ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ EXPLAIN В ОПТИМИЗАЦИИ ЗАПРОСОВ И СХЕМЫ БАЗЫ ДАННЫХ

АВТОР: ГУСЕВА ТАТЬЯНА

КОМАНДА EXPLAIN

EXPLAIN – это один из самых мощных инструментов, предоставленных в ваше распоряжение для понимания MySQL-запросов и их оптимизации.

КОМАНДА EXPLAIN : СИНТАКСИС

- **EXPLAIN имя_таблицы** (является синонимом операторов DESCRIBE имя_таблицы и SHOW COLUMNS FROM имя_таблицы)

- **EXPLAIN SELECT опции_выборки** (MySQL сообщит о том, как будет производиться обработка SELECT, и предоставит информацию о порядке и методе связывания таблиц)

КОМАНДА EXPLAIN : ПРИМЕР

```
1 explain select t.id, pl.last_name, pl.first_name, pw.nickname, p.phone, p.skype
2 from users t
3 join profile p ON p.user__id=t.id
4 join profile_lang pl ON pl.user__id=t.id
5 join profile_worldgmn pw ON pw.user__id=t.id
6 where t.username !='superadmin'
```

Результат использования EXPLAIN:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	pl	ALL	*NULL*	*NULL*	*NULL*	*NULL*	5673	Using where
1	SIMPLE	t	eq_ref	PRIMARY,UK_users_username	PRIMARY	4	guseva_olimp24.pl.user__id	1	Using where
1	SIMPLE	p	ref	p_user__id_index	p_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*
1	SIMPLE	pw	ref	pw_user__id_index	pw_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*

ЧТО ВЫВОДИТ EXPLAIN?

- **id** - порядковый идентификатор каждого SELECT, находящегося внутри запроса (в случае использования вложенных подзапросов)
- **select_type**– тип SELECT запроса. Возможные значения:
  - SIMPLE – запрос содержит простую выборку без подзапросов и UNION'ов
  - PRIMARY – запрос является внешним запросов в JOIN
  - DERIVED – запрос SELECT является частью подзапроса внутри выражения FROM
  - SUBQUERY – первый SELECT в подзапросе
  - DEPENDENT SUBQUERY - первый SELECT, зависящий от внешнего подзапроса
  - UNCACHEABLE SUBQUERY – некешируемый подзапрос
  - UNION – SELECT является вторым или последующим в UNION
  - DEPENDENT UNION – SELECT является вторым или последующим запросом в UNIONи зависит от внешних запросов/li>
  - UNION RESULT – SELECT является результатом UNION'а

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	pl	ALL	*NULL*	*NULL*	*NULL*	*NULL*	5673	Using where
1	SIMPLE	t	eq_ref	PRIMARY,UK_users_username	PRIMARY	4	guseva_olimp24.pl.user__id	1	Using where
1	SIMPLE	p	ref	p_user__id_index	p_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*
1	SIMPLE	pw	ref	pw_user__id_index	pw_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*

ЧТО ВЫВОДИТ EXPLAIN?

- **table** – таблица, которой относится текущая строка
- **type** – тип связывания таблиц. Это один из самых важных столбцов в результате, потому что по нему можно вычислить потерянные индексы или понять, как можно улучшить запрос. Возможные значения:
  - system – таблица содержит только одну строку (системная таблица);
  - const - таблица содержит не более одной соответствующей строки, которая будет считываться в начале запроса. Поскольку имеется только одна строка, оптимизатор в дальнейшем может расценивать значения этой строки в столбце как константы. Таблицы const являются очень быстрыми, поскольку они читаются только однажды;
  - eq_ref - для каждой комбинации строк из предыдущих таблиц будет считываться одна строка из этой таблицы. Это наилучший возможный тип связывания среди типов, отличных от const. Данный тип применяется, когда все части индекса используются для связывания, а сам индекс - UNIQUE или PRIMARY KEY;
  - ref - из этой таблицы будут считываться все строки с совпадающими значениями индексов для каждой комбинации строк из предыдущих таблиц. Тип ref применяется, если для связывания используется только крайний левый префикс ключа, или если ключ не является UNIQUE или PRIMARY KEY (другими словами, если на основании значения ключа для связывания не может быть выбрана одна строка). Этот тип связывания хорошо работает, если используемый ключ соответствует только нескольким строкам;

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	pl	ALL	*NULL*	*NULL*	*NULL*	*NULL*	5673	Using where
1	SIMPLE	t	eq_ref	PRIMARY,UK_users_username	PRIMARY	4	guseva_olimp24.pl.user__id	1	Using where
1	SIMPLE	p	ref	p_user__id_index	p_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*
1	SIMPLE	pw	ref	pw_user__id_index	pw_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*

ЧТО ВЫВОДИТ EXPLAIN?

(type)

- fulltext – объединение, использующее полнотекстовый (FULLTEXT) индекс таблиц;
- ref_or_null – то же самое, что и ref, только содержащее строки со значением NULL в полях;
- index_merge – объединение, использующее список индексов для получения результата запроса;
- unique_subquery – результат подзапроса в выражении IN возвращает одну строку, используемую в качестве первичного ключа;
- index_subquery – то же самое, что и unique_subquery, только в результате больше одной строки;
- range – в запросе происходит сравнение ключевого поля с диапазоном значений (используются операторы BETWEEN,IN, >, >=);
- index – в процессе выполнения запроса сканируется только дерево индексов;
- all – в процессе выполнения запроса сканируются все таблицы. Это наихудший тип объединения и обычно указывает на отсутствие надлежащих индексов в таблице;

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	pl	ALL	*NULL*	*NULL*	*NULL*	*NULL*	5673	Using where
1	SIMPLE	t	eq_ref	PRIMARY,UK_users_username	PRIMARY	4	guseva_olimp24.pl.user__id	1	Using where
1	SIMPLE	p	ref	p_user__id_index	p_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*
1	SIMPLE	pw	ref	pw_user__id_index	pw_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*

ЧТО ВЫВОДИТ EXPLAIN?

