4 : Distributed Database Systems

IT3306 – Data Management

Level II - Semester 3

Overview

- This lesson discusses about the concepts, advantages and different types of distributed database systems.
- Distributed database design techniques and concepts such as fragmentation, replication, allocation will be discussed in detail.
- Thereafter, we will be looking at the distributed database query optimization techniques.
- Finally, the NoSQL characteristics related to DDB will be discussed.

Intended Learning Outcomes

- At the end of this lesson, you will be able to;
  - Describe the concepts in data distribution and distributed data management.
  - Analyze new technologies that have emerged to manage and process big data.
  - Explain the distributed solutions provided in NoSQL databases.
  - Describe different concepts and systems being used for processing and analysis of big data.
  - Describe cloud computing concepts.

List of subtopics

4.1. Distributed Database Concepts, Components and Advantages

4.2. Types of Distributed Database Systems

4.3. Distributed Database Design Techniques

- 4.3.1. Fragmentation
- 4.3.2. Replication and Allocation
- 4.3.3. Distribution Models: Single Server , Sharding, Master-Slave, Peer-to-Peer

4.4. Query Processing and Optimization in Distributed Databases

- 4.4.1 Distributed Query Processing
- 4.4.2 Data Transfer Costs of Distributed Query Processing

4.5. NoSQL Characteristics related to Distributed Databases and Distributed Systems

4.1 Distributed Database Concepts, Components and Advantage

- A system that performs certain assigned tasks with the help of several sites which are connected via a computer network is known as a distributed computing system.
- The goal of a distributed computing system is to partition a complex problem that requires a large computational power into smaller pieces of work .
- Distributed Database technology has emerged as a result of the merger between database technology and distributed systems technology.

4.1 Distributed Database Concepts, Components and Advantage

- Distributed database (DDB) is a set of logically interrelated databases connected via a computer network.
- To manage the distributed database and to make it transparent to the user, we are using a software called distributed database management system (DDBMS).
- Following are the minimum conditions that should be satisfied by a database to be distributed:
  - Multiple computers (nodes) connected over a network to transmit data.
  - Logical relationship between the information available in different nodes.
  - The hardware, software and data related to each site is not mandatory to be identical.

4.1 Distributed Database Concepts, Components and Advantage

- Location of the nodes either can be with in a same physical location connected via a LAN (Local Area Network) or geographically disperse which is connected via WAN (Wide Area Network).
- We can use different network topologies to establish the communication between sites.
- The topology we select directly affects the performance and the query processing of the distributed database.

4.1 Distributed Database Concepts, Components and Advantage

Transparency

- In general, transparency is not allowing the end user to know implementation details.
- There are several types of transparencies introduced in the distributed database domain because the data is distributed in multiple nodes.
    i. **Location transparency** : Commands issued are not changed according to the location of data or the node.
    ii. **Naming transparency**: When a name is associated with an object, the object can be accessed without giving additional details such as the location of data.

4.1 Distributed Database Concepts, Components and Advantage

Transparency Cont.

iii. **Replication transparency** : User is not aware of the replicas that are available in multiple nodes in order to provide better performance, availability and reliability.

iv. **Fragmentation transparency**: User is not aware of the fragments available.

v. **Design transparency**: User is unaware of the design of the distributed database while he is performing the transactions.

vi. **Execution transparency**: User is unaware of the transaction execution details.

4.1 Distributed Database Concepts, Components and Advantage

Reliability and Availability

- Reliability is the probability of a system in the running state at a given time point.
- Availability is defined as the probability of a system been continuously available at a given time interval.
- There is a direct relationship between reliability & availability with the database faults, errors, and failures.
- If a system deviates from it's defined behaviour, we call it a **Failure**.
- **Errors** contain a subset of states which causes the failures.
- A cause of an error is known as a **Fault**.

4.1 Distributed Database Concepts, Components and Advantage

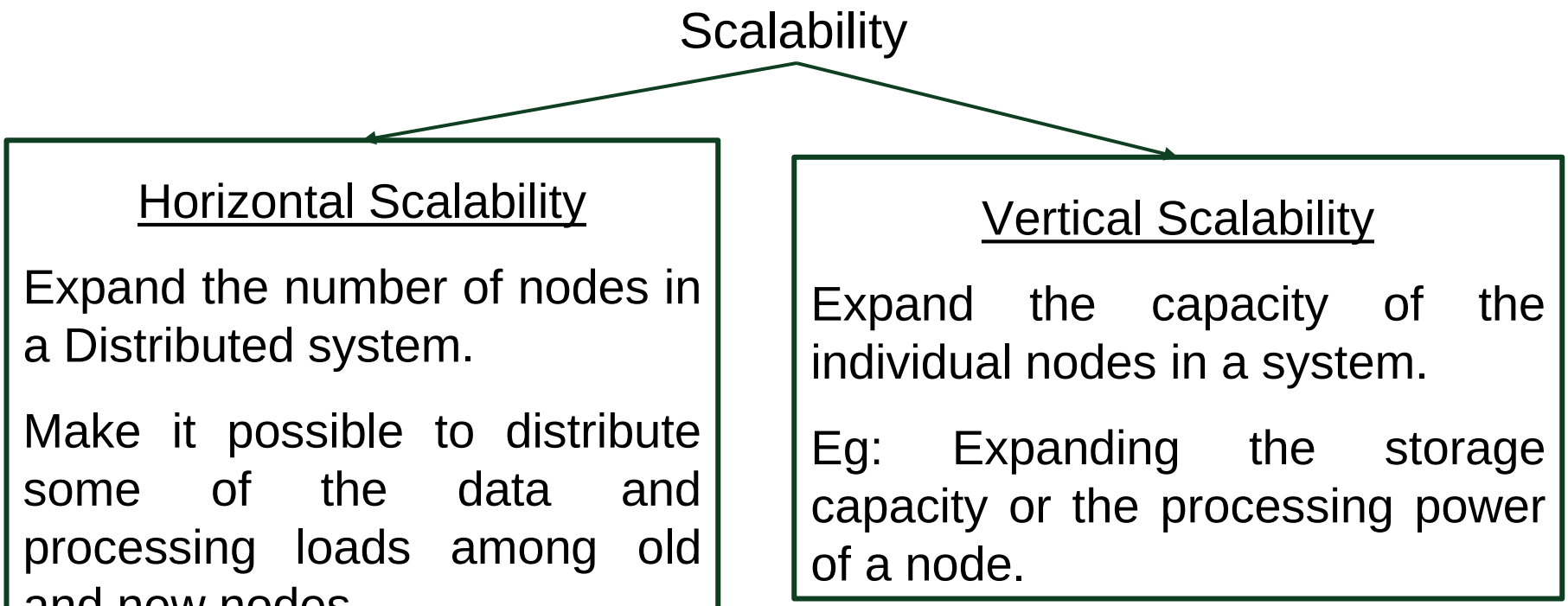
Reliability and Availability Cont.

- There are several approaches to make a system reliable.
- One method is *fault tolerance.*
- In this method, we identify and eliminate faults before they result in system failures.
- Another method is ensuring the system do not contain any faults by conducting quality control measures and testing.
- A reliable DDBMS should be able to process user requests as long as database consistency is preserved.
- The recovery manager in a DDBMS is working on the failures arising from different aspects such as transactions, hardware, and communication networks.

4.1 Distributed Database Concepts, Components and Advantage

Scalability and Partition Tolerance

- Scalability is identifying to which extent the system can be expanded without making a disturbance to the operations.
- There are two main types of scalability as follows.



