WeDataLab

벡터 데이터베이스 실습

2025년 03월

대표적인 벡터 데이터베이스

특징	ChromaDB	Pinecone	Weaviate	Qdrant
오픈소스	✅	❌	✅	✅
Python 친화성	✅	✅	✅	✅
분산 지원	✅	✅	✅	✅
실시간 업데이트	✅	❌	✅	✅
커뮤니티 지원	활발	제한적	활발	활발

출처 1 : https://www.rustfinity.com/open-source/qdrant
출처 2 : https://myscale.com/blog/qdrant-vs-chroma-vector-databases-comparison/
출처 3 : https://www.datacamp.com/blog/the-top-5-vector-databases

목차

ChromaDB 실습

ChromaDB

- 2022년: Chroma 프로젝트 시작. 생성형 AI와 LLM의 성장에 따라 벡터 데이터베이스의 필요성에 대응해 개발 착수.
- 2023년: ChromaDB 오픈소스 출시**. LangChain 및 OpenAI API와 통합**되며 생성형 AI 애플리케이션에서 주목받음.
- 2023년: 대규모 데이터셋 처리 및 실시간 검색 기능 강화, 커뮤니티 성장과 GitHub에서 활발한 활동 진행.
- 2024년 이후 (예상): 실시간 데이터 처리, 멀티모달 데이터 지원 등 기능 확장과 클라우드 기반 확장성 강화 전망.

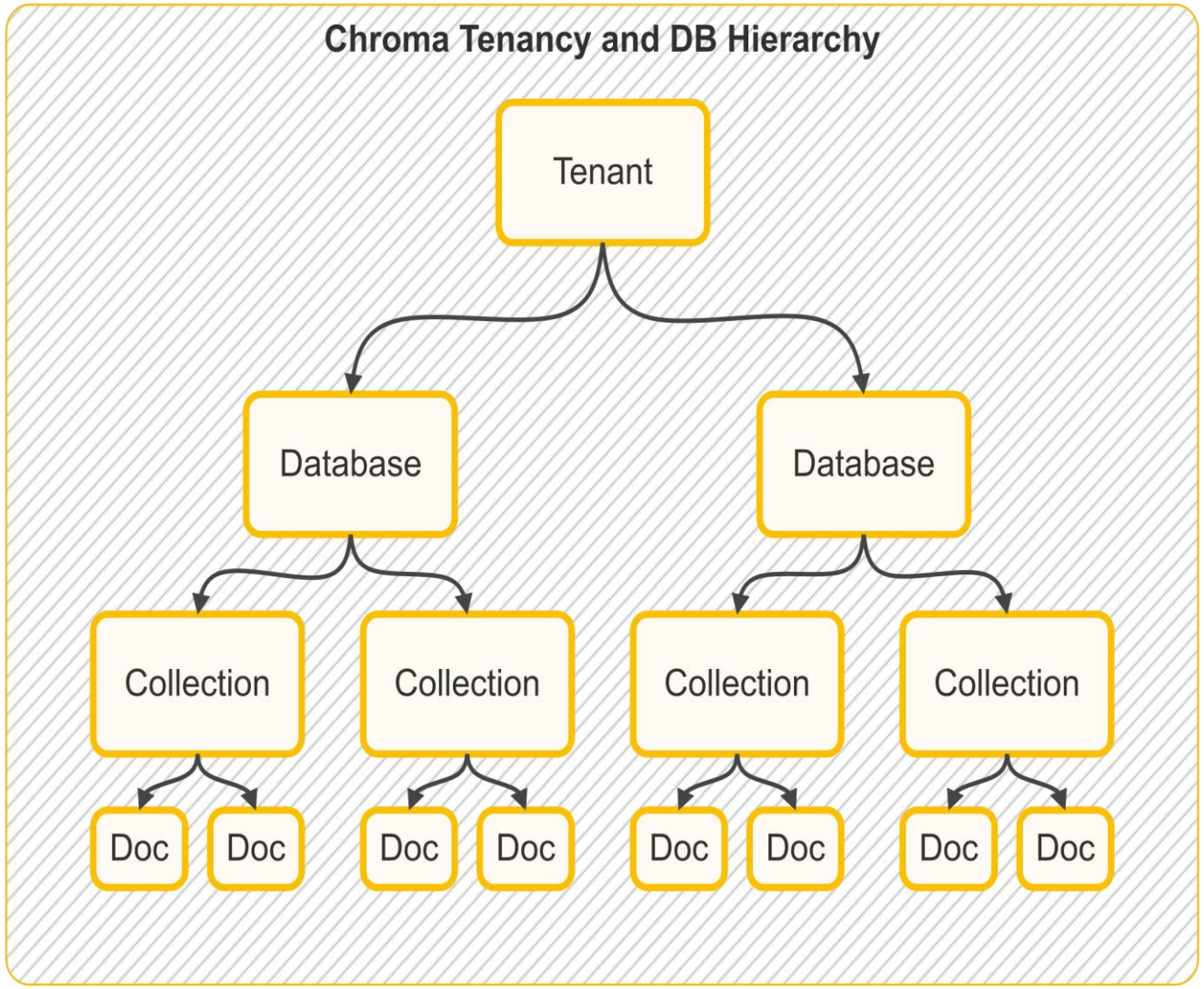
특징	설명
빠른 성능	임베딩 저장 및 검색이 매우 빠르고 효율적
파이썬 네이티브	파이썬으로 개발되어 파이썬 생태계와의 통합이 용이
멀티테넌시	기업 환경에 적합한 멀티테넌트 아키텍처 지원
메타데이터 필터링	강력한 메타데이터 필터링 기능 제공
REST API	API를 통한 접근 가능
로컬 실행	로컬 환경에서 실행 가능하여 개발/테스트 용이
오픈소스	무료로 사용 가능한 오픈소스 라이선스

