Database Issues in Telecommunications Network Management

Ilsoo Ahn
AT&T Bell Laboratories
Columbus, Ohio 43213, U.S.A.

Abstract

Various types of computer systems are used behind the scenes in many parts of the telecommunications network to ensure its efficient and trouble-free operation. These systems are large, complex, and expensive real-time computer systems that are mission critical, and contains a database engine as a critical component. These systems share some of common database issues with conventional applications, but they also exhibit rather unique characteristics that present challenging database issues. Major DBMS issues for network management include choosing the right data model, handling two different kinds of data in terms of integrity and recovery constraints, supporting temporal queries, satisfying real-time performance and high availability requirements, and several miscellaneous issues. Some of these issues have been investigated in various areas of database researches, but most of them largely remain in the research stage. Advances in these areas that result in actual integrated implementations for data-intensive, real-time and temporal applications are eagerly awaited.

1. Introduction

The modern telecommunications network offers many types of services critical to various personal and business activities. They range from simple phone calls, whether local, long distance or international, to advanced services such as toll-free services, private virtual networks, calling card verifications, and multi-way conference calls. It carries not only voice but also data files, picture images and video signals. The share of multi-media traffic will continue to increase as the distinction among computers, telephones, wireless communications, and even televisions gets ever more blurred.

SIGMOD 94- 5/94 Minneapolis, Minnesota, USA
© 1994 ACM 0-89791-639-5/94/0005..$3.50

The telecommunications network is made up of several types of special purpose computers, usually called *switches* or *exchanges*, interconnected through a complex web of transmission links. The more sophisticated the services offered by the network grow, the more complicated the structure and the facilities of the network to support those services become. Any failure of the network to deliver expected levels of services may result in serious consequences to the customers with costly losses not only in financial but also in human terms. Therefore various types of computer systems are used behind the scenes in many parts of the network to ensure its efficient and trouble-free operation.

Most of these systems need to deal with a large volume of data, either entered by humans or generated by the machines, so some form of database management systems (DBMSs) is usually an essential component of these systems. It is interesting to note that the DBMS technology plays an important role in the telecommunications network, though the two fields may appear to be rather disjoint.

These systems share some of common concerns in terms of database issues with more conventional applications, but they also exhibit rather unique characteristics that are not handled well by existing commercial DBMSs. In this paper, we will describe the characteristics of the network management systems for telecommunications, and discuss the database issues and challenges posed by such systems. We believe that many of these issues are also applicable to other kinds of applications that are data-intensive with real-time and temporal requirements such as computer aided manufacturing, process control systems, transportation management systems, and stock market control systems [Ramamritham 92].

2. Network Management Systems

In principle, the telecommunications network is designed to provide an optimal level of service based on the traffic load of an average business day. But the traffic load can change greatly due to various reasons. For example, there are hourly, daily, seasonal fluctuations, and unpredictable variations depending on business conditions or even

weather. Besides, residential traffic shows a quite different pattern from business traffic. It is usually acceptable since the volume of residential traffic is lower, but presents serious problems on special holidays like like Christmas and Mother's day. Then it is necessary to reconfigure the network dynamically depending on the traffic load across the geographic regions. Problems also occur unexpectedly due to natural disasters like earthquakes or cyclones, mass calling for special events, and facility failures like fiber cut or fire at an exchange site.

When these abnormal situations which the network is not engineered for occur, the network management system must detect such a situation as early as possible, identify its cause, and apply appropriate controls to the network to minimize its effects and prevent the problem from spreading to other parts of the network. Hence the network management system is indispensable for supporting the modern telecommunications network and utilizing the resources efficiently. It also provides traffic data to forecast and plan for the future, can be used as effective tools for public relations, and offers better quality of services to customers for eventual increase in revenues. Some types of systems are also necessary to meet regulatory requirements, while others are pre-requisites for offering certain services, *e.g.* to provision and maintain configurations of the network facilities or customer dependent information.

There are many types of computer systems, literally hundreds or even thousands, being used behind the scenes to support the operation and management of the telecommunications network. Depending on the functions they perform, they are called *operations support systems*, *service support systems*, *service management systems*, *element management systems*, *network management systems*, *network traffic management systems*, *etc.* We refer to these systems as the network management systems (NMS) in a generic sense in this paper.

