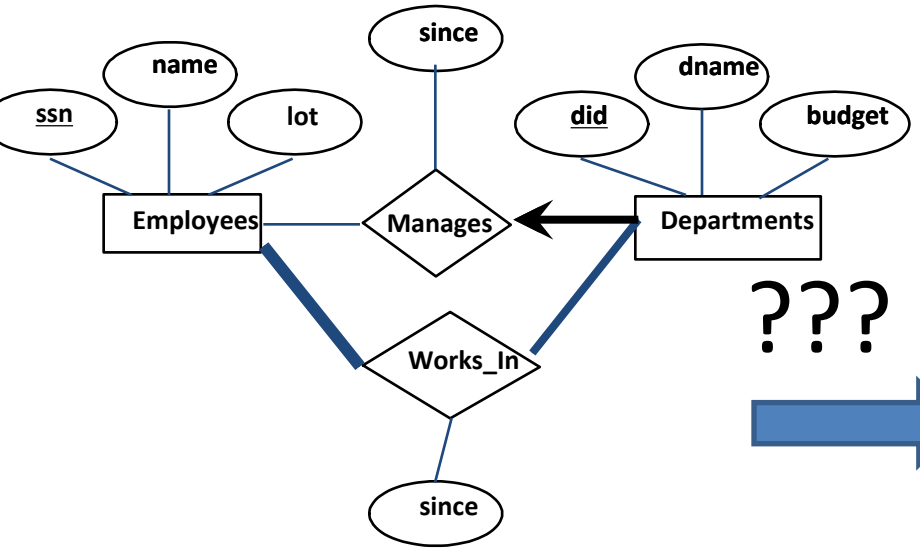
ER to Relational Mapping (Part I)

So Far We have



???

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

ER diagram

Relational table

- Question:
  - How can we go from the ER-diagrams to relational tables?

From ER Model to Relational Model

So… how do we convert an ER diagram into a table??
Simple!!

Basic Ideas:

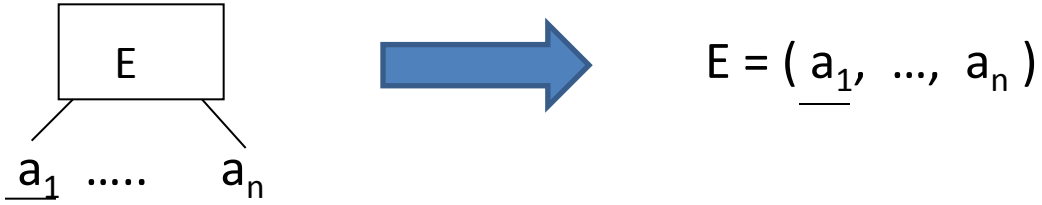
- Build a table for each entity set
- Build a table for each relationship set if necessary (more on this later)
- Decide the attributes of these tables
- Decide primary key and foreign keys of these tables

Today's lecture

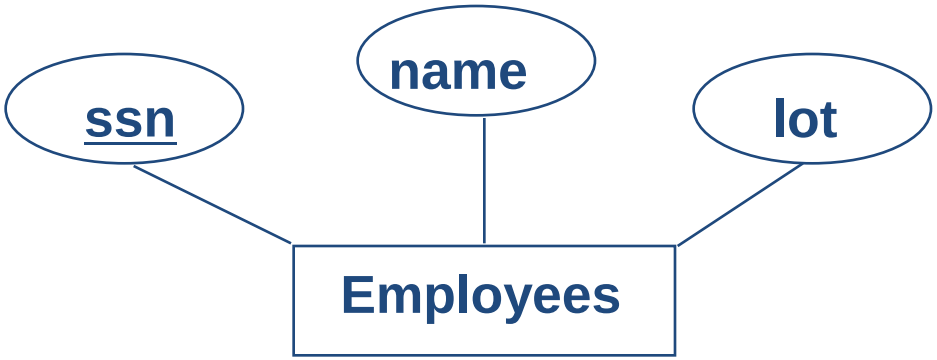
- We will learn how to translated ER diagrams into relational tables
  - Entity sets
    - Strong entity sets
    - Weak entity sets
  - Relationship sets
    - M:n cardinality constraint
    - 1:n cardinality constraint

