University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1996-30

Platform Requirements of a Real-Time Object-Oriented Database Management System for Telecommunications

Juha Taina, Petri Elovaara, Tiina Niklander, and Kimmo Raatikainen

Helsinki, May 1996

Report C-1996-30

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

Platform Requirements of a Real-Time Object-Oriented Database Management System for Telecommunications

Juha Taina, Petri Elovaara, Tiina Niklander, and Kimmo Raatikainen

Department of Computer Science, University of Helsinki

Report C-1996-30
May 1996
18 pages

Abstract. Today the need for real-time services is rapidly growing in the telecommunications. Another important trend is the willingness to provided new services exploiting software solutions that are based on object-orientation. Therefore, the telecommunication solutions of tomorrow will need real-time object-oriented database management system (RT-OODBMS). An RT-OODBMS can be implemented on the top of a real-time platform if key features are supported in the platform. In this paper we discuss the platform support needed by a particular database architecture developed in our research projects. Several commercial real-time platforms have nice features but the situation is roughly the same as in ordinary database systems: general purpose (real-time) platforms can only provide rudimentary support. Several essential properties must be implemented in the database system.

Key Words: Real-time databases, object-orientation, real-time platforms.

CR Classification: C.3, D.4.7, H.2.4, J.7

Contents

1 Introduction

Databases will have a dominant role in future telecommunication services, for example in those based on the Intelligent Network (IN) architecture [Q.1201] and on the Telecommunications Information Networking Architecture (TINA) [Tina95,DuNI95]. Databases will hold the information needed in operations and management of telecommunication services and networks. The performance, reliability, and availability requirements of data access operations are hard. Thousands of retrievals must be executed in a second. The allowed downtime is only a few seconds in a year.

Management of telecommunications services and networks requires that the database and its elements must be manageable through the Telecommunications Management Network (TMN) [M.3010]. In other words, the database system must have the functionality of an OSI Management Agent [X.701], since database and its elements are managed objects. Additional requirements for database services (see [Raat95]) can be derived from ITU-T Recommendations on Intelligent Networks [Q.1204,Q.1211]. In the IN architecture the database services are provided by a functional entity called Service Data Function (SDF). The SDF is defined to contain customer and network data for real time access. The SDF is also required to provide consistency checks on data. In other words, the SDF should provide real-time transaction processing capabilities.

In this paper we examine the functional requirements that a real-time object-oriented database management system (RT-OODBMS) puts on the platform providing computing and communication capabilities. In particular, we are interested in features of a real-time operating system (RT-OS) and of supporting middleware useful in the implementation of an RT-OODBMS. Our starting point is the flexible database architecture developed in the Darfin[1] project. In our current project called RODAIN[1] we work toward a refinement of the architecture as well as toward a prototype implementation.

The rest of the paper is organized as follows. In Section 2 we briefly summarize the database operations needed in telecommunications. In Section 3 we discuss the essential features of a database architecture developed in our research projects. In Section 4 we discuss issues in the design and implementation of an RT-OODBMS. Finally, in Section 5 we present the requirements for platforms needed in the implementation of an RT-OODBMS.

[1]Darfin and RODAIN are acronyms of the project names *Database ARchitecture For Intelligent Networks* and *Real-time Object-Based Database Architecture for Intelligent Networks*, respectively.

2 Database Services Needed in Telecommunications

In telecommunications there are currently several architectures that must be taken into account when a database architecture to be used in telecommunications is to be designed. In addition to the current ITU-T Recommendations on Intelligent Networks (Q.1200-series), related architectures such as Telecommunications Management Network (M.3000-series), Telecommunications Information Networking Architecture (TINA), and the third generation mobile systems, in particular UMTS [IEEE95] and FPLMTS or IMT-200 [Call94], create design constraints.

Intelligent Network is a design platform for design, implementation, and provision of real-time telecommunication services. The following five basic types of database operations needed in the IN CS-1 were identified in [Raat94]:

1) *Retrieval of structured objects from persistent subscriber data objects.* Some of the retrievals trigger a later update.
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.
3) *Verification of Personal Identification Number.* For security reasons the PIN verification must be done in the database system when services are globalized.
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.

The most common operations are retrieving structured objects and writing sequential log records. Security operations, such as PIN verification, are also quite common. Management updates are not as common as the three first ones. Finally, mass calling and televoting occur only on special occasions. We anticipate that they will need special arrangements also in the future. Database service needed in the Intelligent Network architecture are described in details in [Raat95].

Telecommunications Management Network (TMN) is an architecture for supporting management activities associated with telecommunication networks [M.3010]. TMN provides management functions for telecommunication networks and services and offers communications between itself and the managed network. Conceptually TMN is a separate network that communicates with the managed network to control its operations.

Of the three TMN architectures (functional, information and physical) the TMN information architecture [X.720] is the most interesting for the database design. The TMN information network is based on OSI X.700 Management model. The OSI Management is based on interactions between management applications that can take the roles of *manager* and *agent*. Agents and managers interact with each other in terms of *management operations* and *notifications*. An agent manages *Managed Objects* (MOs) within its local environment. It performs management operations on MOs that a manager has requested. An agent also forwards notifications from MOs to a manager. The agent maintains a part of the *Management Information Tree* (MIT) that is a special kind of a distributed hierarchical object database. The TMN presumes that the database system has a management interface or functionality of an OSI Management Agent. In particular, the database system must support triggering of notifications.

