

Oracle 23ai 向量数据库

向量数据库动手实验(一)

技术文档

SE Hub

Hysun He

November, 2024

变更记录

日期	作者	版本	变更参考
11/12/2024	贺友胜 (Hysun He)	1.0	初始版本

审核

日期	审核人	版本	变更记录

概要

本实验以熟悉 Oracle 向量数据库的一些实际操作为主要目的，主要内容包括 Oracle 向量数据类型、向量模型、数据向量化（库外向量化与库内向量化两种方式）、向量索引（HNSW 和 IVF）、向量检索（非索引精确检索和索引近似检索）。

为方便拷贝粘贴，使用过程中也可以借助本文档的 Markdown 版本：

https://github.com/HysunHe/23ai_workshop_prep/blob/main/Oracle%E5%90%91%E9%87%8F%E6%95%B0%E6%8D%AE%E5%BA%93_lab1.md

预计时间：1.5 小时

目标

- 了解 Oracle 向量数据类型及基本操作
- 了解向量相似度检索，包括精确检索和近似检索
- 了解常用的向量索引类型
- 了解向量模型的部署以及 Oracle 数据库与外部向量模型的结合
- 了解向量模型的导入以及 Oracle 数据库的库内向量化操作
- 了解库外向量化及库内向量化的优劣现状及前景

前提条件

- 本实验重点在于动手操作，非对向量数据库及 Oracle 向量数据库进行理论上的讲解，因此，需要参与者已经参加过 Oracle 向量数据库的介绍或有所了解。
- 有基本的 PL/SQL 知识，能够看简单的 PL/SQL 示例代码，能够利用客户端（如 sqlplus）等运行提供的 PL/SQL 示例代码。
- 最好有基本的 Python 知识，能够看懂简单的 Python 示例代码（非必需）。

环境准备

本文中涉及的账号密码，请参考《动手试验环境说明.pdf》文档 或 咨询现场技术人员。

目录

1 Oracle 向量基本操作	1
1.1 字符串转换为向量	1
1.2 向量转换为字符串	1
1.3 向量间距离计算	1
1.3.1 利用欧氏距离(L2)策略计算两个向量之间的距离	1
1.3.2 利用余弦距离策略计算两个向量之间的距离	1
1.4 向量类型字段及样列表	2
1.5 样例数据	2
2 向量检索	3
2.1 向量精确检索	3
2.2 向量近似检索	4
2.2.1 向量内存池	4
2.2.2 向量内存池视图	5
2.2.3 创建 HNSW 索引	6
2.2.4 HNSW 近似检索	7
2.2.5 创建 IVF 索引	7
2.2.6 IVF 近似检索	8
3 部署向量嵌入模型（仅讲师操作）	10
3.1 向量嵌入模型部署	10
3.2 向量嵌入模型访问	10
3.3 库外向量化操作	11
3.3.1 数据加载	11
3.3.2 向量检索	13

4 库内向量化操作.....	16
4.1 导入向量嵌入模型.....	16
4.2 库内向量化及检索.....	18
4.2.1 准备数据.....	18
4.2.2 库内向量化.....	19
4.2.3 相似度检索.....	20
5 与第三方向量嵌入模型服务集成（仅演示）.....	21
5.1 开通第三方 API 服务.....	21
5.2 创建访问凭证.....	21
5.3 直接在 SQL 中调用外部 Embedding 服务.....	22
6 总结.....	24

1 Oracle 向量基本操作

1.1 字符串转换为向量

TO_VECTOR()用来将字符串类型的数字数组转换为向量类型。

```
SELECT TO_VECTOR( '[3,3]' );
```

1.2 向量转换为字符串

FROM_VECTOR()用来将向量类型转换为字符串类型。

```
SELECT FROM_VECTOR( TO_VECTOR( '[3,3]' ) );
```

1.3 向量间距离计算

VECTOR_DISTANCE(v1, v2, 距离策略) 是向量检索的关键操作，用来比较两个向量的距离（相似度）。距离越大，说明相似度越小；反之，说明两个向量越相似。 Oracle 支持的距离策略主要有：EUCLIDEAN, COSINE, DOT, HAMMING

1.3.1 利用欧氏距离(L2)策略计算两个向量之间的距离

```
SELECT VECTOR_DISTANCE( vector('[2,2]'), vector('[5,6]'), EUCLIDEAN ) as distance;
```

注：欧几里得距离是指连接这两点的线段的长度（二维空间中），上述 [2,2] 和 [5,6] 两点间的距离由勾股定理可直接算出为 5

1.3.2 利用余弦距离策略计算两个向量之间的距离

```
SELECT VECTOR_DISTANCE( vector('[2,2]'), vector('[5,5]'), COSINE) as
distance;
```

注：余弦距离策略关注的是两个向量在方向上的一致性，上述 [2,2] 和 [5,5] 在方向上完全一致，因此，它们的距离为 0，代表两个向量完全匹配。

1.4 向量类型字段及样例表

Oracle 23ai 引入了向量数据类型：VECTOR (dimentions, format)，该类型可指定两个参数，第一个是向量的维度，如 [2,2] 是一个二维向量；第二个是数据格式，如 FLOAT32。也可以不指定。

建立一个测试表 **galaxies**:

```
create table galaxies (
  id number,
  name varchar2(50),
  doc varchar2(500),
  embedding VECTOR
);
```

1.5 样例数据

向 **galaxies** 表中插入如下样例数据：

```
insert into galaxies values (1, 'M31', 'Messier 31 is a barred spiral galaxy
in the Andromeda constellation.', '[0,1,1,0,0]');
insert into galaxies values (2, 'M33', 'Messier 33 is a spiral galaxy in the
Triangulum constellation.', '[0,0,1,0,0]');
insert into galaxies values (3, 'M58', 'Messier 58 is an intermediate barred
spiral galaxy in the Virgo constellation.', '[1,1,1,0,0]');
insert into galaxies values (4, 'M63', 'Messier 63 is a spiral galaxy in the
Canes Venatici constellation.', '[0,0,1,0,0]');
insert into galaxies values (5, 'M77', 'Messier 77 is a barred spiral galaxy
in the Cetus constellation.', '[0,2,2,0,0]');
insert into galaxies values (6, 'M91', 'Messier 91 is a barred spiral galaxy
in the Coma Berenices constellation.', '[0,3,3,0,0]');
insert into galaxies values (7, 'M49', 'Messier 49 is a giant elliptical
galaxy in the Virgo constellation.', '[0,0,0,1,1]');
insert into galaxies values (8, 'M60', 'Messier 60 is an elliptical galaxy in
the Virgo constellation.', '[0,0,0,0,1]');
insert into galaxies values (9, 'NGC1073', 'NGC 1073 is a barred spiral
galaxy in Cetus constellation.', '[0,3,3,0,0]');
commit;
```

数据准备好后，接下来，我们就可以根据数据进行检索了。

2 向量检索

2.1 向量精确检索

向量精确检索（Exact Search）类似于关系数据查询时的全表扫描，是指库中的每一个向量都与查询向量进行匹配，这样就能计算出每个向量与查询向量之间的相似度，从而精确的返回与查询向量最相似的 N 条记录，不会漏掉任何一条记录（也就是说，召回率始终能达到 100%）。

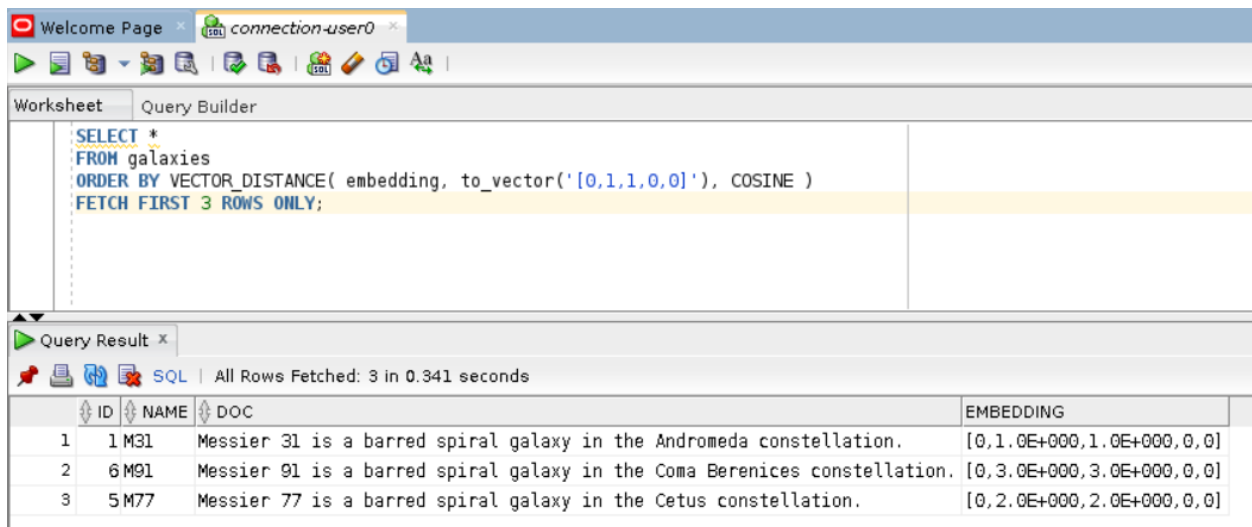
由于结果的准确性，毫无疑问，在需要遍历的向量数据集较小时，精确检索是较优的方式。

