MySQL Server 8.0

COSCUP 2017 in Taiwan

created: 2017/08/02

MySQL Global Business Unit
Master Principal Sales Consultant/Shinya Sugiyama

MySQL

Safe Harbor Statement

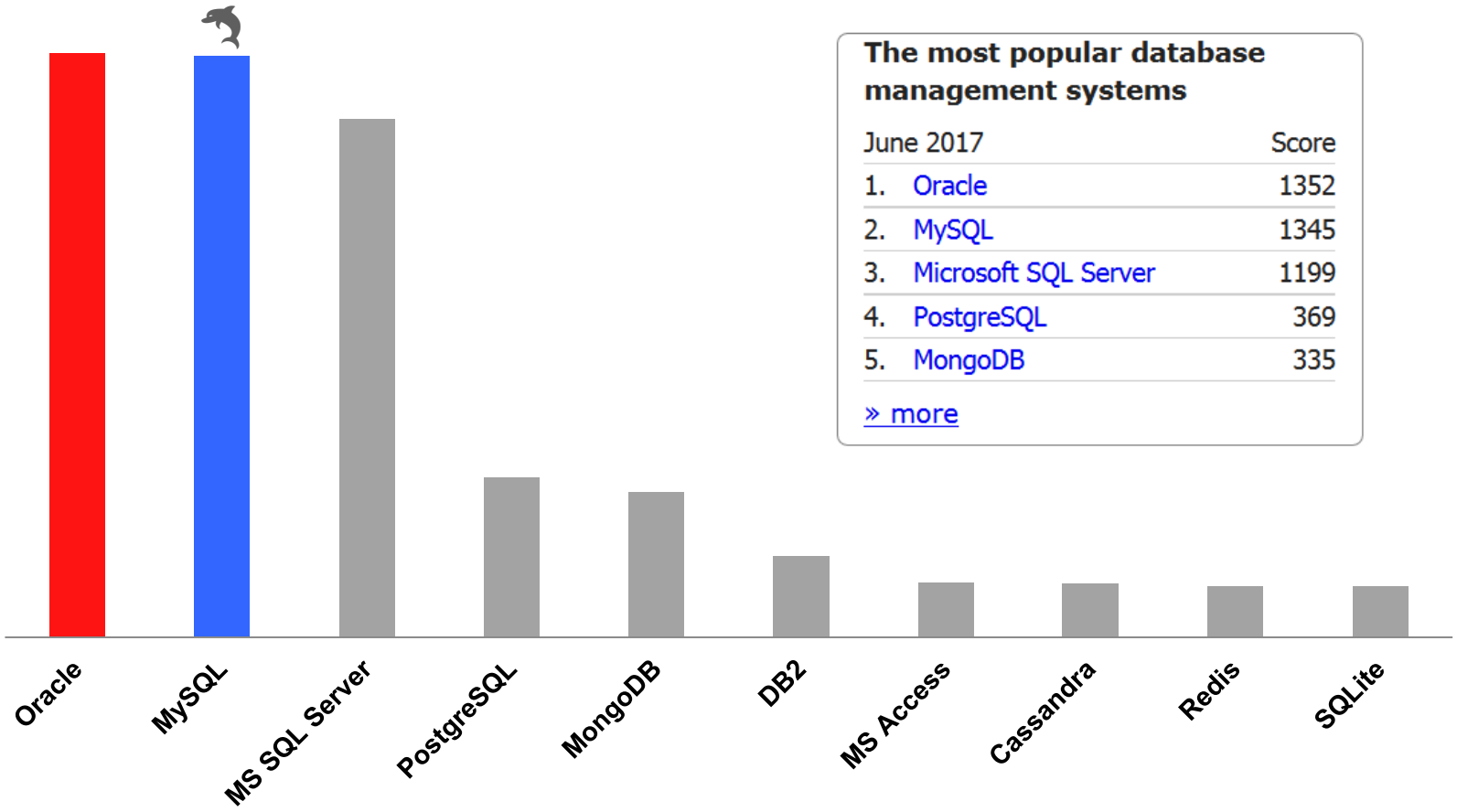
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

The world's one of the most popular open source database

http://db-engines.com/en/

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

https://db-engines.com/en/ranking_definition



The most popular database management systems

June 2017		Score
1.	Oracle	1352
2.	MySQL	1345
3.	Microsoft SQL Server	1199
4.	PostgreSQL	369
5.	MongoDB	335

» more

In Global

Many Web Site Use MySQL as their default database.

In Japan

Many Web Companies also use MySQL for their backed Database.

http://www.alexa.com/topsites/countries/JP

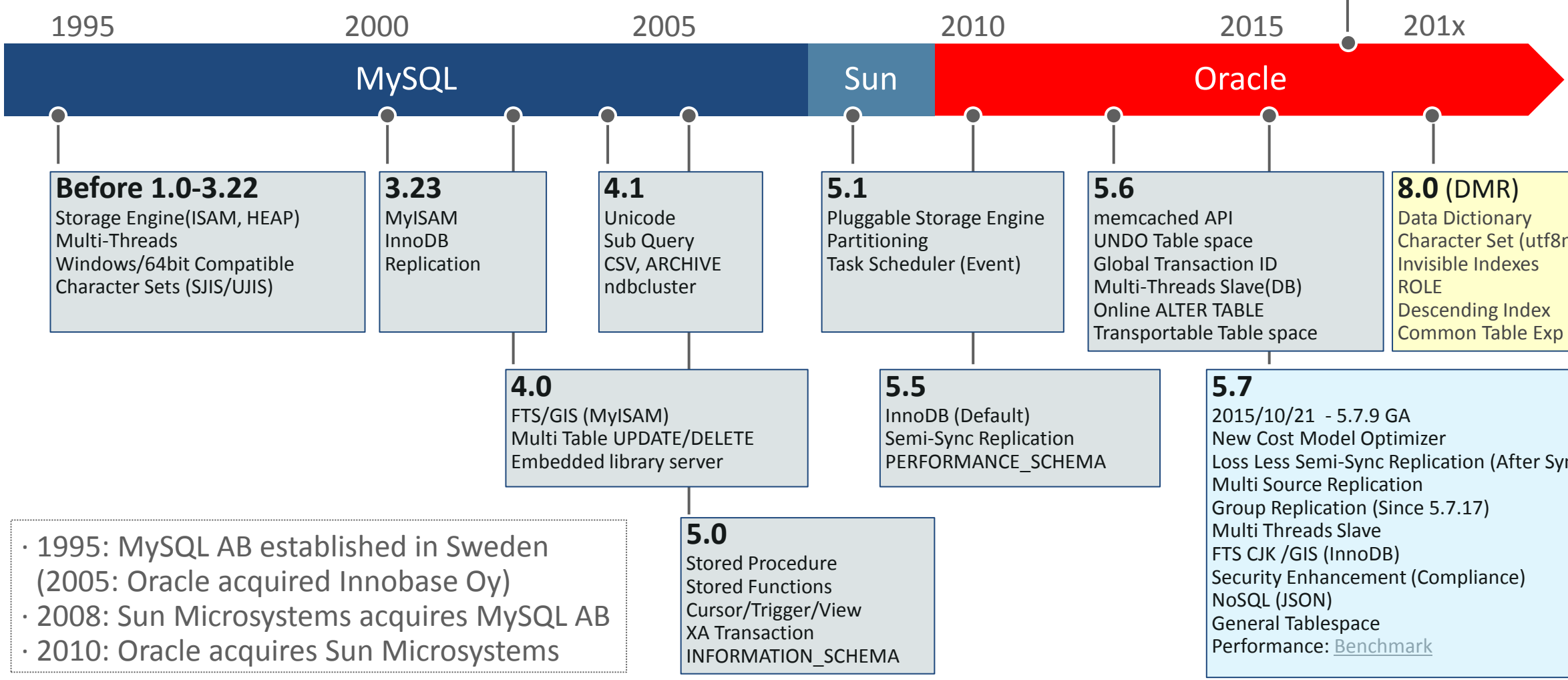
Company	URL
Square Enix	http://www.jp.square-enix.com/
Oricon ME	http://www.oricon.co.jp/
forTravel	http://4travel.jp/
Sony	https://www.sony.net/

分野	OSS	使用率
OS	Linux 系	67.3%
	BSD 系	12.9%
RDBMS	MySQL	53.1%
	PostgreSQL	35.0%
アプリケーションサーバー	Tomcat	35.6%
	JBoss	12.0%
システム運用管理	Zabbix	16.2%
	Nagios	7.1%
	Chef	3.9%
	Hinemos	1.9%
システムソフトウェア	Samba	21.4%
	BIND	13.6%
ハイパーバイザー	Xen	16.2%
	KVM	10.7%
クラウド基盤	OpenStack	6.1%
	Docker	4.5%
	CloudStack	3.6%
	Cloud Foundry	2.9%
データ分散処理	Hadoop	6.8%
	Spark	1.3%
NoSQL	MongoDB	4.5%
	Scalaris	4.2%
	Cassandra	2.6%
	Hypertable	2.6%

n=309

OSS usage ratio in Japanese company that use OSS in their business (Source IDC Japan, 2/2016)

History of MySQL



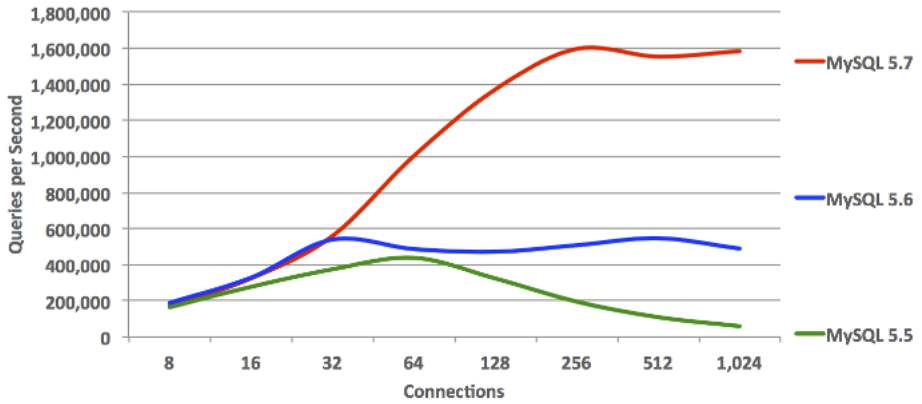
- 1995: MySQL AB established in Sweden
  (2005: Oracle acquired Innobase Oy)
- 2008: Sun Microsystems acquires MySQL AB
- 2010: Oracle acquires Sun Microsystems

Looking Back MySQL 5.7 – Improvements across the board

- Replication
- InnoDB
- Optimizer
- Security
- Performance Schema
- GIS
- FTS (Chinese, Korean, Japanese)

- Triggers
- Partitioning
- New! SYS Schema
- New! JSON
- Performance

200+ new features In total!



