Учебно-методическое пособие по теме:

«Реализация запросов и представлений по заданным параметрам в PostgreSQL»

Цель: Рассмотрение реализации запросов к базе данных.

Содержание:

Предложение WHERE

Предложение WHERE записывается так:

WHERE условие_ограничения

где условие_ограничения — любое выражение значения, выдающее результат типа boolean.

После обработки предложения FROM каждая строка полученной виртуальной таблицы проходит проверку по условию ограничения. Если результат условия равен true, эта строка остаётся в выходной таблице, а иначе (если результат равен false или NULL) отбрасывается. В условии ограничения, как правило, задействуется минимум один столбец из таблицы, полученной на выходе FROM. Хотя строго говоря, это не требуется, но в противном случае предложение WHERE будет бессмысленным.

Пример запроса с WHERE:

```
SELECT … FROM tabl1 WHERE name = Alex
```

tabl1 – название таблицы откуда происходит выборка, т.е. строки где колонка name не равна Alex исключаются из blabla и не выводятся на экран.

Метки столбцов:

Элементам в списке выборки можно назначить имена для последующей обработки, например, для указания в предложении ORDER BY. Например:

```
SELECT A AS value, b + c AS sum FROM …
```

Если выходное имя столбца не определено (с помощью AS), система назначает имя сама. Для простых ссылок на столбцы этим именем становится имя целевого столбца, а для вызовов функций это имя функции. Для сложных выражений система генерирует некоторое подходящее имя.

Слово AS можно опустить, но только если имя нового столбца не является ключевым словом PostgreSQL . Во избежание случайного совпадения имени с ключевым словом это имя можно заключить в кавычки. Например, VALUE — ключевое слово, поэтому такой вариант не будет работать:

```
SELECT a  value, b + c AS sum FROM …
```

А такой будет:

```
SELECT a  “value”, b + c AS sum FROM …
```

Для предотвращения конфликта с ключевыми словами, которые могут появиться в будущем, рекомендуется всегда писать AS или заключать метки выходных столбцов в кавычки.

DISTINCT

После обработки списка выборки в результирующей таблице можно дополнительно исключить дублирующиеся строки. Для этого сразу после SELECT добавляется ключевое слово DISTINCT:

```
SELECT DISTINCT список_выборки …
```

(Чтобы явно включить поведение по умолчанию, когда возвращаются все строки, вместо DISTINCT можно указать ключевое слово ALL.)

Две строки считаются разными, если они содержат различные значения минимум в одном столбце. При этом значения NULL полагаются равными.

Предложение DISTINCT ON не описано в стандарте SQL и иногда его применение считается плохим стилем из-за возможной неопределённости в результатах. При разумном использовании GROUP BY и подзапросов во FROMможно обойтись без этой конструкции, но часто она бывает удобнее.

ПРЕДСТАВЛЕНИЕ (VIEW) объект данных который не содержит никаких данных его владельца. Это - тип таблицы, чье содержание выбирается из других таблиц с помощью выполнения запроса. Поскольку значения в этих таблицах меняются, то автоматически, их значения могут быть показаны представлением. Использование представлений основанных на улучшенных средствах запросов,

таких как объединение и подзапрос, разработанных очень тщательно, в некоторых случаях даст больший выигрыш по сравнению с запросами.

Представление - это фактически запрос, который выполняется всякий раз, когда представление становится темой команды. Вывод запроса при этом в каждый момент становится содержанием представления.

