FOSDEM 2016

ANALYZE for statements

MariaDB's new tool for diagnosing the optimizer

Sergei Petrunia
MariaDB

Background: optimizer troubleshooting

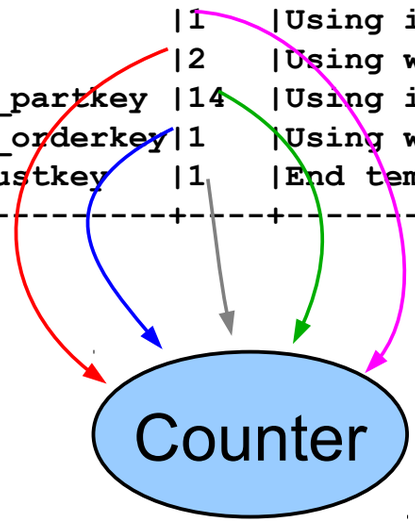
- EXPLAIN shows the query plan

```
+----+-----------+--------+----+-------------+------+------+----------------+------+-----------+
|id  |select_type|table   |type|possible_keys|key   |key_len|ref             |rows  |Extra      |
+----+-----------+--------+----+-------------+------+------+----------------+------+-----------+
|   1|SIMPLE     |orders  |ALL |PRIMARY,...  |NULL  |NULL  |NULL            |1507320|Using where|
|   1|SIMPLE     |lineitem|ref |PRIMARY,...  |PRIMARY|4     |orders.o_orderkey|      1|Using where|
+----+-----------+--------+----+-------------+------+------+----------------+------+-----------+
```

- Sometimes problem is apparent
- Sometimes not
  - Query plan vs reality?
  - Where the time was spent?

Counters give summary information

```
+--+-----------+--------+------+-------------+--------+-------+------------------+----+--------------------------+
|id|select_type|table   |type  |possible_keys|key     |key_len|ref               |rows|Extra                     |
+--+-----------+--------+------+-------------+--------+-------+------------------+----+--------------------------+
|1 |PRIMARY    |orders  |const |PRIMARY      |PRIMARY |4      |const             |1   |Using index               |
|1 |PRIMARY    |lineitem|ref   |PRIMARY,i_...|PRIMARY |4      |const             |2   |Using where; Start temporary|
|1 |PRIMARY    |lineitem|ref   |PRIMARY,i_...|i_suppkey|5     |lineitem.l_partkey|14  |Using index               |
|1 |PRIMARY    |orders  |eq_ref|PRIMARY,i_...|PRIMARY |4      |lineitem.l_orderkey|1  |Using where               |
|1 |PRIMARY    |customer|eq_ref|PRIMARY      |PRIMARY |4      |orders.o_custkey  |1   |End temporary             |
+--+-----------+--------+------+-------------+--------+-------+------------------+----+--------------------------+
```



- Slow query log: **Rows_examined**
- **Handler_XXX** status variables
- Userstat:

  `SHOW (TABLE|INDEX)_STATISTICS`
- PERFORMANCE_SCHEMA:

  `table_io_waits_summary_by_table`

- All are query-global
- Or server-global
- => Analysis is difficult.

Solution in MariaDB 10.1

ANALYZE statement

- Like EXPLAIN ANALYZE in PostgreSQL
- Or V$SQL_PLAN_STATISTICS in Oracle

ANALYZE vs EXPLAIN

EXPLAIN

- Optimize the query
- Produce EXPLAIN output

ANALYZE

- Optimize the query
- Run the query
  - Collect execution statistics
  - Discard query output
- Produce EXPLAIN output
  - With also execution statistics

Tabular ANALYZE statement

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

```
+--+-----------+-------+----+-------------+------+-------+----------------+-------+-------+--------+----------+-----------+
|id|select_type|table  |type|possible_keys|key   |key_len|ref             |rows   |r_rows |filtered|r_filtered|Extra      |
+--+-----------+-------+----+-------------+------+-------+----------------+-------+-------+--------+----------+-----------+
|1 |SIMPLE     |orders |ALL |PRIMARY,i_...|NULL  |NULL   |NULL            |1504278|1500000|  50.00 |   100.00 |Using where|
|1 |SIMPLE     |lineitem|ref |PRIMARY,i_...|PRIMARY|4      |orders.o_orderkey|2      |4.00   | 100.00 |     0.00 |Using where|
+--+-----------+-------+----+-------------+------+-------+----------------+-------+-------+--------+----------+-----------+
```

r_ is for “real”

- **r_rows –** observed #rows
- **r_filtered** – observed condition selectivity.

