ORACLE

LLMに興味があるデータベースエンジニアの方にお勧め！ SQLだけでLLMを使用できるHeatWave GenAIのご紹介

Open Source Conference 2025 Nagoya

山﨑 由章 / Yoshiaki YAMASAKI
MySQL HeatWave Data Architect
日本オラクル株式会社

アジェンダ

1. HeatWave 概要
2. HeatWave GenAI 概要
3. HeatWave GenAI で使用できるLLMの種類や日本語対応状況について
4. SQLだけでLLMを使用する方法
5. SQLだけでRAGを実現する方法
6. ベクトルデータベース(ベクトルストア) としての使用方法

HeatWave 概要

HeatWave

インメモリー高速分散処理エンジン

HeatWave MySQL

Transactional

クラウド版MySQL
Enterprise Edition

HeatWave MySQL

Analytics

大規模データ分析を
大幅に高速化

HeatWave Lakehouse

Lakehouse

オブジェクト・
ストレージとの統合

HeatWave AutoML

AutoML

機械学習の
パイプラインを自動化

HeatWave GenAI

GenAI

専門知識なしで
生成AIの活用

HeatWave MySQL の特徴（MySQLのマネージドサービスとしての特徴）

MySQL開発ベンダーであるオラクル社からのサポートも受けられる
高性能かつコストパフォーマンスも高いフルマネージドデータベース！！

高性能	MDSでは高パフォーマンスなブロック・ボリュームを標準採用 (ブロック・ボリュームのIOPS：75 IOPS/GB)
低価格	同等スペック(CPU、Memory)で比較すると、 他社製のMySQLマネージドサービスの1/2～1/3程度
MySQL開発 ベンダーが提供	オラクルのMySQLチームが100%開発、運用、サポート MySQL部分についてもコアなサポートを受けられる

顧客事例：ファンコミュニケーションズ様
老舗のブログサービス「Seesaaブログ」をAWSからOCIに移行し、約50%の劇的なコスト削減を実現

Seesaaブログ

- 2003年にサービス開始。日記やアフィリエイト、まとめブログなど様々なブログライフをサポート。200万人以上のユーザーが利用

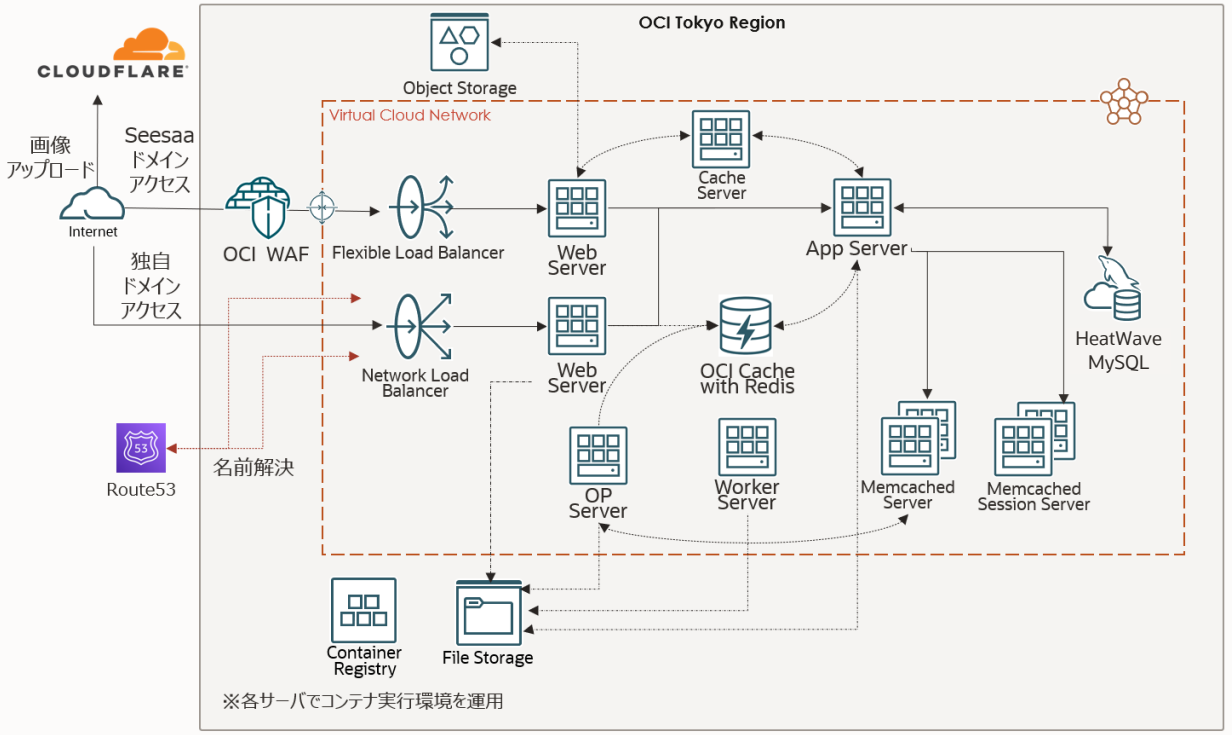
従来の課題

- 既存のAWS環境のコスト削減の必要性があった
- VMのサイズが柔軟に選択できず、CPUコア数とメモリサイズがアンバランスで過剰なリソースコストを支払っていた

