데이터베이스 클러스터 기반의 데이터 웨어하우스에서 실체화 뷰 저장 기법

최준호[0*], 장용일*, 박순영*, 배해영*
*인하대학교 컴퓨터·정보공학과
{jhChoi[0], himalia, sunny, hybae}@dblab.inha.ac.kr

The Materialized View Storage Method in a Data Warehouse using Database Cluster

Jun-Ho Choi[0*], Yong-Il Jang*, Soon-Young Park*, Hae-Young Bae*
*Dept. of Computer Science & Engineering, Inha Univ

요 약

데이터 웨어하우스는 OLAP의 질의 처리 성능을 높이고 사용자에게 빠른 응답을 제공하기 위해 데이터 큐브의 결과를 실체화된 뷰로 저장한다. 최적의 사용자 응답 시간을 제공하기 위해서는 데이터 큐브의 전체를 저장하는 것이 좋지만 실체화 뷰는 일반적으로 물리적 저장소에 저장되기 때문에 데이터 큐브 전체를 저장하는 것은 저장 공간의 오버헤드를 초래하는 문제점을 가진다.

본 논문에서는 데이터베이스 클러스터에 대용량의 실체화 뷰를 저장하는 기법을 제안한다. 제안하는 기법은 실체화 뷰의 선택 기준으로 뷰의 실체화 이익과 뷰들 간의 의존성을 데이터베이스 클러스터 환경에 맞게 제시하고 선택 기준에 따라 실체화 뷰를 서로 다른 노드에 저장함으로서 각 노드들의 실체화 이익을 균등하게 유지한다. 이는 질의가 하나의 노드에 집중되는 현상을 방지함으로서 각 노드의 효율성을 최대로 높일 수 있는 기법이다.

1. 서 론

데이터 웨어하우스는 의사 결정을 효과적으로 지원하기 위해 통합 및 정제의 과정을 통해 수년 간 축적된 데이터를 주제별로 통합하여 저장해 놓은 데이터 저장소이며, OLAP(On-Line Analytical Processing)은 의사 결정자가 데이터 웨어하우스에 접근하여 대화 식으로 다차원적 정보를 분석하고 의사 결정에 활용하는 과정이라 할 수 있다[1]. OLAP 응용 프로그램의 사용자는 의사 결정시 다양하고, 동적인 요청에 대해 빠른 응답 시간을 요구한다. 그러나 일반적으로 대부분의 OLAP 질의는 특정 레코드를 다루는 것 보다는 전반적인 동향을 분석하기 위한 것으로 하나 이상의 집계(aggregate) 함수와 group-by 연산자가 포함되기 때문에 사용자가 요구하는 응답 시간을 충족시키기 어렵다. 따라서 이러한 문제점을 해결하기 위해 사용자가 자주 요청하는 질의의 결과를 미리 계산하여 결과를 저장하는 실체화 뷰(materialized view)기법이 제안되었다. 실체화 기법은 질의 수행에 필요한 데이터 큐브의 일부를 미리 계산하여 저장해 두고, 실제 질의 처리 시에 저장된 결과를 이용하여 사용자에게 빠른 응답 시간을 제공하는 방법이다. 일반적으로 데이터 큐브 모두를 저장하는 경우 최적의 질의 수행 속도를 얻을 수 있지만, 저장 공간의 제한으로 인해 저장할 수 있는 뷰의 개수는 제한되어 있다[2xx]. 이러한 저장 공간의 문제점을 해결하기 위해서 실체화 할 뷰의 선택 문제에 관한 많은 연구가 진행되어 왔으며, 기존 연구의 접근 방향은 저장 공간의 제한 문제를 해결하기 위해서 실체화 뷰의 개수를 줄이거나 실체화 뷰의 크기를 줄이기 위해 뷰의 일부분만을 선택해 저장하는 기법들이 제안되었다[2, 3, 4, 5, 6].

