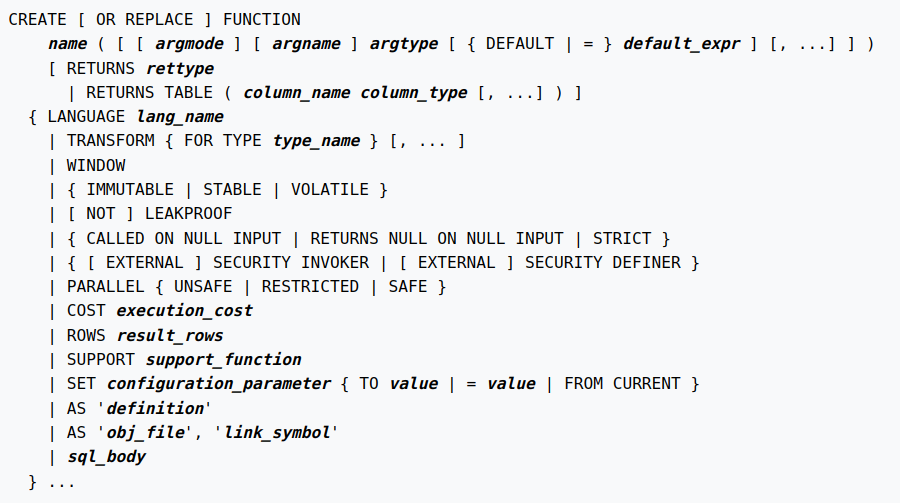
**TÓM TẮT CÚ PHÁP ĐỂ TẠO FUNCTION**

**MIÊU TẢ**

**CREATE** **FUNCTION**: dùng để tạo một function mới

**CREATE** **OR** **REPLACE** **FUNCTION**: dùng để tạo mới 1 function hoặc thay thế 1 function hiện có.

Nếu một tên lược đồ được bao gồm, thì hàm sẽ được tạo trong lược đồ đã chỉ định. Mặc khác thì nó sẽ tạo trong lược đồ hiện tại. Tên của hàm mới không được trùng với bất kỳ hàm hoặc thủ tục hiện có nào có cùng loại đối số đầu vào trong cùng một lược đồ. Tuy nhiên nếu các hàm và thủ tục của các loại đối số khác nhau có thể dùng chung một tên( overloading).

Sử dụng **CREATE** **OR** **REPLACE** **FUNCTION** để thay đổi định nghĩa hàm mà không phá vỡ các đối tượng tham chiếu đến hàm. Ngoài ra, **ALTER** **FUNCTION** có thể được sử dụng để thay đổi hầu hết các thuộc tính phụ trợ của một chức năng hiện có.

Các thông số:

***Name***: Tên (tùy chọn đủ điều kiện lược đồ) của hàm cần tạo.

***Argmode***: Chế độ của một đối số: **IN**, **OUT**, **INOUT** hoặc **VARIADIC**. Nếu bỏ qua, mặc định là **IN**. Chỉ các đối số **OUT** mới có thể tuân theo đối số **VARIADIC**. Ngoài ra, không thể sử dụng các đối số **OUT** và **INOUT** cùng với ký hiệu **RETURNS** **TABLE**.

***Argname***: Tên của một đối số. Tên của một đối số đầu ra là quan trọng, bởi vì nó xác định tên cột trong loại hàng kết quả. (Nếu bạn bỏ qua tên cho đối số đầu ra, hệ thống sẽ chọn tên cột mặc định).

***Argtype***: (Các ) loại dữ liệu của các đối số của hàm. Các loại đối số có thể là loại cơ sở, hỗn hợp hoặc miền hoặc có thể tham chiếu loại cột của bảng.

***Default\_expr***: Một biểu thức được sử dụng làm giá trị mặc định nếu tham số không được chỉ định. Biểu thức phải bắt buộc đối với loại đối số của tham số. Chỉ các tham số đầu vào (bao gồm cả **INOUT**) mới có thể có giá trị mặc định. Tất cả các tham số đầu vào theo sau một tham số có giá trị mặc định cũng phải có giá trị mặc định.

