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. 1303 _____ Королева П.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

Цель работы.

Проверить время выполнения запросов, написанных в ЛР3, проверить влияние индексов на скорость выполнения.

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

Выполнение работы.

Для генерации большого количества данных была использована библиотека faker.js.

Замерено время выполнения запросов в разных ситуациях, результаты представлены ниже:

Изначально:

Записей	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	7 [ms]
100	12.59	4.351	3.791	15.091	4.709	2.519	12.246
1000	15.87	4.194	3.769	14.043	4.272	1.996	14.02
10 000	20.022	7.78	8.37	14.739	8.204	1.908	98.036
100 000	21.955	12.195	10.512	6.166	18.94	2.145	349.386
1 000 000	421.77	143.36	232.24	49.73	191.97	263.05	19 381

С сортировкой:

Записей	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	7 [ms]
100	7.92	1.768	2.397	14.213	7.927m	3.512	13.35
1000	6.08	2.26	2.587	20.48	6.025	3.60	18.081
10 000	11.845	9.425	7.873	17.73	7.097	9.094	58.007
100 000	32.1	22.263	21.75	7.082	20.029	51.865	370.90
1 000 000	440.773	136.13	119.6	15.84	164.852	250.38	19 841

В начале сортировка сработала быстрее, возможно из-за того, что запрос на нахождение записей с максимальными/минимальными значениями оптимальнее делать через сортировку, нежели через вложенный запрос.

С сортировкой и индексацией:

Записей	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	7 [ms]
100	7.234	1.382	1.93	14.38	5.002	2.513	6.36
1000	10.36	2.353	2.341	14.032	4.263	2.601	9.499
10 000	11.40	2.802	2.291	17.30	6.77	3.849	40.113
100 000	13.0	2.902	2.455	15.42	6.045	3.9	382.5
1 000 000	139.20	3.227	9.614	21.34	6.929	22.189	19 846

В большинстве запросов индексация значительно ускорила процесс. Но в последнем и самом долгом запросе с большим количеством join и созданием временной таблицы все же погоды не сделала.

Только с индексацией:

Записей	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	7 [ms]
100	9.961	3.602	3.834	13.458	3.707	2.227	8.097
1000	12.876	3.788	4.373	13.872	4.02	2.844	14.634
10 000	14.213	4.51	4.763	14.05	4.3	2.81	37.358
100 000	14.67	4.886	7.65	14.34	4.3	3.0	350.284
1 000 000	160.34	12.19	11.389	15.071	6.242	18.266	18 9

Итак, видно, что индексация заметно снижает время выполнения запроса. Однако изредка бывают ситуации (как например последний запрос), когда индексация оказывается не эффективной.

Вывод.

Было проверено время выполнения запросов, написанных в ЛР3, проверено положительное влияние индексов на скорость выполнения.

Приложение А

Ссылка на пуллреквест:

https://github.com/moevm/sql-2023-1303/pull/57