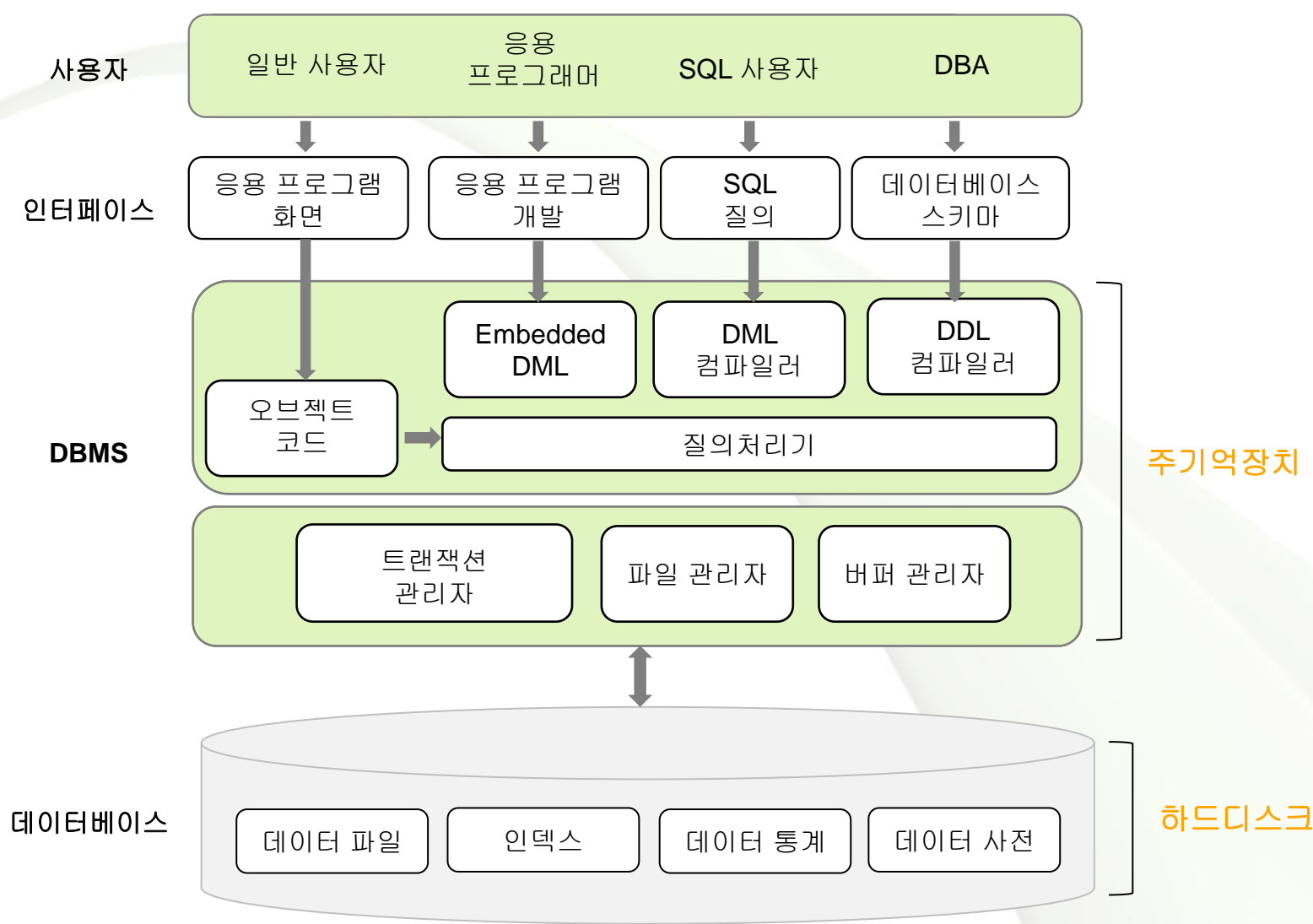


The background features a series of flowing, wavy green lines that create a sense of movement and depth. These lines are layered, with some appearing more prominent than others, and they curve across the frame. A solid dark green horizontal bar is positioned at the very bottom of the image.

