삼차원 가상 공간 객체 저장 시스템의 설계 및 구현

윤동하, 임윤주°, 이종학
대구가톨릭대학교 컴퓨터정보통신공학부
e-mail:{s7348921, zasmine8071, jhlee11}@cu.ac.kr

Design and Implementation of Three-dimensional Virtual Space Object Storage System

Dong-Ha Yoon, Yun-Ju Lim°, Jong-Hak Lee
School of Computer and Informaion Communications Engineering, Catholic Univ. of Daegu

요 약

최근 삼차원 가상 공간 객체를 편리하게 제작하기 위해서 가상 공간 객체 저작도구들이 많이 사용되고 있다. 이러한 저작 도구를 이용하여 만들어진 삼차원 가상 공간 객체는 기계, 화학, 건축, 교육, 상업 등의 다양한 분야에서 그 응용이 확산되고 있다. 그러나 기존의 저작도구에서는 삼차원 공간 객체를 순차 접근 방식을 사용한 파일 저장 시스템을 사용하기 때문에 객체의 삽입, 삭제, 갱신 및 탐색에 비효율적인 면이 있다. 그뿐만 아니라 고정 길이 속성을 가지는 객체의 관리에는 효율적이나 가변 길이 속성을 가지는 공간 객체의 관리에는 효율적이지 못하다. 따라서, 본 논문에서는 현재 널리 사용되어지고 있는 관계형 데이타베이스 관리 시스템 중의 하나인 MySQL의 저장시스템을 변경하여 가변 길이 속성을 가지는 삼차원 공간 객체의 저장과 관리에 효율적인 저장 시스템을 설계하고 구현한다. 그리고 확장된 B⁺-트리를 사용하여 삼차원 객체의 빠른 검색을 지원한다.

1. 서론

최근 다양한 삼차원 가상 공간 객체를 쉽게 만들 수 있는 저작 도구가 많이 사용되고 있다. 이들은 저작 도구를 사용하는 이에게 친숙하고 편리한 인터페이스를 제공하여 빠른 시간 내에 원하는 삼차원 가상 공간 객체를 만들 수 있게 하여 준다. 삼차원 가상 공간 객체 저작 도구의 예로는 Cosmo Software사의 Cosmo World[12], Ligos Technology사의 V-Realm Builder[4], Caligari사의 TrueSpace[12], ParallelGraphics사의 Internet Space Builder[9], 그리고 Kinetix사의 3D Studio Max[1] 등이 있다.

삼차원 가상 공간 객체는 기계, 화학, 건축, 교육, 상업 등의 다양한 분야에서 그 응용이 확산되고 있다. 하지만, 기존의 저작 도구에서는 삼차원 공간 객체를 순차 접근 방식을 사용한 파일 저장 시스템을 이용하므로 객체의 삽입, 삭제, 갱신 및 탐색에 효율적이지 못하다.

또한, 삼차원 가상 공간 객체는 하나 이상의 기본 객체들의 조합으로 더욱 다양한 형태의 삼차원 가상 공간 객체를 형성한다. 이러한 객체 형성의 특성 때문에, 객체 저장에 필요한 공간은 각 객체마다 가변적으로 변하게 된다. 그러나, 현재 상용 데이타베이스 관리 시스템인 관계형 데이타베이스 관리 시스템은 고정 길이(fixed length) 속성을 가지는 객체의 관리에는 효율적이나 가변 길이 속성[10]을 가지는 공간 객체의 관리에는 효율적이지 못하다.

본 논문에서는 위의 문제를 해결하기 위해서 데이타베이스 관리 시스템의 저장 시스템을 가변 길이 속성을 잘 지원할 수 있도록 변경하여 가상 공간 저작 도구의 저장 시스템으로 이용한다. 현재 널리 사용되어 지고 있는 관계형 데이타베이스 관리 시스템 중에 하나인 MySQL[5, 6, 13]의 저장시스템을 변경하여 가변 길이 속성을 가지는 삼차원 가상 공간 객체의 저장과 관리에 효율적인 저장 시스템을 설계하고 구현한다. 그리고, 기존 저장 시스템에서 많이 사용하고 있는 인덱스 구조인 B⁺-트리를 확장하여 삼차원 객체를 효율적으로 액세스할 수 있게 한다.

