Assignment:

Question

What is data abstraction? What are the three levels of data abstraction?

Answer

Data abstraction in technical terms is the reduction of a particular body of data to a simplified representation of the whole. In terms of Database we can explain data abstraction as a technique that enables users to view and interact with the data in a simplified and meaningful way. It involves hiding the details of the physical storage and logical structure of the database and presenting a conceptual view of the data to the user. It can be achieved through the use of data models, which provide a high-level representation of the data and the relationships between them. There are different data models that are used in databases which provide the abstractions in different levels according to their structure/model. The three levels of data abstraction is explained below:

- Physical Level:
  At the physical level, data abstraction is concerned with the implementation details of how the data is stored in the system. This includes details such as data structures, file formats, storage devices, and access methods. For example, at this level, data might be stored in a specific file format, using a particular data structure and access method. The physical level is important for system administrators and database developers, as it provides information on how the data is stored and accessed in the system.
- Logical Level:
  At the logical level, data abstraction is concerned with the meaning of the data and how it is organized. This level provides a conceptual view of the data, independent of the physical implementation. The logical level includes information such as data schemas, data models, and relationships between different data elements. For example, at this level, data might be organized into tables, with relationships between tables defined using foreign keys. The logical level is important for data analysts and developers, as it provides a way to understand the meaning of the data and how it is structured.
- View Level:
  At the view level, data abstraction is concerned with how the data is presented to the user. This level provides customized views of the data based on the user's requirements. Views can be created to provide access to specific subsets of data or to present data in a particular format. For example, at this level, a user might have a view that shows only the customer data they are interested in, with the data presented in a specific format that

is relevant to their work. The view level is important for end-users, as it provides a way to interact with the data that is tailored to their needs.

In summary, each level of data abstraction provides a different perspective on the data, from the physical implementation of the data to the conceptual representation of the data to the customized views presented to the user. These levels work together to provide a flexible and efficient way of managing and using data in complex systems.

Question

Explain different data models with examples.

Answer

The different data models are explained below:

1. Hierarchical Model: The hierarchical model organizes data in a tree-like structure, with each record having a parent-child relationship with other records. This model is useful for managing data with a clear hierarchical structure, such as organizational charts or file systems. An example of a hierarchical database management system is IBM's Information Management System (IMS).
2. Network Model: The network model is similar to the hierarchical model but allows for more complex relationships between records. In this model, records can have multiple parent and child relationships, forming a network of interconnected records. The network model is useful for managing complex relationships between data, such as those found in social networks. An example of a network database management system is Integrated Data Store (IDS).
3. Entity-Relationship Model: The entity-relationship model is a conceptual model that represents the relationships between entities. Entities can be thought of as objects or concepts in the real world, and relationships between entities are represented by lines connecting them. The entity-relationship model is useful for modeling complex data relationships and is commonly used in database design. An example of an entity-relationship model is the diagram used to design a database for a library, with entities such as "books," "authors," and "borrowers" and relationships such as "borrowed by."
4. Relational Model: The relational model is the most widely used data model in database systems. It organizes data into tables, with each table consisting of rows and columns. Each row represents a record, and each column represents a field. Relationships between tables are established using foreign keys. The relational model is useful for managing large amounts of structured data. Examples of relational database management systems include MySQL, Oracle, and Microsoft SQL Server.

5. Object-Oriented Data Model: The object-oriented data model is based on the concept of objects, which are instances of classes. Each object has its own set of properties, and objects can be related to each other using inheritance or association. The object-oriented data model is useful for managing complex data structures, such as those found in software applications. Examples of object-oriented database management systems include MongoDB and Apache Cassandra.
6. Object-Relational Data Model: The object-relational data model combines features of both the relational and object-oriented models. It allows for the storage of complex data types, such as arrays and user-defined types, and also supports object-oriented concepts such as inheritance and encapsulation. The object-relational data model is useful for managing complex data structures in applications that also require efficient query performance. Examples of object-relational database management systems include Oracle and PostgreSQL.
7. Flat Data Model: The flat data model is the simplest data model, with a single two-dimensional table that contains all the data. This model is useful for managing small amounts of data with a simple structure. An example of a flat file database is a spreadsheet that contains a list of contacts.
8. Semi-Structured Data Model: The semi-structured data model is used for data that has a flexible or varying structure, such as XML or JSON documents. In this model, data is stored in a hierarchical or graph-like structure, with some level of schema or structure imposed on the data. Examples of semi-structured data management systems include Apache CouchDB and MongoDB.
9. Associative Data Model: The associative data model is used for data that is stored as a set of key-value pairs. This model is useful for managing data that doesn't fit into a predefined structure, such as user preferences or clickstream data. An example of an associative data model is a NoSQL database that stores data in key-value pairs, such as Redis.
10. Context Data Model: The context data model is used for managing data that is specific to a particular context, such as a specific location or time period. This model is useful for managing data that needs to be analyzed in a specific context or for making decisions based on a specific set of criteria. An example of a context data model is a weather database that stores data on temperature, humidity, and precipitation for a specific location and time period.

