제 3 장. 관계 모델

- 데이터 표현
- 무결성 제약조건
- ER을 관계형 모델로

관계 모델 개요

- 관계 데이터 모델 : 1970년 Codd 에 의해 제안
- 관계 데이터베이스 시스템 : 70년대 중반
  - IBM : System R
  - U.C. Berkeley : Ingres
- 현재 사용중인 주된 데이터베이스 시스템
  - IBM의 DB2 계열
  - Informix, Oracle, Sybase
  - Microsoft 의 Access, SQL Server
- 장점
  - 단순한 데이터 표현법
  - 복잡한 질의도 쉽게 표현

관계형 모델 소개

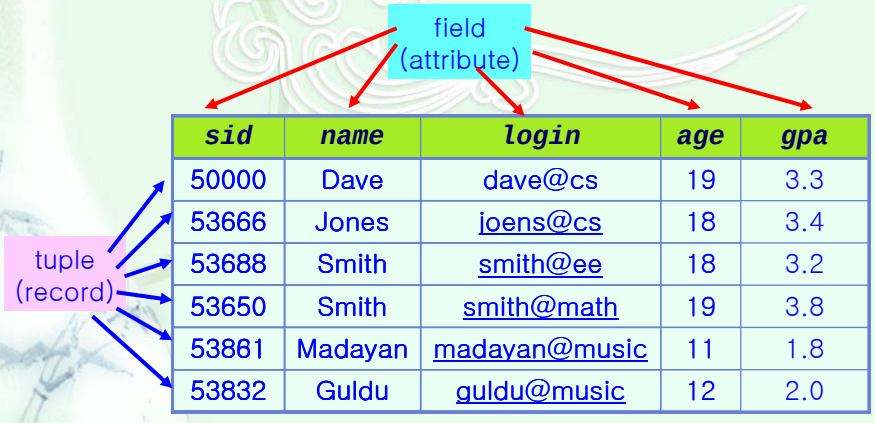
- 관계 데이터베이스 : relation의 모임
- relation : 다음 두 부분으로 구성
  - schema : DB의 논리적 구조
    - relation 이름 + 각 field(attribute)의 이름과 domain
    - 예) **Students**(*sid*:integer, *name*:string, *login*:string, *age*:integer, *gpa*:real)

sid	name	login	age	gpa

  - instance : tuple의 집합
    행과 열로 구성된 일종의 table
    - 행의 수 = 카디널리티(cardinality)
    - 필드의 수 = 차수(degree / arity)

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

예제 instance



sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- instance의 원소수(cardinality) : tuple의 수 6
- relation의 차수(degree) : filed 의 수 5

도메인 제약조건

- 도메인 제약조건(domain constraints)
  - 각 field의 값은 주어진 domain에 속하는 값으로 제한
  - 프로그래밍 언어의 data type 와 같은 개념

- relation schema : $\mathcal{R}(f_1:\mathcal{D}_1, \ldots, f_n:\mathcal{D}_n)$ 일 때
  - 도메인 $\mathcal{D}_i$ 에 속한 값 : d_i
  - $\mathcal{R}$의 한 instance

    $\{<f_1:d_1, \ldots, f_n:d_n> \mid d_1 \in \mathcal{D}_1, \ldots, d_n \in \mathcal{D}_n\}$

  Ex) <*sid*: 50000, *name*: Dave, *login*: dave@cs, *age*: 19, *gpa*: 3.3>

SQL로 relation 생성

- table 생성, 삭제, 수정 : 데이터 정의어(DDL) 사용
- 새로운 table 생성 : CREATE TABLE 문 사용

```
CREATE TABLE Students ( sid    NUMBER(5),
                        name   CHAR(30),
                        login  CHAR(20),
                        age    INTEGER,
                        gpa    REAL )
```

sid	*name*	*login*	*age*	*gpa*
53688	Smith	smith@ee	18	3.2

- INSERT 명령 : 새 tuple 입력

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES      (53688, 'Smith', 'smith@ee', 18, 3.2)
```

Tuple 삭제 및 수정

- DELETE 명령
  - 해당 tuple 삭제

