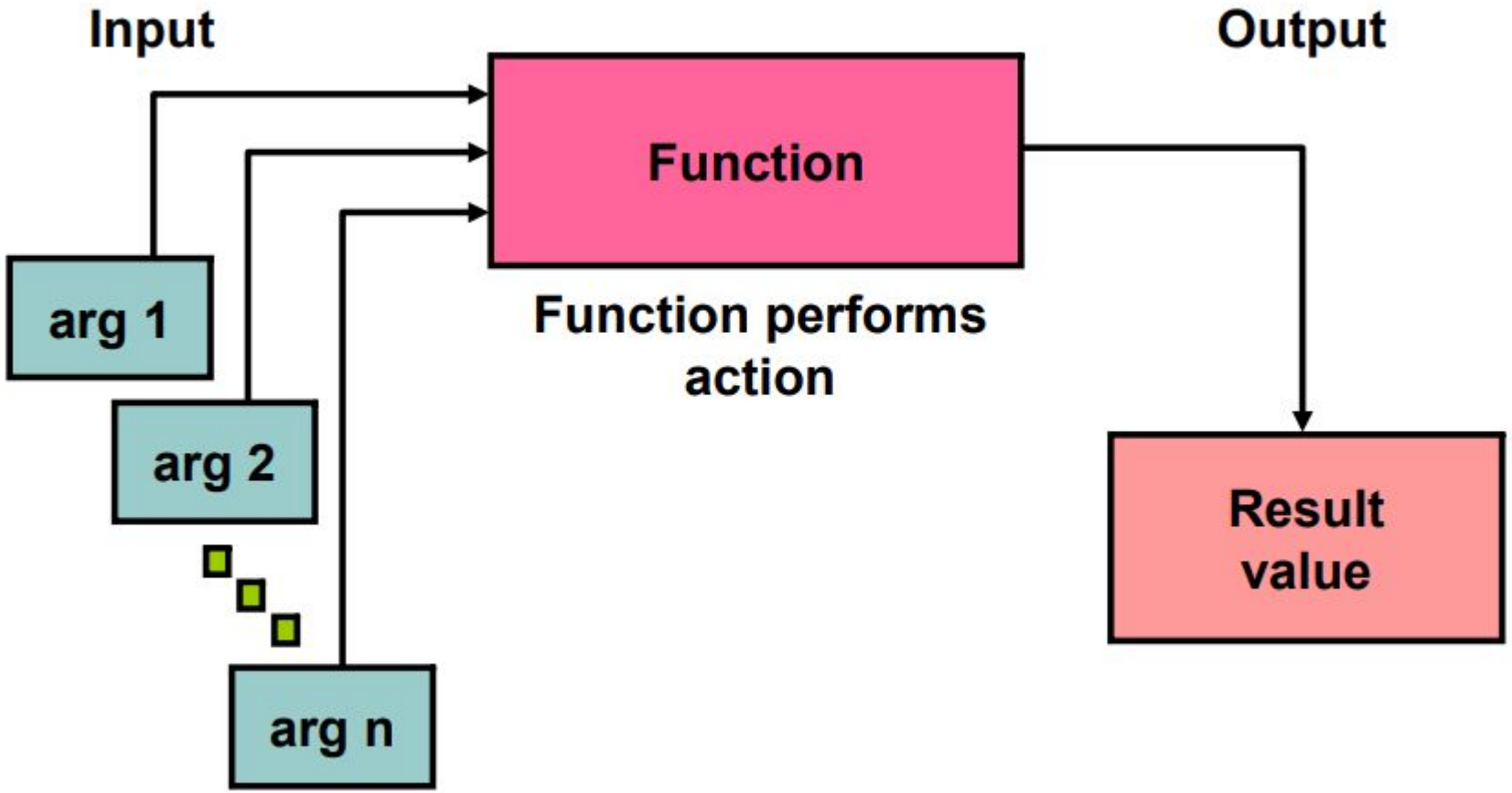
Использование Single-Row функция для пользовательского вывода

