RÀNG BUỘC (CONSTRAINT)

▼ CÔNG DỤNG

- Ràng buộc trong SQL là những quy tắc được áp dụng cho bảng, cột hoặc dữ liệu trong bảng nhằm đảm bảo tính toàn vẹn và nhất quán của dữ liệu.
- Ràng buộc giúp ngăn chặn việc nhập dữ liệu sai sót, đảm bảo dữ liệu chính xác và có ý nghĩa.

▼ NOT NULL

Đảm bảo rằng một cột không thể chứa giá trị NULL. Điều này có nghĩa là mỗi hàng phải có một giá trị cho cột đó.

```
CREATE TABLE Employees (
EmployeeID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
Age int
);
```

▼ UNIQUE

Đảm bảo rằng tất cả các giá trị trong một cột đều khác nhau, không có sự trùng lặp.

```
CREATE TABLE Employees (
    EmployeeID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

▼ PRIMARY KEY

Ràng buộc này kết hợp các tính năng của **NOT NULL** và **UNIQUE**. Nó đảm bảo rằng mỗi hàng trong bảng có **một giá trị duy nhất** và **không NULL**.

```
CREATE TABLE Employees (
EmployeeID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
Age int,
PRIMARY KEY (EmployeeID)
);
```

```
CREATE TABLE Employees (
ID INT PRIMARY KEY,
LastName VARCHAR(50),
FirstName VARCHAR(50)
);
```

▼ FOREIGN KEY

Dùng để liên kết hai bảng với nhau. Nó đảm bảo rằng giá trị của một cột (hoặc tập hợp cột) trong bảng này phải khớp với giá trị của cột tương ứng trong một bảng khác.

```
CREATE TABLE Orders (
OrderID int NOT NULL,
OrderNumber int NOT NULL,
EmployeeID int,
PRIMARY KEY (OrderID),
FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

Trong SQL, việc tạo một ràng buộc FOREIGN KEY không tự động khiến cho cột đó trở thành NOT NULL. Điều này có nghĩa là bạn vẫn có thể có các giá trị NULL trong cột đó trừ khi bạn đặt ràng buộc NOT NULL riêng biệt. Nếu bạn muốn cột đó không cho phép giá trị NULL, bạn cần phải rõ ràng chỉ định điều này khi tạo bảng.

```
CREATE TABLE Departments (
DeptID INT PRIMARY KEY,
```

```
DeptName VARCHAR(50)
);
```

```
CREATE TABLE Employees (
ID INT PRIMARY KEY,
LastName VARCHAR(50),
FirstName VARCHAR(50),
DeptID INT NOT NULL,  -- Chỉ định rõ ràng rằng DeptID không
FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);
```

Trong ví dụ này:

- Bảng `Departments` có một cột `DeptID` là khóa chính.
- Bảng `Employees` có cột `DeptID` là khóa ngoại, tham chiếu đến `DeptID` trong bảng `Departments`. Đồng thời, cột `DeptID` được chỉ định là NOT NULL để đảm bảo rằng mỗi nhân viên phải thuộc một phòng ban hợp lệ.

▼ CHECK

Ràng buộc này đảm bảo rằng tất cả các giá trị trong một cột thỏa mãn một điều kiện cụ thể.

```
CREATE TABLE Employees (
ID INT PRIMARY KEY,
LastName VARCHAR(50),
FirstName VARCHAR(50),
Age INT,
CHECK (Age >= 18)
);
```

▼ DEFAULT

Ràng buộc này cung cấp một giá trị mặc định cho một cột khi không có giá trị nào được chỉ định.

```
CREATE TABLE Employees (
EmployeeID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
```

```
Age int DEFAULT 30
);
```

Trong bảng `Employees`, nếu không có giá trị nào được cung cấp cho cột `HireDate`, giá trị mặc định sẽ là ngày hiện tại (sử dụng hàm `GETDATE()`).

```
CREATE TABLE Employees (
ID INT PRIMARY KEY,
LastName VARCHAR(50),
FirstName VARCHAR(50),
HireDate DATE DEFAULT GETDATE()
);
```

```
CREATE TABLE Users (
UserID INT PRIMARY KEY AUTO_INCREMENT,
Username VARCHAR(255) NOT NULL,
Email VARCHAR(255) NOT NULL,
Active BOOLEAN DEFAULT TRUE
);
```

```
CREATE TABLE Users (
UserID INT PRIMARY KEY AUTO_INCREMENT,
Username VARCHAR(255) NOT NULL,
Email VARCHAR(255) NOT NULL,
Active BOOLEAN DEFAULT TRUE
);
```

▼ SỬ DỤNG RÀNG BUỘC

Các ràng buộc có thể được khai báo khi tạo bảng mới bằng câu lệnh `CREATE TABLE` hoặc có thể được thêm vào bảng hiện tại bằng câu lệnh `ALTER TABLE`.

