Using the CALANDA[1] Time Series Management System[2]

Werner Dreyer, Angelika Kotz Dittrich, Duri Schmidt
{dreyer, dittrich, schmidt}@ubilab.ubs.ch

UBILAB, Union Bank of Switzerland, Zurich

Abstract

The analysis of time series in financial and scientific applications requires database functionality with complex specialized modeling capabilities and at the same time an easy-to-use interface. We present the time series management system CALANDA which combines both, a powerful dedicated data model and an intuitive GUI. The focus of this paper and the demonstration is to show how CALANDA is accessed by end users.

1 Introduction

In financial institutions such as banks, time series are important for areas like economic and financial research as well as for various non-research related activities like portfolio management. The reason for this importance is the increasing application of empirical methods with an ever growing amount of data [DKS94]. Scientific applications in general make extensive use of time series, an example domain being earth sciences where a huge number of time series related to temperature, rainfall etc. are analyzed.

Time series management faces several challenges. The following are relevant for the scope of this article:

- Time series encountered in real-world applications have complex structure. Furthermore, the requirements to process these time series - like statistical methods or time scale conversion - are very demanding. Therefore, a time series management system (TSMS) must have adequate modeling power which can only be covered by a sophisticated data model.

- In large time series bases, it is often difficult to find the time series needed for a specific task or project. The TSMS must provide powerful and efficient means to search the data space for relevant time series.

- Users of a TSMS are specialists within their problem domain, but they are not necessarily computer experts trained in using a DBMS. Therefore, a comfortable, easy-to-use interface is essential.

1 CALANDA is the name of a mountain (2805m) in the Grisons, Switzerland.
2 This paper has been submitted to the 1995 ACM SIGMOD Exhibits Program.

In this paper, we show how to cope with the problem that time series management systems need to provide a complex data model and at the same time a user-friendly interface. We demonstrate how the mapping between the components of the data model and the elements of the interface is done in our TSMS CALANDA and how the system presents itself to end users.

There have been various approaches to time series management (see [DKS 94] for a detailed discussion of the state of the art). Time series management based on conventional systems like relational DBMS or even simple files does not satisfy the complex data modeling requirements of the domain. Other approaches can be found in the fields of temporal, scientific and statistical databases, like e.g. the model published by Segev et al. [SS 93] [SC 93]. There are even some commercial TSMS products like FAME [FAME 90]. These systems have specific time series-oriented data models. However, their data models are mapped onto a command-oriented interfaces, users having to master a language like SQL or even a product-specific language. In our opinion, this approach is not suitable for end users.

In contrast to these approaches, CALANDA has been designed under the assumption that application specialists will use it rather than computer experts. Therefore, our data model is accessed via a graphical user interface (GUI). Users familiar with spreadsheets or graphics-based database software need very little education to use the system productively. In addition, CALANDA concentrates on the functionality for time series management while other TSMSs are enhancements of existing DBMSs. This focus on time series restricts the complexity of the model to the constructs really needed for the problem domain, not burdening the user with the additional intricacies of a general-purpose data model.

CALANDA has been implemented using object-oriented technology. For the underlying platform, we seamlessly integrated the object-oriented DBMS ObjectStore and the C++-based application framework ET++ [WGM 89]. This platform provides persistent storage of objects, basic database functionality as well as a framework for application and interface building. Using that, we implemented the TSMS kernel with a specialized type system, the interface layer as well as a number of other system components. CALANDA runs under Unix on Sun workstations.

Section 2 of this paper gives a survey of the main characteristics of our TSMS. Section 3 describes the various parts of the demonstration.

2 Survey of the CALANDA Time Series Management System

In the following, we will give a survey of the main characteristics of our time series management system **CALANDA** which are relevant for this article:

- **Specialized object-oriented data model with time series and group as root classes:** Our data model is object-oriented and it is specialized for the domain of time series management. It has two base classes: the time series class and the group class. These classes have a rich functionality adapted to the problem domain. Therefore, a user is not concerned with the implementation of the basics of time series or groups but only with their adaptation to his or her special needs (see [DKS 94] for a detailed discussion of the data model).