採用ポイントと導入効果

- 既存AWS環境と比較して、**約50%のコスト削減**ができた
- OCIのフレキシブル・シェイプのVMは**CPUコア数とメモリサイズに対してシンデレラフィットの設定が可能**で、サーバ利用全体にて、必要な性能に関して適切なコストで利用できるようになった
- コスト削減フレームワークの利用により、**AWSユーザー観点での移行アセスメント**など、Oracleメンバーの適切な支援を気軽に得られた。また、Oracleから提供される**MySQLのQ&Aサポートは安心感があり有益**だった
- OCI Cache with Redisでは、既存利用のAmazon ElastiCache for Redisから**約25%のコスト削減効果**があった

システム構成イメージ



利用サービス(クラウドサービス/その他)

- Compute VM, Block Volume, **HeatWave MySQL**, OCI Cache with Redis, OCI Registry (Container Registry), etc.
- コスト削減フレームワーク

顧客事例：アプルーシッド様

「ドクセル」を、OCI Container Instancesを活用してGoogle Cloudから移行し、圧倒的なコスト削減を実現

ドクセル（Docswell）

- PDFやパワーポイントのスライドを共有できる国産のスライドシェアサービス。ユーザー約30万人、月間アクセス約600万回。

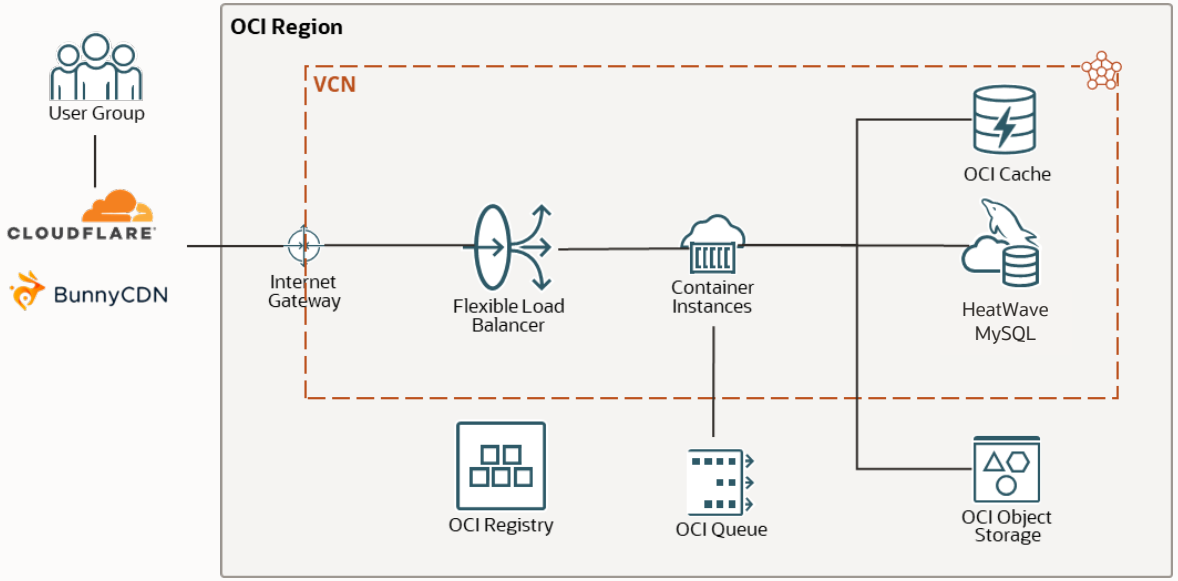
従来の課題

既存のGoogle Cloud環境のコスト削減の必要性があった(特に、アウトバウンドのデータ転送のコスト)。
ユーザーアクセス時にコンテナが都度起動するスピンアップの仕組みにより、時に遅延が発生していた。

採用ポイントと導入効果

既存Google Cloud環境から移行し、特にデータ転送が毎月10TB無料の利点を享受し、全体で**約65%のコスト削減**を実現。HeatWave MySQL の高可用性構成の部分は、
要件を実現しつつ**約50%のコスト削減**を実現。
OCI Container Instancesを活用し、**Kubernetesを利用せずに既存のコンテナアプケーションをシームレスに移行**できた。
コスト面で有利なContainer Instancesを必要個数、常時起動しておくことで、スピンアップの仕組みにする必要が無くなった。結果、**遅延のない安定的な処理**を実現。
複数のコンテナ間の**疎結合な処理にOCI Queueを活用**。