4.1 Distributed Database Concepts, Components and Advantage

Scalability and Partition Tolerance Cont.

- When the number of nodes are increased, the possibility of network failures also grows up, resulting the nodes to be partitioned into subgroups.
- In this situation, the nodes within a single subnetwork can communicate each other while the communication among partitions are lost.
- The ability of the system to keep operating even though the network is divided into separate groups is known as partition tolerance.

4.1 Distributed Database Concepts, Components and Advantage

Autonomy

- The extent to which a single node (database) have the capacity to be worked independently is refer to as Autonomy.
- Higher flexibility is given to the nodes when there is high autonomy.
- Autonomy can be applied in many aspects such as,
  - *Design autonomy*: Independence of data model usage and transaction management techniques.
  - *Communication autonomy*: The extent to which each node can decide on sharing of information with other nodes.
  - *Execution autonomy*: Independence of users to operate as they prefer.

4.1 Distributed Database Concepts, Components and Advantage

Advantages of DDB

1. Improves the flexibility of application development
   - The ability of carrying out application development and maintenance from different physical locations.
2. Improve Availability
   - Faults are isolated to the site of origin without disturbing the other nodes connected.
   - Even though a single node fails, the other nodes continue to operate without failing the entire system. (However, in a centralized system, failure at a single site makes the whole system unavailable to all users). Therefore, availability is improved with a DDB.

4.1 Distributed Database Concepts, Components and Advantage

Advantages of DDB Cont.

3. Improve performance
   - Data items are stored closer to where it is needed the most. It reduces the competition for CPU and I/O services required. The access delays involved in wide area networks are also brought down.
   - Since each node holds only a partition of the entire DB, the number of transactions executed in each site is smaller compared to the situation where all transactions are submitted to a single centralized database.
   - Execution of queries in parallel by executing multiple queries at different sites, or by splitting the query into a number of subqueries also improves the performance.

4.1 Distributed Database Concepts, Components and Advantage

Advantages of DDB Cont.

4. Easy expansion

- Ability to make the system expanded by adding more nodes or increasing the database size helps to facilitate the growth of data much easier when compared to a centralized system.

Activity

Select the advantages of a distributed database over a centralized database from the following features given.

- Less cost
- Slow responses
- Less complexity
- Improved performance
- Easier Scalability
- Availability improvement
- Maintainability
- Flexibility in application development

Activity

Fill in the blanks with the most suitable words given.

(same,multiple,network,location,replication,execution, design, horizontal, vertical, communication, same, fragmentation, naming)

With _____ transparency, user is unaware about the different locations, where the data is stored.

Making the user unaware of having multiple copies of the same data item in different sites is referred to as _____ transparency.

The ability of increasing the number of nodes in a distributed database is _____ scalability.

Increasing the storage capacity of nodes is known as _____ scalability.

Activity

State whether the given statements are True or False.

1. A system with high transparency offers a lot of flexibility to the application developer. (True / False)
2. It is mandatory for all the nodes to be identical in terms of data, hardware, and software. (True / False)
3. A distributed Database should be connected via a local area network. (True / False)
4. In DDB systems, expanding the processing power of nodes is ***not*** considered as a way of increasing scalability. (True / False)
5. With data localization, number of CPU and I/O services required can be reduced. (True / False)

4.2 Types of Distributed Database Systems

- There are different types of Distributed Database Management Systems classified based on the degree of homogeneity.
  - **Homogeneous system**: All the sites(servers) in the DDB use identical software and all the clients use the identical software.
  - **Heterogeneous system**: Different software installed in the servers or if the users involved in DDB use different software.

4.2 Types of Distributed Database Systems

- Degree of local autonomy is another factor relevant to the degree of homogeneity.
- If the local site is not allowed to be operated as an independent site, there is no local autonomy.
- If local transaction granted permission for direct access to the server, then there is some degree of local autonomy.

4.2 Types of Distributed Database Systems

- Classification of DDBMS with regards to distribution, autonomy, and heterogeneity can be explained as below.
  - **Centralized DB**: Got complete autonomy but a complete loss of distribution and heterogeneity.
  - **Pure distributed database systems**: There is only one conceptual schema. A site, which is a part of the DDBMS provides access to the system. Therefore, no local autonomy exists.
  - Further classification of centralized DBMS can be done with level of autonomy. Those are federated database systems and multi database systems. These systems consist of independent servers, centralized DBMS with local users, local transactions and DBA, facilitating very higher degree of local autonomy.

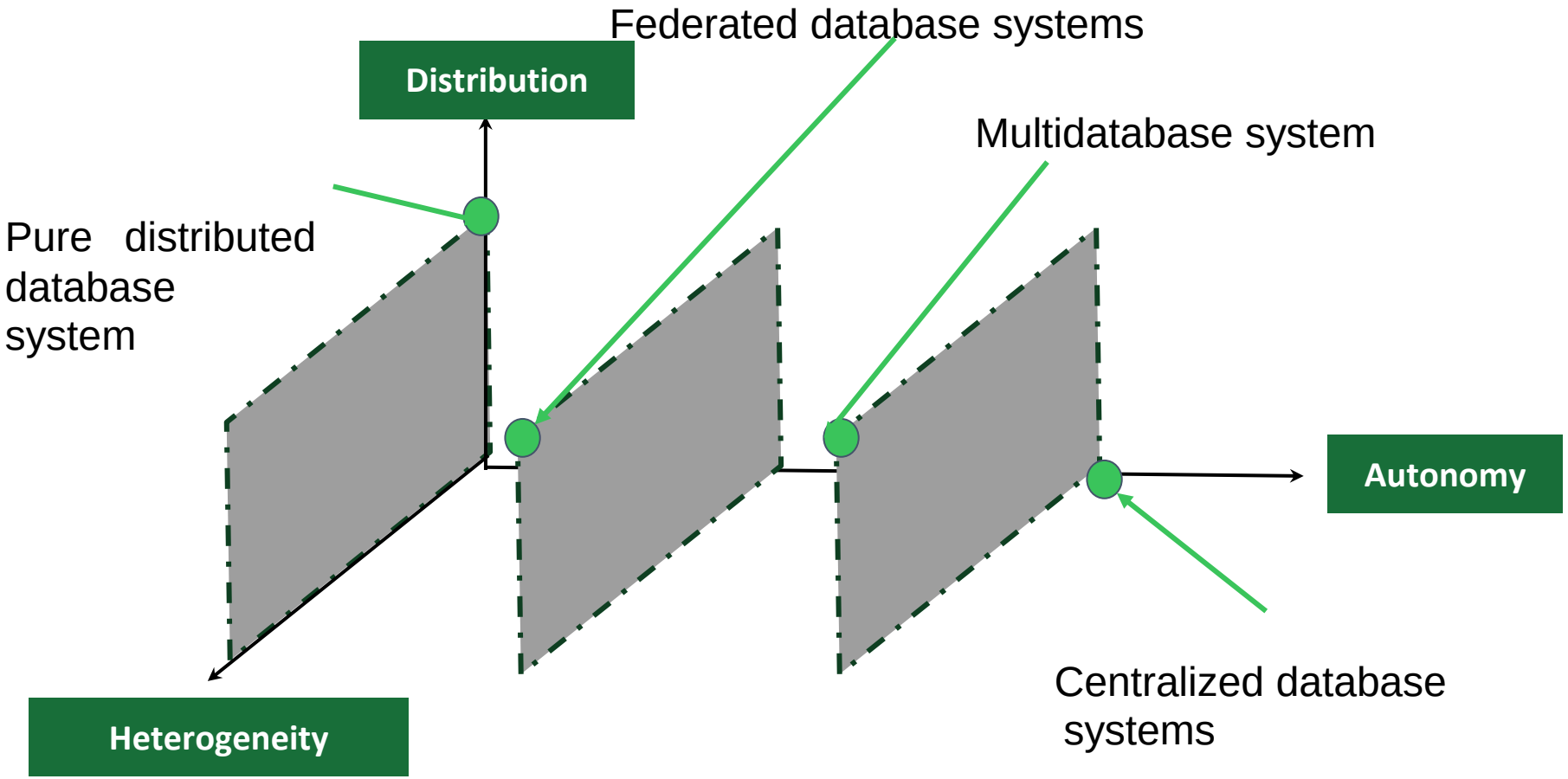
4.2 Types of Distributed Database Systems

