

# Oracle DB

KarL

# 목차

1. 데이터베이스와 데이터베이스 시스템
2. 데이터베이스 시스템의 발전
3. 파일 시스템과 DBMS
4. 데이터베이스 시스템의 구성

# 학습목표

- 데이터베이스의 유형을 알아보고 개념 및 특징을 이해한다.
- 데이터베이스 시스템을 중심으로 한 정보 시스템의 발전 과정을 알아본다.
- 프로그램과 데이터가 컴퓨터에 어떻게 저장되는지 이해한다.
- 데이터베이스 시스템의 구성 요소를 알아본다.

# 01. 데이터베이스와 데이터베이스 시스템

---

- 데이터, 정보, 지식
- 일상생활의 데이터베이스
- 데이터베이스의 개념 및 특징
- 데이터베이스 시스템의 구성

# 1. 데이터, 정보, 지식

- 데이터 : 관찰의 결과로 나타난 정량적 혹은 정성적인 실제 값
- 정보 : 데이터에 의미를 부여한 것
- 지식 : 사물이나 현상에 대한 이해



그림 1-1 데이터, 정보, 지식

## 2. 일상생활의 데이터베이스

- 데이터베이스 : 조직에 필요한 정보를 얻기 위해 논리적으로 연관된 데이터를 모아 구조적으로 통합해 놓은 것

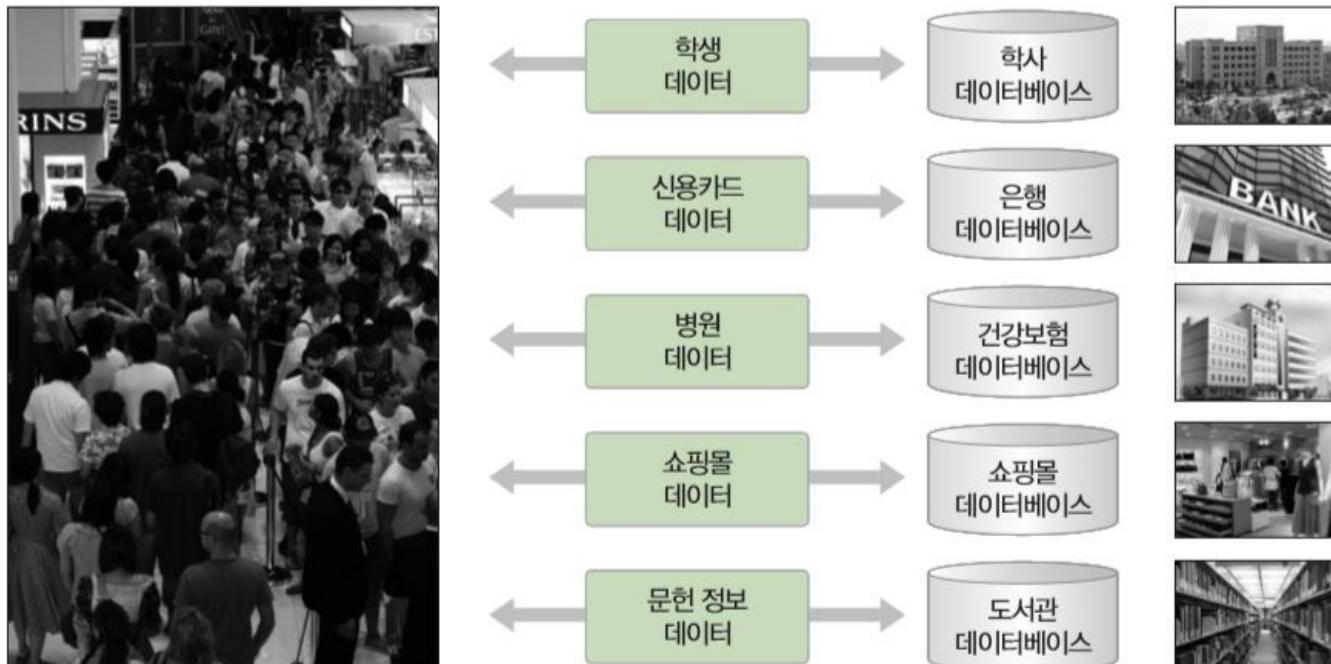


그림 1-2 일상생활에서 생성되는 데이터베이스

## 2. 일상생활의 데이터베이스

#### ■ 간단한 거래도 많은 데이터가 포함



그림 1-3 패스트푸드 체인점 데이터베이스

## 2. 일상생활의 데이터베이스

- 데이터베이스 시스템은 데이터의 검색과 변경 작업을 주로 수행함
- 변경이란 시간에 따라 변하는 데이터 값을 데이터베이스에 반영하기 위해 수행하는 삽입, 삭제, 수정 등의 작업을 말함

표 1-1 검색과 변경 빈도에 따른 데이터베이스 유형

구축이 쉬움 ↑ ↓ 구축이 어려움	유형	검색 빈도	변경 빈도	특징
	유형 1	적다	적다	<ul style="list-style-type: none"><li>• 검색이 많지 않아 데이터베이스를 구축할 필요 없음</li><li>• 보존가치가 있는 경우에 구축</li></ul> <p>예) 공룡 데이터베이스</p>
	유형 2	많다	적다	<ul style="list-style-type: none"><li>• 사용자 수 보통</li><li>• 검색은 많지만, 데이터에 대한 변경은 적음</li></ul> <p>예) 도서 데이터베이스</p>
	유형 3	적다	많다	<ul style="list-style-type: none"><li>• 예약 변경/취소 등 데이터 변경은 많지만, 검색은 적음</li><li>• 실시간 검색 및 변경이 중요함</li></ul> <p>예) 비행기 예약 데이터베이스</p>
	유형 4	많다	많다	<ul style="list-style-type: none"><li>• 사용자 수 많음</li><li>• 검색도 많고, 거래로 인한 변경도 많음</li></ul> <p>예) 증권 데이터베이스</p>

## 2. 일상생활의 데이터베이스

---

### ① 통합된 데이터(integrated data)

데이터를 통합하는 개념으로, 각자 사용하던 데이터의 중복을 최소화하여 중복으로 인한 데이터 불일치 현상을 제거

### ② 저장된 데이터(stored data)

문서로 보관된 데이터가 아니라 디스크, 테이프 같은 컴퓨터 저장장치에 저장된 데이터를 의미

### ③ 운영 데이터(operational data)

조직의 목적을 위해 사용되는 데이터를 의미한다. 즉 업무를 위한 검색을 할 목적으로 저장된 데이터

### ④ 공용 데이터(shared data)

한 사람 또는 한 업무를 위해 사용되는 데이터가 아니라 공동으로 사용되는 데이터를 의미

### 3. 데이터베이스의 개념

- 데이터베이스는 운영 데이터를 통합하여 저장하며 공용으로 사용

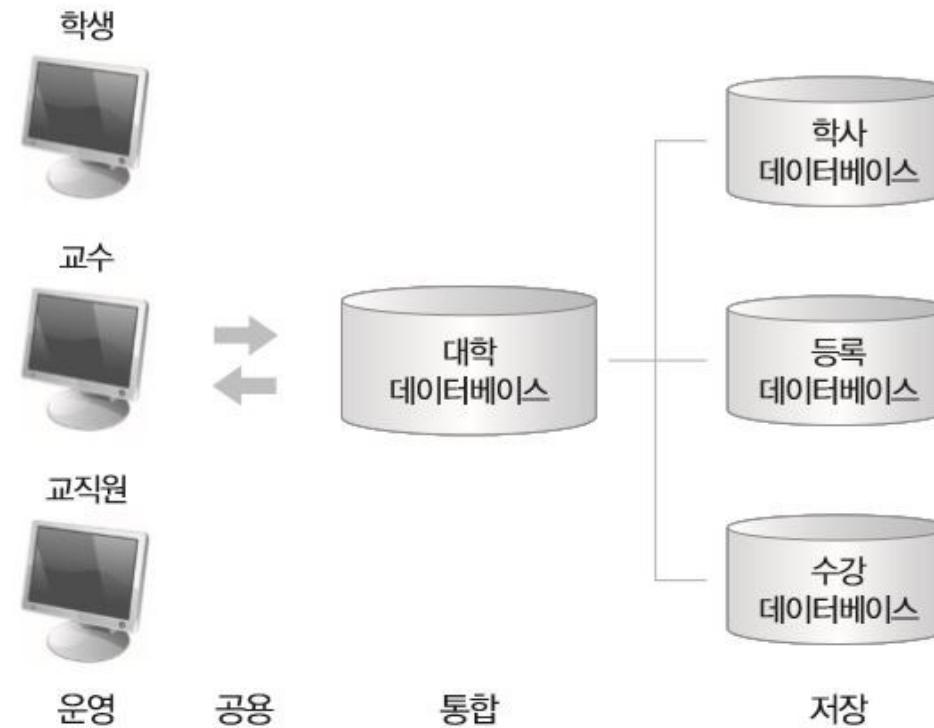


그림 1-4 데이터베이스의 개념

### 3. 데이터베이스의 특징

#### ① 실시간 접근성(real time accessibility)

데이터베이스는 실시간으로 서비스, 사용자가 데이터를 요청하면 몇 시간이나 몇 일 뒤에 결과를 전송하는 것이 아니라 수 초 내에 결과를 서비스

#### ② 계속적인 변화(continuous change)

데이터베이스에 저장된 내용은 어느 한 순간의 상태를 나타내지만, 데이터 값은 시간에 따라 항상 바뀜. 데이터베이스는 삽입(insert), 삭제(delete), 수정(update) 등의 작업을 통하여 바뀐 데이터 값을 저장

#### ③ 동시 공유(concurrent sharing)

데이터베이스는 서로 다른 업무 또는 여러 사용자에게 동시에 공유동시(concurrent)는 병행이라고도 하며, 데이터베이스에 접근하는 프로그램이 여러 개 있다는 의미

#### ④ 내용에 따른 참조(reference by content)

데이터베이스에 저장된 데이터는 데이터의 물리적인 위치가 아니라 데이터 값에 따라 참조

## 4. 데이터베이스 시스템의 구성

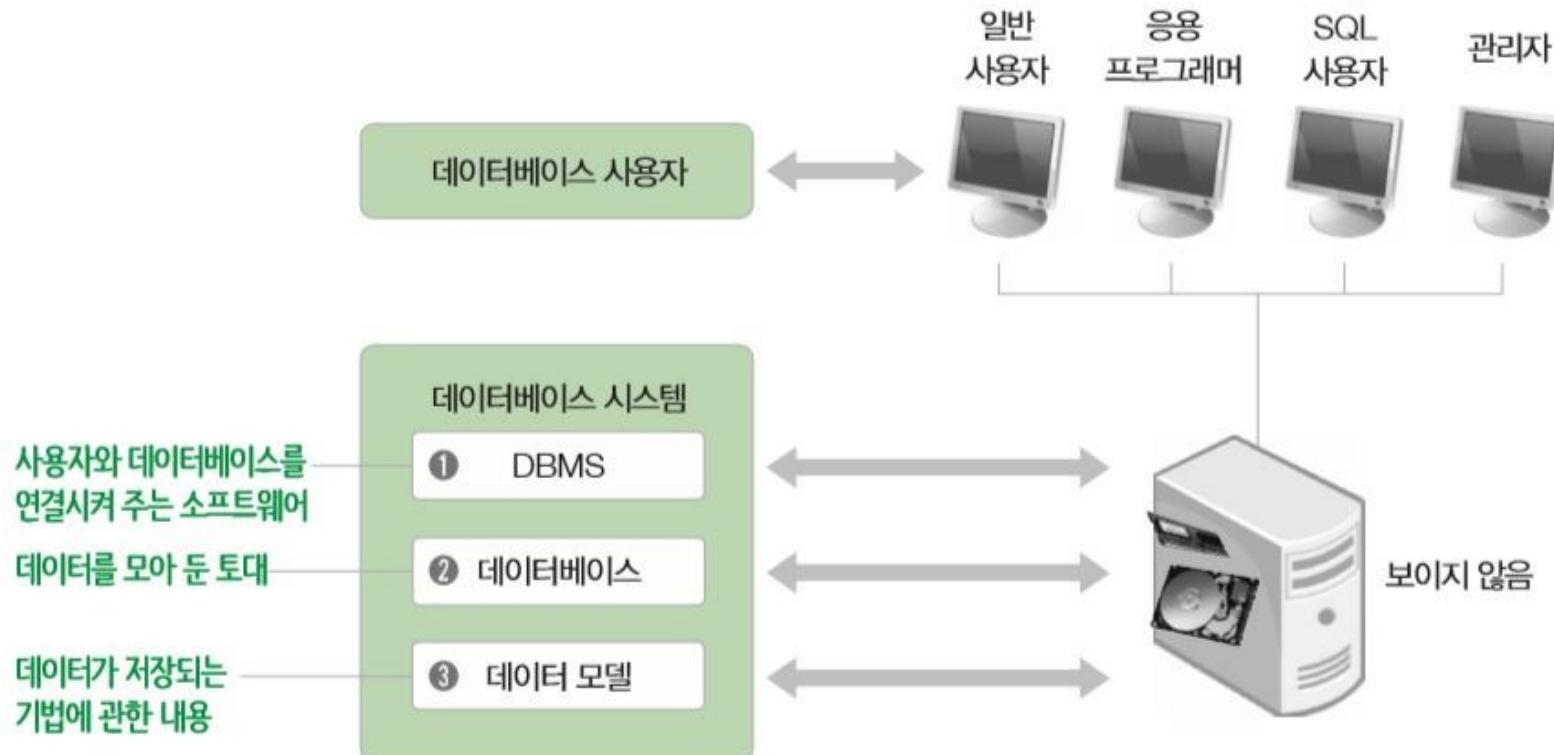


그림 1-5 데이터베이스 시스템의 구성 요소와 물리적인 위치

## 02. 데이터베이스 시스템의 발전

---

- 마당서점과 데이터베이스 시스템
- 정보 시스템의 발전

# [1단계] 마당서점의 시작



- 도서 : 100권
- 고객 : 근처 학교의 학생, 지역 주민
- 고객 서비스 : 사장이 직접 도서 안내
- 업무 : 회계업무(계산기 사용), 장부에 기록

그림 : 마당서점 초기

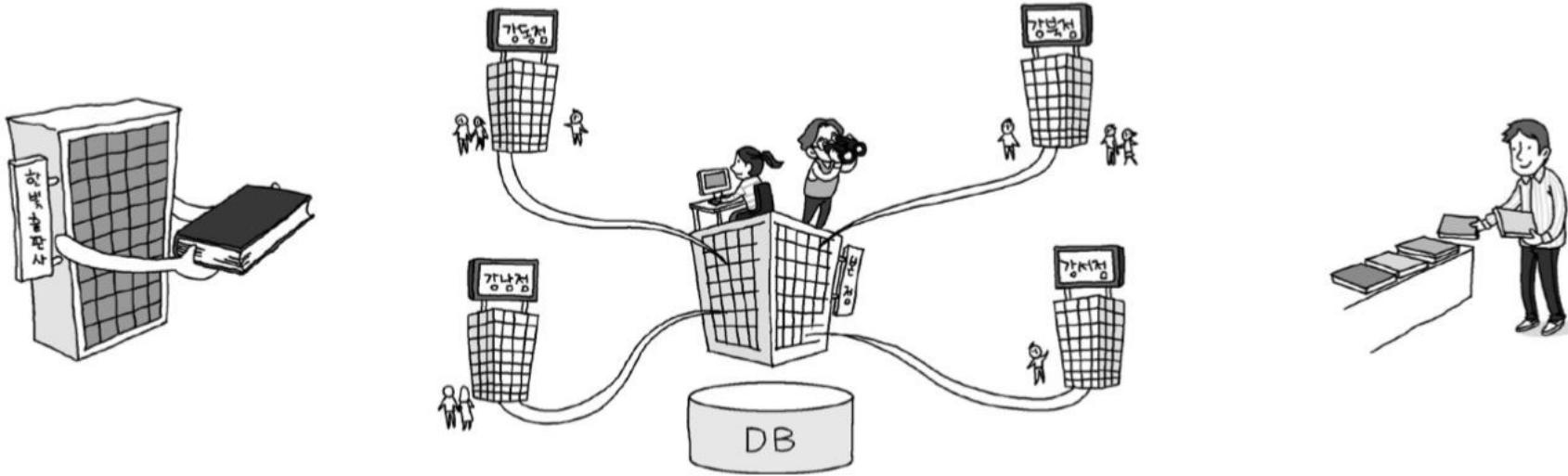
## [2단계] 컴퓨터의 도입



- 도서 : 1000권
- 고객 : 근처 학교의 학생, 지역 주민
- 고객 서비스 : 컴퓨터를 이용하여 도서 검색, 직원 고용
- 업무 : 회계업무(컴퓨터 사용), 파일 시스템

그림 : 마당서점 초기 전산화(+컴퓨터)

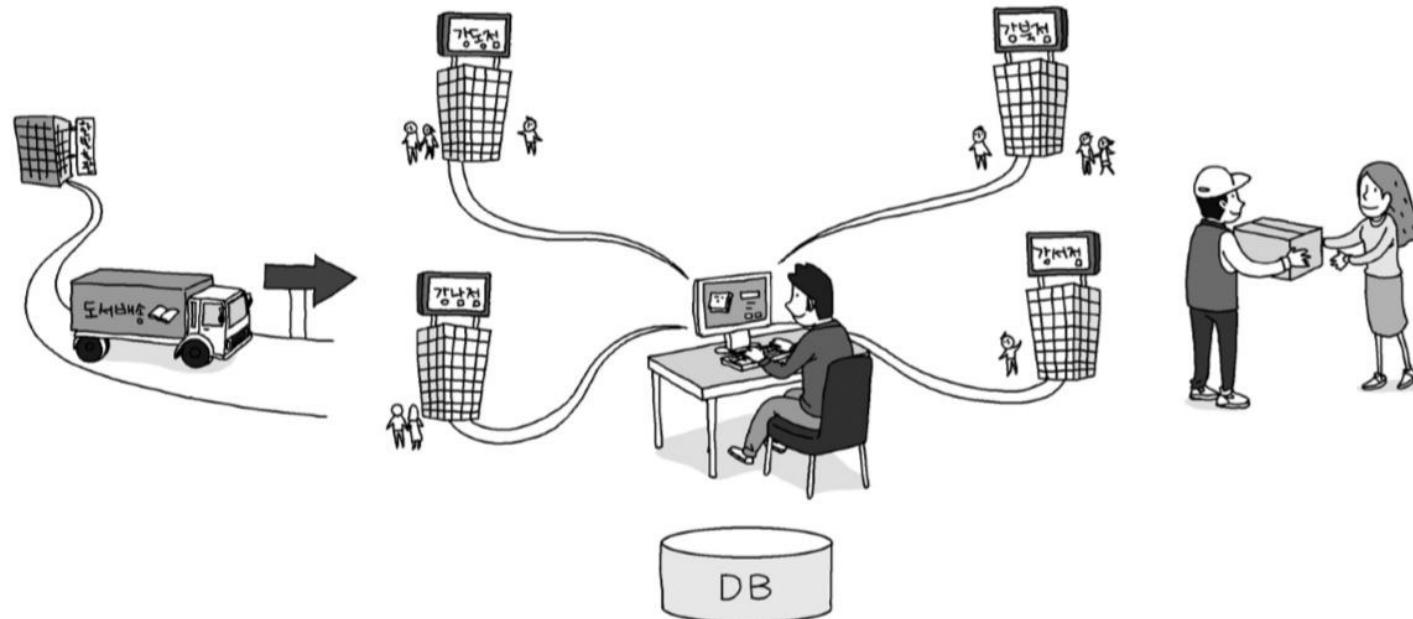
## [3단계] 지점 개설 및 데이터베이스 구축



- 도서 : 10,000권
- 고객 : 서울 지역 고객
- 고객 서비스 : 클라이언트/서버 시스템으로 지점을 연결하여 도서 검색 서비스 제공
- 업무 : 회계업무(컴퓨터 사용), 데이터베이스 시스템 도입

그림 : 마당서점 데이터베이스(DBMS) 도입(+원격통신)

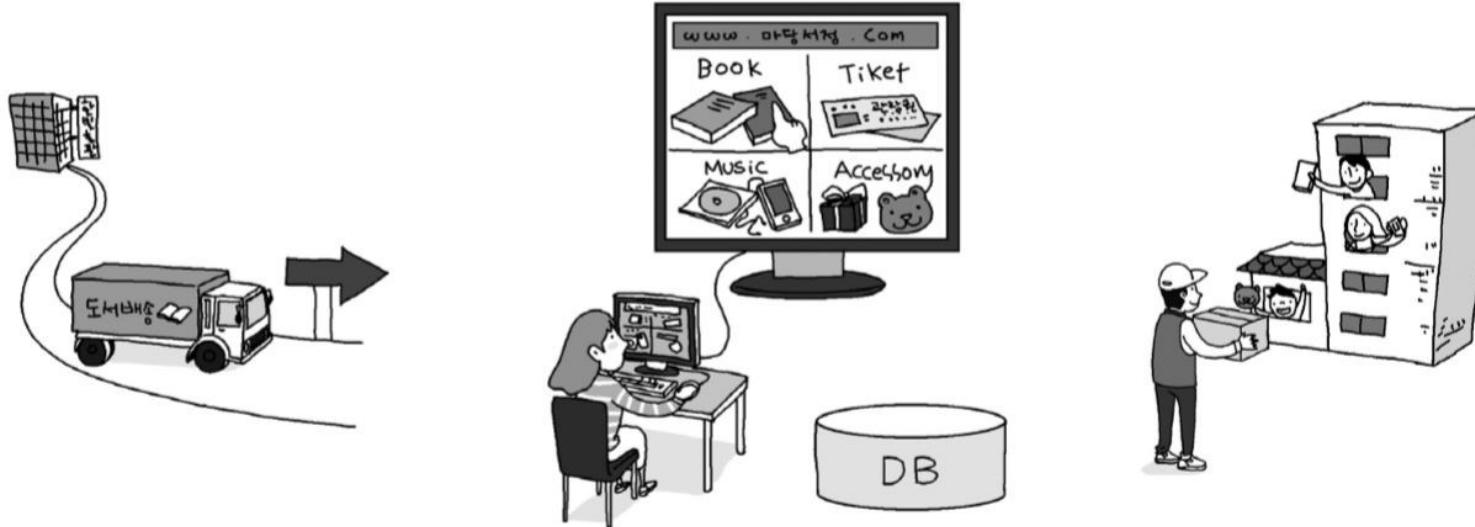
## [4단계] 홈페이지 구축



- 도서 : 100,000권
- 고객 : 국민(전국으로 배송)
- 고객 서비스 : 인터넷으로 도서 검색 및 주문
- 업무 : 회계/인사업무(컴퓨터와 인터넷 사용), 웹 DB 시스템으로 지점 간 연계

그림 : 마당서점 홈페이지 구축(+인터넷)

## [5단계] 인터넷 쇼핑몰 운영



- 도서 : 1,000,000권
- 고객 : 국민(전국으로 배송)
- 고객 서비스 : 인터넷 종합 쇼핑 서비스 제공
- 업무 : 회계/인사업무(컴퓨터와 인터넷 사용), DB 서버 여러 개 구축

그림 : 마당서점 인터넷 쇼핑몰 운영

# 1. 마당서점과 데이터베이스 시스템

표 1-2 정보통신기술의 발전과 마당서점의 성장

단계	시기		주요 특징
	정보통신기술		
1단계 마당서점	1970년대		<ul style="list-style-type: none"><li>사장이 모든 도서의 제목과 가격을 기억</li><li>매출과 판매가 컴퓨터 없이 관리됨</li><li>매출에 대한 내용이 정확하지 않음</li></ul>
	없음		
2단계 초기 전산화	1980년대		<ul style="list-style-type: none"><li>컴퓨터를 이용한 초기 응용 프로그램으로 업무 처리</li><li>파일 시스템 사용</li><li>한 대의 컴퓨터에서만 판매 및 매출 관리</li></ul>
	컴퓨터		
3단계 데이터베이스 구축	1990년대		<ul style="list-style-type: none"><li>지점 간 클라이언트/서버 시스템을 도입하여 업무 처리</li><li>데이터베이스 관리 시스템(DBMS) 도입</li></ul>
	컴퓨터+원격통신		
4단계 홈페이지 구축	2000년대		<ul style="list-style-type: none"><li>인터넷을 이용하여 도서 검색 및 주문</li><li>웹 DB 시스템으로 불특정 다수 고객 유치</li><li>고객이 지리적으로 넓게 분산됨</li></ul>
	컴퓨터+인터넷		
5단계 인터넷 쇼핑몰	2010년대		<ul style="list-style-type: none"><li>도서뿐만 아니라 음반, 액세서리, 문구, 공연 티켓까지 판매하는 인터넷 쇼핑몰로 확대</li><li>도서 외 상품의 매출 비중이 50% 이상으로 늘어남</li></ul>
	컴퓨터+인터넷 +스마트폰		

## 2. 정보 시스템의 발전

### ① 파일 시스템

- 데이터를 파일 단위로 파일 서버에 저장
- 각 컴퓨터는 LAN을 통하여 파일 서버에 연결되어 있고, 파일 서버에 저장된 데이터를 사용하기 위해 각 컴퓨터의 응용 프로그램에서 열기/닫기(open/close)를 요청
- 각 응용 프로그램이 독립적으로 파일을 다루기 때문에 데이터가 중복 저장될 가능성이 있음
- 동시에 파일을 다루기 때문에 데이터의 일관성이 훼손될 수 있음

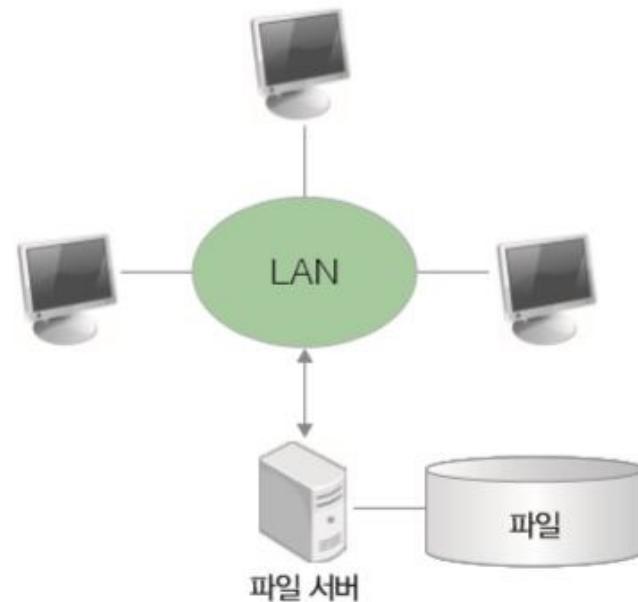


그림 : 파일 시스템

## 2. 정보 시스템의 발전

### ② 데이터베이스 시스템

- DBMS를 도입하여 데이터를 통합 관리하는 시스템
- DBMS가 설치되어 데이터를 가진 쪽을 서버(server), 외부에서 데이터 요청하는 쪽을 클라이언트(client)라고 함
- DBMS 서버가 파일을 다루며 데이터의 일관성 유지, 복구, 동시 접근 제어 등의 기능을 수행
- 데이터의 중복을 줄이고 데이터를 표준화하며 무결성을 유지

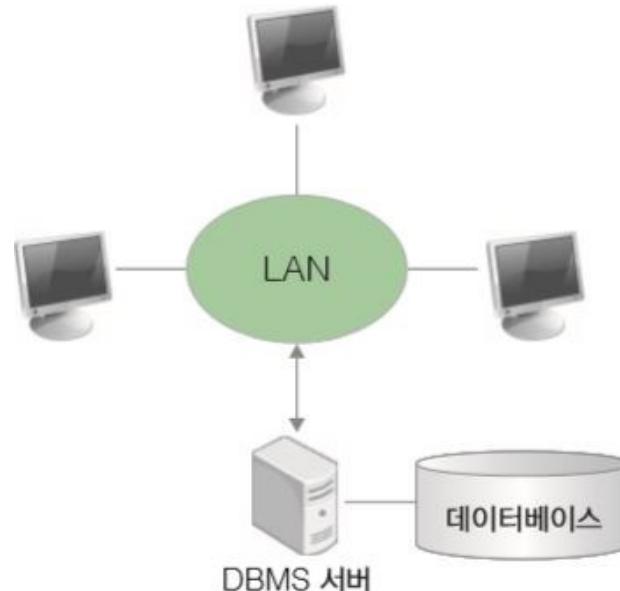


그림 : 데이터베이스 시스템

## 2. 정보 시스템의 발전

### ③ 웹 데이터베이스 시스템

- 데이터베이스를 웹 브라우저에서 사용할 수 있도록 서비스하는 시스템
- 불특정 다수 고객을 상대로 하는 온라인 상거래나 공공 민원 서비스 등에 사용됨

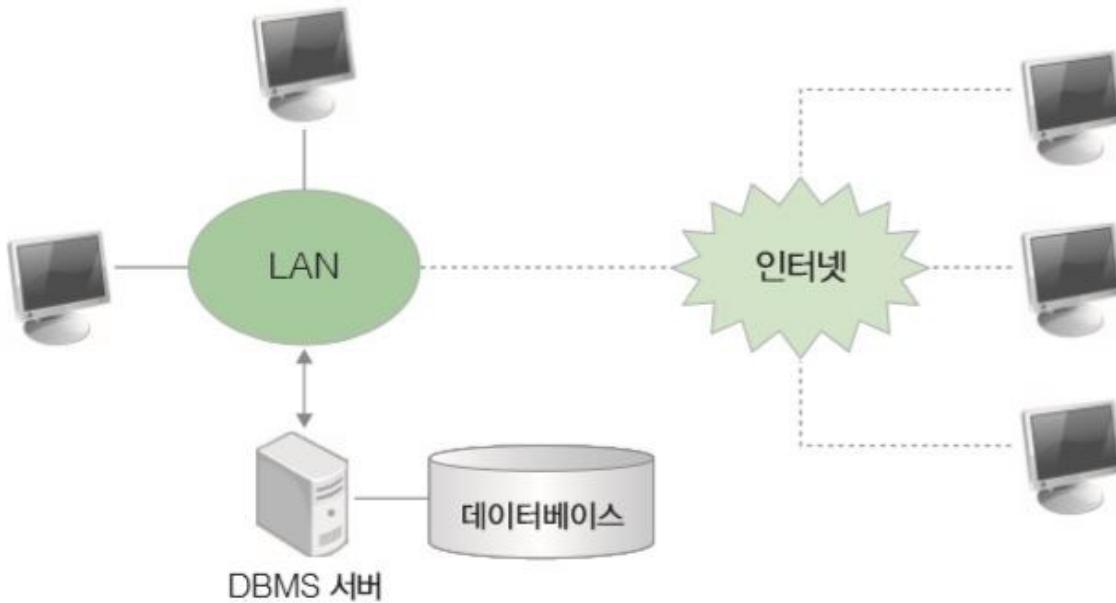


그림 : 웹 데이터베이스 시스템

## 2. 정보 시스템의 발전

### ④ 분산 데이터베이스 시스템

- 여러 곳에 분산된 DBMS 서버를 연결하여 운영하는 시스템
- 대규모의 응용 시스템에 사용됨

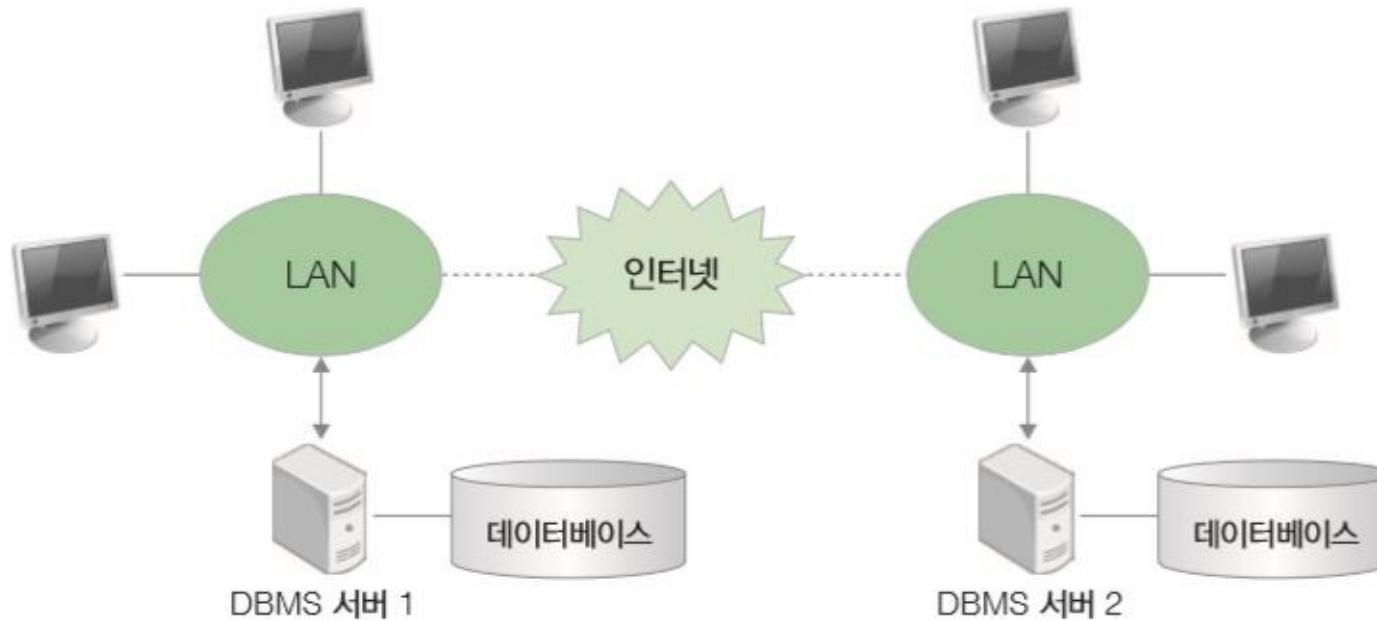
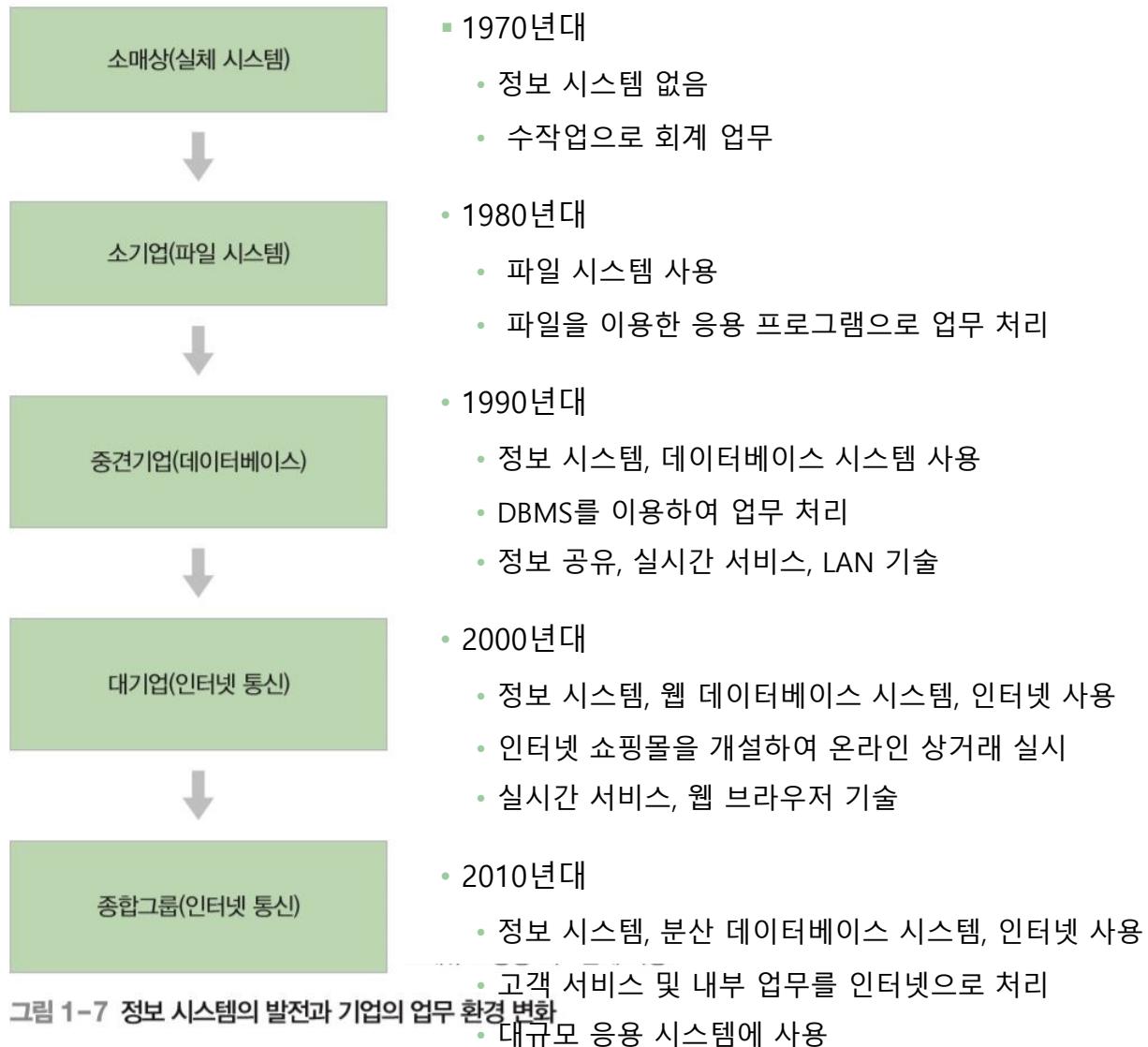


