



Distributed University Registration Database System Using Oracle 9i

Sana J. Alyaseri

Department of Computer Engineering

College of Engineering

University of Basrah

E-mail: sana_jabbar@yahoo.com

Abstract

In database and information systems, increasing attention has been focused lately on parallel and distributed database systems. The future of large database systems lies into the realm of distributed computing. The main reason for this is that distributed computing can be constructed at a low cost without the need for any specialized technology, using existing sequential computers and relatively cheap computer networks. The basic motivations for distributed databases are improved performance, increased availability, share ability, expandability, and access flexibility. Oracle9i is one of the most famous database management systems that provide distributed features. Oracle9i supports transparent distributed gains to access data from multiple databases. It also provides many other distributed features, such as transparent distributed transactions and transparent automatic two-phase commit. This paper presents a distributed registration database system using Oracle9i distributed features such as replication and fragmentation. The analysis and design steps of the distributed registration database system are shown as well as to the practical steps required for implementing the distributed database system using Oracle9i.

Keywords: Distributed database, Oracle9i distributed database, Replication, Fragmentation

1. Introduction

Distributed computing is one of the most recent and important development in the computing era. The next decade of research in database era will focus toward delivering widely available access to unprecedented amounts of constantly expanding data that is distributed all over the net (Loose & Church, 2004; Abiteboul et al., 2003; Ohta & Ishikawa, 2003). Users will benefit from new machine learning technologies that mine new knowledge by integrating an analyzing huge amounts of widely distributed data to uncover and report upon subtle relationships and patterns of events that are not immediately recognized by direct human inspection (Aljanaby et al., 2005; Kossman, 2000).

Distributed Database (DDB) technology emerged as merger of two technologies; database technology and data communication technology. The maturation of Database Management System (DBMS) technology has coincided with significant development in distributed computing and parallel processing technology. The end result is the emergence of Distributed Database Management System (DDBMS) and parallel database management system. These systems have started to become the dominant data management tools for highly accessed data. (Ozsu, 2007) A distributed database is a collection of multiple logically interconnected databases distributed over a computer network, or a single logical database that is spread physically across computers in multiple locations that are connected by a data communications link and a DDBMS as a software system that managed a distributed database, while making the distribution transparent to the users. There are two main types of distributed databases. The first type is homogeneous database that has same DBMS at each node and the other type is heterogeneous database that has different DBMSs at different nodes (Ozsu, 2007).

Reliability and availability are the most common potential advantages cited for a distributed database. Reliability is defined as the possibility that a system is running at a certain time point. Availability is the probability that the system is continuously available during a time interval. When the data and DBMS software are distributed over several sites, one site may fail while other sites continue to operate. This improves both reliability and availability (Loose & Church, 2004).

Oracle introduced inter-database connectivity with SQL*Net in Oracle Version 5 and simplified its usage considerably with the database links feature in Oracle Version 6, opening up a world of distributed possibilities. Oracle9i now supplies a variety of techniques that can be used to establish inter-database connectivity and data sharing. It provides a

robust and complete solution that addresses all the needs that may arise when operating in a distributed environment. It includes communication between users on the Oracle database using queues, data replication, and distributed data access in both homogenous and heterogeneous environments (Oracle Corporation, 2002a; Oracle Corporation, 2002c). Oracle9i even moved beyond that and was built using the Grid structure. This will dramatically lower the cost of computing, extend the availability of computing resources, deliver higher productivity, and higher quality (Oracle Corporation, 2002b).

The remainder of this paper is organized as follows. In section 2, some of the concepts and terminology related to the distributed features of Oracle9i are introduced. The analysis and design of the distributed registration database system is presented in section 3. Development and implementation is presented in section 4. Finally, section 5 concludes the paper.

2. Oracle9i Distributed Features

There is a great deal of confusion surrounding the various products and terminology from Oracle. It's worthwhile to clarify some of these terms up front so the reader gets the most benefit from this paper. This section introduces Oracle9i terminology and distributed features related to the work presented in this paper.

