



MySQL Vector and AI

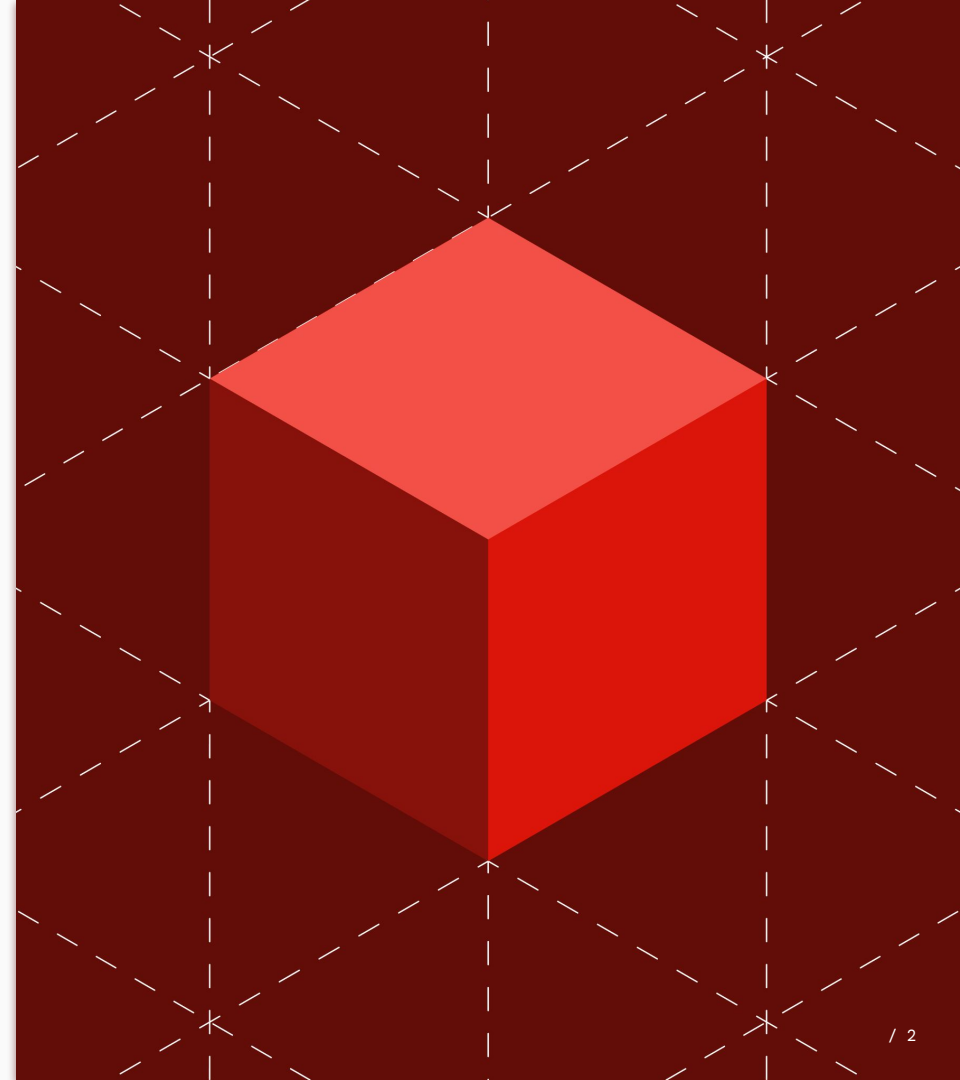
Adding AI Capabilities to Your Database

2 Feb 2025 - Mattias Jonsson, PingCAP

AGENDA

LLM, RAG, AI, Vector ?!?

