Why are we excited about MySQL 8.0

Ronen Baram

Sales Consultants team leader, MySQL
Sep, 2018

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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

MySQL 5.7…

Improvements across the board!

200+ new features In total!

- Replication
- InnoDB
- Optimizer
- Security
- Performance Schema
- GIS
- Triggers
- Partitioning
- New! SYS Schema
- New! JSON
- Performance

Enabling Customer Innovation

MySQL 5.7

- 3x Better Performance
- Replication Enhancements
- JSON Support
- Improved Security

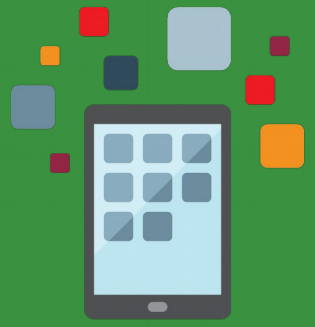
MySQL InnoDB Cluster

- MySQL Group Replication
- MySQL Router
- MySQL Shell

MySQL 8.0

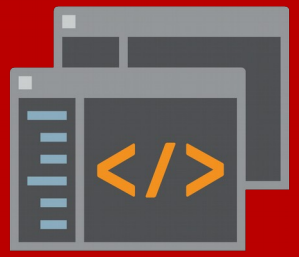
- Data Dictionary
- Roles
- Unicode
- CTEs
- Window Functions
- Security
- Replication

MySQL 8.0: Enables Modern Web Applications



Mobile Friendly

Ready for location based services. Handling Emoji and Unicode characters😃



Developer First

Hybrid data model and data access APIs for flexibility for developers

Data Driven

Optimizing services with real time data analysis

24x7 at Scale

Scalable & Stable

Better handling of high contention, improved security, and minimizing downtime

ORACLE

Transactional Data Dictionary

- Increased Reliability.
- Using InnoDB internally for data dictionary.
  - No FRM files
  - No DB.OPT files
  - No TRG files
  - No TRN files
  - No PAR files
- MySQL 8.0 default install no longer contains MyISAM tables.

Data Dictionary

Increased reliability and consistency with transactional meta data repository

Transactional Data Dictionary

Additional Benefits

- Better cross-platform experience.
  - No dependencies on filesystem semantics.
- Atomic DDL.
  - Better Replication.
  - Simplifies server edge cases.
- MDL for Foreign Keys.
- Flexible Metadata API.
  - Easier path to adding new features.

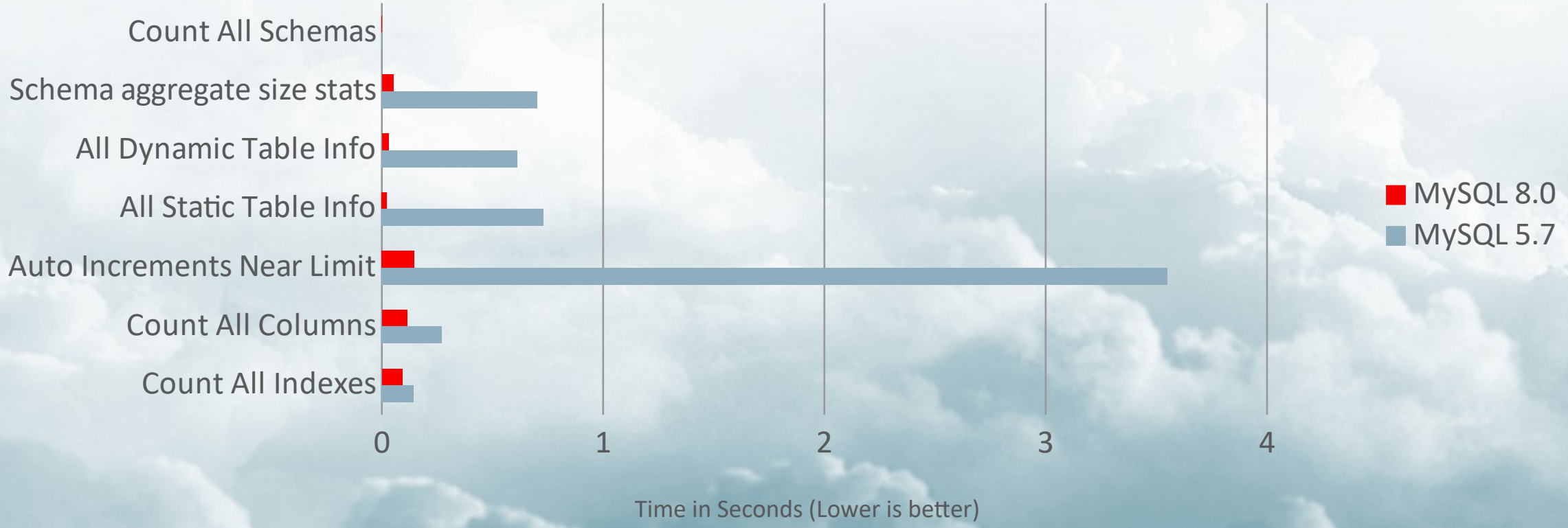
Data Dictionary

Increased reliability and consistency with transactional meta data repository

Information Schema Performance

100 schemas times 50 tables (5000 tables)

Already faster at **7/10 queries** in our test suite!



30x Faster

```
SELECT TABLE_SCHEMA,
 TABLE_NAME, TABLE_TYPE,
 ENGINE, ROW_FORMAT
FROM information_schema.tables
WHERE TABLE_SCHEMA LIKE 'db%';
```

Test Performed with 100 schemas, each with 50 tables.

Common Table Expressions (CTE)

- “With queries”
- Both Recursive and Non-Recursive Forms
- Simplifies writing complex SQL:

```
WITH t1 AS
   (SELECT * FROM tblA WHERE a='b')
SELECT * FROM t1;
```

Common Table Expressions (CTEs)

- Alternative to derived table of subquery, so called “WITH clause”
- For improvement of readability and performance

```
WITH tickets_filtered AS (
 SELECT tickets.*, seats.doc
 FROM tickets
 INNER JOIN seats ON
  tickets.seat_id = seats.id
 WHERE tickets.event_id = 3
)
SELECT * FROM tickets_filtered
 WHERE doc->"$.section" = 201\G
```

ORACLE

Recursive CTE

```
WITH RECURSIVE cte AS
(SELECT ... FROM table_name
   UNION ALL
   SELECT ... FROM cte where ...)
SELECT ... FROM cte;
```

Seed select

Recursive select

- A recursive CTE refers to itself in a subquery.
- The “seed” SELECT is executed once to create the initial data subset, the recursive SELECT is repeatedly executed to return subsets of data until the complete result set is obtained.
- Useful to dig in hierarchies (parent/child, part/subpart).
- Similar to Oracle's CONNECT BY.

