User-Defined Query Language Functions

Nassib Nassar
Director of Metadb Analytics Platform
Head of Research
Index Data ApS
Introduction

Nearly all database users are familiar with the idea of a function or have used one at some point while writing a query.

\[ \sqrt{2} \rightarrow 1.4142135623730951 \]

\[ \text{length('some text')} \rightarrow 9 \]

\[ \text{jsonb_extract_path_text('{"a":"b"}', 'a')} \rightarrow 'b' \]

A user-defined function is a function created by a database user. We will look at a special kind of user-defined function called a user-defined query language function, which is described entirely in SQL, and we will learn how to create one.

This method can be used to create reports.
A sample SQL query

Suppose we have a query that counts the number of loans in a library for each circulated item within a range of dates:

```sql
SELECT item_id,
       count(*) AS loan_count
FROM folio_circulation.loan__t
WHERE '2023-01-01' <= loan_date AND
       loan_date < '2024-01-01'
GROUP BY item_id;
```

The range of dates is defined by a start and end date, in this case, '2023-01-01' and '2024-01-01'.

We can make this query more general by defining the start and end dates as parameters in a user-defined function.
CREATE FUNCTION lisa.count_loans(
    start_date date DEFAULT '2000-01-01',
    end_date date DEFAULT '2050-01-01')
RETURNS TABLE(
    item_id uuid,
    loan_count integer) AS
$$
SELECT item_id,
    count(*) AS loan_count
FROM folio_circulation.loan__t
WHERE start_date <= loan_date AND
    loan_date < end_date
GROUP BY item_id
$$
LANGUAGE SQL;
Calling the function

Since the function returns a table, a good way to call it is to SELECT from it:

```sql
SELECT * FROM lisa.count_loans(
    start_date => '2022-01-01',
    end_date => '2023-01-01');
```

Note that $p => a_p$ defines the parameter name $p$ for argument $a_p$. This should not be confused with the inequality operator in $x >= y$ which means $x$ is greater than or equal to $y$.

Function parameters that have default values can be omitted. For example

```sql
SELECT * FROM lisa.count_loans(
    start_date => '2023-01-01');
```

omits the parameter

```sql
date DEFAULT '2050-01-01'
```
Sharing the function

Suppose that a user lisa has created lisa.count_loans and would like to share it with the users celia and rosalind, so that they also can call it.

First we have to grant them the privilege to use the lisa schema (unless that has been done before):

```
GRANT USAGE ON SCHEMA lisa
    TO celia, rosalind;
```

Then grant the privilege to execute the function:

```
GRANT EXECUTE ON FUNCTION lisa.count_loans
    TO celia, rosalind;
```

This method can be used with a web-based database tool such as CloudBeaver to make reports available to users that do not have a database tool installed locally.