SQL

1. Почитать про типы данных в postgres
2. Rf выполняется запрос
3. Distinct
4. Like

Чтобы поддерживать максимальную гибкость, строки таблицы, по определению, не должны находиться в каком-то определенном порядке. С этой точки зрения структура БД отличается от нашей адресной книги.

Последовательность выполнения SQL запроса:

1. FROM (выбор таблицы)
2. JOIN (комбинация с подходящими по условию данными из других таблиц)
3. WHERE (фильтрация строк)
4. GROUP BY (агрегирование данных)
5. HAVING (фильтрация агрегированных данных)
6. SELECT (возврат результирующего датасета)
7. ORDER BY (сортировка).
8. **РЕЛЯЦИОННЫЕ ОПЕРАЦИИ**
9. Реляционная операция - математический символ, который указывает на определённый тип сравнения двух значений. Вы уже видели, как используются равенства, такие как 2 + 3 = 5 или city = "London". Но имеются также и другие реляционные операции. Предположим, что вы хотите видеть всех Продавцов с их комиссионными, выше определенного значения. Вы можете использовать тип сравнения "больше чем" (>). Вот реляционные операции (операции сравнения), которыми располагает SQL:
10. =
11. Равно
12. >
13. Больше
14. <
15. Меньше
16. >=
17. Больше или равно
18. <=
19. Меньше или равно
20. <>
21. Не равно
22. Эти операции имеют стандартное значение для чисел. Для символов их определение зависит от формата преобразования, ASCII или EBCDIC, который вы используете.
23. SQL сравнивает символьные значения в терминах основных чисел, как определено в формате преобразования. Даже значение символа, такого как "1", который представляет число, не обязательно равняется числу, которое он представляет. Вы можете использовать реляционные операции, чтобы установить алфавитный порядок, например, "a" < "n", где a идёт раньше в алфавитном порядке, но всё это ограничивается с помощью параметра преобразования формата.
24. И в ASCII, и в EBCDIC символы сортируются по значению: символ имеет значение меньше, чем все другие символы, которым он предшествует в алфавитном порядке и которые имеют с ним один вариант регистра (верхний или нижний). В ASCII все символы верхнего регистра меньше, чем все символы нижнего регистра, поэтому "Z" < "a", а все числа - меньше чем все символы, поэтому "1" < "Z". То же относится и к EBCDIC.
25. Чтобы сохранить обсуждение более простым, мы допустим, что вы будете использовать текстовый формат ASCII. Проконсультируйтесь в документации вашей системы, если вы не уверены, какой формат вы используете или как он работает.
26. Значения, сравниваемые здесь, называются скалярными значениями. Скалярные значения производятся скалярными выражениями; 1 + 2 это скалярное выражение, которое производит скалярное значение 3. Скалярное значение может быть символом или числом, хотя очевидно, что только числа используются с арифметическими операциями, такими как + (сложение) или \* (умножение).
27. Предикаты обычно сравнивают значения скалярных величин, используя реляционные операции или специальные операции SQL, чтобы увидеть, верно ли это сравнение. Некоторые операции/операторы SQL описаны в [Главе 5](https://www.opennet.ru/docs/RUS/rusql/ch5.html).
28. Предположим, что вы хотите увидеть всех заказчиков с оценкой (rating) выше 200. Так как 200 - скалярное значение, как и значение в столбце оценки, для их сравнения вы можете использовать реляционную операцию.
29. SELECT \*
30. FROM Customers
31. WHERE rating > 200;
32. Вывод для этого запроса показан на Рисунке 4.1.
33. Конечно, если бы мы захотели увидеть ещё и заказчиков с оценкой, равной 200, мы использовали бы предикат
34. rating > = 200

Используется в предложении where для поиска строк с заданным шаблоном

Предикат Like для текстовых значений

% - строка любой длины

[^] - исключающий диапазон или последовательность символов

[ ] – одиночный символ из набора символов или диапазона [a-z]

Пример : not like ‘%[0-9]%’ – не содержит символов из диапазона 0-9

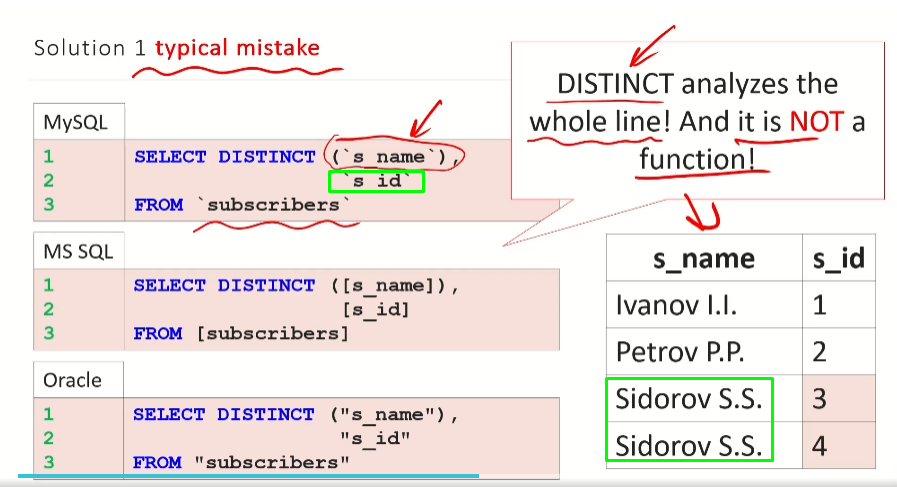
Символы \_ и % можно экранировать с помощью escape

