Design and Architecture for Database Solutions in Future Telecommunication Services

Kimmo Raatikainen

University of Helsinki, Department of Computer Science

P.O.Box 26 (Teollisuuskatu 23), FIN-00014 University of Helsinki, Finland

1. Introduction

Databases will already in the near future have a central role in telecommunications networks. The information needed in operations and management of the nets will be collected into a logically uniform database. The world-wide nature of telecommunications prescribes that the only possibility to obtain the logical uniformity is the co-operation of autonomous databases.

During the next ten years great changes are expected in the ways of doing business on the telecommunications markets. The relative importance of traditional transmission and switching services will decrease. This is due both to liberalisation of telecommunications and to developments in the technology. We anticipate that in the future transmission and switching are done using generic technology based on international standards.

Most of the recommendations issued in the 90's by ITU-T and other standardisation bodies are object oriented. Therefore, object orientation is a natural starting point for the telecommunications database system, too. However, object-oriented databases are still emerging. They are primarily designed for situations where objects have complex structures and transactions are long-living. In telecommunications the situations are different. Most objects have quite simple structures and transactions are short. In addition, we anticipate that real-time properties, guaranteed response time in particular, are soon needed.

The paper overviews the key design issues in developing a real-time database system for telecommunications. In particular, we address the problems and solutions when the concurrency control of data access must be combined with timing constraints. The issues in concurrency control includes conflict detection, conflict resolution, and run policy. When the concurrency control is integrated with real-time scheduling we must consider issues like priority inversion, scheduling policy, overload management, and IO scheduling.

The RODAIN database architecture is a real-time, object-oriented, fault-tolerant, and distributed database management system. The RODAIN is design to fulfil the requirements of a telecommunications database systems. The requirements are derived from the most important telecommunications standards including Intelligent Network (IN), Telecommunications Management Network (TMN), and Telecommunication Information Networking Architecture (TINA). The requirements of the telecommunications database architectures originate in the following areas: real-time access to data, fault tolerance, distribution, object orientation, efficiency, flexibility, multiple interfaces, and compatibility.

The most challenging issue in designing a real-time transaction processing system for telecommunications is the handling of transactions in three categories having very different characteristics. A telecommunications database system should be able to support short but voluminous simple read transactions, long but voluminous simple updating transactions, and a few very long complex updating transactions in the same real-time database system. The RODAIN Database is designed to meet those constraints. In RODAIN the object-oriented approach was chosen because a special purpose object model for real-time transactions can be used to provide the information needed in the concurrency control and real-time scheduling of heterogeneous transactions.

2. Use of Databases in Telecommunications Systems

In telecommunications there are currently several architectures that need database services. In this section we briefly introduce four of them: 1) Intelligent Network (IN), 2) Telecommunications Management Network (TMN), 3) Telecommunications Information Networking Architecture (TINA), and 4) the third generation mobile systems (UMTS and FPLMTS/IMT-2000). We also discuss the key database requirements.

2.1 Intelligent Network

The reference model of Intelligent Network (IN) has four planes of which the Distributed Functional Plane (DFP) is the most important one in database design. The DFP defines the functional entities that constitute the IN functional architecture. The functional entities and their relationships that are of interest in the database design are: 1) the Service Control Function (SCF) is responsible of IN service control; 2) the Service Data Function (SDF) is responsible of IN data management; and 3) the Service Management Function (SMF) is responsible of IN management functions. [KaHM95]

The SDF is responsible for real-time access and maintenance of global network related data. IT is a data server for SCFs that provides the functionality needed for storing, managing, and accessing information. Thus it offers regular data management services but not necessarily database management services. The following five basic types of database operations needed in the IN Capability Set 1 (CS-1) have been identified [Raat94]:

1) ***Retrieval of structured objects*** from persistent subscriber data objects. Some of the retrievals trigger a later update. An example is the Freephone Service with option to provide routing statistics to the subscriber.
2) ***User management actions*** that modify persistent subscriber data objects. These actions must be protected against unauthorised use. Some updates may also have to be verified against other criteria. For example, the subscriber must have the possibility to cancel or prohibit call forwarding to his or her phone number.

3) ***Verification of Personal Identification Number*** (PIN). For security reasons the PIN verification must be done in the database system when services are made global. 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 serialised. A special aspect of these updates is that they can be done in large blocks.

It should also be noted that IN CS-1 defines the data distribution to be hidden from SCF. This implies that every database providing the functionality of the IN SDF is a member of a large distributed database. Detailed description of database access in Intelligent Networks can be found in [Raat95].

