



GenAI Meets Your Data

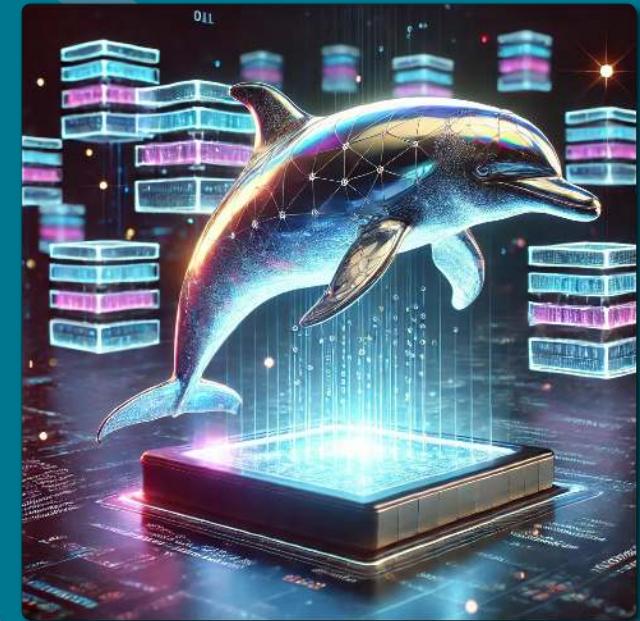
**Building a RAG System with
Your Own Content**

Frédéric Descamps

Community Manager

Oracle MySQL

MySQL BR Conf 2025 - October 2025



Who am I ?

about.me/lefred



Frédéric Descamps

- X @lefred
- 🦋 @lefredbe.bsky.social
- 🏡 @lefred@fosstodon.org
- [matrix] @lefred14:matrix.org
- MySQL Evangelist
- using MySQL since version 3.20
- devops believer
- living in 🇧🇪
- <https://lefred.be>



GenAI and Data *Theory*



GenAI

GenAI is very trendy and powerful. It's a set of tools and techniques that allow machines to learn from data and mimic some things humans can do, like providing answers.

For DBAs or developers using databases, it's an excellent opportunity to query their content differently and provide more value to the existing data.

When we use a chat service, like chatGPT, we can ask questions and get answers generated with data from all over ressources that was processed from the internet.

But what if we could do the same with our existing data stored in MySQL?

Disclaimer

I'm not a GenAI expert, but I want to share the processes and steps I learned while working on this project.



GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

embeddings

embeddings

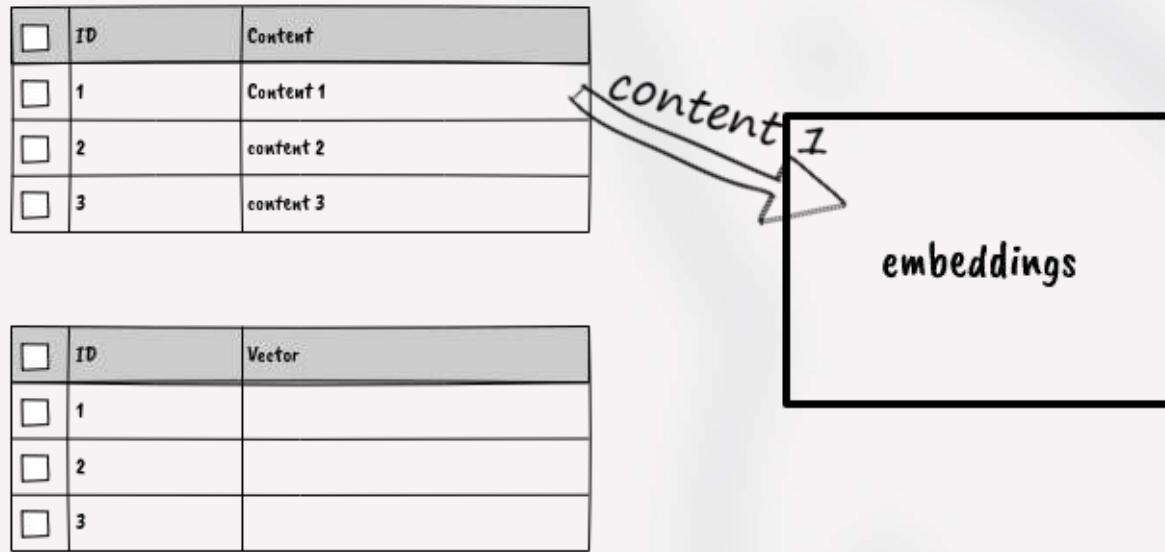
This is the process of converting text into a numerical representation. This is the first step in the process of using GenAI to query data.

For example:

"content 1" --> [1.2, 3.0, -1.5]

embeddings

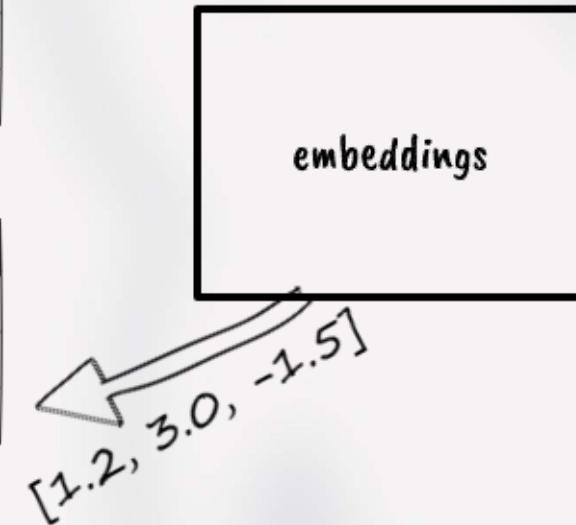
GenAI and Data - the process



GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3



GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3

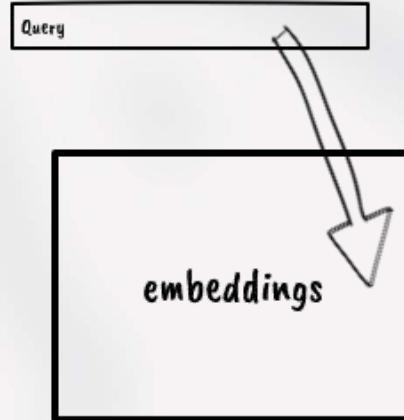
Query

embeddings

GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3



GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3



GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3

Query

embeddings

Query as Vector

Vector Search
(distance)

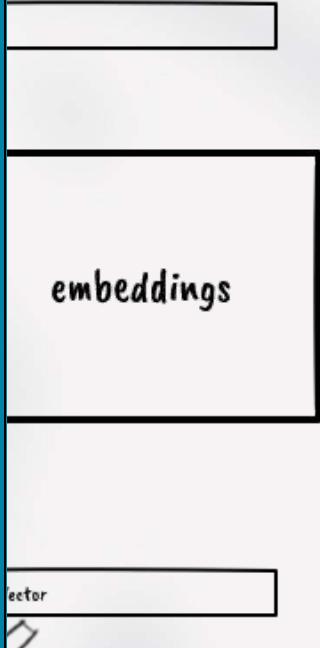
vector search

This is the process of querying the data using the embeddings generated in the previous step.

This is the third step in the process of using GenAI to query data.

To do this, we use the `distance()` function in MySQL HeatWave.

On other databases, this is where "vector indexes" are used to speed up the search.



GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3

Query

embeddings

Query as Vector

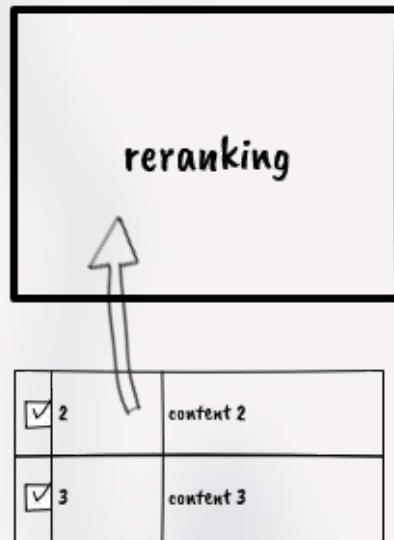
Vector Search
(distance)

<input checked="" type="checkbox"/>	2	content 2
<input checked="" type="checkbox"/>	3	content 3

GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3



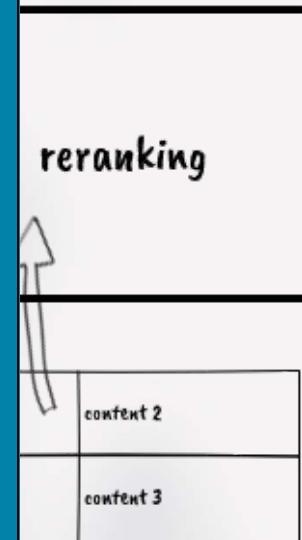
reranking

Reranking is the process where an initial set of results (the retrieved documents) is reordered based on more sophisticated criteria, such as relevance, user intent, or quality related to the query.

Initial retrieval methods (like a search engine) might not always surface the most relevant results at the top.

Reranking ensures that the most contextually appropriate or high-quality results are prioritized.

This step is optional but recommended to improve the quality of the results.



GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3

<input checked="" type="checkbox"/>	3	content 3
<input checked="" type="checkbox"/>	2	content 2



<input checked="" type="checkbox"/>	2	content 2
<input checked="" type="checkbox"/>	3	content 3

GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3

Query

<input checked="" type="checkbox"/>	3	content 3
<input checked="" type="checkbox"/>	2	content 2

rag / completion

GenAI and Data – the process

RAG

Retrieval-Augmented Generation (RAG)

This is the final step in the process of using GenAI to query data.

Inference

The ability of a large language model (LLM) to generate a response based on instructions and context provided by the user in the prompt. An LLM can generate new data, make predictions, or draw conclusions based on its learned patterns and relationships in the training data, without having been explicitly programmed.

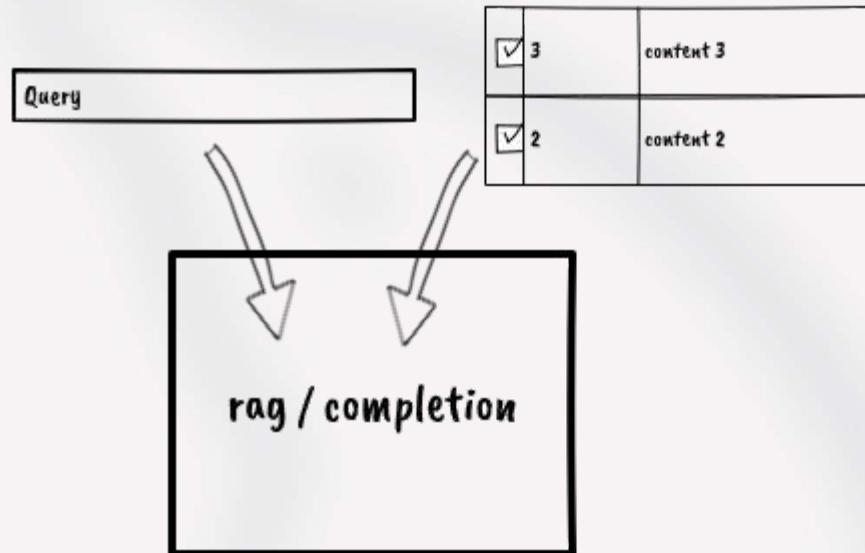
<input checked="" type="checkbox"/> 3		content 3
<input checked="" type="checkbox"/> 2		content 2

/ completion

GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3



GenAI and Data - the process

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3

Query

<input checked="" type="checkbox"/>	3	content 3
<input checked="" type="checkbox"/>	2	content 2

rag / completion

Answer

Our Project in OCI

WordPress



WordPress - lefred.be

Our goal is to use the content of my blog, lefred.be, and use it to generate answers with GenAI.

*We will create **embeddings** of the content using **GenAI** service in OCI and store the vectors in **MySQL HeatWave**.*

*Then we will use the **similarity** function, **distance()** to query the content.*

We have 2 options to generate answers based on the content of the blog:

- *Use of GPU with OCI GenAI service*
- *Use of CPU with HeatWave*

Prerequisites

- A *WordPress database*
- An *OCI account*
- A *MySQL HeatWave instance (version 9.x)*
- Access to *GenAI service in OCI*
- A *compute instance to run our code (or similar)*
- *Python or JavaScript*

WordPress database on HeatWave

We need to first check that we are running *MySQL HeatWave* 9.x:

The screenshot shows the MySQL Cloud interface for managing DB systems. The main title is "HeatWave MySQL > DB systems > DB system details". The DB system is named "wordpress-genai". On the left, there's a green hexagonal icon with "DBS" and "ACTIVE" text. The main content area has tabs for "DB system information", "Connections", and "Tags". Under "DB system information", there are sections for "General information" and "DB system configuration". In the "General information" section, the "Database version" field is highlighted with a red box and a red arrow pointing to it. The value shown is "9.2.0". Other fields in this section include OCID, Description (Wordpress MySQL for OCI GenAI), Compartment, Created (Sat, Jan 18, 2025, 07:37:03 UTC), and Last updated (Sat, Jan 18, 2025, 07:52:25 UTC). The "DB system configuration" section includes Shape (MySQL.8), ECPU Count (8), Memory (64 GB), Automatic storage expansion (Disabled), Storage size (1 TB), and Database version (9.2.0). To the right, there are sections for "Associated services", "High availability", "HeatWave cluster", and "Deletion plan".

