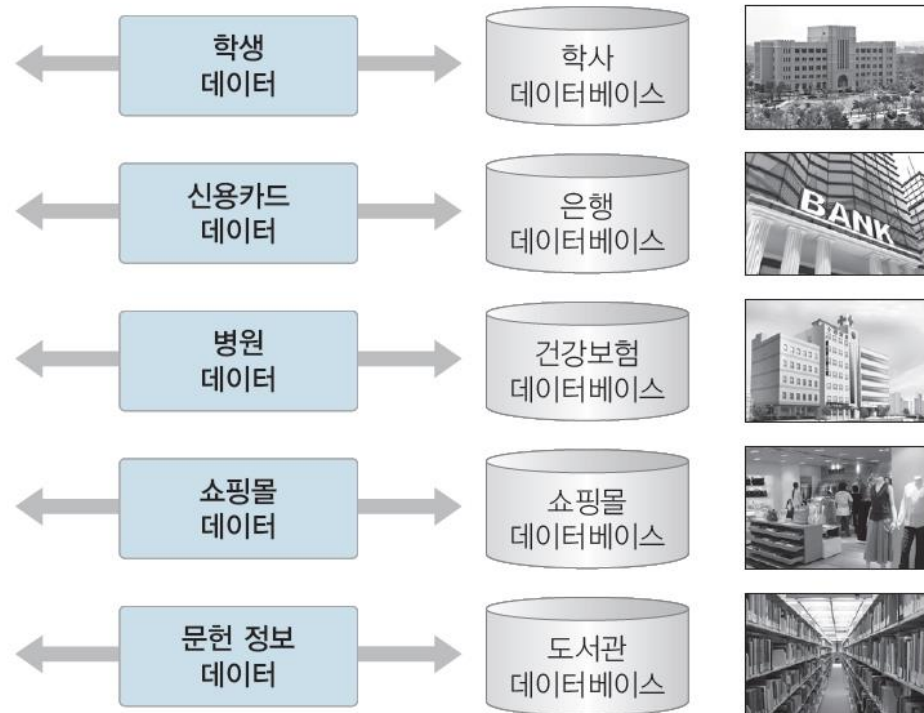
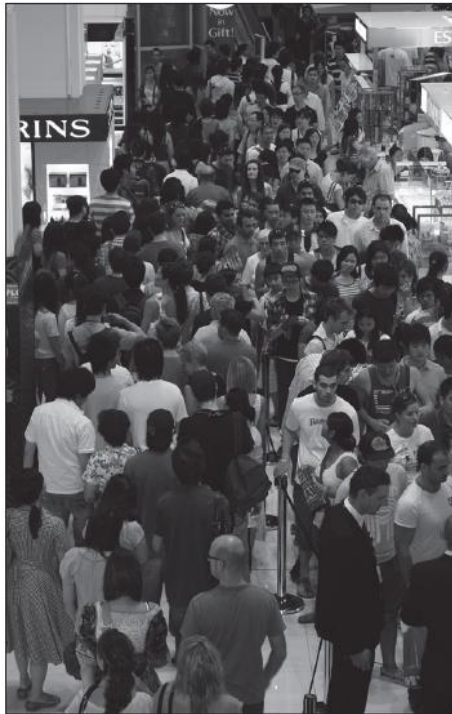


데이터베이스

❖ 데이터베이스란?

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



데이터베이스의 개념 및 특징

❖ 데이터베이스의 개념

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 데이터, 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

데이터베이스의 개념 및 특징

❖ 데이터베이스의 특징

① 실시간 접근성(real time accessibility)

데이터베이스는 실시간으로 서비스된다. 사용자가 데이터를 요청하면 몇 시간이나 몇 일 뒤에 결과를 전송하는 것이 아니라 수 초 내에 결과를 서비스한다.

② 계속적인 변화(continuous change)

데이터베이스에 저장된 내용은 어느 한 순간의 상태를 나타내지만, 데이터 값은 시간에 따라 항상 바뀐다. 데이터베이스는 삽입, 삭제, 수정 등의 작업을 통하여 바뀐 데이터 값을 저장한다.

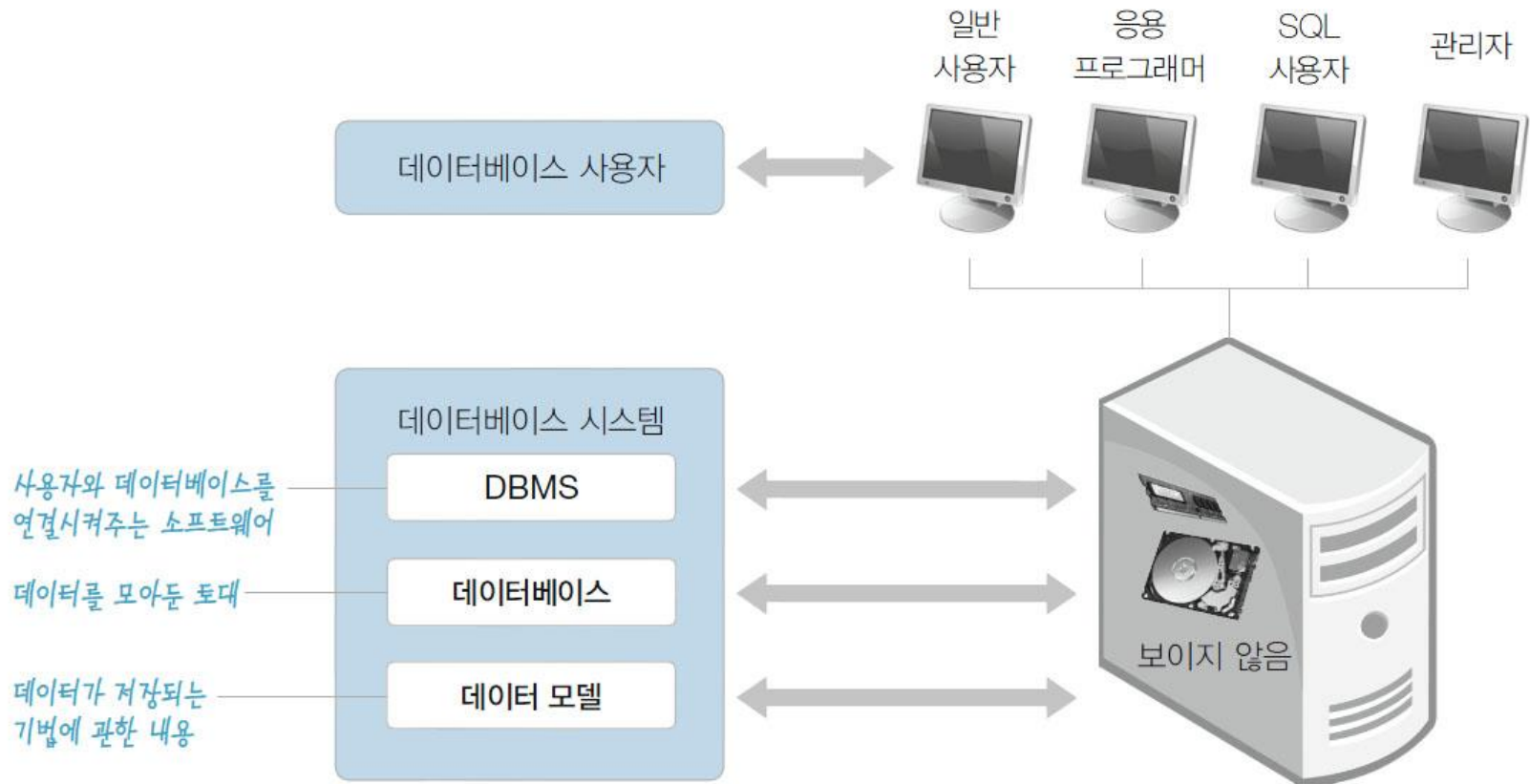
③ 동시 공유(concurrent sharing)

데이터베이스는 서로 다른 업무 또는 여러 사용자에게 동시에 공유된다. 동시는 병행이라고도 하며, 데이터베이스에 접근하는 프로그램이 여러 개 있다는 의미이다.

