복합객체 질의 처리를 위한 인덱스 구조의 설계
Design of Index Structure for Complex Object Query Processing

서상훈(사무자동화과)
Sang-Hoon Suh (Dept. of Office Automation)

Key Words : 객체지향데이타베이스, 인덱스 구조, 저장구조, 복합객체

ABSTRACT: Although many of the problems that must be solved by object-oriented databases are similar to relational databases, there are also many problems that are unique to the object-oriented model. One of those is indexing structure. Because of the wide application of object-oriented database systems, query processing and indexing have become an importance issue in the success of object-oriented database systems, especially when we deal with composite objects. Index for complex object hierarchy must support the superclass and subclass relationship, and nested relation of class. In this paper, I propose a index structure for complex object query processing. Index method for the class hierarchies is the class-hierarchy index. In the nested object, the nested index, the path index and multi index can be used. However, there is no index method for the complex object hierarchy, and the combination of single index and class-hierarchy index has been used. This index technique is inadequate to efficiently support queries on complex object hierarchies. In this paper I address the existing index methods and the problem of retrieving the object-oriented databases using those methods. The primary goal of the research is to develop novel indexing technique which can be used in complex object hierarchy.

1. 서 론

일반적으로 객체지향 데이터베이스 시스템은 복합 객체 데이터베이스 시스템에서 주어진 질의를 효율적으로 처리하는데 있어서 선택된 인덱스 기법에 따라 많은 영향을 받는다. 따라서 커다란 데이타베이스에 있는 데이타의 작은 일부를 검색하는 질의를 효율적으로 처리하기 위해서는 적당한 인덱스의 선택이 매우 중요하다[1,2,3]. 기존의 관계형 데이터베이스 시스템에 대해서는 많은 인덱스 기법이 연구되어 왔으며 B 트리 구조의 변형된 형태나 해싱 기법을 제공하고 있다. 그러나 객체지향 데이타 모델은 관계형 모델과는 많은 차이점을 가지므로 관계형 데이타베이스를 위한 인덱싱 기법은 객체지향 데이타베이스에는 부적절하다. 그러므로 객체지향 질의를 효율적으로 지원하기 위한 인덱싱 기법이 개발되어야 한다[1,2,3,6,7,9]. 따라서 객체지향 데이터베이스에서의 중첩객체에 대한 질의처리를 지원하기 위한 많은 기법들이 제안되어 왔다.

객체지향 데이타 모델은 객체, 객체 식별자, 캡슐화, 클래스, 메소드, 클래스 계층, 상속성, 복합 객체 등과 같은 많은 개념에 기초하고 있다[7,11]. 특히 복합 객체 개념은 CAD/CAM, SE, OA, GIS등과 같은 많은 분야에서 중요하게 이용되고 있다. 복합 객체 스키마에서 한

객체의 속성의 값은 객체일 수도 있고, 다른 클래스의 객체 집합일 수도 있다. 이러한 복합 객체는 A_PART_OF 관계성을 나타내며, 중첩 관계형 데이타베이스에서의 중첩 객체와 유사하다. 객체 지향 데이타베이스와 중첩 관계형 데이타베이스에서는 중첩인덱스, 경로 인덱스, 다중 인덱스와 같은 인덱스 기법들이 중첩 객체에 대한 질의를 효율적으로 지원하도록 제공되고 있다[1,2,3,6,12].

중첩 관계형 데이타 모델과는 달리 객체 지향 데이타 모델에서의 클래스는 여러개의 하위 클래스로 분류될 수 있다. 그림 1에 주어진 스키마에서 클래스 Computer와 클래스 Main이 이러한 관계성을 나타낸다. 여기서는 상위클래스와 하위클래스의 관계성을 지니고 있는 복합 객체 스키마를 복합 객체 계층이라고 한다. 복합 객체 계층에 대한 질의는 한 클래스와 그 클래스 어트리뷰트의 도메인이 클래스인 클래스 뿐만 아니라 클래스 계층에 따른 모든 하위 클래스에 이르기 까지에 대한 액세스를 요구한다. 복합 객체나 중첩 객체에 대한 기존의 인덱싱 기법은 복합 객체 계층에 대한 질의를 지원하기에는 부적절하다.

