Лабораторная работа №5
Индексы и язык запросов SQL

Цель работы: Изучение индексов в PostgreSQL и их влияния на скорость поиска информации в базе данных. Изучение команды *EXPLAIN*. Получение навыков по работе с командой *SELECT* в среде PostgreSQL .

Задание. Общая часть: Ознакомиться с теоретическими сведениями о возможностях создания индексов и ограничений в среде PostgreSQL. С помощью функции *add_n(int)* добавить в какую-либо из таблиц, созданных в лабораторной работе №1, 1000 записей (таблица не должно содержать полей типа *date*, но должно быть хотя бы одно поле типа *char*). Выполнить команду *EXPLAIN ANALYSE select * from таблица where поле(условие)значение.* Здесь *таблица* – имя таблицы, в которую добавлялись записи, *поле* – имя поля, по которому осуществляется выборка, *условие* – поочередно каждое из условий =, <, >, between, *значение* – значение поля, являющееся критерием выборки. Проанализировать полученную после выполнения операции информацию. Создать индекс типа *BTREE* для поля, по которому ранее осуществлялась выборка. Выполнить команду *EXPLAIN ANALYSE* с теми же условиями, что и без индекса, проанализировать полученную после выполнения операции информацию, сравнить результаты выполнения операции по каждому из условий выборки. Удалить индекс. Создать индекс типа *HASH* для того же поля. Выполнить команду *EXPLAIN ANALYSE* с теми же условиями, что и ранее, проанализировать полученную после выполнения операции информацию, сравнить результаты выполнения операции по каждому из условий выборки. Удалить индекс. Выполнить команду *EXPLAIN ANALYSE select * from таблица where upper(поле)(условие)значение, условия* те же, что и ранее, *значение* вводится большими буквами, *поле* должно быть типа *char*. По полю типа *char* той же таблицы создать функциональный индекс с функцией *upper(поле)*. Выполнить команду *EXPLAIN ANALYSE* с теми же

условиями, что и ранее, проанализировать полученную после выполнения операции информацию, сравнить результаты выполнения операции по каждому из условий выборки с информацией, полученной при выборке по этому полю без индекса. Наложить на выбранные поля идентификаторов в таблицах ограничение уникальности. Наложить ограничения согласно своему варианту. Проверить работоспособность ограничений путем добавления в таблицы данных, удовлетворяющих и не удовлетворяющих условиям ограничений. Ознакомиться с теоретическими сведениями о способах выборки данных из базы командой *SELECT*. Выполнить запросы согласно своему варианту. Если в базе нет данных, удовлетворяющих условиям запроса, изменить условия, либо добавить подходящие данные. Просмотреть и проанализировать полученную в результате выполнения операций информацию. Сохранить базу в файл для использования в следующих работах.

Для заполнения таблиц создать функцию:

```
create function add_n(int) returns char
as 'declare t int; begin
select max(идентификатор) into t from таблица;
for k in (t+1)..($1+t+1) loop
insert into таблица(...) values(k, round(random()*10^12) || " " || round(random()*10^5), ...);
end loop; return "Done!";
end;' language 'plpgsql';
```

Здесь *идентификатор* – название поля идентификатора в таблице, в которую будет производиться вставка, *таблица* – название таблицы. Оператор *values(i,....)* должен содержать столько значений, сколько колонок в таблице.

Вариант 1: Найти всех нападающих ростом выше 185 см, премия которых составляет более 15% и набравших более 15 очков (по системе

гол+пас). Найти для всех игроков их заработок (с учетом премий и штрафов) на текущий момент. Найти всех недисциплинированных игроков (штрафное время более 35% от игрового), вывести все данные о них. Найти всех игроков, набравших менее 5 очков (по системе гол+пас) в возрасте до 20 лет с правым хватом клюшки. Найти всех нападающих и полузащитников из Канады и США, игравших на стадионах «Спартак-арена» и «Ростов-арена».