- **possible_keys** – показаны возможные индексы, которые могут использоваться MySQL для поиска данных в таблице. На самом деле, значение этого столбца, очень часто помогает оптимизировать запросы. Если значение равно NULL, значит, никаких индексов не используется.
- **key** – отображается текущий ключ, используемый MySQL в данный момент. В этом столбце может отображаться индекс, отсутствующий в possible_keys. Оптимизатор запросов MySQL всегда пытается найти оптимальный ключ, который будет использоваться в запросе. При объединении нескольких таблиц, MySQL может использовать индексы, также не указанные в possible_keys.
- **key_len** – содержит длину ключа, выбранного оптимизатором запросов MySQL. Если значение key равно NULL, то key_len тоже NULL. По значению длины ключа можно определить, сколько частей составного ключа в действительности будет использовать MySQL.
- **ref** – показаны поля или константы, которые используются совместно с ключом, указанным в столбце key.
- **rows** – количество строк, которые анализируются MySQL в процессе запроса. Это еще один важный показатель, указывающий на необходимость **оптимизации запросов**, особенно тех, которые содержат JOIN и подзапросы.
- **extra** – содержит дополнительную информацию о процессе выполнения запроса. Если значениями этого столбца являются "Using temporary", "Using filesort" и т.п, то это говорит о

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	pl	ALL	*NULL*	*NULL*	*NULL*	*NULL*	5673	Using where
1	SIMPLE	t	eq_ref	PRIMARY,UK_users_username	PRIMARY	4	guseva_olimp24.pl.user__id	1	Using where
1	SIMPLE	p	ref	p_user__id_index	p_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*
1	SIMPLE	pw	ref	pw_user__id_index	pw_user__id_index	5	guseva_olimp24.pl.user__id	1	*NULL*

УСТРАНЕНИЕ ПРОБЛЕМ С ПРОИЗВОДИТЕЛЬНОСТЬЮ С ПОМОЩЬЮ EXPLAIN

Отображение строк 0 - 24 (34 всего, Запрос занял 2.5598 сек.)

```
1 select t.id, pl.last_name, pl.first_name, pw.referal_link, p.phone, p.skype
2 from users t
3 join profile p ON p.user__id=t.id
4 join profile_lang pl ON pl.user__id=t.id
5 join profile_worldgmn pw ON pw.user__id=t.id
6 where t.username !='superadmin' AND t.username !='admin' AND t.created_at BETWEEN '2015-09-09 13:09:31' AND '2016-05-10 11:08:16'
7 order by pw.user__id DESC
```

Использование EXPLAIN

Тип объединения равен ”**ALL**” (это наихудший вариант). Это значит, что MySQL не может найти ни одного ключа, который может участвовать в объединении, поэтому значение столбцов possible_keys и key равно NULL. Хуже всего то, что в процессе запроса MySQL будет сканировать все записи во всех таблицах, об этом говорит значение столбцов rows. При выполнении запроса будут просмотрены записи (15410 × 1 × 184443 × 193851), чтобы получить результат из 34 записей. Это действительно ужасно, и будет только хуже, когда количество записей в базе данных будет увеличиваться.

+ Параметры

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	pw	ALL	*NULL*	*NULL*	*NULL*	*NULL*	15410	Using where; Using temporary; Using filesort
1	SIMPLE	t	eq_ref	PRIMARY	PRIMARY	4	guseva_altclub.pw.user__id	1	Using where
1	SIMPLE	pl	ALL	*NULL*	*NULL*	*NULL*	*NULL*	184443	Using where; Using join buffer (Block Nested Loop)
1	SIMPLE	p	ALL	*NULL*	*NULL*	*NULL*	*NULL*	193851	Using where; Using join buffer (Block Nested Loop)

УСТРАНЕНИЕ ПРОБЛЕМ С ПРОИЗВОДИТЕЛЬНОСТЬЮ С ПОМОЩЬЮ EXPLAIN

Отображение строк 0 - 24 (34 всего, Запрос занял 0.0005 сек.)

```
1 select t.id, pl.last_name, pl.first_name, pw.referal_link, p.phone, p.skype
2 from users t
3 join profile p ON p.user__id=t.id
4 join profile_lang pl ON pl.user__id=t.id
5 join profile_worldgmn pw ON pw.user__id=t.id
6 where t.username !='superadmin' AND t.username !='admin' AND t.created_at BETWEEN '2015-09-09 13:09:31' AND '2016-05-10 11:08:16'
7 order by pw.user__id DESC
```

+ Параметры

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	pw	ALL	IDX_profile_worldgmn_user__id	*NULL*	*NULL*	*NULL*	15410	Using where; Using filesort
1	SIMPLE	t	eq_ref	PRIMARY,UK_users_username,IDX_user__id	PRIMARY	4	guseva_altclub.pw.user__id	1	Using where
1	SIMPLE	pl	ref	IDX_profile_lang_user__id	IDX_profile_lang_user__id	5	guseva_altclub.pw.user__id	1	*NULL*
1	SIMPLE	p	ref	IDX_profile_user__id	IDX_profile_user__id	5	guseva_altclub.pw.user__id	1	*NULL*

Спасибо
за внимание!