JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 4, pp. 123-131, April 2020
https://doi.org/10.9708/jksci.2020.25.04.123

Is-A Node Type Modeling Methodology to Improve Pattern Query Performance in Graph Database

Uchang Park*
*Professor, Dept. of Computer Engineering, Duksung Women's University, Seoul, Korea

[Abstract]

The pattern query in graph database has advantages of easy query expression and high query processing performance compared to relational database SQL. However, unlike the relational database, the graph database may not utilize the advantages of pattern query depending on modeling because the methodology for building the logical data model is not defined. In this study, in the is-a node modeling method that appears during the graph modeling process, we experiment that there is a difference in performance between graph pattern query when designing with a generalization model and designing with a specialization model. As a result of the experiment, it was shown that better performance can be obtained when the is-a node is designed as a specialization model. In addition, when writing a pattern query, we show that if a variable is bound to a node or edge, performance may be better than that of the variable of not bounded. The experimental results can be presented as an is-a node modeling method for pattern query and a graph query writing method in the graph database.

▸ **Key words**: Graph Pattern Query, Is-a modeling, Cypher, Graph Database, SQL

[요 약]

그래프 데이터베이스에서 패턴질의는 관계 데이터베이스 SQL과 비교하여 질의의 쉬운 표현, 높은 질의 처리 성능을 기대할 수 있는 장점이 있다. 그러나 그래프 데이터베이스는 관계 데이터베이스와 달리 논리적 데이터 모델을 구축하는 방법론이 정의되어 있지 않아 모델링에 따라 패턴질의의 장점을 활용하지 못할 수 있다. 본 연구는 그래프 모델링 과정 중 나타나는 is-a 노드 모델링 방법에서 일반화 모델로 설계할 경우와 특수화 모델로 설계할 경우 그래프 패턴질의의 성능 차이가 있음을 실험하였다. 실험 결과 is-a 노드 설계를 특수화 모델로 설계할 경우 더 우수한 성능을 얻을 수 있음을 보였다. 또 추가로 패턴질의를 작성할 때 변수를 노드나 간선에 바인딩시키는 경우 그렇지 않는 경우보다 성능이 우수할 수 있음을 보였다. 실험 결과들은 그래프 데이터베이스에서 패턴질의에 대한 is-a 노드 모델링 방법 및 그래프 질의 작성 방법으로 제시될 수 있다.

▸ **주제어**: 그래프 패턴질의, Is-a 모델링, Cypher, 그래프 데이터베이스, SQL

• First Author: Uchang Park, Corresponding Author: Uchang Park
*Uchang Park (ucpark@duksung.ac.kr), Dept. of Computer Engineering, Duksung Women's University
• Received: 2020. 02. 06, Revised: 2020. 04. 08, Accepted: 2020. 04. 17.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

그래프 데이터베이스는 NoSQL 데이터베이스의 일종으로 패턴질의를 사용하여 패턴의 탐색, 추천 시스템 등 응용에 좋은 질의 성능을 보여주고 있다[1][2]. 그래프 데이터베이스는 그래프 정의에 따라 설계되지만, 관계 데이터베이스와 달리 논리적 데이터 모델을 구축하는 표준화된 방법론이 별도로 정의되어 있지 않다.

현재까지 제시된 방법들로는 예제 데이터를 화이트보드에 그려보는 방식을 통한 노드 및 관계 추출, 관계 데이터베이스 관점에서 모델링 한 후 그래프 모델로 변환하는 방법이 있다. 전자의 방법은 화이트보드에 예제 데이터 노드와 노드 간의 관계를 그려보고 노드 타입과 관계를 특성그래프로 나타내는 방법이다. 이 방법은 경험이 필요하며 정확성을 검증하기 어려운 점이 있다. 후자의 방법은 관계 데이터 모델에 익숙한 경우 사용할 수 있는 방법으로 관계 데이터베이스 모델을 그래프 모델로 변환하는 방법이다. 테이블은 노드 레이블로, 행은 노드로, 외래키는 관계로, 교차 테이블은 관계로 변환하는 등의 몇 가지 규칙을 사용하여 특성 그래프로 모델링한다. 그래프 데이터 모델링의 결과의 적정성을 판단하는 기준은 따로 없지만, 모델에 따라 질의 패턴에 따라 성능이 달라질 수는 있다. 예를 들면 노드 특성값의 카테고리를 찾는 질의이면 효율성을 위하여 같은 특성을 갖는 노드들을 높은 계층의 노드 범주로 모델링하는 것을 검토할 수 있다. 어떤 모델링이 더 적절한가는 질의 패턴의 성능을 측정하는 것이 좋지만 현실적으로는 어려운 문제이다[3][4].

모델링 이후에는 그래프 질의어를 잘 작성하여 성능을 높이도록 하여야 한다. 대표적인 그래프 데이터베이스 Neo4j의 Cypher 질의어의 경우 SQL과 같은 선언적인 질의어이며 SQL과 다른 장점은 노드나 에지의 레이블을 바운드하지 않더라도(명시하지 않더라도) 검색 경로상의 노드나 에지를 찾아서 바운드시키는 장점이 있다. 큰 그래프 데이터의 경우 유용한 장점이다[5].