ChromaDB 구성

•Tenant (테넌트)
- ✓ 가장 상위 수준의 격리 단위
- ✓ 다수의 데이터베이스
- ✓ 조직이나 사용자 그룹 분리

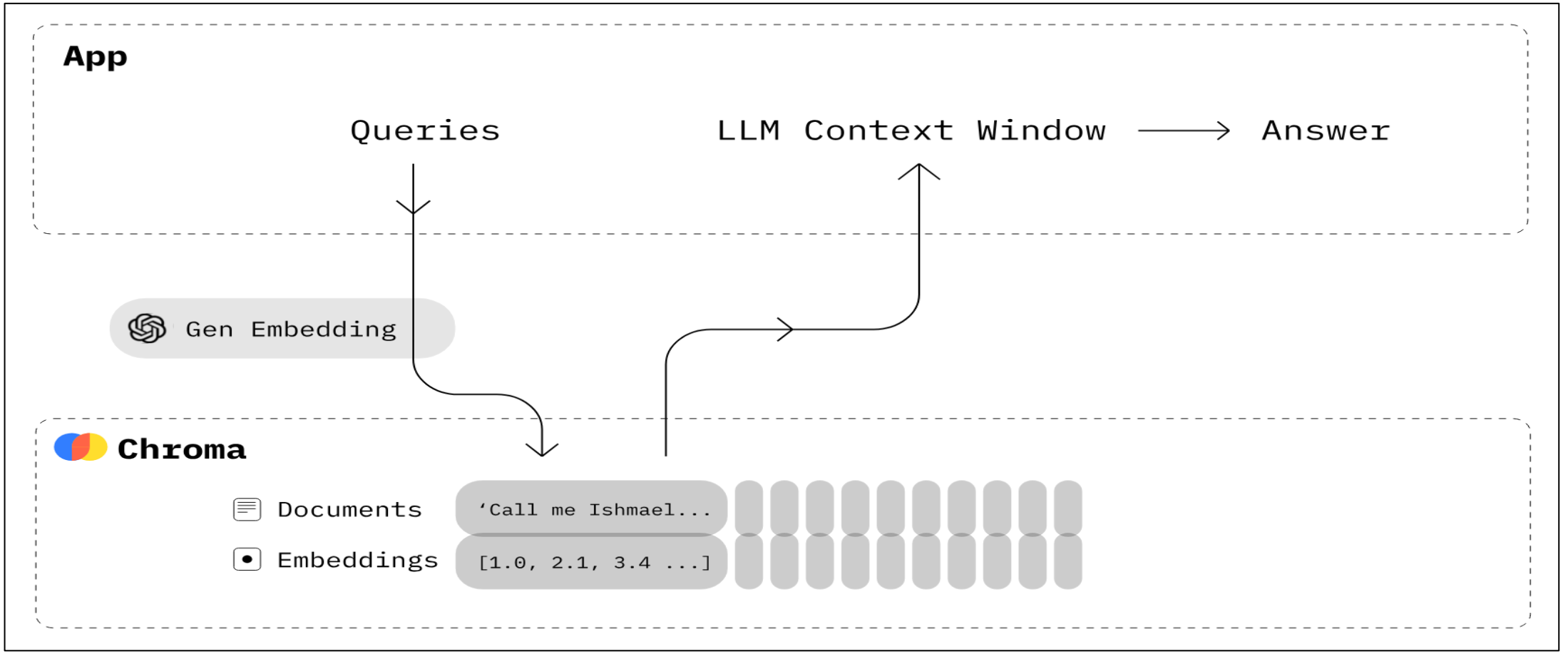
•Database (데이터베이스)
- ✓ 테넌트 내에 논리적 컨테이너
- ✓ 독립적인 collection 포함

•Collection (컬렉션)
- ✓ 임베딩과 메타데이터를 저장하는 기본 단위
- ✓ 유사한 특성을 가진 문서나 데이터를 그룹화
- ✓ 고유한 스키마와 설정
- ✓ 벡터 검색, 필터링, CRUD 작업 수행