이 논문에서는 객체지향 데이터 모델을 살펴보고, 기존의 인덱싱 기법와 질의 형태와의 관계를 살펴보고 복합 객체 계층 질의에 적용가능한 복합객체계층 인덱스 구조를 제시한다.

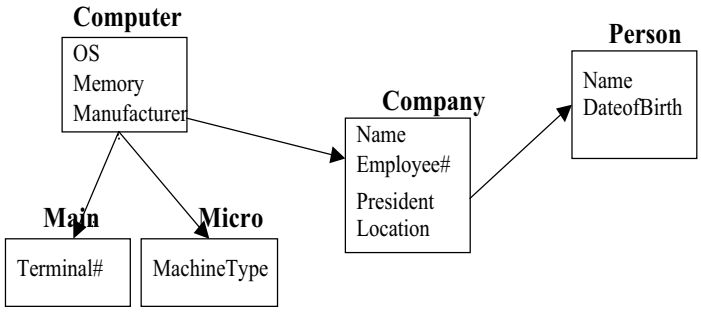


그림 1. 데이터 베이스 스키마

2. 데이터 모델과 인덱싱 구조

2.1 객체 지향 데이터 모델의 기본 개념

객체지향 데이타 모델은 다음과 같은 개념을 가지고 있다.

- 객체와 ID : 각각의 실세계의 개체는 객체로 표현되어지고 각 객체는 유일한 식별자를 가지고 있다.
- 복합 객체 : 어트리뷰트 집합은 각 객체와 연관되어진다. 각 어트리뷰트 값은 하나의 객체일수도 있고 객체들의 집합일 수도 있다. 이러한 특성으로 인해 임의의 복합 객체가 다른 객체들의 항으로 정의될 수 있다.
- 캡슐화 : 각 객체는 다른 객체에 의해 접근되어지고 조작되어질 수 있는 메소드와 인터페이스를 가지고 있다. 한 객체의 인터페이스는 다른 객체에 적용될 수 있는 연산들의 집합으로 구성되어 있다. 객체의 상태는 해당 연산에 의해 야기된 메소드에 의해 조절되어진다.
- 클래스 : 어트리뷰트와 메소드를 같이 공유하는 모든 객체들은 클래스로 그룹지어진다. 각 객체는 어떠한 클래스에 속하게 된다.
- 상속성 : 어떤 클래스는 하나 이상의 다른 클래스의 또다른 인스턴스로 정의될 수도 있고 그러한 클래스들의 어트리뷰트와 메소드를 상속할 수 있다. 그렇게 정의된 클래스를 하위클래

스라하고 인스턴스를 가지고 있는 다른 클래스를 상위클래스라 한다.

2.2 인덱싱 구조

객체 지향 데이타베이스에서 질의를 효율적으로 처리하기 위해서는 각 형태의 질의에 맞는 적절한 인덱스 기법을 사용해야 한다. 객체 지향 데이타베이스에서의 질의 형태와 인덱스 기법과는 밀절한 관계를 가지고 있다. 단일 클래스는 관계형 데이타베이스의 인덱싱 기법, 클래스 계층은 단일 클래스 인덱스, 클래스-계층 인덱스, 복합 객체는 중첩인덱스, 경로인덱스, 다중인덱스 그리고 복합 객체는 계층 중첩상속인덱스가 적합한 인덱스 구조이다.

복합 객체들의 집합에 대한 검색에서는 프리디킷에 포함된 속성들과 목표 클래스들이 복합 구조의 내부에 깊게 중첩될 수 있다. 그러므로 복합객체에 대한 질의를 수행하기 위해서는 중첩된 속성으로 객체를 인덱싱하는 것을 고려해야 한다. 이러한 인덱싱 기법들로는 단순 인덱스, 경로 인덱스, 중첩 인덱스, 중첩 상속 인덱스 등이 있다.