2. 관련 연구

하나의 기본 객체는 여러 개의 속성으로 이루어진다. 속성으로는 객체의 이름, 모양, 위치, 크기, 색(color), 이미지 텍스처(image texture) 등이 있다. 삼차원 가상 공간 객체는 하나 이상의 기본 객체들의 조합으로 형성된다. 따라서, 객체의 저장에 필요한 레코드 길이는 가상 공간 객체를 표현하기 위해 조합된 기본 객체의 수에 따라 가변적으로 변한다. 그림 1은 이에 대한 간단한 예이다.

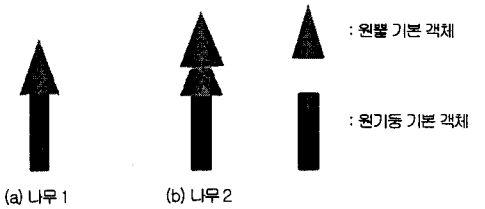


그림 1. 삼차원 가상 공간 객체의 형성 예

기존의 삼차원 가상 공간 객체 저작 도구에서는 순차 접근 방식을 사용한 파일 저장 시스템을 이용한다. 순차 접근 방식은 저장 공간에 있는 객체를 처음부터 또는 마지막부터 차례대로 읽어들여서 처리하기 때문에 객체 관리를 위한 복잡한 조작이 없어도 된다는 장점은 있으나 객체 삽입(insertion), 삭제(deletion), 갱신(update), 그리고 탐색(search) 처리의 경우에 처리 시간이 많이 걸리는 단점이 있다[11]. 그리고, 파일 저장 시스템을 이용할 경우, 사용자마다 자신의 특정한 응용을 위해서 객체를 별도로 정의하고 구현해야 한다. 이로 인해 객체의 중복 저장이 일어나게 되며 저장 공간의 낭비가 발생한다.

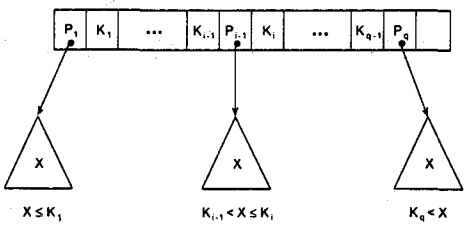
이러한 단점을 해결하기 위해서 데이타베이스 관리 시스템을 이용한다. 데이타베이스 관리 시스템은 사용자가 데이타베이스를 생성하고 관리할 수 있도록 편리한 기능을 제공하는 프로그램들의 모임이다[7]. 프로그램들이 제공하는 기능에는 객체의 중복성을 제어하는 기능과 여러 사용자가 객체를 공유할 수 있게 하는 기능 등이 있다.

관계형 데이타베이스 관리 시스템은 관계형 데이타베이스를 만들거나 수정하고 관리할 수 있게 해주는 프로그램으로서 1970년에 아이비엠(IBM)의 E. F. Codd[2]에 의해서 개발되었으며 일련의 정형화된 테이블로 구성된다. 그리고, 각 테이블은 고정 길이 레코드들의 집합으로 이루어진다.

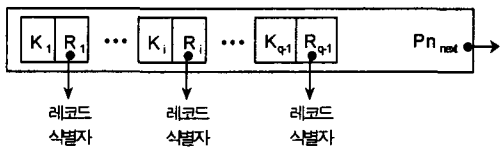
오픈 소스(open source) 형태로 널리 사용되어 지고 있는 MySQL[5, 6, 13]은 관계형 데이타베이스 관리 시스템중의 하나이다. MySQL의 저장 시스템은 데이타 페이지와 레코드를 관리하는 부분과 키를 사용하여 레코드를 액세스하는 부분 등으로 구성된다. 데이타 페이지와 레코드를 관리하는 부분은 페이지내의 저장 공간을 효율적으로 관리하고, 레코드의 삽입, 삭제 및 갱신 등의 기능을 제공한다. 그리고, 키를 사용하여 레코드를 액세스하는 부분은 보다 빠르게 원하는 레코드를 탐색하는 기능을 제공한다.

데이타베이스 저장 시스템에서 데이타의 직접적인 액세스를 위하여 가장 많이 사용되는 동적 인덱스 구조는 B^+-트리이다[8]. B^+-트리는 탐색 트리(search tree)의 일종인 B-트리를 변형한 형태로 레코드를 직접 가리키는 레코드 식별자(record identifier)가 모두 단말 노드에 있다[3, 7, 8]. B^+-트리는 내부노드(internal node)와 단말 노드(leaf node)로 구성된다. 내부 노드는 <트리 포인터, 키>의 집합으로, 단말 노드는 <키, 레코드 식별자>의 집합으로 구성된다.

