## MySQL/MariaDB DB Optimizer history

<table>
<thead>
<tr>
<th>Disk access</th>
<th>Algorithms</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Observability
MariaDB 5.5

Disk access
- Batched Key Access
- Index Condition Pushdown

Algorithms
- Subquery optimizations
- Extended Keys

Statistics

Observability
MySQL 5.6

Disk access
- Batched Key Access
- Index Condition Pushdown

Algorithms
- Subquery optimizations
- Extended Keys

Statistics
- InnoDB persistent statistics
- Filesort w/ small limit

Observability
- EXPLAIN FORMAT=JSON
- EXPLAIN UPDATE/DELETE
- Optimizer trace
MariaDB 10.0

Disk access
- Batched Key Access
- Index Condition Pushdown

Algorithms
- Subquery optimizations
- Extended Keys
- EXISTS-to-IN
- Filesort w/ small limit

Statistics
- InnoDB persistent statistics
- Engine-Independent Statistics (Histograms)

Observability
- EXPLAIN FORMAT=JSON
- SHOW EXPLAIN
- EXPLAIN in slow query log
- EXPLAIN UPDATE/DELETE
- Optimizer trace
New features in MariaDB 10.0 Optimizer

- **Statistics**
  - Engine-independent statistics (Histograms)

- **Algorithms**
  - Subquery optimizations: EXISTS-to-IN

- **EXPLAIN improvements**
  - SHOW EXPLAIN
  - EXPLAIN in the slow query log
  - EXPLAIN for UPDATE/DELETE
New features in MariaDB 10.0 Optimizer

- **Statistics**
  - Engine-independent statistics (Histograms)

- **Algorithms**
  - Subquery optimizations: EXISTS-to-IN

- **EXPLAIN improvements**
  - SHOW EXPLAIN
  - EXPLAIN in the slow query log
  - EXPLAIN for UPDATE/DELETE
Query optimizer needs statistics

```sql
select *
from
    lineitem, orders
where
    o_orderkey=l_orderkey and
    o_orderdate between '1990-01-01' and '1998-12-06' and
    l_extendedprice > 1000000
```
Sources of statistics

- Total number of rows in table
  - orders - …
  - lineitem -…

- Number of rows that match parts of WHERE

```
select *
from
  lineitem, orders

where
  o_orderkey=l_orderkey and
  o_orderdate between '1990-01-01' and '1998-12-06' and
  l_extendedprice > 1000000;
```

- “key CMP const” - range condition
  - Precise estimates
  - Requires an index
  - B-Tree index dives
  - Expensive to get.
Sources of statistics (2)

where

\[ o_{\text{orderkey}} = l_{\text{orderkey}} \text{ and } o_{\text{orderdate}} \text{ between '1990-01-01' and '1998-12-06'} \text{ and } l_{\text{extendedprice}} > 1000000; \]

- \text{orders}.o_{\text{orderkey}}=\ldots
  \text{lineitem}.l_{\text{orderkey}}=\ldots
  \begin{itemize}
    \item “how many lineitem-s for some $orderkey$”?
    \item Index statistics
  \end{itemize}

- InnoDB
  \begin{itemize}
    \item Random sampling (\texttt{innodb_stats_sample_pages})
    \item At “random” times
  \end{itemize}

- MySQL 5.6: persistent statistics
  \begin{itemize}
    \item Still random sampling.
Problems with statistics

• Index statistics: randomness
  • Collected via sampling
  • Different query plans
    • Even for DBT-3 benchmark
  • InnoDB Persistent Statistics is a partial solution

• Not index? no statistics
  • not_indexed_column= 'foo'
  • key_column2= 'bar'.
MariaDB 10.0: Engine independent statistics

