HIỂU BIẾT VỀ RÀNG BUỘC

RÀNG BUỘC (Constraint) là các quy tắc được áp dụng trên các cột dữ liệu của một bảng. Chúng được sử dụng để kiểm tra tính hợp lệ của dữ liệu đầu vào, đảm bảo tính chính xác, độ tin cậy và tính toàn vẹn của dữ liệu trong database.

Ví dụ:

- Bảng Công dân với trường CCCD (Căn cước công dân) trong Hệ thống quản lý dữ liệu dân cư: Không thể có sự trùng lặp số CCCD giữa hai công dân với nhau;
- Bảng Đặt phòng trong Hệ thống quản lý khách sạn: Không thể có Ngày trả phòng lại trước Ngày nhận phòng.
- …

MỘT SỐ LOẠI RÀNG BUỘC:

- **NOT NULL** : đảm bảo dữ liệu cột luôn khác NULL. Ví dụ: Một người được sinh ra trên đời, ai cũng có Họ, Tên, vì vậy việc nhập giá trị của trường này không thể NULL.
- **UNIQUE** : không cho pháp dữ liệu cột này giống nhau ở các hàng khác nhau.
- **PRIMARY KEY**: dữ liệu trong nó là sự kết hợp giữa NOT NULL và UNIQUE, ràng buộc này giúp tạo chỉ mục (index) để tìm kiếm trong bảng nhanh hơn. Một bảng chỉ có thể có MỘT khóa chính và trong bảng, khóa chính này có thể bao gồm một hoặc nhiều cột (trường).
- **FOREIGN KEY**: là một khóa ngoại dùng để liên kết hai bảng với nhau. KHÓA NGOẠI là một trường (hoặc tập hợp các trường) trong một bảng tham chiếu đến KHÓA CHÍNH trong một bảng khác. Bảng chứa khóa ngoại được gọi là bảng con và bảng chứa khóa ứng viên được gọi là bảng tham chiếu hoặc bảng cha.
- **CHECK** : ràng buộc này đảm bảo dữ liệu đã được kiểm theo một điều kiện bạn đặt ra khi chèn dữ liệu.
- **DEFAULT** : ràng buộc thiết lập giá trị mặc định cho cột, nếu chèn hàng mà không truyền giá trị cho cột này nó tự nhận giá trị mặc định
- **INDEX** : ràng buộc tạo chỉ mục cho bảng, mục đích để truy vấn dữ liệu nhanh hơn.

Một ví dụ:

```
CREATE TABLE khachhangmoi(
    ID        SERIAL PRIMARY KEY,
    HoTen     VARCHAR (255) NOT NULL,
    Tuoi      INTEGER CHECK (Tuoi >= 18),
    CMT       CHAR (12)  NOT NULL UNIQUE,
    Tinhthanh VARCHAR (255) DEFAULT 'HANOI',
```

)

ID kiểu dữ liệu số nguyên tự động tăng (SERIAL), là khóa chính (do ràng buộc PRIMARY KEY)

HoTen kiểu varchar, và không chấp nhận chứa NULL do ràng buộc NOT NULL

Tuoi kiểu số nguyên, mỗi khi cập nhật, chèn cột sẽ kiểm tra đảm bảo giá trị >= 18 do ràng buộc CHECK (Tuoi >=18)

CMT số chứng minh thư, kiểu char(12), nó không chấp nhận NULL (do dang buộc NOT NULL), **Tinhthanh**: Tên tỉnh thành. Khi chèn dữ liệu nếu không truyền giá trị cho cột này nó tự nhận giá trị mặc định là 'HANOI' do ràng buộc DEFAULT 'HANOI'

*** PHÂN TÍCH CHI TIẾT PRIMARY KEY VÀ FOREGIN KEY ***

1. Nguốc gốc sâu xa của việc chia tách tập dữ liệu lớn thành nhiều BẢNG.

Hãy tưởng tượng 1 ví dụ về Bảng Đơn hàng của 1 sàn thương mại điện tử:

OrderID	OrderNumber	LastName	FirstName	Age	Mobile
1	77895	Tran Thi Nam	Tran	20	0888111222
2	44678	Tran Thi Nam	Tran	20	0888111222
3	22456	Nguyen Trieu Dieu	Linh	23	0999111333
4	24562	Hoang Anh	Duong	30	0912444555
5	1234	Tran Thi Nam	Tran	20	0888111222
6	1567	Tran Thi Nam	Tran	20	0888111222
7	3578	Tran Thi Nam	Tran	20	0888111222
8	8979	Tran Thi Nam	Tran	20	0888111222
9	46645	Tran Thi Nam	Tran	20	0888111222
10	23332	Tran Thi Nam	Tran	20	0888111222

Vấn đề: 1 Khách hàng (ví dụ Trần Thị Nam Trần với các thông tin kèm theo như LastName, FirstName, Age, Mobile) được lưu trữ lặp lại nhiều lần trong Bảng (Dư thừa). Hơn nữa, khi khách này thay đổi Số điện thoại liên hệ, Hệ thống có thể phải cập nhật lại tất cả các bản ghi. Hãy tưởng tượng 1 Hệ thống lớn với 1 tỉ người dùng, lượng dữ liệu dư thừa là bao nhiêu?

Chú ý quan hệ: Một khách hàng có thể đặt Nhiều đơn hàng.

Giải pháp: Phân tách cấu trúc trên thành nhiều cấu trúc con (Trong kỹ thuật của SQL được gọi là Chuẩn hóa cơ sở dữ liệu, chúng ta sẽ ko đề cập trong khóa này).

Tách bảng trên thành 2 bảng con (Customers - Khách hàng: Lưu trữ thông tin cá nhân khách hàng) và (Order - Đơn hàng: Lưu trữ các thông tin Đơn hàng)

Bảng **Customers**

PersonID	LastName	FirstName	Age	Mobile
1	Hoang Anh	Duong	30	0912444555
2	Nguyen Trieu Dieu	Linh	23	0999111333
3	Tran Thi Nam	Tran	20	0888111222

Bảng **Order**

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Chúng ta hãy quan sát 2 bảng trên với Bảng ban đầu: Để trả lời câu hỏi "Cho biết **tên khách hàng** có lượng đặt mua nhiều nhất trên hệ thống" để hỗ trợ công việc chăm sóc khách hàng.

>>> Bảng Order lúc này không còn lưu Tên khách hàng nhưng có lưu Mã khách hàng (PersonID). Nhờ có mã này, chúng ta có thể tìm ra tên Khách hàng. **Trong trường hợp này, PersonID ở bảng Customers là Primary Key của nó nhưng lại là Foreign Key ở bảng Order.**

>>> Chú ý quan hệ giữa 2 bảng: 1 Customers có thể có – n Order (đây là dạng quan hệ 1 – n).

2. Ràng buộc FOREIGN KEY

Ràng buộc FOREIGN KEY được sử dụng để ngăn chặn các hành động phá hủy liên kết giữa các bảng.

Bảng chứa FOREIGN KEY được gọi là Bảng con, Bảng tham chiếu tới được gọi là Bảng cha.

- DỮ LIỆU được thêm mới vào bảng Order chứa cột PersonID: Điều gì sẽ xảy ra nếu giá trị PersonID chưa có trang bản Customers >>> Vi phạm ràng buộc >>> Dữ liệu chèn vào Bảng con phải tồn tại trong Bảng cha.
- DỮ LIỆU PersonID ở Bảng cha bị xóa đi >>> Điều gì sẽ xảy ra với dữ liệu PersonID ở Bảng con??? >>> Chúng ta phải xác định hàng động sẽ thực hiện trên Bảng con: Dữ liệu đó cũng

bị xóa đi (CASCADE) hoặc Dữ liệu đó vẫn được giữ nguyên (NO ACTION) hoặc Dữ liệu đó được thiết lập Giá trị mới theo một quy tắc nào đó.

Ví dụ: Một phóng viên viết rất nhiều Bài báo trên trang VnExpress. Giả sử vị phóng viên này bị kỉ luật và cho thôi việc, các bài báo của phóng viên này đã từng viết sẽ bị xóa đi hay vẫn giữ lại? Nếu giữ lại có cập nhật lại tên tác giả?