***Rettype***: Loại dữ liệu trả về. Loại trả về có thể là loại cơ sở, hỗn hợp hoặc miền hoặc có thể tham chiếu loại cột trong bảng. Nếu hàm không được phép trả về một giá trị, hãy chỉ định void làm kiểu trả về. Khi có tham số **OUT** hoặc **INOUT**, có thể bỏ qua mệnh đề **RETURNS**. Nếu có, nó phải phù hợp với loại kết quả được ngụ ý bởi các tham số đầu ra: **RECORD** nếu có nhiều tham số đầu ra hoặc cùng loại với tham số đầu ra duy nhất. Loại cột được tham chiếu bằng cách viết ***table\_name.column\_name%TYPE.***

***Column\_name***: Tên của một cột đầu ra trong cú pháp **RETURN** **TABLE**. Đây thực sự là một cách khác để khai báo tham số **OUT** có tên, ngoại trừ **RETURNS** **TABLE** cũng hàm ý **RETURNS** **SETOF**.

***Column\_type***: Kiểu dữ liệu của một cột đầu ra trong cú pháp **RETURNS** **TABLE**.

***Lang\_name***: Tên của ngôn ngữ mà hàm được triển khai.

**TRANSFORM { FOR TYPE type\_name} [,…]}**: Nên áp dụng các danh sách chuyển đổi lệnh gọi hàm. Biến đổi chuyển đổi giữa các loại SQL và các loại dữ liệu dành riêng cho ngôn ngữ; xem [**CREATE TRANSFORM**](https://www-postgresql-org.translate.goog/docs/15/sql-createtransform.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp) . Việc triển khai ngôn ngữ thủ tục thường có kiến thức được mã hóa cứng về các loại tích hợp sẵn, vì vậy những loại này không cần phải liệt kê ở đây. Nếu việc triển khai ngôn ngữ thủ tục không biết cách xử lý một loại và không có biến đổi nào được cung cấp, thì nó sẽ quay trở lại hành vi mặc định để chuyển đổi các loại dữ liệu, nhưng điều này phụ thuộc vào việc triển khai.

**WINDOW**: **WINDOW** chỉ ra rằng chức năng này là một chức năng cửa sổ chứ không phải là một chức năng đơn giản. Điều này hiện chỉ hữu ích cho các hàm được viết bằng C. Không thể thay đổi thuộc tính **WINDOW** khi thay thế một định nghĩa hàm hiện có.

**IMMUTABLE**

**STABLE**

**VOLATILE**

Các thuộc tính này thông báo cho trình tối ưu hóa truy vấn hành vi của mình. Có nhiều nhất một lựa chọn được chỉ định. Nếu không có cái nào trong số này được xuất hiện, VOLATILE sẽ xuất hiện.

**IMMUTABLE:** chỉ ra rằng hàm không thể sửa đổi cơ sở dữ liệu và luôn trả về cùng một kết quả khi được cung cấp cùng các giá trị đối số; nghĩa là, nó không thực hiện tra cứu cơ sở dữ liệu hoặc sử dụng thông tin không trực tiếp có trong danh sách đối số của nó. Nếu tùy chọn này được cung cấp, bất kỳ lệnh gọi hàm nào có đối số toàn hằng có thể được thay thế ngay lập tức bằng giá trị hàm.

**STABLE:** chỉ ra rằng hàm không thể sửa đổi cơ sở dữ liệu và trong một lần quét bảng, hàm sẽ luôn trả về cùng một kết quả cho cùng các giá trị đối số, nhưng kết quả của nó có thể thay đổi qua các câu lệnh SQL. Đây là lựa chọn phù hợp cho các hàm có kết quả phụ thuộc vào tra cứu cơ sở dữ liệu, các biến tham số (chẳng hạn như múi giờ hiện tại “***current time zone***”)

**VOLATILE:** chỉ ra rằng giá trị hàm có thể thay đổi ngay cả trong một lần quét bảng, do đó không thể thực hiện tối ưu hóa. Theo nghĩa này, tương đối ít chức năng cơ sở dữ liệu dễ bay hơi; một số ví dụ là ***random(), currval(), timeofday().*** Nhưng lưu ý rằng bất kỳ chức năng nào có tác dụng phụ phải được phân loại là không ổn định, ngay cả khi kết quả của nó khá dễ đoán, để ngăn chặn các cuộc gọi được tối ưu hóa; một ví dụ là setval().