在使用如 Oracle 这类融合数据库时，很多情况下，可以使用关系数据的业务属性字段（标量字段）缩小需要进行向量匹配的数据，因此，结合关系数据库特征，可以很大程度上提高向量检索的精确性和性能。

SQL 查询语句：利用余弦策略检索出与向量 $[0,1,1,0,0]$ 最相近的 3 条记录：

```
SELECT *
FROM galaxies
ORDER BY VECTOR_DISTANCE( embedding, to_vector('[0,1,1,0,0]'), COSINE )
FETCH FIRST 3 ROWS ONLY;
```

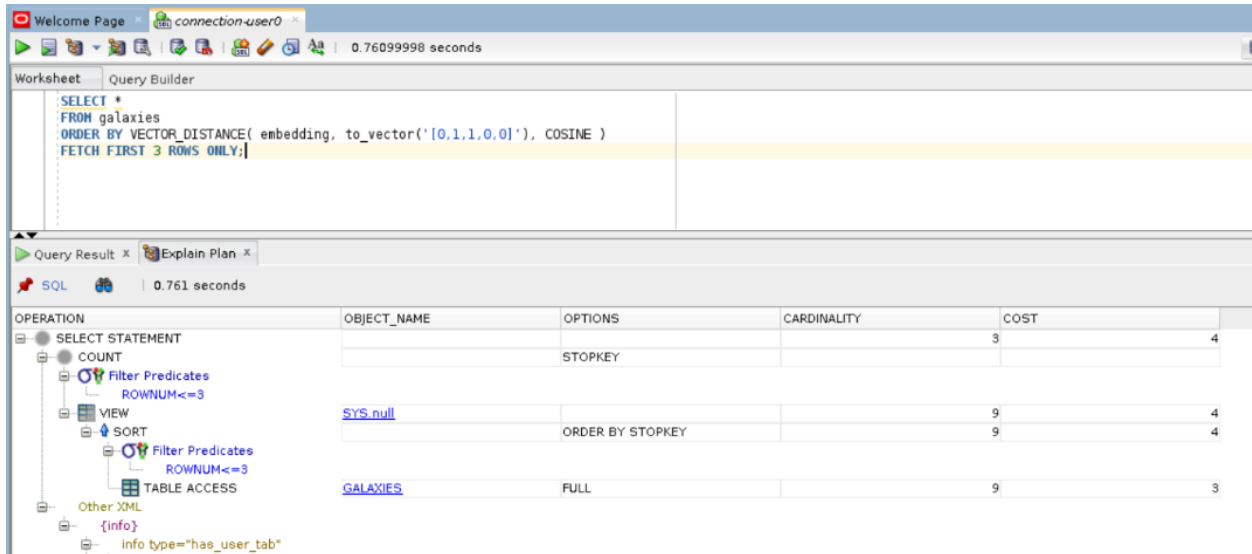
查询结果：



The screenshot shows the Oracle SQL Developer interface. The top toolbar includes icons for running queries, saving, and other standard database operations. The main window is divided into two panes: 'Worksheet' and 'Query Result'. The 'Worksheet' pane contains the SQL query: `SELECT * FROM galaxies ORDER BY VECTOR_DISTANCE(embedding, to_vector('[0,1,1,0,0]'), COSINE) FETCH FIRST 3 ROWS ONLY;`. The 'Query Result' pane shows the results of the query, displaying a table with four columns: ID, NAME, DOC, and EMBEDDING. The results are sorted by the cosine distance from the query vector, with the closest matches at the top.

ID	NAME	DOC	EMBEDDING
1	M31	Messier 31 is a barred spiral galaxy in the Andromeda constellation.	[0,1.0E+000,1.0E+000,0,0]
2	M91	Messier 91 is a barred spiral galaxy in the Coma Berenices constellation.	[0,3.0E+000,3.0E+000,0,0]
3	M77	Messier 77 is a barred spiral galaxy in the Cetus constellation.	[0,2.0E+000,2.0E+000,0,0]

查看执行计划：



The screenshot shows the Oracle SQL Developer interface. The top pane displays a query in the Query Builder:

```
SELECT *
FROM galaxies
ORDER BY VECTOR_DISTANCE( embedding, to_vector('[0.1,1,0,0]'), COSINE )
FETCH FIRST 3 ROWS ONLY;
```

The bottom pane shows the Explain Plan for the query, with a total execution time of 0.761 seconds. The plan details are as follows:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				4
COUNT		STOPKEY		3
Filter Predicates ROWNUM<=3				
VIEW	SYS_null			4
SORT		ORDER BY STOPKEY		4
Filter Predicates ROWNUM<=3				
TABLE ACCESS	GALAXIES	FULL		9

Additional information at the bottom of the plan includes:

- Other XML: {info}
- Info type="has_user_tab"

2.2 向量近似检索

向量近似检索（Approximate Search）

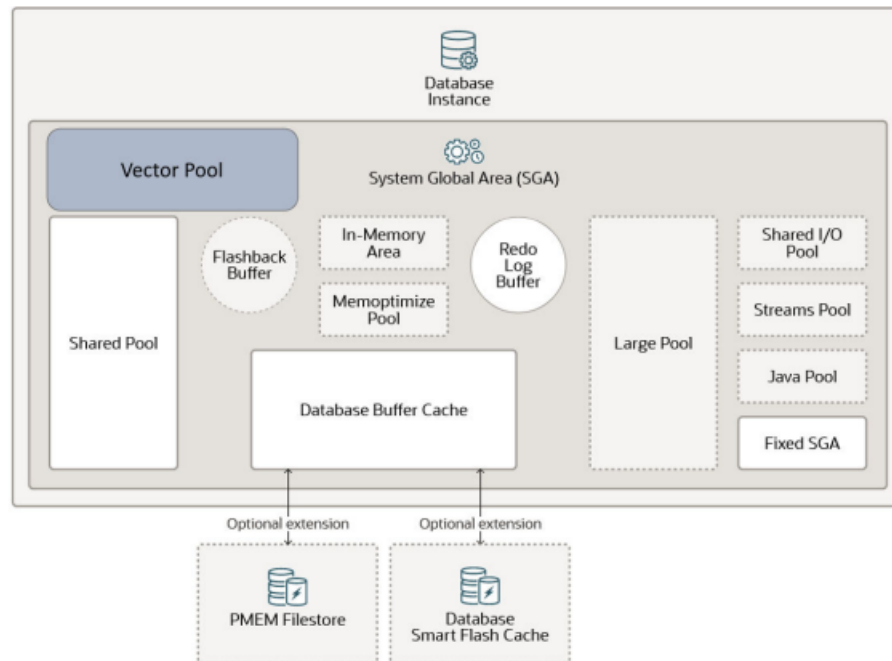
精确检索获得了最高的准确率，但需要遍历所有向量数据集，因此，在向量数据集比较大时，性能很可能会成为问题。向量检索中，准确率和性能之间，往往需要寻找一个平衡。在大数据集上，为了提高性能，利用索引进行向量近似检索是常用的方式。