2.2 Telecommunications Management Network

TMN or Telecommunications Management Network (ITU-T Recommendations in M.3000 Series) is a generic architecture to be used for all kinds of management services. TMN is based on the principles of the OSI Management (ITU-T Recommendations in X.700 Series).

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*. 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 organised as a hierarchical database tree. In brief, the principles of OSI Management (and TMN) require that the database system contains the functionality of an OSI Management Agent.

2.3 Telecommunication Information Networking Architecture

TINA-C, a world wide consortium developing the Telecommunication Information Networking Architecture (TINA), has the goal to define and to validate an open architecture for future telecommunications services. 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), Telecommunications Management Networks (TMN), Asynchronous Transfer Mode (ATM), and Common Object Request Broker Architecture (CORBA). [NiDC95]

The focus in development of TINA architecture has been on information and computational viewpoints. Therefore, the database requirements are not (at least in public) examined in details. The distribution in TINA is based on ODP (ITU-T

Recommendations in X.900 Series) and CORBA, both of which relies on object-orientation. Therefore, TINA needs a storage for persistent objects, that is an object-oriented database supporting ODP transparencies. Moreover, when telecommunication services based on the TINA architecture are in operation, the access to persistent objects must be fast. In the current public TINA-C documentation real-time is not mentioned but the inclusion of IN functionality implies that also TINA requires real-time features. As an external interface the database system must support the object invocation mechanisms of CORBA and communication through an ODP channel. The ODP interface requirements of database management systems are examined in [DuLN93].

2.4 Third Generation Mobile Networks

Today, the specification of the third generation mobile networks is very active. These networks will rely on IN services supporting multimedia applications. Such systems are the Future Public Land Mobile Telecommunication Systems (FPLMTS) [Call94] and Universal Mobile Telecommunication System (UMTS) [UMTS95].

The mobility specific control functions that UMTS needs are planned to be available in the IN Capability Set 3. Already IN Capability Set 2 will support some areas required in UMTS. While IN CS-1 services use mostly read operations on IN databases, UMTS functions also frequently update the database. Access delay to the data is of crucial importance. The data required to support a certain user might not be stored near the location of where it is used [Mitt95].

It is anticipated that future mobile systems, such as UMTS, will generate 7-8 thousand real-time query transactions per second in a regional calling area and 7-8 thousand real-time updates per second in a regional calling area. In addition, the performance and reliability requirements are stringent. For 96 % of queries the response time should be 150 ms or less. The response time of updates should be 1-3 seconds. The availability should be 7 days a week and 24 hours a day and practically no downtime at all. [Wirt95]

The main functions of UMTS database services are:

1) the basic functions that provide read and write access,
2) to locate the requested data within the database,
3) to move data to locations close to the user when the user has roamed in order to provide fast access in locations where data is frequently used,
4) to provide mechanisms that ensure the integrity and reliability even when data is copied between locations and possibly duplicated, and
5) to ensure that data is only accessed by entities that have been authorised to do so.

2.5 Key Database Requirements

If a database only supports CS-1 service logic programs, a good choice would be a distributed relational database that is fault tolerant and that is primarily tailored for reads and writing sequential log records. However, the future trends give new functional requirements. Below we summarise the key requirements ***including data distribution***, ***data replication***, ***object orientation***, ***object directories***, ***multiple***

application interfaces, ***fault tolerance***, and ***real-time transactions***.

Data distribution. A set of database nodes may co-operate to accomplish database tasks. A node may request information from other nodes in case information needed is not locally available. It is possible to implement the underlying database system without data distribution. In such an implementation all applications use a single database server. This differentiates logical data distribution from physical data distribution among database nodes. The former is necessary but the latter is a decision internal to the design of the database architecture. Our current belief is that only a few requests need access more than one database node.

Data replication. It is stated in ITU-T Recommendation in Q.1204 that a database node contains customer and network data for real time access. We believe that currently most distributed operations are reads. Therefore, replication is an effective way to speed up distributed operations because it gives better throughput.

Object orientation. It is a common belief that the best long term telecommunications architectures are object oriented. The implication is that a database system must support object access. In other words, the database system must either be object oriented or have an object interface.

Object directories. The conceptual model of IN Service Control Function (ITU-T Recommendation Q.1214) defines a Service Data Object Directory that is used to access data stored in the SDFs. This implies that object directories must be supported. The invocation mechanism of CORBA is a good alternative to implement the functionality.

Multiple application interfaces. In telecommunications different architectures define different access interfaces. The IN Capability Set 1 defines IN Application Protocol (INAP). TMN has its own access methods based on the OSI (X.700) management protocols. The current definition in TINA is based on TINA ODL which is quite similar to the OMG IDL interface. Therefore, OMG IDL interface is also needed. In addition, OQL interface—probably without response-time guarantees—would be convenient for ad hoc queries and database maintenance.