- **Khai báo trong `CREATE TABLE`:**

```
CREATE TABLE Employees (
EmployeeID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
```

```
Age int,
CONSTRAINT PK_Employee PRIMARY KEY (EmployeeID),
CONSTRAINT CK_Employee_Age CHECK (Age >= 18)
);
```

- **Thêm vào bảng** `ALTER TABLE`:

```
ALTER TABLE Employees
ADD CONSTRAINT CK_Employee_Age CHECK (Age >= 18);
```

▼ NGOÀI LỀ

AUTO_INCREMENT là một tính năng trong SQL cho phép tự động tạo giá trị số liên tiếp cho một cột trong bảng. Giá trị này thường được sử dụng làm khóa chính cho bảng.

Cách sử dụng AUTO_INCREMENT:

Để sử dụng AUTO_INCREMENT, bạn cần khai báo thuộc tính này cho cột khi tạo bảng hoặc sửa đổi bảng hiện có.

Cú pháp:

- Khi tạo bảng:

```
CREATE TABLE [TABLE_NAME] (
[COLUMN_NAME] [DATA_TYPE] [CONSTRAINT] AUTO_INCREMENT,
...
);
```

- Khi sửa đổi bảng hiện có:

```
ALTER TABLE [TABLE_NAME]
MODIFY [COLUMN_NAME] [DATA_TYPE] AUTO_INCREMENT;
```

- Ví dụ

```
CREATE TABLE Customers (
CustomerID INT PRIMARY KEY AUTO_INCREMENT,
CustomerName VARCHAR(255) NOT NULL,
```

```
Email VARCHAR(255) NOT NULL
);
```

Trong ví dụ trên, cột `CustomerID` được khai báo với thuộc tính **AUTO_INCREMENT**. Khi INSERT dữ liệu mới vào bảng `Customers`, giá trị cho cột `CustomerID` sẽ được tự động tạo là một số nguyên liên tiếp, bắt đầu từ 1.

Lưu ý:

- **AUTO_INCREMENT** chỉ hoạt động với các kiểu dữ liệu số nguyên (như INT, BIGINT, SMALLINT).
- Giá trị bắt đầu mặc định cho **AUTO_INCREMENT** là 1. Bạn có thể thay đổi giá trị này bằng cách sử dụng lệnh `ALTER TABLE`.
- Nên sử dụng **AUTO_INCREMENT** làm khóa chính cho bảng vì nó đảm bảo tính duy nhất và tự động tạo giá trị cho mỗi hàng mới.

Lợi ích của AUTO_INCREMENT:

- Giảm thiểu lỗi do người dùng nhập sai giá trị cho khóa chính.
- Đảm bảo tính duy nhất cho mỗi hàng trong bảng.
- Dễ dàng quản lý dữ liệu.

Ví dụ minh họa:

Giả sử bạn có một bảng `Products` lưu trữ thông tin về sản phẩm. Bảng này có các cột sau:

- `ProductID`: Khóa chính, kiểu dữ liệu INT
- `ProductName`: Tên sản phẩm, kiểu dữ liệu VARCHAR(255)
- `UnitPrice`: Giá bán sản phẩm, kiểu dữ liệu DECIMAL(10,2)

Bạn có thể tạo bảng này bằng cách sử dụng lệnh SQL sau:

```
CREATE TABLE Products (
ProductID INT PRIMARY KEY AUTO_INCREMENT,
ProductName VARCHAR(255) NOT NULL,
UnitPrice DECIMAL(10,2) NOT NULL
);
```

Khi bạn INSERT dữ liệu mới vào bảng `Products`, giá trị cho cột `ProductID` sẽ được tự động tạo. Ví dụ:

```
INSERT INTO Products (ProductName, UnitPrice)
VALUES ('Laptop', 12000000),
('Phone', 8000000),
('Tablet', 5000000);
```

Kết quả:

ProductID	ProductName	UnitPrice
1	Laptop	12000000
2	Phone	8000000
3	Tablet	5000000

Như bạn có thể thấy, giá trị cho cột `ProductID` được tự động tạo là 1, 2, 3 cho các hàng mới được INSERT.