. Write short notes on: [2+4]

a) Parallel databases [2×3]
b) Data warehouse

***

9. Write short notes on: [2×3]

a) Features of Object Oriented Database.
b) Parallel database architecture.

***

discuss on the shadow paging technique for recovery. [4+4]

11. Write short notes on [4+4]

a) Important of Data Warehouses
b) Spatial database.

***

9. Write short notes on: [2×3]

a) Fragmentation in distributed databases.
b) Object Relational Mapping (ORM)

***

10. Write short notes on: [3+3]

a) Distributed database
b) Remote Backup System

***

9. Write short notes on: [3+3]

a) Data Warehousing b) Parallel database architecture

***

9. a) Write about data warehouse with its components. [4]

b) Write about spatial database. [2]

***

10. a) Explain homogenous and heterogeneous distributed database. [4]

b) What is Spatial Database System? [2]

***

10. Write short notes on: [3+3]

a) Spatial database b) Remote Backup System

***

9. a) Describe briefly about object oriented database. [3]

b) Explain the differences between homogenous and heterogeneous distributed database. [3]

***

9. Write short notes on the following: [3×2]

i) Distributed database system
ii) Spatial database system

***

10. What is the significance of object-oriented databases? Briefly explain parallel database architectures. [3+3]

***

10. Explain the importance of data warehouse in decision making. Write the application areas of spatial database. [3+3]

***

a) Briefly explain horizontal and vertical fragmentation in distributed databases. [3]
b) Write a short note on Data warehouse and associated applications. [3]

***

10. a) What is object-oriented databases? Explain briefly. [3]
b) Explain the benefit of parallel database? [3]

11. a) What is the advantage of an object-oriented database compared to a traditional relational database? [3]
b) What is horizontal and vertical data fragmentation? [3]

***

11. a) What is an ORM? [3]
b) What is the difference between a homogeneous and a heterogeneous distributed database? [3]

***

Object-Oriented Database (OODB)

-> An Object-Oriented Database (OODB) is a type of database that stores data in the form of objects, similar to object-oriented programming (OOP). It combines the features of databases with object-oriented principles such as encapsulation, inheritance, and polymorphism.

Key Features

-> **Objects & Classes** – Data is stored as objects, which belong to specific classes.

-> **Encapsulation** – Objects store both data and methods that operate on the data.

-> **Inheritance** – New classes can inherit properties and behaviors from existing ones.

-> **Polymorphism** – Objects can take multiple forms, allowing flexible data manipulation.

-> **Complex Data Types** – Supports storage of multimedia, graphs, and hierarchical data structures.

Advantages

-> Efficient for complex applications like CAD, AI, and multimedia.

-> Supports reusable and modular code.

Disadvantages

-> More complex than relational databases.

-> Limited standardization compared to SQL-based databases.

-> Can be slower for simple transactions.

Distributed Database Management System (DDBMS)

-> A Distributed Database Management System (DDBMS) is a software system that manages a database spread across multiple locations, either on different computers or networks. It ensures that the distributed data appears as a single database to users.

Key Features

-> **Data Distribution** – The database is stored across multiple sites but managed collectively.

-> **Transparency** – Users experience the system as a single database despite data distribution.

-> **Replication & Fragmentation** – Data can be duplicated (replication) or divided into smaller parts (fragmentation) for efficiency.

-> **Fault Tolerance** – Ensures system reliability by distributing data across multiple nodes.

-> **Concurrency Control** – Manages multiple transactions simultaneously without conflicts.

Advantages

-> **Improved Reliability** – Failure of one site does not affect the entire system.

-> **Faster Access** – Data is stored closer to users, reducing response time.

-> **Scalability** – Easily expands by adding more nodes.

-> **Load Balancing** – Distributes workload across multiple servers for efficiency.

Disadvantages

-> **Complexity** – Managing distributed databases is more challenging.

-> **Security Risks** – More vulnerable to cyber threats due to multiple access points.

-> **Data Synchronization Issues** – Ensuring consistency across multiple locations can be difficult.

Types of Distributed Database Management System (DDBMS)

1. Homogeneous DDBMS

-> All databases in the system use the same DBMS software.

-> Follows a uniform schema and operates under a single administration.

-> Easier to manage, synchronize, and maintain consistency.

-> Example: A multinational company using MySQL across all branches.

2. Heterogeneous DDBMS

-> Different sites use different DBMS software and data models.

-> May have varying schemas and need middleware for communication.

-> More flexible but complex to manage due to differences in data representation.

-> example: A bank using Oracle in one branch and MongoDB in another.

Distributed Database Storage

-> In a Distributed Database, data is stored across multiple sites to improve efficiency, availability, and reliability. The way relations (tables) are stored in a distributed database follows two main approaches:

1. Replication

-> A complete copy of the database or specific relations is stored at multiple sites.