常见的向量索引有 HNSW 和 IVF 两种。

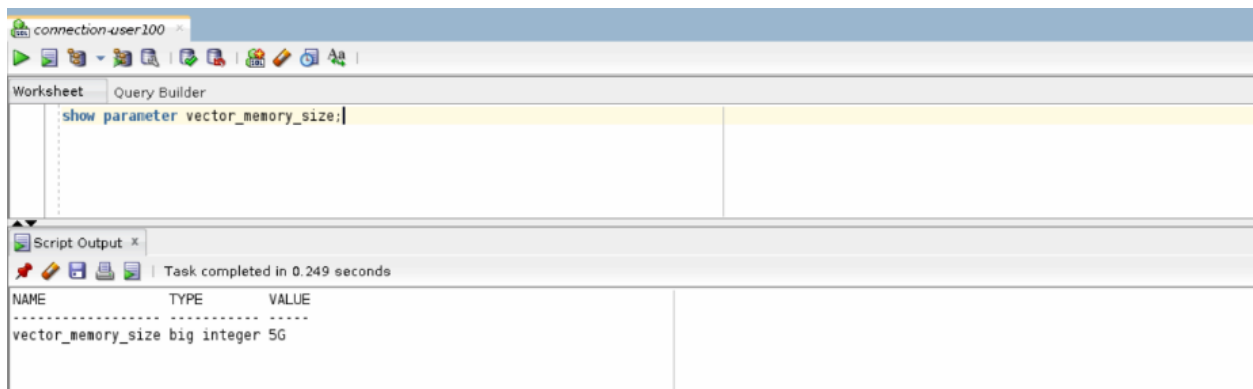
2.2.1 向量内存池

要允许创建向量索引，必须在 SGA 中启用一块新的内存区域，称为向量内存池。向量内存池用于存储 HNSW 向量索引和所有相关元数据，以及用于加速 IVF 索引创建和维护。

向量内存池参数可以由管理员用户进行修改。



```
show parameter vector_memory_size;
```

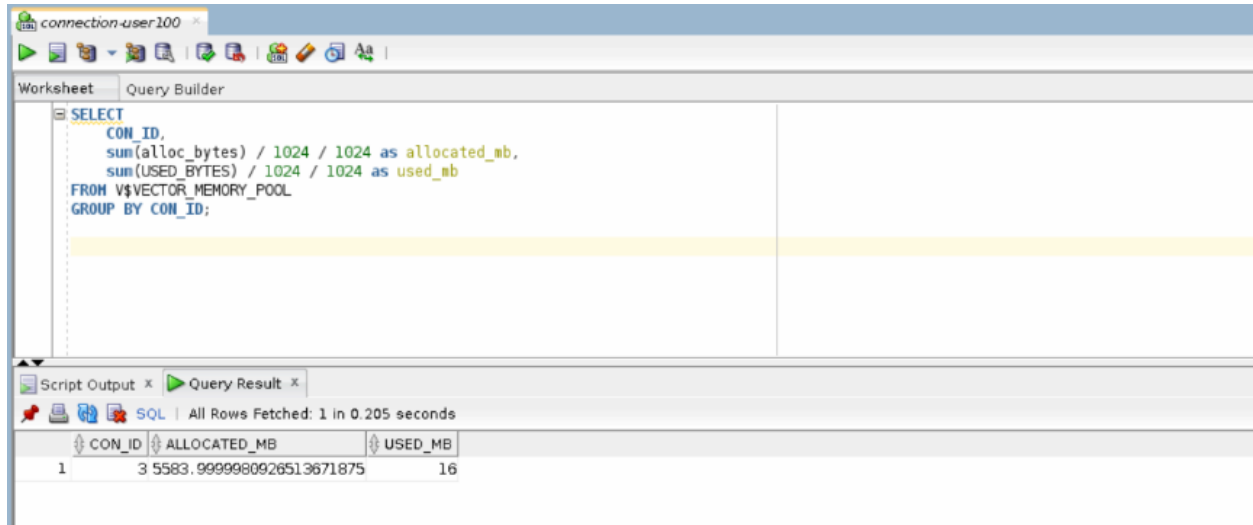


向量内存池大小估算公式： $\text{size of vector pool} = 1.3 * \text{number of vectors} * \text{number of dimensions} * \text{size of vector dimension type}$

2.2.2 向量内存池视图

V\$VECTOR_MEMORY_POOL 视图包含了向量内存的分配和使用情况。比如：

```
SELECT
    CON_ID,
    sum(alloc_bytes) / 1024 / 1024 as allocated_mb,
    sum(USED_BYTES) / 1024 / 1024 as used_mb
FROM V$VECTOR_MEMORY_POOL
GROUP BY CON_ID;
```



2.2.3 创建 HNSW 索引

创建索引语句：

```
CREATE VECTOR INDEX galaxies_hnsw_idx ON galaxies (embedding)
ORGANIZATION INMEMORY NEIGHBOR GRAPH
DISTANCE COSINE
WITH TARGET ACCURACY 90;
-- PARAMETERS (type HNSW, neighbors 32, efconstruction 200)
-- parallel 2;
```

创建 HNSW 索引时，我们可以指定目标准确率 `target accuracy`，并行执行；还可以指定 HNSW 的参数 `M` (即 `neighbors`) 和 `efConstruction` (如上面注释掉的 `Parameters` 一行)。关于 HNSW 相关参数的说明可以参考如下文档：

<https://docs.oracle.com/en/database/oracle/oracle-database/23/vecse/oracle-ai-vector-search-users-guide.pdf> (184 页)

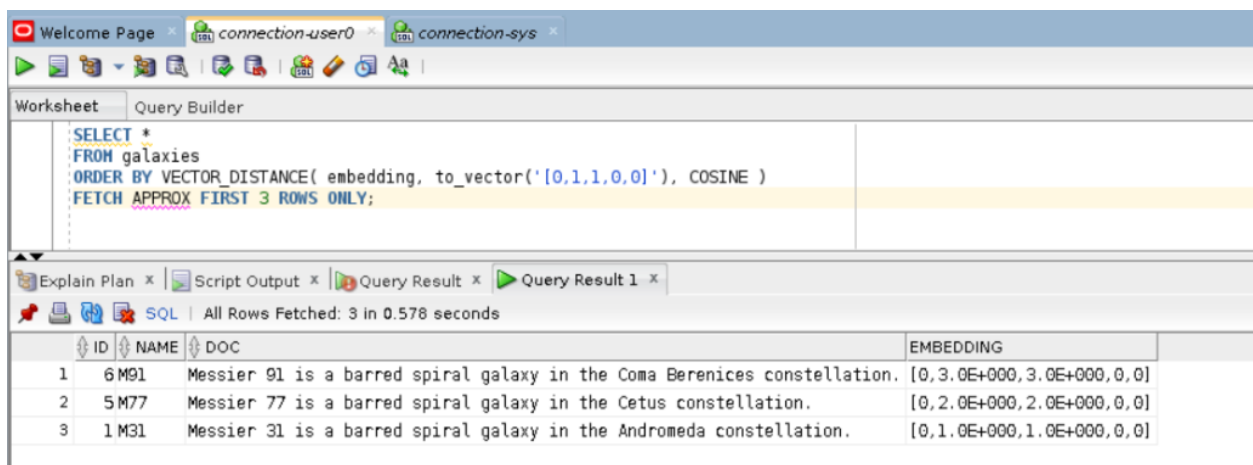
<https://learn.microsoft.com/en-us/javascript/api/@azure/search-documents/hnswparameters?view=azure-node-latest>

2.2.4 HNSW 近似检索

查询 SQL:

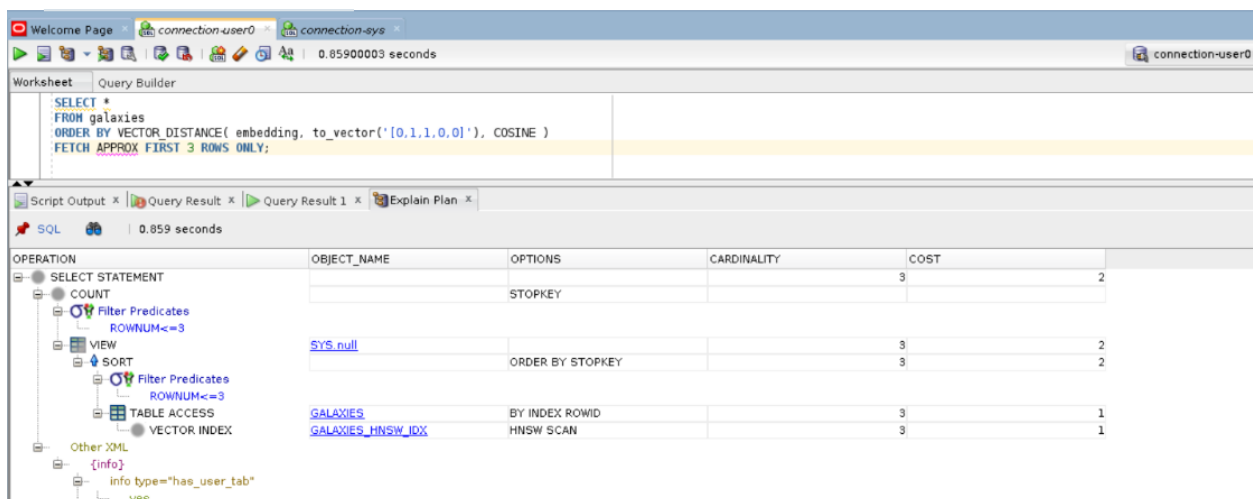
```
SELECT *
FROM galaxies
ORDER BY VECTOR_DISTANCE( embedding, to_vector('[0,1,1,0,0]'), COSINE )
FETCH APPROX FIRST 3 ROWS ONLY;
```

