

ChromaDB

```
pip install langchain-chroma # LangChain integration
```

ChromaDB (or just **Chroma**) is a **vector database** – a tool that stores **text + embeddings** so that we can **search semantically** using a query.

In simpler words:

ChromaDB helps a computer understand and remember text in a smart way, so that when you ask a question, it finds relevant info even if words don't exactly match.

What is ChromaDB?

A lightweight, open-source **vector database** designed for AI applications. It:

- Stores **text embeddings** (vector representations)
- Enables fast **semantic search**
- Runs locally or on a server
- Integrates seamlessly with LangChain

ChromaDB is one of the most used vector stores in LangChain because:

- It's **free, open-source**
- Works **locally** (no internet needed)
- Very **fast** and **simple to use**
- No external account or API key needed

How It Works (Step by Step)?

1. Input: Documents

You give ChromaDB some text documents.

```
texts = ["Apple is a fruit", "Quantum computing uses qubits"]
```

2. Embedding: Convert text to numbers

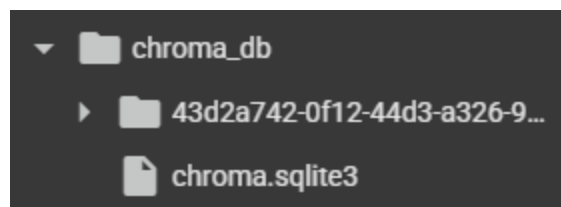
You use an **embedding model** (like from HuggingFace or OpenAI) to turn text into vectors.

```
from langchain_huggingface import HuggingFaceEmbeddings
embedding = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
```

3. Store in Chroma

```
from langchain.vectorstores import Chroma

vectorstore = Chroma.from_texts(
    texts=texts,
    embedding=embedding,
    persist_directory="./chroma_db"
)
```

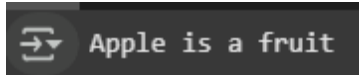


This creates a folder `chroma_db/` on disk where it stores the vectors.

4. Similarity Search

Now you can search with a question:

```
query = "What are fruits?"
results = vectorstore.similarity_search(query, k=1)
print(results[0].page_content)
```



Even if your question doesn't match the exact text, it finds semantically similar results.

persist_directory

- If you give `persist_directory="..."`, Chroma will **save the database to disk**.
- You can **load it later** using:


```
Chroma(persist_directory="...", embedding_function=embedding)
```

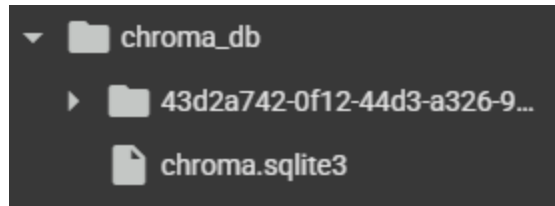
Core Features

1. **Persistent Storage:** Saves vectors to disk (unlike FAISS)
2. **Metadata Filtering:** Search with conditions (e.g., `{"source": "file.pdf"}`)
3. **Hybrid Search:** Combine vector + keyword search (in paid versions)

Saving to the disk

```
## Saving to the disk
vectordb=Chroma.from_documents(documents=splits,embedding=embedding,persist_directory="./chroma_db")

db.persist() #  Saves to disk
```



