# **Encapsulate data transformations in a stored procedure**

Completed100 XP

* 4 minutes

While you can run a CREATE EXTERNAL TABLE AS SELECT (CETAS) statement in a script whenever you need to transform data, it's good practice to encapsulate the transformation operation in stored procedure. This approach can make it easier to operationalize data transformations by enabling you to supply parameters, retrieve outputs, and include additional logic in a single procedure call.

For example, the following code creates a stored procedure that drops the external table if it already exists before recreating it with order data for the specified year:

SQLCopy

CREATE PROCEDURE usp\_special\_orders\_by\_year @order\_year INT  
AS  
BEGIN  
  
 -- Drop the table if it already exists  
 IF EXISTS (  
 SELECT \* FROM sys.external\_tables  
 WHERE name = 'SpecialOrders'  
 )  
 DROP EXTERNAL TABLE SpecialOrders  
  
 -- Create external table with special orders  
 -- from the specified year  
 CREATE EXTERNAL TABLE SpecialOrders  
 WITH (  
 LOCATION = 'special\_orders/',  
 DATA\_SOURCE = files,  
 FILE\_FORMAT = ParquetFormat  
 )  
 AS  
 SELECT OrderID, CustomerName, OrderTotal  
 FROM  
 OPENROWSET(  
 BULK 'sales\_orders/\*.csv',  
 DATA\_SOURCE = 'files',  
 FORMAT = 'CSV',  
 PARSER\_VERSION = '2.0',  
 HEADER\_ROW = TRUE  
 ) AS source\_data  
 WHERE OrderType = 'Special Order'  
 AND YEAR(OrderDate) = @order\_year  
END

**Note**

As discussed previously, dropping an existing external table does not delete the folder containing its data files. You must explicitly delete the target folder if it exists before running the stored procedure, or an error will occur.

In addition to encapsulating Transact-SQL logic, stored procedures also provide the following benefits:

## **Reduces client to server network traffic**

The commands in a procedure are executed as a single batch of code; which can significantly reduce network traffic between the server and client because only the call to execute the procedure is sent across the network.

## **Provides a security boundary**

Multiple users and client programs can perform operations on underlying database objects through a procedure, even if the users and programs don't have direct permissions on those underlying objects. The procedure controls what processes and activities are performed and protects the underlying database objects; eliminating the requirement to grant permissions at the individual object level and simplifies the security layers.

## **Eases maintenance**

Any changes in the logic or file system locations involved in the data transformation can be applied only to the stored procedure; without requiring updates to client applications or other calling functions.

## **Improved performance**

Stored procedures are compiled the first time they're executed, and the resulting execution plan is held in the cache and reused on subsequent runs of the same stored procedure. As a result, it takes less time to process the procedure.

## **đóng gói các chuyển đổi dữ liệu trong một thủ tục lưu trữ (stored procedure)**

Mặc dù bạn có thể chạy câu lệnh CREATE EXTERNAL TABLE AS SELECT (CETAS) trong một tập lệnh bất cứ khi nào bạn cần chuyển đổi dữ liệu, nhưng việc đóng gói hoạt động chuyển đổi trong thủ tục lưu trữ là một thực hành tốt. Cách tiếp cận này có thể giúp dễ dàng thực hiện các chuyển đổi dữ liệu hơn bằng cách cho phép bạn cung cấp tham số, truy xuất kết quả đầu ra và bao gồm logic bổ sung trong một lần gọi thủ tục.

Ví dụ: mã sau tạo một thủ tục lưu trữ cho phép xóa bảng ngoài nếu nó đã tồn tại trước khi tạo lại bảng đó với dữ liệu đơn hàng cho năm được chỉ định:

SQL

CREATE PROCEDURE usp\_special\_orders\_by\_year @order\_year INT  
AS  
BEGIN  
  
 -- Xóa bảng nếu nó đã tồn tại  
 IF EXISTS (  
 SELECT \* FROM sys.external\_tables  
 WHERE name = 'SpecialOrders'  
 )  
 DROP EXTERNAL TABLE SpecialOrders  
  
 -- Tạo bảng ngoài với các đơn hàng đặc biệt  
 -- từ năm được chỉ định  
 CREATE EXTERNAL TABLE SpecialOrders  
 WITH (  
 LOCATION = 'special\_orders/',  
 DATA\_SOURCE = files,  
 FILE\_FORMAT = ParquetFormat  
 )  
 AS  
 SELECT OrderID, CustomerName, OrderTotal  
 FROM  
 OPENROWSET(  
 BULK 'sales\_orders/\*.csv',  
 DATA\_SOURCE = 'files',  
 FORMAT = 'CSV',  
 PARSER\_VERSION = '2.0',  
 HEADER\_ROW = TRUE  
 ) AS source\_data  
 WHERE OrderType = 'Special Order'  
 AND YEAR(OrderDate) = @order\_year  
END

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

**Lưu ý**

Như đã thảo luận trước đây, việc xóa một bảng ngoài hiện có không xóa thư mục chứa các tệp dữ liệu của nó. Bạn phải xóa thư mục đích một cách rõ ràng nếu nó tồn tại trước khi chạy thủ tục lưu trữ, nếu không sẽ xảy ra lỗi.

Ngoài việc đóng gói logic Transact-SQL, các thủ tục lưu trữ còn cung cấp các lợi ích sau:

* Giảm lưu lượng truy cập mạng giữa máy khách và máy chủ Các lệnh trong một thủ tục được thực thi dưới dạng một lô mã duy nhất; điều này có thể làm giảm đáng kể lưu lượng mạng giữa máy chủ và máy khách vì chỉ có cuộc gọi để thực thi thủ tục được gửi qua mạng.
* Cung cấp ranh giới bảo mật Nhiều người dùng và chương trình khách có thể thực hiện các hoạt động trên các đối tượng cơ sở dữ liệu nền tảng thông qua một thủ tục, ngay cả khi người dùng và chương trình không có quyền trực tiếp trên các đối tượng nền tảng đó. Thủ tục kiểm soát các quy trình và hoạt động được thực hiện và bảo vệ các đối tượng cơ sở dữ liệu nền tảng; loại bỏ yêu cầu cấp quyền ở cấp độ đối tượng riêng lẻ và đơn giản hóa các lớp bảo mật.
* Dễ dàng bảo trì Bất kỳ thay đổi nào trong logic hoặc vị trí hệ thống tệp liên quan đến chuyển đổi dữ liệu đều có thể được áp dụng chỉ cho thủ tục lưu trữ; mà không cần cập nhật cho các ứng dụng khách hoặc các hàm gọi khác.
* Cải thiện hiệu suất Các thủ tục lưu trữ được biên dịch lần đầu tiên chúng được thực thi và lược đồ thực thi kết quả được giữ trong bộ nhớ cache và sử dụng lại trong các lần chạy tiếp theo của cùng một thủ tục lưu trữ. Do đó, việc xử lý thủ tục mất ít thời gian hơn.