查询结果：



ID	NAME	DOC	EMBEDDING
1	6M91	Messier 91 is a barred spiral galaxy in the Coma Berenices constellation.	[0,3.0E+000,3.0E+000,0,0]
2	5M77	Messier 77 is a barred spiral galaxy in the Cetus constellation.	[0,2.0E+000,2.0E+000,0,0]
3	1M31	Messier 31 is a barred spiral galaxy in the Andromeda constellation.	[0,1.0E+000,1.0E+000,0,0]

查看执行计划：



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2
COUNT		STOPKEY	3	2
Filter Predicates ROWNUM<=3				
VIEW	SYS.null		3	2
SORT		ORDER BY STOPKEY	3	2
Filter Predicates ROWNUM<=3				
TABLE ACCESS	GALAXIES	BY INDEX ROWID	3	1
VECTOR INDEX	GALAXIES_HNSW_IDX	HNSW SCAN	3	1

2.2.5 创建 IVF 索引

如果之前已经在对应的列上创建了向量索引，那么先将其删除，如：

```
drop index galaxies_hnsw_idx;
```

创建 IVF 索引语句：

```
CREATE VECTOR INDEX galaxies_ivf_idx ON galaxies(embedding)
ORGANIZATION NEIGHBOR PARTITIONS
DISTANCE COSINE
WITH TARGET ACCURACY 90;
-- PARAMETERS (type IVF, neighbor partitions 32)
-- parallel 2;
```

创建 IVF 索引时，我们可以指定目标准确率 **target accuracy**、并行执行参数，还可以指定 **partition** 数量等参数。关于 IVF 参数的说明，可以参考如下文档：

<https://docs.oracle.com/en/database/oracle/oracle-database/23/vecse/oracle-ai-vector-search-users-guide.pdf> (196 页)

2.2.6 IVF 近似检索

创建了 IVF 索引之后，我们利用索引进行近似检索（注：由于我们的实验用的数据集很小，所以优化器很可能不会选择走 IVF 索引）

```
SELECT /*+ VECTOR_INDEX_TRANSFORM(galaxies galaxies_ivf_idx) */ *
FROM galaxies
ORDER BY VECTOR_DISTANCE( embedding, to_vector('[0,1,1,0,0]'), COSINE )
FETCH APPROX FIRST 3 ROWS ONLY;
```

查询结果：

Worksheet Query Builder

```
SELECT /*+ VECTOR_INDEX_TRANSFORM(galaxies galaxies_ivf_idx) */ *
FROM galaxies
ORDER BY VECTOR_DISTANCE( embedding, to_vector('[0.1,1.0,0]'), COSINE )
FETCH APPROX FIRST 3 ROWS ONLY;
```

Script Output x Query Result x Explain Plan x Query Result 1 x

SQL All Rows Fetched: 3 in 0.352 seconds

ID	NAME	DOC	EMBEDDING
1	M31	Messier 31 is a barred spiral galaxy in the Andromeda constellation.	[0.1.0E+000,1.0E+000,0.0]
2	M77	Messier 77 is a barred spiral galaxy in the Cetus constellation.	[0.2.0E+000,2.0E+000,0.0]
3	M91	Messier 91 is a barred spiral galaxy in the Coma Berenices constellation.	[0.3.0E+000,3.0E+000,0.0]

查看执行计划：

Worksheet Query Builder

```
SELECT /*+ VECTOR_INDEX_TRANSFORM(galaxies galaxies_ivf_idx) */ *
FROM galaxies
ORDER BY VECTOR_DISTANCE( embedding, to_vector('[0.1,1.0,0]'), COSINE )
FETCH APPROX FIRST 3 ROWS ONLY;
```

Script Output x Query Result x Query Result 1 x Explain Plan x

SQL 0.76 seconds

OPERATION	OBJECT_NAME	OPTIONS	PAR
Filter Predicates ROWNUM<=3			
VIEW	SYS.VW_IVPSI_578B79F1		
SORT		ORDER BY STOPKEY	
Filter Predicates ROWNUM<=3			
HASH JOIN			
Access Predicates VW_IVCR_B5B87E67.CENTROID_ID=VTDX_CNPART.CENTROID_ID			
PART JOIN FILTER	SYS.BF0000	CREATE	
VIEW	SYS.VW_IVCR_B5B87E67		
COUNT		STOPKEY	
Filter Predicates ROWNUM<=2			
VIEW	SYS.VW_IVCN_9A1D2119		
SORT		ORDER BY STOPKEY	
Filter Predicates ROWNUM<=2			
TABLE ACCESS	VECTOR\$GALAXIES_IVF_IDX\$89791_90755_08IVF_FLAT_CENTROIDS	FULL	
PARTITION LIST	VECTOR\$GALAXIES_IVF_IDX\$89791_90755_08IVF_FLAT_CENTROID_PARTITIONS	JOIN-FILTER	:BF
TABLE ACCESS	GALAXIES	FULL	:BF
		BY USER ROWID	

3 部署向量嵌入模型（仅讲师操作）

此节内容仅讲师动手操作及讲解。

以上我们介绍了向量的基本操作。在上面的例子中，我们的向量数据是手工造的，向量的维度也很小。那么，在现实环境中，向量数据是如何来的？答案是向量嵌入模型。

在本实验中，我们将使用开源的向量嵌入模型 `text2vec-large-chinese`

3.1 向量嵌入模型部署

考虑到硬件资源因素，没有足够的资源让每个人都部署一份模型，因此，本操作仅由讲师完成。讲师将向量嵌入模型部分为 REST API 的方式，供大家调用；同时展示源代码并讲解。

源代码：https://github.com/HysunHe/23ai_workshop_prep

```
# 创建 Python 环境
conda create -n ws23ai python=3.12

# 进入新创建的 Python 环境
conda activate ws23ai

# 安装依赖
pip install -r requirements.txt

# 下载源码
git clone https://github.com/HysunHe/23ai_workshop_prep

# 启动模型
cd 23ai_workshop_prep

nohup python -u main.py > lab.out 2>&1 &
```

3.2 向量嵌入模型访问

向量嵌入模型部署完成后，就可以根据提供的 REST API 进行访问了。提供了如下两个 API：

1. 文本向量化 API（后续将用到）

```
curl -X 'POST' \
  'http://<ip>:<port>/workshop/embedding' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "text": "<需要向量化的文本>"
  }'
```

2. 批量数据准备 API（后续将用到）

```
curl -X 'POST' \
  'http://<ip>:<port>/workshop/prepare-data' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "db_user": "<数据库用户名>",
    "db_password": "<数据库用户密码>",
    "table_name": "<表名>",
    "dataset_name": "<数据集名称>"
  }'
```

3.3 库外向量化操作

3.3.1 数据加载

库外向量化指源数据由外部程序向量化之后，再插入或加载到数据库表中。在本例中，我们将使用 Python 程序将文本数据向量化之后，再调用 Oracle 客户包将数据插入到数据库中。这是常用的一种方法，操作方式也与平时的数据加载操作一致。

为了让接下来的实验更接近真实场景，我们将创建另一张表 lab_vecstore：

```
CREATE TABLE lab_vecstore (
  id VARCHAR2(50) DEFAULT SYS_GUID() PRIMARY KEY,
  dataset_name VARCHAR2(50) NOT NULL,
  document CLOB,
  cmetadata JSON,
  embedding VECTOR(*, FLOAT32)
);
```


这里我们没有指定向量的维度，但指定了数据类型格式是 FLOAT32，与向量模型的输出一致。下面我们将源数据文件（源数据集）加载进 lab_vecstore 表。

源数据集：讲师展示源数据集。

接下来，请调用 批量数据准备 API（API 会将上述源数据集进行向量化之后，再插入到数据库中）：

```
curl -X 'POST' \
  'http://10.113.121.221:8099/workshop/prepare-data' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "db_user": "<userx>",
    "db_password": "<password>",
    "table_name": "lab_vecstore",
    "dataset_name": "oracledb_docs"
  }'
```