Load:

```
db2= Chroma(persist_directory="./chroma_db", embedding_function=embedd  
ing)
```

```
docs=db2.similarity_search(query)  
docs
```

```
[Document(id='6c002673-48b4-4bab-ac1f-b2e1cf15f2f0', metadata={'source': '/content/speech.txt'}, page_content='It is a distressing and op  
performed in thus addressing you. There are, it may be, many months of fiery trial and sacrifice ahead of us. It is a fearful thing to le  
terrible and disastrous of all wars, civilization itself seeming to be in the balance. But the right is more precious than peace, and we  
nearest our hearts—for democracy,'),  
Document(id='f7d8ab16-b9b4-43cb-8906-aed7f79cadc5', metadata={'source': '/content/speech.txt'}, page_content='Just because we fight with  
for ourselves but what we shall wish to share with all free peoples, we shall, I feel confident, conduct our operations as belligerents w  
the principles of right and of fair play we profess to be fighting for.\n\n'),  
Document(id='cef631b9-d5f6-41ee-8068-43af7aad0e18', metadata={'source': '/content/speech.txt'}, page_content='for the right of those who  
governments, for the rights and liberties of small nations, for a universal dominion of right by such a concert of free peoples as shall  
itself at last free.'),  
Document(id='7850e7b7-6fc6-438a-a6e1-5e0844ab949c', metadata={'source': '/content/speech.txt'}, page_content='The world must be made saf  
tested foundations of political liberty. We have no selfish ends to serve. We desire no conquest, no dominion. We seek no indemnities for  
we shall freely make. We are but one of the champions of the rights of mankind. We shall be satisfied when those rights have been made as  
them.')] ]
```

! If You Skip **persist_directory** in Chroma?

Then it behaves as an **in-memory** vector store:

- Works fine while the script runs
- But **nothing gets saved** to disk
- You **cannot reload** the index later

Retriever option

```

### Retriever option
retriever=vectordb.as_retriever()
retriever.invoke(query)[0].page_content

```

```
'To such a task we can dedicate our lives and our fortunes.
```

Chroma vs FAISS

Feature	Chroma	FAISS
Saving Method	<code>persist_directory="..."</code>	<code>db. save_local ("faiss_index")</code>
Necessary to Save Path?	✅ Yes (or it'll store in-memory only)	❌ Optional but recommended
What If Not Provided?	Stays in-memory (lost after script ends)	Also in-memory (unless explicitly saved)
Loading Later?	<code>Chroma(..., persist_directory=...)</code>	<code>FAISS. load_local ("faiss_index", ...)</code>
Stores Metadata?	✅ Yes	❌ Limited metadata
Search Performance	Fast, and includes metadata filters	Fast, lower-level (closer to raw vectors)

1. Saving in FAISS

Proper Way:

```

from langchain_community.vectorstores import FAISS

# Method 1: Save during creation (not recommended)
db = FAISS.from_documents(docs, embeddings) # Creates in memory
db.save_local("faiss_index") # Explicit save to disk

```

```
# Method 2: Preferred one-liner
FAISS.from_documents(docs, embeddings).save_local("faiss_index")
```

Key Points:

- FAISS **doesn't auto-save** - you must manually call `save_local()`
- Without `save_local()`, vectors are **lost when program exits**
- Saved to disk as:
 - `faiss_index/index.faiss` (vectors)
 - `faiss_index/index.pkl` (metadata)

2. Saving in ChromaDB

Proper Way:

```
from langchain_chroma import Chroma

# Method 1: Persistent directory (recommended)
db = Chroma.from_documents(
    docs,
    embeddings,
    persist_directory="./chroma_db" # Auto-saves
)

# Method 2: In-memory (not saved)
db = Chroma.from_documents(docs, embeddings) # Temp storage
```

Key Points:

- Chroma **auto-saves** when `persist_directory` is provided
- Without `persist_directory`:
 - Data stays **in memory only**

- **Lost when program closes**
- Saved as an **SQLite database** (`./chroma_db/chroma.sqlite3` + files)

Key Differences

Feature	FAISS	ChromaDB
Persistence	Manual (<code>save_local()</code>)	Auto-save with <code>persist_directory</code>
Storage Format	Binary files	SQLite + embedded files
Default Location	Current directory	Memory (if no path given)
Metadata Handling	Pickle file	Native database storage
Reloading	<code>FAISS.load_local("faiss_index", embeddings)</code>	<code>Chroma(persist_directory="./chroma_db", embedding_function=embeddings)</code>