Vector datatype in MySQL
and other DBs



AGENDA

LLMs

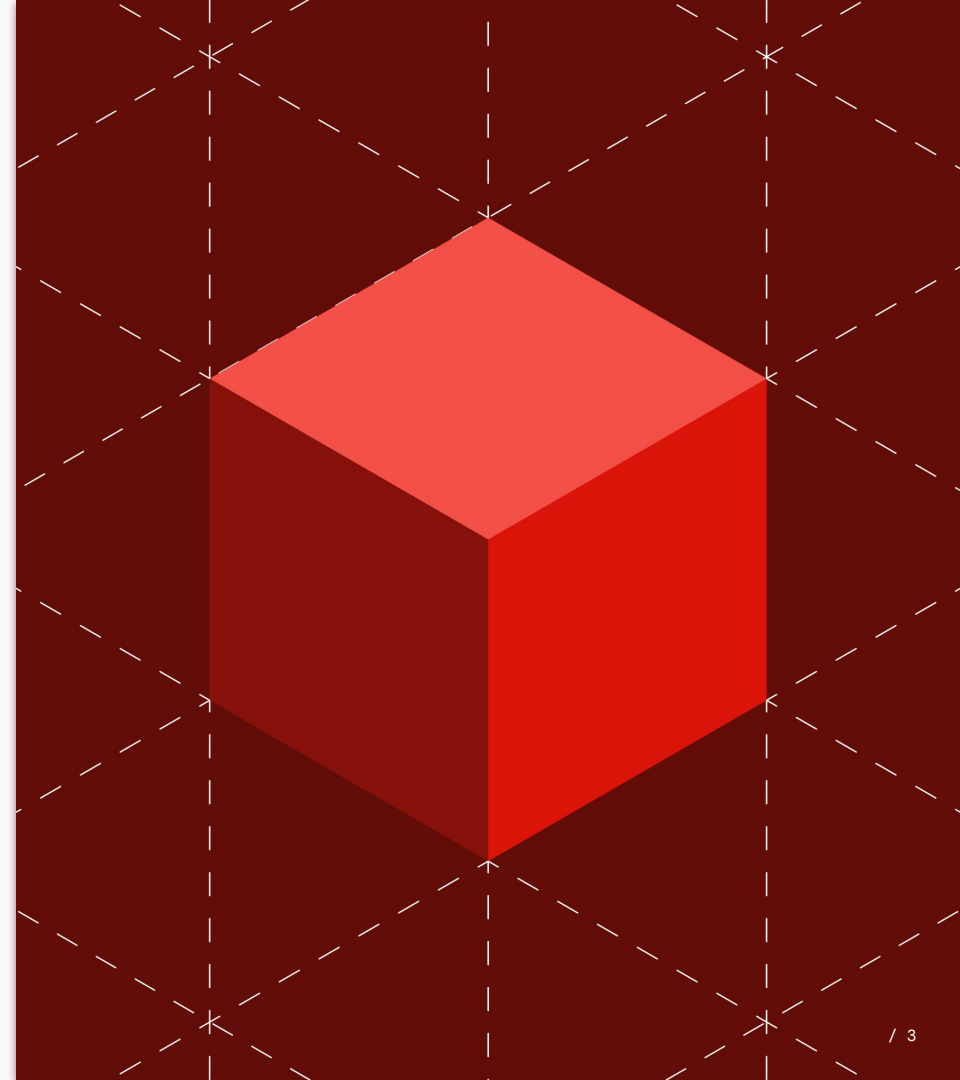
Semantic search

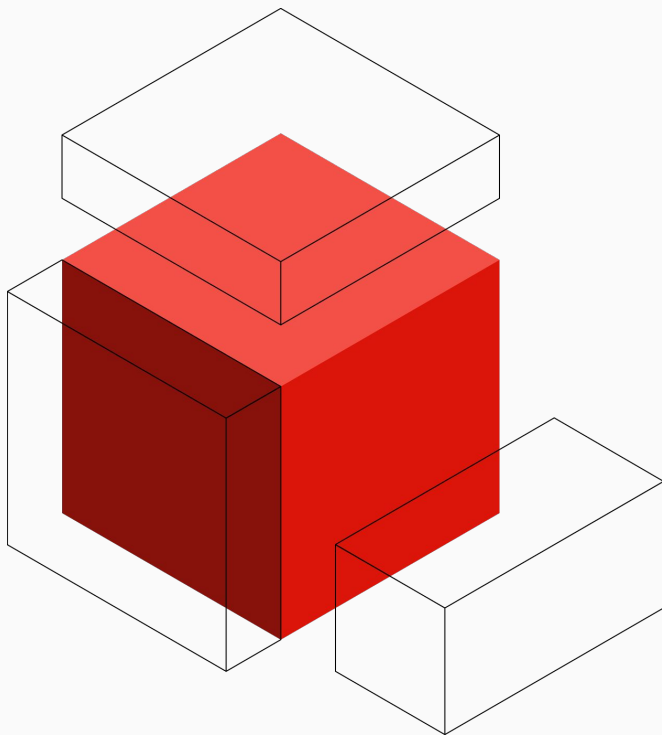
Vector embeddings

Vector search

RAG

What?!?





Large Language Model

- AI/Statistical model, trained on massive amount of data.
- Takes input/question/prompt and generates output (Generative AI).
- Not up-to-date since it was trained.
- May miss your specific content.
- Will generate an answer even if not sure (hallucination).



>>> What is TiDB?

<think> </think>

TiDB is a modern, in-memory database engine optimized for speed and memory efficiency. It's designed to work seamlessly with the NoSQL query language **Pymongo** (which is similar to MongoDB's Python library), providing a fast, lightweight alternative to external databases.

...



>>> What is TiDB?

<think> </think>

TiDB 是一个开源的全栈数据库解决方案, 旨在为开发者提供简单、高效、易用的数据库服务。它包括以下核心组件:

...



>>> What is TiDB? Please reply in english
<think> ... some reasoning text ... </think>

TiDB is an open-source distributed database system designed for high-throughput, low-latency, and reliable data processing in modern distributed systems. It is built on top of the **Bolt** language and its runtime, offering a unified framework for SQL and bolt-based operations.

...



ollama run deepseek-r1:32B

>>> What is TiDB?

<think> ... a lot of reasoning text ... </think>

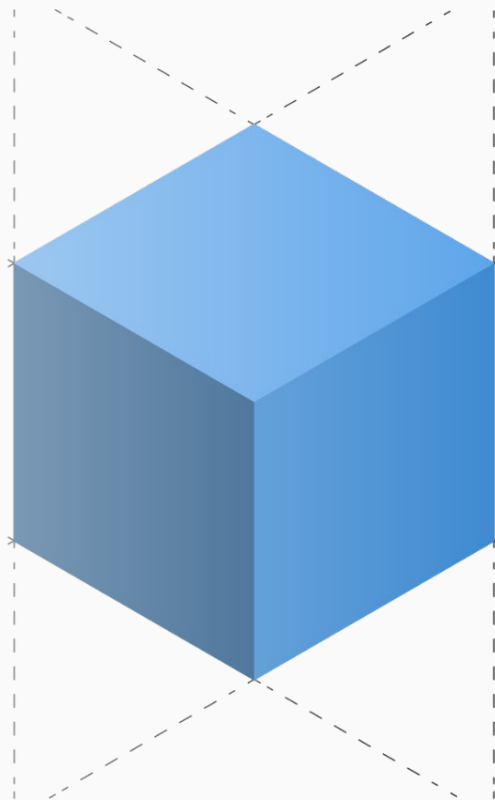
TiDB is an open-source distributed SQL database designed to handle both transactional (OLTP) and analytical (OLAP) workloads efficiently, making it a versatile solution for large-scale applications. Here's a structured overview of TiDB:

...



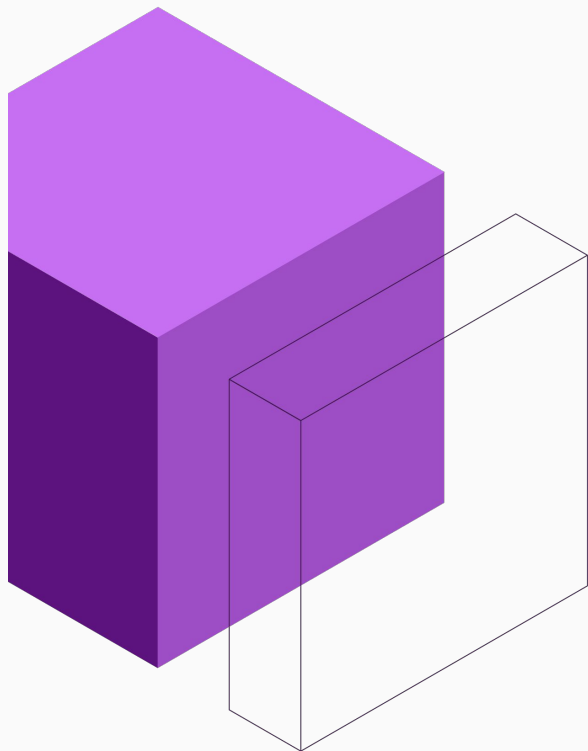
Semantic search

- **Search what I mean, not what I say!**
- **Not using literal words for searching, more likely to include automobile when searching for car.**
- **Can use a multi dimensional Vector to representation the question / document / object.**
- **Using tailored models for generating vectors.**





Vector embeddings



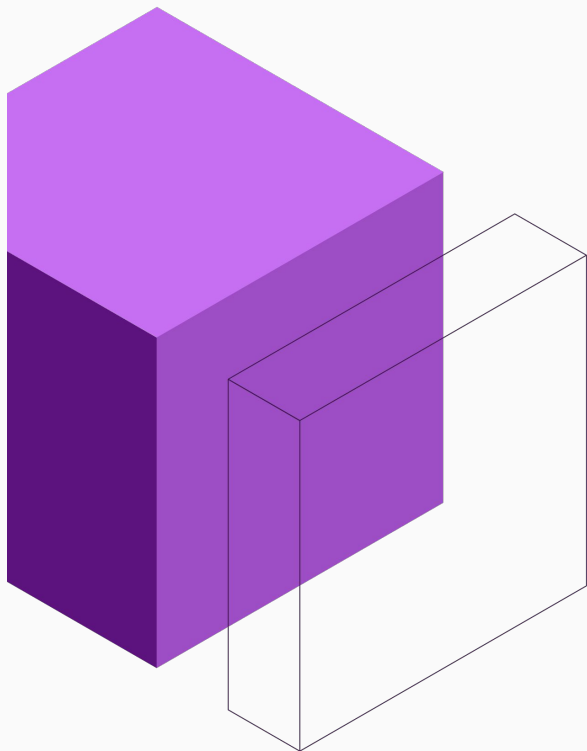
Vectors are stored as referencing or together with its objects (text etc.) and are then called embeddings.

