# MySQL開発最新動向
## MySQL 8.0、MySQL InnoDB Clusterなどのご紹介

updated : 2018/01/26


Yoshiaki Yamasaki / 山﨑 由章

MySQL Global Business Unit
MySQL Senior Sales Consultant

# Safe Harbor Statement

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。
また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメントするものではない為、購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

# アジェンダ

1. Oracle MySQL Cloud Service
2. MySQL 8.0 RC 新機能
3. MySQL Group Replication、MySQL InnoDB Cluster
4. MySQL Enterprise Edition
5. 参考情報

# アジェンダ

1. **Oracle MySQL Cloud Service**
2. **MySQL 8.0 RC 新機能**
3. **MySQL Group Replication、MySQL InnoDB Cluster**
4. **MySQL Enterprise Edition**
5. **参考情報**

# Oracle MySQLクラウドサービスによる TCOの最適化


ORACLE


最高レベルのセキュリティ

スケーラビリティと可用性

MySQLエキスパート
テクニカルサポート

Oracleクラウド環境へ統合

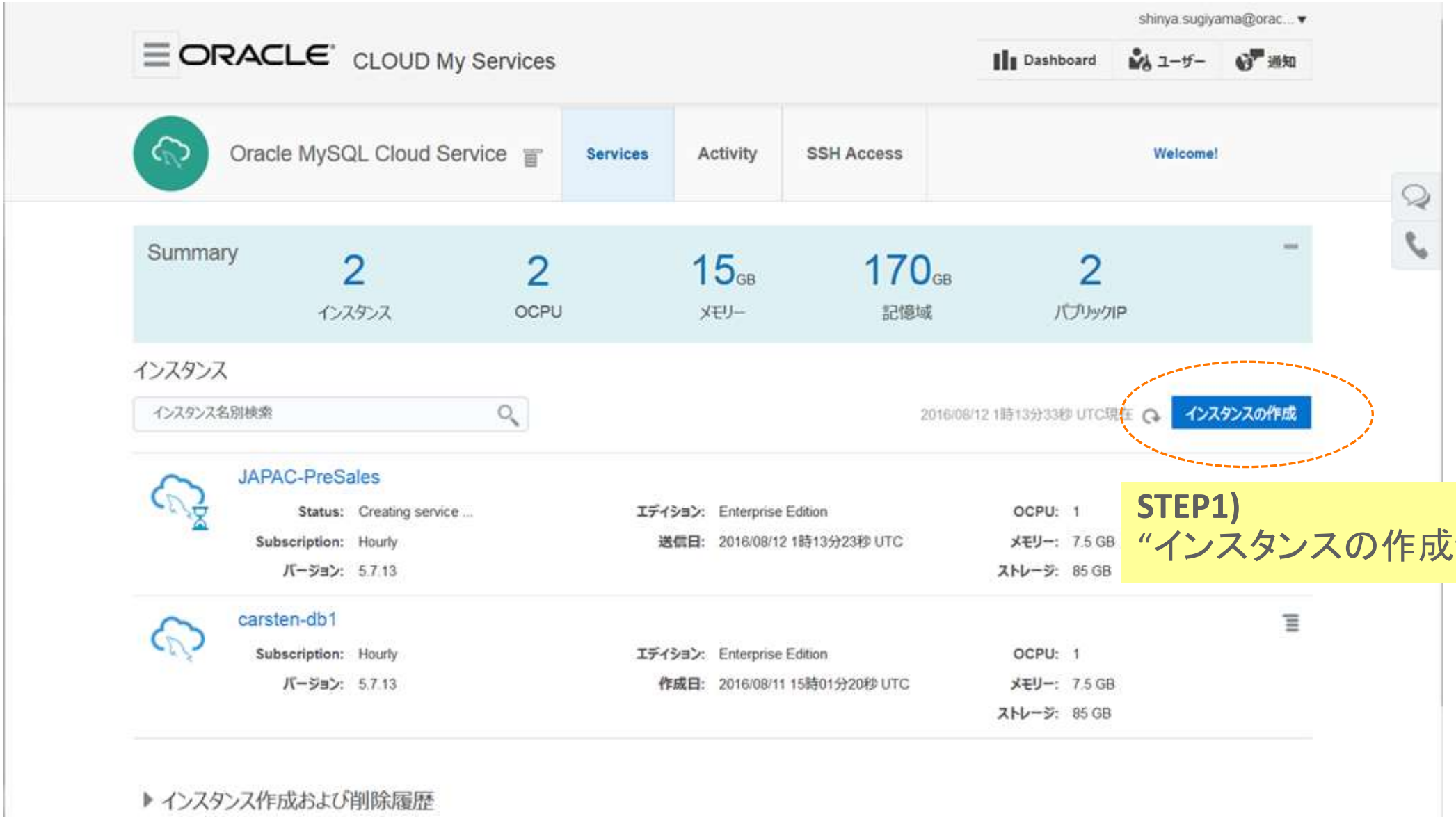
ハイブリッドにデプロイ可能
クラウド& オンプレミス

TCOの削減

ORACLE

# MySQL Cloud Service: 価値提案


- シンプル
  - わずか数回のクリックで,素早くMySQLデータベース・インスタンスが利用可能。
- 自動化
  - データベース管理を自動化するツールで簡単にMySQLを管理する事が可能。
- 統合
  - 迅速な開発と展開の為に、Oracleクラウドサービスとの統合
- エンタープライズ対応
  - パフォーマンス、セキュリティ&アップタイム用のOracleの実証済みのMySQLエンタープライズ・エディションを標準提供。




**STEP1)**
“インスタンスの作成”をクリック

Provide details for this Oracle MySQL Cloud Service [Enterprise Edition] instance.

Selection Summary

## Service Configuration

* Service Name: JAPACSC01
Service Description: New Release Evaluation
* SSH Public Key: ssh-rsa AAAAB3NzaC1yc2EAA 編集
* Compute Shape: OC3 - 1.0 OCPU, 7.5GB RAM

## Backup and Recovery Configuration

Backup Destination: Both Cloud and Disk Storage
* Cloud Storage Container: Storage-mysqlsc/JAPACSC01
* Cloud Storage User Name: shinya.sugiyama@oracle.com
* Cloud Storage Password: ••••••••••
Create Cloud Storage Container: ☑

## Configuration

* Usable Database Storage (GB): 25
* Administration User: root
* Administration Password: ••••••
* Confirm Administration Password: ••••••
* Database Schema Name: JAPAC
Configure Enterprise Monitor: Yes
* Manager User: mem_admin
* Manager Password: •••••••••
* Confirm Manager Password: •••••••••
* Agent User: mem_agent
* Agent Password: •••••••••
* Confirm Agent Password: •••••••••

**STEP2)**
ホスト名を入力しカタログからサーバータイプを選択し作成。必要に応じてObject Storage, MySQL Enterprise Monitorの設定を入力し完了。

作成時間: 約10分

**Restart Service Completed**
**Service Name:** JAPAC-PreSales
**Operation:** Restart Service
**Status:** Succeeded
**Start Time:** 2016/08/12 7時46分39秒 UTC
**End Time:** 2016/08/12 7時52分37秒 UTC

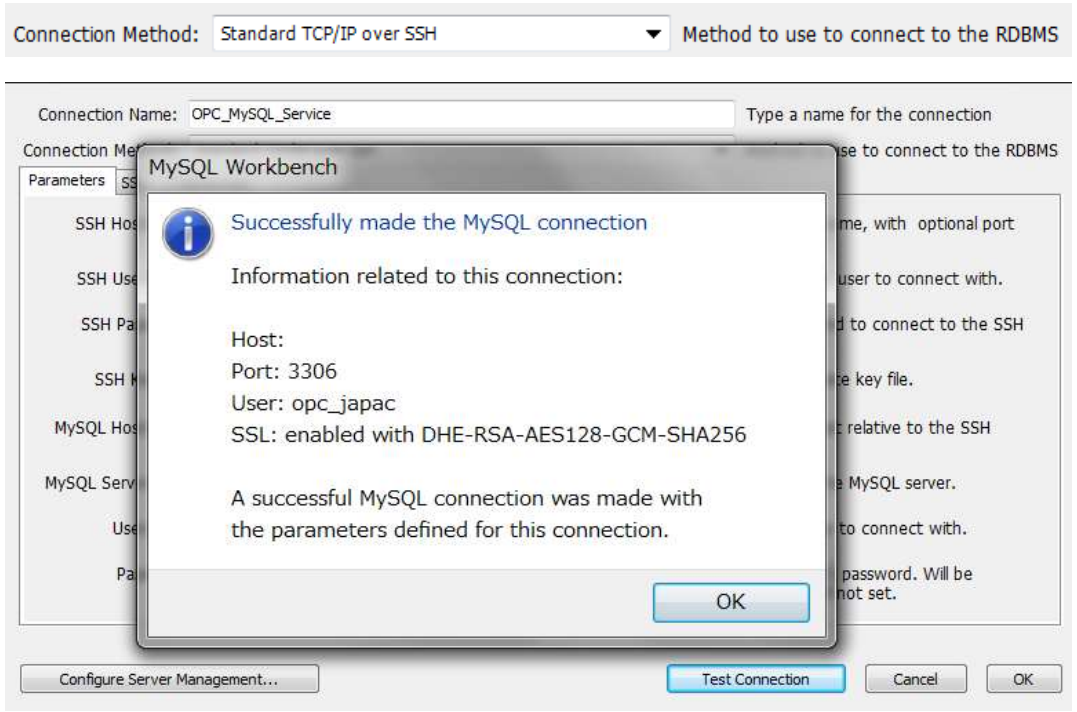
**Create Service Completed**
**Service Name:** JAPAC-PreSales
**Operation:** Create Service
**Status:** Succeeded
**Start Time:** 2016/08/12 1時13分23秒 UTC
**End Time:** 2016/08/12 1時21分01秒 UTC

**STEP3)**
Public IPが設定されているので、アサインされたIPに対して鍵認証でログインする事が可能。

## SSHを利用した接続

```
**********************************************************************************************
*                                    Welcome to                                              *
*                                  MySQL Cloud Service                                       *
*                                       by                                                   *
*                                     Oracle                                                 *
*          If you are an unauthorised user please disconnect IMMEDIATELY                     *
**********************************************************************************************
********************************** MySQL Information **********************************
* Status:  RUNNING                                                                           *
* Version:   5.7.13                                                                          *
**********************************************************************************************
************************** Storage Volume Information **************************
* Volume      Used             Use%          Available   Size     Mounted on    *
* MySQLlog    6.3G -------- 34%                  13G    20G   /u01/translog *
* bin         2.6G ------- 28%                 6.7G   9.8G   /u01/bin      *
* data        151M -- 1%                        24G    25G   /u01/data     *
**********************************************************************************************
[opc@japac-presales-mysql-1 ~]$
```

## Workbench経由でのSSH接続



```
mysql> select PLUGIN_NAME,PLUGIN_STATUS,PLUGIN_TYPE,LOAD_OPTION from PLUGINS
    -> where PLUGIN_TYPE <> 'INFORMATION SCHEMA';
+----------------------+---------------+--------------------+----------------------+
| PLUGIN_NAME          | PLUGIN_STATUS | PLUGIN_TYPE        | LOAD_OPTION          |
+----------------------+---------------+--------------------+----------------------+
| binlog               | ACTIVE        | STORAGE ENGINE     | FORCE                |
| mysql_native_password | ACTIVE        | AUTHENTICATION     | FORCE                |
| sha256_password      | ACTIVE        | AUTHENTICATION     | F                    |
| InnoDB               | ACTIVE        | STORAGE ENGINE     | F                    |
| PERFORMANCE_SCHEMA   | ACTIVE        | STORAGE ENGINE     | F                    |
| MRG_MYISAM           | ACTIVE        | STORAGE ENGINE     | F                    |
| MyISAM               | ACTIVE        | STORAGE ENGINE     | FORCE                |
| MEMORY               | ACTIVE        | STORAGE ENGINE     | FORCE                |
| CSV                  | ACTIVE        | STORAGE ENGINE     | FORCE                |
| BLACKHOLE            | DISABLED      | STORAGE ENGINE     | OFF                  |
| partition            | ACTIVE        | STORAGE ENGINE     | ON                   |
| FEDERATED            | DISABLED      | STORAGE ENGINE     | OFF                  |
| ARCHIVE              | DISABLED      | STORAGE ENGINE     | OFF                  |
| ngram                | ACTIVE        | FTPARSER           | ON                   |
| audit_log            | ACTIVE        | AUDIT              | FORCE_PLUS_PERMANENT |
| thread_pool          | ACTIVE        | DAEMON             | ON                   |
| authentication_pam   | ACTIVE        | AUTHENTICATION     | ON                   |
| auth_socket          | ACTIVE        | AUTHENTICATION     | ON                   |
| validate_password    | ACTIVE        | VALIDATE PASSWORD  | ON                   |
+----------------------+---------------+--------------------+----------------------+
```

MySQL Enterprise版のバイナリーがインストール済みの為、Enterprise版の機能が利用する事が可能。

Backup | Patching

2016/08/30 6時57分34秒 UTC現在

Summary

| 0 MB | 0 MB | 0 % |
|---|---|---|
| Storage Cloud Volume Used | Backup Volume Used | Backup Volume Percent Used |
| Daily at 16時15分00秒 UTC | Every Tuesday at 16時15分00秒 UTC | Not Available |
| Incremental Backups | Full Backups | Last Successful Backup |

MySQL Enterprise Backupも実装されていて、
DashboardからバックアップジョブⅠ設定、
Point In Timeリカバリー含めて管理可能。

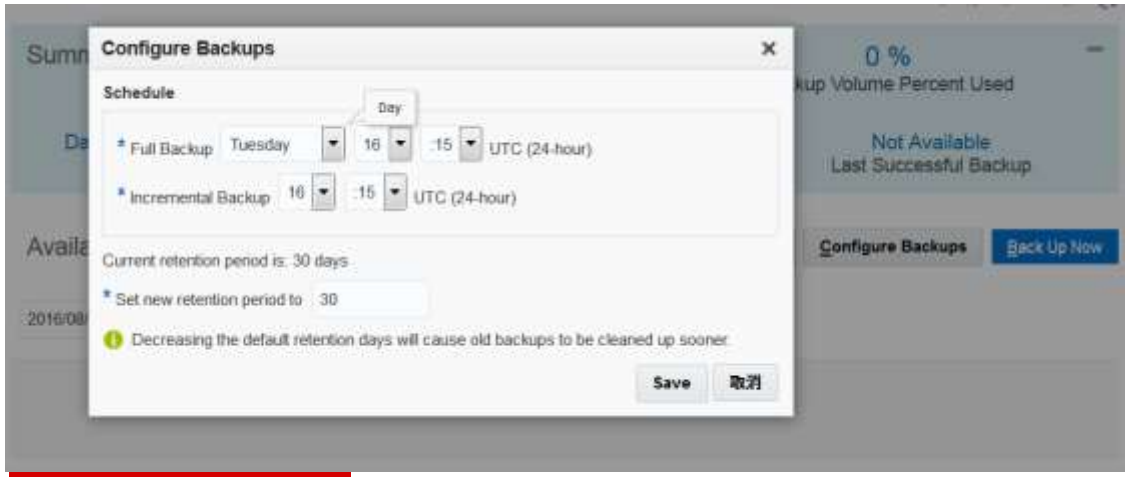
Available Backups

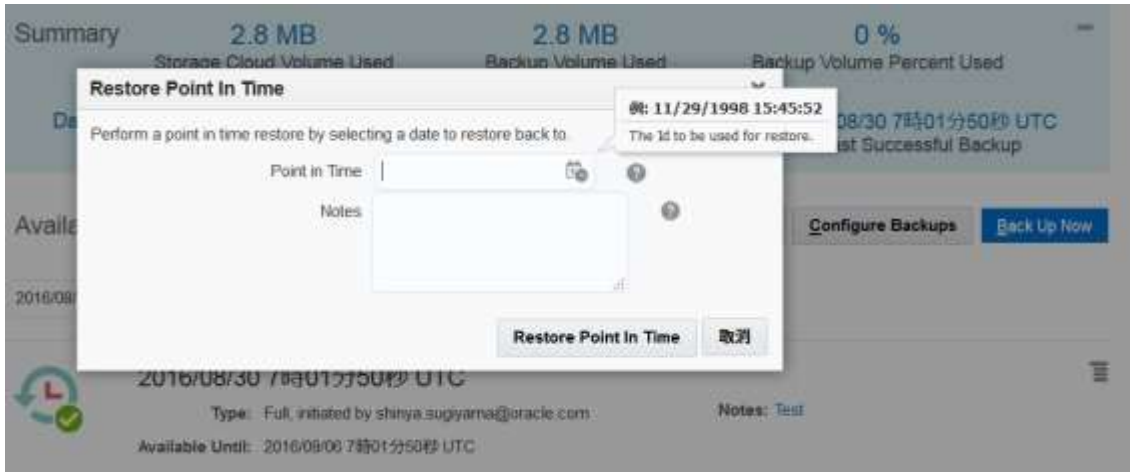
Restore Point In Time | Configure Backups | Back Up Now