システム構成イメージ



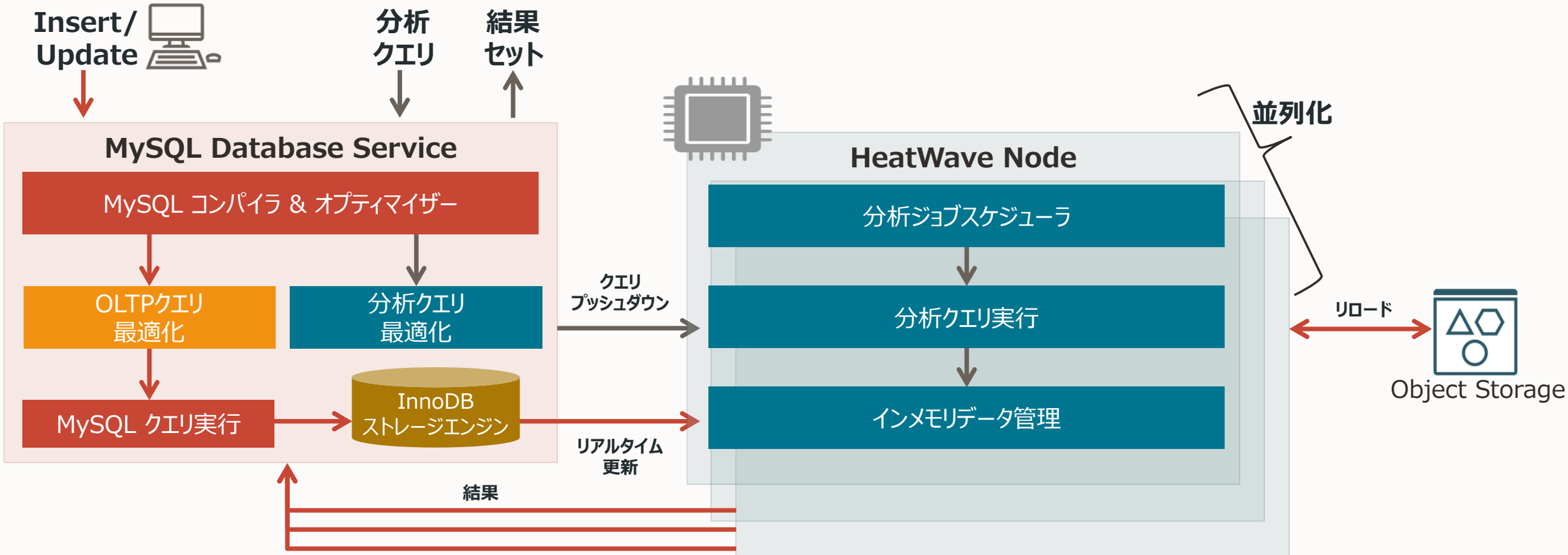
利用サービス（クラウドサービス／その他）

- OCI Container Instances
- OCI Registry
- HeatWave MySQL
- OCI Queue
- OCI Cache
- コスト削減フレームワーク (Oracleによる移行アセスメント支援)

HeatWave MySQLのアーキテクチャ

大量データの集計処理などをHeatWave Nodeで高速に処理

- アプリケーションからは従来通りMySQLに対してSQLを実行するだけで高速化される

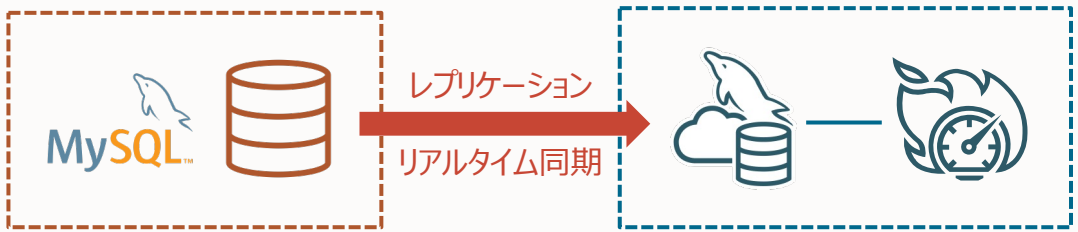


顧客事例：NTTソルマーレ 様

HeatWave MySQL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

コミックシーモア

システム構成イメージ



コミックシーモアの
サービス基盤

HeatWaveによる
データ分析基盤

利用サービス・製品

- HeatWave MySQL

お客様のコメント

「『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『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。

『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 GenAI 概要

HeatWave GenAI を使うとデータベースだけで簡単にLLMを活用可能

自然言語で会話

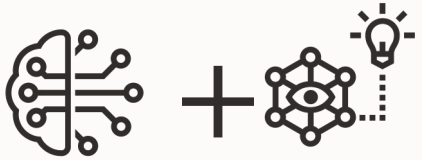
- 自然言語を使って非構造化ドキュメントの情報から生成される会話
- 後続の質問のため HeatWaveがコンテキストを保持

