Assignment – 3

[1. Define ACID properties of Transaction. Explain its types in details. 3](#_Toc212547509)

[2. What is sql join ? explain its different types of join Operations with examples. 3](#_Toc212547510)

[3. Compare the DDL and DML with Example. 3](#_Toc212547511)

[4. Define the time stamp base protocol . explain the condition for R/W  operations 3](#_Toc212547512)

[5. A company stores information about EMP(ssn\_no, phone , salary , address,details,remarks ), department (did, name, budget, dno) ,make your own assumptions about the relationships between the entities. Write Inner join , Outer join ,cross join relations for those entities. 3](#_Toc212547513)

[6. Define Referential Integrity Constraints. What is the main purpose of implementing referential Integrity in dbms? 3](#_Toc212547514)

[7. From the given relations, write SQL statements for given queries: 3](#_Toc212547515)

[COMPANY\_details (companyID, companyName, address, phone, estd Year) 3](#_Toc212547516)

[JOBS(ID, jobTitle, jobDescription, company ID, postedDate, lastDateOfApplication) 3](#_Toc212547517)

[APPLICANT (applicantID, fullName, email, phone) 4](#_Toc212547518)

[APPLICATION\_SUBMISSION (jobID, applicantID, submittedDate) 4](#_Toc212547519)

[Find the list of all the companies that were established in the Year 2023. 4](#_Toc212547520)

[Find the name and phone number of all the applicants. 4](#_Toc212547521)

[Display all the applicant whoso name start by S and P letter. 4](#_Toc212547522)

[8. What is a database management system? Discuss the Advantage of DBMS over File System. 4](#_Toc212547523)

[i. Data Redundancy Control: 4](#_Toc212547524)

[ii. Data Consistency: 4](#_Toc212547525)

[iii. Data Sharing: 4](#_Toc212547526)

[iv. Data Integrity: 4](#_Toc212547527)

[v. Security Features: 4](#_Toc212547528)

[vi. Backup and Recovery: 4](#_Toc212547529)

[vii. Data Independence: 4](#_Toc212547530)

[viii. Efficient Query Processing: 4](#_Toc212547531)

[ix. Centralized Management: 4](#_Toc212547532)

[x. Concurrency Control: 4](#_Toc212547533)

[9. What is functional dependency? Explain Different Inference rules. 5](#_Toc212547534)

[10. Explain two-phase locking protocol in detail. 5](#_Toc212547535)

[11. What do you mean by Functional Dependency , write with example. 5](#_Toc212547536)

[12. Explain 2NF, 3NF with suitable examples. 5](#_Toc212547537)

[13. What is a serializable schedule? Characterize schedule based on recoverability. 5](#_Toc212547538)

[14. Draw a transaction state diagram. Discuss each state that a transaction goes through. 5](#_Toc212547539)

[15. What do you mean by view? What is the use of view in SQL. Write the syntax with relevant examples of view. 5](#_Toc212547540)

[16. Write sql and relational algebra statement for the following scheme 5](#_Toc212547541)

[Food(code, packName, price, description) 5](#_Toc212547542)

[Orders(cNo, cName, qty, date, Time) 5](#_Toc212547543)

[Customers(cNo, cName, Contact, address) 5](#_Toc212547544)

[Find name and contact no of all customers from 'Balaju' 5](#_Toc212547545)

[Calculate total price of food ordered by customer "Ishmi" till now. 5](#_Toc212547546)

[17. Construct the ER diagram of the bank using all possible components like different types of entities, attributes, relationship etc  and convert it into a database schema. 5](#_Toc212547547)

[18. Describe different types of join operations used in SQL? 5](#_Toc212547548)

[19. What is data abstraction? Explain with its level. 5](#_Toc212547549)

[20. What is Normalization? Explain any two normal forms in detail. 6](#_Toc212547550)

[21. Define roles of DBA on Relational Database 6](#_Toc212547551)

[22. Explain in detail about ACID properties. 6](#_Toc212547552)

[23. Describe different components of DBMS. 6](#_Toc212547553)

[24. What is the difference between physical and logical data independence? 6](#_Toc212547554)

[25. Define the time stamp base protocol. explain the condition for R/W operations. 7](#_Toc212547555)

[26. Explain different types of locking protocol. 7](#_Toc212547556)

[27. Distinguish between generalization and aggregation. 7](#_Toc212547557)

[28. Distinguish between tuple and domain. 7](#_Toc212547558)

[29. Distinguish between Where and Having clause. 7](#_Toc212547559)

[30. Distinguish between composite attribute and derived attribute. 7](#_Toc212547560)

[31. What do you mean by log based recovery? 7](#_Toc212547561)

[32. Write sql and relational algebra statement for the following scheme 7](#_Toc212547562)

[Food(code, packName, price, description) 7](#_Toc212547563)

[Orders(cNo, cName, qty, date, Time) 7](#_Toc212547564)

[Customers(cNo, cName, Contact, address) 7](#_Toc212547565)

[Find name and contact no of all customers from ‘Tokha’ 8](#_Toc212547566)

[Calculate total price of food ordered by customer "Grishma" till now. 8](#_Toc212547567)

[33. Construct an ER diagram for a university using all possible components like different types of entities, attributes, relationship etc and convert it into a database schema. 8](#_Toc212547568)

[34. List the roles of Database administrator. Explain different types of Database user. 8](#_Toc212547569)

[I. End Users: 8](#_Toc212547570)

[II. Application Programmers: 8](#_Toc212547571)

[III. Database Administrators (DBAs): 8](#_Toc212547572)

[35. List the steps to draw E-R diagrams and Design E-R diagram for Education Portal 9](#_Toc212547573)

[36. Explain about Concurrency Control and also explain about Time stamp based protocol. 9](#_Toc212547574)

[37. Explain Serial schedule with suitable Example. 9](#_Toc212547575)

[38. Define Integrity Constraints. What is the main purpose of implementing Integrity in dbms? Explain its types. 9](#_Toc212547576)

1. Define ACID properties of Transaction. Explain its types in details.

A transaction is a collection of several operations on the database that appears to be a single unit from the point of view of the database user. For example, a transfer of funds from one account to another account is a single operation from customer’s point of view but within the database system, however, it consists of several tasks.

