Subqueries to the people:
MariaDB making the impossible possible and the slow fast

Igor Babaev  
Sergey Petrunya  
Timour Katchaounov

igor@montyprogram.com  
psergey@montyprogram.com  
timour@montyprogram.com
What is the talk about

- Summarize all subquery improvements in MariaDB 5.3

- Use example queries (DBT3) to show
  - Typical performance improvements
  - Explain optimizer switches

- No complex algorithms/diagrams
What's possible and what's faster

• What's possible (in practice)
  • Efficiently execute any subquery
  • Consistent subquery performance
  • Run EXPLAIN with subqueries
  • Backwards compatibility and fine-grained performance control via switches

• What's faster
  • Save a lot of development time
  • Easier migration from other DBMS
  • Derived tables
  • [NOT IN] Subqueries
  • And any other subquery
• DBT3 scale 10
• 60 M LINEITEM rows
• 29 GB InnoDB database
• Extended schema – few more
  • indexes
  • columns
• Simplified queries
Subquery strategies

• Flattenning into JOINs (semijoin):
  • Duplicate elimination, Pull out, Loose scan, First match, Materialization-scan

• Materialization for non-correlated subqueries
  • Subqueries with NOT, GROUP BY, ORDER BY, in the SELECT clause
  • efficient NULL-aware materialized execution

• Derived tables
  • Derived tables are merged as views
  • Late materialization of derived tables and views
  • Dynamic indexing

• Subquery caching

• Cost-based/manual choice of most strategies
• Fast EXPLAIN with subqueries
Subquery flattening

• Basic idea
  • Transform subquery predicates into JOINs
    • SELECT ...
      FROM outer_table
      WHERE outer_col IN (SELECT inner_col
                           FROM inner_table
                           WHERE subq_where)
      AND outer_where;

    SELECT ...
    FROM outer_table [SEMI]JOIN inner_table
    WHERE subq_where AND outer_where;

• No:
  • GROUP
  • Aggregation
  • Disjunction in the WHERE
Query: orders from customers with negative balance:

```sql
SELECT * FROM orders
WHERE o_custkey IN
  (SELECT c_custkey FROM customer
   WHERE c_acctbal < -500);
```

MariaDB 5.2 (any MySQL): 45 sec (slow)

```
+-----------------+--------+-------+----------+---------+
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>orders</td>
<td>index</td>
<td>i_o_custkey</td>
<td>NULL</td>
<td>1493631</td>
<td>Using where; Using index</td>
</tr>
<tr>
<td>2</td>
<td>SUBQUERY</td>
<td>customer</td>
<td>range</td>
<td>c_acctbal</td>
<td>NULL</td>
<td>10536</td>
<td>Using where; Using index</td>
</tr>
</tbody>
</table>
```

MariaDB 5.3: 0.43 sec (faster ~ 100x)

```
+-----------------+--------+-------+----------+---------+
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>customer</td>
<td>range</td>
<td>c_acctbal</td>
<td>NULL</td>
<td>10536</td>
<td>Using where; Using index</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>orders</td>
<td>ref</td>
<td>i_o_custkey</td>
<td>dbt3sf1.customer.c_custkey</td>
<td>7</td>
<td>Using index</td>
</tr>
</tbody>
</table>
```

---

7 / 32 11:35:28
Pullout strategy (2)

• General idea
  Based on unique indexes, rewrite subqueries into equivalent joins

• Optimizer switches
  • optimizer_switch='semijoin=on';
Query: find customers with top balance in their nations:

```
SELECT c_name, c_address
FROM customer
WHERE c_acctbal IN (SELECT max(c_acctbal)
                      FROM customer GROUP BY c_nationkey);
```

MariaDB 5.2 (any MySQL): > 1.5 hours (impossible)

MariaDB 5.3: 3.2 sec (faster ~ INF)
General idea:
- The subquery materializes into a small dataset
- The materialized subquery drives index-based access to the outer table