Continuing the JSON Developer Experience

MySQL 8.0

- MySQL Document Store
  - Document collections
  - Relational tables
  - Combine them as you like
- Administer MySQL via the MySQL Shell
  - One stop DevOps tool for MySQL
  - Out of the Box HA
  - Use your preferred language: SQL, JavaScript, Python, …
  - Relational or Document
- Additional JSON Functions
  - JSON_ARRAYAGG(), JSON_OBJECTAGG(), JSON_PRETTY(),
  - JSON_STORAGE_FREE(), JSON_STORAGE_SIZE()

```
select json_pretty(body) from T_JSON_DOC where id = 1¥G
********************** 1. row **********************
json_pretty(body): {
  "id": 1,
  "name": "",
  "price": 10000,
  "Conditions": [
    "NEW",
    2015,
    "Excellent"
  ]
}
```

WL#7987: JSON aggregation functions
https://dev.mysql.com/worklog/task/?id=7987
WL#9191: Add JSON_PRETTY function
https://dev.mysql.com/worklog/task/?id=9191
WL#9192: Add JSON_STORAGE_SIZE / JSON_STORAGE_FREE functions
https://dev.mysql.com/worklog/task/?id=9192

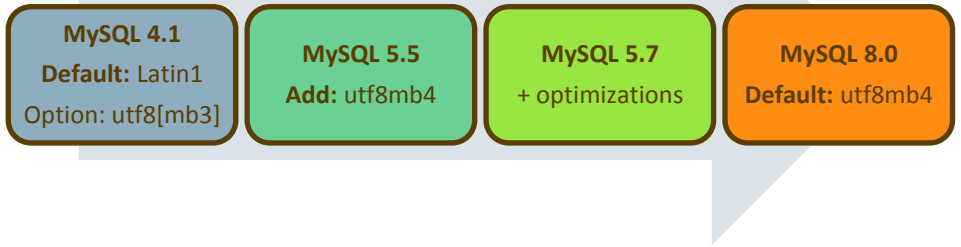
GIS

- 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)

```
mysql> SET @g1 = ST_GeomFromText('POINT(1 1)', 4326);
Query OK, 0 rows affected (0.01 sec)

mysql> SET @g2 = ST_GeomFromText('POINT(2 2)', 4326);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT ST_Distance(@g1, @g2);
+-----------------------+
| ST_Distance(@g1, @g2) |
+-----------------------+
|     156874.3859490455 |
+-----------------------+
1 row in set (0.06 sec)
```

WL#9347: Ellipsoidal ST_Distance for point and multipoint
https://dev.mysql.com/worklog/task/?id=9347
WL#8606: Mutator ST_X and ST_Y
https://dev.mysql.com/worklog/task/?id=8606
WL#8543: Mutator ST_SRID
https://dev.mysql.com/worklog/task/?id=8543

utf8mb4 as default character set



MySQL 8.0

- Support for the latest Unicode 9.0
- Accent (ai, as) and case sensitive (ci,cs) collations
- Language specific collations (including Japanese!)

```
mysql> select *,@@version from information_schema.COLLATIONS where CHARACTER_SET_NAME = 'utf8mb4' and IS_DEFAULT = 'YES';
+--------------------+--------------------+----+------------+-------------+---------+----------------------------+
| COLLATION_NAME     | CHARACTER_SET_NAME | ID | IS_DEFAULT | IS_COMPILED | SORTLEN | @@version                  |
+--------------------+--------------------+----+------------+-------------+---------+----------------------------+
| utf8mb4_general_ci | utf8mb4            | 45 | Yes        | Yes         |       1 | 5.7.18-enterprise-commercial |
+--------------------+--------------------+----+------------+-------------+---------+----------------------------+

mysql> select *,@@version from information_schema.COLLATIONS where CHARACTER_SET_NAME = 'utf8mb4' and IS_DEFAULT = 'YES';
+--------------------+--------------------+-----+------------+-------------+---------+---------------+-----------+
| COLLATION_NAME     | CHARACTER_SET_NAME | ID  | IS_DEFAULT | IS_COMPILED | SORTLEN | PAD_ATTRIBUTE | @@version |
+--------------------+--------------------+-----+------------+-------------+---------+---------------+-----------+
| utf8mb4_0900_ai_ci | utf8mb4            | 255 | Yes        | Yes         |       0 | NO PAD        | 8.0.2-dmr |
+--------------------+--------------------+-----+------------+-------------+---------+---------------+-----------+
```

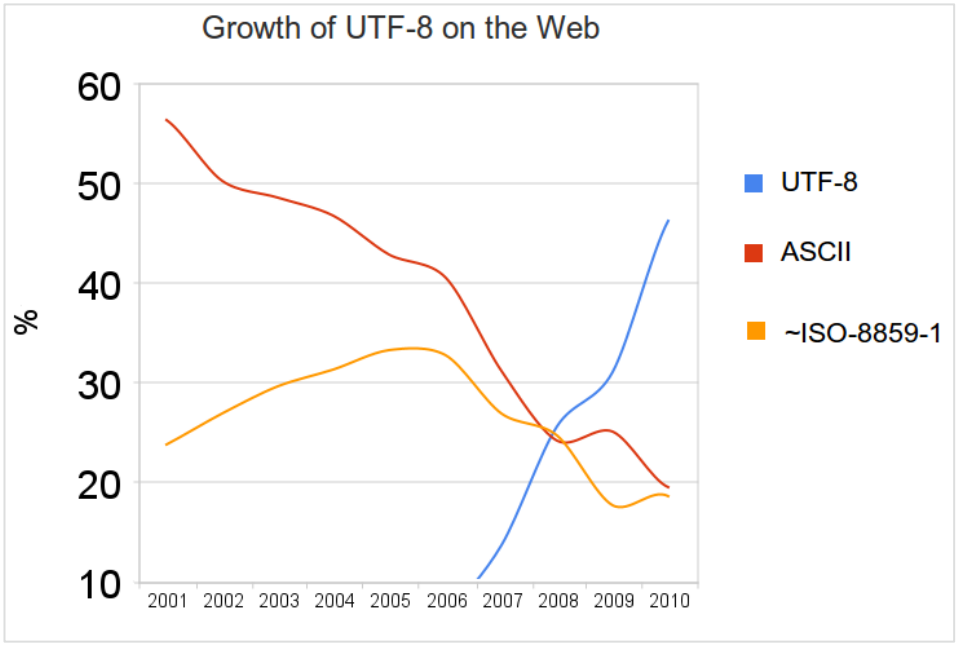
http://www.unicode.org/

WL#10818: Add utf8mb4 accent sensitive and case insensitive collation
https://dev.mysql.com/worklog/task/?id=10818
WL#7554: Switch to new default character set and change mtr test cases
https://dev.mysql.com/worklog/task/?id=7554

UTF-8

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



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

1 byte	Basic Latin letters, digits, and punctuation signs use.
2 byte	Most European and Middle East script letters, extended Latin letters
3 byte ~ 4 byte	Korean, Chinese, and Japanese

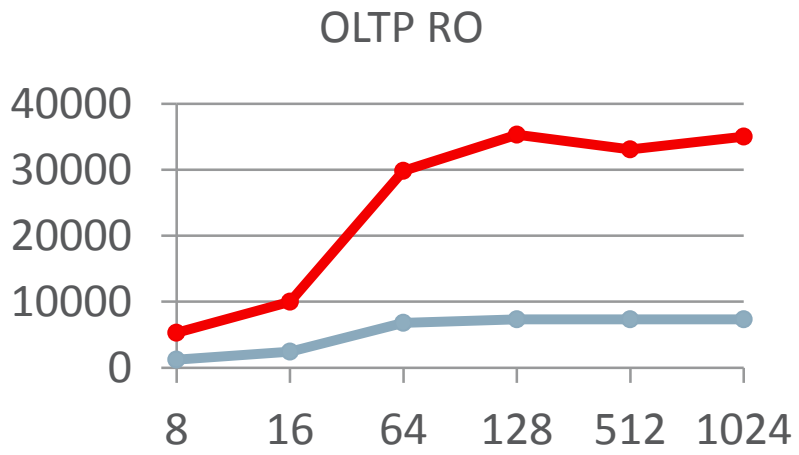
	UTF8(UTF8MB3)	UTF8MB4
🐬	X	◯
鮫	X	◯

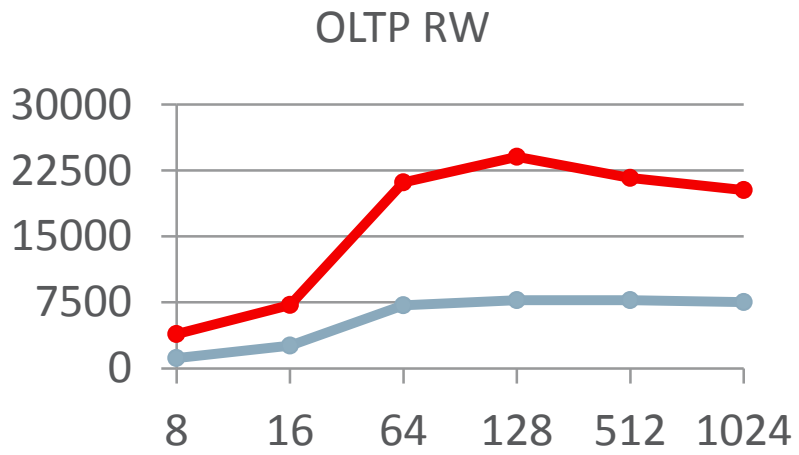
http://mysqlserverteam.com/mysql-8-0-collations-migrating-from-older-collations/

