

Data Warehousing With DB2 for z/OS . . . *Again!*

By Willie Favero

Decision support has always been in DB2's genetic make-up; it's just been a bit recessive for a while. It's been evolving over time, so suggesting DB2 for z/OS for your database warehousing shouldn't be a surprise. Today, DB2 for z/OS is a real warehouse contender and you should be giving it some serious consideration—again. Recall the original DB2 announcement materials from June 7, 1983, which said: "DB2 offers a powerful new data base management system for use where business and application requirements and data structures are subject to frequent changes. ... As such, DB2 may be used to address decision support systems as well as traditional application areas."

Decision Support Systems (DSS) are still actively used today; they contributed to the need for a data warehouse.

When DB2 became generally available in 1985, it was initially difficult to convince customers to use it for Online Transaction Processing (OLTP). At that point, many customers thought DB2's only purpose was decision support and information centers. Decision support prompted talk about warehousing and Business Intelligence (BI). It took a few years for DB2 to evolve into the transactional support database manager it is today. DB2 is more ready for your data warehouse today than ever before.

DB2's recently renewed presence in the warehouse world isn't coincidental, nor is it just DB2 trying to re-establish itself. Warehousing has been changing. Instead of determining what happened in the past, customers want to use all that information to make immediate decisions today. Instead of making information kept in the warehouse

available to just a few, it's being leveraged by larger numbers of customers, including their customers. The focus is getting the information to the correct person when that person needs it. Information must arrive rapidly and accurately. It's starting to sound a lot like OLTP, isn't it? What better DBMS than DB2 for z/OS to satisfy these modern challenges.

Before evaluating DB2's usefulness for warehousing, let's establish some terms. *Webster's New Millennium Dictionary* defines a data warehouse as "a computer system that collects, stores, and manages large amounts of data for a company, to be used for business analysis and strategy." DB2 for z/OS definitely satisfies this definition. That same dictionary defines BI as "a broad range of applications and technologies for gathering, storing, analyzing, and providing



LIGHTNING SPEED.

No matter how large the task!



**Along with speed comes the 100% FOCUS on DB2,
FLEXIBLE automation, and scalability to handle any size DB2 objects.**

CDB Software helps enterprise IT discover more performance and availability from their complex DB2 z/Series environments. Through advanced processing techniques, CDB Software solutions reduce costs while improving reliability and efficiency. So, no matter how much data you manage, you remain flexible and in control.

Call today for your free analysis!

P. O. Box 420789 • Houston, TX 77242-0789 • 800-627-6561 • www.CDBsoftware.com



access to data to help make business decisions.” DB2 helps with BI, but needs additional application support for this area and gets it from Cognos (an IBM company), IBM DataQuant, and DB2 Alphablox.

Your data warehouse is how you store the data; BI is how the data is analyzed. Both are integral parts of a comprehensive solution. BI applications serve no purpose if they have nothing to analyze and a data warehouse is of little use without the tools to interrogate the information it contains.

Finally, this article assumes that the terms data mart, Operational Data Store (ODS), and DSS are all similar to the data warehouse concept and are sometimes used in conjunction with, and sometimes in place of, the term data warehouse.

DB2's Effect on Your Warehouse

Enhancements in DB2 Version 8 support data warehousing and associated analytics. DB2 9 for z/OS could improve your warehouse experience even more. The latest version continues a tradition of progress that's worth briefly reviewing.

DSS support has been improved in some way with every release of DB2. Here are some highlights:

- DB2 V2.1 introduced Resource Limit Facility (RLF), which lets you control the amount of CPU that a resource, in this case a query, can actually use. This is for dynamic SQL that will probably make up most of the warehouse SQL workload. This can be critical in controlling system resources, and RLF also can help you control the degree of parallelism obtained by a query.
- DB2 V3 delivered compression, which still has a major, immediate effect on data warehousing. Enabling compression for tablespaces can yield a significant disk savings. In testing, numbers as high as 80 percent have been observed, though your mileage will definitely vary.
- DB2 V3 introduced I/O parallelism. With this first flavor, multiple I/O could be started in parallel to satisfy a request.
- DB2 V4 introduced CP parallelism, which allowed a query to run across two or more CPs. A query could be broken into multiple parts and each part could run against its own Service Request Block (SRB), performing its own I/O.