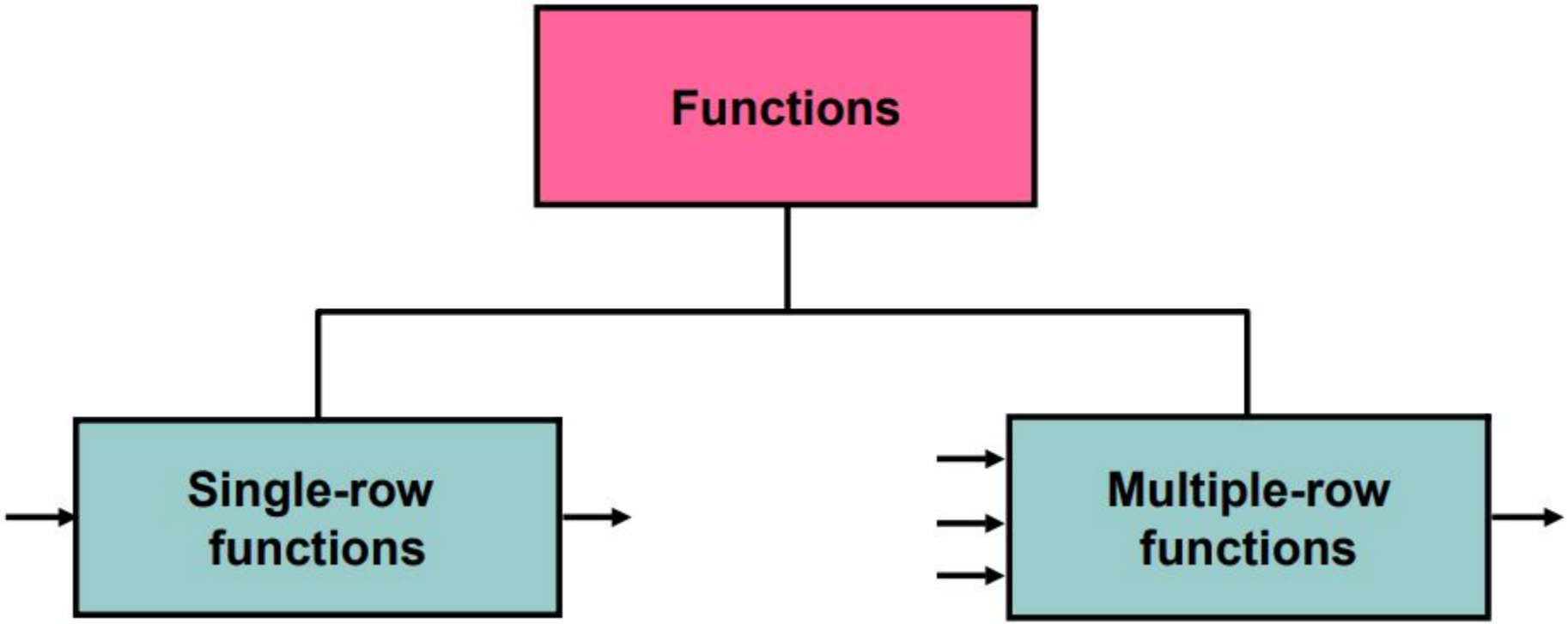
Цели

- Знакомство с функциями различного типа, доступных в SQL
- Использование числовых, символьных и функций для работы с датами в секции SELECT
- Использование функций, для преобразования типов