그림 : 분산 데이터베이스 시스템

## 2. 정보 시스템의 발전



## 03. 파일 시스템과 DBMS

---

- 마당서점 데이터를 저장하는 방법
- 마당서점 데이터의 저장 방법 비교
- 파일 시스템과 DBMS의 비교

# 1. 마당서점 데이터를 저장하는 방법

## [프로그램 1]

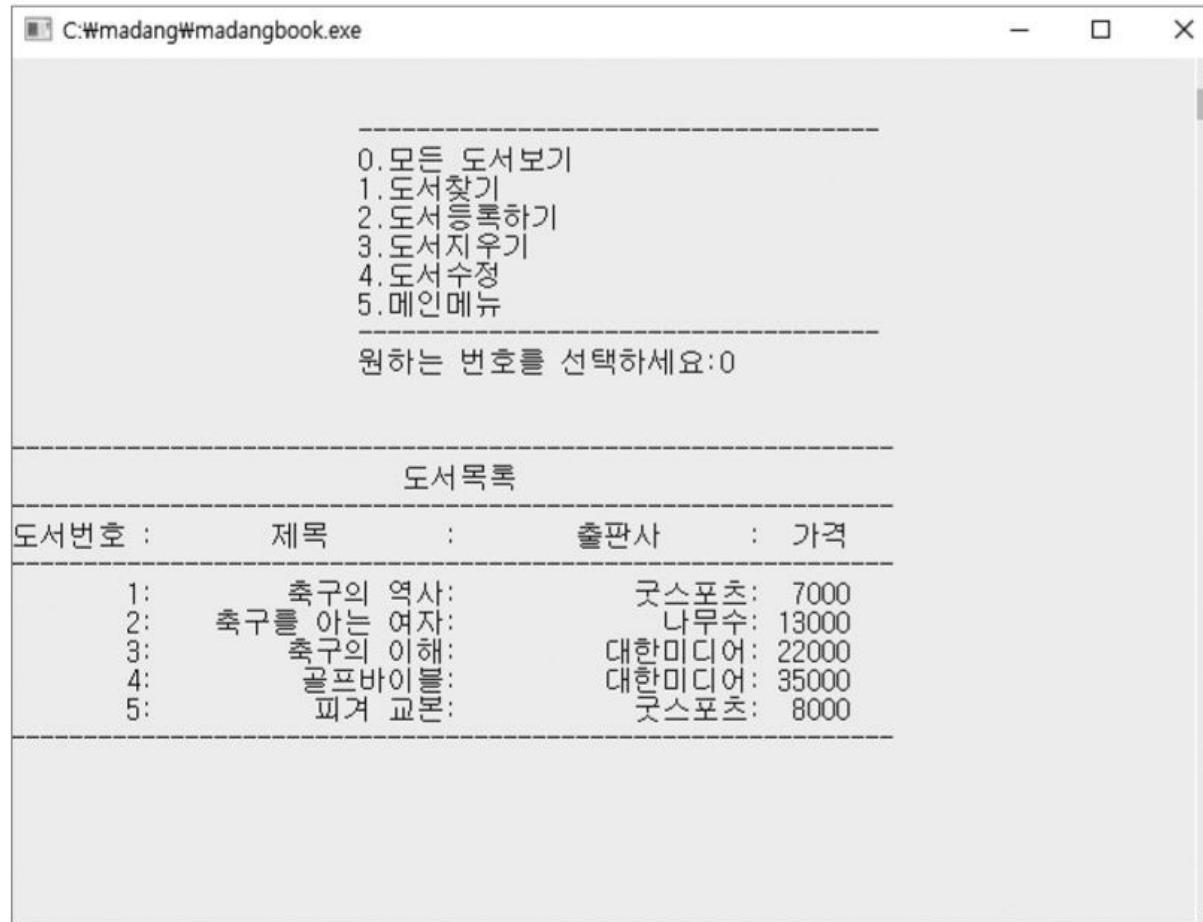


그림 1-8 도서 검색 프로그램

# 1.1 데이터를 프로그램 내부에 저장하는 방법

## [프로그램 1]

- 데이터를 프로그램 내부에 저장
  - C 언어의 구조체 BOOK을 먼저 선언하고 main() 프로그램에서 구조체 배열 변수 BOOKS[ ]에
  - 데이터를 저장
  - 도서 데이터는 프로그램 내 구조체 변수에 저장됨
- 문제점 : 새로운 데이터가 생길 때마다 프로그램을 수정한 후 다시 컴파일해야 함

### ■ 프로그램 1: 데이터를 프로그램 내부에 저장

### 소스코드

```
/* BOOK 데이터 구조 정의 */
typedef struct {
    int bookid[5];
    char bookname[20];
    char publisher[20];
    int price;
} BOOK;

int main() {
    BOOK BOOKS[10]; /* 구조체 배열 변수에 데이터 저장 */

    /* 첫 번째 도서 저장 */
    BOOKS[1].bookid = 1;
    strcpy(BOOKS[1].bookname, "축구의 역사");
    strcpy(BOOKS[1].publisher, "굿스포츠");
    BOOKS[1].price = 7000;

    /* 두 번째 도서 저장 */
    BOOKS[2].bookid = 2;
    strcpy(BOOKS[2].bookname, "축구 아는 여자");
    strcpy(BOOKS[2].publisher, "나무수");
    BOOKS[2].price = 13000

    /* 나머지 다른 도서 저장 ( 생략 ) */
    .....

    /* 모든 도서보기 프로그램 호출 */
    search_all();

    /* 기타 프로그램 코드 */
    .....
}
```

## 1.2 파일 시스템을 사용하는 방법

### ■ [프로그램 2]



그림 1-9 도서 검색 프로그램에서 도서를 등록하는 화면

# 1.2 파일 시스템을 사용하는 방법

## [프로그램 2]

- 데이터를 프로그램 내부에 저장
  - BOOK 데이터 구조를 먼저 선언하고  
main() 프로그램에서 파일로부터 데이터를  
불러와 구조체 배열 변수 BOOKS[]에 저장
  - 새로운 데이터가 추가되어도  
프로그램을 수정할 필요 없음
  - 문제점 : 같은 파일을 두 개의 프로그램이 공유하는  
것이 운영체제의 도움없이 불가능

■ 프로그램 2: 데이터를 파일에 저장

소스코드

```
/* BOOK 데이터 구조 정의 */
typedef struct {
    int     bookid[5];
    char   bookname[20];
    char   publisher[20];
    int    price;
} BOOK;

int main() {
    BOOK BOOKS[10];
    int i = 1;
    insert(); /* 도서 입력 함수 */

    /* 파일에 저장된 데이터를 배열 BOOKS[]에 저장 */
    fp = fopen("book.dat","rb");
    bp = (BOOK *)calloc(1,sizeof(BOOK));
    /* 파일에서 책을 읽는다 */
    while(fread(bp, sizeof(BOOK), 1, fp) != 0) {
        BOOKS[i].bookid = bp->bookid;
        strcpy(BOOKS[i].bookname, bp->bookname);
        strcpy(BOOKS[i].publisher, bp->publisher);
        BOOKS[i].price = bp->price;
        i++;
    }

    /* 모든 도서보기 프로그램 호출 */
    search_all();

    /* 기타 프로그램 코드 */
    ....
}
```

# 1.3 DBMS를 사용하는 방법

## [프로그램 3]

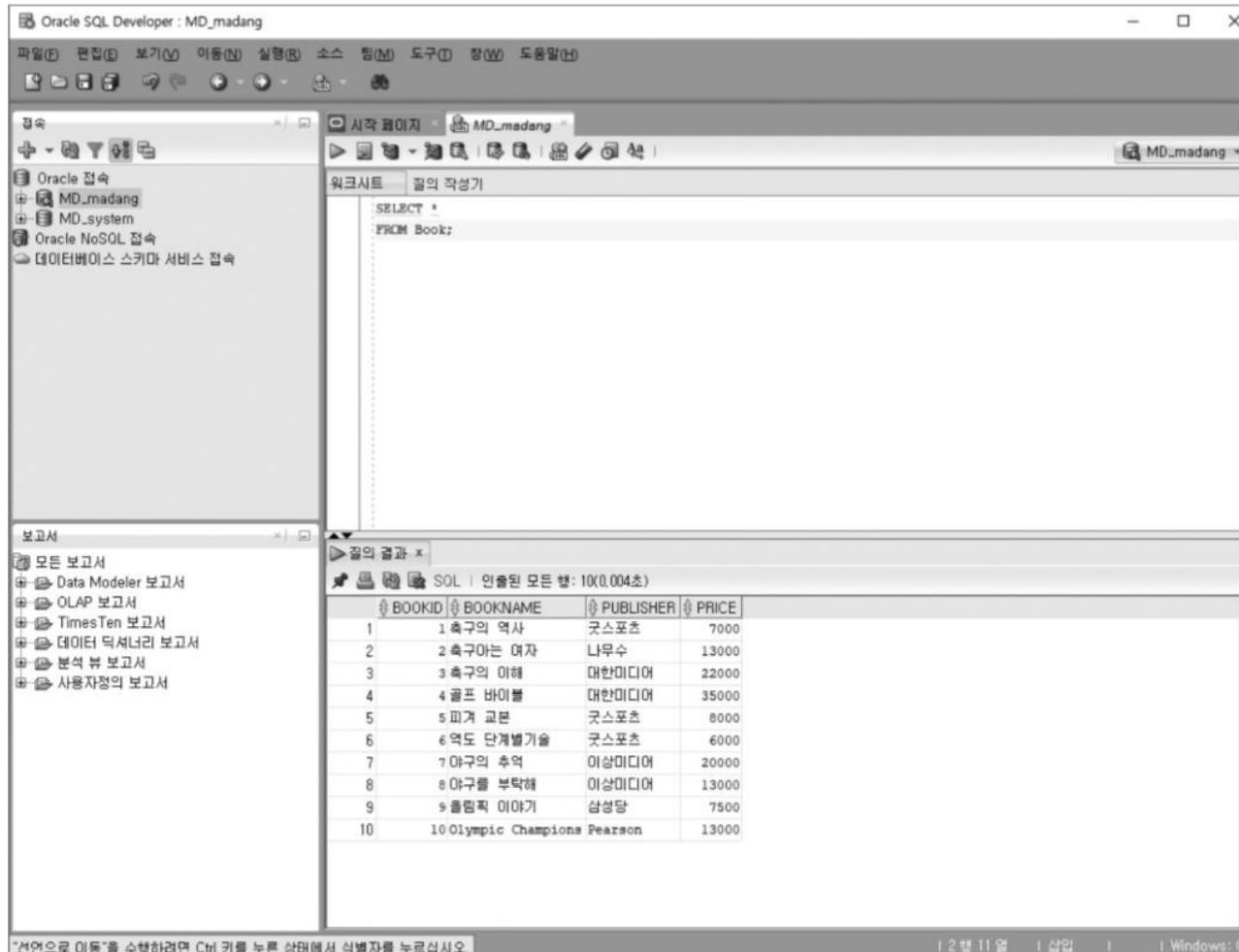


그림 1-10 오라클(SQL Developer)의 데이터베이스 관리 화면

# 1.3 DBMS를 사용하는 방법

## [프로그램 3]

- 데이터를 DBMS 내부에 저장
  - 데이터 정의와 데이터 값을 DBMS가 관리
  - DBMS는 데이터 정의, 데이터 변경 등의 작업을 할 수 있는 별도의 프로그램을 갖고 있음
  - 프로그램에 데이터 정의나 데이터 값을 포함하지 않기 때문에 데이터 구조가 바뀌어도 다시 컴파일할 필요가 없음

### ■ 프로그램 3: 데이터를 DBMS에 저장

### 소스코드

```
int main() {
    /* 반환된 행의 수 */
    int num_ret;
    /* DBMS에 접속 */
    EXEC SQL CONNECT :username IDENTIFIED BY :password;

    /* SQL 문 실행 */
    EXEC SQL DECLARE c1 CURSOR FOR
        SELECT bookname, publisher, price FROM BOOK;
    EXEC SQL OPEN c1;

    /* 모든 도서보기 프로그램 호출 */
    search_all();

    /* SQL 문 실행 결과 출력 */
    for (;;) {
        EXEC SQL FETCH c1 INTO :BOOK_rec;
        print_rows(num_ret);
    }
    EXEC SQL CLOSE c1;

    /* 접속 해제 */
    EXEC SQL COMMIT WORK RELEASE;
}
```

## 2. 마당서점 데이터의 저장 방법 비교

### ■ [프로그램 1]

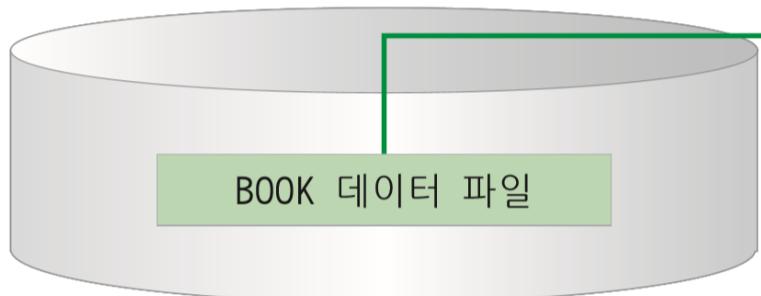
```
{  
    BOOK 데이터 타입 선언;  
  
    BOOK 데이터 구조  
  
    프로그램 내에서  
    BOOKS[] 배열에 데이터 저장;  
  
    BOOK 데이터  
  
    ...  
    검색 및 데이터 변경 프로그램 수행;  
}
```

- 프로그램에 데이터 정의와 데이터 값을 모두 포함하는 방식
- 프로그램에 BOOK 데이터 구조를 정의하고 데이터 값도 직접 변수에 저장함
- 데이터 구조 혹은 데이터 값이 바뀌면 프로그램을 다시 컴파일해야 함

## 2. 마당서점 데이터의 저장 방법 비교

### [프로그램 2]

```
{  
    BOOK 데이터 타입 선언;  
  
    BOOK 데이터 구조  
  
    파일로부터 데이터를 불러와  
    BOOKS[] 배열에 저장;  
    ...  
    검색 및 데이터 변경 프로그램 수행;  
}
```

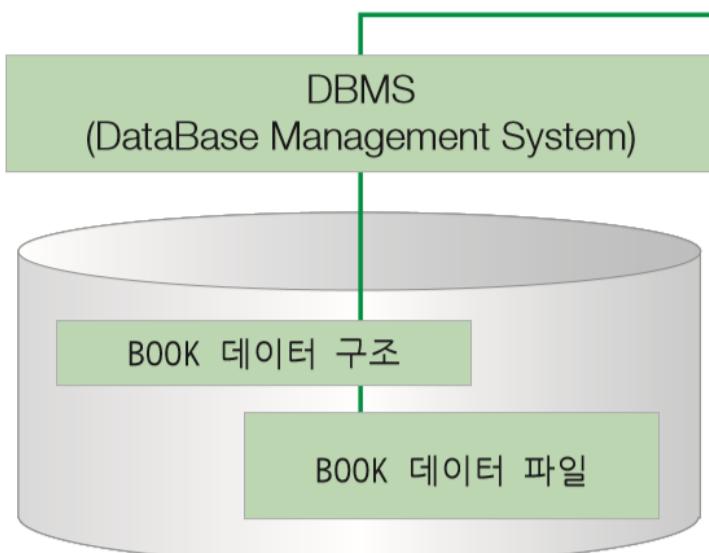


- 파일에 데이터 값, 프로그램에 데이터 정의를 포함하는 방식
- 프로그램에 BOOK 데이터 구조만 정의하고, 데이터 값은 book.dat라는 파일에 저장됨
- 데이터 값이 바뀌면 프로그램에 변경이 없지만, 데이터 구조가 바뀌면 프로그램을 다시 컴파일해야 함

## 2. 마당서점 데이터의 저장 방법 비교

### [프로그램 3]

```
{  
    /* BOOK 데이터 타입 선언 필요 없음 */  
  
    SQL 문을 실행하여 결과를 가져옴;  
    ...  
    SQL 문으로 데이터 변경;  
}
```



- DBMS가 데이터 정의와 데이터 값을 관리하는 방식
- BOOK 데이터 구조는 DBMS가 관리하고, 데이터 값은 데이터베이스에 저장됨
- 데이터 값이 바뀌거나 [데이터 구조](#)가 바뀌어도 프로그램을 다시 컴파일할 필요 없음

### 3. 파일 시스템과 DBMS의 비교

표 1-4 파일 시스템과 DBMS의 비교

구분	파일 시스템	DBMS
데이터 정의	응용 프로그램	DBMS
데이터 저장	파일 시스템	데이터베이스
데이터 접근 방법	응용 프로그램이 파일에 직접 접근	응용 프로그램이 DBMS에 파일 접근을 요청
사용 언어	자바, C++, C 등	자바, C++, C 등과 SQL
CPU/주기억장치 사용	적음	많음

### 3. 파일 시스템과 DBMS의 비교



그림 1-11 파일 시스템으로 구축된 구매 및 판매 응용 프로그램

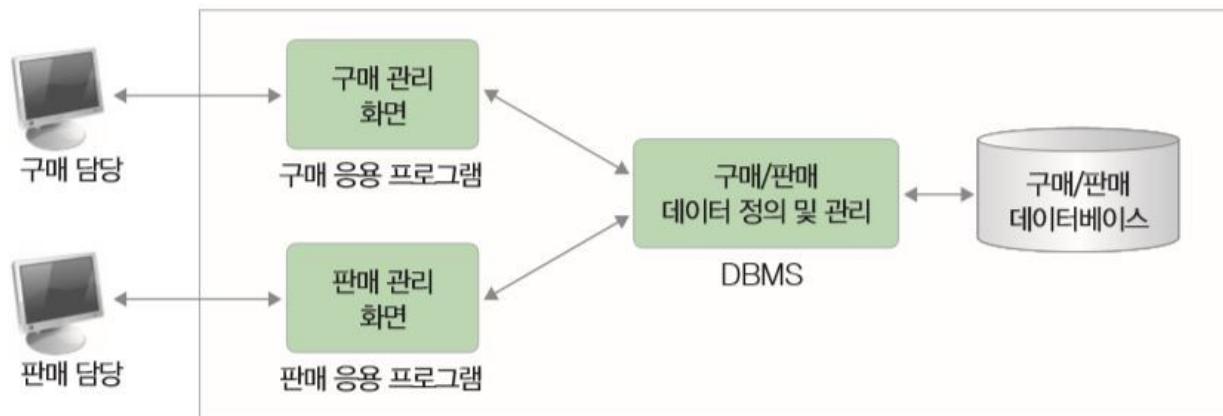


그림 1-12 DBMS로 구축된 구매 및 판매 응용 프로그램

### 3. 파일 시스템과 DBMS의 비교

표 1-5 DBMS의 장점

장점	설명
데이터 중복 최소화	DBMS를 이용하여 데이터를 공유하기 때문에 중복 가능성 낮음
데이터 일관성 유지	중복 제거로 데이터의 일관성이 유지됨
데이터 독립성 유지	데이터 정의와 프로그램의 독립성 유지 가능
관리 기능 제공	데이터 복구, 보안, 동시성 제어, 데이터 관리 기능 등을 수행
프로그램 개발 생산성 향상	짧은 시간에 큰 프로그램을 개발할 수 있음
기타	데이터 무결성 유지, 데이터 표준 준수 용이

## 04. 데이터베이스 시스템의 구성

---

- 데이터베이스 언어
- 데이터베이스 사용자
- DBMS
- 데이터 모델
- 데이터베이스의 개념적 구조

## 04. 데이터베이스 시스템의 구성

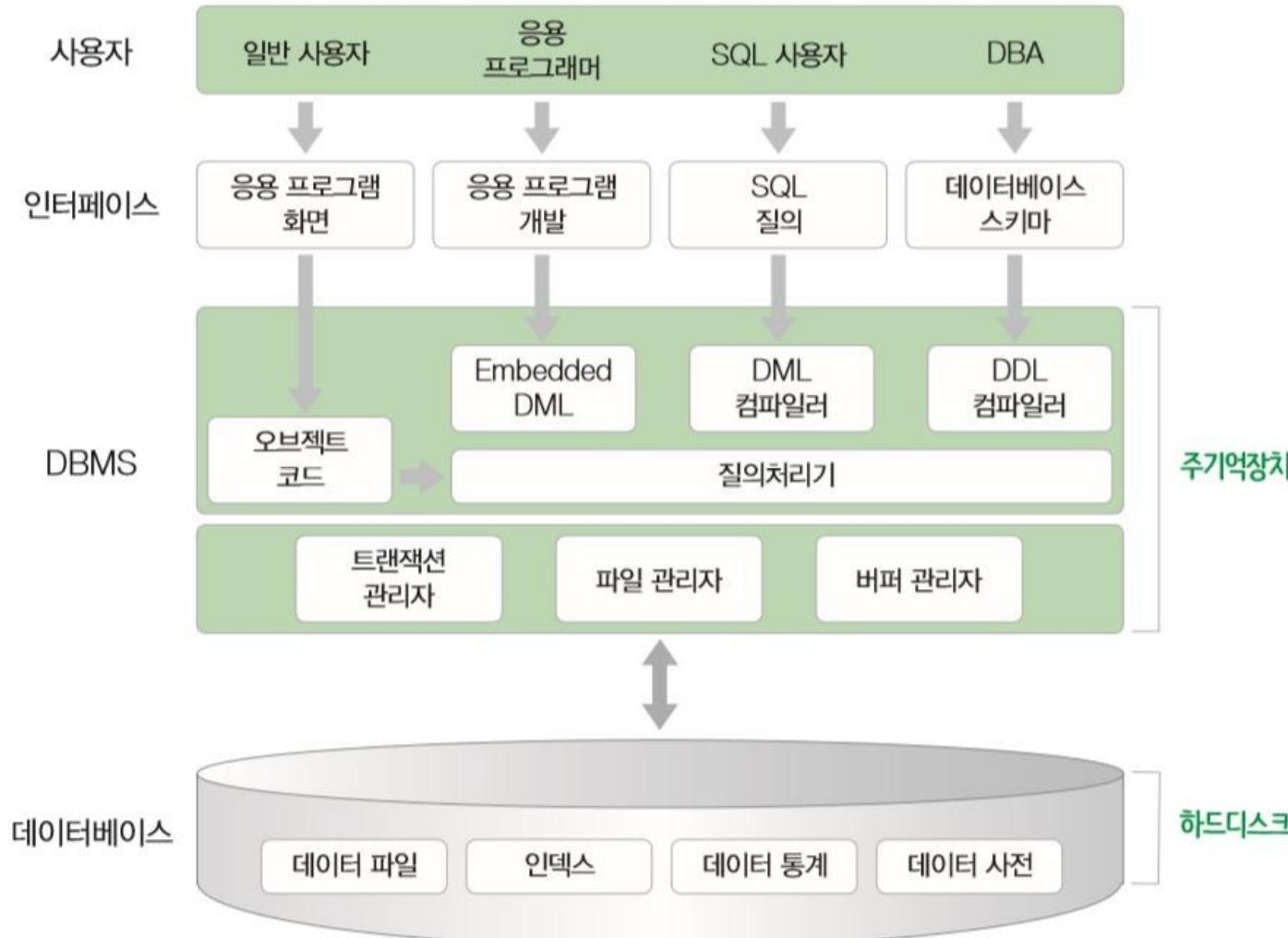


그림 1-13 데이터베이스 시스템의 구성

# 1. 데이터베이스 언어

## ■ SQL

- 데이터 정의어(DDL, Data Definition Language)
- 데이터 조작어(DML, Data Manipulation Language)
- 데이터 제어어(DCL, Data Control Language)

질의 1-1 Book 테이블에서 모든 도서이름(bookname)과 출판사(publisher)를 검색하시오

풀이

```
SELECT bookname, publisher  
FROM Book;
```

bookname	publisher
축구의 역사	굿스포츠
축구 아는 여자	나무수
축구의 이해	대한미디어
골프 바이블	대한미디어
피겨 교본	굿스포츠

Book 테이블에서 모든 도서의 bookname과 publisher 값을 보여 준다.

Book 테이블

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000



# 1. 데이터베이스 언어

질의 1-2 가격(price)이 10,000원 이상인 도서이름(bookname)과 출판사(publisher)를 검색하시오

```
SELECT bookname, publisher  
FROM Book  
WHERE price >=10000;
```

bookname	publisher
축구 아는 여자	나무수
축구의 이해	대한미디어
골프 바이블	대한미디어

Book 테이블에서 가격이 10,000원 이상인 도서의 bookname과 publisher 값을 보여준다.

Book 테이블

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000



## 2. 데이터베이스 사용자

### ■ 일반사용자

- 은행의 창구 혹은 관공서의 민원 접수처 등에서 데이터를 다루는 업무를 하는 사람
- 프로그래머가 개발한 프로그램을 이용하여 데이터베이스에 접근 일반인

### ■ 응용프로그래머

- 일반 사용자가 사용할 수 있도록 프로그램을 만드는 사람
- 자바, C, JSP 등의 프로그래밍 언어와 SQL을 사용하여 일반 사용자를 위한 사용자 인터페이스와 데이터를 관리하는 응용 로직을 개발

### ■ SQL 사용자

- SQL을 사용하여 업무를 처리하는 IT 부서의 담당자
- 응용 프로그램으로 구현되어 있지 않은 업무를 SQL을 사용하여 처리

### ■ 데이터베이스 관리자(DBA, Database Administrator)

- 데이터베이스 운영 조직의 데이터베이스 시스템을 총괄하는 사람
- 데이터 설계, 구현, 유지보수의 전 과정을 담당
- 데이터베이스 사용자 통제, 보안, 성능 모니터링, 데이터 전체 파악 및 관리, 데이터 이동 및 복사 등 제반 업무를 함

## 2. 데이터베이스 사용자

표 1-6 데이터베이스 사용자별로 갖추어야 할 지식 수준(×: 없음, ○: 보통, ◎: 높음)

구분	SQL 언어	프로그래밍 능력	DBMS 지식	데이터 구성
일반 사용자	×	×	×	×
SQL 사용자	◎	×	○	○
응용 프로그래머	◎	◎	○	○
데이터베이스 관리자	◎	○	◎	◎

### 3. DBMS

표 1-7 DBMS의 기능

기능	설명
데이터 정의	<ul style="list-style-type: none"><li>데이터의 구조를 정의하고 데이터 구조에 대한 삭제 및 변경 기능을 수행</li></ul>
데이터 조작	<ul style="list-style-type: none"><li>데이터를 조작하는 소프트웨어(응용 프로그램)가 요청하는 데이터의 검색, 삽입, 수정, 삭제 작업을 지원</li></ul>
데이터 추출	<ul style="list-style-type: none"><li>사용자가 조회하는 데이터 혹은 응용 프로그램의 데이터를 추출</li></ul>
데이터 제어	<ul style="list-style-type: none"><li>데이터베이스 사용자를 생성하고 모니터링하며 접근을 제어</li><li>백업과 회복, 동시성 제어 등의 기능을 지원</li></ul>

## 4. 데이터 모델

- 계층 데이터 모델(hierarchical data model)
- 네트워크 데이터 모델(network data model)
- 객체 데이터 모델(object data model)
- 관계 데이터 모델(relational data model) → 가장 많이 쓰인다
- 객체-관계 데이터 모델(object-relational data model)
  - 관계 데이터 모델과 객체 데이터 모델의 장점을 결합한 모델

## 4. 데이터 모델

### ■ 학생-강좌 관계(relationship)를 표현

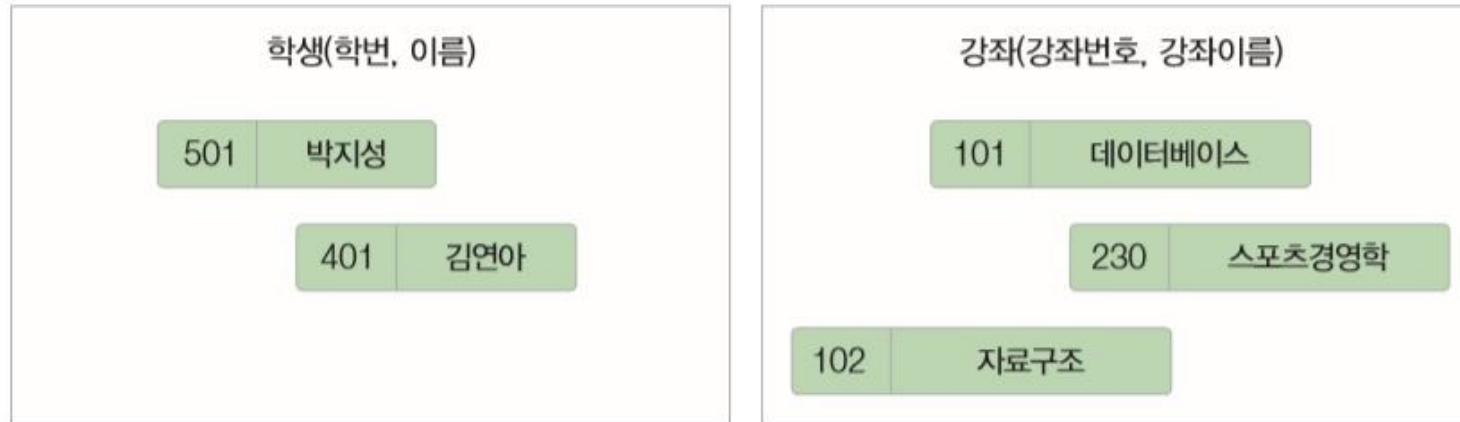
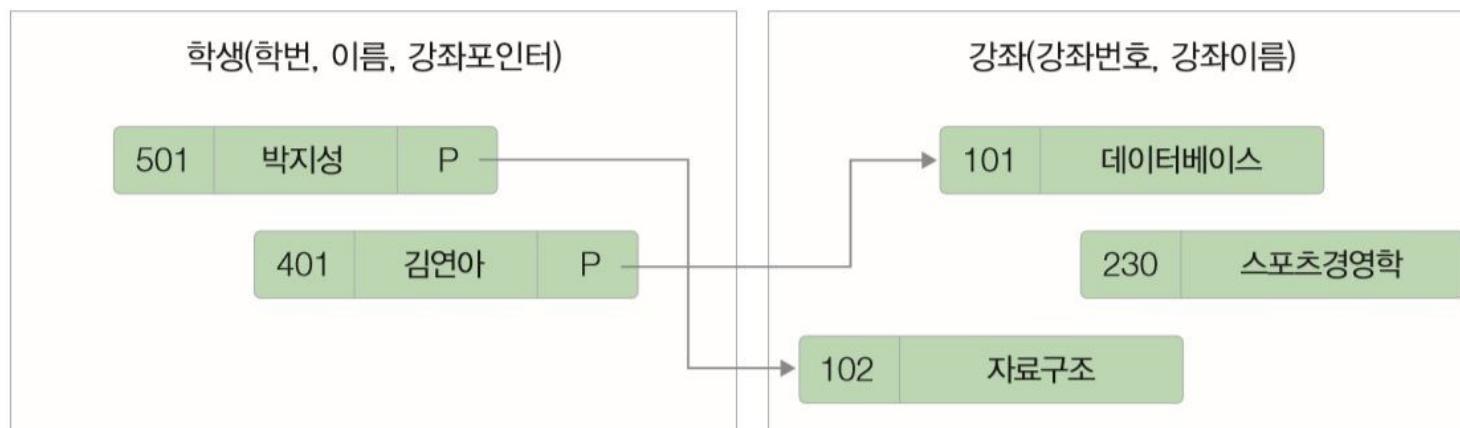


그림 1-14 관계 표현을 위한 예시

### ① 포인터 사용 : 계층 데이터 모델, 네트워크 데이터 모델



## 4. 데이터 모델

### ■ 학생-강좌 관계(relationship)를 표현

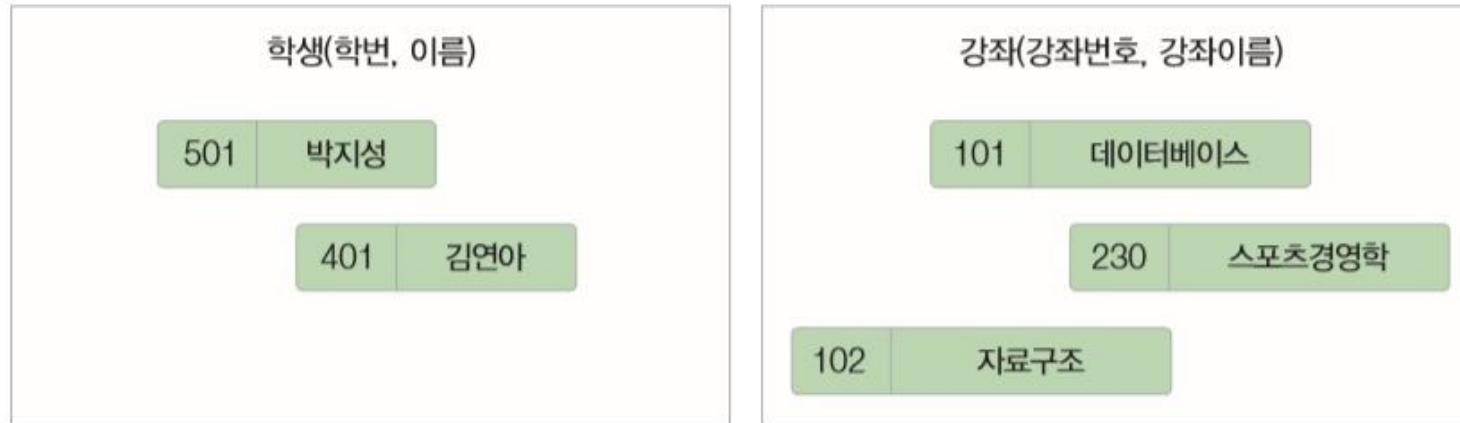


그림 1-14 관계 표현을 위한 예시

### ② 속성 값 사용 : 관계 데이터 모델



## 4. 데이터 모델

### ■ 학생-강좌 관계(relationship)를 표현

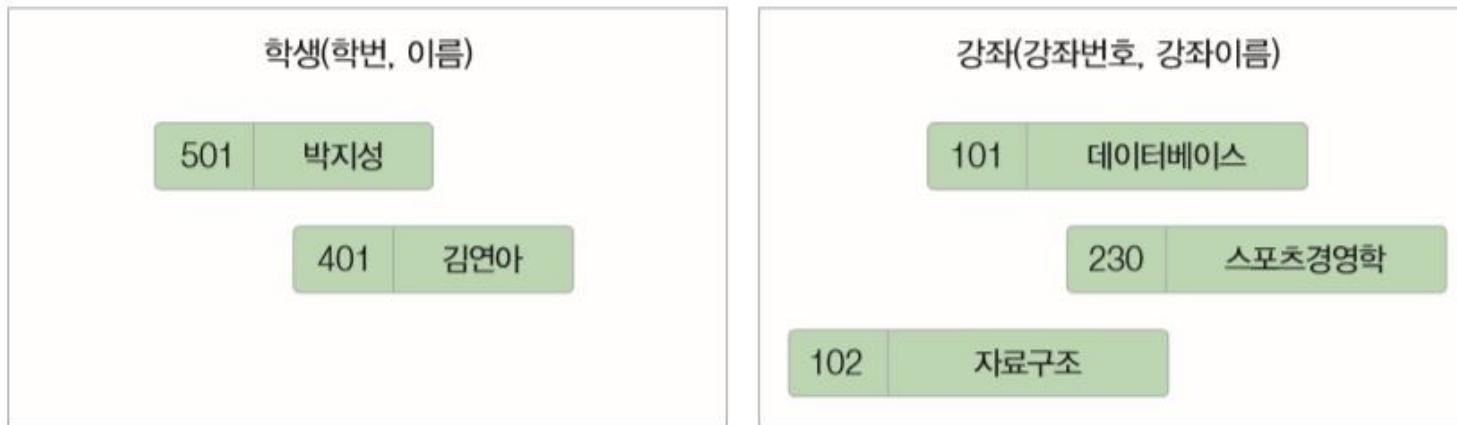


그림 1-14 관계 표현을 위한 예시

### ③ 객체식별자 사용: 객체 데이터 모델



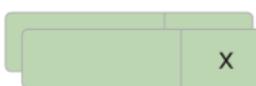
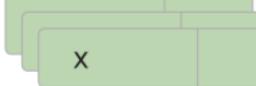
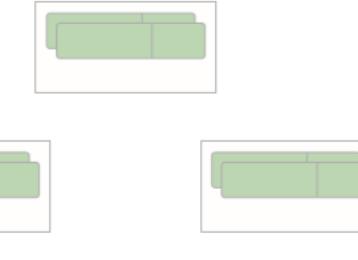
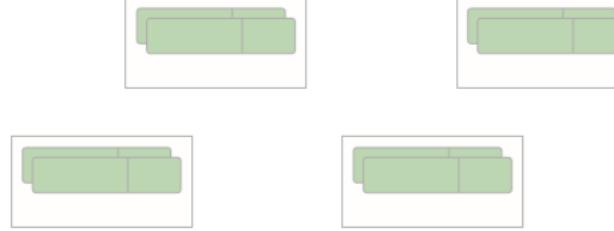
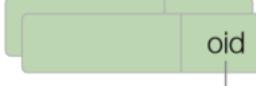
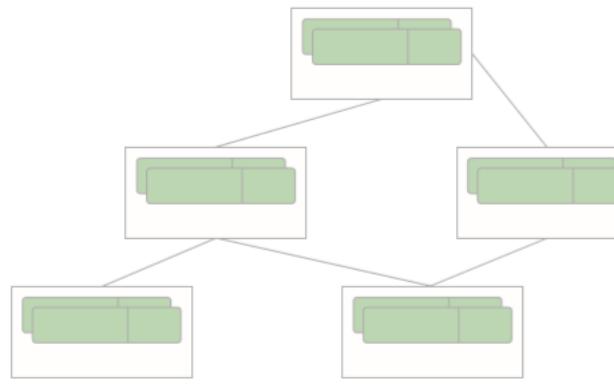
## 4. 데이터 모델

