Requirements and Design of Replication Services for a Time Series Management System

Werner Dreyer, Duri Schmidt, Angelika Kotz Dittrich, Manuel Bleichenbacher
UBILAB, Union Bank of Switzerland, Zurich
{dreyer, schmidt, dittrich, bleichenbacher}@ubilab.ubs.ch

Abstract

This paper analyzes how financial researchers manage large numbers of time series and how they work with these data. We show that replication services are a central facility of a time series management system and we define the requirements for such replication services. An evaluation of current time series management systems shows that they do not support replication. Neither replication systems of other database management systems nor other kinds of currently available replication systems cover our requirements. Thus, we present our design of replication services that are adapted to the needs of time series management. It is based on a publish-and-subscribe mechanism, a sophisticated scheduling facility, dynamic replication, and integrated directory services.

1 Introduction

Time series are of growing importance to financial researchers like economic analysts, portfolio managers or investment researchers. These researchers employ quantitative and statistical methods based on time series for different purposes like economic forecasts, defining portfolios with a certain return/risk-ratio and proposing shares to invest in.

There are various design alternatives for a time series management system (TSMS). In this paper we discuss issues regarding centralization versus decentralization of data, the degree of database management system (DBMS) usage, and the necessity of replication of time series. We propose an architecture that stores all time series in databases of a TSMS. In our case, this is the CALANDA TSMS [1] [2]. The databases are connected by a replication system.

The examples, experiences and problems described in this paper are taken from financial time series management at Union Bank of Switzerland (UBS), although our investigations and discussions with other institutions revealed that their situation is very similar to ours.

In financial institutions a typical setup is as follows (see figure 1): first, a centralized department gets the data from external sources like ticker services and online databases and stores them in a centralized time series base (a). This department also takes care of tasks such as data quality management or the replacement of missing values. Second, users access this time series base to get the data they are interested in (b). Furthermore, they also receive time series from external sources (c). These sources can be the same as the sources of the centralized database, or they are additional ones. The local data of the users are stored in files. Third, the researchers further process the time series (d). For example, they filter out trends to make forecasts. For these purposes, they use applications like statistics software or spreadsheets. Fourth, the resulting data are again stored in files (e). This description does not necessarily mean that all departments in the institution get the time series from one centralized database. In large organizations, several such systems are often operated simultaneously.

The storage of project data in files instead of databases results in considerable drawbacks, such as missing concurrency control and recovery, or the difficult manual management of a large number of files which may even be in different formats. Additionally, project data could also be of interest to other researchers. However, data that are stored in local files without global system control are hardly reusable by someone else.

A first approach to avoiding the storage of project data in files would consist of keeping all data in the centralized database. To bring the data to the client applications, researchers would directly access this database without locally storing the data. After being processed by the applications, the resulting time series would be reentered into the centralized database. As we will show in section two, an approach which uses a system of time series bases that are connected by replication better suits our application domain.

In this paper, the terms “replication services” and “replication system” have the following meanings: replication services encompass the functionality dealing with the duplication of data between different databases. They do not only consist of some basic functionality that actually copies the data, but also of additional capabilities like the management of relations between replicates or the facilities to define replication procedures with their specific characteristics. These replication services are offered by a replication system.

This paper appeared in the Proceedings of the Eighth International Conference on Scientific and Statistical Database Management, Stockholm, Sweden, June 1996, pp. 208-215.

