DBMS Key Concepts

Each question carries 2 marks (2 x 5)

1. Examine and compare is foreign key, primary key and composite key? Demonstrate with 2 example.

or

Differentiate candidate key from super key. Justify your understanding with an example.

2. Describe Armstrong's axiom rules of inference.

or

Explain trivial dependency in DBMS with Example.

2

2

2

3. Describe about the mechanisms used to enforce the referential integrity in the relational 2 database.

or

How do constraints implement the data integrity in the relational database?

4. Why do you need to retrieve only necessary columns from tables?

or

Demonstrate the main differences between the DDL and DML with some examples?

5. With the help of an example explain Cascading rollback.

or

What is serializability? What are the different types of serializability?

Part - B

Attempt 2 questions

Each question carries 5 marks (5 x 2)

6. The relational schema R (A, B, C, D) has following functional dependencies-

AB→ CD

CA

D B Solve and analyze how many candidate keys are there in this relation? What are they? Show the detailed working.

or

Page 1 of 2

2

2

2

2

2

5

Here's a structured approach to address the given questions:

Part - A

Each question carries 2 marks (2 x 5):

1.

a. Examine and compare foreign key, primary key, and composite key.

- **Primary Key**: A unique identifier for records in a table. Example: `StudentID` in a `Students` table.
- **Foreign Key**: A column in one table referencing the primary key of another table to establish a relationship. Example: `ClassID` in a `Students` table referencing the `Classes` table.
- **Composite Key**: A combination of two or more columns to uniquely identify a record. Example: `OrderID` and `ProductID` together as a primary key in an `OrderDetails` table.

Comparison:

- A primary key uniquely identifies a row within the same table, while a foreign key establishes a relationship between two tables.
- A composite key is a special case of a primary key that uses multiple columns.

b. Differentiate candidate key and super key.

- **Super Key**: Any set of attributes that can uniquely identify a row.
- **Candidate Key**: A minimal super key, meaning no subset of attributes can uniquely identify a row.

Example: In a table `Employee (EmpID, Email, Phone)`, the super keys are:

- `{EmpID}`, `{Email}`, `{EmpID, Email}`, `{EmpID, Phone}`, etc.
- Candidate keys: `{EmpID}` and `{Email}` (minimal sets).

2.

a. Describe Armstrong's axiom rules of inference.
Armstrong's Axioms are a set of rules used to infer all functional dependencies (FDs) from a given set of FDs:

- **Reflexivity**: If `Y` is a subset of `X`, then `X → Y`.
- **Augmentation**: If `X → Y`, then `XZ → YZ`.
- **Transitivity**: If `X → Y` and `Y → Z`, then `X → Z`.

Additional Rules Derived: Union, decomposition, and pseudotransitivity.

b. Explain trivial dependency in DBMS.
A **trivial dependency** occurs when the dependent attribute(s) are a subset of the determinant. Example:

- `X → Y` is trivial if `Y ⊆ X`.
- For a relation `R(A, B)`: `AB → A` is trivial since `A` is part of `AB`.

3.

a. Mechanisms used to enforce referential integrity in a relational database:

- **Foreign Key Constraints**: Ensure a value in a foreign key column exists in the referenced table.
- **ON DELETE/ON UPDATE**: Actions like `CASCADE`, `SET NULL`, `RESTRICT`.
- **Triggers**: Custom logic for complex scenarios.

b. Constraints for data integrity:

- **Domain Constraints**: Restrict column values (e.g., data type, range).
- **Entity Integrity**: Primary key must be unique and not null.
- **Referential Integrity**: Foreign key values must match primary key values in the referenced table.

4.

a. Why retrieve only necessary columns?

- Reduces memory and bandwidth usage.
- Improves query performance.
- Minimizes risk of exposing sensitive data unnecessarily.

b. Differences between DDL and DML with examples:

- **DDL (Data Definition Language)**: Defines database schema. Example:

```sql
CREATE TABLE Students (ID INT, Name VARCHAR(50));
ALTER TABLE Students ADD Email VARCHAR(100);
```

- **DML (Data Manipulation Language)**: Manipulates data in tables. Example:

```sql
INSERT INTO Students (ID, Name) VALUES (1, 'John');
SELECT * FROM Students;
```

5.

a. Cascading rollback with example:

- Occurs when a failure in one transaction causes dependent transactions to fail.
  Example:
- `T1` transfers $100 and updates balance.
- `T2` calculates interest based on the new balance.
- If `T1` fails, `T2` must also roll back.

b. Serializability:
Ensures that the execution of transactions is equivalent to some serial order.
Types of Serializability:

- **Conflict Serializability**: No conflicting operations (e.g., read-write on the same data).
- **View Serializability**: Final result matches a serial schedule, even if operations differ.

Part - B

Attempt 2 questions, each carries 5 marks (5 x 2):

6. **Candidate keys in schema `R(A, B, C, D)` with FDs `AB → CD` and `CA → B`:**
   - Attributes: `A, B, C, D`.
   - Closure: Compute for combinations.
     1. `{A, B}`: `{A, B}+ = {A, B, C, D}` → Candidate key.
     2. `{C, A}`: `{C, A}+ = {A, B, C, D}` → Candidate key.
     3. No other subset forms a candidate key.