표 1-8 데이터 모델과 각 모델에서 관계의 표현 방법

데이터 모델	관계의 표현	데이터 구성
계층 데이터 모델 (포인터 사용)	<p>학생</p> <p>The diagram shows two student objects represented as green rectangles. A pointer labeled 'P' points from the first student's address to the second student's address, indicating a one-to-one relationship.</p> <p>강좌</p> <p>The diagram shows two course objects represented as green rectangles. A pointer points from the first course's address to the second course's address, indicating a one-to-one relationship.</p>	<p>A tree diagram illustrating a hierarchical structure. At the top level, there is a single node containing two student objects. This node branches down to two nodes, each containing two student objects. These further branch down to four nodes, each containing two student objects. This represents a 2-to-1 relationship at every level of the hierarchy.</p>
네트워크 데이터 모델 (포인터 사용)	<p>학생</p> <p>The diagram shows two student objects. A pointer labeled 'P' points from the first student's address to the second student's address, similar to the hierarchical model.</p> <p>강좌</p> <p>The diagram shows two course objects. A pointer points from the first course's address to the second course's address, similar to the hierarchical model.</p>	<p>A tree diagram illustrating a networked structure. The root node contains two student objects. It has two children, each containing two student objects. These children further have two children each, each containing two student objects. This represents a fully connected network where every node is connected to every other node at every level of the hierarchy.</p>

## 4. 데이터 모델

표 1-8 데이터 모델과 각 모델에서 관계의 표현 방법

데이터 모델	관계의 표현	데이터 구성
관계 데이터 모델 (속성값 사용)	학생  강좌 	
		
객체 데이터 모델 (객체식별자 사용)	학생  강좌  객체 번호 oid	

## 4. 데이터 모델

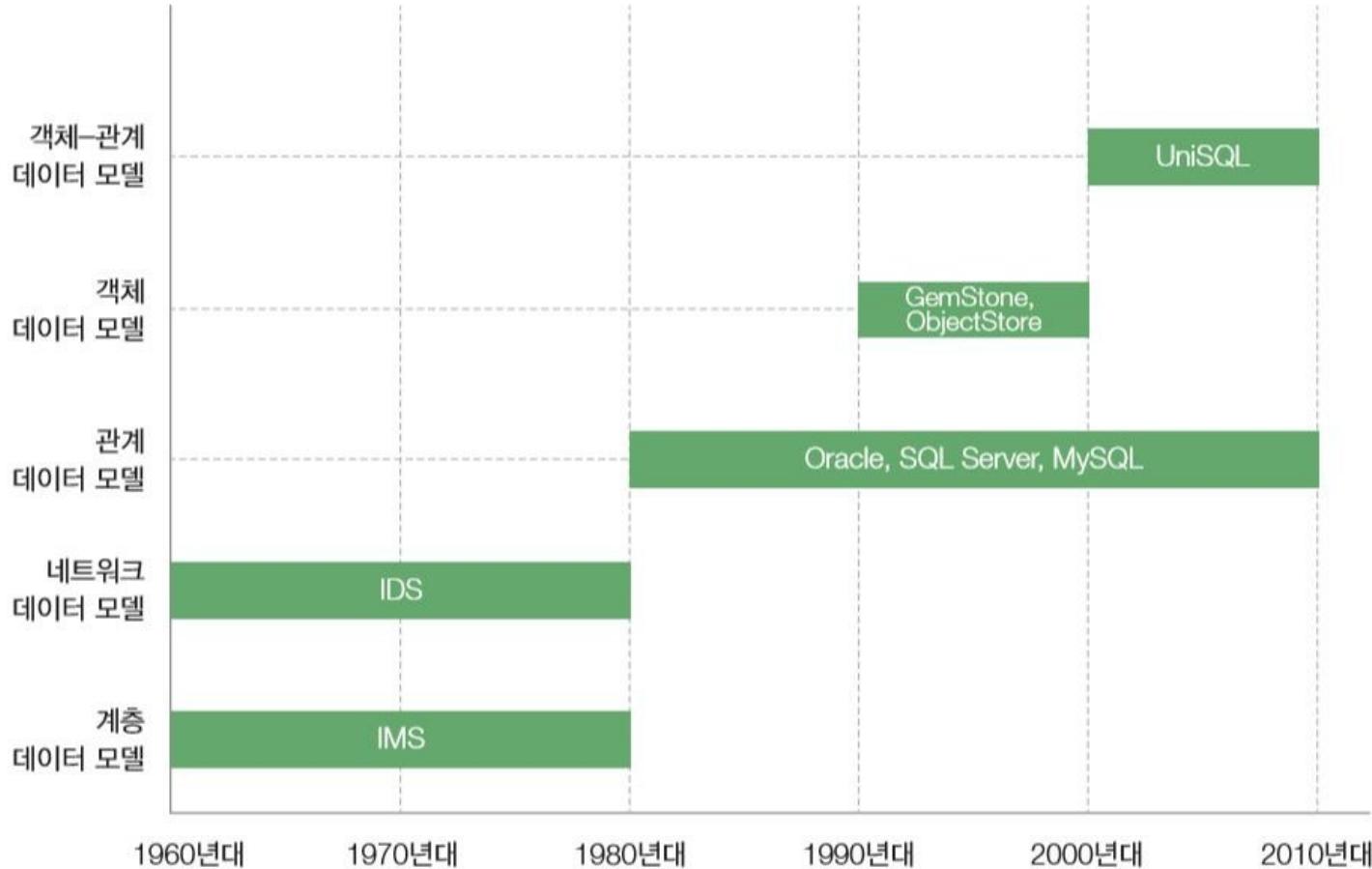


그림 1-15 데이터 모델과 제품의 역사

(주) 그림은 데이터 모델이 주로 사용되는 시기를 표시  
계층과 네트워크 모델은 1960년대, 관계 데이터모델은 1970년대에 처음 사용되기 시작

## 5.1 3단계 데이터베이스 구조

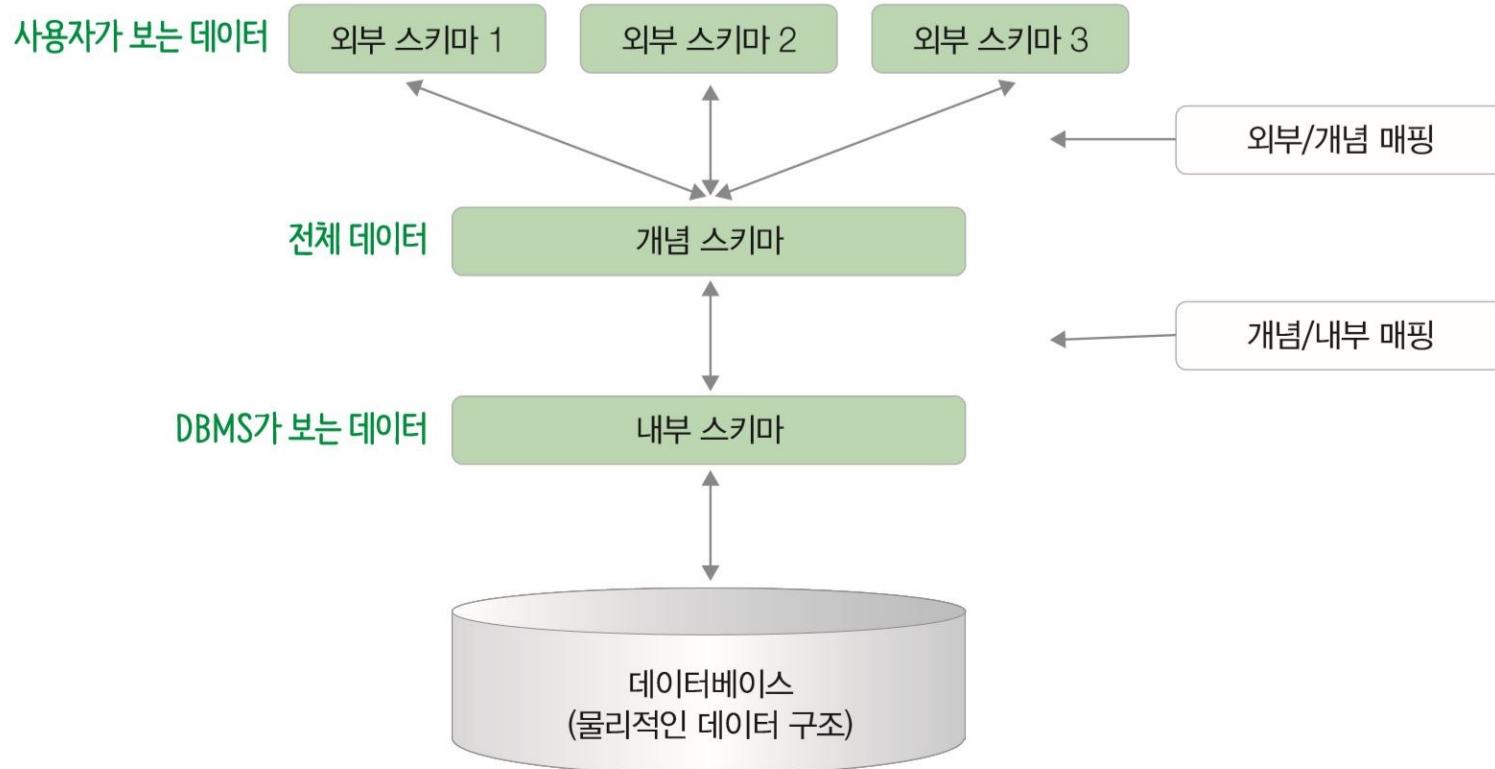


그림 1-16 ANSI의 3단계 데이터베이스 구조

# 5.1 3단계 데이터베이스 구조

## ■ 외부 스키마

- 일반 사용자나 응용 프로그래머가 접근하는 계층, 전체 데이터베이스 중에서 하나의 논리적인 부분을 의미
- 여러 개의 외부 스키마(external schema)가 있을 수 있음
- 서브 스키마(sub schema)라고도 하며, 뷰(view)의 개념임

## ■ 개념 스키마

- 전체 데이터베이스의 정의를 의미
- 통합 조직별로 하나만 존재하며 DBA가 관리함
- 하나의 데이터베이스에는 하나의 개념 스키마(conceptual schema)가 있음

## ■ 내부 스키마

- 물리적 저장 장치에 데이터베이스가 실제로 저장되는 방법의 표현
- 내부 스키마(internal schema)는 하나
- 인덱스, 데이터 레코드의 배치 방법, 데이터 압축 등에 관한 사항이 포함됨

## 5.1 3단계 데이터베이스 구조

---

### ■ 외부/개념 매핑

- 사용자의 외부 스키마와 개념 스키마 간의 매핑(사상)
- 외부 스키마의 데이터가 개념 스키마의 어느 부분에 해당되는지 대응시킴

### ■ 개념/내부 매핑

- 개념 스키마의 데이터가 내부 스키마의 물리적 장치 어디에 어떤 방법으로 저장되는지 대응시킴

## 5.1 3단계 데이터베이스 구조

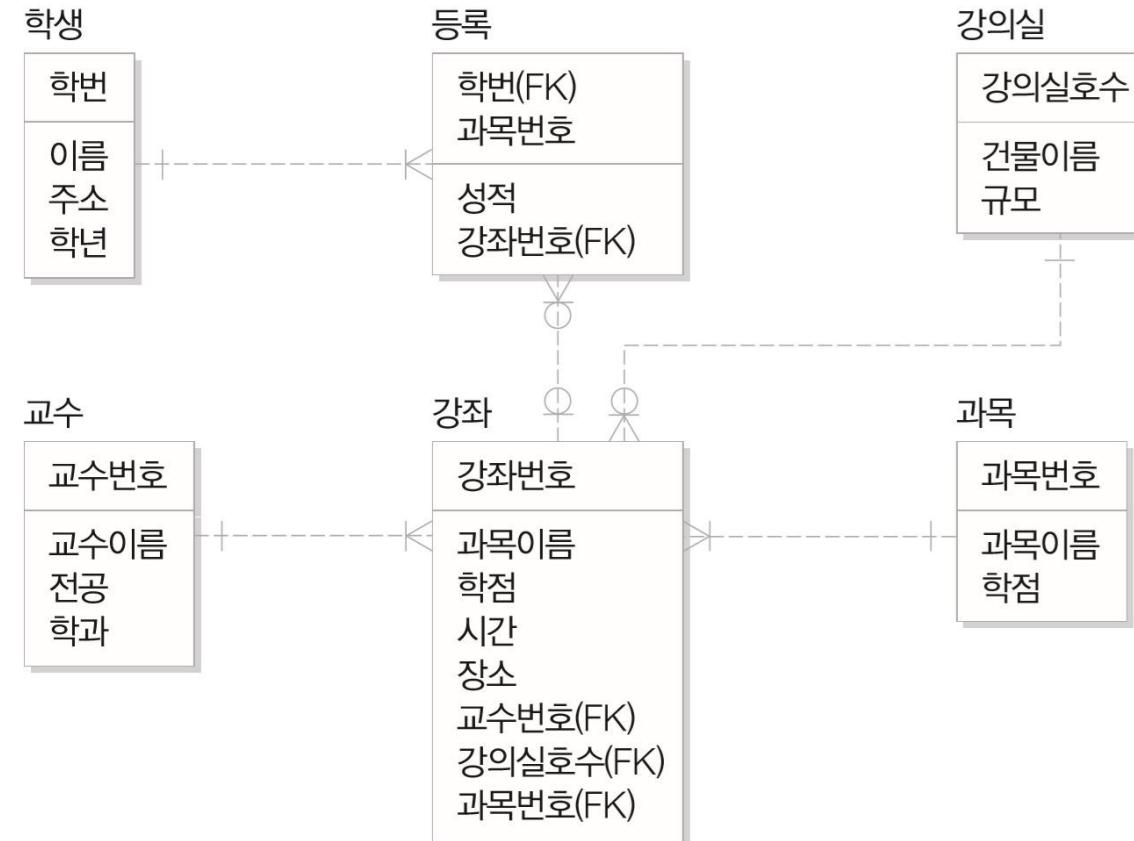


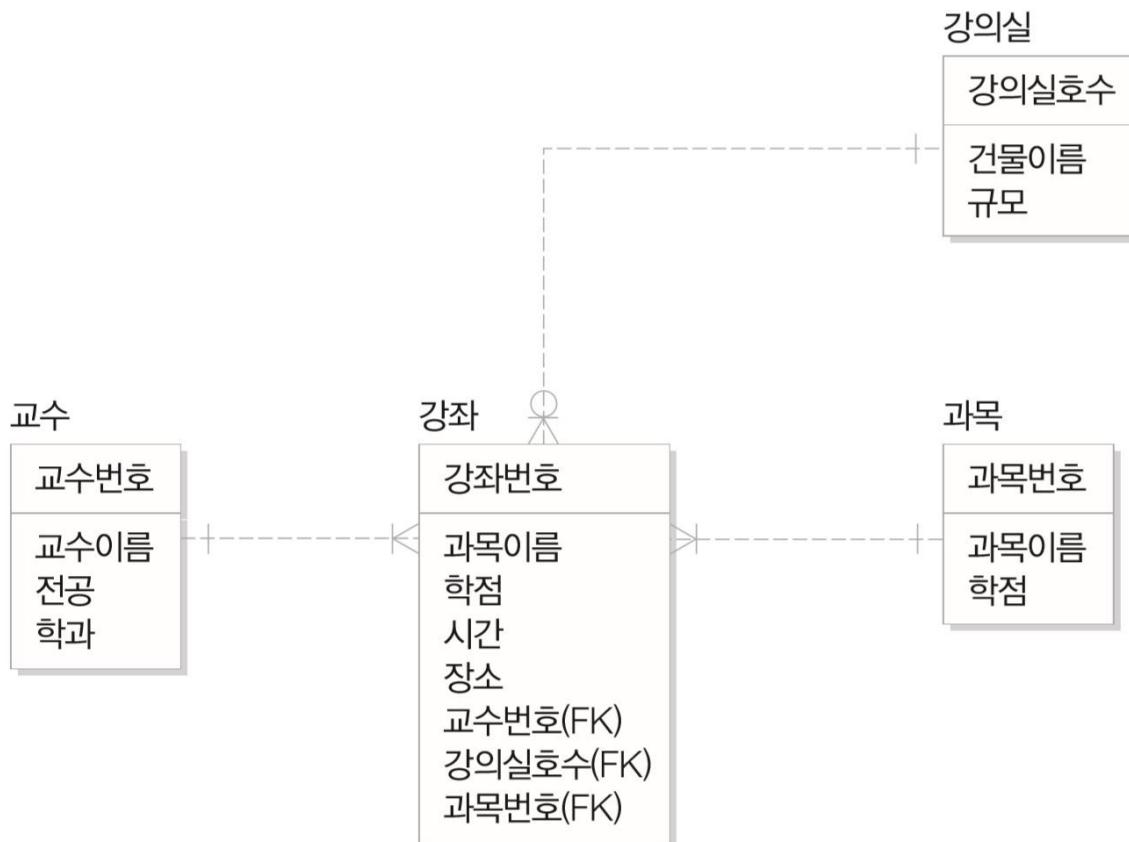
그림 1-17 수강신청 데이터베이스의 개념 스키마

## 5.1 3단계 데이터베이스 구조



(a) 수강등록 업무를 하는 학사관리과에 필요한 데이터베이스(외부 스키마 1)

## 5.1 3단계 데이터베이스 구조



(b) 시간표 작성 업무를 하는 수업관리과에 필요한 데이터베이스(외부 스키마 2)

그림 1-18 수강신청 데이터베이스의 외부 스키마

## 5.1 3단계 데이터베이스 구조

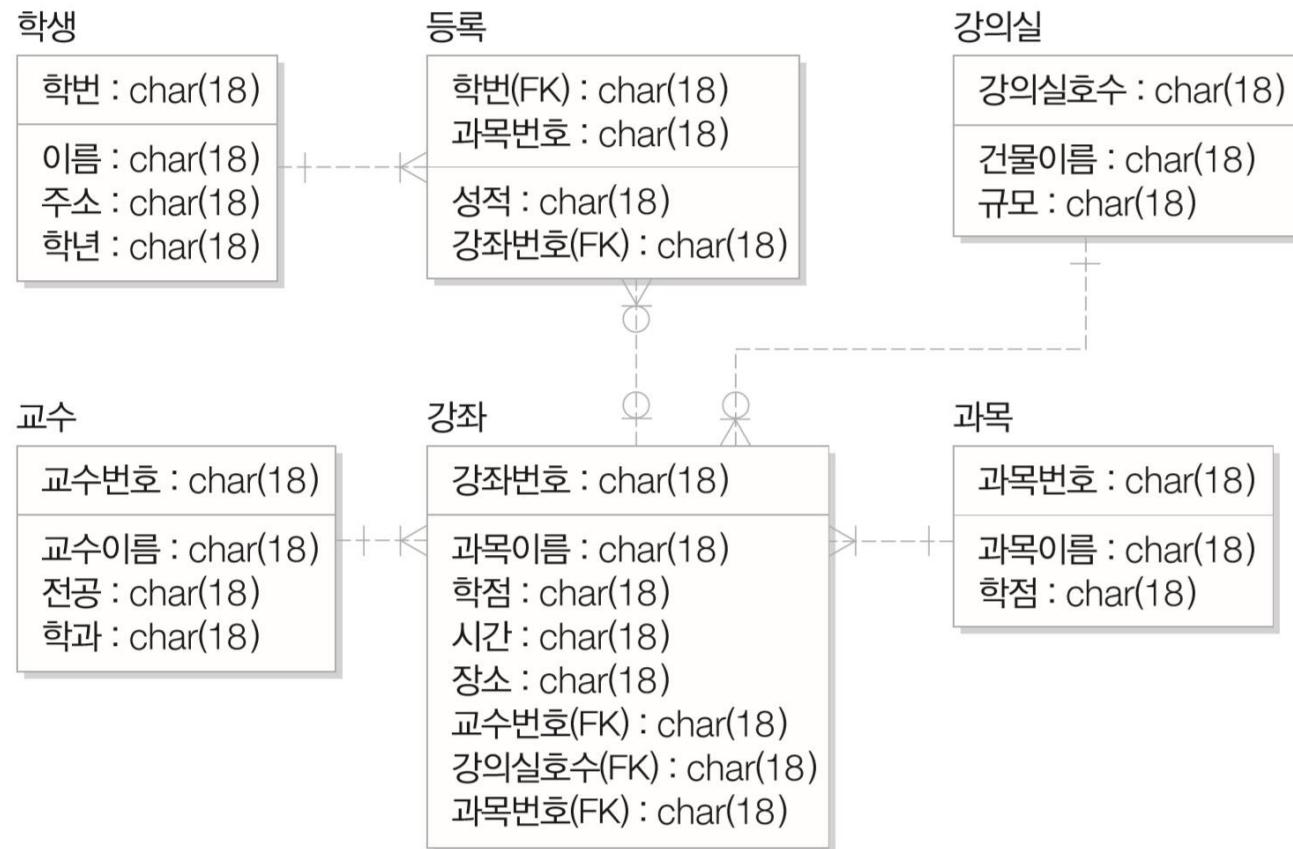


그림 1-19 수강신청 데이터베이스의 내부 스키마

# 5.1 3단계 데이터베이스 구조

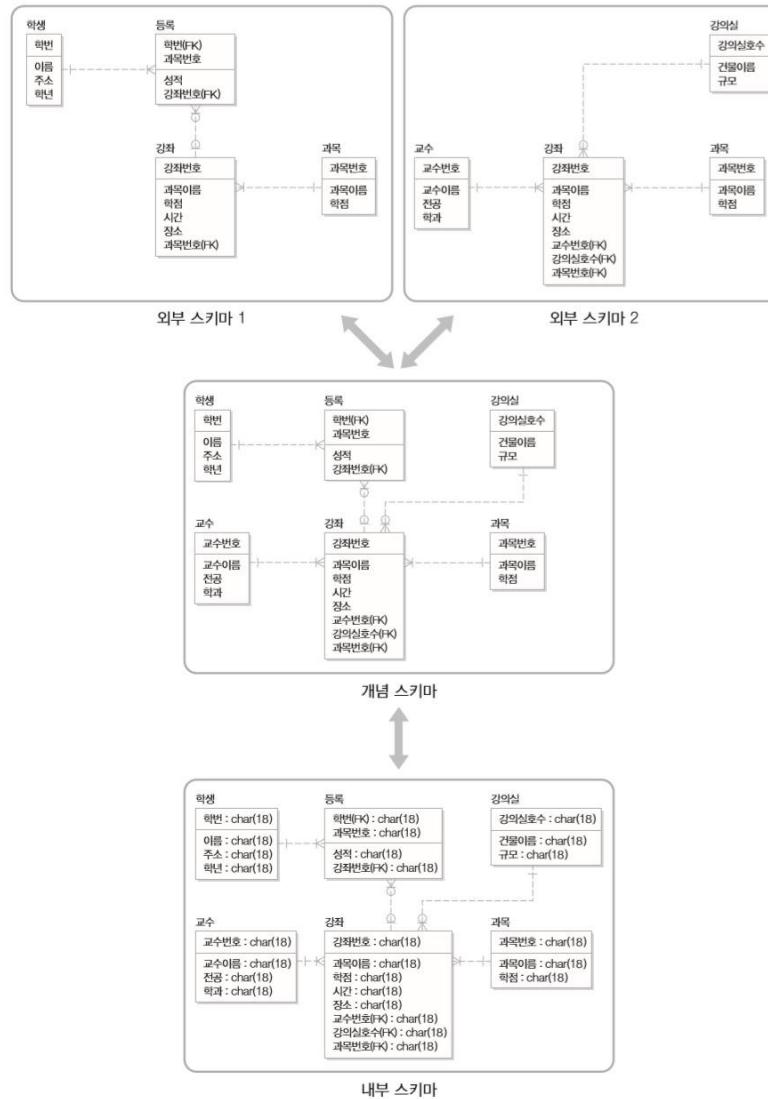


그림 1-20 수강신청 데이터베이스의 3단계 구조

## 5.2 데이터 독립성

### ■ 논리적 데이터 독립성(logical data independence)

- 외부 단계(외부 스키마)와 개념 단계(개념 스키마) 사이의 독립성
- 개념 스키마가 변경되어도 외부 스키마에는 영향을 미치지 않도록 지원
- 논리적 구조가 변경되어도 응용 프로그램에는 영향이 없도록 하는 개념
- 개념 스키마의 테이블을 생성하거나 변경하여도 외부 스키마가 직접 다루는 테이블이 아니면 영향이 없음

### ■ 물리적 데이터 독립성(physical data independence)

- 개념 단계(개념 스키마)와 내부 단계(내부 스키마) 사이의 독립성
- 저장장치 구조 변경과 같이 내부 스키마가 변경되어도 개념 스키마에 영향을 미치지 않도록 지원
- 성능 개선을 위하여 물리적 저장 장치를 재구성할 경우 개념 스키마나 응용 프로그램 같은 외부 스키마에 영향이 없음
- 물리적 독립성은 논리적 독립성보다 구현하기 쉬움

[참고] <https://www.guru99.com/dbms-data-independence.html>

# 요약

---

1. 데이터베이스
2. 데이터베이스의 개념
3. 데이터베이스의 특징
4. 데이터베이스 시스템의 구성
5. 정보 시스템의 발전
6. DBMS의 장점
7. SQL
8. 데이터베이스 관리자(DBA)
9. 데이터 모델
10. 3단계 데이터베이스 구조
11. 데이터 독립성

# 목차

1. 관계 데이터 모델의 개념
2. 무결성 제약조건
3. 관계대수

# 학습목표

- 관계 데이터 모델의 개념을 이해한다.
- 관계 데이터 모델의 제약조건을 알아본다.
- 관계 데이터 모델의 연산인 관계대수의 종류와 작성법을 알아본다.

# 01. 관계 데이터 모델의 개념

---

- 릴레이션
- 릴레이션 스키마와 인스턴스
- 릴레이션의 특징
- 관계 데이터 모델

# 1.1 릴레이션의 개념

- 릴레이션(relation) : 행과 열로 구성된 테이블

표 2-1 relation과 관련된 한글 용어

용어	한글 용어
relation	릴레이션, 테이블('관계'라고 하지 않음)
relational data model	관계 데이터 모델
relational database	관계 데이터베이스
relational algebra	관계대수
relationship	관계

# 1.1 릴레이션의 개념

## ■ 릴레이션이란?

The diagram illustrates the concept of a relation. On the left, five data points are listed in boxes:

- 도서 1, 축구의 역사, 굿스포츠, 7000
- 도서 2, 축구 아는 여자, 나무수, 13000
- 도서 3, 축구의 이해, 대한미디어, 22000
- 도서 4, 골프 바이블, 대한미디어, 35000
- 도서 5, 피겨 교본, 굿스포츠, 8000

An arrow points from these data points to a table on the right, which represents the same information in a structured format:

도서번호	도서이름	출판사	가격
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000

그림 2-1 데이터와 테이블(릴레이션)

도서번호	= {1,2,3,4,5}
도서이름	= {축구의 역사, 축구 아는 여자, 축구의 이해, 골프 바이블, 피겨 교본}
출판사	= {굿스포츠, 나무수, 대한미디어}
가격	= {7000, 13000, 22000, 35000, 8000}

- 첫 번째 행(1, 축구의 역사, 굿스포츠, 7000)의 경우 네 개의 집합에서 각각 원소 한 개씩 선택하여 만들어진 것으로 이 원소들이 관계(relationship)를 맺음

# 1.1 릴레이션의 개념

## ■ 관계(relationship)

- ① 릴레이션 내에서 생성되는 관계 : 릴레이션 내 데이터들의 관계
- ② 릴레이션 간에 생성되는 관계 : 릴레이션 간의 관계



그림 2-2 릴레이션 간의 관계

## 1.2 릴레이션 스키마와 인스턴스

속성(애트리뷰트, 열)

도서			
도서번호	도서이름	출판사	가격
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000

↑ 투플(행)

← 스키마(내포)

← 인스턴스(외연)

그림 2-3 도서 릴레이션

## 1.2 릴레이션 스키마와 인스턴스

### ■ 릴레이션 스키마

- 속성(attribute) : 릴레이션 스키마의 열
- 도메인(domain) : 속성이 가질 수 있는 값의 집합
- 차수(degree) : 속성의 개수

### ■ 스키마의 표현

- 릴레이션 이름(속성1 : 도메인1, 속성2 : 도메인2, 속성3 : 도메인3 ...)
  - 예) 도서 (도서번호, 도서이름, 출판사, 가격)
  - 도서 (도서번호:integer, 도서이름:char(40), 출판사:char(40), 가격:integer))

## 1.2 릴레이션 스키마와 인스턴스

### ■ 릴레이션 인스턴스

- 투플(tuple) : 릴레이션의 행
- 카디널리티(cardinality) : 투플의 수

→ 투플이 가지는 속성의 개수는 릴레이션 스키마의 차수와 동일하고,  
릴레이션 내의 모든 투플들은 서로 중복되지 않아야 함

표 2-2 릴레이션 구조와 관련된 용어

릴레이션 용어	같은 의미로 통용되는 용어	파일 시스템 용어
릴레이션(relation)	테이블(table)	파일(file)
스키마(schema)	내포(intension)	헤더(header)
인스턴스(instance)	외연(extension)	데이터(data)
투플(tuple)	행(row)	레코드(record)
속성(attribute)	열(column)	필드(field)

## 1.3 릴레이션의 특징

---

### ■ 속성은 단일 값을 가진다

- 각 속성의 값은 도메인에 정의된 값만을 가지며 그 값은 모두 단일 값이여야 함.

### ■ 속성은 서로 다른 이름을 가진다

- 속성은 한 릴레이션에서 서로 다른 이름을 가져야만 함.

### ■ 한 속성의 값은 모두 같은 도메인 값을 가진다

- 한 속성에 속한 열은 모두 그 속성에서 정의한 도메인 값만 가질 수 있음.

### ■ 속성의 순서는 상관없다

- 속성의 순서가 달라도 릴레이션 스키마는 같음.
- 예) 릴레이션 스키마에서 (이름, 주소) 순으로 속성을 표시하거나 (주소, 이름) 순으로 표시하여도 상관없음.

### ■ 릴레이션 내의 중복된 투플은 허용하지 않는다

- 하나의 릴레이션 인스턴스 내에서는 서로 중복된 값을 가질 수 없음. 즉 모든 투플은 서로 값이 달라야 함.

### ■ 투플의 순서는 상관없다

- 투플의 순서가 달라도 같은 릴레이션임. 관계 데이터 모델의 투플은 실제적인 값을 가지고 있으며 이 값은 시간이 지남에 따라 데이터의 삭제, 수정, 삽입에 따라 순서가 바뀔 수 있음.

## 1.3 릴레이션의 특징

도서번호	도서이름	출판사	가격
1	축구의 역사	굿스포츠	7000
2	축구아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
5	피겨 교본	굿스포츠	8000
6	피겨 교본, 피겨 기초	굿스포츠	8000

속성의 같은 단일 값이어야 함

동일한 튜플이 중복되면 안 됨

그림 2-4 릴레이션의 특징에 위배된 경우

## 1.4 관계 데이터 모델

- 관계 데이터 모델은 데이터를 2차원 테이블 형태인 릴레이션으로 표현함.  
릴레이션에 대한 제약조건(**constraints**)과 관계 연산을 위한 관계대수(**relational algebra**)를 정의함.

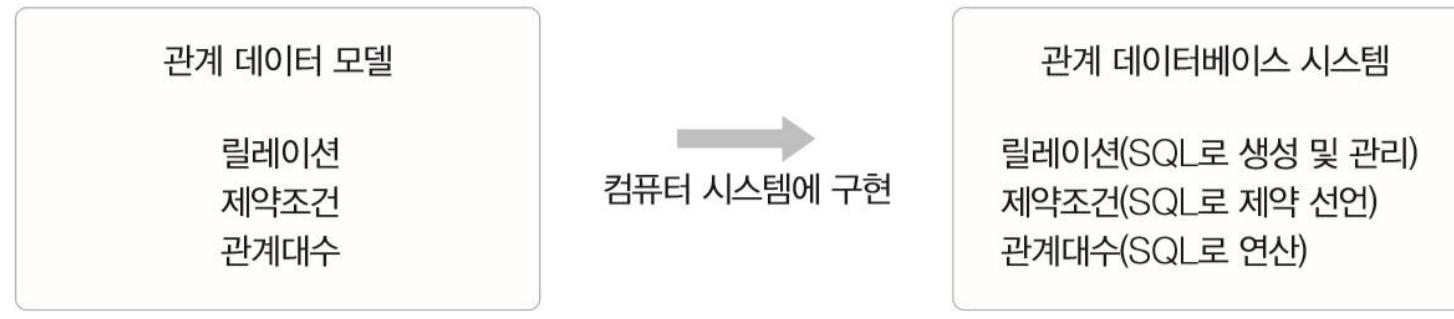


그림 2-5 관계 데이터베이스 시스템

## 02. 무결성 제약조건

---

- 키
- 무결성 제약조건
- 무결성 제약조건의 수행

## 2.1 키

- 특정 투플을 식별할 때 사용하는 속성 혹은 속성의 집합
- 릴레이션은 중복된 투플을 허용하지 않기 때문에 각각의 투플에 포함된 속성들 중 어느 하나(혹은 하나 이상)는 값이 달라야 함. 즉 키가 되는 속성(혹은 속성의 집합)은 반드시 값이 달라서 투플들을 서로 구별할 수 있어야 함
- 키는 릴레이션 간의 관계를 맺는 데도 사용



그림 2-6 자동차 한 대당 키는 단 하나

## 2.1 키

### 고객

고객번호	이름	주민번호	주소	핸드폰
1	박지성	810101-1111111	영국 맨체스터	000-5000-0001
2	김연아	900101-2222222	대한민국 서울	000-6000-0001
3	장미란	830101-2333333	대한민국 강원도	000-7000-0001
4	추신수	820101-1444444	미국 클리블랜드	000-8000-0001

### 도서

도서번호	도서이름	출판사	가격
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000

### 주문

고객번호	도서번호	판매가격	주문일자
1	1	7000	2020-07-01
1	2	13000	2020-07-03
2	5	8000	2020-07-03
3	2	13000	2020-07-04
4	4	35000	2020-07-05
1	3	22000	2020-07-07
4	3	22000	2020-07-07

그림 2-7 마당서점 데이터베이스

# 슈퍼키

## ■ 투플을 유일하게 식별할 수 있는 하나의 속성 혹은 속성의 집합

투플을 유일하게 식별할 수 있는 값이면 모두 슈퍼키가 될 수 있음

### ■ (고객 릴레이션 예)

- 고객번호 : 고객별로 유일한 값이 부여되어 있기 때문에 투플을 식별할 수 있음.
- 이름 : 동명이인이 있을 경우 투플을 유일하게 식별할 수 없음.
- 주민번호 : 개인별로 유일한 값이 부여되어 있기 때문에 투플을 식별할 수 있음.
- 주소 : 가족끼리는 같은 정보를 사용하므로 투플을 식별할 수 없음.
- 핸드폰 : 한 사람이 여러 개의 핸드폰을 사용할 수 있고 반대로 핸드폰을 사용하지 않는 사람이 있을 수 있기 때문에 투플을 식별할 수 없음.

## ■ 고객 릴레이션은 고객번호와 주민번호를 포함한 모든 속성의 집합이 슈퍼키가 됨.

EX) (주민번호), (주민번호, 이름), (주민번호, 이름, 주소), (주민번호, 이름, 핸드폰),

(고객번호), (고객번호, 이름, 주소), (고객번호, 이름, 주민번호, 주소, 핸드폰) 등

## 후보키

### ■ 투플을 유일하게 식별할 수 있는 속성의 최소 집합

(주문 릴레이션 예)

- 고객번호 : 한 명의 고객이 여러 권의 도서를 구입할 수 있으므로 후보키가 될 수 없음. 고객번호가 1인 박지성 고객은 세 번의 주문 기록이 있으므로 투플을 유일하게 식별할 수 없음.
- 도서번호 : 도서번호가 2인 '축구하는 여자'는 두 번의 주문 기록이 있으므로 투플을 유일하게 식별할 수 없음.