These vector embeddings are then used for matching against similar objects.

Can have more than 1000 dimensions, but always a fixed number, depending on the model.



Vector embeddings

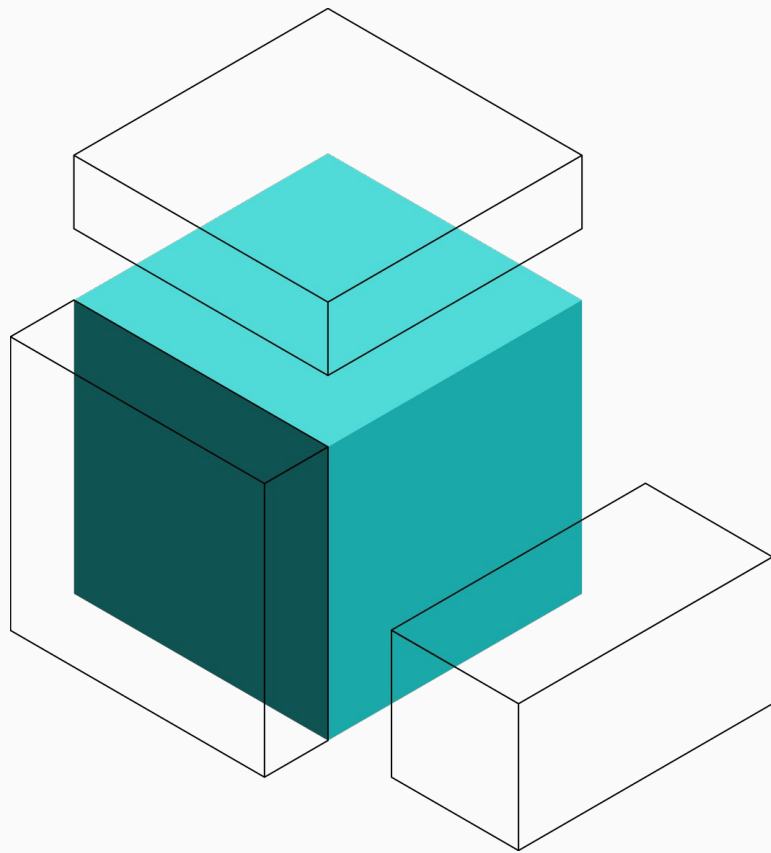


```
mysql> CREATE TABLE vector_table(embedding VECTOR);
Query OK, 0 rows affected (0.05 sec)
mysql> INSERT INTO vector_table VALUES
  -> ('[5.3, 6.2, 4.7, 9.4, 3.2]'),
  -> ('[7.4, 8.3, 3.6, 9.5, 1.5]'),
  -> ('[1.6, 5.3, 3.9, 4.9, 3.4]'),
  -> ('[4.6, 6.2, 2.9, 5.5, 2.4]'),
  -> ('[8.2, 2.7, 5.9, 4.5, 1.1]');
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT
  -> embedding,
  -> VEC_Cosine_Distance(embedding, '[1,2,3,4,5]') AS d
  -> FROM vector_table
  -> ORDER BY d;
+-----+-----+
| embedding          | d          |
+-----+-----+
| [1.6,5.3,3.9,4.9,3.4] | 0.09597214606787163 |
| [5.3,6.2,4.7,9.4,3.2] | 0.15841034048519986 |
| [4.6,6.2,2.9,5.5,2.4] | 0.21071371150541895 |
| [7.4,8.3,3.6,9.5,1.5] | 0.28466052143741205 |
| [8.2,2.7,5.9,4.5,1.1] | 0.35390651635892556 |
+-----+-----+
5 rows in set (0.02 sec)
```

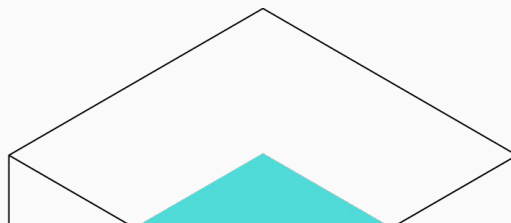
Vector Search

- Find closest neighbour vector
- Small dataset can use brute force
- Larger datasets needs some kind of index
- Often approximate, since very resource intensive for exact search





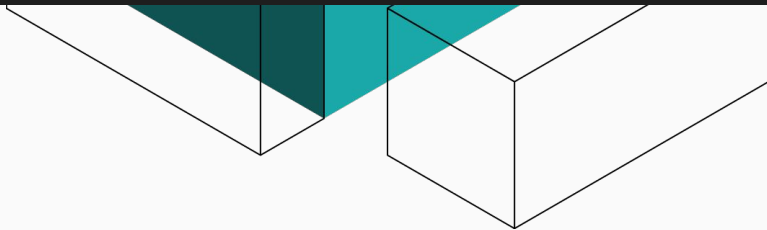
Vector Search



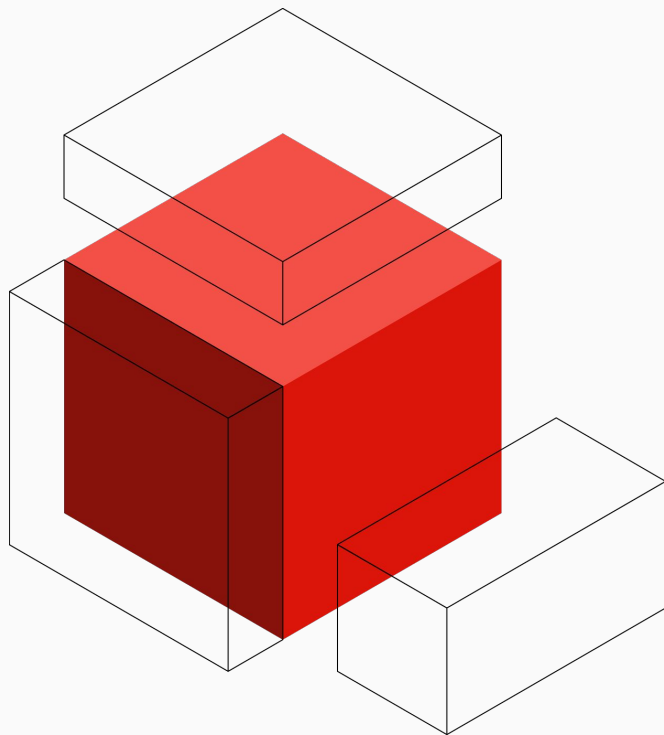
```
AIDemo> explain select * from entity order by VEC_L2_DISTANCE(content_vec, '[-0.030583884567022324, -0.045540183782577515, ..., 0.025173302739858627, 0.021252980455756187]') limit 2;
```

