

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №1
по дисциплине «Базы данных»
Оптимизация SQL-запросов

Выполнил

студент гр. 43501/3

Е.А. Никитин

Преподаватель

А.В. Мяснов

«__»_____2015г.

Санкт-Петербург

2015

Цели 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

Ход работы:

Индекс (англ. *index*) — объект базы данных, 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.

В таблице сгенерировано большое количество данных (100000 записей). «Тяжёлый» запрос – поиск и вывод 5 станций, на которых больше всего входят и выходят пассажиры. Текст запроса (создается представление):

```
create view top5stations
as
select first 5 T1.st as st_name, T1.num + T2.num_start as summ from
(select ticket.stop_st as st, count(all ticket.stop_st) as num
from ticket
group by st order by num desc)T1,
(select ticket.start_st as st_start, count(all ticket.start_st) as num_start
from ticket
group by st_start order by num_start desc)T2
where T1.st = T2.st_start
order by summ desc;
```

Номер запроса	Время выполнения
---------------	------------------

1	1s 154ms
2	1s 14ms
3	1s 108ms
4	1s 45ms
5	1s 217ms
Среднее время: 1s 108ms	

Добавим индексы для полей stop_st и start_st таблицы ticket:

```
create index ticket_stop on ticket (stop_st);
```

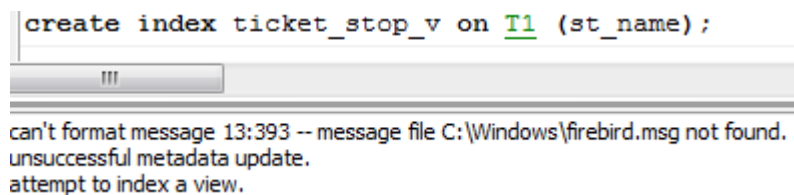
```
create index ticket_start on ticket (start_st);
```

Выполним тот же запрос:

Номер запроса	Время выполнения
1	1s 279ms
2	1s 170ms
3	1s 170ms
4	1s 155ms
5	1s 154ms
Среднее время: 1s 186ms	

Среднее время выполнения запроса немного изменилось, но не в лучшую сторону. Можно сделать вывод, что в данном случае индексы не помогут в оптимизации, т.к. нет конкретного поиска по значениям полей таблицы ticket – запрос создает выборку из 5 полей, которые в свою очередь выбираются из двух других выборок T1 и T2.

Некоторые СУБД расширяют возможности индексов введением возможности создания индексов по столбцам представлений или индексов по выражениям. Но СУБД Firebird 2.5 не позволяет так делать:



```
create index ticket_stop_v on T1 (st_name);
```

can't format message 13:393 -- message file C:\Windows\firebird.msg not found.
unsuccessful metadata update.
attempt to index a view.

Второй способ оптимизации – модификация запроса.

В данном запросе были найдены ненужные действия, а именно сортировки в формировании выборок T1 и T2. Текст измененного запроса:

```
CREATE OR ALTER VIEW TOP5STATIONSMOD(
    ST_NAME,
    SUMM)
AS
select first 5 T1.st as st_name, T1.num + T2.num_start as summ from
(select ticket.stop_st as st, count(all ticket.stop_st) as num
from ticket
```

```

group by st )T1,
(select ticket.start_st as st_start, count(all ticket.start_st) as num_start
from ticket
group by st_start )T2
where T1.st = T2.st_start
order by summ desc;

```

Выполним тот же запрос:

Номер запроса	Время выполнения
1	983ms
2	998ms
3	796ms
4	967ms
5	920ms
Среднее время: 993ms	

Среднее время уменьшилось примерно на 15%, что говорит об эффективности модификации.

Вывод:

Наиболее популярные оптимизации SQL-запросов с использованием:

- индексов
- модификации запроса
- создания собственного плана запроса
- денормализации БД

Не всегда подходит денормализация, т.к. это приводит к избыточности данных, также некоторые СУБД работают с сильно нормализованными БД. Индексирование помогает, когда нужно искать какие-либо данные, такие запросы наиболее частые.