- **Graphical user interface and API for other applications:** The primary user interface of our system is a graphical user interface. Users already knowing how to use a spreadsheet or a database software with a graphical user interface need very little education to use the system productively. Furthermore, an API lets other applications directly load and store data in a time series base without using file transfer.

- **Multivariate time series with query capabilities and time scale conversion as basic abstraction:** Multivariate time series are pivotal for the problem domain addressed by this project. For this reason, the basic time series class is designed for the modeling of multivariate time series. Furthermore, queries can be executed on time series and time scale conversion can easily be done. This built-in functionality solves common time series problems without requiring the user to make any further implementation.

Our requirements analysis showed, that real world problems require a rather complex time series model. Therefore, in our system every time series consists of two parts (see Fig. 1). The first part is the header. It contains data concerning the whole time series such as its name. A header element is either a simple value as, for example, an integer, a float or a time point, or it is an array of simple values. The second part is a sequence of equally structured events. Every element of an event is again either a simple value or an array of simple values.

Name:	UBS registered
Security_number:	136 102
Total_trading_vol.:	90869
Quality coefficients:	0.95 0.87 0.92 0.94

Date	Open	Close	Prices	Daily_vol.
20.12.93	319	323	319, 320, 318, 325, 324, 324, 323	23249
21.12.93	322	328	322, 320, 325, 324, 329, 327, 328	19403
22.12.93	328	330	328, 325, 328, 326, 324, 329, 330	35845
23.12.93	331	328	331, 325, 328, 325, 324, 324, 328	12372

Fig. 1: An example time series

As one would expect in an object-oriented data model, time series also have methods. This makes it possible to implement time series with customized behavior as required by the problem domain.

- **Groups as an effective categorization and aggregation instrument:** A large time series base may contain thousands of time series. Without partitioning all these time series according to different criteria, it would be difficult for users to find the time series relevant to their work. In our data model, groups serve the purpose of partitioning. An example of a hierarchy of groups partitioning a stock time series base can be seen in Fig. 2.

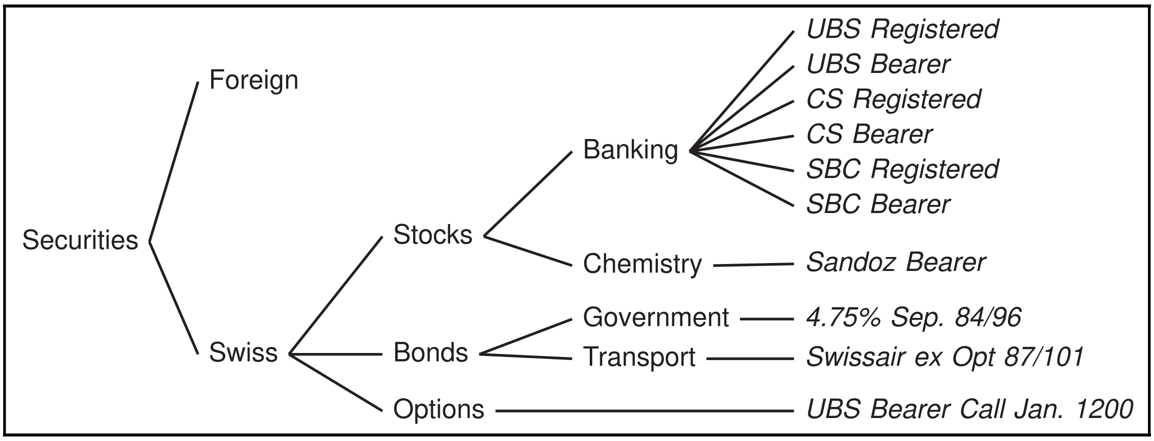

Fig. 2: An example group hierarchy

Similarly to time series, also groups have a header which contains data concerning the whole group. Every element of the header is again either a simple value or an array of simple values. Furthermore, groups have a possibly heterogeneous member set consisting of arbitrary time series and groups (see Fig. 3).

Queries may be executed on the member set and set operations are provided to manipulate it. Groups do not only function as a means of categorizing. Because of their methods, groups can also be used as a flexible means to do computations on their members, such as aggregate some value over all the members.

