**Памятка SQL ЗАПРОСЫ**

**SELECT** [**DISTINCT**]

<имя поля> as <псевдоним>

[**CASE** WHEN <условие> THEN <выражение>

[ WHEN ... ]

[ ELSE <выражение>]

**END** AS <псевдоним>]

**FROM** <имя таблицы1>[, имя таблицы N] – при указании нескольких таблиц делается полное соединение, декартовое соединение

# создание виртуальной таблицы (можно использовать в разных #местах: from, join)

( **VALUES** (<значение поля 1 записи 1>, <значение поля 2 записи 1>),

(<значение поля 1 записи n>, <значение поля 2 записи n>)

) AS <псевдоним> ( <имя поля 1>, <имя поля 2>)

[**JOIN** <имя таблицы> ON <условие>] – полное соединение с условием

[**CROSS JOIN** <имя таблицы>] – полное соединение

[**LEFT OUTER** **JOIN** <имя таблицы> ON <условие>] – левое соединение с

условием

[**RIGHT OUTER** **JOIN** <имя таблицы> ON <условие>] – левое соединение

с условием

[**FULL OUTER JOIN**  <имя таблицы> ON <условие>] – полное внешнее

соединение с условием

*В этом случае в выборку включаются строки из левой таблицы, для которых не найдено соответствующих строк в правой, и строки из правой таблицы, для которых не найдено соответствующих строк в левой.*

[**WHERE** < условие >]

[**LIMIT** <число первых строк для вывода>]

[**OFFSET** <число первых строк для исключения>]

# операции с выборками как с множествами

[**UNION** [ALL] SELECT <…>] – строка включается в итоговое множество (выборку), если она присутствует хотя бы в одном из них.

[**INTERSECT** [ALL] SELECT <…> ]– строка включается в итоговое множество (выборку), если она присутствует в каждом из них. Более приоритетное связывание. Если нужно поменять приоритет – использовать скобки.

[**EXCEPT** [ALL] SELECT <…> ]– строка включается в итоговое множество (выборку), если она присутствует в первом множестве (выборке), но отсутствует во втором.

[**GROUP BY** <имена полей> **HANING** <УСЛОВИЕ> ]

[**ORDER BY** <имена полей>] [DESC | ASC] ]

**Подзапросы**

SELECT … from (SELECT)

В подзапросе можно ссылаться на таблицу внешнего запроса, в этом случае подзапрос будет выполняться для каждой строки основного запроса.

Могут присутствовать в предложениях SELECT, FROM, WHERE, HAVING, WITH

SELECT a.model,

( SELECT count( \* )

FROM seats s

WHERE s.aircraft\_code = a.aircraft\_code

AND s.fare\_conditions = 'Business'

) AS business,

( SELECT count( \* )

FROM seats s

WHERE s.aircraft\_code = a.aircraft\_code

AND s.fare\_conditions = 'Comfort'

) AS comfort,

( SELECT count( \* )

FROM seats s

WHERE s.aircraft\_code = a.aircraft\_code

AND s.fare\_conditions = 'Economy'

) AS economy

FROM aircrafts a

ORDER BY 1;

Если возвращается одна колонка и одна строка, то можно использовать операторы сравнения (<, >, = b и т.д.)

**IN**

Если возвращается одна колонка и несколько строк, то можно использовать предикат **IN**

SELECT … WHERE <имя поля> **IN** (SELECT …)

можно задавать несколько значений с помощью скалярных подзапросов

SELECT … WHERE <имя поля> **IN** ((SELECT …),(SELECT …))

**NOT IN**

**EXISTS** (SELECT) – проверяет есть ли в результате запроса хоть одна строка

SELECT … WHERE <имя поля> **EXISTS** (SELECT 1 from ….)

**NOT EXISTS**

**Common Table Expression** – созд. временной таблицы для исп. в разн. местах

**WITH** [RECURSIVE] <имя временной таблицы> [(имена полей для временной таблицы)] **AS** (SELECT … )

**Оконные функции**

SELECT <обычные поля запроса>, <агрегатная функция> **OVER ([partition BY <имя поля для определения групп>] [ORDER BY <имя поля сортировки>])**

OVER определяет вычисление агрегатной функции без свертки строк.

