Business Intelligence Software Solutions
White paper

DB2® Information Management Software

DB2 Cube Views Teraplex Case Study

by Virginia Hoffman
and Dan Graham

August 2003

Contents

Background and goals

In April of 2003, an IBM SurfAid[1] client visited the Teraplex center to determine the effects of a large parallel IBM @server pSeries™ system configuration on their clickstream analysis workload. The client is an Internet-based media company organized into branded communities across multiple topics of importance to women. They offer interactive services, peer support, content, online access to experts and tailored shopping opportunities. Good Web site design and click-through ordering rates are paramount to this IBM e-business client, who has used the SurfAid Gold service for many years and wanted to step up to the next level of analysis.

IBM clients utilize the Teraplex Integration Centers[2] to do proof of concept, scalability and stress testing on business intelligence (BI) workloads. Unlike benchmark testing, Teraplex testing often focuses on best practices, viability of architecture and general questions of system behavior under load. For IBM's customers, the Teraplex Centers help minimize risk inherent in large BI projects, transfer skills to the clients and resolve any scalability issues before customer systems are put into production.

At the time the SurfAid Teraplex testing began, the IBM Silicon Valley Laboratory released a new product called IBM DB2® Cube Views. DB2 Cube Views is an online analytical processing (OLAP) accelerator that enhances the performance of IBM DB2 Universal Database™ when multidimensional query tools issue complex structured query language (SQL) queries. The SurfAid experts and the Silicon Valley Laboratory recognized the opportunity to "test drive" DB2 Cube Views at the Teraplex on a large, complex customer workload. Customer permission to use their data for testing was acquired under the agreement that the client remains anonymous.

Working with the SurfAid experts, the team set the following goals for the opportunistic DB2 Cube Views project:

- Measure query performance acceleration derived from DB2 Cube Views
- Identify the steps and complexity of adding on DB2 Cube Views
- Provide insights and recommendations to the laboratory
- Observe Cube Views behavior in a partitioned database configuration.

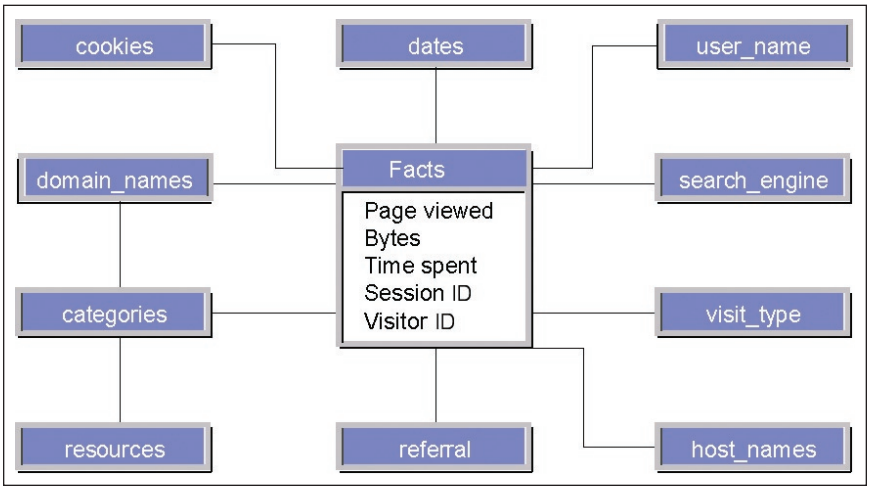
It is important to note that the testing team members were DB2, V7 experts but had no training or experience with DB2 Cube Views—we learned as we went along. Also, for some members of the team, this was the first exposure to specific DB2, Version 8 features and OLAP concepts. We believe our experiences in this project are similar to the way in which many DB2, V7 clients may encounter DB2 Cube Views for the first time.

The database, application and configuration

The customer in this study offers several Web portals and regularly uses the SurfAid Gold service to measure and enhance each portal. Their production database is composed of pre-aggregated clickstream data derived from their Web pages and arranged in a simple snowflake schema. Due to the large volume of clickstream data—several terabytes per month—the client discards most of the data quickly, keeping only the data that is directly needed for business analytics. Consequently, they aggregate critical measurements and maintain a subset of detail data from the recent period. Although starting with a mart of pre-aggregated data deviates from what might be considered the most common customer scenario entering OLAP exploration, it is nonetheless a very valid scenario and not uncommon.

