_W_TAX ,
  SUM ( PROFIT_WO_TAX ) AS SPROFIT_WO_TAX , SUM ( QUANTITY ) AS SQUANTITY ,
  COUNT ( * ) AS NUMBER_ITEMS_PER_GROUP

  FROM STAR1M.ITEM_FACT
  GROUP BY SHIPDATE ) DATA INITIALLY DEFERRED REFRESH DEFERRED MAINTAINED BY
  USER
  ENABLE QUERY OPTIMIZATION ;
```

QUERY

```
Select C.customer, F.Shipdate,
  SUM ( REVENUE_W_TAX ) AS SREVENUE_W_TAX ,
  SUM ( REVENUE_WO_TAX ) AS SREVENUE_WO_TAX ,
  SUM ( PROFIT_W_TAX ) AS SPROFIT_W_TAX ,
  SUM ( PROFIT_WO_TAX ) AS SPROFIT_WO_TAX ,
  SUM ( QUANTITY ) AS SQUANTITY ,
  COUNT ( * ) AS NUMBER_ITEMS_PER_GROUP

From STAR1M.ITEM_FACT f, STAR1M.CUST_DIM c
where f.CUSTKEY= C.CUSTKEY
Group by C.Customer,F.Shipdate Order by SPROFIT_WO_TAX DESC;
```

Can't use that MQT

Visual Explain - Tplxe1(Tplxe1)

File View Actions Options Help

Attribute	Value
Statement Outcome	Unsuccessful
SQL Return Code	-666
SQLSTATE	57005
Cursor Name	CRSR0026
Package Name	
Package Library	
Statement Text	Select c.customer, f.Shipdate
Rows Fetched	0
Total Times Query Was Run	1
Total Time For All Runs, in Millis...	210
List of Materialized Query Table...	STAR1M/ITEM_00001 4, STAR1M/TEST_MQT 4, STAR1M/TEST_MQT2 4
Additional information about SQ...	
CLOSQLCSR Value	
ALWCPYDTA Value	Any Time
Pseudo Open	No
Pseudo Close	No
Hard Close Reason Code	Not Available
ODP Implementation	Reusable
Dynamic Replan Reason Code	Access plan was not rebuilt
Timestamp When Plan Was Cre...	2008-11-17-07.39.22.0974
Data Conversion Reason Code	Not applicable
Dynamic Replan Reason Code	Access plan was not rebuilt

```
Select c.customer, f.Shipdate, SUM (REVENUE_W_TAX) AS SREVENUE_W_TAX, SUM (REVENUE_WO_TAX) AS SREVENUE_WO_TAX, SUM (PROFIT_W_TAX) AS SPROFIT_W_TAX, SUM (PROFIT_WO_TAX) AS SPROFIT_WO_TAX, SUM (QUANTITY) AS SQUANTITY, COUNT (*) AS NUMBER_ITEMS_PER_GROUP From STAR1M.ITEM_FACT f, STAR1M.CUST_DIM c where f.CUSTKEY= C.CUSTKEY Group by C.Customer,F.Shipdate Order by SPROFIT_WO_TAX DESC
```

Statement text

Including the Join Column in the MQT

```
CREATE TABLE STAR1M.ITEM_REV_PROF_DAY_CUST
AS (
  SELECT SHIPDATE , CUSTKEY ,
  SUM ( REVENUE_W_TAX ) AS SREVENUE_W_TAX ,
  SUM ( REVENUE_WO_TAX ) AS SREVENUE_WO_TAX ,
  SUM ( PROFIT_W_TAX ) AS SPROFIT_W_TAX ,
  SUM ( PROFIT_WO_TAX ) AS SPROFIT_WO_TAX ,
  SUM ( QUANTITY ) AS SQUANTITY ,
  COUNT ( * ) AS NUMBER_ITEMS_PER_GROUP
FROM STAR1M.ITEM_FACT
GROUP BY SHIPDATE , CUSTKEY )
DATA INITIALLY DEFERRED REFRESH DEFERRED MAINTAINED
BY USER
ENABLE QUERY OPTIMIZATION ;
```


Using the MQT with the added Join Column

Visual Explain - Tplxe1(Tplxe1)

File View Actions Options Help

Attribute	Value
Statement Number	14
Statement Function	Select
Statement Operation	Open
Statement Type	Dynamic
Statement Name	STMT0011
Statement Outcome	Successful
SQL Return Code	0
SQLSTATE	00000
Cursor Name	CRSR0011
Package Name	
Package Library	
Statement Text	Select customer, Shipdate, SU...