1) 단순 인덱스

단순 인덱스는 클래스의 한 어트리뷰트에 대한 인덱스로서 각 클래스는 각자의 인덱스를 가지고 있다. 즉, 클래스의 각 어트리뷰트에 대한 인덱스 리스트를 유지한다.

2) 경로 인덱스

경로상의 시작에서 끝에 나타난 객체에 대한 인덱스값을 유지하고 있다. 전향 순회가 요구됨으로써 경로 인덱스에 대한 갱신은 비용이 많이 들지만 역 참조가 필요없다.

3) 중첩 인덱스

갱신을 하기 위해서는 전향 순회와 후향순회를 요구한다. 전향 순회는 변경된 어트리뷰트에 대한 인덱스된 어트리뷰트의 값을 결정해야 한다. 역 참조를 하기 위해서는 경로의 앞부분에서 인스턴스를 결정해야 한다. 역 순회는 객체들 사이의 역 참조가 있지 않기 때문에 매우 많은 비용을 필요로 한다. 그러한 경우 중첩 인덱스 방법은 유용하지 않다.

4) 중첩 상속 인덱스

중첩 상속 인덱스는 주어진 경로상의 모든 클래스와 하위클래스에 대해 인덱스를 생성한다.

3. 복합 객체 계층 인덱스 구조

3.1 기본 정의

복합 객체 계층을 지원하기 위한 인덱스 기법은 상위클래스와 하위클래스와의 관계와 클래스의 중첩 관계를 지원해야 한다. 그러기 위해서는 질의에 나타난 모든 클래스에 대한 인덱싱이 요구된다. 복합 객체 인덱스는 스키마 그래프 구조를 잘 표현하고 잘 반영한다.

객체 지향 데이타 베이스에서의 검색에서는 중첩된 속성을 위해서 경로라는 개념이 사용된다. 경로는 클래스와 클래스의 어트리뷰트 도메인 리스트로 구성되어 있다.

〔정의 1〕 복합 객체 스키마 H에 대하여, H에 대한 경로 P는 다음과 같이 정의된다.

$$P = C_1.A_1.A_2.\ldots.A_n (n \geq 1)$$

단, C_1 는 클래스이다.

A_1 는 클래스 C_1의 어트리뷰트이다.

A_i 는 클래스 C_{i-1}의 어트리뷰트 A_{i-1}의 도메인인 C_i의 어트리뷰트이다. ($1 < i \leq n$)

〔정의 2〕 루트가 C인 클래스-계층 스키마 H에 대하여, 클래스 C에 속하는 인스턴스의 확장 집합은 다음과 같이 정의한다.

I_C = { x | x는 클래스 C 혹은 C_i의 인스턴스이다. 단, C_i 는 C의 하위클래스이다.}

정의 2는 어떤 한 클래스와 그 클래스에 대한 하위클래스에 속하는 인스턴스에 관해 정의하고 있다. 상위클래스에 대한 접근이 그 클래스뿐만 아니라 그 클래스의 하위클래스에 대해 순회를 하고자하는 경우에는 확장집합을 이용하여 수행된다.

〈보기 1〉 그림 1과 그림2에서 클래스 'Computer'에 대한 확장 집합은 클래스 'Computer'뿐만 아니라 하위 클래스인 'Main'과 'Micro'에 대한 인스턴스를 포함한다.

{Computer〔1〕, Computer〔2〕, Computer〔3〕, Main〔4〕, Main〔5〕, Micro〔8〕, Micro〔9〕}이다.

〔정의 3〕 경로 $P = C_1.A_1.A_2....A_n$가 있을때, P에 대하여 인스턴스화된 p는

$p = c_1.a_1.a_2.....a_n$ 로 정의된다.

단, c_1는 클래스 C_1의 한 인스턴스이다.

a_i는 객체 a_{i-1} $(1 \le i \le n)$ 의 어트리뷰트 A_i에 대한 값이다.

