DB프로그래밍

11주차

담당교수: 김희숙
(jasmin11@hanmail.net)

동양미래대학교
DONGYANG MIRAE UNIVERSITY

SQL: 테이블 생성(cascade)

담당교수: 김희숙
(jasmin11@hanmail.net)

[ex08-01]

학생

sno	sname	sdept
11002	김철수	컴퓨터
24036	이상철	정보통신
30419	강정아	컴퓨터

수강과목

sno	eno	ename	egrade
11002	CS401	자료구조	97
11002	CS404	C언어	86
24036	CS401	자료구조	83

1) [수강과목]의 카디날리티(cardinality)와 차수(degree)는 각각 몇인가?
2) [학생]에서 (24036, '이상철', '정보통신') 투플 삭제는 가능한가? 왜?
3) [학생]에서 (30419, '염경섭', '컴퓨터') 투플 입력은 가능한가? 왜?
4) [수강과목]에서 (30419, Λ, 'JAVA', 95) 입력이 가능한가? 왜?
5) [수강과목]에서 (11002, 'CS401', '자료구조', 97) 투플 삭제는 가능한가?
6) [학생]에서 ('AB414', 123, '컴퓨터') 투플 입력은 가능한가? 왜?

[요약] 무결성 제약조건(integrity Constraint)

1) 개체 무결성 제약조건(엔티티 무결성 제약조건)

기본키는 널 값을 가질 수 없다

2) 참조 무결성 제약조건

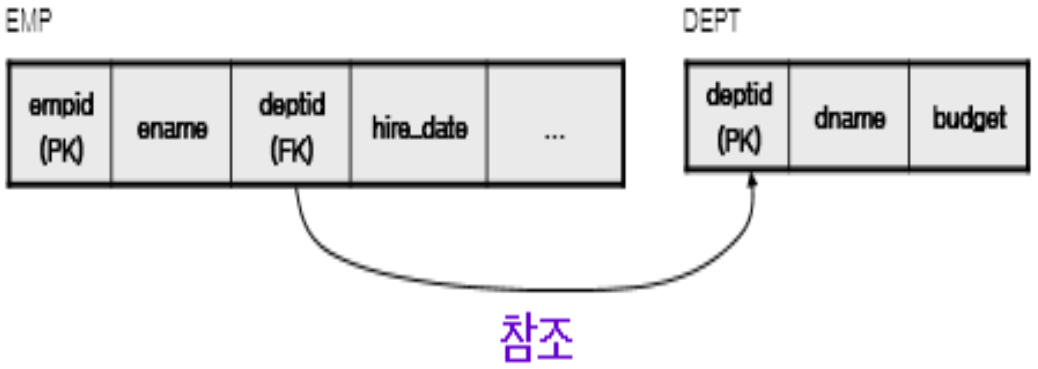
참조하는 테이블의 외래키 값은 참조되는 테이블의 기본키 값에 반드시 존재해야 한다

제약조건	부모 테이블	자식 테이블
입력	**제약 없음**	부모테이블에 데이터가 존재하는지 검증
수정	수정하려는 데이터를 자식테이블에서 참조하고 있는지를 검증	부모테이블에 존재하는 다른 데이터로 변경가능
삭제	삭제하려는 데이터를 자식테이블에서 참조하고 있는지를 검증	**제약 없음**

[요약] 무결성 제약조건

교재 p34

❑ 외래키(Foreign key)



- 두 테이블간에 외래키에 의한 참조관계에 있다면 두테이블간 데이터 불일치가 발생하는 상황이 되면 DBMS는 다음과 같은 조치를 취할 수 있다
  - 제한(restrict), 연쇄(cascade), 널값으로 대체(nullify)

❑ 외래키(Foreign key)

부서 테이블의 첫번째 튜플을 삭제하려 할 때

- 제한(restrict)
  - 삭제하려는 튜플의 부서번호 값을 사원 테이블에서 가지고 있는 튜플이 있으므로 삭제 연산을 거절
- 연쇄(cascade)
  - 삭제된 부서번호 값을 갖는 사원 테이블의 튜플도 함께 삭제
- 널값으로 대체(nullify)
  - 삭제연산을 수행한 뒤 삭제된 부서번호 값을 갖는 사원 테이블의 튜플에서 부서번호를 null 값으로 대체

* 외래키를 통해 두 테이블간의 데이터 무결성을 유지하는 것을 '참조 무결성 제약조건'이라고 한다.

[요약] 무결성 제약조건

2) 참조 무결성 제약조건

참조하는 테이블의 외래키 값은 참조되는 테이블의 기본키 값에 반드시 존재해야 한다

제약조건	부모 테이블	자식 테이블
입력	**제약 없음**	부모테이블에 데이터가 존재하는지 검증
수정	수정하려는 데이터를 자식테이블에서 참조하고 있는지를 검증	부모테이블에 존재하는 다른 데이터로 변경가능
삭제	삭제하려는 데이터를 자식테이블에서 참조하고 있는지를 검증	**제약 없음**

-ON DELETE CASCADE 조건: FOREIGN KEY 에 설정

부모 테이블의 데이터가 삭제되면 자식 테이블의 데이터도 함께 삭제된다

-ON DELETE SET NULL 조건: FOREIGN KEY 에 설정

부모 테이블의 데이터가 삭제되면 자식 테이블의 데이터를 NULL 로 설정한다

Oracle 실습

```
--Oracle 실행
C:₩>sqlplus /nolog

SQL>conn / as sysdba

SQL>create user week08
identified by week08
default tablespace users;

SQL>grant connect, resource to week08;
SQL>conn week08/week08
```

[Tip] 실습 환경
실습한 내용이 그대로 남아 있기 때문에
→다시 실습할 때, 테이블 작성에 어려움이 생길 수 있음
⇒week08 사용자계정을 임시 사용자계정으로 활용

```
REM 사용자 생성
CREATE USER week08
IDENTIFIED BY week08
DEFAULT TABLESPACE users;
```

[Tip] 실습 환경
실습한 후에, week08 사용자계정 삭제
(즉, week08 사용자계정을 임시 사용자계정으로 활용)

```
REM 사용자 삭제
DROP USER week08 CASCADE;
```

테이블 생성(restricted)

학생

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

학과

학과코드	학과명
1001	컴퓨터학과
2001	체육학과

```
--학과(학과코드, 학과명)
--학생(학번, 이름, 학과코드)
CREATE TABLE 학과 (
        학과코드          char(4)     NOT NULL ,
        학과명            varchar(30) ,
        CONSTRAINT pk_학과_학과코드  PRIMARY KEY(학과코드)
);
CREATE TABLE 학생 (
        학번          char(3)     NOT NULL  ,
        이름          varchar(10) ,
        학과코드      char(4) ,
        CONSTRAINT pk_학생_학번    PRIMARY KEY(학번) ,
        CONSTRAINT fk_학생_학과코드 FOREIGN KEY (학과코드)
                                    REFERENCES 학과(학과코드)
);
```

* 엔티티 무결성 제약조건

--오류 (이유는?)

```
insert into 학과 values('1001', '컴퓨터학과');
```

* 참조무결성 제약조건

--오류 (이유는?)

```
insert into 학생 values('601','박세리','3001');
```

1) 제한(restricted)

학생

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

학과

학과코드	학과명
1001	컴퓨터학과
2001	체육학과

```
--학과(학과코드, 학과명)
--학생(학번, 이름, 학과코드)
CREATE TABLE 학과 (
        학과코드          char(4)     NOT NULL ,
        학과명            varchar(30) ,
        CONSTRAINT pk_학과_학과코드  PRIMARY KEY(학과코드)
);
CREATE TABLE 학생 (
        학번          char(3)     NOT NULL  ,
        이름          varchar(10) ,
        학과코드      char(4) ,
        CONSTRAINT pk_학생_학번    PRIMARY KEY(학번) ,
        CONSTRAINT fk_학생_학과코드 FOREIGN KEY (학과코드)
                                    REFERENCES 학과(학과코드)
);
```

```
--1) 제한(RECTRICT) 인 경우
--오류 (이유는?)
UPDATE   학과
SET      학과코드 = 'A001'
WHERE    학과코드 = '1001';
--오류 (이유는?)
DELETE
FROM     학과
WHERE    학과코드 = '2001';
```

2) 연쇄(cascade)

```
CREATE TABLE 학과2 (
        학과코드          char(4)      NOT NULL ,
        학과명            varchar(30) ,
        CONSTRAINT pk_학과2_학과코드   PRIMARY KEY(학과코드)
);
CREATE TABLE 학생2 (
        학번              char(3)      NOT NULL   ,
        이름              varchar(10) ,
        학과코드          char(4) ,
        CONSTRAINT pk_학생2_학번    PRIMARY KEY(학번) ,
        CONSTRAINT fk_학생2_학과코드 FOREIGN KEY (학과코드)
                                          REFERENCES 학과2(학과코드)
                                          ON DELETE CASCADE
);
```