그림 2는 차수(degree)가 p인 B^+-트리의 내부 노드와 단말 노드의 구조를 나타낸다. 내부 노드는 [$<P_1, K_1>$, $<P_2, K_2>$, ⋯, $<P_{q-1}, K_{q-1}>$, P_q]의 형태를 가지며, 여기에서 $q \le p$이고, 각 P_i는 트리 포인터이다. 각 내부 노드에서 키는 $K_1 < K_2 < \cdots < K_{q-1}$의 순이고, P_i가 가리키는 서브 트리에 있는 임의의 키 X는 $1 < i < q$일 때 $K_{i-1} < X \le K_i$이고, $i = 1$일 때 $X \le K_i$이고, $i = q$일 때 $K_{i-1} < X$이다. 그리고, 루트 노드를 제외한 각 내부 노드는 최소한 $\lceil P / 2 \rceil$ 개의 트리 포인터를 가지며 루트 노드가 내부 노드라면 최소한 두 개의 트리 포인터를 갖는다. 단말 노드는 [$<K_1, R_1>$, $<K_2, R_2>$, ⋯, $<K_{q-1}, R_{q-1}>$, Pn_{next}]의 형태를 가진다. 여기서 $q \le p$이고, 각 R_i는 레코드 식별자이고, Pn_{next}는 B^+-트리의 다음 단말 노드를 가리킨다. 각 단말 노드에서 키는 $K_1 < K_2 < \cdots < K_{q-1}$의 순이다. 그리고, 각 리프 노드는 최소한 $\lfloor P / 2 \rfloor$ 개의 값들을 가지며 모든 단말 노드는 같은 단계에 있다.



(a) 내부 노드



(b) 단말 노드

그림 2. B^+-트리의 내부 노드와 단말 노드 구조

3. 삼차원 가상 공간 객체 저장 시스템의 설계

본 장에서는 가변 길이 속성을 가지는 삼차원 가상 공간 객체를 효율적으로 저장하기 위한 저장 시스템의 구조를 기술한다. 그리고 삼차원 가상 공간 객체 저장 시스템에 저장되어있는 객체를 효율적으로 액세스할 수 있는 인덱스 구조를 기술한다.

테이블을 구성하는 페이지는 데이타 페이지(data page)와 프리 페이지(free page)로 구성된다. 그림 3은 데이타 페이지와 프리 페이지를 나타낸다. 데이타 페이지는 사용되고 있는 페이지로 페이지 내에서 현재 사용 가능한 영역의 시작 위치와 페이지 내에 포함된 레코드의 수를 유지한다. 그리고, 데이타 페이지를 상호 연결하기 위하여 양방향 링크 구조를 구성하는 두 개의 페이지 포인터를 가진다. 프리 페이지는 사용되지 않는 페이지로 단방향 링크 구조를 위한 페이지 포인터를 가진다.

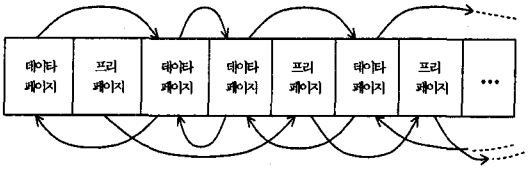


그림 3. 데이타 페이지와 프리 페이지

하나의 페이지에서 삼차원 가상 공간 객체를 저장하는 레코드는 페이지 내의 임의의 위치에 존재할 수 있으며 레코드의 길이는 가변적으로 변할 수 있다. 그림 4는 가변 길이 레코드를 위한 데이타 페이지의 구성을 나타낸다. 가변길이 속성을 지원하기 위하여 각 페이지마다 페이지 마지막에 슬롯 디렉토리(slot directory)를 두어 관리한다. 슬롯 디렉토리는 <레코드 오프셋(record offset), 레코드 길이(record length)>의 쌍으로 구성된다. 레코드 오프셋은 페이지 내의 사용 가능한 영역인 데이타 영역(data space)에서 레코드의 시작 위치를 나타낸다.

