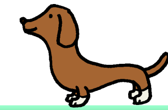
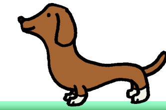
チューニンガソン5の復習 MySQL 5.6 新機能編

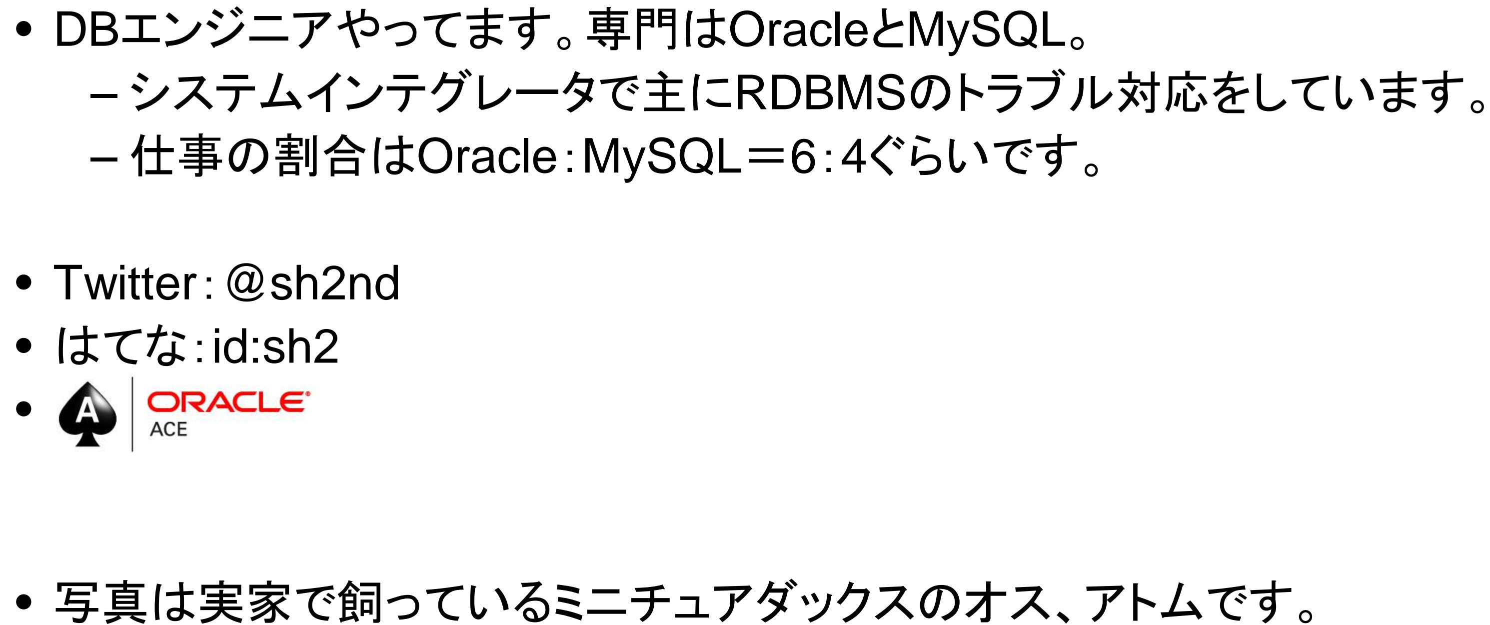


MyNA(日本MySQLユーザ会)会 2013年3月
2013/03/15 平塚 貞夫

自己紹介

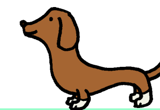


- DBエンジニアやってます。専門はOracleとMySQL。
  - システムインテグレータで主にRDBMSのトラブル対応をしています。
  - 仕事の割合はOracle：MySQL＝6：4ぐらいです。

