ORACLE

MySQLをベクトルデータベースとして使用する方法と MySQLのマネージドサービスを無料で使用する方法

オープンソースカンファレンス2025 Osaka

山﨑 由章 / Yoshiaki YAMASAKI
MySQL Master Principal Solution Engineer & MySQL Cloud Evangelist
MySQL Global Business Unit
日本オラクル株式会社

MySQLとは?

世界でもっとも普及している、オープンソースデータベース

LAMPスタックの"M"

- Linux + Apache + **MySQL** + PHP/Perl/Python
- Webアプリケーションを開発する時のデファクトスタンダード

マルチプラットフォーム対応

- Windows, Linux, macOS

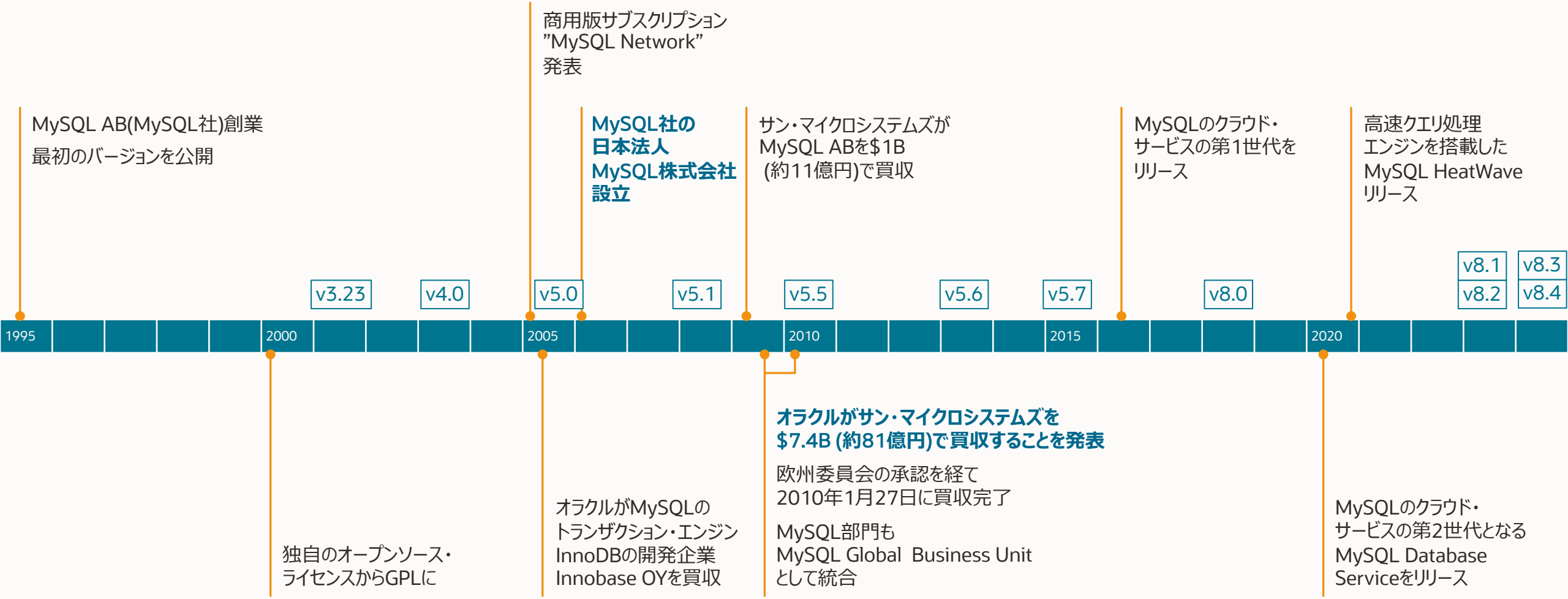
高性能、軽量、高信頼

- 特に同時多発的に行われる更新処理の対応が得意

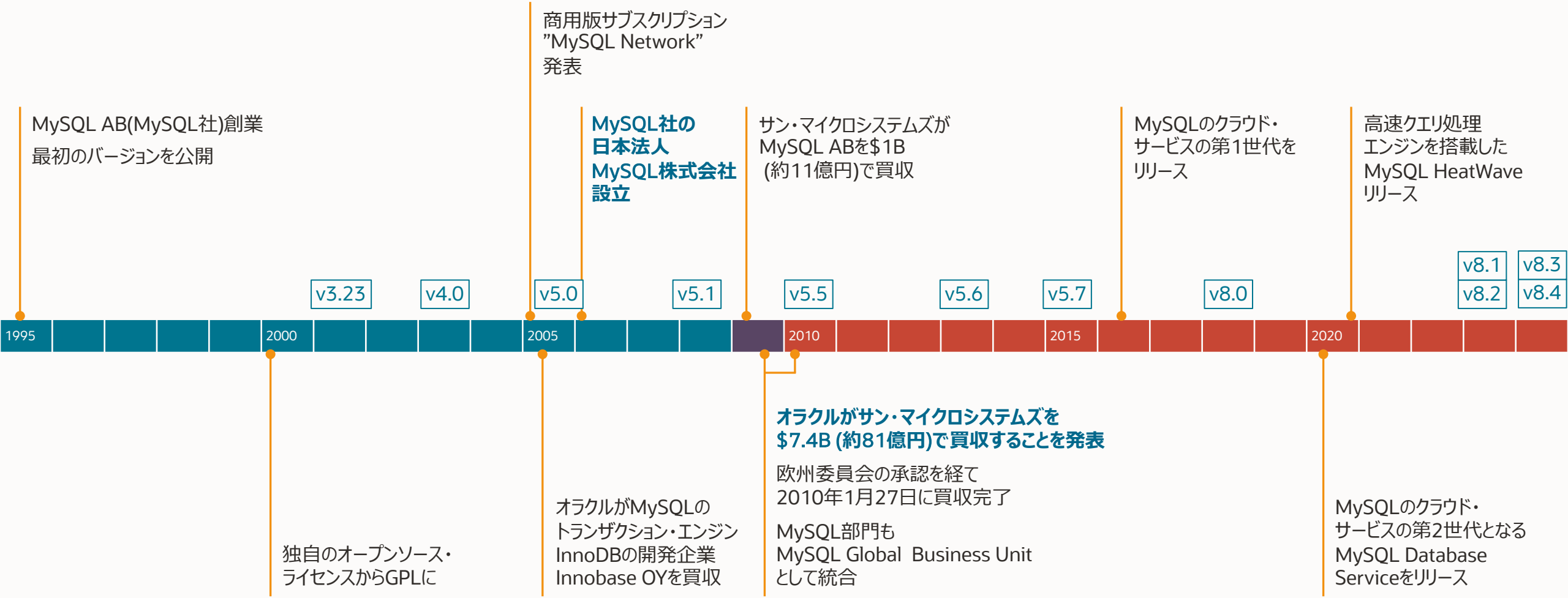
導入や運用が簡単

- ダウンロードからデータベース起動まで15分以内
- 管理不要なシンプルさ

MySQLの道のり



MySQLの道のり



柔軟なMySQLの利用方法

MySQLサーバーは全て共通のソースコードのためハイブリッド構成も可能

MySQLを自社で運用管理

コミュニティ版MySQL

- レプリケーションや透過的暗号化など運用に重要な機能を実装
- GPLv2
  *OpenSSLに関する追加条項あり

商用版MySQL

- サポートサービスや高度なセキュリティ機能を提供
- SE/EE/CGEが選択可能

MySQLのマネージドサービス

HeatWave MySQL Database Service

- MySQLチームが100%開発・提供するクラウド・サービス
- 高速データ分析エンジン&機械学習エンジンを組み込み

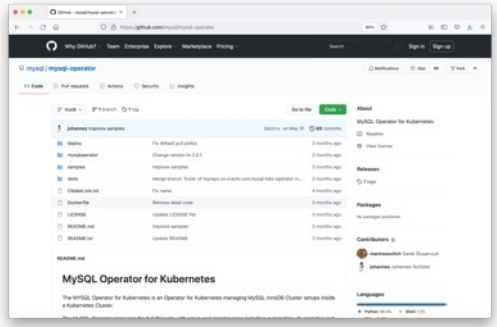
15倍	18倍	35倍
Redshift より高速	Snowflake より高速	BigQuery より高速

TPC-H 500TB MySQL HeatWave Lakehouseとのクエリ処理性能比較

クラウドネイティブなMySQL

MySQL Operator for k8s

- MySQLの高可用性構成をKubernetes上に構築&運用管理
- 商用版MySQL EEがベース



いずれの利用方法でもMySQL開発チームと連携した専門部隊によるサポートサービスをご利用いただけます

ベクトルデータベース(ベクトルストア) とは？

ベクトルデータベース(ベクトルストア) とは？

- データを高次元のベクトルとして保存し、活用できるデータベース
- データに対してエンベディングを生成し(データの意味を考慮してベクトル化し)、
  ベクトル間の距離を計算することにより、従来のデータベースではできなかった意味に基づく検索が可能になる
- LLM(大規模言語モデル) を活用する時に、LLMが知らない知識を補完する
  RAG(Retrival-Augmented Generation) を実現するために活用されることもある

ベクトルデータベース(ベクトルストア) とは？

本セッションでは
この用途について解説

- データを高次元のベクトルとして保存し、活用できるデータベース
- データに対してエンベディングを生成し(データの意味を考慮してベクトル化し)、
  ベクトル間の距離を計算することにより、従来のデータベースではできなかった意味に基づく検索が可能になる
