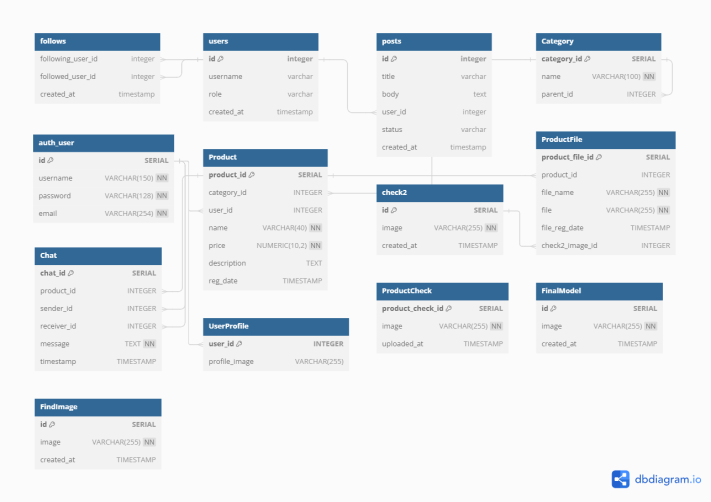
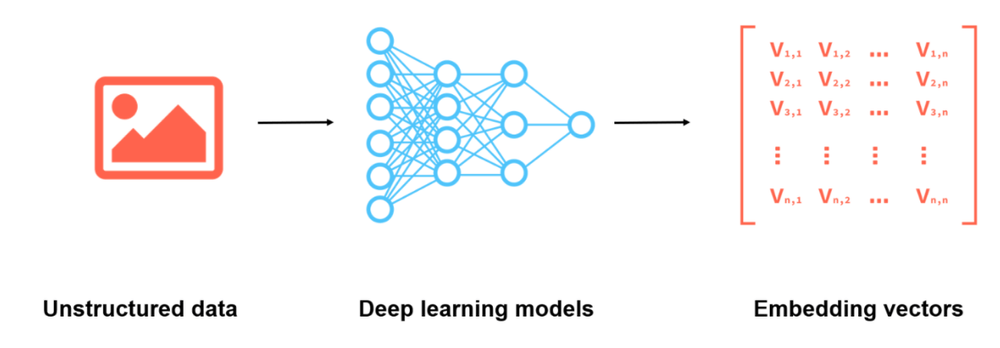
1. ERD



1. 고객 데이터
2. 회원가입이 이루어졌을 경우 auth\_user, users, UserProfile에 데이터저장
3. 유저 프로필 변경 시 UserProfile 데이터 수정
4. 상품 정보
   1. 상품 판매시 Product, posts, Category, ProductFile, FinalModel에 데이터저장
   2. FindImage, Category, FineModel 등을 조회

2. Chroma DB

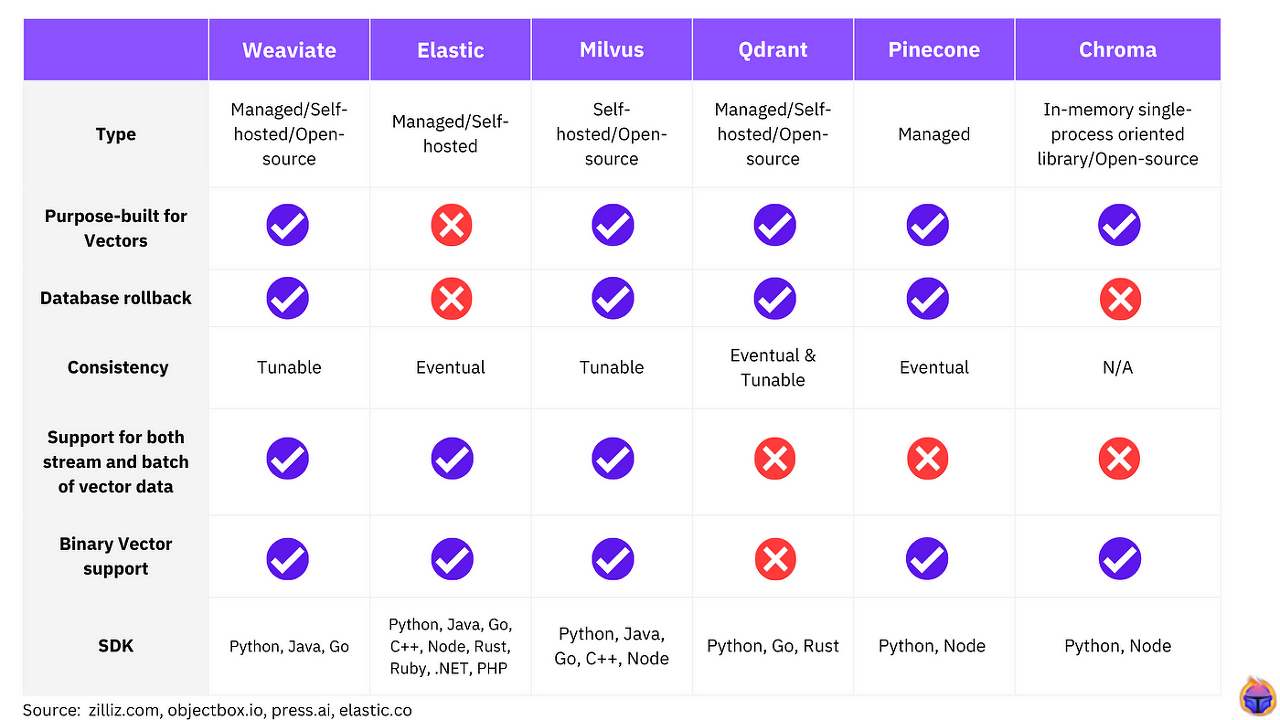
**벡터 임베딩**은 텍스트, 이미지, 영상, 오디오 등 다양한 유형의 데이터를 컴퓨터가 이해하고 처리할 수 있도록 수치 형태로 변환하는 방법입니다. 이 변환 과정은 사람이 직접 정의하는 것이 아니라, 데이터를 AI모델에 입력하여 모델 스스로가 해당 데이터를 의미 있는 벡터로 변환합니다.