The customer's main desire was to realize performance improvement on some particularly slow and troublesome queries from their production environment. A secondary desire was to be able to expose more ad-hoc query reporting to the end user. They, therefore, wanted to test the cube model theory offered by DB2 Cube Views to see if it would provide good coverage and perform well.

SurfAid collects either Web server log files or uses a small JavaScript on the Web pages to supply even richer details of consumer interaction. The data is then mined, organized, sessionized and stored in DB2. Once in DB2, advanced reporting and data mining can be applied. The database design is a snowflake schema.



This diagram shows the star-schema subset of the snowflake design.

The client brought five months of clickstream data to the Teraplex for testing. This subset of data was approximately 250 gigabytes, comprised of 1.47 billion data rows in the fact table and 1.9 million rows in the dimension tables. For prototyping purposes, the project team also maintained a one-month sample of the database with only 361 million rows in the fact table. In the next sections, we will refer to the 1-month and 5-month database sizes.

The system under test was a pSeries Model 690 (a.k.a. "Regatta") with 32 processors and 128 gigabytes of memory. Clearly, this server was much too powerful for the amount of live data we had. However, the goal was a proof-of-concept test, not a benchmark, so the pSeries system served us well. The system was partitioned into 11 logical partitions[3] comprised of a catalog partition, a small dimension tables partition, a materialized query table (MQT[4]) partition to hold the cube slices and 8 fact table partitions. The fact table partitions also held the larger dimension tables. Two IBM TotalStorage® FAStT700 Storage Server disk subsystems were used in a RAID 5 configuration. DB2, V8.1 was used with a beta code version of DB2 Cube Views. The DB2 AutoConfigure tool was used for the initial performance settings, which were modified slightly. This provided a good starting point for setting up the database configuration parameters based on the available system resources. The Teraplex server had substantially more memory than was needed for the testing. Consequently, the team purposely did not expand the DB2 buffering structures to ensure the emphasis was not on large memory caching but rather on DB2 Cube Views and true response time.

Cube modeling and testing

During the modeling phase, the project team needed to develop the best possible DB2 Cube View design from the snowflake schema. The modeling phase would have two important sub-steps:

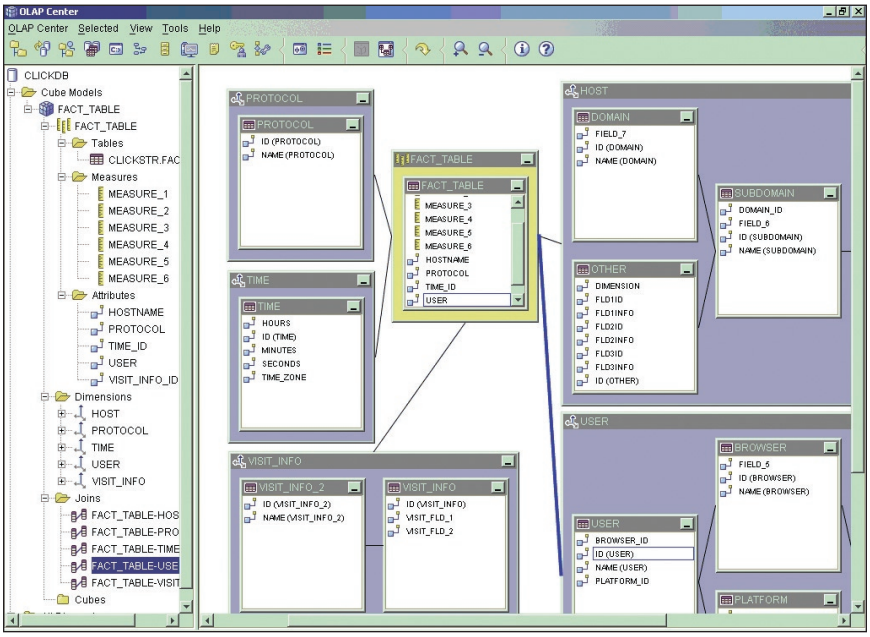
- Produce an OLAP-aware meta data definition called a cube model.
- Get an automatic recommendation for one or more MQT tables based on the cube model vis-à-vis an actual data sample.

