KHỞI TẠO VÀ SỬA ĐỔI DỮ LIỆU

1) TẠO CƠ SỞ DỮ LIỆU

- - Câu lệnh CREATE DATABASE được sử dụng để tạo cơ sở dữ liệu SQL mới.

CREATE DATABASE databasename;

Mẹo: Đảm bảo bạn có đặc quyền quản trị trước khi tạo bất kỳ cơ sở dữ liệu nào. Sau khi cơ sở dữ liệu được tạo, bạn có thể kiểm tra cơ sở dữ liệu đó trong danh sách cơ sở dữ liệu bằng lệnh SQL sau: \l;

2) XÓA CƠ SỞ DỮ LIỆU

- - Câu lệnh DROP DATABASE được sử dụng để xoá cơ sở dữ liệu SQL hiện có.

DROP DATABASE databasename;

Lưu ý: Hãy cẩn thận trước khi xoá cơ sở dữ liệu. Xóa cơ sở dữ liệu sẽ làm mất thông tin đầy đủ được lưu trữ trong cơ sở dữ liệu!

3) TẠO BẢNG

- - Câu lệnh CREATE TABLE được sử dụng để tạo một bảng mới trong cơ sở dữ liệu.

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Các tham số cột(column) chỉ định tên của các cột trong bảng.
Tham số kiểu dữ liệu(datatype) chỉ định loại dữ liệu mà cột có thể giữ (ví dụ: varchar, số nguyên, ngày, v.v.).

Ví dụ sau tạo một bảng có tên “Persons” có năm cột: PersonID, LastName, FirstName, Address và City:

```
CREATE TABLE Persons (
    PersonID serial,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

- - TẠO BẢNG MỚI TỪ MỘT BẢNG KHÁC ĐÃ CÓ DỮ LIỆU: THỰC CHẤT LÀ KẾT HỢP VỚI CÂU LỆNH TRUY VẤN LẤY DỮ LIỆU SẼ ĐƯỢC GIỚI THIỆU CHI TIẾT Ở BÀI SAU

```
CREATE TABLE NEW_TABLE_NAME AS
SELECT COLUMN1, COLUMN2,...
FROM EXISTING_TABLE_NAME
WHERE ....;
```

VÍ DỤ:

```
CREATE TABLE TESTTABLE AS
SELECT CUSTOMERNAME, CONTACTNAME
FROM CUSTOMERS;
```

4) XÓA BẢNG

- - Câu lệnh DROP TABLE được sử dụng để thả một bảng hiện có trong cơ sở dữ liệu.

```
DROP TABLE table_name;
```

- - Câu lệnh TRUNCATE TABLE được sử dụng để xóa dữ liệu bên trong bảng, nhưng không xóa chính bảng đó.

