한국지리정보학회지 1권 2호 : 67~79(1998)

# GIS 응용을 위한 바다-III의 다단계 사전인출과 지연쓰기의 설계 및 구현*

박준호 · 박성철 · 심광훈 · 성준화 · 박영철[1]

# Design and Implementation of the Multi-level Pre-fetch and Deferred-flush in BADA-III for GIS Applications

Jun-Ho Park · Sung-Chul Park · Kwang-Hoon Shim
Jun-Hwa Seong · Young-Chul Park[1]

## 요 약

대부분의 GIS 응용 프로그램은 다수의 공간객체에 대하여 주로 읽기 연산을 수행하며 접근하는 공간객체가 복합 객체인 경우 그 복합객체와 그 복합객체가 포함하는 공간객체에 모두 접근하게 된다. GIS 응용 프로그램에서 공간객체의 생성, 삭제, 변경연산은 매우 드물게 일어나지만 다수의 공간객체에 대하여 수행된다. 본 논문은 GIS 응용 프로그램의 이러한 특성을 고려하여 다수의 공간객체들을 신속히 탐색하기 위한 다단계 사전인출 질의의 개념을 제시하고 생성하는 영속객체들을 최적의 성능으로 데이타베이스에 반영하기 위한 지연쓰기 기능을 객체지향 데이타베이스 시스템인 바다-III에 설계하고 구현한 내용을 제시한다. 다단계 사전인출 질의는 그 질의를 만족하는 객체들뿐만 아니라 그 객체들이 포함하는 객체들을 사용자가 명시한 단계만큼 데이타베이스로부터 인출하여 클라이언트 캐쉬에 등록하는 기능이다. 지연쓰기 기능은 응용 프로그램이 생성한 객체들에 대하여 서버의 부담을 최소화하고 클라이언트와 서버간의 통신을 최소화하면서 데이타베이스에 반영하는 기법이다. 이들 두 기능은 GIS 응용과 같이 다수의 복합객체를 탐색하고 생성하는 응용에 적합하다.

주요어: GIS, 바다-III, 다단계 사전인출, 지연쓰기

## ABSTRACT

Most GIS applications are read-intensive on a large number of spatial objects and when the spatial objects are composite objects, the contained objects within the composite objects are also accessed. In GIS applications, creation, deletion, and update operations on spatial objects occur very rarely, but once they occur they deal with a large number of spatial objects. This paper proposes the concept of the multi-level pre-fetch query to retrieve a large number of spatial objects efficiently, and the functionality of the deferred-flush on the newly created persistent objects into the database with the optimal performance, and presents the design and implementation details of those ideas into an object-oriented DBMS BADA-III while considering these characteristics of GIS applications. The

1998년 12월 23일 접수 Received on December 23, 1998
* 본 논문은 정보통신부 지원 1997년도 산 · 학 · 연 공동기술 개발사업과 한국과학재단지원 특정기초연구(과제번호: 96-01-01-03-01-3)비의 지원으로 수행되었으며, 지원에 감사를 드립니다.
1 경북대학교 컴퓨터과학과
Department of Computer Science, Kyungpook National University

multi-level pre-fetch query retrieves the objects that satisfy the query and the objects that are contained within the objects up to the level specified by users, and registers the retrieved objects on the client cache. The deferred-flush flushes a large number of composite objects that are created by the application with a minimal overhead of the server and a minimal number of communications between the client and the server. These two functionality are suitable for the applications that search or create a large number of composite objects like GIS applications.

*KEYWORDS: GIS, BADA-III, Multi-level pre-fetch, Deferred-flush*

## 서 론

GIS 응용은 그 특성상 방대한 양의 공간객체를 한꺼번에 생성하거나 이들 공간객체에 대하여 한꺼번에 읽기 연산을 수행하며 접근하는 공간객체가 복합객체인 경우 그 복합객체와 그 복합객체가 포함하는 공간객체에 모두 접근한다. 따라서, GIS 응용이 기반으로 사용하는 데이타베이스 시스템은 이러한 GIS 응용의 특성을 고려하여 방대한 양의 공간객체를 데이타베이스로부터 신속히 인출하고 방대한 양의 공간객체를 데이타베이스에 신속히 생성할 수 있어야 한다.

객체, 속성, 메소드, 클래스, 계승 등의 객체지향 데이타 모델을 지원하는 객체지향 데이타베이스 시스템인 바다-III(조옥자와 전성택, 1995; Chae 등, 1995; 조옥자 등, 1996)는 공간객체의 저장 및 관리에 용이한 구조를 가진다. 특히, 바다-III의 복합 객체(composite object)는 공간객체간의 상대적 관계를 나타내는 공간 정보를 표현하는데 용이하다. 바다-III에서 복합객체는 객체 식별자(object identifier, OID) 또는 객체 식별자 집합을 그 속성으로 가질 수 있으며 객체 식별자에 해당하는 객체와 참조-피참조 관계를 가진다.

ODMG 표준(Jordan, 1998)을 따르는 바다-III는 데이타베이스로부터 객체인출을 위한 3가지 인터페이스 질의함수를 제공한다. 첫 번째 질의함수는 단일 질의함수로서 질의조건을 만족하는 객체가 최대 하나인 경우에 사용하는 함수이다. 두 번째 질의 함수는 질의조건을 만족하는 모든 객체들을 인출하는 리스트 질의함수이다. 리스트 질의함수는 질의조건을 만족하는 객체들에 대하여 사전인출 기능을 수행한다. 세 번째 질의함수는 질의조건을 만족하는 객체들을 커서 메카니즘으로 접근할 수 있도록 지원하는 요구시 인출(fetch on-demand) 순환자(iterator) 질의함수이다. 요구시 인출 순환자 질의함수는 순환자가 질의조건을 만족하는 객체에 접근할 시점에 그 객체를 클라이언트의 캐쉬에 적제하는 기능을 제공한다.

바다-III가 제공하는 단순 질의함수, 리스트 질의함수, 그리고 요구시 인출 순환자 질의함수는 방대한 양의 복합객체(composite object)를 인출하는 GIS 응용에는 비효율적이다. 왜냐하면, 리스트 질의함수는 질의조건을 만족하는 객체들만을 인출할 뿐 그 객체가 포함하는 객체들은 한꺼번에 인출하지 않기 때문이며 요구시 인출 질의함수는 객체를 인출하는 방법으로 요구시 인출방법을 사용함으로 질의조건을 만족하는 각 객체와 그 객체가 포함하는 객체에 접근할 때마다 그 객체가 클라이언트 캐쉬에 등록되어 있지 않다면 서버와의 통신을 수행하여야 하기 때문이다.

