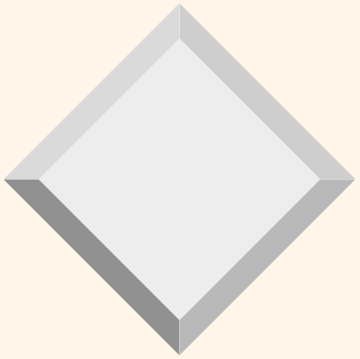


The Relational Model

Why Study the Relational Model?

- Most widely used model.
  - Vendors: Oracle, Microsoft, Informix, Ingres, Sybase, etc. → RDBMS (RDB)
- Recent competitor: object-oriented model
  - ObjectStore, Jasmine, Versant, Ontos
  - A synthesis emerging: *object-relational model*
    - Oracle, Informix Universal Server, UniSQL, O2, DB2

Relational Database: Definitions

- *Relational database:* a set of *relations*
- *Relation:* made up of 2 parts:
  - *Schema* : specifies name of relation, plus name and type of each column.
    - E.G. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real). → Schema
  - *Instance* : a state of *table*, with rows and columns.
- Can think of a relation as a *set* of rows or *tuples* (i.e., all rows are distinct).

Example Instance of Students Relation

field, column, attribute

record, tuple, row →

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2

- Schema

  Students (*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)

  Note: In Oracle, there is no data type string. Instead, use char or varchar. There is no real data type either. Use number.

Relational Query Languages

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
  - The key: precise semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

Relational Query Languages Cont'd

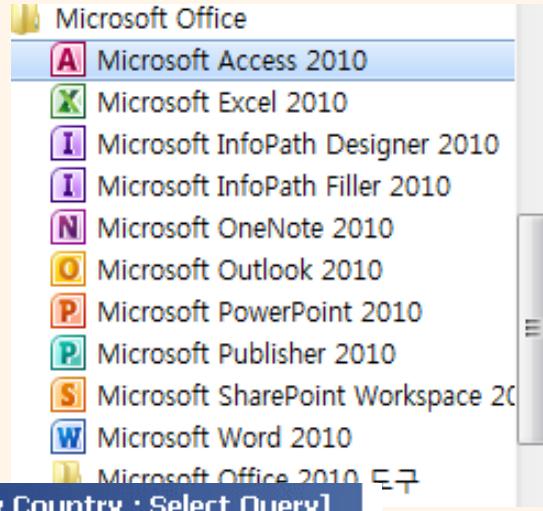
- Relational Algebra
- Relational Calculus
- SQL (Structured Query Language)
- QBE (Query By Example)

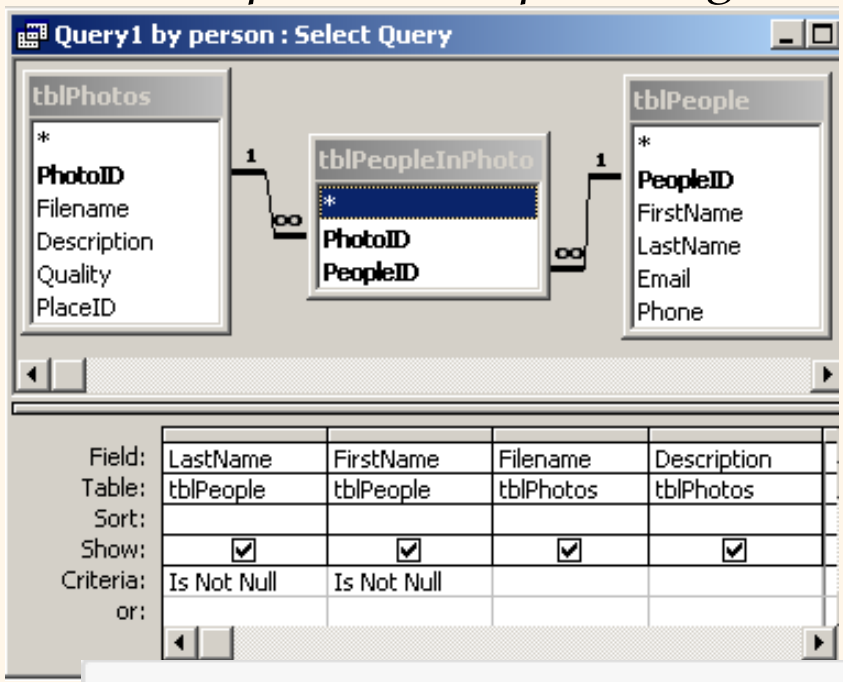
- Relational Algebra and SQL will be covered in this course.

