**Integrity Constraints in MySQL**

**Integrity constraints** in MySQL are rules applied to columns or tables to ensure that the data entered into the database is valid and maintains its consistency and accuracy. These constraints are implemented through column definitions and are enforced by MySQL to prevent data corruption.

**1. Entity Integrity**

* **Definition:** Ensures that each record in a table is uniquely identifiable. This is enforced by the **Primary Key** constraint.
* **Example:**
  + Let's create a table called Employees:

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

emp\_salary DECIMAL(10,2)

);

* + - **emp\_id** is the **Primary Key** for the Employees table. The **Primary Key** constraint ensures that:
      * Every record in the table must have a unique **emp\_id**.
      * The **emp\_id** cannot be NULL.

**2. Referential Integrity**

* **Definition:** Ensures that a relationship between two tables is consistent. This is enforced by the **Foreign Key** constraint, which links columns in one table to the **Primary Key** in another table.
* **Example:**
  + Let's create two tables: Departments and Employees:

CREATE TABLE Departments (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(50)

);

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES Departments(dept\_id)

);

* + - The **Foreign Key** constraint ensures that any value inserted into the dept\_id column of the Employees table must already exist in the dept\_id column of the Departments table.
    - If you try to insert an employee with a dept\_id that doesn’t exist in the Departments table, MySQL will reject the operation to preserve referential integrity.

**3. Domain Integrity**

* **Definition:** Ensures that values in a column belong to a specific set of values or range. This is handled by the **CHECK** constraint and **Data Types**.
* **Example:**
  + Suppose we want to ensure that employees' ages are non-negative:

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

emp\_age INT CHECK (emp\_age >= 0)

);

* + - The **CHECK** constraint ensures that the value of **emp\_age** cannot be less than 0.
    - Note that in MySQL 5.6 and earlier, the **CHECK** constraint was not enforced. In later versions (MySQL 8.0+), **CHECK** constraints are supported.

**4. User-Defined Integrity**

* **Definition:** Refers to custom business rules specific to the application. This includes enforcing conditions that go beyond the built-in constraints (such as enforcing minimum salary or ensuring a business rule).
* **Example:**
  + Suppose that the company has a rule that the minimum salary for an employee must be 3000. We can enforce this using the **CHECK** constraint:

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

emp\_salary DECIMAL(10,2),

CHECK (emp\_salary >= 3000)

);

* + - The **CHECK** constraint enforces that the **emp\_salary** must be at least 3000. If a salary less than 3000 is inserted, MySQL will reject the insertion.

**5. Unique Constraint**

* **Definition:** Ensures that all values in a column (or combination of columns) are unique across all rows in the table. It allows NULL values, but NULL can appear multiple times.
* **Example:**
  + Let’s ensure that each employee's email is unique:

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

email VARCHAR(100) UNIQUE

);

* + - The **email** column must have a unique value across all records in the Employees table. If a duplicate email is inserted, MySQL will reject it.

**6. Not Null Constraint**

* **Definition:** Ensures that a column cannot have a NULL value. This is useful when you need to enforce that a column must always have a value.
* **Example:**
  + Let’s ensure that the email column cannot have a NULL value:

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

email VARCHAR(100) NOT NULL

);

* + - The **NOT NULL** constraint ensures that an **email** must always be provided when inserting or updating a record. If you try to insert a record without an email, MySQL will reject it.

**7. Default Constraint**

* **Definition:** Sets a default value for a column when no value is provided during insert. It ensures the column has a valid value even if one is not supplied by the user.
* **Example:**
  + Suppose we want to set a default value of 0 for the emp\_salary if no value is provided:

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

emp\_salary DECIMAL(10,2) DEFAULT 0

);

* + - If no salary is provided when a new employee is added, the default salary of 0 will be used.

**8. Auto Increment**

* **Definition:** Automatically generates a unique value for a column (often used for primary keys). This is particularly useful for generating unique identifiers for new records.
* **Example:**
  + Let’s modify the emp\_id to be auto-incremented:

CREATE TABLE Employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_name VARCHAR(50),

emp\_salary DECIMAL(10,2)

);

* + - The **AUTO\_INCREMENT** attribute ensures that the **emp\_id** will automatically be assigned a unique integer value for every new row inserted into the table.

**Example of a Table with Multiple Constraints in MySQL:**

CREATE TABLE Employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_name VARCHAR(50) NOT NULL,

emp\_salary DECIMAL(10,2) CHECK (emp\_salary >= 3000),

email VARCHAR(100) UNIQUE,

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES Departments(dept\_id),

CHECK (emp\_salary >= 3000)

);

**Explanation:**

1. **Primary Key (emp\_id)**: Uniquely identifies each record and automatically increments with every new insertion.
2. **NOT NULL (emp\_name)**: Ensures the employee's name is always provided.
3. **CHECK (emp\_salary >= 3000)**: Ensures that the salary is greater than or equal to 3000.
4. **Unique (email)**: Ensures that each employee has a unique email.
5. **Foreign Key (dept\_id)**: Links the Employees table to the Departments table, ensuring that the department ID exists in the Departments table.