분산 소스 환경에서 데이터 웨어하우스의 뷰 유지

이현창°
경인 여자 대학 컴퓨터 정보기술학부
hclee@dove.kyungin-c.ac.kr

View Maintenance of Data Warehouse in Distributed Sources Environment

Hyun-Chang Lee°
Dept. of Computer Information Technology, Kyungin Women College

요 약

일반적으로 효율적인 질의· 검색· 분석을 수행하기 위해서 이질적이고 분산된 정보 소스들부터 통합된 정보를 포함한 데이터 저장고를 데이터 웨어하우스라 일컬으며, 이를 웹기술과 접목한 기술을 웹 웨어하우징이라 한다. 본 연구에서는 웹 웨어하우징 기술의 모토가 될 수 있는 기술로서 분산되어 저장된 다양한 소스 데이터에 대해 실체 뷰로 간과되어지는 데이터 웨어하우스에 관한 유지 방법에 관한 연구이다.

본 연구의 성능 평가를 위해서 기존에 알려진 보상알고리즘, 모든 기본 릴레이션에서 키 애트리뷰트들을 포함해야만 하는 스트로브와 이를 완화시켜서 성능향상을 보이는 스윕 알고리즘들을 각각 특성별로 비교하며, 전송된 바이트 수와 갱신된 회수에 따른 성능 평가 및 갱신 유형에 따른 성능 평가를 수행하여 결과를 보인다.

1. 서론

웹 웨어하우징은 주로 데이터 웨어하우스로 부터 정보를 알리기 위한 목적으로 웹기술을 사용하는 데이터 웨어하우징이다[2]. 일반적으로 웹 웨어하우징은 웹 브라우저를 사용하여 접근할 수 있어야하며, 인터넷이나 인트라넷을 통해 연결되며 데이터베이스로부터 동적으로 내용을 생성시킬 수 있는 기능을 제공해야한다.

이와같은 웹 웨어하우징을 구성하고 있는 데이터 웨어하우스는 사용자의 의사 결정에 필요한 정보를 제공하여 효율적인 데이터 마이닝 질의 처리 및 그에 대한 응답을 이루도록 해준다. 이를 위해서 데이터 웨어하우스는 소스 데이터로부터 유도된 실체 뷰를 저장하고 있다.

실체 뷰에 대한 대부분의 연구는 뷰 생성에 사용된 소스 테이블에 변경이 일어날 때 실체 뷰를 점진적으로 변경하는 기법들이다[5]. 실체 뷰에 관한 기존의 연구 방법들에서는 주로 각 소스에서 뷰에 관한 관리 방법을 알고 있으며, 질의에 따른 뷰와의 관련성을 알고 있다는 가정에서 출발한다. 그러나 웹 환경 및 데이터 웨어하우스 환경에서 소스는 뷰에 관한 정보를 알지못한다[1].

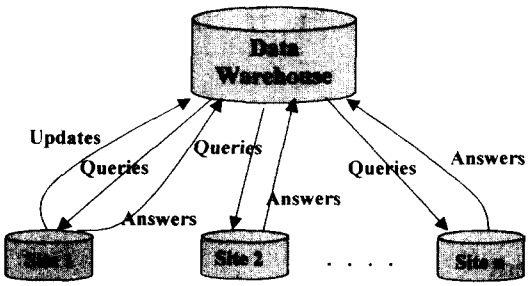
이로 인하여 소스의 변경이 실체 뷰에 바로 적용될 수 없으며, 이것은 단일 소스 및 분산된 다중 소스 환경에서 데이터 웨어하우스의 부정확한 실체뷰의 원인이 된다. 그러므로 뷰와 소스를 일관성있는 뷰로 유지할 필요가 있다[7].

