CNF의 수평적 분리를 이용한 공간 질의 최적화 기법의 제안

이환재*°,정보흥*,조숙경*,이순조**,배해영*
*인하대학교 전자계산공학과
**서원대학교 컴퓨터 정보통신 공학부
xcom73@netian.com

The Design of Spatial Query Optimization Technique using Horizontal Splitting of CNF

Hwan-Jae Lee, Bo-Heung Jung, Sook-Kyoung Cho, Soon-Jo Lee, Hae-Young Bae
*Dept. of Computer Science and Engineering, Inha University
**School of Computer, Information & Communication, Seowon University

요약

공간 데이터베이스 시스템에서의 질의처리 과정 중 질의 재작성 과정에 의해 다중 블록 질의가 단일 블록으로 변환되면 공간 서술자와 비공간 서술자가 OR와 AND에 의해 연결되어있는 복잡한 CNF가 생성된다. CNF 내의 공간 서술자는 공간연산의 정제단계의 수행 비용이 비공간 연산에 비해 상당히 많이 들기 때문에 비공간 서술자와는 다른 최적화 기법이 필요하다. 본 논문에서는 공간 서술자가 포함된 복잡한 CNF를 수평적으로 분리하여 질의를 재작성 하고 수행순서를 재조정하는 기법을 제안한다. 제안하는 기법은 원시 CNF를 수행 비용이 상대적으로 적은 전처리 단계의 CNF와 이에 비해 수행비용이 많이 드는 후처리 단계의 CNF로 분리하고 질의를 재작성 한 후 비용 모델에 의거해서 실행 트리를 최적화 한다. 본 논문에서 제시하는 기법은 질의 최적화 단계에서 공간연산의 단계별 실행 특성을 감안한 효율적인 실행 계획 생성이 가능하다는 장점이 있다.

1. 서론

최근 들어 많은 데이터베이스 응용분야에서 비공간 데이터베이스와 공간 데이터베이스를 하나의 시스템에 구축하여 운용하는 경우가 많아지고 있다[3]. 두 종류의 데이터에 대한 질의에서는 공간서술자가 포함된 조건절이 다른 서술자보다 많은 부하를 발생시킨다. 또한 공간 뷰에 대한 질의나 여러 개의 뷰 질의를 가진 형태의 질의, 그리고 사용자 정의 함수 등의 다중 블록 형태의 질의가 단일 블록 형태의 질의로 바뀌면 조건절은 하나 이상의 공간 및 비공간 조건들이 AND 와 OR로 연결되어 복잡하고 길어진다[1]. 이런 형태의 질의는 기존의 질의 최적화 방법과는 다른 최적화 기법에 대한 연구의 필요성을 증가시킨다.

본 논문은[1] 공간 데이터베이스 시스템에서 복잡한 CNF로 구성된 조건절을 이와 동등한 2개의 조건절로 분리하여 재작성하고 재배치하는 CSQR(CNF Splitting And Query Rewriting) 기법을 제안한다. CSQR 기법은 공간과 비공간 서술자가 섞여있는 질의를 효율적으로 처리할 수 있는 통합구조의 공간 데이터베이스 시스템에 적용되는 질의 최적화 기법이다. CSQR 기법은 원시 CNF안에 있는 OR들만의 묶음을 공간 및 비공간 조건의 포함 유형에 따라 3가지로 분류하여, 원시 CNF와 동등한 2개의 새로운 CNF로 분리한다. 분리된 CNF중 하나는 수행 비용이 적은 전처리 단계 CNF이고 다른 하나는 비용이 많이 드는 후처리 단계 CNF이며, 이들은 서로 다른 대수노드에 포함된다. 전처리 단계의 CNF는 OR들만의 묶음 단위로 여러 개의 죠인과 스캔 대수 노드에 분산되어져서 각각 최적화 되며, 후처리 단계의 CNF는 새로운 스캔 대수 노드에 실린 후 실행이 지연되도록 재배치 된다.

CSQR 기법의 장점은 실행 비용이 비공간 조건보다 많이 드는 공간조건의 수행 중 정제단계를 지연할 수 있도록 실행 트리 생성시에 이를 분리하여 최적화 계획을 세울 수 있다는 점이다.