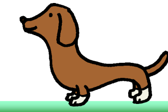
- Twitter：@sh2nd
- はてな：id:sh2
- 

- 写真は実家で飼っているミニチュアダックスのオス、アトムです。

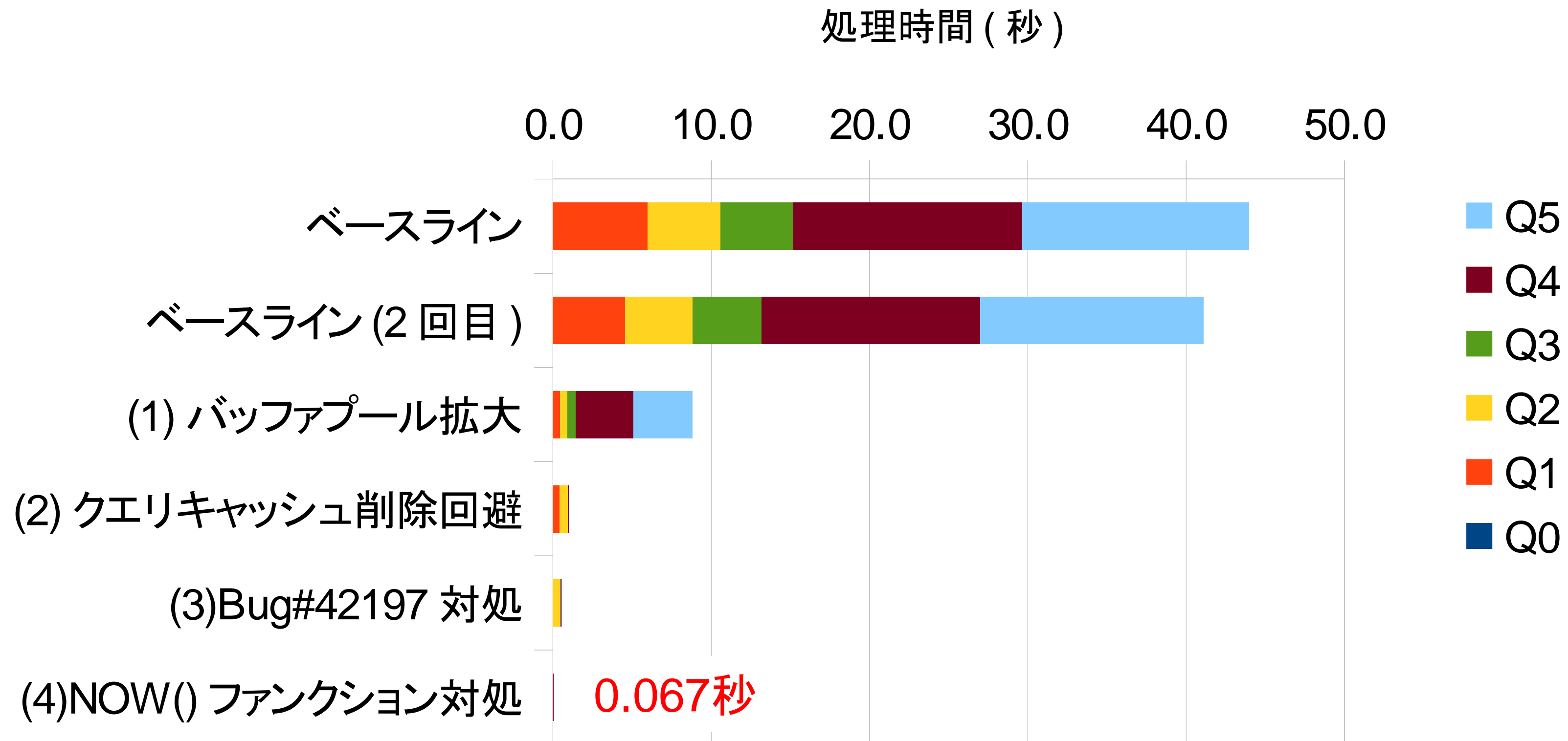
前回のあらすじ



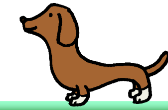
前回のあらすじ



- チューニンガソン5の復習 MySQL 5.5 チート編
  http://d.hatena.ne.jp/sh2/20130304



本日のお題



- 前回はMySQL Proxyを用いてRESET QUERY CACHEを回避するという、わりとひどいチューニングをしました。

- 今回は、できるだけ業務に応用可能な状態を維持しつつチューニングしていきます。

RESET QUERY CACHE

tgbench_mysql_2.py

TCP 3306

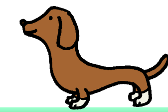
MySQL Proxy

FLUSH QUERY CACHE

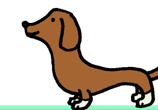
TCP 3307

MySQL Server

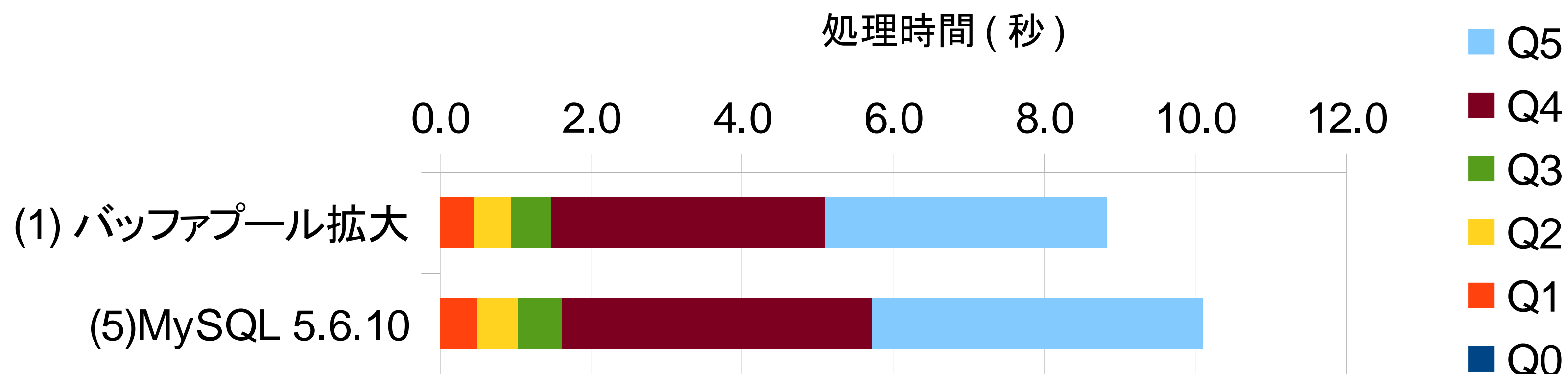
チューニングその5：MySQL 5.6にアップグレード



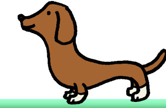
その5：MySQL 5.6にアップグレード



- MySQL Proxyを外して、MySQL 5.6.10にアップグレードしました。
- 遅くなりました。

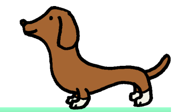


MySQL 5.6に対する性能改善の取り組み

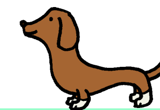


- 本件FacebookとPerconaが食いついています。
  - My MySQL is faster than your MySQL
    https://www.facebook.com/notes/mysql-at-facebook/my-mysql-is-faster-than-your-mysql/10151250402570933
  - Is MySQL 5.6 slower than MySQL 5.5?
    http://www.mysqlperformanceblog.com/2013/02/18/is-mysql-5-6-slower-than-mysql-5-5/
- Performance Schemaがデフォルトで有効化されたこと、メタデータロックの仕組みにCPUスケーラビリティがないことが原因と見られています。
  - MySQL Bugs: #68413:
    performance_schema overhead is at least 10%
    http://bugs.mysql.com/bug.php?id=68413
  - MySQL Bugs: #66473:
    MySQL 5.6.6-m9 has more mutex contention than MySQL 5.1
    http://bugs.mysql.com/bug.php?id=66473
- 今回はMySQL 5.5に戻したりはせず、このまま作業を進めます。

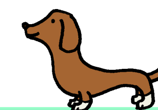
チューニングその6:インデックスの追加



その前に、MediaWikiのデータベース設計について



ER図の一部



page
- page_id INT(10)
- page_namespace INT(11)
- page_title VARCHAR(255)
- page_restrictions TINYBLOB
- page_counter BIGINT(20)
- page_is_redirect TINYINT(3)
- page_is_new TINYINT(3)
- page_random DOUBLE
- page_touched BINARY(14)
- page_latest INT(10)
- page_len INT(10)

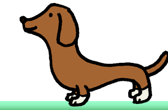
text
- old_id INT(10)
- old_text MEDIUMBLOB
- old_flags TINYBLOB

revision
- rev_id INT(10)
- rev_page INT(10)
- rev_text_id INT(10)
- rev_comment TINYBLOB
- rev_user INT(10)
- rev_user_text VARCHAR(255)
- rev_timestamp BINARY(14)
- rev_minor_edit TINYINT(3)
- rev_deleted TINYINT(3)
- rev_len INT(10)
- rev_parent_id INT(10)
- rev_sha1 VARBINARY(32)

user
- user_id INT(10)
- user_name VARCHAR(255)
- user_real_name VARCHAR(255)
- user_password TINYBLOB
- user_newpassword TINYBLOB
- user_newpass_time BINARY(14)
- user_email TINYTEXT
- user_touched BINARY(14)
- user_token BINARY(32)
- user_email_authenticated BINARY(14)
- user_email_token BINARY(32)
- user_email_token_expires BINARY(14)
- user_registration BINARY(14)
- user_editcount INT(11)

Relationships: page (1) — (∞) revision.rev_page; text (1) — (∞) revision.rev_text_id; user (1) — (∞) revision.rev_user

pageとrevisionの関係



「MySQL」の変更履歴 - Wikipedia - Mozilla Firefox

ファイル(F) 編集(E) 表示(V) 履歴(S) ブックマーク(B) GMarks(G) ツール(T) ヘルプ(H)

「MySQL」の変更履歴 - Wikipedia

ja.wikipedia.org/w/index.php?title=MySQL&action=history

アカウント作成 ログイン

ページ ノート　閲覧 編集 履歴表示 検索

page

「MySQL」の変更履歴

このページの記録を閲覧

ウィキペディア
フリー百科事典

メインページ
コミュニティ・ポータル
最近の出来事
新しいページ
最近の更新
おまかせ表示
練習用ページ
アップロード (ウィキメディア・コモンズ)

井戸端
お知らせ
バグの報告
寄付
ウィキペディアに関するお問い合わせ

履歴の閲覧

これ以前の年: 2013 これ以前の月: すべて タグ絞り込み: 表示

差分を表示するには比較したい版のラジオボタンを選択し、エンターキーを押すか、下部のボタンを押します。

凡例: **(最新)** = 最新版との比較、**(前)** = 直前の版との比較、**m** = 細部の編集 … 設定で未設定ならUTC

user

(最新 | 最古) (以後の50件 | 以前の50件) (20 | 50 | 100 | 250 | 500 件) を表示

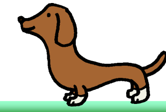
選択した版同士を比較

revision

- (最新 | 前) 2013年3月7日 (木) 18:00 Addbot (会話 | 投稿記録) m . . (17,939バイト) **(-903)** . . *(ボット: 言語間リンク 59 件をウィキデータ上の d:q850 に転記)* (取り消し)
- (最新 | 前) 2013年2月20日 (水) 05:16 ケイ20003 (会話 | 投稿記録) . . (18,842バイト) (-11) . . *(Shift_JISの表記修正)* (取り消し)
- (最新 | 前) 2013年2月8日 (金) 10:32 Space Travel (会話 | 投稿記録) . . (18,853バイト) (-2) . . (取り消し)
- (最新 | 前) 2013年2月8日 (金) 10:29 Space Travel (会話 | 投稿記録) . . (18,855バイト) (+432) . . (取り消し)
- (最新 | 前) 2012年11月22日 (木) 13:30 AvicBot (会話 | 投稿記録) m . . (18,423バイト) (0) . . *(r2.6.5) (ロボットによる 変更: sv:MySQL)* (取り消し)
- (最新 | 前) 2012年11月14日 (水) 11:08 Jkr2255 (会話 | 投稿記録) m . . (18,423バイト) (+32) . . *(→トランザクション: dab)* (取り消し)

完了

主なカラム



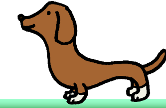
- page_is_redirect
  ほとんどが0ですが、別のページに飛ばされる場合に0以外が入ります。
  「World Baseball Classic」⇒「ワールド・ベースボール・クラシック」など。

- page_namespace
  ページの種別を示しています。
  0が通常ページ、6がファイル、14がカテゴリなど。0が81%を占めています。

- page_touched
  ページの更新日時を示しています。
  キャッシュを破棄する条件判定に用いられます。

- rev_user
  ページを最後に更新したユーザを示しています。
  0は匿名ユーザを表しており、全体の18%を占めています。

インデックスの張り方(1)

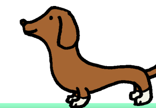


- クエリ1番を題材にして、インデックスの張り方をおさらいします。

- クエリ1番の意味は以下の通りです。
  「リダイレクトページではなく実際にコンテンツがあって、ファイルやカテゴリではなく通常のページであって、最後に更新をしたのが匿名ユーザであるページの数」