WordPress database on HeatWave (2)

If this is the production of our WordPress database, we are fine to continue to the next step, if not, we have to import the data from our WordPress database to the MySQL HeatWave instance.

```
MySQL ➤ 10.0.1.180:3306 🔒 ➤ JS ➤ util.loadDump('wp_lefred', {'ignoreVersion': 'true'})  
Loading DDL and Data from 'wp_lefred' using 4 threads.  
Opening dump - done  
Target is MySQL 9.2.0-cloud (MySQL HeatWave Service). Dump was produced from MySQL 8.4.3  
WARNING: Destination is a MySQL HeatWave Service DB System instance but the dump was produced without the compatibility option.  
The 'ignoreVersion' option is enabled, so loading anyway. If this operation fails, create the dump once again with the 'ociimds'  
option enabled.  
WARNING: Destination MySQL version is different than the value of the 'targetVersion' option set when the dump was created: 8.4.  
3  
NOTE: Destination MySQL version is newer than the one where the dump was created.  
Scanning metadata - done  
Checking for pre-existing objects - done  
Executing common preamble SQL - done  
Executing DDL - done  
Executing view DDL - done  
Starting data load  
100% (102.89 MB / 102.89 MB), 102.89 MB/s (123.23K rows/s), 29 / 29 tables done  
Building indexes - done  
Executing common postamble SQL - done  
29 chunks (79.87K rows, 102.89 MB) for 29 tables in 1 schemas were loaded in 1 sec (avg throughput 102.85 MB/s, 79.84K rows/s)  
30 DDL files were executed in 0 sec.  
Data load duration: 1 sec  
Total duration: 1 sec  
0 warnings were reported during the load.  
MySQL ➤ 10.0.1.180:3306 🔒 ➤ JS
```

Prepare a table for embeddings

We create a table to store the embeddings of the content of our blog:

```
SQL > CREATE TABLE `wp_embeddings` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `vec` vector(1024) DEFAULT NULL,
  `wp_post_id` bigint unsigned DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Use of GPU

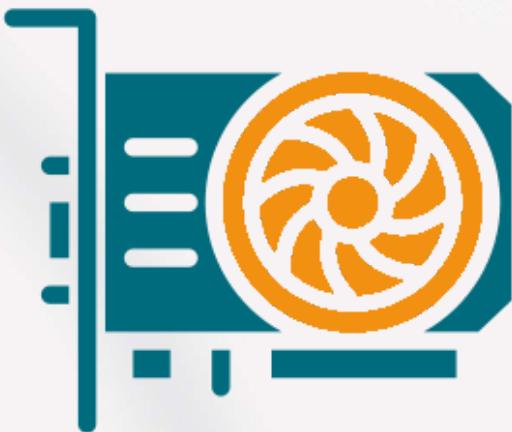
OCI GenAI service



OCI GenAI service

If we want to use GenAI related actions with GPU, we need to use the OCI GenAI service:

- we use an API to interact with the service for:
 - embeddings
 - reranking
 - inference
- we use an external application (in Python)



Python

OCI Account Configuration & modules



We use Python for GenAI

We will use Python to interact with the GenAI service in OCI therefore we need to provide our OCI credentials to the Python script.

And install the required modules:

- *oci*
- *mysql-connector-python*
- *unstructured*
- *beautifulsoup4*

Natural Language Toolkit

*The python **unstructured** module requires the Natural Language Toolkit (**nltk**) to be installed.*

unstructured is a Python library that provides a set of tools for working with unstructured data, such as text, images, and audio.

nltk is a Python library that provides a set of tools for building Programs to work with human language data.

Some data needs to be installed before being able to use it.

Natural Language Toolkit (2)

```
$ python3.9
Python 3.9.20 (main, Oct 24 2024, 07:04:44)
>>>
```

Natural Language Toolkit (2)

```
$ python3.9
Python 3.9.20 (main, Oct 24 2024, 07:04:44)
>>>
```

```
>>> import nltk
>>> nltk.download('punkt')
>>> nltk.download('punkt_tab')
>>> nltk.download('averaged_perceptron_tagger_eng')
```

Natural Language Toolkit (2)

```
$ python3.9  
Python 3.9.20 (main, Oct 24 2024, 07:04:44)  
>>>
```

```
>>> import nltk  
>>> nltk.download('punkt')  
>>> nltk.download('punkt_tab')  
>>> nltk.download('averaged_perceptron_tagger_eng')
```

```
$ du -sh nltk_data/*  
7.0M    nltk_data/taggers  
64M    nltk_data/tokenizers
```

OCI & MySQL HeatWave Configuration

We create a file to store all our configuration settings, `wp_config.py`:

```
COMPARTMENT = "<ENTER HERE THE GENAI COMPARTMENT ID>"  
CONFIG_FILE = "~/.oci/config"  
CONFIG_PROFILE = "DEFAULT"  
  
# Service endpoint  
ENDPOINT = "https://inference.generativeai.us-chicago-1.oci.oraclecloud.com"  
LLM_MODEL_ID = "<LLM CHAT GENAI MODEL ID>"  
  
DB_USER = "<YOUR DB USER ACCOUNT>"  
DB_PASSWORD = "<YOUR DB USER PASSWORD>"  
DB_HOST = "<YOUR DB HOST IP>"  
DB_PORT = 3306  
DB_SCHEMA = "<WORDPRESS SCHEMA>"
```

GenAI Compartment

The best region for GenAI in OCI is **us-chicago-1**, we need to find the **Compartment ID** where our GenAI service is running to add in the configuration file.

The screenshot shows the Oracle Cloud Infrastructure (OCI) console interface. The top navigation bar includes the MySQL logo, a Cloud icon, a search bar, and a dropdown for the region "US Midwest (Chicago)". The left sidebar has a "Analytics & AI" section highlighted with a red dashed border, containing links for Home, Compute, Storage, Networking, Oracle Database, Databases, Developer Services, Identity & Security, and Observability & Management. The main content area is titled "Analytics & AI" and contains several service categories: Analytics (Analytics Cloud), Data Lake (Big Data Service, Data Catalog, Data Integration, Data Flow), Messaging (Streaming, Connector Hub), Machine Learning (Data Science, Data Labeling, AI Services, Generative AI, Language, Speech, Vision, Document Understanding, Digital Assistant, Generative AI Agents), and Media Services (Media Flow, Media Flow Jobs, Media Streams). A red arrow points to the "Generative AI" link under the AI Services category.

GenAI Compartment (2)

☰ Cloud Search resources, services, documentation, and Marketplace US Midwest (Chicago) ⚙️ 🔔 ⓘ 🧑

Generative AI

Overview

Playground

Chat

Generation

Summarization

Embedding

Dedicated AI clusters

Custom models

Endpoints

Scope

Compartment

sandbox-lefred

x/sandbox-lefred

Embedding ?

To get started, select a model and a preset prompt example. Then, refine the prompts and parameters to fit your use cases. All model responses have [moderation filtering](#) applied for explicit content. Note that some models have [deprecation/retirement dates](#). View our [model list](#) for more details.

Model cohere.embed-english-light-v2.0 View model details Example Choose example View code

Sentence input *i*

Add a list of sentences or phrases to generate [embeddings](#) (maximum of 96 inputs).

+ Add sentence Upload file

1. X

Run Clear Character count - 0 | Token limit for each input - 512

Output vector projection *i*

Parameters

Truncate *i*

None

Output vector projection *i*

GenAI Compartment (3)

The screenshot shows the Oracle Cloud GenAI Embedding interface. On the left, a sidebar menu lists various AI services: Generative AI, Overview, Playground, Chat, Generation, Summarization, Embedding (which is selected), Dedicated AI clusters, Custom models, and Endpoints. Below this is a Scope section with a dropdown set to 'sandbox-lefred'. The main area is titled 'Embedding' and contains a 'Model' dropdown set to 'cohere.embed-english-light-v1'. A 'Sentence input' section allows adding sentences, with one sentence '1.' entered. Below this are 'Run' and 'Clear' buttons. To the right, a 'View code' section displays Python code for generating embeddings:

```
# Supports Python 3
#####
import oci

# Setup basic variables
# Auth Config
# TODO: Please update config profile name and use the compartmentId that has policies grant permissions to
CONFIG_PROFILE = "DEFAULT"
config = oci.config.from_file('~/oci/config', CONFIG_PROFILE)
# Service endpoint
endpoint = "https://inference.generativeai.us-chicago-1.oci.oraclecloud.com"
generative_ai_inference_client = oci.generative_ai_inference.GenerativeAiInferenceClient(config=config, s
inputs = []
embed_text_detail = oci.generative_ai_inference.models.EmbedTextDetails()
embed_text_detail.serving_mode = oci.generative_ai_inference.models.OnDemandServingMode(model_id="cohere.
embed_text_detail.inputs = inputs
embed_text_detail.truncate = "NONE"
embed_text_detail.compartment_id = "ocid1.compartment.oc1.."
embed_text_response = generative_ai_inference_client.embed_text(embed_text_detail)
# Print result
print("*****Embed Texts Result*****")
print(embed_text_response.data)

Copy code
```

We copy the `compartment_id` and paste it in our `wp_config.py` file.

Embeddings with OCI GenAI

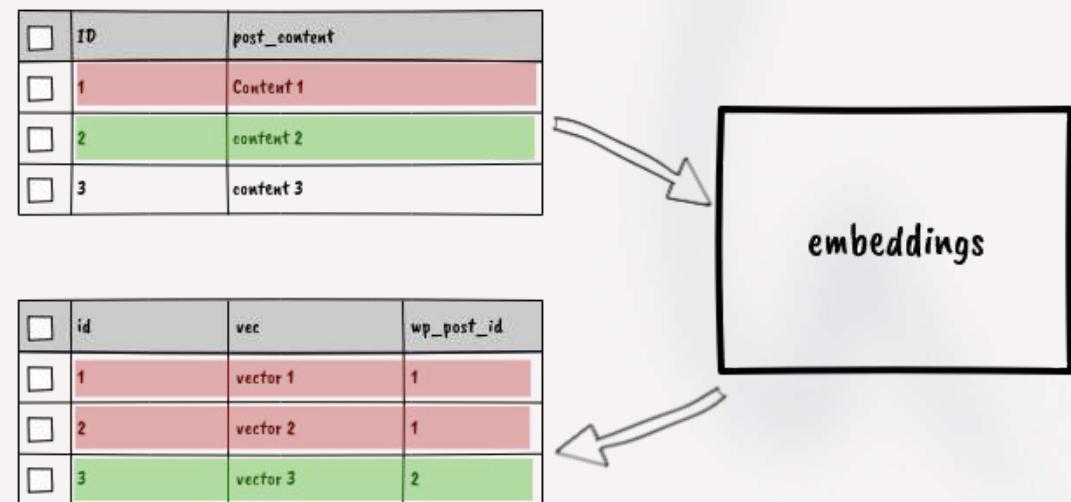
From Text to Vectors



Embeddings with OCI GenAI

Now we need to create the embeddings of the content of our blog, we will generate vectors of 1024 dimensions from the content of our blog transformed into text and cut into pieces of 96 characters.

We will store these vectors in the table `wp_embeddings` we created earlier.



Embeddings - code

```
cursor.execute("SELECT ID, post_content from wp_posts WHERE post_type='post' AND post_status='publish'")  
results = cursor.fetchall()  
  
for row in results:  
    print(".", flush=True, end="")  
    inputs = []  
    content = row[1]  
    content_text = BeautifulSoup(content, "html.parser")  
    for pi in content_text.find_all(string=lambda text: isinstance(text, bs4.element.ProcessingInstruction)):  
        pi.extract()  
  
    clean_html = str(content_text)  
    try:  
        content_text = partition_html(text=clean_html)  
        for content_text_el in content_text:  
            content = clean(content_text_el.text, extra_whitespace=True)  
            if len(content) > 1:  
                inputs.append(content)  
  
    except Exception as e:  
        print(f"Error with wp_post_id: {row[0]} !")
```

Embeddings - code (2)

```
if len(inputs) > 1:
    for input_block in range(0, len(inputs), 96):
        block = inputs[input_block : input_block + 96]
        pdebug(block)
        embed_text_detail = (
            oci.generative_ai_inference.models.EmbedTextDetails()
        )
        embed_text_detail.inputs = block
        embed_text_detail.truncate = embed_text_detail.TRUNCATE_END
        embed_text_detail.serving_mode = (
            oci.generative_ai_inference.models.OnDemandServingMode(
                model_id="cohere.embed-english-v3.0"
            )
        )
        embed_text_detail.compartment_id = compartment_id
        embed_text_detail.input_type = (
            embed_text_detail.INPUT_TYPE_SEARCH_DOCUMENT
        )

    try:
        embed_text_response = generative_ai_inference_client.embed_text(
            embed_text_detail
        )
    except Exception as e:
        print("Error while creating embeddings ", e)
        embeddings = []
```

Embeddings - code (3)

