객체 데이터베이스에서 스타 조인의 빠른처리를 위한 비트맵 색인 기법과 그의 선정 문제

조 완 섭* · 정 태 성** · 이 현 철** · 장 혜 경* · 안 명 상*

Bit-map Indexes and Their Selection Problem for Efficient Processing of Star Joins in Object Databases

W. S. Cho* · T. S. Jung** · H. C. Lee** · H. G. Jahng* · M. S. Ahn*

Abstract

We propose an indexing technique and an index selection algorithm for optimal OLAP query processing in object database systems. Although there are many research results on the relational database systems for OLAP query processing, few researches have been done on the object database systems. Since OLAP queries represent complex business logic on a huge data warehouse, object database systems supporting the OLAP queries should have higher performance. Proposed bitmap index structure is an extension of conventional bitmap indexes for adapting object databases and provides higher performance with lower space overhead. We also propose a linear time solution of the index selection problem that will be used in the OLAP query optimization process.

KeyWords : Database, Data Warehouse, OLAP, Index

* 충북대학교 경영정보학과
** 충북대학교 정보산업공학과
※ 이 논문은 한국과학재단의 해외 Post-Doc. 연수지원비에 의하여 연구되었음.

1. 개 요

*데이터 웨어하우스(data warehouse)*는 기업 최고 경영자의 의사 결정을 과학적인 근거를 갖고서 지원하기 위하여 주제 중심적이고, 통합적이며, 비휘발성이고, 이력성을 가지도록 수집한 자료이다[Colliat, 1996 ; Deshpande, 1997 ; Do et al., 1998 ; Inmon, 1997]. 최근 들어 데이터베이스 구축이 보편화되고, 데이터베이스로부터 축적되는 자료가 급증함에 따라서 이들 자료를 가공하고 분석하여 조직의 의사 결정에 활용할 수 있도록 지원하는 데이터 웨어하우스의 구축이 활발하게 이루어지고 있다[Colliat, 1996 ; Do et al., 1998 ; Inmon, 1997]. 예를 들어, 백화점이나 인터넷 쇼핑몰의 경영자는 지난 1년간의 고객 구매 정보를 시간대별로, 고객의 지역별로, 혹은 판매된 제품의 종류별로 구분하여 분석하는 작업이 필요하며, 이를 위하여 데이터 웨어하우스를 구축하는 것이 필요하다.

*온라인 분석 처리(On-Line Analytical Processing : OLAP)*는 최종 사용자가 대규모 기업 데이터 웨어하우스에 접근하여 대화식으로 정보를 분석하는 과정이다[Colliat, 1996, Inmon, 1997]. OLAP의 구현 방식은 다차원 데이터베이스(multi-dimensional database)를 이용하여 자료를 저장하고 분석하는 MOLAP(multidimensional OLAP : MOLAP)과 관계 데이터베이스를 이용하여 자료를 저장하고 분석하는 ROLAP(relational OLAP : ROLAP)으로 나눌 수 있다[Colliat, 1996, Do et al., 1998, Inmon, 1997].

데이터 웨어하우스는 기존의 데이터베이스 보다 대용량의 데이터를 저장하고 있으며, OLAP 질의도 복잡한 분석 작업을 표현해야 하므로 기존 데이터베이스 응용의 질의보다 복잡한 것이 특징이다. 데이터 웨어하우스에서는 현재 데이터뿐 아니라 과거 수년간의 자료를 대상으로 데이터 변화의 추세를 분석하는 경우가 많으므로 대개의 경우 기가~테라바이트 이상의 자료 크기를 가정한다. 또한, 의사 결정의 수행을 위하여 제시되는 분석용 질의도 대부분의 경우 다수의 조인과 그루핑 연산 및 집계 함수를 포함하므로 처리 시간이 오래 걸린다. <그림 1>은 TPC-D [TPC-D, 2002]에서 성능 평가 기준으로 제시된 대표적인 OLAP 질의 중 하나를 보여준다. 이 질의는 6개의 테이블과 6개의 조인 및 그루핑을 포함하며, 테이블 LINEITEM에는 60억개의 튜플을 가정하고 있다.

특히, OLAP 질의는 대부분 대규모 사실 테이블과 다수의 차원 테이블이 조인되는 형태이므로 조인 비용이 특히 높다. 실제로 스타 조인의 처리 성능을 높이기 위한 다양한 방법들이 연구되고 있다[O'neil & Graefe, 1995 ; O'neil & Quass, 1997 ; Valduriez, 1987 ; Wu & Buchmann, 1998 ; Zhao et al., 1998].

```
SELECT  N_NAME, SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT)(DECIMAL(15, 2)))(NAMED REVENUE)
FROM    CUSTOMER, ORDERTBL, LINEITEM, SUPPLIER, NATION, REGION
WHERE   C_CUSTKEY = O_CUSTKEY  AND  O_ORDERKEY = L_ORDERKEY
AND     L_SUPPKEY = S_SUPPKEY  AND  C_NATIONKEY = S_NATIONKEY
AND     S_NATIONKEY = N_NATIONKEY  AND N_REGIONKEY = R_REGIONKEY
AND     R_NAME = 'ASIA' AND O_ORDERDATE >= '1994-01-01'
AND     O_ORDERDATE <  ADD_MONTHS('1994-01-01', 12)
GROUP BY  N_NAME
ORDER BY  REVENUE DESC ;
```

<그림 1> TPC-D에서 제시된 OLAP 질의의 예

본 논문에서는 객체 데이터베이스 관리 시스템(ODBMS)를 이용하여 데이터 웨어하우스를 저장하고, OLAP 질의를 처리하는 문제를 다룬다. 본 논문에서는 이러한 경우의 OLAP을 ***OOLAP (Object OLAP)***이라고 부른다. 지금까지 주로 관계형 DBMS나 다차원 DBMS를 이용하는 ROLAP과 MOLAP을 가정하고 성능 향상을 위한 새로운 색인 구조와 질의 처리 및 질의 최적화 기법을 연구하여 왔으나, 최근들어 객체 DBMS의 보급이 확대됨에 따라 OOLAP에 관한 연구가 새롭게 시작되고 있다[Binh et al., 2000 ; Czejdo et al., 2001 ; Gopalkrishnan, 1999 ; Gu et al., 2000 ; Huynh et al., 2000 ; Pedersen et al., 2000 ; Ravat & Teste, 2000 ; Zhao et al., 1998]. 특히, 국내에서는 객체 DBMS인 UniSQL[3] [UniSQL, 1991]이나 Odysseus[4] 등의 제품에서 OOLAP의 지원에 관한 필요성이 높아지고 있는 실정이며, Oracle 등의 기존 DBMS에도 객체 개념을 추가하여 객체-관계 DBMS로 발전하고 있으므로 향후 OOLAP에 관한 관심이 증가할 것으로 예상된다.