Функция SQL


Input
Output
Function
Function performs action
arg 1
arg 2
arg n
Result value

Типы функций SQL

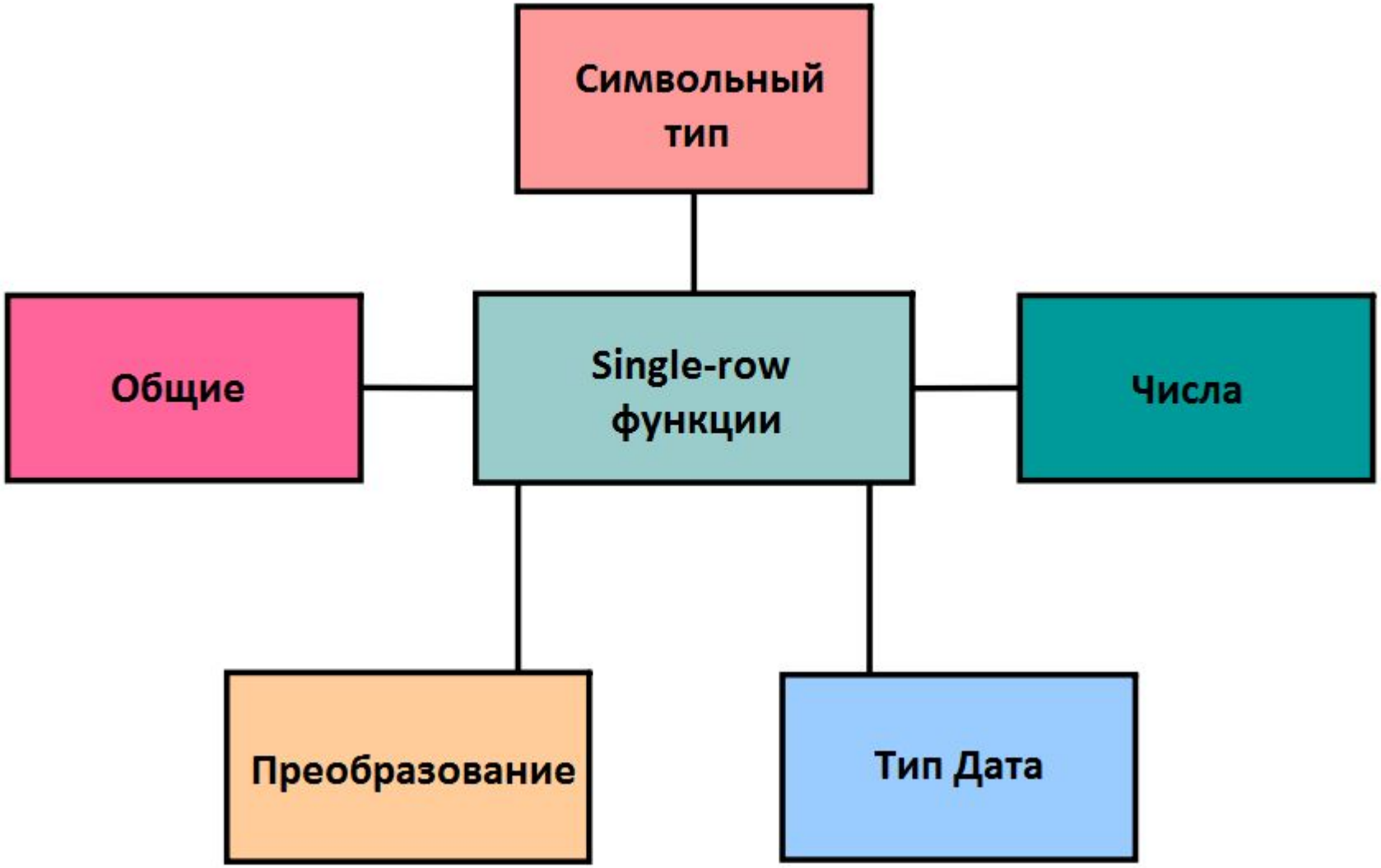


Single-Row функции

- Манипулирование данными
- Принимает несколько аргументов, возвращает единственное значение
- Применяется к каждой возвращаемой строке
- Возвращает единственное значение для строки
- Может изменять тип данных
- Может быть вложенной
- Принимает в качестве аргумента колонку или выражение

```
function_name [(arg1, arg2,...)]
```

Single-Row функции



Функции для символьного типа

- Оперирование регистром
  - **lower**
  - **upper**
  - **initcap**

Оперирование содержимым

- **concat**
- **concat_ws**
- **substring | substr**
- **length**
- **position | strops**
- **lpad | rpad**
- **trim | ltrim |rtrim**
- **replace**
- **repeat**
- **reverse**
- **format**
- **left | right**
- **starts_with**

Функции, для оперирования регистром

Функция	Результат
`lower('SQL Course')`	`sql course`
`upper('SQL Course')`	`SQL COURSE`
`initcap('SQL Course')`	`Sql Course`

Функции для оперирования регистром

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE last_name = initcap('higgins');
```

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE lower(last_name) = 'higgins';
```

	employee_id [PK] integer	last_name character varying (25)	department_id integer
1	205	Higgins	110