Fault tolerance. Real time access implies that data must be continuously available. The result of the implications is that the database system must be fault tolerant. In the current definitions the maximum allowed down time is a few seconds a year.

Real-time transactions. Although the real time data access as stated in ITU-T Recommendation Q.1204 does not directly imply that the underlying database system must support real time transactions, we believe that the most convenient way to support real time access to data is to use a real-time database system.

In telecommunications we will need both soft and firm transactions. We do not believe that hard transactions will be used in near future because systems supporting hard transactions are too expensive for open telecommunication markets. Detailed discussion can be found in [TaRa96].

3. Real-Time Databases

Recommendation Q.1204 (***Intelligent Network Distributed Functional Plane Architecture***) 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***) 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

Real-Time Database System (RT-DBS) [HaCL90] 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 maximise the fraction of transactions which meet their deadlines. This is in contrast to the goal of traditional database systems (DBMSs) which is to minimise 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. [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.

Since the main goal of a firm or soft RT-DBS is to maximise 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 a 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) [EGLT76]. This is due to the fact that under most operating circumstances locking algorithms give shorter response times than optimistic algorithms; see e.g. [AgCL87].

Recent performance studies ([AbG92], [HaCL90], [HSR+92], and [LiSo90], 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 *serialisation 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 penalised through appropriate actions selected by the conflict resolution mechanism. The actions most commonly used are ***blocking*** (wait), ***abort*** (restart), ***multiversioning***, and ***dynamically readjusting the serialisation 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* [TaSG85], *wound-wait* [RoSL78], *wait-die* [RoSL78], *running priority* [FrRo85], *wait depth limited* [FrRT92], and *delayed abort* [Yu90]. In addition to the pure *optimistic concurrency control* (OCC) *broadcast OCC* [KuRo81] and a combination of pure and broadcast OCC [YuDL93] have been proposed. *Locking with deferred blocking* [YuDi93] is a scheme that combines 2PL and OCC. Multiversioning opens a wide variety of conflict resolution mechanisms. A detailed comparison can be found in [WuYC93].

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* [YuDi92]. 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* [ShRL90] the low-priority transaction inherits the priority of the high-priority transaction. In the *priority ceiling protocol* [SRSC91] blocking time of the high-priority transaction is bounded. The *dynamic priority ceiling protocol* [ChLi90] is a modification of the previous based on earliest deadline first instead of fixed external priorities.

Scheduling Priority

Earliest Deadline First (EDF) is most widely used in the experimental real-time database systems as the scheduling priority. It can minimise the number of late transactions expect when the system is highly loaded [LiLa73]. 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. [AbGa92], [BMHD89], [SRSC91], [HSTR89], [HSRT91]. The fundamental design criteria are the heterogeneity 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 [HaLC91] an *adaptive earliest 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 [PaLC92] an *adaptive earliest 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 [SpKS88] and in [AbGa90]. 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. RODAIN Database Architecture

The RODAIN database architecture is a real-time, object-oriented, distributed, and fault-tolerant architecture of a database management system designed for telecommunications. Below we briefly summarise the essentials of the architecture while a detailed description can be found in [TaRa96a-c].

A RODAIN Database consists of a set of autonomous RODAIN Database Nodes that interact with each other. Each database node may communicate with one or more applications, and an application may communicate with one or more database nodes. A RODAIN Database Node consists of Database Primary Node, Database Mirror Node, and Reliable Secondary Storage Subsystem (Figure 1). The primary and mirror node are identical having a set of subsystems:

- User Request Interpreter Subsystem (URIS),
- Distributed Database Subsystem (DDBS),
- Fault-Tolerance and Recovery Subsystem (FTRS),

- Watchdog Subsystem (WDS), and
- Object-Oriented Database Management Subsystem (OO-DBMS).