Interpreting **r_rows**

```
+--+-----------+--------+----+-------------+-------+-------+-----------------+-------+-------+--------+----------+-----------+
|id|select_type|table   |type|possible_keys|key    |key_len|ref              |rows   |r_rows |filtered|r_filtered|Extra      |
+--+-----------+--------+----+-------------+-------+-------+-----------------+-------+-------+--------+----------+-----------+
|1 |SIMPLE     |orders  |ALL |PRIMARY,i_...|NULL   |NULL   |NULL             |1504278|1500000|  50.00 |   100.00 |Using where|
|1 |SIMPLE     |lineitem|ref |PRIMARY,i_...|PRIMARY|4      |orders.o_orderkey|2      |4.00   | 100.00 |     0.00 |Using where|
+--+-----------+--------+----+-------------+-------+-------+-----------------+-------+-------+--------+----------+-----------+
```

- ALL/index
  - r_rows ≈ rows
  - Different in case of LIMIT or subqueries
- range/index_merge
  - Up to ~2x difference with InnoDB
  - Bigger difference in edge cases (IGNORE INDEX?)

Interpreting **r_rows** (2)

```
+--+-----------+--------+----+-------------+-------+-------+-----------------+-------+-------+--------+----------+-----------+
|id|select_type|table   |type|possible_keys|key    |key_len|ref              |rows   |r_rows |filtered|r_filtered|Extra      |
+--+-----------+--------+----+-------------+-------+-------+-----------------+-------+-------+--------+----------+-----------+
|1 |SIMPLE     |orders  |ALL |PRIMARY,i_...|NULL   |NULL   |NULL             |1504278|1500000|  50.00 |   100.00 |Using where|
|1 |SIMPLE     |lineitem|ref |PRIMARY,i_...|PRIMARY|4      |orders.o_orderkey|2      |4.00   | 100.00 |     0.00 |Using where|
+--+-----------+--------+----+-------------+-------+-------+-----------------+-------+-------+--------+----------+-----------+
```

For **ref** access

- rows is AVG(records for a key)
- Some discrepancy is normal
- Big discrepancy (>10x) is worth investigating
  - rows=1, r_rows >> rows ? No index statistics (ANALYZE TABLE)
  - Column has lots of NULL values? (innodb_stats_method)
  - Skewed data distribution?
    - Complex, IGNORE INDEX.

Interpreting **r_filtered**

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

```
+--+-----------+--------+----+-------------+-------+-------+----------------+-------+-------+--------+----------+-----------+
|id|select_type|table   |type|possible_keys|key    |key_len|ref             |rows   |r_rows |filtered|r_filtered|Extra      |
+--+-----------+--------+----+-------------+-------+-------+----------------+-------+-------+--------+----------+-----------+
|1 |SIMPLE     |orders  |ALL |PRIMARY,i_...|NULL   |NULL   |NULL            |1504278|1500000|  50.00 |   100.00 |Using where|
|1 |SIMPLE     |lineitem|ref |PRIMARY,i_...|PRIMARY|4      |orders.o_orderkey|2      |4.00   | 100.00 |     0.00 |Using where|
+--+-----------+--------+----+-------------+-------+-------+----------------+-------+-------+--------+----------+-----------+
```

- **(r_)filtered** is selectivity of “Using where”
- **r_filtered** << 100% → reading and discarding lots of rows
  - Check if conditions allow index use
    - Use EXPLAIN|ANALYZE FORMAT=JSON to see the condition
  - Consider adding indexes
  - Don't chase r_filtered=100.00, tradeoff between reads and writes.

r_filtered and query plans

- **filtered**
  - Shows how many rows will be removed from consideration
  - Is important for N-way join optimization
- **r_filtered ≉ filtered**
  - Optimizer doesn't know condition selectivity → poor plans
- MariaDB's data for filtered: EITS Statistics (Histograms)
  - Improves plans for complex queries

```
SET histogram_size=100;
for each table $tbl with r_filtered ≉ filtered {
    ANALYZE TABLE $tbl PERSISTENT FOR ALL;
}
```