```
DELETE FROM  Students
WHERE  name = 'Smith'
```

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- UPDATE 명령
  - 이미 있는 field 값 수정

```
UPDATE  Students
SET     gpa = gpa - 0.1
WHERE   gpa >= 3.3
```

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.2
53666	Jones	joens@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

테이블 생성 및 관리는 6장에서 자세히 다룸

무결성 제약조건

- 무결성 제약조건(integrity constraints: IC) :부정확한 정보가 입력되는 것을 방지하는 조건
  - 데이터베이스 schema 정의시 IC 표시
  - 데이터베이스 수정 실행될 때, IC 위반하는지를 검사
  - 위반하는 데이터 변경은 불허됨
  - 예)
    - domain 제약조건
    - key 제약조건
    - 외래 key 제약조건

제약조건은 10장에서 자세히 다룸

Key 제약조건

- key : tuple을 유일하게 식별하는 field 집합
- 정의 : 후보(candidate) key
  - 모든 tuple들은 key field의 값이 서로 다름 (유일성: uniqueness)
  - key field의 어떤 부분집합도 유일 식별자가 아님 (최소성: minimality)
  - 예) **Students** table의 key는 {**sid**}
    {**name**} ?

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- super key : key를 포함하는 집합
  - 예) {**sid, name**} 은 super key

SQL에서의 key 제약조건

```
CREATE TABLE Students ( sid         NUMBER(5),
                        name        CHAR(30),
                        login       CHAR(20),
                        age         INTEGER,
                        gpa         REAL,
                        UNIQUE      (name, age),
                        PRIMARY KEY (sid) )
```

- 한 relation은 여러 개의 후보 key를 가질 수 있음
  - UNIQUE 로 표시
  - 예) {**name, age**} 후보 key
- 여러 개의 후보 key 중, 기본(primary) key는 하나
- 개체 무결성(entity integrity) : 기본 key는 유일하여야 하고, 널이 될 수 없음

기본 key 와 후보 key 표현

```
CREATE TABLE Enrolled ( studid      NUMBER(5),
                        cid         CHAR(20),
                        grade       CHAR(10),
                        PRIMARY KEY (studid, cid))
```

- 한 학생이 여러 과목 수강 가능
- 주어진 학생과 과목에 대해서 평점은 하나

```
CREATE TABLE Enrolled ( studid      NUMBER(5),
                        cid         CHAR(20),
                        grade       CHAR(10),
                        PRIMARY KEY (studid),
                        UNIQUE      (cid, grade))
```

- 학생들은 한 과목씩 만 수강할 수 있으며, 평점도 하나
- 한 과목에서 같은 평점을 두 학생이 받을 수는 없다

외래(foreign) key 제약조건

- foreign key : 다른 relation의 primary key를 참조할 목적으로 가지고 있는, relation의 field 집합
- foreign key 제약조건
  피 참조 relation의 primary key에 존재하지 않는 값이 참조 relation에서 foreign key로 존재하면 안됨
- 참조 무결성(referential integrity)
  - foreign key 제약조건을 준수하는 상태
- 예) **Enrolled**(*studid:*number, *cid:*string, *grade:*string)
  - studid는 **Student** 테이블을 참조하는 외래 키

SQL에서 foreign key 표현

- 학생 테이블에 등록된 학생들만 과목 수강이 가능

```
CREATE TABLE Enrolled ( studid      NUMBER(5),
                        cid         CHAR(20),
                        grade       CHAR(10),
                        PRIMARY KEY (studid, cid),
                        FOREIGN KEY (studid) REFERENCES Students)
```

외래 key

cid	grade	studid
Carnatic101	C	53666
Reggae203	B	53832
Topology112	A	53650
History105	B	53666

기본 key

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- 참조 무결성(referential integrity)
  - **Enrolled** table에 < 55555, Art104, A> 입력하면, 거부됨
  - **Students** table로부터 <53666, Jones, jones@cs, 18, 3.4> 삭제하면, 거부됨

무결성 제약조건 집행

- domain, primary key, unique 제약조건
  - 이들을 위배하는 삽입, 삭제, 갱신 명령을 거부

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

