MySQLにおける実行計画(EXPLAIN)の見方

Ryusuke Kajiyama / 梶山隆輔
MySQL Global Business Unit
MySQL Sales Consulting Senior Manager, Asia Pacific & Japan

Safe Harbor Statement

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。
また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメントするものではない為、購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

実行計画（Query Execution Plan）とは？

- SQL処理する時の内部的処理手順
  - SQLは内部的な処理手順を定めていないため、内部的な処理手順はRDBMSが決めている
  - 内部的な処理手順
    - インデックススキャン、テーブルスキャン、JOIN順番、サブクエリの処理方法、など
- 同じSQLであっても、実行計画が違えばパフォーマンスが大きく変わることがある
- 実行計画はオプティマイザが作成する
- 実行計画はEXPLAINで確認可能

オプティマイザ（Optimizer）とは？

- SQLの実行計画を作成する役割を持つ
- MySQLでは、コストベースのオプティマイザを採用しているため、コストに基づいて実行計画を作成する
  - コストに基づいて、最適な（最もコストが低い）実行計画を作成する（Optimize：最適化する）
- オプティマイザの判断が必ずしも最適だとは限らない
- オプティマイザがより良い実行計画を作成できるように、SQLチューニングを行う

実行計画の出力

Explain <実行計画を見たいSQL文>;

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

EXPLAINの各項目

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

id	クエリのID（テーブルのIDではないので注意）
select_type	クエリの種類
table	対象のテーブル
partitions	対象のパーティション（パーティションテーブルでない場合はNULLが出力される）
type	**レコードアクセスタイプ（どのようにテーブルにアクセスされるかを示す）**
possible_keys	利用可能なインデックス
key	選択されたインデックス
key_len	選択されたインデックスの長さ
ref	インデックスと比較される列
rows	行数の概算見積もり
filtered	条件によってフィルタリングされる行の割合
Extra	追加情報

id

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key      | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- SELECT 識別子
  - クエリ内の SELECT の連番
  - 同じ番号は、一回の処理に含まれているという意味
  - 必ずしも番号順に処理されるわけではない
    - SUBQUERYが含まれている場合は、そちらが先に処理される

select type

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- **SELECT の種類**で、次の表のいずれかになる

SIMPLE	単純な SELECT （UNION やサブクエリを使用しない）
PRIMARY	JOINで最も外側の SELECT
UNION	UNION 内の 2 つめ以降の SELECT ステートメント
DEPENDENT UNION	UNION 内の 2 つめ以降の SELECT ステートメントで、外側のクエリに依存
UNION RESULT	UNION の結果
SUBQUERY	サブクエリ内の最初の SELECT
DEPENDENT SUBQUERY	サブクエリ内の最初の SELECT で、外側のクエリに依存
DERIVED	派生テーブル SELECT （FROM 句内のサブクエリ）
MATERIALIZED	実体化されたサブクエリ
UNCACHEABLE SUBQUERY	結果をキャッシュできず、外側のクエリの行ごとに再評価される必要があるサブクエリ
UNCACHEABLE UNION	キャッシュ不可能なサブクエリ （UNCACHEABLE SUBQUERY を参照してください） に属する UNION 内の 2 つめ以降の SELECT

table

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- Table
  - SELECT 識別子出力の行で参照しているテーブルの名前

partitions

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

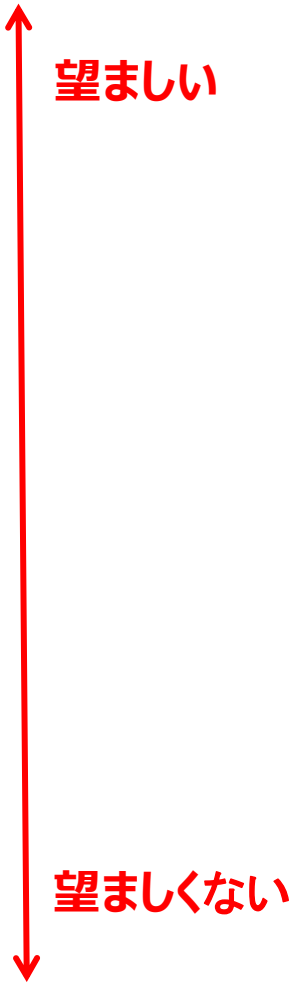
- Partitions
  - クエリでレコードが照合されるパーティション
    - このカラムは、PARTITIONS キーワードが使用されている場合にのみ表示
    - この値が**NULLの場合**、テーブルはパーティション化されていない

type

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- テーブルの**結合方法**
  - 次ページにて適切な結合型から不適切な結合型の順で記載

typeの値	意味
system	1行しかないテーブル(systemテーブル) ※constの特殊な例
const	PRIMARY/UNIQUEインデックスによる等価検索（一意検索）
eq_ref	PRIMARY/UNIQUEインデックスによるJOIN
ref	ユニークでないインデクスによる等価検索、JOIN
fulltext	全文検索インデックスを使用した全文検索
ref_or_null	ユニークでないインデックスによる等価検索とIS NULLのOR
index_merge	複数のインデックスをマージ
unique_subquery	サブクエリ内で、PRIMARY/UNIQUEインデックスで等価検索（一意検索）
index_subquery	サブクエリ内で、ユニークでないインデクスによる等価検索、
range	範囲検索
index	インデックスのフルスキャン
ALL	フルテーブルスキャン



