MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

RELATIONAL DATABASE MANAGEMENT SYSTEM

Unit 11

Distributed Databases

Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction	-	-	4
	1.1 Objectives	-	-	
2	Introduction of Distributed Databases	-	1, I	5-9
	2.1 DDBMS architectures	1, 2	-	
	2.2 Functions of distributed database management system	-	-	
	2.3 Components of distributed database management system	-	-	
3	Homogeneous and Heterogeneous Database	-	2	10-11
4	Distributed Data Storage	-	3, II	11-18
	4.1 Data fragmentation	1, 2, 3, 4, 5, 6, 7	-	
	4.2 Data replication	-	-	
5	Advantages and Disadvantages of Data Distribution	-	4	19-20
	5.1 Advantages of data distribution	-	-	
	5.2 Disadvantages of data distribution	-	-	
6	Distributed Transaction	-	5	21
7	Commit Protocols	-	6	22-23
	7.1 Components of atomic commit	-	-	
	7.2 Two phase commit	-	-	
8	Concurrency Control	-	7	23
9	Recovery of Distributed Database	-	8	24-25
10	Directory Systems	-	9	25
11	DDBMS Transparency Features	-	10	26
12	Distribution Transparency	8	11	27-28

13	Summary	-	-	28
14	Glossary	-	-	29
15	Terminal Questions	-	-	29
16	Answers	-	-	30-31
17	References	-	-	31

1. INTRODUCTION

In the previous unit, you studied about Object Oriented DBMS. You read about the various OODBMS architectural approaches. You also read about object identity, procedures, encapsulation, relationship, identifiers and inheritance. You became familiar with basic interface and class structure, type hierarchies, type extent, persistent programming languages and OODBMS storage issues. In this unit we will study about distributed databases.

Distributed database technology is expected to have a significant impact on data processing in the upcoming years. With the introduction of commercial products, expectations are that distributed database management systems will by and large replace centralised ones within the next decade.

This unit explains the role, technologies, and unique database design features of distributed databases. The goal and trade-offs for distributed databases, data replication uses, advantages and disadvantages of distribution databases, and distributed transaction and data storage are covered. This unit contains a thorough coverage of database concurrency and recovery in data distribution.

1.1 Objectives

After studying this unit, you should be able to:

- *Explain the DDBMS Architecture*
- *Discuss homogeneous and heterogeneous database*
- *Explain distributed data storage*
- *Explain data fragmentation and data replication*
- *Identify and demonstrate advantages and disadvantages of data distribution*
- *Explain distribution transaction*
- *Discuss commit protocols*
- *Demonstrate concurrency control and recovery in distributed databases*
- *Identify directory system*
- *Explain distributed database transparency features*
- *Discuss distribution transparency*

2. INTRODUCTION OF DISTRIBUTED DATABASES

A database that physically resides entirely on one machine under a single DBMS is known as local database management system. Database management system that resides entirely on a machine different from that of the user connected through a network is known as remote database.

In either case, the entire database is controlled by a single site and hence is known as Centralised Database System. In contrast to this a database may be fragmented and each of its fragments is stored on different machines connected through network(s) or is controlled by different DBMSs or operates under different operating systems. Such a multiple-source and multiple-location database is called distributed database.

Usually, distributed database is a set of several logically consistent databases which are spread over a computer network. Distributed Database Management System (DDBMS) is software system that handles the distributed database system for making the distribution clear to user.

The user is not aware about the database which is fragmented. A Distributed Database Management System makes sure that the users access the distributed database.

2.1 DDBMS Architectures

Generally, the distributed database contains a pool of sites, every one of which has a local database system (Figure 11.1). Each one of the sites is capable of processing local transactions; i.e. the transactions which access data in that particular site only. Additionally, a site may participate in the processing of overall transactions, those transactions that access data is various sites. The processing of universal transactions needs communication between the sites usually through a network.

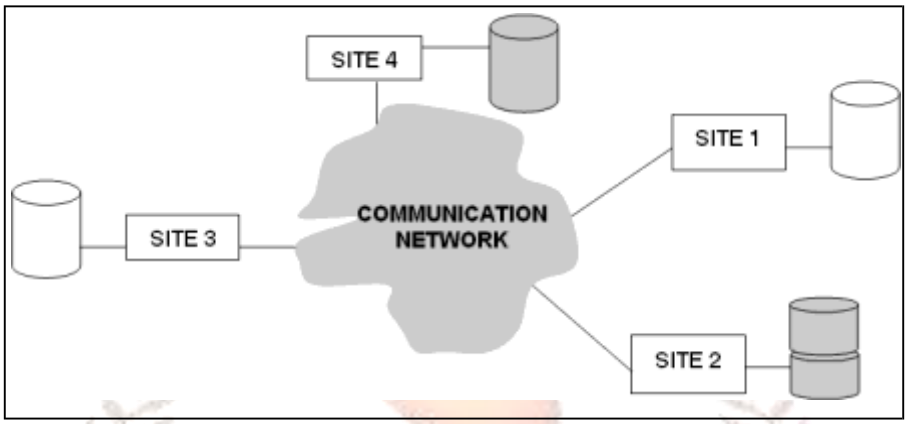

Fig 11.1: Distributed Database Architecture

The sites within system can be linked physically in numerous ways. Different topologies are shown as graphs whose nodes match up to sites. Direct connection between the two sites is given by a link from node 'A' to node 'B'.

