Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №1**

**Дисциплина**: Базы данных

**Оптимизация SQL-запросов**

Выполнил студент гр. 43501/4 Хрусталева М.С.

Преподаватель: Мяснов А. В.

Санкт-Петербург

2016

## 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

### Индексы

**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).

Создание индексов:

CREATE INDEX <index\_name> ON <table\_name> (<column\_name> [DESC],…)

### Модификация запросов

Явное указание порядка обхода таблиц с помощью конструкций JOIN.

### Планы выполнения запросов

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 БД

Введение функциональной избыточности (дополнительных полей) для ускорения запросов.

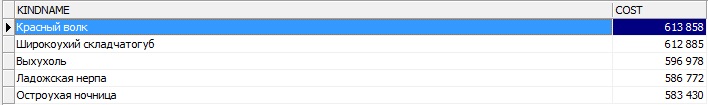
**3. Выполнение работы**

1. **Проведено ознакомление со способами оптимизации SQL-запросов с использованием:** 
   1. индексов
   2. модификации запроса
   3. создания собственного плана запроса
   4. денормализации БД
2. **Нагенерированы данные во всех таблицах.**
3. **Оценивание производительности запроса.**

Вид запроса: **5 видов, на питание которых потрачено больше всего денег за выбранный период времени.**

|  |
| --- |
| select first 5 kind.name as KindName,  sum(eat.amount\*(deliveries.cost/linkprod.amount)) as Cost  from kind, eat, deliveries, animals, linkprod where  deliveries.date\_del between '2000-10-31' and '2016-10-31' and -- за выбранный период  eat.deliv\_id = deliveries.iddeliveries and  animals.idanimals = eat.animal\_id and  animals.kind\_id = kind.idkind and  deliveries.iddeliveries = linkprod.deliver\_id  group by kind.name order by Cost desc; |

Результат запроса:



Скрипт выполняется на большом объеме данных(~100000 записей в таблице).

Время выполнения запроса.

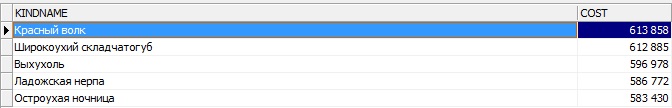
|  |  |
| --- | --- |
| № Запроса | Время выполнения запроса |
| 1 | 2s 574ms |
| 2 | 2s 480ms |
| 3 | 2s 450ms |
| 4 | 2s 402ms |
| 5 | 2s 434ms |
| Среднее | 2s 468ms |

1. **Выполнение оптимизации**

Изменим запрос с использованием конструкции join.

|  |
| --- |
| select first 5 kind.name as KindName,  sum(eat.amount\*(deliveries.cost/linkprod.amount)) as Cost  from eat, animals, kind, linkprod  join deliveries on eat.deliv\_id = deliveries.iddeliveries  where  deliveries.date\_del between '2000-10-31' and '2016-10-31' and -- за выбранный период  animals.idanimals = eat.animal\_id and  animals.kind\_id = kind.idkind and  deliveries.iddeliveries = linkprod.deliver\_id  group by kind.name order by Cost desc; |

Результат запроса:



Время выполнения запроса.

|  |  |
| --- | --- |
| № Запроса | Время выполнения запроса |
| 1 | 2s 402ms |
| 2 | 2s 450ms |
| 3 | 2s 433ms |
| 4 | 2s 403ms |
| 5 | 2s 418ms |
| Среднее | 2s 421.2ms |

Время выполнения уменьшилось на 46.8ms.

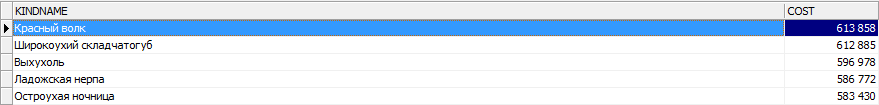
**Добавим индексы в таблицы eat, deliveries, linkprod, kind для атрибутов amount, cost, amount, name:**

|  |
| --- |
| create index index\_amount on eat (amount);  create index ind\_cost on deliveries (cost);  create index ind\_amount on linkprod(amount);  create index ind\_name on kind (name); |

Запрос:

|  |
| --- |
| select first 5 kind.name as KindName,  sum(eat.amount\*(deliveries.cost/linkprod.amount)) as Cost  from kind, eat, deliveries, animals, linkprod where  deliveries.date\_del between '2000-10-31' and '2016-10-31' and -- за выбранный период  eat.deliv\_id = deliveries.iddeliveries and  animals.idanimals = eat.animal\_id and  animals.kind\_id = kind.idkind and  deliveries.iddeliveries = linkprod.deliver\_id  group by kind.name order by Cost desc  ; |

Результат запроса:



Время выполнения запроса.

|  |  |
| --- | --- |
| № Запроса | Время выполнения запроса |
| 1 | 2s 246ms |
| 2 | 2s 277ms |
| 3 | 2s 293ms |
| 4 | 2s 293ms |
| 5 | 2s 293ms |
| Среднее | 2s 280.4ms |

Время выполнения запроса уменьшилось на 187.6 ms.