### ■ 주문 릴레이션의 후보키는 2개의 속성을 합한 (고객번호, 도서번호)가 됨.

참고로 이렇게 2개 이상의 속성으로 이루어진 키를 복합키(composite key)라고 함

## 기본키

- 여러 후보키 중 하나를 선정하여 대표로 삼는 키
- 후보키가 하나뿐이라면 그 후보키를 기본키로 사용하면 되고 여러 개라면 릴레이션의 특성을 반영하여 하나를 선택하면 됨.

### ■ 기본키 선정 시 고려사항

- 릴레이션 내 투플을 식별할 수 있는 고유한 값을 가져야 함.
- NULL 값은 허용하지 않음.
- 키 값의 변동이 일어나지 않아야 함.
- 최대한 적은 수의 속성을 가진 것이라야 함.
- 향후 키를 사용하는 데 있어서 문제 발생 소지가 없어야 함.
- 릴레이션 스키마를 표현할 때 기본키는 밑줄을 그어 표시함  
    릴레이션 이름(속성1, 속성2, ..., 속성N)  
    EX) 고객(고객번호, 이름, 주민번호, 주소, 핸드폰)  
        도서(도서번호, 도서이름, 출판사, 가격)

## 대리키

- 기본키가 보안을 요하거나, 여러 개의 속성으로 구성되어 복잡하거나, 마땅한 기본키가 없을 때는 일련번호 같은 가상의 속성을 만들어 기본키로 삼는 경우가 있음. 이러한 키를 대리키(surrogate key) 혹은 인조키(artificial key)라고 함.
- 대리키는 DBMS나 관련 소프트웨어에서 임의로 생성하는 값으로 사용자가 직관적으로 그 값의 의미를 알 수 없음.

주문

주문번호	고객번호	도서번호	판매가격	주문일자
1	1	1	7000	2020-07-01
2	1	2	13000	2020-07-03
3	2	5	8000	2020-07-03
4	3	2	13000	2020-07-04
5	4	4	35000	2020-07-05
6	1	3	22000	2020-07-07
7	4	3	22000	2020-07-07

그림 2-8 대리키를 사용하도록 변경된 주문 릴레이션

## 대체키

- 대체키(alternate key)는 기본키로 선정되지 않은 후보키를 말함.
- 고객 릴레이션의 경우 고객번호와 주민번호 중 고객번호를 기본키로 정하면 주민 번호가 대체키가 됨.

# 외래키

- 다른 릴레이션의 기본키를 참조하는 속성을 말함. 다른 릴레이션의 기본키를 참조하여 관계 데이터 모델의 특징인 릴레이션 간의 관계(relationship)를 표현함.

## ■ 외래키의 특징

- 관계 데이터 모델의 릴레이션 간의 관계를 표현함.
- 다른 릴레이션의 기본키를 참조하는 속성임.
- 참조하고(외래키) 참조되는(기본키) 양쪽 릴레이션의 도메인은 서로 같아야 함.
- 참조되는(기본키) 값이 변경되면 참조하는(외래키) 값도 변경됨.
- NULL 값과 중복 값 등이 허용됨.
- 자기 자신의 기본키를 참조하는 외래키도 가능함.
- 외래키가 기본키의 일부가 될 수 있음.

# 외래키

고객

고객번호	이름	주민번호	주소	핸드폰
1	박지성	810101-1111111	영국 맨체스터	000-5000-0001
2	김연아	900101-2222222	대한민국 서울	000-6000-0001
3	장미란	830101-2333333	대한민국 강원도	000-7000-0001
4	추신수	820101-1444444	미국 클리블랜드	000-8000-0001

기본키

도서

도서번호	도서이름	출판사	가격
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000

기본키

참조

주문

외래키

참조

주문번호	고객번호	도서번호	판매가격	주문일자
1	1	1	7000	2020-07-01
2	1	2	13000	2020-07-03
3	2	5	8000	2020-07-03
4	3	2	13000	2020-07-04
5	4	4	35000	2020-07-05
6	1	3	22000	2020-07-07
7	4	3	22000	2020-07-07

기본키

그림 2-9 릴레이션 간의 참조 관계

## 외래키

- 외래키 사용 시 참조하는 릴레이션과 참조되는 릴레이션이 꼭 다른 릴레이션일 필요는 없음. 즉 자기 자신의 기본 키를 참조할 수도 있음.

참조				
기본키				외래키
선수번호	이름	주소	멘토번호	
1	박지성	영국 맨체스터	NULL	
2	김연아	대한민국 서울	3	
3	장미란	대한민국 강원도	4	
4	추신수	미국 클리블랜드	NULL	

그림 2-10 멘토 릴레이션

## 2.1 키 – 내용 요약

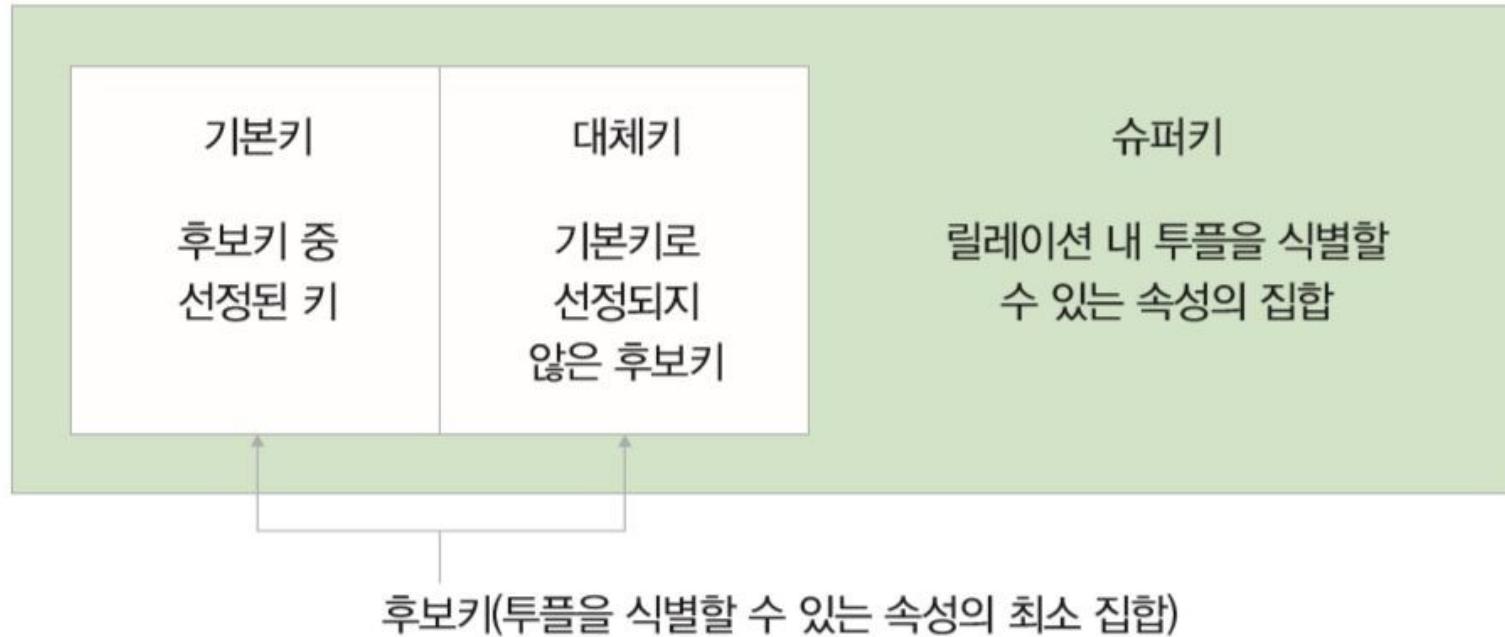


그림 2-11 키의 포함 관계

## 2.2 무결성 제약조건

### ■ 데이터 무결성(integrity, 無缺性)

- 데이터베이스에 저장된 데이터의 일관성과 정확성을 지키는 것을 말함.

### ■ 도메인 무결성 제약조건

- 도메인 제약(domain constraint)이라고도 하며, 릴레이션 내의 투플들이 각 속성의 도메인에 지정된 값만을 가져야 한다는 조건
- SQL 문에서 데이터 형식(type), 널(null/not null), 기본 값(default), 체크(check) 등을 사용하여 지정할 수 있음.

### ■ 개체 무결성 제약조건

- 기본키 제약(primary key constraint)이라고도 함.
- 릴레이션은 기본키를 지정하고 그에 따른 무결성 원칙 즉, 기본키는 NULL 값을 가져서는 안 되며 릴레이션 내에 오직 하나의 값만 존재해야 한다는 조건임.

### ■ 참조 무결성 제약조건

- 외래키 제약(foreign key constraint)이라고도 하며, 릴레이션 간의 참조 관계를 선언하는 제약조건
- 자식 릴레이션의 외래키는 부모 릴레이션의 기본키와 도메인이 동일해야 하며, 자식 릴레이션의 값이 변경될 때 부모 릴레이션의 제약을 받는다는 것

## 2.2 무결성 제약조건

표 2-3 제약조건의 정리

구분	도메인	키	
	도메인 무결성 제약조건	개체 무결성 제약조건	참조 무결성 제약조건
제약 대상	속성	튜플	속성과 튜플
같은 용어	도메인 제약 (domain constraint)	기본키 제약 (primary key constraint)	외래키 제약 (foreign key constraint)
해당되는 키	-	기본키	외래키
NULL 값	허용	불가	허용
릴레이션 내 제약조건의 개수	속성의 개수와 동일	1개	0~여러 개
기타	• 튜플 삽입/수정 시 제약사항 우선 확인	• 튜플 삽입/수정 시 제약 사항 우선 확인	• 튜플 삽입/수정 시 제약사항 우선 확인 • 부모 릴레이션의 튜플 수정/삭제 시 제약사항 우선 확인

- <여기서 잠깐> **UNIQUE 제약조건(unique constraint, 유일성 제약조건, 고유성 제약조건)**
  - 실제 DBMS에서는 위에서 설명한 세 가지 무결성 제약조건과 함께 UNIQUE 제약조건도 사용  
UNIQUE 제약조건은 속성의 모든 값은 서로 같은 값이 없어야 한다는 것  
이는 릴레이션 내의 각각의 튜플을 유일하게 식별할 수 있는 속성들의 집합으로 볼 수 있음  
UNIQUE 제약조건은 기본키 제약과는 달리 NULL 값을 허용

## 2.3 무결성의 수행 제약조건

### 개체 무결성 제약조건

- 삽입 : 기본키 값이 같으면 삽입이 금지됨.
- 수정 : 기본키 값이 같거나 NULL로도 수정이 금지됨.
- 삭제 : 특별한 확인이 필요하지 않으며 즉시 수행함.

학생

학번	이름	학과코드
501	박지성	1001
401	김연아	2001
402	장미란	2001
502	추신수	1001

그림 2-12 학생 릴레이션

(501, 남슬찬, 1001)



(NULL, 남슬찬, 1001)



학번	이름	학과코드
501	박지성	1001
401	김연아	2001
402	장미란	2001
502	추신수	1001

학번	이름	학과코드
501	박지성	1001
401	김연아	2001
402	장미란	2001
502	추신수	1001

그림 2-13 개체 무결성 제약조건의 수행 예(기본키 충돌 및 NULL 값 삽입)

## 2.3 무결성의 수행 제약조건

### 참조 무결성 제약조건

- 학과(부모 릴레이션) : 투플 삽입한 후 수행하면 정상적으로 진행
- 학생(자식 릴레이션) : 참조받는 테이블에 외래키 값이 없으므로 삽입이 금지



그림 2-14 학생관리 데이터베이스

## 2.3 무결성의 수행 제약조건

### ■ 삭제

- 학과(부모 릴레이션) : 참조하는 테이블을 같이 삭제할 수 있어서 금지하거나 다른 추가 작업이 필요함.
- 학생(자식 릴레이션) : 바로 삭제 가능함.
  
- ※ 부모 릴레이션에서 튜플을 삭제할 경우 참조 무결성 조건을 수행하기 위한 고려사항
  - ① 즉시 작업을 중지
  - ② 자식 릴레이션의 관련 튜플을 삭제
  - ③ 초기에 설정된 다른 어떤 값으로 변경
  - ④ NULL 값으로 설정

### ■ 수정

- 삭제와 삽입 명령이 연속해서 수행됨.
- 부모 릴레이션의 수정이 일어날 경우 삭제 옵션에 따라 처리된 후 문제가 없으면 다시 삽입 제약조건에 따라 처리됨.

## 2.3 무결성의 수행 제약조건

표 2-4 참조 무결성 제약조건의 옵션(부모 릴레이션에서 투플을 삭제할 경우)

명령어	의미	예
RESTRICTED	자식 릴레이션에서 참조하고 있으면 부모 릴레이션의 삭제 작업을 거부함	학과 릴레이션의 투플 삭제 거부
CASCADE	자식 릴레이션의 관련 투플을 같이 삭제함	학생 릴레이션의 관련 투플을 삭제
DEFAULT	자식 릴레이션의 관련 투플을 미리 설정해 둔 값으로 변경함	학생 릴레이션의 학과가 다른 학과로 자동 배정
NULL	자식 릴레이션의 관련 투플을 NULL 값으로 설정함(NULL 값을 허가한 경우)	학생 릴레이션의 학과가 NULL 값으로 변경

## 2.3.2 참조 무결성 제약조건

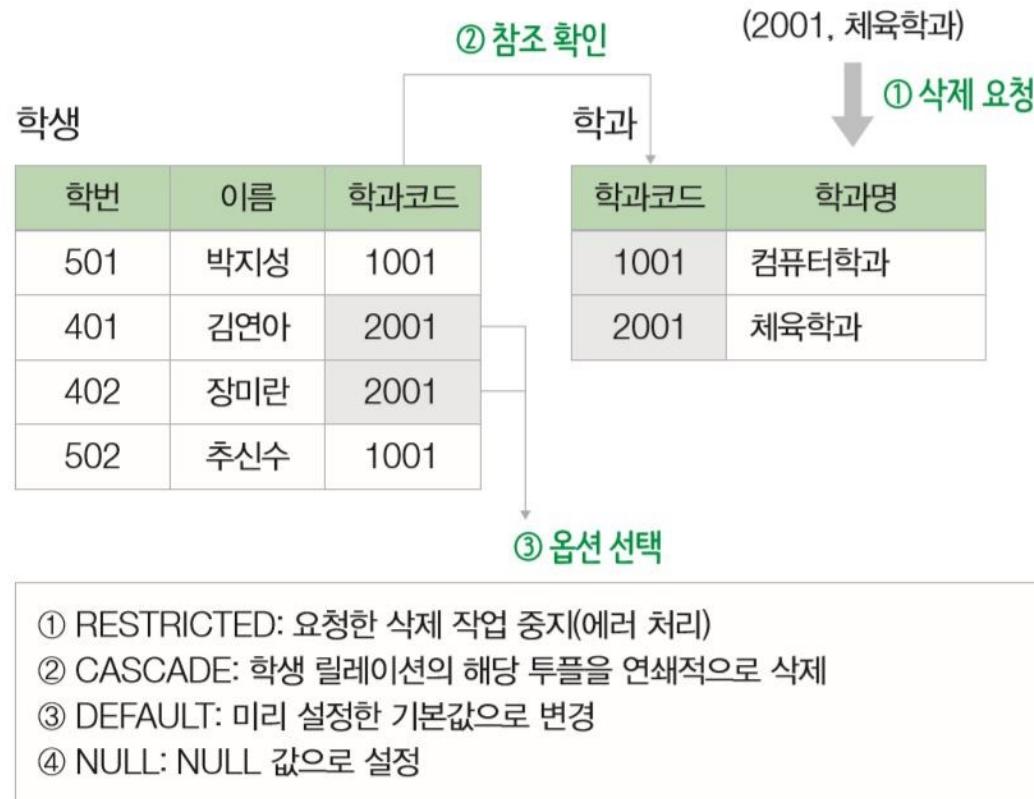


그림 2-15 참조 무결성 제약조건에서 부모 릴레이션의 투플을 삭제할 경우

# 목차

1. SQL 학습에 사용할 마당서점 예제
2. 오라클 설치와 기본 사용법
3. SQL 소개
4. 데이터 조작어–검색
5. 데이터 정의어
6. 데이터 조작어–삽입, 수정, 삭제

# 학습목표

- SQL의 개념과 주요 명령어를 알아본다.
- SELECT 문으로 질의를 처리하는 방법을 알아본다.
- 집계 함수와 GROUP BY 문으로 질의를 처리하는 방법을 알아본다.
- 두 개 이상의 테이블을 조회하여 질의를 처리하는 방법을 알아본다.
- DDL로 테이블의 구조를 정의하고 변경하는 방법을 알아본다.
- DML로 데이터를 삽입, 수정, 삭제하는 방법을 알아본다.

# 01. SQL 학습에 사용할 마당서점 예제

---

- 마당서점의 데이터
- 누가 어떤 정보를 원하는가?
- 오라클과 샘플 데이터 설치

# 1.1 마당서점의 현황과 운영 시스템 환경

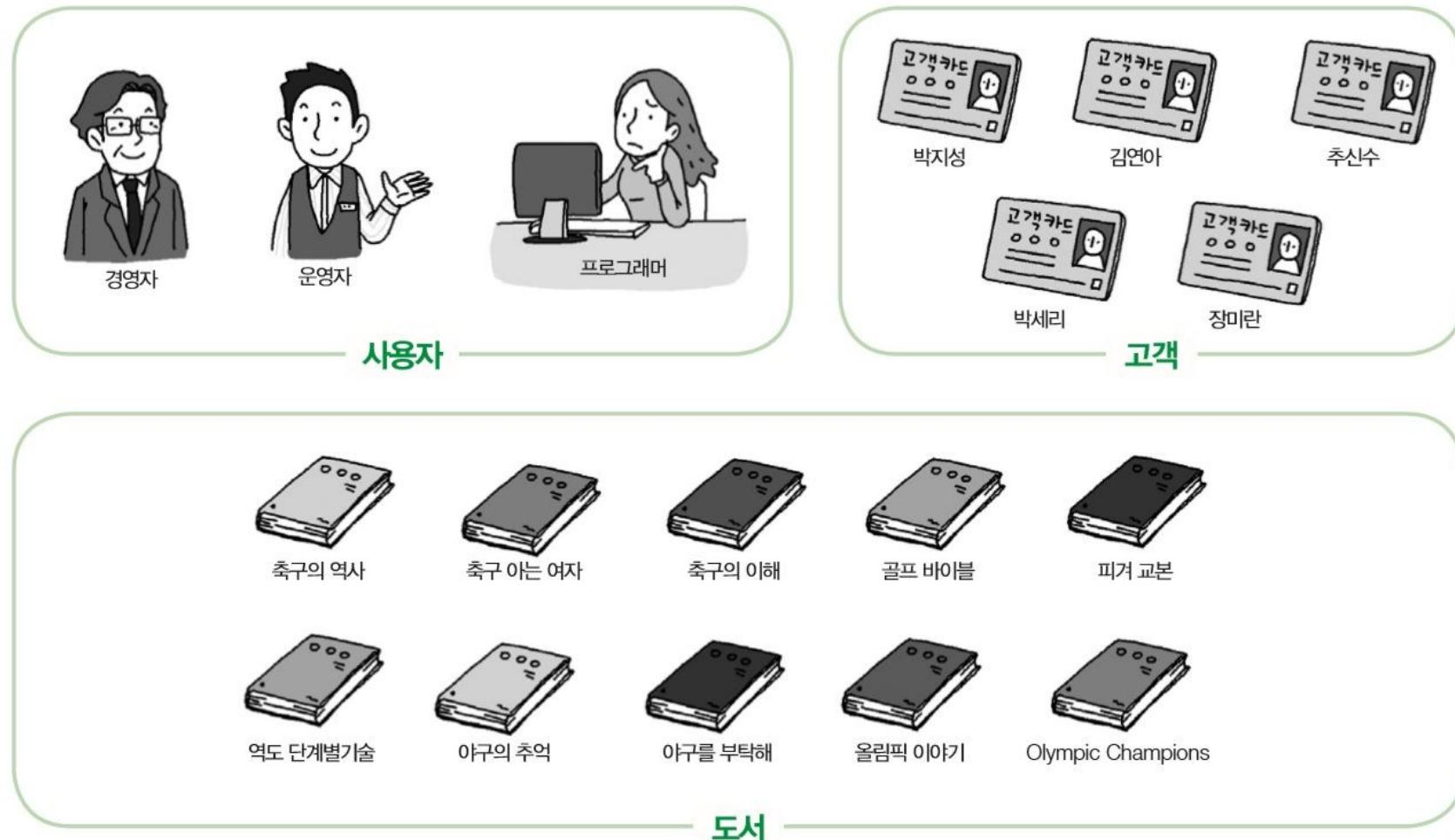


그림 3-1 마당서점 현황

# 1.1 마당서점의 현황과 운영 시스템 환경

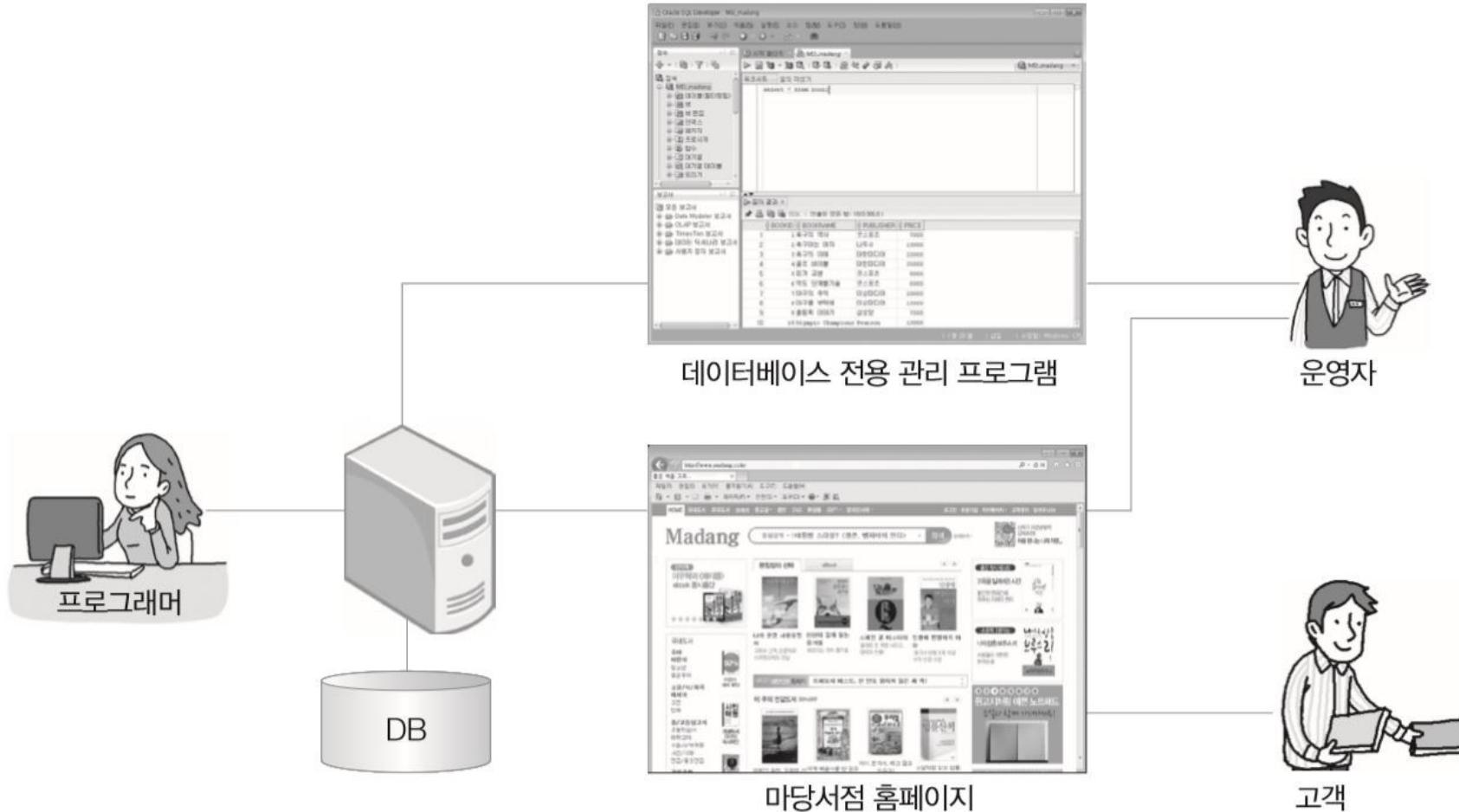


그림 3-2 마당서점 운영 시스템 환경

# 1.2 마당서점의 데이터

Book 테이블

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

Orders 테이블

orderid	custid	bookid	saleprice	orderdate
1	1	1	6000	2020-07-01
2	1	3	21000	2020-07-03
3	2	5	8000	2020-07-03
4	3	6	6000	2020-07-04
5	4	7	20000	2020-07-05
6	1	2	12000	2020-07-07
7	4	8	13000	2020-07-07
8	3	10	12000	2020-07-08
9	2	10	7000	2020-07-09
10	3	8	13000	2020-07-10

Customer 테이블

custid	name	address	phone
1	박지성	영국 맨체스터	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-8000-0001
5	박세리	대한민국 대전	NULL

릴레이션 용어	실무에서 많이 사용되는 용어	같은 의미의 파일 시스템 용어
릴레이션(relation)	테이블(table)	파일(file)
속성(attribute)	열(column)	필드(field)
튜플(tuple)	행(row)	레코드(record)

## 1.2 마당서점의 데이터

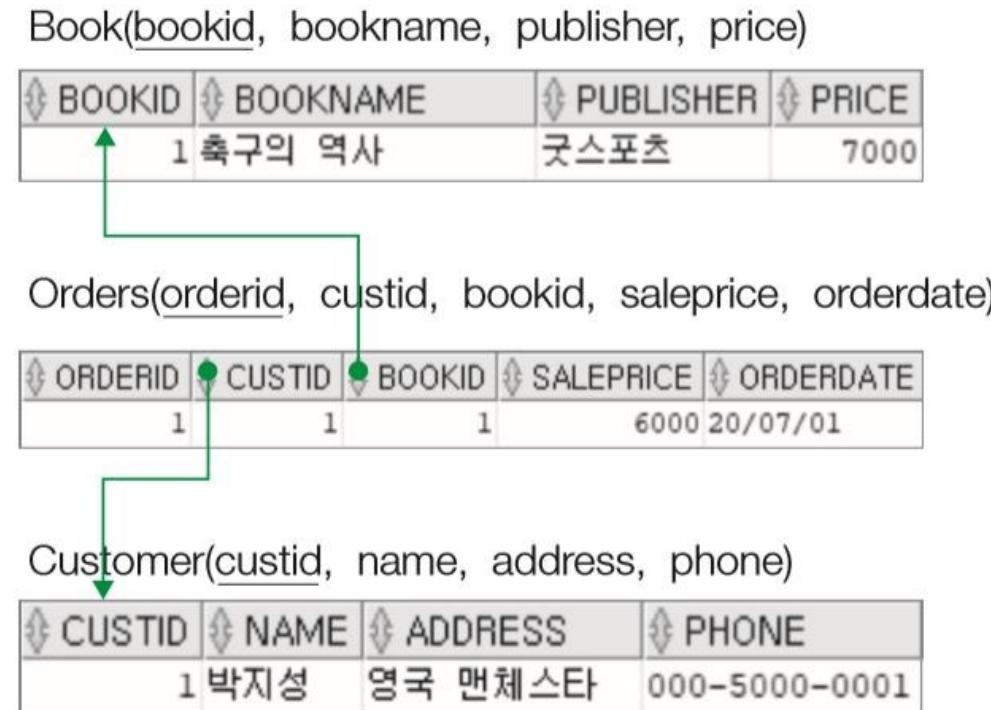


그림 3-3 마당서점의 데이터 구성도

## 1.3 사용자별 요구 정보



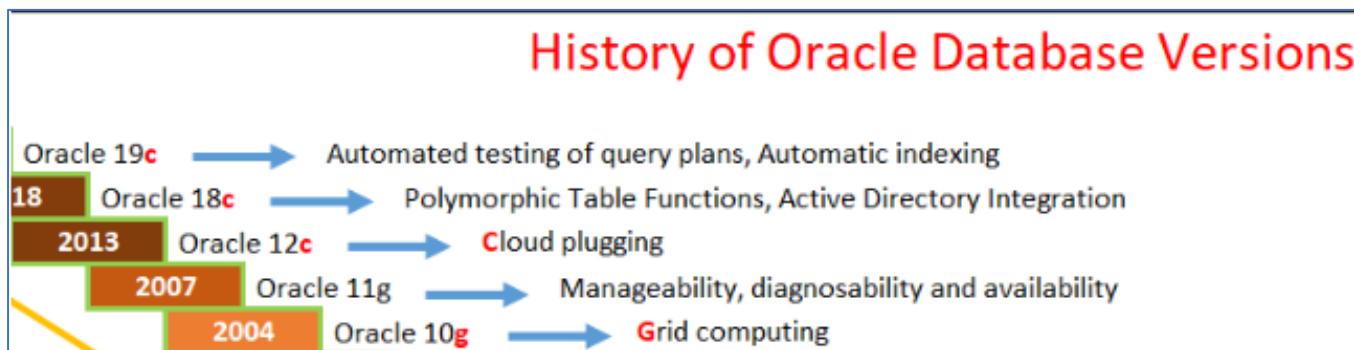
그림 3-4 사용자 그룹별로 원하는 정보

## 02. 오라클 설치와 기본 사용법

### ■ 오라클 18c Express Edition 설치(부록 A 참조)

- 오라클 DBMS를 내려받아 설치함 => <https://www.oracle.com/kr/database/technologies/xe-downloads.html>  
(C:\app\[사용자]\product\18.0.0\dbhomeXE 폴더)
- 시스템 관리자 계정 : system, 비밀번호 : Manager1 (비밀번호에 대문자와 숫자가 필요)
- [시작]-[Oracle-OraDB18\_home1]-[응용 프로그램 개발] 메뉴에 포함된 SQL Plus 프로그램 사용 가능

오라클버전	무료버전
18c XE (Express Edition)	



- <여기서 잠깐> 오라클 18c 설치 소요시간  
오라클 18c 버전은 설치 파일이 용량이 2G가 넘기 때문에 다운로드부터 설치까지 수분~수십분 이상이 소요
- <여기서 잠깐> 설치에 문제가 생기는 여러 원인들  
컴퓨터 이름이 한글로 되어 있을 때  
오라클을 한번 설치한 적이 있을 경우
  - oracle home에서 ./deinstall/deinstall 실행
  - regedit를 이용하여 기록을 삭제(방법 명령창에서 "sc delete OracleServiceXE"를 수행)

## 02. 오라클 설치와 기본 사용법

### ■ 샘플 데이터베이스 설치(부록 B.3~B4 참조)

- madang 데이터베이스(본 교재에서 사용하는 데이터베이스)
- madang 사용자 계정 및 샘플 데이터베이스 설치 : B.3
- (스크립트를 실행한다 - [demo\\_madang.sql](#))

### ■ emp(employee) 데이터베이스(오라클에서 만들어 교육용으로 사용하는 데이터베이스)

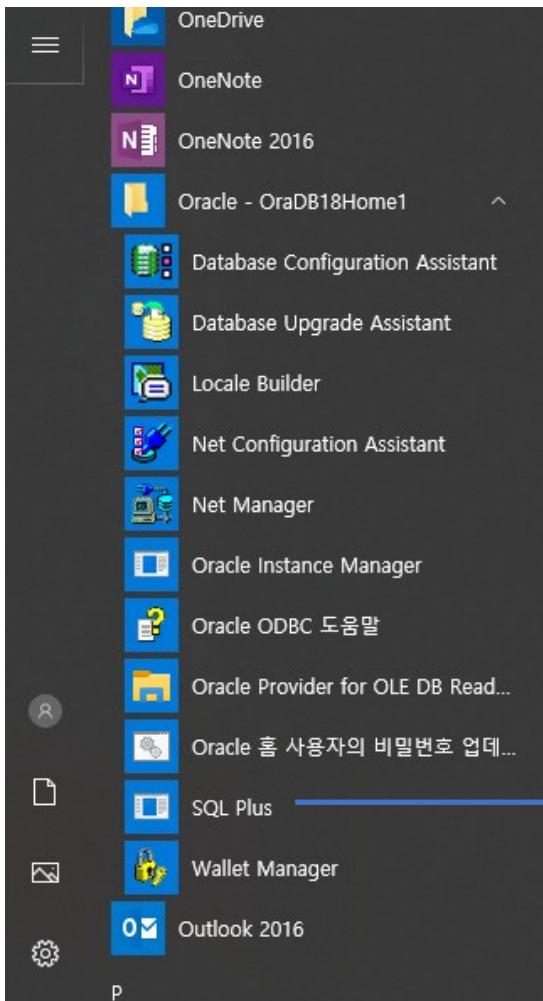
- 사용자 계정은 scott : B.4
- 예제로 emp와 dept 테이블 포함
- (스크립트를 실행한다 - [demo\\_scott.sql](#))
- 이전 버전들에서는 scott 계정이 기본으로 설치되어 있는 경우가 있다.  
(계정 사용(해제) 명령 : ALTER USER c##scott ACCOUNT UNLOCK)

### ■ SQL Developer 설치(부록 B.1 참조)

- 오라클 홈페이지에서 본인의 환경에 맞는 버전 다운로드하여 설치

# SQL Plus

## ① SQL Plus 시작



## ② 쿼리창 열기

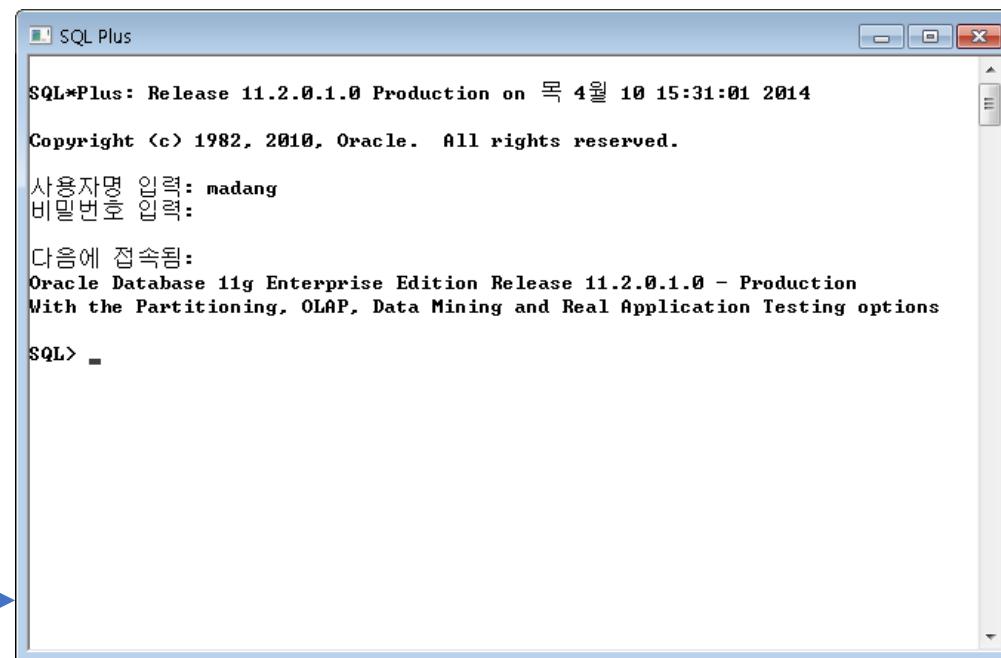


그림 3-5 SQL Plus 시작

# 1.2 마당서점의 데이터

## ■ SQL 문을 작성할 때 주로 사용하는 명령어.

- <Tip> SQL Plus에서 사용하는 명령어에 관한 자세한 설명은 다음의 링크를 참고한다.
  - [http://docs.oracle.com/cd/E11882\\_01/server.112/e16604/ch\\_twelve001.htm](http://docs.oracle.com/cd/E11882_01/server.112/e16604/ch_twelve001.htm) </Tip>
- -데이터베이스 접속 : **conn**
  - [예] conn scott/tiger : scott 계정에 비밀번호 tiger로 접속
- - 명령어 실행 : **run**, /
  - [예] run : 바로 전에 실행했던 명령어를 다시 실행
  - [예] / : run과 같은 의미
- - 명령어 찾기 : **list**
  - [예] list : 마지막에 수행했던 명령어를 출력한다. 직전 명령줄이 길 경우 편리
- - 메모장을 이용하여 명령어 작성 및 실행하기 : **ed** <파일이름>, **run** <파일이름>
  - [예] ed test : test.sql 이름의 파일이 메모장을 이용하여 작성할 수 있도록 열림
  - [예] start test : test.sql 이름에 저장된 명령어 스크립트가 실행
  - [예] @ test : start test와 같은 의미
- - 출력 모양을 조절하는 명령 : **column**
  - [예] column bookname format a20 : bookname을 길이 20의 문자 포맷으로 출력
  - [예] column price format 999999 : price를 길이 6개의 숫자 포맷으로 출력