본 연구에서는 OOLAP 질의에서 자주 사용되며 비용이 큰 스타 조인 (star joins)의 처리 성능을 높이는 색인 기법을 제시하고, 이를 이용한 스타 조인 처리 알고리즘을 제시한다. 그리고, 질의 최적화 기법의 일부로서 스타 조인의 처리 전략을 수립할 때 최소 비용을 보장하는 색인 선정 기법을 제안한다. 제안된 색인 기법은 기존 ROLAP에서 제안된 비트맵 조인 색인을 ODBMS 환경에 적합하도록 수정한 것이며, 질의 처리 기법과 질의 최적화 기법에서는 제안된 비트맵 조인 색인을 적극적으로 이용하는데 초점을 맞춘다. 비트맵 조인 색인이 구축되면 질의 처리기 관점에서는 하나 이상의 색인(들)을 사용하는 새로운 질의 처리 전략(query evaluation plan)들이 생성될 수 있다. 논문에서는 이러한 가능한 처리 전략들 중에서 최소 비용의 처리 전략을 선형 시간(linear time)에 선택하는 알고리즘을 제시한다.

본 연구의 결과는 향후 객체 DBMS를 이용하여 데이터 웨어하우스를 구축하고 OLAP 질의를 처리하는데 있어서 유용하게 사용될 수 있을 것이다. 특히, 기존의 객체 DBMS를 이용하여 대규모 스타 조인을 처리하는 경우에 질의 처리 성능에 문제가 있는 것으로 확인되었으며, 이는 제안된 색인 구조를 활용함으로써 해결될 수 있을 것이다. 논문에서는 색인 기법의 제안과 함께 그 색인을 이용한 질의 처리 기법도 제안함으로써 실제로 제안된 인덱스가 DBMS에서 이용될 수 있도록 하였다. 또한, 새로운 색인 기법의 제안과 함께 질의 최적화에서 제안된 색인의 사용 여부를 선형 시간에 결정하는 색인 선정 기법을 고안함으로써 제안된 색인의 실용성을 제고하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 데이터 웨어하우스와 관련된 연구들을 살펴본다. 특히, ODBMS를 이용한 데이터 웨어하우스 구축에서 스키마 구조와 질의어를 살펴본다. 제 3장에서는 색인 기법과 이를 이용한 질의 처리기법을 소개한다. 제 4장에서는 최소 비용의 색인 집합을 선형 시간에 결정하는 알고리즘을 소개한다. 제 5장에서는 결론을 맺는다.

2. 관련 연구

이 장에서는 객체 DBMS를 사용하여 데이터 웨어하우스를 구축하는 방안을 간단히 소개한 후, OLAP 질의와 질의 처리 성능을 높이기 위하여 제안된 기존의 색인 기법을 살펴본다.

2.1 ODBMS를 이용한 스타 스키마와 OLAP 질의

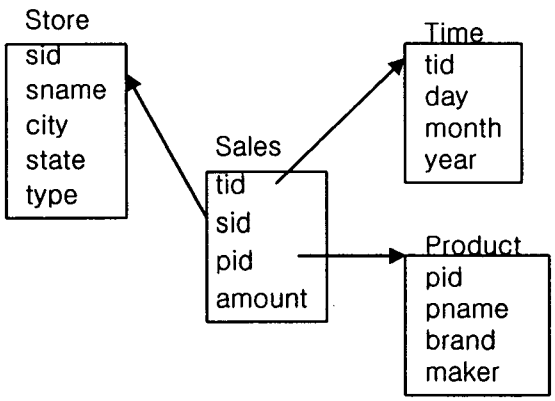
ODBMS를 이용하여 데이터 웨어하우스의 스타 스키마를 구축하는 간단한 방법은 관계형 스타

3) (주) KT Data와 (주) 한국컴퓨터통신에서 상용화하여 판매하고 있는 객체-관계 DBMS임.

4) 한국과학기술원에서 제작하여 보급하고 있는 객체-관계 DBMS임.

스키마에서 외래키-주키 관계를 속성-도메인 관계로 변경하면 된다. 예를들어, <그림 2>에서 Sales는 사실 클래스이고, Time, Store, Product는 차원 클래스이다. 사실 클래스와 차원 클래스 사이의 연결선은 속성-도메인 관계를 나타낸다. 이러한 스타 스키마에 대한 OLAP 분석 질의에서는 속성-도메인 관계를 따라 기술되는 경로식(path expression)을 사용하여 스타 조인이 이루어진다. 또한, 각 클래스는 서브 클래스(subclass)를 가질 수 있고, 각 차원 클래스의 애트리뷰트도 사용자 정의 클래스를 도메인으로 가질 수 있다. 예를 들어, Product 클래스의 속성 maker 도메인으로 String 대신에 또 다른 클래스인 Company를 지정할 수 있다.

또 다른 방법으로 차원 클래스의 속성에서 사실 클래스를 참조하도록 하거나, 차원 클래스와 사실 클래스 사이에 양방향 참조를 설정할 수도 있으나, 전자의 경우 집합값 속성을 가져서 검색하기에 불편하며, 후자의 경우 양방향 참조로 인하여 공간 효율성이 저하되므로 여기서는 이러한 방식을 고려하지 않는다.



<그림 2> 객체형 스타 스키마의 구현 방식

<그림 2>의 객체형 스타 스키마에 대하여 객체 질의어를 사용하여 다양한 분석용 질의를 할 수 있다. 질의 2는 <그림 2>의 스타 스키마에 대하여 작성한 전형적인 OLAP 질의어이다.