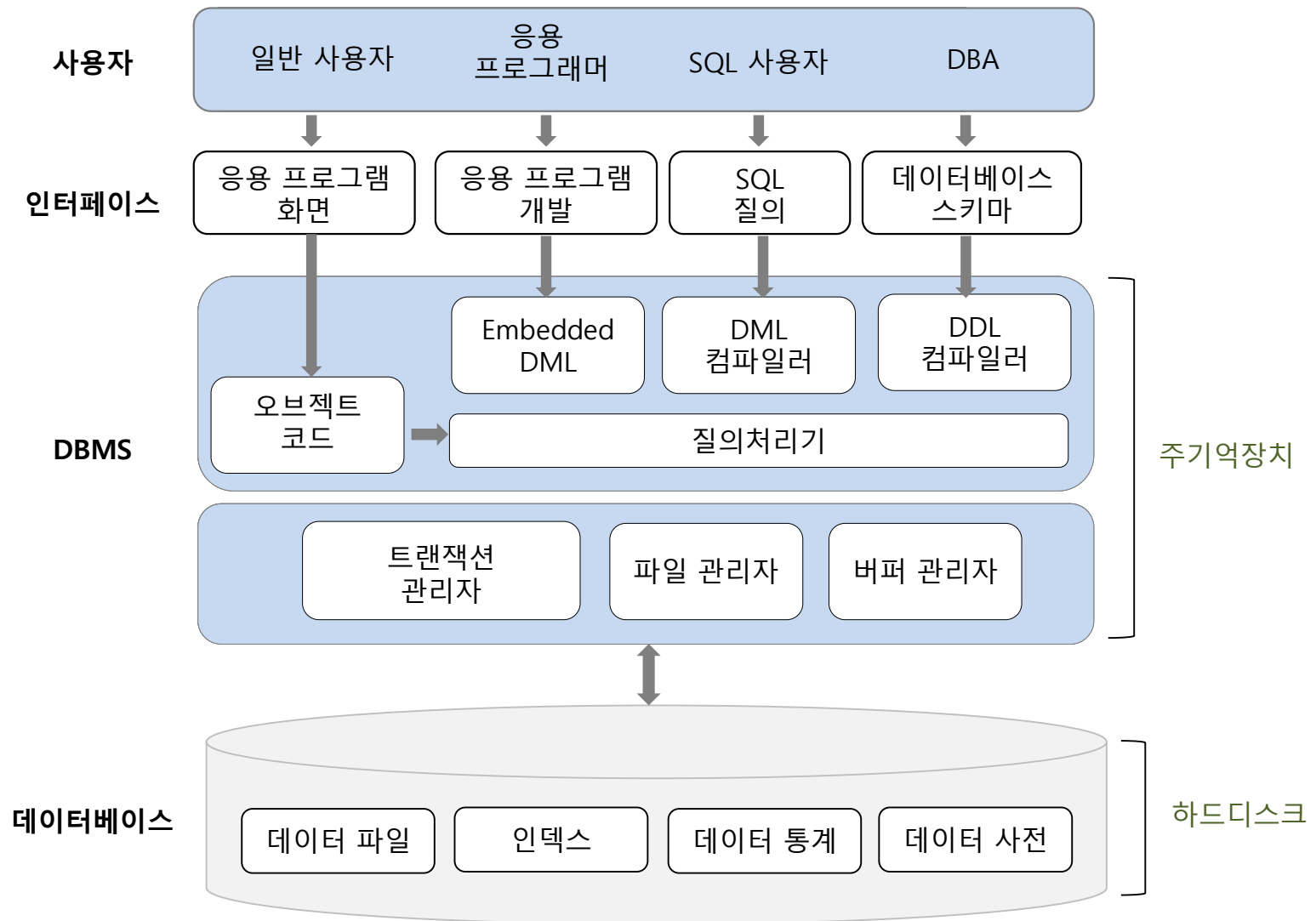
④ 내용에 따른 참조(reference by content)

데이터베이스에 저장된 데이터는 데이터의 물리적인 위치가 아니라 데이터 값에 따라 참조된다.

데이터베이스 시스템의 구성



데이터베이스 시스템의 구성



데이터베이스 언어

■ SQL

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

질의 → Book 테이블에서 모든 도서이름(bookname)과 출판사(publisher)를 검색하시오.

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

Book 테이블

bookid	bookname	publisher	price
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

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

데이터베이스 언어

질의 → 가격(price)이 10,000원 이상인 도서이름(bookname)과 출판사(publisher)를 검색하시오.

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

Book 테이블

bookid	bookname	publisher	price
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

bookname	publisher
축구아는 여자	나무수
축구의 이해	대한미디어
골프 바이블	대한미디어

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 등 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

데이터베이스 사용자

데이터베이스 사용자 별로 갖추어야 할 지식 수준(× : 없음, ○ : 보통, ◎ : 높음)

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

DBMS

DBMS의 기능

데이터 정의(Definition)	데이터의 구조를 정의하고 데이터 구조에 대한 삭제 및 변경 기능을 수행함
데이터 조작(manipulation)	데이터를 조작하는 소프트웨어(응용 프로그램)가 요청하는 데이터의 삽입, 수정, 삭제 작업을 지원함
데이터 추출(Retrieval)	사용자가 조회하는 데이터 혹은 응용 프로그램의 데이터를 추출함
데이터 제어(Control)	데이터베이스 사용자를 생성하고 모니터링하며 접근을 제어함. 백업과 회복, 동시성 제어 등의 기능을 지원함

릴레이션

- 릴레이션(relation) : 행과 열로 구성된 테이블

릴레이션과 관련된 한글 용어

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

릴레이션

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



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

도서번호 = {1, 2, 3, 4, 5}

도서이름 = {축구의 역사, 축구아는 여자, 축구의 이해, 골프 바이블, 피겨 교본}

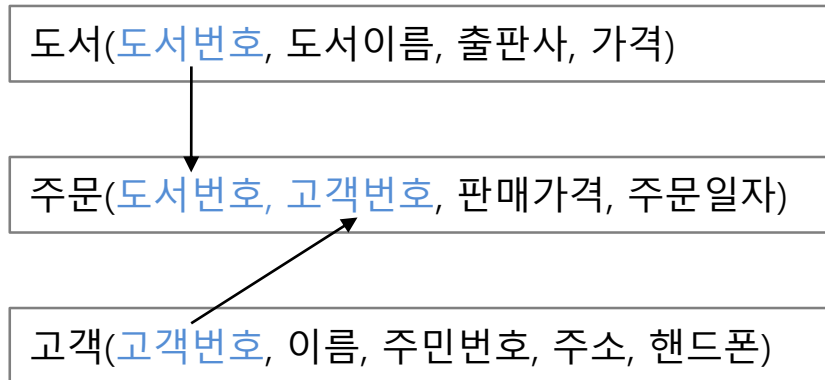
출판사 = {굿스포츠, 나무수, 대한미디어}

가격 = {7000, 13000, 22000, 35000, 8000}

릴레이션

❖ 관계(relationship)

- ❶ 릴레이션 내에서 생성되는 관계 : 릴레이션 내 데이터들의 관계
- ❷ 릴레이션 간에 생성되는 관계 : 릴레이션 간의 관계



릴레이션 스키마와 인스턴스

속성(애트리뷰트),
열(column) 이라고도 함
(차수=4)

도서

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

스키마(내포)
Schema

인스턴스(외연)
Instance

튜플(tuple),
행(row) 이라고도 함
(카디널리티=5)

릴레이션의 특징

① 속성은 단일 값을 가진다

각 속성의 값은 도메인에 정의된 값만을 가지며 그 값은 모두 단일 값이어야 함

② 속성은 서로 다른 이름을 가진다

속성은 한 릴레이션에서 서로 다른 이름을 가져야만 함

③ 한 속성의 값은 모두 같은 도메인 값을 가진다

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

④ 속성의 순서는 상관없다

속성의 순서가 달라도 릴레이션 스키마는 같음

예) 릴레이션 스키마에서 (이름, 주소) 순으로 속성을 표시하거나 (주소, 이름) 순으로 표시하여도 상관없음

릴레이션의 특징

⑤ 릴레이션 내의 중복된 튜플은 허용하지 않는다

하나의 릴레이션 인스턴스 내에서는 서로 중복된 값을 가질 수 없음,
즉 모든 튜플은 서로 값이 달라야 함

⑥ 튜플의 순서는 상관없다

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

키

- 특정 튜플을 식별할 때 사용하는 속성 혹은 속성의 집합
- 릴레이션은 중복된 튜플을 허용하지 않음 → 각각의 튜플에 포함된 속성들 중 어느 하나(혹은 하나 이상)는 값이 달라야 함. 즉 키가 되는 속성(혹은 속성의 집합)은 반드시 값이 달라서 튜플들을 서로 구별할 수 있어야 함
- 키는 릴레이션 간의 관계를 맺는 데도 사용됨

후보키

■ 튜플을 유일하게 식별할 수 있는 속성의 최소 집합

(주문 릴레이션 예)

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

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

■ 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 튜플을 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.