As usual, the Teraplex Centers were full of customer activity, which necessitated the use of a smaller machine for the cube views modeling task. The project team decided to use their personal IBM ThinkPad® laptops for cube views modeling. This necessitated acquiring a subset of the clickstream database from the 1-month database to provide the data samples needed to drive the Cube Views Optimization Advisor. Consequently, a random sample of 6,000 rows from the fact table, along with the supporting dimension tables, were extracted from the 1-month database and loaded onto the ThinkPad laptops.

Producing a cube model

Data models are often comprised of relational tables that represent facts, dimensions and explicit referential constraints between the tables. However, relational databases lack knowledge of hierarchies, joins and measures needed by OLAP reporting. Much of this knowledge is contained in popular BI end user tools or sophisticated database design tools.

Cube Views provides a Quick Start path that automatically translates an existing relational star schema into a meta data model definition, saving the database administrator (DBA) considerable labor. After the Quick Start, the facts, measures, dimensions, attributes and joins are all recorded in meta data and nicely presented to the DBA in interactive graphical displays. Even special situations like multiple join paths between tables are handled automatically. In this project, the database designer did not use a database design tool and was particularly delighted to find the easy-to-use interactive displays provided by DB2 Cube Views.



This diagram depicts the interface for DB2 OLAP Center.

The client needed measures added to the model to specifically address "count distinct" reporting needs. Count distinct is a non-distributive measure, meaning it cannot be naturally aggregated across data slices, like sums and simple counts can. The existing SurfAid schema design required count distinct processing in almost all of its measures, which required additional cube designs in the cube model. Both DB2 Cube Views and the IBM DB2 Universal Database Enterprise Server Edition (ESE) database query optimizer handled this well.

Hierarchies were defined for each of the dimensions. In some cases the fields comprising the hierarchy were all from a single table, and in other cases the fields were from multiple tables.

The DB2 Cube Views OLAP Center Optimization Advisor takes the cube model and recommends one or more MQTs to best fit the query requirements. The model-based recommendation offers these advantages:

- DBA knowledge of the desired querying style is leveraged to produce an MQT that best supports a given access pattern. DB2 Cube Views can recommend MQTs that are optimized for:
  1. Drill-down analysis: a query pattern that starts at the logical "top" of the cube and progresses step-wise down the dimensional hierarchies.
  2. Report-style analysis: ad-hoc queries that may be targeted at any region of the logical cube.
  3. Drill-through or hybrid analysis: a query pattern that is split between compact summarized multidimensional OLAP (MOLAP) structures (DB2 OLAP Server) and lower-level relational data. In this case, DB2 Cube Views optimizes the MQT at the lower levels of summarization.
  4. Extract optimization, in which relational source data is optimized for the loading of MOLAP summary structures (e.g., DB2 OLAP Server).

Finding: An important learning experience in our project was the need to perform the Optimization Advisor step once very early in the modeling process to find any missing prerequisites in the base tables.

The Optimization Advisor factors this knowledge into its recommendations, making content trade-offs based on this. The MQT size can be limited to fit a targeted disk usage, with DB2 Cube Views performing automatic trade-offs to achieve that target.

- DB2 Cube Views uses database statistics and data sampling throughout to optimize to actual customer data demographics.
- The DBA gets MQT index recommendations along with the recommended MQTs.

The Optimization Advisor performs an internal validation of the existing database schema to see if it conforms to the query reroute rules needed for MQT run-time usage. The Optimization Advisor will quickly locate missing referential constraints that might prevent DB2 from rerouting queries to the generated MQT. Sometimes, these rerouting rules may be difficult for DBAs to detect on their own, especially with a large number of tables, keys and columns. In our case, the customer schema lacked the "not null" clause on several columns in the base tables. The customer application itself had already enforced the "not null" constraint. Consequently, adding these to the fact and dimension tables was easy and had no adverse effect. Generalizing from this discovery, implicit referential constraints will often need to be explicitly added to star and snowflake schemas in order to allow DB2 to use the query rewrite with MQTs. Adding the "not nulls" and running the Optimization Advisor again showed we had properly enhanced the base tables for use with DB2 Cube Views. Explicit referential constraints already existed between fact and dimension tables, and between dimension tables and their snowflakes, with four hops from fact to the outermost snowflake.

