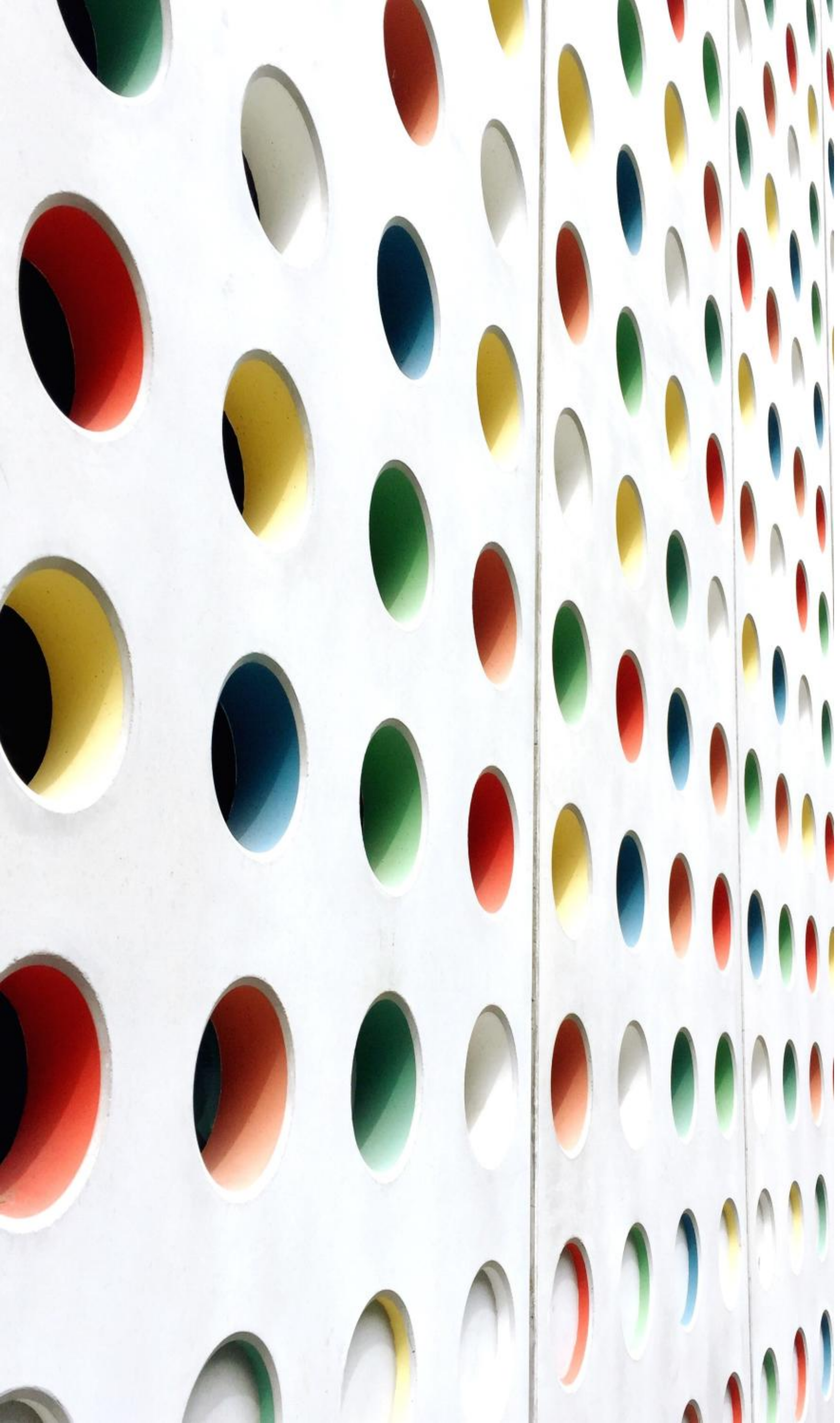

CIS 11051- PRACTICAL FOR DATABASE DESIGN

SQL CONSTRAINTS

Constraints

- Constraints are like rules that you set for the data in a table. These rules help make sure that the data entered into the table is correct and follows certain restrictions.

  - **Column-level constraints:** These rules are applied to one specific column.

  - **Table-level constraints:** A table-level constraint is a rule that involves multiple columns in a table, or even the entire table.

COMMON SQL CONSTRAINTS

- **NOT NULL** : Ensures that a column **cannot have a NULL value**.
- **UNIQUE** : Ensures that **all values in a column are unique and no duplicates are allowed**.
- **PRIMARY KEY** : **A combination of NOT NULL and UNIQUE.** It uniquely identifies each row in a table and must have unique values for each record.
- **FOREIGN KEY** : **Establishes a relationship between two tables.** It prevents actions that would destroy links between the tables, like deleting a row that's being referenced.
- **DEFAULT** : **Sets a default value for a column** when no value is provided during data insertion.
- **CHECK : Limits the range or type of values** that can be inserted into a column. For example, it can ensure that a value is within a specific range.