본 논문[1]에서는 기존의 실체화 뷰의 선택 기법과 데이터베이스 클러스터 시스템을 이용해 실체화 뷰의 저장 공간에 관한 문제점을 해결하는 기법을 제안하고자 한다. 데이터베이스 클러스터 시스템은 여러 개의 노드로 구성되며, 각 노드는 독립적으로 질의를 처리할 수 있는 시스템이다. 보통 2 ~ 4개의 노드가 묶여서 하나의 시스템처럼 작동하며, 빠른 응답 시간과 가용성을 지원하기 위하여 분할과 복제를 이용한다. 이러한 데이터베이스 클러스터 시스템에 연구 논문 [2]의 뷰 선택 기법을 이용해 선택되어진 실체화 뷰를 실체화 이익 비용과 뷰들 간의 의존성 두 가지 요소를 고려해 각 노드에 저장하게 된다. 즉 실체화 이익이 큰 뷰들을 서로 겹치지 않게 각 노드에 저장하고, 이미 존재하는 실체화 뷰로부터 적은 처리 비용으로 유도되어질 수 있는 뷰는 같은 노드에 저장하지 않는다. 이 논문에서 제시한 기법을 따름으로서 모든 노드의 실체화 뷰의 비용은 균등하게 유지가 될 것이고 따라서 어느 하나의 노드로 질의가 집중되는 현상을 방지할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 기반 환경인 데이터 큐브와 데이터베이스 클러스터에 대해서 알아보고 3장에서는 본 논문에서 제안하는 실체화 뷰를 데이터베이스 클러스터 환경에 맞게 저장하는 기법에 대해 기술한다. 마지막으로 4장에서는 결론 및 향후 연구를 기술한다.

2. 관련 연구

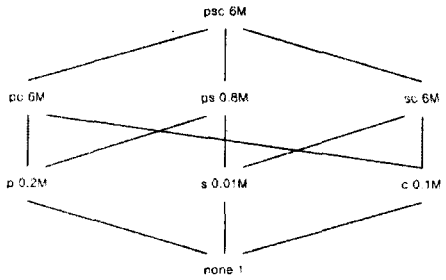
이 장에서는 데이터 큐브와 본 연구의 기반 시스템인 데이터베이스 클러스터에 대해서 기술한다.

2.1 데이터 큐브와 격자 구조

OLAP 시스템에서는 의사 결정자가 다차원적인 관점에서 데이터를 분석할 수 있도록 데이터 큐브를 사용한다. 데이터 큐브의 각 축은 차원을 나타내고, 데이터 큐브의 셀은 각 차원에 의해 결정된다. 예를 들어 *part(p), supplier(s), customer(c)* 라는 3개의 테이블이 있다고 하자. 데이터 큐브 쿼리는 다른

1) 본 연구는 정보통신부 지원 ITRC 프로그램의 지원을 받아 수행되었음

하나의 쿼리로부터 응답이 되어질 수 있다. 두 개의 쿼리 Q_1과 Q_2가 있을 때, Q_1이 Q_2의 결과만을 사용해 응답되어질 수 있다면 $Q_1 \leq Q_2$라고 한다[2]. (Q_1은 Q_2에 의존한다.)