본 논문의 구성은 다음과 같다. 2장에서는 수행 비용이 많이 드는 서술자의 이동을 이용하는 기존의 질의 최적화 기법들에 관해 설명하고, 3장에서는 본 논문에서 제안하는 CSQR 기법의 핵심인 CNF 분리 알고리즘에 대한 설명과 이를 실제 대수 트리에 적용하는 방법을 설명한다. 4장에서는 결론과 향후 연구과제를 제시한다.

2. 수행 비용이 많이 드는 서술자에 대한 질의 최적화 기법

수행비용이 많이 드는 서술자에 대한 질의 최적화는 일반적인 서술자에 적용되는 푸쉬 다운(Push-Down) 기법과는 다르게 풀 업(Pull-Up) 기법을 사용한다.

본 장에서는 비용이 많이 드는 서술자에 대한 질의 최적화 관련 연구로써 풀 업 기법을 기반으로 하여 서술자의 실행 순서를 지연시키는 POSTGRES의 서술자 이동 기법과 공간 연산의 분리 기법을 사용한 ESFAR 기법에 대해 설명한다.

2.1 POSTGRES 에서의 서술자 이동 기법

POSTGRES에서는 비용이 많이 드는 서술자에 대한 질의 최적화의 기본적 원칙으로 서술자 이동 기법(Predicate Migration)을 사용한다[6]. 이 기법은 수행 비용이 많이 드는 서술자를 원시 조건절에서 분리시킨 후, 실행 트리의 상위로 이동시킴으로써 질의 수행 시간을 단축시킨다. 하지만, 이 방법은 원시 CNF에 변형을 가하지 않고 단순히 서술자를 이동시킴으로써 이동하는 서술자가 질의 수행의 초기에 입력 후보객체를 줄이지 못하는 단점이 있다.

2.2 ESFAR (Early Separation of Filter And Refinement)

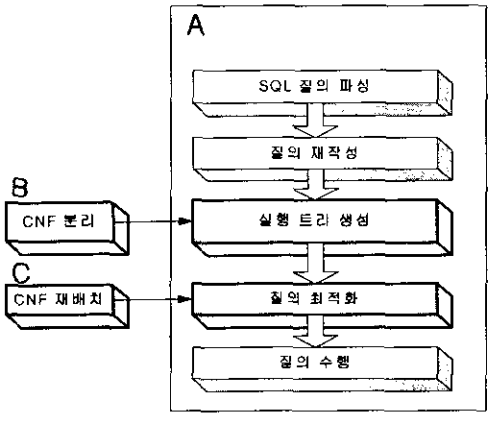
ESFAR[2] 기법은 공간 서술자를 여과 서술자와 정제 서술자로

[1] 본 연구는 정보통신부의 대학 S/W 연구센터 지원사업의 연구 결과임

분리한 후 재배치하는 기법이다. ESFAR 기법은 공간연산의 수행 단계 중 비용이 많이 드는 단계인 정제 과정을 질의 수행 중이 아닌 최적화 과정에서 분리하여 지연시킴으로써, 실행 시에 정제 과정에 들어오는 후보객체의 수를 줄이는 기법이다. 하지만 ESFAR에서는 공간 및 비공간 조건들이 AND와 OR로 복잡하게 얽인 CNF형태의 조건절에 대해서는 고려가 되어있지 않다. 또한, 공간 조건과 비공간 조건을 분리하여 수행한 후 결과로 생성된 임시 테이블의 OID들의 교집합을 구하는 방법(OID Join, OID Intersection)[3]을 사용하기 때문에, 공간 조건과 비공간 조건을 하나의 CNF에서 동시 비교 가능한 통합구조의 공간 데이터베이스 시스템에 적합하지 않다.