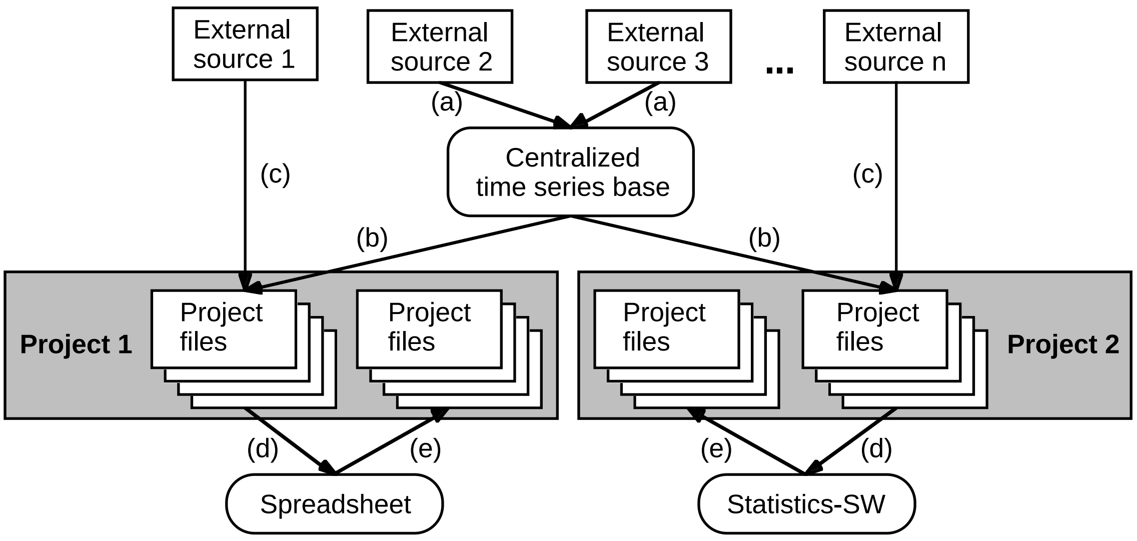

Figure 1. A typical setup for time series analysis.

The structure of the paper is as follows: section 2 explains why TSMSs profit from replication. Section 3 lists the requirements regarding replication that are specific for TSMSs. In section 4, we describe other replication systems. The design of a TSMS with integrated replication services is illustrated in section 5. Section 6 concludes the paper.

2 Reasons for using replication

There are several reasons why a TSMS should not rely on a centralized database but on a system of partially replicated databases. First, financial researchers often have to be completely autonomous regarding the time series they process. Second, the systems become more flexible. Lastly, replication improves the availability of the data and the system performance.

- *Autonomy, local control over the data.* The data that researchers get from external or internal sources often have to be modified to be ready for further processing in client applications. Typical modifications are periodicity transformations, replacing missing values, and data adjustments. Periodicity transformations are necessary to compare time series with different frequencies, such as a time series with daily and another with weekly events. Time series frequently have missing events. To be usable in statistical procedures, these missing values have to be replaced by interpolation values. Data adjustments have various reasons like stock price adjustments after dividend payments, error correction, smoothing of seasonal divergences, or trend filtering. These modifications introduce artefacts into the data. To be able to know whether they observe facts or artefacts in the results of their empirical investigations, financial researchers often prefer to do these modifications autonomously on their own copy of the data. Furthermore, there are various methods for performing these adjustments and replacements of missing values. If a central department within the company maintains all time series, it has some standardized methods to process the data. However, depending on the intended usage of the data, some methods may be more effective than others, or researchers may want to experiment with new methods and therefore cannot work with shared data.

- *Flexibility, speed of adjustments.* Because the researchers use a variety of methods for processing the initial time series, they also produce many new time series or even time series types. These dynamics are difficult to manage in a centrally maintained database. Allowing the researchers to keep their local instances of the time series, even if they deal with the same basic data, makes the entire system much more flexible and more easily adjustable.

- *Insufficient availability and performance.* If time series data are stored on a central host, there may be a considerable performance degradation, depending on the actual usage of the host for other applications. Furthermore, when the central host is down, all time series are unavailable. These problems are intensified by the fact that in a large institution the researchers are geographically dispersed, potentially over the whole world. Allowing the users to store the necessary data locally eliminates these problems: the availability of the data is much enhanced if the users can access them locally. Performance improves too, because there is less access concurrency, because the local databases are smaller than a centralized one, and because they can be tuned to the specific access patterns of the local users. Of course, it is not sensible to replicate all time series for all project groups. This could result in unnecessary overhead for management, communication and storage requirements. Therefore, one should consider carefully what data can be centrally stored and what data are better replicated.

3 Required characteristics of a replication system for a time series management environment

By examining the working style of financial researchers, we can identify some of the requirements of replication for that application domain.

- *Users want control over replication.* As shown in the previous section, researchers explicitly want individual copies of certain time series for specific processing. For this reason, there cannot be autonomous replication of time series by the system; the entire control concerning what time series are to be replicated must be in the hands of the users.