Rows Fetched	0
Total Times Query Was Run	1
Total Time For All Runs, in Millis...	593
List of Materialized Query Table...	STAR1M/ITEM_00001 4, STAR1M/TEST_MQT 4, STAR1M/TEST_MQT2 4, STAR1M/ITEM_MQT1 1, STAR1M/ITEM_REV_PROF_DAY_C UST 0
Additional information about SQ...	
CLOSQLCSR Value	
ALWCPYDTA Value	Any Time
Pseudo Open	No
...	..

Select customer, Shipdate, SUM (REVENUE_W_TAX) AS SREVENUE_W_TAX , SUM (REVENUE_WO_TAX) AS SREVENUE_WO_TAX , SUM (PROFIT_W_TAX) AS SPROFIT_W_TAX , SUM (PROFIT_WO_TAX) AS SPROFIT_WO_TAX , SUM (QUANTITY) AS SQUANTITY , COUNT (*) AS NUMBER_ITEMS_PER_GROUP From STAR1M.ITEM_FACT f, STAR1M.CUST_DIM c where f.CUSTKEY= C.CUSTKEY Group by Customer,Shipdate Order by SPROFIT_WO_TAX DESC

Statement text

Include the Join in the MQT

```
CREATE TABLE STAR1M.ITEM_REV_PROF_DAY_CUSTJ
AS (
  SELECT c.customer,f.SHIPDATE ,
  SUM ( REVENUE_W_TAX ) AS SREVENUE_W_TAX ,
  SUM ( REVENUE_WO_TAX ) AS SREVENUE_WO_TAX ,
  SUM ( PROFIT_W_TAX ) AS SPROFIT_W_TAX ,
  SUM ( PROFIT_WO_TAX ) AS SPROFIT_WO_TAX ,
  SUM ( QUANTITY ) AS SQUANTITY ,
  COUNT ( * ) AS NUMBER_ITEMS_PER_GROUP
  FROM STAR1M.ITEM_FACT F , Star1m.CUST_DIM C
  where f.CUSTKEY= C.CUSTKEY
  GROUP BY SHIPDATE , C.Customer )
DATA INITIALLY DEFERRED REFRESH DEFERRED MAINTAINED
BY USER
ENABLE QUERY OPTIMIZATION ;
```

Join IN MQT definition

Visual Explain - Tplxe1(Tplxe1)

File View Actions Options Help

Attribute	Value
Statement Function	Select
Statement Operation	Open
Statement Type	Dynamic
Statement Name	STMT0045
Statement Outcome	Successful
SQL Return Code	0
SQLSTATE	00000
Cursor Name	CRSR0045
Package Name	
Package Library	
Statement Text	Select c.customer, f.Shipdate, ...
Rows Fetched	0
Total Times Query Was Run	1
Total Time For All Runs, in Milliseconds	147
List of Materialized Query Tables optimized	STAR1M/ITEM_00001 4, STAR1M/TEST_MQT 4, STAR1M/TEST_MQT2 4, STAR1M/ITEM_REV_PROF_DAY_... USTJ 0
Additional information about SQL statement	
CLOSQLCSR Value	
ALWCPYDTA Value	Any Time
Pseudo Open	No
Pseudo Close	No

```
Select c.customer, f.Shipdate, SUM (REVENUE_W_TAX) AS SREVENUE_W_TAX, SUM (REVENUE_WO_TAX) AS SREVENUE_WO_TAX, SUM (PROFIT_W_TAX) AS SPROFIT_W_TAX, SUM (PROFIT_WO_TAX) AS SPROFIT_WO_TAX, SUM (QUANTITY) AS SQUANTITY, COUNT (*) AS NUMBER_ITEMS_PER_GROUP From STAR1M.ITEM_FACT f, STAR1M.CUST_DIM c where f.CUSTKEY= C.CUSTKEY Group by C.Customer,F.Shipdate Order by SPROFIT_WO_TAX DESC
```

start

8:17 AM

Considerations

- DB2 for IBM i does not automatically maintain the MQTs as the data changes in the base tables.