The Telecommunications Information Networking Architecture Consortium (TINA-C) aims to define and validate an open architecture for telecommunications services in the current era. The architecture is based on distributed computing, object orientation, and other standards and recommendations in telecommunications and distributed processing fields, especially Open Distributed Processing (ODP), Intelligent Networks (IN), Telecommunication Management Networks (TMN), Asynchronous Transfer Mode (ATM), and Common Object Request Broker Architecture (CORBA) [NiDC95].

The TINA distributed data management is based on ODP [X.901] and CORBA [MoZa95]. As presented in [DuLN93] the ODP viewpoints may be used to separate the distributed data management issues and to identify the TINA concerns at each level. CORBA defines an object information highway that can be used to deliver ODP and TINA objects. The implications are that the database system must support the object invocation mechanisms of CORBA and communication through an ODP channel [X.903].

Currently the third generation mobile networks are emerging. Examples of such systems are the Future Public Land Mobile Telecommunication Systems (FPLMTS) and Universal Mobile Telecommunication System (UMTS). Standardization of FPLMTS which is also called as International Mobile Telecommunications 2000 (IMT-2000) is a current activity in ITU-T and ITU-R. Standardization of UMTS takes place in ETSI. Today, we do not know how similar the two sets of standards will be. We anticipate that the situation will be the same as in the past: ETSI's UMTS will be the pan-European guidelines for implementation of FPLMTS/IMT-2000.

Universal Mobile Telecommunication System (UMTS) provides wireless communication services to users on the move. The services will include high speed data services and video services like mobile videophone [Mitt95]. Whereas IN CS-1 services mostly issue read operations to IN databases, UMTS functions frequently issue update operations. In UMTS access delay to the data is of crucial importance and the number of accesses per user will be very high. Detailed performance and capacity requirements can be found in [Wirt95].

To summarize the essential requirements for database systems to be used in telecommunications

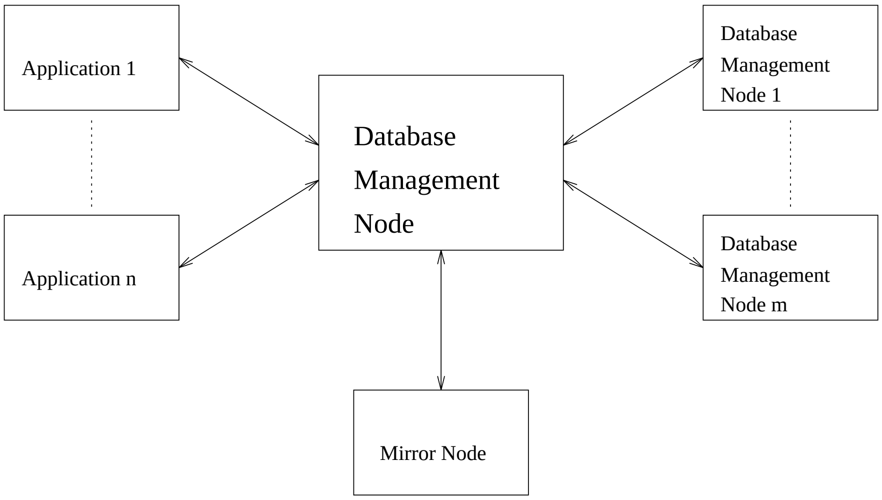

Figure 1: Overall Darfin Database Architecture

include [Tain95]:

- data distribution with and without distribution transparency;
- data replication to improve performance;
- object orientation in order to be conformant with TMN, TINA, and ODP;
- object directories in order to provide transparent access to objects;
- fault tolerance to provide nonstop services;
- real-time transactions in order to follow the recommendations and current trends.

3 Darfin Database Architecture

In the Darfin project we designed a database management system for telecommunications called *Darfin DBMS*. The Darfin database architecture, which is the basis of Darfin DBMS, has three primary levels: global architecture, node architecture, and object-oriented database management system. In this section we briefly outline the essential features of the architecture and Darfin DBMS. A detailed description can be found in [TaRa96].

3.1 Global Architecture

The Darfin DBMS architecture is an architecture for a real-time distributed object-oriented database management system. It consists of a set of autonomous database nodes that interact with each other. Each node may communicate with one or more applications, and an application may communicate