- Federated database systems: Have a global view of the federation of databases that is shared by the applications.
- Multidatabase systems: Have full local autonomy in DB but does not have a global schema.

  eg: A system with full local autonomy and full heterogeneity. (Peer-to-peer database system)

4.2 Types of Distributed Database Systems

Classification of the distributed databases that we discussed in previous two slides can be seen in the following image.



4.3 Distributed Database Design Techniques

Fragmentation

- As the name implies, in distributed architecture, separate portions of data should be stored in different nodes.
- Initially, we have to identify the basic logical unit of data. In a relational database, relations are the simplest logical unit.
- Fragmentation is a process of dividing the whole database into various sub relations so that data can be stored in different systems.

4.3 Distributed Database Design Techniques

Example

- Suppose we have a relational database schema with three tables (EMPLOYEE, DEPARTMENT, WORKS_ON) which we should make partitions in order to store in several nodes.
- Assume there are no replications allowed (*data replication allows storage of certain data in more than one place to gain availability and reliability*).

Employee

FNAME	LNAME	SSN	BDATE	ADDRESS

Works_on

ESSN	DNO	HOURS

Department

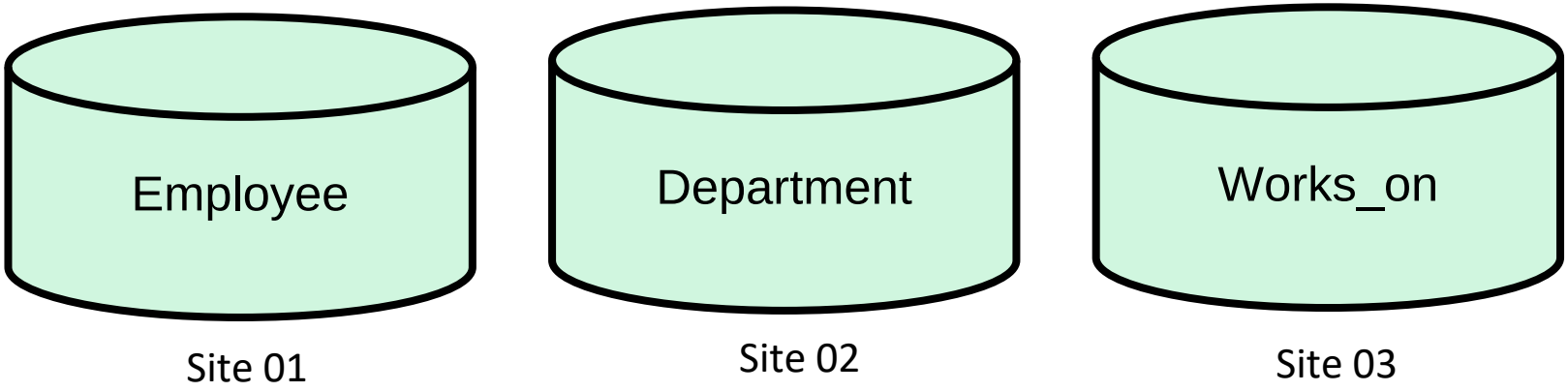
DNO	DNAME	LOCATION

4.3 Distributed Database Design Techniques

Example - Approach 01

- One approach of data distribution is storing each relation in each site.

We can store each relation in one node. In the following example, we have stored the Employee table in Node 1, the Department table in node 2 and Works_on table in node 3.



Data distribution technique 01: Storing each relation in each site.

4.3 Distributed Database Design Techniques

Example - Approach 02

- Another approach is dividing a relation into smaller logical units for distribution.
- For instance, think of a scenario where 3 different departments are located in 3 separate places. Finance department in Colombo, research department in Rathnapura and headquarters in Kandy as given in the below table.

DNO	DNAME	LOCATION
d4	Headquarters	Kandy
d3	Finance	Colombo
d8	Research	Rathnapura

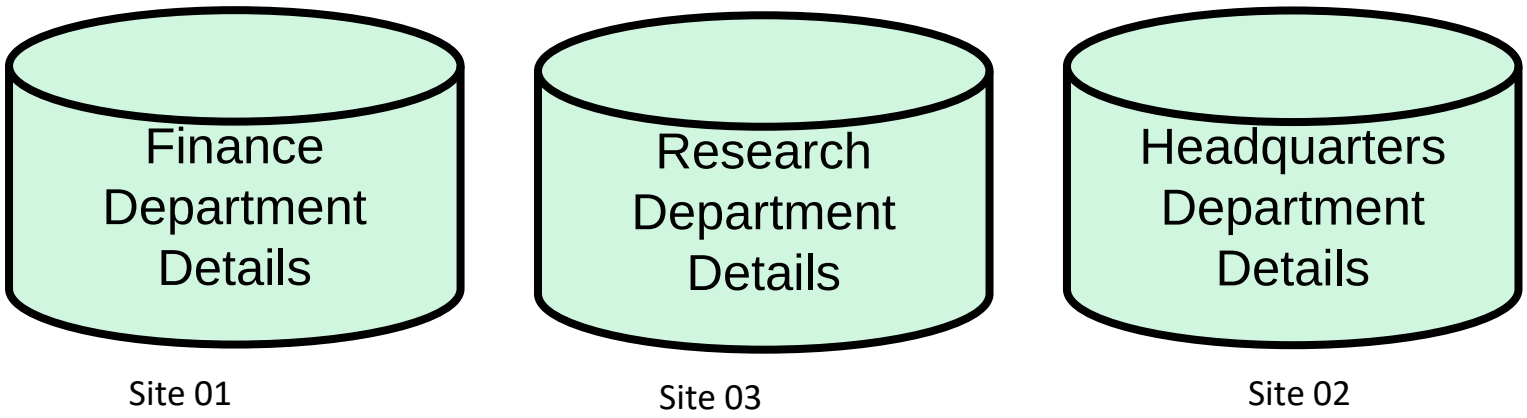
4.3 Distributed Database Design Techniques

Example - Approach 02 Cont.

- For the scenario given in previous slide, we can store data relevant to each department in separate site.
- The details of finance department will be stored in one site.
- Details of headquarters will be stored in another site.
- Details of research department will be stored in another separate site.
- Dividing a relation into smaller logical units can be done by horizontal fragmentation or vertical fragmentation which will be discussed in coming slides.

4.3 Distributed Database Design Techniques

Example - Approach 02 Cont.



