ORACLE

EXPLAIN - Revisited

Øystein Grøvlen
Consulting Member of Technical Staff
MySQL Heatwave
February 2, 2024

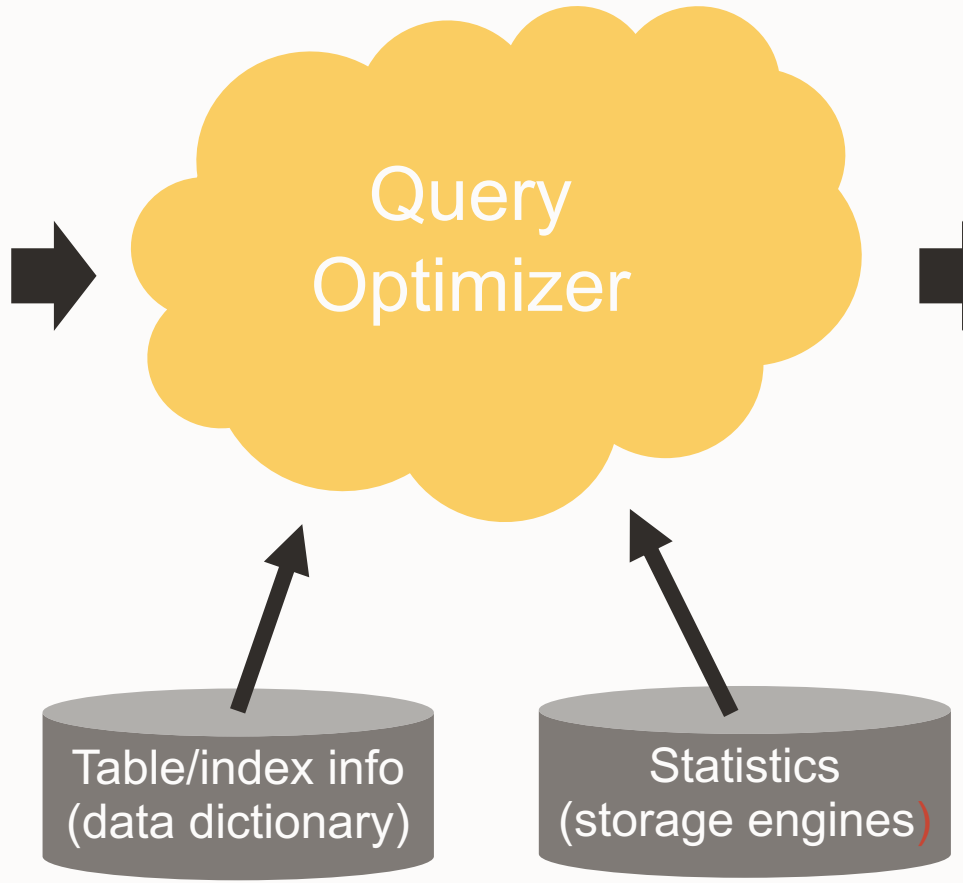
Outline

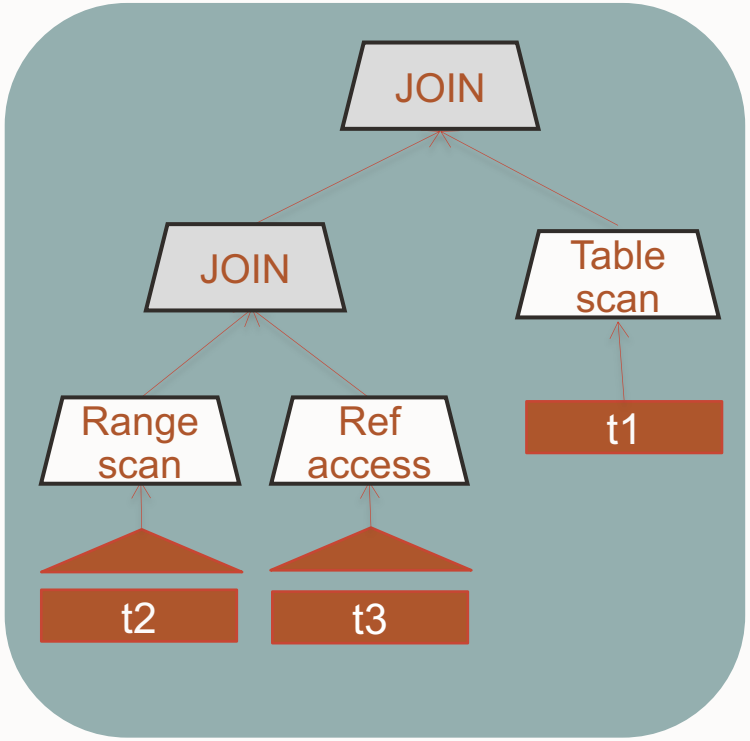
1. EXPLAIN in MySQL 8.0 and before
2. Recent additions to EXPLAIN (MySQL 8.1 - 8.3)
3. EXPLAIN ANALYZE
4. Optimizer Trace