Вариант 2: Найти все товары, прибыль от продажи которых более 10%. Вывести полную информацию о проданных товарах заданного наименования. Найти все товары заданного поставщика, количество которых на складе более 25 кг. Найти все товары, поступившие из заданного города, за последние полгода. Найти все молочные товары и кондитерские изделия из Москвы и Екатеринбурга, производителей «Сибирь-агро» и «Кондитер».

Вариант 3: Вывести полную информацию о водителях и их заработке за последнюю неделю. Найти всех водителей моложе 25 лет, чей заработок за последние сутки составил более 1000 р. Найти всех водителей, получивших права в 18 лет. Найти всех водителей, состоящих в браке. Найти всех водителей со стажем работы от 15 до 20 лет на автомобилях марки «Тойота» и «Форд», приехавших из Красноярска и Кемерово.

Вариант 4: Вывести полную информацию о произведениях, в которых более 300 страниц. Найти все дороже 1000 р. заданного автора. Найти всю литературу заданного издательства, выпущенную за последние три месяца. Найти все книги, чей тираж более 500 экземпляров для жанра «мелодрама». Найти рассказы и новеллы жанров «фантастика» и «комедия» издательств «Сибирь» и «Наука».

Вариант 5: Найти все автомобили, собранные в заданной стране. Вывести полную информацию о машинах-малолитражках (объем двигателя менее 1300 $см^3$), чья стоимость не превышает 600000 р. Найти все полноприводные автомобили Японского производства, выпущенные за последние полгода. Найти все машины типа универсал с левым рулем,

собранные не в России. Найти все машины типа универсал и микроавтобус из Китая и Канады производителей «AVTO» и «CAR».

Вариант 6: Вывести полную информацию о яблоках, проданных за последние 3 дня. Найти все фрукты и овощи заданного поставщика, вес которых на складе более 70 кг. Вывести всю информацию о помидорах на складе, чья общая стоимость (вес*цена) более 18000 р. Найти все товары первого сорта для поставщиков из Москвы. Найти все огурцы и перцы поставщиков «Сибирь-агро» и «Томск-агро», чей срок хранения от 10 до 15 дней.

Вариант 7: Найти все журналы, чье производство оплачено безналичным расчетом, и заказ составляет более 300 экземпляров. Вывести всю информацию о продукции, прибыль от производства которой составляет более 15000 р. Найти продукцию, которую необходимо произвести в ближайший месяц в количестве до 500 экземпляров, с безналичным расчетом. Найти всю продукцию заданного заказчика из Москвы. Найти все газеты и журналы, тираж которых от 500 до 1000 экземпляров, выпущенные издательствами «Наука» и «БХВ-Петербург».

Вариант 8: Вывести информацию обо всех ремонтах, выполненных за последние полгода мастерами с опытом работы от 3 лет. Найти всех мастеров в возрасте до 35 лет, которые занимаются заданным видом ремонта. Найти все виды ремонта со стоимостью от 500 руб. Найти все ремонтные работы, оплаченные за наличный расчет. Найти все работы типа «Электрика» и «Ремонт ограды», выполненные мастерами из Красноярска и Кемерово, ремонт которых оплачен частично путем безналичного расчета.

Вариант 9: Вывести информацию обо всех мастерах в возрасте старше 50 лет, заработавших за последний день более 700 р. Найти всех мастеров в возрасте до 30 лет, которые занимаются ремонтом заданной техники. Найти всех мастеров, чья работа стоит более 500 р. Найти все отремонтированные предметы, выполненные мастерами из Москвы и оплаченные путем безналичного расчета. Найти всех мастеров из Красноярска и Кемерово,

которые занимаются ремонтом техники вида «холодильник» и «пылесос», и со стажем работы более 10 лет.

Вариант 10: Вывести информацию обо всех руководителей туров, выполняющие туры типа «авиа» в город Москву. Найти все туры заданной длительности и стоимостью от 5000 руб. до 25000 руб. Найти все автобусные туры, которые начинаются в ближайший месяц. Найти все железнодорожные туры, осуществленные руководителями туров со стажем работы от 5 до 10 лет. Найти все туры типа «авиа» и «автобусные» в страны Польша и Германия для групп с количеством более 20 человек.

