Hotwax Training

Assignment Day 1

TASK 1: Levels of isolation.

The levels of isolation define the degree up to which the isolation between multiple concurrent transactions.

There are major 4 isolation levels:

a) **Dirty Read:** Dirty read is a problem in which a transaction reads the value of uncommitted transaction, called as Uncommitted read also.
At this stage, the transactions are not properly isolated from each other.

b) **Committed Read:** At this isolation level, a transaction only reads a data if it is committed, hence dirty read problem doesn't occur here.
In this level, the transaction applies the lock while operating on some data, thus prevents the reading, writing or updating from other transactions.

c) **Repeatable Read:** When a transaction reads the value twice with the same data, then we call it a repeatable read problem.
More specifically, a transaction applies locks on the data which it is accessing due to which, other transactions cannot read, update, or write the same data value.

d) **Serializable:** This is the highest isolation level. A serializable execution is guaranteed to be serializable. Serializable execution is defined to be an execution of operations in which concurrently executing transactions appears to be serially executing.

TASK 2: Difference between Query and Statement.

Query - A query is a command or operation that returns some result from a database.
Example: SELECT, SHOW, DESCRIBE etc.

Statement - A statement is the operation that is performed on a database, and may or may not return results.
Example: INSERT, UPDATE, DELETE, ALTER, etc.

TASK 3: Difference between a Key and Primary Key.

Key - A key is a unique identifier for each row that may be composed of a single or multiple attributes.
There types of keys include:

a) Primary Key
b) Candidate Key
c) Super Key
d) Composite Key
e) Foreign Key

Primary Key - A Primary Key is an attribute that uniquely identifies each tuple/row in a relation. The two most important properties of primary key are:

a) It must be unique for each row i.e. no duplicacy, and
b) It cannot be null

TASK 4: Difference between DROP, DELETE and TRUNCATE.

DROP:

- It's a Data Definition Language (DDL) Command. It deletes the table structure along with the data completely.
- The deleted table cannot be rolled back after performing the DROP command.
- This can be performed to drop/remove all the schemas like, Databases, Views, Tables, etc.

Example:

- DROP TABLE <table_name>;
- DROP <database_name>;
- DROP <view_name>;

DELETE:

- It is a Data Manipulation Language (DML) command. The function of this command is to delete the selected rows in a table but not the table itself.
- We can use the WHERE clause with this command to select the desired tuples.
- The deleted data can be rolled back upto a SAVEPOINT if the Savepoint is created.
- This can be used to delete tuples in the table, not the table or any kind of schema.

Example:

- DELETE FROM <table_name> WHERE (condition);
- DELETE FROM students WHERE id = 2;
- DELETE * FROM students; (This will delete all the rows from the table)

TRUNCATE:

- This is a Data Definition Language (DDL) Command.

- Function is to delete the entire table data at once but not the schema.
- This doesn't support the rollback of deleted data in most of the databases.

Example:

- TRUNCATE TABLE <table_name>;

TASK 5: Which Command is Faster, DELETE or TRUNCATE? Give reason for the answer.
TRUNCATE is faster than DELETE.

Reason:

DELETE removes rows one by one, and logs each row deletion in the transaction log.
TRUNCATE deallocates entire data pages (removes all rows at once) and logs only the page deallocation, not each row.
More specifically, Truncate does not generate any transaction logs and hence it is faster than Delete.

Example: Delete means, erasing each line written in a page one by one in a notebook.
Truncate means tearing the entire page at once.

TASK 6: What is the second and third normal form? Write a conversion query.

2NF:
We used to convert the existing table in 2nd normal form to reduce the redundancy in the database.
A table is in 2nd normal form:
- If it is already in its first normal form.
- If it has no partial dependency.

Partial Dependency: The non functional attributes should be derived just from the primary key, not by its any part.

3NF:
A table in database is said to be in 3rd normal form if:
- It is already in 2nd normal form.
- It has no transitive dependency.

Transitive dependency: When a non-prime attribute is derived by a non-prime attribute not by Primary Key.

CONVERSION INTO 2NF:

Table: Student_Course
Columns: (student_id, course_id, student_name, course_name, instructor_name)

Primary Key: (student_id, course_id)

S_id(PK)	S_name	C_id(PK)	C_name	Ins_name
01	Payal	C01	DBMS	Mr. Rajput
02	Payal	C02	OOPM	Mr. Namdev
03	Shreya	C05	UI/UX	Mr. Jadhav

This table has partial dependency as student name and course name depends on Student ID and Course ID, and if student leaves, course will also be gone. So it produces deletion anomaly and redundant data.

To convert it into 2NF, we can split this into three tables as shown below:

1. Student:

S_id(PK)	S_name
01	Payal
03	Shreya

2. Course:

C_id(PK)	C_name	Ins_name
C01	DBMS	Mr. Rajput

C02	OOPM	Mr. Namdev
C05	UI/UX	Mr. Jadhav

3. Student_course:

S_id	**C_id**
01	C01
01	C02
02	C05

This way, the table is now converted into 2NF.

CONVERSION INTO 3NF:

In the above three tables, there exists a transitive dependency, i.e. if from the course table, if the name of any teacher/instructor be deleted, then the subject will also be deleted. Means, when a teacher leaves the college, it doesn’t mean that the course will be removed from college. Thus we need to convert this into 3NF shown below:

Now create the tables like this:

1. Student:

S_id(PK)	**S_name**
01	Payal
03	Shreya

2. Courses:

C_id(PK)	**C_name**	**Ins_id**
C01	DBMS	I001
C02	OOPM	I002
C05	UI/UX	I003

3. Instructor:

Ins_id(PK)	**Ins_name**
I001	Mr. Rajput
I002	Mr. Namdev
I003	Mr. Jadhav

4. Student_course:

S_id	**C_id**
01	C01
01	C02
02	C05

So in this way, the two nf can be converted into 3nf to get rid of anomalies.