본 논문은 이러한 GIS 응용을 고려하여 ODMG 표준의 기존 질의함수에 추가하여 다단계 사전인출(multi-level pre-fetch) 순환자 질의함수를 제안한다. 다단계 사전인출 순환자 질의함수는 클라이언트 캐쉬 영역이 허락하는 범위 내에서 질의조건을 만족하는 모든 복합객체와 그 객체들이 포함하는 객체들을 사용자가 요

구하는 단계만큼 데이타베이스 서버로부터 인출하여 클라이언트 캐쉬에 등록한다. 복합객체는 그 포함관계의 어느 단계까지 객체들을 사전인출할 것인지를 사용자로부터 제공받는다.

ODMG 표준은 사용자에 의하여 생성된 객체들을 언제 어떤 방법으로 서버에 반영하며 이들에게 객체 식별자를 부여하는지에 대하여 기술하지 않고 있다. 이는 구현에 관련된 사항으로서 어떻게 구현하는지에 따라 시스템의 성능에 지대한 영향을 미친다. 이를 구현하는 한가지 방법으로 각 객체가 생성될 때마다 서버에 이를 반영하고 하나의 객체 식별자를 반환받는 방법이 있을 수 있다. 이 방법은 각 객체가 생성될 때 그 객체의 각 속성을 정확히 명시하지 않는 경우 서버는 각 속성 값이 비어있는 또는 디폴트 값을 가지는 객체를 생성할 수밖에 없다. GIS 응용 프로그램과 같이 한번에 매우 많은 객체들을 생성해야 하는 경우, 이 방법은 클라이언트와 서버간에 객체 수효에 비례하는 메세지 교환을 요구하며 생성된 객체의 각 속성이 가변길이인 경우, 그 객체의 변경내용을 서버에 반영하면 그 객체는 서버의 데이타 디스크에서 원래 자리가 아닌 다른 위치로 이동하여야 한다. 이 경우, 생성된 객체를 탐색하기 위해서는 원래의 디스크 페이지와 이동한 디스크 페이지를 모두 검색해야 하므로 검색 효율이 저하된다. 본 논문은 객체의 생성을 효율적으로 수행하고 생성된 객체에 대한 효율적인 탐색을 제공하기 위하여 아래와 같은 기법을 제시한다.

생성되는 각 객체에 대하여 클라이언트는 가상 객체 식별자(virtual object identifier)를 부여한다. 가상 객체 식별자는 응용 프로그램이 생성한 영속객체(persistent object)를 데이타베이스에 반영하기 전에 클라이언트에 의하여 임시로 부여되는 논리적 식별자이다. 가상 객체 식별자를 가지는 영속객체는 응용 프로그램에 의하여 조작된 후 지연쓰기(deferred-flush) 기능에 의하여 아래의 경우 서버에 반영된다. (1) 트랜잭션이 완료할 때, (2) 질의(query)를 수행하기 전에 클라이언트 캐쉬 내의 생성, 삭제, 또는 변경된 객체들이 데이타베이스에 플러쉬(flush) 될 때, (3) checkpoint() 수행 시, 그리고 (4) savepoint() 수행으로 인하여 일괄적으로 데이타베이스에 저장되고 서버로부터 물리적 객체 식별자를 부여받는다. 이러한 경우 외에 가상 객체 식별자를 가진 영속객체는 클라이언트 캐쉬의 영역이 부족하여 클라이언트 캐쉬 관리자에 의하여 희생자로 선택되어 데이타베이스에 저장되고 서버로부터 물리적 객체 식별자를 부여받는다.

본 논문은 방대한 양의 객체들을 다루는 GIS 응용을 고려하여 다단계 사전인출 순환자 질의함수와 생성된 객체에 대한 가상 객체 식별자의 부여 및  지연쓰기 기능을 바다-III에 설계하고 구현한 내용을 제시한다.

본 논문의 나머지 구성은 다음과 같다. 먼저 바다-III가 제공하는 기존 인터페이스 질의함수를 통한 객체의 인출과정을 설명한 후 GIS 응용을 위하여 본 논문이 제시하는 다단계 사전인출 순환자 질의함수의 설계 및 구현 내용을 설명한다. 다음으로 객체의 생성을 효율적으로 수행하기 위하여 본 논문이 제안하는 가상 객체 식별자와 지연 쓰기의 설계 및 구현 내용을 설명하며, 마지막으로 결론 및 향후 연구과제를 기술한다.

## 다단계 사전인출 기능의 설계 및 구현

객체지향 데이타베이스 시스템의 질의평가 방법(query evaluation scheme)으로는 단일 버퍼 평가 방법(single-buffer evaluation scheme)과 이중 버퍼 평가 방법(dual-buffer evaluation scheme)이 있다(Kim, 1990). 단일 버퍼 평가 방법은 질의를 평가하기 전에 클라이언트 캐쉬 내에서 생성, 삭제 또는 변경된 객체를 데이타베이스에 반영한 후, 데이타베이스 내의 객체들을 대상으로 질의조건을 만족하는 객체들을 찾는

방법이며 이중 버퍼 평가 방법은 클라이언트 캐쉬 내의 객체들을 대상으로 질의 조건을 만족하는 객체들을 찾고 이어서 데이타베이스 내의 객체들을 대상으로 질의 조건을 만족하는 객체들을 찾아서 두 객체 집합에 대하여 합집합 연산을 취하는 방법이다. 바다-III는 질의평가 방법으로 단일 버퍼 평가 방법을 사용한다.

### 1. 요구시 인출 순환자 질의함수

순환자는 객체들을 커서 메카니즘으로 접근할 수 있도록 해주는 인터페이스로서 d_Iterator<T> 클래스의 인스턴스 객체이다. 요구시 인출 순환자는 요구시 인출 순환자 질의함수에 사용되는 순환자로서 <질의 실행 계획 식별자, 서버로부터 인출한 객체의 수, 질의버퍼(query buffer), 커서의 위치, end_flag>의 멤버 속성을 가진다.