Translating Strong Entity Sets

- Rule

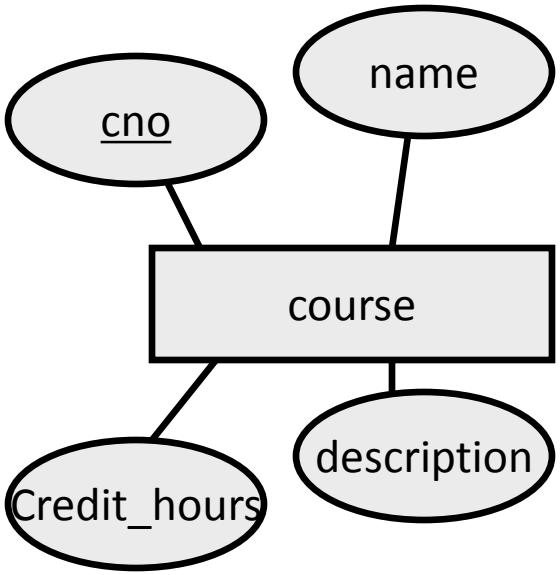


Schema: Employees (ssn, name, lot)




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

Exercise



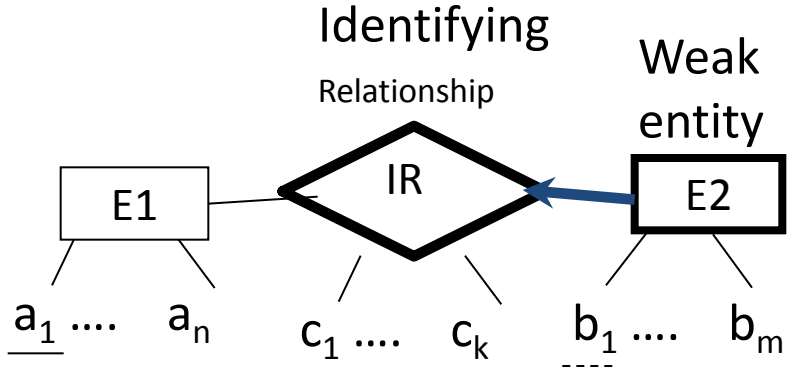
Translate this ER diagram to relational tables.

Schema: course (cno, name, credit_hours, description)

```
CREATE TABLE course
      (cno INTEGER,
       name CHAR(20),
       credit_hours  INTEGER,
       description CHAR(50),
       PRIMARY KEY  (cno)
);
```

Translating Weak Entity Sets

- Weak entity set and identifying relationship set together are translated into a single table.



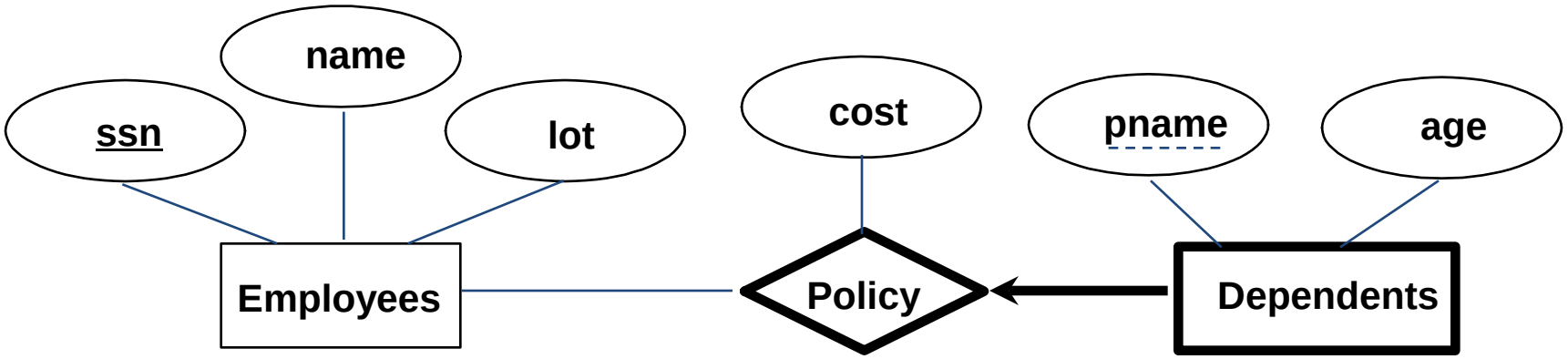
Strong entity: E1 = ($\underline{a_1}$, …, a_n)

