Why Temporal Database Technology?

An ***if…*** White Paper

by John Bair and Michael Soo

November 1998

Introduction

We are often asked why we need temporal database technology to manage and query historical data. The argument put forth by skeptics usually goes something like this:

> "We already have a date-time data type in SQL. It should be easy to manage history. We'll just timestamp[1] our tables. Why do we need a new technology?"

The data warehouse variation goes something like:

> "We're already managing historical data. Why can't we just add timestamps to our data model? We'll just use our regular OLAP/ROLAP/SchmOLAP products to query changes over time. Why do we need a new technology?"

Well, the simple answer is that their intuition is misleading. Managing and querying temporal data is a lot harder than it looks and current products and technologies do a surprisingly poor job of this.

To understand how and why, we first need to know what we mean by "temporal database".

What Is A Temporal Database?

The word temporal (pronounced 'temp-er-ul") means "of or relating to time", or "time-varying", or "time-dependent".

In the most general sense, one might argue that any database containing dates or times can be called a temporal database. But if this were the case, then nearly every database in existence qualifies as a temporal database. The definition of temporal database must be more specific than this.

Alternatively, one might define a temporal database as any database containing time-varying or time-dependent data. But doesn't nearly all data vary over time? (In fact, data that does not change is not very interesting in terms of data management!) Most data is valid only for a certain time period, anyway. A travel reservation, for example, is time-dependent in that it is valid only for specific dates. This doesn't seem to be a workable definition, either.

A Useful Definition

What is important is *change*. Think about defining a temporal database as a database that manages a history of changes to time-varying data. We contend that in order to qualify as a temporal database, a database must preserve and permit access to historical data while supporting the addition of new or changed data.

Using this definition, a data warehouse is a temporal database system because it manages the historical record of the time-varying data in an enterprise. Likewise, a dimensional data mart whose fact table contains time-varying data also qualifies as a temporal database. In contrast, a travel reservation database does not qualify as a temporal database, unless it maintains a history of changes to reservations. Now we're getting somewhere!

[1] The origin of the word *timestamp* can probably be traced to the practice of rubber stamping the date and time on business documents to record the time of processing.

Note that a temporal database doesn't really track time – it tracks how *changes occur over time*. Another way to explain this is to say that a temporal database system possesses enhanced *data retention* facilities over conventional database systems.

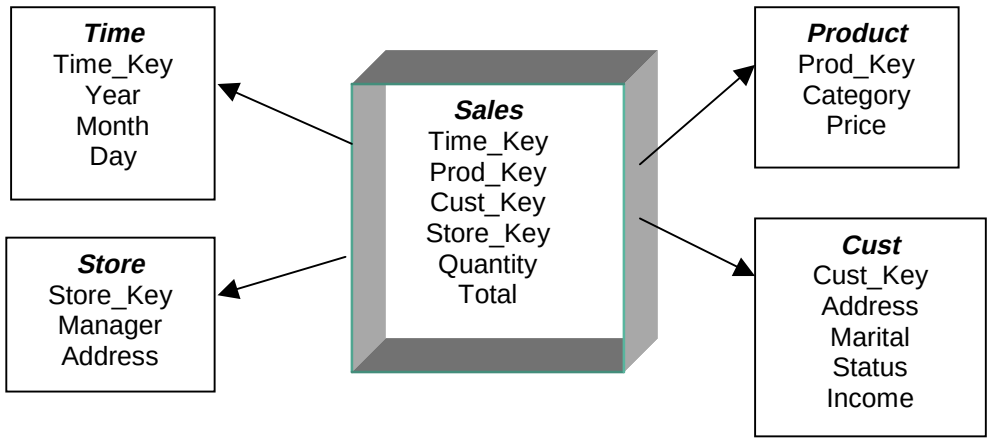
Some form of time-stamp must be used in any temporal database. A *timestamp* is any attribute that represents the time or time period for which an associated object or data is valid. And the skeptic's intuition is partly correct - a temporal database *does* timestamp data. But their intuition leads one to believe that the whole subject ends there, whereas that is really just the beginning!

Defining Temporality

Let's define the term *temporality* as the degree of temporal support provided by a database system. Database systems can provide different degrees of temporality. A database system that provides a small amount of support for temporal data can be thought of as exhibiting weak temporality. A database system that provides more robust support for temporal data has a stronger temporality.