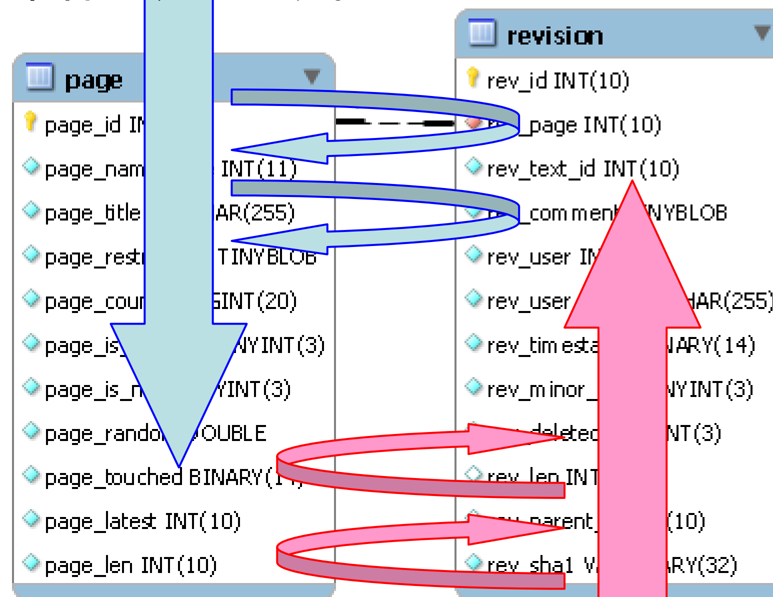
```
SELECT /* Q1 */ COUNT(*)
  FROM page p
  JOIN revision r
    ON p.page_id = r.rev_page
 WHERE p.page_is_redirect = 0
   AND p.page_namespace = 0
   AND r.rev_user = 0;
```

インデックスの張り方(2)



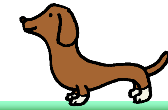
- SQL実行計画は、大きく分けて2つ候補があります。

(A) pageテーブルから条件を満たすレコードを抽出し、それぞれのレコードについて、内部結合されたrevisionテーブルからレコードを抽出して条件判定します。



(B) revisionテーブルから条件を満たすレコードを抽出し、それぞれのレコードについて、内部結合されたpageテーブルからレコードを抽出して条件判定します。

インデックスの張り方(3)



- (A)を狙う場合、以下のインデックスを作成します。

```
CREATE INDEX NEW_page_ix01 ON page (page_namespace, page_is_redirect);
CREATE INDEX NEW_revision_ix01 ON revision (rev_page);
```

```
SELECT /* Q1 */ COUNT(*)
  FROM page p
  JOIN revision r
    ON p.page_id = r.rev_page
 WHERE p.page_is_redirect = 0
   AND p.page_namespace = 0
   AND r.rev_user = 0;
```

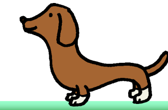
(4) レコード数を数えます。

(2) NEW_revision_ix01を利用して、pageと結合されたrevisionのレコードを抽出します。

(1) NEW_page_ix01を利用して、この条件を満たすレコードを抽出します。

(3) revisionテーブルにアクセスし条件判定します。

インデックスの張り方(4)



- (B)を狙う場合、以下のインデックスを作成します。

```
CREATE INDEX NEW_revision_ix02 ON revision (rev_user);
```

```
SELECT /* Q1 */ COUNT(*)
  FROM page p
  JOIN revision r
    ON p.page_id = r.rev_page
 WHERE p.page_is_redirect = 0
   AND p.page_namespace = 0
   AND r.rev_user = 0;
```

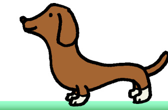
(4) レコード数を数えます。

(2) PRIMARYを利用して、revisionと結合されたpageのレコードを抽出します。

(3) pageテーブルにアクセスし条件判定します。

(1) NEW_revision_ix02を利用して、この条件を満たすレコードを抽出します。

インデックスの張り方(5)



- カバーリングインデックスを狙う場合、以下のインデックスを作成します。

- (A')

```
CREATE INDEX NEW_page_ix01 ON page (page_namespace, page_is_redirect);
CREATE INDEX NEW_revision_ix03 ON revision (rev_page, rev_user);
```

- (B')

```
CREATE INDEX NEW_revision_ix04 ON revision (rev_user, rev_page);
```

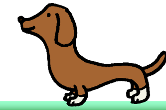
```
SELECT /* Q1 */ COUNT(*)
  FROM page p
  JOIN revision r
    ON p.page_id = r.rev_page
 WHERE p.page_is_redirect = 0
   AND p.page_namespace = 0
   AND r.rev_user = 0;
```

(A'-2) InnoDBの仕様により、page_idは
NEW_page_ix01に含まれています。

(A'-3) NEW_revision_ix03によって、rev_user
のためにrevisionテーブルにアクセスする
必要がなくなります。

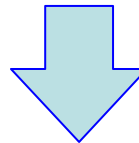
(B'-2) NEW_revision_ix04によって、rev_page
のためにrevisionテーブルにアクセスする
必要がなくなります。

(A')と(B')はどちらが効率的か



- (A')
  - NEW_page_ix01にアクセスして1,348,094レコード抽出します。
    (通常ページは全体の81%、リダイレクトページは今回のテストデータにはなし)
  - NEW_revision_ix03に1,348,094回アクセスし、1回あたり1レコード抽出します。

- (B')
  - NEW_revision_ix04にアクセスして299,781レコード抽出します。
    (匿名ユーザによる更新は全体の18%)
  - pageのPRIMARYに299,781回アクセスし、1回あたり1レコード抽出します。

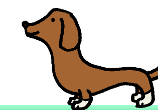


- (B')の方が効率的です。NEW_revision_ix04を採用しましょう。

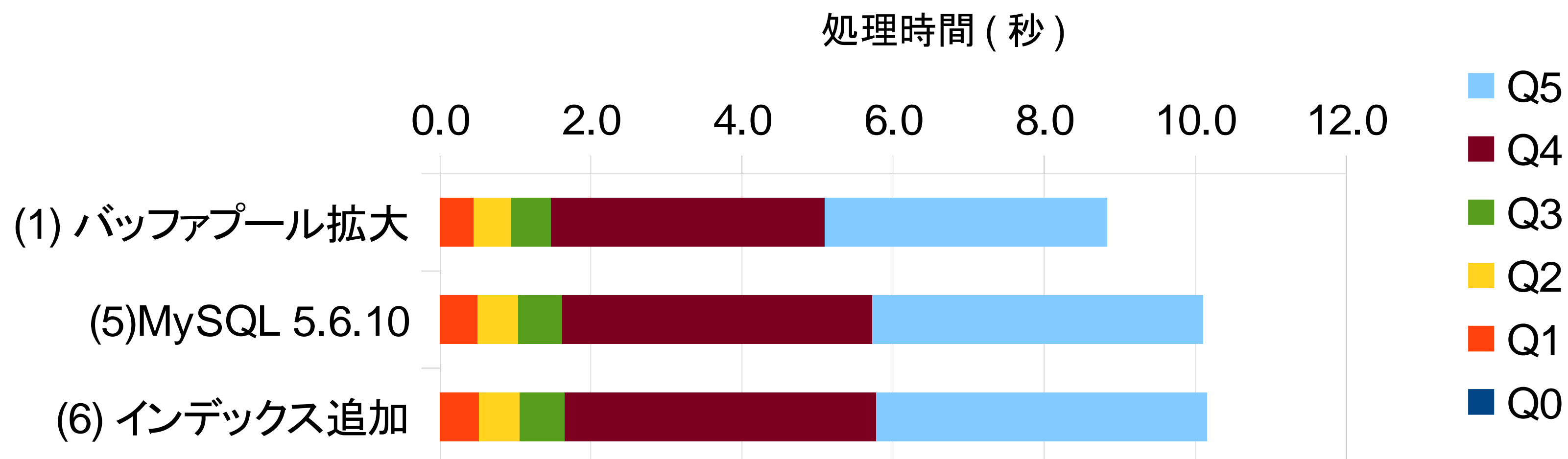
- なお、このインデックスはクエリ2番、3番、4番にも効果があります。

```
CREATE INDEX NEW_revision_ix04 ON revision (rev_user, rev_page);
```

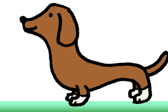
その6:インデックスを追加



- NEW_revision_ix04を作成しました。
- 何も変わりませんでした。



SQL実行計画



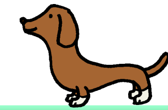
- EXPLAINを取ってみると、NEW_revision_ix04が使われていないようです。

- よくあることですが、もう少し調べてみましょう。

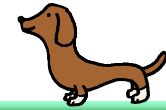
```
KEY `user_timestamp` (`rev_user`,`rev_timestamp`)
```

```
+----+-------------+-------+--------+---------------------------+----------------+---------+-------------------+--------+-------------+
| id | select_type | table | type   | possible_keys             | key            | key_len | ref               | rows   | Extra       |
+----+-------------+-------+--------+---------------------------+----------------+---------+-------------------+--------+-------------+
|  1 | SIMPLE      | r     | ref    | rev_page_id,              | user_timestamp | 4       | const             | 339218 | NULL        |
|    |             |       |        | page_timestamp,           |                |         |                   |        |             |
|    |             |       |        | user_timestamp,           |                |         |                   |        |             |
|    |             |       |        | page_user_timestamp,      |                |         |                   |        |             |
|    |             |       |        | NEW_revision_ix04         |                |         |                   |        |             |
|  1 | SIMPLE      | p     | eq_ref | PRIMARY,                  | PRIMARY        | 4       | wikipedia.r.rev_page |   1 | Using where |
|    |             |       |        | name_title,               |                |         |                   |        |             |
|    |             |       |        | page_redirect_namespace_len |              |         |                   |        |             |
+----+-------------+-------+--------+---------------------------+----------------+---------+-------------------+--------+-------------+
```