- The decision to implement MQTs depends on answers to the following questions:
 - Is it acceptable if the query gets different results depending on whether the query uses the MQT or the base tables directly?
 - What is the acceptable latency of data for the query?
 - Are the performance benefits of implementing MQTs significant enough to offset the overhead of their creation and maintenance?

Create parameters

CREATE TABLE -MQT name-

AS(SELECT-grouping columns- -aggregate/summary columns-
FROM-table(s)

WHERE-selection columns-

GROUP BY-grouping columns-

ORDER BY-ordering columns-)

DATA INITIALLY IMMEDIATE Or DATA INITIALLY DEFERRED

/ Insert data at creation or not */*

REFRESH DEFERRED Or REFRESH IMMEDIATE

/ Rerun defining query deferred or immediately when base changes */*

ENABLE QUERY OPTIMIZATION or DISABLE QUERY OPTIMIZATION

/ Allow optimizer to implicitly use this or not*

MAINTAINED BY USER or MAINTAINED BY SYSTEM

/ Who maintains MQT contents, only User supported */*

VALUES IN RED NOT SUPPORTED

QAQQINI setting to allow use

- **MATERIALIZED_QUERY_TABLE_USAGE** - This option controls the query optimizer's recognition and use of MQTs:
 - *DEFAULT - The default value is *NONE.
 - *NONE - MQTs are not used in query optimization or implementation.
 - *ALL – The user-maintained, refresh-deferred query tables can be used.
 - *USER - user-maintained MQTs can be used.
- **MATERIALIZED_QUERY_TABLE_REFRESH_AGE** - This option further determines which MQTs are eligible to be used, based on the last time a REFRESH TABLE statement was done:
 - *DEFAULT - The default value is 0. No MQTs can be used.
 - *ANY - Any tables indicated by the MATERIALIZED_QUERY_TABLE_USAGE QAQQINI parameter can be used.
 - Timestamp_duration - Only tables indicated by the MATERIALIZED_QUERY_TABLE_USAGE QAQQINI option that have a REFRESH TABLE performed within the specified timestamp duration are used.

Populating and Maintaining MQTs

Consider Cost of Initial population

- **DATA INITIALLY IMMEDIATE Or DATA INITIALLY DEFERRED At Create time?**
 - Need to understand this cost and schedule to batch window if needed
 - Submit multiple queries running concurrently in separate Jobs to populate distinct sets of rows from the base table(s).

JOB1

```
INSERT INTO Year_Customer_MQT (SELECT Year, Customer, SUM(Revenue) AS Total_Revenue,  
SUM(Quantity) AS Total_Quantity, COUNT(*) AS Rows_per_Group  
FROM Transaction_table WHERE Year = 2003  
GROUP BY Year, Customer);
```

JOB2

```
INSERT INTO Year_Customer_MQT (SELECT Year, Customer, SUM(Revenue) AS Total_Revenue,  
SUM(Quantity) AS Total_Quantity, COUNT(*) AS Rows_per_Group  
FROM Transaction_table WHERE Year = 2004  
GROUP BY Year, Customer);
```

JOB3

```
INSERT INTO Year_Customer_MQT (SELECT Year, Customer, SUM(Revenue) AS Total_Revenue,  
SUM(Quantity) AS Total_Quantity, COUNT(*) AS Rows_per_Group  
FROM Transaction_table WHERE Year = 2005  
GROUP BY Year, Customer);
```


Consider Repopulation Cost

- **Option 1:** Issue the **REFRESH TABLE** statement:

REFRESH TABLE Sale_Geo_Reg_YR_MON_MQT ;

- Be aware that this is a full refresh of the MQT and causes the following: **(Could be very Expensive!!!)**
 - Any indexes on the MQT are removed.