그래프 질의는 패턴질의(pattern matching query)와 분석 질의(analytical query)로 나눈다[6]. 패턴질의는 그래프의 부속 그래프(subgraph)를 찾는 질의로, 노드 레이블로 경로 필터링, 간선 레이블로 경로 필터링, 사이클 패턴 등 그래프의 일부분에 접근하는 질의이고, 분석질의는 노드 차수, 차수의 집중도, 연결성, 최단 거리 계산 등 그래프의 많은 부분에 접근하는 질의이다. 최단거리 문제 같은 분석질의는 그래프 데이터베이스가 관계 데이터베이스보다 성능이 더 뛰어나지만, 패턴질의의 경우는 데이터베이스 시스템에 따라 다른 성능을 보이는 것으로 알려져 있다[7]. 그래프 패턴질의는 그래프 데이터베이스에서 중요한 질의이며 관계 데이터베이스 질의로 변환할 경우 질의 형태가 조인문으로 수행되기 때문에 문장이 복잡하게 나타난다[8].

본 연구는 그래프 데이터베이스를 논리적으로 모델링할 때 부모노드 타입 A가 서로소(disjoint)인 자식노드 타입 (B1, B2, ..., Bn)으로 구성된 is-a 관계 사례에서 모델링 방법 선택에 따른 패턴질의 성능을 검토한다. 관계 데이터베이스의 is-a 상속관계를 나타내는 여러 모델링 방법이 있는 것과 같이 그래프 데이터베이스에서도 여러 방법이 있을 수 있다[9][10].

Is-a 관계 모델링의 선택 방법의 예로 부모노드가 'Person'이고 자식노드가 'Employee'와 'Student'인 경우, 자식노드를 Person 노드로 통합하여 표현하고 Person 노드의 속성으로 Employee 혹은 Student 값을 저장하는 일반화(generalization) 방법, 부모노드 Person을 없애고 Employee와 Student를 독립된 노드 타입으로 설계하는 특수화(specialization) 방법이 있다. 각각의 방법을 모델1(일반화 모델)과 모델2(특수화 모델)로 부르기로 하자.

그래프 노드의 is-a 타입 모델링 설계 시 일반화/특수화 방법에 따른 그래프 패턴질의 성능에 관하여 기존 연구가 진행된 바 없다[11]. 본 연구에서는 그래프 데이터 모델링의 적절성을 데이터베이스 구축 후 패턴질의를 수행하여 판단하는 것보다 패턴질의의 여러 유형에 대하여 성능 평가를 통하여 일반적인 가이드라인을 제시하고자 한다. 모델에 따른 성능 평가 비교는 그래프 패턴질의를 사용하며, 그래프 데이터베이스의 대표적인 시스템인 Neo4j[12]를 사용한다. Neo4j는 질의어로 Cypher를 사용하며 특성 그래프에 대한 강력한 질의 기능이 있다.

또 추가로 일반화 모델의 경우 Cypher 질의와 관계 데이터베이스의 SQL 패턴질의에 대하여 성능을 비교하고자 한다.

2장에서는 본 연구의 is-a 모델링 방법에 관해 설명하고, 3장에서는 그래프에 대한 패턴질의, 4장에서는 is-a 모델에 따른 패턴질의 성능을 평가하고, 5장에서는 결론 및 향후 연구에 대하여 논의한다.

II. Is-a modeling

본 연구에서는 자식노드 타입이 서로소(disjoint) 관계가 있는 is-a 모델에 대한 모델링 방법 모델1(일반화 모델), 모델2(특수화 모델)에 대한 실험을 통하여 패턴질의

의 성능을 테스트한다.

본 연구의 테스트 데이터는 LUBM(Lehigh University BenchMark) 벤치마크를 이용하였다[13]. LUBM 벤치마크는 대학의 학사 시스템을 모델링하며 is-a 관계가 많이 포함되어 있다. 관계 데이터베이스로 나타낸 학사 데이터베이스 스키마는 그림 1의 ER(Entity Relationship) 다이어그램과 같이 Departments, Professor, Students, Courses, TakesCourse, Publications, PublicationAuthor, Coauthors 등으로 나타난다.

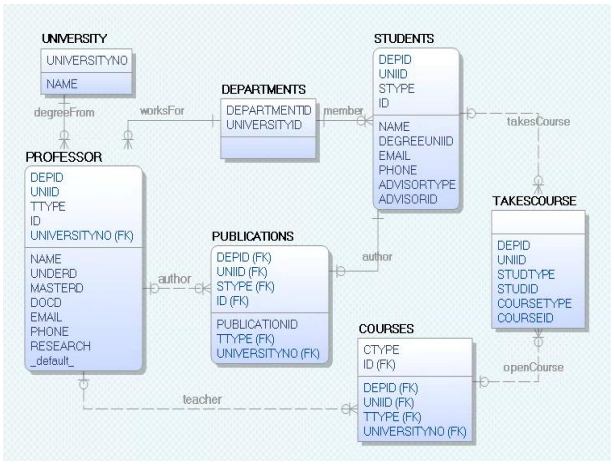

Fig. 1. ER diagram for LUBM schema

관계 데이터베이스를 그래프 데이터베이스로 변환하면 릴레이션은 노드로, 1:1, 1:N 외래키 관계는 간선으로, M:N 교차테이블은 그래프 간선으로 표현된다.