```
else:  
    embeddings = embed_text_response.data.embeddings  
    insert_stmt = (  
        "INSERT INTO wp_embeddings(vec, wp_post_id) "  
        "VALUES (string_to_vector(%s), %s)"  
    )  
  
    for emb in embeddings:  
        myvectorStr = ",".join(  
            str(item) for item in list(emb)  
        )  
        myvectorStr = "[" + myvectorStr + "]"  
        data = (myvectorStr, row[0])  
        cursor.execute(insert_stmt, data)  
  
cnx.commit()
```

Embeddings Models in OCI

*Those models also depend on the **region** where the GenAI service is running.*

*As per September 2025, on **us-chicago-1**, we have the following models for embeddings:*

- *cohere.embed-english-light-v2.0*
- *cohere.embed-english-light-v3.0*
- *cohere.embed-english-v3.0*
- *cohere.embed-multilingual-image-v3.0*
- *cohere.embed-multilingual-light-v3.0*
- *cohere.embed-multilingual-v3.0*
- *cohere.embed-v4.0*

Embeddins - Run

Embeddings - Check

```
SQL > SELECT COUNT(*) FROM wp_embeddings;  
+-----+  
| count(*) |  
+-----+  
| 10934 |  
+-----+  
1 row in set (0.0052 sec)
```

```
SQL > SELECT wp_post_id, COUNT(*) FROM wp_embeddings GROUP BY wp_post_id ORDER BY 2 DESC LIMIT 5;  
+-----+-----+  
| wp_post_id | count(*) |  
+-----+-----+  
| 5395 | 91 |  
| 45 | 86 |  
| 6609 | 83 |  
| 3068 | 80 |  
| 6452 | 79 |  
+-----+-----+
```

RAG

Retrieval-Augmented Generation



From prompt to relevant documents

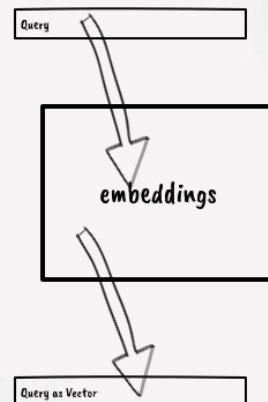
We need to reduce all the content of our blog to only the most relevant ones in relation to the prompt.

We use the **distance()** function to query the content. We perform a **COSINE** similarity search to find the most relevant documents.

We need first to convert the prompt to a vector using the same model we used to generate the embeddings.

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3



From prompt to relevant documents (2)

```
def generate_embeddings_for_question(question_list):
    print("Performing Embeddings of the prompt...")
    embed_text_detail = oci.generative_ai_inference.models.EmbedTextDetails()
    embed_text_detail.inputs = question_list
    embed_text_detail.input_type = embed_text_detail.INPUT_TYPE_SEARCH_QUERY
    embed_text_detail.serving_mode = (
        oci.generative_ai_inference.models.OnDemandServingMode(
            model_id="cohere.embed-english-v3.0"
        )
    )
    embed_text_detail.compartment_id = compartment_id
    embed_text_response = generative_ai_inference_client.embed_text(embed_text_detail)
    return embed_text_response
```

From prompt to relevant documents (3)

We need to find similar docs in the database:

```
def search_data(cursor, query_vec, list_dict_docs):  
    print("Performing Vector Search Similarity...")  
    myvectorStr = ",".join(str(item) for item in query_vec)  
    myvectorStr = "[" + myvectorStr + "]"  
  
    relevant_docs = []  
    mydata = myvectorStr  
    cursor.execute(  
        """  
            select distinct wp_post_id from (  
                select id, wp_post_id, distance from  
                    (select id, wp_post_id,  
                        DISTANCE(string_to_vector(%s), vec, 'COSINE') distance  
                     from {}.wp_embeddings  
                     order by distance limit 100) a  
                     where distance < 1 order by distance) b limit 50  
        """.format(  
            wp_config.DB_SCHEMA  
)  
        ,  
        [myvectorStr],  
    )
```

From prompt to relevant documents (4)

We clean the content of the documents by removing the HTML tags and we will use the RAG model to generate the answer passing the content of the documents and the prompt:

```
prompt_template = """  
{question} \n  
Answer the question based on the text provided and also return the relevant document numbers  
where you found the answer. If the text doesn't contain the answer, reply that the  
answer is not available.  
"""  
  
prompt = prompt_template.format(question=query)  
  
llm_response_result = query_llm_with_prompt(similar_docs, prompt)  
response = {}  
response["message"] = query  
response["text"] = llm_response_result  
response["documents"] = [  
    {"id": doc.doc_id, "snippet": doc.doc_text} for doc in similar_docs_subset  
]
```

LLM Model ID - On Demand Chat

We need to add the `model_id` in our `wp_config.py` file:

The screenshot shows the Oracle Cloud Generative AI Chat interface. The left sidebar has a red underline under 'Chat'. The main area is titled 'Chat' with a note: 'To get started, select a model for explicit content. Note that some models may have explicit content.' A 'Model' dropdown is set to 'cohere.command-r-08-2024'. The right panel is titled 'View code' and contains Python code examples:

```
# Service endpoint
endpoint = "https://inference.generativeai.us-chicago-1.oci.oraclecloud.com"

generative_ai_inference_client = oci.generative_ai_inference.GenerativeAiInferenceClient(config=config,
chat_detail = oci.generative_ai_inference.models.ChatDetails()

chat_request = oci.generative_ai_inference.models.CohereChatRequest()
chat_request.message = "{input}"
chat_request.max_tokens = 600
chat_request.temperature = 1
chat_request.frequency_penalty = 0
chat_request.top_p = 0.75
chat_request.top_k = 0

chat_detail.serving_mode = oci.generative_ai_inference.models.OnDemandServingMode(model_id="ocid1.general-00000000-0000-0000-0000-000000000000")
chat_detail.chat_request = chat_request
chat_detail.compartment_id = compartment_id
chat_response = generative_ai_inference_client.chat(chat_detail)
# Print result
print("*****Chat Result*****")
print(vars(chat_response))
```

LLM Models in OCI

Availability of models in OCI GenAI service depends on the region.

From coh~~e~~re to meta.llama and xai.grok like:

- coh~~e~~.command-a-03-2025
- coh~~e~~.command-r-08-2024
- coh~~e~~.command-r-plus-08-2024
- meta.llama-3.1-405b-instruct
- meta.llama-3.1-70b-instruct
- meta.llama-3.2-90b-vision-instruct
- meta.llama-3.3-70b-instruct
- meta.llama-4-scout-17b-16e-instruct
- xai.grok-3
- xai.grok-3-fast
- xai.grok-3-mini
- xai.grok-3-min-fast
- xai.grok-4

LLM Response

```
def query_llm_with_prompt(documents, prompt):
    print("Generating the Answer...")
    my_documents = []
    for docs in documents:
        my_documents.append({"id": f'{docs.doc_id}', "text": docs.doc_text})

    generative_ai_inference_client = (
        oci.generative_ai_inference.GenerativeAiInferenceClient(
            config=config,
            service_endpoint=endpoint,
            retry_strategy=oci.retry.NoneRetryStrategy(),
            timeout=(10, 240),
        )
    )
    chat_detail = oci.generative_ai_inference.models.ChatDetails()
    chat_request = oci.generative_ai_inference.models.CohereChatRequest()
    chat_request.documents = my_documents
    chat_request.message = prompt
    chat_request.max_tokens = 600
    chat_request.temperature = 1
    chat_request.frequency_penalty = 0
    chat_request.top_p = 0.75
    chat_request.top_k = 0
    chat_detail.serving_mode = oci.generative_ai_inference.models.OnDemandServingMode(
        model_id=wp_config.LLM_MODEL_ID
    )
    chat_detail.chat_request = chat_request
    chat_detail.compartment_id = compartment_id
    chat_response = generative_ai_inference_client.chat(chat_detail)

    return vars(chat_response)
```

LLM Response - limiting documents

We saw, that we send several documents to the RAG model, in the previous code<br we limited to 50, but too many documents can be confusing for the model:

```
x - + opc@compute-genai:~/genai
[opc@compute-genai genai]$ python3.9 wp_genai.py
You are now connected to MySQL Enterprise - Cloud 9.2.0-cloud
What is your question? What is MySQL HeatWave?
Performing Embeddings of the prompt...
Performing Vector Search Similarity...
Generating the Answer...
MySQL HeatWave is a high-performance in-memory <co: 0,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42>query</co: 0,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19
- [6452]: MySQL Database Service - find the info: part 5 - HeatWave
- [4375]: Oracle Live: MySQL & HeatWave
- [6714]: Moodle on OCI with MySQL HeatWave: Extended Architectures - part 2
- [7058]: Is MySQL HeatWave really faster ?
- [4953]: Using MySQL Database Service in OCI - Part 1: creating a MySQL DB System
- [4973]: Using MySQL Database Service in OCI - Part 3: faster with MySQL HeatWave !
- [7149]: JSON in MySQL HeatWave: why is data or my query not off-loaded to HeatWave Cluster ?
- [6918]: Backup your MySQL instance to the Cloud
- [6168]: Deploying Zabbix with MySQL HeatWave Database Service in OCI
[opc@compute-genai genai]$
```

LLM Response - limiting documents (2)

If we limit the document returned by the vector search to 10, we can see that the answer is more relevant:

```
x - + opc@compute-genai:~/genai
[opc@compute-genai genai]$ python3.9 wp_genai.py
You are now connected to MySQL Enterprise - Cloud 9.2.0-cloud
What is your question? What is MySQL HeatWave?
Performing Embeddings of the prompt...
Performing Vector Search Similarity...
Generating the Answer...
MySQL HeatWave is a high-performance in-memory query accelerator for MySQL Database Service on Oracle Cloud Infrastructure. It can boost the performance of MySQL queries, providing faster, more reliable, and cost-effective access to data. It can also accelerate analytics workloads and increase the performance of MySQL databases.

HeatWave is available on Oracle Cloud Infrastructure and some other clouds. It can be used to run analytics queries quickly and to run such queries on files using LakeHouse. It can also be used to store data for Moodle on OCI and to add High Availability and Query Acceleration.

The first version of MySQL HeatWave was released under the banner of artificial intelligence and it includes a VECTOR datatype.
- [6452]: MySQL Database Service - find the info: part 5 - HeatWave
- [7427]: Using Oracle Analytics Cloud with MySQL HeatWave
- [6677]: Deploy Moodle on OCI with Ampere and MySQL HeatWave
- [7545]: Use OCI GenAI, and MySQL HeatWave to interact with your WordPress content
[opc@compute-genai genai]$
```

Reranking

Improve the quality of the results

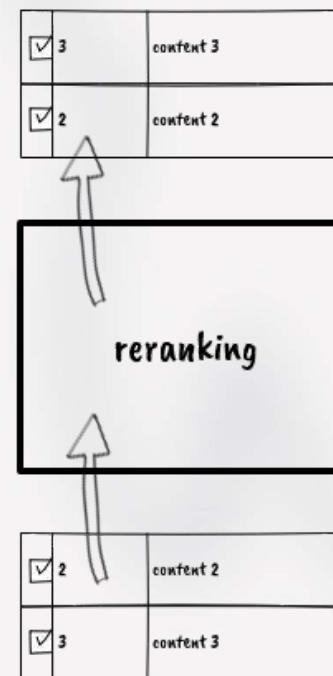


Reranking

The vector search is not always returning the most relevant documents, we can improve the quality of the results by reranking the documents, using a model that will reorder the documents based on more sophisticated criteria, such as relevance, user intent, or quality related to the query.

	ID	Content
	1	Content 1
	2	content 2
	3	content 3

	ID	Vector
	1	vector 1
	2	vector 2
	3	vector 3



Reranking - OCI

The documentation of the GenAI service in OCI provides information related to the API:

The screenshot shows a web browser displaying the Oracle Cloud Infrastructure Documentation. The title bar includes the Oracle logo, Help Center, and a search bar. The main content area is titled "Oracle Cloud Infrastructure Documentation / API Reference and Endpoints". Below this, a specific API endpoint is detailed:

RerankText GENERATIVEAIINFERENCE

POST /20231130/actions/rerankText

Reranks the text responses based on the input documents and a prompt.

Rerank assigns an index and a relevance score to each document, indicating which document is most related to the prompt.

Request

RerankText

Parameters

Name	Where	Description
opc-retry-token	header	• Required: no

Reranking - OCI (2)

But there are no examples yet as it seems no models are supporting this feature yet in OCI on demand GenAI service:

Body

The request body must contain a single [RerankTextDetails](#) resource.

Examples

[Java SDK](#) [Python SDK](#) [Go SDK](#) [TypeScript SDK](#) [.NET SDK](#) [Ruby SDK](#)

Sorry, we do not have a Python SDK example for this request.

Responses

Status: 200

The text response was successfully generated.

Reranking - OCI (3)

To confirm:

```
$ python3.9 test.py
Error: cohere.embed-multilingual-v3.0 does not support TextRerank
Error: meta.llama-3.1-405b-instruct does not support TextRerank
Error: meta.llama-3.2-90b-vision-instruct does not support TextRerank
Error: meta.llama-3.1-70b-instruct does not support TextRerank
Error: cohere.command-r-08-2024 does not support TextRerank
Error: cohere.command-r-plus-08-2024 does not support TextRerank
Error: cohere.embed-english-v3.0 does not support TextRerank
Error: cohere.embed-english-light-v3.0 does not support TextRerank
Error: cohere.embed-multilingual-light-v3.0 does not support TextRerank
```

Reranking - using cohere service

We can use the **cohere** service to rerank the documents.

We need to add our `COHERE_API_KEY` in our `wp_config.py` file.

You can get a free one from <https://dashboard.cohere.com/>.



Reranking - using cohere service (2)

```
import cohere

cohere_client = cohere.Client(wp_config.COHERE_API_KEY)

...
if len(rerank_docs) > 1:
    print("Performing Reranking...")
    rerank_results = cohere_client.rerank(
        query=query,
        documents=rerank_docs,
        top_n=5,
        model="rerank-english-v3.0",
        return_documents=True,
    )
```

Reranking - using cohere service (3)

Now we can test, we still limit our vector search to 50 documents but we rerank them and use only the top 5 for Generating the answer:

```
x - □ opc@compute-genai:~/genai
[opc@compute-genai genai]$ python3.9 wp_rerank_genai.py
You are now connected to MySQL Enterprise - Cloud 9.2.0-cloud
What is your question? What is MySQL HeatWave?
Performing Embeddings of the prompt...
Performing Vector Search Similarity...
Performing Reranking...
Generating the Answer...
MySQL HeatWave is a query accelerator that boosts the performance of MySQL queries. It is a Database as a Service (DBaaS) provided by Oracle, designed to accelerate analytics workloads and increase the performance of MySQL databases.