Common Table Expressions (CTEs)

- Alternative to derived table of subquery, so called “`WITH` clause”
- For improvement of readability and performance

```
WITH tickets_filtered AS (
 SELECT tickets.*, seats.doc
 FROM tickets
 INNER JOIN seats ON
  tickets.seat_id = seats.id
 WHERE tickets.event_id = 3
)
SELECT * FROM tickets_filtered
 WHERE doc->"$.section" = 201\G
```

Recursive CTE

Print 1 to 10

```
mysql> WITH RECURSIVE qn AS
    (
        SELECT 1 AS a
        UNION ALL
        SELECT 1+a FROM qn WHERE a<10
    )
    SELECT * FROM qn;
```

```
+------+
| a    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
|    7 |
|    8 |
|    9 |
|   10 |
+------+
10 rows in set (0.00 sec)
```

Window Functions

- Frequently requested feature for data analysis like ranking of data.
- Calculation across a set of rows that are related to the current row.

```
SELECT name, dept_id,
salary,
 RANK() OVER w AS `rank`
FROM employee
 WINDOW w AS
   (PARTITION BY dept_id
    ORDER BY salary DESC);
```

Window Functions

```
mysql> SELECT * FROM sales ORDER BY country, year, product;
+------+---------+------------+--------+
| year | country | product    | profit |
+------+---------+------------+--------+
| 2000 | Finland | Computer   |   1500 |
| 2000 | Finland | Phone      |    100 |
| 2001 | Finland | Phone      |     10 |
| 2000 | India   | Calculator |     75 |
| 2000 | India   | Calculator |     75 |
| 2000 | India   | Computer   |   1200 |
| 2000 | USA     | Calculator |     75 |
| 2000 | USA     | Computer   |   1500 |
| 2001 | USA     | Calculator |     50 |
| 2001 | USA     | Computer   |   1500 |
| 2001 | USA     | Computer   |   1200 |
| 2001 | USA     | TV         |    150 |
| 2001 | USA     | TV         |    100 |
+------+---------+------------+--------+
```

Window Functions

- Frequently requested feature for data analysis like ranking of data
- Calculation across a set of rows that are related to the current row

```
SELECT name, dept_id,
salary,
 RANK() OVER w AS `rank`
FROM employee
 WINDOW w AS
  (PARTITION BY dept_id
   ORDER BY salary DESC);
```

Window Functions

```
mysql> SELECT SUM(profit) AS total_profit
       FROM sales;
+--------------+
| total_profit |
+--------------+
|         7535 |
+--------------+
```

```
mysql> SELECT country, SUM(profit) AS country_profit
       FROM sales
       GROUP BY country
       ORDER BY country;
+---------+----------------+
| country | country_profit |
+---------+----------------+
| Finland |           1610 |
| India   |           1350 |
| USA     |           4575 |
+---------+----------------+
```

Window Functions

- Frequently requested feature for data analysis like ranking of data
- Calculation across a set of rows that are related to the current row

```
SELECT name, dept_id,
salary,
 RANK() OVER w AS `rank`
FROM employee
 WINDOW w AS
  (PARTITION BY dept_id
   ORDER BY salary DESC);
```

Window Functions

```
mysql> SELECT
         year, country, product, profit,
         SUM(profit) OVER() AS total_profit,
         SUM(profit) OVER(PARTITION BY country) AS country_profit
         SUM(profit) OVER(PARTITION BY country
                ROWS UNBOUNDED PRECEDING)
             AS running_total,
       FROM sales
       ORDER BY country, year, product, profit;
```

year	country	product	profit	total_profit	country_profit	running_total
2000	Finland	Computer	1500	7535	1610	1500
2000	Finland	Phone	100	7535	1610	1600
2001	Finland	Phone	10	7535	1610	1610
2000	India	Calculator	75	7535	1350	75
2000	India	Calculator	75	7535	1350	150
2000	India	Computer	1200	7535	1350	1350
2000	USA	Calculator	75	7535	4575	75
2000	USA	Computer	1500	7535	4575	1575
2001	USA	Calculator	50	7535	4575	1625
2001	USA	Computer	1200	7535	4575	2825
2001	USA	Computer	1500	7535	4575	4325
2001	USA	TV	100	7535	4575	4425
2001	USA	TV	150	7535	4575	4575

Window Functions

- Frequently requested feature for data analysis like ranking of data
- Calculation across a set of rows that are related to the current row

```
SELECT name, dept_id,
salary,
 RANK() OVER w AS `rank`
FROM employee
 WINDOW w AS
  (PARTITION BY dept_id
   ORDER BY salary DESC);
```

MySQL 8.0: JSON datatype & Document Store API

Data Type

JSON Datatype

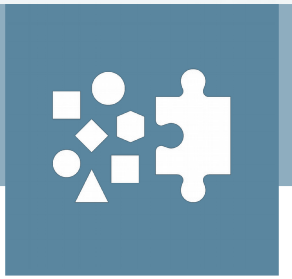
Seamlessly managing "unstructured" data in RDBMS tables with efficient update performance

SQL Function

JSON Functions

Various SQL functions to search and modify JSON. Analysing JSON with SQL by converting into table with `JSON_TABLE()`