- LLM(大規模言語モデル) を活用する時に、LLMが知らない知識を補完する
  RAG(Retrival-Augmented Generation) を実現するために活用されることもある

MySQLをベクトルデータベースとして使用するには？

MySQLをベクトルデータベースとして使用するには？

- データを高次元のベクトルとして保存する

- データに対してエンベディングを生成し(データの意味を考慮してベクトル化し)、
  ベクトル間の距離を計算することにより、従来のデータベースではできなかった意味に基づく検索が可能

MySQLをベクトルデータベースとして使用するには？

- データを高次元のベクトルとして保存する

  => ベクトルデータを保存できるベクトルデータ型が必要

- データに対してエンベディングを生成し(データの意味を考慮してベクトル化し)、
  ベクトル間の距離を計算することにより、従来のデータベースではできなかった意味に基づく検索が可能

  => ベクトル間の距離を計算する関数が必要

MySQLをベクトルデータベースとして使用するには？

- データを高次元のベクトルとして保存する

  => ベクトルデータを保存できるベクトルデータ型が必要

  MySQL 9.0以降でベクトルデータ型が使える

- データに対してエンベディングを生成し(データの意味を考慮してベクトル化し)、
  ベクトル間の距離を計算することにより、従来のデータベースではできなかった意味に基づく検索が可能

  => ベクトル間の距離を計算する関数が必要

MySQLをベクトルデータベースとして使用するには？

- データを高次元のベクトルとして保存する

  => ベクトルデータを保存できるベクトルデータ型が必要

  MySQL 9.0以降でベクトルデータ型が使える

- データに対してエンベディングを生成し(データの意味を考慮してベクトル化し)、

  ベクトル間の距離を計算することにより、従来のデータベースではできなかった意味に基づく検索が可能

  => ベクトル間の距離を計算する関数が必要

  HeatWave MySQLを使用すると、DISTANCE関数でベクトル間の距離を計算可能

MySQLをベクトルデータベースとして使用するには？

- データを高次元のベクトルとして保存する

  => ベクトルデータを保存できるベクトルデータ型が必要

  MySQL 9.0以降でベクトルデータ型が使える

- データに対してエンベディングを生成し(データの意味を考慮してベクトル化し)、
  ベクトル間の距離を計算することにより、従来のデータベースではできなかった意味に基づく検索が可能

  => ベクトル間の距離を計算する関数が必要

  HeatWave MySQLを使用すると、DISTANCE関数でベクトル間の距離を計算可能

  HeatWave MySQLではML_EMBED_ROW/TABLEルーチンでDB内でエンベディングも生成可能

MySQLをベクトルデータベースとして使用するには？

- 外部のLLMを使ってエンベディングを生成したり、ベクトル間の距離をアプリケーション側で求めるのであれば、MySQL 9.0以降をベクトルデータベースとして使用可能
- ベクトル間の距離計算やエンベディングの生成までMySQL側で行いたい場合は、HeatWave MySQLを使用する

MySQLでベクトルデータを扱うための機能

ベクトルデータを扱うための関数 (MySQL 9.0以降で使用可能)

- STRING_TO_VECTOR関数：文字列データをベクトルデータに変換する関数
- VECTOR_TO_STRING関数：ベクトルデータを文字列データに変換する関数
- VECTOR_DIM関数：ベクトルデータの次元を求める関数

