Benchmark for Real-Time Database Systems for Telecommunications

Jan Lindström and Tiina Niklander

University of Helsinki, Department of Computer Science
P.O. Box 26, FIN-00014 UNIVERSITY OF HELSINKI, Finland
{jan.lindstrom,tiina.niklander}@cs.helsinki.fi

Abstract. As long as there have been databases there has been a large interest in measuring their performance. Therefore, several different benchmarks have been proposed. However, previous proposals do not consider timely response and other telecommunication application requirements. This paper shortly reviews telecommunication requirements and previous work on real-time databases. These requirements are used to prepare the benchmark proposal for real-time database systems in telecommunication. The benchmark models a hypothetical telecommunication operator, which provides some services to subscribers. The services provided are selected to represent a wide range of real services from different application needs e.g. Intelligent Networks, 800 service and GSM user roaming.

1 Introduction

The requirements of the telecommunications database architectures originate in the following areas [8, 10]: real-time access to data, fault tolerance, distribution, object orientation, efficiency, flexibility, multiple interfaces, security and compatibility [1, 11, 15]. In summary, Network Services Databases is a soft/firm real-time system that contains rich data and transaction semantics, which can be exploited to design better methods for concurrency control, recovery, and scheduling. It has data with varying consistency criteria, recovery criteria, access patterns, and durability needs. This can potentially lead to development of various consistency and correctness criteria that will improve the performance and predictability of such systems.

The traditional database benchmarks, such as Wisconsin, TPC-A, TPC-B, and TPC-C (see [4]), do not consider the service time as correctness criteria. The response time or residence time is only used to describe the database performance. The response time is an important correctness criterion of a real-time database, where the client expects to receive the response within a given time limit.

Previous work on real-time databases in general has been based on simulation. However, several prototypes of general-purpose real-time databases have been introduced. The StarBase real-time database architecture [9] based on relational model has been developed over a real-time microkernel operating system. The performance of the StarBase database has been evaluated in [14]. RTSORAC [19] is a prototype of object-oriented real-time database architecture. It is based on open

OODB architecture with real-time extensions and implemented over a thread-based POSIX-compliant operating system. Another object-oriented architecture called M^2RTSS [3] is a main-memory database system with classes that implement the core functionality of storage manager, real-time transaction scheduling, and recovery.

In this paper we present a benchmark that can be used to evaluate database performance for telecommunication services. The model covers most important features of the Intelligent Network concept and GSM roaming. These indicate the need for timely response and access to distributed data [17]. The services are modelled using simple transactions that represent the services. The benchmark also uses a workload model from telecommunication. In addition to the measurement of timely response, the database and service availability measurement is presented.

2 Telecommunication Requirements

The telecommunication field has different services, which have different database needs. The intelligent network (IN) concept models its services in a traditional fixed-line network. The GSM services are wireless. Telecommunication Management Network (TMN) has its own requirements for the databases. This section concentrates on the requirements for distributed databases. Requirements for separate databases are listed in [10, 12, 16].

The Intelligent Network (IN) [6] services do not necessarily require the support from a distributed database. A centralized database is enough to support them. The caller and called profiles can be fetched separately from their own databases at both ends of the call. Even the 800 Service does not require co-operation of multiple database nodes [7]. According to [10], the intelligent network has databases for traffic data, service data and customer data. The requirements analysis shows that transactions in the IN system are commonly small (only a few operations per transaction) and require short response time. This creates problems for most commercial database systems.

The best-known IN architecture is probably the Datacycle architecture by AT&T [2]. It is based on special hardware that allows fast data access. A regular type IN database has been presented by Norwegian Telecom Research [5, 18]. Their architecture is a shared-nothing parallel database where parallel relational database nodes communicate with each other via an ATM network. The traditional IN services can be implemented without transactions accessing multiple databases.

In wireless communication, like GSM or UMTS, roaming is an important basic service. Because the user may roam between different service providers, the service providers have a home location register for their own users and a visitor location register for the users currently in their network. A visiting user's move to a remote service provider requires an update on multiple places. These updates must be done in one atomic transaction to avoid losing the user's current location. The user is removed from the visitor location register at the departure service provider. It is added to the visitor location register at the arrival service provider. This change is also updated on the user’s home location register. The authentication and roaming agreements are omitted from this paper, but a description can be found in [13]. The requirement

analysis shows that mobile telecom transactions are commonly small and require short response time [12].