Data distribution technique 02: Storing details of different departments in each site.

4.3 Distributed Database Design Techniques

Horizontal Fragmentation

- A subset of rows in a relation is known as horizontal fragment or shard.
- Selection of the tuple subset is based on a condition of one or more attributes.
- With horizontal fragmentation, we can divide tables horizontally by creating subsets of tuples which has a logical meaning for each of the subset.
- Then these fragments are assigned to different nodes in the distributed system.
- Each horizontal fragment on a relation R can be specified in the relational algebra by $\sigma_{Ci}(R)$ operation.(Ci → condition, R→ relation).
- Reconstruction of the original relation is done by taking the **union** of all fragments.

- Ex:- If we want to store sales employee details and marketing employee details separately in 2 nodes, we can use horizontal fragmentation.

Employee

Name	Salary	Department
Kasun	120000	Sales
Rishad	135000	Sales
Kirushanthi	45900	Marketing
Anna	47900	Marketing

Sales Employee

Name	Salary	Department
Kasun	120000	Sales
Rishad	135000	Sales

Marketing Employee

Name	Salary	Department
Kirushanthi	45900	Marketing
Anna	47900	Marketing

4.3 Distributed Database Design Techniques

Explanation →

- Original table (Employee) is divided into two subset of rows.
- First horizontal fragment created (Sales_employee) consists of details of employees who are working in the sales department.

Sales_employee ← σDepartment = “sales” (Employee)

- Second horizontal fragment created (Marketing_employee) consists of details of employees who are working in the marketing department.

Marketing_employee ← σDepartment = “marketing” (Employee)

4.3 Distributed Database Design Techniques

Vertical Fragmentation

- With vertical fragmentation, we can divide the table by columns.
- There can be situations where we do not need to store all the attributes of a relation in a certain site.
- Therefore, with the technique of vertical fragmentation, we can keep only required columns of a relation within a single site.
- In vertical fragmentation, it is a must to include the primary key or some unique key attribute in every vertical fragment. Otherwise, we will not be able to create the original table by putting the fragments together.

4.3 Distributed Database Design Techniques

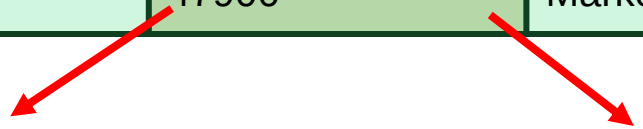
Vertical Fragmentation Cont.

- A vertical fragment on a relation R can be specified by a $\pi_{Ai}(R)$ operation in the relational algebra. (Ai → attributes, R→ relation)
- The **Outer Union** on vertical fragments can generate the original table.

- Ex:- If we want to store employees' pay details and department details separately in 2 nodes, we can use vertical fragmentation.

Employee

Name	Salary	Department
Kasun	120000	Sales
Rishad	135000	Sales
Kirushanthi	45900	Marketing
Anna	47900	Marketing



pay data

Name	Salary
Kasun	120000
Rishad	135000
Kirushanthi	45900
Anna	47900

Dept. data

Name	Department
Kasun	Sales
Rishad	Sales
Kirushanthi	Marketing
Anna	Marketing

4.3 Distributed Database Design Techniques

Explanation →

- Original table (Employee) is divided into two subset of columns.
- First vertical fragment created (Pay_data) consists of salary details of employees.

  Pay_data ← πname, salary(Employee)

- Second vertical fragment created (Dept_data) consists of department details of employees.

  Dept_data ← πname, Department (Employee)

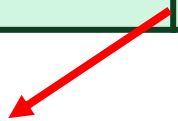
4.3 Distributed Database Design Techniques

Mixed Fragmentation

- Another fragmentation technique is the hybrid (mixed) fragmentation where we can use a combination of both the horizontal and vertical fragmentations.
- For example, take the EMPLOYEE table that we used before.
- Employee table is vertically split into payment data and department data. (vertical fragmentation)
- Then the department table is again separated by the department, where the horizontal fragmentation is taking place. (horizontal fragmentation)
- Relevant fragmentations with data can be seen in the next slide.

Employee

Name	Salary	Department
Kasun	120000	Sales
Rishad	135000	Sales
Kirushanthi	45900	Marketing
Anna	47900	Marketing



pay data

Name	Salary
Kasun	120000
Rishad	135000
Kirushanthi	45900
Anna	47900

Sales -Deptdata

Name	Department
Kasun	Sales
Rishad	Sales

Marketing -Deptdata

Name	Department
Kirushanthi	45900
Anna	47900

Activity

1) Give horizontally fragmented relations (with data) for the Project relation given below so that projects with budgets less than 150,000 are separated from projects with budgets greater than or equal to150,000. Express the fragmentation conditions using relational algebra for each fragment (with data).

2) Indicate how the original relation would be reconstructed.

ProjNo	ProjName	Budget	Location
23	Boks	100000	Colombo
4	Goods	50000	Galle
65	Furniture	75000	Jaffna
87	Clothes	200000	Matara

Activity

1) Give a vertical fragmentation of the above Project relation into two sub-relations (with data), so that one contains only the information about project budgets (i.e. ProjNo, Budget), whereas the other contains project names and locations (i.e. ProjNo, ProjName, Location). Express the fragmentation condition using relational algebra for each fragment.

2) Indicate how the original relation would be reconstructed.

ProjNo	ProjName	Budget	Location
P1	Books	100,000	Colombo
P2	Goods	50,000	Galle
P3	Furniture	75,000	Colombo
P4	Clothes	200,000	Kandy

Activity

Fill in the blanks with the most suitable word given.

(vertical, horizontal, mixed, union, outer join, projection, selection)

When we divide a relation based on columns, it is known as _____ fragmentation while, the relation divided based on rows know as _____ fragmentation.

A combination of these 2 fragmentations is referred to as _____ fragmentation.

The re-constructability of the relation from its fragments ensures that constraints defined on the data in the form of dependencies are preserved. A set of vertical fragments can be organized into the original table using _____ operation.

With _____ operation, we can create the original relation from a set of horizontal fragments.

4.3 Distributed Database Design Techniques

Replication

- The main purpose of having data replicated in several nodes is to ensure the availability of data.
- One extreme of data replication is having a copy of the entire database at every node (full replication).
- The other extreme is not having replication at all. Here, every data item is stored only at one site. (no replication)

4.3 Distributed Database Design Techniques

Replication Cont.

- Full replication

  -With full replication, we can achieve a higher degree of availability. The reason for this is, the entire system keeps running, even with only one site up, because every site contains the whole DB.

  -The other advantage is improved performance of read queries, as the results can be obtained from any site by locally processing at the site where it submitted.

  -However, there are drawback of full replication.

  -One is, degrading the write performance, because each update should be performed at every copy of data to maintain the consistency.

  -Making the concurrency control and recovery techniques are more complex and expensive.

4.3 Distributed Database Design Techniques

Replication Cont.

- No replication

  -When there are no replications, all fragments must be disjoint (no tuple in relation R, can be seen in more than one site.) But the repetition of primary key should be expected for the vertical fragments or mixed fragments.

  -Also known as non-redundant allocation.

  -Suitable for systems with high write traffic.

  -Lesser degree of availability is a disadvantage of no replication.

4.3 Distributed Database Design Techniques

Replication Cont.

- To get a balance between the pros and cons we discussed, we can select a degree of replication suitable for our application.
- Some fragments of the database may be replicated, and others may not according to the requirements.
- It is also possible to have some fragments replicated in all the nodes in the distributed system.
- Any way, all the replicas should be synchronized when an update is taken place.