MySQL 5.6 Optimizer Trace



Optimizer Trace

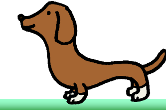


- SQLオプティマイザの内部動作を調べる、MySQL 5.6の新機能です。

- 基本的な使い方を以下に示します。

```
mysql> SET SESSION optimizer_trace = 1;
mysql> SET SESSION end_markers_in_json = 1;
mysql> SET SESSION optimizer_trace_max_mem_size = 131072;

(SQLを実行、EXPLAINでも可)

mysql> pager less
mysql> SELECT * FROM information_schema.optimizer_trace¥G
mysql> pager

mysql> SET SESSION optimizer_trace = 0;
```

設定パラメータ



- optimizer_trace
  0：enabled=off, one_line=off
  1：enabled=on, one_line=off
  3：enabled=on, one_line=on

- end_markers_in_json
  0：デフォルト設定です。閉じタグにコメントがつきません。
  1：「} /* range_analysis */」などと閉じタグにコメントがつきます。

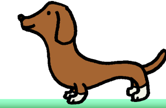
- optimizer_trace_max_mem_size
  トレースが保持される量をバイト単位で指定します。デフォルトは16,384バイトです。
  デフォルトだと今回のクエリ1番でも足りなくなるので、少し増やしておきましょう。

出力例

- クエリ1番で586行あります。


```
{
  "steps": [
    {
      "join_preparation": {
        "select#": 1,
        "steps": [
          {
            "expanded_query": "/* select#1 */ select count(0) AS `COUNT(*)` from (`page` `p` join `revision` `r` on((`p`.`page_id` =
`r`.`rev_page`))) where ((`p`.`page_is_redirect` = 0) and (`p`.`page_namespace` = 0) and (`r`.`rev_user` = 0))"
          }
        ]
      }
    },
    {
      "join_optimization": {
        "select#": 1,
        "steps": [
          {
            "transformations_to_nested_joins": {
              "transformations": [
                "JOIN_condition_to_WHERE",
                "parenthesis_removal"
              ],
              "expanded_query": "/* select#1 */ select count(0) AS `COUNT(*)` from `page` `p` join `revision` `r` where
((`p`.`page_is_redirect` = 0) and (`p`.`page_namespace` = 0) and (`r`.`rev_user` = 0) and (`p`.`page_id` = `r`.`rev_page`))"
            }
          },
          {
            "condition_processing": {
              "condition": "WHERE",
              "original_condition": "((`p`.`page_is_redirect` = 0) and (`p`.`page_namespace` = 0) and (`r`.`rev_user` = 0) and (`p`.`page_id` =
`r`.`rev_page`))",
              "steps": [
                {
                  "transformation": "equality_propagation",
                  "resulting_condition": "(multiple equal(0, `p`.`page_is_redirect`) and multiple equal(0, `p`.`page_namespace`) and multiple
equal(0, `r`.`rev_user`) and multiple equal(`p`.`page_id`, `r`.`rev_page`))"
                },
                {
                  "transformation": "constant_propagation",
                  "resulting_condition": "(multiple equal(0, `p`.`page_is_redirect`) and multiple equal(0, `p`.`page_namespace`) and multiple
equal(0, `r`.`rev_user`) and multiple equal(`p`.`page_id`, `r`.`rev_page`))"
                },
                {
                  "transformation": "trivial_condition_removal",
                  "resulting_condition": "(multiple equal(0, `p`.`page_is_redirect`) and multiple equal(0, `p`.`page_namespace`) and multiple
equal(0, `r`.`rev_user`) and multiple equal(`p`.`page_id`, `r`.`rev_page`))"
                }
              ]
            }
          },
          {
            "table_dependencies": [
              {
                "table": "`page` `p`",
                "row_may_be_null": false,
                "map_bit": 0,
                "depends_on_map_bits": [
                ]
              },
              {
                "table": "`revision` `r`",
                "row_may_be_null": false,
                "map_bit": 1,
                "depends_on_map_bits": [
                ]
              }
            ]
          },
          {
            "ref_optimizer_key_uses": [
              {
                "table": "`page` `p`",
                "field": "page_id",
                "equals": "`r`.`rev_page`",
                "null_rejecting": false
              },
              {
                "table": "`page` `p`",
                "field": "page_namespace",
                "equals": "0",
                "null_rejecting": false
              },
              {
                "table": "`page` `p`",
                "field": "page_is_redirect",
                "equals": "0",
                "null_rejecting": false
              },
              {
                "table": "`page` `p`",
                "field": "page_namespace",
                "equals": "0",
                "null_rejecting": false
              },
              {
                "table": "`revision` `r`",
                "field": "rev_page",
                "equals": "`p`.`page_id`",
                "null_rejecting": false
              },
              {
                "table": "`revision` `r`",
                "field": "rev_page",
                "equals": "`p`.`page_id`",
                "null_rejecting": false
              },
              {
                "table": "`revision` `r`",
                "field": "rev_user",
                "equals": "0",
                "null_rejecting": false
              },
              {
                "table": "`revision` `r`",
                "field": "rev_page",
                "equals": "`p`.`page_id`",
                "null_rejecting": false
              },
              {
                "table": "`revision` `r`",
                "field": "rev_user",
                "equals": "0",
                "null_rejecting": false
              },
              {
                "table": "`revision` `r`",
                "field": "rev_user",
                "equals": "0",
                "null_rejecting": false
              },
              {
                "table": "`revision` `r`",
                "field": "rev_page",
                "equals": "`p`.`page_id`",
                "null_rejecting": false
              }
            ]
          },
          {
            "rows_estimation": [
              {
                "table": "`page` `p`",
                "range_analysis": {
                  "table_scan": {
                    "rows": 1650004,
                    "cost": 340029
                  },
                  "potential_range_indices": [
                    {
                      "index": "PRIMARY",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "name_title",
                      "usable": true,
                      "key_parts": [
                        "page_namespace",
                        "page_title"
                      ]
                    },
                    {
                      "index": "page_random",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "page_len",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "page_redirect_namespace_len",
                      "usable": true,
                      "key_parts": [
                        "page_is_redirect",
                        "page_namespace",
                        "page_len",
                        "page_id"
                      ]
                    }
                  ],
                  "best_covering_index_scan": {
                    "index": "page_redirect_namespace_len",
                    "cost": 332917,
                    "chosen": true
                  },
                  "setup_range_conditions": [
                  ],
                  "group_index_range": {
                    "chosen": false,
                    "cause": "not_single_table"
                  },
                  "analyzing_range_alternatives": {
                    "range_scan_alternatives": [
                      {
                        "index": "name_title",
                        "ranges": [
                          "0 <= page_namespace <= 0"
                        ],
                        "index_dives_for_eq_ranges": true,
                        "rowid_ordered": false,
                        "using_mrr": false,
                        "index_only": false,
                        "rows": 825002,
                        "cost": 990003,
                        "chosen": false,
                        "cause": "cost"
                      },
                      {
                        "index": "page_redirect_namespace_len",
                        "ranges": [
                          "0 <= page_is_redirect <= 0 AND 0 <= page_namespace <= 0"
                        ],
                        "index_dives_for_eq_ranges": true,
                        "rowid_ordered": false,
                        "using_mrr": false,
                        "index_only": true,
                        "rows": 825002,
                        "cost": 166309,
                        "chosen": true
                      }
                    ],
                    "analyzing_roworder_intersect": {
                      "usable": false,
                      "cause": "too_few_roworder_scans"
                    }
                  },
                  "chosen_range_access_summary": {
                    "range_access_plan": {
                      "type": "range_scan",
                      "index": "page_redirect_namespace_len",
                      "rows": 825002,
                      "ranges": [
                        "0 <= page_is_redirect <= 0 AND 0 <= page_namespace <= 0"
                      ]
                    },
                    "rows_for_plan": 825002,
                    "cost_for_plan": 166309,
                    "chosen": true
                  }
                }
              },
              {
                "table": "`revision` `r`",
                "range_analysis": {
                  "table_scan": {
                    "rows": 1629590,
                    "cost": 338894
                  },
                  "potential_range_indices": [
                    {
                      "index": "PRIMARY",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "rev_page_id",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "rev_timestamp",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "page_timestamp",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "user_timestamp",
                      "usable": true,
                      "key_parts": [
                        "rev_user",
                        "rev_timestamp",
                        "rev_id"
                      ]
                    },
                    {
                      "index": "usertext_timestamp",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "page_user_timestamp",
                      "usable": false,
                      "cause": "not_applicable"
                    },
                    {
                      "index": "NEW_revision_ix04",
                      "usable": true,
                      "key_parts": [
                        "rev_user",
                        "rev_page",
                        "rev_id"
                      ]
                    }
                  ],
                  "best_covering_index_scan": {
                    "index": "NEW_revision_ix04",
                    "cost": 328305,
                    "chosen": true
                  },
                  "setup_range_conditions": [
                  ],
                  "group_index_range": {
                    "chosen": false,
                    "cause": "not_single_table"
                  },
                  "analyzing_range_alternatives": {
                    "range_scan_alternatives": [
                      {
                        "index": "user_timestamp",
                        "ranges": [
                          "0 <= rev_user <= 0"
                        ],
                        "index_dives_for_eq_ranges": true,
                        "rowid_ordered": false,
                        "using_mrr": false,
                        "index_only": false,
                        "rows": 339218,
                        "cost": 407063,
                        "chosen": false,
                        "cause": "cost"
                      },
                      {
                        "index": "NEW_revision_ix04",
                        "ranges": [
                          "0 <= rev_user <= 0"
                        ],
                        "index_dives_for_eq_ranges": true,
                        "rowid_ordered": false,
                        "using_mrr": false,
                        "index_only": true,
                        "rows": 601812,
                        "cost": 121245,
                        "chosen": true
                      }
                    ],
                    "analyzing_roworder_intersect": {
                      "usable": false,
                      "cause": "too_few_roworder_scans"
                    }
                  },
                  "chosen_range_access_summary": {
                    "range_access_plan": {
                      "type": "range_scan",
                      "index": "NEW_revision_ix04",
                      "rows": 601812,
                      "ranges": [
                        "0 <= rev_user <= 0"
                      ]
                    },
                    "rows_for_plan": 601812,
                    "cost_for_plan": 121245,
                    "chosen": true
                  }
                }
              }
            ]
          },
          {
            "considered_execution_plans": [
              {
                "plan_prefix": [
                ],
                "table": "`revision` `r`",
                "best_access_path": {
                  "considered_access_paths": [
                    {
                      "access_type": "ref",
                      "index": "rev_page_id",
                      "usable": false,
                      "chosen": false
                    },
                    {
                      "access_type": "ref",
                      "index": "page_timestamp",
                      "usable": false,
                      "chosen": false
                    },
                    {
                      "access_type": "ref",
                      "index": "user_timestamp",
                      "rows": 339218,
                      "cost": 106766,
                      "chosen": true
                    },
                    {
                      "access_type": "ref",
                      "index": "page_user_timestamp",
                      "rows": 339218,
                      "cost": 174609,
                      "chosen": false
                    },
                    {
                      "access_type": "ref",
                      "index": "NEW_revision_ix04",
                      "rows": 601812,
                      "cost": 121245,
                      "chosen": false
                    },
                    {
                      "access_type": "range",
                      "cost": 241606,
                      "rows": 601812,
                      "cause": "cost",
                      "chosen": false
                    }
                  ]
                },
                "cost_for_plan": 106766,
                "rows_for_plan": 339218,
                "rest_of_plan": [
                  {
                    "plan_prefix": [
                      "`revision` `r`"
                    ],
                    "table": "`page` `p`",
                    "best_access_path": {
                      "considered_access_paths": [
                        {
                          "access_type": "ref",
                          "index": "PRIMARY",
                          "rows": 1,
                          "cost": 339218,
                          "chosen": true
                        },
                        {
                          "access_type": "ref",
                          "index": "name_title",
                          "rows": 825002,
                          "cost": 1e10,
                          "chosen": false
                        },
                        {
                          "access_type": "ref",
                          "index": "page_redirect_namespace_len",
                          "rows": 825002,
                          "cost": 4.44e8,
                          "chosen": false
                        },
                        {
                          "access_type": "range",
                          "rows": 618752,
                          "cost": 1.1e11,
                          "chosen": false
                        }
                      ]
                    },
                    "cost_for_plan": 513827,
                    "rows_for_plan": 339218,
                    "chosen": true
                  }
                ]
              },
              {
                "plan_prefix": [
                ],
                "table": "`page` `p`",
                "best_access_path": {
                  "considered_access_paths": [
                    {
                      "access_type": "ref",
                      "index": "PRIMARY",
                      "usable": false,
                      "chosen": false
                    },
                    {
                      "access_type": "ref",
                      "index": "name_title",
                      "rows": 825002,
                      "cost": 195078,
                      "chosen": true
                    },
                    {
                      "access_type": "ref",
                      "index": "page_redirect_namespace_len",
                      "rows": 825002,
                      "cost": 166309,
                      "chosen": true
                    },
                    {
                      "access_type": "range",
                      "cause": "heuristic_index_cheaper",
                      "chosen": false
                    }
                  ]
                },
                "cost_for_plan": 166309,
                "rows_for_plan": 825002,
                "rest_of_plan": [
                  {
                    "plan_prefix": [
                      "`page` `p`"
                    ],
                    "table": "`revision` `r`",
                    "best_access_path": {
                      "considered_access_paths": [
                        {
                          "access_type": "ref",
                          "index": "rev_page_id",
                          "rows": 1,
                          "cost": 825002,
                          "chosen": true
                        },
                        {
                          "access_type": "ref",
                          "index": "page_timestamp",
                          "rows": 1,
                          "cost": 825002,
                          "chosen": false
                        },
                        {
                          "access_type": "ref",
                          "index": "user_timestamp",
                          "rows": 339218,
                          "cost": 3.2e10,
                          "chosen": false
                        },
                        {
                          "access_type": "ref",
                          "index": "page_user_timestamp",
                          "rows": 1,
                          "cost": 825002,
                          "chosen": false
                        },
                        {
                          "access_type": "ref",
                          "index": "NEW_revision_ix04",
                          "rows": 1,
                          "cost": 825002,
                          "chosen": false
                        },
                        {
                          "access_type": "range",
                          "rows": 339218,
                          "cost": 2e11,
                          "chosen": false
                        }
                      ]
                    },
                    "cost_for_plan": 1.16e6,
                    "rows_for_plan": 825002,
                    "pruned_by_cost": true
                  }
                ]
              }
            ]
          },
          {
            "attaching_conditions_to_tables": {
              "original_condition": "((`p`.`page_id` = `r`.`rev_page`) and (`r`.`rev_user` = 0) and (`p`.`page_namespace` = 0) and
(`p`.`page_is_redirect` = 0))",
              "attached_conditions_computation": [
              ],
              "attached_conditions_summary": [
                {
                  "table": "`revision` `r`",
                  "attached": null
                },
                {
                  "table": "`page` `p`",
                  "attached": "((`p`.`page_namespace` = 0) and (`p`.`page_is_redirect` = 0))"
                }
              ]
            }
          },
          {
            "refine_plan": [
              {
                "table": "`revision` `r`"
              },
              {
                "table": "`page` `p`"
              }
            ]
          }
        ]
      }
    },
    {
      "join_execution": {
        "select#": 1,
        "steps": [
        ]
      }
    }
  ]
}
```