Precisely how a database is distributed is known from its configuration and how they differ from each other is shown in following aspects:

- ***Installation Cost:*** It is the cost of connecting the sites physically in system.
- ***Communication Cost:*** Cost of time required and total expenditure to transmit a message from site 'A' to site 'B'.
- ***Reliability:*** Reliability means the frequency of failures in link or sites.
- ***Availability:*** The level at which data could be accessed in spite of the malfunction of some sites or links.

These differences perform a vital part in selecting the suitable mechanism for managing the allocation of data.

The contributing or collaborating sites of one distributed database might be spread physically on a huge geographical region through networks. For example,

- the all-Indian state capitals or a small physical region
- one solo building or many adjoining buildings

The first type of network is known as wide area network, while the latter is known as a local area network.

The links of a network between its nodes may be of different patterns known as its topology. Some of the network topologies are depicted in Figure 11.2.

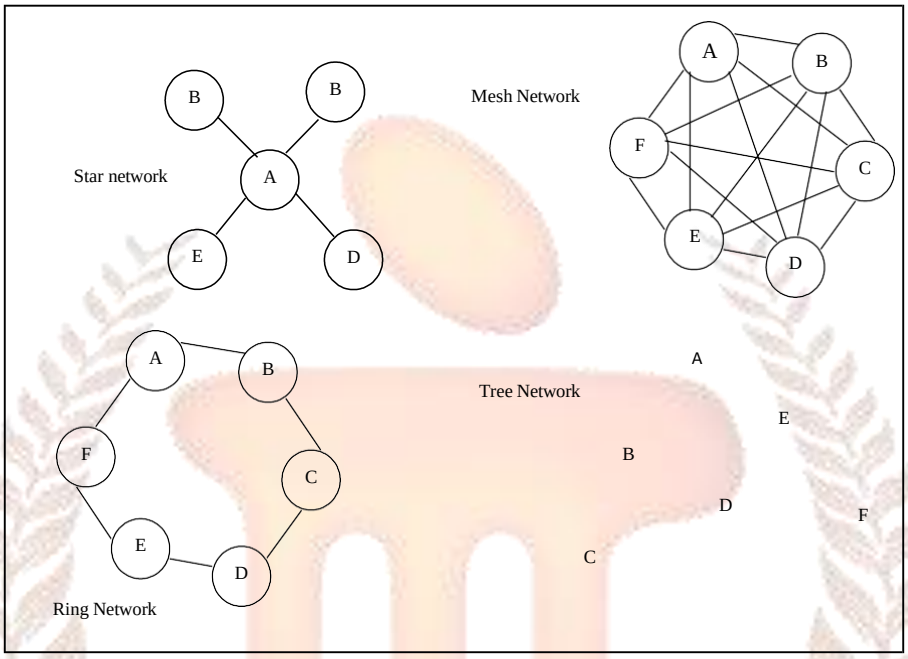

Fig 11.2: Network Topologies

2.2 Functions of Distributed Database Management System

Distribution results in enhanced complexity in system design and implementation. To obtain the maximum advantages of DDBMS; the DDBMS software should be able to offer the following functions besides those of a centralised DBMS:

- ***Tracking of data:*** The ability to track data distribution, replication and fragmentation by enlarging the DDBMS catalogue.
- ***Distributed query processing:*** The capability to transmit queries and access remote sites between different sites via communication network.
- ***Distributed transaction management:*** Capability to plan implementation strategies for transaction processes and queries from multiple sites and to coordinate access to data and sustain reliability of overall database.
- ***Replicated data management:*** The capability to choose which copy of one copied data to access and to sustain the reliability of copy of replicated data.

- ***Distributed database recovery:*** The capability to recover data from case-by-case crashes and from different types of malfunctions, for example; breakdown of communication link.
- ***Security:*** Distributed transactions should be carried out with adequate security management of data and the access/authorisation rights of users.