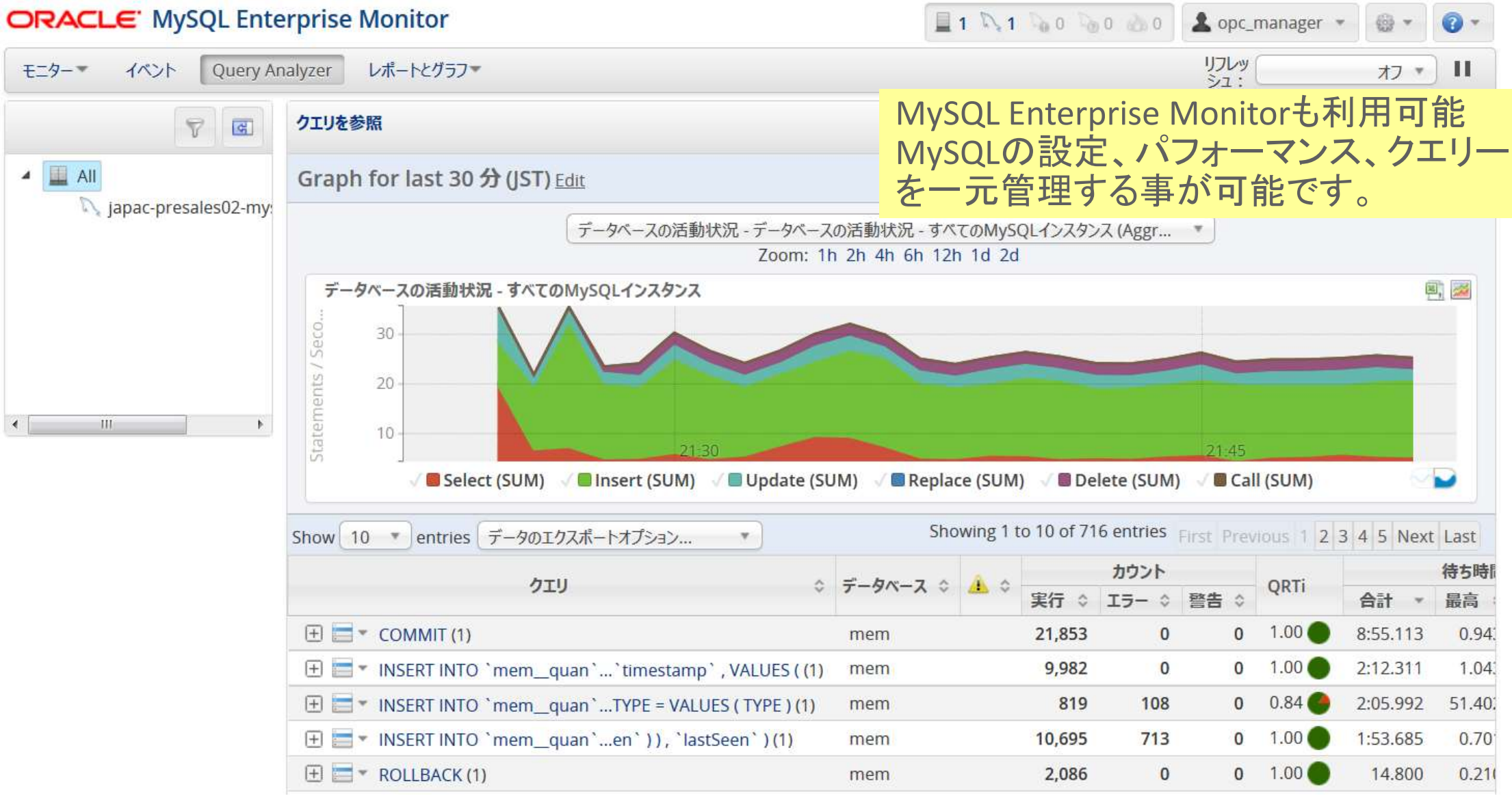
2016/08/23 to Enter end date

*No backups available.*

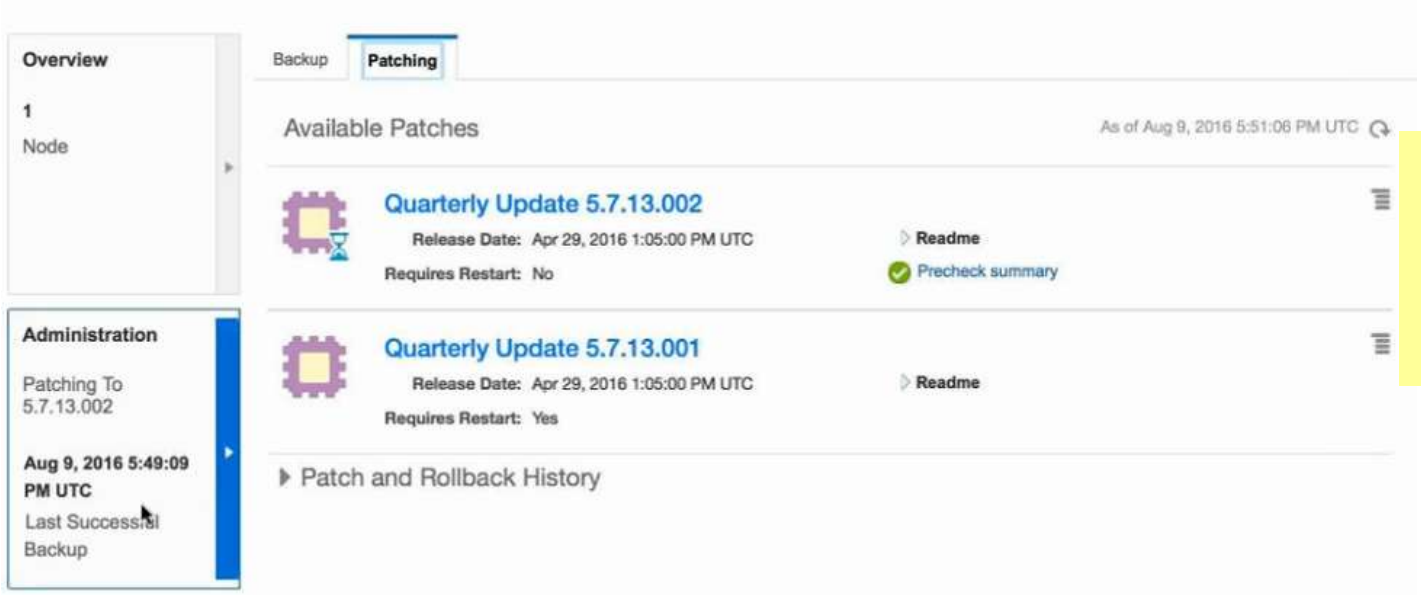
Restore History

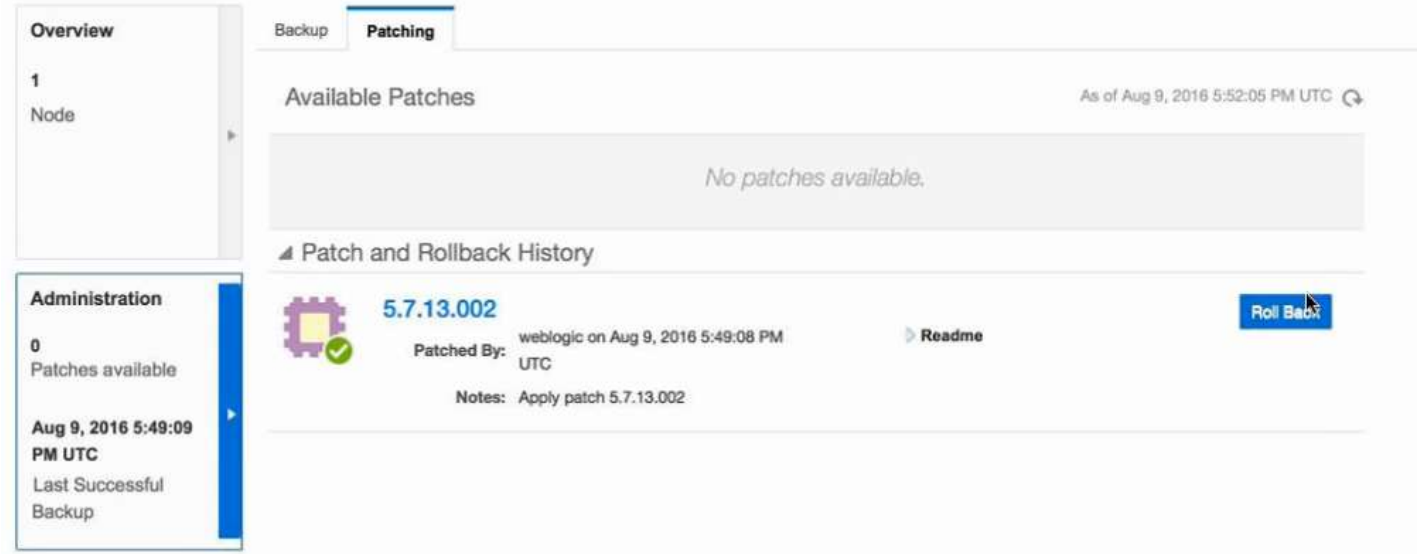





MySQL Enterprise Monitorも利用可能
MySQLの設定、パフォーマンス、クエリー等を一元管理する事が可能です。




Dashboardから、MySQLのパッチ適用、適用前の事前検証と適用後のロールバックを実施する事が可能

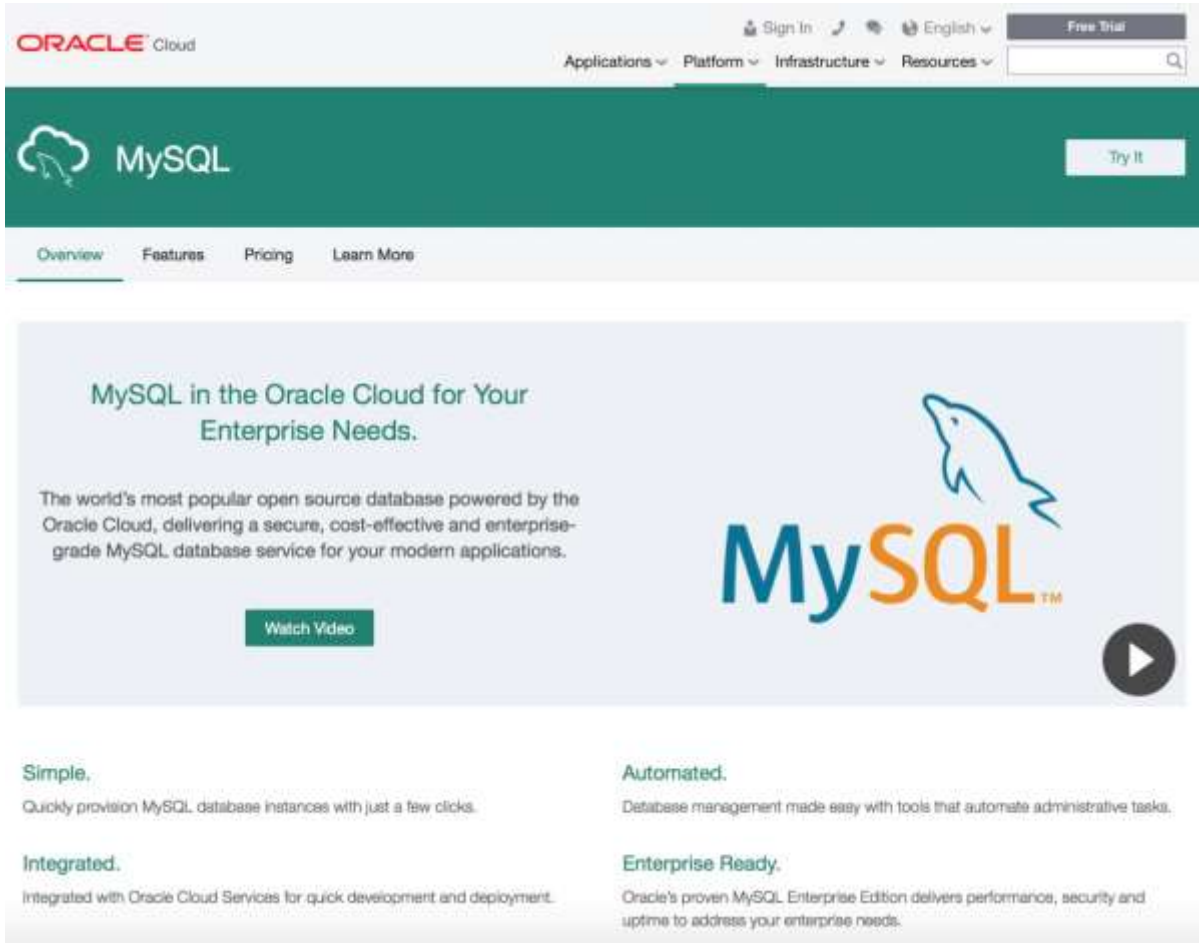
※適用前にMEBでバックアップが自動取得されます。

# MySQL Cloud Service: ビジネス上のメリット


- **ビジネスの俊敏性を向上**:
  イノベーションにリソースを集中し、迅速に最新のアプリケーションを提供。
- **確実なセキュリティ,パフォーマンス,稼働時間**:
  ソースレベルから、最も包括的なMySQL Cloudプラットホームを利用する事が可能。
- **TCO（総所有コスト）を削減:**
  稼働時間を向上させながら、インフラストラクチャ及びデータベース管理操作コストを節約可能。

REST APIを利用して、自動化する事も可能です。
http://docs.oracle.com/cloud/latest/mysql-cloud/CSMCS/toc.htm
http://docs.oracle.com/en/cloud/iaas/compute-iaas-cloud/stcsa/toc.htm

# 詳細情報 @ cloud.oracle.com/mysql




Sign up today for a free trial @

https://cloud.oracle.com/mysql

# アジェンダ

1. Oracle MySQL Cloud Service
2. **MySQL 8.0 RC 新機能**
3. MySQL Group Replication、MySQL InnoDB Cluster
4. MySQL Enterprise Edition
5. 参考情報

RC

# MySQL 8.0

# MySQL 8.0 RC（リリース候補版）

- 2018年1月26日時点の最新版はMySQL 8.0.4 RC
- フィードバック募集中
- バグ報告や機能追加要望はこちらから
  - MySQL Bugs
    https://bugs.mysql.com/

# MySQL 8.0：Webアプリケーション開発効率向上を実現

## Mobile Friendly

位置情報ベースのサービス向けの機能強化と絵文字を含めたユニコード対応😃


## Developer First

ハイブリッド型のデータモデルとアクセスAPIによる開発柔軟性

## Data Driven

アプリケーションデータ分析による運用中サービス改良支援

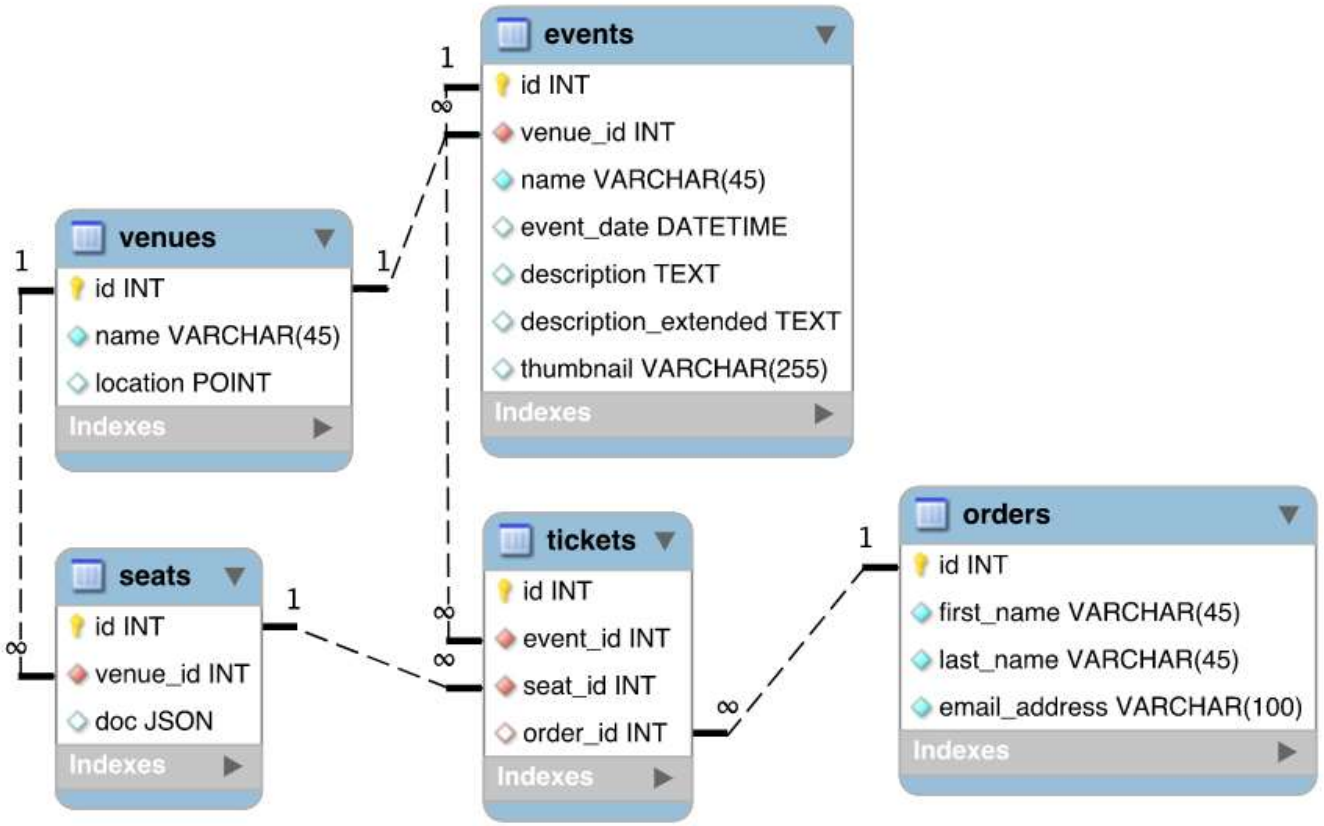
## 24x7 at Scale

## Scalable & Stable

アクセス集中時の処理改良、セキュリティと耐障害性強化

# Building a Sample Application Ticket Booking System

# **Case Study:** Our Booking System



# MySQL 8.0 : モバイルアプリとの親和性

## GIS(空間図形情報)サポートの強化

- 位置情報ベースのサービスとの連携の改良
- MySQL 5.7 にて Boost.Geometry ライブライリーを統合
- MySQL 8.0 にて球面座標と測地座標系(SRS)サポート

## ユニコードをデフォルトキャラクタセットに

- 絵文字をサポートする **`utf8mb4` がデフォルトのキャラクタセットに**
- ユニコード文字列の処理性能が16倍以上向上するケースも
- Unicode 9.0 をサポート
- UCA(Unicode照合アルゴリズム)ベースの新しい各言語用の照合

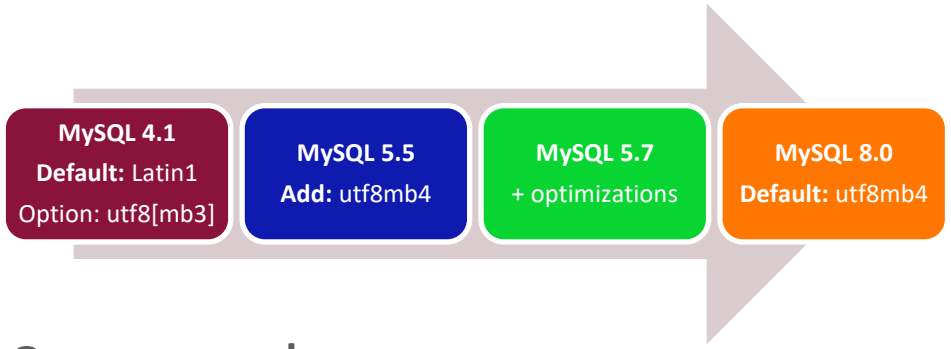
# st_distance()を利用した距離の測定

```
mysql> SELECT ST_Distance_Sphere(ST_GeomFromText('POINT(139.718754 35.671148)'),
ST_GeomFromText('POINT(135.492778 34.695758)')) as 'From TOKYO Office To Osaka Office';
+-----------------------------------+
| From TOKYO Office To Osaka Office |
+-----------------------------------+
|                 399041.1417772843 |
+-----------------------------------+

mysql> SELECT ST_Distance(ST_GeomFromText('POINT(35.671148 139.718754)', 4326),
ST_GeomFromText('POINT(34.695758 135.492778)', 4326)) as 'From TOKYO Office To Osaka
Office';
+-----------------------------------+
| From TOKYO Office To Osaka Office |
+-----------------------------------+
|                 399801.5254154028 |
+-----------------------------------+
```


参考: https://dev.mysql.com/doc/refman/8.0/en/spatial-analysis-functions.html

参照: Google Map

# utf8mb4 as default character set



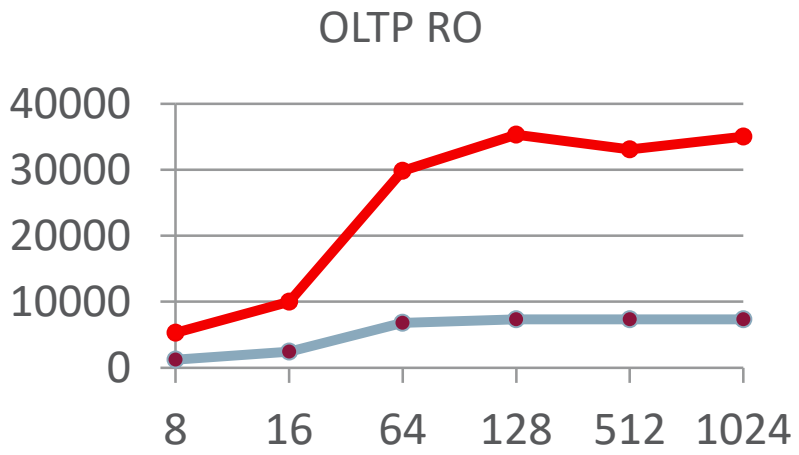

- 最新のUnicode 9.0をサポート
- 国ごとの照合順序を実装(日本語を含む)
  例) utf8mb4_ja_0900_as_cs, utf8mb4_ja_0900_as_cs_ks
  - ai as : アクセントセンシティブ(アクセント,濁音,破裂音の区別)
  - ci cs : ケースセンシティブ(大文字,小文字の区別)
  - ks : カナセンシティブ('あ','ア','ァ'を区別)

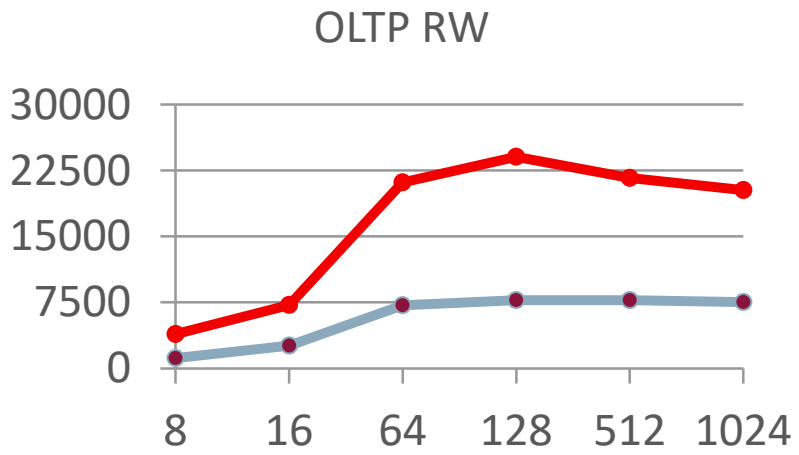
```
mysql> select *,@@version from information_schema.COLLATIONS where CHARACTER_SET_NAME = 'utf8mb4' and IS_DEFAULT = 'YES';
+--------------------+--------------------+----+------------+-------------+---------+-----------------------------+
| COLLATION_NAME     | CHARACTER_SET_NAME | ID | IS_DEFAULT | 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 |
+--------------------+--------------------+-----+------------+-------------+---------+---------------+-----------+
```

UNICODE詳細: http://www.unicode.org/

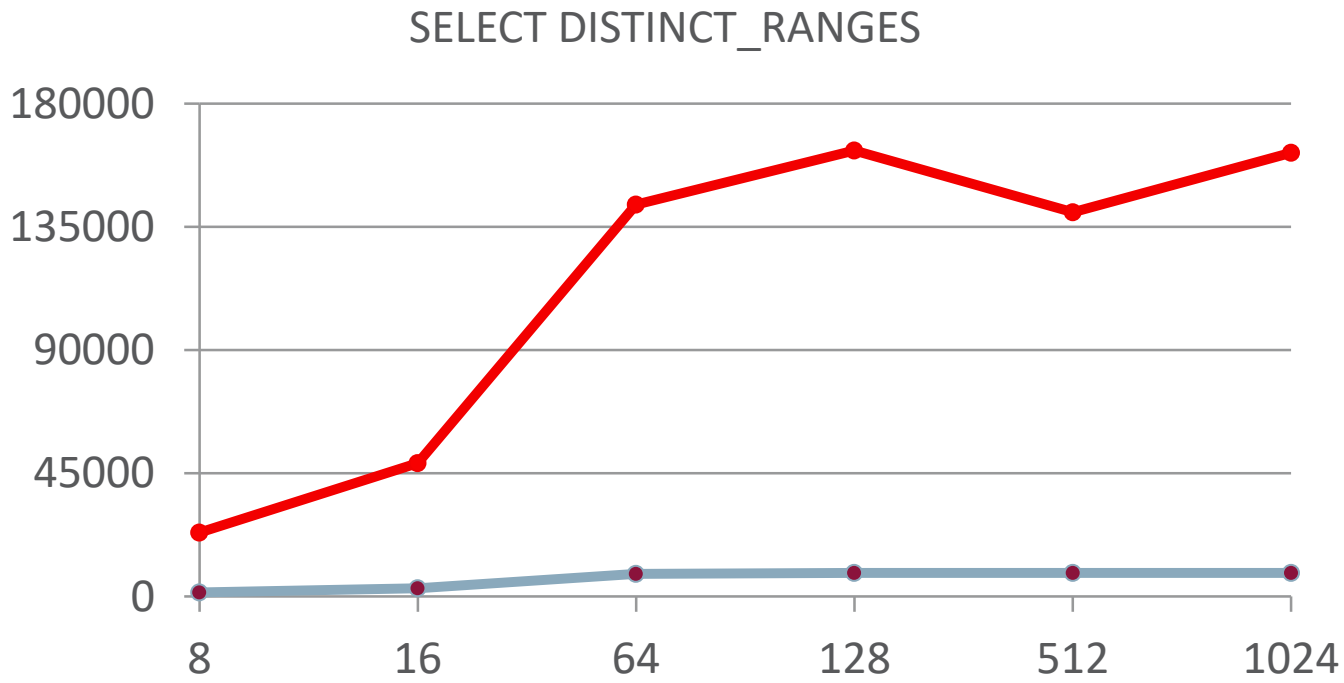
WL#10818: Add utf8mb4 accent sensitive and case insensitive collation
https://dev.mysql.com/worklog/task/?id=10818
WL#7554: Switch to new default character set and change mtr test cases
https://dev.mysql.com/worklog/task/?id=7554

# MySQL 8.0 vs MySQL 5.7 utf8mb4




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



# MySQL 8.0：アプリケーション開発者に柔軟性を

## データ型

### JSON データ型

リレーショナルなテーブルと非構造データとシームレスに統合。さらに MySQL 8.0 では更新性能の最適化

## SQL 関数

### JSON 関数

JSON データの参照更新のための各種 SQL 関数を実装。MySQL 8.0 では JSON データを SQL で分析するための変換関数も追加

## ハイブリッドAPI

### MySQL X DevAPI

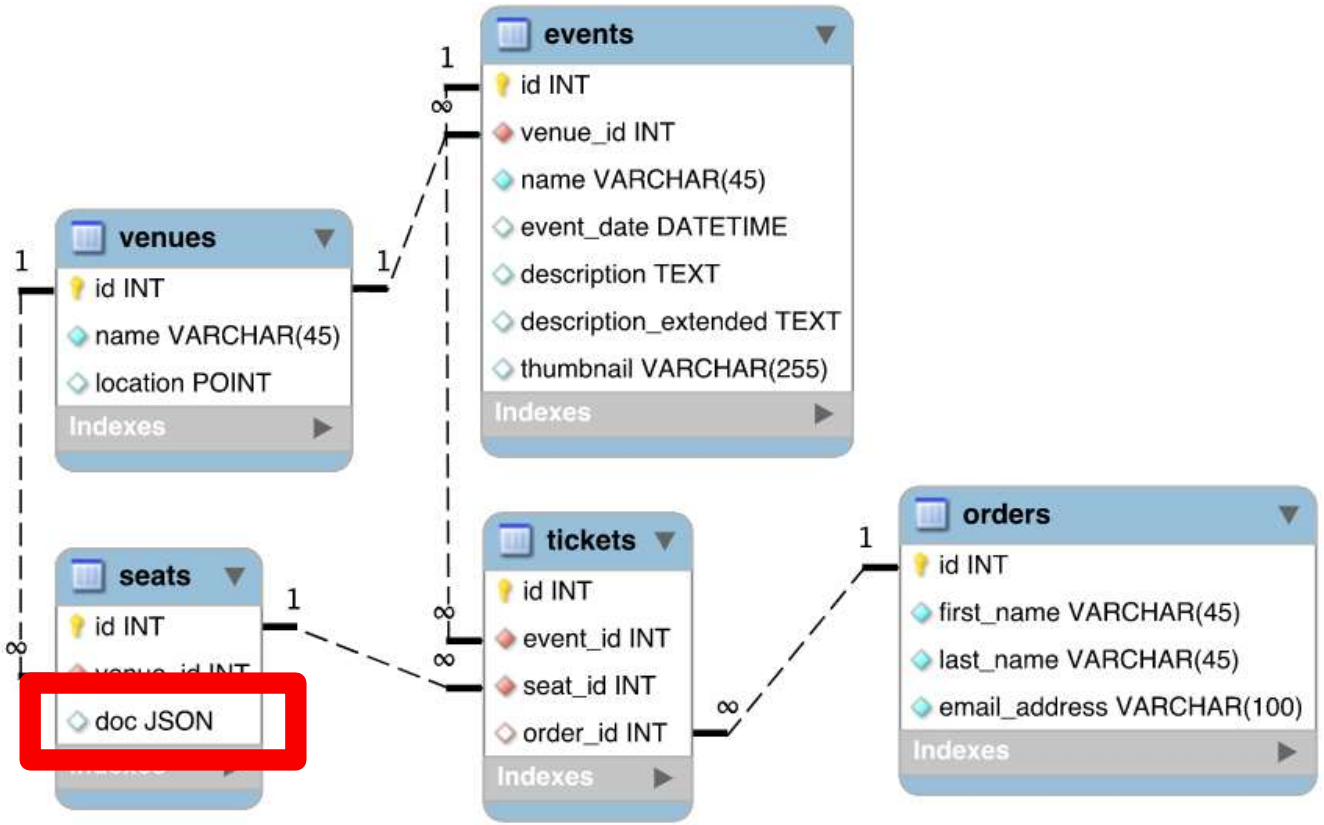
SQL と CRUD な NoSQL のハイブリッドAPIによる開発柔軟性

# Javascript Everywhere

**Backend:**


**Frontend:**

# Flexible Schema



```
mysql> SELECT id, doc FROM seats WHERE id = 28100¥G
*************************** 1. row ***************************
 id: 28100
doc: {"row": 10, "seat": 13, "section": 215, "properties":
{"amenities": [{"type": "washroom", "distance_in_meters":
38.564358156700024}, {"type": "bar", "distance_in_meters":
152.33173722618423}, {"type": "snacks", "distance_in_meters":
35.965617807550004}, {"type": "souvenirs", "distance_in_meters":
215.66576701185272}], "accessible": false, "emergency_exits":
[{"exit 1": 100.66892563427699}, {"exit 2": 374.19603448751946},
{"exit 3": 563.9332987311606}, {"exit 4": 886.7355222969646},
{"exit 5": 1900.9778593955355}], "entrance_number": 2}}
1 row in set (0.00 sec)
```

