

[AI & Vectors](#) > [Third-Party Tools](#) > [LangChain](#) >

LangChain

[LangChain](#) is a popular framework for working with AI, Vectors, and embeddings. LangChain supports using Supabase as a [vector store](#), using the `pgvector` extension.

Initializing your database

Prepare your database with the relevant tables:

[Dashboard](#) [SQL](#)

- 1 Go to the [SQL Editor](#) page in the Dashboard.
- 2 Click **LangChain** in the Quick start section.
- 3 Click **Run**.

Usage

You can now search your documents using any Node.js application. This is intended to be run on a secure server route.

```
1 import { SupabaseVectorStore } from 'langchain/vectorstores/supabase'
2 import { OpenAIEmbeddings } from 'langchain/embeddings/openai'
3 import { createClient } from '@supabase/supabase-js'
4
5 const supabaseKey = process.env.SUPABASE_SERVICE_ROLE_KEY
6 if (!supabaseKey) throw new Error(`Expected SUPABASE_SERVICE_ROLE_KEY`)
7
8 const url = process.env.SUPABASE_URL
9 if (!url) throw new Error(`Expected env var SUPABASE_URL`)
```

```
10
11 export const run = async () => {
12   const client = createClient(url, supabaseKey)
13
14   const vectorStore = await SupabaseVectorStore.fromTexts(
15     ['Hello world', 'Bye bye', "What's this?"],
16     [{ id: 2 }, { id: 1 }, { id: 3 }],
17     new OpenAIEmbeddings(),
18     {
19       client,
20       tableName: 'documents',
21       queryName: 'match_documents',
22     }
23   )
24
25   const resultOne = await vectorStore.similaritySearch('Hello world', 1)
26
27   console.log(resultOne)
28 }
```

Simple metadata filtering

Given the above `match_documents` Postgres function, you can also pass a filter parameter to only return documents with a specific metadata field value. This filter parameter is a JSON object, and the `match_documents` function will use the Postgres JSONB Containment operator `@>` to filter documents by the metadata field values you specify. See details on the [Postgres JSONB Containment operator](#) for more information.

```
1 import { SupabaseVectorStore } from 'langchain/vectorstores/supabase'
2 import { OpenAIEmbeddings } from 'langchain/embeddings/openai'
3 import { createClient } from '@supabase/supabase-js'
4
5 // First, follow set-up instructions above
6
7 const privateKey = process.env.SUPABASE_SERVICE_ROLE_KEY
8 if (!privateKey) throw new Error(`Expected env var SUPABASE_SERVICE_ROLE_KEY`)
9
10 const url = process.env.SUPABASE_URL
11 if (!url) throw new Error(`Expected env var SUPABASE_URL`)
12
13 export const run = async () => {
14   const client = createClient(url, privateKey)
15
16   const vectorStore = await SupabaseVectorStore.fromTexts(
17     ['Hello world', 'Hello world', 'Hello world'],
```

```
18   [{ user_id: 2 }, { user_id: 1 }, { user_id: 3 }],
19   new OpenAIEmbeddings(),
20   {
21     client,
22     tableName: 'documents',
23     queryName: 'match_documents',
24   }
25 )
26
27 const result = await vectorStore.similaritySearch('Hello world', 1, {
28   user_id: 3,
29 })
30
31 console.log(result)
32 }
```

Advanced metadata filtering

You can also use query builder-style filtering (similar to how the Supabase JavaScript library works) instead of passing an object. Note that since the filter properties will be in the metadata column, you need to use arrow operators (`->` for integer or `->>` for text) as defined in Postgres API documentation and specify the data type of the property (e.g. the column should look something like `metadata->some_int_value::int`).

```
1 import { SupabaseFilterRPCCall, SupabaseVectorStore } from 'langchain/vectorstores/supabase'
2 import { OpenAIEmbeddings } from 'langchain/embeddings/openai'
3 import { createClient } from '@supabase/supabase-js'
4
5 // First, follow set-up instructions above
6
7 const privateKey = process.env.SUPABASE_SERVICE_ROLE_KEY
8 if (!privateKey) throw new Error(`Expected env var SUPABASE_SERVICE_ROLE_KEY`)
9
10 const url = process.env.SUPABASE_URL
11 if (!url) throw new Error(`Expected env var SUPABASE_URL`)
12
13 export const run = async () => {
14   const client = createClient(url, privateKey)
15
16   const embeddings = new OpenAIEmbeddings()
17
18   const store = new SupabaseVectorStore(embeddings, {
19     client,
20     tableName: 'documents',
21   })
```

```

22
23   const docs = [
24     {
25       pageContent:
26         'This is a long text, but it actually means something because vector database c
27       metadata: { b: 1, c: 10, stuff: 'right' },
28     },
29     {
30       pageContent:
31         'This is a long text, but it actually means something because vector database c
32       metadata: { b: 2, c: 9, stuff: 'right' },
33     },
34     { pageContent: 'hello', metadata: { b: 1, c: 9, stuff: 'right' } },
35     { pageContent: 'hello', metadata: { b: 1, c: 9, stuff: 'wrong' } },
36     { pageContent: 'hi', metadata: { b: 2, c: 8, stuff: 'right' } },
37     { pageContent: 'bye', metadata: { b: 3, c: 7, stuff: 'right' } },
38     { pageContent: "what's this", metadata: { b: 4, c: 6, stuff: 'right' } },
39   ]
40
41   await store.addDocuments(docs)
42
43   const funcFilterA: SupabaseFilterRPCall = (rpc) =>
44     rpc
45       .filter('metadata->b::int', 'lt', 3)
46       .filter('metadata->c::int', 'gt', 7)
47       .textSearch('content', `multidimensional` & `spaces`, {
48         config: 'english',
49       })
50
51   const resultA = await store.similaritySearch('quantum', 4, funcFilterA)
52
53   const funcFilterB: SupabaseFilterRPCall = (rpc) =>
54     rpc
55       .filter('metadata->b::int', 'lt', 3)
56       .filter('metadata->c::int', 'gt', 7)
57       .filter('metadata->stuff', 'eq', 'right')
58
59   const resultB = await store.similaritySearch('hello', 2, funcFilterB)
60
61   console.log(resultA, resultB)
62 }

```

Hybrid search

LangChain supports the concept of a hybrid search, which combines Similarity Search with Full Text Search. Read the official docs to get started: [Supabase Hybrid Search](https://supabase.com/docs/guides/ai/langchain).

You can install the LangChain Hybrid Search function through our [database.dev package manager](#).


Resources

Official [LangChain site](#).


Official [LangChain docs](#).

Supabase [Hybrid Search](#).

Edit this page on GitHub [↗](#)

 Need some help? [Contact support](#)

 Latest product updates? [See Changelog](#)

 Something's not right? [Check system status](#)

© Supabase Inc

—

Contributing

Author Styleguide

Open Source

SupaSquad

Privacy Settings

