Informix®

Managing Time Series Data in Financial Services Applications: The Informix TimeSeries DataBlade Module

Table of Contents

Introduction

In today's competitive financial services industry, fast access to critical business information is a key factor in achieving and sustaining profitability. The ability to manage complex, time-sensitive information and turn it into *smart data* speeds decision making, increases your competitive advantage, and maximizes the return on your investment in information technology.

Financial applications require high-performance storage, retrieval, and analysis of *time series* data. A time series is a time stamped series of data entries, such as minute-by-minute reports of stock prices and trading volumes. Time series data is used extensively in the financial world for corporate financial reporting, stock prices, bond yields, and derivative securities. Enterprises must be able to effectively manipulate that data by combining proprietary business models and algorithms with standard calculations. For example, a financial enterprise might need to perform the following types of analysis on time series data:

- 30-day moving average computations;
- portfolio summary maintenance;
- risk-factor calculations;
- advanced charting support; and,
- on-line individualized analysis and reporting functions.

Although traditional relational database management systems (RDBMSs) can manage standard time series data types by storing one row per time stamped data entry, this leads to poor performance and inefficient data storage. Storing time series as BLOBs improves performance, but at the cost of huge amounts of custom coding, because the data is sent to the client application for processing. Developing or changing applications is difficult because many aspects of business semantics must be managed by each client application.

Some financial institutions have turned to non-relational database systems to support time series data. However, non-relational database systems are often more costly to develop and maintain than those using the relational model. The lack of a common query language such as SQL greatly increases the complexity of development and integration.

Non-relational, file-system-based time series implementations suffer from other limitations, including the lack of generality and extensibility, poor support for ad hoc queries, pre-defined limits on the type and structure of the data, and an inability to combine time series data with other information.

Informix® Internet Foundation.2000™ and Informix's advanced, extensible DataBlade® module technology are the answer. DataBlade modules are software modules that can be easily plugged into Informix Internet Foundation.2000 to extend the server with domain-specific data management expertise. Informix Internet Foundation.2000 enables you to move business processes and logic from the application into the database, increasing productivity and simplifying the process of making changes.

Furthermore, Informix Internet Foundation.2000 can manage any type of information, enabling you to deploy new types of applications seamlessly across Internet, as well as traditional client/server systems. You have the flexibility to custom-tailor your database server to adapt to new business requirements as they evolve, putting you in control of your own data management destiny.

When you plug the Informix TimeSeries DataBlade module into Informix Internet Foundation.2000, you gain instant access to pre-defined time series data types and routines to manage your time series data. An open and flexible interface means that you can still use traditional tools to manipulate and analyze the data. The unique advantages of Informix's technology include:

- the ability to store time series data in its native format;
- the ability to associate active calendars to define pattern activity;
- the flexibility to model business data; and,
- the ability to encapsulate organizational logic into database logic.

The TimeSeries DataBlade module extends the database management system to "understand" time series data as a first-class data type—managed as an intelligent object that the database recognizes and manipulates efficiently. The TimeSeries DataBlade module also provides a rich set of time series analysis functions, enabling developers to focus on interpretation of the data rather than on the details of data management. Sophisticated data evaluation models can be added, modified, and combined with other functions within the database server, and executed in either the database or the client workstation, enabling any number of applications to reuse these models.

Technical Overview

The TimeSeries DataBlade module takes advantage of Informix Internet Foundation.2000's architecture, which provides full support of complex data structures within the database server. The TimeSeries DataBlade module extends Informix Internet Foundation.2000 to make time series data a first-class data type recognized and manipulated by the database server, rather than just being stored as a simple block of data.

TimeSeries Data Type

The TimeSeries data type acts as a type constructor for a time series subtype. The resulting TimeSeries(subtype) data type is a collection of row data types. A row data type consists of a group of named columns, of the same or different data types, within a single database column. Figure1 illustrates the time series data type.