EXPLAIN in MySQL 8.0 and before

MySQL Query Optimizer

```
SELECT a, b
FROM t1, t2, t3
WHERE t1.a = t2.b
    AND t2.b = t3.c
    AND t2.d > 20
    AND t2.d < 30;
```





EXPLAIN

Understand the query plan

```
EXPLAIN SELECT * FROM t1 JOIN t2 ON t1.a = t2.a WHERE b > 10 AND c > 10;
```

id	select_type	table	partitions	type	possible keys	key	key len	ref	rows	filtered	Extra
1	SIMPLE	t1	NULL	range	PRIMARY, idx1	idx1	4	NULL	12	33.33	Using index condition
2	SIMPLE	t2	NULL	ref	idx2	idx2	4	t1.a	1	100.00	NULL

Explain for a running query:

```
EXPLAIN FOR CONNECTION connection_id …
```

Structured EXPLAIN

JSON format

```
EXPLAIN FORMAT=JSON SELECT …
```

Contains more information:

- Used index parts
- Pushed index conditions
- Cost estimates
- Data estimates

```
EXPLAIN FORMAT=JSON
SELECT * FROM t1 WHERE b > 10 AND c > 10;
EXPLAIN
{
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "17.81"
    },
    "table": {
      "table_name": "t1",
      "access_type": "range",
      "possible_keys": [
        "idx1"
      ],
      "key": "idx1",
      "used_key_parts": [
        "b"
      ],
      "key_length": "4",
      "rows_examined_per_scan": 12,
      "rows_produced_per_join": 3,
      "filtered": "33.33",
      "index_condition": "(`test`.`t1`.`b` > 10)",
      "cost_info": {
        "read_cost": "17.01",
        "eval_cost": "0.80",
        "prefix_cost": "17.81",
        "data_read_per_join": "63"
      },
      .........
      "attached_condition": "(`test`.`t1`.`c` > 10)"
    }
  }
}
```