Type:system

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- テーブルに行が1行ある場合のみ選択される
  ※これは、const 結合型の特殊なケース

Type:const

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- PRIMARY KEYまたはUNIQUE KEYによる等価比較(イコールでの比較)が行われた時のタイプ
- 結果は必ず1行になる
- オプティマイザは検索結果を「const(定数)」と見なすことからつけられた

Type:eq_ref

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- JOINにおいて、PRIMARY KEYまたはUNIQUE KEYが利用される場合のタイプ
- constと似ているが、内部表のアクセスに使われるという点が異なる
- **JOINの場合の最適な結合型**

Type:ref

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- ユニークでないインデックス(PRIMARY KEYまたはUNIQUE以外）を使って等価評価(イコールを使ったもの）が行われる場合のレコードアクセスタイプ
- JOINの場合とSIMPLEの場合両方に出現

Type:fulltext

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- 結合は FULLTEXT インデックスを使用して実行

Type:ref_or_null

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- ref と似ているが、NULL 値を含む行の追加検索を実行
- 次の例では、ref_or_null 結合を使用して、ref_table をSELECT

```
SELECT * FROM ref_table
  WHERE key_column=expr OR key_column IS NULL;
```

Type:index_merge

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- 2種類のインデックスをバラバラに使用して、フェッチした各行をマージする処理が行われる場合のタイプ

Type:unique_subquery

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- DEPENDENT SUBQUERY(最も遅いSELECTタイプ)で、PRIMARY KEYもしくはUNIQUEキーによって評価が行われるタイプ
- インデックスでサブクエリが完結するため、サブクエリの中では高速

Type:index_subquery

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- unique_subquery と似ている
- PRIMARY KEY/UNIQUEキーでないインデックスを使って評価する

Type:range

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- インデックスを使って、特定の範囲のレコードを取得する処理タイプ。
- BETWEENや不等号(>, <=, <, >=)、INを使うと出現

Type:index

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- フルインデックススキャン
- インデックスを使っていて効率よさそうな名前だが、実際は重い処理
- アンチパターン
- refまたはeq_refになるようにテーブル構造か、SQLを変更すべき
- ただし、ORDERBY + LIMITが入っている場合は出てきても問題ない

Type:ALL

typeの値
system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

- データの多いテーブルで実行されると悪
- テーブルフルスキャン（インデックスを一切用いない処理）
- JOINの中で出てきたら完全にアンチパターン(最悪)
- 全行を処理するような処理以外はさけるべき
- WorkbenchのVisual Explainでも赤四角で表示される

possible_keys

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- テーブルのアクセスに利用可能なインデックスの候補として挙げたキー
- ヒント句の「FORCE INDEX」を使っても、ここに挙がっていないキーは無視
- テーブルにあるインデックスを確認するには、

SHOW INDEX FROM tbl_name を使用

  - 5.5ではSHOW INDEX文により自動的に統計情報再収集が行われるので注意。

key

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- 実際に使用することを決定したキー （インデックス） を示す
- key は possible_keys 値に存在しないインデックスを指定している可能性あり
  - これは possible_keys インデックスのどれも行のルックアップに適していない場合に発生することがある

key_len

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- 使用することを決定した**キーの長さ**
- key カラムがNULL の場合、この長さも NULL

ref

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- 検索条件でkeyと比較されるカラムまたは定数

rows

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- クエリを実行するためにフェッチされる行数の推測値
  - 実際の値はやってみないとわからない
  - ただしサブクエリ部分のrowsは必ず正確な値
    （実際にExplain時にサブクエリが行われるため）

filtered

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- フェッチされる行数の中で実際に検索に使われそうな行数
  - 100%と出る事が多い
  - あまり参考にならない