3. CSQR 기법

일반적인 데이터베이스 시스템에서의 질의처리는 [그림 1]의 A 박스와 같이 SQL 질의의 파싱, 질의 재작성, 실행 트리의 생성, 질의 최적화, 그리고 질의 수행의 순서로 수행된다. CSQR 기법은 기존의 실행 트리 생성 단계에 [그림 1]의 B과정을 추가하여 조건절인 CNF를 CNF 분리 알고리즘을 이용하여 분리한다. 분리된 각각의 CNF들은 질의 최적화 단계에 추가된 [그림 1]의 C과정을 통하여 수행 비용에 따라 재배치 된다. 본 장에서는 B과정에서 사용하는 CNF 분리 알고리즘을 제안하고, 이를 이용한 대수노드의 분리 및 재배치 방법, 그리고 실제로 적용한 예를 설명한다.



[그림 1] CSQR 기법을 적용한 질의 처리 수행과정

3.1 CNF 분리 알고리즘

일반적으로 사용자가 던진 질의 안에 있는 모든 조건절은 효율적인 수행을 위하여 실행트리 생성시에 CNF 또는 DNF (Disjunctive Normal Form)형태로 변형된다. 공간 데이터베이스에 대한 질의의 조건절을 CNF로 만들면 하나의 컨정션(Conjunction, OR 연산자들만의 묶음) 속에 공간과 비공간 서술자들이 섞여서 존재하게 된다. 이런 컨정션은 비공간 서술자만을 가진 것(순수 비공간 컨정션), 비공간과 공간 서술자를 모두 가진 것(공간 비공간 혼합 컨정션), 공간 서술자만을 가진 것(순수 공간 컨정션) 이렇게 세 가지로 나누어 진다. CSQR 기법에서는 비공간 컨정션은 그대로 유지하고, 순수 공간 컨정션과 공간 비공간 혼합 컨정션은 2개로 분리하는 알고리즘을 사용한다.

CNF 분리 알고리즘에서는 효율적인 분리를 위하여 다음의 세가지를 고려한다. 첫째, 하나의 컨정션 내의 모든 서술자들은 OR로 연결되어있기 때문에 특정 서술자가 만족되면 그 뒤에 있는 서술자들은 수행될 필요가 없다. 따라서, 비용이 많이 드는 공간 서술자는 같은 컨정션 내에서도 비공간 서술자들보다 뒤쪽에 배치되어야 한다. CSQR 기법은 CNF 분리 알고리즘의 첫 단계로 컨정션 내의 모든 공간 서술자들을 비공간 서술자들 보다 뒤로 이동시킨다

둘째, CNF 분리 알고리즘은 질의의 효율적 수행을 위하여 단순히 컨정션 단위의 분리가 아닌, 컨정션 내의 연산자의 수행단계 단위의 분리를 한다. 공간연산자는 본 알고리즘에 의해 여과 연산자와 정제 연산자로 분리된다. 여과 연산자는 일반적으로 색인을 이용하여 구현되는데, 통합구조의 공간 데이터베이스 시스템에서 공간과 비공간 서술자가 섞여 있는 조건절의 수행 시에 B-Tree등의 비공간 색인을 사용하게 되면 기존의 R-Tree와 같은 공간 색인을 사용한 여과연산이 불가능해 진다. 본 논문에서는 공간객체의 저장구조에 여과연산을 위한 추가적인 정보(예를 들면 MBR(Minimum Bounding Rectangle))를 가지고 있어서 이를 이용하여 여과 연산이 가능한 경우를 가정하며, 이를 통하여 추가적인 연산비용이 드는 OID 죠인이나 OID 인터섹션 기법의 사용을 방지한다.

셋째, CNF의 분리시에 공간 비공간 혼합 컨정션은 두 종류의 서술자가 컨정션 단위로 분리될 수 없기 때문에, 다음의 세 가지 방법 중 한가지로 처리해야 한다.