```
INSERT
INTO   Students(sid, name, login, age, gpa)
VALUES ('53111', 'Mike', 'mike@ee', 17, 3.4)
```

'53111' 은 문자열 domain제약조건 위배

무결성 제약조건 집행

- domain, primary key, unique 제약조건
  - 이들을 위배하는 삽입, 삭제, 갱신 명령을 거부

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

```
INSERT
INTO    Students(sid, name, login, age, gpa)
VALUES  (53688, 'Mike', 'mike@ee', 17, 3.4)
```

53688 이미 존재
primary key 제약조건 위배

```
INSERT
INTO    Students(sid, name, login, age, gpa)
VALUES  (null, 'Mike', 'mike@ee', 17, 3.4)
```

primary key는 **null** 값을 가질 수 없다는 제약조건 위배

```
UPDATE  Students
SET     sid = 50000
WHERE   sid = 53688
```

50000 이미 존재
primary key 제약조건 위배

참조 무결성 집행

Students

sid	name	login	age	gpa

Enrolled

cid	grade	studid

```
PRIMARY KEY (sid)  FOREIGN KEY (studid) REFERENCES Students
```

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- 기본적인 3가지 사항
  - ① **학생** table에 없는 학번 값을 가진 tuple이 **수강**에 insert 될 때 ⇒ insert 명령 거부

```
INSERT
INTO    Enrolled(cid, grade, studid)
VALUES  ('Hindi101', 'B', 51111)
```

참조 무결성 집행

Students

sid	name	login	age	gpa

Enrolled

cid	grade	studid

```
PRIMARY KEY (sid)   FOREIGN KEY (studid) REFERENCES Students
```

- 기본적인 3가지 사항
  - ② **학생** tuple 하나가 삭제될 때, 다음 중 하나 선택
    - 삭제될 **학생** tuple을 참조하는 모든 **수강** tuple 삭제
    - **수강** tuple에 의해서 참조되고 있다면, 그 **학생** tuple 삭제될 수 없도록
    - 그 tuple을 참조하는 **수강** tuple의 학번값을 내정값으로 설정
    - 그 tuple을 참조하는 **수강** tuple의 학번값을 null로 설정
  - ③ **학생** tuple의 primary key 값이 갱신될 때, 앞의 경우와 마찬가지

```
DELETE  FROM  Students
WHERE   name = 'Jones'
```

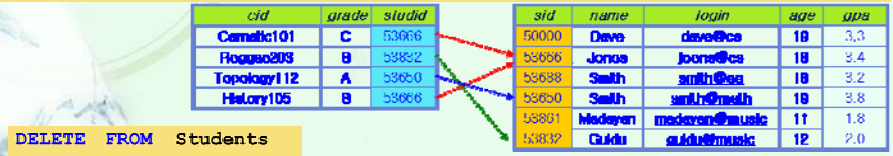

SQL에서 참조 무결성

```
CREATE TABLE Enrolled ( studid      NUMBER(5),
                        cid         CHAR(20),
                        grade       CHAR(10),
                        PRIMARY KEY (studid, cid),
                        FOREIGN KEY (studid) REFERENCES Students
                                    ON DELETE CASCADE)
```

- 4 가지 option
  - NO ACTION : 작업을 거부
  - CASCADE : 참조측도 삭제
  - SET DEFAULT : 참조측 값을 default 값으로
  - SET NULL : 참조측 값을 null 값으로

SQL에서 참조 무결성