Extra

```
mysql> explain select * from nodes where id = 31236601;
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | nodes | NULL       | const | i_nodeids     | i_nodeids | 9       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

- クエリを解決する方法に関する追加情報

Using filesort
Using index
Using index condition
Using index for group-by
Using join buffer (Block Nested Loop) Using join buffer (Batched Key Access)
Using MRR
Using temporary
Using where

Extra:Using filesort

Using filesort
Using index
Using index condition
Using index for group-by
Using join buffer (Block Nested Loop) Using join buffer (Batched Key Access)
Using MRR
Using temporary
Using where

- ORDER BYがインデックスの並び順以外になっている場合、使えるインデックスが無い場合に出現。内部で行をソート処理する。ソートの時にテンポラリファイルおよびバッファ領域を使用するため非常に遅い。
- ただし、クイックソートを使用しているため行数が少なければ影響は少ない
- JOINの中で出てきたら、非常に注意

Extra:Using index

Using filesort
Using index
Using index condition
Using index for group-by
Using join buffer (Block Nested Loop) Using join buffer (Batched Key Access)
Using MRR
Using temporary
Using where

- クエリが1つのインデックスにアクセスするだけで完結するもの。インデックス以外の行データにアクセスする必要がないため、非常に高速。これが出てくるようにチューニングすべき。

Extra:Using index condition / Using index for group-by

Using filesort
Using index
Using index condition
Using index for group-by
Using join buffer (Block Nested Loop) Using join buffer (Batched Key Access)
Using MRR
Using temporary
Using where

- Using index condition
  - インデックスコンディションプッシュダウンを行う（ストレージエンジン側でWHERE句による絞込みを行う）
- Using index for group-by
  - GROUP BY または DISTINCT を使ったクエリーを、インデックスだけで処理できる

Extra:Using join buffer

Using filesort
Using index
Using index condition
Using index for group-by
Using join buffer (Block Nested Loop) Using join buffer (Batched Key Access)
Using MRR
Using temporary
Using where

- JOINバッファを使用して結合処理を行う
- 結合アルゴリズムによって、Block Nested Loop / Batched Key Access が出力される

Extra:Using MRR

Using filesort
Using index
Using index condition
Using index for group-by
Using join buffer (Block Nested Loop) Using join buffer (Batched Key Access)
Using MRR
Using temporary
Using where

- MMR(Multi-Range Read)最適化を行う
  - セカンダリインデックスでの範囲スキャンを使用して行を読み取ると、テーブルが大きく、ストレージエンジンのキャッシュに格納されていない場合、ベーステーブルへのランダムディスクアクセスが多発する結果になることがあります。
  - Disk-Sweep Multi-Range Read (MRR) 最適化を使用すると、MySQL は、最初にインデックスだけをスキャンし、該当する行のキーを収集することによって、範囲スキャンのランダムディスクアクセスの回数を軽減しようとします。続いてキーがソートされ、最後に主キーの順序を使用してベーステーブルから行が取得されます。

https://dev.mysql.com/doc/refman/5.6/ja/mrr-optimization.html

Extra:Using temporary

Using filesort
Using index
Using index condition
Using index for group-by
Using join buffer (Block Nested Loop) Using join buffer (Batched Key Access)
Using MRR
Using temporary
Using where

- クエリの実行に内部的なテンポラリテーブルを使う。遅くなる傾向あり。
  - JOINの結果をさらにソート、ORDER BYとDISTINCTの併用、UNION、集合関数(SUM等)を使うとこれ。

Extra:Using where

Using filesort
Using index
Using index condition
Using index for group-by
Using join buffer (Block Nested Loop) Using join buffer (Batched Key Access)
Using MRR
Using temporary
Using where

- テーブルから行をフェッチした後に、さらにWHERE条件での絞り込みが必要なもの。WHERE句があって、インデックスが使われない場合にこれが出てくる。インデックスが使われても、インデックスでの検索後にさらにもう一段階のWHEREによる絞込みが必要な場合にも出てくる。

MySQL EXPLAIN結果まとめ

- select_typeで気をつけるべき事項
  - DEPENDENT UNIONおよび、DEPENDENT SUBQUERYは可能な限り避ける
    - サブクエリの結果1行1行に対して、外側の表のマッチング処理が行われるため、非常に工数が大きい
- typeで気をつけるべき事項
  - indexおよび、ALLは可能な限り避ける
  - ただしindexの場合で対象行数が少ない場合は例外としてindexでもOK
  - indexはたいていrefか、eq_refにできる
- Extraで気をつけるべき事項
  - Using Indexが理想的
  - Using where, Using filesort, Using temporaryは可能な限り避ける
- ＊は使わない

EXPLAINの出力フォーマット

- デフォルトは表形式

