MariaDB Optimizer

Current state, comparison with other branches, development plans

Sergei Petrunia <sergey@mariadb.com>

MariaDB Tampere Unconference

June 2018

Let’s review recent history

- MariaDB 10.2
- MariaDB 10.3
- MySQL 8.0
- MariaDB 10.4

Optimizer features in MariaDB 10.2

MariaDB 10.2 (Stable in May 2017)

- Window functions
- Common Table Expressions
  - Non-recursive
  - Recursive
- Condition pushdown into derived tables

Optimizer features in MariaDB 10.3

MariaDB 10.3 (Stable in May 2018)

- Split grouping
- Condition pushdown through window functions
- Table value constructors
- Transform [NOT] IN predicate with big list into subquery

Optimizer features in MySQL 8.0

MySQL 8.0 (Stable in May 2018, 5.7 was in Oct 2015)

- Histograms
- Common Table Expressions
  - Recursive
  - Non-recursive
- Invisible indexes
- Descending indexes
- More Oracle-style hints

Observations

- MySQL 8.0 re-implements a few big features
  - Window functions
  - Common Table Expressions
    - Recursive
    - Non-recursive
  - Histograms
- MySQL still misses some of MariaDB features
- But it also has some extra features

Let’s compare common features

MariaDB vs MySQL

Non-recursive CTEs

- Another syntax for derived tables/VIEWs
  - Optimizations for derived tables are applicable
- One exception: a CTE may be used multiple times

Non-recursive CTEs optimizations

	Merge	Condition pushdown	Lateral derived	CTE reuse
MariaDB 10.3	✔	✔	✔(10.3)	✘
MS SQL Server	✔	✔	?	✘
PostgreSQL	✘	✘	✘	✔
MySQL 8.0	✔	✘	✘	✔

- Merge and Condition Pushdown are the most important
  - MariaDB supports them, like MS SQL.
- PostgreSQL's approach is *weird*: "CTEs are optimization barriers"
- MySQL 8.0: "try merging, otherwise reuse"

Recursive CTEs

- The standard specifies how RCTE should be computed
  - Both MySQL and MariaDB follow it.
- MariaDB: also supports non-standard CTE computation
  - set standard_compliant_cte=off …
  - Allows the user to do more
- Performance/optimizations
  - Not aware of practically important performance-sensitive cases.

Window function optimizations

- Condition pushdown
- Reduce the number of sorting passes
- Streamed computation
- ORDER BY-like optimizations

Window Functions optimizations

	Reuse compatible sorts	Streamed computation	Condition pushdown	ORDER BY LIMIT-like optimizations
MariaDB 10.3	✔	~✔	✔	✘
MS SQL Server	✔	~✔	✔	✔
PostgreSQL	✔	~✔	✔	✘
MySQL 8.0	✔	~✔	✘	✘

Everyone has this since it's mandatory for identical sorts

Essential, otherwise O(N) computation becomes O(N^2)

Very nice to have for analytic queries

Sometimes used for TOP-n queries by those with "big database" background

MariaDB

Histograms

Why histograms?

- The optimizer needs data about condition selectivity
- Research papers: selectivity data is much more important than cost model
  - Confirms our experience.
- Histograms provide selectivity data
  - The optimizer needs to be able to use it

Histograms in MariaDB

- Available in MariaDB 10.0 (stable since March 2014)
  - Also called “Engine Independent statistics”
- Have been useful in the real world
  - “Make query plans better” according to the user
- Have some limitations

Histogram storage in MariaDB

```
CREATE TABLE mysql.column_stats (
   db_name varchar(64) NOT NULL,
   table_name varchar(64) NOT NULL,
   column_name varchar(64) NOT NULL,
   min_value varbinary(255) DEFAULT NULL,
   max_value varbinary(255) DEFAULT NULL,
   nulls_ratio decimal(12,4) DEFAULT NULL,
   avg_length decimal(12,4) DEFAULT NULL,
   avg_frequency decimal(12,4) DEFAULT NULL,
   hist_size tinyint unsigned,
   hist_type enum('SINGLE_PREC_HB','DOUBLE_PREC_HB'),
   histogram varbinary(255),
   PRIMARY KEY (db_name,table_name,column_name)
);
```

- min_value and max_value are stored in full
- Bucket bounds are stored as fractions between min and max
  - Compact but imprecise!

Histogram collection in MariaDB

- Do a full table scan and collect values into Unique object
- Now we know the exact **rows_in_table**
- Enumerate sorted values
  - Each (**rows_in_table / n_buckets**) there is a value that starts the next bucket.
  - First and last values are **min_val** and **max_val**

✔ Predictable

✔ Deterministic

✔ Produces exact result

✗ Requires a full table scan

✗ Unique will store entire column population on disk

✗ For varchar(N) each value takes N chars!