Introduction to DBMS

데이터베이스 시스템의 구성



데이터베이스 언어

▪ SQL

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

```
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 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

데이터베이스 사용자

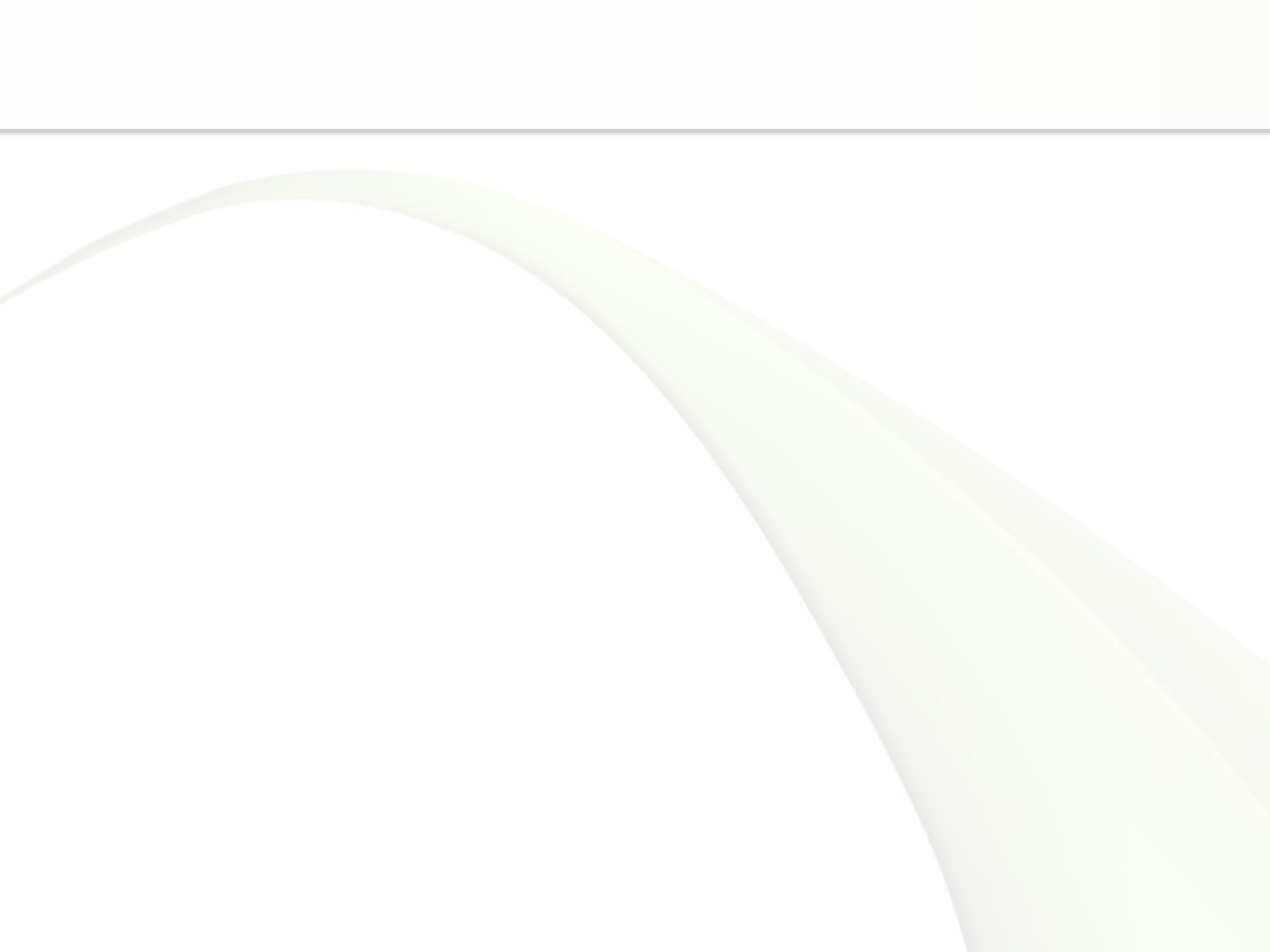
데이터베이스 사용자 별로 갖추어야 할 지식 수준(× : 없음, ○ : 보통, ◎ : 높음)

	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

■ 릴레이션이란?

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

데이터와 테이블(릴레이션)

도서번호 = {1, 2, 3, 4, 5}

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

출판사 = {굿스포츠, 나무수, 대한미디어}

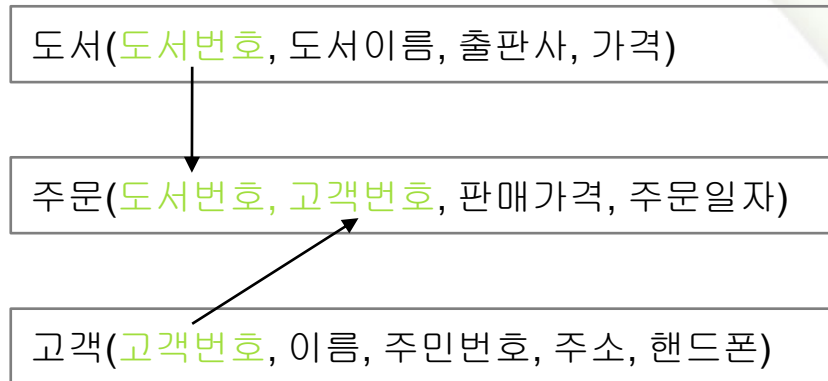
가격 = {7000, 13000, 22000, 35000, 8000}

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 맺고 있다.

릴레이션

▪ 관계(relationship)

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



릴레이션 간의 관계

릴레이션 스키마와 인스턴스

속성(애틀리뷰트),
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)

릴레이션 인스턴스

■ 인스턴스 요소

- » 튜플(tuple) : 릴레이션의 행 → 튜플이 가지는 속성의 개수는 릴레이션 스키마의 차수와 동일하고, 릴레이션 내의 모든 튜플들은 서로 중복되지 않아야 함
- » 카디널리티(cardinality) : 튜플의 수

릴레이션 구조와 관련된 용어

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

릴레이선의 특징

- 속성은 단일 값을 가진다
각 속성의 값은 도메인에 정의된 값만을 가지며 그 값은 모두 단일 값이어야 함.
- 속성은 서로 다른 이름을 가진다
속성은 한 릴레이선에서 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 릴레이선 스키마는 같음.
예) 릴레이선 스키마에서 (이름, 주소) 순으로 속성을 표시하거나 (주소, 이름) 순으로 표시하여도 상관 없음.