The ACID properties are required to maintain integrity of database. They are:

1. Atomicity: It ensures that the transaction is either completely executed or not at all. If a transaction is unsuccessful due to failure or constraints, any changes applied to the database during the transaction must be undone through ROLLBACK/ABORT command. It prevents the database from staying in partial state.
2. Consistency: It ensures that the transaction, after complete execution without interference from other concurrent transactions, takes the database from one consistent state to another. The state during execution is inconsistent state which must be hidden to user.
3. Isolation: The transaction is not interfered by the concurrent execution of other transactions. The state of database after concurrent execution of transaction must be equivalent to the state reached after serial execution of transaction in same order,
4. Durability: The changes applied to a database after execution of transaction must be persistent. Once a transaction is committed, all updates to the database must remain permanently even during failures like system crash and power loss.
5. What is sql join ? explain its different types of join Operations with examples.

|  |  |
| --- | --- |
| Player | Jersey |
| Son | 7 |

SQL join is a binary relation operation used to combine tuples of multiple relation on the basis of matching values in common attributes. It can be understood as a cartesian product of relation followed by selection criteria.

|  |  |
| --- | --- |
| Club | Player |
| TOT | Bale |
| MUN | Son |

The types of join operations are:

* 1. Inner Join: By using inner join, the resulting join relation only includes the tuples that satisfy the matching criteria, excluding others.

1. Theta Join: It is a type of join operation in which the join consdition can be any comparison operator (<,>,=,<=,>=,!=).

Club |><|(c.Player!=p.Player) Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| TOT | Bale | NULL |

1. Equi Join: It is a type of join operation in which the join consdition can be only equality comparison operator (=).

Club |><|(c.Player=p.Player) Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| MUN | Son | 7 |

1. Natural Join: It is a type of join operation in which tuples are joined by matching same values in common attribute and eliminating redundant attributes.

Club |><| Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| MUN | Son | 7 |

1. Outer Join: By using outer join, the resulting join relation includes the tuples that satisfy the matching criteria as well other tupels with missing criteira.
2. Left Outer Join: It is a type of join operation in which all tuples of left relation are kept. The tuples in left relation without matching values in right relation, have their attributes filled by NULL value.

Club =|><| Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| TOT | Bale | NULL |
| MUN | Son | 7 |

1. Right Outer Join: It is a type of join operation in which tuples of right relation are kept. The tuples in right relation without matching values in left relation, have their attributes filled by NULL value.
2. Club |><| =Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| MUN | Son | 7 |

1. Full Outer Join: It is a type of join operation in which all tuples of both relation are kept. The tuples in any relation without matching values in other relation, have their attributes filled by NULL value.

Club =|><|= Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| TOT | Bale | NULL |
| MUN | Son | 7 |

1. Compare the DDL and DML with Example.

|  |  |  |
| --- | --- | --- |
| Feature | DDL (Data Definition Language) | DML (Data Manipulation Language) |
| Purpose/Role | Used to define data structures and database schemas | Used to perform operations at the conceptual/external level. |
| What it Modifies | Modifies the database schema or data structure. | Modifies the data stored in the database, but not the schema. |
| Control | Defines storage structure and access methods. | Can be procedural (low-level, specifying *how* to get data) or non-procedural (high-level, specifying *what* data is needed). |
| Auto Commit | Yes — changes are permanent immediately. | No — requires explicit COMMIT or ROLLBACK. |
| User | Used by DBA | Used by end user and application programmers |
| Example Statements | CREATE, ALTER, DROP. | Query, Delete, Update, Insert. |

1. Define the time stamp base protocol . explain the condition for R/W  operations.

Time stamp base protocol is a concurrency control technique in which unique transaction identiferes called timestqamps (TS) are associated with erach transaction t ensure serializability. A TS can be generated either by assigning the current value of clock to the transaction or by attaching the value of a logical coiunter. The ording of TS is determined by age of tyransaction. Example: a transaction created at 002 clock time is older than all other transactions which come after and prioiroty may be given to older one.

The time stamp must have following two properties.

1. **Uniqueness:** Ensures that no equal time stamp values can exists
2. **Monotonicity:** Ensures that time stamp values always increase

Data items X in the database are managed using two special timestamps:

* 1. **Write-TS(X):** The latest time the data item has been successfully written into.
  2. **Read-TS(X):** The latest time the data item has been successfully read from.

Whenever a transaction T tries to issue a Read(x) or write(x) operation, the basic timestamp ordering algorithm compares the timestamp of T with ReadTS(x) and WriteTS(x) to ensure that the timestamp order of transaction execution is not violated.

**Case 1:** Transaction T issues Read(x) Operation

If WriteTS(x)>TS(T) then abort and rollback T and reject the operation. If WriteTS(x)<=TS(T) then execute the Read(x) operation of T and set ReadTS(x) to the larger to TS(T) and the current ReadTS(x).

**Case 2:** Transaction T issues Write(x) Operation

If ReadTS(x)>TS(T) or if WriteTS(x)>TS(T) then abort and rollback T and reject the operation, else then execute the Write(x) operation and set WriteTS(x) to TS(T).