ベクトルデータを扱うための関数 (MySQL 9.0以降で使用可能)

- STRING_TO_VECTOR関数：文字列データをベクトルデータに変換する関数
- VECTOR_TO_STRING関数：ベクトルデータを文字列データに変換する関数
- VECTOR_DIM関数：ベクトルデータの次元を求める関数

外部のLLMを使って生成されたベクトルデータは
STRING_TO_VECTOR関数を使ってベクトルデータ型に変換すれば、MySQLに格納可能

ベクトルデータ型 (MySQL 9.0以降で使用可能)

データ型の定義方法：VECTOR(*N*)

- ”N” には次元を設定する (最大値は16383)
- ベクトルデータ型の列には、主キー、外部キー、ユニークキー、パーティショニングのキーは設定できない

※ 内部的には、ベクトルデータは4バイトのfloat型の配列として保持される

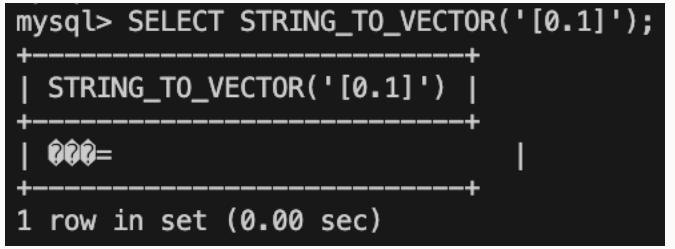
実行例：STRING_TO_VECTOR 関数、VECTOR_TO_STRING 関数

```
mysql> SELECT STRING_TO_VECTOR('[0.1]');
+---------------------------+
| STRING_TO_VECTOR('[0.1]') |
+---------------------------+
| □□□=                      |
+---------------------------+
1 row in set (0.00 sec)

mysql> SELECT STRING_TO_VECTOR('[0.1, 0.01]');
+--------------------------------+
| STRING_TO_VECTOR('[0.1, 0.01]') |
+--------------------------------+
| □□□=
□#<                            |
+--------------------------------+
1 row in set (0.00 sec)

mysql> SELECT VECTOR_TO_STRING(STRING_TO_VECTOR('[0.1, 0.01]'));
+-----------------------------------------------------+
| VECTOR_TO_STRING(STRING_TO_VECTOR('[0.1, 0.01]')) |
+-----------------------------------------------------+
| [1.00000e-01,1.00000e-02]                           |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

※”□”部分は文字化け発生(バイナリデータであるため)



実行例：VECTOR_DIM 関数

```
mysql> SELECT VECTOR_DIM(STRING_TO_VECTOR('[1]'));
+---------------------------------------+
| VECTOR_DIM(STRING_TO_VECTOR('[1]') ) |
+---------------------------------------+
|                                     1 |
+---------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT VECTOR_DIM(STRING_TO_VECTOR('[0.1, 0.2]'));
+-------------------------------------------+
| VECTOR_DIM(STRING_TO_VECTOR('[0.1, 0.2]') ) |
+-------------------------------------------+
|                                         2 |
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT VECTOR_DIM(STRING_TO_VECTOR('[0.99, 0.2, 0]'));
+-----------------------------------------------+
| VECTOR_DIM(STRING_TO_VECTOR('[0.99, 0.2, 0]') ) |
+-----------------------------------------------+
|                                             3 |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

実行例：ベクトルデータ型

```
mysql> CREATE TABLE vec (id INT PRIMARY KEY AUTO_INCREMENT, vec VECTOR(2));
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO vec VALUES(1, STRING_TO_VECTOR('[0.1, 0.1]'));
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO vec VALUES(2, STRING_TO_VECTOR('[0.2, 0.2]'));
Query OK, 1 row affected (0.01 sec)

mysql> SELECT id, VECTOR_TO_STRING(vec) as vector from vec;
+----+---------------------------+
| id | vector                    |
+----+---------------------------+
|  1 | [1.00000e-01,1.00000e-01] |
|  2 | [2.00000e-01,2.00000e-01] |
+----+---------------------------+
2 rows in set (0.00 sec)
```

ベクトルデータを生成するためのルーチン (HeatWave MySQLで使用可能)

- ML_EMBED_ROW関数：1行のデータの対してエンベディングを生成する
- ML_EMBED_TABLEプロシージャー：テーブルのデータに対して(複数行のデータに対して)エンベディングを生成する

エンベディングを生成する時に使用できるLLMはHeatWave MySQL 9.1.2時点で以下の4種類

1. all_minilm_l12_v2 ：英語のエンベディング用
2. multilingual-e5-small ：日本語などの英語以外のエンベディング用
3. cohere.embed-english-v3.0 ：英語のエンベディング用（※）
4. cohere.embed-multilingual-v3.0：日本語などの英語以外のエンベディング用（※）

※OCIのGenerative AIサービスと連携することで使用可能になるLLM

今後サポートされるLLMが追加される計画もあるため、最新情報は以下のドキュメントを参照ください

HeatWave User Guide / Supported Languages, Embedding Models, and LLMs
https://dev.mysql.com/doc/heatwave/en/mys-hw-genai-supported-models.html

備考：OCI Generative AIサービスとの連携方法

- OCI Generative AIサービスと連携するためには、事前設定が必要です
- 設定方法は以下のドキュメントを参照して下さい
  - HeatWave User Guide / Authenticating OCI Generative AI Service
    https://dev.mysql.com/doc/heatwave/en/mys-hw-genai-authenticate-service.html

ベクトルデータの距離を求める関数 (HeatWave MySQLで使用可能)

- DISTANCE関数：2つのベクトルデータの間の距離を求める関数
  - コサイン距離、内積、ユークリッド距離の3種類を計算可能 (COSINE, DOT, EUCLIDEAN)

※VECTOR_DISTANCE関数もあるが、DISTANCE関数のシノニムであるため実態は同じもの

実行例：ML_EMBED_ROW関数、DISTANCE関数

```
mysql> SELECT sys.ML_EMBED_ROW('犬', JSON_OBJECT('model_id', 'multilingual-e5-small')) INTO @v1;
Query OK, 1 row affected (0.64 sec)

mysql> SELECT sys.ML_EMBED_ROW('猫', JSON_OBJECT('model_id', 'multilingual-e5-small')) INTO @v2;
Query OK, 1 row affected (0.57 sec)

mysql> SELECT DISTANCE(@v1, @v2, 'COSINE');
+-----------------------------+
| DISTANCE(@v1, @v2, 'COSINE') |
+-----------------------------+
|          0.10679465532302856 |
+-----------------------------+
1 row in set (0.00 sec)
```