# JSON Functions

## MySQL 5.7 and 8.0

JSON_ARRAY_APPEND()
JSON_ARRAY_INSERT()
JSON_ARRAY()
JSON_CONTAINS_PATH()
JSON_CONTAINS()
JSON_DEPTH()
JSON_EXTRACT()
JSON_INSERT()
JSON_KEYS()
JSON_LENGTH()

JSON_MERGE[_PRESERVE]()
JSON_OBJECT()
JSON_QUOTE()
JSON_REMOVE()
JSON_REPLACE()
JSON_SEARCH()
JSON_SET()
JSON_TYPE()
JSON_UNQUOTE()
JSON_VALID()

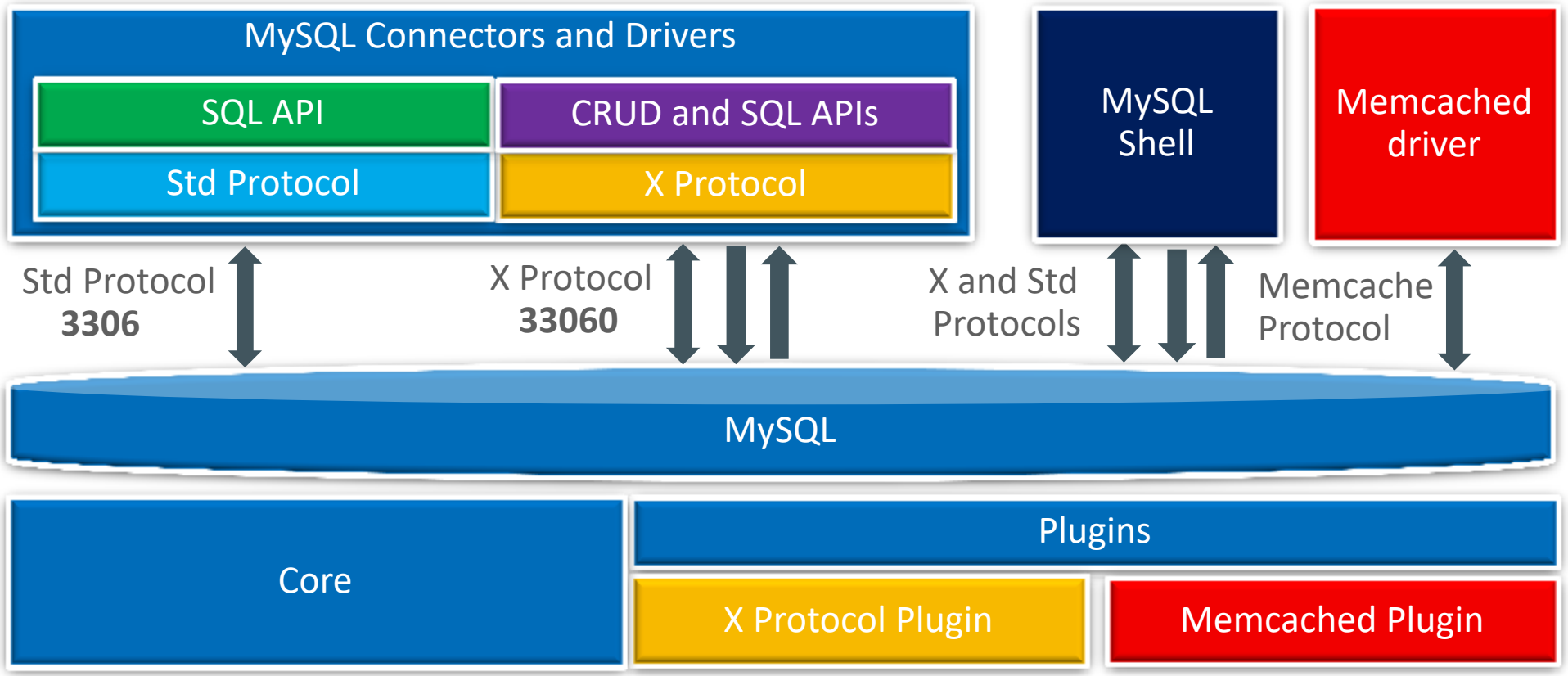
**JSON_PRETTY()**
**JSON_STORAGE_SIZE()**
**JSON_STORAGE_FREE()**
**JSON_ARRAYAGG()**
**JSON_OBJECTAGG()**
**JSON_MERGE_PATCH()**
**JSON_TABLE()**

# 【例】JSON_TABLE()

```
SELECT * FROM seats,
 JSON_TABLE(doc, "$.properties.amenities[*]" COLUMNS (
  id for ordinality,
  amenity_type VARCHAR(100) PATH "$.type",
  distance float PATH '$.distance_in_meters')
) AS amenities
WHERE seats.id = 28100
  AND amenities.amenity_type IN ('snacks', 'bar')
ORDER BY amenities.distance;
+-----+---------------+-----------+
| id  | amenity_type  | distance  |
+-----+---------------+-----------+
|  2  | bar           | 100.538   |
|  3  | snacks        | 136.647   |
+-----+---------------+-----------+
2 rows in set (0.00 sec)
```

# コネクター, ドライバー, プロトコル拡張機能

```
+-------------+---------------+--------------------+
| PLUGIN_NAME | PLUGIN_VERSION | PLUGIN_DESCRIPTION |
+-------------+---------------+--------------------+
| mysqlx      | 1.0           | X Plugin for MySQL |
+-------------+---------------+--------------------+
```



# X DevAPI


X Plugin (MySQL) ⇔ X Protocol ⇔ X DevAPI (Driver)

- X Pluginを有効にする事で、X Protocol経由で通信可能
- ドキュメントとテーブルのコレクションに対してのCRUD処理
- NoSQLライクな構文でドキュメントに対しCRUD処理可能
- Fluent API

✓ MySQL Connector/node.js (1.0.x)
✓ MySQL Connector/J (6.0.x)
✓ MySQL Connector/Net (7.0.x)
✓ MySQL Connector/python(2.2.x)
✓ MySQL Shell (1.0.x)

```
prod = sess.getSchema("prod")
res = prod.users.
    find("$.name = 'Milk'").
    fields(["name", "properties"])
```

参照 : http://dev.mysql.com/downloads/connector/

# MySQL Connectors include X Dev API

- Use SQL, CRUD APIs

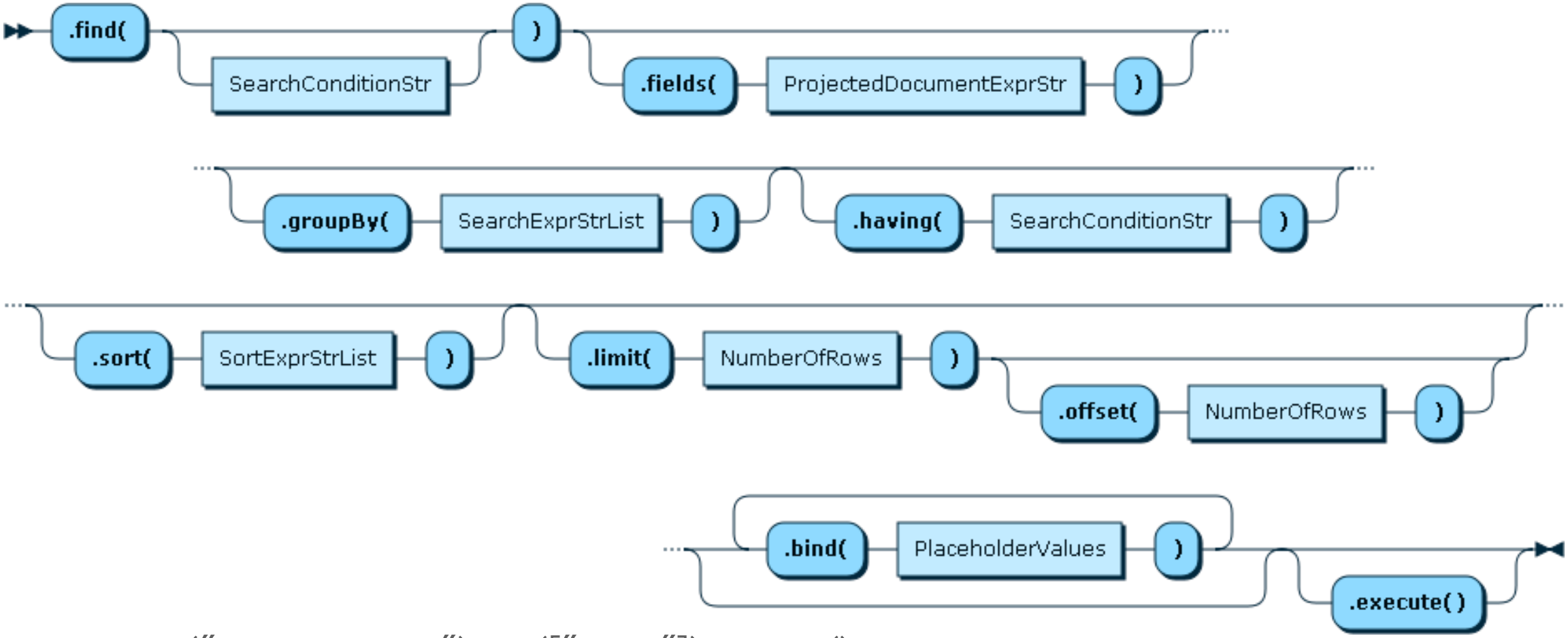
  スキーマレスドキュメントおよびリレーショナルテーブルに対応

  - Classic APIsに加えて、これらの全てが追加されます

| Operation | Document | Relational |
|---|---|---|
| Create | Collection.add() | Table.insert() |
| Read | Collection.find() | Table.select() |
| Update | Collection.modify() | Table.update() |
| Delete | Collection.remove() | Table.delete() |

参照) http://dev.mysql.com/doc/x-devapi-userguide/en/crud-operations-overview.html

# ドキュメントの検索




products.find(″color = 'yellow'″).sort([″name″]).execute();

# MySQL 8.0：データ分析処理の効率向上

## 共通テーブル式 (CTEs)

- サブクエリの導出表 (derived table) の代替
- WITH 句と呼ばれることも
- 分析処理 SQL 文の可読性や処理性能の向上、階層構造データ利用にも

```
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
```

## Window 関数

- ランキング作成などの分析処理用途で
  ユーザーからの追加要望の多かった機能
- 検索対象のレコードと周辺データとの関連を
  集計や分析

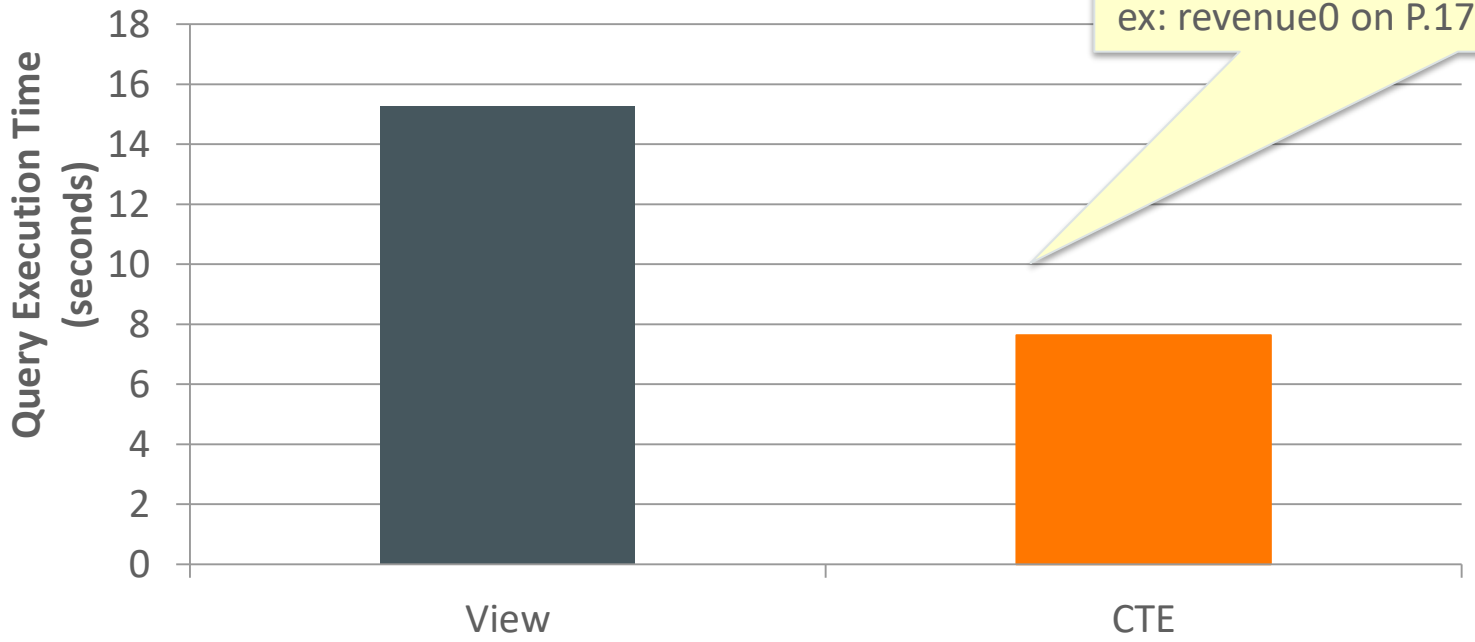
```
SELECT name, dept_id, salary,
 RANK() OVER w AS `rank`
FROM employee
 WINDOW w AS
  (PARTITION BY dept_id
   ORDER BY salary DESC);
```

```
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
*************************** 1. row
***************************
      id: 14447
event_id: 3
 seat_id: 16430
order_id: NULL
     doc: {"row": 2, "seat": 1, "section": 201,
"properties": {"amenities": [{"type": "washroom",
"distance_in_meters": 171.80304788220957}, {"type":
"bar", "distance_in_meters": 58.53288591702737},
{"type": "snacks", "distance_in_meters":
..
```

# DBT-3 Query 15

## クエリーパフォーマンス

ビューの場合, 2回マテリアライズされ,
CTEは1回のみマテリアライズされるため,
クエリの実行時間は半分に短縮されます。
ex: revenue0 on P.17




サブクエリの場合、同じ処理が複数回発生:

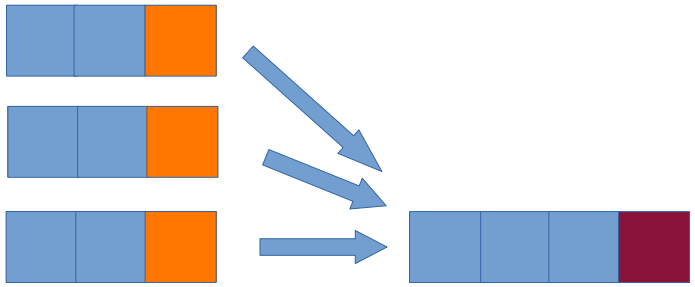
```
   SELECT ...FROM (SELECT a, b, SUM(c) s FROM t1 GROUP BY a, b) AS d1 JOIN (SELECT a, b, SUM(c) s
FROM t1 GROUP BY a, b) AS d2 ON d1.b = d2.a;
```

共通テーブル式の場合:

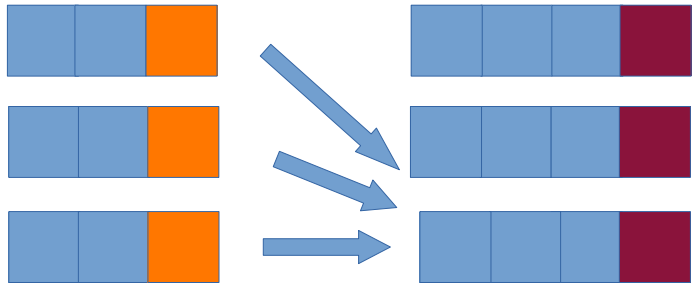
```
   WITH d AS (SELECT a, b, SUM(c) s FROM t1 GROUP BY a, b)  SELECT ... FROM d AS d1 JOIN d AS d2
ON d1.b = d2.a;
```

# Window関数とは？

- Window関数は、現在の行に関連する行セットについて、集計関数と同様に計算を行える
- 集計関数のように行を単一の出力にグループ化するのではなく、複数行を出力する
- Window関数は現在の行の近くの行（関連する行）にアクセスできる



集計関数



Window関数

# Window関数の例

- RANK関数
  - ランキングを求めることが出来る
- LAG関数
  - 1行前の値を参照できる
- SUM関数
  - ウィンドウごとの合計値を求めることが出来る

# Window関数: 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 |

```
SELECT
  seats.doc->"$.row" as row_no,
  seats.doc->"$.seat" as seat_no,
  LAG(seats.doc->"$.seat", 1) OVER w AS prev_seat,
  LEAD(seats.doc->"$.seat", 1) OVER w AS next_seat
FROM tickets
JOIN seats ON tickets.seat_id=seats.id
WHERE seats.doc->"$.section" = 201
WINDOW w AS
  (PARTITION BY seats.doc->"$.row"
   ORDER BY seats.doc->"$.seat")
ORDER BY row_no, seat_no;
+--------+---------+-----------+-----------+
| row_no | seat_no | prev_seat | next_seat |
+--------+---------+-----------+-----------+
| 1      | 1       | NULL      | 2         |
| 1      | 2       | 1         | 3         |
| 1      | 3       | 2         | 4         |
+--------+---------+-----------+-----------+
```

# MySQL 8.0：アプリケーションの性能拡張性向上

## アクセス集中時の対応改善

`SELECT FOR UPDATE` 文の `NOWAIT` や `SKIP LOCKED` オプションによるロック解放待ち削減

## 不可視インデックス

オプティマイザーからインデックスを隠蔽。インデックスを残した仮削除や段階的なインデックス追加を実現

## パフォーマンススキーマ

デフォルトで取得する性能統計情報の項目を拡張。パフォーマンススキーマへの参照性能向上

## ヒント句によるセッション変数変更

新しいヒント句の `SET_VAR` により一つのSQL文内でセッション変数を一時的に変更

## 降順インデックス

昇順と降順の複合インデックス利用時にも後方索引スキャンをしないためより高速に

## ヒント句の拡張

SQL文本体のテーブル指定順はそのままにJOIN順序やインデックスマージ対象の指定が可能に

# New! SELECT… FOR UPDATE の拡張

```
SELECT * FROM tickets
WHERE id IN (1,2,3,4)
AND order_id IS NULL
FOR UPDATE
NOWAIT;
```

行が既にロックされていれば、直ぐにエラーを返す

```
SELECT * FROM tickets
WHERE id IN (1,2,3,4)
AND order_id IS NULL
FOR UPDATE
SKIP LOCKED;
```

行が既にロックされていれば、その行に対するロック取得はあきらめる

# 不可視インデックス (Invisible Indexes)

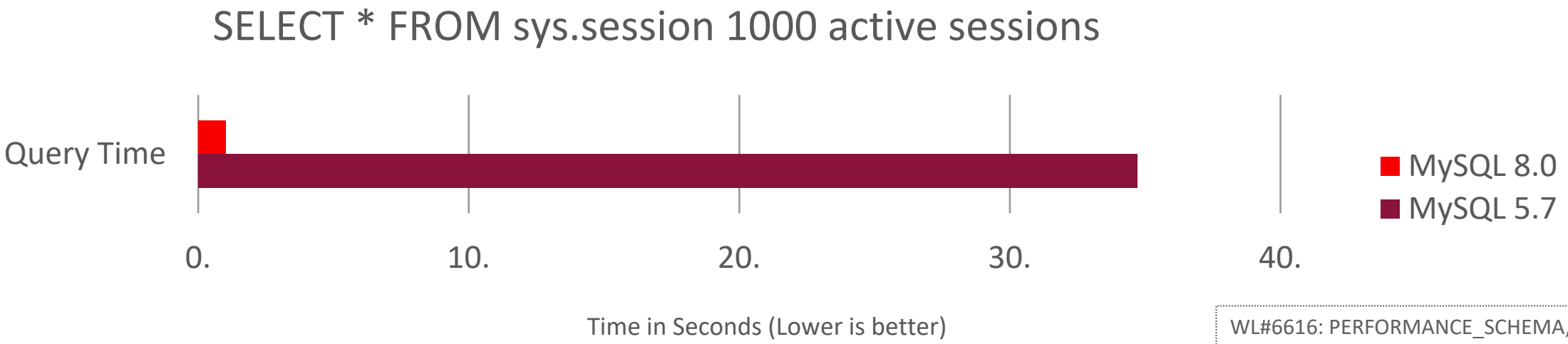
- オプティマイザーから見えない索引
  - 索引の無効化とは異なる
  - データ更新時にInvisible Indexesも更新される
- 2つのユースケース:
  - 仮削除(ゴミ箱)
  - 段階的な展開にてインデックスの有効性の確認

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

# Performance Schema Indexes

**Over 30x faster!**

- パフォーマンススキーマのテーブルへのより効率的なアクセスが可能
- 89個のテーブルに対し合計90個のインデックス
- オーバーヘッドを削減
  - 物理インデックスは内部的には維持されません
  - 索引の実装により、オプティマイザがより良い実行計画を選択