The key features of MySQL HeatWave include:
- Allowing users to run analytics queries (aka OLAP) very quickly using the HeatWave cluster.
- Scaling out to thousands of cores, which makes it faster at query performance than Amazon Aurora and Amazon Redshift.
- Incorporating Machine Learning to provide database administrators and developers with an optimal experience.
- Offering straightforward tools that streamline the process of creating predictive models through data training and scoring.
- Integrating easily with third-party tools like Grafana through a simple MySQL connection.

MySQL HeatWave can be used on Oracle Cloud Infrastructure (OCI) and other clouds.
- [6452]: MySQL Database Service - find the info: part 5 - HeatWave
- [7058]: Is MySQL HeatWave really faster ?
- [7427]: Using Oracle Analytics Cloud with MySQL HeatWave
- [7457]: Machine Learning with MySQL HeatWave and predictions in Grafana
- [4375]: Oracle Live: MySQL & HeatWave
[opc@compute-genai genai]$
```

Reranking - using OCI GenAI service **NEW**

OCI GenAI service now supports reranking with the `cohere.rerank.3-5` model that was released on **2025-05-14** in OCI.

But this model is only available on *dedicated AI Clusters*:

```
cohere.rerank-v3.5
  State: ACTIVE - Creating Base Model
  capabilities: ['TEXT_RERANK']
  created: 2025-05-14 20:05:05.321000+00:00 - deprecated: 2025-05-01 00:00:00+00:00 - on demand retired: 2025-05-01 00:00:00+00:00

cohere.rerank-multilingual-v3.1
  State: ACTIVE - Creating Base Model
  capabilities: ['TEXT_RERANK']
  created: 2024-11-26 20:25:33.745000+00:00 - deprecated: None - on demand retired: None

cohere.rerank-english-v3.1
  State: ACTIVE - Creating Base Model
  capabilities: ['TEXT_RERANK']
  created: 2024-06-04 00:34:49.790000+00:00 - deprecated: None - on demand retired: None
```

Reranking - using OCI GenAI service **NEW**

Dedicated Cluster

The screenshot shows the Oracle Cloud GenAI interface for creating a Dedicated AI cluster. On the left, the sidebar includes options like Overview, Playground, Chat, Generation, Summarization, Embedding, Dedicated AI clusters, Custom models, Endpoints, Scope, Compartment (set to sandbox-lefred), Tag filters, and Custom filters. The main area is titled "Dedicated AI cluster" and "Create dedicated compute infrastructure". It features a "Create dedicated AI cluster" button and a table for managing clusters. A modal window is open, listing various AI models under categories: Generation, Summarization, Embedding, and Rerank. The "Rerank" section is highlighted with a red arrow pointing to the "cohere.rerank-v3.5" option, which is currently selected and highlighted in blue. Other options in the Rerank list include "cohere.rerank-english-v3.1" and "cohere.rerank-multilingual-v3.1". Below the dropdown, there's a note: "This will provision - unit of - .", a checkbox for committing hours, and buttons for "Create" and "Cancel". The top right corner shows the location "US Midwest (Chicago)" and other navigation icons.

Reranking - using OCI GenAI service **NEW**

Dedicated Cluster

The screenshot shows the OCI GenAI service interface. On the left, there's a sidebar with options like Overview, Playground, Chat, Generation, Summarization, Embedding, and Dedicated AI clusters (which is selected and underlined). Below that are Custom models and Endpoints. The main content area is titled "Dedicated AI clusters" and contains a sub-instruction: "Create dedicated compute infrastructure for fine-tuning custom models and hosting them. Learn about [dedicated AI clusters](#)". A "Create dedicated AI cluster" button is present. To the right is a table with columns: Name, State, Created on, Type, Remaining endpoint capacity, Unit size, Number of units, Created by, and Description. One row is shown, corresponding to the "ai-rerank-cluster" entry in the sidebar. The row details are: Name (ai-rerank-cluster), State (Creating), Created on (Tue, 27 May 2025 08:42:18 UTC), Type (Hosting), Remaining endpoint capacity (50), Unit size (RERANK_COHERE), Number of units (1), Created by (default/ [redacted]), and Description (lefred reranking test). A green toast message at the top right says: "Successfully initiated creation of ai-rerank-cluster : ocid1.generativeaidedicatedaicluster.oc1.us-chicago-1. [redacted]".

Name	State	Created on	Type	Remaining endpoint capacity	Unit size	Number of units	Created by	Description
ai-rerank-cluster	Creating	Tue, 27 May 2025 08:42:18 UTC	Hosting	50	RERANK_COHERE	1	default/ [redacted]	lefred reranking test

Reranking - using OCI GenAI service **NEW**

Dedicated Cluster - endpoint

The screenshot shows the OCI GenAI service interface. On the left, there's a sidebar for the 'ai-rerank-cluster' dedicated AI cluster, displaying its general information, compartment (sandbox-lefred), OCID, description, lifecycle details (Created Dedicated AI Cluster), state (Active), and cluster type (Hosting). Below this are sections for Resources (Endpoints selected), Endpoints, and Create endpoint.

The main area is titled 'Create endpoint'. It has fields for Compartment (set to sandbox-lefred), Name (optional), Description (optional), and Hosting Configuration. Under Hosting Configuration, it lists 'Models in sandbox-lefred' (cohere.rerank-v3.5 selected), 'Model version' (3.5), and a note: 'You must have a hosting dedicated AI cluster that matches the base model of the selected model. Either select a listed hosting cluster or create one. Learn about [matching clusters to base models](#)'. It also lists 'Dedicated AI clusters in sandbox-lefred' (ai-rerank-cluster selected, with 49 remaining endpoint capacity).

At the bottom of the 'Create endpoint' form are 'Create endpoint' and 'Cancel' buttons.

Reranking - using OCI GenAI service **NEW**

Dedicated Cluster - endpoint

The screenshot shows the OCI GenAI service interface. At the top, there's a navigation bar with 'Cloud' and a search bar. Below it, the location is set to 'US Midwest (Chicago)'.

The main page title is 'ai-rerank-cluster'. A note below the title says: 'If this dedicated AI cluster is type Fine-Tuning, select this cluster when creating custom models. If it is type Hosting, select it when creating endpoints. Learn about [dedicated AI clusters](#)'.

Below the note are several buttons: 'Edit', 'Add tags', 'Move dedicated AI cluster', and 'Delete'.

The 'General information' tab is selected, showing the following details:

- Compartment: [REDACTED]tfcq [Show](#) [Copy](#)
- OCID: [REDACTED]b2q [Show](#) [Copy](#)
- Description: lefred reranking test
- Lifecycle details: Created Dedicated AI Cluster
- State: ● Active
- Cluster type: Hosting
- Created on: Tue, 27 May 2025 08:42:18 UTC
- Created by: [REDACTED]
- Remaining endpoint capacity: 49
- Unit size: RERANK_COHERE
- Number of units: 1

The 'Endpoints' section contains a table:

Name	State	Created on	Model name	Model version	Dedicated AI Cluster	Description
generativeaiendpoint202505[REDACTED]11	● Active	Tue, 27 May 2025 08:45:11 UTC	cohere.rerank-v3.5	3.5	[REDACTED]vuo2g	Show Copy

On the left, a sidebar lists 'Resources' with 'Endpoints' selected, and other options like 'Cluster metrics' and 'Work requests'.

Reranking - using OCI GenAI service **NEW**

In action

```
x - + v      opc@my-oci-compute:~/oci-genai-wp
[opc@my-oci-compute oci-genai-wp]$ python wp_rerank_genai_oci.py
You are now connected to MySQL Enterprise - Cloud 9.3.0-u1-cloud
What is your question? What is MySQL HeatWave?
Performing Embeddings of the prompt...
Performing Vector Search Similarity...
Performing Reranking...
Generating the Answer...
MySQL HeatWave is the MySQL DBaaS provided by Oracle in OCI and some other clouds.

It allows you to run analytics queries (aka OLAP) very quickly using the HeatWave cluster. It is also
possible to run such queries on files using LakeHouse.

In OCI, you can use Oracle Analytics Cloud.

It is a high-performance in-memory query accelerator for MySQL Database Service on Oracle Cloud Infras
tructure.

It is designed to accelerate analytics workloads (OLAP) and increase the performance of your MySQL dat
abases. This is achieved through the use of in-memory processing, advanced algorithms, and machine lea
rning techniques to optimise query performance.

