MySQL 8.0.18 latest updates

Hash join and EXPLAIN ANALYZE

Norvald H. Ryeng
Software Development Senior Manager
MySQL Optimizer Team
October 1, 2019
Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle’s products may change and remains at the sole discretion of Oracle Corporation.
Program agenda

1. Quick demo
2. How did we get here?
   - Phase separation
   - Volcano iterator model
3. Hash join
4. EXPLAIN ANALYZE
Quick demo
MySQL 8.0.18
CREATE TABLE t1 (a INT, b INT);
Query OK, 0 rows affected (0.0248 sec)

INSERT INTO t1 VALUES (1,1), (2,2), (3,3);
Query OK, 3 rows affected (0.0133 sec)

CREATE TABLE t2 (a INT, b INT);
Query OK, 0 rows affected (0.0247 sec)

INSERT INTO t2 VALUES (1,1), (2,2), (3,3);
Query OK, 3 rows affected (0.0074 sec)

EXPLAIN ANALYZE SELECT * FROM t1 JOIN t2 ON t1.a = t2.a;

| EXPLAIN

| -> Inner hash join (t2.a = t1.a) (cost=1.70 rows=3) (actual time=0.796..0.965 rows=3 loops=1)
  |  -> Table scan on t2 (cost=0.12 rows=3) (actual time=0.128..0.281 rows=3 loops=1)
  |  -> Hash
  |  |  -> Table scan on t1 (cost=0.55 rows=3) (actual time=0.237..0.423 rows=3 loops=1)

1 row in set (0.0036 sec)
How did we get here?

A long term investment that finally pays off
@MySQL 8 is light years away from 5.x versions. You now have:

- CTE and Recursive CTE
- Window Functions
- SKIP LOCKED, NO WAIT
- Hash Joins (Coming in 8.0.18)
- Explain Analyze giving you the Actual Plan (Coming in 8.0.18)

https://twitter.com/vlad_mihalcea/status/1173842312398614528
MySQL refactoring: Separating phases

- **Started ~10 years ago**
  Considered finished now
- **A clear separation between query processing phases**
- **Fixed a large number of bugs**
  Improved stability
- **Faster feature development**
  Fewer surprises and complications during development
MySQL refactoring: Parsing and preparing

- **Still ongoing**
  Implemented piece by piece

- **Separating parsing and resolving phases**
  Eliminate semantic actions that do too much
  Get a true bottom-up parser

- **Makes it easier to extend with new SQL syntax**
  Parsing doesn’t have unintended side effects

- **Consistent name and type resolving**
  Names resolved top-down
  Types resolved bottom-up

- **Transformations done in the prepare phase**
  Bottom-up
MySQL features made possible because we invested in refactoring

CTEs
- Recursive and non-recursive CTEs
- Traverse hierarchies
- Write more readable SQL

LATERAL
- "For-each loops"

... and many, many more!

Window functions
- Aggregation, ranking, analytics
- Sliding windows

JSON
- JSON_TABLE
- JSON window functions
MySQL refactoring: Iterator executor

- Volcano iterator model
- Possible because phases were separated
- Ongoing for ~1,5 year
- Much more modular executor
  - Common iterator interface for all operations
  - Each operation is contained within an iterator
- Able to put together plans in new ways
  - Immediate benefit: Removes temporary tables in some cases
- Join is just an iterator
  - Nested loop join is just an iterator
  - Hash join is just an iterator
  - Your favorite join method is just an iterator
Old MySQL executor vs. iterator executor

Old executor
- Nested loop focused
- Hard to extend
- Code for one operation spread out
- Different interfaces for each operation
- Combination of operations hard coded

Iterator executor
- Modular
- Easy to extend
- Each iterator encapsulates one operation
- Same interface for all iterators
- All operations can be connected
MySQL 8.0 features based on the iterator executor

- **EXPLAIN FORMAT=TREE**
  - Print the iterator tree

- **EXPLAIN ANALYZE**
  1. Insert instrumentation nodes in the tree
  2. Execute the query
  3. Print the iterator tree