For example, if a database contains a single temporal table (i.e., a table that contains a historical record of change over time) it does qualify as a temporal database, per our definition. However, such a database exhibits weak temporality, because only one of the tables is temporal and the rest are not. With only one temporal table, it is not possible to relate changes in the temporal table to any other tables, since *none* of the other tables are temporal.

For instance, consider the following data mart star schema, so called because the fact table sits in the center with its dimension tables arrayed in a star-like shape. The schema shows a temporal fact table recording sales information, and non-temporal dimension tables for recording time, product, customer, and store information.

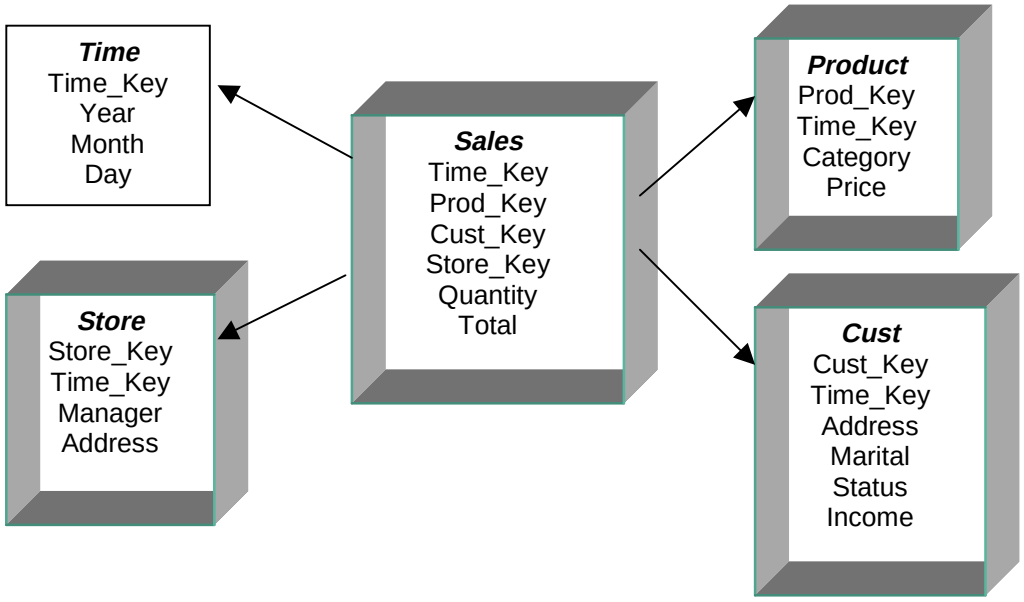


Suppose an unmarried customer buys a widget in January. In the data mart, the widget sale is inserted into the Sales fact table, and the customer information is inserted into the Cust dimension table. Later, if the same customer marries, an update to her marital status will be made in the Cust table. Since the dimension table is not temporal, the historical marital status of the customer has now been lost, and the information about the widget sale is now out of context. We know that the customer was single when she bought the widget, but the database incorrectly shows that she was married at that time.

This type of weak temporality exists in all conventional data marts, where a temporal fact table is related to a number of non-temporal dimension tables. SQL handles this type of temporal database rather well (because it is easy to do). Unfortunately - as this example shows - since only the fact table is temporal, facts are often interpreted inaccurately without their historical context.

Attempts to remedy this problem without the benefit of temporal database technology are partial solutions at best. These solutions often require custom data structures, which are frequently not compatible with existing schemas. In addition, these custom structures are not compatible with existing queries and other database operations, and are neither easily extensible nor reusable.

Let's look at the other extreme, where all tables in the data mart are temporal. Our data mart example is changed to make each dimension table a temporal table. (Surprisingly, the time dimension is *not* a temporal table, since it does not track changes over time!)



Now the Cust table records the full history of changes to customer information; the same is true of the Product and Store tables. And the information in the Sales table becomes significantly more valuable since all of the sales facts can be accurately viewed and interpreted in their proper historical context. In fact, it is now possible to relate changes in any table to changes in any other table, dimension or fact!