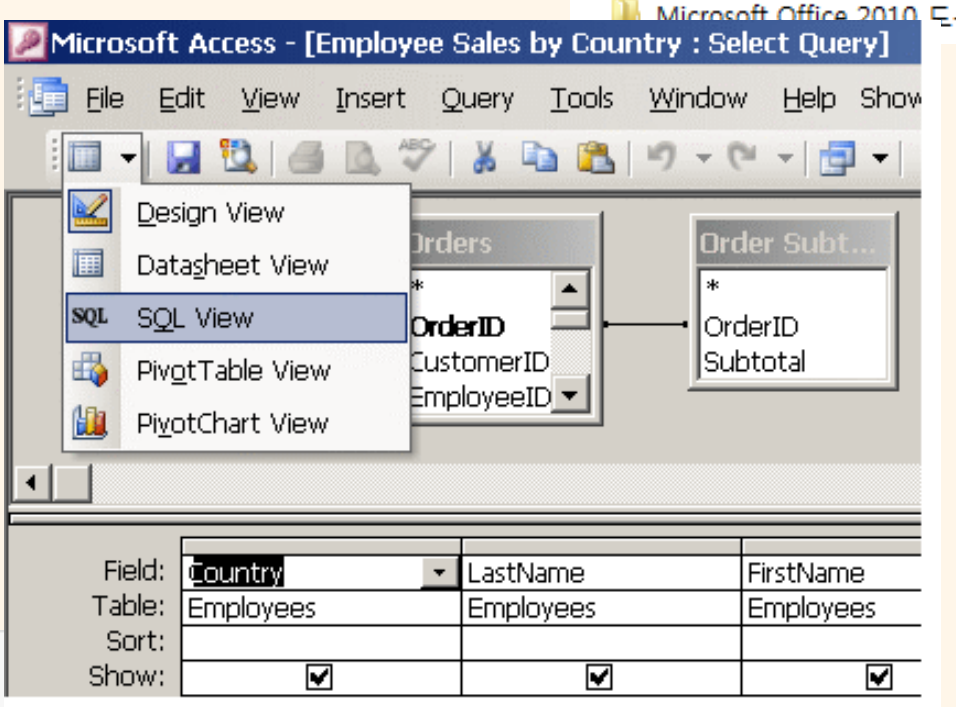
Example: MS Access Query

(QBE, SQL views)

- *https://en.wikipedia.org/wiki/Query_by_Example*








```
SELECT tblPeople.LastName,
tblPeople.FirstName,
tblPhotos.Filename,
tblPhotos.Description
FROM (tblPeople INNER JOIN tblPeopleInPhoto ON tblPeople.PeopleID = tblPeop
INNER JOIN tblPhotos ON tblPeopleInPhoto.PhotoID = tblPhotos.PhotoID
WHERE (tblPeople.LastName Is Not Null) AND (tblPeople.FirstName Is Not Null
```

Relational model
relation algebra → relational query language
relation calculus

The SQL Query Language

cf) SEQUEL

structured

- Developed by IBM (system R) in the 1970s
  - *https://en.wikipedia.org/wiki/SQL*
- Need for a standard since it is used by many vendors
- Standards:
  - SQL-86 (SQL-1)
  - ~~SQL-89 (minor revision)~~
  - SQL-92 (major revision, called SQL-2)
  - SQL-99 (major extensions) (SQL-3)

지식기반 DB

Object개념

↳ 요새

The SQL Query Language

Read operation

- To find all 18 year old students, we can write:

```
SELECT *
FROM Students
WHERE age=18;
```

Student 테이블 그 자체의 레코드를 가지고 있는 인스턴스

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```
SELECT name, login
FROM Students
WHERE age=18;
```

Creating Relations in SQL

- Creates the Students relation. Observe that the data type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

```
CREATE TABLE Students
    (sid CHAR(20),
     name CHAR(20),
     login CHAR(10),
     age INTEGER,
     gpa NUMBER);
```

- As another example, the Enrolled table holds information about courses that students take.

```
CREATE TABLE Enrolled
    (sid CHAR(20),
     cid CHAR(20),
     grade CHAR(2));
```

Destroying and Altering Relations

DROP TABLE Students; → DB안에서 삭제, DELETE(X)

- Destroys the relation Students. The schema information *and* the tuples are deleted.

modify

ALTER TABLE Students ⇒ 구조를 바꾸는것!

동적크기할당 : 최대 100자 < 저장공간낭비↓ 오버헤드↑

예쁘게 ADD (Address VARCHAR(100));

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field.

ALTER TABLE Students

MODIFY (Address VARCHAR(150)); → 타입변경

Adding and Deleting Tuples

- Can insert a single tuple using:

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

- Can delete all tuples satisfying some condition (e.g., name = Smith):

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

Integrity Constraints (ICs)

- IC: condition that must be true for *any* instance of the database; e.g., *domain constraints.*
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.

```
CREATE TABLE Worker
    (WID CHAR(9),
     Name CHAR(20),
     Age Number CHECK (Age Between 18 and 65));
```

- DBMS should not allow any IC to be violated anytime.

Primary Key Constraints

- A set of fields is a *key* for a relation if :
  1. No two distinct tuples can have same values in the key fields, and
  2. This is not true for any subset of the key.
     - Part 2 false? A *superkey*.

  If there's >1 key for a relation, one of the keys is chosen (by DBA) to be the *primary key*.
- E.g., *sid* is a key for Students. (What about *name*?) The set {*sid, gpa*} is a superkey.

Primary and Candidate Keys in SQL

- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.

cf) secondary key (보조키) 또는 candidate key
: 선택받지 못한 키

```
CREATE TABLE Enrolled_1
  (sid CHAR(20)
   cid  CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid) );
```

```
CREATE TABLE Enrolled_2
  (sid CHAR(20)
   cid  CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid),
   UNIQUE (cid, grade) );
```

key가 2개

→ PRIMARY KEY는 아니지만 key로 선언 (secondary key)

Foreign Keys, Referential Integrity

참조하는 다른 테이블의 PRIMARY KEY

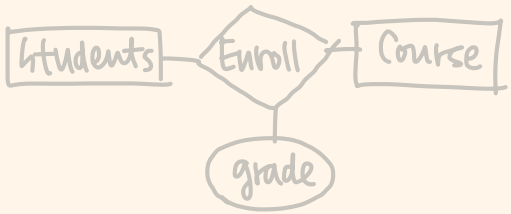
- *Foreign key* : Set of fields in one relation that is used to `refer' to a tuple in another relation. (Must correspond to primary key of the second relation.) Works like a `pointer(link)'.

반드시 PRIMARY KEY일 필요는 없으나 레코드를 구별해야하므로 강제함

- E.g. *sid* is a foreign key referring to Students:
  - Enrolled(*sid*, *cid*, *grade*)

Students Course

cf) ER모델에서는



- *Referential Integrity* : if the value of the foreign key is not null, the value must appear in the corresponding primary key. i.e., no dangling references.

참조무결성규칙 (참조제약)

dangling reference problem (null pointer segment fail)

  - Can you name a data model w/o referential integrity?
    - Links in HTML!

Foreign Keys and Referential Integrity in SQL

- Only students listed in the Students relation should be allowed to enroll for courses. → foreign key 선언을 했으니까

선언을 안하면 DBMS는 모름
↳ DB가 깨짐!! 존재하지 않는 학생정보...