- *Users need to select replication schedules.* Different researchers want their replicates to be updated at different points in time. For example, a portfolio manager may need hourly updates of some stock prices, while an economic analyst is satisfied with daily or even weekly updates of the same time series. Besides, some researchers want to specify additional conditions that restrict the update of some replicates. For instance, an investment researcher may only examine new data about a specific share when its price goes beyond a certain limit.

- *Replication must be unidirectional.* The individual replicates must be independent of each other: researchers must be allowed to modify replicated time series any way they want. Thus, replicates cannot be read-only as in some other replication systems. There are no mutual updates between replicates, either. However, a time series can well be a replication target and a source at the same time. In this case the replication topology corresponds to a tree. The absence of cycles eliminates all problems regarding mutual consistency.

- *The set of replicated time series can change.* When setting up a time series management environment, it cannot be determined what data are to be replicated. For example, a company-wide time series base may increase the number of stored time series by accessing another data provider. An investment researcher may then decide to broaden the individual set of time series to investigate, which means that the system must allow replication procedures to be defined or canceled at any time.

- *Users need support in finding the relevant time series.* A typical time series management environment consists of numerous time series spread over many data stores. This makes it difficult for users to find the relevant time series. The replication of time series even aggravates the problem, because the number of available time series is further increased.

- *The replication system must be able to replicate the schema.* When a user replicates a time series for the first time, its schema may not be in the target database as yet. In that case, it must replicate not only the data but also the schema.

- *The replication system must deal with schema changes.* After a time series is replicated for the first time, the schema of either the source or the target time series may be changed. The replication system must detect this and handle these changes.

- *Replication is asynchronous.* Temporary inconsistencies between two time series can well be allowed for the intended applications. Most of the time series are updated with daily or even lower frequencies. Even time series bases that store tick-by-tick data have no problems tolerating certain delays. Synchronous replication where the changes of the source and of the target time series are contained in one transaction would result in considerable overhead without obvious advantages.

4 State of the art

4.1 Current TSMSs

Current setups, as we described them in the introductory section, do not at all support replication in a satisfactory way. Consequently, researchers perform replication manually by copying time series from the centralized time series base into files on their local data store. Nor do commercially available TSMSs, like FAME [3] or Illustra Time Series DataBlade [4], have any replication facilities.

In contrast to TSMSs, other DBMSs offer replication systems, and one can also find some middleware products. To see whether these could be used successfully in conjunction with a TSMS, we are going to examine them in the remainder of this section.

4.2 Replication features of other DBMSs

There exists a set of replication systems designed to be used with commercial DBMSs. They come closest to our requirements. Most of them either originate from vendors of relational DBMSs, or they are middleware products targeted at relational DBMSs. Well-known systems originate from Sybase [5] [6] [7], Oracle [5] [7] [8], IBM [5] [7] [9], Ingres [5] [10], Microsoft [11], Trinzic Corp. [5] [12], Praxis, Inc. [13], and Lotus [5] [14] [15] [16] [17] [18]. Of course, these systems have different strengths and weaknesses. In the following, we give only a short survey of the main weaknesses. However, we wish to clearly state that when we say “weakness”, we mean this from our own point of view. Some design decisions of those systems turn out to be a drawback for our TSMS environment, but they are well justified for their intended application domains.

- *Lack of flexibility.* These systems usually define statically which databases participate in a replication, and which data are to be replicated. Furthermore, they employ a static replication setup, that is, the schema of the replicated data cannot be changed while the system is running. Last, the scheduling facilities are rather limited. One obvious reason for this is that most of these systems are used with relational DBMSs which also have many inherent limitations. For example, the database schema cannot be defined on the fly. Additionally, these replication systems are typically designed for applications whose characteristics are very different from applications that process time series. Examples are transaction processing applications where sales and stock data from branch stores are replicated to the headquarters, or decision support applications where data warehouses get some predefined set of data from the transaction processing system. In these applications, it can be decided in advance which data are to be replicated, the schema remains unchanged, and the scheduling is simple, too (for example, the transaction processing environment replicates the sales and stock data as soon as possible, the decision support system replicates the data daily).

- *No directory services.* The existing replication systems do not provide any directory service functionality. This is related to the application characteristics described before: when it is known in advance what data are to be replicated into which database, replication does not bring up additional data retrieval problems.

