МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Базы данных»

Тема: Нагрузочное тестирование БД

Студентка гр. 1304 _____ Чернякова В.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

Цель работы.

Проведение нагрузочного тестирования на СУБД PostgreSQL при помощи ORM.

Задание.

Вариант 3(25).

- Написать скрипт, заполняющий БД большим количеством тестовых данных.
- Измерить время выполнения запросов, написанных в ЛР3.

Проверить для числа записей:

100 записей в каждой табличке

1000 записей

10000 записей

1000000 записей

Все запросы выполнять с фиксированным ограничением на вывод (LIMIT), т.к. запросы без LIMIT всегда будет выполняться O(n) от кол-ва записей

Проверить влияние сортировки на скорость выполнения запросов.

Для измерения использовать фактическое (не процессорное и т.п.) время.

- Добавить в БД индексы (хотя бы 5 штук). Измерить влияние (или его отсутствие) индексов на скорость выполнения запросов. Обратите внимание на:

Скорость сортировки больших табличек

Скорость JOIN

Выполнение работы.

1. Выбор скрипта, для заполнения тестовыми данными

В третей лабораторной работы была выбрана *ORM – GORM*. Для создания данных использовалось *gofakeit*. Ссылка: https://github.com/brianvoe/gofakeit.

2. Установка

В *IDE GoLand* был установлен *faker* с помощью следующей команды:

```
go get github.com/brianvoe/gofakeit/v6
```

3. Заполнение тестовыми данными

Для заполнения таблиц классов, кабинетов, кабинетов учителей и предметов учителей был написан код на питоне, генерирующий данные. Так как в первых двух таблицах рандомно сгенерированные данные являются ключами, то было возможно их совпадение, поэтому данные были созданы и передаются в таблицу в формате json. Для двух остальных таблиц принято такое решение, так как в них должны быть обязательно данные, которые есть и в первых двух таблицах.

Основное использование *faker*. Где необходимо было создать рандомные данные прописывалась строчка *fake:"{}"*, где в скобках указывалась функция, генерирующая соответствующие данные.

4. Проведение нагрузочного тестирования

Сравнение производится с учетом сортировки и без, а также при наличии индексации и ее отсутствии.

Индексация была добавлена в таблицы, которые чаще всего используются при запросах:

Class (ClassNumber, ClassLetter);

Schedule (DayName, LessonNumber, SubjectId, TeacherId, ClassroomNumber, ClassNumber, ClassLetter);

Classroom (ClassroomNumber);

Subject (SubjectId, SubjectName);

Student (StudentId, ClassNumber, ClassLetter);

Teacher (TeacherId, TeacherName, TeacherSurname).

Основные результаты тестирования представлены в таблице 1.

Таблица 1. Нагрузочное тестирование БД

	Без Order	Без Order с индексами	Order	Order с индексами
100	2.8265ms	3.1307ms	3.0959ms	3.6907ms
1000	4.0725ms	3.6802ms	4.5912ms	4.0545ms

10.000	10.2762ms	5.0008ms	10.8613ms	6.0272ms
1.000.000	10.4818ms	6.1017ms	10.6719ms	7.2376ms

По данным из таблицы 1 сделаем вывод. Скорость запросов без сортировки быстрее, чем с сортировкой. Индексация также дает выигрыш по времени при реализации запросов. Однако при 100 записях индексация лишь увеличила время поиска, это может быть связано с тем, что при таком достаточно маленьком количестве данных проще просмотреть по всей таблице, чем использовать индексы.

5. Нагрузочное тестирование БД. Проверка скорости JOIN

В таблице 2 приведены результаты тестирования 5 запроса, где используется 2 INNER JOIN.

Таблица 2. Нагрузочное тестирование БД. Проверка скорости JOIN

	Без order	Без Order с индексами	Order	Order с индексами
100	1.2197ms	1.4556ms	1.5663ms	2.0198ms
1000	1.5906ms	1.9934ms	1.9749ms	2.0392ms
10.000	2.9378ms	2.402ms	3.2001ms	2.7888ms
1.000.000	8.4116ms	6.1512ms	12.0744ms	8.2406ms

По данным из таблицы 2 сделаем вывод. Скорость запроса, где используется JOIN, без сортировки быстрее, чем с сортировкой. Индексация также дает выигрыш по времени при реализации запросов. Однако при 100 и 1000 записях индексация лишь увеличило время поиска, это может быть связано с тем, что при таком относительно маленьком количестве данных проще просмотреть по всей таблице, чем использовать индексы.

В приложении А предоставлена ссылка на PR.

Выводы.

В данной лабораторной работе проведено нагрузочное тестирование БД.

По итогам тестирования были сделаны соответствующие выводы.

ПРИЛОЖЕНИЕ А
ССЫЛКИ

Ссылка на PR:

https://github.com/moevm/sql-2023-1304/pull/78