|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 4 |

**Название:**

Оптимизация процессов в PostgresSQL

**Дисциплина:** Базы данных

**Вариант:** 23

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-33Б |  |  | Д. И. Мироненко |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | М. А. Скворцова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Часть 1. Функции и язык PL/pgSQL**

**Задание**

* Составить SQL-скрипты для создания нескольких функций, упрощающих манипуляции с данными.
* Продемонстрировать полученные знания о возможностях языка PL/pgSQL. В скриптах должны использоваться:
  + Циклы;
  + Ветвления;
  + Переменные;
  + Курсоры;
  + Исключения.
* Обосновать преимущества механизма функций перед механизмом представлений.

*Создание функции для получения всех покупок указанного пользователя*

|  |
| --- |
| CREATE OR REPLACE FUNCTION total\_purchases(target\_user uuid)  RETURNS integer AS  $$  DECLARE  purchases\_cursor CURSOR(target\_user uuid)  FOR SELECT software\_id, price  FROM store.purchase\_history  WHERE user\_id = target\_user;  total\_price integer DEFAULT 0;  purchase\_record record;  BEGIN  OPEN purchases\_cursor(target\_user);  LOOP  FETCH purchases\_cursor INTO purchase\_record;  EXIT WHEN NOT FOUND;  total\_price := total\_price + purchase\_record.price;  END LOOP;  CLOSE purchases\_cursor;  IF total\_price = 0 THEN  RAISE EXCEPTION 'No purchases found for the user %!', target\_user;  END IF;  RETURN total\_price;  END;  $$  LANGUAGE 'plpgsql'; |

A screenshot of a computer

Description automatically generated

**Рисунок 1** – тестирование функции для получения всех покупок указанного пользователя

*Создание функции, проверяющей доступна ли миграция для указанного пользователя*

|  |
| --- |
| CREATE OR REPLACE FUNCTION is\_migration\_available(target\_user uuid, target\_software uuid)  RETURNS boolean AS  $$  DECLARE  purchases CURSOR(target\_user uuid, target\_software uuid)  FOR SELECT id  FROM store.purchase\_history  WHERE user\_id = target\_user  AND software\_id = target\_software;  purchase\_record record;  BEGIN  OPEN purchases(target\_user, target\_software);  FETCH purchases INTO purchase\_record;  IF NOT FOUND THEN  CLOSE purchases;  RETURN false;  END IF;  CLOSE purchases;  RETURN true;  END;  $$  LANGUAGE 'plpgsql'; |

A screenshot of a computer

Description automatically generated

**Рисунок 2** – тестирование функции, проверяющей доступна ли миграция для указанного пользователя

**В чем преимущества механизма функций перед механизмом представлений?**

Функции могут принимать параметры и возвращать результаты, что делает их более гибкими для обработки различных вариантов запросов к данным. Они также могут быть повторно использованы в различных запросах и процедурах, что улучшает эффективность разработки и сокращает дублирование кода. Кроме того, в отличие от представлений, функции могут оптимизироваться для выполнения сложных операций, обеспечивая более эффективную обработку данных.

**Часть 2. Оптимизация запросов. Индексы**

**Задание**

Для выполнения задания необходим достаточно большой объем данных, чтобы оптимизация была целесообразной (порядка 1 млн. строк в каждой таблице).

Необходимо подготовить два запроса:

* Запрос к одной таблице, содержащий фильтрацию по нескольким полям.
* Запрос к нескольким связанным таблицам, содержащий фильтрацию по нескольким полям.

Для каждого из этих запросов необходимо провести следующие шаги:

* Получить план выполнения запроса без использования индексов.
* Получить статистику (IO и Time) выполнения запроса без использования индексов.
* Создать нужные индексы, позволяющие ускорить запрос.
* Получить план выполнения запроса с использованием индексов и сравнить с первоначальным планом.
* Получить статистику выполнения запроса с использованием индексов и сравнить с первоначальной статистикой.
* Оценить эффективность выполнения оптимизированного запроса. Также необходимо продемонстрировать полезность индексов для организации полнотекстового поиска, фильтрации с использованием массива и json-формата.

Для таблицы объемом больше 100 млн. записей произвести оптимизацию, позволяющую быстро удалять старые данные, ускорить вставку и чтение данных.

*Запрос на получение наиболее важной информации о ПО*

|  |
| --- |
| SELECT  name,  description,  tags,  license\_price  FROM store.softwares  WHERE  category = 'education'  AND 'creation' = ANY(tags); |

A screenshot of a computer

Description automatically generated

**Рисунок 3** – анализ запроса на получение наиболее важной информации о ПО

*Добавление нужных индексов для оптимизации запроса выше*

|  |
| --- |
| CREATE INDEX idx\_category ON "store"."softwares" ("category");  CREATE INDEX idx\_tags ON "store"."softwares" USING GIN("tags"); |

A screenshot of a computer program

Description automatically generated

**Рисунок 4** – анализ запроса на получение наиболее важной информации о ПО после оптимизации

*Запрос на получение отрицательных отзывов о ПО из категории “education”*

|  |
| --- |
| SELECT  softwares.name,  reviews.rating,  reviews.review\_content  FROM store.reviews  JOIN store.softwares ON reviews.software\_id = softwares.id  WHERE  softwares.category = 'education'  AND reviews.rating <= 3; |

A screenshot of a computer

Description automatically generated

**Рисунок 5** – анализ запроса на получение отрицательных отзывов о ПО из категории “education”

*Добавление нужных индексов для оптимизации запроса выше*

|  |
| --- |
| CREATE INDEX idx\_category ON "store"."softwares" ("category");  CREATE INDEX idx\_rating ON "store"."reviews" ("rating"); |

A screenshot of a computer

Description automatically generated

**Рисунок 6** – анализ запроса на получение отрицательных отзывов о ПО из категории “education” после оптимизации

**Вывод**

Реализованы полезные функции для анализа базы данных; добавлены индексы для некоторых колонок с целью повысить эффективность часто используемых запросов.