rows_estimation(1)



- ポイントを絞って確認していきましょう。rows_estimationセクションで、アクセスパスごとの見積もりレコード数を確認することができます。

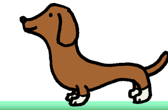
```
"rows_estimation": [
  {
    "table": "`revision` `r`",
    "range_analysis": {
      "table_scan": {
        "rows": 1629590,
        "cost": 338894
      },
      …
      "best_covering_index_scan": {
        "index": "NEW_revision_ix04",
        "cost": 328305,
        "chosen": true
      },
      …
```

テーブルフルスキャンのときの見積もりレコード数です。SHOW TABLE STATUSで取得できるものと同じです。

分かりづらいですが、ここはインデックスレンジスキャンではなくインデックスフルスキャンのときの見積もりコストです。

※インデックスフルスキャンについては、以下のエントリを参照してください。
http://d.hatena.ne.jp/sh2/20111217

rows_estimation(2)



- rows_estimation内のanalyzing_range_alternatives以降で、InnoDBのレンジ分析を用いたレコード数の見積もりが行われます。

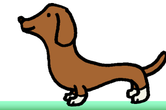
```
"analyzing_range_alternatives": {
  "range_scan_alternatives": [
    {
      "index": "user_timestamp",
      "ranges": [
        "0 <= rev_user <= 0"
      ],
      "index_dives_for_eq_ranges": true,
      "rowid_ordered": false,
      "using_mrr": false,
      "index_only": false,
      "rows": 339218,
      "cost": 407063,
      "chosen": false,
      "cause": "cost"
    },
```

```
    {
      "index": "NEW_revision_ix04",
      "ranges": [
        "0 <= rev_user <= 0"
      ],
      "index_dives_for_eq_ranges": true,
      "rowid_ordered": false,
      "using_mrr": false,
      "index_only": true,
      "rows": 601812,
      "cost": 121245,
      "chosen": true
    }
  ],
```

```
KEY `user_timestamp` (`rev_user`,`rev_timestamp`)
```

