데이터웨어하우징에서 MIN/MAX질의를 고려한 뷰관리 정책

김근형* · 김두경**

View Maintenance Policy for considering MIN/MAX query in Data warehousing

Keun-hyung Kim* · Du-gyung Kim**

요 약

데이터웨어하우스에서 실체뷰는 사용자의 정보처리 요구에 신속하게 응답하기 위한 주요한 수단이다. 데이터웨어하우스내에 실체뷰의 개수가 많을수록 사용자의 질의요구를 실체뷰내에서 처리할 수 있는 확률이 높아지므로 신속한 응답이 가능하다. 데이터웨어하우스내에 유지할 수 있는 실체뷰의 개수에 대한 주요 제약은 기본릴레이션의 변화에 기인하는 실체뷰 갱신시간이다.

본 논문에서는 MIN/MAX 함수를 포함하는 실체뷰에 대해서 기본릴레이션의 MIN/MAX값의 변화가 빈번할 경우에도 실체뷰 갱신시간을 절약할 수 있는 효율적인 실체뷰 갱신정책을 제안한다. 기본릴레이션의 MIN/MAX값의 변경을 삽입/삭제연산으로 구분하여 실체뷰를 갱신하면 실체뷰의 MIN/MAX값 갱신을 위하여 기본릴레이션에 접근해야 할 횟수가 줄어든다.

ABSTRACT

Materialized views in data warehouse play important roles in rapidly answering to users's requests for information processing. More views in data warehouse, can respond to the users more rapidly because the user's requests may be processed by using only the materialized views with higher probabilities rather than accessing base relations. The limited duration, during which the materialized views are updated due to base relations's changes, limits the number of materialized views in data warehouse.

In this paper, we propose efficient policy for updating the materialized views, which can save the update duration of views although MIN/MAX values frequently change in base relation. The policy updates the materialized views by distinguishing whether MIN/MAX values's changes in the base relation are insert value or delete value. Then, the number of accesses to the base relation is descreased when updating the MIN/MAX values in the materialized views.

*제주대학교 경영정보학과 전임강사

**제주대학교 경영정보학과 교수

접수일자 : 2002. 9. 2

Ⅰ. 서 론

의사결정지원시스템(Decision Support System)은 기업 경영활동에서 비즈니스 구성원들의 합리적인 의사결정을 지원하기 위한 정보시스템으로, 사용자의 요구에 신속하게 대응하여 필요한 데이터를 추출하고 분석하는 기능을 제공한다. 90년대 중반 이후 주요 정보기술중의 하나로 떠오른 데이터웨어하우징(data warehousing)과 OLAP(On Line Analytical Processing)은 의사결정지원시스템의 중요한 요소가 되었고 현재 기업경영혁신 및 정보 전략화 프로젝트의 핵심이 되고 있다. 데이터웨어하우징이란, 조직내의 각 데이터 저장소들로부터 추출한 데이터를 중앙의 단일 데이터베이스에 통합하여 저장한 '데이터웨어하우스(data warehouse)'를 관리하는 기법을 의미한다[1]. OLAP시스템은 이러한 데이터웨어하우스에 저장된 방대한 양의 데이터에 대하여, 의사결정과정에 도움을 주는 빠르고 직접적인 자료분석 작업을 수행한다[1].

데이터웨어하우스에는 일반적으로 수백 기가 바이트(GB)에서 테라 바이트(TB)에 이르는 매우 방대한 양의 데이터가 저장된다. 또한 대부분의 의사결정지원 질의(decision support query)는 개별적인 레코드 정보의 검색이 아닌, 레코드 집합의 전체적인 정보와 경향분석 등을 수행하므로 많은 수의 집단연산(aggregation)을 포함한다. 이와 같이 데이터의 양이 방대하고 수행되는 질의가 복잡하기 때문에, 대부분의 의사결정지원 질의는 처리시 많은 시간이 소요된다. 그러나 OLAP시스템은 의사결정과정에 즉시 반영될 수 있는 직접적인 데이터 분석을 수행해야 하므로, 빠른 질의응답 속도의 보장이 필수적이다. 이를 위하여 데이터웨어하우징 분야에서는 자주 수행되는 질의의 처리에 필요한 정보를 미리 계산하여 저장하고, 실제 질의처리 시에는 저장된 정보를 활용하여 질의 응답속도를 줄이고자 하는 선계산(pre-computation)기법이 활발히 연구되고 있다.

선계산 기법은 기본릴레이션에서 자주 사용되는 종류의 데이터에 대해 SUM, AVG, COUNT, MAX, MIN등과 같은 집단함수를 적용하여 요약한 내용을 실체뷰(materialization view)의 형태로 저장한다[2]. 기업활동의 결과로 기본릴레이션의 변경이 발생하면 이를 모아두었다가 기업활동이 중단되는 밤시간을 이용하여 실체뷰에 반영 및 갱신한다. 이러한 갱신작업은 기업활동이 재개되는 다음날 아침까지는 완료되어야 하는 시간제약이 있다. 실체뷰가 많아질수록 사용자 질의를 기본릴레이션 대신 실체뷰에서 처리할 확률이 높아지게 되고 이로 인하여 신속한 처리가 가능하게 되므로 사용자의 정보요구에 대한 만족도는 높아지게 된다. 그러나 실체뷰가 많아질 수록 실체뷰 갱신시간 또한 길어지므로 유지할 수 있는 실체뷰의 개수에는 한계가 있다. 실체뷰의 갱신시간을 단축시킬 수 있다면 유지할 수 있는 실체뷰의 개수를 증가시킬 수 있으므로 사용자의 정보요구에 대한 만족도를 높일 수 있다.