A Telecommunication Management Network may also need a distributed update operation, when a new link is established between two switches. At the minimum, this new link must be updated on routing information on the two participating switches. This update should be done atomically, to allow traffic passing over it. Of course, the update can be done sequentially but then some traffic going in one direction may not be able to return, until the update is performed on the other end also.

In a fixed network a conference call could need a distributed transaction to access the database on multiple service providers. Each participant of a conference call is using different operator. This requires an update to subscription information on the client databases of all operators used.

In televoting the distribution need can also exist. This scenario assumes that the vote collector is not a telecom operator. Each voter is allowed to vote only once. To reduce the load on the telecommunication network the vote collector and the telecom operator co-operate. A user registers his wish to vote on a telecom operator, which dispatches this wish to a vote collector. Then the user makes the vote, which is registered at the vote collector. The information that the user has voted is also registered to the operator. The operator can now prevent this user from voting several times without loading the vote collector and the communication lines.

Previous work has proposed that consistency of some subsets of the telecommunication transactions can be relaxed [10]. However, we recommend using full consistency on all transactions in the telecommunication database because it is easier to support the full consistency with all transactions than the full consistency with a subset of transactions and a relaxed consistency with another subset. Relaxed consistency would require additional semantic information of the transactions. This information can originate only from application developer. Application developer would need to include additional code to applications to maintain database consistency when integrity constraints are present. This would complicate and lengthen the development time. Therefore, the proposed benchmark requires full consistency with all transactions.

Compared to the traditional benchmarking of OLAP applications (TPC-B, TPC-C), which originate from banking applications and business applications, telecommunication databases contain much less data. This means that telecommunication databases fit nicely into the main memory of a computer or a cluster of computers [12]. Thus TPC-C and many other traditional benchmarks are focused too much on disk performance. Additionally, traditional benchmarks do not consider the service time as correctness criteria. Therefore, this paper proposes a new benchmark developed for telecommunication databases.

3 Benchmark

This benchmark models a hypothetical telecommunication operator. The network has multiple service providers. The service providers may belong to one operator or they may belong to multiple operators. Each service provider has its own database, but for

this benchmark the databases are similar. The service provider has many customers, each with one or more subscriptions for different available services. The database represents the telecommunication services and billing information of each entity (service provider and service).

The components of the database are defined as consisting of five separate and individual classes: ServiceProvider, HomeProfile, VisitorProfile, ServiceInfo, and Subscriptions. The relationships among these classes are defined in the following UML-diagram (Figure 1). This diagram is a logical description of the classes. It has no implication to the actual, physical implementation.

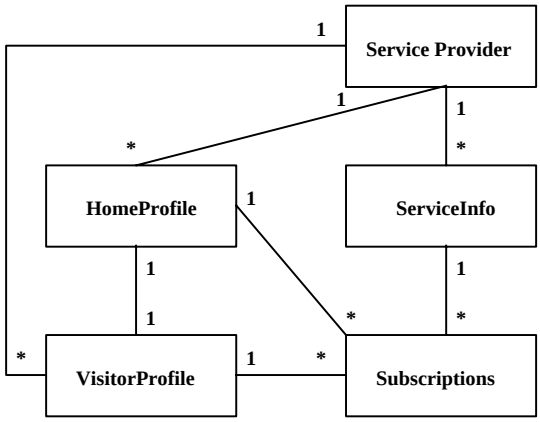

Fig. 1. Schema of the benchmark database.

Although the entity names in the model are collected from the wireless world, they can be used to model the IN services as well. The Virtual Private Network as described in [7] maps to our model by simply changing the names of the entities. The simple read and write transactions are also similar in both worlds. Likewise, the link establishment in TMN is only a distributed insert and occurs even less often than a user roaming in the wireless world.

In order for the transactions to represent a similar amount of work to all the systems, it is important that the records handled by the database servers, file systems, etc. are of the same size. Therefore, the records/rows must be stored in an uncompressed format.

The size of each item in the ServiceProvider class must be at least 100 bytes. The actual attributes of the class are not important for the benchmark itself. However, one of the attributes, called ProviderId, must uniquely identify the service provider and the information attached to it. In this benchmark we assume the ServiceProvider class (Table 1) to contain in any order or representation the attributes ProviderId, ProviderName and ProviderInfo.

Table 1. ServiceProvider class.