 - The contents of the MQT are removed.
 - The underlying MQT query is run.
 - The MQT is repopulated from the base tables.
 - Any indexes on the MQT are recreated.

Maintenance Strategy

- **Option 2:** Repopulate entire Table, by employing parallel Jobs same as you might have when MQT was originally populated.
 1. Disable MQT
 2. Drop Indexes
 3. Clear MQT (CLRPFM is probably the best)
 4. Submit Multiple parallel Insert jobs
 5. Recreate indexes using SMP
 6. enable MQT

Maintenance Strategy

- **Option 3:** Extract changes since last maintenance and update MQTs programmatically

Insert into Sale_Geo_Reg_YR_MON_MQT

```
select Geography, Region, Year, Month,  
SUM(Revenue) AS Total_Revenue,  
SUM(Quantity) AS Total_Quantity,  
COUNT(*) AS Rows_per_Group  
FROM Sales_Transaction  
Where Transaction_Date > LastMQTmaintenance_Date  
GROUP BY Geography, Region, Year, Month;
```

Or

Use Update if some of the data for a summary is already "In progress" in the MQT.

Maintenance Strategy

OR

- Use a trigger on the base tables to automatically reflect the base table changes in the MQTs
 - After Insert
 - After Update
 - After Delete
- Recommend doing the MQT table change asynchronous to the base table change
 - Send record images in a message to a queue that can be serviced by the MQT maintenance job(s).

MQT Strategy

When do they provide benefit?

- MQTs provide the most benefit when the queries are frequently aggregating or summarizing similar data from many rows
- MQTs provide the most benefit when user queries are frequently aggregating or summarizing data that results in only a few groups.
- Highly selective short running queries will not likely benefit from MQT
 - Where INVOICE_ID= 8675309
- The more often the MQT has to be refreshed, the less effective the MQT might be. This assumes minimal latency between the base tables and the MQTs. If the MQTs require refreshing less often and there is an adequate window of time to perform the refreshes, more MQTs can be employed.

MQT creation

- The general steps for implementing MQT are:
 - Analyze the data model and queries
 - Design and layout the MQT definitions
 - Create and verify the MQTs
 - Verify requires use of Optimizer feedback tools
 - Populate the MQTs
 - Considering the Cost of doing this
 - Test and tune the MQTs
 - Requires use of Optimizer feedback
 - Design and layout the MQT refresh strategies
 - Need to consider this before ever creating and enabling MQTs
 - Test and tune the MQT refresh strategies

Consider the hierarchy in your data.

- Create more granular MQTs and let those rows get re-summarized for the higher level of summarization
 - MQT --> GROUP BY YEAR, QUARTER, MONTH, WEEK, DAY
 - GROUP BY YEAR, QUARTER, MONTH, WEEK (7 MQT rows per result)
 - GROUP BY YEAR, QUARTER, MONTH (~30 MQT rows per result)
 - GROUP BY YEAR, QUARTER (~90 MQT rows Per Result)
 - GROUP BY YEAR (365 MQT rows Per result)

Tools to help

Finding Queries that can benefit from MQTs

- Utilize SQE plan cache to see expensive statements and understand how often they are run.