**LEAKPROOF**: chỉ ra rằng hàm không có tác dụng phụ. Nó không tiết lộ thông tin nào về các đối số của nó ngoài giá trị trả về của nó. Hệ thống sẽ thực thi các điều kiện từ chính sách bảo mật và chế độ xem hàng rào bảo mật trước bất kỳ điều kiện nào do người dùng cung cấp từ chính truy vấn có chứa các chức năng không chống rò rỉ, nhằm ngăn chặn việc vô tình làm lộ dữ liệu. Các chức năng và toán tử được đánh dấu là chống rò rỉ được giả định là đáng tin cậy và có thể được thực thi trước các điều kiện từ chính sách bảo mật và chế độ xem hàng rào bảo mật. Ngoài ra, các hàm không nhận đối số hoặc không được truyền bất kỳ đối số nào từ chế độ xem hoặc bảng hàng rào bảo mật không cần phải đánh dấu là không bị rò rỉ để được thực thi trước các điều kiện bảo mật.

**CALLED ON NULL INPUT**

**RETURNS NULL ON NULL INPUT**

**STRICT**

**CALLED ON NULL INPUT**( **the default**): chỉ ra rằng hàm sẽ được gọi bình thường khi một đối số của nó là null. Sau đó, trách nhiệm của tác giả hàm là kiểm tra các giá trị null nếu cần và phản hồi một cách thích hợp.

**RETURNS NULL ON NULL INPUT** or **STRICT**: chỉ ra rằng hàm luôn trả về null bất cứ khi nào bất kỳ đối số nào của nó là null. Nếu tham số này được chỉ định, hàm sẽ không được thực thi khi có đối số null; thay vào đó, một kết quả null được giả định tự động.

**[EXTERNAL] SECURITY INVOKER**

**[EXTERNAL] SECURITY DEFINER**

**SECURITY INVOKER:** chỉ ra rằng chức năng sẽ được thực thi với các đặc quyền của người dùng gọi nó.

**SECURITY DEFINER:** chỉ định rằng chức năng sẽ được thực thi với các đặc quyền của người dùng sở hữu nó.

Từ khóa **EXTERNAL** được cho phép để tuân thủ SQL, nhưng nó là tùy chọn vì không giống như trong SQL, tính năng này áp dụng cho tất cả các chức năng không chỉ các chức năng bên ngoài.

**PARALLEL**

**PARALLEL UNSAFE** chỉ ra rằng chức năng không thể được thực thi ở chế độ song song và sự hiện diện của một chức năng như vậy trong câu lệnh SQL buộc một kế hoạch thực hiện nối tiếp

**PARALLEL RESTRICTED** chỉ ra rằng chức năng có thể được thực thi ở chế độ song song, nhưng việc thực thi bị hạn chế đối với trưởng nhóm song song.

**PARALLEL SAFE** chỉ ra rằng chức năng an toàn để chạy ở chế độ song song mà không bị hạn chế.

Các hàm nên được gắn nhãn không an toàn song song nếu chúng sửa đổi bất kỳ trạng thái cơ sở dữ liệu nào hoặc nếu chúng thực hiện các thay đổi đối với giao dịch, chẳng hạn như sử dụng giao dịch phụ hoặc nếu chúng truy cập các chuỗi hoặc cố gắng thực hiện các thay đổi liên tục đối với cài đặt (ví dụ: **setval**). Chúng phải được gắn nhãn là bị hạn chế song song nếu chúng truy cập các bảng tạm thời, trạng thái kết nối máy khách, con trỏ, câu lệnh đã chuẩn bị hoặc trạng thái cục bộ phụ trợ linh tinh mà hệ thống không thể đồng bộ hóa ở chế độ song song(ví dụ: **setseed** không thể được thực thi ngoài trưởng nhóm vì thay đổi được thực hiện bởi quy trình khác sẽ không được phản ánh trong trưởng nhóm).  
 Nói chung, nếu một chức năng được gắn nhãn là an toàn khi nó bị hạn chế hoặc không an toàn, hoặc nếu nó được gắn nhãn là bị hạn chế khi nó thực sự không an toàn, thì nó có thể đưa ra lỗi hoặc đưa ra câu trả lời sai khi được sử dụng trong một truy vấn song song.