Содержание отчета

Отчет должен содержать титульный лист, цель работы, задание, коды команд создания индексов и ограничений, результаты команды EXPLAIN по всем типам индексов (скриншоты), тексты запросов и их результаты (скриншоты), выводы и анализ результатов работы.

Использование индексов в СУБД PostgreSQL

Индексом называется объект базы данных, позволяющий значительно повысить скорость обращения к базе за счет ускоренной обработки команд, содержащих сравнительные критерии. Хранимая в индексах информация о размещении данных по одному или нескольким полям таблицы повышает эффективность поиска записей при условной выборке (например, с использованием секции *WHERE*).

Индексы ускоряют доступ к базе, но они также сопряжены с дополнительными затратами ресурсов. При изменении данных поле индекс приходится обновлять, поэтому поддержание редко используемых индексов отрицательно сказывается на быстродействии системы (затраты времени на

поддержание индексов превышают экономию от их использования). Как правило, индексы определяются только для полей, часто указываемых в условиях поиска.

Индексы создаются командой SQL *CREATE INDEX*. Синтаксис команды:

CREATE [UNIQUE] INDEX индекс ON таблица

[USING тип] (поле [класс] [...])

Здесь *индекс* - имя создаваемого индекса, *таблица* - имя таблицы, для которой строится индекс, а *поле* - имя индексируемого поля. Необязательный параметр *тип* позволяет выбрать нужную реализацию индекса, а параметр *класс* описывает операторный класс, используемый для сортировки входных данных.

Команда может содержать список из нескольких индексируемых полей, разделенных запятыми, в этом случае индекс строится для всех перечисленных полей.

Создание индекса с ключевым словом *UNIQUE* говорит о том, что индекс является уникальным, то есть индексируемое поле (или поля) не может содержать повторяющихся значений.

Реализация R-дерева, основанная на квадратичном разбиении по алгоритму Гуттмана, применяется главным образом при операциях с геометрическими типами данных. Реализация хэша основана на алгоритмах линейного хэширования Литвина, которые традиционно используются для индексов с частой проверкой равенства (тесть ориентированы на оператор =).

Как сказано выше, реализацию на основе R-дерева рекомендуется использовать для индексации геометрических типов данных, но при этом необходимо помнить о специфике этого типа. Например, для R-дерева нельзя построить уникальный индекс или провести индексацию по нескольким полям. В таких случаях лучше положиться на реализацию B-дерева, обладающую более широкими возможностями.

Тип индекса задается в секции *USING* при помощи ключевых слов *BTREE, RTREE* и *HASH*. По умолчанию используется тип *BTREE*.

В слегка измененном виде команда *CREATE INDEX* позволяет индексировать данные не по значениям поля, а по некоторой функции этих значений. Такая форма индекса называется функциональной.

Команда создания функционального индекса имеет следующий синтаксис:

CREATE [UNIQUE] INDEX индекс ON таблице

[USING тип] (функция (поле [...]) [класс])

Единственное отличие этой команды от описанной выше заключается в том, что индекс строится по результатам применения функции к каждому значению поля. Остальные элементы те же.

Функциональные индексы часто строятся для полей, значения которых проходят предварительную обработку перед сравнением в команде SQL. Например, при сравнении строковых данных без учета регистра символов часто используется функция upper(). Создание функционального индекса с функцией upper() улучшает эффективность таких сравнений.

Команда *CLUSTER* производит кластеризацию таблицы по заданному индексу. Таблица должна существовать, индекс также должен существовать и относиться к указанной таблице.

В процессе кластеризации содержимое таблицы упорядочивается так, чтобы повысить эффективность поиска по конкретному индексу. Кластеризация является «разовой» операцией; чтобы провести повторную кластеризацию, команду *CLUSTER* необходимо выполнить заново.

Во время кластеризации записи таблицы копируются во временную таблицу в порядке, определяемом заданным индексом, после чего временная таблица переименовывается в исходную. Таким образом, кластеризация приводит к уничтожению остальных индексов. Кластеризацию рекомендуется проводить только для таблиц, практически никогда не меняющихся и содержащих большое число записей.

