A CONCEPT OF DISTRIBUTED DATABASE SYSTEM

Preety Khatri*

ABSTRACT

A database that consists of two or more data files located at different sites on a computer network. Because the database is distributed, different users can access it without interfering with one another. However, the DBMS *must periodically synchronize the scattered databases to make sure that they all have consistent data. To ensure that the distributive databases are up to date and current, there are two processes: replication and duplication. Replication involves using specialized software that looks for changes in the distributive database. This process can also require a lot of time and computer resources. Duplication It basically identifies one database as a master and then duplicates that database. A distributed database system allows applications to access data from local and remote databases. Besides distributed database replication and fragmentation, there are many other distributed database design technologies. For example, local autonomy, synchronous and asynchronous distributed database technologies. In this paper we will discuss about database architecture, design etc. & how these technologies implementation and depend on the needs of the business and the sensitivity/confidentiality of the data to be stored in the database.*

*Senior Lecturer, Northern India Engineering College, New Delhi

INTRODUTION

Before going on to **Distributed Database Systems**, let us briefly explain you about **Centralised Database Systems**. Here all system components i.e. the database and the Database Management System (DBMS) reside at a single computer or site. Users may be able to access the Centralised Database System remotely via terminals connected to the site; however all the data access and processing takes place at the central site. The following figure shows a Centralised DB System.

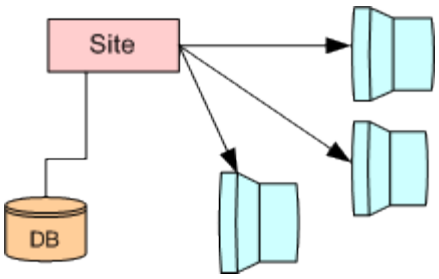


Figure1: Centralized Database System

Distributed database management system (DDBMS) In a DDS, database applications running at any of the system's sites should be able to operate on any of the database fragments transparently i.e., as if the data come from a single database managed by one DBMS. The software that manages a distributed database in such a way is called DDBMS.

The notion of **distributed database** is different from that of **decentralized database**. The latter does not imply sharing of data by a communication network. The former implies a collection of sites connected together with some kind of network and where each site has a database in its own right, but the sites work together as if data was stored at only one site.

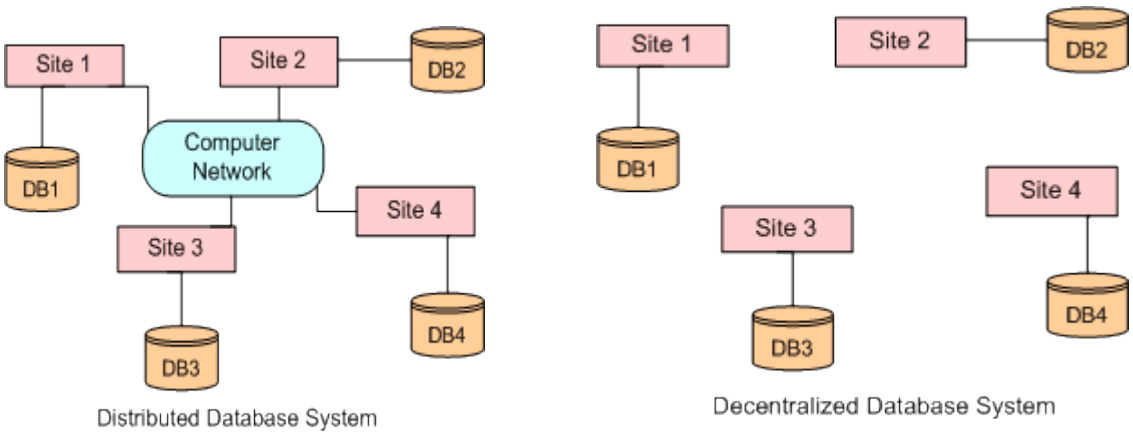


Figure 2: Distributed and Decentralized Database System

DISTRIBUTED DATABASE ARCHITECTURE

A distributed database system allows applications to access data from local and remote databases. In a homogenous distributed database system, each database is an Oracle Database. In a heterogeneous distributed database system, at least one of the databases is not an Oracle Database. Distributed databases use client/server architecture to process information requests. In this I explained the following terms:

- Homogenous Distributed Database Systems
- Heterogeneous Distributed Database Systems
- Client/Server Database Architecture

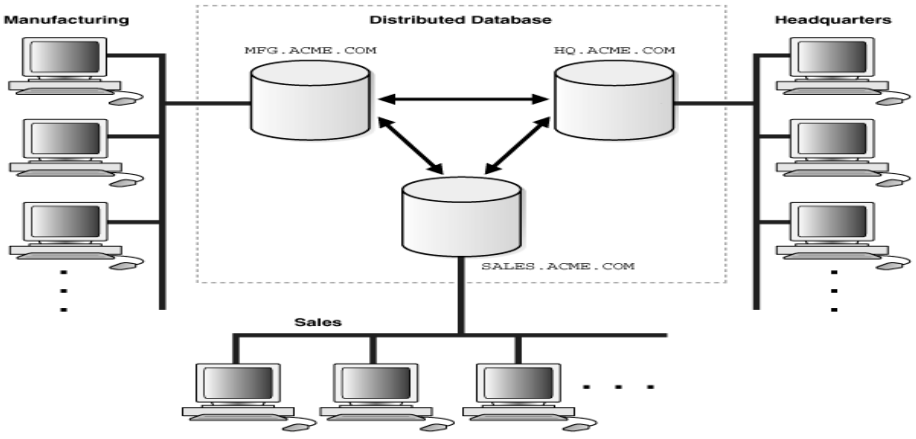
(a) Homogenous Distributed Database Systems: A homogenous distributed database system is a network of two or more Oracle Databases that reside on one or more machines. Figure-3 illustrates a distributed system that connects three databases: hq, mfg, and sales. An application can simultaneously access or modify the data in several databases in a single distributed environment. For example, a single query from a Manufacturing client on local database mfg can retrieve joined data from the products table on the local database and the dept table on the remote hq database.

For a client application, the location and platform of the databases are transparent. You can also create synonyms for remote objects in the distributed system so that users can access them with the same syntax as local objects. For example, if you are connected to database mfg but want to access data on database hq, creating a synonym on mfg for the remote dept table enables you to issue this query:

SELECT * FROM dept;

In this way, a distributed system gives the appearance of native data access. Users on mfg do not have to know that the data they access resides on remote databases.

Figure 3: Homogeneous Distributed Database



Distributed Databases Versus Distributed Processing:

The terms distributed database and distributed processing are closely related, yet have distinct meanings. There definitions are as follows:

- ***Distributed database:*** A set of databases in a distributed system that can appear to applications as a single data source.
- ***Distributed processing:*** The operations that occurs when an application distributes its tasks among different computers in a network. For example, a database application typically distributes front-end presentation tasks to client computers and allows a back-end database server to manage shared access to a database. Consequently, a distributed database application processing system is more commonly referred to as a client/server database application system.

Distributed database systems employ a distributed processing architecture. For example, an Oracle Database server acts as a client when it requests data that another Oracle Database server manages.

Distributed Databases Versus Replicated Databases:

The terms distributed database system and database replication are related, yet distinct. In a pure (that is, not replicated) distributed database, the system manages a single copy of all data and supporting database objects. Typically, distributed database applications use distributed transactions to access both local and remote data and modify the global database in real-time.

The term replication refers to the operation of copying and maintaining database objects in multiple databases belonging to a distributed system. While replication relies on distributed database technology, database replication offers applications benefits that are not possible within a pure distributed database environment.

Most commonly, replication is used to improve local database performance and protect the availability of applications because alternate data access options exist. For example, an application may normally access a local database rather than a remote server to minimize network traffic and achieve maximum performance. Furthermore, the application can continue to function if the local server experiences a failure, but other servers with replicated data remain accessible.