Функции для оперирования содержимым

concat(val1 "any" [, val2 "any" [, ...]]) => text *Соединяет текстовые представления всех аргументов, игнорируя NULL.* **concat('abcde', 2, NULL, 22) => abcde222**
concat_ws(*sep* text, *val1* "any" [, *val2* "any" [, ...]]) => text *Соединяет вместе все аргументы, кроме первого, через разделитель.* *Разделитель задаётся в первом аргументе и должен быть отличен от NULL.* **concat_ws(',', 'abcde', 2, NULL, 22) => abcde,2,22**
left(*string* text, *n* integer) => text *Возвращает первые n символов в строке. Когда n меньше нуля, возвращаются все символы слева,* *кроме последних |n|.* **left('abcde', 2) => ab**
right(*string* text, *n* integer) => text *Возвращает последние n символов в строке. Когда n меньше нуля, возвращаются все символы справа,* *кроме первых |n|.* **right('abcde', 2) => de**

Функции для оперирования содержимым

```
SELECT employee_id,
     first_name,
     last_name,
     concat(first_name, last_name) bad_name,
     concat_ws(' ', first_name, last_name) good_name,
     left(first_name, 2) as left_fn,
     right(last_name, 3) as right_ln
  FROM employees
 WHERE job_id = 'SA_REP';
```

	employee_id [PK] integer	first_name character varying (20)	last_name character varying (25)	bad_name text	good_name text	left_fn text	right_ln text
1	150	Peter	Tucker	PeterTucker	Peter Tucker	Pe	ker
2	151	David	Bernstein	DavidBernstein	David Bernstein	Da	ein
3	152	Peter	Hall	PeterHall	Peter Hall	Pe	all

Функции для оперирования содержимым

substring(*string* text [FROM *start* integer] [FOR *count* integer]) => text

Извлекает из **string** *подстроку, начиная с позиции* **start** *(если она указана), длиной до* **count** *символов (если она указана). Параметры* **start** *и* **count** *могут опускаться, но не оба сразу.*

```
substring('Thomas' from 2 for 3) => hom
substring('Thomas' from 3)       => omas
substring('Thomas' for 2)        => Th
```

substr(*string* text, *start* integer [, *count* integer]) => text

Извлекает из **string** *подстроку, начиная с позиции* **start**, *длиной до* **count** *символов.*

```
substr('alphabet', 3)    => phabet
substr('alphabet', 3, 2) => ph
```

length(text) => integer

Возвращает число символов в строке.

```
length('jose') => 4
```

starts_with(*string* text, *prefix* text) => boolean

Возвращает **true**, *если строка* **string** *начинается с подстроки* **prefix**.

```
starts_with('alphabet', 'alph') => t
```

Функции для оперирования содержимым

```
SELECT first_name,
     last_name,
     length(first_name) as len_fn,
     job_id,
     substring(job_id from 4 for 3) as job_name,
     starts_with(first_name, 'P') "Starts with 'P'?"
  FROM employees
 WHERE substr(job_id, 4) = 'REP';
```

	first_name character varying (20)	last_name character varying (25)	len_fn integer	job_id character varying (10)	job_name text	Starts with 'P'? boolean
1	Peter	Tucker	5	SA_REP	REP	true
2	David	Bernstein	5	SA_REP	REP	false
3	Peter	Hall	5	SA_REP	REP	true

Функции для оперирования содержимым

position(*substring* text IN *string* text) => integer

*Возвращает начальную позицию первого вхождения **substring** в строке **string** либо 0, если такого вхождения нет.*

position('om' in 'Thomas') => 3

strpos(*string* text, *substring* text) => integer

*Возвращает начальную позицию первого вхождения **substring** в строке **string** либо 0, если такого вхождения нет.*

strpos('Thomas', 'om') => 3

lpad(*string* text, *length* integer [, *fill* text]) => text

*Дополняет строку **string** слева до длины **length** символами **fill** (по умолчанию пробелами). Если длина строки уже больше заданной, она обрезается справа.*

lpad('hi', 5, 'xy') => xyxhi

rpad(*string* text, *length* integer [, *fill* text]) => text

Дополняет строку ***string*** справа до длины ***length*** символами ***fill*** (по умолчанию пробелами). Если длина строки уже больше заданной, она обрезается.