Relevant optimizer switches:
- semijoin= on,
- materialization=on
**Duplicate elimination (1)**

**Query:** find customers who have orders exceeding their balance:

```sql
SELECT * FROM customer
WHERE c_custkey IN
  (SELECT o_custkey FROM orders
   WHERE o_totalprice > c_acctbal and o_orderdate between '1997-01-01' and '1997-01-02')
```

**MariaDB 5.2 (any MySQL): 1:28 min (slow)**

```
+----------------+--------+--------+-------+-------+-------+-----------+
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>PRIMARY</td>
<td>customerindex</td>
<td>c_acctbal</td>
<td>NULL</td>
<td>150081</td>
<td>Using where; Using index</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DEPENDENT SUBQUERY</td>
<td>orders</td>
<td>index_subquery</td>
<td>i_o_custkey</td>
<td>func</td>
<td>7</td>
<td>Using where</td>
</tr>
</tbody>
</table>
+----------------+--------+--------+-------+-------+-------+-----------+
```

**MariaDB 5.3: 36 sec (faster ~ 2.4x)**

```
+----------------+--------+--------+-------+-------+-------+-----------+
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>orders</td>
<td>range</td>
<td>i_o_orderdate</td>
<td>NULL</td>
<td>5184</td>
<td>Using index condition; Using where;</td>
</tr>
<tr>
<td>2</td>
<td>PRIMARY</td>
<td>customer</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>o_custkey</td>
<td>1</td>
<td>Using where; End temporary</td>
</tr>
<tr>
<td>3</td>
<td>PRIMARY</td>
<td>customer</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>o_custkey</td>
<td>1</td>
<td>Using where; End temporary</td>
</tr>
</tbody>
</table>
+----------------+--------+--------+-------+-------+-------+-----------+
```
Duplicate elimination (2)

• General idea:
  • Convert to JOIN,
  • Remove duplicates (since it is semi-join)

• Relevant optimizer switches:
  • optimizer_switch='semijoin=on';
  • No other strategy specific switches,
  • it is the “catch-all” strategy
First match

• Same as original IN-TO-EXISTS, but:
  • Cost-based choice
  • First match compared to other strategies

• General idea:
  • Cheap when subquery re-execution cost is very low
  • Typically REF or EQ_REF access types
Subquery materialization

- Non-correlated subqueries, with either/or:
  - GROUP / ORDER BY,
  - NOT IN
  - Aggregation
  - Disjunctive (OR-ed) WHERE clause,
  - Subquery in the SELECT clause
Materialization lookup (1)

Query:

```
SELECT * FROM part
WHERE p_partkey IN
  (SELECT l_partkey FROM lineitem
   WHERE l_shipdate between '1997-01-01' and '1997-02-01')
ORDER BY p_retailprice DESC LIMIT 10;
```

MariaDB 5.2 (any MySQL): > 1 h (slow)

```
+-----------------+---------+---------+----------------+-------+-------+-------|
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>part</td>
<td>ALL</td>
<td>NULL</td>
<td>NULL</td>
<td>199755</td>
<td>Using where; Using filesort</td>
</tr>
<tr>
<td>2</td>
<td>DEPENDENT SUBQUERY</td>
<td>lineitem</td>
<td>index_subquery</td>
<td>i_l_suppkey_partkey</td>
<td>func</td>
<td>14</td>
<td>Using where</td>
</tr>
</tbody>
</table>
+-----------------+---------+---------+----------------+-------+-------+-------|
```