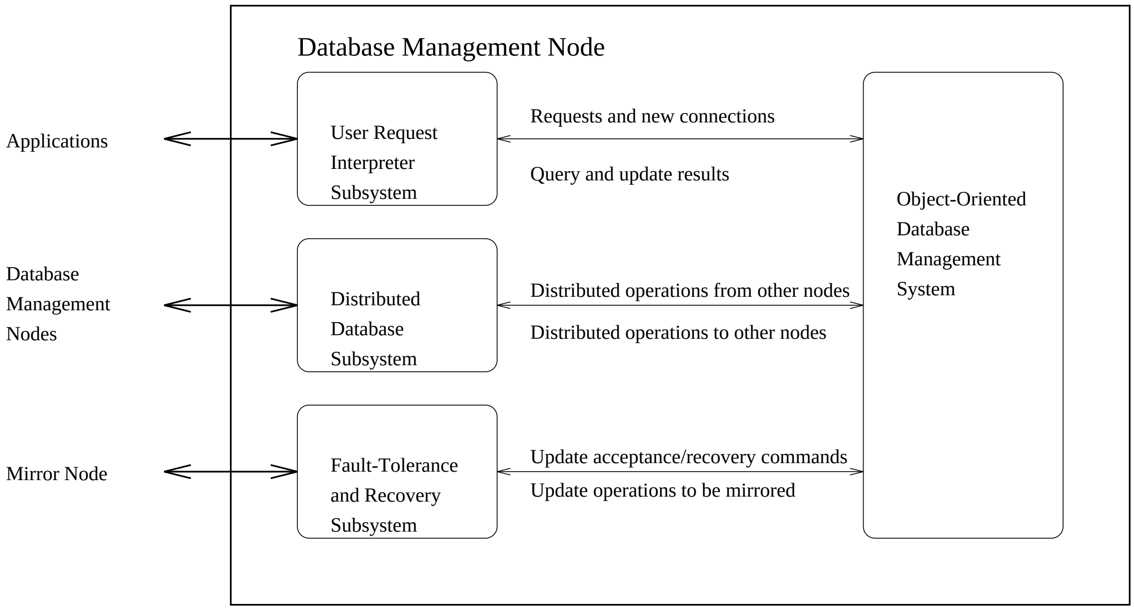

Figure 2: Darfin Database Management Node

with one or more database nodes. In order to reach the level of fault tolerance required in telecommunications each node is replicated in a mirror node. When the primary node fails, its mirror node takes control of database management during the recovery process. Figure 1 depicts the overall Darfin database architecture.

3.2 Node Architecture

A Darfin DBMS node consists of a set of subsystems that communicate with each other. The subsystems are: User Request Interpreter Subsystem, Distributed Database Subsystem, Fault-Tolerance and Recovery Subsystem, and Object-Oriented Database Management System. Figure 2 shows logical relations between the subsystems.

User Request Interpreter Subsystem. A database management system for telecommunications must support several interfaces to the database. Currently, the IN Capability Set 1 (CS-1) [Q.1218] defines Intelligent Network Application Protocol (INAP) interface between the functional entities of IN. The INAP protocol includes an interface definition to database services. In addition to INAP, at least Common Management Information Protocol (CMIP) and Common Management Information Service Element (CMISE) [Yemi93,Rama93], which are used to manage all elements of telecommunications networks, must be supported. In the future, the TINA will issue requests using a Distributed Processing Environment (DPE) like CORBA or other Object Request Brokers (ORB).

The User Request Interpreter Subsystem translates various interfaces into a common connection language used by the database management system. Darfin DBMS architecture supports several simultaneous interpreters each of which handles requests of one protocol.

Distributed Database Subsystem. A Darfin DBMS node may either be used alone or in the distributed database. The distribution management in the node is left to the Distributed Database Subsystem. The use of the subsystem frees the actual database management system from distribution control. When the database management system receives a request that it cannot resolve, it forwards it to the Distributed Database Subsystem. Similarly, when the Distributed Database Subsystem receives a request that belongs to this node, it forwards it to the database management system. This solution allows the Distributed Database Subsystem to be the only subsystem that is aware of the other nodes.

Fault-Tolerance and Recovery Subsystem. This subsystem controls communication with the mirror node and offers watchdog services for the database management node. When the node fails, the Fault-Tolerance and Recovery Subsystem informs the mirror node to take control of the node traffic. It also asks the User Request Interpreter Subsystems to forward the requests to the mirror node. Finally, it starts a recovery process in the database management system. If the whole Database Management Node fails, The Fault-Tolerance and Recovery Subsystem in the mirror node is responsible to change to role of the mirror node to the primary node.

3.3 Object-Oriented Database Management System

The Object-Oriented Database Management System (OODBMS) consists of a set of database processes that use database services to resolve requests from other subsystems and of a set of manager services that implement database functionality. The processes and manager services are shown in Figure 3.

3.3.1 Database Manager Services and Service Layers

The manager services are divided into four layers. The layers are Database Interface Layer that is the visible layer to common database processes, Global Entity Layer that is the visible interface layer to distribution and replication processing in the OODBMS, Object Layer that implements object functionality, and Real-Time Core Layer that implements low level database functionality such as real-time and physical data control.