Stronger temporality lets you to look at the context within which business changes are made, and allows tracking of those changes and their effects. It's no exaggeration to say that databases that fail to preserve the full historical context of events literally throw away valuable information with each passing day.

How do you rate your current database system on preserving the historical context of your data? Can you think of an instance where you could make use of historical data?

The Power of Temporal Queries

A temporal query is a query whose result set measures change over time. According to this definition, many - if not most - data warehouses or data mart queries are actually temporal queries: a simple example is a report that summarizes sales by month. But just as a database with a single temporal table is a weak temporal database, conventional queries are only weakly temporal.

Here are some common examples of temporal queries that are supported by today's temporal database products:

- **Simple time constraints** – Which customers were single at a specific time during the last year?
- **Grouping and aggregation by time** – What were total revenues by fiscal year?
- **Simple joins between a temporal table and one or more non-temporal tables** – Which customers purchased which products and when?

Occasionally, a real SQL guru somewhere might formulate some hideously complex queries that do more than this. (This paper is not about SQL, so we will refrain from graphically illustrating this particular point!) But one shouldn't have to be an SQL guru - or have to know SQL at all, for that matter - to be able to formulate and run temporal queries. After all, people don't need to know HTTP and HTML to use the web; they don't need to know C++ and MIME to send email; they shouldn't need to know SQL to use a database system. SQL gurus may design database systems, but end-users use the systems.

When the complexity of the end-user's queries exceeds the capabilities of their query tools, either the SQL gurus will spend the rest of their days building queries for the users, or the systems will be underutilized, or not used at all. What the end-user needs is a flexible, powerful, and intuitive temporal query tool - then they can query the database themselves, without having to delve into the complexities of SQL.

Emerging temporal database products are providing some of these capabilities. Here are some examples:

- **Event detection** – Which customers got married so far this year? Which products underwent packaging changes? Which employees were promoted?
- **Event correlation** – What percentage of customers experienced a decline in their account balances shortly after they married? What is the average percentage increase in profit after packaging changes were made? What percentage of the total payroll increases was due to promotions?
- **State detection** – What percentage of customers have lived in the same home for over 3 years, and what percentage of overall sales do they account for? Compare the average percentage increase in profit for products with packaging changes against products with no packaging changes over the same time period. Which employees have held the same job for more than two years?

How to formulate these queries may seem obvious to some. But beware, because even though we may think we are familiar with our data, quite often our intuition about relationships in our data is wrong. Bad information breeds bad decisions, and the quality of decisions can make or break a business. Note that these are not *new* questions. Businesses have always asked these types of temporal queries. What is new is the capability for end users to ask and answer these types of temporal queries without resorting to help from SQL gurus or spending time and money on

custom programming. A data warehouse or data mart is only as valuable as the information it produces. Think about the types of temporal information that could be valuable to your business.

Die-hard skeptics may point out that it is possible to construct a specialized data model to answer any one of these questions, and they would be right. But they'd be missing the point. You can't predict all the questions that will be asked. You shouldn't have to build specialized data models to answer individual questions. And you ought to be able to run these types of queries against your existing data warehouses and data marts. If not, then you are losing valuable information.

If you remain skeptical, try to use your existing query tools to run some similar queries against your database. If you can't do it with your query tool, ask your local SQL guru how much work it would take to formulate and run these queries.

State of the Art

What does the future hold for temporal database technology? (That's a temporal question!) Let's examine the features of current and emerging temporal database products. We'll see that today's temporal database technology is a significant improvement over the "timestamping a database" solution used in conventional systems.

Let's call existing temporal database products, *first generation temporal database products*. First generation solutions provide weak temporality, and can be found in most data warehouse and data mart products. Granted, data warehouses are more than just databases, and data warehouse products incorporate many other technologies besides temporal database technologies. Nevertheless, a fundamental purpose of a data warehouse is to maintain a historical record of the data of an enterprise. Temporal technologies are used to extract and transform data, and load and query the warehouse. So, if you are using data warehouse technologies and products, and didn't think that you needed temporal database technology, you might be surprised to learn that you are already using the fruits of first generation temporal database technologies!

Second generation temporal database products are only now beginning to appear in the marketplace. These systems offer several exciting features, including more complex temporal queries and data mining, that go above and beyond the simple features of first generation products.

