

데이터베이스관리

7주차

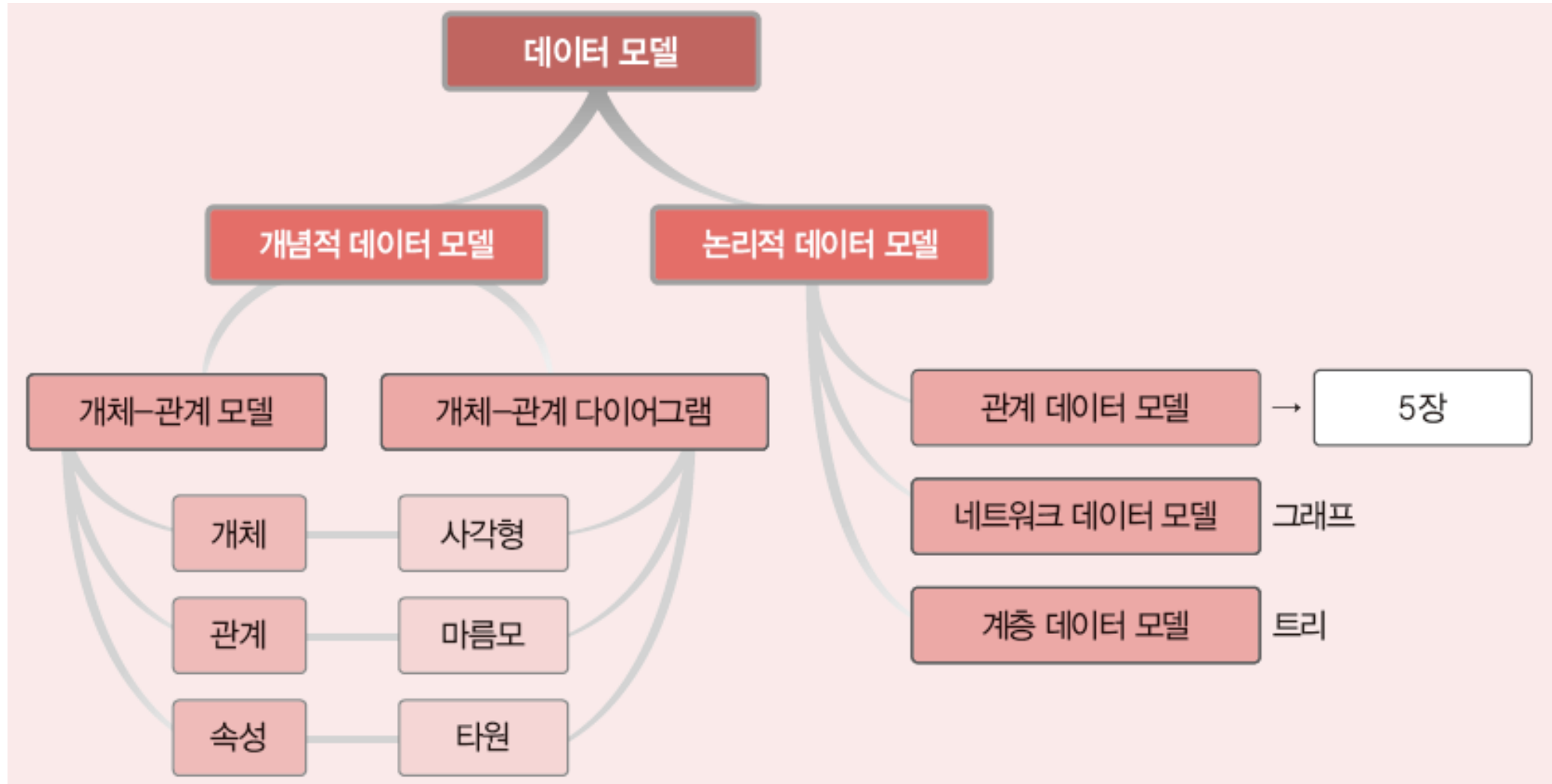
담당교수: 김희숙
(jasmin11@hanmail.net)

데이터 모델링

7주차 7-01

담당교수: 김희숙
(jasmin11@hanmail.net)

[요약] 데이터 모델링



[요약] 데이터 모델링



❖ 데이터 모델링: 현실 세계에 존재하는 데이터를 컴퓨터 세계의 데이터베이스로 옮기는 변환 과정

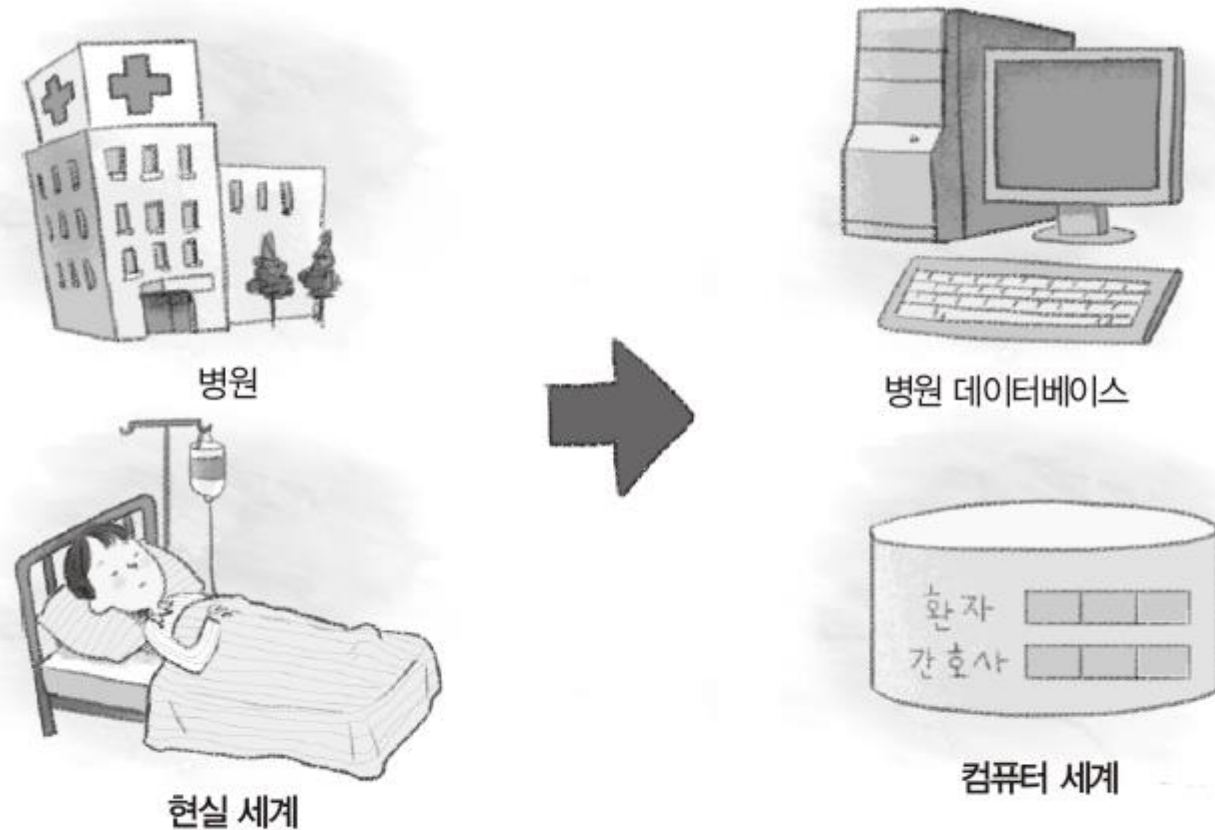


그림 4-1 현실 세계와 컴퓨터 세계

[요약] 데이터 모델링

❖ 데이터 모델링: 추상화 과정

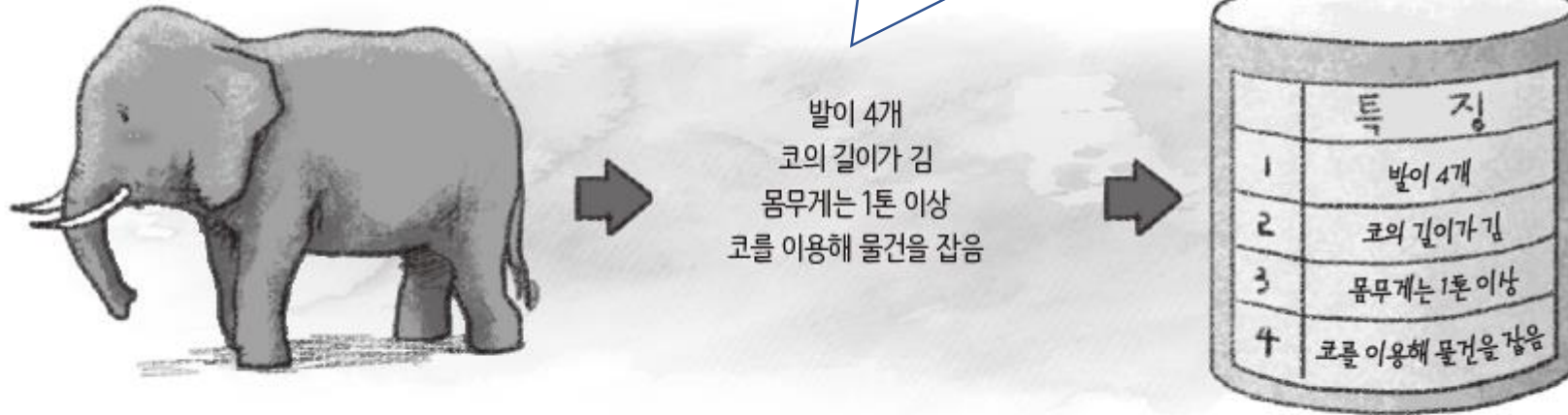


그림 4-2 코끼리의 데이터 모델링 예

[요약] 데이터베이스 설계

❖ 데이터베이스 설계

1. 개념적 데이터 모델링(conceptual modeling)

현실 세계의 중요 데이터를 추출하여 개념 세계로 옮기는 작업

→ 개체-관계 모델

2. 논리적 데이터 모델링(logical modeling)

개념 세계의 데이터를 데이터베이스에 저장하는 구조로 표현하는 작업

→ 관계 데이터 모델



그림 4-3 코끼리의 2단계 데이터 모델링 예

[요약] 데이터 모델 구성



❖ 데이터 모델: 데이터 구조(data structure), 연산(operation), 제약조건(constraint)