```
CREATE TABLE Enrolled
  (sid CHAR(20),  cid CHAR(20),  grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid) REFERENCES Students (sid) );
```

→ 옵션: 없는 경우는 students 테이블에서 동일한 이름의 column을 가져옴

↳ 명시해줘야함 (FOREIGN KEY)

↳ 어디서 가져왔는지? (REFERENCES)

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

↳ sid만 PRIMARY KEY로 설정하면 다른 과목이 못들어감

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enforcing Referential Integrity

- Consider Students and Enrolled; *sid* in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted? (*Reject it!*)
- What should be done if a Students tuple is deleted?
  - Also delete all Enrolled tuples that refer to it.
  - Disallow deletion of a Students tuple that is referred to.
  - Set sid in Enrolled tuples that refer to it to a *default sid or NULL value*.

    (NULL means `*unknown*' or `*inapplicable*'.)
- Similar if primary key of Students tuple is updated.

Referential Integrity in SQL/92

SQL 2 는 오로지 이것만 지원

- SQL/92 supports all 4 options on deletes and updates.
  - Default is NO ACTION (*delete/update is rejected*)
  - ON DELETE CASCADE (also delete all tuples that refer to deleted tuple) +) on update cascade, ...
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
   (sid CHAR(20),
    cid CHAR(20),
    grade CHAR(2),
    PRIMARY KEY  (sid,cid),
    FOREIGN KEY (sid)
      REFERENCES Students (sid)
          ON DELETE CASCADE
          ON UPDATE SET DEFAULT )
```

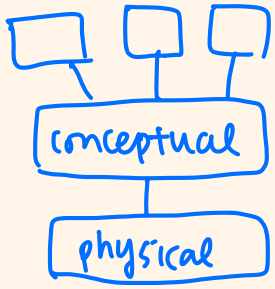
Referential Integrity in ORACLE

- Oracle 8i supports
  - Default is NO ACTION (*delete/update is rejected*)
  - ON DELETE CASCADE (also delete all tuples that refer to deleted tuple)

↳ 2가지만지원함!

```
CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid)
     REFERENCES Students
          ON DELETE CASCADE)
```

Views



← view : 유저가 원하는 schema만 보기 위해 만듦 (external schema)

- A *view* is just a relation, *but we store a definition*, rather than a set of tuples.

→ relation을 선언하듯이 (애크로) 또는 매번 conceptual schema로 내려가서 써야되니까 relation으로 저장 (materialized view)

```
CREATE VIEW YoungActiveStudents (name, grade)
    AS SELECT S.name, E.grade
    FROM Students S, Enrolled E
    WHERE S.sid = E.sid and S.age<21
```

↳ column의 data type 선언 X
conceptual schema (base table) 로부터 오는거니까

- Views can be dropped using the DROP VIEW command.

relational algebra & sql에서 다시 review

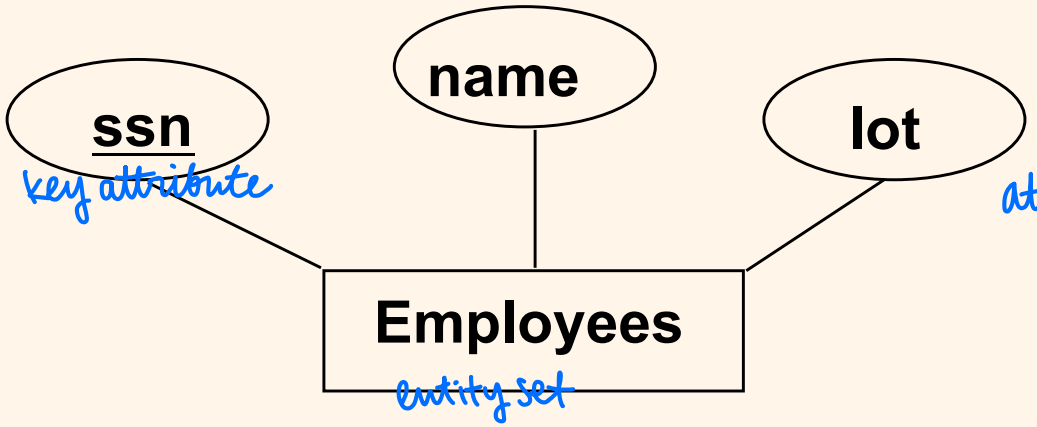
Logical Design: ER to Relations

relational model : db안에 relation들이있는것
(ERdiagram)