본 논문에서는 MIN/MAX 함수를 고려한 실체뷰 갱신정책을 제안한다. 기본릴레이션에 대한 갱신연산을 삽입/삭제 연산으로 구분하여 관리함으로써, 실체뷰의 MIN/MAX값 갱신시 기본릴레이션으로의 접근 횟수를 최소화시킬 수 있고 결과적으로 실체뷰 갱신시간을 단축시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 간주하는 데이터웨어하우스 모델을 제시하고 3장에서는 데이터웨어하우징에서의 뷰관리 정책과 관련된 기존 연구들을 고찰한다. 4장에서 MIN/MAX질의를 고려한 새로운 점진적 뷰관리 정책을 제안한다. 5장에서 제안한 정책의 성능을 평가 분석하고 6장에서 결론을 맺는다.

Ⅱ. 데이터웨어하우스 모델

그림 1에서 볼 수 있는 것처럼, 일반적으로 데이터웨어하우스에는 외부의 운영계 데이터(operational data) 또는 데이터웨어하우스 내에 저장된 데이터에 의해 정의된 실체뷰들(MV1, MV2, MV3,...)이 저장된다. 흔히 데이터웨어하우스에는 외부의 데이터들이 가공 및 정제의 과정을 거쳐 「사실테이블(fact table)」이나 「차원테이블(dimension table)」로 저장된다. 이때 사실테이블과 차원테이블들로부터 「요약테이블(summary table)」이 정의되어 저장되기도 하는데 이 경우 요약테이블은 사실테이블과 차원테이블을 기본릴레이션(Base Relation)으로 하는 실체뷰라고

볼 수 있다.

데이터웨어하우징은 데이터웨어하우스내의 기본릴레이션이나 실체뷰 등을 생성하고 갱신하는 등 데이터웨어하우스를 관리하는 작업이라고 할 수 있다. 실체뷰의 생성과 관련하여 사용자의 신속한 응답요구를 만족시킬 수 있고 저장공간을 절약할 수 있는 실체뷰들이 선택되어야 한다. 운영계 데이터의 변화는 데이터웨어하우스내의 기본릴레이션들의 변경을 유도하고 또한 실체뷰의 변경을 유발한다. 이러한 실체뷰 및 기본릴레이션의 갱신작업을 데이터웨어하우징의 뷰관리 기능이라고 한다[2].

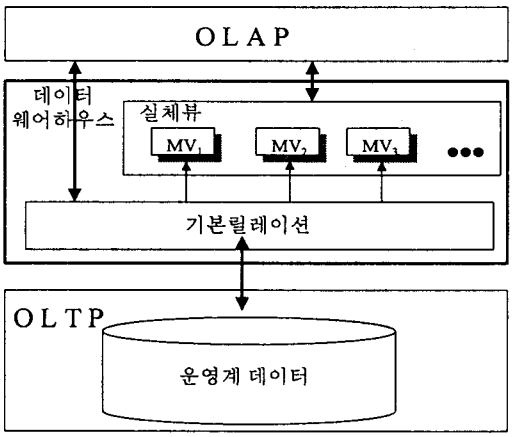

그림 1. 데이터웨어하우스 모델

전통적인 뷰관리 정책으로 즉각갱신(immediate update), 지연갱신(deferred update), 스냅샷갱신(snapshot update)이 있는데 데이터웨어하우징에서는 스냅샷갱신이 바람직하다[3, 4]. 기본릴레이션의 변경을 즉각적으로 실체뷰에 반영하면 시스템의 부담으로 사용자 요구에 대한 응답시간이 느려지므로 하루나 일주일의 간격을 두어 주기적으로 실체뷰의 갱신작업을 수행하는 스냅샷 갱신기법이 일반적이다[4]. 스냅샷갱신은 기업활동으로 발생한 기본릴레이션의 변경을 모아두었다가 기업활동이 중단되는 밤시간을 이용하여 실체뷰에 반영 및 갱신한다. 이러한 갱신작업은 기업활동이 재개되는 다음날 아침까지는 완료되어야 하는 시간제약이 있다. 실체뷰가 많아질수록 사용자 질의를 기본릴레이션 대신 실체뷰에서 처리할 확률이 높아지게 되고 이로 인하여 신속한 처리가 가능하게 되므로 사용자의 정보요구에 대한 만족도는 높아지게 된다. 그러나 실체뷰가 많아질 수록 실체뷰 갱신시간 또한 길어지므로 유지할 수 있는 실체뷰의 개수에는 한계가 있다. 스냅샷 갱신시간을 단축시킬 수 있는 뷰관리 정책이라면 유지할 수 있는 실체뷰의 개수를 증가시킬 수 있으므로 사용자의 정보요구에 대한 만족도를 높일 수 있다. 점진적 뷰관리 정책은 실체뷰 갱신시간을 단축시킬 수 있는 방법중의 하나로, 기본릴레이션의 변화가 생길 때 뷰를 전부 새로 계산하지 않고 뷰의 변경될 부분만을 계산하는 방법이다[2, 5, 6]. 실체뷰 중에서 집단함수를 포함하는 실체뷰는 갱신작업에 많은 시간이 소요된다. SQL에서는 표준집단함수로서 SUM, AVG, COUNT, MAX, MIN의 5가지를 지정하고 있다. SUN, AVG, COUNT함수의 실체뷰 갱신은 점진적 뷰관리 정책으로 처리할 수 있지만, MIN/MAX의 처리는 기본릴레이션에 접근해야 한다고 알려져 있다[3]. 따라서 MIN/MAX의 처리와 관련한 연구들은 기본릴레이션에서의 MIN/MAX값을 효율적으로 처리하기 위한 방안들이었다[7, 8].