학생

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

학과

학과코드	학과명
1001	컴퓨터학과
2001	체육학과

```
--2) 연쇄(CASCADE) 인 경우
--오류 (이유는?)
UPDATE   학과2
SET      학과코드 = 'A001'
WHERE    학과코드 = '1001';
--실행결과는?
DELETE
FROM     학과2
WHERE    학과코드 = '2001';
```

3) 널 값(set null)

```
CREATE TABLE 학과3 (
        학과코드          char(4)     NOT NULL ,
        학과명            varchar(30) ,
        CONSTRAINT pk_학과3_학과코드   PRIMARY KEY(학과코드)
);
CREATE TABLE 학생3 (
        학번              char(3)     NOT NULL  ,
        이름              varchar(10) ,
        학과코드          char(4) ,
        CONSTRAINT pk_학생3_학번    PRIMARY KEY(학번) ,
        CONSTRAINT fk_학생3_학과코드 FOREIGN KEY (학과코드)
                                        REFERENCES 학과3(학과코드)
                                        ON DELETE SET NULL
);
```

학생

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

학과

학과코드	학과명
1001	컴퓨터학과
2001	체육학과

```
--3) 널 값(SET NULL) 인 경우
--오류 (이유는?)
UPDATE   학과3
SET      학과코드 = 'A001'
WHERE    학과코드 = '1001';
--실행결과는?
DELETE
FROM     학과3
WHERE    학과코드 = '2001';
```

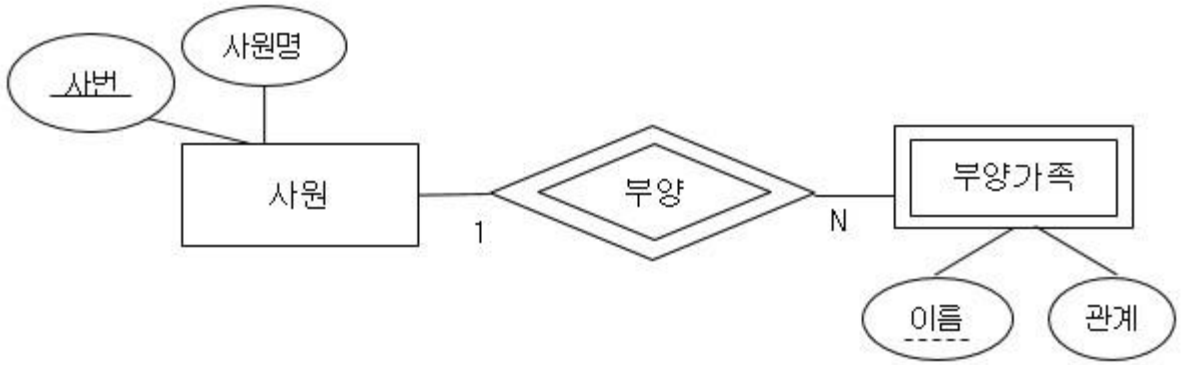
약 엔티티

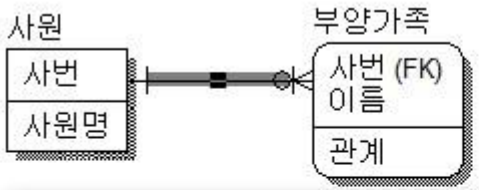
사원

사번	사원명
100	김
200	홍
300	박
400	최

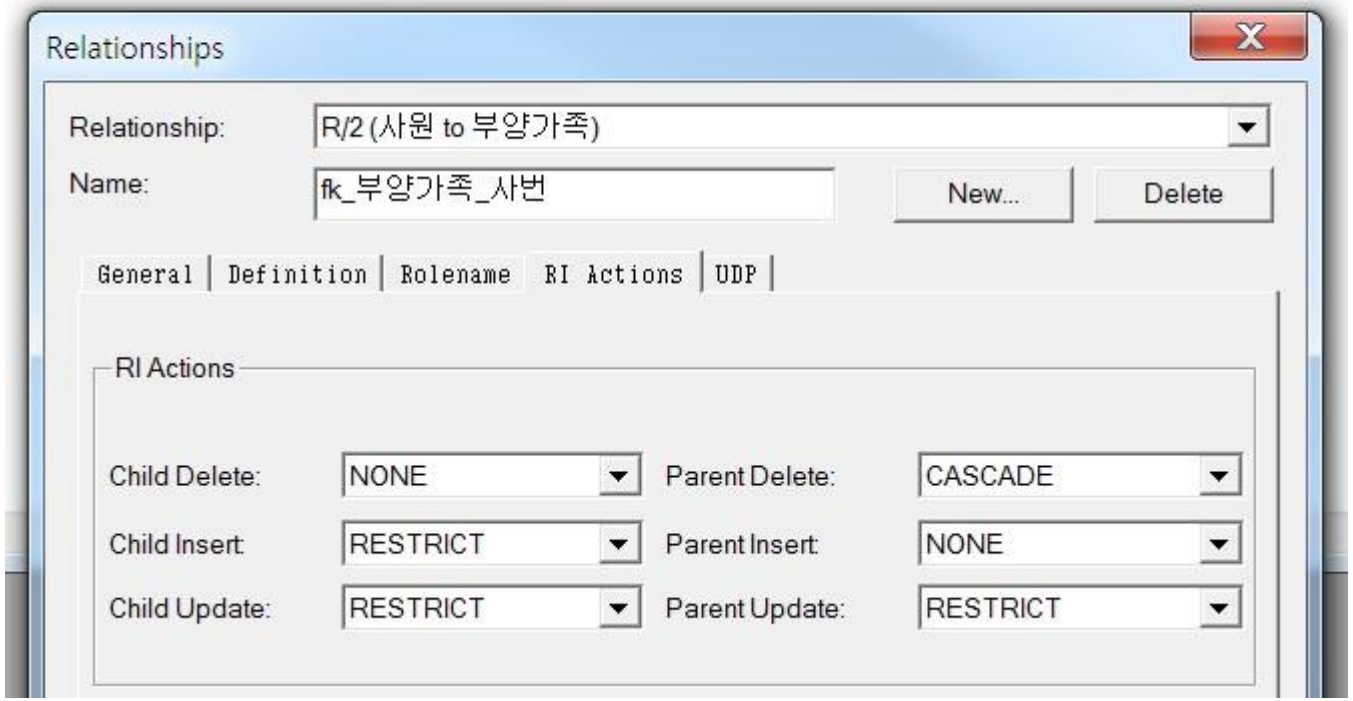
부양가족

사번	이름	관계
100	이다원	장남
100	전준원	차남
100	김태영	장녀
400	이호은	손자






```
-- 사원 100 번 퇴사한 경우
delete
from 사원
where 사번=100;

select * from 사원;
select * from 부양가족;
```



[Quiz08-01]

67. 아래의 릴레이션 T, S, R이 각각 다음과 같이 선언되었다.

```
CREATE TABLE T
(C INTEGER PRIMARY KEY,
 D INTEGER);

CREATE TABLE S
(B INTEGER PRIMARY KEY,
 C INTEGER REFERENCES T(C) ON DELETE CASCADE);

CREATE TABLE R
(A INTEGER PRIMARY KEY,
 B INTEGER REFERENCES S(B) ON DELETE SET NULL);
```

현재의 릴레이션 T, S, R의 상태는 다음과 같다.

T :

C	D
1	1
2	1

S :

B	C
1	1
2	1

R :

A	B
1	1
2	2

DELETE FROM T;
를 수행한 후에 릴레이션 R에는 어떤 투플들만 들어 있겠는가?

① (1, NULL)과 (2, 2)
② (1, NULL)과 (2, NULL)
③ (2, 2)
④ 아무 투플도 안 들어 있음

```
drop table r;
drop table s;
drop table t;
```

T

C	D
1	1
2	1

S

B	C
1	1
2	1

R

A	B
1	1
2	2

```
--3개의 테이블 T, S, R 각각 작성하는 CREATE TABLE 문법을 이 위치에 입력하세요
insert into T values(1,1);
insert into T values(2,1);

insert into S values(1,1);
insert into S values(2,1);

insert into R values(1,1);
insert into R values(2,2);
--다음 문법을 수행한 이후에 릴레이션 R 에는 어떤 투플들이 있는가?
DELETE FROM T;
select * from R;
```

SQL: 뷰(view)

담당교수: 김희숙
(jasmin11@hanmail.net)

[Quiz08-02]

아래 CREATE TABLE 문으로 릴레이션 R을 생성하였다

```
CREATE TABLE R (
  NAME    VARCHAR(20) PRIMARY KEY,
  SALARY  INTEGER       CHECK(SALARY <= 4000)
);
```

릴레이션 R의 현재 내용은 다음과 같다.