- **2개의 노드로 분리 수행** : 공간 조건과 비공간 조건이 섞인 컨정션에 대해서는 각각의 수행을 따로 한 후 결과 OID들의 합집합을 구해서 처리하는 방법이다. 이 방법은 두개로 나뉘어진 조건의 수행비용 외에 OID들의 합집합을 구하는 또 다른 대수노드가 추가되어야 하는데, 이 대수 노드의 처리 시에는 결과 테이블에 대한 정렬 혹은 색인 구축이 필요하기 때문에 무시하지 못할 추가 비용이 들어 적당하지 않다.
- **순수 비공간 컨정션과 동일하게 처리** : 정제를 포함한 공간연산의 모든 단계를 테이블의 스캔 단계에서 수행하기 때문에 공간과 비공간 혼합 컨정션의 공간 서술자를 위한 별도의 최적화를 하지 않는 것과 동일하다.
- **순수 공간 조건과 동일하게 처리** : 이 방법은 OR연산의 특성상 공간 비공간 혼합 컨정션안에 있는 비공간 조건이 전,후처리 모든 단계에서 매번 공간 조건과 같이 계산되어야 하기 때문에 이를 위한 추가 비용이 필요하다.

위의 세가지 방법 중 마지막의 비공간 서술자의 중복된 수행에 의한 추가 비용은 다른 2개의 방법에 의해 발생되는 공간 연산 비용에 비하면 무시할 수 있을 만큼 작으므로 이 방법이 가장 적당하다. CSQR 기법의 CNF 분리 알고리즘에서는 공간과 비공간 서술자가 혼합된 컨정션에 대해서 순수 공간 컨정션과 동일하게 처리한다.

위의 사항을 고려하여 CSQR 기법에서는 다음의 [알고리즘 1]과 같은 CNF 분리 알고리즘을 사용한다.

[알고리즘 1] SPLIT_CNF

단계 1. 원시 CNF의 각 컨정션 안에 있는 공간 술부들을 뒤로 모은다.

단계 2. 원시 CNF를 복사하여 새로운 전처리 단계의 CNF를 만든다.

단계 3. 전처리 단계의 CNF 내에 있는 모든 공간 연산자를 여과 연산자로 치환한다.

단계 4. 원시 CNF 중 공간 술부를 가진 컨정션 만을 가진 새로운 후처리 CNF를 생성한다.(혼합 컨정션 포함)

단계 5. 후처리 CNF 내의 모든 공간 연산자를 정제 연산자로 치환한다.

3.2 대수 노드의 분리 및 재배치

위의 CNF 분리 알고리즘을 수행하면 원시 CNF가 전처리 단계를 가진 CNF와 후처리 단계를 가진 CNF로 분리된다. 전자는 수행 비용이 적게 드는 서술자들로 구성된 CNF이므로 기존의 원시 CNF와 교체하여 푸쉬 다운 시켜서 여러 대수 노드들에 분리된다. 비용이 많이 드는 정제 연산자를 가지고 있는 후자는, 새로운 스캔 대수 노드를 생성하여 포함시키고 입력되는 임시 테이블의 크기가 최소가 되는 지점까지 끌어 올린다.

3.3 CSQR 기법의 적용 예

CSQR 기법이 실제 적용되는 예를 보이기 위해 다음과 같이 정의된 공간 뷰가 있다고 가정하자.

```
CREATE VIEW GANGNAM_VIEW AS
  ( SELECT DONG.* FROM DONG, GU
    WHERE GU.NAME = '강남구' AND
    DONG.OBJ INTERSECTS (GU.OBJ) );
```

사용자가 위의 공간 뷰에 다음과 같은 질의를 입력했다면,

```
SELECT  OBJ
FROM GANGNAM_VIEW
WHERE NAME = '논현동' OR
      OBJ CONTAINS(SEL_POINT());
```

이 질의는 질의 재작성 과정에서 다음과 같은 단일 블록 질의로 변경된다.