Structured EXPLAIN

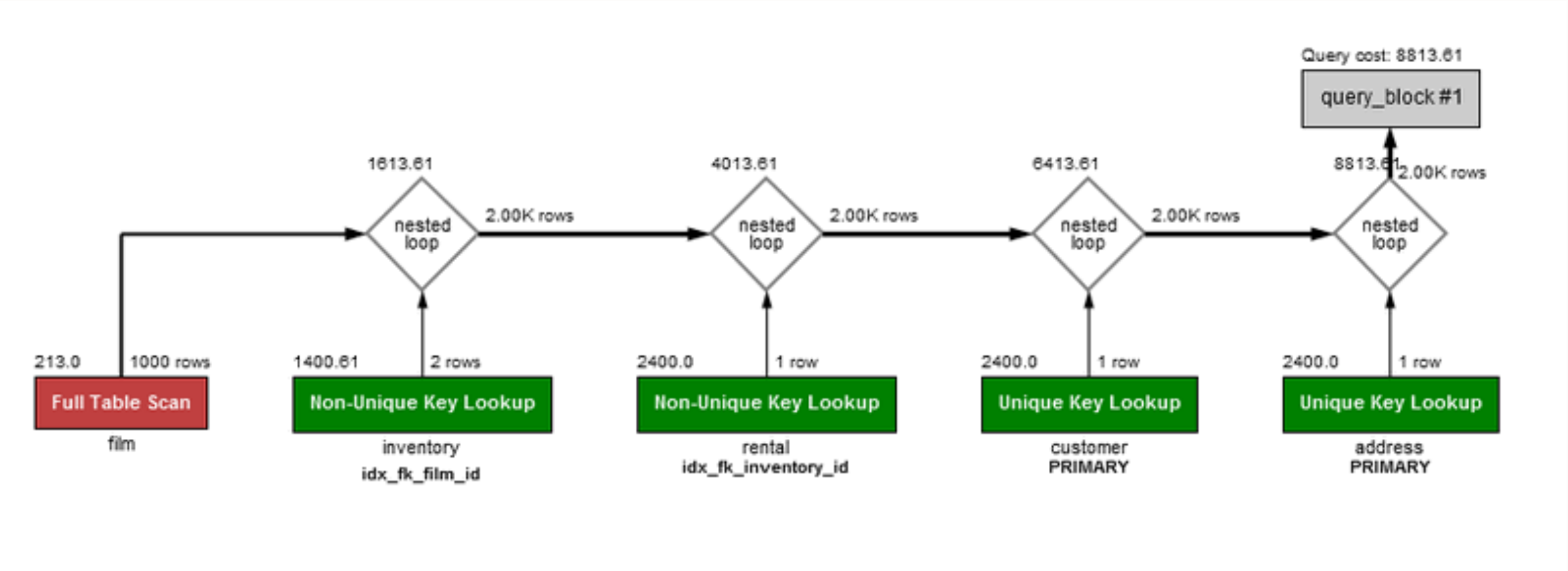
Assigning Conditions to Tables

```
EXPLAIN FORMAT=JSON SELECT * FROM t1 JOIN t2 ON t1.a=t2.a
WHERE t2.a=9 AND (NOT (t1.a > 10 OR t2.b >3) OR (t1.b=t2.b+7 AND t2.b = 5));
```

EXPLAIN


```
{
 "query_block": {
  "select_id": 1,
  "nested_loop": [
   {
    "table": {
     "table_name": "t1",
     "access_type": "ALL",
     "rows": 10,
     "filtered": 100,
     "attached_condition": "(t1.a = 9)"
    } /* table */
   },
   {
    "table": {
     "table_name": "t2",
     "access_type": "ALL",
     "rows": 10,
     "filtered": 100,
     "using_join_buffer": "Block Nested Loop",
     "attached_condition": "((t2.a = 9) and ((t2.b <= 3) or ((t2.b = 5) and (t1.b = 12))))"
    } /* table */
   }
  ] /* nested_loop */
 } /* query_block */
}
```

Visual EXPLAIN (MySQL Workbench)



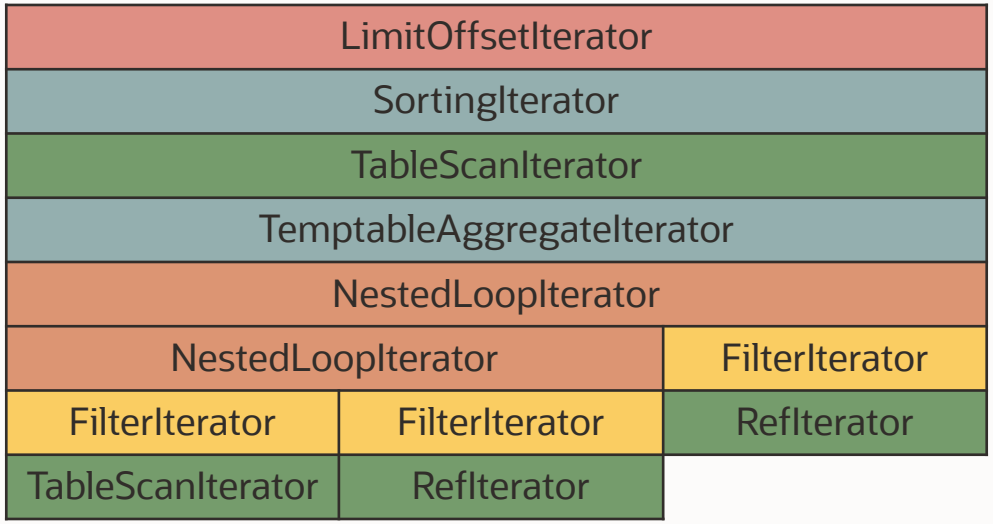
Old MySQL Query Executor vs Iterator Executor