이에 본 논문에서는 웨어하우스에 저장된 데이터를 이상 현상이 발생하지 않도록 정확한 실체 뷰를 유지할 수 있는 알고리즘을 제시하며, 기존의 알고리즘들이 가지고있는 문제점을 감소시킬 수 있는 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 기존에 이루어져왔던 연구들에 대해 살펴보며, 제 3장에서는 본 논문에서 제시하고있는 실체 뷰 유지 알고리즘에 대해 설명한다. 제 4장에서는 본 논문에서 제시하고 있는 알고리즘의 성능 평가를 다른 알고리즘들과 비교하였으며, 제 5장에서 결론 및 향후 연구 방향으로 본 논문을 구성하였다.

2. 관련연구

다음 <그림 1>은 본 논문에서 제시하는 갱신 처리 모델인 분산된 소스 데이터 웨어하우스 환경에서 이루어지는 모델을 그림으로 도시하였다.



<그림 1> 웹 웨어하우징의 구조

<그림 1>은 발생된 하난의 갱신 정보가 다중 데이터 소스 웨어하우스에서 어떻게 처리되는지를 보여주고 있다. 즉, 데이터 소스에서 발생된 갱신 정보가 뷰에 반영되기 위한 처리 과정을 보이고 있다. 상기 과정을 수행하는 과정중에 다중 소스 데이터 웨어하우스에서 이상 현상이 발생할 수 있으며, 이를 방지하면서 원할하게 수행할 수 있도록 제안된 알고리즘으로서 먼저, 보상 알고리즘을 들 수 있다.

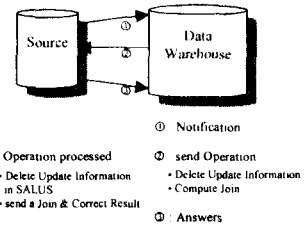
보상알고리즘의 문제점은 단일 소스 환경이라는 제약사항 이외에도 뷰를 유지하기 위해서 재계산에 드는 오버헤드 비용이 크다는데 있다[1]. 다른 알고리즘으로 Strobe 알고리즘을 들 수 있다[10]. 이 알고리즘은 기본적으로 키(key)를 포함하여 뷰를 유지하는 방법이다. 키를 사용함으로써 얻을 수 있는 장점은 질의 응답 속도 및 통신 비용을 줄일 수 있다는 사실이다. 그러나 보상 알고리즘과 마찬가지로 소스에서 발생한 모든 갱신 정보에 대한 응답 릴레이션을 받을 때까지 실체 뷰를 정확하게 유지할 수 없는 조용한 상태(quiescent state)가 발생한다는 사실이다. 또한 모든 갱신 정보의 결과 릴레이션을 받을 때까지 기다려야 하므로 정확성 측면에서도 떨어진다.

이와 같은 단점을 극복한 알고리즘으로 스윕(Sweep) 알고리즘을 들 수 있다[11]. 이 알고리즘은 스트로브 알고리즘의 단점인 킷값을 포함하지 않고서 실체 뷰를 유지하는 방법이다. 그러나 질의에 대한 정확한 결과를 얻기 위해서 질의 관리를 데이터 웨어하우스에서 수행해야 하며, 이는 곧 웨어하우스의 관리 오버헤드를 초래한다.

3. 분산 소스 환경에서 뷰 유지 알고리즘

본 장에서는 다중 소스 데이터 웨어하우스 실체 뷰 유지를 위한 알고리즘에 관하여 살펴본다. 먼저, 다중 소스 웨어하우스 환경에서 소스 데이터에서 발생된 갱신 정보에 대해 처리되는 과정을 <그림 2>에 도시하였다.

처리 과정은 다음과 같다. 먼저, 소스는 발생된 갱신을 SALUS에 등록한 후 웨어하우스로 갱신정보를 통보한다. 웨어하우스에서는 소스로부터 받은 갱신 정보에 대해 질의를 생성하여 소스로 보낸다. 이때 보내는 메시지의 형태는 두 가지로서 하나는 소스의 SALUS에 등록된 갱



<그림 2> 다중 소스 데이터 웨어하우스에서 처리과정

신 정보를 삭제하라는 메시지와 다른 하나는 조인을 수행하기 위한 질의 형태 메시지이다.

