University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1995-16

Design Issues in Database Systems for Telecommunication Services*

Kimmo Raatikainen and Juha Taina

Helsinki, April 1995

Report C-1995-16

University of Helsinki
Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, FINLAND

* This report is based on work done in Darfin-project (Project No 30018) funded by Telecom Finland.
The papers in the series are intended for internal use and are distributed by the author. Copies may be ordered from the library of Department of Computer Science.

Design Issues in Database Systems for Telecommunication Services[†]

Kimmo Raatikainen and Juha Taina

Report C-1995-16
13 pages
Helsinki, April 1995

Abstract. In future telecommunication services based on the Intelligent Network databases will have an important role. The persistent and temporal information needed in operations and management of the telecommunication networks and services will be in databases. The current IN Recommendations of ITU-T (Q.1200 Series) imply that real-time transaction processing capabilities should be provided. We examine the design issues of real-time transaction processing in database architectures that can be integrated with teleoperator's service provision architecture. The issues examined include integration of concurency control and real-time scheduling, main-memory databases, and correctness criteria of transactions. In addition, we briefly outline how real-time features can be incorporated into the industrial object database standard called ODMG-93 and how the OMG Object Request Broker (CORBA) can be exploited in the future IN architectures.

Key Words: Real-Time Databases, Concurrency Control, Relaxed Serializability, Main Memory Databases, Telecommunication Software

CR Classification: C.2.1, H.2.4

[†]This report is based on work done for Darfin-project (Project No 30018) funded by Telecom Finland.

Contents

1. Introduction **1**

2. External Interfaces and Design Constraints **1**

2.1 IN Architectural Framework 1

2.2 Basic Reference Model of Open Distributed Processing 2

2.3 Telecommunication Management Network 3

2.4 Anticipated Database Needs 4

3. Real-Time Databases **4**

3.1 Principles of the Real-Time Database System 5

3.2 Designing an Real-Time Database System 5

3.2.1 Design Issues in Concurrency Control 6

3.2.2 Integrating Concurrency Control and Real-Time Scheduling 7

4. Main Memory Databases **8**

5. Correctness Criteria of Transactions **9**

6. Discussion **9**

References **11**

1. Introduction

Databases will, already in the near future, have an important role in telecommunication. The databases will contain the persistent and temporal information needed in operations and management of the telecommunication networks and services. Such databases exhibit stringent reliability, availability, and performance requirements. Thousands of retrievals must be delivered in a second. Allowed downtime is only a few seconds in a year. The globalization of services implies that databases managed by different operators must cooperate. The interoperability requires that the database systems must be open, must use standardized protocols, must be protected against misuse and intruders, and must have a common logical view of information.

In this paper we examine some fundamental design issues that affect the feasible architectures of database systems which can be integrated into teleoperator's service provision architecture. Our premises are primarily derived from the *Intelligent Network Long-Term Architecture* (IN LTA) framework as described in Recommendation Q.1201 [ITU 1993b]. The key premises are compatibility with the OSI *Reference Model of Open Distributed Processing* (RM-ODP) [ITU 1994c] and with the *Telecommunications Management Network* (TMN) [ITU 1993a]. Since both RM-ODP and TMN are based on object-oriented modelling, an object-oriented database is our first presumption. Our second presumption is scalability which implies that the database system may be distributed.

The rest of the paper is organized as follows. In Section 2 we briefly discuss our premises — the ways how the IN architecture, together with the compatibility with RM-ODP and TMN, affects the database architecture. In addition, we summarize our assumptions about the future database needs in telecommunication. In Section 3 we examine the design issues in real-time databases. In Section 4 the focus is on main memory databases and in Section 5 on correctness criteria of transactions. Finally, in Section 6, we relate the development to the work done in the Object Management Group (OMG) and in the Object Database Management System (ODMG) consortium.

2. External Interfaces and Design Constraints

In addition to the current ITU-T recommendations in Q.1200 Series (*Intelligent Networks*), the Basic Reference Model of ODP and the concepts of TMN create the architectural framework into which the database system is to be embedded. The architectural framework sets the requirements for external interfaces that the database system may have to support.

2.1 IN Architectural Framework

On the *distributed functional plane* in the IN architecture is described in terms of *functional entities*. In that model the database services are available in *Service Data Function* (SDF). The database users (clients of SDF) are *Service Control Functions* (SDFs), *Service Management Functions* (SMFs), and other SDFs. Figure 1 outlines the part of the IN functional architecture involving database services. The internal structures of SCF and SDF in the figure are based on the figures 4-19 and 4-20 in ITU-T Recommendation Q.1214 [ITU 1994b], respectively. The outlined internal structure of the SMF and its impact on the SDF are based on the principles of OSI Management [ITU 1991–4]. Detailed description of database access in the IN, especially in the Capability Set 1 (CS-1), can be found in Raatikainen [1994a].