출처 : https://cookbook.chromadb.dev/core/concepts/#tenancy-and-db-hierarchies

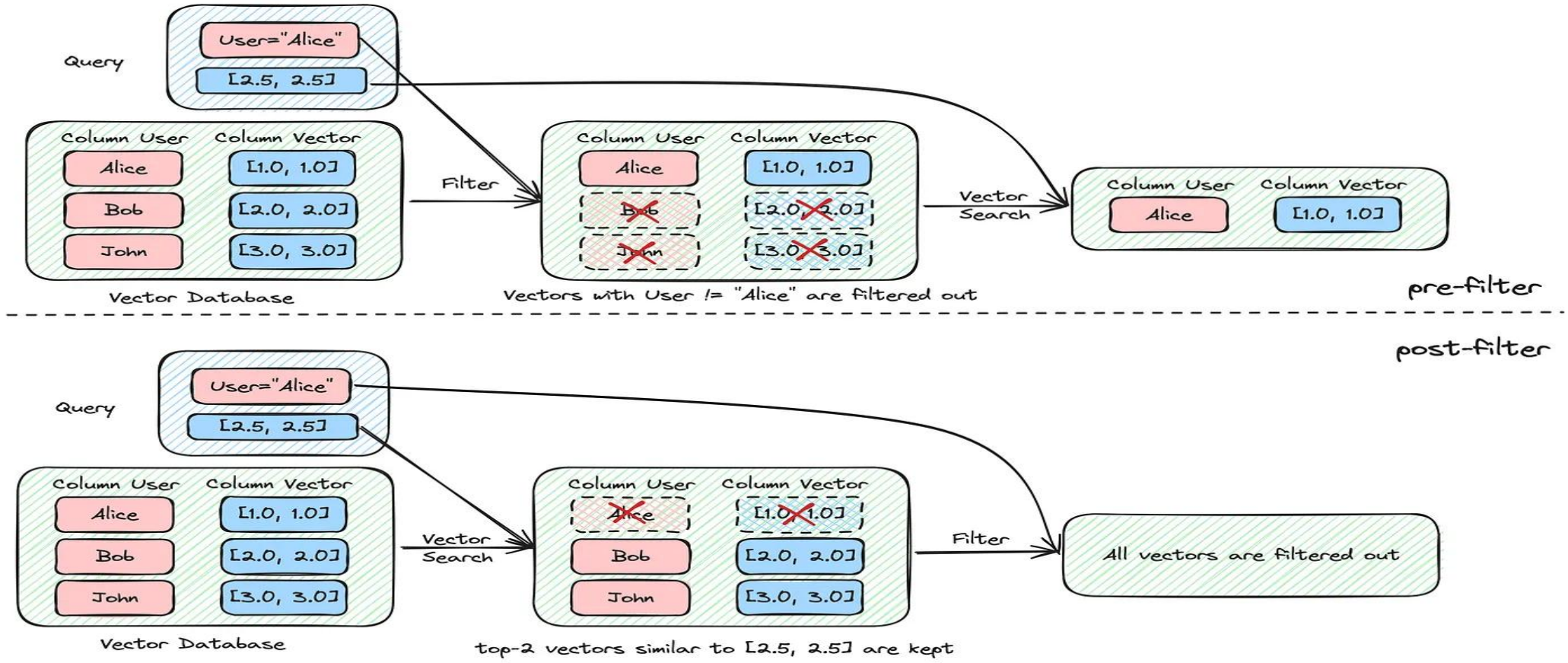
ChromaDB 실습



출처 : https://opentutorials.org/module/6369

ChromaDB + Embedding 실습
실습 : https://colab.research.google.com/drive/1QIv8Te_U_-X0C7YFvfxiW2doddVSoP7p?usp=sharing