릴레이션의 특징

- 릴레이션 내의 중복된 튜플은 허용하지 않는다

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

- 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.

릴레이션의 특징

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
5	피겨 교본	굿스포츠	8000
6	피겨 교본, 피겨 기초	굿스포츠	8000

동일한 튜플이 중복되면 안 됨

속성의 값은 단일 값이어야 함

릴레이션의 특징에 위배된 경우

키

- 특정 튜플을 식별할 때 사용하는 속성 혹은 속성의 집합임
- 릴레이션은 중복된 튜플을 허용하지 않기 때문에 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 없음. 고객번호가 1인 박지성 고객은 세 번의 주문 기록이 있으므로 튜플을 유일하게 식별할 수 없음
 - » 도서번호 : 도서번호가 2인 '축구하는 여자'는 두 번의 주문 기록이 있으므로 튜플을 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 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
- 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

대체키

- 대체키(alternate key)는 기본키로 선정되지 않은 후보키를 말함
- 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

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	축구아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000

기본키

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

참조

기본키

외래키

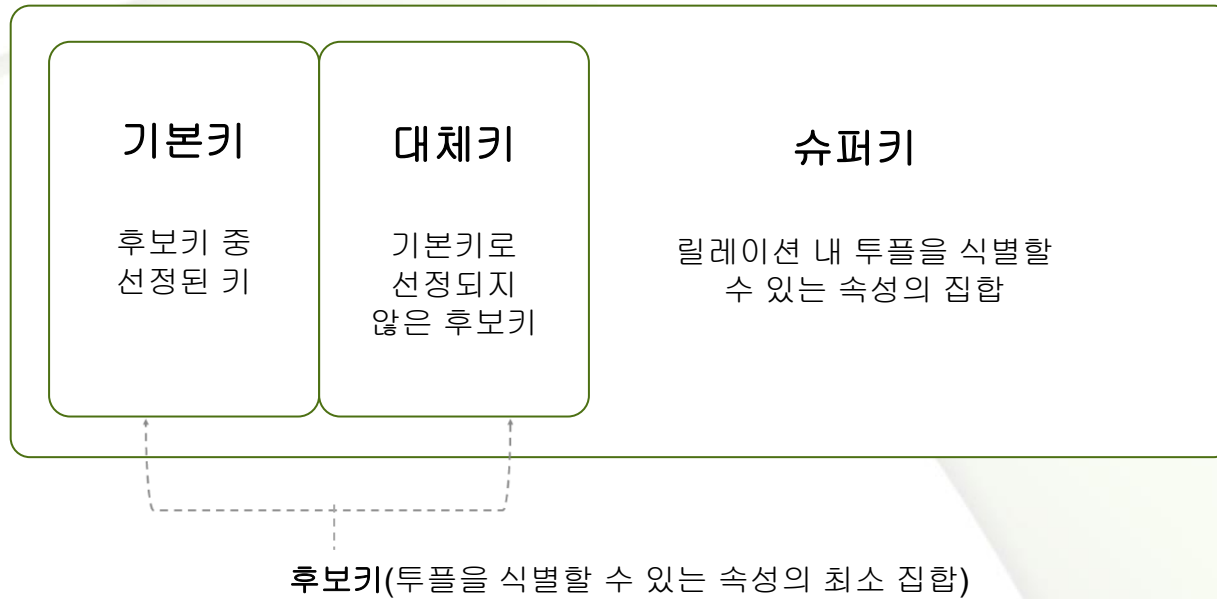
- 외래키 사용 시 참조하는 릴레이션과 참조되는 릴레이션이 꼭 다른 릴레이션일 필요는 없음. 즉 자기 자신의 기본키를 참조할 수도 있음



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

멘토 릴레이션

키 - 내용 요약



키의 포함 관계

무결성 제약조건

- 데이터 무결성(integrity)은 데이터베이스에 저장된 데이터의 일관성과 정확성을 지키는 것을 말함

- 도메인 무결성 제약조건