「犬と猫」のCOSINE距離は近い(意味的に近い)

```
mysql>
mysql> SELECT sys.ML_EMBED_ROW('イルカ', JSON_OBJECT('model_id', 'multilingual-e5-small')) INTO @v3;
Query OK, 1 row affected (0.64 sec)

mysql> SELECT DISTANCE(@v1, @v3, 'COSINE');
+-----------------------------+
| DISTANCE(@v1, @v3, 'COSINE') |
+-----------------------------+
|          0.19882500171661377 |
+-----------------------------+
1 row in set (0.00 sec)
```

「犬とイルカ」のCOSINE距離は「犬と猫」よりも遠い

実行例：ML_EMBED_ROW関数、DISTANCE関数

```
mysql> SELECT sys.ML_EMBED_ROW('小屋', JSON_OBJECT('model_id', 'multilingual-e5-small')) INTO @v4;
Query OK, 1 row affected (0.68 sec)

mysql> SELECT DISTANCE(@v1, @v4, 'COSINE');
+------------------------------+
| DISTANCE(@v1, @v4, 'COSINE') |
+------------------------------+
|            0.137731671333313 |
+------------------------------+
1 row in set (0.00 sec)
```

「犬と小屋」のCOSINE距離は「犬と猫」よりは遠いが
「犬とイルカ」よりは近い

実行例：ML_EMBED_TABLEプロシージャー、DISTANCE関数

マンガに対するレビューが投稿され、commentテーブルのcomment列に格納されていることを想定

```
mysql> SELECT * FROM vectordb.comment;
+------------+-------------+---------+---------------------------------------------------------------------------------------------------------------------------+
| comment_id | customer_id | book_id | comment
+------------+-------------+---------+---------------------------------------------------------------------------------------------------------------------------+
|          1 |           1 |       1 | 設定が作り込まれていて面白い。遅効性SFにハマっています！
|          2 |           2 |       2 | 普通の野球漫画とは全然違うけど、ギャンブル＆野球という斬新な設定で面白い。
|          3 |           1 |       3 | 読み始めた時は、こんな風にストーリーが展開するなんて思わなかった。
|          4 |           3 |       4 | まさかの犯人の視点でのスピンオフ！
|          5 |           4 |       3 | 最初は転生ものっぽかったけど、よくある転生ものとは全然違った。
|          6 |           2 |       1 | モブキャラがいない。全キャラが魅力的。バトルものなのにインフレしないままずっと面白いのも凄い。
|          7 |           3 |       5 | 周りで殺人事件起き過ぎw
+------------+-------------+---------+---------------------------------------------------------------------------------------------------------------------------+
7 rows in set (0.00 sec)
```


実行例：ML_EMBED_TABLEプロシージャー、DISTANCE関数

```
mysql> DESC vectordb.comment;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| comment_id  | int         | NO   | PRI | NULL    |       |
| customer_id | int         | YES  |     | NULL    |       |
| book_id     | int         | YES  |     | NULL    |       |
| comment     | varchar(50) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> CALL sys.ML_EMBED_TABLE('vectordb.comment.comment', 'vectordb.comment.comment_vec',
    ->  JSON_OBJECT('model_id', 'multilingual-e5-small'));
Query OK, 0 rows affected (1.25 sec)

mysql> DESC vectordb.comment;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| comment_id  | int          | NO   | PRI | NULL    |       |
| customer_id | int          | YES  |     | NULL    |       |
| book_id     | int          | YES  |     | NULL    |       |
| comment     | varchar(50)  | YES  |     | NULL    |       |
| comment_vec | vector(2048) | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

- 第一引数にはエンベディングを生成したいテキストデータが入っている列を、第二引数にはベクトルデータを格納したい列を指定する
  (「スキーマ名.テーブル名.列名」の形式)
- 同じテーブルを指定した場合は、既存のテーブルに列が追加される

実行例：ML_EMBED_TABLEプロシージャー、DISTANCE関数

・第二引数に存在しないテーブルを指定した場合は、「元テーブルの主キー＋ベクトルデータ型」のテーブルが自動的に作成される

```
mysql> CALL sys.ML_EMBED_TABLE('vectordb.comment.comment', 'vectordb.comment_vec.comment_vec',
    ->  JSON_OBJECT('model_id', 'multilingual-e5-small'));
Query OK, 0 rows affected (1.27 sec)

mysql> DESC vectordb.comment_vec;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| comment_id  | int          | NO   | PRI | NULL    |       |
| comment_vec | vector(2048) | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

実行例：ML_EMBED_TABLEプロシージャー、DISTANCE関数

```
mysql> SELECT sys.ML_EMBED_ROW('ミステリー', JSON_OBJECT('model_id', 'multilingual-e5-small'))
    ->  INTO @query_embedding;
Query OK, 1 row affected (1.31 sec)

mysql> SELECT c.comment_id, c.customer_id, c.book_id, c.comment,
    ->  DISTANCE(c.comment_vec, @query_embedding, 'COSINE') AS distance
    -> FROM vectordb.comment c
    -> ORDER BY DISTANCE(c.comment_vec, @query_embedding, 'COSINE') ASC¥G
*************************** 1. row ***************************
 comment_id: 7
customer_id: 3
    book_id: 5
    comment: 周りで殺人事件起き過ぎw
   distance: 0.11284857988357544
*************************** 2. row ***************************
 comment_id: 4
customer_id: 3
    book_id: 4
    comment: まさかの犯人の視点でのスピンオフ！
   distance: 0.13298475742340088
*************************** 3. row ***************************
 comment_id: 3
customer_id: 1
    book_id: 3
    comment: 読み始めた時は、こんな風にストーリーが展開するなんて思わなかった。
   distance: 0.16685640811920166
```

comment列に入っているレビューと「ミステリー」という単語の意味的な近さを基準にソートしてレビューを検索した結果、「ミステリー」と関連の深いレビューを上位に表示できている

