# 사용자 기반 Capability를 이용한 질의 재작성 기법

박 석 안규연 [0]
서 강 대 학 교 전 산 학 과
{spark, rigel}@dblab.sogang.ac.kr

# Query rewriting using user based capability

Seok Park Kyu_Yearn Ahn [0]
Dept. of Computer Science, Sogang University

**요 약**

데이터베이스 시스템은 웹의 발달과 함께 웹 데이터베이스라는 새로운 단계로 진화하고 있다. 웹 데이터베이스에서는 다양한 데이터 소스의 통합이 가장 큰 이슈로 미디에이터에서 각 래퍼가 가져온 스키마를 통합하는 작업을 수행한다. 미디에이터에서는 이와 같이 데이터 소스의 통합 외에 사용자에게 들어온 질의를 각 데이터 소스에 대하여 재작성 하는 기능이 중요하고 이에 대한 다양한 연구가 진행 중이다. 이때 질의를 각 소스가 처리할 수 있는 부질의로 재작성 할 때 필요한 개념이 capability이다.

본 논문에서는 capability를 기반으로 한 질의 시스템에서 사용자에게 제공되는 뷰와 질의에 대해서도 capability를 적용한 사용자 기반 capability 기법을 제안한다. 지금까지 타 연구에서는 소스에 대한 capability만을 고려하여 질의를 재작성 하였지만 본 논문에서는 사용자 기반의 capability를 함께 고려함으로써 사용자 요구사항을 충족시키면서도 비교적 간단한 분석과 매칭 과정에 의해 컴파일 시간 내에 계산이 가능하도록 하는 알고리즘을 제시한다. 또한 네트워크의 문제 등으로 인해 소스 중 일부분이 사용할 수 없는 상황인 경우 소스에 대해 active selection이 가능하도록 하여 대체 질의를 생성할 수 있게 하였다. 간단한 구현과 제시한 기법이 비용을 절감하는 실제적인 방법이며 타 연구와의 비교를 통해 좋은 성능을 가짐을 보인다.

## 1. 서론

웹 데이터베이스는 래퍼와 미디에이터로 구성되는 3-tier 형태를 띈다. 미디에이터의 모델을 기반으로 하여 구성된 뷰를 통해 사용자는 질의를 던질 수 있으며 미디에이터는 질의를 받고 이를 분해하여야 한다. 이러한 경우 미디에이터는 각 소스가 가지고 있는 질의에 대한 정보를 알아야만 해당 소스에 정확한 질의를 분해할 수 있으며 사용자의 질의가 소스의 질의 능력보다 큰 범위를 가지는 경우에는 얻어진 결과값에 적절한 연산을 가할 수 있는 능력 또한 요구된다[1].

미디에이터에서 사용자에게 들어온 질의를 분해하여 각 래퍼에게 할당하는 방법에 대해서는 여러 가지 연구가 진행 중이며 다수의 연구들은 주로 소스의 내용이나 사용자에게 제공되는 통합된 뷰와 소스와의 관계에 대해서 초점을 맞추어 진행되었다. 그러나 이러한 방식은 소스가 처리하지 못하는 질의를 생성해 낼 여지를 제공하게 되고 이러한 문제를 처리하기 위한 질의 재작성 과정에 대한 연구가 미디에이터에서 각 소스가 가지고 있는 질의 능력을 나타내는 방법으로 **capability** 개념을 이용한 질의 재작성이다[2]. **Capability** 개념은 소스가 제공하는 정보를 추출할 수 있는 키 필드와 해당 필드를 이용하여 추출되는 이외의 정보를 구분하여 묘사하는 방식이다. 이러한 접근 방식은 실제 소스에서 처리 가능한 질의를 나누어 줄 수 있다는 점에서 실질적인 접근 방식이라 볼 수 있다. 현재 **capability** 기반의 질의 재작성은 소스에서 제공하는 **capability**를 그대로 사용하는 소스 기반 **capability** 기법이라 할 수 있다.

본 논문에서는 기존의 연구에서 나타내지 못했던 사용자에게 제공되는 뷰에 capability를 적용함으로써 얻어질 수 있는 장점과 이러한 사용자 기반의 capability를 적용하는 시스템의 구조 및 기존 미디에이터에서 보유하고 있는 소스의 capability에 대한 정보인 소스 기반 capability를 매칭시키는 알고리즘에 대하여 알아보도록 하겠다.

