## **Firebird 3.0 stored functions**

The following is an excerpt from the The Firebird 3.0 Release Notes (29 November 2014 - Document v.0300-16 - for Firebird 3.0 Beta 1) chapter, PSQL Stored Functions:

## **PSQL Stored Functions**

### **Dmitry Yemanov**

It is now possible to write a scalar function in PSQL and call it just like an internal function.

#### Syntax for the DDL

```
{CREATE [OR ALTER] | ALTER | RECREATE} FUNCTION <name>
[(param1 [, ...])]
RETURNS lt;type>
AS
BEGIN
...
END
```

**Tip**: The CREATE statement is the *declaration syntax* for PSQL functions, parallel to DECLARE for legacy UDFs.

#### Example

```
CREATE FUNCTION F(X INT) RETURNS INT
AS
BEGIN
RETURN X+1;
END;
SELECT F(5) FROM RDB$DATABASE;
```

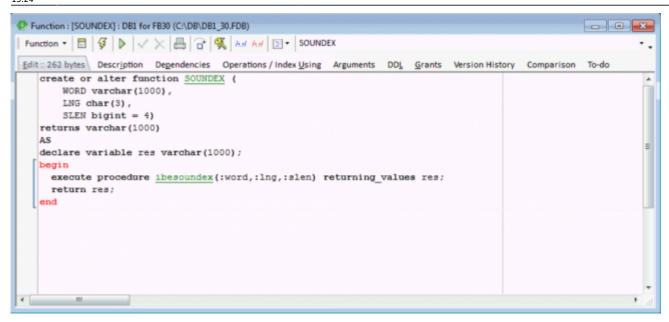
**Source**: *Firebird 3.0 Release Notes* by Helen Borrie (Collator/Editor): 29 November 2014 - Document v.0300-16 - for Firebird 3.0 Beta 1

back to top of page

## **Stored Functions Editor**

The Stored Functions Editor offers a wide range of functions and features that are also available in the Stored Procedure Editor and the Trigger Editor.

Last update: 2023/09/19 02-ibexpert:02-03-database-objects:firebird3-stored-functions http://ibexpert.com/docu/doku.php?id=02-ibexpert:02-03-database-objects:firebird3-stored-functions 13:24



To fix already existing records with an incorrect object type you can execute following UPDATE:

back to top of page

# Why should you use Firebird stored functions?

Stored procedures and stored functions are technically identical. However there are a couple of reasons why you should consider using stored functions rather than stored procedures:

#### Easier to call than procedures

Stored functions are easier to call than procedures. For example:

Example of a stored procedure:

```
select
name,
(select res from sp(kunde.name)) res
from kunde
```

Example of a stored function:

select name, sf(kunde.name) res from kunde

#### More flexible than procedures

Stored functions also offer considerably more flexibility, for example, when being called in a where condition:

select ... where brpsoundex(kunde.name, 'ENG')='K123'

Such nesting in stored procedures can quickly become very confusing.

back to top of page

## **Example using Soundex**

To demonstrate Firebird 3.0 stored functions, we have used Soundex. Soundex searches for similarsounding words. For those of you not familiar with Soundex, please refer first to the Wikipedia definition: https://en.wikipedia.org/wiki/Soundex.

To understand what the Soundex functionality is, we've first created a Soundex procedure, IBESOUNDEX, as a simple implementation, with 3 parameters for the word, language and string length:

```
| Procedure 🔹 🗟 🖗 🕨 🗸 🚬 🖳 📇 🔂 🚧 🛛 🛐 🗷 IBESOUNDEX
Edit Description Dependencies Operations / Index Using Parameters Performance Analysis
                                                                           Plan Analyzer
   create or alter procedure IBESOUNDEX (
       WORD varchar(1000),
       LNG char(3),
       SLEN bigint = 4)
   returns (
       SOUNDEX varchar(1000))
   39
   declare variable CURRC char(1);
   declare variable LASTC char(1);
   declare variable LEN bigint;
   declare variable WEIGHT bigint;
   BEGIN
           - 0
     SOUNDEX -
                             2
     LEN
                            2
     LASTC = ASCII_CHAR(0);
     FOR
         SELECT CASE
                 WHEN POS = 1
                     /* First Position? then use first character. */
                    THEN UPPER (C)
                    /* else use weigth */
                    ELSE CAST (MAXVALUE (WEIGHT, 0) AS CHAR(1))
                END.
                WEIGHT
           FROM
                 SRIROT TRESOUNDERC DOS
```

When we start it, and enter the word that we are searching for - here KLEMT and the language GER,

the result is K453.