4.3 Distributed Database Design Techniques

Allocation

- There cannot be any site which is not assigned to a site in a DDB.
- The process of distributing data into nodes is known as data allocation.
- The decisions of selecting the site to hold each fragment and the number of replicas available for each data depends on the,
    - Performance requirement of the system
    - Types of transactions
    - Availability goals
    - Transaction frequency

4.3 Distributed Database Design Techniques

Allocation Cont.

Consider the following scenarios and the suggested allocation mechanisms:

- Requires high availability of the system with high number of retrievals,

  - Recommend to have a fully replicated database.

- Requires to retrieve a subsection of data frequently,

  -Recommend to allocate the required fragment into multiple sites.

- Requires to perform a higher number of updates,

  -Recommend to have a less number of replicas.

However, It is hard to find an optimal solution to distributed data allocation since it is a complex optimization problem.

Activity

Select advantages of data replication in DDB.

1. Improves availability of data.
2. Improves performance of data retrieval.
3. Improves performance of data write operations.
4. Slow down update queries.
5. Hard recovery.
6. Expensive concurrency control.
7. Slow down select queries.
8. Easy notions used for data query.

Activity

Mark the following statements as true (T) and false (F).

1. With data replication, we can have multiple copies of the same data item in many sites. ( )
2. Data replication would slow down the read and write operations of a database. ( )
3. There can be some data fragments which are replicated in all nodes of the distributed database. ( )
4. The number of copies created for each fragment should be equal. ( )
5. In a replicated system, there can be fragments which are not replicated in another site. ( )
6. For a system with frequent updates, it is advised to use a larger number of replications. ( )

4.3 Distributed Database Design Techniques

Distribution Models

When the data volume increases, we can add more nodes within our distributed database system to handle it. There are different models for distributing data among these nodes.

1. Single server

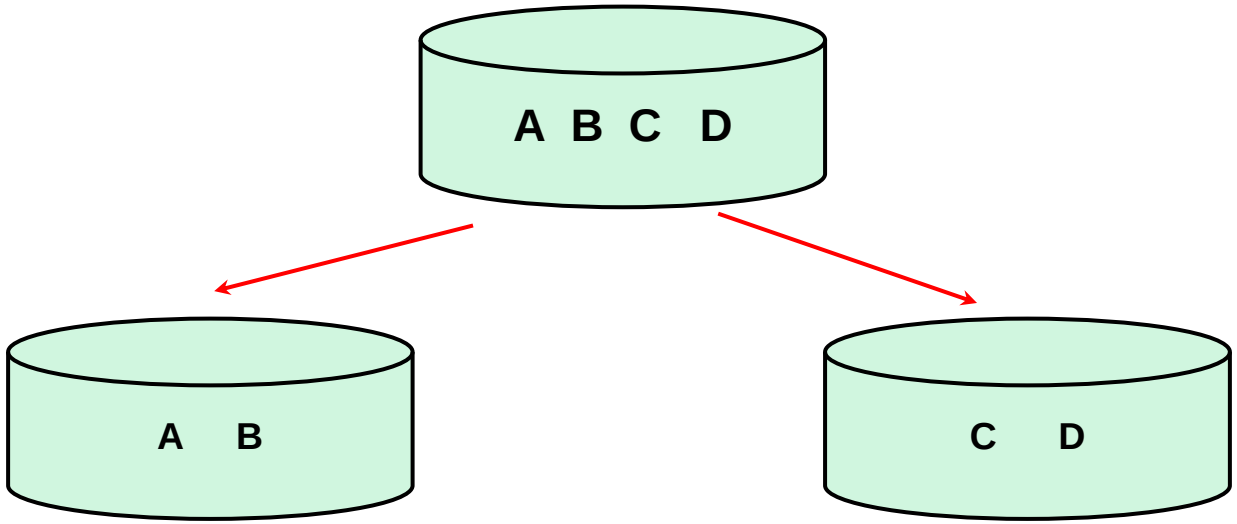
- This is the minimum form of distribution and most often the recommended option.
- Here, the database will be running in a single server without any distribution.
- Since all read and write operations occur at a single node, it would reduce the complexity by making the management process easy.

4.3 Distributed Database Design Techniques

Distribution Models Cont.

2. Sharding

- A database can get busy when several users access different data in different parts of the database at the same time.
- This can be solved by splitting data into several parts and storing them in different nodes. This is called **sharding**.



4.3 Distributed Database Design Techniques

Distribution Models Cont.

- In the best-case scenario of sharding, different users will access different parts of the database stored in separate nodes, so that each user will only communicate with a single node.
- This technique will help in load balancing.
- It is necessary to segregate data correctly, for this technique to be effective. Data that are accessed together should be stored in a single node.

4.3 Distributed Database Design Techniques

Distribution Models Cont.

- The following considerations should be made when segregating (or sharding) the data
  - **Location:** Place data close to the physical location of access.
  - **Load Balancing:** Make sure that each node will get approximately similar number of requests.
  - **Order of access:** Aggregates that will be read in sequence can be stored in a single node.

4.3 Distributed Database Design Techniques

Distribution Models Cont.

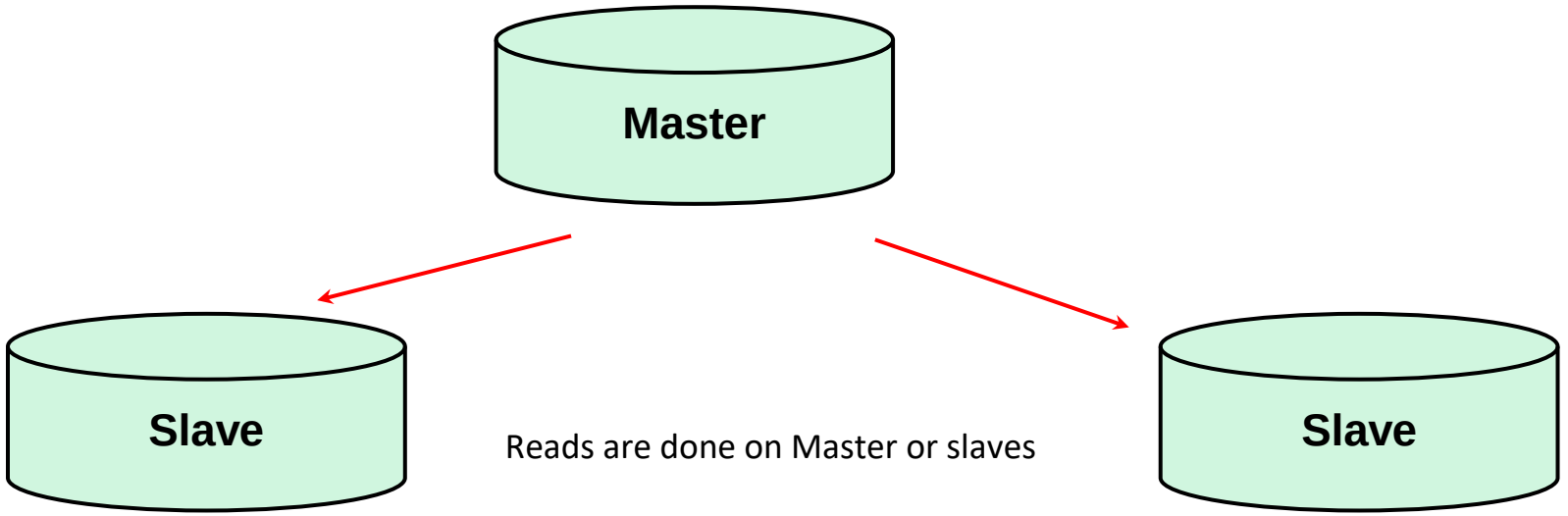
- Auto-sharding is a feature given by most of the NoSQL databases, where the responsibility of splitting and storing data is given to the database itself, ensuring that data goes to the correct shard.
- Sharding will improve read performance as well as write performance.
  - Improve read performance by replication and caching
  - Improve write performance by horizontally scaling writes.
- It is hard to achieve reliability only with the use of sharding.
- To improve reliability, it is necessary to use data replication along with sharding. Otherwise, even though the data can be accessed from different nodes, a failure of a node can make the shard unavailable.