Weak entity: E2 = ($\underline{a_1, b_1}$, …, b_m , c_1, …, c_k)

Foreign key of E2:

$\underline{a_1}$ (reference table E1)

Translating Weak Entity Sets: Example 1

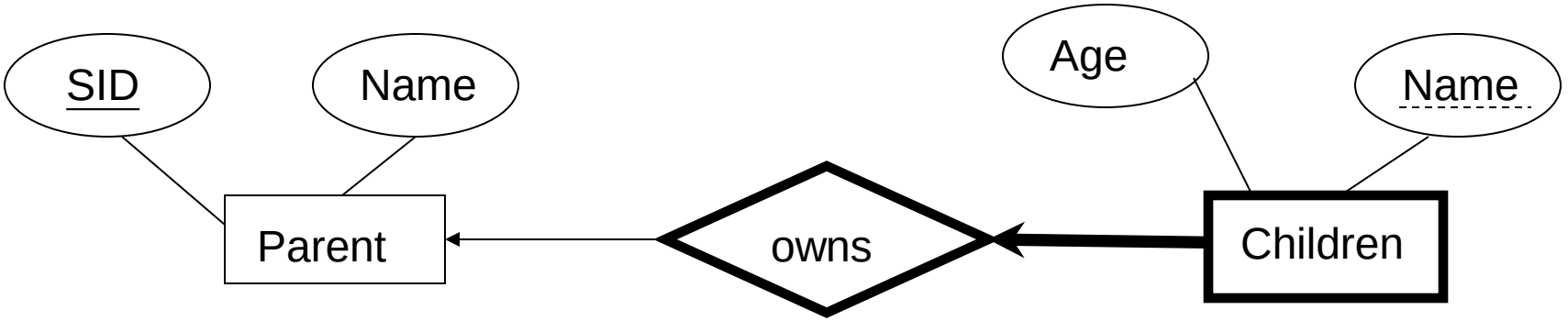


Translate this ER diagram to relational tables.

Schema:
Strong entity: ***Employees (ssn, name, lot),***
Weak entity: ***Dep_Policy(pname, ssn_employee, age, cost)***

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

Translating Weak Entity Sets: Example 2



Translate this ER diagram to relational tables.

Schema:

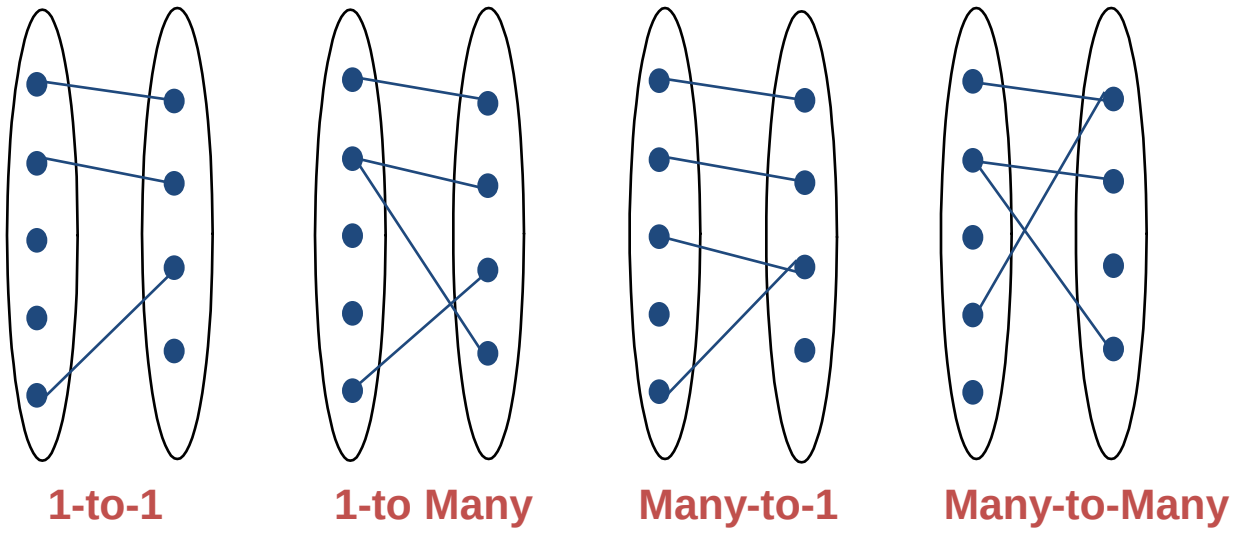
- Strong entity: Parent(SID, name)
- Weak entity: Child(name, SID, Age)