[그림 1] 데이터 큐브 격자 구조

예를 들어서 *part* 차원에 대해서 group-by 되어진 질의는 (*part*) 뿐만 아니라 (*part, supplier*), (*part, customer*) 또는 (*part, supplier, customer*)에 대해서 집계 연산된 결과에 의해서도 응답되어질 수 있다. (이를 (*part*) ≤ (*part, supplier*), (*part*) ≤ (*part, customer*), (*part*) ≤ (*part, supplier, customer*)로 나타낸다.) 차원 애트리뷰트에 의해 나누어진 데이터 큐브의 의존 관계는 격자(lattice) 구조에 의해 표현될 수 있다. [그림 1]은 데이터 큐브의 격자 구조를 나타낸 것이다. 의존 관계는 특정 부분이 실체화됨으로써 실체화된 부분에 대한 질의 처리뿐만 아니라, 의존 관계에 있는 다른 질의 처리에도 영향을 주기 때문에, 최상의 질의 처리 성능을 위해서 어떤 부분을 선택하여 미리 계산하고 저장할 것인가 하는 것은 매우 중요한 문제다. 실체화 뷰를 선택하는 방법으로 [2]에서 제시된 방법은 ROLAP(Relational OLAP) 환경에서 격자의 각 노드에 대한 질의 비용이 해당 뷰의 튜플의 수에 비례한다는 가정하에, 비용 모델로 튜플의 수를 사용한다. 뷰를 선택하는 과정은 각 뷰를 실체화했을 때 얻을 수 있는 이익을 구하여, 그 중 이익이 최대인 뷰를 선택하는 방법이다. 실체화 할 뷰의 개수를 먼저 정하고 주어진 저장 공간을 넘지 않는 한도 내에서 위의 과정을 반복하여 원하는 만큼의 뷰를 선택한다.

2.2 데이터베이스 클러스터

데이터베이스 클러스터는 다음과 같이 크게 3가지 구조로 분류된다. 모든 노드가 독립된 프로세서와 메모리를 가지고 공통의 디스크를 직접 액세스하는 공유 디스크 구조, 모든 노드가 독립된 프로세서를 가지고 공통의 메모리와 디스크를 직접 액세스하는 공유 메모리 구조, 모든 노드가 독립된 시스템으로 구성되고 각 노드의 통신을 위해 고속의 네트워크에 연결되어 있는 비공유 구조로 분류된다[7, 8].

또한 빠른 응답시간과 가용성을 지원하기 위해 분할과 복제를 이용한다. 분할은 테이블을 튜플 단위로 분할하는 수평 분할과 필드를 기준으로 분할하는 수직 분할이 있다. 이러한 분할을 통해 많은 갱신 질의를 동시에 처리할 수 있으므로 빠른 응답시간을 갖는다[9, 10]. 복제는 한 노드의 테이블을 똑같이 복사하여 다른 노드에 두는 것으로 서비스를 하던 노드가 정지되었을 때 복제 본을 가진 노드가 대신 서비스를 함으로서 가용성을 높인다[11].

3. 실체화 뷰 저장 기법

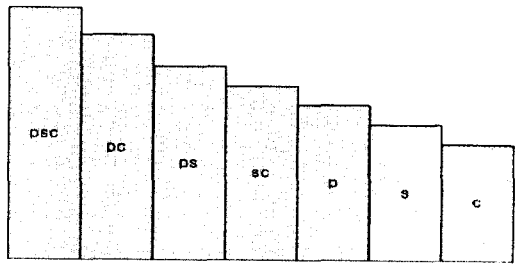
이 장에서는 본 논문에서 제안하는 실체화 뷰를 데이터베이스 클러스터 환경에 맞게 저장하는 기준과 기법에 대해 기술한다.

3.1 데이터베이스 클러스터 기반의 데이터 웨어하우스

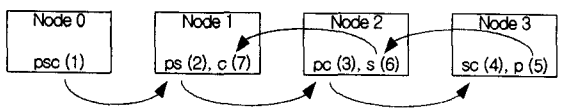
실세계에서 운영되는 데이터 웨어하우스는 수십 개 이상의 차원으로 구성되어 있는 경우가 대부분이다. 차원의 수가 늘어나면 늘어날수록 데이터 큐브는 기하급수적으로 크기가 증가하게 된다. 본 논문에서는 2장에서 살펴본 기존의 실체화 뷰 선택 방법과 함께 데이터베이스 클러스터를 활용해 저장 공간 문제를 해결하고자 한다. 데이터베이스 클러스터는 여러 개의 노드들이 모여 하나의 대용량 시스템으로 동작되기 때문에 대용량 저장 공간의 문제를 해결할 수 있다.

