PostgreSQL - How to Return a Result Set from a Stored Procedure

Both stored procedures and user-defined functions are created with CREATE FUNCTION statement in PostgreSQL.

- PostgreSQL Stored Procedures and Functions - Getting Started

To return one or more result sets (cursors in terms of PostgreSQL), you have to use *refcursor* return type.

Quick Example:

```
   -- Procedure that returns a single result set (cursor)
   CREATE OR REPLACE FUNCTION show_cities() RETURNS refcursor AS $$
    DECLARE
      ref refcursor;                                                    -- Declare a cursor variable
    BEGIN
      OPEN ref FOR SELECT city, state FROM cities;   -- Open a cursor
      RETURN ref;                                                      -- Return the cursor to the caller
    END;
    $$ LANGUAGE plpgsql;
```

Overview:

Return Multiple Result Sets	✓	
Cursor Lifetime	Until the *end* of transaction	
Auto-commit	Must be **off**	Transaction must be active so the caller can see a result set

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 the caller have to start a transaction.

Returning Multiple Result Sets (Cursors)

To return multiple result sets, specify *SETOF refcursor* return type and use *RETURN NEXT* to return each cursor:

```
    -- Procedure that returns multiple result sets (cursors)
   CREATE OR REPLACE FUNCTION show_cities_multiple() RETURNS SETOF refcursor AS $$
    DECLARE
      ref1 refcursor;           -- Declare cursor variables
      ref2 refcursor;
    BEGIN
      OPEN ref1 FOR SELECT city, state FROM cities WHERE state = 'CA';   -- Open the first cursor
      RETURN NEXT ref1;                                                                  -- Return the cursor to the caller
      OPEN ref2 FOR SELECT city, state FROM cities WHERE state = 'TX';   -- Open the second cursor
      RETURN NEXT ref2;                                                                    -- Return the cursor to the caller
    END;
    $$ LANGUAGE plpgsql;
```

Processing the result sets and designing the procedures returning result sets may depend on the caller.

When Caller is PSQL, pgAdmin Query or Another Procedure/Function

Let's assume you need to call a procedure and output the result set in PSQL tool, pgAdmin Query tool or another function:

```
   SELECT show_cities();
```

The result:

show_cities refcursor
<unnamed portal 1>

The query returns the *name* of the cursor, it does *not* output the rows of the result set. To get the rows you need to use FETCH statement and specify the cursor name:

```
   FETCH ALL IN "<unnamed portal 1>";
   -- ERROR:  cursor "<unnamed portal 4>" does not exist
```

The problem is that the cursor already closed, as we did not use a transaction. Let's start a transaction, execute the procedure, and fetch rows again:

```
   -- Start a transaction
   BEGIN;

   SELECT show_cities();
   -- Returns: <unnamed portal 2>

   FETCH ALL IN "<unnamed portal 2>";
   COMMIT;
```

Output:

city	state
San Francisco	CA
San Diego	CA
Los Angeles	CA
Austin	TX
Houston	TX
St.Louis	MO

Cursor Name Problem

As you may have noticed, the name of the cursor may change, and it is quite inconvenient to fetch the cursor name first, and then use it in the FETCH statement.

As an option you can slightly redesign a procedure and pass the cursor name as a parameter, so the caller always knows which cursor to fetch:

```
   -- Procedure that returns a cursor (its name specified as the parameter)
   CREATE OR REPLACE FUNCTION show_cities2(ref refcursor) RETURNS refcursor AS $$
    BEGIN
      OPEN ref FOR SELECT city, state FROM cities;   -- Open a cursor
      RETURN ref;                                                  -- Return the cursor to the caller
    END;
    $$ LANGUAGE plpgsql;
```

Now the caller can specify a predefined name:

```
   -- Start a transaction
   BEGIN;

   SELECT show_cities2('cities_cur');
   -- Returns: cities_cur
   FETCH ALL IN "cities_cur";
   COMMIT;
```

Processing Multiple Result Sets

If you call a procedure that returns multiple result sets in PSQL tool, pgAdmin Query tool or another function, the query returns cursor names:

```
   SELECT show_cities_multiple();
```

The result:

show_cities_multiple refcursor
<unnamed portal 3>
<unnamed portal 4>

So to fetch data, you can use a separate FETCH statements for each cursor.

```
   -- Start a transaction
   BEGIN;

   SELECT show_cities_multiple();

   FETCH ALL IN "<unnamed portal 3>";
   FETCH ALL IN "<unnamed portal 4>";
   COMMIT;
```

Output (2 result sets):

city	state
San Francisco	CA
San Diego	CA
Los Angeles	CA

city	state
Austin	TX
Houston	TX

You can also redesign the function, and pass all cursor names as parameters to get predefined cursor names:

```
    -- Procedure that accepts cursor names as parameters
   CREATE OR REPLACE FUNCTION show_cities_multiple2(ref1 refcursor, ref2 refcursor)
   RETURNS SETOF refcursor AS $$
    BEGIN
      OPEN ref1 FOR SELECT city, state FROM cities WHERE state = 'CA';   -- Open the first cursor
      RETURN NEXT ref1;                                                                    -- Return the cursor to the caller

      OPEN ref2 FOR SELECT city, state FROM cities WHERE state = 'TX';   -- Open the second cursor
      RETURN NEXT ref2;                                                                    -- Return the cursor to the caller
    END;
    $$ LANGUAGE plpgsql;
```

Now you can supply cursor names:

```
   -- Start a transaction
   BEGIN;

   SELECT show_cities_multiple2('ca_cur', 'tx_cur');

   FETCH ALL IN "ca_cur";
   FETCH ALL IN "tx_cur";
   COMMIT;
```

Processing a Result Set from a .NET Application

You can call a PostgreSQL stored procedure and process a result set in a .NET application, for example, in C# application using Npgsql .NET data provider.

Note that you do *not* need to know the name of the cursor to process the result set.

```
    // Start a transaction
    NpgsqlTransaction t = conn.BeginTransaction();
    // Specify command StoredProcedure
    NpgsqlCommand command = new NpgsqlCommand("show_cities", conn);
    command.CommandType = CommandType.StoredProcedure;

    // Execute procedure and obtain a result set
    NpgsqlDataReader dr = command.ExecuteReader();

    // Output rows
    while (dr.Read())
        Console.Write("{0}\t{1} \n", dr[0], dr[1]);
```

For a full example as well as processing multiple result sets in .NET, see PostgreSQL and C# - Working with Result Sets

Resources

- PostgreSQL Articles and Reference
- PostgreSQL Stored Procedures and Functions - Getting Started
- PostgreSQL and C# - Working with Result Sets
- PostgreSQL and C# - Npgsql .NET Data Provider - Getting Started
-
- DOKUWIKI