질의 실행 계획 식별자는 서버가 질의를 처리하기 위하여 생성한 질의계획의 식별자이며, 서버로부터 인출한 객체의 수는 질의조건을 만족하는 객체들 중, 클라이언트로 읽어들인 객체의 수를 나타낸다. 질의버퍼는 질의조건을 만족하는 객체들 중, 클라이언트 캐쉬에 등록된 각 객체의 객체 설명자의 주소를 엔트리로 가진다. 커서의 위치는 질의조건을 만족하는 객체들의 집합에서 현재 커서가 위치한 곳을 가리키며 end_flag는 커서가 질의조건을 만족하는 마지막 객체까지 질의버퍼에 읽어들였을 경우 TRUE로 설정된다.

d_Iterator<T> 클래스는 질의조건을 만족하는 객체에 접근하기 위한 메소드로서 next, previous, cursor_pos를 제공하며 각 메소드는 커서를 순방향, 역방향, 또는 특정 위치로 옮기면서 해당 위치의 객체를 참조한다. next와 previous 메소드는 영속객체 참조자를 인자로 가지며 해당위치의 영속객체를 그 영속객체 참조자가 참조하도록 한다. cursor_pos 메소드는 영속객체 참조자, 옵셋(Offset), 시작위치를 인자로 가지며 질의조건을 만족하는 객체들의 집합에서 해당위치의 객체를 영속객체 참조자가 참조하도록 한다.

바다-III에서 요구시 인출 순환자 질의함수의 수행은 다음과 같다.

1) 클라이언트 캐쉬 내의 객체 중에서 생성, 삭제 또는 변경된 객체들을 데이타베이스에 반영하고 질의를 서버로 전송한다.

2) 서버는 클라이언트로부터 수신한 질의에 대해 파싱, 정규화, 정당성 검사를 수행하고 질의 실행 계획(query execution plan)을 생성한다. 서버는 생성한 질의계획의 식별자를 클라이언트로 전송한다.

3) 클라이언트는 서버로부터 수신한 질의 실행 계획의 식별자를 순환자에 저장한다.

요구시 인출 순환자 질의함수의 수행 후, 요구시 인출 순환자의 next, previous, 또는 cursor_pos 메소드를 이용하여 질의결과를 만족하는 객체에 접근하고자 하는 경우, 해당 객체가 요구시 인출 순환자의 질의버퍼에 등록되어 있지 않다면 다음을 수행한다.

1) 질의 실행 계획 식별자와 커서의 위치를 서버로 전송한다.

2) 서버는 클라이언트로부터 수신한 질의 실행 계획 식별자와 커서의 위치를 통해 질의 조건을 만족하는 해당 위치의 객체를 검색하여 그 객체를 클라이언트로 전송한다.

3) 클라이언트는 서버로부터 수신한 객체에 대응되는 메모리 형식 객체를 생성하여 클라이언트 캐쉬에 등록하고 그 객체의 객체 설명자의 클라이언트 캐쉬 주소를 요구시 인출 순환자의 질의버퍼에 등록함으로써 영속객체 참조자가 해당 위치의 영속객체를 참조하도록 한다.

바다-III에서 요구시 인출 순환자 질의함수는 GIS 응용과 같이 매우 많은 양의 객체들을 인출하는 경우에는 비효율적이다. 왜냐하면, 하나의 객체를 인출하기 위해 클이언트와 서버는 매번 통신을 하여야 하기 때문이다.

다음은 요구시 인출 순환자 질의함수를 이용하여 영속 클래스 River에 속한 모든 공간객체들을 화면에 출력하는 GIS 응용의 한 부분이다.

```
(GIS 응용 1)
  d_Iterator<d_Ref<River>>  it;
  d_Ref<River> r;
  d_OQL_Query query("select * from River");
  d_oql_execute(query, it);
  while(it.next(r) != -1) {r->Display();}
```

GIS 응용 1에서 질의 조건을 만족하는 객체의 수가1000개라면 클라이언트와 서버 사이의 메시지 교환 횟수는 1000회 이상이다. 이와 같이, 다수의 객체를 인출하는 응용 프로그램에서 요구시 인출 순환자 질의함수를 이용한 객체의 인출은 클라이언트와 서버간에 많은 수의 메시지 교환으로 인한 성능상의 문제를 초래할 수 있다.

## 2. 리스트 질의함수

바다-III의 리스트 질의함수는 질의조건을 만족하는 모든 객체들을 인출하는 한단계(one-level) 사전인출 함수이다. 리스트 질의함수의 수행은 다음과 같다.

1) 클라이언트 캐쉬 내의 객체 중에서 생성, 삭제 또는 변경된 객체들을 데이타베이스에 반영하고 질의를 서버로 전송한다.

2) 서버는 클라이언트로부터 수신한 질의를 실행하여 그 질의의 질의조건을 만족하는 객체들을 데이타베이스로부터 인출하고 그 객체들을 클라이언트에게 송신한다.

3) 클라이언트는 서버로부터 수신한 객체들에 대응되는 메모리 형식 객체를 생성하여 클라이언트 캐쉬에 등록하고 등록한 각 객체에 대하여 그 객체의 객체 설명자의 클라이언트 캐쉬 주소를 리스트 질의함수의 인자로 제공된 리스트 객체에 등록한다.

바다-III의 리스트 질의함수는 질의조건을 만족하는 객체에 대하여 한단계 사전인출만을 수행함으로 GIS 응용과 같이 다수의 복합객체와 그 복합객체가 포함하는 객체들을 모두 접근하는 응용에는 비효율적이다.

다음은 리스트 질의함수를 이용하여 영속 클래스 River에 속한 모든 공간객체들을 화면에 출력하는 GIS 응용의 한 부분이다.

```
(GIS 응용 2)
  d_List<d_Ref<River>> river;
  d_Ref<River> r;
  d_Iterator<d_Ref<River>> it;
  d_OQL_Query q("select * from River");
  d_oql_execute(q, river);
  it = river.create_iterator();
  while(it.next(r) != -1) {r->Display();}
```

GIS 응용 2에서 질의조건을 만족하는 객체의 수가 다수인 경우라도 클라이언트와 서버 사이의 메시지 교환 횟수는 1회이지만 응용 프로그램에서 질의조건을 만족하는 복합객체가 포함하는 객체에 접근할 때마다 그 객체가 클라이언트 캐쉬에 등록되어 있지 않다면 클라이언트는 서버와의 통신을 수행하여야 한다.