**COST** ***execution\_cost***

Một số dương cho biết chi phí thực thi ước tính cho hàm, theo đơn vị ***cpu\_operator\_cost.*** Nếu hàm trả về một tập hợp, thì đây là chi phí cho mỗi hàng được trả về.

**ROW** ***result\_rows***

Một số dương cho biết số hàng ước tính mà người lập kế hoạch sẽ mong đợi hàm trả về. Điều này chỉ được phép khi hàm được khai báo trả về một tập hợp. Giả định mặc định là 1000 hàng.

**SUPPORT** ***support\_function***

Tên (tùy chọn đủ điều kiện cho lược đồ) của chức năng hỗ trợ trình lập kế hoạch để sử dụng cho chức năng này. Bạn phải là siêu người dùng để sử dụng tùy chọn này.

***configuration\_parameter***

***value***

Mệnh đề **SET** làm cho tham số cấu hình đã chỉ định được đặt thành giá trị đã chỉ định khi chức năng được nhập, sau đó được khôi phục về giá trị trước đó khi chức năng thoát. **SET FROM** **CURRENT** lưu giá trị của tham số hiện tại khi **CREATE FUNCTION** được thực thi dưới dạng giá trị sẽ được áp dụng khi hàm được nhập.

Nếu một mệnh đề **SET** được gắn vào một hàm, thì hiệu ứng của lệnh **SET** **LOCAL** được thực thi bên trong hàm đối với cùng một biến sẽ bị hạn chế đối với hàm: giá trị trước đó của tham số cấu hình vẫn được khôi phục khi thoát chức năng. Tuy nhiên, một lệnh **SET** thông thường (không có **LOCAL**) sẽ ghi đè mệnh đề **SET**, giống như lệnh **SET** **LOCAL** trước đó: hiệu ứng của lệnh đó sẽ tiếp tục sau khi thoát khỏi chức năng, trừ khi giao dịch hiện tại được khôi phục.

***Definition***

Một hằng chuỗi xác định hàm; ý nghĩa phụ thuộc vào ngôn ngữ. Nó có thể là tên hàm bên trong, đường dẫn đến tệp đối tượng, lệnh SQL hoặc văn bản bằng ngôn ngữ thủ tục.

Việc sử dụng trích dẫn đô la (**[xem Phần 4.1.2.4](https://www-postgresql-org.translate.goog/docs/15/sql-syntax-lexical.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp" \l "SQL-SYNTAX-DOLLAR-QUOTING)** ) để viết chuỗi định nghĩa hàm thường hữu ích, thay vì cú pháp trích dẫn đơn thông thường. Không có trích dẫn đô la, bất kỳ dấu ngoặc đơn hoặc dấu gạch chéo ngược nào trong định nghĩa hàm phải được thoát bằng cách nhân đôi chúng.

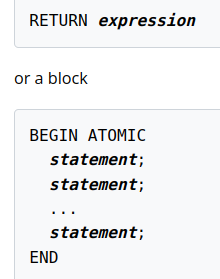
***obj\_file, link\_symbol***

Dạng mệnh đề **AS** này được sử dụng cho các hàm ngôn ngữ C có thể tải động khi tên hàm trong mã nguồn ngôn ngữ C không giống với tên của hàm SQL. Chuỗi **obj\_file** là tên của tệp thư viện chia sẻ chứa hàm C đã biên dịch và được hiểu là đối với lệnh **LOAD**. Chuỗi **link\_symbol** là ký hiệu liên kết của hàm, nghĩa là tên của hàm trong mã nguồn ngôn ngữ C. Nếu biểu tượng liên kết bị bỏ qua, nó được coi là giống với tên của hàm SQL được xác định. Tên C của tất cả các hàm phải khác nhau, vì vậy bạn phải đặt cho các hàm C quá tải các tên C khác nhau (ví dụ: sử dụng các loại đối số như một phần của tên C).

