Анализ и оптимизация на софтуерни приложения

Александър Пенев

Васил Василев

Оптимизация на SQL

Съдържание

- СУБД, SQL, Таблици, Индекси, ...
- Анализ на производителността: EXPLAIN, PLAN, SHOW STATUS, SHOW PROCESSLIST, Профайлинг инструменти, ...
- Оптимизация – Основни подходи
- Оптимизация на JOIN заявки
- Оптимизация на Sub-SELECT
- Схема на БД
- Денормализация и други подходи

Какво е целта?

- Да се ускорят SQL заявките
- При възможност да се запази големината на БД

Подход

1. Анализират се производителността и нещата, които ѝ влияят:
   Винаги извършвайте анализите върху БД попълнена с реални или генерирани данни с обем и съдържание близко до реално очакваното!!! От значение са и параметрите на SQL заявките.
   - SQL кода на заявките
   - Код на приложението – ако е възможно да се преместят обработките в SQL заявката
   - Схемата на БД
   - Данните
   - Конфигурация на СУБД
   - СУБД. Друго?
2. Ако производителността не е приемлива се правят промени по по-горните
3. Отиваме на 1.

SQL код на заявките

- SQL кода може да е ненужно сложен
- Извличане на по-малко полета, ако е възможно
- Извличане на по-малко редове, ако е възможно
- Премахване на Sub-SELECT-тите
- Има ли нужда от сортировките?
- ...

SQL SELECT

```
SELECT …, …, …
   FROM …, …, … JOIN …
   WHERE … and …
   ORDER BY …, …, …
   GROUP BY …, …, …
   HAVING …, …, …
```

полета, агрегации

таблици и сливане

филтриране на записи

сортиране

групиране

филтриране след агрегация

Схема на БД

- Таблици
- Нормална форма
- Индекси – добавяне, а понякога и премахване
- Съхранени процедури
- …

Пример – Филми (MySQL)

```
CREATE TABLE `title` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` text NOT NULL,
  `kind_id` int(11) NOT NULL,
  `production_year` int(11) DEFAULT NULL,
  `episode_of_id` int(11) DEFAULT NULL,
  `season_nr` int(11) DEFAULT NULL,
  `episode_nr` int(11) DEFAULT NULL,
  `series_years` varchar(49) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB;
```

Пример – Филми (MySQL) ~ 3s.

```
mysql> EXPLAIN SELECT id,title,production_year
   FROM title WHERE title  = 'Star Wars'
   ORDER BY production_year\G
******************** 1. row ********************
           id: 1
  select_type: SIMPLE
        table: title
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1900087
        Extra: Using where; Using filesort;
1 row in set (0.00 sec)
```

Пример – Филми (MySQL)

```
mysql> ALTER TABLE title ADD INDEX (title(50));

Query OK, 1543719 rows affected (1 min 19 sec)
Records: 1543719  Duplicates: 0  Warnings: 0
```

Пример – Филми (MySQL) < 0.01s

```
mysql> EXPLAIN SELECT id,title,production_year
   FROM title WHERE title = 'Star Wars'
   ORDER by production_year\G
******************** 1. row ********************
           id: 1
  select_type: SIMPLE
        table: title
         type: ref
possible_keys: title
          key: title
      key_len: 152
          ref: const
         rows: 4
        Extra: Using where; Using filesort;
1 row in set (0.14 sec)
```

Пример – Филми (MySQL) < 0.01s

```
mysql> EXPLAIN SELECT id,title,production_year
   FROM title WHERE id = 55327\G
******************** 1. row ********************
           id: 1
  select_type: SIMPLE
        table: title
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: const
         rows: 1
        Extra:
1 row in set (0.06 sec)
```

Пример – Филми (MySQL)

```
mysql> EXPLAIN SELECT id,title,production_year
   FROM title WHERE title LIKE 'Star%'\G
******************** 1. row ********************
           id: 1
  select_type: SIMPLE
        table: title
         type: range
possible_keys: title
          key: title
      key_len: 152
          ref: NULL
         rows: 98
        Extra: Using where
1 row in set (0.00 sec)
```

Пример – Филми (MySQL) обаче ~ 3s

```
mysql> EXPLAIN SELECT id,title,production_year
   FROM title WHERE title LIKE '%Fiction'\G
******************** 1. row ********************
           id: 1
  select_type: SIMPLE
        table: title
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1900087
        Extra: Using where;
1 row in set (0.00 sec)
```

Индекси

- В повечето случаи помагат за по-бързо изпълнение на заявките (най-вече филтриране в SELECT)
- Също помагат и за: Избягване на сортиране, Необходимост от създаване на временни таблици, и др.

