CHAPTER - 11

ADVANCED DATABASE CONCEPTS

11.1 Object Oriented Model

This data model is a method of representing real world objects. Object oriented model considers each object in the world as objects and isolates it from each other. Object oriented model groups its related functionalities together and allows inheriting its functionality to others related sub groups.

Element of Object Oriented Model

- **Objects**

  The real world entities and situations are represented as objects in the object oriented database model.

- **Attributes and Method**

  Every object has certain characteristics. These are represented using attributes. The behavior of the object is represented using methods.

- **Class**

  Similar attributes and methods are grouped together using a class. An object can be called as an instance of the class.

- **Inheritance**

  A new class can be derived from the original class. The derived class contains attributes and methods of the original class as its own.

  Example:

  Shape, circle, Rectangle and Triangle are all objects in this model.

  Circle has the attributes center and radius. Rectangle has the attributes length and breadth. Triangle has the attributes base and height. The objects circle, rectangle and triangle inherit from the object shape

11.2 Object Relational Model

A data modeling and manipulation technique that connects object-oriented programming and conventional relational databases is called the Object-Relational Model (ORM).

A programmer can specify a mapping between objects in the programming language and tuples in database relations using object-relational mapping technologies.

A selection condition on an attribute of an object, or set of objects, can be used to retrieve them. Based on the selection conditions, relevant data are then retrieved from the underlying database, and one or more objects are created from the retrieved data according to the predetermined mapping between objects and relations.

The mapping from objects to relations is then used to appropriately update, insert, or delete tuples in the database. The software can change retrieved objects, create new objects, or declare that an item is to be removed, and then issue a save command.

Object-relational mapping systems' main objective is to make it easier for programmers to construct applications by giving them access to an object model while yet keeping the advantages using a robust relational database underneath. Object-relational systems can also offer significant speed improvements over direct access to the underlying database when working with objects cached in memory.

Developers can map database tables to classes, database columns to object properties, and database relationships to object associations using the tools and libraries offered by ORM frameworks. Metadata, which details the connections between database tables, columns, and rows, is often used for this mapping.

Some popular ORM frameworks include Hibernate for Java, Entity Framework for .NET, SQLAlchemy for Python, and Doctrine for PHP.

Advantages:

i. **Object-oriented approach**

ORM provides a more object-oriented approach to database access, allowing developers to work with database data as objects rather than writing low-level SQL queries. This simplifies database access and makes it easier to write maintainable code.

ii. **Reduced development time**

ORM can reduce development time by automating the process of mapping database tables to classes, database columns to object properties, and database relationships to object associations. This allows developers to focus on writing business logic and application behavior rather than writing low-level database access code.

iii. **Portability**

ORM can make applications more portable by abstracting the underlying database technology. This makes it easier to switch between different database vendors and even different database types, such as SQL and NoSQL databases.

iv. **Improved security**

ORM can improve security by providing built-in protection against SQL injection attacks, a common security vulnerability in database-driven applications.

v. **Scalability**

ORM can improve scalability by providing support for caching, lazy loading, and other performance optimization techniques. This can help to improve application performance and reduce the load on the database server.

vi. **Easier testing**

ORM can make it easier to write unit tests and integration tests by providing a more object-oriented approach to database access. This can simplify the testing process and help to ensure that the application behaves correctly under different scenarios.

Disadvantages

i. **Performance overhead**

ORM introduces a performance overhead as it needs to translate between object-oriented code and relational databases. This can slow down database access and impact application performance, especially when working with large datasets.

ii. **Complex configurations**

ORM can be complex to configure and set up, especially when working with complex data models or non-standard database schemas. This can require additional time and effort during the development process.

iii. **Limited control**

ORM abstracts the underlying database, which can limit the level of control that developers have over database access and optimization. This can make it harder to optimize database queries for specific use cases or to use advanced database features.

iv. **Lack of flexibility**

ORM can be less flexible than writing custom SQL queries, especially when working with complex data models or non-standard database schemas. This can limit the ability to optimize database access for specific use cases.

v. **Learning curve**

ORM can require additional learning for developers who are not familiar with object-oriented programming or relational databases. This can slow down the development process and increase the learning curve for new developers.

