Data Warehousing Guidelines for DB2

By Craig S. Mullins

More and more organizations are building their data warehouses using DB2 for OS/390 because of the scalability, reliability, and robust architecture that it provides. You can use the following guidelines as rules of thumb when you're designing, implementing, and using your DB2-based data warehouse. Some of the advice is platform-independent and useful regardless of the DBMS being used to build your data warehouse. However, the guidelines were written with DB2 for OS/390 specifically in mind.

Do Not Implement a Data Warehouse as a Panacea
Many data warehouse development projects begin with "pie in the sky" expectations. One of the biggest problems with a data warehouse project is a situation in which the data warehouse is viewed as a "magic bullet" that will solve all of management's information problems.

To alleviate these types of problems, you should manage expectations by securing an executive sponsor, limiting the scope of the project, and implementing the data warehouse in stages (or possibly by implementing multiple data marts for each department).

Do Not Become 100% Technology-Focused
When you're developing a data warehouse, be sure to include tools, people, and methods in your warehouse blueprint. Too often, the focus is solely on the technology and tools aspect. To be successful, a data warehouse project requires more than just sound technology. You need careful planning and implementation (methods) as well as a means to learn from the efforts of others (people) through mentoring, consulting, education, seminars, and user groups.

Do Not Mix Operational Needs into the Data Warehouse Project
When a data warehousing project is first initiated, it may have a mixture of operational and analytical/informational objectives. This mixture is a recipe for disaster. Redefine the project to concentrate on non-operational, informational needs only. The primary reason for the existence of the data warehouse in the first place is to segregate operational processing from reporting.

Ensure Read-Only Data
Create the data warehouse as a decision support vehicle. The data should be periodically updated and summarized. If your design calls for a data warehouse in which all the data is modified immediately as it is changed in production, you need to rethink your data warehouse design.

Consider starting DB2 data warehouse databases as ACCESS(RO) to ensure read-only access. Doing so has the additional effect of eliminating locking on the read-only databases. When the data warehouse is refreshed, the databases have to be restarted in read/write mode.

Consider Using Dirty Reads
Because data warehouses are read only in nature, locking is not truly required. You can specify ISOLATION(UR) for all plans, packages, and queries used in the data warehouse environment. With ISOLATION(UR) DB2 will take fewer locks, thereby enhancing performance. However, DB2 may read uncommitted data when ISOLATION(UR) is specified. This should not be a major concern in the read only data warehouse.

Be Aware of the Complexity of Implementing a Data Warehouse
Moving data into a data warehouse is a complex task. Detailed knowledge of the applications accessing the source databases that feed the data warehouse must be available. Be sure to allot development time for learning the complexities of the source systems. Frequently, the systems documentation for production system is inadequate or non-existent.

Additionally, be sure to analyze the source data to determine what level of data scrubbing is required. This process can be an immense, time-consuming task.

Prepare to Manage Data Quality Issues Constantly
Maintaining data quality will be an ongoing concern. Both the end users and the data warehouse construction and maintenance team are responsible for promoting and fostering data quality. Data problems will be discovered not only throughout the development phase of the data warehouse, but throughout the useful life of the data warehouse.

Be sure to establish a policy for how data anomalies are to be reported and corrected before the data warehouse is made generally available to its end users. Additionally, be sure to involve the end users in the creation and support of this policy; otherwise, it is doomed to fail. The end users understand the data better than anyone else in the organization, including the data warehouse developers and DBAs.

Do Not Operate in a Vacuum
As business needs change, operational systems change. When operational data stores change, the data warehouse will be affected as well. When a data warehouse is involved, however, both the operational database and the data warehouse must be analyzed for the impact of changing any data formats. This is true because the data warehouse stores historical data that you might not be able to change to the new format. Before the change is made to the operational system, the data warehouse team must be prepared first to accept the new format as input to the data warehouse, and second, to either maintain multiple data formats for the changed data element or to implement a conversion mechanism as part of the data transformation process. Conversion, however, can result in lost or confusing data.