注：如果没安装 curl 等 api 调用工具，也可以通过如下界面的方式执行：

1. 打开链接（注意 IP 地址为 ODA 数据库的 IP）
http://x.x.x.x:8099/workshop/docs#/default/prepare_data_workshop_prepare_data_post
2. 点击 "Try it out" 按钮
3. 在 "Request body" 输入框中，输入分配给你的 db_user 和 db_password 参数
4. 点击 "Execute" 按钮执行。

API 执行完成后，可以查看一下表中的数据：

```
-- 本数据集总共有 231 条记录
select count(*) from lab_vecstore;

-- 查看数据
select t.cmetadata.source as src_file, embedding as embedding
from lab_vecstore t;
```

SQL Worksheet (lab-110)

Welcome Page connection-user0

Worksheet Query Builder

```
select count(*) from lab_vecstore;

select json_value(cmetadata, '$.source') as src_file, embedding as embedding
from lab_vecstore;
```

Query Result

SQL | Fetched 50 rows in 1.551 seconds

SRC_FILE	EMBEDDING
1 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_080_461.txt	[-5.82970008E-002,2.38936022E-001,-3.94879937E-001,3.3099255E-001,1.1908...
2 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_08aa5_381.txt	[-6.41105354E-001,-2.3126018E-001,7.26056755E-001,-1.37406036E-001,9.095...
3 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_General_334.txt	[2.43672639E-001,9.22110438E-001,-2.95025021E-001,1.76169169E-001,2.0077...
4 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_General_333.txt	[1.26379564E-001,3.12080902E-001,9.65255695E-003,6.49302208E-004,3.14651...
5 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_08aa5_379.txt	[-2.62475222E-001,-3.69362772E-001,-2.7088809E-001,-2.70177901E-001,-2.7...
6 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_General_343.txt	[6.44048825E-002,1.56352296E-001,-4.92639691E-001,-1.51497915E-001,1.843...
7 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_396.txt	[1.67943021E-001,-1.14038014E+000,-2.22395107E-001,-6.78466499E-001,2.50...
8 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_397.txt	[1.38248339E-001,-9.32742357E-001,-1.21381006E+000,-1.1212616E+000,-4.8...
9 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_General_335.txt	[1.17967568E-001,3.47747684E-001,4.62302417E-001,7.22383559E-001,8.52854...
10 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_General_336.txt	[-5.36106229E-002,5.62875988E-001,4.272255E-001,6.32641554E-001,3.491906...
11 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_General_340.txt	[-1.67837575E-001,9.65398788E-001,-5.64283133E-001,1.35987863E-001,6.889...
12 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_General_341.txt	[-2.4456555E-002,2.46404588E-001,-2.96509922E-001,-1.29925862E-001,1.609...
13 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_General_342.txt	[-2.089068E-001,7.86589026E-001,-4.57248807E-001,2.1126838E-001,6.048071...
14 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_401.txt	[-4.16520745E-001,-6.46936953E-001,-2.27507874E-001,-3.94764066E-001,-9...
15 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_404.txt	[-4.29525167E-001,1.17663704E-001,-9.24015284E-001,2.23181114E-001,4.514...
16 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_402.txt	[-3.93528789E-001,-1.1993473E-001,5.34688699E-002,-9.43941116E-001,-5.86...
17 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_403.txt	[-1.00450134E+000,2.03116611E-001,8.17067504E-001,-1.78284526E-001,3.279...
18 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_398.txt	[3.47476184E-001,-1.70679837E-001,-2.30540708E-001,3.94960463E-001,1.953...
19 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_399.txt	[-3.14316511E-001,1.7514573E-001,2.40410075E-001,-4.3961671E-001,1.04514...
20 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_MAA_400.txt	[-4.40898031E-001,-5.74280977E-001,-4.12225015E-002,-6.83319196E-002,1.2...
21 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_优化_388.txt	[-6.16750198E-001,5.18626372E-001,1.01464093E+000,-6.71621025E-001,7.825...
22 /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/20240108_优化_389.txt	[-7.0783639E-001,-2.38721609E-001,2.92639792E-001,-5.46296504E-001,1.292...

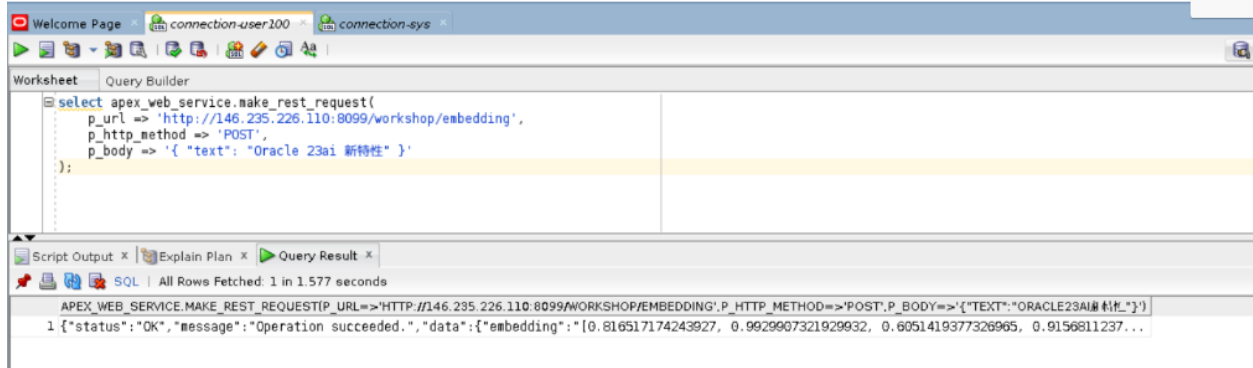
至此，源数据集已经向量化完成，并成功入库（讲师展示并讲解外部向量化的源代码）

3.3.2 向量检索

本实验中，我们使用“Oracle 23ai 新特性”这个文本进行相似度检索。

第一步，先将要检索的文本在库外向量化。我们调用上述提供的 API 完成这一步。API 将返回向量数据。

```
-- 第一步：向量化用户问题
select apex_web_service.make_rest_request(
  p_url => 'http://146.235.226.110:8099/workshop/embedding',
  p_http_method => 'POST',
  p_body => '{ "text": "Oracle 23ai 新特性" }'
);
```



第二步，执行 SQL 语句检索相似的数据，将上一步中返回的向量传入到 VECTOR_DISTANCE 函数中：

```

set serveroutput on;

declare
  l_question varchar2(500) := 'Oracle 23ai 新特性';
  l_input CLOB;
  l_clob CLOB;
  j apex_json.t_values;
  l_embedding CLOB;
begin
  apex_web_service.g_request_headers(1).name := 'Content-Type';
  apex_web_service.g_request_headers(1).value := 'application/json';
  l_input := '{"text": "' || l_question || '"}';

  -- 第一步：向量化用户问题
  l_clob := apex_web_service.make_rest_request(
    p_url => 'http://146.235.226.110:8099/workshop/embedding',
    p_http_method => 'POST',
    p_body => l_input
  );
  apex_json.parse(j, l_clob);
  l_embedding := apex_json.get_varchar2(p_path => 'data.embedding',
p_values => j);
  -- dbms_output.put_line('*** embedding: ' || l_embedding);

  -- 第二步：执行 SQL 语句检索相似的数据，将上一步中返回的向量传入到 VECTOR_DISTANCE 函
数中，从向量数据库中检索出与问题相似的内容
  for rec in (
    select document, json_value(cmetadata, '$.source') as src_file
    from lab_vecstore
    where dataset_name='oracledb_docs'
    order by VECTOR_DISTANCE(embedding, to_vector(l_embedding))
    FETCH FIRST 3 ROWS ONLY
  ) loop
    DBMS_OUTPUT.put_line(chr(10) ||
'#####');
    DBMS_OUTPUT.put_line(rec.document || ' | ' || rec.src_file);
    DBMS_OUTPUT.put_line('#####' ||
chr(10));
  end loop;
end;

