



2-6 October, 2006

Hilton Vienna

Vienna, Austria

IDUG<sup>®</sup> 2006  
Europe

Session: G01

# Data Warehouse Design and Performance



David Beulke  
*Pragmatic Solutions, Inc.*

Monday, October 2, 2006 11:30-12:30

Platform: Cross Platform

GoFurther





IDUG® 2006 - Europe

## Dave Beulke from Pragmatic Solutions, Inc.

- IBM Gold Consultant
- Past President of IDUG
- Best speaker at CMG conference & former TDWI instructor
- Co-Author of certification tests
  - DB2 V8 & V7 DBA certification test
  - IBM Business Intelligence certification test
- Columnist for DB2 Magazine and former editor of the IDUG Solutions Journal
- Seminar Author
  - Data Warehouse Performance Seminar
  - How to do your own DB2 Performance Review
  - NEW! Performance for SOA Java DB2 Environments
- Author of Syspedia - Data element analytics - [www.syspedia.com](http://www.syspedia.com)
- Extensive experience in performance and design of large database and data warehouse systems
  - Working with DB2 on z/OS since V1.2
  - Working with DB2 on LUW since OS/2 Extended Edition



DaveBeulke@cs.com or (703) 798-3283

Pragmatic Solutions, Inc.

2

DaveBeulke@cs.com

**David Beulke is an internationally recognized DB2 consultant, author and lecturer. He is known for his extensive expertise in database performance, data warehouses and internet applications. He is Past President of the International DB2 Users Group (IDUG), a member of IBM DB2 Gold Consultant program, a columnist for DB2 Magazine, co-author of the IBM V8 and V7 z/OS DB2 Administration Certification exam, co-author of the Business Intelligence Certification exam, former instructor for The Data Warehouse Institute (TDWI), and former editor of the IDUG Solutions Journal.**

**His performance tuning expertise has helped clients with their mainframe, UNIX and Windows DB2 systems. His clients have saved millions of dollars in CPU charges and avoided unnecessary hardware upgrades by following his performance tuning advice.**

[DaveBeulke@cs.com](mailto:DaveBeulke@cs.com)

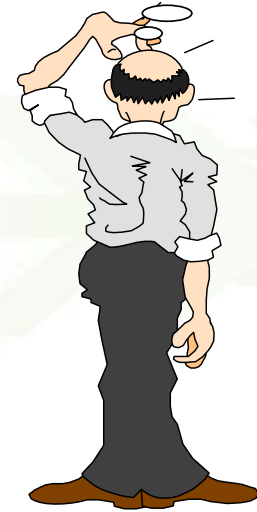
(703) 798-3283

© Copyright Pragmatic Solutions, Inc.



## Agenda

- Performance Components
  - V8 Enhancements
  - Features & Basics
- Best database design options
  - Requirements and Options
  - Partitioning and Parallelism
  - Gathering the workload
- MQT - Considerations
  - Distributed platform
  - z/OS Version 8
  - Strategies for MQTs with Views



**Performance is made up of several factors from the beginning of the analysis to the end. Sometimes there is not an end because the workload keeps changing.**

- 1. DB2 LUW DW Performance Factors**
- 2. DB2 DW Configuration Parameters**
- 3. Database and Index Definition Considerations**
- 4. DASD and other hardware considerations**
- 5. MQTs can make your applications sub second**



## SQL Features

- OLAP functions
- RANK
- DENSE\_RANK
- ROW\_NUMBER
  - ORDER BY clause
- OVER clause
  - PARTITION BY clause
  - RANGE clause
  - ROW clause
- ROLLUP
- CUBE
  - Group By or Grouping Sets
- DB2 Cube Views
  - Virtual cube backed by real structures
- XML and Metadata extensions

**RANK Example:**  
 SELECT WORKDEPT,  
 AVG(SALARY+BONUS)  
 AS AVG\_TOTAL\_SALARY,  
 RANK() OVER (ORDER BY  
 AVG(SALARY+BONUS) DESC)  
 AS RANK\_AVG\_SAL  
 FROM BEULKE.EMPLOYEE  
 GROUP BY WORKDEPT  
 ORDER BY RANK\_AVG\_SAL

**DENSE RANK Example:**  
 SELECT WORKDEPT, EMPNO,  
 LASTNAME, FIRSTNAME, EDLEVEL  
 DENSE\_RANK()  
 OVER (PARTITION BY  
 WORKDEPT  
 ORDER BY EDLEVEL DESC)  
 AS RANK\_EDLEVEL  
 FROM EMPLOYEE  
 ORDER BY WORKDEPT, LASTNAME

**ROW NUMBER Example:**  
 SELECT ROW\_NUMBER()  
 OVER (ORDER BY WORKDEPT, LASTNAME)  
 AS NUMBER,  
 LASTNAME,  
 SALARY  
 FROM EMPLOYEE  
 ORDER BY WORKDEPT, LASTNAME



The SQL OLAP functions performed inside DB2 provide the answers much more efficiently than manipulating the data in a program. Like join activities and other data manipulation that DB2 can do directly the SQL OLAP functions can greatly reduce overall I/O and CPU utilization.