As the interests in network management grow bigger and the telecommunications network becomes more heterogeneous involving various types of equipments from multiple vendors, the international standards to manage all the network elements in an integrated manner becomes necessary. So ITU-TSS (formally CCITT) has recommended a set of international standards for the Telecommunications Management Network (TMN) [Klever 88, CCITT 92a] based on the model of Open Systems Interconnection (OSI) [ISO 84]. Though some of its details still need to be finalized and actual implementations are not available yet, it is gaining large followings among the telecommunications companies as well as various computer vendors [OMNIPoint 93].

The TMN recommendation defines the following application functions:

- Performance Management: to monitor the status of network resources, traffic load, equipment utilization, and identify exceptional conditions.
- Fault Management: to detect alarms, diagnose problems and apply controls.
- Configuration Management: to provision new equipments or services, audit and reconfigure network resources.
- Accounting Management: to provide billing data, resource usage reports, and cost calculations.
- Network Planning Management: to prepare for capacity growth, contingency and strategic planning.
- Security Management: to handle authorization and authentication.

To perform these functions, NMSs in general collect data from the network elements, and store them into a database. They analyze the data to find any exceptions against some threshold levels, detect any abnormal patterns, and present the results to human operators through graphical and/or tabular displays. Hence these systems usually consist of three major subsystems:

- Data Communications Subsystem : to receive data and send controls to network elements such as exchanges, signaling transfer points, service control points, *etc.*
- Management Information Base (MIB) Subsystem : to store and process information for network management in a database.
- User Interface Subsystem : to interact with network managers preferably through graphical user interfaces.

These systems are large, complex, and expensive real-time computer systems that are mission critical, and need to be fault tolerant or highly available. Building these systems requires state-of-the-art computing technologies and multi-disciplinary expertise in such areas as computers, communications, database management, software engineering, object-oriented technologies, and computer graphics. In this paper, we concentrate on the database issues and challenges presented by these systems.

3. Database Issues & Challenges

In the heart of the MIB subsystem is a database engine that needs to meet the specific requirements of the NMSs. These systems share common database issues with conventional applications to some extent, *e.g* concurrency, storage structures, query processing, performance optimization, recovery, *etc.* But they also exhibit rather unique characteristics as discussed in this section. So many NMSs used to rely on special purpose database engines, either main-memory based or file based, built in house to meet specific requirements. But this practice is

expensive in terms of development and maintenance costs, and recent advances in the hardware and the DBMS technologies make it more attractive to employ the off-the-shelf DBMSs commercially available. Yet commercial DBMSs available nowadays usually address the needs of decision support (DS) or on-line transaction processing (OLTP) applications, and are not adequate to handle some of these requirements properly.

Major DBMS issues for the NMSs include choosing the right data model for information models of network management, handling two different kinds of data in terms of integrity and recovery constraints, supporting temporal queries, satisfying real-time performance and high availability requirements, and several miscellaneous issues. Some of these issues have been investigated in various areas of database researches, *e.g.* real-time databases [Ramamritham 92, Ulusoy 92], temporal databases [Snodgrass 90, Soo 91], distributed and parallel databases, query processing and optimization, *etc.* But most of them largely remain in the research stage, and no DBMS implementation is expected to be available in a foreseeable future as an integrated solution for actual applications that deal with a large volume of temporal data under real-time performance constraints.

3.1 The Right Data Model

The NMSs deal with many types of objects, both physical and conceptual. The first task in developing a database application is to model the information domain and map the result to a data model. For modeling the objects of network management, the international standards from ITU-TSS and ISO are based on the object-oriented paradigm [CCITT 92b, NMF 89, Klever 93]. When information modeling is done using the principles of encapsulation and inheritance to capture the complex relationships among the managed objects, the result tends to have a large number of objects with many subtypes in deep hierarchies.

An object-oriented DBMS would be considered a good match to implement a database for the information model with such a complexity. Despite the enormous amount of attention received in recent years from the research community and the trade journals, object-oriented DBMSs are not used as widely, at least yet. A reason may be that they are still new and evolving, or that people still need time to learn about those. A bigger reason would be the fact that there is no standard query language or application programming interface, and all the products available in the market are incompatible with one another without a clear winner. So users are reluctant to make a pick and get stuck with a wrong choice. Recognizing this problem, object-oriented DBMS vendors started on a standardization effort [ODMG 93], but it will be a while before details are worked out and actual implementations show up to get accepted widely for practical applications. It is also uncertain that object-oriented DBMSs can deliver the kind of performance in handling the large amount of network measurement data as claimed by some vendors for CAD-like applications.