```
CREATE TABLE Enrolled ( studid      NUMBER(5),
                        cid         CHAR(20),
                        grade       CHAR(10),
                        PRIMARY KEY (studid, cid),
                        FOREIGN KEY (studid) REFERENCES Students
                                    ON DELETE NO ACTION)
```



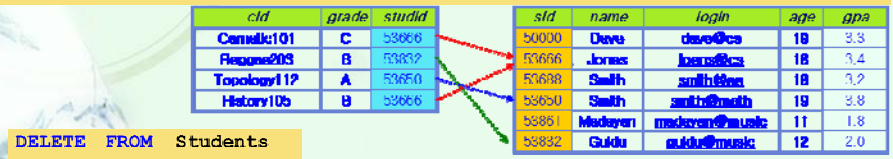
```
DELETE FROM Students
WHERE name = 'Jones'
```

cid	grade	studid
Carnatic101	C	53666
Reggae203	B	53832
Topology112	A	53650
History105	B	53666

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	joens@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

SQL에서 참조 무결성

```
CREATE TABLE Enrolled ( studid      NUMBER(5),
                        cid         CHAR(20),
                        grade       CHAR(10),
                        PRIMARY KEY (studid, cid),
                        FOREIGN KEY (studid) REFERENCES Students
                                    ON DELETE CASCADE)
```



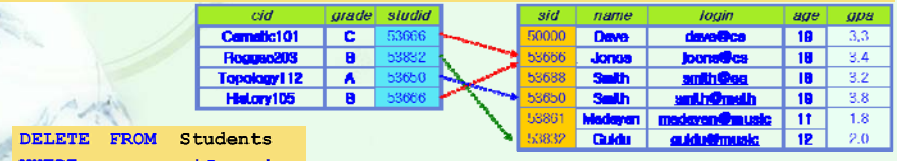
```
DELETE FROM Students
WHERE name = 'Jones'
```

cid	grade	studid
Reggae203	B	53832
Topology112	A	53650

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

SQL에서 참조 무결성

```
CREATE TABLE Enrolled ( studid      NUMBER(5) default 99999,
                        cid         CHAR(20),
                        grade       CHAR(10),
                        PRIMARY KEY (studid, cid),
                        FOREIGN KEY (studid) REFERENCES Students
                                    ON DELETE SET DEFAULT)
```



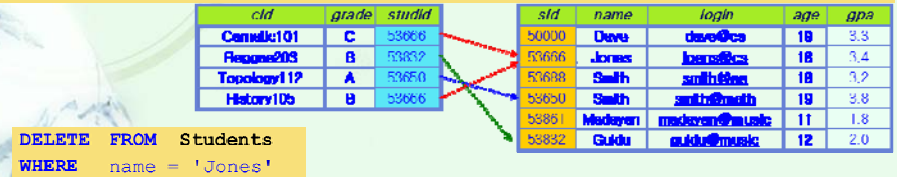
```
DELETE FROM Students
WHERE  name = 'Jones'
```

cid	grade	studid
Carnatic101	C	99999
Reggae203	B	53832
Topology112	A	53650
History105	B	99999

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

SQL에서 참조 무결성

```
CREATE TABLE Enrolled ( studid      NUMBER(5),
                        cid         CHAR(20),
                        grade       CHAR(10),
                        PRIMARY KEY (studid, cid),
                        FOREIGN KEY (studid) REFERENCES Students
                                    ON DELETE SET NULL)
```



