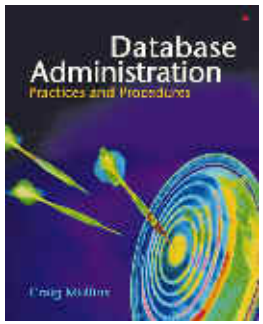
March 2005

DATABASE TRENDS AND APPLICATIONS

The DBA Corner

by Craig S. Mullins



Often Misunderstood, Don't Ignore the Use of Null

From time to time database professionals must deal with the prospect of missing information in their databases. The relational model popularized the concept of nulls for handling this situation.

A null represents missing or unknown information at the column level. If a column “value” can be null, it can mean one of two things: the attribute is not applicable for certain occurrences of the entity, or the attribute applies to all entity occurrences, but the information may not always be known. Of course, it could be a combination of these two situations, too.

A null is not the same as 0 or blank. Null means no entry has been made for the column and it implies that the value is either unknown or not applicable. A null column is not equal to, greater than, or less than any other column or value, even another null. It is just unknown.

A database that supports nulls enables the user to be able to distinguish between a deliberate entry of 0 (for numerical columns) or a blank (for character columns) and an unknown or inapplicable entry (NULL for both numerical and character columns). Null indicates that the user did not explicitly make an entry or has explicitly entered NULL for the column. For example, a “value” of null in the Price column of the ITEM table does not mean that the item is being given away for free; instead it means that the price is not known or has not yet been set. In general, defining a column as nullable provides a place holder for data you might not yet know.

Nulls sometimes are inappropriately referred to as “null values.” Using the term value to describe a null is inaccurate because a null implies the lack of a value. Therefore, simply use the term null or nulls (without appending the term “value” or “values” to it).

When are nulls useful? Suppose that we also capture employee’s hair color when they are hired. Consider three potential entity occurrences: a man with black hair, a woman with unknown hair color, and a bald man. The woman with the unknown hair color and the bald man both could be assigned as null, but for different reasons. The woman’s hair color would be null meaning presently unknown; the bald man’s hair color could be null too, in this case meaning not applicable.

How could you handle this without using nulls? You would need to create special values for the HairColor column that mean “bald” and “unknown.” This is possible for a CHAR column like HairColor. But what about a DB2 DATE column? All occurrences of a column assigned as a DATE data type are valid dates. It might not be possible to use a special date value to mean “unknown.” This is where using nulls is most practical.

Let’s consider another example, this time for a DATE column. When a new employee is hired and is inserted into the EMP table, what should the employee termination date column be set to? I don’t know about you, but I wouldn’t want any valid date to be set in that column for my employee record. Instead, null can be used to specify that the termination date is currently unknown.

Today's database systems do not differentiate between nulls that signify unknown data and those that signify inapplicable data. This distinction must be made by the program logic of each application. Keep in mind, though, that using null to indicate "not applicable" can be an indication of improper database design. By properly modeling and normalizing your data structures you can usually eliminate the need to use nulls to indicate that a column is inapplicable for a specific row.

Whenever possible, avoid nulls in columns that must participate in arithmetic logic (for example, DECIMAL money values), and especially when functions will be used. Results can be confusing when using functions on nullable columns. For every other column, determine whether nullability can be of benefit before allowing nulls. And be sure that you understand the exact manner in which your DBMS treats nulls. Different products can treat nulls differently with regard to how nulls sort in an ORDER BY, whether nulls are considered equal for eliminating duplicates (for example, when using DISTINCT), how indexes treat null columns, and so on.

Nulls are certainly one of the most misunderstood features of today's DBMS products. Most of that has to do with the less than clear implementation and the confusion of the requisite three-valued logic. Although nulls can be confusing, you cannot bury your head in the sand and ignore nulls if your DBMS supports them. It is quite possible to create a database with no nullable columns, while still being able to code a query that returns a null. Understanding what nulls are, and how best to use them, can help you to create usable databases and design useful and correct queries in your applications.

From Database Trends and Applications, March 2005.

Home.