-> Ensures high availability and fault tolerance.

-> Can be Full Replication (entire database at each site) or Partial Replication (selected data at certain sites).

Advantages:

-> Improves data availability and reliability.

-> Reduces access time for local users.

Disadvantages:

-> Requires synchronization to maintain consistency.

-> Increases storage requirements.

2. Fragmentation

-> The database is divided into smaller pieces (fragments) and stored at different sites.

-> Each site stores only a relevant portion of the database.

-> Ensures efficient access and reduces data transfer costs.

Types of Fragmentation:

1. Horizontal Fragmentation:

-> Divides a relation (table) row-wise based on certain conditions.

-> Each fragment contains a subset of rows but all columns.

-> Useful when different sites need specific data based on criteria (e.g., region-wise sales data).

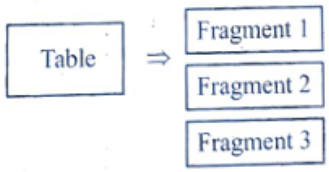


Figure: Horizontal data fragmentation

Consider the relation:

Customer_Id	Name	Area	Payment Type	Sex
1	Bob	London	Credit Card	M
2	Mike	Manchester	Cash	M
3	Ruby	London	Cash	F

Horizontal fragmentation are subsets of tuple (rows)

Fragment 1

Customer_Id	Name	Area	Payment Type	Sex
1	Bob	London	Credit Card	M
2	Mike	Manchester	Cash	M

Fragment 2

Customer_Id	Name	Area	Payment Type	Sex
3	Ruby	London	Cash	F

2. Vertical Fragmentation:

-> Divides a relation column-wise, keeping only necessary attributes at each site.

-> Each fragment must include a primary key to allow reconstruction.

-> Useful when different sites require different attributes of the same table.

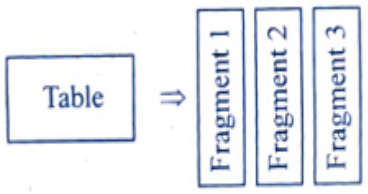
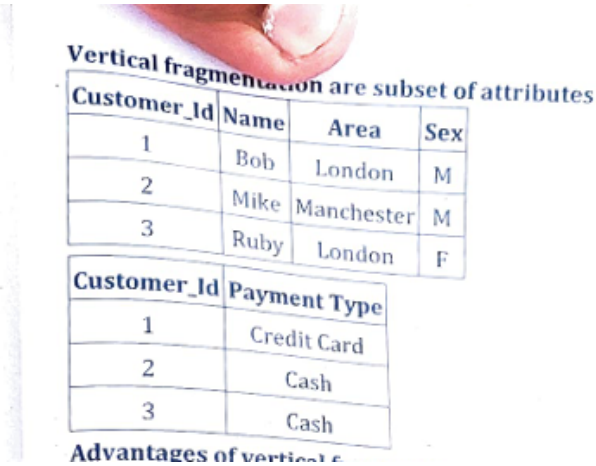


Figure: Vertical data fragmentation



Vertical fragmentation are subset of attributes

Customer_Id	Name	Area	Sex
1	Bob	London	M
2	Mike	Manchester	M
3	Ruby	London	F

Customer_Id	Payment Type
1	Credit Card
2	Cash
3	Cash

Advantages of vertical f

3. Hybrid (Mixed) Fragmentation:

-> A combination of horizontal and vertical fragmentation.

-> First, horizontal fragmentation is performed, and then vertical fragmentation is applied to each subset.

-> Ensures maximum efficiency by distributing only necessary data.

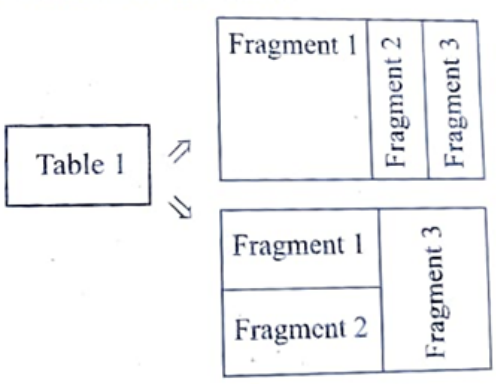


***Figure:** Hybrid data fragmentation*

Parallel Database Management System (Parallel DBMS)

-> A Parallel DBMS is a type of database management system that uses multiple processors and storage devices to process queries faster by dividing tasks among multiple units. It improves performance, scalability, and fault tolerance in large-scale database applications.

