韓國컴퓨터情報學會 論文誌
第16卷 第8號, 2011. 8.

2011-16-8-1-4

스타 스키마 조인 처리에 대한 세로-지향 데이터베이스 시스템과 가로-지향 데이터베이스 시스템의 성능 비교

오 병 중*, 안 수 민*, 김 경 창*

Performance Comparison of Column-Oriented and Row-Oriented Database Systems for Star Schema Join Processing

Byung-Jung Oh*, Soo-Min Ahn*, Kyung-Chang Kim*

요 약

세로-지향 데이터베이스 시스템은 기존의 가로-지향 데이터베이스 시스템과 달리 데이터를 가로(row) 위주가 아닌 세로(column) 위주로 저장한다. 최근에는 데이터 웨어하우스나 의사 결정 시스템 같은 대용량 데이터를 갖는 읽기 위주의 응용들에서 세로-지향 데이터베이스의 우수성이 관찰되었다. 본 논문에서는 세로-지향 데이터베이스에서의 조인 전략을 구체적으로 분석하고 데이터 웨어하우스 시스템에서 세로-지향 데이터베이스의 우수성을 검증하고자 한다. 두 시스템간의 객관적인 비교를 위해 데이터 웨어하우스 분석 모델인 스타 스키마 벤치마크를 통해 스타 스키마 조인 질의에 대한 성능분석을 실시하고자 한다. 또한 세로-지향 데이터베이스 의 조인 전략으로 조기 실체화(early materialization)와 지연 실체화(late materialization)를 고려하였다. 성능 분석을 통해 스타 스키마 조인 질의 처리에 있어 가로-지향 시스템 보다는 세로-지향 시스템에서 디스크 I/O 비용이 더 효율적인 결과를 확인할 수 있었다. 세로-지향 데이터베이스 시스템 측면에서는 조기 실체화 보다는 지연 실체화 조인 전략이 훨씬 우수한 성능을 보였다.

▸ Keyword : 세로-지향 데이터베이스 시스템, 가로-지향 데이터베이스 시스템, 데이터 웨어하우스, 조기 실체화, 지연 실체화, 스타 스키마 벤치마크, 스타 스키마 조인

Abstract

Unlike in traditional row-oriented database systems, a column-oriented database system stores data in column-oriented and not row-oriented order. Recently, research results revealed the effectiveness of

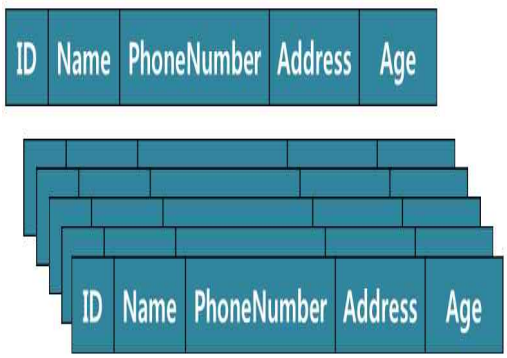
• 제1저자 : 오병중 • 제2저자 : 안수민 • 교신저자 : 김경창
• 투고일 : 2011. 07. 06, 심사일 : 2011. 07. 18, 게재확정일 : 2011. 08. 09.
* 홍익대학교 컴퓨터공학과(Dept. of Computer Engineering, Hong-Ik University)
※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2010-0022714)

column-oriented databases for applications such as data warehouse and decision support systems that access large volumes of data in a read only manner. In this paper, we investigate the join strategies for column-oriented databases and prove the effectiveness of column-oriented databases in data warehouse systems. For unbiased comparison, the two database systems are analyzed using the star schema benchmark and the performance analysis of a star schema join query is carried out. We experimented with well-known join algorithms and considered early materialization and late materialization join strategies for column-oriented databases. The performance results confirm that star schema join queries perform better in terms of disk I/O cost in column-oriented databases than in row-oriented databases. In addition, the late materialization strategy showed more performance gain than the early materialization strategy in column-oriented databases.

▸ Keyword : column-oriented database system, row-oriented database system, data warehouse, early materialization, late materialization, star schema benchmark, star schema join

I. 서 론

최근 몇 년 사이 MonetDB[1,2]와 C-Store[3, 10]를 포함한 몇 개의 세로-지향(column-oriented) 데이터베이스 시스템들이 소개되었다. 세로-지향 데이터베이스 시스템 혹은 세로-저장소(column-store)의 특징은 세로(column) 위주로 데이터가 저장됨으로써 해당 속성의 연결된 값들이 디스크에 연속적으로 저장된다. 이러한 데이터 저장 방식은 기존의 가로-지향(row-oriented) 상용 DBMS들인 Oracle, IBM, DB2, MS SQL Server들이 사용하는 가로-저장소(row-store)와 비교된다. 가로-저장소에서는 릴레이션(relation)을 가로(row) 위주로 저장하는데 같은 튜플(tuple)에 있는 다른 속성 값들이 디스크에 연속적으로 저장된다. 그림 1은 가로-지향 시스템과 세로-지향 시스템에서 데이터가 저장되는 모습을 보여주고 있다.



