Встроенные функции

Типы функций

- Обзор встроенных функций
- Функции для работы с текстом
- Функции для работы с датой/временем
- Функции для работы с числами
- Функции преобразования и форматирования
- Функции для работы с NULL
- Системные информационные функции

Обзор встроенных функций

- PostgreSQL предоставляет широкий набор встроенных функций, работающих со встроенными типами данных
- Все встроенные функции разделяются на **стандартные функции SQL92** и **функции в стиле PostgreSQL**:
  - В функциях SQL92 аргументы разделяются специальными ключевыми словами SQL (такими, как ***FROM***, ***FOR*** и ***USING***)

    функция_в_стиле_sq192 ({ аргумент \ КЛЮЧЕВОЕ_СЛОВО } [.:.])

  - Функции в стиле PostgreSQL получают аргументы, разделенные запятыми

    функция_в_стиле_pgsql (аргумент [, ...])

- ВНИМАНИЕ
  - Круглые скобки не обязательны **только** для ряда функций SQL92

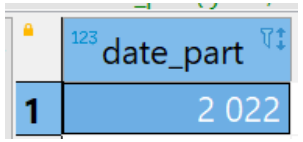
```
select current_date, current_time, current_timestamp ,
       current_user, current_catalog, current_role, current_schema;
```

Использование вложенных функций

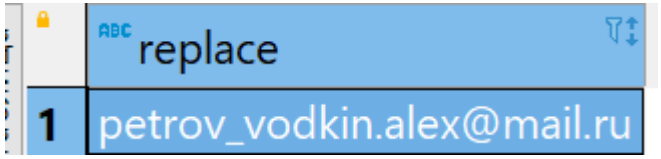
- Вызовы функций могут быть вложенными
  - тип данных, возвращаемый внутренней функцией должен быть совместим с типом соответствующего аргумента внешней функции
- Допускается вложение вызовов на произвольную глубину:

```
имя_внешней_функции (имя_вложенной_функции (аргументы [. ... ] ) [. ...] )
```

```
select date_part('year', current_date);
```




```
select replace(lower(concat('Petrov-Vodkin','.','Alex','@mail.ru')),'-','_');
```



Функции для работы с текстом

Строковые функции

- В PostgreSQL существует множество разнообразных строковых функций, предназначенных для форматирования, анализа и сравнения строк
  - Под строками в данном контексте подразумеваются значения типов character, character varying и text

```
select
   substring('Иванов, Максим' from '^(.+),') as "Lname1",
   substring('Иванов, Максим' from 1 for position(',' in 'Иванов, Максим')-1) as "Lname2",
   left('Иванов, Максим', -- 1-й аргумент
         position(',' in 'Иванов, Максим')-1) as "Lname3", -- 2-й аргумент
   right('Иванов, Максим', -- 1-й аргумент
         char_length('Иванов, Максим')-position(',' in 'Иванов, Максим')-1) as "Fname1", -- 2-й аргумент
   right('Иванов, Максим', -- 1-й аргумент
         -position(',' in 'Иванов, Максим')-1) as "Fname2"; -- 2-й аргумент
```