The functions are particularly good for getting the top number of data rows that match a criteria.

The OLAP RANK Function can be used to order and prioritize your data according to your specific criteria. RANK orders your data assigning successive sequential numbers to the rows returned from the SQL query. The ranked rows can be individual data rows or groups of data rows.

The OLAP DENSE\_RANK Function can also be used to order and prioritize your data according to your specific criteria.

DENSE\_RANK orders your data and assigns successive sequential numbers based on the OVER PARTITION data values found. DENSE\_RANK differs from RANK because common values or ties are assigned the same number.

The OLAP ROW NUMBER function can be used to assign row numbers to the retrieved data. In addition ROW NUMBER can be used to order your data assigning successive sequential numbers for evaluation for application decisions or end-user screen processing.

Success, ROI, and performance are all determined by the end-user getting their questions answered. The common preliminary questions for a sales based DW that must be answered are:

- How much product did we sell last (year, month or time period)?
- How do these sales figures compare to last (year, month or AP)?
- How much profit did we make on sales during this period?
- Who did we sell our products to?
- What categories or classifications separate our customers ?
- What products were sold to which customers classifications?
- What was the response rate on the marketing promotion?
- What percentage of our customer are new?

How many I/Os does it take with your DW design to get these common questions answered?



## Z/OS SQL Features

- CTEs – Common Table Expressions
  - Recursive SQL
  - Bill of Material
    - Parts of parts
- Multi-Row Actions
  - INSERT
  - FETCH
    - NEXT/PRIOR Cursors
  - UPDATE/DELETE with cursor



**Next the new Version 8 features of Common Table Expressions (CTEs) and recursive SQL can be combined to provide a powerful data warehousing design solution for extracting and working on a distinct set of information. CTEs provide a new way to extract a result set from the database based on desired criteria. Next this unique result set can be referenced in additional SQL, further refining the answer for the end-user. CTEs avoid the catalog overhead of views, provide the ability to use host variables and avoid data changes from other INSERT, UPDATE or DELETE SQL operations. With recursive SQL, SQL that references itself, distinct result sets can have the power of SQL repeatedly applied to quickly and efficiently derive the answers. Combining these techniques provides a great way to give the data warehouse end-users answers to their unique criteria while avoiding conflicts with other users, maintaining data security and easily repeating the power of SQL criteria.**



## Z/OS SQL Features

- Scrollable Cursors
  - Sensitive Dynamic
- INSERT with SELECT
- Identity column improvements
- New Sequence column
- Group by A+B
- Multiple DISTINCT



**Next the new Version 8 features of Common Table Expressions (CTEs) and recursive SQL can be combined to provide a powerful data warehousing design solution for extracting and working on a distinct set of information. CTEs provide a new way to extract a result set from the database based on desired criteria. Next this unique result set can be referenced in additional SQL, further refining the answer for the end-user. CTEs avoid the catalog overhead of views, provide the ability to use host variables and avoid data changes from other INSERT, UPDATE or DELETE SQL operations. With recursive SQL, SQL that references itself, distinct result sets can have the power of SQL repeatedly applied to quickly and efficiently derive the answers. Combining these techniques provides a great way to give the data warehouse end-users answers to their unique criteria while avoiding conflicts with other users, maintaining data security and easily repeating the power of SQL criteria.**



## Altering DBMS reality

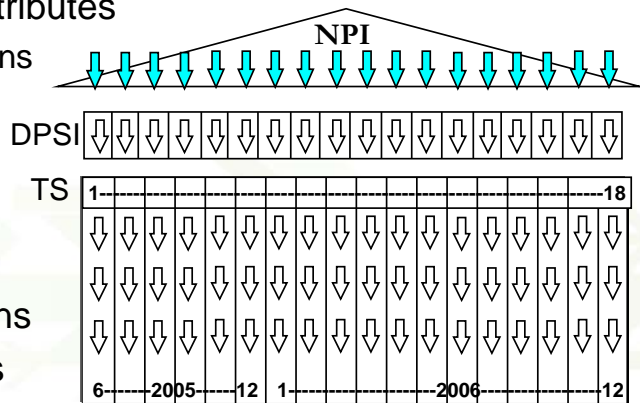
- Change table/column attributes

- Expand char columns
- Numeric

- Table Versioning

- Up to 4096 partitions

- Add or rotate partitions
- More flexible designs
  - A partition a day...