Distributed Database Design Techniques

Distribution Models Cont.

3. Master-slave replication

- In this model, one node is selected as the master (primary) and it is considered as the authorized source for data.
- Master is the node which is responsible for updates.
- All the other nodes are treated as slaves (secondary).
- There is a process called synchronization to sync data inside master with the slaves.



4.3 Distributed Database Design Techniques

Distribution Models Cont.

- Master-slave model is suitable for a system with read-intensive dataset.
- By adding more slaves, you can increase the efficiency of read operations since the read requests can be processed by any slave node.
- However, there is still a limitation on writes, because only master can process the writes to the database.
- If the master fails, it should be recovered or a slave node has to be appointed as the new master.

4.3 Distributed Database Design Techniques

Distribution Models Cont.

- Appointment of the new master can be either an automatic or a manual process.
- The disadvantage of having replicated nodes is the inconsistency that may occur in between nodes.
- If the changes are not propagated to all the slave nodes, there is a chance of different clients who are accessing various slave nodes read different values.

4.3 Distributed Database Design Techniques

Distribution Models Cont.

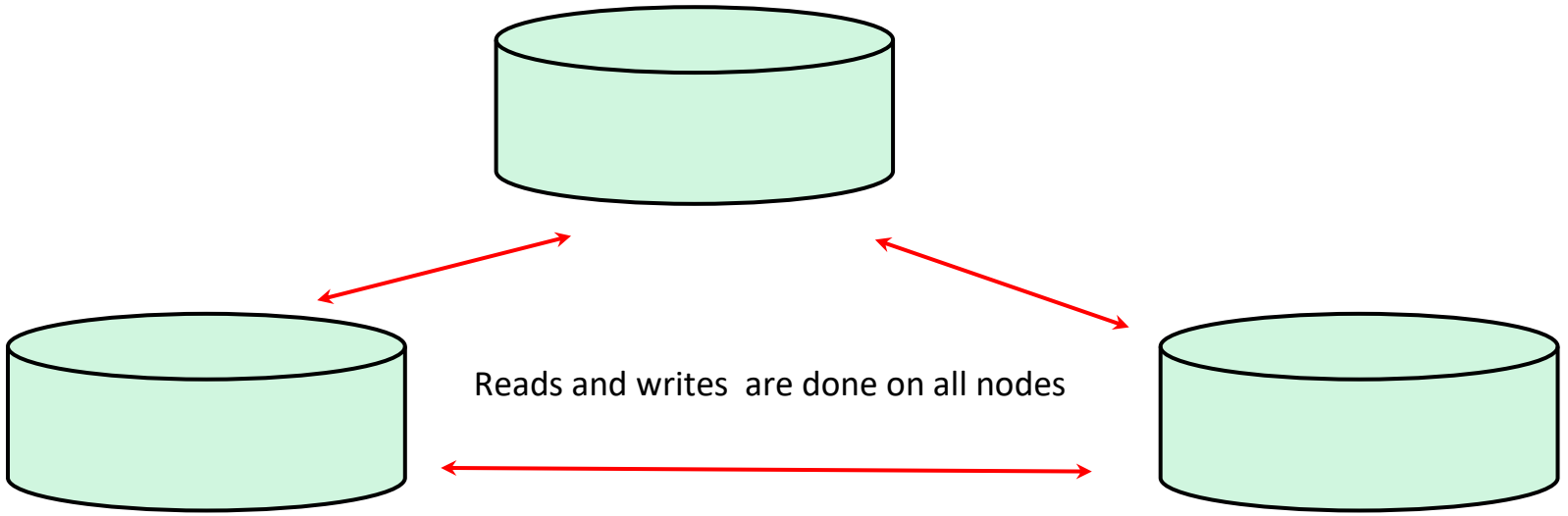
4. Peer-to-peer replication

- In master-slave model, the master is still a bottleneck and a single point of failure.
- In peer-to-peer model, there is no master and all the nodes are of the equal weight.
- All the replicas can accept writes. Due to this reason, there will be no loss of access to data due to failure of a single node.
- However, with this model, we have to accept the problem of inconsistency.
- After you write on a node, two users who are accessing that changed data item from different nodes may read two different values until data propagation is completed.

4.3 Distributed Database Design Techniques

Distribution Models Cont.

- One solution for this inconsistency problem is, ensuring the coordination between replicas to synchronize with all the nodes after performing a write operation.
- Another solution would be coping with an inconsistent write.



4.3 Distributed Database Design Techniques

Distribution Models Cont.

Combining sharding and replication

- We can use both master-slave replication and sharding together.
    - In that approach, we have multiple masters. But there is only one master for each data item.
- Also, we can combine peer-to-peer replication and sharding.
    - A common application of this can be seen in column-family databases.

Activity

Mark the following statements as true (T) and false (F).

1. Scaling up is including larger data servers with higher storage capacity to cater the increasing data storage requirement. ( )
2. Scale out is the process of running the database on a cluster of servers. ( )
3. We can ensure reliability of a DDB by using the technique sharding. ( )
4. Single server is the most recommended distribution model ( )
5. Read reliance is one of the advantages of master-slave replication model. ( )

4.4 Query Processing and Optimization in Distributed Databases

These are the steps involved in distributed query processing. We will discuss each step in detail.

4.4 Query Processing and Optimization in Distributed Databases

Step 01: Query Mapping.

- The query inserted is specified in query language.
- Then it is translated into an algebraic query.
- The translation process is referred to global conceptual schema; here it does not consider the replicas and shards.
- The algebraic query is then normalized and analyzed for semantic errors.
- This step is performed at a central control site.

4.4 Query Processing and Optimization in Distributed Databases

Step 02: Localization.

- In this phase, the distributed query in global schema is mapped to separate queries on fragments.
- For this, data distribution and fragmentation details are used.
- performed at a central control site.

4.4 Query Processing and Optimization in Distributed Databases

Step 03: Global Query Optimization.

- Optimization is selecting the optimal strategy from a list of candidate strategies.
- These candidate strategies can be obtained by permuting the order of operations generated in previous step.
- To measure the cost associated with each set of operations, we use the execution time.
- The total cost is calculated using costs such as CPU cost, I/O costs, and communication costs.
- Since the nodes are connected via network in a DDB, the most significant cost is for the communication between these nodes.

4.4 Query Processing and Optimization in Distributed Databases

Step 04: Local Query Optimization.

- This stage is common to all sites in the DDB.
- The techniques are similar to those used in centralized systems.
- performed locally at each site.