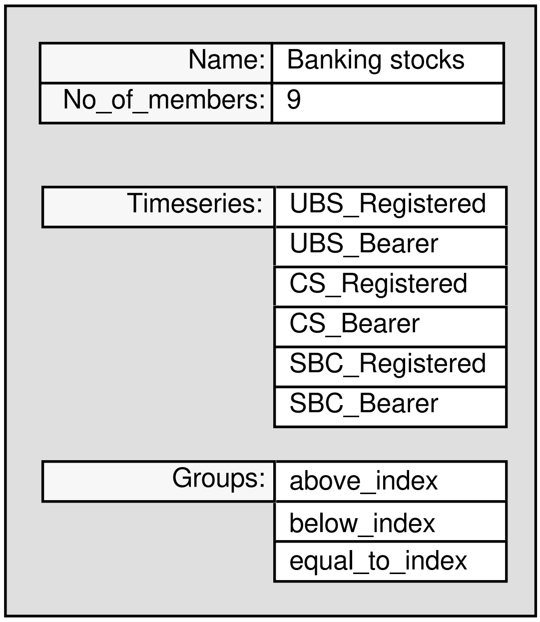

Fig. 3: An example group

Together, all these data modeling features form a sophisticated data model as required by the application domain. The problem of the user interface is to present the various features of the system in such a way that the complexity of the data model are hidden as much from the user as possible and that the usage of the system is intuitive and productive.

3 Using the System

The interface to CALANDA has been designed according to three major principles:

- It is an interface with intuitive graphical elements and a menu-based way of manipulation. The elements of the data model are directly mapped to graphical components, the operations applicable in the model are represented by menu items.
- The interface has been designed to resemble well-known paradigms like spreadsheets or 4GL tools for relational databases.
- Tools in this interface serve two purposes: They provide an overview of the data base and they allow to retrieve and/or manipulate individual elements. Tools of these two kinds exist for meta data, i.e. information on classes, as well as for primary data, i.e. time series and groups.

In the sequel, we will describe how an end-user is going to work with CALANDA via the graphical user interface. We will explain the most important activities of the user: navigating through a time series base, browsing a group, browsing a time series, querying a time series and defining a new time series or group class. The demonstration of the system will follow these lines.

3.1 Browsing a time series base

After opening a time series base, the user can choose between two kinds of browsers to survey the contents of this time series base and navigate among its groups and time series. Figure 4 shows the first kind of browser, the base browser. It is a list view on the group structure, similar to a list view on a file system. In fig. 4, part of a time series base *tsb1* is shown with group *swiss securities* to the left, selected subordinate groups *stocks* and *banking stocks* are displayed to its right. Group names within the lists are preceded by an asterisk. The user may vary the number of lists on display as well as navigate through the group structure by selecting subgroups or scrolling left and right between the lists.