You can find this information in documents 0, 1, 2 and 4.
- [7427]: Using Oracle Analytics Cloud with MySQL HeatWave
- [4375]: Oracle Live: MySQL & HeatWave
- [6452]: MySQL Database Service - find the info: part 5 - HeatWave
- [7058]: Is MySQL HeatWave really faster ?
[opc@my-oci-compute oci-genai-wp]$
```

Conclusion

We saw how we can use GenAI to also query existing data stored in MySQL and generate answers based on that data.

It doesn't require too much coding skills and the result is stunning!

Welcome to a new world of GenAI, LLMS, RAG for your data, and this service is available in OCI!

In OCI with a free account

The screenshot shows the Oracle Cloud Generative AI Chat interface. On the left, a sidebar lists categories like Overview, Playground, Chat (which is selected), Embedding, Dedicated AI clusters, Custom models, and Endpoints. Below these are sections for Scope and Compartment, with a red box highlighting the compartment dropdown. The main area is titled "Chat" and contains a message card with the question "What are the preFOSDEM MySQL Days?". The response explains that these events are focused on MySQL database system, held before the FOSDEM conference, and provide opportunities for developers, users, and enthusiasts to learn and network. A large "Parameters" panel on the right allows adjusting settings such as Maximum output tokens (600), Preamble override, Temperature (1), Top p (0.75), Top k (0), Frequency penalty (0), and Presence penalty (0).

To use this service or resource, you must [upgrade](#) to a paid account.

Cloud

Search resources, services, documentation, and Marketplace

Germany Central (Frankfurt)

Upgrade

Generative AI

Overview

Playground

Chat

Embedding

Dedicated AI clusters

Custom models

Endpoints

Scope

Compartment

Type a message...

Submit

Clear chat

Model

cohere.command-r-08-2024 v1.7

View model details

Example

Choose example

View code

What are the preFOSDEM MySQL Days?

The preFOSDEM MySQL Days are a series of events and workshops focused on the MySQL database system, held in the days leading up to the FOSDEM conference. FOSDEM, which stands for Free and Open Source Developers' European Meeting, is a large annual event for the open-source software community. The preFOSDEM MySQL Days provide an opportunity for developers, users, and enthusiasts to come together and discuss various aspects of MySQL, including new features, performance tuning, development, and best practices. It offers a platform for sharing knowledge, learning from experts, and networking with the MySQL community. These events often feature talks, tutorials, and hands-on sessions led by experienced MySQL developers and contributors. They cover a wide range of topics, catering to both beginners and advanced users. The schedule typically includes presentations on the latest MySQL innovations, case studies, and practical workshops where attendees can apply their knowledge directly. The preFOSDEM MySQL Days are an integral part of the FOSDEM conference experience, offering a dedicated space for MySQL enthusiasts to engage deeply with the technology and its community.

Parameters

Maximum output tokens (600)

Preamble override

Temperature (1)

Top p (0.75)

Top k (0)

Frequency penalty (0)

Presence penalty (0)

The Source Code

<https://github.com/lefred/oci-genai-wp>

Use of CPU

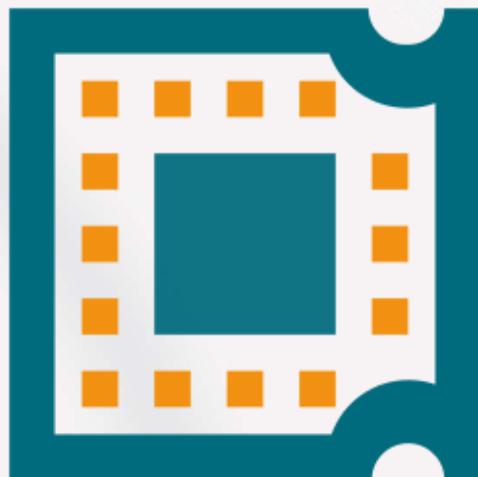
HeatWave GenAI



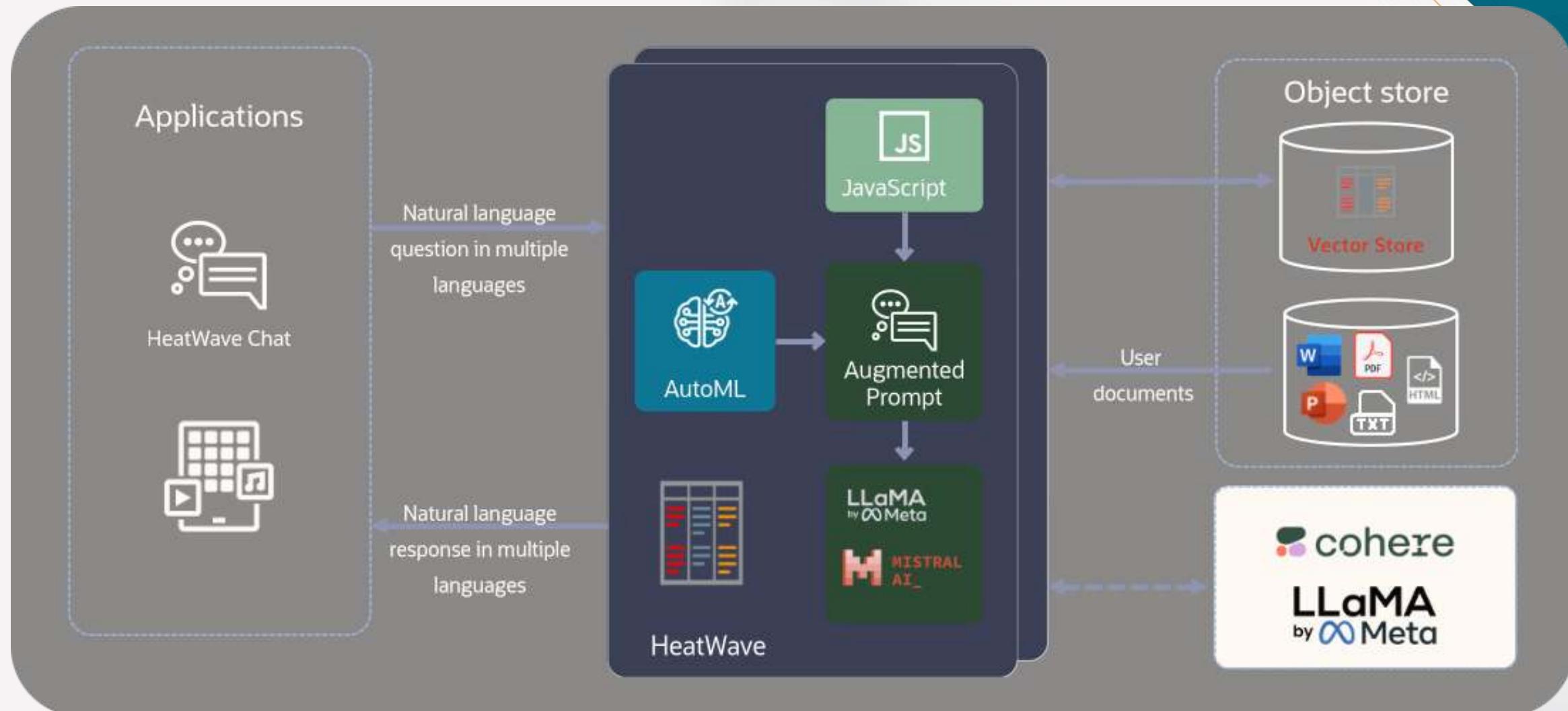
HeatWave GenAI

With HeatWave GenAI, we have simplified all the process and everything can be done directly in the database.

- *we use optimized models for CPU in HeatWave*
 - *everything is done in HeatWave*
- *we use in database stored procedures (in JavaScript)*



HeatWave Vector Store



Embeddings with HeatWave

To be able to use the GenAI features in HeatWave, we need to enable the HeatWave cluster:

The screenshot shows the MySQL Cloud console interface for managing a MySQL HeatWave instance named "MySQLHW-OCI".

DBS ACTIVE

MySQLHW-OCI

DB system information Connections Tags

General information

- OCID: [REDACTED] [Show](#) [Copy](#)
- Description: MySQL HeatWave instance in OCI [Edit](#)
- Compartment: [REDACTED] x/sandbox-lefred
- Created: Fri, Mar 7, 2025, 06:46:22 UTC
- Last updated: Wed, May 21, 2025, 09:52:50 UTC

DB system configuration

- Shape: MySQL.4 [Edit](#) ⓘ
- ECPU Count: 4 ⓘ
- Memory: 32 GB ⓘ
- Automatic storage expansion: Disabled [Edit](#)
- Storage size: 50 GB ⓘ [Edit](#)
- Database version: 9.3.0 - Innovation [Edit](#)
- Configuration: MySQL.4.Standalone [Edit](#)

Management

- Database mode: Read/write ⓘ
- Access mode: All users ⓘ [Edit](#)

Associated services

- Database Management: Disabled [Enable](#) ⓘ
- Ops Insights: Disabled [Enable](#) ⓘ

High availability

- High availability: Disabled [Enable](#) ⓘ

HeatWave cluster

- HeatWave cluster: [Details](#) ⓘ
- State: ⓘ Inactive
- HeatWave Lakehouse: Disabled ⓘ

HeatWave GenAI - Supported Models

If the HeatWave cluster is not enabled or inactive, the models are not available:

```
MySQL > SELECT * FROM sys.ML_SUPPORTED_LLMS WHERE model_type = "embedding";  
ERROR: 1146 (42S02): Table 'sys.ML_SUPPORTED_LLMS' doesn't exist
```

HeatWave GenAI - Supported Models

When active, we can see the supported models:

```
MySQL > SELECT * FROM sys.ML_SUPPORTED_LLMS WHERE model_type = "embedding";  
+-----+-----+  
| model_name          | model_type |  
+-----+-----+  
| minilm              | embedding  |  
| all_minilm_l12_v2   | embedding  |  
| multilingual-e5-small | embedding  |  
| cohere.embed-english-light-v3.0 | embedding  |  
| cohere.embed-multilingual-v3.0 | embedding  |  
| cohere.embed-multilingual-light-v3.0 | embedding  |  
| cohere.embed-english-v3.0 | embedding  |  
+-----+-----+  
7 rows in set (0.0017 sec)
```

Embeddings with HeatWave (2)

post_title (simple text)

To perform the embeddings, we need use the `sys.ML_EMBED_ROW()` function.

Let's do it for the titles of each post:

```
MySQL > SET @embeddOptions = '{"model_id": "minilm"}';
MySQL > INSERT INTO wp_embeddings SELECT 0,
           sys.ML_EMBED_ROW(post_title, @embeddOptions), ID
           FROM wp_posts WHERE post_type="post" AND post_status="publish";
Query OK, 607 rows affected (6 min 20.0115 sec)
```

Embeddings with HeatWave (3)

post_content (complex text)

The content of an article is more complex as it is usually longer and can contain HTML tags.

As we don't want to use any external tool or API to leave our data inside MySQL HeatWave, we use a stored procedure written in JavaScript to clean the content and perform the embeddings.

JavaScript



Embeddings with HeatWave (4)

wp_create_embb_js

```
DROP PROCEDURE IF EXISTS wp_create_embb_js;
CREATE PROCEDURE wp_create_embb_js()
LANGUAGE JAVASCRIPT
AS $$
let s = session.sql("SELECT count(*) FROM wp_posts where post_type='post' and post_status='publish'")
let res = s.execute()
let row = res.fetchOne()
let total = row[0]
let batch = 10
let processed = 0
let last_id = 0
while (processed < total) {
    s = session.sql("SELECT ID FROM wp_posts where post_type='post'
        and post_status='publish' and ID > " + last_id
        +" ORDER BY ID LIMIT " + batch)
    res = s.execute()
    row = res.fetchOne()
    while(row) {
        let id = row[0]
        last_id = id
        let s2 = session.sql("CALL wp_create_row_embb_js(" + id + ")")
        s2.execute()
        processed += 1
        row = res.fetchOne()
    }
}
let stmt_out = session.sql('Select "Embeddings done! ' + total +
    ' posts processed" as "wp_create_embb_js"', {passResultToClient: true})
stmt_out.execute()
$$;
```

Embeddings with HeatWave (4)

wp_create_row_embb_js

```
DROP PROCEDURE IF EXISTS wp_create_row_embb_js;
CREATE PROCEDURE wp_create_row_embb_js(id INT)
LANGUAGE JAVASCRIPT
AS $$
let s = session.sql("SELECT post_content FROM wp_posts WHERE ID=" + id)
let res = s.execute()
let row = res.fetchOne()

while(row) {
    let content = row[0]
    let content_text = content.replace(/<\/?[^>]+(>|$/g, "");
    content_text = content_text.replace(/\r\n/g, " ");
    content_text = content_text.replace(/\n/g, " ");
    // replace all double quotes with \" to escape them
    content_text = content_text.replace(/(?<!\\")/g, '\\\"');
    // replace all single quotes with \
    content_text = content_text.replace(/(?<!\\')/g, '\\\'');
    // replace all lines starting with +- with empty string
    content_text = content_text.replace(/[+-].*[+-]/gm, "")
    // remove all empty lines
    content_text = content_text.replace(/^\s*\r\n/gm, "")

    // check if content_text is empty
    if (content_text.length == 0) {
        return "Content is empty"
    }
}
```

```
// split content_text into strings of 100 characters
// but not breaking words
let tokens = []
let token = ""
let words = content_text.split(" ")
for (let i = 0; i < words.length; i++) {
    if (token.length + words[i].length < 100) {
        token += " " + words[i]
    } else {
        tokens.push(token)
        token = words[i]
    }
}
tokens.push(token)

// insert into wp_embeddings for each token
for (let i = 0; i < tokens.length; i++) {
    // check if tokens[i] is not empty
    if (tokens[i].length == 0) {
        row = res.fetchone()
        continue
    }
    if (tokens[i].trim().length > 0) {
        let q = "insert into wp_embeddings values (0, sys.ML_EMBED_ROW(\""+
            tokens[i] +"\", {'model_id': \"minilm\", \"truncate\": false}), " + id + ")"
        let s2 = session.sql(q)
        s2.execute()
    }
    row = res.fetchone()
}
return "Done"
$$;
```

Running the Embeddings

```
MySQL > call wp_create_embb_js;
+-----+
| wp_create_embb_js |
+-----+
| Embeddings done! 607 posts processed |
+-----+
1 row in set (1 hour 12 min 52.9051 sec)
```

Running the Embeddings

```
MySQL > call wp_create_embb_js;
+-----+
| wp_create_embb_js |
+-----+
| Embeddings done! 607 posts processed |
+-----+
1 row in set (1 hour 12 min 52.9051 sec)
```

```
MySQL > select count(*) from wp_embeddings;
+-----+
| count(*) |
+-----+
| 7347 |
+-----+
1 row in set (0.0199 sec)
```

Filtering out - distance

To filter out some records, we need to use the **distance()** function between our request and the content of the database.

We need of course to use de same model:

```
MySQL > SET @embeddOptions = '{"model_id": "minilm"}';  
MySQL > select sys.ML_EMBED_ROW("What are roles in MySQL 8?",  
    @embeddOptions) into @questionEmb;  
Query OK, 1 row affected (0.5705 sec)
```

Filtering out - distance (2)

We can now use the `distance()` function to filter out the records that are not relevant and keep just some of the top ones having the less distance:

```
MySQL > select wp_post_id, post_title, min(distance(@questionEmb, vec, 'COSINE')) as distance
      from wp_embeddings e join wp_posts p where p.ID = e.wp_post_id group by wp_post_id, post_title
      order by distance limit 3\G
***** 1. row *****
wp_post_id: 2494
post_title: Some queries related to MySQL Roles
distance: 0.10369491577148438
***** 2. row *****
wp_post_id: 1478
post_title: MySQL 8.0: Listing Roles
distance: 0.19010812044143677
***** 3. row *****
wp_post_id: 1402
post_title: MySQL 8.0 Roles and Graphml
distance: 0.19620484113693237
3 rows in set (0.0124 sec)
```

RAG with HeatWave

Now we will create a new JavaScript stored procedure to perform the RAG and generate the answer based on the content of the database. This procedure will include the distance search and the generation of the answer using the LLM model (using the `sys.ML_GENERATE()` function):

wp_ask_js

```
DROP PROCEDURE IF EXISTS wp_ask_js;
CREATE PROCEDURE wp_ask_js(question TEXT)
LANGUAGE JAVASCRIPT
AS $$
    let s = session.sql("SELECT sys.ML_EMBED_ROW(\"" + question + "\",
        '{\"model_id\": \"minilm\", \"truncate\": false}') into @questionEmb")
    let res = s.execute()
    let q = "select wp_post_id from (select wp_post_id, post_title, min(distance(@questionEmb, vec, 'COSINE')) as distance
        from wp_embeddings e join wp_posts p where p.ID = e.wp_post_id group by wp_post_id, post_title
        order by distance limit 5) a where distance < 0.25"
    s = session.sql(q)
    res = s.execute()
    let row = res.fetchone()
    let output = []
```