Синтаксис:

CLUSTER индекс ON таблица

Параметры:

индекс - имя индекса, используемого при кластеризации. *таблица* - имя таблицы, для которой производится кластеризация[6].

Для удаления индексов из таблицы используется команда *DROP INDEX*. Синтаксис команды *DROP INDEX*:

DROP INDEX индекс [...]

В качестве параметра команде передается имя удаляемого индекса. Допускается одновременное удаление нескольких индексов, перечисленных через запятую[2].

Для отображения времени выполнения запросов и других действий над базой данных нужно ввести команду *\timing*. После выполнения данной команды после каждой последующей операции будет показываться время ее выполнения[7].

В PostgreSQL предусмотрено несколько вариантов ограничения данных, участвующих в операциях вставки и обновления. Один из них заключается в установке ограничений для таблиц и полей.

Команда EXPLAIN

Команда EXPLAIN отображает план выполнения запроса, сгенерированный планировщиком PostgreSQL. Планировщиком называется компонент PostgreSQL, пытающийся выработать наиболее эффективный способ выполнения запроса SQL. В плане выполнения содержится информация о том, как будет организован просмотр таблиц, задействованных в запросе, сервером базы данных. В зависимости от ситуации возможен последовательный перебор или выборка по индексу. В план включена информация обо всех таблицах, участвующих в запросе.

По выходным данным команды EXPLAIN можно оценить относительную стоимость планов выполнения запроса, измеряемую в количестве операций выборки страниц с диска. PostgreSQL не пытается преобразовать это число в более привычные промежутки времени, поскольку результат в значительной мере зависит от требований к оборудованию и загрузки операционной системы. Таким образом, стоимость плана выполнения запроса может использоваться только как относительный критерий для сравнения с другими, альтернативными запросами.

Стоимость выражается двумя числами, разделенными точкой. Первое число определяет приблизительную стоимость инициализации (интервал времени перед получением первой записи). Второе число определяет приблизительную оценку общего времени, необходимого для полной обработки запроса.

При наличии ключевого слова VERBOSE команда EXPLAIN выводит внутреннее представление плана в виде дерева. Для среднего пользователя эта информация не представляет интереса; она предназначена для программистов, знакомых с внутренним устройством PostgreSQL.

Синтаксис

EXPLAIN [ANALYSE][VERBOSE] запрос

Параметры

ANALYSE. Выполнение запроса и вывод фактического времени выполнения операции.

VERBOSE. При наличии необязательного ключевого слова VERBOSE в плане запроса выводится дополнительная информация.

запрос. Запрос, план выполнения которого вы хотите получить.

Результаты

NOTICE: QUERY PLAN: plan. За сообщением следует план выполнения запроса, полученный от сервера.

EXPLAIN. Сообщение выводится после плана запроса и является признаком завершения выходных данных.

Выборка данных с использованием языка запросов SQL

Центральное место в SQL занимает команда *SELECT*, предназначенная для построения запросов и выборки данных из таблиц и представлений. Данные, возвращаемые в результате запроса, называются итоговым набором; как и таблицы, они состоят из записей и полей.

Данные итогового набора не хранятся на диске в какой-либо постоянной форме. Итоговый набор является лишь временным представлением данных, полученных в результате запроса. Структура полей итогового набора может соответствовать структуре исходной таблицы, но может и радикально отличаться от нее. Итоговые наборы даже могут содержать поля, выбранные из других таблиц.

Из-за своей особой роли в PostgreSQL команда *SELECT* также является самой сложной командой, обладающей многочисленными секциями и параметрами. Ниже приведено общее определение синтаксиса *SELECT*.