- Entities => Relations
- Relationships => Relations

relational model로 변경하면
모두 relation으로 변환

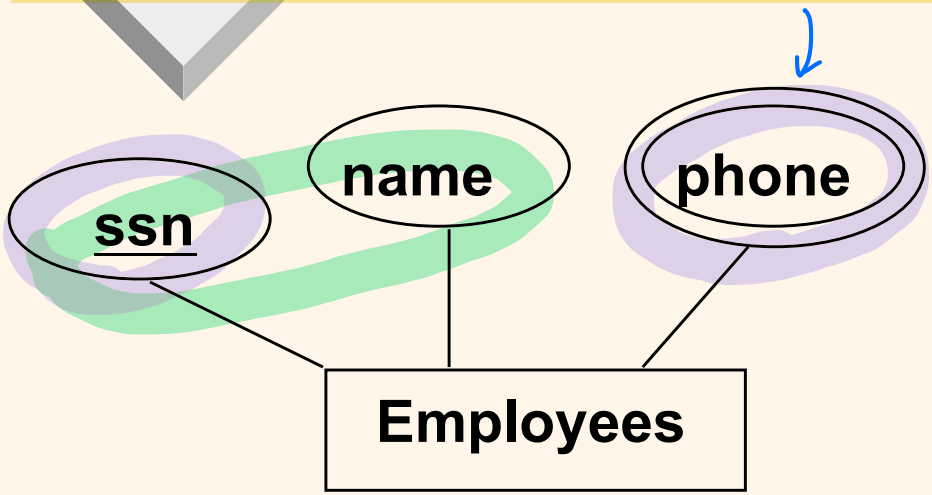
Entity Sets to Tables



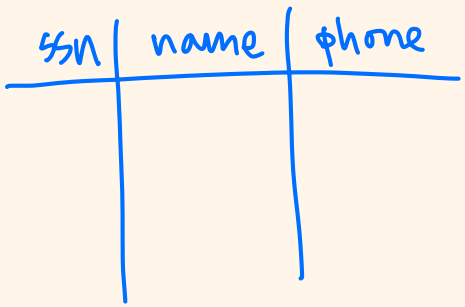

```
CREATE TABLE Employees
    (ssn CHAR(11),
    name CHAR(20),
    lot  INTEGER,
    PRIMARY KEY  (ssn))
```

↳ conceptual schema에는 data type x
(툴에는 입력하는 곳이 있음)

Multi-valued attributes




```
CREATE TABLE Employees
(ssn CHAR(11),
 name CHAR(20),
PRIMARY KEY  (ssn));
```

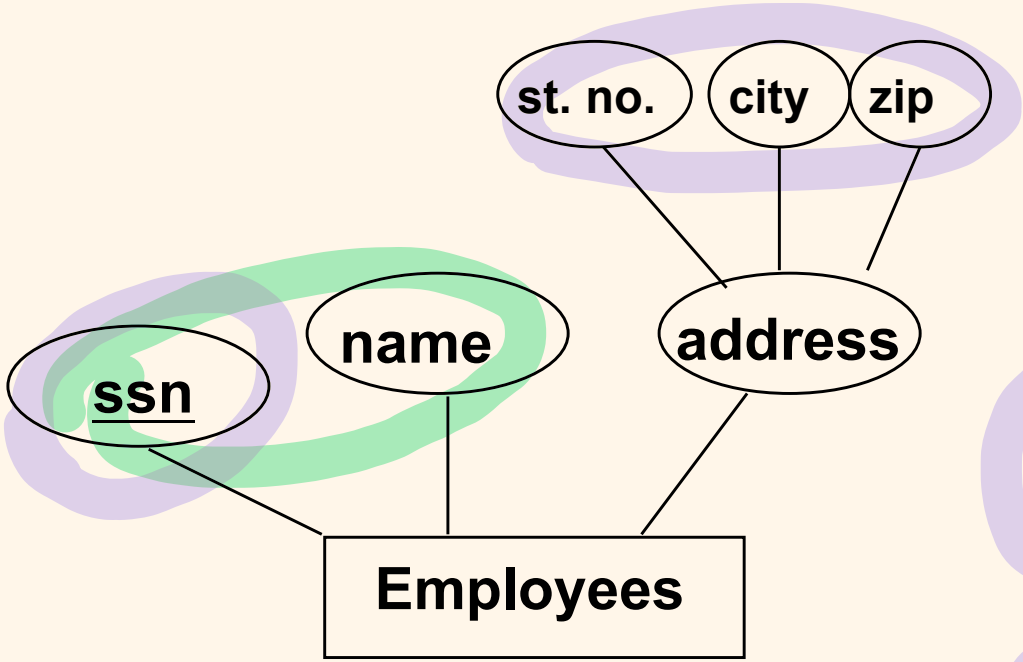


↳ 하나의 employee가 2개 이상의 phone을 가지는 x

```
CREATE TABLE Emp_Phones
(ssn CHAR(11),
phone CHAR(12),
FOREIGN KEY (ssn)
      REFERENCES Employees,
PRIMARY KEY  (ssn, phone));
```

↓
여러개 갖을 수

Composite attributes



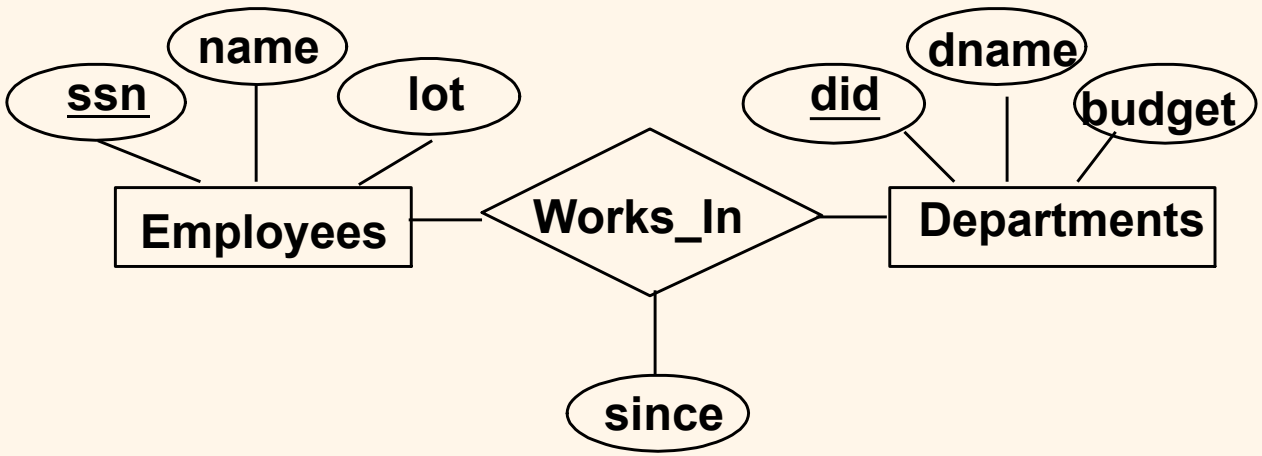

```
CREATE TABLE Employees
(ssn CHAR(11),
 name CHAR(20),
PRIMARY KEY  (ssn));

CREATE TABLE Emp_Address
(ssn CHAR(11),
St_no CHAR(20),
City char(20),
Zip char(10),
FOREIGN KEY (ssn)
      REFERENCES Employees,
PRIMARY KEY  (ssn));
```

