The Resource for Users of IBM zSeries & S/390 Systems
zJOURNAL

August / September 2011

zData Perspectives

On The Need for Temporal Data

by Craig S. Mullins

A traditional DBMS stores data that is implied to be valid at the current point-in-time, it does not track the past or future states of the data. Temporal support makes it possible to store different database states and to query the data "as of" those different states. Many types of data change over time, and different users and applications have requirements to access that data at different points in time. For some, the current, up-to-date values for the data are sufficient. But for others, accessing earlier versions of the data is needed.

Data stored in a DBMS that supports temporal features differs

regulatory mandate may require you to be able to track changes to a piece of personally identifiable information (or PII). Examples of PII include name, phone number, social security number, and drivers license number (there are many others). Tracking changes to PII is a requirement some industry and governmental regulations.

System time support stores every change made to the data in a related history table. System time tracking is sometime referred to as data versioning, because every version of the data's state is stored and can be queried. Support for managing system changes enables users to be able to query the database as of a point in time returning the value of the data as of that timeframe

Data stored in a DBMS that supports temporal features differs from traditional data in that a time period is attached to the data to indicate when it was valid or changed in the database. Many readers can probably imagine a situation when temporal database support might have been helpful in their past development endeavors. But if not, consider an insurance company which sells policies to its customers. The terms of any specific insurance policy are valid over a period of time – a ***business time***. Over time, the customer can choose to decline further coverage, continue with the existing coverage, or modify the terms of their coverage. So at any specific point in time, the terms of the customer's policy can differ. Over time, customers will make claims against their policies. This claim information needs to be stored, managed, and analyzed. Accident histories for customers are also important pieces of data with a temporal element.

Now consider the complexity inherent in trying to develop not only a database design that accommodates changing policies, claims, and historical details, but also allows queries such that a user might be able to access a customer's coverage at a given point in time. In other words, what policies were in effect for that customer as of, say August 5, 2008?

This concept of business time can become quite complex. Consider the situation where a customer has multiple policies that expire on different schedules. Add in the possibility for periods where coverage lapses. The database is growing in size, and the queries are getting more complex.

Of course, there are many other types of applications for which temporal support would be useful, it is not just limited to timeframe.

The business time indicates the period during which the data is accurate with respect to the world. The system time indicates the period during which the data is stored in the database. These two time periods do not have to be the same for a single piece of data. Suppose, for example, that we wish to store temporal information about 20th Century events. The valid business time for this data would be within the range of 1900 and 1999. But if we were to add the information to the database now, perhaps the valid system time for the data would be at some point in 2011. So, of course, you may need to be able to support both business temporal data and system temporal data in the same table. This is called bi-temporal support.

DB2 10 for z/OS supports both types of temporal data: business and system. To implement temporal support, a time period must be defined using two additional columns per row: the beginning and end date for the time period.

Additionally, DB2's SQL support has been modified to include temporal extensions. There are three period specifications that can be added to SQL statements to access data by system and business time periods: AS OF, FROM/TO, and BETWEEN. For example, the following SQL selects the price of the ENGLISH 101 course offering as of December 25, 2011:

```
SELECT PRICE
FROM COURSE FOR BUSINESS_TIME AS OF '2011-12-25'
WHERE TITLE = 'ENGLISH 101';
```

Of course, you should implement temporal tables with care.

insurance. For example, financial applications, credit history, personnel management, transportation applications, reservation systems, and medical information management all may benefit from maintaining temporal data.

There is another concept of temporal data that needs to be considered, as well. Instead of business time, you also may need to track and manage ***system time***. For example, a

Define time periods only for those table with a business or regulatory requirement, and make sure you define your tables appropriately with business and system time periods.

But the bottom line is that applications requiring temporal support can greatly benefit from the efficiency of DB2's built-in facilities for temporal data.

From zJournal, Aug / Sept 2011.