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

# 【例】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”が増加している
（テーブルスキャンが実行されている）

```
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”は0のまま増加していない
（テーブルスキャンが実行されていない）

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 |

# 【例】パフォーマンススキーマSQLエラーの計測

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

```
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 16:07:10 | 2016-10-27 18:24:06 |
| root |         1146 | ER_NO_SUCH_TABLE          | 42S02     |                 1 |                 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 |
+------+--------------+---------------------------+-----------+-------------------+-------------------+---------------------+---------------------+
```

存在しないテーブルへの参照エラー発生

```
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     |                 1 |                 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 |
+--------------+---------------------------+-----------+-------------------+-------------------+---------------------+---------------------+
```

# パフォーマンススキーマ ヒストグラム

## 例: mysqlslap実行時のクエリー実行時間の分布

```
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)
```

events_statements_histogram_globalとCTEを利用して実行時間の分布をビジュアライズした例

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

# パフォーマンススキーマの拡張

## ロック発生状況の確認

- どのデータがロックされているか？誰がロックを所有しているか？
  誰がデータを待っているか？

```
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        |
+-----------+-------------+------------+---------------+-----------+-------------+
```

ロックされているデータを表示

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

# オプティマイザヒントの拡張

- SET_VARヒント
  - SQL単位でシステム変数を変更できるヒント
  - セッション単位で変更可能なシステム変数をSQL単位で変更可能に（max_allowed_packetなど一部のセッション変数は変更不可）

使用例

```
SELECT /*+ SET_VAR(sort_buffer_size = 16M) */ name FROM people ORDER BY name;

INSERT /*+ SET_VAR(foreign_key_checks=OFF) */ INTO t2 VALUES(2);

SELECT /*+ SET_VAR(optimizer_switch='use_invisible_indexes=ON') */ name,region FROM
country WHERE region='Eastern Asia';
```

# 降順索引 (Descending Indexes)

**For B+tree indexes**

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

- 5.7: 昇順インデックスが作成され,サーバーがそれを逆方向にスキャンします
- 8.0: 降順でインデックスが作成され,サーバはそれをフォワードスキャンします
- メリット:
  - 前方索引スキャンは後方索引スキャンより高速
  - ASC / DESCソートキーでORDER BYにてfilesortの代わりにインデックスを使用可

# 降順索引 (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

# MySQL 8.0：MySQLサーバーの性能拡張性向上

## InnoDB専用サーバー構成

仮想マシンやクラウド環境の構成にあわせてInnoDBの最適なパラメタを自動的に設定

## クラウドフレンドリーな設定永続化

**SET PERSIST** コマンドにより、SQLインターフェースからの設定変更を永続化。どこから変更された設定値かを確認するテーブルも追加

## リソースグループ

スレッドとCPUのマッピングを行うことにより処理効率と性能向上を図る

## カラムヒストグラム

インデックスが設定されていない列の統計情報をオプティマイザーに提供し実行計画をさらに改善

## コスト見積もりの最適化

最新のストレージ技術への対応やデータのキャッシュ状況に応じたオプティマイザーでの実行計画

## トランザクションスケジューリング

“Contention-Aware Transaction Scheduling”がInnoDBのデフォルトのスケジューリングアルゴリズムとなり性能が劇的に向上

# InnoDB専用サーバーの自動構成

- システムメモリー量を確認し、ログファイルサイズ、バッファプールサイズ、フラッシュメソッドを自動的に調整
- 使用方法

```
SET PERSIST_ONLY innodb_dedicated_server = TRUE
```

参照: http://mysqlserverteam.com/plan-to-improve-the-out-of-the-box-experience-in-mysql-8-0/

**WL#9193: Autoscale InnoDB resources based on system resources by default**
https://dev.mysql.com/worklog/task/?id=9193

# InnoDB専用サーバーの自動構成

- MySQL 8.0.3での設定内容

```
[innodb_buffer_pool_size]
server_memory < 1G ? 128M（現在のデフォルトと同じ）
server_memory <= 4G ? server_memory * 0.5
server_memory > 4G ? server_memory * 0.75

[innodb_log_file_size]
server_memory < 1G ? 48M（現在のデフォルトと同じ）
server_memory <= 4G ? 128M
server_memory <= 8G ? 512M
server_memory <= 16G ? 1024M
server_memory > 16G ? 2048M

[innodb_flush_method]
O_DIRECT_NO_FSYNC
```

# 設定変更の永続化

- 以下の構文でシステム変数の変更を永続化可能
  - SET PERSIST max_connections = 500;
  - SET PERSIST_ONLY innodb_log_file_size = 128*1024*1024;
- システム変数変更のためにファイルシステムへのアクセス不要
- いつ、誰に設定されたかを確認出来る情報も追加

```
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         | 2017-10-14 14:48:28 | root     | localhost |
+---------------+-----------------+-------------------------------+-----------+-----------+---------------------+----------+-----------+
```

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

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

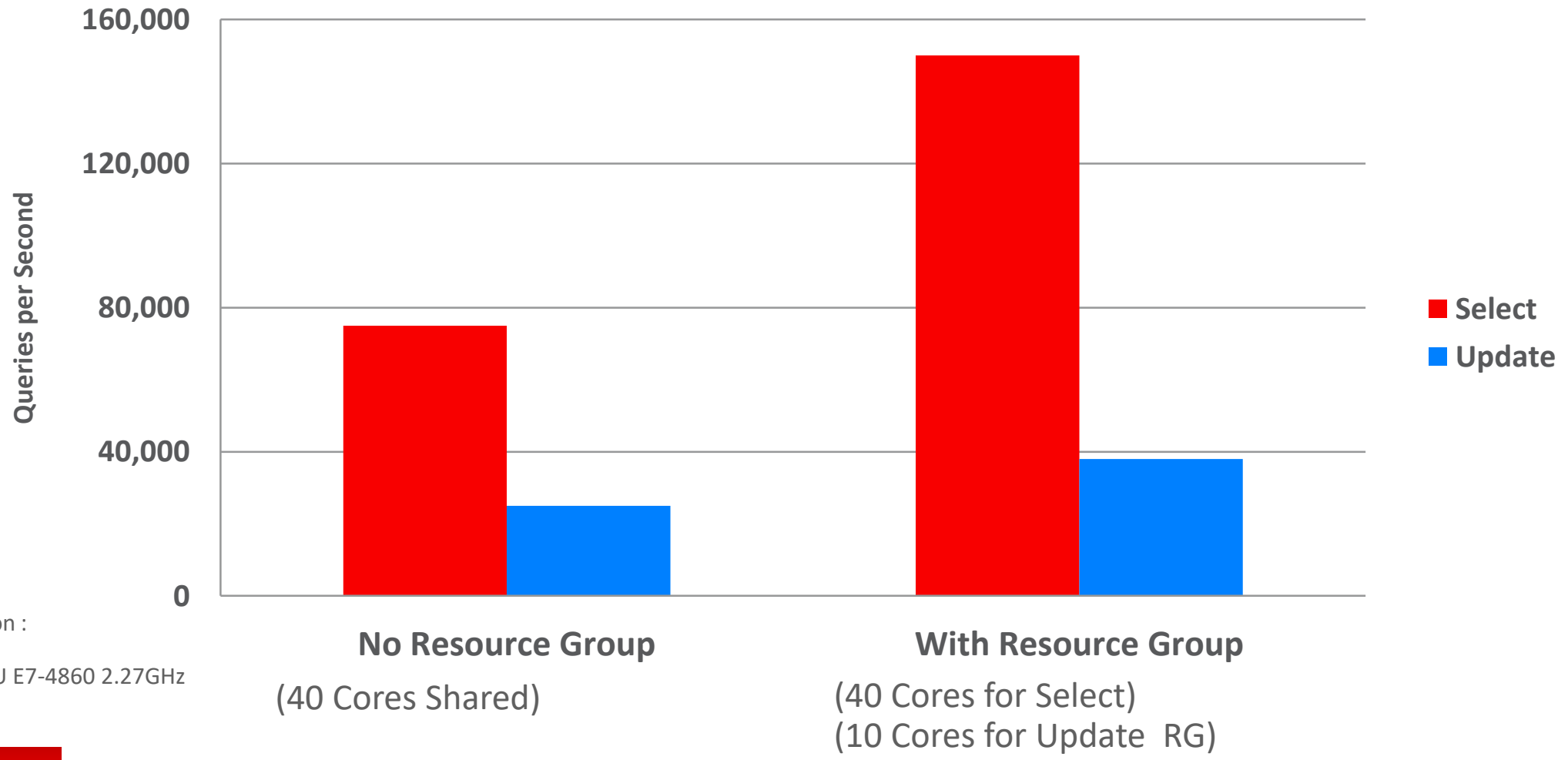
# リソースグループ

リソースグループを使用すると、そのリソースの制御が可能になり、グループ内のスレッドによるリソース消費を有効または制限できます。DBAは、さまざまな作業負荷に応じてこれらの属性を変更できます。

```
shell> cat /proc/cpuinfo | grep processor
processor       : 0
processor       : 1

mysql> CREATE RESOURCE GROUP CPU1 TYPE=USER VCPU=1;
Query OK, 0 rows affected (0.24 sec)

mysql> SELECT * from INFORMATION_SCHEMA.RESOURCE_GROUPS;
+----------------------+----------------------+--------------------------+----------+------------------+
| RESOURCE_GROUP_NAME | RESOURCE_GROUP_TYPE | RESOURCE_GROUP_ENABLED | VCPU_IDS | THREAD_PRIORITY |
+----------------------+----------------------+--------------------------+----------+------------------+
| USR_default          | USER                 |                        1 | 0-1      |                0 |
| SYS_default          | SYSTEM               |                        1 | 0-1      |                0 |
| CPU0                 | USER                 |                        1 | 0        |                0 |
| CPU1                 | USER                 |                        1 | 1        |                0 |
+----------------------+----------------------+--------------------------+----------+------------------+
4 rows in set (0.00 sec)

mysql> SET RESOURCE GROUP CPU0;select "This user connection will use Processor 0 Only";
mysql> SELECT /*+ RESOURCE_GROUP(CPU0) */ "This user connection will use Processor 0 Only";
```

WL#9467: Resource Groups
https://dev.mysql.com/worklog/task/?id=9467

# 【例】リソースグループパフォーマンス




System Configuration :
Oracle Linux 7,
Intel(R) Xeon(R) CPU E7-4860 2.27GHz
40 cores-HT

# ヒストグラム

- データが偏っている場合のクエリーの精度向上
- ヒストグラムはインデックスを作成よりコストが低い
- 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: a5980f0634db05c87a7aeb
         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**
クエリの50%は11.48マイクロ秒未満で実行

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

# 新しいオプティマイザー・コストモデル

- バッファプールのヒット率を意識した改善

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

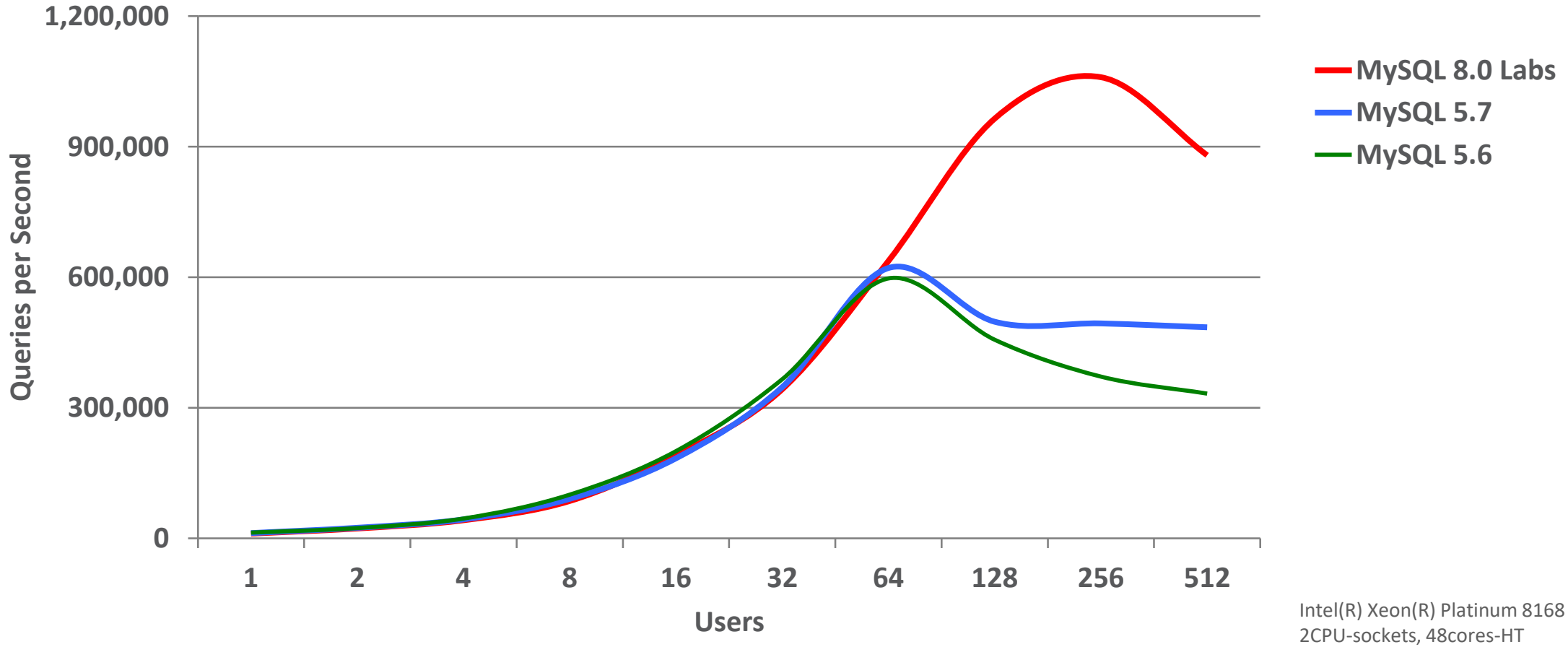
この例では、全データがメモリ上にある場合にはレンジスキャンの方がコストが低い
※IOブロックリードコストのデフォル値
・ディスク上：1
・メモリ上：0.25

データがメモリ上にある/ないによって、
レンジスキャンのパフォーマンスの差が大きい。
INNODB_CACHED_INDEXESからヒット率を判断し、適切な実行計画を選択

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

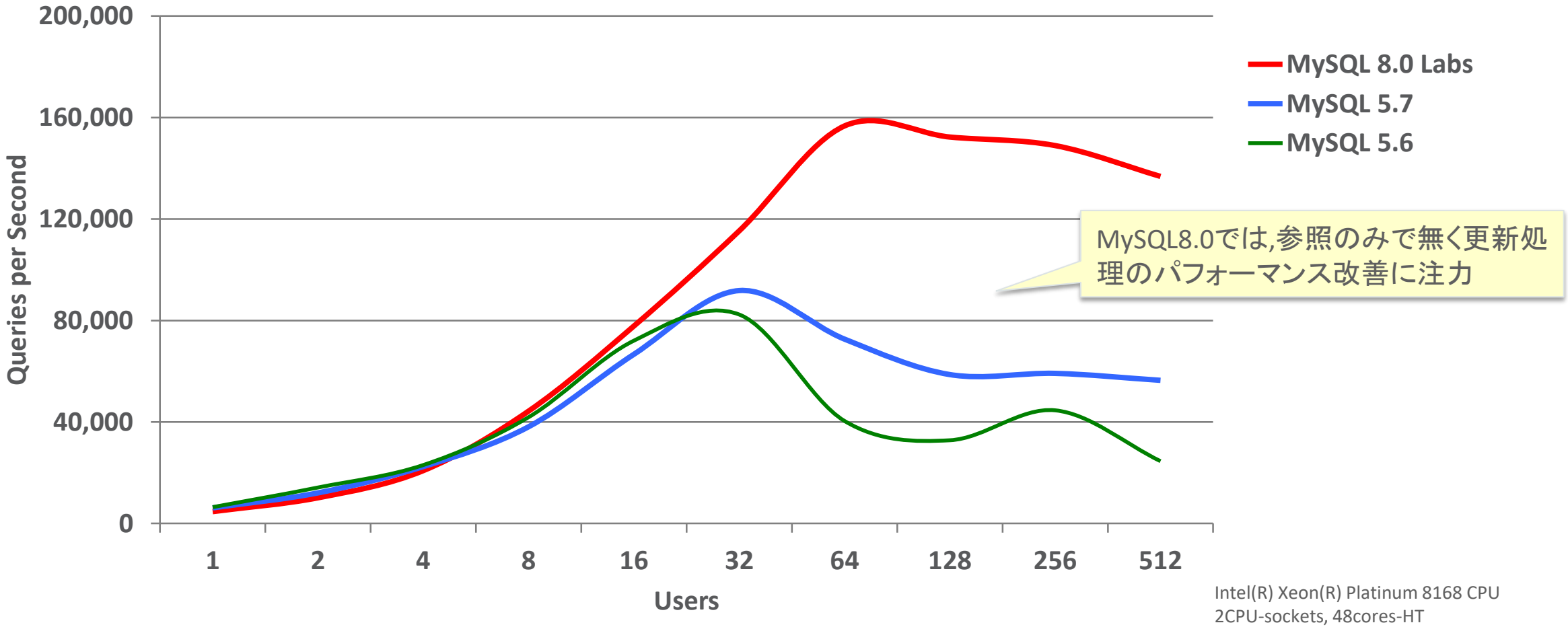
# Sysbench: **IO-bound** OLTP RO Point-Selects

**1,000,000+ QPS**




Intel(R) Xeon(R) Platinum 8168 CPU
2CPU-sockets, 48cores-HT
2x Intel Optane
Oracle Linux 7.3

# Sysbench: IO-bound OLTP RW Updates-only




Intel(R) Xeon(R) Platinum 8168 CPU
2CPU-sockets, 48cores-HT
2x Intel Optane
Oracle Linux 7.3

# MySQL 8.0 : アプリケーションのセキュリティ&可用性強化

## MySQL InnoDB Cluster

MySQL 標準の高可用性パッケージ。自動フェールオーバー & リカバリ、矛盾検知

## データディクショナリ

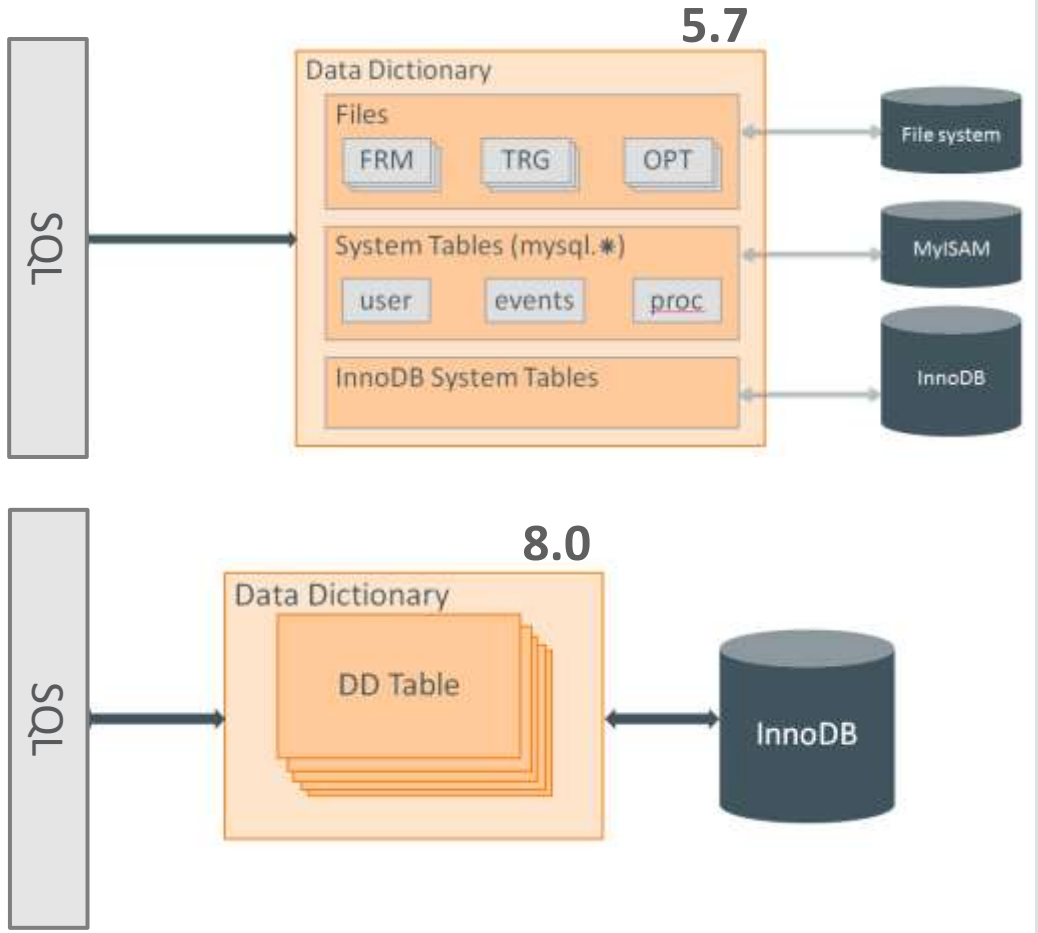
メタデータ管理を InnoDB のテーブルで一元管理。メタデータの一貫性と信頼性向上

## セキュリティ強化

ロール & Dynamic Privileges 機能追加。透過的データ暗号化を拡張。

# Transactional Data Dictionary

- 信頼性の向上
- InnoDB内部にデータディクショナリーを格納
  - No FRM files
  - No DB.OPT files
  - No TRG files
  - No TRN files
  - No PAR files
- MyISAMテーブルは含まれなくなりました