2.3 Components of Distributed Database Management System

A DDBMS has a lot of components linked together. Some of the components that a DDBMS should have are:

- ***Sites or nodes (workstations):*** The end users machines (mostly PCs) that form the network. The distributed database system is independent of the hardware of the workstations.
- ***Network hardware and software:*** Each workstation should have essential hardware and software that allow them to establish a network with other components on the distributed database system. The DDB system should be independent of the network type of each workstation.
- ***Transaction processor (TP):*** Every data-requesting workstation should have this software component that receives & processes the request for data (local or remote). Data access is transparent to the user. Transaction Processor is also called Transaction Manager (TM) or Application Processor (AP).
- ***Data processor (DP):*** It is a software component on every computer in the distributed database system. This component stores & retrieves data located on that single site. It is also known as Data Manager (DM).

SELF-ASSESSMENT QUESTIONS – 1

1. Distributed database system consists of a collection of _____, every one of which has a local database system.
2. The consumer is aware about the database which is fragmented. (True/ False)
3. The most widespread channels are _____ base band coaxial, fibre optics and broadband coaxial.
4. Which of the following is the basic component of Distributed Database Management System?
   a) Data Processor
   b) Data Definition
   c) Data warehousing
   d) Data Mining

Activity 1

Analyse how DDB system is independent of the network type of each workstation.

3. HOMOGENEOUS AND HETEROGENEOUS DATABASE

When the database technology is the same at each of the locations and the data at the several locations are also compatible, that data is known as homogenous database. The following conditions should exist for homogeneous database:

- The operating system used at each of the locations is the same or at least they should be extremely compatible.
- The data models used at every location should be the same.
- The database management systems used at every location should be the same or at least they should be extremely compatible.
- The data at the different locations should have common definitions and formats.

Homogenous database make the sharing of data between the different users simpler. This signifies the design goal for the distributed database. Achieving this objective needs a very high level planning during the planning phase.

Heterogeneous database: In heterogeneous DDBMS, every one of the site might manage different types of DBMS wares, which does not need to be established on the similar original data model and so the system should be made of RDBMS, OODBMS & ORDBMS products.

- In heterogeneous database, contact among various DBMS is needed for translations.
- So as to give DBMS transparency, users should be able to make requests in DBMS language at the local site.
- Data from other sites might have a variety of hardware, diverse DBMS products and mixture of a variety of hardware & DBMS products.
- The job of finding these data & executing any essential translation are the capabilities of the heterogeneous DDBMS.

SELF-ASSESSMENT QUESTIONS – 2

5. Homogenous database allows sharing of data among different simpler.
6. In which type of database, communication between various DBMS is required for translations?
   a) Relation Database
   b) Heterogeneous Database
   c) Flat-file database
   d) Operational Database

4. DISTRIBUTED DATA STORAGE

A distributed data store means either the Distributed Database where users store their information on a number of nodes, or a network in which a user stores their information on a *number of peer network nodes.* It provides the following functions:

- Replication
  - ❖ System keeps several copies of data, stored at various sites, for quick retrieval fault tolerance.
- Fragmentation
  - ❖ Relation is divided into various fragments stored at separate sites
- Replication & fragmentation can be united
  - ❖ Relation is divided into various fragments: system keeps and maintains several similar replicas of each of these fragments.

Data Fragmentation and Data Replication are explained in detail below:

4.1 Data Fragmentation

It is clear that in the distributed database system the database is broken into smaller pieces. Here we will discuss the techniques that are used to break up the database into logical units, known as fragments, which may be assigned for storage at the various sites.

If a relation N is fragmented, N is split into numerous fragment relations N1, N2…... Nn. These fragments consists enough information to rebuild the primary relation N. Such reconstruction could happen by the application of union operation or also by one special kind of join operation on different fragments depending on how they were obtained from the original relation. Of many methods of fragmentation, two of them shall be discussed here: horizontal fragmentation & vertical fragmentation.

For illustration purposes, let us consider the customer relation CUSTOMER of some company:

CUSTOMER (CUS_ID, CUS_NAME, CUS_STATE, CUS_ DEPOSIT, CUS_BALANCE, CUS_RATING, CUS_DUE)

A sample instance of the CUSTOMER relation is shown in Table 11.1 below:

Table 11.1: Sample Customer Relation

CUSTOMER	CUS_ID	CUS_NAME	CUS_STATE	CUS_DEPOSIT	CUS_BALANCE	CUS_RATING	CUS_DOE
	10	Puranchand	Haryana	3000	2000	3	1000
	11	Rohit	Punjab	4000	3000	2	1500
	21	Ramlal	Haryana	2000	190	3	280
	23	Pankaj	Bihar	2300	230	3	320
	33	Rahul	Punjab	3300	450	2	400
	43	Satbir	Haryana	4500	1000	1	900

Horizontal fragmentation: Under this fragmentation scheme, a able (or relation) N is partitioned and subsets, N1, N2, N3 ……..are created. Each subset Ni (i=1, 2, 3…) are composed of a number of tuples (of relation N). Every tuple of relation N should be of the fragments, in order that the primary relation can be rebuild when needed.