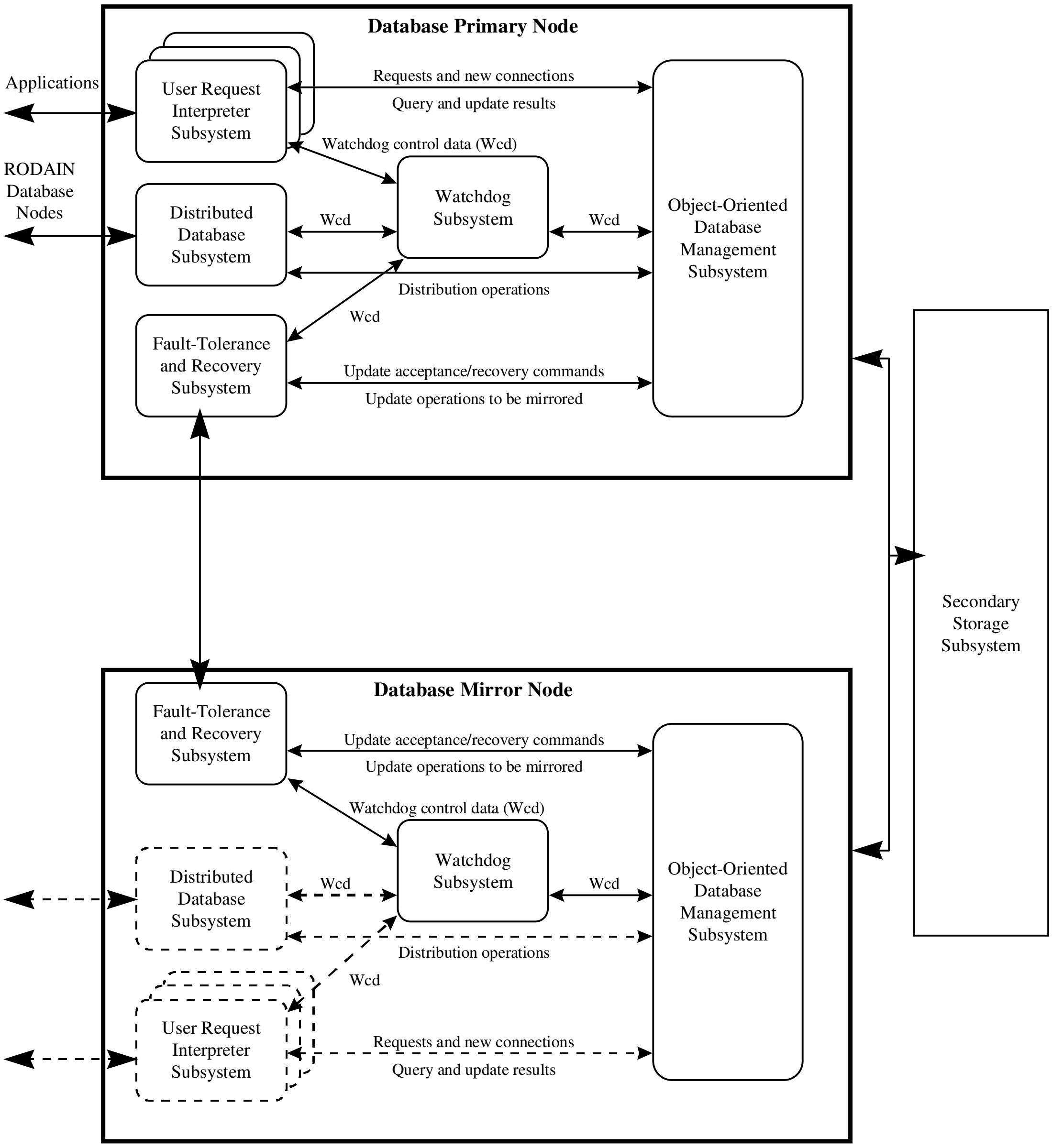

Figure 1: RODAIN database node.

4.1 User Request Interpreter Subsystem

A database management system to be used in telecommunications must support several interfaces to the database including CS-1/CS-2 INAP, CMIP, CORBA, TINA. User Request Interpreter Subsystems translate various interfaces into a common connection language. Each URIS takes care of one specific interface. The URIS on the Primary Node is active. On the Mirror Node the URIS is passive.

4.2 Distributed Database Subsystem

A RODAIN Database Node may either be used as a stand-alone system or in co-operation with the other autonomous RODAIN Database Nodes. The database co-operation management in the Database Primary Node is left to the Distributed Database Subsystem. The Distributed Database Subsystem on Mirror Node is passive.

4.3 Fault-Tolerance and Recovery Subsystem

The subsystem controls communication between the Database Primary Node and the Database Mirror Node. It also co-operates with the Watchdog Subsystem to support fault-tolerance.

The FTRS on Primary Node handles transaction logs and recovery data. It takes care of saving transaction logs either into the Mirror Node or into the Secondary Storage Subsystem. Since the database only contains committed data, the FTRS only needs to handle redo logs.

The FTRS on the Mirror Node receives the logs from the FTRS on the Primary Node. It saves the logs into the Secondary Storage Subsystem and sends the corresponding update instructions to the OO-DBMS on the Mirror Node. The FTRS on the Mirror Node also receives the recovery data from the Primary Node and passes it to the local OO-DBMS.