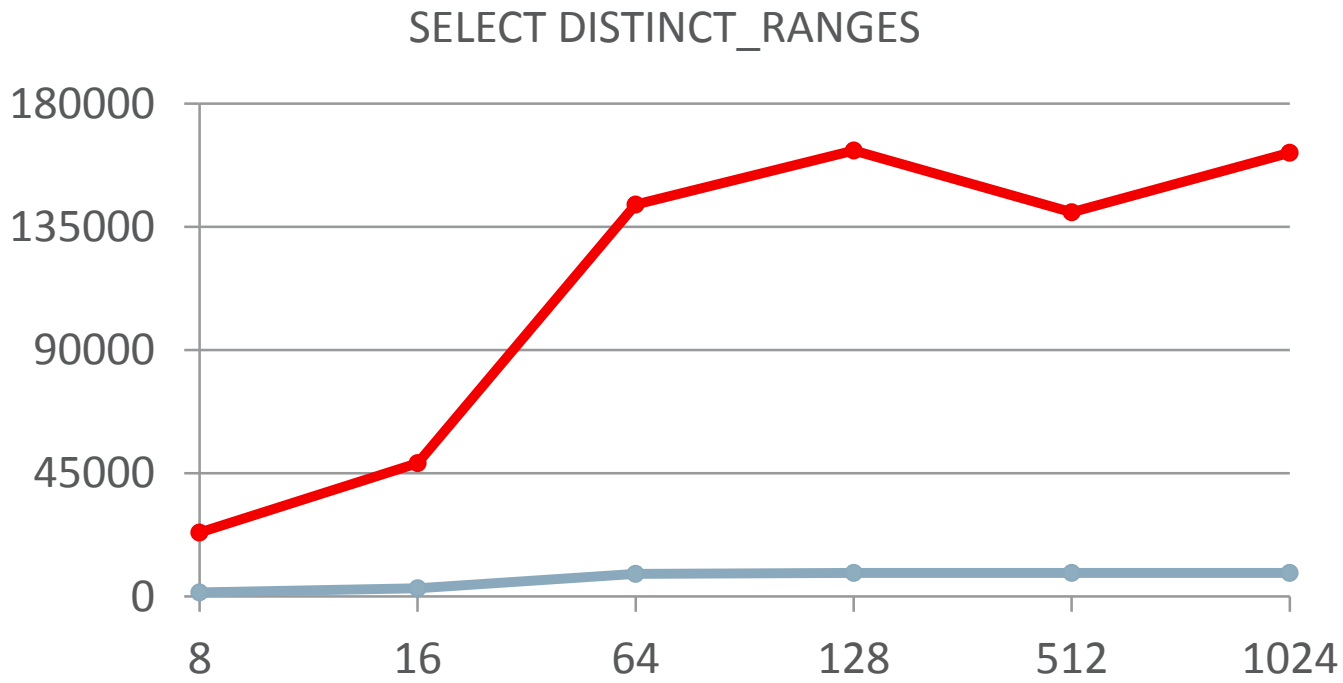
MySQL 8.0 *vs* MySQL 5.7 utf8mb4

TPS

+300-350% in OLTP RO
+176-233% in OLTP RW
+1500-1800% in SELECT DISTINCT_RANGES



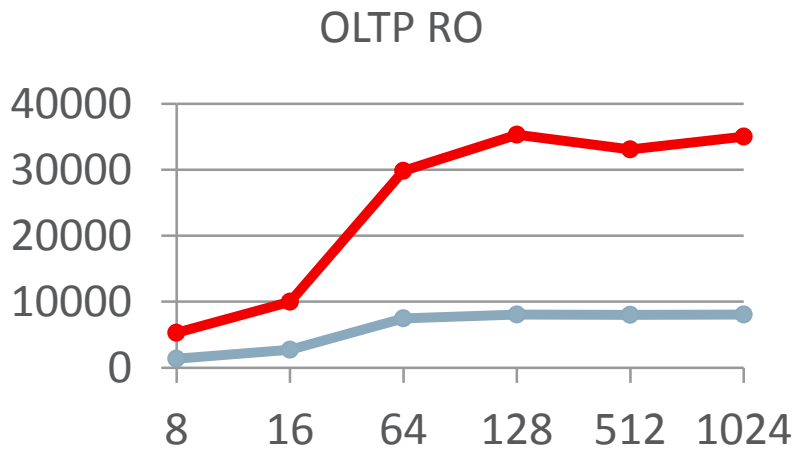


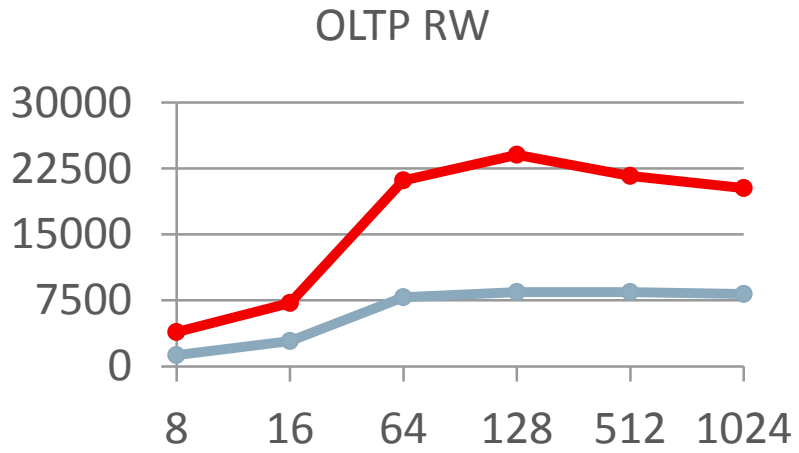


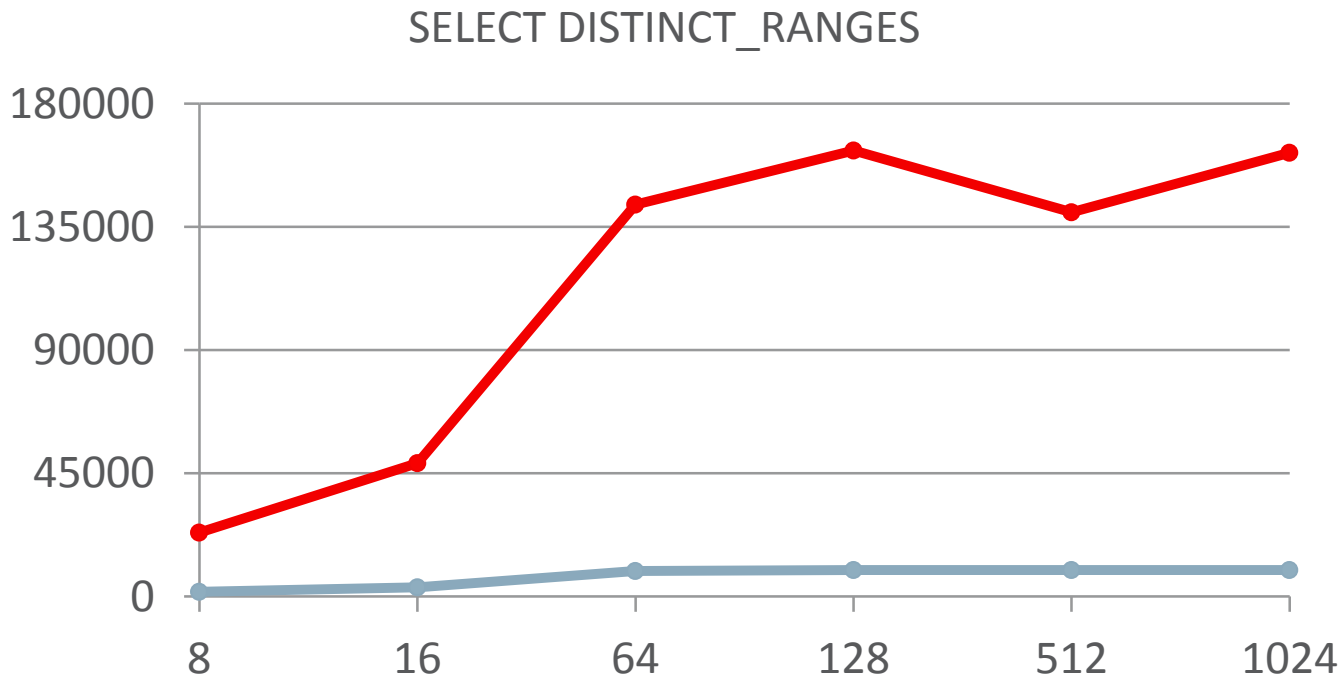
MySQL 8.0 utf8mb4 *vs* MySQL 5.7 utf8mb3

TPS

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







UUID and Bit-wise Improvements

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

UUIDs are a good alternative to AUTO_INCREMENT PRIMARY KEY.
But they also come with some disadvantages:
– increased storage: 36 characters
– more difficult to debug
– performance issues
  Size and not being ordered

```
mysql> select INET6_NTOA(INET6_ATON('59b0:c4d6:48b4:3717:f031:d05b:705d:6c65'));
+-----------------------------------------------------------------+
| INET6_NTOA(INET6_ATON('59b0:c4d6:48b4:3717:f031:d05b:705d:6c65')) |
+-----------------------------------------------------------------+
| 59b0:c4d6:48b4:3717:f031:d05b:705d:6c65                         |
+-----------------------------------------------------------------+

mysql> SELECT INET6_NTOA(subnet(ip_address, net_len)) as range_from,
    -> INET6_NTOA(subnet(ip_address, net_len) | host_mask(net_len)) as range_to FROM cidr LIMIT 2;
+-------------------------------------+-----------------------------------------+
| range_from                          | range_to                                |
+-------------------------------------+-----------------------------------------+
| d1b:7977:5915:437:6830::            | d1b:7977:5915:437:683f:ffff:ffff:ffff   |
| 8337:1c78:f38f:2805:4648:38ab:2ec2:: | 8337:1c78:f38f:2805:4648:38ab:2ec3:ffff |
+-------------------------------------+-----------------------------------------+
```

WL#8920: Improve usability of UUID manipulations
https://dev.mysql.com/worklog/task/?id=8920
WL#8699: Bit-wise operations on binary data types
https://dev.mysql.com/worklog/task/?id=8699

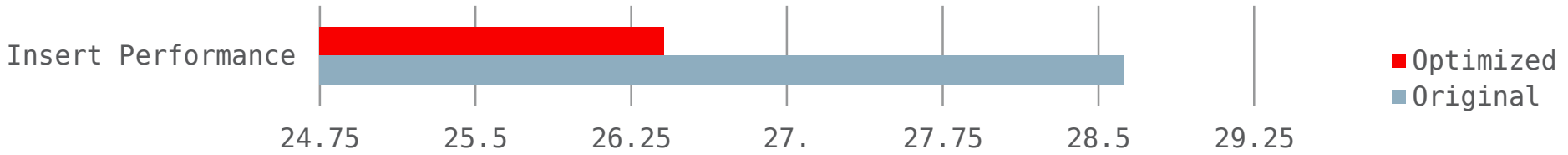
UUID_TO_BIN Optimization

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

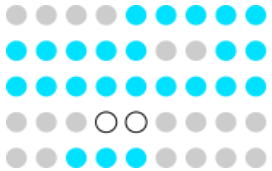
From **VARCHAR(36)** 53303f87-78fe-11e6-a477-8c89a52c4f3b