(질의 2)

```
SELECT  pid.maker, pid.pname, sum(amount)
FROM    Sales f
WHERE   f.tid.month = 5 AND f.sid.city = MA
GROUP BY f.pid.maker, f.pid.pname ;
```

질의 2에서는 경로식(path expression)을 이용하여 스타 조인들을 표현하고 있다. 즉, 질의에서는 Sales-Store-Time-Product 사이의 조인을 포함하고 있으며, 특히 Sales 클래스의 객체수가 많으므로 스타 조인의 처리에 비용이 커질 수 있게 된다. 질의의 복잡성 측면에서 보면, 질의 2의 FROM 절에는 사실 클래스만 명시하며, WHERE 절에서 속성-도메인 관계의 경로식을 사용하여 각 차원 클래스와 조인을 수행한다. GROUP BY 절과 SELECT 절에도 길이가 2이상인 경로식을 사용하여 각각 그루핑과 프로젝션을 간단하게 명시하고 있다.

2.2 색인과 질의 처리 기법

일반적으로 데이터 웨어하우스에서는 사실 클래스의 객체수가 수백만개 이상으로 많다고 가정하므로 스타 조인의 비용이 높다. 그래서 스타 조인의 비용을 줄이는 방안으로 새로운 색인이 제안되고 있으며, 대표적으로 비트맵 색인과 비트맵 조인 색인이 그 예이다[Jurgens & Lenz, 1999 ; O'neil & Graefe, 1995 ; Wu & Buchmann, 1998]. 비트맵 색인은 임의의 속성 a에 포함된 각 값(value)에 대하여 그 값을 가지는 사실 테이블의 객체에 대한 위치 정보를 비트로 표시한 것이다. 비트맵 색인은 일반적으로 트리 형태의 색인보다 공간 효율성이 뛰어나며, 질의 처리 성능도 좋으므로 데이터 웨어하우스에서 널리 사용된다.

비트맵 색인을 이용하면 스타 조인을 저렴한 비용으로 처리할 수 있다. 예를 들어, <그림 2>의 데이터베이스에 대하여 Product.maker = "HP"인 제품의 Sales.amount 합을 구하려면 Sa-

Store

sid	sname	city	state
s1	aa	LA	CA
s2	bb	MA	FL
s3	cc	DA	TX
s4	dd	MA	FL
s5	ee	SF	CA
s6	ff	SF	CA
s7	gg	LA	CA
s8	hh	MA	FL
...	...	...	...

Sales

sid	tid	pid	amount
s1	t1	p1	1
s2	t2	p1	11
s1	t3	p2	22
s2	t4	p2	33
s1	t4	p3	44
s3	t4	p2	55
s2	t5	p1	66
s5	t6	p8	77
s1	t6	p6	88
s4	t7	p5	99
s6	t9	p5	12
...	...	...	...

Time

tid	year	month	day
t1	1999	1	1
t2	1999	1	2
t3	1999	1	3
...	...	...	...

s1	1 0 1 0 1 0 0 0 1 ...
s2	0 0 0 1 0 1 0 0 0 ...
...	...

<그림 3>bid Bit-mapindex

les와 Product의 스타 조인이 발생한다. 이 경우 Product 클래스에서 name = "HP"인 객체를 구한 후 (편의상 p2, p3라고 하자), 비트맵 색인의 p2 비트맵과 p3 비트맵으로부터 원하는 객체를 바로 액세스 할 수 있게 된다.

데이터 웨어하우스에서 널리 사용되고 있는 색인으로는 비트맵 색인(bitmap index)과 비트맵 조인 색인(bitmap join index)[Chan & Ioannidis, 1998 ; O'neil & Graefe, 1995 ; O'neil & Quass, 1997 ; Sarawahi, 1997]이 있으며, 이 밖에도 프로젝션 색인(projection index)[On97, Sa97], 그룹-셋 색인(group-set index)[On97, Sa97], 다이나믹 비트맵(dynamic bitmaps)[O'neil & Quass, 1997 ; Sarawahi, 1997], 범위-기반 비트맵 색인(range-based bitmap indexing) [O'neil & Quass, 1997 ; Sarawahi, 1997], 인코딩된 비트맵 색인(encoded bitmap index)[Johnson, 1999; Wu & Buchmann, 1998] 등이 있다.

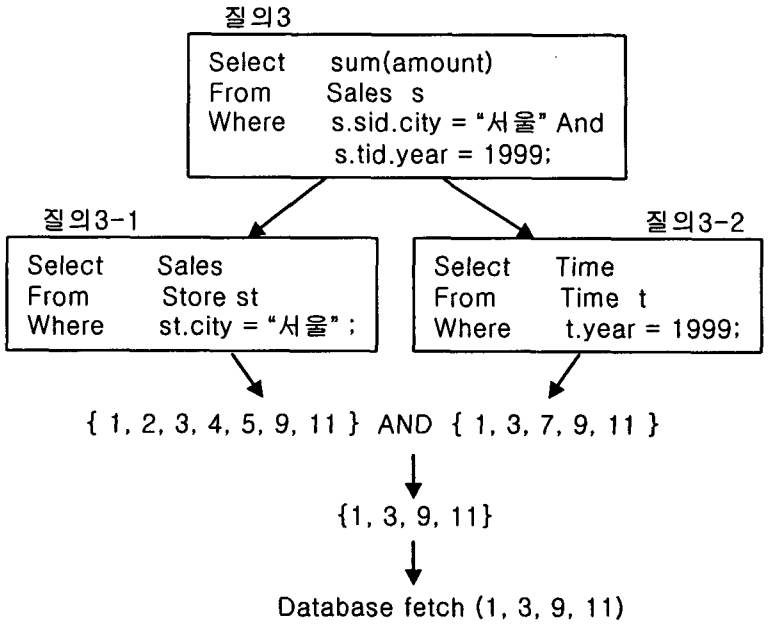
그러나, 지금까지 객체 DBMS를 사용하여 OLAP 질의를 처리하는 환경에서 비트맵 색인을 사용하는 연구는 이루어지지 않고 있다. 관계 DBMS에서의 연구 결과를 이용할 수도 있으나 이 경우에도 객체 DBMS에 적합하도록 수정하는 것이 필요하다. 본 논문에서는 이러한 점을 감안하여 기존의 비트맵 인덱스를 객체 데이터베이스에 적합하도록 수정하여 제안하고, 이를 이용한 새로운 질의 처리 기법과 인덱스 선정 기법을 제안한다.

3. OOLAP에서의 색인 기법과 질의 처리 알고리즘

여기서는 ODBMS를 이용한 환경에서 OLAP 질의를 처리하는데 적합한 비트맵 색인 기법을 제안하고, 이를 이용한 질의 처리 방법과 색인 생성 방안 등을 기술한다.