↳ address가 multi-valued 속성임

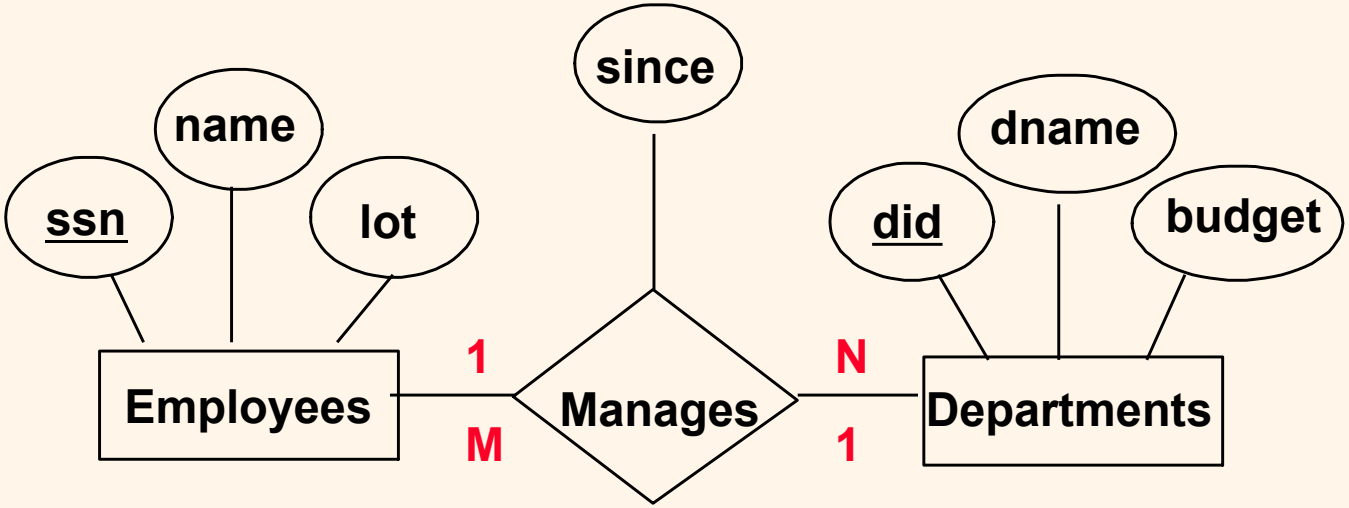
Relationship Sets to Tables

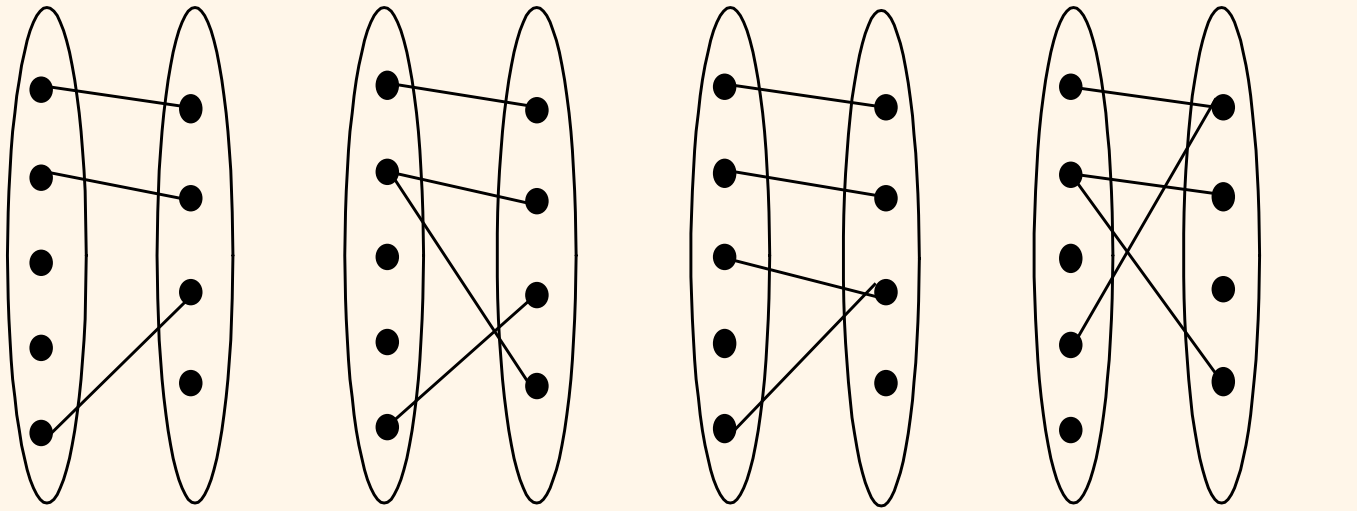
- In translating a relationship set to a relation, attributes of the relation must include:
  - Keys for each participating entity set (as foreign keys).
    - This set of attributes forms a *primary key* for the relation.
  - All descriptive attributes.

```
CREATE TABLE Works_In(
  ssn  CHAR(11),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
      REFERENCES Employees,
  FOREIGN KEY (did)
      REFERENCES Departments)
```