id	estRows	task	access object	operator info
Projection_28	2.00	root		aidemo.entity.id, aidemo.entity.content, aidemo.entity.content_vec
└TopN_10	2.00	root		Column#5, offset:0, count:2
└Projection_29	2.00	root		aidemo.entity.id, aidemo.entity.content, aidemo.entity.content_vec, vec_l2_dis
└TableReader_25	2.00	root		MppVersion: 1, data:ExchangeSender_24
└ExchangeSender_24	2.00	mpp[tiflash]		ExchangeType: PassThrough
└Projection_26	2.00	mpp[tiflash]		aidemo.entity.id, aidemo.entity.content, aidemo.entity.content_vec
└TopN_23	2.00	mpp[tiflash]		Column#4, offset:0, count:2
└Projection_27	2.00	mpp[tiflash]		aidemo.entity.id, aidemo.entity.content, aidemo.entity.content_vec, vec_l2_dis
└TableFullScan_22	2.00	mpp[tiflash]	table:entity, index:idx_vec(content_vec)	keep order:false, stats:pseudo, annIndex:L2(content_vec.[-0.031,-0.046,0.0017

9 rows in set (0.07 sec)



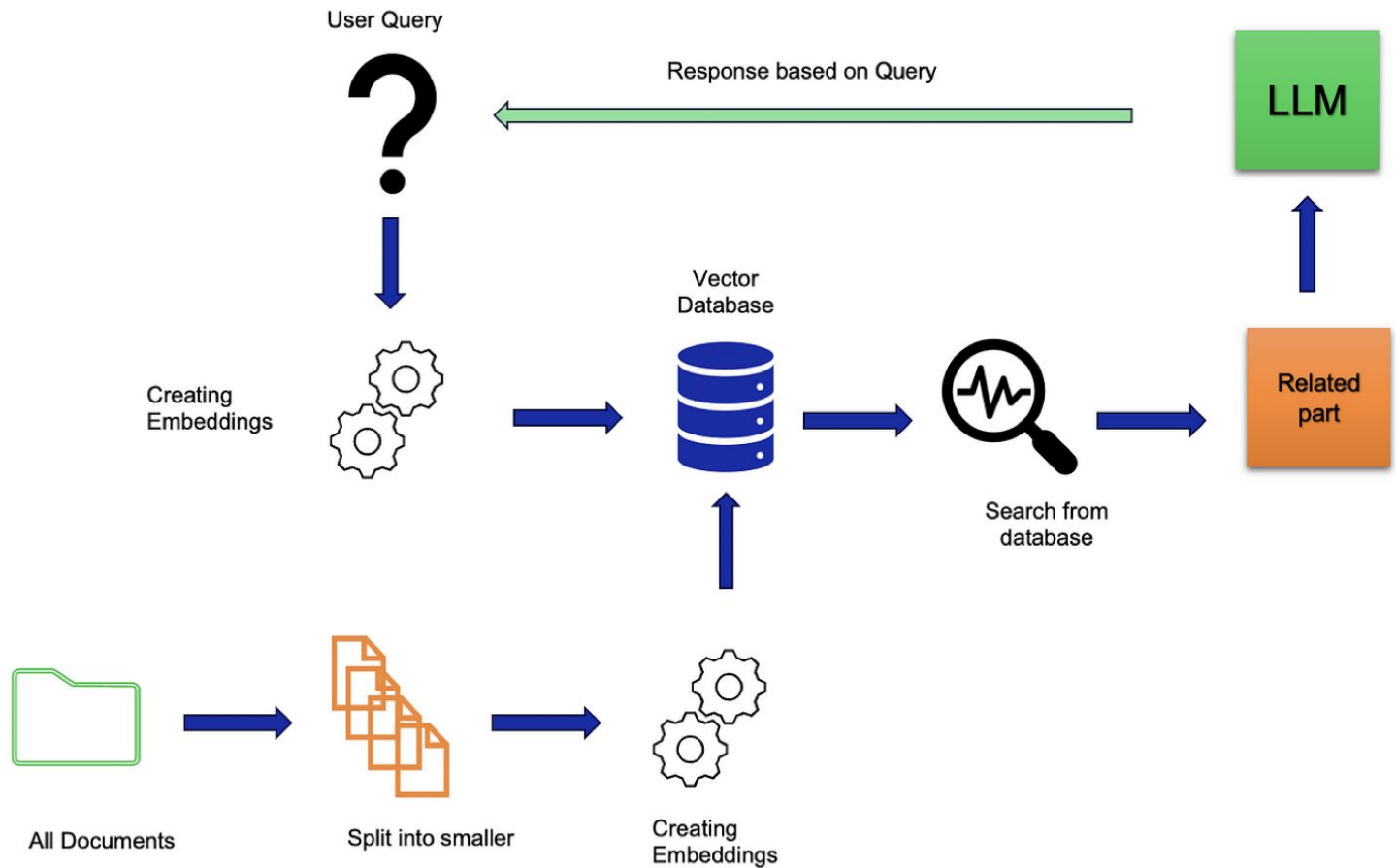
Putting it all together



What we have:

- LLM to generate nice answers from questions and context
- Documents/objects with knowledge/context about our area
- Models to map objects to vectors
- Vector search for finding closest matching objects
- Use the result as added context