```
DELETE FROM Students
WHERE  name = 'Jones'
```

cid	grade	studid
Carnatic101	C	null
Reggae203	B	53832
Topology112	A	53650
History105	B	null

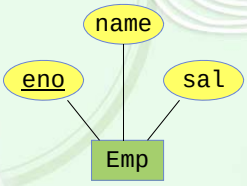
sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53861	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

ER에서 관계모델로

- primary key는 개체집합과 관계집합을 table로 표현하는 것을 허용
- ER-diagram을 table의 집합으로 표현 가능
- 각 개체집합과 관계집합에 대 유일한 table 존재
- 각 table은 attribute와 관계되는 여러 column을 가짐
- ER을 table로 바꾸는 것은 relational database 설계를 위한 기초

개체를 table로

- 개체 하나는 하나의 table로 변경
- 개체의 attribute들은 relation의 attribute로



eno	name	sal
3666	Smith	4800
5368	Paul	2200
3650	John	3500
6316	Peter	3800

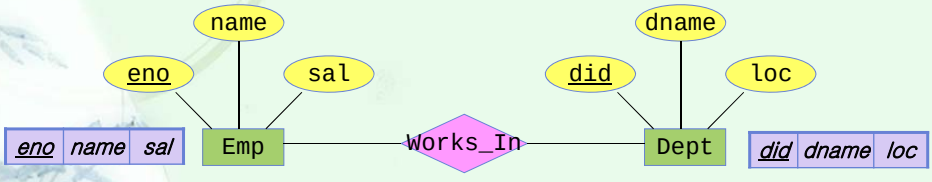
```
CREATE TABLE Emp ( eno          NUMBER(4),
                   name         CHAR(20),
                   sal          INTEGER,
                   PRIMARY KEY (eno))
```

```
CREATE TABLE Dept ( did          NUMBER(2),
                    dname        CHAR(20),
                    loc          CHAR(10),
                    PRIMARY KEY (did))
```

~ 24 ~

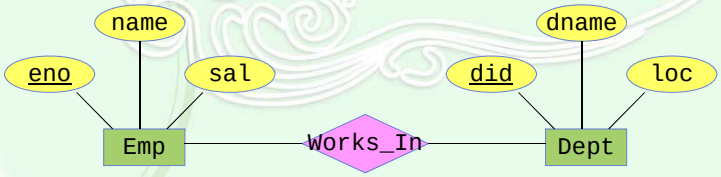
관계집합을 table로

- 관계집합 하나는 하나의 table로 변경
- table의 attribute
  - 참여하는 각 개체집합의 primary key (foreign key가 됨)
- primary key는 모든 개체집합의 primary key의 조합




```
CREATE TABLE Works_In ( eno          NUMBER(4),
                        did          NUMBER(2),
                        PRIMARY KEY (eno, did),
                        FOREIGN KEY (eno) REFERENCES Emp,
                        FOREIGN KEY (did) REFERENCES Dept)
```


관계집합을 table로




```
CREATE TABLE Emp ( eno          NUMBER(4),
                   name         CHAR(20),
                   sal          INTEGER,
                   PRIMARY KEY (eno))
```

```
CREATE TABLE Dept ( did          NUMBER(2),
                    dname        CHAR(20),
                    loc          CHAR(10),
                    PRIMARY KEY (did))
```

```
CREATE TABLE Works_In ( eno          NUMBER(4),
                        did          NUMBER(2),
                        PRIMARY KEY (eno, did),
                        FOREIGN KEY (eno) REFERENCES Emp,
                        FOREIGN KEY (did) REFERENCES Dept)
```

다대다 관계

- **Emp, Dept, Works_In** 테이블

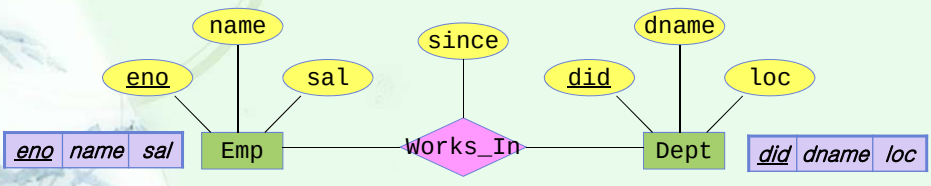
eno	name	sal
3666	Smith	4800
5368	Paul	2200
3650	John	3500
6316	Peter	3800

did	dname	loc
51	Accounting	NY
56	Sales	LA
60	Research	SF

eno	did
3666	51
5368	51
5368	56
3650	60
6316	60

관계집합을 table로

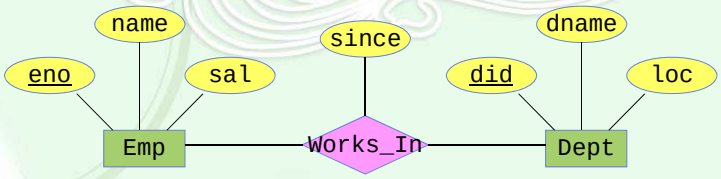
- 관계집합 하나는 하나의 table로 변경
- table의 attribute
  - 참여하는 각 개체집합의 primary key (foreign key가 됨)
  - 관계집합 자체의 관계 attribute
- primary key는 모든 개체집합의 primary key의 조합