КОМАНДА CREATE VIEW:

Вы создаете представление командой CREATE VIEW. Она состоит из слов CREATE VIEW (создать представление), имени представления которое нужно создать, слова AS (как), и далее запроса, как в следующем примере:

```
CREATE VIEW Londonstaff
          AS SELECT *
          FROM Salespeople
          WHERE city = 'London';
```

Теперь мы имеем представление, называемое Londonstaff. Вы можете использовать это представление точно так же как и любую другую таблицу. Она может быть запрошена, модифицирована, вставлена в, удалена из, и соединена с, другими таблицами и представлениями.

```
               SELECT *
               FROM Londonstaff;


         ===============  SQL Execution Log ============
        |                                               |
        | SELECT *                                      |
        | FROM  Londonstaff;                            |
        |                                               |
        | ==============================================|
        |   snum      sname         city         comm   |
        | ------    ----------   -----------   -------  |
        |   1001      Peel         London       0.1200  |
        |   1004      Motika       London       0.1100  |
        |                                               |
         ===============================================
```

Когда вы приказываете SQL выбрать(SELECT) все строки (*) из представления, он выполняет запрос содержащий в определении - Loncfonstaff, и возвращает все из его вывода.

Представления значительно расширяют управление вашими данными. Это - превосходный способ дать публичный доступ к некоторой, но не всей информации в таблице.

ГРУППОВЫЕ ПРЕДСТАВЛЕНИЯ:

Групповые представления - это представления, который содержит предложение GROUP BY, или который основывается на других групповых представлениях. Групповые представления могут стать превосходным способом обрабатывать полученную информацию непрерывно. Предположим, что каждый день вы должны следить за порядком номеров заказчиков, номерами продавцов принимающих порядки, номерами порядков, средним от порядков, и общей суммой приобретений в порядках.

Чем конструировать каждый раз сложный запрос, можно просто создать следующее представление:

```
CREATE VIEW Totalforday
       AS SELECT odate, COUNT (DISTINCT cnum), COUNT
                (DISTINCT snum), COUNT (onum), AVG
                (amt), SUM (amt)
                FROM Orders
       GROUP BY odate;
```

Теперь можно увидеть всю эту информацию с помощью простого запроса:

```
SELECT *
   FROM Totalforday;
```

SQL запросы могут дать полный комплекс возможностей, так что представления обеспечивают чрезвычайно гибким и мощным инструментом чтобы определить точно, как наши данные могут быть использованы.[1]

Предложение GROUP BY группирует строки таблицы, объединяя их в одну группу при совпадении значений во всех перечисленных столбцах. Порядок, в котором указаны столбцы, не имеет значения. В результате наборы строк с

одинаковыми значениями преобразуются в отдельные строки, представляющие все строки группы. Это может быть полезно для устранения избыточности выходных данных и/или для вычисления агрегатных функций, применённых к этим группам. Например:

```
=> SELECT * FROM test1;
 x | y
---+---
 a | 3
 c | 2
 b | 5
 a | 1
(4 rows)

=> SELECT x FROM test1 GROUP BY x;
 x
---
 a
 b
 c
```

Во втором запросе мы не могли написать SELECT * FROM test1 GROUP BY x, так как для столбца y нет единого значения, связанного с каждой группой. Однако столбцы, по которым выполняется группировка, можно использовать в списке выборки, так как они имеют единственное значение в каждой группе.

INNER JOIN

Это наиболее часто используемое в SQL соединение. Оно возвращает пересечение двух множеств. В терминах таблиц, оно возвращает только записи из обеих таблиц, отвечающие указанному критерию. Результат операции – закрашенная область. (Рисунок 1)

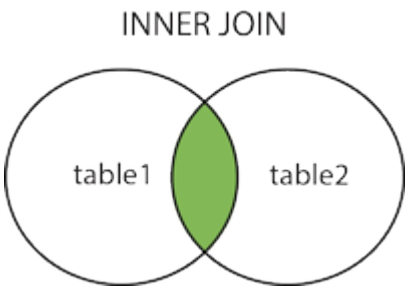

Рисунок 1

FULL JOIN

Полностью это соединение называется FULL OUTER JOIN (зарезервированное слово OUTER необязательно). FULL JOIN работает как объединение двух множеств. (Рисунок 2)