	Lname1	Lname2	Lname3	Fname1	Fname2
1	Иванов	Иванов	Иванов	Максим	Максим

Строковые функции

Функция	Функция
char_length(‘*string*’) **character_length**(‘*string*’) **length**(‘*string*’)	**trim**([**leading** | **trailing** | **both**] [‘*characters*’] **from** ‘*string*’) **ltrim**(‘*string*’ [, ‘*characters*’]) **rtrim**(‘*string*’ [, ‘*characters*’])
lower(‘*string*’) **upper**(‘*string*’)	**right**(‘*string*’, count_int), --*count_int м.б. + или -* **left**(‘*string*’, count_int)
strpos(‘*string*’, ‘*substring*’) **position**(‘*substring*’ **in** ‘*string*’)	**concat**(*arg1* [, *arg2* [, ...]]), **concat_ws**(‘*separator*’ , *arg1* [, *arg2* [, ...]])
substr(‘*string*’ , from_int [, count_int]) **substring**(‘*string*’ [**from** int] [**for** int]) **substring**(‘*string*’ **from** ’*шаблон* ***POSIX***’) **substring**(‘*string*’ **from** ‘*шаблон* ***SQL***’ **for** ‘*спецсимвол*’) **regexp_matches** (*string*’, ‘*шаблон* ***POSIX***’)	**replace**(‘*string*’, ‘*old text*’, ‘*new text*’), **translate**(‘*string*’, ‘*old text*’, ‘*new text*’) **overlay**(‘*string*’ **placing** ‘*substring*’ **from** int [for int]) **regexp_replace**(‘*string*’, ’*шаблон* ***POSIX***’, ‘*replacement*’)
repeat(‘*string*’, count_int) **reverse**(‘*string*’)	**split_part**(‘*string*’ text, ‘*delimiter*’, item_int) **regexp_split_to_array**(‘*string*’, ’*шаблон* ***POSIX***’)

https://postgrespro.ru/docs/postgresql/12/functions-string

Использование шаблонов

- Для определения шаблонов в PostgreSQL поддерживается два типа регулярных выражений:
  - Регулярные выражения в стиле SQL
  - Регулярные выражения в стиле POSIX

Регулярные выражения в стиле SQL

- Для определения шаблона в стиле SQL используются:

_	любой один символ	'_етров' => 'Ветров', 'Петров' …
%	любая строка, содержащая ноль или более символов	'компьютер%'=>'компьютер','компьютеры', 'компьютерный'…

```
select substring('очень длинная строка' from '%*"д%я*"%' for '*');
```

	substring
1	длинная

Регулярные выражения POSIX

- Для определения шаблона в стиле POSIX используются следующие метасимволы:

.	любой один символ
[...]	любой одиночный символ в диапазоне или наборе
[^...]	любой символ, кроме указанных в диапазоне или наборе
*****	повторение предыдущего элемента 0 и более раз
+	повторение предыдущего элемента 1 и более раз
?	вхождение предыдущего элемента 0 или 1 раз
{m}	повторение предыдущего элемента ровно m раз
{m,}	повторение предыдущего элемента m или более раз
{m,n}	повторение предыдущего элемента не менее чем m и не более чем n раз
()	объединение нескольких элементов в одну логическую группу
|	выбор (одного из двух вариантов)
^	привязывает шаблон к началу строки
$	привязывает шаблон к концу строки

https://postgrespro.ru/docs/postgresql/12/functions-matching#FUNCTIONS-LIKE

Регулярные выражения POSIX

```
select substring('очень длинная% строка' from '.{5}(.+%).+');

select substring('очень длинная% строка' from '^.+\s(.+%)');

select substring('очень длинная% строка' from '(.+%).+$');
```

	substring
1	длинная%

Функция format

- Функция **format** выдаёт текст, отформатированный в соответствии со строкой формата

```
format(formatstr text [, formatarg "any" [, ...] ])
```

- ***formatstr*** – спецификаторы формата
  - **позиция** - строка вида **n$**, где n — индекс выводимого аргумента. Если позиция опускается, по умолчанию используется следующий аргумент по порядку.
  - **флаги** - поддерживается только **знак минус (-),** который выравнивает результат спецификатора по левому краю если определена ширина
  - **ширина** - минимальное число символов, которое будет занимать результат данного спецификатора
  - **тип** спецификатора - определяет преобразование соответствующего выводимого значения
    - S – строка
    - I - SQL-идентификатор, при необходимости заключается в кавычки
    - L - значение аргумента заключается в апострофы, как строка SQL

```
%[позиция][флаги][ширина]тип
```

```
select format('Hello, dear %2$-10s  %1$15s', e.lastname ,e.firstname)
from "HR"."Employees" e;
```

```
                format
-------------------------------------
Hello, dear Don                 Funk
Hello, dear Judy                 Lew
Hello, dear Yael                Peled
Hello, dear Maria             Cameron
Hello, dear Sven                Buck
Hello, dear Paul                Suurs
Hello, dear Russell             King
Hello, dear Zoya          Dolgopyatova
Hello, dear Sara                Device
Hello, dear Allen Klod         Johnson
```

