DB2 magazine

http://www.db2mag.com/db_area/archives/1998/q3/98fdba.shtml

Logical Database Design

Guidelines for a successful design project.

By David Beulke

Fall 1998

In my last column, I explained how essential it is to have a precise mission statement before designing a database. In this column, I'll relate some unique ideas and design techniques related to getting the most out of your database design whether it is for that 7TB data warehouse or for the new billing system.

SELECTING YOUR DATA

To determine what data to include in your database, begin by identifying and analyzing your existing data sources, processes, and outputs. This analysis helps identify the depth and range of the necessary data. Ask who will be using the system, who -- if anyone -- currently provides the type of information it will provide, how these people are managed, and what their division does for the company. Interviewing these people and documenting their interaction with the data is vital for determining the system's data requirements.

The next way to research what data you'll need is to look at the systems being replaced or consolidated. These systems may have provided the same or a similar type of functionality that your system is to provide. What was the critical data factor or data function difficulty that led to their demise? Analyzing and documenting these systems' functionality is critical for your database system's success. If you fail to replace some data or functionality provided by the old system, you may end up having to undertake costly reengineering later, even if your new system has additional features or other important capabilities.

Another way to come up with breakthrough processing or data ideas is to think about what is "impossible" or what the perfect database system would do. You're designing a new system, so you might as well try to make the best system possible. Listen to input from different levels of management, dedicated users of the existing system, and especially people who do not like the system or its concept. People who don't like the system can help you identify potential flaws and weaknesses.

You should completely document and categorize all this analysis. The documentation, or metadata, should include information on the data source, where it's used, activity against it, its primary and secondary keys, and any relationships with other data. The analysis efforts should focus on identifying data groupings, duplicate data, important keys, relationships, and timeliness of the data. As business intelligence requirements grow and OLAP database tools

become easier and less intrusive, timeliness and "transaction dimension" information (such as who, what, when, where, and under what conditions or promotion) becomes much more valuable. Analysis should also determine the aggregate knowledge that the data represents and ensure that it corresponds to the new system's overall processing objectives. This knowledge should match the system's mission statement and contain all the data knowledge necessary to provide the functionality and features desired.

NORMALIZATION

Once you've taken all the metadata into consideration, you're ready to begin the database design normalization process. This process breaks the data into groups, identifying keys, repeating groups, and distinct elements. Normalization puts the keys and data together properly so that it can be retrieved, updated, inserted, and deleted without jeopardizing the information's integrity. Validating the data integrity of the logical modeling process can be quite time consuming, but it is critical to ensuring that interaction between data groups is correct. Following your shop's standard database normalization process is the best way to get everyone to agree and endorse your design. And it's important to make your logical design the simplest possible representation of the data so everyone understands what knowledge and information the database represents.

Through your database normalization design process, eliminate repeating groups to expose relationships within the data. Document these relationships and dependencies and develop referential integrity constraints to govern them in your physical database design. Determination of the proper physical design for relationship constraint rules, such as Delete Restrict, Cascade, or Set Null, should be handled carefully to retain the integrity of the knowledge within the database. You can ask several questions to determine the proper referential integrity constraint: Is the data valid without other data? Is other data dependent on a child or parent relationship? Is only part of that other data affected by the data relationship? For example, you could represent a customer purchasing a piece of merchandise in a data relationship. The purchase could not exist on its own or without the customer. The merchandise transaction is also dependent on the purchase time and, potentially, the price. You could implement these relationships in several different ways: by not letting customers be deleted from the database if they have transactions (Delete Restrict); by cascading or deleting all customer transaction data when the customer information is deleted (Cascade); or by only setting the transaction customer information to null when the customer data is deleted (Set Null).

IDENTIFYING DOMAINS

Another factor critical to the success of any database design effort is identifying the data's domain, range, and indicator or code values and ensuring that these elements are compatible across data groups and system interfaces. You'll need to conduct domain recognition to determine how the data is represented -- for example, whether as a number, character field, audio, image, or video. To determine the domain of a data element, ask people who work with the data for typical and extreme examples of its use. Also, to ensure that you're identifying the proper domain definitions, make sure that any composite data is broken down to its smallest elements. Don't make the common mistake of misrepresenting the data to conform to an older system's incorrect definition. Misrepresenting the data can cause tremendous data cleansing

difficulties when adding data from outside sources for data warehousing or marketing systems.

Logical database analysis should also identify data range or scope to facilitate physical database definitions. Certain physical data element implementations, such as SMALLINT, can only represent up to a certain range of values. Understanding an element's range is also critical when using the element to define physical database partitioning or spread very large databases out for DASD I/O and parallelism considerations. Range information is especially important for indicator or code data elements: You can avoid programming and data population nightmares by staying with common consistent definitions. Standardization and validation software tools are a tremendous help on this front. Using common industry codes or abbreviations, such as stock keeping unit numbers, can also be a great help in settling tedious data range disputes.

NAMING STANDARDS

Another important element of successful database design is the use of proper naming standards or conventions. Every IS organization has standards, but often they aren't used because of internal politics, disputes, outdated names, or lengthy integration processes. Have your design team address standards issues at the very beginning of the database design effort so you have time to battle through the various opinions to a resolution. Working with a repository, standard abbreviation list, or existing interface or system can help guide your team in developing naming standards. Because these naming standards will exist for the lifetime of your system, it's important to ensure that they make sense and are easy to understand.

Designing new databases is a lot of fun. I hope this article has helped you understand some techniques of database design a little better: the time-tested data normalization process, data standardization, and naming standard techniques that will help reinforce your database design methodology.

David Beulke has been building database systems for more than 14 years and is currently building one of the largest databases in the world as the director of corporate data management at Bell Atlantic Inc. You can reach him at dbeulke@compuserve.com .