MariaDB 5.3: 43 sec (faster ~ INF)

```
+-----------------+---------+---------+-----------------+-------+-------+-------|
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>part</td>
<td>ALL</td>
<td>NULL</td>
<td>NULL</td>
<td>199755</td>
<td>Using temporary; Using filesort</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>&lt;subquery2&gt;</td>
<td>eq_ref</td>
<td>distinct_key</td>
<td>func</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SUBQUERY</td>
<td>lineitem</td>
<td>range</td>
<td>l_shipdate_partkey</td>
<td>NULL</td>
<td>160060</td>
<td>Using where; Using index</td>
</tr>
</tbody>
</table>
+-----------------+---------+---------+-----------------+-------+-------+-------|
```
Materialization lookup (2)

• General guidelines
  • Fast when 'pre-processing' the subquery and
  • making lookups is cheaper than re-executing it every time.
  • Typically when the subquery has relatively small result set, and
  • The outer query performs many lookups

• Optimizer switches
  • semijoin=on,
  • materialization=on
Materialization with NULLs (1)

Find all customers that didn't buy from a preferred country, and from a preferred brand.

```sql
SELECT count(*)
FROM customer
WHERE (c_custkey, c_pref_nationkey_05, c_pref_brand_05) NOT IN
  (SELECT o_custkey, s_nationkey, p_brand
   FROM orders, supplier, part, lineitem
   WHERE l_orderkey = o_orderkey and
     l_suppkey = s_suppkey and
     l_partkey = p_partkey and
     p_retailprice < 1200 and
     l_shipdate >= '1996-04-01' and
     l_shipdate < '1996-04-05' and
     o_orderdate >= '1996-04-01' and
     o_orderdate < '1996-04-05');
```
Materialization with NULLs (2)

MariaDB 5.2 (any MySQL): 40 sec (slow)

<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</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>NULL</td>
<td>NULL</td>
<td>150841</td>
<td>Using where</td>
</tr>
<tr>
<td>2</td>
<td>DEPENDENT SUBQUERY</td>
<td>orders</td>
<td>ref_or_null</td>
<td>i_o_custkey</td>
<td>func</td>
<td>14</td>
<td>Using index condition; Using where</td>
</tr>
<tr>
<td>2</td>
<td>DEPENDENT SUBQUERY</td>
<td>lineitem</td>
<td>ref</td>
<td>PRIMARY</td>
<td>o_orderkey</td>
<td>1</td>
<td>Using where</td>
</tr>
<tr>
<td>2</td>
<td>DEPENDENT SUBQUERY</td>
<td>supplier</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>l_suppkey</td>
<td>1</td>
<td>Using where</td>
</tr>
<tr>
<td>2</td>
<td>DEPENDENT SUBQUERY</td>
<td>part</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>l_partkey</td>
<td>1</td>
<td>Using where</td>
</tr>
</tbody>
</table>

MariaDB 5.3: 2 sec (faster ~ 20x)

<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</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>NULL</td>
<td>NULL</td>
<td>150841</td>
<td>Using where</td>
</tr>
<tr>
<td>2</td>
<td>SUBQUERY</td>
<td>orders</td>
<td>range</td>
<td>i_o_orderdate</td>
<td>NULL</td>
<td>4680</td>
<td>Using index condition; Using MRR</td>
</tr>
<tr>
<td>2</td>
<td>SUBQUERY</td>
<td>lineitem</td>
<td>ref</td>
<td>PRIMARY</td>
<td>o_orderkey</td>
<td>1</td>
<td>Using where</td>
</tr>
<tr>
<td>2</td>
<td>SUBQUERY</td>
<td>supplier</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>l_suppkey</td>
<td>1</td>
<td>Using where</td>
</tr>
<tr>
<td>2</td>
<td>SUBQUERY</td>
<td>part</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>l_partkey</td>
<td>1</td>
<td>Using where</td>
</tr>
</tbody>
</table>
Materialization with NULLs (3)

• General idea
  • NULL acts as a “match-any” operator
  • If possible perform a lookup in the materialized subquery
  • Otherwise perform partial matching
    • Specialized per-column index merge (big data), or
    • Table scan + NULL test per column