<KIM, 1000>
<LEE, 2000>
<PARK,3000>

이 릴레이션 R에 대하여 아래의 순서로 SQL문을 수행하였다.
이들 중에서 일부는 릴레이션에 정의된 제약조건 때문에
거절될 수 있다.

```
INSERT INTO R VALUES('CHOI',1200);

UPDATE R SET SALARY=5000 WHERE NAME='PARK';

INSERT INTO R VALUES('KIM',1300);

DELETE FROM R WHERE NAME='LEE';
```

릴레이션 R 에 들어 있는
모든 투플(Tuple) 들의 SALARY 의 합은 얼마인가?

1.4 뷰(view)

교재 p35~38

- 뷰의 필요성

EMP

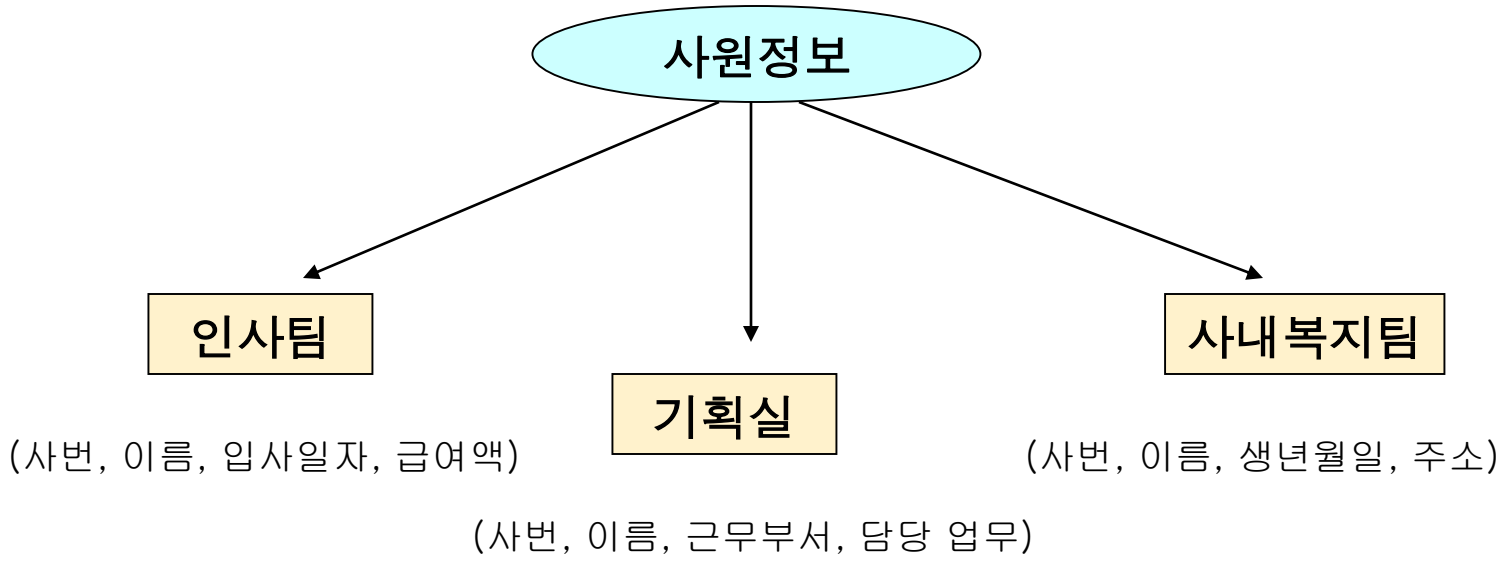
empid	ename	dept	hire_date	birthday	Address	job	salary
1001	홍성길	영업부	2001.2.1	1985.10.12	서울 대림동	특수영업	350
1002	곽희준	영업부	1999.1.1	1984.9.10	안양 용봉동	영업관리	400
1003	김동준	생산부	2000.9.1	1986.5.16	부산 대하동	품질관리	300
1004	성재규	인사부	1997.2.1	1982.4.10	대구 달성동	급여	450
1005	박성범	구매부	2000.2.1	1986.12.4	광주 금남동	수입자재	320

<그림 1.16> 전체 조직 관점에서의 사원 테이블

1.4 뷰(view)

교재 p35~38

- 뷰의 필요성
  - 하나의 테이블, 혹은 여러 테이블에 대하여 특정 사용자나 조직의 관점에서 데이터를 바라볼 수 있도록 해주는 수단



1.4 뷰(view)

교재 p35~38

VIEW_EMP1

empid	ename	hire_date	salary
1001	홍성길	2001.2.1	350
1002	곽희준	1999.1.1	400
1003	김동준	2000.9.1	300
1004	성재규	1997.2.1	450
1005	박성범	2000.2.1	320

VIEW_EMP2

empid	ename	dept	job
1001	홍성길	영업부	특수영업
1002	곽희준	영업부	영업관리
1003	김동준	생산부	품질관리
1004	성재규	인사부	급여
1005	박성범	구매부	수입자재

VIEW_EMP3

empid	ename	birthday	Address
1001	홍성길	1985.10.12	서울 대림동
1002	곽희준	1984.9.10	안양 용봉동
1003	김동준	1986.5.16	부산 대하동
1004	성재규	1982.4.10	대구 달성동
1005	박성범	1986.12.4	광주 금남동

<그림 1.17> 사원 테이블에 대한 세가지 뷰

[ex08-02] view

-- (Oracle)
SQL>grant create view to week08;

VIEW_EMP1

empid	ename	hire_date	salary
1001	홍성길	2001.2.1	350
1002	곽희준	1999.1.1	400
1003	김동준	2000.9.1	300
1004	성재규	1997.2.1	450
1005	박성범	2000.2.1	320

VIEW_EMP2

empid	ename	dept	job
1001	홍성길	영업부	특수영업
1002	곽희준	영업부	영업관리
1003	김동준	생산부	품질관리
1004	성재규	인사부	급여
1005	박성범	구매부	수입자재

VIEW_EMP3

empid	ename	birthday	Address
1001	홍성길	1985.10.12	서울 대림동
1002	곽희준	1984.9.10	안양 용봉동
1003	김동준	1986.5.16	부산 대하동
1004	성재규	1982.4.10	대구 달성동
1005	박성범	1986.12.4	광주 금남동

```
CREATE VIEW view_emp1
as
   SELECT empid, ename, hire_date, salary
   FROM   emp;

select * from view_emp1;
```

스크립트 출력 | 질의 결과 | 작업이 완료되었습니다.(0.021초)

```
명령의 3 행에서 시작하는 중 오류 발생 -
CREATE VIEW view_emp1
as
   SELECT empid, ename, hire_date, salary
   FROM   emp
오류 보고 -
ORA-01031: insufficient privileges
01031. 00000 -  "insufficient privileges"
*Cause:    An attempt was made to perform a database operation without
           the necessary privileges.
```

```
SQL> conn / as sysdba
Connected.
SQL>
SQL> grant create view to week08;

Grant succeeded.
```

```
select * from emp;
-- 뷰 생성
CREATE VIEW view_emp1
as
   SELECT empid, ename, hire_date, salary
   FROM   emp;
-- 뷰 조회
select * from view_emp1;
```

[ex08-02] view

-- (Oracle)
SQL>grant create view to week08;

VIEW_EMP1

empid	ename	hire_date	salary
1001	홍성길	2001.2.1	350
1002	곽희준	1999.1.1	400
1003	김동준	2000.9.1	300
1004	성재규	1997.2.1	450
1005	박성범	2000.2.1	320

```
CREATE VIEW view_emp1
as
   SELECT empid, ename, hire_date, salary
   FROM   emp;

select * from view_emp1;
```

스크립트 출력 | 질의 결과 | 질의 결과 1

SQL | 인출된 모든 행: 5(0.003초)

	EMPID	ENAME	HIRE_DATE	SALARY
1	1001	홍성길	2001-02-01	350
2	1002	곽희준	1999-01-01	400
3	1003	김동준	2000-09-01	300
4	1004	성재규	1997-02-01	450
5	1005	박성범	2000-02-01	320

```
CREATE VIEW view_emp1
as
   SELECT empid, ename, hire_date, salary
   FROM   emp;

select * from view_emp1;
```

스크립트 출력 | 질의 결과 | 작업이 완료되었습니다.(0.021초)

```
명령의 3 행에서 시작하는 중 오류 발생 -
CREATE VIEW view_emp1
as
   SELECT empid, ename, hire_date, salary
   FROM   emp
오류 보고 -
ORA-01031: insufficient privileges
01031. 00000 -  "insufficient privileges"
*Cause:    An attempt was made to perform a database operation without
           the necessary privileges.
```

```
SQL> conn / as sysdba
Connected.
SQL>
SQL> grant create view to week08;

Grant succeeded.
```

select * from emp;
-- 뷰 생성
CREATE VIEW view_emp1
as
SELECT empid, ename, hire_date, salary
FROM emp;
-- 뷰 조회
select * from view_emp1;