# SQL Developer

## ■ SQL Developer 설치

- 오라클에서 내려받아 설치함 => <https://www.oracle.com/tools/downloads/sqldev-downloads.html>
- 임의의 폴더에 압축파일을 해제하고 실행은 폴더에 포함된 sqldeveloper.exe 파일을 실행시킨다.  
(단축아이콘을 만들어두면 편리하다)

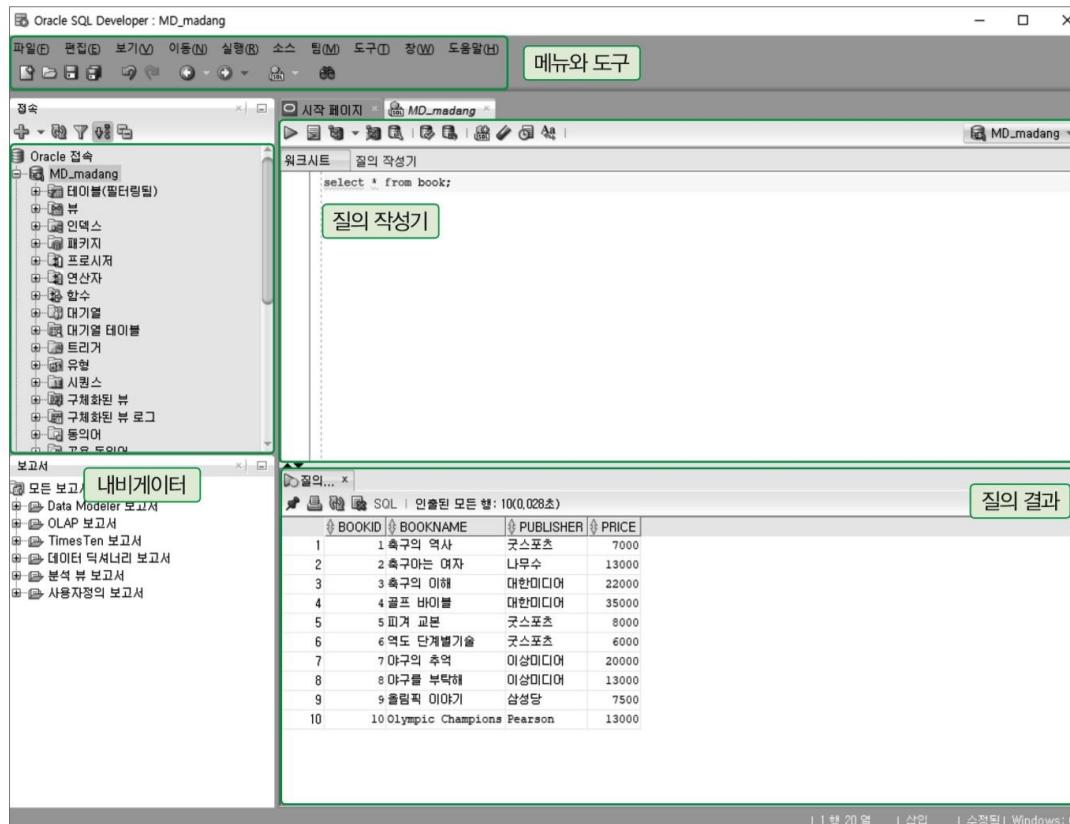


그림 3-6 SQL Developer에서 SQL 문을 실행한 화면

# SQL Developer

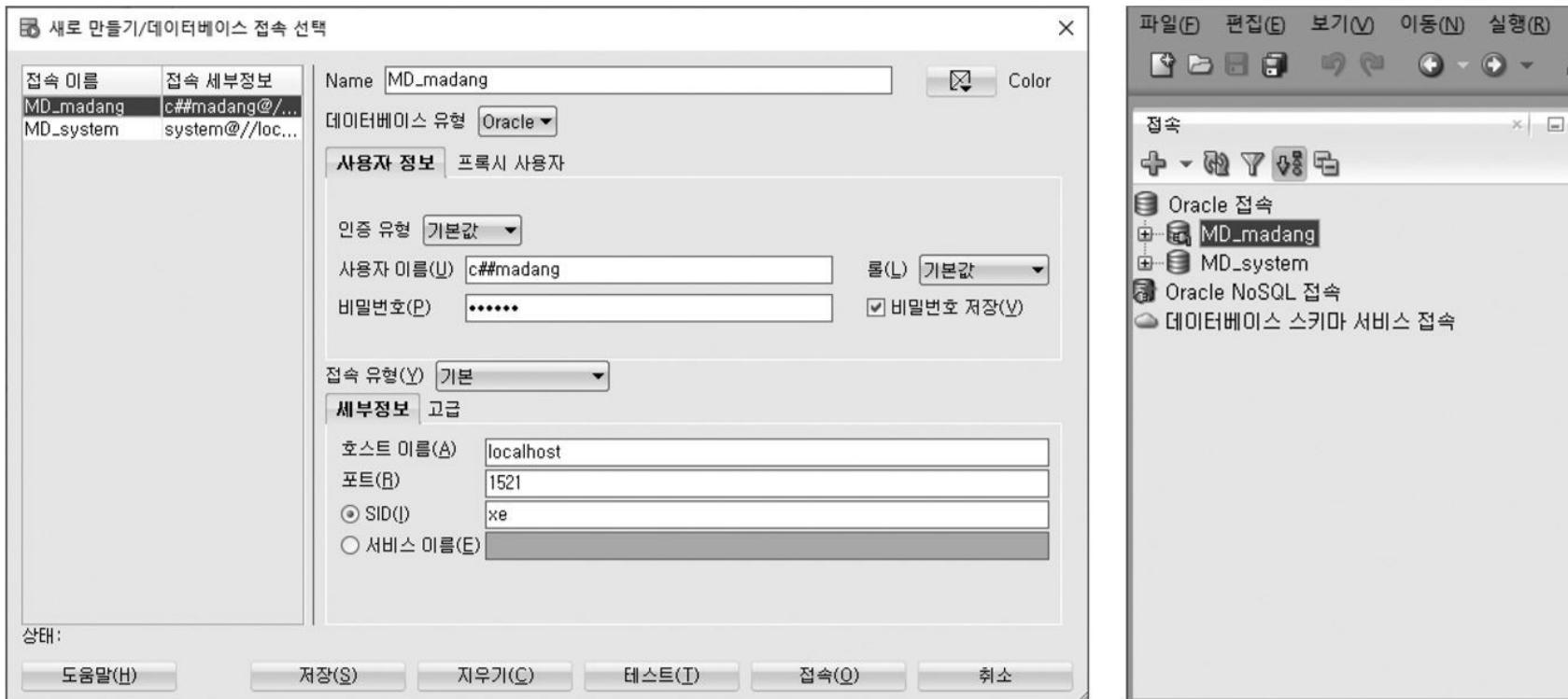


그림 3-7 접속 아이콘 생성 화면

## 03. SQL 소개

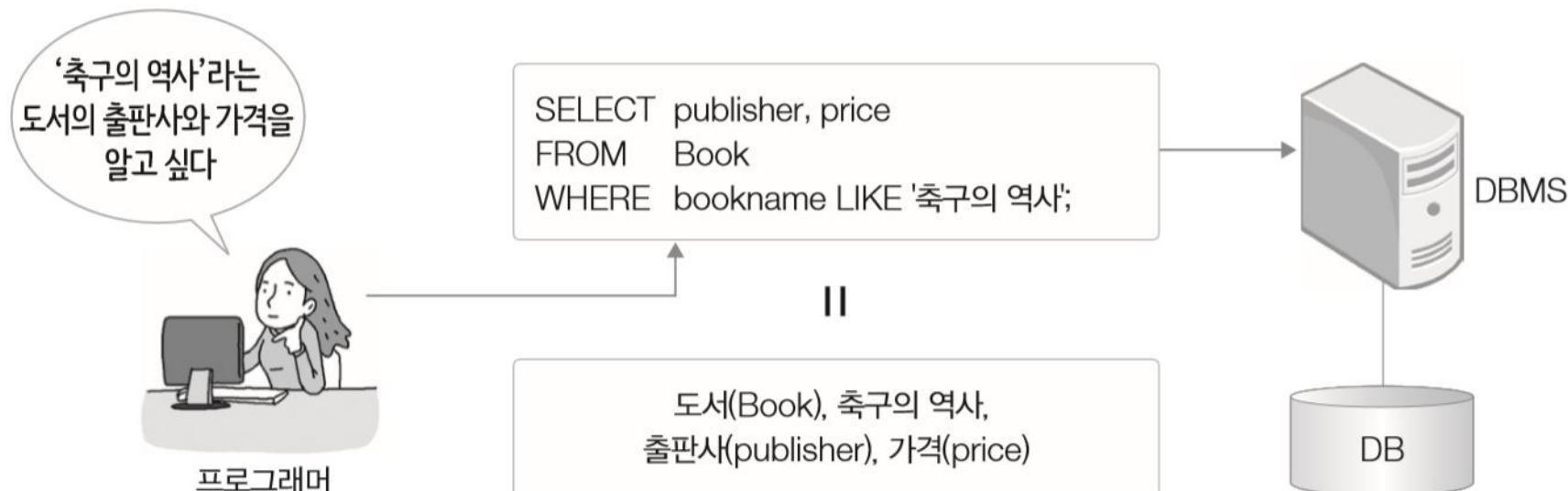


그림 3-8 SQL을 사용해 자료를 찾는 과정

## 03. SQL 소개

표 3-2 SQL과 일반 프로그래밍 언어의 차이점

구분	SQL	일반 프로그래밍 언어
용도	데이터베이스에서 데이터를 추출하여 문제 해결	모든 문제 해결
입출력	입력은 테이블, 출력도 테이블	모든 형태의 입출력 가능
번역	DBMS	컴파일러
문법	SELECT * FROM Book;	int main( ) { ... }

# 03. SQL 소개

## ■ SQL 기능에 따른 분류

- 데이터 정의어(**DDL**)
  - 테이블이나 관계의 구조를 생성하는 데 사용
  - CREATE, ALTER, DROP 문 등이 있음.
- 데이터 조작어(**DML**)
  - 테이블에 데이터를 검색, 삽입, 수정, 삭제하는 데 사용
  - SELECT, INSERT, DELETE, UPDATE 문 등이 있음.
    - SELECT 문은 특별히 질의어(query)라고 함.
- 데이터 제어어(**DCL**)
  - 데이터의 사용 권한을 관리하는 데 사용
  - GRANT, REVOKE 문 등이 있음.

## 03. SQL 소개

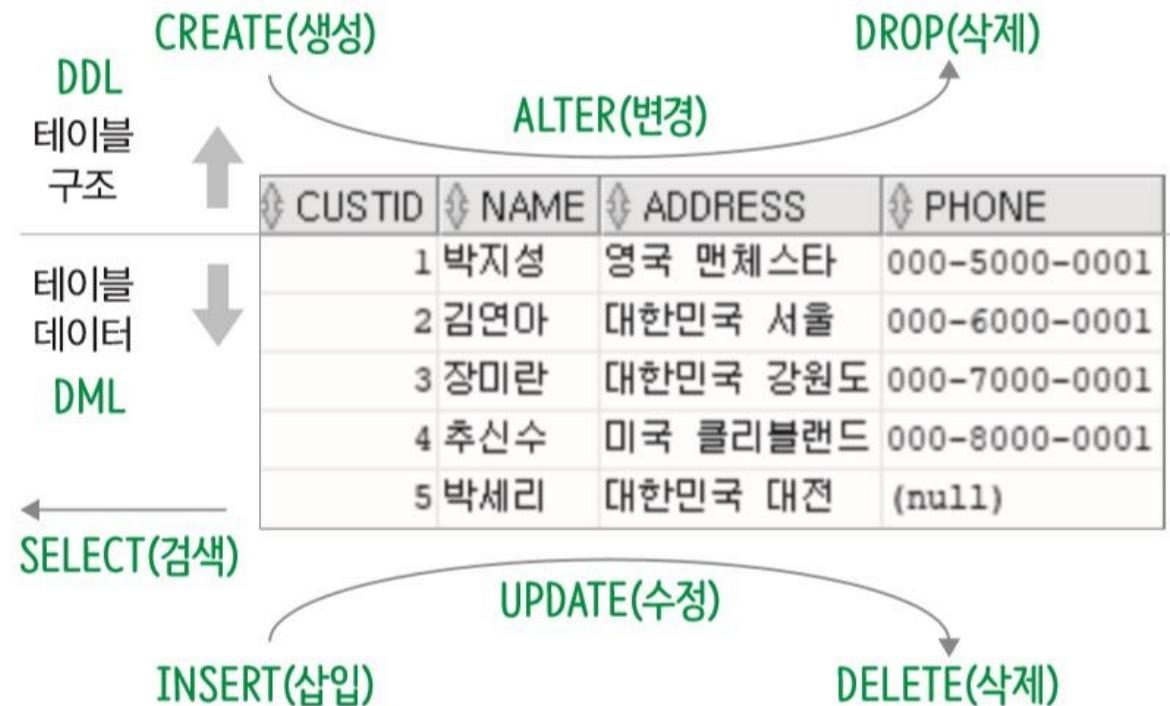


그림 3-9 데이터 정의어와 데이터 조작어의 주요 명령어

# 03. SQL 소개

예) 김연아 고객의 전화번호를 찾으시오.

```
SELECT      phone  
FROM        Customer  
WHERE       name='김연아';
```

SELECT: 질의 결과 추출되는 속성 리스트를 열거한다.

FROM: 질의에 어느 테이블이 사용되는지 열거한다.

WHERE: 질의의 조건을 작성한다.

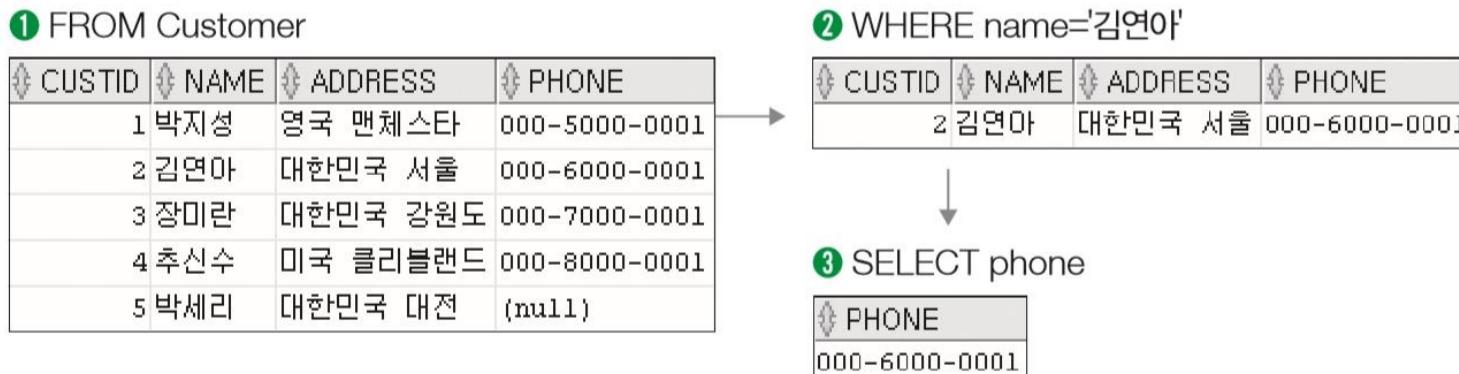


그림 3-10 SQL 문의 내부적 실행 순서

## 04. 데이터 조작어 - 검색

---

- SELECT 문
- 집계 함수와 GROUP BY
- 두 개 이상 테이블에서 SQL 질의

# 04. 데이터 조작어 - 검색

## ■ SELECT 문의 구성 요소

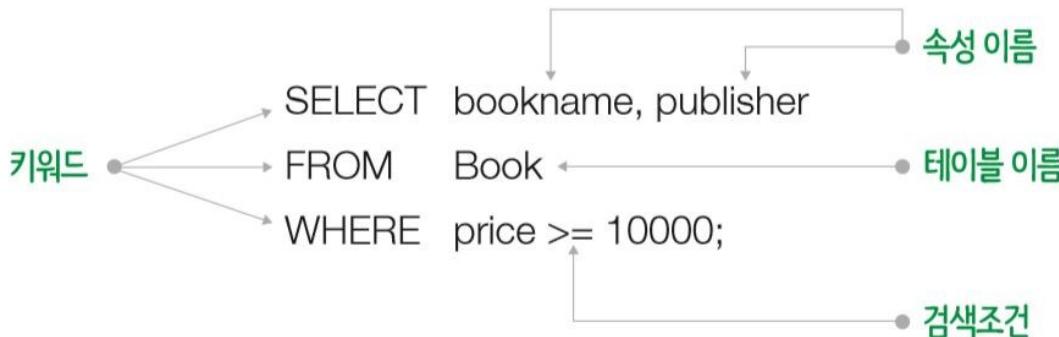


그림 3-11 SELECT 문의 예

## ■ SELECT 문의 기본 문법

```
SELECT [ALL | DISTINCT] 속성이름(들)
FROM      테이블이름(들)
[WHERE    검색조건(들)]
[GROUP BY 속성이름]
[HAVING   검색조건(들)]
[ORDER BY 속성이름 [ASC | DESC]]
```

[ ]: 대괄호 안의 SQL 예약어들은 선택적으로 사용한다.

|: 선택 가능한 문법들 중 한 개를 사용할 수 있다.

# 04. 데이터 조작어 - 검색

## ■ SELECT 문의 기본 문법

```
SELECT
    [ALL | DISTINCT]
    [테이블이름.] { * | 속성 이름 [[ AS ] 속성 이름 별칭] }
[FROM
    { 테이블 이름 [AS 테이블 이름 별칭] }

    [INNER JOIN | LEFT [OUTER] JOIN | RIGHT [OUTER] JOIN
    { 테이블 이름 [ON 검색 조건] }
    | FULL [OUTER] JOIN { 테이블 이름 } ]
[WHERE 검색 조건(들)]
[GROUP BY { 속성 이름, [..., n] }]
[HAVING 검색 조건(들)]
[질의 UNION 질의 | 질의 UNION ALL 질의]
[ORDER BY { 속성 이름 [ASC | DESC], [..., n] }]
```

[ ]: 대괄호 안의 SQL 예약어들은 선택적으로 사용한다.

{ }: 중괄호 안의 SQL 예약어들은 필수적으로 사용한다.

|: 선택 가능한 문법 중 한 개를 사용할 수 있다.

# SELECT 문 예제

질의 3-1 모든 도서의 이름과 가격을 검색하시오

```
SELECT bookname, price  
FROM Book;
```

BOOKNAME	PRICE
축구의 역사	7000
축구마는 여자	13000
축구의 이해	22000
골프 바이블	35000
피겨 교본	8000
역도 단계별기술	6000
야구의 추억	20000
야구를 부탁해	13000
올림픽 이야기	7500
Olympic Champions	13000

질의 3-1 변형 모든 도서의 이름과 가격을 검색하시오

```
SELECT price, bookname  
FROM Book;
```

PRICE	BOOKNAME
7000	축구의 역사
13000	축구마는 여자
22000	축구의 이해
35000	골프 바이블
8000	피겨 교본
6000	역도 단계별기술
20000	야구의 추억
13000	야구를 부탁해
7500	올림픽 이야기
13000	Olympic Champions

# SELECT 문 예제

질의 3-2 모든 도서의 도서번호, 도서이름, 출판사, 가격을 검색하시오.

```
SELECT bookid, bookname, publisher, price  
FROM Book;
```

```
SELECT *  
FROM Book;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
2	축구마는 여자	나무수	13000
3	축구의 미해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	파겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	미상미디어	20000
8	야구를 부탁해	미상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

# SELECT 문 예제

질의 3-3 도서 테이블에 있는 모든 출판사를 검색하시오.

```
SELECT publisher  
FROM Book;
```

PUBLISHER
굿스포츠
나무수
대한미디어
대한미디어
굿스포츠
굿스포츠
미상미디어
미상미디어
삼성당
Pearson

※ 중복을 제거하고 싶으면 DISTINCT 키워드를 사용

```
SELECT DISTINCT publisher  
FROM Book;
```

PUBLISHER
굿스포츠
삼성당
대한미디어
Pearson
나무수
미상미디어

# 조건 검색 WHERE

표 3-3 WHERE 절에 조건으로 사용할 수 있는 술어

술어	연산자	사용 예
비교	=, <>, <, <=, >, >=	price < 20000
범위	BETWEEN	price BETWEEN 10000 AND 20000
집합	IN, NOT IN	price IN (10000, 20000, 30000)
패턴	LIKE	bookname LIKE '축구의 역사'
NULL	IS NULL, IS NOT NULL	price IS NULL
복합조건	AND, OR, NOT	(price < 20000) AND (bookname LIKE '축구의 역사')

## ■ 비교

질의 3-4 가격이 20,000원 미만인 도서를 검색하시오.

```
SELECT *
FROM Book
WHERE price < 20000;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
2	축구마는 여자	나무수	13000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

# 조건 검색 WHERE

## ■ 비교

질의 3-5 가격이 10,000원 이상 20,000 이하인 도서를 검색하시오.

```
SELECT *  
FROM Book  
WHERE price BETWEEN 10000 AND 20000;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
2	축구하는 여자	나무수	13000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
10	Olympic Champions	Pearson	13000

```
SELECT *  
FROM Book  
WHERE price >= 10000 AND price <= 20000;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
2	축구하는 여자	나무수	13000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
10	Olympic Champions	Pearson	13000

# 조건 검색 WHERE

## ■ 집합

질의 3-6 출판사가 '굿스포츠' 혹은 '대한미디어'인 도서를 검색하시오.

```
SELECT *
FROM Book
WHERE publisher IN ('굿스포츠', '대한미디어');
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000

## ■ 출판사가 '굿스포츠' 혹은 '대한미디어'가 아닌 도서를 검색

```
SELECT *
FROM Book
WHERE publisher NOT IN ('굿스포츠', '대한미디어');
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
2	축구마는 여자	나무수	13000
7	야구의 추억	미상미디어	20000
8	야구를 부탁해	미상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

# 조건 검색 WHERE

## ■ 패턴

질의 3-7 '축구의 역사'를 출간한 출판사를 검색하시오.

```
SELECT bookname, publisher  
FROM Book  
WHERE bookname LIKE '축구의 역사';
```

BOOKNAME	PUBLISHER
축구의 역사	굿스포츠

질의 3-8 도서이름에 '축구'가 포함된 출판사를 검색하시오.

```
SELECT bookname, publisher  
FROM Book  
WHERE bookname LIKE '%축구%';
```

BOOKNAME	PUBLISHER
축구의 역사	굿스포츠
축구하는 여자	나무수
축구의 이해	대한미디어

# 조건 검색 WHERE

## ■ 패턴

질의 3-9 도서이름의 왼쪽 두 번째 위치에 '구'라는 문자열을 갖는 도서를 검색하시오.

```
SELECT *
FROM Book
WHERE bookname LIKE '_구%';
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000

표 3-4 와일드 문자의 종류

와일드 문자	의미	사용 예
+	문자열을 연결	'골프' + '바이블': '골프 바이블'
%	0개 이상의 문자열과 일치	'%축구%': 축구를 포함하는 문자열
[ ]	한 개의 문자와 일치	'[0~5]%' : 0~5 사이 숫자로 시작하는 문자열
[^]	한 개의 문자와 불일치	'[^0~5]%' : 0~5 사이 숫자로 시작하지 않는 문자열
-	특정 위치의 한 개의 문자와 일치	'_구%': 두 번째 위치에 '구'가 들어가는 문자열

# 조건 검색 WHERE

## 복합조건

질의 3-10 축구에 관한 도서 중 가격이 20,000원 이상인 도서를 검색하시오.

```
SELECT *
FROM Book
WHERE bookname LIKE '%축구%' AND price >=20000;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
3	축구의 이해	대한미디어	22000

질의 3-11 출판사가 '굿스포츠' 혹은 '대한미디어'인 도서를 검색하시오.

```
SELECT *
FROM Book
WHERE publisher='굿스포츠' OR publisher='대한미디어';
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000

# 검색 결과의 정렬\_ORDER BY

질의 3-12 도서를 이름순으로 검색하시오.

```
SELECT *
FROM Book
ORDER BY bookname;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
10	Olympic Champions	Pearson	13000
4	골프 바이블	대한미디어	35000
8	야구를 부탁해	이상미디어	13000
7	야구의 추억	이상미디어	20000
6	역도 단계별기술	굿스포츠	6000
9	올림픽 이야기	삼성당	7500
2	축구마는 여자	나무수	13000
1	축구의 역사	굿스포츠	7000
3	축구의 이해	대한미디어	22000
5	Ⅱ 경 교본	굿스포츠	8000

# 검색 결과의 정렬\_ORDER BY

질의 3-13 도서를 가격순으로 검색하고, 가격이 같으면 이름순으로 검색하시오.

```
SELECT *
FROM Book
ORDER BY price, bookname;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
6	역도 단계별기술	굿스포츠	6000
1	축구의 역사	굿스포츠	7000
9	올림픽 이야기	삼성당	7500
5	피겨 교본	굿스포츠	8000
10	Olympic Champions Pearson	Pearson	13000
8	야구를 부탁해	이상미디어	13000
2	축구마는 여자	나무수	13000
7	야구의 추억	이상미디어	20000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000

# 검색 결과의 정렬 ORDER BY

질의 3-14 도서를 가격의 내림차순으로 검색하시오. 만약 가격이 같다면 출판사의 오름차순으로 검색하시오

```
SELECT *
FROM Book
ORDER BY price DESC, publisher ASC;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
4	골프 바이블	대한미디어	35000
3	축구의 이해	대한미디어	22000
7	야구의 추억	미상미디어	20000
10	Olympic Champions	Pearson	13000
2	축구하는 여자	나무수	13000
8	야구를 부탁해	미상미디어	13000
5	피겨 교본	굿스포츠	8000
9	올림픽 이야기	삼성당	7500
1	축구의 역사	굿스포츠	7000
6	역도 단계별기술	굿스포츠	6000

# 집계 함수

질의 3-15 고객이 주문한 도서의 총 판매액을 구하시오.

```
SELECT      SUM(saleprice)
FROM        Orders;
```

⌚ SUM(SALEPRICE)
118000

- 의미 있는 열 이름을 출력하고 싶으면 속성이름의 별칭을 지정하는 AS 키워드를 사용하여  
 열 이름을 부여

```
SELECT      SUM(saleprice) AS 총매출
FROM        Orders;
```

⌚ 총매출
118000

# 집계 함수

질의 3-16 2번 김연아 고객이 주문한 도서의 총 판매액을 구하시오.

```
SELECT      SUM(saleprice) AS 총매출  
FROM        Orders  
WHERE       custid=2;
```

총매출
15000

질의 3-17 고객이 주문한 도서의 총 판매액, 평균값, 최저가, 최고가를 구하시오.

```
SELECT      SUM(saleprice) AS Total,  
                  AVG(saleprice) AS Average,  
                  MIN(saleprice) AS Minimum,  
                  MAX(saleprice) AS Maximum  
FROM        Orders;
```

TOTAL	AVERAGE	MINIMUM	MAXIMUM
118000	11800	6000	21000

# 집계 함수

질의 3-18 마당서점의 도서 판매 건수를 구하시오.

```
SELECT COUNT(*)  
FROM Orders;
```

COUNT(*)
10

표 3-5 집계 함수의 종류

집계 함수	문법	사용 예
SUM	SUM([ALL   DISTINCT] 속성이름)	SUM(price)
AVG	AVG([ALL   DISTINCT] 속성이름)	AVG(price)
COUNT	COUNT({[ALL   DISTINCT] 속성이름]   *})	COUNT(*)
MAX	MAX([ALL   DISTINCT] 속성이름)	MAX(price)
MIN	MIN([ALL   DISTINCT] 속성이름)	MIN(price)

# GROUP BY 검색

질의 3-19 고객별로 주문한 도서의 총 수량과 총 판매액을 구하시오.

```
SELECT      custid, COUNT(*) AS 도서수량, SUM(saleprice) AS 총액  
FROM        Orders  
GROUP BY    custid;
```

CUSTID	도서수량	총액
1	3	39000
2	2	15000
4	2	33000
3	3	31000

ORDERID	CUSTID	BOOKID	SALEPRICE	ORDERDATE	CUSTID	도서수량	총액
2	1	3	21000	20/07/03	1	3	39000
6	1	2	12000	20/07/07	2	2	15000
1	1	1	6000	20/07/01	3	3	31000
9	2	10	7000	20/07/09	4	2	33000
3	2	5	8000	20/07/03			
4	3	6	6000	20/07/04			
10	3	8	13000	20/07/10			
8	3	10	12000	20/07/08			
7	4	8	13000	20/07/07			
5	4	7	20000	20/07/05			

그림 3-12 GROUP BY 질의 수행

# GROUP BY 검색

## ■ HAVING 절

질의 3-20 가격이 8,000원 이상인 도서를 구매한 고객에 대하여 고객별 주문 도서의 총 수량을 구하시오. 단, 두 권 이상 구매한 고객만 구한다.

```
SELECT      custid, COUNT(*) AS 도서수량
FROM        Orders
WHERE       saleprice >=8000
GROUP BY    custid
HAVING     count(*) >=2;
```

CUSTID	도서수량
1	2
4	2
3	2

# GROUP BY 검색

## ■ GROUP BY와 HAVING 절의 문법과 주의사항

표 3-6 GROUP BY와 HAVING 절의 문법과 주의사항

GROUP BY <속성>	
주의사항	GROUP BY로 투플을 그룹으로 묶은 후 SELECT 절에는 GROUP BY에서 사용한 <속성>과 집계 함수만 나올 수 있다.
맞는 예	<pre>SELECT      custid, SUM(saleprice) FROM        Orders GROUP BY    custid;</pre>
틀린 예	<pre>SELECT      bookid, SUM(saleprice) /* SELECT 절에 bookid 속성이 올 수 없다 */ FROM        Orders GROUP BY    custid;</pre>
HAVING <검색조건>	
주의사항	WHERE 절과 HAVING 절이 같이 포함된 SQL 문은 검색조건이 모호해질 수 있다. HAVING 절은 ① 반드시 GROUP BY 절과 같이 작성해야 하고 ② WHERE 절보다 뒤에 나와야 한다. 그리고 ③ <검색조건>에는 SUM, AVG, MAX, MIN, COUNT와 같은 집계 함수가 와야 한다.
맞는 예	<pre>SELECT      custid, COUNT(*) AS 도서수량 FROM        Orders WHERE       saleprice &gt;= 8000 GROUP BY   custid HAVING     count(*) &gt;= 2;</pre>
틀린 예	<pre>SELECT      custid, COUNT(*) AS 도서수량 FROM        Orders HAVING     count(*) &gt;= 2 /* 순서가 틀렸다 */ WHERE       saleprice &gt;= 8000 GROUP BY   custid;</pre>

# 연습문제

## 1. 마당서점의 고객이 요구하는 다음 질문에 대해 SQL 문을 작성하시오.

- (1) 도서번호가 1인 도서의 이름
- (2) 가격이 20,000원 이상인 도서의 이름
- (3) 박지성의 총 구매액(박지성의 고객번호는 1번으로 놓고 작성)
- (4) 박지성이 구매한 도서의 수(박지성의 고객번호는 1번으로 놓고 작성)

## 2. 마당서점의 운영자와 경영자가 요구하는 다음 질문에 대해 SQL 문을 작성하시오.

- (1) 마당서점 도서의 총 개수
- (2) 마당서점에 도서를 출고하는 출판사의 총 개수
- (3) 모든 고객의 이름, 주소
- (4) 2014년 7월 4일~7월 7일 사이에 주문 받은 도서의 주문번호
- (5) 2014년 7월 4일~7월 7일 사이에 주문 받은 도서를 제외한 도서의 주문번호
- (6) 성이 '김' 씨인 고객의 이름과 주소
- (7) 성이 '김' 씨이고 이름이 '아'로 끝나는 고객의 이름과 주소

## 3.4 두 개 이상 테이블에서 SQL 질의

### ■ Customer 테이블을 Orders 테이블과 조건 없이 연결

- Customer와 Orders 테이블의 합체 결과 투플의 개수는 고객이 다섯 명이고 주문이 열 개이므로  $5 \times 10$  해서 50이 됨

```
SELECT      *
FROM        Customer, Orders;
```

	CUSTID	NAME	ADDRESS	PHONE	ORDERID	CUSTID_1	BOOKID	SALEPRICE	ORDERDATE
1	1	박지성	영국 맨체스터	000-5000-0001	1	1	1	6000	20/07/01
2	1	박지성	영국 맨체스터	000-5000-0001	2	1	3	21000	20/07/03
3	1	박지성	영국 맨체스터	000-5000-0001	3	2	5	8000	20/07/03
4	1	박지성	영국 맨체스터	000-5000-0001	4	3	6	6000	20/07/04
5	1	박지성	영국 맨체스터	000-5000-0001	5	4	7	20000	20/07/05
6	1	박지성	영국 맨체스터	000-5000-0001	6	1	2	12000	20/07/07
7	1	박지성	영국 맨체스터	000-5000-0001	7	4	8	13000	20/07/07
8	1	박지성	영국 맨체스터	000-5000-0001	8	3	10	12000	20/07/08
9	1	박지성	영국 맨체스터	000-5000-0001	9	2	10	7000	20/07/09
10	1	박지성	영국 맨체스터	000-5000-0001	10	3	8	13000	20/07/10
11	2	김연아	대한민국 서울	000-6000-0001	1	1	1	6000	20/07/01
12	2	김연아	대한민국 서울	000-6000-0001	2	1	3	21000	20/07/03
13	2	김연아	대한민국 서울	000-6000-0001	3	2	5	8000	20/07/03
14	2	김연아	대한민국 서울	000-6000-0001	4	3	6	6000	20/07/04
15	2	김연아	대한민국 서울	000-6000-0001	5	4	7	20000	20/07/05
16	2	김연아	대한민국 서울	000-6000-0001	6	1	2	12000	20/07/07
17	2	김연아	대한민국 서울	000-6000-0001	7	4	8	13000	20/07/07
18	2	김연아	대한민국 서울	000-6000-0001	8	3	10	12000	20/07/08
19	2	김연아	대한민국 서울	000-6000-0001	9	2	10	7000	20/07/09
20	2	김연아	대한민국 서울	000-6000-0001	10	3	8	13000	20/07/10
... 연속됨 ...									
45	5	박세리	대한민국 대전	(null)	5	4	7	20000	20/07/05
46	5	박세리	대한민국 대전	(null)	6	1	2	12000	20/07/07
47	5	박세리	대한민국 대전	(null)	7	4	8	13000	20/07/07
48	5	박세리	대한민국 대전	(null)	8	3	10	12000	20/07/08
49	5	박세리	대한민국 대전	(null)	9	2	10	7000	20/07/09
50	5	박세리	대한민국 대전	(null)	10	3	8	13000	20/07/10

그림 3-13 Customer와 Orders 테이블의 합체

# 조인

질의 3-21 고객과 고객의 주문에 관한 데이터를 모두 보이시오.

```
SELECT *
FROM Customer, Orders
WHERE Customer.custid=Orders.custid;
```

CUSTID	NAME	ADDRESS	PHONE	ORDERID	CUSTID_1	BOOKID	SALEPRICE	ORDERDATE
1	박지성	영국 맨체스터	000-5000-0001	2	1	3	21000	20/07/03
1	박지성	영국 맨체스터	000-5000-0001	6	1	2	12000	20/07/07
1	박지성	영국 맨체스터	000-5000-0001	1	1	1	6000	20/07/01
2	김연아	대한민국 서울	000-6000-0001	9	2	10	7000	20/07/09
2	김연아	대한민국 서울	000-6000-0001	3	2	5	8000	20/07/03
3	장미란	대한민국 강원도	000-7000-0001	4	3	6	6000	20/07/04
3	장미란	대한민국 강원도	000-7000-0001	10	3	8	13000	20/07/10
3	장미란	대한민국 강원도	000-7000-0001	8	3	10	12000	20/07/08
4	추신수	미국 클리블랜드	000-8000-0001	7	4	8	13000	20/07/07
4	추신수	미국 클리블랜드	000-8000-0001	5	4	7	20000	20/07/05

# 조인

질의 3-22 고객과 고객의 주문에 관한 데이터를 고객번호 순으로 정렬하여 보이시오.

```
SELECT      *
FROM        Customer, Orders
WHERE       Customer.custid=Orders.custid
ORDER BY    Customer.custid;
```

CUSTID	NAME	ADDRESS	PHONE	ORDERID	CUSTID_1	BOOKID	SALEPRICE	ORDERDATE
1	박지성	영국 맨체스터	000-5000-0001	2	1	3	21000	20/07/03
1	박지성	영국 맨체스터	000-5000-0001	6	1	2	12000	20/07/07
1	박지성	영국 맨체스터	000-5000-0001	1	1	1	6000	20/07/01
2	김연마	대한민국 서울	000-6000-0001	9	2	10	7000	20/07/09
2	김연마	대한민국 서울	000-6000-0001	3	2	5	8000	20/07/03
3	장미란	대한민국 강원도	000-7000-0001	4	3	6	6000	20/07/04
3	장미란	대한민국 강원도	000-7000-0001	10	3	8	13000	20/07/10
3	장미란	대한민국 강원도	000-7000-0001	8	3	10	12000	20/07/08
4	추신수	미국 클리블랜드	000-8000-0001	7	4	8	13000	20/07/07
4	추신수	미국 클리블랜드	000-8000-0001	5	4	7	20000	20/07/05

# 조인

질의 3-23 고객의 이름과 고객이 주문한 도서의 판매가격을 검색하시오.

```
SELECT      name, saleprice  
FROM        Customer, Orders  
WHERE       Customer.custid=Orders.custid;
```

NAME	SALEPRICE
박지성	21000
박지성	12000
박지성	6000
김연아	7000
김연아	8000
장미란	6000
장미란	13000
장미란	12000
추신수	13000
추신수	20000

질의 3-24 고객별로 주문한 모든 도서의 총 판매액을 구하고, 고객별로 정렬하시오.

```
SELECT      name, SUM(saleprice)  
FROM        Customer, Orders  
WHERE       Customer.custid=Orders.custid  
GROUP BY   Customer.name  
ORDER BY   Customer.name;
```

NAME	SUM(SALEPRICE)
김연아	15000
박지성	39000
장미란	31000
추신수	33000

# 조인

다음과 같은 질의는 어떻게 구할 수 있는가?