도메인 제약(domain constraint)이라고도 하며, 릴레이션 내의 튜플들이 각 속성의 도메인에 지정된 값만을 가져야 한다는 조건임. 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

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 여부	허용	불가	허용
릴레이션 내 제약조건의 개수	속성의 개수와 동일	1개	0~여러 개
기타	<ul style="list-style-type: none"> • 튜플 삽입, 수정 시 제약 사항 우선 확인 	<ul style="list-style-type: none"> • 튜플 삽입/수정 시 제약 사항 우선 확인 	<ul style="list-style-type: none"> • 튜플 삽입/수정 시 제약사항 우선 확인 • 부모 릴레이션의 튜플 수정/삭제 시 제약사항 우선 확인

개체 무결성 제약조건

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

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

학생 릴레이션

(501, 남슬찬, 1001)



삽입 거부

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

개체 무결성 제약조건을 수행 예(기본키 충돌 및 NULL 값 삽입)

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

학생관리 데이터베이스

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

- ① 즉시 작업을 중지
- ② 자식 릴레이션의 관련 튜플을 삭제
- ③ 초기에 설정된 다른 어떤 값으로 변경
- ④ NULL 값으로 설정

- 수정

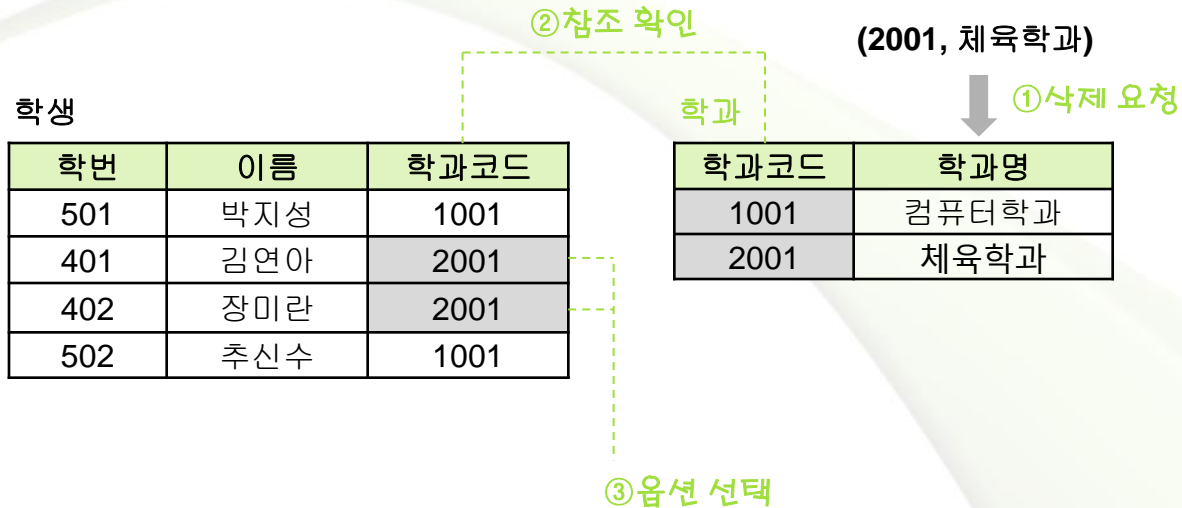
- » 삭제와 삽입 명령이 연속해서 수행됨
- » 부모 릴레이션의 수정이 일어날 경우 삭제 옵션에 따라 처리된 후 문제가 없으면 다시 삽입 제약조건에 따라 처리됨

참조 무결성 제약조건

참조 무결성 제약조건의 옵션(부모 릴레이션에서 튜플을 삭제할 경우)

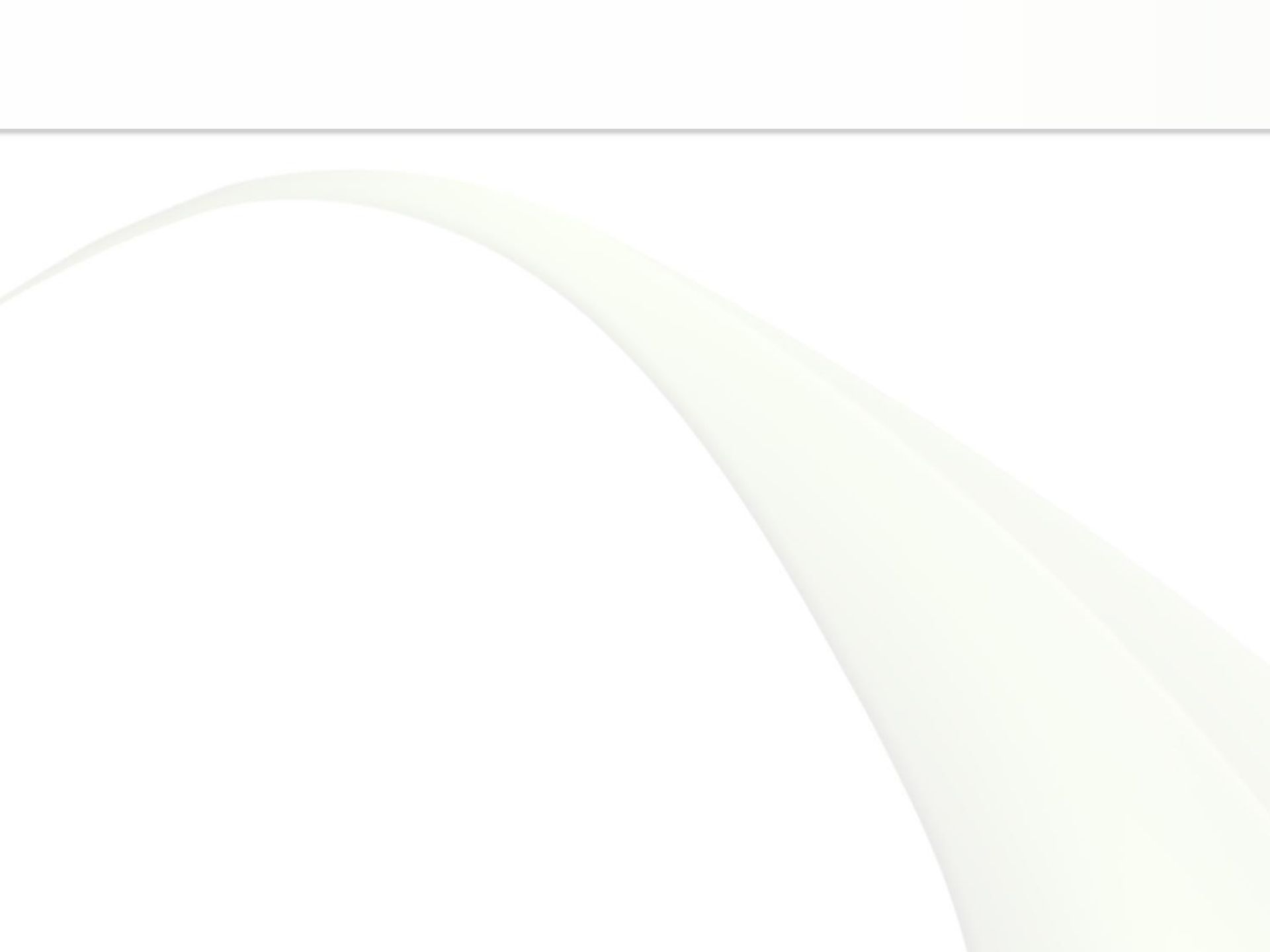
명령어	의미	예
RESTRICTED	자식 릴레이션에서 참조하고 있을 경우 부모 릴레이션의 삭제 작업을 거부함	학과 릴레이션의 튜플 삭제 거부
CASCADE	자식 릴레이션의 관련 튜플을 같이 삭제 처리함	학생 릴레이션의 관련 튜플을 삭제
DEFAULT	자식 릴레이션의 관련 튜플을 미리 설정해둔 값으로 변경함	학생 릴레이션의 학과가 다른 학과로 자동 배정