 - Via System i navigator show statements (V6R1 version is recommended)
- May look for statements that are run very frequently
- Take snapshot of plan cache, or DB monitor to look for common aggregations
- Use visual explain to see MQT reason codes (documented in Database Performance and Query Optimization guide)

Visual Explain Can highlight MQTs

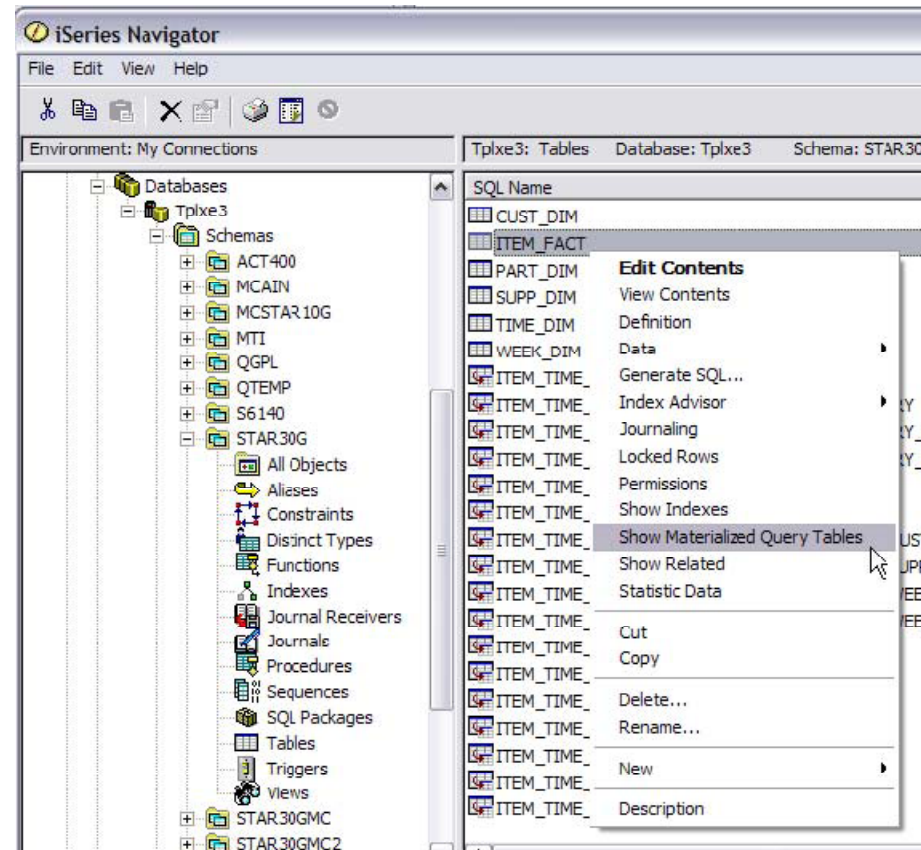
The screenshot shows the Visual Explain interface for a query. The query plan on the left consists of a 'Table Scan' node (highlighted in yellow) feeding into a 'Final Select' node. The 'Table Scan' node is labeled 'STAR30G.ITEM_TIME_MQT_YEAR'. The right-hand panel displays various statistics and information about the SQL statement.

Attribute	Val
Time Information	
Timestamp for Creation of Monit...	2006
Statement Start Timestamp	2006
Statement End Timestamp	2006
Optimization Time, in Milliseconds	2516
Run Time, in Microseconds	1666
Statement Open Time, in Micros...	1666
Statement Fetch Time, in Micros...	Not a
Statement Close Time, in Micros...	Not a
Information about SQL stateme...	