Khi các cuộc gọi **CREATE** **FUNCTION** lặp đi lặp lại đề cập đến cùng một tệp đối tượng, tệp chỉ được tải một lần cho mỗi phiên. Để dỡ và tải lại tệp (có thể trong quá trình phát triển), hãy bắt đầu một phiên mới.

sql\_body

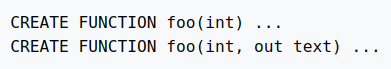
Phần thân của hàm LANGUAGE SQL. Đây có thể là một tuyên bố duy nhất

 Điều này tương tự như viết văn bản của thân hàm dưới dạng hằng chuỗi, nhưng có một số điểm khác biệt: Biểu mẫu này chỉ hoạt động đối với NGÔN NGỮ SQL, dạng hằng chuỗi hoạt động đối với tất cả các ngôn ngữ. Dạng này được phân tích cú pháp tại thời điểm định nghĩa hàm, dạng hằng chuỗi được phân tích cú pháp tại thời điểm thực thi; do đó, biểu mẫu này không thể hỗ trợ các loại đối số đa hình và các cấu trúc khác không thể giải quyết được tại thời điểm định nghĩa hàm. Biểu mẫu này theo dõi các phụ thuộc giữa hàm và các đối tượng được sử dụng trong thân hàm, do đó, DROP ... CASCADE sẽ hoạt động chính xác, trong khi biểu mẫu sử dụng chuỗi ký tự có thể để lại các hàm lơ lửng.

**OVERLOADING**

PostgreSQL cho phép nạp chồng hàm; nghĩa là, cùng một tên có thể được sử dụng cho một số chức năng khác nhau miễn là chúng có các loại đối số đầu vào riêng biệt. Cho dù bạn có sử dụng nó hay không, khả năng này đòi hỏi các biện pháp phòng ngừa bảo mật khi gọi các chức năng trong cơ sở dữ liệu nơi một số người dùng không tin tưởng những người dùng khác

Hai hàm được coi là giống nhau nếu chúng có cùng tên và kiểu đối số đầu vào, bỏ qua mọi tham số OUT. Vì vậy, ví dụ, những tuyên bố xung đột:



Các hàm có danh sách loại đối số khác nhau sẽ không được coi là xung đột tại thời điểm tạo, nhưng nếu các giá trị mặc định được cung cấp thì chúng có thể xung đột khi sử dụng. Ví dụ:

**GHI CHÚ**

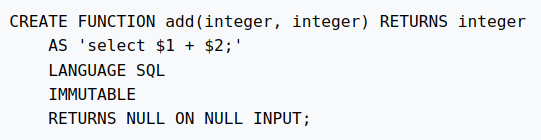
Cú pháp kiểu SQL đầy đủ được phép khai báo các đối số và giá trị trả về của hàm. Tuy nhiên, các công cụ sửa đổi loại được đặt trong ngoặc đơn (ví dụ: trường chính xác cho loại numeric) bị loại bỏ bởi CREATE FUNCTION. Vì vậy, ví dụ CREATE FUNCTION foo (varchar(10)) ...hoàn toàn giống với CREATE FUNCTION foo (varchar) .…

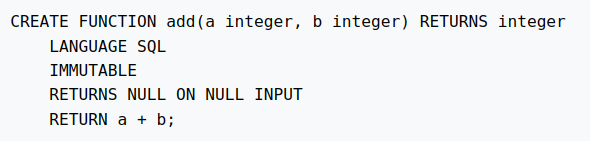
Khi thay thế một hàm hiện có bằng CREATE OR REPLACE FUNCTION, sẽ có những hạn chế về việc thay đổi tên tham số. Bạn không thể thay đổi tên đã được gán cho bất kỳ tham số đầu vào nào (mặc dù bạn có thể thêm tên cho các tham số chưa có trước đó). Nếu có nhiều hơn một tham số đầu ra, bạn không thể thay đổi tên của các tham số đầu ra, vì điều đó sẽ thay đổi tên cột của loại hỗn hợp ẩn danh mô tả kết quả của hàm. Những hạn chế này được thực hiện để đảm bảo rằng các lời gọi hàm hiện có không ngừng hoạt động khi nó được thay thế.

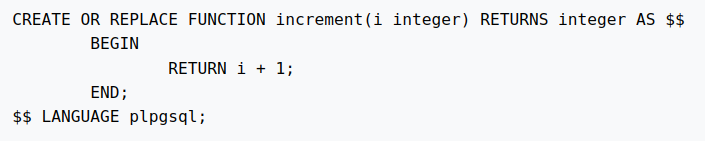
Nếu một hàm được khai báo STRICT với đối số VARIADIC, kiểm tra tính nghiêm ngặt sẽ kiểm tra xem toàn bộ mảng variadic có khác rỗng không. Hàm vẫn sẽ được gọi nếu mảng có các phần tử null.

