Oracle

1Z0-184-25 Exam

Oracle AI Vector Search Professional

Questions & Answers
Demo

Version: 6.0

Question: 1

When generating vector embeddings outside the database, what is the most suitable option for storing the embeddings for later use?

A. In a CSV file
B. In a binary FVEC file with the relational data in a CSV file
C. In the database as BLOB (Binary Large Object) data
D. In a dedicated vector database

Answer: D

Explanation:

When vector embeddings are generated outside the database, the storage choice must balance efficiency, scalability, and usability for similarity search. A CSV file (A) is simple and human-readable but inefficient for large-scale vector operations due to text parsing overhead and lack of indexing support. A binary FVEC file (B) offers a compact format for vectors, reducing storage size and improving read performance, but separating relational data into a CSV complicates integration and querying, making it suboptimal for unified workflows. Storing embeddings as BLOBs in a relational database (C) integrates well with structured data and supports SQL access, but it lacks the specialized indexing (e.g., HNSW, IVF) and query optimizations that dedicated vector databases provide. A dedicated vector database (D), such as Milvus or Pinecone (or Oracle 23ai's vector capabilities if internal), is purpose-built for high-dimensional vectors, offering efficient storage, advanced indexing, and fast approximate nearest neighbor (ANN) searches. For external generation scenarios, where embeddings are not immediately integrated into Oracle 23ai, a dedicated vector database is the most suitable due to its performance and scalability advantages. Oracle's AI Vector Search documentation indirectly supports this by emphasizing optimized vector storage for search efficiency, though it focuses on in-database solutions.
Reference:Oracle Database 23ai AI Vector Search Guide, Chapter on Vector Storage and Indexing.

Question: 2

When generating vector embeddings for a new dataset outside of Oracle Database 23ai, which factor is crucial to ensure meaningful similarity search results?

A. The choice of programming language used to process the dataset (e.g., Python, Java)
B. The physical location where the vector embeddings are stored
C. The storage format of the new dataset (e.g., CSV, JSON)

D. The same vector embedding model must be used for vectorizing the data and creating a query vector

Answer: D

Explanation:

Meaningful similarity search relies on the consistency of the vector space in which embeddings reside. Vector embeddings are generated by models (e.g., BERT, SentenceTransformer) that map data into a high-dimensional space, where proximity reflects semantic similarity. If different models are used for the dataset and query vector, the embeddings will be in incompatible spaces, rendering distance metrics (e.g., cosine, Euclidean) unreliable. The programming language (A) affects implementation but not the semantic consistency of embeddings—Python or Java can use the same model equally well. The physical storage location (B) impacts accessibility and latency but not the mathematical validity of similarity comparisons. The storage format (C) influences parsing andingestion but does not determine the embedding space. Oracle 23ai's vector search framework explicitly requires the same embedding model for data and queries to ensure accurate results, a principle that applies universally, even outside the database.
Reference:Oracle Database 23ai AI Vector Search Guide, Section on Vector Embedding Consistency.

Question: 3

You are working with vector search in Oracle Database 23ai and need to ensure the integrity of your vector data during storage and retrieval. Which factor is crucial for maintaining the accuracy and reliability of your vector search results?

A. Using the same embedding model for both vector creation and similarity search
B. Regularly updating vector embeddings to reflect changes in the source data
C. The specific distance algorithm employed for vector comparisons
D. The physical storage location of the vector data

Answer: A

Explanation:

In Oracle Database 23ai, vector search accuracy hinges on the consistency of the embedding model. The VECTOR data type stores embeddings as fixed-dimensional arrays, and similarity searches (e.g., using VECTOR_DISTANCE) assume that all vectors—stored and query—are generated by the same model. This ensures they occupy the same semantic space, making distance calculations meaningful. Regular updates (B) maintain data freshness, but if the model changes, integrity is compromised unless all embeddings are regenerated consistently. The distance algorithm (C) (e.g., cosine, Euclidean) defines how similarity is measured but relies on consistent embeddings; an incorrect model mismatch undermines any algorithm. Physical storage location (D) affects performance, not integrity. Oracle's documentation stresses model consistency as a prerequisite for reliable vector search within its native capabilities.

Reference:Oracle Database 23ai AI Vector Search Guide, Chapter on Vector Search Prerequisites.

Question: 4

Which DDL operation is NOT permitted on a table containing a VECTOR column in Oracle Database 23ai?

A. Creating a new table using CTAS (CREATE TABLE AS SELECT) that includes the VECTOR column from the original table
B. Dropping an existing VECTOR column from the table
C. Modifying the data type of an existing VECTOR column to a non-VECTOR type
D. Adding a new VECTOR column to the table

Answer: C

Explanation:

Oracle Database 23ai imposes restrictions on DDL operations for tables with VECTOR columns to preserve data integrity. CTAS (A) is permitted, as it copies the VECTOR column intact into a new table, maintaining its structure. Dropping a VECTOR column (B) is allowed via ALTER TABLE DROP COLUMN, as it simply removes the column without altering its type. Adding a new VECTOR column (D) is supported with ALTER TABLE ADD, enabling schema evolution. However, modifying an existing VECTOR column's data type to a non-VECTOR type (C) (e.g., VARCHAR2, NUMBER) is not permitted because VECTOR is a specialized type with dimensional and format constraints (e.g., FLOAT32), and Oracle does not support direct type conversion due to potential loss of semantic meaning and structure. This restriction is documented in Oracle's SQL reference.
Reference:Oracle Database 23ai SQL Language Reference, Section on VECTOR Data Type Restrictions.

Question: 5

Which SQL statement correctly adds a VECTOR column named "v" with 4 dimensions and FLOAT32 format to an existing table named "my_table"?

A. ALTER TABLE my_table MODIFY (v VECTOR(4, FLOAT32))
B. ALTER TABLE my_table ADD (v VECTOR(4, FLOAT32))
C. UPDATE my_table SET v = VECTOR(4, FLOAT32)
D. ALTER TABLE my_table ADD v VECTOR(4, FLOAT32)

Answer: B

Explanation:

To add a new column to an existing table, Oracle uses the ALTER TABLE statement with the ADD clause. Option B, ALTER TABLE my_table ADD (v VECTOR(4, FLOAT32)), correctly specifies the column name "v", the VECTOR type, and its attributes (4 dimensions, FLOAT32 precision) within parentheses,

aligning with Oracle's DDL syntax for VECTOR columns. Option A uses MODIFY, which alters existing columns, not adds new ones, making it incorrect here. Option C uses UPDATE, a DML statement for updating data, not a DDL operation for schema changes. Option D omits parentheses around the VECTOR specification, which is syntactically invalid as Oracle requires dimensions and format to be enclosed. The SQL Language Reference confirms this syntax for adding VECTOR columns.
Reference:Oracle Database 23ai SQL Language Reference, Section on ALTER TABLE.