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

Студент гр. 3341 _____ Шаповаленко Е.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2025

Цель работы.

Целью данной лабораторной работы является изучение индексов и нагрузочного тестирования баз данных, проведение нагрузочного тестирования для ранее разработанной базы данных.

Задание.

- Написать скрипт, заполняющий БД большим количеством тестовых данных, рекомендуется использовать faker.js.
- Измерить время выполнения запросов, написанных в ЛР3.
  - Проверить для числа записей:
    - 100 записей в каждой табличке
    - 1.000 записей
    - 1.0000 записей
    - 1.000.000 записей
    - можно больше.
  - 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. Для node.js есть console.time и console.timeEnd.
- 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

(но остальные запросы тоже проверьте)

Вариант 21

Пусть требуется создать программную систему, предназначенную для продавца журналов/комиксов. Такая система должна обеспечивать хранение сведений об имеющихся в магазине журналах, о читателях журналов и списке магазинов. Для каждого журнала в БД должны храниться следующие сведения: название журнала, серия, автор (ы), издательство, год издания, число экземпляров в каждом магазине, а также ISBN и дата продажи журнала. Сведения о читателях библиотеки должны включать почту (email), ФИО покупателя, дату рождения, адрес, номер телефон. Нужно учесть, что покупатели могут делать заказы с разных магазинов, но нужно сохранять информацию, о предпочтительных магазинах (в которых чаще делают заказы). Магазин имеет несколько отделов, которые характеризуются номером, названием и кол-во журналов (вместимость). Магазин может получать новые журналы и списывать старые. ISBN может измениться в результате переклассификации, а почта в результате перерегистрации. Продавцу могут потребоваться следующие сведения о текущем состоянии библиотеки:

1. Какие журналы были куплены определенным покупателем?
2. Как называется журнал с заданным ISBN?
3. Какой ISBN у журнала с заданным названием?
4. Когда журнал был куплен?
5. Кто из покупателей купил журнал более месяца тому назад?
6. Найти покупателя самых редких журналов (по наличию в магазине)?
7. Какое число покупателей пользуется определенным магазином?
8. Сколько покупателей младше 20 лет?

Проведение нагрузочного тестирования.

Для начала необходимо установить зависимости (прописаны в *package.json*) при помощи *nmp i*.

Подключение к базе данных описано в файле *config/database.js*. Для того, чтобы убрать из непосредственно кода программы пароль, была подключена библиотека *dotenv*. Перед запуском программы необходимо создать/отредактировать файл *.env*, указав в нем необходимые параметры:

```
NODE_ENV=development

DB_HOST=host
DB_PORT=port
DB_NAME=name
DB_USER=user
DB_PASS=pass

DB_LOGGING=true
```

Здесь *host* — адрес хоста (т.к. работа происходила в среде WSL Ubuntu, а сама база данных была на этой же машине в среде Windows, использовался шлюз по умолчанию 172.22.0.1), *port* — порт подключения (по умолчанию 5432), *name* — имя базы данных (*lab3*), *user* — имя пользователя (по умолчанию *postgres*), *pass* — пароль.

Запуск тестов производится при помощи комманды *nmp run bench*. Параметры тестирования можно поменять в файле *package.json*. Полные результаты тестирования сохранены в *bench.txt*. В таблицах 1 и 2 представлены результаты тестирования для 100000 и 100 записей соответственно.

Таблица 1, усредненные результаты тестирования для 1000000 записей (время выполнения запроса в мс).

Запрос (см. раздел Вариант)	Без сортировок и индексов	С сортировками, без индексов	Без сортировок, с индексами	С сортировками и индексами
1)	80.2	90.6	2.4	1.2
2)	98.4	91.6	0.8	0.4
3)	96.6	102	0.8	0.4
4)	108	94	0.6	1.2
5)	4.6	190.8	3.4	3.4
6)	1446.6	1350	1417	1407.4
7)	85	88.6	109	87.4
8)	80.6	91.6	0.8	1.2

Как видно из таблицы 1, время выполнения большинства запросов на порядок быстрее, если применяются индексы. Запросы 6 и 7 выделяются тем, что у них низкая селективность, поэтому предусмотренные для них индексы не дают выигрыша. Поэтому использование индексов для них приводит только к увеличению времени, так как индексы только приводят к накладным расходам без преимущества. Также можно заметить, что без использования индексов сортировка практически гарантированно приводит к увеличению времени запроса, в то время как с индексами она зачастую даже быстрее неупорядоченного запроса.

Таблица 2, усредненные результаты тестирования для 100 записей (время выполнения запроса в мс).

Запрос (см. раздел Вариант)	Без сортировок и индексов	С сортировками, без индексов	Без сортировок, с индексами	С сортировками и индексами
1)	0.8	0.6	1	0.8
2)	0.2	0.4	0.4	0.6
3)	0.4	0.4	0.4	0.4
4)	0.8	0.4	0.4	0.4
5)	1.2	0.8	1	1.4

6)	0.8	0.4	0.6	0.8
7)	0.8	0.4	0.4	0
8)	0.2	0.6	0.4	1

Как видно из таблицы 2, время выполнения одной половины запросов увеличилась, а другой — уменьшилось. Такая противоречивость объясняется малым количеством записей, при котором накладные расходы часто перекрывают выигрыш в производительности.

Таким образом, индексы следует применять для большого объема записей в запросах с высокой селективностью. Иначе время выполнения может только увеличиться, в связи с накладными расходами.

Ссылку на исходный код см. в приложении А.

Выводы.

В результате выполнения лабораторной работы были изучены индексов и нагрузочного тестирования баз данных, было проведено нагрузочное тестирование для ранее разработанной базы данных. Было установлено, что индексы следует применять для большого объема записей в запросах с высокой селективностью. Иначе время выполнения может только увеличиться, в связи с накладными расходами.

ПРИЛОЖЕНИЕ А
ССЫЛКИ НА РЕЗУЛЬТАТЫ РАБОТЫ

Ссылка на Pull-request с результатами выполненной лабораторной работы: https://github.com/moevm/sql-2025-3341/pull/43