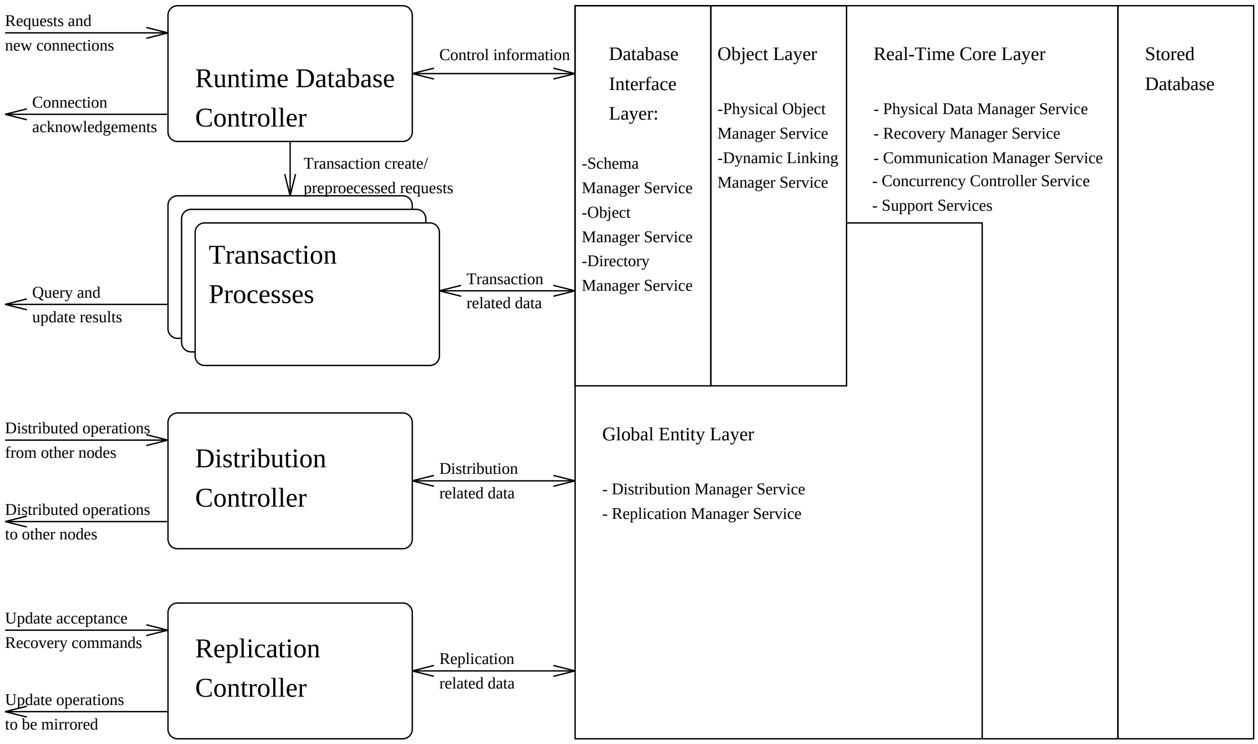

Figure 3: Processes and Database Services in Darfin OODBMS Subsystem

Database Interface Layer provides database services for the database processes. It offers schema information services for metadata use and maintenance, object information services for object oriented interface, and directory services for directory maintenance.

Object Layer offers the actual object implementation services: Physical Object Manager service and Dynamic Linking Manager Service. Managers at the Database Interface Layer use these services to implement the visible object model. At the Object Layer the objects are all alike and the object model is no longer visible.

The Physical Object Manager Service is the core manager service for objects and object operations. It offers services for object creation, object deletion, object access, object attribute access, and object operation access. It uses the Global Entity Layer when a replicated operation occurs, or when it cannot resolve an object request on the local database. It uses services of the Real-Time Core Layer to implement object functionality. The Dynamic Linking Manager Service offers services to dynamically link new object operations into the object database. It uses the services of the Real-Time Core Layer to implement object operations.

Global Entity Layer offers services for other layers and special controller processes to support distribution and replication in the OODBMS. It consists of Distribution Manager Service and Replication Manager Service. The Distribution Manager Service is used by the Distribution Controller process when it needs database services. It is also used by other services when an unresolved database operation occurs. The Replication Manager Service is used by the Replication Controller process when it needs database services. It is also used by other database services when a database update operation that need to be mirrored occurs.

Real-Time Core Layer implements the core operations of real-time database management system. The layer consists of Physical Data Manager Service, Recovery Manager Service, Communication Manager Service, Concurrency Controller Service, and Support Services. Of the services the Communication Manager Service and Support Services are used on all other layers.

The Physical Data Manager service is the only interface to the physical data in the database. It is the only manager that has knowledge of physical data locations in the database. It offers services for data storage, retrieval, and access estimates. The Recovery Manager Service is the first service to gain control when the Fault-Tolerance and Recovery Subsystem has started OODBMS recovery operations. It restarts database processes and offers services to the Replication Manager Service to re-establish data integrity and lost data.

The Communication Manager Service offers services to the other managers to maintain communication channels to the other subsystems. The Concurrency Controller Service allows transactions to run in parallel. It offers control services to other database managers and to the Real-Time Scheduler. It may also awake the Real-Time Scheduler when a conflict occurs. The Support Services are used by other manager services for common low level operations, such as dynamic memory management.

3.3.2 Database Processes

The OODBMS processes are either interacting or non-interacting. An interacting process receives requests from a database subsystem. A non-interacting process is a control process for the interacting processes. The interacting database processes in Darfin DBMS are Runtime Database Controller, Transaction Processes, Distribution Controller, and Replication Controller. The non-interacting process is the Real-Time Scheduler.