NULL	자식 릴레이션의 관련 튜플을 NULL 값으로 설정함(NULL 값을 허가한 경우)	학과 릴레이션의 학과가 NULL 값으로 변경

참조 무결성 제약조건

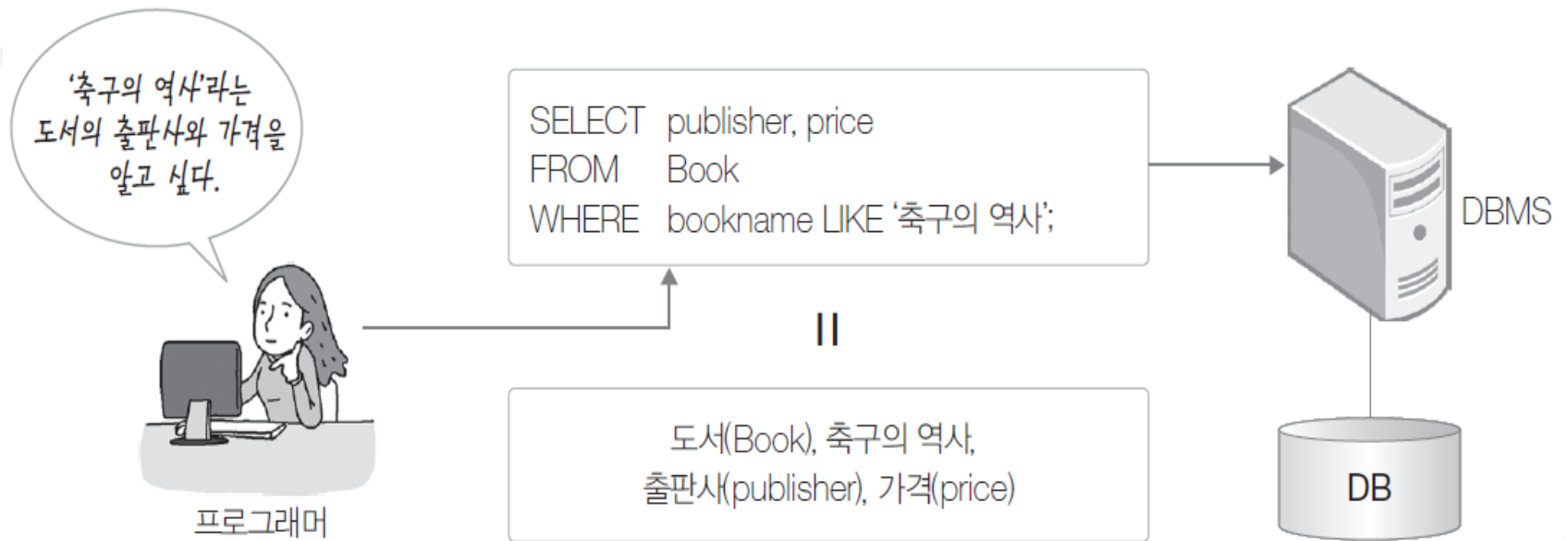


- ① RESTRICTED : 요청한 삭제 작업중지(에러 처리)
- ② CASCADE : 학생 릴레이션의 해당 튜플을 같이 연쇄적으로 삭제(CASCADE)
- ③ 기본값으로 변경(미리 설정한 값, DEFAULT)
- ④ NULL 값으로 설정

참조 무결성 제약조건에서 부모 릴레이션의 튜플을 삭제할 경우



SQL 개요



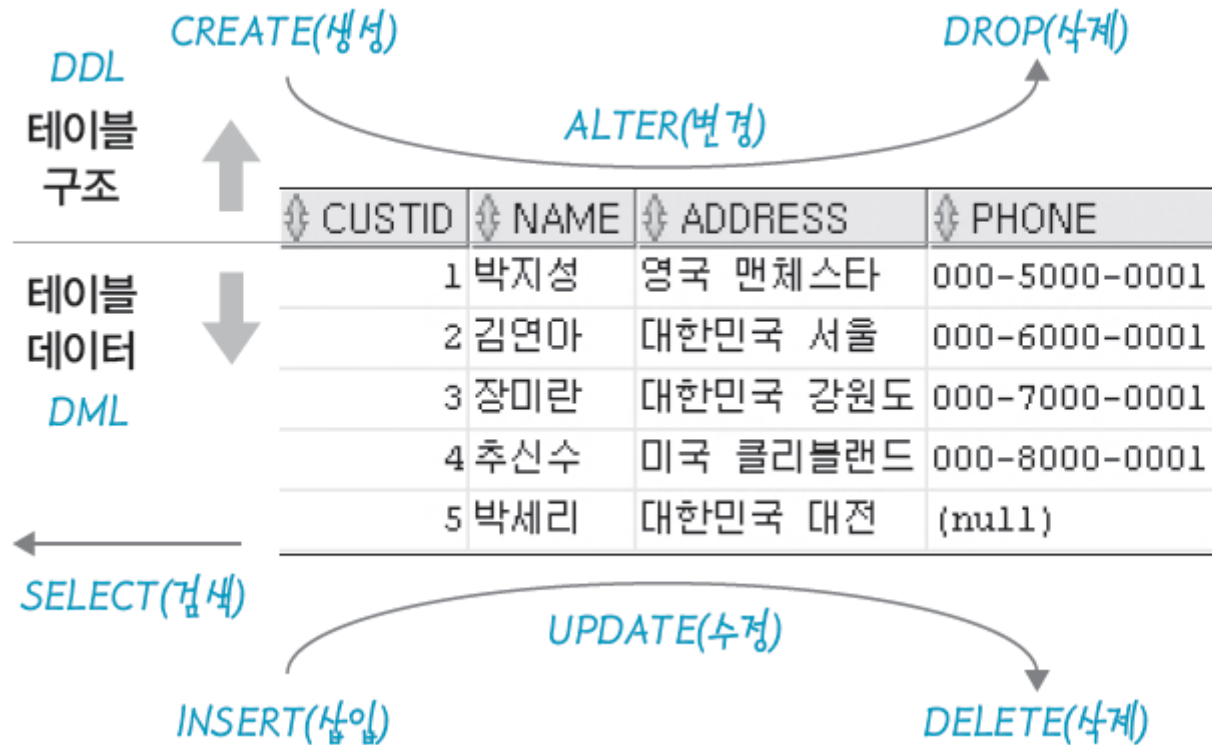
SQL 개요

■ 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.

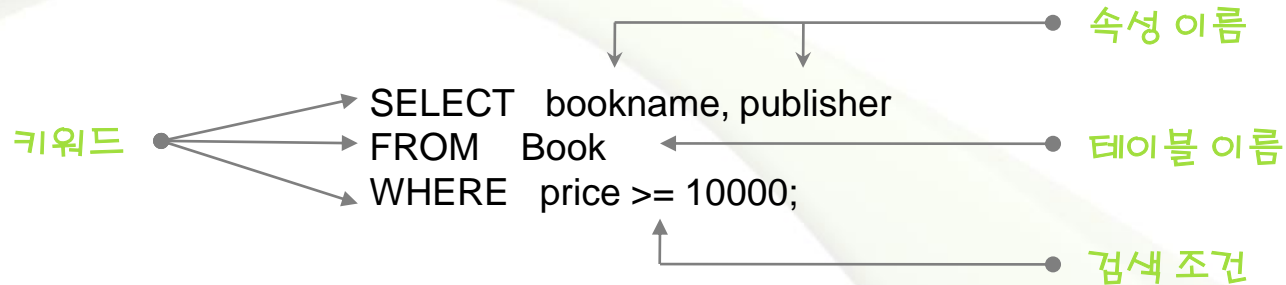
SQL 개요

데이터 정의어와 데이터 조작용의 주요 명령어



데이터 조작용어 - 검색

■ SELECT 문의 구성 요소



■ 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.

데이터 정의어

- CREATE 문
- ALTER 문
- DROP 문

CREATE 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 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

- 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.

CREATE 문

■ 속성의 데이터 타입 종류

데이터 타입	설명	비슷한 타입
NUMBER(p, s)	실수형 p자리 정수, s자리 소수 부분. P와 s를 생략하여 NUMBER라고 쓰면 NUMBER(8, 2)로 저장됨.	DECIMAL(p, s) NUMBER[(p,s)] INTEGER, INT SMALLINT
CHAR(n)	문자형 고정길이. 문자를 저장하고 남은 공간은 공 백으로 채움.	CHARACTER(n) CHAR(n)
VARCHAR2(n)	문자형 가변길이. 4000바이트까지 저장됨.	CHARACTER(n) VARYING(n) CHAR(n) VARYING(n)
DATE	날짜형, 연도/월/날/시간을 지정함.	

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.
- 그리고 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]

DROP 문

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

