데이터 웨어하우스에서 참조 무결성 제약 조건을 이용한 병렬 뷰 일관성 관리 기법

이병숙[1)0], 김진호[1)], 옥수호[2)], 이우기[3)]
[1)] 강원대학교 전자계산학과
[2)] 고신대학교 컴퓨터과학부
[3)]성결대학교 컴퓨터 공학과
bsdream@mail.kangwon.ac.kr, jhkim@cc.kangwon.ac.kr, shok@kosin.ac.kr, wook@hana.sungkyul.ac.kr

Parallel View Consistency Maintenance Using Referential Integrity Constraints in Data Warehouse Environment

Byoung-Suk Lee[1)0], Jin-Ho Kim[1)], Soo-Ho Ok[2)],Woo-Key Lee[3)]
[1)]Dept. of Computer Science, Kangwon National University
[2)]Dept. of Computer Science, Kosin University
[3)]Dept. of Computer Engineering, Sungkyul University

요 약

데이터 웨어하우스는 물리적으로 여러 사이트에 위치한 분산된 데이터 소스로부터 추출한 온라인 분석 정보를 유지하는 실체 뷰의 집합으로 구성된다. 따라서 데이터 소스에 변경 사항이 발생하면 데이터 웨어하우스와 일관성을 유지하기 위해 뷰에도 그 변경 사항을 반영하는 뷰 관리가 필요하다. 동시에 변경되는 여러 데이터 소스와 뷰의 상태 사이에 일관성을 보장하기 위해서는 각 소스의 변경 사항을 순서대로 뷰에 반영해야 한다. 이때 각 소스의 변경 사항을 뷰 정의와 관련된 다른 소스들과 조인을 수행해야 하는 등 뷰 갱신을 위해 많은 비용이 소요된다. 이러한 뷰 갱신 비용을 줄이는 방법중의 하나로 병렬처리 기법을 활용하는 연구가 시도되고 있다. 따라서 이 논문에서는 뷰의 일관성을 보장하기 위해 수행해야 하는 서브질의를 병렬로 처리하는 알고리즘을 제시하였다. 이 방법에서는 서브질의의 조인 연산들을 소스 릴레이션들 간의 참조 무결성 제약 조건을 이용하여 병렬로 처리한다. 질의의 조인 처리를 병렬화 하기 위해 소스 릴레이션간의 참조 무결성 제약조건의 특성을 이용하여, 여러 릴레이션을 참조하는 릴레이션에서 발생하는 변경 사항에 대해 참조하는 릴레이션의 수만큼 병렬로 조인 연산을 수행하는 알고리즘을 제시하였다. 이렇게 함으로써 여러 소스 릴레이션의 조인으로 구성된 실체 뷰를 갱신하는 시간을 크게 단축하여 효율적으로 뷰를 관리하도록 하였으며, 소스의 증가에 따른 뷰 갱신 시간의 증가를 줄일 수 있도록 하였다.

1. 서 론[1)]

데이터 웨어하우스(Data Warehouse : DW)는 효과적인 질의와 분석을 위해 통합된 정보를 소스 데이터로부터 추출하여 저장한 데이터 저장소로써, 사용자의 요구에 적합하도록 만들어진 소스 데이터에 대한 실체뷰(Materialized View)로 간주된다. 그러므로 사용자들은 분석과 질의에 DW의 실체뷰를 이용하여 보다 빠른 분석결과를 얻을 수 있다[1]. DW는 소스 데이터의 갱신에 따라 실체뷰를 최신 정보로 유지해야 하는데 이를 실체뷰 관리라고 한다. 소스 데이타의 변경사항만을 이용하여 DW의 실체뷰를 효과적으로 관리하는 점진적 뷰 관리 기법이 일반적으로 많이 사용된다[1][2]. 이러한 뷰 관리 기법을 이용하여 소스 데이타에서 발생한 변경사항을 실체 뷰에 반영할 때, 사용자 질의들이 올바른 (일관성 있는) 결과를 제공할 수 있도록 소스 데이터의 변경 순서 및 상태와 뷰의 상태가 일치하도록 서로의 일관성을 보장해야 한다[3][4][5].