Today's lecture

- We will learn how to translated ER diagrams into relational tables
  - Entity sets
    - Strong entity sets
    - Weak entity sets
  - Relationship sets
    - M:n cardinality constraint
    - 1:n cardinality constraint
    - IsA hierarchy

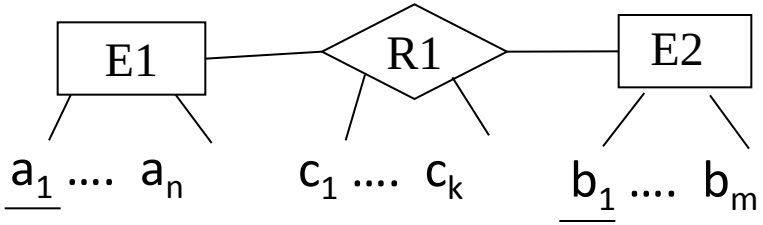
Translating Relationship Sets to Tables

- The translation scheme varies for different cardinality constraints.



Many-to-many Relationship Sets (Binary relationships)

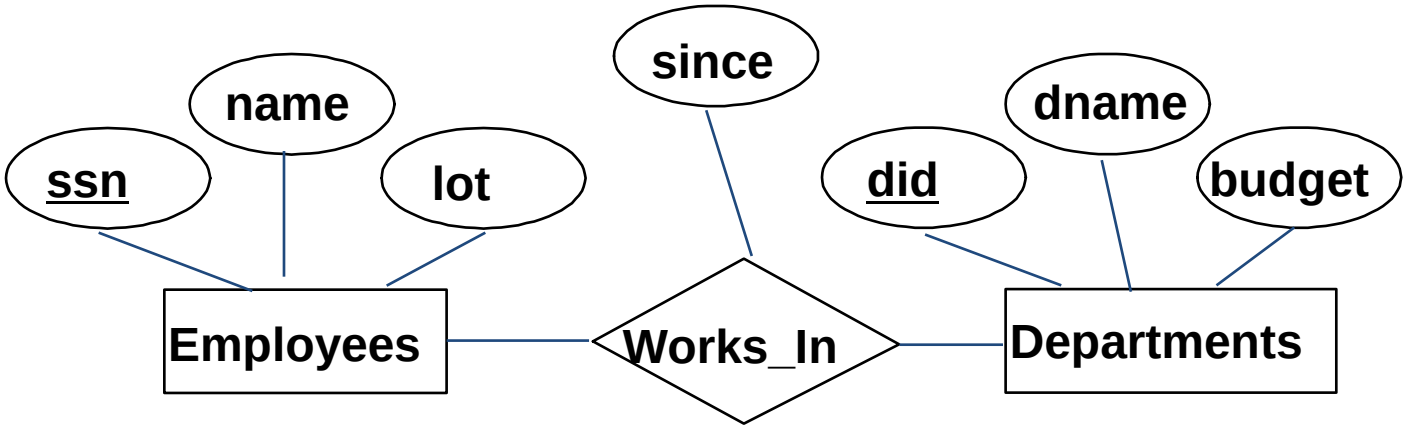
- In translating a M:N relationship set to a relation R1, R1 includes the following attributes:
  1. All attributes of the relationship set in ER diagram; and
  2. The primary key of each participating entity set.
     - This set of attributes forms the *primary key* for the relation.
     - These attributes are defined as the foreign keys of R1.