Question

What is the difference between logical and physical data independence?

Answer

Physical Data Independence	Logical Data Independence
Definition: The ability to change the physical structure of the database without affecting the application programs that use it.	**Definition:** The ability to change the logical structure of the database without affecting the application programs that use it.
Focus: The physical schema, which describes how the data is physically stored on disk and accessed by the database management system.	**Focus:** The logical schema, which describes the logical organization of the data and the relationships between entities.
Goal: To optimize the performance and storage efficiency of the database, and to shield application programs from changes.	**Goal:** To allow for flexibility in the design and maintenance of the database, and to separate the logical view of the data from its physical implementation.
Achievability: It is easy to achieve physical data independence as compared to logical data independence.	**Achievability:** It is not easy to achieve logical data independence as compared to physical data independence.
Impact on applications: Any change at the physical level does not require changes at the application level.	**Impact on applications:** A change in the logical level requires a change at the application level.
Concerned with: The internal schema.	**Concerned with:** The conceptual schema.
Example: Changing compression techniques, hashing algorithms, and storage devices.	**Example:** Adding, modifying, or deleting a new attribute.

Question

What are the components of ER diagram? Explain the functions of various symbols used in ER diagram.

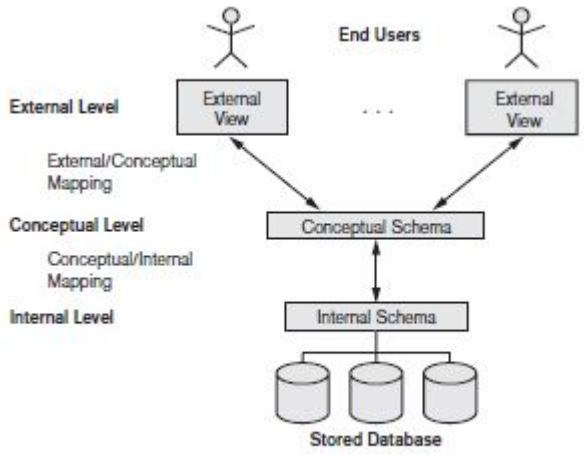
Answer

An ER (Entity-Relationship) diagram is a graphical representation of the logical structure of a database system. It is used to model the entities, attributes, and relationships between entities in a database in a clear and concise manner. ER diagrams are commonly used in database design and provide a visual representation of the database schema.

Question

Explain the three schema architecture.

The three-schema architecture is a conceptual framework that separates a database system into three different levels or schemas. This architecture is commonly used in modern database management systems (DBMS) to separate the way data is perceived by different groups of users. The three schema architecture includes the following levels:



1. External schema or view level: This is the highest level of the three-schema architecture and represents the user's view of the data. It is the way data is presented to the end-users or application programs that interact with the database. Each user or application has its own external schema, which provides a customized view of the data that meets their specific needs. External schemas define how data is seen and accessed by different users and can be modified without affecting the overall structure of the database.
2. Conceptual schema or logical level: This level defines the overall logical structure of the database. It describes the relationships between the different entities and attributes in the database and provides a conceptual framework for the entire database system. The conceptual schema represents the global view of the database, which is independent of any particular user or application. It is used as a basis for designing the database and creating the internal schema.
3. Internal schema or physical level: This is the lowest level of the three-schema architecture and represents the way data is physically stored in the database system. It defines the physical structure of the database, including storage structures, indexing, access paths, and other implementation details. The internal schema maps the conceptual schema to the physical storage details of the database. It is designed to optimize the performance of the database and provide efficient access to the data.

The three-schema architecture provides a clear separation between the way data is perceived by different users and the way data is stored physically in the database system. It allows for changes to be made at one level without affecting the other levels, providing greater flexibility, and reducing the complexity of the database system. This architecture also enables multiple views of the same data, providing a way to customize data access for different users and applications.