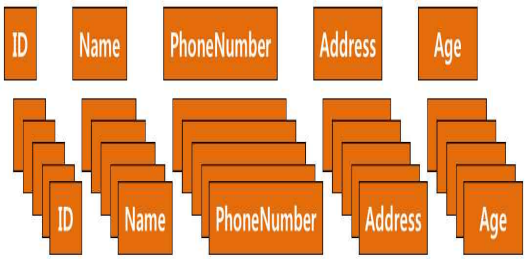


그림 1. 가로-지향 시스템과 세로-지향 시스템의 저장 구조
Fig. 1. Storage Architecture of Row-Oriented and Column-Oriented System

이들 세로-지향 데이터베이스 시스템들의 주장은 데이터 웨어하우스(data warehouse)에서 흔히 볼 수 있는 읽기 위주의 분석용 처리 작업량(workload)에서는 기존의 가로-지향 데이터베이스 시스템과 비교하여 자릿수 이상의 성능 향상을 보인다는 것이다. 사실 세로-지향 데이터베이스 시스템을 기술하는 많은 연구들에서는 기존 가로-지향(row-oriented) 데이터베이스에 비해 성능 결과의 우수성을 포함하고 있다. 이러한 성능 차이는 어쩌면 당연하다고 볼 수 있다. 세로-저장소는 데이터를 가로로 저장하는 기존 데이터베이스 시스템과 달리 세로로 저장하기 때문에 읽기-전용 질의에 대해서 보다 I/O 효율이 높다. 그 이유는 질의가 접근하는 속성들만 디스크나 메모리에서 읽어 들이기 때문이다. 하지만 이들 평가들에서는 논리적인 스키마에 있는 테이블들과 1대1로 매핑 되는 가로-지향 테이블들의 집합으로 구성된 기존 물리적 설계방식을 사용하는 가로-지향 시스템들과 단순 비교한 것이 대부분이다.

비교 결과 세로-지향 접근 방식의 가능성을 명확하게 보여주고 있으나 성능 향상의 근본적인 원인이 세로-지향 DBMS의 내부적인 구조의 근본 방식에 의함인지 아니면 기존의 가로-지향 시스템에서 좀 더 세로-지향적인 물리적 설계를 통해

서도 성능 향상이 가능한지에 대한 정확한 원인을 규명하기가 어려운 현실이다. 따라서 읽기-지향(read-oriented) 혹은 스타 스키마 조인 질의 처리에 있어서 세로-지향 접근 방식과 가로-지향 접근 방식을 비교 분석하여 성능 향상의 근본적인 원인을 규명하는 것이 필요하다. 본 논문의 기여는 다음과 같다. 먼저 객관적인 성능 비교를 위해 데이터 웨어하우스 스타 스키마 벤치마크에 있는 스타 스키마 조인 질의를 가로-지향 및 세로 지향 데이터베이스에서 처리하였다. 또한 종합적인 성능 비교를 위해 스타 스키마 조인 질의 처리에 있어 기존의 모든 조인 알고리즘들을 구현하여 두 시스템에 동시에 적용하였다. 또한 세로-지향 데이터베이스의 특성을 반영한 조인 전략들을 고려하였다.

기존의 가로-지향 시스템에서는 데이터의 입출력과 수정이 쉬운 장점을 가지고 있었다. 하지만 이는 데이터 웨어하우스 조인 질의 처리 관점에서 보았을 때, 조인 질의에 참여하지 않는 데이터까지 읽어와 처리하기 때문에 디스크 I/O 비용 측면에서 비효율적인 모습을 보인다. 이에 반해 세로-지향 시스템에서는 데이터 레코드를 세로로 저장하기 때문에 조인 수행 시, 조인에 참여하는 컬럼만을 읽어와 조인을 수행할 수 있다. 이는 상대적으로 저렴한 비용으로 훨씬 빠른 처리 속도를 구현할 수 있는 장점을 가지고 있다. 따라서 데이터 웨어하우스와 같은 대용량 데이터베이스의 조인 수행에서는 세로-지향 시스템이 성능 개선의 효과를 거둘 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해서 언급하고, 3장에서는 세로-지향 데이터베이스 시스템에서의 조인 처리에 대해 자세히 다룬다. 4장에서는 가로-지향 데이터베이스 시스템과 세로-지향 데이터베이스 시스템간의 객관적인 디스크 I/O 비용 성능비교를 한다. 5장에서는 결론으로 맺는다.

II. 관련 연구

기존의 데이터베이스 시스템인 가로-지향 데이터베이스 시스템에 대해서 많이 소개되었기 때문에 본 논문에서는 언급을 자제하기로 한다. 이 장에서는 세로-지향 데이터베이스 시스템에서의 특징들인 압축 기법과 실체화(materialization) 조인 방법에 대해서 알아본다.

1. 세로-지향 데이터베이스의 압축 기법

세로 지향 시스템에서는 질의를 효율적으로 수행하기 위해 각 컬럼들에 대한 데이터들을 압축하는 기법에 대한 연구가 많이 이루어졌다.

[5]에서는 다양한 데이터 압축 기법들이 소개가 되고 있다. Run-length encoding 기법은 같은 데이터 값들에 대해 singular한 표현으로 데이터 압축을 한다. 예를 들어 4라는 값이 12번째 포지션부터 18번째 포지션까지 중복으로 나타나면 다음과 같이 (4, 12, 7)로 표현한다. 이는 4의 값이 12번째 포지션부터 7개가 반복된다는 의미이다. 이와 유사한 압축 기법으로는 Bit-Vector encoding이 있다. 이는 같은 값들에 대해서 Bit-string으로 표현하는 방법이다. 예로 데이터 값들이 1 1 3 2 2 3 1로 표현이 된다면, 값 1에 대한 Bit-string으로 표현하게 되면 1100001이 된다. 값 2에 대한 Bit-string은 0001100이 되고, 값 3에 대한 Bit-string은 0010010과 같이 표현이 되어 이러한 표현법들은 질의를 수행할 때, 다양한 데이터 값들의 처리 시 효율적인 질의 수행이 가능하다. Dictionary encoding 기법은 주기적으로 나타나는 데이터 패턴을 딕셔너리 엔트리에 코드 값으로 미리 정의하여 이 딕셔너리의 정의 값으로 대체하는 방법이다. 예를 들어 Q1 Q2 Q4 Q1 Q3 Q1 Q1 Q1 Q2 Q4 Q3 Q3 이라는 값들이 컬럼에 존재한다고 가정하여 보자. 그렇다면 딕셔너리에 다음과 같이 Q1은 0, Q2는 1, Q3는 2, Q4는 3으로 미리 정의해 둔다면 이제 이 데이터 레코드들은 013020001322와 같이 단순하게 표현 가능할 것이다. 이는 데이터에 대한 사이즈를 획기적으로 줄여 저장 공간의 효율을 가져오거나 질의 수행의 비용을 낮추는 것이 가능하다.