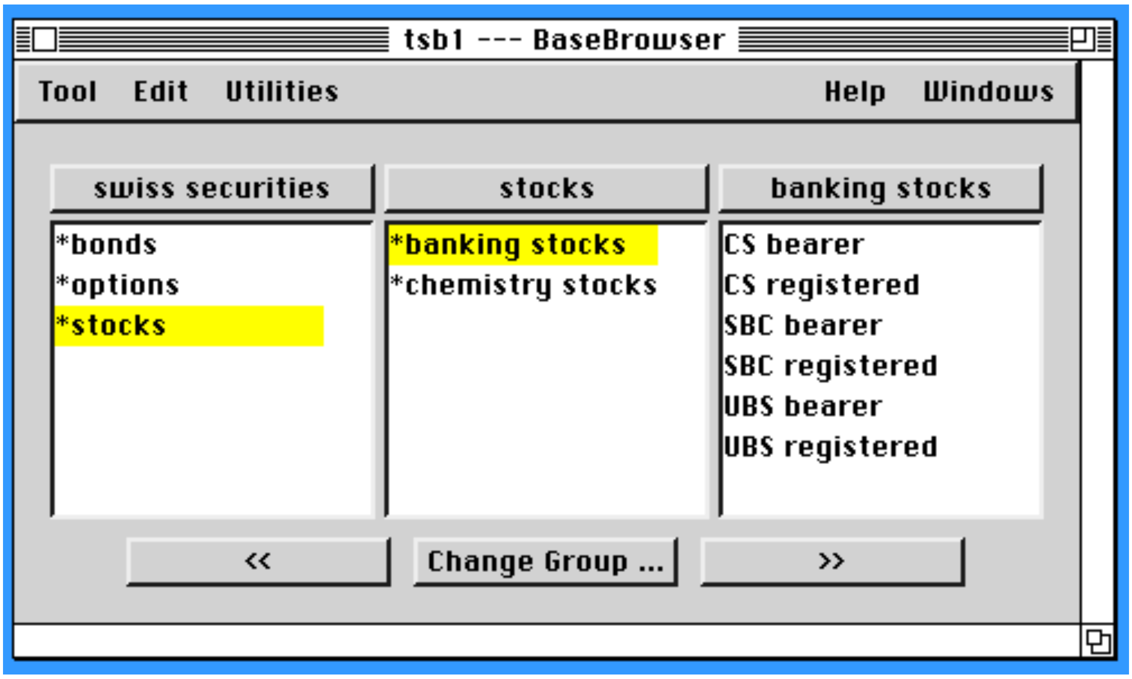

Figure 4: Base browser on time series base *tsb1*

An alternative way to look at the time series base is the tree browser shown in fig. 5. It shows the group structure in a tree-like fashion. The main categories of the time series base, i.e. the groups that are not themselves members of any other group, are shown to the left underneath "Roots". In the example, the user may choose to see the time series base categorized either according to security type or according to performance.

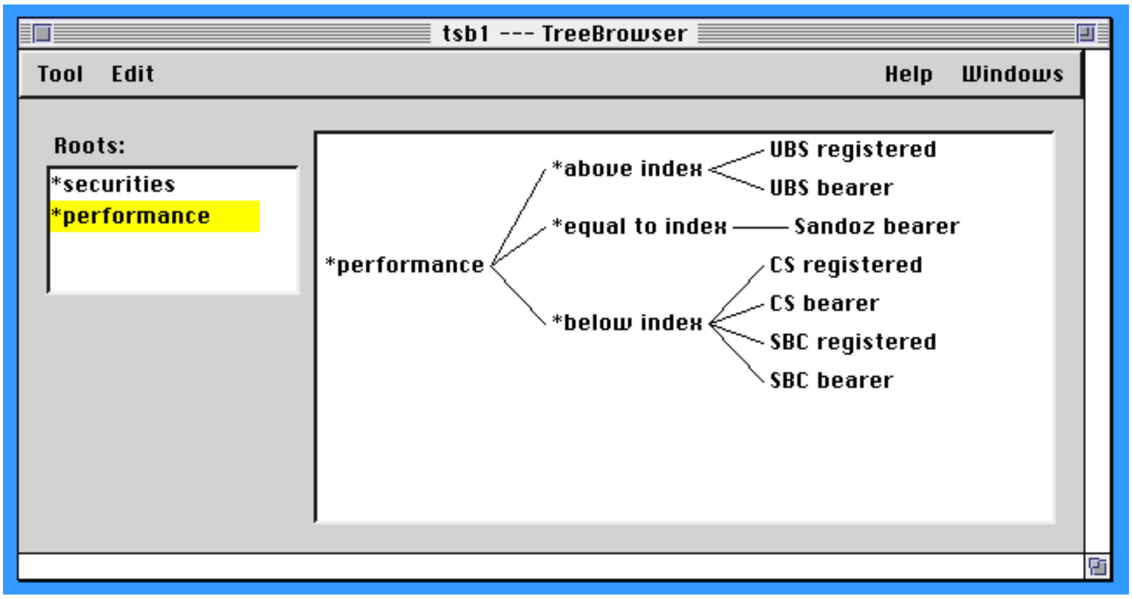

Figure 5: Tree browser on time series base *tsb1*

3.2 Browsing a group

From the base browser or the tree browser, the user can select a group and by double-clicking it bring up a group browser. The group browser (fig. 6) shows information about one group, namely its header fields (to the left) and the time series and groups which constitute its members.