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

대체키

- 기본키로 선정되지 않은 후보키
- 고객 릴레이션의 경우 고객번호와 주민번호 중 고객번호를 기본키로 정하면 주민번호가 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

기본키 / 외래키

고객

고객번호	이름	주민번호	주소	핸드폰
1	박지성	810101-1111111	영국 맨체스타	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	축구아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000

기본키

참조

외래키

참조

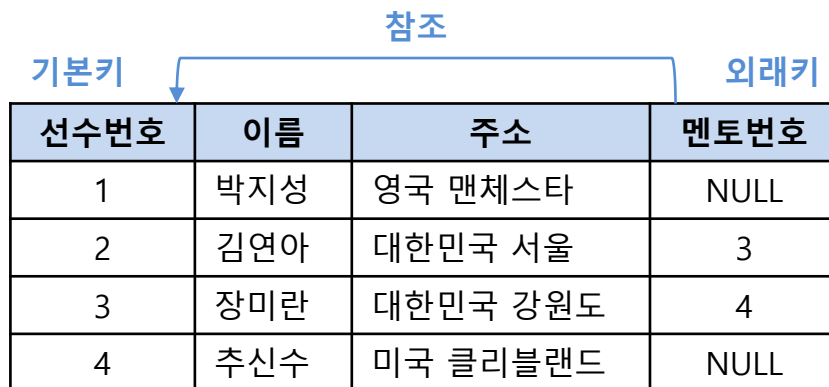
주문

주문번호	고객번호	도서번호	판매가격	주문일자
1	1	1	7000	2014-07-01
2	1	2	13000	2014-07-03
3	2	5	8000	2014-07-03
4	3	2	13000	2014-07-04
5	4	4	35000	2014-07-05
6	1	3	22000	2014-07-07
7	4	3	22000	2014-07-07

외래키

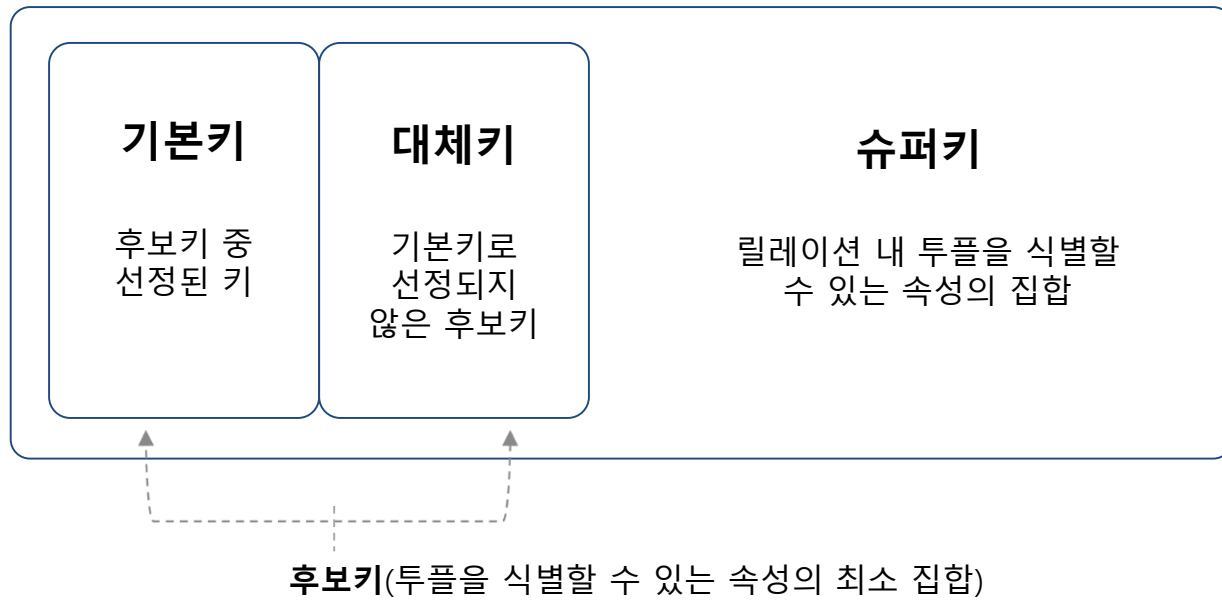
❖ 외래키

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



선수번호	이름	주소	멘토번호
1	박지성	영국 맨체스타	NULL
2	김연아	대한민국 서울	3
3	장미란	대한민국 강원도	4
4	추신수	미국 클리블랜드	NULL

키 요약



무결성 제약조건

■ 데이터 무결성(integrity, 無缺性)

데이터베이스에 저장된 데이터의 일관성과 정확성을 지키는 것

■ 도메인 무결성 제약조건

도메인 제약(domain constraint)이라고도 하며, 릴레이션 내의 튜플들이 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 제약조건임. 자식 릴레이션의 외래키는 부모 릴레이션의 기본키와 도메인이 동일해야 하며, 자식 릴레이션의 값이 변경될 때 부모 릴레이션의 제약을 받는다는 것임

무결성 제약조건

구분	도메인	키	
	도메인 무결성 제약조건	개체 무결성 제약조건	참조 무결성 제약조건
제약 대상	속성	투플	속성과 투플
같은 용어	도메인 제약 (Domain Constraint)	기본키 제약 (Primary Key Constraint)	외래키 제약 (Foreign Key Constraint)
해당되는 키	-	기본키	외래키
NULL 값 허용 여부	허용	불가	허용
릴레이션 내 제약조건의 개수	속성의 개수와 동일	1개	0~여러 개
기타	<ul style="list-style-type: none"> 투플 삽입, 수정 시 제약 사항 우선 확인 	<ul style="list-style-type: none"> 투플 삽입/수정 시 제약 사항 우선 확인 	<ul style="list-style-type: none"> 투플 삽입/수정 시 제약사항 우선 확인 부모 릴레이션의 투플 수정/삭제 시 제약사항 우선 확인

무결성 제약조건

❖ 개체 무결성 제약조건

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

(501, 남슬찬, 1001)



삽입 거부

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

(NULL, 남슬찬, 1001)



삽입 거부

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

무결성 제약조건

❖ 참조 무결성 제약조건

■ 삽입

- 학과(부모 릴레이션) : 튜플 삽입한 후 수행하면 정상적으로 진행된다.
- 학생(자식 릴레이션) : 참조받는 테이블에 외래키 값이 없으므로 삽입이 금지된다.

학생(자식 릴레이션)

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

학과(부모 릴레이션)

학과코드	학과명
1001	컴퓨터학과
2001	체육학과

참조

그림 2-14 학생관리 데이터베이스

무결성 제약조건

❖ 참조 무결성 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

- ❶ 즉시 작업을 중지
- ❷ 자식 릴레이션의 관련 튜플을 삭제
- ❸ 초기에 설정된 다른 어떤 값으로 변경
- ❹ NULL 값으로 설정

무결성 제약조건

❖ 참조 무결성 제약조건

■ 수정

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

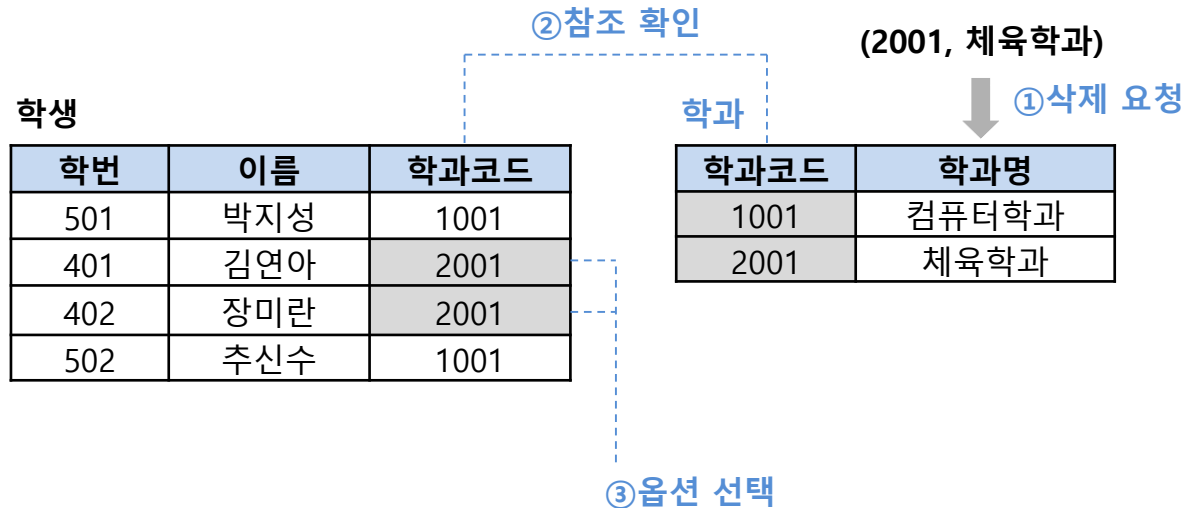
무결성 제약조건

❖ 참조 무결성 제약조건

참조 무결성 제약조건의 옵션(부모 릴레이션에서 튜플을 삭제할 경우)