**Теперь изменим запрос с использованием конструкции join(индексы присутствуют).**

|  |
| --- |
| select first 5 kind.name as KindName,  sum(eat.amount\*(deliveries.cost/linkprod.amount)) as Cost  from eat, animals, kind, linkprod  join deliveries on eat.deliv\_id = deliveries.iddeliveries  where  deliveries.date\_del between '2000-10-31' and '2016-10-31' and -- за выбранный период  animals.idanimals = eat.animal\_id and  animals.kind\_id = kind.idkind and  deliveries.iddeliveries = linkprod.deliver\_id  group by kind.name order by Cost desc; |

Время выполнения запроса.

|  |  |
| --- | --- |
| № Запроса | Время выполнения запроса |
| 1 | 2s 293ms |
| 2 | 2s 309ms |
| 3 | 2s 324ms |
| 4 | 2s 277ms |
| 5 | 2s 277ms |
| Среднее | 2s 296ms |

Время выполнения запроса уменьшилось на 172ms, но этот результат хуже, чем в предыдущем опыте с использованием только индексов. Зато производительность запроса отдельно с конструкцией join заметно ниже.

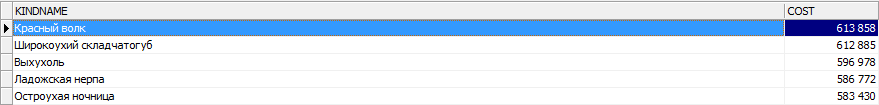
**Создадим новую, денормализованную таблицу deliveries\_t,** которая будет копией таблицы deliveries и в которой заранее отметим строки, удовлетворяющие запросу deliveries.date\_del between '2000-10-31' and ‘2016-10-31'.

|  |
| --- |
| alter table deliveries\_t  add time\_const int;  update deliveries\_t  set time\_const=1  where deliveries\_t.date\_del BETWEEN '2000-10-31' AND '2016-10-31'  ; |

Запрос в денормализованную таблицу

|  |
| --- |
| select first 5 kind.name as KindName,  sum(eat.amount\*(deliveries\_t.cost/linkprod.amount)) as Cost  from kind, eat, deliveries\_t, animals, linkprod where  deliveries\_t.time\_const = 1 and  eat.deliv\_id = deliveries\_t.iddeliveries and  animals.idanimals = eat.animal\_id and  animals.kind\_id = kind.idkind and  deliveries\_t.iddeliveries = linkprod.deliver\_id  group by kind.name order by Cost desc; |

Результат запроса:



Время выполнения запроса.

|  |  |
| --- | --- |
| № Запроса | Время выполнения запроса |
| 1 | 2s 137ms |
| 2 | 2s 200ms |
| 3 | 2s 309ms |
| 4 | 2s 153ms |
| 5 | 2s 137ms |
| Среднее | 2s 187.2ms |

Время выполнения запроса улучшилось, теперь запрос выполняется на 280.8ms быстрее.

Выводы

Произведена попытка оптимизировать запрос тремя способами: индексами, денормализацией и изменением структуры запроса. К уменьшению времени запроса привели все три способа. При совместном использовании модификации запроса и индексов результат оказался хуже, чем только с индексами.

Использование денормализации помогло ускорить выполнение запроса на 280.8ms, что в данных условиях стало наилучшим вариантом для оптимизации. Хотя в некоторых случаях использование денормализации может быть необходимо для повышения производительности, нужно помнить, что денормализация увеличивает число повторений данных, следовательно, возрастает используемое дисковое пространство, а также усложняется изменение БД при изменении окружающей среды.

Использование индексов также позволило снизить время выполнения запроса. Индексы ускоряют поиск данных, но замедляют операции добавления, удаления и модификации данных в индексируемых столбцах, поскольку при каждом изменении данных в такой таблице приходится перестраивать индекс. При создании большого количества индексов, размер индексного файла может быстро достичь максимального размера.

Таким образом, мы научились простым способам оптимизации запросов.