Data Attribute	*Description*
ProviderId	Unique identifier across the ServiceProviders
ProviderName	Name of the ServiceProvider
ProviderInfo	Additional information of the provider

The ServiceInfo class contains the information of the available service at each service provider. In this benchmark the services must be uniquely identified over all services on all service providers. The benchmark concentrates only on the service price and its usage in the service. The size of each item in the ServiceInfo class must be at least 100 bytes. For the benchmark it must have attributes (see Table 2): ServiceId, ServicePrice and ServiceName. The actual order of these attributes is not described.

Table 2. ServiceInfo class.

Data Attribute	*Description*
ServiceId	Unique identifier across the range of ServiceInfo
ServicePrice	Price of the service, at least 10 significant decimal digits and sign
ServiceName	Service name in uncompressed format

Each client of the system has one home service provider. The client information is located in the HomeProfile of that service provider. The size of the data item in the HomeProfile must be at least 100 bytes. The HomeProfile (see Table 3) contains the client's subscriber identity (SubsId), which identifies the client over all clients in the whole system. The clients also have a local identity (ClientId), which usually is much smaller than the SubsId. This id is used to connect the client with her local subscribed services. The home profile must also contain the client's true phone number, the current roaming position as the service provider id, and the client address as the connection information to the client.

Table 3. HomeProfile class.

Data Attribute	*Description*
SubsId	Unique identifier across the range of all HomeProfile
ClientId	Subscriber identification
PhoneNumber	Subscriber real phone number
CurPosition	Current position, i.e., provider identification
SubsAddress	Subscriber's address
SubscriberInfo	Additional information on the subscriber

The service provider keeps information about current roaming users in VisitorProfile (see Table 4). The class must at the minimum contain the identification of the roaming user and the identification of the user's own service provider. The size of each item in the VisitorProfile is assumed to be small, only 16 bytes. To map the visitors with the services, the visitors are also attached with a temporary ClientId for the mapping.

Table 4. VisitorProfile class.

Data Attribute	*Description*
SubsId	Visitor identification, the same as in HomeProfile
ClientId	Subscriber identification
HomeLocation	Service provider's ProviderId, used for locating HomeProfile

The class Subscriptions (see Table 5) connects the subscribers and services together. It must have the identification to the user and the service. These identifications together identify each data item in this class. The size of the data item must be at least 50 bytes. In addition to the identification attributes SubServiceId and SubClientId, the class contains information specific to this subscription. The information is stored in SubType, SubValue, and SubName attributes.

Table 5. Subscriptions class.

Data Attribute	*Description*
SubClientId	Identification of the subscriber from HomeProfile or VisitorProfile
SubServiceId	Identification of the service from Service class
SubType	Subscription type
SubValue	Connection information, normally real phone number
SubName	Subscription name

The data item identifiers of the ServiceProvider, ServiceInfo, Subscriptions, HomeProfile, and VisitorProfile must not directly represent the physical disk addresses of the items or any offsets thereof. The applications may not reference records using relative record numbers since they are simply offsets from the beginning of a file. For each nominal configuration, the test must use the minimum database size given in Table 6.

Table 6. Database size.

Table/Class	*Number of rows*
ServiceProvider	2
ServiceInfo	10
Subscriptions	50000
HomeProfile	30000
VisitorProfile	10000

The classes presented above are necessary in the databases of each service provider. The class descriptions, however, do not present the distribution aspect of the whole database system. The VisitorProfile of each service provider mainly points to HomeProfile and ServiceProviders outside the local database. The visitors are added to the VisitorProfile, when they enter the roaming area of one service provider. The current location is updated in the HomeProfile in the 'home' provider's database. On the service provider level there is also a connection, because roaming may not be allowed between two random service providers. The connections crossing over the database borders are presented in Figure 2.

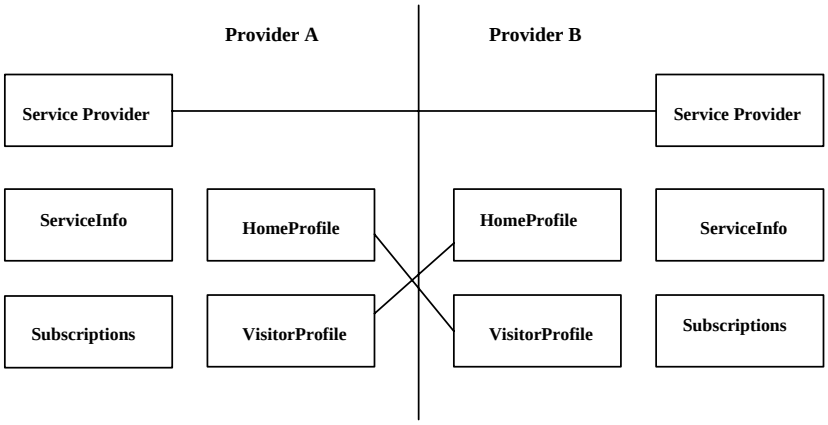

Fig. 2. Connections between two separate service provider databases.