Tackle Operational Problems in the Data Warehousing Project
You will encounter problems in operational systems that feed the data warehouse. These problems may have been in production for year, running undetected. The data warehousing project will uncover many such errors. Be prepared to find them and have a plan for handling them.

Only three options are available:

- Ignore the problem with the understanding that the problem will exist in the data warehouse if not corrected.
- Fix the problem in the operational system.
- If possible, fix the problem during the data transformation phase of data warehouse population.

Of course, the second and third options are the favored approaches.

Determine When Data Is to Be Purged
Even in the data warehouse environment, when certain thresholds are reached, maintaining certain data in the data warehouse does not make sense. This situation may occur because of technology reasons (such as reaching a capacity limit), regulatory reasons (change in regulations or laws), or business reasons

(restructuring data, instituting different processes and so on).

Plan to arrange for methods of purging data from the data warehouse without dropping the data forever. A good tactic is to prepare a generic plan for offloading warehouse data to tape or optical disk.

Use Denormalization Strategies
Experiment with denormalized tables. The opposite of normalization, denormalization is the process of putting one fact in many places. Because the data warehouse is a read-only database, you should optimize query at the expense of update. Denormalization will achieve this. Analyze the data access requirements of the most frequent queries, and plan to denormalize to optimize those queries.

There are ten types of denormalization that can be useful when implementing DB2-based data warehouses:

Denormalization	**Use**
Prejoined Tables	Combining two tables together into a single table when the cost of joining is prohibitive
Report Tables	Creating a table to store specialized critical reports that require fast access
Mirror Tables	Creating copies of tables when the data is required concurrently by two types of environments
Split Tables	Breaking a table into two parts when distinct groups use different parts of the table
Combined Tables	Combining two tables together when one-to-one relationships exist
Redundant Data	Carrying redundant columns in multiple tables to reduce the number of table joins required
Repeating Groups	Storing repeating groups in a single row to reduce I/O and (possibly) DASD usage
Derivable Data	Storing calculated results to eliminate calculations and algorithms

To Avoid BP32K	Splitting columns of very large rows across multiple tables to avoid using pages larger than 4K in size
Speed Tables	Storing pre-traversed hierarchies to support bill-of-material processing

As you design the data warehouses be alert for situations where each of these types of denormalization may be useful. In general, denormalization speeds data retrieval, which is desirable for a data warehouse. However, denormalize only when a completely normalized design will not perform optimally.

Be Generous with Indexes
The use of indexes is a major factor in creating efficient data retrieval. You usually can use indexes more liberally in the read-only setting of the data warehouse. Remember, though, you must make a trade-off between data loading and modification and the number of indexes.

The data warehouse indexes do not have to be the same indexes that exist in the operational system, even if the data warehouse is nothing more than an exact replica or snapshot of the operational databases. You should optimize the indexes based on the access patterns and query needs of the decision support environment of the data warehouse.

Also, use type 2 indexes to remove index locking as a consideration for the data warehouse.

Avoid Referential Integrity and Check Constraints
Because data is cleansed and scrubbed during the data transformation process, implementing data integrity mechanisms such as referential integrity (RI) and check constraints on data warehouse tables is not efficient. Even without a comprehensive cleansing during data transformation, the data in the warehouse will be as good as the data in the source operational systems (which should utilize RI and check constraints).

Encourage Parallelism
Use partitioned table spaces and specify DEGREE(ANY) to encourage I/O, CPU, and Sysplex parallelism.

Parallelism helps to reduce overall elapsed time when accessing large databases such as those common in a data warehouse.

Consider partitioning simple and segmented table spaces to take advantage of DB2’s parallelism features. Additionally, consider repartitioning partitioned table spaces to take full advantage of DB2 parallelism based on the usage patterns of your data warehouse access.

Consider Data Compression
As of DB2 V3 data compression can be specified directly in a table space. Compression is indicated in the DDL by specifying COMPRESS YES for the table space. Likewise, it can be turned off in the DDL by specifying COMPRESS NO. When compression is specified, DB2 builds a static dictionary to control compression. It saves from 2 to 17 dictionary pages in the table space. These pages are stored after the header and first space map page.

