Neo4j와 LangChain을 활용한 지식그래프 입문

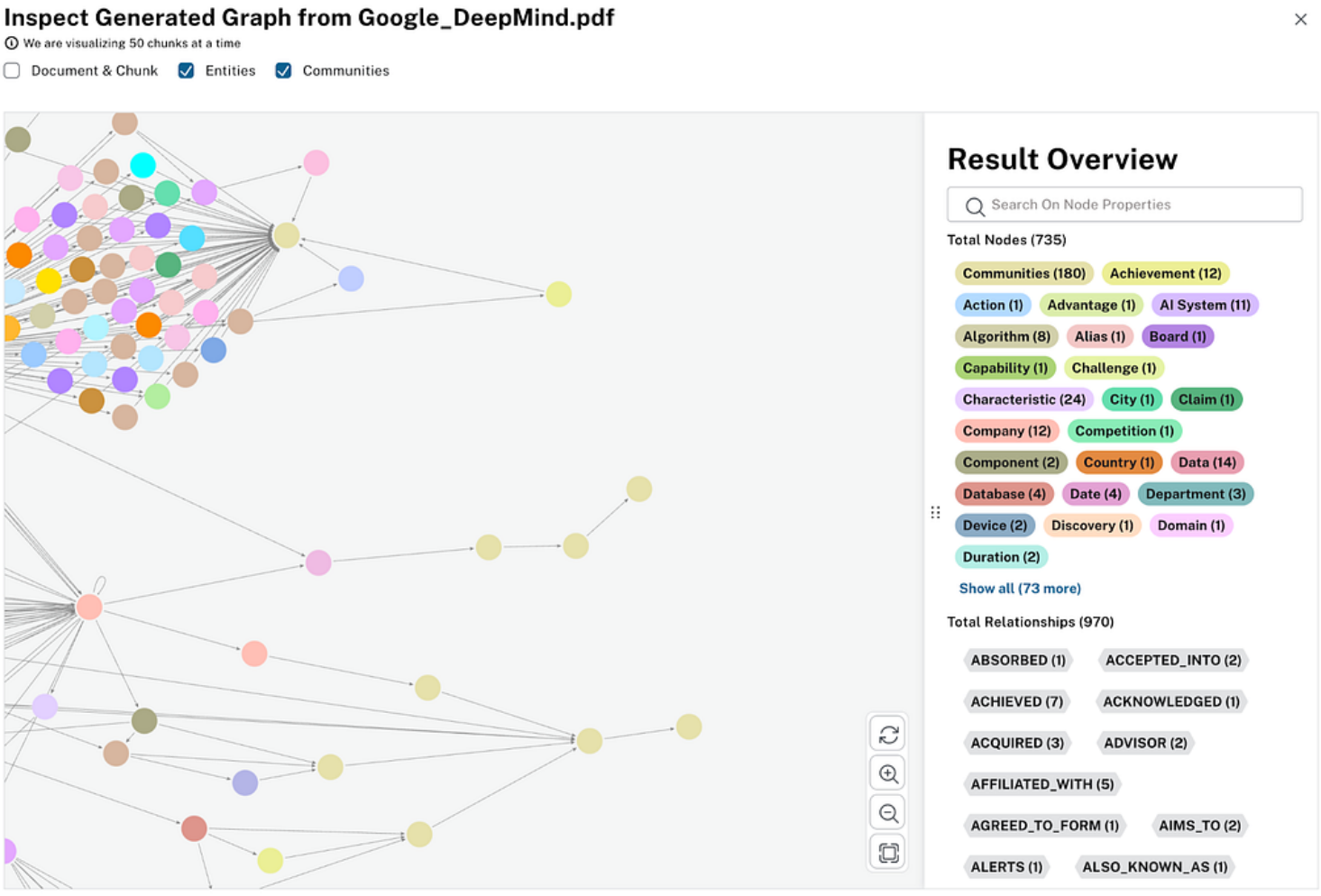
1. 지식그래프 소개

지식그래프란 무엇인가?

지식그래프(Knowledge Graph)는 현실 세계의 개체(entity)와 그들 사이의 관계(relationship)를 그래프 형태로 구조화한 데이터베이스입니다. 지식그래프는 단순한 사실(fact)들의 모음이 아니라, 정보 간의 의미론적 연결성을 통해 풍부한 지식 표현과 추론이 가능한 형태로 데이터를 구조화합니다.