In the future products, we can expect to see some or all of the following features in temporal database products:

- **Stronger temporal queries** – Query and reporting tools will support event detection, event correlation and state detection.
- **Temporal data mining** – Data mining products will perform temporal pattern recognition. For example, stock analysts may mine historical price patterns to find similar patterns that may indicate future increases or decreases.
- **Schema versioning** – Schema changes, e.g., via an SQL ADD, DROP or CREATE statement, occur less frequently than data changes. However, in order to view data in its full historical context, it is necessary to maintain the full history of schema changes as well.
- **Data lineages** – As a data warehouse evolves, the data sources feeding the warehouse are bound to change. Knowing the "lineage" of the data can help determine if data conflicts are present, and if the data is reliable.

- **Business rules** – The methods or rules that are used to compute values such as profit and costs may change over time. A temporal database system might preserve old rules while permitting changes to the rules. A stronger temporal database system might permit queries that compare query results of different versions of business rules.
- **Data hierarchies** – Hierarchies are relationships in your data, for example, how products are grouped into product categories, or how locales are organized into sales regions. These hierarchies can change over time, e.g., products are added or removed from categories when suppliers change, or sales regions split or merge. A temporal database might permit new hierarchies to be entered while preserving and permitting access to the historical hierarchies. A stronger temporal database system might implement user-defined rules to facilitate conversions for queries that span hierarchy changes.

So how far can temporal database systems take us? The ultimate system will maintain the full historical record of both data and "metadata", including schemas, data lineages, business rules and data lineage. Some day in the future we can expect to see the emergence of such "zero-loss" systems. In the meantime, though, temporal database technologies will deliver competitive advantages in strategic business applications.

Product Generation	***Temporal Queries***	***Data Mining***	***Schema Versioning***	***Data Lineages***	***Business Rules***	***Data Hierarchies***
First	Simple	No	No	No	No	No
Second	Complex	Yes	No	No	No	No
Future	Very Complex	Yes	Yes	Yes	Yes	Yes

Temporal database technologies, having gained a foothold and some initial acceptance, will become pervasive in all areas of data warehousing and historical data management. As always, application demand will drive the development of these new products.

Applications

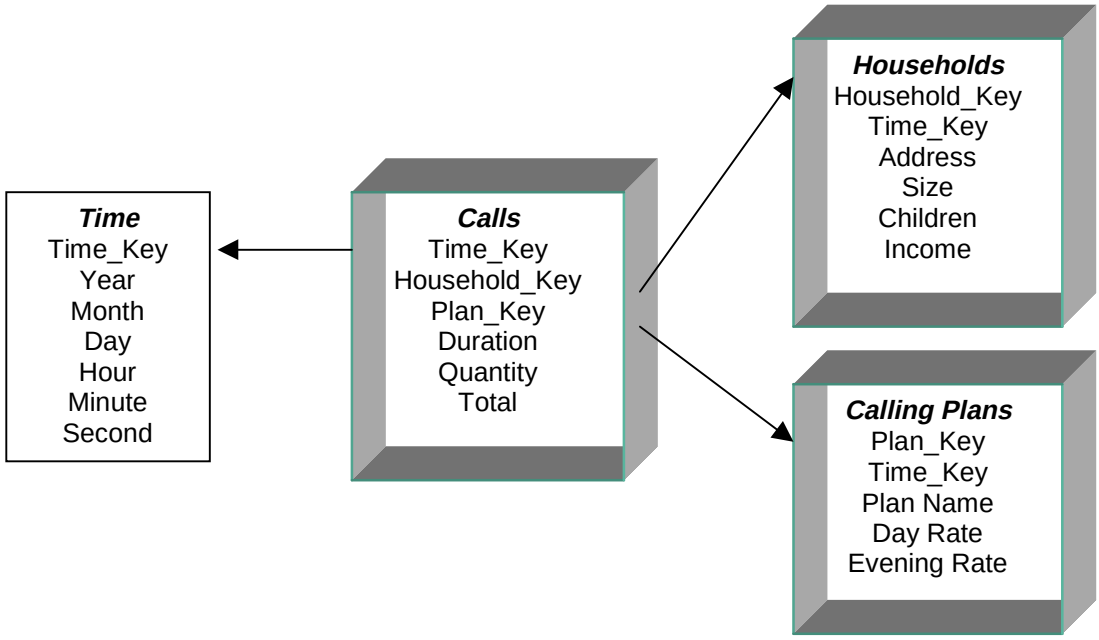
Now that you know a little more about temporal database technology, so what? What sorts of applications can benefit from temporal database technology? What is the business value of temporal database technology? Where does the rubber meet the road?