```
DROP TABLE 테이블이름
```

INSERT 문

- INSERT 문은 테이블에 새로운 튜플을 삽입하는 명령임.

- INSERT 문의 기본 문법

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

UPDATE 문

- UPDATE 문은 특정 속성 값을 수정하는 명령이다.

- UPDATE 문의 기본 문법

UPDATE 테이블이름

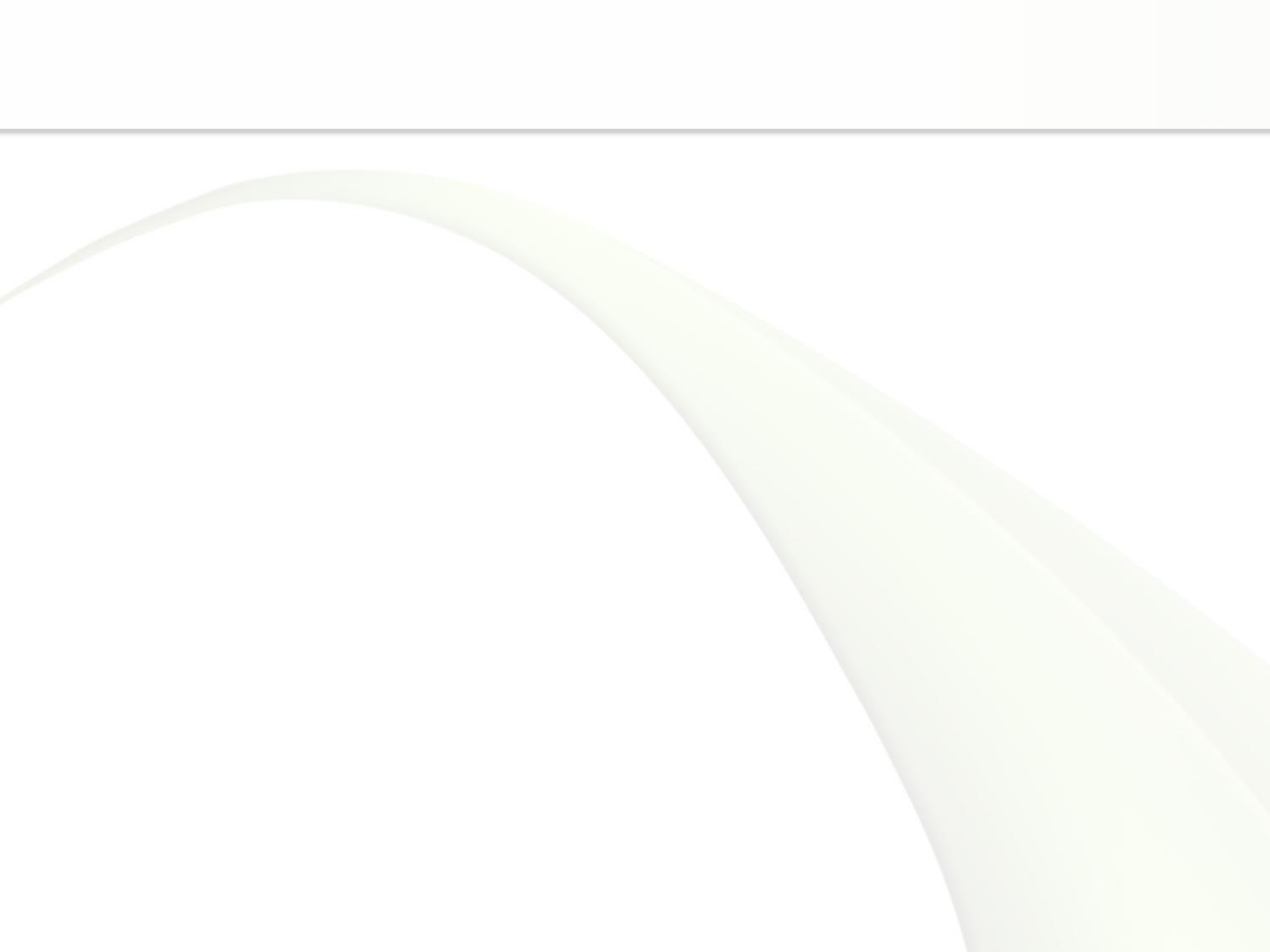
SET 속성이름1=값1[, 속성이름2=값2, ...]

[WHERE <검색조건>];

DELETE 문

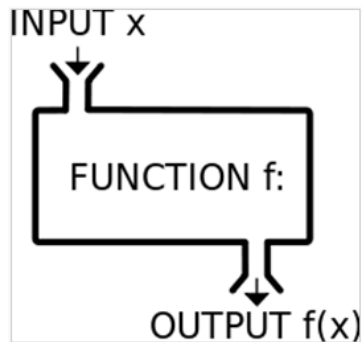
- DELETE 문은 테이블에 있는 기존 튜플을 삭제하는 명령임.
- DELETE 문의 기본 문법

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



내장 함수

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



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

SQL 내장 함수

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

오라클에서 제공하는 주요 내장 함수

분류	설명	종류
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	ASCIISTR, 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
	NULL 관련 함수	COALESCE, NULLIF, NVL
집계 함수	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

함수	설명	예
ABS(숫자)	절대값 계산	$ABS(-4.5)=4.5$
CEILING(숫자)	숫자보다 크거나 같은 최소의 정수	$CEIL(4.1)=5$
FLOOR(숫자)	숫자보다 작거나 같은 최소의 정수	$FLOOR(4.1)=4$
ROUND(숫자, m)	숫자의 반올림, m은 반올림 기준 자릿수	$ROUND(5.36, 1)=5.40$
LOG(숫자)	숫자의 자연로그 값을 반환	$LOG(10)=2.30259$
POWER(숫자, n)	숫자 n제곱 값을 계산	$POWER(2, 3)=8$
SQRT(숫자)	숫자의 제곱근 값을 계산(숫자는 양수)	$SQRT(9.0)=3.0$
SIGN(숫자)	숫자가 음수면 -1, 0이면 0, 양수면 1	$SIGN(3.45)=1$

문자 함수

반환 구분	함수	설명
문자값 반환 함수 s : 문자열 c : 문자 n : 정수 k : 정수	CHR(k)	정수 아스키 코드를 문자로 반환 CHR(68) = 'D'
	CONCAT(s1,s2)	두 문자열을 연결 CONCAT('마당', ' 서점') = '마당 서점'
	INITCAP(s)	문자열의 첫 번째 알파벳을 대문자로 변환 INITCAP('the soap') = 'The Soap'
	LOWER(s)	대상 문자열을 모두 소문자로 변환 LOWER('MR. SCOTT') = 'mr. scott'