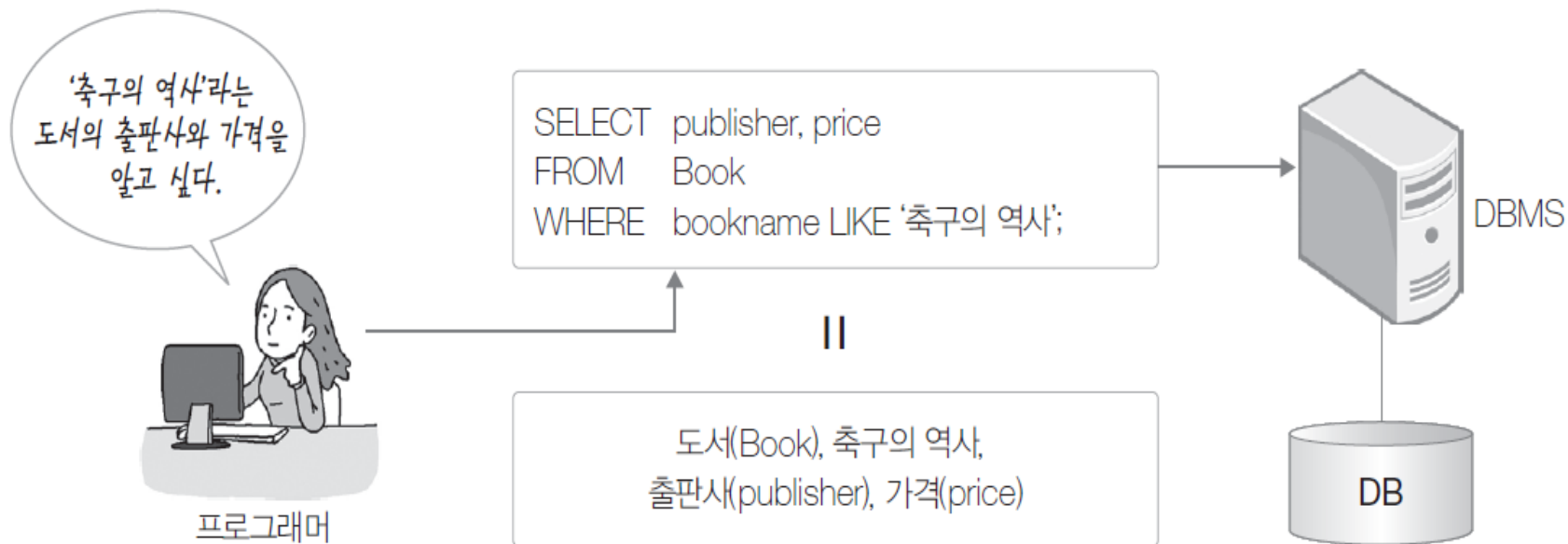
명령어	의미	예
RESTRICTED	자식 릴레이션에서 참조하고 있을 경우 부모 릴레이션의 삭제 작업을 거부함	학과 릴레이션의 튜플 삭제 거부
CASCADE	자식 릴레이션의 관련 튜플을 같이 삭제 처리함	학생 릴레이션의 관련 튜플을 삭제
DEFAULT	자식 릴레이션의 관련 튜플을 미리 설정해둔 값으로 변경함	학생 릴레이션의 학과가 다른 학과로 자동 배정
NULL	자식 릴레이션의 관련 튜플을 NULL 값으로 설정함(NULL 값을 허가한 경우)	학과 릴레이션의 학과가 NULL 값으로 변경

무결성 제약조건



참조 무결성 제약조건에서 부모 릴레이션의 투표를 삭제할 경우

SQL 개요



SQL 개요

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
사용 예	SELECT * FROM Book;	int main() {...}

SQL 개요

❖ SQL 기능에 따른 분류

■ 데이터 정의어(DDL)

테이블이나 관계의 구조를 생성하는 데 사용하며 CREATE, ALTER, DROP 문 등이 있음

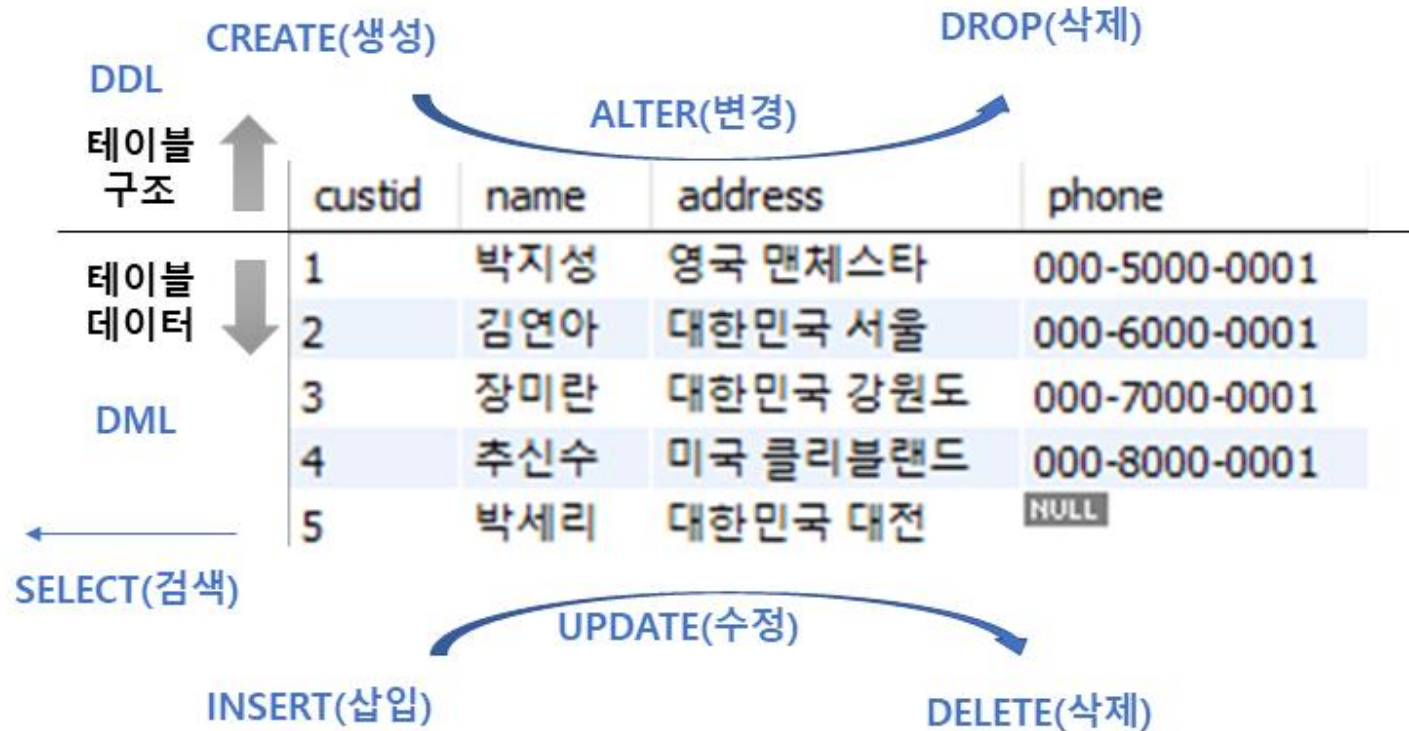
■ 데이터 조작어(DML)

테이블에 데이터를 검색, 삽입, 수정, 삭제하는 데 사용하며 SELECT, INSERT, DELETE, UPDATE 문 등이 있음. 여기서 SELECT 문은 특별히 질의어(query)라고 함

■ 데이터 제어어(DCL)

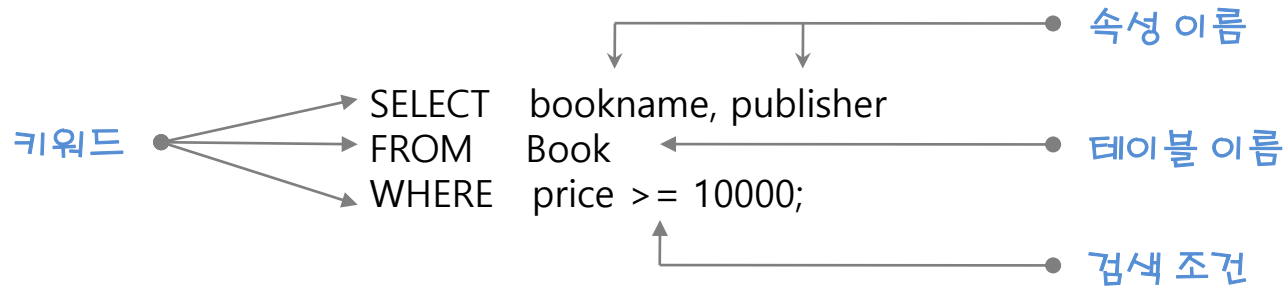
데이터의 사용 권한을 관리하는 데 사용하며 GRANT, REVOKE 문 등이 있음

SQL 개요



SELECT 문

❖ SELECT 문의 구성요소



❖ 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.