Функции для работы с датой/временем

Функции даты/времени

Функция	Функция
age (timestamp) **age** (timestamp, timestamp)	**date_trunc** (‘part’, timestamp) **date_trunc** (text, interval) **date_trunc** (‘part’, timestamp with time zone, ‘time_zone_name’)
current_date **current_time** **current_time** (integer) **current_timestamp** **current_timestamp** (integer) **clock_timestamp** ( )	**date_part** (‘part’, timestamp) **date_part** (’part’, interval) **extract** (part **from** timestamp) **extract** (part **from** interval)
now () **localtime** **localtimestamp**	**make_date** (year int, month int, day int) **make_time** (hour int, min int, sec double precision) **make_timestamp** (year int, month int, day int, hour int, min int, sec double precision)

Date_part

- microseconds
- milliseconds
- second
- minute
- hour
- day
- week
- month
- quarter
- year
- decade
- century
- millennium

https://postgrespro.ru/docs/postgresql/14/functions-datetime#FUNCTIONS-DATETIME-TABLE

Часовые пояса (timezone names)

```
select name, abbrev, utc_offset
from  pg_timezone_names;
```

	name	abbrev	utc_offset
1	Africa/Abidjan	GMT	00:00:00
2	Africa/Accra	GMT	00:00:00
3	Africa/Addis_Ababa	EAT	03:00:00
4	Africa/Algiers	CET	01:00:00
591	US/Samoa	SST	-11:00:00
592	UTC	UTC	00:00:00
593	W-SU	MSK	03:00:00
594	WET	WEST	01:00:00
595	Zulu	UTC	00:00:00

```
SELECT TIMESTAMP '2022-02-16 20:38:40' AT
                TIME ZONE 'America/Denver';
```

	timezone
1	2022-02-17 06:38:40

```
select date_trunc('day', timestamptz '2022-10-16 20:38:40+00', 'Australia/Sydney'),
       date_trunc('day', timestamptz '2022-10-16 20:38:40+00', 'US/Samoa');
```

	date_trunc	date_trunc
1	2022-10-16 16:00:00	2022-10-16 14:00:00

Функции даты/времени

```
SELECT  age('2022-06-25 12:34'::timestamp ),
        clock_timestamp( ),
        clock_timestamp( );
```

age interval	clock_timestamp timestamp with time zone	clock_timestamp timestamp with time zone
4 mons 1 day 11:26:00	2022-10-27 07:36:33.702299+00	2022-10-27 07:36:33.7023+00

```
SELECT extract(hour from timestamp '2001-02-16 20:38:40'),
       date_part('hour', timestamp '2001-02-16 20:38:40');
```

123 date_part	123 date_part
20	20

Математические функции

Математические функции

Функция	Функция
random()	**abs** (***x***)
ceil (dp или numeric) **ceiling** (dp или numeric) **floor** (dp или numeric) **round** (dp или numeric) **round** (*v* numeric, *s* int) **trunc** (dp или numeric) **trunc** (*v* numeric, *s* int)	**div** (*y* numeric, *x* numeric) **mod** (*y*, *x*)
power (*a* dp, *b* dp) **power** (*a* numeric, *b* numeric)	**sqrt** (dp или numeric) **cbrt** (dp)