The Runtime Database Controller is the main controller of the OODBMS. It accepts new connections from Interpreter Subsystems, creates new Transaction Processes, and forwards request information to appropriate transaction processes. The Transaction Processes are the main elements of

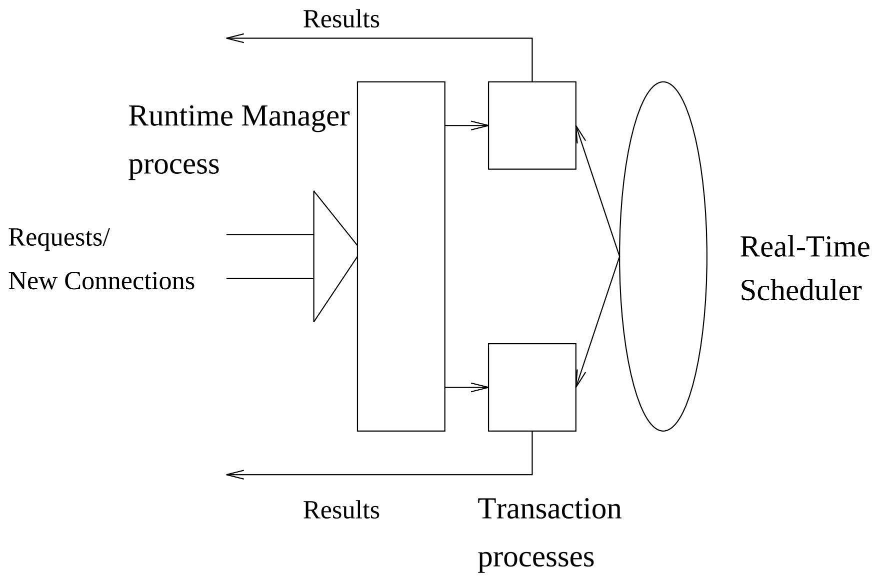

Figure 4: Process Control in Database Management Prototype

transaction processing in the OODBMS. They process the requests that the Interpreter Subsystems have sent and send results back.

The Distribution Controller is the process that maintains the connection to the Distributed Database Subsystem. It hides distribution management in the OODBMS from the other database processes. It is triggered when a transaction process initiates a database operation that involves another database node.

The Replication Controller is the process that maintains the connection to the Fault-Tolerance and Recovery Subsystem. It receives and processes recovery commands from the subsystem. It also forwards update commands to the subsystem. Replication Controller is triggered when other processes invoke database operations that need to be mirrored.

Real-Time Scheduler schedules the other database processes. Figure 4 characterizes the relations between the processes. Every database process has a priority. The Real-Time Scheduler process has the highest possible priority, i.e. it belongs to the priority group A. It executes either periodically or when a conflict in data access is detected. The Runtime Database Controller belongs to the priority group B. It executes only when requests or new connections arrive from the Interpreter Subsystems. The Distribution Controller and Replication Controller processes belong to the priority group C. They execute either when the appropriate subsystem sends a request or when a database operation triggers them. The transaction processes belong to the priority group D. In this group there are also priorities within the group that dictate the scheduling order.

4 Design and Implementation Issues in RT-OODMBS

As presented in [YWLS94] for a general real-time framework and in [TaRa96] for telecommunications context, a real-time database system involves several important design issues. When the implementation aspects are also considered, the decision spectrum is wide. In this section we discuss the major issues affecting the design and implementation of a real-time database system. Our presentation is completely restricted to database systems tailored for use in telecommunications.

Real-time systems are usually classified into three categories according to the effects of missing a transaction deadline: *hard*, *firm*, and *soft*. The deadlines raise several difficult questions. How are deadlines assigned? Are deadlines firm or soft? Currently, we believe that hard real-time transactions are not needed in telecommunications. In addition, there will be transactions that do not have a deadline at all. It must also be noted that a transaction accessing the database is only a subtask in a telecommunication service. Unfortunately *subtask deadline assignment* is still almost an unexamined problem [KaGa93]. 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. Object-orientation primarily through inheritance allows us to give database designers and application developers simple ways to express the deadlines. The implication of transactions being mainly subtasks is a need for flexible but still simple *gain*-function for soft deadline transactions. The ultimate goal of the real-time database system is to maximize the sum of gain-functions of all executed transactions.

One of the primary difficulties in designing a real-time database system is the fact that the concurrency control of data access must be combined with timing constraints [YWLS94]. Performance studies, for example [AbGa92, HaCL90, HSR+92, LiSo90], indicate that the relative performance of different 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. The implication is that the database system needs a flexible architecture in which cc-algorithm must be as autonomous module as possible so that it can be chosen according to the specific needs of actual database usage.

The major design issues in concurrency control include conflict detection and resolution as well as run policy. Several mechanisms have been proposed in the literature; see e.g. [YWLS94]. 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*). We are currently developing a hybrid method, something similar to the method of *Locking with Deferred Blocking* [YuDi93]. Our hybrid approach is based on locks. We have *weak* locks as well as normal (*strong*) locks. We have also two types of *write* locks: an *update lock* (ordinary write lock) and a *replace lock* (blind write lock). In order to support a correction criterion called τ-*serializability*

[RKMT95] our locks have timestamps. The key idea in τ-*serializability* is to allow transactions to read old values but not too old ones.

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. Our hybrid cc-algorithm will make the problem almost negligible provide that the durations of the initial optimistics phases and the completing pessimistic phases are in balance.

Scheduling Priority. *Earlist Deadline First* (EDF) is most widely used in the experimental real-time database systems as the scheduling priority. Since EDF does not work properly when the system is highly loaded [LiLa73], we decided to develop a new two-level scheduling algorithm. It will provide the flexibility needed when transactions are quite heterogeneous. On the top level we use the scheduling policy provided by the real-time platform to schedule the scheduling groups. On the lower level different scheduling policies and priority calculation schemes can be used in different scheduling groups.