11.3 Distributed Databases

A distributed database is a collection of multiple interconnected databases which are spread physically across various locations that communicate via a computer network. Transactions may access data at one or more sites. The main differences between the centralized and distributed system is the data reside in on single location where as in the later the data reside in a several locations. A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

General Architecture of Pure Distributed Databases

The general schema architecture of a DDB is shown in Figure below. The enterprise is given a consistent, unified view that reveals the logical structure of the underlying data across all nodes. The global conceptual schema (GCS), which offers network transparency, is a representation of this view. Each node is displayed as having its own local internal schema (LIS) based on physical organizational details at that particular location in order to account for potential heterogeneity in the DDB.The local conceptual schema (LCS) describes how data are logically organized at each site. The fragmentation and replication transparency are provided by the GCS, LCS, and their underlying mappings.

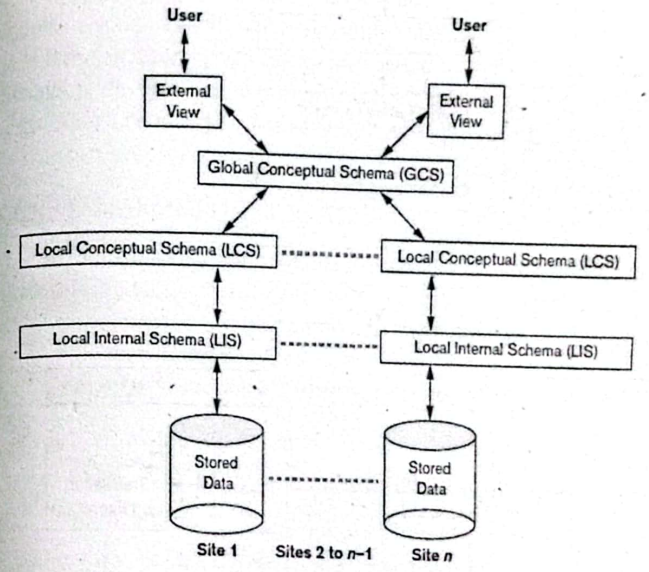

Figure: Schema architecture of distributed databases.

The component architecture of a DDB is shown in the figure below. In order to validate and impose defined constraints, the global query compiler refers to the global conceptual schema from the global system catalog. The global query optimizer creates optimal local queries from global queries by referring to both

global and local conceptual schemas. Using a cost function that calculates costs based on response times (CPU, I/O, and network latencies) and estimated sizes of intermediate outcomes, it assesses all potential solutions. The optimizer chooses the candidate with the lowest cost for execution after computing the costs for each candidate. Each local DBMS would have a local system catalog that holds the local schemas, a local query optimizer, transaction manager, and execution engines. Together with the local transaction manager at each of those sites, the global transaction manager is in charge of coordinating the execution across many locations.

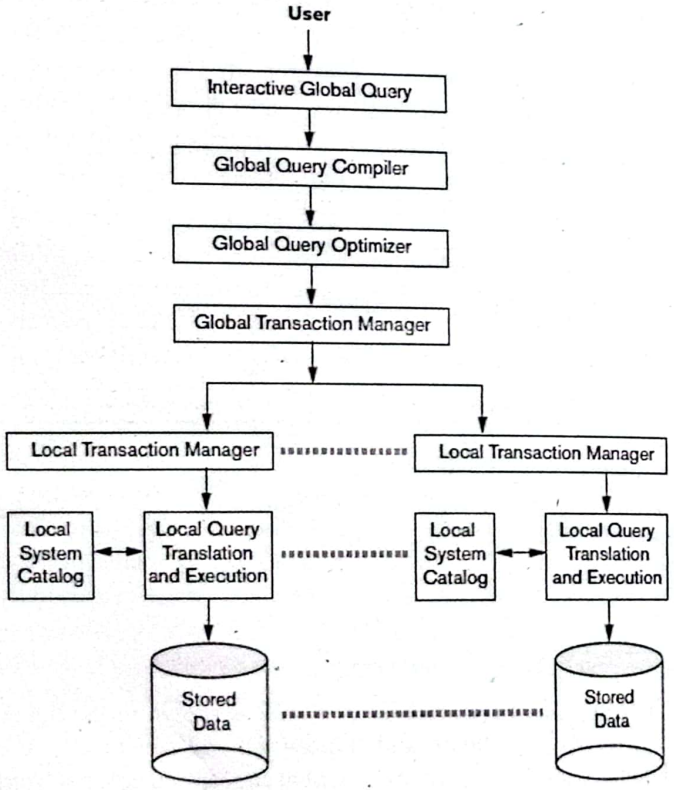

Figure: Component architecture of distributed databases.

Types:

i. Homogenous Database

ii. Heterogenous Database

11.3.1 Homogeneous Distributed Database

In homogeneous database, all sites have identical software and are aware of each other and agree to co-operate in processing user requests. Each site surrenders part of its autonomy in terms of right to change schemas or software. Database appears to user as a single system.