No adjustments to the Quick Start derived joins were needed for the cube views schema modeling. The automatically derived joins from the Quick Start were correct and did not need adjusting for inner or outer join conditions, or for cardinality. Clearly, the Quick Start autonomic service had served us well.

Finding: Either do Optimization Advisor with no sampling against an imported RunStats statistics database, or do it with sampling against a statistics-correct database. But don't mix the two. Be sure all the statistics for all tables are up to date where the Optimization Advisor processing is executing. This includes dimension tables, not just fact tables.

The resulting cube model consisted of:

- One fact table.
- Eight measures—two count distinct, three averages, one count(*), two sums.
- 13 logical dimensions made up of 21 physical tables; a few tables are members of more than one logical dimension.
- 13 joins.
- 23 hierarchies with up to 4 levels.

Typically, the DB2 Cube Views Optimization Advisor will recommend one or two MQTs, but in our case DB2 Cube Views recommended a total of 20 different MQTs. This unusual situation was caused by numerous count distinct measures in the model. The designers felt it would be easier to manage two "cubes" within the "cube model"—one aimed at count distinct measures and another for general aggregates. One cube view design was targeted at drill-through queries while the other was aimed at performance gains for a broader range of drill-down queries. Second, it was easier and more productive with two designers working in independent locations to produce two independent models rather than try to constantly exchange model designs. The trade-off was that the Optimization Advisor had to be run twice on the 5-month data, once for each model. A quick check assured the team that the DB2 optimizer would reroute incoming SQL appropriately to any MQT derived from either cube within the same database.

Briefcase cube views

Most of the cube modeling with the DB2 Cube Views occurred on the laptop, not on the Teraplex machine. The laptop had a reduced sample of customer data—only 6,000 rows—and just enough dimension table data to support the 6,000 rows. For the modeling steps and the initial optimizations, this was a nice setup for query testing and the initial creation of the MQTs. We were even able to test query rerouting using SQL EXPLAINs and other features on the ThinkPad.

To ensure the ThinkPad development produced relevant MQTs, we imported DB2 database statistics from the 1-month customer database. These database statistics reflected a much bigger database than the 6,000 rows loaded onto the laptop. This was possible because the Optimization Advisor offers a choice of either "with sampling" or "not sampling." Selecting the not sampling option means that only the cube model definition and the DB2 statistics will be examined as input to generate the resulting MQT. This method will produce a serviceable MQT, but will not necessarily lead to the best design because it ignores the data demographics. The "with sampling" approach used on the large Teraplex server caused OLAP Center Optimization Advisor to examine database statistics plus data values, producing the optimum MQT results.

Running the OLAP Center Optimization Advisor produced an excellent MQT design and populated it successfully on the ThinkPad database. When we ran SQL against the ThinkPad MQT, the EXPLAIN process did not show anything unusual or unexpected. The team was able to do a number of tests and learn a lot about DB2 Cube Views in this configuration.

Next, we exported the ThinkPad MQT design to the much larger Teraplex database. For example, we copied the data definition language (DDL) from the DB2 Cube Views Optimization Advisor to the big pSeries server. Unfortunately, the design created by the OLAP Center Optimization Advisor using "borrowed" database statistics and a tiny database was now inappropriate for the larger database. Sampling a 6,000-row database on the ThinkPad is not the same as sampling a 360 million-row table on the Teraplex server. Consequently, on the ThinkPad, the Optimization Advisor built an MQT that was nearly as large as the entire 6000-row database because we did not limit the MQT disk space allocation. However, the guidelines given by the DB2 Cube Views manuals suggest the generated MQT should be an order of magnitude smaller than the production data if done correctly. Because our ThinkPad DB2 database was so small, no one noticed that the generated MQT was the same size as the entire database, less than a megabyte.