```

Welcome Page
connection-user0

Worksheet
Query Builder

```

declare
  l_question varchar2(500) := 'Oracle 23ai 新特性';
  l_input CLOB;
  l_clob CLOB;
  j apex_json.t_values;
  l_embedding CLOB;
begin
  apex_web_service.g_request_headers(1).name := 'Content-Type';
  apex_web_service.g_request_headers(1).value := 'application/json';
  l_input := '{"text": "' || l_question || '"}';

  -- 第一步：向量化用户问题
  l_clob := apex_web_service.make_rest_request(
    p_url => 'http://146.235.226.110:8099/workshop/embedding',
    p_http_method => 'POST',
    p_body => l_input
  );
  apex_json.parse(j, l_clob);
  l_embedding := apex_json.get_varchar2(p_path => 'data.embedding', p_values => j);
  -- dbms_output.put_line('*** embedding: ' || l_embedding);

  -- 第二步：从向量数据库中检索出与问题相似的内容
  for rec in (
    select document, json_value(cmetadata, '$.source') as src_file_
    from lab_sarstora
  )

```

Script Output x
Query Result x

Task completed in 0.35 seconds

问题：关于Oracle 23ai？

解答：Oracle Database 23ai 是 Oracle Database 的下一个长期支持版本。它包含 300 多个新功能，重点关注人工智能（AI）和开发人员生产力。AI 向量搜索等功能使它能够利用新一代 AI 模型来生成和存储文档、图像、声音。

问题：什么是JSON二元性视图（JSON Duality Views）？

解答：JSON 关系二元性视图将我们的关系数据展示为 JSON 文档，允许使用传统 SQL 或直接使用 JSON 执行查询和 DML 操作。

Oracle Database 23ai JSON Relational Duality 通过在单个数据库中统一关系和文档数据模型的优势，彻底改变了 AppDev。

实例参考：<https://oracle-base.com/articles/23/json-relational-duality-views-23> | /home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/202408_23ai_4.txt

问题：Oracle 23ai 有哪些新特性？

解答：Oracle 23ai 发布了数百个新特性，尤其是在AI方面和提高开发人员生产力方面。以下列出一些重大的特性：

- AI向量检索（向量数据库）
- JSON二元性视图（JSON Duality Views）

4 库内向量化操作

Oracle 数据库提供了库内向量化的特性，其允许用户导入向量嵌入模型到数据库中，然后可以直接在 SQL 中对数据进行向量化操作，无需依赖外部的程序，这种方式很大程序的简化了向量数据的加载和检索，非常方便。

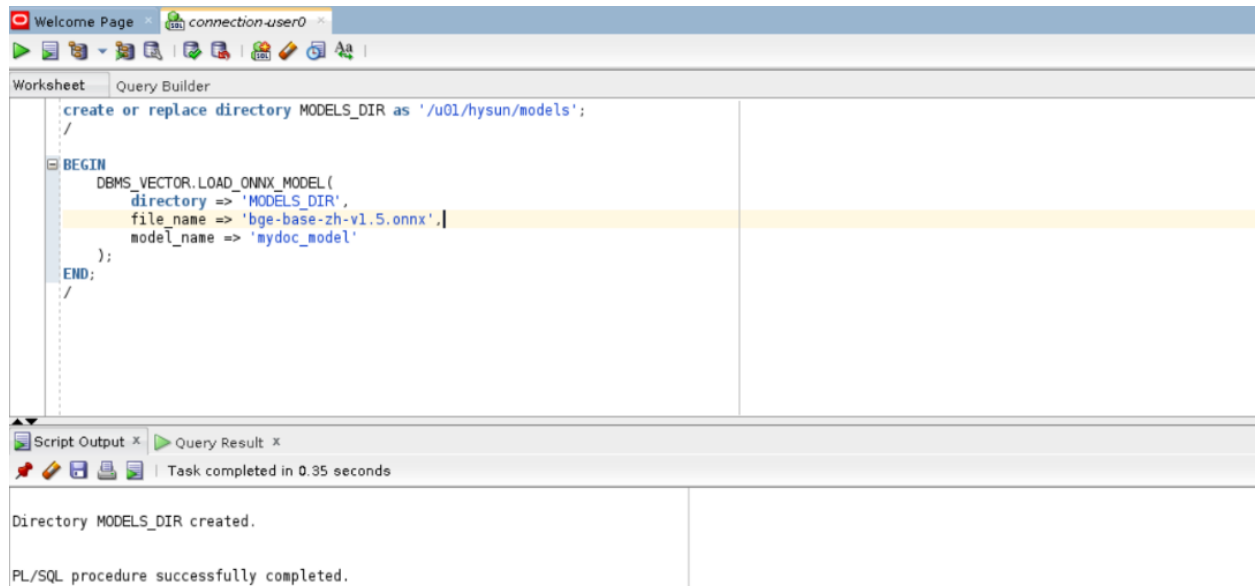
4.1 导入向量嵌入模型

需要加载进 Oracle 数据库的向量嵌入模型必须为标准的 ONNX 格式，且大小在 1G 之内。

```
-- 先将模型文件 bge-base-zh-v1.5.onnx 上传到/u01/hysun/models 目录
-- 创建数据库目录指向模型文件所在目录
create or replace directory MODELS_DIR as '/u01/hysun/models';

-- 导入模型
-- 先删除已经存在的模型（如果存在）：
EXEC DBMS_VECTOR.DROP_ONNX_MODEL(model_name => 'mydoc_model', force => true);

-- 导入模型
BEGIN
    DBMS_VECTOR.LOAD_ONNX_MODEL(
        directory => 'MODELS_DIR',
        file_name => 'bge-base-zh-v1.5.onnx',
        model_name => 'mydoc_model'
    );
END;
/
```



The screenshot shows the Oracle SQL Developer interface. The top bar includes a 'Welcome Page' tab and a 'connection-user0' tab. Below the toolbar, the 'Worksheet' tab is active, displaying a PL/SQL procedure:

```

create or replace directory MODELS_DIR as '/u01/hysun/models';
/

BEGIN
  DBMS_VECTOR.LOAD_ONNX_MODEL(
    directory => 'MODELS_DIR',
    file_name => 'bge-base-zh-v1.5.onnx',
    model_name => 'mydoc_model'
  );
END;
/
  
```

The 'Script Output' tab at the bottom shows the execution results:

```

Directory MODELS_DIR created.

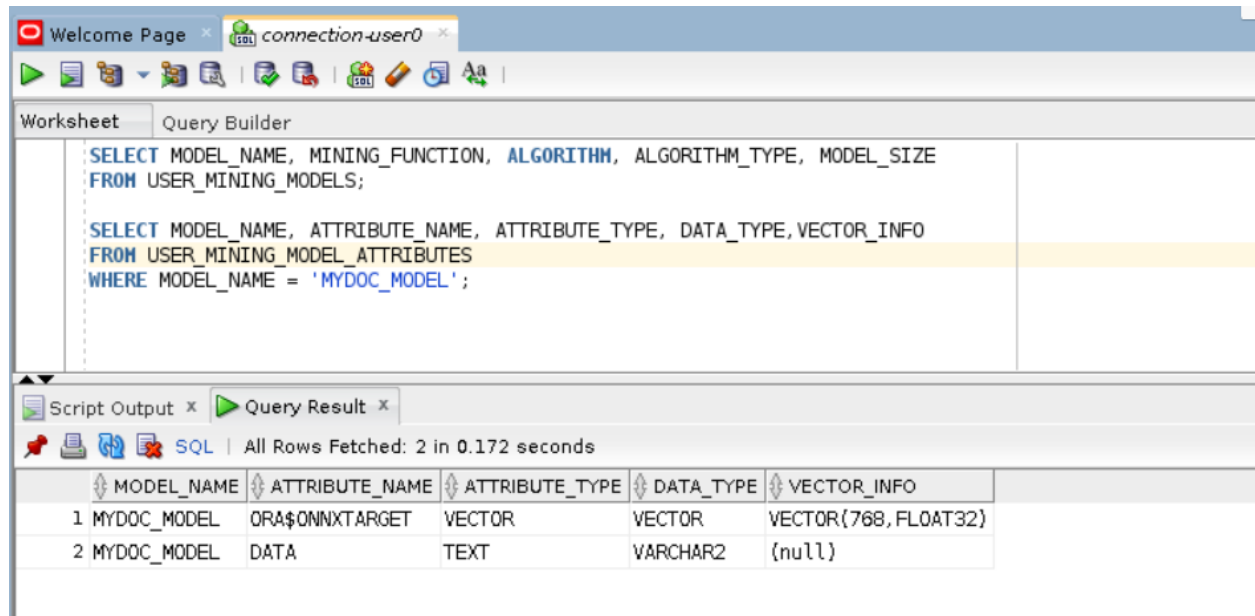
PL/SQL procedure successfully completed.
  