Example:

Assume timestamps of T1 and t2 is 100 and 110 respectively and initial value of x is 500.

A table with text and numbers

AI-generated content may be incorrect.

In this example if the transaction T1 is not aborted it will suffer from lost update problem and will make final value of x = 500 rather than 700.

1. A company stores information about EMP(ssn\_no, phone , salary , address,details,remarks ), department (did, name, budget, dno) ,make your own assumptions about the relationships between the entities. Write Inner join , Outer join ,cross join relations for those entities.

We assume that the employee (EMP) table contains a foreign key, dno, which references the primary key did in the DEPARTMENT table.

* 1. Inner Join: SELECT \* FROM EMP AS E INNER JOIN DEPARTMENT AS D ON E.dno=D.did;
  2. Outer Join: SELECT \* FROM EMP AS E LEFT OUTER JOIN DEPARTMENT AS D ON E.dno=D.did;
  3. Inner Join: SELECT \* FROM EMP , DEPARTMENT;

1. Define Referential Integrity Constraints. What is the main purpose of implementing referential Integrity in dbms?

Referential Integrity constraints are rules defined between 2 relations in a database. It is implemented by using a foreign key in one relation(R1) that points to te primary key of another relation(R2).

The main purpose of implementing referential integrity in a DBMS is to maintain consistency among the tuples in the two related tables. This mechanism ensures that if a record exists in one table (the dependent table, R1), the record it references in the other table (the parent table, R2) must also exist. Referential integrity prevents the creation of orphan data—situations where a foreign key points to a primary key value that does not exist or has been deleted.

1. From the given relations, write SQL statements for given queries:

COMPANY\_DETAILS (companyID, companyName, address, phone, estd Year)

JOBS(ID, jobTitle, jobDescription, company ID, postedDate, lastDateOfApplication)

APPLICANT (applicantID, fullName, email, phone)

APPLICATION\_SUBMISSION (jobID, applicantID, submittedDate)

Find the list of all the companies that were established in the Year 2023.

Find the name and phone number of all the applicants.

Display all the applicant whoso name start by S and P letter.

SELECT companyName FROM COMPANY\_DETAILS WHERE estd\_Year=2023;

SELECT fullName,phone FROM APPLICANT;

SELECT \* FROM APPLICANT WHERE fullName LIKE ‘S%’ OR fullName LIKE ‘P%’;

1. What is a database management system? Discuss the Advantage of DBMS over File System.

A Database Management System (DBMS) is a software system designed to store, retrieve, manage, and manipulate data efficiently. It provides a structured way to organize data in databases, ensuring data integrity, security, and easy access.. A Traditional File System is a method of storing data in flat files (e.g., .txt, .csv, .dat) without a structured database. Each file contains records, but there is no relationship between files, leading to data redundancy and inconsistency.

The advantages of using DBMS over traditional filing system are given below:

1. Data Redundancy Control: DBMS minimizes duplication of data through normalization, while file systems often have redundant data across multiple files stored in different locations.
2. Data Consistency: Ensures all data conforms to rules and constraints, unlike file systems where consistency must be manually maintained.
3. Data Sharing: Allows concurrent access by multiple users/applications, while file systems typically allow only one user at a time.
4. Data Integrity: Provides mechanisms to maintain accuracy and validity of data through constraints and validation rules.
5. Security Features: Offers user authentication, authorization, and access controls at various levels.
6. Backup and Recovery: Built-in mechanisms for data backup and recovery from failures.
7. Data Independence: Separates physical storage from logical representation, allowing changes without affecting applications.
8. Efficient Query Processing: Provides powerful query languages (like SQL) for complex data retrieval.
9. Centralized Management: Single repository for data with centralized administration.
10. Concurrency Control: Manages simultaneous access by multiple users to prevent inconsistencies.
11. What is functional dependency? Explain Different Inference rules.

A functional dependency is a constraint defined between two sets of attributes (X and Y) belonging to a relation.

Constraint Definition: If X->Y holds, it means that for any two tuples t1 and t2 in the relation, if their values for X are equal (t1[X]=t2[X]), then their values for Y must also be equal (t1[Y]=t2[Y]). Essentially, the values of Y are functionally dependent on (or determined by) the values of X.

X is referred to as the left-hand side (LHS).

Y is referred to as the right-hand side (RHS).

Example: In a relation containing employee details, if {Employee number} is the primary key, we have the functional dependency:

Employee number -> {Employee name,salary,city}

This means that knowing the employee number is sufficient to determine all other listed attributes.

1. Explain two-phase locking protocol in detail.

The Two-Phase Locking (2PL) protocol is a robust concurrency control mechanism designed to guarantee serializability. It is a process used to gain ownership of shared resources without creating the possibility of deadlock, particularly in distributed systems. A transaction adhering to 2PL is said to follow the protocol if all locking operations precede the first unlock operation.

The protocol divides the transaction's lifetime into two distinct phases:

I. Growing Phase:

1. Acquisition Only: In this phase, a transaction can only acquire locks (Shared (S) or Exclusive (X)) but cannot release any lock

2. Lock Point: The transaction continues to acquire locks until it has obtained all the locks it needs for its execution. The point when the final lock is acquired is called the lock point.

3. Upgrading: Lock conversion, such as upgrading a Shared lock to an Exclusive lock, may be performed during the growing phase.

II. Shrinking Phase

1. Release Only: In this phase, a transaction can only release locks but cannot acquire any new locks.

2. Downgrading: Lock conversion, such as downgrading an Exclusive lock to a Shared lock, must be done during the shrinking phase.