   Candidate Keys: `{A, B}` and `{C, A}`.

Would you like detailed working for any specific question?

X

Y

Z

4

5

2

3

3

2

5

1

3

6

2

Following are the functional dependencies given. Analyse if YZ→X, YZ, XZY, ZY are valid FD. Justify your answer.

7.

Consider the following relations: -

Employee (EmpID, Name, DeptID),

Department (DeptID, DeptName, Location), Project (ProjectID, DeptID, ProjectName).

5

Write a Relational Algebra expression to find the names of employees who work in the departments that have projects located in "New York".

or

A database consists of the following tables: -

5

EMPLOYEE (EMP_NO, EMP_NAME, Dept_ID, Birth_date, Address, Gender, Salary, Manager_ID)

DEPARTMENT (Dept_Id, Dept_Name).

(i) Create a view that has the employee's name, manager name, and employee salary for each employee who works in the 'Human Resources' department.

(ii) Retrieve all the rows from the created view.

Part - C

Attempt 1 question

Each question carries 10 marks (10 x 1)

8. Check whether the given schedule S is conflict serializable and recoverable or not- S: R2(X), W3(X), Commit3, W1(X), Commit1, W2(Y), R2(Z), Commit2, R4(X), R4(Y), Commit4.

10

Is the schedule view serializable? Give proper justification.

or

"Show that the two-phase locking protocol ensures conflict serializability, and that transactions can be serialized according to their lock points"- Comment on the correctness of the statement.

10

Answers

Part - A

1. **Analyze the validity of the functional dependencies (FDs):**
   YZ → X, YZ, XZY, ZY
   - **YZ → X:**
     Valid if `YZ+` (closure of YZ) includes `X`. To verify:
     Calculate `YZ+` considering all given FDs. If `YZ → X` is derivable, it's valid.
   - **YZ:**
     Valid if `YZ` is a candidate key, meaning `YZ+` includes all attributes in the relation.
   - **XZY:**
     This is redundant since it includes `YZ`, which already determines other attributes if `YZ` is a candidate key.
   - **ZY:**
     Equivalent to `YZ`, so its validity depends on the derivation of other attributes using its closure.

   Detailed analysis would depend on the specific functional dependencies provided.

---

2. **Relational Algebra Expression:**

Find the names of employees who work in departments that have projects located in "New York":

Relations:

- `Employee (EmpID, Name, DeptID)`
- `Department (DeptID, DeptName, Location)`
- `Project (ProjectID, DeptID, ProjectName)`

Relational Algebra Expression:

```
scss


  Π_Name(σ_Location='New York' (Department ⋈ DeptID=DeptID Project ⋈ DeptID=DeptID Employ
  ee))
```

Explanation:

- Perform a natural join between `Department` and `Project` on `DeptID` to find departments with projects.
- Select rows where `Location = 'New York'`.
- Join the result with `Employee` using `DeptID`.
- Project the `Name` column.

---

3. **View Creation and Retrieval (SQL Example):**

(i) Create a view for employees in 'Human Resources':

```
sql


  CREATE VIEW HR_Employees AS
  SELECT
      E.EMP_NAME AS EmployeeName,
      M.EMP_NAME AS ManagerName,
      E.Salary
  FROM EMPLOYEE E
  JOIN EMPLOYEE M ON E.Manager_ID = M.EMP_NO
  JOIN DEPARTMENT D ON E.Dept_ID = D.Dept_ID
  WHERE D.Dept_Name = 'Human Resources';
```

(ii) Retrieve all rows from the view:

```
sql


  SELECT * FROM HR_Employees;
```

Part - C

8. **Conflict Serializability and Recoverability Analysis:**

Given Schedule:

`S: R2(X), W3(X), Commit3, W1(X), Commit1, W2(Y), R2(Z), Commit2, R4(X), R4(Y), Commit4`

Steps to Analyze Conflict Serializability:

1. Identify the transactions (`T1, T2, T3, T4`).
2. Draw a precedence graph:
   - Add an edge between transactions if a conflict exists (e.g., read-write, write-read, or write-write on the same data item).
3. If the graph has no cycles, the schedule is conflict serializable.

Recoverability Check:

- A schedule is recoverable if no transaction commits before all transactions whose changes it depends on have committed.
- Example: If `T3` writes `X` and `T2` reads it, `T2` should commit only after `T3`.

View Serializability Analysis:

- Check if the final state of the schedule matches any serial schedule of the same transactions.

Result:

- Conflict serializability depends on the presence of cycles in the precedence graph.
- For recoverability, ensure no commits violate dependency order.

OR

Two-Phase Locking Protocol and Conflict Serializability:

- **Correctness Statement:**
  The two-phase locking (2PL) protocol ensures conflict serializability because:
  - All locking actions (acquire locks) occur before unlocking actions (release locks).
  - The serialization order corresponds to the order of lock points (the point when a transaction acquires its final lock).
- **Proof:**
  - **Conflict Serializability:**
    Transactions following 2PL ensure no conflicting operations overlap because of exclusive and shared locks. This avoids cycles in the precedence graph.
  - **Serialization by Lock Points:**
    If `T1` acquires its last lock before `T2`, then `T1`'s operations can be serialized before `T2`.

Would you like a detailed precedence graph for the conflict serializability question?