2.1 Distributed Terminology

The terms database and database instance are often used interchangeably, but they are not the same. In Oracle parlance, a *database* is the set of physical files containing data. These files comprise table spaces, redo logs, and control files. A *database instance*, or simply *instance*, is the set of processes and memory structures that manipulate a database. A database may be accessed by one or more database instances, and a database instance may access exactly one database (Oracle Corporation, 2002c). *Oracle Parallel Server (OPS)* is a technology that allows two or more database instances, generally on different machines, to open and manipulate one database. In other words, the physical data files in a database can be seen, inserted, updated, and deleted by users logging on to two or more different instances; the instances run on different machines but access the same physical database (Oracle Corporation, 2002a).

The Parallel Query Option (PQO) is a technology that can divide complicated or long-running queries into several independent queries and allocate separate processes to execute the smaller queries. A coordinator process collects the results of the smaller queries and constructs the final result set. Parallel queries are effective only on machines that have multiple CPUs. Oracle also introduced the parallel DML feature in Oracle9i. Parallel DML is similar to parallel query, except that the independent processes perform DML (Baumgartel, 2002). For example, an update of several hundred thousand rows can be doled out to several processes that execute the update on separate ranges of the table.

Figure 1 illustrates an environment in which data in two or more database instances is accessible as though this data were in a single instance. This access may be read-only, or it may permit updates to one or many instances. The referenced data may be real time, or it may be seconds, hours, or days old. Generally, the different database instances are housed on different server nodes, and communication between them is via Net8 (for Oracle9i). In addition to database servers, a distributed database system usually includes application servers and clients. Application servers, like database servers, typically are high-capacity machines that run intensive utilities such as web applications, Oracle's application cartridges, report generators, and so forth (Oracle Corporation, 2002a).

All Oracle databases in a distributed database system use Oracle's networking software, Net8, to facilitate inter-database communication across a network. Just as Net8 connects clients and servers that operate on different computers of a network, it also allows database servers to communicate across networks to support remote and distributed transactions in a distributed database. Net8 makes transparent the connectivity that is necessary to transmit SQL requests and receive data for applications that use the system. Net8 performs all processing independent of the underlying network operating system (Oracle Corporation, 2002c).

Optionally, an Oracle network can use Oracle names to provide the system with a global directory service (Oracle Corporation, 2002a). When an Oracle network supports a distributed database system, you can use Oracle names servers as central repositories of information about each database in the system to ease the configuration of distributed database access. Database links are the invisible glue that makes location transparency possible (Oracle Corporation, 2002a). In more technical terms, a database link defines a connection from one database instance to another, and this definition is stored in the Oracle data dictionary. Since database link connections log in to a normal account in the remote database instance, this provides a complete control over its privileges and quotas.

2.2 Transparency

Users of a distributed database system need not to be aware of the location and functioning of the parts of the database with which they work. The goal of transparency is to make a distributed database system appear as though it is a single Oracle database (Oracle Corporation, 2002c). The DBA and network administrators can ensure that the distributed nature of the database remains transparent to users. An Oracle distributed database system has features that allow

application developers and administrators to hide the physical location of database objects from application and users. Location transparency exists when a user can universally refer to a database objects such as a table. Local views can provide location transparency for local and remote tables in a distributed database system. For example, assume that table EMP is stored in a local database. Another table, DEPT, is stored in a remote database. To make the location of, and relationship between, these tables transparent to users of the system, a view named company can be created in the local database that joins the data of the local and the remote servers.

```
CREATE VIEW company AS
SELECT empno, ename, dname
FROM emp a, dept@Basrah_link b
WHERE a.deptno = b.deptno;
```

When users access this view, they do not know, or need to know, where the data is physically stored, or if data from more than one table is being accessed. Thus, it is easier for them to get required information. The following example provides data from both the local and remote database table.

```
SELECT * FROM company;
```

Synonyms are very useful in both distributed and non-distributed environments because they hide the identity of the underlying object, including its location in a distributed database system. If the underlying object must be renamed or be moved, only the synonym needs to be redefined; applications based on the synonym continue to function without modification. A synonym can be created for any table, type, view, snapshot, sequence, procedure, function, or package. All synonyms are stored in the data dictionary of the database in which they are created. To simplify remote table access through database links, a synonym can allow single-word access to remote data, isolating the specific object name and the location from users of the synonym (Oracle Corporation, 2002a).