```
[root@DockerHost oracle]# ls -l /docker/docker802/world   8.0
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
```

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

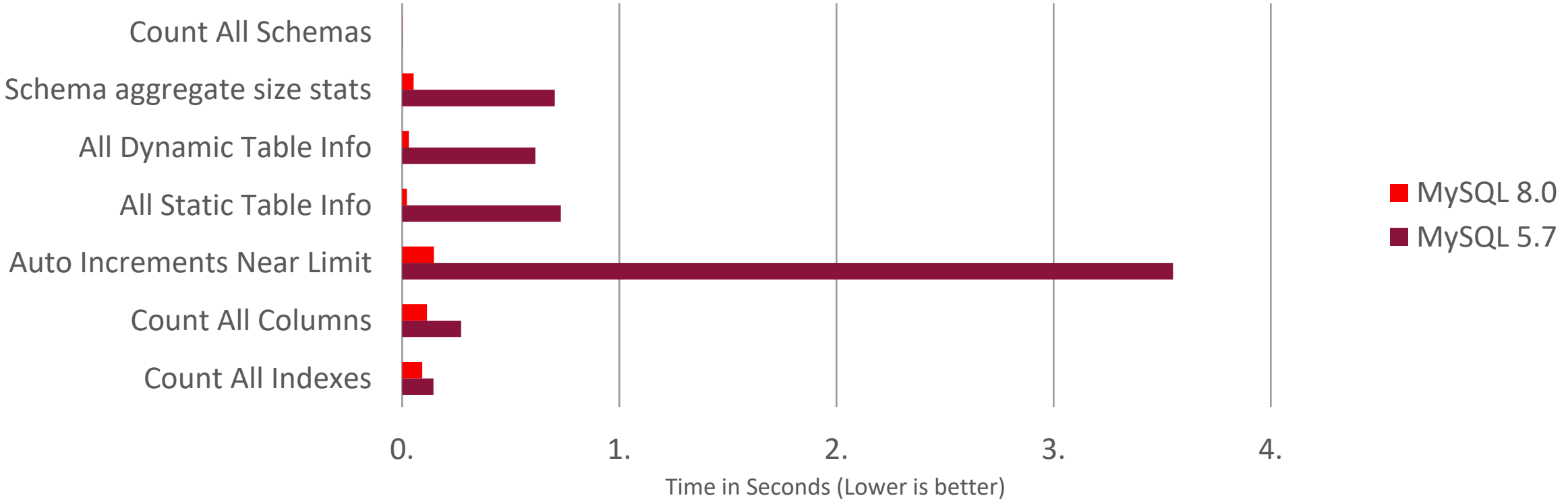
- 柔軟な異種OSとの互換性の向上
  - ファイルシステムのセマンティクスに依存しない
- アトミックDDL
  - レプリケーションの改善、サーバーのエッジケースの簡素化
- 外部キーのMDL(メタデータロック)
- 柔軟なメタデータAPI
  - 新しい機能の容易な追加

| Ver | Delete Tables | Delete Stored Programs | Delete Schema | ATOMICITY |
|---|---|---|---|---|
| 5.7 | - Metadata, TRN/TRG/FRM files<br>- 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<br>- 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)

独自のテストで7/10のクエリがより高速化！




X30 FAST: SELECT TABLE_SCHEMA,TABLE_NAME,TABLE_TYPE,ENGINE,ROW_FORMAT FROM information_schema.tables WHERE TABLE_SCHEMA LIKE ‘db%';

# MySQL 8.0：セキュリティの強化

## SQLロールの実装

Easier to manage user and applications rights and SQL standard compliant

## ログファイルの透過的暗号化

AES 256 encryption of REDO, UNDO and Binary Log in addition to tablespace files

## メタデータ変更がアトミックに

New InnoDB based data dictionary enables ACL statements atomic and reliable

## パスワード管理強化

Establish password-reuse policy with Password History, and faster with caching

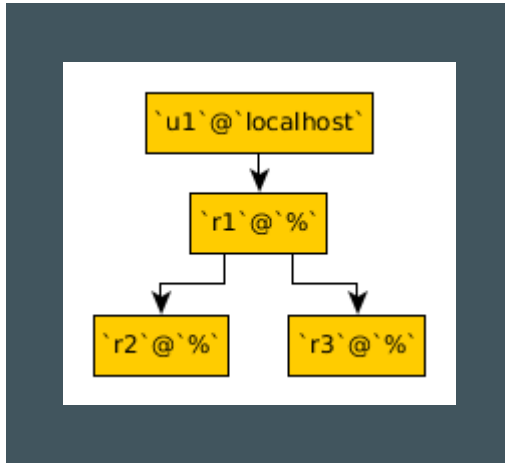
## Dynamic Privileges

Provides finer grained administrative level access controls for less use of root user

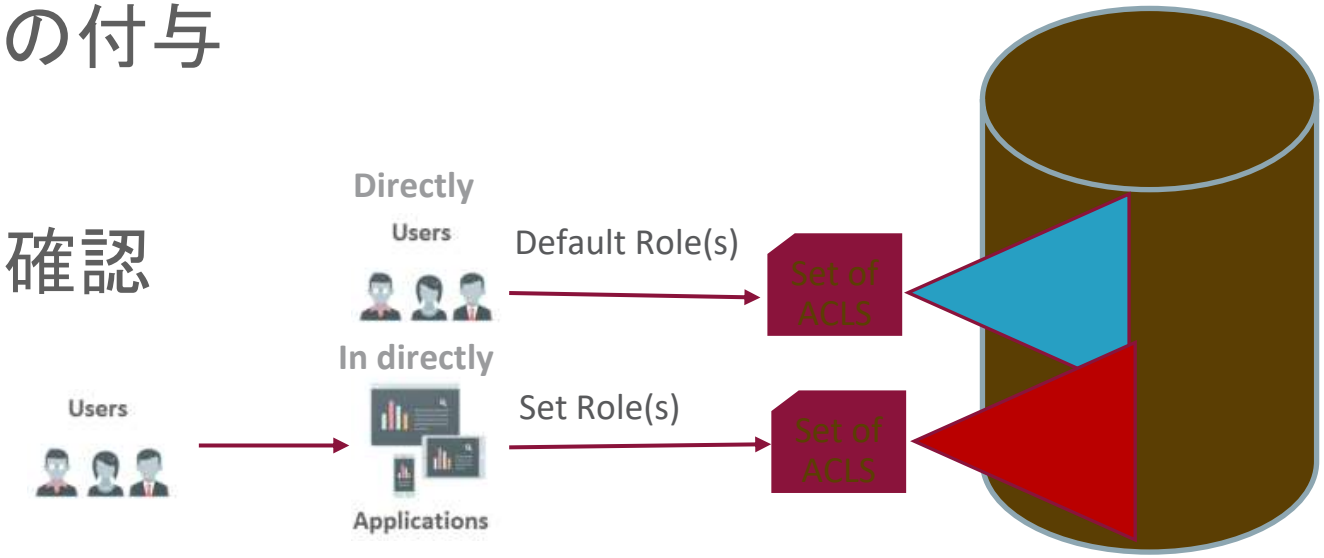
# ユーザーロール

MySQLアクセスコントロールの改善

- 8.0.0 DMRで導入されました (5.7: PROXY USER)
- ユーザーとアプリケーションの権限管理を容易に
- ユーザー/ロールに対してロールの付与
- デフォルトロールを定義
- ROLES_GRAPHML()関数でロール確認




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




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

# アトミック ACLステートメント

- 長年にわたるMySQLの課題！
  - レプリケーション、HA、バックアップ等
- ACLテーブルをInnoDB Data Dictionaryに格納
- テーブル操作だけでなく、メモリキャッシュも更新が必要
- 複数の論理演算を実行するステートメントに適用されます

例:

  - `CREATE USER u1, u2`
  - `GRANT SELECT ON *.* TO u1, u2`
- ACLのキャッシュとテーブルを変更時
  - カスタムMDLロックを使用してACL関連のアクティビティをブロック

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

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

# InnoDB Redo/Undo 暗号化

- AES 256 暗号化
- Redo/Undoログがディスクに書き出し時に暗号化される
- Redo/Undoログがディスクから読み出し時に複合される
- メモリ上ではRedo/Undoログデータは暗号化されていない
- InnoDB表領域暗号化と同様の2層暗号化鍵管理
  - 鍵のローテーションが高速、高パフォーマンス
- *容易に使用可能*
  - *システム変数 innodb_redo_log_encrypt, innodb_undo_log_encrypt で制御*

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

# パスワード強化

- New! パスワード履歴 - より幅広いセキュリティポリシーに対応
  - Require new passwords not reuse old ones - By number of changes and/or time.
  - Establish password-reuse policy globally as well as on a per-account basis.
- New! キャッシュ付きHA2
  - Strong and Fast
  - Strong - SHA-256 password hashing (many rounds, seeds, …)
  - Fast - Caching
    - Greatly reduces latency
- New! より多くのプロトコルのサポート
- New! シームレスなRSAパスワード交換 (OpenSSLのリンク不要)

# アジェンダ

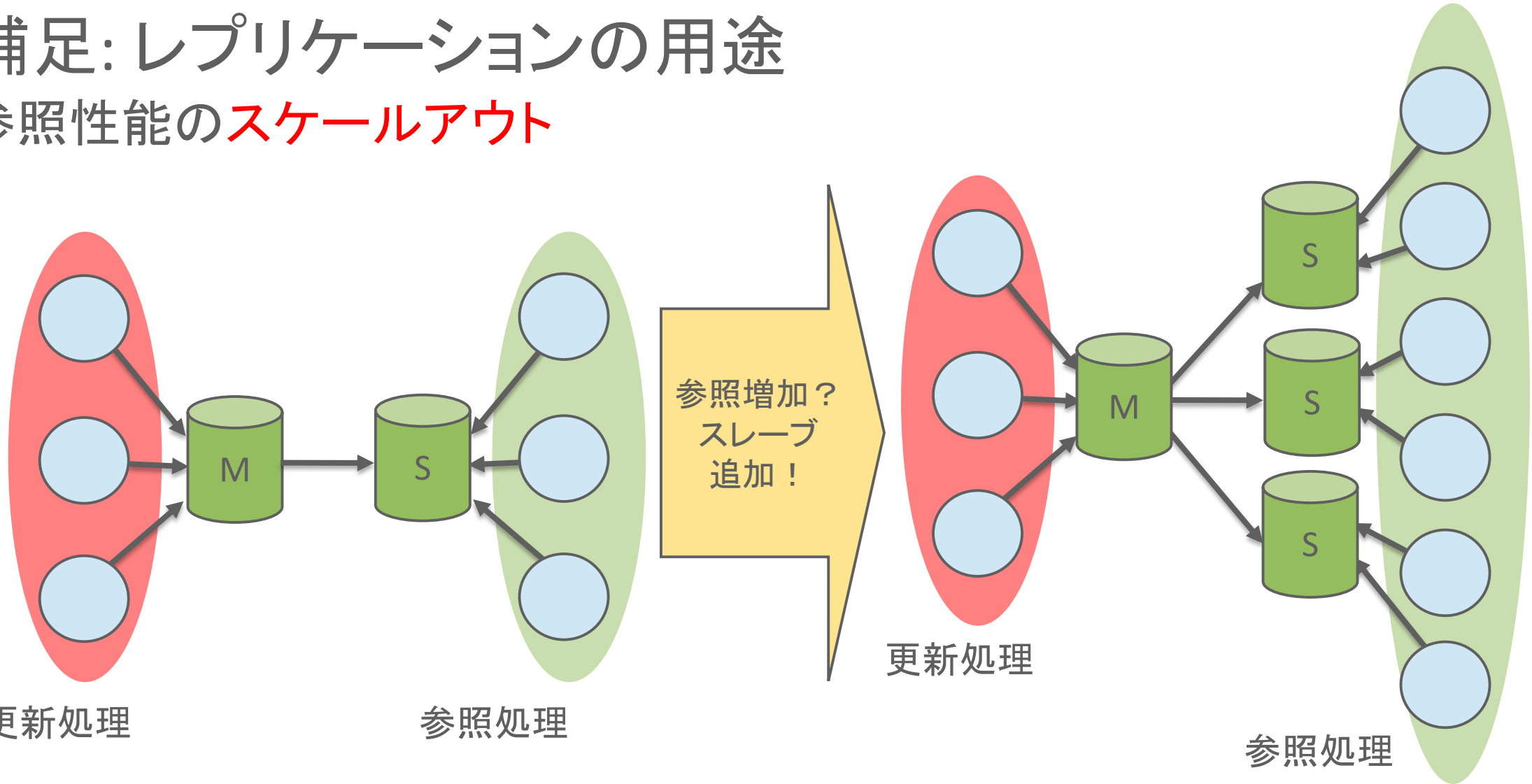
1. Oracle MySQL Cloud Service
2. MySQL 8.0 RC 新機能
3. **MySQL Group Replication、MySQL InnoDB Cluster**
4. MySQL Enterprise Edition
5. 参考情報

# グループレプリケーションとは？

- MySQL5.7以降で利用可能な仮想同期レプリケーション
- MySQLがサポートする全てのプラットフォームに対応
  - Linux, Windows, Solaris, OSX, FreeBSD
- 手軽に高可用性構成を実現可能
  - 複数台でグループを組み、全台が同じデータを持つ
  - 3台以上の奇数でグループを構成することを推奨
- シングルプライマリーモード（デフォルト）とマルチマスターモードが使用可能

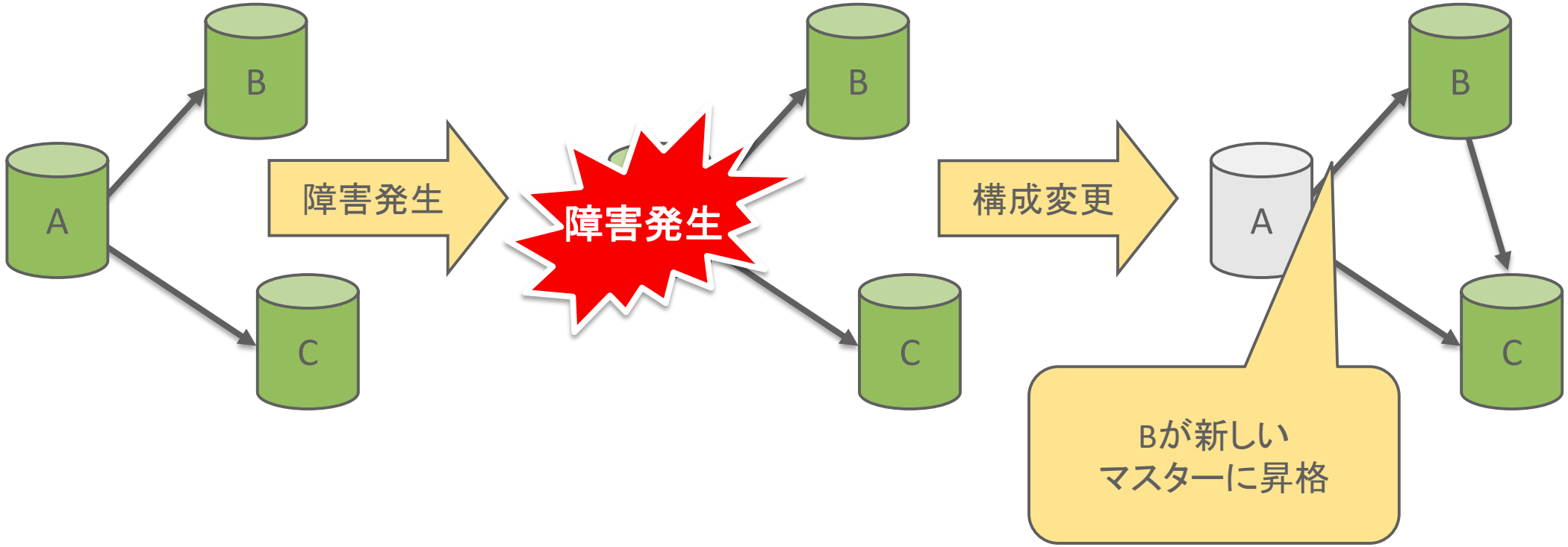
# 補足: レプリケーションの用途

## 参照性能のスケールアウト



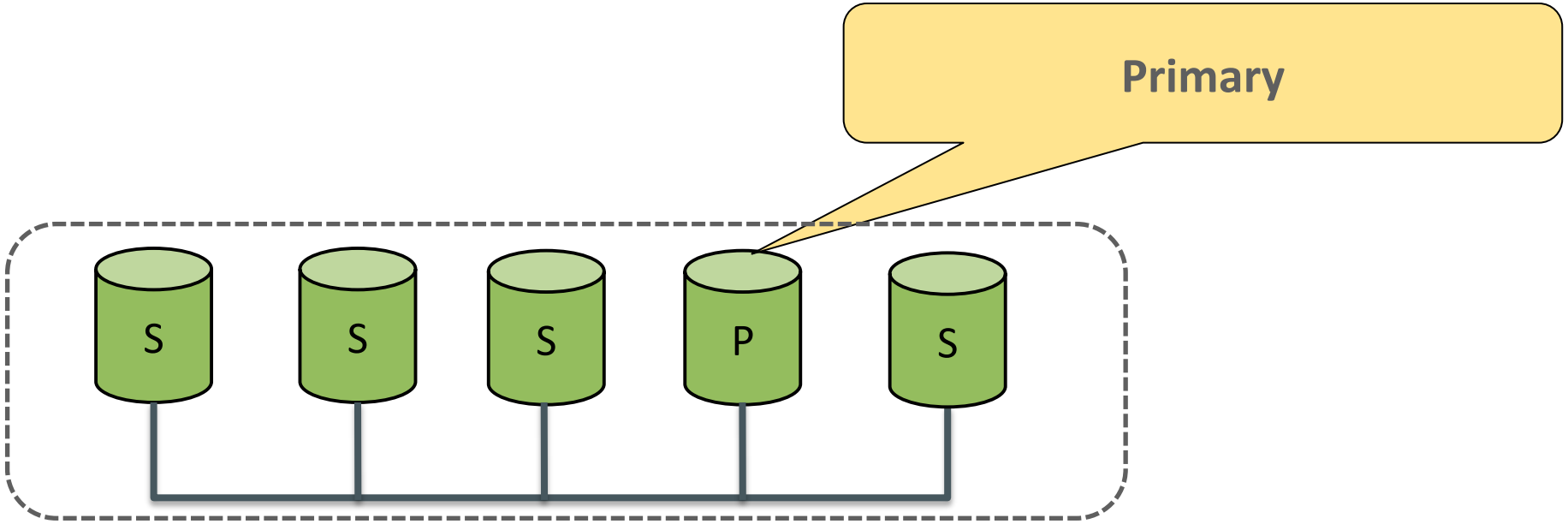
# 補足: レプリケーションの用途

## 冗長性: マスターがクラッシュした場合, スレーブをマスターに昇格



# シングルプライマリーモード

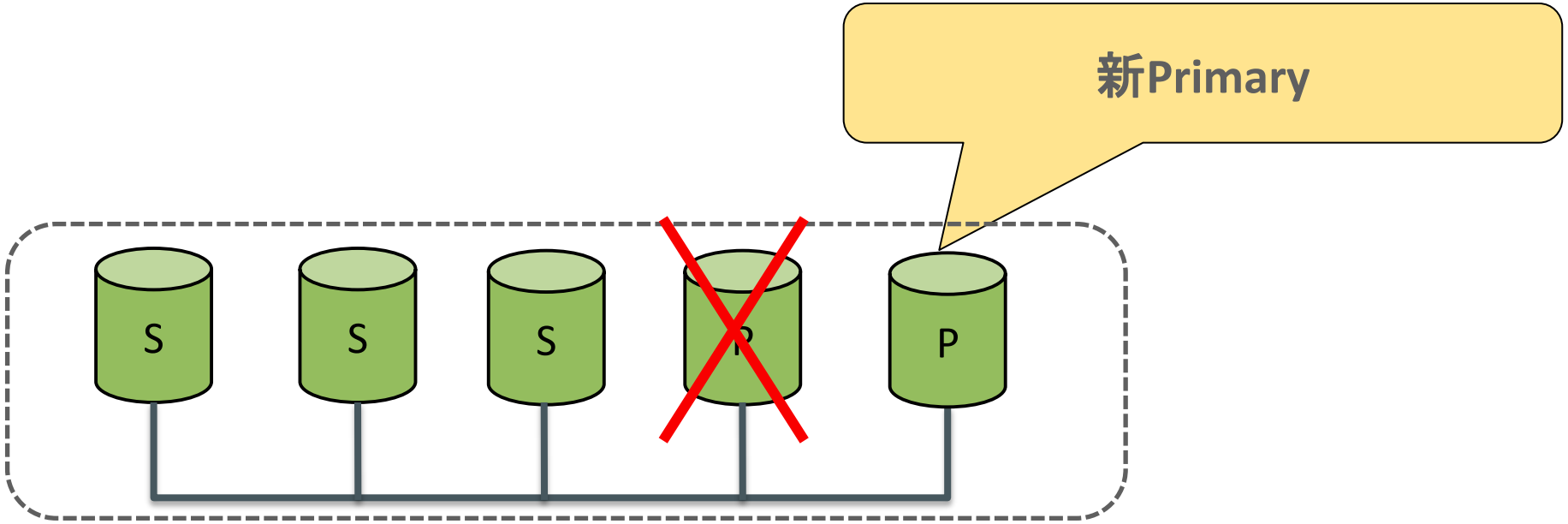
- 自動的なリーダー選択メカニズム
  - Secondaryノードは参照のみ可能 (書き込んだ場合: ERROR 1290)