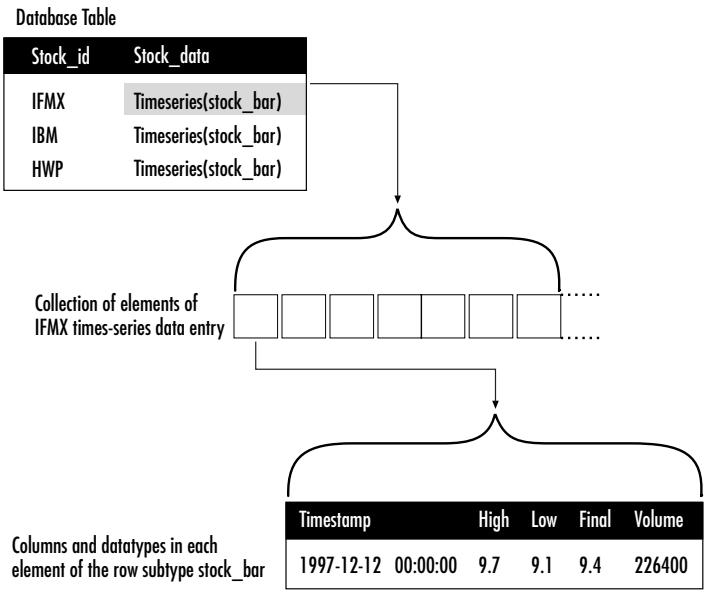


Figure 1: Time series data type.

The database table in this example has two columns: a **stock_id** column containing the stock name and a **stock_data** column containing the time series. Each row in the table contains a different time series. In this example, all three rows have a time series of subtype **stock_bar**.

The time series row subtype defines the structure of the elements. Each element is a row data type containing columns, beginning with a time stamp column. The time stamp has the precision to store a time range from one year to 10 microseconds. Time stamps must be unique because the elements are ordered by time stamp.

The number of the columns in an element is not restricted. The preceding example shows columns for high, low, and final stock values, as well as the volume of stock traded. In this example, each time series of type **stock_bar** necessarily has the same columns; however, each time series can have its own calendar and time series starting time.

Time Series Data Storage

Time series data can reside in the table column, or, if the data is larger than a user-defined threshhold, in a *container*. A container is a structure that the TimeSeries DataBlade module creates and maintains to hold time series data. When a time series has data in a container, only a small header is left in the original table.

Whether the time series data is held entirely in the table column or in a container, the header holds information about the time series data and can also contain user-defined metadata. User-defined metadata allows the time series to be self describing. The metadata can be information usually contained in additional columns of the table, such as the name of the stock, the type of time series, or exceptional conditions about the time series. Keeping this type of information in the metadata is an advantage when using API routines because it is easier to retrieve metadata than to pass additional columns to the routines. When you create a time series, you can specify whether or not to include metadata.

Kinds of Time Series

There are two kinds of time series data types: *regular* and *irregular*. A regular time series data type stores data at fixed intervals, while an irregular time series data type stores data for arbitrary time points. Regular time series data types are appropriate for applications that record entries at predictable time points, such as daily closing stock prices. Irregular time series data types are appropriate when the data arrives unpredictably, such as applications that record every stock trade as it happens.

Each element in a time series represents data associated with a time interval. The time stamp associated with an interval marks the beginning of the interval. Regular elements *persist* only for the length of an interval as defined by the calendar associated with the time series data, and missing elements are null. Irregular elements persist until the next element; there are no null elements.

The main distinction between the two kinds of time series data types is that irregular time series data types lack the concept of an *offset*—a mapping between the time point associated with an element and its position relative to the start of the time series. Because of this, regular time series data types are stored more efficiently than irregular time series data types. Time stamps are not stored with regular time series data; instead they are computed from an offset.

Both regular and irregular time series data types have many of the same characteristics; therefore, most time series routines can be applied to both types of time series data. Time series data is defined as regular or irregular when it is created.

Time Series Data Organization

Time series data is controlled by calendars using the ***calendar*** and ***calendar pattern*** data types.

A time series *calendar* defines a set of valid times at which the time series data type can record data and determines when and how often entries are accepted. For regular time series data types, calendars create the vector structure by converting time stamps into offsets. Irregular time series data types do not use offsets, but still use calendars to define the valid entry times.

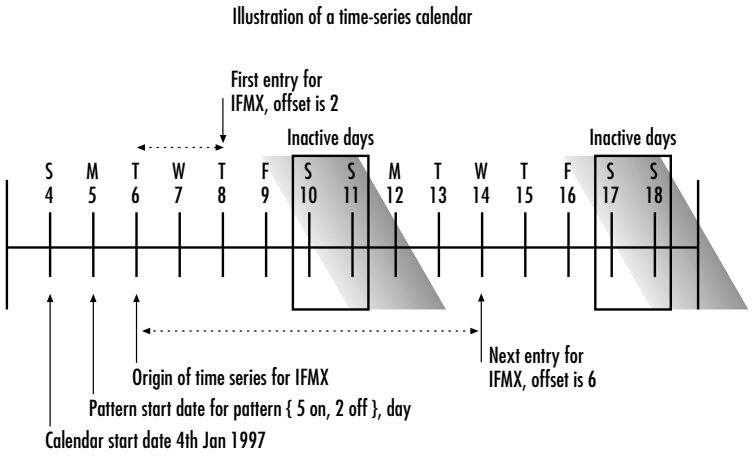
Calendars are represented by the calendar data type and have the following properties:

Start date	The date the calendar begins.
Pattern	A regularly occurring set of valid and invalid time intervals.
Interval	The calibration of a calendar pattern.
Pattern start date	The date the calendar pattern starts.