SELECT [ALL | DISTINCT [ON (выражение [...])]]
цель [AS имя] [...] [FROM источник [...]]
[[NATURAL] тип_объединения источник
[ON условие | USING (список_полей)]] [...]
[WHERE условие] [GROUP BY критерий [...]]
[HAVING условие [...]]
[{UNION | INTERSECT | EXCEPT} [ALL] подзапрос]
[ORDER BY выражение [ASC | DESC | USING оператор] [...]]
[FOR UPDATE [OF таблица [...]]]
[LIMIT {число | ALL} [OFFSET начало]]

В этом описании *источник* представляет собой имя таблицы или подзапрос. Эти общие формы имеют следующий синтаксис:

FROM {[ONLY] таблица
[[AS] синоним [(синоним_поля[...])]]

(запрос) [AS] синоним [(синоним_поля [...])]}

ALL. Необязательное ключевое слово *ALL* указывает на то, что в выборку включаются все найденные записи.

DISTINCT [ON (выражение [...])]. Секция *DISTINCT* определяет поле или выражение, значения которого должны входить в итоговый набор не более одного раза.

Цель [AS имя] [...]. В качестве *цели* обычно указывается имя поля, хотя *цель* также может быть константой, идентификатором, функцией или общим выражением. Перечисляемые *цели* разделяются запятыми, существует возможность динамического назначения имен *целей* в секции *AS*. Звездочка (*) является сокращенным обозначением всех несистемных полей, вместе с ней в списке могут присутствовать и другие *цели*.

FROM источник [...]. В секции *FROM* указывается *источник*, в котором PostgreSQL ищет заданные *цели*. В данном случае *источник* является именем таблицы или подзапроса. Допускается перечисление нескольких *источников*, разделенных запятыми (примерный аналог перекрестного запроса). Синтаксис секции *FROM* подробно описан ниже.

[NATURAL] тип_объединения источник [ON условие | USING (список_полей)]. Источники *FROM* могут группироваться в секции *JOIN* с указанием типа объединения (*INNER*, *FULL*, *OUTER*, *CROSS*). В зависимости от типа объединения также может потребоваться уточняющее условие или список полей.

WHERE условие. Секция *WHERE* ограничивает итоговый набор заданными критериями. Условие должно возвращать простое логическое значение (true или false), но оно может состоять из нескольких внутренних условий, объединенных логическими операторами (например, AND или OR).

GROUP BY критерий [...]. Секция *GROUP BY* обеспечивает группировку записей по заданному *критерию*. Причем *критерий* может быть простым именем поля или произвольным выражением, примененным к значениям итогового набора.

HAVING условие [...]. Секция *HAVING* похожа на секцию *WHERE*, но условие проверяется на уровне целых групп, а не отдельных записей.

{UNION | INTERSECT | EXCEPT} [ALL] подзапрос. Выполнение одной из трех операций, в которых участвуют два запроса (исходный и дополнительный); итоговые данные возвращаются в виде набора с обобщенной структурой, из которого удаляются дубликаты записей (если не было задано ключевое слово *ALL*). *UNION* - объединение (записи, присутствующие в любом из двух наборов). *INTERSECT* - пересечение (записи, присутствующие одновременно в двух наборах). *EXCEPT* - исключение (записи, присутствующие в основном наборе *SELECT*, но не входящие в подзапрос).

ORDER BY выражение. Сортировка результатов команды *SELECT* по заданному *выражению*.

[ASC | DESC | USING оператор]. Порядок сортировки, определяемой секцией *ORDER BY выражение*: по возрастанию (ASC) или по убыванию (DESC). С ключевым словом *USING* может задаваться *оператор*, определяющий порядок сортировки (например, < или >).

FOR UPDATE [OF таблица [...]]. Возможность монопольной блокировки возвращаемых записей. В транзакционных блоках *FOR UPDATE* блокирует записи указанной таблицы до завершения транзакции. Заблокированные записи не могут обновляться другими транзакциями.

LIMIT {число | ALL}. Ограничение максимального количества возвращаемых записей или возвращение всей выборки (*ALL*).

OFFSET начало. Точка отсчета записей для секции *LIMIT*. Например, если в секции *LIMIT* установлено ограничение в 100 записей, а в секции *OFFSET* -50, запрос вернет записи с номерами 50-150 (если в итоговом наборе найдется столько записей).

