Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

«Оптимизация»

по дисциплине «Базы данных»

Работу выполнил:

студент гр. 43501/3

Хуторной Я. В.

Руководитель:

Мяснов А. В.

«\_\_» \_\_\_\_\_\_\_\_\_ 2016 г

Санкт-Петербург

2016

**Цели работы**

Получить практические навыки создания эффективных SQL-запросов.

**Программа работы**

* Ознакомьтесь со способами профилирования и интерпретации планов выполнения SQL-запросов;
* Ознакомьтесь со способами оптимизации SQL-запросов с использованием:

- индексов

- модификации запроса

- создания собственного плана запроса

- денормализации БД

* Нагенерируйте данные во всех таблицах, если это ещё не сделано;
* Выберите один из существующих или получите у преподавателя новый "тяжёлый" запрос к Вашей БД;
* Оцените производительность запроса и проанализируйте результаты профилирования (для этого используйте SQL Editor в средстве IBExpert);
* Выполните оптимизацию запроса двумя или более из указанных способов, сравните полученные результаты;
* Продемонстрируйте результаты преподавателю;
* Напишите отчёт с подробным описанием всех этапов оптимизации и выложите его в Subversion;

**Выполнение работы**

**1. Планы выполнения запросов**

Проанализируем влияние плана выполнения запроса на быстродействие выполнения запроса. Исследуемый запрос: Вывести топ-10 композиций, проданных за заданный год.

Попробуем получить наихудшее время выполнения запроса с помощью явного указания плана выполнения. Для этого будем использовать операцию NATURAL. NATURAL - полный перебор всей таблицы до тех пор, пока не буду найдены требуемые данные.

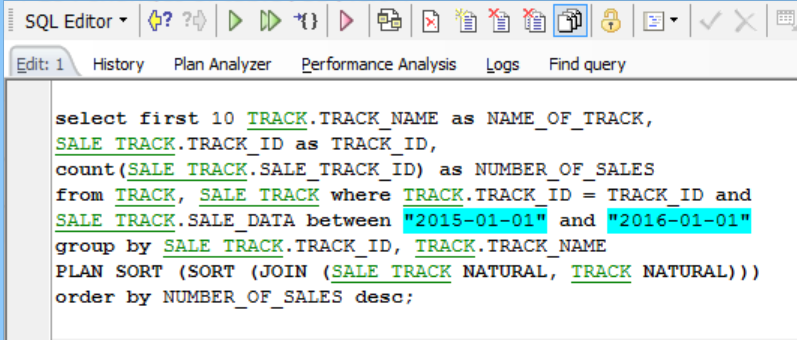


Рис. 1. Текст SQL-запроса с операцией извлечения NATURAL.

Анализ быстродействия приведен во вкладке Plan Analyzer:

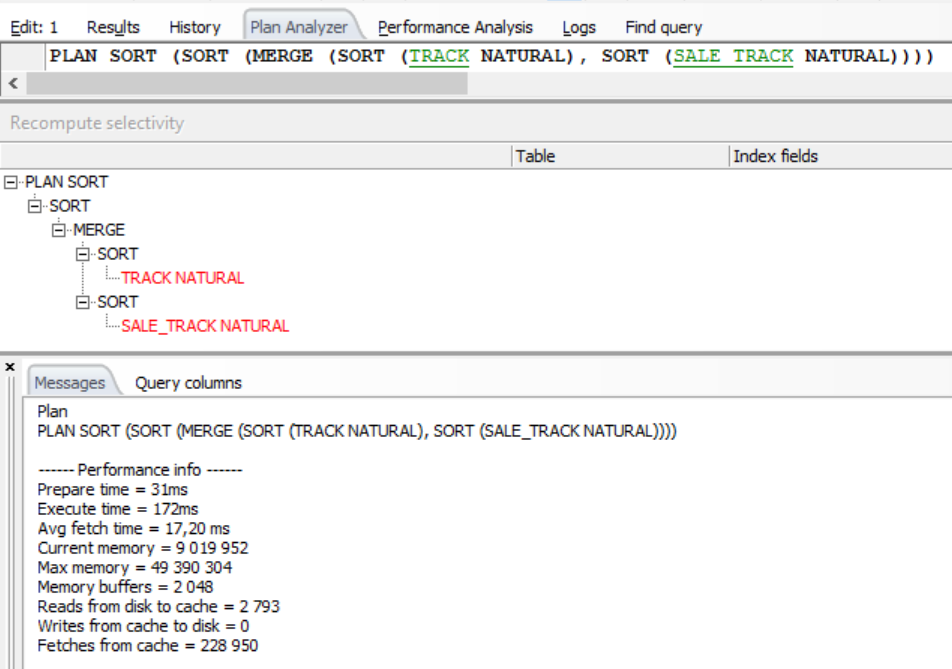


Рис. 2. Результаты анализа SQL-запроса с операцией извлечения NATURAL.

Время выполнения в таком случае составляет 172 мс.

Попробуем повысить производительность, добавив операцию извлечения по индексу в таблице TRACK. Результат оптимизации:

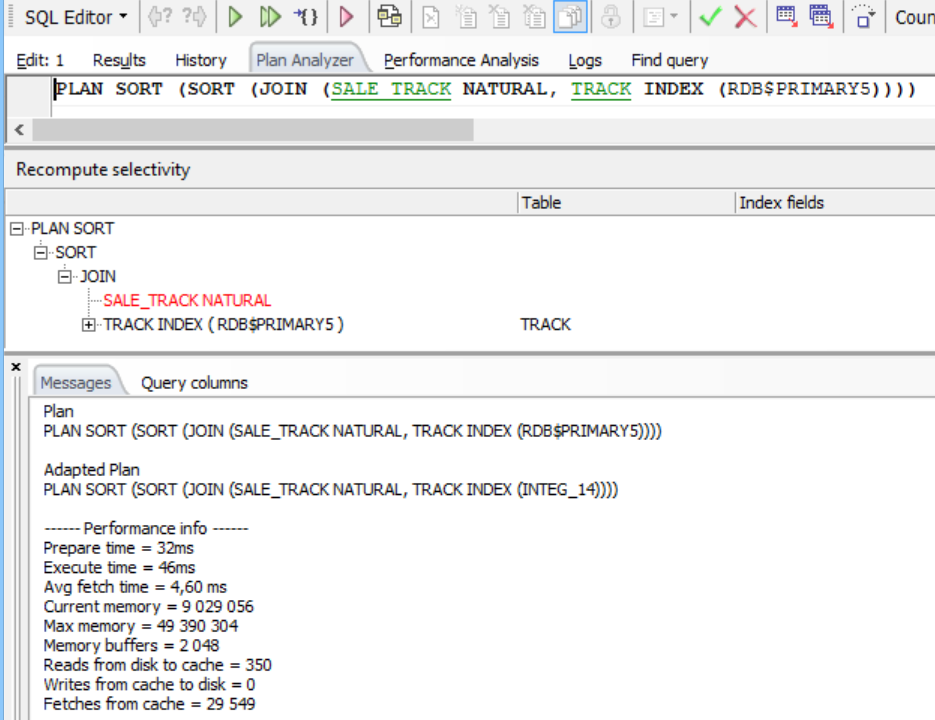


Рис. 3. Результаты анализа SQL-запроса с операцией извлечения по индексу в одной из таблиц.

В этом случае время выполнения составило 46 мс. Явное указание использования операции извлечения по индексу таблицы в плане выполнения запроса позволило уменьшить время выполнения в 4 раза.