따라서, 복합객체에 포함된 객체가 매우 많을 경우, GIS 응용과 같이 다수의 복합객체들과 그 복합객체들이 포함하는 객체에 모든 접근하는 응용에서는 리스트 질의함수를 이용한 객체의 인출이 클라이언트와 서버간에 많은 수의 메시지 교환을 초래할 수 있으므로 적합하지 않다.

## 3. 다단계 사전인출 순환자 질의함수

GIS 응용과 같이 방대한 양의 복합객체들과 그 객체들이 참조하는 피참조 객체들을 모두 필요로 하는 응용의 경우, 요구시 인출 순환자 질의함수와 리스트 질의함수는 클라이언트와 서버의 통신 부담으로 인하여 비효율적이다.

따라서, 바다-III는 데이타베이스 질의를 위한 기존 인터페이스 함수에 추가하여 GIS 응용을 위하여 다단계 사전인출 순환자 질의함수를 제공한다.

바다-III의 다단계 사전인출 순환자 질의함수를 위하여 다음과 같은 사항을 설계 및 구현하였다.

- 다단계 사전인출 순환자의 객체 인출을 위한 인터페이스 메소드인 next(), previous(), cursor_pos()에 사전인출 기능을 추가하였다.
- 다단계 사전인출 순환자의 메소드로서 prefetch()를 설계하였다. prefetch()는 객체 식별자 리스트와 커서의 방향을 인자로 가지며, 다단계 사전인출 순환자의 인터페이스 메소드인 next(), previous(), cursor_pos()에 의하여 호출된다.
- 인출할 객체가 복합객체인 경우, 그 객체가 직접 혹은 간접으로 참조하는 객체들을 인출하기 위하여 prefetch_referenced_objects_by_level() 함수를 다단계 사전인출 순환자의 내부함수로 추가하였다. prefetch_referenced_objects_by_level() 함수는 prefetch() 메소드에 의하여 호출된다.

1) 다단계 사전인출 순환자의 구조

그림 1은 다단계 사전인출 순환자의 구조를 나타내며 각 속성은 아래와 같다.

- db_query_object : 서버가 질의를 처리하기 위하여 생성한 질의 실행 계획의 식별자를 나타낸다.
- end_flag : 커서가 질의조건을 만족하는 마지막 객체까지 질의버퍼로 읽어들였을 경우 TRUE로 설정된다.
- read_occurence : 서버로부터 읽어들인 객체의 수를 나타낸다.
- object_list : 질의버퍼를 가리키는 필드이며 인출한 각 객체의 객체 설명자의 클라이언트 캐쉬 주소를 엔트리로 가진다.
- current_cursor : 질의조건을 만족하는 객체들의 집합에서 커서의 현재 위치를 나타낸다.
- prefetch_flag : 사전인출 플래그로서 다단계 사전인출 순환자인 경우 prefetch_flag는 TRUE로 설정된다.
- num_of_total_object : 질의조건을 만족하는 객체의 수를 나타낸다.
- oid_set : 질의조건을 만족하는 객체들의 객체 식별자를 저장한다.
- fetch_level : 사전인출 단계를 나타낸다.

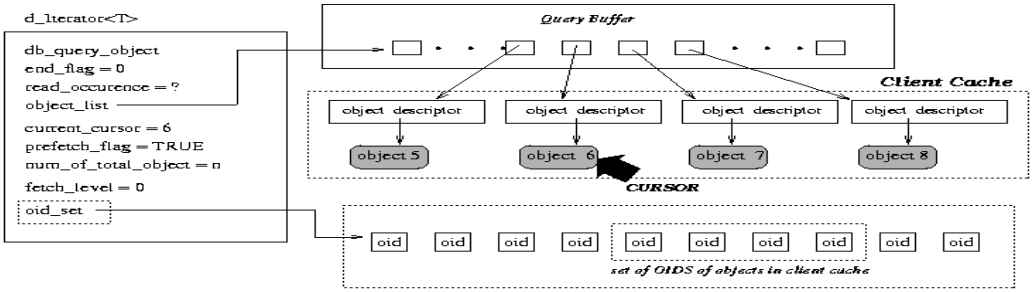


**그림 1.** 다단계 사전인출 순환자의 구조

2) 다단계 사전인출 순환자 질의함수의 설계 및 구현

다단계 사전인출 순환자 질의함수는 질의, 다단계 사전인출 순환자 그리고 사전인출 단계의 값을 인자로 가진다. 사전인출 단계는 정수 값으로 질의조건을 만족하는 객체가 복합객체인 경우 그 객체가 직접 혹은 간접 참조하는 객체들을 어느 단계까지 인출할 지를 나타내는 값이다. 표 1은 순환자 질의함수를 사전인출 기능의 지원 여부와 사전 인출 단계의 지원 여부에 따라 분류한 것이다.

**TABLE 1.** The interface of query functions

| 함수이름 | 사전인출 | 사전인출대상 |
|---|---|---|
| d_oql_execute (query,iterator) | 지원안함 | 없음 |
| d_oql_execute (query, list) | 지 원 | 질의조건을 만족하는 객체들 |
| d_oql_execute (query,iterator, level) | 지 원 | 질의조건을 만족하는 객체들과 그 객체들이 집적 혹은 간접으로 참조하는 객체들 |

d_oql_execute(query, iterator, level) 인터페이스 함수는 level의 값에 따라 사전인출 단계가 달라진다. 그림 2는 질의 조건을 만족하는 객체들과 그 객체들이 직접 혹은 간접으로 참조하는 객체들을 나타낸다. level의 값이 주어지지 않거나 level의 값이 0인 경우, 사전인출 단계는 질의 조건을 만족하는 객체들인 Polygon1과 Polygon2이다. level의 값이 1인 경우, 사전인출 단계는 (Polygon1, Polygon2)와 이들이 직접 참조하는 객체들인(Line1, Line2, ..., Line8)이다. level의 값이 3인 경우, 사전인출 단계는 (Polygon1, Polygon2)와 이들이 직접 참조하는 객체들인 (Line1, Line2, ..., Line8), 그리고 (Line1, Line2, ..., Line8)이 참조하는 객체들인 (Point1, Point2, ..., Point14)이다.