The fact is that almost any application can realize long-term benefits. Temporal database technology, however, can show a rapid return on investment for applications that depend on analyzing and understanding complex relationships that change over time.

For example, consider *customer churn analysis*. Churn analysis tries to answer questions like:

- Which customers come and go?
- What are the attributes of a quality customer, and why do they leave?
- What types of customers churn the fastest? How are these customers affecting profitability?

Suppose your company is a long-distance telecommunications service provider. Your primary business objectives are to attract new customers and retain existing customers by offering service plans and incentives to keep them loyal. To do this, you construct a data warehouse that consists of three dimension tables recording time, calling plan, and household information, and a fact table recording detailed call information.



The Calls fact table contains the history of calls made by each household. The Calling Plans table contains the service plans offered by the company. The Households table records information about each household served by the company.

Each of these tables is temporal: different calling plans have been offered at different times in the past, household demographics change over time, and phone calls are placed at all different times.

The data mart can be queried to determine those households that have terminated their service and when. This information can then be temporally correlated with the service plan in use when they terminated their service and their actual service usage, i.e., call history, at that time. For example, it may be that married households with two or more children have heavy call usage between the hours of 3pm, when the children return home from school and 5pm, when the parents return from work. Given these calling patterns, it may be appropriate to offer an adjusted rate plan between 3pm and 5pm.

Here are some other applications that can benefit:

- **Database Marketing** – The success of marketing programs hinges on the ability to respond to and predict changes to customer demographics and customer behavior. Such relationships are temporal by their very nature. Investments in temporal database technologies could reap large rewards.
- **Customer Relationship Management** – Managing churn is one aspect of the larger problem of customer relationship management. Changes to your products and service levels can have a great impact on customer relationships. What do customers want and how do their wants and needs change over time?

- **Human Resource Management** – What are the optimum levels of resource utilization and personnel productivity? Which personnel changes correlate with productivity changes? What percentage of employees is approaching retirement? What positions have high turnover and what contributes to the most frequent attrition?

There are many other applications that can reap big rewards with temporal database technology.

Summary

A temporal database system preserves the history of changes to time-varying data; it maintains the historical context of the data. Losing this context means your information becomes inaccurate, and relying on inaccurate information leads to bad business decisions.

Attempts to manage temporal data without the benefit of temporal database technology are partial solutions at best. These strategies often require custom data structures, and are frequently not compatible with existing schemas, queries and other database operations. In addition, they are not easily extensible or reusable.

Temporal technology isn't pie-in-the-sky. If you're using today's data warehouse or data mart products, it is likely that you are using first generation temporal database technology. The next generation of temporal database products is now emerging with the promise of more complex temporal queries, stronger metadata storage and better performance. These and future products are being driven by the needs of strategic planning and customer relationship management applications for businesses such as yours.

Understanding more about temporal database technology, ask your IT department the following questions:

- How do we use historical data in its decision making process?
- How well does our current database system preserve the historical context of our data?
- Which of our existing applications could benefit from improved historical data context and stronger temporal query capability?
- What issues do we have with temporal data for applications we are planning?

John Bair is Co-Founder, CTO and Sr. VP Product Development at ***if…*** Mr. Bair has spent the past 15 years developing technology for software start-up companies that he co-founded. Mr. Bair has two patents pending in temporal database technology.

Michael Soo is a Senior Technologist at ***if…*** Dr. Soo has performed research and development in temporal database systems for the past ten years, and was an Assistant Professor at the University of South Florida where he devised techniques to model temporal data and developed algorithms to evaluate temporal queries. Dr. Soo holds a B.S. in Mathematics/Computer Science from UCLA and the M.S. and Ph.D. degrees in Computer Science from the University of Arizona.

if... contact information***:***

www.iftime.com

510-864-3480 (voice)
510-864-3489 (fax)