[ex08-02] view

VIEW_EMP1

empid	ename	hire_date	salary
1001	홍성길	2001.2.1	350
1002	곽희준	1999.1.1	400
1003	김동준	2000.9.1	300
1004	성재규	1997.2.1	450
1005	박성범	2000.2.1	320

```
-- 뷰 입력
--view_emp1(empid,ename,hire_date,salary)
insert into view_emp1
values('9999','김복기','2020-03-16',500);
-- 뷰 조회
select * from view_emp1;
-- 테이블 조회
select * from emp;
```

```
insert into view_emp1 values('9999','김복기','2020-03-16',500);

-- 뷰 조회
select * from view_emp1;
```

스크립트 출력 | 질의 결과 | 질의 결과 1

SQL | 인출된 모든 행: 6(0.002초)

	EMPID	ENAME	HIRE_DATE	SALARY
1	1001	홍성길	2001-02-01	350
2	1002	곽희준	1999-01-01	400
3	1003	김동준	2000-09-01	300
4	1004	성재규	1997-02-01	450
5	1005	박성범	2000-02-01	320
6	9999	김복기	2020-03-16	500

```
-- 뷰 입력
--view_emp1(empid,ename,hire_date,salary)

insert into view_emp1 values('9999','김복기','2020-03-16',500);

-- 테이블 조회
select * from emp;
```

스크립트 출력 | 질의 결과 | 질의 결과 1

SQL | 인출된 모든 행: 6(0.003초)

	EMPID	ENAME	DEPT	HIRE_DATE	BIRTHDAY	ADDRESS	JOB	SALARY
1	1001	홍성길	영업부	2001-02-01	1985-10-12	서울 대림동	특수영업	350
2	1002	곽희준	영업부	1999-01-01	1984-09-10	안양 용봉동	영업관리	400
3	1003	김동준	생산부	2000-09-01	1986-05-16	부산 대하동	품질관리	300
4	1004	성재규	인사부	1997-02-01	1982-04-10	대구 달성동	급여	450
5	1005	박성범	구매부	2000-02-01	1986-12-04	광주 금남동	수입자재	320
6	9999	김복기	(null)	2020-03-16	(null)	(null)	(null)	500

1.4 뷰(view)

교재 p35~38

- 뷰의 정의

```
CREATE VIEW high_salary
 AS SELECT empid, ename, dept, salary
    FROM   emp
    WHERE  salary >= 350;
```

  - 뷰에 대한 질의는 뷰가 정의된 기본 테이블에 대한 질의로 바뀌어 실행된다

```
SELECT empid, ename, salary
FROM   high_salary
WHERE dept = '영업부';
```

```
SELECT empid, ename, salary
FROM   emp
WHERE salary >= 350
AND    dept = '영업부';
```

1.4 뷰(view)

교재
p35~38

- 뷰를 사용하는 경우
  - <그림 1.17>의 경우와 같이 하나의 테이블에 대하여 여러 부서에서 서로 다른 관점으로 보기를 원할 때
  - 테이블에 급여와 같이 일반 사용자에게는 감추어야 할 컬럼이 있을 때 그것을 제외하고 뷰를 만들어 제공함으로써 보안을 유지.
  - 자주 사용하는 복잡한 질의문을 미리 뷰로 정의하여 두고 간편하게 쓰고자 할 때

• 뷰는 물리적인 데이터를 갖지 않는다
• 뷰에 대한 갱신 연산은 경우에 따라 실행될수도 있고 실행되지 않을수도 있다

[요약] 뷰(view)

❖뷰(view): 가상의 테이블

실제 물리적으로 존재하지 않는다

뷰는 스키마

뷰 데이터가 테이블에 별도로 저장되지 않는다.

❖뷰 장점

1. 보안
2. 사용의 편의성

❖ 뷰 입력, 삭제 불가능 한 경우,

1. 조인으로 작성된 뷰
2. 그룹함수 사용한 경우

```
-- (Oracle)
SQL>grant create view to week08;
```

SQL: SELECT_기본

담당교수: 김희숙
(jasmin11@hanmail.net)

```
--영화(영화코드, 영화제목, 장르, 관람기준, 상영시간)
--배우(배우번호, 배우이름, 성별)
--출연(일련번호, 영화코드, 배우번호, 출연료)

create table 영화 (
   영화코드   char(4)        NOT NULL PRIMARY KEY ,
   영화제목   varchar(50)     NOT NULL ,
   장르       varchar(50) ,
   관람기준   int ,
   상영시간   int
)
insert into 영화 values('MC01','과속스캔들','코미디',12,108);
insert into 영화 values('MC02','엽기적인그녀','코미디',15,122);
insert into 영화 values('MD01','키다리아저씨','드라마',12,110);
insert into 영화 values('MD02','바보','드라마',12,99);
```

```
--영화(영화코드, 영화제목, 장르, 관람기준, 상영시간)
--배우(배우번호, 배우이름, 성별)
--출연(일련번호, 영화코드, 배우번호, 출연료)

create table 배우 (
   배우번호     char(4) NOT NULL PRIMARY KEY ,
   배우이름     varchar(8) NOT NULL ,
   성별         char(3)
);

insert into 배우 values('a001','차태현','남');
insert into 배우 values('a002','하지원','여');
```

[실습] [ex08-03]


```
--영화(영화코드, 영화제목, 장르, 관람기준, 상영시간)
--배우(배우번호, 배우이름, 성별)
--출연(일련번호, 영화코드, 배우번호, 출연료)
create table 출연 (
        일련번호 int ,
        영화코드 char(4) NOT NULL ,
        배우번호 char(4) NOT NULL ,
        출연료    int ,
        CONSTRAINT PK_출연_일련번호 PRIMARY KEY(일련번호) ,
        FOREIGN KEY(영화코드) REFERENCES 영화코드(번호) ,
        FOREIGN KEY(배우번호) REFERENCES 배우번호(번호) ,
);
insert into 출연 values('MC01','a001',1000);
insert into 출연 values('MC02','a001',700);
insert into 출연 values('MD01','a002',400);
insert into 출연 values('MD02','a001',600);
insert into 출연 values('MD02','a002',600);
```

[ex08-03]

--영화(영화코드, 영화제목, 장르, 관람기준, 상영시간)
--배우(배우번호, 배우이름, 성별)
--출연(일련번호, 영화코드, 배우번호, 출연료)

-- 1) 장르가 코미디 이거나 드라마 인 영화제목, 장르를 검색하라(IN 사용)
-- 2) 장르별 최고 상영시간을 검색하라
-- 3) 영화제목에 '아저씨'가 들어가는 모든 정보 검색하라
-- 4) 성별이 남인 배우번호, 배우이름을 배우이름의 오름차순으로 검색하라
-- 5) 출연료가 500에서 1000인 영화코드, 배우번호, 출연료 검색하라

-- 6) 다음 SQL문법의 실행결과를 작성하고 간략히 설명

```
SELECT COUNT(장르), COUNT(DISTINCT 장르)
FROM    영화
```

--7) 차태현이 출연한 영화의 출연료를 검색하라

--8) 영화 '키다리아저씨'의 장르를 '스릴러', 관람기준을 15 로 수정하라

[Quiz08-03]

차량

사원번호	종류
23	A
25	B
43	C
56	D

인사

성명	소속	사원번호
김이순	총무과	25
박이준	자재과	56
이형수	영업과	23
오영우	보육과	43

--다음 문법을 수행한 이후의 실행결과는?

```
-- 1) 성명, 소속, 차량종류를 검색하라
SELECT 성명, 소속, 종류 as 차량종류
FROM   차량, 인사
WHERE  차량.사원번호 = 인사.사원번호;

-- 2) 박이준의 성명, 소속, 차량종류를 검색하라
SELECT 성명, 소속, 종류 as 차량종류
FROM   차량, 인사
WHERE  차량.사원번호 = 인사.사원번호
  AND  성명 = '박이준';
```

[Quiz08-04]

61. 아래의 [인사] 테이블과 [차량] 테이블을 이용하여 SQL문을 수행했을 경우의 결과는?

```
select 종류
from 차량
where 사원번호 = (select 사원번호 from 인사 where 성명=오영우);
```

[인사]

성명	소속	사원번호
김이순	총무과	25
박이준	자재과	56
이형수	영업과	23
오영우	보육과	43

[차량]