이러한 압축 기법들이 가지고 있는 장점은 I/O 측면에서 보았을 때, 질의 수행 시 CPU 비용을 낮출 수 있으며, 메모리 공간의 요구가 줄어들어 효율적인 저장이 가능하며, 압축된 데이터들에 직접적인 수행이 가능하게 되어 질의 수행 시간의 단축도 가능한 이점들을 보유하고 있다.

2. 세로-지향 데이터베이스의 조인 전략

기존의 데이터베이스 시스템과 달리 세로-지향 데이터베이스 시스템에서 조인을 하기 위한 전략으로는 조기 실체화 방법과 지연 실체화 방법이 있다[4]. 조기 실체화(early materialization) 조인은 질의 처리 계획의 가장 앞 단계에서 컬럼들을 가로-지향 데이터베이스 형태로 구성하여 질의를 처리하는 조인 전략이다. 먼저 조인에 참여할 릴레이션의 컬럼 A의 값들과 컬럼 B의 값들을 읽어와 가로-지향 시스템의 형태로 구성한다. 다음으로 질의의 조인 조건에 따라 조인을 수행하여 컬럼 A와 B의 값들의 쌍(pair)을 반환받아 최종 결과를 보여주는 과정을 거친다.

조기 실체화 조인은 질의의 출력 데이터가 집계가 되어있

지 않고, 질의 선택도가 높으며, 입력 데이터가 압축이 되어 있지 않으면 유리한 결과를 보이는 장점을 가진다.

지연 실체화(late materialization) 조인은 가능한 질의 처리 계획의 마지막에 가로-지향 시스템의 형태로 구성하는 조인 질의 전략이다. 먼저 조인에 참여할 컬럼 A를 읽어와 조인 조건 절을 적용하여 부합하는 컬럼 값들의 위치를 나타내는 position을 반환 받는다. 마찬가지로 컬럼 B의 값들을 읽어와 조인 조건을 적용하여 부합하는 값들의 위치인 position을 반환받는다. 다음으로 이 두 position을 AND 연산하여 최종 f_position을 반환받는다. 이 최종 f_position의 의미는 조인 조건을 만족하는 값을 가진 각 컬럼 값의 위치를 나타낸다. 이 f_position을 각 컬럼에 적용하여 부합하는 컬럼 값을 읽어와 최종 단계에서 가로-지향 시스템의 형태로 구성하여 결과를 보여준다.

지연 실체화 조인은 위치 데이터에 직접 연산을 하고, 관련 있는 튜플만 생성하고, 압축된 데이터에 직접 연산을 하고 값의 반복 속도가 빠르기 때문에 조기 실체화 조인에 비해서 성능이 월등이 높다고 할 수 있다. 하지만 역으로 컬럼을 여러 번 접근하기 때문에 성능의 저하가 올 수 있다.

III. 세로-지향 시스템에서의 조인 전략

이번 절에서는 기존의 가로-지향 시스템에서 사용되는 조인 기법들을 적용하여 세로-지향 시스템에서 조인 전략들이 어떻게 수행되는지 알아본다. 본 논문에서 비교 분석하고자 하는 세로-지향 시스템에서의 조인 전략들을 소개하며, 예제 스타 스키마 질의를 통해 조인 질의가 어떠한 과정으로 처리되는지 설명한다. 사용된 스타 스키마 질의는 표준 데이터 웨어하우스 벤치마크인 스타 스키마 벤치마크(Star Schema Benchmark)[7]에 있는 질의 1.1이며 다음과 같다.

```
Select sum(Lo.extendedprice*Lo.discount) as revenue
From Lineorder Lo, Date D
Where Lo.orderdate=D.datekey
      And D.year=1993
      And 1≤Lo.discount≤3
      And Lo.quantity<25;
```

위의 질의에서 Where절의 각 selection 조건들을 만족하는 튜플들만을 반환받기 위한 Filter Factor(FF)는 D.year=1993의 경우 1/7, 1≤Lo.discount≤3은 3/11, Lo.quantity<25은 47/100이 적용된다. FF는 전체 튜플 중에서 해당 selection 조건을 만족하는 튜플들의 분수를 의미한다.

세로-지향 데이터베이스 시스템에서의 조인 전략은 조기 실체화 방법과 지연 실체화 방법이 함께 사용된다[4]. Join 단계에서는 기존의 가로-지향 시스템에서 사용되는 조인 알고리즘인 Nested-loop 조인, Sort-merge 조인, Hash 조인이며, Nested-loop 조인은 다시 Simple nested loop 조인(SNL 조인), Index nested loop 조인(INL 조인), Page-oriented nested loop 조인(PNL 조인)으로 구분되어 수행한다[8].