The syntax to create a synonym is:

```
CREATE [PUBLIC] synonym_name
FOR [schema.] object_name [@database_link_name]
```

Oracle allows the following standard DML statements to reference remote tables:

```
SELECT (queries), INSERT, UPDATE, DELETE
SELECT... FOR UPDATE
LOCK TABLE
```

Queries including joins, aggregates, sub queries, and SELECT ... FOR UPDATE can reference any number of local and remote tables and views.

2.3 Replication

Oracle offer two modes of replication; the choice of mode will depend on the reason for replication deployment. In some cases, a hybrid is appropriate. Multi-master replication (also called peer-to-peer or n-way replication) enables multiple sites, acting as equal peers, to manage groups of replicated database objects. Each site in a multi-master replication environment is a master site, and each site communicates with the other master sites. Asynchronous replication is the most common way to implement multi-master replication. Other ways include synchronous replication and procedural replication. Using asynchronous replication, information about a DML change on a table is stored in the deferred transactions queue at the master site, where the change occurred. These changes are called deferred transactions. The deferred transactions are pushed (or propagated) to the other participating master sites at regular intervals. The amount of time in an interval can be controlled. Using asynchronous replication means that data conflicts are possible, because the same row value might be updated at two different master sites at nearly the same time. However, some techniques can be used to avoid conflicts and, if conflicts occur, Oracle provides rebuilt mechanisms that can be implemented to resolve them. Information about unresolved conflicts is stored in an error log (Baumgartel, 2002; Pratt, 2001).

A materialized view contains a complete or partial copy of a target master from a single point in time. The target master can be either a master table at a master site or a master materialized view at a materialized view site. A master materialized view is a materialized view that functions as a master for another materialized view. A multi-master materialized view is one that is based on another materialized view, instead of a master table. Materialized views provide the following benefits:

- Enable local access, which improves response time and availability.
- Offload queries from the master site, because users can query the local materialized view instead.

- Increase data security by allowing you to replicate only a selected subset of the target master's data set.

A materialized view may be read-only, updatable, or writeable, and these types of materialized views provide benefits in addition to those listed previously.

2.4 Database Link

A database link is a pointer that defines a one-way communication path from an Oracle database server to another database server. The link pointer is actually defined as an entry in a data dictionary table. To access the link, you must be connected to the local database that contains the data dictionary entry. A database link connection allows local users to access data on a remote database. The great advantage of database links is that they allow users to access another user's objects in a remote database so that they are bounded by the privilege set of the object's owner.

3. Design of the Distributed Registration System

In this section, the main steps for analyzing and designing distributed registration database system will be described. The distributed registration system is for a university has two campuses, the first campus in Basrah governorate and the other in Mesan governorate. The central campus will be in Basrah campus and it is responsible for defining all colleges and departments that are allowed to be added in both campuses. The instructor can be added from any site. The central campus is responsible for producing courses for both sites. Each site is responsible for producing its schedule. Student can be added or registered from any site regardless of his study location. The tables of colleges, departments, instructors and courses are the same as in both sites, that's why the tables are replicated in both sites. The information about the students in each campus is stored in its site, so the student table; teach table (schedule), and the registration table are fragmented. Table 1 shows the distributed technique used for each table and the corresponding technique in Oracle9i (Fodor, 2007; Hoffer et al., 2008). The entity relationship diagram for the distributed registration system is shown in Figure 2. The architecture of Oracle distributed registration database system is shown in Figure 3.

4. Development and Implementation

In this section, the implementation steps of the distributed registration database system using Oracle9i will be described. Oracle9i DBMS was installed with its global name in both sites, Basrah_db and Mesan_db. After setting a connection between the two sites, the parameters that must be set or added to the initSID.ora file are initialized as in Table 2. The recommended initial table space Requirements are shown in Table 3.

To setup Net8, an Oracle Net listener must be running on each of the servers involved in the replication environment and the listener.ora file must contain an entry for the instance. Net8 is then set on each site using Net Configuration Assistant tool by filling the service name and the IP address. For a successful transaction process, a database link is created for each site (Basrah link and Mesan link). Before setting up database links, it is important that all replicated databases have unique global name.

