Cypher 쿼리

Cypher 쿼리

- 그래프 데이터베이스의 기본 구성 요소는 노드(Node)와 관계(Relationship)
- 노드는 그래프 데이터베이스의 데이터 레코드를 나타내며, 관계는 이러한 노드 사이의 연결을 정의
- 각 노드와 관계는 속성(Property)을 가질 수 있으며, 노드는 레이블(Label)을 통해 분류
- 복잡한 연결 관계를 가진 데이터를 효과적으로 표현하고 탐색

그래프 데이터베이스의 핵심 요소

노드(Node)

- 그래프 데이터베이스의 기본 데이터 레코드
- 관계형 데이터베이스의 행(row)과 유사한 개념이지만, 다른 노드와의 관계를 직접적으로 표현할 수 있다는 점이 큰 차이

레이블(Label)

- 노드의 집합을 정의
- 레이블을 통해 같은 특성을 가진 노드들을 그룹화하여 도메인을 형성할 수 있음

관계(Relationship)

- 노드 사이의 연결을 정의
- 항상 방향성을 가지며, 두 노드 간에 여러 관계가 존재할 수 있음

속성(Property)

- 노드와 관계에 부여되는 이름-값 쌍
- 노드와 관계에 추가적인 정보를 저장

Cypher 쿼리 기초 문법

- 여러 절(clause)을 연결하는 방식으로 쿼리를 구성

- 중간 결과를 서로 연결하며, 각 절의 매칭 변수는 다음 절이 작동하는 컨텍스트

주요 Cypher 키워드

MATCH

- 데이터베이스에서 노드, 관계, 레이블, 속성 또는 패턴을 검색하는 데 사용.
- SQL의 SELECT와 유사한 역할을 하지만, 그래프 패턴을 지정하여 검색할 수 있다는 점이 큰 차이
- 그래프에서 데이터를 가져오는 가장 일반적인 방법

RETURN

- 쿼리 결과로 반환할 값을 지정
- 노드, 관계, 속성 또는 패턴을 반환하도록 지시
- RETURN을 반드시 사용해야 함, 생략할 경우 구문 오류가 발생.

WHERE

- MATCH, OPTIONAL MATCH 등의 일부로 사용되며, 패턴에 제약 조건을 추가하거나 중간 결과를 필터링하는 역할

CREATE

- 노드와 관계를 생성하는 데 사용
- 속성과 레이블을 지정하여 새로운 데이터를 그래프에 추가

노드와 관계 조작하기

노드 생성

- 노드 생성 시 레이블과 속성을 함께 지정할 수 있음

```
CREATE (p:Person {name: '테스트'})
```

노드 조회

- 'Person' 레이블을 가진 노드 중에서 name 속성이 '테스트'인 노드를 찾아 반환

```
MATCH (p:Person{name:'테스트'})
RETURN p
-- 모든 Person 노드 조회회
MATCH (p:Person)
RETURN p
```

관계 생성

- 두 노드를 찾은 다음, CREATE 문을 사용하여 관계를 설정

```
MATCH (a:Person {name:'테스트'}), (b:Person{name: 'Dan'})
CREATE (a)-[r:LIKES]→(b)
RETURN r
```

노드와 관계 삭제

- 노드를 삭제하기 전에는 해당 노드와 연결된 모든 관계를 먼저 제거

```
-- 관계 삭제
MATCH (p:Person)-[r:LIKES]→(Person{name:'Dan'}) DELETE r
-- 노드 삭제
MATCH (p:Person {name: 'Dan'}) DELETE p
```

고급 쿼리 기법

패턴 매칭을 통한 복잡한 관계 탐색

```
MATCH (john {name: 'John'})-[:FRIEND]→()-[:FRIEND]→(fof)
RETURN john.name AS me, fof.name AS `two hops`
```

- 'John'이라는 이름을 가진 노드에서 시작하여 'FRIEND' 관계를 두 번 거친 노드를 찾는다.
- AS 키워드를 사용하여 결과 열의 이름을 변경할 수도 있음

조건부 패턴과 선택적 매칭

- OPTIONAL MATCH를 사용하면 패턴이 존재하지 않을 경우에도 NULL 값을 반환하여 쿼리 결과를 유지

```
MATCH (p:Person)
WHERE p.name STARTS WITH 'J'
OPTIONAL MATCH (p)-[:WORKS_FOR]-(other:Company)
RETURN p.name, other.name
```

- 이름이 'J'로 시작하는 모든 Person 노드를 찾고, 해당 사람이 일하는 회사 정보가 있다면 함께 반환

집계 함수 사용

- SQL과 달리 그룹화 키를 명시적으로 지정할 필요가 없다

```
MATCH (p:Person)-[:IS_FRIENDS_WITH]→(friend:Person)
RETURN p.name, collect(friend.name) AS friend
```

- 각 사람과 그 사람의 친구 목록을 반환
- collect() 함수는 여러 값을 배열로 모아주는 역할

제약조건 최적화

- 유니크 제약 추가

```
CREATE CONSTRAINT FOR (n:Person) REQUIRE n.name IS UNIQUE
```

결과 제한과 정렬

- 결과의 수를 제한하거나 특정 기준으로 정렬하여 쿼리 결과를 관리

```
MATCH (m:Movie)
WHERE m.released > 2000
RETURN m
LIMIT 5
```

- 2000년 이후에 출시된 영화 중 5개만 반환