(b).Heterogeneous Distributed Database Systems:

In a heterogeneous distributed database system, at least one of the databases is a non-Oracle Database system. To the application, the heterogeneous distributed database system appears as a single, local, Oracle Database. The local Oracle Database server hides the distribution and heterogeneity of the data.

The Oracle Database server accesses the non-Oracle Database system using Oracle Heterogeneous Services in conjunction with an agent. If you access the non-Oracle Database data store using an Oracle Transparent Gateway, then the agent is a system-specific application. For example, if you include a Sybase database in an Oracle Database distributed system, then you need to obtain a Sybase-specific transparent gateway so that the Oracle Database in the system can communicate with it.

Alternatively, you can use generic connectivity to access non-Oracle Database data stores so long as the non-Oracle Database system supports the ODBC or OLE DB protocols.

Heterogeneous Services:

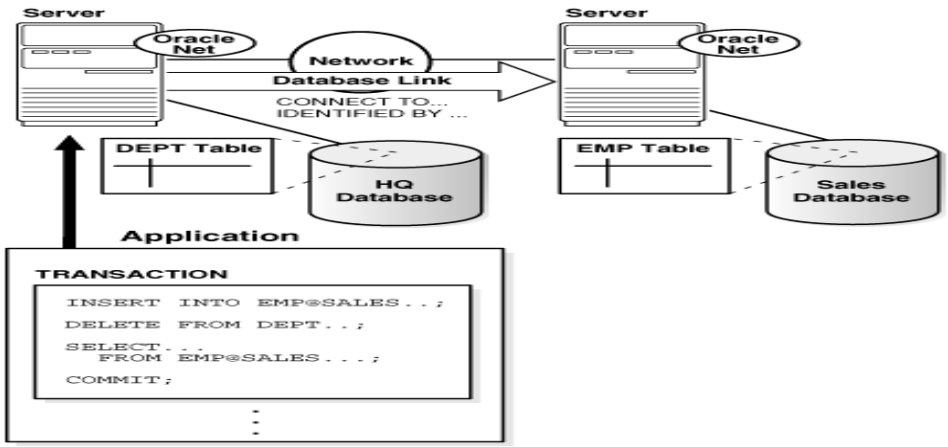
Heterogeneous Services (HS) is an integrated component within the Oracle Database server and the enabling technology for the current suite of Oracle Transparent Gateway products. HS provides the common architecture and administration mechanisms for Oracle Database gateway products and other heterogeneous access facilities. Also, it provides upwardly compatible functionality for users of most of the earlier Oracle Transparent Gateway releases.

(c.) Client/Server Database Architecture

A database server is the Oracle software managing a database, and a client is an application that requests information from a server. Each computer in a network is a node that can host one or more databases. Each node in a distributed database system can act as a client, a server, or both, depending on the situation.

In Figure-4 , the host for the hq database is acting as a database server when a statement is issued against its local data (for example, the second statement in each transaction issues a statement against the local dept table), but is acting as a client when it issues a statement against remote data (for example, the first statement in each transaction is issued against the remote table emp in the sales database).

Figure -4: An Oracle Database Distributed Database System



A client can connect directly or indirectly to a database server. A direct connection occurs when a client connects to a server and accesses information from a database contained on that server. For example, if you connect to the hq database and access the dept table on this database as in Figure-4 , you can issue the following:

SELECT * FROM dept;

This query is direct because you are not accessing an object on a remote database.

In contrast, an indirect connection occurs when a client connects to a server and then accesses information contained in a database on a different server. For example, if you connect to the hq database but access the emp table on the remote sales database as in Figure-4, you can issue the following:

SELECT * FROM emp@sales;

This query is indirect because the object you are accessing is not on the database to which you are directly connected.

DISTRIBUTED DATABASE DESIGN

The methodology used for the logical design of a centralized database applies to the design of the distributed one as well. However, for a distributed database **three** additional factors have to be considered.