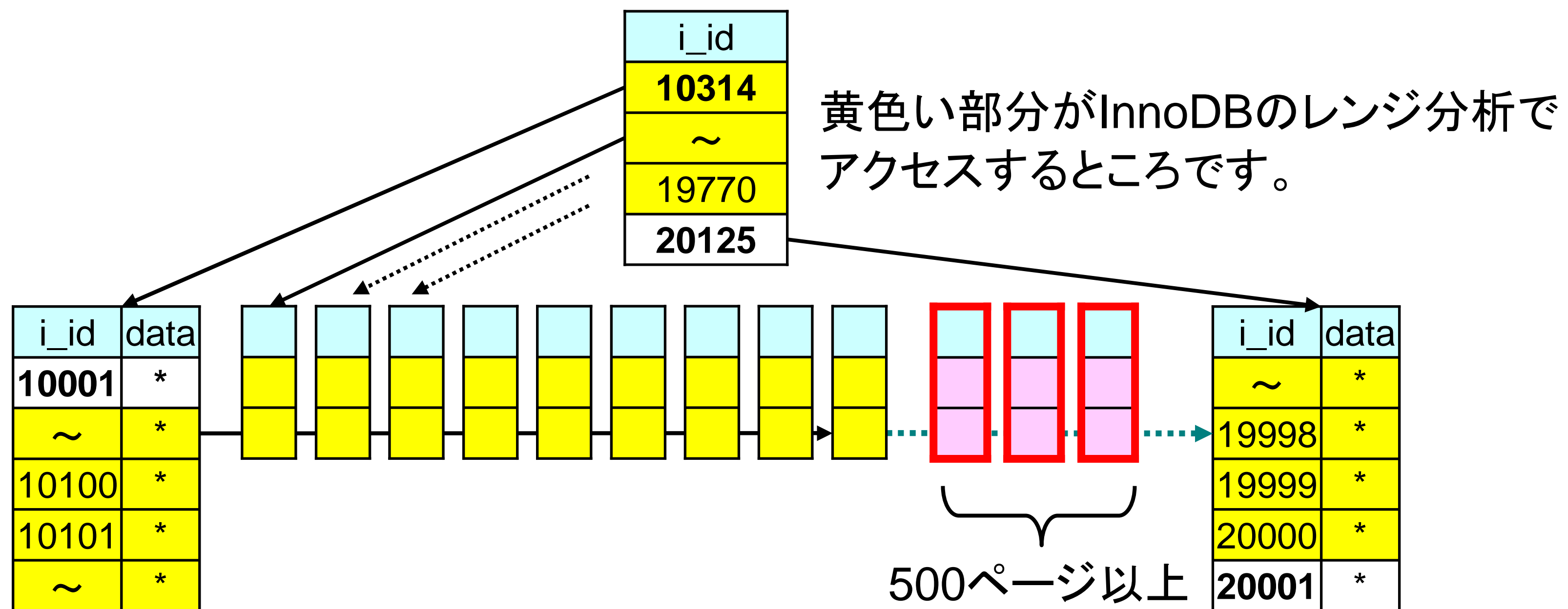
```
KEY `NEW_revision_ix04` (`rev_user`,`rev_page`)
```

レコード数の見積もり精度が低い



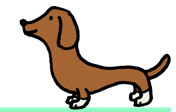
- user_timestampとNEW_revision_ix04は絞り込みに有効なカラムがrev_userで同じですから、実際に抽出されるレコード数も同じです。にもかかわらず見積もりレコード数に大きな差があるのは、InnoDBのレンジ分析の精度が低いためです。

- 今回のテストデータでは、500以上のInnoDBページに格納されているレコード数を、11のリーフページにアクセスするだけで見積もらなければなりません。

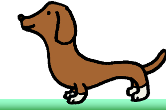


※InnoDBのレンジ分析については、以下のエントリを参照してください。
http://d.hatena.ne.jp/sh2/20120310

チューニングその7：オプティマイザの精度向上



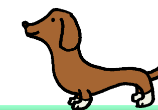
サンプル数の増加



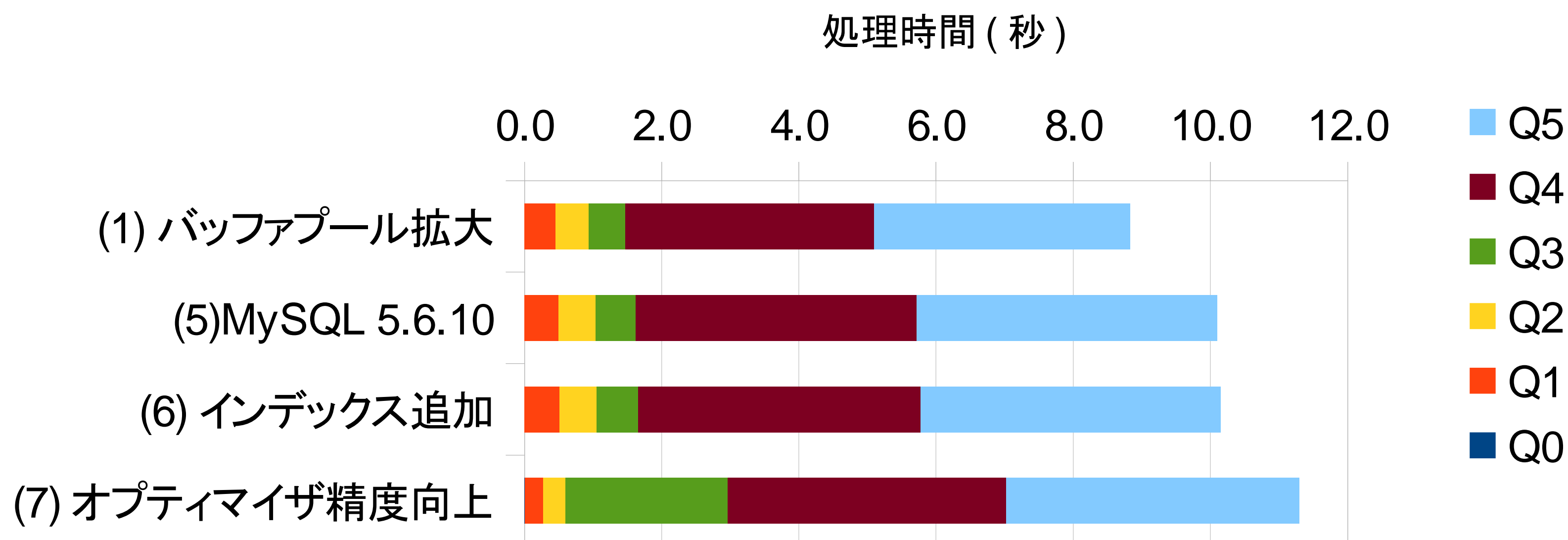
- InnoDBのレンジ分析でアクセスするページ数を10倍に増やしてみました。
- パラメータで変更できるようにはなっていませんので、MySQL本体を改造します。

```
storage/innobase/btr/btr0cur.cc

static ib_int64_t btr_estimate_n_rows_in_range_on_level(…
{
…
/* Do not read more than this number of pages in order not to hurt
performance with this code which is just an estimation. If we read
this many pages before reaching slot2->page_no then we estimate the
average from the pages scanned so far */

#define N_PAGES_READ_LIMIT 100
```

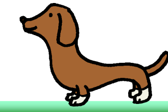
その7:オプティマイザの精度向上



- クエリ1番、2番は無事NEW_revision_ix04が使われるようになり、速くなりました。
- しかし、クエリ3番が遅くなってしまいました。



SQL実行計画



- クエリ3番のEXPLAINを取ってみると、テーブルの結合順序が変わっていました。

- BEFORE

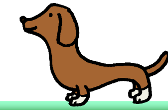
id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	r	ref	rev_page_id, page_timestamp, user_timestamp, page_user_timestamp, NEW_revision_ix04	user_timestamp	4	const	339218	Using temporary; Using filesort
1	SIMPLE	p	eq_ref	PRIMARY, name_title, page_redirect_namespace_len	PRIMARY	4	wikipedia.r.rev_page	1	Using where

- AFTER

813,709

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	p	ref	PRIMARY, name_title, page_redirect_namespace_len	name_title	4	const	813709	Using where; Using filesort
1	SIMPLE	r	ref	rev_page_id, page_timestamp, user_timestamp, page_user_timestamp, NEW_revision_ix04	rev_page_id	4	wikipedia.p.page_id	1	Using where

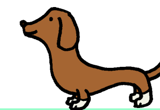
見積もりレコード数が少ない



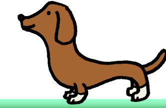
- pageテーブルの見積もりレコード数が813,709となっていますが、これは実際の値よりもかなり少ないです。テーブル全体のレコード数は1,668,431、抽出条件を満たすレコードは1,348,094あります。ソースコードの以下の部分がポイントです。

```
storage/innobase/btr/btr0cur.cc

UNIV_INTERN ib_int64_t btr_estimate_n_rows_in_range(…
{
…
    if (i > divergence_level + 1 && !is_n_rows_exact) {
        /* In trees whose height is > 1 our algorithm
        tends to underestimate: multiply the estimate
        by 2: */
        n_rows = n_rows * 2;
    }

    /* Do not estimate the number of rows in the range
    to over 1 / 2 of the estimated rows in the whole
    table */
    if (n_rows > table_n_rows / 2 && !is_n_rows_exact) {
        n_rows = table_n_rows / 2;
        …
    }
```