**Version 8 also provides a powerful data warehouse design alternatives with the expansion and rotation capabilities of DB2 z/OS table partitioning. In Version 8 the maximum number of partitions has been expanded from 254 to 4096.**

**Having the capability to define a separate table partition for every day for over ten years is capability unique of all DBMSs to DB2 z/OS and remarkable design flexibility for data warehouses. Also having the ability to rotate partitions for implemented partitioned table designs provides the ability to enhance existing partitioned databases.**

**In Version 8 star join related features have been improved in several new ways. The most important features are the new sparse indexing capability, enhancements to the DB2 optimizer materialization and a dedicated virtual memory pool. These features along with new additional ZPARMs and the RUNSTATS statistical distribution and sampling capabilities directly help the DB2 optimizer choose the most efficient star join data warehouse table access path.**



## Partitioning parallelism reduces time

Determine I/O requirements per year/month/week/day

- Formula = CPU ms + 2-20 ms per call  
5B per year = rows per month = 400,000,000 rows =  
400,000 CPU seconds + 800,000- 8,000,000 I/O secs
- *Elapsed time 222 to 2,222 hours processing each month*

- 10 parallel weekly schedules

- $(2,222 / 4) / 10 = 55.55$  hours
- Same CPU requirements

- SQL Queries

- Partitioning encourages query parallelism



### How many partitions?

**Determine how many partitions and parallel processing streams your CPU, I/O and network can support.**

### Do your calculations

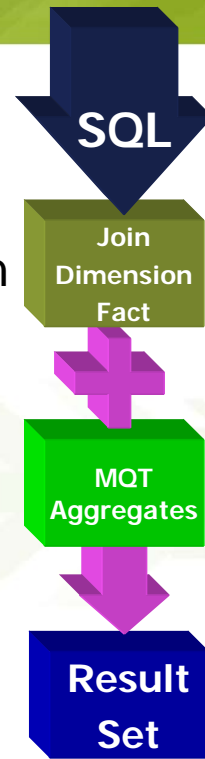
**Make sure you include all the various SQL validation and dimensional key translation work.**



IDUG® 2006 - Europe

## Parallelism – All Aspects

- **All** aspects
  - ETL, validation and transformation
  - Loading query and correction
- What keys provide parallelism?
  - Partitioning key only?
- How are MQT involved?
  - Can they be joined within a query



Pragmatic Solutions, Inc.

9

DaveBeulke@cs.com

### Extract, Transformation, Load and Maintenance

Parallelism designs for reducing the time windows for ETL, maintenance and end-user access processing should be weighed against each other to come up with the best overall design.

### Partition Keys

When analyzing different partitioning key candidates, study the keys that are available to the various processes that maintain the warehouse. Try to find a key that is common among all the processes for the partitioning and data distribution scheme. The various processes can then be split via the partitioning key and run in parallel drastically cutting the overall processing time.

### MQT - Aggregates Keys

Also verify that the same partitioning keys and scheme are available to the aggregate tables. This is critical to joining the tables effectively and efficiently.

### Evenly distribute the data

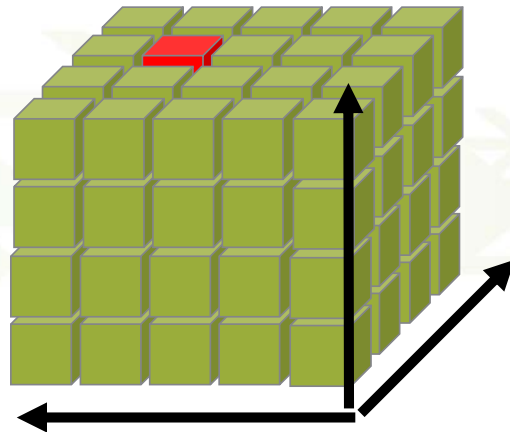
The separation rules are very important because of the potential to evenly distribute the data. Distributing the data allows multiple entry points into the data based on the rules or keys that separate the data.



IDUG® 2006 - Europe

## MDC- Multidimensional Clustering

- New type of table
  - Special MDC syntax
- Clustering along multiple dimension keys
  - Possible to cluster on many different dimensions of a DW
- Clustered equally across all dimensions
  - **Block Space management**
- New smaller Dimension Block indexes
- Faster loads, queries and updates



Pragmatic Solutions, Inc.

10

DaveBeulke@cs.com

Another new feature with DB2 is the new patented clustering technique MDC - Multi-Dimensional Clustering. This feature does exactly as the name implies, it clusters the data against multiple dimensional keys. The MDC clustering is achieved by managing data row placement into page extent blocks based on their dimensional key values. The management, placement and access are facilitated through a new Version 8 Block Index object type. This new Block Index object type is created for each of the dimensions, is similar in structure to a normal index but cross references rows to a dimensional data block instead of an individual row.

The extent data page block sizes are chosen at Multi-Dimensional Clustering definition time and if additional space is needed consecutive block extents are defined. Since the rows are managed to a data block, the cross-referencing Block index information needed is smaller, resulting in a smaller index structure. With consecutive pages and the Block index only referencing data blocks, Block index reorganization will not be needed as often as a regular indexes referencing individual rows.