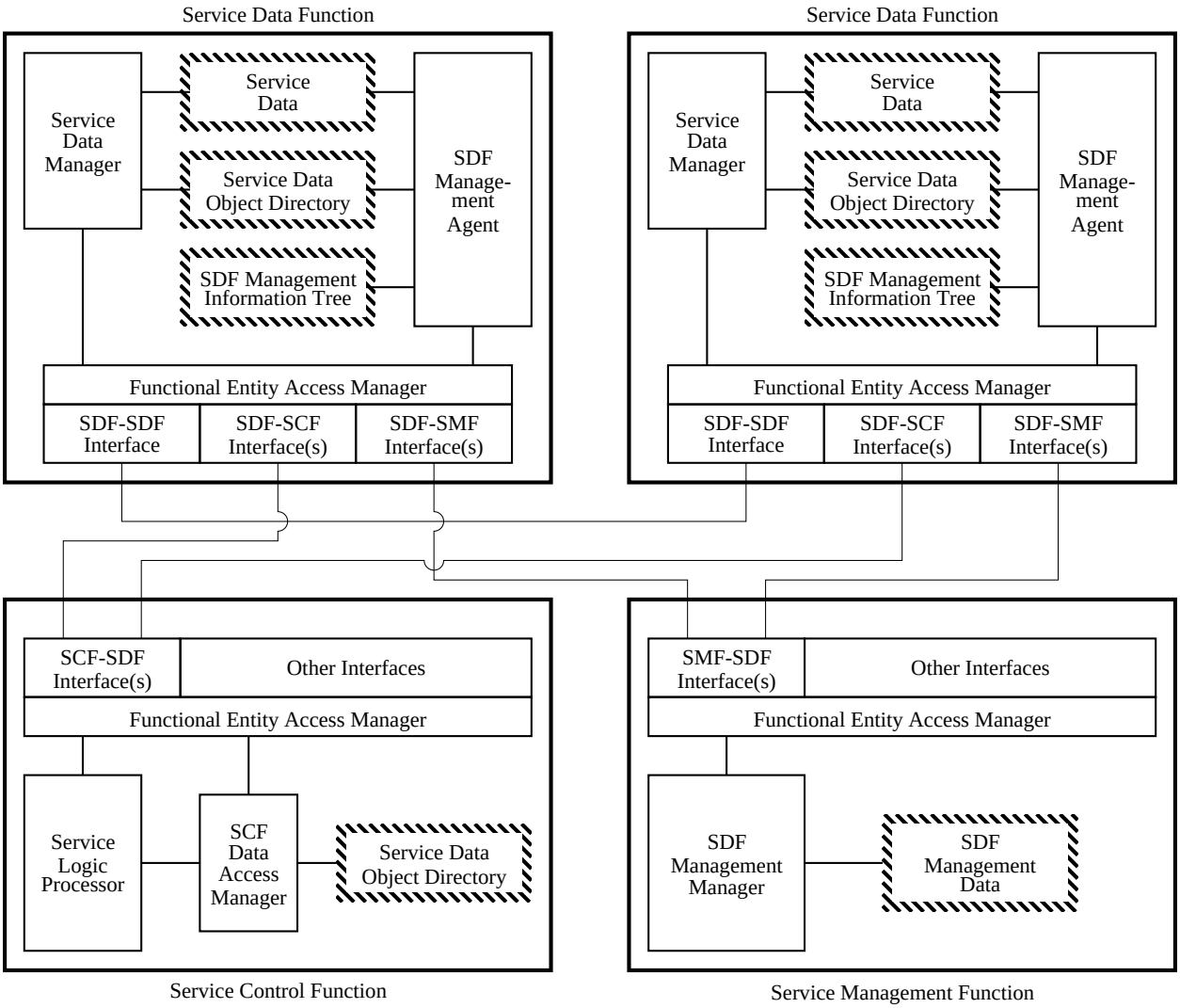

Figure 1: Elements of Database Servers and Clients in the IN Functional Architecture

2.2 Basic Reference Model of Open Distributed Processing

Open Distributed Processing is a joint standardization initiative by ISO and ITU-T. The Basic Reference Model of ODP is a framework for standardization of ODP. The objective of RM-ODP is to create an architecture that supports distribution, interworking, interoperability, and portability. All these are important goals for future telecommunication.

The engineering viewpoint of RM-ODP defines a model for the infrastructure of distributed system. The model, in turn, can be viewed as an abstract platform for ODP applications. The fundamental entities described in the engineering viewpoint are basic engineering objects, infrastructure objects, and channels. A channel provides the communication mechanism. In addition, the transparency functions are contained or controlled by a channel. The compatibility with RM-ODP requires that