実行例：ML_EMBED_TABLEプロシージャー、DISTANCE関数

```
*************************** 4. row ***************************
 comment_id: 5
customer_id: 4
    book_id: 3
    comment: 最初は転生ものっぽかったけど、よくある転生ものとは全然違った。
   distance: 0.17830097675323486
*************************** 5. row ***************************
 comment_id: 1
customer_id: 1
    book_id: 1
    comment: 設定が作り込まれていて面白い。遅効性SFにハマっています！
   distance: 0.183958888053894O4
*************************** 6. row ***************************
 comment_id: 2
customer_id: 2
    book_id: 2
    comment: 普通の野球漫画とは全然違うけど、ギャンブル＆野球という斬新な設定で面白い。
   distance: 0.19708287715911865
*************************** 7. row ***************************
 comment_id: 6
customer_id: 2
    book_id: 1
    comment: モブキャラがいない。全キャラが魅力的。バトルものなのにインフレしないままずっと面白いのも凄い。
   distance: 0.20563781261444092
7 rows in set (0.00 sec)
```

HeatWave GenAI：SQLだけでLLMを使用可能

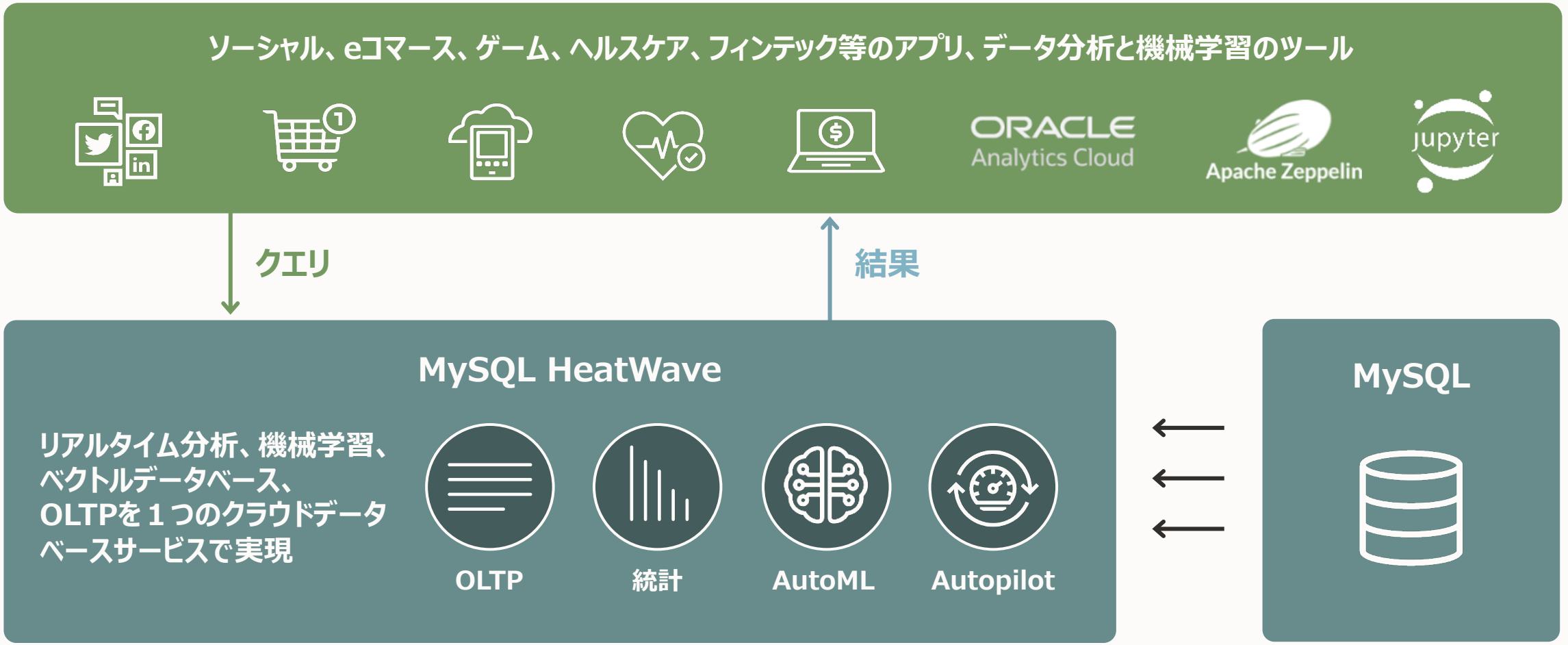
HeatWave GenAI を使って、SQLでLLMを使用する方法の解説記事

- HeatWave GenAI によるインデータベースLLMと自動化されたインデータベース・ベクトル・ストア #15
  https://enterprisezine.jp/article/detail/20127

