1. What are the advantages of using Database Management System over traditional filing system? Explain different data models with example.

Database Management System DBMS is a software designed to define, retrieve, manipulate and manage data in a database. It serves as an interface between users or applications and the underlying database, providing mechanisms for data organization, storage, retrieval, security, and integrity.

The advantages of Database Management Systems (DBMS) include:

Data redundancy and inconsistency: Redundancy is the concept of repetition of data i.e. each data may have more than a single copy. The file system cannot control the redundancy of data as each user defines and maintains the needed files for a specific application to run. There may be a possibility that two users are maintaining the data of the same file for different applications. Hence changes made by one user do not reflect in files used by second users, which leads to inconsistency of data. Whereas DBMS controls redundancy by maintaining a single repository of data that is defined once and is accessed by many users. As there is no or less redundancy, data remains consistent.

Data Integrity: DBMS ensures data integrity by enforcing constraints and rules, preventing inconsistencies and errors in the stored data.

Data Security: DBMS offers robust security features such as access control, authentication, and encryption to protect sensitive data from unauthorized access or tampering.

Data Sharing: The file system does not allow sharing of data or sharing is too complex. Whereas in DBMS, data can be shared easily due to a centralized system.

Data Scalability: DBMS can handle large volumes of data efficiently, scaling to accommodate growing data requirements without sacrificing performance.

Flexible: Database Management systems are more flexible than file processing system.

Backup and Recovery: DBMS offers backup and recovery mechanisms to safeguard data against loss or corruption, ensuring data availability and continuity.

Improved Data Access: DBMS provides efficient data access methods, including indexing and caching, to optimize data retrieval performance.

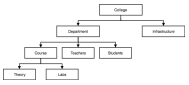
Centralized Management: DBMS centralizes data management tasks such as storage, retrieval, and maintenance, simplifying administrative tasks and reducing management overhead.

Interfaces: It provides different multiple user interfaces like graphical user interface and application program interface.

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed, and updated in a database management system. While the Relational Model is the most widely used database model, there are other models too:

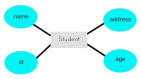
- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

1. Hierarchical Model: This database model organizes data into a tree-like structure, with a single root, to which all the other data is linked. The hierarchy starts from the root data, and expands like a tree, adding child nodes to the parent node.

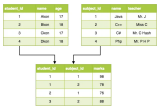


2. Network Model: This is an extension of the Hierarchical model. In this model, data is organized more like a graph, and are allowed to have more than one parent node.

3. Entity-relationship Model: In this database model, relationships are created by dividing objects of interest into entities and their characteristics into attributes. Different entities are related using relationships.



4. Relational Modal: In this model, data is organized in two-dimensional tables and the relationship is maintained by storing a common field.



student_id	name	age
1	Alan	17
2	Ben	18
3	Dan	17
4	Don	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. PHP

student_id	subject_id	marks
1	1	89
1	2	79
2	1	79
3	2	89

2. What is concurrency control? Name various methods of controlling the concurrency control? Differentiate between Binary lock and shared/Exclusive lock.

Concurrency control is a crucial aspect of database management systems (DBMS) that ensures data integrity and consistency in multi-user environments where multiple transactions are executing concurrently. Concurrency control mechanisms prevent interference between transactions and maintain the ACID properties (Atomicity, Consistency, Isolation, Durability) of database transactions.

Concurrency Control Protocol can be broadly divided into two categories:

(i) Lock Based Protocol (ii) Time Stamp Based Protocol

1. Lock Based Protocol: Lock-based protocols involve acquiring and releasing locks on data items to regulate access and ensure data consistency. Transactions request locks before accessing data items and release them after completing their operations. Locks are of two kinds: *Binary Locks and Shared/Exclusive Locks:*

i. Binary Locks: Binary locks, also known as binary semaphores, are the simplest form of locks used in concurrency control. They have two states: locked and unlocked. Normally locked state is represented by 1 and unlocked state is represented by 0. A data item protected by a binary lock can be accessed by only one transaction at a time. When a transaction acquires a binary lock on a data item, it prevents other transactions from accessing that data item until it releases the lock. Binary locks are typically used for basic synchronization tasks and provide exclusive access to data items.

ii. Shared/Exclusive Locks: Shared/Exclusive locks are more sophisticated than binary locks and offer a higher level of granularity in controlling data access. This type of lock is also called multiple mode lock. Shared locks, also known as read locks, allow multiple transactions to read a data item simultaneously without conflicting with each other. Exclusive locks, also known as write locks, allow only one transaction to write to a data item at a time, preventing any other transactions (both read and write) from accessing it. Shared locks are compatible with other shared locks, meaning multiple transactions can hold shared locks on the same data item concurrently.