-- 데이터 모델 <S, O, C>

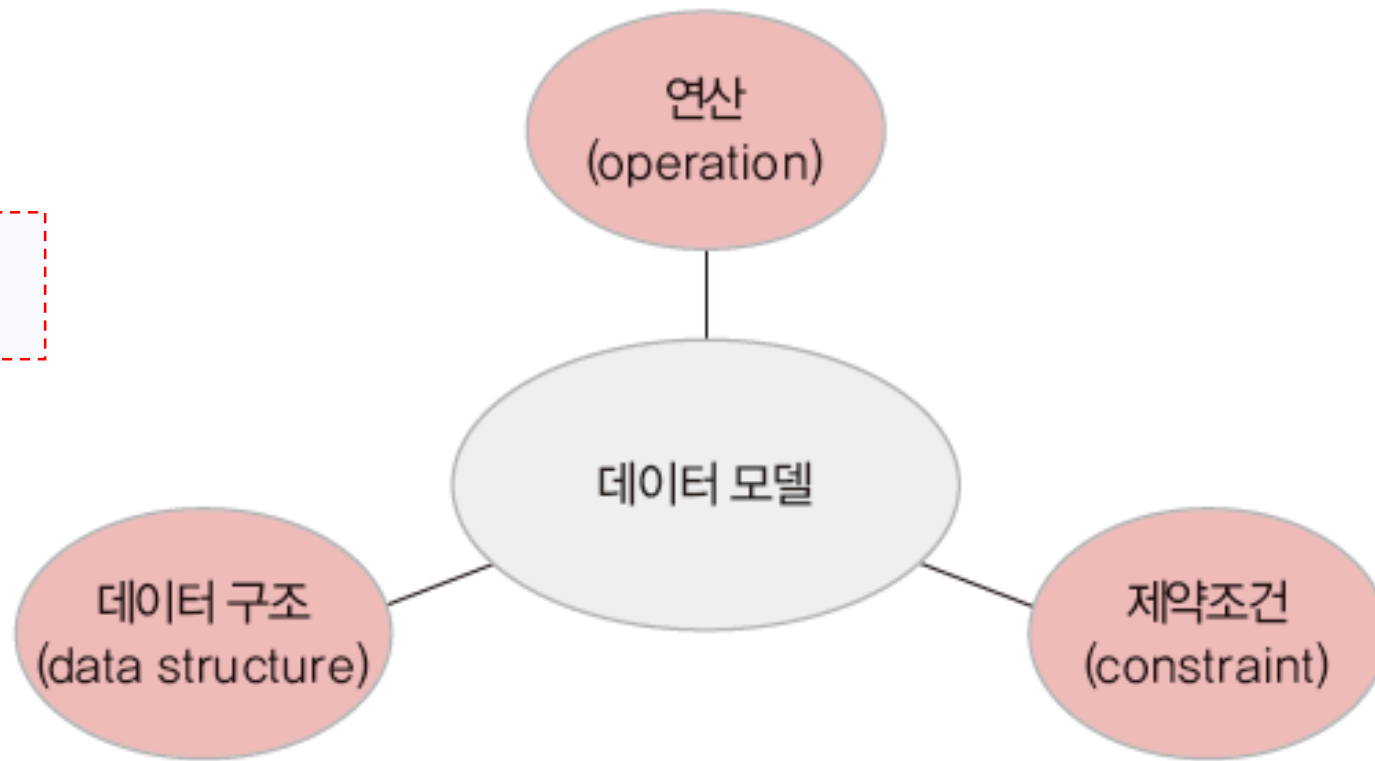


그림 4-4 데이터 모델의 구성



[요약] 개체-관계 모델

❖ 개체-관계 모델(E-R model; Entity-Relationship model)

피터 첸(**Peter Chen**) 제안

개체와 개체 간의 관계를 이용해 현실 세계를 개념적 구조로 표현

❖ 개체-관계 다이어그램(E-R diagram)

개체-관계 모델을 이용해 현실 세계를 개념적으로 모델링한 결과물을 그림으로 표현한 것

-- ER 모델: **개체, 속성, 관계**

[요약] 개체(엔티티) entity



❖ 개체(entity)

1. 현실 세계에서 조직을 운영하는 데 꼭 필요한 사람이나 사물과 같이 구별되는 모든 것
2. 저장할 가치가 있는 중요 데이터를 가지고 있는 사람이나 사물, 개념, 사건 등
3. 다른 개체와 구별되는 이름을 가지고 있고, 각 개체만의 고유한 특성이나 상태 즉 속성을 하나 이상 가지고 있음

예) 서점 개체 : 고객, 책

예) 학교 개체 : 학과, 과목

고객

그림 4-6 개체의 E-R 다이어그램 표현 예 : 고객 개체

[요약] 개체 타입과 개체 인스턴스

❖ 개체 타입:

개체를 고유 이름과 속성으로 정의한 것

❖ 개체 인스턴스:

개체를 구성하고 있는

속성이 실제 값을 가져 실체화된 개체

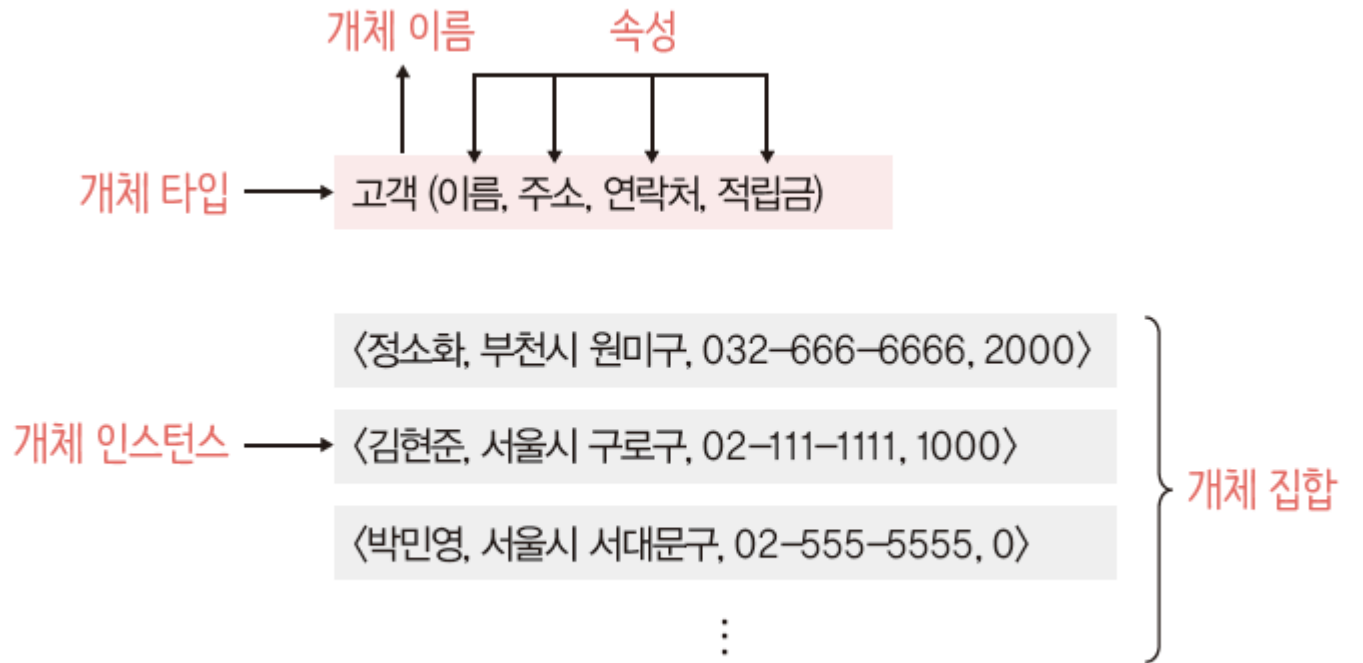


그림 4-5 개체 타입과 개체 인스턴스의 예 : 고객 개체 타입과 고객 개체 인스턴스



[요약] 속성(애트리뷰트) attribute

❖ 속성(attribute)

1. 개체나 관계가 가지고 있는 고유의 특성
2. 의미 있는 데이터의 가장 작은 논리적 단위
3. 파일 구조의 필드(field)와 대응

속성: 타원으로 표시

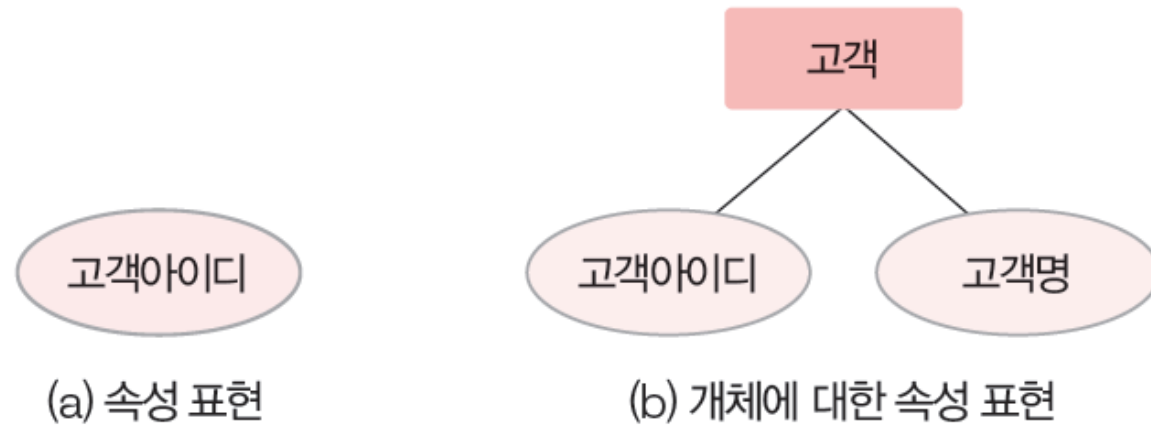


그림 4-7 속성의 E-R 다이어그램 표현 예



[요약] 속성(애트리뷰트) attribute

속성 종류:

단순 속성

복합 속성

다중값 속성

유도 속성

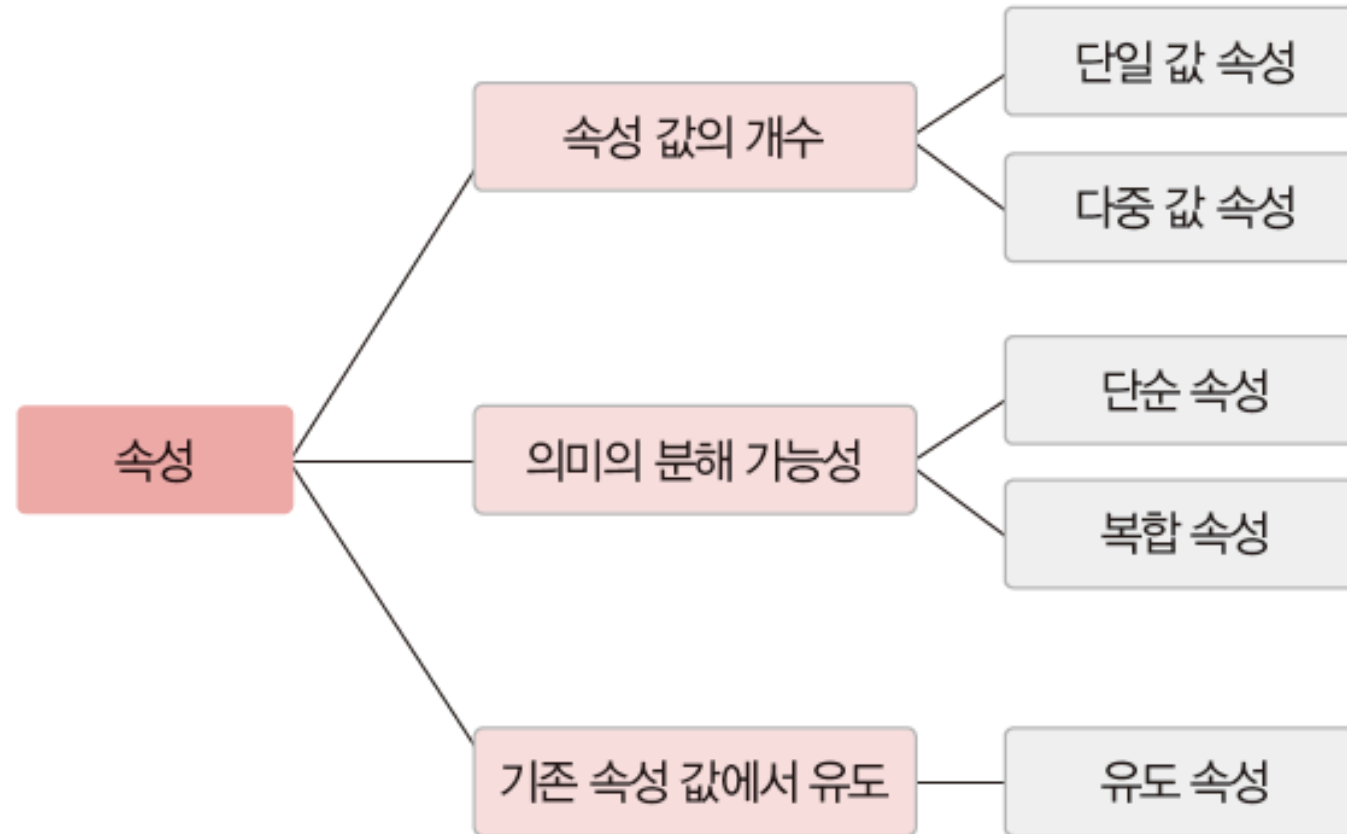


그림 4-8 속성의 분류

[요약] 속성(애트리뷰트) attribute



속성 종류:

단순 속성: 예) 이름, 적립금

복합 속성

다중값 속성 예) 연락처, 저자

유도 속성

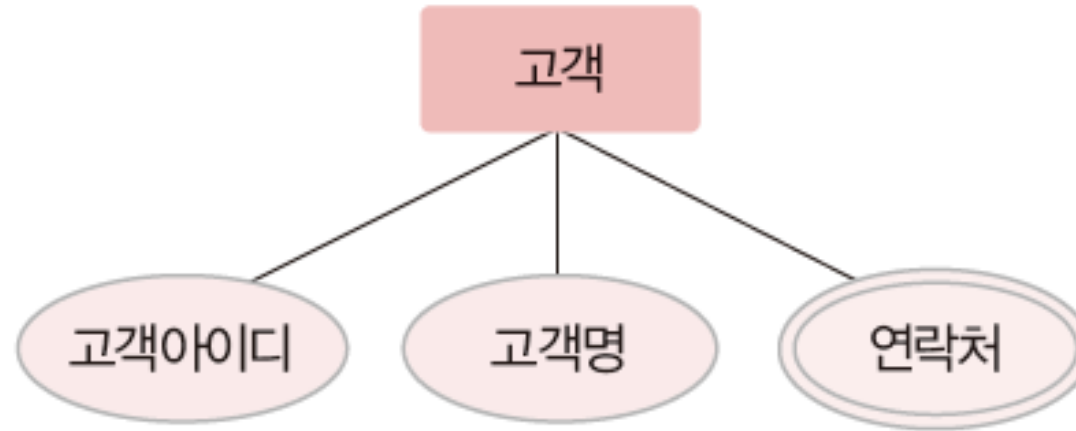


그림 4-9 다중 값 속성의 E-R 다이어그램 표현 예 : 연락처 속성

[요약] 속성(애트리뷰트) attribute



속성 종류:

단순 속성: 예) 고객아이디, 고객명

복합 속성: 예) 생년월일, 주소

다중값 속성

유도 속성

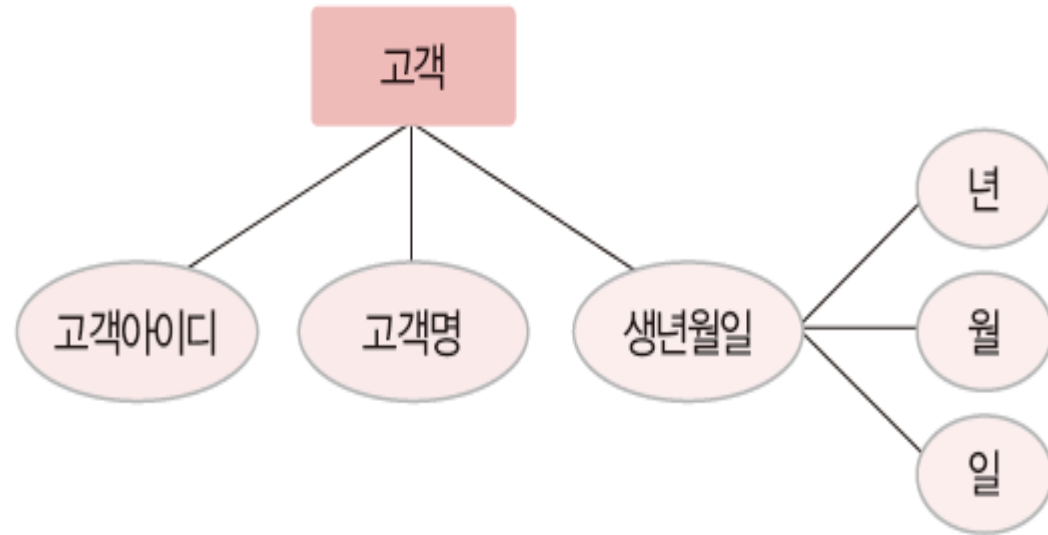


그림 4-10 복합 속성의 E-R 다이어그램 표현 예 : 생년월일 속성

[요약] 속성(애트리뷰트) attribute



속성 종류:

단순 속성:

복합 속성:

다중값 속성

유도 속성: 예) 할인율, 판매가격

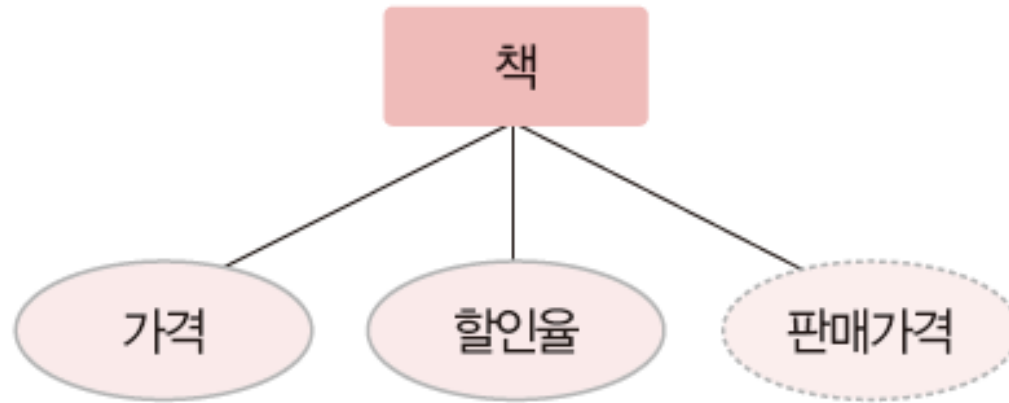


그림 4-11 유도 속성의 E-R 다이어그램 표현 예 : 판매가격 속성

❖ 유도속성:

기존의 다른 속성의 값에서 유도되어 결정되는 속성

02 개체-관계 모델

- **널 속성(null attribute)**

- 널 값이 허용되는 속성

- **널(null) 값**

- 아직 결정되지 않았거나 모르는 값 또는 존재하지 않는 값
- 공백이나 0과는 의미가 다름
- 예) 등급 속성이 널 값 → 등급이 아직 결정되지 않았음을 의미

02 개체-관계 모델

- 키 속성(key attribute)

- 각 개체 인스턴스를 식별하는 데 사용되는 속성
- 모든 개체 인스턴스의 키 속성 값이 다름
- 둘 이상의 속성들로 구성되기도 함
- 예) 고객 개체의 고객아이디 속성
- E-R 다이어그램에서 밑줄로 표현

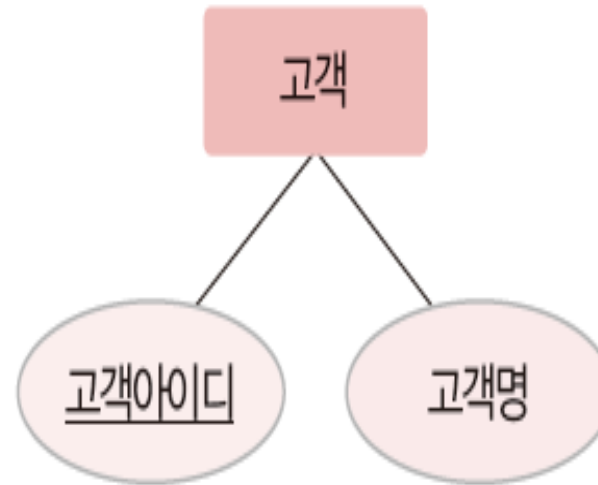


그림 4-12 키 속성의 E-R 다이어그램 표현 예: 고객아이디 속성

[요약] 관계 relationship



관계 종류:

일대일 관계

일대다 관계

다대다 관계

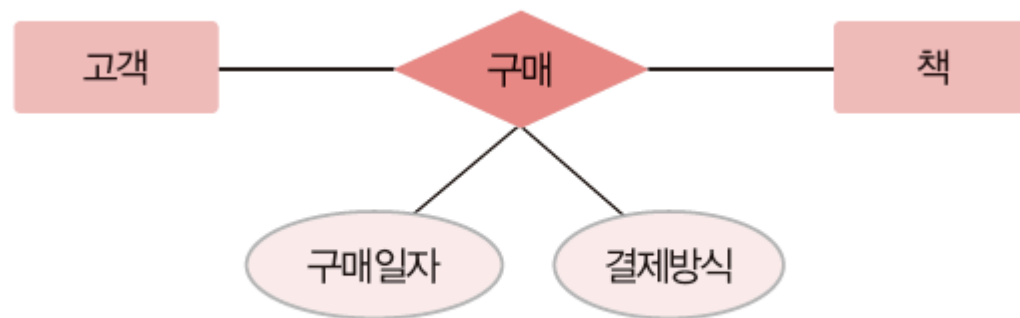


그림 4-13 관계의 E-R 다이어그램 표현 예 : 구매 관계

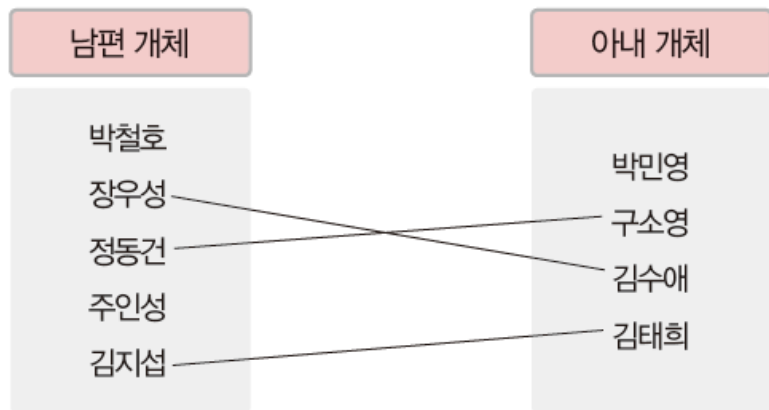


그림 4-14 일대일 관계의 예 : 남편과 아내 개체의 혼인 관계

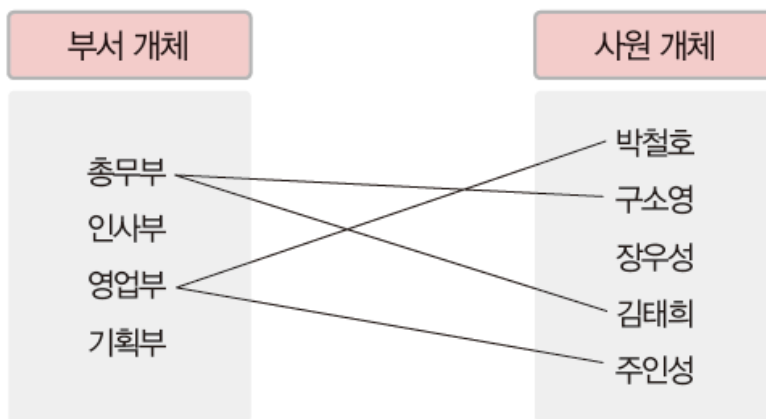


그림 4-15 일대다 관계의 예 : 부서와 사원 개체의 소속 관계

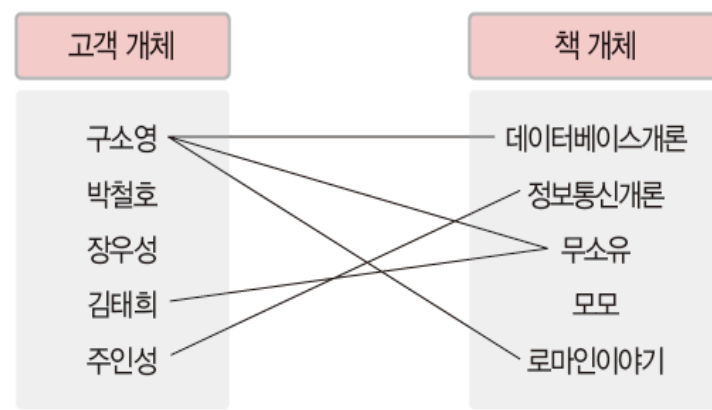


그림 4-16 다대다 관계의 예 : 고객과 책 개체의 구매 관계

[요약] 관계: 필수참여 vs. 선택참여

❖ 관계 참여도

필수 참여(전체 참여)

선택 참여(부분 참여)



그림 4-17 필수적 참여 관계의 E-R 다이어그램 표현 예: 고객 개체의 필수적 참여 관계

[요약] 개체

❖ 개체

정규 개체

약한 개체(weak entity)



그림 4-18 관계 종속성의 E-R 다이어그램 표현 예 : 약한 개체인
부양가족 개체

[요약] ERD

❖ ERD:
개체
속성
관계

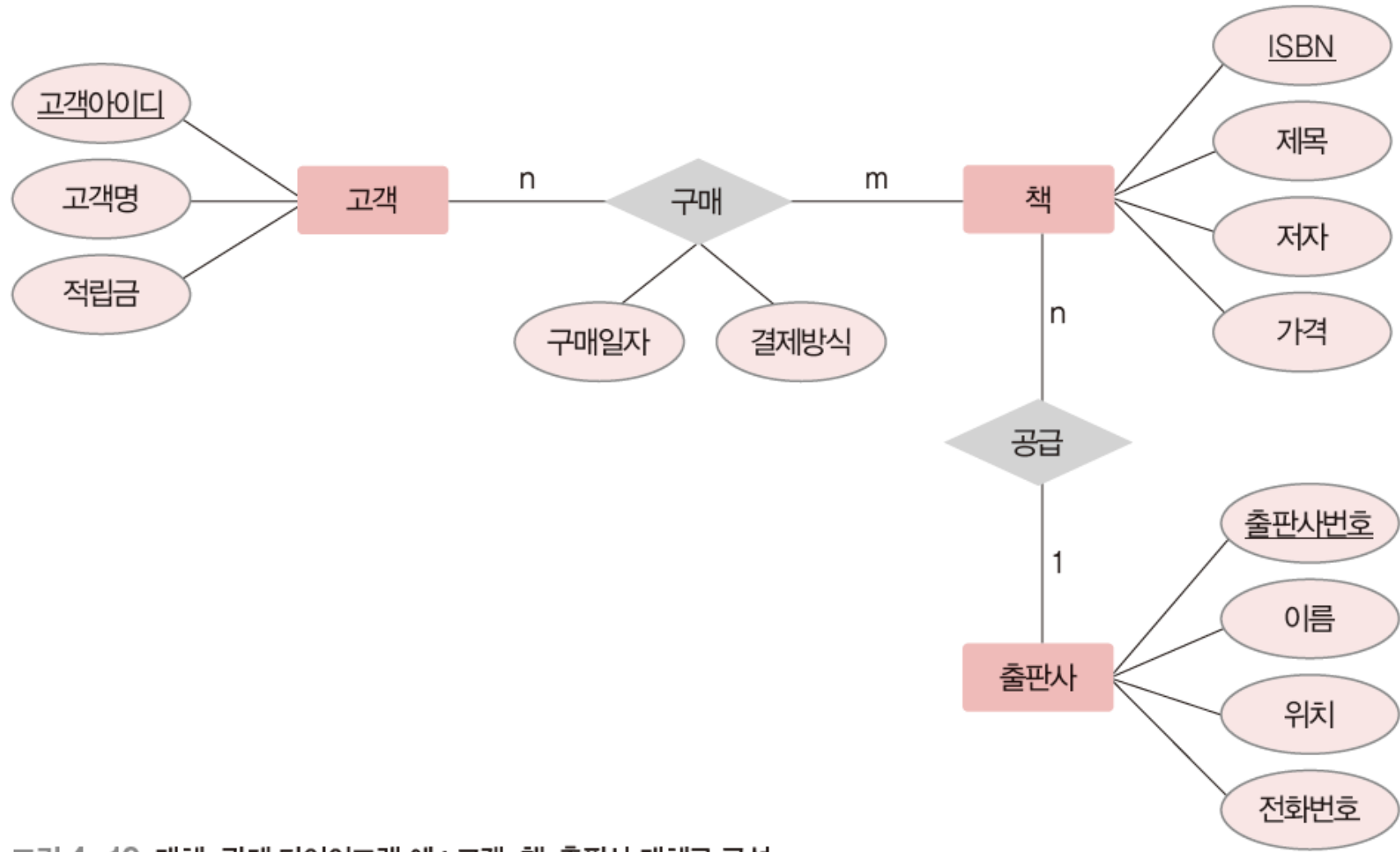


그림 4-19 개체-관계 다이어그램 예 : 고객, 책, 출판사 개체로 구성

[요약] 논리적 데이터 모델

❖ 논리적 데이터 모델:

E-R 다이어그램으로 표현된 개념적 구조를 데이터베이스에 저장할 형태로 표현한 논리적 구조

❖ 논리적 데이터 모델

계층 데이터 모델

네트워크 데이터 모델

관계 데이터 모델

❖ 관계 데이터 모델

- 일반적으로 많이 사용되는 논리적 데이터 모델
- 데이터베이스의 논리적 구조가 2차원 테이블 형태

[요약] 논리적 데이터 모델

❖ 계층 데이터 모델(Hierarchical data model)

- 트리(tree) 구조
- 루트 역할을 하는 개체가 존재하고 사이클이 존재하지 않음
- 개체 간에 상하 관계가 성립
- 부모 개체 / 자식 개체
- 부모와 자식 개체는 일대다(1:n) 관계만 허용
- 두 개체 사이에 하나의 관계만 정의할 수 있음
- 다대다(M:N) 관계를 직접 표현할 수 없음