사원번호	종류
23	A
25	B
43	C
56	D

가. 43　　나. 56

다. C　　라. D

--다음 문법을 수행한 이후의 실행결과는?

```
select  종류
from    차량
where   사원번호 = (select 사원번호
                    from 인사
                    where 성명='오영우');

-- 인 라인 뷰(in-line view)

-- 이름 중에 '이' 가 들어가는 사원의 성명, 소속과
-- 차량 종류를 검색하라
select  t.성명, t.소속, 종류
from    (select 성명, 소속, 사원번호
         from 인사
         where 성명 LIKE '%이%')  t, 차량
where   t.사원번호 = 차량.사원번호;
```

[Quiz08-05]

환자

환자번호	이름	질병코드	나이
P1001	김철수	A01	30
P1002	양길현	A03	29
P1003	임영수	A01	50
Q1001	박한나	A04	40

질병

질병코드	질병명	증상
A01	뇌졸증	어지럼증
A02	콜레라	설사
A03	기관지염	발열
A04	장티푸스	발열

--1) 증상이 발열인 질병코드와 질병명, 증상을 조회하라
--2) 양길현의 이름과 환자번호, 나이를 조회하라
--3) 나이가 30 이하인 환자번호와 이름, 나이를 조회하라
--4) 환자 김철수의 환자 이름, 질병명, 증상을 조회하라

--5) 각 환자의 이름과 질병명을 환자 이름의 오름차순으로 조회하라

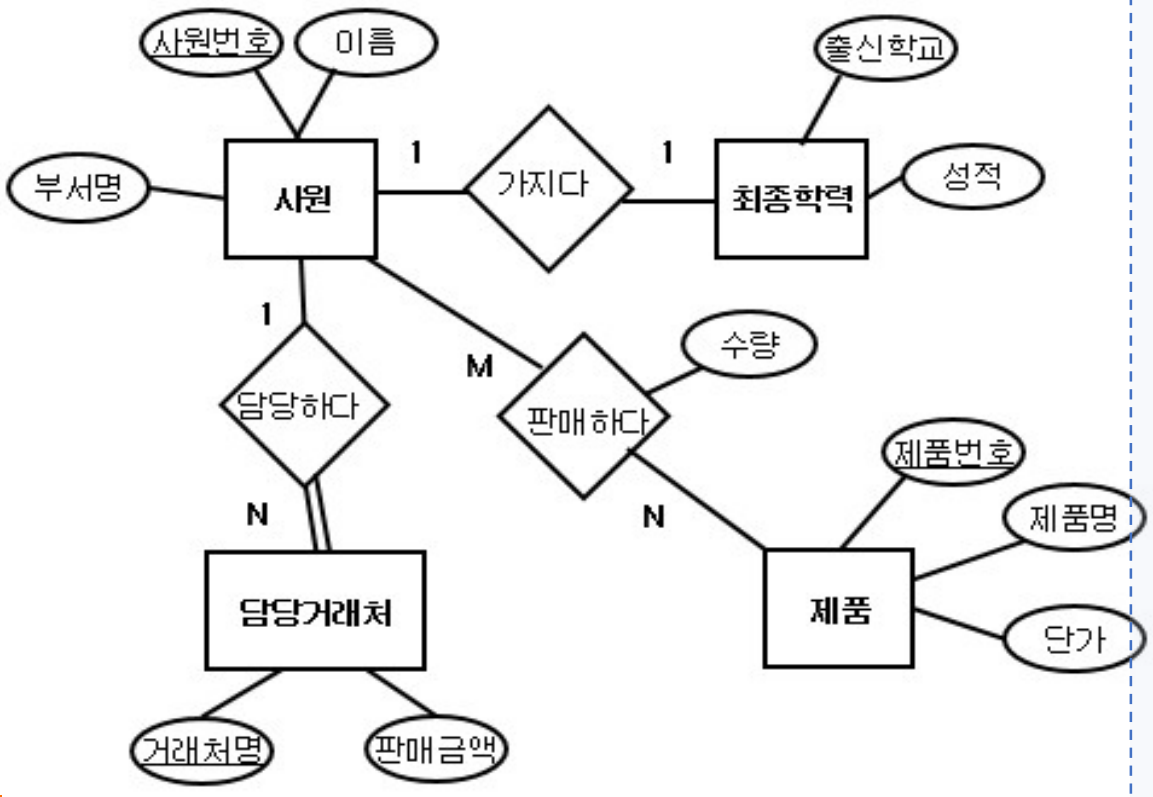
--6) 질병의 종류 중 현재 환자테이블에 있는 환자가 걸리지 않은 질병을 조회하라

```
select 질병코드, 질병명, 증상
from  질병 D
where NOT EXISTS (select *
                  from 환자 P
                  where P.질병코드 = D.질병코드)
```

SQL: SELECT_조인

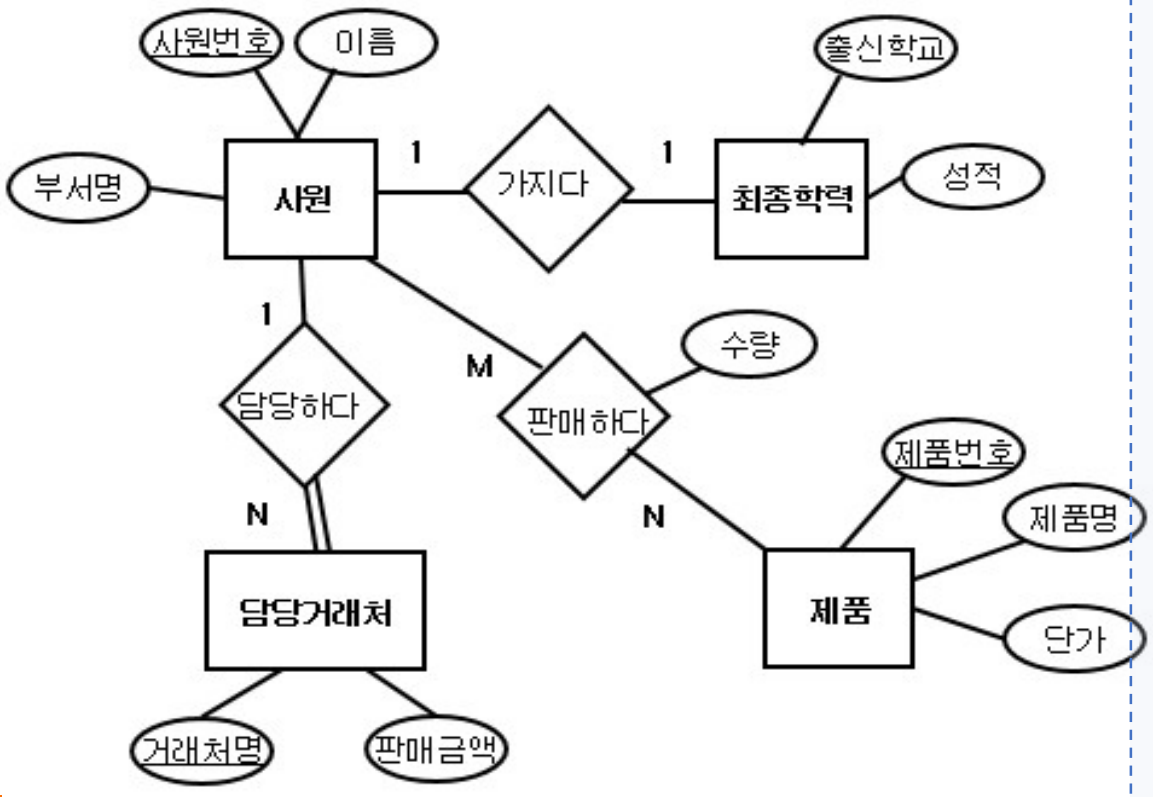
담당교수: 김희숙
(jasmin11@hanmail.net)

[ex08-05]



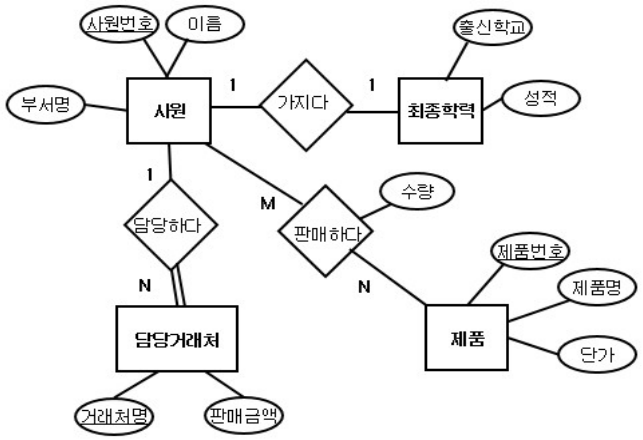
사원은 사원번호로 식별되며, 이름, 부서명을 가진다
사원의 최종학력에는 출신학교, 성적을 저장한다
각 사원마다 한 개의 최종학력을 저장하고,
각 최종학력은 한 사원에 소속된다
담당거래처에는 거래처명(식별), 판매금액을 가진다
각 사원은 한 개 이상의 담당거래처를 담당한다.
사원은 담당하는 거래처가 없을 수도 있다 (선택)
각 담당거래처는 담당하는 사원이 한 명 있고, 반드시 담당하는 사원이 있다 (필수)
제품에는 유일하게 식별할 수 있는 제품번호와 제품명, 단가에 대한 정보가 있다
각 사원은 여러 제품을 판매할 수 있고, 각 제품은 여러 명의 사원이 판매할 수 있다
제품에는 수량을 저장한다