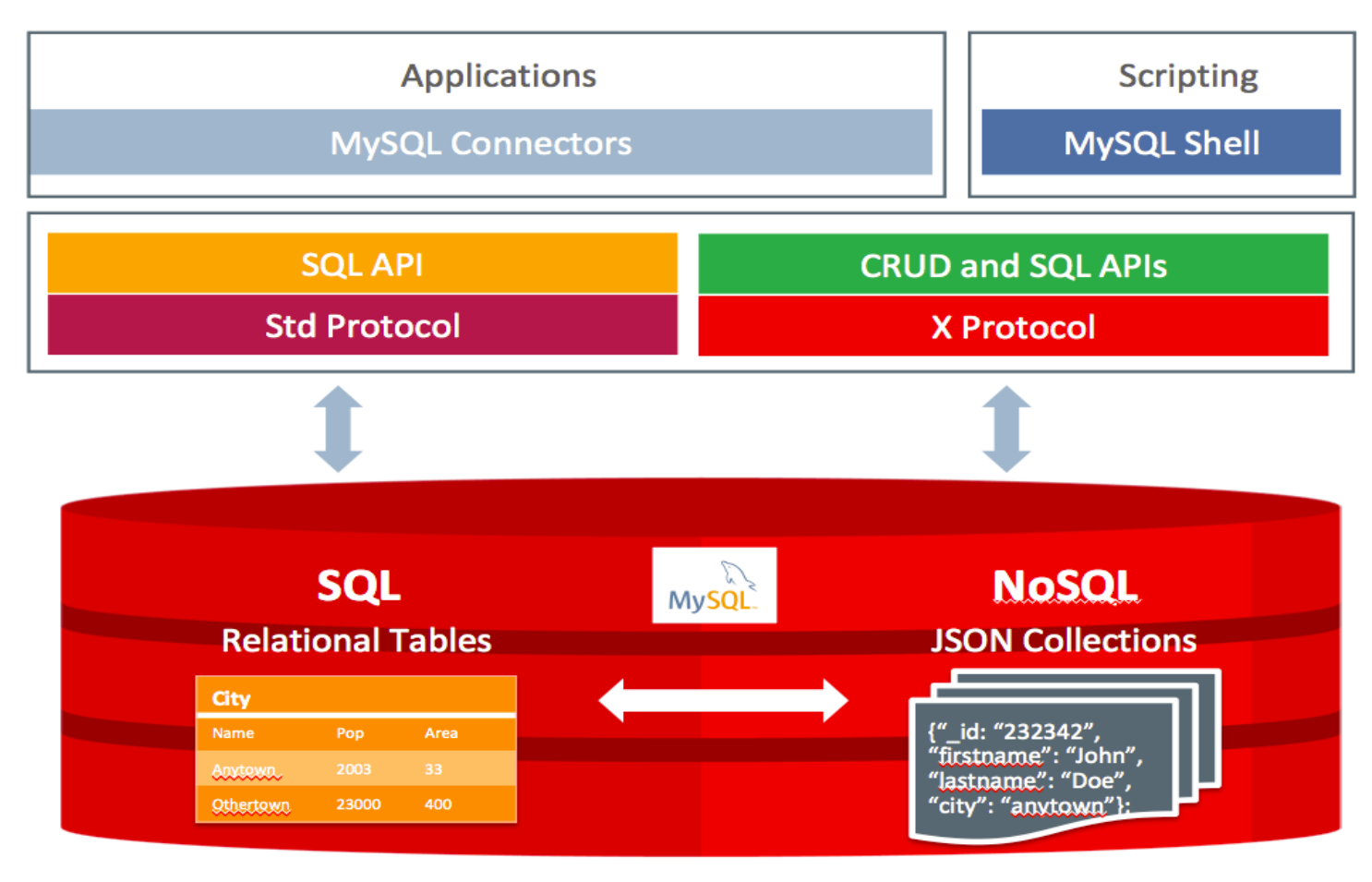
Hybrid API



MySQL X DevAPI

Hybrid CRUD API of both SQL and NoSQL provides more flexibility for development

MySQL 8.0: JSON datatype & Document Store API

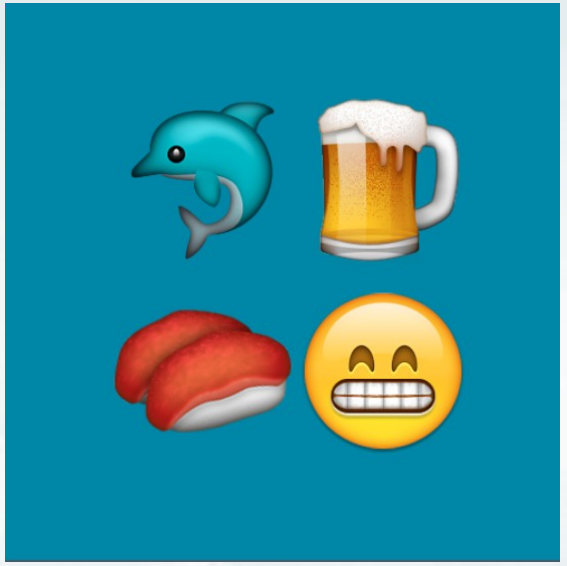


ORACLE

Enhanced GIS Support

- Geography support.
  - st_distance()
- Spatial Reference Systems (SRS) Support.
- SQL/MM Information Schema views.
- Standard compliant axis ordering in import/export functions.
- Helper functions to manipulate and convert data.
  - st_x(geom, x)
  - st_y(geom, y)
  - st_srid(geom, srid)

utf8mb4 as default character set

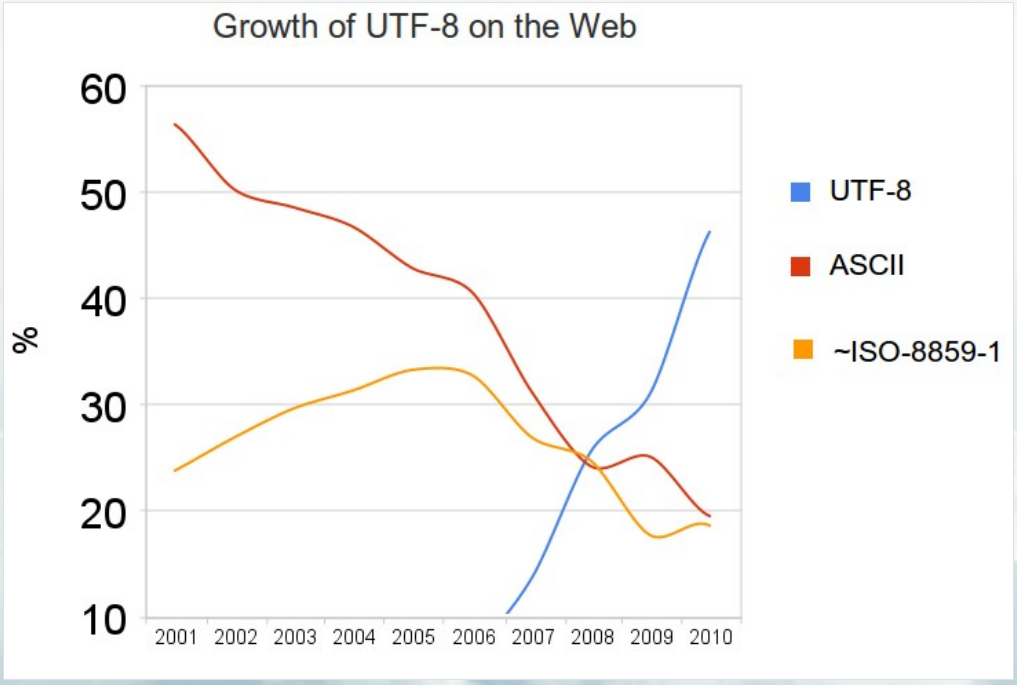


- Default Character Set of MySQL 8.0.
- utf8mb4 support Emoji.
- Up to 16x Faster Performance.
- Based on Unicode 9.0.
- New collations based on UCA with Accent/Case sensitivity.
  - including Japanese!