ChromaDB 실습



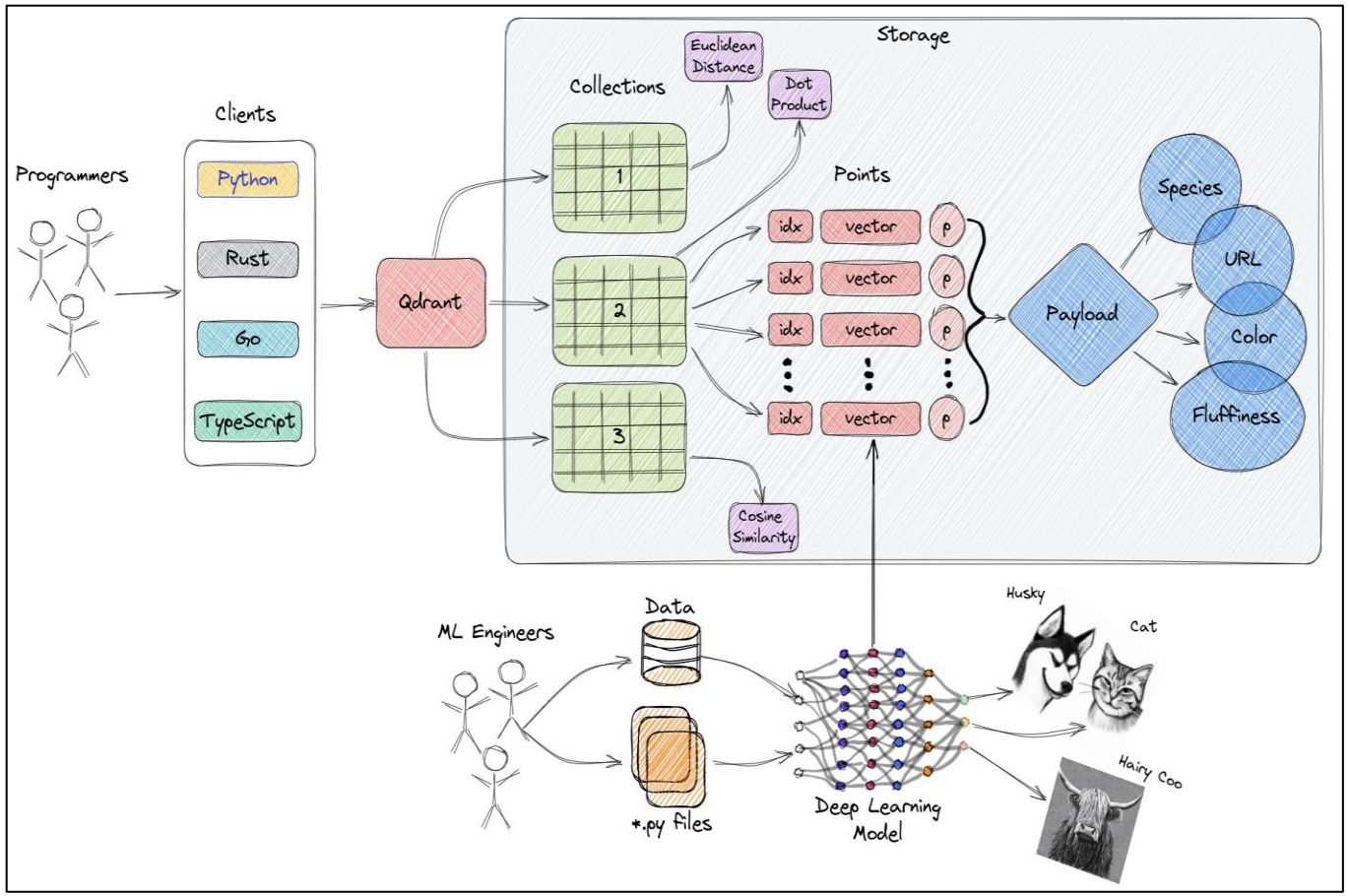
출처 : https://medium.com/@myscale/optimizing-filtered-vector-search-in-myscale-77675aaa849c

ChromaDB + 필터 실습
실습 : https://colab.research.google.com/drive/1isvHKC4s6dZHxEhZVZEEX8J1natTYlvF?usp=sharing

Qdrant 실습

Qdrant 실습

- Collection
  - ✓ 벡터 데이터를 저장하는 독립된 논리적 컨테이너
  - ✓ 데이터 분할 및 관리를 위한 단위
- Point
  - ✓ 고유 ID를 가진 벡터 및 관련 메타데이터
  - ✓ 벡터 값, 페이로드 및 다양한 메타데이터 포함
- Vector
  - ✓ 이미지, 텍스트, 오디오 등이 임베딩 된 벡터
  - ✓ 유사성 검색 및 군집화에 사용
- Payload
  - ✓ 벡터에 연결된 메타데이터로 검색에 사용 가능
  - ✓ 키-값 쌍 형식으로 저장
- Indexing
  - ✓ 검색 성능 향상을 위해 벡터 데이터 인덱싱
  - ✓ 다양한 인덱싱 알고리즘 사용 가능
- Similarity Search
  - ✓ 주어진 벡터와 가장 유사한 벡터 찾기
  - ✓ 거리 메트릭(L2 거리 등)을 사용하여 유사도 계산