2. Time Stamp Based Protocol: The most commonly used concurrency protocol is time stamp-based protocol. Transactions are assigned timestamps based on their start times, and conflicts between transactions are resolved based on their timestamps. Older transactions are given priority over newer transactions. Timestamps are used to determine the execution order of transactions and to detect conflicts between them. Timestamp-based protocols ensure serializability by resolving conflicts based on transaction timestamps. Examples of timestamp-based protocols include Timestamp Ordering and Thomas Write Rule.

Binary Lock	Shared / Exclusive Lock
Allows only one transaction to hold the lock at a time	Allows multiple transactions to hold shared locks simultaneously, but only one transaction can hold an exclusive lock at a time.
Two states: locked or unlocked.	Two states: shared or exclusive.
Not compatible with any other lock type.	Shared locks are compatible with other shared locks, but not with exclusive locks. Exclusive locks are not compatible with any other locks.
Less flexible in controlling access to resources.	More flexible in controlling access to resources.
Simple and efficient to manage.	May be more complex to manage.

3. What is normal form? Explain their types. Explain about loss-less join decomposition.

Normalization is the process of organizing the data in the database. Normalization is used to minimize the redundancy of a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update and Deletion Anomalies. Normal forms (NF) are a series of guidelines or rules that define the level of normalization achieved in a database schema. Normal forms are used to eliminate or reduce redundancy in database tables. The types of Normal form are

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)

1. First Normal Form:

For a table to be in the First Normal Form, it should follow the following rules:

- Each table cell should contain a single value.
- All the columns in a table should have unique names.
- Example: Splitting a table with multiple values in a single column into separate columns.

2. Second Normal Form:

For a table to be in the Second Normal Form:

- It should be in the First Normal form.
- And, it should not have partial dependency.
- Example: Splitting a table with a composite primary key into multiple tables to ensure full functional dependency.

3. Third Normal Form:

A table is said to be in the Third Normal Form when,

- It is in the Second Normal form.
- And, it doesn't have transitive functional dependency.
- Example: Removing attributes that depend on other non-key attributes rather than solely on the primary key

4. Boyce and Codd Normal Form:

BCNF is a higher version of the 3NF which deals with a certain type of anomaly that is not handled by 3NF. A 3NF table that does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, the following conditions must be satisfied:

- R must be in the 3rd Normal Form
- And, for each functional dependency (X → Y), X should be a super Key.
- Example: Decomposing a table to remove non-trivial functional dependencies.

5. Fourth Normal Form: A table is said to be in the Fourth Normal Form when,

- It is in the Boyce-Codd Normal Form.
- And, it doesn't have Multi-Valued Dependency.
- Example: Decomposing a table with multi-valued dependencies to eliminate redundancy.

Loss-less join decomposition is a technique used in database normalization to decompose a relation (table) into multiple smaller relations while ensuring that the original relation can be reconstructed through a join operation without losing any information. In other words, loss-less join decomposition ensures that no information is lost during the decomposition process.

By decomposing a relation into smaller relations using loss-less join decomposition, redundancy can be minimized, and data integrity can be improved. This technique is commonly used in higher normal forms, such as Boyce-Codd Normal Form (BCNF) and Fourth Normal Form (4NF), to ensure that the resulting relations are well-structured and free from anomalies.

4. What is data abstraction? What are three levels of data abstraction? Explain.

Data abstraction is the process of hiding complex implementation details while exposing only the necessary and relevant information to users. It allows users to interact with data and systems at higher, more understandable levels without needing to understand the underlying complexities.

The three levels of data abstraction are:

Physical Level:

- The lowest level of data abstraction that describes how data is stored in the database system.
- It deals with the physical storage details such as data structures, storage mechanisms, and access methods.
- At this level, users are concerned with the physical organization of data on storage devices, such as disks and tapes.
- DBMS developer is the person who deals with this level. Database administrator may be certain aware of certain details of the physical organization of data.
- For example, users may be aware of concepts like blocks, sectors, and file systems, which represent the physical organization of data on the disk.