한 페이지 내에는 여러 개의 레코드가 존재할 수 있는

데 이들을 구별하기 위한 식별자로 레코드 아이디(record id: rid)를 사용한다. 레코드 아이디는 <페이지 아이디(page id), 슬롯 번호(slot number)>의 쌍으로 구성된다. 페이지 아이디는 레코드가 존재하는 페이지가 몇 번째 페이지인지 식별하기 위한 것이며, 슬롯번호는 슬롯 디렉토리의 몇 번째에 위치하는가를 나타낸다. 그리고, 자유 영역(free space)에 있는 슬롯 디렉토리 옆에 두 개의 저장 공간이 있다. 하나는 슬롯 디렉토리에 저장된 엔트리의 수를 나타내는 공간이며, 나머지 하나는 자유 영역의 시작을 가리키는 포인터를 저장하는 공간이다.

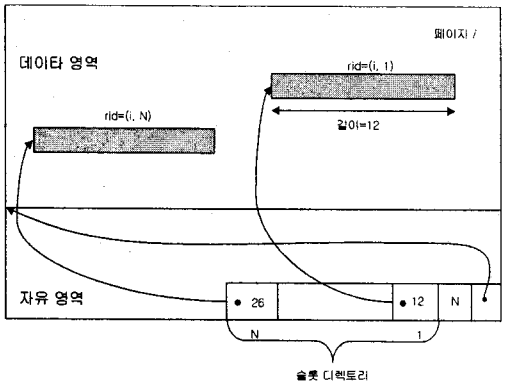


그림 4. 가변 길이 레코드를 위한 데이타 페이지의 구성

하나의 레코드는 여러 개의 필드(field)로 구성되며 필드의 크기도 역시 가변적으로 변하는 특징이 있다. 그림 5는 가변 길이 필드를 위한 레코드의 구성을 나타낸다. 레코드의 시작 부분에는 레코드를 구성하는 필드의 수를 저장하는 공간과 레코드에서 각 필드의 시작 위치를 가지고 있는 오프셋을 저장하기 위한 공간들이 있다. 그리고, 레코드의 끝을 가리키는 오프셋을 저장하기 위한 공간이 있는데 이곳에 저장되는 오프셋은 마지막 필드의 끝을 인식하기 위해서도 필요하다.

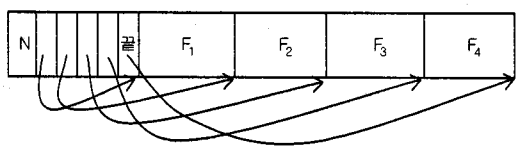


그림 5. 가변 길이 필드를 위한 레코드의 구성

삼차원 가상 공간 객체는 하나 이상의 기본 객체 조합으로 구성된다. 이렇게 구성된 삼차원 가상 공간 객체는 여러 개의 속성 값(attribute value)을 가지게 된다. 따라서, 하나의 속성 값을 가지는 키를 이용하여 객체를 탐색하는 것보다는 각각의 기본 객체가 가지는 속성 값들을 조합하여 구성한 다중키를 이용하여 객체를 탐색하는 것이 빠르고 효율적인 탐색을 가능하게 한다.

그림 6은 다중키를 지원하기 위한 키의 구성을 나타낸다. 하나의 레코드는 여러 개의 필드들로 구성되며, 이들 필드가 가지는 속성 값을 하나 이상 조합하여 다중키를 구성한다. 그리고, 키를 구성하는 속성 값 중 하나 이상의 길이가 가변적이면 키 또한 가변 길이 속성을 가진다.

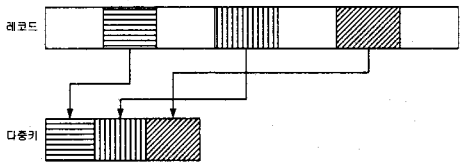
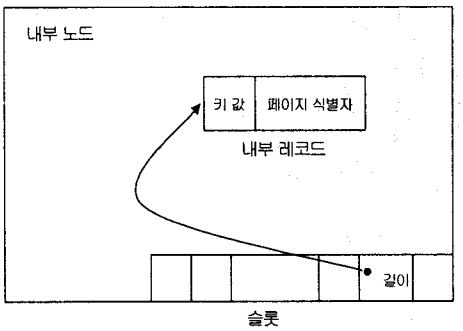
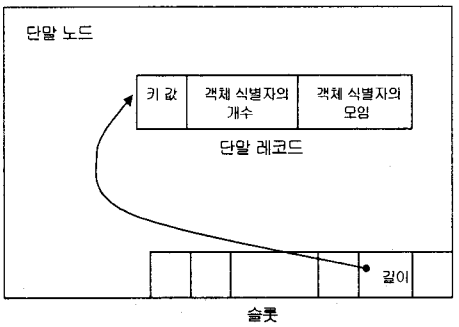