コンテンツの作成と要約

- 企業内ドキュメントから示唆を生成 / レポートを作成
- PDF のインストラクションマニュアルからブログを生成
- コンテンツを要約

RAGと類似性検索

- RAGの企業内データを使ってより正確で関連性が高い文脈の生成 AI を利用
- 非構造化データから類似性検索を実行

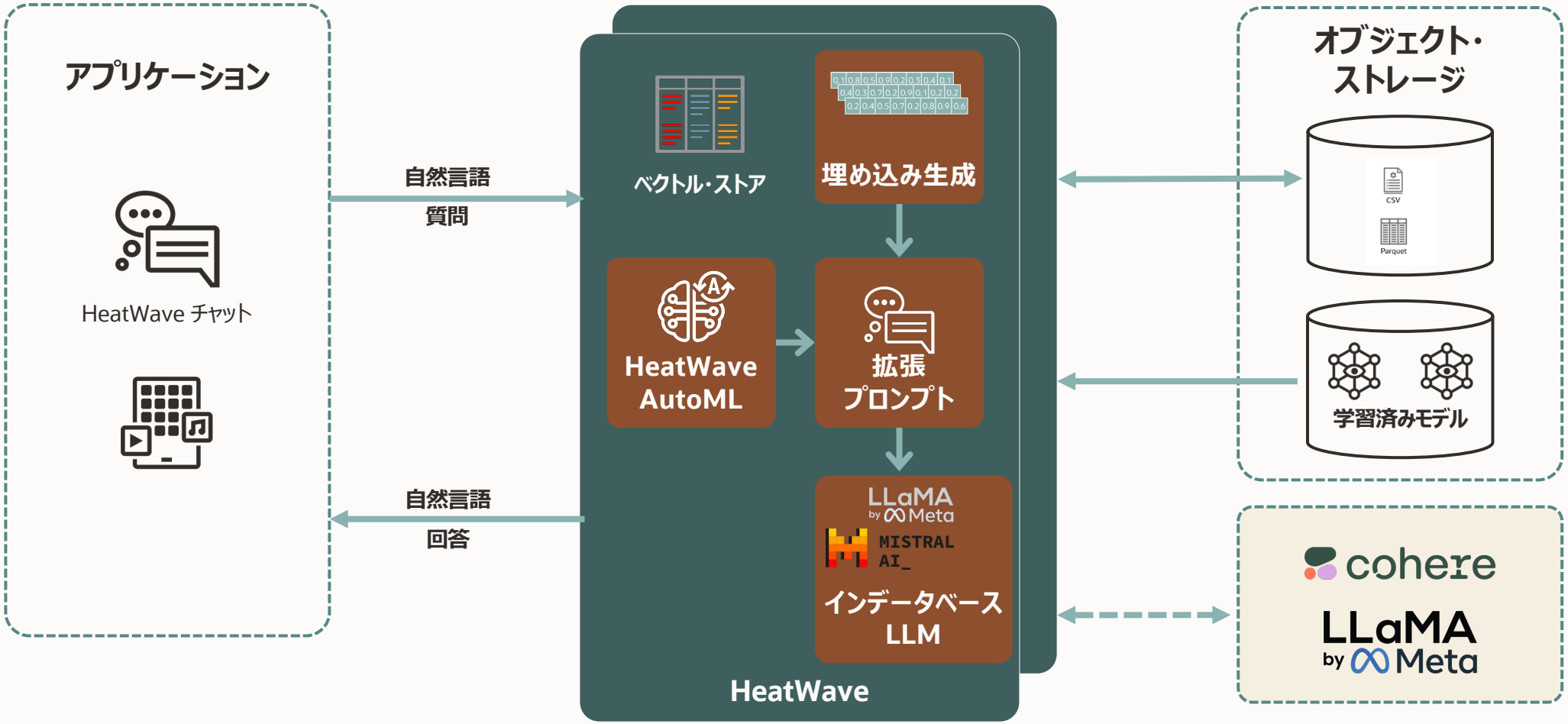


生成AIと機械学習の相乗効果

- 機械学習と生成 AI の結合によって時間を節約し、より多くの価値をお客様へ
- 機械学習 でフィルタされたデータを伴う生成 AI を利用することによって低コストに精度を向上

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

OCI Generative AI serviceとの統合も



HeatWave GenAI で使用できるルーチン(プロシージャー、関数)

ルーチン名	説明
ML_GENERATE	LLMに対して問い合わせを実行
ML_GENERATE_TABLE	LLMに対してバッチ処理で問い合わせを実行（事前にテーブルに問い合わせを格納しておく）
VECTOR_STORE_LOAD	オブジェクトストレージ上のファイルをエンベディングしてHeatWave GenAIのベクトルストアにロードする
ML_RAG	HeatWave GenAIのベクトルストアにロードされたデータを使って、 LLMに対してRAGを使った問い合わせを実行
ML_RAG_TABLE	HeatWave GenAIのベクトルストアにロードされたデータを使って、 LLMに対してRAGを使った問い合わせをバッチ処理で実行（事前にテーブルに問い合わせを格納しておく）
HEATWAVE_CHAT	コンテキストを保持し、チャット形式でLLMに対して問い合わせを実行
ML_EMBED_ROW	入力したテキストデータに対するエンベディングを生成
ML_EMBED_TABLE	テーブル内のテキストデータを入力として、バッチ処理でエンベディングを生成