- “고객의 이름과 고객이 주문한 도서의 이름을 구하시오”

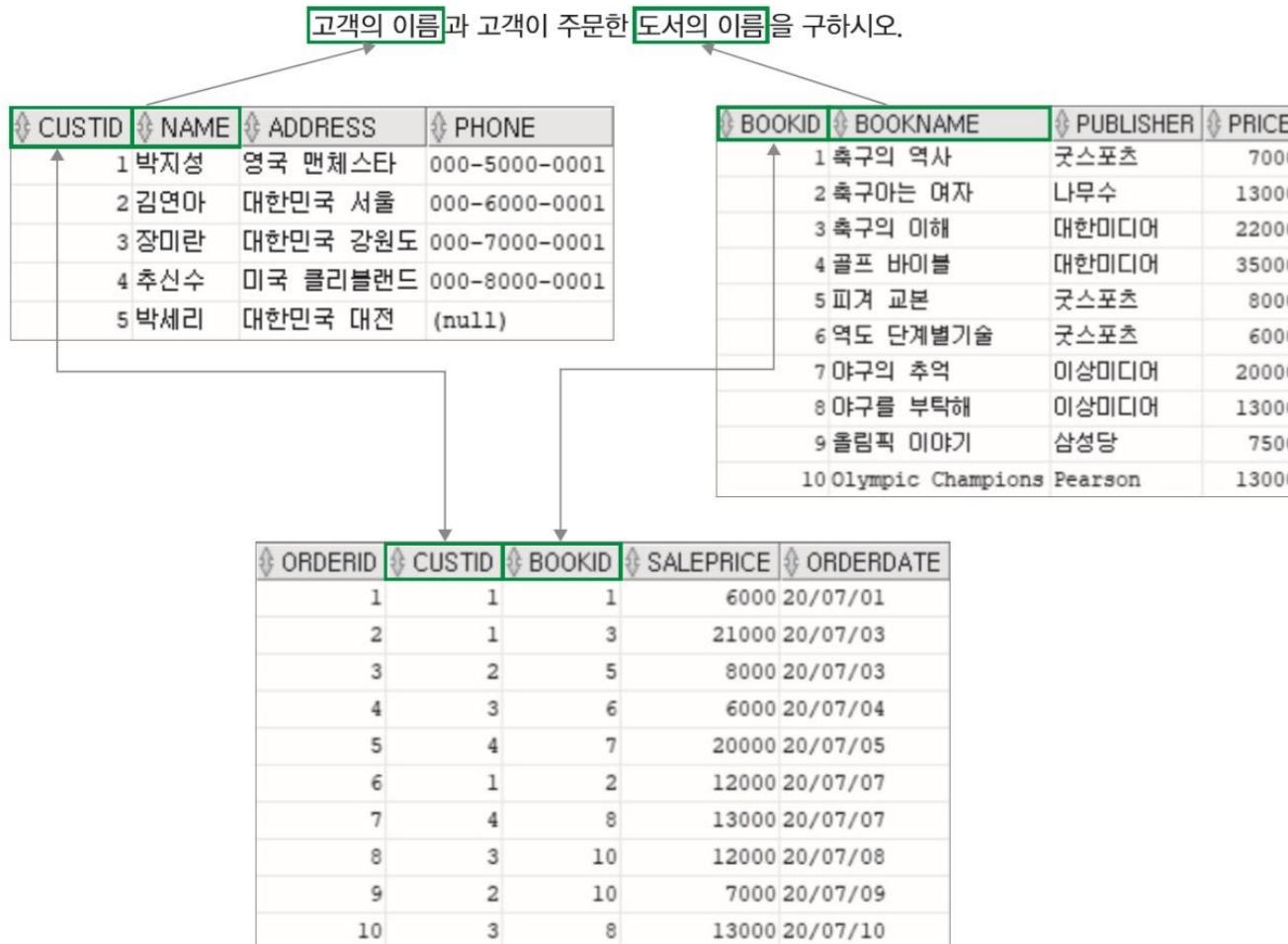


그림 3-14 마당서점 데이터 간의 연결

# 조인

질의 3-25 고객의 이름과 고객이 주문한 도서의 이름을 구하시오.

```
SELECT Customer.name, book.bookname  
FROM Customer, Orders, Book  
WHERE Customer.custid=Orders.custid AND Orders.bookid=Book.bookid;
```

NAME	BOOKNAME
박지성	축구의 역사
박지성	축구마는 여자
박지성	축구의 이해
김연아	피겨 교본
장미란	역도 단계별기술
추신수	야구의 추억
장미란	야구를 부탁해
추신수	야구를 부탁해
김연아	Olympic Champions
장미란	Olympic Champions

질의 3-26 가격이 20,000원인 도서를 주문한 고객의 이름과 도서의 이름을 구하시오.

```
SELECT Customer.name, book.bookname  
FROM Customer, Orders, Book  
WHERE Customer.custid=Orders.custid AND Orders.bookid=Book.bookid  
      AND Book.price=20000;
```

NAME	BOOKNAME
추신수	야구의 추억

# 조인

## ■ 외부조인

질의 3-27 도서를 구매하지 않은 고객을 포함하여 고객의 이름과 고객이 주문한 도서의 판매가격을 구하시오.

```
SELECT Customer.name, saleprice  
FROM Customer LEFT OUTER JOIN Orders  
    ON Customer.custid=Orders.custid;
```

NAME	SALEPRICE
박지성	21000
박지성	12000
박지성	6000
김연아	7000
김연아	8000
장미란	6000
장미란	13000
장미란	12000
추신수	13000
추신수	20000
박세리	(null)

# 조인

표 3-7 조인 문법

명령	문법
내부조인	SELECT <속성들> FROM 테이블 1, 테이블 2 WHERE <조인조건> AND <검색조건>
	SELECT <속성들> FROM 테이블 1 INNER JOIN 테이블 2 ON <조인조건> WHERE <검색조건>
외부조인	SELECT <속성들> FROM 테이블 1 {LEFT   RIGHT   FULL [OUTER]} JOIN 테이블 2 ON <조인조건> WHERE <검색조건>

# 부속질의

질의 3-28 가장 비싼 도서의 이름을 보이시오.

```
SELECT bookname  
FROM Book  
WHERE price=(SELECT MAX(price)  
              FROM Book);
```

BOOKNAME
골프 바이블

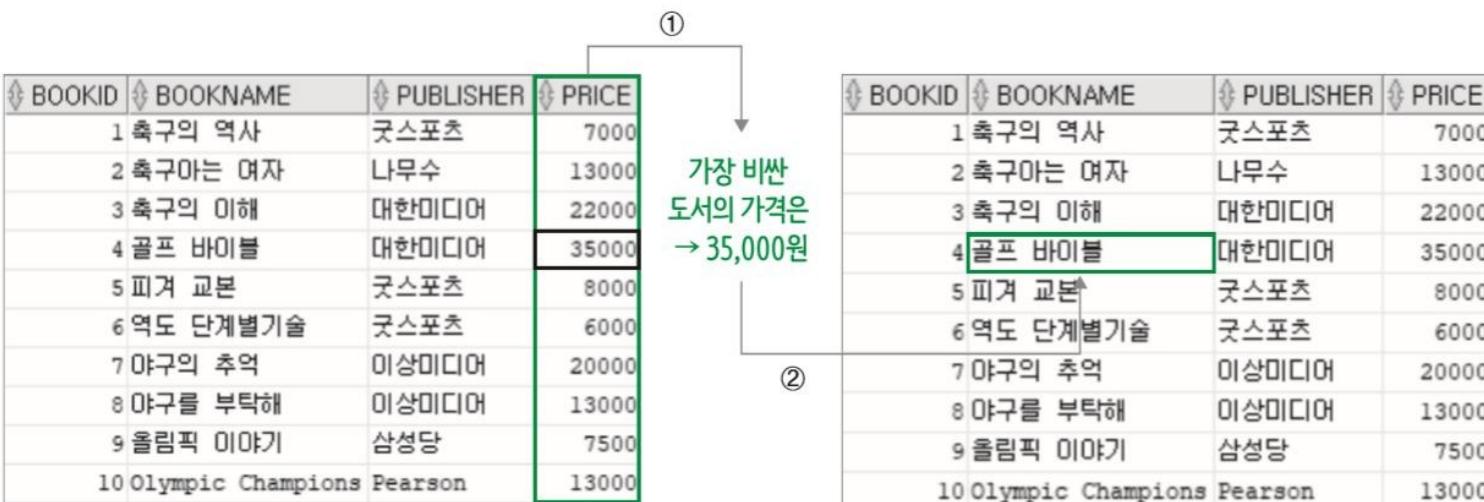


그림 3-15 부속질의의 실행 순서

# 부속질의

질의 3-29 도서를 구매한 적이 있는 고객의 이름을 검색하시오.

```
SELECT      name
FROM        Customer
WHERE       custid IN (SELECT custid
                      FROM Orders);
```

NAME
박지성
김연아
장미란
추신수

질의 3-30 대한미디어에서 출판한 도서를 구매한 고객의 이름을 보이시오.

```
SELECT      name
FROM        Customer
WHERE       custid IN(SELECT      custid
                      FROM        Orders
                      WHERE       bookid IN(SELECT      bookid
                                         FROM        Book
                                         WHERE       publisher='대한미디어'));
```

NAME
박지성

# 부속질의

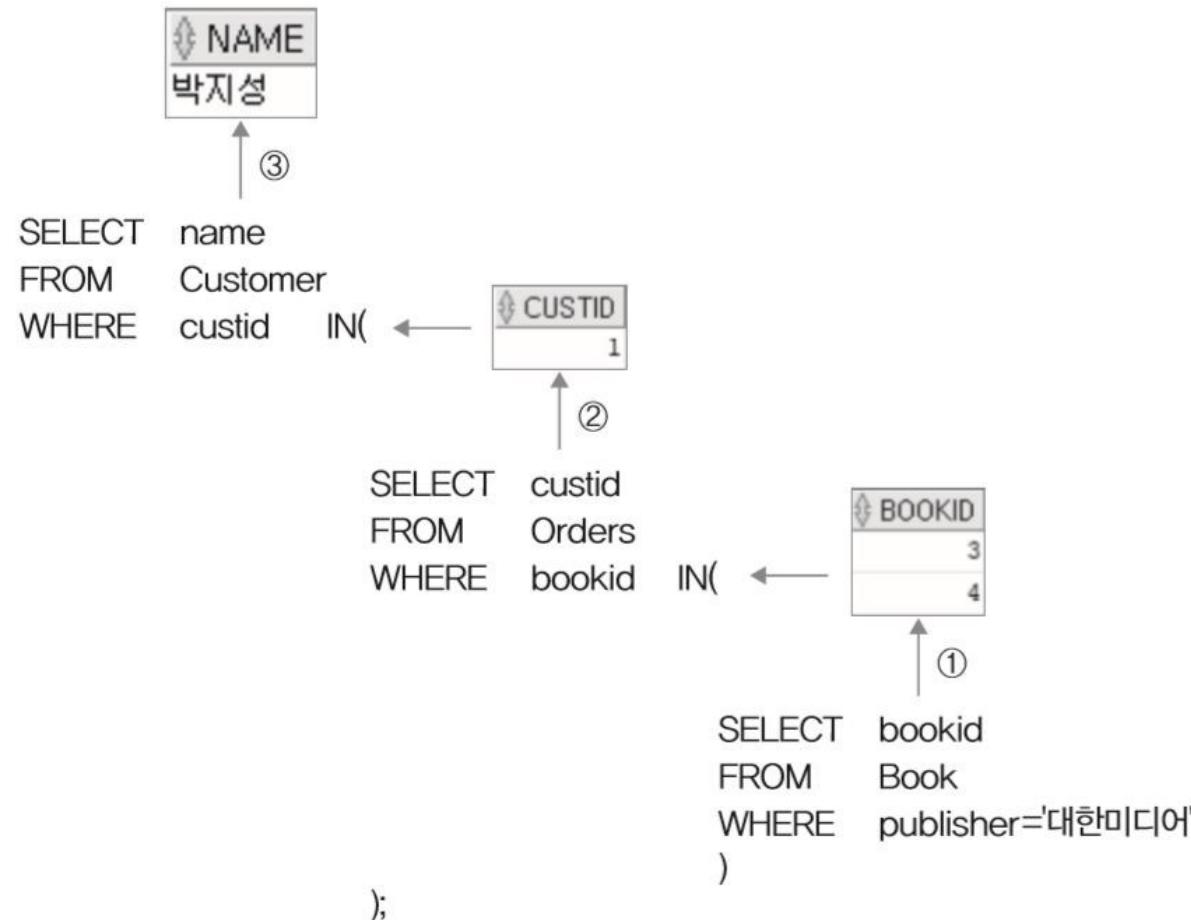


그림 3-16 3단계 부속질의의 실행 순서

# 부속질의

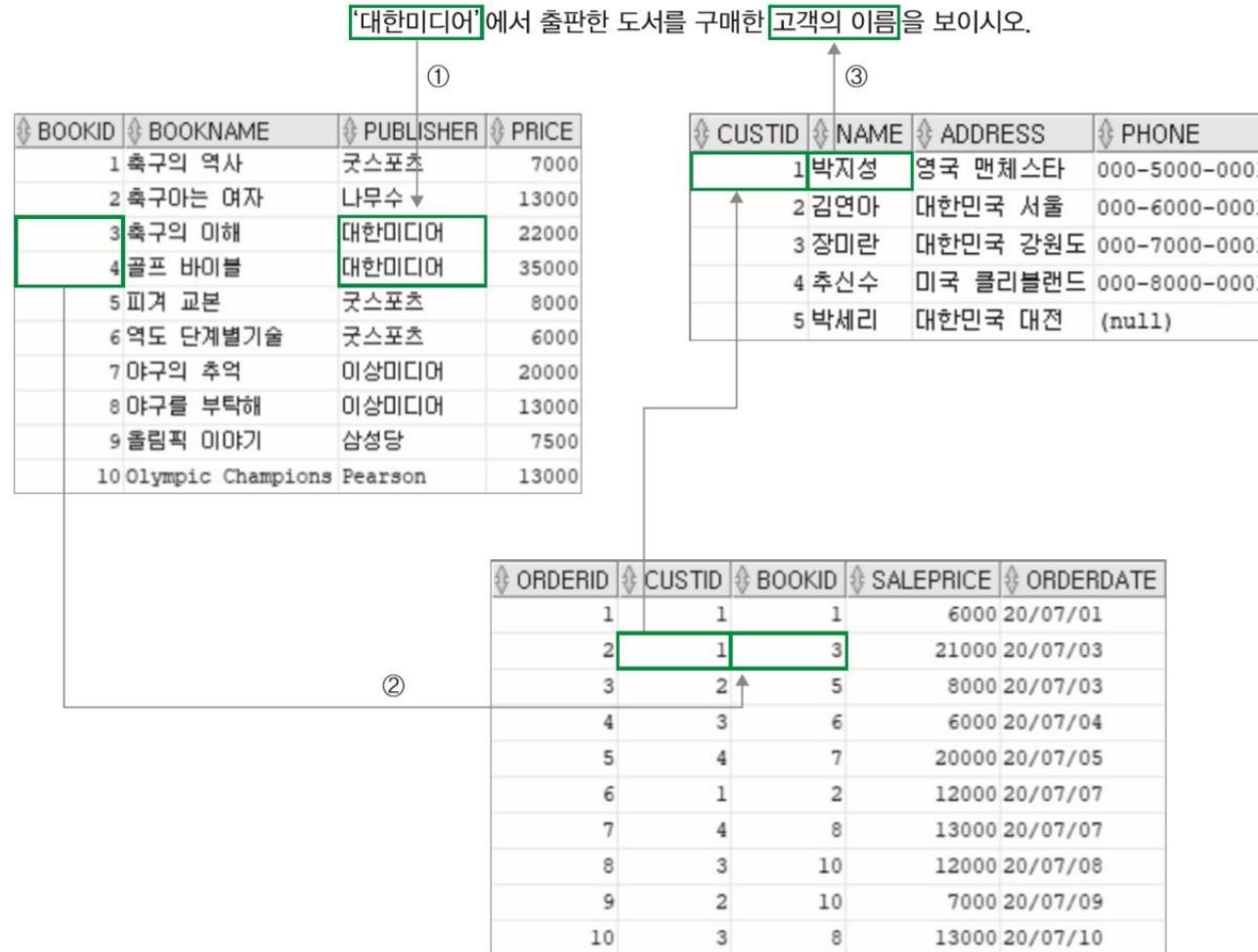


그림 3-17 3단계 부속질의의 실행 순서와 데이터 예

# 부속질의

## ■ 상관 부속질의(correlated subquery)

- 상위 부속질의의 투플을 이용하여 하위 부속질의를 계산함.
- 상위 부속질의와 하위 부속질의가 독립적이지 않고 서로 관련을 맺고 있음.

질의 3-31 출판사별로 출판사의 평균 도서 가격보다 비싼 도서를 구하시오.

```
SELECT      b1.bookname
FROM        Book b1
WHERE       b1.price > (SELECT      avg(b2.price)
                        FROM        Book b2
                        WHERE       b2.publisher=b1.publisher);
```

BOOKNAME
골프 바이블
피겨 교본
야구의 추억

# 부속질의



그림 3-18 상관 부속질의의 데이터 예

# 집합연산

## ■ 합집합 UNION, 차집합 MINUS, 교집합 INTERSECT

{도서를 주문하지 않은 고객} = {모든 고객} - {도서를 주문한 고객}

질의 3-32 도서를 주문하지 않은 고객의 이름을 보이시오.

```
SELECT      name
FROM        Customer
MINUS
SELECT      name
FROM        Customer
WHERE       custid IN (SELECT custid FROM Orders);
```

NAME
박세리

- 주의할점: Oracle은 차집합을 **MINUS**로 하지만 SQL 표준에서는 **EXCEPT** 를 사용

# EXISTS

## ■ EXISTS

- 원래 단어에서 의미하는 것과 같이 조건에 맞는 튜플이 존재하면 결과에 포함시킴
- 즉 부속질의문의 어떤 행이 조건에 만족하면 참임
  - 반면 NOT EXISTS는 부속질의문의 모든 행이 조건에 만족하지 않을 때만 참임.

질의 3-33 주문이 있는 고객의 이름과 주소를 보이시오.

```
SELECT      name, address
FROM        Customer cs
WHERE       EXISTS (SELECT *
                    FROM      Orders od
                    WHERE      cs.custid=od.custid);
```

NAME	ADDRESS
박지성	영국 맨체스터
김연아	대한민국 서울
장미란	대한민국 강원도
주신수	미국 클리블랜드

# EXISTS



그림 3-21 EXIST 상관 부속질의문의 데이터 예

# 연습문제

## 1. 마당서점의 고객이 요구하는 다음 질문에 대해 SQL 문을 작성하시오.

- (5) 박지성이 구매한 도서의 출판사 수
- (6) 박지성이 구매한 도서의 이름, 가격, 정가와 판매가격의 차이
- (7) 박지성이 구매하지 않은 도서의 이름

## 2. 마당서점의 운영자와 경영자가 요구하는 다음 질문에 대해 SQL 문을 작성하시오.

- (8) 주문하지 않은 고객의 이름(부속질의 사용)
- (9) 주문 금액의 총액과 주문의 평균 금액
- (10) 고객의 이름과 고객별 구매액
- (11) 고객의 이름과 고객이 구매한 도서 목록
- (12) 도서의 가격(Book 테이블)과 판매가격(Orders 테이블)의 차이가 가장 많은 주문
- (13) 도서의 판매액 평균보다 자신의 구매액 평균이 더 높은 고객의 이름

## 05. 데이터 정의어

---

- CREATE TABLE 문
- ALTER TABLE 문
- DROP TABLE 문

## 4.1 CREATE TABLE 문

- 테이블을 구성하고, 속성과 속성에 관한 제약을 정의하며, 기본키 및 외래키를 정의하는 명령
- PRIMARY KEY는 기본키를 정할 때 사용하고 FOREIGN KEY는 외래키를 지정할 때 사용하며, ON UPDATE와 ON DELETE는 외래키 속성의 수정과 투플 삭제 시 동작을 나타냄.
- CREATE TABLE 문의 기본 문법

```
CREATE TABLE 테이블이름
( {속성이름 데이터타입
    [NULL | NOT NULL | UNIQUE | DEFAULT 기본값 | CHECK 체크조건]
}
    [PRIMARY KEY 속성이름(들)]
    [FOREIGN KEY 속성이름 REFERENCES 테이블이름(속성이름)]
        [ON DELETE {CASCADE | SET NULL}]
)
```

## 4.1 CREATE TABLE 문

- 테이블을 구성하고, 속성과 속성에 관한 제약을 정의하며, 기본키 및 외래키를 정의하는 명령
- PRIMARY KEY는 기본키를 정할 때 사용하고 FOREIGN KEY는 외래키를 지정할 때 사용하며, ON UPDATE와 ON DELETE는 외래키 속성의 수정과 투플 삭제 시 동작을 나타냄.
- CREATE TABLE 문의 기본 문법

```
CREATE TABLE 테이블이름
( {속성이름 데이터타입
    [NULL | NOT NULL | UNIQUE | DEFAULT 기본값 | CHECK 체크조건]
}
    [PRIMARY KEY 속성이름(들)]
    [FOREIGN KEY 속성이름 REFERENCES 테이블이름(속성이름)]
        [ON DELETE {CASCADE | SET NULL}]
)
```

## 4.1 CREATE TABLE 문

질의 3-34 다음과 같은 속성을 가진 NewBook 테이블을 생성하시오,  
정수형은 NUMBER를, 문자형은 가변형 문자타입인 VARCHAR2를 사용  
bookid(도서번호) – NUMBER  
bookname(도서이름) – VARCHAR2(20)  
publisher(출판사) – VARCHAR2(20)  
price(가격) – NUMBER

```
CREATE TABLE      NewBook (
    bookid        NUMBER,
    bookname      VARCHAR2(20),
    publisher     VARCHAR2(20),
    price         NUMBER);
```

- 기본키를 지정하고

```
CREATE TABLE      NewBook (
    bookid        NUMBER,
    bookname      VARCHAR2(20),
    publisher     VARCHAR2(20),
    price         NUMBER,
    PRIMARY KEY    (bookid));
```

## 4.1 CREATE TABLE 문

- bookid 속성이 없어서 두 개의 속성 bookname, publisher가 기본키가 된다면 괄호를 사용하여 복합키를 지정

```
CREATE TABLE      NewBook (
    bookname        VARCHAR2(20),
    publisher       VARCHAR2(20),
    price           NUMBER,
    PRIMARY KEY     (bookname, publisher));
```

bookname은 NULL 값을 가질 수 없고, publisher는 같은 값이 있으면 안 된다. price에 값이 입력되지 않을 경우 기본값 10000을 저장한다. 또 가격은 최소 1,000원 이상으로 한다.

- NewBook 테이블의 CREATE 문에 좀 더 복잡한 제약사항을 추가

```
CREATE TABLE      NewBook (
    bookname        VARCHAR2(20) NOT NULL,
    publisher       VARCHAR2(20) UNIQUE,
    price           NUMBER DEFAULT 10000 CHECK(price > 1000),
    PRIMARY KEY     (bookname, publisher));
```

## 4.1 CREATE TABLE 문

질의 3-35 다음과 같은 속성을 가진 NewCustomer 테이블을 생성하시오

custid(고객번호)	- NUMBER, 기본키
name(이름)	- VARCHAR2(40)
address(주소)	- VARCHAR2(40)
phone(전화번호)	- VARCHAR2(30)

```
CREATE TABLE      NewCustomer (
  custid          NUMBER           PRIMARY KEY,
  name            VARCHAR2(40),
  address         VARCHAR2(40),
  phone           VARCHAR2(30));
```

## 4.1 CREATE TABLE 문

질의 3-36 다음과 같은 속성을 가진 NewOrders 테이블을 생성하시오.

orderid(주문번호) - NUMBER, 기본키

custid(고객번호) - NUMBER, NOT NULL 제약조건, 외래키(NewCustomer.custid, 연쇄삭제)

bookid(도서번호) - NUMBER, NOT NULL 제약조건

saleprice(판매가격) - NUMBER

orderdate(판매일자) - DATE

```
CREATE TABLE      NewOrders (
    orderid        NUMBER,
    custid         NUMBER          NOT NULL,
    bookid         NUMBER          NOT NULL,
    saleprice      NUMBER,
    orderdate      DATE,
    PRIMARY KEY(orderid),
    FOREIGN KEY(custid) REFERENCES NewCustomer(custid) ON DELETE CASCADE);
```

## 4.1 CREATE TABLE 문

- 외래키 제약조건을 명시할 때는 반드시 참조되는 테이블(부모 릴레이션)이 존재해야 함
- 참조되는 테이블의 기본키여야 함
- 외래키 지정 시 ON DELETE 또는 ON UPDATE 옵션은 참조되는 테이블의 튜플이 삭제되거나 수정될 때 취할 수 있는 동작을 지정
- NO ACTION은 어떠한 동작도 취하지 않음.

표 3-8 속성의 데이터 타입 종류

데이터 타입	설명	ANSI SQL 표준 타입
NUMBER(p, s)	실수형 p자리 정수부분, s자리 소수부분, p와 s를 생략하여 NUMBER라고 쓰면 NUMBER(8, 2)로 저장된다.	DECIMAL(p, s) NUMERIC[(p,s)] INTEGER, INT SMALLINT
CHAR(n)	문자형 고정길이, 문자를 저장하고 남은 공간은 공백으로 채운다.	CHARACTER(n) CHAR(n)
VARCHAR2(n)	문자형 가변길이, 4,000바이트까지 저장된다.	CHARACTER VARYING(n) CHAR VARYING(n)
DATE	날짜형, 연도, 월, 일, 시간을 저장한다.	

## 4.2 ALTER TABLE 문

- ALTER 문은 생성된 테이블의 속성과 속성에 관한 제약을 변경하며, 기본키 및 외래키를 변경함
- ADD, DROP은 속성을 추가하거나 제거할 때 사용
- MODIFY는 속성의 기본값을 설정하거나 삭제할 때 사용
- ADD <제약이름>, DROP <제약이름>은 제약사항을 추가하거나 삭제할 때 사용
- ALTER 문의 기본 문법

```
ALTER TABLE 테이블이름  
[ADD 속성이름 데이터타입]  
[DROP COLUMN 속성이름]  
[ALTER COLUMN 속성이름 데이터타입]  
[ALTER COLUMN 속성이름 [NULL | NOT NULL]]  
[ADD PRIMARY KEY(속성이름)]  
[[ADD | DROP] 제약이름]
```

실습을 위하여 NewBook 테이블을 지우고 질의3-34 NewBook 테이블을 새로 생성한다.

```
CREATE TABLE      NewBook (  
bookid          NUMBER,  
bookname        VARCHAR2(20),  
publisher       VARCHAR2(20),  
price           NUMBER);
```

## 4.2 ALTER TABLE 문

질의 3-37 NewBook 테이블에 VARCHAR2(13)의 자료형을 가진 isbn 속성을 추가하시오

```
ALTER TABLE NewBook ADD isbn VARCHAR2(13);
```

질의 3-38 NewBook 테이블의 isbn 속성의 데이터 타입을 NUMBER형으로 변경하시오

```
ALTER TABLE NewBook MODIFY isbn NUMBER;
```

질의 3-39 NewBook 테이블의 isbn 속성을 삭제하시오

```
ALTER TABLE NewBook DROP COLUMN isbn;
```

질의 3-40 NewBook 테이블의 bookid 속성에 NOT NULL 제약조건을 적용하시오

```
ALTER TABLE NewBook MODIFY bookid NUMBER NOT NULL;
```

질의 3-41 NewBook 테이블의 bookid 속성을 기본키로 변경하시오

```
ALTER TABLE NewBook ADD PRIMARY KEY(bookid);
```

## 4.3 DROP TABLE 문

### ■ DROP 문

- DROP 문은 테이블을 삭제하는 명령
- DROP 문은 테이블의 구조와 데이터를 모두 삭제하므로 사용에 주의해야 함
  - (데이터만 삭제하려면 DELETE 문을 사용)

### ■ DROP문의 기본 문법

```
DROP TABLE 테이블이름
```

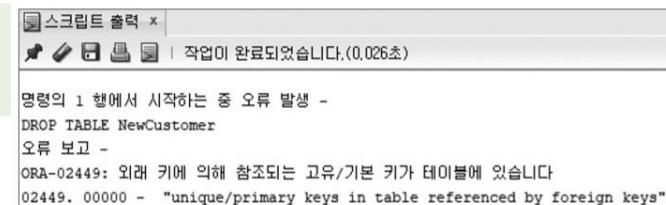
질의 3-42 NewBook 테이블을 삭제하시오

```
DROP TABLE NewBook;
```

질의 3-43 NewCustomer 테이블을 삭제하시오. 만약 삭제가 거절된다면 원인을  
파악하고 관련된 테이블을 같이 삭제하시오

(NewOrders 테이블이 NewCustomer를 참조하고 있음)

```
DROP TABLE NewCustomer;
```



## 05. 데이터 조작어 – 삽입, 수정, 삭제

---

- INSERT 문
- UPDATE 문
- DELETE 문

# 5.1 INSERT 문

- INSERT 문은 테이블에 새로운 투플을 삽입하는 명령임.

- INSERT 문의 기본 문법

```
INSERT INTO 테이블이름[(속성리스트)]  
VALUES (값리스트);
```

질의 3-44 Book 테이블에 새로운 도서 '스포츠 의학'을 삽입하시오. 스포츠 의학은  
한솔의학서적에서 출간했으며 가격은 90,000원이다.

```
INSERT INTO Book(bookid, bookname, publisher, price)  
VALUES (11, '스포츠 의학', '한솔의학서적', 90000);
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
11	스포츠 의학	한솔의학서적	90000
1	축구의 역사	굿스포츠	7000
2	축구마는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 주역	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

## 5.1 INSERT 문

질의 3-45 Book 테이블에 새로운 도서 '스포츠 의학'을 삽입하시오. 스포츠 의학은 한솔의학서적에서 출간했으며 가격은 미정이다.

```
INSERT INTO Book(bookid, bookname, publisher)
VALUES (14, '스포츠 의학', '한솔의학서적');
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
11	스포츠 의학	한솔의학서적	90000
14	스포츠 의학	한솔의학서적	(null)
1	축구의 역사	굿스포츠	7000
2	축구마는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

## 5.1 INSERT 문

- 대량 삽입(bulk insert)이란 한꺼번에 여러 개의 투플을 삽입하는 방법임.

질의 3-46 수입도서 목록(Imported\_book)을 Book 테이블에 모두 삽입하시오.

```
INSERT INTO Book(bookid, bookname, price, publisher)
SELECT bookid, bookname, price, publisher
FROM Imported_book;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
11	스포츠 의학	한솔의학서적	90000
14	스포츠 의학	한솔의학서적	(null)
21	Zen Golf	Pearson	12000
22	Soccer Skills	Human Kinetics	15000
1	축구의 역사	굿스포츠	7000
2	축구마는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	미상미디어	20000
8	야구를 부탁해	미상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

## 5.2 UPDATE 문

- UPDATE 문은 특정 속성 값을 수정하는 명령

- UPDATE 문의 기본 문법

```
UPDATE    테이블이름  
SET        속성이름 1=값 1[, 속성이름 2=값 2, ...]  
[WHERE     <검색조건>];
```

## 5.2 UPDATE 문

질의 3-47 Customer 테이블에서 고객번호가 5인 고객의 주소를 '대한민국 부산'으로 변경하시오.

```
UPDATE Customer  
SET address='대한민국 부산'  
WHERE custid=5;
```

TIP 결과를 확인하기 위해서는 'SELECT \* FROM Customer;' 명령을 실행해야 한다.

CUSTID	NAME	ADDRESS	PHONE
1	박지성	영국 맨체스터	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-8000-0001
5	박세리	대한민국 부산	(null)

질의 3-48 Customer 테이블에서 박세리 고객의 주소를 김연아 고객의 주소로 변경하시오.

```
UPDATE Customer  
SET address=(SELECT address  
FROM Customer  
WHERE name='김연아')  
WHERE name='박세리';
```

CUSTID	NAME	ADDRESS	PHONE
1	박지성	영국 맨체스터	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-8000-0001
5	박세리	대한민국 서울	(null)

## 5.3 DELETE 문

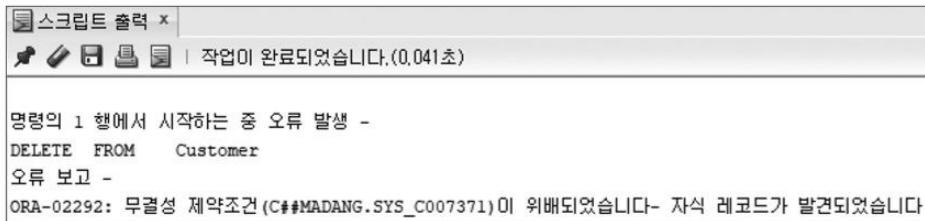
- **DELETE 문은 테이블에 있는 기존 투플을 삭제하는 명령**
- **DELETE 문의 기본 문법**

```
DELETE FROM 테이블이름  
[WHERE 검색조건];
```

## 5.3 DELETE 문

질의 3-49 Customer 테이블에서 고객번호가 5인 고객을 삭제하시오.

```
DELETE FROM Customer;
```



스크립트 출력 x | 작업이 완료되었습니다.(0.041초)

명령의 1 행에서 시작하는 중 오류 발생 -  
DELETE FROM Customer  
오류 보고 -  
ORA-02292: 무결성 제약조건 (C#MADANG.SYS\_C007371)이 위배되었습니다- 자식 레코드가 발견되었습니다

질의 3-50 모든 고객을 삭제하시오.