- **Hash join**
  - Just another iterator type
Hash join
MySQL internals

New in 8.0.18
MySQL hash join

- Hybrid hash join
- Three execution modes
  - Everything fits in memory
  - Spill to disk (GRACE hash join)
  - Looping
- Equi-join
- xxHash64
  - Fast
  - Good distribution

```
SELECT * FROM t1 JOIN t2 ON t1.a = t2.a;
```
MySQL hash join — everything fits in memory

1) Hash one table into memory
   Smallest table
   The whole table fits in memory (if not, use next method)

2) Scan the other table and match with rows in memory
MySQL hash join — spill to disk

1) Hash the smallest table into memory until the buffer is full
   When the buffer is full, dump the rest as chunks on disk
   Produce up to 128 hash buckets (chunks) on disk

2) Scan the other table and match with rows in memory
   Write to chunks on disk at the same time
   Produce up to 128 hash buckets (chunks) on disk

3) Hash one chunk of the smallest table into memory
   Hash using a different seed than used to create disk buckets
   (If the bucket doesn't fit, see next method)

4) Scan the corresponding chunk and match with rows in memory

5) Repeat 3-4 until all chunks have been processed
MySQL hash join — looping

1) Hash a batch of the chunk into memory
   When the buffer is full, pause reading this operand
2) Scan the corresponding chunk and match with rows in memory
3) Hash the next batch into memory
   Resume from where it was paused
4) Scan the corresponding chunk and match with rows in memory
5) Repeat 3-4 until the whole chunk has been processed
MySQL hash join optimization

- Replaces BNL
- Currently optimized as BNL
  Replace with hash join after optimization
  A conservative (safe) choice
- Optimizer switch to turn on/off
  SET optimizer_switch='hash_join=on';
- Hints
  /*+ HASH_JOIN(tables or query blocks) */
  /*+ NO_HASH_JOIN(tables or query blocks) */
- Buffer size
  SET join_buffer_size=number;
MySQL hash join performance

- BNL compared to hash join
- Force BNL/hash join in DBT-3/TPC-H
  DBT-3/TPC-H without indexes
  Optimizer selects BNL
  Automatic conversion to hash join

- Hash join is much faster than BNL
- Can’t expect same improvement when indexes are available
New in 8.0.18

EXPLAIN ANALYZE

MySQL internals
MySQL EXPLAIN ANALYZE

- Wrap iterators in instrumentation nodes
- Measurements
  - Time (in ms) to first row
  - Time (in ms) to last row
  - Number of rows
  - Number of loops
- Execute the query and dump the stats
What's wrong with Q2?

```sql
SELECT s_acctbal, s_name, n_name, p_partkey, p_mfgr, s_address, s_phone, s_comment
FROM part, supplier, partsupp, nation, region
WHERE p_partkey = ps_partkey AND s_suppkey = ps_suppkey AND p_size = 4
AND p_type LIKE '%TIN' AND s_nationkey = n_nationkey
AND n_regionkey = r_regionkey AND r_name = 'AMERICA'
AND ps_supplycost = (
    SELECT min(ps_supplycost)
    FROM partsupp, supplier, nation, region
    WHERE p_partkey = ps_partkey AND s_suppkey = ps_suppkey
    AND s_nationkey = n_nationkey AND n_regionkey = r_regionkey
    AND r_name = 'AMERICA'
)
ORDER BY s_acctbal DESC, n_name, s_name, p_partkey
LIMIT 100;
```

MySQL EXPLAIN ANALYZE to the rescue!
Make your queries run faster with MySQL 8.0

- **Temporary table elimination**
  - Recursive CTEs
  - Some derived tables
  - UNION into derived tables
  - Input to sorting

- **Faster duplicate removal**
- **Hash join instead of BNL**
- **Analyze queries to find out where time is spent**
  - `EXPLAIN FORMAT=Tree`
  - `EXPLAIN ANALYZE`
  - Optimizer trace

Automatic
Feature descriptions and design details directly from the source

https://mysqlserverteam.com/
Thank you!

Norvald H. Ryeng
Software Development Senior Manager
MySQL Optimizer Team