- *Confinement to the relational data model.* Most of these replication systems are designed to be used with relational database systems (gateways to some other database systems provide only limited functionality). There is no straightforward way to use these replication systems for other database systems that are not based on the relational model.

4.3 Further replication systems

Besides the mentioned products, there are systems like ISIS [19], Arjuna [20] [21], OSCAR [22] [23], the SOM replication framework [24] [25], and BOAR [26] which also offer replication as a central or subsidiary capability. All these systems aim at applications whose replication requirements differ considerably from time series management. They use replication mainly for performance and availability improvements. Ideally, replication in such systems is transparent to users. Updates between replicates are bidirectional. There is no flexible scheduling, changes to one replica are usually transmitted to its peers as soon as possible. Access may be limited while the replicas are in an inconsistent state. The schema is static, there are no changes while the system is running. None of these systems provides a directory server.

ISIS supports the development of distributed applications by offering groups of replicated processes with mechanisms like reliable multicast and event ordering. Arjuna also aims at building distributed applications, but it focuses on transactions, that is, the objects to be manipulated are usually long-living. These objects are potentially replicated, too. OSCAR provides weak-consistency replication between databases in an internetwork where communication is unreliable or where the network is partitioned. The SOM replication framework is mainly targeted at applications in domains like computer-supported cooperative work. Here, all replicates usually have the same structure as the original data, and updates to one copy are transmitted immediately to all other instances. BOAR is a library of replicated objects. It provides replicated objects based on lower-level fragmented objects. By appropriately combining objects of this library, developers can realize replicated objects with the desired degree of fault tolerance, availability, etc.

As the description of all these replication systems shows, it appears that none of these systems covers our requirements. They aim at application domains whose characteristics are substantially different from applications that process time series.

5 Design of a TSMS with integrated replication services

To overcome the weaknesses of current solutions, we now propose a design for a TSMS with integrated replication services, as shown in figure 2. This approach fully relies on databases, no data are stored in files. Moreover, there is not one centralized database, but a number of public and project-specific databases which are connected through replication. Directory services support users in finding the relevant data. We will elaborate this design in the remainder of the section.

5.1 Time series bases, data model

In this architecture, all time series are stored in databases. Time series that result from further processing in client applications will also be stored in a time series base. Compared to a file-based solution, this approach bears all the advantages that databases offer, like consistency maintenance or query facilities. Furthermore, the client applications are less sensitive to modifications of the time series schema than they are when they access files. The query facility of the DBMS serves as an isolating layer between the data and the applications. If, for example, an attribute is added to a time series, an old database query that explicitly enumerates the resulting attributes (like "select attr1, attr2 from ...") does not have to be modified. In contrast, if researchers directly access a file, they have to know its structure.

One could argue that the introduction of a database raises the number of necessary data format transformations. There is not only one transformation between the file format of the data provider and the format of the client application, but an additional conversion into the database import format as well. However, the introduction of an intermediary database often reduces the necessary transformations. If one uses m data providers and n client applications, there are m·n transformations at worst. When several client applications are used in a row, the worst case increases to m·n·(n-1) transformations. With an intermediary format, there are at maximum 2·m·n transformations.

There is a potentially high number of autonomous databases. Several DBMS instances may be running, each of which possibly manages several databases. These databases are connected by replication. All time series bases use the same kind of database management system, and therefore also the same data model. In our case, this is the CALANDA TSMS with its special-purpose time series data model. However, the databases differ in the database schema. Uniformity of the database management system has several advantages: first, there is no semantic gap between data models; second, researchers only have to be familiar with the characteristics of one TSMS.

Some of these databases serve mainly to import the data of the publicly available, external data providers. These data are of interest to a large number of financial researchers and to various projects. In figure 2, these would be the "OECD" and the "SNB" databases. According to our experience, the data of each provider are either kept in a single database or they are split into several such time series bases. These databases shield the users from the specifics of the data providers. One or several specialists are responsible for a time series base. They are not only experts in the usage of the TSMS, but they also have detailed knowledge about the exact content of their databases. This is very important, as most of the providers offer very limited or no search facilities at all, and there are hardly any on-line descriptions of the available data. These experts increase the semantics of the time series that are stored in such a time series base. For example, they bring the

