Санкт-Петербургский Политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт о лабораторной работе №1**

**Дисциплина**: Базы данных

**Тема**: Оптимизация SQL-запросов

Выполнил студент гр. 43501/3 Никитенко А.П.

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2016 г.

Санкт-Петербург

2016

**Цель работы**

Получить практические навыки создания эффективных SQL-запросов.

## Программа работы

1. Ознакомьтесь со способами профилирования и интерпретации планов выполнения SQL-запросов
2. Ознакомьтесь со способами оптимизации SQL-запросов с использованием:
   * индексов
   * модификации запроса
   * создания собственного плана запроса
   * денормализации БД
3. Нагенерируйте данные во всех таблицах, если это ещё не сделано
4. Выберите один из существующих или получите у преподавателя новый "тяжёлый" запрос к Вашей БД
5. Оцените производительность запроса и проанализируйте результаты профилирования (для этого используйте SQL Editor в средстве IBExpert)
6. Выполните оптимизацию запроса двумя или более из указанных способов, сравните полученные результаты
7. Продемонстрируйте результаты преподавателю
8. Напишите отчёт с подробным описанием всех этапов оптимизации и выложите его в Subversion

**Индекс** — объект [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путем последовательного просмотра таблицы строка за строкой может занимать много времени. Индекс формируется из значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы и, таким образом, позволяет искать строки, удовлетворяющие критерию поиска. Ускорение работы с использованием индексов достигается в первую очередь за счёт того, что индекс имеет структуру, оптимизированную под поиск — например, [сбалансированного дерева](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE).

**Модификация запросов**

Явное указание порядка обхода таблиц с помощью конструкций JOIN.

**План выполнения запроса** — последовательность операций, необходимых для получения результата [SQL](https://ru.wikipedia.org/wiki/SQL)-запроса в [реляционной СУБД](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94).

Перед выполнением запроса происходит его подготовка – составление плана выполнения запроса (последовательности обхода и соединения таблиц).

Операции извлечения данных:

* NATURAL – полный перебор, до тех пор пока не найдет требуемые данные, допустим при извлечении всех данных таблицы;
* INDEX(<index1>,…) – поиск по индексу, обычно более эффективно, используется при соединениях и вычислении условий;
* ORDER <index> - полный перебор с упорядочиванием по заданному индексу, можно использовать при order by и group by.
* JOIN (<select1>,<select2>,…) – соединение двух или более потоков в один, осуществляется перевод всех записей <select1> и поиск для них записей

<select2> и т.д., эффективное слияние при наличии индексов;

* MERGE (<select1>,<select2>,…) – выбирает и сортирует сразу все потоки и производит слияние за один проход, эффективен при отсутствии индексов
* SORT(<select>) – сортировка потока.

При автоматическом создании планов используется статистика по индексам.

Возможно явное указание плана в запросе – перед ORDER BY.

**Денормализация** — намеренное приведение структуры базы данных в состояние, не соответствующее критериям [нормализации](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0), обычно проводимое с целью ускорения операций [чтения](https://ru.wikipedia.org/wiki/Select_(SQL)) из базы за счет добавления избыточных данных.

**Выполнение работы**

Для выполнения работы предложенные преподавателем запрос не подошел, т.к. все участвующие в запросе поля являлись ключами и запрос выполнялся очень быстро.

Поэтому, преподавателем был предложен следующий запрос:

*Для заданного отдела вывести все фактические посещение его сотрудниками комнат заданного здания в определенное время.*

Код запроса:

select factualvisit2.emp\_id,factualvisit2.room\_num,factualvisit2.visit\_date,factualvisit2.start\_of\_vis,factualvisit2.end\_of\_vis,room\_build.building\_num from factualvisit2,room\_build,attachment

where factualvisit2.emp\_id=attachment.emp\_id

and attachment.dep\_id=2022

and factualvisit2.room\_num=room\_build.room\_num

and room\_build.building\_num=3

and factualvisit2.start\_of\_vis<'04:00:00'

and factualvisit2.end\_of\_vis>'04:30:00';

Для оценки скорости выполнения запроса он был выполнен несколько раз с разными наборами данных. Получили следующие времена выполнения:735ms, 610ms, 734ms, 656ms, 828ms. Средняя скорость выполнения запроса 713ms.

