04 Relational Model Constraints

Types of Relational Model Constraints

1. **Domain Constraints**
2. **Entity Integrity Constraint**
3. **Referential Integrity Constraint**
4. **Key Constraints**
5. **Unique Constraint**
6. **Not Null Constraint**
7. **Check Constraint**

01 Domain Constraints

- **Domain Constraints** specify that the values in a column must be from a specific domain
- A domain is essentially the set of valid values for a column
- This constraint ensures that only the correct data type and value range are entered into a column

Example

- If a column is defined to store ages, the domain constraint might specify that only integer values between 0 and 120 are allowed

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(255),
    Age INT CHECK (Age >= 0 AND Age <= 120)
);
```

- In this example, the `Age` column has a domain constraint ensuring that only values between 0 and 120 are valid.

02 Entity Integrity Constraint

- **Entity Integrity Constraint** ensures that every table has a primary key, and that the primary key must be unique and not null
- This constraint ensures that each record in the table can be uniquely identified

Example

- A table of `Employees` must have a unique `EmployeeID` for each record, which serves as the primary key

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(255)
);
```

- Here, `EmployeeID` is the primary key, and the entity integrity constraint ensures that it is unique and cannot be null

03 Referential Integrity Constraint

- **Referential Integrity Constraint** ensures that a foreign key value in one table corresponds to a primary key value in another table
- This constraint maintains the relationships between tables and ensures that the database doesn’t contain orphaned records

Example

- In an `Orders` table, the `CustomerID` must match an existing `CustomerID` in the `Customers` table

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(255)
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

- In this example, `CustomerID` in the `Orders` table is a foreign key that must match a `CustomerID` in the `Customers` table, ensuring referential integrity

04 Key Constraints

- **Key Constraints** are rules that apply to keys in a table

- The most common key constraints are the **Primary Key** and **Foreign Key** constraints
- **Primary Key** : Enforces uniqueness for the specified column(s). No two rows can have the same value for the primary key, and it cannot be null
- **Foreign Key** : Establishes a relationship between two tables and ensures that the value in one table corresponds to a valid primary key in another table

Primary Keys

- If a relation schema has more than one key, then each key is called a *candidate key*
- One of the candidate keys is designated as the primary key, and the others are called *secondary keys*
- In a practical relational database, each relation schema must have a primary key

Rules for primary keys

- The value of the Primary Key must be unique for each instance of the entity
- There can be no missing values(ie. Not Null) for Primary Keys. If the Primary Key is composed of multiple attributes, each of those attributes must have a value for each instance
- The Primary Key is immutable.i.e., once created the value of the Primary Key cannot be changed
- If the Primary Key consists of multiple attributes, none of these values can be updated

Example

- The `EmployeeID` in an `Employees` table might be the primary key, while `DepartmentID` in an `Employees` table might be a foreign key that references a `Departments` table

```sql
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(255)
);

CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(255),
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
```

- Here, `DepartmentID` is a foreign key in the `Employees` table, linking each employee to a department in the `Departments` table

05 Unique Constraint

- The **Unique Constraint** ensures that all the values in a column are different
- Unlike a primary key, a table can have multiple unique constraints, and a unique column can accept null values (though only one null value is typically allowed, depending on the DBMS)

Example

- A `Users` table might enforce a unique constraint on the `Email` column to ensure no two users share the same email address

```
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Username VARCHAR(255) UNIQUE,
    Email VARCHAR(255) UNIQUE
);
```

- In this example, both `Username` and `Email` columns are constrained to have unique values across the table

06 Not Null Constraint

- The **Not Null Constraint** ensures that a column cannot contain a null value
- This constraint is useful for columns that must always have a value, like a primary key or any other field where missing data is not acceptable

Example

- A `Products` table might require that every product have a `Name`

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Price DECIMAL(10, 2)
);
```

- Here, the `Name` column is constrained to never be null, ensuring that every product has a name

07 Check Constraint

- The **Check Constraint** is used to enforce a condition on the values in a column

- If the condition evaluates to false, the data is rejected
- It is used to limit the range of values that can be entered into a column

Example

- A `Students` table might require that the `Age` column contains values between 18 and 25

```
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(255),
    Age INT CHECK (Age >= 18 AND Age <= 25)
);
```

- In this example, the `Age` column must have values between 18 and 25, as specified by the check constraint
- **Domain Constraints** : Restrict the type of data that can be stored in a column
- **Entity Integrity Constraint** : Ensures that each table has a unique primary key and that the primary key cannot be null
- **Referential Integrity Constraint** : Maintains consistency between related tables through foreign keys
- **Key Constraints** : Enforce the uniqueness of primary keys and the validity of foreign keys
- **Unique Constraint** : Ensures that all values in a column are unique
- **Not Null Constraint** : Ensures that a column cannot contain null values
- **Check Constraint** : Validates that data entered into a column meets a specified condition