- Увеличават размера на БД
  Това невинаги е допустимо
- Може да забавят някои заявки (INSERT...)
  Трябва да се обновят и индексите. Ако те са много това забавя
- Данните в БД имат значение
  Повечето съвременни СУБД пазят и обновяват информация за „селективността“ на индексите. Плана на заявките се определя динамично и може една и съща заявка да се изпълни по различен начин (с използване на различни индекси и др.) в зависимост от селективността на наличните за използване индекси

Оптимизация на JOIN заявки

- JOIN заявки
- Сливане на таблици

  Това понякога води до денормализация на БД. Но не винаги това е проблем.
- Преобразуване на Sub-SELECT в JOIN

  Невинаги това е възможно или може да не води до подобрение.

Пример JOIN (MySQL)

```
mysql> EXPLAIN SELECT person_info.*
    FROM name INNER JOIN person_info ON (name.id=person_info.person_id) AND
    name.name='Ford, Harrison'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: person_info
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1885354
        Extra:
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: name
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: imdb.person_info.person_id
         rows: 1
        Extra: Using where
```

Пример JOIN (MySQL)

```
mysql> EXPLAIN SELECT person_info.*
    FROM name INNER JOIN person_info ON (name.id=person_info.person_id) AND
    name.name='Ford, Harrison'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: name
         type: ref
possible_keys: PRIMARY,name
          key: name
      key_len: 152
          ref: const
         rows: 1
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: person_info
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1885354
        Extra: Using where; Using join buffer
2 rows in set (0.00 sec)
```

След:

```
ALTER TABLE name
ADD INDEX (name(50));
```

Пример JOIN (MySQL)

```
mysql> EXPLAIN SELECT person_info.*
    FROM name INNER JOIN person_info ON (name.id=person_info.person_id) AND
    name.name='Ford, Harrison'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: name
         type: ref
possible_keys: PRIMARY,name
          key: name
      key_len: 152
          ref: const
         rows: 1
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: person_info
         type: ref
possible_keys: person_id
          key: person_id
      key_len: 4
          ref: imdb.name.id
         rows: 2
        Extra:
2 rows in set (0.01 sec)
```

След:

```
ALTER TABLE person_info
ADD INDEX (person_id);
```

Пример JOIN 2 (MySQL) ~ 4min

```
mysql> EXPLAIN SELECT name.* FROM name
      INNER JOIN cast_info ON name.id=cast_info.person_id
      INNER JOIN char_name ON cast_info.person_role_id=char_name.id
      WHERE char_name.name = 'James Bond'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: cast_info
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 22743540
        Extra:
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: name
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: imdb.cast_info.person_id
         rows: 1
        Extra:
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: char_name
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: imdb.cast_info.person_role_id
         rows: 1
        Extra: Using where
3 rows in set (0.07 sec)
```

Пример JOIN 2 (MySQL) ~ 1.5min

```
mysql> EXPLAIN SELECT name.* FROM name
      INNER JOIN cast_info ON name.id=cast_info.person_id
      INNER JOIN char_name ON cast_info.person_role_id=char_name.id
      WHERE char_name.name = 'James Bond'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: char_name
         type: ref
possible_keys: PRIMARY,name_idx
          key: name_idx
      key_len: 152
          ref: const
         rows: 1
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: cast_info
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 22743540
        Extra: Using where; Using join buffer
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: name
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: imdb.cast_info.person_id
         rows: 1
        Extra:
3 rows in set (0.24 sec)
```

Първо да филтрираме по име:

```
ALTER TABLE char_name
ADD index name_idx (name(50));
```

Пример JOIN 2 (MySQL) < 0.01s

```
mysql> EXPLAIN SELECT name.* FROM name
      INNER JOIN cast_info ON name.id=cast_info.person_id
      INNER JOIN char_name ON cast_info.person_role_id=char_name.id
      WHERE char_name.name = 'James Bond'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: char_name
         type: ref
possible_keys: PRIMARY,name_idx
          key: name_idx
      key_len: 152
          ref: const
         rows: 1
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: cast_info
         type: ref
possible_keys: person_role_id_person_id
          key: person_role_id_person_id
      key_len: 5
          ref: imdb.char_name.id
         rows: 4
        Extra: Using where; Using index
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: name
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: imdb.cast_info.person_id
         rows: 1
        Extra:
3 rows in set (0.17 sec)
```

След това да ускорим първия JOIN:

```
ALTER TABLE cast_info
ADD INDEX person_role_id_person_id
(person_role_id, person_id);
```