3.1 비트맵 색인 기법

제안된 색인 기법은 ODBMS을 이용하여 데이터 웨어하우스를 구축하고 분석용 질의를 처리하는 OOLAP 환경에 적합한 비트맵 조인 색인이다. <그림 3>은 <그림 2>의 스타 스키마에 대하여 클래스에 대한 객체와 비트맵 색인의 모습을 보여주고 있다. Sales 클래스에서 1, 3, 5, 9번째 객체가 Store의 s1 객체를 참조하므로 비트맵의 bid=1인 비트 슬라이스에서 이들 위치에 1을 기록하였다. 그림에서는 sid에 대한 비트맵 색인만 표시하였지만 tid, pid 등의 속성에 대해서도 동일한 형태의 색인을 생성한다. 또한, <그

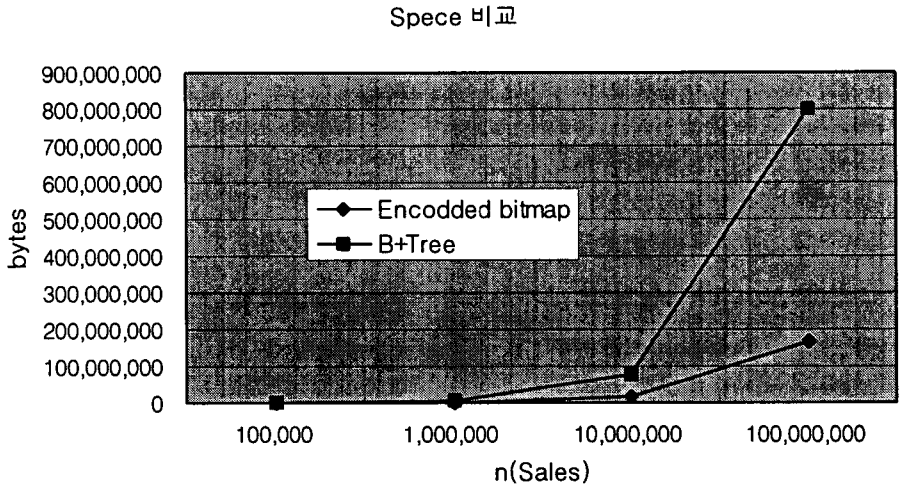


<그림 4> 비트맵 색인을 이용한 스타 조인의 처리 과정

림 3>에서는 설명을 쉽게 하기 위하여 비트맵 색인을 보여주지만 실제로는 공간 효율성이 뛰어난 인코딩된 비트맵 색인을 사용한다. 인코딩된 비트맵 색인은 간단한 조작으로 구현할 수 있으므로 논문에서는 이해도를 증진하기 위하여 비트맵 색인을 사용하여 설명한다. <그림 3>에서는 sid 속성에 대한 색인을 보여주고 있지만 실제로는 sname이나 city 혹은 state 등에 대한 색인도 동일한 구조와 동일한 방법으로 생성할 수 있다. 이러한 색인은 객체 데이터베이스에서의 경로 색인[Bertino, 1989]와 동일하며, 기존의 객체 데이터베이스에서는 그 유용성은 인정하고 있지만[Bertino, 1989] 인덱스 갱신 문제 때문에 실제로는 구현되지 않고 있다. 그러나, 데이터 웨어하우스의 객체는 분석 도중에 변경되지 않으므로 이러한 경로 색인 구조가 갱신 오버헤드 없이 유용할 것으로 판단된다. 경로 색인을 사용하면 속성에 포함된 값의 개수가 적어지므로 (<그림 3>에서 sid의 경우 값의 개수가 많지만 city나 state의 경우는 값의 개수가 적음) 더욱 효율적인 비트맵 인덱스의 구성이 가능하게 된다. 일반적으로 비트맵 색인은 키 값의 개수가 작을수록 B^+Tree 보다 효율적인 것으로 알려져 있기 때문이다[Jurgens & Lenz, 1999]. 또한, state나 city 값으로부터 관련된 Sales 객체를 색인으로부터 바로 찾을수 있으므로 질의 조건을 처리하는데 더욱 효과적이다.

3.2 비트맵 색인을 이용한 질의처리

사용자가 입력한 OLAP 질의에 대하여 비트맵 색인을 이용하기 위해서는 적절한 질의 변환이 요구된다. <그림 4>에서 이러한 질의 변환을 통한 질의 처리 과정을 보여주고 있다. 사용자가 <그림 2>의 스키마를 보고 질의 3을 입력하면 OLAP 질의 처리기에서 질의 3-1과 질의 3-2로 변환하고, 각각 비트맵 색인을 이용하여 해당 객체의 위치를 구한 다음, 두 집합의 교집합을 구함으로써 두 조건을 동시에 만족하는 객체의 위치를 구해 낸다. 객체의 위치가 구해지면 사실



<그림 5> 비트맵 색인과 B⁺tree 색인의 크기 비교

테이블로부터 데이터베이스 API(application program interface)에서 제공되는 함수를 이용하여 해당 객체를 액세스한 후 amount 합계를 구한다.

3.3 비트맵 색인의 생성

여기서는 사실 클래스에 새로운 객체를 로딩할 때 비트맵 색인을 생성하는 방법을 소개한다. 차원 클래스의 객체들은 고정되어 분석 도중에 거의 변경되지 않지만 사실 테이블에는 주기적으로 분석의 대상이 되는 데이터를 교체할 필요가 있으므로 로딩시에 필요한 색인을 생성해야 한다.

다음은 Sales.sid 속성에 대하여 비트맵 색인을 구성하고, 객체가 입력될 때 색인을 구성하는 알고리즘을 보여주고 있다. 알고리즘에서 보는 바와 같이 객체를 입력할 때 비트맵 색인을 생성하므로 색인 생성에서 별도의 페이지 출입은 필요하지 않다.

색인 생성 알고리즘 : CreateIndex( )

(1) 차원 클래스 주키(sid)를 기준으로 다음과 같은 **비트맵**을 생성한다. 비트맵은 주키의 각 값에 대하여 비트들의 1차원 배열인 bit-slice를 구성한 것이며, 처음에는 모두 0으로 초기화된다. 비트 슬라이스의 크기는 사실 클래스의 객체수와 동일하다.