DB2's hardware-based data compression techniques are optimal for the data warehousing environment. Consider compressing tables that are infrequently accessed to save disk space. Furthermore, consider compressing all tables if possible.

Back Up the Data Warehouse
Putting in place a backup and recovery plan for data warehouses is imperative. Even though most of the data comes from operational systems originally, you cannot always rebuild data warehouses in the event of a media failure (or a disaster). As operational data ages, it is removed from the operational databases, but it may still exist in the data warehouse. Furthermore, data warehouses often contain external data that, if lost, may have to be purchased again (creating a financial drain).

Follow "The 10 Steps to Clean Data"
The following list is a short compendium of the top 10 things you can do to ensure data quality in your data warehouse environment:

1. Foster an understanding for the value of data and information within the organization. In short, treat data as a corporate asset. What does this mean? Consider the other assets of your organization. The capital assets ($) are modeled using a chart of accounts. Human resources (personnel) are modeled using

management structures, reporting hierarchies, and personnel files. From building blueprints to item bills of material, every asset that is truly treated as an asset is modeled. If your corporation does not model data, it does not treat data as an asset and is at a disadvantage.
Acceptance of these ideals can be accomplished through lobbying the users and managers you know, starting an internal newsletter, circulating relevant articles and books throughout your company, and treating data as a corporate asset yourself. A great deal of salesmanship, patience, politics, and good luck will be required, so be prepared.

2. Never cover up data integrity problems. Document them and bring them to the attention of your manager and the users who rely on the data. Usually, the business units using the data are empowered to make changes to it.

3. Do not underestimate the amount of time and effort that will be required to clean up dirty data. Understand the scope of the problem and the process required to rectify it. Take into account the politics of your organization and the automated tools that are available. The more political the battle, the longer the task will take. The fewer tools available, the longer the task will be. Even if you have tools, if no one understands them properly, the situation will probably be worse than having no tools at all as people struggle to use what they do not understand.

4. Understand what is meant by "data warehouse" within the context of your projects. What is the scope of the "warehouse": enterprise or departmental? What technology is used? If OLAP is a component of the environment, is it ROLAP or MOLAP?

5. Educate those people implementing the data warehouse by sending them to courses and industry conferences, purchasing books, and encouraging them to read periodicals. A lack of education has killed many potentially rewarding projects.

6. Physically design the data stores for the data warehouse differently than the similar, corresponding production data stores. For example, the file and table structures, indexes, and clustering sequence should be different in the warehouse because the data access requirements are different.

7. You will often hear that denormalization is desirable in the data warehouse environment, but proceed with caution. Because denormalized data is optimized for data access, and the data warehouse is "read-only", you might think that denormalization is a natural for this environment. However, the data must be populated into the data warehouse at some point. Denormalized data is still difficult to maintain and should be avoided if performance is acceptable.

8. Understand the enabling technologies for data warehousing. Replication and propagation are different technologies with different availability and performance effects on both the production (OLTP) and the warehouse (OLAP) systems.

9. Only after you understand the basics should you delve into the more complex aspects of data warehousing such as implementing an ODS, very large databases, or multidimensional databases.

10. Reread steps 1 through 9 whenever you think you are overworked, underpaid, or both!

Data in the warehouse is only as good as the sources from which it was gleaned. Failure to clean dirty data can result in the creation of a data outhouse instead of a data warehouse.

Use Good DB2 Database Design Techniques
Use efficient DB2 DDL design techniques such as you would use with any DB2 database design. This includes using the optimal table space type (segmented vs. partitioned), locking strategy, data set closing parameter, etc., etc. Good DB2 database design practices still must be followed when implementing DB2 data warehouses.

Summary
Data warehouses can provide organizations with a competitive advantage as users begin to analyze data in conjunction with business trends. After a data warehouse is implemented, you cannot turn back because your users will be hooked, your organization will be more profitable, and you'll have the satisfaction of contributing to the success of the business and, just maybe, a big fat raise.

From *Enterprise Systems Journal*, August 1998.