III. 선행연구

1. 기존 관련연구들의 고찰

사용자의 정보요구에 대한 신속한 응답과 저장공간의 절약을 목적으로 데이터웨어하우징에서 실체뷰를 효율적으로 선택하려는 연구들이 있었다. [9]는 조인 비용을 기반으로 한 실체뷰 선택 알고리즘을 제안하고 있고 [10]은 다양한 목적하의 뷰선택 문제를 일반화시킬 수 있는 프레임웍을 제안하였다. [11]은 데이터큐브의 일부를 나타내는 세분화된 뷰의 표현 및 선택 문제를 다루고 있다. 데이터웨어하우징에서 실체뷰의 갱신작업을 효율적으로 수행하여 사용자의 정보요구에 대한 신속한 응답을 제공하려는 연구들이 있었는데 [2, 5, 6]은 운영계 데이터 또는 기본릴레이션의 변화가 생길 때 뷰를 전부 새로 계산하지 않고 뷰의 변경될 부분만을 계산하여 신속한 뷰갱신작업을 처리하기 위한 점진적 뷰관리 기법들을 제안하고 있다. [4]는 OLTP(On Line Transaction Processing) 환경에서의 뷰관리 정책을 다루고 있는데, 지연갱신,

즉각갱신, 스냅샷 갱신의 장단점을 고려하여 효율적으로 혼합 적용할 수 있는 방안을 제안하고 있다. [12, 13]은 데이터웨어하우징 환경에서 임의의 사용자 질의를 특정 척도를 기준으로 평가하여 자동적으로 실체뷰화시키는 정책을 제안하고 있다. [3]은 데이터웨어하우징 환경에서 점진적 뷰관리 정책과 스냅샷 갱신방식이 적용될 때, 실체뷰 갱신을 위한 선계산 작업을 하는 과정인 전파(propagation)단계와 실질적인 갱신작업을 하는 과정인 회복(refresh)단계로 나누어 갱신시간을 단축시킬 수 있는 알고리즘을 제안하고 있다. [3]에서 제안하고 있는 알고리즘은 MIN/MAX 집단함수를 포함하는 실체뷰의 경우, 기본릴레이션의 MIN/MAX값의 변경 정도가 많아지면 실체뷰의 갱신시간은 길어지는 단점이 있다. 현실적인 예로 날짜와 부서를 차원으로 하면서 영업사원의 최저 실적과 최고 실적을 보유하는 실체뷰의 경우, 기본릴레이션에서 영업사원 실적의 MIN/ MAX 값은 매일 변경될 것이다. [7, 8]은 효율적인 인덱싱 등의 기법을 사용하여 기본릴레이션에서 MIN/MAX 집단함수를 신속하게 처리할 수 있는 방안을 제안하고 있다. 본 논문에서는 [3]의 알고리즘을 개선하여 기본릴레이션의 MIN/MAX값의 변경이 자주 일어나더라도 실체뷰의 갱신시간을 단축시킬 수 있는 알고리즘을 제안할 것이다.

2. 점진적 뷰관리 기법

어떤 실체뷰가 기본릴레이션들에 의해 정의되어 있을 때 기본릴레이션의 변화를 뷰에 반영하는 방법은 크게 다음의 두 가지로 나뉜다.

▶ 재계산(recomputation): 각각의 기본릴레이션들을 먼저 갱신하고 갱신된 기본릴레이션들로부터 실체뷰들을 새로 계산한다. 기본릴레이션의 변화가 적을 경우에 이렇게 실체뷰 전체를 다시 계산하는 일은 비효율적이다.

▶ 점진적 관리(incremental maintenance): 기본릴레이션들과 이들 중 변경된 튜플들만으로 된 차이릴레이션(delta relation)들로부터 변경된 뷰의 내용만을 계산한 뒤, 이 계산된 내용을 뷰에 반영하는 방법이다. 기본릴레이션의 변화가 적은 경우, 이 방법은 재계산에 비해 효율적이다.

예제: 데이터웨어하우스에 기본릴레이션 pos, items가 저장되어 있고 이들로부터 실체뷰 sales가 정의되어 있다고 하자.

pos(storeID, itemID, date, qty, price)
items(itemID, name, category, cost)
sales(storeID, category) =
pos(storeID, itemID, date, qty, price)
⋈ items(itemID, name, category, cost)

pos와 sales의 차이릴레이션을 각각 Δpos와 Δsales, pos와 sales의 갱신결과를 각각 pos', sales'이라고 할 때 표 1은 재계산 방법과 점진적 관리에 의한 실체뷰의 갱신작업과정을 나타내고 있다.