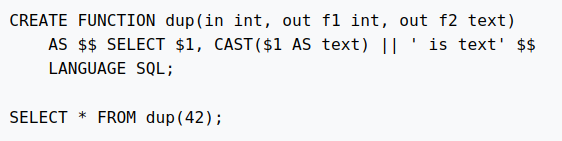
**VÍ DỤ**

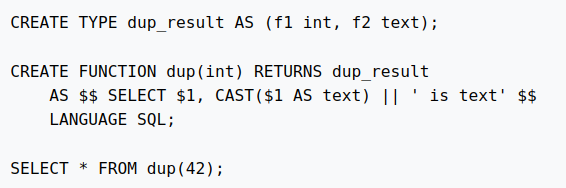
Cộng hai số nguyên bằng hàm SQL:

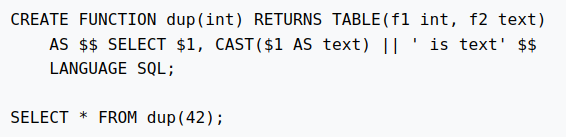
 Hàm tương tự được viết theo kiểu tuân thủ SQL hơn, sử dụng tên đối số và phần thân không được trích dẫn:

 Tăng một số nguyên, sử dụng tên đối số, trong PL/pgSQL:

 Trả về một bản ghi chứa nhiều tham số đầu ra:

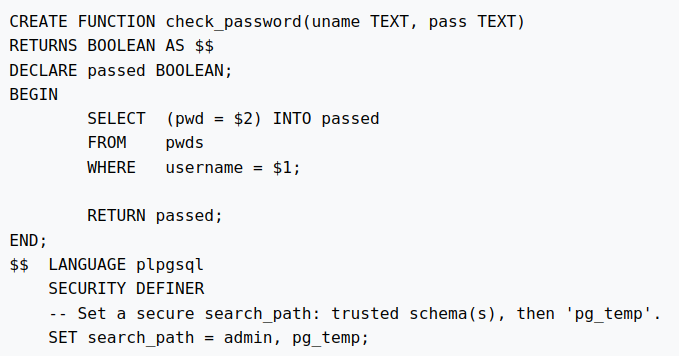
 Bạn có thể làm điều tương tự chi tiết hơn với một loại hỗn hợp được đặt tên rõ ràng:

 Một cách khác để trả về nhiều cột là sử dụng hàm TABLE:

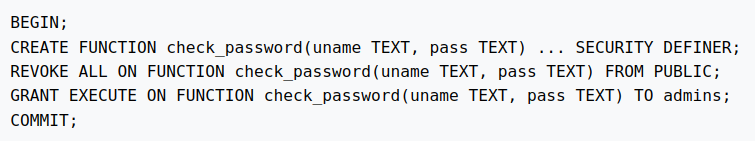
 Tuy nhiên, một hàm TABLE khác với các ví dụ trước, bởi vì nó thực sự trả về một tập hợp các bản ghi, không chỉ một bản ghi.

**WRITING SECURITY DEFINER FUNCTIONS SAFELY**

Vì hàm **SECURITY DEFINER** được thực thi với các đặc quyền của người dùng sở hữu nó, nên cần cẩn thận để đảm bảo rằng hàm này không thể bị lạm dụng. Để bảo mật, **[search\_path](https://www-postgresql-org.translate.goog/docs/15/runtime-config-client.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp" \l "GUC-SEARCH-PATH)** phải được đặt để loại trừ mọi lược đồ có thể ghi bởi người dùng không đáng tin cậy. Điều này ngăn những người dùng độc hại tạo các đối tượng (ví dụ: bảng, hàm và toán tử) che dấu các đối tượng mà hàm dự định sử dụng. Đặc biệt quan trọng về vấn đề này là lược đồ bảng tạm thời, lược đồ này được tìm kiếm đầu tiên theo mặc định và thường được ghi bởi bất kỳ ai. Có thể thu được một sự sắp xếp an toàn bằng cách buộc lược đồ tạm thời được tìm kiếm sau cùng. Để thực hiện việc này, hãy viết **pg\_temp** làm mục nhập cuối cùng trong **search\_path**. Hàm này minh họa cách sử dụng an toàn:

 Mục đích của hàm này là truy cập bảng admin.pwds. Nhưng nếu không có mệnh đề SET hoặc với mệnh đề SET chỉ đề cập đến quản trị viên, hàm này có thể bị hủy bỏ bằng cách tạo một bảng tạm thời có tên là pwds.