The database contains a distributed integrity constraint between HomeProfile in the user's initial service provider and current location found from a remote service provider's VisitorProfile. Update to the current location in either class requires an atomic transaction to ensure the integrity of this constraint.

4 Test Transactions

The transactions represent the work performed when a customer uses some telecommunication services. The transactions are performed in a database of some service provider(s). This set of transactions presents the minimum that can be used for evaluating a distributed database for telecommunication. Each transaction has its own purpose in the test set.

The **GetSubscriber** transaction is used to search a specific subscriber from the database. It is mainly a simple local transaction, but can become distributed, if the query is given to a database that is not the home of the queried client. These transactions require subscriber identification, i.e. a Sid attribute as an input variable.

```
BEGIN TRANSACTION GetSubscriber
  SELECT PhoneNumber
    FROM HomeProfile
    WHERE SubsId = Sid;
  if NOT EXISTS then
    SELECT HomeLocation
      FROM VisitorProfile
      WHERE SubsId = Sid;
    SELECT PhoneNumber
      FROM HomeProfile@HomeLocation
      WHERE SubsId = Sid;
```

```
  end
END TRANSACTION
```

The **UpdateSubscriber** transaction is used to modify some of the subscriber data. This simple update transaction represents a large set of update transactions that access data in one class. The chosen update transaction updates the subscriber's name and number. It is actually the same as the *Modify Subscriber Number* service. This query also represents the *Customer Management* service in IN CS-1.

```
BEGIN TRANSACTION UpdateSubscriber
  SELECT SubscriberAddress, SubscriberAddInfo
    FROM HomeProfile
    WHERE SubsId = Sid;
    UPDATE HomeProfile
      SET SubscriberAddress = data1,
      SubscriberAddInfo = data2
    WHERE SubsId = Sid;
  END TRANSACTION
```

The **GetAccessData** transaction can be used in three different scenarios. In the first scenario, the query is used to fetch the new destination number from the database in case the number given in the parameter is an abbreviated number (e.g. this query implements the *Abbreviated Dialing* service in IN). The query returns the destination number, which is active when the query is executed. The second scenario for the **GetAccessData** is to fetch the new destination number from the database in case the number given in the parameter is forwarded to another number (e.g. this query implements the *Call Forwarding* service). The query returns the destination number, which is active when the query is executed. Finally, the third possible scenario for the **GetAccessData** query is to fetch the new destination number from the database in case the number given in the parameter is a group number (e.g. this query implements the *Universal Access Number* service). When executed the query returns the current active destination number. All of these three scenarios in this database map to the same transaction. The actual phone number is fetched from the Subscription table.

```
BEGIN TRANSACTION GetAccessData
  SELECT SubValue
    FROM HomeProfile h, Subscription s
    WHERE h.SubsId = Sid and
          s.SubClientId = h.ClientId;
  if NOT EXISTS then
    SELECT SubValue
      FROM VisitorProfile h, Subscription s
      WHERE h.SubsId = Sid and
           s.SubClientId = h.ClientId;
  end
END TRANSACTION
```

The **RoamingUser** transaction is executed when a mobile user enters the area of another service provider. It is assumed that this transaction is executed at the home service provider. In the beginning of the roaming agreement the service providers have authenticated each other. This is done before this transaction is initialised. Firstly, the current location of the mobile user is determined. Secondly, the VisitorProfile data from the current visiting service provider’s VisitorProfile is

removed, i.e. this can be a distributed write. Thirdly, a new VisitorProfile data is inserted in the VisitorProfile of the service provider in the mobile user's new position. Fourthly, the current location in the HomeProfile is updated to the new location.

```
BEGIN TRANSACTION RoamingUser
  SELECT CurPosition
    FROM HomeProfile
    WHERE SubsId = Sid;
  DELETE VisitorProfile@CurPosition
    WHERE SubsId = Sid;
  INSERT INTO VisitorProfile@NewPosition
    VALUES(data)
  UPDATE HomeProfile
    SET CurLocation = PosId
    WHERE SubsId = Sid;
END TRANSACTION
```