표 1. 재계산과 점진적 관리방법

	실체뷰 갱신과정
재계산	[step1]: pos' = pos + Δpos [step2]: sales' = pos' ⋈ items
점진적 관리	[step1]: Δsales = Δpos ⋈ items [step2]: pos' = Δpos ∪ pos [step3]: sales' = Δsales ∪ sales

점진적 관리방식의 기존 연구들은 두 개 이상의 기본릴레이션이 변경될 때, 실체뷰를 갱신하기 위한 효율적인 표준식들을 제안하고 있으나 본 논문에서는 하나의 기본릴레이션의 변경만을 가정하겠다. 반면, 점진적 관리 관련 대부분의 기존 연구들은 SPJ (Selection-Projection-Join) 식으로 표현되는 뷰만을 주로 고려하였지만 본 논문에서는 집단함수나 Group By와 같은 식이 포함되어 있는 뷰로 확장하여 논의한다. 아래의 질의는 실체뷰 sales의 구체적인 정의를 나타내고 있다.

```
CREATE VIEW sales(storeID, category,
    TotalCount, EarliestSale, TotalQuantity) As
SELECT storeID, category,
        COUNT(*) AS TotalCount,
        MIN(date) AS EarliestSale, ,
        SUM(qty) AS TotalQuantity
```

```
FROM pos, items
WHERE pos.itemID = item.storeID
GROUP BY storeID, category
```

3. 자립유지(self-maintainable) 집단함수

집단함수(Aggregation Function)는 세 부류, 즉 분배 집단함수(distributive), 대수 집단함수(algebraic), 홀리스틱 집단함수(holistic)로 구분된다[3]. 분배 집단함수는 처리할 영역의 데이터들을 분할하여 각각을 계산한 다음, 각각의 결과를 통합하여 계산하면 전체 영역에 대한 타당한 결과를 얻을 수 있는 함수이다. SQL 표준 집단함수 중 COUNT, SUM, MIN, MAX는 분배 집단함수이다. 예를 들면, COUNT는 분할된 영역의 COUNT 결과값들을 다시 합하여 전체 영역의 최종적인 결과를 유도할 수 있다. 대수 집단함수는 분배 집단함수의 수식 계산으로 표현될 수 있는 함수이다. 예를 들면, AVG는 SUM/COUNT로 표현될 수 있다. 홀리스틱 함수는 계산할 영역을 분할하여 처리할 수 없는 함수이다. MEDIAN이 그 예이다.

<정의 1> 자립유지 집단함수

실체뷰에 포함된 집단함수의 새로운 결과값(new value)을 구하는 경우, 기본릴레이션의 변경부분과 집단함수의 이전 결과값(old value)에 의해서 타당하게 계산될 수 있다면, 그 집단함수는 자립유지가 가능하다. □

분배 집단함수는 삽입 및 삭제갱신에 대해서 일반적으로 자립유지가 가능하지만 MIN/MAX의 경우는 삽입갱신일 때만 자립유지가 가능하고 삭제갱신일 경우는 자립유지가 불가능하다[3].

Ⅳ. 새로운 실체뷰관리 정책의 제안

1. 개요

데이터웨어하우스에서 유지하는 실체뷰들의 갯수가 많을수록 사용자의 정보요구에 대한 만족도는 높아진다. 왜냐하면, 사용자들의 다양한 요구질의들에 대하여 기본릴레이션에 의존하기보다는 실체뷰들을 이용하여 보다 신속하게 응답할 수 있는 확률이 높아지기 때문이다. 많은 실체뷰들을 유지하기 위한 제약사항으로 첫번째, 저장공간의 한계를 고려해야 하지만 기억장치 가격의 점차적 하락으로 저장용량의 제약은 비교적 민감하지 않다[12]. 두번째 제약사항으로 기본릴레이션의 변화에 따른 실체뷰의 갱신시간이다. 실체뷰의 갱신작업이 이루어지는 동안에는 데이터베이스 일관성(consistency)유지 측면에서 실체뷰들을 잠금상태(lock)로 만들어야 하고 데이터웨어하우스로의 정보조회 요구는 중단된다. 따라서 실체뷰의 갱신작업은 기업 경영활동이 중단되는 밤시간(night duration)을 이용하여 이루어져야 하는데, 각 실체뷰의 갱신시간이 짧을수록 보다 많은 수의 실체뷰들을 제한된 시간(밤시간)내에 갱신할 수 있다. 이런 관점에서 볼 때, 데이터웨어하우징에서의 뷰관리의 중요한 목적중의 하나는 실체뷰들의 갱신시간을 단축시키는 것이다[3]. 본 논문에서는 실체뷰 갱신시간을 줄이기 위하여 실체뷰 갱신작업을 3단계로 나누어서 처리할 수 있다. 첫번째 단계인 전파단계(P, propagation phase)에서는 실체뷰를 잠금상태로 만들지 않고 신속한 갱신작업을 하기 위한 선계산(pre-computation)을 하는 과정이다. 기본릴레이션의 변화부분인 차이릴레이션에 대하여 실체뷰의 정의를 적용하여 실체뷰의 차이릴레이션을 만드는 과정이다. 두번째 단계인 식별단계(D, distinguish phase)에서도 신속한 갱신작업을 위한 선계산 과정의 일부인데, P단계 동안에 생성된 실체뷰의 차이릴레이션의 MIN/MAX 값이 삽입된 값인지 삭제된 값인지의 여부를 판별하는 과정이다. 삽입된 값이면 자립유지가 가능하여 기본릴레이션에 접근할 필요가 없고 삭제된 값이면 기본릴레이션에 접근하여 새로운 MIN/MAX값을 계산해야 하기 때문이다. [3]에서는 이러한 부분을 고려하지 않기 때문에 기본릴레이션의 MIN/MAX값의 변경이 잦은 경우 실체뷰의 MIN/MAX값을 구하기 위하여 기본릴레이션에 접근하는 횟수가 많아지게 되어 갱신시간이 느려지게 된다. 세번째 단계인 회복단계(R, refresh phase)에서는 P단계, D단계동안에 생성된 실체뷰의 차이릴레이션을 기반으로 실체뷰를 갱신하는 과정이다.