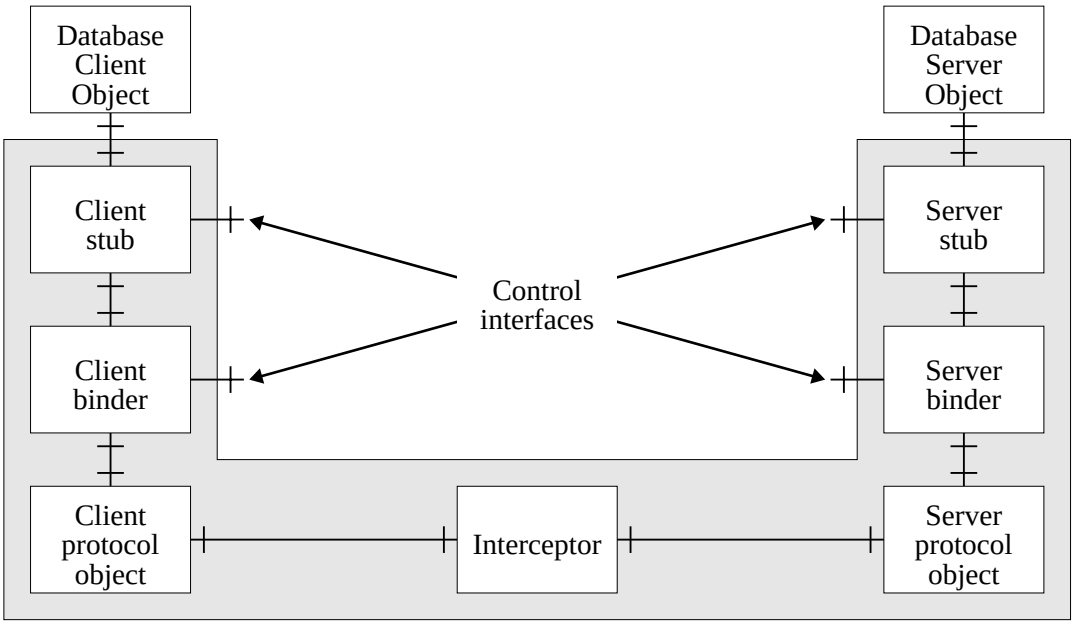

Figure 2: Structure of an ODP Channel

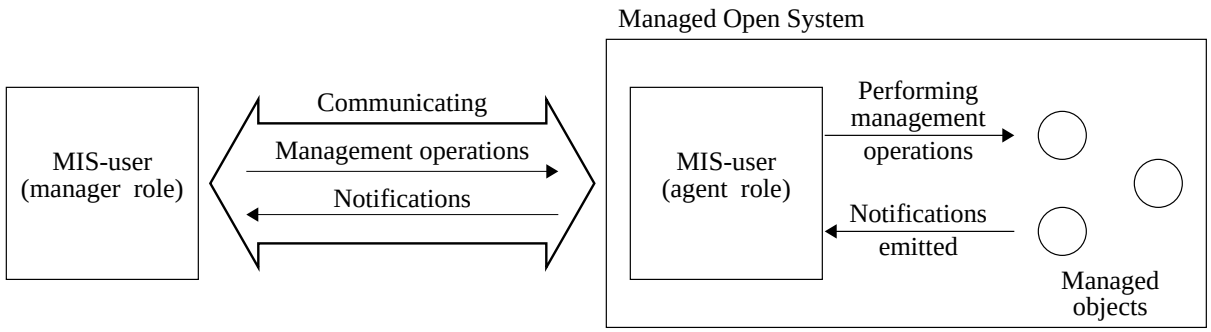

Figure 3: OSI System Management Interactions

the database system must be accessible through an ODP-channel. Figure 2 illustrates the structure of channel between the Database Server Object and a Database Client Object.

2.3 Telecommunication Management Network

Telecommunication Management Network is a generic architecture to be used for all kinds of management services. TMN is based on the principles of the OSI Management as specified in the Recommendations in the X.700 Series [ITU 1991–4].

The fundamental idea in the OSI Management is that the knowledge representing the information used for management is separated from the functional modules performing the management actions. OSI Management is based on interactions between management applications that can take the roles of *manager* and *agent*. As shown in Figure 3 the interactions that take place are abstracted in

terms of *management operations* and *notifications*. Management activities are effected through the manipulation of *managed objects* (MOs).

An agent manages the MOs within its local system environment. It performs management operations on MOs as a consequence of management operations issued by a manager. An agent may also forward notifications emitted by MOs to a manager. The agent maintains a part of the *Management Information Tree* (MIT), which is a dynamic database. The MIT contains instances of MOs organized as a hierarchical database tree.

Agents and managers exchange information using the services of Common Management Information Service Element (CMISE). CMISE uses the *Common Management Information Protocol* (CMIP) and utilizes the services of ACSE and ROSE. In brief, the principles of OSI Management (and TMN) require that the database system contains the functionality of an OSI Management Agent.

2.4 Anticipated Database Needs

In Raatikainen [1994b] we have identified five basic types of database operations needed in services and service features of IN Capability Set 1:

1) Retrieval of structured objects from persistent subscriber data objects. Some of the retrievals trigger a later update. For example, if Freephone service provides call routing statistics to the subscriber, an update is necessary.
2) User management actions that modify persistent subscriber data objects. These actions must be protected against unauthorized use. Some updates may also have to be verified against other criteria. For example, the owner of a telephone number must have the possibility to cancel or prohibit call forwarding to his or her number.
3) Verification of Personal Identification Number. For security reasons the PIN verification must be done in the database system when services are globalized. The actual PIN cannot be delivered into a foreign network.
4) Writing sequential log records.
5) Mass calling and Televoting. They need high-volume small updates that must be serialized. A special aspect of these updates is that they can be done in large blocks.

In addition to the IN services, future databases must support mobility and management. From databases' point of view, the most challenging aspect of mobility is the maintenance of location information. This contradicts the traditional claim that most database operations needed in telecommunication services are reads. The management implies a twofold functionality. Firstly, the database and its elements are managed objects. Secondly, a management agent needs an efficient object-oriented database to maintain the Management Information Base.

3. Real-Time Databases

Recommendation Q.1204 (*Intelligent Network Distributed Functional Plane Architecture*) [ITU 1994a] specifies that the *Service Data Function* (SDF) provides real-time access to customer and network data. This statement raises the question whether a real-time database is needed. Recommendation Q.1211 (*Introduction to Intelligent Network Capability Set 1*) [ITU 1993c] augments the functional specification of SDF by requiring that SDF provides consistency checks on data. This raises the fundamental question whether real-time transaction processing capabilities are needed.