## MDC Considerations

- Best used with < 7 dimensions
  - Must calculate the possible block population
    - 4 dimension:
      - Region:5 State:50 Store: 250 Product: 10000
      - $5 \times 50 \times 250 \times 10,000 = 625,000,000$
    - How many rows go to each MDC Block?
  - How many partitioning dimensions?
    - Can become a space problem
  - Design Advisor: suggests - Generated columns?
    - Utilities still needed?
  - REORG needed to reclaim space



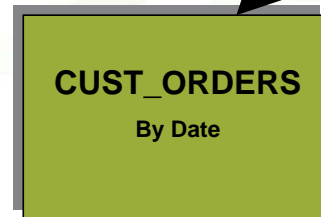
**Clustering along multi dimensional keys also has tremendous value for regular insert, update activities also. With a regular table, the data is placed via a single clustering value and becomes un-clustered with more insert and update activity. Multi-Dimensional Clustering tables maintain their clustering continuously over time because the clustering is based on multiple clustering keys that point to data blocks instead of individual rows.**



## New Indexes Opportunities

- Separate clustering and partitioning indexes
  - Clustering is not defined through partitioning index
  - Partitioning can be done in table definition DDL
    - PARTITION ENDING AT clause
- Cluster for biggest workload
  - Data load/inserts/maintenance
  - SQL activity usually ?10-25%? scanned
  - Compliment MQT aggregates

ORDER\_DEPT\_NBR\_IX



### Clustering for Performance

**Eliminating and minimizing I/Os can make or break performance objectives. A prototype design should be put together and typical queries estimated. SQL Traces on the system during the user testing can help point out system issues. Clustering should be modeled against the most popular usage of the table and to eliminate as many sorts as possible.**



## Index Enhancements

- Better comparison capabilities
  - Unlike Column and Host variable definitions
  - Character comparisons of unequal length
  - Decimal and integer comparisons
    - Especially important for Java, C#, .NET apps
- Better local and remote join operations
  - WHERE host= x + z
- EBCDIC and UNICODE Joins index-able
- Varchar padding or not
  - More index entries per page fewer pages/levels
  - Index only access available



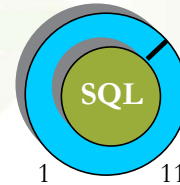
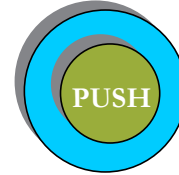
**DB2 Version 8 can now use an index when it is comparing variables of different data types and lengths. This ability allows more indexes to be used during join operations and when comparing host variables to db2 columns.**

**VARCHAR columns that are used in indexes now have the option of being not padded to their largest length. By reducing the index entry size, more index entries can be fit per DB2 page. This will reduce the number of index pages and potentially the number of levels within an index structure.**



## Index Enhancements

- More knobs - zParms
  - Size of SQL now 2M
  - Tables\_join\_threshold
  - MXTBJOIN – max tables joined default 225
  - MXQBCE - max combinations considered
  - Optimization resource thresholds
    - Max\_Opt\_Stor - RDS storage 20m
    - Max\_Opt\_CPU – CPU 100 seconds
    - Max\_Opt\_Elap – Elapse Time 100 seconds
  - SJMXPOOL – VPool to 1,024 – 20m



**The sparse index capability reduces I/O associated with unqualified rows that might have been considered and then eliminated through a sort of a large result set. The new sparse index capability builds an index of only qualified rows. This eliminates the unqualified data early in the access path greatly improving query response time and CPU performance.**

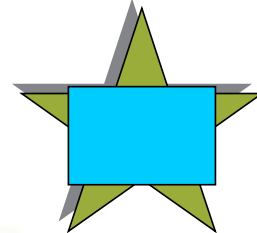
**Improvements in DB2 optimizer algorithms also determine the cost effectiveness of when the snowflake or raw data warehouse dimensions should be materialized. The improved algorithms help DB2 analyze the SQL and determine whether and when the access path would benefit from materializing the snowflake dimensions. All of these improvements can have a dramatic performance impact on your ever-increasing complex data warehousing front-end OLAP tool queries.**



IDUG® 2006 - Europe

## STAR Join Enhancements

- Dynamic creation and use of sparse index
  - Binary search of index
  - Up to 240kb index
    - V7 PQ61458
- Dedicated Vpool for STAR Join work-file
  - Better utilization of memory
- Improve parallel sort optimization
  - Through multiple DSNDB07 allocations



Pragmatic Solutions, Inc.

15

DaveBeulke@cs.com

**Also new in Version 8 is the ability to create a dedicated virtual pool for star join workfiles usually created for materialized dimensions or composite data. By dedicating a data warehousing workfile virtual pool, repeated scanning or access to this work data is done in memory. Since this star join workfile pool is in addition to any other buffer pools, sort operations and query performance is improved.**



## Capture the environment characteristics

- Number of CPUs per LPAR available
  - Virtualization of CPUs in AIX 5.3
- Amount of LPAR memory available for workload
  - Amount of paging that is happening
- Number of disk drives
  - Amount of I/O to individual drives



**Some of the latest Unix computers are being LPARed into several virtual machines to handle test and production workloads. These LPAR configurations and their workloads need to be documented so that the workload runtime characteristics can be monitored with a stable environment.**

**Since these LPARs can have a small amount of memory and CPU capacity even small changes can make a big difference. Monitor your environment with stable settings and workloads to get a comprehensive profile of your environment. This accurate profile can then be analyzed to determine the best changes to improve the overall throughput of the environment.**