DW에 존재하는 데이터 소스들은 지역적으로 떨어져 있으며 서로 독립적으로 수정될 수 있기 때문에, 동시에 여러 소스에 변경사항이 발생하는 상황에서 DW의 뷰를 일관되게 관리할 수 있는 점진적 뷰 관리 기법이 필요하다. 소스의 여러 변경 사항에 대해 뷰의 일관성을 보장하기 위한 방법으로 ECA, Strobe, SWEEP, PVM등의 알고리즘들이 개발되었다.

ECA 알고리즘은 단일 사이트의 소스 환경에서, Strobe와 SWEEP은 분산 소스 환경에서 일관성을 만족하는 뷰 관리 기법들을 제시하였다. SWEEP은 순차적으로 변경사항들을 뷰에 반영하므로 소스의 수가 증가함에 따라 뷰 갱신비용이 비례적으로 증가하게 된다. 따라서 대규모의 DW에 이용되기에 적절하지 못하다. 이러한 뷰 갱신 비용을 줄이기 위해 변경사항들을 병렬로 처리하는 방법이 [4]에서 제시되었다. [4]에서 제시된 병렬처리 알고리즘 PVM은 SWEEP과 동일한 방법으로 변경 사항을 뷰에 반영하는 쓰레드를 여러 개 만들어 병렬로 처리하는 방법이다. 여러 변경사항들을 병렬로 처리함으로써 처리비용을 줄일 수 있지만, 하나의 변경사항에 대한 뷰 갱신값을 얻기 위한 조인들은 여전히 SWEEP과 동일하게 순차적으로 처리하고 있다. 따라서 본 논문은 뷰 갱신 값을 얻기 위해 수행하는 조인 연산들을 병렬로 처리하는 기법인 PSWEEP/RI (Parallel SWEEP with Referential Integrity)을 제시한다. PSWEEP/RI은 변경 소스와 다른 소스들간의 조인을 병렬화하기 위해 조인관계에 있는 소스 릴레이션들간의 참조 무결성을 이용한다. 조인이 릴레이션간의 참조 무결성 제약조건에 참여하는 기본 키와 외래 키 사이에 발생할 경우, 참조되는 릴레이션의 변경사항은 이를 참조하는 튜플이 아직 없으므로 이를 참조하는 릴레이션과 조인할 필요가 없음을 미리 알 수 있으며[1], 참조하는 릴레이션의 변경사항은 이 릴레이션이 참조하는 여러 릴레이션들과 조인을 동시에 해도 동일한 조인 결과를 얻을 수 있다. 따라서 본 논문에서 제시하는 PSWEEP/RI은 소스 릴레이션들간의 참조 무결성관계를 설정한 후, 다른 릴레이션을 참조하는 릴레이션에서 변경사항이 발생할 경우 뷰갱신 값을 계산하는 조인 처리시 참조하는 릴레이션의 개수만큼 병렬로 조인을 수행한다. 또한 참조되는 릴레이션의 변경사항은 뷰에 영향을 주지 않으므로 이 경우는 조인 계산을 하지 않고 필터링한다. 이렇게 함으로써 뷰의 일관성을 관리하기 위해 SWEEP에서 요구되는 갱신비용을 크게 감소시키며, 소스가 증가하더라도 조인계산이 병렬처리 되므로 갱신비용이 크게 증가하지 않도록 효율적으로 관리할 수 있도록 하였다.

2. 관련연구

실체뷰 관리에 있어 중요한 사항 중의 하나는 데이터 소스와의 일관성(consistency)을 유지하는 문제이다[1][3][5]. 뷰 관리시 데이터 웨어하우스는 소스 데이터에서 발생한 변경사항을 관련된 다른 소스 사이트의

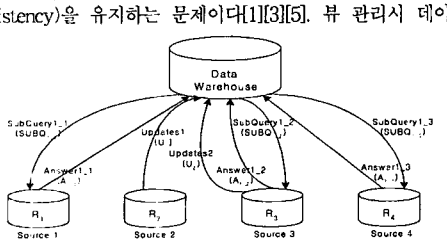


그림 1 분산 소스 환경에서의 뷰 갱신 과정

1) 이 논문은 첨단정보기술연구센터(AITrc)를 통하여 한국과학재단의 지원을 받았음.