R1= (a_1, b_1, c_1, …, c_k)

Foreign keys of R1:
a_1 (reference table E1),
b_1 (reference table E2)

Example

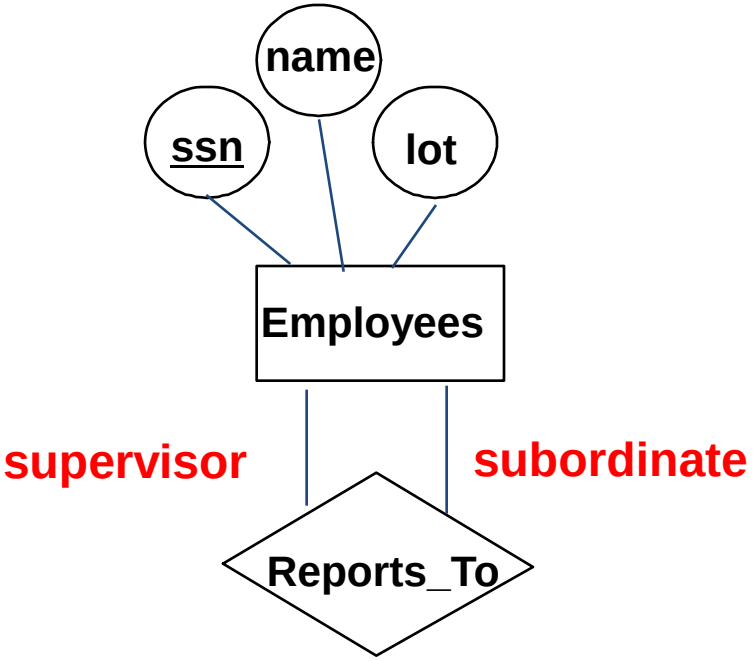


Construct the *Works_In* table:

Works_In(ssn, did, since)

```
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);
```

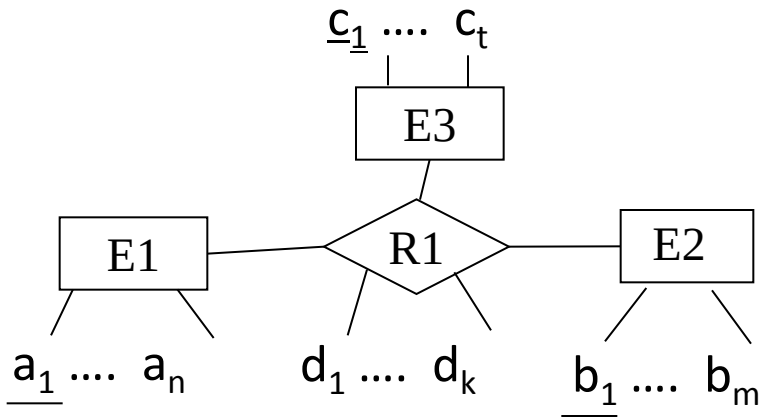
Relationship Sets with Different Roles




```
CREATE TABLE Reports_To(
  supervisor_ssn  CHAR(11),
  subordinate_ssn  CHAR(11),
  PRIMARY KEY (supervisor_ssn,
   subordinate_ssn),
  FOREIGN KEY (supervisor_ssn)
       REFERENCES Employees(ssn),
  FOREIGN KEY (subordinate_ssn)
       REFERENCES Employees(ssn)
);
```

Many-to-many Relationship Sets (N-ary Relationships)

Translating a M:N relationship set that involves more than 2 entity sets



R1= ($\underline{a_1}$, $\underline{b_1}$, $\underline{c_1}$, d1, …, d_k)

Foreign keys of R1:
$\underline{a_1}$ (reference table E1),
$\underline{b_1}$ (reference table E2),
$\underline{c_1}$ (reference table E3),
…