SELECT 문

❖ SELECT/FROM

질의 → 모든 도서의 이름과 가격을 검색하시오.

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

bookname	price
축구의 역사	7000
축구하는 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 → 모든 도서의 가격과 이름을 검색하시오.

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

price	bookname
7000	축구의 역사
13000	축구하는 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

❖ SELECT/FROM

질의 → 모든 도서의 도서번호, 도서이름, 출판사, 가격을 검색하시오.

```
SELECT    bookid, 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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

```
SELECT    *
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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

SELECT 문

❖ SELECT/FROM

질의 → 도서 테이블에 있는 모든 출판사를 검색하시오.

```
SELECT publisher
FROM Book;
```

publisher	
굿스포츠	
나무수	
대한미디어	
대한미디어	
굿스포츠	
굿스포츠	
이상미디어	
이상미디어	
삼성당	
Pearson	

※ 중복을 제거하고 싶으면 DISTINCT라는 키워드를 사용한다.

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

publisher	
굿스포츠	
나무수	
대한미디어	
이상미디어	
삼성당	
Pearson	

SELECT 문

❖ WHERE 조건

WHERE 절에 조건으로 사용할 수 있는 술어

술어	연산자	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')

SELECT 문

❖ WHERE 조건

■ 비교

질의 → 가격이 20,000원 미만인 도서를 검색하시오.

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

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구아는 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

SELECT 문

❖ WHERE 조건

■ 범위

질의 → 가격이 10,000원 이상 20,000 이하인 도서를 검색하시오.

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

bookid	bookname	publisher	price
2	축구아는 여자	나무수	13000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
10	Olympic Champions	Pearson	13000

※ BETWEEN은 논리 연산자인 AND를 사용할 수 있다.

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

bookid	bookname	publisher	price
2	축구아는 여자	나무수	13000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
10	Olympic Champions	Pearson	13000

SELECT 문

❖ WHERE 조건

■ 집합

질의 → 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 검색하시오.

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

bookid	bookname	publisher	price
2	축구하는 여자	나무수	13000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

SELECT 문

❖ WHERE 조건

■ 패턴

질의 → '축구의 역사'를 출간한 출판사를 검색하시오.

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

bookname	publisher
축구의 역사	굿스포츠

질의 → 도서이름에 '축구'가 포함된 출판사를 검색하시오.

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

bookname	publisher
축구의 역사	굿스포츠
축구아는 여자	나무수
축구의 이해	대한미디어

SELECT 문

❖ WHERE 조건

질의 → 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

와일드 문자의 종류

와일드 문자	의미	사용 예
+	문자열을 연결	'골프' + '바이블' : '골프 바이블'
%	0개 이상의 문자열과 일치	'%축구%' : 축구를 포함하는 문자열
[]	1개의 문자와 일치	'[0-5]%' : 0-5 사이 숫자로 시작하는 문자열
[^]	1개의 문자와 불일치	'[^0-5]%' : 0-5 사이 숫자로 시작하지 않는 문자열
_	특정 위치의 1개의 문자와 일치	'_구%' : 두 번째 위치에 '구'가 들어가는 문자열

SELECT 문

❖ WHERE 조건

■ 복합조건

질의 → 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 → 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	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000

SELECT 문

❖ ORDER BY

질의 → 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	축구아는 여자	나무수	13000
1	축구의 역사	굿스포츠	7000
3	축구의 이해	대한미디어	22000
5	피겨 교본	굿스포츠	8000

SELECT 문

❖ ORDER BY

질의 → 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	13000
8	야구를 부탁해	이상미디어	13000
2	축구하는 여자	나무수	13000
7	야구의 추억	이상미디어	20000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000