```
CREATE TABLE Works_In ( eno          NUMBER(4),
                        did          NUMBER(2),
                        since        DATE,
                        PRIMARY KEY (eno, did),
                        FOREIGN KEY (eno) REFERENCES Emp,
                        FOREIGN KEY (did) REFERENCES Dept)
```

다대다 관계

- **Emp, Dept, Works_In** 테이블



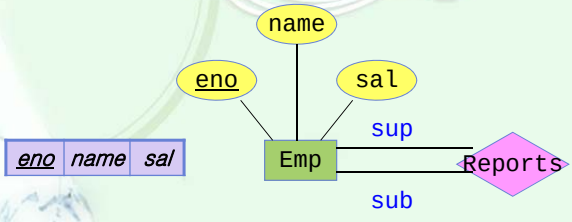
eno	name	sal
3666	Smith	4800
5368	Paul	2200
3650	John	3500
6316	Peter	3800

eno	since	did
3666	1/1/91	51
5368	3/3/93	51
5368	2/2/92	56
3650	3/1/92	60
6316	3/1/92	60

did	dname	loc
51	Accounting	NY
56	Sales	LA
60	Research	SF

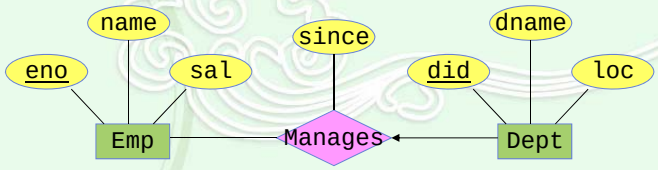
관계집합을 table로(계속)

- 역할(role) 이 있는 경우
- 역할 지시자 : 상급자(sup), 하급자(sub)




```
CREATE  TABLE Reports
      ( sup_eno       CHAR(11),
        sub_eno       CHAR(11),
        PRIMARY KEY   (sup_eno, sub_eno),
        FOREIGN KEY   (sup_eno) REFERENCES Emp(eno),
        FOREIGN KEY   (sub_eno) REFERENCES Emp(eno))
```

Key 제약조건이 있는 관계집합



- 부서당 관리자가 한명이므로, did가 같으면서 eno가 다른 tuple은 존재 불가
- 따라서 did 가 관리의 primary key

```
CREATE TABLE Manages ( eno        NUMBER(4),
                       did        NUMBER(2),
                       since      DATE,
                       PRIMARY KEY (did),
                       FOREIGN KEY (eno) REFERENCES Emp,
                       FOREIGN KEY (did) REFERENCES Dept)
```

eno	*name*	*sal*

did	*dname*	*loc*

다대다 관계

- **Emp, Dept, Manage** 테이블

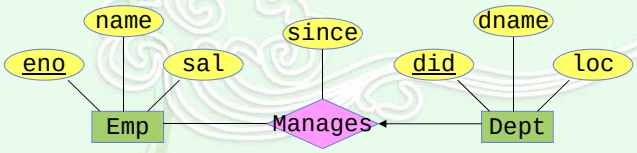
eno	name	sal
3666	Smith	4800
5368	Paul	2200
3650	John	3500
6316	Peter	3800

did	dname	loc
51	Accounting	NY
56	Sales	LA
60	Research	SF

eno	since	did
5368	3/3/93	51
5368	2/2/92	56

- 관계집합 관리와 부서 table 의 키(did)가 같음
- 관계집합 관리와 대한 table 따로 만들 필요 없음

Key 제약조건에 대한 다른 방법



- 관계집합 정보를 key 제약조건에 참여하는 개체집합의 table에 표시
- 한 부서에 최대 한명의 부서장, eno 와 since 를 **부서** table에 표시
- 부서장이 없는 부서가 존재할 때, null 로 채워짐
- 2 개의 table 만 필요 : **직원**, **부서-부서장**

eno	*name*	*sal*