=> RAG – Retrieval Augmented Generation



AGENDA

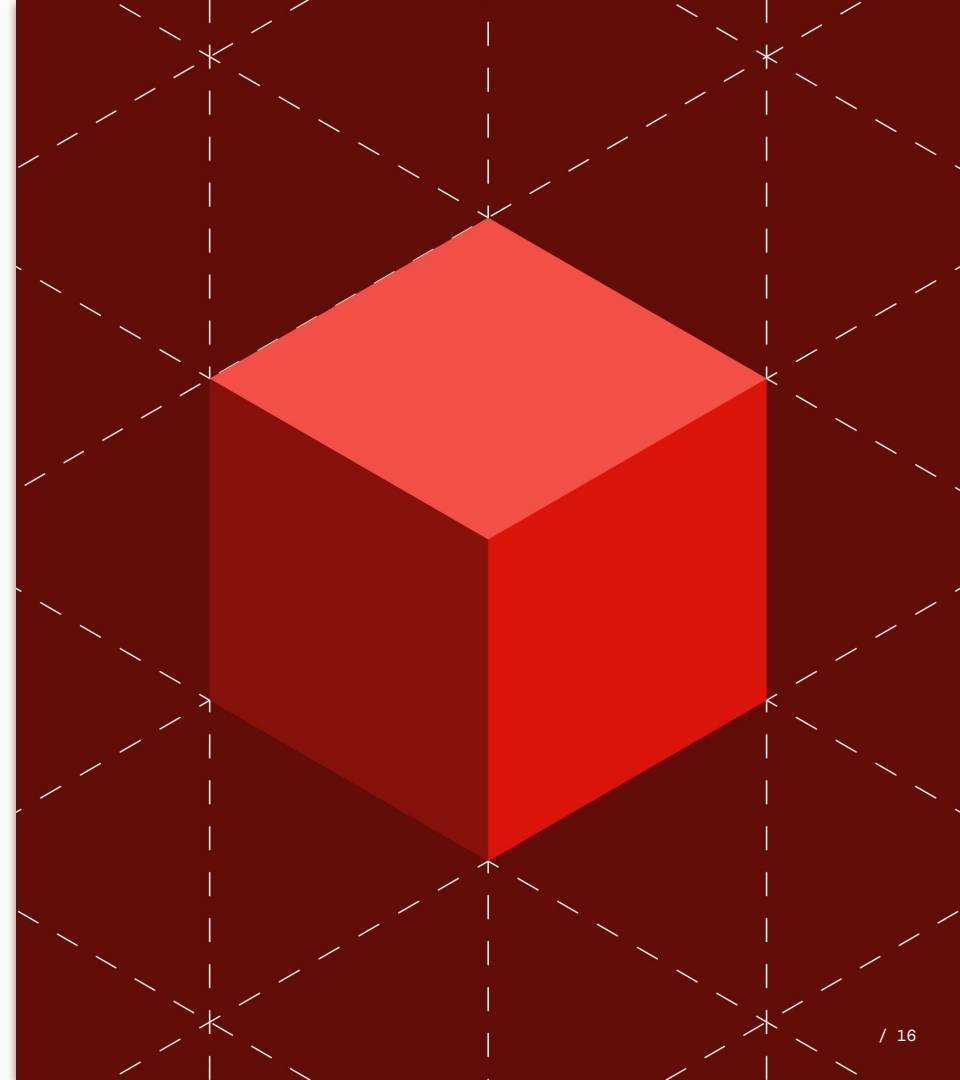
MySQL and compatible DBs

Vector datatype

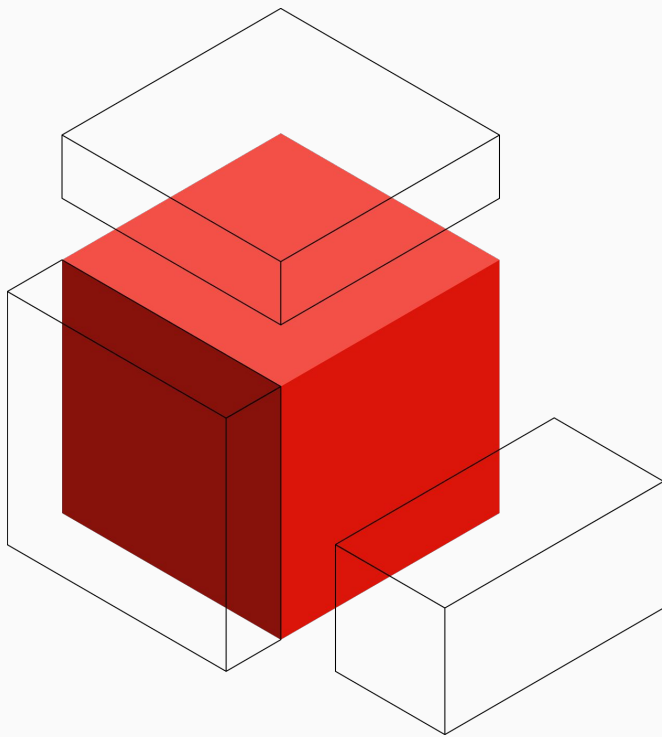
Vector functions

- **To/From string**
- **Distance**

Indexing on Distance



Vector Datatype

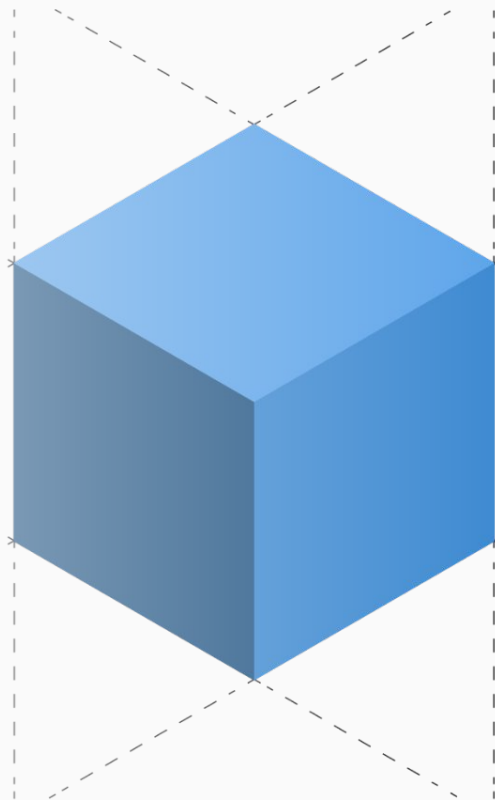


- Embeds vector into a column
- Normally a fixed number of dimensions
- Can be NULL
- Normally one 32-bit float per dimension
- Only useful as a whole vector, individual values are not queried.
- Compared with other vectors from same model for similarity



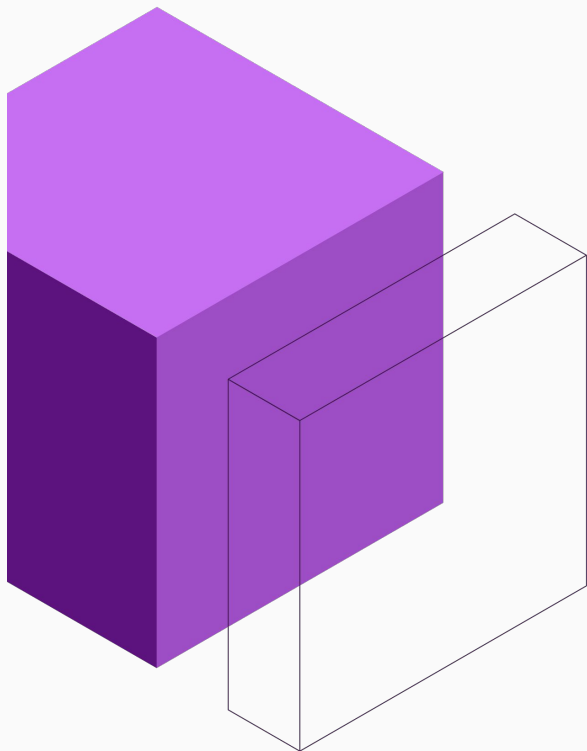
Vector parsing - INSERT INTO t VALUES (...)