```
+----+-------------+---------+------------+-------+---------------+-------------+---------+--------------------+------+----------+-------------+
| id | select_type | table   | partitions | type  | possible_keys | key         | key_len | ref                | rows | filtered | Extra       |
+----+-------------+---------+------------+-------+---------------+-------------+---------+--------------------+------+----------+-------------+
|  1 | PRIMARY     | Country | NULL       | const | PRIMARY       | PRIMARY     | 3       | const              |    1 |      100 | Using index |
|  1 | PRIMARY     | a       | NULL       | ref   | CountryCode   | CountryCode | 3       | const              |  248 |    33.33 | Using where |
|  2 | SUBQUERY    | Country | NULL       | ALL   | PRIMARY       | NULL        | NULL    | NULL               |  239 |       10 | Using where |
|  2 | SUBQUERY    | City    | NULL       | ref   | CountryCode   | CountryCode | 3       | world.Country.Code |   18 |      100 | NULL        |
+----+-------------+---------+------------+-------+---------------+-------------+---------+--------------------+------+----------+-------------+
4 rows in set (0.0018 sec)
```

- JSON形式の出力も可能
  - **EXPLAIN FORMAT=JSON …**

```
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "43.55"
    },
    "nested_loop": [
 {
        "table": {
          "table_name": "Country",
```

```
 "access_type": "const",
          "possible_keys": [
            "PRIMARY"
          ],
          "key": "PRIMARY",
          "used_key_parts": [
            "Code"
 ],
          "key_length": "3",
…
```

他のコネクションで実行中のSQLの実行計画

- SQL文の文字列をコピーしてEXPLAINした場合と実際の実行時の挙動が異なる可能性がある
  - データそのものや統計情報が異なる場合など
- 実行中のSQLの実行計画を取得
  - **`EXPLAIN FOR CONNECTION connection_id;`**
  - ***`connection_id`***は**`SHOW PROCESS LIST`**で確認

https://dev.mysql.com/doc/refman/8.0/en/explain-for-connection.html

SQLチューニング実施

SQLチューニングの基本：インデックスの活用

- インデックスが使えていないクエリーは、インデックスを使って処理できないか検討する
  - インデックスを使うことで、表の中から少量のデータを高速に取り出せる
  - 大量データにアクセスする場合(※)は、インデックスを使わない方が高速になる

    ※例：表データの全件を取得する場合
  - UPDATEでインデックスが使えていない場合は、ロック待ちを過剰に発生させる可能性があるので、要注意
    - InnoDBでは、処理した行では無く、アクセスした行に対してロックを取得するため、1件しか更新しないUPDATE文であっても、インデックスが使えていない場合はテーブルロックになってしまう
    - トランザクション分離レベルをREAD COMMITTEDに変更している場合は、処理した行に対してのみロックを取得する（ギャップロックが無効化されるため）

SQLチューニングの基本：複数テーブルのJOIN

- JOINの順番と、各テーブルに対するアクセスパスが重要
  - 小さなテーブル（取り出す行数が少量のテーブル）から順番にJOINするのが基本
  - 2テーブルのJOINでも、どちらのテーブルに先にアクセスするかによっても効率は変わる
- 3テーブル以上JOINする時は、結果セットが少量になるテーブルからJOINする方が効率が良い(より絞込みができるテーブルからJOINする)
  - JOIN対象の列で絞込みをする場合など、WHERE句による絞込みをどのテーブルに対して実施する方が効率的かも考えて指定する

オプティマイザの制御

- オプティマイザ関連の設定を変更することで、オプティマイザの判断を変えることが出来る。その結果、実行計画が変わることがある。
  - コストの調整（MySQL 5.7以降）
  - optimizer_switchの変更
  - バッファサイズの変更（例：sort_buffer_size 、max_heap_table_size ）
  - その他の設定の変更（例：optimizer_prune_level 、optimizer_search_depth）
- ヒントを使うことで、オプティマイザに指示を出して実行計画を変更できる
  - より直接的な指定が可能

ヒントの種類

- インデックスヒント
  - 使用する（使用しない）インデックスを指定
- STRAIGHT_JOINヒント
  - JOINの順番を指定
- オプティマイザヒント
  - インデックスヒント、STRAIGHT_JOINヒント では指定できない、より詳細な制御が可能

インデックスヒントの種類

- USE INDEX
  - 特定のインデックスを使用するように、オプティマイザに指示を出す
- FORCE INDEX
  - USE INDEXに似ているが、USE INDEXに加えて
    「テーブルスキャンを選択しない」という指示を出す
- IGNORE INDEX
  - 特定のインデックスを使用しないように、オプティマイザに指示を出す

STRAIGHT_JOINヒント

- JOIN（結合）の順番を指定できるヒント
- 指定すると、左側のテーブルが右側のテーブルより先に読み取られる
  - MySQLがJOINで使用するアルゴリズムは Nested Loop JOIN とその改良系
  - Nested Loop JOIN では、基本的に先に読み取る表（駆動表）が後に読み取る表（内部表）よりも小さい方が効率的に結合できる
  - 3テーブル以上JOINする時は、結果セットが少量のテーブルからJOINする方が効率が良い(より絞込みができるテーブルからJOINする)
- OUTER JOIN（外部結合）の時は使えない
  - OUTER JOINの時は、結合条件に一致しない行を含むテーブルを先に読み取る必要があるため

Optimizer Hints

- ヒント構文:
  - **SELECT /*+ HINT1(*args*) HINT2(*args*) */ … FROM …**
- New hints:
  - BKA(*tables*)/NO_BKA(*tables*), BNL(*tables*)/NO_BNL(*tables*)
  - MRR(*table indexes*)/NO_MRR(*table indexes*)
  - SEMIJOIN/NO_SEMIJOIN(*strategies*), SUBQUERY(*strategy)*
  - NO_ICP(*table indexes*)
  - NO_RANGE_OPTIMIZATION(*table indexes*)
  - QB_NAME(*name*)
- optimizer_switchセッション変数より細かい粒度が可能

Query Rewrite Plugin

- アプリケーションの変更を加えることなく問題のあるクエリを書き直す
  - ヒントを追加
  - 結合順序の変更
  - その他 もろもろ …
- テーブルに書き換えルールを追加する:

```
INSERT INTO query_rewrite.rewrite_rules (pattern, replacement ) VALUES
  ("SELECT * FROM t1 WHERE a > ? AND b = ?",
    "SELECT * FROM t1 FORCE INDEX (a_idx) WHERE a > ? AND b = ?");
```

- 新しい解析前および解析後のクエリー・リライトAPI
  - ユーザーは独自のプラグインを作成できます

Query Rewrite Plugin

```
[mysqld]
rewriter_enabled=ON
```

```
SET GLOBAL rewriter_enabled = ON;
SET GLOBAL rewriter_enabled = OFF;
```

- MySQL 8.0.12以降: SELECT, INSERT, REPLACE, UPDATE, DELETE
- MySQL 8.0.11 まで: SELECTのみ

https://dev.mysql.com/doc/refman/8.0/en/rewriter-query-rewrite-plugin.html

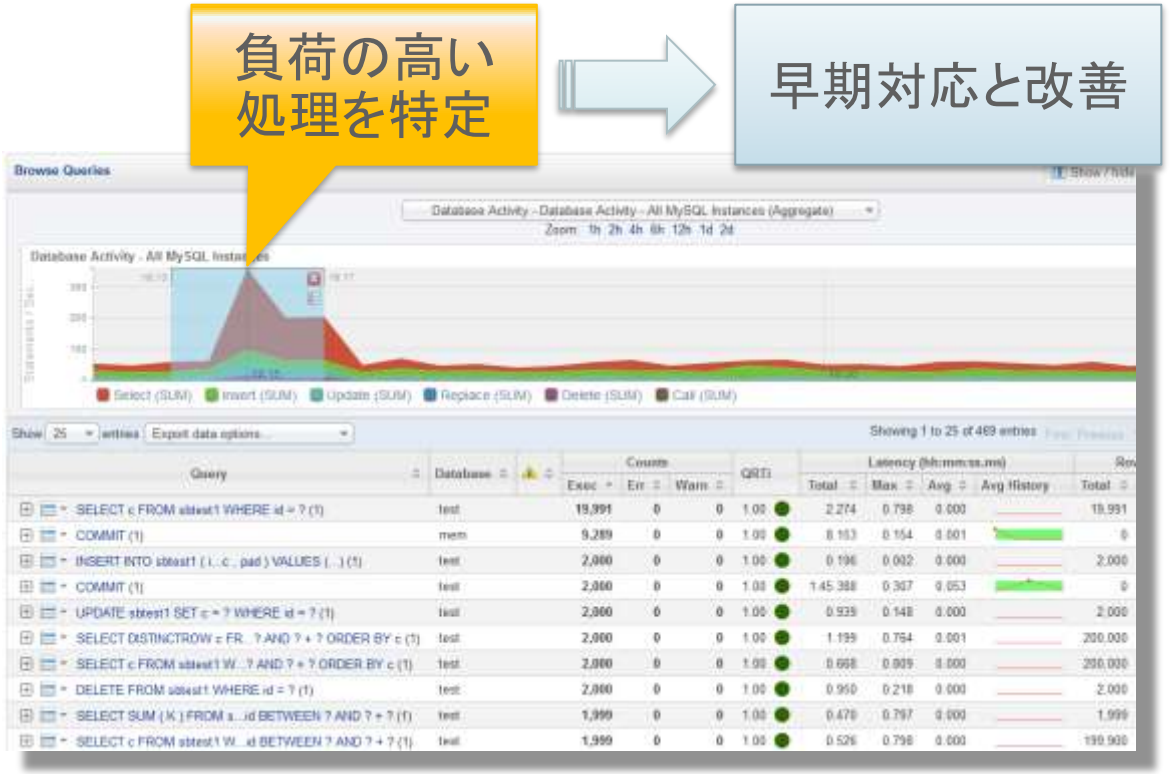
MySQL Subscription

- MySQL Standard Edition/Enterprise Edition(商用版)の契約があれば、チューニングに関する問合せもサポート対象になるため、弊社のサポートエンジニアからアドバイスを受けることが可能
- 他にも、パーティショニング設計のレビューやスキーマ設計のレビュー等も対応可能
- 詳細はこちらをご確認下さい
  - MySQL コンサルティング・サポート
    https://www-jp.mysql.com/support/consultative.html

クエリ解析機能 - MySQL Query Analyzer

商用版（EE）
限定機能

- 全てのMySQLサーバの
  全てのSQL文を一括監視
- vmstatなどのOSコマンドやMySQLの
  SHOWコマンドの実行、
  ログファイルの個別の監視は不要
- クエリの実行回数、エラー回数、
  実行時間、転送データ量などを
  一覧表示
- チューニングのための
  解析作業を省力化



Query Response Time Index (QRTi)

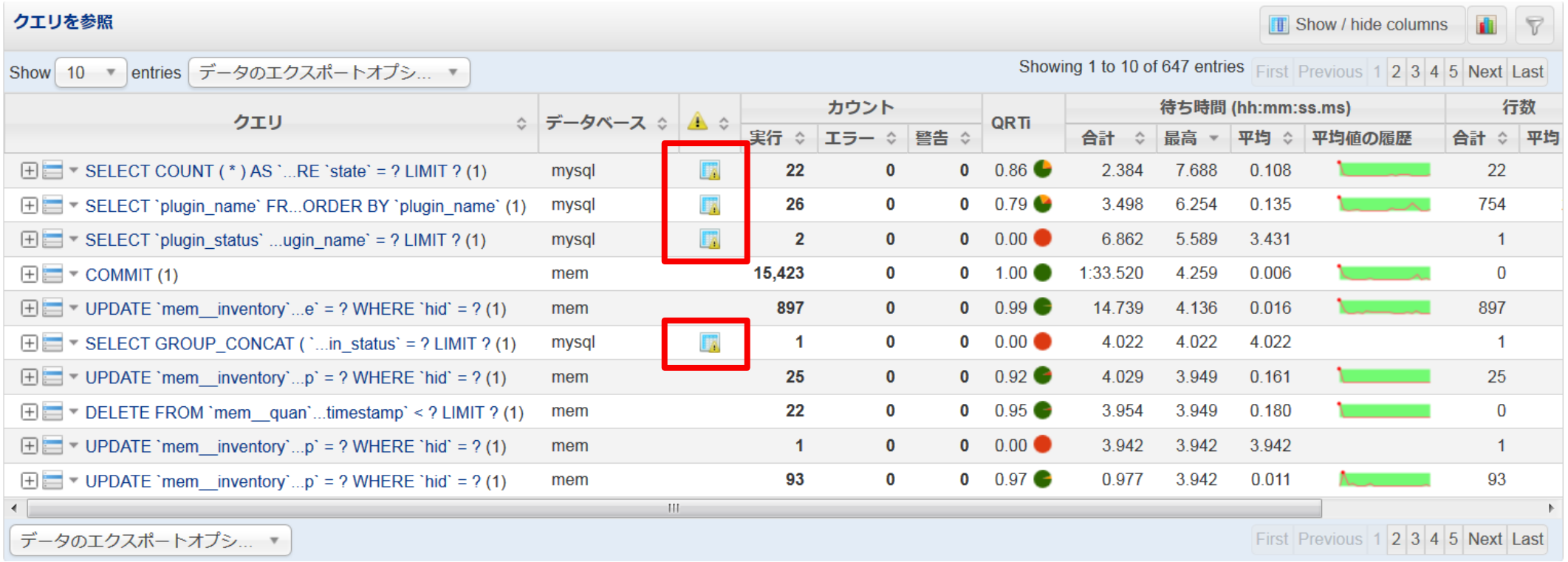
- 各クエリの「サービス品質」（QoS）を測定
- サーバ、グループ、またはすべてのインスタンスのQoS測定
- クエリパフォーマンス確認の為の単一測定基準

Query Response Time	
緑（最適）	< 100ms
橙（許容）	100ms < 400ms
赤（範囲外）	400ms <

Query	Database	⚠	Counts			QRTi