Categories of Two-Phase Locking (2PL Variants)

* 1. Strict 2PL: This variant requires that all Exclusive (X) locks held by the transaction are released only after the transaction commits. Strict 2PL ensures the schedule is recoverable and cascadeless, though it remains possible for deadlocks to occur.
  2. Rigorous 2PL: This is the most restrictive variant. It requires that both Exclusive (X) and Shared (S) locks are held until after the transaction commits.
  3. Conservative 2PL: This variant aims to prevent deadlocks completely. It requires a transaction to pre-declare and lock all items it needs (its read-set and write-set) before execution begins. If it cannot acquire all necessary locks, it waits without locking any items until all are available.

1. What do you mean by Functional Dependency , write with example.

A functional dependency is a constraint defined between two sets of attributes (X and Y) belonging to a relation.

Constraint Definition: If X->Y holds, it means that for any two tuples t1 and t2 in the relation, if their values for X are equal (t1[X]=t2[X]), then their values for Y must also be equal (t1[Y]=t2[Y]). Essentially, the values of Y are functionally dependent on (or determined by) the values of X.

X is referred to as the left-hand side (LHS).

Y is referred to as the right-hand side (RHS).

Example: In a relation containing employee details, if {Employee number} is the primary key, we have the functional dependency:

Employee number -> {Employee name,salary,city}

This means that knowing the employee number is sufficient to determine all other listed attributes.

1. Explain 2NF, 3NF with suitable examples.

Normalization is the process of decomposing bad relation by breaking down their attributes into smaller relations. It is used to minimize, insert, update and delete anomalies and help maintain data consistency in database.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Jersey | Player | Club | Competition | | |
|  |  |  | Domestic | European | |
| 9 | Haaland | MCI | PL | | UCL |
| 7 | Son | TOT,LAFC | PL,MLS | | - |

1. 1NF: A relation is said to be in 1 NF if it does not have multi-valued attributes, composite attributes and their combinations. It states that the domain of an attribute must include only atomic (simple invisible) values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jersey | Player | Club | Competition | Type |
| 9 | Haaland | MCI | PL | Domestic |
| 9 | Haaland | MCI | UCL | European |
| 7 | Son | TOT | PL | Domestic |
| 7 | Son | LAFC | MLS | Domestic |

1. 2NF: A relation is said to be in 2 NF if and only if It is already in 1 NF and Every non-prime attribute is fully dependent on the primary key of the relation(no partial dependency.

|  |  |
| --- | --- |
| Jersey | Player |
| 9 | Haaland |
| 7 | Son |

|  |  |  |  |
| --- | --- | --- | --- |
| Jersey | Club | Competition | Type |
| 9 | MCI | PL | Domestic |
| 9 | MCI | UCL | European |
| 7 | TOT | PL | Domestic |
| 7 | LAFC | MLS | Domestic |

1. 3NF: A relation is said to be in third normal form if and only if: It is already in 2 NF and Every non-prime attribute is non-transitively dependent on the primary key (no transisitve dependency).

|  |  |
| --- | --- |
| Jersey | Player |
| 9 | Haaland |
| 7 | Son |

|  |  |
| --- | --- |
| Competition | Type |
| UCL | European |
| PL | Domestic |
| MLS | Domestic |

|  |  |  |
| --- | --- | --- |
| Jersey | Club | Competition |
| 9 | MCI | PL |
| 9 | MCI | UCL |
| 7 | TOT | PL |
| 7 | LAFC | MLS |

1. What is a serializable schedule? Characterize schedule based on recoverability.

When transactions are executing concurrently in an interleaved fashion, then the order of execution of operations from the various transactions is known as a schedules (history).

A schedule is considered serializable if executing the transactions concurrently produces the totally same final result as running the transactions sequentially in some serial order. The goal of concurrency control is to ensure serializability.

Types of Serializable Schedules:

1. Conflict Serializable Schedule: A schedule that can be transformed into a serial schedule by swapping only non-conflicting operations.

2. View Serializable Schedule: A schedule whose view (or final result) equals a serial schedule.

Characterizing Schedules Based on Recoverability:

Characterizing schedules based on recoverability determines how easily the system can recover from transaction and system failures:

1. Recoverable Schedule: A schedule is recoverable if, whenever a transaction Ti commits, all transactions Tj that wrote an item read by Ti must have already committed. This ensures that once a transaction commits, it will not need to be rolled back.

2. Cascadeless Schedule: A schedule where every transaction reads only those data items that have been written by committed transactions.

3. Schedules Requiring Cascaded Rollback: These are schedules in which an uncommitted transaction reads an item written by a failed transaction. If the first transaction fails, the second (reading) transaction must also be rolled back, leading to cascading rollbacks.

4. Strict Schedules: The most stringent type, where a transaction cannot read nor write a data item until the last transaction that wrote has committed.

1. Draw a transaction state diagram. Discuss each state that a transaction goes through.

A transaction goes through various states during its execution from satert to fninish which is tracked by the recovery manager.

A diagram of a diagram

AI-generated content may be incorrect.

1. Active State: Initial state of transaction in which the transaction is being executed. The transaction performs its READ and WRITE operations in this state.
2. Partially Comiited State: When transaction executes its final operation, it is said to be in this state. After execution of all operations, the database system performs some checks eg, consistency state of database.
3. Comitted State: If transaction executes all its operations successfully it is said to be committed. All its effects are now permanently made on database system.
4. Failed State: If any check made by database recovery system fails, the transaction is said to be in failed state, from where it can no longer proceed further.
5. Aborted State: If any of the checks fails and transaction reached in failed state, the recovery manager rolls back all its write operations on the database to make database in the state where it was prior to the start off execution of transaction. Transactions in this state are called aborted. Database recovery module can select either of the following operations:

Restart the transaction

Kill the transaction

1. Terminated State: The final state which makrs the conclusion of transaction execution, whether whether successfully (committed) or unsuccessfully (aborted).
2. What do you mean by view? What is the use of view in SQL. Write the syntax with relevant examples of view.