Statement Number	98
Statement Function	Sele
Statement Operation	Ope
Statement Type	Dyna
Statement Name	STM
Statement Outcome	Uns
SQL Return Code	-666
SQLSTATE	5700
Cursor Name	CRE
Package Name	
Package Library	

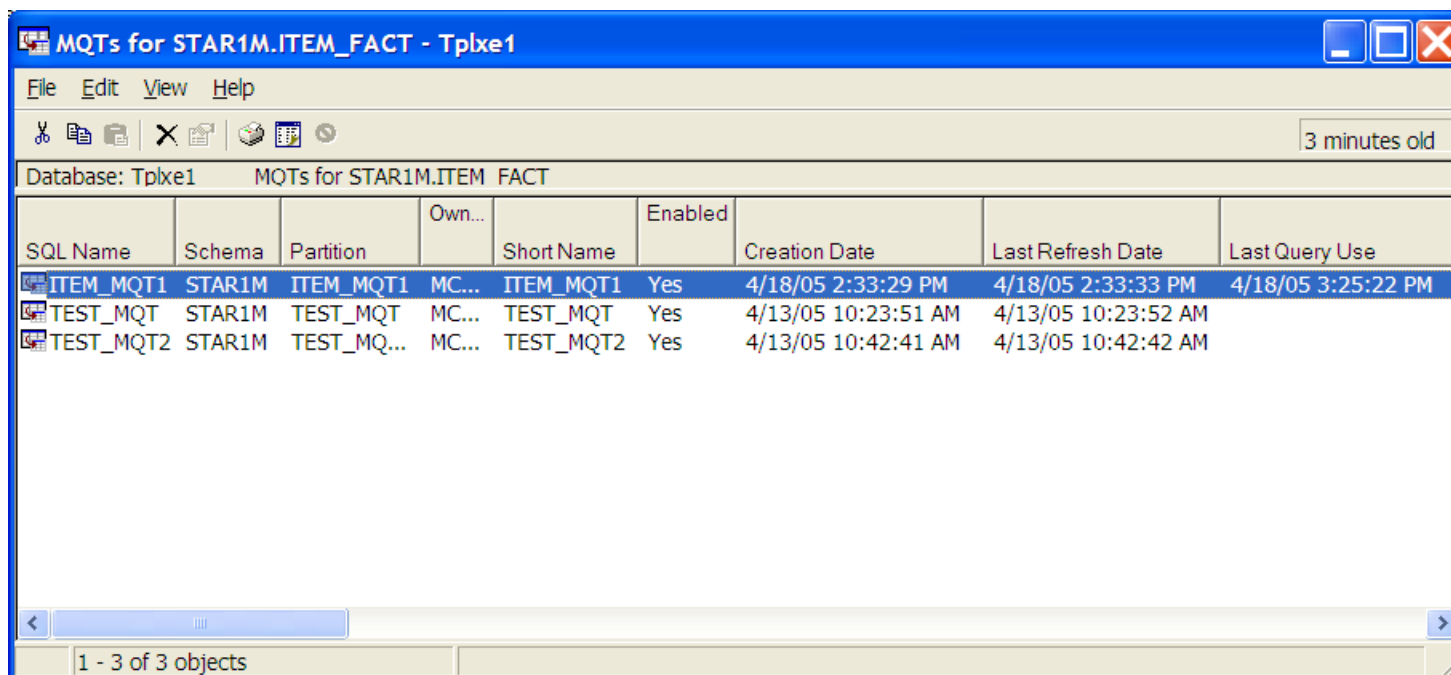
```
SELECT T.YEAR , SUM ( REVENUE_W_TAX ) AS SREVENUE_W_TAX , SUM (
REVENUE_WO_TAX ) AS SREVENUE_WO_TAX , SUM ( PROFIT_W_TAX ) AS SPROFIT_W_TAX ,
SUM ( PROFIT_WO_TAX ) AS SPROFIT_WO_TAX , SUM ( QUANTITY ) AS SQUANTITY FROM
STAR30G.ITEM FACT I , STAR30G.TIME DIM T WHERE I.SHIPDATE = T.DATEKEY GROUP BY
```

Statement text | Optimizer messages

MQTs related to a table and usage statistics



I can see MQTs that exist over a table



MQTs for STAR1M.ITEM_FACT - Tplxe1

File Edit View Help

3 minutes old

Database: Tplxe1 MQTs for STAR1M.ITEM_FACT

SQL Name	Schema	Partition	Own...	Short Name	Enabled	Creation Date	Last Refresh Date	Last Query Use