출처 : https://qdrant.tech/documentation/overview/

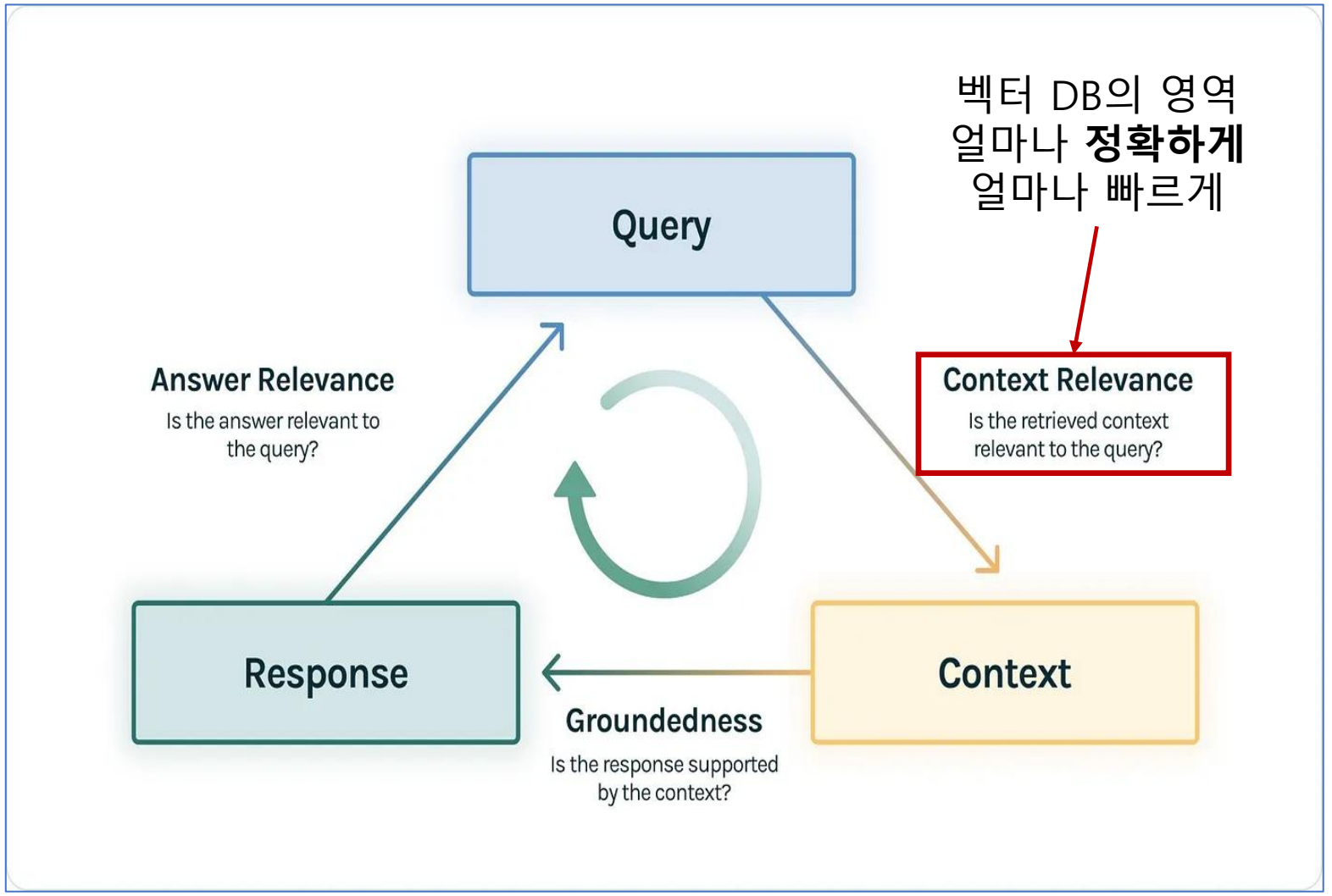
Qdrant filtering
실습 : https://colab.research.google.com/drive/1hv1PHd55p_wK49ACVj4hh9nZ_Cc_8boD?usp=sharing

RAG : 검색을 기반으로 한 답변 생성

- Context Relevance
  - ✓ Query-Contexts
- Groundedness
  - ✓ Contexts-Response
- Answer Relevance
  - ✓ Query-Response

➔ 잘못된 문맥은 잘못된 답변을 생성

➔ **검색된 문맥이 답변의 기반**



출처 : https://medium.com/@rajib76.gcp/rag-triad-introduction-4e5ecba26741

Vector Database : Relevancy based metrics

Relevancy based metrics

Precision@k

The proportion of relevant documents among the top k results

$$precision@k = \frac{|relevant\ documents\ in\ the\ top\ k\ results|}{k}$$

Recall@k

The proportion of relevant documents that are retrieved among the top k results in relation to all relevant documents in the dataset

$$recall@k = \frac{|relevant\ documents\ in\ the\ top\ k\ results|}{|all\ relevant\ documents|}$$

Precision@k

- 상위 k개 결과 중 관련 있는 결과의 비율
- 정확한 검색 능력 평가
- 예시: 상위 5개 결과 중 3개가 관련 있다면,

  Precision@5 = 3/5 = 0.6

Recall@k

- 전체 관련 결과 중 상위 k개에 포함된 관련 결과의 비율
- 관련 있는 결과를 검색 능력 평가
- 예시: 전체 관련 결과 20개 중

  상위 10개에 7개가 포함되었다면,

  Recall@10 = 7/20 = 0.35

Vector Database : Relevancy based metrics

Ranking related metrics

Mean Reciprocal Rank (MRR)

It is based on the position of the first relevant item in the list of results.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

$rank_i$ is a position of the first relevant item.

Application: It is used to assess the effectiveness of a system to return highly relevant results as early as possible.

