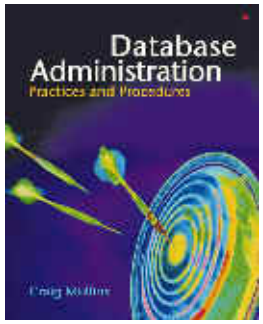
February 2006

DATABASE TRENDS AND APPLICATIONS

The DBA Corner

by Craig S. Mullins



Database Design and the Internet

When databases are designed for e-business applications the tendency is for the DBA to get swept up in the dynamics of web-based design and development. This can be a dangerous mistake when designing databases

One of the biggest problems when moving from traditional development to e-business development is coping with the mad rush to “get it done

NOW!" Industry pundits have coined the phrase "Internet time" to describe this phenomenon. Basically, when a business starts operating on "Internet time" things move faster. One "web month" is said to be equivalent to about three standard months. The nugget of truth in this load of malarkey is that web projects move very fast for a number of reasons:

- Because business executives want to conduct business over the web to save costs and to connect better with their clients.
- Because someone read an article in an airline magazine saying that web projects should move fast.
- Because everyone else is moving fast so you better move fast too or risk losing business.

Well, two of these three reasons are quite valid. Rapid application development (RAD) techniques have been around for over two decades now and they have been used with varying degrees of success. Sometimes RAD is required for certain projects. But RAD can be bad for database design. Why? Applications are temporary but data is permanent. Organizations are forever coding and re-coding their applications – sometimes the next incarnation of an application is being developed before the last one even has been moved to production.

But when did you ever throw away data? Oh, sure, you may redesign a database or move from one DBMS to another. But chances you most likely saved the data and migrated it from the old database to the new one.

Some changes had to be made, maybe some external data was purchased to combine with the existing data, and maybe some parts of the database were not completely populated. But data lives forever. To better enable you to glean value from your data it is wise to take care when designing the database. A well-designed database is easy to navigate and therefore, it is easier to retrieve meaningful data from the database.

The DBA always should create databases by transforming logical data models into physical implementation. It is not wise to dive directly into a physical design without first conducting an in-depth examination of the data needs of the business.

A proper database design cannot be thrown together quickly by novices. A practiced and formal approach to gathering data requirements and modeling data is needed. Such an effort requires a formal approach to the discovery and identification of entities and data elements. Normalization is a big part of data modeling and database design. A normalized data model reduces data redundancy and inconsistencies by ensuring that the data elements are designed appropriately.

Once the logical data model has been created the DBA uses his knowledge of the DBMS to be used to transform logical entities and data elements into physical database tables and columns. To successfully create a physical database design you will need to have a good working knowledge of the features of the DBMS.

Basically I am advocating that you follow database design best practices regardless of whether the database is being designed for traditional or e-business purposes. But what if you'd like to but are forced to operate on "Internet time" for certain databases? The best advice I can give you is to be aware of design failures that can result in a hostile database. A hostile database is one that is difficult to understand, hard to query, and takes an enormous amount of effort to change.

Of course, it is impossible to list every type of database design flaw that could be introduced to create a hostile database. But the following are common database design failures to watch out for:

- Assigning inappropriate or difficult-to-remember table and column names.
- Designing the database with output in mind. This can lead to flaws such as storing numbers in character columns because leading zeroes need to be displayed on reports. This is a bad idea.
- Another common database design problem is overstuffing columns. Sometimes a single column is used for convenience to store what should be two or three columns. Such design flaws are introduced when the DBA does not analyze the data for patterns and relationships. An example of overstuffing would be storing a person's name in a single column instead of capturing first name, middle initial, and last name as individual columns.

- Poorly designed keys can wreck the usability of a database. A primary key should be unique and non-volatile.

- Failing to account for International issues can have greater repercussions. For example, when storing addresses how do you define zip code? Zip code is USA code but many countries have similar codes; though they are not necessarily numeric.

Without proper up-front analysis and design the database is unlikely to be flexible enough to easily support the changing requirements of the user. With sufficient preparation flexibility can be designed into the database to support the user's anticipated changes.

If data is the heart of today's modern e-business, then the database design is the armor that protects that heart. Data modeling and database design is the most important part of a database application. If proper design is not a component of the database creation process, you will wind up with a confusing mess of a database that may work fine for the first application, but not for subsequent applications.

From Database Trends and Applications, February 2006.

Home.