Finding: Contemplating what we had just experienced produced an epiphany: DB2 Cube Views knew more about how to use the database statistics, the data and how to generate the MQTs than the team did. Without DB2 Cube Views, an expert DBA would have to spend many hours of trial and error experimentation to get an ideal MQT or "cube slice." Optimizing for the amount of disk space alone would be a daunting task requiring tremendous skill and testing. With DB2 Cube Views and index advisor, DBAs can let the system sort out the parameters and designs for them. This made the learning experience for those of us new to DB2 V8 or OLAP concepts a lot easier. In retrospect, the team now knows that DB2 autonomics is a powerful ease-of-use substitute for the expertise they didn't have when starting the project.

As soon as we moved the MQT design to the Teraplex server, DB2 did what it was told and started aggregating the entire 1-month database into an MQT of equal size. What worked well on a small laptop computer was now an extremely long-running aggregation and roll-up of the production database. In hindsight the mistake was obvious, but it was not until the Teraplex MQT generation began taking too long that the team caught the error. While it was possible to do a lot of design and development on smaller test machines, it was also necessary to run the final Cube Views Optimization Advisor on the production database. Using the correct "with sampling" database and statistics yielded much better MQTs than non-sampled recommendations.

Once the above mismatch was corrected and the Optimization Advisor performed on the Teraplex machine against the imported cube model, there was one more thing lacking. In a rush to get back on track, the team had forgotten to limit the amount of disk space allowed for the generated MQTs. Again, the DB2 OLAP Center Optimization Advisor built the best possible MQT—with hundreds of gigabytes of disk—which again produced an excessively long-running aggregation of the entire database. We halted the process and limited the OLAP Center Optimization Advisor to 50 gigabytes of disk. DB2 Cube Views then produced the ideal "order of magnitude smaller" MQT.

This leads to the obvious conclusion that the DBA should be familiar with MQT maintenance and remember to get new OLAP Center Optimizer Advisor recommendations if the data or available storage changes substantially. It is appropriate to rerun the advisor any time a significant server upgrade is done or perhaps a couple of times a year if the database size is expanding. Like any high-performance system, occasional maintenance and tuning keeps it performing its best. Fortunately for us, once we moved the cube view designs from the laptops to the Teraplex server, OLAP Center Optimizer Advisor used the statistics from the larger partitioned database to produce the correct MQT slices. Using the DB2 logical hash partitioning was transparent to us for most of the Teraplex experience.

SQL testing with MQTs

Once the various MQT tables were generated, we ran EXPLAINs to see if the SQL queries would reroute to the new MQTs. The first few EXPLAINs did not show the anticipated rerouting on 20 out of 26 queries. After analyzing the MQT versus the SQL EXPLAINs, it was clear that the MQT hierarchies did not match the queries. Also, we determined that two "measures" were missing from the cube model. The team was getting quite a lesson in OLAP and multidimensional theory at this point. We adjusted the cube model to include the correct OLAP hierarchies and measures and regenerated the MQT. Once this was done, only three queries would not reroute, but now it seemed this was a DB2 Optimizer decision versus a missing hierarchy. Measures must be in the SQL queries as well as in the MQT DDL specification. Ideally, this is done by putting the measures into the cube model and running the OLAP Center Optimization Advisor. Said another way, your cube model and SQL should match, or you won't get the performance gains.