In contrast, relational DBMSs are dominant in the conventional applications such as DS and OLTP systems. Most relational DBMSs support the Standard Query Language [ANSI 86, ANSI 89], and are compatible with one another to a certain extent. A variety of off-the-shelf tools are available for relational DBMSs from independent vendors to help data retrieval, report generation, and applications development. These advantages make relational DBMSs as the preferred candidate for the database engine of many NMSs.

But they lack the capability to model the inheritance hierarchy representing the complex relationships among the managed objects. It is a difficult task to map such a model onto a relational DBMS which does not support type hierarchies. A solution is to create one table for every leaf object of the model, but it results in many similar tables for the set of objects in a hierarchy. This causes performance problems in locating the proper tables when storing measurement data collected from the network elements under the constraint of real-time performance. Performance of relational DBMSs is usually the biggest concern in many NMSs, because performance overhead can easily overwhelm even the most powerful machines available today when dealing with a large amount of measurement data in some systems.

3.2 Two Kinds of Data

The data processed by the NMSs can be classified into two different kinds: *reference* data and *measurement* data. They exhibit different properties in many aspects such as the source of data, integrity constraints, recovery strategy, performance requirements, storage characteristics, and temporal queries. It is desirable for the DBMSs to handle each piece of of data differently depending on the kind.

3.2.1 Maintaining Reference Data

Reference data represent the configuration of the network resources. They are relatively static, yet need to be changed occasionally. They can also be modified proactively or on a scheduled basis, as new facilities are added to the network or the status of some resources changes. They are usually entered into the system through a manual process, but they may be supplied from other computer systems through some degree of automation. A user-friendly human interface is necessary to maintain such data accurately with a minimal effort.

It is important to maintain the integrity of the reference data. So the DBMS needs to provide automatic integrity checking on both referential and other constraints, either

declaratively or through triggers and stored procedures. It is desirable to have the traditional support for database transactions on this type of data, including the capabilities for transaction logging and data recovery in case of database or media failures.

We also need to validate them regularly to avoid or minimize the discrepancies from the actual configuration of the network resources. This is done by the *audit* process, which polls or receives the configuration data from the network elements on demand or on a periodic basis, and compares them with the reference data. If discrepancies are found, they should be corrected by manual or automatic procedures. And these activities should be performed without disrupting the normal operation of the system. Note that maintaining the reference data shares common concerns with conventional decision support applications.

3.2.2 Processing Traffic Data

Traffic data, on the other hand, are collected continuously from the network. They are stored into the database, and analyzed to identify network problems in *ad-hoc* as well as in pre-programmed manners. These can be grouped into three categories:

- Periodic data: collected periodically from the network elements. The period ranges from 30 seconds up to 15 minutes in most cases.
- On-demand data: supplied from the network elements when requested by the NMS. Data for the audit process mentioned above is an example.
- Alarm data: sent to the NMS by the network elements to indicate status changes or alarm conditions that deserve a human attention.

The volume of these data is very high in many cases even for a network of reasonable size, especially for data representing the status of communication links which tend to be $O(n^2)$ where n is the number of nodes in the network. Storing and processing these data in real-time is a very CPU intensive operation, presenting serious performance problems to the DBMS.

Integrity requirement of measurement data is not as high as the reference data. It is usually acceptable to lose some portion of measurement data, since similar data can be obtained again from the network elements later. This is quite different from the principle of perfect data integrity for the conventional databases dealing with financial transactions, for example. Therefore transaction support and data logging for redo or undo operations are not really necessary. They are even undesirable for measurement data from the viewpoint of performance overhead in data storage and longer delays in recovery.

3.3 Recovery

When the DBMS recovers from a failure, it needs to recover the reference data as accurately as possible to the latest state before the failure occurred using the information in the transaction logs. It, however, is more important to return to the normal operating mode quickly rather than spending a long time to recover the measurement data. In fact, it would be unacceptable to recover all the measurement data that have been collected since the last backup, because it can literally take hours or even days before the system can be brought up.

It is desirable to control the process of transaction logging on and off selectively depending on the type of data. Some commercial DBMSs offer the option of choosing the mode of logging for a database as a unit, but tables belonging to two different databases with different logging modes cannot be accessed together in a query. Such an option to control logging on a per-table basis is not available. It is necessary to have multiple levels of integrity and recovery strategies depending on the nature of data with varying degrees of associated costs in terms of performance and resource usages.