2. P단계

기본릴레이션의 차이릴레이션은 삽입화일과 삭제

화일로 구분하여 로그화일 등의 형태로 저장되어 있다. 실체뷰의 차이릴레이션을 만들기 위하여 뷰정의에 포함된 집단함수 SUM, COUNT를 기본릴레이션의 차이릴레이션에 적용할 때, 삭제화일에는 음수(-) 연산을 적용한다. 그림 2는 실체뷰 sales의 차이릴레이션 D_sales를 계산하는 과정을 나타내고 있다. 그림 2에서 pos_ins는 기본릴레이션 pos의 삽입갱신 튜플들을 포함하는 삽입화일이고 pos_del은 pos의 삭제화일이다. 실체뷰 D_sales의 DEL 속성은 식별단계에서 기본릴레이션의 차이릴레이션들(pos_ins, pos_del)에 의해 계산된 MIN/MAX 값이 삽입된 값인지 삭제된 값인지를 기록하기 위하여 필요하다.

```
CREATE VIEW D_sales (
SELECT storeID, category,
        SUM(count) AS D_count,
        MIN(date) AS  EarliestSale,
        SUM(quantity) AS D_quantity
        "No" AS DEL
FROM ((SELECT storeID, category, date
                1 AS count, qty AS quantity
         FROM pos_ins, items
         WHERE pos_ins.itemID = items.itemID)
         UNION ALL
         (SELECT storeID, category, date
                -1 AS count, -qty AS quantity
         FROM pos_del, items
         WHERE pos_del.itemID = items.itemID))
GROUP BY storeID, category
```

그림 2. P단계의 예

3. D단계

이 단계에서는 기본릴레이션의 차이릴레이션들(pos_ins, pos_del)에 의해 계산된 실체뷰의 MIN/MAX 값이 삽입된 값인지 삭제된 값인지를 판별하는 과정이다. D_sales의 MIN/MAX 속성인 EarliestSale의 값은 삽입튜플과 삭제튜플을 구분하지 않고 계산된 것이므로 이를 구분하기 위하여 삭제화일의 특정 속성값들과 비교하는 과정이 필요하다. 그림 3은 D_sales의 MIN/MAX 속성인 EarliestSale 의 값의 삭제여부를 판별하여 DEL 속성값으로 삭제 태그 "yes" 또는 "no"값을 설정하는 SQL질의이다.

```
UPDATE D_sales
 SET  DEL = "yes"
 WHERE  EXISTS (
        select *
        from  pos_del
        where D_sales.storeID = pos_del.storeID
        and D_sales.EarliestSale = pos_del.date);
```

그림 3. D단계의 예

4. R단계

이 단계에서는 P단계, D단계 동안에 생성된 실체뷰의 차이릴레이션 DRi을 기반으로 실체뷰 MVi를 갱신하는 과정이다. 그림 4는 R단계를 위한 알고리즘을 나타내고 있다.

```
     /* MVi 는 실체뷰 중의 하나 */
     /*DRi 는 실체뷰 MVi 의 차이릴레이션*/
   For each tuple δt in DRi {
     Let tuple t = tuple in MVi having the same values
           for its group-by attributes as δt
     If t is not found
(1)    { Insert tuple δt into MVi } /* 튜플 삽입 */
     Else   /* COUNT함수를 이용하여 튜플 t가 삭제될
           필요가 있는지 체크 */
       If δt.COUNT(*) + t..COUNT(*)= 0
(2)    { Delete tuple t from MVi }
     /* MIN/MAX 값을 계산하기 위하여 기본릴레이션으
        로의 접근여부 체크 */
     Else {
(3)    recompute = false
     for each aggregation function a(e) in the MVi
(4)    { if ((a is MIN function) AND
             (δt.MIN(e) ≤ t.MIN(e)) AND
             (δt.DEL = "YES")) OR
             ((a is MAX function) AND
             (δt.MAX(e) ≥ t.MAX(e)) AND
             (δt.DEL = "YES"))
(5)              { recompute = false }
       if (recompute)
(6)              { Update tuple t by recomputing its
                   aggregate functions from the
                   base data from t's group.}
       else {
          if a is COUNT or SUM
(7)            t.a = t.a + δt.a
          else If a is MIN
(8)            t.a = MIN(t.a, δt.a)
          else If a is MAX
(9)            t.a = MAN(t.a, δt.a) }
     }      }
```