(2) 사실 클래스에 n(≥ 0) 번째 객체 [s_i, t_k, p_l, amount]가 입력되면 s_i에 해당하는 bit-slice를 찾아서 n번째 위치를 1로 세팅한다.

(3) 모든 객체가 입력될 때까지 단계 (2)를 반복한다.

한편, 인코딩된 비트맵 색인의 경우에는 매핑 테이블과 비트맵으로 구성된다[Wu & Buchmann, 1998]. Store.sid 속성에 대한 매핑 테이블은 sid 값들을 2 진수로 대응시켜 생성한 것이며, sid 속성에 dv(sid) 개의 값이 포함되어 있으면 이를 대응시키기 위하여 $\log_2$dv(sid) 개의 비트가 생성된다. 인코딩된 비트맵에는 $\log_2$dv(sid) 개의 비트를 가진 n(sales) 개의 bit-slice가 생성되며, i번째 객체가 속성 sid의 s_i와 연관된다면 i번째 bit-slice에는 매핑 테이블에서 s_i에 대응하는 이진수를 가지게 된다. 인코딩된 비트맵

색인은 참고문헌[Wu & Buchmann, 1998]에 잘 나타나 있으며, 간단하므로 여기서는 생략한다.

비트맵 색인에 대하여 발생하는 연산은 일반 데이터베이스 색인에 대한 연산보다 간단하다. 데이터 웨어하우스의 경우 주기적으로 사실 클래스의 객체를 로딩하여 (필요하면 기존 사실 테이블의 객체를 삭제함) 차원 클래스를 기준으로 분석하기 위한 것이므로 인덱스의 변경보다는 로딩시 인덱스를 재구성하는 것이 바람직하다.

3.4 비트맵 색인의 저장 공간 분석

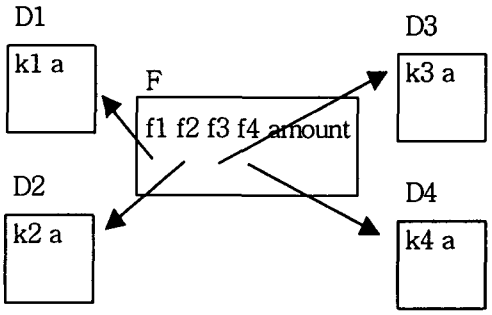
비트맵 색인을 활용하면 기존의 B$^+$tree를 이용한 색인 구조보다 저장 공간의 측면에서 큰 잇점이 있다. 그러나 색인이 구축되는 속성에 값의 종류가 너무 많으면 비트맵 색인의 공간은 급격하게 증가하므로 인코딩하여 사용하도록 한다. 여기서는 인코딩된 비트맵 색인을 기준으로 B$^+$Tree와 비교하여 저장 공간을 분석한다.

먼저 분석에 필요한 몇가지 인자를 실제 데이터 웨어하우스의 경우와 유사하게 가정한다. Store 테이블의 속성 sid의 값의 개수를 10,000개로 가정하고, 사실 테이블의 객체수 n(Sales)를 100,000개에서 100,000,000개까지 증가시키면서 인코딩된 비트맵 색인과 B$^+$tree의 저장 공간을 비교하였다. 단, OID 길이는 UniSQL[UniSQL, 1991]에서와 같이 8바이트로 가정한다. <그림 5>에서 보는 바와 같이 사실 테이블의 객체수가 증가하여 10,000,000개를 넘어 서면서 B$^+$ tree 색인의 크기가 급격히 증가한다. 따라서 사실 테이블의 객체수가 대규모임을 가정하는 데이터 웨어하우스에서는 대부분의 경우 B$^+$Tree 보다는 인코딩된 비트맵 색인을 사용하고 있다. 한편, 인코딩되지 않은 비트맵의 경우에는 sid 속성의 값의 개수가 50개 정도를 넘어서면서 급격히 크기가 증가하므로 여기서는 인코딩된 비트맵 색인을 가정하였다.

4. 색인 선정 기법

스타 조인에 대하여 다수개의 비트맵 색인이 구축된 경우에 질의 최적화에서는 이들중에서 몇 개를 사용할 것인가, 그리고 어느 것을 사용할 것인가 에 관한 정책을 가져야 한다. 색인을 몇 개나 사용하는가는 질의 처리 전략을 수립하는데 중요한 요소이기 때문이다. 일반적으로 n개의 색인이 존재하는 경우에 주어진 질의를 처리하기 위하여 다음의 처리전략을 고려해야 한다.

- 색인을 사용하지 않는 전략
- 색인 하나를 사용하는 전략 (이 경우, 어느 색인을 사용하는지 결정해야 함)
- ...
- 색인을 두 개, 세 개, ..., n-1개 사용하는 전략
- 색인을 모두 사용하는 전략



<그림 6> 스타 스키마와 질의어

(질의 4)

```
Select   Avg (t.amount)
From     F  t
Where    t.f1.a1 = 10    // pd1
And      t.f2.a2 = 20    // pd2
And      t.f3.a3 = 30    // pd3
And      t.f4.a4 = 40;   // pd4
```

즉, n개의 색인이 존재하면 색인의 사용과 관련하여 2^n 가지의 처리 전략이 존재하며, 질의

<표 1> 비용 함수에 사용되는 기호

기 호	설 명
cost(I_1, I_2, …)	색인 I_1, I_2, …를 이용하여 질의를 처리하는 비용
b(b, bf, k)	b개의 블록으로 구성된 파일에서 (각 블록은 bf 개의 튜플을 가짐) k 개의 레코드를 주소 순서로 액세스할 때 예상되는 블록 출입회수 [Ya77]
p(f)	화일 f의 블록 수
s(pdi)	조건 pdi의 선택률 ; 조건 pdi와 색인 I_i가 일치하는 경우 s(I_i)라고 표시하는 경우도 있음