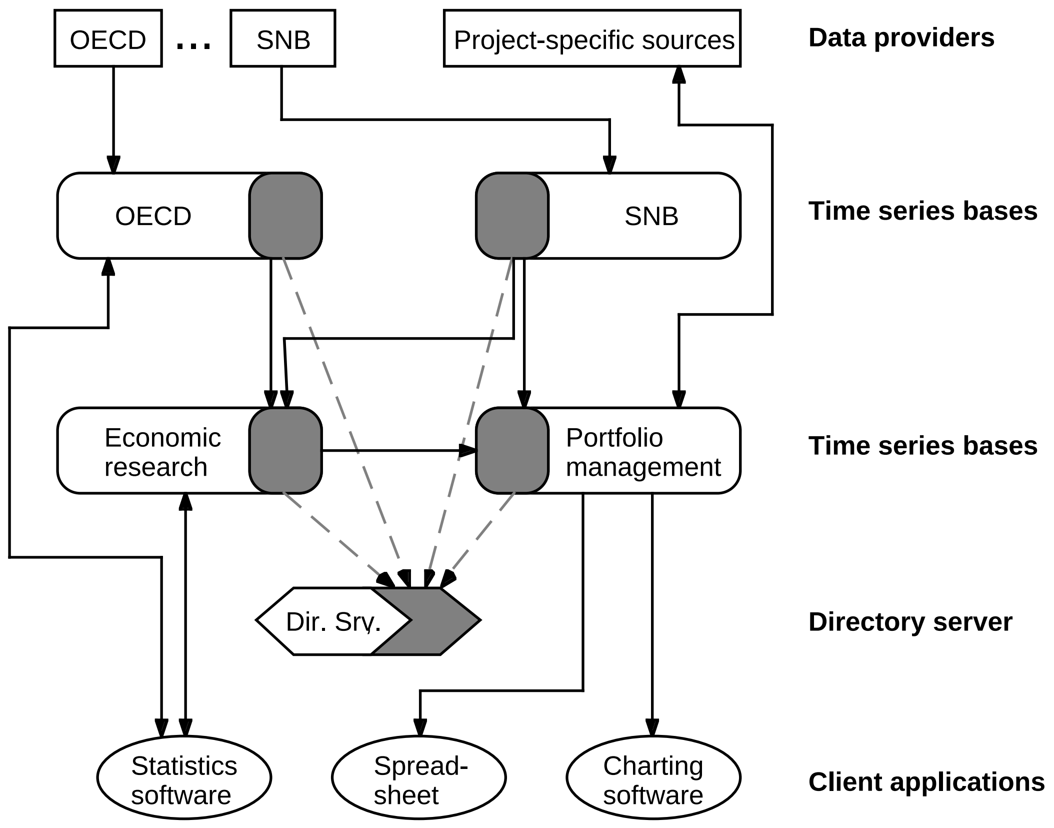

Figure 2. A time series management environment with integrated replication services. The dark gray boxes represent the components that offer replication services.

time series into an appropriate group structure [2], or they add additional attributes, like a comment about the data quality or the name of the provider.

Other time series bases contain more project-specific data. In figure 2, these are the “Economic research” and the “Portfolio management” databases. As described in section two, researchers often first have to adjust the data to analyze them in the desired way. These modifications are performed locally, because they are only relevant to a project. The time series in such a database stem from different sources: one part is replicated from time series bases that import data from external providers. Another part is replicated from further project-specific time series bases. The last part originates from project-specific data sources because these data are not of interest to a wide range of financial researchers and therefore are directly entered in a project time series base.

5.2 Publish-and-subscribe mechanism

Replication of time series is realized by a publish-and-subscribe mechanism. If some time series are thought to be of interest to other researchers, the owner makes them available via publication. Published time series are visible from outside the time series base. Users can then browse through the published time series. When they want a particular time series to be replicated into their own time series base, they subscribe to it and choose the desired replication scheduling. The replication system then automatically sets up the replication procedures.

This mechanism best supports the necessary degree of replication flexibility. Publishers make their time series available without having to know who will replicate the data later on. They can alter the set of published time series at any time. Subscribers can always subscribe or unsubscribe to some time series. Of course, the publication of data that originate from external providers has to accord with any contracts that are made with these providers.