Ниже описаны компоненты секции *FROM*.

[ONLY] таблица. Имя таблицы, используемой в качестве источника для команды *SELECT*. Ключевое слово *ONLY* исключает из запроса записи всех таблиц-потомков.

[AS] синоним. Источникам *FROM* могут назначаться необязательные псевдонимы, упрощающие запрос. Ключевое слово *AS* является необязательным.

(запрос) [AS] синоним. В круглых скобках находится любая синтаксически правильная команда *SELECT*. Итоговый набор, созданный запросом, используется в качестве источника *FROM* так, словно выборка производится из статической таблицы. При выборке из подзапроса обязательно должен назначаться синоним.

(синоним_поля [...]). Синонимы могут назначаться не только всему источнику, но и его отдельным полям. Перечисляемые синонимы нолей разделяются запятыми и группируются в круглых скобках за синонимом источника *FROM*. Синонимы перечисляются в порядке следования полей в таблице, к которой они относятся.

Целями команды *SELECT* могут быть не только простые поля, но и произвольные выражения (включающие вызовы функций или различные операции с идентификаторами) и константы. Синтаксис команды практически не изменяется, появляется лишь одно дополнительное требование - все самостоятельные выражения, идентификаторы и константы должны разделяться запятыми. В списке разрешены произвольные комбинации разнотипных целей.

Команда *SELECT* также может использоваться для простого вычисления и вывода результатов выражений и констант. В этом случае она не содержит секции *FROM* или имен столбцов.

Для каждой цели в списке может задаваться необязательная секция *AS*, которая назначает синоним (новое произвольное имя) для каждого поля в итоговом наборе. Имена синонимов подчиняются тем же правилам, что и имена обычных идентификаторов (в частности, они могут содержать

внутренние пробелы или совпадать с ключевыми словами при условии заключения их в апострофы и т. д.)

Назначение синонима не влияет на исходное поле и действует лишь в контексте итогового набора, возвращаемого запросом. Секция *AS* особенно удобна при «выборке» выражений и констант, поскольку синонимы позволяют уточнить смысл неочевидных выражений или констант.

В секции *FROM* указывается источник данных - таблица или итоговый набор. Секция может содержать несколько источников, разделенных запятыми.

Использование нескольких источников данных в PostgreSQL требует осторожности. В результате выполнения команды *SELECT* для нескольких источников без секций *WHERE* и *JOIN*, уточняющих связи между источниками, возвращается полное декартово произведение источников. Иначе говоря, итоговый набор содержит все возможные комбинации записей из всех источников. Обычно для уточнения связей между источниками, перечисленными через запятую в секции *FROM*, используется секция *WHERE*.

Для предотвращения неоднозначности в «полные» имена столбцов включается имя таблицы. При этом используется специальный синтаксис, называемый точечной записью (название связано с тем, что имя таблицы отделяется от имени поля точкой).

Точечная запись обязательна только при наличии неоднозначности между наборами данных.

Если в качестве источника используется итоговый набор, созданный подзапросом, то весь подзапрос заключается в круглые скобки. По наличию круглых скобок PostgreSQL узнает о том, что команда *SELECT* интерпретируется как подзапрос, и выполняет ее перед выполнением внешней команды *SELECT*.

Источникам данных в секции *FROM* - таблицам, подзапросам и т. д. - можно назначать синонимы в секции *AS* (по аналогии с отдельными полями).

Синонимы часто используются для упрощения точечной записи. Наличие синонима для набора данных позволяет обращаться к нему при помощи точечной записи, что делает команды SQL более компактными и наглядными.

Синонимы можно назначать не только для источников данных в секции *FROM*, но и для отдельных полей внутри этих источников. Для этого за синонимом источника данных приводится список синонимов полей, разделенный запятыми и заключенный в круглые скобки. Список синонимов полей представляет собой последовательность идентификаторов, перечисленных в порядке следования полей в структуре таблицы (слева направо).