The next step is creating all tables for Basrah, the centre site. Using the Enterprise Manager Console in Oracle9i, Mesan_db is added from Basrah site, and Basrah_db from Mesan site. After all the previous steps have been completed at all sites to be involved in a master replication, the OEM replication management setup wizard is used to complete the configuration. The Enterprise Manager Console is used to setup the master sites. Through the master sites setup, the master sites, Basrah_db and Mesan_db are defined. The default user admin (repadmin) is also defined during this process. This user is responsible of the multi-master replication process and he is allowed to be the propagator and receiver for transaction. Then, a default link scheduling (refresh interval) is set. To create a master group, it must to login to the Enterprise Manager Console as a repadmin. Then, setting the refresh type which is synchronous and specified the database link on which the multi-master replication was made on. The multi-master tables is then specified in the object of the created master group and generate replication supports on multi master tables to other sites. During that actually two process occurred

- 1) Setup the deferred queue to be pushed automatically to run the following at each master site.

```

Connect repadmin/<password>
Begin
  Dbms_defer_sys.schedule_push(
    Destination    => '<destination databases global name>.world',
    Interval       => '/10:mins*/sysdate+10/(60*24)',
    Next_date      => sysdate,
    Stop_on error  => false,
    Delay_seconds  => 0,
    Parallelism    => 1);

```

End;

- 2) Setup the deferred queue to be purged automatically to run the following at each master site.

```
Connect repadmin/<password>
```

```
Begin
```

```
Dbms_defer_sys.schedule_purge(
  Next_date      => sysdate,
  Interval       => '1:hr*/sysdate*1/24',
  Delay_seconds  => 0,
  Rollback_segment => '');
```

```
End;
```

Completing the above steps will reach the finishing of multi-master replication. Through the same tool, that is the Enterprise Manager Console, the materialized view sites are then setuped. Through the setup process the materialized view master site-Basrah and materialized view site (Mesan) is defined. A schema to be created in the materialized view sites is selected which is a registrar schema. The default user admin (MVadmin) which is responsible of the materialized view replication and he is the propagator and receiver for transactions. The next process is to specify the interval of refresh for each materialized view and its name and query, the fast refresh type for some materialized view and the complete refresh for the others were then selected. There were some problems in the materialized view refresh, so some of the materialized views were manually created as shown below:

Example about fast refresh:

```
create materialized view log on Department
create materialized view Department
  refresh fast
  next sysdate+1/1440
as
select * from Department @Basrah_db
```

Example about complete refresh:

```
create materialized view Course
  refresh complete
  next sysdate+1/1440
as
select * from Course @Basrah_db
```

After determining the fragmented tables, the same tables into the two sites (Basrah_db and Mesan_db) are created. In each site a remote synonym for each fragment table has been created. The insert, delete and update for DML commands are done as shown in the following algorithm:

5. Conclusion

The future of huge databases relies on the network based parallelism. Distributed and parallel databases are becoming one of the hot topics in the last decade. The need for more distribution of data is increasing as we rely more on the net. The internet and the World Wide Web significant developments are encouraging DBMS vendors on making their products web-enabled. This is leading to distributing data in many servers over the net.

Distributed database management systems are commercially available. Oracle is one of the leading database corporations who moved early toward distribution. Oracle9i is one of the most common database management systems that supports distributed features. It supports transparent data access across multiple databases. Also, it provides many other distributed features, such as transparent distributed transactions and transparent automatic two-phase commit. This paper presented a distributed database registration information system using Oracle9i distributed features such as replication and fragmentation. In addition to that, the paper shows the practical steps in implementing a distributed database using Oracle9i in a local area network.

References

Abiteboul, S., Bonifati, A., Cobena, G., Manolesco, I., & Milo, T. (2003). Dynamic XML documents with distribution and replication. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (pp. 527

– 538), San Diego, California, USA.

Aljanaby, A., Abuelrub, E., & Odeh, M., (2005). A survey of parallel and distributed query optimization". *International Arab Journal of Information Technology (IAJIT)*, 2(1), 48-57.

Baumgartel, P. (2002). *Oracle Replication: An Introduction*. New York, USA: Addison Wesley.