- **Implicit**
 - '[1,2.4,0.9]'
- **Explicit**
 - `STRING_TO_VECTOR('[1,2]')`
 - alias `TO_VECTOR('[1,2]')`
 - `MYVECTOR_CONSTRUCT('[1,2]')`
 - `VEC_FromText('[1,2]')`
 - `VEC_FROM_TEXT('[1,2]')`
- **Binary**
 - `x'0000803F9A9919406666663F'`
 - `_binary '...'`





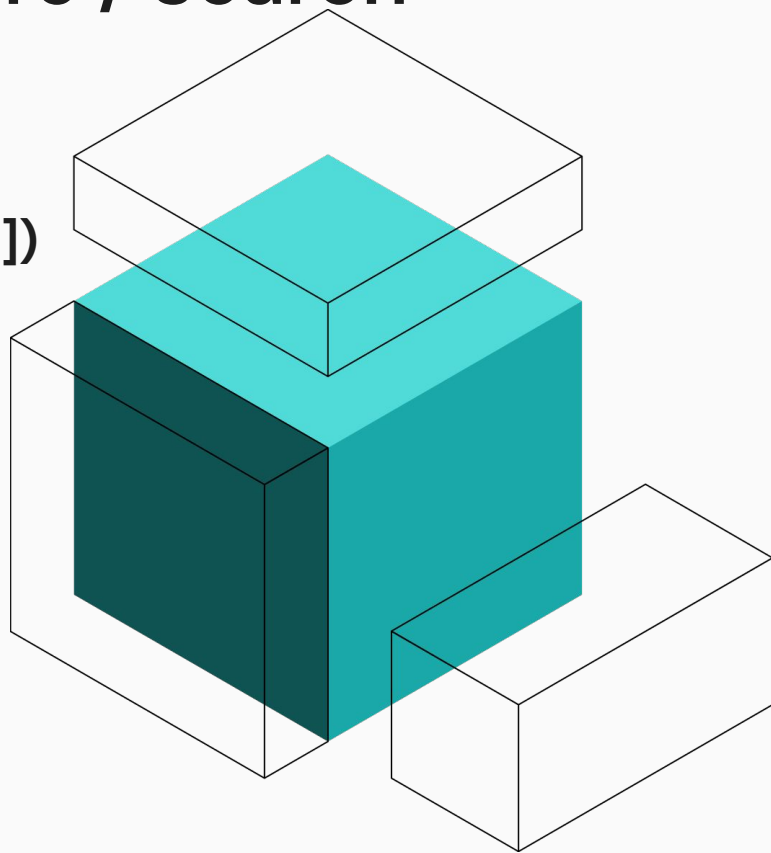
Vector output – SELECT vec_col FROM t



- **Direct**
 - `vec_col -> _binary '...'`
 - `vec_col -> 0x0000803F9A9919406666663F`
 - `vec_col -> '[1,2.4,0.9]'`
- **Convert**
 - `VECTOR_TO_STRING(vec_col)`
 - alias `FROM_VECTOR(vec_col)`
 - `MYVECTOR_DISPLAY(vec_col)`
 - `VEC_ToText(vec_col)`
 - `VEC_AS_TEXT(vec_col)`
 - `CAST(vec_col AS CHAR)`

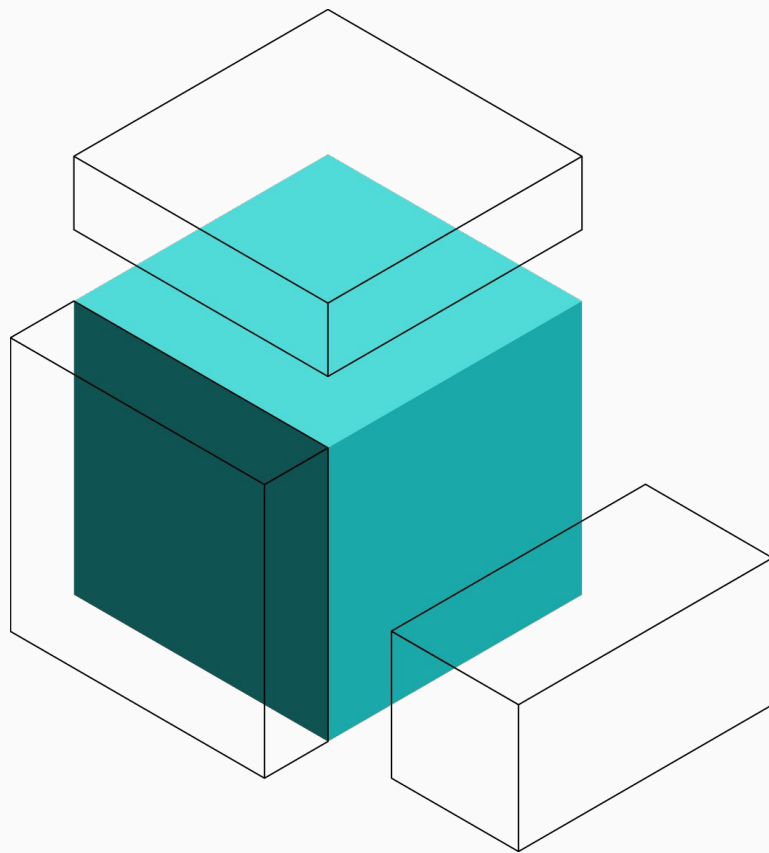
Vector Distance - Compare / Search

- Cosine, Euclidean/L2, DOT, ...
- `DISTANCE(v1,v2,metric)`
- `MYVECTOR_DISTANCE(v1,v2[,metric])`
- `VEC_DISTANCE_EUCLIDEAN(v1,v2)`
- `VEC_DISTANCE_COSINE(v1,v2)`
- `VEC_DISTANCE(v1,v2)`
- `VEC_L2_DISTANCE(v1,v2)`
- `VEC_COSINE_DISTANCE(v1,v2)`

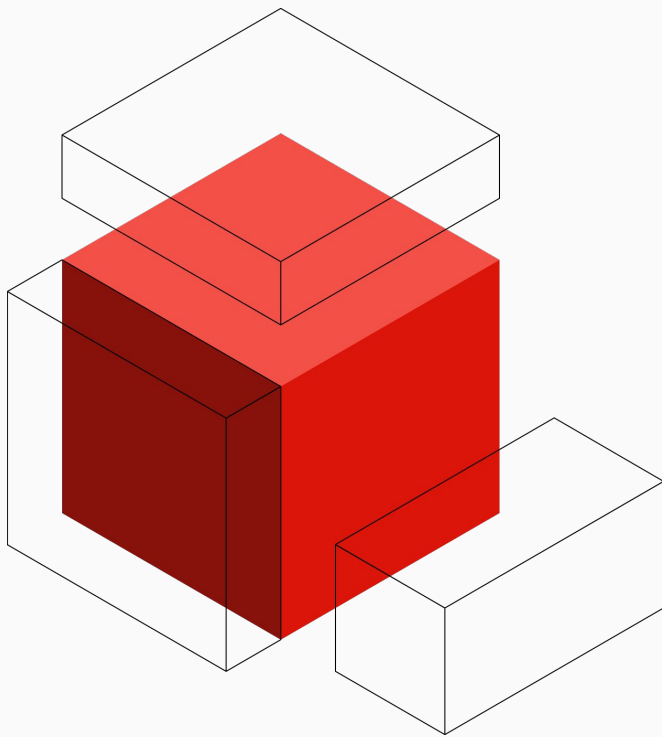


Vector – Utility functions

- `VECTOR_DIM()` / `VEC_DIMS()` – number of dimensions of the vector
- `VEC_L2_NORM()` – Euclidean norm