			Exec	Err	Warn	
INSERT INTO sbtest1 (i...c , pad) VALUES (...)	test		161,133	0	0	1.00
INSERT INTO sbtest1 (k...LUES (...) /* , ... */	test		20	0	0	0.50
SELECT SUM (K) FROM s...id BETWEEN ? AND ? + ?	test		161,156	0	0	1.00
SELECT DISTINCTROW c FR...? AND ? + ? ORDER BY c	test		161,160	0	0	1.00
CREATE INDEX k_1 ON sbtest1 (k)	test		1	0	0	0.00
SELECT c FROM sbtest1 W...? AND ? + ? ORDER BY c	test		161,157	0	0	1.00
BEGIN	test		161,139	0	0	1.00
CREATE TABLE sbtest1 (... = innodb MAX_ROWS = ?	test		1	0	0	0.50
COMMIT	test		161,091	0	0	0.69
DROP TABLE sbtest1	test		1	0	0	0.50

インデックスが使えていないクエリー

チューニング対象の可能性が高い



クエリを参照

Show / hide columns

Show 10 entries データのエクスポートオプシ...

Showing 1 to 10 of 647 entries First Previous 1 2 3 4 5 Next Last

クエリ	データベース		カウント 実行	エラー	警告	QRTi	待ち時間 (hh:mm:ss.ms) 合計	最高	平均	平均値の履歴	行数 合計	平均
SELECT COUNT (*) AS `...RE `state` = ? LIMIT ? (1)	mysql		22	0	0	0.86	2.384	7.688	0.108		22	
SELECT `plugin_name` FR...ORDER BY `plugin_name` (1)	mysql		26	0	0	0.79	3.498	6.254	0.135		754	
SELECT `plugin_status` ...ugin_name` = ? LIMIT ? (1)	mysql		2	0	0	0.00	6.862	5.589	3.431		1	
COMMIT (1)	mem		15,423	0	0	1.00	1:33.520	4.259	0.006		0	
UPDATE `mem__inventory`...e` = ? WHERE `hid` = ? (1)	mem		897	0	0	0.99	14.739	4.136	0.016		897	
SELECT GROUP_CONCAT (`...in_status` = ? LIMIT ? (1)	mysql		1	0	0	0.00	4.022	4.022	4.022		1	