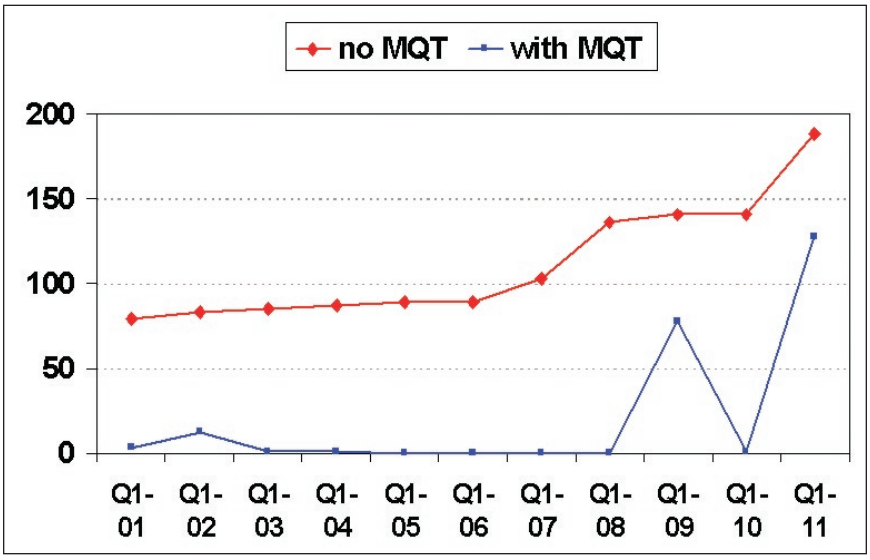
Performance results

The testing was done with a small database (1-month) and the full database (5-month) that the client supplied. The 1-month fact table was 361.6 million rows, whereas the 5-month fact table was 1.469 billion rows.

Benchmarking for the high-water performance measurement was not the primary goal of this proof-of-concept testing. Consequently, there was no effort to ensure that the 1-month and 5-month tests could be compared. For example, one MQT set used time dimensions in seconds while the other used whole minutes, radically changing the number and type of the time dimension. Because of the changing nature of the MQT designs between one and five months, no conclusions or comparisons should be drawn between the tests. Furthermore, when comparing non-MQT performance to DB2 Cube Views MQT-based query results, differences less than five percent are insignificant and within the margin of measurement error.

Having spent the design time to ensure that most queries would reroute to an appropriate MQT, our subsecond and order-of-magnitude faster response times became predictable.

1-month database tests

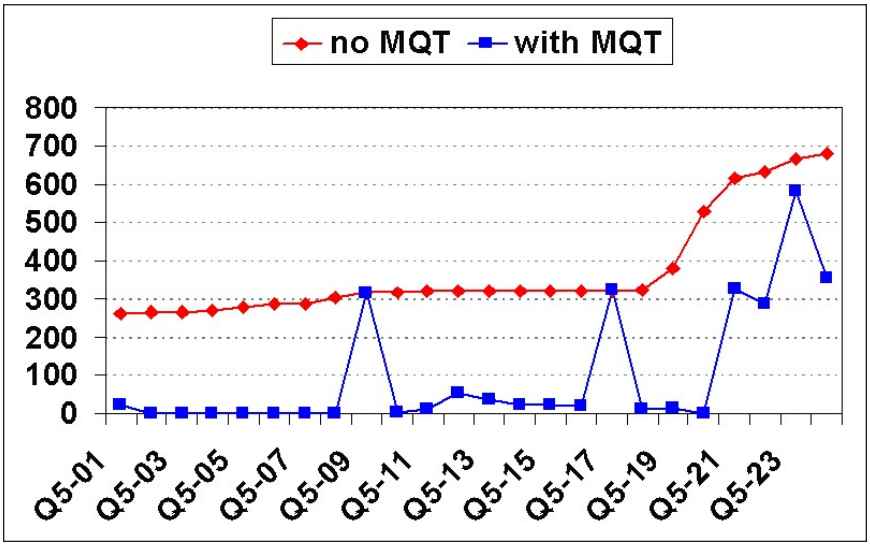


For most queries in this test, the elapsed time was reduced significantly.

Using the 1-month database queries, all but a couple of SQL requests were completely satisfied by the DB2 Cube Views MQTs. Most queries achieved an elapsed time reduction between 86 and 99 percent. Two queries achieved only a 30 percent elapsed time reduction, primarily because the measures and generated MQT did not contain complete aggregated results for those queries. That is, the limit on disk usage as well as design constraints eliminated the generation of an MQT that would solve that query request completely. Instead, the query partially used the MQT but also had to retrieve detailed data records to finish the user request. Some queries will use a multi-part SQL expression where some of it is solved by the MQT but the rest is not.

Finding: Not all queries will reroute to the MQTs, nor should they. DB2 Cube Views is meant to accelerate a multitude of queries within the constraints of disk space allocated.

5-month database tests



Most queries in this test rerouted to the MQT generated by DB2 Cube Views, substantially reducing elapsed time.