Fragment might be described as one selection on global relation N. That is, the union of all the fragments must be able to generate the original relation.

In our sample relation CUSTOMER, suppose that each state headquarters requires data belonging to that state only. Therefore, the relation can be horizontally fragmented as shown in table 11.2 below:

Table 11.2: Horizontally fragmented Relation

Fragment Name	Location of fragment	Selection Condition	Node name	Customer_IDs	Number of row
CUS_BHR	Patna	CUS_STATE="Bihar"	BHS	23	1
CUS_HAR	Hisar	CUS_STATE="Haryana"	HRS	10,21,43	3
CUS_PUN	Amritsar	CUS_STATE="Punjab"	PNS	11,33	2

The three resulting fragment relations are shown in Table 11.3 below:

Table 11.3: Three Resulting Fragment Relations

Fragment Name:	CUS_BHR	Location: Patna		Node:BHS		
CUS_ID	CUS_NAME	CUS_STATE	CUS_DEPOSIT	CUS_BALANCE	CUS_RATING	CUS_DUE
23	Pankaj	Bihar	2300	230	3	320

Fragment Name:	CUS_HAR	Location: Hisar		Node: HRS		
CUS_ID	CUS_NAME	CUS_STATE	CUS_DEPOSIT	CUS_BALANCE	CUS_RATING	CUS_DUE
10	Puranchand	Haryana	3000	2000	3	1000
21	Ramlal	Haryana	2000	190	3	280
43	Satbir	Haryana	4500	1000	1	900

Fragment Name:	CUS_PUN	Location: Amritsar		Node: PNS		
CUS_ID	CUS_NAME	CUS_STATE	CUS_DEPOSIT	CUS_BALANCE	CUS_RATING	CUS_DUE
11	Rohit	Punjab	4000	3000	2	1500
33	Rahul	Punjab	3300	450	2	400

Vertical fragmentation: Vertical fragmentation is similar as decomposition. Vertical fragmentation of a relation or a table can be acquired by dividing the table into a number of sub-tables having disjoint columns.

Relation N can be reconstructed from the fragments by taking the natural join operation. Suppose, now that the company is divided into two departments – customer department and collection department. The two departments are concerned with their respective data only. Therefore, the relation CUSTOMER can be vertically fragmented into two fragments as given in Table 11.4 below:

Table 11.4: Example of Vertical Fragmentation

Fragment name	**Location**	**Node name**	**Attributes**
CUS_DEPT	Customer Office	CUS	CUS_ID, CUS_NAME, CUS_STATE
COL_DEPT	Collection Office	COL	CUS_ID, CUS_DEPOSIT, CUS_BALANCE, CUS_RATING, CUS_DUE

The resulting two fragment relations are given in Table 11.5:

Table 11.5: Resulting Fragment Relations

Fragment name: CUS_DEPT Location: Customer Office Node:CUS		
CUS_ID	***CUS_NAME***	***CUS_STATE***
10	Puranchand	Haryana
11	Rohit	Punjab
21	Ramlal	Haryana
23	Pankaj	Bihar
33	Rahul	Punjab
43	Satbir	Haryana

Fragment name: COL_DEPT		Location: Collection Office		Node: COL
CUS_ID	***CUS_DEPOSIT***	***CUS_BALANCE***	***CUS_RATING***	***CUS_DUE***
10	3000	2000	3	1000
11	4000	3000	2	1500
21	2000	190	3	280
23	2300	230	3	320
33	3300	450	2	400
43	4500	1000	1	900

Usually, vertical fragmentation can be achieved by addition of a particular attribute called 'tuple-id' to the scheme N (CUS_ID in our case). A 'tuple-id' is a logical or physical address for one tuple. As each tuple in N must have an exclusive address, the 'tuple-id' feature is key for augmented scheme.

To rebuild the initial deposit relation from fragments, we calculate

CUSTOMER = (CUS_DEPT_ COL_DEPT)

Note that the phrase (CUS_DEPT COL_DEPT) is unique form of the natural join. The join trait is CUS_ID. Since the tupled-value constitutes an address, it's possible to join a tuple of CUS_DEPT with matching tuple of COL_DEPT by means of address provided by the CUS_ID value. This address permits straight retrieval of the tuple with no need for the index. Therefore, this natural join could be worked out much more effectively than the typical natural joins.

Even though, the tuple-id characteristic is essential in the execution of vertical portioning, it is vital that this trait is not seen by the users. If the users are provided access to tuple-ids, then it is impossible for the system to modify tuple addresses. In addition, the availability of the internal addresses goes against the concept of data independence, which is one of the major qualities of relational model.