One-to-many Relationship Set

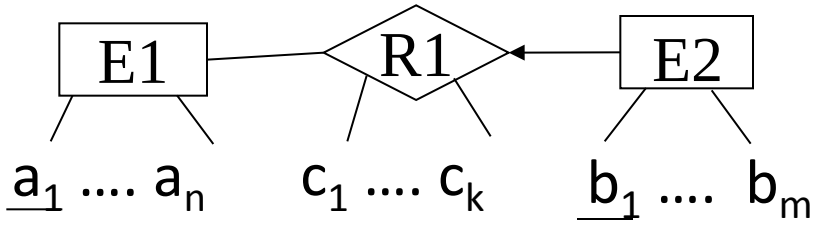
- In translating a 1:M relationship set to a relation R, R includes the following attributes:
  - All attributes of the relationship set; and
  - The primary key of each participating entity set (as foreign keys)

Same as m:n)

- Key of the relation:
  - The primary keys of all participating entity sets forms a *superkey* for the relation.
  - Pick the key of the entity set at the "many" side as the primary key of the relation

Diff. from m:n)

So We have:

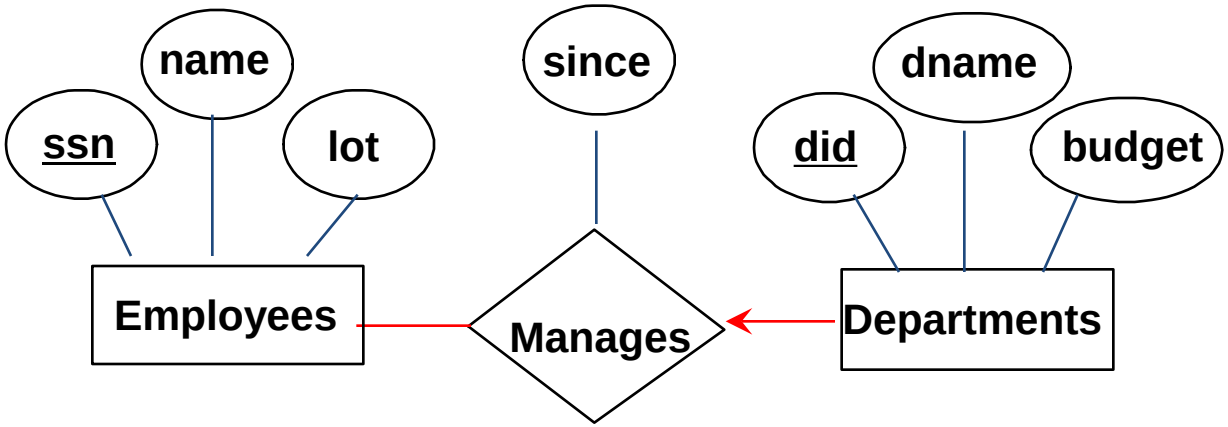


Could have :

$R1 = (a_1, \underline{b_1}, c_1, \ldots, c_k)$

Only b1 is used as the key!