Logical Level:

- The middle level of data abstraction that describes the logical structure of the database and the relationships among the data.
- It deals with the logical organization of data and the operations that can be performed on data.
- At this level, users interact with the database using a conceptual schema, which defines the database structure and constraints.
- Programmers and database administrators work on this level of abstraction.
- For example, users may define tables such as "Employee" with attributes like "EmployeeID," "FirstName," and "LastName," which represent the logical organization of data in the database

View Level:

- The highest level of data abstraction that presents a subset of the database to users based on their specific requirements and hide some
- It deals with how data is presented to different users or applications based on their access privileges and preferences.
- At this level, users interact with the database through views, which are virtual tables derived from the underlying base tables.
- Example user interface: a sales dashboard may provide sales representatives with a summarized view of customer information and sales data, tailored to their specific needs and preferences.

5. What is difference between Entities and Entity sets? Explain with example.

An entity is a real-world thing that can be distinctly identified as a person, place, or concept. Entity is an object which is distinguishable from others. If we cannot distinguish it from others, then it is an object but not an entity. An entity can be of two types:

Tangible Entity: Tangible entities are those entities which exist in real world physically. Example person car.

Intangible Entity: Intangible entities are those which exist only logically and have no physical existence. Example Ban account etc.

Example: If we have a table of a Student (Roll_no, Student_name, Age, Mobile_no) then each student in that table is an entity and can be uniquely identified by their Roll Number i.e Roll_no.

Student

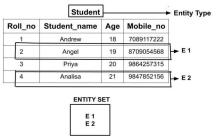
Roll_no	Student_name	Age	Mobile_no
1	Andrew	18	7089117222
2	Angel	19	8709054568
3	Priya	20	9864257315
4	Analisa	21	9847852156

→ Entity

Entity Set:

Entity Set is a collection of entities of the same entity type. In the above example of the STUDENT entity type, a collection of entities from the student entity type would form an entity set. We can say that entity type is a superset of the entity set as all the entities are included in the entity type.

Example: In the below example, two entities E1 (2, Angel, 19, 8709054568) and E2(4, Analisa, 21, 9847852156) form an entity set.



The difference between Entities and Entity sets are

Entity	Entity Set
A thing in the real world with independent existence	Set of all entities of a particular entity type.
Any particular row (a record) in a relation(table) is known as an entity.	All rows of a relation (table) in RDBMS are entity set
Has a distinct identity and can be uniquely identified.	Contains multiple instances sharing the same attributes.

6. Create two table Courses (CID, Course, Dept) and HoD (Dept, Head) using SQL language with all constraints [Primary key, Foreign key and Referential Integrity]. Assume the types of attributes by your own.

```sql
CREATE TABLE Courses (
    CID INT(11),
    Course VARCHAR(50),
    Dept VARCHAR(70),
    PRIMARY KEY (CID)
);

CREATE TABLE HoD (
    Dept VARCHAR(70),
    Head TEXT,
    FOREIGN KEY (Dept) REFERENCES Courses(Dept)
);
```

7. Differentiate between Integrity and Security with example.

The difference between security and integrity are:

Data Security	Data Integrity
Data security define the prevention of data corruption through the use of controlled access mechanisms.	Data integrity defines the quality of data, which assures the data is complete and has a whole structure.
Data security deals with the protection of data.	Data integrity deals with the validity of data.
Data security makes sure that only the people who should have access to the data are the only ones who can access the data.	Data integrity makes sure that the data is correct and not corrupt.
It refers to ensuring that data is accessed by its intended users, thus ensuring data privacy and protection.	It refers to the structure of the data and how it matches the database's schema.
Some of the popular means of data security are authentication/authorization, masking, and encryptions.	Some of the means to preserve integrity are backing up, error detection, designing a suitable user interface, and correcting data.
Protects against accidental or intentional loss or misuse of data.	Ensures data correctness, completeness, and consistency.
It avoids unauthorized access of data.	It avoids human error when data is entered.
Implemented through user accounts, authentication schemes.	Implemented using primary keys, foreign keys, and database relationships.