본 논문의 나머지는 다음과 같이 구성되어 있다. 2장에서는 capability를 이용하여 질의를 재작성 하는

과정에서 일어나는 현실세계의 상황과 문제점을 지적하고, 이러한 문제점을 보완하기 위해 사용자 기반 capability란 개념을 포함한 capability를 가미한 클래스 기반의 모델을 제안하고, 사용자 기반 capability의 개념과 알고리즘을 살펴본다. 3장에서는 제안하는 모델을 기존의 타 모델과 비교하여 장단점을 평가하여 분석한다. 마지막으로 4장에서는 결론과 추후 연구과제를 밝히도록 하겠다.

## 2. 관련연구

### 2.1 연구의 동기

현실적으로 사용되는 뷰와 질의의 형태는 다음과 같은 특성을 띄고 있다고 볼 수 있다.

- 사용자에게 제공되는 질의 필드 고정
- 제공할 속성과 소스와의 포함관계
- 소스가 일시적으로 사용할 수 없는 경우

본 논문에서 제안하는 방법은 개발자가 질의를 설계할 때 질의에 capability를 접합시킨 방법으로 이를 소스 기반 capability와 구분하여 사용자 기반 capability라고 부르기로 하겠다.

본 논문에서 제안하는 이러한 방법은 미리 정의해 놓은 클래스 구조에 맞게 웹 페이지의 정보를 맞추어 가지고 오는 방법을 사용함으로써 부분적인 유연성을 희생하고 전체적인 통일성과 스키마를 명시할 수 있는 클래스 기반 시스템에 적합하다. 이러한 스키마가 제공되어 소스의 스키마에 상관없이 전체적으로 통일된 구조와 형태를 취하고 있는 클래스 기반의 시스템에서는 뷰를 정의하고 질의를 설계하는 것이 보다 명확하여지고 따라서 개발자는 보다 쉽게 사용자에게 제공할 폼 문의 형태를 만들어 낼 수 있게 된다.

### 2.2 사용 용어

본 논문에서 주로 사용하게 될 용어는 다음과 같다.

- 키 필드 : 검색 키. Capability를 적용한 뷰 정의에서 변수 이름 앞에 '$'를 덧붙여서 나타낸다.
- 속성 : 키 필드로 검색하여 추출된 다른 필드 값들
- 소스 capability : 소스가 제공하는 키 필드와 키 필드로 추출해 낼 수 있는 다른 속성으로 ($A, B, C, D, E)과 같이 표기된다.
- user capability : 사용자에게 제공할 키 필드와 속성을 추가한 뷰 정의로 ($A', B', C', D', E') 과 같이 표기된다.
- 커버 : 하나의 소스 키 필드를 가지고 추출하는 모든 소스.

### 2.3 시스템 구조

개발자는 구조 생성기에서 소스의 스키마를 참조하여 class wizard에서 클래스를 생성하고 래퍼는 소스의 capability를 추출하여 query generator에 저장한다. 개발자는 Application generator에서 질의를 생성하면서 user capability를 생성하고 이를 capability based rewriter에서 소스 capability와 매칭을 수행한다. 이러한 과정은 시스템 셋업시, 소스가 추가, 제거, 변경된 경우, 수행시간 시 지정된 소스가 사용 불가능한 경우 수행된다.

먼저 사용자의 요구를 분석하여 만든 사용자 기반 capability를 분석하는 단계를 거쳐 실질적으로 수행될 plan을 만들고 이를 수행하는 엔진에 보낸다. 즉 실제로 매칭과 시퀀싱이 결정되는 부분은 plan generator라고 할 수 있는데 이 부분은 크게 세 부분으로 이루어져 있다. 먼저 analyzer에서는 매칭에 필요한 리스트를 생성하고 질의가 조건에 정확히 일치하지 않는 경우 어떤 순서로 행할 것인가를 결정하며 이를 이용하여 matcher에서는 작업의 시퀀싱을 결정하고 기타 부가적인 연산을 composer에서 행하는 것이다.

### 2.4 알고리즘

#### 2.4.1 Analyzing algorithm

**[단계 1] 키 필드 점검**

1.1 Listing : 속성 별로 해당 소스를 배열

각 속성을 가지고 있는 소스를 나열한다. 즉 해당 속성에 저장되어 있는 소스를 테이블 형태를 사용하여 나열한다.

1.2 Checking : Key field가 2개 이상 존재하는지 체크

속성 중 사용자 기반 capability의 키필드에 해당하는 속성에 대하여 사용자 기반 capability의 키필드가 2개 이상 존재하는지 체크한다. 이런 경우는 사용자에게 제공되어 받아 들이는 질의 값이 2개 이상인 경우로써 어떤 질의 값부터 처리할 것인지를 결정하여야 한다. 키 필드가 1개만 존재하는 경우 알고리즘의 수행을 종료하며 그렇지 않은 경우, 즉 2개 이상인 경우는 질의의 처리를 단계화 하기 위하여 2단계로 넘어간다.