A view is a single table that is derived from other base tables or previously defined views. It is often referred to as a virtual table because, while it provides a logical structure of data, it does not necessarily exist in a physical form on disk. Views belong to the External Level (View Schema), representing the highest level of data abstraction. Views are crucial because they describe how individual users or applications see the data, giving each user a customized perspective.

1. Simplification: Views can simplify complex queries by pre-defining joins or aggregations, making data access easier for users.

2. Security and Access Control: Views can be used to restrict access to sensitive data, ensuring users only interact with a subset of the database, thereby supporting access control and security.

3. Abstraction: They separate the user's perception of the data from the overall conceptual schema, contributing to data independence.

General Syntax:

CREATE VIEW view\_name AS

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

Example: Creating a view named DetailsView that only shows the Name and Address for students whose ID is 5 or less.

CREATE VIEW DetailsView AS

SELECT Name, Address

FROM Student\_Details

WHERE STD <= 5;

1. Write sql and relational algebra statement for the following scheme

Food(code, packName, price, description)

Orders(cNo, cName, qty, date, Time)

Customers(cNo, cName, Contact, address)

Find name and contact no of all customers from 'Balaju'

Calculate total price of food ordered by customer "Ishmi" till now.

SELECT cName, Contact FROM Customers WHERE address=’Balaju’;

SELECT SUM(O.qty \* F.price) FROM Orders AS O JOIN Customers AS C ON O.cNo = C.cNo JOIN Food AS F ON C.cName = F.packName WHERE C.cName = 'Ishmi';

1. Construct the ER diagram of the bank using all possible components like different types of entities, attributes, relationship etc  and convert it into a database schema.
2. Customer

CustomerID *(PK)*

Name

Address

Phone

Email

1. Account

AccountNo *(PK)*

Type

Balance

OpenDate

1. Transaction

TransactionID *(PK)*

Date

Amount

Type

1. Branch

BranchID *(PK)*

Name

Location

1. Loan

LoanID *(PK)*

Amount

IssueDate

Duration

1. Describe different types of join operations used in SQL?

SQL join is a binary relation operation used to combine tuples of multiple relation on the basis of matching values in common attributes. It can be understood as a cartesian product of relation followed by selection criteria.

|  |  |
| --- | --- |
| Club | Player |
| TOT | Bale |
| MUN | Son |

The types of join operations are:

* 1. Inner Join: By using inner join, the resulting join relation only includes the tuples that satisfy the matching criteria, excluding others.

1. Theta Join: It is a type of join operation in which the join consdition can be any comparison operator (<,>,=,<=,>=,!=).

Club |><|(c.Player!=p.Player) Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| TOT | Bale | NULL |

1. Equi Join: It is a type of join operation in which the join consdition can be only equality comparison operator (=).

Club |><|(c.Player=p.Player) Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| MUN | Son | 7 |

1. Natural Join: It is a type of join operation in which tuples are joined by matching same values in common attribute and eliminating redundant attributes.

Club |><| Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| MUN | Son | 7 |

1. Outer Join: By using outer join, the resulting join relation includes the tuples that satisfy the matching criteria as well other tupels with missing criteira.
2. Left Outer Join: It is a type of join operation in which all tuples of left relation are kept. The tuples in left relation without matching values in right relation, have their attributes filled by NULL value.

Club =|><| Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| TOT | Bale | NULL |
| MUN | Son | 7 |

1. Right Outer Join: It is a type of join operation in which tuples of right relation are kept. The tuples in right relation without matching values in left relation, have their attributes filled by NULL value.
2. Club |><| =Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| MUN | Son | 7 |

1. Full Outer Join: It is a type of join operation in which all tuples of both relation are kept. The tuples in any relation without matching values in other relation, have their attributes filled by NULL value.

Club =|><|= Player

|  |  |  |
| --- | --- | --- |
| Club | Player | Jersey |
| TOT | Bale | NULL |
| MUN | Son | 7 |

1. What is data abstraction? Explain with its level.

Data abstraction refers to the mechanism of hiding the details regarding data organization and storage. The three-schema architecture is a framework to separate the database into three levels of abstraction. Its purpose is to seaparte user applications from physical database, to allow data indepenece and to simplify database management, maintenance and security.



Figure 1: 3-schema architecture

The layers of abstraction are:

1. Internal Level (Physical Schema):

It is the lowest level of abstraction. It describes how data is physically stored in the database. Includes file structures, indexes, compression, storage devices, etc. Example: Student data is stored as binary records in files on a hard disk, sorted by roll.

2. Conceptual Level (Logical Schema):

It is the midlle level of abstraction. It describes the structure of the entire database for a community of users. It is independent of physical storage through logical data independence .It includes entities, relationships, constraints(primary & foreign keys), data types, etc. Example: Student(Roll, Name, Major) is a logical table, with Roll as primary key and possibly foreign keys to other tables.

