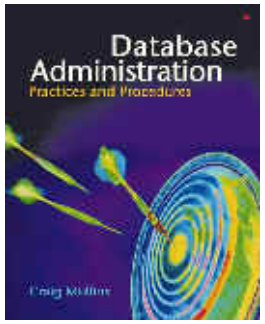
Craig S. Mullins

Return to Home Page

August 2007

The DBA Corner

by Craig S. Mullins



Data Quality Starts With Proper Data Type

Data type and length are the most fundamental integrity constraints applied to data in a database. Simply by specifying the data type for each column when a table is created, the DBMS automatically ensures that only the correct type of data is stored in that column. Processes that attempt to insert or update the data

to a non-conforming value will be rejected. Furthermore, a maximum length is assigned to the column to prohibit larger values from being stored in the table.

The DBA must choose the data type and length of each column wisely. It is almost always best to choose the data type that most closely matches the domain of correct values for the column. In general, adhere to the following rules:

- If the data is numeric, favor SMALLINT, INTEGER, or DECIMAL data types. FLOAT is also an option for very large numbers.
- If the data is character, use CHAR or VARCHAR data types.
- If the data is date and time, use DATE, TIME, or TIMESTAMP data types.
- If the data is multimedia, use GRAPHIC, VARGRAPHIC, BLOB, CLOB, or DBCLOB data types.

Now, I don’t know exactly why using improper data types is such a widespread practice, but I can guarantee you that it is. I am constantly being bombarded with questions that bounce around this topic.

One of the most common data type issues I see “out there” is storing dates in character columns. This causes all sorts of problems. Now, I know that every DBMS product supports date/time date a little bit differently, but they all provide the following benefits that you don’t get storing your dates in character columns:

- Assuring data integrity because the DBMS will ensure that only valid date/time values are stored. There is no way to store a non-date value date in a column defined with a date data type.
- Date/time columns can participate in date and time arithmetic functions. Adding one date to another or subtracting a duration from a date is difficult to program if the data is not stored as a date.
- Every DBMS provides a vast array of built-in functions that operate on and transform date and time values.
- Date/time columns can be formatted in multiple ways by the DBMS using DBMS commands or functions.

So, the next time you see a character date column containing '02302006' or something even more nonsensical, blame the database schema – or the DBA who created that schema.

Another common mistake is to define numeric data as character. For example, consider the following scenario. A four-byte code is required to identify an entity; all of the codes are numeric and will stay that way. But, for reporting purposes, users wish the codes to print out with leading zeroes. Should the column be defined as CHAR(4) or SMALLINT?

Without proper edit checks, inserts and updates could place invalid alphabetic characters into the product code. This can be a very valid concern if ad hoc data modifications are permitted. This is rare in production databases, but data problems can still occur if the proper edit checks are not coded into every program that can modify the data. If proper edit checks are coded and will never be bypassed, this removes the data integrity question. Check constraints are not a very viable solution in this case, either. Consider the following constraint

```
COL  CHAR(4) CONSTRAINT NUMBER CHECK
             (COL >= '0000' AND
              COL <= '9999'),
```

This constraint would allow the following value '0ABC'. It is greater than '0000' and less than '9999', so it fits within the constraint.

Choosing the wrong data type can impact performance, too. Relational optimizers can do a better job of calculating filter factors and building proper access paths when columns are defined according to their domains. Consider our example again. There are many more possible values for a CHAR(4) column than a SMALLINT column. Even if programmatic edit checks are coded for each, the DBMS is not aware of these and the optimization process will assume that all combinations of characters are permitted.

But what about those leading zeroes? If the data is stored in a CHAR(4) column we can just input the data with leading zeroes and then they will always be there. Well, this “problem” can be solved using other methods. Better to transform the data when it is accessed than trying to store it that way in the first place. After all, most reporting and query tools have features that will automatically insert those leading zeroes. And if you are accessing the data using a program, it is easy to insert leading zeroes, right?

In general, it is best to assign your columns the data type that best matches the values in its domain. By choosing that data type that most closely matches your data you will be doing yourself, your systems, and your users a big favor by improving overall data quality.

From Database Trends and Applications, August 2007.

Home.