정보와 통합하여 실체 뷰에 반영하게 되는데, 이때 다른 사이트들도 동시에 변경될 수 있기 때문에 각 소스의 변경 사항과 일관된 정보들을 다른 사이트로부터 추출하여 뷰를 일관성있는 상태로 수정/유지하는 것은 상당히 어렵다.

예를 들어, 소스 릴레이션 R_1, R_2, R_3, R_4의 조인으로 구성된 데이터 웨어하우스 뷰가 있다고 하자. 소스 릴레이션 R_2에서 발생한 변경사항 $\triangle R_2(U_1)$가 DW에 도달하면 DW는 뷰의 갱신사항($\triangle V$)를 계산하기 위해 아래와 같은 질의, Query1을 생성하고 이 Query1을 계산하기 위한 서브질의(SubQuery)를 각 소스에 순차적으로 내려보내, 해당 소스와 내부 조인을 함으로써 Query 1의 응답을 얻는다[1]. (여기서 Answer 1_1은 SubQuery 1_1의 응답이다.)

$$Query\ 1 = R_1 \bowtie \triangle R_2 \bowtie R_3 \bowtie R_4$$
$$SubQuery\ 1\_1 = R_1 \bowtie \triangle R_2$$
$$SubQuery\ 1\_2 = Answer\ 1\_1 \bowtie R_3$$
$$\cdots \qquad \cdots \qquad \cdots$$

위의 경우와 같이 하나의 변경사항에 대한 뷰 갱신을 완료한 후 소스 데이타의 다음 변경사항이 발생한다면, 일반적인 점진적 뷰 관리기법으로 DW의 뷰를 효과적으로 관리 할 수 있다[4][5].

그러나 DW에 존재하는 데이터 소스들은 지역적으로 떨어져 있으며 서로 독립적으로 수정될 수 있기 때문에, 실제로 위와 같이 충분한 시간 간격을 두고 각각의 변경사항들이 발생하지는 않는다. 즉 위의 그림 1에서처럼 Answer 1_2가 DW에 도달하기 전에 R_3의 변경사항인 $\triangle R_3(U_2)$가 발생하면 Query 1의 결과는 U_2가 발생한 후의 상태를 반영함으로써 소스의 상태와 DW 상태의 일관성을 잃게 된다.

그러므로 위와 같은 동시 변경사항(Concurrent Update)이 발생하는 상황에서 DW의 뷰를 일관되게 관리할 수 있는 점진적 뷰 관리 기법이 필요하다. 이러한 문제점을 해결하기 위한 접근법으로 ECA, Strobe, SWEEP, PVM등의 알고리즘들이 개발되었다. ECA 알고리즘은 단일 사이트의 소스 환경에서, Strobe와 SWEEP 알고리즘들은 분산 소스 환경에서 일관성을 만족하는 뷰 관리 기법들을 제시하였다. SWEEP 알고리즘은 소스에서 발생하는 모든 변경사항들을 데이터 웨어하우스에 도착한 순서대로 보관, 처리, 보상하므로 완전 일관성(complete consistency)를 만족한다. (완전 일관성이란 소스의 모든 변경 사항 각각에 대해 일관성을 유지하도록 뷰를 갱신하는 것을 말한다.) 따라서 DW에서 수행되는 모든 query들은 분산 데이터베이스의 일관된 뷰를 보상하고, 동시 변경사항에 따른 보상(compensation) 연산을 DW내에서만 수행된다.