HeatWave MySQL

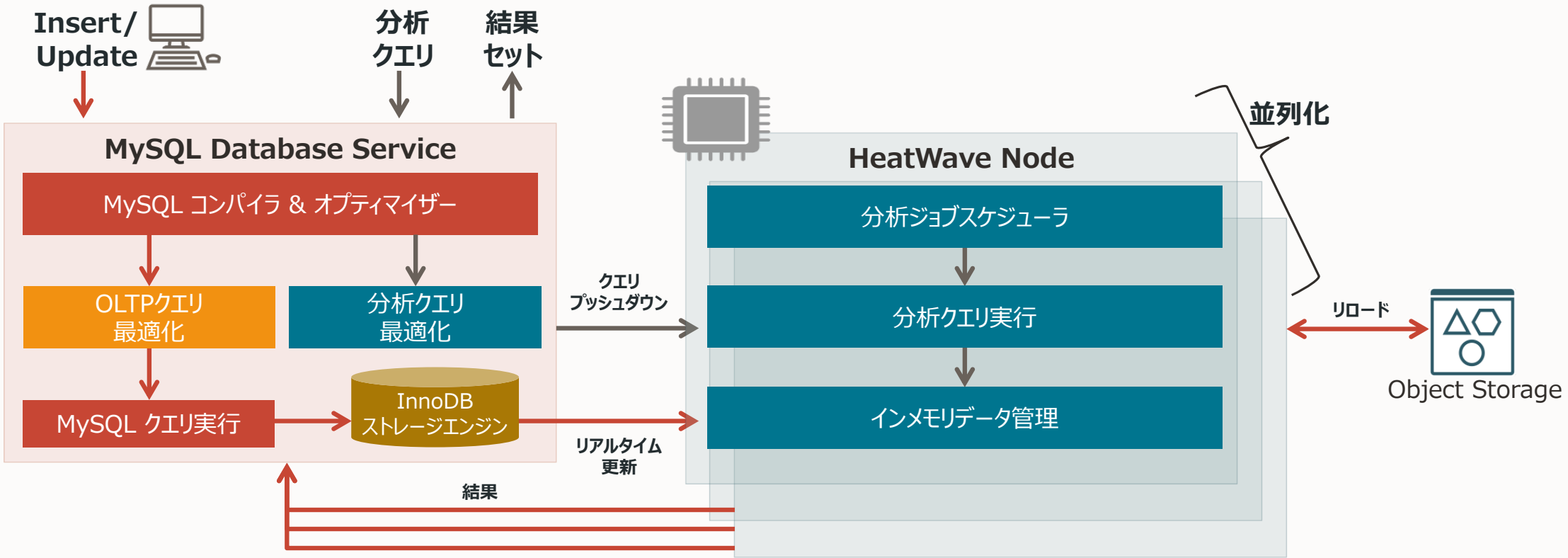
HeatWave MySQL

OLTPだけでなく、DWH、機械学習、ベクトルデータベースにも最適化されたMySQLマネージドサービス



HeatWave MySQL のアーキテクチャ

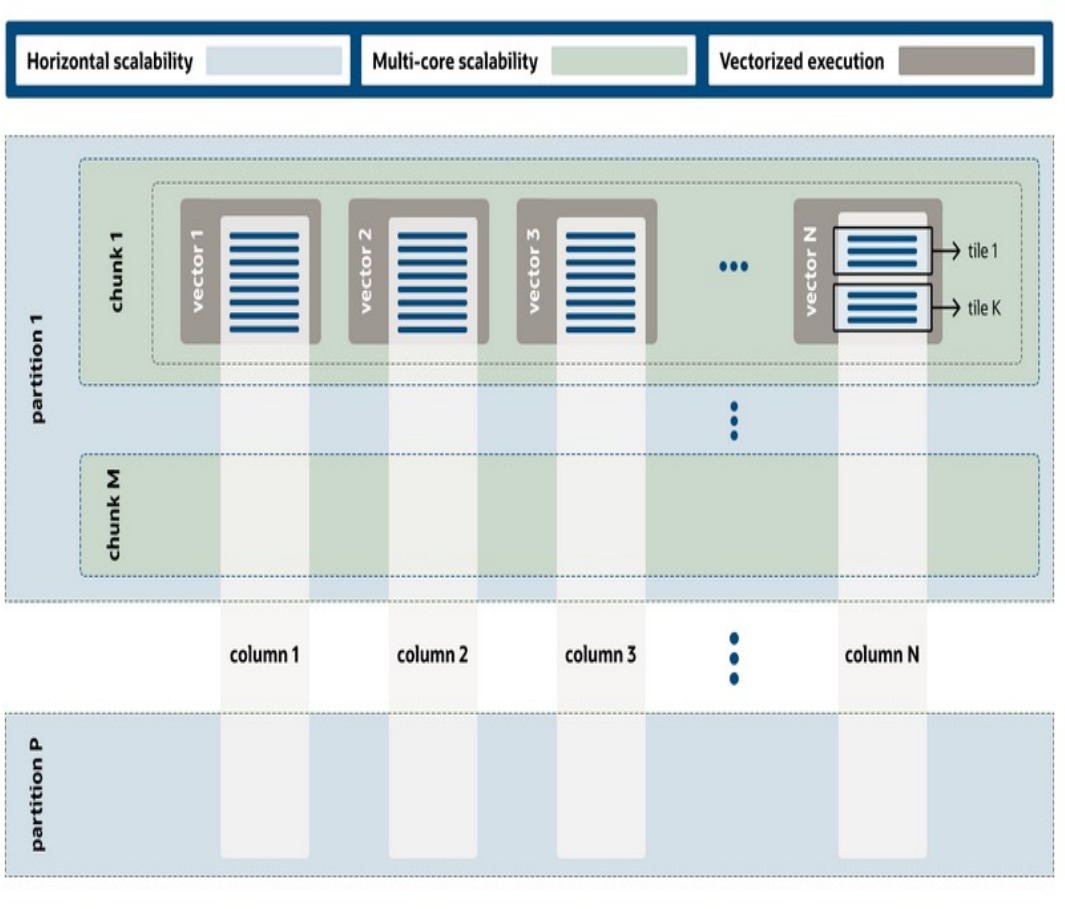
- MySQLに対してSQLを実行するだけで自動的に高速化される
  (HeatWaveの制限事項に該当する場合は、フロントにあるMDSで処理される)
- データの更新はフロントにあるMDSで処理され、更新データは随時HeatWaveノードに反映される



※HeatWaveの制限事項
https://dev.mysql.com/doc/heatwave/en/heatwave-limitations.html

HeatWave が高速に処理できる理由

- インメモリデータベースである
- カラムナーデータベース(列指向データベース)である
- 複数ノードで分散処理できる
  - 最低ノード数は1台、最大ノード数は64台
- Oracle Labsで長年研究していたProject RAPIDの研究成果を活用している
  - **R**APID **A**nalytics **P**rocessing **I**n **D**RAM (メモリ上での高速分析処理)
  - RAPIDでは、ハードウェアリソースを最大限活用して、超並列処理できるアーキテクチャーになっている



HeatWave MySQL：
MySQLのマネージドサービスとしての導入事例

※この後のスライドでは、「MySQL Database Service」という名称を使用しています。
(サービス名称変更前の名称)

りらく様によるMySQL Database ServiceによるDB統合事例

- Amazon Auroraで構築していた3つのDBを1つのMySQL Database Serviceに統合
- セラピスト系、店舗系/予約系、顧客系の3つのDBを1つに統合
- DB統合とシステム基盤をAWSからOCIへ移行したことにより、同様の処理性能を実現する構成でも大幅なコスト削減とパフォーマンス向上を実現