〔정의 4〕 복합 객체 계층 스키마 S에 있는 한 경로는 $P = C_1.A_1.A_2.....A_n$의 한 인스턴스 $p = c_1.a_1.a_2.....a_n$ 에 대하여, 인스턴스화 p에 대한 인스턴스화 집합 Sp는 다음과 같이 정의된다.

Sp = { 〈C, { o_i | o_i는 C의 인스턴스 } 〉 | C는 C_1이거나 C_1의 하위클래스이고, 경로 인스턴스인 $o_i.a_1.a_2.....a_n$ 가 존재한다. }

복합 객체 계층 인덱스는 위의 정의에 기초하여 다음과 같이 정의한다.

〔정의 5〕 복합 객체 계층 스키마 S에 있는 한 경로 $P = C_0.A_1.A_2.....A_n (n \ge 1)$에 대하여 복합 객체 계층 인덱스는 (O, 〈 S_p | p ∈ P 〉)의 쌍의 집합으로 정의된다. 이때 O 는 클래스 C_n의 한 어트리뷰트 A_n의 값이다. P 는 H에 있는 경로 p에 대한 모든 인스턴스화 집합이다.

S_p 는 정의 3을 따른다.

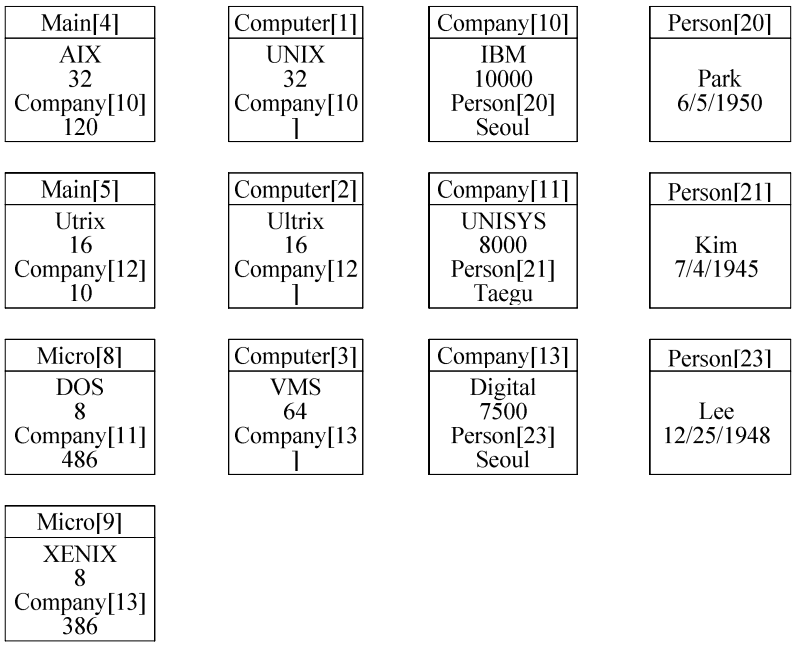


그림 2. 그림1에 해당하는 인스턴스 예

〈보기 2〉 그림 2의 객체들은 그림 1에 보여진 클래스의 인스턴스들이다. 그림2에서는 경로 P=Computer.Manufacturer.Location로 가정하고 있다. 그러면 이 경로에 대한 복합객체 계층 인덱스는 지정된 'Location'과 그 'Location'에 위치한 'Company'에서 생산된 모든 'Computer'를 연결시킨다. 이 예는 다음의 항값을 가진다.

(Seoul,
〈{〈Computer,{Computer[1]}〉,
〈Main,{Main[4]}〉,〈Company,{Company[10]}〉},
{〈Computer,{Computer[13]}〉,
〈Micro,{Micro[9]}〉,〈Company,{Company[3]}〉}〉)
(Taegu,
〈{〈Micro,{Micro[8]}〉, 〈Company,{Company[11]}〉}〉)

3.2 특성

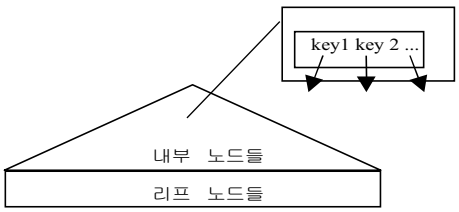
여기에서 제안된 인덱싱 기법은 중첩 인덱스, 경로 인덱스, 중첩 경로 인덱스의 특징이 조합된 기법으로서 다음의 특성을 가지고 있다.