4.4 Watchdog Subsystem

The subsystem watches over the other running subsystems locally on both Primary and Mirror Node. It communicates with the other subsystems to state the current node status. Upon a failure it restores the node. The Watchdog Subsystem notices when a system is down and starts a recovery process.

4.5 Object-Oriented Database Management Subsystem

The OO-DBMS is the main component of a RODAIN Database Node both on Primary Node and on Mirror Node. It maintains databases, real-time constraints, integrity, and concurrency control. It offers object storing, querying, and transaction services for the URISes and the DDBS. The OO-DBMS 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 Object-Oriented Database Management Subsystem (OO-DBMS) implements the functionality of the RODAIN database. It offers object storing, querying, and transaction services for User Request Interpreter Subsystem, which communicates with an application outside RODAIN Database Node.

Transactions are executed in the Database Primary Node. The Database Mirror Node does not accept incoming transactions. When running as the Database Primary Node the node is either in the normal primary mode or in the transient mode. The key difference between the two modes is in the way how the transaction log is handled.

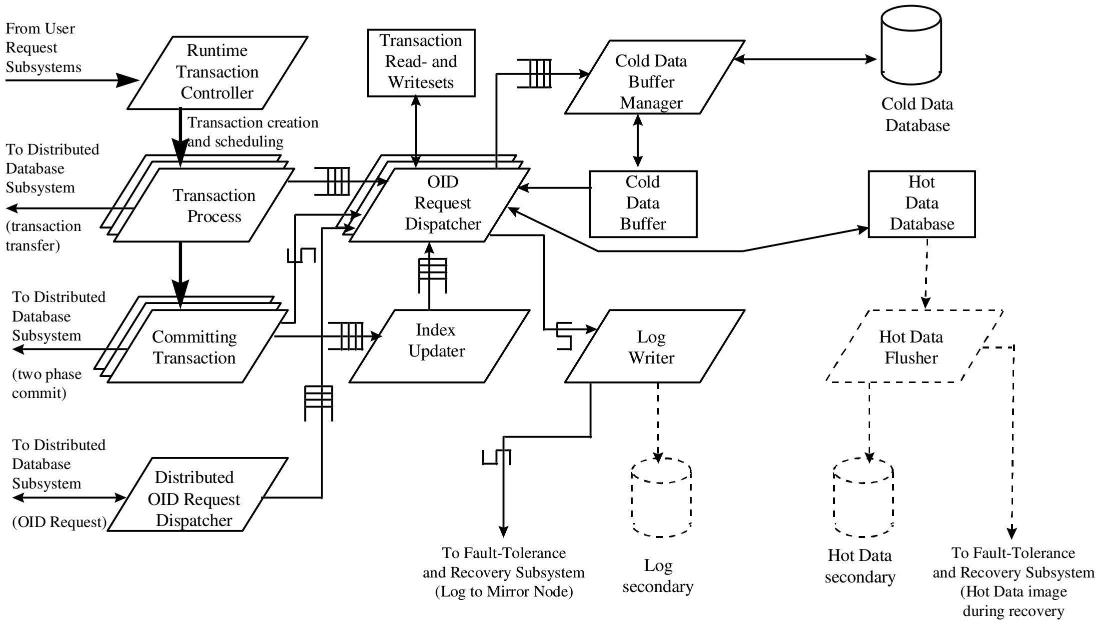

Figure 2 Processes in RODAIN Database Primary Node.

Processes in a RODAIN Node are induced from different sources. The sources include operating system, network communication system, database processes, and executing transactions. Processes induced by the RODAIN Database system are divided into priority levels. Database services other than transactions have usually a fixed priority. The Transaction Processes have a priority area, in which their priorities can vary. The priority of a Transaction Process depends on the deadline, importance, and other properties of the transaction. The priority can (and usually does) change during the runtime of the transaction.

The processes in the Database Primary Node are depicted in Figure 2 using a formalism of an extended DARTS software design method [Goma84] for real-time system. Below we briefly summarise the basic concepts of processes that are relevant when transactions are processed.

4.5.1 Runtime Transaction Controller

The Runtime Transaction Controller (RTC) accepts new transaction requests from applications. The RTC allocates one Transaction Process from the pool of Transaction Processes to serve the incoming transaction. Based on the attribute values of the transaction instance the RTC assigns a priority to the Transaction Process. The Runtime Transaction Controller can deny an incoming transaction request in an overload situation.