**[단계 2] 질의 단계화**

2개 이상의 user 키 필드가 존재하는 경우에는 질의가 2개 이상으로 나누어 진다. 그러나 해당하는 질의들 중에서 먼저 수행되어야 할 것과 동시에 수행될 수 있는 것들이 있으며 이를 구하는 과정을 거쳐야 한다.

2.1 [case 1] 키 필드 중 하나에만 소스의 키 필드가 존재하는 경우

소스 키 필드가 존재하는 user 키 필드 값을 수행하여 진행한 후 이 값을 통해 얻을 수 있는 다른 키 필드의 값을 체크하여 결과를 추출한다.

2.2 [case 2] 키 필드 중 두개 이상에 대해 소스의 키 필드가 존재하는 경우

소스 키 필드가 존재하는 키 필드를 동시에 수행하여 나간다. 이 때 커버되는 범위가 넓은 쪽을 우선으로 한다.

#### 2.4.2 Matching algorithm

**[단계 1] 키 필드 점검**

1.1 user 키 필드에 해당하는 소스 키 필드가 존재하는지 체크

user 키 필드과 일치하는 소스 키 필드가 존재한다는 것은 사용자에게 입력 받은 값을 바로 이용하여 소스에 대해 질의할 수 있다는 것이다. 소스 키 필드가 존재한다면 다음 단계로 넘어가고 존재하지 않는 다면 단계 3으로 간다.

**[단계 2] 키 필드 처리**

단계 2는 사용자 기반 capability의 키 필드에 해당하는 소스 based capability의 키 필드가 존재하는 경우로 사용자 기반 capability의 속성이 모두 커버될 때까지 반복한다.

2.1 커버되는 속성을 체크

user 키 필드와 일치하는 소스 키 필드가 있다면 소스에서 추출되는 다른 속성을 살펴 보아서 user 속성이 얼마나 커버되는지를 살핀다. 만약 user 속성이 모두 커버된다면 이 뷰가 제공하는 값은 하나의 소스에서 모두 얻을 수 있음을 의미하며 알고리즘을 종료한다. 그러나

그렇지 않는 경우가 일반적이라 볼 수 있다.

2.2 커버되는 속성을 이용하여 다른 소스에서 속성 추출

위에서 커버되는 속성 중에서 다른 소스의 키필드가 존재하는지를 체크한다. 존재한다면 질의는 단계적으로 수행될 수 있음을 뜻하고 다른 소스의 키필드로 추출될 수 있는 속성들을 체크하여 사용자 기반 capability의 속성이 모두 커버될 때 까지 위의 과정을 반복한다

**[단계 3] 소스 list 생성**

단계 3은 사용자 기반 capability의 키 필드에 해당하는 소스 based capability의 키 필드가 존재하지 않는 경우로 개발자는 사용자가 던진 키 필드 값으로 바로 소스 based capability에 대해 질의를 수행할 수 없기 때문에 소스 키 필드에 대한 list를 생성하여야 한다.

3.1 user 키 필드 값에 대한 소스 키 필드 값을 list로 작성

3.2 list를 이용하여 질의

수행시간에는 user 키 필드 값을 이용하여 이를 list에서 비교하여 해당 소스 키 필드값을 추출한 후 추출된 값을 소스 키 필드로 이용하여 다른 속성을 추출한다. 이 후의 과정은 단계 1.2부터 반복하여 수행한다.

## 3. 비교

이 장에서는 기존에 존재하는 capability 모델과의 비교 분석을 통하여 제안한 모델의 장단점에 대하여 알아본다.

비교 기준은 크게 제공되는 질의에 대한 측면과 재작성 측면에서 선정해 볼 수 있다. 기준은 있다. 즉 질의의 재작성이 일어날 때 재작성에 대한 비용과 질의 자체에 대한 내용에 대해 각기 살펴본다.