- Old executor (MySQL 5.7)
  - Focus on nested-loop joins
  - Hard to Extend
  - Code for one operation spread out
  - Different Interfaces for each operation
  - Hard-coded combinations of operations

JOIN_TAB

table_name: t1 access_type: ALL condition: …	table_name: t2 access_type: ref key: idx1 condition: …	table_name: t3 access_type: ref key: idx2 condition: … grouping: … filesort: … limit: …

- Iterator executor (MySQL 8.0):
  - Modular
  - Easy to extend
  - One iterator per operation
  - Same interface for all iterators
  - All operations could be connected



EXPLAIN FORMAT = TREE

```
-> Limit: 10 row(s)
    -> Sort: revenue DESC, orders.o_orderDATE, limit input to 10 row(s) per chunk
        -> Table scan on <temporary>
            -> Aggregate using temporary table
                -> Nested loop inner join  (cost=314857 rows=442882)
                    -> Nested loop inner join  (cost=170434 rows=221396)
                        -> Filter: (customer.c_mktsegment = 'AUTOMOBILE')  (cost=15457 rows=29518)
                            -> Table scan on customer  (cost=15457 rows=150000)
                        -> Filter: (orders.o_orderDATE < DATE'1995-03-24')  (cost=3.75 rows=7.5)
                            -> Index lookup on orders using i_o_custkey (o_custkey=customer.c_custkey)  (cost=3.75 rows=15)
                    -> Filter: (lineitem.l_shipDATE > DATE'1995-03-24')  (cost=0.252 rows=2)
                        -> Index lookup on lineitem using PRIMARY (l_orderkey=orders.o_orderkey)  (cost=0.252 rows=4)
```


EXPLAIN ANALYZE

```
-> Limit: 10 row(s)  (actual time=12951..12951 rows=10 loops=1)
    -> Sort: revenue DESC, orders.o_orderDATE, limit input to 10 row(s) per chunk  (actual time=12951..12951 rows=10 loops=1)
        -> Table scan on <temporary>  (actual time=12946..12948 rows=11437 loops=1)
            -> Aggregate using temporary table  (actual time=12946..12946 rows=11437 loops=1)
                -> Nested loop inner join  (cost=314857 rows=442882) (actual time=22.8..12857 rows=30124 loops=1)
                    -> Nested loop inner join  (cost=170434 rows=221396) (actual time=20.9..4419 rows=145762 loops=1)
                        -> Filter: (customer.c_mktsegment = 'AUTOMOBILE')  (cost=15457 rows=29518) (actual time=5.45..329
rows=29752 loops=1)
                            -> Table scan on customer  (cost=15457 rows=150000) (actual time=5.44..296 rows=150000 loops=1)
                        -> Filter: (orders.o_orderDATE < DATE'1995-03-24')  (cost=3.75 rows=7.5) (actual time=0.118..0.137
rows=4.9 loops=29752)
                            -> Index lookup on orders using i_o_custkey (o_custkey=customer.c_custkey)  (cost=3.75 rows=15)
(actual time=0.118..0.135 rows=10 loops=29752)
                    -> Filter: (lineitem.l_shipDATE > DATE'1995-03-24')  (cost=0.252 rows=2) (actual time=0.0575..0.0576
rows=0.207 loops=145762)
                        -> Index lookup on lineitem using PRIMARY (l_orderkey=orders.o_orderkey)  (cost=0.252 rows=4) (actual
time=0.056..0.0571 rows=4 loops=145762)
```


Recent additions to EXPLAIN

MySQL 8.1 - 8.3

New version of EXPLAIN FORMAT=JSON

- JSON structure should reflect iterator executor
- New session variable: explain_json_format_version
  - Old version: **`set explain_json_format_version = 1`**
  - New version: **`set explain_json_format_version = 2`**

New version of EXPLAIN FORMAT=JSON

Comparison

`explain_json_format_version = 1`

`explain_json_format_version = 2`

Choose default EXPLAIN format

Session variable: **explain_format**

SET explain_format = 'TRADITIONAL';

```
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | t1    | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    8 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
```

SET explain_format = 'TREE';

```
-> Table scan on t1  (cost=1.05 rows=8)
```

SET explain_format = 'JSON';

```
{
  "query": "/* select#1 */ select `test`.`t1`.`a` AS `a`,`test`.`t1`.`b` AS `b` from `test`.`t1`",
  "operation": "Table scan on t1",
  "table_name": "t1",
  "access_type": "table",
  "schema_name": "test",
  "used_columns": [ … ],
  "estimated_rows": 8.0,
  "estimated_total_cost": 1.05
}
```