Translating ER Diagrams with Cardinality Constraints

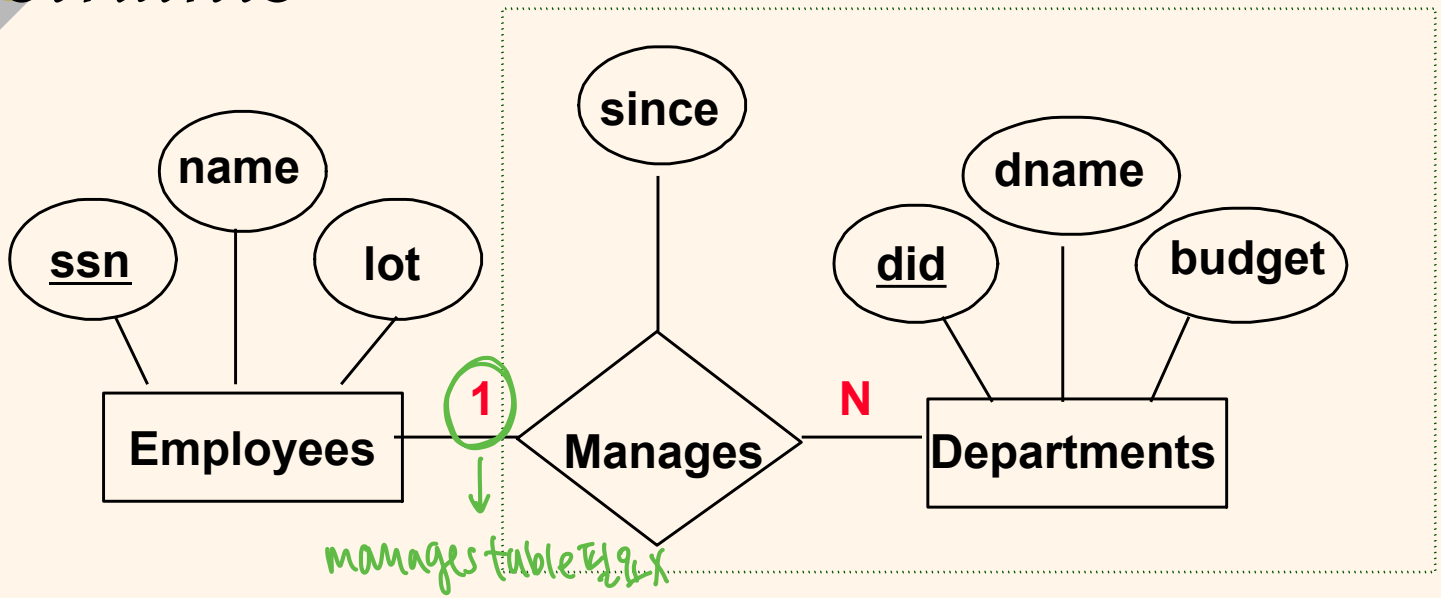




1-to-1 | 1-to Many | Many-to-1 | Many-to-Many

Translation to relational model?

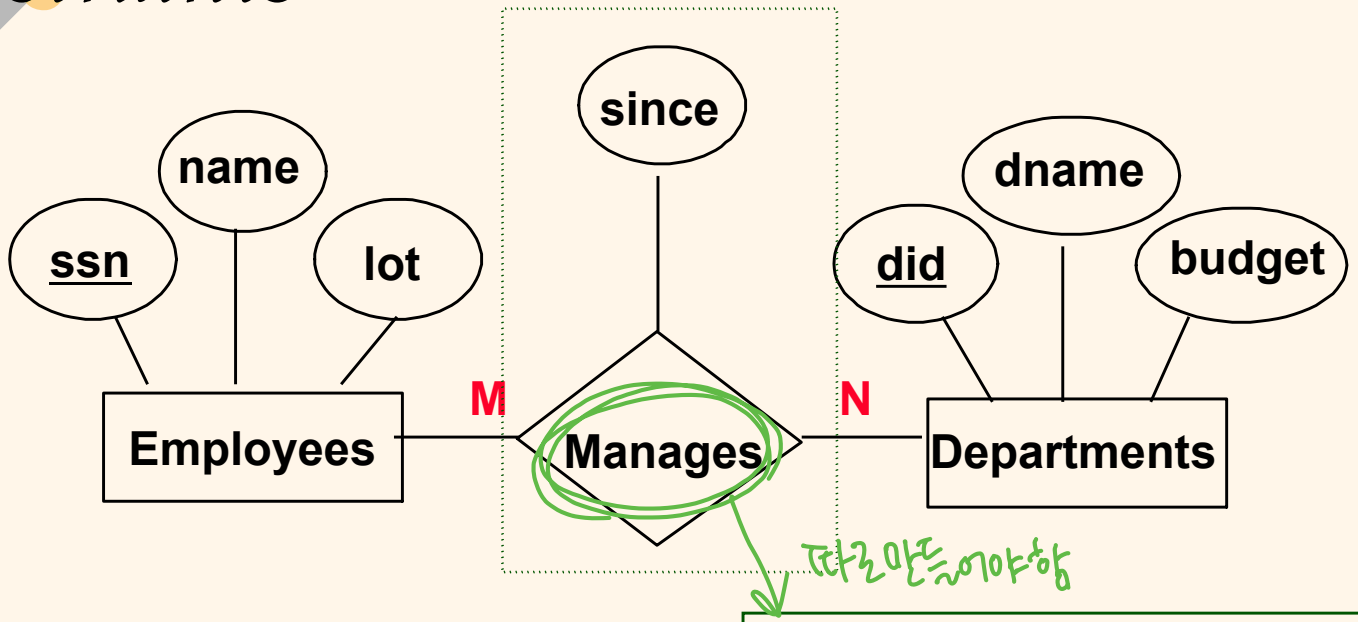
Translating ER Diagrams with Cardinality Constraints



- Since each department has a unique manager, we could instead combine Manages and Departments.

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  MGRssn CHAR(11), 있는다
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (MGRssn) REFERENCES Employees(ssn))
```

Translating ER Diagrams with Cardinality Constraints



- Since a department or a employee may have multiple managers or be the manager of multiple departments respectively, we can't combine Manages into either Employee or Departments.