지식그래프의 핵심적인 특징:

1. **의미론적 구조**: 데이터 간의 의미적인 관계를 명시적으로 표현
2. **유연한 스키마**: 필요에 따라 쉽게 확장 가능한 데이터 구조
3. **연결 중심 접근**: 데이터 포인트들의 연결성에 초점을 맞춤
4. **그래프 기반 탐색**: 복잡한 관계를 통한 효율적인 데이터 탐색 가능

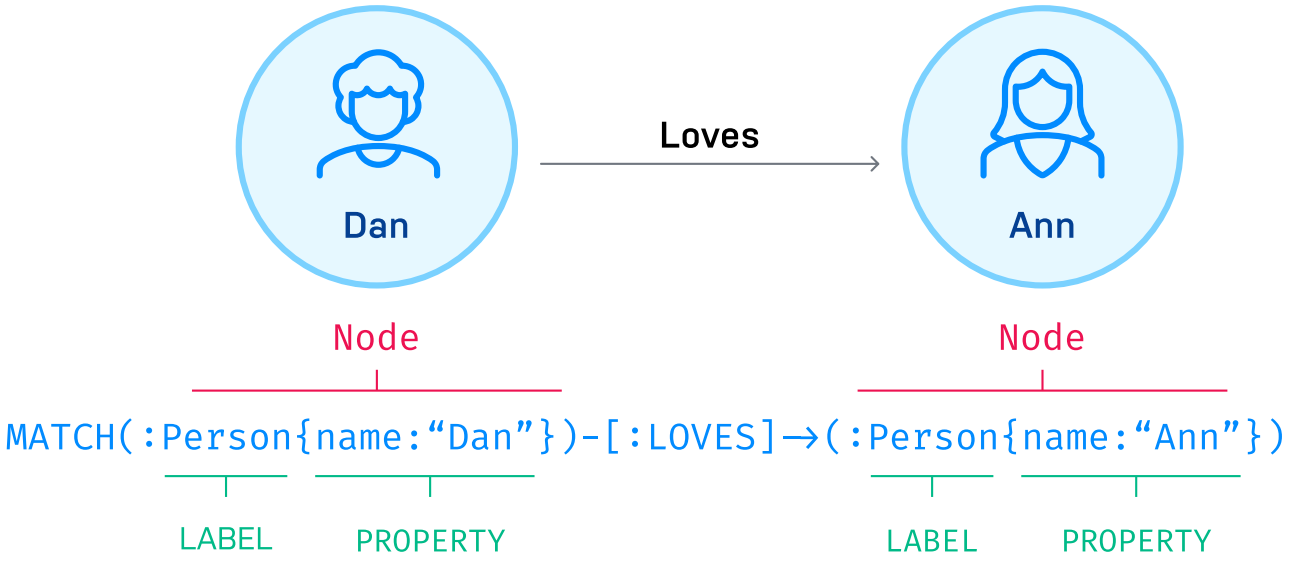


[이미지 출처] https://neo4j.com/blog/developer/knowledge-graph-builder-first/

그래프 데이터베이스의 주요 개념

그래프 데이터베이스는 다음과 같은 주요 요소들로 구성됩니다:

1. **노드(Node)**: 개체를 나타내는 기본 요소로, 레이블(label)과 속성(property)을 가짐
   ```
   (person:Person {name: "홍길동", age: 30})
   ```
2. **관계(Relationship)**: 노드 간의 연결을 나타내며, 타입과 방향성을 가짐
   ```
   (person1)-[:KNOWS {since: 2020}]->(person2)
   ```
3. **레이블(Label)**: 노드의 유형이나 역할을 분류하는 태그
   ```
   :Person, :Movie, :Product 등
   ```
4. **속성(Property)**: 노드나 관계가 가지는 값들
   ```
   {name: "홍길동", age: 30, email: "hong@example.com"}
   ```
5. **패턴(Pattern)**: 그래프 내에서 찾고자 하는 노드와 관계의 구조
   ```
   (person:Person)-[:LIKES]->(movie:Movie)
   ```



[이미지 출처] https://neo4j.com/docs/getting-started/_images/sample-cypher.svg

지식그래프의 활용 분야

지식그래프는 다양한 분야에서 활용되고 있습니다:

1. **검색 엔진**: 구글의 Knowledge Graph, 네이버 지식백과 등
2. **추천 시스템**: 사용자의 선호도와 아이템 간의 관계를 모델링
3. **AI와 자연어 처리**: 챗봇, 질의응답 시스템 등의 배경 지식으로 활용
4. **생명과학 및 의료**: 약물, 질병, 단백질 등의 복잡한 관계 모델링
5. **금융 및 사기 탐지**: 거래 패턴 분석 및 이상 거래 탐지
6. **소셜 네트워크 분석**: 사용자 간 관계 및 영향력 분석
7. **지식 관리 시스템**: 기업 내 지식과 정보의 체계적인 관리