The results are always based on groups of similar sounding letters, and groups for the SOUNDEX functionality come from a library. Here we've also used an implementation for the German language, the IBESOUNDEXWEIGHT library.

	Table •	4	1->	- 🖻 🗟 📇	🔊 🚠 📧 Get	record count 1	BESOUNDEXWE	IGHT
E	ields (	onst	raints	Indices Degende	ncies Triggers	Data Maste	er/Detail View	Desci
Ζ	. To	4	Record:	1 🌻 🗊 Σ	<ul> <li>Ν</li> <li>Ν</li></ul>	H <b>+</b> - A	~ % C	
	ID	LNG	C	WEIGHT				
₽	110.547	GER	a	7				
	110.548	GER	b	1				
	110.549	GER	с	2				
	110.550	GER	d	3				
	110.551	GER	e	7				
	110.552	GER	f	8				
	110.553	GER	g	2				
	110.554	GER	h	-1				
	110.555	GER	i i .	7				
	110.556	GER	1	9				
	110.557	GER	k	2				
	110.558	GER	L 1	4				
	110,559	GER	m	5				

It always uses the first character directly, the letter L for example is the number 4; E number 7, M is number 5 and T number 5.

When I enter the German word Maier the result is M760. It can also be spelt Mayer. Again M760. And Meier = M760. You always get the same result, because the names sound the same, even when spelt slightly differently.

The problem is with this kind of implementation, it can take the computer a very long time.

So you can use a function: Define a new function with the same structure, word, language and string length. Define the returns as no name, varchar(1000) and then execute the same procedure:

```
create or alter function SOUNDEX (
    WORD varchar(1000),
    LNG char(3),
    SLEN bigint = 4)
returns varchar(1000)
AS
declare variable res varchar(1000);
begin
    execute procedure ibesoundex(:word,:lng,:slen) returning_values res;
    return res;
end
```

With this statement we now have the possibility to:

select soundex('Maier','GER',4) from rdb\$database; -- result M760

<pre>select soundex('Meier','GER',4)</pre>	<pre>from rdb\$database;</pre>	result M760
<pre>select soundex('Meyer','GER',4)</pre>	<pre>from rdb\$database;</pre>	result M760
<pre>select soundex('Mayer','GER',4)</pre>	<pre>from rdb\$database;</pre>	result M760

or

```
select customer.lastname, soundex(customer.lastname,'GER',4) from customer;
```

There are many potential implementations for this, for internal bug management, data evaluation, and so on.

Using a simple SQL we can view all names in a database, grouped by similarity:

```
select soundex(customer.lastname, 'ENG', 5), count(*), list (distinct
customer.lastname)
from customer
group by 1
order by 2 desc
```

Here we can see the number of names found corresponding to which Soundex group:

🐨 🛛 🗗 🖓 Count reco	tords TiL: Read Committed, RW *	
It: 1 History Plan	Analyzer Berformance Analysis Logs Find query	
select <u>soundex</u> from <u>customer</u> group by 1 order by 2 desc	<pre>customer.lastname, "ENG",5 ,count(*), list (distinct cust count(*), list (distinct cust customer.lastname, "ENG", 5 ,count(*), list (distinct cust customer.lastname, "ENG", customer.lastna</pre>	<u>iomer</u> .lastname)
		3
1		
Messages Results	Query columns Explain plan	
		866 records fetch
	t:1 😨 🖾 Σ όα н ч н c	866 records fetch
SOUNDEX COUNT	t:1 😨 🖾 Σ όα н ч н c	866 records fetch
SOUNDEX COUNT		866 records fetch
SOUNDEX COUNT	1: 1 😧 😰 Σ όα н ч н н 🤅 T UST 58 PAUL/PELIALPOOLE,POWELL,PULLEY,PYWELL	866 records fetch
Y         Y         Record           SOUNDEX         COUNT           P4000	I:     I:    <	866 records fetch
Ya         Ya         Record:           SOUNDEX         COUNT           P4000         86500           86300         86300	I: 1 D DQ H + H C T LIST 58 PAUL, PELLAI, POOLE, POWELL, PULLEY, PYWELL 55 BARONE, BARRON, BORUM, BOURNE, BROWNE 49 BAIRD, BARRETT, BERTEI, BIRD, BURT	866 records fetch
Ym         Ym         Ym         Record:           SOUNDEX         COUNT         P4000         B6500         B6300         B6300         B6000         B6000 <td>Image: Image: Imag</td> <td>866 records fetch</td>	Image: Imag	866 records fetch
Y         Y         Record:           SOUNDEX         COUNT           P4000         66500           86300         86000           M6000         M6000	Image: Paul, PELLAI, POOLE, POWELL, PULLEY, PYWELL         S8         PAUL, PELLAI, POOLE, POWELL, PULLEY, PYWELL         S5         BAROHE, BARRON, BORUM, BOURNE, BROWNE         49         BAIRD, BARRETT, BERTELBIRD, BURT         48         BARR, BARROW, BAUER, BERRY         46         MAHER, MEYER, MOORE, MORIE, MURRAY	866 records fetch
Y         Y         Record:           SOUNDEX         COUNT           P4000         65500           86300         86000           M6000         W4900	Image: Paul, PELLAL POOLE, POWELL, PULLEY, PYWELL         SB         SB <td>866 records fetch</td>	866 records fetch
Y         Y         Record:           SOUNDEX         COUNT           P4000         66500           86300         66000           M6000         W4900           H4000         6000	Image: Second state     Image: Second state       Im	866 records fetch
Y         Y         Record:           SOUNDEX         COUNT           P4000         66500           86300         66000           M6000         W4900           H4000         P6300	Image: Paul, PILLAL POOLE, POWELL, PULLEY, PYWELL         S8         PAUL, PILLAL POOLE, POWELL, PULLEY, PYWELL         S8         PAUL, PILLAL POOLE, POWELL, PULLEY, PYWELL         S8         S8         PAUL, PILLAL POOLE, POWELL, PULLEY, PYWELL         S8         S8         PAUL, PILLAL POOLE, POWELL, PULLEY, PYWELL         S8	866 records fetch
Y         Y         Record:           SOUNDEX         COUNT           P4000         66500           86300         66000           M6000         W4900           H4000         P6300	Image: Second state       OQ       Image: Second state         T       LIST         S8       PAUL, PILLAL POOLE, POWELL, PULLEY, PYWELL         S5       BARONE, BARRON, BORUM, BOURNE, BROWN, BROWNE         49       BARD, BARRETT, BERTELBIRD, BURT         48       BARR, BARROW, BALER, BERRY         46       MAHER, MEYER, MOORE, MORIE, MURRAY         43       WALLS, WELLS, WELSH, WILLIS         41       HALL, HEALY, HILL, HOLLEY, HOWELL         41       VANG, WINGO, WOMMACK	866 records fetch

For example, paul, pillai, poole, powell, pulley, pywell. Or barone, barron etc. they are all similar sounding.

This implementation can be very useful, even if it can occasionally show up some surprising results.

So you can see that using the internal function makes it much easier to implement. You can even use this functionality to index your data.

We find stored functions a really great new feature, as they offer a lot of possibilities inside the database. We hope you do too!

From: http://ibexpert.com/docu/ - **IBExpert** 

Permanent link: http://ibexpert.com/docu/doku.php?id=02-ibexpert:02-03-database-objects:firebird3-stored-functions

Last update: 2023/09/19 13:24