[ex08-05]




```
insert into 사원 values(1,'김하나','영업1과');
insert into 사원 values(2,'이두한','영업2과');
insert into 사원 values(3,'박태성','영업3과');

insert into 최종학력 values(1,'한국대학교','B0');
insert into 최종학력 values(2,'신한고등학교','90');
insert into 최종학력 values(3,'제일대학교','B+');

insert into 담당거래처 values('하나로주식회사',24583500,1);
insert into 담당거래처 values('신영주식회사',35416200,2);
insert into 담당거래처 values('혜성실업',13678200,1);

insert into 제품 values('PD01','RS-101',21500);
insert into 제품 values('PD02','RS-102',18700);
insert into 제품 values('PD03','RS-103',23100);

insert into 판매 values(1,'PD01',500);
insert into 판매 values(2,'PD03',487);
insert into 판매 values(1,'PD03',95);
```

[ex08-05]



사원번호	이름	부서명
1	김하나	영업1과
2	이두한	영업2과
3	박태성	영업3과

제품번호	제품명	단가
PD01	RS-101	21500
PD02	RS-102	18700
PD03	RS-103	23100

사원번호	제품번호	수량
1	PD01	500
1	PD03	95
2	PD03	487

```
--사원(사원번호, 이름, 부서명)
--제품(제품번호, 제품명, 단가)
--판매(사원번호, 제품번호, 수량)

--(3개 테이블 조인)
--제품을 판매한 사원번호, 사원명, 제품번호, 제품명, 수량을 검색하라

--1) SELECT ... FROM ... WHERE 절 사용
SELECT 사원.사원번호, 이름, 제품.제품번호, 제품명, 수량
FROM   사원, 판매, 제품
WHERE  사원.사원번호 = 판매.사원번호
 AND   판매.제품번호 = 제품.제품번호;

--2) SELECT ... FROM ... ON 절 사용
SELECT 사원.사원번호, 이름, 제품.제품번호, 제품명, 수량
FROM   사원 INNER JOIN 판매
  ON   사원.사원번호 = 판매.사원번호
     INNER JOIN 제품
  ON   판매.제품번호 = 제품.제품번호;
```

재귀적 관계

11주차 02-02

담당교수: 김희숙
(jasmin11@hanmail.net)

관계의 종류

❖관계

1. 관계 차수

   1진 관계, 2진 관계, 3진 관계

2. 관계 카디날리티

   일대일 관계(1:1)

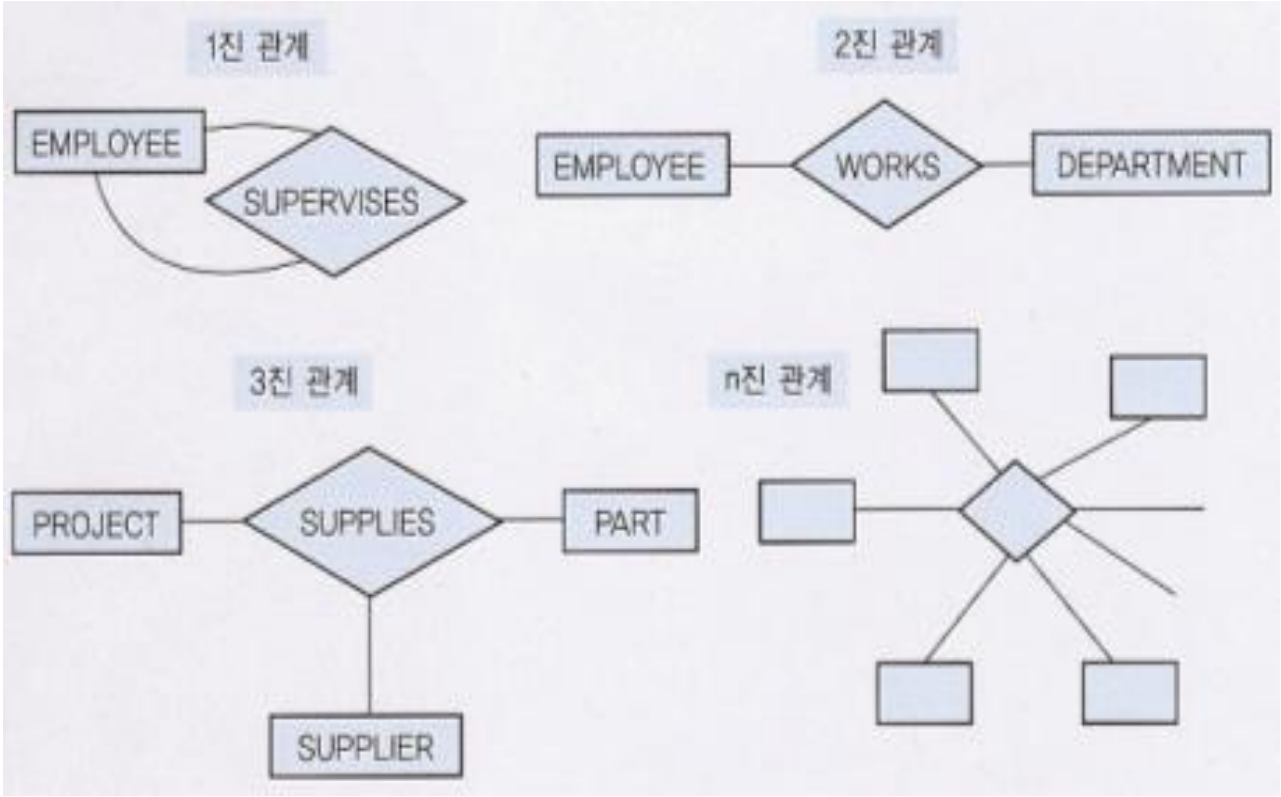
   일대다 관계(1:N)

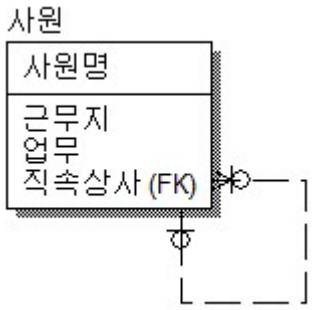
   다대다 관계(M:N)

3. 관계 존재성

   필수(Mandatory)

   선택(Optional)





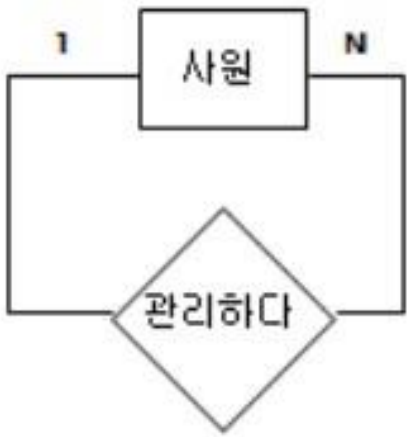
사원

fk

사번	사원명	근무지	업무	직속상사
21001	김철수	서울	회계	
21002	양길현	인천	총무	21001
21003	임영수	부산	영업	21001
21004	박한나	부산	영업	21004

재귀적 관계

❖ 자기자신과 관계를 맺는 것



사원 E

```
select * from 사원
```

	사원번호	이름	주민번호	관리사번
1	1	박찬호	720201-1034343	NULL
2	2	선동열	680709-1078656	1
3	3	임꺽정	700101-1027362	2
4	4	차범근	600809-1987766	3
5	5	홍길동	651214-1078767	4

사원 M

```
select * from 사원
```

	사원번호	이름	주민번호	관리사번
1	1	박찬호	720201-1034343	NULL
2	2	선동열	680709-1078656	1
3	3	임꺽정	700101-1027362	2
4	4	차범근	600809-1987766	3
5	5	홍길동	651214-1078767	4

--자체조인 문법

```
SELECT E.이름 as '사원명',
       M.이름 as '관리자명'
FROM   사원 E, 사원 M
WHERE  E.관리사번 = M.사원번호
```