Overload Management. In order to cope with the occasional periods of high load, the scheduling needs an overload management policy. Our current plan is to change the scheduling policy within the scheduling group during the overload period. However, we have not yet studied this problem in details.

IO Scheduling. When the primary copy of the database (or part of it) is on the disk, IO operations are extremely important for the real-time database systems. Traditional disk scheduling algorithms must be replaced by novel ones as proposed in [SpKS88] and in [AbGa90]. When a hybrid database is used, that is a combination of disk-based and main memory databases, the IO scheduling needs to be flexible.

Hybrid Databases. In real-time systems the unpredictability in durations of database operations is decreased when the primary copy of the database is permanently kept in the main memory. However, such a main memory database containing all information needed in telecommunications database is not economically feasible. Therefore, we will develop a hybrid database in which the primary copy of some data is in the main memory while the primary copy of the rest of data is on disks.

As presented in [GaSa92] the data can often be classified into two sets: *hot data* that are accessed frequently or that have stringent deadlined, and *cold data* that are accessed rarely and that do not have stringed deadlines. Based on the "data heat" it is possible to divide the database into two or more partitions. The primary copy of the hottest partition is permanently kept in the main memory while the primary copies of other partitions are kept on disks.

Our current plan is to have three partitions:

1) *hot data* is always in main memory,
2) *warm data* is in main memory when read or updated, and
3) *could data* is in main memory only when it is read.

For example, number translation data is hot, user profile data is either hot or warm, and statistical data is cold. It should be noticed that data items may dynamically change from one partition to another depending on hot spot.

Although we are on early stages of our data management algorithms, we believe that the simple rules based on application semantics can be found. We are working for the details of the algorithm but an efficient implementation is possible on a suitable real-time operating system. However, a hybrid database puts several requirements on the underlying platform. We need flexible and efficient tools for memory management and IO scheduling. In particular, we cannot use hardware-based virtual memory because virtual memory algorithms are not suitable to be used in database systems [KLVA93].

5 Platform Requirements

A database system fulfilling the requirements and features described in the previous sections puts various requirements on the platform on which the database system will be implemented. None of the requirements is strictly necessary in the sense that we can always implement the database system on raw hardware. However, some features supported by the underlying platform will be very convenient in the implementation. Below we discuss four areas that we consider as the most important ones: *processes and threads*, *memory management*, *io scheduling*, and *communication between nodes*. One should note that the areas are not orthogonal, i.e. sophisticated support in one area can compensate rudimentary support in another area.

Processes and Threads. We regard the presence of light-weighted threads as an essential feature to implement the flexibility needed in scheduling. Processes should be static. Their scheduling can be either round-robin with unequal time slots or pre-emptive based on priorities. In both cases the context switch must be fast.

Our current plan is to have a few, say 1–4, transaction processes. The transactions (transaction processes in Section 3.3.2) are executed in threads. The real-time operating system schedules the processes while the transaction processes schedule their own transactions. This seems to be the easiest way to obtain a suitable compromise between urgency and fairness.

The processes need communication through shared main memory. Other interprocess communication (IPC) methods are not likely to be efficient enough in order to achieve the required transaction rate.

If the creation of threads is not fast enough, then the processes must have a fixed number of threads created beforehand. This clearly restricts the dynamics of the system but is still feasible. The threads have to be scheduled as one group in the operating system level. In the group they need to be scheduled using suitable method based on deadlines, priorities or resource needs. The platform should provide some support to this kind of two level scheduling. Some kind of support for processes to schedule their threads is a clear advantage.

Memory Management. Although the RODAIN database will not use a separate buffer pool, we will have similar functionalities as in the Priority-Hints algorithm [JaCL90], Priority Memory Management (PMM) algorithm [PaCL94], and Priority Adaptation Query Resource Scheduling (PAQRS) algorithm [PaCL95]. The crucial point is to provide access to data items in the warm and cold partitions, the primary copies of which are on the disks. The Physical Data Manager Service in the Real-Time Core Layer maintains the partitions.

If the platform supports virtual memory, we must write our own replacement and fetch algorithms. In addition, locking of pages permanently into the physical main memory must be supported. As mentioned before sharing memory between processes is a mandatory feature. Protection of memory access is a useful feature that alleviates the implementation.

IO Scheduling. The RODAIN database needs an efficient I/O scheduling algorithm for managing the read requests on warm and cold database objects. Write requests on all objects can be deferred with a lower priority. In particular, the priority of updating the hot objects is the lowest one since their backup copy is sent to the mirror node.

The reads on warm and cold objects should be buffered in separate queues as presented in [CaJL89]. Therefore, the platform should support multiple I/O queues. Within each queue the requests have a priority and a deadline, which is the basic information for I/O optimization together with the current disk arm position. We must implement an elevator-like policy serving other requests while the disk head is moving to the target direction. In addition, we need platform support to implement the Shortest Seek and Earliest Deadline by Ordering (SSEDO) algorithm [CSKT92] for servicing requests on warm data objects and the Shortest Seek and Earliest Deadline by Value (SSEDV) algorithm [CSKT92] for managing large deadline queries on cold data objects.