Data Fragmentation: Before we decide how to distribute the data we must determine the logical units of distribution. The database may be broken up into logical units called fragments which will be stored at different sites. The simplest logical units are the tables themselves.

- ***Horizontal fragmentation:*** A horizontal fragment of a table is a subset of rows in it. So horizontal fragmentation divides a table 'horizontally' by selecting the relevant rows and these fragments can be assigned to different sides in the distributed system (for ex. Euston Road branch gets the fragment where myTable.branch ='Euston Road').
- ***Vertical fragmentation:*** a vertical fragment of a table keeps only certain attributes of it. It divides a table vertically by columns. It is necessary to include the primary key of the table in each vertical fragment so that the full table can be reconstructed if needed.
- ***Mixed fragmentation:*** in a mixed fragmentation each fragment can be specified by a SELECT-PROJECT combination of operations. In this case the original table can be reconstructed be applying union and natural join operations in the appropriate order.

Data Replication: A copy of each fragment can be maintained at several sites. Data replication is the design process of deciding which fragments will be replicated.

Data Allocation: Each fragment has to be allocated to one or more sites, where it'll be stored.

There are **three** strategies regarding the allocation of data:

- ***Fragmented (or Partitioned)*:** The database is partitioned into disjoint fragments, with each fragment assigned to one site (no replication). This is also called 'non-redundant allocation'.
- ***Complete replication:*** A complete copy of the database is maintained at each site (no fragmentation). Here, storage costs and communication costs for updates are most expensive. To overcome some of these problems, snapshots are sometimes used. A snapshot is a copy of the data at a given time. Copies are updated periodically.
- ***Selective replication:*** A combination of fragmentation and replication.

DATABASE LINKS

A database link is a connection between two physical database servers that allows a client to access them as one logical database.

What are Database Links:

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. For this connection to occur, each database in the distributed system must have a unique **global database name** in the network domain. The global database name uniquely identifies a database server in a distributed system.

Figure-5 shows an example of user scott accessing the emp table on the remote database with the global name hq.acme.com:

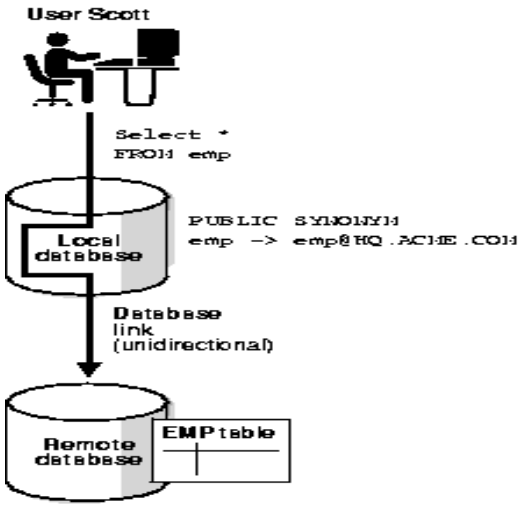


Figure-5: Database Links

Database links are either private or public. If they are private, then only the user who created the link has access; if they are public, then all database users have access.

Type of Link Description:

Connected user link: Users connect as themselves, which means that they must have an account on the remote database with the same username as their account on the local database.

Fixed user link: Users connect using the username and password referenced in the link. For example, if Jane uses a fixed user link that connects to the hq database with the username and password scott/tiger, then she connects as scott, Jane has all the privileges in hq granted to scott directly, and all the default roles that scott has been granted in the hq database.

Current user link: A user connects as a global user. A local user can connect as a global user in the context of a stored procedure--without storing the global user's password in a link definition. For example, Jane can access a procedure that Scott wrote, accessing Scott's account and Scott's schema on the hq database. Current user links are an aspect of Oracle Advanced Security.

Why Use Database Links:

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. In other words, a local user can access a link to a remote database without having to be a user on the remote database.