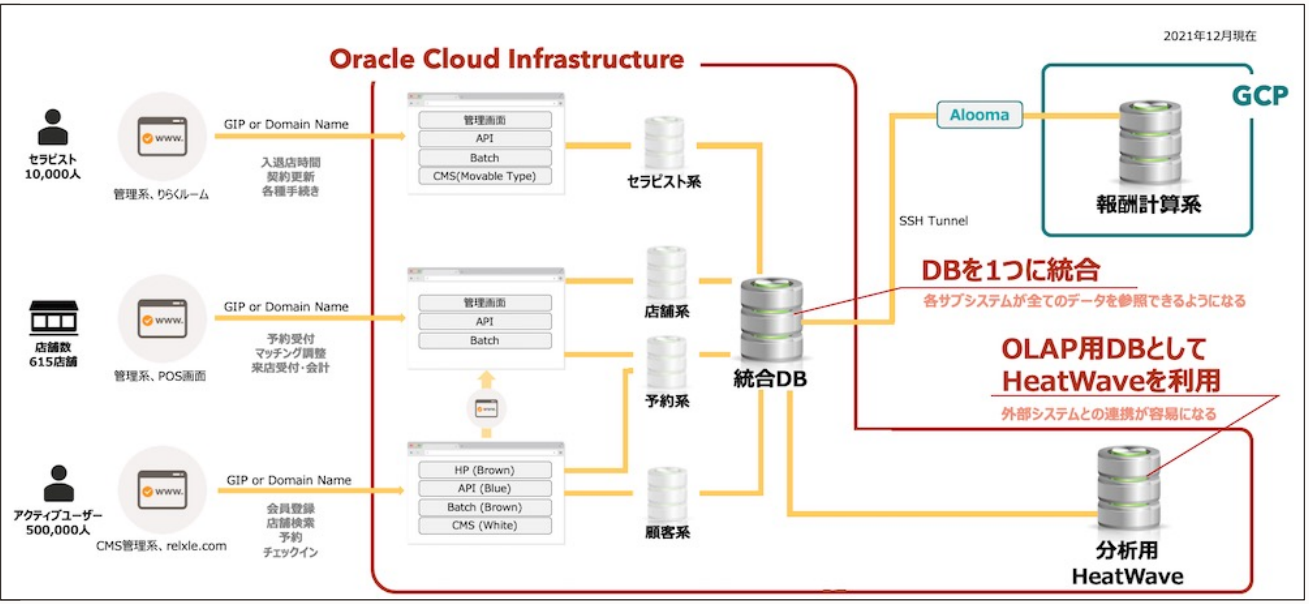
りらく
riraku

株式会社りらく、Oracle Cloud InfrastructureとMySQL のマネージド・データベース・サービスを採用して、統合DBを構築
コスト削減とパフォーマンス向上を実現

概要

リラクゼーション施設「りらくる」を全国に展開する株式会社りらくは、2009年の創業から約10年間で620店舗、14,500人以上のセラピス トを有する(2021年3月現在)、日本のリラクゼーション業界1位に急成長した企業です。りらくはOracle Cloud Infrastructure（OCI）とMySQL のマネージド・データベース・サービスを採用してデータを一元管理する統合DBを構築しています。

また、システム基盤をAWSからOCIへ移行したことで、同様の処理性能を実現する構成でも大幅なコスト削減とパフォーマンス向上を実現しています。



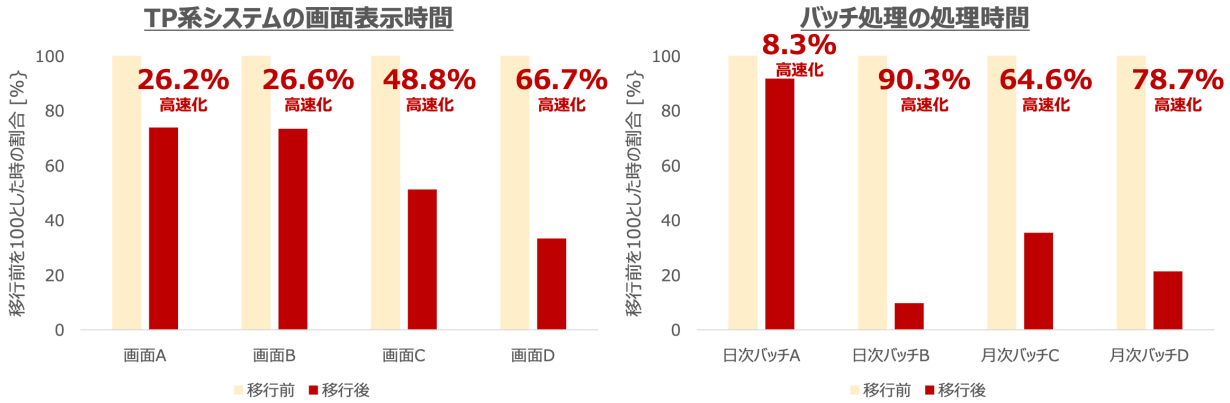
出典：https://www.mysql.com/jp/why-mysql/case-studies/riraku-migrates-from-amazon-aurora-to-oci-to-reduce-costs-and-improve-performance.html
事例発表講演資料&動画： https://www.oracle.com/jp/cloud/events/cloud-days/on-demand/#tab3

りらく様によるMySQL Database ServiceによるDB統合事例

- OCI、MDSへの移行によりコスト削減とパフォーマンス改善を実現
- MDSを使ってDB統合することで、後からHeatWaveを追加すれば、統合DB上に存在するデータを全て分析対象にできるため分析対象データの抜け漏れがなくなるだけでなく、ETLツールを使ったデータ連携も不要になる

3. OCI採用の理由

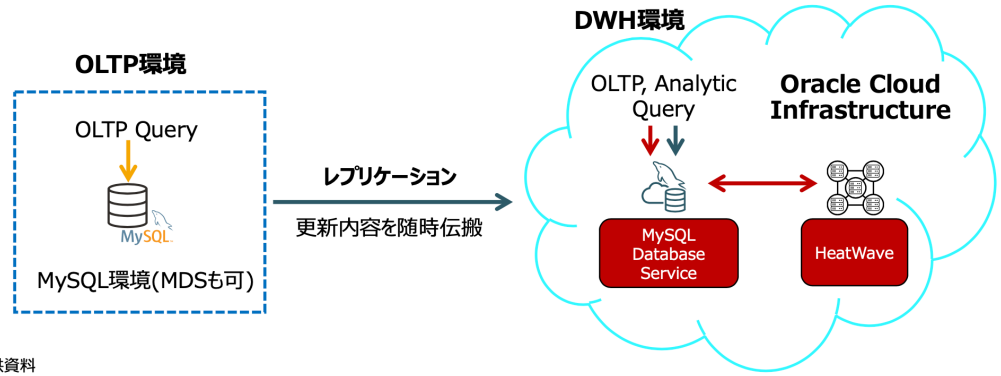
OCI、MDSへの移行により、日中の処理もバッチ処理もパフォーマンス改善！！



3. OCI採用の理由

- **HeatWaveにより、MySQL上のデータをそのまま分析対象にできるため、FlyDataを使って複数のサブシステムからデータ連携する必要が無い**
- **ETLツール不要でシンプルなシステム構成のまま、最新データも分析が可能**