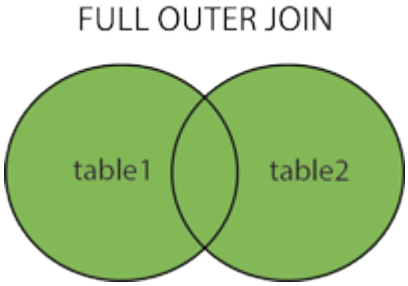

Рисунок 2

LEFT JOIN

Также известен как LEFT OUTER JOIN, и является частным случаем FULL JOIN. Дает все запрошенные данные из таблицы в левой части JOIN плюс данные из правой таблицы, пересекающиеся с первой таблицей. (Рисунок 3)

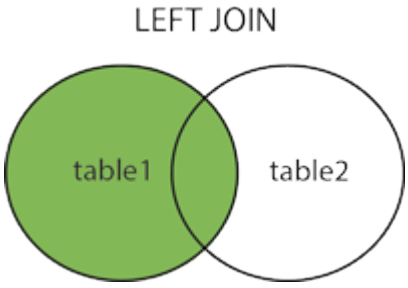

Рисунок 3

RIGHT JOIN

Также известен как RIGHT OUTER JOIN, и является еще одним частным случаем FULL JOIN. Он выдает все запрошенные данные из таблицы, стоящей в

правой части оператора JOIN, плюс данные из левой таблицы, пересекающиеся с правой. (Рисунок 4) [2]

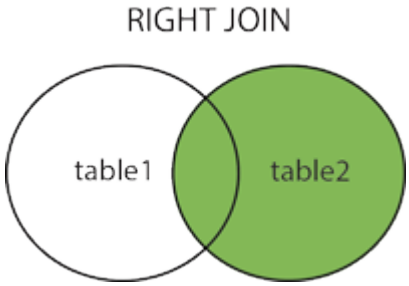

Рисунок 4

Агрегатное выражение представляет собой применение агрегатной функции к строкам, выбранным запросом. Агрегатная функция сводит множество входных значений к одному выходному, как например, сумма или среднее. Агрегатное выражение может записываться следующим образом:

```
агрегатная_функция (выражение [ , ... ] [ предложение_order_by ] ) [ FILTER
( WHERE условие_фильтра ) ]

агрегатная_функция (ALL выражение [ , ... ] [ предложение_order_by ] ) [ FI
LTER ( WHERE условие_фильтра ) ]

агрегатная_функция (DISTINCT выражение [ , ... ] [ предложение_order_by ] )
[ FILTER ( WHERE условие_фильтра ) ]

агрегатная_функция ( * ) [ FILTER ( WHERE условие_фильтра ) ]

агрегатная_функция ( [ выражение [ , ... ] ] ) WITHIN GROUP ( предложение_о
rder_by ) [ FILTER ( WHERE условие_фильтра ) ]
```

Здесь агрегатная_функция — имя ранее определённой агрегатной функции (возможно, дополненное именем схемы), выражение — любое выражение значения, не содержащее в себе агрегатного выражения или вызова оконной функции.

В первой форме агрегатного выражения агрегатная функция вызывается для каждой строки. Вторая форма эквивалентна первой, так как указание ALL подразумевается по умолчанию. В третьей форме агрегатная функция вызывается для всех различных значений выражения (или набора различных значений, для нескольких выражений), выделенных во входных данных. В четвёртой

форме агрегатная функция вызывается для каждой строки, так как никакого конкретного значения не указано (обычно это имеет смысл только для функции count(*)). В последней форме используются сортирующие агрегатные функции, которые будут описаны ниже.

Большинство агрегатных функций игнорируют значения NULL, так что строки, для которых выражения выдают одно или несколько значений NULL, отбрасываются. Это можно считать истинным для всех встроенных операторов, если явно не говорится об обратном.