그림 4. R단계 알고리즘

DRi에 있는 튜플 δt와 MVi의 튜플 t는 Group By 연산이 적용된 튜플이다. R단계 알고리즘은 DRi에 있는 각 튜플 δt에 대하여 실체뷰의 Group By 속성값을 기준으로 MVi에서 부합되는 튜플 t를 찾아 갱신작업을 수행한다. MVi에 있는 각 튜플들은 Group By 연산이 적용된 결과이기 때문에 DRi에 있는 튜플 δt와 부합하는 튜플은 유일하거나 존재하지 않는다. R 단계 알고리즘의 (1)의 단계를 통하여 DRi에는 존재하고 MVi에는 존재하지 않는 튜플은 삽입된다. 즉, DRi에 있는 임의의 튜플 δt에 대하여 실체뷰의 Group By 속성값을 기준으로 부합하는 튜플이 MVi에 존재하지 않는 경우 그 튜플 δt는 MVi에 삽입된다. δt.COUNT(*)는 Gropu By연산에 의하여 δt를 유도한 튜플들의 갯수이다. δt를 유도한 튜플들은 삽입 튜플들과 삭제튜플들이 혼합되어 있는 경우이므로 δt. COUNT(*) 값은 정수값이다. δt.COUNT(*)값이 음수인 경우 δt를 group by 연산에 의하여 유도한 튜플들은 삽입튜플들보다 삭제튜플들이 더 많다는 의미이다. δt와 t가 group by 속성값을 기준으로 부합하고 「δt .COUNT(*) + t.COUNT(*)= 0」이면 (2)의 단계를 통하여 MVi 의 튜플 t 는 삭제되어야 한다.

δt 와 t 가 Group By 속성값을 기준으로 부합하고 「δt .COUNT(*) + t.COUNT(*) > 0」이면 튜플 t 의 속성값들을 δt 의 속성값들을 기반으로 갱신해야 한다. 튜플 t 의 속성값이 MIN/MAX 함수에 의하여 유도되었을 경우 δt 의 MIN/MAX값의 크기와 비교하고 삭제 태그를 체크하여 기본릴레이션으로부터의 재계산(recomputation) 여부를 결정한다. 재계산이 필요없을 경우 δt 의 속성값들과 기존의 t 의 속성값을 이용하여 새로운 t 의 속성값들을 계산한다.

V. 성능평가 및 분석

본 논문에서는 [3]에서 제안하고 있는 알고리즘(이후로 delta 알고리즘으로 명명함)을 개선하여 P단계, D단계, R단계의 3단계로 구성된 실체뷰 관리정책(이후로 proposed 알고리즘으로 명명함)을 제안하였다. proposed 알고리즘은 MIN/MAX 함수가 실체뷰 정의에 포함되고 기본릴레이션의 MIN/MAX값의 변화가 빈번한 상황에 효율적으로 적용되어 실체뷰의 갱신시간을 단축시킬 수 있다.

proposed 알고리즘의 성능을 평가하기 위하여, 표 1에서 고찰했던 재계산(이후로 recomp로 명명함)방법과 점진적 뷰관리 방법중의 하나인 delta 알고리즘과 비교하였다. 성능평가를 위하여 비교대상의 각 알고리즘들을 실제로 구현하였다. 구현은 펜티엄III dual CPU를 장착한 PC서버상에서 윈도2000기반의 오라클9i버전과 응용프로그램 개발툴인 Pro-C/C++을 이용하여 이루어졌다. 테스트 데이터베이스 스키마는 3.2절의 예제에서 고려했던 기본릴레이션 pos, item과 실체뷰 sales와 동일하다. pos 릴레이션은 1,000,000튜플들을 포함하고 item 릴레이션은 1,000 튜플들을 포함한다. pos테이블은 (storeID, itemID, date)를 기준으로 인덱싱되어 있고 실체뷰 sales도 group by 속성들을 기준으로 인덱싱되어 있다. 기본릴레이션 pos의 차이릴레이션들 pos_ins와 pos_del의 크기는 1,000튜플에서 10,000튜플까지 변한다. 기본릴레이션 pos의 차이릴레이션들의 유형은 다음과 같이 2가지로 분류된다.

▶ 갱신유발변화(update-generating changes)

기본릴레이션의 임의의 튜플들의 속성값들을 변경한다. 특정 튜플의 속성값을 변경하는 과정은 그 튜플을 삭제한 후 변경된 값을 포함하는 새로운 튜플을 삽입하는 것과 같다. 따라서, 동일한 수의 삽입튜플과 삭제튜플을 포함하는 변경이다.

▶ 삽입유발변화(insertion-generating changes)

기본 릴레이션에 임의의 튜플들을 삽입하는 변경이다. 데이터웨어하우스의 기본릴레이션의 변화는 삽입유발변화가 일반적이다[3].

그림 5는 기본릴레이션 pos에 대한 갱신유발변화가 발생한 경우 실체뷰 sales의 갱신작업을 위하여 경과한 전체적인 시간(delta의 경우 P+R, proposed의 경우, P+D+R)을 비교하고 있다. pos_ins 및 pos_del 안의 MIN값은 랜덤함수에 의해서 결정된다. 실험결과, 실체뷰 sales의 MIN값보다 작을 확률이 1% 미만이었다. proposed 알고리즘은 delta 알고리즘에 비하여 D단계를 더 포함하고 있지만 R단계의 경과시간이 단축되므로 전체적인 경과시간은 줄어든다.