소스는 웨어하우스로부터 받은 메시지 중에서 갱신 정보 삭제 메시지일 경우 SALUS에서 해당 갱신 정보 삭제를 수행한다. 그렇지 않으면 조인을 수행하라는 메시지에 대해 조인을 수행한다. 조인을 수행한 결과 릴레이션은 본래 의도한 결과 릴레이션이 아닐 수 있다. 이에 소스에서는 결과 릴레이션을 정확하고 의도된 결과 릴레이션 값을 얻기 위해서 교정 작업을 수행한다. 즉, 수행하고 있는 연산 이후에 발생된 갱신들에 대해 보상 작업을 수행하는 것이다. 상기 과정을 마치게되면 본래 의도하였던 정확한 결과 응답 릴레이션을 얻을 수 있게 된다. 다음은 본 논문에서 제시하는 알고리즘을 사건의 종류에 따라서 이해를 돕고자 한다.

3.1 사건

사건은 같은 사이트에서 수행되는 일련의 연산에 대응된다고 말할 수 있다. 하나의 사건 내에서는 상기 기술된 순서대로 수행이 이루어지며, 서로 다른 사건들 사이의 순서는 무관하게 수행될 수 있다고 가정한다. 다음은 소스와 웨어하우스에서 일어날 수 있는 사건들이다.

1) 소스에서의 사건

· S_req_i : 발생한 갱신 정보 U_i가 뷰 정보와 관련성이 존재하는지 확인하기 위해 웨어하우스로 정보 송신한다.

· S_del_i : SALUS에서 갱신 정보를 삭제한다.

· S_eva_i : 웨어하우스에서 전송된 질의를 평가하여 임시 결과 릴레이션을 생성한다. 생성된 임시 릴레이션에 대해 교정 작업을 수행하여 의도한대로 정확한 결과 릴레이션 A를 웨어하우스로 전송한다.

2) 웨어하우스에서의 사건

· W_view_i : 소스로부터 갱신 정보(S_req_i)를 받아서 SALUSList에 등록후 질의 Q_i^1, Q_i^2를 생성한다. 갱신 정보를 보내온 사이트로 갱신 정보 삭제 메시지를 보낸다. 그 이외의 다른 두 방향(소스S_{i-1}, $i+1$)으로 질의 평가를 위해서 소스로 두 질의를 전송한다.

· W_ans_i : 소스에서 보내온 응답 테이블 A_i^1, A_i^2를 받아서 두 결과 릴레이션을 조인한다. 조인된 결과를 실체 뷰에 반영한다.

3.2 알고리즘

본 절에서는 데이터 소스들과 웨어하우스 사이에서 전송된 메시지 크기와 서버 로드를 줄이며, 처리 응답 시간

을 감소시킬 수 있는 알고리즘에 관하여 간략히 살펴본다. 먼저, 소스에서 발생된 갱신 처리를 위한 메인은 다음과 같다.

```
MODULE UpdateQueryatSource;
  GLOBAL DATA
    SALUSList, : LIST Initialized to ∅;
    Temporary : RELATION /* Initially ∅*/
  CONSTANT
    Index = i;
BEGIN /* Initialization */
    StartProcess(SendUpdates);
    StartProcess(ProcessQuery);
END UpdateandQueryatSource
```

다음은 각 소스에서 발생되어 서버로 보내온 질의를 위한 서버측의 처리 알고리즘을 나타내었다.

```
MODULE DataWarehouse;
  GLOBAL DATA
    MV : RELATION; /*Initialized to the correct view*/
    SALUSList: List Initially ∅;
  BEGIN /* Start DataWarehouse Processes */
    StartProcess(UpdateSALUS);
    StartProcess(UpdateView);
END DataWarehouse.
```

상기 과정에서 언급 하였듯이 소스와 데이터 웨어하우스에서 처리되는 단계를 간략히 기술하였다. 위 과정을 통해서 소스는 웨어하우스에 정확한 결과를 반환할 수 있게 되며, 본래 데이터 웨어하우스가 수행하던 질의 관리 및 결과 릴레이션에 대한 보상 역할을 데이터 소스로 이전시킴으로서 서버의 로드를 줄일 수 있게 된다.