그림 6. 다중키를 지원하기 위한 키의 구성

그림 7은 내부 노드와 단말 노드의 인덱스 구성을 나타낸다. 키의 길이가 가변적이기 때문에 인덱스의 내부 노드에서도 데이타 페이지에서와 같이 슬롯을 사용한다. 구현한 인덱스 내부 노드의 구조는 <키, 트리 포인터>의 집합으로 구성되고, [$<K_1, P_1>$, $<K_2, P_2>$, …, $<K_{q-1}, P_{q-1}>$, P_q]의 형태를 가진다. 그리고, 내부 노드가 최소한2/3이상의 트리 포인터를 가지는 것을 제외한 나머지는 제 2절에서 기술한 B$^+$-트리와 동일하다. 단말 노드 역시 가변 길이 키를 지원하기 위하여 슬롯을 사용한다. 또한, 중복키를 허용하기 위하여 객체 식별자 부분은 동일한 키 값을 가지는 객체 식별자의 개수와 객체 식별자의 모임으로 이루어진다.



(a) 내부 노드와 내부 레코드의 구조



(b) 단말 노드와 단말 레코드의 구조

그림 7. 내부 노드와 단말 노드의 인덱스 구조

인덱스 알고리즘에서 기존의 B$^+$-트리는 각 노드에 데이타가 1/2 이상이 되도록 트리(tree)를 운영한다[3, 7]. 하지만, 본 논문에서는 각 노드에 데이타가 2/3 이상이 되도록 트리를 운영함으로써 인덱스 구조의 공간 사용률을 높일 수 있다.

4. 삼차원 가상 공간 객체 저장 시스템의 구현

본 논문에서는 삼차원 가상 공간 객체 저장 시스템의 구현을 위해서 인텔 펜티엄Ⅳ 시스템 사양에서 운영체제로 리눅스(Linux)의 한 종류인 레드햇(Redhat) 8.0 버전을 사용한다. 오픈 소스 형태의 관계형 데이타베이스 관리 시스템인 MySQL 4.0 버전을 참조하며, 이를 가변 길이 레코드 속성을 지원하도록 C 언어를 이용하여 변형한다. 그리고, 삼차원 가상 공간 저작 서버와의 연동을 위하여 JDBC(Java Database Connectivity)를 이용한다.

그림 8은 클라이언트에서 삼차원 가상 공간 객체를 저작하고 서버에 있는 삼차원 가상 공간 객체 저장 시스템에 저장하는 사용자 인터페이스이다. 객체 저장을 위한 사용자 인터페이스에는 객체를 저장할 경로를 지정하는 부분, 저장할 파일의 이름을 지정하는 부분, 그리고 파일에 대한 설명을 기술하는 부분으로 이루어진다.

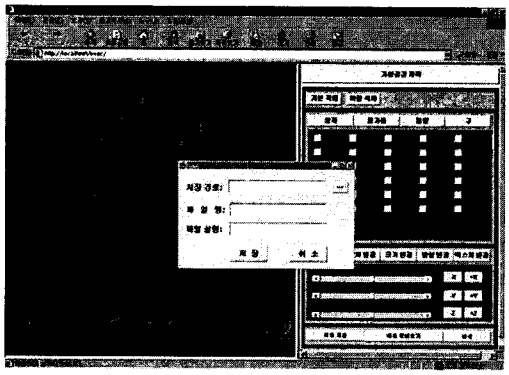


그림 8. 삼차원 가상 공간 객체 저장을 위한 사용자 인터페이스

그림 9는 클라이언트에서 서버의 삼차원 가상 공간 객체 저장 시스템에 저장되어있는 객체를 검색하고자할 때 이용하는 사용자 인터페이스이다. 객체의 검색을 위한 사용자 인터페이스는 객체가 가지는 속성 중에서 하나 이상의 속성을 조합하여 검색하는 기능을 제공한다.

