Cyber Sandwich

Adding Custom Delimiters to a String with Regex, PostgreSQL and PL/pgSQL

April 24, 2018
DatabaseProgramming
PL/pgSQLpostgrespostgreSQL

On this post I will be covering how to use regex to add custom delimiters to a string. The code referenced in this post was written to work in Postgres 10.3. The code will not work as expected in Postgres 9.6 or below. The function that I created as an example for this post takes a string as input and add a '_' as the delimiter between digits and letters. Below is the output of the function.

```
SELECT add_delimiter('test12foo34');
add_delimiter
----------------
test_12_foo_34
```

Below is the complete code for the add_delimiter function. I will cover each section in detail throughout this post.

```
CREATE OR REPLACE FUNCTION add_delimiter(str VARCHAR) RETURNS text AS $$
DECLARE
new_string text;
split_char varchar;
regex varchar;
prefixes varchar[];
regex_parts varchar[];
regex_parts_length int;
i int;
BEGIN
split_char := '_';
regex := '([a-zA-Z]{1}\d{1}|\d{1}[a-zA-Z]{1})';
prefixes := regexp_split_to_array(str,regex);
regex_parts := (SELECT ARRAY(select array_to_string(regexp_matches(str,regex,'g'),'')));
regex_parts_length := array_upper(regex_parts,1);
new_string := '';
FOR i IN 1..array_upper(prefixes,1)
LOOP
new_string := new_string || prefixes[i];
```

```
IF i <= regex_parts_length THEN
new_string := new_string || substring(regex_parts[i],1,1)||split_char||substring(regex_parts[i],2,1);
END IF;
END LOOP;
RETURN new_string;
END; $$
LANGUAGE PLPGSQL;
```

In the first section of the function we define our delimiter and regex.

```
split_char := '_';
regex := '([a-zA-Z]{1}\d{1}|\d{1}[a-zA-Z]{1})';
```

The regex in my example matches any occurrence of a digit next to a letter and vice-versa.
The next statement:

```
prefixes := regexp_split_to_array(str,regex);
```

Returns the parts of the string which do not match the regular expression. From the Postgres docs: “Split string using a POSIX regular expression as the delimiter”. Here is what regexp_split_to_array returns with my example:

```
select regexp_split_to_array('test12foo34','([a-zA-Z]{1}\d{1}|\d{1}[a-zA-Z]{1})');
regexp_split_to_array
----------------------
{tes,"",o,4}
```

The following section:

```
regex_parts := (SELECT ARRAY(select array_to_string(regexp_matches(str,regex,'g'),'')));
```

returns all the sections of the string that match the regex.

```
SELECT ARRAY(select array_to_string(regexp_matches('test12foo34','([a-zA-Z]{1}\d{1}|\d{1}[a-zA-Z]
{1})','g'),''));
array
------------
{t1,2f,o3}
```

The outer function ARRAY() is necessary to return the result of regexp_matches as a single array, because regexp_matches will return multiple array rows as seen here:

```
SELECT regexp_matches('test12foo34','([a-zA-Z]{1}\d{1}|\d{1}[a-zA-Z]{1})','g');
regexp_matches
---------------
{t1}
{2f}
{o3}
```

The next code section loops over the sections of the string that do not match the regex and creates a new string by concatenating the parts that do not match the regex with the parts that do. The function array_upper() is used to get the number of elements in the array. What is important to note is that in PL/pgSQL arrays start from index 1. In my example the sections that match the regex must be 2 characters in length, therefore it is trivial to add a delimiter where it is required.

Creating this function was my first introduction to the power of PL/pgSQL, and I'm sure now that I am more familiar with it I will be using it again in the future. Here are the links to the PostgreSQL documentation that I used for this post: String Functions, Array Functions, and PL/pgSQL Tutorial. I hope you enjoyed! Thanks for reading.

Next Post
Previous Post

- Search for: Search
- **Recent Posts**
  - Next Closest Time (Leetcode Challenge)
  - Adding Custom Delimiters to a String with Regex, PostgreSQL and PL/pgSQL
  - Dynamically Update Multiple Rows with PostgreSQL and Python
  - CTEs, Recursion, and DFS in postgreSQL
  - Generating Custom CSV Reports With Tshark
- **Archives**
  - February 2019
  - April 2018
  - November 2017
  - March 2017
  - November 2016
  - August 2016
- **Categories**
  - Database
  - Networking
  - Programming
  - Reverse Engineering
- **Friends**
  - Zeall
  - Lucas
  - Travis

- Home
- About Me