Question

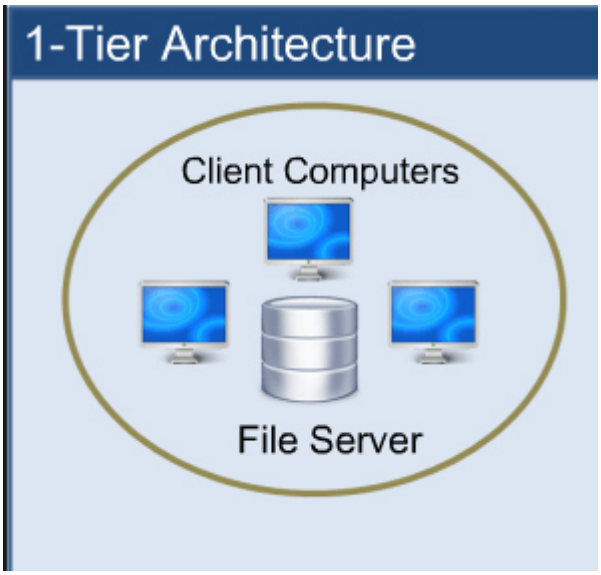
Explain client-server architecture of DBMS

Client-server architecture is a popular database architecture that separates the functions of a database system into two parts: the client and the server. The client is the end-user application or computer, and the server is the centralized system that provides data storage, processing, and management.

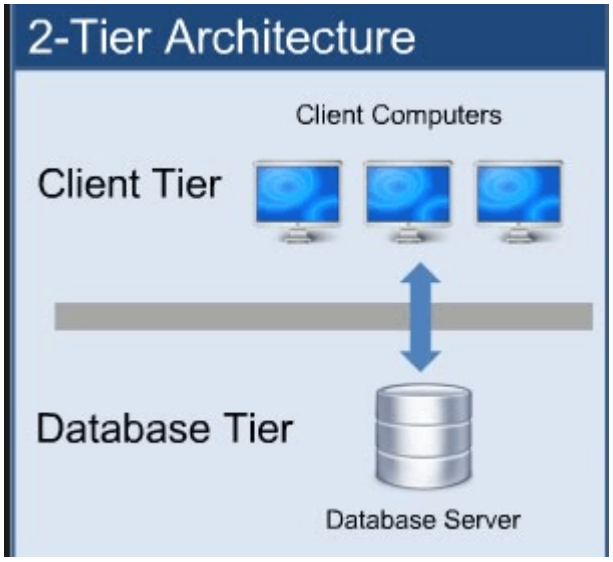
In a client-server architecture, the client sends requests to the server, which responds with the requested data or performs the requested action. The server provides a centralized location for storing data, which is accessed by multiple clients simultaneously. The clients can be desktop applications, mobile applications, web applications, or any other type of software that communicates with the server to retrieve or modify data.

There are two main types of client-server architectures:

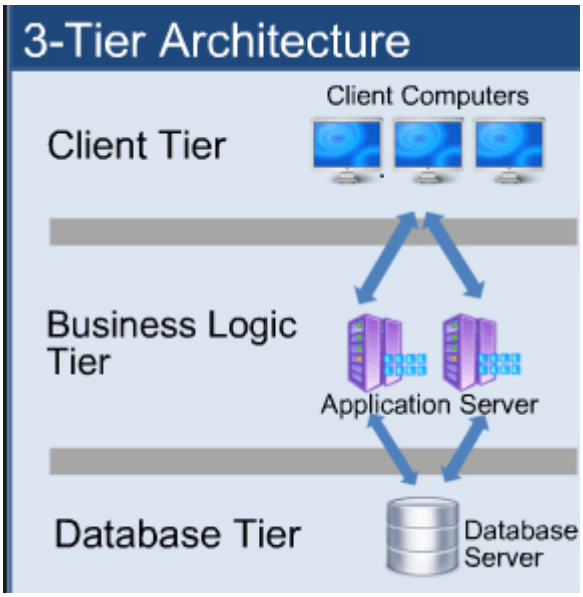
1. One-tier architecture: In this category of client server architecture, the architecture contains all kinds of settings, such as configuration setting and marketing logic, on a single device. While the diversity of services offered by 1-tier architecture makes it one of the reliable sources, handling such an architecture is difficult. This is primarily due to the data variance. It often results in replication of work. 1-tier architecture consists of several layers, such as presentation layer, business layer, and data layer, that are combined with the help of a unique software package. The data present in this layer is usually stored in local systems or on a shared drive.



2. Two-tier architecture: In this architecture, the client communicates directly with the database server. The client sends SQL queries to the server, and the server responds with the requested data. This architecture is simple and easy to implement but can become slow and inefficient as the number of users increases.



