SQL Constraints

In MS SQL Server, constraints are rules applied to table columns to ensure the accuracy, integrity, and reliability of the data in the database. Here is a list of the major constraints:

1. NOT NULL Constraint

Ensures that a column cannot have NULL values.

Example:

```
CREATE TABLE Employees (
  EmployeeID int NOT NULL,
  FirstName varchar(50) NOT NULL
);
```

2. UNIQUE Constraint

Ensures that all values in a column (or a combination of columns) are distinct.

Example:

```
CREATE TABLE Employees (
  EmployeeID int NOT NULL UNIQUE,
  Email varchar(100) UNIQUE
);
```

3. PRIMARY KEY Constraint

- Uniquely identifies each row in a table. It combines the NOT NULL and UNIQUE constraints and ensures that the column (or group of columns) is unique and nonnull.
- A table can have only one PRIMARY KEY.

Example:

```
CREATE TABLE Employees (
  EmployeeID int NOT NULL PRIMARY KEY,
  FirstName varchar(50) NOT NULL
);
```

4. FOREIGN KEY Constraint

Ensures that the value in one column corresponds to values in another table, maintaining referential integrity between two tables.

Example:

```
CREATE TABLE Orders (
  OrderID int PRIMARY KEY,
  EmployeeID int,
  FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

5. CHECK Constraint

Ensures that the value in a column satisfies a specific condition.

Example:

```
CREATE TABLE Employees (
  EmployeeID int PRIMARY KEY,
  Age int CHECK (Age >= 18)
);
```

6. DEFAULT Constraint

Provides a default value for a column when no value is specified.

Example:

```
CREATE TABLE Employees (
  EmployeeID int PRIMARY KEY,
  Salary decimal(10,2) DEFAULT 5000.00
);
```

7. INDEX (NonConstraint, but Related)

Improves the speed of data retrieval operations on a table by creating indexes.

Example:

```
CREATE INDEX idx_employee_name ON Employees(FirstName);
```

8. UNIQUE Constraint on Multiple Columns

Ensures that the combined values of two or more columns are unique.

Example:

```
CREATE TABLE EmployeeContacts (
  EmployeeID int,
  ContactType varchar(50),
  ContactValue varchar(100),
  CONSTRAINT UQ_EmployeeContact UNIQUE (EmployeeID, ContactType)
);
```

9. ON DELETE / ON UPDATE (Referential Actions for FOREIGN KEY)

Specifies what happens to the rows in the child table when a corresponding row in the parent table is deleted or updated.

Options:

- ON DELETE CASCADE: Deletes related rows in the child table.
- ON UPDATE CASCADE: Updates related rows in the child table.
- SET NULL: Sets the foreign key column to NULL when the parent row is deleted/updated.
- SET DEFAULT: Sets the foreign key column to its default value.
- NO ACTION: Prevents deletion or updating of the parent row if related rows exist in the child table.

Example:

```
CREATE TABLE Orders (
  OrderID int PRIMARY KEY,
  EmployeeID int,
  FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

Summary of Constraints:
1. NOT NULL: Ensures a column cannot contain NULL values.
2. UNIQUE: Ensures all values in a column or a set of columns are unique.
3. PRIMARY KEY: Uniquely identifies each row in the table.
4. FOREIGN KEY: Establishes a relationship between two tables.
5. CHECK: Ensures values in a column meet a specific condition.
6. DEFAULT: Provides a default value for a column if no value is provided.

These constraints are essential for maintaining data integrity, consistency, and reliability in a SQL Server database.

Scenario 1: Employee Management System

Imagine you're building a table to store employee information for a company. You need to ensure that the following constraints are applied:

1. **NOT NULL**: Employees must have an `EmployeeID`, `FirstName`, and `LastName`. These fields should never be empty.
2. **UNIQUE**: Each employee must have a unique `Email`, and the `EmployeeID` must also be unique across all employees.
3. **PRIMARY KEY**: The `EmployeeID` uniquely identifies each employee.
4. **FOREIGN KEY**: Each employee belongs to a `Department`. The `DepartmentID` in the Employee table must reference the `DepartmentID` from a `Departments` table to ensure the department exists.
5. **CHECK**: Ensure that the employee's `Age` is at least 18 and their `Salary` is not negative.
6. **DEFAULT**: If an employee doesn't provide a `Country`, assume the employee is from "USA".
7. **ON DELETE CASCADE**: If a department is deleted from the `Departments` table, automatically delete all employees in that department.