그림 6은 delta 알고리즘과 proposed 알고리즘에 대하여 갱신유발변화가 발생한 경우 P단계와 R단계를 완성하기 위하여 경과한 시간을 비교하고 있다. P단계는 비슷하고 R단계는 proposed 알고리즘이 우수함을 알 수 있다. 실제로 R단계에서의 경과시간이 중요한데, P단계나 D단계에서는 실체뷰를 잠금상태로 만들지 않으므로 사용자의 접근을 허락하지만 R단계에서는 실체뷰를 잠금상태로 설정하므로 R단계를 처리하기 위한 시간은 가능한한 짧아야 한다.

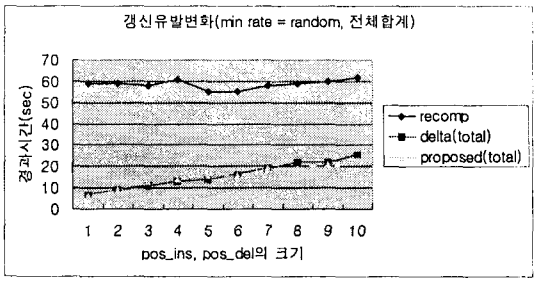

그림 5. 전체적인 갱신작업의 경과시간 비교

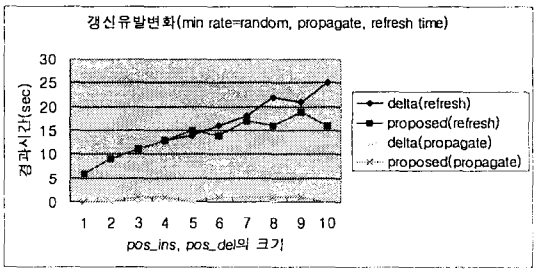

그림 6. P단계, R단계의 경과시간 비교

그림 7은 proposed 알고리즘에서 D단계를 처리하기 위하여 D_sales와 pos_del, pos_ins에 인덱스를 설정한 경우와 그렇지 않은 경우의 경과시간을 보여주고 있다. 'distinguish(index)'는 인덱스를 생성하는 시간과 삭제여부를 판별하는 시간을 포함한다. 그림5에서의 D단계의 처리시간은 distinguish(index)를 가정하고 있다.

그림 8은 기본릴레이션 pos에 대한 삽입유발변화가 발생한 경우 실체뷰 sales의 갱신작업을 위하여 경과한 전체적인 시간(delta의 경우 P+R, proposed의 경우, P+D+R)을 비교하고 있다. proposed 알고리즘이 타 알고리즘에 비하여 우수함을 알 수 있다.

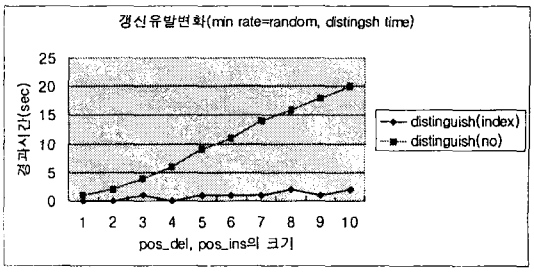

그림 7. D단계의 경과시간 비교

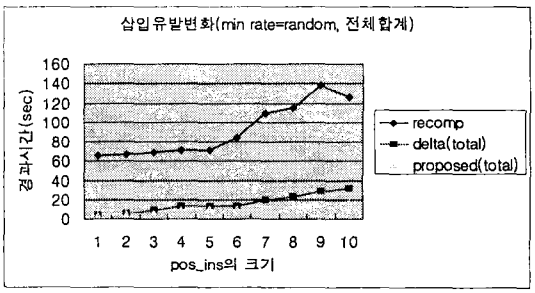

그림 8. 삽입유발변화의 전체 경과시간 비교

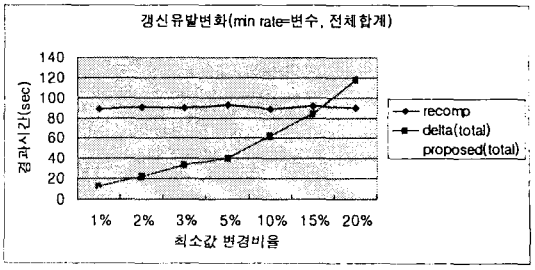

그림 9. 최소값 변경비율에따른 경과시간 비교

그림 9는 기본릴레이션 pos에 대한 갱신유발변화가 발생한 경우 실체뷰 sales의 갱신작업을 위하여 경과한 전체적인 시간(delta의 경우 P+R, proposed의 경우, P+D+R)을 비교하고 있다. 그러나 pos_ins 및 pos_del 안의 MIN값을 인위적으로 조절하여 갱신작업의 처리시간을 비교하고 있다. pos_ins 및 pos_del 안의 MIN값이 실체뷰 sales의 MIN값보다 작을 확률이 높아질수록 proposed 알고리즘은 뛰어난 성능을 보여주고 있다.