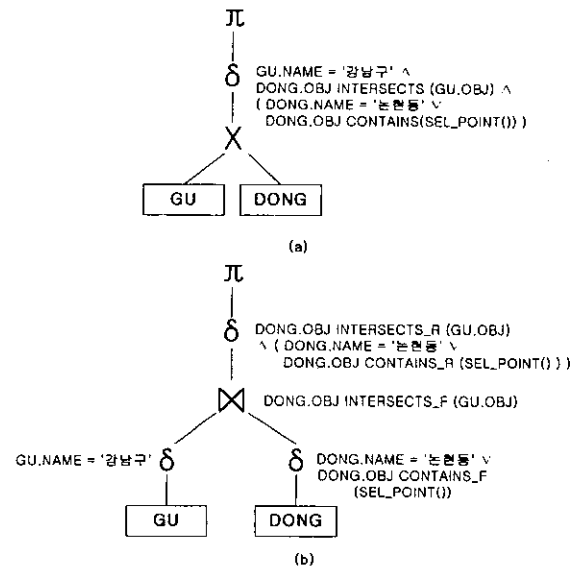
```
SELECT DONG.OBJ
FROM DONG, GU
WHERE GU.NAME = '강남구' AND
      DONG.OBJ INTERSECTS (GU.OBJ) AND
      ( DONG.NAME = '논현동' OR
        DONG.OBJ CONTAINS(SEL_POINT()) );
```

위의 재작성된 질의중 조건절의 CNF는 CNF 분리 알고리즘에 따라 다음과 같이 2개로 분리된다.

```
INTERSECTS_F, CONTAINS_F : 각 공간연산의 여과연산자
INTERSECTS_R, CONTAINS_R : 각 공간연산의 정제연산자

F1  : GU.NAME = '강남구'
      ∧ DONG.OBJ INTERSECTS_F (GU.OBJ)
      ∧ ( DONG.NAME = '논현동' ∨
            DONG.OBJ CONTAINS_F (SEL_POINT() ) )
F2  : DONG.OBJ INTERSECTS_R (GU.OBJ)
      ∧ ( DONG.NAME = '논현동' ∨
            DONG.OBJ CONTAINS_R (SEL_POINT()) )
```

분리된 2개의 CNF를 이용해 원시질의는 [그림 2-a]과 같이 질의 재작성 된 후, [그림 2-b]와 같이 최적화 된다.



[그림 2] CSQR 기법에 의한 실행트리의 재작성

4. 결 론

본 논문은 ESFAR에서 제안한 공간연산의 분리기법을 복잡한 CNF에 대해 적용할 수 있도록 확장한 CSQR 기법을 제시했다. CSQR 기법은 공간과 비공간 조건들을 가진 대수 노드들을 동등한 2개의 연산으로 분리하여 재작성하고 다시 재배치함으로써 비용이 많이 드는 정제단계에 입력되는 후보객체의 수를 최소로 만들어 공간 질의에 대한 최적화를 하는 기법이다. CSQR 기법은 공간연산이 포함된 질의를 질의 최적화 단계에서 비공간 연산과 다른 수행 전략을 세울 수 있다.

제안한 기법은 공간 및 비공간 조건들이 많이 섞여 있는 질의와 선택성(Selectivity)이 좋은 쵸인 서술자를 가지는 질의에 효과적으로 적용할 수 있다.

향후 연구과제는 CSQR에서 공간 비공간 혼합 컨정션의 분리시에 발생하는 비공간 서술자의 중복을 방지하는 방법에 관한 연구가 있다.

5. 참고 문헌

[1] Surajit Chaudhuri. An Overview of Query Optimization in Relatinal Systems. PODS, June, 1998. pp34-43

[2] Ho-Hyun Park, Chan-Gun Lee, Yong-Ju Lee, Chin-Wan Chung . Early Separation of Filter and Refinement Steps in Spatial Query Optimization. Proceedings of the 6th International Conference on Database Systems for Advanced Application.

[3] Walid G. Aref. Optimization Strategies for Spatial Query Processing. VLDB 17th. 1991. pp81-90

[4] Yannis Theodoridis, Emmanuel Stefanakis, and Timos Sellis. Efficient Cost Models for Spatial Queries Using R-Trees. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 12, NO. 1, JANUARY/FEBRUARY 2000 pp19-32

[5] Jack A. Orenstein. Spatial Query Processing in an Object-Oriented Database System. SIGMOD Conference 1986: pp326-336

[6] Joseph M. Hellerstein, Michael Stonebraker. Predicate Migration: Optimizing Queries with Expensive Predicates. . SIGMOD Conference 1993. pp267-276