```
if (output.length == 0) {
    let smt_out = session.sql('Select "No results found" as "Asking WordPress with HeatWave GenAI"',
        {passResultToClient: true})
    smt_out.execute()
    return "No results found"
}
let post_contents = []
for (let i = 0; i < output.length; i++) {
    let wp_post_id = output[i][0]
    let s3 = session.sql("SELECT post_content FROM wp_posts WHERE ID=" + wp_post_id)
    let res3 = s3.execute()
    let row3 = res3.fetchOne()
    let content = row3[0]
    let content_text = content.replace(/<\/?[^>]+(>|$/g, "");
    content_text = content_text.replace(/\r\n/g, " ");
    content_text = content_text.replace(/\n/g, " ");
    // replace all double quotes with \" to escape them
    content_text = content_text.replace(/(?<!\\")/g, '\\\"');
    // replace all single quotes with \
    content_text = content_text.replace(/(?<!\\')/g, '\\\'');
    // replace all lines starting with +- with empty string
    content_text = content_text.replace(/[+-].*[+-]/gm, "")
    // remove all empty lines
    content_text = content_text.replace(/^\s*[\r\n]/gm, "")
    post_contents.push(content_text)
}
```

```
let context_to_send = post_contents.join(" ")
s = session.sql("SET @inferenceSetup = JSON_OBJECT('task', 'generation', 'model_id', 'mistral-7b-instruct-v1',
               'context', \\""+ context_to_send +"\");")
s.execute()
q = "Select sys.ML_GENERATE(\"" + question + "\", @inferenceSetup)"
s = session.sql(q)
res = s.execute()
row = res.fetchone()
let answer = row[0]["text"]
let smt_out = session.sql('Select "' + answer + '" as "Asking WordPress with HeatWave GenAI"', {passResultToClient: true})
smt_out.execute()

return "Done"

$$;
```

RAG with HeatWave (2) - models

In the previous procedure, we used `mistral-7b-instruct-v1`, but we could have used any of the LLM language provided by HeatWave:

```
MySQL> SELECT * FROM sys.ML_SUPPORTED_LLMS
      WHERE model_type = "generation";
+-----+-----+
| model_name          | model_type |
+-----+-----+
| llama2-7b-v1        | generation |
| mistral-7b-instruct-v1 | generation |
| llama3-8b-instruct-v1 | generation |
| cohere.command-text-v14 | generation |
| cohere.command-light-text-v14 | generation |
| cohere.command-r-v1:0   | generation |
| cohere.command-r-plus-v1:0 | generation |
```

```
| meta.llama3-70b-instruct-v1:0    | generation |
| meta.llama3-1-70b-instruct-v1:0  | generation |
| meta.llama3-1-405b-instruct-v1:0 | generation |
| meta.llama-3.1-405b-instruct     | generation |
| meta.llama-3.2-90b-vision-instruct | generation |
| meta.llama-3.3-70b-instruct      | generation |
| cohere.command-r-08-2024         | generation |
| cohere.command-r-plus-08-2024    | generation |
+-----+-----+
15 rows in set (0.0010 sec)
```

RAG with HeatWave (3) - run

We can now run the RAG procedure with a simple question:

```
SQL> call wp_ask_js("How can I split reads and writes with MySQL Router?")\G  
***** 1. row *****
```

Asking WordPress `with` HeatWave GenAI: Answer: MySQL Router can be used `to` split `reads` and `writes` in a WordPress application `by` configuring it `to use` Transparent `Read/Write` Splitting. This involves setting up a MySQL Router instance that acts `as` a proxy between the WordPress application server and the MySQL HeatWave `Database` Service, `with` the WordPress application server configured `to connect` to the MySQL Router instance instead `of` the `primary database` instance directly. The MySQL Router instance can `then` automatically direct `read` and `write` queries `to` the appropriate instance based `on` the configuration settings. This allows `for` automatic `load` balancing and fault tolerance in the WordPress application, without requiring `any` modifications or configuration changes `within` WordPress itself.

```
1 row in set (11.4867 sec)
```

RAG with HeatWave (4) - run

To demonstrate that we are indeed using our content, let's pose a question about a topic not covered in my blog:

```
MSQL> call wp_ask_js("Who won the last World Cup ?")\G
***** 1. row *****
Asking WordPress with HeatWave GenAI: No results found
1 row in set (0.6156 sec)
```

GenAI with MySQL HeatWave - Summary

When using **MySQL HeatWave**, we can use the GenAI features directly in the database without the need to use any external service or API.

We can directly use internal functions such as `sys.ML_EMBED_ROW()` to generate embeddings and `sys.ML_GENERATE()` to generate answers based on the content of the database.

But we can also put everything together in a stored procedure written in JavaScript that will make it even easier to use.

Wait! 

... and locally ?



GenAI on my machine

Will it be possible to run something similar on my machine?

GenAI on my machine

Will it be possible to run something similar on my machine?

Yes! But this will require some work and a lot of patience...

GenAI on my machine

Will it be possible to run something similar on my machine?

Yes! But this will requires some work and a lot of patience...

You need:

- *a LLM inference server*
- *LLM models*
- *a MySQL server supporting the VECTOR datatype (>= 9)*
- *being able to perform a COSINE similarity search*

LLM Inference Server



I use `llama.cpp` to run the LLM models on my machine:

<https://github.com/ggerganov/llama.cpp>

LLM Models

The models are available on huggingface.

I used the following models:

```
$ ls -lh
total 6.2G
-rw-r--r-- 1 fred fred 419M Jan 22 09:35 bge-reranker-v2-m3-Q4_K_M.gguf
-rw-r--r-- 1 fred fred 140M Jan 23 11:52 jina-embeddings-v2-base-en-q8_0.gguf
-rw-r--r-- 1 fred fred 5.7G Jan 22 09:56 meta-llama-3.1-8b-instruct-q4_0.gguf
```



Hugging Face

Models

Models names represent the type of model and the version of the model.

It provides:

- *model name or family (e.g., meta)*
- *model architecture and version (e.g., llama-3.1)*
- *model size (e.g., 8b or 8 billion parameters)*
- *instruction-tuned variant (e.g., instruct)*
- *quantization or compression format (e.g., q8_0, Q4_K_M)*
- *file format (e.g., gguf)*

Models (2)

The model `meta-llama-3.1-8b-instruct-q4_0.gguf` is:

Models (2)

The model `meta-llama-3.1-8b-instruct-q4_0.gguf` is:

- *Created by Meta as part of the LLaMA 3.1 series.*

Models (2)

The model `meta-llama-3.1-8b-instruct-q4_0.gguf` is:

- *Created by Meta as part of the LLaMA 3.1 series.*
- *A mid-sized model with 8 billion parameters.*

Models (2)

The model `meta-llama-3.1-8b-instruct-q4_0.gguf` is:

- *Created by Meta as part of the LLaMA 3.1 series.*
- *A mid-sized model with 8 billion parameters.*
- *Fine-tuned for instruction-following tasks, making it user-friendly for conversational or task-specific applications.*

Models (2)

The model `meta-llama-3.1-8b-instruct-q4_0.gguf` is:

- *Created by Meta as part of the LLaMA 3.1 series.*
- *A mid-sized model with 8 billion parameters.*
- *Fine-tuned for instruction-following tasks, making it user-friendly for conversational or task-specific applications.*
- *Optimized using 4-bit quantization for more efficient deployment on lower-resource hardware.*

Models (2)

The model `meta-llama-3.1-8b-instruct-q4_0.gguf` is:

- *Created by Meta as part of the LLaMA 3.1 series.*
- *A mid-sized model with 8 billion parameters.*
- *Fine-tuned for instruction-following tasks, making it user-friendly for conversational or task-specific applications.*
- *Optimized using 4-bit quantization for more efficient deployment on lower-resource hardware.*
- *Packaged in the .gguf format for easy integration with specific tools or environments.*

Embeddings Service

To serve the embeddings, I use `llama.cpp` server with a text embeddings model by Jina AI like this:

```
$ llama-server -m MODELS/jina-embeddings-v2-base-en-q8_0.gguf \
--embedding --pooling cls -ub 8192 --no-webui
```

This will serve the embeddings on `http://localhost:8080`.



Reranking Service

For reranking, I use `llama.cpp` server with a reranker model from the Beijing Academy of Artificial Intelligence like this:

```
$ llama-server -m MODELS/bge-reranker-v2-m3-Q4_K_M.gguf \
  -b 81920 -ub 81920 -c 16384 --reranking --port 8082 --no-webui
```

This will serve the reranking on `http://localhost:8082`.



Inference Service

And finally, the inference service with the LLM model from Meta-Llama is served like this:

```
$ llama-server -m MODELS/meta-llama-3.1-8b-instruct-q4_0.gguf --port 8081
```

This will serve the inference on <http://localhost:8081> and can also be accessed via a browser.

LLaMA
by  **Meta**

Inference Service - Webui

Conversations

+ New conversation

What is MySQL?

llama.cpp



What is MySQL?

MySQL is a popular open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage and manipulate data. It's a widely-used database system that allows you to store, manage, and retrieve data in a structured and organized way.

Here are some key features of MySQL:

1. **Open-source:** MySQL is free and open-source, which means that it's maintained by a community of developers and is available for anyone to use, modify, and distribute.
2. **Relational database:** MySQL uses a relational database model, which means that it stores data in a structured and organized way, with each piece of data related to other data through tables, columns, and relationships.
3. **SQL support:** MySQL supports standard SQL commands, making it easy to interact with the database and perform operations such as querying, inserting, updating, and deleting data.
4. **Scalability:** MySQL is designed to handle large amounts of data and can scale to meet the needs of small to very large applications.
5. **Cross-platform:** MySQL can run on multiple platforms, including Windows, Linux, and macOS.

Type a message (Shift+Enter to add a new line)



Send

Conversations are saved to
browser's localStorage

MySQL Similarity Search

In *MySQL Community*, there is not yet any support for vector indexes or similarity search functions.

But a COSINE distance function is relatively easy to implement as UDF:

1. Compute the dot product $\mathbf{A} \cdot \mathbf{B}: \sum_{i=1}^n A_i B_i$.

2. Compute the magnitudes of \mathbf{A} and \mathbf{B} :

- $\|\mathbf{A}\| = \sqrt{\sum_{i=1}^n A_i^2}$
- $\|\mathbf{B}\| = \sqrt{\sum_{i=1}^n B_i^2}$

3. Calculate cosine similarity:

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

4. Subtract cosine similarity from 1 to obtain cosine distance:

$$\text{Cosine Distance} = 1 - \text{Cosine Similarity}$$

MySQL Similarity Search (2)

This can be translated to something like this in C++:

```
for (size_t i = 0; i < vector1.size(); ++i) {
    sum += vector1[i] * vector2[i];
    magnitude_vec1 += vector1[i] * vector1[i];
    magnitude_vec2 += vector2[i] * vector2[i];
}

if (magnitude_vec1 == 0.0 || magnitude_vec2 == 0.0) {
    // Error with zero vector
    printf("Vectors must not be zero vectors.\n");
    return 0;
}

double cosine_similarity = sum / (sqrt(magnitude_vec1) * sqrt(magnitude_vec2));
return 1.0 - cosine_similarity;
```

MySQL Similarity Search (3)

```
SQL > SELECT UDF_NAME FROM performance_schema.user_defined_functions  
      WHERE UDF_NAME LIKE 'distance%';  
+-----+  
| UDF_NAME |  
+-----+  
| distance_cosine |  
+-----+  
1 row in set (0.0104 sec)
```

But of course such function will perform a full table scan.

It's also possible to create a JavaScript function to perform the same operation if you use MySQL Enterprise Edition.

MySQL Similarity Search (4)

```

MySQL > 127.0.0.1:3306 wp_lefred 2025-01-23 10:16:11
SQL > select id, wp_post_id, distance cosine(vec, @a) distance from wp_embeddings order by distance limit 5;
+-----+-----+
| id   | wp_post_id | distance          |
+-----+-----+
| 3245 |      2271 |      0 |
| 8093 |      5395 | 0.05191023275256157 |
| 7541 |      5072 | 0.05191023275256157 |
| 5288 |      3687 | 0.05191023275256157 |
| 6132 |      4323 | 0.05191023275256157 |
+-----+-----+
5 rows in set (0.0964 sec)

MySQL > 127.0.0.1:3306 wp_lefred 2025-01-23 10:16:58
SQL > explain select id, wp_post_id, distance_cosine(vec, @a) distance from wp_embeddings order by distance limit 5\G
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: wp_embeddings
    partitions: NULL
        type: ALL
possible_keys: NULL
        key: NULL
    key_len: NULL
        ref: NULL
        rows: 9199
    filtered: 100
        Extra: Using temporary; Using filesort
1 row in set, 1 warning (0.0115 sec)
Note (code 1003): /* select#1 */ select `wp_lefred`.`wp_embeddings`.`id` AS `id`, `wp_lefred`.`wp_embeddings`.`wp_post_id` AS `wp_
tance_cosine(`wp_lefred`.`wp_embeddings`.`vec` AS `vec`,(@`a`)) AS `distance` from `wp_lefred`.`wp_embeddings` order by
mit 5

MySQL > 127.0.0.1:3306 wp_lefred 2025-01-23 10:17:11
SQL > alter table wp_embeddings add index vec_idx(vec);
ERROR: 6134 (HY000): Non-scalar (e.g., vector) column 'vec' cannot be used as key.

```

Testing Embeddings

We changed a bit the previous code to use the local services APIs:

```
LLAMA_EMBEDDINGS_URL = "http://localhost:8080/v1/embeddings"
LLAMA_COMPLETIONS_URL = "http://localhost:8081/v1/completions"
LLAMA_RERANK_URL = "http://localhost:8082/v1/rerank"
```

```
[fred@dynabook ~workspace/local_genai/local-genai-wp] $ python3.12 wp_embeddings.py
```

```
You are now connected to MySQL Community Server - GPL 9.2.0
Starting Embeddings....
```

```
.....Error with wp_post_id: 7063 !
```

```
Done with Embeddings.
```

RAG Test

```
[fred@dynabook ~/workspace/local_genai/local-genai-wp] $ python3.12 wp_rerank_genai.py
```

You are now connected to MySQL Community Server - GPL 9.2.0

What is your question? What are the MySQL Belgian Days and when happened them and when are the next ones?

Performing Embeddings of the prompt...

Performing Vector Search Similarity...

Performing Reranking...

Generating the Answer...

I will use the following blog posts to generate my answer:

- [7217]: MySQL Belgian Days 2024 – The agenda
- [7734]: preFOSDEM MySQL Belgian Days 2025 – Agenda
- [7649]: preFOSDEM MySQL Belgian Days 2025 – save the dates
- [7265]: Unveiling the Highlights: A Look Back at MySQL Belgian Days 2024
- [6215]: preFOSDEM MySQL Days 2023 – the Agenda

The MySQL Belgian Days are a conference that takes place in Brussels, Belgium, just before FOSDEM. The event is organized by the MySQL Community Team and features sessions from the MySQL Community and the MySQL Team from Oracle. The conference is free and in-person, with limited seating. The next MySQL Belgian Days will take place on January 30th and 31st, 2025.

The first MySQL Belgian Days were held on February 1st and 2nd, 2024, and the second ones were held on January 30th and 31st, 2025. The conference is a unique opportunity to meet MySQL developers, engineers, and professionals, and to learn about the latest developments in the MySQL ecosystem



Cool isn't it?



And what about DeepSeek?

`llama.cpp` is also compatible with DeepSeek:

```
$ ls -lh *gguf  
-rw-r--r-- 1 fred fred 4.6G Jan 28 10:29 DeepSeek-R1-Distill-Llama-8B-Q4_K_M.gguf
```



Disclaimer: Due to suspicions of potential issues with this model in certain countries, this test was conducted on a non-Oracle machine and within a VM for research purposes.

`llama.cpp` is also compatible with DeepSeek:

```
$ ls -lh *gguf
-rw-r--r-- 1 fred fred 4.6G Jan 28 10:29 DeepSeek-R1-Distill-Llama-8B-Q4_K_M.gguf
```



DeepSeek

We use a similar approach to the one we used for the other models, with just some minor changes in the code generating the prompt.

Running the server:

```
$ llama-server -m MODELS/DeepSeek-R1-Distill-Llama-8B-Q4_K_M.gguf --port 8081 \
--cache-type q8_0 --threads 16
```

DeepSeek - Example

```
[fred@dynabook ~/workspace/local_genai/local-genai-wp] $ python wp_rerank_genai_deepseek.py
You are now connected to MySQL Community Server - GPL 9.2.0
What is your question? What are the MySQL Belgian Days and when happenend them and when are the next ones?
Performing Embeddings of the prompt...
Performing Vector Search Similarity...
Performing Reranking...
DEBUG: Sorted Indices: [0, 2, 1, 7, 8, 3, 4, 6, 5, 9]
Generating the Answer...
I will use the following blog posts to generate my answer:
- [7217]: MySQL Belgian Days 2024 - The agenda
- [7734]: preFOSDEM MySQL Belgian Days 2025 - Agenda
- [7649]: preFOSDEM MySQL Belgian Days 2025 - save the dates
- [7265]: Unveiling the Highlights: A Look Back at MySQL Belgian Days 2024
- [6215]: preFOSDEM MySQL Days 2023 - the Agenda
```

The MySQL Belgian Days are an annual event organized by the MySQL Community Team, typically held in Brussels, Belgium, prior to FOSDEM. They usually take place on the last days of January or early February. The next MySQL Belgian Days are scheduled to occur on January 30th and 31st, 2025.

Local GenAI RAG

The code is available on my GitHub:

<https://github.com/lefred/local-genai-wp>



MySQL AI - released on Sep 11, 2025

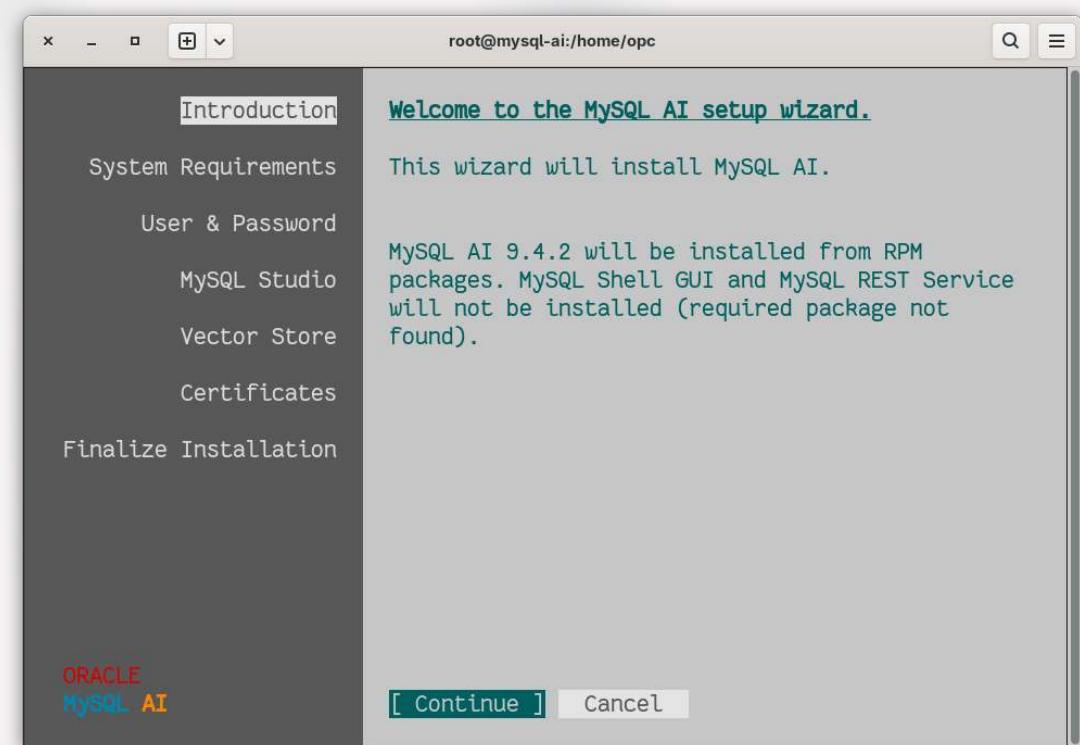
MySQL AI consists of the following components:

- *MySQL Enterprise Edition* for *MySQL AI*, which includes:
 - MySQL Commercial Server
 - AI Engine
 - MySQL AI Plugin
- *MySQL Shell AI Edition*, which supports MySQL Shell Workbench.
- *MySQL Router AI Edition*, which supports the MySQL REST Service

MySQL AI - Installation

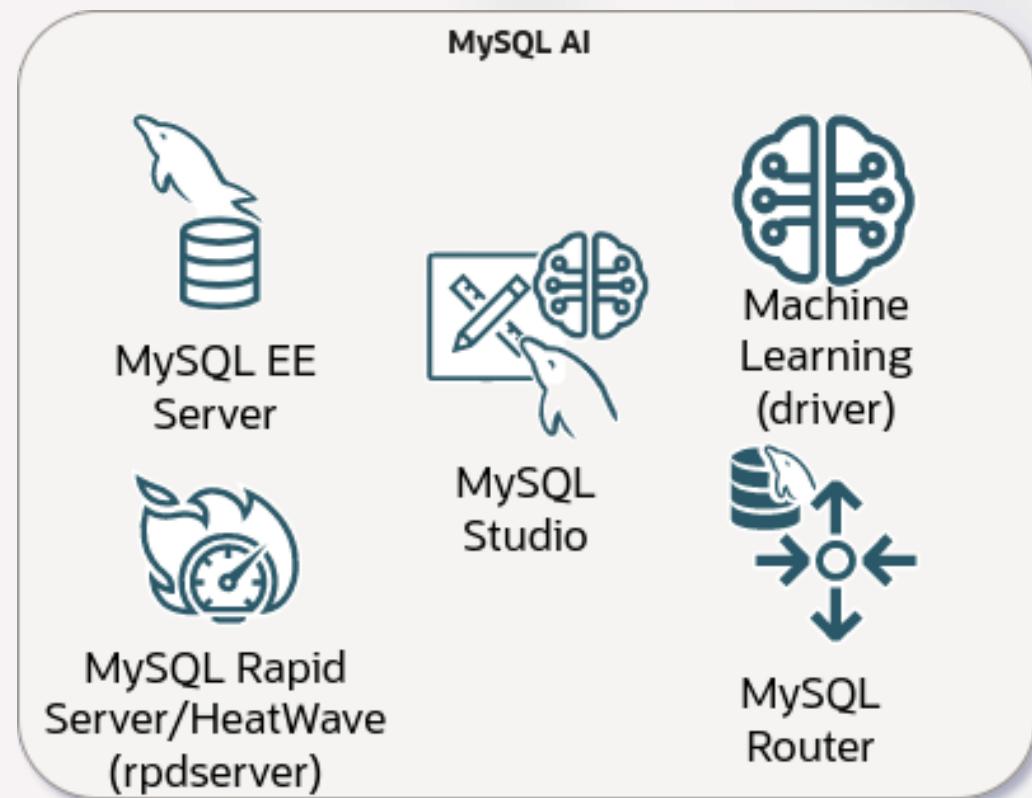
The installation is made using RPMs but you install only the setup package: `mysql-ai-setup-9.xxxx.x86_64.rpm`.

After that you run the `mysql-ai-setup` command that will install all the required packages and configure the different components:



MySQL AI - Architecture

MySQL AI is composed of several components that work together to provide AI capabilities:



MySQL AI - Models

The current models included in MySQL AI are:

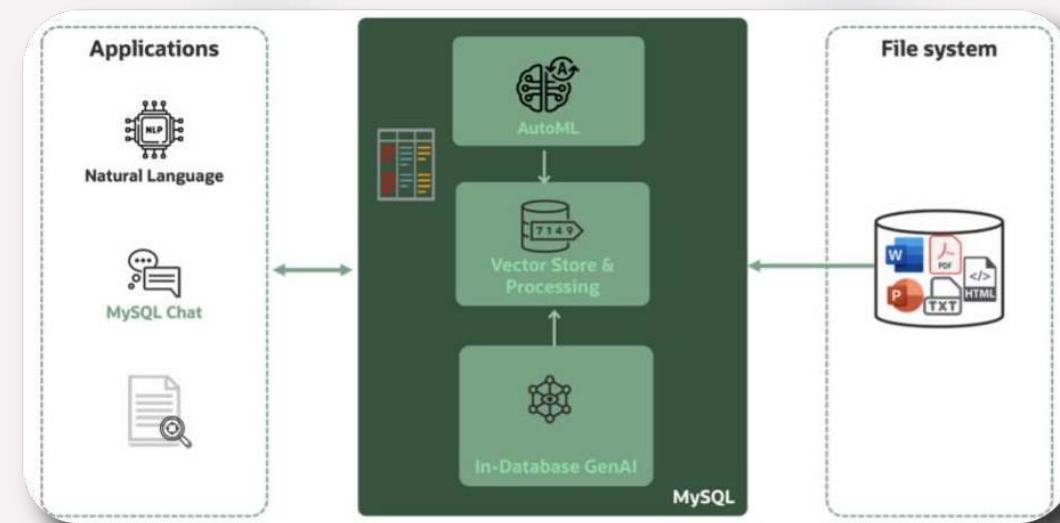
```
SQL > SELECT model_id, availability_date, capabilities FROM sys.ML_SUPPORTED_LLMS;
+-----+-----+-----+
| model_id          | availability_date | capabilities      |
+-----+-----+-----+
| llama3.2-3b-instruct-v1 | 2025-05-20        | ["GENERATION"]    |
| all_minilm_l12_v2     | 2024-07-01        | ["TEXT_EMBEDDINGS"]|
| multilingual-e5-small | 2024-07-24        | ["TEXT_EMBEDDINGS"]|
+-----+-----+-----+
3 rows in set (0.1054 sec)
```

MySQL AI - Blog's content to GenAI Chat

MySQL AI includes a **Vector Store** to store the embeddings.

We use the same approach as before to create the embeddings and store them in a new table.

The embeddings is included in **MySQL Studio** when performed from unstructured documents like PDF, Word, Text files...



MySQL AI - Blog's content to GenAI Chat (2)

For our blog, we need to create the embeddings ourselves using `sys.ML_EMBED_ROW()` that we encapsulate in JS procedures.

And we keep the same table structure as used by [MySQL AI](#).