3. Kỹ thuật thiết lập PRIMARY KEY và FOREIGN KEY

Bảng Customers:

- - Trường hợp 01: PRIMARY KEY được thiết lập trực tiếp tại thuộc tính chỉ định. Cách này chỉ áp dụng khi BẢNG chỉ có 1 thuộc tính là KHÓA

```
CREATE TABLE customers(
        personID serial PRIMARY KEY,
        lastName varchar(30) NOT NULL,
        firstName varchar(30) NOT NULL,
        age integer CHECK(age > 16),
        mobile varchar(20)
)
```

- - Trường hợp 02: PRIMARY KEY được thiết lập sau khi khai báo hết các thuộc tính. Cách này có thể sử dụng thiết lập cho trường hợp KHÓA CHÍNH có nhiều thuộc tính.

```
CREATE TABLE customers(
        personID serial,
        lastName varchar(30) NOT NULL,
        firstName varchar(30) NOT NULL,
        age integer CHECK(age > 16),
        mobile varchar(20),
        PRIMARY KEY(personID)
)
```

- - Trường hợp 03: PRIMARY KEY được thiết lập sau khi khai báo hết các thuộc tính. Cách này có thể sử dụng thiết lập cho trường hợp KHÓA CHÍNH có nhiều thuộc tính. Giả định KHÓA CHÍNH có thêm 1 thuộc tính nữa là example.

```
CREATE TABLE customers(
        personID serial,
        lastName varchar(30) NOT NULL,
        firstName varchar(30) NOT NULL,
```

```
        age integer CHECK(age > 16),
        mobile varchar(20),
        example varchar(10),
        PRIMARY KEY(personID, example)
)
```

- - Trường hợp 04: PRIMARY KEY được thiết lập sau khi Bảng đã được tạo >>> Chúng ta phải sửa đổi Bảng.

```
CREATE TABLE customers(
        personID serial,
        lastName varchar(30) NOT NULL,
        firstName varchar(30) NOT NULL,
        age integer CHECK(age > 16),
        mobile varchar(20),
)

>>>
ALTER TABLE Customers
ADD PRIMARY KEY(personID)

hoặc

ALTER TABLE Customers
ADD CONSTRAINT PK_Customers PRIMARY KEY (personID);
```

- - Trường hợp 05: Xóa PRIMARY KEY khỏi Bảng đã được tạo >>> Chúng ta phải sửa đổi Bảng.

```
CREATE TABLE customers(
        personID serial,
        lastName varchar(30) NOT NULL,
        firstName varchar(30) NOT NULL,
        age integer CHECK(age > 16),
        mobile varchar(20),
)

>>>
ALTER TABLE Customers
```

DROP PRIMARY KEY

Bảng Orders: Bảng này vừa có khóa chính, vừa có khóa ngoại

```
- - Trường hợp 01: PRIMARY KEY giống như phần trên, chú ý cách khai báo FOREIGN KEY
CREATE TABLE Orders (
   OrderID serial,
   OrderNumber integer NOT NULL,
   PersonID serial,
   PRIMARY KEY (OrderID),
   FOREIGN KEY (PersonID) REFERENCES customers(PersonID)
);
```

```
-- INSERT:
Khi chèn dữ liệu vào bảng có KHÓA NGOẠI, phải đảm báo giá trị đã tồn tại ở
-- [ON DELETE delete_action]
        SET NULL
        SET DEFAULT
        RESTRICT
        NO ACTION
        CASCADE

-- [ON UPDATE update_action]
        SET NULL
        SET DEFAULT
        RESTRICT
        NO ACTION
        CASCADE
Ví dụ:
ALTER TABLE orders
ADD CONSTRAINT orders_personid_fkey FOREIGN KEY(personID) REFERENCES
customers(personID)
ON DELETE CASCADE;
```

```
- - Trường hợp 03: Lưu và đặt tên ràng buộc
CREATE TABLE Orders (
   OrderID serial,
```

```
    OrderNumber integer NOT NULL,
    PersonID serial,
    PRIMARY KEY (OrderID),
 CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

```
- - Trường hợp 04: Thêm khóa ngoại khi Bảng đã tạo và cần sửa đổi
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

```
- - Trường hợp 04: Thêm khóa ngoại khi Bảng đã tạo và cần sửa đổi, cần lưu lại tên Ràng buộc
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

```
- - Trường hợp 05: Xóa ràng buộc
ALTER TABLE Orders
DROP CONSTRAINT FK_PersonOrder;
```

***** PHÂN TÍCH CHI TIẾT CÁC LOẠI RÀNG BUỘC CÒN LẠI *****

Ràng buộc NOT NULL

Theo mặc định, một cột có thể chứa các giá trị NULL. Ràng buộc NOT NULL thực thi cột KHÔNG chấp nhận giá trị NULL.

Điều này buộc một trường luôn chứa giá trị, có nghĩa là bạn không thể chèn bản ghi mới hoặc cập nhật bản ghi mà không thêm giá trị vào trường này.