• Optimizer switches
  • materialization=on,
  • partial_match_rowid_merge=on
  • partial_match_table_scan=on
```sql
SELECT max(t.l_extendedprice)
FROM orders,
     (SELECT * FROM lineitem
      WHERE l_shipdate between '1992-07-01' and '1992-08-31') AS t
WHERE t.l_orderkey=o_orderkey AND
     o_orderdate between '1992-05-01' AND '1992-08-31';
```

**MariaDB 5.2 (any MySQL): 6:37 min (slow)**

```
+----------+----------+--------+----------+----------+----------+        +
| id       | select_type | table      | type | key | ref | rows | Extra          |
+----------+----------+--------+----------+--------+----------+----------+        +
| 1        | PRIMARY   | <derived2> | ALL | NULL | NULL | 1543786 |        |
| 1        | PRIMARY   | orders | eq_ref | PRIMARY | t.l_orderkey | 1 | Using where |
| 2        | DERIVED   | lineitem | range | i_l_shipdate | NULL | 2420940 | Using where |
```

**EXPLAIN ~6–8 min**

**MariaDB 5.3: 4:20 min (faster ~ 1.5x)**

```
+----------+----------+----------+--------+----------+----------+        +
| id       | select_type | table      | type | key | ref | rows | Extra |
+----------+----------+--------+----------+--------+----------+----------+        +
| 1        | SIMPLE   | orders | range | i_o_orderdate | NULL | 353592 | Using where; Using index |
| 1        | SIMPLE   | lineitem | ref | PRIMARY | o_orderkey | 2 | Using where |
```

**EXPLAIN 0 sec**
SELECT MAX(o_totalprice)
FROM
(SELECT * FROM orders
   WHERE o_orderdate between '1992-04-01' and '1992-06-30') AS r,
(SELECT l_orderkey, MAX(l_shipdate) AS m FROM lineitem
   GROUP BY l_orderkey) AS t
WHERE t.l_orderkey=r.o_orderkey AND
   r.o_orderdate > m - interval 3 month AND
   r.o_orderstatus='F' AND r.o_totalprice > 400000;
## Derived tables – dynamic index (3)

### MariaDB 5.2 (any MySQL): 4:14 min (slow)

<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>&lt;derived2&gt;</td>
<td>ALL</td>
<td>NULL</td>
<td>NULL</td>
<td>1543786</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>orders</td>
<td>eg_ref</td>
<td>PRIMARY</td>
<td>t.l_orderkey</td>
<td>1</td>
<td>Using where</td>
</tr>
<tr>
<td>2</td>
<td>DERIVED</td>
<td>lineitem</td>
<td>range</td>
<td>i_l_shipdate</td>
<td>NULL</td>
<td>2420940</td>
<td>Using where</td>
</tr>
</tbody>
</table>

### MariaDB 5.3: 44.9 sec (faster ~ 5.6x)

<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>key</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SIMPLE</td>
<td>orders</td>
<td>range</td>
<td>i_o_orderdate</td>
<td>NULL</td>
<td>353592</td>
<td>Using where; Using index</td>
</tr>
<tr>
<td>1</td>
<td>SIMPLE</td>
<td>lineitem</td>
<td>ref</td>
<td>PRIMARY</td>
<td>o_orderkey</td>
<td>2</td>
<td>Using where</td>
</tr>
</tbody>
</table>
Subquery in FROM

```sql
select a, b, c
from (select ... as t1, t2
where some_cond;
```

Subquery in SELECT

```sql
select a, (select b, c from t1 ...)
from t2 where ...;
```

Single-row subquery

```sql
select a, b, c
from t1
where (select sum(d) from ...) / 
      (select count(*) from ...) > 5;
```
• Subqueries in MariaDB run 2x, 10x, 100x, Infinitely faster
• Subqueries are usually as fast or faster than manually optimized queries
• Instant EXPLAIN in all cases

• Much less machine time
• Much less development time

• Questions? Suggestions?