Fodor, A.. (2007). Aspects of data allocation in distributed database systems. In *Proceedings of the 7th European Conference of Young Research and Science Workers (TRANSCOM 2007)*, pp. 51-53.

Hoffer, J., George J., & Valacich J. (2008). *Modern Systems Analysis and Design* (5th ed.). New Jersey, USA: Prentice Hall.

Kossmann, D. (2000). The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4), 422-469.

Loose, R., & Church, L. (2004). Information retrieval with distributed databases: Analytic models of performance. *IEEE Transactions on Parallel and Distributed Systems*, 15(1), 18-27.

Ohta, M., & Ishikawa, H. (2003). *An Active Web-based Distributed Database System for E-Commerce*. Technical Report, Department of Electronics and Information Engineering, Tokyo Metropolitan University, Japan.

Oracle Corporation. (2002a). *Oracle9i Database Administrator's Guide Release 2 (9.2)*. Oracle Corporation: California, USA.

Oracle Corporation. (2002b). *Oracle and the Grid (an Oracle White Paper)*. Oracle Corporation, World Headquarters, California, USA, November 2002.

Oracle Corporation. (2002c). *Oracle's Solutions for the Distributed Environment (an Oracle White Paper)*. Oracle Corporation, World Headquarters, California, USA, June 2002.

Ozsu, T. (2007). *Principles of Distributed Database Systems* (3rd ed.). New Jersey, USA: Prentice Hall.

Pratt, M. (2001). *Oracle9i Replication*. California, USA: Oracle Corporation.

Table 1. Distributed database techniques used for different tables

Table Name	Distributed Database Technique	Oracle9i Distributed Database Technique
Location	Replication Technique	Materialized View (Readable)
College	Replication Technique	Materialized View (Readable)
College_Location_Department Intersection	Fully Replication Technique	Multi-master Replication
Department	Replication Technique	Materialized View (Readable)
Student	Horizontal Fragmentation	Manually (because there is no Oracle tool that supports this technique)
Instructor	Fully Replication	Multi-master Replication
Course	Replication Technique	Materialized View (Readable)
Student_Course (Teach) Intersection	Horizontal Fragmentation	Manually (because there is no Oracle tool that supports this technique)
Instructor_Course (Registration) Intersection	Horizontal Fragmentation	Manually (because there is no Oracle tool that supports this technique)

Table 2. Parameters initialization values

Parameter Name	Initial Value
COMPATIBLE	9.0.1.0.0(minimum 9.X) OSS recommended this equate to the server release
SHARED_POOL_SIZE	30M (for basic)
PROCESSES	Added 12 to the current value
GLOBAL_NAMES	TRUE
DB_DOMAIN	extension component of the local Database global name
OPEN_LINKS	4 was added
DISTRIBUTED TRANSACTION	5 was added
REPLICATION_DEPENDENCY_TRACKING	TRUE
JOP_QUEUE_INTEREVAL	10 seconds
JOP_QUEUE_PROCESSES	3 was added
PARALLEL_MAX_SERVERS	10
PARALLEL_MIN_SERVERS	2

Table 3. Recommended initial table-space Requirements

Table-space	Initial Value
SYSTEM	At least 80 Mb free
ROLLBACK SEGMENT	At least 60 Mb free
TEMPORARY	At least 40 Mb free