The 5-month database was more representative of the customer workload than the 1-month database. Like the 1-month database, a majority of queries were able to reroute to the DB2 Cube Views-generated MQT, producing 90 to 99 percent reductions in elapsed time. A few queries (Q5-22, Q5-24, etc.) were able to partially leverage the MQTs but also had to go back into the snowflake schema to locate additional information yielding more modest 50 percent elapsed time improvements. A few queries (Q5-9, Q5-18) got no benefit at all from the DB2 Cube Views generated MQTs. However, this is not necessarily a "bad" situation—DB2 simply uses traditional techniques to achieve good relational data warehousing query performance. If a DBA wanted to accelerate all queries with DB2 Cube Views, a few visits to the OLAP Center Optimization Advisor and a lot more disk space would do it. In these cases, the limits of disk space allowed for MQTs as well as the removal of measures and constraints eliminated the generation of MQTs to the depth that would have accelerated all queries.

Changing many 4-, 5- or 10-minute queries to subsecond response time is clearly a success. The Surf Aid team can now process larger amounts of data (> 5-month sizes) or deliver more interactive customer experiences.

Conclusions: what we learned

The project was an excellent learning experience for the team, accelerating our skills as well as the queries. In retrospect, the faster performance from DB2 Cube Views was just one of the discoveries and perhaps the most predictable one. Some of the other conclusions we reached were:

- *Consistent response time improvements.* When the cube model is able to generate an MQT that queries will reroute to, the resulting response times decreased 90 to 99 percent. While some will get only a 30 to 50 percent reduction, they were worth the effort as well. We anticipate clients will use this acceleration to either add more concurrent users or to delay server upgrades with the DB2 data warehouse.
- *64-bit is wonderful.* It's easy to forget the joys of 64-bit IBM AIX® when you work on Microsoft® Windows® servers daily. Sort heaps, buffer caching, even disk files could be substantially larger with 64-bit operating systems.
- *DB2 Cube Views works well with partitioned DB2 tables.* DB2 partitioning facilities made it fairly easy to spread the fact and dimension tables across the partitions. Minimal time was spent designing or managing the multi-partition DB2 configuration. It was easy to assign the DB2 Cube Views MQTs to their own partition and get outstanding performance.
- *ThinkPad R&D:* Developing cube views on our laptop computers worked well. Transferring the designs to the central server was not difficult once we got in the habit of running OLAP Center Optimization Advisor each time.

- *Autonomics — we are now believers.* Every step of the project, there was a DB2 advisor making our job easier. It was straightforward to run the index advisor, load a little data, check for skew, then run the complete data load with confidence. The auto configuration advisor examined the CPUs, the memory, disk controllers, etc. and gave us good recommendations for the number of page cleaners, pre-fetchers, etc. needed to provide an optimum BI environment. But the best autonomic capability was DB2 Cube Views itself. It did the data sampling, the statistics analysis, observed the relationships between the data and the multidimensional design, balancing all this vis-à-vis the disk allotted and the depth of the OLAP hierarchy. DB2 Cube Views eventually taught us how to build the best MQTs.

DB2 Cube Views worked "as advertised." The clickstream analysis SurfAid experts are now looking at ways to implement DB2 Cube Views for faster multidimensional reporting for their clients.

Acknowledgements
The various phases of this project involved participants from IBM Austin, Dallas and Silicon Valley Laboratories. The testing was done at IBM's Teraplex Center in Poughkeepsie, New York. Many thanks to these outstanding contributors:

IBM Austin: Virginia Hoffman, James Platt
IBM Dallas: Steven Howard, Brian Shorter
IBM Silicon Valley: Michael Alcorn, Nathan Colossi, John Poelman, Marty Yarnall
Teraplex: Tom Moskalik, Surendra Parlapalli, Tina Tarquinio
DB2 Benchmarking: Acie Nobles, Ron Sherman

Appendix A - Performance tables

1-Month Query Comparisons

Query	no MQT seconds	with MQT seconds	rows in result	% speed up
Q1-01	79.1	3.3	60558	96%
Q1-02	82.8	12.0	1	86%
Q1-03	85.2	0.6	431	99%
Q1-04	86.9	0.8	431	99%
Q1-05	89.0	0.1	256	100%
Q1-06	89.3	0.1	256	100%
Q1-07	102.9	0.1	431	100%
Q1-08	135.9	0.2	432	100%
Q1-09	140.7	77.8	1091	45%
Q1-10	141.1	0.4	50	100%
Q1-11	188.1	127.7	200	32%
Q1-12	1155.3	804.6	1	30%