Each calendar has a *calendar pattern* of time intervals that are either valid or invalid, with the beginning of the calendar pattern specified by the calendar pattern start date. The calendar must contain a calendar pattern starting time as well as a calendar starting time because calendar and calendar pattern starting times might not coincide. For example, most years do not begin on the first day of the week.

Figure 2 illustrates these concepts:

Figure 2: Time series calendar.



In the previous example, the calendar is defined with a start date of January 4, 1997. However, the pattern of five working days of valid entries (with two weekend days where entries are not taken) starts with Monday, January 5, 1997. The example entries for IFMX, which is a regular time series data type, demonstrate that only an offset is required from the origin to record the date element. As a result, the offset for the two recordings is calculated below:

M	T	W	T	F	M	T	W	T	F
.	0	1	2	.	.	.	.	.	.
.	0	1	2	3	4	5	6	.	.

SQL and API Routines

The TimeSeries DataBlade module and Informix Internet Foundation.2000 provide both SQL and API routines to manipulate time series, giving you the flexibility to call routines from interactive query processing tools such as DB-Access or from within an application on either the client or the server machine.

The TimeSeries DataBlade module provides routines to manipulate:

- Calendars
- Calendar patterns
- Time series data

Calendar and Calendar Pattern SQL Routines

Calendar and calendar pattern SQL routines perform the following operations:

- create the intersection of calendars or calendar patterns; and,
- create the union of calendars or calendar patterns.

TimeSeries SQL Routines

Time series SQL routines perform the following types of operations:

- return information on the time series definition;
- manipulate elements, either as individuals or as sets;
- create and load time series data;
- perform statistical calculations;
- perform arithmetic calculations;
- manage containers;
- convert between time stamps and offsets; and,
- extract periods of time series.

Informix DataBlade Module API Routines

The Informix DataBlade module API is the application programming interface for Informix Internet Foundation.2000. It can be used to develop client and server applications that access data stored in extended data structures. The DataBlade module API can be used to build functions that perform the following operations:

- data handling;
- session, thread, and transaction management;
- query processing;
- function execution;
- memory management;
- manipulate save sets of data;
- exception handling; and,
- operating system interfaces.

Informix Virtual Table Interface (VTI)

An Informix virtual table interface (VTI) enables tools that do not understand extended data types to use time series (and other rich) data.

The VTI takes the data encapsulated in the time series data type and produces a virtual relational table containing the same data. The virtual table has the same schema as the base table, except for the time series column. The time series column is replaced with the columns of the individual time series elements. The virtual table is not a real table stored in the database, so there is no duplicate storage of data. The VTI gives you a way to use standard load utilities for time series data and makes it easy to query time series data using standard tools that don't deal with extended data types. You realize the benefits of time series (calendar support, access speed, and so on) while also using standard tools that don't understand extended data types.

Benefits of the Informix TimeSeries DataBlade Module

The TimeSeries DataBlade module provides the ultimate platform for financial time series innovation. You can use this technology to improve your time to market, gain high performance at the lowest possible cost, and change on demand as your business needs change—keeping you ahead of the competition. Benefits of the TimeSeries DataBlade module include:

- fast, low-cost development;
- ease of integration;
- industry-leading performance and scalability; and,
- investment protection through extensibility.

Fast, Low-Cost Development

While the benefits of object-oriented development methodologies are well documented, the critical advantage is rapid, iterative, and modular application development. The extensibility of Informix Internet Foundation.2000 enables you to use standard object-oriented techniques to extend the database and take advantage of already created extensions. Because you can extend the server using SQL, C, C++, and Java, you are able to leverage existing expertise in industry-standard development languages.

One of the most powerful development features of the TimeSeries DataBlade module is its calendar functions. Using the TimeSeries DataBlade module, you can define business calendars and create time series data based on any of these calendars—for example, the FTSE calendar. It is also simple to subsequently overlay an alternative calendar and extract particular interest points, such as the average volume of transactions of a specific stock instrument over a six-month period.

Ease of Integration

Informix Internet Foundation.2000 enables you to integrate time series data with alphanumeric data (numbers and text) and complex data (such as research reports, analytics, and business newsfeeds) all using a common query language—SQL. The TimeSeries DataBlade module makes it easy to incorporate time series data into new or legacy database applications.