- No random sampling
  - ANALYZE TABLE makes full table scan
    - No auto-updates
  - Statistics is deterministic
- More statistics data
  - #rows in the table (same)
  - Index statistics: cardinality (same)
  + Statistics on non-indexed columns
    + MIN/MAX values
    + Frequency
    + #NULL values
    + Histograms.
New statistics test run

```sql
select *
from
    lineitem, orders
where
    o_orderkey=l_orderkey and
    o_orderdate between '1990-01-01' and '1998-12-06' and
    l_extendedprice > 1000000
```

```
+----+----------+---------+---------------+-------------+-------+-----------------+-------+--------+-----------+
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>possible_keys</th>
<th>key</th>
<th>key_len</th>
<th>ref</th>
<th>rows</th>
<th>filtered</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SIMPLE</td>
<td>orders</td>
<td>ALL</td>
<td>PRIMARY</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>1494230</td>
<td>100.00</td>
<td>Using where</td>
</tr>
<tr>
<td>1</td>
<td>SIMPLE</td>
<td>lineitem</td>
<td>ref</td>
<td>PRIMARY,i_...</td>
<td>PRIMARY</td>
<td>4</td>
<td>orders.o_orderkey</td>
<td>2</td>
<td>100.00</td>
<td>Using where</td>
</tr>
</tbody>
</table>
```

- 4.2 seconds
- filtered=100%
  - Close to truth for `o_orderdate between ...`
  - Far from truth for `l_extendedprice > 1000000`. 
New statistics test run (2)

• Collect table statistics

```sql
set histogram_size=200;
set use_stat_tables='preferably'
analyze table lineitem, orders;
```

<table>
<thead>
<tr>
<th>Table</th>
<th>Op</th>
<th>Msg_type</th>
<th>Msg_text</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dbt3sf1.lineitem</td>
<td>analyze</td>
<td>status</td>
<td>Engine-independent statistics collected</td>
<td></td>
</tr>
<tr>
<td>dbt3sf1.lineitem</td>
<td>analyze</td>
<td>status</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>dbt3sf1.orders</td>
<td>analyze</td>
<td>status</td>
<td>Engine-independent statistics collected</td>
<td></td>
</tr>
<tr>
<td>dbt3sf1.orders</td>
<td>analyze</td>
<td>status</td>
<td>OK</td>
<td></td>
</tr>
</tbody>
</table>

• Make the optimizer use it

```sql
set optimizer_use_condition_selectivity=4;
```
### New statistics test run (3)

- **Re-run the query**

```sql
select *
from
  lineitem, orders
where
  o_orderkey=l_orderkey and
  o_orderdate between '1990-01-01' and '1998-12-06' and
  l_extendedprice > 1000000
```

- lineitem.filtered=0.5% - **l_extendedprice > 1000000**
- 1.5 sec (from 4.2 sec)
  - Can be more for many-table joins.
New statistics summary

- Persistent, stable statistics
- Statistics for
  - Indexes
  - Columns (Histograms)
- Need to enable it and collect manually
  1. Collect statistics
     
     ```
     set histogram_size=...;
     set use_stat_tables='preferably' -- 'complementary'
     analyze table tbl [persistent for columns (...) indexes (...)]
     ```
  2. Make optimizer use it
     
     ```
     set optimizer_use_condition_selectivity = 4;
     ```
  3. Watch EXPLAIN EXTENDED, filtered%.
Further details

- Documentation update pending
  - Will blog
- Percona Live talk today:
  - MariaDB Optimizer: See? No indexes!
  - 4:30PM - 5:20PM @ Ballroom E.
New features in MariaDB 10.0 Optimizer

- **Statistics**
  - Engine-independent statistics (Histograms)

- **Algorithms**
  - Subquery optimizations: EXISTS-to-IN

- **EXPLAIN improvements**
  - SHOW EXPLAIN
  - EXPLAIN in the slow query log
  - EXPLAIN for UPDATE/DELETE
Subquery optimizations background

- Traditional way: “straightforward” subquery execution

```sql
select * from customer
where
customer.c_acctbal < 0 and
exists (select 1
  from orders
where
  orders.o_custkey= customer.c_custkey AND
  orders.o_orderDATE between $DAY1 and $DAY2)
```