utf8mb4 as default character set

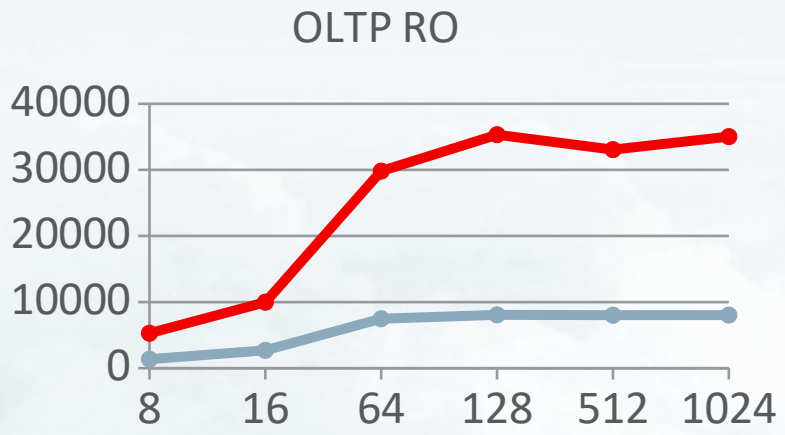
The character set for the Web

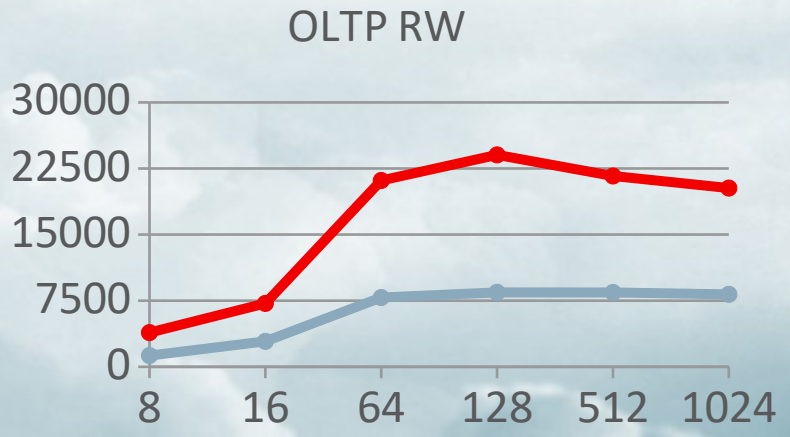
- UTF-8 is the dominating character set in today's applications.
- Requires 1-4 bytes for storing characters.
- Historically a performance problem.
  - But no any more!



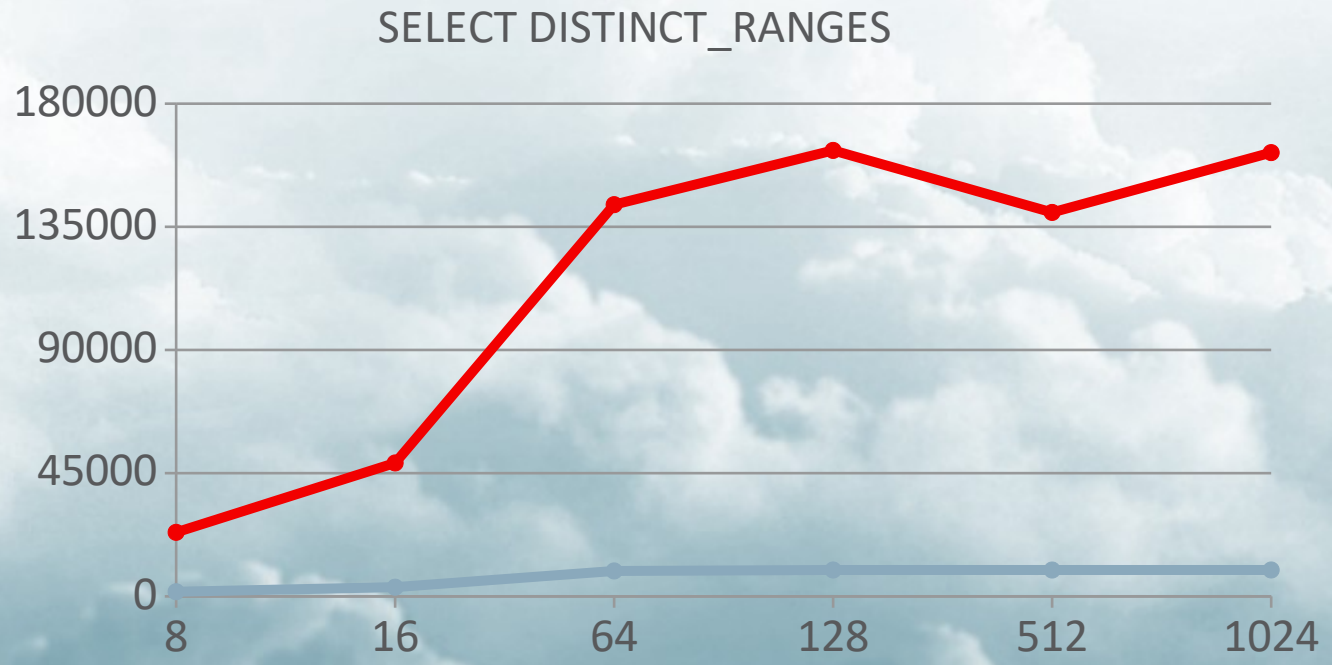
https://en.wikipedia.org/wiki/UTF-8

MySQL 8.0 utf8mb4 *vs* MySQL 5.7 utf8mb3





+270-340% in OLTP RO
+150-200% in OLTP RW
+1300-1600% in SELECT DISTINCT_RANGES



Better Handing of Hot Rows


```
SELECT seat_no
   FROM seats
   JOIN seat_rows USING ( row_no )
WHERE seat_no IN (3,4)
   AND seat_rows.row_no IN (12)
   AND booked = 'NO'
FOR UPDATE OF seats SKIP LOCKED
FOR SHARE OF seat_rows NOWAIT;
```

Non deterministically skip over locked rows

Error immediately if a row is already locked

Optimizer Cost Model

Improved to consider buffer pool fit


```
SELECT * FROM Country
   WHERE population > 20000000;
```

Model for a table scan:

pages in table *
(IO_BLOCK_READ_COST |
MEMORY_BLOCK_READ_COST)

records * ROW_EVALUATE_COST

= 25.4 100% in memory
= 29.9 100% on disk

Model for a range scan:

records_in_range *
(IO_BLOCK_READ_COST |
MEMORY_BLOCK_READ_COST)

records_in_range *
ROW_EVALUATE_COST + #
records_in_range *
ROW_EVALUATE_COST

= 22.5 100% in memory
= 60 100% on disk

Model accounts for memory fit.

