

# Шпаргалки базового SQL

## Введение в базы данных

```
-- выбор всех полей в таблице  
SELECT *  
FROM таблица;
```

**База данных** — это хранилище структурированной информации.

**Реляционные базы данных** — базы, в которых данные представлены в виде связанных таблиц.

**СУБД** (система управления базами данных) — это комплекс программ, который позволяет создать базу данных, наполнить её новыми таблицами, отобразить содержимое, редактировать существующие таблицы.

**Таблица** — это совокупность строк и столбцов.

**Поле** — столбец таблицы с характеристикой объекта, уникальным именем и характерным типом данных.

**Запись** — строка таблицы, содержащая информацию об одном объекте.

**Ячейка** — место пересечения строки и столбца.

**Первичный ключ** — поле или группа полей, использующиеся для однозначного определения записи. Все значения первичного ключа уникальны.

**SQL** — язык программирования, предназначенный для управления данными в реляционной базе.

**Запрос** — это сформулированное в соответствии с синтаксисом SQL требование, в котором объявляют, какие данные выбрать, и как именно их обработать.

# Шпаргалки базового SQL

## Срезы данных в SQL

```
-- выбор определённых полей из таблицы  
SELECT  поле_1,  
        поле_2,  
        поле_3 ...  
FROM    таблица;
```

```
-- однострочный комментарий на языке SQL  
/*  
многострочный  
комментарий  
*/
```

```
-- изменение типа данных поля при  
выгрузке  
SELECT  CAST(поле AS тип данных)  
FROM    таблица,
```

```
-- фильтрация данных по условию  
SELECT  поле_1,  
        поле_2 -- выбор полей  
FROM    таблица -- таблица, из которой выгружают данные  
WHERE   условие; -- условие для среза данных
```

```
/*  
фильтрация, в которой значение в поле_1 находится  
между значением_1 и значением_2 включительно  
*/  
SELECT  *  
FROM    таблица  
WHERE   поле_1 BETWEEN значение_1 AND значение_2;
```

## Шпаргалки базового SQL

### Срезы данных в SQL

```
-- фильтрация, в которой все значения поля находятся в
-- списке
SELECT *
FROM таблица
WHERE поле IN ('значение_1', 'значение_2', 'значение_3');
```

```
-- выбор записей с пропусками в поле
SELECT *
FROM таблица
WHERE поле IS NULL;
```

```
-- выбор записей с пропусками в поле
SELECT *
FROM таблица
WHERE поле IS NULL;
```

```
-- выбор записей без пропусков в поле
SELECT *
FROM таблица
WHERE поле IS NOT NULL;
```

```
-- действия в зависимости от условий
CASE
    WHEN условие_1 THEN результат_1
        WHEN условие_2 THEN результат_2
        WHEN условие_3 THEN результат_3
    ELSE результат_4
END;
```

# Шпаргалки базового SQL

## Срезы данных в SQL

```
-- извлечение части даты  
SELECT EXTRACT(часть_даты FROM поле) AS новое_поле_с_датой  
FROM таблица;
```

```
-- усечение даты до части  
SELECT DATE_TRUNC('часть_даты_до_которой_усекаем', поле) AS  
новое_поле_с_датой  
FROM таблица;
```

```
/* используя функции EXTRACT и DATE_TRUNC, не забывайте привести  
поле к типу timestamp, чтобы решить проблему с часовыми поясами  
в PostgreSQL */  
SELECT EXTRACT(MONTH FROM CAST(поле AS timestamp)) AS  
первое_поле_с_датой,  
       DATE_TRUNC ('month', CAST(поле AS timestamp)) AS  
второе_поле_с_датой  
FROM таблица;
```

Параметры функции DATE\_TRUNC :

'microseconds' микросекунды;

'milliseconds' миллисекунды;

'second' секунда;

'minute' минута;

'hour' час;

'day' день;

'week' неделя;

'month' месяц;

# Шпаргалки базового SQL

## Срезы данных в SQL

Параметры функции DATE\_TRUNC:

'quarter'	квартал;
'year'	год;
'decade'	декада года;
'century'	век.

Параметры функции EXTRACT:

CENTURY	век;
DAY	день;
DOY	день года, выраженный числом от 1 до 365 или 366, если год високосный;
DOW	день недели, выраженный числом от 0 до 6, где понедельник — 1, воскресенье — 0.
ISODOW	— день недели, выраженный числом от 1 до 7, где понедельник — 1, воскресенье — 7.
HOUR	час;
MILLISECOND	миллисекунда;
MINUTE	минута;
MONTH	месяц;
SECOND	секунда;
QUARTER	квартал;
WEEK	неделя в году;
YEAR	год;

## Шпаргалки базового SQL

### Агрегирующие функции. Группировка и сортировка данных.

```
SELECT
COUNT(*), -- возвращает число записей в таблице
COUNT(column), -- возвращает число записей в поле column
COUNT(DISTINCT column),
/* возвращает количество уникальных значений
в поле column */
SUM(column), -- сумма значений в поле
AVG(column), -- среднее значений в поле
MIN(column), -- минимум значений в поле
MAX(column) -- максимум значений в поле
FROM таблица;
```

```
-- группировка данных
SELECT поле_1,
       поле_2,
       поле_3,
       АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле)
FROM таблица
GROUP BY поле_1,
         поле_2,
         поле_3;
-- ВАЖНО! Сколько полей без агрегации в SELECT, столько и должно
быть в GROUP BY
```

```
-- сортировка данных
SELECT поле_1,
       поле_2,
       поле_3,
       АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле)
FROM таблица
ORDER BY поле_1, -- сортировка по возрастанию
         поле_2 DESC, -- сортировка по убыванию
         поле_3 ASC; -- сортировка по возрастанию
```

## Шпаргалки базового SQL

### Агрегирующие функции. Группировка и сортировка данных.

```
-- сразу после группировки
SELECT поле_1,
       поле_2,
       поле_3,
       АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле)
FROM таблица
GROUP BY поле_1
HAVING АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле) > n;
-- ВАЖНО! WHERE работает только с изначальными данными,
-- HAVING – только с агрегированными
```