SWEEP에서는 발생한 모든 변경사항들을 DW에 도착한 순서대로 UMQ(Update Message Queue)에 저장하여 순차적으로 처리된다. 또한 각 변경사항에 대한 뷰 갱신 값을 계산하기 위해 다른 소스 사이트와 조인하는 서브질의들을 순차적으로 처리한다. 따라서 소스의 수가 많거나 변경사항들이 증가할 경우 많은 처리 비용이 소요된다는 단점이 있다. 이러한 단점을 보완하기 위해 변경사항들을 병렬 처리하는 방법이 [4]에서 제시되었다. [4]에서 제시된 병렬처리 알고리즘인 PVM은 뷰 갱신 쓰레드를 여러개 두어 SWEEP과 동일한 방법으로 여러 개의 변경사항들을 동시에 처리하는 개념이다. PVM은 변경사항이 병렬처리되므로, 병렬 프로세스의 수만큼 변경사항에 뷰갱신시간은 단축된다. 그러나 이 방법에서도 각 변경사항을 반영하기 위해 필요한 여러 개의 조인 연산등의 서브질의들을 SWEEP에서와 같이 순차적으로 수행하고 있어 이에 대한 수행 시간은 여전히 동일하게 소요된다. 이 논문에서는 이들 서브질의들을 병렬로 처리하는 방법을 사용하여 각 변경사항에 대한 뷰 갱신을 효율적으로 수행할 수 있는 방법을 제시한다. 이렇게 함으로써 뷰 갱신 비용을 줄일 수 있을 뿐만 아니라 참여하는 소스 수의 증가에 따른 뷰 갱신 비용의 증가 문제를 줄일 수 있도록 하였다.

3. PSWEEP/RI 알고리즘의 기본 개념

PSWEEP/RI는 서브질의의 소스와의 조인 연산을 병렬화하기 위해 조인관계에 있는 소스 릴레이션들간의 참조 무결성 제약 조건을 이용한다. 조인되는 릴레이션들간에 참조 무결성 제약 조건을 만족하는 경우, 참조 릴레이션의 변경사항은 여러 피참조 릴레이션들과의 조인을 병렬로 처리할 수 있다. 또한 피참조의 삽입(또는 삭제)의 변경사항은 참조 무결성 제약 조건 때문에 이를 참조하는 투플들이 다른 참조 릴레이션에 존재할 수 없으므로 이들과의 조인을 아예 수행할 필요가 없다. 따라서 본 논문에서 제시하는 PSWEEP/RI는 소스 데이터인 릴레이션들간의 참조 무결성 제약 조건(Referential Integrity Constraints : RI)을 설정한후, 변경사항이 발생하는 릴레이션의 참조관계에 따라 생성된 서브질의의 조인 연산들을 병렬로 처리하거나 필터링시키는 방법을 제시한다.

3.1 기본 접근 방법

PSWEEP/RI은 소스와의 조인을 병렬화하기 위해 조인관계에 있는 소스 릴레이션들의 참조 무결성을 이용하므로, 각 소스의 기본 릴레이션간에 그림 3과 같이 참조 무결성 제약 조건을 설정하여 참조 무결성 그래프를 생성한다. 그림 2에서처럼 릴레이션 수강은 릴레이션 학생, 릴레이션 강좌, 릴레이션 교수와 RI관계에 있다. 수강의 경우처럼 외래키를 가지고 있어 학생 릴레이션을 참조하는 경우를 RI 참조 관계라하고(여러 릴레이션을 참조할 경우 다중 참조라 한다), 교수나 학생같이 참조되는 경우를 피참조관계(여러 릴레이션에 참조 될 경우 다중 피참조라 한다)라 한다. 만약 수강에서 변경사항($\triangle R_2$)이 발생하면 이를 처리하기 위해 변경사항 메시지를 전달받은 데이터 웨어하우스에서는 다음과 같은 질의를 생성한다.

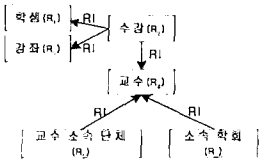


그림 2 참조 무결성 그래프

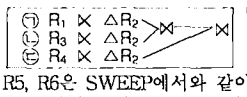
$$\triangle V = R_1 \bowtie \triangle R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5 \bowtie R_6$$

SWEEP에서는 $\triangle R_2$를 기준으로 순차적으로 하나의 변경사항에 대한 질의의 적당한 부분을 소스로 내려보내, 변경사항과 소스를 조인 계산 처리하여 서브질의의 결과를 얻는다. 따라서 기본 릴레이션이 증가되면 조인횟수가 증가하므로 조인 처리에 많은 처리비용과 시간이 요구될 것이다. 이 조인 연산들을 병렬로 수행할 경우 그 처리 시간을 현저히 줄일 수 있을 것이다. 이들 릴레이션들 간의 RI 관계를 이용하면 병렬로 처리될 수 있는 조인 연산을 찾아낼 수 있다.