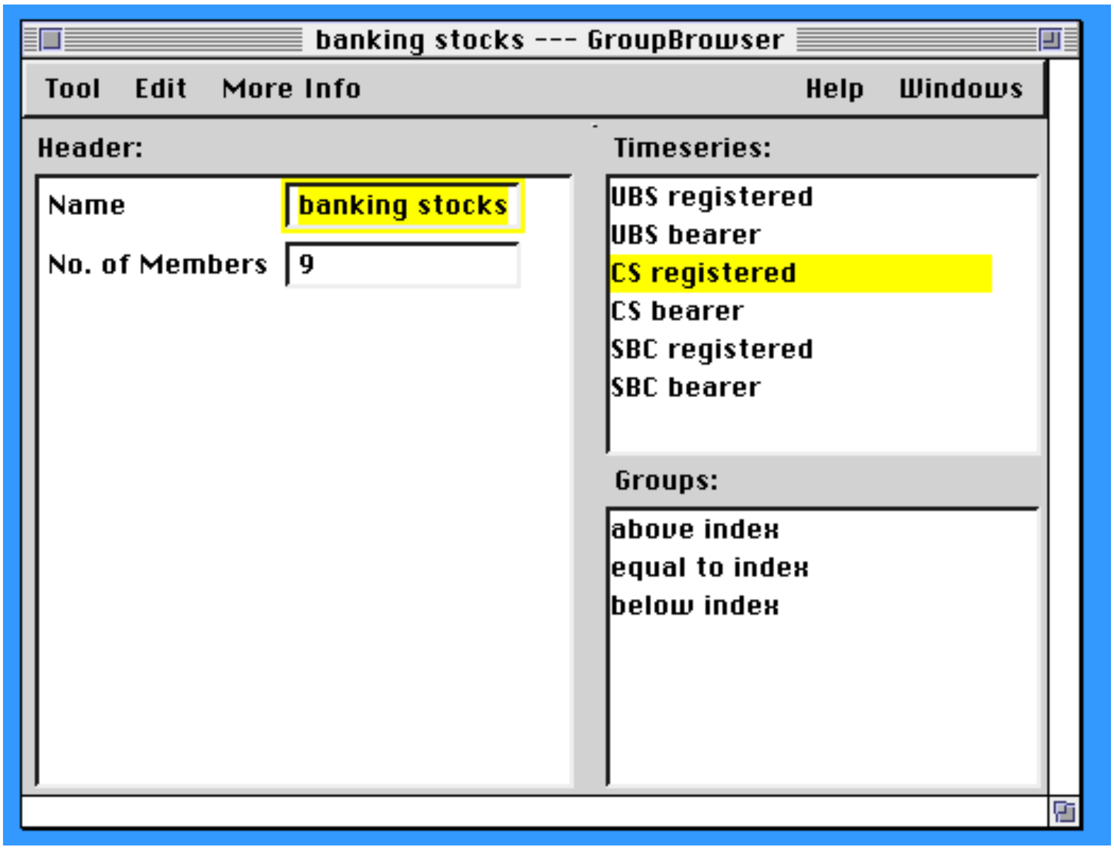

Figure 6: Group browser on group *banking stocks*

3.3 Browsing a time series

By selection from the base, tree or group browsers, the user may choose a time series for closer inspection. For this purpose, a time series browser as in fig. 7 is provided. This browser shows the header fields of the time series as well as the sequence of events[3]. There are various ways for a user to select and permute attributes for presentation. Time series data may be modified directly by entering data via the time series browser. Events may be appended or deleted interactively. The event display can be operated on very much like a spreadsheet, a presentation which is certainly well known to most end users.