1. 세로-지향 시스템의 조기 실체화 조인 절차

조기 실체화 조인은 질의 처리 계획의 가장 앞 단계에서 컬럼들을 가로-지향 시스템의 형태로 먼저 구성하여 질의를 처리하는 조인 전략이다. 조기 실체화 조인의 과정은 그림 2와 같다. 이 전략은 세로-지향 데이터베이스의 장점인 압축 기법을 전혀 사용할 수 없다는 단점이 있다.

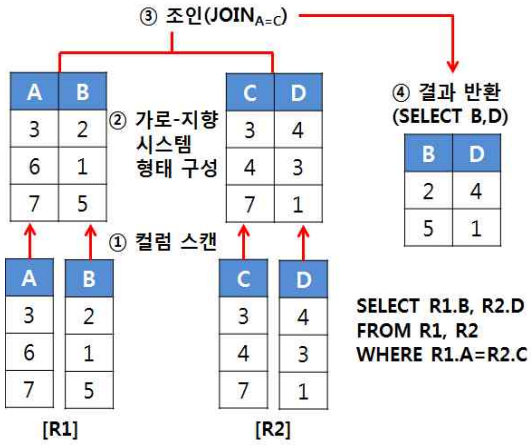


그림 2. 조기 실체화 조인 과정
Fig. 2. Early materialization join process

조기 실체화 조인의 실행은 세 단계로 나뉘어 진행되는데 selection 단계, join 단계, 그리고 result 단계로 이루어진다.

Selection 단계

Lineorder 테이블과 Date 테이블에서 질의에 사용되는 컬럼들을 메모리로 읽어온다. Lineorder 테이블에서는 extendprice, discount, orderdate, quantity 컬럼, Date 테이블에서는 datekey, year 컬럼이 해당된다. 이를 가로-

지향 시스템의 테이블 형식으로 컬럼들을 연결(stitch) 한다. 다음으로 Where절의 selection 컬럼에 Filter factor(FF)를 적용하여 조인에 참여할 튜플만을 추출한다. 여기에서 SNL 조인은 Inner 테이블의 페이지 단위 조인을 위해 Date에서 추출된 튜플은 페이지로 구성하여 디스크에 임시 저장한다. 또한 PNL 조인, Sort merge 조인, Hash 조인의 경우에는 페이지 단위 조인을 위해 Lineorder와 Date에서 추출한 튜플을 페이지로 재구성하여 디스크에 임시 저장한다. Selection 단계의 절차들을 종합적으로 표현하면 그림 3과 같다.

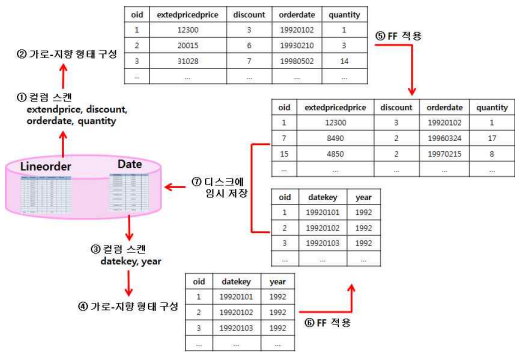


그림 3. 조기 실체화 조인의 Selection 단계
Fig. 3 Early materialization join Selection phase

Join 단계

Selection 단계에서 재구성하여 임시 저장한 각 테이블을 이용하여 Lo.orderdate=D.datekey의 조건에 따라 조인을 수행한다. 이 때, Lineorder가 Outer 테이블이 되며 Date가 Inner 테이블이 된다. SNL 조인의 경우 메모리상의 Lineorder와 임시 저장된 Date를 한 페이지씩 읽어오며 조인을 수행한다. PNL 조인, Sort-merge 조인, Hash 조인의 경우는 디스크에 임시 저장된 Lineorder와 Date를 메모리상으로 읽어오며 조인을 수행한다.

Result 단계

조인의 결과로 반환받은 튜플들에서 Select절의 sum(Lo.extendprice*Lo.discount) as revenue의 연산을 위해 Lo.extendprice, Lo.discount 컬럼만을 Projection 한다. Projection된 두 컬럼의 튜플마다 extend 컬럼의 값과 discount 컬럼의 값끼리 extend*discount의 연산을 수행하고 각 결과들을 모두 더해 revenue의 이름으로 질의의 최종 결과를 보여준다.

2. 세로-지향 시스템의 지연 실체화 조인 절차

지연 실체화 조인은 가능한 질의 처리 계획의 마지막 단계에 가로-지향 시스템의 형태로 구성하는 조인 질의 전략이다. 조기 실체화 기법과 달리 압축 기법 등을 적용할 수 있다. 지연 실체화 조인의 과정은 그림 4와 같다. 조인 컬럼(A 와 C) 값들을 비교하여 포지션 리스트를 반환하여 이 리스트를 이용하여 Select 절에 명시된 컬럼(B와 D)의 해당 값을 반환받아 가로-지향 형태의 레코드로 연결하여 구성한다.

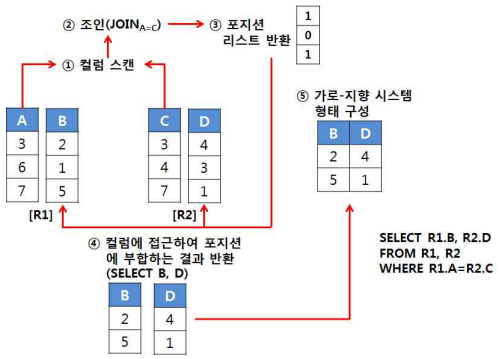


그림 4. 지연 실체화 조인 과정
Fig. 4. Late materialization join process

지연 실체화 조인의 실행도 조기 실체화 기법과 마찬가지로 세 단계인 selection 단계, join 단계 그리고 result 단계로 나뉘어 진행된다.