```
ERROR 1290 (HY000): The MySQL server is running with the --super-read-only option so it cannot execute this statement
```

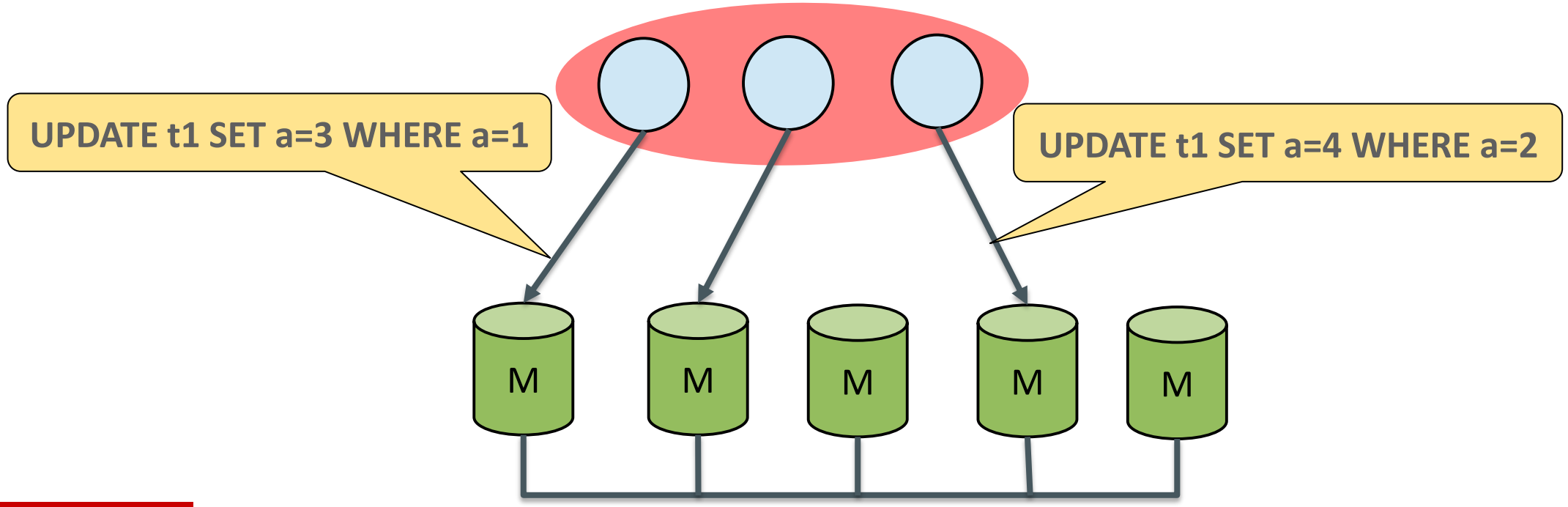
# シングルプライマリーモード

- 自動的なリーダー選択メカニズム
  - 障害発生時は自動的にフェイルオーバー



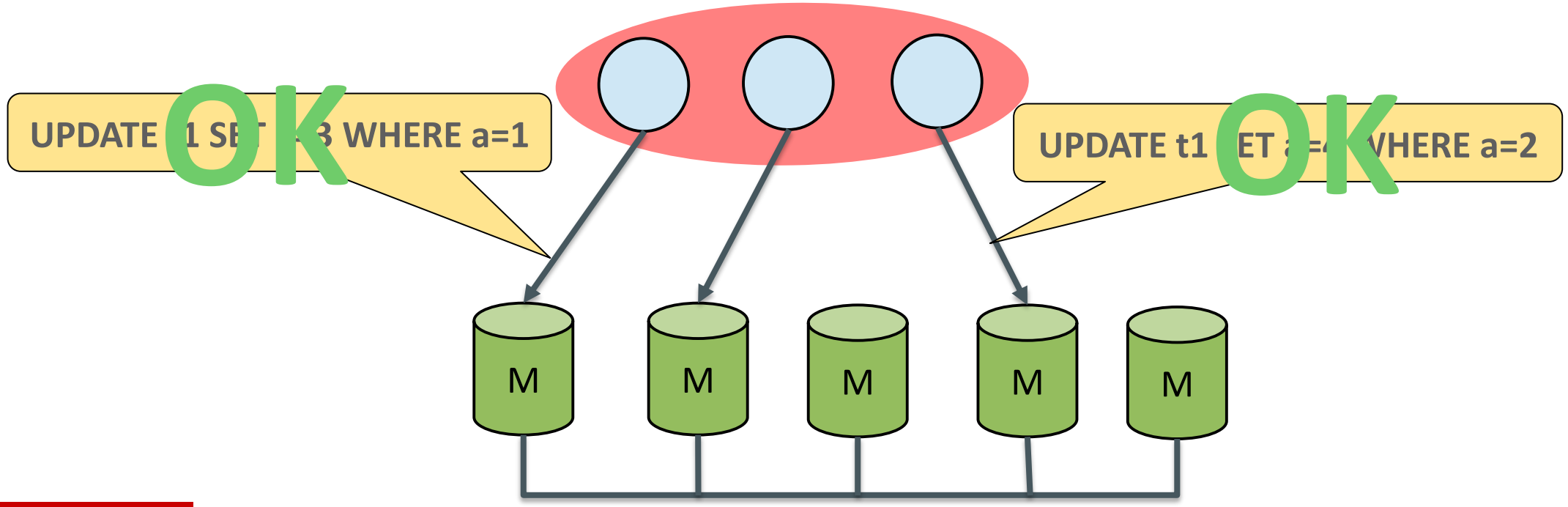
# マルチマスターモード：どこでも更新可能

- 異なるサーバー上での2つのトランザクションは、同じデータを更新可能
- 競合が検出された場合、自動的に対処される
  - 先にコミットしたトランザクションが優先される



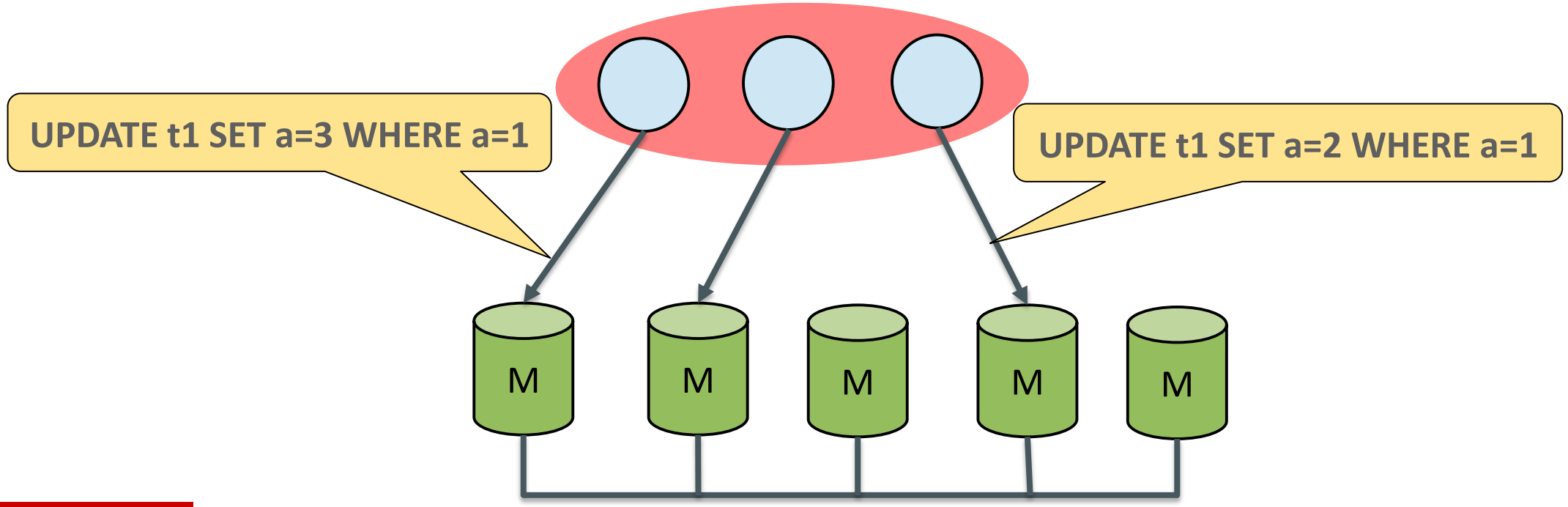
# マルチマスターモード：どこでも更新可能

- 異なるサーバー上での2つのトランザクションは、同じデータを更新可能
- 競合が検出された場合、自動的に対処される
  - 先にコミットしたトランザクションが優先される



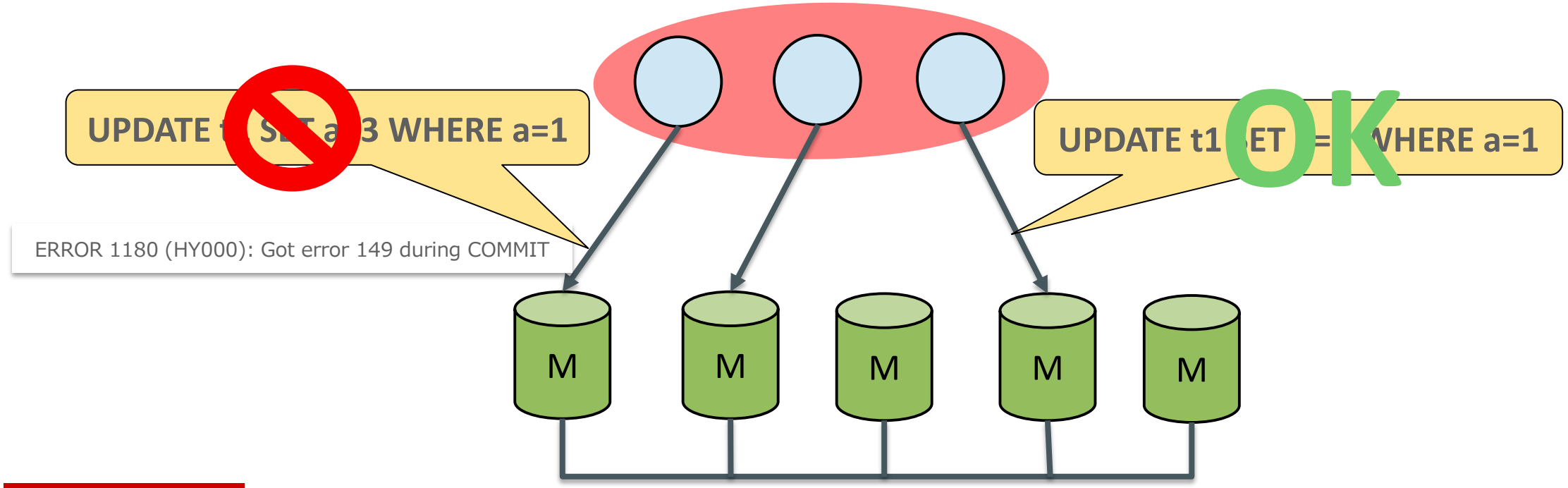
# マルチマスターモード：どこでも更新可能

- 異なるサーバー上での2つのトランザクションは、同じデータを更新可能
- 競合が検出された場合、自動的に対処される
  - 先にコミットしたトランザクションが優先される



# マルチマスターモード：どこでも更新可能

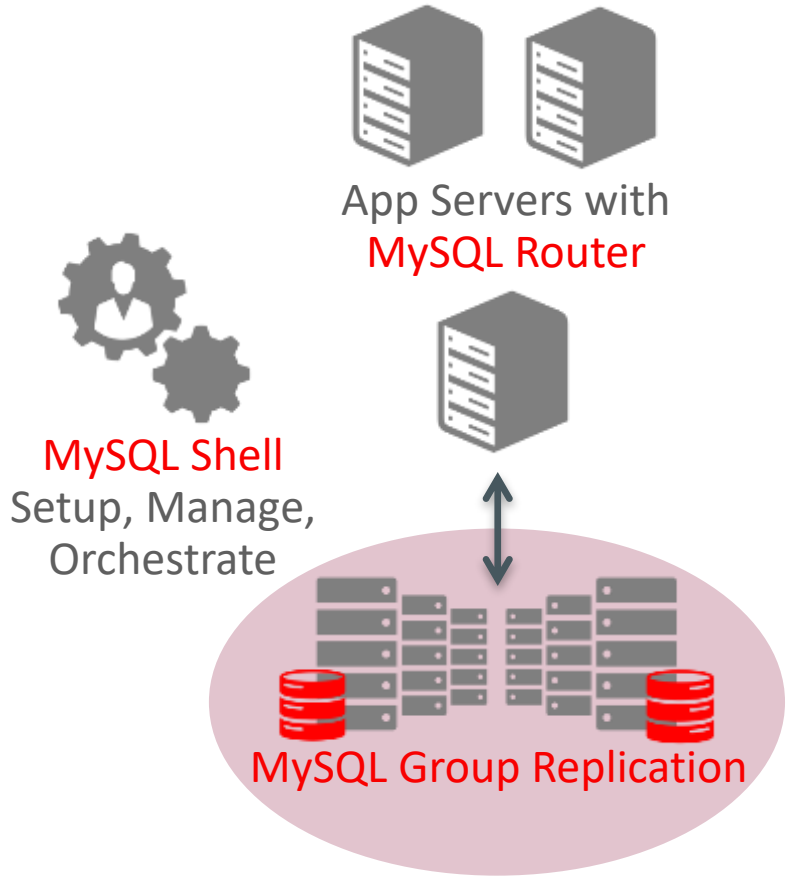
- 異なるサーバー上での2つのトランザクションは、同じデータを更新可能
- 競合が検出された場合、自動的に対処される
  - 先にコミットしたトランザクションが優先される



# MySQL InnoDB Cluster

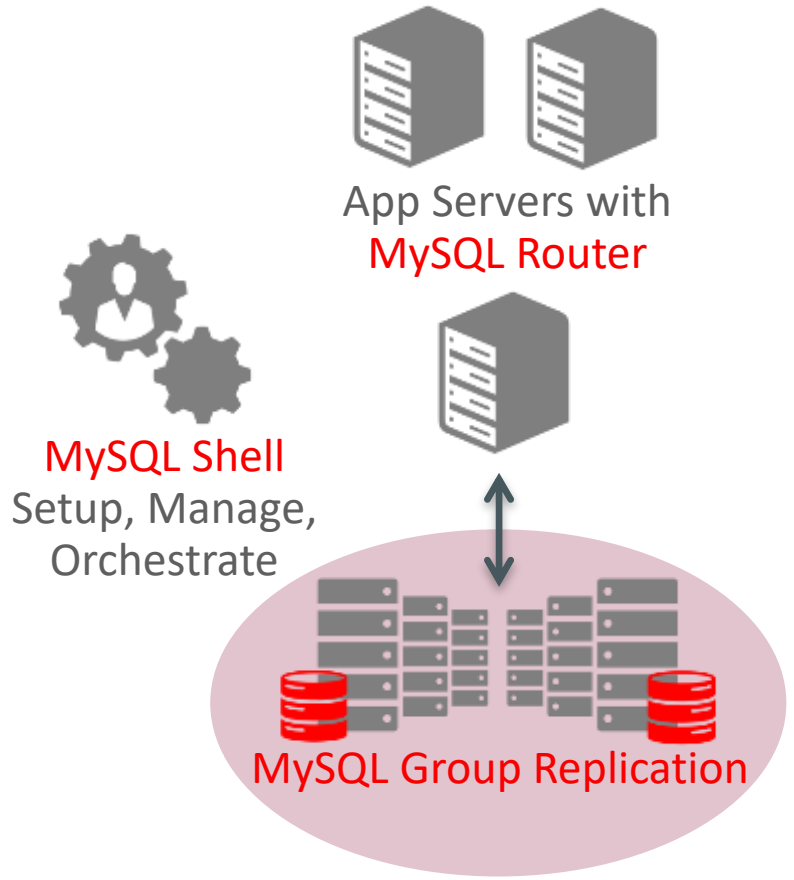
# MySQL InnoDB Clusterとは？

- 以下のコンポーネントの組み合わせから構成される MySQLの高可用性フレームワーク
  - MySQL Group Replication
    - DBの読み取り拡張性、自動フェイルオーバーを提供
  - MySQL Router
    - アプリ接続先の自動フェイルオーバーを提供
  - MySQL Shell
    - グループ・レプリケーション環境の構築、設定、Routerの設定



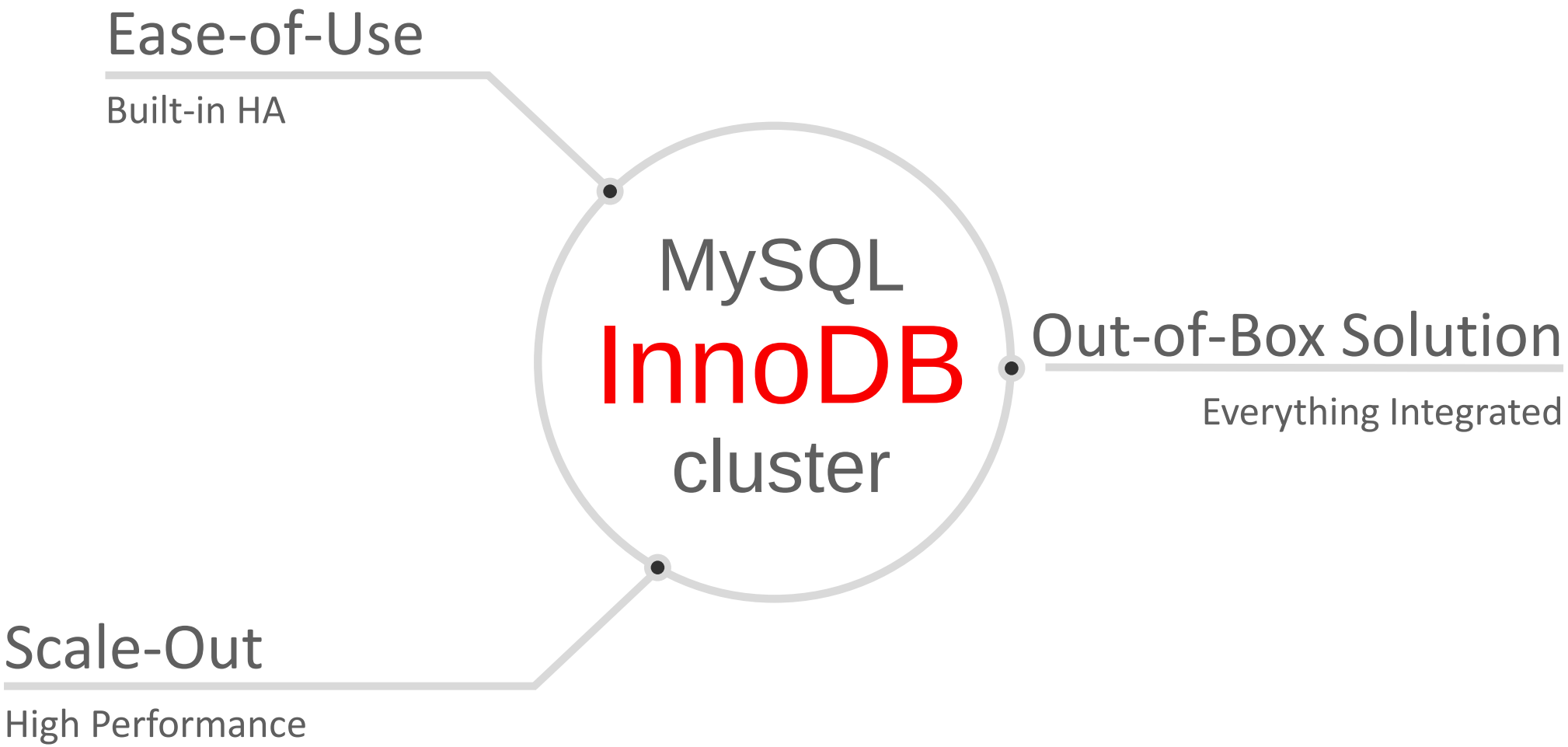
# MySQL InnoDB Clusterとは？

- 2017年4月12日 GA
  - 以下の製品を個別にインストールすることで使用可能
    - MySQL 5.7.21 (2018-01-15)
    - MySQL Router 2.1.4 (2017-07-24)
    - MySQL Shell 1.0.11 (2017-11-30)



# MySQL InnoDB Clusterのチュートリアル

- 以下のセミナー資料に、コマンド付きのMySQL InnoDB Clusterのチュートリアルが含まれています
  - MySQLの新しい高可用性構成<br>MySQLグループ・レプリケーションとMySQL InnoDB Cluster<br>https://www.mysql.com/jp/why-mysql/presentations/mysql-innodb-cluster-201704-ja/
- チュートリアル動画もあります
  - https://youtu.be/RfyxlGS4Zks



## Ease-of-Use

- 15分でインストール, HA ,スケールアウト設定が可能
- MySQLユーザーの為のシングルインターフェイス
- 簡単にセットアップ, スケールアウト, 管理 & モニタリング
- 優れた品質

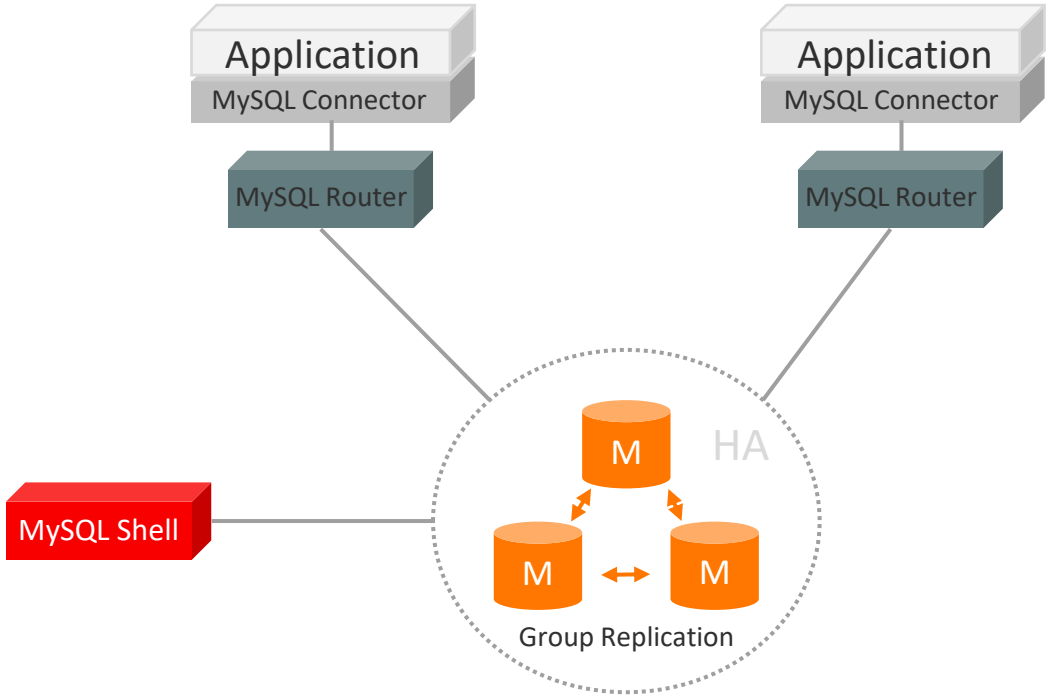
## Out-of-Box Solution

- 統合ソリューション vs. 個別のコンポーネント
- 設計& 開発済み環境を同時に提供
- 検証済み環境を同時に提供
- 管理及び監視環境を同時に提供

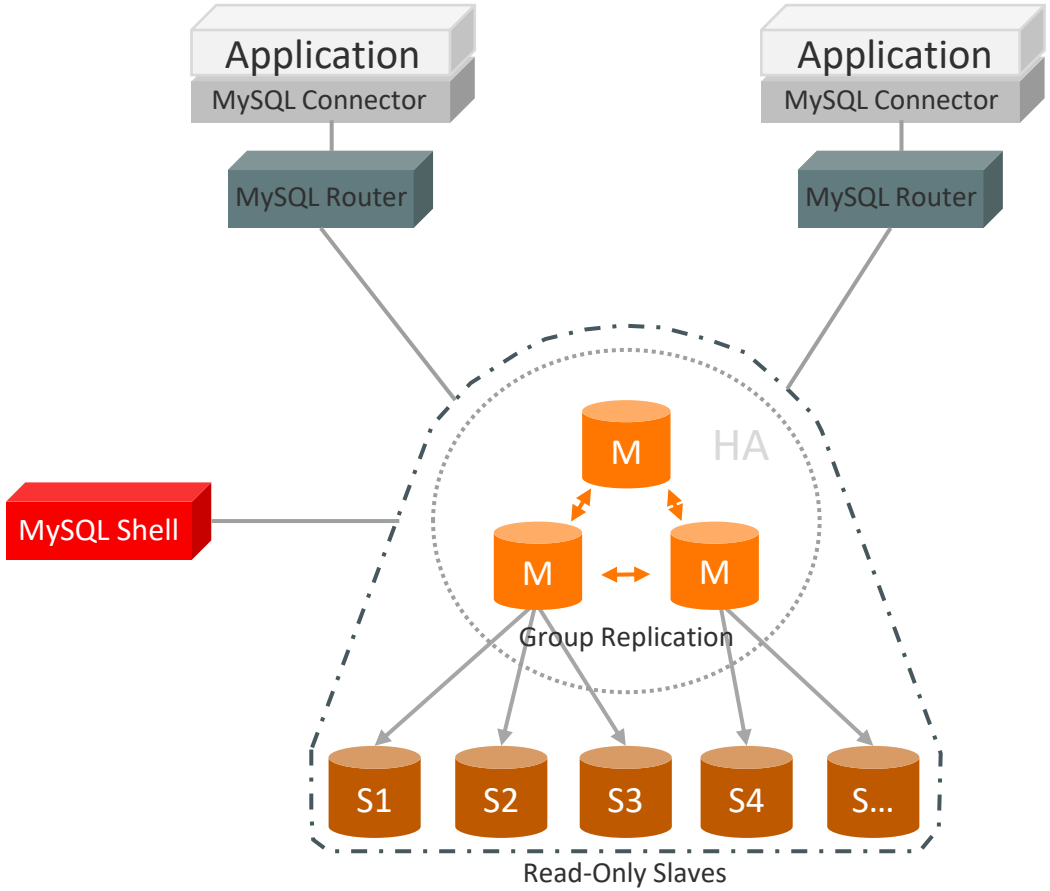
## Scale-Out

- ワールドクラスの性能を維持
- 自動フェイルオーバー含め信頼性のあるHAをサーバー側で提供
- 参照処理の拡張:レプリケーション
- 書き込み処理の拡張:シャーディング

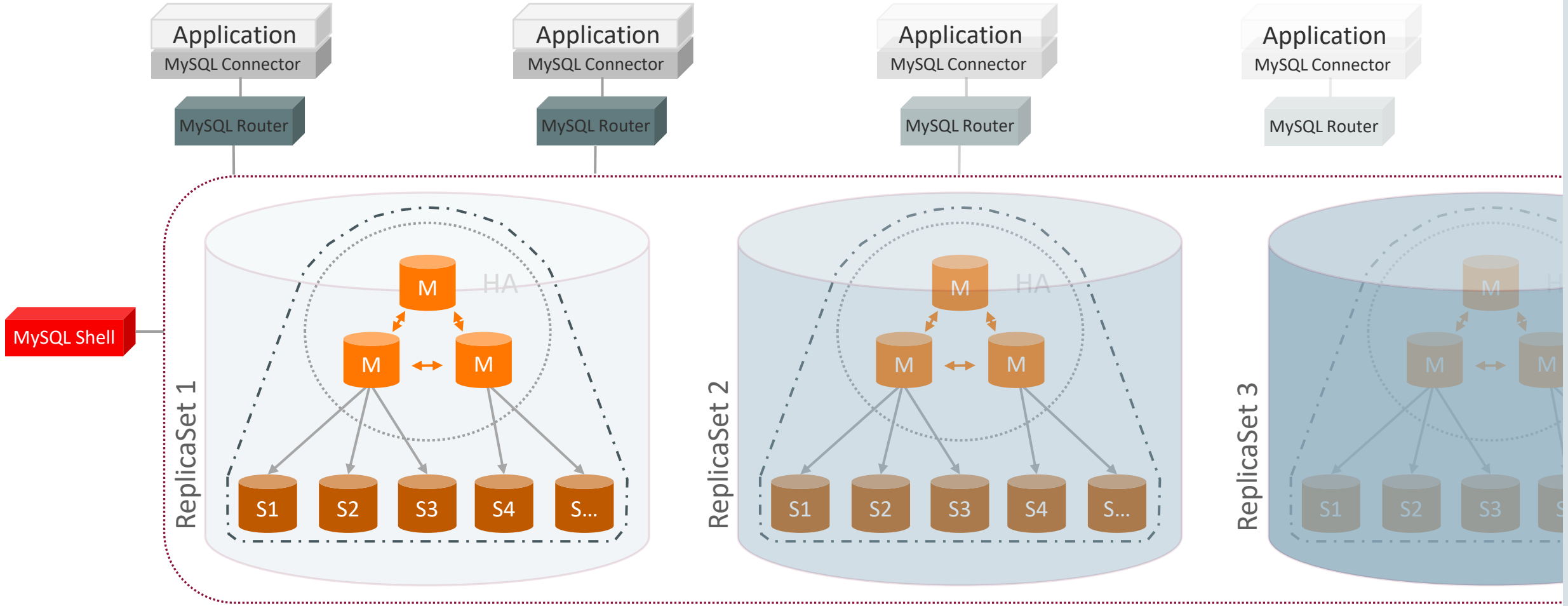
# MySQL InnoDB Cluster: Architecture – Step 1



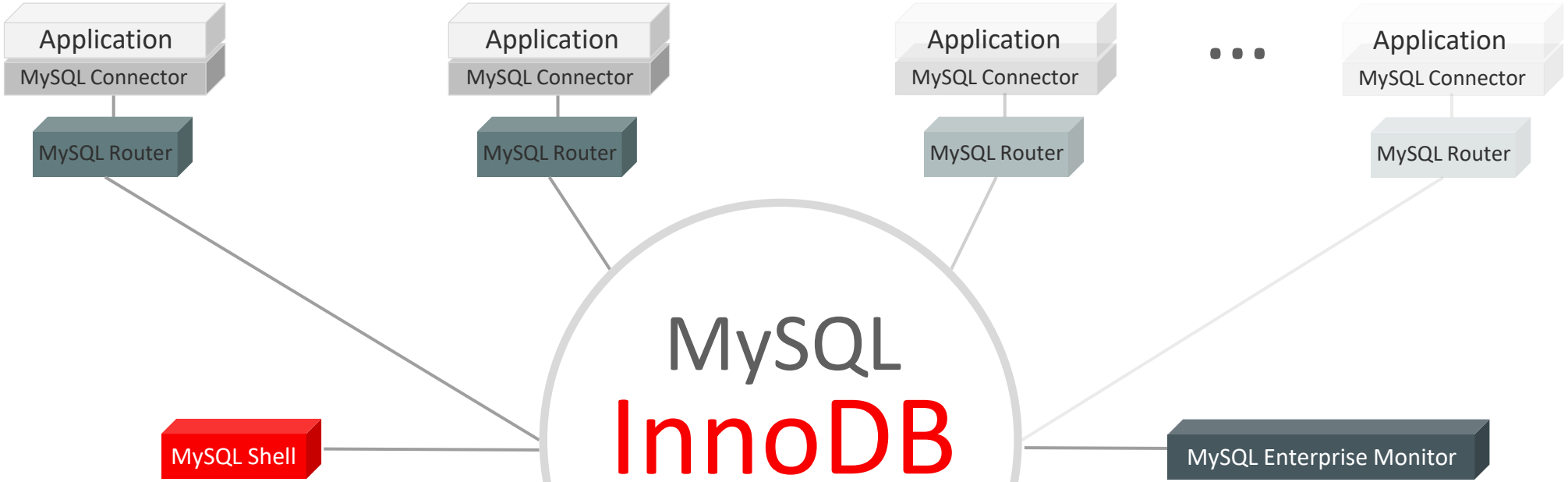
# MySQL InnoDB Cluster: Architecture – Step 2



# MySQL InnoDB Cluster: Architecture – Step 3

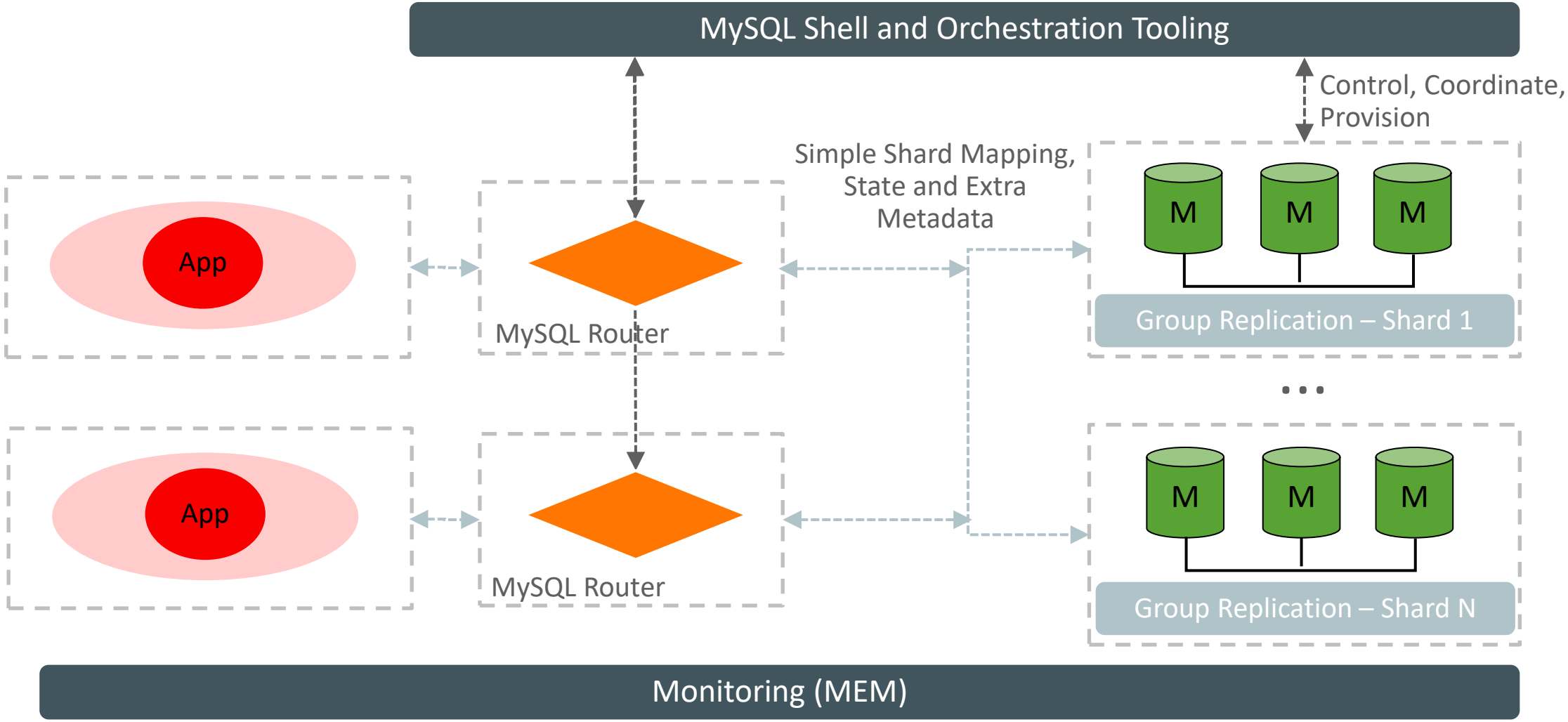


# MySQL InnoDB Cluster: High Level Architecture