Mixed fragmentation: A relation can also be fragmented both vertically and horizontally. In such cases both fragmentation norms are specified. Suppose in our example of the CUSTOMER relation, we need each department data individually in two different offices in the state headquarters. The required fragments will be as shown in table 11.6 below:

Table 11.6: Example of Mixed Fragmentation

Fragment name	Location	Horizontal criterion	Node name	Attributes
CUS_BHR_CUS	Patna	CUS_STATE="Bihar"	BHRCUS	CUS_ID, CUS_NAME, CUS_STATE
CUS_BHR_COL	Gaya	CUS_STATE="Bihar"	BHRCOL	CUS_ID, CUS_DEPOSIT, CUS_BALANCE, CUS_DUE
CUS_HAR_CUS	Hisar	CUS_STATE="Haryana"	HARCUS	CUS_ID, CUS_NAME, CUS_STATE
CUS_HAR_COL	Karnal	CUS_STATE="Haryana"	HARCOL	CUS_ID, CUS_DEPOSIT, CUS_BALANCE, CUS_DUE
CUS_PUN_CUS	Amritsar	CUS_STATE="Punjab"	PUNCUS	CUS_ID, CUS_NAME, CUS_STATE
CUS_PUN_COL	Bhatinda	CUS_STATE="Punjab"	PUNCOL	CUS_ID, CUS_DEPOSIT, CUS_BALANCE, CUS_DUE

The resulting fragments are shown in Table 11.7.

Table 11.7: Resulting Fragments

Fragment name: CUS_BHR_CUS Location: Patna Node:BHRCUS		
CUS_ID	***CUS_NAME***	***CUS_STATE***
23	Pankaj	Bihar

Fragment name: CUS_BHR_COL		**Location: Gaya**		**Node: BHRCOL**
CUS_ID	***CUS_DEPOSIT***	***CUS_BALANCE***	***CUS_RATING***	***CUS_DUE***
23	2300	230	3	320

Fragment name: CUS_HAR_CUS Location: Hisar Node:HARCUS		
CUS_ID	**CUS_NAME**	**CUS_STATE**
10	Puranchand	Haryana
21	Ramlal	Haryana
43	Satbir	Haryana

Fragment name: CUS_HAR_COL		**Location: Karnal**		**Node: HARCOL**
CUS_ID	***CUS_DEPOSIT***	***CUS_BALANCE***	***CUS_RATING***	***CUS_DUE***
10	3000	2000	3	1000
21	2000	190	3	280
43	4500	1000	1	900

Fragment name: CUS_PUN_CUS Location: Amitsar Node:PUNCUS		
CUS_ID	***CUS_NAME***	***CUS_STATE***
11	Rohit	Punjab
33	Rahul	Punjab

Fragment name: CUS_PUN_COL		**Location: Bhatinda**		**Node: PUNCOL**
CUS_ID	***CUS_DEPOSIT***	***CUS_BALANCE***	***CUS_RATING***	***CUS_DUE***
11	4000	3000	2	1500
33	3300	450	2	400

4.2 Data Replication

In easy words, Replication is making a copy of the relation. When a relation N is modified or replicated, a copy of relation N is stored in other sites. The copies may be kept at only a few selected sites or each site may keep a copy. In case each site of the system has a copy of the relation, it is known as full replication.

Replication comes in helpful, when you want to improve the accessibility of data. Most severe case would be to replicate whole database at every site of distributed system, which will result in creating a completely replicated distributed database. This improves accessibility

to a great degree because the system will keep on operating even if there is only one site working.

It moreover improves execution of retrieving global queries, as the effect of such a query can be obtained from any one of the local sites; therefore, a retrieval query could be worked at the local site at which it is submitted, condition being the site should include one server module.

One drawback of full replication is; it could decrease update operations radically, as one single logical update should be executed on every copy of the database to keep them reliable. This especially applies if there are many copies of database.

Full Replication makes the recovery techniques and concurrency control much expensive then if there was no replication.

SELF-ASSESSMENT QUESTIONS – 3

7. Horizontal fragmentation splits the relation by allotting each node of N to one fragment. (True/ False)
8. Vertical fragmentation of a relation or a table can be acquired by splitting the table into a number of sub-tables having disjoint columns. (True/ False)
9. _____ makes concurrency control and recovery techniques more expensive.

Activity 2

Examine how the information concerning data fragmentation and replication is stored in a global directory.

5. ADVANTAGES AND DISADVANTAGES OF DATA DISTRIBUTION

In the following sections we will discuss about the benefits and drawbacks of data distribution.