For example, assume that employees submit expense reports to Accounts Payable (A/P), and further suppose that a user using an A/P application needs to retrieve information about employees from the hq database. The A/P users should be able to connect to the hq database and execute a stored procedure in the remote hq database that retrieves the desired information. The A/P users should not need to be hq database users to do their jobs; they should only be able to access hq information in a controlled way as limited by the procedure.

Database links allow you to grant limited access on remote databases to local users. By using current user links, you can create centrally managed global users whose password information is hidden from both administrators *and* non-administrators. For example, A/P users can access the hq database as scott, but unlike fixed user links, scott's credentials are not stored where database users can see them.

Types of Database Links:

Oracle lets you create **private**, **public**, and **global** database links. These basic link types differ according to which users are allowed access to the remote database:

Type	**Owner**	**Description**
Private	User who created the link. View ownership data through: • DBA_DB_LINKS • ALL_DB_LINKS • USER_DB_LINKS	Creates link in a specific schema of the local database. Only the owner of a private database link or PL/SQL subprograms in the schema can use this link to access database objects in the corresponding remote database.
Public	User called PUBLIC. View ownership data through views shown above.	Creates a database-wide link. All users and PL/SQL subprograms in the database can use the link to access database objects in the corresponding remote database.
Global	User called PUBLIC. View ownership data through views	Creates a network-wide link. When an Oracle

	shown above.	network uses Oracle Names, the names servers in the system automatically create and manage global database links for every Oracle database in the network. Users and PL/SQL subprograms in any database can use a global link to access objects in the corresponding remote database.

Distributed Database Administration:

The following sections explain some of the topics relating to database management in an Oracle distributed database system:

(a). Site autonomy: means that each server participating in a distributed database is administered independently from all other databases. Although several databases can work together, each database is a separate repository of data that is managed individually. It provides the benefits like Nodes of the system can mirror the logical organization of companies or groups that need to maintain independence.

(b). Distributed Database Security: Oracle supports all of the security features that are available with a nondistributed database environment for distributed database systems, including: Password authentication for users and roles, Login packet encryption for client-to-server and server-to-server connections.

(c). Auditing Dtabase Links: You must always perform auditing operations locally. That is, if a user acts in a local database and accesses a remote database through a database link, the local actions are audited in the local database, and the remote actions are audited in the remote database--provided appropriate audit options are set in the respective databases.

(d).Adminstration Tools: The database administrator has several choices for tools to use when managing an Oracle distributed database system:

- ***Enterprise Manager:*** Enterprise Manager is Oracle's database administration tool that provides a graphical user interface (GUI). Enterprise Manager provides administrative functionality for distributed databases through an easy-to-use interface. You can use

Enterprise Manager to Administer multiple databases, Centralize database administration tasks.

- ***Third Party Administration Tools:*** Currently more than 60 companies produce more than 150 products that help manage Oracle databases and networks, providing a truly open environment.
- ***SNMP Support:*** Besides its network administration capabilities, Oracle **Simple Network Management Protocol** (*SNMP*) support allows an Oracle database server to be located and queried by any SNMP-based network management system.

TRANSACTION PROCESSING IN DISTRIBUTED SYSTEM

A transaction is a logical unit of work constituted by one or more SQL statements executed by a single user. A transaction begins with the user's first executable SQL statement and ends when it is committed or rolled back by that user.

A **remote transaction** contains only statements that access a single remote node. A **distributed transaction** contains statements that access more than one node.

Transaction processing is designed to maintain a computer system (typically a database or some modern file systems) in a known, consistent state, by ensuring that any operations carried out on the system that are interdependent are either all completed successfully or all canceled successfully.

For example, consider a typical banking transaction that involves moving $700 from a customer's savings account to a customer's checking account. This transaction is a single operation in the eyes of the bank, but it involves at least two separate operations in computer terms: debiting the savings account by $700, and crediting the checking account by $700. If the debit operation succeeds but the credit does not (or *vice versa),* the books of the bank will not balance at the end of the day. There must therefore be a way to ensure that either both operations succeed or both fail, so that there is never any inconsistency in the bank's database as a whole. Transaction processing is designed to provide this.