- DB2 V4 also introduced data sharing, which can be an asset in a warehouse environment. It allows access to the operational data by the warehouse and analytics yet still lets you separate those applications into their own DB2, reducing the chances of the warehouse activity impacting operational transactions.
- DB2 V5 introduced sysplex query parallelism, which runs within a parallel sysplex and requires DB2's data sharing. A query is run across multiple CPs on multiple Central Electronic Complexes (CECs) in the parallel sysplex. Although there's some additional CPU used for setup when DB2 first decides to run a query in parallelism, there's a correlation between the degree of parallelism achieved and the elapsed time reduction. There also are DSNZPARMs and bind parameters that need to be set before parallelism can be used.

With those as highlights in DB2's evolution, let's review in greater detail two key concepts, compression and parallelism:

Compression: DB2's compression is specified at the tablespace level, is based on the Lempel-Ziv lossless compression algorithm, uses a dictionary, and is assisted by the System z hardware. Compressed data also is carried through into the buffer pools. This means compression could have a positive effect on reducing the amount of logging you do because the compressed information is carried into the logs. This will reduce your active log size and the amount of

archive log space needed. Compression also can improve your buffer pool hit ratios. With more rows in a single page after compression, fewer pages need to be brought into the buffer pool to satisfy a query's getpage request. One of the additional advantages of DB2's hardware compression is the hard speed. As hardware processor speeds increase, so does the speed of the compression built into the hardware's chipset.

When implementing a data warehouse, the size is often considered problematic, regardless of platform. DB2's hardware compression can help address that concern by reducing the amount of disk needed to fulfill your data warehouse storage requirements.

Parallelism: One method of reducing the elapsed time of a long-running query is to split that query across more than one processor. This is what DB2's parallelism does. Parallelism allows a query to run across two or more CPs. A query is broken into multiple parts, with each part running under its own Service Request Block (SRB), and performing its own I/O. Although there's some additional CPU used when DB2 first decides to take advantage of query parallelism for its setup, there's a close correlation between the degree of parallelism achieved and the query's elapsed time reduction. There also are DSNZPARMs and bind parameters that need to be set before parallelism can be used.

Star Schema: There's a specialized case of parallelism called a star schema—a relational database's way of representing

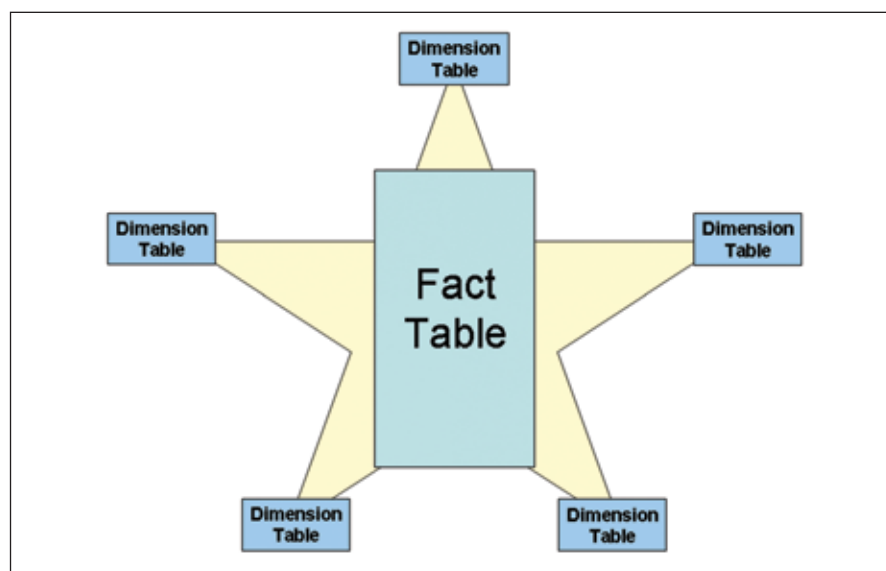


Figure 1: A Star Schema

multi-dimensional data, which is often popular with data warehousing applications. A star schema is usually a large fact table with lots of smaller dimension tables (see Figure 1). For example, you might have a fact table for sales information. This sales table would hold most of your data. The dimension tables could represent products that were sold, the stores where those products were sold, the date the sale occurred, any promotional data associated with the sale, and the employee responsible for the sale. Using star joins in DB2 requires enabling the feature through a DSNZPARM keyword. You also should check a few other ZPARMs before using star joins because they can affect a star join's performance.