Vector Index



- Supported?
- transactional/async/manually updated?
- Configurable more than just distance type?
- VECTOR INDEX (v) [M=m DISTANCE=d]
- VECTOR INDEX ((VEC_COSINE_DISTANCE(v))
[USING HNSW])
- Custom
 - call `mysql.myvector_index_build(...)`
 - `MYVECTOR_SEARCH`

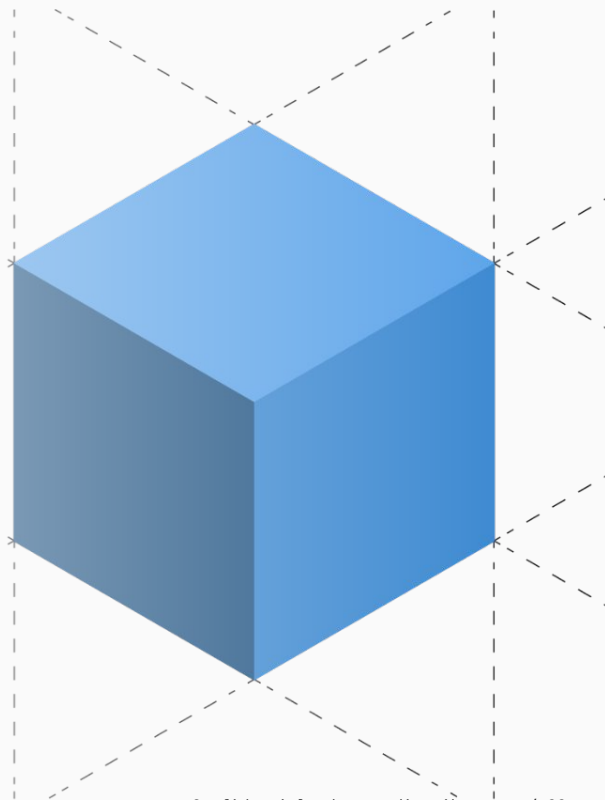


MySQL

VECTOR[(M)]

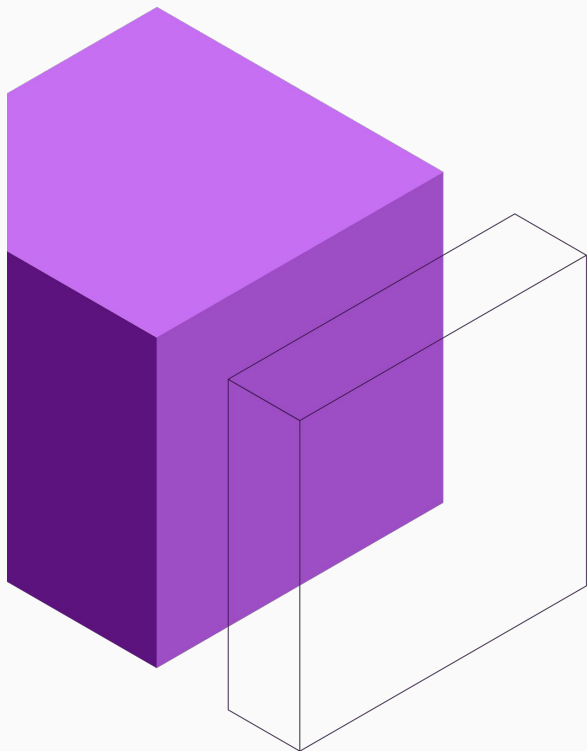
- OK without specifying dimensions -> default 2048
- Only max M dimensions!
 - Accepts < M dimensions!
- `STRING_TO_VECTOR()/TO_VECTOR()`
- `VECTOR_TO_STRING/FROM_VECTOR()`
- `VEC_DIM(vec_col)` utility func
- No `DISTANCE()` function!
- No `VECTOR INDEX!`

So you can only store and retrieve vectors out of the box in MySQL open source versions!





MySQL + myvector (adds the missing stuff!)

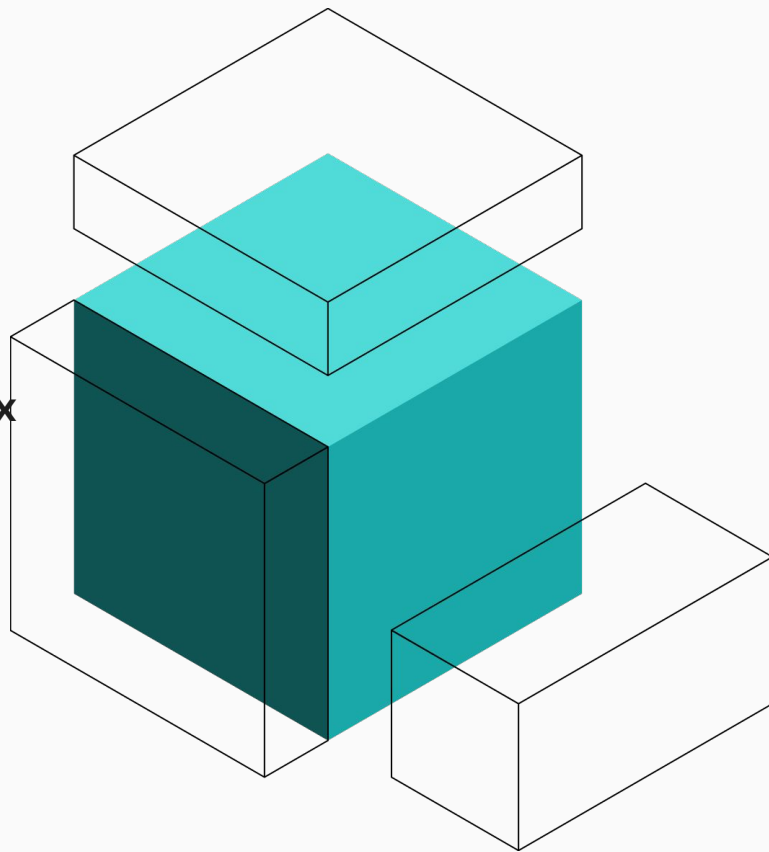


- Supports MySQL VECTOR datatype (\geq MySQL 9.0)
 - Own MYVECTOR(<options>) datatype ($<$ 9.0)
- Using audit_plugin / query rewrite
- Adds MYVECTOR_DISTANCE(v1,v2,type)
- In-memory vector index, manually managed through MYVECTOR_SEARCH_ADMIN()
- MYVECTOR_IS_ANN()
- MYVECTOR_SEARCH()
- MYVECTOR_DISTANCE_ANN()