MRR (Mean Reciprocal Rank : 평균 역순위)

- 첫 번째 관련 결과의 역순위의 평균값
- 값이 클수록 좋은 성능을 나타냄
- 예시: 질의에 대해 1번째 결과가 관련 있다면

  MRR = 1, 2번째 결과가 관련 있다면 MRR = 0.5

Vector Database : Relevancy based metrics

Score related metrics

Discounted Cumulative Gain (DCG)

It gives more weight to documents appearing earlier in the list and thereby reflecting the diminishing relevance through a logarithmic scale for the position.

$$DCG@k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}$$

DCG (Discounted Cumulative Gain : 할인 누적 이득)

•상위 순위 관련 결과에 더 많은 가중치 부여

•로그 기반 가중치 할당으로 상위 순위 중요도 강조

•예: rel(1)=3, rel(2)=2, rel(3)=0, rel(4)=1일 때

DCG = 3 + 2/log2(3) + 0 + 1/log2(5) = 5.63

Vector Database : Relevancy based metrics

Score related metrics

Normalized Discounted Cumulative Gain (nDCG)

If we have the relevancy scores in our ground truth, we can calculate the Ideal Discounted Cumulative Gain, with the same formula as DCG.

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

nDCG (Normalized Discounted Cumulative Gain

정규 할인 누적 이득)

•DCG를 최적의 DCG로 정규화한 값

•0과 1 사이의 값을 가짐 (1이 최적)

•예: DCG=5.63, 최적 DCG=7.94일 때,

nDCG = 5.63/7.94 = 0.71

Qdrant 실습(Demo) : Search Relevance

ranx

```python
from ranx import Qrels, Run

qrels_dict = { "q_1": { "d_12": 5, "d_25": 3 },
               "q_2": { "d_11": 6, "d_22": 1 } }

run_dict = { "q_1": { "d_12": 0.9, "d_23": 0.8, "d_25": 0.7,
                      "d_36": 0.6, "d_32": 0.5, "d_35": 0.4  },
             "q_2": { "d_12": 0.9, "d_11": 0.8, "d_25": 0.7,
                      "d_36": 0.6, "d_22": 0.5, "d_35": 0.4  } }

qrels = Qrels(qrels_dict)
run = Run(run_dict)
```

ranx is a Python library for evaluating ranking metrics. Two major data types for ranx are:

- `Qrels`, or query relevance judgments
- `Run`, an output of the retrieval system

Retrieval quality in RAG setup

Retrieval Augmented Generation is all about **extending prompts with relevant context** so the LLM can use it to formulate the correct answer.

Each RAG might be **only as good, as the retrieval component.**

It is also the easiest to evaluate, as end-to-end testing usually involves another LLM.

Retrieval quality evaluation should be a part of each mature RAG pipeline.

Qdrant 검색 연관성 측정
실습 : https://colab.research.google.com/drive/1Q-xwK1OdXy6yGsoz3cqbOz0Z6_ZUiKif?usp=sharing

Weaviate 실습

Weaviate

- 2018년: 네덜란드의 'Semi Technologies 에서 비정형 데이터 관리를 위한 Weaviate 프로젝트 시작
- 2019년: 첫 오픈소스 벡터 검색 데이터베이스 출시. NLP와 벡터 저장 중심.
- 2022년: Weaviate의 성능과 확장성 강화. 대규모 데이터셋 처리 및 새로운 인덱스 기술 도입.
- 2023년: 커뮤니티 성장 . RAG(검색 증강 생성) 기술과 통합된 애플리케이션 개발 활성화.
- 2024년: 다중 테넌시 기능 개선, 실시간 업데이트 및 멀티모달 데이터 지원 확대

특징	설명
확장성	수평적 확장이 가능한 분산 아키텍처 지원
GraphQL API	직관적인 GraphQL 인터페이스 제공
실시간 벡터화	데이터 입력 시 자동 벡터화 지원
다중 샤딩	대규모 데이터셋 처리를 위한 샤딩 지원
CRUD 작업	벡터와 객체에 대한 완전한 CRUD 작업 지원
멀티 테넌시	기업용 멀티 테넌트 지원
보안 기능	인증, 권한 관리 등 엔터프라이즈급 보안 제공
다양한 인덱싱 지원	여러 벡터 인덱스 백엔드 지원 (HNSW, LSH 등)

Weaviate 구성

- **클래스(Class):** 데이터의 카테고리 또는 타입

예: "Book", "Movie", "Product" 등.

- **속성(Properties):** 클래스 안에 저장되는 데이터의 세부 항목

예: 제목, 저자, 출판연도 등.