Histograms in MySQL 8.0

- Are stored as JSON
  - No apparent limit on size
- Two histogram types are supported
  - “singleton” (list of values + frequencies)
  - “equi-height”, with exact values for min/max bound
- Collection
  - Full table scan with Bernoulli sampling (rolls the dice for each row)
  - Uses a specified limited memory for collection

Histograms in MySQL 8.0

```
{
  "last-updated": "2015-11-04 15:19:51.000000",
  "histogram-type": "equi-height",
  "null-values": 0.1, // Fraction of NULL values

  "buckets":
  [
    [
      "bar",  // Lower inclusive value
      "foo",  // Upper inclusive value
      0.00197872866683156l, // Cumulative frequency

      10 // Number of distinct values in this bucket
    ],
    ...
  ]
}
```

Histograms in PostgreSQL

- A histogram is both
  - A list of Most-Common-Values (MCV) with frequencies
  - A height-balanced histogram of values not in MCV

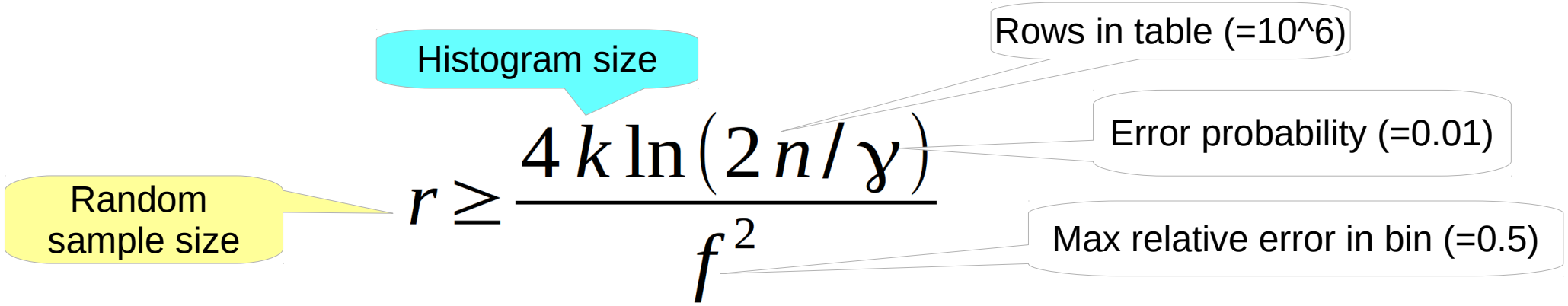
```
select * from pg_stats where tablename='pop1980';

tablename             | pop1980
attname               | firstname
null_frac             | 0
avg_width             | 7
n_distinct            | 9320
most_common_vals      | {Michael,Jennifer,Christopher,Jason,David,James,
                        Matthew,John,Joshua,Amanda}
most_common_freqs     | {0.0201067,0.0172667,0.0149067,0.0139,0.0124533,
                        0.01164,0.0109667,0.0107133,0.0106067,0.01028}
histogram_bounds      | {Aaliyah,Belinda,Christine,Elsie,Jaron,Kamia,
                        Lindsay,Natasha,Robin,Steven,Zuriel}
correlation           | 0.0066454
most_common_elems     |
```

MariaDB

Histograms are collected with sampling

- src/backend/commands/analyze.c, std_typanalyze() refers to
- **"Random Sampling for Histogram Construction: How much is enough?"**
  – Surajit Chaudhuri, Rajeev Motwani, Vivek Narasayya, ACM SIGMOD, 1998.



$$r \geq 305.82 \cdot k$$

- 100 buckets = 30,000 rows sample

MariaDB

Histogram collection in PostgreSQL

- The process: sample 30K rows from random locations in the table
  - Single pass, a skip scan forward
  - “Randomly chosen rows in randomly chosen blocks”

- Collection triggered by
  - ANALYZE command
  - Autovacuum seeing that number of modified tuples in the table exceeded a threshold

Histograms summary

- MariaDB 10.2 has histograms
  - Histogram collection is a full table scan + expensive processing
  - Histograms are very compact (more than necessary?)
- MySQL 8.0 has larger histograms
  - The optimizer is not as powerful when using them
  - Histogram collection is a full scan + less expensive processing
- PostgreSQL does genuine sampling

MariaDB 10.4

Optimizer features in MariaDB 10.4 (1)

Completed

- MDEV-12387: Push conditions into materialized IN subqueries (Galina, Igor)

In progress

- MDEV-7486: Condition Pushdown from HAVING into WHERE (Galina, Igor)
- MDEV-15253: Change the optimizer defaults to include newer features (Varun, SergeiP)
- MDEV-11953: support of brackets (parentheses) in UNION/ EXCEPT/ INTERSECT operations (Igor, Sanja)
  - Has an optimizer-related part

Optimizer features in MariaDB 10.4 (2)

GSoC 2018 projects - In progress

- MDEV-6111: Optimizer trace (Zhzhzoo Zhang + SergeiP, Varun)
  - Project is at risk due to student inactivity
- MDEV-12313: Improved Histograms (Teodor + Vicentiu)

GSoC 2017 projects

- MDEV-11107: Use table check constraints in optimizer (Igor + Galina)
  - Basic variant works
  - Unresolved issues with datatypes like date[time].

Optimizer features in MariaDB 10.4 (3)

Planned

- MDEV-16188: Use in-memory PK filters built from range index scans (“Pre-filtering” for short) (Igor, Galina)
- MDEV-11588: Extended strict mode in GROUP BY (Varun)
- MDEV-9062: ColumnStore integration: join pushdown to storage engines (Igor)

Planned 2

- MDEV-7487: Semi-join optimization for single-table UPDATE/DELETEs
  - Not allocated to anyone ATM
- A few smaller that cannot be put into a stable release

Thanks!

Discussion