**2. Индексы**

В данном SQL-запросе ищутся данные из таблицы SALE\_TRACK, в которых поле SALE\_DATA находится в заданном временном диапазоне:



Поэтому целесообразно будет добавить индекс на поле SALE\_TRACK.SALE\_DATA:

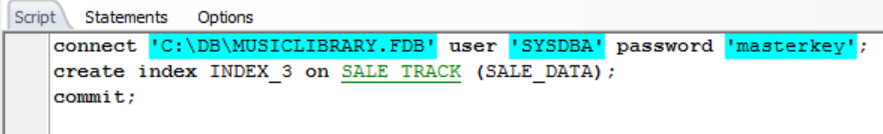


Рис. 4. Создание индекса.

Анализ быстродействия приведен во вкладке Plan Analyzer:

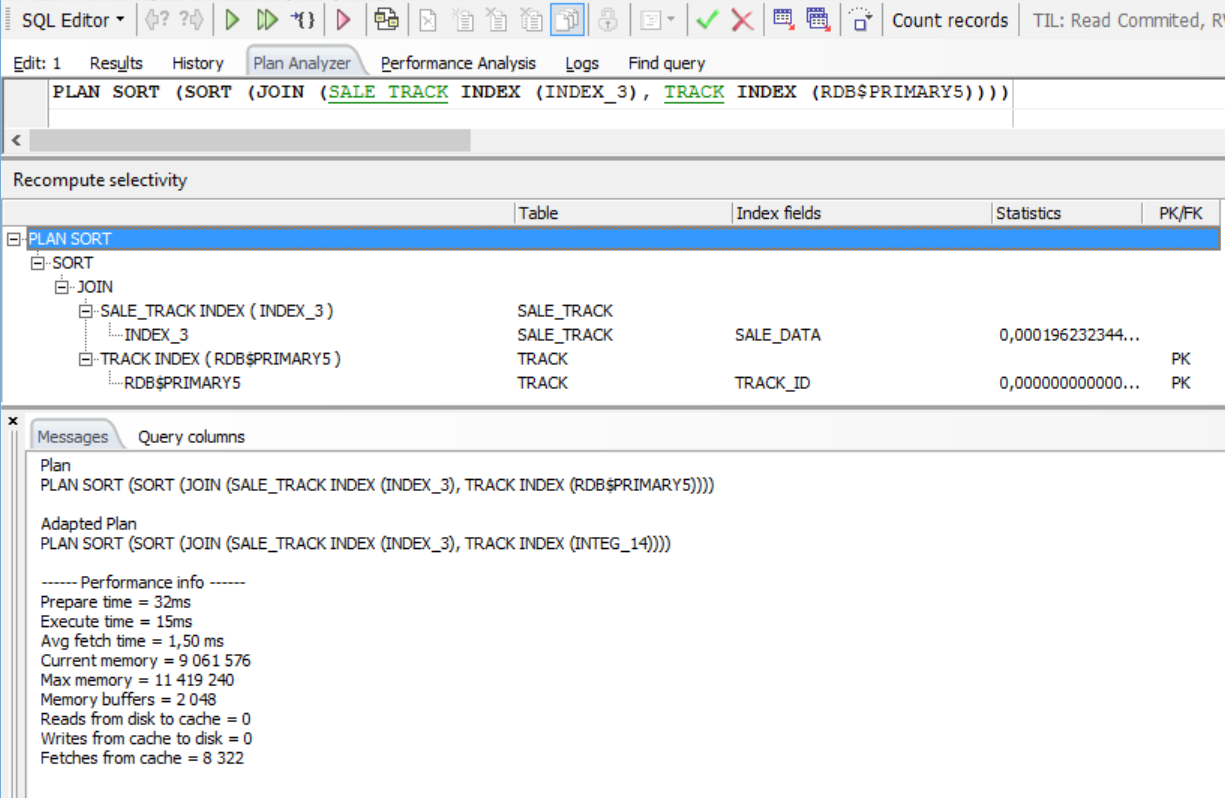


Рис. 5. Результаты анализа SQL-запроса с операцией извлечения по индексу в двух таблицах.

После создания индекса оптимизатор автоматически создал план, в котором используется операция извлечения по индексу в таблице SALE\_TRACK. Благодаря этому время выполнения уменьшилось еще в 3 раза и составило 15 мс.