```
CREATE TABLE Manages(
  ssn  CHAR(11),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
      REFERENCES Employees,
  FOREIGN KEY (did)
      REFERENCES Departments)
```

Translating ER Diagrams with Key Constraints

- Map relationship to a table:
  - Note that did is the key now!
  - Separate tables for Employees and Departments.

```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```

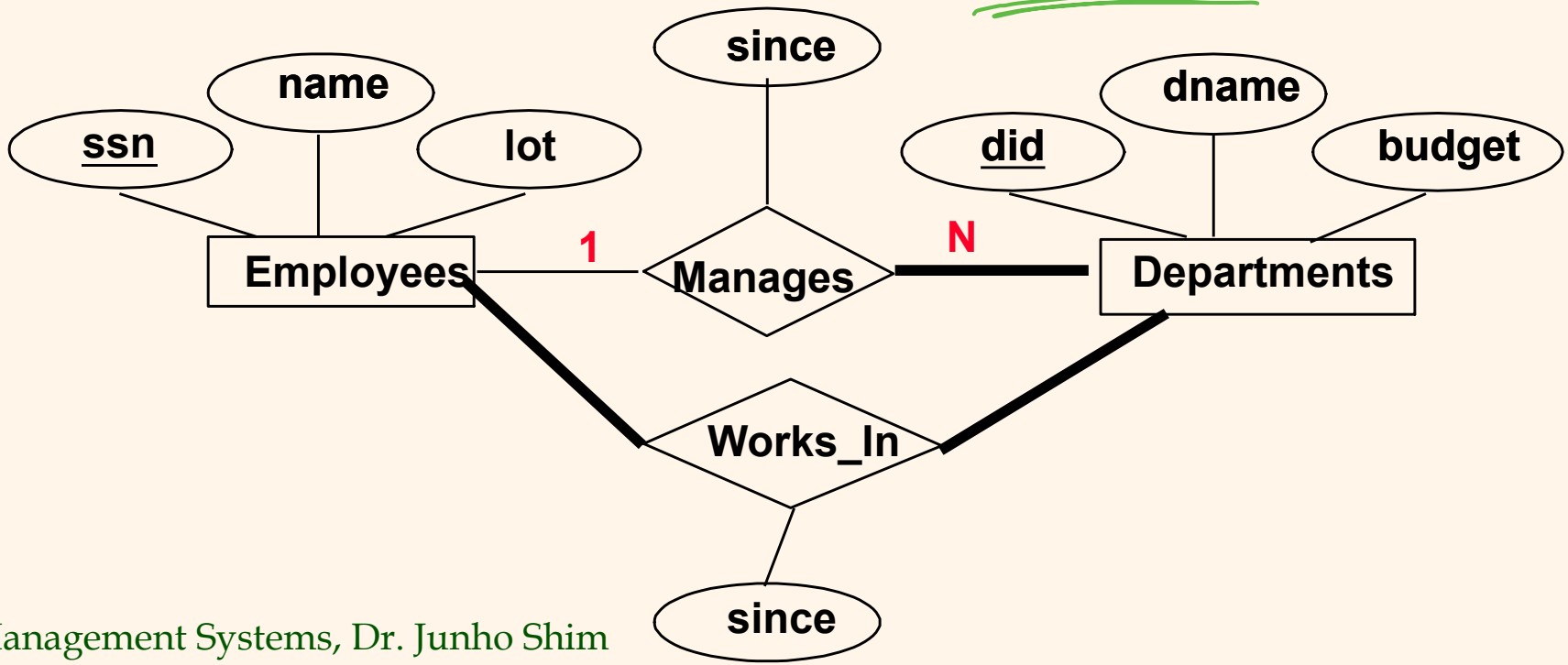
(희박하게) null값은 가질수도 있음 ⇒ 지울수있다

- Since each department has a unique manager, we could instead combine Manages and Departments.

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```

Review: Participation Constraints

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship.

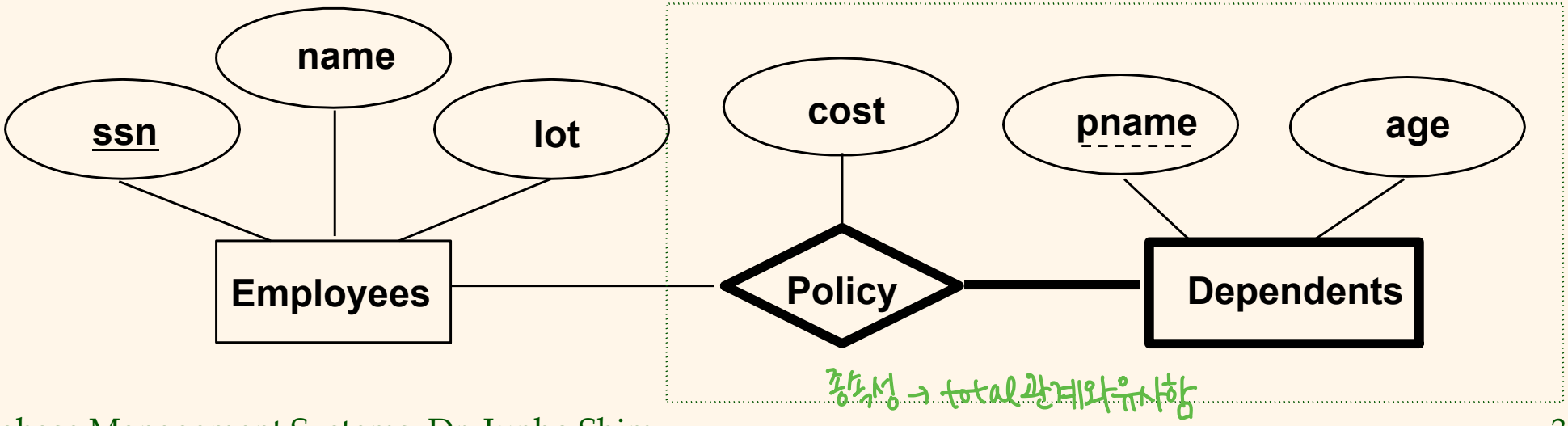
```
CREATE TABLE Dept_Mgr(
   did INTEGER,
   dname CHAR(20),
   budget REAL,
   ssn CHAR(11) NOT NULL,
   since DATE,
   PRIMARY KEY (did),
   FOREIGN KEY (ssn) REFERENCES Employees
      ON DELETE NO ACTION)
```

total 관계이므로

↳ 지울수없다! 고치든하고 지우셔야함

Review: Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (1 owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.