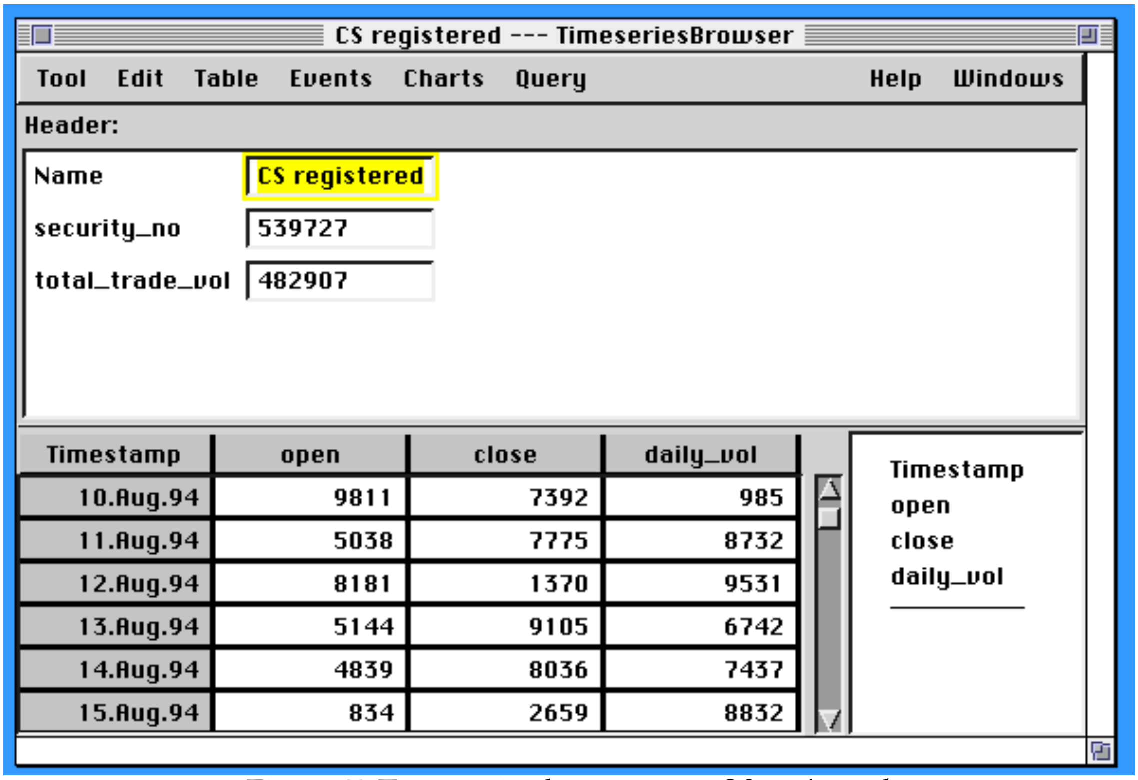

Figure 7: Time series browser on *CS registered*

Beside the textual presentation, the end-user may ask for a time series to be drawn in a charting tool (fig. 8). The tool allows to chart all or part of the variables over time or to draw a scatter view correlating arbitrary variables. In fig. 8, the close and open price are charted over time, the other variables are currently not displayed. There are a lot of options to vary the type of chart, color, range of displayed values etc. Modification of values can also be done by direct manipulation via the charting tool.

3 Note that array attributes are not displayed in figure 7. We are currently working on an additional browser for arrays.