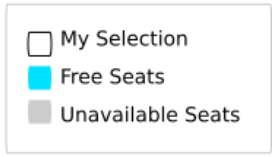
To **VARBINARY(16)** 11e678fe53303f87a4778c89a52c4f3b

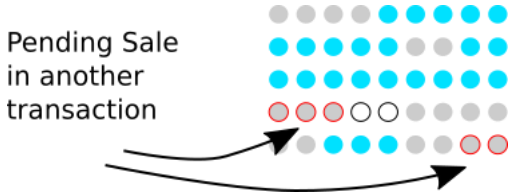
```
mysql> select bin_to_uuid(id_bin),id_var from T_UUID limit 1;
+--------------------------------------+--------------------------------------+
| bin_to_uuid(id_bin)                  | id_var                               |
+--------------------------------------+--------------------------------------+
| 1acebc1d-4f6d-11e7-a11b-0242ac110002 | 1acebc6d-4f6d-11e7-a11b-0242ac110002 |
+--------------------------------------+--------------------------------------+

mysql> select hex(id_bin),hex(id_var) from T_UUID limit 1;
+----------------------------------+------------------------------------------------------------------------------------+
| hex(id_bin)                      | hex(id_var)                                                                        |
+----------------------------------+------------------------------------------------------------------------------------+
| 1ACEBC1D4F6D11E7A11B0242AC110002 | 3161636526336642D346636642D313165372D613131622D303234326163313130303032 |
+----------------------------------+------------------------------------------------------------------------------------+
```



MySQL 8.0: 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
Ex) skip orders which are pending

Error immediately if a row is already locked

http://mysqlserverteam.com/mysql-8-0-1-using-skip-locked-and-nowait-to-handle-hot-rows/

WL#3597: Implement NOWAIT and SKIP LOCKED
https://dev.mysql.com/worklog/task/?id=3597
WL#8919: InnoDB: Implement NOWAIT and SKIP LOCKED
https://dev.mysql.com/worklog/task/?id=8919

Common Table Expressions

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

```
WITH t1 AS (SELECT * FROM city WHERE CountryCode = 'TWN')
SELECT * FROM t1 limit 5;
+------+-----------+-------------+-----------+------------+
| ID   | Name      | CountryCode | District  | Population |
+------+-----------+-------------+-----------+------------+
| 3263 | Taipei    | TWN         | Taipei    |    2641312 |
| 3264 | Kaohsiung | TWN         | Kaohsiung |    1475505 |
| 3265 | Taichung  | TWN         | Taichung  |     940589 |
| 3266 | Tainan    | TWN         | Tainan    |     728060 |
| 3267 | Panchiao  | TWN         | Taipei    |     523850 |
+------+-----------+-------------+-----------+------------+
5 rows in set (0.01 sec)
```

WL#3634: Recursive WITH (CTE)
https://dev.mysql.com/worklog/task/?id=3634
WL#883: Non-recursive WITH clause (CTE)
https://dev.mysql.com/worklog/task/?id=883

DBT3 Query 15 Top Supplier Query

Using view

```
CREATE VIEW revenue0 (supplier_no, total_revenue) AS
SELECT l_suppkey, SUM(l_extendedprice * (1-l_discount))
FROM  lineitem
WHERE l_shipdate >= '1996-07-01'
    AND l_shipdate < DATE_ADD('1996-07-01',
                INTERVAL '90' day)
GROUP BY  l_suppkey;
```

rewrite

Using CTE

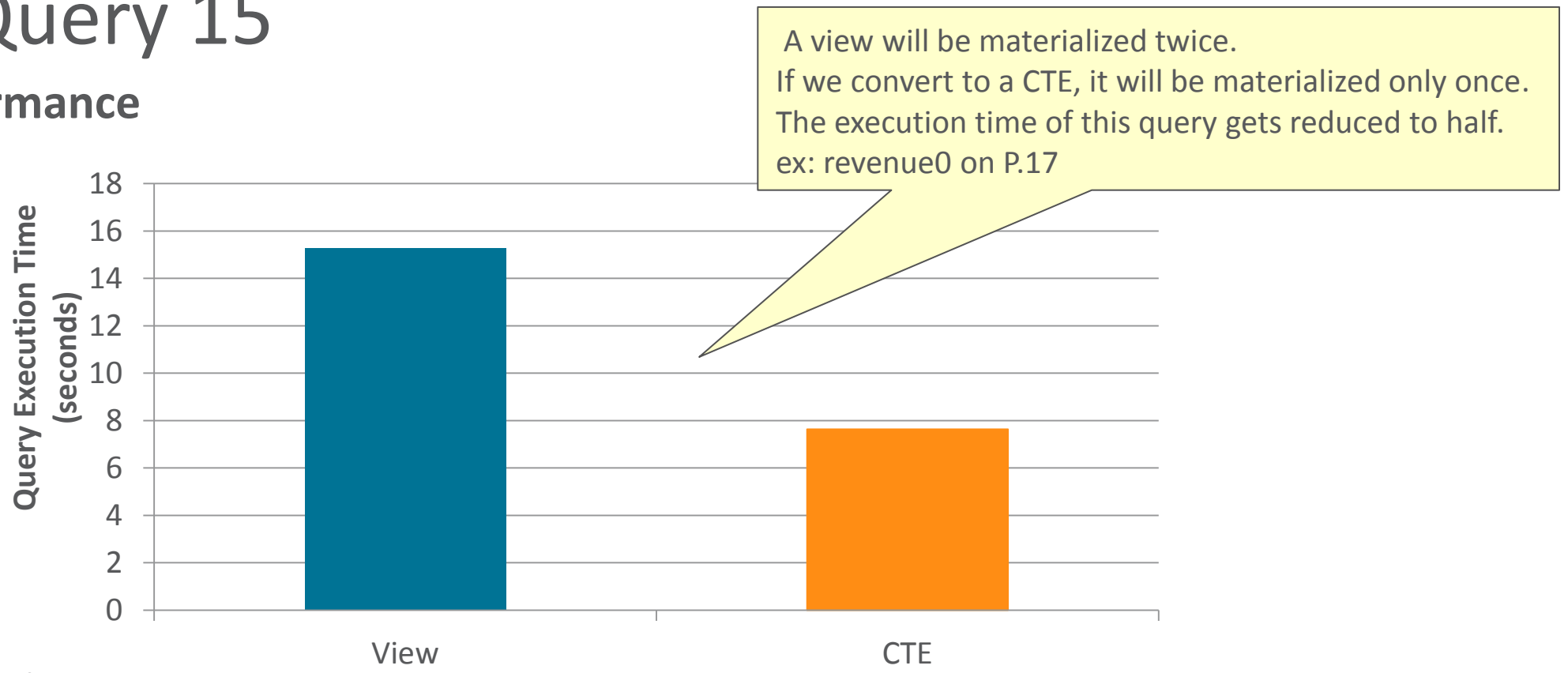
```
WITH revenue0 (supplier_no, total_revenue) AS
(SELECT l_suppkey, SUM(l_extendedprice *  (1-l_discount))
FROM  lineitem
WHERE l_shipdate >= '1996-07-01'
    AND l_shipdate < DATE_ADD('1996-07-01',
                INTERVAL '90' day)
GROUP BY  l_suppkey)
```

```
SELECT s_suppkey, s_name, s_address, s_phone, total_revenue
FROM supplier, revenue0
WHERE s_suppkey = supplier_no AND total_revenue = (SELECT MAX(total_revenue) FROM revenue0)
ORDER BY s_suppkey;
```

http://mysqlserverteam.com/mysql-8-0-labs-recursive-common-table-expressions-in-mysql-ctes/

DBT-3 Query 15

Query Performance



Another Example:

Derived table can not be referenced twice:

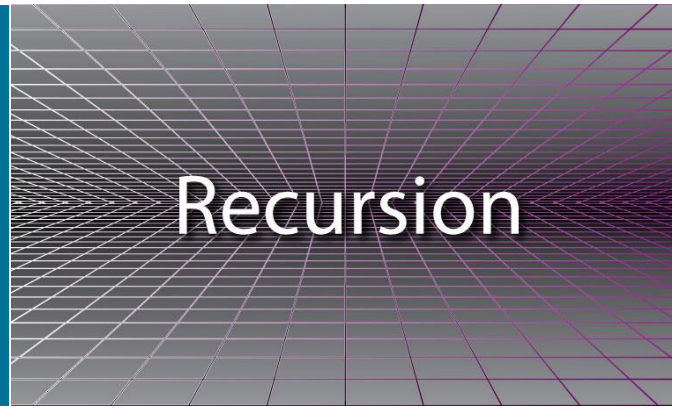
```
SELECT ...FROM (SELECT a, b, SUM(c) s FROM t1 GROUP BY a, b) AS d1 JOIN (SELECT a, b, SUM(c) s FROM t1 GROUP BY a, b) AS d2 ON d1.b = d2.a;
```

CTE can:

```
WITH d AS (SELECT a, b, SUM(c) s FROM t1 GROUP BY a, b)  SELECT ... FROM d AS d1 JOIN d AS d2 ON d1.b = d2.a;
```

Recursive CTE

```
WITH RECURSIVE cte AS
( SELECT ... FROM table_name /* "seed" SELECT */
   UNION ALL
   SELECT ... FROM cte, table_name) /* "recursive"  SELECT */
SELECT ... FROM cte;/
```



- 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

Example: Recursive CTE

Print 1 to 10

```
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

Example: Recursive CTE

Hierarchy Traversal (List reporting chain)

```
mysql> select * from employees;
+------+---------+------------+
| id   | name    | manager_id |
+------+---------+------------+
|   29 | Pedro   |        198 |
|   72 | Pierre  |         29 |
|  123 | Adil    |        692 |
|  198 | John    |        333 |
|  333 | Yasmina |       NULL |
|  692 | Tarek   |        333 |
| 4610 | Sarah   |         29 |
+------+---------+------------+
7 rows in set (0.00 sec)
```