```
TRUNCATE TABLE table_name;
```

5) SỬA ĐỔI BẢNG

Lưu ý: Công việc thiết kế CSDL nói chung và thiết kế Bảng nói riêng có thể thiếu tính chính xác từ lúc đầu do người Thiết kế thiếu thông tin hoặc thiếu kinh nghiệm, cũng có thể do sự thay đổi yêu cầu từ Khách hàng. Có rất nhiều tình huống khác nhau cần xử lý, đặc biệt là khi chúng ta thao tác trên MỘT BẢNG ĐÃ CÓ SẴN DỮ LIỆU.

- - Câu lệnh ALTER TABLE được sử dụng để thêm, xóa hoặc sửa đổi các cột trong bảng hiện có.
- - Câu lệnh ALTER TABLE cũng được sử dụng để thêm và bỏ các ràng buộc khác nhau trên một bảng hiện có.

**** ALTER TABLE – ADD Column: Thêm một cột mới**

ALTER TABLE table_name
ADD column_name datatype;

Ví dụ: Thêm mới cột Email vào bảng Customers

ALTER TABLE Customers
ADD Email varchar(255);

Bạn cũng có thể xác định các ràng buộc trên cột cùng một lúc, sử dụng cú pháp thông thường:

ALTER TABLE products **ADD COLUMN** description text CHECK (description <> '');

Trên thực tế, tất cả các tùy chọn có thể được áp dụng cho mô tả cột trong TẠO BẢNG có thể được sử dụng ở đây. Tuy nhiên, hãy nhớ rằng giá trị mặc định phải thỏa mãn các ràng buộc đã cho, nếu không việc **ADD** sẽ không thành công. Ngoài ra, bạn có thể thêm các ràng buộc sau (xem bên dưới) sau khi đã điền chính xác cột mới.

Mẹo: Việc thêm cột với giá trị mặc định yêu cầu cập nhật từng hàng của bảng (để lưu trữ giá trị cột mới). Tuy nhiên, nếu không có mặc định nào được chỉ định, thì PostgreSQL có thể tránh cập nhật vật lý. Vì vậy, nếu bạn có ý định điền vào cột với hầu hết các giá trị không mặc định, tốt nhất là thêm cột không có mặc định, chèn các giá trị chính xác bằng cách sử dụng UPDATE, sau đó thêm bất kỳ giá trị mặc định mong muốn nào như được mô tả bên dưới.

**** ALTER TABLE – DROP COLUMN: Xóa bỏ một cột**

ALTER TABLE table_name **DROP COLUMN** column_name;

Ví dụ:

ALTER TABLE products **DROP COLUMN** description CASCADE;

Bất kỳ dữ liệu nào trong cột sẽ biến mất. Các ràng buộc bảng liên quan đến cột cũng bị loại bỏ. Tuy nhiên, nếu cột được tham chiếu bởi một ràng buộc khóa ngoại của một bảng khác, PostgreSQL sẽ không âm thầm loại bỏ ràng buộc đó. Bạn có thể cho phép bỏ mọi thứ phụ thuộc vào cột bằng cách thêm CASCADE:

**** SỬA ĐỔI CỘT - THÊM RÀNG BUỘC**

Để thêm một ràng buộc, cú pháp ràng buộc bảng được sử dụng. Ví dụ:

- - Thêm ràng buộc CHECK

ALTER TABLE products **ADD** CHECK (name <> '');

– Thêm ràng buộc UNIQUE, có đặt tên cụ thể cho Ràng buộc

ALTER TABLE products **ADD** CONSTRAINT some_name UNIQUE (product_no);

– Thêm ràng buộc Khóa ngoại

ALTER TABLE products **ADD** FOREIGN KEY (product_group_id) REFERENCES product_groups;

Để thêm một ràng buộc NOT NULL, không thể được viết dưới dạng một ràng buộc bảng, hãy sử dụng cú pháp sau:

ALTER TABLE products **ALTER COLUMN** product_no SET NOT NULL;

**** SỬA ĐỔI CỘT - XÓA RÀNG BUỘC**

Để loại bỏ một ràng buộc, bạn cần biết tên của nó. Nếu bạn đã đặt cho nó một cái tên thì điều đó thật dễ dàng. Nếu không, hệ thống đã gán một tên đã tạo, bạn cần tìm hiểu. Lệnh psql \d tên bảng có thể hữu ích ở đây; các giao diện khác cũng có thể cung cấp một cách để kiểm tra chi tiết bảng. Sau đó, lệnh là:

ALTER TABLE products **DROP CONSTRAINT** some_name;

Giống như việc xóa một cột, bạn cần thêm CASCADE nếu bạn muốn bỏ một ràng buộc mà thứ khác phụ thuộc vào. Một ví dụ là ràng buộc khóa ngoại phụ thuộc vào ràng buộc khóa chính hoặc ràng buộc Unique trên (các) cột được tham chiếu. Điều này hoạt động giống nhau đối với tất cả các loại ràng buộc ngoại trừ các ràng buộc NOT NULL. Để loại bỏ một ràng buộc NOT NULL, hãy sử dụng:

ALTER TABLE products **ALTER COLUMN** product_no **DROP** NOT NULL;

**** THAY ĐỔI GIÁ TRỊ MẶC ĐỊNH CỦA CỘT**

Để đặt mặc định mới cho một cột, hãy sử dụng lệnh như:

ALTER TABLE products **ALTER COLUMN** price **SET DEFAULT** 7.77;

Lưu ý rằng điều này không ảnh hưởng đến bất kỳ hàng hiện có nào trong bảng, nó chỉ thay đổi giá trị mặc định cho các lệnh INSERT trong tương lai. Để loại bỏ bất kỳ giá trị mặc định nào, hãy sử dụng:

ALTER TABLE products **ALTER COLUMN** price **DROP DEFAULT**;

Điều này thực sự giống như đặt giá trị mặc định thành null. Do đó, không phải là lỗi khi bỏ mặc định ở nơi chưa được xác định, bởi vì giá trị mặc định hoàn toàn là giá trị null.

**** THAY ĐỔI KIỂU DỮ LIỆU CỦA CỘT**

Để chuyển đổi một cột sang một kiểu dữ liệu khác, hãy sử dụng lệnh như:

ALTER TABLE products **ALTER COLUMN** price **TYPE** numeric(10,2);

Điều này sẽ chỉ thành công nếu mỗi mục nhập hiện có trong cột có thể được chuyển đổi thành kiểu mới bằng một phép ép kiểu ngầm định. Nếu cần chuyển đổi phức tạp hơn, bạn có thể thêm mệnh đề USING chỉ định cách tính giá trị mới từ giá trị cũ.

PostgreSQL sẽ cố gắng chuyển đổi giá trị mặc định của cột (nếu có) sang kiểu mới, cũng như bất kỳ ràng buộc nào liên quan đến cột. Nhưng những chuyển đổi này có thể không thành công hoặc có thể

tạo ra kết quả đáng ngạc nhiên. Tốt nhất bạn nên loại bỏ bất kỳ ràng buộc nào trên cột trước khi thay đổi kiểu của nó và sau đó thêm lại các ràng buộc được sửa đổi phù hợp sau đó.

**** ĐỔI TÊN CỘT**

ALTER TABLE products **RENAME COLUMN** product_no **TO** product_number;

**** ĐỔI TÊN BẢNG**

ALTER TABLE products **RENAME TO** items;

6) CHÈN DỮ LIỆU: INSERT INTO

Khi một bảng được tạo, nó không chứa dữ liệu. Điều đầu tiên cần làm trước khi một cơ sở dữ liệu có thể được sử dụng nhiều là chèn dữ liệu. Dữ liệu được chèn vào từng hàng một. Tất nhiên bạn cũng có thể chèn nhiều hơn một hàng, nhưng không có cách nào để chèn ít hơn một hàng. Ngay cả khi bạn chỉ biết một số giá trị cột, một hàng hoàn chỉnh phải được tạo.

Để tạo một hàng mới, hãy sử dụng lệnh INSERT. Lệnh yêu cầu tên bảng và các giá trị cột. Ví dụ với bảng đã được tạo Products:

```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric
);
```

Một lệnh ví dụ để chèn một hàng sẽ là:

```
INSERT INTO products(product_no, name, price) VALUES (1, 'Cheese', 9.99);
```

Trong trường hợp số lượng giá trị chèn vào = số lượng cột và thứ tự cột không thay đổi, bạn có thể sử dụng cú pháp rút gọn:

```
INSERT INTO products VALUES (1, 'Cheese', 9.99);
```

Các giá trị dữ liệu được liệt kê theo thứ tự các cột xuất hiện trong bảng, được phân tách bằng dấu phẩy. Thông thường, các giá trị dữ liệu sẽ là các chữ (hằng số), nhưng các biểu thức vô hướng cũng được phép.

Cú pháp trên có điểm hạn chế là bạn cần biết thứ tự của các cột trong bảng. Để tránh điều này, bạn cũng có thể liệt kê các cột một cách rõ ràng. Ví dụ: cả hai lệnh sau đều có cùng tác dụng như lệnh trên:

```
INSERT INTO products (product_no, name, price) VALUES (1, 'Cheese', 9.99);
INSERT INTO products (name, price, product_no) VALUES ('Cheese', 9.99, 1);
```

Nhiều người dùng coi việc luôn liệt kê tên cột là một phương pháp hay. Nếu bạn không có giá trị cho tất cả các cột, bạn có thể bỏ qua một số trong số chúng. Trong trường hợp đó, các cột sẽ được lấp đầy bằng các giá trị mặc định của chúng. Ví dụ:

```
INSERT INTO products (product_no, name) VALUES (1, 'Cheese');
INSERT INTO products VALUES (1, 'Cheese');
```

Dạng thứ hai là một phần mở rộng PostgreSQL. Nó lấp đầy các cột từ bên trái với nhiều giá trị nhất định và phần còn lại sẽ được mặc định.

Để rõ ràng, bạn cũng có thể yêu cầu các giá trị mặc định một cách rõ ràng, cho các cột riêng lẻ hoặc cho toàn bộ hàng:

```
INSERT INTO products (product_no, name, price) VALUES (1, 'Cheese', DEFAULT);
INSERT INTO products DEFAULT VALUES;
```

Bạn có thể chèn nhiều hàng trong một lệnh duy nhất:

```
INSERT INTO products (product_no, name, price) VALUES
    (1, 'Cheese', 9.99),
    (2, 'Bread', 1.99),
    (3, 'Milk', 2.99);
```

Mẹo: Khi chèn nhiều dữ liệu cùng lúc, hãy cân nhắc sử dụng lệnh COPY. Nó không linh hoạt như lệnh INSERT, nhưng hiệu quả hơn.

7) CẬP NHẬT DỮ LIỆU: UPDATE

Việc sửa đổi dữ liệu đã có trong cơ sở dữ liệu được gọi là cập nhật. Bạn có thể cập nhật các hàng riêng lẻ, tất cả các hàng trong bảng hoặc một tập hợp con của tất cả các hàng. Mỗi cột có thể được cập nhật riêng biệt; các cột khác không bị ảnh hưởng.

Để cập nhật các hàng hiện có, hãy sử dụng lệnh UPDATE. Điều này yêu cầu ba phần thông tin:

Tên của bảng và cột cần cập nhật
Giá trị mới của cột
(Các) hàng cần cập nhật

Hãy nhớ lại rằng SQL nói chung không cung cấp một mã định danh duy nhất cho các hàng. Do đó không phải lúc nào cũng có thể chỉ định trực tiếp hàng nào cần cập nhật. Thay vào đó, bạn chỉ định các điều kiện mà hàng phải đáp ứng để được cập nhật. Chỉ khi bạn có khóa chính trong bảng (không phụ thuộc vào việc bạn đã khai báo hay không) thì bạn mới có thể xác định địa chỉ các hàng riêng lẻ một cách đáng tin cậy bằng cách chọn một điều kiện phù hợp với khóa chính. Các công cụ truy cập cơ sở dữ liệu đồ họa dựa trên thực tế này để cho phép bạn cập nhật các hàng riêng lẻ.

Ví dụ: lệnh này cập nhật tất cả các sản phẩm có giá là 5 để có giá là 10:

```
UPDATE products SET price = 10 WHERE price = 5;
```

Điều này có thể khiến không, một hoặc nhiều hàng được cập nhật. Nó không phải là một lỗi khi cố gắng cập nhật không khớp với bất kỳ hàng nào. Hãy xem xét lệnh đó một cách chi tiết.
Đầu tiên là từ khóa UPDATE sau đó là tên bảng. Tiếp theo là từ khóa SET, theo sau là tên cột, dấu bằng và giá trị cột mới. Giá trị cột mới có thể là bất kỳ biểu thức vô hướng nào, không chỉ là một hằng số. Ví dụ: nếu bạn muốn tăng giá của tất cả các sản phẩm lên 10%, bạn có thể sử dụng:

```
UPDATE products SET price = price * 1.10;
```

Như bạn thấy, biểu thức cho giá trị mới có thể tham chiếu đến (các) giá trị hiện có trong hàng. Chúng ta cũng bỏ qua mệnh đề WHERE. Nếu nó bị bỏ qua, điều đó có nghĩa là tất cả các hàng trong bảng đều được cập nhật. Nếu nó hiện diện, chỉ những hàng phù hợp với điều kiện WHERE mới được cập nhật. Lưu ý rằng dấu bằng trong mệnh đề SET là một phép gán trong khi dấu trong mệnh đề WHERE là một phép so sánh, nhưng điều này không tạo ra bất kỳ sự mơ hồ nào.

Bạn có thể cập nhật nhiều hơn một cột trong lệnh UPDATE bằng cách liệt kê nhiều hơn một phép gán trong mệnh đề SET. Ví dụ:

UPDATE mytable **SET** a = 5, b = 3, c = 1 **WHERE** a > 0;

8) XÓA DỮ LIỆU TRONG BẢNG: DELETE

Cho đến nay chúng tôi đã giải thích cách thêm dữ liệu vào bảng và cách thay đổi dữ liệu. Những gì còn lại là thảo luận về cách loại bỏ dữ liệu không còn cần thiết. Cũng giống như chỉ có thể thêm dữ liệu trong toàn bộ các hàng, bạn chỉ có thể xóa toàn bộ các hàng khỏi bảng.

Trong phần trước, chúng tôi đã giải thích rằng SQL không cung cấp cách để giải quyết trực tiếp các hàng riêng lẻ. Do đó, việc loại bỏ các hàng chỉ có thể được thực hiện bằng cách xác định các điều kiện mà các hàng cần loại bỏ phải phù hợp. Nếu bạn có khóa chính trong bảng thì bạn có thể chỉ định hàng chính xác. Nhưng bạn cũng có thể xóa các nhóm hàng phù hợp với một điều kiện hoặc bạn có thể xóa tất cả các hàng trong bảng cùng một lúc.

Bạn sử dụng lệnh DELETE để loại bỏ các hàng; cú pháp rất giống với lệnh UPDATE.

Ví dụ: để xóa tất cả các hàng khỏi bảng sản phẩm có giá là 10, hãy sử dụng:

DELETE FROM products **WHERE** price = 10;

Nếu bạn chỉ viết:

DELETE FROM products;

thì tất cả các hàng trong bảng sẽ bị xóa!