EXPLAIN in other schema

```
mysql> SELECT schema_name, query_sample_text
    -> FROM performance_schema.events_statements_summary_by_digest
    -> ORDER BY sum_timer_wait DESC LIMIT 3;
+-------------+-----------------------------------------------------------------------------------------
| schema_name | query_sample_text
+-------------+-----------------------------------------------------------------------------------------
| airportdb   | select count(*) from passengerdetails where NOT validate_email(emailaddress)
| airportdb   | select emailaddress  from passengerdetails where country = 'Canada' and NOT validate_email(emailaddr
| dbt3_sf1    | select l_orderkey, sum(l_extendedprice * (1 - l_discount)) as revenue, o_orderdate, o_shippriority
+-------------+-----------------------------------------------------------------------------------------

mysql> EXPLAIN FOR SCHEMA airportdb
    -> select count(*) from passengerdetails where NOT validate_email(emailaddress);
```


EXPLAIN into user-defined variable

```
EXPLAIN FORMAT=JSON INTO @plan SELECT …
```

Does the query have a limit clause?

```
SELECT JSON_SEARCH(@plan, 'one', 'limit', NULL, '$**.access_type') IS NOT NULL;
```

Show the access of the `orders` table:

```
SELECT JSON_EXTRACT(@plan,
       REPLACE(JSON_UNQUOTE(
       JSON_SEARCH(@plan, 'one', 'orders', NULL, '$**.table_name')),
       'table_name', 'operation'));
```

```
"Index range scan on orders using i_o_orderdate over ('1993-11-01' <= o_orderDATE <
'1994-02-01'), with index condition: ((orders.o_orderDATE >= DATE'1993-11-01') and
(orders.o_orderDATE < <cache>(('1993-11-01' + interval '3' month))))"
```

EXPLAIN ANALYZE

TPC-H Q20 (Potential Part Promotion Query)

```
SELECT s_name, s_address
FROM supplier, nation
WHERE s_suppkey IN (
      SELECT ps_suppkey
      FROM partsupp
      WHERE ps_partkey IN ( SELECT p_partkey FROM part WHERE p_name LIKE 'dodger%' )
        AND ps_availqty > (
            SELECT 0.5 * SUM(l_quantity)
            FROM lineitem
            WHERE l_partkey = ps_partkey AND l_suppkey = ps_suppkey
              AND l_shipdate >= '1994-01-01’
              AND l_shipdate < DATE_ADD( '1994-01-01', INTERVAL '1' YEAR)
        )
  )
  AND s_nationkey = n_nationkey AND n_name = 'INDIA’
ORDER BY s_name;
```