チューニングその8:ヒューリスティクスの排除



実態にそぐわないヒューリスティクス



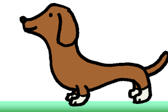
- 先ほどの処理を言葉で説明すると、以下のようになります。
  - 11ページ(改造版では101ページ)アクセスしても正確なレコード数が分からなかった場合は、それまでに推定した見積もりレコード数を2倍にします。
  - 見積もりレコード数がテーブル全体のレコード数の半分を超えていた場合は、テーブル全体のレコード数の半分に制限します。

- このアルゴリズムによって、テーブル全体の25%以上のレコードを抽出するアクセスパスはすべて同じコストになってしまいます。pageテーブルからは81%のレコードを抽出するのですから、もっと高いコストになるべきです。

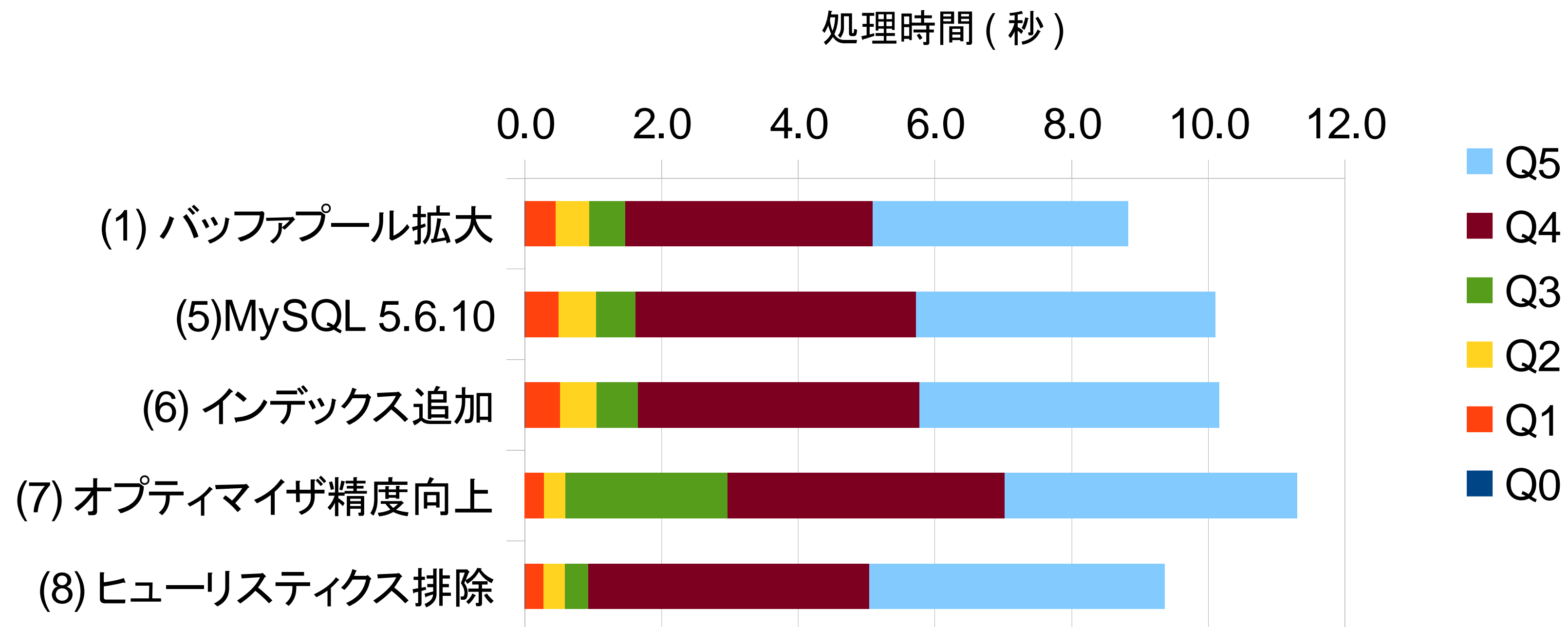
- 外してみましょう。

```
if (i > divergence_level + 1 && !is_n_rows_exact) {
    // n_rows = n_rows * 2;
}

if (n_rows > table_n_rows && !is_n_rows_exact) {
    n_rows = table_n_rows;
    …
}
```

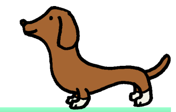
その8:ヒューリスティクスの排除



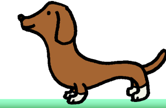
- クエリ3番が直りました。



チューニングその9：オプティマイザ・ヒントの付与



STRAIGHT_JOINとFORCE INDEX



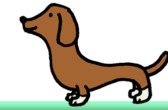
- クエリ4番はrevisionテーブルを駆動表としてNEW_revision_ix04のインデックスフルスキャンをすると速いのですが、Optimizer Traceを見る限りMySQLはどうしてもこのSQL実行計画を選ぶことができないようです。

- そこで、オプティマイザ・ヒントを付与してこのSQL実行計画を強制します。なおチューニンガソン5ではレギュレーションでクエリ書き換えが禁止されていましたので、ここから先は業務での応用を見越したチューニング案となります。

```
 SELECT /* Q4 */ STRAIGHT_JOIN r.rev_user, COUNT(*) AS c
    FROM revision r FORCE INDEX (NEW_revision_ix04)
    JOIN page p
      ON p.page_id = r.rev_page
   WHERE p.page_is_redirect = 0
     AND p.page_namespace = 0
GROUP BY r.rev_user
ORDER BY c DESC;
```

```
KEY `NEW_revision_ix04` (`rev_user`,`rev_page`)
```

Loose Index Scan

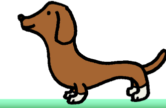


- クエリ4番で行っているチューニングは、GROUP BY句で集約されるカラムをインデックスの先頭に配置するというものです。これによって集約の際のソート処理をなくすことができます。Loose Index Scanと呼ばれているテクニックです。
  http://dev.mysql.com/doc/refman/5.6/en/loose-index-scan.html

- Optimizer Traceのreconsidering_access_paths_for_index_orderingセクションでLoose Index Scanが効いていることを確認できます。これはEXPLAINでは確認できません。

```
"reconsidering_access_paths_for_index_ordering": {
  "clause": "GROUP BY",
  "index_order_summary": {
    "table": "`revision` `r` FORCE INDEX (`NEW_revision_ix04`)",
    "index_provides_order": true,
    "order_direction": "asc",
    "index": "NEW_revision_ix04",
    "plan_changed": true,
    "access_type": "index_scan"
  } /* index_order_summary */
} /* reconsidering_access_paths_for_index_ordering */
```

sort_buffer_sizeの拡大



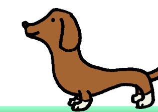
- クエリ4番は結果セットが39,434レコードと大きく、デフォルト設定ではORDER BY句の処理でディスクソートが実行されます。