```

模型导入后，可以查看模型的属性：

```

SELECT MODEL_NAME, MINING_FUNCTION, ALGORITHM, ALGORITHM_TYPE, MODEL_SIZE
FROM USER_MINING_MODELS;

SELECT MODEL_NAME, ATTRIBUTE_NAME, ATTRIBUTE_TYPE, DATA_TYPE, VECTOR_INFO
FROM USER_MINING_MODEL_ATTRIBUTES
WHERE MODEL_NAME = 'MYDOC_MODEL';
  
```



The screenshot shows the Oracle SQL Developer interface with two SQL queries entered in the 'Worksheet' tab:

```

SELECT MODEL_NAME, MINING_FUNCTION, ALGORITHM, ALGORITHM_TYPE, MODEL_SIZE
FROM USER_MINING_MODELS;

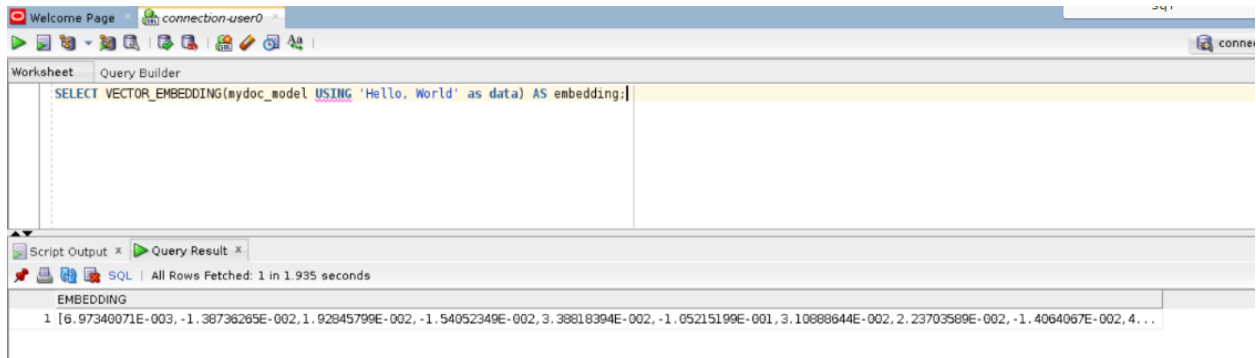
SELECT MODEL_NAME, ATTRIBUTE_NAME, ATTRIBUTE_TYPE, DATA_TYPE, VECTOR_INFO
FROM USER_MINING_MODEL_ATTRIBUTES
WHERE MODEL_NAME = 'MYDOC_MODEL';
  
```

The 'Query Result' tab at the bottom shows the results of the second query, which fetched 2 rows in 0.172 seconds:

	MODEL_NAME	ATTRIBUTE_NAME	ATTRIBUTE_TYPE	DATA_TYPE	VECTOR_INFO
1	MYDOC_MODEL	ORA\$ONNXTARGET	VECTOR	VECTOR	VECTOR(768, FLOAT32)
2	MYDOC_MODEL	DATA	TEXT	VARCHAR2	{null}

做一个简单的向量化操作，测试一下导入的模型是否如期工作：

```
SELECT VECTOR_EMBEDDING(mydoc_model USING 'Hello, World' as data) AS  
embedding;
```



4.2 库内向量化及检索

4.2.1 准备数据

为了排除干扰，我们新建同样的一张表 `lab_vecstore2`：

```
CREATE TABLE lab_vecstore2 (  
    id VARCHAR2(50) DEFAULT SYS_GUID() PRIMARY KEY,  
    dataset_name VARCHAR2(50) NOT NULL,  
    document CLOB,  
    cmetadata JSON,  
    embedding VECTOR(*, FLOAT32)  
);
```

然后从原来的表中拷贝几条数据（作为实验，建议不要拷贝太多数据，以避免造成资源紧张）：

```
insert into lab_vecstore2(dataset_name, document, cmetadata)  
select dataset_name, document, cmetadata  
from lab_vecstore --  
where json_value(cmetadata, '$.source') like '%202408_23ai%';  
commit;  
  
select * from lab_vecstore2;
```

此时, `lab_vecstore2` 表中的 `Embedding` 字段为空（尚未做向量化操作）。

Worksheet Query Builder

select * from lab_vecstore2;

Script Output x Explain Plan x Query Result x

SQL All Rows Fetched: 5 in 0.349 seconds

DATASET_NAME	DOCUMENT	CMETADATA	EMBEDDING
1 36 oracledb_docs	问题：关于Oracle 23ai? 解答：Oracle Database 23ai 是 Oracle Database 的下一个长期支持版本。它包含 300 多个新功能...	oracle.sql.json.JsonDatum@1f12d23a	(null)
2 36 oracledb_docs	问题：Oracle 23ai 有哪些新特性? 解答：Oracle 23ai 发布了数百个新特性，尤其是在AI方面和提高开发人员生产力方面。以下...	oracle.sql.json.JsonDatum@419bca14	(null)
3 36 oracledb_docs	问题：什么是Oracle 向量数据库 (Oracle AI Vector Search)? 解答：Oracle AI Vector Search 专为人工智能 (AI) 工...	oracle.sql.json.JsonDatum@1593f80e	(null)
4 36 oracledb_docs	问题：什么是JSON二元性视图 (JSON Duality Views)? 解答：JSON 关系二元视图将我们的关系数据显式为 JSON 文档，允许使...	oracle.sql.json.JsonDatum@2c6ccf20	(null)
5 36 oracledb_docs	问题：什么是True Cache? 解答：Oracle True Cache 是一种用于 Oracle 数据库的内存缓存方案。它具有数据一致性、自动管...	oracle.sql.json.JsonDatum@25664b95	(null)

4.2.2 库内向量化

```
-- 向量化之前，先查看一下表中的数据，此时 EMBEDDING 字段是空
select * from lab_vecstore2;

-- 执行 SQL 完成向量化
update lab_vecstore2 set embedding=VECTOR_EMBEDDING(mydoc_model USING
document as data);
commit;

-- 向量化之后，再次查看一下表中的数据，此时 EMBEDDING 字段是已经有值了。
select * from lab_vecstore2;
```

Worksheet Query Builder

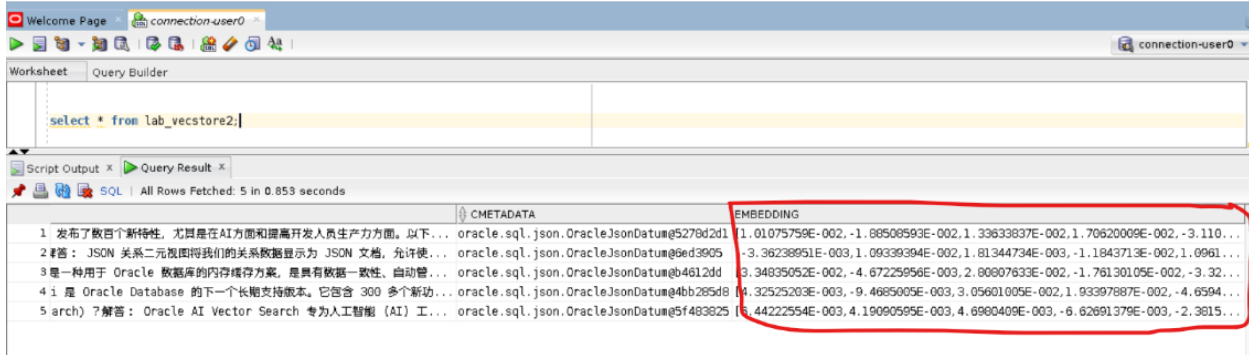
update lab_vecstore2 set embedding=VECTOR_EMBEDDING(mydoc_model USING document as data);
commit;

Script Output x Query Result x

Task completed in 0.376 seconds

5 rows updated.

Commit complete.