Tabular **ANALYZE** summary

```
+--+-----------+--------+----+-------------+------+-------+-----------------+-------+-------+--------+----------+-----------+
|id|select_type|table   |type|possible_keys|key   |key_len|ref              |rows   |r_rows |filtered|r_filtered|Extra      |
+--+-----------+--------+----+-------------+------+-------+-----------------+-------+-------+--------+----------+-----------+
|1 |SIMPLE     |orders  |ALL |PRIMARY,i_...|NULL  |NULL   |NULL             |1504278|1500000|  50.00 |   100.00 |Using where|
|1 |SIMPLE     |lineitem|ref |PRIMARY,i_...|PRIMARY|4      |orders.o_orderkey|2      |4.00   | 100.00 |     0.00 |Using where|
+--+-----------+--------+----+-------------+------+-------+-----------------+-------+-------+--------+----------+-----------+
```

- New columns
  - **r_rows**
  - **r_filtered**
- Can check estimates vs reality
- Can find [possibly] sub-optimal plans
- Can find where optimizer is not aware of condition selectivity.

ANALYZE FORMAT=JSON

EXPLAIN FORMAT=JSON + ANALYZE = ANALYZE FORMAT=JSON

ANALYZE FORMAT=JSON


```
{
  "query_block": {
    "select_id": 1,
    "r_loops": 1,
    "r_total_time_ms": 191747,
    "table": {
      "table_name": "orders",
      "access_type": "ALL",
      "possible_keys": ["PRIMARY", "i_o_orderdate"],
      "r_loops": 1,
      "rows": 1498194,
      "r_rows": 1.5e6,
      "r_total_time_ms": 14261,
      "filtered": 50,
      "r_filtered": 100,
      "attached_condition": "(orders.o_orderDATE between 1990-01-01 and 1998-12-06)"
    },
    "table": {
      "table_name": "lineitem",
      "access_type": "ref",
      "possible_keys": ["PRIMARY", "i_l_orderkey",  "i_l_orderkey_quantity"],
      "key": "PRIMARY",
      "key_length": "4",
      "used_key_parts": ["l_orderkey"],
      "ref": ["dbt3sf1.orders.o_orderkey"],
      "r_loops": 1500000,
      "rows": 1,
      "r_rows": 4.0008,
      "r_total_time_ms": 170456,
      "filtered": 100,
      "r_filtered": 0,
      "attached_condition": "(lineitem.l_extendedprice > 1000000)"
    }
  }
}
```

ANALYZE FORMAT=JSON


```
{
  "query_block": {
    "select_id": 1,
    "r_loops": 1,
    "r_total_time_ms": 191747,
    "table": {
      "table_name": "orders",
      "access_type": "ALL",
      "possible_keys": ["PRIMARY", "i_o_orderdate"],
      "r_loops": 1,
      "rows": 1498194,
      "r_rows": 1.5e6,
      "r_total_time_ms": 14261,
      "filtered": 50,
      "r_filtered": 100,
      "attached_condition": "(orders.o_orderDATE between 1990-01-01 and 1998-12-06)"
    },
    "table": {
      "table_name": "lineitem",
      "access_type": "ref",
      "possible_keys": ["PRIMARY", "i_l_orderkey",  "i_l_orderkey_quantity"],
      "key": "PRIMARY",
      "key_length": "4",
      "used_key_parts": ["l_orderkey"],
      "ref": ["dbt3sf1.orders.o_orderkey"],
      "r_loops": 1500000,
      "rows": 1,
      "r_rows": 4.0008,
      "r_total_time_ms": 170456,
      "filtered": 100,
      "r_filtered": 0,
      "attached_condition": "(lineitem.l_extendedprice > 1000000)"
    }
  }
}
```



```
"table_name": "orders",
"access_type": "ALL",
"possible_keys": ["PRIMARY
"r_loops": 1,
"rows": 1498194,
"r_rows": 1.5e6,
"r_total_time_ms": 14261,
```

```
"table_name": "lineitem",
"r_loops": 1500000,
"rows": 1,
"r_rows": 4.0008,
"r_total_time_ms": 170456,
"filtered": 100,
"r_filtered": 0,
```