<실행결과>

	사원명	관리자명
1	선동열	박찬호
2	임꺽정	선동열
3	차범근	임꺽정
4	홍길동	차범근

재귀적 관계

* 스키마 표기법

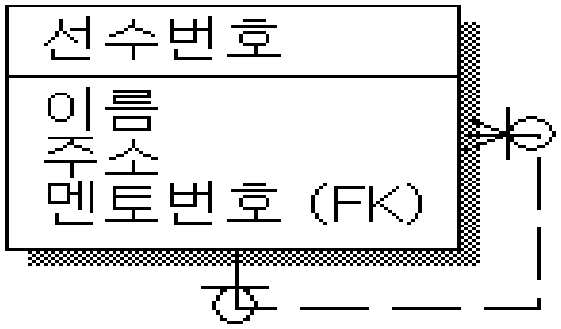
--멘토(선수번호,이름,주소,멘토번호(FK))

참조

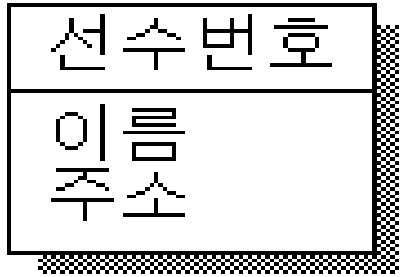
기본키 외래키

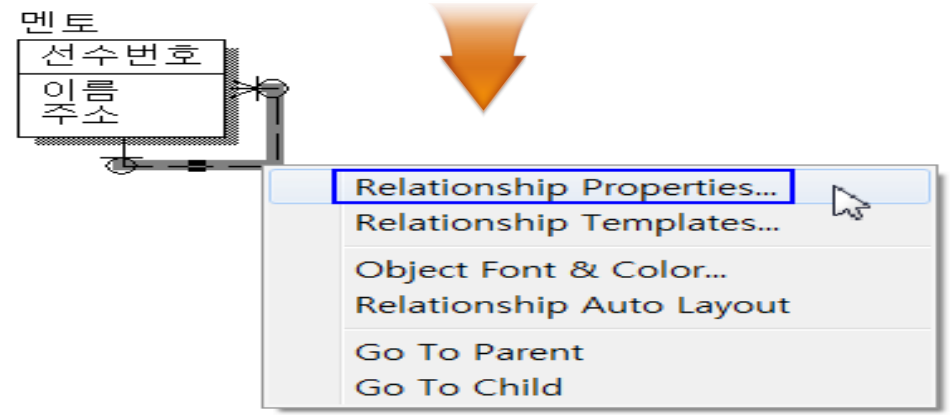
선수번호	이름	주소	멘토번호
1	박지성	영국 맨체스타	NULL
2	김연아	대한민국 서울	3
3	장미란	대한민국 강원도	4
4	주신수	미국 클리블랜드	NULL

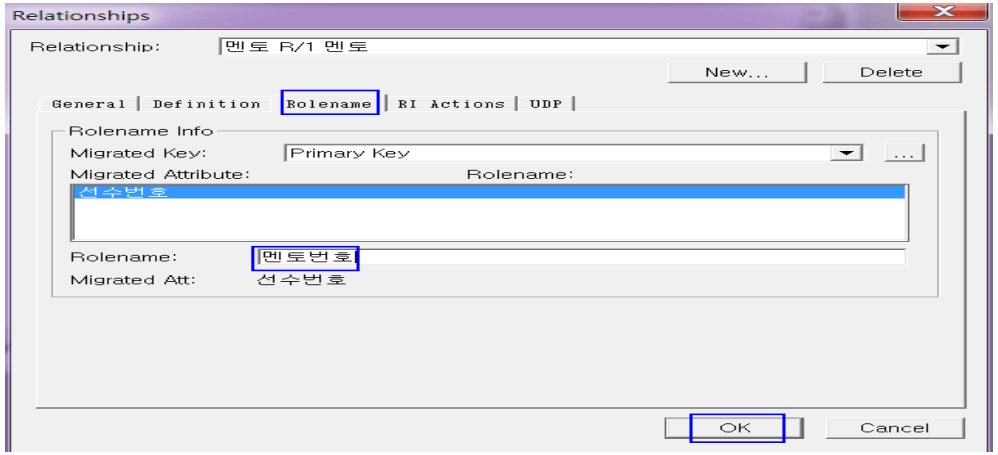
멘토



멘토







재귀적 관계

* 스키마 표기법

--멘토(선수번호,이름,주소,멘토번호(FK))

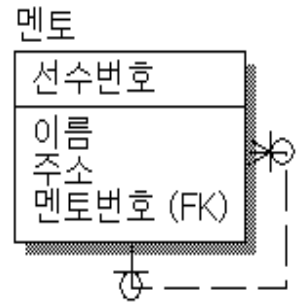
참조

기본키 / 외래키

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(선수번호, 이름, 주소, 멘토번호)

insert into 멘토 values(1,'박지성','영국 멘체스타',NULL);
insert into 멘토 values(4,'추신수','미국 클리블랜드',NULL);

insert into 멘토 values(3,'장미란','대한민국 강원도',4);
insert into 멘토 values(2,'김연아','대한민국 서울',3);

select * from 멘토;

--형식 명령어
COL 선수번호 FOR 999
COL 이름 FOR a10
COL 주소 FOR a20
COL 멘토번호 FOR 999

select * from 멘토;
```

```
선수번호 이름       주소                 멘토번호
-------- ---------- -------------------- --------
       1 박지성     영국 멘체스타
       4 추신수     미국 클리블랜드
       3 장미란     대한민국 강원도             4
       2 김연아     대한민국 서울               3
```



자체조인(self-jon): 멘토 테이블에서 선수이름, 멘토이름을 검색하라

```
select  M1.이름 "선수이름", M2.이름 "멘토이름"
from    멘토 M1, 멘토 M2
where   M1.멘토번호 = M2.선수번호;
```

멘토 M1

```
선수번호 이름       주소                 멘토번호
-------- ---------- -------------------- --------
       1 박지성     영국 멘체스타
       4 추신수     미국 클리블랜드
       3 장미란     대한민국 강원도             4
       2 김연아     대한민국 서울               3
```

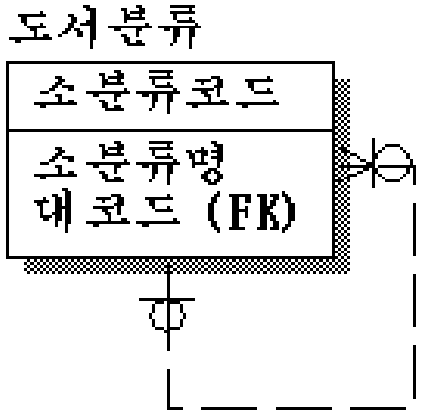
멘토 M2

```
선수번호 이름       주소                 멘토번호
-------- ---------- -------------------- --------
       1 박지성     영국 멘체스타
       4 추신수     미국 클리블랜드
       3 장미란     대한민국 강원도             4
       2 김연아     대한민국 서울               3
```

```
선수이름 멘토이름
-------- --------
장미란   추신수
김연아   장미란
```

관계(1진 관계: 재귀적관계)

도서분류 : 재귀적 관계 → 계층적 질의 (Oracle)

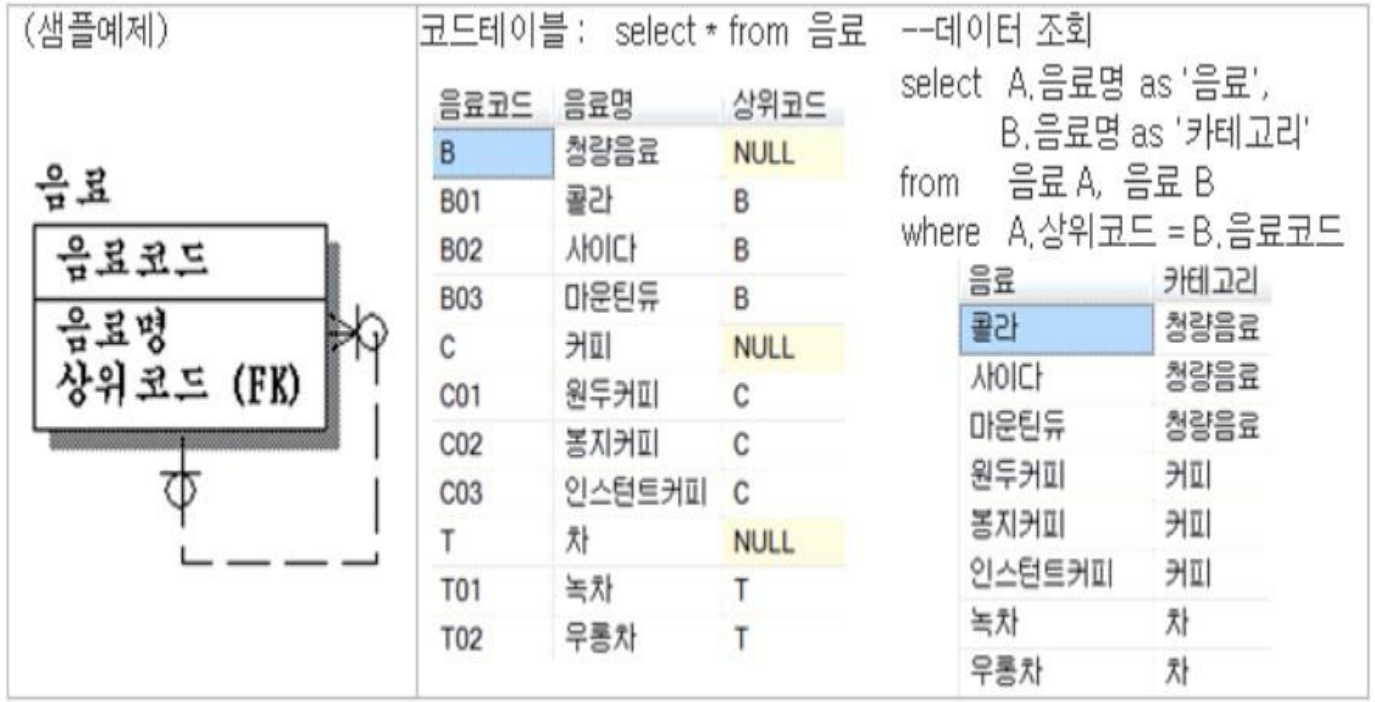