Data Warehousing and Today's DB2

Many enhancements in DB2 V8 could directly impact a warehouse implementation; here are the ones that will have a positive effect on your data warehouse and application analytics:

- Clustering decoupled from partitioning
- Indexes created as deferred are ignored


- by DB2 optimizer
- Fast cached SQL invalidation
- Automatic space management
- Statements IDs of cached statements as input to EXPLAIN
- Long-running, non-committing reader alerts
- Check In (CI) size larger than 4KB
- Multi-row INSERT/FETCH
- REOPT(ONCE) to reduce host variable's impact on access paths
- Index-only access for VARCHAR columns
- Backward index scan
- Distributed Data Facility (DDF) performance enhancements
- Up to 4,096 partitions
- Longer table and column names
- SQL statements up to 2MB
- Sparse index for star join
- More tables in join
- Common table expressions
- Recursive SQL
- Indexable unlike types
- Materialized Query Table (MQT).

Let's look at some of these in more detail:

Backward index scan: Indexes are a

huge performance asset to a data warehouse. Having the ability to read an index backward lets you avoid building both an ascending index and a descending index structure. Reducing the number of indexes improves the cost of doing inserts and deletes. Every insert or delete must update every index on the table being changed. Backward index scan also can reduce the cost of doing updates if the update occurs against a column participating in the index. In addition, backward index scan reduces the amount of disk storage required to build all those indexes. With backward index scan, you will need to use only one index; you previously needed two.

Multi-row FETCH and INSERT: This enhancement lets you read or insert multiple rows with a single SQL statement via an array. This reduces the CPU cost of running a FETCH or INSERT. This feature is completely usable in distributed applications processing using Open Database Connectivity (ODBC) with arrays and dynamic SQL. This offers significant performance advantages; it could



XTINCT. Permanent DASD and TAPE data eradication.

- Provides various speed and security options
- Full Audit trail and comprehensive reports to satisfy regulators
- Meets NIST guidelines for 'cleaning' and 'purging' data
- EAL certification pending
- Satisfies requirements under Sarbanes-Oxley, HIPAA, HSPD-12, Basel II, SOX, Gramm-Leach-Bliley and all other Federal data security and privacy laws

TRY BRINGING THIS BACK TO LIFE.
XTINCT. Secure data erasure.
 Once it's gone...it's gone forever.

Dino-Software
 www.dino-software.com

Copyright 2008 Dino Software Corporation. All rights reserved.

increase the performance of FETCH processing by 50 percent and INSERT processing by 20 percent. In fact, in some customer testing, this feature averaged 76 percent improvement for FETCH and 20 percent improvement for INSERT. Improving INSERT performance and reducing INSERT CPU consumption could be a significant help to your Extract, Transform, Load (ETL) processing.

Indexable unlike types: Mismatched data types can now be stage 1 and indexable. This is huge for those applications that don't support all data types available in DB2 for z/OS.

Sparse index in memory work files: For star join processing, sparse indexes can use many work files. DB2 V8 will attempt to put these work files in memory. This can result in a significant performance improvement for warehouse queries using star joins.

More partitions (4,096) and automatic space management: Growth is one of those inherent qualities of a warehouse, something you just expect to happen. With 4,096 partitions, a warehouse could grow to 16TB for a 4KB page size; 128TB for 32KB page. This is for just one table. DB2 V8 also gives you automatic space management, the ability to let DB2 manage your primary and secondary space allocations. With DSNZPARM MGEXTSZ activated, DB2 will manage the allocation of a tablespace's extents, ensuring that the tablespace can grow to its maximum size without running out of extents.