최적화에서는 이들 중에서 최소 처리 비용을 가지는 하나를 선택해야 한다. 가장 쉬운 결정 방법으로는 2^n가지의 처리 전략 각각에 대하여 처리 비용을 계산한 다음에, 최소 비용의 처리 전략을 선택하는 것이지만, 이 경우 너무 많은 처리 전략에 대하여 비용을 계산해야 한다는 문제가 있다. 즉, 비트맵 색인이 10개 이라면 색인 사용과 관련된 처리 전략이 2^{10} = 1024가지이고, 이들 각각에 대하여 비용을 계산해야 한다(즉, 계산 비용이 색인 개수에 대한 지수 함수로 증가함). 여기서는 주어진 질의를 처리하는데 몇 개의 색인을 사용하는 것이 좋은가를 선형 시간에 결정하는 기법을 소개한다. 이 문제를 풀기 위해서는 색인을 사용함으로써 얻는 이익과 손해를 잘 이해해야 한다. 비트맵 조인 색인을 사용하면 색인으로부터 조건을 만족하는 사실 클래스의 객체들만 선택하여 읽을 수 있으므로 액세스되는 객체수를 줄이는 이익이 있다. 반면에 비트맵 자체를 읽어야 하는 것은 손해가 된다. 따라서, 색인 I를 추가로 사용하여 액세스되는 객체수를 감소시킴으로써 줄어드는 페이지 출입회수를 P1이라 하고, I에 대한 비트맵을 읽음으로써 증가하는 페이지 출입 회수를 P2라고 할 때 P1-P2 > 0인 경우에만 색인 I를 사용해야 한다.

다음에는 비용 수식을 유도하기 위하여 사용하는 스타 스키마와 질의 4를 살펴본다. <그림 6>에서 사실 테이블 F의 각 속성 fi(i = 1, 2, 3, 4)에는 인코딩된 비트맵 색인 I_i개 생성되었다고 가정한다.

비용 수식에서는 <표 1>의 기호를 사용한다.

다음은 <그림 6>의 스키마와 질의 4에 대하여 색인 I_1, I_2, I_3, I_4를 사용할 때 질의 처리 비용을 모델링한 것이다.

- cost(I_1) = p(I_1) + b(p(F), bf(F), n(F) * s(pd1))
- cost(I_1, I_2) = p(I_1) + p(I_2) + b(p(F), bf(F), n(F) * s(pd1) * **s(pd2)**)
- cost(I_1, I_2, I_3) = p(I_1) + p(I_2) + **p(I_3)** + b(p(F), bf(F), n(F) * **s(pd1)** * **s(pd2)** * **s(pd3)**)
- cost(I_1, I_2, I_3, I_4) = p(I_1) + p(I_2) + **p(I_3)** + **p(I_4)** + b(p(F), bf(F), n(F) * **s(pd1)** * **s(pd2)** * **s(pd3)** * **s(pd4)**)

비용 cost(I_1, I_2)는 색인 I_1과 I_2를 읽는 비용인 p(I_1) + p(I_2)과 두 색인으로부터 얻은 객체들의 OID 집합(n(F) * s(pd1) * s(pd2)개로 추정됨)을 사용하여 실제로 사실 테이블 F에서 객체를 액세스하는 비용인 b(p(F), bf(F), n(F) * s(pd1) * s(pd2))로 구성된다. 나머지 비용 수식도 유사하게 설명된다.

예를들어, 이 수식으로부터 색인 집합 (I_1, I_2)를 사용할 때 보다 색인 집합 (I_1, I_2, I_3)을 사용하는 경우에 이익은 클래스 F로부터 더 적은 수의 객체를 액세스 한다는 점이며, 이로 인하여 액세스 필요가 없어지는 페이지 수는 b(p(F), bf(F), n(F) * s(pd1) * s(pd2)) - b(p(F), bf(F), n(F)

* s(pd1) * s(pd2) * s(pd3))로 표시된다. 그리고, 색인 I_3을 사용함으로써 발생하는 손해는 추가로 비트맵 색인을 하나 더 읽는다는 점이며, 이와 관련된 페이지 수는 $p(I_1) + p(I_2) + p(I_3) - [p(I_1) + p(I_2)] = p(I_3)$가 된다.

본 논문에서는 이러한 비트맵 조인 색인의 사용과 관련된 이익과 손해를 고려하여 이익이 있는 범위에서만 색인을 사용하도록 한다. 다음은 색인을 결정하는 최적화 알고리즘을 자연어로 기술한 것이다.

알고리즘 : IndexSelection()

입력 : 데이터베이스 통계치, 질의어, 비트맵 색인 집합(I_1, I_2, I_3, I_4 …), 질의어

출력 : 최소비용의 색인집합과 질의처리 전략

(1) Compute sequential access cost : C_0

(2) Find the best index as follows :

For each index I_i in the index set

$cost(I_i) = p(I_i) + b(p(F), bf(F), k)$,

where $k = n(F) * s(I_i)$

$C_1 = Min\ \{cost(I_i)\},\ I = 1, 2, \cdots$

The best index B_1 is the index that has the minimum $cost(I_i)$

(3) Find the i-th (i ≥ 2) best index as follows (let the cost as $cost(B_1, \cdots B_{i-1}, I_i)$) :

For each index I_i in the remaining index set

$cost(B_1, \cdots B_{i-1}, I_i) = \sum_{j=1,2,\cdots,i-1} p(B_j) + p(I_i) + b(p(F), bf(F), k)$,

where, $k = n(F) \times s(B_1, \cdots B_{i-1}, I_i)$

$Ci = Min\ \{cost(B_1, \cdots B_{i-1}, I_i)\},\ I = 1, 2, \cdots$

The i-th best index B_i is the index that has the minimum $cost(B_1, \cdots B_{i-1}, I_i)$

(4) Repeat step (3) until ($C_i < C_{i-1}$)

단계 (1)에서는 사실 클래스 F를 순차접근 방식으로 읽는 비용인 $p(F)$이다. 단계 (2)에서는 하나의 색인을 사용하는 경우에 필요한 질의 처리 비용이다. 이 비용은 색인 자체를 읽는 비용 $p(I_i)$와 색인으로부터 얻은 조건을 만족하는 객체를 읽는 비용인 함수 $b(\cdots)$로 구성된다. 이 때, 조건을 만족하는 객체수는 $n(F) * s(I_i)$로 추정된다. 하나의 색인을 사용하는 경우에 최소의 질의 처리 비용을 C_1이라 표시하고, 그 때의 색인을 B_1로 표시하였다. 단계 (3)에서는 일단 색인 B_1이 선택된 상황에서 추가로 사용될 색인을 구하는 과정이다. 나머지 색인 각각에 대하여 B_1과 함께 사용하는 경우 질의 처리 비용을 계산하고, 그들 중에서 최소 비용을 보장하는 색인 집합 (B_1과 또 다른 하나)을 찾는다. 이 때, 두 번째 색인으로 선정된 색인을 B_2라고 표시하고, 질의 처리 비용을 C_2라고 표시한다. 단계 (4)에서는 색인을 추가로 사용함으로써 비용이 줄어드는가를 검사하고, 줄어든다면 동일한 방식으로 단계 3을 반복 수행하여 세 번째 이후의 색인들을 선정한다.