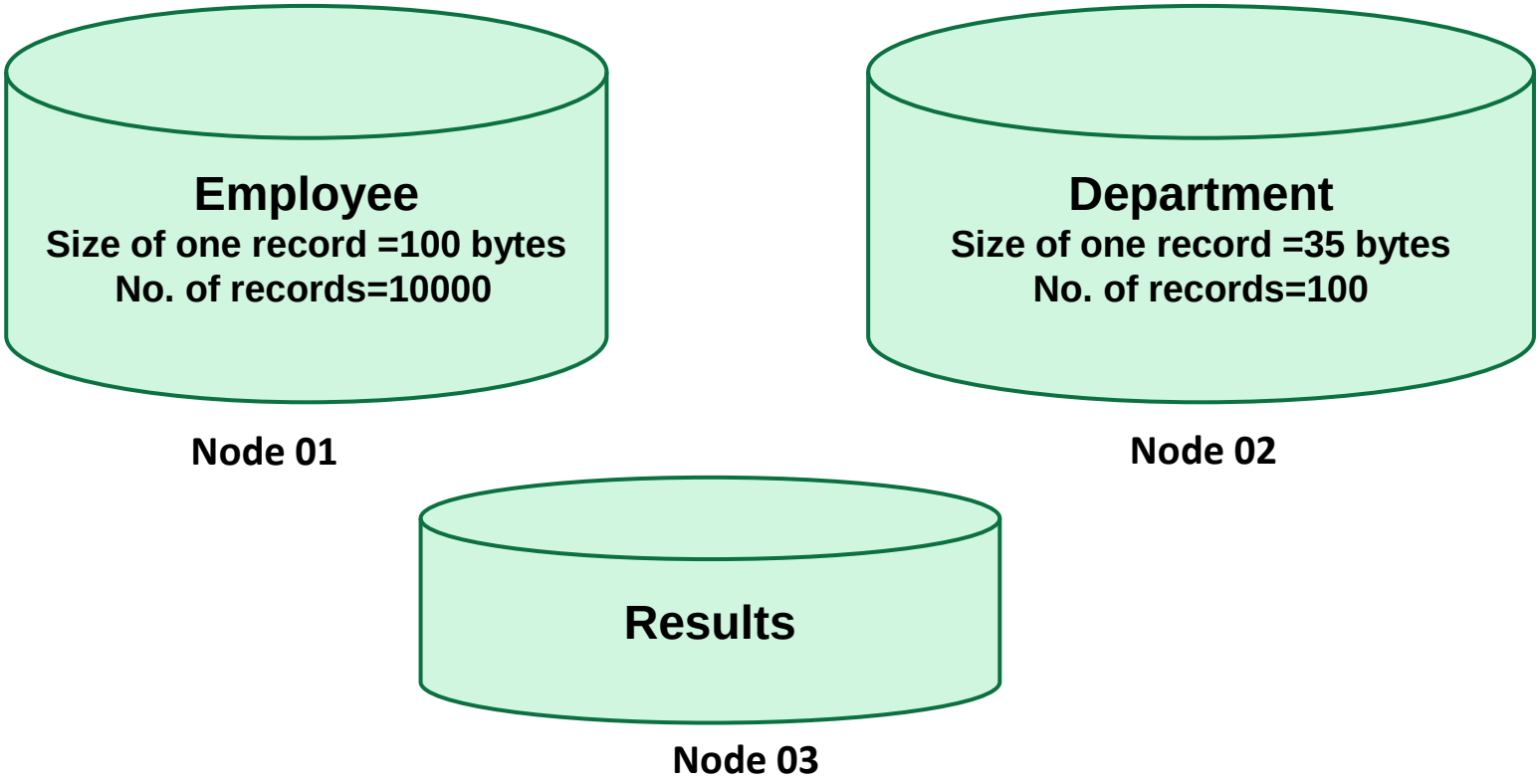
4.4 Query Processing and Optimization in Distributed Databases

- In comparison to a centralized database system, in a distributed database there is additional complexity involved in query processing.
- One is the cost of transferring data among sites.
- Intermediate files or the final result set can be transferred in between nodes via the network.
- Reducing the amount of data to be transferred among nodes is considered as an optimization criteria in the query optimization algorithms used in DDBMS.

4.4 Query Processing and Optimization in Distributed Databases

Example

Suppose Employee table and Department table are stored at node 01 and node 02 respectively. Results are expected to be presented in node 03.



4.4 Query Processing and Optimization in Distributed Databases

Example

According to the details given, let's calculate the size of each relation.

No. of records in Employee relation = 10000

size of 1 record in Employee relation= 100

Size of the Employee relation = 100*10000 = 1000000 bytes

No. of records in Department relation = 100

size of 1 record in Department relation= 35

Size of the Department relation = 100*35 =3500 bytes

4.4 Query Processing and Optimization in Distributed Databases

The sizes of attributes in Employee and Department relations are given below.

EMPLOYEE

Fname	Lname	Ssn	Bdate	Address	Sex	Salary

Fname field is 15 bytes long, Lname field is 15 bytes long, Address field is 10 bytes long

DEPARTMENT

Dname	DNumber	Mgr_ssn	Mgr_start_date

Dnumber field is 4 bytes long, Dname field is 10 bytes long, Mgr_ssn field is 9 bytes long

4.4 Query Processing and Optimization in Distributed Databases

Assume we want to write a query to retrieve first name, last name and department for each employee.

We can represent it in relational algebra as follows.

Let's call this query, Q.

Q: π Fname,Lname,Dname (EMPLOYEE $\infty_{Dno=Dnumber}$ DEPARTMENT)

4.4 Query Processing and Optimization in Distributed Databases

Q: π Fname,Lname,Dname (EMPLOYEE $\infty_{Dno=Dnumber}$ DEPARTMENT)

We will discuss 3 strategies to execute this distributed query .

Method 1

Explanation ⟶ Transfer data in the EMPLOYEE relation and the DEPARTMENT relation into the result site (node 03). Then perform the join operation at node 3.

Calculation ⟶

Total no. of bytes to be transferred= Size of the Employee relation + Size of the Department relation

= 1,000,000 + 3,500

= 1,003,500 bytes

4.4 Query Processing and Optimization in Distributed Databases

Method 2

Explanation ⟶ Transfer the EMPLOYEE relation to site 2. Execute the join at site 2. Send the result to site 3.

Calculation ⟶

Total no. of bytes to be transferred= Size of the Employee table + The size of the query result

=

1,000,000 + (40 * 10,000)

= 1,400,000 bytes

Note: One record in result query consist of Fname (15 bytes), LName (15 bytes) and Dname (10 bytes). Altogether 40 bytes.
There are 10,000 records retrieve as result.
Therefor size if the result query is 40 * 10000

4.4 Query Processing and Optimization in Distributed Databases

Method 3

Explanation ⟶ Transfer the DEPARTMENT relation to site 1. Execute the join at site 1. Send the result to site 3.

Calculation ⟶

Total no. of bytes to be transferred = Size of the Department table + size of the query result

= 3,500 + (40 * 10,000)

= 403,500 bytes

4.4 Query Processing and Optimization in Distributed Databases

When considering the three methods we discussed,

Total no. of bytes to be transferred in method 1 = 1,003,500

Total no. of bytes to be transferred in method 2 = 1,400,000

Total no. of bytes to be transferred in method 3 = 403,500

The least amount of data transfer occurs in method 3. Therefore, we choose method 3 as the optimal solution, since it transfers the minimum amount of data.