ITEM_MQT1	STAR1M	ITEM_MQT1	MC...	ITEM_MQT1	Yes	4/18/05 2:33:29 PM	4/18/05 2:33:33 PM	4/18/05 3:25:22 PM
TEST_MQT	STAR1M	TEST_MQT	MC...	TEST_MQT	Yes	4/13/05 10:23:51 AM	4/13/05 10:23:52 AM	
TEST_MQT2	STAR1M	TEST_MQ...	MC...	TEST_MQT2	Yes	4/13/05 10:42:41 AM	4/13/05 10:42:42 AM	

1 - 3 of 3 objects

See list of MQTs from Visual Explain Table Icon

The screenshot shows the Visual Explain interface for a query plan. The plan consists of the following steps:

- Table Scan** (STAR1M.ITEM...): 600,572 rows
- Hash Probe**: 2 rows
- Nested Loop Join**: 601,335 rows
- Temporary Distinct Hash Table**
- Hash Scan**

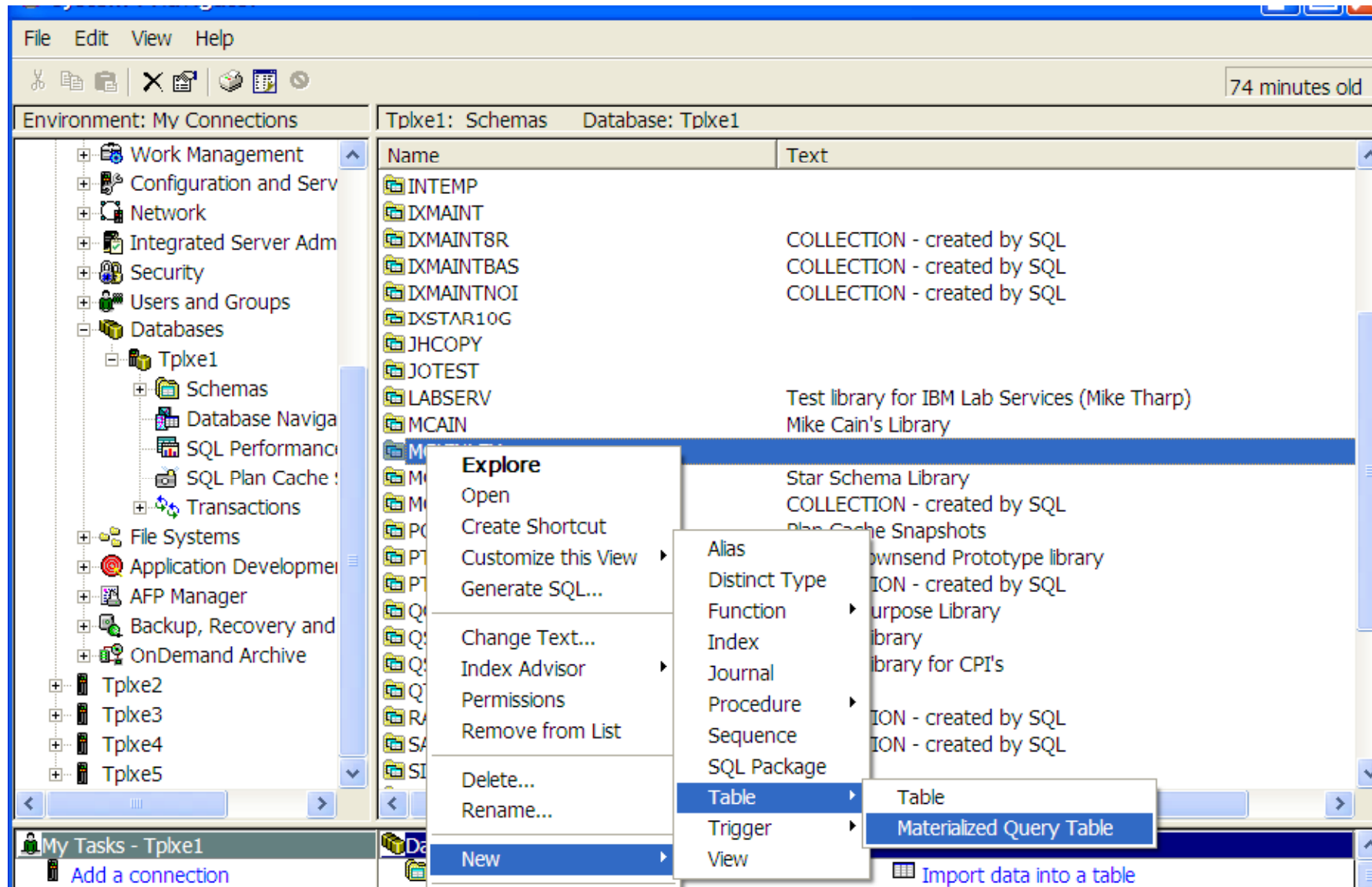
A context menu is open over the Table Scan icon, with the following options:

- Table Definition
- Statistic Data
- Table Description
- Show Indexes
- Show Materialized Query Tables
- Show Related
- Create Index
- Help

The right-hand pane displays the following table of attributes and values:

Attribute	Value
Table name, base table name, in...	