<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>possible_keys</th>
<th>key</th>
<th>key_len</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>customer</td>
<td>ALL</td>
<td>c_acctbal,i_c_acctbal_nationkey</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>150303</td>
<td>Using where</td>
</tr>
<tr>
<td>2</td>
<td>DEPENDENT SUBQUERY</td>
<td>orders</td>
<td>ref</td>
<td>i_o_orderdate,i_o_custkey</td>
<td>i_o_custkey</td>
<td>5</td>
<td>customer.c_custkey</td>
<td>7</td>
<td>Using where</td>
</tr>
</tbody>
</table>
```
Subquery optimizations background

- Traditional way: “straightforward” subquery execution

```
select * from customer
where
  customer.c_acctbal < 0 and
exists (select 1
  from orders
where
  orders.o_custkey= customer.c_custkey AND
  orders.o_orderDATE between $DAY1 and $DAY2)
```

- Outer-to-inner execution
- Every subquery re-execution re-runs it from the start
  - MariaDB 5.5 added subquery predicate cache.
Lots of optimizations for IN (SELECT ...)

- Materialization
  - Subquery must be uncorrelated
- Semi-join optimizations
  - Both inner-to-outer and outer-to-inner execution
  - Materialization
  - Join-like execution methods

```sql
select * from customer
where
  customer.c_acctbal < 0 and
  customer.c_custkey IN (select orders.o_custkey
                           from orders
                           where
                             orders.o_orderDATE between ...
                           )
```
MariaDB 10.0 – EXISTS subqueries

1. Convert EXISTS into semi-join

```sql
select * from customer
where
customer.c_acctbal < 0 and
exists (select 1
from orders
where
orders.o_custkey = customer.c_custkey AND
orders.o_orderDATE between $DAY1 and $DAY2)
```

Semi-join optimizations
- Both inner-to-outer and outer-to-inner execution
- Join-like execution methods
2. Convert **trivially-correlated** EXISTS into IN

```sql
EXISTS (select 1
    from orders
    where
    orders.o_custkey=customer.c_custkey AND
    orders.o_orderDATE between $DAY1 and $DAY2)
```

↓

```sql
orders.o_custkey IN (select customer.c_custkey
    from orders
    where
    orders.o_orderDATE between $DAY1 and $DAY2)
```

Uncorrelated subquery => can use Materialization
EXISTS-to-IN summary

• Enables new subquery optimizations for EXISTS
  • WHERE EXISTS(...) gets semi-joins
    • Inner-to-outer execution
  • Trivially-correlated EXISTS get Materialization

• How to use
  • set optimizer_switch='exists_to_in=on'
  • Enjoy
New features in MariaDB 10.0 Optimizer

- **Statistics**
  - Engine-independent statistics (Histograms)

- **Algorithms**
  - Subquery optimizations: EXISTS-to-IN

- **EXPLAIN improvements**
  - SHOW EXPLAIN
  - EXPLAIN in the slow query log
  - EXPLAIN for UPDATE/DELETE
SHOW EXPLAIN

• Shows EXPLAIN of a running query

client1> select count(*) from customer, orders, nation where c_custkey=o_custkey and c_nationkey=n_nationkey ...

client2> show processlist;
+----+------+-----------+------+---------+------+--------------+--------------------------------------------------------+
| Id | User | Host      | db   | Command | Time | State        | Info                                                                 |
+----+------+-----------+------+---------+------+--------------+---------------------------------------------------------------------+
|  2 | root | localhost | test | Query   |    0 | init         | show processlist                                                     |
|  3 | root | localhost | db   | Query   |    4 | Sending data | select count(*) from customer, orders, nation where ...              |
+----+------+-----------+------+---------+------+--------------+---------------------------------------------------------------------+

client2> show explain for 3;
+----+-------------+----------+------+---------------+---------------+---------+-----------------------+-------+-------------+
| id | select_type | table    | type | possible_keys | key           | key_len | ref                   | rows  | Extra       |
+----+-------------+----------+------+---------------+---------------+---------+-----------------------+-------+-------------+
|  1 | SIMPLE      | nation   | ALL  | PRIMARY       | NULL          | NULL    | NULL                  |    25 | Using where |
|  1 | SIMPLE      | customer | ref  | i_c_nationkey | i_c_nationkey | 5       | db.nation.n_nationkey | 83512 | Using index |
|  1 | SIMPLE      | orders   | ref  | i_o_custkey   | i_o_custkey   | 5       | db.customer.c_custkey |     8 | Using index |
+----+-------------+----------+------+---------------+---------------+---------+-----------------------+-------+-------------+
3 rows in set, 1 warning (0.00 sec)

client2> show warnings;
+-------+------+--------------------------------------------------------------------------------------------------------------------------+
| Level | Code | Message                                                                                                                  |
+-------+------+--------------------------------------------------------------------------------------------------------------------------+
| Note  | 1003 | select count(*) from customer, orders, nation where c_custkey=o_custkey and c_nationkey=n_nationkey and n_name='GERMANY' |
+-------+------+--------------------------------------------------------------------------------------------------------------------------+
SHOW EXPLAIN (2)

- SHOW EXPLAIN vs EXPLAIN $query_str
  - Can see exactly what query plan is running
  - EXPLAIN $query_str can be difficult
    - @session_vars
    - Per-connection settings (@@optimizer_switch?)
    - temporary tables
    - Uncommitted changes in a transaction
    - Etc, etc

- MySQL 5.7: EXPLAIN FOR CONNECTION $id
EXPLAIN in the slow query log

- **log_slowverbosity=explain**