※ドキュメント：HeatWave User Guide / HeatWave GenAI Routines
https://dev.mysql.com/doc/heatwave/en/mys-hwgenai-routines.html

HeatWave GenAI で使用できるLLMの種類や日本語対応状況について

HeatWave GenAI で使用できるLLMの種類 (HeatWaveだけで使えるLLM)

テキスト生成用

- mistral-7b-instruct-v1
- (llama2-7b-v1)　　* 既に非推奨になっているLLM
- llama3-8b-instruct-v1

エンベディング生成用

- multilingual-e5-small　　* 日本語に対応したLLM
- all_minilm_l12_v2

※2025年5月時点の情報です。今後アップデートされる可能性も高いため、最新の情報はドキュメントからご確認下さい。

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

HeatWave GenAI で使用できるLLMの種類 (OCI GenAI Serviceと連携して使えるLLM)

テキスト生成用

- meta.llama-3.1-405b-instruct
- meta.llama-3.2-90b-vision-instruct
- meta.llama-3.3-70b-instruct
- cohere.command-r-08-2024　　* 日本語に対応したLLM
- cohere.command-r-plus-08-2024　　* 日本語に対応したLLM

エンベディング生成用

- cohere.embed-english-v3.0
- cohere.embed-multilingual-v3.0　　* 日本語に対応したLLM

※2025年5月時点の情報です。今後アップデートされる可能性も高いため、最新の情報はドキュメントからご確認下さい。

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

HeatWave GenAI でSQLだけでLLMを使用する方法

環境準備

- HeatWaveノードとして HeatWave.512GB シェイプのノードを1ノード以上追加したHeatWave MySQL環境を用意
- HeatWave MySQLのバージョンは9.xの最新バージョンを選択
- OCIのGenerative AI Serviceと連携する場合はHeatWave MySQLと連携するための手続きが必要
  - OCI Generative AI Serviceとの連携方法
    https://enterprisezine.jp/article/detail/21524?p=3&anchor=1
  - OCI Generative AI Service のLLMが提供されているリージョンを選択（日本では大阪リージョン）

SQLからLLMを使用する方法

- HeatWave MySQLに接続して、ML_GENERATE、ML_GENERATE_TABLE ルーチンを使うことで SQLからLLMを使用可能
- model_id オプションで使用するLLMを指定可能
- context、temperature、top_k、frequency_penalty などのオプションも指定可能
  - ML_GENERATE
    https://dev.mysql.com/doc/heatwave/en/mys-hwgenai-ml-generate.html
  - ML_GENERATE_TABLE
    https://dev.mysql.com/doc/heatwave/en/mys-hwgenai-ml-generate-table.html

実行例：基本的な使い方

```
mysql> SET @query="MySQLについて説明して下さい";
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT sys.ML_GENERATE(@query, JSON_OBJECT("task", "generation", "model_id", "llama3-8b-instruct-v1", "language", "ja"))¥G
*************************** 1. row ***************************
sys.ML_GENERATE(@query, JSON_OBJECT("task", "generation", "model_id", "llama3-8b-instruct-v1", "language", "ja")): {"text": "MySQLは、世界で最も人気の高いオープンソースのRDBMS（Relational Database Management System）です。1995年にMichael Wideniusによって開発されました。¥n¥n MySQLは、SQL（Structured Query Language）を使用してデータベースを管理することができます。主にWebアプリケーションや企業の情報システムで使用されています。¥n¥nMySQLの特徴として、以下のような点が挙げられます。¥n¥n* オープンソース：MySQLはオープンソースソフトウェアであり、無料でダウンロード・インストールすることができます。¥n* スケーラビリティ： MySQLは大規模なデータベースを管理することができ、大量のトラフィックに対応することができます。¥n* セキュリティー：MySQLにはセキュリティー機能が充実しており、パスワードや暗号化されたデータを安全に保存することができます。¥n* プログラミング言語： MySQLは多くのプログラミング言語（例えばPHP、Python、Javaなど）との互換性があります。¥n¥n MySQLの使用例として、以下のようなものが挙げられます。¥n¥n"}
1 row in set (14.99 sec)
```

