SQL Server to Trino
Teradata to Trino

Migration to Netezza

Oracle to Netezza

Migration to Greenplum

IBM DB2 to Greenplum
Oracle to Greenplum

Migration to EsgynDB

Oracle to EsgynDB
Teradata to EsgynDB

Application Conversion

Java
C# .NET
PowerBuilder
COBOL

Database Reference

Oracle
SQL Server
IBM DB2
MariaDB
MySQL
PostgreSQL
Sybase
Sybase ASA
Informix
Teradata

Stored Procedures and Functions in PostgreSQL - Getting Started

A stored procedure and user-defined function (UDF) is a set of SQL and procedural statements (declarations, assignments, loops, flow-of-control etc.) that stored on the database server and can be invoked using the SQL interface.

Quick Example:

```
-- Function increments the input value by 1
CREATE OR REPLACE FUNCTION increment(i INT) RETURNS INT AS $$
 BEGIN
   RETURN i + 1;
 END;
 $$ LANGUAGE plpgsql;

 -- An example how to use the function (Returns: 11)
 SELECT increment(10);
```

In PostgreSQL, *both* stored procedures and user-defined functions are created with CREATE FUNCTION statement. There are differences between *the notion* of stored procedures and functions in database systems:

	Stored Procedure	Function
Use in an expression	✗	✓
Return a value	✗	✓

Return values as OUT parameters	✓	✗
Return a single result set	✓	✓ (as a table function)
Return multiple result sets	✓	✗

So in most cases, the purpose of a stored procedure is to:

- Perform actions without returning any result (INSERT, UPDATE operations i.e.)
- Return one or more scalar values as OUT parameters
- Return one or more result sets

Usually the purpose of a user-defined function is to process the input parameters and return a new value.

Reporting Tools

Many reporting tools (Crystal Reports, Reporting Services, BI tools etc.) allow you to specify a query (SQL SELECT statement) or a stored procedure returning a result set to define a data set for a report.

Stored procedures are very useful when you need to perform complex calculations before the data is available for a report.

Stored Procedures in PostgreSQL

Usually stored procedures do not return any value, or return one or more result sets.

No Value Returned

If a stored procedure does not return any value, you can specify *void* as the return type:

```
    -- Procedure to insert a new city
    CREATE OR REPLACE FUNCTION add_city(city VARCHAR(70), state CHAR(2))
    RETURNS void AS $$
    BEGIN
      INSERT INTO cities VALUES (city, state);
    END;
    $$ LANGUAGE plpgsql;
```

You can use SELECT statement to invoke the *add_city* procedure:

```
   -- Add a new city
   SELECT add_city('St.Louis', 'MO');
```

You can also use *PERFORM add_city()* statement to invoke *add_city* from another procedure or function.

Return a Single Result Set - Return a Cursor

To return a result set from a PostgreSQL procedure, you have to specify *refcursor* return type, open and return a cursor:

```
   CREATE OR REPLACE FUNCTION show_cities() RETURNS refcursor AS $$
    DECLARE
      ref refcursor;
    BEGIN
      OPEN ref FOR SELECT city, state FROM cities;
      RETURN ref;
    END;
    $$ LANGUAGE plpgsql;
```

Important Note: The cursor remains open until the **end** of transaction, and since PostgreSQL works in auto-commit mode by default, the cursor is closed **immediately** after the procedure call, so it is not available to the caller. To work with cursors you have to start a transaction (turn auto-commit off).

For more information, see PostgreSQL - How to Return a Result Set from a Stored Procedure

Resources