The Runtime Transaction Controller is also responsible for participating in transaction scheduling. Transaction scheduling is done by modifying the priority of each Transaction Process. The RTC adjusts the priorities of each transaction based on the selected scheduling policy. When a transaction is aborted and restarted due to concurrency control policy, the transaction is restarted in the same Transaction Process.

The Runtime Transaction Controller also handles transactions that have missed their deadlines. According to the over-deadline handling scheme, which is a mandatory attribute of any real-time transaction in the RODAIN data model [KiRa96], the RTC either aborts the transaction or lets it run with a lowered priority and importance. In both cases, the originator of the transaction is notified about the missed deadline.

4.5.2 Transaction Processes

A Transaction Process is invoked to handle requests sent by an application. A request is either a request to access an object or an invocation of method in an object (method call). Transaction Processes are permanent processes that are allocated to execute transactions. When a transaction is completed, the Transaction Process that carried out the execution of the transaction is returned into the pool of free Transaction Processes. When a transaction is aborted and restarted due to concurrency control, the same Transaction Process is used.

A Transaction Process offers the functionality specified in the RODAIN data model as well as query optimisation and usage of indices in queries. The object methods are executed in the memory space of the Transaction Process.

4.5.3 OID Request Dispatchers

An OID Request Dispatcher is a process that receives OID read and write requests from Transaction Processes and executes them. When a Transaction Process performs a request to access an object, the request is placed into the request queue. One OID Request Dispatcher process gets the request from the queue. When a request is completed, a result message is sent back through a buffered communication channel. Therefore, the requesting Transaction Process can decide, whether it waits for the completion of one request or whether it sends multiple requests into the queue before waiting a completion.

An OID Request Dispatcher offers services for reading and (pre)writing an object, and for validating and committing a transaction. A data accessing method depends on whether the requested object belongs to the hot data or to the cold data. When the accessed object is in the hot data, the OID Request Dispatcher computes its direct physical address in the main memory database and performs the operation. When the accessed object is in the cold data, the OID Request Dispatcher first tries to access data in the cold data buffer. If the object is not in the buffer, the request is forwarded to the Cold Data Buffer Manager. When an object is accessed, the readset or writeset of the requesting transaction is updated.

All OID Request Dispatcher processes are installed during the startup of the database. The number of processes is specified as an initialisation parameter of the RODAIN Database.

4.5.4 Committing Transactions

A Committing Transaction is a Transaction Process that has entered the commit phase. The Committing Transactions have the same functionality as Transaction Processes but each Committing Transaction has a higher priority than any Transaction Process. A

Committing Transaction performs the validation of the transaction commit using the readset and the writeset of the transaction. If the validation is successful, all modified objects are written into the database and the transaction is committed. If the concurrency control detects a conflict, at least one transaction is aborted and restarted.

4.5.5 Cold Data Buffer Manager

The Cold Data Buffer Manager (CDBM) receives read and write requests to the cold data from the OID Request Dispatcher. Since a read request sent to the CDBM can not be resolved from the Cold Data Buffer, the request is resolved from the Secondary Storage Subsystem. A write request causes the written object to be pinned into the Cold Data Buffer. In the case of transaction commit, the modified objects are written into the Secondary Storage Subsystem and unpinned before the commit is accepted.

When the Cold Data Buffer is full, the CDBM makes free space by removing some objects from the buffer. The pinned objects are the last ones to be removed, since they need to be written into a temporary disk file. When ever possible the objects to be removed are selected from the set of unpinned objects. A priority-LRU method [CaJL98] is used in the selection.

4.5.6 Log Writer

The Log Writer handles log write commands. When the Database Primary Node is in the normal primary mode, the requests are passed to the Mirror Node. When the Database Primary Node is in the transient mode, the log write requests are forwarded to the Secondary Storage Subsystem. In both cases the write process is synchronous. Thus, a log write operation is finished when it is guaranteed that the entry is permanently stored either into the Mirror Node or into the Secondary Storage Subsystem.

4.5.7 Hot Data Flusher

The Hot Data Flusher writes the contents of the main memory database into the Secondary Storage Subsystem. This process is used in the Primary Node only when the node is in the transient mode.

5. References

[AbGa90] Abbott, R. and Garcia-Molina, H., "Scheduling I/O Requests with Deadlines: A Performance Evaluation," in Proc. of the 11th Real-Time Systems Symposium, 1990, pp. 113-124.

[AbGa92] Abbott, R. and Garcia-Molina, H., "Scheduling Real-Time Transactions: A Performance Evaluation," ACM Transaction on Database Systems 17, 3, Sep 1992, pp. 513-560.