3.4 Temporal Queries

The reference and the measurement data exhibit different properties in terms of temporal characteristics. Since measurement data are collected continuously for the network resources, multiple instances of data will come in for the same resource. Instead of updating existing instances in place, multiple versions of measurement data should be maintained on-line by attaching time stamps to the data records or by storing the data for each period in a separate table. The measurement data are the snapshots of the network status, and can be treated as the *transaction event* relation [Snodgrass & Ahn 86]. It is useful to maintain the history of the reference data as well. The reference data can be modeled as the *valid interval* relations to represents the status of the network configuration valid for an interval.

It is necessary to compare the measurement data for multiple, at least three or four, periods to identify any pattern or trend. Data from a previous day or a previous week need to be compared in some cases to detect an abnormal situation. Thus these systems are required to maintain data for several days on line for real-time queries, and for ad-hoc analyses or report generations.

It is also required to look at historical data about some events that occurred in the past. For example, measurement data on a peak day such as Christmas or those on an earthquake need to be studied later to prepare plans for the future. Such data need to be copied into off-line storage devices, before they are deleted from the on-line disks, for a *play back* operation in the future. It is also

necessary to prune the database by deleting the old data and to make room for the new data, which can be an expensive operation depending on the database design.

These are good candidates for the temporal database applications, but the DBMSs currently available do not provide such capabilities [Snodgrass 90, Soo 91]. Therefore NMSs have to deal with temporal requirements in an ad-hoc manner. One scheme is to organize the data for each day into a separate database. To archive the data of a particular day, the entire database corresponding to the day is backed up to an archival tape. Then the tape is restored to the system when the historical data need to be reviewed. This brute-force approach does not work well because of mismatches that occur between the reference data and the measurement data. It is also difficult to access and compare data across the day boundaries.

An alternative is to simulate a limited form of temporal database on a conventional DBMS in a systematic manner [Ahn 93A]. In this approach, a single database contains data for multiple days eliminating the problem of crossing the day boundaries. Then the backup operation becomes more complicated. A complete backup of the entire database becomes unmanageable because it often requires the database to be shut down, let alone the problems of enormous size and long elapsed time. A partial backup of a particular day in a conventional manner is not useful for the play back operation, because we cannot expect to restore the partial backup and get a consistent view of the database in the past because of the eventual mismatches with the schema that are changing continually. A solution for this problem is to use the export and import features available in most DBMSs, but it tends to be rather slow.

3.5 Real-Time Performance

The NMSs operate under strict real-time constraints to perform various activities such as data collection, data storage, query processing, data analysis, and sending out controls to the network. If the DBMS cannot process the load in real-time, the NMSs may miss the time window of certain critical operations, and result in service degradation or lost revenues.

The biggest issue of the DBMSs, especially the relational ones, for the NMSs in many cases is the performance. The DBMS for such systems should be able to support a high throughput of update operations to store a large volume of data in real-time. The amount of data is much bigger than conventional applications because the data are generated by many machines making up the network. In addition, it needs to run many complex queries with heavy computations to analyze the measurement data, display the results, generate reports, and transfer data to other systems. In many cases, commercially available DBMSs could not meet the performance requirements satisfactorily. So the NMSs often had to rely on home grown DBMSs specialized to specific tasks, such as file based databases, main-memory databases, or hardware database machines.

Nowadays with better hardware and performance improvement in query processing, relational DBMSs are favorably considered as the database engine for the NMSs. As it turned out, the requirements of some NMSs is still stretching the limit of the performance attainable with the relational DBMS on some of the largest machines available these days. Therefore it is necessary to optimize the database configuration and the queries very carefully. A same query could easily cost 10 to 20 times more, demonstrating the importance of performance optimizations on the relational DBMS.

Test results show a large variance depending on the configuration and the parameters of the DBMSs in many cases. Installing and bringing up a DBMS using the default parameters may exhibit an unacceptably poor performance. It is necessary to tune the major factors to affect the performance, identify the bottlenecks, and optimize the configuration to the maximum. By changing various parameters and setting up proper configurations, performance improvement of over a factor of 10 could be observed in many cases.

It is important to estimate or characterize the rate and the complexity of queries, then to evaluate the performance and resource usages of such queries. Note that the performance is highly dependent on various parameters such as the size of the block, amount of the buffer space, data contents of the buffer, *etc.* Some queries can affect the system performance seriously by monopolizing the CPU and disrupt the data buffer, so care should be taken when real-time applications and decision support applications run in the same machine.