Communication Between Nodes. The communication provided by the platform should guarantee bounded message transfer times from one node to another. This feature is necessary both for the fault tolerance and for distributed transactions. Without real-time communication features we believe that the level of fault-tolerance will be insufficient and distributed real-time transactions are impossible.

When the Replication Controller knows the maximum transfer time of messages, it is much better prepared to handle the commit of a transaction that needs updates to be mirrored in another node. An alternative is to relax the "D"-component (durability) in the ACID-property. The bounded message transfer times are also important in distributed transactions. For example, in an ATM network when the ATM Adaption layer 1 (AAL1) is used, we can use the algorithm proposed in [RaKZ95] to estimate the worst case end-to-end delay.

6 Summary

A real-time object-oriented database system needs a real-time operating system as the implementation platform. The most important properties needed in the implementation of an RT-OODBMS include shared memory between processes, light-weighted treads, fast context switch, and a real-time communication protocol. Flexible scheduling policy of processes, support for tread scheduling within the processes, and support for multiple priority-based io-queues alleviate the implementation.

Current commercial real-time operating systems, such as Chorus, Lynx, OSE, pSOS, QNX, meet the requirements in varying degree. No one of the products we have examined is superior to others for implementing an RT-OODBMS. There are also research operating systems like Chimera and Maruti which have a wider set of features but insufficient maintenance support. Therefore, we are in a difficult tradeoff situation in the RODAIN project: Should we choose a commercial product with rudimentary features or should we spend a significant amount of resources in developing a platform of our own?

Acknowledgements

This work has been carried out in the research projects Darfin (1994–5) and RODAIN (1996–). The former was funded by Telecom Finland and the latter by the Finnish Technology Development Center (TEKES) together with Nokia Telecommunications, Solid Information Technology, System Software Partners, and Telecom Finland.

References

[AbGa90] R. Abbott and H. Garcia-Molina, Scheduling I/O Requests with Deadlines: A Performance Evaluation, in *Proceedings of the 11th Real-Time Systems Symposium*, IEEE Computer Society Press, 1990, 113–24.

[AbGa92] R. Abbott and H. Garcia-Molina, Scheduling Real-Time Transactions: A Performance Evaluation, *ACM Transaction on Database Systems* **17**, 3 (Sep. 1992) 513–60.

[CaJL89] M.J. Carey, R. Jauhari, and M. Livny, Priority in DBMS Resource Scheduling, in *Proceedings of the 15th VLDB Conference*, Morgan Kaufmann, 1989, 397–410.

[Call94] M.H. Callendar, Future Public Land Mobile Telecommunications System, *IEEE Personal Communications* **1**, 4 (Dec. 1994) 18–22.

[CSKT91] S. Chen, J.A. Stankovic, J.F. Kurose, and D. Towsley, Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time Systems, *Real Time Systems* **3**, 3 (Sep. 1991) 307–36.

[DuLN93] F. Dupuy, Y. Lepetit, and B. Nicolas, Data Management in an ODP-Conformant Information Networking Architecture, in *Proceedings of TINA'93 Workshop*, Volume II, 1993, 3–19.

[DuNI95] F. Dupuy, G.C. Nilsson, and Y. Inoue, The TINA Consortium Towards Networking Telecommunications Information Services, in *Proceedings of ISS'95*, 1995.

[GaSa92] H. Garcia-Molina and K. Salem, Main Memory Database Systems: An Overview, *IEEE Transaction on Knowledge and Data Engineering* **4**, 6 (Dec. 1992) 509–16.

[HaCL90] J. Haritsa, M.J. Carey, and M. Livny, On Being Optimistic about Real-Time Constraints, in *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, ACM Press, 1990, 331–43.

[HSR+92] J. Huang, J.A Stankovic, K. Ramamritham, D. Towsley, and B. Purimetla, Priority Inheritance in Soft Real-Time Databases, *The Journal of Real-Time Systems* **4**, 2 (June 1992) 243–268.

[IEEE95] *IEEE Personal Communications*, Special Issue on The European Path Toward UMTS, **2**, 1, Feb. 1995.

[JaCL90] R. Jauhari, M.J. Carey, and M. Livny, Priority-Hints: An Algorithm for Priority-Based Buffer Management, in *Proceedings of the 16th VLDB Conference*, Morgan Kaufmann, 1990, 708–21.

[KaGa93] B. Kao and H. Garcia-Molina, Deadline Assignment in a Distributed Soft Real-Time System, in *Proceedings of the 13th Int. Conf. on Distributed Computing Systems*, IEEE Computer Society Press, 1993, 428–37.

[KLVA93] K. Krueger, D. Loftesness, A. Vahdat, and T. Anderson, Tools for the Development of Application-Specific Virtual Memory Management, in *Proceedings of OOPSLA'93*, ACM Press, 1993, 48–64.

[LiLa73] C.L. Liu and J.W. Layland, Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, *Journal of the ACM* **20**, 1 (Jan. 1973) 46–61.

[LiSo90] Y. Lin and S.H. Son, Concurrency Control in Real-Time Databases by Dynamic Adjustment of Serialization Order, in *Proceedings of the 11th Real-Time Systems Symposium*, IEEE Computer Society Press, 1990, 104–12.