Вече се използва индекса

Пример Sub-SELECT (MySQL) ~ 5min

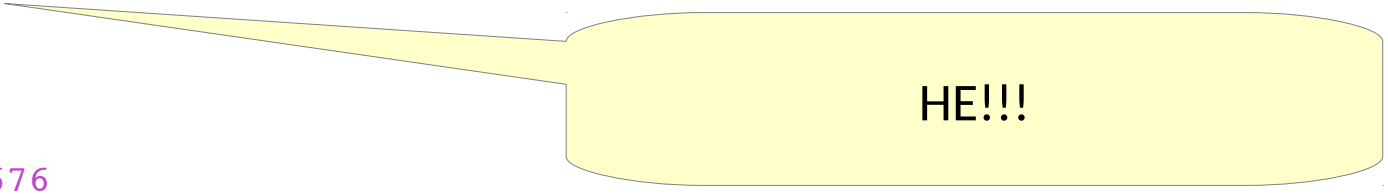
```
mysql> EXPLAIN SELECT * FROM title
    WHERE kind_id IN (SELECT id FROM kind_type WHERE kind='video game')\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: title
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1567676
        Extra: Using where
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: kind_type
         type: const
possible_keys: PRIMARY,kind_id
          key: kind_id
      key_len: 47
          ref: const
         rows: 1
        Extra: Using index
2 rows in set (0.00 sec)
```

Лошо!

Добавяме индекс по title.kind_id

Пример Sub-SELECT (MySQL) ~ 5min

```
mysql> EXPLAIN SELECT * FROM title
    WHERE kind_id IN (SELECT id FROM kind_type WHERE kind='video game')\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: title
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1567676
        Extra: Using where
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: kind_type
         type: const
possible_keys: PRIMARY,kind_id
          key: kind_id
      key_len: 47
          ref: const
         rows: 1
        Extra: Using index
2 rows in set (0.00 sec)
```



Пример Sub-SELECT (MySQL) ~ 0.07s

```
mysql> EXPLAIN SELECT * FROM title
    WHERE kind_id = (SELECT id FROM kind_type WHERE kind='video game')\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: title
         type: ref
possible_keys: k
          key: k
      key_len: 4
          ref: const
         rows: 8502
        Extra: Using where
*************************** 2. row ***************************
           id: 2
  select_type: SUBQUERY
        table: kind_type
         type: const
possible_keys: kind_id
          key: kind_id
      key_len: 47
          ref:
         rows: 1
        Extra: Using index
2 rows in set (0.10 sec)
```

Много по-добре след
замяна на 'IN' с '='.
Защо може да го направим?

Пример Sub-SELECT (MySQL) ~ 0.06s

```
mysql> EXPLAIN SELECT title.* FROM title
    INNER JOIN kind_type ON (title.kind_id = kind_type.id)
    WHERE kind_type.kind IN ('video game')\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: kind_type
         type: const
possible_keys: PRIMARY,kind_id
          key: kind_id
      key_len: 47
          ref: const
         rows: 1
        Extra: Using index
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: title
         type: ref
possible_keys: kind_id
          key: kind_id
      key_len: 4
          ref: const
         rows: 8502
        Extra:
2 rows in set (0.00 sec)
```

Още по-добре да се премахне Sub-SELECT. И след:

ALTER TABLE title
ADD KEY (kind_id);

Други оптимизации

- Денормализация
- Използване (или емулация) на Hash индекси
- Явно задаване на плана
- Внимание да не се прекомерно идексира (прекалено много индекси)
- Анализ на поведението на СУБД
- Клъстери

Извод

Внимателно със СУБД!

Why avoid indexes (cont.)

- We benchmarked this on a different schema:

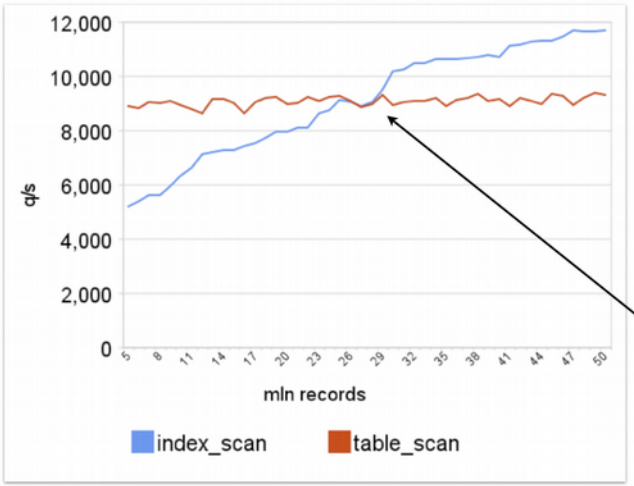


Table scan has a relatively fixed cost (red line).

The index has completely different effectiveness depending on how much it can filter.

Hopefully MySQL switches at the right point.