## Capture the workload

- Know your DBM and DB configuration settings
- Snapshot of the workload
  - Point in time of what is executing
    - Repeat the process at regular intervals
- Event Monitor – capture everything over given period
  - Get as much information as possible
  - Try to gather the information on a regular basis
- SQL statement executed are the most important item
  - Statement table
    - Capture statements as many as possible
  - Buffer pool data reflect I/O activity



**Gathering information about the DB2 environment should start with the Database Manager and the database configurations parameters. These various settings can have a dramatic effect on the performance of the workload.**

**Using the DB2 Snapshot and Event monitor capabilities you can capture everything that is happening in the DB2 environment and database. DB2 can quickly capture the data and put it in DB2 monitor tables for easy SQL access.**



## Easy steps to set up the monitors

- Connect and check monitor switches
  - Verify MON\_HEAP and APP\_CTL\_HEAP\_SZ are big enough
- Start capturing the information
  - `CREATE EVENT MONITOR beulke1 FOR BUFFERPOOLS, STATEMENTS, TRANSACTIONS WRITE TO TABLE`
- Turn all the switches on
  - `DB2 update monitor switches using lock on sort on bufferpool on statement on table on uow on timestamp on`
- Turn on the monitor switches
  - `DB2 SET EVENT MONITOR beulke1 STATE 1`



### Capturing all the workload causes overhead

**Capturing performance data and setting up the monitoring is very easy and can be done through these simple steps. With these commands and settings DB2 will automatically create and write the monitoring data to DB2 tables. DB2 generates the data based on the workload that it is monitoring and you need to make sure there is enough DASD available for your performance data.**

**Once the data is collected turn off the monitors and analyze the data. The SQL statements are one of the best items to analyze because the CPU, elapsed time and sort overflows are captured. Also counting the number of statements executed and understanding their frequency and data requirements can point to I/O and data bottlenecks.**



## SQL to gather Statement data

- Gives you the most popular SQL statements
- Count the SQL types executed
- Determine the bad applications
  - Most expensive processes
    - CPU usage
    - Elapse time
    - Number of Sorts
    - Number of Sort overflows

```
-- show the top 10 longest running statements,
select
int(st.stmt_elapsed_time_s/60) as Elapsed_min,
substr(st.stmt_text,1,250) as SQL_Text
from root.stmt_beulke1
order by 1 desc
fetch first 10 rows only;
DaveBeulke@cs.com
```



**Some SQL statements for the monitor statement table for further research. There is a great amount of information found in the event monitor tables. Research the many details in the other monitor tables to get authorization id and TCPIP address of your workload.**

```
-- show the top 10 longest running statements,
select
int(st.stmt_elapsed_time_s/60) as Elapsed_min,
substr(st.stmt_text,1,250) as SQL_Text
from root.stmt_beulke1
order by 1 desc
fetch first 10 rows only;
```



## DB2 LUW Sort Parameters

- **Sort capacity**
- **Private Sort**
- **Shared Sort**
- **Binary Short - DB2\_BINSORT**
- **Sort overflows**
- **Sort Heap Threshold**
- **Sort memory overflow**



**Sorting within the smaller DB2 LUW capacity environments can be an performance issue because the large number of rows to be sorted or the memory constraints of the machine. These sorting issues can sometimes be addressed adjusting the various DB2 LUW sort parameters.**

**Capturing the sort activity and the SQL statements causing the sorts is the first priority when trying to understand the workload sort capacity requirements.**



## LUW Performance - Background

- Every client, database, workload & solution unique
  - Many parameters available MQT formerly known as AST
    - Design are sometime flexible
  - Workload characteristics drive the solutions
- Captured workload can then be used for tuning
  - The workload can be fed to the DB2 design advisor
    - The frequencies of the SQL statements
    - The settings within the environment are used
- Provides design and index suggestions



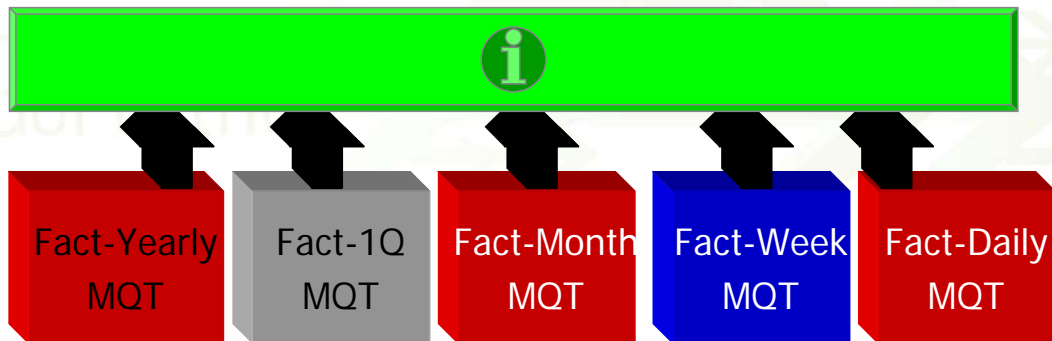
**Every client situation is different and the performance issues are never the same. I try to discover the database design flaws through the performance of the application SQL. Capturing the SQL is great information and it can be very time consuming to examine.**