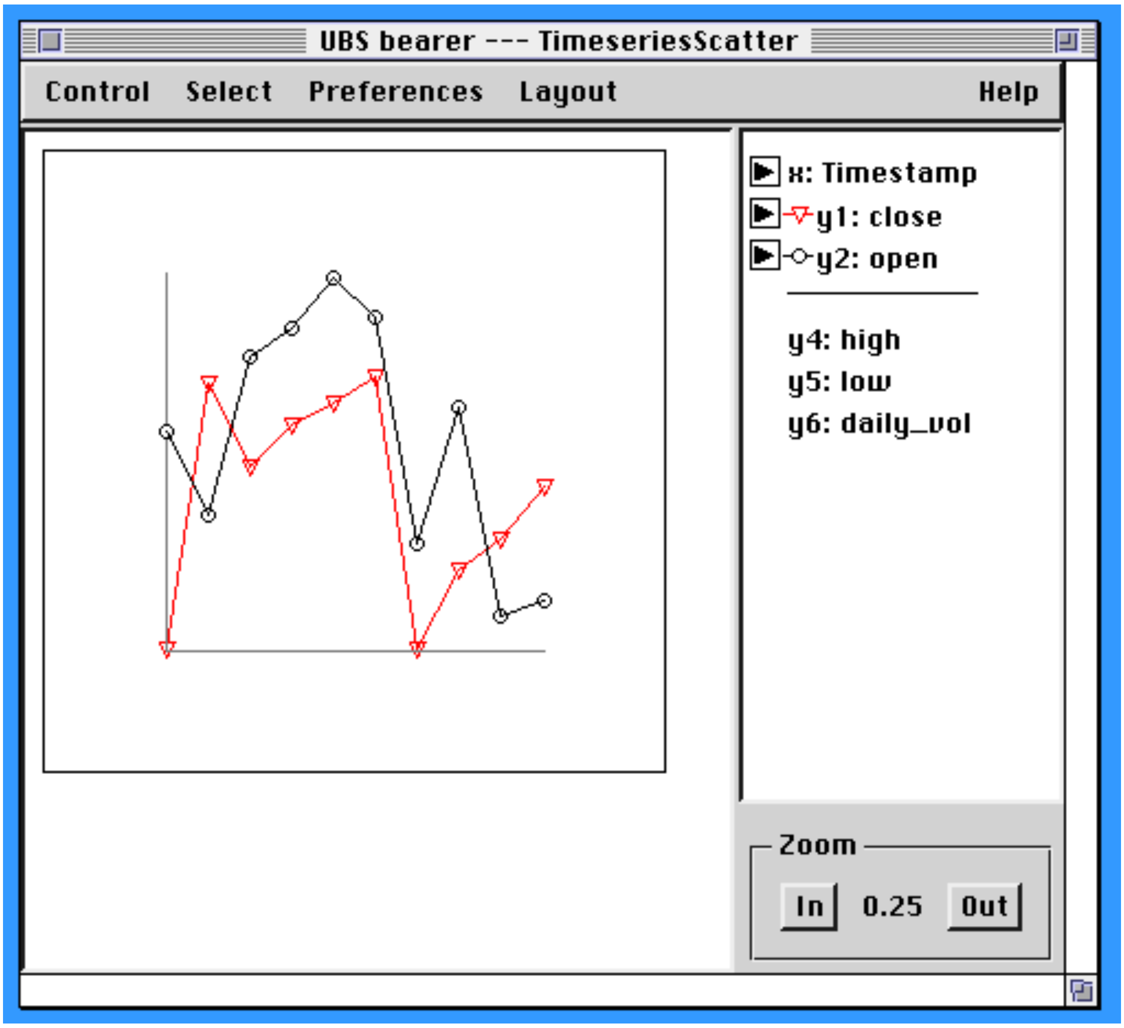

Figure 8: Charting time series *UBS bearer*

From the time series browser, the user may use a straightforward query facility to retrieve selected events from the time series. Fig. 9 shows how a simple query is entered to select only the events with an opening price larger than 2000.

3.4 Defining a time series or group class

There are further tools currently under development by which the user can interactively look at and define time series and group classes. The class browser gives a survey of all classes in a time series base. With the help of the class editor, the user can specify the name and type of header and event attributes, the calendar underlying a time series etc. A further editor is provided to define the methods applicable to the instances of a class. From a time series or group browser , the user can of course go back to the class browser to have a look at the type information.

3.5 Further Tools

A number a other GUI components are also under development. To name the most important, these are a tool for separately displaying and editing array attributes, a system browser that allows to navigate among different time series bases, a tool for import/export specifications and a "listener" for directly entering method calls.