План выполнения запроса:

Plan

PLAN JOIN (FACTUALVISIT2 NATURAL, ROOM\_BUILD INDEX (RDB$18), ATTACHMENT INDEX (RDB$FOREIGN15, RDB$FOREIGN16))

Adapted Plan

PLAN JOIN (FACTUALVISIT2 NATURAL, ROOM\_BUILD INDEX (INTEG\_61), ATTACHMENT INDEX (INTEG\_50, INTEG\_52))

Попробуем оптимизировать выполнение запроса с помощью денормализации БД. Для этого введем в таблицу factualvisit2 еще одно поле- BUILD\_NUM, которое будет зависеть только от поля ROOM\_NUM. Таким образом, из запроса можно убрать таблицу ROOM\_BUILD.

alter table factualvisit2 add build\_num int not null;

update factualvisit2 set build\_num=(select room\_build.building\_num from room\_build where room\_build.room\_num=factualvisit2.room\_num);

В связи с изменениями в таблицах, сам запрос тоже немного изменился:

select factualvisit2.emp\_id,factualvisit2.room\_num,factualvisit2.visit\_date,factualvisit2.start\_of\_vis,factualvisit2.end\_of\_vis,factualvisit2.build\_num from factualvisit2,attachment

where factualvisit2.emp\_id=attachment.emp\_id

and attachment.dep\_id=2022

and factualvisit2.build\_num=3

and factualvisit2.start\_of\_vis<'04:00:00'

and factualvisit2.end\_of\_vis>'04:30:00';

Теперь время выполнения запросов: 391ms, 390ms, 391ms, 390ms, 390ms.

Как видно из результатов, денормализация БД также помогла повысить скорость выполнения запросов примерно на 45%(среднее время выполнения запроса после денормализации - 390ms)

План выполнения запроса:

Plan

PLAN JOIN (FACTUALVISIT2 NATURAL, ATTACHMENT INDEX (RDB$FOREIGN15, RDB$FOREIGN16))

Adapted Plan

PLAN JOIN (FACTUALVISIT2 NATURAL, ATTACHMENT INDEX (INTEG\_50, INTEG\_52))

Попробуем уменьшить время выполнения запроса, создав индексы для таблицы factualVisit2(в других таблицах все задействованные поля уже проиндексированы, т.к. являются ключами), изменения, внесенные при денормализации БД были устранены:

create index ind\_start on factualVisit2 (start\_of\_vis);

create index ind\_end on factualVisit2 (end\_of\_vis);

Время выполнения запроса было проверено на тех же наборах данных: 532ms, 406ms, 547ms, 484ms, 719ms

Среднее время выполнения запроса теперь: 538ms. Видно, что время выполнения запроса уменьшилось примерно на 25%.

План выполнения запроса:

Plan

PLAN JOIN (FACTUALVISIT2 INDEX (IND\_END, IND\_START), ROOM\_BUILD INDEX (RDB$18), ATTACHMENT INDEX (RDB$FOREIGN15, RDB$FOREIGN16))

Adapted Plan

PLAN JOIN (FACTUALVISIT2 INDEX (IND\_END, IND\_START), ROOM\_BUILD INDEX (INTEG\_61), ATTACHMENT INDEX (INTEG\_50, INTEG\_52))

**Вывод**

В ходу выполнения работы была произведена попытка оптимизировать запрос двумя способами: с помощью добавления индексов и попыткой денормализации БД. В обоих случаях время выполнения запроса уменьшилось. Из двух рассмотренных способов оптимизации наибольшую выгоду принесла денормализация БД, потому что у нас получилось оставить в запросе всего 2 задействованных таблицы, вместо изначальных трех, и таким способом существенно уменьшить количество анализируемых данных во время выполнения запроса. Добавление индексов также оказалось полезным при оптимизации запроса.

.