**Using the execution frequency counts and the SQL statements as input, the DB2 design advisor can suggest new indexes and database design changes or new database objects. Since I design and work with many data warehouse and business intelligence applications I often investigate the possibility of using Materialize Query Tables (MQTs). These MQTs are summarized information built from other source tables.**



## Materialized Query Tables

- Available on z/OS and LUW
- Improved data refresh options
- Aggregate via multiple tables
- Design and aggregate for users



### Materialized Views

Another method of speeding analysis is through the use of Materialized Query Tables (MQTs) as aggregate data stores that specialize in a limited dimensional function. These MQTs are good for taking complicated join predicates and stabilizing the access path for end-users. Warehouse history can also be summarized into MQTs to provide standard comparisons for standard accounting periods or management reports.

### Comparison Points

MQTs function best when defined to existing end-user comparison points. These aggregates can be used extensively for functions and formulas because of their totaled data

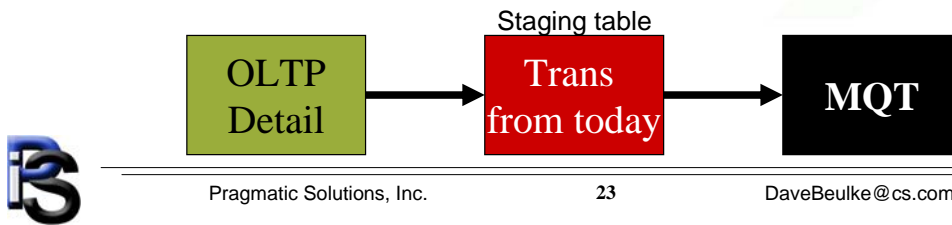
### Intelligent Queries

Meta-data about aggregates must be well documented and published to all end-users and their tools. Tools should be aggregate aware and be able to include the different appropriate aggregate if needed.



## Design Advisor Restrictions

- Design Advisor does not recommend incremental MQTs
  - If you want to create incremental MQTs, you can take REFRESH IMMEDIATE MQTs and convert these to incremental with a staging table.
- MQT indexes should be defined for workload performance and not refresh performance
  - REFRESH IMMEDIATE MQTs should have unique indexes created on their GROUP BY clause unique key.



**The DB2 Design Advisor has many options and many parameters. Unfortunately it sometimes suggests too many indexes or changes. During one client engagement the DB2 design advisor suggested over 100+ indexes to be added to a 42 table database. So take the suggestions with great care and implement what you believe are the best suggestions.**



## MQTs – Basics

- MQT save millions of I/Os for applications
  - MQTs improve performance more than any system or parameter changes
  - Millions of I/Os and CPU cycles saved every day
- Create MQTs for DW or common questions
  - Best selling division, department or product
  - Most productive region, state, county or store
- Create MQTs that can be used everywhere
  - Within Views
  - Joined with other base tables
  - Partnered with other MQTs



**MQTs are very easily to implement since a MQT is simply a table that is created from the result set of a SQL query. Even though there are some minor MQT creation restrictions they can quickly help all types of operational and data warehousing systems.**

**For example I recently created MQTs that pre calculated weekly, monthly and quarter totals, counts and different calculations. Research showed that these figures and calculations were being done repeatedly and were resulting in numerous scans and sorting within the system workload.**

**By creating these MQTs that did these calculations once the automatic query optimization and query rewrite capabilities in DB2 re-wrote and re-optimized the SQL workload to use the pre-calculated MQT table saving tremendous amounts of CPU, I/O and sorting to get the answers.**



## MQTs - Requirements & Options

- Find all SQL functions used in workload
  - Analyze base tables and their columns - NULLs
  - Analyze the types of functions used
  - SUM, AVG, Count etc....
- Know data change frequency
  - SQL UPDATE, INSERT, DELETE
  - What is the schedule of change activity
  - Do you need a staging table



**How active is the data that you are trying to create MQTs from? If the data is very active the MQT needs to be controlled and refreshed at the appropriate time. This is to control the number of times the data is refreshed and to minimize the performance impact of building the summarized data.**

**If the base data is extremely large, build a staging table to capture the data changes since the last refresh for the adding into the MQT. This will help minimize the number of rows that need to be added to the MQT and help minimize the impact on the base table data.**



## MQT - Restrictions

- MQTs can not be created with:
  - Host variable
  - Sub-selects
  - Views requiring materialization
  - Special Register
  - Scalar Fullselect
  - Row Expression Predicate
  - Joined tables with multiple CCSIDs
  - Some exotic built-in functions