```
DELETE FROM Customer  
WHERE custid=5;
```

```
SELECT * FROM Customer;
```

CUSTID	NAME	ADDRESS	PHONE
1	박지성	영국 맨체스터	000-5000-0001
2	김연마	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-8000-0001

# 요약

- 
- 1. SQL
  - 2. 데이터 정의어(DDL)
  - 3. 데이터 조작어(DML)
  - 4. WHERE 조건
  - 5. 집계 함수
  - 6. GROUP BY
  - 7. HAVING
  - 8. 조인
  - 9. 동등조인(내부조인)
  - 10. 부속질의
  - 11. 상관 부속질의
  - 12. 투플 변수
  - 13. 집합연산
  - 14. 집합연산
  - 15. EXISTS
  - 16. CREATE
  - 17. ALTER
  - 18. DROP
  - 19. INSERT
  - 20. UPDATE
  - 21. DELETE

# 목차

1. 내장 함수
2. 부속질의
3. 뷰
4. 인덱스

# 학습목표

- 내장 함수의 의미를 알아보고 자주 사용되는 내장 함수 몇 가지를 직접 실습해 본다
- 부속질의의 의미와 종류를 알아보고 직접 실습해 본다.
- 뷰의 의미를 알아보고 뷰를 직접 생성, 수정, 삭제해 본다.
- 데이터베이스의 저장 구조와 인덱스의 관계를 알아보고 인덱스를 직접 생성, 수정, 삭제해 본다.

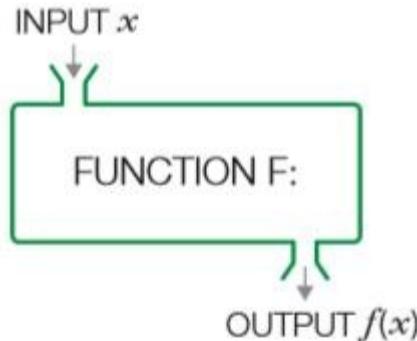
# 01. 내장함수

---

- SQL 내장 함수
- NULL 값 처리
- ROWNUM

# 01. 내장함수

- SQL에서는 함수의 개념을 사용하는데 수학의 함수와 마찬가지로 특정 값이나 열의 값을 입력받아 그 값을 계산하여 결과 값을 돌려줌.
  - 함수의 원리



- SQL의 함수는 DBMS가 제공하는 내장 함수(built-in function)와 사용자가 필요에 따라 직접 만드는 사용자 정의 함수(user-defined function)로 나뉨.

# 1.1 SQL 내장함수

- SQL 내장 함수는 상수나 속성 이름을 입력 값으로 받아 단일 값을 결과로 반환함.
- 모든 내장 함수는 최초에 선언될 때 유효한 입력 값을 받아야 함.

표 4-1 오라클에서 제공하는 주요 내장 함수

구분	함수	
단일행 함수	숫자 함수	ABS, CEIL, COS, EXP, FLOOR, LN, LOG, MOD, POWER, ROUND(number), SIGN, TRUNC(number)
	문자 함수 (문자 반환)	CHR, CONCAT, LOWER, LPAD, LTRIM, STR, REPLACE, RPAD, RTRIM, SUBSTR, TRIM, UPPER
	문자 함수 (숫자 반환)	ASCII, INSTR, LENGTH
	날짜 · 시간 함수	ADD_MONTHS, LAST_DAY, NEXT_DAY, ROUND(date), SYSDATE, TO_CHAR(datetime)
	변환 함수	ASCIIISTR, CONVERT, TO_BINARY_DOUBLE, TO_BINARY_FLOAT, TO_CHAR(character), TO_CHAR(datetime), TO_CHAR(number), TO_DATE, TO_NUMBER
	인코딩과 디코딩	DECODE, DUMP, VSIZE
집계 함수	NULL 관련 함수	COALESCE, NULLIF, NVL
		AVG, COUNT, CUME_DIST, FIRST, LAST, MAX, MEDIAN, MIN, PERCENT_RANK, PERCENTILE_CONT, SUM
분석 함수		AVG, CORR, COUNT, CUME_DIST, DENSE_RANK, FIRST, FIRST_VALUE, LAST_VALUE, LEAD, MAX, MIN, RANK, SUM

# 숫자 함수

표 4-2 숫자 함수의 종류

함수	설명	예
ABS(숫자)	숫자의 절댓값 계산	$\text{ABS}(-4.5) = 4.5$
CEIL(숫자)	숫자보다 크거나 같은 최소의 정수	$\text{CEIL}(4.1) = 5$
FLOOR(숫자)	숫자보다 작거나 같은 최소의 정수	$\text{FLOOR}(4.1) = 4$
ROUND(숫자, m)	m 자리를 기준으로 숫자 반올림	$\text{ROUND}(5.36, 1) = 5.40$
LOG(n, 숫자)	숫자의 자연로그 값 반환	$\text{LOG}(10) = 2.30259$
POWER(숫자, n)	숫자의 n제곱 값 계산	$\text{POWER}(2, 3) = 8$
SQRT(숫자)	숫자의 제곱근 값 계산(숫자는 양수)	$\text{SQRT}(9.0) = 3.0$
SIGN(숫자)	숫자가 음수이면 -1, 0이면 0, 양수이면 1	$\text{SIGN}(3.45) = 1$

# 숫자 함수

## ■ ABS 함수 : 절댓값을 구하는 함수

질의 4-1 -78과 +78의 절댓값을 구하시오.

```
SELECT    ABS(-78), ABS(+78)
FROM      Dual;
```

ABS(-78)	ABS(+78)
78	78

## ■ ROUND 함수 : 반올림한 값을 구하는 함수

질의 4-2 4.875를 소수 첫째 자리까지 반올림한 값을 구하시요.

```
SELECT    ROUND(4.875, 1)
FROM      Dual;
```

ROUND(4.875,1)
4.9

## ■ 숫자 함수의 연산

질의 4-3 고객별 평균 주문 금액을 배 원 단위로 반올림한 값을 구하시요.

```
SELECT    custid "고객번호", ROUND(SUM(saleprice)/COUNT(*), -2) "평균금액"
FROM      Orders
GROUP BY  custid;
```

고객번호	평균금액
1	13000
2	7500
4	16500
3	10300

# 문자 함수

표 4-3 문자 함수의 종류: 문자값 반환 함수(s는 문자열, c는 문자, n과 k는 정수)

함수	설명과 예
CHR(k)	정수 아스키코드를 문자로 반환 예 CHR(68) = 'D'
CONCAT(s1, s2)	두 문자열을 연결 예 CONCAT('마당', '서점') = '마당서점'
INITCAP(s)	문자열의 첫 번째 알파벳을 대문자로 변환 예 INITCAP('the soap') = 'The Soap'
LOWER(s)	대상 문자열을 모두 소문자로 변환 예 LOWER('MR. SCOTT') = 'mr. scott'
LPAD(s, n, c)	대상 문자열의 왼쪽부터 지정한 자릿수까지 지정한 문자로 채움 예 LPAD('Page 1', 10, '*') = '****Page 1'
LTRIM(s1, s2)	대상 문자열의 왼쪽부터 지정한 문자들을 제거 예 LTRIM('<==>BROWNING<==>', '<>=') = 'BROWNING<==>'
REPLACE(s1, s2, s3)	대상 문자열의 지정한 문자를 원하는 문자로 변경 예 REPLACE('JACK and JUE', 'J', 'BL') = 'BLACK and BLUE'
RPAD(s, n, c)	대상 문자열의 오른쪽부터 지정한 자릿수까지 지정한 문자로 채움 예 RPAD('AbC', 5, '*') = 'AbC**'
RTRIM(s1, s2)	대상 문자열의 오른쪽부터 지정한 문자들을 제거 예 RTRIM('<==>BROWNING<==>', '<>=') = '<==>BROWNING'
SUBSTR(s, n, k)	대상 문자열의 지정된 자리에서부터 지정된 길이만큼 잘라서 반환 예 SUBSTR('ABCDEFG', 3, 4) = 'CDEF'
TRIM(c FROM s)	대상 문자열의 양쪽에서 지정된 문자를 삭제(문자열만 넣으면 기본값으로 공백 제거) 예 TRIM(' ' FROM '<==>BROWNING<==>') = '>BROWNING<'
UPPER(s)	대상 문자열을 모두 대문자로 변환 예 UPPER('mr. scott') = 'MR. SCOTT'
ASCII(c)	대상 알파벳 문자의 아스키코드 값을 반환 예 ASCII('D') = 68
INSTR(s1, s2, n, k)	문자열 중 n번째 문자부터 시작하여 찾고자 하는 문자열 s2가 k번째 나타나는 문자열 위치 반환, 예제에서 3번째부터 OR가 2번째 나타나는 자릿수 예 INSTR('CORPORATE FLOOR', 'OR', 3, 2) = 14
LENGTH(s)	대상 문자열의 글자 수를 반환 예 LENGTH('CANDIDE') = 7

# 문자 함수

## ■ REPLACE : 문자열을 치환하는 함수

질의 4-4 도서제목에 야구가 포함된 도서를 농구로 변경한 후 도서 목록을 보이시오.

```
SELECT      bookid, REPLACE(bookname, '야구', '농구') bookname, publisher, price  
FROM        Book;
```

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	농구의 추억	미상미디어	20000
8	농구를 부탁해	미상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions Pearson	Pearson	13000

# 문자 함수

## ■ LENGTH : 글자의 수를 세어주는 함수 (단위가 바이트(byte)가 아닌 문자 단위)

질의 4-5 굿스포츠에서 출판한 도서의 제목과 제목의 글자 수를 확인하시오.  
(한글은 2바이트 혹은 UNICODE 경우는 3바이트를 차지함)

```
SELECT bookname "제목", LENGTH(bookname) "글자수",
       LENGTHB(bookname) "바이트수"
  FROM Book
 WHERE publisher='굿스포츠';
```

제목	글자수	바이트수
죽구의 역사	6	16
파견 교본	5	13
역도 단계별기술	8	22

## ■ SUBSTR : 지정한 길이만큼의 문자열을 반환하는 함수

질의 4-6 마당서점의 고객 중에서 같은 성(姓)을 가진 사람이 몇 명이나 되는지 성별  
인원수를 구하시오.

```
SELECT SUBSTR(name, 1, 1) "성", COUNT(*) "인원"
  FROM Customer
 GROUP BY SUBSTR(name, 1, 1);
```

성
1
1
1
2

# 날짜 · 시간 함수

표 4-5 날짜 · 시간 함수의 종류

함수	설명과 예
TO_DATE (char, datetime)	문자형(CHAR) 데이터를 DATE형으로 반환 예 TO_DATE('2020-09-14', 'yyyy-mm-dd')=2020-09-14
TO_CHAR (date, datetime)	DATE형 데이터를 문자열(VARCHAR2)로 반환 예 TO_CHAR(TO_DATE('2020-09-14', 'yyyy-mm-dd'), 'yyyymmdd')='20200914'
ADD_MONTHS (date, 숫자)	날짜에 지정한 달을 더해 DATE형으로 반환(1: 다음 달, -1: 이전 달) 예 ADD_MONTHS(TO_DATE('2020-09-14', 'yyyy-mm-dd'), 12)=2021-09-14
LAST_DAY (date)	날짜에 달의 마지막 날을 DATE형으로 반환 예 LAST_DAY(TO_DATE('2020-09-14', 'yyyy-mm-dd'))=2020-09-30
SYSDATE	DBMS 시스템상의 당일 날짜를 DATE형으로 반환하는 인자가 없는 함수 예 SYSDATE=20/09/20

# 날짜 · 시간 함수

표 4-6 datetime의 주요 인자

인자	설명
d	요일 순서(1~7, 월=1)
day	요일(월요일~일요일)
dy	요일의 약자(월~일)
dd	1달 중 날짜(1~31)
ddd	1년 중 날짜(1~366)
hh, hh12	12시간(1~12)
hh24	24시간(0~23)
mi	분(0~59)
mm	월 순서(01~12, January=01)
mon	월 이름 약어(Jan~Dec)
month	월 이름(January~December)
ss	초(0~59)
yyyy	4자리 연도
yy, yy, y	4자리 연도의 마지막 3, 2, 1자리

## 날짜 · 시간 함수

질의 4-7 마당서점은 주문일로부터 10일 후 매출을 확정한다. 각 주문의 확정일자를 구하시오

```
SELECT      orderid "주문번호", orderdate "주문일", orderdate+10 "확정"  
FROM        Orders;
```

주문번호	주문일	확정
1	20/07/01	20/07/11
2	20/07/03	20/07/13
3	20/07/03	20/07/13
4	20/07/04	20/07/14
5	20/07/05	20/07/15
6	20/07/07	20/07/17
7	20/07/07	20/07/17
8	20/07/08	20/07/18
9	20/07/09	20/07/19
10	20/07/10	20/07/20

# 날짜 · 시간 함수

- **TO\_DATE** : 문자형으로 저장된 날짜를 날짜형으로 변환하는 함수
- **TO\_CHAR** : 날짜형을 문자형으로 변환하는 함수

질의 4-8 마당서점이 2014년 7월 7일에 주문받은 도서의 주문번호, 주문일, 고객번호, 도서번호를 모두 보이시오. 단 주문일은 'yyyy-mm-dd 요일' 형태로 표시

```
SELECT      orderid "주문번호", TO_CHAR(orderdate, 'yyyy-mm-dd dy') "주문일",
            custid "고객번호", bookid "도서번호"
FROM        Orders
WHERE       orderdate=TO_DATE('20200707', 'yyyymmdd');
```

주문번호	주문일	고객번호	도서번호
6	2020-07-07 화	1	2
7	2020-07-07 화	4	8

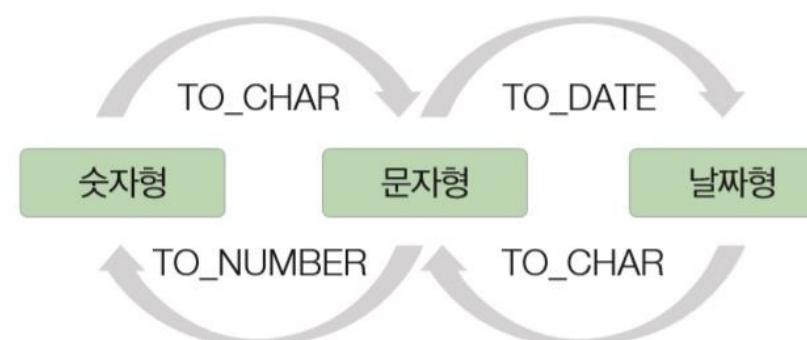


그림 4-1 형 변환 함수 TO\_CHAR, TO\_DATE, TO\_NUMBER

# 날짜 · 시간 함수

- **SYSDATETIME** : 오라클의 현재 날짜와 시간을 반환하는 함수
- **SYSTIMESTAMP** : 현재 날짜, 시간과 함께 초 이하의 시간과 서버의 TIMEZONE까지 출력함

질의 4-9 DBMS 서버에 설정된 현재 시간과 오늘 날짜를 확인하시오.

```
SELECT    SYSDATE,  
          TO_CHAR(SYSDATE, 'yyyy/mm/dd dy hh24:mi:ss') "SYSDATE_1"  
FROM      Dual;
```

SYSDATE	SYSDATE_1
20/05/28	2020/05/28 목 00:18:48

# 연습문제

## 1. 다음 내장 함수의 결과를 적으시오.

- ABS(-15)
- CEIL(15.7)
- COS(3.14159)
- FLOOR(15.7)
- LOG(10,100)
- MOD(11,4)
- POWER(3,2)
- ROUND(15.7)
- SIGN(-15)
- TRUNC(15.7)
- CHR(67)
- CONCAT('HAPPY ', 'Birthday')
- LOWER('Birthday')
- LPAD('Page 1', 15, '\*.')
- LTRIM('Page 1', 'ae')
- REPLACE('JACK', 'J', 'BL')
- RPAD('Page 1', 15, '\*.')
- RTRIM('Page 1', 'ae')
- SUBSTR('ABCDEFG', 3, 4)
- TRIM.LEADING 0 FROM '00AA00'
- UPPER('Birthday')
- ASCII('A')
- INSTR('CORPORATE FLOOR','OR', 3, 2)
- LENGTH('Birthday')
- ADD\_MONTHS('14/05/21', 1)
- LAST\_DAY(SYSDATE)
- NEXT\_DAY(SYSDATE, '화')
- ROUND(SYSDATE)
- SYSDATE
- TO\_CHAR(SYSDATE)
- TO\_CHAR(123)
- TO\_DATE('12 05 2014', 'DD MM YYYY')
- TO\_NUMBER('12.3')
- DECODE(1,1,'aa','bb')
- NULLIF(123, 345)
- NVL(NULL, 123)

## 1.2 NULL 값 처리

### ■ NULL 값이란?

- 아직 지정되지 않은 값
- NULL 값은 '0', '' (빈 문자), ' ' (공백) 등과 다른 특별한 값
- NULL 값은 비교 연산자로 비교가 불가능함.
- NULL 값의 연산을 수행하면 결과 역시 NULL 값으로 반환됨.

### ■ 집계 함수를 사용할 때 주의할 점

- 'NULL+숫자' 연산의 결과는 NULL
- 집계 함수 계산 시 NULL이 포함된 행은 집계에서 빠짐
- 해당되는 행이 하나도 없을 경우 SUM, AVG 함수의 결과는 NULL이 되며, COUNT 함수의 결과는 0.

# 1.2 NULL 값 처리

## ■ NULL 값에 대한 연산과 집계 함수

- (\* Mybook 테이블 생성은 웹페이지 스크립트 참조)

```
SELECT      price+100
FROM        Mybook
WHERE       bookid=3;
```

Mybook	
bookid	price
1	10000
2	20000
3	NULL
	PRICE+100 (null)

```
SELECT      SUM(price), AVG(price), COUNT(*), COUNT(price)
FROM        Mybook;
```

SUM(PRICE)	AVG(PRICE)	COUNT(*)	COUNT(PRICE)
30000	15000	3	2

```
SELECT      SUM(price), AVG(price), COUNT(*)
FROM        Mybook
WHERE       bookid >=4;
```

SUM(PRICE)	AVG(PRICE)	COUNT(*)
(null)	(null)	0

## 1.2 NULL 값 처리

### ■ NULL 값을 확인하는 방법 – IS NULL, IS NOT NULL

- NULL 값을 찾을 때는 '=' 연산자가 아닌 'IS NULL'을 사용,
- NULL이 아닌 값을 찾을 때는 '< >' 연산자가 아닌 'IS NOT NULL'을 사용함

Mybook

bookid	price
1	10000
2	20000
3	NULL

```
SELECT *  
FROM Mybook  
WHERE price IS NULL;
```

BOOKID	PRICE
3	{null}

(a) 옳은 예

```
SELECT *  
FROM Mybook  
WHERE price=' ';
```

BOOKID	PRICE
3	

(b) 틀린 예

그림 4-2 NULL 값 찾기의 옳은 예와 틀린 예

## 1.2 NULL 값 처리

### ■ NVL : NULL 값을 다른 값으로 대치하여 연산하거나 다른 값으로 출력

- NVL(속성, 값) /\* 속성 값이 NULL이면 '값'으로 대치한다 \*/

질의 4-10 이름, 전화번호가 포함된 고객목록을 보이시오.

단, 전화번호가 없는 고객은 '연락처없음'으로 표시한다.

```
SELECT      name "이름", NVL(phone, '연락처없음') "전화번호"  
FROM        Customer;
```

이름	전화번호
박지성	000-5000-0001
김연아	000-6000-0001
장미란	000-7000-0001
추신수	000-8000-0001
박세리	연락처없음

## 1.3 ROWNUM

- 내장 함수는 아니지만 자주 사용되는 문법임.
- 오라클에서 내부적으로 생성되는 가상 컬럼으로 SQL 조회 결과의 순번을 나타냄.
- 자료를 일부분만 확인하여 처리할 때 유용함.

질의 4-11 고객 목록에서 고객번호, 이름, 전화번호를 앞의 두 명만 보이시오.

```
SELECT      ROWNUM "순번", custid, name, phone  
FROM        Customer  
WHERE       ROWNUM <=2;
```

순번	CUSTID	NAME	PHONE
1	1	박지성	000-5000-0001
2	2	김연마	000-6000-0001

# 연습문제

## 2. Mybook 테이블을 생성하고 NULL에 관한 다음 SQL 문에 답하시오.

질의의 결과를 보면서 NULL에 대한 개념을 정리해보시오.

Mybook

bookid	price
1	10000
2	20000
3	NULL

SQL 문

- (1) SELECT \*  
FROM Mybook;
- (2) SELECT bookid, NVL(price, 0)  
FROM Mybook;
- (3) SELECT \*  
FROM Mybook  
WHERE price IS NULL;
- (4) SELECT \*  
FROM Mybook;  
WHERE price="";
- (5) SELECT bookid, price+100  
FROM Mybook;
- (6) SELECT SUM(price), AVG(price), COUNT(\*)  
FROM Mybook  
WHERE bookid >= 4;
- (7) SELECT COUNT(\*), COUNT(price)  
FROM Mybook;
- (8) SELECT SUM(price), AVG(price)  
FROM Mybook;

결과

### 3. ROWNUM에 관한 다음 SQL 문에 답하시오. 데이터는 마당서점 데이터베이스를 이용

SQL 문	결과
(1) SELECT * FROM Book;	
(2) SELECT * FROM Book WHERE ROWNUM <= 5;	
(3) SELECT * FROM Book WHERE ROWNUM <= 5 ORDER BY price;	
(4) SELECT * FROM (SELECT * FROM Book ORDER BY price) b WHERE ROWNUM <= 5;	
(5) SELECT * FROM (SELECT * FROM Book WHERE ROWNUM<=5) b ORDER BY price;	
(6) SELECT * FROM (SELECT * FROM Book WHERE ROWNUM <= 5 ORDER BY price) b;	

## 02. 부속질의

---

- 중첩질의                    – WHERE 부속질의
- 스칼라 부속질의        – SELECT 부속질의
- 인라인 뷰                    – FROM 부속질의

## 02. 부속질의

### ■ 부속질의(subquery)란?

- 하나의 SQL 문 안에 다른 SQL 문이 중첩된 nested 질의를 말함.
- 다른 테이블에서 가져온 데이터로 현재 테이블에 있는 정보를 찾거나 가공할 때 사용함.
- 보통 데이터가 대량일 때 데이터를 모두 합쳐서 연산하는 조인보다 필요한 데이터만 찾아서 공급해주는 부속질의가 성능이 더 좋음.
- 주 질의(main query, 외부질의)와 부속질의(sub query, 내부질의)로 구성됨.

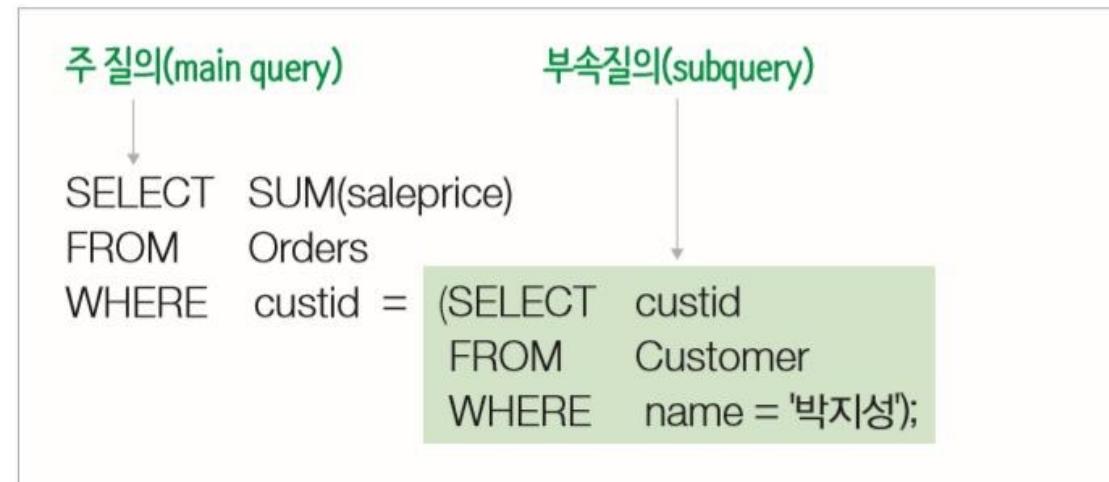


그림 4-3 부속질의

## 02. 부속질의

표 4-8 부속질의의 종류

명칭	위치	영문 및 동의어	설명
중첩질의	WHERE 절	nested subquery, predicate subquery	WHERE 절에 술어와 같이 사용되며 결과를 한정시키기 위해 사용된다. 상관 혹은 비상관 형태이다.
스칼라 부속질의	SELECT 절	scalar subquery	SELECT 절에서 사용되며 단일 값을 반환하기 때문에 스칼라 부속질의라고 한다.
인라인 뷰	FROM 절	inline view, table subquery	FROM 절에서 결과를 뷰(view) 형태로 반환하기 때문에 인라인 뷰라고 한다.

## 2.1 중첩질의 - WHERE 부속질의

- 중첩질의(nested subquery)는 WHERE 절에서 사용되는 부속질의.
- WHERE 절은 보통 데이터를 선택하는 조건 혹은 술어(predicate)와 같이 사용됨.  
그래서 중첩질의를 술어 부속질의(predicate subquery)라고도 함.

표 4-9 중첩질의 연산자의 종류

술어	연산자	반환 행	반환 열	상관
비교	=, >, <, >=, <=, <>	단일	단일	가능
집합	IN, NOT IN	다중	다중	가능
한정	ALL, SOME(ANY)	다중	단일	가능
존재	EXISTS, NOT EXISTS	다중	다중	필수

## 2.1 중첩질의 - WHERE 부속질의

### ■ 비교 연산자

- 부속질의가 반드시 단일 행, 단일 열을 반환해야 하며, 아닐 경우 질의를 처리할 수 없음.

질의 4-12 평균 주문금액 이하의 주문에 대해서 주문번호와 금액을 보이시오.

```
SELECT      orderid, saleprice
FROM        Orders
WHERE       saleprice <=(SELECT AVG(saleprice)
                           FROM   Orders);
```

ORDERID	SALEPRICE
1	6000
3	8000
4	6000
9	7000

질의 4-13 각 고객의 평균 주문금액보다 큰 금액의 주문 내역에 대해서 주문번호, 고객번호, 금액을 보이시오.

```
SELECT      orderid, custid, saleprice
FROM        Orders md
WHERE       saleprice > (SELECT      AVG(saleprice)
                           FROM        Orders so
                           WHERE       md.custid=so.custid);
```

ORDERID	CUSTID	SALEPRICE
2	1	21000
3	2	8000
5	4	20000
8	3	12000
10	3	13000

## 2.1 중첩질의 - WHERE 부속질의

### ■ IN, NOT IN

- IN 연산자는 주질의 속성 값이 부속질의에서 제공한 결과 집합에 있는지 확인(check)하는 역할을 함. IN 연산자는 부속질의의 결과 다중 행을 가질 수 있음. 주질의는 WHERE 절에 사용되는 속성 값을 부속질의의 결과 집합과 비교해 하나라도 있으면 참이 된다. NOT IN은 이와 반대로 값이 존재하지 않으면 참이 됨.

질의 4-14 대한민국에 거주하는 고객에게 판매한 도서의 총판매액을 구하시오.

```
SELECT      SUM(saleprice) "total"
FROM        Orders
WHERE       custid IN (SELECT    custid
                      FROM      Customer
                      WHERE     address LIKE '%대한민국%');
```

total
46000

## 2.1 중첩질의 - WHERE 부속질의

### ■ ALL, SOME(ANY)

- ALL은 모두, SOME(ANY)은 어떠한(최소한 하나라도)이라는 의미를 가짐.

### ■ 구문 구조

```
scalar_expression { 비교 연산자 ( =, <>, !=, >, >=, !=, <, <=, !< ) }  
{ ALL | SOME | ANY } (부속질의)
```

질의 4-15 3번 고객이 주문한 도서의 최고 금액보다 더 비싼 도서를 구입한 주문의 주문번호와 금액을 보이시오.

```
SELECT      orderid, saleprice  
FROM        Orders  
WHERE       saleprice > ALL (SELECT      saleprice  
                           FROM        Orders  
                           WHERE       custid='3');
```

ORDERID	SALEPRICE
5	20000
2	21000

## 2.1 중첩질의 - WHERE 부속질의

### ■ EXISTS, NOT EXISTS

- 데이터의 존재 유무를 확인하는 연산자
- 주질의에서 부속질의로 제공된 속성의 값을 가지고 부속질의에 조건을 만족하여 값이 존재하면 참이 되고, 주질의는 해당 행의 데이터를 출력함.
- NOT EXIST의 경우 이와 반대로 동작함.

### ■ 구문 구조

질의 4-16 EXISTS 연산자로 대한민국에 거주하는 고객에게 판매한 도서의 총 판매액을 구하시오.

```
SELECT      SUM(saleprice) "total"
FROM        Orders od
WHERE       EXISTS (SELECT *
                    FROM   Customer cs
                    WHERE  address LIKE '%대한민국%' AND cs.custid=od.custid);
```

total
46000

## 2.2 스칼라 부속질의 – SELECT 부속질의

### ■ 스칼라 부속질의(scalar subquery)란?

- SELECT 절에서 사용되는 부속질의로, 부속질의의 결과 값을 단일 행, 단일 열의 스칼라 값으로 반환함.
- 스칼라 부속질의는 원칙적으로 스칼라 값이 들어갈 수 있는 모든 곳에 사용 가능하며, 일반적으로 SELECT 문과 UPDATE SET 절에 사용됨.
- 주질의와 부속질의와의 관계는 상관/비상관 모두 가능함.

표 4-4 문자 함수의 종류: 숫자값 반환 함수

함수	설명
ASCII(c)	대상 알파벳 문자의 아스키코드 값을 반환 예 ASCII('D') = 68
INSTR(s1, s2, n, k)	문자열 중 n번째 문자부터 시작하여 찾고자 하는 문자열 s2가 k번째 나타나는 문자열 위치 반환, 예제에서 3번째부터 OR가 2번째 나타나는 자릿수 예 INSTR('CORPORATE FLOOR', 'OR', 3, 2) = 14
LENGTH(s)	대상 문자열의 글자 수를 반환 예 LENGTH('CANDIDE') = 7

## 2.2 스칼라 부속질의 – SELECT 부속질의

질의 4-17 마당서점의 고객별 판매액을 보이시오(결과는 고객이름과 고객별 판매액을 출력).

```
SELECT  (SELECT      name  
          FROM        Customer cs  
          WHERE       cs.custid=od.custid) "name", SUM(saleprice) "total"  
FROM    Orders od  
GROUP BY od.custid;
```

name	total
박지성	39000
김연아	15000
추신수	33000
장미란	31000

## 2.2 스칼라 부속질의 – SELECT 부속질의

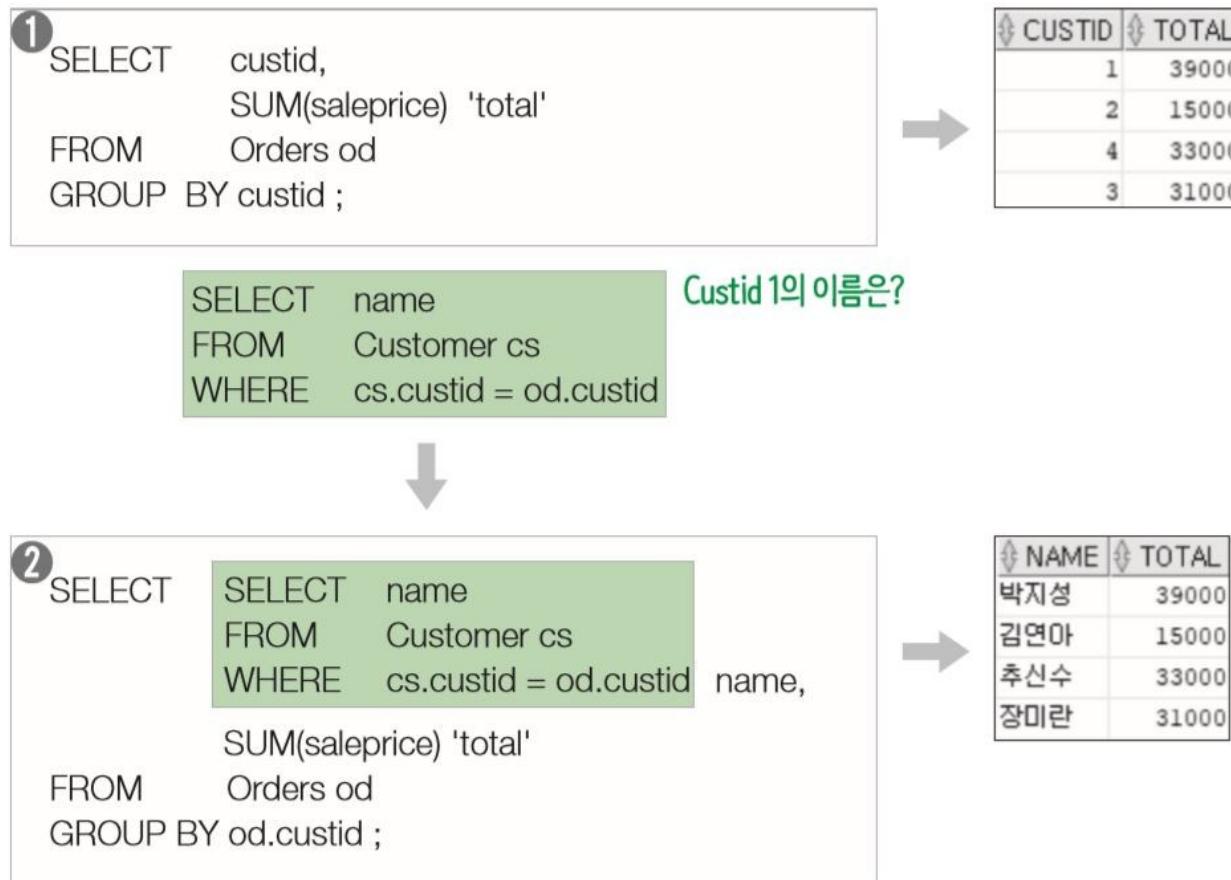


그림 4-5 마당서점의 고객별 판매액

## 2.2 스칼라 부속질의 – SELECT 부속질의

질의 4-18 Orders 테이블에 각 주문에 맞는 도서이름을 입력하시오.

- 먼저 Orders 테이블에 bookname 속성을 추가

```
ALTER TABLE Orders ADD bookname VARCHAR2(40);
UPDATE    Orders
SET        bookname=(SELECT      bookname
                      FROM        Book
                      WHERE       Book.bookid=Orders.bookid);
```

ORDERID	CUSTID	BOOKID	SALEPRICE	ORDERDATE	BOOKNAME
1	1	1	6000	20/07/01	축구의 역사
2	1	3	21000	20/07/03	축구의 이해
3	2	5	8000	20/07/03	피겨 교본
4	3	6	6000	20/07/04	역도 단계별기술
5	4	7	20000	20/07/05	야구의 추억
6	1	2	12000	20/07/07	축구하는 여자
7	4	8	13000	20/07/07	야구를 부탁해
8	3	10	12000	20/07/08	Olympic Champions
9	2	10	7000	20/07/09	Olympic Champions
10	3	8	13000	20/07/10	야구를 부탁해

## 2.3 인라인 뷰- FROM 부속질의

### ■ 인라인 뷰(inline view)란?

- FROM 절에서 사용되는 부속질의.
- 테이블 이름 대신 인라인 뷰 부속질의를 사용하면 보통의 테이블과 같은 형태로 사용할 수 있음.
- 부속질의 결과 반환되는 데이터는 다중 행, 다중 열이어도 상관없음.
- 다만 가상의 테이블인 뷰 형태로 제공되어 상관 부속질의로 사용될 수는 없음.

질의 4-19 고객번호가 2 이하인 고객의 판매액을 보이시오

(결과는 고객이름과 고객별 판매액 출력)

```
SELECT      cs.name, SUM(od.saleprice) "total"
FROM        (SELECT      custid, name
            FROM        Customer
            WHERE       custid <= 2) cs,
            Orders    od
WHERE       cs.custid=od.custid
GROUP BY   cs.name;
```

NAME	total
박지성	39000
김연마	15000

## 2.3 인라인 뷰- FROM 부속질의

주 질의

```
SELECT cs.name, SUM(od.saleprice) "total"
FROM (SELECT custid, name
      FROM Customer
     WHERE custid <= 2)
      Orders od
     WHERE cs.custid = od.custid
     GROUP BY cs.name ;
```

인라인 뷰

그림 4-6 인라인 뷰

# 연습문제

## 5. 부속질의에 관한 다음 SQL 문을 수행해보고 어떤 질의에 대한 답인지 설명하시오.

(1) SELECT            custid, (SELECT address  
                        FROM Customer cs  
                        WHERE cs.custid = od.custid) "address",  
                        SUM(saleprice) "total"  
                FROM Orders od  
                GROUP BY od.custid;

(2) SELECT            cs.name, s  
                FROM (SELECT custid, AVG(saleprice) s  
                        FROM Orders  
                        GROUP BY custid) od, Customer cs  
                WHERE cs.custid = od.custid;

(3) SELECT            SUM(saleprice) "total"  
                FROM Orders od  
                WHERE EXISTS (SELECT \*  
                        FROM Customer cs  
                        WHERE custid <= 3 AND cs.custid = od.custid);

## 03. 뷰

---

- 뷰의 생성
- 뷰의 수정
- 뷰의 삭제

## 03. 뷰

■ 뷰(view)는 하나 이상의 테이블을 합하여 만든 가상의 테이블.

### ■ 장점

- 편리성( 및 재사용성) : 자주 사용되는 복잡한 질의를 뷰로 미리 정의해 놓을 수 있음.  
=> 복잡한 질의를 간단히 작성, 미리 정의된 뷰를 일반 테이블처럼 사용하여 편리함
- 보안성 : 각 사용자 별로 필요 한 데이터만 선별하여 보여줄 수 있음. 중요한 질의의 경우 질의 내용을 암호화할 수 있음.  
=> 개인정보(주민번호)나 급여, 건강 같은 민감한 정보를 제외한 테이블을 만들어 사용
- 논리적 데이터 독립성 제공 :  
=> 개념 스키마의 데이터베이스 구조가 변하여도 외부 스키마에 영향을 주지 않도록 하는 논리적 데이터 독립성 제공

### ■ 뷰의 특징

- 1. 원본 데이터 값에 따라 같이 변함
- 2. 독립적인 인덱스 생성이 어려움
- 3. 삽입, 삭제, 갱신 연산에 많은 제약이 따름

## 03. 뷰

뷰 Vorders

ORDERID	CUSTID	NAME	BOOKID	BOOKNAME	SALEPRICE	ORDERDATE
1	1	박지성	1	축구의 역사	6000	20/07/01
6	1	박지성	2	축구마는 여자	12000	20/07/07
2	1	박지성	3	축구의 이해	21000	20/07/03
3	2	김연아	5	피겨 교본	8000	20/07/03

뷰 생성문

```
CREATE VIEW Vorders
AS SELECT O.orderid, O.custid, C.name, O.bookid, B.bookname, O.saleprice, O.orderdate
FROM Orders O, Customer C, Book B
WHERE O.custid=C.custid AND O.bookid=B.bookid;
```

기본 릴레이션 Customer, Orders, Book

Diagram illustrating the basic relationships between Customer, Orders, and Book tables:

- Customer (C)**: Contains columns CUSTID, NAME, ADDRESS, and PHONE.
- Orders (O)**: Contains columns ORDERID, CUSTID, BOOKID, SALEPRICE, and ORDERDATE.
- Book (B)**: Contains columns BOOKID, BOOKNAME, PUBLISHER, and PRICE.

The diagram shows the following relationships:

- Dashed arrows point from Customer (C) to Orders (O).
- Dashed arrows point from Customer (C) to Book (B).
- Dashed arrows point from Orders (O) to Customer (C).
- Dashed arrows point from Orders (O) to Book (B).
- A dashed arrow points from Book (B) to Customer (C).

CUSTID	NAME	ADDRESS	PHONE
1	박지성	영국 맨체스터	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001

ORDERID	CUSTID	BOOKID	SALEPRICE	ORDERDATE
1	1	1	6000	20/07/01
2	1	3	21000	20/07/03
3	2	5	8000	20/07/03
4	3	6	6000	20/07/04
5	4	7	20000	20/07/05

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
2	축구마는 여자	나무수	13000
3	축구의 이해	대한미디어	22000

그림 4-7 뷰

212

## 3.1 뷰의 생성

### ■ 기본 문법

```
CREATE VIEW 뷰이름 [(열이름 [... n])]  
AS <SELECT 문>
```

### ■ Book 테이블에서 '축구'라는 문구가 포함된 자료만 보여주는 뷰

```
SELECT      *  
FROM        Book  
WHERE       bookname LIKE '%축구%';
```

### ■ 위 SELECT 문을 이용해 작성한 뷰 정의문

```
CREATE VIEW vw_Book  
AS SELECT      *  
              FROM Book  
              WHERE bookname LIKE '%축구%';
```

## 3.1 뷰의 생성

질의 4-20 주소에 '대한민국'을 포함하는 고객들로 구성된 뷰를 만들고 조회하시오.

뷰의 이름은 vw\_Customer로 설정하시오

```
CREATE VIEW vw_Customer  
AS SELECT *  
FROM Customer  
WHERE address LIKE '%대한민국';
```

- 결과화이

```
SELECT *  
FROM vw_Customer;
```

CUSTID	NAME	ADDRESS	PHONE
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
5	박세리	대한민국 대전	(null)

## 3.1 뷰의 생성

질의 4-21 Orders 테이블에 고객이름과 도서이름을 바로 확인할 수 있는 뷰를 생성한 후,  
‘김연아’ 고객이 구입한 도서의 주문번호, 도서이름, 주문액을 보이시오.