Selection 단계

Date 테이블의 year 컬럼을 메모리로 읽어와 selection 조건 컬럼의 FF를 적용하여 만족하는 튜플의 oid 리스트를 반환한다. 마찬가지로 Lineorder 테이블에서 각각 discount, quantity 컬럼을 메모리로 읽어와 selection 조건의 FF를 적용하여 만족하는 각각 튜플의 oid 리스트를 반환한다. 여기에서 별도로 discount 컬럼의 FF가 적용된 oid 리스트에 해당하는 튜플들을 디스크에 임시 저장해둔다. 이는 Result 단계에서 Select절의 연산을 위해 discount 컬럼을 메모리로 읽어오는 단계를 수행하는데, FF가 적용된 튜플들만이 최종 연산에 관계가 있기 때문에 컬럼 전체를 읽어오는 것에 비하여 훨씬 효율적이기 때문이다. 반환받은 두 컬럼 discount, quantity의 oid 리스트를 AND-WISE 연산을 수행하여 공통된 oid들의 리스트를 생성한다. 이 리스트의 의미는 1≤Lo.discount≤3, Lo.quantity<25의 조건을 모두

만족하는 튜플들의 oid 리스트이다. 그림 5에서 위의 절차들을 그림으로 표현하고 있다.

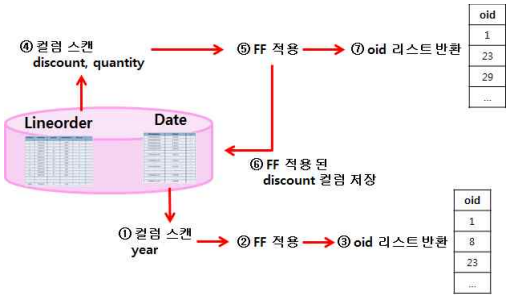


그림 5. 지연 실체화 조인의 Selection 단계
Fig. 5. Late materialization join Selection phase

Join 단계

Outer로 설정될 Lineorder 테이블의 orderdate 컬럼을 메모리로 읽어온다. 이 컬럼의 oid와 Selection 단계에서 discount, quantity의 AND-WISE 연산을 통해 생성한 oid 리스트의 oid와 비교하여 같은 oid의 튜플들만을 추출한다. 이렇게 반환받은 튜플들이 조인에 참여하게 된다. 이 경우에 orderdate 컬럼의 스캔 비용의 best case는 전체 페이지 수의 (3/11 * 47/100) 만큼이 될 것이며, worst case는 풀 스캔이 될 것이다. 마찬가지로 Inner로 설정 될 Date 테이블의 datekey 컬럼을 메모리로 읽어와 Selection 단계의 Date에서 생성된 oid 리스트와 비교해 oid가 같은 튜플들만을 추출하며, 이 튜플들이 조인에 참여하게 된다. 여기에서 페이지로 재구성하여 디스크에 임시 저장하여 조인을 수행하게 될 기법들은 SNL 조인의 경우는 datekey의 재구성 된 페이지를 저장하게 되며, PNL 조인, Sort-merge 조인, Hash 조인의 경우에는 재구성 된 orderdate, datekey 모두를 임시 저장하게 된다. Lo.orderdate=D.datekey 조건에 따라 Outer로 설정된 orderdate 컬럼과 Inner로 설정된 datekey 컬럼과의 조인을 수행한다. SNL 조인의 경우 메모리 상에 있던 orderdate와 디스크에 임시 저장한 datekey간의 조인을 수행하며, PNL 조인, Sort-merge 조인, Hash 조인의 경우 디스크에 임시 저장한 orderdate, datekey를 스캔하여 조인을 수행한다. 마지막으로 조건에 부합하는 Outer의 oid 리스트만을 조인의 결과로 반환한다. 전체적인 Join 단계의 절차는 그림 6에서 볼 수 있다.

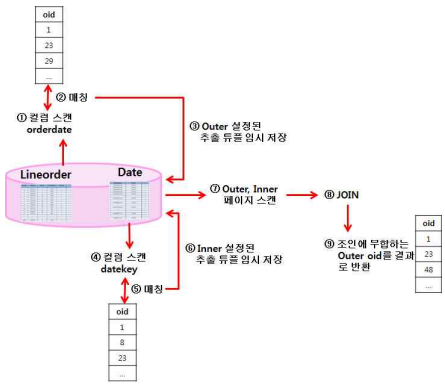


그림 6. 지연 실체화 조인의 Join 단계
Fig. 6. Late materialization join Join phase

Result 단계

Select절의 sum(Lo.extendprice*Lo.discount) as revenue의 연산을 위해 Lineorder의 extendprice 컬럼과 Selection 단계에서 임시 저장한 discount 컬럼을 메모리로 읽어온다. 읽어온 extendprice 컬럼들의 oid와 조인의 결과로 반환된 튜플들의 oid를 비교하여 같은 oid의 튜플들만을 연산에 참여시킨다. 마찬가지로 메모리로 읽어온 discount 컬럼의 oid와 조인 결과 oid 리스트와 비교하여 같은 oid의 튜플들만을 연산에 참여시킨다. 연산에 참여할 두 컬럼의 같은 oid의 튜플들끼리 extendprice*discount의 연산을 수행하고 각 결과들을 모두 더해 revenue의 이름으로 질의의 최종 결과를 보여준다.