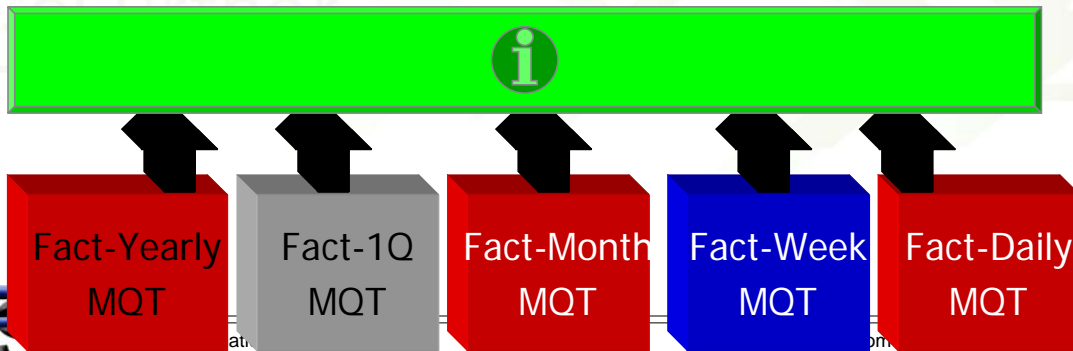


**MQTs can not be built from all types of tables and there are restrictions. These restrictions are very minimal since MQTs are built from standard SQL. Check the manuals for a full list of the restrictions. I usually just try creating the MQT and see if it works. DB2 will tell you if it can't build the MQT.**



## Materialized Query Tables

- Optimizer can dynamically use MQT
  - Year to date aggregates
  - Subtotals for departments
- Make sure to use common indexes
  - Partition and cluster for parallelism
- Can register existing tables as MQT
  - Plan Table Type = "M"



### MQTs and Horizontal Aggregation

Tracking query activity can sometimes point to special data requirements or queries that happen on a frequent basis. These queries may be totaling particular products or regions that could be optimized through a Materialized Query Table (MQT or horizontal aggregate).

### Eliminate I/Os for answers

Analysis must be done to justify the definition of an MQT to make sure it is used enough. Like all aggregates, MQTs and DGTTs if used enough can eliminate I/Os and conserve CPU resources. MQTs aggregates work from a particular dimension key that is can be easily separated from the rest of the data.



## MQT Parameters - Optimization

- Setting the optimization level
  - Three different ways to achieve SQL optimization
- System level
  - DFT\_QUERYOPT configuration parameter
- BIND level
  - QUERYOPT *optimization-level* bind option parameter
- Statement level – SQL statement
  - SET CURRENT QUERY OPTIMIZATION = *host variable or number*



**The MQT will only be considered by the DB2 optimizer when the optimization level is greater than or equal to 5. The optimization level can be setup to have a system, plan/package or statement scope through different parameters.**

**The optimization level is set system wide within the Database Manager configuration parameters. The DFT\_QUERYOPT is the default optimization level set through the installation process or modified through the UPDATE CFG DB2 commands. The bind process provides a parameter so a plan or package can also override the system default. Next a statement level optimization parameter can also be set through the SET CURRENT QUERY OPTIMZATION = statement.**



## MQT Parameters - Optimization

- Query rewrite considerations
  - MQT column definitions
  - Isolation Level
  - Special Registers
    - REFRESH AGE
    - CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
      - System, User, All, or None
- Only dynamic queries or during BIND
  - Query rewrite at the query block level
  - Columns, Predicates, IN-List, GROUP BY items, Derived columns
  - EXPLAIN: Table\_Type = M



**When DB2 tries to optimize the SQL statement(s) the optimizer will consider more complex access methods and query rewrites the higher the optimization level is set. Since DB2 knows the MQTs are built from base tables it can automatically rewrite the queries to use MQT data as opposed to the base table.**

**DB2 considers the data refresh age, the process isolation level and the number of I/O and CPU instructions needed to provide the SQL query answer. These factors will determine when the DB2 optimizer will use the MQT and or the base table for every query block within the SQL.**



## Leveraging MQTs

- Have correct sorting order
  - Include all possible calculation
    - Any averages, counts or sums or partial totals

```
CREATE TABLE BEULKE.MQT_TOTS_BY_STOR_DEPT (  
  DSAL_STOR_ID, DSAL_PROD_DEPT, TOTL_DSAL_PRCE )  
AS (  
  SELECT  
  DSAL_STOR_ID, DSAL_PROD_DEPT, SUM(DSAL_DSAL_PRCE)  
  FROM BEULKE.DAILY_SALES  
  GROUP BY DSAL_STOR_ID, DSAL_PROD_DEPT  
  ORDER BY  
  DSAL_PROD_DEPT, DSAL_STOR_ID)  
DATA INITIALLY DEFERRED  
REFRESH DEFERRED  
MAINTAINED BY USER  
ENABLE QUERY OPTIMIZATION  
IN BEULKE_MQTSPACE;
```



**The MQT should include all and any possible calculations and groupings done through the SQL workload. This lets the DB2 optimizer to chose the MQT rather than the base table. Also when creating the MQTs make sure the sorting order is properly defined for the SQL workload. This will ease sorting requirements and improve buffer pool utilization.**

**The slide example creates a simple MQT for daily sales queries. MQTs can be used within views and with other MQTs to solve query workloads of all different types.**