다단계 사전인출 순환자 질의함수의 실행 과정은 다음과 같다. (1) 클라이언트 캐쉬에서 생성, 삭제 또는 변경된 모든 객체들을 서버에 반영한 후 (2) 서버로 질의를 전송하며 (3) 질의결과로 질의조건을 만족하는 객체들의 객체 식별자 집합을 서버로부터 전송 받은 후 (4) 다단계 사전인출 순환자 질의함수의 인자로 제공된 다단계 사전인출 순환자를 다음과 같이 초기화한다. 첫째, 서버로부터 전송받은 객체 식별자 집합을 다단계 사전인출 순환자의 oid_list 멤버 속성에 저장한다. 둘째, 사전인출 플래그를 TRUE로 설정하여 사전인출을 위한 순환자임을 표시한다. 셋째, 질의조건을 만족하는 객체들 중 클라이언트 캐쉬에 등록된 각 객체의 객체 설명자의 주소를 엔트리로 가지는 질의버퍼를 초기화한다. 넷째, 다단계 사전인출 순환자 질의함수의 인자인 level의 값을 fetch_level에 저장한다.

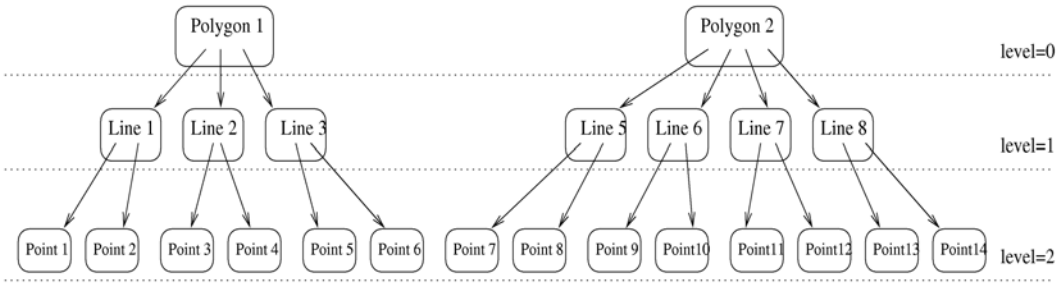


**그림 2.** 질의조건을 만족하는 공간객체와 그 공간객체가 포함하는 객체의 예

3) 다단계 사전인출 순환자의 메소드 : prefetch_referenced_objects_by_level()

다단계 사전인출 순환자의 prefetch_referenced_objects_by_level() 메소드는 다단계 사전인출 순환자의 prefetch() 메소드에 의하여 호출되며 prefetch() 메소드에 의하여 사전인출된 객체가 직접 또는 간접으로 참조하는 객체들을 데이타베이스로부터 인출하기 위해 사용된다.

prefetch_referenced_objects_by_level() 메소드는 클래스 식별자, 사전인출된 객체들, 그리고 사전인출 단계를 나타내는 정수 값을 인자로 가지며 사전인출된 객체들이 직접 참조하는 모든 객체들을 클라이언트와 서버사이의 한번의 통신으로 인출하여 클라이언트 캐쉬에 등록한다. 이 메소드는 사용자가 요구한 사전인출 단계의 범위 내에서, 인출한 객체들에 대하여 그 객체들이 직접 참조하는 객체들이 존재하지 않을 때까지 그 객체들이 참조하는 객체들을 데이타베이스로부터 인출하여 클라이언트 캐쉬에 등록한다.

4) 다단계 사전인출 순환자의 메소드: prefetch()

prefetch()는 객체 식별자 리스트와 커서의 방향을 인자로 받아 그 객체 식별자 리스트에 대응되는 객체들을 클라이언트와 서버사이의 한번의 통신으로 데이타베이스로부터 인출하여 클라이언트 캐쉬에 적재하고 커서의 방향을 참조하여 질의버퍼의 해당 엔트리에 각 객체의 객체 설명자의 클라이언트 캐쉬 주소를 등록한다. 그리고, 다단계 사전인출 순환자의 fetch_level 멤버 속성 값을 확인하여 그 값이 1 이상이면 prefetch_referenced_objects_by_level() 메소드를 호출하여 데이타베이스로부터 인출한 객체가 직접 또는 간접으로 참조하는 객체들을 인출한다.

그림 3은 사전인출 단계가 0인 경우, prefetch()의 실행과정을 나타내며 그 내용은 다음과 같다.

1. 클라이언트는 <객체 식별자 집합, 클라이언트 캐쉬의 여유 공간, 메모리 형식 객체의 크기>로 구성된 메시지를 서버에게 전송한다.
2. 클라이언트가 전송한 메시지를 수신한 서버는 그 메시지의 객체 식별자 집합에 대응되는 객체들을 클라이언트 캐쉬의 여유 공간이 허락하는 범위 내에서 모두 인출하여 클라이언트에게 전송한다.
3. 클라이언트는 서버가 전송한 객체 집합 내의 각 객체에 대하여 메모리 형식 객체를 생성하여 클라이언트 캐쉬에 적재하고 그 객체의 객체 설명자의 클라이언트 캐쉬 주소를 다단계 사전인출 순환자의 질의버퍼에 등록한다.

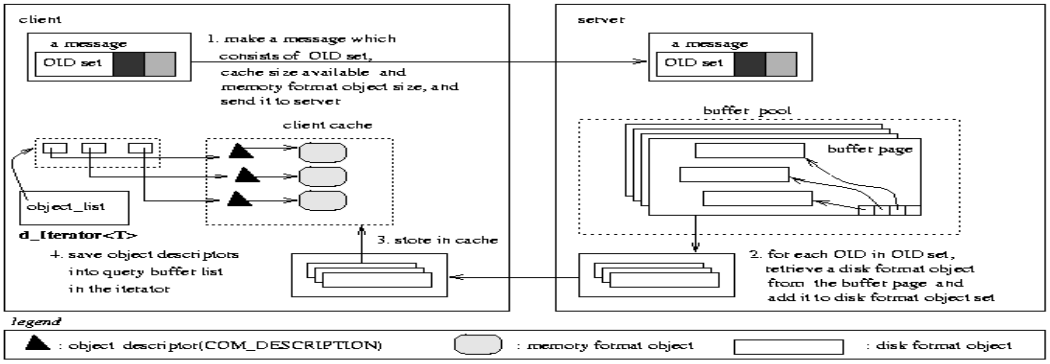


**그림 3.** 질의조건을 만족하는 객체들을 사전인출하는 과정

사전인출 수준이 1이상인 경우에는, 위의 세 단계를 수행한 후 prefetch_referenced_objects_by_level() 메소드를 호출하여 인출한 객체가 직접 또는 간접으로 참조하는 객체들을 데이타베이스로부터 인출한다.