실험 데이터를 Renzo[14]가 제안한 특성 그래프 스키마(Property Graph Schema)로 표현하면 그림 2와 같다. 그림 2는 6개의 노드로 구성되어있으며 is-a 관계의 자식노드가 부모노드 1개로 통합된 형태이다. 그림 2를 모델1 스키마라고 부르기로 한다.

모델1 스키마의 Professor, Students, Courses 노드의 경우 is-a 관계를 갖는 부모 개체이다. Professor, Students, Courses 노드는 다음과 같은 서로소인 자식 개체로 구성된다.

Professor = {FullProf, AssociateProf, AssistantProf, Lecturer}
Students={UnderStudent, GradStudent }
Courses={Course, GradCourse}

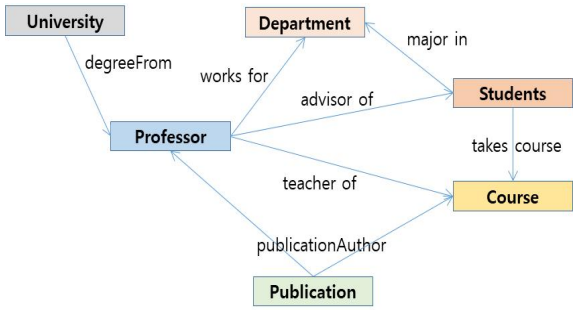

Fig. 2. Property graph schema(model 1 schema)

그래프 모델에서 노드 타입이 부모노드 타입/자식노드 타입의 is-a 계층 관계를 갖고 자식노드의 원소가 한 개의 자식노드 타입에만 속하는 경우 모델링을 자식노드 타입 각각을 독립된 노드 타입으로 구성할 수 있다. 이 방법을 이용하여 모델링된 특성 그래프 스키마를 표현하면 그림 3과 같다. 그림 3을 모델2 스키마라고 부르기로 한다.

그림 3은 is-a 관계의 부모노드 타입을 없애고 자식노드 타입만을 표현한 방법으로 Professor는 FullProfessor, AssociateProf, AssistantProf, Lecturer의 4개의 하위노드 타입으로, Students는 GraduateStududent, UnderStududent의 2개의 하위노드 타입으로, Courses는 GradCourse, Course의 2개의 하위노드 타입으로 분리되어 표현한다.

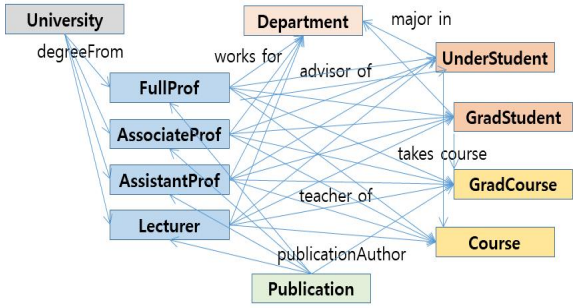

Fig. 3. Property graph schema(model 2 schema)

III. Graph Pattern Query

먼저 패턴질의 성능을 검토하기 위하여 패턴을 정의하고 패턴질의의 유형을 소개한다.

(정의) 그래프 패턴 P

그래프 G=(V, E)가 주어졌을 때, 그래프 패턴 P는 P=(Vp, Ep)로 나타내며 부분 그래프가 노드와 간선에 대한 조건 p를 각각 만족한다.

본 연구에서는 모델1 스키마와 모델2 스키마에 대하여 그래프 패턴질의의 성능을 비교한다. 패턴질의 타입은 Jürgen Hölsch[6] 연구에서 구분한 4가지 타입으로, 노드 레이블로 경로 필터링, 간선 레이블로 경로 필터링, 노드/간선 레이블로 경로 필터링, 사이클 경로 타입의 질의이다. 패턴질의의 4가지 타입에 대한 예시는 표 1과 같다. 패턴 타입1은 노드를 바인딩한 질의, 패턴 타입2는 간선을 바인딩한 질의, 패턴 타입3은 노드/간선 모두를 바인딩한 질의, 패턴 타입4는 패턴 타입3에 사이클이 포함된 경우이다.

본 연구에서는 성능 평가를 좀 더 정밀하게 하도록 4가지 질의 패턴 타입 각각에 패턴 경로 길이가 1~2인 단순질의, 패턴 경로 질의가 2 이상인 복잡질의의 2가지 형으로 나누어 Neo4j Cypher 언어로 실험을 한다. Cypher 언어는 노드나 간선을 변수로 작성할 수 있으므로 특정 노드나 간선에 바인딩시키지 않고 질의를 작성할 수 있다. 질의를 바인딩을 없이 작성할 수 있으면 질의가 간단해진다.