3. External Level (View Schema):

It is the highest level of abstraction. It describes how individual users or applications see the data. Each user can have a customized view. It allows access control, simplification, and security. Example:

A teacher’s view: Student(RollNo, Name)

A finance department’s view: Student(RollNo, FeeStatus)

1. What is Normalization? Explain any two normal forms in detail.

Normalization is the process of decomposing bad relation by breaking down their attributes into smaller relations. It is used to minimize, insert, update and delete anomalies and help maintain data consistency in database.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Jersey | Player | Club | Competition | | |
|  |  |  | Domestic | European | |
| 9 | Haaland | MCI | PL | | UCL |
| 7 | Son | TOT,LAFC | PL,MLS | | - |

1. 1NF: A relation is said to be in 1 NF if it does not have multi-valued attributes, composite attributes and their combinations. It states that the domain of an attribute must include only atomic (simple invisible) values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jersey | Player | Club | Competition | Type |
| 9 | Haaland | MCI | PL | Domestic |
| 9 | Haaland | MCI | UCL | European |
| 7 | Son | TOT | PL | Domestic |
| 7 | Son | LAFC | MLS | Domestic |

1. 2NF: A relation is said to be in 2 NF if and only if It is already in 1 NF and Every non-prime attribute is fully dependent on the primary key of the relation(no partial dependency.

|  |  |
| --- | --- |
| Jersey | Player |
| 9 | Haaland |
| 7 | Son |

|  |  |  |  |
| --- | --- | --- | --- |
| Jersey | Club | Competition | Type |
| 9 | MCI | PL | Domestic |
| 9 | MCI | UCL | European |
| 7 | TOT | PL | Domestic |
| 7 | LAFC | MLS | Domestic |

1. 3NF: A relation is said to be in third normal form if and only if: It is already in 2 NF and Every non-prime attribute is non-transitively dependent on the primary key (no transisitve dependency).

|  |  |
| --- | --- |
| Jersey | Player |
| 9 | Haaland |
| 7 | Son |

|  |  |
| --- | --- |
| Competition | Type |
| UCL | European |
| PL | Domestic |
| MLS | Domestic |

|  |  |  |
| --- | --- | --- |
| Jersey | Club | Competition |
| 9 | MCI | PL |
| 9 | MCI | UCL |
| 7 | TOT | PL |
| 7 | LAFC | MLS |

1. Define roles of DBA on Relational Database

DBAs are database users who maintains the database description in original form. It is responsible for overall control of the database system. Example: Database managers ensuring that the product database runs smoothly, securely, and efficiently—supporting developers, customers, and business operations.

The responsibilities of a DBA are:

1. Schema definition and modification:

The creation and modification of the original description of the database structure and the way that structure is reflected by the files of the physical database.

1. Storage structure and access method definition:

The DBA determines how data is physically stored on disk, including file organization, portioning and tablespace management. The DBA also chooses the best access methods for query efficiency through indexing strategies like B-tree, hashmap, bitmaps.

1. Granting authorization for data access:

Granting access to the database to different users.

1. Routine maintenance:

Making backup copies of the database and repairing damage to the database due to hardware or software failures or misuse.

1. Explain in detail about ACID properties.

A transaction is a collection of several operations on the database that appears to be a single unit from the point of view of the database user. For example, a transfer of funds from one account to another account is a single operation from customer’s point of view but within the database system, however, it consists of several tasks.

The ACID properties are required to maintain integrity of database. They are:

1. Atomicity: It ensures that the transaction is either completely executed or not at all. If a transaction is unsuccessful due to failure or constraints, any changes applied to the database during the transaction must be undone through ROLLBACK/ABORT command. It prevents the database from staying in partial state.
2. Consistency: It ensures that the transaction, after complete execution without interference from other concurrent transactions, takes the database from one consistent state to another. The state during execution is inconsistent state which must be hidden to user.
3. Isolation: The transaction is not interfered by the concurrent execution of other transactions. The state of database after concurrent execution of transaction must be equivalent to the state reached after serial execution of transaction in same order.
4. Durability: The changes applied to a database after execution of transaction must be persistent. Once a transaction is committed, all updates to the database must remain permanently even during failures like system crash and power loss.
5. Describe different components of DBMS.

A database system environment is a collective system of components that work together to comprise and regulate the data, management, and use of the database.

The main components are:

* 1. Hardware: This includes the physical computers, storage devices (disk), and computer peripherals used to manage the database.
  2. Software: This encompasses three types of software:

◦ Operating System (OS): The base software required to run the system.

◦ DBMS Software: The core system responsible for processing application programs and user queries and accessing the stored data.

◦ Application Programs: Software written by application programmers that users employ to interact with the database.

* 1. People (Users/Programmers): This includes all individuals who interact with the system:

◦ End Users: Individuals who query, update, and generate reports.

◦ Application Programmers: Those who develop the applications.

◦ Database Administrators (DBAs): Those who administer and ensure the smooth running of the system.

1. Techniques: This refers to the rules, concepts, and instructions given to both the people and the software to handle and manage the data within the environment.
2. What is the difference between physical and logical data independence?

| **Aspect** | **Logical Data Independence** | **Physical Data Independence** |
| --- | --- | --- |
| **Definition** | Ability to change the conceptual schema without affecting external schemas (user views). | Ability to change the internal schema without affecting the conceptual schema. |
| **Level Affected** | Between conceptual and external levels. | Between internal and conceptual levels. |
| **What Changes?** | Modifications to tables, relationships, or constraints (e.g., adding/removing entities or attributes). | Changes in file structures, storage devices, or access methods (e.g., indexing, hashing). |
| **Impact on Users** | Applications not using the modified schema remain unaffected. | No impact on application programs or user queries. |
| **Purpose** | Protects applications from changes in the logical database design. | Protects the database design from changes in physical storage. |
| **Complexity** | Harder to achieve (may require view definitions). | Easier to achieve (handled by the DBMS). |
| **Example** | Adding a phone\_number column to a Student table without breaking existing apps that don’t use this field. | Switching from B-tree to hash indexing for faster searches without altering table structures. |

1. Define the time stamp base protocol. explain the condition for R/W operations.

Time stamp base protocol is a concurrency control technique in which unique transaction identiferes called timestqamps (TS) are associated with erach transaction t ensure serializability. A TS can be generated either by assigning the current value of clock to the transaction or by attaching the value of a logical coiunter. The ording of TS is determined by age of tyransaction. Example: a transaction created at 002 clock time is older than all other transactions which come after and prioiroty may be given to older one.