Parallel Database Architectures

Parallel databases can be classified into four major architectures based on how processors and memory interact:

1. Shared Memory Architecture

-> All processors share a common memory and storage (disk).
-> Communication between processors happens via shared memory.
-> Uses a single DBMS instance.

Advantages:

✓ Fast communication between processors.
✓ Easy to program and manage.

Disadvantages:

✕ Scalability is limited due to memory contention.
✕ A single memory failure can affect all processors.

2. Shared Disk Architecture

-> All processors have their own memory but share a common disk (storage).
-> The DBMS ensures consistency through a locking mechanism.

Advantages:

✓ Better fault tolerance than shared memory.
✓ Scalability is better as processors have independent memory.

Disadvantages:

✗ Contention on disk access can slow down performance.

✗ High coordination overhead is required for consistency.

3. Shared Nothing Architecture

-> Each processor has its own memory and disk (no sharing).

-> Each processor works independently, and data is partitioned among them.

Advantages:

✓ Highly scalable (ideal for large distributed systems).

✓ No contention on memory or disk.

✓ High fault tolerance (failure of one node does not affect others).

Disadvantages:

✗ Complex to manage and requires sophisticated load balancing.

✗ Query optimization across nodes can be challenging.

4. Hierarchical Architecture

-> A combination of shared memory, shared disk, and shared nothing architectures.

-> Typically used in multi-level systems where multiple nodes are organized in a tree-like structure.

-> Higher-level nodes manage coordination, while lower nodes handle computations.

Advantages:

✓ Optimized for complex, large-scale systems.

✓ Can balance load dynamically.

Disadvantages:

✗ Highly complex and expensive to implement.

✗ Requires advanced data distribution strategies.

Data Warehouse

-> A Data Warehouse (DW) is a centralized repository that stores large volumes of structured data collected from multiple sources. It is designed for analytical processing enabling businesses to make data-driven decisions.

Key Characteristics

-> **Subject-Oriented** – Organized around key business subjects (e.g., sales, customers).

-> **Integrated** – Combines data from multiple sources into a unified format.

-> **Time-Variant** – Stores historical data for trend analysis.

-> **Non-Volatile** – Data is read-only and does not change frequently.

Architecture of Data Warehouse

-> **Data Sources** – Operational databases, external sources, and other applications.

-> **ETL Process (Extract, Transform, Load)** – Data is extracted, transformed, and loaded into the warehouse.

-> **Data Storage** – Centralized storage for historical and current data.

-> **OLAP (Online Analytical Processing)** – Enables multidimensional data analysis.

-> **BI Tools & Reporting** – Used for data visualization and decision-making.

Advantages

✓ Improves decision-making with historical data analysis.

✓ Enhances data consistency and quality.

✓ Supports complex queries efficiently.

Disadvantages

✕ Expensive to implement and maintain.

✕ Requires significant processing power and storage.

✕ Data integration from different sources can be complex.

Object-Relational Mapping (ORM)

-> Object-Relational Mapping (ORM) is a technique that allows developers to interact with a relational database using object-oriented programming languages. It maps database tables to objects, making database operations easier and more intuitive.

Key Features

-> **Abstraction** – Developers interact with the database using objects instead of SQL queries.

-> **Automatic Query Generation** – ORM frameworks generate SQL queries automatically.

-> **Schema Mapping** – Maps database tables to class objects and rows to instances.

-> **Data Manipulation** – CRUD operations (Create, Read, Update, Delete) are performed through object methods.

Advantages

✓ Reduces the need for writing complex SQL queries.

✓ Improves code readability and maintainability.

✓ Provides database independence by supporting multiple DBMS.

Disadvantages

✕ Can introduce performance overhead compared to raw SQL.

✕ Limited flexibility for complex queries and optimizations.

✕ Learning curve for developers unfamiliar with ORM concepts.

ex: Django ORM (Python)

Spatial Database System

-> A Spatial Database System is a type of database system that is designed to store, manage, and query spatial data (data related to objects in space, such as locations, shapes, and geometric properties). It extends traditional databases by incorporating spatial data types, indexing, and query functions.

Key Features:

1. **Spatial Data Types** – Supports geometric objects like points, lines, and polygons.
2. **Spatial Indexing** – Uses structures like R-trees and Quad-trees for efficient spatial queries.
3. **Spatial Queries** – Enables location-based queries (e.g., "Find all restaurants within 5 km").
4. **GIS Integration** – Often integrated with Geographic Information Systems (GIS).
5. **Spatial Relationships** – Supports operations like intersection, containment, and proximity.

Examples of Spatial Databases:

- PostGIS(Extension for PostgreSQL)
- Oracle Spatial
- MySQL Spatial Extensions
- Microsoft SQL Server with Spatial Data

Spatial databases are widely used in GIS applications, urban planning, navigation systems, and environmental monitoring.