본 연구에서는 표 1의 4가지 패턴질의에 대하여 패턴 경로를 단순형, 복잡형으로 나누어 모델1 스키마, 모델2 스키마의 총 16개의 질의(패턴질의 타입 4가지, 단순/복잡형, 모델 2가지)에 대하여 예제 질의를 제시하고 성능을 비교하였다. 또 질의 타입3, 질의 타입4의 경우는 오라클 관계 데이터베이스의 4가지 SQL 질의(질의 타입 2가지, 단순/복잡형)에 대하여 추가 비교하였다. Cypher 질의는 노드와 간선이 바인딩하지 않고도 노드와 간선을 찾을 수 있지만, SQL 질의에서는 테이블 이름이나 외래키 관계를 명시하지 않고는 질의 작성이 어려우므로 타입1, 타입2는 비교할 수 없다. 각 질의 내용은 부록에 수록하였다.

Table 1. 4 pattern query types and examples

Type	Explanation Cypher Query Example
1	Path filtering on node labels MATCH (a:FullProfessor)--(b:Course) RETURN count(*)
2	Path filtering on edge types MATCH (a)-[:teacherOf]->(b) RETURN count(*)
3	Path filtering on node labels and edge types MATCH (a:FullProfessor)-[:teacherOf]->(b:Course) RETURN count(*)
4	Cycles with path filtering on node labels and edge types MATCH (a:FullProfessor)-[:teacherOf]->(b:Course)<-[:takesCourse]-(c:UndergraduateStudent)-[:advisor]->(d:FullProfessor) WHERE a=d RETURN count(*)

IV. Pattern Query Performance

1. Performance Experiments

LUBM 벤치마크는 RDF(Resource Description Framework) 벤치마크로 사용되며 그래프 데이터베이스 모델로 변환할 수 있다. 테스트 데이터는 LUBM 벤치마크 2개 대학용 학사 데이터베이스를 생성하였으며 데이터는 교수(1,242명), 학생(17,878명), 강좌(3,728개), 수강(49,306건)이다.

질의는 부록에 수록된 Q1~Q8에 대하여 2가지 그래프 모델링 방법 총 16가지 패턴질의를 테스트하였다. 추가로 Q5~Q8에 대하여서는 SQL 질의 성능을 모델1 방법으로 비교하였다. 모델2로 SQL 성능을 측정하지 않는 이유는 질의가 복잡해지며 성능도 좋지 않아 비교 대상에서 제외하였다. 질의는 결과를 출력하는 시간에 영향을 받지 않기 위하여 SELECT count(*) 문을 이용하였고, 성능 측정 시간은 5번 이상 측정하여 가능한 warm cache[15] 상태를 측정한 시간이 나오도록 하였다. 실험 결과는 표 2로 그림 4와 같다.

Table 2. Pattern query performance for each model(ms)

Type	Query style	No	Graph Model1	Graph Model2	SQL Query
1	simple	Q1	3	2	NA
	complex	Q2	4	3	NA
2	simple	Q3	3	2	NA
	complex	Q4	5	4	NA
3	simple	Q5	3	2	10
	complex	Q6	5	3	10
4	simple	Q7	20	12	8
	complex	Q8	33	15	12

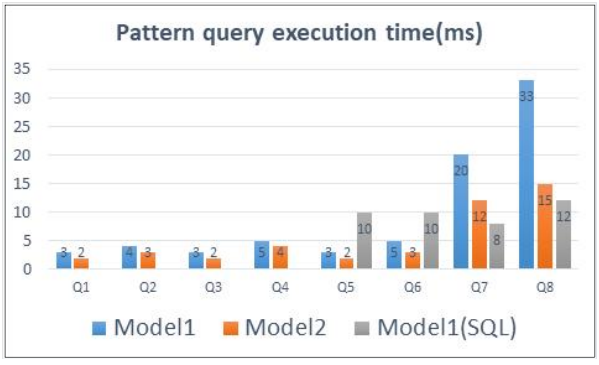

Fig. 4. Pattern query execution performance graph

표 2를 분석해보면 다음과 같은 내용을 관찰할 수 있다.

(1) 그래프 모델1과 모델2의 Q1~Q6 질의 비교

그래프 모델1은 슈퍼클래스 노드를 하나의 노드 타입으로 구성한 방식이며, 그래프 모델2는 서브 클래스를 각각의 노드 타입으로 구성한 방식이다. 그래프 모델2는 Q1~Q6에서 그래프 모델1보다 약간 우수한 성능을 나타내었다. 성능 차이는 모델1에서 슈퍼클래스 노드에서 값을 바인딩하는 것보다 모델2에서 서브클래스 노드 전체를 바인딩하는 방식이 성능에 차이를 나타낸 것으로 관찰되었다.