8. Define schedule and serializability. How can you test the serializability?

A series of operations from one transaction to another transaction is known as a schedule. It is used to preserve the order of the operation in each individual transaction. There are three types of schedules in databases:

i. Serial Schedule
ii. Non-Serial Schedule
iii. Serializable Schedule

Serializability is a property of schedules in database systems that ensures transactions produce the same outcome as if they were executed in some serial order, without overlapping in their execution. A schedule is considered serializable if it preserves the consistency and integrity of the database, regardless of the order in which transactions are executed. Serializability prevents concurrency-related anomalies such as lost updates, uncommitted data, and inconsistent reads.

Testing Serializability:

There are two main methods for testing serializability:

- Conflict Serializability
- View Serializability.

a. Conflict Serializability:

Conflict serializability is a concept in database transaction management that ensures transactions produce the same result as if they were executed in some serial order, even though they may be interleaved in a concurrent schedule. Conflict serializability provides a mechanism to analyze and ensure the consistency and integrity of the database despite concurrent execution of transactions. It determines whether a schedule is conflict-serializable by examining the conflicting operations (read-write and write-write conflicts) between transactions.

- A schedule is conflict-serializable if and only if its precedence graph is acyclic.
- Precedence graph: A directed graph where each transaction is represented by a node, and edges represent conflicts between transactions.
- If the precedence graph contains no cycles, the schedule is conflict serializable.

b. View Serializability:

View serializability is a concept in database transaction management that ensures transactions produce the same result as if they were executed in some serial order, even though they may be interleaved in a concurrent schedule. Unlike conflict serializability, which focuses on conflicts between individual read and write operations, view serializability considers the overall effect of transactions on the database.

- It determines whether a schedule is view-serializable by comparing the final state of the database (after executing the schedule) with the final states of all possible serial schedules.
- If the final state of the database in the given schedule is identical to the final state of the database in at least one serial schedule, the schedule is view-serializable.
- Testing for serializability typically involves constructing the precedence graph (for conflict serializability) or comparing final states (for view serializability) to determine if the schedule satisfies the serializability property.

9. What is Granularity of data items? How does it effect in concurrency control?

Granularity of data items refers to the level of detail or size of the data that is accessed, manipulated, or locked by transactions in a database system. It determines the extent to which data is divided into smaller components for processing.

In the context of concurrency control, granularity of data items plays a significant role in determining the effectiveness and efficiency of concurrency management mechanisms. Here's how it affects concurrency control:

Fine Granularity:

- Fine granularity refers to dividing data into smaller, more specific units or items.
- Fine-grained locking or concurrency control mechanisms operate at a lower level of granularity, such as individual data items or records.
- Fine-grained locking allows for greater concurrency by allowing multiple transactions to access and modify different data items simultaneously.
- However, fine-grained locking can also lead to increased overhead due to the need to manage a larger number of locks and increased contention for resources.

Coarse Granularity:

- Coarse granularity refers to grouping data into larger, more generalized units or items.
- Coarse-grained locking or concurrency control mechanisms operate at a higher level of granularity, such as entire tables or database objects.
- Coarse-grained locking reduces concurrency by restricting access to larger portions of data, potentially leading to more contention and reduced parallelism among transactions.
- However, coarse-grained locking can also reduce overhead and complexity by requiring fewer locks to be managed.

10. Explain 2 phase locking technique in brief.

A lock is a synchronization mechanism used in computer science and database systems to control access to shared resources such as data or code segments. It ensures that only one process or thread can access the resource at any given time, preventing concurrent access that could lead to data corruption or inconsistency.

The Two-Phase Locking (2PL) technique is a concurrency control method used in database systems to ensure serializability of transactions by preventing certain types of conflicts, such as lost updates and inconsistent reads. It consists of two phases: the growing phase and the shrinking phase.

Growing Phase: During the growing phase, a transaction can acquire locks on data items but cannot release any locks. As a transaction progresses, it acquires locks on data items it needs to read or write. Once a lock is acquired on a data item, it cannot be released until the transaction reaches the shrinking phase. Transactions continue to acquire locks until they have obtained all the necessary locks to complete their execution.

