Data Manipulation and Transactions

Celestin Niyomugabo

Common SQL commands

Data Manipulation Language (DML)

- **SELECT** – Retrieve data.
- **INSERT** – Add new records.
- **UPDATE** – Modify existing records.
- **DELETE** – Remove records.

Data Definition Language (DDL)

- **ALTER TABLE** – Modify a table structure.
- **ADD COLUMN** – Add a new column.
- **DROP COLUMN** – Remove an existing column.
- **CREATE TABLE** – Define a new table.
- **DROP TABLE** – Delete an entire table.

Transactions and Data Integrity

Transactions ensure **atomicity, consistency, isolation, and durability (ACID).**

Used to maintain **data integrity** in case of failures.

ACID property explained

Atomicity

- Ensures that a transaction is **all-or-nothing**.
- If one part of the transaction fails, the entire transaction is rolled back.
- Example: If a customer rents a movie and the payment process fails, the rental should not be recorded.

Consistency

- Ensures the database remains in a valid state before and after a transaction.
- Transactions should follow all database constraints (e.g., foreign keys, unique keys).
- Example: A rental entry should not be recorded if the movie does not exist in inventory.

ACID property explained

Isolation

- Ensures that concurrent transactions do not interfere with each other.
- Different transactions should execute independently until committed.
- Example: If two customers try to rent the same DVD copy, isolation prevents double booking..

Durability

- Ensures that once a transaction is committed, it remains saved even in case of system failure.
- The changes are permanently recorded in the database.
- Example: A completed payment transaction should not be lost after a system crash.

Commands used to ensure data integrity and ACID compliance

- **BEGIN TRANSACTION** – Start a transaction.
- **COMMIT** – Save changes.
- **ROLLBACK** – Undo changes if an error occurs.

Example:

```
START TRANSACTION;
INSERT INTO rental (rental_date, inventory_id, customer_id, staff_id, return_date)
VALUES (NOW(), 5, 23, 1, NULL);

UPDATE inventory SET status = 'rented' WHERE inventory_id = 5;

COMMIT;
```

Safely roll back in case of failure

```
START TRANSACTION;
UPDATE payment SET amount = amount - 5 WHERE customer_id = 23;
IF @@ERROR THEN ROLLBACK; ELSE COMMIT;
```