3.2 실체화 뷰의 정렬 순서에 따른 저장 방법

연구 논문 [2]에서 제시된 방법의 비용 모델에 따라 실체화 뷰를 선택하고 선택된 실체화 뷰를 데이터베이스 클러스터에 저장하는 방법을 알아보겠다. 본 논문에서는 각 뷰의 실체화 이익과 뷰 간의 의존성을 고려해서 데이터베이스 클러스터의 각 노드에 실체화 뷰를 분할 저장한다. 2장에서 살펴본 연구 논문 [2]의 방법에 따라 선택된 실체화 뷰를 실체화 이익이 큰 순서대로 정렬을 한다. 2.1절에서 예를 들어 설명한 테이블들의 실체화 이익이 psc > pc > ps > sc > p > s > c라고 하면 [그림 2]와 같이 실체화 뷰들이 정렬되어질 것이고, 데이터베이스 클러스터의 각 노드에 [그림 3]과 같이 화살표 진행 방향을 따라 실체화 뷰들이 저장될 것이다. 괄호안의 숫자는 실체화 뷰들이 각 노드에 저장되는 순서를 의미하며 각 노드의 실체화 이익이 균등하게 유지되도록 하는 방향이다. 예를 들어 sc를 노드 3에 저장한 후 p를 어느 노드에 저장을 할 것인지 결정을 해야 한다. psc의 실체화 이익이 가장 크므로 p를 노드 0에 저장하는 대신 노드 3에 저장을 함으로서 실체화 이익을 균등하게 유지한다.



[그림 2] 실체화 이익에 따른 실체화 뷰의 정렬

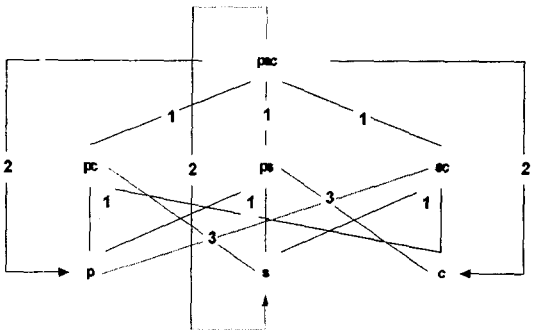


[그림 3] 실체화 이익에 따른 실체화 뷰의 저장

3.3 실체화 뷰의 의존성에 따른 저장 방법

3.2절의 기법과 더불어 보다 더 성능을 향상시키기 위해 실체화 뷰들 사이의 의존성을 고려한다. 3.2절과 같이 각 노드에

실체화 뷰를 저장하는 것은 실체화 뷰들 사이의 의존성을 고려하지 않아 뷰들이 각 노드에 저장된 후 노드의 비효율성 문제점이 생긴다. 실체화 뷰들 간의 의존성은 [그림 4]와 같이 가중치를 두어 계산할 수 있다. 의존성이 있는 뷰는 하나의 뷰에서 다른 하나의 뷰로 유도가 가능하기 때문에 같은 노드에 저장하지 않는 것이 가장 효율적인 저장 방법이다. 하지만 psc, ps, pc, sc, p, s, c 모두 실체화 뷰로 데이터베이스 클러스터에 저장되어야 하는 상황이 실제의 데이터 웨어하우스 환경에서 발생할 수 있으며 데이터베이스 클러스터가 일반적으로 2 ~ 4개의 노드로 구성되기 때문에 실체화 뷰들은 하나의 노드에 2개 이상의 뷰가 함께 저장되어야 한다.



[그림 4] 실체화 뷰들 간의 의존성

이런 상황에서 가중치를 고려하게 되는데 [그림 4]에서 ps, pc, sc는 psc로부터 가중치 1(검정색), p, s, c는 psc로부터 가중치 2(파란색)를 가지고 유도되어지는 것을 볼 수 있다. p, s, c의 가중치가 상대적으로 더 높은 이유는 psc의 크기가 ps, pc, sc보다 더 많기 때문에 psc로부터 직접 p, s, c를 유도하기 위해서는 더 많은 수의 레코드를 처리해야 한다. 즉 p, s, c는 psc보다 ps, pc, sc에서 유도되는 것이 더 효율적임을 의미한다. 그렇기 때문에 데이터베이스 클러스터의 같은 노드에 (psc, ps), (psc, sc), (psc, pc)를 함께 저장하는 것 보다는 (psc, p), (psc, s), (psc, c)를 같은 노드에 저장하는 것이 더 효율적인 저장 방법이다. 또한 [그림 4]에서 알 수 있듯이 sc에서 p, pc에서 s, ps에서 c는 유도되어질 수가 없으며, 가장 높은 가중치인 3(빨간색)으로 계산된다. 서로 유도되어질 수 없는 뷰들이 함께 저장되는 것도 효율적인 저장 방식이다.