Ⅳ. 성능 분석

본 절에서는 기존의 가로-지향 데이터베이스 시스템과 세로-지향 데이터베이스 시스템의 조인 처리 성능을 알아보기 위하여 여러 조인 알고리즘을 고려한 조인 질의 처리를 수행하고 디스크 I/O 비용을 기반으로 성능 비교 분석을 실시한다.

1. 성능 분석 환경

가로-지향 시스템에서는 기존의 스타 스키마 벤치마크의 사실 테이블과 차원 테이블의 구조를 그대로 사용하였고, 세로-지향 시스템에서는 MonetDB 아키텍쳐[6]를 사용하여 가로-지향 시스템의 릴레이션을 세로-지향 시스템으로 매핑하였다. MonetDB의 구조는 그림 7과 같으며 각 테이블은 세로로 컬럼별로 저장되는데 각 컬럼은 위치를 나타내는 oid

컬럼이 추가된다.

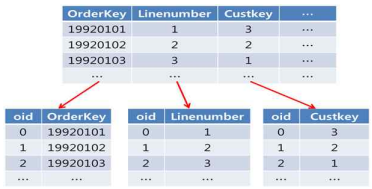


그림 7. MonetDB 아키텍처를 이용한 세로-지향 시스템
Fig. 7. Column-oriented system using MonetDB architecture

성능 분석에 사용된 스타 스키마 벤치마크[7, 10]의 테이블 구조는 그림 8과 같다. 사실 테이블은 Lineorder 테이블이며 차원 테이블들은 Customer, Supplier, Part, Date 테이블로 구성되었다. Lineorder 테이블의 Custkey, Partk ey, Suppkey, OrderDate는 왜래 키로서 각각 Custo mer, Part, Supplier, 및 Date 테이블을 참조한다.

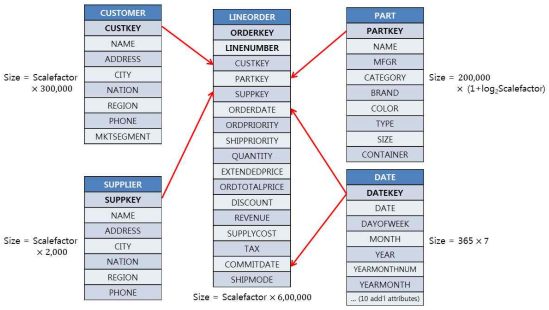


그림 8. 스타 스키마 벤치마크 테이블 구조
Fig. 8. Star Schema Benchmark table layout

분석에 사용될 스타 스키마 질의는 스타 스키마 벤치마크 질의 1.1이며 질의의 구체적인 표현과 selection을 위한 Where절의 Filter factor(FF)는 다음의 표 1과 같다. 이 스타 스키마 질의를 처리하기 위해서는 Lineorder 사실 테이블과 Date 차원 테이블의 조인 처리가 필요하다.

표 1. 스타 스키마 벤치마크 질의 1.1 및 Filter Factor
Table 1. Star Schema Benchmark Query 1.1 and Filter Factor

질의	Select sum(Lo.extendedprice*Lo.discount) as revenue From Lineorder Lo, Date D Where Lo.orderdate=D.datekey And D.year=1993 —— ① And 1≤Lo.discount≤3 — ② And Lo.quantity<25; —— ③
Filter Factor	① FF = 1/7 ② FF = 3/11 ③ FF = 47/100

질의에 사용될 Lineorder 테이블과 Date 테이블의 구조는 다음의 표 2와 같다. 이 테이블 구조에는 테이블의 크기, 튜플의 수, 페이지의 크기 등 비교에 필요한 테이블에 대한 수치 값이 포함된다. 비교의 단순화를 위해 Scale Factor는 0.01을 적용하였다.

표 2. 가로-지향 시스템과 세로-지향 시스템의 테이블 구조
Table 2. Table architecture of Row-oriented and Column-oriented system

가로-지향 시스템의 구조		세로-지향 시스템의 구조	
Line order	82bytes/tuple 60000tuples 49tuples/page 1225 pages	Line order	8bytes/tuple 60000tuples 512tuples/page 118pages/column
Date	90bytes/tuple 2555tuples 45tuples/page 57pages	Date	8bytes/tuple 2555tuples 512tuples/page 5pages/column

Selection조건을 만족하는 값을 검색하기 위해 B-tree 인덱스[9]를 사용한다면 직접적으로 조인에 참여할 컬럼의 튜플만을 검색해서 읽어올 수 있으므로 더욱 효율적인 성능 수행이 가능할 것이다.

이를 위한 가정으로 B-tree 인덱스는 Lineorder 릴레이션의 discount 컬럼, Date 릴레이션의 year 컬럼으로 선정하였다. 각 컬럼의 Leaf node는 튜플의 value와 value의 저장된 물리적 주소 RID(Record Identifier)를 가지고 있다. 스타 스키마 벤치마크에서는 Lo.discount 컬럼은 0부터 10까지의 값을 가지며, D.year 컬럼은 1992부터 1998까지의 값을 가지므로 D.yaer 컬럼의 Leaf node는 7개, Lo.discount 컬럼은 11개의 Leaf node를 가지게 된다. 이렇게 B-tree 인덱스를 사용하여 특정 value를 검색하는 데에는 (Height+1)의 비용이 소요되며, 해당 value를 메모리로 읽어오는 비용으로는 (튜플 수 * FF)를 수행한 다음 이를 페이지화하여 구성한 페이지 수만큼의 비용이 들게 될 것이므로 컬럼 전체의 튜플을 읽어오는 것보다 효율적인 컬럼 스캔을 수행할 수 있게 된다. 더불어 이 Hash 인덱스는 앞의 장에서 설명한 Nested loop 조인, Sort merge 조인 및 Hash 조인 모두에서 Selection을 위한 컬럼을 읽어오는 단계에 디스크의 I/O 비용을 줄이는 방법으로 사용될 수 있다.