SELECT 문

❖ ORDER BY

질의 → 도서를 가격의 내림차순으로 검색하시오. 만약 가격이 같다면 출판사의 오름차순으로 검색한다.

```
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	이상미디어	20000
10	Olympic Champions	Pearson	13000
2	축구아는 여자	나무수	13000
8	야구를 부탁해	이상미디어	13000
5	피겨 교본	굿스포츠	8000
9	올림픽 이야기	삼성당	7500
1	축구의 역사	굿스포츠	7000
6	역도 단계별기술	굿스포츠	6000

집계 함수와 GROUP BY

❖ 집계 함수

질의 → 고객이 주문한 도서의 총 판매액을 구하시오.

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

SUM(saleprice)
118000

※ 의미 있는 열 이름을 출력하고 싶으면 속성이름의 별칭을 지칭하는 AS 키워드를 사용하여 열 이름을 부여한다.

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

총매출
118000

집계 함수와 GROUP BY

❖ 집계 함수

질의 → 2번 김연아 고객이 주문한 도서의 총 판매액을 구하시오.

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

총매출
15000

질의 → 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	11800.0000	6000	21000

집계 함수와 GROUP BY

❖ 집계 함수

질의 → 마당서점의 도서 판매 건수를 구하시오.

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

	COUNT(*)
▶	10

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)

집계 함수와 GROUP BY

❖ GROUP BY

질의 → 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
3	3	31000
4	2	33000

orderid	custid	bookid	saleprice	orderdate
1	1	1	6000	2014-07-01
2	1	3	21000	2014-07-03
6	1	2	12000	2014-07-07
3	2	5	8000	2014-07-03
9	2	10	7000	2014-07-09
4	3	6	6000	2014-07-04
8	3	10	12000	2014-07-08
10	3	8	13000	2014-07-10
5	4	7	20000	2014-07-05
7	4	8	13000	2014-07-07

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

GROUP BY 절의 수행

집계 함수와 GROUP BY

❖ GROUP BY

질의 → 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

집계 함수와 GROUP BY

GROUP BY와 HAVING 절의 문법과 주의사항

문법	주의사항
GROUP BY <속성>	<p>GROUP BY로 튜플을 그룹으로 묶은 후 SELECT 절에는 GROUP BY에서 사용한 <속성>과 집계함수만 나올 수 있음</p> <ul style="list-style-type: none">▪ 맞는 예 SELECT custid, SUM(saleprice) FROM Orders GROUP BY custid;• 틀린 예 SELECT bookid, SUM(saleprice) /* SELECT 절에 bookid 속성이 올 수 없다 */ FROM Orders GROUP BY custid;
HAVING <검색조건>	<p>WHERE 절과 HAVING 절이 같이 포함된 SQL 문은 검색조건이 모호해질 수 있음. HAVING 절은 ① 반드시 GROUP BY 절과 같이 작성해야 하고 ② WHERE 절보다 뒤에 나와야 함. 그리고 ③ <검색조건>에는 SUM, AVG, MAX, MIN, COUNT와 같은 집계함수가 와야 함.</p> <ul style="list-style-type: none">• 맞는 예 SELECT custid, COUNT(*) AS 도서수량 FROM Orders WHERE saleprice >= 8000 GROUP BY custid HAVING COUNT(*) >= 2;• 틀린 예 SELECT custid, COUNT(*) AS 도서수량 FROM Orders HAVING COUNT(*) >= 2 /* 순서가 틀렸다 */ WHERE saleprice >= 8000 GROUP BY custid;

두 개 이상 테이블에서 SQL 질의

- Customer 테이블을 Orders 테이블과 조건 없이 연결해보자.
Customer와 Orders 테이블의 합체 결과 튜플의 개수는 고객이 다섯 명이고 주문이 열 개이므로 5×10 해서 50이 된다.

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

custid	name	address	phone	orderid	custid	bookid	saleprice	orderdate
1	박지성	영국 맨체스타	000-5000-0001	1	1	1	6000	2014-07-01
2	김연아	대한민국 서울	000-6000-0001	1	1	1	6000	2014-07-01
3	장미란	대한민국 강원도	000-7000-0001	1	1	1	6000	2014-07-01
4	추신수	미국 클리블랜드	000-8000-0001	1	1	1	6000	2014-07-01
5	박세리	대한민국 대전	NULL	1	1	1	6000	2014-07-01
1	박지성	영국 맨체스타	000-5000-0001	2	1	3	21000	2014-07-03
2	김연아	대한민국 서울	000-6000-0001	2	1	3	21000	2014-07-03
3	장미란	대한민국 강원도	000-7000-0001	2	1	3	21000	2014-07-03
4	추신수	미국 클리블랜드	000-8000-0001	2	1	3	21000	2014-07-03
5	박세리	대한민국 대전	NULL	2	1	3	21000	2014-07-03
1	박지성	영국 맨체스타	000-5000-0001	3	2	5	8000	2014-07-03
2	김연아	대한민국 서울	000-6000-0001	3	2	5	8000	2014-07-03
3	장미란	대한민국 강원도	000-7000-0001	3	2	5	8000	2014-07-03
4	추신수	미국 클리블랜드	000-8000-0001	3	2	5	8000	2014-07-03
5	박세리	대한민국 대전	NULL	3	2	5	8000	2014-07-03
1	박지성	영국 맨체스타	000-5000-0001	4	3	6	6000	2014-07-04
2	김연아	대한민국 서울	000-6000-0001	4	3	6	6000	2014-07-04
3	장미란	대한민국 강원도	000-7000-0001	4	3	6	6000	2014-07-04
4	추신수	미국 클리블랜드	000-8000-0001	4	3	6	6000	2014-07-04
... 중략 ...								
1	박지성	영국 맨체스타	000-5000-0001	10	3	8	13000	2014-07-10
2	김연아	대한민국 서울	000-6000-0001	10	3	8	13000	2014-07-10
3	장미란	대한민국 강원도	000-7000-0001	10	3	8	13000	2014-07-10
4	추신수	미국 클리블랜드	000-8000-0001	10	3	8	13000	2014-07-10
5	박세리	대한민국 대전	NULL	10	3	8	13000	2014-07-10

두 개 이상 테이블에서 SQL 질의

❖ 조인

질의 → 고객과 고객의 주문에 관한 데이터를 모두 보이시오.

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

custid	name	address	phone	orderid	custid	bookid	saleprice	orderdate
1	박지성	영국 맨체스타	000-5000-0001	1	1	1	6000	2014-07-01
1	박지성	영국 맨체스타	000-5000-0001	2	1	3	21000	2014-07-03
2	김연아	대한민국 서울	000-6000-0001	3	2	5	8000	2014-07-03
3	장미란	대한민국 강원도	000-7000-0001	4	3	6	6000	2014-07-04
4	추신수	미국 클리블랜드	000-8000-0001	5	4	7	20000	2014-07-05
1	박지성	영국 맨체스타	000-5000-0001	6	1	2	12000	2014-07-07
4	추신수	미국 클리블랜드	000-8000-0001	7	4	8	13000	2014-07-07
3	장미란	대한민국 강원도	000-7000-0001	8	3	10	12000	2014-07-08
2	김연아	대한민국 서울	000-6000-0001	9	2	10	7000	2014-07-09
3	장미란	대한민국 강원도	000-7000-0001	10	3	8	13000	2014-07-10

두 개 이상 테이블에서 SQL 질의

❖ 조인

질의 → 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	bookid	saleprice	orderdate
1	박지성	영국 맨체스타	000-5000-0001	6	1	2	12000	2014-07-07
1	박지성	영국 맨체스타	000-5000-0001	1	1	1	6000	2014-07-01
1	박지성	영국 맨체스타	000-5000-0001	2	1	3	21000	2014-07-03
2	김연아	대한민국 서울	000-6000-0001	3	2	5	8000	2014-07-03
2	김연아	대한민국 서울	000-6000-0001	9	2	10	7000	2014-07-09
3	장미란	대한민국 강원도	000-7000-0001	4	3	6	6000	2014-07-04
3	장미란	대한민국 강원도	000-7000-0001	8	3	10	12000	2014-07-08
3	장미란	대한민국 강원도	000-7000-0001	10	3	8	13000	2014-07-10
4	추신수	미국 클리블랜드	000-8000-0001	5	4	7	20000	2014-07-05
4	추신수	미국 클리블랜드	000-8000-0001	7	4	8	13000	2014-07-07

두 개 이상 테이블에서 SQL 질의

❖ 조인

질의 → 고객의 이름과 고객이 주문한 도서의 판매가격을 검색하시오.

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

name	saleprice
박지성	6000
박지성	21000
김연아	8000
장미란	6000
추신수	20000
박지성	12000
추신수	13000
장미란	12000
김연아	7000
장미란	13000

두 개 이상 테이블에서 SQL 질의

❖ 조인

질의 → 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

두 개 이상 테이블에서 SQL 질의

❖ 조인

고객의 이름과 고객이 주문한 도서의 이름을 구하시오.

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}

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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

ORDERID	CUSTID	BOOKID	SALEPRICE	ORDERDATE
1	1	1	6000	14/07/01
2	1	3	21000	14/07/03
3	2	5	8000	14/07/03
4	3	6	6000	14/07/04
5	4	7	20000	14/07/05
6	1	2	12000	14/07/07
7	4	8	13000	14/07/07
8	3	10	12000	14/07/08
9	2	10	7000	14/07/09
10	3	8	13000	14/07/10

두 개 이상 테이블에서 SQL 질의

❖ 조인

질의 → 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	축구의 이해
김연아	피겨 교본
장미란	역도 단계별기술
추신수	야구의 추억
박지성	축구하는 여자
추신수	야구를 부탁해
장미란	Olympic Champions
김연아	Olympic Champions
장미란	역도를 부탁해

두 개 이상 테이블에서 SQL 질의

❖ 조인

질의 → 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

두 개 이상 테이블에서 SQL 질의

❖ 조인

■ 외부조인

질의 → 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	6000
박지성	21000
김연아	8000
장미란	6000
추신수	20000
박지성	12000
추신수	13000
장미란	12000
김연아	7000
장미란	13000
박세리	NULL

두 개 이상 테이블에서 SQL 질의

❖ 조인

조인 문법

명령	문법	설명
일반 조인	SELECT <속성들> FROM 테이블1, 테이블2 WHERE <조인조건> AND <검색조건>	SQL 문에서는 주로 동등조인을 사용함. 두 가지 문법 중 하나를 사용할 수 있음.
	SELECT <속성들> FROM 테이블1 INNER JOIN 테이블2 ON <조인조건> WHERE <검색조건>	
외부조인	SELECT <속성들> FROM 테이블1 {LEFT RIGHT FULL [OUTER]} JOIN 테이블2 ON <조인조건> WHERE <검색조건>	외부조인은 FROM 절에 조인 종류를 적고 ON을 이용하여 조인조건을 명시함.

두 개 이상 테이블에서 SQL 질의

❖ 부속질의

질의 → 가장 비싼 도서의 이름을 보이시오.

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

bookname
골프 바이블

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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

①

가장 비싼
도서의 가격은
→ 35,000원

②

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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

두 개 이상 테이블에서 SQL 질의

❖ 부속질의

질의 → 도서를 구매한 적이 있는 고객의 이름을 검색하시오.

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

name
박지성
김연아
장미란
추신수

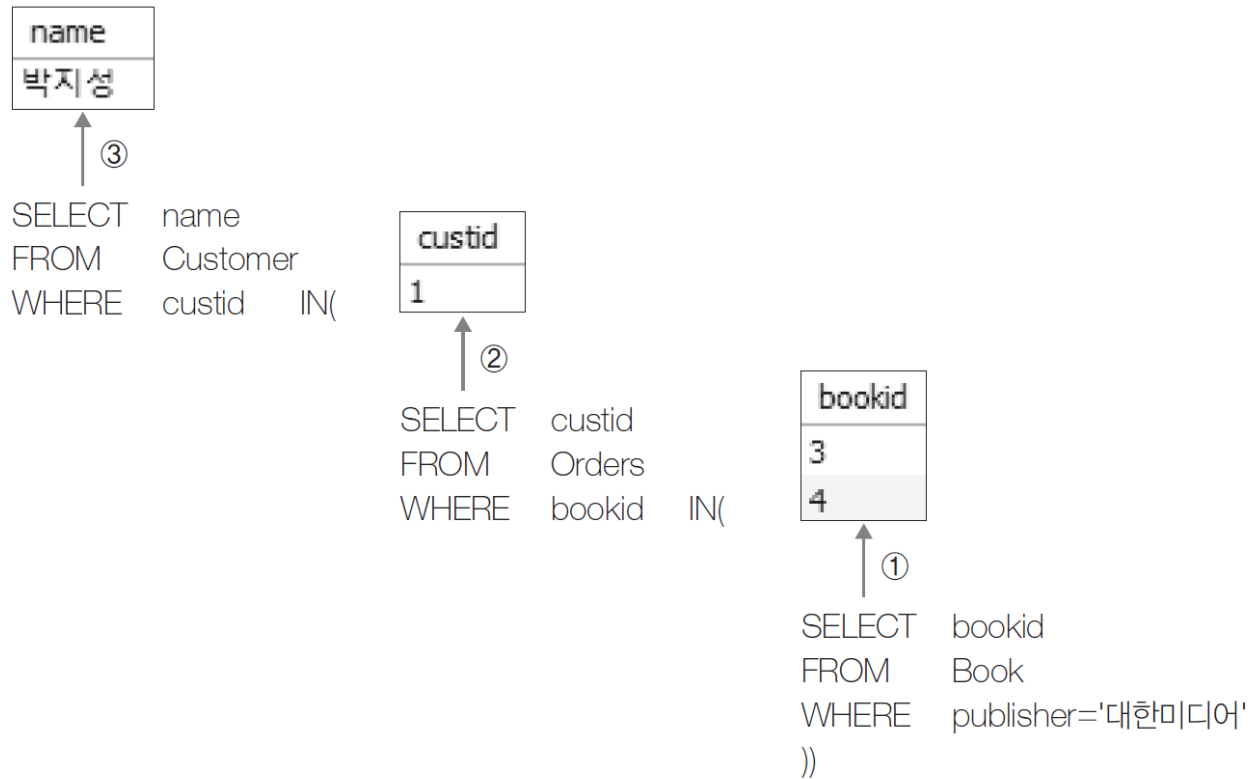
질의 → 대한미디어에서 출판한 도서를 구매한 고객의 이름을 보이시오.

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

name
박지성

두 개 이상 테이블에서 SQL 질의

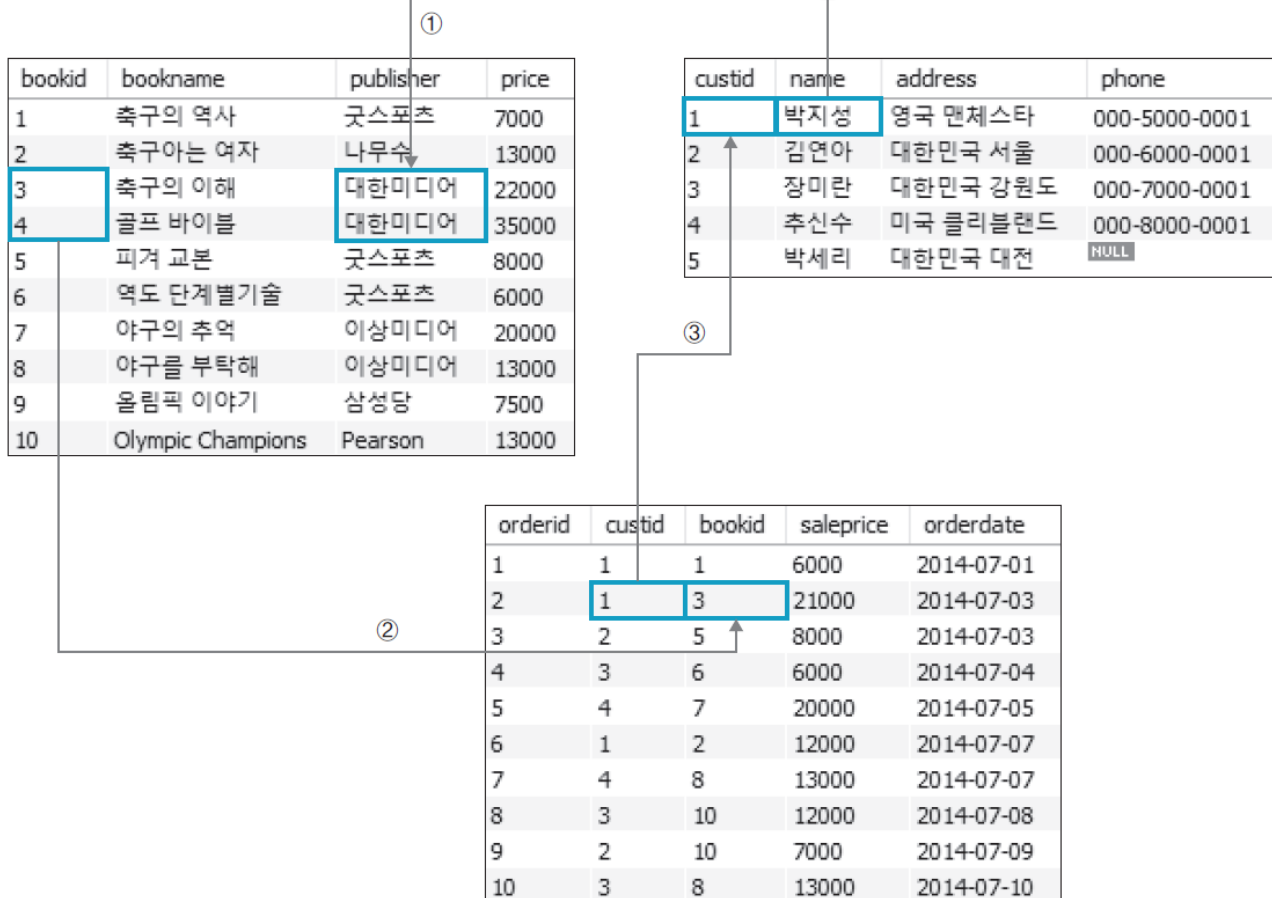
❖ 부속질의



두 개 이상 테이블에서 SQL 질의

❖ 부속질의

'대한미디어'에서 출판한 도서를 구매한 **고객의 이름**을 보이시오.



두 개 이상 테이블에서 SQL 질의

❖ 부속질의

- 상관 부속질의(correlated subquery)는 상위 부속질의의 튜플을 이용하여 하위 부속질을 계산함. 즉 상위 부속질의와 하위 부속질의가 독립적이지 않고 서로 관련을 맺고 있음.

질의 → 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

두 개 이상 테이블에서 SQL 질의

❖ 부속질의

Book 테이블 : b1으로 나타냄

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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

Book 테이블 : b2로 나타냄

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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

b1 테이블의
투플 t에
해당하는
출판사를
b2 테이블로
가져가서,
같은 출판사를
가진 투플들의
price 평균을
구한다.

avg(b2.price)

28500

b1.price > avg(b2.price)

두 개 이상 테이블에서 SQL 질의

❖ 집합 연산

■ 합집합 UNION, 차집합 MINUS, 교집합 INTERSECT

질의 → 대한민국에서 거주하는 고객의 이름과 도서를 주문한 고객의 이름을 보이시오.