Если искомая строка содержит трафаретный символ, то следует задать управляющий символ в предложении **ESCAPE**. Этот управляющий символ должен использоваться в образце перед трафаретным символом, сообщая о том, что последний следует трактовать как обычный символ. Например, если в некотором поле следует отыскать все значения, содержащие символ «\_», то шаблон ‘%\_%’ приведет к тому, что будут возвращены все записи из таблицы. В данном случае шаблон следует записать следующим образом:

ДОБАВИТЬ ПРИМЕР!!!

SQL DISTINCT

Важно помнить, что с использованием distinct идет анализ всей строки!

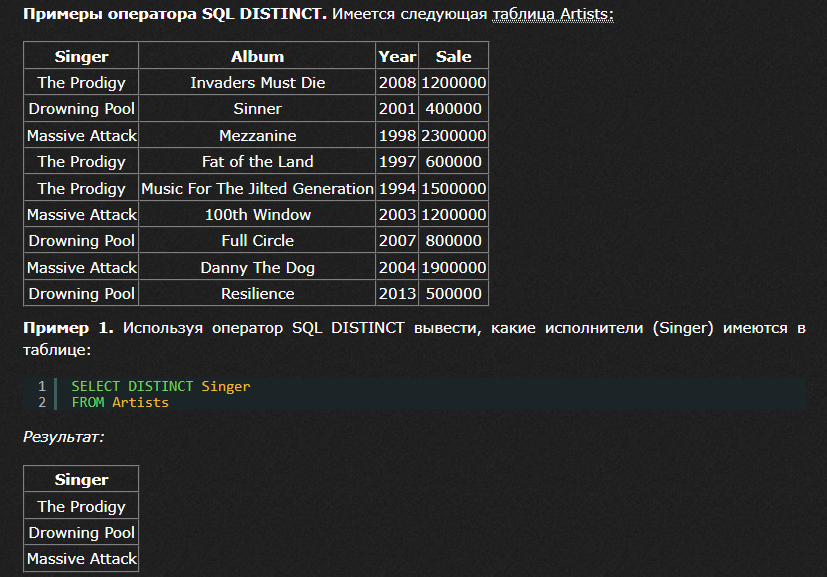


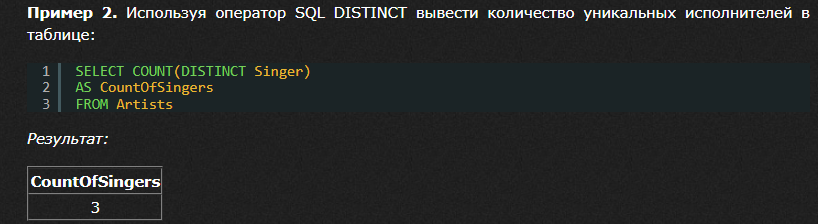
Для выборки уникальных значений из таблицы

Используется также с агрегатными функциями

ПРИМЕРЫ!

SELECT DISTINCT column\_name FROM table\_name





Следующая инструкция SQL перечисляет количество различных (отдельных) стран-клиентов:

Пример

SELECT COUNT(DISTINCT Country)

FROM Customers;

Функция COUNT может быть использована в следующих пяти форматах:

• COUNT(\*) — классический вариант, используемый для подсчёта количества записей;

• COUNT(1) — альтернативная запись классического варианта;

• COUNT(первичный\_ключ) — альтернативная запись классического варианта;

• COUNT(поле) — подсчёт записей, в указанном поле которых нет NULLзначений;

• COUNT(DISTINCT поле) — подсчёт без повторения записей, в указанном поле которых нет NULL-значений.

Предикат **BETWEEN** проверяет, попадают ли значения проверяемого выражения в диапазон, задаваемый пограничными выражениями, соединяемыми служебным словом **AND**. Естественно, как и для предиката сравнения, выражения в предикате **BETWEEN** должны быть совместимы по типам.

BETWEEN…AND равносилен предикату

1. exp1 >= exp2 AND exp1 <= exp3

А предикат

1. exp1 NOT BETWEEN exp2 AND exp3

равносилен предикату

1. NOT (exp1 BETWEEN exp2 AND exp3)

Если значение предиката *exp1 BETWEEN exp2 AND exp3* равно TRUE, в общем случае это отнюдь не означает, что значение предиката *exp1 BETWEEN exp3 AND exp2* тоже будет TRUE, так как первый будет интерпретироваться как предикат:

1. exp1 >= exp2 AND exp1 <= exp3

а второй как:

1. exp1 >= exp3 AND exp1 <= exp2

# **SQL NULL Значения**

IS NULL

IS NOT NULL

Сравнение с NULL дает результат unknown, а строка попадает в результирующий набор, если предикат true.

Аналогично для HAVING.