(정리 1) 알고리즘 IndexSelection()은 동적 프로그래밍 기법의 특성인 Principle of optimality 성질[Ho96]을 만족한다.

(증명) 단계 (1)에서 구한 첫 번째 색인이 I_1이라고 하자. 이 경우 I_1을 사용하는 경우의 비용이 다른 색인을 사용하는 경우의 비용보다 낮다. 즉, $cost(I_1) < cost(I_3)$가 된다. 다음으로, 두 번째 단계에서 I_2가 선택되었다고 가정하자. 그런데, $cost(I_3, I_2) < cost(I_1, I_2)$이라면 (즉, 첫 번째 단계에서 최적이라고 구한 색인 I_1이 두 번째 단계에서 선정되지 않는다면), 이는 다음과 같이 모순을 발생시킨다.

$cost(I_1) = p(I_1) + b(p(F), bf(F), n(F) * s(I_1))$

$< cost(I_3) = p(I_3) + b(p(F), bf(F), n(F) * s(I_3))$--(1)

임에도 불구하고, $cost(I_1, I_2) = p(I_1) + p(I_2) + b(p(F), bf(F), n(F) * s(I_1) * s(I_2)) > cost(I_3, I_2) = p(I_3) + p(I_2) + b(p(F), bf(F), n(F) * s(I_3) * s(I_2))$--(2)

이라면 모순이 발생한다. 그 이유는 수식 (2)에서 부등호 양변에 $p(I_2)$를 제거하여 수식 (3)을

만들면, cost(I_1, I_2) = p(I_1) + b(p(F), bf(F), n(F) * s(I_1) * **s(I_2)**) > cost(I_3, I_2) = p(I_3) + b(p(F), bf(F), n(F) * s(I_3)***s(I_2)**) --(3)이 된다. 수식 (3)은 수식 (1)에서 n(F)에 **s(I_2)**를 추가로 곱한 것 외에는 동일하다. 그런데, 함수 b()는 마지막 인자 k의 크기에 따라 단조 증가하는 함수이므로 n(F)에 **s(I_2)**를 추가로 곱하여도 수식 (1)의 부등호 방향은 변하지 않는다. 즉, 수식 (1)과 수식 (3)은 서로 양립할 수 없는 모순이다.

다음에서 알고리즘-2의 동작 과정을 예를 들어 상세히 살펴본다.

(예-2) 인덱스 선정 알고리즘의 동작과정

<그림 6>에서 4개의 술어 pd1, pd2, pd3, pd4의 선택률과 통계치를 다음과 같이 가정하자.

(1) 술어들의 선택률

s(pd1) = 1/5, s(pd2) = 1/20, s(pd3) = 1/20, s(pd4) = 1/10

(2) 통계치

n(F) = 1,000,000 objects // F의 객체수
p(F) = 62,500 pages // F의 페이지 수
bf(F) = 16 objects // F의 페이지당 객체수
p(I_1) = 500 pages // 색인 I_1이 차지하는 페이지 개수
p(I_2) = 1,000 pages ; p(I_3) = 2,000 pages ; p(I_4) = 4,000 pages

알고리즘 IndexSelection( )에서는 다음 단계를 거쳐서 최소 비용을 제공하는 색인 집합을 구한다.

단계0) 순차접근 비용

C0 = p(F) = **62,500 pages**

단계1) 1st 색인의 선정

cost(I_1) = p(I_1) + b(p(F), bf(F), k = n(F) * s(p1))
= 500 + b(62500, 16, 1000000 * 1/5)
= 500 + 60,741
= 61,241 pages

cost(I_2) = 1,000 + b(62500, 16, 1000000 * 1/20)
= 1,000 + 34,993
= **35,992 pages**

cost(I_3) = 2,000 + b(62500, 16, 1000000 * 1/20)
= 2,000 + 43,992
= 45,992 pages

cost(I_4) = 4,000 + b(62500, 16, 1000000 * 1/10)
= 4,000 + 50,918
= 54,918 pages

따라서 하나의 색인을 사용하는 경우 최소 비용은 I_2를 사용하는 경우이며, 그 때의 질의 처리 비용 C1 = **35,992 pages** 이다.

단계 2) 두 번째 색인 선정

cost(I_2, I_1) = 1,000 + 500 + b(62500, 16, 1000000 * 1/20 * 1/5) = 1,500 + 9,284
= 10,784 pages

cost(I_2, I_3) = 1,000 + 2,000 + b(62500, 16, 1000000 * 1/20 * 1/20) = 3,000 + 2,454
= **5,454 pages**

cost(I_2, I_4) = 1,000 + 4,000 + b(62500, 16, 1000000 * 1/20 * 1/10) = 5,000 + 4,816
= 9,816 pages

따라서 I_3을 추가로 사용하는 전략이 최소비용을 제공하며, C2 = 5,454 pages이다. 이 값은 I_2 하나만을 사용하는 전략의 비용(35,992 pages)보다 저렴하므로 두 번째 단계에서 색인 집합을 (I_2, I_4)로 선정한다. 색인 추가 사용으로 인하여 더 이상의 비용 감소가 없을때까지 이러한 과정을 반복 수행하여 주어진 질의를 처리하는데 가장 최적의 색인 집합을 구한다.

5. 결 론

본 연구에서는 객체 DBMS를 사용하여 데이터 웨어하우스를 구축하고 OLAP 질의를 처리할 때 유용한 비트맵 색인 기법을 제안하고, 이를 이용한 질의 처리 알고리즘과 색인 선정 알고리즘을 제안하였다. 새로운 색인 기법은 기존의 비트맵 조인 색인을 객체 데이터베이스에 적합하도록 적용한 것이다. 또한, 색인의 유용성을 높이기 위하여 색인을 이용한 질의 처리 기법과 최소 비용을 가지는 색인 집합을 결정하는 색인 선정 기법을 제시하였다. 색인 선정 기법은 전통적으로 색인 개수에 대하여 지수 함수로 증가하

는 계산 복잡도를 가지는 문제이지만, 본 연구에서는 비용 함수의 특성을 이용하여 선형시간 이내에 최소 비용의 색인 집합을 구할 수 있도록 하였다. 제안된 색인 선정 기법은 질의 최적화 문제 뿐 아니라 물리적 데이터베이스 설계 문제에서도 제안된 비트맵 인덱스를 선정하는 기법으로 이용될 수 있다.