Shrinking Phase: It is the reverse of growing phase. In the shrinking phase, a transaction can release locks but cannot acquire any new locks. Once a transaction has acquired all the necessary locks and completed its operations, it enters the shrinking phase. During this phase, the transaction releases all the locks it holds, making them available for other transactions to acquire. Once all locks have been released, the transaction completes and commits, making its changes permanent.

11. What are the different approaches of Database recover? What should log file maintain in log-based recovery?

Database recovery is the process of restoring a database to a consistent state after a failure or crash. In other words, it is the process of resotring the databaseto the most recent consistent statethat existed shorty before the time of a failure. There are several approaches to database recovery, including:

Shadow Paging:

- Shadow paging is a recovery technique that maintains multiple versions of the database pages.
- When a transaction modifies a page, it creates a shadow copy of the page containing the changes.
- In the event of a failure, the system can discard the changes by reverting to the previous shadow copy of the page.
- This approach minimizes the overhead of recovery by avoiding the need for undo and redo operations.

Checkpoint Recovery:

- Checkpoint recovery involves periodically taking checkpoints, which are consistent snapshots of the database state.
- In the event of a failure, the system can restore the database to the most recent checkpoint and then apply the necessary redo operations to bring it up to date.
- This approach reduces the amount of work required for recovery by starting from a known consistent state rather than replaying all transactions from the beginning.

Rollback and Rollforward Recovery:

- Rollback recovery involves undoing the changes made by incomplete transactions at the time of failure, bringing the database back to a consistent state.
- Rollforward recovery involves reapplying the changes made by completed transactions from a consistent checkpoint forward to the point of failure, ensuring that all committed transactions are reflected in the database.

The log file in log-based recovery should maintain the following information:

Transaction ID (TID): Identifies the transaction associated with the operation.

Operation Type: Specifies the type of operation (e.g., insert, update, delete).

Checkpoint Records: Checkpoint records indicate when a checkpoint was taken and which transactions had committed at that time.

Before and After Images: Records both the before and after images of data items modified by transactions, enabling undo and redo operations during recovery.

Affected Data: Indicates the data item(s) affected by the operation.

Old Value: Stores the old value of the data item before the operation (for undo operations).

New Value: Stores the new value of the data item after the operation (for redo operations).

Transaction Commit or Abort: Indicates whether the transaction commits or aborts.

12. Explain the use of primary and foreign key in DBMS with example. What is the role of foreign key?

In a relational database management system (DBMS), primary and foreign keys are essential components used to establish relationships between tables and maintain data integrity.

Primary Key:

- A primary key is a column or a set of columns in a table that uniquely identifies each row or record in the table.
- It must have unique values and cannot contain NULL values.
- Primary keys are used to enforce entity integrity, ensuring that each record in a table is uniquely identifiable.
- Typically, primary keys are implemented using an index, which allows for fast data retrieval and efficient querying.
- Example: Consider a table named "Students" with columns such as StudentID, Name, and Age. If StudentID is designated as the primary key, it ensures that each student in the table has a unique identifier.

Foreign Key:

- A foreign key is a column or a set of columns in a table that establishes a link between data in two related tables.
- It creates a parent-child relationship between tables, where the foreign key column in one table refers to the primary key column in another table.
- Foreign keys ensure referential integrity, maintaining the consistency and accuracy of data across related tables.
- When a foreign key constraint is applied, it ensures that the values in the foreign key column must match values in the corresponding primary key column in the referenced table.
- Example: Continuing with the "Students" example, consider another table named "Grades" with columns such as StudentID (foreign key), CourseID, and Grade. Here, the StudentID column in the "Grades" table is a foreign key that references the StudentID primary key in the "Students" table, establishing a relationship between student grades and student information.

The role of a foreign key is to maintain data consistency and enforce referential integrity between related tables in a relational database. It ensures that relationships between data in different tables remain valid, preventing orphaned records and maintaining the integrity of the database structure. When properly utilized, foreign keys help in structuring and organizing data in a way that reflects real-world relationships and dependencies.

- Foreign keys help you to migrate entities using a primary key from the parent table.
- A foreign key enables you to link two or more tables together.
- It makes your database data consistent.
- A foreign key can be used to match a column or combination of columns with primary key in a parent table.
- SQL foreign key constraint is used to make sure the referential integrity of the data parent to match values in the child table.