	LPAD(s,n,c)	대상 문자열의 왼쪽부터 지정한 자리 수까지 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 자리 수까지 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('ABCDEFGH', 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'
숫자값 반환 함수	ASCII(c)	대상 알파벳 문자의 아스키 코드 값을 반환 (예) ASCII('D') = 68
	INSTR(s1,s2,n,k)	문자열 중 n번째 문자부터 시작하여 찾고자 하는 문자열 s2가 k 번째 나타나는 문자. 열 위치 반환, 예제에서 3번째부터 OR가 2번째 나타나는 자리 수 (예) INSTR('CORPORATE FLOOR', 'OR', 3, 2) = 14
	LENGTH(s)	대상 문자열의 글자 수를 반환 (예)LENGTH('CANDIDE') = 7

날짜 · 시간 함수

함수	반환형	설명
TO_DATE(char, datetime)	DATE	문자형(CHAR) 데이터를 날짜형(DATE)으로 반환 TO_DATE('2014-02-14', 'yyyy-mm-dd') = 2014-02-14
TO_CHAR(date, datetime)	CHAR	날짜형(DATE) 데이터를 문자열(VARCHAR2)로 반환 TO_CHAR(TO_DATE('2014-02-14', 'yyyy-mm-dd'), 'yyyymmdd') = '20140214'
ADD_MONTHS(date, 숫자)	DATE	date 형의 날짜에서 지정한 달만큼 더함(1 : 다음달, -1 : 이전달) ADD_MONTHS(TO_DATE('2014-02-14', 'yyyy-mm-dd'), 12) = 2015-02-14
LAST_DAY(date)	DATE	date 형의 날짜에서 달의 마지막 날을 반환 LAST_DAY(TO_DATE('2014-02-14', 'yyyy-mm-dd')) = 2014-02-28
SYSDATE	DATE	DBMS 시스템상의 오늘 날짜를 반환하는 인자없는 함수 SYSDATE = 14/04/20

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.

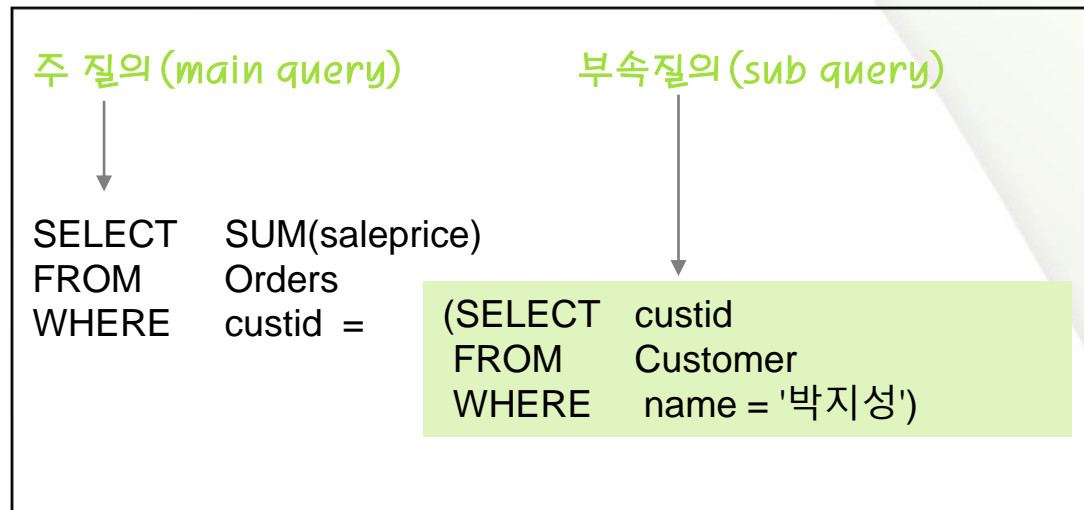
ROWNUM

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

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.
- 주질의(main query, 외부질의)와 부속질의(sub query, 내부질의)로 구성됨.



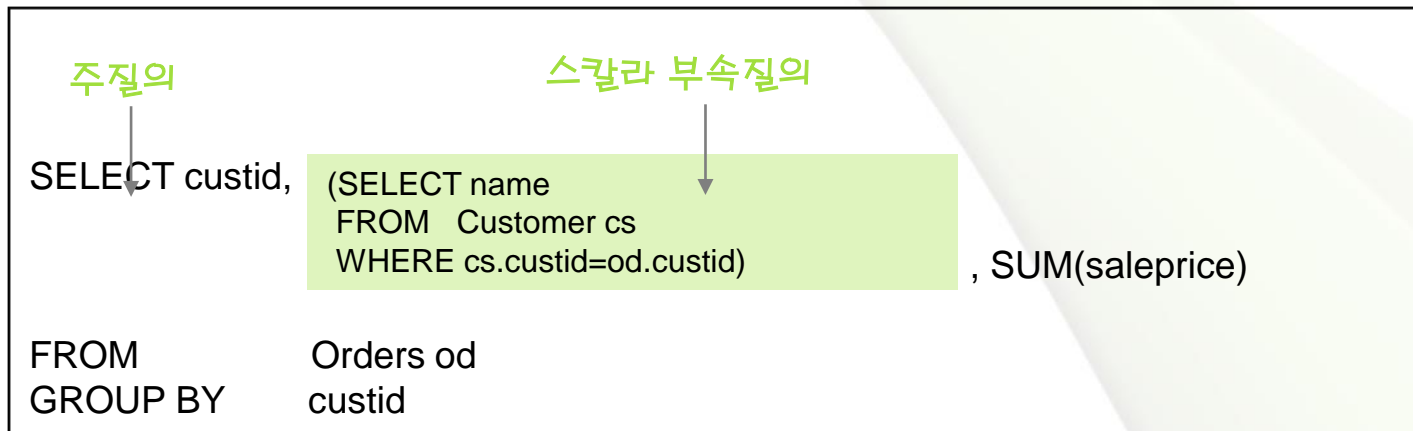
부속질의

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

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.



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.

인라인 뷰 - FROM 부속질의

주질의