Ⅵ. 결 론

데이터웨어하우스내의 기본릴레이션에서 자주 사용되는 종류의 데이터에 대해 이를 정리하고 요약한 실체뷰는 실제 질의처리 시에 질의응답 속도를 줄일 수 있어서 사용자의 정보요구 만족도를 높이는 주요한 수단이다. 데이터웨어하우스에서 유지하는 실체뷰들의 갯수가 많을수록 사용자의 정보요구에 대한 만족도는 높아지지만 전체적인 실체뷰 갱신시간의 제약으로 인하여 유지할 수 있는 실체뷰의 개수에는 한계가 있다. 전체적인 실체뷰 갱신작업시간의 제약하에서, 각 실체뷰의 갱신시간이 짧을수록 보다 많은 수의 실체뷰들을 데이터웨어하우스 내에 포함할 수 있다.

본 논문에서는 MIN/MAX 집단함수가 포함된 실체뷰의 갱신시간을 단축시키기 위한 알고리즘을 제안하였다. 기존의 delta 알고리즘은 실체뷰의 차이릴레이션을 계산할 때 MIN/MAX값의 삽입·삭제 여부를 구별하지 않음으로써 MIN/MAX값의 갱신을 위한 재계산 확률이 높았었다. 그러나 proposed 알고리즘은 실체뷰의 차이릴레이션을 계산할 때 MIN/MAX값의 삽입·삭제 여부를 구별함으로써 실질적인 갱신작업시 삭제된 MIN/MAX값에 대해서만 재계산을 수행하므로 갱신작업시간이 단축된다. 실험결과, MIN/MAX값의 삽입·삭제 여부를 구별하는 과정도 많은 시간이 소요되지 않음을 알 수 있었고, 실체뷰의 전체적인 갱신시간이 단축되었음을 알 수 있었다. 특히, 기본릴레이션의 차이릴레이션 안에 있는 MIN값이 실체뷰내의 MIN값보다 작을 확률이 높아질수록 proposed 알고리즘은 뛰어난 성능을 보여줌을 알 수 있었다.

참고문헌

[1] 조재희, "데이터웨어하우징 기술을 이용한 DB 마케팅 전략에 관한 연구", 정보기술과 데이터베이스 저널, 제 6권, 제1호, pp.104-113, 1998.

[2] 이기용, 김명호, "데이타웨어하우스에서 효과적인 점진적 뷰 관리", 정보과학회논문지, VOL. 27 NO. 02, pp. 175-184, 2000.

[3] I.S. Mumick, D. Quass, B.S. Mumick, "Maintenance of Data Cubes and Summary Tables in a Warehouse", In proceedings of ACM SIGMOD Conference, pp.100-111, 1997.

[4] L.S. Colby, A. Kawaguchi, D.F. Lieuwen, I.S. Mumick, "Supporting Multiple View Maintenance Policies", In proceedings of ACM SIGMOD Conference, pp.405-416, 1997.

[5] Y. Zhuge, H. Garcia-Molina, J. Hammer, J. Wodim, "View Maintenance in a Warehousing Environment", In proceedings of ACM SIGMOD Conference, pp.316-327, 1995.

[6] D. Agrawal, A.El Abbadi, A. Singh, T. Yurek, "Efficient View Maintenence at Data Warehouse", In proceedings of ACM SIGMOD Conference, pp.417-427, 1997.

[7] 양우석, 김명호, "OLAP 환경에서 다중점 MAX/MIN 질의의 효율적인 처리기법", 정보과학회논문지, VOL.27 NO.1, pp. 13-21, 2000.

[8] 정희정, 김동욱, 김종수, 이윤준, 김명호, "OLAP에서 MAX-of-SUM 질의의 효율적인 처리기법", 정보과학회논문지, VOL. 27, NO. 02, pp. 165-174, 2000.

[9] 윤원식, 신동천, "데이타웨어하우스 환경에서 조인비용을 기반으로 한 실체뷰 선택알고리즘", VOL.28, NO.1, 03, pp.31-41, 2001.

[10] D. Theodoratos, M. Mouzeghoub," A General Framework for the View Selection Problem for Data Warehouse Design and Evolution", In proceedings of ACM SIGMOD Conference, pp.1-8, 2000.

[11] 김민정, 정연돈, 박웅제, 김명호, "데이터 큐브에서 세분화된 뷰실체화 기법", 정보과학회논문지, VOL.28, NO.4, pp.587-595, 2001.

[12] Y. Kotidis, N. Roussopoulos," DynaMat: A Dynamic View Management System for Data Warehouses", In proceedings of ACM SIGMOD Conference, pp.371-382, 1999.

[13] Y. Kotidis, N. Roussopoulos,, "A Case for Dynamic View Management", ACM Transactions on Database Systems, Vol.26, No.4, pp.388-423, December 2001.

저자소개

김근형(Keun-hyung Kim)

1990년 2월 서강대학교 컴퓨터학과(공학사)

1992년 2월 서강대학교 컴퓨터학과(공학석사)

2001년 2월 서강대학교 컴퓨터학과(공학박사)

1992년~1994년 현대전자 소프트웨어연구소

2001년~현재 제주대학교 경영정보학과 전임강사

※관심분야: 데이터베이스, MIS, 데이터마이닝, 데이터웨어하우스

김두경(Du-gyung Kim)

1976년 2월 서강대학교 전자공학과(공학사)

1983년 8월 중앙대학교 전자공학과(공학석사)

1993년 2월 중앙대학교 전자공학과(공학박사)

1998년~현재 제주대 경영정보학과 교수

※관심분야: 시스템분석, 정보통신