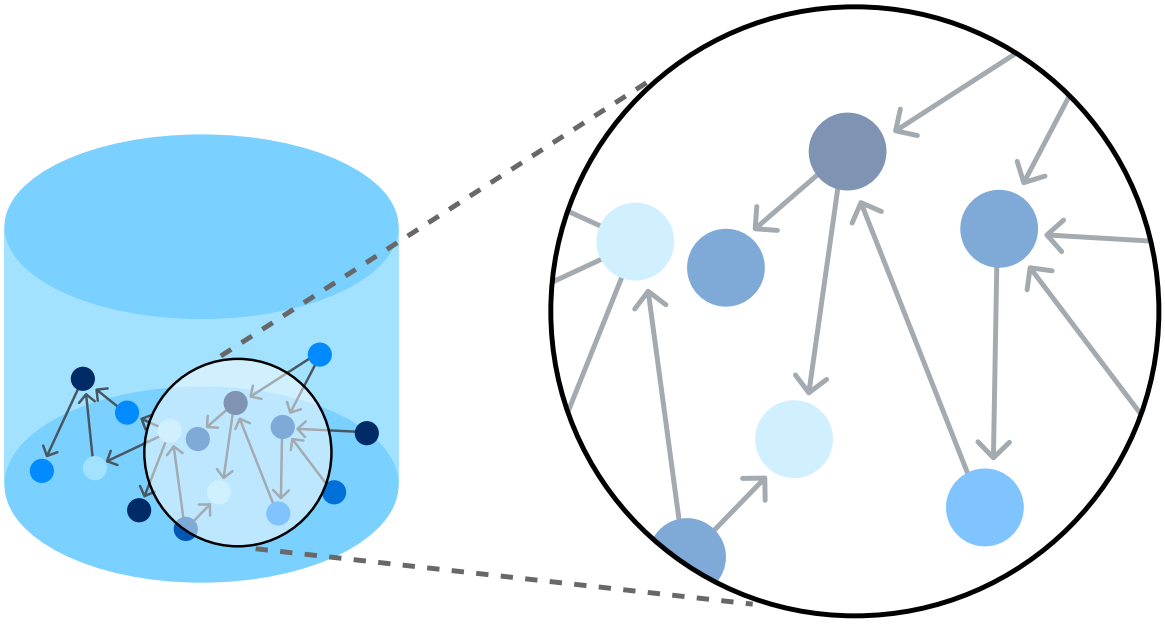
2. Neo4j 기초

Neo4j 소개

Neo4j는 가장 널리 사용되는 그래프 데이터베이스 관리 시스템 중 하나로, 노드와 관계를 효율적으로 저장하고 쿼리하기 위해 설계되었습니다.

Neo4j의 주요 특징:

1. **네이티브 그래프 저장소**: 그래프 데이터를 위해 처음부터 설계된 저장소
2. **선언적 쿼리 언어**: Cypher를 통한 직관적인 그래프 쿼리
3. **고성능 그래프 처리**: 인접 노드 조회 등의 그래프 연산에 최적화
4. **클러스터링 및 고가용성**: 엔터프라이즈 환경을 위한 확장성 제공
5. **시각화 도구**: 내장된 브라우저를 통한 그래프 시각화 기능



[이미지 출처] https://neo4j.com/product/#neo4j-desktop

Neo4J AuraDB 환경 설정

Neo4J AuraDB 소개

- 정의: Neo4j AuraDB는 완전 관리형 클라우드 서비스로 제공되는 Neo4j의 그래프 데이터베이스
- 특징:
  - 자동 확장 및 관리
  - 항상 최신 버전 유지
  - 자동 백업 및 장애 복구
  - 즉시 사용 가능한 환경 제공

AuraDB 인스턴스 생성 및 설정 방법

1. 회원가입
   - https://neo4j.com/product/auradb/ 방문
   - "Start Free" 버튼 클릭
   - 이메일로 가입 또는 Google 계정으로 가입 선택
2. AuraDB 인스턴스 생성
   - "New Instance" 선택
   - "Aura DB Free" 선택 (무료 티어)
3. 데이터베이스 자격 증명 저장
   - Username (기본값: neo4j)
   - Password (자동 생성됨)
4. 연결 정보 확인
   - 대시보드에서 인스턴스 선택
   - Connection URL 확인
5. 로컬 개발 환경 설정

```
NEO4J_URI=neo4j+s://xxxxxxxx.databases.neo4j.io
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=your_password
NEO4J_DATABASE=neo4j
```

6. Python 환경 설정

```
pip install -U langchain-neo4j  # 또는
poetry add langchain-neo4j
```