Например, count(*) подсчитает общее количество строк, а count(f1) только количество строк, в которых f1 не NULL (так как count игнорирует NULL), а count(distinct f1) подсчитает число различных и отличных от NULL значений колонки f1.[3]

Задания для выполнения:

Предметная область – машины
Выборки:

- Выбрать машины у которых закончился срок действия техосмотра.
- Выбрать водителей, у которых машина моложе двух лет и расположить в порядке убывания
- Выбрать группу из 10 негабаритных машин разных марок и годов выпуска
- Выбрать машины, у которых не истёк срок прохождения техосмотра на 2019 год

Представления:

- Марка, пробег, габариты, техосмотр, водителей, машин 2010 года выпуска.
- Определить среднюю цену машины 1995 года выпуска, пробег, габариты, марка, принадлежность организации.
- Год, распределение машин по пройденному техосмотру в этом году, если техосмотр закончился ранее, то не учитывать эти машины.
- Выписать все машины марки «BMW» выпущенных с 2002 по 2019 год в порядке возрастания.

Предметная область – Книжный магазин

Выборки:

- Определить для каждого автора те случаи, когда произведения не переводились на другие языки
- Вывести автора книги и жанр произведения которые разошлись тиражом более 7000 экземпляров

- Определить количество произведений «Война и мир» их общую стоимость и имя издательства.
- Вывести список автором, количество их произведений, язык написания оригинала

Представления

- Творческое направление, количество человек , количество книг, стоимость которых превышает 200 рублей
- Тираж направление автор страна – только для произведений, которые переведены на 10 языков
- Страна , количество издательств которые опубликовались в этой стране общая стоимость всего литературного наследия
- Направление, количество авторов, для которых это направление не было основным.

Предметная область – музыка

Выборки

- Определить количество исполнителей, которые заказывали текст песен у одного автора
- Определить количество музыкальных произведений исполненных группой «дзидзьо» и общие сборы за период существования группы
- Выбрать исполнителей которые выпустили свои последние песни в один день
- Выбрать названия групп в которых иполнителем был «merlin manson»

Представления

- Группы, количество исполнителей работающий в жанре «рок» сборы которых превышали 100000 долларов
- Выбрать группу из 10 самых успешных менеджеров (менеджеров с группой у которой самые большие сборы)
- Выбрать группу которая выступила более чем на 30 сценах в одной стране
- Выбрать названия групп которые выступили на сцене «Славянский базар» в период с 2017 по 2018

Предметная область – Кинотеатр

Выборки

- определить количество фильмом фёдора бондарчука и их общие кассовые сборы
- определить самый успешный фильм по количеству сборов за 2018 год
- определить какой продюссер учавствовал в создании больше всего фильмом за последние 10 лет, вывести названия фильмов

- вывести список фильмов с самым длительным временем проката

Представления

- Автор сценария название фильма у которого кассовые сборы превышают 2млн долларов
- Продюскер, название фильма в котором актёр «_» сыгравший больше всего ролей
- Дата премьеры время проката фильма со сборами менее 30000 юаней
- Актёр, название фильмов , кассовые сборы жанры фильмов только для фильмов в которых играл «киану ривз»

Список использованных источников:

1. Postgres Pro Standart: Документация [Электронный ресурс] – 22.05.2019 – Режим доступа: https://postgrespro.ru/docs/postgrespro

2. Множества [Электронный ресурс] – 22.05.2019 – Режим доступа: http://www.k-press.ru/cs/2009/3/join/join.asp

3. Postgres Pro Standart: Документация . [Электронный ресурс] – 22.05.2019 – Режим доступа: https://postgrespro.ru/docs/postgresql/9.4/sql-expressions

Источники:

1. Нормализация отношений. Шесть нормальных форм

https://habr.com/ru/post/254773/

2. НФ Бойса-Кодда: Основные определения и правила преобразования.

https://sites.google.com/site/gosyvmkss12/bazy-dannyh/24-nf-bojsa-kodda-osnovnye-opredelenia-i-pravila-preobrazovania

4.