3.1 Principles of the Real-Time Database System

According to Haritsa *et al.* [1990] *Real-Time Database System* (RT-DBS) is a transaction processing system that tries to satisfy the explicit timing constraints associated with each incoming transaction. The timing constraint is usually given as a *deadline*. The ultimate goal of an RT-DBS is to maximize the fraction of transactions which meet their deadlines. This is in contrast to the goal of traditional database systems (DBMSs) which is to minimize the average response time.

At any time, all transactions in an RT-DBS can be divided into two classes: *feasible* transactions and *late* transactions. A feasible transaction still has a possibility to meet its deadline. A late transaction has either already missed its deadline or cannot any more meet its deadline. The RT-DBS executes feasible transactions until they complete in time or until the system detects that they can no more meet their deadlines. The policy how late transactions are handled depends on the application needs. The two primary policies are: 1) to continue the execution with a low priority and 2) to abort the transaction without restart.

Real-time systems are usually classified into three categories according to the effects that missing the transaction's deadline has. *Hard* deadline transactions are those which may result in a catastrophe if the deadline is missed. A hard real-time system must guarantee that any transaction never misses its deadline. *Firm* deadline transactions are those the results of which are not any more of any value if the deadline is missed but have no serious consequences. *Soft* deadline transactions are those the results of which gradually decrease their value when the deadlines are missed.

The deadlines raise several difficult questions. How are deadlines assigned? Are deadlines firm or soft? It must be remembered that a transaction accessing the database is only a subtask in a telecommunication service. Unfortunately *subtask deadline assignment* is still almost an unexamined problem [see e.g. Kao and Garcia-Molina 1993]. The primary implication of transactions being subtasks is that the task can still meet (miss) its deadline even if the transaction misses (meets) its deadline.

Since the main goal of a firm or soft RT-DBS is to maximize the fraction of transactions that meet their deadlines, a key issue in transaction processing is *predictability* of transactions' execution and turnaround times. In a database system, data and hardware resource conflicts are an important source of unpredictability. Another significant source is the unpredictability of execution time caused by execution sequence depending on data values and by disk operations. Distributed databases have additional problems due to communication delays and site failures.

3.2 Designing an Real-Time Database System

One of the primary difficulties in designing an RT-DBS is the fact that the concurrency control of data access must be combined with timing constraints. The two primary approaches to arrange the concurrency control in database systems are based on *locking* and on the *optimistic approach*. Most commercial DBMSs are based on *two-phase locking* (2PL) [Eswaran *et al.* 1976]. This is due to the fact that under most operating circumstances locking algorithms give shorter response times than optimistic algorithms [see e.g. Agrawal *et al.* 1987].

Recent performance studies [Abbott and Garcia-Molina 1992, Haritsa *et al.* 1990, Huang *et al.* 1992, and Lin and Son 1990, among many others] indicate that the results obtained from traditional DBMSs do not hold in RT-DBSs. The relative performance of concurrency control algorithms

in real-time database systems is heavily affected by several factors including policy with late transactions, a priori knowledge of transaction resource requirements, and the availability of resources. This leaves the floor open to several design issues that must be evaluated.

3.2.1 Design Issues in Concurrency Control

The major design issues in concurrency control are conflict detection and conflict resolution. In addition, the run policy is an important issue. Below we discuss these issues separately although they are not independent. We also name several mechanisms proposed in the literature.

Conflict Detection. Conflicts in data granule access that violates the correctness criteria of a transaction can be detected either before the granule access (*pessimistic approach*) or after the granule access (*optimistic approach*). In the optimistic approach the correctness is later verified at the certification time. The mechanisms proposed to be used in the detection include *locks*, *time stamps*, and *serialization graphs*.

When a lock-based scheme is used, a lock must be obtained before an access is allowed. In pessimistic approaches, such as various 2PL schemes, the lock is either shared (*read*) or exclusive (*write*). When an optimistic approach is based on locks, there are *weak* locks as well as normal (*strong*) locks. When a transaction has a weak lock, either read or write lock, on a granule, it only indicates that the granule is accessed. A weak lock conflicts with a strong write lock but is compatible with other weak locks (both read and write) and with strong read locks. Typically, the weak locks are converted to strong locks at the beginning of the certification time.

In some situations two different kinds of write locks may be profitable: an *update lock* (ordinary write lock) and a *replace lock* (blind write lock). Two *blind write* operations do not conflict with each other, since the new value becomes outside the database.

Conflict Resolution. A conflict resolution mechanism is invoked once a conflict is detected. In the resolution at least one transaction is penalized through appropriate actions selected by the conflict resolution mechanism. The actions most commonly used are *blocking* (wait), *abort* (restart), *multiversioning*, and *dynamically readjusting the serialization order* (delayed commit or abort). It should be noted that the resolution mechanism is not independent from the detection mechanism. For example, if the conflict is detected after the granule access, the action must be abort.

Several conflict resolution mechanisms have been proposed in the literature. Variants of the 2PL include *no wait* [Tay *et al.* 1985], *wound-wait* [Rosenkrantz *et al.* 1978], *wait-die* [Rosenkrantz *et al.* 1978], *running priority* [Franaszek and Robinson 1985], *wait depth limited* [Franaszek *et al.* 1992], and *delayed abort* [Yu 1990]. In addition to the pure *optimistic concurrency control* (OCC) *broadcast OCC* [Kung and Robinson 1981] and a combination of pure and broadcast OCC [Yu *et al.* 1993] have been proposed. *Locking with deferred blocking* [Yu and Dias 1993] is a scheme that combines 2PL and OCC. Multiversioning opens a wide variety of conflict resolution mechanisms. A detailed comparison can be found in [Wu *et al.* 1993].