Usual publish-and-subscribe mechanisms allow anyone to subscribe to a published item. However, this is not always desirable in our environment, and one prefers to restrict foreign access to certain time series. For example, an economic researcher experiments with a new forecast method and wishes to make the forecasted time series available to other researchers of the group for comments on the methods and the results. However, the data must not be available for further users as long as the method is not approved. The system must therefore offer the possibility to limit the replication of data to a restricted set of users.

A well-known application that also uses publish-and-subscribe is Usenet [27]. Usenet has some characteristics that are similar to our environment: there are numerous news groups, the set of news groups varies over time, and a person who posts an article cannot know who might be interested in the article.

5.3 Scheduling

When researchers subscribe to a time series, they also define the appropriate scheduling. As different time series use various update schedules, the replication system must offer flexible scheduling facilities. The definition of a schedule can be compared with event-condition-action rules found in active database systems (ADBS) [28].

Events are either temporal or non-temporal. The most obvious events are temporal ones, that is, one specifies the point(s) in time at which a replication is initiated. Temporal events are either defined by one or several explicit time points, or by a calculation rule. Examples are “tomorrow at 4:00 p.m.”, “daily at 9:00 a.m.” or “every first working day of the month at 8:00 a.m.”. We assume that the majority of events in our environment are temporal.

Besides temporal events, there are also non-temporal ones, like changes of some values in a time series. The replication system must be able to detect non-temporal events either at the source or at the target database, i.e., one can specify pushed or pulled replication.

Conditions have the same meaning as in ADBSs, an event only leads to a replication if the condition is true. The action is always replication. The complexity of event-condition-action rules for a TSMS is an open issue. It remains to be seen how complex replication events and conditions have to be in practice - whether, for example, composite events are needed or not.

5.4 Dynamic replication

Dynamic replication deals with divergences of the schema between sources and targets of replication. As mentioned in section three, two problems may arise: first, the target database does not yet have a schema of the time series that is to be replicated for the first time. Second, the replication is already set up, but the schema of the source time series has changed, that is, schema evolution takes place.

The first case is easy to handle. The system automatically replicates the time series schema before the time series itself is replicated for the first time.

In the second case, a semi-automatic mechanism is employed. When one changes the schema of a time series that is a source of replication, this time series is no longer replicated, and the owners of the target time series are notified by the system. They can then choose whether they want their local schema to be adjusted automatically or not. It would not be sensible to automatically adapt the local schema without notifying the user. Although the query facility of the database offers some isolation between the schema of the database and the client applications, some of the interfaces to the client applications or the client applications themselves possibly need to be adjusted. For example, it could be that spreadsheets have to be enhanced by a new event attribute. After being notified, researchers will usually choose automatic schema update and will perform the necessary adaptations of the client applications. As soon as the schema is updated, replication starts again.

A static approach would not support users in handling schema changes. This is not sensible for an environment with numerous schema changes. Additionally, because many databases may have replicates of the same time series, schema changes in a static replication environment cause a great deal of duplicated work.

5.5 Integrated directory services

To ease the retrieval of the relevant time series out of many time series bases, we incorporate a directory that stores metadata about all the published time series. The connection between a time series base and the directory server is also realized by the replication system. When a time series is published, a replication task for the metadata is defined, and the metadata are automatically sent to the directory server. When the metadata of a time series change, the directory is updated accordingly.

We distinguish between two kinds of metadata, namely schema data and application-specific metadata. Schema data originate from the system catalog, like the names and types of the attributes or the periodicity of the events. These metadata are implicitly defined as such. Application-specific metadata are explicitly defined by a user. The owner of a time series can declare arbitrary header attributes [2] as metadata. Usually, these will be attributes that best distinguish between different time series.

The directory server keeps track of the origin of the metadata that it stores. This makes subscriptions via the directory server possible. One can issue a query over the metadata and the directory server presents all time series that match this query. Users can now subscribe to the resulting time series in the same way as they do when they are directly browsing through a time series base, because the directory server sets up the appropriate replication procedure.

The directory itself, of course, can also be replicated at various sites. Mutual updates of the directories are realized by the same infrastructure that replicates time series.

6 Conclusion

We analyzed the way researchers manage and work with time series. Financial researchers have special needs regarding replication. An appropriate replication facility is a key factor for a satisfactory solution. Current TSMSs offer no replication functionality at all, while the replication facilities of general-purpose DBMSs do not cover our requirements. In contrast to those approaches, our replication services are designed to meet these needs.