EXPLAIN ANALYZE

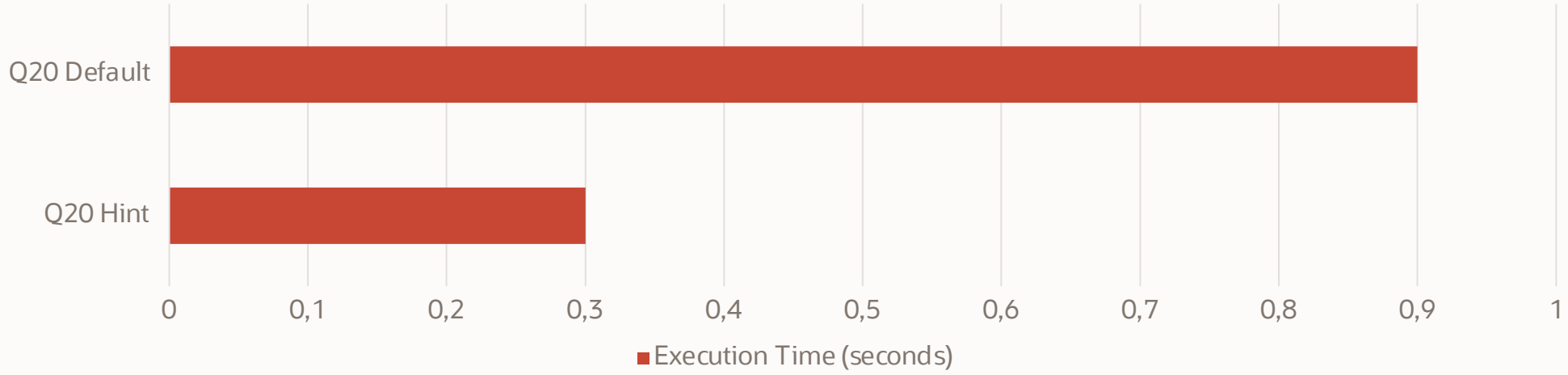
TPCH Q20 (Potential Part Promotion Query)

```
 -> Sort: supplier.s_name  (actual time=945..945 rows=164 loops=1)
    -> Stream results  (cost=8498 rows=3555) (actual time=0.905..945 rows=164 loops=1)
        -> Nested loop semijoin  (cost=8498 rows=3555) (actual time=0.901..944 rows=164 loops=1)
            -> Nested loop inner join  (cost=143 rows=400) (actual time=0.33..1.1 rows=415 loops=1)
                -> Filter: (nation.n_name = 'INDIA')  (cost=2.75 rows=1) (actual time=0.037..0.0466 rows=1 loops=1)
                    -> Table scan on nation  (cost=2.75 rows=25) (actual time=0.0283..0.037 rows=25 loops=1)
                -> Index lookup on supplier using i_s_nationkey (s_nationkey=nation.n_nationkey)  (cost=140 rows=400) (actual
time=0.291..1 rows=415 loops=1)
            -> Nested loop inner join  (cost=8028 rows=8.89) (actual time=2.27..2.27 rows=0.395 loops=415)
                -> Filter: (partsupp.ps_availqty > (select #4))  (cost=20 rows=80) (actual time=0.261..2.11 rows=42.9
loops=415)
                    -> Index lookup on partsupp using i_ps_suppkey (ps_suppkey=supplier.s_suppkey)  (cost=20 rows=80) (actual
time=0.212..0.22 rows=63.5 loops=415)
                    -> Select #4 (subquery in condition; dependent)
                        -> Aggregate: sum(lineitem.l_quantity)  (cost=2.33 rows=1) (actual time=0.029..0.029 rows=1
loops=26332)
                            -> Filter: ((lineitem.l_shipDATE >= DATE'1994-01-01') and (lineitem.l_shipDATE < <cache>(('1994-
01-01' + interval '1' year))))  (cost=2.1 rows=2.28) (actual time=0.0263..0.0284 rows=1.13 loops=26332)
                                -> Index lookup on lineitem using i_l_partkey_suppkey (l_partkey=partsupp.ps_partkey,
l_suppkey=partsupp.ps_suppkey)  (cost=2.1 rows=7.51) (actual time=0.0248..0.0275 rows=7.51 loops=26332)
                -> Filter: (part.p_name like 'dodger%')  (cost=2.22 rows=0.111) (actual time=0.00361..0.00361 rows=0.00922
loops=17795)
                    -> Single-row index lookup on part using PRIMARY (p_partkey=partsupp.ps_partkey)  (cost=2.22 rows=1)
(actual time=0.0033..0.00333 rows=1 loops=17795)
```

Optimizer hint

Change join order

```
SELECT /*+ JOIN_ORDER(part, partsupp) */ s_name, s_address
FROM supplier, nation
WHERE s_suppkey IN (
      SELECT ps_suppkey
      FROM partsupp
      WHERE ps_partkey IN ( SELECT p_partkey FROM part WHERE p_name LIKE 'dodger%' )
        AND …
```



Optimizer Trace

Optimizer Trace
How to generate it

- EXPLAIN shows the selected plan
- Optimizer trace shows WHY the plan was selected

```
SET optimizer_trace= "enabled=on";
SELECT * FROM t1, t2 WHERE f1=1 AND f1=f2 AND f2>0;
SELECT trace FROM information_schema.optimizer_trace
INTO OUTFILE filename LINES TERMINATED BY ”;
SET optimizer_trace="enabled=off"
```

QUERY	SELECT * FROM t1, t2 WHERE f1=1 AND f1=f2 AND f2>0;
TRACE	{ "steps": [{ "join_preparation": { "select#": 1,... } ... } ...] }
MISSING_BYTES_BEYOND_MAX_MEM_SIZE	0
INSUFFICIENT_PRIVILEGES	0