Run Policy. When abort is used as an action in the conflict resolution, a transaction may need to be restarted several times. The restarts create an effect known as *buffer retention* [Yu and Dias 1992]. In the first run it may be profitable to run a transaction that will be aborted to end so that all granules accessed will be in the buffer and the CC manager learns all the locks the transaction needs.

3.2.2 Integrating Concurrency Control and Real-Time Scheduling

There are four major problem areas when a real-time transaction scheduler is to be developed: *priority inversion*, *scheduling priority*, *overload management*, and *IO scheduling*.

Priority Inversion. A phenomenon called *priority inversion* arises when a low-priority transaction blocks a high-priority transaction due to the concurrency control. At least three different approaches have been proposed to solve the problem. In *priority inheritance* [Sha *et al.* 1990] the low-priority transaction inherits the priority of the high-priority transaction. In the *priority ceiling protocol* [Sha *et al.* 1991] blocking time of the high-priority transaction is bounded. The *dynamic priority ceiling protocol* [Chen and Lin 1990] is a modification of the previous based on earlist deadline first instead of fixed external priorities.

Scheduling Priority. *Earlist Deadline First* (EDF) is most widely used in the experimental real-time database systems as the scheduling priority. It can minimize the number of late transactions expect when the system is highly loaded [Liu and Layland 1973]. Another drawback in the EDF is that large transactions are discriminated. Alternatives to the EDF include *Least Slack First* (LSF), *fixed priorities*, *criticality of transaction*, *weighted priority scheduling*: see e.g. Abbott and Garcia-Molina [1992], Buchmann *et al.* [1989], Sha *et al.* [1991], Huang *et al.* [1989], Huang *et al.* [1991]. The fundamental design criteria are the heterogenity of transactions and transaction characteristics available a priori.

Overload Management. If the EDF scheduling is used, then occasional periods of high load can significantly decrease the fraction of transactions that complete in time. In order to cope with the overload situations the scheduling needs an overload management policy. In Haritsa *et al.* [1991] an *adaptive earlist deadline* (AED) scheduling scheme was proposed. In AED, arriving transactions are placed in a HIT group or a MISS group depending on the system condition. In the HIT group the transactions are treated in the normal way but in the MISS group the transactions get low priority. In Pang *et al.* [1992] an *adaptive earlist virtual deadline* (AEVD) scheduling scheme was proposed. AEVD is an extension of AED that tries to take into account the fairness issue in an overloaded system.

IO Scheduling. When the primary copy of the database is on the disk, IO operations are extremely important for the RT-DBS. Traditional disk scheduling algorithms should be replaced by novel ones as in Sprunt *et al.* [1988] and in Abbott and Garcia-Molina [1990]. When a large fraction of database can be held in the main memory buffers, the situation is completely different. We believe that in such cases hybrid databases (see the next section) should be used.

4. Main Memory Databases

In real-time systems the unpredictability of database operations is decreased by letting the principal version of the database permanently reside in the main memory. However, the main memory database systems (MM-DBS) have several consequences that cannot be solved in the same way as in traditional DBMSs.

The differences between main memory and disk affect almost every aspect of database management. The key differences are: 1) access times (microseconds vs. milliseconds), 2) main memory is usually volatile, 3) sequential and random access times are almost the same for main memory but not for disk, 4) access time for main memory does not have a long startup time as for disk, 5) data on disk are only accessible through system software.

When we consider the suitability of an MM-DBS, the fundamental question is whether it is reasonable to assume that the entire database fits in a single main memory. Our premise of scalability dictates that the assumption can not be done. This implies that we must design an distributed MM-DBS or we must keep only a part of the database permanently in main memory.

As presented in Garcia-Molina and Salem [1992] the data can often be classified into: 1) *hot* data that are accessed frequently or that have stringent deadlines and 2) *cold* data that are accessed rarely and that do not have stringent deadlines. Based on the data heat, it may be possible to partition the database into two or more logical databases. Only the hottest partition is permanently kept in main memory. The motivation is the so called 80/20 rule: Only 20% of data is accessed 80% of time. It is also possible to design a database architecture, in which the partitioning is dynamic. The data resides in main memory whenever possible. The location changes dynamically from main memory to disk and back according to some dynamic priority. Naturally, such modifications to a pure MM-DBS,which are known as hybrid databases, increase the complexity of the database architecture.

Because the principal version of the database is kept in the main memory, the database also needs a backup copy on the secondary nonvolatile storage. Despite of recent developments in memory technology (battery-backed up memory boards, uninterruptable power supplies, error detecting and correcting memories, etc.), we can not assume that the probability of main memory faults is zero. Moreover, even if the probability of media failure were zero, system crashes due to software faults will still (even in the future) have a nonzero probability.