[M.3010] ITU-T Recommendation M.3010: *Principles for a Telecommunications Management Network*, Aug. 1993.

[Mitt95] H. Mitts, Use of INtelligent Networks in the Universal Mobile Telecommunication System (UMTS), in *Intelligent Networks*, J. Harju, T. Karttunen and O. Martikainen, eds., Chapman & Hall, 1995, 236–45.

[MoZa95] T.J. Mowbray and R. Zahavi, *The Essential CORBA*, OMG, 1995.

[NiDC95] G. Nilsson, F. Dupuy, and M. Chapman, An Overview of the Telecommunications Information Networking Architecture, in *Proceedings of TINA'95 Conference*, Volume 1, 1995, 1–12.

[Q.1201] ITU-T Recommendation Q.1201: *Principles of Intelligent Network*, Jan. 1994.

[Q.1204] ITU-T Recommendation Q.1204: *Intelligent Network Distributed Functional Plane Architecture*, May 1993.

[Q.1211] ITU-T Recommendation Q.1211: *Introduction to Intelligent Network Capability Set 1*, Dec. 1993.

[Q.1218] ITU-T Recommendation Q.1218: *Interface Recommendation for Intelligent Network CS-1*, Jan. 1994.

[PaCL94] H.H. Pang, M.J. Carey, and M. Livny, Managing Memory for Real-Time Queries, in *Proceedings of the ACM SIGMOD'94 Conference*, ACM Press, 1994, 221–32.

[PaCL95] H.H. Pang, M.J. Carey, and M. Livny, Multiclass Query Scheduling in Real-Time Database Systems, *IEEE Transactions on Knowledge and Data Engineering* **7**, 4 (Aug. 1995) 533–51.

[Raat94] K. Raatikainen, *Information Aspects of Services and Service Features in Intelligent Network Capability Set 1*, Report C-1994-46, University of Helsinki, Department of Computer Science, Oct. 1994.

[Raat95] K. Raatikainen, Database Access in Intelligent Networks, in *Intelligent Networks*, J. Harju, T. Karttunen and O. Martikainen, eds., Chapman & Hall, 1995, 173–93.

[RaKZ95] A. Raha, S. Kamat, and W. Zhao, Guaranteeing End-to-End Deadlines in ATM Networks, in *Proceedings of the 15th International Conference on Distributed Computing Systems*, IEEE Computer Society Press, 1995, 60–8.

[Rama93] L. Raman, CMISE Functions and Services, *IEEE Communications Magazine* **31**, 5 (May 1993) 46–51.

[RKMT95] K. Raatikainen, T. Karttunen, O. Martikainen, and J. Taina, Evaluation of Database Architectures for Intelligent Networks, in *Proceedings of Telecom 95 Technology Summit*, Volume II, ITU, 1995, 549–53.

[ShRL90] L. Sha, R. Rajkumar, and J.P. Lehoczky, Priority Inheritance Protocols: An Approach to Real-Time Synchronization, *IEEE Transactions on Computers* **C-39**, 9 (Sep. 1990) 1175–85.

[SpKS88] B. Sprunt, D. Kirk, and L. Sha, Priority-Driven, Preemptive I/O Controllers for Real-Time Systems, in *Proceedings of the 15th Annual International Symposium on Computer Architecture*, IEEE Computer Society Press, 1988, 152–9.

[Tain95] J. Taina, *Requirements Analysis for Database Services in Telecommunications*, Report C-1995-17, University of Helsinki, Dept. of Computer Science, Apr. 1995.

[TaRa96] J. Taina and K. Raatikainen, Design Issues and Experimental Database Architecture for Telecommunications, to appear in *Intelligent Networks and New Services*, B. Iversen, O. Martikainen and J. Nørgraad, eds., Chapman & Hall, 1996.

[Tina95] M. Chapman and S. Montesi, *Overall Concepts and Principles of TINA*, TINA-C Deliverable TB_MDC.018_1.0_94, Feb. 1995.

[Wirt95] P.E. Wirth, Teletraffic Implications of Database Architectures in Mobile and Personal Communications, *IEEE Communications Magazine* **33**, 6 (June 1995) 54–65.

[X.701] ITU-T Recommendation X.701: *Systems Management Overview*, Oct. 1992.

[X.720] ITU-T Recommendation X.720: *Structure of Management Information: Management Information Model*, May 1993.

[X.901] ITU-T Recommendation X.901 (Draft): *Open Distributed Processing - Reference Model - Part 1: Overview*, June 1995.

[X.903] ITU-T Recommendation X.903: *Open Distributed Processing - Reference Model - Part 3: Architecture*, May 1995.

[Yemi93] Y. Yemini, The OSI Network Management Model, *IEEE Communications Magazine* **31**, 5 (May 1993) 20–29.

[YuDi93] P.S. Yu and D.M. Dias, Performance Analysis of Concurrency Control Using Locking With Deferred Blocking, *IEEE Transactions on Software Engineering* **SE-19**, 10 (Oct. 1993) 982–96.

[YWLS94] P.S. Yu, K.L. Wu, K.J. Lin, and A.H. Son, On Real-Time Databases: Concurrency Control and Scheduling, *Proceesings of the IEEE* **82**, 1 (Jan. 1994) 140–57.