Một điểm khác cần lưu ý là theo mặc định, đặc quyền thực thi được cấp cho PUBLIC đối với các hàm mới được tạo. Thông thường, bạn sẽ muốn hạn chế việc sử dụng hàm xác định bảo mật chỉ cho một số người dùng. Để làm điều đó, bạn phải thu hồi các đặc quyền CÔNG CỘNG mặc định và sau đó cấp đặc quyền thực thi một cách có chọn lọc. Để tránh có một cửa sổ trong đó tất cả mọi người đều có thể truy cập chức năng mới, hãy tạo nó và đặt các đặc quyền trong một giao dịch. Ví dụ:



**COMPATIBILITY (KHẢ NĂNG TƯƠNG THÍCH)**

Lệnh **CREATE** **FUNCTION** được định nghĩa trong tiêu chuẩn SQL. Việc triển khai PostgreSQL có thể được sử dụng theo cách tương thích nhưng có nhiều phần mở rộng. Ngược lại, tiêu chuẩn SQL chỉ định một số tính năng tùy chọn không được triển khai trong PostgreSQL.

Sau đây là các vấn đề tương thích quan trọng:

* **OR** **REPLACE** là một phần mở rộng PostgreSQL.
* Để tương thích với một số hệ thống cơ sở dữ liệu khác, **argmode** có thể được viết trước hoặc sau **argname**. Nhưng chỉ có cách đầu tiên là tuân thủ tiêu chuẩn.
* Đối với các giá trị mặc định của tham số, tiêu chuẩn SQL chỉ xác định cú pháp với từ khóa **DEFAULT**. Cú pháp với = được sử dụng trong T-SQL và Firebird.
* Công cụ sửa đổi **SETOF** là một phần mở rộng PostgreSQL.
* Chỉ SQL được chuẩn hóa như một ngôn ngữ.
* Tất cả các thuộc tính khác ngoại trừ **CALLED** **ON** **NULL** **INPUT** và **RETURNS** **NULL** **ON** **NULL** **INPUT** đều không được chuẩn hóa.
* Đối với phần thân của các hàm **LANGUAGE** **SQL**, tiêu chuẩn SQL chỉ xác định biểu mẫu **sql\_body**.

**MỘT SỐ SYNTAX PHỔ BIẾN KHÁC**

[**ALTER FUNCTION**](https://www-postgresql-org.translate.goog/docs/15/sql-alterfunction.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp)

[**DROP FUNCTION**](https://www-postgresql-org.translate.goog/docs/15/sql-dropfunction.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp)

[**GRANT**](https://www-postgresql-org.translate.goog/docs/15/sql-grant.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp)

[**LOAD**](https://www-postgresql-org.translate.goog/docs/15/sql-load.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp)

[**REVOKE**](https://www-postgresql-org.translate.goog/docs/15/sql-revoke.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp)

**Link tiếng việt:**

[**https://www-postgresql-org.translate.goog/docs/15/sql-creategroup.html?\_x\_tr\_sl=en&\_x\_tr\_tl=vi&\_x\_tr\_hl=vi&\_x\_tr\_pto=wapp**](https://www-postgresql-org.translate.goog/docs/15/sql-creategroup.html?_x_tr_sl=en&_x_tr_tl=vi&_x_tr_hl=vi&_x_tr_pto=wapp)

**Link tiếng anh:**

[**https://www.postgresql.org/docs/15/sql-createfunction.html**](https://www.postgresql.org/docs/15/sql-createfunction.html)

**Link SQL:**

[**https://sqliteonline.com/**](https://sqliteonline.com/)

**Link function có csdl:**

[**https://www.postgresqltutorial.com/postgresql-plpgsql/postgresql-create-function/**](https://www.postgresqltutorial.com/postgresql-plpgsql/postgresql-create-function/)