In the recovery, the worst case scenario is that the data must be reloaded from the secondary storage. The reload will take a long time, since data rates for commercial disks devices are still today under 5 MB/s [Wood and Hodges 1993]. There are two primary ways to reduce the downtime due to memory crashes. The first one is based on duplicated systems. The main memory of one system is mirrored in another system. The side effect is that each update must take place in two systems and these updates must be synchronized. The second way is based on incremental recovery; see e.g. Levy and Silberschatz [1992]. The data are reloaded on demand when needed as a background task. The consequence is that the recovery will take a long time during which the performance of the database system is degenerated.

According to Eich *et al.* [1989] concurrency control may not at all be needed in an MM-DBS. However, this is possible only when the whole database is in one main memory and only one

processor is used. Although concurrency control is a very important question in databases, in MM-DBSs the question has not systematically been examined.

5. Correctness Criteria of Transactions

In the traditional DBMSs concurrency control restricts concurrent execution of transactions so that the transactions *serialize* [Eswaran *et al.* 1976]. Serializability, which is equivalent to a serial execution of the transactions, is well established as an appropriate notion of correctness for schedules of interleaved transactions. However, it also increases the number of data conflicts.

In real-time applications concurrent transactions do not necessarily need serializability to produce a correct result. In many situations the result of a transaction can be regarded as correct as long as the result corresponds to the real-world situation presented in the database even if the transaction does not serialize with other transactions. According to Yu *et al.* [1994] *temporal consistency* is often more important than the internal consistency obtained through serializability. Temporal consistency requires that data values used by a transaction have existed about the same time.

In telecommunication there are several database operations that do not need to serialize. However, some transactions need serializability as identified in Section 2.4. If the situation is such that some data items need strict serializability while other items do not, the situation can be solved by the explicit use of weak and strong locks. However, if some transactions require strict serializability while some other transactions are satisfied by weaker constraints and these transactions access the same data items, the situation is quite complex.

An interesting correctness criteria, which we call τ-*serializability*, can be used to reduce the number of read-write conflicts. Suppose that transaction T_A updates the data item x and gets a write lock on x at time t_a. Later (but before T_A either commits or aborts) transaction T_B wants to read the data item x. Let t_b be the time when T_B requests the read lock on x. In τ-serializability, the two locks do not conflict if $t_a + \tau > t_b$. The tolerance τ (how long the old value is useful) may depend on x (on data semantics) as well as on T_A and T_B (on application semantics).

An important open question is global distributed transactions, in which several autonomous database systems are involved. If future telecommunication services need such global transactions, the database must support the Distributed Transaction Processing services as specified in Recommendation X.861 [ITU 1993d]. It should be noted that this distribution is different from the internal distribution when clients of the database system see a single database system even if the actual DBS is distributed.

6. Discussion

When a database systems for telecommunication services is to be designed the most important issues include *real-time*. We believe that an RT-DBS will in the future have an important role in telecommunication and the both *firm* and *soft* deadlines will be needed.

Since the needs for database services have quite different characteristics, the database system should be quite flexible. We believe that the required functionality can most easily and economically be achieved by an *object-oriented* approach. The primary advantage in the object orientation

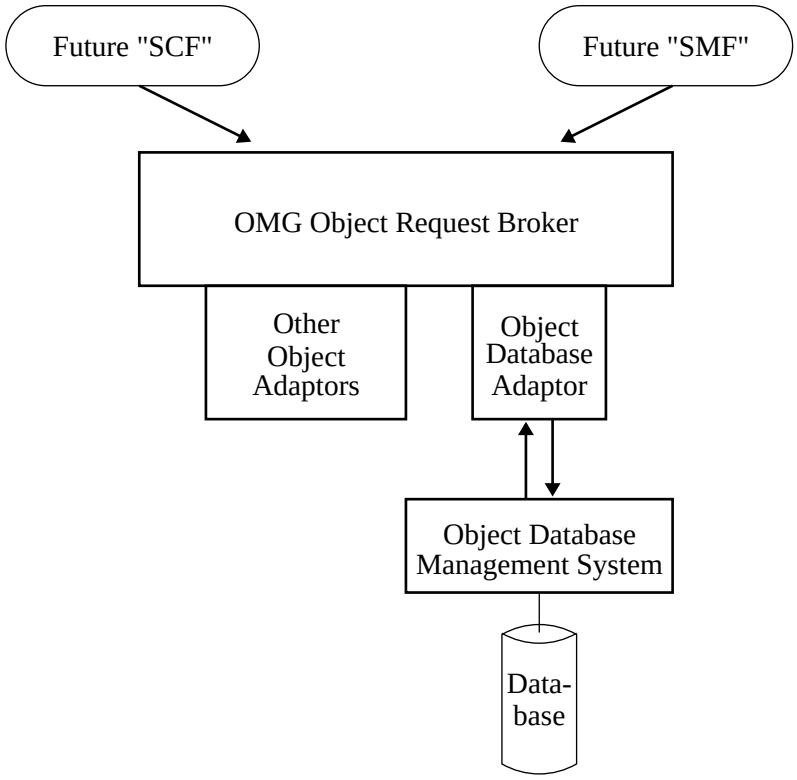

Figure 4: Database Service Architecture for future IN

is that we obtain an elegant way to incorporate the additional information needed in real-time concurrency control and scheduling into the database. Data objects can have attributes that specify the correctness criteria. Operations can have attributes that tell the resource consumptions of the operation. Transactions are also objects; either transient or persistent. They can have attributes that specify the priority, deadline, criticality, correctness criterion, etc. However, we must remember that *the real-time* is an almostly unexplored area in object-oriented database research.