We claim that a system based on our design helps financial researchers in concentrating on their primary task, namely, economic analysis. Besides, we expect that further application areas where other statistical data are processed have similar requirements concerning replication. Thus, the DBMSs that are used for these applications would also benefit from our work.

References

[1] W. Dreyer, A. Kotz Dittrich, D. Schmidt: Research Perspectives for Time Series Management Systems. SIGMOD RECORD, Vol. 23, No. 1, March 1994.

[2] 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, Charlottesville, Virginia USA, Sept. 28-30, 1994.

[3] FAME Software Corporation: User's Guide to FAME, 1990.

[4] Illustra Information Technologies, Inc.: Illustra TimeSeries DataBlade, technical information 1994.

[5] H. Edelstein: The Challenge of Replication. Part 1 in DBMS, Mar. 95, pp. 46-52, part 2 in DBMS, Apr. 95, pp. 62-103.

[6] A. Moissis: Sybase Replication Server: A Practical Architecture for Distributing and Sharing Corporate Information. Sybase White Paper, http://www.sybase.com /Products/Whitepapers/repserver_wpaper.html, 1995.

[7] D. Stacey: Replication: DB2, Oracle, or Sybase? Database Programming and Design, Dec. 1994.

[8] Oracle: Oracle7 Asynchronous Distributed Capability Overview. Oracle White Paper, http://www.oracle.com /info/products/symrep/chapter4.html, 1995.

[9] IBM Corp.: Data Replication: The IBM Solution. IBM White Paper, May 1994.

[10] Computer Associates, Inc.: CA-OpenIngres/Replicator. Product Description, 1994.

[11] Microsoft, Inc.: SQL Server Data Replication. http://www.microsoft.com/SQL/sqlrevg4.htm, 1995.

12] Trinzic Corp.: InfoPump: Client/Server Middleware for Routing, Integrating and Synchronizing Dissimilar Data. Trinzic Product Description, 1993.

[13] Praxis International, Inc.: OmniReplicator: Concepts and Facilities. Praxis International, Inc., 1994.

[14] Notes and Database Management Systems: Complementary Application Types. Lotus White Paper, http://www.lotus.com/corpcomm/26c6.html, Aug. 1994.

[15] Lotus Notes: An Overview. Lotus White Paper, http://www.lotus.com /corpcomm/2952.html, Feb. 1995.

[16] J. Mackenzie: Document Repositories. Byte, Apr. 1995.

[17] D. Yavin: Optimizing Notes Replication. Byte, Sep. 1994.

[18] D. Yavin: Replication's Fast Track. Byte, Aug. 1995.

[19] K. Birman, R. Cooper: The ISIS Project: Real Experience with a Fault Tolerant Programming System. ACM Operating Systems Review, Apr. 1991.

[20] G. Parrington, S. Shrivastava, S.Wheater, M. Little: The Design and Implementation of Arjuna. USENIX Computing Systems Journal, Vol. 8, No. 3, 1995.

[21] M. Little, S. Shrivastava: Object Replication in Arjuna. Technical Report, University of Newcastle upon Tyne, UK, Aug. 1993.

[22] A. Downing, I. Greenberg, J. Peha: OSCAR: A System for Weak-Consistency Replication. Proceedings of the Workshop on Management of Replicated Data, Houston, Texas, Nov. 1990.

[23] A. Downing, I. Greenberg, J. Peha: OSCAR: An Architecture for Weak-Consistency Replication. Proceedings of the International Conference on Databases, Parallel Architectures, and their Applications (Parbase-90). Miami Beach, Florida, Mar. 1990.

[24] IBM Corp.: SOMobjects Developer - Toolkit Users Guide. Version 2.0, June 1993. On Apple Developer CD Series, Nov. 1994.

[25] IBM Corp.: SOMobjects Developer - Programmers Reference Manual. Version 2.0, June 1993. On Apple Developer CD Series, Nov. 1994.

[26] G. Brun-Cottan, M. Makpangou: Adaptable Replicated Objects in Distributed Environments. INRIA Technical Report RR-2593, May 1995.

[27] Network Working Group: Network News Transfer Protocol - A Proposed Standard for the Stream-Based Transmission of News. Request for Comments (RFC) 977, Feb. 1986.