효율적인 Regular Path Query 처리에 관한 연구 조사

이태성*, 백은진*, 황준승**, 김경민**, 한욱신***
*포항공과대학교 컴퓨터공학과
**포항공과대학교 창의 IT 융합공학과
***포항공과대학교 창의 IT 융합공학과/컴퓨터공학과
e-mail : tslee@dblab.postech.ac.kr, eunjinb@postech.ac.kr
jshwang@dblab.postech.ac.kr, kmkim@dblab.postech.ac.kr, wshan@postech.ac.kr

A Study for Efficient Regular Path Query Evaluation

TaeSung Lee*, Eun-Jin Baek*, Junseung Hwang**, Kyoungmin Kim**, Wook-Shin Han***
*Dept. of Computer Science, POSTECH
**Dept. of Creative IT Convergence Engineering, POSTECH
***Dept. of Creative IT Engineering/Dept. of Computer Science and Engineering, POSTECH

요 약

본 연구는 regular path query 를 효율적으로 처리하는 디스크 기반 시스템을 만들기 위해서는 그래프 데이터를 효율적으로 저장하여야 하며, regular path query 수행 시 발생하는 cost 가 작도록 하는 evaluation algorithm 이 필요하다. 이에 본 연구에서는 그래프데이터 저장 방법을 제안하고, regular path query 수행시 발생하는 cost 및 오버헤드를 분석한다.

1. 서론

시맨틱 웹, 소셜 네트워크 서비스와 같이 데이터가 그래프 형태로 나타내어지는 응용들이 빠르게 성장함에 따라 그래프 데이터를 효율적으로 처리하는 것이 중요해지고 있다[2-5]. 최근에는 이러한 그래프 데이터로부터 특정 조건을 만족하는 path 를 효율적으로 찾기 위한 연구들[2-5]이 활발히 이루어지고 있다. Regular path query 는 regular expression 이 query 로 주어졌을 때, data graph 에서 label 의 sequence 가 regular expression 을 만족하는 path 를 찾는 질의이다[2-5].

Regular path query 를 효율적으로 처리하는 시스템을 만들기 위해서는 효율적인 저장시스템이 필요하고, regular path query 수행 시 발생하는 cost 가 작도록 하는 cost-based algorithm 이 필요하다. 본 연구에서는 regular path query 를 효율적으로 처리하는 디스크 기반 시스템을 만들기 위하여 graph data 저장 방법을 제안하고, query 수행 시 발생하는 cost 및 오버헤드를 분석한다.

2. 문제 정의

2.1. 데이터 모델

본 연구에서는 사용하는 그래프 데이터의 데이터 모델은 기존 연구들[2,4,5]과 같이한다. 그래프 G = {V, Σ, E}로 정의되며, V 는 vertex 의 집합, Σ 는 symbol 의 집합, E 는 (v_i, s, v_j)인 edge 의 집합이다. ($v_i, v_j \in V$ and $a \in \Sigma$)

"이 논문은 삼성전자 미래기술육성센터의 지원을 받아 수행된 연구임" (과제 번호 SRFC-IT1401-04)

2.2. Regular Path Query

Regular path query 는 regular expression 으로 표현되며 regular expression 의 정의는 다음과 같다[1].

R = ε | s (s ∈ Σ) | R + R(alternation) |
R · R(concatenation) | R^*(Kleene Star)

그래프 데이터 G 에서 regular path query α 의 질의 결과를 $[\alpha]_G$라 할 때 regular path query 의 정의는 다음과 같다.

$[\varepsilon]_G = \{(u, u) \mid u \in V\}$
$[s]_G = \{(u, v) \mid u, v \in V \text{ and } s \in \Sigma \text{ and } (u, s, v) \in E\}$
$[\alpha+\beta]_G = [\alpha]_G \cup [\beta]_G$
$[\alpha \cdot \beta]_G = \{(u, w) \mid (u, v) \in [\alpha]_G \text{ and } (v, w) \in [\beta]_G\}$
$[\alpha^*]_G = [\varepsilon]_G \cup [\alpha]_G \cup [\alpha \cdot \alpha]_G \cup \cdots$

3. 관련 연구

Regular path query 를 효율적으로 처리하기 위한 연구는 index 를 사용하는 방법[2], SQL 을 사용하는 방법[4], automata 를 사용하는 방법[3,5]으로 분류할 수 있다.

먼저, Index 를 사용하는 방법[2]은 길이가 3 인 모든 path 를 인덱스로 만든 후 질의 시간에 이 인덱스들을 이용하여 query 를 수행한다. 그러나 이 방법은 선처리에 많은 시간이 필요하고, 인덱스를 구축하기 위해 많은 저장공간이 필요하다.

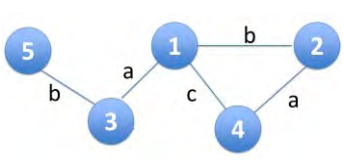
SQL 을 사용하는 방법[4]은 regular path query 의 각 연산을 recursive SQL 로 변환하여 수행한다. 이 방법은 DBMS 를 저장시스템으로 이용할 수 있는 장점이 있지만 불필요한 element 들에 대해 join 연산을 수행하는 오버헤드가 발생한다.