The time stamp must have following two properties.

1. **Uniqueness:** Ensures that no equal time stamp values can exists
2. **Monotonicity:** Ensures that time stamp values always increase

Data items X in the database are managed using two special timestamps:

* 1. **Write-TS(X):** The latest time the data item has been successfully written into.
  2. **Read-TS(X):** The latest time the data item has been successfully read from.

Whenever a transaction T tries to issue a Read(x) or write(x) operation, the basic timestamp ordering algorithm compares the timestamp of T with ReadTS(x) and WriteTS(x) to ensure that the timestamp order of transaction execution is not violated.

**Case 1:** Transaction T issues Read(x) Operation

If WriteTS(x)>TS(T) then abort and rollback T and reject the operation. If WriteTS(x)<=TS(T) then execute the Read(x) operation of T and set ReadTS(x) to the larger to TS(T) and the current ReadTS(x).

**Case 2:** Transaction T issues Write(x) Operation

If ReadTS(x)>TS(T) or if WriteTS(x)>TS(T) then abort and rollback T and reject the operation, else then execute the Write(x) operation and set WriteTS(x) to TS(T).

Example:

Assume timestamps of T1 and t2 is 100 and 110 respectively and initial value of x is 500.

A table with text and numbers

AI-generated content may be incorrect.

In this example if the transaction T1 is not aborted it will suffer from lost update problem and will make final value of x = 500 rather than 700.

1. Explain different types of locking protocol.

Locking protocols are fundamental concurrency control techniques that use locks (variables associated with data items) to synchronize access

. The sources describe the primary types of locks used:

1. Binary Locks:

    ◦ A binary lock has only two possible states or values: locked (1) or unlocked (0).

    ◦ If a data item is locked (1), it cannot be accessed by any transaction attempting to perform a database operation on it. The item can only be accessed when the lock value is 0 (unlocked).

2. Shared (S) / Exclusive (X) Locks (Read/Write Locks):

    ◦ These locks allow for finer control over access by distinguishing between reading and writing operations. A data item can be in one of three states: read-locked, write-locked, or unlocked.

    ◦ Shared (S) Lock (Read-locked): Grants permission to read the data item

. Because reading does not change the data, multiple transactions can hold an S lock concurrently on the same item.

    ◦ Exclusive (X) Lock (Write-locked): Grants permission to read and write the data item

. This lock is exclusive, meaning only a single transaction can hold an X lock on the item at any given time.

1. Distinguish between generalization and aggregation.

|  |  |  |
| --- | --- | --- |
| Aspect | Generalization | Aggregation |
| Approach | Bottom-up approach. | Modeling concept where relationships are treated as higher-level entities. |
| Concept | Combining two or more lower-level entities (subclasses) to form a higher-level entity (superclass). | Treats a relationship set and the participating entity sets as a single, higher-level entity set. |
| Purpose | To maximize the differences between attributes of the members, allowing various lower-level entities to group together. | To model relationships among relationship sets. In the relational model, this results in no distinction between the relationship sets and the entity sets. |
| Relational Mapping | The superclass attributes are included in the subclass relations, or the subclasses are related to the superclass using foreign keys (if subclasses are disjoint and complete). | A separate relation is created containing the primary key of the associated entity set and the relationship set attributes. |
| Example | Player and Manager generalized into Employee. | A relationship between Student and Course is aggregated and related to another entity like College. |

1. Distinguish between tuple and domain.

|  |  |  |
| --- | --- | --- |
| Aspect | Tuple (Row or Record) | Domain (Set of Acceptable Values) |
| Definition | A tuple is a single row in a relation (table). | A domain is a set of acceptable values that a column (attribute) is allowed to contain. |
| Content | It contains the data records—a collection of values corresponding to the attributes in the relation. | It specifies the constraints on the data, including the data type, allowed range, and data characteristics. |
| Atomicity | Contains data values that are considered indivisible as far as the relation model is concerned. | The values within the domain must be atomic. |
| Example | A specific student entry: (10, 'Ramesh', 'Buthwal', 5). | The domain for Marital Status only permits values like {'Married', 'Single', 'Divorced'}. |

1. Distinguish between Where and Having clause.
2. Distinguish between composite attribute and derived attribute.

|  |  |  |
| --- | --- | --- |
| Aspect | Composite Attribute | Derived Attribute |
| Definition | An attribute that is composed of several meaningful sub-attributes. | An attribute whose value is derived or computed from the values of other related attributes. |
| ER Representation | Represented by an oval connected via lines to other sub-ovals. | Represented by a dashed oval. |
| Storage/Mapping | Only its simple components become actual columns in the database table; the composite attribute itself is not mapped as a column. | Generally not stored in the database physically, as it can be calculated when needed. |
| Example | Address is composite, consisting of sub-attributes like House\_No., City , and State\_no. | Age is derived from the Date of Birth attribute. |

1. What do you mean by log based recovery?

Log-based recovery is the standard technique used by the recovery subsystem of a DBMS to handle transaction failures or system crashes. The primary goal is to restore the database to the most recent consistent state prior to the time of failure.

The System Log

To achieve recovery, the system maintains a persistent record of all changes called the system log.

• The log is a sequential, append-only file stored on disk, making it resistant to catastrophic failures.

• The log records entries for every significant transaction operation, including:

    ◦ [start transaction, T]: When transaction begins execution.