- **데이터 타입(DataType):** 각 속성에 저장되는데이터의 유형

예: 문자열(string), 숫자(int), 벡터(vector) 등

json

```
{
  "class": "Book",
  "properties": {
    "title": "1984",
    "author": "George Orwell",
    "publicationYear": 1949,
    "embedding": [0.12, 0.85, 0.33, 0.44]
  }
}
```

json 복사 편집

```
{
  "classes": [
    {
      "class": "Book",
      "description": "A collection of written works",
      "properties": [
        {
          "name": "title",
          "dataType": ["string"],
          "description": "The title of the book"
        },
        {
          "name": "author",
          "dataType": ["string"],
          "description": "The author of the book"
        },
        {
          "name": "publicationYear",
          "dataType": ["int"],
          "description": "The year the book was published"
        },
        {
          "name": "embedding",
          "dataType": ["vector"],
          "description": "Vector representation of the book's content"
        }
      ]
    }
  ]
}
```

Weaviate 실습

1. `with_near_text`

- **설명:** 텍스트 기반으로 데이터의 유사성을 평가.
- **사용 예:** 특정 개념과 의미적으로 유사한 데이터를 검색.
- **옵션 필드:**
  - `concepts`: 텍스트 벡터화의 기준이 되는 단어나 문장.
  - `distance`: (선택 사항) 결과와의 최대 거리.
  - `certainty`: (선택 사항) 검색 결과의 최소 유사도 (0~1 사이의 값).

예시:

python 복사 편집

```python
.with_near_text({
    "concepts": ["biology", "organisms"],
    "certainty": 0.8
})
```

2. `with_near_vector`

- **설명:** 벡터 데이터를 직접 사용해 유사성을 평가.
- **사용 예:** 특정 벡터와 가장 가까운 데이터를 검색.
- **옵션 필드:**
  - `vector`: 유사성 비교에 사용할 벡터 값.
  - `distance`: (선택 사항) 최대 허용 거리.
  - `certainty`: (선택 사항) 최소 유사도.

예시:

python

```python
.with_near_vector({
    "vector": [0.12, 0.56, 0.78],
    "distance": 0.1
})
```

3. `with_near_object`

- **설명:** 기존 Weaviate 객체의 ID를 기준으로 유사한 데이터를 검색.
- **사용 예:** 특정 객체와 유사한 데이터를 찾고 싶을 때.
- **옵션 필드:**
  - `id`: 기준이 되는 객체의 UUID.
  - `distance`: (선택 사항) 최대 허용 거리.
  - `certainty`: (선택 사항) 최소 유사도.

예시:

python

```python
.with_near_object({
    "id": "123e4567-e89b-12d3-a456-426614174000"
})
```

4. `with_near_image`

- **설명:** 이미지 데이터를 기준으로 유사한 이미지를 검색.
- **사용 예:** 업로드된 이미지와 유사한 이미지를 찾고 싶을 때.
- **옵션 필드:**
  - `image`: Base64로 인코딩된 이미지 데이터.
  - `distance`: (선택 사항) 최대 허용 거리.
  - `certainty`: (선택 사항) 최소 유사도.

예시:

python

```python
.with_near_image({
    "image": "..."
})
```

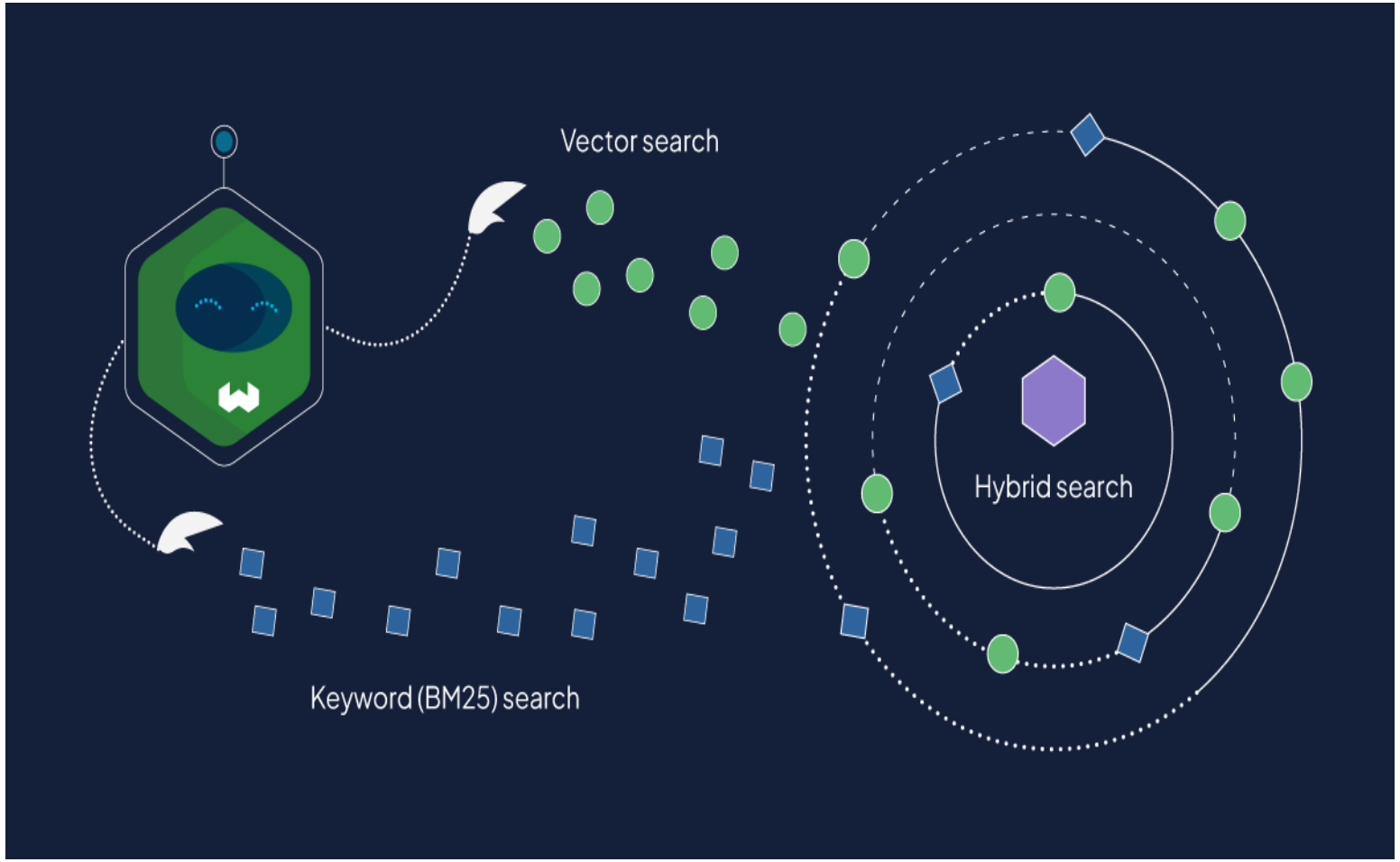
Weaviate CRUD
실습 링크 : https://colab.research.google.com/drive/1LEi1jMTwhNv2VDa4IBS7UTiVWrQAItwq?usp=sharing