마지막으로 또 하나의 다른 데이터 큐브에서 실체화 이익이 가장 큰 뷰를 cst라고 할 때, 서로 다른 데이터 큐브의 실체화 뷰를 저장 시 실체화 이익의 극대화를 위해서 psc 실체화 뷰가 저장된 노드에 cst 실체화 뷰를 동일 노드에 저장하지 않도록 하여야 한다. 이것은 전체 노드가 균등한 실체화 이익 비용을 가지도록 하며 실체화 이익이 가장 높은 하나의 노드로 질의가 집중되는 것을 방지해 효율성을 높이기 위한 방식이다.

4. 결론 및 향후 연구

본 논문에서는 실체화 뷰의 저장 공간의 문제점을 줄이기 위해 데이터베이스 클러스터를 이용한 데이터 웨어하우스 접근 방법을 제안했다. 데이터베이스 클러스터를 이용함으로서 대용량 저장 공간에 대한 문제를 해결하고, 실체화 뷰를 저장할 때 데이터베이스 클러스터 환경에 맞는 기법을 제안했다. 실체화 뷰의 정렬 순서에 따라 실체화 이익이 균등하게 유지되도록 데이터베이스 클러스터의 각 노드에 저장을 하며 실체화 뷰들간의 의존성 또한 고려되어야 하는 요소이다.

향후 연구로는 데이터베이스 클러스터가 분할을 사용해 차원 테이블을 각 노드에 저장함으로 네트워크 조인 비용을 고려한 실체화 뷰의 저장이 연구되어야 하겠다.

5. 관련 논문

[1] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology", ACM SIGMOD Record 26(1), pp. 65-74, 1997.

[2] V. Harinarayan, A. Rajaraman, and J. D. Ullman. "Implementing Data Cubes Efficiently", In Proceedings of ACM SIGMOD, pp. 205-227, 1996.

[3] N. Roussopoulos, "Materialized Views and Data Warehouse", SIGMOD Record, 27(1), pp. 21-26, March 1998.

[4] A. Y. Levy, A. Rajaraman, A. O. Mendelzon, Y. Sagiv, D. Srivastava. "Answering Queries using Views", In Proceedings of ACM PODS, 1995.

[5] D. Srivastava, S. Dar, H. V. Jagadish and A. Y. Levy, "Answering Queries with Aggregation Using Views", In Proceedings of the 22rd International VLDB Conference, pp 116-125, 1996.

[6] H. Gupta, "Selection of Views to Materialize in a data warehouse", In Proceedings of ICDT, pp 98-112, 1997.

[7] 유병섭, 김명근, 김재홍, 배해영, "GMS/Cluster 설계 및 구현", 개방형 지리정보시스템 학회, pp. 225-231, 2003.

[8] David J. DeWitt and Jim Gray, "Parallel Database Systems : The Future of Database Processing or a Passing Fad?", Microsoft, http://research.microsoft.com/~gray/CacmParallel-DB.doc

[9] B. Kemme, "Database Replication for Clusters of Workstations", PhD thesis, Department of Computer Science, ETH Zurich, Switzerland, 2000.

[10] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, DATABASE SYSTEM CONCEPTS Third Edition, McGraw-Hill, pp. 588-593, 1997.

[11] Roel Vandewall, "Database Replication Prototype", Masters thesis, Department of Mathematics and Computer Science, University of Groningen, The Netherlands, 2000.