rpad('hi', 5, 'xy') => hixyx

Функции для оперирования содержимым

```
SELECT first_name,
     last_name,
     position('t' in first_name) as "Have t ?",
     lpad(first_name, 10, '$') lpad_fn,
     rpad(last_name, 10, '#') lpad_fn
  FROM employees
 WHERE strpos(job_id, 'REP') > 0
```

	first_name character varying (20)	last_name character varying (25)	Have t ? integer	lpad_fn text	lpad_fn text
1	Peter	Tucker	3	$$$$$Peter	Tucker####
2	David	Bernstein	0	$$$$$David	Bernstein#
3	Peter	Hall	3	$$$$$Peter	Hall######

Функции для оперирования содержимым

trim([LEADING|TRAILING|BOTH] [*characters* text] FROM *string* text) => text

Удаляет наибольшую подстроку, содержащую только символы ***characters*** (по умолчанию пробелы),

с начала, с конца или с обеих сторон (**BOTH**, по умолчанию) строки ***string***.

trim(both 'xyz' from 'yxTomxx') => Tom

ltrim(*string* text [, *characters* text]) => text

Удаляет наибольшую подстроку, содержащую только символы ***characters*** (по умолчанию пробелы),

с начала строки ***string***.

ltrim('zzzytest', 'xyz') => test

rtrim(*string* text [, *characters* text]) => text

Удаляет наибольшую подстроку, содержащую только символы ***characters*** (по умолчанию пробелы),

с конца строки ***string***.

rtrim('testxxzx', 'xyz') => test

replace(*string* text, *from* text, *to* text) => text

Заменяет все вхождения в ***string*** подстроки ***from*** подстрокой ***to***.

replace('abcdefabcdef', 'cd', 'XX') => abXXefabXXef

Функции для оперирования содержимым

```
SELECT  first_name,
        last_name,
        trim('P' from first_name) as "Without P",
        ltrim(last_name, 'B') as "Without B",
        rtrim(last_name, 'l') as "without l"
    FROM employees
   WHERE replace(job_id, 'SA_', '') = 'REP';
```

	first_name character varying (20)	last_name character varying (25)	Without P text	Without B text	without l text
1	Peter	Tucker	eter	Tucker	Tucker
2	David	Bernstein	David	ernstein	Bernstein
3	Peter	Hall	eter	Hall	Ha
4	Christopher	Olsen	Christopher	Olsen	Olsen

Функции для оперирования содержимым

`repeat(string text, number integer) => text` *Повторяет содержимое **`string`** указанное число (**`number`**) раз.* **`repeat('Pg', 4) => PgPgPgPg`**
`reverse(text) => text` *Переставляет символы в строке в обратном порядке.* **`reverse('abcde') => edcba`**
`format(formatstr text [, formatarg "any" [, ...]]) => text` *Форматирует аргументы в соответствии со строкой формата.* *Эта функция работает подобно **`sprintf`** в языке C.* **`format('Hello %s, %1$s', 'World') => Hello World, World`**

Функции для оперирования содержимым

```
SELECT
      first_name,
      last_name,
      job_id,
      repeat(job_id, 2) as double_job,
      reverse(job_id) as reverse_job,
      format('Name: %1$s, Surname: %2$s', first_name, last_name) as fname
FROM employees WHERE replace(job_id, 'SA_', '') = 'REP';
```

	first_name character varying (20)	last_name character varying (25)	job_id character varying (10)	double_job text	reverse_job text	fname text
1	Peter	Tucker	SA_REP	SA_REPSA_REP	PER_AS	Name: Peter, Surname: Tucker
2	David	Bernstein	SA_REP	SA_REPSA_REP	PER_AS	Name: David, Surname: Bernstein
3	Peter	Hall	SA_REP	SA_REPSA_REP	PER_AS	Name: Peter, Surname: Hall