Long Name of Table Being Queried	ITEM_
Library of Table Being Queried	STAR
Member of Table Being Queried	ITEM_
Estimated Time Information (Sta...	
Processing Time(ms)	911.0;
Cumulative Time(ms)	911.0;
Additional Table Info	
Total Rows in Table	600,5;
Table Size(bytes)	160,9;
Active Table Rows	600,5;
Deleted Table Rows	0
Estimated rows selected and qu...	
Total Selected Row Count	600,5;
Total Rows Processed	600,5;
Optimize for N Rows	All
Plan Step Iterations	1
Percent Selectivity	100
Cumulative Percent Selectivity	100
Fetch N Rows	All

Use GUI to create



Find existing MQTs

```
select Table_name, Table_Schema,  
       Table_Type, Enabled FROM  
qsys2.systables  
where TABLE_TYPE='M';
```


Considerations

Resources

- Temporary space when creating and populating the MQTs and associated indexes
- Permanent space to house the MQTs and associated indexes
- Processing resources when creating and maintaining the MQTs and associated indexes
- Time available to create and maintain the MQTs and associated indexes

You can alter an existing table and add/remove the MQT designation.

```
ALTER TABLE Sales_Aggregation
ADD MATERIALIZED QUERY
  (SELECT Geography,
        Region,
        Year,
        Month,
        SUM(Revenue) AS Total_Revenue,
        SUM(Quantity) AS Total_Quantity,
        COUNT(*) AS Rows_per_Group
  FROM SALES_Transaction
  GROUP BY Geography, Region, Year, Month)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
ENABLE QUERY OPTIMIZATION
MAINTAINED BY USER;
```

Getting better performance

- MQTs need indexes too
 - Even If they are small
 - Especially helps if there is selection in the query so that we can probe a small number of rows in the MQT
 - For smaller number of columns Look for perfect index that has all the commonly used fields. Allows for Index only Access.
 - Since, MQT may not be changed as frequently, the index maintenance may not be high overhead.

Miscellaneous tips

- Too many MQTs increase optimization time
- SQE Only
- Mainly use for summarization, not de-normalization.
- Consider explicitly querying the MQTs
- Could use them from Views
- Save restore
 - MQTs put in Check Pending (REASON CODE 19) when restored (see V5R4 PTF (SI33106) . Read special instructions.
- Cascading Creation/Maintenance is a good strategy
 - MQT_YQMD Group by YEAR, QUARTER, Month, DAY
 - MQT_YQ Group by YEAR , QUARTER populate this using MQT_YQMD

Additional info

- **YOU MUST READ THE FOLLOWING PAPER !!!!!**
 - **Creating and using materialized query tables (MQT) in IBM DB2 for i**
 - http://www-03.ibm.com/servers/enable/site/education/abstracts/438a_abs.html
- See DB2 for i Database Performance and Query Optimization guide
- SQL Programmers guide
- SQL Reference

Summary

- MQTs can provide significant benefit. IF LIVE DATA IS NOT REQUIRED.
- System does NOT automatically maintain MQT
- Look for expensive queries and queries run a large number of times.
 - Making 30 second queries run sub second might be huge benefit
- Test MQTs usage, Creation and Maintenance before going live
- MQTs need indexes.
- Learn to use the DB2 for i optimizer feedback tools
- READ “Creating and using materialized query table (MQT) in IBM DB2 for i5/OS” Paper

Questions?