



# 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);
```

	date_part
1	2 022

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

	replace
1	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
<code>char_length('string')</code> <code>character_length('string')</code> <code>length('string')</code>	<code>trim([leading   trailing   both] ['characters'] from 'string')</code> <code>ltrim('string' [, 'characters'])</code> <code>rtrim('string' [, 'characters'])</code>
<code>lower('string')</code> <code>upper('string')</code>	<code>right('string', count_int), --count_int м.б. + или -</code> <code>left('string', count_int)</code>
<code>strpos('string', 'substring' )</code> <code>position('substring' in 'string')</code>	<code>concat(arg1 [, arg2 [, ...] ]),</code> <code>concat_ws('separator', arg1 [, arg2 [, ...] ])</code>
<code>substr('string' , from_int [, count_int])</code> <code>substring('string' [from int] [for int])</code> <code>substring('string' from 'шаблон POSIX')</code> <code>substring('string' from 'шаблон SQL' for 'специсимвол')</code> <code>regexp_matches (string, 'шаблон POSIX' )</code>	<code>replace('string', 'old text', 'new text'),</code> <code>translate('string', 'old text', 'new text')</code> <code>overlay('string' placing 'substring' from int [for int])</code> <code>regexp_replace('string', 'шаблон POSIX', 'replacement')</code>
<code>repeat('string', count_int)</code> <code>reverse('string')</code>	<code>split_part('string' text, 'delimiter', item_int)</code> <code>regexp_split_to_array('string', 'шаблон POSIX')</code>

# Использование шаблонов

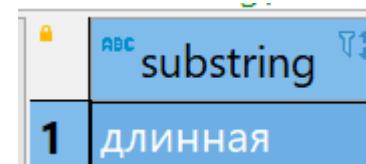
- Для определения шаблонов в PostgreSQL поддерживается два типа регулярных выражений:
  - Регулярные выражения в стиле SQL
  - Регулярные выражения в стиле POSIX

# 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 '*');
```



A screenshot of a PostgreSQL query result. The query is:

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

The result shows one row with the value 'длинная'.

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 '( .+%).+$') ;
```

A ABC	substring
1	длинная%

# ФУНКЦИЯ format

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

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

- formatstr** – спецификаторы формата % [позиция] [флаги] [ширина] тип

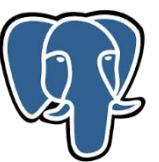
- **позиция** - строка вида **n\$**, где **n** — индекс выводимого аргумента. Если позиция опускается, по умолчанию используется следующий аргумент по порядку.
- **флаги** - параметры, управляющие форматированием данного спецификатора. Поддерживается только знак минус (-), который выравнивает результат 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 строкау SQL

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

	format	
1	Hello, dear Don	Funk
2	Hello, dear Judy	Lew
3	Hello, dear Yael	Peled
4	Hello, dear Maria	Cameron



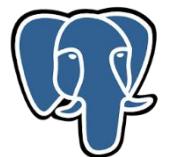
# Функции для работы с датой/временем



PostgreSQL

# ФУНКЦИИ ДАТЫ/ВРЕМЕНИ

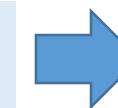
Функция	Функция	Date_part
<b>age</b> ( timestamp ) <b>age</b> ( timestamp, timestamp )	<b>date_trunc</b> ('part', timestamp ) <b>date_trunc</b> (text, interval) <b>date_trunc</b> ('part', timestamp with time zone, 'time_zone_name')	microseconds milliseconds second minute hour day week month quarter year decade century millennium
<b>current_date</b> <b>current_time</b> <b>current_time</b> ( integer ) <b>current_timestamp</b> <b>current_timestamp</b> ( integer ) <b>clock_timestamp</b> ( )	<b>date_part</b> ( 'part', timestamp ) <b>date_part</b> ( 'part', interval ) <b>extract</b> (part from timestamp) <b>extract</b> (part from interval)	
<b>now</b> () <b>localtime</b> <b>localtimestamp</b>	<b>make_date</b> (year int, month int, day int) <b>make_time</b> (hour int, min int, sec double precision) <b>make_timestamp</b> (year int, month int, day int, hour int, min int, sec double precision)	



PostgreSQL

# 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
2022-02-17 06:38:40

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

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

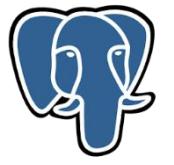
# ФУНКЦИИ ДАТЫ/ВРЕМЕНИ

```
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');
```

date_part	date_part
20	20



# Математические функции

# Математические функции

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

\*dp - double precision

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



Postgre**SQL**

# 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

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



Postgre**SQL**

# 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**
- 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**.

**COALESCE(значение [, ...])**

**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]

# Системные информационные функции

# ФУНКЦИИ ПОЛУЧЕНИЯ ИНФОРМАЦИИ О СЕАНСЕ

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