

Module 2 - Lecture 4

# INSERT, UPDATE, DELETE

&

# Transactions, Constraints, and Referential Integrity



# REVIEW

- Keys
- Cardinality
- Joins
- Unions



# Inserting information

## INSERT

- Inserts one row into a table.

```
INSERT INTO table_1 (column_1, ... , column_n)
VALUES (value_1, ... , value_n);
```

- Inserts 0 to many rows into a table from another table

```
INSERT INTO table_1 (column_1, ... , column_n)
SELECT column_1, ... , column_n
FROM table_2
[WHERE [...]];
```



# Updating information

**UPDATE** - Updates 0 to many rows in a table.

```
UPDATE table_1  
SET      column_1 = value_1  
WHERE    column_2 = value_2;
```



# Deleting information

**DELETE** - Deletes 0 to many rows in a table.

**\*\* There are many reasons NOT to delete data \*\***

```
DELETE FROM table_1 WHERE column_1 = value_1;
```

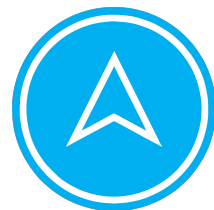


# Constraints

A **constraint** is associated with a table and defines properties that the column data must comply with.

## Types of Constraints

1. **NOT NULL**
2. **UNIQUE**
3. **PRIMARY KEY** - allows FKs to establish a relationship, and enforces NOT NULL and UNIQUE,
4. **FOREIGN KEY** - enforces valid PK values, and limits deletion of the PK row if FK row exists
5. **CHECK** - specifies acceptable values that can be entered in the column
6. **DEFAULT** - provides a default value for the column



# Transactions

A **transaction** is a single unit of work.

We can use a transaction to execute multiple statements and commit them if they are all successful.

If any statement is unsuccessful, we can rollback a transaction to prevent any of the statements from applying to our database.

**BEGIN TRANSACTION**

[SQL statements]

[**ROLLBACK** | **COMMIT**] **TRANSACTION;**



# A.C.I.D.

**The ACID Test** to determine whether a series of actions is a transaction, they need to have the following characteristics.

1. **Atomicity:** Within a transaction, a series of database operations all occur or none occur.
  - A withdrawal from savings should not be recorded unless the deposit to checking was successful.
2. **Consistency:** The completed transaction leaves things remaining in a consistent state at the end. Any rules in place before the transaction still pass after the transaction.
  - \$100 cannot be withdrawn from one account and never deposited to the other account. The consistency (balance) before the transaction must be the same after the transaction.
3. **Isolation:** Ensures that the concurrent execution of a transaction results as if the operations were executed serially.
  - Prevents two customers from transferring the same \$100 at the same time.
  - If two customers transfer the same \$ from their account to their friend's account concurrently, the database should treat them as sequential operations.
  - A typical system will revert to the last known good state.
4. **Durability:** Once a transaction has been committed it will remain so, even during a power loss, crash, or an error.
  - This is generally handled through journaling.





QUESTIONS?