```
SELECT  name
FROM    Customer
WHERE   address LIKE '대한민국%'
UNION
SELECT  name
FROM    Customer
WHERE   custid IN (SELECT custid FROM Orders);
```

name
김연아
장미란
박세리
박지성
추신수

{고객 이름} = {대한민국에 거주하는 고객 이름} ∪ {도서를 주문한 고객 이름}

두 개 이상 테이블에서 SQL 질의

■ <여기서 잠깐> MINUS, INTERSECT 연산자

MySQL에는 MINUS, INTERSECT 연산자가 없으므로 다음과 같이 표현한다.

[질의 3-32]에서 **MINUS** 연산을 수행한 "대한민국에서 거주하는 고객의 이름에서 도서를 주문한 고객의 이름 빼고 보이시오." 질의를 NOT IN 연산자를 사용하면 다음과 같다.

```
SELECT    name
FROM      Customer
WHERE address LIKE '대한민국%' AND
        name NOT IN (SELECT name
                      FROM      Customer
                      WHERE      custid IN (SELECT custid FROM Orders));
```

[질의 3-32]에서 **INTERSECT** 연산을 수행한 "대한민국에서 거주하는 고객 중 도서를 주문한 고객의 이름 보이시오." 질의를 IN 연산자를 사용하면 다음과 같다.