Disk IO block read defaults to 1.0.
Memory defaults to 0.25.

Better performance
for range scan
not in memory.

Optimizer Cost Model

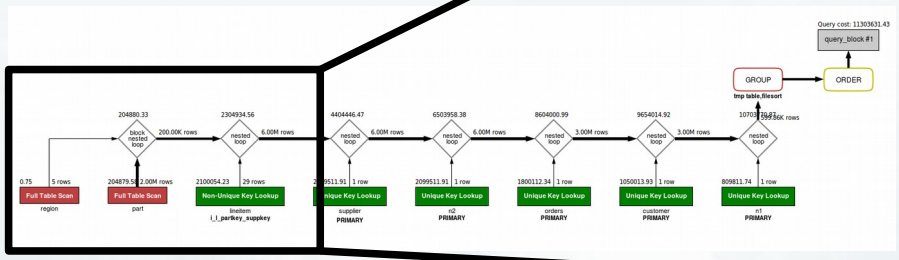
Example: DBT-3 Query 8

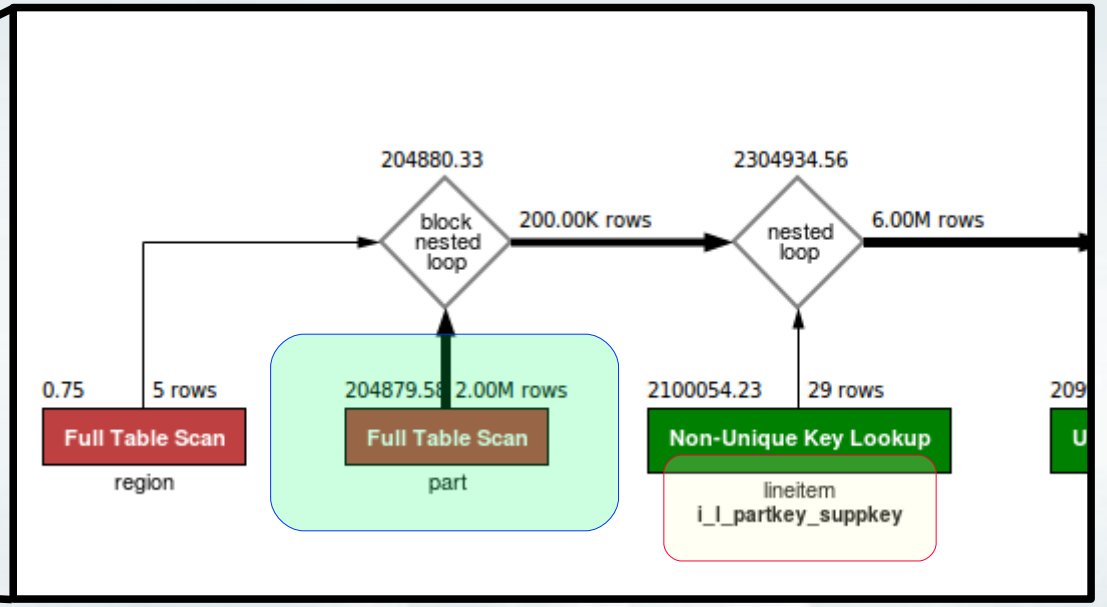
```
SELECT o_year,
       SUM(CASE WHEN nation = 'FRANCE' THEN volume ELSE 0 END) / SUM(volume) AS
mkt_share
FROM (
    SELECT EXTRACT(YEAR FROM o_orderdate) AS o_year,
           l_extendedprice * (1 - l_discount) AS volume, n2.n_name AS nation
    FROM part
    JOIN lineitem ON p_partkey = l_partkey
    JOIN supplier ON s_suppkey = l_suppkey
    JOIN orders ON l_orderkey = o_orderkey
    JOIN customer ON o_custkey = c_custkey
    JOIN nation n1 ON c_nationkey = n1.n_nationkey
    JOIN region ON  n1.n_regionkey = r_regionkey
    JOIN nation n2 ON s_nationkey = n2.n_nationkey
    WHERE r_name = 'EUROPE' AND o_orderdate BETWEEN '1995-01-01' AND '1996-12-31'
      AND p_type = 'PROMO BRUSHED STEEL'
) AS all_nations GROUP BY o_year ORDER BY o_year;
```