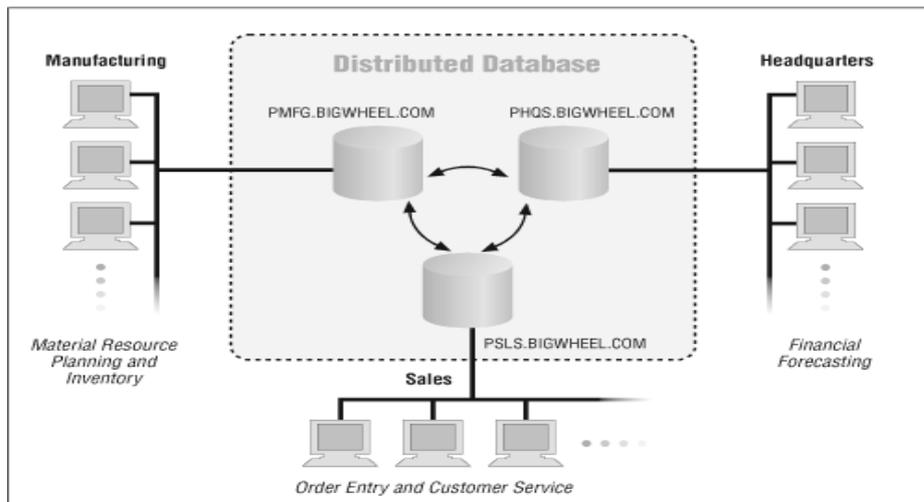


Figure 1. Distributed Database System

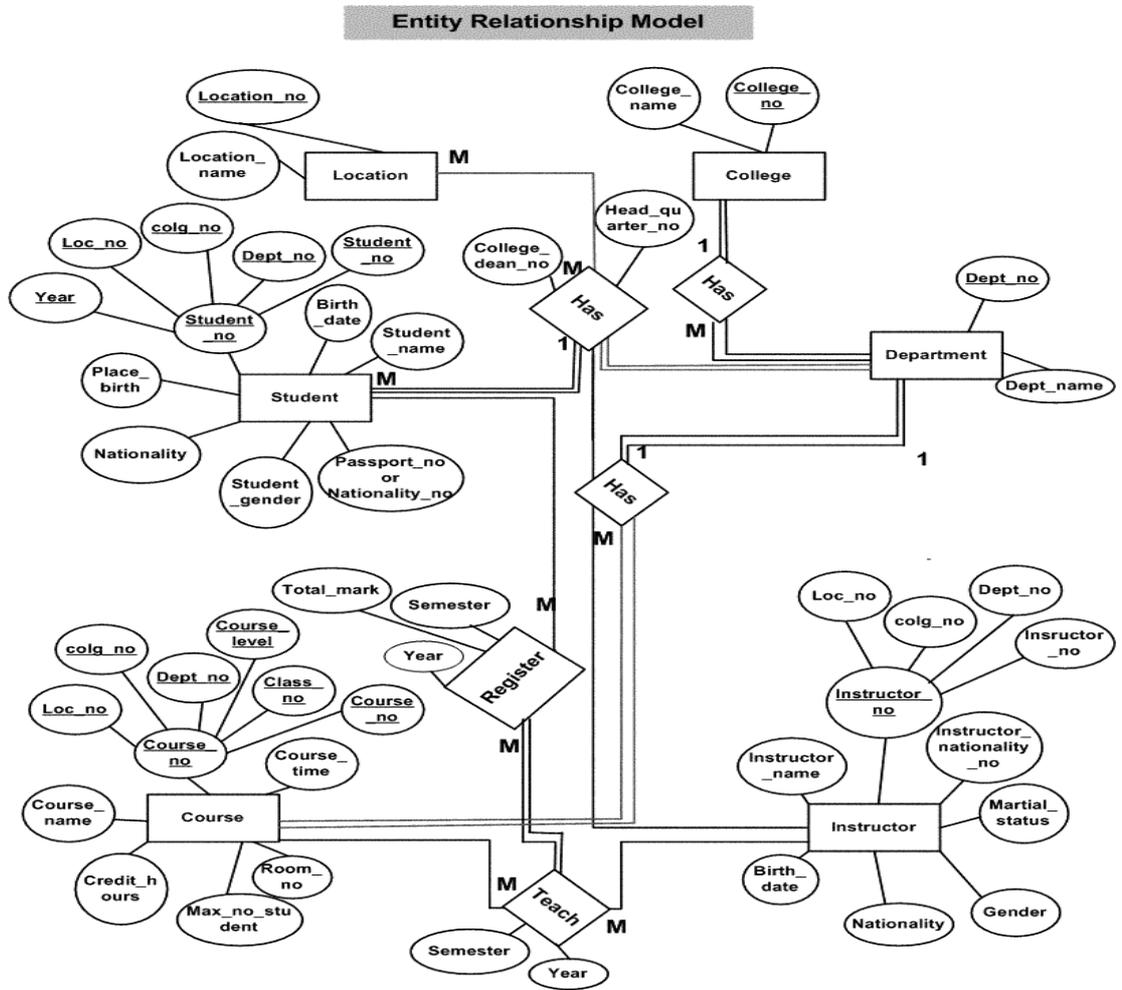