[AgCL87] Agrawal, R., Carey, M. J., and Livny, M., "Concurrency Control Performance Modeling: Alternatives and Implications," ACM Transaction on Database Systems 12, 4, Dec 1987, pp. 609-654.

[BMHD89] Buchmann, A. P., McCarthy, D. R., Hsu, M., and Dayal, U., "Time-Critical Database Scheduling: A Framework For Integrating Real-Time Scheduling and Concurrency Control," in Proc. of the 5th International Conference on Data Engineering, 1989, pp. 470-480.

[CaJL89] Carey, M.J., Jauhari, R. and Livny, M., "Priority in DBMS Resource Scheduling," in Proc. of the 15th Very Large DataBase Conference, 1989, pp.397-410.

[Call94] Callendar, M. H., "Future Public Land Mobile Telecommunications System," IEEE Personal Communications 1, 4, Dec 1994, pp. 18-22.

[ChLi90] Chen, M.-I. and Lin, K.-J., "Dynamic Priority Ceilings: A Concurrency Control Protocol for Real-Time Systems," The Journal of Real-Time Systems 2, 4, Dec 1990, pp. 325-346.

[DuLN93] Dupuy, F., Lepetit, Y., and Nicolas, B., "Data Management in an ODP-conformant Information Networking Architecture," in Proc. Of TINA 93 Workshop, 1993, Vol. 2, pp. 3-19.

[EGLT76] Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L., "The Notions of Consistency and Predicate Locks in a Database System," Communications of the ACM 19, 11, Nov 1976, pp. 624-633.

[FrRo85] Franaszek, P. A. and Robinson, J. T., "Limitations of Concurrency in Transaction Processing," ACM Transactions on Database Systems 10, 1, Mar 1985, pp. 1-28.

[FrRT92] Franaszek, P. A., Robinson, J. T., and Thomasian, A., "Concurrency Control for High Contention Environments," ACM Transactions on Database Systems 17, 2, Jun 1992, pp. 304-345.

[Goma84] Gomaa, H., "A Software Design Methods for Real-Time Systems," Communications of the ACM, 27, 9, Sep 1984, pp. 938-949.

[HaCL90] Haritsa, J. R., Carey, M. J., and Livny, M., "On Being Optimistic about Real-Time Constraints," in Proc. of the 9th ACM Symposium on Principles of Database Systems, 1990, pp. 331-343.

[HaLC91] Haritsa, J. R., Livny, M., and Carey, M. J., "Earliest Deadline Scheduling for Real-Time Database Systems," in Proc. of the 12th Real-Time Symposium, 1991, pp. 232-242.

[HSRT91] Huang, J., Stankovic, J. A., Ramamritham, K., and Towsley, D., "Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes," in Proc. of the 17th VLDB Conference, 1991, pp. 35-46.

[HSR+92] Huang, J., Stankovic, J. A., Ramamritham, K., Towsley, D., and Purimetla, B., "Priority Inheritance in Soft Real-Time Databases," The

Journal of Real-Time Systems 4, 2, Jun 1992, pp. 243-268.

[HSTR89] Huang, J., Stankovic, J. A., Towsley, D., and Ramamritham, K., "Experimental Evaluation of Real-Time Transaction Processing," in Proc. of the 10th Real-Time Systems Symposium, 1989, pp. 144-153.

[KaGa93] Kao, B. and Garcia-Molina, H., "Deadline Assignment in a Distributed Soft Real-Time System," in Proceedings of the 13th Int. Conf. on Distributed Computing Systems, 1993, pp. 428-437.

[KaHO95] Karttunen, T., Harju, O., and Martikainen, O., "Introduction to Intelligent Networks," in Intelligent Networks, Harju, O., Karttunen, T., and Martikainen, O., eds., Chapman & Hall, 1995, pp. 1-33.

[KiRa96] Kiviniemi, J. and Raatikainen, K., "Object-Oriented Data Model for Telecommunications," Technical Report C-1996-75, Department of Computer Science, University of Helsinki, Finland, 1996.

[KuRo81] Kung, H. T. and Robinson, J. T., "On Optimistic Methods for Concurrency Control," ACM Transaction on Database Systems 6, 2, Jun 1981, pp. 213-226.

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

[LiSo90] Lin, K.-J. and Son, S. H., "Concurrency Control in Real-Time Databases by Dynamic Adjustment of Serialization Order," in Proc. of the 11th Real-Time Systems Symposium, 1990, pp. 104-112.

[Mitt95] Mitts, H., "Use of INtelligent networks in the Universal Mobile Telecommunication System (UMTS)," in Intelligent Networks, Harju, O., Karttunen, T., and Martikainen, O. (eds), Chapman & Hall, 1995, pp. 236-245.

