Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

По дисциплине «Базы данных»

«Оптимизация SQL запросов»

Работу выполнил студент группы № 43501/4 Смирнов Л.А. \_\_\_\_\_\_\_

Работу принял преподаватель Мяснов А.В. \_\_\_\_\_\_\_

Санкт-Петербург

2016

1. **Цель работы**

Получить практические навыки создания эффективных SQL-запросов.

1. **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

**3. Выполнение работы**

В таблицах нагенерированно по 100к записей в каждой.

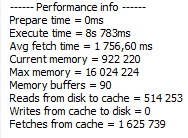
В качестве исследуемого запроса будем использовать следующий запрос:

|  |
| --- |
| *Вывести 5 призывников, которые провели наибольшее количество дней в мероприятиях не связанных с призывом.* |

SQL запрос:

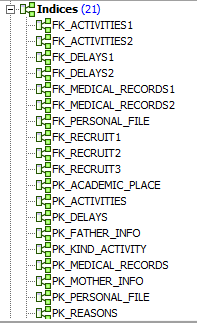
|  |
| --- |
| select p3.name, sum(p1.activity\_date\_off - p1.activity\_date) from activities as p1, personal\_file as p2, recruit as p3  where p2.id\_file = p1.id\_file and p1.activity\_date\_off is not NULL and p2.id\_recruit = p3.id\_recruit and p1.id\_activity in  (  select p1.id\_activity from kind\_activity as p1  where p1.kind\_activity not like 'Military%'  )  group by p3.name order by sum(p1.activity\_date\_off - p1.activity\_date) DESC, p3.name asc rows 1 to 5  ; |

Без оптимизации производительность запроса следующая:



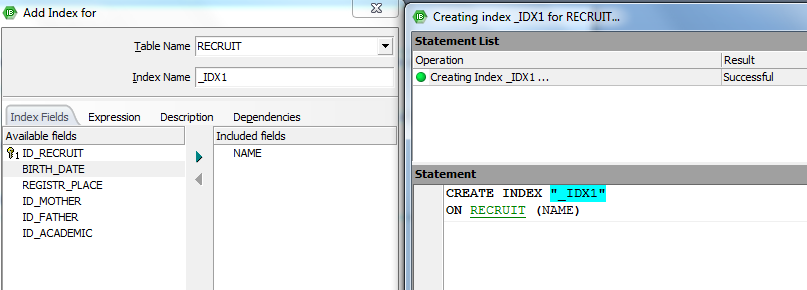
**Индексы**

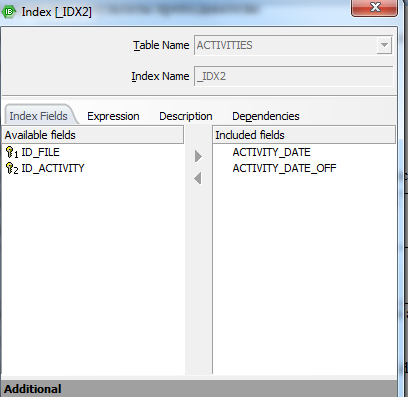
По умолчанию Firebird создает индексы для первичных и внешних ключей.



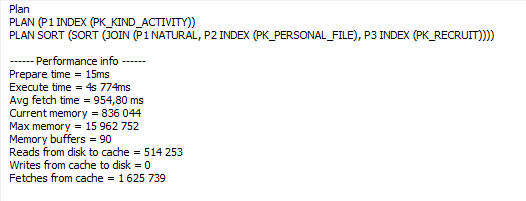
В приведенном выше запросе индексированы почти все атрибуты кроме атрибутов Recruit.name и Activities.Activity\_date/Activity\_date\_off. Создадим пользовательские индексы.

**Атрибут Recruit.Name**

****

**Атрибут Activities.Activity\_date, Activities.Activity\_date\_off**

Повторно выполним запрос и увидим, что время выполнения уменьшилось примерно на 4 секунды.

****

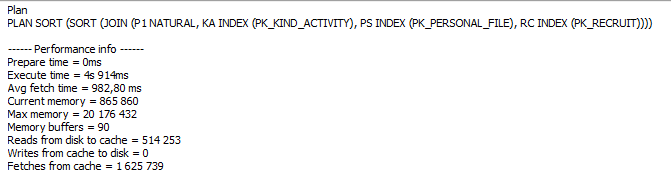
**Использование оператора JOIN**

Перепишем наш запрос с использованием оператора inner join:

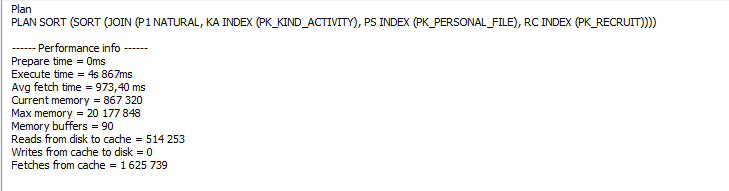
|  |
| --- |
| select rc.name, sum(p1.activity\_date\_off - p1.activity\_date) from activities as p1  inner join personal\_file ps on p1.id\_file = ps.id\_file  inner join recruit rc on ps.id\_recruit = rc.id\_recruit  inner join kind\_activity ka on p1.id\_activity = ka.id\_activity  where ka.kind\_activity not like 'Military%' and p1.activity\_date\_off is not NULL  group by rc.name order by sum(p1.activity\_date\_off - p1.activity\_date) DESC, rc.name asc rows 1 to 5  ; |

Оценим производительность:

1) без использования пользовательского индекса

****

2) с использованием пользовательского индекса



Использование оператора привело к большому повышению производительности. Но использование пользовательского индекса почти не повысило производительность.

**Вывод**

В ходе работы получили навыки создания эффективных SQL-запросов.

Индексы в большинстве случаев позволяют увеличить производительность запросов. Две базовые функции индексов:

* увеличение скорости доступа к данным
* поддержка уникальности данных

Несмотря на достоинства, индексы так же имеют и ряд недостатков. Первый из них – индексы занимают дополнительное место на диске и в оперативной памяти. Каждый раз когда мы создаем индекс, мы сохраняете ключи в порядке убывания или возрастания, которые могут иметь многоуровневую структуру. И чем больше/длиннее ключ, тем больше размер индекса. Второй недостаток – замедляются операции вставки, обновления и удаления записей.

Можно дать следующие советы по их использованию:

1) Не создавайте ненужные индексы

Помните, каждый индекс ускоряя выборку данных, замедляет добавление/изменение

/удаление записей.

2) Не стройте индексы по полям, содержащим наименьшее количество разных

значений (например только 0 и 1, пол мужской/женский и т.п.)

Если Вам все же надо такой индекс, то добавьте к нему еще одно поле, которое

сделает индекс более "уникальным".

3) Не стройте индексы по полям PK, FK, UNIQUE

IB автоматически создает соответствующие индексы при создании этих constraints.

4) Окончательное решение о применении конкретного индекса необходимо принимать

после тестирования производительности запроса.

Денормализация позволяет выделить данные, которые необходимы для запроса,

что увеличивает скорость его выполнения. Метод имеет недостаток – необходимо

дублировать имеющиеся данные в соответствующей таблице.

IB Expert автоматически адаптирует план запроса для увеличения его

производительности, что облегчает работу программиста.