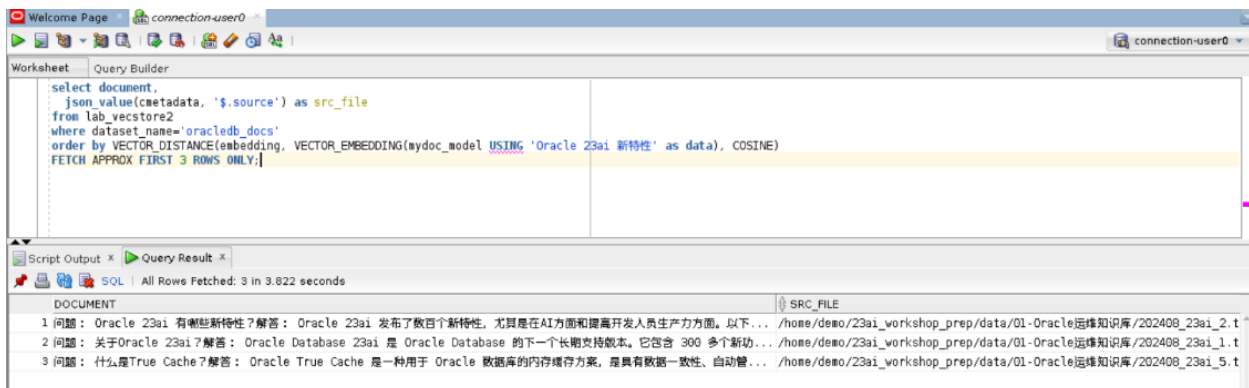
	CMETADATA	EMBEDDING
1 发布了数百个新特性，尤其是在AI方面和开发生产力方面。以下...	oracle.sql.json.JsonObject@5278d2d1	1.01075759E-002,-1.88508593E-002,1.33633837E-002,1.70620009E-002,-3.110...
2 答：JSON 关系二元视图将我们的关系数据展示为 JSON 文档，允许使...	oracle.sql.json.JsonObject@6ed3905	-3.36238951E-003,1.09339394E-002,1.81344734E-003,-1.1843713E-002,1.0961...
3 是一种用于 Oracle 数据库的内存缓存方案，具有数据一致性、自动替...	oracle.sql.json.JsonObject@4612dd	3.34835052E-002,-4.67225956E-003,2.80807633E-002,-1.76130105E-002,-3.32...
4 是 Oracle Database 的下一个长期支持版本。它包含 300 多个新功...	oracle.sql.json.JsonObject@4bb285d8	4.32525203E-003,-9.4685005E-003,3.05601005E-002,1.93397887E-002,-4.6594...
5 arch) ? 解答：Oracle AI Vector Search 专为人工智能 (AI) 工...	oracle.sql.json.JsonObject@5f483825	5.44222554E-003,4.19090595E-003,4.6980409E-003,-6.62691379E-003,-2.3815...

上述操作我们直接用标准的 SQL update 语句对表中的源数据进行了向量化。

4.2.3 相似度检索

由于我们已经在数据库中导入了向量嵌入模型，这里我们可以直接把文本传入 VECTOR_EMBEDDING，进行相似度检索了。

```
select document,
       json_value(cmetadata, '$.source') as src_file
from lab_vecstore2
where dataset_name='oracledb_docs'
order by VECTOR_DISTANCE(embedding, VECTOR_EMBEDDING(mydoc_model USING
'Oracle 23ai 新特性' as data), COSINE)
FETCH APPROX FIRST 3 ROWS ONLY;
```



DOCUMENT	SRC_FILE
1 问题：Oracle 23ai 有哪些新特性？解答：Oracle 23ai 发布了数百个新特性，尤其是在AI方面和开发生产力方面。以下...	/home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/202408_23ai_2.t
2 问题：关于Oracle 23ai？解答：Oracle Database 23ai 是 Oracle Database 的下一个长期支持版本。它包含 300 多个新功...	/home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/202408_23ai_1.t
3 问题：什么是True Cache？解答：Oracle True Cache 是一种用于 Oracle 数据库的内存缓存方案，具有数据一致性、自动替...	/home/demo/23ai_workshop_prep/data/01-Oracle运维知识库/202408_23ai_5.t

5 与第三方向量嵌入模型服务集成（仅演示）

Oracle 数据库向量化操作能支持众多外部提供商提供的 API，包括：

- OCI GenAI (Oracle OCI)
- OpenAI
- Cohere
- HuggingFace
- GoogleAI
- VertexAI
- 以及所有能兼容 OpenAI API 规范的其它服务接口。

本节以腾讯混元 Embeddings 模型为例，演示如何在 Oracle 中直接用简单的 SQL 调用腾讯混元 Embedding 模型，实现数据的向量化。对于其它的 API 提供商，做法上是一样的。

5.1 开通第三方 API 服务

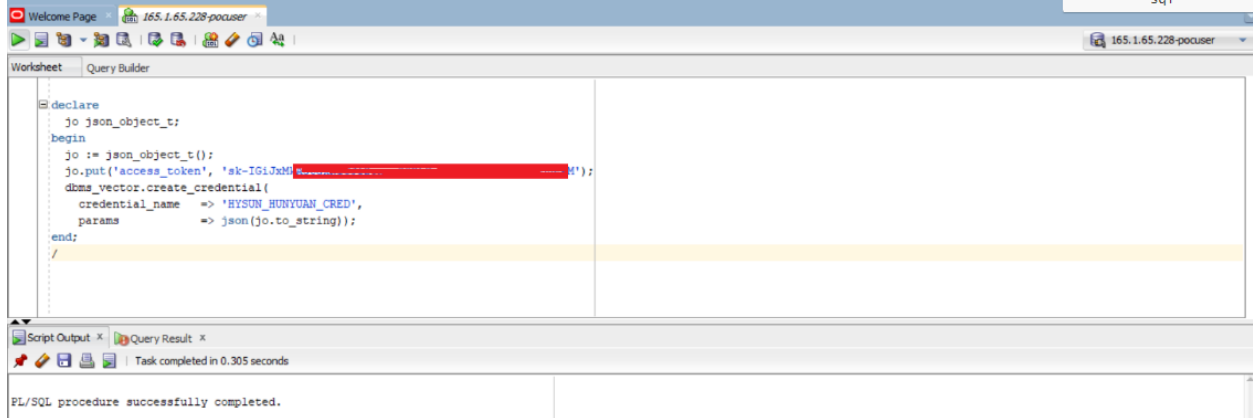
首先，开通腾讯混元大模型服务，并注册 API Key：

<https://console.cloud.tencent.com/hunyuan/api-key>。

5.2 创建访问凭证

利用刚才创建的 API Key，在 Oracle 数据库中创建访问凭证。

```
declare
  jo json_object_t;
begin
  jo := json_object_t();
  jo.put('access_token', 'sk-IGiJxMkAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx');
  dbms_vector.create_credential(
    credential_name => 'HYSUN_HUNYUAN_CRED',
    params          => json(jo.to_string));
end;
/
```



5.3 直接在 SQL 中调用外部 Embedding 服务

在 SQL 中直接调用 `dbms_vector.utl_to_embedding` 或 `dbms_vector.utl_to_embeddings` 将数据转化为向量。

首先，如果当前用户访问 API 的 URL 地址不被允许(ACL 错误)，则先创建 ACE:

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'api.hunyuan.cloud.tencent.com',
    lower_port => 443,
    upper_port => 443,
    ace => xs$ace_type(privilege_list => xs$name_list('http'),
      principal_name => '<数据库用户名>',
      principal_type => xs_acl.ptype_db)
  );
END;
```

直接在 SQL 中调用混元 API Embedding 服务, 如:

```
SELECT
  dbms_vector.utl_to_embedding(
    'Oracle 向量数据库动手实验培训',
    json('{
      "provider": "OpenAI",
      "credential_name": "HYSUN_HUNYUAN_CRED",
      "url": "https://api.hunyuan.cloud.tencent.com/v1/embeddings",
      "model": "hunyuan-embedding"
    }')
  ) embedding
FROM dual;
```



6 总结

至此，我们已经完成了 Oracle 向量数据库的动手实验第一部分。

本节内容中，我们实现了利用向量检索的精确检索和近似检索两种方式。现实中，在相对较大的数据集中，精确检索往往只有在融合数据库中才能发挥出真正的优势。比如，在我们的实验中，我们使用标量字段 `dataset_name='oracledb_docs'` 将需要进行向量检索的数据集大幅度缩小了，有效弥补了精确检索的性能问题。

同时，我们还实现了 Oracle 库外向量化和库内向量化两种方式。库内向量化因其简单便捷的特点，有可能成为未来向量化的一个重要方向。然而，就目前而言，局限于数据库硬件资源现状，往往库外向量化方式使用更多。

最后，我们还介绍了如何通过第三方 Embedding API 服务集成，在 SQL 中调用第三方服务完成向量化的过程。

下一节我们将进行第二部分的实验：结合 Oracle 向量检索的 RAG 应用。