ANALYZE FORMAT=JSON basics

- Structured like EXPLAIN FORMAT=JSON
  - Subquery – **query_block**
  - Table access – **table**
  - Also nodes for **filesort**, **materialized**, **temptable**, etc
- ANALYZE data in each node
  - **r_** members
  - **r_rows**, **r_filtered** – like in tabular form
  - **r_loops** – number of times node executed
  - **r_total_time_ms** ← !!!

r_total_time_ms is very useful

```
+--+-----------+--------+----+-----------------------------------------+------------+-------+------------------+-------+-----------+
|id|select_type |table    |type|possible_keys                            |key         |key_len|ref               |rows   |Extra      |
+--+-----------+--------+----+-----------------------------------------+------------+-------+------------------+-------+-----------+
|1 |PRIMARY     |customer|ref |PRIMARY,i_c_nationkey                    |i_c_nationkey|5      |const             |6008   |           |
|1 |PRIMARY     |orders  |ref |PRIMARY,i_o_custkey                      |i_o_custkey  |5      |customer.c_custkey|7      |           |
|1 |PRIMARY     |lineitem|ref |PRIMARY,i_l_orderkey,i_l_orderkey_quantity|PRIMARY      |4      |orders.o_orderkey |2      |Using where|
|2 |MATERIALIZED|orders  |ALL |i_o_custkey                              |NULL        |NULL   |NULL              |1501236|Using where|
+--+-----------+--------+----+-----------------------------------------+------------+-------+------------------+-------+-----------+
```

ANALYZE FORMAT=JSON

```
"table_name": "customer",
"r_total_time_ms": 1354.8,
```

```
"table_name": "orders",
"r_total_time_ms": 11444,
```

#2

```
"table_name": "lineitem",
"r_total_time_ms": 22040,
```

#1

```
"table_name": "orders",
"r_total_time_ms": 10493,
```

#3

- **rows**
  - can be confusing with joins or subqueries
  - Not all rows are equal
- **r_total_time_ms** shows which table/subquery took the time.

Buffer sizes

- join_buffer_size, sort_buffer_size
  - What size is used? Is it enough?

```
select * from t1, t2
where t1.col1<100 and t2.col1<100 and t1.col2=t2.col2
```

```
+--+-----------+-----+----+-------------+----+-------+----+----+-----------------------------------------------+
|id|select_type|table|type|possible_keys|key |key_len|ref |rows|Extra                                          |
+--+-----------+-----+----+-------------+----+-------+----+----+-----------------------------------------------+
|1 |SIMPLE     |t2   |ALL |NULL         |NULL|NULL   |NULL|820 |Using where                                    |
|1 |SIMPLE     |t1   |ALL |NULL         |NULL|NULL   |NULL|889 |Using where; Using join buffer (flat, BNL join)|
+--+-----------+-----+----+-------------+----+-------+----+----+-----------------------------------------------+
```

Join buffer optimization

- Reads rows into buffer, then sorts
- Can see it used in EXPLAIN
  - But what about buffer size?

ANALYZE and buffer sizes

```
"query_block": {
  "select_id": 1,
  "r_loops": 1,
  "r_total_time_ms": 3.5363,
  "table": {
    "table_name": "t2",
    "access_type": "ALL",
    "r_loops": 1,
    "rows": 820,
    "r_rows": 1000,
    "r_total_time_ms": 0.8818,
    "filtered": 100,
    "r_filtered": 10,
    "attached_condition": "(t2.col1 < 100)"
},
```

```
"block-nl-join": {
  "table": {
    "table_name": "t1",
    "access_type": "ALL",
    "r_loops": 1,
    "rows": 889,
    "r_rows": 1000,
    "r_total_time_ms": 0.875,
    "filtered": 100,
    "r_filtered": 10,
    "attached condition": "(t1.col1 < 100)"
  },
  "buffer_type": "flat",
  "buffer_size": "128Kb",
  "join_type": "BNL",
  "attached_condition": "(t1.col2 = t2.col2)",
}
```

Can see buffers used

- Can see used buffer size
- Child's **r_loops** gives a clue whether buffer refilled.