The underlying architecture enables you to quickly develop new, innovative financial applications utilizing time series and deliver them to market using channels such as the Internet. When you develop TimeSeries DataBlade module applications, the core functionality is implemented in the server and can therefore be used by any Informix application, including Informix Web DataBlade module Application Pages (AppPages).

Because you can extend the server using C, C++, and Java, you are also able to integrate new applications with legacy application frameworks quickly.

Industry-Leading Performance and Scalability

The TimeSeries DataBlade module benefits from the multithreaded, parallel processing capabilities of Informix Internet Foundation.2000, providing superior database performance and scalability for both high-volume transactions and resource-intensive queries.

With Informix Internet Foundation.2000, each database request is dynamically distributed across a configurable pool of virtual database processes and dynamically allocated across all available hardware processors to deliver the highest possible system performance. In addition, the extensibility of Informix Internet Foundation.2000 means that the execution of business rules and other complex data analyses are performed inside the database server, right next to the data. This design maximizes the use of all available hardware resources and minimizes the impact on limited network resources.

The TimeSeries DataBlade module also takes full advantage of Informix Internet Foundation.2000's industry-leading ability to scale from small, gigabyte-sized systems to massive, terabyte-level implementations. Because of the unique ability to store time series data as a first class data type, Informix Internet Foundation.2000 overcomes the limitations encountered by other relational and non-relational approaches to managing time series data.

Investment Protection Through Extensibility

The TimeSeries DataBlade module provides a flexible framework that enables you to easily expand your database application capabilities and more closely model data and business rules to align with your organization's business processes.

For example, the Informix DataBlade API can be used as a set of building blocks to develop customized analytics, data loading modules, application interfaces, and more. The Informix DataBlade API also enables developers to create new data types and develop their own functions for business rule processing and complex time series analysis using C, C++, and Java—from inside the database server.

Customer Example: Investment Management

Customer Profile

The following example is drawn from an investment advisory firm that manages over $1 billion in assets for institutional clients and wealthy individuals. The firm manages stock portfolios using quantitative techniques designed to perform better than financial indices such as the S&P 500.

The firm focuses on researching financial assets and engineering structured portfolios. It maintains research data on a broad population of stocks, across many characteristics of those stocks, with over 20 years of historical data. Real-time data feeds are received from stock exchanges by satellite and integrated with historical information. Financial models select optimal positions and recommend stock trading opportunities based on this data.

Requirements

Much of the data for financial assets is structured as time series information at regular (monthly or quarterly observations) or irregular (trade and bid/ask changes) time intervals. The data extends across several thousand securities.

The database is updated daily. Data is made available to risk/return optimization programs that consider the correlation, risks, and anticipated returns for each stock to form optimal portfolios. These portfolios are structured to have lower risk and higher expected returns than the index benchmarks.

The firm needed an information technology solution that would help it achieve the following objectives:

- increase the scope, timeliness, and readability of usable research data; and,
- increase the quality and precision of the data used in the investment management process.

Solution

Prior to using Informix technology, the firm used a proprietary software system based on flat data files. When they deployed Informix Internet Foundation.2000 with the TimeSeries DataBlade module, the firm found that recording time series information on 3,000 securities across 20 years (240 months) and 65 different characteristics could be represented as a single table of 3,000 rows. Time series data makes up the bulk of their database, approximately 400 megabytes.

The firm reported the following advantages after deploying the Informix-based solution:

Faster Data Loading

The first tasks were to load the previous 20 years of data into the new database and develop a loading process for future data feeds. The flat file format was not suitable for doing a bulk load—regenerating all 20 years of data into a useful format would have required tremendous effort. The solution was to write a simple load utility running inside the database server. The utility, written as a server function, understood the flat file format and converted the ASCII data into time series elements using the well-defined Informix DataBlade API. Using a server function meant that the data files did not need to be regenerated or even converted into a more usable format. It also resulted in high performance because many interface layers of software were circumvented. Using this technique, the loading of the entire database, including time series, ancillary tables, and all indices, was quickly completed.

Optimal Performance

Once the data was loaded, the next step was to create a prototype of the risk/return optimizer application. Prior to using Informix, the optimizer application typically required three to four minutes from start to finish. Using the TimeSeries DataBlade module along with a server function tailored specifically to their needs, the optimizer application completed these transactions in less than 10 seconds.

In this application, a query selects the targeted time series values, clips them for a particular time interval, then casts the required stock characteristics into parallel arrays (the format the optimizer requires) before returning the data to the client. This straightforward use of time series functionality highlights the DataBlade module's ability to access specific elements very quickly.