❖ 논리적 데이터 모델

계층 데이터 모델

네트워크 데이터 모델

관계 데이터 모델

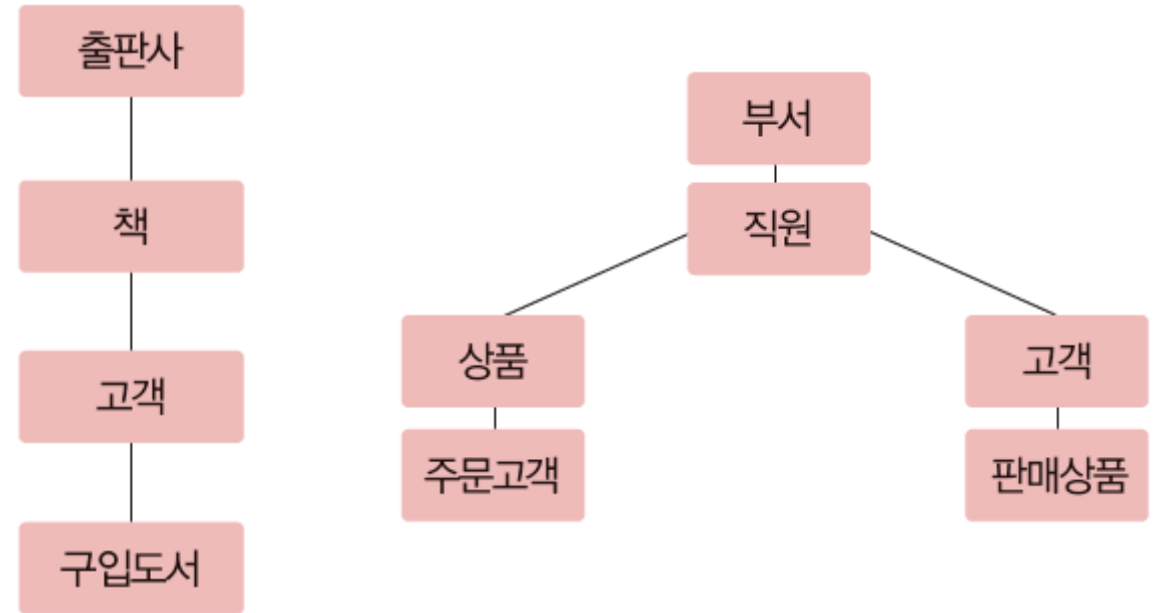


그림 4-20 계층 데이터 모델의 예

❖ 단점:

개념적 구조를 모델링하기 어려워 구조가 복잡
데이터의 삽입·삭제·수정·검색이 쉽지 않다

[요약] 논리적 데이터 모델

❖ 네트워크 데이터 모델(Network data model)

- 그래프(graph) 구조
- 개체 간에는 일대다(1:n) 관계만 허용됨
- 오너(owner) / 멤버(member)
- 두 개체 사이에 여러 관계를 정의할 수 있어 이름으로 구별함
- 다대다(M:N) 관계를 직접 표현할 수 없음

❖ 논리적 데이터 모델
계층 데이터 모델
네트워크 데이터 모델
관계 데이터 모델

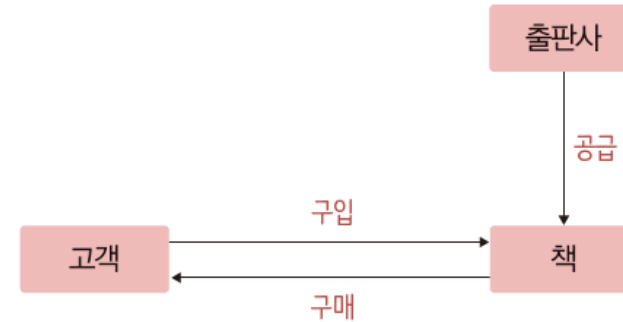
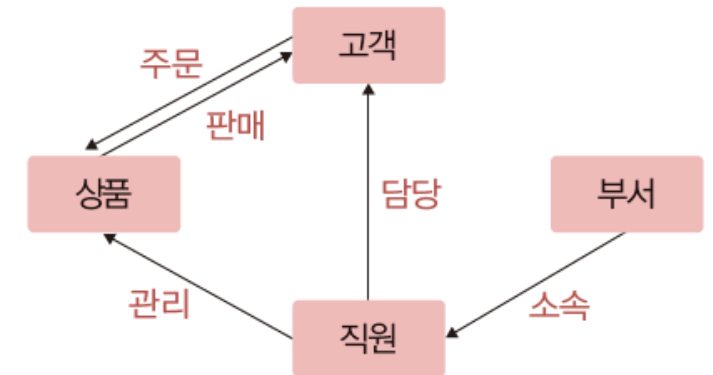


그림 4-21 네트워크 데이터 모델의 예



❖ 단점:

개념적 구조를 모델링하기 어려워 구조가 복잡
데이터의 삽입·삭제·수정·검색이 쉽지 않다

SQL: 뷰(view)

7주차 7-02

담당교수: 김희숙
(jasmin11@hanmail.net)

[실습] 실습환경 설정

- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)

고객, 제품, 주문 테이블 생성

데이터베이스이름: handb

[실습환경]

1. C 드라이브에 DBDATA 폴더 생성
2. han-mysql.sql 을 복사
3. cmd 에서 mysql 접속

C:W>mysql -uroot -p

4. mysql 에서 다음 명령을 실행

mysql>source c:/dbdata/han-mysql.sql;

[실습] 실습환경 설정

- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)

고객, 제품, 주문 테이블 생성

데이터베이스이름: handb

고객 테이블

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
orange	김용욱	22	silver	학생	0
melon	성원용	35	gold	회사원	5000
peach	오형준	NULL	silver	의사	300
pear	채광주	31	silver	회사원	500

주문 테이블

주문번호	주문고객	주문제품	수량	배송지	주문일자
o01	apple	p03	10	서울시 마포구	2019-01-01
o02	melon	p01	5	인천시 계양구	2019-01-10
o03	banana	p06	45	경기도 부천시	2019-01-11
o04	carrot	p02	8	부산시 금정구	2019-02-01
o05	melon	p06	36	경기도 용인시	2019-02-20
o06	banana	p01	19	충청북도 보은군	2019-03-02
o07	apple	p03	22	서울시 영등포구	2019-03-15
o08	pear	p02	50	강원도 춘천시	2019-04-10
o09	banana	p04	15	전라남도 목포시	2019-04-11
o10	carrot	p03	20	경기도 안양시	2019-05-22

그림 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

제품 테이블

제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운짬면	2500	5500	민국푸드
p03	콩떡파이	3600	2600	한빛제과
p04	맛난초콜릿	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	달콤비스킷	1650	1500	한빛제과

3. cmd 에서 mysql 접속

mysql -uroot -p

4. mysql 에서 다음 명령을 실행

source c:/dbdata/han-mysql.sql;

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -p
```

```
C:\>mysql -uroot -p
```

```
Enter password: *****
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 12
```

```
Server version: 8.0.17 MySQL Community Server - GPL
```

```
Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current statement.
```

```
mysql> -- 교재 7장 실습환경 준비
```

```
mysql> source c:/dbdata/han-mysql.sql;
```

```
ERROR 1007 (HY000): Can't create database 'handb'; database exists
```

```
Database changed
```

```
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

```
Query OK, 0 rows affected (0.07 sec)
```

```
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
Query OK, 0 rows affected (0.06 sec)
```

```
Query OK, 0 rows affected (0.06 sec)
```

[실습] 실습환경 설정

- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)

C:\WINDOWS\system32\cmd.exe - mysql -uroot -p

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
melon	성원용	35	gold	회사원	5000
orange	김용욱	22	silver	학생	0
peach	오형준	NULL	silver	의사	300
pear	채광주	31	silver	회사원	500

7 rows in set (0.00 sec)

제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운쫄면	2500	5500	민국푸드
p03	콩떡파이	3600	2600	한빛제과
p04	맛난초콜렛	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	달콤비스킷	1650	1500	한빛제과

7 rows in set (0.00 sec)

[실습] (han-mysql.sql)

주문번호	주문고객	주문제품	수량	배송지	주문일자
o01	apple	p03	10	서울시 마포구	2013-01-01
o02	melon	p01	5	인천시 계양구	2013-01-10
o03	banana	p06	45	경기도 부천시	2013-01-11
o04	carrot	p02	8	부산시 금정구	2013-02-01
o05	melon	p06	36	경기도 용인시	2013-02-20
o06	banana	p01	19	충청북도 보은군	2013-03-02
o07	apple	p03	22	서울시 영등포구	2013-03-15
o08	pear	p02	50	강원도 춘천시	2013-04-10
o09	banana	p04	15	전라남도 목포시	2013-04-11
o10	carrot	p03	20	경기도 안양시	2013-05-22

10 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>



[요약] 뷰(view) 생성

- **뷰(view):**
 - 가상의 테이블(다른 테이블을 기반으로 만들어진 가상의 테이블)
 - 장점: **사용의 편의성, 보안**

```
-- 뷰 정의(뷰 생성)
CREATE VIEW 뷰이름
as
SELECT
FROM
[with check option]
```

❖ 뷰 정의할 때, ORDER BY 사용 불가능

```
DROP TABLE 테이블명;
```

```
DROP VIEW 뷰이름;
```

❖ **WITH CHECK OPTION:**

뷰 삽입/수정할 때,
뷰의 정의 조건을 위반하면 수행되지 않도록 하는
제약조건





[요약] 뷰(view) 삽입/수정/삭제

-- 뷰 삽입/수정/삭제: 실제 테이블에 반영

❖ 뷰 삽입/수정/삭제가 되지 않는 경우:

1. 기본 테이블의 기본키가 누락된 뷰
2. 집계 함수로 작성된 뷰
3. DISTINCT 키워드 포함한 뷰
4. GROUP BY 문법 포함한 뷰
5. 조인으로 작성된 뷰


```

1  -- (schema_view-ex.sql)
2  -- 3단계 스키마 구조
3  • use studydb;
4
5  • drop table 가입고객;
6
7  • CREATE TABLE 가입고객 (
8      번호 int NOT NULL,
9      이름 char(10),
10     성별 char(2) ,
11     나이 int ,
12     직업 char(10),
13     주소 char(20) ,
14     연락처 char(20) ,
15     PRIMARY KEY(번호)
16 );
17
18 • INSERT INTO 가입고객 VALUES (1, '홍길동', '남', 20, '학생', '서울', NULL) ,
19                                (2, '임꺽정', '남', 22, '학생', '인천', '010-1111-1111') ,
20                                (3, '신아로미', '여', 20, '학생', '서울', '010-2222-2222')
21 ;
22
23 • select * from 가입고객;