```
CREATE TABLE Dept_Mgr ( did          NUMBER(2),
                        dname        CHAR(20),
                        loc          CHAR(10),
                        eno          NUMBER(4),
                        since        DATE,
                        PRIMARY KEY (did),
                        FOREIGN KEY (eno) REFERENCES Emp
                                    ON DELETE SET NULL)
```

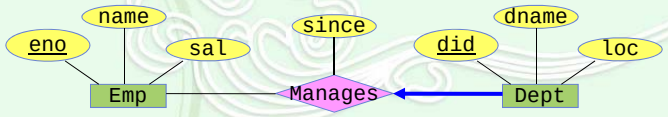
1 대 다 관계

- **Emp, Dept_Mgr** 테이블

eno	name	sal
3666	Smith	4800
5368	Paul	2200
3650	John	3500
6316	Peter	3800

did	dname	loc	eno	since
51	Accounting	NY	5368	3/3/93
56	Sales	LA	5368	2/2/92
60	Research	SF	null	null

참여 제약조건이 있는 관계집합



- 참여 제약조건 : 부서마다 반드시 부서장 존재
- key 제약조건 : 부서마다 한명의 부서장

eno	*name*	*sal*

```
CREATE TABLE Dept_Mgr ( did         NUMBER(2),
                        dname       CHAR(20),
                        loc         CHAR(10),
                        eno         NUMBER(4) NOT NULL,
                        since       DATE,
                        PRIMARY KEY (did),
                        FOREIGN KEY (eno) REFERENCES Emp
                                    ON DELETE NO ACTION)
```

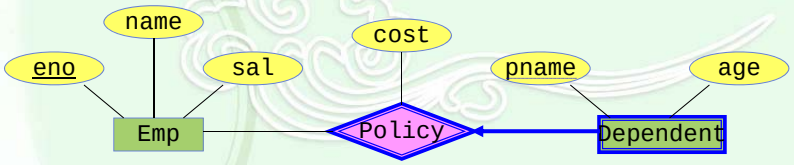

1대다 관계

- **Emp, Dept_Mgr** 테이블

eno	name	sal
3666	Smith	4800
5368	Paul	2200
3650	John	3500
6316	Peter	3800

did	dname	loc	eno	since
51	Accounting	NY	5368	3/3/93
56	Sales	LA	5368	2/2/92
60	Research	SF	6316	3/1/92

약개체(weak entity)를 table로

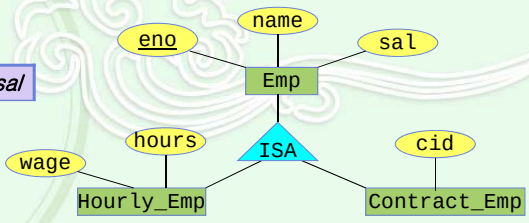


- 약개체 table의 attribute : 약개체 attribute + 소유자개체의 primary key + 설명용 attribute
- table의 primary key : (약개체의 부분 key, 소유자개체의 primary key
- 존재종속 개념 : 소유자개체가 삭제되면 관련된 모든 약개체 들도 삭제

eno	*name*	*sal*

```
CREATE TABLE Dep_Policy ( pname       CHAR(20),
                          age         INTEGER,
                          cost        REAL,
                          eno         CHAR(11),
                          PRIMARY KEY (pname, eno),
                          FOREIGN KEY (eno) REFERENCES Emp
                                      ON DELETE CASCADE)
```


ISA 계층을 table로



eno	*name*	*sal*

eno	*cid*

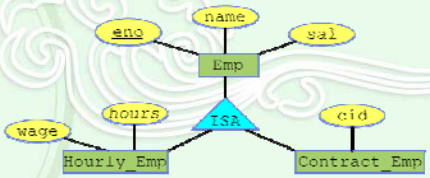
① 상위 level(superclass)에 대한 table 생성

- 개체집합 **직원, 시간제직원, 계약제직원** 을 각각 다른 relation으로 표시

  Hourly_Emp = (eno, wage, hours) : eno는 시간제직원의 primary key 임과 동시에 superclass(**직원**)을 참조하는 외래 key