実行例：出力サイズを調整

```
mysql> SET @query="MySQLについて説明して下さい";
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT sys.ML_GENERATE(@query, JSON_OBJECT("task", "generation", "model_id", "llama3-8b-instruct-v1", "language", "ja", "max_tokens", 500))¥G
*************************** 1. row ***************************
sys.ML_GENERATE(@query, JSON_OBJECT("task", "generation", "model_id", "llama3-8b-instruct-v1", "language", "ja", "max_tokens", 500)): {"text": "MySQLは、世界で最も人気の高いオープンソースのRDBMS（Relational Database Management System）です。1995年にMichael Wideniusによって開発されました。¥n¥n MySQLは、SQL（Structured Query Language）を使用してデータベースを管理することができます。主にWebアプリケーションや企業の情報システムで使用されています。¥n¥nMySQLの特徴として、以下のような点が挙げられます。¥n¥n* オープンソース：MySQLはオープンソースソフトウェアであり、無料でダウンロード・インストールすることができます。¥n* スケーラビリティ： MySQLは大規模なデータベースを管理することができ、大量のトラフィックに対応することができます。¥n* セキュリティー：MySQLにはセキュリティー機能が充実しており、パスワードや暗号化されたデータを安全に保存することができます。¥n* プログラミング言語：MySQLは多くのプログラミング言語（例えばPHP、Python、Javaなど）との互換性があります。¥n¥n MySQLの使用例として、以下のようなものが挙げられます。¥n¥n* Webアプリケーション：MySQLを使用してWebアプリケーションにデータベースを実装することができます。¥n* 企業情報システム：MySQLを使用して企業の情報システム（例えば会計システム、人事システムなど）を実装することができます。¥n* ゲーム開発：MySQLを使用してゲームのデータベースを実装することができます。¥n¥n以上のように、MySQLはRDBMSとして非常に有用なツールであり、多くの企業や個人で使用されています。"}
1 row in set (21.72 sec)
```

実行例：バッチ処理で実行する場合

```
mysql> SELECT * FROM input_table;
+----+-----------------------------------------------------+
| id | Input                                               |
+----+-----------------------------------------------------+
|  1 | MySQLについて説明して下さい                                 |
|  2 | データベースについて説明して下さい                                 |
+----+-----------------------------------------------------+
2 rows in set (0.00 sec)

mysql> CALL sys.ML_GENERATE_TABLE("GenAi.input_table.Input", "GenAi.input_table.Output",
JSON_OBJECT("task", "generation", "model_id", "llama3-8b-instruct-v1", "language", "ja",
    -> "max_tokens", 1000));
Query OK, 0 rows affected (45.50 sec)
```

実行例：バッチ処理で実行する場合

```
mysql> SELECT * FROM GenAi.input_table¥G
*************************** 1. row ***************************
    id: 1
 Input: MySQLについて説明して下さい
Output: {"text": "MySQLは、世界で最も人気の高いオープンソースのRDBMS（Relational Database Management System）です。
<中略>
* ゲーム： MySQLを使用してゲームのデータベースを実装することができます。¥n¥n以上のように、MySQLはRDBMSとして広く使われており、多くのアプリケーションで使用されています。"}
*************************** 2. row ***************************
    id: 2
 Input: データベースについて説明して下さい
Output: {"text": "データベース（Database）は、コンピュータープログラムが使用するための、情報を格納した構造化されたコレクションです。
<中略>
1. **リレーショナルデータベース**：テーブル間で関係がある情報を保持するデータベース。¥n2. **NoSQLデータベース**：スケーラビリティや高速性に優れたデータベース。¥n3. **グラフデータベース**：ネットワークやグラフ構造の情報を保持するデータベース。¥n¥n以上、データベースについて簡単に説明しました。"}
2 rows in set (0.00 sec)
```

HeatWave GenAI でSQLだけでRAGを実現する方法

事前準備

- HeatWave GenAIのベクトルストアにベクトルデータを格納するために事前にmysql_task_management_ensure_schema() を実行する
  - 実行するとmysql_task_management データベースが作成され、その中に関連テーブルが作成される

```
mysql> SELECT mysql_task_management_ensure_schema();
```

SQLからRAGを使ってLLMに対して問い合わせをする方法

- HeatWave MySQLに接続して、ML_RAG、ML_RAG_TABLE ルーチンを使うことで
  SQLからRAGを使ってLLMに対して問い合わせを実行可能
- RAGで参照するデータは事前にVECTOR_STORE_LOADルーチンを使ってHeatWave GenAIのベクトルストアに格納しておく
- model_id オプションで使用するLLMを指定可能
- n_citations(コンテキスト検索時に考慮するセグメントの数)、exclude_document_name、retrieval_options などのオプションも指定可能
  - ML_RAG
    https://dev.mysql.com/doc/heatwave/en/mys-hwgenai-ml-rag.html
  - ML_RAG_TABLE
    https://dev.mysql.com/doc/heatwave/en/mys-hwgenai-ml-rag-table.html

実行例：ベクトルストアへロードするテキストドキュメント

■HeatWave_MySQL.txtの内容

HeatWave MySQLは、オラクルが提供するMySQLのマネージドサービスです。通常のMySQLのマネージドサービスとして使用することもできますが、HeatWaveという拡張機能を利用することで検索クエリーを通常のMySQLよりも高速に実行できます。従来MySQLは大量データの分析/集計処理などには不向きとされてきましたが、HeatWave MySQLでは大量データの集計処理も超高速に実行できます。