```
mysql> WITH RECURSIVE
    -> emp_ext (id, name, path) AS
    -> (SELECT id, name, CAST(id AS CHAR(200))
    -> FROM employees WHERE manager_id IS NULL
    -> UNION ALL SELECT s.id, s.name,CONCAT(m.path, ",", s.id)
    -> FROM emp_ext m JOIN  employees s ON m.id=s.manager_id )
    -> SELECT * FROM emp_ext ORDER BY path;
+------+---------+-----------------+
| id   | name    | path            |
+------+---------+-----------------+
|  333 | Yasmina | 333             |
|  198 | John    | 333,198         |
|   29 | Pedro   | 333,198,29      |
| 4610 | Sarah   | 333,198,29,4610 |
|   72 | Pierre  | 333,198,29,72   |
|  692 | Tarek   | 333,692         |
|  123 | Adil    | 333,692,123     |
+------+---------+-----------------+
7 rows in set (0.00 sec)
```

Window Functions

SELECT name, department_id, salary,SUM(salary)
OVER (PARTITION BY **department_id**) AS department_total
FROM employee ORDER BY department_id, name;

The **OVER** keyword signals a window function.
PARTITION == disjoint set of rows in result set.

name	department_id	salary	department_total
Ed	10	100000	370000
Fred	10	60000	370000
Lebedev	20	65000	130000
Pete	20	65000	130000
Jeff	30	300000	370000
Will	30	70000	370000

Ranking Windows Functions: RANK, DENSE_RANK, PERCENT_RANK, CUME_DIST, ROW_NUMBER
Analytical Windows Functions: NTILE,LEAD, LAG, FIRST_VALUE, LAST_VALUE, NTH_VALUE
Aggregates Windows Function: COUNT, SUM, AVG, MAX, MIN

https://dev.mysql.com/doc/refman/8.0/en/window-functions-usage.html
https://www.slideshare.net/DagHWanvik/sql-window-functions-for-mysql

WL#9236: Add first batch of SQL window functions
https://dev.mysql.com/worklog/task/?id=9236
WL#9727: Additional aggregate window functions
https://dev.mysql.com/worklog/task/?id=9727
WL#9603: Add remaining non-aggregate window functions
https://dev.mysql.com/worklog/task/?id=9603

Window Functions : OVER

over_clause: {OVER (window_spec) | OVER window_name}

```
mysql> select * from t;
+------+
| i    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
+------+
```

```
mysql> SELECT i,(SELECT SUM(i) FROM t) FROM t;
+------+-------------------------+
| i    | (SELECT SUM(i) FROM t) |
+------+-------------------------+
|    1 |                      10 |
|    2 |                      10 |
|    3 |                      10 |
|    4 |                      10 |
+------+-------------------------+
```

The **OVER** keyword signals a window function. If OVER() is empty, the window consists of all query rows and the window function computes a result using all rows.

```
mysql> SELECT i, SUM(i) OVER() AS sum FROM t;
+------+------+
| i    | sum  |
+------+------+
|    1 |   10 |
|    2 |   10 |
|    3 |   10 |
|    4 |   10 |
+------+------+
```

http://mysqlserverteam.com/mysql-8-0-2-introducing-window-functions/

Window Functions: PARTITION

partition_clause: A PARTITION BY clause indicates how to divide the query rows into groups.

```
mysql> select * from sales;
+----------+------------+------+
| employee | date       | sale |
+----------+------------+------+
| odin     | 2017-03-01 |  200 |
| odin     | 2017-04-01 |  300 |
| odin     | 2017-05-01 |  400 |
| thor     | 2017-03-01 |  400 |
| thor     | 2017-04-01 |  300 |
| thor     | 2017-05-01 |  500 |
+----------+------------+------+
```

PARTITION == disjoint set of rows in result set.
employee (Odin, Thor)

```
mysql> select employee, SUM(sale)
    -> FROM sales GROUP BY employee;
+----------+-----------+
| employee | SUM(sale) |
+----------+-----------+
| odin     |       900 |
| thor     |      1200 |
+----------+-----------+

mysql> select employee,date,sale,SUM(sale)
    -> OVER (PARTITION BY employee) AS sum FROM sales;
+----------+------------+------+------+
| employee | date       | sale | sum  |
+----------+------------+------+------+
| odin     | 2017-03-01 |  200 |  900 |
| odin     | 2017-04-01 |  300 |  900 |
| odin     | 2017-05-01 |  400 |  900 |
| thor     | 2017-03-01 |  400 | 1200 |
| thor     | 2017-04-01 |  300 | 1200 |
| thor     | 2017-05-01 |  500 | 1200 |
+----------+------------+------+------+
```

Window Functions: RANK

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

name	dept_id	salary	rank
Newt	NULL	75000	1
Ed	10	100000	1
Newt	10	80000	2
Fred	10	70000	3
Michael	10	70000	3
Jon	10	60000	5
Dag	10	NULL	6
Pete	20	65000	1
Lebedev	20	65000	1
Jeff	30	300000	1
Will	30	70000	2

Window Functions: ROW_NUMBER

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

name	dept_id	salary	rank	dense	rowno
Newt	NULL	75000	1	1	**1**
Ed	10	100000	1	1	**1**
Newt	10	80000	2	2	**2**
Fred	10	70000	3	3	**3**
Michael	10	70000	3	3	**4**
Jon	10	60000	5	4	**5**
Dag	10	NULL	6	5	**6**
Pete	20	65000	1	1	**1**
Lebedev	20	65000	1	1	**2**
Jeff	30	300000	1	1	**1**
Will	30	70000	2	2	**2**

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

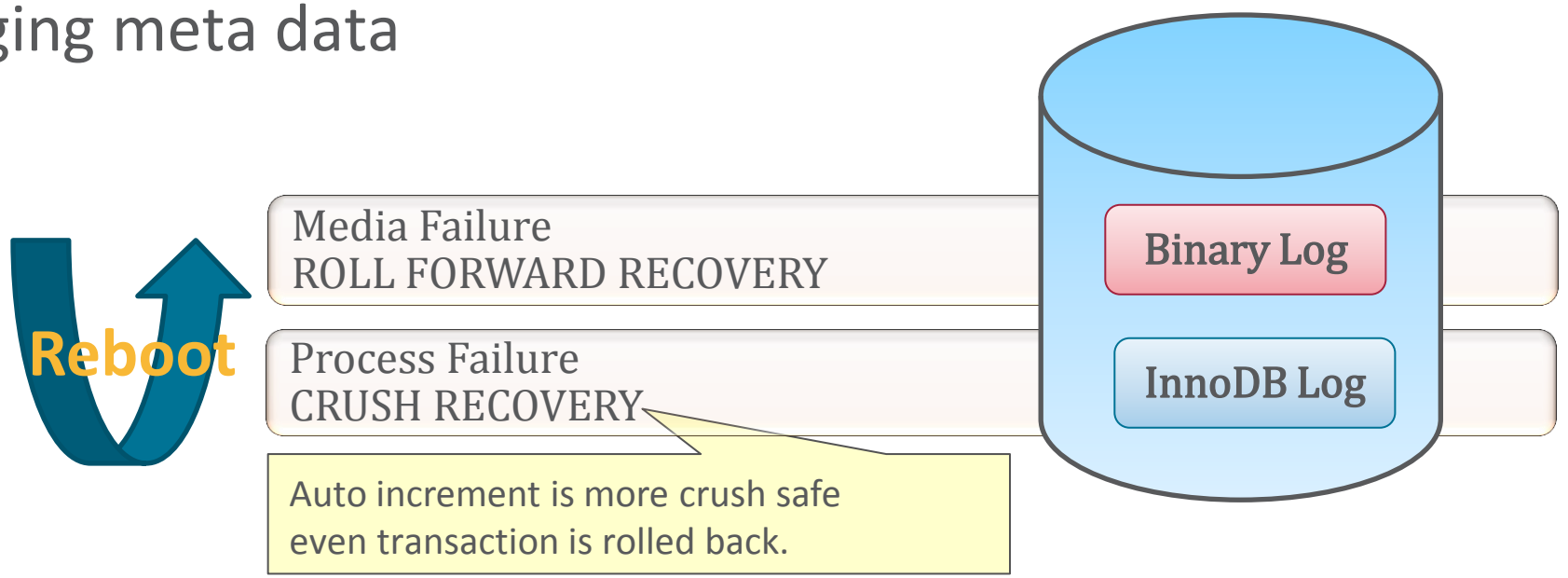
Model accounts for memory fit. For data on disk an IO block read defaults to 1.0. In memory defaults to 0.25.

Much larger performance difference for range scan not in memory (good)

WL#7093: Optimizer provides InnoDB with a bigger buffer
https://dev.mysql.com/worklog/task/?id=7093

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



WL#6204: InnoDB persistent max value for autoinc columns
https://dev.mysql.com/worklog/task/?id=6204

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

WL#1074: Add Descending indexes support
https://dev.mysql.com/worklog/task/?id=1074

EX: Descending Indexes

```
mysql> explain select * from city2 order by city_id asc limit 3;
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key             | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+-------+
|  1 | SIMPLE      | city2 | NULL       | index | NULL          | idx_asc_city_id | 2       | NULL |    3 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+-------+

mysql> explain select * from city2 order by city_id desc limit 3;
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+----------------------+
| id | select_type | table | partitions | type  | possible_keys | key             | key_len | ref  | rows | filtered | Extra                |
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+----------------------+
|  1 | SIMPLE      | city2 | NULL       | index | NULL          | idx_asc_city_id | 2       | NULL |    3 |   100.00 | Backward index scan  |
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+----------------------+
```

MySQL 5.5 ~ 5.7: can create ASC index Only.

