

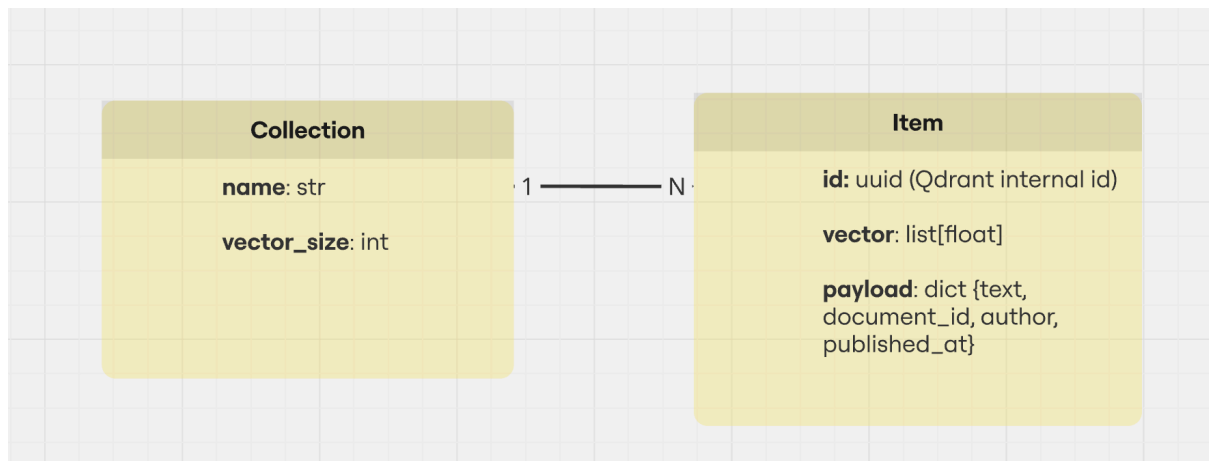
Sposób reprezentacji danych

System korzysta z **bazy wektorowej Qdrant**, która przechowuje:

- kolekcje wektorów (**Collection**) – odpowiednik tabeli w relacyjnej bazie,
- elementy wektorowe (**Item**) z metadanymi w formacie JSON,
- wektory do wyszukiwania (np. cosine similarity).

Reprezentacja trwała jest **całkowicie nierelacyjna**, zoptymalizowana pod wyszukiwanie wektorowe.

Pojęciowy model danych



Sposób reprezentacji danych trwałych

- **Qdrant** jako baza nierelacyjna (wektory + JSON metadata)
- Wektory: **list[float]**
- Metadane: **dict (JSON)**
- Daty publikacji konwertowane do formatu **YYYYMMDD** (int)

Struktury danych

Kolekcja – odpowiada obiektowi w Qdrant:

```
client.create_collection(  
    collection_name=name,  
    vectors_config=VectorParams(size=vector_size,  
    distance=Distance.COSINE)  
)
```

Element wektorowy – punkt w kolekcji:

```
client.upsert(  
    collection_name=name,  
    points=[  
        {  
            "id": str(uuid.uuid4()),  
            "vector": request.vector,  
            "payload": {  
                "text": "...",  
                "document_id": str(uuid.uuid4()),  
                "author": "...",  
                "published_at": 20251126  
            }  
        }  
    ]  
)
```

Sposób komunikacji aplikacji z bazą

- **Biblioteka:** `qdrant-client` (Python)
- **Endpointy:** serwisowa warstwa abstrakcji (`CollectionService`, `ItemService`)
- **Operacje:**
 - `create_collection` → `client.create_collection`
 - `delete_collection` → `client.delete_collection`
 - `add_item` → `client.upsert`
 - `delete_item` → `client.delete` + filtr po `document_id`
- Wyszukiwanie wektorowe z filtrami po metadanych