```
CREATE TABLE Persons (
    ID integer NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age integer
);
```

>>> SQL sau đảm bảo rằng các cột “ID”, “LastName” và “FirstName” sẽ KHÔNG chấp nhận giá trị NULL khi bảng “Persons” được tạo:

Để tạo ràng buộc KHÔNG ĐẦY ĐỦ trên cột “Age” khi bảng “Persons” đã được tạo, hãy sử dụng SQL sau:

```
ALTER TABLE Persons
ALTER COLUMN Age SET NOT NULL;
```

Ràng buộc duy nhất (UNIQUE)

Ràng buộc UNIQUE đảm bảo rằng tất cả các giá trị trong một cột là khác nhau.

Cả ràng buộc UNIQUE và PRIMARY KEY đều đảm bảo tính duy nhất cho một cột hoặc tập hợp các cột.
Ràng buộc PRIMARY KEY tự động có một ràng buộc UNIQUE constraint.
Tuy nhiên, bạn có thể có nhiều ràng buộc DUY NHẤT(UNIQUE constraints) trên mỗi bảng, nhưng chỉ có một ràng buộc KEY CHÍNH(PRIMARY KEY) cho mỗi bảng.

```
CREATE TABLE Persons (
    ID integer NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age integer
);
```

>>> SQL tạo ràng buộc DUY NHẤT(UNIQUE) trên cột “ID” khi bảng “Persons” được tạo:

Để đặt tên cho một ràng buộc UNIQUE và để xác định một ràng buộc UNIQUE trên nhiều cột, hãy sử dụng cú pháp SQL sau:

```
CREATE TABLE Persons (
    ID integer NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age integer,
```

```
    CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```

Để tạo ràng buộc DUY NHẤT trên cột "ID" khi bảng đã được tạo, hãy sử dụng SQL sau:

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

Để đặt tên cho một ràng buộc UNIQUE và để xác định một ràng buộc UNIQUE trên nhiều cột, hãy sử dụng cú pháp SQL sau:

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

Để loại bỏ một ràng buộc UNIQUE, hãy sử dụng SQL sau:

```
ALTER TABLE Persons
DROP CONSTRAINT UC_Person;
```

Ràng buộc kiểm tra (CHECK)

Ràng buộc KIỂM TRA (CHECK Constraint) là 1 loại ràng buộc cho phép bạn chỉ định nếu 1 giá trị trong cột (column) phải đáp ứng 1 yêu cầu cụ thể. Ràng buộc CHECK sử dụng biểu thức Boolean để đánh giá các giá trị trước khi thực hiện Inserrt hoặc Update vào 1 bản ghi trong CSDL. Nếu các giá trị vượt qua điều kiện kiểm tra, PostgreSQL sẽ chèn hoặc cập nhật các giá trị này vào vào.

```
- - Định nghĩa rằng buộc kiểm tra khi tạo mới bảng CSDL

CREATE TABLE <table_name> (
<column_name> <data_type> CHECK (<check_condition>)
);

Hoặc chỉ định 1 tên cho rằng buộc kiểm tra

CREATE TABLE <table_name> (
<column_name> <data_type> CONSTRAINT <constraint_name> CHECK (<check_condition>)
```

);

Ví dụ: Tạo bảng employees có trường Salary có rằng buộc kiểm tra là salary > 0

```
CREATE TABLE employees (
    employee_id serial PRIMARY KEY,
    fullname character varying,
    salary numeric CHECK(salary > 0)
  );
```

- - Định nghĩa rằng buộc kiểm tra khi cho bảng CSDL đã tồn tại

Chúng ta sử dụng lệnh SQL: ALTER TABLE với action ADD CONSTRAINT

```
ALTER TABLE <table_name> ADD CONSTRAINT <constraint_name> CHECK (
<check_condition>
);
```

```
ALTER TABLE product ADD CONSTRAINT qty_price_check CHECK (qty > 0 AND price >
100000
  );
```