11.3.2 Heterogenous Distributed Database

In heterogenous distributed database different sites may use different schemas and software. Difference in schema is a major problem for query processing. Difference in software is a major problem for transaction processing. Sites may not be aware of each other and may provide only limited facilities for co-operation in transaction processing.

11.3.3 Distributed Database Storage

In distributed data storage information is stored on more than one node, often in a replicated fashion. Assume relational data model stored in the database. Two approaches to store relation in distributed database are:

i. **Replication**

System maintains multiple copies of data stored in different sites for faster retrieval and fault tolerance.

ii. **Fragmentation**

Relation is partitioned into several fragments stored in distinct sites. Replication and fragmentation can be combined. Relation is partitioned into several fragments, system maintains several identical replicas of each fragments

Data Replication

A relation or fragment of a relation is replicated if it is stored redundantly in two or more sites. Full replication of a relation is

the case where the relation is stored at all sites. Fully redundant databases are those in which every sites contains a copy of the entire database.

Advantages of Replication

i. **Availability**

Failure of site containing relation r does not result in unavailability of r if replicas exist.

ii. **Parallelism**

Queries on r may be processed by several nodes in parallel.

iii. **Reduced data transfer**

Relation r is available locally at each site containing a replica of r.

Disadvantages of Replication

i. **Increased cost of updates**

Each replica of relation r must be updated.

ii. **Increased complexity of concurrency control**

Concurrent updates to distinct replicas may lead to inconsistent data unless special concurrency control mechanisms are implemented. One solution may be to choose one copy and apply concurrency control operations on primary copy.

Data Fragmentation

Data fragmentation is division of relation r into fragments r1, r2, r3, ...rn which contains sufficient information to reconstruct relation r. These fragments may be stored at different locations. The data fragmentation process should be carried out in such a way that the reconstruction of original database from the fragments is possible

Types of Data Fragmentation

1. **Horizontal Data Fragmentation**

Each tuple of r is assigned to one or more fragments

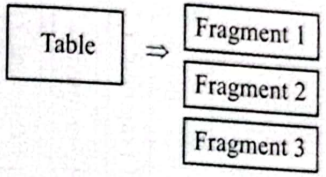

Figure: Horizontal data fragmentation

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

Advantages of horizontal fragmentation

i. Allows parallel processing on fragments of a relation

ii. Allows a relation to be split so that tuples are located where they most frequently accessed

2. **Vertical Data Fragmentation**

The schema for relation r is split into several smaller schemas. This is the vertical subset of a relation that means a relation/table is fragmented by considering the columns of it.

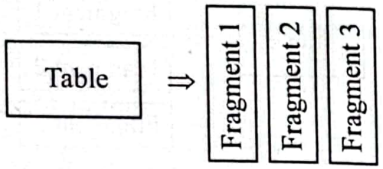

Figure: Vertical data fragmentation

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 fragmentation

i. Allows tuples to be split so that each part of the tuple is stored where it is most frequently accessed.

ii. Tuple id attribute allows efficient joining of vertical fragments.

iii. Allows parallel processing on a relation

3. Hybrid Data Fragmentation

This is the combination of horizontal as well as vertical fragmentation. This type of fragmentation will have horizontal fragmentation to have subsets of data to be distributed over the DB, and vertical fragmentation to have subset of columns of the table.

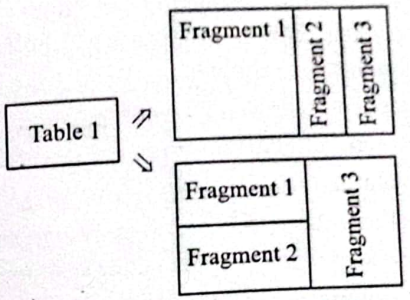

Figure: Hybrid data fragmentation

As we observe in above diagram this type of fragmentation can be done in any order. It does not have any particular order. It is solely based on the user requirements. But it should satisfy fragmentation conditions.

Reasons for Distributed Database

Distributed database is used because of :

i. Business unit autonomy and distribution.

ii. Data sharing

iii. Data communication costs.

iv. Data communication reliability and costs.

v. Multiple application vendors.

vi. Database recovery.

vii. Transaction and analytic processing

11.4 Concept of Data Warehouse

A data warehouse is a collection of corporate information and data derived from operational systems and external data sources. A data warehouse is designed to support business decisions by allowing data consolidated analysis and reporting at different aggregate levels. Data is populated into data warehouse through the process of extraction transformation loading.