MQTs: Aggregate data can be important for warehousing. Summary information can be stored in an MQT and in V8, and the optimizer can make the decision to use the MQT in place of a query performing the summarization inline. This can significantly improve the performance of a warehouse query doing aggregations.

Your data warehouse and application analytics also will benefit from the other V8 changes previously listed but not detailed here.

DB2 9 for z/OS delivers more changes that will directly impact your warehouse and application analytics, including:

- New row internal structure for faster VARCHAR processing

DB2 is more ready for your data warehouse today than ever before.

- Fast delete of all the rows in a partition (TRUNCATE)
- Deleting first n rows
- Skipping uncommitted inserted/updated qualifying rows
- Index on expression
- Dynamic index ANDing
- Reduce temporary tables materialization
- Generalizing sparse index/in-memory data caching
- Clustering decoupled from partitioning
- Indexes created as deferred are ignored by DB2 optimizer
- Fast cached SQL invalidation
- Statements IDs of cached statements as input to EXPLAIN
- Universal tablespaces
- Partition-by-growth as a means to remove non-partitioned tablespace size limit
- Implicit objects creation
- Clone tables
- MERGE statement
- Identifying unused indexes

- Simulating indexes in EXPLAIN (Optimization Service Center)
- More autonomic buffer pools tuning for Workload Manager (WLM) synergy
- Resource Limit Facility (RLF) support for end-user correlation
- RANK, DENSE_RANK, and ROW_NUMBER
- EXCEPT, and INTERSECT
- pureXML.

Let's look at some of these in detail:

Universal tablespace: This is at the top of my DB2 enhancements and data warehousing lists. Consider the sometimes unpredictable but expected growth of a warehouse and the high possibility that many tables could be frequently refreshed. A universal tablespace is a cross between a partitioned tablespace and a segmented tablespace, giving you many of both of its parents' best features. When using a universal tablespace, you get the size and growth of partitioning while retaining the space management, mass delete performance, and insert performance of a segmented tablespace. It's kind of like having a segmented tablespace that can grow to 128TB of data, assuming the right DSSIZE and right number of partitions are specified, and that also gives you partition independence.

Index compression: Your first line of defense against a warehouse performance problem, after a well-written query, is creating indexes, lots of indexes. With warehousing, and for some types of OLTP, it's possible you could use as much, if not more, disk space for indexes than for the data. DB2 9 for z/OS index compression can make a huge difference when it comes to saving disk space. The implementation of index compression is nothing like data compression. It doesn't use a dictionary and there's no hardware assist. However, the lack of a dictionary could be a plus. With no dictionary, there's no need to run the REORG or LOAD utilities before compressing your index data. When compression is turned on for an index, key and Record Identifier (RID) compression immediately begins.

SQL: Two new SQL statements, MERGE and TRUNCATE, can be highly effective when used with a data warehouse. Merge, sometimes called "upsert," lets you change data without needing to know if the row already exists. With

MERGE, if it finds an existing row, it updates it. If it doesn't, then it performs an INSERT. No more doing a SELECT first to see if the row exists or doing an INSERT or UPDATE and checking to see if it failed. TRUNCATE is an easy way to remove all the rows from a table with a single SQL statement. It's especially handy if you're using DELETE triggers. Also, there's now an APPEND option on the CREATE/ALTER table that tells DB2 to ignore clustering during INSERT processing. This should improve INSERT performance by eliminating the need to figure out where the row should go and just placing it at the end of the table.

Clone tables: Tables can often be completely replaced on a weekly, even daily, basis in a warehouse. Replacing a table can cause an outage, even if that outage seems short. Clone table support in DB2 9 gives you an easy way to create a replacement table while still accessing the original table and using a command to switch which table SQL actually accesses. The concept is a lot like an online LOAD REPLACE, something unavailable in DB2.

RANK: DB2 9 gives you a little bit of Online Analytical Processing (OLAP) functionality with RANK, DENSE_RANK, and ROW_NUMBER. When used in an SQL statement, they'll return the ranking and row number as a scalar value. Rank is the ordinal value of a row in a defined set of rows. You can specify that the result be returned with (RANK) and without (DENSE_RANK) gaps. ROW_NUMBER is a sequential row number assigned a result row.