One interesting path is to examine how the real-time features can be introduced into the ODMG-93 standard [Cattell 1994]. We are, however, sure that all features presented in the ODMG-93 Release 1.1 are neither appropriate nor needed in a real-time database. Therefore, an *"RT-ODMG"* will be a subset of ODMG-93 with predefined types of `RealTimeObject` and `RealTimeTransaction`. Type `RealTimeObject` will include attributes that specify the correctness criteria and resource demands of operations. Type `RealTimeTransaction` will include attributes that may overwrite the correctness criteria of operations and that specify the deadline and criticality for the instances of this type.

The ODMG Object Model is a superset of the OMG (Object Management Group) Object Model [OMG 1992a]. The OMG *Object Request Broker* (CORBA) [OMG 1992b] is one possible way to invoke database services. Figure 4 outlines how future SCFs and SMFs may obtain database services through the OMG ORB.

Acknowledgements

The authors are thankful to the rest of the Darfin[1]-team, particularly to Mika Rautila, Tapani Karttunen, and Pekka Olli.

References

Abbott, R. and Garcia-Molina, H. 1990. Scheduling I/O Requests with Deadlines: A Performance Evaluation. In *Proceedings of the 11th Real-Time Systems Symposium*. IEEE, Los Alamitos, Calif., 113–124.

Abbott, R. and Garcia-Molina, H. 1992. Scheduling Real-Time Transactions: A Performance Evaluation. *ACM Transaction on Database Systems 17*, 3 (Sept.), 513–560.

Agrawal, R., Carey, M. J., and Livny, M. 1987. Concurrency Control Performance Modeling: Alternatives and Implications. *ACM Transaction on Database Systems 12*, 4 (Dec.), 609–654.

Buchmann, A. P., McCarthy, D. R., Hsu, M., and Dayal, U. 1989. Time-Critical Database Scheduling: A Framework For Integrating Real-Time Scheduling and Concurrency Control. In *Proceedings of the 5th International Conference on Data Engineering*. IEEE, Los Alamitos, Calif., 470–480.

Cattell, R. G. G., Editor. 1994. *The Object Databse Standard: ODMG-93*, Release 1.1. Morgan Kauffmann, San Francisco, Calif.

Chen, M.-I. and Lin, K.-J. 1990. Dynamic Priority Ceilings: A Concurrency Control Protocol for Real-Time Systems. *The Journal of Real-Time Systems 2*, 4 (Dec.), 325–346.

Eich, M. H., Corti, H., Gruenwald, L., and Mariategui, F. 1989. Main Memory Recoverable Database with Stable Log. Tech. Rep. 89-CSE-38, Southern Methodisth University, Dept. of Computer Science.

Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L. 1976. The Notions of Consistency and Predicate Locks in a Database System. *Commun. ACM 19*, 11 (Nov.), 624–633.

Franaszek, P. A. and Robinson, J. T. 1985. Limitations of Concurrency in Transaction Processing. *ACM Transactions on Database Systems 10*, 1 (Mar.), 1–28.

Franaszek, P. A., Robinson, J. T., and Thomasian, A. 1992. Concurrency Control for High Contention Environments. *ACM Transactions on Database Systems 17*, 2 (June), 304–345.

Garcia-Molina, H. and Salem, K. 1992. Main Memory Database Systems: An Overview. *IEEE Transactions on Knowledge and Data Engineering 4*, 6 (Dec.), 509–516.

Haritsa, J. R., Carey, M. J., and Livny, M. 1990. On Being Optimistic about Real-Time Constraints. In *Proceedings of the 9th ACM Symposium on Principles of Database Systems*. ACM, 331–343.

Haritsa, J. R., Livny, M., and Carey, M. J. 1991. Earliest Deadline Scheduling for Real-Time Database Systems. In *Proceedings of the 12th Real-Time Symposium*. IEEE, Los Alamitos, Calif., 232–242.

Huang, J., Stankovic, J. A., Towsley, D., and Ramamritham, K. 1989. Experimental Evaluation of Real-Time Transaction Processing. In *Proceedings of the 10th Real-Time Systems Symposium*. IEEE, Los Alamitos, Calif., 144–153.

Huang, J., Stankovic, J. A., Ramamritham, K., and Towsley, D. 1991. Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes. In *Proceedings of the 17th VLDB Conference*. Morgan Kaufmann, San Mateo, Calif., 35–46.

Huang, J., Stankovic, J. A., Ramamritham, K., Towsley, D., and Purimetla, B. 1992. Priority Inheritance in Soft Real-Time Databases. *The Journal of Real-Time Systems 4*, 2 (June), 243–268.

ITU. 1991–4. *Recommendations in X.700-Series: OSI Management*. International Telecommunications Union, Geneva, Switzerland.

ITU. 1993a. *Recommendations in M.3000-Series: Telecommunication Network Management*. International Telecommunications Union, Geneva, Switzerland.

[1]Research project *Database ARchitecture For Intelligent Networks* funded by Telecom Finland

ITU. 1993b. *Principles of Intelligent Network. Recommendation Q.1201.* International Telecommunications Union, Geneva, Switzerland.

ITU. 1993c. *Introduction to Intelligent Network Capability Set 1. Recommendation Q.1211.* International Telecommunications Union, Geneva, Switzerland.