A single complete and consistent store of data obtained from a variety of different sources made available to end users in what they can understood and use in a business context is called warehouse. A data warehouse is subjected oriented integrated,

time-variant nonvolatile collection of data in support of management's decision-making process. Following are the characteristics of data warehouse:

i. Subject oriented
ii. Integrated
iii. Non-Volatile
iv. Time- Variant
v. Summarized
vi. Large Volume
vii. Non normalized
viii. Metadata
ix. Data Sources

Subject Oriented

A data warehouse is subject oriented because it provides information around a subject rather than the organization's ongoing operations. These subjects can be product, customers, suppliers, sales, revenue, etc. A data warehouse does not focus on the ongoing operations, rather it focuses on modelling and analysis of data for decision making. Excludes data not useful in decision support process.

Integrated

A data warehouse is constructed by integrating data from heterogeneous sources such as relational databases, flat files, etc. This integration enhances the effective analysis of data. Data preprocessing are applied to ensure consistency. The data warehouse is a centralized consolidated database that integrates data from entire organization. Multiple sources, diverse sources and diverse formats

Time Variant

The data collected in a data warehouse is identified with a particular time period. The data in a data warehouse provides information from the historical point of view. The data warehouse represents the flow of data through time. Every key structure contains either implicitly or explicitly an element of time.

Non-volatile

Non-volatile means the previous data is not erased when new data is added to it. A data warehouse is kept separate from the operational database and therefore frequent changes in operational database is not reflected in the data warehouse.

Note: A data warehouse does not require transaction processing, recovery, and concurrency controls, because it is physically stored and separate from the operational database.

Data warehouse represents company's entire history. Near term history is continually added to it . It is always growing. Most of the warehouse support terabyte databases and multiprocessors. Read only database is used for data analysis and query processing Data warehouse requires two operations in data accessing

- Initial loading of data
- Access of data.

11.4.1 Rules of Data Warehouse

- Data warehouse and operational Environments ar separated.
- Data is integrated.
- Contains historical data over a long period of time.
- Data is a snapshot data captured at a given point in time.
- Data is subject oriented.
- Mainly read only with periodic batch updates.
- Development lifecycle has a data driven approach versus traditional process driven approach
- Data contains several levels of detail. Current, old, lightly summarized, highly summarized.
- Environment is characterized by read only transactions to very datasets.
- System that traces data sources transformations and storage.
- Metadata is a critical component. Source, transformation. integration, storage, relationships, history etc.

11.4.2 Need for Data Warehousing

- Industry has huge amount of operational data. Knowledge worker wants this data into useful information. This information is used by them to support strategic decision making.
- It is a platform for consolidated historical data for analysis.
- It stores data for good quality so that knowledge worker can make correct decisions.
- From business perspective:
  - It is latest marketing weapon
  - Helps to keep customers by learning more about their needs
  - Valuable tool in today's competitive fast evolving world

11.4.3 Data Warehouse Architecture

Data warehouse server

A data warehouse server is the physical storage used by a data warehouse system. The physical storage consists of almost always a relational DBMS and rarely flat files. Various processed data and other relevant information that comes from several applications and sources are stored in a data warehouse server where it is organized for future business analysis and user query purposes.

Online Analysis Processing Servers

Online analysis processing servers are used to support and operate on multi-dimensional data structures. It allows managers, and analysts to get an insight of the information through fast, consistent, and interactive access to information.

Clients

- Query and reporting tools
- Analysis tools
- Data mining tools

SOLUTION TO EXAMS' AND OTHER IMPORTANT QUESTIONS

1. Explain the advantage and disadvantage of object oriented database over relational database. [2021 Spring]

ANS: Advantages of Object-Oriented Database (OODB) over Relational Database (RDB):

i. Enhanced Modeling Capabilities

In comparison to RDB, OODB offers improved modeling capabilities since it enables more accurate depiction of complex real-world objects, relationships, and behaviors. This makes it ideal for systems like multimedia systems, CAD/CAM programs, and scientific simulations where the object structure is a key factor.

ii. Native Object Persistence

Objects can be stored directly in the database without the requirement for object-relational mapping or laborious normalization because OODB inherently supports object persistence. This streamlines development and lessens the impedance mismatch between the database and the object-oriented programming language.

iii. Complex Data Structures

With the aid of OODB, complex data structures, such as those involving object association, composition, and inheritance, can be stored. This encourages more accurate depictions of actual relationships, the encapsulation of data and activity, and more organic modeling of complex domain concepts.

iv. Improved Performance

For some applications, OODB delivers better performance. Compared to RDB's complicated join processes, direct access to objects enables quicker retrieval and navigation. Additionally, OODB does away with the requirement for pricey normalization procedures, which may enhance performance for specific query patterns.