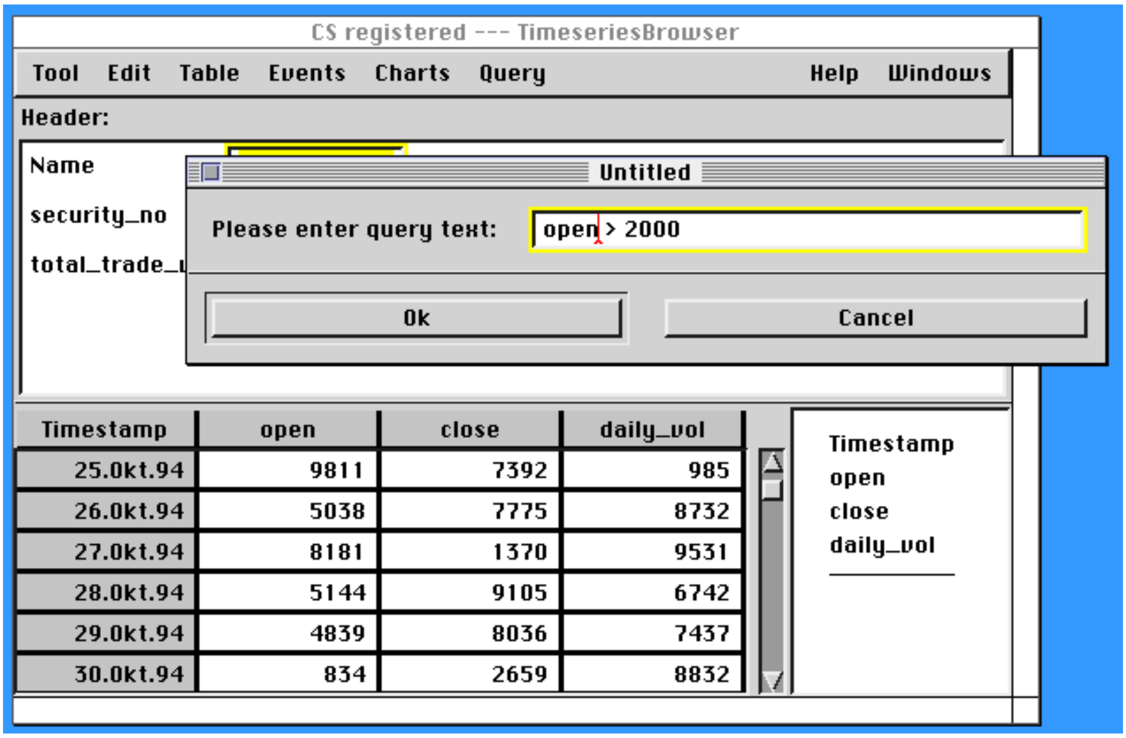

Figure 9: Simple query on time series *CS registered*

4 Conclusion

As this paper has shown and the system demonstration will further illustrate, we have built a TSMS with a graphical user interface as the primary means of access. This system is targeted directly at end users with no or very little experience in using a database, but with profound knowledge in time series analysis. Our early experiences with economic researchers have shown that this system really meets their needs as to easy handling and straightforward but powerful manipulation of time series. Future work will be done to enhance this special interface but also to interface existing software packages like spreadsheet or statistical programs to the TSMS kernel.

Acknowledgment

We thank Prof. Robert Marti from the Institute for Information Systems (ETH Zurich) for his helpful comments.

References

[DKS 94] W. Dreyer, A. Kotz Dittrich, D. Schmidt: An Object-Oriented Data Model for a Time Series Management System. Proceedings of the 7th International Working Conference on Scientific and Statistical Database Management (SSDBM'94), Charlottesville, Virginina, Sep. 1994.

[FAME 90] FAME Software Corporation: User’s Guide to Fame, 1990.

[SC 93] A. Segev, R. Chandra: A Data Model for Time-Series Analysis. Workshop on Current Issues in Databases and Applications, Rutgers Univ., Oct 1992. In: Advanced Database Systems, editors: N. Adam and B. Bhargava, Lectures Notes in Computer Science Series, Springer Verlag, 1993.

[SS 93] A. Segev, A. Shoshani: A Temporal Data Model Based on Time Sequences. In [15], chapter 11, pp. 248 - 269.

[WGM 89] A. Weinand, E. Gamma, R. Marty: ET++ – An Object-Oriented Application Framework in C++. Structured Programming, Vol. 10, No. 2, June 1989.