また、HeatWave MySQLにはMySQL Databaseだけで（SQLだけで）機械学習を実現できるHeatWave AutoML、オブジェクトストレージ上のデータを高速にHeatWaveでの分析/集計対象にできるHeatWave Lakehouse、SQLだけでLLMを活用したりMySQLをベクトルストアとして使用できるHeatWave GenAIなどの拡張機能もあります。

実行例：ベクトルストアへのドキュメントのロード

```
mysql> SET @options = JSON_OBJECT("schema_name", "RAG", "table_name", "test_embeddings", "language",
"ja");
Query OK, 0 rows affected (0.00 sec)

mysql> CALL sys.VECTOR_STORE_LOAD("https://objectstorage.ap-osaka-1.oraclecloud.com/省略
/HeatWave_MySQL.txt", @options);
+---------+-------------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
| task_id | task_status_query
|
+---------+-------------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
|       1 | SELECT id, name, message, progress, status, scheduled_time,estimated_completion_time,
estimated_remaining_time, progress_bar FROM mysql_task_management.task_status WHERE id=1¥G |
+---------+-------------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
```

ロードの進捗状況を確認するためのSQLが表示される

実行例：ベクトルストアへのデータロード状況の確認

```
mysql> SELECT id, name, message, progress, status, scheduled_time,estimated_completion_time,
estimated_remaining_time, progress_bar FROM mysql_task_management.task_status WHERE id=1¥G
*************************** 1. row ***************************
                      id: 1
                    name: Vector Store Loader
                 message: Loading in progress...
                progress: 10
                  status: RUNNING
          scheduled_time: 2025-03-25 23:57:24
estimated_completion_time: 2025-03-25 23:59:54
 estimated_remaining_time: 135.00000
             progress_bar: #_________
1 row in set (0.00 sec)
```

実行例：ベクトルストアへのデータロード状況の確認

```
mysql> SELECT id, name, message, progress, status, scheduled_time,estimated_completion_time,
estimated_remaining_time, progress_bar FROM mysql_task_management.task_status WHERE id=1¥G
*************************** 1. row ***************************
                      id: 1
                    name: Vector Store Loader
                 message: Loading in progress...
                progress: 40
                  status: RUNNING
          scheduled_time: 2025-03-25 23:57:24
estimated_completion_time: 2025-03-25 23:58:27
 estimated_remaining_time: 37.50000
             progress_bar: ####______
1 row in set (0.00 sec)
```

実行例：ベクトルストアへのデータロード状況の確認

```
mysql> SELECT id, name, message, progress, status, scheduled_time,estimated_completion_time,
estimated_remaining_time, progress_bar FROM mysql_task_management.task_status WHERE id=1¥G
*************************** 1. row ***************************
                      id: 1
                    name: Vector Store Loader
                 message: Task completed.
                progress: 100
                  status: COMPLETED
          scheduled_time: 2025-03-25 23:57:24
estimated_completion_time: 2025-03-25 23:57:53
 estimated_remaining_time: 0.00000
             progress_bar: ##########
1 row in set (0.00 sec)
```

実行例：RAGを使わないLLMへの問い合わせ

```
mysql> SET @query="HeatWaveとはなにか教えてください";
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT sys.ML_GENERATE(@query, JSON_OBJECT("task", "generation", "model_id", "cohere.command-r-08-2024", "language", "ja", "max_tokens", 1000))¥G
*************************** 1. row ***************************
sys.ML_GENERATE(@query, JSON_OBJECT("task", "generation", "model_id", "cohere.command-r-08-2024", "language", "ja", "max_tokens", 1000)): {"text": "HeatWave とは、コヒアが開発した大規模言語モデルです。このモデルは、自然言語処理タスクに特化し、さまざまな言語のテキストを理解し、生成することができます。HeatWave は、コヒアの他の言語モデルと同様に、膨大な量のテキストデータを学習し、その知識と文脈理解に基づいて、人間のような応答やテキストの生成を行うことができます。¥n¥nHeatWave は、自然言語処理のさまざまな分野で活用され、テキストの要約、質問応答、文章生成、翻訳など、幅広いタスクをこなすことができます。このモデルは、その柔軟性と適応性により、さまざまな業界や分野で有用なツールとして利用されています。¥n¥nHeatWave の特徴の一つは、その多言語対応能力です。このモデルは、複数の言語を学習し、異なる言語間の翻訳や、多言語環境でのテキスト処理を可能にします。これにより、グローバルなコミュニケーションや、多言語コンテンツの処理に役立ちます。¥n¥nまた、HeatWave は、コヒアの他のモデルと同様に、継続的な学習と改善が可能です。新しいデータやフィードバックを取り入れることで、その性能と正確性を向上させ、より人間らしい応答や、複雑なタスクへの対応力を高めることができます。¥n¥nHeatWave は、自然言語処理の分野で重要な役割を果たし、人工知能と人間のコミュニケーションをより自然で効率的なものにするために貢献しています。"}
1 row in set (5.39 sec)
```