```plaintext
my.cnf

... slow-query-log log-long-query-time=1 log-slow-verbosity=query_plan,explain

$datadir/hostname-slow.log
```

```plaintext
# Time: 140402  6:55:35  
# User@Host: root[root] @ localhost []  
# Thread_id: 2  Schema: dbt3sf10  QC_hit: No  
# Query_time: 15.532745  Lock_time: 0.378632  Rows_sent: 1  Rows_examined: 661743  
# Full_scan: Yes  Full_join: No  Tmp_table: No  Tmp_table_on_disk: No  
# Filesort: No  Filesort_on_disk: No  Merge_passes: 0  
# explain: id  select_type  table  type  possible_keys  key  key_len  ref  rows  Extra  
# explain: 1  SIMPLE  nation  ALL  PRIMARY  NULL  NULL  NULL  25  Using where  
# explain: 1  SIMPLE  customer  ref  i_c_nationkey  i_c_nationkey  5  db.nation.n_nationkey82649  Using  
# explain: 1  SIMPLE  orders  ref  i_o_custkey  i_o_custkey  5  db.customer.c_custkey7  Using  
```

```plaintext
use dbt3sf10;  
SET timestamp=1396446935;  
select count(*) from customer, orders, nation where c_custkey=0_custkey and c_nationkey=n_nationkey and n_name='GERMANY';
```
EXPLAIN in the slow query log (2)

- **Machine-parseable**
  - Lines start with "# explain: "
  - Cells are tab-delimited

- Logs with explains are parsed by `pt-query-digest`
  - Would like to make it report query plan stats
Other EXPLAIN improvements

• EXPLAIN UPDATE/DELETE/INSERT
  • MySQL 5.6 → MariaDB 10.0
• Not ported from MySQL 5.6 so far:
  • EXPLAIN FORMAT=JSON
  • Optimizer trace
MariaDB 10.0 summary

- Porting features from MySQL 5.6
- Also adding our own new features
- Actually leading the development
Beyond MariaDB 10.0

- EXPLAIN FORMAT=JSON
- EXPLAIN ANALYZE [in the slow query log]
- More statistics
  - More histogram types
  - Observability
  - Better use in the optimizer
- ORDER/GROUP BY ... LIMIT: fix the mess
- Your input
  - ...

Notice: MySQL is a registered trademark of Sun Microsystems, Inc.
Thanks!

Q&A