1. NOT NULL CONSTRAINT

- NOT NULL ensures that a column cannot have a NULL value.
- You can apply the NOT NULL constraint both when creating a table (e.g., CREATE TABLE) or when modifying an existing table (e.g., ALTER TABLE).
- It's also true that you can have more than one NOT NULL column in a table. In fact, a table can have multiple columns marked as NOT NULL depending on the design.

NOT NULL CONSTRAINT ON CREATE TABLE.

```
CREATE TABLE Person (
    PersonID int NOT NULL,
    LastName varchar(50) NOT NULL,
    FirstName varchar(50) NOT NULL,
    Age int
);
```

- PersonID, LastName, and FirstName columns are defined with the NOT NULL constraint, meaning **they will not accept NULL values.**
- The Age column does not have the NOT NULL constraint, so it can accept NULL values.

NOT NULL ON ALTER TABLE

- To add a NOT NULL constraint to an existing column in a table using the ALTER TABLE statement, you can use the following syntax:

```
ALTER TABLE table_name
MODIFY column_name column_type NOT NULL;
```

E.g. :

If you want to add the NOT NULL constraint to the Age column in the Person table, the query will look like this:

```
ALTER TABLE Person
MODIFY Age int NOT NULL;
```

DROP NOT NULL CONSTRAINT

- In MySQL, you can't directly drop a NOT NULL constraint. Instead, you modify the column to remove the NOT NULL property:

```
ALTER TABLE table_name
MODIFY column_name datatype;
```

E.g. :

For example, if the Age column in the person table has a NOT NULL constraint and you want to remove it:

```
ALTER TABLE person
MODIFY Age INT;
```

2. UNIQUE CONSTRAINTS

- UNIQUE Constraint ensures that all values in a column (or a combination of columns) are distinct, meaning no two rows can have the same value in that column or set of columns. This helps maintain data integrity.
- User can have more than one column with the UNIQUE constraint in a table. Each of these columns will have unique values.
- UNIQUE and PRIMARY KEY constraints both ensure uniqueness. However, there are differences:
  - A PRIMARY KEY constraint also implicitly includes a NOT NULL constraint, meaning the column(s) cannot have NULL values.
  - A UNIQUE constraint does not inherently prevent NULL values. You can have multiple NULL values in a column with a UNIQUE constraint.

UNIQUE CONSTRAINT ON CREATE TABLE

- When creating a table, user can define a column with the UNIQUE constraint as follows:

Syntax:

```
CREATE TABLE table_name (
column_name datatype UNIQUE,
another_column datatype,
...
);
```

UNIQUE CONSTRAINT ON ALTER TABLE

- Syntax:

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name UNIQUE (column_name);
```

- E.g.:

```
ALTER TABLE Employees
ADD CONSTRAINT unique_email UNIQUE (Email);
```

- This statement adds a UNIQUE constraint to the Email column in the existing Employees table and the constraint is named unique_email.

DROP UNIQUE CONSTRAINT

- To drop a UNIQUE constraint, you must first know the name of the constraint. You can find the constraint name by checking the table structure or information schema.

```
ALTER TABLE table_name
DROP INDEX constraint_name;
```

E.g. :

For example, if the UNIQUE constraint on the email column of the person table is named unique_email, you would run:

```
ALTER TABLE person
DROP INDEX unique_email;
```

3. PRIMARY KEY CONSTRAINT

- PRIMARY KEY constraint uniquely identifies each record in a table. It ensures that each record has a unique identifier.
- Primary keys must contain unique values and cannot contain NULL values.
- Primary Key on Create table

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY,
);
```

- Primary Key on Alter table

```
ALTER TABLE table_name
ADD PRIMARY KEY (column_name);
```

DROP A PRIMARY KEY CONSTRAINT

- **You cannot drop the primary key if other constraints (like foreign keys) are dependent on it.** You need to remove those foreign key constraints before dropping the primary key.
- **If the primary key is part of a composite primary key, it will drop the whole composite primary key constraint.**
- Syntax:

```
ALTER TABLE table_name
DROP PRIMARY KEY;
```

COMPOSITE PRIMARY KEY

- **A composite primary key is a primary key that consists of more than one column in a table.** It is used when a single column is not sufficient to uniquely identify each record in a table, and a combination of multiple columns is needed to ensure the uniqueness of each row.
- Key Points:
  - It combines multiple columns into a single primary key.
  - All the columns in the composite primary key together must contain unique values.
  - None of the columns involved in a composite primary key can have NULL values.
  - It is commonly used when there is no single column that can uniquely identify each row

COMPOSITE PRIMARY KEY ON CREATE TABLE

- Syntax:

```
CREATE TABLE table_name (
            column1 datatype,
            column2 datatype,
            ..,
            PRIMARY KEY (column1, column2)
);
```

- E.g.:

```
CREATE TABLE OrderDetails (
   OrderID int,
   ItemID int,
   Quantity int,
   PRIMARY KEY (OrderID, ItemID)
);
```

COMPOSITE PRIMARY KEY ON ALTER TABLE

- Syntax:

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name
PRIMARY KEY (column1, column2);
```

- E.g.:

```
ALTER TABLE OrderDetails
ADD CONSTRAINT pk_order_item
PRIMARY KEY (OrderID, ItemID);
```

DROP A COMPOSITE KEY

- **Example 1: Dropping a Composite Primary Key (without constraint name):**

If you created a composite primary key on the OrderDetails table (with columns OrderID and ItemID), and you want to drop the primary key, you can use:

```
ALTER TABLE OrderDetails
DROP PRIMARY KEY;
```

- **Example 2: Dropping a Composite Primary Key (with constraint name):**

If you named the composite primary key constraint (e.g., pk_order_item), you can drop it using the constraint name:

```
ALTER TABLE OrderDetails
DROP CONSTRAINT pk_order_item;
```

4. Foreign Key Constraint

- The table that contains the **foreign key** is called the **child table**
- The table that contains the **primary key** is called the **parent table or referenced table.**
- A foreign key is a field (or collection of fields) in a table **that uniquely identifies a row in another table.** The foreign key points to the primary key of the referenced (or parent) table.
- The foreign key constraint ensures that the relationship between the child and parent tables remains intact by preventing actions (like deleting or updating data) that would break the link between the tables.

Example: **Connect the Customers and Orders table using foreign key**

- **Customers (Parent Table)**

```
CREATE TABLE Customers (
    CustomerID int PRIMARY KEY,
    Name varchar(100)
);
```

- **Orders (Child Table)**

```
CREATE TABLE Orders (
    OrderID int PRIMARY KEY,
    NewCus_ID int,
    OrderDate date,
    FOREIGN KEY (NewCust_ID) REFERENCES Customers(CustomerID)
);
```

Ensure that, NewCus_ID in the Orders table and CustomerID in the Customers table have the same data type.

4. FOREIGN KEY ON ALTER TABLE

- You can use the ALTER TABLE statement to add a foreign key to an existing table
- Syntax

```
ALTER TABLE child_table
ADD CONSTRAINT constraint_name
FOREIGN KEY (child_column) REFERENCES parent_table (parent_column);
```

- Assume you want to add a foreign key to the Employees table that references the CustomerID in the Customers table.
- Step 1: Ensure that both tables are set up
  - The Customers table should have a CustomerID column (usually the primary key).
  - The Employees table should have a column (e.g., CusID) to store the foreign key.
- Step 2: Add CustomerID column to Employees table **(if not already there)**

```
ALTER TABLE Employees

ADD CusID int;
```

- Step 3: Add the Foreign Key to the Child Table (Employees):

```
ALTER TABLE Employees

ADD CONSTRAINT fok_customer

FOREIGN KEY (CusID)

REFERENCES Customers(CustomerID);
```

DROP A FOREIGN KEY

- To drop a foreign key constraint from a table, you need to use the ALTER TABLE statement. If you created the foreign key with a specific constraint name, you can drop it by referencing that name.

```
ALTER TABLE table_name
DROP FOREIGN KEY constraint_name;
```

- E.g. :

```
ALTER TABLE Employees
DROP FOREIGN KEY fok_customer;
```

5. DEFAULT CONSTRAINT

- A default constraint in SQL is used to assign a default value to a column when no value is provided during an INSERT operation.
- This ensures that a column always has a value, even if one is not explicitly specified by the user
- Syntax:

```
CREATE TABLE table_name (
column_name datatype DEFAULT default_value
);
```

- E.g. :

```
CREATE TABLE Persons (
        City varchar(255) DEFAULT 'Washington_DC'
        );
```

- **DEFAULT CONSTRAINT ON ALTER TABLE**

```
ALTER TABLE table_name
ALTER COLUMN column_name SET DEFAULT default_value;
```

- E.g.:

```
ALTER TABLE drinks
ALTER COLUMN Company SET DEFAULT 'PepsiCo' ;
```

DROP DEFAULT CONSTRAINT

- To remove the default value from a column in MySQL, you can use the ALTER TABLE statement, **as MySQL does not support directly dropping a default value using DROP CONSTRAINT.**
- Syntax:

```
ALTER TABLE table_name
MODIFY column_name column_definition;
```

- E.g.:

```
ALTER TABLE employees
MODIFY City VARCHAR(255);
```

6. CHECK CONSTRAINT

- The CHECK constraint ensures that the values in a column meet a certain condition
- For MySQL 8.0 and above, CHECK constraints are supported, but they are not enforced in older versions.
- Syntax:

```
CREATE TABLE table_name (
column_name datatype,
...
CONSTRAINT constraint_name CHECK (condition)
);
```

- E.g. :

```
CREATE TABLE person (
salary INT,
Age INT,
CONSTRAINT chk_age CHECK (Age >= 18)
);
```

- **CHECK CONSTRAINT ON ALTER TABLE**
- Syntax :

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name CHECK (condition);
```

- E.g.:

```
ALTER TABLE person
ADD CONSTRAINT chk_salary CHECK (salary >= 30000);
```

DROP CHECK CONSTRAINT

- To drop a CHECK constraint, you need to know its constraint name. After identifying the constraint name, you can use:
- Syntax:

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

- E.g.:

```
ALTER TABLE person
DROP CONSTRAINT chk_age;
```

THANK YOU !