PARTITION BY - определяет поле, по значениям которого выявляются группы для вычисления агрегатной функции.

ORDER BY – определяет поля сортировки, для специфического определения дополнительных подгрупп строк (у разных агрегатных функций может отличаться)

Можно использовать только в SELECT B ORDER BY

**Чтобы наложить фильтр на результат оконной функции**, можно выполнить такой запрос как вложенный и потом отфильтровать.

**Встроенные функции**

array\_length(<имя поля типа массив>, номер измерения массива)

extract(‘<часть данных составного типа>’ from <имя поля>) – извлечь из значения составного типа часть данных

extract( 'day' from b.book\_date ) AS day

date\_trunc(‘<значение точности>’ from <имя поля>) – округляет дату до нужного значения точности

date\_trunc( 'month', b.book\_date ) – вернет дату годом и месяцем

string\_agg

**Сравнение строк на подобие**

like <’шаблон>’

специальные символы шаблона

% - любое количество любых символов

\_ - один любой символ

~ <’шаблон’>

<https://postgrespro.ru/docs/enterprise/14/functions-matching#FUNCTIONS-POSIX-REGEXP>

| **Ограничение** | **Описание** |
| --- | --- |
| ^ | соответствует началу строки |
| $ | соответствует концу строки |
| (?=***re***) | *позитивный просмотр вперёд* находит соответствие там, где начинается подстрока, соответствующая ***re*** (только для ARE) |
| (?!***re***) | *негативный просмотр вперёд* находит соответствие там, где не начинается подстрока, соответствующая ***re*** (только для ARE) |
| (?<=***re***) | *позитивный просмотр назад* находит соответствие там, где заканчивается подстрока, соответствующая ***re*** (только для ARE) |
| (?<!***re***) | *негативный просмотр назад* находит соответствие там, где не заканчивается подстрока, соответствующая ***re*** (только для ARE) |

| **Атом** | **Описание** |
| --- | --- |
| (***re***) | (где ***re*** — любое регулярное выражение) описывает соответствие ***re***, при этом данное соответствие захватывается для последующей обработки |
| (?:***re***) | подобно предыдущему, но соответствие не захватывается (т. е. это набор скобок «без захвата») (применимо только к ARE) |
| . | соответствует любому символу |
| [***символы***] | выражение в квадратных скобках, соответствует любому из ***символов*** (подробнее это описано в [Подразделе 9.7.3.2](https://postgrespro.ru/docs/postgresql/14/functions-matching#POSIX-BRACKET-EXPRESSIONS)) |
| \***k*** | (где ***k*** — не алфавитно-цифровой символ) соответствует обычному символу буквально, т. е. \\ соответствует обратной косой черте |
| \***c*** | где ***c*** — алфавитно-цифровой символ (за которым могут следовать другие символы), это спецсимвол, см. [Подраздел 9.7.3.3](https://postgrespro.ru/docs/postgresql/14/functions-matching#POSIX-ESCAPE-SEQUENCES) (применим только к ARE; в ERE и BRE этому атому соответствует ***c***) |
| { | когда за этим символом следует любой символ, кроме цифры, этот атом соответствует левой фигурной скобке ({), если же за ним следует цифра, это обозначает начало ***границы*** (см. ниже) |
| ***x*** | (где ***x*** — один символ, не имеющий специального значения) соответствует этому символу |

| **Определитель** | **Соответствует** |
| --- | --- |
| \* | 0 или более вхождений атома |
| + | 1 или более вхождений атома |
| ? | 0 или 1 вхождение атома |
| {***m***} | ровно ***m*** вхождений атома |
| {***m***,} | ***m*** или более вхождений атома |
| {***m***,***n***} | от ***m*** до ***n*** (включая границы) вхождений атома; ***m*** не может быть больше ***n*** |
| \*? | не жадная версия \* |
| +? | не жадная версия + |
| ?? | не жадная версия ? |
| {***m***}? | не жадная версия {***m***} |
| {***m***,}? | не жадная версия {***m***,} |
| {***m***,***n***}? | не жадная версия {***m***,***n***} |