```
mysql> explain select * from city2 order by city_id asc limit 3;
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key             | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+-------+
|  1 | SIMPLE      | city2 | NULL       | index | NULL          | idx_asc_city_id | 2       | NULL |    3 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+-----------------+---------+------+------+----------+-------+

mysql> explain select * from city2 order by city_id desc limit 3;
+----+-------------+-------+------------+-------+---------------+------------------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key              | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+------------------+---------+------+------+----------+-------+
|  1 | SIMPLE      | city2 | NULL       | index | NULL          | idx_desc_city_id | 2       | NULL |    3 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+------------------+---------+------+------+----------+-------+
```

MySQL8.0 ~: can create both ASC and DESC index

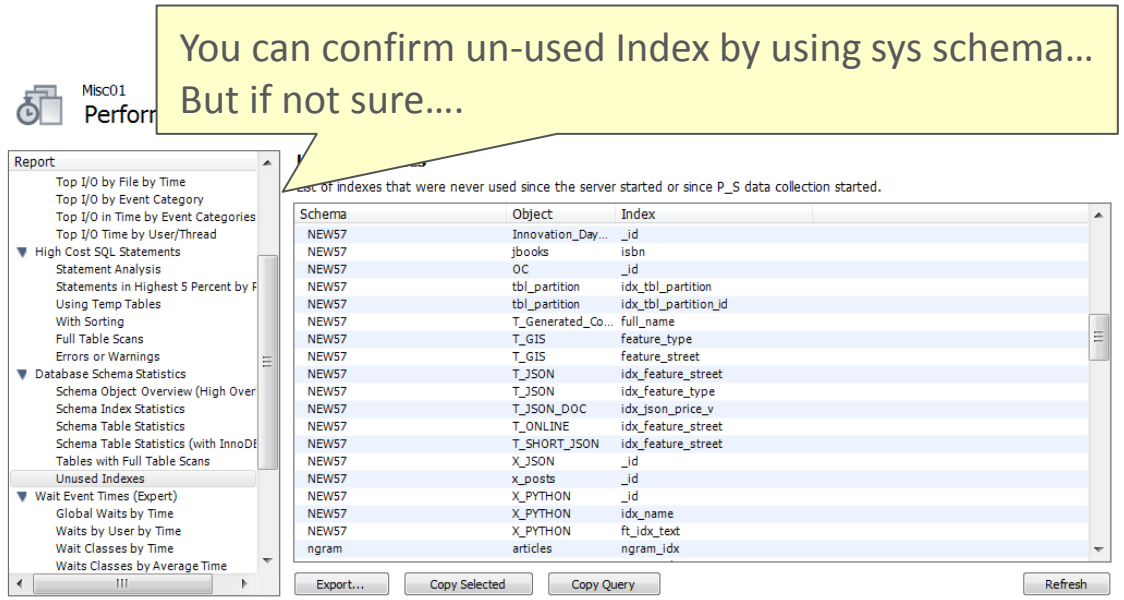
Invisible Indexes

- 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

WL#8697: Support for INVISIBLE indexes
https://dev.mysql.com/worklog/task/?id=8697

Soft Delete

Example Usage



- 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;

https://github.com/mysql/mysql-sys

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

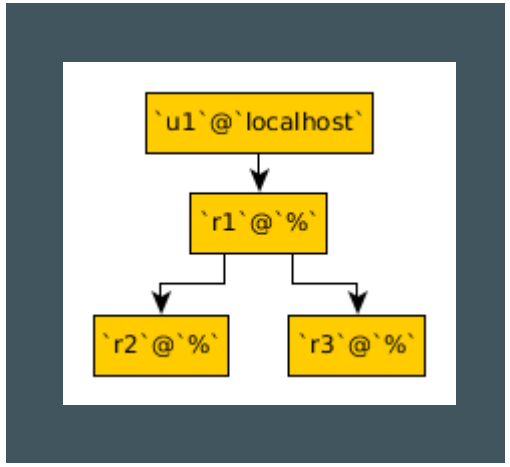
```
ALTER TABLE Country ADD INDEX c (Continent) INVISIBLE;
# after some time
ALTER TABLE Country ALTER INDEX c VISIBLE;
```

```
SELECT * FROM information_schema.statistics WHERE is_visible='NO';
*************************** 1. row ***************************
TABLE_CATALOG: def
 TABLE_SCHEMA: world
   TABLE_NAME: Country
   NON_UNIQUE: 1
 INDEX_SCHEMA: world
   INDEX_NAME: c
 SEQ_IN_INDEX: 1
  COLUMN_NAME: Continent
    COLLATION: A
  CARDINALITY: 7
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE:
   INDEX_TYPE: BTREE
      COMMENT: disabled
INDEX_COMMENT:
   IS_VISIBLE: NO
```

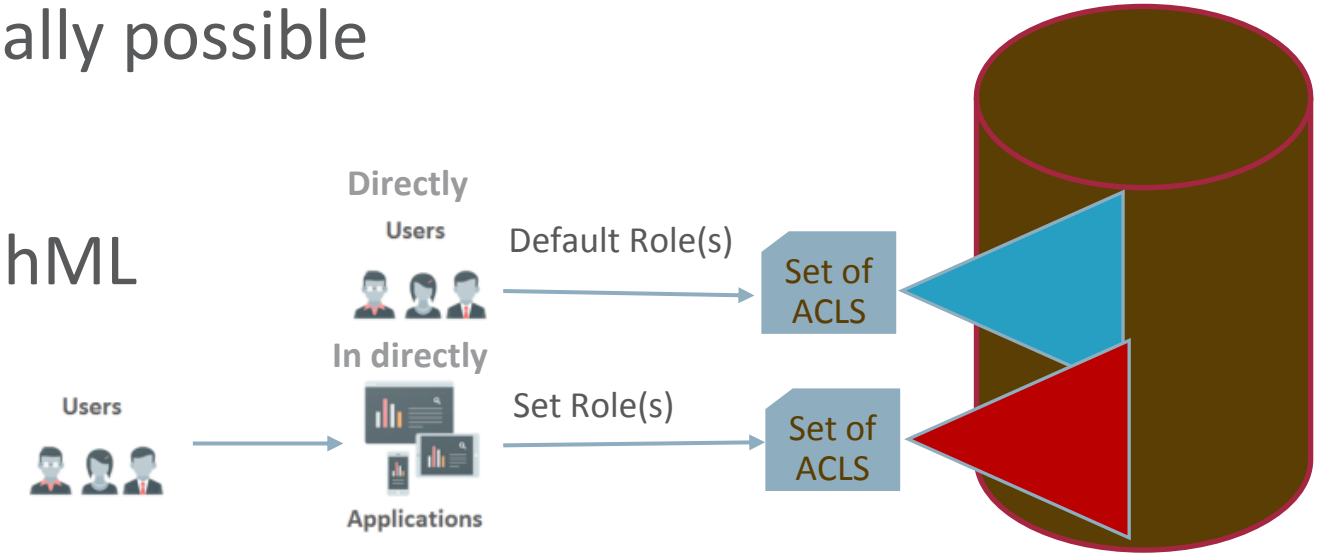
MySQL Roles

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




```
mysql> select user(),current_role();
+------------------+----------------+
| user()           | current_role() |
+------------------+----------------+
| user01@localhost | `role80`@`%`   |
+------------------+----------------+
```



WL#988: Roles
https://dev.mysql.com/worklog/task/?id=988

Atomic ACL Statements

```
+--------------+------------+--------+
| TABLE_SCHEMA | TABLE_NAME | ENGINE |
+--------------+------------+--------+
| mysql        | user       | InnoDB |
+--------------+------------+--------+
```

- Long standing MySQL issue!
  - For Replication, HA, Backups, etc.
- Possible now - ACL tables reside in InnoDB Data Dictionary
- Not just a table operation: memory caches need update too
- Applies to statements performing multiple logical operations, e.g.
  - CREATE USER u1, u2
  - GRANT SELECT ON *.* TO u1, u2
- Uses a custom MDL lock to block ACL related activity
  - While altering the ACL caches and tables

WL#9045: Make user management DDLs atomic
https://dev.mysql.com/worklog/task/?id=9045

InnoDB Redo and Undo Encryption

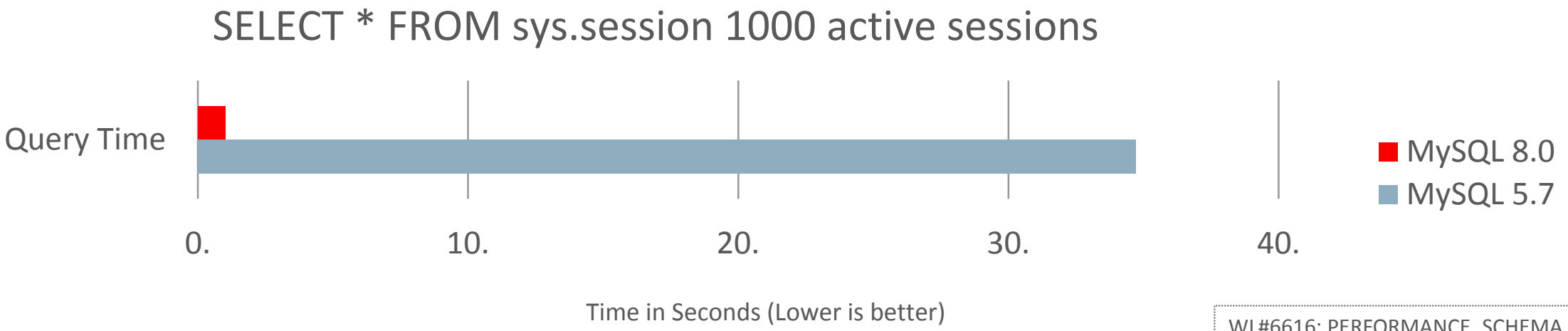
- AES 256 encryption
- *Encrypted when redo/undo log data is written to disk*
- Decryption occurs when redo/undo log data is read from disk
- Once redo/undo log data is read into memory, it is in unencrypted form.
- Two tiered encryption – like Innodb tablepace encryption
  - Fast key rotation, high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

WL#9289: InnoDB: Support Transparent Data Encryption for Undo Tablespaces
https://dev.mysql.com/worklog/task/?id=9289
WL#9290: InnoDB: Support Transparent Data Encryption for Redo Log
https://dev.mysql.com/worklog/task/?id=9290

Performance Schema Indexes

Over 30x faster!

- 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



WL#6616: PERFORMANCE_SCHEMA, INDEXES
https://dev.mysql.com/worklog/task/?id=6616

EX: Performance Schema Indexes

```
mysql> SELECT * FROM variables_by_thread IGNORE INDEX (primary)
WHERE thread_id = 34 AND variable_name = 'time_zone';
 +-----------+---------------+----------------+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE |
+-----------+---------------+----------------+
|        34 | time_zone     | SYSTEM         |
+-----------+---------------+----------------+
1 row in set (0.00 sec)

mysql> show status like 'Handler_read_%';
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| Handler_read_first    | 0     |
| Handler_read_key      | 0     |
| Handler_read_last     | 0     |
| Handler_read_next     | 0     |
| Handler_read_prev     | 0     |
| Handler_read_rnd      | 0     |
| Handler_read_rnd_next | 136   |
+-----------------------+-------+
7 rows in set (0.01 sec)
```

Without Index
"Handler_read_rnd_next"
Is increased.