Disadvantages of Object-Oriented Database (OODB) over Relational Database (RDB):

i. Lack of Standardization

Like SQL in RDB, OODB does not have a standardized query language. As a result, working with various OODB systems may be more difficult, and system compatibility may be constrained.

ii. Limited Industry Support

Compared to RDB, OODB has not experienced extensive industrial adoption. There is an absence of reliable and developed OODB solutions because the vast majority of database systems and tools are created for relational databases. This may present difficulties in terms of support, compatibility, and availability with current software ecosystems.

iii. Complexity and Learning Curve

In comparison to RDB, designing, implementing, and maintaining OODB systems can be more difficult. For developers and administrators, mastering the complex nature of mapping objects to a database schema as well as object-oriented concepts like inheritance and polymorphism can be challenging.

iv. Data Integrity and Consistency

Due to the absence of standardized procedures like constraints and triggers found in RDB, OODB may have trouble assuring data consistency and integrity. In OODB, it may be more challenging to ensure data integrity in complicated object interactions and to enforce business rules, requiring customized implementation.

2. Under which situations will be beneficial to have replication or fragmentation of data? Explain with suitable example. [2014 Fall]

ANS: Depending on the unique requirements and characteristics of the system, both data replication and data fragmentation

might be advantageous in various circumstances. The benefits of each strategy are explained below, along with appropriate examples:

i. Data Replication

Creating and maintaining numerous copies of the same data across several locations or nodes in a distributed system is known as data replication. It has a number of advantages, including as increased availability, fault tolerance, and decreased network latency.

ii. Improved Availability

By providing numerous independent copies of the data, replication can increase system availability. Data can still be accessed from other replicas, ensuring continuous service even if one node or location fails. This is especially beneficial in situations where high availability is essential, such online banking systems or e-commerce platforms.

iii. Load Balancing

The task can be split among several nodes using replication. The system can balance read and write activities by replicating data and letting clients access the closest replica, which lessens the burden on individual nodes and boosts speed. Data replication is used by content delivery networks (CDNs) to cache material closer to end consumers, lowering latency and enhancing user experience.

iv. Geographic Distribution

Data replication makes it possible to store data across several different places, allowing for localized access and lowering network latency. In worldwide systems with users scattered across many regions, this is advantageous. A multinational corporation might, for instance, duplicate its customer data in regional databases to offer quick and responsive services to clients in several locations around the world.

v. Data Fragmentation

In a distributed system, data fragmentation refers to the splitting of a database into smaller pieces or partitions that can be stored on several nodes or servers. A portion of the data is present in every piece. Improvements in efficiency, scalability, and data localization are just a few advantages of fragmentation.

vi. Performance

By dispersing the data among several nodes, fragmentation might enhance query performance. Faster response times can be achieved by parallelizing the execution of queries that only need a portion of the data. For instance, client data may be dispersed based on geographic regions in a sizable e-commerce system. The amount of data that has to be processed is decreased when processing orders since the system may query the pertinent fragment comprising client data from the particular region.

vii. Scalability

Data fragmentation enables the addition of new nodes to the system as the data volume increases, supporting horizontal scalability. The task can be distributed across each new node, allowing the system to manage bigger datasets. Data fragmentation is frequently used by social media platforms to divide user data across various servers and handle the increasing number of users and the data they generate.

viii. Data Localization:

When sensitive data must be stored in specific locations due to legal or privacy restrictions, fragmentation might be advantageous. For instance, due to data protection laws, patient data in a healthcare system may need to be held only inside the borders of a specific nation or region. Data fragmentation and localized database storage assure adherence to these specifications.

3. Briefly explain on the requirement of data fragmentation and replication in the context of distributed databases.

ANS: Data fragmentation is required if improving at least one of the following is our goal:

- Single-user response time
- Concurrency
- Availability
- Backup-and-restore characteristics
- Loading of data

Data replication is required in context of distributed databases because of the following advantages:

i. **Availability:** Failure of site containing relation r does not result in unavailability of r if replicas exist.

ii. **Parallelism:** Queries on r may be processed by several nodes in parallel.

iii. **Reduced data transfer:** Relation r is available locally at each site containing a replica of r.