다음에서 RI관계를 이용한 서브질의의 병렬 처리 방법을 제시한다. 변경사항이 발생한 릴레이션은 수강(R_2)이고, 따라서 $\triangle R_2$는 수강에서 발생한 변경사항을 말한다. 또한 변경사항은 삽입과 삭제만으로 제한되고, 갱신(update)는 고려하지 않는다. 지면 관계상 예제는 생략한다.

3.2 다중 참조 릴레이션의 변경시 처리 방법

변경사항이 DW로 전달되면 데이터 웨어하우스는 뷰 갱신값을 계산하기 위해 질의를 생성한다. 질의의 병렬처리 수행 여부를 결정하기위해 먼저 앞에서 제시한 RI 그래프를 탐색한다. 우리의 예제인 변경사항 $\triangle R_2$를 위해 RI 그래프를 검색하면 R_1, R_2, R_3와 참조관계임을 알 수 있다. 따라서 질의 처리시 R_1, R_2, R_3와 $\triangle R_2$와의 조인을 동시에 병렬 처리 할 수 있다. 만약 질의 처리시 조인 계산으로 서브질의를 수행하면, 애트리뷰트의 중복되므로 세미 조인을 이용한다. RI 그래프를 탐색해서 다음과 같은 병렬 처리식을 얻을 수 있다.



㉠,㉡,㉢을 병렬로 세미 조인 계산 처리하고 난 후 그 결과를 다시 조인하게 된다. 그리고 RI 관계가 아닌 나머지 릴레이션 R5, R6은 SWEEP에서와 같이 순차적으로 조인 처리한다.

본 방법은 참조 관계가 많을수록 질의 처리시 소스들과의 병렬조인 처리가 증가한다. 따라서 SWEEP에 비해 뷰갱신 시간이 단축되고, 또한 이때 세미조인을 이용하므로 일반 조인을 사용하는 경우에 비해 더욱 처리시간이 단축된다. 따라서 소스의 증가하여도 뷰 갱신시간은 크게 증가되지 않는다.

3.3 다중 피참조 릴레이션의 변경시 처리 방법

변경사항($\triangle R$)이 발생하는 릴레이션이 다중 피참조 관계인 경우는 삽입이나 삭제시 뷰 테이블에 영향을 주지 않는다. 왜냐하면 RI 관계에서 참조되는 테이블의 경우는 기본키를 가지고 있는 테이블이다. 따라서 해당 투플을 참조하는 다른 릴레이션이 있는 경우는 투플을 제거할 수 없다. 또한 삽입의 경우도 마찬가지로 참조되는 테이블이므로 삽입된 변경사항이 뷰에 영향을 주지 않는다. 그러므로 피참조 릴레이션의 변경사항은 필터링 된다. 본 논문에서 제안하는 이 방법은 뷰 갱신의 불필요한 질의 처리를 줄일 수 있으므로 SWEEP의 경우보다 매우 효율적이다.

4. PSWEEP/RI 아키텍춰 및 알고리즘

본 절에서는 이 알고리즘을 사용하는 DW 시스템의 전체 아키텍춰와

알고리즘에 대해 설명한다.

4.1 PSWEEP/RI 아키텍쳐

그림 3에서 보여주는 PSWEEP/RI의 기본 아키텍처는 SWEEP과 유사하다. 본 아키텍처는 소스 모듈과 DW 모듈로 이루워져 있고 두 모듈은 FIFO 네트워크으로 연결된다.