***dp - double precision**

https://postgrespro.ru/docs/postgresql/12/functions-math

Функции преобразования и форматирования

Преобразование типов

- Для явного преобразования типов используется
  - Функция **CAST (стандарт SQL) -** **CAST (выражение AS type)**
  - Конструкция **:: (PostgreSQL) -** **выражение ::type**
  - Синтаксис функций приведения к типу - **typename (выражение)**
    - **будет работать только для типов, имена которых являются допустимыми именами функций!**
  - Запись вида **typename** 'string'

```
select   cast('2022-02-12' as date),
         date($$2022-02-12$$),
         '2022-02-12'::date,
         date '2022-02-12';
```

	date	date	date	date
1	2022-02-12	2022-02-12	2022-02-12	2022-02-12

https://postgrespro.ru/docs/postgresql/12/sql-syntax-lexical

Внимание

- Приведение будет успешным, только если определён подходящий оператор преобразования типов
- Явное приведение типа можно опустить, если возможно однозначно определить, какой тип должно иметь выражение (неявное преобразование)
- Запись **typename** 'string‘
  - можно использовать **только** для указания типа простой текстовой константы
  - не работает для массивов

https://postgrespro.ru/docs/postgresql/12/sql-expressions#SQL-SYNTAX-TYPE-CASTS

Функции форматирования

Функция	Описание	Пример
to_char (timestamp, text)	преобразует время в текст	to_char(current_timestamp, 'HH12:MI:SS')
to_char (interval, text)	преобразует интервал в текст	to_char(interval '15h 2m 12s', 'HH24:MI:SS')
to_char (int, text)	преобразует целое в текст	to_char(125, '999')
to_char (double precision, text)	преобразует плавающее одинарной/двойной точности в текст	to_char(125.8::real, '999D9')
to_char (numeric, text)	преобразует числовое значение в текст	to_char(-125.8, '999D99S')
to_date (text, text)	преобразует текст в дату	to_date('05 Dec 2000', 'DD Mon YYYY')
to_number (text, text)	преобразует текст в число	to_number('12,454.8-', '99G999D9S')
to_timestamp (text, text)	преобразует строку во время	to_timestamp('05 Dec 2000', 'DD Mon YYYY')

https://postgrespro.ru/docs/postgresql/12/functions-formatting

Функции для работы с NULL

Функции для работы с NULL

- Функция **COALESCE** возвращает первый попавшийся аргумент, отличный от NULL.
  - Если же все аргументы равны NULL, результатом тоже будет NULL

```
COALESCE(значение [, ...])
```

- Функция **NULLIF** выдаёт значение NULL, если значение1 равно значение2; в противном случае она возвращает значение1.
  - Это может быть полезно для реализации обратной операции к COALESCE.

```
NULLIF(значение1, значение2)
```

```
select COALESCE(null,'Addr2', 'Addr3') ,
       NULLIF('Address 1', 'Address 2'),
       NULLIF('Address 1', 'Address 1');
```

	coalesce	nullif	nullif
1	Addr2	Address 1	[NULL]

https://postgrespro.ru/docs/postgresql/12/functions-conditional#FUNCTIONS-COALESCE-NVL-IFNULL

Системные информационные функции

Функции получения информации о сеансе

Имя	Тип рез.	Описание
current_catalog **current_database** ()	name	имя текущей базы данных (в стандарте SQL она называется «каталогом»)
current_user **current_role**	name	имя пользователя в текущем контексте выполнения
session_user	name	имя пользователя сеанса
current_schema [()]	name	имя текущей схемы
pg_backend_pid ()	int	код серверного процесса, обслуживающего текущий сеанс
pg_blocking_pids (int)	int[]	идентификаторы процессов, не дающих серверному процессу с определённым ID получить блокировку
version ()	text	информация о версии PostgreSQL

```
SELECT current_database(), current_user, pg_backend_pid(), version();
```

	current_database	current_user	pg_backend_pid	version
1	dbSQL	postgres	15 384	PostgreSQL 12.4, compiled by Visual C++ build 1914, 64-bit

https://postgrespro.ru/docs/postgresql/12/functions-info