1-Month Query MQT Generation

Summary table	seconds	Minutes (sec/60)	MQT rows
Mqt1	3017.6	50.3	13,877
Mqt2	1831.0	30.5	8,954
Mqt3	4611.0	76.9	856
Mqt4	1920.0	32.0	5,834,738
Mqt5	1110.0	18.5	2,320,225
Mqt6	752.9	12.5	1,947,650
Mqt9	209.0	3.5	738
Mqt10	285.0	4.8	7,491
Mqt15	408.0	6.8	31

5-Month Query Comparisons

Query	no MQT seconds	with MQT seconds	rows in result	% speed up
Q5-01	262.6	22.7	11	91.3%
Q5-02	265.3	0.2	431	99.9%
Q5-03	266.8	0.1	431	99.9%
Q5-04	269.8	0.2	256	99.9%
Q5-05	278.9	0.2	256	99.9%
Q5-06	284.2	0.2	1	99.9%
Q5-07	286.8	0.2	431	99.9%
Q5-08	302.3	0.1	432	99.9%
Q5-09	315.9	317.0	1	0%
Q5-10	317.3	2.7	60558	99.1%
Q5-11	317.7	11.0	26	96.5%
Q5-12	318.3	54.7	11468	82.8%
Q5-13	318.9	37.3	24	88.3%
Q5-14	319.1	23.9	24	92.5%
Q5-15	319.6	24.1	25	92.4%
Q5-16	320.0	20.6	24	93.5%
Q5-17	320.4	321.7	11	0%
Q5-18	322.0	11.6	670	96.3%
Q5-19	379.3	12.7	26	96.6%
Q5-20	529.1	1.1	50	99.7%
Q5-21	616.4	325.2	1091	47.2%
Q5-22	632.8	287.2	50	54.6%
Q5-23	664.7	581.4	200	12.5%
Q5-24	679.4	352.6	200	48.1%
Q5-25	2016.3	1911.9	2445	5.1%
Q5-26	5032.8	2066.8	1	59.9%

5-Month Query MQT Generation

5 month	Seconds	Rows
5mqt1new	13361.0	57,496
5mqt2new	2915.0	32,396
5mqt3new	7355.0	3,643
5mqt4new	3058.0	19,170,491
5mqt5new	2093.0	6,637,728
5mqt6new	1817.0	8,313,075
5mqt9new	1058.0	3,037
5mqt10new	8144.0	91,128
5mqt15new	1885.0	134
5mqt52	4259.00	382,550
Totals	45945.0	
Fact table		1,469,518,976

IBM United States
Software Group
Route 100
Somers, NY 10589
U.S.A.

Printed in the United States of America
08-03

[1] SurfAid is an IBM Business Consulting Service that performs Internet clickstream analysis via out-sourcing of the data. Analysis can be done once, weekly, monthly, etc., delivering interactive custom reports online. The analysis is used to improve consumer click-through ordering success rates or Web site usability. See http://surfaid.dfw.ibm.com/web/home/index.html

[2] **ibm.com**/solutions/businessintelligence/teraplex/index.htm

[3] DB2 extracts additional scalability from a large SMP by using logical partitions. Each partition is a separate set of hash domains, disk units and buffer pools. While all CPUs in the server can be allocated to any task, each partition is dedicated to a "logical" node or hash segment of the database. Data retrieval and update requests are decomposed automatically into sub-requests, and executed in parallel among the applicable database partitions transparent to users. This technique is used to partially mimic a hardware cluster and get additional scalability without incurring the additional labor of a true cluster design.

[4] An MQT design is the primary output of DB2 Cube Views. An MQT — materialized query table — is a DB2 maintained summary table hidden from the end user but usable by the SQL nonetheless because of the query rewrite capability of DB2.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

♻ Printed in the United States on recycled paper containing 10% recovered post-consumer fiber.

GC18-9064-00

e business software