예를 들어 Q2 질의를 살펴보면 모델1과 모델2의 그래프 질의 실행계획은 동일하지만, 노드 필터링 연산에서 모델1은 Student 노드 내에서 “UnderGraduate” 속성값을 검색하는 반면 모델2는 UnderGraduate 타입 노드를 검색하는 차이가 있다. Cypher의 PROFILE 명령어를 이용하여 DbHits를 측정한 결과에 따르면 모델1 질의가 DbHits가 더 높게 나온 결과이다. DbHits는 Neo4j 저장 엔진(storage engine)이 데이터 검색시 작업량을 계산하는 단위이다. Cypher 질의최적기에서 제공하는 DbHits 단위는 그래프의 노드와 간선에 대한 추상적인 작업의 단위이다.

(2) 그래프 모델1과 그래프 모델2의 Q7~Q8 사이클 질의 비교

그래프 모델1과 모델2 질의의 Q7, Q8 사이클 질의의 경우 사이클이 아닌 경우와 비교하면 모델1의 성능 저하가 더 많이 나타났다. 질의어 실행 후 Q7의 PROFILE을 비교해 보면 그림 5와 같다. Cypher의 EXPLAIN 명령어를 이용하여 살펴보면 두 질의의 옵티마이저 실행계획이 차이가 있음을 알 수 있다. 그림 5의 Cypher의 실행계획은 NodeByLabelScan(특정 레이블의 모든 노드를 찾음), EXPAND(노드의 연결된 에지를 모두 찾음), FILTER(술어에 해당되는 노드를 추출), NodeHashJoin(shem 번호로 해시 조인), EagerAggregation(그룹 수식을 계산) 연산 등이 사용되었다.

(3) 그래프 모델과 관계 모델의 Q5~Q8 질의 비교

그래프 모델은 관계 모델 질의와 비교하면 일반적인 패턴질의 Q5-Q6 질의에서는 우수한 성능을 보였다. 그래프 모델의 경우 노드/간선을 바인딩시키는 경우 그래프 경로를 바인딩된 노드/간선을 직접 찾아가기 때문에 관계 모델보다는 성능이 우수한 것으로 판단된다. 단, 그래프 모델의 경우 사이클이 있는 Q7-Q8 질의에 대해서는 관계 모델과 비교하면 속도 저하를 나타내었다. 관계 모델의 경우 많은 양의 데이터에 대한 조인을 HASH 조인과 기본키 인덱스를 활용하여 최적화된 알고리즘으로 사이클도 비교적 빠른 속도로 처리를 하는 것으로 판단된다.