앞선 절에서 설명한 각 조인 전략들에 따라 질의를 처리하였고, B-tree 인덱스의 유무에 따라서도 어떻게 성능의 차이가 발생하였는가에 대해 알아보는 과정을 거쳤다. 분석 결과는 디스크 I/O 비용만을 산정하였으며, 메모리상에서 수행되

는 연산에 대한 비용과 CPU 비용은 고려 대상에서 제외하였다. 조인 전략들에 대한 질의 처리 단계는 앞선 장에서 설명한 조인 단계를 그대로 수행한 결과에 따른 디스크 I/O 비용 소모를 반영하였다.

2. 성능 비교 분석

Nested loop 조인에서는 SNL 조인, INL 조인, PNL 조인을 수행하였으며 이에 따른 성능 분석의 결과는 다음의 그림 9, 10, 11와 같다.

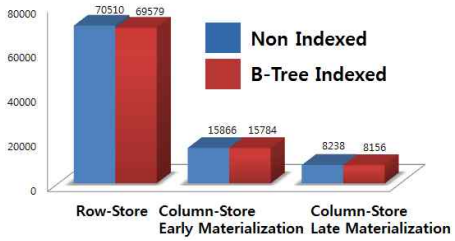


그림 9. Simple nested loop 조인 비교 분석 결과
Fig. 9. Simple nested loop join comparison result

각각의 조인 전략들에서 질의 처리 분석 결과를 보면 전체적으로 가로-지향 데이터베이스보다 세로-지향 데이터베이스에서의 질의 처리 성능이 SNL 조인의 경우 조기 실체화 조인에서 4배 이상, 지연 실체화 조인에서는 8배 이상의 성능 개선 효과가 나타났으며, selection에 해당하는 컬럼에 B-tree 인덱스를 적용한 질의 처리가 B-tree 인덱스를 적용하지 않은 질의 처리보다 소폭의 성능 개선 효과가 있었음을 확인할 수 있다. 또한 INL 조인에서는 가로-지향 데이터베이스에 비해 세로-지향 데이터베이스의 조기 실체화와 지연 실체화 조인에서 각각 소폭의 성능 개선 효과를 거두었다. 마지막으로 PNL 조인에서는 세로-지향 데이터베이스의 조기 실체화와 지연 실체화 조인에서 가로-지향 데이터베이스 시스템에 비해 약 5배의 성능 개선 효과를 거두었다.

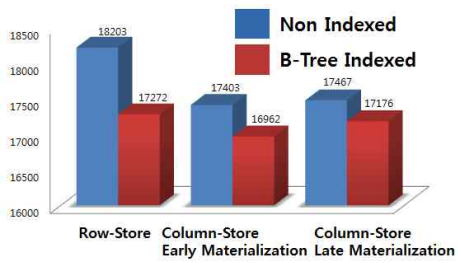


그림 10. Index nested loop 조인 비교 분석 결과
Fig. 10. Index nested loop join comparison result

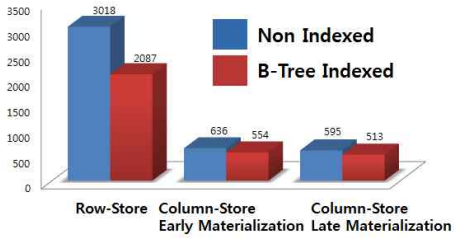


그림 11. Page-oriented nested loop 조인 비교 분석 결과
Fig. 11. Page-oriented nested loop join comparison result

Sort merge 조인의 성능 분석 결과는 다음의 그림 12와 같다.

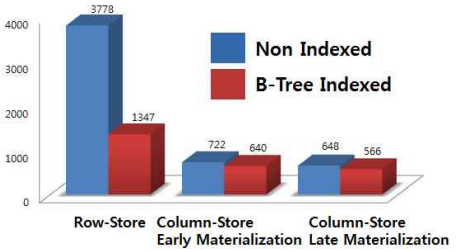


그림 12. Sort merge 조인 비교 분석 결과
Fig. 12. Sort merge join comparison analysis result

Sort merge 조인 역시 가로-지향 시스템 보다는 세로-지향 시스템의 조기 실체화 조인에서 약 5배, 지연 실체화 조인에서 약 6배의 성능 개선 효과를 거두었으며, 그리고 B-tree 인덱스를 사용한 기법이 사용하지 않은 기법보다 세로-지향 데이터베이스에서는 소폭의 성능 개선 효과를, 가로-지향 데이터베이스에서는 약 3배의 성능 개선 효과를 보임을 확인할 수 있다.

Hash 조인의 성능 분석 결과는 다음의 그림 13과 같다.

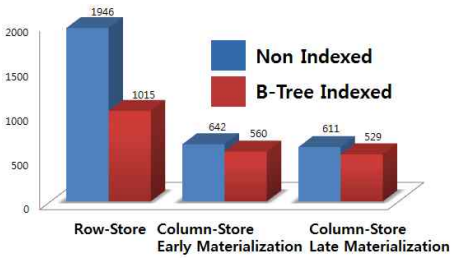


그림 13. Hash 조인 비교 분석 결과
Fig. 13. Hash join comparison result

마찬가지로 Hash 조인에서도 세로-지향 시스템의 조기

실체화 조인에서 3배, 지연 실체화 조인에서 약 3배의 성능 개선 효과를 보여주었다. B-tree 인덱스를 사용한 기법이 가로-지향 시스템과 B-tree 인덱스를 사용하지 않은 기법들에 비해 소폭의 성능 개선 효과를 나타내었다.