5.1 Advantages of Data Distribution

Distributed database systems have a number of advantages over their centralised counterparts. The primary goal of distributed database systems is to achieve the capacity to access and share data stored in databases spread across different machines, operating systems and DBMSs, in reliable, fast and efficient method. The benefits of distributed database are explained below:

- ***Space independence:*** If various sites are linked to one another, then the user at a site might access data that is present at some other site. The user does not have to be present physically at the database site. Therefore, the database becomes space independent.
- ***Availability of data where it is required:*** The data in the distributed database system are so dispersed as to match the data needs of the users.
- ***Faster data access:*** The end-users use only a subset of the whole database. If this section of the database is locally stored and accessed, it will be many times quicker than when remotely located.
- ***Distributed control:*** The main advantage of achieving data sharing by the method of the data distribution is that each of the sites is capable of keeping some degree of control on the stored data.
- ***User-friendly interface:*** The end users are free to have interfaces of their own choice at their sites.
- ***Increased Reliability:*** In case of a centralised database system, a failure renders the entire system useless. Such is not the case with the distributed database systems. Even in case of a failure the end users can still access their own database stored locally.
- ***Query speedup:*** If there is a query involving data at different sites, it might be possible to break query into sub-queries that could be parallelly executed by different sites. This parallel computation enables quicker processing of a user's query.

5.2 Disadvantages of Data Distribution

Distributed database systems are not entirely free from limitations. The main drawback of distributed database systems is its extra complexity needed to make sure proper coordination between the sites. Such increased complexity is in the shape of:

- ***Complexity of management and control:*** All the related management activities and control of the same becomes very complex with degree of distribution.
- ***Cost of software development:*** It is more costly and also tough to implement a distributed database system as compared to centralised local database.
- ***Higher possibility of bugs:*** As the sites included in the distributed system work in parallel, it is more difficult to assure the accuracy of the algorithms. This mode of operation makes them very susceptible to bugs. The field of creating perfect distributed algorithms is still a big area or research.
- ***Processing overheads:*** The expenditures needed in the exchange of messages, data and additional computing to accomplish coordination in distributed database systems is not there in centralised systems.

SELF-ASSESSMENT QUESTIONS – 4

10. The _____ in them distributed database system are so scattered as to match the requirements of the users.
11. What is the primary drawback of distributed database systems?
    a) complexity
    b) standards
    c) overheads
    d) control

6. DISTRIBUTED TRANSACTION

The updation of data on more than two networked computer systems is termed as a distributed transaction. Distributed transactions expand the advantages of transactions to those applications that should update the distributed data.

Applying effective applications is not easy since these applications encounter various failures, such as client failure, server failure, and the network failure between server and client & server. Application program by itself has to identify and recover in absence of the distributed transactions.

In distributed transactions, every computer system is allocated one transaction manager. When one transaction accomplishes work at various computers, transaction managers interact with one another through a subordinate or superior relationship. These associations are valid only for one specific transaction.

Each manager does all the organising, enlistment, and aborting of calls for its listed resource managers (typically those that exist in that specific computer). Resource managers handle constant data and perform in collaboration with Distributed Transaction Coordinator (DTC) to guarantee isolation & atomicity to the application.

In each distributed transaction, every one of the contributing component should agree to execute a change action (for example database update) before execution of the transaction. The DTC achieves the transaction co- ordination role for the involved components.

When executing a distributed transaction between many computers, the transaction manager transmits prepare/organises, commits and aborts messages to all of its assistant transaction managers.

SELF-ASSESSMENT QUESTIONS – 5

12. For distributed transactions, each computer does not require to have local transaction manager. (True/ False)
13. _____ manage constant or durable data and work in cooperation with the Distributed Transaction Coordinator.

7. COMMIT PROTOCOLS

Commit protocols are utilised to make sure atomicity across sites. A transaction is said to be distributed across 'n' number of processes. Every process can itself choose to abort or commit the transaction, butthe transaction must either commit or abort on all sites.

The *two phase commit* (2PC) protocol is extensively used.

The three phase commit (3PC) protocol is much more complex and costly, but has several advantages over 2PC. This protocol is generally not used in practical life.

7.1 Components of Atomic Commit

Termination protocol

- When a site is unsuccessful, the correct sites should still be capable to decide on the result of pending transactions.
- To make a decision on all pending transactions they execute a termination protocol.

Recovery

- When a site is unsuccessful or fails and then you restart, it has to perform recovery for all transactions that it has not yet committed suitably.
- Single site recovery, must terminate all transactions that were active at the time of the failure.
- Distributed system, must ask around; possibly an active transaction was committed in the remaining system.
- Independent recovery, a site does not have to communicate with other sites at restart.

7.2 Two Phase Commit

- Two phase commit is one transaction protocol intended for those complexities that happen with distributed resource managers.
- Two phase commit protocol is majorly utilised in stock market transactions, credit card systems, hotel and airline reservations, and banking applications.
- Distributed transaction manager with two phase commit protocol uses a coordinator to handle individual resource managers.

SELF-ASSESSMENT QUESTIONS – 6

14. _____ are utilised to make sure atomicity across sites.
15. _____ is a transaction protocol which is used for the complexities that are associated with distributed resource managers.