Список синонимов полей не обязан содержать данные обо всех полях. К тем полям, для которых не были заданы синонимы, можно обращаться по обычным именам. Если единственное поле, которому требуется назначить синоним, находится после других полей, вам придется включить в список все предшествующие поля (синоним поля может совпадать с его именем). В противном случае PostgreSQL не сможет определить, какому полю назначается синоним, и решит, что речь идет о первом поле таблицы.

Необязательное ключевое слово *DISTINCT* исключает дубликаты из итогового набора. Если ключевое слово *ON* отсутствует, из результатов запроса с ключевым словом *DISTINCT* исключаются записи с повторяющимися значениями целевых полей.

Проверяются только поля, входящие в целевой список *SELECT*.

В общем случае PostgreSQL выбирает записи, исключаемые из итогового набора при наличии секции *ON*, по своему усмотрению. Если в запрос вместе с *DISTINCT* входит секция *ORDER BY*, вы можете самостоятельно задать порядок выборки полей так, чтобы нужные записи оказались в начале.

Если вместо исключения всех дубликатов достаточно сгруппировать записи с повторяющимися значениями некоторого критерия, воспользуйтесь секцией *GROUP BY*.

В секции *WHERE* задается логическое условие, которому должны удовлетворять записи итогового набора. На практике команда *SELECT* почти всегда содержит как минимум одну уточняющую секцию *WHERE*. Секция *WHERE* может содержать несколько условий, объединенных логическими операторами (например, AND или OR) и возвращающими одно логическое значение. Количество условий, объединяемых в секции *WHERE*, не ограничено, хотя при наличии двух и более условий обычно выполняется группировка при помощи круглых скобок, наглядно демонстрирующая логическую связь между условиями.

Контрольные вопросы

1. Что такое индекс, для чего он нужен?
2. Какие бывают типы индексов, в чем их различия?
3. Какой формат имеет команда EXPLAIN?
4. Для чего нужна команда EXPLAIN?
5. В чем заключаются различия PRIMARY KEY и UNIQUE?
6. Почему не выгодно создавать редко используемые индексы?
7. Какой формат имеет команда CREATE INDEX?
8. Что делает опция DISTINCT команды SELECT?
9. Как включить в результаты выборки все поля таблицы?
10. Как можно отсортировать результаты выборки?
11. Для чего нужна секция WHERE команды SELECT?
12. Как выглядит структура команды SELECT?
13. Можно ли в результаты выборки добавлять поля, которых нет в таблице?
14. Каким образом можно изменить имена полей, являющихся результатом выборки?

Список источников

1. Малыхина М.П. Базы данных: основы, проектирование, использование: учебное пособие для вузов по направлению подготовки "Информатика и вычислительная техника" / М. П. Малыхина. - СПб., 2006. - 517 с. : ил.
2. Хомоненко А.Д. Базы данных: учебник для вузов по техническим и экономическим специальностям / [Хомоненко А. Д., Цыганков В. М., Мальцев М. Г.] ; под ред. А. Д. Хомоненко. - М., 2006. - 736 с. : ил., табл.
3. Разработка приложений на C# с использованием СУБД PostgreSQL : учебное пособие / [И. А. Васюткина и др.] ; Новосиб. гос. техн. ун-т. - Новосибирск, 2015. - 141, [1] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000220068
4. Дж Уорсли Дж., Дрейк Дж. PostgreSQL. Для профессионалов - СПб.: Питер, 2003. -496 с.
5. PostgreSQL (русский) [Электронный ресурс]: Документация по PostgreSQL. – Режим доступа: http://postgresql.ru.net . - загл. с экрана.
6. Википедия свободная энциклопедия (русский) [Электронный ресурс]: PostgreSQL. – Режим доступа: http://ru.wikipedia.org. - загл. с экрана.
7. Компьютерная документация (русский) [Электронный ресурс]: Функции для работы с датой и временем. – Режим доступа: http://www.hardline.ru/ . - загл. с экрана.
8. Linux и Windows: помощь админам и пользователям (русский) [Электронный ресурс]: 15 команд для управления PostgreSQL. – Режим доступа: http://www.guruadmin.ru . - загл. с экрана.