Числовые функции

abs(*числовой_тип*) => *числовой_тип*

Абсолютное значение

abs(-17.4) => 17.4

---

div(*y* numeric, *x* numeric) => numeric

*Целочисленный результат **y/x** (округлённый в направлении нуля)*

div(9, 4) => 2

---

factorial(bigint) => numeric

Факториал

factorial(5) => 120

---

power(*a* numeric, *b* numeric) => numeric

power(*a* double precision, *b* double precision) => double precision

a** возводится в степень **b

power(9, 3) => 729

---

mod(*y числовой_тип*, *x числовой_тип*) => *числовой_тип*

*Остаток от деления **y/x**; имеется для типов **smallint, integer, bigint и numeric***

mod(9, 4) => 1

Числовые функции

```
SELECT employee_id,
       first_name,
       salary,
    abs(50000 - salary) as max_delta,
    div(salary, 21) as div_day,
    factorial(4) as fct
  FROM employees
 WHERE commission_pct is not null
```

	employee_id [PK] integer	first_name character varying (20)	salary numeric (8,2)	max_delta numeric	div_day numeric	fct numeric
1	145	John	14000.00	36000.00	666	24
2	146	Karen	13500.00	36500.00	642	24
3	147	Alberto	12000.00	38000.00	571	24

Числовые функции

`round(v numeric, s integer) => numeric` *Округление v до s десятичных знаков. Половина (.5) округляется до 1 по модулю.* **`round(42.4382, 2) => 42.44`**
`sqrt(numeric) => numeric` **`sqrt(double precision) => double precision`** *Квадратный корень* **`sqrt(2) => 1.4142135623730951`**
`trunc(v numeric, s integer) => numeric` *Округление v до s десятичных знаков* **`trunc(42.4382, 2) => 42.43`**
`random() => double precision` *Возвращает случайное число в диапазоне 0.0 <= x < 1.0* **`random() => 0.897124072839091`**

Числовые функции

```
SELECT employee_id,
     commission_pct,
     sqrt(commission_pct) as square_root_bad,
     round(sqrt(commission_pct)) as square_root_good,
     power(commission_pct, 2) as comm_sq_bad,
     trunc(power(commission_pct, 2)* 100) as comm_sq_good,
     random() as rnd
  FROM employees
 WHERE commission_pct is not null
```

	employee_id [PK] integer	commission_pct numeric (2,2)	square_root_bad numeric	square_root_good numeric	comm_sq_bad numeric	comm_sq_good numeric	rnd double precision
1	145	0.40	0.63245553203367587	1	0.1600000000000000	16	0.5212796452569819
2	146	0.30	0.54772255750516611	1	0.0900000000000000	9	0.042643228304807934
3	147	0.30	0.54772255750516611	1	0.0900000000000000	9	0.4164268484034679
4	148	0.30	0.54772255750516611	1	0.0900000000000000	9	0.9525676033592738

Работа с датами

- Дата хранится во внутреннем числовом формате, содержащим век, год, месяц, день, час, минуту, секунду.
- Формат даты по умолчанию YYYY-MM-DD HH24:mi:ss
- now() это функция, возвращающая:
  - Дату
  - Время

Тип даты и времени

Имя	Размер	Описание	Наименьшее значение	Наибольшее значение	Точность
timestamp [(p)] [without time zone]	8 байт	дата и время (без часового пояса)	4713 до н. э.	294276 н. э.	1 микросекунда
timestamp [(p)] with time zone	8 байт	дата и время (с часовым поясом)	4713 до н. э.	294276 н. э.	1 микросекунда
date	4 байта	дата (без времени суток)	4713 до н. э.	5874897 н. э.	1 день
time [(p)] [without time zone]	8 байт	время суток (без даты)	00:00:00	24:00:00	1 микросекунда
time [(p)] with time zone	12 байт	время дня (без даты), с часовым поясом	00:00:00+1559	24:00:00-1559	1 микросекунда
interval [поля] [(p)]	16 байт	временной интервал	-178000000 лет	178000000 лет	1 микросекунда