8. CONCURRENCY CONTROL

Concurrency control technique is being used in distributed database environment. We presume that every site contributes in the working of a commit protocol to make sure global transaction atomicity. We presume all copies of any of the item are up to date.

- ***Locking protocols:*** We present some possible schemes that are applicable to an environment where data can be replicated in numerous sites.
- ***Single lock manager approach:*** System has a single lock manager existing in a single site, say $\mathbf{S_i}$. Whenever a transaction wants to lock an item, it transmits one lock request to $\mathbf{S_i}$ and lock manager decides if the lock could be granted right away. If the answer is yes, the manager transmits a message to that site which has started the request. If the answer is no, request is held over till it can be permitted, and at that time a message is transmitted to the starting site.
- ***Distributed lock manager:*** In this technique, locking functionality is executed by lock managers in the entire site. Lock managers have the responsibility to manage access to local data items. But particular protocols may be used for replicas.

SELF-ASSESSMENT QUESTIONS – 7

16. A transaction is distributed across _____.
17. System maintains a single _____ that exists in a single chosen site.

9. RECOVERY OF DISTRIBUTED DATABASE

When one distributed database node fails, its main memory contents get vanished, and should somehow be recovered. Usually, the most common technique is to keep, on stable storage, a log of all updates performed on the node's data. When a node improves, it reads all the log entries in the database.

While the log can grow large at random, logging is almost always going with the taking of the periodic checking points. This check pointing decreases the time needed for recovery, since only the log records postdating the most recent checkpoint have to be read through the recovery process.

In distributed databases, the consistency of the recovered data should also be considered. If instant consistency is compulsory, once the node is reintegrated in the system, its data should be consistent with the data on the other nodes.

If only ultimate consistency is essential, the reintegrated node should still be in a state that is finally uniform, i.e., the state would develop into consistent subsequent to a decided time period in a system.

Diskless distributive recovery: For various embedded systems, for example those working in environments with electromagnetic radiation or intense vibrations, it might not be practical to utilise disks as permanent storage for checkpoints and logs. Moreover, many disk drivers are unpredictable, which makes them inappropriate for real-time systems. Additionally, disks add to the cost of building the system. We thus see a requirement for recovery mechanisms which are not dependent on the disks.

In the replicated database approach, the system's inherent redundancy can be utilised for recovery. Rather than reading a checkpoint from disk, the contents of another node in the system (the recovery source) can be copied to the recovering node (the recovery target). If updates are executed at recovery source simultaneously with recovery process, then those updates should also be moved to the recovery target subsequent to the copying of database image.

If an entire recovery approach is executed on one distributed database system, then no other action is needed on other databases. If a partial recovery is executed on one distributed database system, a co-ordinated time based & change based recovery should be done on all the databases having dependencies.

SELF-ASSESSMENT QUESTIONS – 8

18. In a _____, the inherent redundancy in the system can be utilised for recovery.
19. In _____, the consistency of the recovered data should also be considered.

10. DIRECTORY SYSTEMS

Employee information, for example, name, id, email, phone, office address and even private information can be accessed from many places such as Web-browser bookmarks. In directory systems there are two types of entries:

- In White pages, entries are planned by identifier or name and intended for forward lookup to locate more about entry
- In Yellow pages, entries are designed/planned by properties to locate entries matching particular requirements

SELF-ASSESSMENT QUESTIONS – 9

20. In _____, entries are planned by identifier or name and intended for forward lookup to locate more about entry.
21. In _____, entries are designed/planned by properties to locate entries matching particular requirements.

11. DDBMS TRANSPARENCY FEATURES

This feature of DDBMS makes sure that each of its users thinks that she is the only user of the system. That the database is located at various sites or that there are many users accessing the data at the same time is not known to the users. All the implementation difficulties are hidden from the user. The different DDBMS transparency features are:

1. ***Distribution transparency:*** This feature permits one distributed database to be treated as a single (albeit logical) database. The user need not know – that the database is partitioned; that the database is replicated at many sites; the database location. She can work as if the database were locally stored.
2. ***Transaction transparency:*** This feature permits a transaction to be executed either completely or not at all just as in case of a local database system. This feature ensures the integrity of the distributed database system.
3. ***Failure transparency:*** This feature permits the distributed database to continue to be operational even if there is a failure at some nodes. The functions performed by the failed node are carried out by some other operational nodes.
4. ***Heterogeneity transparency:*** The fact that the constituent databases are of different types; that the hardware systems are of different types etc. are not known to the users due to this feature of the DDBMS.

SELF-ASSESSMENT QUESTIONS – 10

22. Distribution transparency feature permits one distributed database to be treated as a single (albeit logical) database (True/False).
23. _____ feature permits distributed database to continue to be operational even if there is a failure at some nodes.