```

번호	이름	성별	나이	직업	주소	연락처
1	홍길동	남	20	학생	서울	NULL
2	임꺽정	남	22	학생	인천	010-1111-1111
3	신아로미	여	20	학생	서울	010-2222-2222

가입고객 4 x

```

-- 3단계 스키마 구조
use studydb;

```

```
drop table 가입고객;
```

```

CREATE TABLE 가입고객 (
    번호 int NOT NULL,
    이름 char(10),
    성별 char(2) ,
    나이 int ,
    직업 char(10),
    주소 char(20) ,
    연락처 char(20) ,
    PRIMARY KEY(번호)
);

```

```

INSERT INTO 가입고객 VALUES (1, '홍길동', '남', 20, '학생', '서울', NULL) ,
                              (2, '임꺽정', '남', 22, '학생', '인천', '010-1111-1111') ,
                              (3, '신아로미', '여', 20, '학생', '서울', '010-2222-2222')
;

```

```
select * from 가입고객;
```



```
-- 가입고객(번호,이름,성별,나이,직업,주소,연락처)
-- 고객분석팀(성별,나이,직업)
-- 상품배송팀(고객번호,고객이름,주소,연락처)
```



-- 뷰 생성

```
CREATE VIEW 고객분석팀
as
```

```
SELECT 성별, 나이, 직업
FROM   가입고객;
```

```
select * from 고객분석팀;
```

```
25 -- 가입고객(번호,이름,성별,나이,직업,주소,연락처)
26 -- 고객분석팀(성별,나이,직업)
27 -- 상품배송팀(고객번호,고객이름,주소,연락처)
28
29 -- 뷰 생성
30 • CREATE VIEW 고객분석팀
31   as
32     SELECT 성별, 나이, 직업
33     FROM   가입고객;
34
35 • select * from 고객분석팀;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	성별	나이	직업
▶	남	20	학생
	남	22	학생
	여	20	학생

```
-- 가입고객(번호,이름,성별,나이,직업,주소,연락처)
-- 고객분석팀(성별,나이,직업)
-- 상품배송팀(고객번호,고객이름,주소,연락처)
```

-- 뷰 생성

CREATE VIEW 상품배송팀(고객번호,고객이름,주소,연락처)

as

```
SELECT 번호, 이름, 주소, 연락처
FROM 가입고객;
```

```
select * from 상품배송팀;
```

```
36
37 -- 가입고객(번호,이름,성별,나이,직업,주소,연락처)
38 -- 고객분석팀(성별,나이,직업)
39 -- 상품배송팀(고객번호,고객이름,주소,연락처)
40
41 -- 뷰 생성
42 • CREATE VIEW 상품배송팀(고객번호,고객이름,주소,연락처)
43 as
44     SELECT 번호, 이름, 주소, 연락처
45     FROM 가입고객;
46
47 • select * from 상품배송팀;
```

Result Grid

	고객 번호	고객 이름	주소	연락처
▶	1	홍길동	서울	NULL
	2	임궽정	인천	010-1111-1111
	3	신아로미	서울	010-2222-2222

[예제] 뷰 생성(CREATE VIEW)

- (수정) 예제7-55 고객 테이블에서 등급이 vip인 고객의 고객아이디, 고객이름, 나이, 등급으로 구성된
- 뷰를 우수고객이라는 이름으로 생성해보자
- 우수고객 뷰의 모든 내용을 검색해보자

```
CREATE VIEW 우수고객(고객아이디, 고객이름, 나이, 등급)
as
SELECT  고객아이디, 고객이름, 나이, 등급
FROM    고객
WHERE   등급 = 'vip'
WITH CHECK OPTION;
```

```
-- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
-- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
-- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)
```

```
-- 우수고객(고객아이디, 고객이름, 나이, 등급)
```

```
-- 뷰 생성
```

```
CREATE VIEW 우수고객(고객아이디, 고객이름, 나이, 등급)
```

```
as
```

```
    SELECT 고객아이디, 고객이름, 나이, 등급
```

```
    FROM    고객
```

```
    WHERE   등급 = 'vip'
```

```
WITH CHECK OPTION;
```

```
select * from 우수고객;
```

```
-- 뷰 필드 생략 가능
```

```
CREATE VIEW 우수고객
```

```
as
```

```
    SELECT 고객아이디, 고객이름, 나이, 등급
```

```
3  -- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
4  -- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
5  -- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)
6
7  -- 우수고객(고객아이디, 고객이름, 나이, 등급)
8
9  -- (수정) 예제7-55 고객 테이블에서 등급이 vip인 고객의
10 -- |고객아이디, 고객이름, 나이, 등급으로 구성된
11 -- 뷰를 우수고객이라는 이름으로 생성해보자
12 -- 우수고객 뷰의 모든 내용을 검색해보자
13 • drop view 우수고객;
14
15 • CREATE VIEW 우수고객(고객아이디, 고객이름, 나이, 등급)
16 as
17     SELECT  고객아이디, 고객이름, 나이, 등급
18     FROM    고객
19     WHERE   등급 = 'vip'
20     WITH CHECK OPTION;
21
22 • select * from 우수고객;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export: <input type="button" value="Export"/> Wrap Cell Content: <input type="button" value="Wrap"/>			
고객아이디	고객이름	나이	등급
banana	김선우	25	vip

```
-- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
-- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
-- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)
```

```
-- 우수고객(고객아이디, 고객이름, 나이, 등급)
```

```
-- 뷰 생성
```

```
CREATE VIEW 우수고객(고객아이디, 고객이름, 나이, 등급)
as
```

```
    SELECT 고객아이디, 고객이름, 나이, 등급
    FROM   고객
    WHERE  등급 = 'vip'
```

```
WITH CHECK OPTION;
```

```
select * from 우수고객;
```

```
-- WITH CHECK OPTION
```

```
뷰로 작성된 조건에 만족하지 않는 경우,
입력, 수정, 삭제 불가능
```

```
31 -- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
32 -- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
33 -- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)
34
35 -- 우수고객(고객아이디, 고객이름, 나이, 등급)
```

```
37 • select * from 고객;
```

```
39 • select * from 우수고객;
```

```
41 -- 뷰에 입력/수정/삭제 시도(with check option 설정 경우)
42 • insert into 우수고객(고객아이디, 고객이름, 나이, 등급)
43   values('shinmi', '신아로미', 20, 'gold');
```

```
45 • update 우수고객
46   set   등급 = 'gold'
47   where 고객아이디 = 'banana';
```

```
48
49 • delete
50   from   우수고객
51   where  고객아이디 = 'banana';
```

참조무결성 제약조건 위배

[예제] 뷰 생성(CREATE VIEW)

- 예제7-56 제품 테이블에서 제조업체별 제품수로 구성된 뷰를
- 업체별제품수라는 이름으로 생성해보자
- 업체별제품수 뷰의 모든 내용을 검색해보자

CREATE VIEW 업체별제품수(제조업체, 제품수)

as

SELECT 제조업체, COUNT(*)

FROM 제품

GROUP BY 제조업체

[실습] (view) 집계함수

-- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
 -- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
 -- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)

-- 업체별제품수(제조업체, 제품수)

-- 뷰 생성

CREATE VIEW 업체별제품수(제조업체, 제품수)

as

SELECT 제조업체, COUNT(*)
 FROM 제품
 GROUP BY 제조업체

select * from 업체별제품수;

-- 뷰 필드 생략 불가능

CREATE VIEW 업체별제품수(제조업체, 제품수)

as

SELECT 제조업체, COUNT(*)

[실습] 뷰(view-ex.txt)

```

3  -- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
4  -- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
5  -- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)
6
7  • select * from 제품;
8
9  -- 예제7-56 제품 테이블에서 제조업체별 제품수로 구성된 뷰를
10 -- 업체별제품수라는 이름으로 생성해보자
11 -- 업체별제품수 뷰의 모든 내용을 검색해보자
12
13 • CREATE VIEW 업체별제품수(제조업체, 제품수)
14 as
15   SELECT 제조업체, COUNT(*)
16   FROM   제품
17   GROUP BY 제조업체;
18
19 • select * from 업체별제품수;
```

제조업체	제품수
대한식품	2
민국푸드	2
한빛제과	3

제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운짜면	2500	5500	민국푸드
p03	콩떡파이	3600	2600	한빛제과
p04	맛난초콜렛	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	달콤비스킷	1650	1500	한빛제과
NULL	NULL	NULL	NULL	NULL

[예제] 뷰 조회(SELECT)

-- 예제7-57 우수고객 뷰에서 나이가 25세 이상인 고객에 대한 모든 내용을 검색해보자

```
SELECT *  
FROM   우수고객  
WHERE  나이 >= 25;
```


[실습] (view) 뷰 조회

-- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
 -- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
 -- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)

-- 뷰 조회
SELECT *
FROM 우수고객
WHERE 나이 >= 25;

select 고객아이디, 고객이름, 나이, 등급
from 고객
where 등급 = 'vip'
AND 나이 >= 25;

-- 뷰 조회
 뷰에 **SELECT** : 내부적으로 기본테이블에 대한
SELECT 문으로 변환하여 수행

[실습] 뷰(view-ex.txt)

3 -- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
 4 -- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
 5 -- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)
 6
 7 -- 예제7-57 우수고객 뷰에서 나이가 25세 이상인 고객에 대한 모든 내용을 검색해보자
 8

9 • **SELECT ***
 10 **FROM** 우수고객
 11 **WHERE** 나이 >= 25;

13 • **select** 고객아이디, 고객이름, 나이, 등급
 14 **from** 고객
 15 **where** 등급 = 'vip'
 16 **AND** 나이 >= 25;

고객아이디	고객이름	나이	등급
banana	김선우	25	vip
NULL	NULL	NULL	NULL

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
melon	성원용	35	gold	회사원	5000
orange	김용욱	22	silver	학생	0
peach	오형준	NULL	silver	의사	300
pear	채광주	31	silver	회사원	500
NULL	NULL	NULL	NULL	NULL	NULL