```
SELECT    name
FROM      Customer
WHERE address LIKE '대한민국%' AND
        name IN (SELECT    name
                   FROM      Customer
                   WHERE      custid IN (SELECT custid FROM Orders));
```

두 개 이상 테이블에서 SQL 질의

❖ EXISTS

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

질의 → 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
추신수	미국 클리블랜드

두 개 이상 테이블에서 SQL 질의

❖ EXISTS

Customer

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

Orders

orderid	custid	bookid	saleprice	orderdate
1	1	1	6000	2014-07-01
2	1	3	21000	2014-07-03
3	2	5	8000	2014-07-03
4	3	6	6000	2014-07-04
5	4	7	20000	2014-07-05
6	1	2	12000	2014-07-07
7	4	8	13000	2014-07-07
8	3	10	12000	2014-07-08
9	2	10	7000	2014-07-09
10	3	8	13000	2014-07-10

①

②

③

④

⑤

x
없음
false

true

true

true

true

CREATE 문

- 테이블 구성, 속성과 속성에 관한 제약 정의, 기본키 및 외래키를 정의하는 명령
- PRIMARY KEY : 기본키를 정할 때 사용
- FOREIGN KEY : 외래키를 지정할 때 사용
- ON UPDATE와 ON DELETE : 외래키 속성의 수정과 튜플 삭제 시 동작을 나타냄
- CREATE 문의 기본 문법

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

CREATE 문

질의 → 다음과 같은 속성을 가진 NewBook 테이블을 생성하시오. 정수형은 INTEGER를 사용하며 문자형은 가변형 문자타입인 VARCHAR을 사용한다.

- bookid(도서번호)-INTEGER
- bookname(도서이름)-VARCHAR(20)
- publisher(출판사)-VARCHAR(20)
- price(가격)-INTEGER

```
CREATE TABLE      NewBook (  
    bookid          INTEGER,  
    bookname        VARCHAR(20),  
    publisher        VARCHAR(20),  
    price            INTEGER);
```

※ 기본키를 지정하고 싶다면 다음과 같이 생성한다.

CREATE TABLE NewBook (bookid INTEGER, bookname VARCHAR(20), publisher VARCHAR(20), price INTEGER, PRIMARY KEY (bookid);	=	CREATE TABLE NewBook (bookid INTEGER PRIMARY KEY, bookname VARCHAR(20), publisher VARCHAR(20), price INTEGER);
--	---	--

CREATE 문

※ **bookid** 속성이 없어서 두 개의 속성 **bookname**, **publisher**가 기본키가 된다면 괄호를 사용하여 복합키를 지정한다.

```
CREATE TABLE      NewBook (  
    bookname        VARCHAR(20),  
    publisher        VARCHAR(20),  
    price            INTEGER,  
    PRIMARY KEY     (bookname, publisher));
```

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

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

CREATE 문

질의 → 다음과 같은 속성을 가진 **NewCustomer** 테이블을 생성하시오.

- custid(고객번호) - INTEGER, 기본키
- name(이름) - VARCHAR(40)
- address(주소) - VARCHAR(40)
- phone(전화번호) - VARCHAR(30)

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

CREATE 문

질의 → 다음과 같은 속성을 가진 **NewOrders** 테이블을 생성하시오.

- orderid(주문번호) - INTEGER, 기본키
- custid(고객번호) - INTEGER, NOT NULL 제약조건, 외래키(NewCustomer.custid, 연쇄삭제)
- bookid(도서번호) - INTEGER, NOT NULL 제약조건
- saleprice(판매가격) - INTEGER
- orderdate(판매일자) - DATE

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

CREATE 문

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

데이터 타입 종류

데이터 타입	설명	ANSI SQL 표준 타입
INTEGER INT	4바이트 정수형	INTEGER, INT SMALLINT
NUMERIC(m,d) DECIMAL(m,d)	전체자리수 m, 소수점이하 자리수 d를 가진 숫자형	DECIMAL(p, s) NUMERIC[(p,s)]
CHAR(n)	문자형 고정길이, 문자를 저장하고 남은 공간은 공백으로 채운다.	CHARACTER(n) CHAR(n)
VARCHAR(n)	문자형 가변길이	CHARACTER VARYING(n) CHAR VARYING(n)
DATE	날짜형, 연도, 월, 날, 시간을 저장한다.	

ALTER 문

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

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

ALTER 문

질의 → NewBook 테이블에 VARCHAR(13)의 자료형을 가진 isbn 속성을 추가하시오.

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

질의 → NewBook 테이블의 isbn 속성의 데이터 타입을 INTEGER형으로 변경하시오.

```
ALTER TABLE NewBook MODIFY isbn INTEGER;
```

질의 → NewBook 테이블의 isbn 속성을 삭제하시오.

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

질의 → NewBook 테이블의 bookid 속성에 NOT NULL 제약조건을 적용하시오.

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

질의 → NewBook 테이블의 bookid 속성을 기본키로 변경하시오.

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

DROP 문

- DROP 문은 테이블을 삭제하는 명령
- DROP 문은 테이블의 구조와 데이터를 모두 삭제하므로 사용에 주의해야 함 (데이터만 삭제하려면 DELETE 문을 사용함).
- DROP문의 기본 문법

```
DROP TABLE 테이블이름
```

질의 → NewBook 테이블을 삭제하시오.

```
DROP TABLE NewBook;
```

질의 → NewCustomer 테이블을 삭제하시오. 만약 삭제가 거절된다면 원인을 파악하고 관련된 테이블을 같이 삭제하시오(NewOrders 테이블이 NewCustomer를 참조하고 있음).

```
DROP TABLE NewCustomer;
```

INSERT 문

- INSERT 문은 테이블에 새로운 튜플을 삽입하는 명령임.
- INSERT 문의 기본 문법

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

질의 → 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
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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
11	스포츠 의학	한솔의학...	90000

INSERT 문

질의 → 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
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
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
11	스포츠 의학	한솔의학...	90000
12	스포츠 의학	한솔의학...	90000
13	스포츠 의학	한솔의학...	90000
14	스포츠 의학	한솔의학...	NULL

INSERT 문

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

질의 → 수입도서 목록(Imported_book)을 Book 테이블에 모두 삽입하시오.
(Imported_book 테이블은 스크립트 Book 테이블과 같이 이미 만들어져 있음)

```
INSERT INTO Book(bookid, bookname, price, publisher)
SELECT bookid, bookname, price, publisher
FROM Imported_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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
11	스포츠 의학	한솔의학...	90000
12	스포츠 의학	한솔의학...	90000
13	스포츠 의학	한솔의학...	90000
14	스포츠 의학	한솔의학...	NULL
21	Zen Golf	Pearson	12000
22	Soccer Skills	Human Kin...	15000

UPDATE 문

- UPDATE 문은 특정 속성 값을 수정하는 명령이다.
- UPDATE 문의 기본 문법

```
UPDATE 테이블이름  
SET    속성이름1=값1[, 속성이름2=값2, ...]  
[WHERE <검색조건>];
```

질의 → Customer 테이블에서 고객번호가 5인 고객의 주소를 '대한민국 부산'으로 변경하시오.

```
SET SQL_SAFE_UPDATES=0; /* Safe Updates 옵션 미 해제 시 실행 */
```

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

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

UPDATE 문

질의 → Book 테이블에서 14번 '스포츠 의학'의 출판사를 imported_book 테이블의 21번 책의 출판사와 동일하게 변경하시오.

```
UPDATE Book
SET publisher = (SELECT publisher
                  FROM imported_book
                  WHERE bookid = '21')
WHERE bookid = '14' ;
```

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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
11	스포츠 의학	한솔의학...	90000
12	스포츠 의학	한솔의학...	90000
13	스포츠 의학	한솔의학...	90000
14	스포츠 의학	Pearson	NULL
21	Zen Golf	Pearson	12000
22	Soccer Skills	Human Kin...	15000

DELETE 문

- DELETE 문은 테이블에 있는 기존 튜플을 삭제하는 명령
- DELETE 문의 기본 문법

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

DELETE 문

질의 → Book 테이블에서 도서번호가 11인 도서를 삭제하시오.

```
DELETE FROM Book  
WHERE bookid = '11';
```

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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
12	스포츠 의학	한솔의학...	90000
13	스포츠 의학	한솔의학...	90000
14	스포츠 의학	Pearson	NULL
21	Zen Golf	Pearson	12000
22	Soccer Skills	Human Kin...	15000

질의 → 모든 고객을 삭제하시오.

```
DELETE FROM Customer;
```



27 14:36:08 DELETEFROMCustomer

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('madang`.`orders`,... 0.125 sec