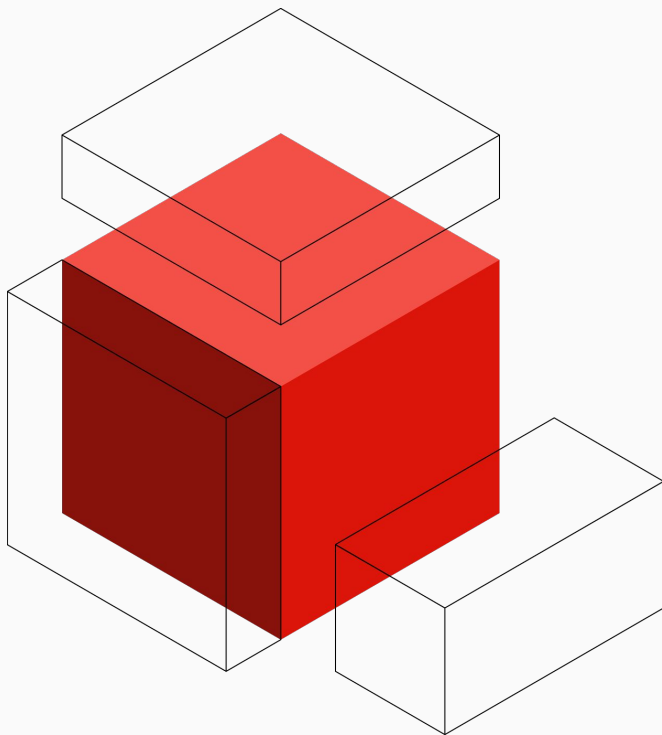
- See next presentation!

MariaDB 11.7+

- VECTOR(M) datatype
- VEC_FromText()/VEC_ToText()
- VEC_DISTANCE_COSINE(v1,v2)
- VEC_DISTANCE_EUCLIDEAN(v1,v2)
- VEC_DISTANCE(v1,v2)
 - Using the first matching vector index distance type. 11.8
- VECTOR INDEX
- Engine independent!



TiDB



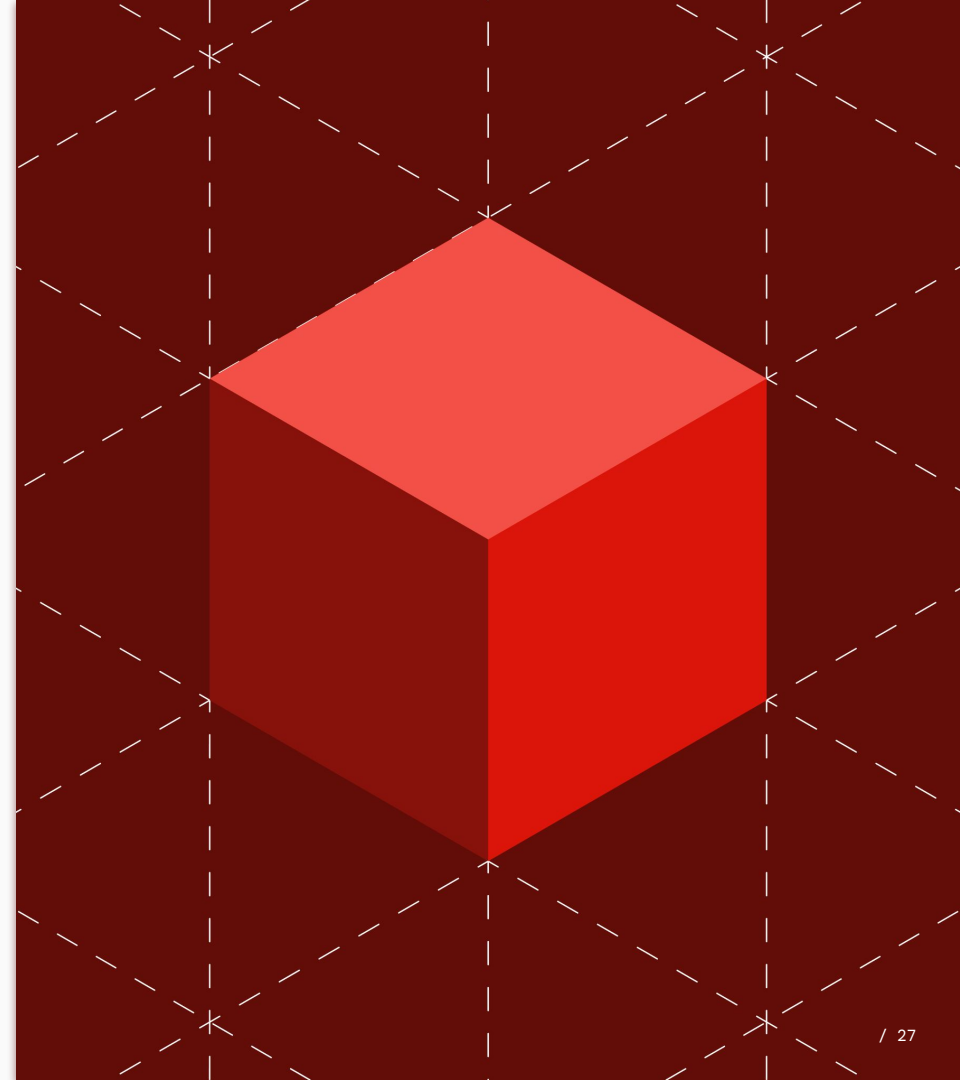
- VECTOR datatype[(M)]
 - Both fixed M and non-fixed
- Implicit conversion to/from vector
- VEC_FROM_TEXT()
- VEC_AS_TEXT()
- VEC_L2_NORM()
- VEC_L2_DISTANCE() - Euclidean
- VEC_COSINE_DISTANCE()
- VEC_NEGATIVE_INNER_PRODUCT()
- VEC_L1_DISTANCE()
- VEC_DIMS()
- Extra features:
 - Comparison
 - Arithmetic

AGENDA

Can we agree on the same
function names?

To keep the MySQL community
compatible :)

I propose:
`VECTOR_DISTANCE(v1,v2,<type>)`
`CAST(str AS VECTOR)`





Other resources

- <https://github.com/pingcap/autoflow>
 - AutoFlow is an open source graph rag. Create your own AI Assistant / conversational search page!
- <https://github.com/mjonss/mysql-vector-compare> small utilities/hacks for comparing vector implementations between MySQL/MySQL+MYVECTOR/MariaDB/TiDB for this presentation.
- Mattias.Jonsson@PingCAP.com
- <https://www.linkedin.com/in/mjonss/>