[NiDC95] Nilsson, G., Dupuy, F., and Chapman, M., "An Overview of the Telecommunications Information Networking Architecture," in Proc. of TINA 95 Conference, 1995, Vol. 1, pp. 1-11.

[OMG91] OMG, "The Common Object Request Broker: Architecture and Specification," Object Management Group, 1991.

[PaLC92] Pang, H., Livny, M., and Carey, M. J., "Transaction Scheduling in Mulriclass Real-Time Database Systems," in Proc. of the 13th Real-Time Systems Symposium, 1992, pp. 23-34.

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

[Raat95] Raatikainen, K., "Database Access in Intelligent Networks," in Intelligent Networks, Harju, O., Karttunen, T., and Martikainen, O. (eds), Chapman & Hall, 1995, pp. 173-193.

[RaTa96] Raatikainen, K. and Taina, J., "Design Issues in Database Systems for Telecommunication Services," in Proc. of IFIP IN'95 Working Conference, 1995, pp.71-81.

[RoSL78] Rosenkrantz, D. J., Stearns, R. E., and Lewis II, P. M., "System Level Concurrency Control for Distributed Database Systems," ACM Transactions on Database Systems 3, 2, Jun 1978, pp. 178-198.

[ShRL90] Sha, L., Rajkumar, R., and Lehoczky, J. P., "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," IEEE Transactions on Computers C-39, 9, Sep 1990, pp. 1175-1185.

[SpKS88] Sprunt, B., Kirj, D., and Sha, L., "Priority-Driven, Preemptive I/O Controllers for Real-Time Systems," in Proc. of Int. Symp. on Computer Architecture, 1988, pp. 152-159.

[SRSC91] Sha, L., Rajkumar, R., Son, S. H., and Chang, C.-H., "A Real-Time Locking Protocol," IEEE Transactions on Computers 40, 7, Jan 1991, pp. 793-800.

[TaRa96a] Taina, J. and Raatikainen, K., "Design Issues and Experimental Database Architecture for Telecommunications," in Intelligent Networks and New Technologies, ed. J. Nørgaard and V.B. Iversen, pp. 121–39, Chapman & Hall, London, 1996.

[TaRa96b] Taina, J. and Raatikainen, K., "Experimental real-time object-oriented database architecture for intelligent networks," Journal of Engineering Intelligent Systems, 4, 3, Sep 1996, pp. 57–63.

[TaRa96c] Taina, J. and Raatikainen, K., "RODAIN: A Real-Time Object-Oriented Database System for Telecommunications, in Proc. of the DART'96 Workshop, 1996, pp. 12–15.

[TaSG85] Tay, Y. C., Suri, R., and Goodman, N., "A Mean Value Performance Model for Locking in Databases: The No-Waiting Case," Journal of the ACM 32, 3, Jul 1985, pp. 618-651.

[UMTS95] Special Issue on "The European Path Towards UMTS," IEEE Personal Communications 2, 1, Feb 1995.

[Wirt95] Wirth, P. E., "Teletraffic Implications of Database Architectures in Mobile and Personal Communications," IEEE Communications Magazine 33, 6, Jun 1995, pp. 54-65.

[WuYC93] Wu, K.-L., Yu, P. S., and Chen, M.-S., "Dynamic Finite Versioning: An Effective Versioning Approach to Concurrent Transaction and

Query Processing," in Proc. of the 9th Int. Conf. on Data Engineering, 1993, pp. 577-586.

[Yu90] Yu, P. S., "Effective Concurrency Control With No False Killing," IBM Tech. Disclosure Bull. 33}, 2, Jul 1990, pp. 193-194.

[YuDi92] Yu, P. S. and Dias, D. M., "Analysis of Hybrid Concurrency Control for a High Data Contention Environment," IEEE Transactions on Software Engineering SE-18, 2, Feb 1992, pp. 118-129.

[YuDi93] Yu, P. S. and Dias, D. M., "Performance Analysis of Concurrency Control Using Locking With Deferred Blocking," IEEE Transactions on Software Engineering SE-19, 10, Oct 1993, pp. 982-996.

[YuDL93] Yu, P. S., Dias, D. M., and Lavenberg, S. S., "On the Analytical Modeling of Database Concurrency Control," Journal of the ACM 40, 4, Sep 1993, pp. 831-872.

[YWLS94] Yu, P. S., Wu, K.-L., Lin, K.-J., and Son, S. H., "On Real-Time Databases: Concurrency Control and Scheduling," Proceedings of the IEEE 82, 1, Jan 1994, pp. 140-157.