[예제] 뷰 입력

- 예제7-58 제품번호가 p08, 재고량이 1000, 제조업체가 신선식품인 새로운 제품의 정보를
- 제품1 뷰에 삽입해보자.
- 제품1 뷰에 있는 모든 내용을 검색해보자

```
drop view 제품1;
```

```
CREATE VIEW 제품1
```

```
as
```

```
SELECT 제품번호, 재고량, 제조업체  
FROM 제품;
```

```
-- 뷰에 삽입
```

```
insert into 제품1
```

```
values('p08',1000,'신선식품');
```

```
select * from 제품1;
```

```
select * from 제품;
```

[실습] (view) 뷰 입력

-- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
-- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
-- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)

-- 뷰 생성

CREATE VIEW 제품1

as

SELECT 제품번호, 재고량, 제조업체
FROM 제품;

-- 뷰 입력

insert into 제품1 values('p08',1000,'신선식품');

-- 뷰 조회

select * from 제품1;

-- 기본 테이블(base table) 조회

select * from 제품;

-- 뷰 입력

뷰에 INSERT : 뷰에 입력하지 않은 필드는
내부적으로 기본 테이블에 **널 값**으로
입력된다

[실습] 뷰(view-ex.txt)

3 -- 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
4 -- 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
5 -- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)
6
7 -- 예제7-58 제품번호가 p08, 재고량이 1000, 제조업체가 신선식품인
8 -- 새로운 제품의 정보를 제품1 뷰에 삽입해보자.
9 -- 제품1 뷰에 있는 모든 내용을 검색해보자

10
11 • **drop view 제품1;**

12
13 • **CREATE VIEW 제품1**

14 **as**

15 **SELECT** 제품번호, 재고량, 제조업체
16 **FROM** 제품;

17
18 • **insert into 제품1 values('p08',1000,'신선식품');**

19
20 • **select * from 제품1;**

21

Result Grid

제품번호	재고량	제조업체
p01	5000	대한식품
p02	2500	민국푸드
p03	3600	한빛제과
p04	1250	한빛제과
p05	2200	대한식품
p06	1000	민국푸드
p07	1650	한빛제과
p08	1000	신선식품

제품 12 x

22 • **select * from 제품;**

Result Grid

제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운쫄면	2500	5500	민국푸드
p03	쿵덕파이	3600	2600	한빛제과
p04	맛난초콜렛	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	할금비스켓	1650	1500	한빛제과
p08	NULL	1000	NULL	신선식품

제품 13 x

[예제] 뷰 조작 불가능한 경우

- 기본키 제외한 필드를 제품2 뷰 생성하고
- 제품2 뷰에 삽입해보자.
- 제품2 뷰에 있는 모든 내용을 검색해보자

```
drop view 제품2;
```

```
CREATE VIEW 제품2
```

```
as
```

```
SELECT 제품명, 재고량, 제조업체  
FROM 제품;
```

```
-- 뷰에 삽입
```

```
insert into 제품2
```

```
values('사원냉면',1000,'신선식품');
```

```
select * from 제품2;
```

```
select * from 제품;
```



[요약] 뷰(view) 삽입/수정/삭제

-- 뷰 삽입/수정/삭제: 실제 테이블에 반영

❖ 뷰 삽입/수정/삭제가 되지 않는 경우:

1. 기본 테이블의 기본키가 누락된 뷰
2. 집계 함수로 작성된 뷰
3. DISTINCT 키워드 포함한 뷰
4. GROUP BY 문법 포함한 뷰
5. 조인으로 작성된 뷰

SQL: SELECT 문법(집계함수,그룹화)

7주차 7-03

담당교수: 김희숙
(jasmin11@hanmail.net)

데이터베이스 언어(SQL)

□ 데이터베이스 언어(SQL)

- ✓ 데이터 정의어(DDL)
- ✓ 데이터 조작어(DML)
- ✓ 데이터 제어어(DCL)

SQL	명령어
DDL	CREATE ALTER DROP
DML	INSERT UPDATE DELETE SELECT
DCL	GRANT REVOKE

[요약] 집계함수

COUNT(), MIN(), MAX() : 문자형, 숫자형
SUM(), AVG() : 숫자형



- 집계함수(aggregate function):
 - 여러 개 행의 값들을 계산하여 하나의 결과를 산출하는 함수

COUNT(*) : NULL 포함하여 계산
COUNT(필드): NULL 제외하고 계산
SUM(필드): 합계를 계산
AVG(필드): 평균을 계산

/* SELECT 문법 순서 */
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

[예제] SELECT (집계함수)

-- 집단함수

CREATE TABLE 성적(

이름 varchar(9) NOT NULL primary key,

점수 int

);

INSERT INTO 성적 (이름, 점수) VALUES ('홍길동', 87);

INSERT INTO 성적 (이름, 점수) VALUES ('임꺽정', 60);

INSERT INTO 성적 (이름, 점수) VALUES ('박찬호', 75);

INSERT INTO 성적 (이름, 점수) VALUES ('선동열', 70);

INSERT INTO 성적 (이름, 점수) VALUES ('홍명보', 90);

INSERT INTO 성적 (이름, 점수) VALUES ('차범근', 75);

INSERT INTO 성적 (이름, 점수) VALUES ('강성범', 68);

INSERT INTO 성적 (이름, 점수) VALUES ('신동엽', null);

SQL> select * from 성적;

이름	점수
홍길동	87
임꺽정	60
박찬호	75
선동열	70
홍명보	90
차범근	75
강성범	68
신동엽	

	이름	점수
▶	강성범	68
	박찬호	75
	선동열	70
	신동엽	NULL
	임꺽정	60
	차범근	75
	홍길동	87
	홍명보	90
*	NULL	NULL

성적 14 x

8 개의 행이 선택되었습니다.

[예제] SELECT (Group by)

```
-- 그룹화
-- GROUP BY, HAVING
CREATE TABLE 성적2 (
  이름 varchar(9) NOT NULL primary key ,
  과목 varchar(8),
  점수 int
);
INSERT INTO 성적2 VALUES ('홍길동', '영어', 87 );
INSERT INTO 성적2 VALUES ('임꺽정', '수학', 60 );
INSERT INTO 성적2 VALUES ('박찬호', '국어', 75 );
INSERT INTO 성적2 VALUES ('선동열', '영어', 70 );
INSERT INTO 성적2 VALUES ('홍명보', '수학', 90 );
INSERT INTO 성적2 VALUES ('차범근', '수학', 75 );
INSERT INTO 성적2 VALUES ('강성범', '수학', 68 );
INSERT INTO 성적2 VALUES ('신동엽', '영어', null);
```

SQL> select * from 성적2;

이름	과목	점수
홍길동	영어	87
임꺽정	수학	60
박찬호	국어	75
선동열	영어	70
홍명보	수학	90
차범근	수학	75
강성범	수학	68
신동엽	영어	

8 개의 행이 선택되었습니다.

[실습2] SELECT (집계함수)

- [실습 2] 집계함수, Group by : (sungjuk_group.sql)
- 성적(이름, 점수)
- 성적2(이름, 과목, 점수)

- 1-1) 최고 점수를 검색하라
- 1-2) 최저 점수를 검색하라
- 1-3) 점수합계를 검색하라
- 1-4) 평균점수를 검색하라
- 1-5) 학생수는 모두 몇 명인지 검색하라
- 1-6) 시험에 응시한 학생수는 모두 몇 명인지 검색하라

	이름	점수
▶	강성범	68
	박찬호	75
	선동열	70
	신동엽	NULL
	임격정	60
	차범근	75
	홍길동	87
	홍명보	90
*	NULL	NULL

성적 14 ×

[실습2] SELECT (집계함수)

[실습 2]

- 성적(이름, 점수)
- 1-1) 최고 점수를 검색하라
- 1-2) 최저 점수를 검색하라

```
SELECT MAX(점수)
FROM 성적;
```

```
SELECT MIN(점수)
FROM 성적;
```

[실습] (sungjuk_group.sql)

	이름	점수
▶	강성범	68
	박찬호	75
	선동열	70
	신동엽	NULL
	임격정	60
	차범근	75
	홍길동	87
	홍명보	90
*	NULL	NULL

성적 14 ×



[실습2] SELECT (집계함수)

[실습 2]

- 성적(이름, 점수)
- 1-3) 점수합계를 검색하라
- 1-4) 평균점수를 검색하라

```
SELECT SUM(점수)
FROM 성적;
```

```
SELECT AVG(점수)
FROM 성적;
```

[실습] (sungjuk_group.sql)

	이름	점수
▶	강성범	68
	박찬호	75
	선동열	70
	신동엽	NULL
	임격정	60
	차범근	75
	홍길동	87
	홍명보	90
*	NULL	NULL

성적 14 ×



[실습2] SELECT (집계함수)

[실습] (sungjuk_group.sql)

[실습 2]

- 성적(이름, 점수)
- 1-5) 학생수는 모두 몇 명인지 검색하라
- 1-6) 시험에 응시한 학생수는 모두 몇 명인지 검색하라

	이름	점수
▶	강성범	68
	박찬호	75
	선동열	70
	신동엽	NULL
	임격정	60
	차범근	75
	홍길동	87
	홍명보	90
*	NULL	NULL

성적 14 ×



SELECT COUNT(*)

FROM 성적;

SELECT COUNT(점수)

FROM 성적;

```
1  -- 1-5) 학생수는 모두 몇 명인지 검색하라
2  • SELECT COUNT(*) FROM 성적;
3
4  -- 1-6) 시험에 응시한 학생수는 모두 몇 명인지 검색하라
5  -- (MySQL)
6  • SELECT COUNT(점수) as '응시 학생수'
7    FROM 성적;
8
9  -- (MySQL)
10 • SELECT COUNT(점수) `응시 학생수`
11    FROM 성적;
12
```

[실습2] SELECT (Group by)

- [실습 2] 집계함수, Group by : (sungjuk_group.sql)
- 성적(이름, 점수)
- 성적2(이름, 과목, 점수)
- 2-1) 각 과목수는 몇 개인지 검색하라(DISTINCT 사용)
- 2-2) 과목별 수강생은 몇 명인지 검색하라(GROUP BY)
- 2-3) 과목별 평균점수를 검색하라(GROUP BY)
- 2-4) 과목별 평균점수 75 보다 높은 학생의 과목별 평균점수를 검색하라(HAVING)
- 2-5) 점수가 70 이상인 과목이름, 과목 평균점수를 과목의 과목별 평균점수가 75 이상인 것만
과목별 평균점수가 높은 순으로 검색하라(ORDER BY)