Range checked for each record

```
select * from orders A, orders B
where
  A.o_clerk='Clerk#000000001' and
  B.o_orderdate between DATE_SUB(A.o_orderdate, interval 1 day) and
                        DATE_ADD(A.o_orderdate, interval 1 day)
  and
  B.o_shipdate between DATE_SUB(A.o_shipdate, interval 1 day) and
                       DATE_ADD(A.o_shipdate, interval 1 day)
```

```
+--+-----------+-----+----+----------------------+----------+-------+-...
|id|select_type|table|type|possible_keys         |key       |key_len|
+--+-----------+-----+----+----------------------+----------+-------+-...
|1 |SIMPLE     |A    |ref |i_o_order_clerk_date  |i_o_clerk |16     |
|1 |SIMPLE     |B    |ALL |i_o_orderdate,o_shipDATE|NULL    |NULL   |
+--+-----------+-----+----+----------------------+----------+-------+-...
```

```
..-+-----+-------+-----------------------------------------------+
   |ref  |rows   |Extra                                          |
..-+-----+-------+-----------------------------------------------+
   |const|1466   |Using index condition                          |
   |NULL |1499649|Range checked for each record (index map: 0x22)|
..-+-----+-------+-----------------------------------------------+
```

Range checked for each record (2)

```
ANALYZE: {
  "query_block": {
    "select_id": 1,
    "r_loops": 1,
    "r_total_time_ms": 5769,
    "table": {
      "table_name": "A",
      "access_type": "ref",
      "possible_keys": ["i_o_order_clerk_date"],
      "key": "i_o_order_clerk_date",
      "key_length": "16",
      "used_key_parts": ["o_clerk"],
      "ref": ["const"],
      "r_loops": 1,
      "rows": 1466,
      "r_rows": 1467,
      "r_total_time_ms": 3.6642,
      "filtered": 100,
      "r_filtered": 100,
      "index_condition": "(A.o_clerk =
                          'Clerk#00001')"
},
```

```
"range-checked-for-each-record": {
  "keys": ["i_o_orderdate", "o_shipDATE"],
  "r_keys": {
    "full_scan": 0,
    "index_merge": 0,
    "range": {
      "i_o_orderdate": 1467,
      "o_shipDATE": 0
    }
  },
  "table": {
    "table_name": "B",
    "access_type": "ALL",
    "possible_keys": ["i_o_orderdate", "o_shipDATE
    "r_loops": 1467,
    "rows": 1499649,
    "r_rows": 1871.2,
    "r_total_time_ms": 3649.9,
    "filtered": 100,
    "r_filtered": 100
  }
}
    }
}.
```

- **r_keys** shows what keys were useful
  - Not useful? IGNORE INDEX

ANALYZE FORMAT=JSON is true EXPLAIN

- EXPLAIN lies!

  Bug#69795: EXPLAIN FORMAT=JSON doesn't show Using filesort for UNION

  Bug#74462: EXPLAIN FORMAT=JSON produces ordering_operation when no ordering ...

  Bug#74661: EXPLAIN FORMAT=JSON says two temptables are used, execution shows just one

  Bug#74744: EXPLAIN FORMAT=JSON produces duplicates_removal where there is none

  Bug#76679: EXPLAIN incorrectly shows Distinct for tables using join buffer

- Scared yet?
- Both Oracle and MariaDB are working to fix this
- ANALYZE FORMAT=JSON
  - Is already available
  - Cannot lie, by design.

ANALYZE FORMAT=JSON data

- Query plan node
  - **r_loops** – number of executions
  - **r_total_time_ms** – total time spent
  - **r_rows** – number of rows read
- Conditions
  - **r_filtered -** %rows left after conditions
- Join buffer, filesort,etc
  - **r_buffer_size** – Buffer size used
- filesort
  - **r_limit**, **r_output_rows**
  - **r_priority_queue_used**
- Subquery cache
  - **r_hit_ratio**
- Range checked for each record
  - **r_indexes**
- ...

Summary

- MariaDB 10.1 GA
- New statements
  - ANALYZE <statement>
  - ANALYZE FORMAT=JSON <statement>
- Gives EXPLAIN + execution statistics
  - Allows to spot query optimization problems.

Thanks!