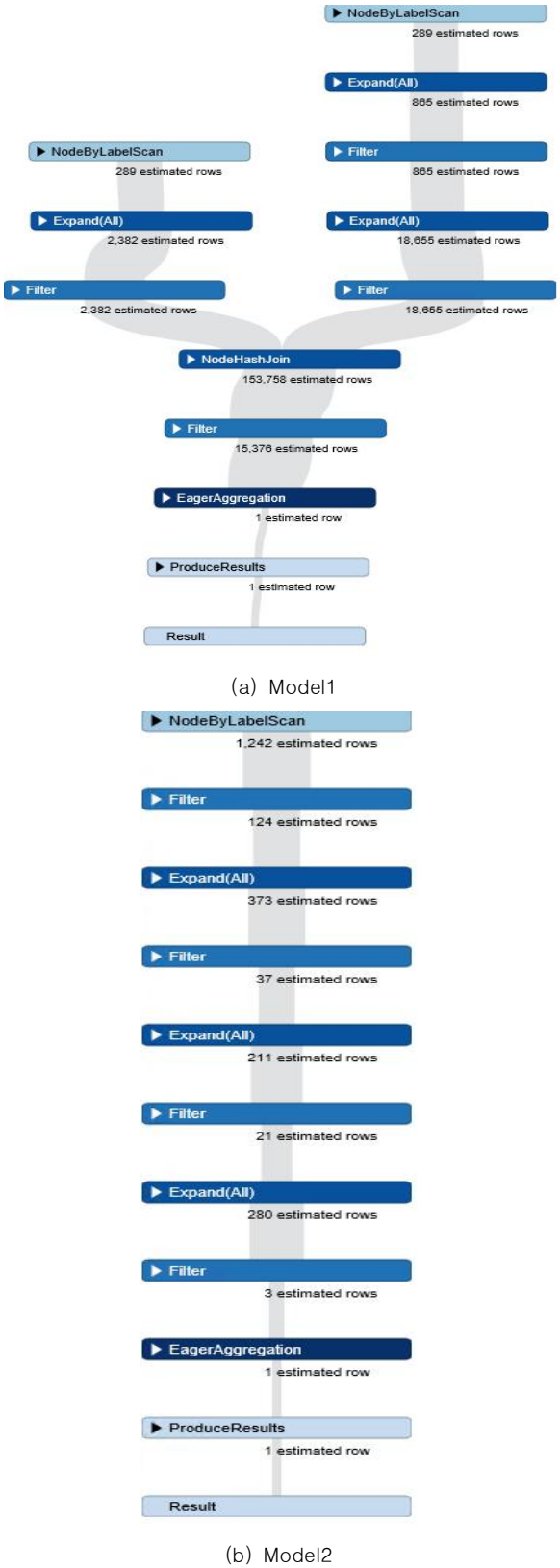

Fig. 5. Q7 query execution plan comparison

(4) 추가적인 실험 결과로 모델1, 모델2에 상관없이 그래프 질의에 노드 타입과 간선 타입을 바인딩시키는 경우 그렇지

않은 경우보다 표 3의 결과와 같이 수행 속도가 빠르게 나타난다. Q6 질의에 대하여 노드 혹은 간선에 바인딩 여부에 따라 실행시간을 보인 것이다. Cypher 질의의 특징은 노드 혹은 간선에 노드 타입 혹은 간선 타입을 바인딩시키지 않아도 되는 장점이 있지만 가능한 노드 타입이나 간선 타입을 바인딩시키는 것이 더 유리한 것으로 판단된다. 노드는 바인딩시키지 않아도 유일하게 찾아가지만, 에지를 바인딩시키지 않을 경우 노드에서 나가는 에지 타입이 2개 이상일 경우 수행 속도는 현저하게 차이가 남을 알 수 있었다.

Table 3. Q6 query execution performance for node/edge bound(ms)

binding / Model	Node /Edge binding	without Node binding	without Edge binding
Model 1	5	5	10
Model 2	4	4	8

2. Is-a Modeling Comparison

관계 데이터베이스에서 is-a 관계의 경우 응용에서 나타나는 질의 유형 빈도에 따라 모델1(일반화 모델), 모델2(특수화 모델) 혹은 혼합된 모델이 선택되지만, 그래프 데이터베이스에서는 그래프 데이터베이스 질의 언어에 노드 타입이나 에지 타입에 변수를 사용할 수 있어서(바인딩하지 않아도 되는), 모델2와 같이 설계를 하는 것이 질의가 간편하고 성능 면에서 바람직함을 알 수 있다.

그래프 질의의 간편성과 성능을 비교하면 표 4와 같다.

Table 4. is-a modeling comparison between model1 and model2

질의 / is-a 모델	Query Performance, Parent/child node selection
Model 1 (Generalization)	Low performance compared to Model 2 Querying to child node by property selection
Model 2 (Specialization)	Good Performance compared to Model 1 Querying to parent node by unbinding to edge selection

V. Conclusions

그래프 데이터베이스에서 패턴질의는 중요한 질의로 관계 데이터베이스와 비교하여 질의의 쉬운 표현, 우수한 질의 처리 성능을 기대할 수 있는 장점이 있다. 본 연구에서는 이러한 장점을 활용할 수 있는 여건 중 그래프 데이터베이스 is-a 모델링 방법의 선택과 Cypher 질의 바인딩 여부에 관한 사례에 대하여 실험을 진행하였다. 실험에 사용된 데이터베이스는 LUBM 벤치마크에서 사용하는 대학 학사 데이터베이스를 사용하였으며, Neo4j 그래프 데이터베이스로 실험하였다.

실험 결과 첫째, 그래프 데이터 모델에서 is-a 클래스의 경우 부모노드로 통합하여 일반화 모델링하는 것보다 자식 노드들로 특수화 모델링하는 것이 패턴질의 성능이 더 우수함을 보였다. 둘째, 그래프 패턴질의의 경우 가능한 간선을 바인딩시키는 경우가 그렇지 않은 경우보다 질의 성능이 우수함을 보여 패턴질의 작성에 지침을 제시하였다. 셋째 추가적인 실험을 통하여 Cypher 질의에서 노드나 간선을 바인딩시키는 경우 관계 데이터베이스의 SQL 질의보다 패턴질의에서 성능이 우수함을 보였다. 다만, 패턴질의에 사이클이 있는 경우 Cypher 질의가 성능이 저하됨을 보였다.

그래프 데이터베이스는 관계 데이터베이스에서처럼 정규화 등 모델의 적절성을 평가할 방법은 없는 것으로 알려져있다. 따라서 모델의 적절성은 응용에 따라 그래프 패턴질의의 성능을 높이는 방향으로 설계되어야 하고 그 중 하나의 방법이 is-a 모델링의 선택에 관한 것이다. 본 연구에서 패턴질의의 성능 실험결과를 통하여 is-a 모델링의 경우 특수화 모델링이 성능이 우수함을 알 수 있었다. 이 결과는 그래프 데이터베이스 모델링에서 is-a 노드 모델링 방법의 중요한 지침이 될 수 있다.

향후 연구계획은 대용량 그래프 데이터베이스의 경우 본 연구에서 제시된 실험 결과를 더 확인하고 그래프 모델링의 유용한 지침을 더 찾아내는 것이다.

ACKNOWLEDGEMENT

This Research was supported by Duksung Women's University Research Grants 2019.

REFERENCES

[1] Renzo Angles and Claudio Gutierrez, et al., "An Introduction to Graph Data Management," Graph Data Management, Springer, pp. 1-43, 2018.