Automata 를 사용하는 방법[3,5]은 regular path query 를 automata 로 변환한 후 이를 recognizer 로 graph 를 traverse 함으로써 query 를 수행한다. 이 방법은 불필요한 element 들과의 join 연산은 수행하지 않지만 subquery 의 결과를 반복해서 구하는 오버헤드가 있다. 예를 들어, 질의 abc 의 결과가 {(1, 3), (3, 2)}인 그래프 데이터에서 질의 $(abc)^*$가 주어졌을 때 abc 의 결과를 찾기 위해 path (3, 2)를 찾기 위해 한번 traverse 한 경로들을 abcabc 를 찾을 때 path (1, 3)으로부터 반복해서 traverse 한다. 이러한 문제를 해결하기 위한 연구[5]로, Kleene star 내부의 질의를 먼저 수행하고 이 결과를 이용할 수 있는 query plan 이 제안되었지만 이를 optimization 하는 알고리즘이 필요하다.

4. 제안하는 방법

4.1. Data 저장 방법

Regular path query 를 효율적으로 처리하기 위해서는 주어진 symbol 이 label 인 edge 들의 list 를 빠르게 찾을 수 있어야 한다. 이를 위해 그림 1 과 같이 symbol table 에 각 label 별로 별도의 edge 들의 list 를 저장한다.



label	(source, destination)
a	(1, 3), (2, 4), ...
b	(1, 2), (2, 1), ...
c	(1, 4), (4, 1)

(그림 1) Symbol 테이블의 예

Kleene star 연산은 regular path query 수행 시 병목이 될 수 있다. 이러한 문제를 해결하기 위해 각 symbol 의 Kleene star 연산의 결과를 선처리하여 별도로 저장하고 질의 시간에 이를 이용하여 질의를 수행한다.

4.2. Cost 및 오버헤드 분석

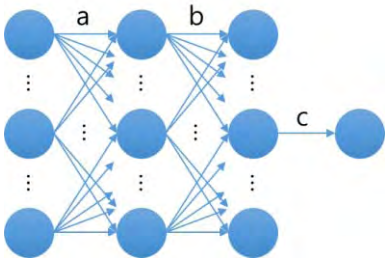
본 연구에서는 regular path query 를 효율적으로 처리하는 디스크 기반 시스템을 만들기 위하여 graph data 저장 방법을 제안하고, query 수행 시 발생하는 cost 및 오버헤드를 분석한다.

Duplicate subquery

주어진 Query 에 길이가 2 이상인 동일한 substring 이 있을 경우 이 substring 의 결과를 반복해서 찾는 것은 비효율적이다. 더욱이, 이 substring 이 Kleene star 일 경우 결과를 반복해서 찾는 것은 성능의 급격한 저하를 가져올 수 있다. 또한, query 를 연속해서 수행하거나 여러 query 를 동시에 수행할 때 각각의 동일한 subquery 들을 중복해서 수행하는 것은 비효율적이다. 따라서, 중복된 subquery 중 하나를 먼저 수행하고 먼저 수행한 결과를 이용하여 질의를 수행할 수 있도록 cost model 을 만들 필요가 있다.

Matching order

Regular path query 는 symbol 의 matching order 에 따라 탐색해야 하는 그래프의 크기는 달라진다. 그림 2 와 같은 그래프 데이터에서 query abc 가 주어졌을 때 a 부터 매칭할 경우 전체 그래프를 탐색해야 하지만 c 부터 탐색할 경우 label 이 b 인 많은 edge 들을 탐색할 필요가 없다. 즉, 매칭 순서를 정하는 것이 regular path query 를 효율적으로 수행하는데 큰 영향을 미친다. 따라서, edge 의 수가 작은 label 들이 먼저 matching 될 수 있도록 cost model 을 만들 필요가 있다.



(그림 2) 그래프 데이터의 예

5. Conclusion & Future work

본 연구에서는 regular path query 를 효율적으로 처리하는 디스크 기반 시스템을 만들기 위해 graph data 저장 방법을 제안하고, 효율적인 query 수행 알고리즘을 제안하기 위해 regular path query 수행 시 발생하는 cost 및 오버헤드를 분석하였다.

이 연구의 연장으로 다음의 업무들을 수행할 필요가 있다. 먼저, duplicate subquery 문제와 matching order 문제를 효율적으로 반영할 수 있는 cost-based algorithm 을 개발한다. 이 후 제안한 저장 시스템을 기반으로 하고, 개발한 cost-based algorithm 을 사용하는 시스템을 만들고, 이를 기존 시스템들과 비교 분석한다.

참고문헌

[1] Gelade, W., and Neven, F. “Succinctness of the Complement and Intersection of Regular Expression.” In *Proc. 25th Int’l Symposium on Theoretical Aspects of Coomputer Science (STACS 2008)*, pp. 325-336, Bordeaux, France, February 21-23, 2008.

[2] Fletcher, G., Peters J. and Poulovassilis, A., “Efficient regular path query evaluation using path indexes” In *Proc. 19th Int’l Conf. on Extending Database Technology*, pp. 636-639, Bordeaux, France, March 15-16, 2016.

[3] Koschmieder, A. and Leser, U., “Regular path queries on large graphs” In *Int’l Conf. on Scientific and Statistical Database Management*. pp. 177-194, Chania, Crete, Greece, June 25-27, 2012.

[4] Yakovets, N., Godfrey, P. and Gryz, J., “Evaluation of SPARQL Property Paths via Recursive SQL” In *7th Alberto Mendelzon Int’l Workshop on Foundations of Data Management*, Vol. 1087, Puebla/Cholula, Mexico, May 21-23, 2013.

[5] Yakovets, N., Godfrey, P. and Gryz, J. "Query planning for evaluating SPARQL property paths" In *Proc. 2016 ACM SIGMOD Int’l Conf. on Management of* Data, pp. 1875-1889, San Francisco, USA, June 26-July 1, 2016.