Optimizer Cost Model

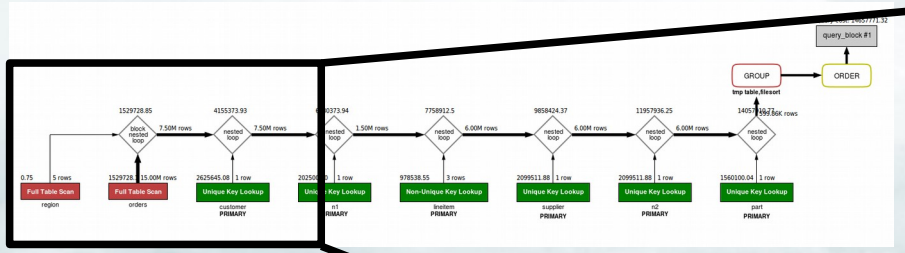
Example: DBT-3 Query 8

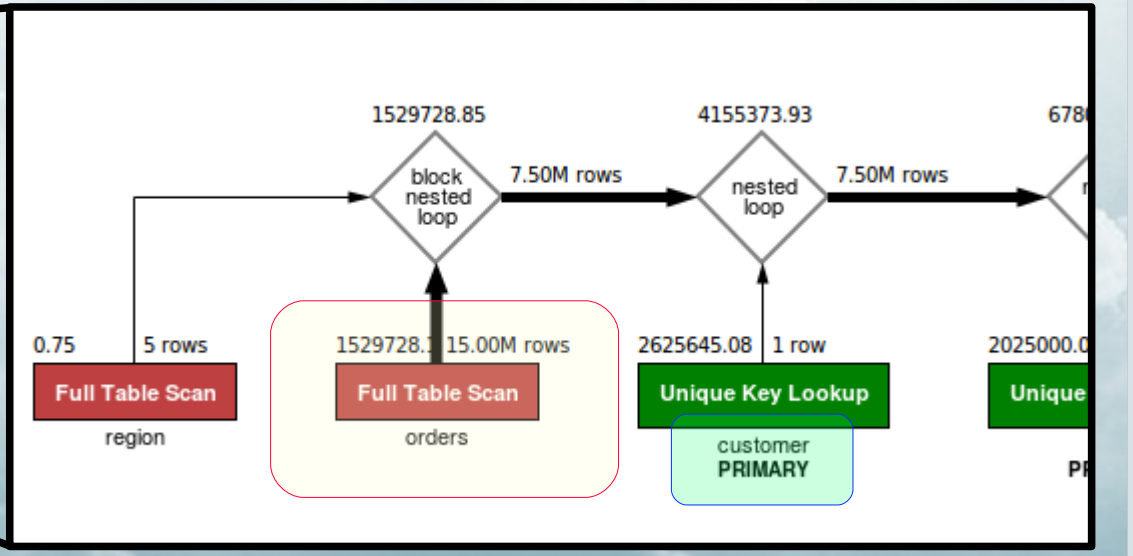
Plan A





Plan B





Optimizer Cost Model

Example: DBT-3 Query 8

	In Memory innodb_buffer_pool=32G	**Disk Bond** innodb_buffer_pool=1G
Plan A	5.8 secs	9 min 47 secs
Plan B	77.5 secs	3 min 49 secs

	In Memory innodb_buffer_pool=32G	**Disk Bond** innodb_buffer_pool=1G
MySQL 5.6	Plan B	
MySQL 5.7	Plan A	
MySQL 8.0	Plan A	Plan B

Histograms

Better query plans

- More consistent query execution for cases when data is skewed
- Lower cost to maintain than an index

Performance Schema Histograms

Showing distribution of query time from a run of mysqlslap


```
+---------+-------------------------------------------------------------+-------+
| bucket  | visualization                                               | count |
+---------+-------------------------------------------------------------+-------+
| 0us+    | #                                                           |  1253 |
| 10us+   | ##############################                              | 43102 |
| 100us+  | ################                                            | 25013 |
| 1ms+    | #                                                           |  2003 |
| 10ms+   |                                                             |   325 |
| 100ms+  |                                                             |    17 |
| 1s+     |                                                             |     0 |
| 10s+    |                                                             |     0 |
+---------+-------------------------------------------------------------+-------+
8 rows in set (0.08 sec)
```

Generated with a quick CTE over events_statements_histogram_global

Performance Schema Histograms

```
query: INSERT INTO `t1` VALUES (...)
    db: mysqlslap
    total_latency: 54.43 s
    exec_count: 58377
    ock_latency: 1.70 s
    ...
    digest: 4e0c5b796c4052b0da4548fd7cb694be
    first_seen: 2017-04-16 20:59:16
    last_seen: 2017-04-16 21:00:34
    latency_distribution:
        0us+
        10us+     ##############################
        100us+    ####################
        1ms+      #
        10ms+
        100ms+
        1s+
        10s+
```

Available on a per statement digest level. Can quickly aggregate top-N statements with latency distribution.

Performance Schema Data Locks


```
SELECT thread_id, object_name, index_name, lock_type, lock_mode, lock_data
    FROM performance_schema.data_locks WHERE object_name = 'seats';
+-----------+-------------------+-----------------+---------------+---------------+-------------+
| thread_id | object_name       | index_name      | lock_type     | lock_mode     | lock_data   |
+-----------+-------------------+-----------------+---------------+---------------+-------------+
|        33 | seats             | NULL            | TABLE         | IX            | NULL        |
|        33 | seats             | PRIMARY         | RECORD        | X             | 3, 5        |
|        33 | seats             | PRIMARY         | RECORD        | X             | 3, 6        |
|        33 | seats             | PRIMARY         | RECORD        | X             | 4, 5        |
|        33 | seats             | PRIMARY         | RECORD        | X             | 4, 6        |
+-----------+-------------------+-----------------+---------------+---------------+-------------+
5 rows in set (0.00 sec)
```

Descending Indexes

For B+tree indexes

```
CREATE TABLE t1 (
    a INT,
    b INT,
    INDEX a_b (a DESC, b ASC)
);
```

- **In 5.7**: Index in ascending order is created, server scans it backwards.
- **In 8.0**: Index in descending order is created, server scans it forwards.

Benefits:

- Forward index scan is faster than backward index scan.
- Use indexes instead of filesort for ORDER BY clause with ASC/DESC sort key.

Invisible Indexes

- Indexes are “hidden” to the MySQL Optimizer
  - Not the same as “disabled indexes”.
  - Contents are fully up to date and maintained by DML.
- Two use cases:
  - Soft Delete *(Recycle Bin)*
  - Staged Rollout

Invisible Indexes

Soft Delete

- I don’t think this index is used any more:

```
ALTER TABLE Country ALTER INDEX c INVISIBLE;
```

- I need to revert:

```
ALTER TABLE Country ALTER INDEX c VISIBLE;
```

- It is now safe to drop:

```
ALTER TABLE Country DROP INDEX c;
```

Invisible Indexes

Staged Rollout

- Adding any new index can change existing execution plans.
- All change introduces risk of regression.
- Invisible indexes allows you to stage all changes.
  - i.e. put the database in a “prepared” state.
- Turn on changes at an opportune time.

```
ALTER TABLE Country ADD INDEX c (Continent) INVISIBLE;
```

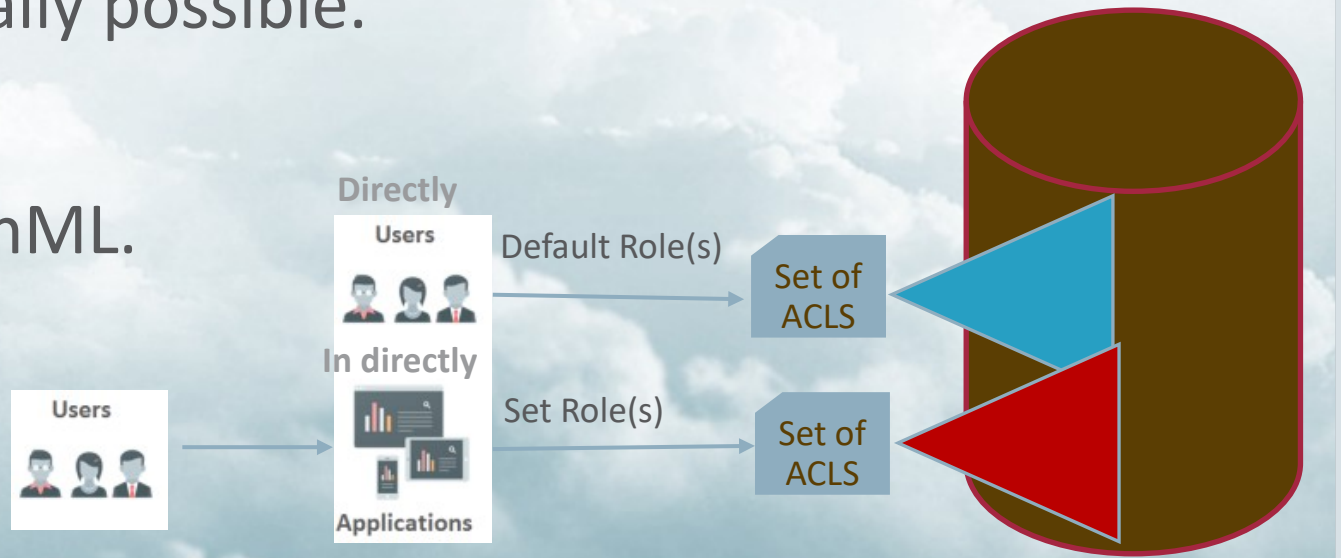
after some time

```
ALTER TABLE Country ALTER INDEX c VISIBLE;
```