実行例：RAGを使ったLLMへの問い合わせ

```
mysql> SET @query="HeatWaveとはなにか教えてください";
Query OK, 0 rows affected (0.00 sec)

mysql> SET @options = JSON_OBJECT("vector_store", JSON_ARRAY("RAG.test_embeddings"), "model_options",
JSON_OBJECT("model_id", "cohere.command-r-08-2024", "language", "ja", "max_tokens", 1000));
Query OK, 0 rows affected (0.00 sec)

mysql> CALL sys.ML_RAG(@query, @output, @options);
Query OK, 1 row affected (7.85 sec)
```

実行例：RAGを使ったLLMへの問い合わせ

```
mysql> SELECT JSON_PRETTY(@output)¥G
*************************** 1. row ***************************
JSON_PRETTY(@output): {
  "text": "HeatWave は、オラクルが提供する MySQL の拡張機能で、検索クエリを高速に実行できるようにする機能です。これにより、大量データの分析や集計処理を従来よりも高速に行うことができます。また、HeatWave には、機械学習や LLM 活用、ベクトルストアとしての MySQL 利用など、さまざまな拡張機能が含まれています。",
  "citations": [
    {
      "segment": "HeatWave MySQLは、オラクルが提供するMySQLのマネージドサービスです。通常のMySQLのマネージドサービスとして使用することもできますが、HeatWaveという拡張機能を利用することで検索クエリーを通常のMySQLよりも高速に実行できます。従来MySQLは大量データの分析/集計処理などには不向きとされてきましたが、HeatWave MySQLでは大量データの集計処理も超高速に実行できます。¥n¥nまた、HeatWave MySQLにはMySQL Databaseだけで(SQLだけで)機械学習を実現できるHeatWave AutoML、オブジェクトストレージ上のデータを高速にHeatWaveでの分析/集計対象にできるHeatWave Lakehouse、SQLだけでLLMを活用したりMySQLをベクトルストアとして使用できるHeatWave GenAIなどの拡張機能もあります。",
      "distance": 0.0958,
      "document_name": "https://objectstorage.ap-osaka-1.oraclecloud.com/省略/HeatWave_MySQL.txt"
    }
  ],
```

実行例：RAGを使ったLLMへの問い合わせ

```
  "vector_store": [
    "`RAG`.`test_embeddings`"
  ],
  "retrieval_info": {
    "method": "n_citations",
    "threshold": 0.0958
  }
}
1 row in set (0.00 sec)
```


ベクトルデータベース(ベクトルストア) としての使用方法

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？

この後のパートでは
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

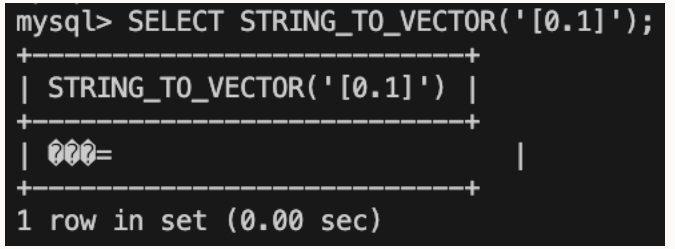
実行例：STRING_TO_VECTOR 関数、VECTOR_TO_STRING 関数

```
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 9.2.0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

補足：同じデータでもエンベディング方式が変わればベクトル間の距離は変わる

■ OCI Generative AIサービスと連携し、”cohere.embed-multilingual-v3.0”でエンベディングした場合のCOSINE距離

- 犬と猫：0.1125444769859314
- 犬とイルカ：0.23323941230773926
- 犬と小屋：0.21109145879745483

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: 0.18395888805389404
*************************** 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)
```

備考：”cohere.embed-multilingual-v3.0”でエンベディングした場合の実行例

```
mysql> SELECT sys.ML_EMBED_ROW('ミステリー', JSON_OBJECT('model_id', 'cohere.embed-multilingual-v3.0'))
    ->  INTO @query_embedding;
Query OK, 1 row affected (0.65 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: 0.2302156686782837
*************************** 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: 0.24487388134002686
*************************** 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: 0.259919524192810006
```

備考：”cohere.embed-multilingual-v3.0”でエンベディングした場合の実行例

```
*************************** 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: 0.275298237800598l4
*************************** 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: 0.2850002646446228
*************************** 6. 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: 0.3376314640045166
*************************** 7. 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: 0.37797898054122925
7 rows in set (0.00 sec)
```

HeatWave を無料で利用する方法

OCIのAlways Free Servicesで無料でHeatWaveを利用可能

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

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、9.3.0)
- 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 9.3.1以降で使用可能　※9.3.0以前でも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

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/


ORACLE