```
SQL > show create table myvec\G
***** 1. row *****
Table: myvec
Create Table: CREATE TABLE `myvec` (
  `document_name` varchar(1024) NOT NULL COMMENT 'RAPID_COLUMN=ENCODING=VARLEN',
  `metadata` json NOT NULL COMMENT 'RAPID_COLUMN=ENCODING=VARLEN',
  `document_id` int unsigned NOT NULL,
  `segment_number` int unsigned NOT NULL,
  `segment` varchar(1024) NOT NULL COMMENT 'RAPID_COLUMN=ENCODING=VARLEN',
  `segment_embedding` vector(384) NOT NULL COMMENT 'RAPID_COLUMN=ENCODING=VARLEN
    GENAI_OPTIONS=EMBED_MODEL_ID=multilingual-e5-small',
  PRIMARY KEY (`document_id`, `segment_number`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
SECONDARY_ENGINE=RAPID
```

MySQL AI - Blog's content to GenAI Chat (3)

wp_create_embb_js

```
DROP PROCEDURE IF EXISTS wp_create_embb_js;
CREATE PROCEDURE wp_create_embb_js()
LANGUAGE JAVASCRIPT
AS $$
let s = session.sql("SELECT count(*) FROM wp_posts where post_type='post' and post_status='publish'");
let res = s.execute();
let row = res.fetchOne();
let total = row[0];
let batch = 10;
let processed = 0;
let last_id = 0;
while (processed < total) {
    s = session.sql("SELECT ID FROM wp_posts where post_type='post'
        and post_status='publish' and ID > " + last_id
        +" ORDER BY ID LIMIT " + batch)
    res = s.execute()
    row = res.fetchOne()
    while(row) {
        let id = row[0]
        last_id = id
        let s2 = session.sql("CALL wp_create_row_embb_js(" + id + ")")
        s2.execute()
        processed += 1
        row = res.fetchOne()
    }
}
let stmt_out = session.sql('Select "Embeddings done! ' + total +
    ' posts processed" as "wp_create_embb_js"', {passResultToClient: true})
stmt_out.execute()
$$;
```

MySQL AI - Blog's content to GenAI Chat (4)

wp_create_row_embb_js

```
DROP PROCEDURE IF EXISTS wp_create_row_embb_js;
CREATE PROCEDURE wp_create_row_embb_js(id INT)
LANGUAGE JAVASCRIPT
AS $$
let s = session.sql("SELECT post_content, post_title FROM wp_posts WHERE ID=" + id)
let res = s.execute()
let row = res.fetchOne()

while(row) {
    let content = row[0]
    let title = row[1]

    let title_text = title.replace(/<\/?[^>]+(>|$/g, " ");
    title_text = title_text.replace(/\r\n/g, " ");
    title_text = title_text.replace(/\n/g, " ");
    // replace all double quotes with \" to escape them
    title_text = title_text.replace(/(?<!\\")/g, '\\\"');
    // replace all single quotes with \
    title_text = title_text.replace(/(?<!\\')/g, '\\\'');
    // replace all lines starting with +- with empty string
    title_text = title_text.replace(/[+-].*[+-]/gm, "");
    // remove all empty lines
    title_text = title_text.replace(/^\s*[\r\n]/gm, "")
}
```

```
let content_text = content.replace(/<\/?[^>]+(>|$/g, "");
content_text = content_text.replace(/\r\n/g, " ");
content_text = content_text.replace(/\n/g, " ");
// replace all double quotes with \" to escape them
content_text = content_text.replace(/(?<!\\")/g, '\\"');
// replace all single quotes with \
content_text = content_text.replace(/(?<!\\')/g, "\\'");
// replace all lines starting with +- with empty string
content_text = content_text.replace(/[+-].*[+-]/gm, "")
// remove all empty lines
content_text = content_text.replace(/^\s*/[\r\n]/gm, "")
// check if content_text is empty
if (content_text.length == 0) {
    return "Content is empty"
}

// split content_text into strings of 100 characters
// but not breaking words
let tokens = []
let token = ""
let words = content_text.split(" ")
for (let i = 0; i < words.length; i++) {
    if (token.length + words[i].length < 100) {
        token += " " + words[i]
    } else {
        tokens.push(token)
        token = words[i]
    }
}
tokens.push(token)
```

```
// insert into wp_embeddings for each token
for (let i = 0; i < tokens.length; i++) {
    // check if tokeks[i] is not empty
    if (tokens[i].length == 0) {
        row = res.fetchOne()
        continue
    }

    if (tokens[i].trim().length > 0) {
        let seg = i + 1 // because 0 is for the title we already added manually
        let q = "insert into lefred_vector.myvec values ('" + title_text
        + "\", CAST('{" + "parser": "oit_jx+ocr", "text_from_images": "false"}' AS JSON), "
        + id + ", " + seg + ",\"" + tokens[i] + "\",sys.ML_EMBED_ROW('" + tokens[i]
        +"", '{"model_id": "multilingual-e5-small", "truncate": false}'))"
        console.log(q)
        let s2 = session.sql(q)
        s2.execute()
    }
}
row = res.fetchOne()
}

return "Done"

$$;
```

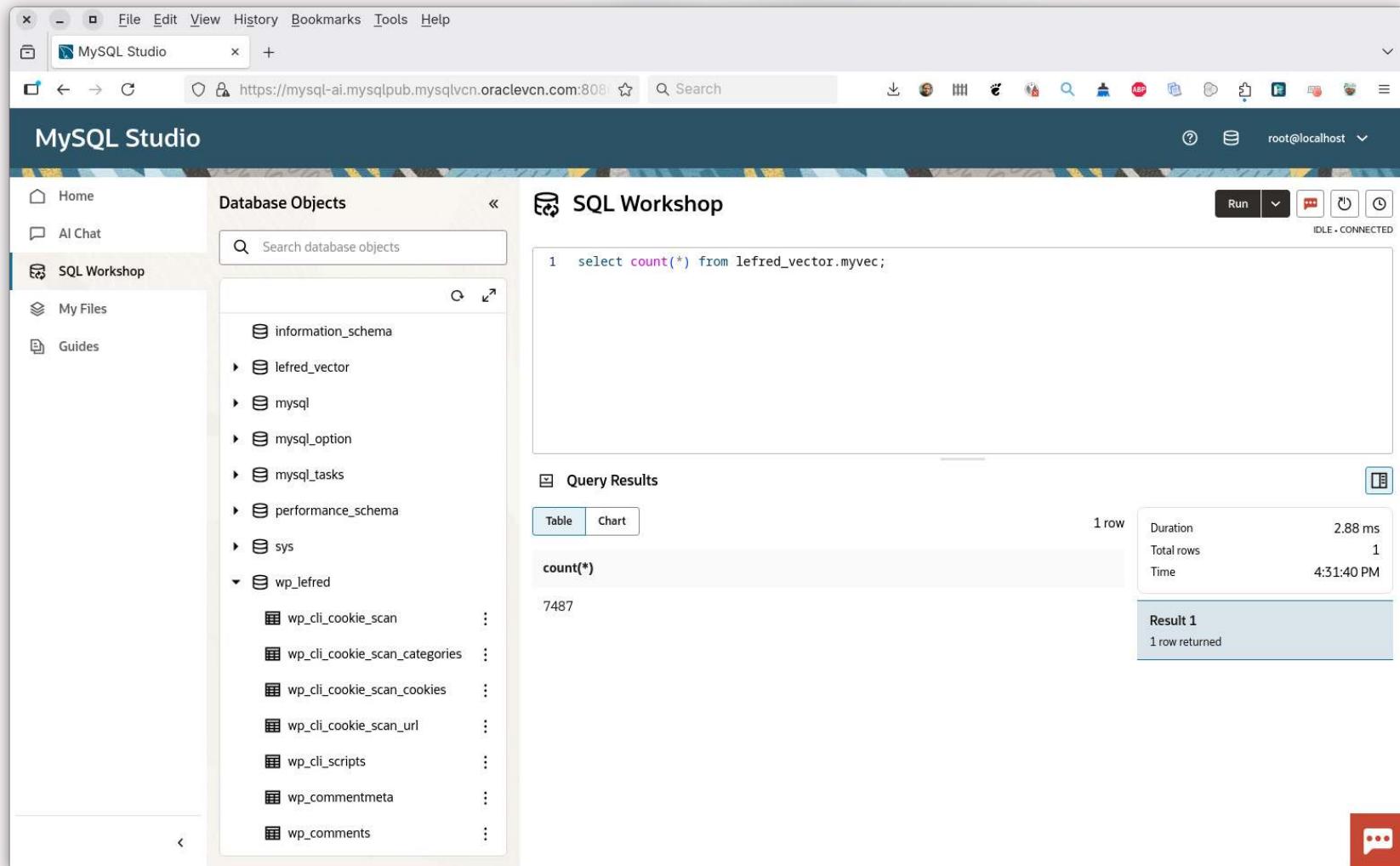
MySQL AI - Blog's content to GenAI Chat (5)

The screenshot shows the MySQL Studio interface. The left sidebar has 'Home' and 'AI Chat' buttons, and the 'SQL Workshop' section is selected, showing 'My Files' and 'Guides'. The main area is titled 'SQL Workshop' and displays a query in the 'Query Editor' tab:

```
6 AS $$ BEGIN
65   SELECT COUNT(*) AS count
66   $$;
67   $$;
68
69
70   select mle_session_reset("stdout");
71
72   call wp_create_embb_js;
73
74   select mle_session_state("stdout");
75
```

The 'Query Results' tab is currently empty. A progress indicator shows 'Elapsed time: 14.1 s'.

MySQL AI - Blog's content to GenAI Chat (6)



MySQL AI - Blog's content to GenAI Chat (7)

The screenshot shows the MySQL Studio interface with the "AI Chat" feature integrated. The left sidebar includes "Home", "AI Chat" (which is selected), "SQL Workshop", "My Files", and "Guides". The main area has tabs for "Vector Stores" and "AI Chat". The "Vector Stores" tab shows a list with "lefred_vector" expanded, revealing "myvec". The "AI Chat" tab displays a conversation about MySQL roles. A sidebar titled "Sources" lists "Some queries related to MySQL Roles", "MySQL 8.0: Listing Roles", and "MySQL: who's filling my error log?". A bottom section allows users to "Ask a question" and provides a note about responses referencing selected data sources.

MySQL Studio

File Edit View History Bookmarks Tools Help

MySQL Studio https://mysql-ai.mysqlpub.mysqlvcn.oraclevcn.com:8080 Search

root@localhost

MySQL Studio

Home AI Chat SQL Workshop My Files Guides

Vector Stores

Search names

lefred_vector myvec

AI Chat

What are roles?

As a database assistant for MySQL, I'd be happy to help.

In MySQL, a role is a way to define a set of privileges that can be assigned to a user or group of users. Roles allow you to manage access control and ensure that users only have the necessary permissions to perform specific tasks on your database.

Here are some key aspects of roles in MySQL:

1. **Role creation:** You can create multiple roles in MySQL using the CREATE ROLE statement.
2. **Privileges:** Each role can be assigned a set of privileges, which define what actions a user with that role can perform on the database (e.g., SELECT, INSERT, UPDATE, DELETE).
3. **User assignment:** Roles can be assigned to users or groups of users using the GRANT statement.
4. **Role inheritance:** MySQL supports role inheritance, which allows a parent role to inherit privileges from child roles.

Some common types of roles in MySQL include:

1. **Database administrator (DBA) role:** This role typically has full control over the database and is used by system administrators.
2. **User role:** A user role is assigned to individual users or groups of users, granting them access to specific databases or tables.
3. **Public role:** The public role is a special role that grants all privileges

Sources

Some queries related to MySQL Roles

MySQL 8.0: Listing Roles

MySQL: who's filling my error log?

Ask a question

Responses will reference selected data sources. MySQL AI can make mistakes, please double-check responses.

MySQL AI - Blog's content to GenAI Chat (8)

Of course we can still use the SQL client:

```
SQL > select sys.ML_EMBED_ROW("What are roles in MySQL 8?", @embeddOptions) into @questionEmb;
Query OK, 1 row affected (0.1273 sec)

SQL > select document_name, document_id, segment_number, distance(@questionEmb, segment_embedding, 'COSINE') as distance
      from myvec order by distance limit 10;
+-----+-----+-----+-----+
| document_name          | document_id | segment_number | distance        |
+-----+-----+-----+-----+
| Some queries related to MySQL Roles | 2494       | 10            | 0.04005849361419678 |
| MySQL 8.0: Listing Roles           | 1478       | 0             | 0.0646367073059082 |
| MySQL 8.0 Roles and Graphml        | 1402       | 0             | 0.06661677360534668 |
| TOP 10 MySQL 8.0 features for DBAs & OPS | 1378       | 45            | 0.08692586421966553 |
| MySQL 8.0: Listing Roles           | 1478       | 1             | 0.0871921181678772 |
| MySQL 8.0 Roles and Graphml        | 1402       | 1             | 0.0899040699005127 |
| Some queries related to MySQL Roles | 2494       | 0             | 0.0903623104095459 |
| Some queries related to MySQL Roles | 2494       | 2             | 0.09287041425704956 |
| Some queries related to MySQL Roles | 2494       | 4             | 0.10311764478683472 |
| Build MySQL 8 from the source rpm in OL8 | 6811       | 0             | 0.10328680276870728 |
+-----+-----+-----+-----+
10 rows in set (0.0061 sec)
```

MySQL AI

The code is available on my GitHub:

<https://github.com/lefred/mysql-ai-wp>



Share your ❤️ to MySQL

#mysql #MySQLCommunity



Join our slack channel!

bit.ly/mysql-slack

Questions ?