5) 다단계 사전인출 순환자의 인터페이스 메소드: cursor_pos(), next(), previous()

다단계 사전인출 순환자의 메소드인 next(), previous(), cursor_pos()는 다음의 과정을 거쳐서 사용자가 요구하는 객체를 참조하고 관련 객체들을 사전인출한다.

(1) cursor_pos(d_Ref<T> &ref, int offset, OK_SEEK_POSITION_FLAG where_from) 메소드

질의조건을 만족하는 객체들 중 where_from에 위치한 객체로부터 offset 위치의 객체가 질의버퍼에 등록되어 있다면 영속객체 참조자인 ref가 그 객체를 참조하도록 한다. 해당 객체의 객체 설명자의 클라이언트 캐쉬 주소가 질의버퍼에 존재하지 않으면 다음을 수행한다.

- 질의조건을 만족하는 객체들 중 질의버퍼에는 등록되어 있지 않지만 클라이언트 캐쉬에 등록된 객체가 존재한다면 그 객체의 객체 설명자의 클라이언트 캐쉬 주소를 질의버퍼에 등록한다. 질의조건을 만족하는 객체 식별자들 중에서 질의버퍼에 등록된 객체들의 객체 식별자들을 제외한 나머지 객체 식별자들을 인자로하여 prefetch() 메소드를 호출하고 사용자가 요구한 커서 위치에 대응되는 객체를 영속객체 참조자인 ref가 참조하도록 한다.

(2) next(d_Ref<T> &ref) 메소드

현재 커서 위치 다음에 위치한 객체를 영속객체 참조자인 ref가 참조하도록 한다. 해당 객체의 객체 설명자의 클라이언트 캐쉬 주소가 질의버퍼에 등록되어 있다면 영속객체 참조자인 ref가 그 객체를 참조하도록 한다. 해당 객체의 객체 설명자의 클라이언트 캐쉬 주소가 질의버퍼에 존재하지 않으면 다음을 수행한다.

- 커서의 위치 값이 i이고 질의조건을 만족하는 객체 식별자 리스트가 {oid1, oid2, ... ,oidi, oidi+1, ... ,oidn}이라면 next()는 {oidi+1, oidi+2, ... ,oidn}을 인자로 하여 prefetch() 메소드를 호출하고 oidi+1에 대응하는 객체를 영속객체 참조자인 ref가 참조하도록 한다. 질의조건을 만족하는 객체들 중, 질의버퍼에는 등록되어 있지 않지만 클라이언트 캐쉬에는 등록된 객체가 존재한다면 prefetch() 메소드를 호출하기 전에 그 객체의 객체 설명자의 클라이언트 캐쉬 주소를 질의버퍼에 등록하며 그 객체의 객체 식별자를 prefetch() 메소드의 인자로 제공되는 객체 식별자 리스트에서 제외한다.

(3) previous(d_Ref<T> &ref) 메소드

현재 커서 위치 전에 위치한 객체를 영속객체 참조자인 ref가 참조하도록 한다. 해당 객체의 객체 설명자의 클라이언트 캐쉬 주소가 질의버퍼에 등록되어 있다면 영속객체 참조자인 ref가 그 객체를 참조하도록 한다. 해당 객체의 객체 설명자의 클라이언트 캐쉬 주소가 질의버퍼에 등록되어 있지 않다면 다음을 수행한다.

- 커서의 위치 값이 i이고 질의조건을 만족하는 객체 식별자 리스트가 {oid1, oid2, ... ,oidi-1, oidi, ... ,oidn}이라면 previous()는 {oid1, oid2, ... ,oidi-1}을 인자로 하여 prefetch() 메소드를 호출하고 oidi-1에 대응되는 객체를 영속객체 참조자인 ref가 참조하도록 한다. 질의조건을 만족하는 객체들 중, 질의버퍼에는 등록되어 있지는 않지만 클라이언트 캐쉬에 등록된 객체가 존재한다면 prefetch() 메소드를 호출하기 전에 그 객체의 객체 설명자의 클라이언트 캐쉬 주소를 질의버퍼에 등록하며 그 객체의 객체 식별자

를 prefetch() 메소드의 인자로 제공되는 객체 식별자 리스트에서 제외한다

## 가상 객체 식별자와 지연쓰기 기능의 설계 및 구현

바다-III 응용 프로그램에서 영속가능 클래스의 인스턴스를 영속객체로 생성하기 위해서는 new 연산자의 인자로 OK_PERSISTENT를 명시하여야 하며 임시객체(transient object)로 생성하기 위해서는 인자를 가지지 않는 new 연산자를 사용하거나 new 연산자의 인자로서 OK_TRANSIENT를 명시하여야 한다.

바다-III에 지연쓰기 기능을 부여하기 위해서 가상 객체 식별자 관리자(virtual OID manager)와 지연쓰기 객체 관리자(deferred-flush object manager)를 설계하고 구현한 내용은 아래와 같다.

### 1. 가상 객체 식별자 관리자의 설계

바다-III의 객체 식별자는 <CID, PID, Slot, RSC>로 구성되는 물리적 객체 식별자이다. CID는 객체가 속한 클래스의 식별자이고 PID는 객체가 저장된 페이지의 페이지 식별자이며 Slot은 페이지 내의 슬롯번호이다. RSC(Re-organization Slot Count)는 페이지에서 하나의 객체가 삭제되고 그 객체가 사용한 슬롯이 다른 객체를 위해 다시 사용되는 경우, 삭제된 객체와 새로운 객체의 객체 식별자를 달리하기 위해 사용된다.

가상 객체 식별자 관리자는 응용 프로그램이 생성한 영속객체들에게 가상 객체 식별자를 부여하고 가상 객체 식별자와 실제 객체 식별자의 매핑 테이블을 유지한다. 가상 객체 식별자는 <CID, Serial Number>로 구성된다. 하나의 응용 프로그램에서 처음 생성한 영속객체의 가상 객체 식별자의 일련 번호(Serial Number)는 1이며 다음 생성되는 영속객체들에서는 그 값이 1씩 증가한다.