. 중첩 인덱스와 경로 인덱스의 특징을 지니고 있으며 클래스간의 상속성을 나타낸다.
. 필요한 경로 정보가 유지되므로 갱신시에 역참조가 필요없다.
. 전향 순회와 후향 순회에 대한 지원이 용이하다.

3.3 인데스 트리 구조

데이터베이스 인덱스 트리의 기본 구조는 그림 3과 같다. 트리는 차수가 매우 큰 균형트리 형태를 갖는다. 탐색 트리의 기본구조는 그림 3(a) 와 같이 B+트리에 기본을 두고 있다. 그림 3(b) 는 복합 객체 계층 인덱스에서의 내부 노드를 그림 3(c) 는 리프 노드의 형태

를 보여주고 있다. 내부 노드는 n+1개의 포인터와 n개의 키값으로 구성되어 있다.



(a) 인덱스 트리 기본 구조

b_1	키$_1$	b_2	키$_2$	b_3	...	키$_n$	b_{n+1}

(b) 내부 노드의 형태

레코드 길이	키 길이	키 값	클래스 디렉토리	오프셋 리스트	oid #	oid 리스트$_1$	.....	oid #	oid 리스트$_n$

(c) 리프 노드의 형태

그림 3. 복합 객체 계층 인덱스의 구조

					offset1	
레코드 길이	5	Seoul	Computer Main Micro Company	(offset1, offset4) (offset2, null) (null, offset5) (offset3, offset6)	1	Computer[1]

offset2		offset3		offset4		offset5		offset6	
1	Mainframe[4]	1	Company[13]	1	Computer[3]	2	Micro[8], Micro[9]	1	Company[10]

그림 4. 예 2에 대한 인덱스 저장 내용

그림 3(c)에서처럼 리프 노드는 다음과 같이 구성되어 있다.

- 레코드 길이
- 키 길이
- 키 값
- 클래스 디렉토리
- 오프셋 리스트
- OID 갯수
- OID 목록

이러한 구성요소들 중에서 키 길이는 키값의 길이를 나타내며, 클래스 디렉토리는 그 키값을 갖는 인스턴스가 있는 클래스 구조를 나타낸다. 오프셋 리스트는 클래스 디렉토리에 나타난 클래스의 객체가 위치한 곳을 나타내며, OID 목록과 OID 개수는 그러한 객체 목록과 객체의 개수를 나타낸다.

그림 4는 어트리뷰트 'Location'에 대한 예를 보이고 있다.

세 번째 필드가 가지고 있는 값인 'Seoul'을 키 값을 나타내며, 그에 해당하는 키의 길이가 두 번째 필드의 '5'라는 값이 나타내고 있다. 클래스 디렉토리 필드는 'Computer, Main, Micro, Company'를 필드 값으로 가지며, 오프셋 리스트 필드의 '(offset1, offset4)'는 클래스 'Computer'에 대응하는 오프셋 값으로 'Computer' 클래스에 해당하는 객체가 각각 offset1과 offset4에 있는 Computer[1]과 Computer[3]이라는 것을 의미한다. 또한 'Seoul'을 'Location'으로 하는 'Company'는 Company[10]과 Company[13]이 있음을 나타낸다. 'null'값은 해당하는 필드의 객체가 존재하지 않음을 나타낸다.

3.4 검색 알고리즘

복합 객체 계층 인덱스는 여러 가지 형태의 질의에 대해서 좋은 효율성을 보인다. 질의가 들어왔을 때 인덱스를 통하여 질의 조건은 만족하는 결과를 만들어내는 과정은 다음과 같다.

과정 1) 이 질의의 우선 인덱스 트리로부터 질의에 주어진 키 값과 일치라는 인덱스 레코드를 찾는다.