MySQL Roles

Improving MySQL Access Controls.

- Introduced in the 8.0.0 DMR.
- Easier to manage user and applications rights.
- As standards compliant as practically possible.
- Multiple default roles.
- Can export the role graph in GraphML.



Dynamic Privileges

Provides finer grained administrative level access controls.

- Too often super is required for tasks when less privilege is really needed.
  - Support concept of “least privilege”
- Needed to allow adding administrative access controls.
  - Now can come with new components.
  - Examples
    - Replication
    - HA
    - Backup

Performance Schema Indexes

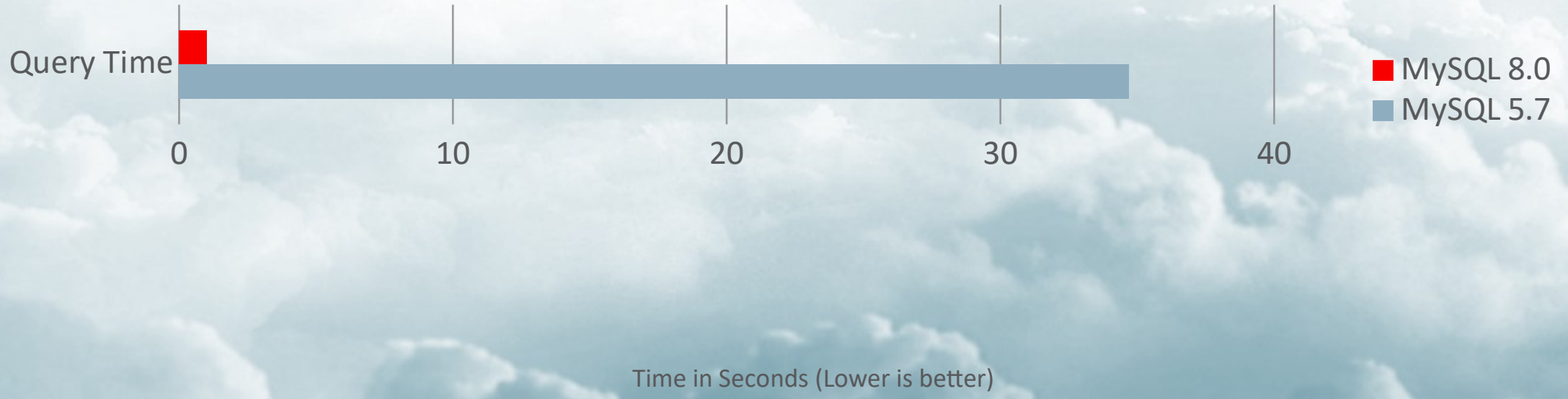
- Allows for more efficient access to Performance Schema tables
- A total of **90 indexes** across **89 tables**
- Adds zero overhead
  - A physical index is not maintained internally
  - Implementation of indexes *tricks* the optimizer into better execution plan

Performance Schema Indexes

Over 30x faster!

```
SELECT * FROM sys.session
```

1000 active sessions



Time in Seconds (Lower is better)

Performance Schema Instrumenting SQL Errors

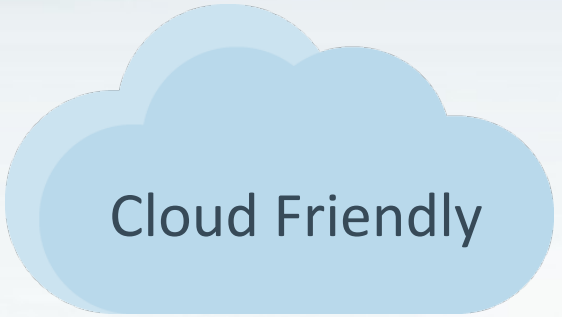
Aggregation	Table Name
By Account	events_errors_summary_by_account_by_error
By Host	events_errors_summary_by_host_by_error
By Thread	events_errors_summary_by_thread_by_error
By User	events_errors_summary_by_user_by_error
Global	events_errors_summary_global_by_error

Performance Schema Instrumenting SQL Errors

```
mysql> SELECT * FROM test.no_table;
ERROR 1146 (42S02): Table 'test.no_table' doesn't exist


mysql> SELECT * FROM performance_schema.events_errors_summary_global_by_error
         WHERE sum_error_handled > 0 OR SUM_ERROR_RAISED > 0\G
*************************** 1. row ***************************
     ERROR_NUMBER: 1146
       ERROR_NAME: ER_NO_SUCH_TABLE
        SQL_STATE: 42S02
 SUM_ERROR_RAISED: 1
SUM_ERROR_HANDLED: 0
       FIRST_SEEN: 2016-09-11 20:52:42
        LAST_SEEN: 2016-09-11 20:52:42
1 row in set (0.00 sec)
```

Persist Configuration



- Persist GLOBAL Server Variables
  - `SET PERSIST max_connections = 500;`
- Examples Include:
  - Offline_mode.
  - Read_Only.
- Requires no filesystem access.
- Includes timestamp and change user.