Операторы, применимые к датам

`date + integer => date` *Добавляет к дате заданное число дней* **`date '2001-09-28' + 7 => 2001-10-05`**
`date - integer => date` *Вычитает из даты заданное число дней* **`date '2001-10-01' - 7 => 2001-09-24`**
`date - date => integer` *Вычитает даты, выдавая разницу в днях* **`date '2001-10-01' - date '2001-09-28' => 3`**
`date + interval => timestamp` *Добавляет к дате интервал* **`date '2001-09-28' + interval '1 hour' => 2001-09-28 01:00:00`**
`date - interval => timestamp` *Вычитает из даты интервал* **`date '2001-09-28' - interval '1 hour' => 2001-09-27 23:00:00`**

Операторы, применимые к датам

timestamp + interval → timestamp

Добавляет к отметке времени интервал

timestamp '2001-09-28 01:00' + interval '23 hours' → 2001-09-29 00:00:00

timestamp - interval → timestamp

Вычитает из отметки времени интервал

timestamp '2001-09-28 23:00' - interval '23 hours' → 2001-09-28 00:00:00

timestamp - timestamp → interval

Вычитает из одной отметки времени другую (преобразуя 24-часовые интервалы в дни)

timestamp '2001-09-29 03:00' - timestamp '2001-07-27 12:00' => 63 days 15:00:00

Операторы, применимые к датам

time + interval → time *Добавляет к времени интервал* **time '01:00' + interval '3 hours' → 04:00:00**
time - time → interval *Вычитает из одного времени другое* **time '05:00' - time '03:00' → 02:00:00**
time - interval → time *Вычитает из времени интервал* **time '05:00' - interval '2 hours' → 03:00:00**

Операторы, применимые к датам

interval + interval → interval

Складывает интервалы

interval '1 day' + interval '1 hour' → 1 day 01:00:00

interval - interval → interval

Вычитает из одного интервала другой

interval '1 day' - interval '1 hour' → 1 day -01:00:00

interval * double precision → interval

Умножает интервал на скалярное значение

interval '1 second' * 900 → 00:15:00

interval '1 day' * 21 → 21 days

interval '1 hour' * 3.5 → 03:30:00

interval / double precision → interval

Делит интервал на скалярное значение

interval '1 hour' / 1.5 → 00:40:00

Операторы, применимые к датам

```
SELECT first_name,
       hire_date,
    hire_date + 7 as next_week,
    hire_date - 7 as prev_week,
    now()::date - hire_date as experience,
    hire_date + interval '1 hour' as plus_hour,
    hire_date - interval '1 hour' as minus_hour
  FROM employees
```

	first_name character varying (20)	hire_date date	next_week date	prev_week date	experience integer	plus_hour timestamp without time zone	minus_hour timestamp without time zone
1	Steven	2003-06-17	2003-06-24	2003-06-10	6877	2003-06-17 01:00:00	2003-06-16 23:00:00
2	Neena	2005-09-21	2005-09-28	2005-09-14	6050	2005-09-21 01:00:00	2005-09-20 23:00:00
3	Lex	2001-01-13	2001-01-20	2001-01-06	7762	2001-01-13 01:00:00	2001-01-12 23:00:00
4	Alexander	2006-01-03	2006-01-10	2005-12-27	5946	2006-01-03 01:00:00	2006-01-02 23:00:00

Функции, применимые к датам

age(timestamp, timestamp) => interval *Вычитает аргументы и выдаёт результат с годами и месяцами, а не просто днями* **age(timestamp '2001-04-10', timestamp '1957-06-13') => 43 years 9 mons 27 days (43 года 9 месяцев 27 дней)**
date_trunc(text, timestamp) => timestamp *Отсекает компоненты даты до заданной точности* **date_trunc('hour', timestamp '2001-02-16 20:38:40') => 2001-02-16 20:00:00**
date_part(text, timestamp) => double precision *Возвращает поле даты/времени (равнозначно extract)* **date_part('hour', timestamp '2001-02-16 20:38:40') => 20**
extract(*field* from timestamp) => numeric *Возвращает поле даты/времени* **extract(hour from timestamp '2001-02-16 20:38:40') => 20**

Пример

```
SELECT first_name,
       hire_date,
    age(now(), hire_date) as experience,
    extract(year from age(now(), hire_date)) years_exp,
    date_part('month', age(now(), hire_date)) months_exp,
    date_trunc('month', age(now(), hire_date))
  FROM employees
```