In order for the transactions to represent a similar amount of work to all the systems, it is important that the fractions of different transactions are specified. We use the traditional 80-20 fractions between read and update transactions. The read-only transactions GetSubscriber and GetAccessData are 80 % of all transactions. Table 7 shows the fraction of the different transactions in the test load. The transactions access both local and distributed data. The difference between the access ratios on different transactions is presented in Table 8. The update transactions use the same distribution on reads, as well. The GetSubscriber and GetAccessData can be made distributed only by knowing the internal data distribution and executing the transaction not on the database where the actual data is stored.

Table 7. Transaction mix.

Transaction name	*Fraction of all transactions*
GetSubscriber	60
UpdateSubscriber	5
GetAccessData	20
RoamingUser	15

Table 8. Distribution of local and remote operations.

Transaction name	*Remote read*	*Remote update*	*Local read*	*Local update*
GetSubscriber	5	0	95	0
UpdateSubscriber	0	0	0	100
GetAccessData	5	0	95	0
RoamingUser	0	80	0	20

5 Database Properties

The ACID (Atomicity, Consistency, Isolation, and Durability) properties of transaction processing systems must be supported by the system under test during the running of this benchmark. These tests are adopted from the TPC-B benchmark (http://www.tpc.org). They are intended to demonstrate that the ACID properties are supported by the system under test and enabled during the performance period.

All mechanisms needed to ensure full ACID properties must be enabled during both the measurement and test periods. For example, if the system under test relies on undo logs, then logging must be enabled even though no transactions are aborted during the measurement period. When this benchmark is implemented on a distributed system, tests must be performed to verify that home and remote transactions, including remote transactions that are processed on two nodes, satisfy the ACID properties.

In addition to the traditional database properties a telecommunication database often implements some mechanism to increase data availability. The availability increment needs to be measured, as well.

5.1 Atomicity Tests

The system under test must guarantee that transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially completed operations leave any effects on the data. This is tested by a telecommunication transaction for randomly selected subscription and by verifying that the appropriate data items have been changed in the HomeProfile, ServiceInfo, Subscription, and VisitorProfile classes. Additionally, perform a telecommunication transaction for randomly selected subscription, substituting an ABORT of the transaction for the COMMIT of the transaction. Verify that the appropriate records have not been changed in HomeProfile, ServiceInfo, Subscription, and VisitorProfile classes.

5.2 Consistency Tests

Consistency is the property of the application that requires any execution of the transaction to take the database from one consistent state to another. A consistent state for the telecommunication database is defined to exist when:

- For all branches, the price of the service in the Service class is the same.
- Subscriber information in the HomeProfile and VisitorProfile, if it exists, is the same in all branches.

To verify the consistency of HomeProfile and VisitorProfile perform the following. First, verify that the HomeProfile and VisitorProfile class is initially consistent by determining the data items in the classes and verify that each client in the HomeProfile has at most one entry in all instances of VisitorProfile classes on separate service providers. Second, verify that HomeProfile and VisitorProfile classes

are consistent after applying transactions to the database by performing the following: Using the standard driving mechanism, submit a number of telecommunication transactions equal to at least 1000 transactions and note the number of transactions that are reported as committed. If the number of committed transactions is not equal to the number of submitted transactions, explain why. Re-verify the consistency of the HomeProfile and VisitorProfile classes of each provider.

5.3 Isolation Tests

Operations of concurrent transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order. This property is commonly called serialisability. Sufficient conditions must be enabled at either the system or application level to ensure serialisability of transactions under any mix of arbitrary transactions. The system or application must have full serialisability enabled.

For conventional optimistic schemes, isolation should be tested as described below, where transactions 1 and 2 are versions of the standard telecommunication transaction. Systems that implement other isolation schemes may require different validation techniques. The isolation tests are the following:

- Start transaction 1 and start transaction 2 before transaction 1 is committed. Transaction 2 attempts to read the same subscription record as transaction 1 is updating. Verify that transaction 2 reads the old value of the subscription. Verify that subscription information reflects the results of transaction 1's update only.
- Start transaction 1 and start transaction 2 before transaction 1 is committed. Transaction 2 attempts to write the same subscription record as transaction 1 is updating. Verify that transaction 2 is aborted and subscription information reflects the results of transaction 1's update only.