Activity

Suppose STUDENT table is stored in site 1 and COURSE table is stored in site 2. The tables are not fragmented and the results are stored in site 3. Every student is assigned to only one course.

STUDENT(Sid, StudentName, Address, Grade, CourseID)
1000 records, each record is 50 bytes long
Sid: 5 bytes, StudentName;10 bytes, Address: 20 bytes

COURSE(Cid, CourseName)

500 records, each record is 30 bytes long
Cid: 5 bytes, CourseName:10 bytes

Query: Retrieve the Student Name and Course Name which the student is following.

Write the relational algebra for the above query.

Activity

Suppose STUDENT table is stored in site 1 and COURSE able is stored in site 2. The tables are not fragmented and the results are stored in site 3. Every student is assigned to one course.

STUDENT(Sid, StudentName, Address, Grade, CourseID)
1000 records, each record is 50 bytes long
Sid: 5 bytes, StudentName;10 bytes, Address: 20 bytes

COURSE(Cid, CourseName)

500 records, each record is 30 bytes long
Cid: 5 bytes, CourseName:10 bytes

Query: Retrieve the Student Name and Course Name which the student is following.

If we are to transfer STUDENT and COURSE relations into node 3 and perform join operation, how many bytes need to be transferred? Explain your answer.

Activity

Suppose STUDENT table is stored in site 1 and COURSE able is stored in site 2. The tables are not fragmented and the results are stored in site 3. Every student is assigned to one course.

STUDENT(Sid, StudentName, Address, Grade, CourseID)
1000 records, each record is 50 bytes long
Sid: 5 bytes, StudentName;10 bytes, Address: 20 bytes

COURSE(Cid, CourseName)

500 records, each record is 30 bytes long
Cid: 5 bytes, CourseName:10 bytes

Query: Retrieve the Student Name and Course Name which the student is following.

If we are to transfer STUDENT table into site 2, and then execute join and send result into site 3, how many bytes need to be transferred? Explain your answer.

Activity

Suppose STUDENT table is stored in site 1 and COURSE able is stored in site 2. The tables are not fragmented and the results are stored in site 3.

STUDENT(Sid, StudentName, Address, Grade, CourseID)
1000 records, each record is 50 bytes long
Sid: 5 bytes, StudentName;10 bytes, Address: 20 bytes

COURSE(Cid, CourseName)

500 records, each record is 30 bytes long
Cid: 5 bytes, CourseName:10 bytes

Query: Retrieve the Student Name and Course Name which the student is following.

If we are to transfer COURSE table into site 1, and then execute join and send result into site 3, how many bytes need to be transferred? Explain your answer.

4.5 NoSQL Characteristics related to Distributed Databases and Distributed System

1. Scalability

- NoSQL databases are typically used in applications with high data growth.
- Scalability is the potential of a system to handle a growing amount of data.
- In Distributed Databases, there are two strategies for scaling a system.
  - Horizontal scalability: When the amount of data increases, distributed system can be expanded by adding more nodes into the system.
  - Vertical scalability: Increasing the storage capacity of existing nodes.
- It is possible to carry out horizontal scalability while the system is on operation. We can distribute the data among newly added sites without disturbing the operations of system.

4.5 NoSQL Characteristics related to Distributed Databases and Distributed System

2. **Availability, Replication and Eventual Consistency:**

- Most of the applications that are using NoSQL DBs, require availability.
- It is achieved by replicating data in several nodes.
- With this technique, even if one node fails, the other nodes who have the replication of same data will response to the data requests.
- Read performance is also improved by having replicas. When the number of read operations are higher, clients can access the replicated nodes without making a single node busy.

4.5 NoSQL Characteristics related to Distributed Databases and Distributed System

Availability, Replication and Eventual Consistency Cont.:

- But having replications may not be effective for write operations because after a write operation, all the nodes having same data item should be updated in order to keep the system consistent.
- Due to this requirement of updating all the nodes with the same data item, the system can get slower.
- However, most of the NoSQL applications prefer ***eventual consistency***.
- Eventual consistency will be discussed in next slide.

4.5 NoSQL Characteristics related to Distributed Databases and Distributed System

Availability, Replication and Eventual Consistency Cont.:

- Eventual Consistency

This means that at any time there may be nodes with replication inconsistencies but if there are no further updates, eventually all the nodes will synchronise and will be updated to the same value.

For example, If Kamal updates the value of **Z** to 10, it will be updated in the *node A*. But if Saman accesses the value of **Z** from *node B* the value will not be 10; as the change hasn't propagated from *node A* to *node B*. After sometime, when Saman access the value of **Z** from *node B*, then it will have the value 10. This means that Saman will eventually see the change in value **Z** made by Kamal. This is the eventual consistency.

4.5 NoSQL Characteristics related to Distributed Databases and Distributed System

3. **Replication Models**:

- The main replication models that are used in NoSQL context is master-slave and master-master replication.
  - *Master-slave replication*: The primary node refers to as master is responsible for all write operations. Then the updates are propagated to slave nodes keeping the eventual consistency. There can be different techniques for read operation. One option is making all reads on master node. Another option would be making all reads on slave nodes. But with this second option, there is no guarantee for all reads to have the same value on all data item after accessing several nodes. (Because the system gets consistent eventually)

4.5 NoSQL Characteristics related to Distributed Databases and Distributed System

Replication Models Cont.:

- *Master-master replication*: All the nodes are treated similarly. Reads and writes can be performed on any of the nodes. But it is not assured that all reads done on different nodes see the same value. Since it is possible for multiple users to write on a single data item at the same time, system can be temporarily inconsistent.

4.5 NoSQL Characteristics related to Distributed Databases and Distributed System

4. **Sharding of Files**:

- We have discussed the concept sharding in slide 55.
- In many NoSQL applications, there can be millions of data records accessed by thousands of users concurrently.
- Effective responses can be provided by storing partitions of data in several nodes.
- By using the technique called sharding (horizontal partitioning), we can distribute the load across multiple sites.
- Combination of sharding and replication improves load balancing and data availability.

4.5 NoSQL Characteristics related to Distributed Databases and Distributed System

5. High-Performance Data Access:

- In many NoSQL applications, it might be necessary to find a single data value or a file among billions of records.
- To achieve this, techniques such as hashing and range partitioning are used.
    - *Hashing*: A hash function h(k) applied on a given key K, provides the location of a particular object.
    - *Range partitioning*: Object's location can be identified from range of key values. For example, location i would hold the objects whose key values K are in the range $Ki_{min} \leq K \leq Ki_{max}$.
- We can use other indexes to locate objects based on attribute conditions (different from the key K).

Activity

Fill in the blanks with the most suitable word given.

(horizontal, vertical, eventual consistency, consistency, master, slave, availability, usability)

_____ scalability can be performed while the system is on operation.

A relaxed form of consistency preferred by most of the NoSQL systems is known as _____.

In master-slave replication model, _____ is used as the source of write operations.

Load balancing and _____ can be achieved in a system which uses the combination of sharding and replication.

Summary

Distributed Database Concepts, Components and Advantages

Types of Distributed Database Systems

Distributed Database Design Techniques

Fragmentation, Replication and Allocation, Distribution Models

Summary

Query Processing and Optimization in Distributed Databases

Distributed Query Processing, Data Transfer Costs of Distributed Query Processing

NoSQL Characteristics related to Distributed Databases and Distributed Systems