## Leveraging MQTs

- Analysis points should be reflected in MQT
  - Store monthly sales figures

```
CREATE TABLE BEULKE.MQT_TOTS_BY_STOR_MONTH (
  DSAL_STOR_ID, DSAL_MONTH, DSAL_YEAR, TOTM_DSAL_PRCE )
AS (
  SELECT
  DSAL_STOR_ID, MONTH(DSAL_TXNS_TS), YEAR(DSAL_TXNS_TS), SUM(DSAL_DSAL_PRCE)
  FROM BEULKE.DAILY_SALES
  GROUP BY
  DSAL_STOR_ID, MONTH(DSAL_TXNS_TS), YEAR(DSAL_TXNS_TS) )
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY USER
ENABLE QUERY OPTIMIZATION
IN BEULKE_MQTSPACE;
```



**MQTs should be defined for all the analysis points that are used often in your environment. Workloads or SQL activity that must be done that sums or total figures that are done only a few times during the day need to be evaluated for being created into a MQT.**

**If the total or sum of daily totals is going to be done, you might as well capture the sum and totals within a MQT so the figures can be used by other queries and users through out the day.**



## View with a MQT

- Views with MQTs allow quick SQL access also
  - Provide further benefit without storing data
  - Determine when to store pre-calculated data

```
CREATE VIEW BEULKE.TOTS_BY_STOR_YEAR AS (  
SELECT  
DSAL_STOR_ID, DSAL_YEAR, SUM(TOTM_DSAL_PRCE) AS TOTY_DSAL_PRCE  
FROM BEULKE.MQT_TOTS_BY_STOR_MONTH  
GROUP BY  
DSAL_STOR_ID, DSAL_YEAR ) ;
```



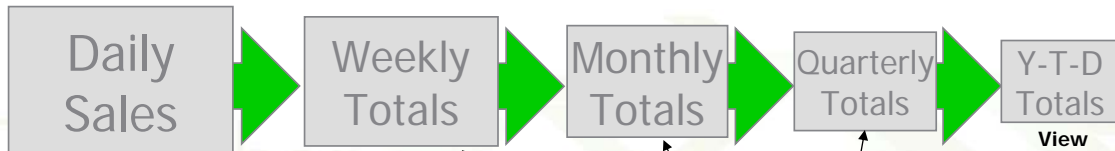
**MQTs can be combined with other tables or MQTs within views. These views can then be created to solve any extraordinary processing or unique business solutions.**

**The MQTs and the base tables can be used to provide totals or sums for any situation.**



## MQT Example

- Daily sales figures feed MQTs
  - Create additive MQTs
  - MQTs created for all analysis comparison points



- MQTs can be built from other MQTs
  - Define Quarterly Sales from Monthly Sales
- Combine MQTs through Views
  - Views over region, territory, store id etc...



Pragmatic Solution

OLTP  
Detail

33

Trans  
from today

ke@cs.com

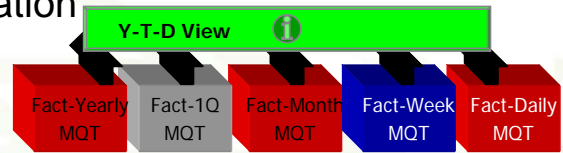
**MQTs can or should be defined for the lowest common SQL totaling or summing that is executed within the environment. These smaller MQTs can then be set up to be additive where totals can be built from other MQT totals.**

**The additive MQTs can then be quickly accessed to provide any level of sum or total across a wide variety of questions. These MQTs when combined with a view over the daily sales table offer up to the moment sales report totals.**



## MQT – 10 to 1000 times improvement!

- 5B rows per year–10 per 4k page= ½B pages
- MQT aggregates save large amounts of everything
  - Create aggregates for every possibility
    - “On Demand” information
    - Sales by department
    - Sales by zip code
    - Sales by time period – day/week/month/quarter/AP
  - All reporting and analysis areas
  - Trace usage to create/eliminate aggregates
- Total by month ½B I/Os versus 12 I/Os



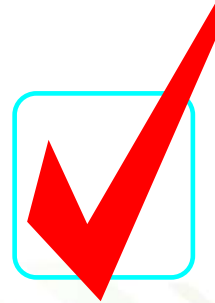
### MQT as aggregates

**Data aggregation and summaries can save a tremendous amount of I/Os and CPU. Make sure the aggregates and summaries are monitored to demonstrate and justify their creation.**



## Checklist for Performance

- **DB2 SQL continues to lead the industry**
  - Performance advantages of new SQL OLAP
  - Functions in z/OS “leapfrog” the competition
- **MDC offer some opportunities**
- **Indexing continues to get better**
- **Partitioning & parallelism for performance**
- **MQTs offer **\$\$HUGE\$\$ opportunities****
  - Must be substantial CPU and I/O savings
  - Can be used independently
  - Data refresh is convenient



**Performance is only related to how much time it takes to answer the SQL question and how much resources are used to get the answer. MQTs provide the ability to preposition data, or quickly reuse a common set of data.**



IDUG® 2006 - Europe

Session#: G01  
Data Warehouse Design and Performance

**David Beulke**

Pragmatic Solutions, Inc

DaveBeulke@cs.com



Pragmatic Solutions, Inc.

36

DaveBeulke@cs.com