```
mysql-js> cluster.status()
 {
   "clusterName": “NewAppCluster",
   "defaultReplicaSet": {
      "status": "Cluster tolerant to up to ONE failure.",
      "topology": {
          “hanode1:3306": {
              "address": “hanode1:3306",
              "status": "ONLINE",
              "role": "HA",
              "mode": "R/W",
```

# 参考) MySQL InnoDB Cluster: The End Goal



# MySQL InnoDB Cluster: ゴール

## 単一製品: MySQL

- 全てのコンポーネントを同時に開発
- 全てのコンポーネントを同時に検証
- 一つのパッケージとして提供

## 近代的な柔軟性

- C++ 11 (ISO標準 ISO/IEC 14882:2011)
- Protocol Buffers
- 開発フレンドリー

## 容易な利用

- シングルクライアント: MySQL Shell
- 容易なパッケージング
- 同種のサーバー群

## スケールアウト

- シャード・クラスター
- Nレプリカセットのフェデレーテッド構成
- 各レプリカセットによるシャードの管理

デモ: https://www.youtube.com/watch?v=JWy7ZLXxtZ4

GA

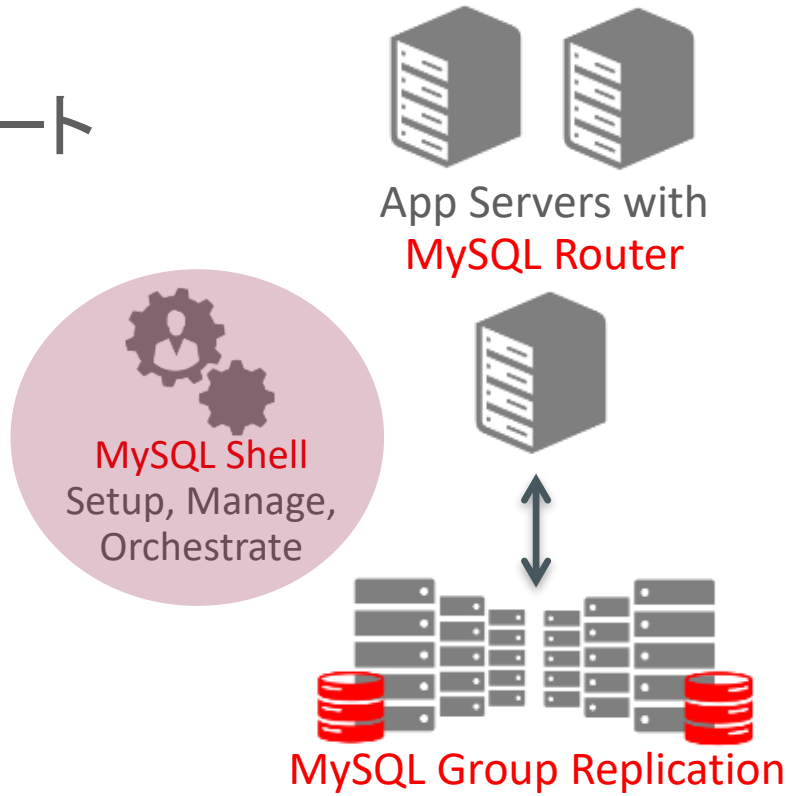
MySQL Shell 1.0.9 GA (2017-04-12)

# MySQL Shell

## すべての運用管理タスクのための、統一された単一クライアント

- 多言語対応: JavaScript, Python, and SQL
- ドキュメントとリレーショナルモデルの両方をサポート
- 開発と管理用に完全なAPIを提供

*”MySQL Shell provides the developer and DBA with a single intuitive, flexible, and powerfull interface for all MySQL related tasks!”*




```
[root@misc01 admin]# mysqlsh --help | egrep -i "Start in"
  --sql                 Start in SQL mode using a node session.
  --sqlc                Start in SQL mode using a classic session.
  --js                  Start in JavaScript mode.
  --py                  Start in Python mode.
[root@misc01 admin]#
```

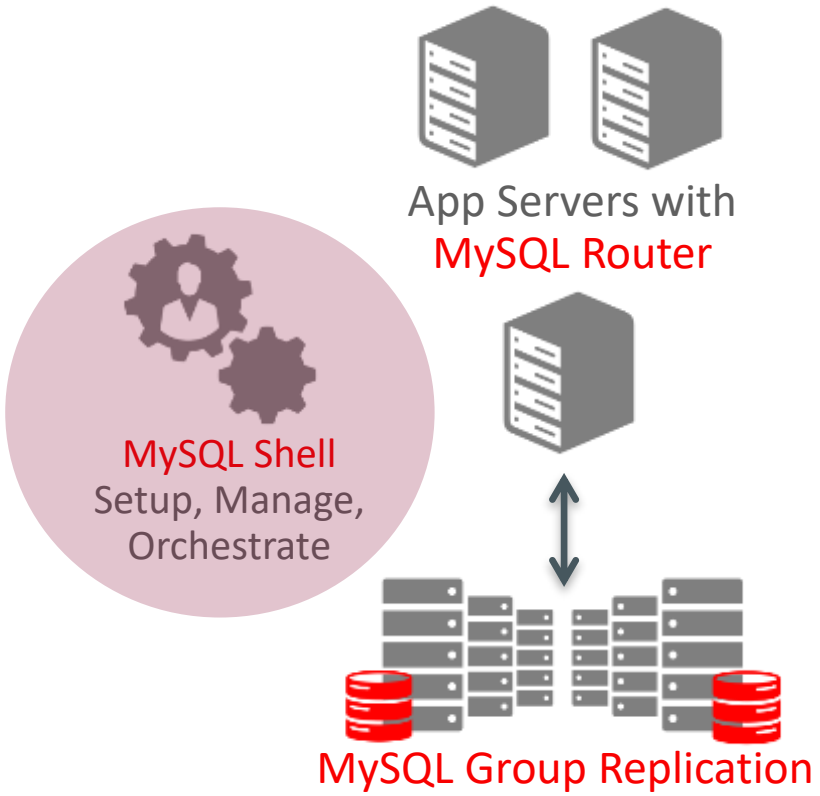
# MySQL Shell: 管理用API

## データベース管理者向けインターフェース

GA

MySQL Shell 1.0.9 GA (2017-04-12)

- `mysql-js> dba.help()`
- グローバル変数 'dba'がMySQLの管理用APIにアクセスする為に使用可能
- DBA管理オペレーション
  - Manage MySQL InnoDB clusters
    - クラスター作成
    - MySQLインスタンスの構築
    - クラスターの状況を確認可能
    - MySQLインスタンスの開始・停止
    - MySQLインスタンスの検証 …



```
mysql-js> dba.help()
The global variable 'dba' is used to access the MySQL AdminAPI functionality
and perform DBA operations.It is used for managing MySQL InnoDB clusters.

The following properties are currently supported.
 - verbose Enables verbose mode on the Dba operations.
The following functions are currently supported.

 - createCluster              Creates a MySQL InnoDB cluster.
 - deleteLocalInstance        Deletes an existing MySQL Server instance on localhost.
 - deployLocalInstance        Creates a new MySQL Server instance on localhost.
 - dropMetadataSchema         Drops the Metadata Schema.
 - getCluster                 Retrieves a cluster from the Metadata Store.
 - help                       Provides help about this class and it's members
 - killLocalInstance          Kills a running MySQL Server instance on localhost.
 - resetSession               Sets the session object to be used on the Dba operations.
 - startLocalInstance         Starts an existing MySQL Server instance on localhost.
 - stopLocalInstance          Stops a running MySQL Server instance on localhost.
 - validateInstance           Validates an instance for usage in Group Replication.

For more help on a specific function use dba.help('<functionName>')
e.g. dba.help('deployLocalInstance')
```

# MySQL Shellの機能拡張

MySQL Shell
Setup, Manage, Orchestrate

GA

ノードの追加, Group Replication設定, Router連携機能を実装

**MySQL Shell – Deploy MySQL Instances**

```
shell> mysqlsh
mysql-js> dba.deployLocalInstance(3306)
mysql-js> dba.deployRemoteInstance('192.168.1.2:3306')
mysql-js> dba.deployRemoteInstance('192.168.1.3:3306')
```

**MySQL Shell – Add MySQL Router**

```
shell> mysqlrouter --bootstrap localhost:3306
shell> mysqlrouter &
shell> mysqlsh --uri root@localhost:6442
```

**MySQL Shell – Create InnoDB Cluster**

```
shell> mysqlsh --uri root@localhost:3306
mysql-js> cluster = dba.createCluster('NewAppCluster')
mysql-js> cluster.addInstance('root@192.168.1.2:3306')
mysql-js> cluster.addInstance('root@192.168.1.3:3306')
```

**MySQL Shell – Check Status**