Storing data into a relational DBMS is an expensive operation, which usually employees a B-tree structure involving sorting and index building. In many applications, storing data through the SQL `INSERT` operation is not fast enough even using the vendor specific optimization features such as array or page insertions. An alternative would be to use the bulk load facility available in some DBMSs, but the method has several constraints which limit its usefulness. For example, a new table should be used for each period, or no index can be maintained on a table. When data for each period are stored into a separate table, retrieving data for multiple periods becomes a union or a join operation on multiple tables. Such idiosyncrasies should be made invisible to the users using the view mechanisms dynamically.

As the measurement data are stored, the NMSs need to perform *exception processing* to identify any unusual conditions in the network. It involves comparing the values of many fields in each measurement record against several types of threshold values to indicate unusual

conditions of low, medium, or high priority. Currently, this problem is well beyond the capacity of the SQL processing in the relational DBMSs, and can be handled only by programs written in procedural languages such as C or C++ with special data structures maintained in the main memory. Optimizing the performance is usually the biggest concern in designing applications in the critical path of real-time processing. Further improvement in the performance, by a factor of multiples, is eagerly awaited.

The DBMS also has to support various types of queries from display terminals, workstations, and report generators. The characteristics of the database queries are usually different between the interactive displays and the report generators. The former are mostly interested in the current data with a quick response time of a few seconds or less, similar to OLTP operations. The latter often involves historical data spanning over multiple periods or even days, and can take hours to complete, similar to DS applications. It is difficult to meet the real-time constraint when different types of operations run on the same machine, since existing DBMSs do not provide a good scheduling mechanism to control the priority of database requests.

3.6 High Availability

The NMSs usually operate without any down time, 24 hours a day and 7 days a week. Thus most systems are designed with a secondary machine for hot stand-by. In case the primary machine fails for either hardware or software problems, the secondary machine takes over the operation with a minimal delay. Some systems even have a tertiary machine, located in a remote site to avoid any disaster situation affecting both the primary and the secondary machines. Systems that cannot tolerate any down time of even a few minutes have to use the fault tolerant hardware with additional contingency strategies, though faults with the software rather than the hardware tend to cause more problems in practice.

In such an environment, the availability of the DBMS is critical for the whole system, and the DBMS should be able to recover quickly if it ever fails. It is essential to have better capabilities to backup and restore both the reference and the measurement data on-line, because the DBMS cannot be shut down for a long period of time to make backups. For a system with a hot stand-by, data replication capability is useful to copy a large portion of the data to the backup machine with a small performance overhead so that it can be used for CPU intensive operations such as ad-hoc analyses or generating reports. This arrangement would let the primary machine keep working on time critical operations without being affected by unexpected loads from non-critical queries.

3.7 Disk Space

The NMSs collect a large volume of data, and need to maintain them on line for up to 10 days or even longer, taking a large amount of disk space. Most relational DBMSs use the B-tree structure to store data, which usually have a space overhead of over 100%. Therefore it is desirable for the DBMS to minimize the storage space using some compression techniques so that the cost for disk space can be reduced. Note the fact that the measurement data do not change, once stored into the database. There may be ways to take advantage of this fact, *e.g.*, storing them on write-once media like optical disks.

3.8 Miscellaneous Issues

In addition, the DBMS should support the following items for the NMSs.

- The DBMS should be easy to operate, administer, and maintain. This is more important for the NMSs, since the DMBS is usually maintained by the personnel in the network operation centers who are not necessarily familiar with database issues. Requirements on high availability and historical data archival make the OA&M issues more complex.
- The DBMS should support parallel query capabilities to utilize SMP or parallel machines for enhancing the performance of data storage and query processing.
- Interaction with external processes : It is desirable for the DBMS to provide a mechanism for the database applications for communicating with one another through the database.
- Support tools : Many off-the-shelf tools from independent vendors are necessary to reduce the efforts for application developments.
- Multiple National Language Support : As more systems are used internationally, the DBMS needs to support multiple national languages and handle different formats for date, time, and currency data.
- Hardware Independence : The DBMS should be available or portable across multiple hardware platforms so that the NMSs can be moved to a new hardware if necessary.
- Vendor Independence : It is desirable to remain vendor independent as much as possible so that the DBMS can be replaced with ease. Though most DBMSs support the ANSI SQL standard, each DBMS contains many extensions of its own outside the scope of the standard. For example, most commercial DBMSs support stored procedures and trigger. Since they are not in the SQL standard yet, they are incompatible with one another. As special features of a particular DBMS is utilized for

to meet the performance requirements, it becomes more difficult to move to a different DBMSs.