```
mysql> SELECT * FROM variables_by_thread
WHERE thread_id = 34 AND variable_name = 'time_zone';
+-----------+---------------+----------------+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE |
+-----------+---------------+----------------+
|        34 | time_zone     | SYSTEM         |
+-----------+---------------+----------------+
1 row in set (0.00 sec)

mysql> show status like 'Handler_read_%';
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| Handler_read_first    | 0     |
| Handler_read_key      | 1     |
| Handler_read_last     | 0     |
| Handler_read_next     | 0     |
| Handler_read_prev     | 0     |
| Handler_read_rnd      | 0     |
| Handler_read_rnd_next | 0     |
+-----------------------+-------+
7 rows in set (0.01 sec)
```

With Index
"Handler_read_rnd_next"
is maintain "0" in this case.

Handler_read_rnd_next: The number of requests to read the next row in the data file. This value is high if you are doing a lot of table scans.

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

```
mysql> select * from performance_schema.events_errors_summary_by_host_by_error where FIRST_SEEN is not NULL;
+-----------+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+
| HOST      | ERROR_NUMBER | ERROR_NAME                | SQL_STATE | SUM_ERROR_RAISED   | SUM_ERROR_HANDLED  | FIRST_SEEN          | LAST_SEEN           |
+-----------+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+
| NULL      |         1045 | ER_ACCESS_DENIED_ERROR    | 28000     |                  1 |                  0 | 2016-10-27 15:57:16 | 2016-10-27 15:57:16 |
| localhost |         1046 | ER_NO_DB_ERROR            | 3D000     |                  1 |                  0 | 2016-10-27 16:00:37 | 2016-10-27 16:00:37 |
| localhost |         1049 | ER_BAD_DB_ERROR           | 42000     |                  1 |                  0 | 2016-10-27 18:21:09 | 2016-10-27 18:21:09 |
| localhost |         1064 | ER_PARSE_ERROR            | 42000     |                 15 |                  0 | 2016-10-27 15:58:01 | 2016-10-27 18:24:06 |
| localhost |         1146 | ER_NO_SUCH_TABLE          | 42S02     |                  2 |                  0 | 2016-10-27 16:08:03 | 2016-10-27 18:14:41 |
| localhost |         1287 | ER_WARN_DEPRECATED_SYNTAX | HY000     |                 24 |                  0 | 2016-10-27 16:07:10 | 2016-10-27 16:07:10 |
| localhost |         3554 | ER_NO_SYSTEM_TABLE_ACCESS | HY000     |                140 |                  0 | 2016-10-27 15:57:30 | 2016-10-27 18:38:09 |
+-----------+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+

mysql> select * from performance_schema.events_errors_summary_by_user_by_error where FIRST_SEEN is not NULL;
+------+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+
| USER | ERROR_NUMBER | ERROR_NAME                | SQL_STATE | SUM_ERROR_RAISED   | SUM_ERROR_HANDLED  | FIRST_SEEN          | LAST_SEEN           |
+------+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+
| NULL |         1045 | ER_ACCESS_DENIED_ERROR    | 28000     |                  1 |                  0 | 2016-10-27 15:57:16 | 2016-10-27 15:57:16 |
| root |         1046 | ER_NO_DB_ERROR            | 3D000     |                  1 |                  0 | 2016-10-27 16:00:37 | 2016-10-27 16:00:37 |
| root |         1049 | ER_BAD_DB_ERROR           | 42000     |                  1 |                  0 | 2016-10-27 18:21:09 | 2016-10-27 18:21:09 |
| root |         1064 | ER_PARSE_ERROR            | 42000     |                 15 |                  0 | 2016-10-27 15:58:01 | 2016-10-27 18:24:06 |
| root |         1146 | ER_NO_SUCH_TABLE          | 42S02     |                  2 |                  0 | 2016-10-27 16:08:03 | 2016-10-27 18:14:41 |
| root |         1287 | ER_WARN_DEPRECATED_SYNTAX | HY000     |                 24 |                  0 | 2016-10-27 16:07:10 | 2016-10-27 16:07:10 |
| root |         3554 | ER_NO_SYSTEM_TABLE_ACCESS | HY000     |                140 |                  0 | 2016-10-27 15:57:30 | 2016-10-27 18:38:09 |
+------+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+

mysql> SELECT * FROM performance_schema.events_errors_summary_global_by_error WHERE sum_error_handled > 0 OR SUM_ERROR_RAISED > 0;
+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+
| ERROR_NUMBER | ERROR_NAME                | SQL_STATE | SUM_ERROR_RAISED   | SUM_ERROR_HANDLED  | FIRST_SEEN          | LAST_SEEN           |
+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+
|         1049 | ER_BAD_DB_ERROR           | 42000     |                  3 |                  0 | 2017-07-07 14:35:47 | 2017-07-07 14:36:49 |
|         1054 | ER_BAD_FIELD_ERROR        | 42S22     |                  1 |                  0 | 2017-07-07 08:20:04 | 2017-07-07 08:20:04 |
|         1062 | ER_DUP_ENTRY              | 23000     |                  1 |                  0 | 2017-07-07 13:30:58 | 2017-07-07 13:30:58 |
|         1064 | ER_PARSE_ERROR            | 42000     |                  6 |                  0 | 2017-07-07 07:49:59 | 2017-07-07 14:36:08 |
|         1146 | ER_NO_SUCH_TABLE          | 42S02     |                  3 |                  0 | 2017-07-07 13:30:38 | 2017-07-07 14:37:33 |
|         1287 | ER_WARN_DEPRECATED_SYNTAX | HY000     |                  7 |                  0 | 2017-07-07 11:41:03 | 2017-07-07 13:39:42 |
|         1305 | ER_SP_DOES_NOT_EXIST      | 42000     |                  4 |                  0 | 2017-07-07 12:44:54 | 2017-07-07 13:30:11 |
|         1411 | ER_WRONG_VALUE_FOR_TYPE   | HY000     |                  9 |                  0 | 2017-07-07 12:45:00 | 2017-07-07 12:45:16 |
|         3568 | ER_UNRESOLVED_TABLE_LOCK  | HY000     |                  4 |                  0 | 2017-07-07 13:44:45 | 2017-07-07 13:46:28 |
+--------------+---------------------------+-----------+--------------------+--------------------+---------------------+---------------------+
```

Histograms

- More consistent query execution for cases when data is skewed
- Lower cost to maintain than an index
- ANALYZE TABLE t UPDATE HISTOGRAM ON c1 WITH 10 BUCKETS;

```
mysql> SELECT * FROM events_statements_histogram_by_digest  WHERE SCHEMA_NAME ='sakila'
    -> AND DIGEST = 'a5980f0634db05c87a7aeb17e1344f84'  AND COUNT_BUCKET > 0 limit 1¥G
*************************** 1. row ***************************
        SCHEMA_NAME: sakila
             DIGEST: a5980f0634db05c87a7aeb17e1344f84
      BUCKET_NUMBER: 153
  BUCKET_TIMER_LOW: 10964781961
 BUCKET_TIMER_HIGH: 11481536214
       COUNT_BUCKET: 1
COUNT_BUCKET_AND_LOWER: 1
    BUCKET_QUANTILE: 0.500000
```

Statement Histogram Summary Tables
these values indicate that 50% of queries run in under 11.48 microseconds:

WL#8706: Persistent storage of Histogram data
https://dev.mysql.com/worklog/task/?id=8706
WL#8707: Classes/structures for Histograms
https://dev.mysql.com/worklog/task/?id=8707
WL#8943: Extend ANALYZE TABLE with histogram support
https://dev.mysql.com/worklog/task/?id=8943

Performance Schema Histograms

Showing distribution of query time from a run of mysqlslap

```
mysql> WITH total as
    ->  (SELECT SUM(count_bucket) as t FROM performance_schema.events_statements_histogram_global)
    -> SELECT sys.decimal_bucket_name(sys.decimal_bucket(MIN(bucket_timer_low))) as bucket,
    ->  sys.visualization(SUM(count_bucket) / (SELECT t FROM total), 50) as visualization,
    ->  SUM(count_bucket) as count
    -> FROM performance_schema.events_statements_histogram_global
    -> GROUP BY sys.decimal_bucket(bucket_timer_low);
+---------+----------------------------------+--------+
| bucket  | visualization                    | count  |
+---------+----------------------------------+--------+
| 0us+    | ################################ | 439148 |
| 10us+   | ############                     | 163434 |
| 100us+  |                                  |   1517 |
| 1ms+    | #######                          |  99453 |
| 10ms+   |                                  |    369 |
| 100ms+  |                                  |     29 |
| 1s+     |                                  |      2 |
| 10s+    |                                  |      1 |
+---------+----------------------------------+--------+
8 rows in set (0.04 sec)
```

Generated with a quick CTE over events_statements_histogram_global

WL#5384: PERFORMANCE_SCHEMA Histograms
https://dev.mysql.com/worklog/task/?id=5384

Performance Schema Histograms (cont.)

```
 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)
```

- which data is locked,
- who owns the lock,
- who waits for the data.

WL#6657: PERFORMANCE_SCHEMA, DATA LOCKS
https://dev.mysql.com/worklog/task/?id=6657
WL#9275: DEPRECATE INFORMATION_SCHEMA.INNODB_LOCKS IN 5.7
https://dev.mysql.com/worklog/task/?id=9275

Persist Configuration

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

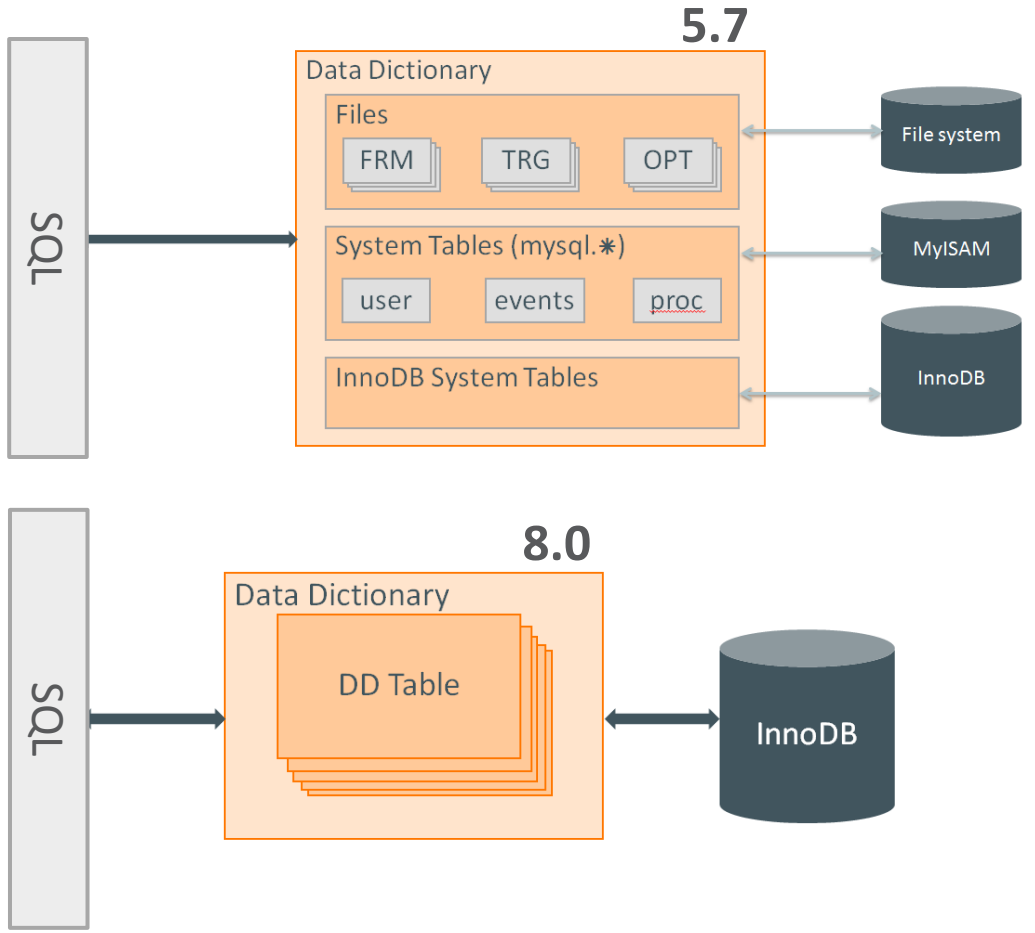
```
#cat /var/lib/mysql/mysqld-auto.cnf
{ "mysql_server": {"log_timestamps": "SYSTEM" } }
```