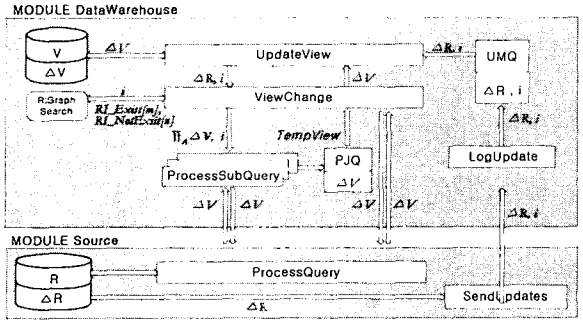


그림 3 PSWEEP/RI 구조도

먼저 소스 모듈은 ProcessQuery와 Sendupdates 쓰레드로 구성된다. ProcessQuery 쓰레드는 DW에서 생성된 점진적 질의에 응답하는 서비스를 제공한다. SendUpdates 쓰레드는 기본 릴레이션에서 발생하는 변경사항(ΔR)를 DW로 보내는 서비스를 제공한다. 소스 모듈 부분은 SWEEP과 동일하다.

PSWEEP/RI의 DW 모듈은 LogUpdate, UpdateView, ViewChange, ProcessSubQuery등의 쓰레드로 이루워져 있다. LogUpdate와 UpdateView는 SWEEP과 동일한 기능을 한다. ViewChange 함수는 실제적인 질의를 수행하는 프로세스다. 먼저 RI관계를 찾기 위해 UpdateView로부터 받은 기본 릴레이션의 인덱스를 이용하여 RIGraphSearch 함수를 호출한다. 함수로부터 넘겨받은 RI관계에 있는 릴레이션의 수 만큼 ProcessSubQuery를 병렬 수행한다. 각각의 ProcessSubQuery의 수행 결과인 질의에 대한 응답(△V)은 PJQ에 추가된다. PJQ는 ParallelJoinQueue로 병렬 수행되는 응답(△V)값을 저장하는 저장소이다. ViewChange는 PJQ내의 뷰 변경사항에 대해 차례대로 join 명령을 수행한다. 또한 RI 관계가 없는 기본 릴레이션의 위해 순차적으로 질의를 수행한다.

ProcessSubQuery는 RI관계가 있는 기본 릴레이션에 병렬로 질의를 수행하여 응답을 받는다. 질의 수행은 기본 릴레이션과 변경사항과의 조인(또는 semi join)으로 계산된다. ProcessSubQuery는 계산결과를 PJQ에 추가한다.

4.2. PSWEEP/RI Algorithm

이 절에서는 PSWEEP/RI 알고리즘의 핵심 코드를 보여준다. 아래 그림 4는 PSWEEP/RI의 알고리즘을 보여주고 있는데, 지면 관계상 SWEEP과 동일한 부분은 생략한다.

```
MODULE DataWarehouse ;
  GLOBAL DATA
  V: RELATION ;  /*Initialized to the correct view */
  UpdatedMessageQueue : QUEUE initially 0;  ParallelJoinQueue : QUEUE initially 0;
  max : integer  /* 전체 relation의 개수 */

FUNCTION ViewChange (ΔR: RELATION ; UpdateSource : INTEGER): RELATION
    VAR
      ΔV , TempView : RELATION ;   k : INTEGER ;
    BEGIN
      ΔV = ΔR ;
      RIGraphSearch(i);
/* RI 관계가 존재하는 Relation에 대한 query process의 병렬화 */
      FOR ( k = 0 ; k < m ; K++ ) DO
        StartProcess ( ProcessSubQuery( ΠAΔV,RI_Exist[k]) ) ;
      ENDFOR ;
/* 병렬처리된 결과의 join 처리*/
      FOR ( k = 0 ; k < m ; K++ ) DO
        REMOVE (Tempview) FROM ParallelJoinQueue ;
        ΔV = ΔV ⋈ Tempview ;
      ENDFOR ;
/* 피참조되는 relation들과의 join 처리 */
      FOR ( k = 0 ; k < n ; K++ ) DO
        SEND ΔV TO DataSource RI_NoExist[k];
        RECEIVE ΔV FROM DataSource RI_NoExist[k];
      ENDFOR ;
  RETURN (ΔV) ;
ENDViewChange;
```