5.4 Durability Tests

The tested system must guarantee the ability to preserve the effects of committed transactions and ensure database consistency after recovery from any one failure:

- Instantaneous interruption in processing, which requires a system reboot to recover.
- Failure of all or part of the memory.

A transaction is considered committed when the transaction manager component of the system has written the commit record(s) associated with the transaction to a durable medium. A durable medium is a data storage medium that is either:

- An inherently non-volatile medium, e.g., magnetic disk, magnetic tape, optical disk, etc., or
- A volatile medium with its own self-contained power supply that will retain and permit the transfer of data, before any data is lost, to an inherently non-volatile medium after the failure of external power. A configured Uninterruptible Power Supply (UPS) is not considered external power, if its price is included in the total system price.

The intent of these tests is to demonstrate that all successful transactions have in fact been committed in spite of any single failure. The system crash test and the loss of memory test must be performed with a full terminal load using a fully scaled database. At the moment the failure is induced, there must be multiple local and remote transactions in progress. Distributed configurations must have distributed transactions in progress. For each of the failure types perform the following steps:

- Start submitting telecommunication transactions. On the driver system record committed transactions in a "success" file.
- Cause a failure selected from the list above.
- Restart the system under test using normal recovery procedures and measure the time required for partial and/or full restore.
- Compare the contents of the "success" file to verify that every record in the "success" file has a corresponding record in the database.

5.5 Availability Tests

On telecommunication a database that provides no services in case of one failure, is often not enough. The telecommunication environment requires more tolerable service. A fault-tolerant server can provide at least a reduced amount of service even during earlier specified failure situations. On such a continuous, or highly available, server the induction of one failure does not force the system to full recovery. However, the durability tests are needed to ensure that the effects of committed transactions are persistent even if a part of the system fails.

The continuous database server that guarantees data persistence must not loose the data even when the system is unable to provide service to its client due to the maximum concurrent number of failures it tolerates. The database content must still contain the modifications of all committed transactions. This can be verified using the consistency test with the failure induction test.

The availability measurement of a continuous server is actually the service level degradation during the failure situation and the duration of the failure recovery period. The measurement of the recovery duration can be done as black-box by using the service level degradation as indication of the system status. This is very coarse and should be avoided. Using a white-box approach, the duration is measured with internal information of the system functionality. The period starts when one recoverable part of the server fails and ends when the part is again a fully functional and operational part of the system.

6 Computation of Ratings

The reported metric, transactions per second (tpsT), is the total number of successfully committed transactions divided by the elapsed time of the interval. A committed transaction is counted as successful only when it has both started and completed during the measurement interval and it has completed before its deadline.

A transaction success ratio (successT) is the total number of successfully committed transactions divided by the total number of transactions entered to a system under test during the measurement interval. The reported value transaction miss ratio (missT) is simply one minus successT. It contains all unsuccessfully ended transactions.

It must be demonstrated that the configuration, when tested at the reported tpsT and missT rates, has the property of stable throughput despite small changes in the number of concurrently active transactions.

The measurement period must:

- Begin after the system reaches a sustained "steady state";
- Be long enough to generate reproducible tpsT and missT results;
- Extend uninterrupted for at least 15 minutes;
- For systems that defer database writes to durable media, the recovery time from instantaneous interruptions must not be appreciably longer at the end of the measurement period than at the beginning of the measurement period.

The duration of the recovery period must be reported, if applicable. On continuous or highly available servers, the reported duration is the recovery period of one recoverable unit. If a highly available server tolerates only a fixed number of concurrent failures, then the server's recovery period must be reported, when more failures happen.

7 Conclusions

The databases are essential for the current telecommunication services and timely response from the database is essential for the client using the services. The traditional database benchmarks do not consider the service time as correctness criteria. However, timely response is an important correctness criterion of a real-time database system such as Network Services Database.

The telecommunication field has different services, which have different database needs. The intelligent network concept models its services in a traditional fixed-line network, GSM services are wireless, and TMN services have their own requirements for the database. These different requirements are modelled in the benchmark proposed in this paper.

The benchmark consists of a simple database schema modelling different telecommunication service requirements. Similarly, the transactions (or services) are selected so that they represent as large a set of real services as possible. The proposed benchmark requires that the database maintain all ACID properties.

Because the telecommunication environment requires more tolerable services, a fault-tolerant server can provide at least a reduced amount of service even during specified failure situations. Therefore, we have designed availability tests for the benchmark.