```
mysql> SET PERSIST log_timestamps='SYSTEM';
Query OK, 0 rows affected (0.00 sec)

mysql> select * from performance_schema.variables_info where variable_source='PERSISTED';
+----------------+-----------------+--------------------------------+-----------+-----------+---------------------+----------+----------+
| VARIABLE_NAME  | VARIABLE_SOURCE | VARIABLE_PATH                  | MIN_VALUE | MAX_VALUE | SET_TIME            | SET_USER | SET_HOST |
+----------------+-----------------+--------------------------------+-----------+-----------+---------------------+----------+----------+
| log_timestamps | PERSISTED       | /var/lib/mysql/mysqld-auto.cnf | 0         | 0         | 2017-07-14 14:48:28 |          |          |
+----------------+-----------------+--------------------------------+-----------+-----------+---------------------+----------+----------+
```

WL#8688: Support ability to persist SET GLOBAL settings
https://dev.mysql.com/worklog/task/?id=8688

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
- No longer contain MyISAM tables




```
[root@DockerHost oracle]# ls -l /docker/docker802/world
total 1084
-rw-r----- 1 27 27 638976 Jul 18 01:25 city.ibd
-rw-r----- 1 27 27 196608 Jul 18 01:25 country.ibd
-rw-r----- 1 27 27 262144 Jul 18 01:25 countrylanguage.ibd
```

8.0

WL#6379: Schema definitions for new DD
https://dev.mysql.com/worklog/task/?id=6379
WL#6392: Upgrade to Transactional Data Dictionary
https://dev.mysql.com/worklog/task/?id=6392
WL#6394: Bootstrap code for new DD
https://dev.mysql.com/worklog/task/?id=6394 and more

Transactional Data Dictionary

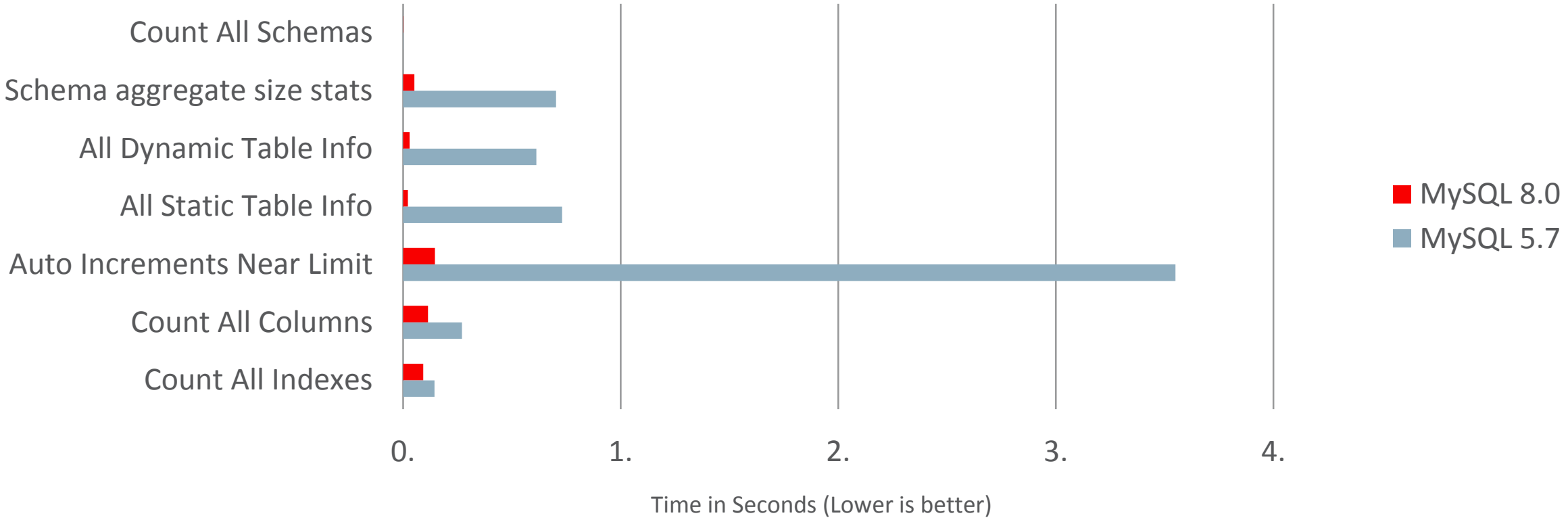
- Better cross-platform experience
  - No dependencies on filesystem semantics
- Atomic DDL
  - Better Replication, Simplifies server edge cases
- MDL(Metadata Lock) for Foreign Keys
- Flexible Metadata API
  - Easier path to adding new features

Ver	Delete Tables	Delete Stored Programs	Delete Schema	ATOMICITY
5.7	- Metadata, TRN/TRG/FRM files - Data, InnoDB tables	- Metadata, rows in MyISAM (non-transactional)	- Metadata, DB.OPT file	Mix of filesystem, non-transactional/transactional storage and multiple commits
8.0	- Metadata, rows in InnoDB - Data, InnoDB tables	- Metadata, rows in InnoDB	- Metadata, rows in InnoDB	Updates to transactional storage, one commit

Information Schema Performance

100 schemas times 50 tables (5000 tables)

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



30x Faster

```
mysql> SELECT TABLE_SCHEMA,TABLE_NAME,TABLE_TYPE,ENGINE,ROW_FORMAT
    -> FROM information_schema.tables WHERE TABLE_SCHEMA LIKE ‘db%';
+--------------+-------------+------------+--------+------------+
| TABLE_SCHEMA | TABLE_NAME  | TABLE_TYPE | ENGINE | ROW_FORMAT |
+--------------+-------------+------------+--------+------------+
| db1000       | T_PARTITION | BASE TABLE | InnoDB | Dynamic    |
| db1000       | actor       | BASE TABLE | InnoDB | Dynamic    |
| db1000       | actor2      | BASE TABLE | InnoDB | Dynamic    |
| db1000       | actor_info  | VIEW       | NULL   | NULL       |
| db1000       | address     | BASE TABLE | InnoDB | Dynamic    |

<SNIP>
```

Test Performed with 100 schemas, each with 50 tables.

Memcached Enhancement (Range Query)

Support of "multiple get" operation

- Improve the read performance.
- User can fetch multiple key value pairs in a single memcached query.
- Reduce communication traffic between client and server.

Add support for "range queries"

With Range queries, user can specify a particular range,
and fetch all the qualified values in this range.

```
get A B C D
get @>B
get @<D
get @<=D
get @>B@<D
```

```
+---------+-----------+-------------+---------+
| city_id | name      | state       | country |
+---------+-----------+-------------+---------+
| A       | BANGALORE | BANGALORE   | IN      |
| B       | CHENNAI   | TAMIL NADU  | IN      |
| C       | DELHI     | DELHI       | IN      |
| D       | HYDERABAD | TELANGANA   | IN      |
+---------+-----------+-------------+---------+
```

WL#6650: InnoDB_Memcached: support multiple get and range search
https://dev.mysql.com/worklog/task/?id=6650

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

Cost Model Improvements

Scalability Improvements

Atomicity in Privileges

Dynamic Privileges

Parser Refactoring

Improvements to Temporary Tables

C++11 and Toolchain Improvements

Improve undo tablespace management

- Replication Applier Thread Progress Reports
- GTID_PURGED always settable
- Improved Parallel Replication
- Multi-Source Replication Filters
- Multi-Source Replication Filters In Performance Schema
- Binary Log Expiration Period in Seconds
- SQL Grouping Function
- Optimizer Trace detailed sort statistics
- Smaller Package Downloads
- JSON performance improvements
- Expanded Query Hints
- Improved usability of cost constant configuration

http://mysqlserverteam.com/the-mysql-8-0-0-milestone-release-is-available/
http://mysqlhighavailability.com/replication-features-in-mysql-8-0-2/
http://www.unofficialmysqlguide.com/index.html

Thank you!!

“Please Enjoy Upcoming New Features. We hope it will support your business.”

Appendix: Just In case…

Oracle Lifetime Support for MySQL

Features	Premier (Years 1-5)	Extended (Years 6-8)	Sustain (Years 9+)
24x7 Support	•	•	•
Unlimited Support Incidents	•	•	•
Knowledge Base	•	•	•
Maintenance Releases, Bug Fixes, Patches, Updates	•	•	Pre-Existing only
MySQL Consultative Support	•	•	•

Do you want to reduce TCO?
Need remote DBA?

Do you want to save more time and focus on service development?

https://www.mysql.com/support/
https://www.mysql.com/support/consultative.html

Concern about security? Require useful tools? Please check our products.
https://www.mysql.com/products/
https://www.mysql.com/products/enterprise/

Do you require Comprehensive security solution?

Health Check (Check your MySQL Environment)
https://www.mysql.com/news-and-events/health-check/

Integrated Cloud

Applications & Platform Services


ORACLE®