**3. Денормализация**

Попробуем повысить производительность путем введения избыточной функциональной зависимости, а именно добавим в таблицу SALE\_TRACK дополнительное поле TRACK\_NAME, взятое из таблицы TRACK. Для этого создадим новую таблицу и скопируем в неё необходимые данные.

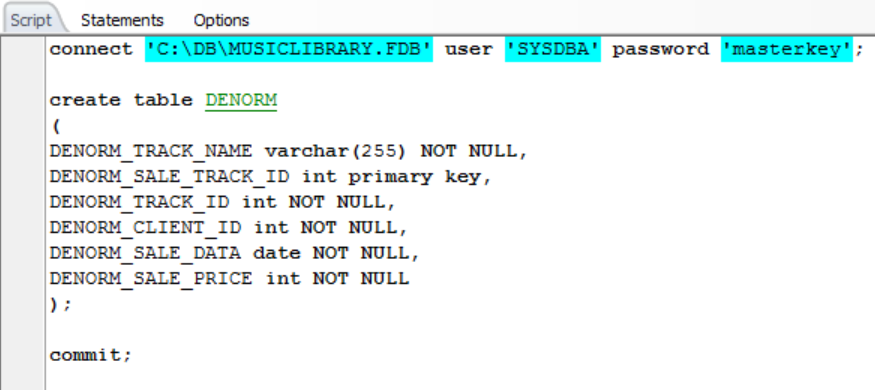


Рис. 6. Создание новой денормализованной таблицы.

Запрос будет выглядеть следующим образом:

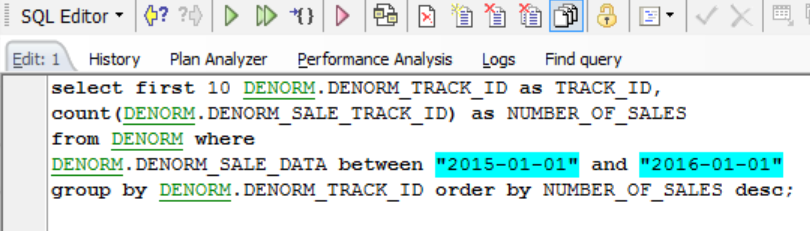


Рис. 7. SQL-запрос.

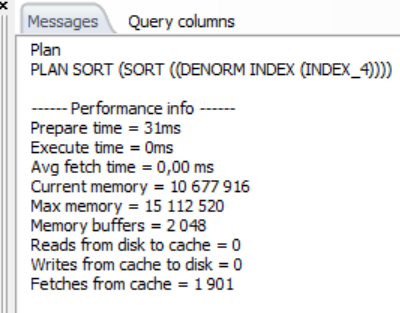


Рис. 8. Результаты анализа SQL-запроса с применением денормализации.

Таким образом, денормализация таблицы привела к наилучшему результату.

**Выводы**

В запросах к полностью нормализованной базе нередко приходится соединять до десятка, а то и больше, таблиц. Каждое соединение — операция весьма ресурсоемкая. В такой ситуации может помочь денормализация путем сокращения количества таблиц или денормализация путём ввода дополнительного поля в одну из таблиц. В денормализованной базе данных повышается избыточность данных, что может повысить производительность, но потребует больше усилий для контроля за связанными данными.

Индексы могут улучшить производительность системы, т.к. они обеспечивают подсистему запросов быстрым путем для нахождения данных. Однако, при изменении данных, индексы также должны быть изменены, чтобы отразить соответствующие действия над данными, что может значительно снизить производительность системы. Поскольку индексы могут занимать значительное дисковое пространство, не следует создавать индексов больше, чем необходимо.

Также для оптимизации возможно использовать явное указание плана выполнения запроса - последовательности обхода и соединения таблиц. Оптимизатор запросов SQL обычно автоматически выбирает наилучший план выполнения запроса.