4. *Conclusions*

There are many types of computer systems being used to support the operation and management of the telecommunications network. Most of these systems need to deal with a large volume of data, so database management systems play a critical role in these systems. Major DBMS issues for the NMSs include choosing the right data model for information models of network management, handling two different kinds of data in terms of integrity and recovery constraints, supporting temporal queries, satisfying real-time performance and high availability requirements, and several miscellaneous issues.

Some of these issues have been investigated in various areas of database researches, *e.g.* real-time databases, temporal databases, distributed and parallel databases, query processing and optimization, *etc.* But most of them largely remain in the research stage, and no DBMS implementation is expected to be available in a foreseeable future as an integrated solution for actual applications that deal with a large volume of temporal data under real-time performance constraints.

The biggest issue of the DBMSs, especially the relational ones, supporting the NMSs in many cases is to meet the real-time performance constraints. Optimizing the performance is usually the utmost concern in designing applications in the critical path of real-time processing. Advances in the CPU power and the parallel architecture coupled with performance enhancements in query processing are eagerly awaited.

Though the relational DBMSs have several weaknesses in serving as the database engine for the NMSs, they are still most popular among all the database models. It would be interesting to see how the object-oriented DBMSs can be accepted for the NMSs as the ODMG standard becomes implemented with standard query language and application programming interface, while relational DBMSs are extended with the object support in SQL3 [Melton 93].

5. *References*

[Ahn 93] I. Ahn: *SQL+T : a Temporal Query Language*, Proceedings of the International Workshop on an Infrastructure for Temporal Databases, Jun. 1993.

[ANSI 86] American National Standard for Information Systems: *Database Language - SQL*, ANSI X3.135-1986, American National Institute, Inc., 1986.

[ANSI 89] American National Standard for Information Systems: *Database Language - SQL with Integrity Enhancement*, ANSI X3.135-1989, American National Institute, Inc., 1989.

[CCITT 92a] CCITT Recommendation M.3010: *Principles for a Telecommunications Management Network*, 1992.

[CCITT 92b] CCITT Recommendation X.722, ISO/IEC 10165-4: *Information Technology - Open Systems Interconnection - Structure of Management Information: Guidelines for the Definition of Managed Objects*, 1992.

[ISO 84] International Standards Organization: *Information Processing Systems - Open Systems interconnection - Basic Reference Model*, 1984.

[Klever 88] S. Klever: *The OSI Management Architecture: An Overview*, IEEE Network, Mar. 1988.

[Klever 93] S. Klever: *System Management Information Modeling*, IEEE Communications Magazine, May 1993.

[Melton 93] J. Melton (editor): *ISO-ANSI (Working Draft) Database Language SQL3*, X3H2-93-091/YOK-003, Feb. 1993.

[NMF 89] OSI Network Management Forum: *Object Specification Framework*, Sep. 1989.

[ODMG 93] Cattell et al: *The Object database standard : ODMG-93*, Morgan Kaufmann, 1993.

[OMNIPoint 93] OMNIPoint: *Discovering OMNIPoint*, Simon & Schuster, 1993.

[Ramamritham 92] K. Ramamritham: *Real-Time Databases*, International Journal of Distributed and Parallel Databases, 1992.

[Snodgrass & Ahn 86] R. Snodgrass, I. Ahn: *Temporal Databases, IEEE Computer*, Sep. 1986.

[Snodgrass 90] R. Snodgrass: *Temporal Databases: Status and Research Directions*, ACM SIGMOD Record, Vol. 19, No. 4, Dec. 1990, pp. 83-89.

[Soo 91] M. Soo: *Bibliography on Temporal Databases*, ACM SIGMOD Record, Vol. 20, No. 1, Mar. 1991, pp. 14-23.

[Ulusoy 92] O. Ulusoy: *Current Research on Real-Time Databases*, ACM SIGMOD Record, Vol. 21, No. 4, Dec. 1992, pp. 16-21.

[Yemini 93] Y. Yemini: *The OSI Network Management Model*, IEEE Communications Magazine, May 1993.