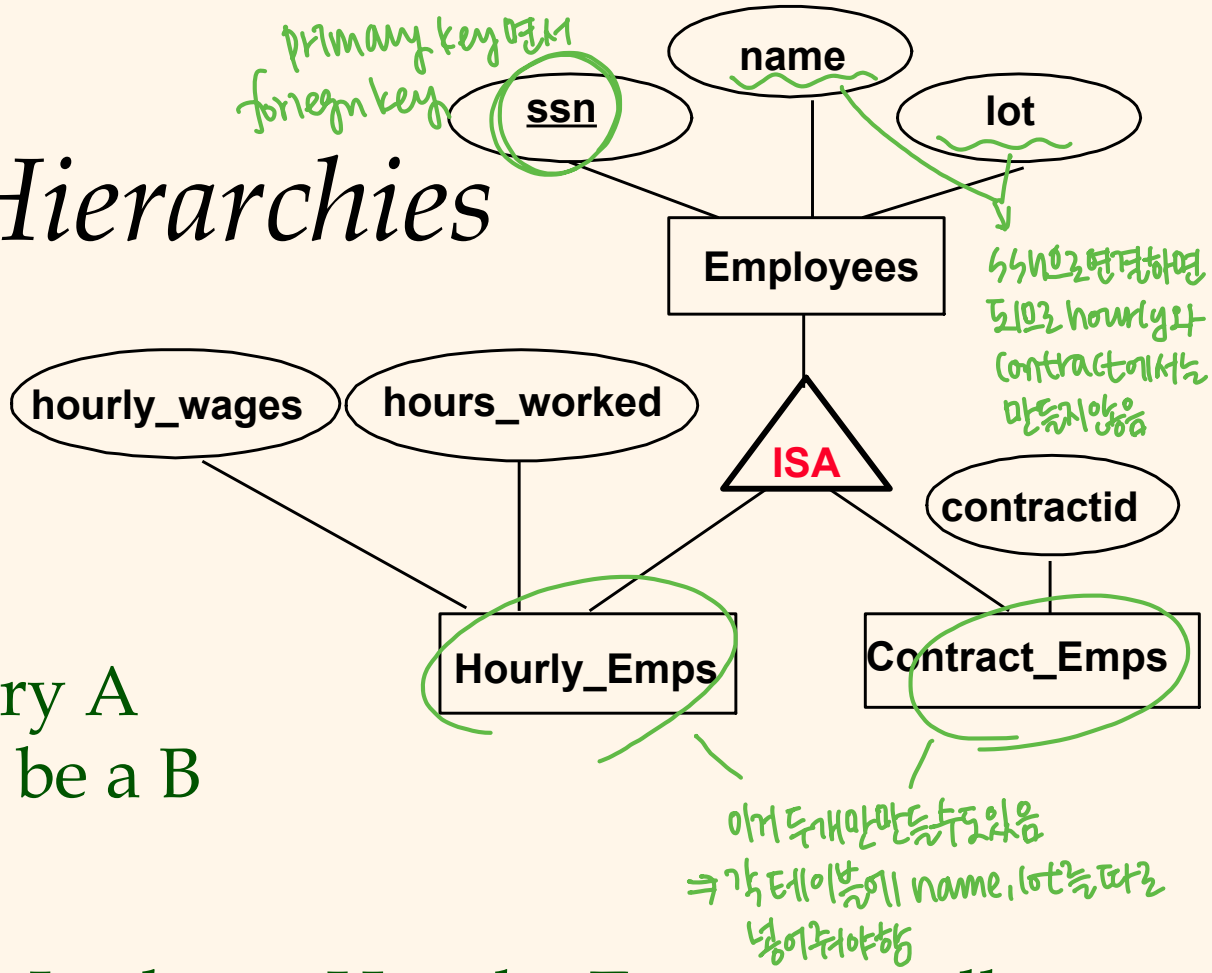


Translating Weak Entity Sets

- Weak entity set and identifying relationship set are translated into a single table.
  - When the owner entity is deleted, all owned weak entities must also be deleted.

```
CREATE TABLE Dep_Policy (
   pname  CHAR(20),
   age  INTEGER,
   cost  REAL,
   ssn  CHAR(11) NOT NULL,
   PRIMARY KEY  (pname, ssn),
   FOREIGN KEY  (ssn) REFERENCES Employees,
      ON DELETE CASCADE)
```

Review: ISA Hierarchies



- As in C++, or other PLs, attributes are inherited.
- If we declare A **ISA** B, every A entity is also considered to be a B entity.

- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)
- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (*Yes/no*)

Translating ISA Hierarchies to Relations

- ***General approach:***
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
    - *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages*, *hours_worked*, *ssn*); must delete Hourly_Emps tuple if referenced Employees tuple is deleted).
    - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.
- Alternative: Just Hourly_Emps and Contract_Emps.
  - *Hourly_Emps*: *ssn*, *name*, *lot*, *hourly_wages*, *hours_worked*.
  - Each employee must be in one of these two subclasses.

Relational Model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used.
- Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
  - Two important ICs: primary and foreign keys
  - In addition, we have domain constraints.
- Powerful and natural query languages exist.
- Rules to translate ER to relational model