As a conclusion, the proposed benchmark very well represents telecommunication requirements and challenges. Therefore, this benchmark will be a very good instrument when comparing different commercial and research databases for telecommunication use.

References

[1] I. Ahn. Database issues in telecommunications network management. ACM SIGMOD Record, 23(2):37-43, June 1994.

[2] T. F. Bowen, G. Gopal, G. Herman, and W. Mansfield Jr. A scale database architecture for network services. IEEE Communications Magazine, 29(1):52-59, January 1991.

[3] S. Cha, B. Park, S. Lee, S. Song, J. Park, J. Lee, S. Park, D. Hur, and G. Kim. Object-oriented design of main-memory dbms for real-time applications. In 2nd Int. Workshop on Real-Time Computing Systems and Applications, pages 109-115, Tokyo, Japan, Oct. 1995.

[4] J. Gray, editor. The Benchmark handbook for database and transaction processing systems, second edition. Morgan Kaufmann, 1993.

[5] S.-O. Hvasshovd, S. E. Bratsberg, and Ø. Torbjørnsen. An ultra highly available dbms. In Proceedings of the VLDB 2000, p. 673.

[6] ITU. Q-Series Intelligent network recommendation overview. Recommendation Q.1200. ITU, International Telecommunications Union, Geneva, Switzerland, 1993.

[7] M. Jarke and M. Nicola. Telecommunication databases – applications and performance analysis. In Databases in Telecommunications, Lecture Notes in Computer Science, 1819, pages 1-15, Edinburgh, UK, Co-located with VLDB-99, 1999.

[8] R. Kerboul, J.-M. Pageot, and V. Robin. Database requirements for intelligent network: How to customize mechanisms to implement policies. In Proceedings of the 4th TINA Workshop, volume 2, pages 35-46, September 1993.

[9] Y.-K. Kim and S. H. Son. Developing a real-time database: The StarBase experience. In A. Bestavros, K. Lin, and S. Son, editors, Real-Time Database Systems: Issues and Applications, pages 305-324, Boston, Mass., 1997. Kluwer.

[10] B. Purimetla, R. M. Sivasankaran, K. Ramamritham, and J. A. Stankovic. Real-time databases: Issues and applications. In S. H. Son, editor, Advances in Real-Time Systems, pages 487-507. Prentice Hall, 1996.

[11] K. Raatikainen. Database access in intelligent networks. In Proceedings of the IFIP TC6 Workshop on Intelligent Networks, pages 163-183, Lappeenranta, Finland, 1994. Lappeenranta University of Technology.

[12] M. Ronström. Design and Modelling of a Parallel Data Server for Telecom Applications. PhD thesis, Ericsson Utveckling AB, 1997.

[13] M. Ronström. Database requirement analysis for a third generation mobile telecom system. In Databases in Telecommunications, Lecture Notes in Computer Science, 1819, pages 90-105, Edinburgh, UK, Co-located with VLDB-99, 1999.

[14] S. Shih, Y-K. Kim, and S. H. Son. Performance evaluation of a firm real-time database system. In 2nd International Workshop on Real-Time Computing Systems and Applications, pages 116-124, Tokyo, Japan, October 1995.

[15] J. Taina and K. Raatikainen. Experimental real-time object-oriented database architecture for intelligent networks. Engineering Intelligent Systems, 4(3):57-63, September 1996.

[16] J. Taina and K. Raatikainen. Database usage and requirements in intelligent networks. In D. Gaïti, editor, Intelligent Networks and Intelligence in Networks, pages 261-280, Paris, France, 1997. Chapman&Hall.

[17] J. Taina and K. Raatikainen. Requirements analysis of distribution in databases for telecommunications. In Databases in Telecommunications, Lecture Notes in Computer Science 1819, pages 74-89, Edinburgh, UK, Co-located with VLDB-99,1999.

[18] Ø. Torbjørnsen, S.-O. Hvasshovd, and Y.-K. Kim. Towards real-time performance in a scalable, continuously available telecom DBMS. In Proceedings of the First Int. Workshop on Real-Time Databases, pages 22-29. Morgan Kaufmann, 1996.

[19] V. Wolfe, L. DiPippo, J. Prichard, J. Peckham, and P. Fortier. The design of real-time extensions to the open object-oriented database system. Technical report TR-94-236, University of Rhode Island, Department of Computer Science and Statistics, February 1994.