```
drop table 도서분류;

--도서분류(소분류코드, 소분류명, 대코드)
insert into 도서분류 values ('1', '도서분류', NULL);

insert into 도서분류 values ('000' , '총류', 1);
insert into 도서분류 values ('100' , '철학', 1);
insert into 도서분류 values ('200' , '종교', 1);
insert into 도서분류 values ('300' , '사회과학', 1);
insert into 도서분류 values ('400' , '순수과학', 1);
insert into 도서분류 values ('500' , '기술과학', 1);
insert into 도서분류 values ('600' , '예술', 1);
insert into 도서분류 values ('700' , '언어', 1);
insert into 도서분류 values ('800' , '문학', 1);
insert into 도서분류 values ('900' , '역사', 1);

insert into 도서분류 values ('010' ,'도서학/서지학' , '000');
insert into 도서분류 values ('020' ,'문헌정보학' , '000');
insert into 도서분류 values ('030' ,'학백과사전' , '000');
insert into 도서분류 values ('040' ,'강연집/수필집' , '000');
insert into 도서분류 values ('050' ,'일반연속간행물' , '000');
insert into 도서분류 values ('060' ,'일반학회/단체/협회' , '000');
insert into 도서분류 values ('070' ,'신문/언론/저널리즘' , '000');
insert into 도서분류 values ('080' ,'일반전집/총서' , '000');
insert into 도서분류 values ('090' ,'향토자료' , '000');

insert into 도서분류 values ('110' ,'형이상학' , '100');
insert into 도서분류 values ('130' ,'철학의체계' , '100');
```

```
select LPAD(' ', (LEVEL-1)*4) || 소분류명 도서분류
from   도서분류
start with 소분류명 = '도서분류'
connect by prior  소분류코드 = 대코드;
```

```
scott>set pagesize 50
 계층적 질의
scott>select LPAD(' ', (LEVEL-1)*4) || 소분류명 도서분류
  2  from 도서분류
  3  start with 소분류명 = '도서분류'
  4  connect by prior 소분류코드 = 대코드;
<실행결과>
도서분류
--------------------------------------------------------------------------------
도서분류
    총류
        도서학/서지학
        문헌정보학
        학백과사전
        강연집/수필집
        일반연속간행물
        일반학회/단체/협회
        신문/언론/저널리즘
        일반전집/총서
        향토자료
    철학
        형이상학
        철학의체계
        학
        아시아철학/사상
        서양철학
        논리학
        윤리학/도덕철학
    종교
        비교종교
        불교
        기독교
        천도교
```

[예제] (음료) 재귀적 관계



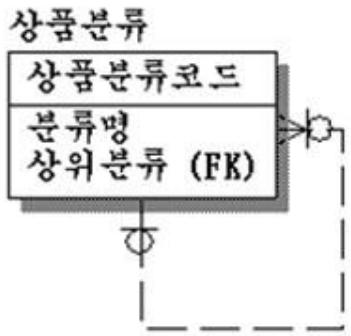
--음료분류코드에 해당하는 음료명과 상위 카테고리를 검색하라 (self join)

```
select A.음료명 as '음료', B.음료명 as '카테고리'
from 음료 A, 음료 B
where A.상위코드 = B.음료코드
```

```
--음료 (코드 테이블)
insert into 음료 values('B','청량음료',NULL);
insert into 음료 values('C','커피',NULL);
insert into 음료 values('T','차',NULL);
insert into 음료 values('B01','콜라','B');
insert into 음료 values('B02','사이다','B');
insert into 음료 values('B03','마운틴듀','B');
insert into 음료 values('C01','원두커피','C');
insert into 음료 values('C02','봉지커피','C');
insert into 음료 values('C03','인스턴트커피','C');
insert into 음료 values('T01','녹차','T');
insert into 음료 values('T02','우롱차','T');
```

[예제] (상품) 재귀적 관계

<샘플예제> 상품분류



(* 파워포인트 : 조직도 기능 사용하여 작성한 그림)

<데이터 입력>

```
--상품분류(상품분류코드, 분류명, 상위분류)

insert into 상품분류 values('1','의류', NULL);

insert into 상품분류 values('L1','니트/셔츠', '1');
insert into 상품분류 values('L2','자켓', '1');
insert into 상품분류 values('L3','바지/스커트', '1');
insert into 상품분류 values('L4','점퍼/코트', '1');

insert into 상품분류 values('S01','니트', 'L1');
insert into 상품분류 values('S02','가죽자켓', 'L2');
insert into 상품분류 values('S03','콤비', 'L2');
insert into 상품분류 values('S04','정장바지', 'L3');
insert into 상품분류 values('S05','사파리', 'L4');
insert into 상품분류 values('S06','골덴점퍼', 'L4');
insert into 상품분류 values('S07','가죽점퍼', 'L4');
insert into 상품분류 values('S08','면점퍼', 'L4');
insert into 상품분류 values('S09','드레스셔츠', 'L1');
insert into 상품분류 values('S10','남방', 'L1');
insert into 상품분류 values('S11','면셔츠', 'L1');
insert into 상품분류 values('S12','청바지', 'L3');
insert into 상품분류 values('S13','반바지', 'L3');
insert into 상품분류 values('S14','칠부바지', 'L3');
insert into 상품분류 values('S15','면바지', 'L3');
insert into 상품 values('S16','면', 'L2');

set pagesize 50

col 상품분류코드 format a12
col 분류명 format a30
col 상위분류 format a8
```

<계층적 질의>

```
scott>
scott>select LPAD(' ', (LEVEL-1)*2) || 분류명 상품분류
  2  from 상품분류
  3  start with 상품분류코드 = '1'
  4  connect by prior 상품분류코드 = 상위분류;
```

<실행결과>

```
상품분류
------------------------------
의류
  니트/셔츠
    니트
    드레스셔츠
    남방
    면셔츠
  자켓
    가죽자켓
    콤비
    면자켓
  바지/스커트
    정장바지
    청바지
    반바지
    칠부바지
    면바지
  점퍼/코트
    사파리
    골덴점퍼
    가죽점퍼
    면점퍼

21 개의 행이 선택되었습니다.
```

* 계층적 질의 :
조인문이나 뷰에서는 사용할 수 없으며,
CONNECT BY 절에서는 서브쿼리 절을 포함할 수 없다

/* 상품분류코드 1 부터 시작하여
자식 키 = 부모 키 에 해당하는 것을 찾아 루트 노드 부터 먼저 출력한다 */

- LPAD() 함수 사용: 계층적 질의의 출력결과를 사용자가 쉽게 이해할 수 있도록 표시

LPAD() 함수:
문자열이 일정한 크기가 되도록 왼쪽부터 지정한 문자를 삽입하는 함수
계층관계를 가지는 데이터를 출력할 때 단계별로 공백을 삽입하여 시각적인 효과를 주거나
게시판에서 답변 글의 정렬순서를 조절할 때 유용하게 사용

LPAD(expr | column, m[,char])

LEVEL : 계층별로 레벨 번호 표시하는데 루트노드는 1, 하위 레벨로 갈수록 1씩 증가
m 폭만큼 공백을 왼쪽에 char 문자 삽입, char 가 없으면 공백 삽입

```
-- 계층적 질의(Oracle)
select LPAD(' ', (LEVEL-1)*2) || 분류명 상품분류
from   상품분류
start with 상품분류코드 = '1'
connect by prior  상품분류코드 = 상위분류;
```