```
CREATE VIEW vw_Orders (orderid, custid, name, bookid, bookname, saleprice,
orderdate)
AS SELECT od.orderid, od.custid, cs.name,
od.bookid, bk.bookname, od.saleprice, od.orderdate
FROM Orders od, Customer cs, Book bk
WHERE od.custid=cs.custid AND od.bookid=bk.bookid;
```

### ■ 결과화이

```
SELECT      orderid, bookname, saleprice
FROM        vw_Orders
WHERE       name = '김연아';
```

◆ ORDERID	◆ BOOKNAME	◆ SALEPRICE
3 피겨 교본		8000
9 Olympic Champions		7000

## 3.2 뷰의 수정

### ■ 기본 문법

```
CREATE OR REPLACE VIEW 뷰이름 [(열이름 [, ... n])]  
AS SELECT 문
```

질의 4-22 [질의 4-20]에서 생성한 뷰 vw\_Customer는 주소가 대한민국인 고객을 보여준다.  
이 뷰를 영국을 주소로 가진 고객으로 변경하시오.  
phone 속성은 필요 없으므로 포함시키지 마시오.

```
CREATE OR REPLACE VIEW vw_Customer (custid, name, address)  
AS SELECT      custid, name, address  
              FROM Customer  
              WHERE      address LIKE '%영국%';
```

### ■ 결과화이

```
SELECT      *  
FROM        vw_Customer;
```

CUSTID	NAME	ADDRESS
1	박지성	영국 맨체스터

### 3.3 뷰의 삭제

#### ■ 기본 문법

```
DROP VIEW 뷰이름 [, ... n];
```

질의 4-23 앞서 생성한 뷰 vw\_Customer를 삭제하시오.

```
DROP VIEW vw_Customer;
```

#### ■ 결과화이

```
SELECT      *
FROM        vw_Customer;
```

ORA-00942: 테이블 또는 뷰가 존재하지 않습니다  
00942. 00000 - "table or view does not exist"  
\*Cause:  
\*Action:  
4행, 6열에서 오류 발생

## 연습문제

08. 다음에 해당하는 뷰를 작성하시오. 데이터베이스는 마당서점 데이터베이스를 이용한다.

- (1) 판매가격이 20,000원 이상인 도서의 도서번호, 도서이름, 고객이름, 출판사, 판매가격을 보여주는 highorders 뷰를 생성하시오.
- (2) 생성한 뷰를 이용하여 판매된 도서의 이름과 고객의 이름을 출력하는 SQL 문을 작성하시오.
- (3) highorders 뷰를 변경하고자 한다. 판매가격 속성을 삭제하는 명령을 수행하시오.  
삭제 후 (2)번 SQL 문을 다시 수행하시오.

## 04. 인덱스

---

- 데이터베이스의 물리적 저장
- 인덱스와 B-tree
- 오라클 인덱스
- 인덱스의 생성
- 인덱스의 재구성과 삭제

## 4.1 데이터베이스의 물리적 저장

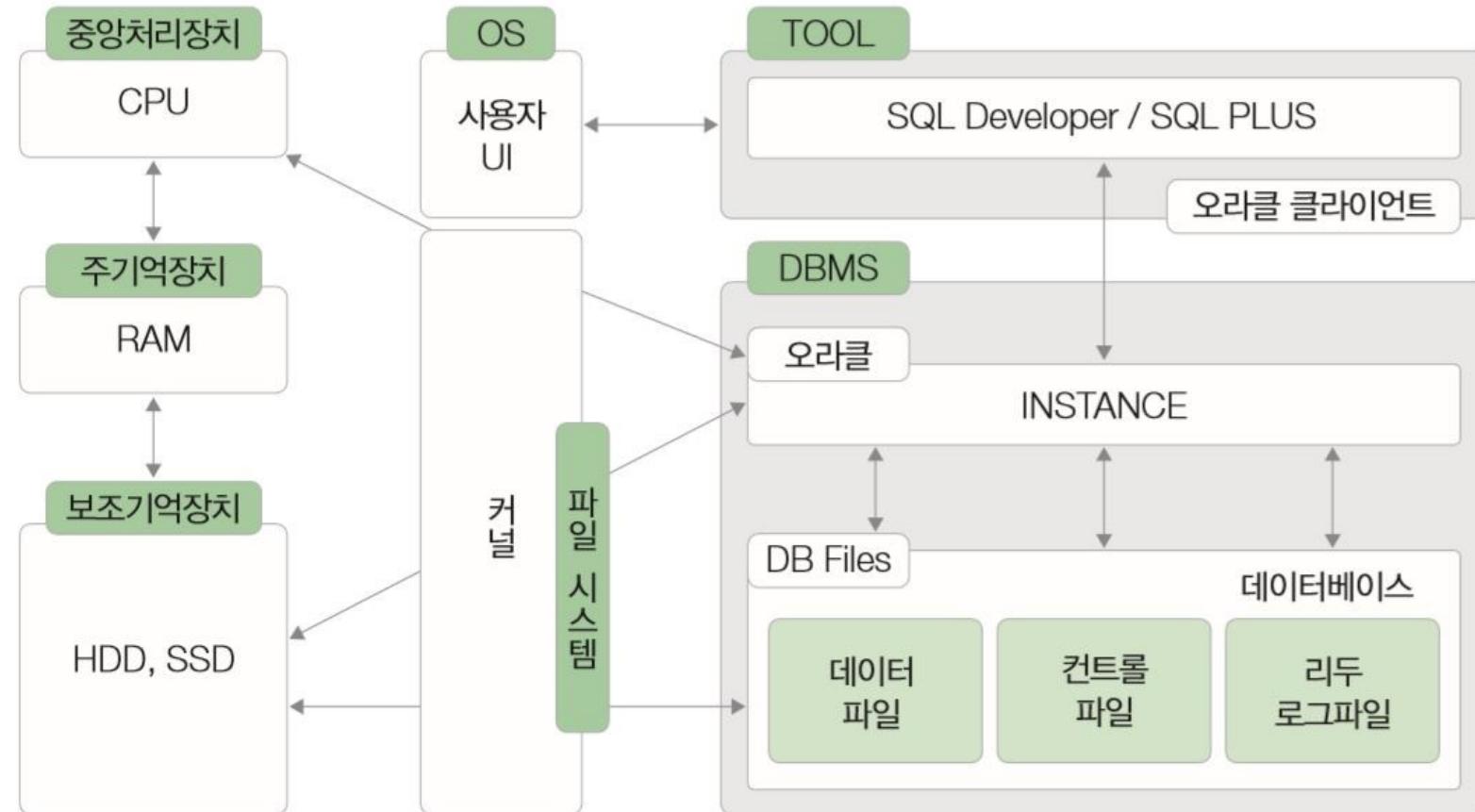


그림 4-8 DBMS와 데이터 파일

## 4.1 데이터베이스의 물리적 저장

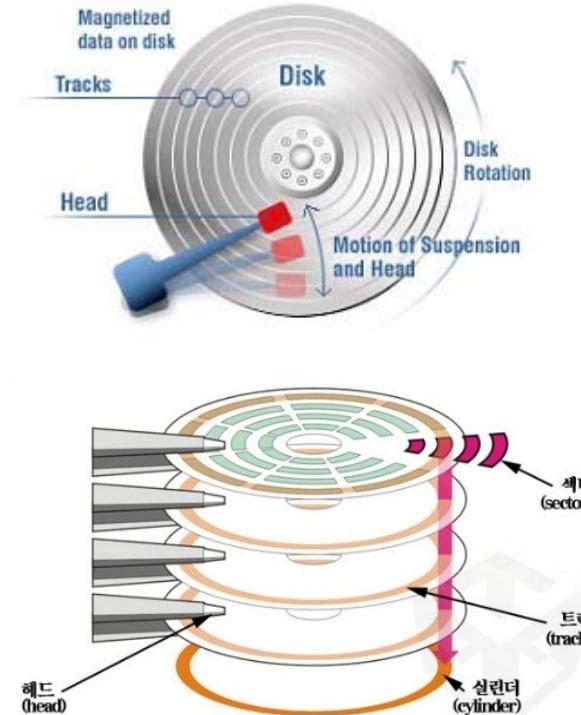
---

- 실제 데이터가 저장되는 곳은 보조기억장치
  - 하드디스크, SSD, USB 메모리 등
  
- 가장 많이 사용되는 장치는 하드디스크
  - 하드디스크는 원형의 플레이트(plate)로 구성되어 있고, 이 플레이트는 논리적으로 트랙으로 나뉘며 트랙은 다시 몇 개의 섹터로 나뉨.
  - 원형의 플레이트는 초당 빠른 속도로 회전하고, 회전하는 플레이트를 하드디스크의 액세스 앰(arm)과 헤더(header)가 접근하여 원하는 섹터에서 데이터를 가져옴.
  - 하드디스크에 저장된 데이터를 읽어 오는 데 걸리는 시간은 모터(motor)에 의해서 분당 회전하는 속도(RPM, Revolutions Per Minute), 데이터를 읽을 때 액세스 앰이 이동하는 시간(latency time), 주기억장치로 읽어오는 시간(transfer time)에 영향을 받음.

## 4.1 데이터베이스의 물리적 저장



그림 4-9 하드디스크의 구조



### ■ 액세스 시간(access time)

- 액세스 시간 = 탐색시간(seek time, 액세스 헤드를 트랙에 이동시키는 시간)
  - + 회전지연시간(rotational latency time, 섹터가 액세스 헤드에 접근하는 시간)
  - + 데이터 전송시간(data transfer time, 데이터를 주기억장치로 읽어오는 시간)

## 4.1 데이터베이스의 물리적 저장

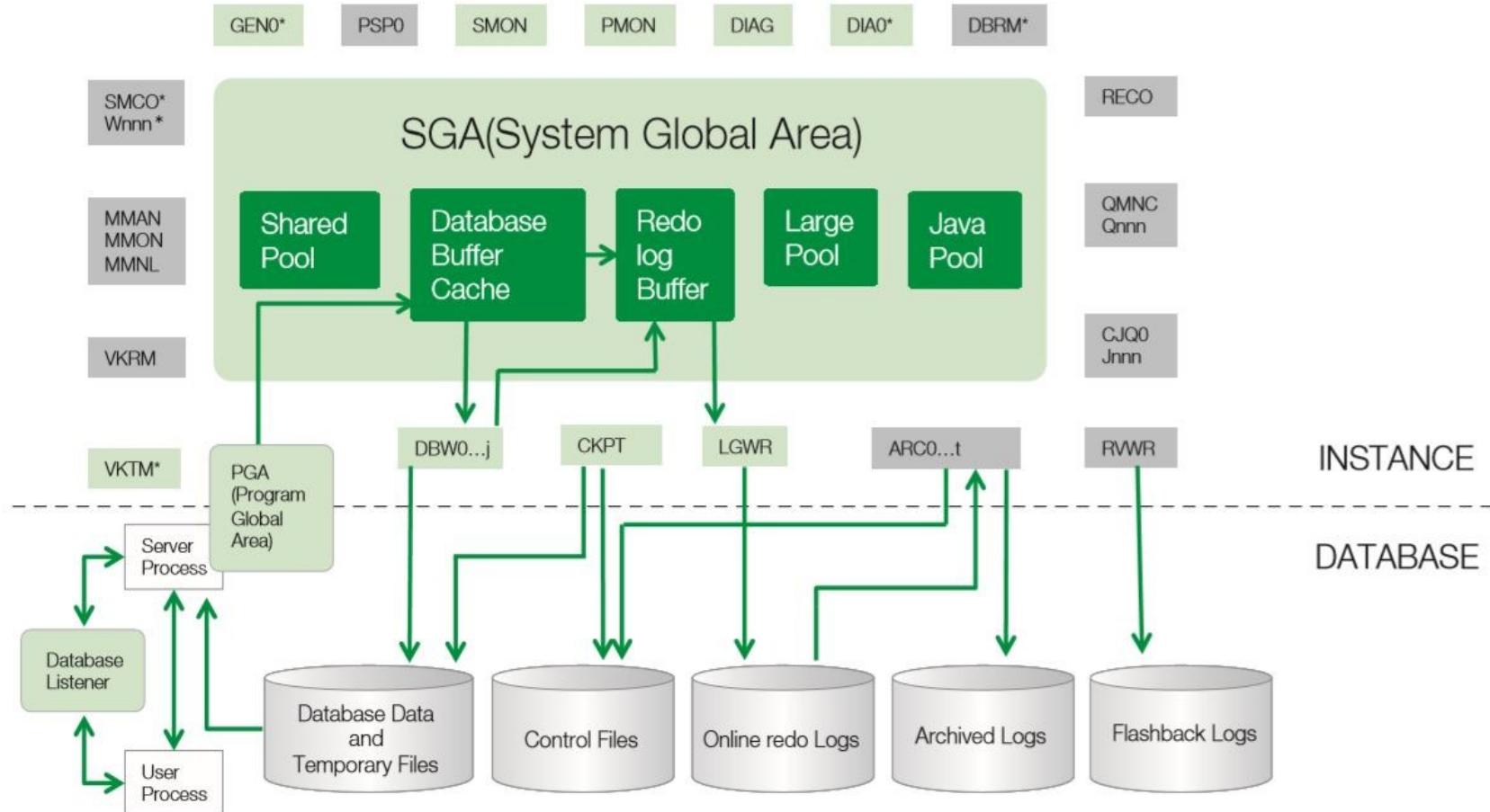


그림 4-10 오라클의 내부 구조

## 4.1 데이터베이스의 물리적 저장

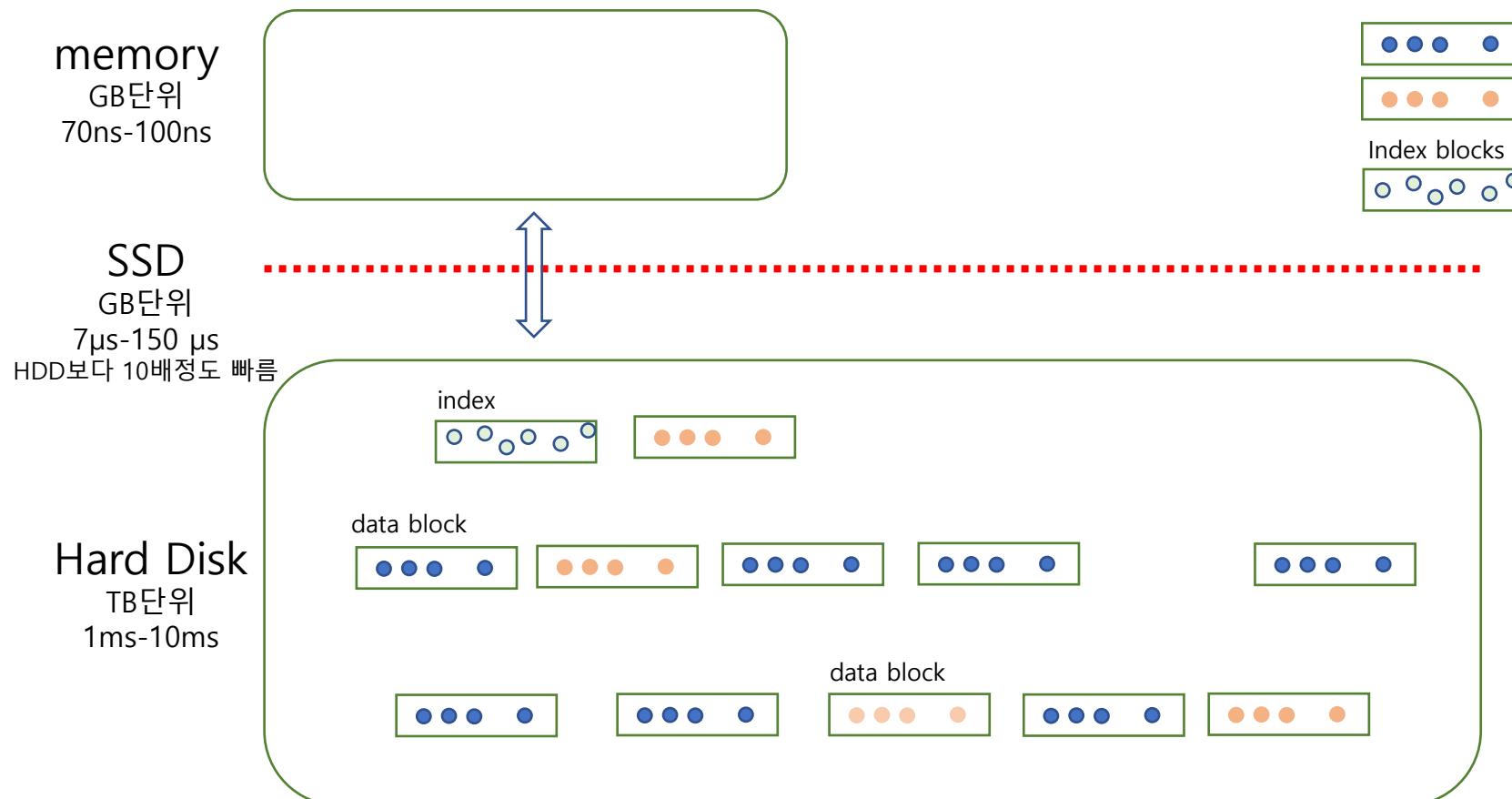
표 4-10 오라클의 주요 파일

파일	설명
데이터 파일 (Data Files)	<ul style="list-style-type: none"><li>운영체제상에 물리적으로 존재</li><li>사용자 데이터와 객체를 저장</li><li>테이블과 인덱스로 구성</li></ul>
온라인 리두 로그 (Online Redo Log)	<ul style="list-style-type: none"><li>데이터의 모든 변경사항을 기록</li><li>데이터베이스 복구에 사용되는 로그 정보 저장</li><li>최소 두 개의 온라인 리두 로그 파일 그룹을 가짐</li></ul>
컨트롤 파일 (Control File)	<ul style="list-style-type: none"><li>오라클이 필요로 하는 다른 파일들(데이터 파일, 로그 파일 등)의 위치 정보를 저장</li><li>데이터베이스 구조 등의 변경사항이 있을 때 자동으로 업데이트됨</li><li>오라클 DB의 마운트, 오픈의 필수 파일</li><li>복구 시 동기화 정보 저장</li></ul>

## 4.2 인덱스와 B-tree

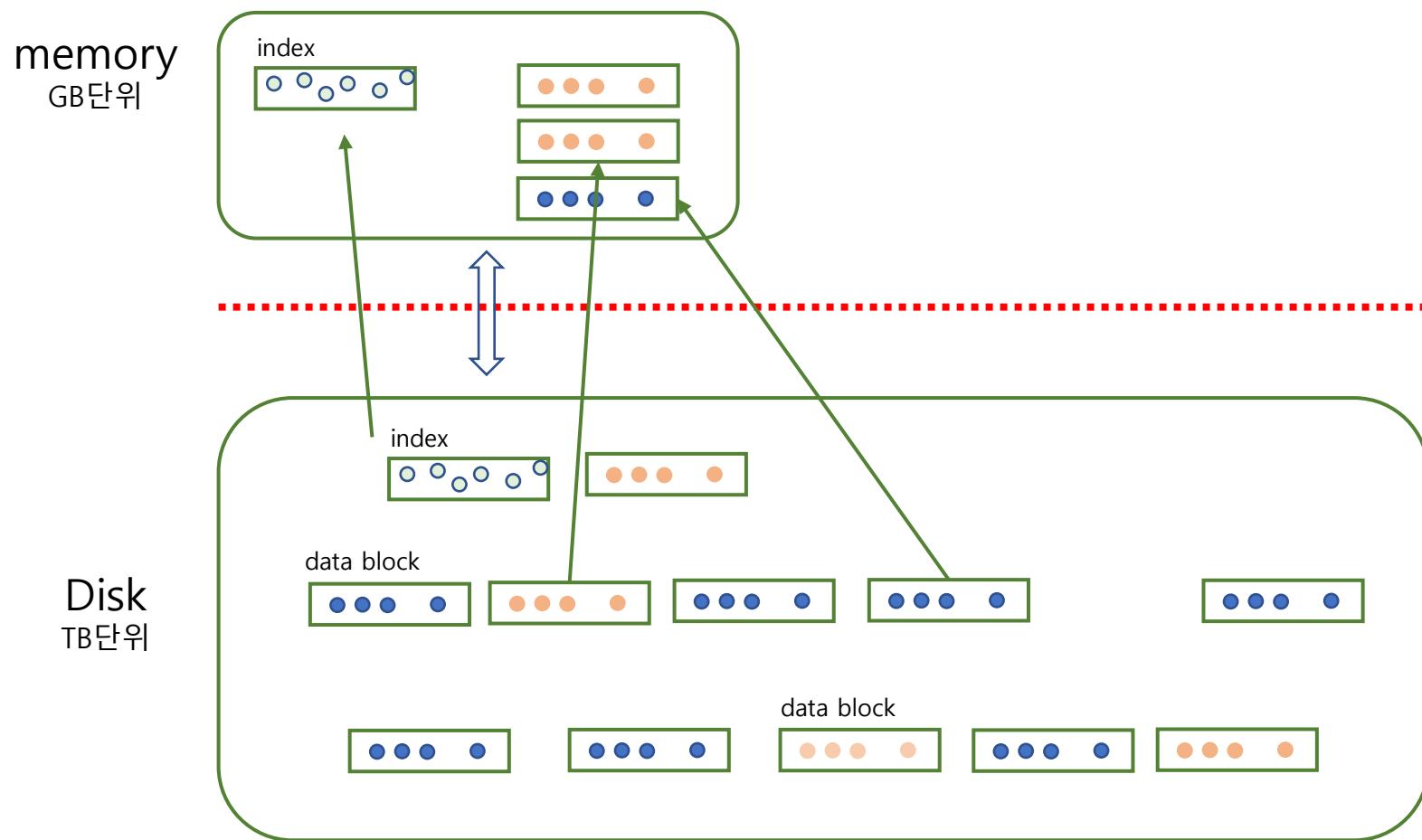
### ■ 인덱스의 필요성

질의 검색시 data block(여러 개의 record를 저장하는 저장단위, 2KB 4KB ...) 읽는 횟수를 최소화 필요  
disk에 있는 데이터는 memory에 있는 데이터에 비하여 읽어들이는 속도가 **10000배** 정도 소요된다.



## 4.2 인덱스와 B-tree

### ■ 인덱스의 필요성 memory



## 4.2 인덱스와 B-tree

### ■ 인덱스(index, 색인)

- 도서의 색인이나 사전과 같이 데이터를 쉽고 빠르게 찾을 수 있도록 만든 데이터 구조

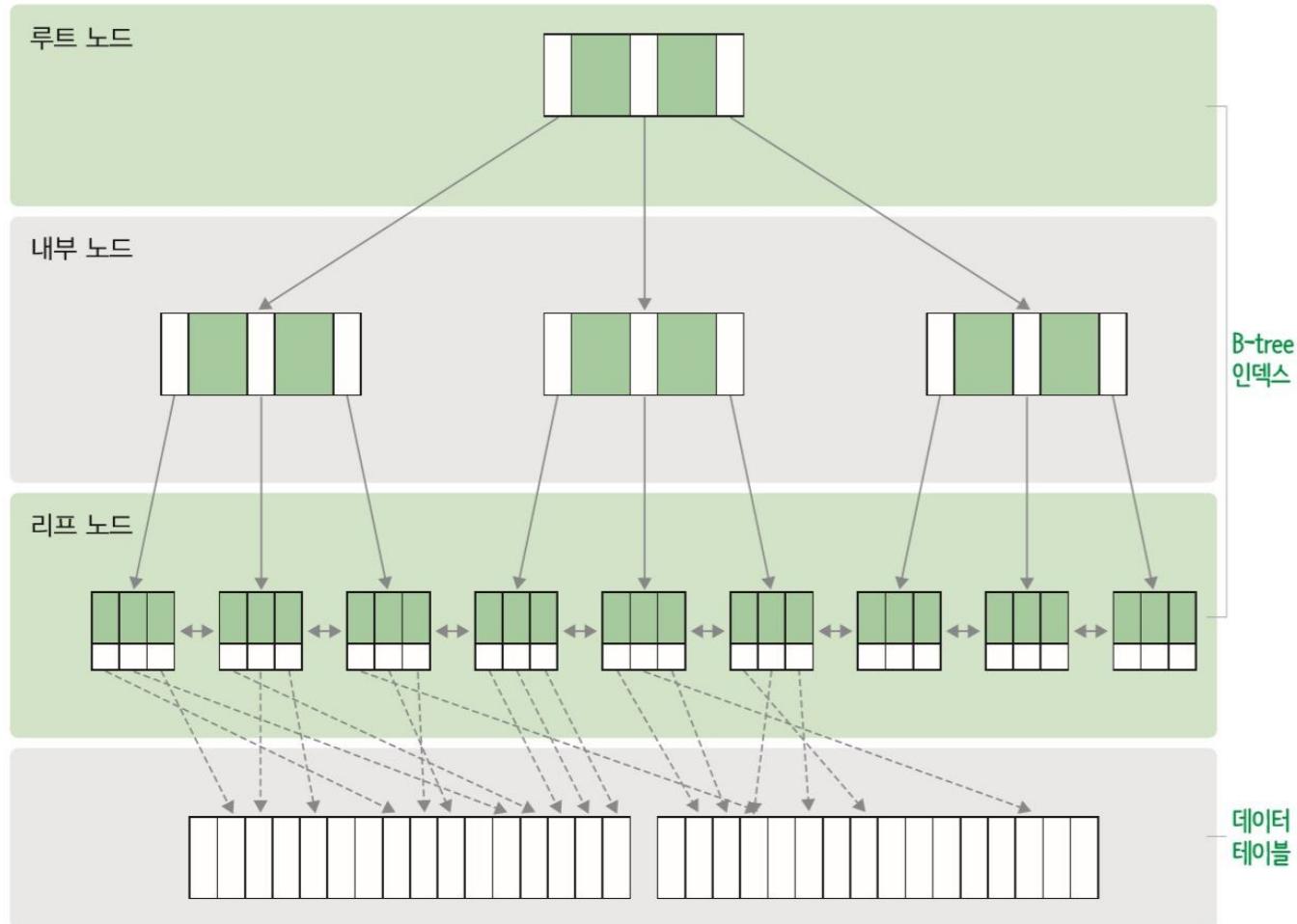


그림 4-12 B-tree의 구조

## 4.2 인덱스와 B-tree

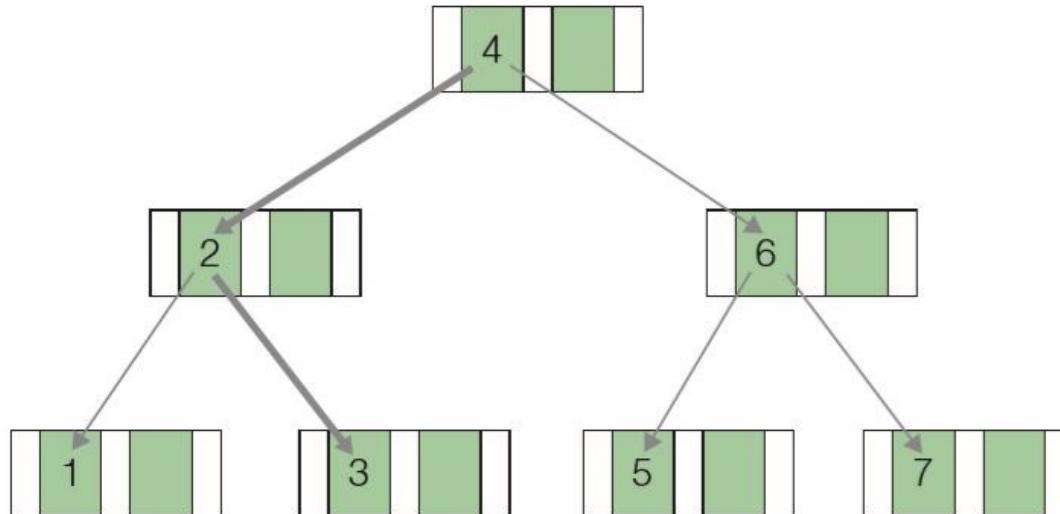


그림 4-13 B-tree에서 검색 예

### ■ 인덱스의 특징

- 인덱스는 테이블에서 한 개 이상의 속성을 이용하여 생성함.
- 빠른 검색과 함께 효율적인 레코드 접근이 가능함.
- 순서대로 정렬된 속성과 데이터의 위치만 보유하므로 테이블보다 작은 공간을 차지함.
- 저장된 값들은 테이블의 부분집합이 됨.
- 일반적으로 B-tree 형태의 구조를 가짐.
- 데이터의 수정, 삭제 등의 변경이 발생하면 인덱스의 재구성이 필요함.

# 오라클 B-tree 인덱스

- 오라클 인덱스는 B-tree를 변형하여 사용하며 명칭은 B-tree로 동일한 이름으로 부름

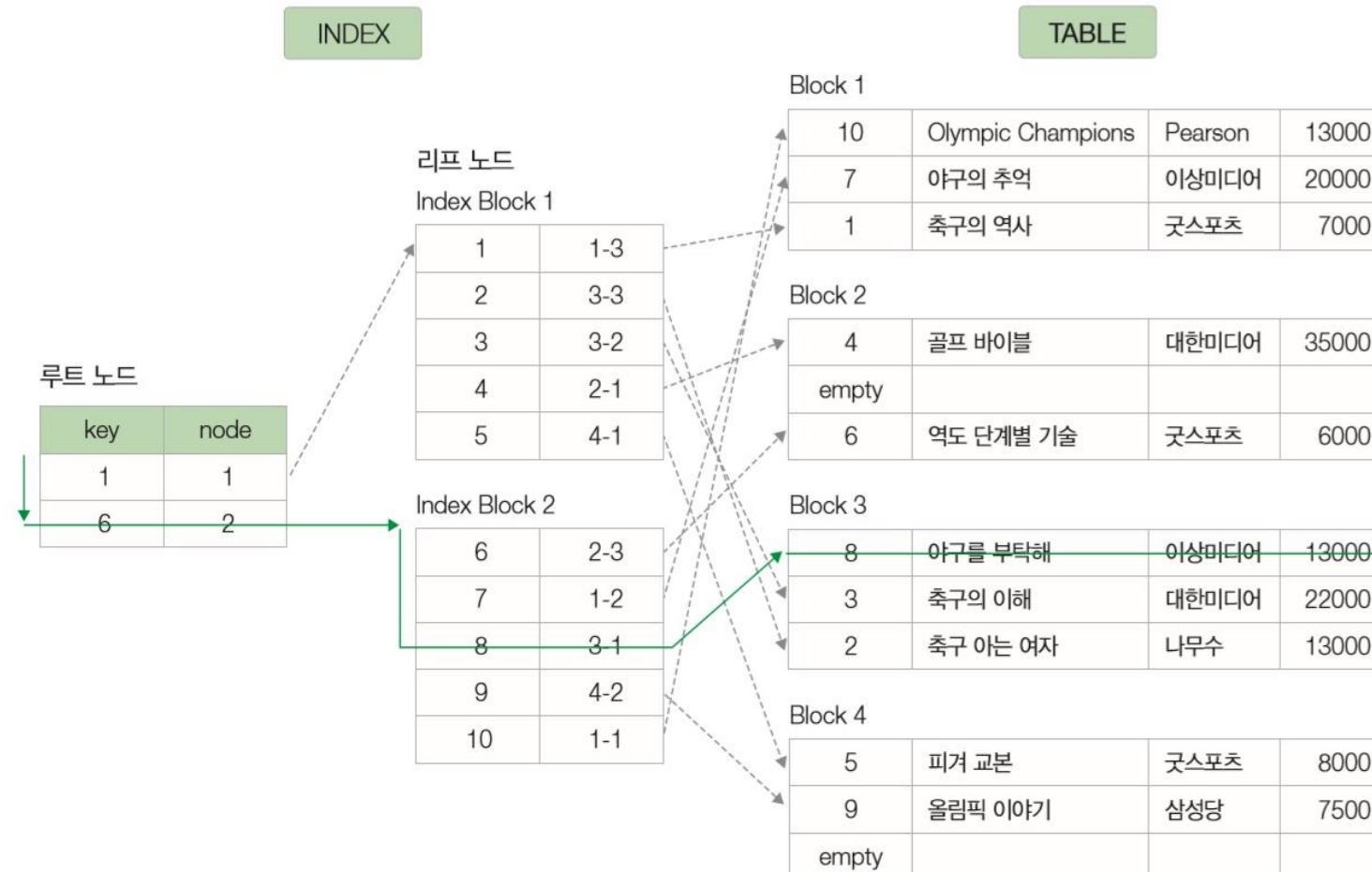


그림 4-14 인덱스의 예

## 4.2 인덱스와 B-tree

표 4-11 오라클 인덱스의 종류

인덱스 명칭	설명 / 생성 예
B-Tree 인덱스	기본적인 인덱스로, 하나의 리프 노드는 하나의 데이터에 대응 예 CREATE INDEX t_ix ON T(key1);
IOT (Index Organized Table, 인덱스 구조 테이블)	B-Tree 구조로 키값이 정렬되면서 인덱스 리프 노드에 실제 데이터가 같이 저장되는 테이블(인덱스+테이블 일체) 예 CREATE TABLE T (key1 NUMBER, BDATA VARCHAR2(10)) ORGANIZATION INDEX;
Bitmap Index (비트맵 인덱스)	비트맵을 사용하여 하나의 엔트리가 여러 행을 가리킬 수 있도록 생성 예 CREATE BITMAP INDEX bt_ix ON Emp(gender);
Function–Base Index (함수 기반 인덱스)	행과 열에 대한 함수의 결과를 저장한 인덱스 예 CREATE INDEX fbi_ix ON T(UPPER(name));

## 4.4 인덱스의 생성

### ■ 인덱스 생성 시 고려사항

- 인덱스는 WHERE 절에 자주 사용되는 속성이어야 함.
- 인덱스는 조인에 자주 사용되는 속성이어야 함.
- 단일 테이블에 인덱스가 많으면 속도가 느려질 수 있음(테이블 당 4~5개 정도 권장).
- 속성이 가공되는 경우 사용하지 않음.
- 속성의 선택도가 낮을 때 유리함(속성의 모든 값이 다른 경우).

### ■ 인덱스의 생성 문법

```
CREATE [REVERSE] [UNIQUE] INDEX [인덱스이름]  
ON 테이블이름 (컬럼 [ASC | DESC] [{, 컬럼 [ASC | DESC]} ...])[]
```