```
CREATE TABLE Hourly_Emp ( eno         CHAR(11),
                          wage        INTEGER,
                          hours       REAL,
                          PRIMARY KEY (eno),
                          FOREIGN KEY (eno) REFERENCES Emp)
```

ISA 계층을 table로



eno	*name*	*sal*	*cid*

② 상위 level에 대한 table 생성하지 않음

- ❖ **시간제직원, 계약제직원** 의 두 relation으로만 표시

  Hourly_Emp = (eno, name, sal, wage, hours) :
  superclass(**직원**)의 모든 attribute도 포함

```
CREATE TABLE Hourly_Emp ( eno          CHAR(11),
                          name         CHAR(10),
                          sal          INTEGER,
                          wage         INTEGER,
                          hours        REAL,
                          PRIMARY KEY (eno))
```


table의 제거

- table(relation) 제거
  - ❖ 모든 tuple 삭제, 해당 table의 schema 정보도 삭제
  - ❖ DROP TABLE 명령 사용

```
DROP TABLE Students
           RESTRICT
```

⇒ 이 **학생** table을 참조하는 view나 무결성 제약조건이 없어야 수행됨

```
DROP TABLE Students
           CASCADE
```

⇒ 이 **학생** table을 참조하는 view나 무결성 제약조건도 연쇄적 제거됨

table의 변경

- table 구조 수정
  - ALTER TABLE 명령 사용

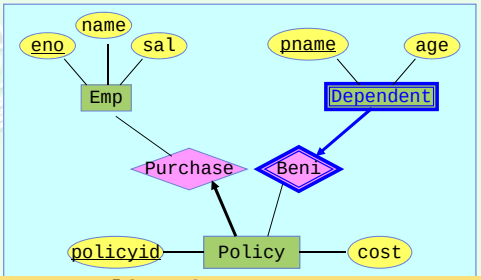
```
ALTER TABLE  Students
             ADD COLUMN maiden-name CHAR(10)
```

⇒ **학생** table의 schema는 새 field 추가,
각 tuple들의 해당 field에 null이 채워짐

sid	*name*	*login*	*age*	*gpa*	*maiden-name*
50000	Dave	dave@cs	19	3.3	null
53666	Jones	joens@cs	18	3.4	null
53688	Smith	smith@ee	18	3.2	null
53650	Smith	smith@math	19	3.8	null
53861	Madayan	madayan@music	11	1.8	null
53832	Guldu	guldu@music	12	2.0	null

기타 예제

- 세 개의 table로 구성 :
  직원, 보험증권, 피부양자
- 참여 및 key 제약조건에 의해서 **구매** table은 따로 만들어지지 않음
- 피부양자는 weak entity



eno	*name*	*sal*

```
CREATE  TABLE Dependent(
        pname           CHAR(20),
        age             INTEGER,
        policyid        INTEGER,
        PRIMARY KEY     (pname, policyid),
        FOREIGN KEY     (policyid) REFERENCES Policy
                        ON DELETE CASCADE )
```

```
CREATE  TABLE Policy  (
        policyid        INTEGER,
        cost            REAL,
        eno             CHAR(11) NOT NULL,
        PRIMARY KEY     (policyid),
        FOREIGN KEY     (eno) REFERENCES Emp)
```

요 약

- 관계모델의 주 구성요소는 relation
- 무결성 제약조건(Integrity Constraints)
  - domain 제약조건
  - key 제약조건
  - foreign key 제약조건
- SQL은 표현력이 풍부한 질의어
- ER 모델을 관계모델로 변환

Q & A