12. DISTRIBUTION TRANSPARENCY

Distribution transparency permits a physically dispersed database to be managed as though it were a centralised database. The degree of transparency maintained by the DDBMS differs from one system to another. 3 degrees of distribution transparency is seen:

- Fragmentation transparency, which is the uppermost degree of transparency. The user has no need to know that the database is partitioned. As a result, neither the fragment names nor the fragment locations are specified before the access to data.
- Location transparency resides when the user should recognise the database fragment names but have no need to declare at which locations these fragments are situated.
- Local mapping transparency subsists when the end user or programmer should specify both the fragment names and their locations.

Transparency features are summarised in Table 11.8.

Table 11.8: Transparency Features

IF THE SQL STATEMENT REQUIRES:			
FRAGEMENT NAME?	**LOCATION NAME?**	**THEN THE DBMS SUPPORTS**	**LEVEL OF DISTRIBUTION TRANSPARENCY**
Yes	Yes	Local mapping	Low
Yes	No	Location transparency	Medium
No	No	Fragmentation transparency	High

As you examine Table 11.8, you might ask why there is not a reference to the state where the fragment is "No" & location is "Yes." The explanation is simple: you cannot incorporate the location name that fails to reference an accessible fragment. (If you don't need to specify a fragment name, its location is clearly unrelated.)

SELF-ASSESSMENT QUESTIONS – 11

24. Which transparency exists when the programmer should indicate both fragment names as well as their locations?
    a) Fragmentation transparency
    b) Local mapping transparency
    c) Location transparency
    d) Failure transparency
25. The end user or programmer needs to recognise that a database is partitioned or not. (True/ False)

13. SUMMARY

Let us recapitulate the important points discussed in this unit:

- A database that physically resides entirely on one machine under a single DBMS is known as local database management system.
- When the database technology is the same at each of the locations and the data at the several locations are also compatible that data is known as homogenous database.
- The transaction manager transmits prepare/organises, commits and aborts messages to all of its assistant transaction managers.
- When one distributed database node fails, its main memory contents get vanished, and should somehow be recovered.
- Horizontal fragmentation divides the relation by assigning each tuple of N to one or more fragments.
- In Horizontal fragmentation scheme, a relation N is partitioned and subsets are created.
- The degree of transparency maintained by the DDBMS is different from one system to another.

14. GLOSSARY

- ***Data fragmentation:*** A system bears data fragmentation, if only data/file could be split into fragments for the purpose of physical storage.
- ***Distributed database:*** Database which is multiple-source and multiple- location is called distributed database.
- ***Distributed transactions:*** Distributed transactions enhance the advantages of transactions for those applications that need to update data.
- ***DTC:*** Distributed Transaction Coordinator; the DTC coordinates transactions that update two or more transaction-protected resources like databases and files systems.
- ***Homogenous database:*** Homogenous database simplifies the data sharing between different users.

15. TERMINAL QUESTIONS

1. Explain the functions and components of DDBMS.
2. What is the difference between homogeneous and heterogeneous databases?
3. Discuss data fragmentation and its types.
4. What are commit protocols?

16. ANSWERS

Self-Assessment Questions

1. Sites
2. False
3. Twisted pair
4. Data Processor
5. (a) User
6. (b) Heterogeneous Database
7. (c) Data
8. (a) Complexity
9. False
10. Resource managers
11. Processes
12. Lock manager
13. True
14. Two-phase commit
15. Commit protocols
16. True
17. False
18. Replicated database approach
19. distributed databases
20. White pages
21. Yellow pages
22. True
23. Failure transparency
24. Local mapping transparency
25. False

Terminal questions

1. Maintaining track of data, processing of distributed query, etc. is the function of DDBMS. Refer Section 2 for more details.
2. A Homogeneous database will have only one DBMS, while heterogeneous databases have numerous DBMS's. Refer Section 3 for more details.
3. Commit protocols are utilised to make sure atomicity across sites. Refer Section 8 for more details.
4. To break up the database into logical units is known as data fragmentation. Refer Section 12 for more details.

17. REFERENCES

- Ramakrishnan, R. & Gehrke, J. (2003), *Database Management Systems*, Third Edition, New Delhi, India: McGraw-Hill.
- Rob, P. & Coronel, C. (2006), *Database Systems: Design, Implementation and Management,* Seventh Edition, Thomson Learning.
- Silberschatz, Korth & Sudarshan (1997), *Database System Concepts,* Fourth Edition, McGraw-Hill
- Navathe, E. (2000), *Fundamentals of Database Systems*, Third Edition, Pearson Education Asia.

E-references

- http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3 ed-english/Ch22b_DistributedDBs-95.pdf
- http://pcbunn.cacr.caltech.edu