Persist Configuration: Variables Info

Find the source of variables changed on your installation

```
SELECT * FROM performance_schema.variables_info WHERE variable_source != 'COMPILED';
+-------------------------+-----------------+--------------------+-----------+-----------+---------------------+----------+-----------+
| VARIABLE_NAME           | VARIABLE_SOURCE | VARIABLE_PATH      | MIN_VALUE | MAX_VALUE | SET_TIME            | SET_USER | SET_HOST  |
+-------------------------+-----------------+--------------------+-----------+-----------+---------------------+----------+-----------+
| autocommit              | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
| basedir                 | COMMAND_LINE    |                    | 0         | 0         | 2017-04-16 21:08:11 |          |           |
| bind_address            | EXPLICIT        | [..]my.sandbox.cnf | 0         | 0         | 2017-04-16 21:08:11 |          |           |
| character_set_client    | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
| character_set_results   | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
| collation_connection    | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
| datadir                 | COMMAND_LINE    |                    | 0         | 0         | 2017-04-16 21:08:11 |          |           |
| foreign_key_checks      | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
| log_error               | COMMAND_LINE    |                    | 0         | 0         | 2017-04-16 21:08:11 |          |           |
| lower_case_table_names  | EXPLICIT        | [..]my.sandbox.cnf | 0         | 2         | 2017-04-16 21:08:11 |          |           |
| pid_file                | COMMAND_LINE    |                    | 0         | 0         | 2017-04-16 21:08:11 |          |           |
| plugin_dir              | COMMAND_LINE    |                    | 0         | 0         | 2017-04-16 21:08:11 |          |           |
| port                    | COMMAND_LINE    |                    | 0         | 65535     | 2017-04-16 21:08:11 |          |           |
| socket                  | COMMAND_LINE    |                    | 0         | 0         | 2017-04-16 21:08:11 |          |           |
| sql_mode                | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
| sql_notes               | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
| time_zone               | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
| tmpdir                  | EXPLICIT        | [..]my.sandbox.cnf | 0         | 0         | 2017-04-16 21:08:11 |          |           |
| unique_checks           | DYNAMIC         |                    | 0         | 0         | 2017-04-16 20:56:53 | msandbox | localhost |
+-------------------------+-----------------+--------------------+-----------+-----------+---------------------+----------+-----------+
19 rows in set (0.00 sec)
```

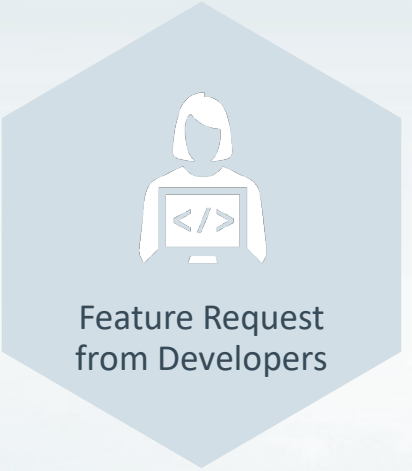
InnoDB Redo and Undo Encryption

- AES 256 encryption
- *Encrypted when redo/undo log data is written to disk*
- Decryption occurs when redo/undo log data is read from disk
- Once redo/undo log data is read into memory, it is in unencrypted form.
- Two tiered encryption – like Innodb tablepace encryption
  - Fast key rotation, high performance
- *Easy to use*
  - *Enabled using innodb_redo_log_encrypt and innodb_undo_log_encrypt*

InnoDB Auto Increment Persists

- First reported as BUG #199.
- Auto increment counters are now written to the REDO log.
- Allows for fast changing meta data.

UUID and Bit-wise Improvements



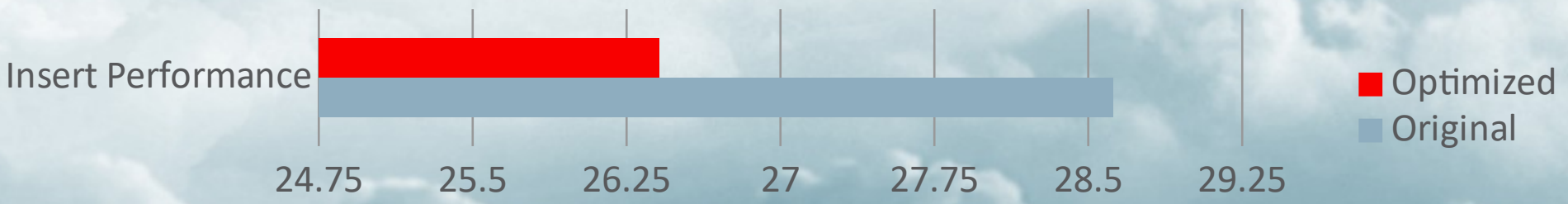
- Functions to convert UUID to and from binary:
  - `UUID_TO_BIN()`
  - `BIN_TO_UUID()`
    - *plus* `IS_UUID()`
- Bit-wise operations on binary data types.
- Bit-wise operations on binary data types.
  - Designed with IPv6 in mind:
    - `INET6_ATON(address)` & `INET6_ATON(network)`

UUID and Bit-wise Improvements

- Binary format is now smaller and insert-order efficient:

From **VARCHAR(36)** 53303f87-78fe-11e6-a477-8c89a52c4f3b

To **VARBINARY(16)** 11e678fe53303f87a4778c89a52c4f3b



All these features plus…

- Source code now documented with Doxygen
- Plugin Infrastructure!
- Expanded GIS Support
- Expanded Query Hints Support
- Improved Scan Query Performance
- Improved BLOB Storage
- Improved Memcached Interface
- Cost Model Improvements
- Scalability Improvements
- Atomicity in Privileges
- Parser Refactoring
- Improvements to Temporary Tables
- C++11 and Toolchain Improvements

- Replication Applier Thread Progress Reports
- GTID_PURGED always settable
- Improved Parallel Replication
- SQL Grouping Function
- Optimizer Trace detailed sort statistics
- Smaller Package Downloads
- JSON Aggregate, Pretty print functions
- JSON performance improvements
- Expanded Query Hints
- Improved usability of cost constant configuration

MySQL Enterprise Edition

Advanced Features

- Scalability
- High Availability
- Security
- Audit
- Encryption
- Firewall

Management Tools

- Monitoring
- Backup
- Development
- Administration
- Migration

Support

- Technical Support
- Consultative Support
- Oracle Certifications

MySQL Enterprise Edition: Security Features

- NEW! MySQL Enterprise Firewall
  - Block SQL Injection Attacks
  - Intrusion Detection
- MySQL Enterprise Encryption
  - Public/Private Key Cryptography
  - Asymmetric Encryption
  - Digital Signatures, Data Validation
- MySQL Enterprise Authentication
  - External Authentication Modules
    - Microsoft AD, Linux PAMs
- MySQL Enterprise Audit
  - User Activity Auditing, Regulatory Compliance
- MySQL Enterprise Monitor
  - Changes in Database Configurations, Users Permissions, Database Schema, Passwords
- MySQL Enterprise Backup
  - Securing Backups, AES 256 encryption

More information available at : http://www.mysql.com/products/enterprise/

Questions?


ORACLE®