향후 연구로써 연구 결과를 스타 조인 뿐 아니라 그루핑 연산이나 롤업/드릴다운 연산에도 적용될 수 있도록 확장하고자 한다. 그리고, 실제로 객체 DBMS에 구현하여 제안된 기법의 유용성을 실제 시스템에서 입증하는 것도 향후 과제로 남아 있다.

참 고 문 헌

[1] Bertino, E. & W. Kim "An indexing technique for query on nested object," *IEEE Trans. on Knowledge and Data Engineering,* 1(2), 1989, pp.196-214.

[2] Binh, N. T., et al., "An Object Oriented Multidimensional Data Model for OLAP," *Web-Age Information Management,* 2000, pp.69-82 [DBLP:conf/waim/NguyenTW00]

[3] Chan, C. Y. & Y. Ioannidis, "Bitmap index design and evaluation," In *Proc. of the ACM SIGMOD*, 1998 pp.355-366.

[4] Colliat, G., "OLAP, relational, and multidimensional database system," In *Proc. Intl ACM SIGMOD Record, 25(2)*, 1996.

[5] Czejdo, B. D., et al., "Design of a Data Warehouse over Object-Oriented and Dynamically Evolving Data Sources," *DEXA Workshop*, 2001, pp.128-132.

[6] Deshpande, P., et al., "Cubing algorithms, storage estimation, and storage and processing alternatives for OLAP," *IEEE Data Engineering Bulletin,* 20(1), 1997, pp.3-11.

[7] Do, L., et al., Issues in developing very large data warehouses, In *Proc. Int'l Conf. on VLDB*, 1998.

[8] Gopalkrishnan, V., et al., "Star/Snow-Flake Schema Driven Object-Relational Data Warehouse - Design and Query Processing Strategies," *DaWaK,* pp. 11- 22, 1999

[9] Gu, J., et al., "OLAP++: Powerful and Easy-to-Use Federations of OLAP and Object Databases," *VLDB*, 2000, pp.599-602[DBLP : conf/vldb/GuPS00].

[10] Horowitz, E., et al., *Computer Algorithm*, Computer Science Press, 1996.

[11] Huynh, T. N., et al., "Metadata for object-relational data warehouse," In *Proc. DMDW*, 2000[DBLP:conf/dmdw/HuynhMT00].

[12] Inmon, W. H., *Building the data warehouse,* John Wiley, 1992.

[13] Johnson, T., "Performance Measurements of Compressed Bitmap indices," In *Proc. Intl. Conf. on VLDB*, 1999, pp.278-289.

[14] Jurgens, M. & H. J. Lenz, "Tree based indexes vs. bitmap indexes : a performance study," In *Proc. Intl. Workshop on Design and Management of Data Warehouses (DMDW'99)*, 1999.

[15] O'neil, P. & G. Graefe, "Multi-table joins through bitmapped join indices," *SIGMOD Record, 24(3)*, Sept. 1995, pp.8-11.

[16] O'Neil, P. & D. Quass, "Improved query performance with variant indexes," In *Proc. of the ACM SIGMOD*, 1997, pp.38-49.

[17] Pedersen, T. B., et al., "Extending OLAP Querying to External Object Databases," *CIKM 2000*, pp.405-413.

[18] Ravat, F. & O. Teste, "A Temporal Object-Oriented Data Warehouse Model,"

DEXA 2000, pp.583-592.

[19] Sarawahi, S., "Indexing OLAP data," *Bulletin of the TC on Data Eng., 0(1),* Mar. 1997.

[20] TPC-D, www.tpc.org, 2002

[21] UniSQL, UniSQL/X 사용자 매뉴얼, Release 1.0, 1991.

[22] Valduriez, P., "Join indices," *ACM Transactions on Database Systems,* 12(2), 1987, pp.218-246.

[23] Wu, M. C. & A. P. Buchmann, "Encoded bitmap indexing for data warehouses," In *Proc. Int'l Conf. ICDE,* 1998.

[24] Wu, K. L. & P. S. Yu, "Range-based bitmap indexing for high cardinality attributes with skew," Research Report, IBM Watson Research Center, May 1996.

[25] Yao, S. B., "Approximating block accesses in database organizations," *Comm. of the ACM,* 20(4), 1977, pp. 260-261.

[26] Zhao, Y., et al., "Array-based evaluation of multi-dimensional queries in object-relational databases system," In *Proc. Int'l Conf ICDE,* 1998.

■ 저 자 소 개

조 완 섭

현재 충북대학교 경영정보학과 부교수로 재직중이다. 경북대학교 이학사(1985), KAIST에서 전산학 전공으로 공학석사(1987)와 공학박사(1996)을 취득하였다. 주요 관심분야는 Databases, XML, OLAP과 Data Warehousing, Data Mining, CRM, Bioinformatics 등이다.

정 태 성

충북대학교 경영정보학과 학사 졸업 후, 현재 충북대학교 정보산업공학과 석사과정에 재학중이다. 관심분야는 데이터베이스, OLAP, GUI, DM, WEB 등이다.

이 현 철

충북대학교 경영정보학과 학사 졸업 후, 현재 충북대학교 정보산업공학과 석사과정에 재학중이다. 관심분야는 데이터베이스, Data Mining, OLAP, Web, XML 등이다.

장 혜 경

충북대학교 전자계산학과 석사 졸업 후, 현재 경영정보학과 박사과정을 이수 중이며, 관심분야는 Data warehouse, OLAP, Data Mining 이다.

연락처는 043-261-3258이고, E-mail은 lodestone@naver.com 이다.

안 명 상

충북대학교 전자계산학과 석사 졸업 후, 현재 경영정보학과 박사과정을 수료중이며, 관심분야는 Data Warehouse, OLAP, Data Mining, BioInformatics 이다.

연락처는 043-261-3258이고 E-mail은 epita55@naver.com이다.

◈ 이 논문은 2002년 12월 10일 접수하여 2차 수정을 거쳐 2003년 5월 30일 게재확정되었습니다.