- クエリの前後でSort_merge_passesの値を調べ、値の増加がなくなるところまでsort_buffer_sizeを増やします。

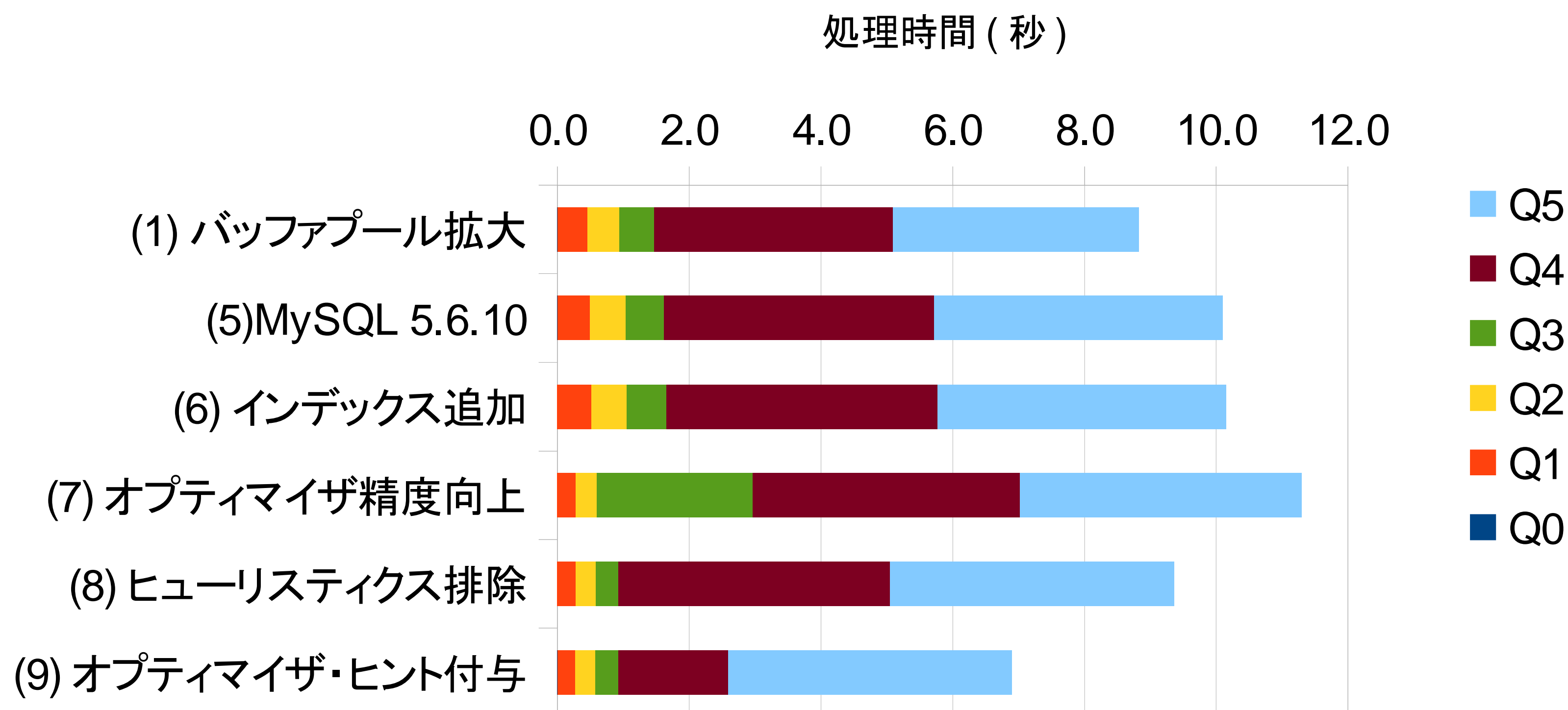
```
mysql> SHOW SESSION STATUS LIKE 'Sort_merge_passes';

mysql> pager head
mysql> SELECT /* Q4 */ STRAIGHT_JOIN r.rev_user, COUNT(*) AS c …
mysql> pager

mysql> SHOW SESSION STATUS LIKE 'Sort_merge_passes';
mysql> SET SESSION sort_buffer_size = …
```

- MySQL 5.5からMySQL 5.6にかけてsort_buffer_sizeのデフォルト値が2MBから256KBに削減されているので、注意が必要です。

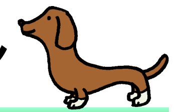
その9：オプティマイザ・ヒントの付与



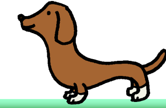
- オプティマイザ・ヒントを付与し、sort_buffer_sizeを1MBに増やしました。
- クエリ4番が2.5倍ほど速くなりました。



チューニングその10：サブクエリのマテリアライゼーション



中間テーブルの作成

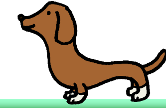


- クエリ5番はGROUP BY句にファンクションが使われているため、クエリ4番よりも厳しい状況です。

- このようなクエリに対して、中間テーブルを作成して高速化を図るというテクニックがあります。

```
 SELECT /* Q5 */ SUBSTRING(r.rev_timestamp, 1, 6), COUNT(*) AS c
    FROM page p
    JOIN revision r
      ON p.page_id = r.rev_page
   WHERE p.page_is_redirect = 0
     AND p.page_namespace = 0
GROUP BY SUBSTRING(r.rev_timestamp, 1, 6)
ORDER BY c DESC;
```

サブクエリのマテリアライゼーション



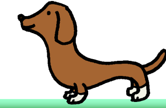
- revisionテーブルをFROM句のサブクエリ内に囲い込んで、同時にSUBSTRINGファンクションを解決しておきます。MySQLには、FROM句のサブクエリに対して中間テーブルを作成するという性質があります。これをマテリアライゼーションと呼びます。

- EXPLAINを取るとDERIVEDというselect_typeが確認できます。処理レコード数が少ない場合はマテリアライゼーションのオーバーヘッドによってむしろ遅くなりますが、処理レコード数が多い場合には試してみる価値があります。

- バッドノウハウなので、あまりおすすめはしません。

```
 SELECT /* Q5 */ r.rev_timestamp_ym, COUNT(*) AS c
    FROM page p JOIN
         (SELECT rev_page, SUBSTRING(rev_timestamp, 1, 6) AS rev_timestamp_ym
            FROM revision) r
      ON p.page_id = r.rev_page
   WHERE p.page_is_redirect = 0
     AND p.page_namespace = 0
GROUP BY r.rev_timestamp_ym
ORDER BY c DESC;
```

tmp_table_sizeとmax_heap_table_sizeの拡大

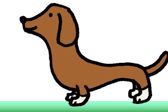


- マテリアライゼーションにはMEMORYストレージエンジンが用いられます。ただし、tmp_table_sizeを超える場合は途中でMyISAMストレージエンジンに切り替わります。もちろんMEMORYストレージエンジンで処理が完結する方が性能がよいです。

- クエリの前後でCreated_tmp_disk_tablesの値を調べ、値の増加がなくなるところまでtmp_table_sizeを増やします。またtmp_table_sizeはmax_heap_table_sizeよりも大きくできないので、max_heap_table_sizeも合わせて増やします。

```
mysql> SHOW SESSION STATUS LIKE 'Created_tmp_disk_tables';

mysql> pager head
mysql> SELECT /* Q5 */ r.rev_timestamp_ym, COUNT(*) AS c …
mysql> pager

mysql> SHOW SESSION STATUS LIKE 'Created_tmp_disk_tables';
mysql> SET SESSION max_heap_table_size = …
mysql> SET SESSION tmp_table_size = …
```

その10:サブクエリのマテリアライゼーション



- クエリを書き換えて、tmp_table_sizeを32MBに増やしました。
- クエリ5番が1.7倍ほど速くなりました。

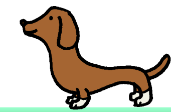
処理時間 (秒)

0.0 2.0 4.0 6.0 8.0 10.0 12.0

(1) バッファプール拡大
(5)MySQL 5.6.10
(6) インデックス追加
(7) オプティマイザ精度向上
(8) ヒューリスティクス排除
(9) オプティマイザ・ヒント付与
(10) マテリアライゼーション

Q5
Q4
Q3
Q2
Q1
Q0

チューニングその11：テーブルの非正規化



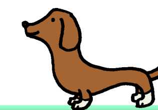
テーブルの非正規化

- チューニングの前提条件をどこまで覆してよいのか悩むところですが、今回のように大量のレコードを処理する分析系のクエリに対しては、テーブルの非正規化がよく効きます。

- page_id、rev_idを主キーとしたテーブルを作成します。あらかじめrev_timestampから年月を取り出したカラムを追加しておくと、クエリ5番で役に立ちます。

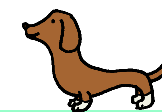
```
CREATE TABLE page_revision (
  `page_id` int(10) unsigned NOT NULL,
  `rev_id` int(10) unsigned NOT NULL,
  …
  `rev_timestamp` binary(14) NOT NULL DEFAULT '¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0',
  `rev_timestamp_ym` binary(14) NOT NULL DEFAULT '¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0¥0',
  PRIMARY KEY (`page_id`, `rev_id`),
  KEY `page_revision_ix01` (`rev_user`,`page_namespace`,`page_is_redirect`,`page_touched`),
  KEY `page_revision_ix02` (`rev_timestamp_ym`,`page_namespace`,`page_is_redirect`)
) DEFAULT CHARSET=utf8;

INSERT INTO page_revision (page_id, rev_id, …, rev_timestamp_ym)
SELECT p.page_id, r.rev_id, …, SUBSTRING(r.rev_timestamp, 1, 6)
  FROM page p
  JOIN revision r
    ON p.page_id = r.rev_page;
```

その11：テーブルの非正規化

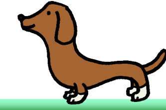


- 全体の処理が3.8倍ほど速くなりました。

まとめ

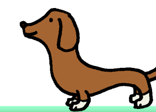


まとめ



- チューニングは、どこまで前提条件を覆せるかの勝負です。「この処理、なくてもいいよね」というちゃぶ台返しを常に狙っていきたいところです。
- さまざまな制約により普通のチューニングをすることになった場合は、MySQL 5.6のOptimizer Traceが役に立つと思います。ただ、今のところ中身を理解するにはMySQLのソースコードと照らし合わせて読み解く必要があります。
- 残念ながらMySQLのSQLオプティマイザはあまり賢くないので、オプティマイザ・ヒントを付与しなければならないこともありますし、クエリの書き換えを強いられることもあります。O/Rマッパーを利用している環境であっても、オプティマイザ・ヒントを挿し込んだりクエリを書き換えたりする手段は確保しておきたいところです。
- チューニンガソン5のテストデータは、SQLオプティマイザの弱点をあぶり出すとてもよいサンプルでした。もう少し詳しく調べてからbugs.mysql.comに改善要望を出していこうと思います。

宿題



1. もっと速くしてください。