```
shell> mysqlsh --uri root@localhost:3306
mysql-js> cluster = dba.getCluster()
mysql-js> cluster.status()
```

https://www.youtube.com/watch?v=JWy7ZLXxtZ4

# アジェンダ

1. Oracle MySQL Cloud Service
2. MySQL 8.0 RC 新機能
3. MySQL Group Replication、MySQL InnoDB Cluster
4. **MySQL Enterprise Edition**
5. 参考情報

# MySQL Enterprise Editionとは？

# MySQL Enterprise Edition

ビジネス・クリティカルな環境において、最高レベルの
MySQLスケーラビリティ、セキュリティ、信頼性、
アップタイムを実現し、ビジネス・クリティカルな
環境においてリスクとコストを削減を実現

**MySQL導入の最適化**

**ROIの最適化をサポート**

**ユーザビリティ・顧客満足の向上**

# MySQL Enterprise Edition のサービスカテゴリー

## 拡張機能

- 拡張性
- 高可用性
- 統合認証
- 監査
- 暗号化
- ファイヤーウオール
- 透過的データ暗号化

## 管理ツール

- 監視
- バックアップ
- 開発
- 管理
- マイグレーション

## サポート

- 技術サポート
- コンサルティングサポート
- オラクル製品との動作保証

| | MySQL Editions | | |
|---|---|---|---|
| | Standard Edition | Enterprise Edition | Cluster CGE |
| **機能概要** | | | |
| MySQL Database | ✔ | ✔ | ✔ |
| MySQL Connectors | ✔ | ✔ | ✔ |
| MySQL Replication | ✔ | ✔ | ✔ |
| MySQL Fabric, MySQL Utilities, MySQL Router | | ✔ | ✔ |
| MySQL Partitioning | | ✔ | ✔ |
| Storage Engine: MyISAM, InnoDB | ✔ | ✔ | ✔ |
| Storage Engine: NDB (ndbcluster) | | | ✔ |
| MySQL Workbench SE/EE* | ✔ | ✔ | ✔ |
| MySQL Enterprise Monitor* | | ✔ | ✔ |
| MySQL Enterprise Backup* | | ✔ | ✔ |
| MySQL Enterprise Authentication（外部認証サポート）* | | ✔ | ✔ |
| MySQL Enterprise TDE（Transparent Data Encryption）* | | ✔ | ✔ |
| MySQL Enterprise Encryption（非対称暗号化）* | | ✔ | ✔ |
| MySQL Enterprise Firewall（SQLインジェクション対策）* | | ✔ | ✔ |
| MySQL Enterprise Audit（ポリシーベース監査機能）* | | ✔ | ✔ |
| MySQL Enterprise Scalability（スレッドプール）* | | ✔ | ✔ |
| MySQL Enterprise High Availability（HAサポート）* | | ✔ | ✔ |
| Oracle Enterprise Manager for MySQL * | | ✔ | ✔ |
| MySQL Cluster Manager（MySQL Cluster管理）* | | | ✔ |
| MySQL Cluster Geo-Replication | | | ✔ |

*商用版のみで利用可能な追加機能

| | MySQL Editions | | |
|---|---|---|---|
| | Standard Edition | Enterprise Edition | Cluster CGE |
| **Oracle Premium Support** | | | |
| 24時間365日サポート | ✔ | ✔ | ✔ |
| インシデント数無制限 | ✔ | ✔ | ✔ |
| ナレッジベース | ✔ | ✔ | ✔ |
| バグ修正&パッチ提供 | ✔ | ✔ | ✔ |
| コンサルティングサポート | ✔ | ✔ | ✔ |
| **オラクル製品との動作保証** | | | |
| Oracle Linux | ✔ | ✔ | ✔ |
| Oracle VM | ✔ | ✔ | ✔ |
| Oracle Solaris | ✔ | ✔ | ✔ |
| Oracle Enterprise Manager | | ✔ | ✔ |
| Oracle GoldenGate | | ✔ | ✔ |
| Oracle Data Integrator | | ✔ | ✔ |
| Oracle Fusion Middleware | | ✔ | ✔ |
| Oracle Secure Backup | | ✔ | ✔ |
| Oracle Audit Vault and Database Firewall | | ✔ | ✔ |

※最新の対比表は、MySQL Editionsのサイトを参照下さい

# MySQL Enterprise Edition管理ツールと拡張機能概要

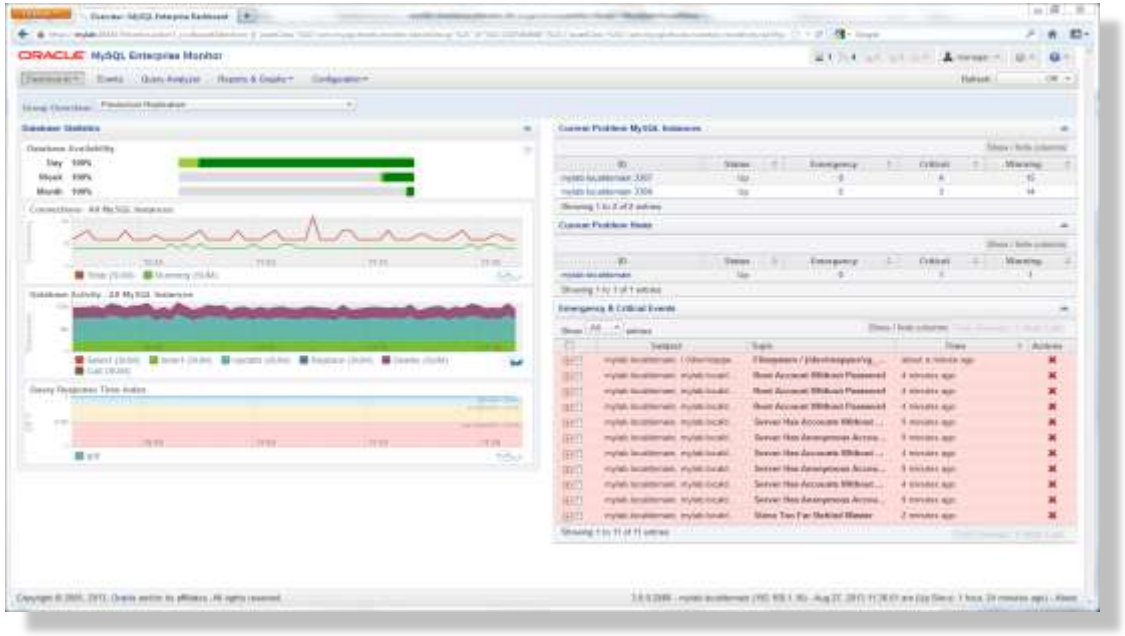
| MySQL Enterprise Edition | |
|---|---|
| MySQL Enterprise Monitor | 複数サーバの一括管理、クエリ性能分析 |
| MySQL Enterprise Backup | 高速なオンラインバックアップ、ポイントインタイムリカバリ |
| MySQL Enterprise Authentication | LDAPやWindows Active Directoryとの外部認証と統合管理 |
| MySQL Enterprise TDE | データベース全体の暗号化（透過的） |
| MySQL Enterprise Encryption | 非対称暗号化(公開鍵暗号)の業界標準機能を提供 |
| MySQL Enterprise Firewall | SQLインジェクション対策 |
| MySQL Enterprise Audit | ユーザ処理の監査、Oracle DBと同じツールで管理可能 |
| MySQL Enterprise Scalability | Thread Poolプラグインによる性能拡張性の向上 |
| Oracle Enterprise Manager for MySQL | Oracle Enterprise ManagerからMySQLを統合管理可能 |
| Oracle Premier Support | 24x7, インシデント無制限、コンサルティングサポート |

# MySQL Enterprise Monitor

- 複数のMySQLサーバを一括監視可能なダッシュボード
- システム中のMySQLサーバやレプリケーション構成を自動的に検出し監視対象に追加
- ルールに基づく監視と警告
- 問題が発生する前に通知
- 問題のあるSQL文の検出、統計情報の分析が可能なQuery Analyzer

参照：MySQL Enterprise Monitor

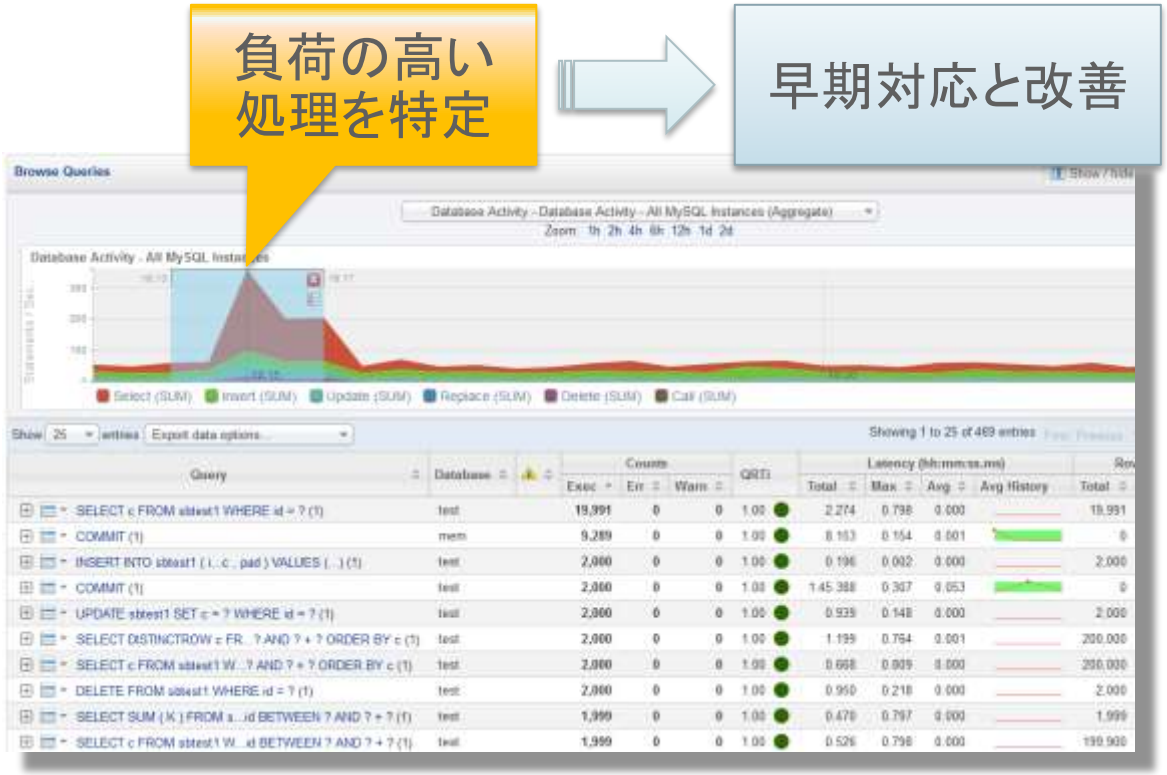
Diagnostic Reportを利用してサポートとの情報共有を容易にする事も可能



"バーチャルなMySQL DBA"

# クエリ解析機能 - MySQL Query Analyzer

- 全てのMySQLサーバの
  全てのSQL文を一括監視
- vmstatなどのOSコマンドやMySQLの
  SHOWコマンドの実行、
  ログファイルの個別の監視は不要
- クエリの実行回数、エラー回数、
  実行時間、転送データ量などを
  一覧表示
- チューニングのための
  解析作業を省力化



# MySQL Enterprise Backup

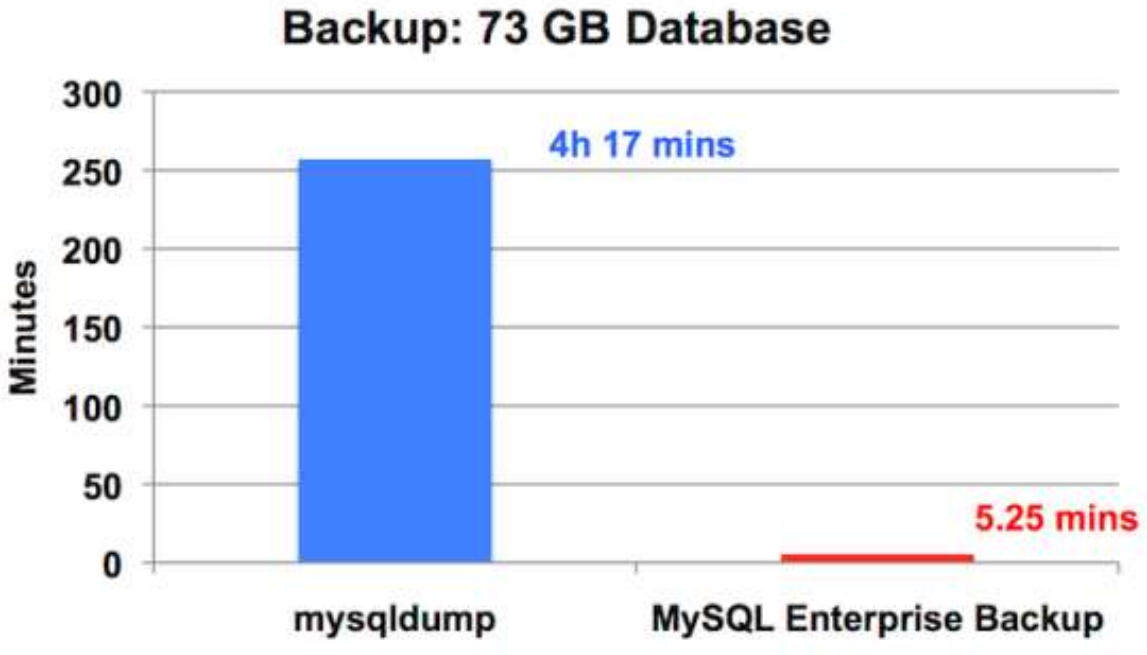
## 高速、オンラインバックアップ & リカバリ

- InnoDBのオンラインバックアップツール
- フル、増分、部分バックアップ(圧縮可能)
- マルチスレッドによる並列バックアップ &リカバリ処理
- クラウドストレージとの直接の連携（S3, Swift API）
- バックアップの暗号化 – AES 256
- Oracle Secure Backupとの連携

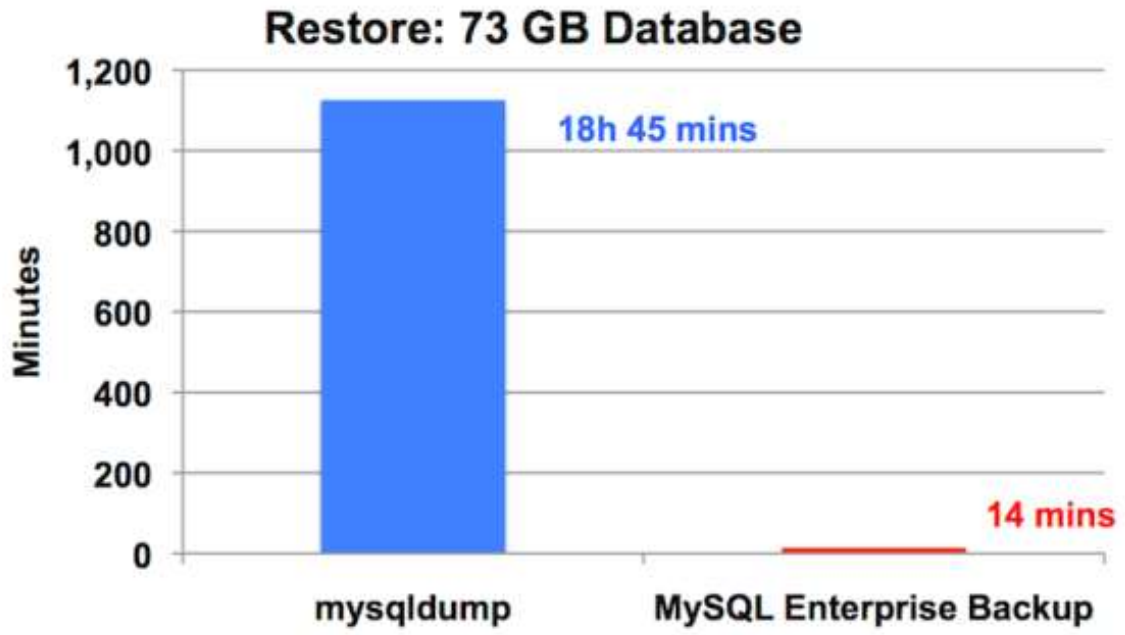
MySQL Enterprise Backup の特徴と利点

# 高速なバックアップとリカバリー

**高速なオンラインバックアップ＆リカバリー処理により、機会損失を最小限に抑える事が可能**

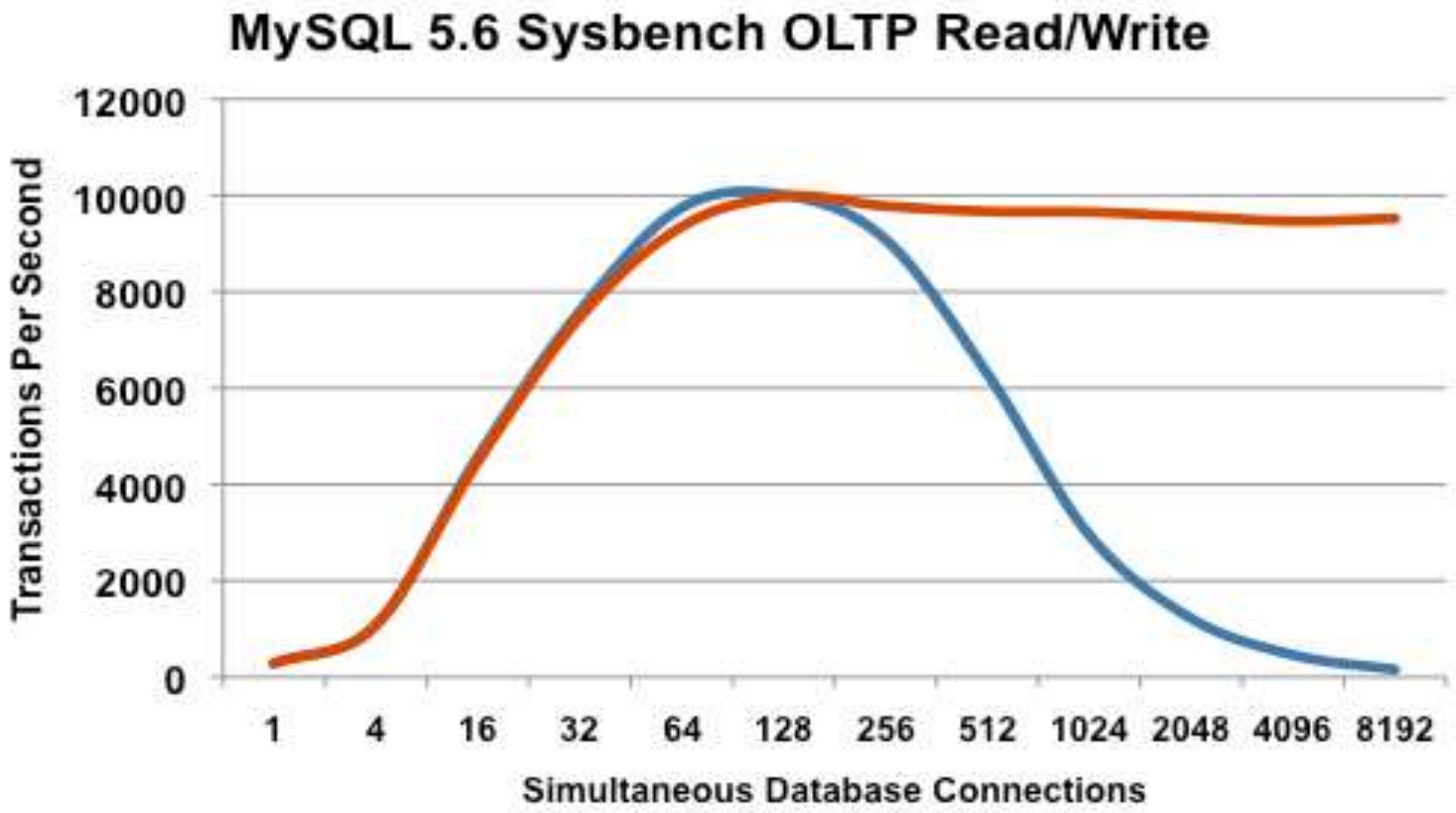



**mysqldumpより49倍速い**




**mysqldumpより80倍速い**

# MySQL Enterprise Scalability : Thread Pool




**MySQL Enterprise Edition**

**Thread Pool有り**

**MySQL Community Edition**

**Thread Pool無し**

MySQL 5.6.11
Oracle Linux 6.3、Unbreakable Kernel 2.6.32
4 sockets、24 cores、48 Threads
Intel(R) Xeon(R) E7540 2GHz CPUs
512GB DDR RAM

**Thread Poolでスケーラビリティが60倍向上**

参照 : MySQL Enterprise Scalability

# MySQL Enterprise Edition 5.7: 統合されたセキュリティ機能

- MySQL Enterprise Firewall
  - SQLインジェクション攻撃をブロック
  - 侵入者を検知
- MySQL Enterprise TDE
  - 保存データの暗号化（透過的）
  - 鍵管理
- MySQL Enterprise Encryption
  - デジタル署名、データバリデーション
  - MySQL KeyRing (Oracle Key Vaultと連携)
- MySQL Enterprise Authentication
  - 外部認証モジュール
    - Microsoft AD, Linux PAMs
- MySQL Enterprise Audit
  - ユーザーアクティビティの監査、法令順守
  - テーブルレベルの監査
- MySQL Enterprise Monitor
  - データベース設定、ユーザー権限、スキーマ、パスワードの変更等のモニタリング
  - MySQL Enterprise Firewall監視
- MySQL Enterprise Backup
  - セキュアなバックアップ、AES 256暗号化

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

# MySQL Enterprise Edition 5.7: 統合されたセキュリティ機能

- MySQL Enterprise Edition を使うことで、セキュリティ対策を強化できます
- セキュリティ要件が厳しいシステムでも、
  是非MySQL Enterprise Editionの採用をご検討ください！！

# 技術サポート

# MySQL Enterprise Support

- 最大のMySQLのエンジニアリングおよびサポート組織
- MySQL開発チームによるサポート
- 29言語で世界クラスのサポートを提供
- メンテナンス・リリース、バグ修正、パッチ、アップデートの提供
- 24時間x365日サポート
- 無制限サポート・インシデント
- MySQL コンサルティング・サポート

**～リモートDBAとして、是非ご活用ください！！～**

Get immediate help for any MySQL issue, plus expert advice

# MySQL Supportの特徴

- 「パフォーマンス・チューニング」や「SQLチューニング」まで通常サポートの範囲内
  - コンサルティングサポートが含まれており、「クエリ・レビュー」、「パフォーマンス・チューニング」、「レプリケーション・レビュー」、「パーティショニング・レビュー」などに対応可能
  - 詳細はこちらを参照下さい http://www-jp.mysql.com/support/consultative.html
- ソースコードレベルでサポート可能
  - ほとんどのサポートエンジニアがソースを読めるため、対応が早い開発エンジニアとサポートエンジニアも密に連携している

サポート、コンサルテーティブ・サポートは共に回数制限が無い為、リモートDBAとして活用頂く事で、自社内の調査・検証工数を大幅に削減する事が可能。TCO削減が可能です。

# MySQL Supportの特徴

- 物理サーバー単位課金
  - CPU数、コア数に依存しない価格体系
  - 4CPUまで（コア数は制限無し）同一料金、5CPU以上の価格は営業問合せ
- コミュニティ版バイナリに対してもサポートを提供可能
  - サブスクリプションを契約することで、バイナリを入れ替えずにサポートを受けられる（バイナリはオラクルが提供しているものをご使用ください）
  - 商用版の機能を使用する場合のバイナリ入れ替えの必要性については、P22参照
  - Oracle CloudのMySQL Cloud Service以外のDBaaSはサポート対象外
- オラクルのライフタイムサポート
  - 詳細はこちらを参照下さい
    http://www.oracle.com/jp/support/lifetime-support/index.html
    http://www-jp.mysql.com/support/

# Oracle製品との動作保証

- Oracle Linux
- Oracle VM
- Oracle Solaris
- Oracle Clusterware
- Oracle Secure Backup
- Oracle Enterprise Manager
- Oracle Fusion Middleware
- Oracle GoldenGate
- Oracle Audit Vault & Database Firewall
- MyOracle Online Support

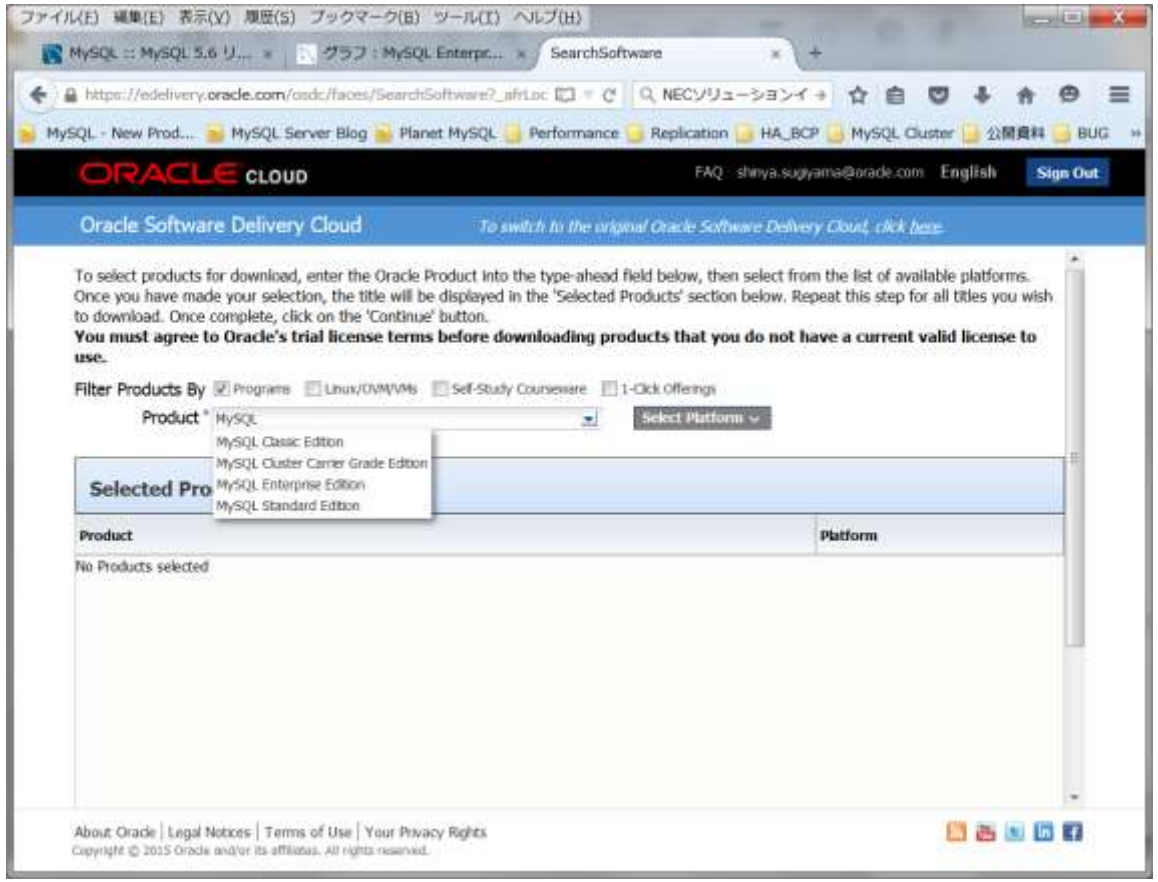
**MySQL Integrates into your Oracle Environment**

ORACLE

# MySQL Enterprise Editionの試用

# MySQL Enterprise Edition & Cluster CGEの試用

## 30日間トライアル




- Oracle Software Delivery Cloud
  http://edelivery.oracle.com/

- 製品パックを選択:
  "Product" にMySQLと入力し、
  OSを選択し"Continue"
- 製品マニュアル
  http://dev.mysql.com/doc/index-enterprise.html

# アジェンダ

1. Oracle MySQL Cloud Service
2. MySQL 8.0 RC 新機能
3. MySQL Group Replication、MySQL InnoDB Cluster
4. MySQL Enterprise Edition
5. **参考情報**

# MySQL 8.0の参考資料

- MySQL 8.0 CTE ＆ Window関数
  - https://www.mysql.com/jp/why-mysql/presentations/mysql-80-cte-window-function-201705-ja/
- MySQL 8.0 Replication改善点
  - https://www.mysql.com/jp/why-mysql/presentations/mysql-80-replication-201705-ja/

# MySQLのイベント情報

- MySQLのイベント情報掲載ページ
  - https://www.mysql.com/jp/news-and-events/events/
- 直近のイベント
  - 2/7（水）名古屋：MySQL 5.7入門 チューニング基礎編、SQLチューニング編
    - https://atnd.org/events/93875
  - 2/9（金）大阪：MySQL 5.7入門 チューニング基礎編、SQLチューニング編
    - https://atnd.org/events/93876
  - 2/28（水） 名古屋：MySQLバージョンアップの基礎知識
    - https://atnd.org/events/94310
  - 3/2（金） 大阪：MySQLバージョンアップの基礎知識
    - https://atnd.org/events/94309

# Oracle OpenWorld 2017の発表資料（英語資料）

- セッション情報確認＆ダウンロードURL
  - https://events.rainfocus.com/catalog/oracle/oow17/catalogoow17?search=mysql
- 資料の一例
  - CON7309 : MySQL 8.0 : What's New in the Optimizer
    - CTEの使用例、ヒストグラム使用例、新しいコストモデルの使用例、など
  - CON7292 : Using MySQL Containers
    - 公式MySQLコンテナの紹介、など
  - CON3081 : Using MySQL Flexible Schema (Document Store JSON) for IoT
    - JSONデータをMySQLに格納して活用する方法、など
  - HOL7299 : MySQL Performance Tuning 101
    - 基本的なチューニング方法、など

# Integrated Cloud

## Applications & Platform Services


ORACLE®