```
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 ;
```

← 인라인 뷰

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.

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

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
 - 보안성 : 각 사용자별로 필요한 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
 - 독립성 제공 : 미리 정의된 뷰를 일반 테이블처럼 사용할 수 있기 때문에 편리함. 또 사용자가 필요한 정보만 요구에 맞게 가공하여 뷰로 만들어 쓸 수 있음.
→ 원본테이블이 구조가 변하여도 응용에 영향을 주지않도록하는 논리적 독립성 제공
- (뷰의 특징)
1. 원본 데이터 값에 따라 같이 변함
 2. 독립적인 인덱스 생성이 어려움
 3. 삽입, 삭제, 갱신 연산에 많은 제약이 따름

뷰 Vorders

ORDERID	CUSTID	NAME	BOOKID	BOOKNAME	SALEPRICE	ORDERDATE
1	1	박지성	1	축구의 역사	6000	14/07/01
6	1	박지성	2	축구하는 여자	12000	14/07/07
2	1	박지성	3	축구의 이해	21000	14/07/03
3	2	김연아	5	피겨 교본	8000	14/07/03

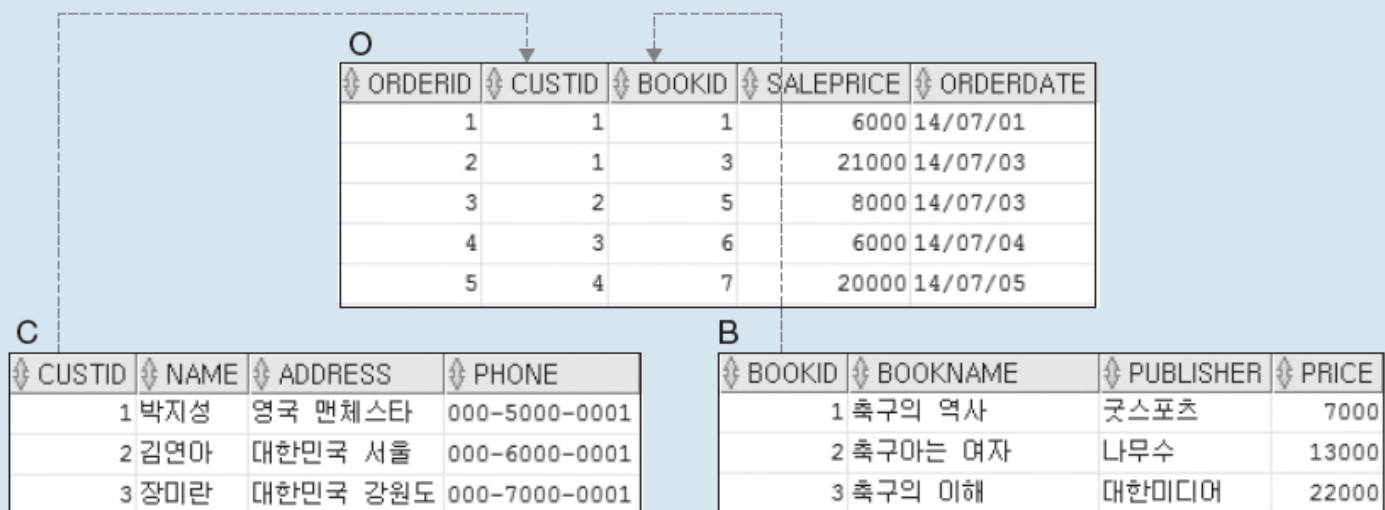
뷰 생성문

```

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

```

베이스 릴레이션 Customer, Orders, Book



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%';
```

뷰의 수정

■ 기본 문법

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

뷰의 삭제

■ 기본 문법

```
DROP VIEW 뷰이름 [ ,...n ];
```


인덱스

- 인덱스(index, 색인)란 문서의 색인이나 사전과 같이 데이터를 쉽고 빠르게 찾을 수 있도록 만든 데이터 구조임.

■ 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.

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]} ...])[;]
```

■ 인덱스 삭제

```
DROP INDEX 인덱스이름
```