    ◦ [write item, T, X, old\_value, new\_value]: When changes data item , recording both the value before the write and the value after the write.

    ◦ [commit, T]: When completes successfully.

Recovery Operations

During recovery, the system reviews the log to identify transactions that need manipulation:

1. UNDO Operation: Used for failed or aborted transactions that may have partially written data to the disk before crashing. The log entries provide the old\_value to restore data items to their state before the transaction began.

2. REDO Operation: Used for committed transactions whose updates might have been residing only in temporary memory buffers and were not force-written to disk before the crash. The log provides the new\_value to ensure durability.

1. Write sql and relational algebra statement for the following scheme

Food(code, packName, price, description)

Orders(cNo, cName, qty, date, Time)

Customers(cNo, cName, Contact, address)

Find name and contact no of all customers from ‘Tokha’

Calculate total price of food ordered by customer "Grishma" till now.

SELECT cName, Contact FROM Customers WHERE address = 'Tokha';

SELECT SUM(O.qty \* F.price) FROM Orders AS O JOIN Customers C ON O.cNo = C.cNo JOIN Food AS F ON C.cName = F.packName WHERE C.cName = 'Grishma';

1. Construct an ER diagram for a university using all possible components like different types of entities, attributes, relationship etc and convert it into a database schema.
2. List the roles of Database administrator. Explain different types of Database user.

A database user is an individual, application, or system that interacts with a database to perform operations such as storing, retrieving, updating, or managing data. Each user is typically assigned specific access rights and privileges based on their role to ensure security and proper data management.

Database users can be categorized based on their level of interaction with the database system:

1. End Users:

End users are database users who interact with database by issuing commands from a terminal through predefined application programs to perform functions like create, retrieve, modify and delete. Example: Bank tellers using an interface to access customer accounts

1. Application Programmers:

Application programmers are database users who develop applications that interact with the database using programming languages and APIs like Access, FoxPro, COBOL, etc. These application programs are used by end users to operate on data. Example: Developers creating e-commerce websites that connect to product databases.

1. Database Administrators (DBAs):

DBAs are database users who maintains the database description in original form. It is responsible for overall control of the database system. Example: Database managers ensuring that the product database runs smoothly, securely, and efficiently—supporting developers, customers, and business operations.

The responsibilities of a DBA are:

1. **Schema definition and modification:**

The creation and modification of the original description of the database structure and the way that structure is reflected by the files of the physical database.

1. **Storage structure and access method definition:**

The DBA determines how data is physically stored on disk, including file organization, portioning and tablespace management. The DBA also chooses the best access methods for query efficiency through indexing strategies like B-tree, hashmap, bitmaps.

1. **Granting authorization for data access:**

Granting access to the database to different users.

1. **Routine maintenance:**

Making backup copies of the database and repairing damage to the database due to hardware or software failures or misuse.

1. List the steps to draw E-R diagrams and Design E-R diagram for Education Portal
2. Explain about Concurrency Control and also explain about Time stamp based protocol.

Concurrency Control

Concurrency control is the essential management procedure required in a DBMS for coordinating the simultaneous execution of transactions without allowing them to conflict with each other.

Necessity: Concurrency control methods are required to:

1. Ensure isolation (mutual exclusion) between conflicting transactions.

2. Resolve potential read-write and write-write conflict issues.

3. Preserve database consistency.

4. Help ensure the schedules executed are serializable, meaning the final result is correct and reliable.

1. Explain Serial schedule with suitable Example.

schedule (or history) refers to the order in which the operations of transactions running concurrently are executed and interleaved.

A Serial Schedule is defined as a schedule where the operations of transactions are executed sequentially. In a serial schedule, the operations of one transaction are executed completely from beginning to end, including commit or abort, before the operations of the next transaction begin.

Importance

Serial schedules are inherently correct because they execute transactions one after the other, making interference impossible. The concept of serializability—a key correctness measure in concurrency—is based on ensuring that a concurrent schedule produces the exact same result as one of the possible serial schedules.

A screenshot of a computer program

AI-generated content may be incorrect.

1. Define Integrity Constraints. What is the main purpose of implementing Integrity in dbms? Explain its types.

Integrity constraints are rules defined upon the database schema that must be satisfied by every instance or state of the database. They are essential rules applied to the type of data stored in a table.

Main Purpose of Implementing Integrity

The primary purpose of implementing constraints is to maintain the accuracy and validity (integrity) of the data. Constraints ensure that any transition (via insertion, deletion, or update operations) takes the database from one valid state to another valid state.

Types of Integrity Constraints

1. Domain Constraints:

    ◦ Definition: These specify that the value of each attribute must be an atomic value and must be drawn from the attribute's defined domain.

    ◦ Detail: Domains define acceptable data types (integers, strings) and often restrict the possible set of values (e.g., using a constraint to limit department numbers).

2. Key Constraints:

    ◦ Definition: These constraints enforce the property of uniqueness on tuples within a relation, ensuring every record is distinguishable.

    ◦ Detail: Every relation must have at least one candidate key, one of which is chosen as the Primary Key (PK). The constraint is often used to specify alternate keys.

3. Entity Integrity Constraints:

    ◦ Definition: This crucial constraint dictates that the value of the Primary Key cannot be NULL.

    ◦ Detail: Because the primary key is meant to uniquely identify individual tuples, allowing NULL values would prevent unique identification and violate the entity's distinct existence.

4. Referential Integrity Constraints:

    ◦ Definition: These constraints are specified between two related tables using a Foreign Key to reference the Primary Key of another table.

    ◦ Detail: This mechanism maintains consistency by ensuring that relationships between records remain valid. The foreign key values must either match a valid primary key value in the referenced table or be NULL.