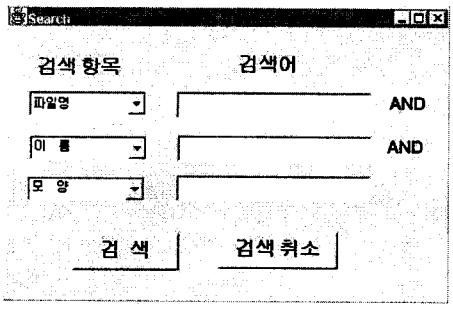


그림 9. 삼차원 가상 공간 객체 검색을 위한 사용자 인터페이스

5. 결론

본 논문에서는 가변 길이 속성을 가지는 삼차원 가상 공간 객체를 효율적으로 저장하고 관리할 수 있는 저장 시스템을 설계하고 구현하였다. 이를 위하여 관계형 데이타베이스 관리 시스템인 MySQL의 저장 시스템을 참조하고, 이를 변경하였다. MySQL의 저장 시스템은 가변 길이 속성을 효율적으로 지원하는 자료 구조를 가지고 있지 않으므로 가변 길이 속성을 지원하는 자료 구조를 설계하고 구현하였다. 가변 길이 속성을 지원하기 위하여 설계하고 구현한 부분은 페이지 구조, 레코드 구조, 그리고 필드의 구조이다. 이를 통해, 삼차원 가상 공간 객체의 저장 및 공유가 용이하게 된다.

삼차원 가상 공간 객체 저장 시스템에 있는 객체를 효율적으로 액세스하기 위하여 가변 길이 속성을 지원하며 공간 사용률을 높인 인덱스를 설계하고 구현하였다. 기존 B^+-트리는 다중키와 중복키를 지원하지 못하며, 각 노드에 데이타가 1/2 이상이 되도록 트리를 운영한다. 이를 하나 이상의 속성을 가진 다중키와 동일한 값을 다룰 수 있는 중복키를 허용하도록 노드의 구조를 변경하고, 가변 길이 속성을 지원하기 위하여 내부 노드와 단말 노드에서 슬롯을 이용하였다. 또, 각 노드에 데이타가 2/3 이상이 되도록 트리를 설계하고 구현하였다.

향후 연구과제로는 저장 시스템의 성능 향상을 위한 버퍼 관리와 보다 편리한 사용자 인터페이스의 개발에 관한 연구이다.

참고문헌

[1] Clifford So, George Baciu, and Hanqiu Sun, "Reconstruction of 3D virtual buildings from 2D architectural floor plans," *Proc. of the ACM symposium on Virtual reality software and technology*, pp. 17—23, Nov. 1998.

[2] Donald D. Chamberlin, "Relational Data-Base Management Systems," *ACM Computing Surveys (CSUR)*, Vol. 8, Issue 1, pp. 43—66, Jan. 1976.

[3] Jan Jannink, "Implementing Deletion in B^+-Trees," *ACM SIGMOD Record*, Vol. 24, No.1, pp.33—38, Mar. 1995.

[4] Ligos Technology co., *V-Realm Builder*, Ver. 2.1, 1999.

[5] MySQL AB., MySQL, Ver. 3.23, 2001.

[6] Paul DuBois and Michael Widenius, *MySQL*, New Riders Publishing, Dec.1999.

[7] Ramez A. Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison-Wesley, Aug. 1999.

[8] V. Srinivasan and Michael J. Carey, "Performance of B^+tree concurrency control algorithms," *The VLDB Journal*, Vol. 2, No. 4, pp. 361—406, Oct. 1993.

[9] 고영덕, *3차원 멀티미디어 홈페이지로의 도전 VRML2.0*, 혜지원, 1998년 7월.

[10] 문양세, *가변길이 속성을 지원하는 저장 시스템의 설계 및 구현*, 석사학위논문, 한국과학기술원 전산학과, 1993년, 2월.

[11] 주낙근 외, *C++로 구현한 자료구조론*, 정익사, 1999년 5월.

[12] 최홍석, *다중 참여형 가상공간 구현에 관한 연구*, 석사학위논문, 동아대학교 컴퓨터공학과, 1998년 8월.

[13] 허정수, *아주 특별한 웹데이타베이스 MySQL & Web DB 연동*, 베스트북, 2000년 11월.