DB2 and the Year 2000

By Craig S. Mullins

Ensuring Year 2000 compliance is a pervasive issue throughout the IT industry. Yet, many people assume that DB2 is immune from the Year 2000 problem because it provides date data types, numerous date formats, and date logic. As welcome as this support is, it can only be used once the offending date columns are identified. Many DB2 applications were implemented using character data types to store date values. This occurred for many reasons foremost of which were either ignorance (did not know about date data types), fear (did not understand date data types), desire to store a non-date and non-null default value, or the application was implemented prior to DB2 V1.3 (the first version of DB2 to provide date data types).

In this article I address the year 2000 crisis as it impacts DB2 from various perspectives including administration and development, data administration, staffing, performance and tools.

Two Approaches to Year 2000

There are two approaches you can take to address the Year 2000 problem in DB2 tables. You can convert the affected columns to DATE type and change the application code appropriately. This is known as the field expansion approach. This involves making changes to both the DB2 tables and the application logic that accesses the tables.

Alternately, you can use the application logic approach. Under this approach you will update each piece of application logic that accesses a date to handle the date without the century (the application logic approach). This involves transformation logic to understand whether the date is pre- or post-Year 2000. This type of code typically takes a form similar to the following pseudo-code:

```
IF YY > 50 THEN
        CC = 19
ELSE
        CC = 20;
```

This code assigns 20 to the century component of the year if the date is less than 50; 19 if it is greater than 50. Of course, the value 50 is somewhat arbitrary and will vary based on the type of processing. Additionally, the actual code can be much more complex due to processing issues such as whether valid dates can fall under both the current and future century (e.g. the years 1925 and 2025 are both valid for the application in question).

For either approach, the following steps need to be taken.

Identify columns that need to be addressed. You must compile a list of the columns that need to be examined. To compile such a list, a good starting point is to look for columns having the CHAR data type with the string "DATE" (or something similar) in its name.

Consider implementing standards for database dates so that all dates in all tables follow the standard. Failure to do so can cause performance degradation. For example, consider the implications when date columns are joined or used in SQL predicates. Comparing a date data type to a character — even if each column contains the same date — either will not match or cause performance to slow as one data type is converted so the match can occur.

Additionally, CHECK constraints on date columns (whether actually of DATE data type or not) should be re-evaluated to ensure that the appropriate checks will be performed when the century changes.

Identify code that references the columns. Once the columns have been identified, you must find all the application code that references the columns. Start by searching for occurrences of the column names in your source code libraries and the DB2 Catalog. But, this may not identify all of the application code using date columns. You could miss dynamic SQL and statements that access the column using a view. Ensure that you have a dependable method of locating all affected code. Automated tools exist that can help you to identify COBOL and SQL code that references date columns.

Set up a test environment. Consider setting up a test environment exclusively for Year 2000 testing. This enables you to isolate Year 2000 from other development and testing efforts. You might create a duplicate of an existing test environment or create a scaled-down version of a production system. You also need some way to generate test data, with dates in the future, for your integrated system testing.

DBAs will need to populate test beds for year 2000 testing. Products that automatically maintain referential integrity of test beds generated from production databases may become cost-justifiable within the scope of year 2000 compliance testing.

Make the changes. For the application logic approach, programmers must modify the source code as needed. For the field expansion approach, you must alter the columns to DATE types. This requires dropping and recreating the tables, including unloading data, converting the dates, and reloading the tables with the converted data. Before dropping the tables, make sure to identify any dependencies on the tables, such as referential integrity, indexes, views, aliases, synonyms, and authorizations, as these also will be dropped. When you recreate the tables, you must also recreate the tables' dependencies.

When changes are made to application programs that expand date columns in RDBMSs, DBAs need to be involved in the process of moving the changes to production because of possible referential integrity changes to primary key (PK)-foreign key (FK) relationships. If the PK is changed, the FK must be changed at the same time, and vice versa. Failure to do so will result in poor performance at best or data integrity violations at worst.

Test the applications. Planned changes to systems, programs, utilities and databases must be tested to ensure the coded changes are working as designed, are producing the desired results and that production processing failures are prevented. Therefore, all applications that are affected must be thoroughly tested using dates up to and beyond the Year 2000. In addition, you should include some critical test dates as part of your test plan. For instance, including dates such as 28-Feb-2000, 29-Feb-2000, and 1-Mar-2000 will help to verify correct leap year processing. The testing phase is crucial as testing will probably account for more than half the time spent on Year 2000 compliance. Although testing will consume the majority of your Year 2000 project lifecycle, if it is done correctly, testing will keep risk exposure at a minimum.

Processing failures can be minimized by performing thorough testing of all system components to simulate real-world processing conditions wherever possible. Various tools are available to assist with Year 2000 testing including: date simulators that roll the system clock forward and terminal capture and replay tools.

Implement the changes. For the field expansion method, once your testing is done, you need to migrate the changed tables to your production systems. For all code changes, a change and configuration management plan is needed. The database and code changes should by synchronized to occur simultaneously. If one is rolled back from production, a process needs to be in place to roll the other back, too.

These are the basic black and white issues of ensuring year 2000 compliance. However, let's examine some of the more gray issues surrounding the year 2000 and its impact on IT.

Future Performance Implications

Sometimes organizations take shortcuts to avoid year 2000 application problems. Consider a credit card application that does not support four digit dates. To "get around" this problem the issuer sets the expiration date for all new and expiring credit cards to be no later than December 1999. The thinking goes: "when the applications are fixed we can re-assign all of the credit cards to future dates." However, this is a "fix" that can cause more problems than it fixes.

Assume that the application is not fixed until late 1998. At that point in time, the issuer can begin to issue credit cards with a post-2000 date. But what happens in 1999 when all of the credit cards that were issued over a multiple year period begin to expire? Are the systems ready for the increased transaction workload that would normally be spread across multiple years? If not, the organization must plan on issuing new cards much earlier — before they begin to expire. In this case, the company opens itself up to questions from its customers regarding why they must start using a new credit card before the old one expires. What a nightmare!

A Data Administration Opportunity

As part of the year 2000 process, automated tools may be used to scan entire production application portfolios. These scans are used to identify and correct date problems within the applications. Consider using this opportunity, when the entire programs library is being scanned anyway, to document and catalog the metadata for these applications in a repository or data dictionary tool. A project to capture such a massive amount of information may be impossible to cost-justify outside the scope of the year 2000 project. As such, do not squander the opportunity to proactively catalog and document your metadata.

Staffing Issues

Many enterprises have the absolute minimum number of DBAs assigned to support DB2. This means that limited administration and technical support is available for Year 2000 projects because the DB2 DBAs are scrambling to support the new development that is occurring. This causes a staffing shortage that will be difficult to alleviate without additional headcount being allocated for Year 2000 support. This may be a good opportunity to get additional database administration support because the purse-strings are often easier to open for Year 2000 projects than for any other type of project. Let's face it – there is a hard and fast deadline that can't be missed!

Synopsis

The Year 2000 problem is pervasive and will consume many development cycles and dollars. Be prepared by automating the process as much as possible using Year 2000 and database administration tools to minimize risk and increase the speed of conversion.

From *DB2 Update (Xephon)*, May 1998.