3. Three-tier architecture: In this architecture, the client communicates with an application server, which acts as an intermediary between the client and the database server. The application server processes the client's requests and sends SQL queries to the database server. The server responds with the requested data, which is sent back to the application server and then to the client. This architecture is more scalable and efficient than the two-tier architecture, as it allows for multiple application servers to handle a large number of client requests.



The advantages of using client-server architecture in database systems include:

- Centralized data management: The server provides a centralized location for storing data, ensuring data consistency and integrity.
- Improved security: The server can enforce security measures, such as access control, to protect sensitive data.
- Scalability: The client-server architecture can be easily scaled by adding more servers to handle a large number of clients.
- Efficient data processing: The server can perform complex data processing tasks, such as aggregation and sorting, which can improve the performance of the database system.
- Flexibility: The client-server architecture can support a variety of client applications, making it a versatile architecture for database systems.

Overall, the client-server architecture provides a powerful and flexible framework for managing database systems, allowing for centralized data management, improved security, and efficient data processing.

Question

Differentiate between distributed and centralized DBMS.

Answer

	Centralized DBMS	Distributed DBMS

	Centralized DBMS	Distributed DBMS
Definition	All data is stored on a single computer or server, and all database management functions are performed on that computer or server	Data is spread across multiple computers or servers, and database management functions are distributed across those computers or servers
Scalability	Limited scalability, as all data is stored on a single computer or server	Highly scalable, as data can be distributed across multiple computers or servers
Reliability	Single point of failure, as all data is stored on a single computer or server	More reliable, as data can be replicated across multiple computers or servers
Performance	Limited performance, as a single computer or server must handle all database management functions	Higher performance, as database management functions can be distributed across multiple computers or servers
Cost	Lower cost, as only a single computer or server is needed	Higher cost, as multiple computers or servers are needed
Data consistency	Easier to maintain data consistency, as all data is stored on a single computer or server	More difficult to maintain data consistency, as data is spread across multiple computers or servers
Security	Easier to secure, as all data is stored on a single computer or server	More difficult to secure, as data is spread across multiple computers or servers

---

Question

What is DDL? How it is different from DML?

Answer

DDL stands for Data Definition Language. As the name suggests, the DDL commands help to define the structure of the databases or schema. When we execute DDL statements, it takes effect immediately. The changes made in the database using this command are saved permanently because its commands are auto-committed. The following commands come under DDL language:

- **CREATE**: It is used to create a new database and its objects such as table, views, function, stored procedure, triggers, etc.

- **DROP**: It is used to delete the database and its objects, including structures, from the server permanently.
- **ALTER**: It's used to update the database structure by modifying the characteristics of an existing attribute or adding new attributes.
- **TRUNCATE**: It is used to completely remove all data from a table, including their structure and space allocates on the server.
- **RENAME**: This command renames the content in the database.

On the other hand, DML stands for Data Manipulation Language, which is used to manipulate data within a database. It includes commands for inserting, updating, and deleting data in tables.

Examples of DML commands include:

- SELECT: used to retrieve data from one or more tables
- INSERT INTO: used to insert new data into a table
- UPDATE: used to update existing data in a table
- DELETE: used to delete data from a table

The key difference between DDL and DML is that DDL deals with the structure of the database, while DML deals with the data within the database. DDL is used to create, modify, or delete database objects, while DML is used to manipulate the data stored in those objects.

Question

What is database system architecture? Describe three levels and benefits of this architecture.

Answer

Database system architecture refers to the structure or design of a database system. It typically consists of three levels of architecture, which are:

1. External Level: This level is concerned with how the data is viewed by the end users or applications. It defines the user's view of the database and provides a way to access the data. Each user or application may have a different view of the data, depending on their specific needs. The external level provides a way to maintain data security, integrity, and independence from changes made to the internal level.

2. Conceptual Level: This level is concerned with the overall logical structure of the database. It defines the relationships between the data elements and provides a high-level view of the entire database. The conceptual level provides a way to maintain data independence from changes made to the physical storage of the data.
3. Internal Level: This level is concerned with how the data is physically stored on the computer system. It defines the way the data is stored and accessed on disk, including the data structures and algorithms used to manage the data. The internal level provides a way to optimize the performance of the database system by using techniques such as indexing and caching.

The benefits of this three-level architecture include:

1. Data Independence: By separating the external, conceptual, and internal levels, it provides a way to maintain data independence between the levels. Changes made to one level will not affect the other levels, which allows for greater flexibility and easier maintenance of the database system.
2. Improved Security: By defining the user's view of the database at the external level, it provides a way to control access to the data and maintain data security.
3. Improved Performance: By optimizing the physical storage and access of the data at the internal level, it provides a way to improve the performance of the database system and reduce the time it takes to access and retrieve data.