Transaction processing allows multiple individual operations to be linked together automatically as a single, indivisible transaction. The transaction-processing system ensures that either all operations in a transaction are completed without error, or none of them are. If some of the operations are completed but errors occur when the others are attempted, the transaction-processing system “rolls back” *all* of the operations of the transaction (including the successful ones), thereby erasing all traces of the transaction and restoring the system to the consistent, known state that it was in before processing of the transaction began. If all

operations of a transaction are completed successfully, the transaction is committed by the system, and all changes to the database are made permanent; the transaction cannot be rolled back once this is done.

Transaction processing guards against hardware and software errors that might leave a transaction partially completed, with the system left in an unknown, inconsistent state. If the computer system crashes in the middle of a transaction, the transaction processing system guarantees that all operations in any *uncommitted* (*i.e.,* not completely processed) transactions are cancelled.

Transactions are processed in a strict chronological order. If transaction *n+1 intend*s to touch the same portion of the database as transaction *n,* transaction *n+1* does not begin until transaction *n* is committed. Before any transaction is committed, all other transactions affecting the same part of the system must also be committed; there can be no "holes" in the sequence of preceding transactions.

DISTRIBUTED QUERY OPTIMIZATION

Distributed query optimization is an Oracle feature that reduces the amount of data transfer required between sites when a transaction retrieves data from remote tables referenced in a distributed SQL statement.

Distributed query optimization uses Oracle's cost-based optimization to find or generate SQL expressions that extract only the necessary data from remote tables, process that data at a remote site or sometimes at the local site, and send the results to the local site for final processing. This operation reduces the amount of required data transfer when compared to the time it takes to transfer all the table data to the local site for processing.

Using various cost-based optimizer hints such as DRIVING_SITE, NO_MERGE, and INDEX, you can control where Oracle processes the data and how it accesses the data.

CONCLUSION

This paper focused on the design, architecture and how trasanctions being processed in distributed database. The transaction-processing system ensures that either all operations in a transaction are completed without error, or none of them are. Whereas query processing distributed database reduces the amount of data transfer required between sites when a transaction retrieves data from remote tables referenced in a distributed SQL statement. So distributed database increase reliability and availability, protect valuable data and reflect organizational structure.

REFERENCES

1. B. G. Lindsey *et al:* "Notes on Distributed Databases", IBM Research Report RJ2571 (July 1979)
2. C. J. Date: "What is a Distributed Database System?" in Relational Database writings 1985-1989, reading, Mass:Addision-Wesley (1990).
3. Distributed Databases by Tata McGraw-Hill Education, 1988
4. David Bell and Jane Grimson: Distributed Database Systems, Reading, Mass: Addison-Wesley (1992).
5. Elmasri and Navathe, *Fundamentals of database systems* (3rd edition), Addison-Wesley Longman, ISBN 0-201-54263-3
6. IBM Corporation: Distributed Databases Architecture Reference, IBM form No. SC26-4651.
7. J. B. Rothnie Jt. Et al: "Introduction to a system for Distributed Databases (SDD-1)", ACM TODS 5, No. 1 (March 1980)
8. M. T. Ozsu and P. Valduriez, *Principles of Distributed Databases* (2nd edition), Prentice-Hall, ISBN 0-13-659707-6
9. O'Brien, J. & Marakas, G.M.(2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin
10. Phipil A. Bernstein *et al:* « Query Processing in a system for Distributed Databases (SDD-1) », ACM TODS 6, No. 4 (December 1981).
11. Rob Goldring: A Discussion of Relational Database Raplication Tachnology", infoDB 8, No. 1 (Spring 994)
12. Stefano Ceri and Giuseppe Pelagatti: Distributed Databases: Principles and Systems, New York, N. Y.: McGrawHill (1984).
13. Yuri Breitbart, Hector Gracia-Molina and Avi Silberschatz: "Overview of Multi-Database Transaction Management," The VLDB Journal, No 2(October 1992)