UPDATE `mem__inventory`...p` = ? WHERE `hid` = ? (1)	mem		25	0	0	0.92	4.029	3.949	0.161		25	
DELETE FROM `mem__quan`...timestamp` < ? LIMIT ? (1)	mem		22	0	0	0.95	3.954	3.949	0.180		0	
UPDATE `mem__inventory`...p` = ? WHERE `hid` = ? (1)	mem		1	0	0	0.00	3.942	3.942	3.942		1	
UPDATE `mem__inventory`...p` = ? WHERE `hid` = ? (1)	mem		93	0	0	0.97	0.977	3.942	0.011		93	

データのエクスポートオプシ...

First Previous 1 2 3 4 5 Next Last

新しく実行されたクエリー

新しく実行されたクエリーは、まだ最適化されていない可能性あり

クエリを参照

Show / hide columns

Show 10 entries データのエクスポートオプシ...

Showing 1 to 10 of 450 entries First Previous 1 2 3 4 5 Next Last

	データベース	⚠	カウント			QRTi	待ち時間 (hh:mm:ss.ms)				行数		Temp Tables		初回実行
			実行	エラー	警告		合計	最高	平均	平均値の履歴	合計	平均	Total	Disk %	
BY TIMESTAMP (1)	mem		1	0	0	1.00	0.001	0.001	0.001		28	28	3	0	10:23:59
BY TIMESTAMP (1)	mem		1	0	0	1.00	0.001	0.001	0.001		28	28	3	0	10:23:59
BY TIMESTAMP (1)	mem		1	0	0	1.00	0.001	0.001	0.001		28	28	3	0	10:23:59
BY TIMESTAMP (1)	mem		1	0	0	1.00	0.001	0.001	0.001		28	28	3	0	10:23:59
ser_id` = ? (1)	mem		3	0	0	1.00	0.006	0.005	0.002		3	1	0	0	10:15:04
63_` , ...(1)	mem		4	0	0	1.00	0.010	0.013	0.002		8	2	0	0	2015/03/01 22:03:10
= ?) (1)	mem		1	0	0	1.00	0.001	0.017	0.001		2	2	0	0	2015/03/01 22:03:09
es` AS ...(1)	mem		2	0	0	1.00	0.002	0.180	0.001		44	22	0	0	2015/03/01 21:48:21