Figure 2. Entity Relationship Model

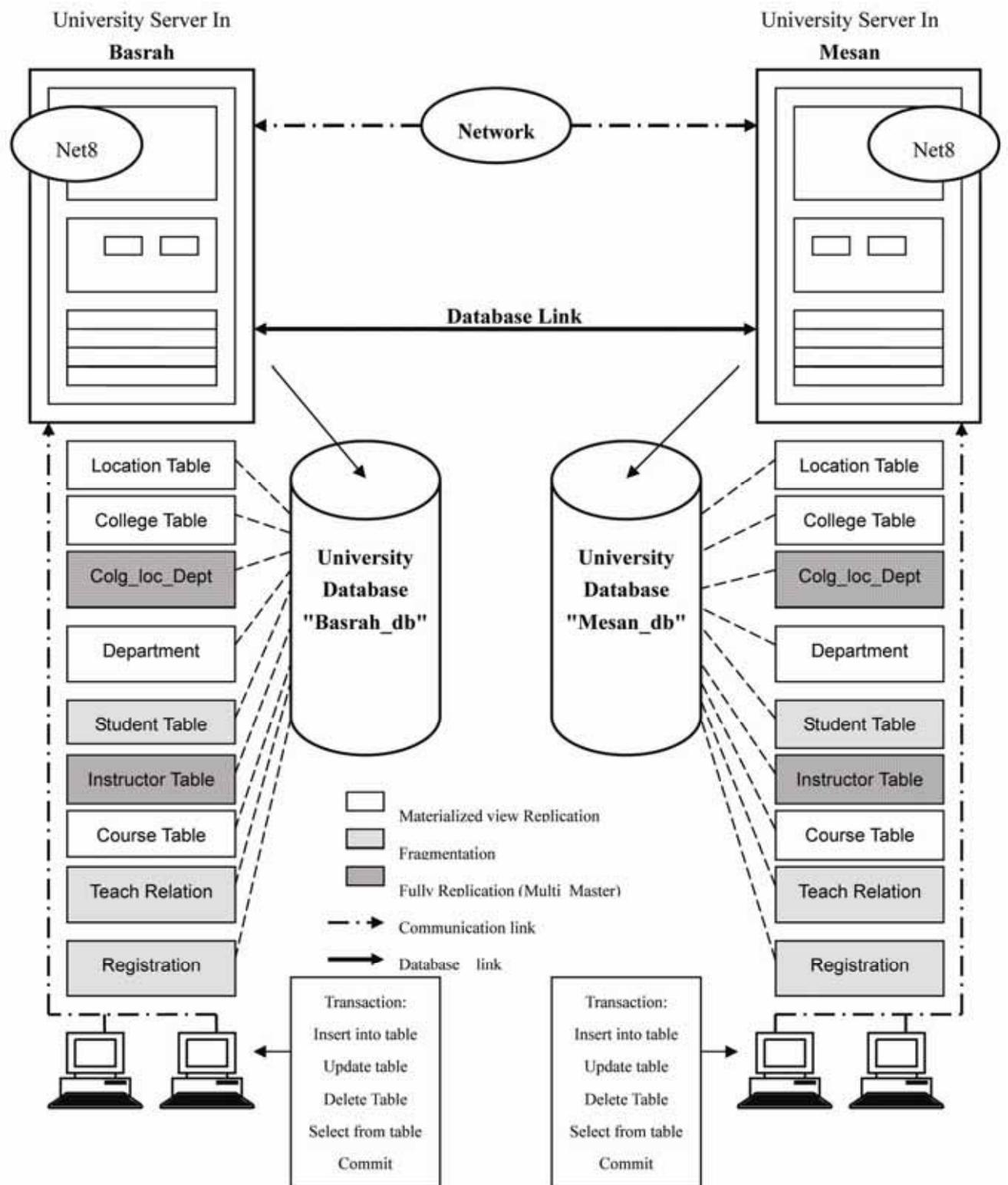


Figure 3. Oracle distributed registration database system