### 2. 지연쓰기 객체 관리자의 설계

바다-III에서 응용 프로그램이 new 연산자를 이용하여 생성한 영속객체는 자동으로 지연쓰기 객체이다. 지연쓰기 객체 관리자는 지연쓰기 객체 그룹(deferred-flush object group)을 관리하며 지연쓰기 객체들의 정보를 저장하였다가 (1) 트랜잭션이 완료할 경우, (2) 질의를 수행하기 전에 클라이언트 캐쉬 내의 생성, 삭제 또는 변경된 객체들을 데이타베이스에 반영할 경우, (3) checkpoint()를 수행할 경우, 그리고 (4) savepoint()를 수행할 경우, 지연쓰기 객체들을 일괄적으로 데이타베이스에 저장하고 그 지연 객체들에게 물리적 객체 식별자를 부여한다. 이러한 경우 외에 가상 객

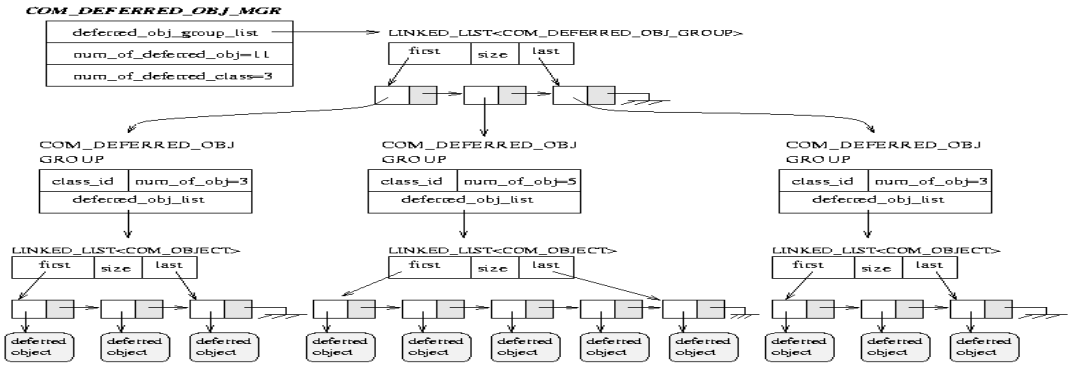


**그림 4.** 지연쓰기 객체 관리자의 구조

체 식별자를 가진 영속객체는 클라이언트 캐쉬 영역이 부족한 경우, 클라이언트 캐쉬 관리자에 의하여 희생자로 선택되어 데이타베이스에 저장되고 서버로부터 물리적 객체 식별자를 부여받는다.

지연쓰기 객체 그룹은 동일한 클래스에 속한 지연쓰기 객체들의 그룹이다. 그림 4는 지연쓰기 객체 관리자와 지연쓰기 객체 그룹, 그리고 이들이 관리하는 지연쓰기 객체들의 예를 나타낸다.

지연쓰기 객체 관리자는 아래의 필드들을 가진다.

- num_of_deferred_obj : 지연쓰기 객체들의 수를 나타낸다.
- num_of_deferred_class : 지연쓰기 객체들이 속한 서로 다른 클래스들의 수를 나타낸다. 이 값은 지연쓰기 객체 그룹의 수와 같다.
- deferred_obj_group_list : 지연쓰기 객체 그룹의 연결 리스트이다.

지연쓰기 객체 그룹은 다음과 같은 필드를 가진다.

- class_id : 지연쓰기 객체 그룹에 속한 지연쓰기 객체들의 클래스 식별자를 나타낸다.
- num_of_obj : 지연쓰기 객체 그룹에 속한 지연쓰기 객체의 수를 나타낸다.
- deferred_obj_list : 지연쓰기 객체들에 대하여 각 객체의 객체 설명자의 클라이언트 캐쉬 주소를 엔트리로 가지는 연결 리스트로 지연쓰기 객체 리스트라 한다.

1) 지연쓰기 객체의 생성 과정

응용 프로그램에서 영속객체를 생성할 경우 지연쓰기 객체 관리자는 그 객체에 대하여 (1) 가상 객체 식별자 관리자로부터 하나의 가상 객체 식별자를 할당받아 해당 객체에 그 가상 객체 식별자를 부여하고 (2) 지연쓰기 객체 그룹 리스트에서 영속객체의 클래스 식별자와 일치하는 클래스 식별자를 가진 지연쓰기 객체 그룹을 찾은 후, 그 지연쓰기 객체 그룹에 생성된 영속객체를 등록한다. 만약, 영속객체의 클래스 식별자와 일치하는 클래스 식별자를 가진 지연쓰기 객체 그룹이 없다면 새로운 지연쓰기 객체 그룹을 생성하고 그 지연쓰기 객체 그룹에 그 영속객체를 등록한다.

지연쓰기 객체 그룹의 영속객체들은 한꺼번에 데이타베이스에 반영되는 특성 때문에, 많은 영속객체에 대하여 배타적 로크를 요구할 때, 작업의 지연 또는 교착상태(deadlock)에 빠질 수 있다. 바다-III는 사용자가 해당 클래스에 대하여 공유모드(S)의 로크 또는 배타적 모드(X)의 로크를 요구할 수 있도록 트랜잭션 인터페이스를 제공하며 이를 사용함으로써 위의 문제를 해결할 수 있다.

2) 데이타베이스에 지연쓰기 객체의 반영과정

지연쓰기 객체 관리자가 영속객체들을 데이타베이스에 반영하는 과정은 다음과 같다.

- 1 단계 : 지연쓰기 객체 그룹 리스트의 각 지연쓰기 객체 그룹에 대하여 그 클래스의 속성정보를 분석하여 지연쓰기 객체 그룹간의 참조관계 그래프를 생성한다. 클래스 A가 다른 클래스 B의 인스턴스의 객체 식별자 또는 객체 식별자 집합을 그 속성으로 가지면 A 클래스는 B 클래스를 참조하고 A 클래스와 B 클래스는 참조관계에 있다. 예를 들어, 응용 프로그램이 그림 2의 공간객체를 생성하였다면 그림 5와 같은 참조관계 그래프가 생성된다.

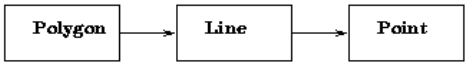


**그림 5.** 그림 2의 지연 객체들을 위한 지연쓰기 객체 그룹간의 참조 그래프

- 2 단계 : 1 단계에서 생성한 클래스간의

참조관계 그래프를 기반으로 각 지연쓰기 객체 그룹의 데이타베이스 반영 순서를 결정한다. 지연쓰기 객체 그룹간의 참조관계 그래프는 크게 두 가지 경우로 나누어진다. 첫째는 참조관계 그래프가 사이클(cycle)을 형성하지 않는 경우이고 둘째는 참조관계 그래프가 사이클을 형성하는 경우이다.