「Redshiftにデータはたくさん入っているが、分析に本当に必要なデータが入っていない」という問題を回避



※オラクル社提供資料

出典：https://www.mysql.com/jp/why-mysql/case-studies/riraku-migrates-from-amazon-aurora-to-oci-to-reduce-costs-and-improve-performance.html
事例発表講演資料&動画： https://www.oracle.com/jp/cloud/events/cloud-days/on-demand/#tab3

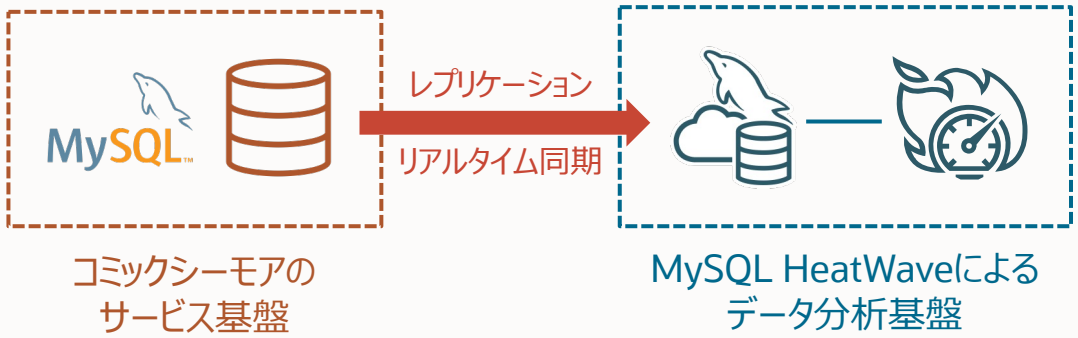
HeatWave MySQL：
MySQLをインターフェースとしたDWHとしての導入事例

※この後のスライドでは、「MySQL HeatWave Database Service」という名称を使用しています。
(サービス名称変更前の名称)

顧客事例：NTTソルマーレ 様

MySQL HeatWaveで国内最大級の電子書籍配信サービス「コミックシーモア」でのデータ利活用を強化

システム構成イメージ



利用サービス・製品

- MySQL HeatWave Database Service

お客様のコメント

「『MySQL HeatWave』の導入によりサービス基盤とデータ分析基盤のリアルタイムなデータ同期が実現できました。さらにこれまで通常のMySQLで1.5時間程度かかっていたバッチ処理が2秒程度で完了するなど性能の良さも実感しています。処理を待つ思考停止の時間が短縮化され、業務効率化にもつながっています。

MySQLに対応したツールは『MySQL HeatWave』でもそのまま活用でき、ユーザーの利便性を維持しながら様々な分析データを更なるサービス向上に役立てることができています。

『MySQL HeatWave』を利用した新たなデータ分析基盤を活用し、今後も更に幅広いお客様に楽しんでいただける書籍配信サービスを提供していきます。」

エヌ・ティ・ティ・ソルマーレ株式会社
電子書籍事業部 サービス開発グループ　木下 氏

HeatWave を無料で利用する方法

Transactions + Analytics + Machine Leaning + Lakehouse

HeatWave MySQL

MySQLだけでトランザクション処理、データ分析、機械学習、データレイクを実現

オラクルのクラウドにはAlways Freeというサービスがあり、
いくつかのリソースが常時無料で使用できます

＋

HeatWave MySQLがAlways Freeの対象に
含まれました！！

オラクルクラウドのAlways Free Servicesで無料でHeatWaveを利用可能

https://www.oracle.com/jp/heatwave/free/

※オラクルクラウド：Oracle Cloud Infrastructure(OCI)

Always Freeサービス

次のものは無期限で使用できます。

- ホームリージョン内の単一ノードHeatWaveクラスタを備えたスタンドアロンHeatWaveデータベースシステム
- 50 GBのストレージ
- 50 GBのバックアップ・ストレージ
- 2つのAMD Compute VM
- 最大4つのArm Ampere A1コンピュート・インスタンス

HeatWave MySQL、HeatWave Lakehouse、HeatWave AutoML、HeatWave Vector Store、HeatWave Autopilotにアクセスして、小規模なアプリケーションを構築および実行できます。

- Oracle Autonomous Transaction Processing, Autonomous Data Warehouseと同じく、HeatWaveも期間の制限なく無料で使用可能
- インスタンス数や容量、一部機能の制限あり

- 容量制限などがない30日間無料トライアルとして300ドルの無料クレジットをあわせて提供
- HeatWave GenAIの試用はトライアルにて

HeatWave MySQL　HeatWave Lakehouse　HeatWave AutoML

Always FreeでHeatWave MySQLを使用する方法

- 以下URLからオラクルクラウドのトライアルアカウントを作成する
  https://signup.cloud.oracle.com/
  - ホームリージョンは後から変更できないので注意
  - 日本には現在東京リージョンと大阪リージョンがあります
- 以下URLのチュートリアルを参考にし、HeatWave MySQL環境を構築する
  - OCIチュートリアル 入門編：その9 - クラウドでMySQL Databaseを使う
    https://oracle-japan.github.io/ocitutorials/beginners/creating-mds/

Always FreeでHeatWave MySQLを使用する場合の制限事項

- 最新バージョンのみ使用可能 (本日時点では、9.1.2)
- MySQL.Freeシェイプ、HeatWave.Freeのみ使用可能 (Always Free専用のスペックがあまり高くないシェイプ)
- ストレージサイズは50GB
- HeatWaveノードは1台のみ追加可能
- HeatWave AutoML と HeatWave Lakehouse は使用可能
- HeatWave GenAI は使用不可　　※DISTANCE関数、ML_EMBED_ROW関数などは使用可能
- レプリケーション機能は使用可能
- 高可用性、リードレプリカは使用不可
- 自動バックアップは1日だけ取得される
- 手動バックアップやポイントインタイムリカバリは使用不可
- Database Management and Ops Insights サービスは使用不可 (データベースの監視ツール)

※原文：https://docs.oracle.com/en-us/iaas/mysql-database/doc/features-mysql-heatwave-service.html#MYAAS-GUID-772BD870-57C1-4B21-9205-FFC5B4290044

The world's most popular open source database
世界で最も普及しているオープンソース データベース


ORACLE