	이름	과목	점수
▶	강성범	수학	68
	박찬호	국어	75
	선동열	영어	70
	신동엽	영어	NULL
	임격정	수학	60
	차범근	수학	75
	홍길동	영어	87
	홍명보	수학	90
★	NULL	NULL	NULL

성적2 15 ×

[실습2] SELECT (Group by)

[실습] (sungjuk_group.sql)

[실습 2]

- 성적2(이름, 과목, 점수)
- 2-1) 각 과목수는 몇 개인지 검색하라(DISTINCT 사용)

```
SELECT COUNT(DISTINCT 과목)
FROM 성적2;
```

	이름	과목	점수
▶	강성범	수학	68
	박찬호	국어	75
	선동열	영어	70
	신동엽	영어	NULL
	임궽정	수학	60
	차범근	수학	75
	홍길동	영어	87
	홍명보	수학	90
*	NULL	NULL	NULL

성적2 15 ×



[실습2] SELECT (Group by)

[실습] (sungjuk_group.sql)

[실습 2]

- 성적2(이름, 과목, 점수)
- 2-2) 과목별 수강생은 몇 명인지 검색하라(GROUP BY)

```
SELECT 과목 , COUNT(점수)
FROM 성적2
GROUP BY 과목;
```

	이름	과목	점수
▶	강성범	수학	68
	박찬호	국어	75
	선동열	영어	70
	신동엽	영어	NULL
	임궽정	수학	60
	차범근	수학	75
	홍길동	영어	87
	홍명보	수학	90
*	NULL	NULL	NULL

성적2 15 ×



(주의) 집계함수

집계 함수는 WHERE 절에서는 사용할 수 없고,
SELECT 절이나 HAVING 절에서만 사용 가능

[실습2] SELECT (Group by)

[실습] (sungjuk_group.sql)

[실습 2]

- 성적2(이름, 과목, 점수)
- 2-3) 과목별 평균점수를 검색하라(GROUP BY)

	이름	과목	점수
▶	강성범	수학	68
	박찬호	국어	75
	선동열	영어	70
	신동엽	영어	NULL
	임궽정	수학	60
	차범근	수학	75
	홍길동	영어	87
	홍명보	수학	90
*	NULL	NULL	NULL

성적2 15 ×



```
SELECT 과목 , AVG(점수)
FROM 성적2
GROUP BY 과목;
```

```
-- ROUND() 함수: 반올림
SELECT 과목 , ROUND(AVG(점수), 1)
FROM 성적2
GROUP BY 과목;
```

[실습2] SELECT (Group by)

[실습] (sungjuk_group.sql)

[실습 2]

-- 성적2(이름, 과목, 점수)

-- 2-4) 과목별 평균점수 75 보다 높은 학생의 과목별 평균점수를 검색하라(HAVING)

```
SELECT 과목 , AVG(점수)
FROM 성적2
GROUP BY 과목
HAVING AVG(점수) >= 75;
```

	이름	과목	점수
▶	강성범	수학	68
	박찬호	국어	75
	선동열	영어	70
	신동엽	영어	NULL
	임꺽정	수학	60
	차범근	수학	75
	홍길동	영어	87
	홍명보	수학	90
*	NULL	NULL	NULL

성적2 15 ×



[실습2] SELECT (Group by)

[실습] (sungjuk_group.sql)

[실습 2]

-- 성적2(이름, 과목, 점수)

-- 2-5) 점수가 70 이상인 과목이름, 과목 평균점수를 과목의 과목별 평균점수가 75 이상인 것만

-- 과목별 평균점수가 높은 순으로 검색하라(ORDER BY)

	이름	과목	점수
▶	강성범	수학	68
	박찬호	국어	75
	선동열	영어	70
	신동엽	영어	NULL
	임꺽정	수학	60
	차범근	수학	75
	홍길동	영어	87
	홍명보	수학	90
*	NULL	NULL	NULL

성적2 15 ×



SELECT 과목 , AVG(점수)

FROM 성적2

WHERE 점수 >= 70

GROUP BY 과목

HAVING AVG(점수) >= 75

ORDER BY AVG(점수) DESC;

03 SQL을 이용한 데이터 조작

❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

예제 7-28

제품 테이블에서 모든 제품의 단가 평균을 검색해보자.

```
▶▶ SELECT  AVG(단가)
FROM      제품;
```

결과 테이블

	AVG(단가)
1	2764.285714285714285714285714285714

제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운짬면	2500	5500	민국푸드
p03	쿵떡파이	3600	2600	한빛제과
p04	맛난초콜릿	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	달콤비스킷	1650	1500	한빛제과

AVG(단가)

2764

그림 7-7 모든 제품의 평균 단가를 계산하는 과정 : 제품 테이블

03 SQL을 이용한 데이터 조작

❖ 데이터 검색 : SELECT 문

■ 집계 함수를 이용한 검색

예제 7-30

고객 테이블에 고객이 몇 명 등록되어 있는지 검색해보자.

▶▶ ❶ 고객아이디 속성을 이용해 계산하는 경우

```
SELECT COUNT(고객아이디) AS 고객수
FROM 고객;
```

결과 테이블

	고객수
1	7

❷ 나이 속성을 이용해 계산하는 경우

```
SELECT COUNT(나이) AS 고객수
FROM 고객;
```

결과 테이블

	고객수
1	6

❸ *를 이용해 계산하는 경우

```
SELECT COUNT(*) AS 고객수
FROM 고객;
```

결과 테이블

	고객수
1	7

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
orange	김용욱	22	silver	학생	0
melon	성원용	35	gold	회사원	5000
pear	채광주	31	silver	회사원	500
peach	오형준	NULL	silver	의사	300

COUNT
(고객아이디)

7

COUNT
(나이)

6

그림 7-8 고객의 수를 계산하는 과정 : 고객 테이블

03 SQL을 이용한 데이터 조작

❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

예제 7-31

제품 테이블에서 제조업체의 수를 검색해보자.

```
▶▶ SELECT COUNT(DISTINCT 제조업체) AS "제조업체 수"
FROM 제품;
```

결과 테이블

	제조업체 수
1	3

```
1 -- 예제7-30 제품 테이블에서 제조업체의 수를 검색한다(중복 없이)
2
3 -- 오류 (MySQL)
4 SELECT COUNT(DISTINCT 제조업체) as "제조업체수"
5 FROM 제품;
6
7 • SELECT COUNT(DISTINCT 제조업체) as 제조업체수
8 FROM 제품;
9
10 -- (MySQL)
11 • SELECT COUNT(DISTINCT 제조업체) as '제조업체'
12 FROM 제품;
13
14 -- (MySQL)
15 • SELECT COUNT(DISTINCT 제조업체) as `제조업체수`
16 FROM 제품;
17
```

(주의) (MySQL)

-- 오류 (MySQL)

```
SELECT COUNT(DISTINCT 제조업체) as "제조업체수"
FROM 제품;
```

03 SQL을 이용한 데이터 조작

❖ 데이터 검색 : SELECT 문

■ 그룹별 검색

예제 7-32

주문 테이블에서 주문제품별 수량의 합계를 검색해보자.

```
▶▶ SELECT  주문제품, SUM(수량) AS 총주문수량
FROM      주문
GROUP BY  주문제품;
```

결과 테이블

	주문제품	총주문수량
1	p03	52
2	p02	58
3	p06	81
4	p04	15
5	p01	24

주문제품	수량
p03	10
p01	5
p06	45
p02	8
p06	36
p01	19
p03	22
p02	50
p04	15
p03	20

	주문제품	총주문수량
1	p03	52
2	p02	58
3	p06	81
4	p04	15
5	p01	24

그림 7-10 주문제품별 수량의 합계를 계산하는 과정

03 SQL을 이용한 데이터 조작

❖ 데이터 검색 : SELECT 문

■ 그룹별 검색

예제 7-34

제품 테이블에서 제품을 3개 이상 제조한 제조업체별로 제품의 개수와, 제품 중 가장 비싼 단가를 검색해보자.

```
▶▶ SELECT  제조업체, COUNT(*) AS 제품수, MAX(단가) AS 최고가
FROM      제품
GROUP BY  제조업체 HAVING COUNT(*)>=3;
```

결과 테이블

제조업체	제품수	최고가
1 한빛제과	3	2600

예제 7-35

고객 테이블에서 적립금 평균이 1,000원 이상인 등급에 대해 등급별 고객수와 적립금 평균을 검색해보자.

```
▶▶ SELECT  등급, COUNT(*) AS 고객수, AVG(적립금) AS 평균적립금
FROM      고객
GROUP BY  등급 HAVING AVG(적립금)>=1000;
```

결과 테이블

등급	고객수	평균적립금
1 gold	3	3500
2 vip	1	2500

03 SQL을 이용한 데이터 조작

❖ 데이터 검색 : SELECT 문

■ 집계 함수를 이용한 검색

예제 7-36

주문 테이블에서 각 주문고객이 주문한 제품의 총주문수량을 주문제품별로 검색해보자.

```
▶▶ SELECT  주문제품, 주문고객, SUM(수량) AS 총주문수량
FROM      주문
GROUP BY  주문제품, 주문고객;
```

결과 테이블

	주문제품	주문고객	총주문수량
1	p02	carrot	8
2	p01	banana	19
3	p06	melon	36
4	p03	apple	32
5	p01	melon	5
6	p02	pear	50
7	p03	carrot	20
8	p06	banana	45
9	p04	banana	15

← 하나의 그룹

(주의) (MySQL)

-- (MySQL) 오류는 나지 않으나 정확한 결과가 아니다

```
SELECT  주문제품, 주문고객, SUM(수량) as 총주문수량
FROM      주문
GROUP BY  주문제품;
```

```
1  -- 주문(주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자)
2
3  -- 예제7-36 주문 테이블에서 각 주문고객이 주문한 제품의 총주문수량을
4  -- 주문제품별로 검색한다
5
6  -- 오류
7  • SELECT  주문제품, SUM(수량) as 총주문수량
8    FROM    주문
9    GROUP BY 주문제품, 주문고객;
```

```
10
11 -- 실행
12 • SELECT  주문제품, 주문고객, SUM(수량) as 총주문수량
13   FROM    주문
14   GROUP BY 주문제품, 주문고객;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

주문 제품	주문 고객	총주문 수량
p01	banana	19
p01	melon	5
p02	carrot	8
p02	pear	50
p03	apple	32
p03	carrot	20
p04	banana	15
p06	banana	45
p06	melon	36