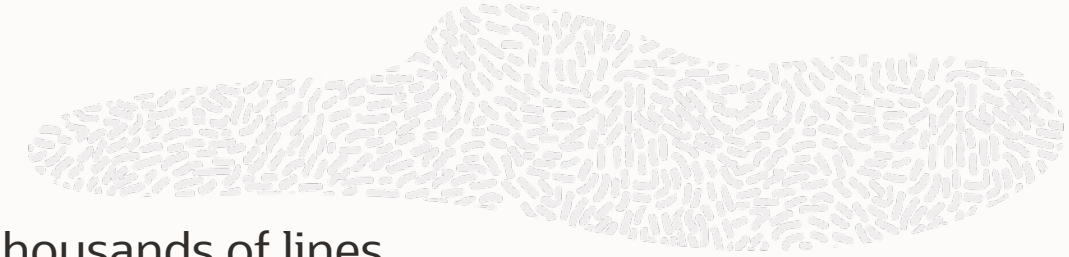
Optimizer Trace
Example


```
{
  "steps": [
    {
      "join_preparation": {
        "select#": 1,
        "steps": [
          {
            "expanded_query": "/* select#1 */ select `t1`.`f1` AS `f1`,`t2`.`f2` AS `f2` from `t1` join `t2` where ((`t1`.`f1` = 1) and (`t1`.`f1` = `t2`.`f2`)
and (`t2`.`f2` > 0))"
          }
        ]
      }
    },
    {
      "join_optimization": {
        "select#": 1,
        "steps": [
          {
            "condition_processing": {
              "condition": "WHERE",
              "original_condition": "((`t1`.`f1` = 1) and (`t1`.`f1` = `t2`.`f2`) and (`t2`.`f2` > 0))",
              "steps": [
                {
...
```

Join Optimizer

Optimizer trace

Condensed Trace For the Join Optimizer



With many tables, the trace for join optimization may be thousands of lines

- The trace for DBT3 Q8 (8 tables) is 16000 lines
- Beware: Size for optimizer trace is limited by session variable **`optimizer_trace_max_mem_size`**
  - Default MySQL 5.7: 16 kB
  - Default MySQL 8.0: 1 MB
- If `information_schema.optimizer_trace.missing_bytes_beyond_max_mem_size > 0`, increase **`optimizer_trace_max_mem_size`**

joinopttrace.js

- A script to present the trace in a condensed form
- Available at https://github.com/ogrovlen/opttrace
- Usage

  `node joinopttrace.js` *tracefile*
- Please:
  - Test it out
  - Suggest/submit improvements
  - Report issues

joinopttrace

Example output (DBT3-Q3)

```
Table AccessType:IndexName Rows/Cost TotalRows/TotalCost
------------------------------------------------------
`customer` scan 2367/244.2 478.9/244.2
  `lineitem` scan 2495/119491 1190000/119735
    `orders` eq_ref:PRIMARY 1/120108 59742/239843 *** NEW BEST PLAN ***
  `orders` scan 2502/101621 478.9/101865
    `lineitem` ref:PRIMARY 1.0155/168.35 486.3/102033 *** NEW BEST PLAN ***
`lineitem` scan 2495/255.5 2495/255.5
  `customer` scan 2367/119716 1190000/119972 PRUNED(cost)
  `orders` eq_ref:PRIMARY 1/873.25 2115.1/1128.8
    `customer` eq_ref:PRIMARY 1/740.27 427.92/1869 *** NEW BEST PLAN ***
`orders` scan 2502/255.7 2121/255.7
  `customer` eq_ref:PRIMARY 1/742.35 429.12/998.05
    `lineitem` ref:PRIMARY 1.0155/150.86 435.76/1148.9 *** NEW BEST PLAN ***
  `lineitem` ref:PRIMARY 1.0155/745.63 2153.8/1001.3 PRUNED(heuristic)
```

Conclusion

Recent Additions to EXPLAIN feature

- `SET explain_json_format_version`
- `SET explain_format`
- `EXPLAIN FOR SCHEMA`
- `EXPLAIN FORMAT=JSON INTO @var`

Thank you

Øystein Grøvlen
oystein.grovlen@oracle.com


ORACLE

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.