. ...(1)	mem		6	0	0	0.92	0.220	0.461	0.037		6	1	0	0	2015/03/01 21:42:33
ctions` ...(1)	mem		6	0	0	0.92	0.227	0.672	0.038		6	1	0	0	2015/03/01 21:42:33

データのエクスポートオプシ...

First Previous 1 2 3 4 5 Next Last

クエリー実行時間（合計、最高、平均）、平均値の履歴

突発的に遅延が発生したクエリーも探しやすい

クエリを参照

Show / hide columns

Show 10 entries データのエクスポートオプシ...

Showing 1 to 10 of 448 entries First Previous 1 2 3 4 5 Next Last

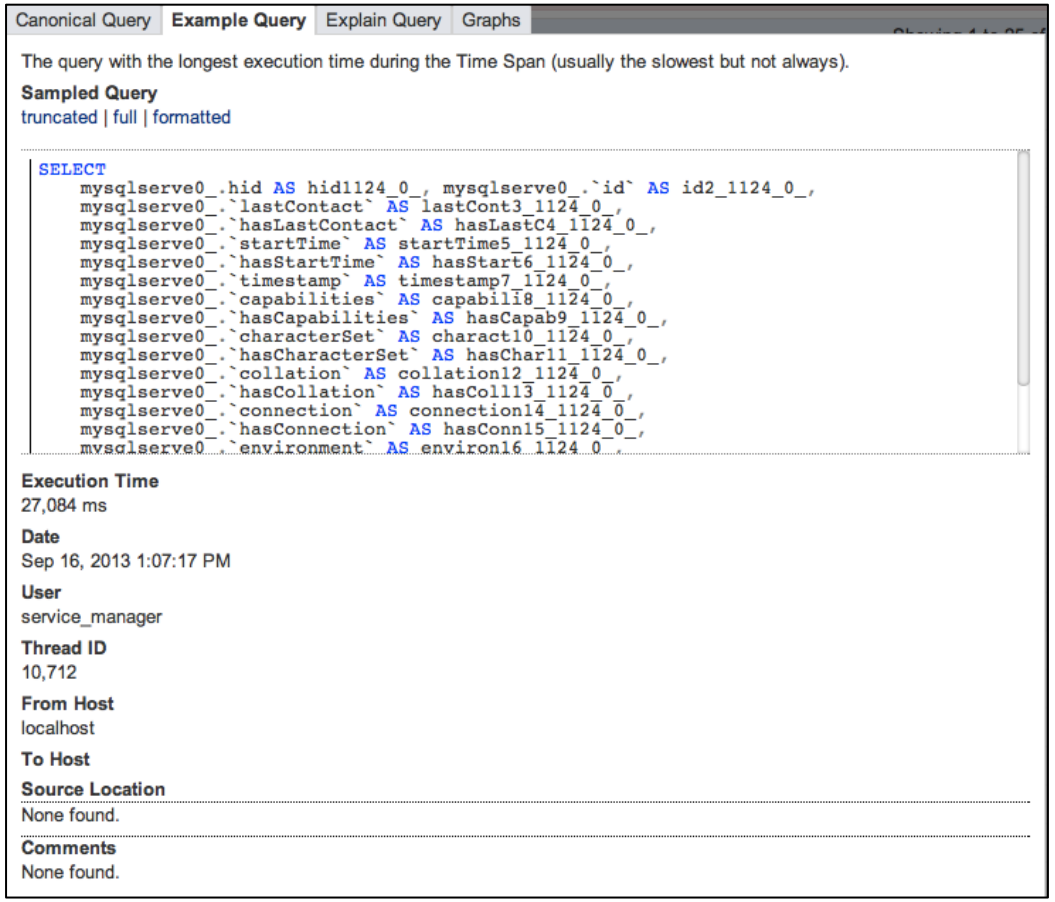
	データベース		カウント			QRTi	待ち時間 (hh:mm:ss.ms)				行数		Temp
			実行	エラー	警告		合計	最高	平均	平均値の履歴	合計	平均	Total
LUES (...(1)	mem		8,476	0	0	0.95	4:28.917	1.660	0.032		8,545	1	0
	mem		23,397	0	0	1.00	1:39.170	4.259	0.004		0	0	0
Seen`) (1)	mem		8,491	5	0	1.00	59.801	1.150	0.007		16,989	2	0
, ...(1)	mem		493	0	0	0.90	19.218	1.071	0.039		493	1	0
dTotal`) AS ...(1)	mem		48	0	0	0.50	12.150	2.220	0.253		480	10	96
OM (SELECT ...(1)	mem		48	0	0	0.50	10.050	0.542	0.209		480	10	96
E `hid` = ? (1)	mem		1,334	0	0	1.00	9.737	4.136	0.007		1,334	1	0
E `id` = ? (1)	mem		662	0	0	0.98	8.684	3.277	0.013		663	1	0
ible`) (1)	mem		29	0	0	0.71	5.071	0.778	0.175		29	1	0
_` . ...(1)	mem		66	0	0	0.89	3.538	1.248	0.054		66	1	0

データのエクスポートオプシ...

First Previous 1 2 3 4 5 Next Last

クエリーの詳細情報

実行時間/回数、影響を受けた行数の履歴、実行ホスト、実行計画、などの詳細を確認可能



Integrated Cloud

Applications & Platform Services


ORACLE®