마지막으로, 성능 분석결과 세로-지향 데이터베이스에서는 조기 실체화 조인 전략 보다는 지연 실체화 조인 전략이 보다 우수한 성능을 보였다.

V. 결 론

최근의 연구들에서는 분석용 작업량을 처리하는 스타 스키마 질의 처리를 위해 기존의 가로-지향 데이터베이스 시스템에 비해 세로-지향 데이터베이스 시스템의 우수성을 언급하였다. 본 논문에서는 이를 구체적으로 입증하기 위해 스타 스키마 벤치마크에 있는 스타 스키마 조인 질의를 이용하여 다양한 조인 기법들에 대한 가로-지향 시스템과 세로-지향 시스템에서의 디스크 I/O 비용 측면에서 질의 처리 성능 분석을 하였다. 성능 분석의 목적은 어떤 데이터베이스 시스템이 스타 스키마 조인 질의 처리에 보다 효율적인가를 알아보는 것이었다.

분석 결과에서 볼 수 있듯이 가로-지향 데이터베이스보다 세로-지향 데이터베이스에서 훨씬 낮은 디스크 I/O 비용의 소모됨을 알 수 있었으며, selection 조건을 위한 B-tree 인덱스를 사용한 기법이 우수한 성능을 나타내었다. 또한 세로-지향 시스템 측면에서 보았을 때는 조기 실체화 기법보다 지연 실체화 조인 기법이 더 적은 디스크 I/O 비용을 필요로 함을 확인할 수 있었다. 이를 통해 대용량 데이터를 읽어와 조인 질의를 수행하는 것이 주된 작업 환경인 데이터 웨어하우스 등에서 조인 질의 처리 시에는 가로-지향 데이터베이스 시스템보다 세로-지향 데이터베이스 시스템이 디스크 I/O 비용 면에서 좋은 성능을 보였다.

본 연구의 추후 과제는 세로-지향 데이터베이스에서 기존의 조인 알고리즘과 비교하여 조인 질의 성능을 높일 수 있는 새로운 조인 알고리즘을 제안하는 것이다. 이 새로운 조인 알고리즘은 본 연구의 결과에서 그 성능의 우수성이 입증된 지연 실체화 조인 전략으로서 기존의 알고리즘보다 성능이 우수한 전략이 될 것이다.

참고문헌

[1] P. Boncz, M. Zukowski, and N. Nes. “Mone tDB/X100: Hyper-pipelining query execution”, Proc. of Intl′ Conf. on Innovative Data System Research (CIDR), 2005.

[2] P.A. Boncz and M.L. Kersten. “MIL primitives for querying a fragmented world”, VLDB Journal, 8(2): 101-119, 1999.

[3] M. Stonebraker, et. al. “C-Store: A Column-Oriented DBMS”, Proc. of VLDB, 553-564, 2005.

[4] Daniel J. Abadi, Daniel S. Myers, David J. DeWitt, Samuel R. Madden. “Materialization Strategies in a Column-Oriented DBMS”. IEEE. 2007.

[5] Daniel J. Abadi, Peter A. Boncz, and Stavros Hariz opoulos. “Column-oriented Database Systems”. VLDB 2009 Tutorial.

[6] P. A. Boncz. “Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications”. Ph.d. thesis, Universiteit van Amsterdam, May 2002.

[7] Patrick E. O'Neil, Elizabeth J. O'Neil, and Xuedong Chen. “The Star Schema Benchmark (SSB)”. Revision 3, June 5, 2009.

[8] Raghu Ramakrishnan, Johannes Gehrke, “Database Managem ent Systems 2nd Edition”, McGrawHill, pp. 333-348, 2000.

[9] Abraham Silberschatz, Henry F. Korth, S. Sud arshan, “Da tabase System Concepts Fifth Edition”, McGrawHill, pp. 481-502, 2006.

[10] Alan Halverson, Jennifer L. Beckmann, Jeffrey F. Naughton, David J. DeWittP, “A Comparison of C-Store and Row-Store in a Common Framework”, Proc. of the 32nd VLDB Conference, 2006.

[11] Daniel J. Abadi. “Query Execution in Colum n-Orie nted Database Systems”, Massachusetts Institute of Technology, pp. 85-93, 2008

저 자 소 개

오 병 중

2008 : 홍익대학교 컴퓨터정보통신공학과 공학사.
현 재 : 홍익대학교 컴퓨터공학과 재학.
관심분야 : 객체지향DB, 세로지향DB
Email : lightclassic@nate.com

안 수 민

2010 : 홍익대학교 컴퓨터공학과 공학사.
현 재 : 홍익대학교컴퓨터공학과 재학.
관심분야 : 객체지향DB, 세로지향DB
Email : happydaygirl@nate.com

김 경 창

1978 : 홍익대학교 전자계산학과 공학사.
1980 : 한국과학기술원 전산학과 공학석사.
1990 : University of Texas at Austin 전산학과 공학박사.
현 재 : 홍익대학교 컴퓨터공학과 교수
관심분야 : 객체지향DB, 주기억DB, 센서DB, 세로지향DB
Email : kckim@hongik.ac.kr