Index on expression: Sometimes, a warehouse query requires changing a column in the WHERE clause via a function or possibly a calculation; some kind of predicate that would preclude an index access for that column or make that predicate stage 2—affecting your query's performance. With index on expression, you can build the index on that expression, and because of the match, get an index access. This could be a real performance boost for some warehouse and analytic queries.

There are many more new features delivered in DB2 Version 8 and DB2 9 that could significantly enhance a warehouse application. Though the functions and features just discussed can be especially handy for a warehouse and its application analytics, they do benefit

anyone using DB2 for z/OS.

Data Archiving

Size is always a major issue for a data warehouse. Warehouses can range in size from a few hundred gigabytes (GB) to dozens of terabytes (TB).

We've already discussed how DB2 can help with your data warehouse's growth by taking advantage of universal tablespace's partition-by-growth feature in DB2 9. You also should consider the need for timely data in your warehouse. How current does it have to be? If you need only the most current information for the last six months for your analysis, but you'd like to have the last two years available for the occasional special processing, don't keep it all in the active portion of your warehouse.

Consider using some form of archiving to remove the less frequently used data. You could use a secondary set of archive tables that you move the data into and have the option to include in your queries only when necessary. You could use something as simple as partitioning, keeping the more current data in the currently accessed partitions with hope that the SQL will select only the partitions with more current data. Of course, you always have the option to use a tool that does archiving for you. You'll want something that makes your data available for access yet isn't in the physical data sets. By removing some of the information not needed daily, you'll reduce the size of the warehouse. This will make the warehouse more manageable and could affect performance of queries that need to scan larger portions of the warehouse.

Linux on System z

Rather than run your analytics on Unix or Windows, consider running them under Linux on System z running on an Integrated Facility for Linux (IFL). Better yet, run them on multiple Linux systems on System z running under z/VM on that IFL. You get the security of running completely within your System z box and still get to take advantage of your IBM System z9 Integrated Information Processor (zIIP) specialty engines.

You can use hipersockets when your Linux application wants to get to DB2 for z/OS in another Logical Partition (LPAR). Hipersockets alone yield a performance advantage over typical network solutions, but that discussion is for another time. Hipersockets use TCP/IP; DB2 work

originated on Linux on System z uses a hipersockets Distributed Relational Database Architecture (DRDA) protocol and therefore uses enclaves. This means that up to approximately 50 percent of this analytic work coming from Linux on System z is zIIP eligible. (zIIPs were described in detail in the article "zIIPing Along," which appeared in the August/September 2006 issue of *z/Journal*.)

Conclusion

Will DB2 for z/OS always be the best choice for your data warehouse? Probably not. As with all things, there's always a best way to do things; sometimes it's DB2 for z/OS and sometimes it's DB2 LUW. There are many factors to examine when making a warehouse platform decision, including placement of the data feeds and available support expertise. The next time a platform conversation comes up, remember that:

- 25 of the top-25 worldwide banks are running on DB2 for z/OS.
- 23 of the top-25 U.S. retailers are running on DB2 for z/OS.
- Nine of the top-10 global life or health insurance providers are running on DB2 for z/OS.

Many companies seemingly know that if your business depends on your database, DB2 for z/OS is a good choice. Then there's the mainframe that DB2 runs on. The numbers there are equally impressive:

- 95 percent of the U.S. *Fortune* 500 companies use System z.
- 45 percent of the U.S. *Fortune* 1000 companies use System z.
- 71 percent of global *Fortune* 500 companies use System z.

Moreover, 80 percent of the world's corporate data resides on or originates on a mainframe. If DB2 for z/OS, combined with System z, is good enough to handle your business, it's good enough to handle your data warehouse ... again! **Z**

About the Author

WILLIE FAVERO is an IBM senior certified IT software specialist and the DB2 SME with IBM's Silicon Valley Lab Data Warehouse on System z Swat Team. He has more than 30 years of experience working with databases with more than 24 years working with DB2. He speaks at major conferences, user groups, publishes articles, and has one of the top technical blogs on the Internet. Email: wfavero@attglobal.net