참조관계 그래프가 사이클을 형성하지 않는 경우 그 그래프를 위상 정렬(topological sort)하여 그 순서대로 각 지연쓰기 객체 그룹을 데이타베이스에 반영하고 반영한 지연쓰기 객체들의 물리적 객체 식별자들을 서버로부터 수신하여 가상 객체 식별자와 실제 객체 식별자의 매핑 테이블에 등록한다. 그림 5의 경우 Point, Line, Polygon의 순으로 각 지연쓰기 객체 그룹의 객체들은 데이타베이스에 반영된다. 각 지연쓰기 객체 그룹의 객체들을 데이타베이스에 반영할 때 그 지연쓰기 객체 그룹이 다른 지연쓰기 객체 그룹을 참조하고 있다면 그 지연쓰기 객체 그룹을 데이타베이스에 반영하기 전에 가상 객체 식별자와 실제 객체 식별자의 매핑 테이블을 해쉬하여 그 그룹의 각 객체가 유지하는 가상 객체 식별자를 실제 객체 식별자로 갱신한 후 데이타베이스에 반영한다.

참조관계 그래프가 사이클을 형성하는 경우, 적어도 하나의 지연쓰기 객체 그룹은 데이타베이스에 반영된 후 이 후에 다시 갱신되어야 한다. 왜냐하면, 참조관계 그래프가 사이클을 형성한다면 어떠한 순서로 지연쓰기 객체 그룹을 데이타베이스에 반영하더라도 적어도 한 지연쓰기 객체 그룹의 객체들은 그 그룹의 객체들을 데이타베이스에 반영할 시점에 그 객체의 속성 값으로 가상 객체 식별자를 유지하고 있기 때문이다. 따라서, 이러한 지연쓰기 객체 그룹의 객체들은 데이타베이스에 반영된 후, 그 객체들이 참조하는 객체들에게 실제 객체 식별자가 할당되면 그 객체들이 속성 값으로 유지하는 가상 객체 식별자를 실제 객체 식별자로 갱신하여야 한다. 바다-III는 지연쓰기 객체 그룹간의 참조관계 그래프가 사이클을 형성하는 경우 다음의 방법을 이용하여 데이타베이스에 반영한 지연쓰기 객체 그룹의 객체들에 대한 갱신작업을 최소화한다.

첫째, 사이클에 참여하지 않는 지연쓰기 객체 그룹이 존재한다면 그 지연쓰기 객체 그룹을 먼저 데이타베이스에 반영한다. 그러한 지연쓰기 객체 그룹이 존재하지 않으면 가장 많은 객체들을 가진 지연쓰기 객체 그룹을 찾고 그 지연쓰기 객체 그룹에서 참조하는 지연쓰기 객체 그룹들의 객체들을 데이타베이스에 반영한다.

둘째, 가장 많은 객체들을 가진 지연쓰기 객체 그룹의 객체들을 데이타베이스에 반영한다. 이 후의 데이타베이스 반영 순서는 바로 전에 반영한 지연쓰기 객체 그룹을 참조하는 지연쓰기 객체 그룹의 순서로 한다.

셋째, 모든 지연쓰기 객체 그룹을 데이타베이스에 반영한 후, 아직 객체의 속성 값으로 가상 객체 식별자를 유지하는 지연쓰기 객체 그룹을 찾고 그 지연쓰기 객체 그룹의 객체들이 실제 객체 식별자를 가지도록 그 객체들을 갱신한다. 이를 위해, 지연쓰기 객체 그룹이 데이타베이스에 반영될 시점에 그 그룹의 객체가 가상 객체 식별자를 속성 값으로 유지한다면 그 그룹을 갱신대상 그룹으로 표시한다.

## 결론 및 향후 연구과제

본 논문은 GIS 응용의 특성을 고려하여 다단계 사전인출과 가상 객체 식별자 및 지연쓰기의 개념을 제시하고 이를 바다-III에 설계 및 구현한 내용을 설명하였다.

GIS 응용은 방대한 양의 공간객체에 대하여 주로 읽기 연산을 수행하고 쓰기 연산은 드물게 수행되지만 다수의 공간객체에 대하여 수행되는 특성을 가진다. GIS 응용이 다루는

공간객체는 일반적으로 복합객체이며 GIS 응용에서 복합객체에 접근할 경우, GIS 응용은 그 특성상 그 복합객체뿐 아니라 그 복합객체가 포함하는 공간객체에도 접근한다.

본 논문은 GIS 응용의 이러한 특성을 고려하여 공간객체의 인출 시, GIS 응용이 요구한 공간객체뿐 아니라 그 공간객체가 포함하는 공간객체들까지 사전인출하는 다단계 사전인출 기능을 제공함으로써 공간객체를 인출하는 GIS 응용의 속도를 향상시켰다. 공간객체 생성 시에는 가상 객체 식별자를 이용한 지연쓰기를 통해 클라이언트와 서버가 객체 생성을 위하여 주고받는 통신의 수를 최소화하여 신속히 객체를 생성하며 탐색시 불필요한 이중 탐색의 가능성을 최소화하였다.

향후 연구과제로는 본 논문에서 제시한 다단계 사전인출과 지연쓰기를 여러 GIS 응용에 적용하여 다단계 사전인출과 지연쓰기로 인한 GIS 응용의 성능향상을 정확히 평가하는 것이다. KAGIS

## 참고문헌

조옥자, 전성택. 1995. 바다-III의 스키마 관리기 설계. 한국 정보과학회, '95 추계 학술 발표 논문집.

조옥자, 이미영, 전성택. 1996. 바다-III 객체 관리자의 설계 및 구현. 한국 정보처리학회, '96 추계 학술 발표 논문집.

David Jordan 1998. C++ Object Databases Programming with the ODMG Standard. Addison Wesley.

Mi-Ok Chae, Ki-Hyung Hong, Mi-Young Lee, Ok-Ja Cho. 1995. Design of the Object Kernel of BADA-III. The 1995 Workshop on Network and System Management.

Won Kim. 1990. Introduction to Object-Oriented Databases. The MIT Press. KAGIS