Weaviate 실습

```
response = (
    client.query
    .get("JeopardyQuestion", ["question", "answer"])
    .with_hybrid(query="food", alpha=0.5)
    .with_limit(5)
    .do()
)
```

```
response = (
    client.query
    .get("JeopardyQuestion", ["question", "answer"])
    .with_near_text({"concepts": ["food"]})
    .with_limit(5)
    .do()
)
```

```
response = (
    client.query
    .get("JeopardyQuestion", ["question", "answer"])
    .with_bm25(query="food")
    .with_limit(5)
    .do()
)
```



출처 : https://weaviate.io/blog/hybrid-search-fusion-algorithms

Weaviate 하이브리드 검색
실습 : https://colab.research.google.com/drive/1VJnj1egMGILGrW7TSFBYAldD_bcDwyQL?usp=sharing

요 약

요약

1. Chroma:

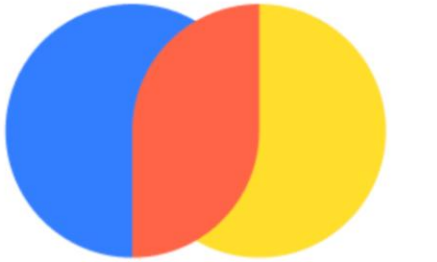
- **"Chroma"**는 그리스어에서 유래하며, **"색"**을 의미합니다.
- **색상**과 관련된 단어로, **다양한 색의 조합과 다양성**을 나타냅니다. 여기서 **Chroma**는 **다양한 데이터의 특성과 차이를 표현하는** 방식이나 **다채로운 정보**를 의미하는 데 사용됩니다.

2. DB:

- **"DB"**는 **Database**의 약어로, **데이터베이스**를 의미합니다.

ChromaDB의 의미:

- **"ChromaDB"**는 **다양한 특성을 지닌 데이터를 다루는 데이터베이스**라는 의미를 내포하고 있습니다. 특히 **다양한 형태의 데이터(텍스트, 이미지, 음성 등)를 처리하고, 그 속성들을 벡터화하여 관리**하는 데 초점을 둡니다.
- 또한, **Chroma**는 **색**을 뜻하기 때문에, **데이터의 "다채로운" 특성을 잘 포착하여 관리하는 시스템**이라는 점에서도 이 이름이 적합합니다.



Chroma

Weaviate의 이름은 **"we"**와 **"navigate"**의 결합으로 만들어졌습니다. 이 이름은 **"우리가 함께 데이터를 탐색하고, 연결한다"**는 의미를 담고 있습니다.

어원 및 의미:

- **"We"**: 사람들이 함께 협력하여 지식을 만들고 탐색하는 의미.
- **"Navigate"**: 데이터를 탐색하고, 그 안에서 필요한 정보를 찾아가는 과정을 의미.

따라서 **Weaviate**는 **"우리가 함께 데이터를 탐색하고, 이해하고, 연결해 나간다"**는 비전을 가진 벡터 데이터베이스라는 의미를 지니고 있습니다. 이는 **정보의 연결**과 **지식의 탐색**을 핵심으로 하는 **Semantic Web**이나 **AI 기반 데이터 처리**의 철학을 반영하는 이름입니다.

Weaviate

WeDataLab

감사합니다.