The clip operation takes two timestamp values converted to time series offsets using the time series calendar. The offset pinpoints exactly an element's location in the time series, enabling the clip operation to jump directly to that location to begin forming the result. The operation calculates a direct mapping from timestamps into time series offsets. Look-ups are guaranteed to perform in constant time—as the database grows in size, the amount of time required for the look-up operation stays the same.

Using a server function to convert time series elements into arrays minimizes the amount of data transmitted over the network to the client, which is important because network transmissions involving the copying and transmitting of data can be expensive. The extensibility of the Informix Internet Foundation.2000 database server also allows application developers to push more and more processing tasks closer to the data, reducing the data path and the amount of data flowing across the network.

Maximum Scalability

After the initial prototyping was successfully completed, the firm's next task was to run the application at full scale. Time series data integrated into Informix Internet Foundation.2000 scaled well—in fact, database operations completed in the prototype exhibited the same performance characteristics when deployed in the full production database. The firm discovered that database access performance with time series data does not depend on data size and therefore does not degrade as the amount of stored data increases.

Software Reuse and Faster Deployment

Once the optimizer application was up and running, the firm wanted to run their factor models application. Although this is a separate application, the firm was able to reuse server functions developed for the optimizer application. Because of the extensibility of Informix Internet Foundation.2000, these tools reside in the server and are readily available to be reused for any new application development requirements.

Ease of Management for Complex Data

With the TimeSeries DataBlade module, Informix Internet Foundation.2000 is extended to understand time series as a first-class data type in the database. This enabled the firm to store 65 monthly stock characteristics for 3,000 securities over 20 years in just 3,000 rows. In other RDBMSs, this would require a table of 3,000 x 240 = 720,000 rows. If the firm chose to store data for the same set of measures over the same 20 years on a daily basis, rather than monthly or yearly, Informix Internet Foundation.2000 would still need the same 3,000 rows—only the length of the individual time series would grow larger. Compare this to other relational systems that would require a table containing 3,000 x 20 x 365, or 21,900,000 rows.

The firm was also able to add sophisticated data evaluation models within the database. These models can be executed either in the database server or on the client workstation, supporting shared use by all application developers. The firm was also able to integrate third-party graphing and data visualization tools, as well as traditional personal productivity tools, such as Microsoft Excel.

Summary

This firm was able to fully meet its business objectives, which were to:

- increase the scope, timeliness, and readability of usable research data; and,
- increase the quality and precision of the data used in the investment management process.

The preceding example demonstrates the TimeSeries DataBlade module's ability to provide high-performance management of time series data so that you are able to focus on the interpretation of data rather than on the details of efficient data management. By encapsulating functions, data types, and rules, Informix Dynamic Server™ 2000 enables you to move business practices from the application into the database, increasing productivity and simplifying changes.

Conclusion

Forward-looking companies in the financial industry gain a competitive advantage by using Informix Internet Foundation.2000 and the TimeSeries DataBlade module. The TimeSeries DataBlade module greatly expands the functionality of the database by adding sophisticated support for the management of time series data.

The time- and calendar-based access methods provided by the TimeSeries DataBlade module offer performance superior to simple relational tables. In addition, in the TimeSeries DataBlade module, time series is a native type, and the supplied functions can appear anywhere in an SQL statement. This simplifies design and coding, and enables ad-hoc complex queries on time series, other complex data (such as text or spatial), and regular relational data and provides high-performance storage, access, and modeling of time series data.

About Informix

Informix Corporation, based in Menlo Park, California, provides innovative database solutions that assist the world's major corporations attain competitive advantage. Informix is widely recognized as the technology leader for corporate computing environments ranging from small workgroups to very large parallel processing applications. Informix's database server, application development tools, superior customer service, and strong partnerships enable the company to be at the forefront of major information technology solution areas including data warehousing, high-performance OLTP, and Web/e-commerce.

For more information, contact the sales office nearest you or visit our Web site at http://www.informix.com.

Regional Sales Offices

Asia/Pacific
65 298 1716

Canada (Toronto)
1 416 730 9009

Europe/Middle East/Africa
44 181 818 1000

Federal
1 703 847 2900

Japan
81 3 5562 4500

Latin America
1 305 265 7545

North America
1 800 331 1763
1 650 926 6300

Informix®

4100 Bohannon Drive
Menlo Park, California 94025
1 650 926 6300
World Wide Web: www.informix.com

© 1999 Informix Corporation. All rights reserved. The following are trademarks of Informix Corporation or its affiliates, one or more of which may be registered in the U.S. or other jurisdictions: Informix®, DataBlade®, Informix Internet Foundation.2000™, and Informix Dynamic Server™.

Printed in U.S.A. 7/99 000-21849-70