**Контрольные вопросы**

1. *Описать в каких случаях целесообразно создавать функции.*Функции могут использоваться для объединения серии операций в единое действие, что улучшает модульность, обслуживаемость и повторное использование кода. Функции могут содержать сложные запросы, которые могут быть повторно использованы в различных контекстах.
2. *Рассказать о курсорах, как и зачем используются.*  
   Курсоры - это механизм, который позволяет выполнять запросы к базе данных построчно, шаг за шагом. Они используются для обработки больших объемов данных и ускорения выполнения запросов.
3. *Рассказать о работе с циклами.*Циклы - это механизм, который позволяет выполнять определенные операции с каждым элементом данных в таблице или наборе данных. Они могут использоваться для решения различных задач.  
   DECLARE c CURSOR FOR SELECT \* FROM <table>;  
   FETCH NEXT FROM c INTO <variable>;  
   WHILE <variable> IS NOT NULL DO  
    -- выполнение операций с каждой строкой данных  
    FETCH NEXT FROM c INTO <variable>;  
   END WHILE;
4. *В чем отличие первичного ключа и уникального индекса?*  
   Основное различие между ними заключается в том, что первичный ключ обычно используется для однозначной идентификации каждой строки в таблице, в то время как уникальный индекс просто гарантирует уникальность значений в столбцах, но не обязательно для идентификации строк. Кроме того, для первичного ключа по умолчанию создается кластеризованный индекс для ускорения производительности доступа к данным.
5. *В каких случаях имеет смысл создавать индексы? Какие колонки следует включать в индекс и почему?*Чтобы ускорить частые операции поиска. Индексы также могут быть использованы при выполнении операций объединения (JOIN) для ускорения запросов. Кроме того, индексы могут улучшить производительность операций сортировки и группировки.  
   В индекс нужно включать колонки, которые используются в условиях WHERE или которые используются в операциях объединения таблиц.
6. *Какие существуют способы внутренней организации индексов?*  
   B-дерево, hash, gin (generalized search tree), GiST (generalized search tree).
7. *Рассказать о проблеме фрагментации индексов. Как бороться с фрагментацией?*  
   Фрагментация индексов возникает, когда данные в таблице часто добавляются, обновляются или удаляются, что приводит к тому, что индексы становятся разрозненными и неоптимальными для поиска. Это может привести к увеличению времени выполнения запросов, уменьшению производительности базы данных и ухудшению общего опыта работы с системой.  
   Для борьбы с фрагментацией индексов можно применять следующие методы:  
   - VACUUM и REINDEX. Нужно регулярно выполнять операцию VACUUM, чтобы освобождать место, занимаемое удаленными строками таблицы, и уменьшать фрагментацию индексов. Также, нужно использовать команду REINDEX для восстановления порядка данных в индексах.  
   - CLUSTER. Команда CLUSTER позволяет упорядочить данные в таблице в соответствии с порядком индекса, что снижает фрагментацию и улучшает производительность запросов.
8. *Имеет ли значение порядок указания колонок при создании индекса?*  
   Да. Порядок колонок определяет, какой тип запросов может использоваться для ускорения. Например, составной индексы на столбцы (A, B) может быть эффективен при выполнении запросов, которые фильтруют данные по столбцу A или используют порядок (A, B). Однако он может быть менее эффективен для запросов, которые ищут данные только по столбцу B.
9. *В чем разница между Index Scan и Index Seek?*  
   Index Scan и Index Seek – это два способа доступа к данным через индексы, и их разница заключается в способе выполнения запроса.  
   Index Scan означает, что база данных просматривает все дерево индекса и находит нужные записи по условию запроса. Это может быть эффективно, если необходимо найти большой диапазон данных или если индекс отсортирован в нужном порядке.  
   Index Seek означает точный поиск определенной записи в индексе. База данных использует индекс, чтобы найти нужную запись без просмотра всех остальных записей. Это обычно более эффективный способ доступа к данным при поиске конкретных значений.
10. *В чем разница между секционированием и наследованием?*Секционирование – это процесс разделения большой таблицы на несколько меньших частей, называемых секциями. При этом каждая секция может иметь свою собственную структуру данных и правила доступа.  
    Наследование — это процесс создания новой таблицы на основе существующей таблицы. Новая таблица наследует структуру данных и правила доступа существующей таблицы. Наследование позволяет упростить проектирование базы данных и улучшить ее производительность, так как запросы к базе данных могут выполняться быстрее, если они выполняются на таблице с более низкой нагрузкой.
11. *Зачем нужен ANALYZE?*  
    ANALYZE - это команда, которая используется для анализа производительности базы данных. Она позволяет определить, какие запросы занимают больше всего времени выполнения и какие операции выполняются медленнее всего.
12. *Могут ли индексы ухудшить производительность?*  
    Да, индексы могут ухудшить производительность в базе данных. При обновлении данных на индексированных столбцах база данных может столкнуться с проблемой медленных операций обновления из-за необходимости обновления индексов. Со временем индексы могут стать фрагментированными, что снижает их эффективность и ведет к ухудшению производительности базы данных.
13. *На что влияет порядок сортировки (ASC\DESC) при создании индекса?*  
    Порядок сортировки при создании индекса влияет на скорость поиска и выборки данных из таблицы. Если указать DESC, то слева в дереве окажутся большие значения, а справа - меньшие.