그림 4 데이터 웨어하우스 모듈 알고리즘

```
FUNCTION RIGraphSearch (RI_Exist[]:array, m:integer, i:integer, RI_NoExist[]:array, n:0integer )
    VAR
      k : INTEGER ;
    BEGIN
      m=0 , n=0 ,
      FOR (k=1; k<max ; k++ ; )
        IF k와 i가 RI관계 THEN /* Rk가 Ri를 참조하는가를 조사 , i : k */
          RI_Exist[m]=k; /* RI_Exist[] = RI(참조)관계가 있는 relation의 index 배열 */
          m++;
        ELSE
      RI_NoExist[n]=k+1; /* RI_NoExist[] = RI(참조)관계가 없는 relation의 index 배열 */
          n++;
          return(RI_Exist[m],RI_NoExist[n]);
        ENDIF
      FOREVER ;
ENDRIGraphSearch;
PROCESS ProcessSubQuery ( ΔV : RELATION ; j: integer ;)
    VAR
      ΔV, TempView : RELATION ;   k : integer ;
    BEGIN
      TempView = ΔV ;
      SEND ΔV TO DataSource j ;
      RECEIVE ΔV FROM DataSource j ;
      APPEND (ΔV) TO ParallelJoinQueue ;
END ParallelProcessQuery ;

PROCESS LogUpdates ;   BEGIN  .....  END LogUpdates ;

PROCESS UpdatesView ;
    BEGIN
      LOOP
  REMOVE (ΔR, i) FROM UpdateMessageQueue ; /* block if queue empty */

        V = V + ViewChange( ΔR, i)
      FOREVER
END UpdateView ;
BEGIN /* Start DataWareHouse Processes */
        StartProcess( LogUpdates) ;
        StartProcess( UpdateView) ;
END DataWarehouse ;
```

그림 4 데이터 웨어하우스 모듈 알고리즘 (계속)

5. 결론

일관성을 보장하는 대표적인 뷰 관리 기법인 SWEEP의 높은 뷰 갱신 시간을 단축하기 위해 이 논문에서는 다른 소스와의 조인을 병렬로 수행하는 PSWEEP/RI의 처리 기법과 알고리즘을 제시하였다. 이 방법에서 서브질의의 조인 연산을 병렬화 하기 위해 소스 릴레이션들간의 참조 무결성 제약조건의 특성을 이용하였다. 참조하는 릴레이션에서 발생하는 변경사항에 대한 질의 처리는 참조하는 릴레이션의 수만큼 병렬로 조인 연산을 수행하고, 참조되는 릴레이션에서 발생하는 변경사항은 뷰에 영향을 주지 않으므로 필터링된다. 따라서 본 기법은 기존 방법에서 요구되는 뷰 갱신시간을 크게 단축하고, 소스의 증가에 따른 뷰 갱신시간의 증가를 가져오는 SWEEP의 단점을 개선하였다.

향후에는 단일뷰 뿐만 아니라 복합뷰를 효율적으로 관리할 수 있는 방법 및 실체뷰에 집계값이 포함되어 있는 경우의 실체뷰 관리 기법에 대한 연구가 필요하다.

6. 감사의 글

이 논문은 첨단정보기술 연구센터를 통하여 한국과학재단의 지원을 받았음.

7. 참고문헌

[1] D. Quess, A. Gupta, I. S. Mumick, and J. Widom, "Making Views Self-Maintainable for Data Warehousing", Proc. of Conf. on Parallel and Database Information Systems, pp.158-169, 1996.

[2] K. A. Ross, D. Stivastava, and S. Sudarshan, "Materialized View Maintenance and Integrity Constraint Checking: Trading Space for Time,"Proc. of ACM SIGMOD Conf, pp.447-458, 1996.

[3] D. Agrawal, A. EL Abbadi, A. Singh, T. Yurek, "Efficient View Maintenance at Data Warehouses", In Proceedings of SIGMOD Conference, pp.417-427, 1997.

[4] Xin Zhang, Lingli Ding, Elke A. Rundensteiner, " Parallel Multi-Source View Maintenance", Submitted for publication,2002

[5] Yue Zhuge, Hector Garcia-Molina, Janet L. Wiener, "The Strobe Algorithms for Multi-Source Warehouse Consistency", In PDIS pp.146-157, 1996.