[2] Chad Vicknair, et al., "A Comparison of a Graph Database and a Relational Database," ACMSE '10, Oxford, MS, USA, Apr. 15-17, 2010.

[3] https://neo4j.com/developer/data-modeling/, [accessed: Aug. 1,

2019]

[4] Subhrajyoti Bordoloi and Bichitra Kalita, "Designing Graph Database Models from Existing Relational Databases," International Journal of Computer Applications, Vol. 74, No. 1, pp. 25-31, Jul. 2013.

[5] https://neo4j.com/developer/cypher-query-language/, [accessed: Apr. 1, 2020]

[6] Jürgen Hölsch and et al., "On the Performance of Analytical and Pattern Matching Graph Queries in Neo4j and a Relational Database," Workshop Proceedings of the EDBT/ICDT 2017 Joint Conference, Venice, Italy, Mar. 2017.

[7] Shalini Batra and Charu Tyagi, "Comparative Analysis of Relational And Graph Databases," International Journal of Soft Computing and Engineering (IJSCE) Vol. 2, No. 2, pp. 509-512, May 2012.

[8] Alexandra Martinez, et al., "A Comparison between a Relational Database and a Graph Database in the context of a Personalized Cancer Treatment Application," Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Jun. 2016.

[9] Enhanced Entity-Relationship Model, http://jcsites.juniata.edu/faculty/rhodes/dbms/ermodel.htm, [accessed: Aug. 1, 2019]

[10] https://www.cs.uct.ac.za/mit_notes/database/htmls/chp07.html, [accessed: Aug. 1, 2019]

[11] Thomas Frisendal, Graph Data Modeling for NoSQL and SQL, Technics Publications, 2016.

[12] Neo4j, https://neo4j.com/, [accessed: Aug. 1, 2019]

[13] Andrey Gubichev and Manuel Then, "Graph Pattern Matching – Do We Have to Reinvent the Wheel?," GRADES'14, June 22-27 2014, Snowbird, UT, USA 2014.

[14] Renzo Angeles, "The Property Graph Database Model," Alberto Mendelzon International Workshop on Foundations of Data Management(AMW), Cali, Colombia, May 21-25, 2018.

[15] warm cache and cold cache, https://stackoverflow.com/questions/22756092/what-does-it-mean-by-cold-cache-and-warm-cache-concept, [accessed: Aug. 1, 2019]

Appendix(부록)

Cypher 및 SQL 질의문 - 패턴질의 타입 4가지 X (단순형, 복잡형) X (모델1, 모델2) : (Q1~Q8)
(Q5~Q8은 관계 모델에 대한 SQL 질의 포함)

Q1 - 패턴질의 타입 1 - 단순형(경로길이 1), (모델 타입1, 모델 타입 2) 질의 교수(정교수, id=0)와 교수의 (학부)강좌에 관한 사항을 보여라.
MATCH (a:Professor{name:'FullProfessor0'})--(b:Course{type:'Course'}) RETURN count(*)
MATCH (a:FullProfessor{name:'FullProfessor0'})--(b:Course) RETURN count(*)
Q2 - 패턴질의 타입 1 - 복잡형(경로길이 2), (모델 타입1, 모델 타입 2) 질의 교수(정교수, id=0)가 지도교수인 (학부)학생들의 수강강좌에 관한 사항을 보여라
MATCH (a:Professor{name:'FullProfessor0'})--(c:Student{type:'UndergraduateStudent'})--(d:Course{type:'Course'}) RETURN count(*)
MATCH (a:FullProfessor{name:'FullProfessor0'})--(b:UndergraduateStudent)--(c:Course) RETURN count(*)
Q3 - 패턴질의 타입 2 - 단순형(경로길이 2), , (모델 타입1, 모델 타입 2) 질의 교수(정교수, id=0)의 강좌와 강좌를 듣는 학생들을 보여라.
MATCH (a:Professor{name:'FullProfessor0'})-[:teacherOf]->(b)<-[:takesCourse]-(c) RETURN count(*)
MATCH (a:FullProfessor{name:'FullProfessor0'})-[:teacherOf]->(b)<-[:takesCourse]-(c) RETURN count(*)
Q4 - 패턴질의 타입 2 - 복잡형(경로길이 4), (모델 타입1, 모델 타입 2) 질의 교수(정교수, id=0)의 강좌를 듣는 학생의 지도교수의 박사학위 대학을 보여라.
MATCH (a:Professor{name:'FullProfessor0'})-[:teacherOf]->(b)<-[:takesCourse]-(c)-[:advisor]->(d)-[:doctoralDegreeFrom]->(e) RETURN count(*)
MATCH (a:FullProfessor{name:'FullProfessor0'})-[:teacherOf]->(b)<-[:takesCourse]-(c)-[:advisor]->(d)-[:doctoralDegreeFrom]->(e) RETURN count(*)
Q5 - 패턴질의 타입 3 - 단순형(경로길이 2), (모델 타입1, 모델 타입 2, SQL) 질의 교수(정교수, id=0)의 강좌와 강좌를 듣는 (학부)학생들을 보여라.
MATCH (a:Professor{name:'FullProfessor0'})-[:teacherOf]->(b:Course{type:'Course'})<-[:takesCourse]-(c:Student{type:'UndergraduateStudent'}) RETURN count(*)
MATCH (a:FullProfessor{name:'FullProfessor0'})-[:teacherOf]->(b:Course)<-[:takesCourse]-(c:UndergraduateStudent) RETURN count(*)
SELECT count(*) FROM teachers t, courses c, takescourse tk, students s WHERE t.ttype=3 AND t.name like 'FullProfessor0' AND t.id=c.teacherid AND c.ctype=0 AND c.id=tk.courseid AND tk.studid=s.id AND s.stype=0;
Q6 - 패턴질의 타입 3 - 복잡형(경로길이 4), (모델 타입1, 모델 타입 2, SQL) 질의 교수(정교수, id=0)의 강좌와 강좌를 듣는 (학부)학생들의 (부)교수인 지도교수의 석사학위 대학을 보여라.
MATCH (a:Professor{name:'FullProfessor0'})-[:teacherOf]->(b:Course{type:'Course'})<-[:takesCourse]-(c:Student{type:'UndergraduateStudent'})-[:advisor]->(d:Professor{type:'AssociateProfessor'})-[:mastersDegreeFrom]->(e:University) RETURN count(*)
MATCH (a:FullProfessor{name:'FullProfessor0'})-[:teacherOf]->(b:Course)<-[:takesCourse]-(c:UndergraduateStudent)-[:advisor]->(d:AssociateProfessor)-[:mastersDegreeFrom]->(e:University) RETURN count(*)

```
SELECT count(*) FROM teachers t, courses c, takescourse tk, students s, teachers tt, university u
WHERE t.ttype=3 AND t.name like 'FullProfessor0' AND t.id=c.teacherid AND c.ctype=0
    AND c.id=tk.courseid AND tk.studid=s.id AND s.stype=0 AND tt.ttype=2
    AND s.advisorid=tt.id AND tt.docd=u.universityid;
```