ITU. 1993d. *Open Systems Interconnection - Distributed Transaction Processing: Service Definition. Recommendation X.861.* International Telecommunications Union, Geneva, Switzerland.

ITU. 1994a. *Intelligent Network Distributed Functional Plan Architecture. Recommendation Q.1204.* International Telecommunications Union, Geneva, Switzerland.

ITU. 1994b. *Distributed Functional Plane for Intelligent Network CS-1. Recommendation Q.1214.* International Telecommunications Union, Geneva, Switzerland.

ITU. 1994c. *Basic Reference Model of Open Distributed Processing — Parts 1–3: Overview, Descriptive Model, and Prescriptive Model. Draft Recommendations X.901, X.902, and X.903.* International Telecommunications Union, Geneva, Switzerland.

Kao, B. and Garcia-Molina, H. 1993. Deadline Assigment in a Distributed Soft Real-Time System. In *Proceedings of the 13th Int. Conf. on Distributed Computing Systems.* IEEE Computer Society Technical Committee on Distributed Processing, Los Alamis, Calif., 428–437.

Kung, H. T. and Robinson, J. T. 1981. On Optimistic Methods for Concurrency Control. *ACM Transaction on Database Systems 6*, 2 (June), 213–226.

Levy, E. and Silberschatz, A. 1992. Incremental Recovery in Main Memory Database Systems. *IEEE Transactions on Knowledge and Data Engineering 4*, 6 (Dec.), 529–540.

Lin, K.-J. and Son, S. H. 1990. Concurrency Control in Real-Time Databases by Dynamic Adjustment of Serialization Order. In *Proceedings of the 11th Real-Time Systems Symposium.* IEEE, Los Alamitos, Calif., 104–112.

Liu, C. L. and Layland, J. W. 1973. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM 20*, 1 (Jan.), 46–61.

OMG. 1992a. *Object Management Architecture Guide.* Number Number 91.11.1. Revision 2.0 in OMG TC Document. John Wiley & Sons, New York, N.Y.

OMG. 1992b. *The Common Object Request Broker: Architecture and Specification.* Number Number 91.12.1. Revision 1.1 in OMG Document. John Wiley & Sons, New York, N.Y.

Pang, H., Livny, M., and Carey, M. J. 1992. Transaction Scheduling in Mulriclass Real-Time Database Systems. In *Proceedings of the 13th Real-Time Systems Symposium.* IEEE, Los Alamitos, Calif., 23–34.

Raatikainen, K. E. E. 1994a. Database Access in Intelligent Networks. In *Proceedings of IFIP TC6 Workshop on Intelligent Networks.* Lappeenranta University of Technology, Lappeenranta, Finland, 163–183.

Raatikainen, K. E. E. 1994b. Information Aspects of Services and Service Features in Intelligent Network Capability Set 1. Report C-1994-45, University of Helsinki, Dept. of Computer Science, Helsinki, Finland.

Rosenkrantz, D. J., Stearns, R. E., and Lewis II, P. M. 1978. System Level Concurrency Control for Distributed Database Systems. *ACM Transactions on Database Systems 3*, 2 (June), 178–198.

Sha, L., Rajkumar, R., and Lehoczky, J. P. 1990. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Transactions on Computers C-39*, 9 (Sept.), 1175–1185.

Sha, L., Rajkumar, R., Son, S. H., and Chang, C.-H. 1991. A Real-Time Locking Protocol. *IEEE Transactions on Computers 40*, 7 (Jan.), 793–800.

Sprunt, B., Kirj, D., and Sha, L. 1988. Priority-Driven, Preemptive I/O Controllers for Real-Time Systems. In *Proceedings of Int. Symp. on Computer Architecture.* ACM, 152–159.

Tay, Y. C., Suri, R., and Goodman, N. 1985. A Mean Value Performance Model for Locking in Databases: The No-Waiting Case. *Journal of the ACM 32*, 3 (July), 618–651.

Wood, C. and Hodges, P. 1993. DASD Trends: Cost, Performance, and Form Factor. *Proceedings of the IEEE 81*, 4 (Apr.), 537–585.

Wu, K.-L., Yu, P. S., and Chen, M.-S. 1993. Dynamic Finite Versioning: An Effective Versioning Approach to Concurrent Transaction and Query Processing. In *Proceedings of the nth Int. Conf. on*

Data Engineering. IEEE, Los Alamitos, Calif., 577–586.

YU, P. S. 1990. Effective Concurrency Control With No False Killing. *IBM Tech. Disclosure Bull. 33*, 2 (July), 193–194.

YU, P. S. AND DIAS, D. M. 1992. Analysis of Hybrid Concurrency Control for a High Data Contention Environment. *IEEE Transactions on Software Engineering SE-18*, 2 (Feb.), 118–129.

YU, P. S. AND DIAS, D. M. 1993. Performance Analysis of Concurrency Control Using Locking With Deferred Blocking. *IEEE Transactions on Software Engineering SE-19*, 10 (Oct.), 982–996.

YU, P. S., DIAS, D. M., AND LAVENBERG, S. S. 1993. On the Analytical Modeling of Database Concurrency Control. *Journal of the ACM 40*, 4 (Sept.), 831–872.

YU, P. S., WU, K.-L., LIN, K.-J., AND SON, S. H. 1994. On Real-Time Databases: Concurrency Control and Scheduling. *Proceedings of the IEEE 82*, 1 (Jan.), 140–157.

Helsinki 1995
Yliopistopaino