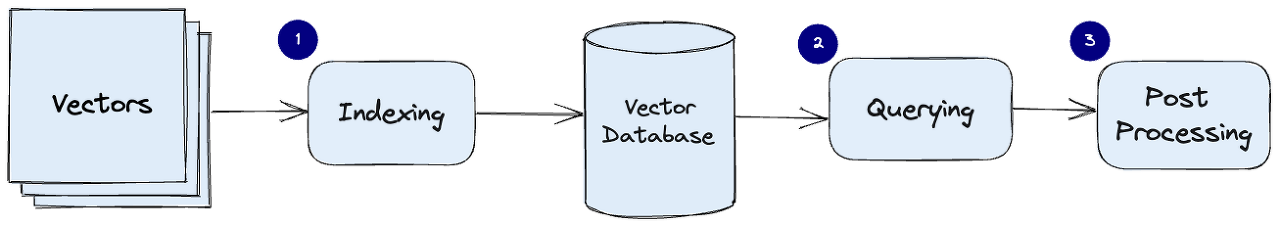
위 과정으로 임베딩된 데이터를 기존의 Oracle, MySQL과 같은 데이터베이스로 저장하기엔 다음과 같은 어려움이 있습니다.

1. 비정형 데이터 저장의 어려움: 관계형 데이터베이스는 주로 표 형태의 구조화된 데이터를 다루는데 적합
2. 유사성 검색의 한계: 관계형 데이터베이스는 쿼리와 완전히 일치하는 행을 찾아 반환하므로, 벡터 간의 유사성을 계산하고 검색하는 기능이 부족합니다.
3. 대용량 데이터 처리의 제약: 관계형 데이터베이스는 벡터 데이터의 대용량 처리에 최적화되지 않았기 때문에 성능 문제를 겪을 수 있습니다.

이러한 한계로부터 여러 **벡터 데이터베이스**가 등장했고, 대표적으로 아래의 표와 같습니다.   
저희는 이 중에서 간단하게 설치하여 사용할 수 있는 **Chroma**를 선택했습니다.



3. Chroma DB를 통한 유사 상품 비교



앞서 말씀드린 것처럼, 관계형 데이터베이스는 쿼리와 정확히 일치하는 행을 찾아 반환합니다. 반면, 벡터 데이터베이스는 유사도 메트릭을 적용하여 가장 유사한 벡터를 찾습니다. 벡터 데이터베이스의 일반적인 파이프라인은 위 사진과 같습니다.

마지막으로 어떤 과정을 거쳐 벡터 DB를 통해 유사 상품 비교를 했는지 설명드리겠습니다.

1. Indexing: 상품의 하위 카테고리를 예측하는 모델로 **Resnet**을 사용하는데 Image Detection 부분만 사용하여, 한 이미지를 [2024,] 사이즈 벡터로 임베딩 하였습니다.
2. Vector Database: 벡터 DB에는 **이미지 임베딩값**, **이미지 ID**를 보관하고, 그 외의 이미지 파일주소 등의 정보들은 관계형 DB에 보관하였습니다.
3. Querying: 벡터간의 유사도를 측정하는 방식으로는 **코사인 유사도**를 선택하였고, 이미지를 통해 상품 검색을 하였을시, Query문이 실행되어 유사한 이미지들을 가져오도록 설계하였습니다.

**< 테스트 결과 >**  