Q7 - 패턴질의 타입 4 - 단순형(사이클, 경로길이 3), (모델 타입1, 모델 타입 2, SQL) 질의
(학부)학생의 자신의 지도교수(정교수)가 강의하는 강좌를 듣는 (학부)학생을 찾아라.

```
MATCH
(a:Student{type:'UndergraduateStudent'})-[:takesCourse]->(b:Course{type:'Course'})<-[:teacherOf]-(c:Professor{type:'FullProfessor'})<-[:advisor]-(d:Student{type:'UndergraduateStudent'})
WHERE a=d
RETURN count(*)
```

```
MATCH
(a:UndergraduateStudent)-[:takesCourse]->(b:Course)<-[:teacherOf]-(c:FullProfessor)<-[:advisor]-(d:UndergraduateStudent)
WHERE a=d
RETURN count(*)
```

```
SELECT count(*) FROM students s, teachers t, courses c, takescourse tk, students ss
WHERE s.stype=0 AND s.advisorid=t.id AND t.ttype=3 AND t.id=c.teacherid AND c.ctype=0
    AND c.id=tk.courseid AND tk.studid=ss.id AND ss.stype=0 AND s.id=ss.id;
```

Q8 - 패턴질의 타입 4 - 복잡형(사이클, 경로길이 5), (모델 타입1, 모델 타입 2, SQL) 질의
자신의 논문 공저자 교수(정교수)의 강의를 수강하는 대학원생을 찾아라.

```
MATCH
(a:Student{type:'GraduateStudent'})<-[:publicationAuthor]-(b:Publication)-[:publicationAuthor]->(c:Professor{type:'FullProfessor'})-[:teacherOf]->(d:Course{type:'GraduateCourse'})<-[:takesCourse]-(e:Student{type:'GraduateStudent'})
WHERE a=e
RETURN count(*)
```

```
MATCH
(a:GraduateStudent)<-[:publicationAuthor]-(b:Publication)-[:publicationAuthor]->(c:FullProfessor)-[:teacherOf] ->(d:GraduateCourse)<-[:takesCourse]-(e:GraduateStudent)
WHERE a=e
RETURN count(*)
```

```
SELECT count(*) FROM students s1, publicationauthor pa, teachers t,
    courses c, takescourse tc, students s2
WHERE s1.stype=1
    AND s1.id=pa.authorid2 AND pa.authortype2=-1 AND pa.authortype=t.ttype
    AND pa.authorid=t.id AND t.ttype=3 AND t.id=c.teacherid AND c.ctype=1
    AND c.id=tc.courseid AND tc.studid=s2.id AND s2.stype=1
    AND s1.id=s2.id;
```

Authors

Uchang Park received the B.S., M.S. and Ph.D. degrees in Computer Science and Statistics from Seoul National University, Korea, in 1982, 1985 and 1993, respectively. Dr. Park joined the faculty of the Department of Computer Engineering at Duksung Women's University, Seoul, Korea, in 1988. He is interested in database query language, graph DBMS, machine learning.