**[표 1] 제안한 기법과 다른 기법과의 비교**

| | 평가항목 | TSIMMIS | Mediator | Relation | 제안방법 |
|---|---|---|---|---|---|
| 질의 재작성 | 질의 재작성 시점 | Run time | Compile time | Compile time | Compile(run) time |
| | 대체 질의 생성 여부 | 불가 | 불가 | 불가 | 가능 |
| | 지원 질의 개수 | 많음 | 적음 | 적음 | 적음 |
| | 질의 재작성 비용 | 적음 | - | 보통 | 보통 |
| 질의 | 계층화 | 불가 | 가능 | 불가 | 가능 |
| | 사용자 요구 반영 | 불가 | 불가 | 불가 | 가능 |
| | 질의 다양성 | 낮음 | 낮음 | 낮음 | 높음 |

위의 표를 통해 비교해 본다면 제안한 기법은 컴파일 시간에 계산이 가능하며 계산시에도 지원 질의가 적어 단순한 비교 연산만으로 후보 재작성 질의를 생성할 수 있는 비용적인 효과와 함께 여러가지 후보 소스를 이용한 대체 질의를 생성하여 유연성을 증대하여 질의의 처리를 원활하게 해준다. 그러나 질의를 재작성 할 때에 대체 질의를 생성하기 위해 소스와 속성의 부가적인 리스트를 유지하여야 되는 등의 재작성 비용은 증가하게 된다.

이와 같은 질의의 재작성 외에도 질의 생성에 있어서도 설계단계에서 사용자의 요구사항을 반영하므로 사용자의 만족도를 향상시키며 리스트를 이용하여 소스 키 필드 외의 항목을 사용자에게 키 필드로 제공할 수 있으므로 다양한 질의를 제공한다. 또한 이러한 사용자에게 제공하는 뷰 정의 자체가 미디에이터의 capability를 포함하고 있으므로 계층화가 가능하여 다른 미디에이팅 시스템의 소스로 사용될 수 있게 된다.

## 4. 결 론

질의 재작성은 통합 모델의 구조 정보를 이용하여 질의를 분해하는 과정으로 각각의 소스에게 실행해야 하는 질의를 할당하는 일련의 절차라고도 볼 수 있다. 이 과정에 있어 기존의 질의 재작성과는 다르게 소스의 다양성을 염두에 둔 질의 재작성 방법이 필요하게 되었는데 그것이 capability를 기반으로 한 질의 재작성 방법이고 지금까지의 연구는 소스의 질의 능력을 표현함에 있어 capability를 첨가한 수준에서 이루어져 왔지만 본 논문에서는 사용자에게 제공하는 뷰 및 질의에 대하여도 capability를 추가하고자 하였다.

본 논문에서 제시된 뷰 생성 기법은 단순히 원하는 속성을 선택하면 해당 속성과 이에 대한 소스를 사용하여 최적화된 질의 재작성을 이루는 방법으로 여타 방법에 비하여 단순하고 효율적이다. 이와 같이 뷰 정의를 사용하지 않으면서 새로운 뷰를 만들어 내는 과정에서 뷰에 부착된 capability 정보를 통해 미리 분석이 가능하게 되므로 컴파일 시간에 계산 가능하며 따라서 사용자가 해당 정보를 검색하고자 하는 시점에 계산이 생략되기 때문에 보다 빠른 처리가 가능해 진다. 또한 조인 연산을 사용하는 것이 아니라 간단한 셀렉트 연산을 사용하므로 질의를 조합하는 비용이 저렴하다.

고정적인 뷰 정의를 사용한 것이 아니므로 네트워크의 이상이 있거나 소스에 문제가 발생하여 일부 소스가 사용할 수 없는 상황인 경우에도 대체 질의를 생성할 수 있으므로 소스에 대한 질의 능력을 보다 풍부하게 가지고 있다고 말할 수 있다.

이후 추가 진행 사항으로는 다음과 같은 것들이 있다. 미디에이터 수준의 연산을 추가하는 것이 필요하다. 또한 부분 결과를 생성할 수 있는 과정에 대한 추가 연구가 필요하며 소스에서 제공되는 셀렉션 이외의 연산을 모두 포함하도록 연구가 진행되어야 할 것이다.

## 5. 참고문헌

[1] Chen Li et al., "Capability based mediation in TSIMMIS", *ACM SIGMOD RECORD proceedings*, pp. 564-566, 1998

[2] Ramana Yerneni, Chen Li, Hector Garcia-Molina and Jeffrey Ullman, "Computing capabilities of mediators", *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pp. 443 - 454, 1999

[3] 최승일, 박석, "클래스 분류화를 통한 웹 데이터 구조화 기법", *Korean database conference 2000 학술발표 논문집*, pp. 250-261, 2000

[4] Yannis Papakonstantinuo, Ashish Gupta, Laura Haas, "Capabilities-based query rewriting in mediator systems", *Parallel and distributed information system(PDIS)*, 1996