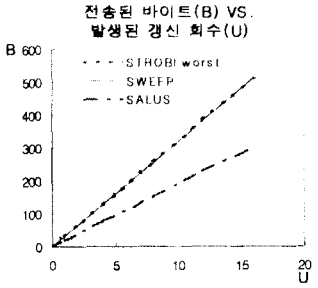
4. 성능 평가

이번 장에서는 본 논문에서 제시하고 있는 뷰 유지 알고리즘에 대한 성능 평가를 위해서 스트로브 알고리즘과 스트로브의 제약사항을 많이 완화시켜서 좋은 성능을 보이는 스윕 알고리즘들과 함께 성능 비교를 하고자한다. 다음 <표 1>에서는 본 논문에서 제시하는 알고리즘과 기존의 알고리즘들과의 특징 및 간략한 비교를 표로 도시하였다.

<표 1> 뷰 유지 알고리즘의 비교

ECA	중앙 집중식 환경	O(1)	원격 보상처리 기하급수적인 메시지 크기 조용한 상태 발생
Strobe	분 산 환 경	O(n)	키 유지 요구 조용한 상태 발생
SWEEP	분 산 환 경	O(n)	DW에서 지역적으로 보상 서버의 로드 증가 긴 응답 시간
SALUS	분 산 환 경	O(n/2)	각 소스에서 지역적으로 보상 빠른 응답 시간

다음 <그림 3>은 갱신회수와 이에따른 전송된 바이트와의 성능 관계를 도식화하였다.



<그림 3> 갱신된 회수와 전송된 바이트와의 관계

5. 결론

웨어하우스는 소스로부터 갱신 정보를 통보받게 되면 뷰를 갱신해야할 필요가 있는지를 파악하기 위해서 소스들에 질의를 전송한다. 이때 소스에서 또다른 갱신이 발생하게 되면 소스의 기본 데이터와 웨어하우스의 실체 뷰 사이에서 이상 현상이 발생하게 되어 부정확한 뷰를 초래할 수 있다.

본 논문은 이 문제점을 모두 해결할 수 있는 알고리즘을 제시하였으며, 웨어하우스가 뷰를 갱신하는 동안 일관된 상태를 유지하면서 성능 향상을 보였다.

6. 참고 문헌

[1]. Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom. "View Maintenance in a Warehousing Environment," *Proc. of the ACM SIGMOD Conference*, pages 316-327, San Jose, California, May 1995.

[2]. R.D.Hackathorn, "Web Farming for the Data Warehouse", 1999.

[3]. J. A. Blakeley, P. Larson, and F. W. Tompa. "Efficiently Updating Materialized Vews," *Proceedings of ACM SIGMOD 1986 International Conference*, pages 61-71, Washington, D.C., May 1986.

[4]. A. Gupta and I. S. Mumick. "Maintenance of Materalized Views: Problems, Techniques, and Applications," *IEEE Data Engineering Bulletin*, Special Issue on Materialized Views and Data Warehousing, 18(2):3-19, June 1995.

[5]. Y.Zhuge,H.Garcia-Molina, and Janet L.Wiener. The Stroe Algorithms for Multi-Source Warehouse Consistency. In Proceedings of the International Conference on Parallel and Distributed Information Systems, December 1996.

[11].D.Agrawal,A. Abbadi,A.Singh,T.Yurek. Efficient View Maintenance at Data Warehouses. ACM SIGMOD pages 417-427, 1997.