과정 2) 키 값이 일치하는 인덱스 레코드를 찾았으면 클래스 디렉토리로부터 원하는 클래스의 오프셋을 읽어들인다.

이때 클래스 디렉토리에는 클래스 계층 구조과 내포되어 있다. 따라서 하위 클래스에 대한 별도의 인덱스 탐색이 불필요하게 된다.

과정 3) 오프셋이 가리키는 곳으로 가서 OID 개수만큼의 OID를 읽어서 결과값은 반환한다.

4. 결론

객체지향 데이터 모델의 특성 때문에 기존의 관계형 데이터 모델에 적용되던 인덱스 구조와는 다른 기법이 요구된다. 또한 질의 형태와 인덱싱 구조는 서로 밀접한 관계를 가지고 있다. 각 질의 형태는 자신에 알맞는 인덱싱 구조를 요구한다. 여기서는 복합 객체 질의 처리를 위한 인덱스 구조를 제안했다. 제안된 인덱스 구조는 기존의 인덱스 구조와는 달리 인덱스에 스키마 구조를 내포하도록 설계되었다. 이 방법은 경로 인덱스, 중첩 인덱스, 중첩 상속 인덱스의 조합된 형태이다. Oid 리스트에 포함된 각 Oid가 개별적으로 나열된다는 점에서 경로 인덱스와 유사하다. 또한 복합 객체 계층 인덱스는 중첩 인덱스의 공간 효율성과 비슷한 성능을 보인며 중첩 인덱스와 경로 인덱스의 특징을 지니고 있으며 클래스간의 상속성을 나타낸다. 그리고 갱신의 경우에는 갱신에 필요한 정보가 유지되므로 역참조 없이 인덱스의 갱신이 가능하다. 또한 전향 순향와 후향 순회에 대한 지원이 용이한 특성을 지니고 있다.

참고문헌

(1) Maier, D., Stein, J., "Indexing in an Object-Oriented DBMS", *Proc. of Intl. Workshops on OODBS*, Sep. 1986., pp171-182

(2) Bertino, E., "A Survey of Indexing Techniques for OODBMSs", *in Query Processing for Advanced Database Systems*, J. C. Freytag, D. Maier, and G. Vossen, Morgan-Kaufmann, 1994.

(3) Bertino, E., Kim, W., "Indexing Techniques for Queries on Nested Objects," *IEEE Trans. On Knowledge and Data Engineering*, Vol. 1, No. 2, June, 1989.,

pp196-214.
(4) Willshire, J., Kim, J., "Properties of Physical Storage Models for Object-Oriented Database." *Intl. Conf. on Parallel Databases, Parallel Architecture and Thier Application*, 1990.
(5) Valduriez, P., "Join Indices," ACM Trans on Database Systems, vol. 12, no. 2, 1987.
(6) Valduriez, P. , Khoshfian, S., Copeland, G., "Implementation Techniques of Complex Objects." *Proc of the 12th Intl. Conf. on VLDB*, 1986.
(7) Kim, W., Kim, K. C., Dale, A., "Indexing Techniques for Object-Oriented Databases," *in Object-Oriented Concepts, Application, and Databases*, MA: Addison-Wesley, 1989.
(8) Hua, K. A., Tripathy, C., "Object Skeletons: An Efficient Navigation Structure for OODBs.", *IEEE*, 1994, pp508-517.
(9) Bertino, E., Guglielmina, C., "Path Index:a Approach to the efficient execution of object Oriented Database queries", *Data and Knowledge Engineering*, pp.1-27, 1993.
(10) Bancilhon, F., Delobel, C., Kanellakis, P., *Object-Oriented Database System - The Story of O2*. Morgan Kaufmann Publishers, Inc., 1991.
(11) Kim, W., *Introduction to Object-Oriented Databases*. The MIT Press, 1990.
(12) Chawathe, S., Chen, M., Yu, P., "On Index Selection for Nested Object Hierarchies", *VLDB*, 1994, pp331-341