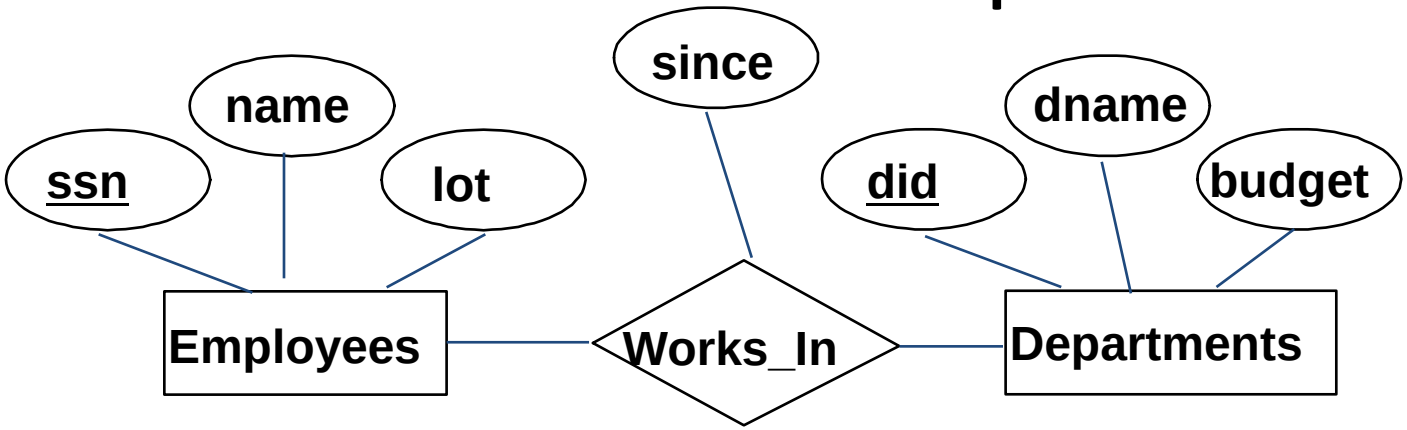
Example




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

Note that did is the key now!

Revisit the Example of many-many Relationship

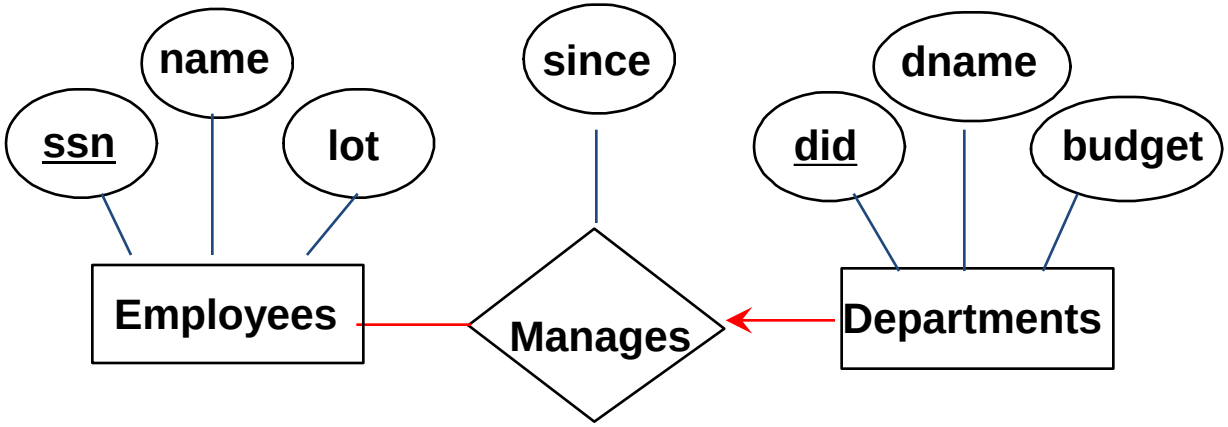


How to create the Works_In table:

```
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)
```

SSN	did	Since
123-22-3666	1	01/01/2010
123-22-3666	2	02/03/2010
231-31-5368	1	01/01/2010

Why SSN can be removed from the Key in the 1-many relationship?



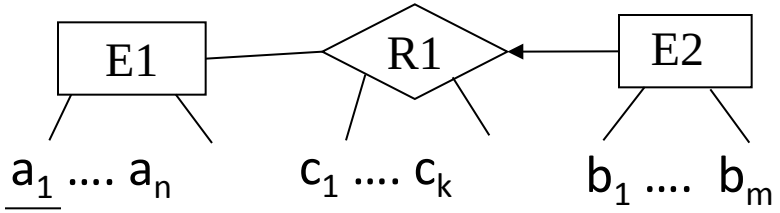
SSN	did	Since
123-22-3666	1	01/01/2010
123-22-3666	2	02/03/2010
231-31-5368	3	01/01/2010

Manages Relationship
(1-to-many relationship)
key: (did)

SSN	did	Since
123-22-3666	1	01/01/2010
123-22-3666	2	02/03/2010
231-31-5368	1	01/01/2010

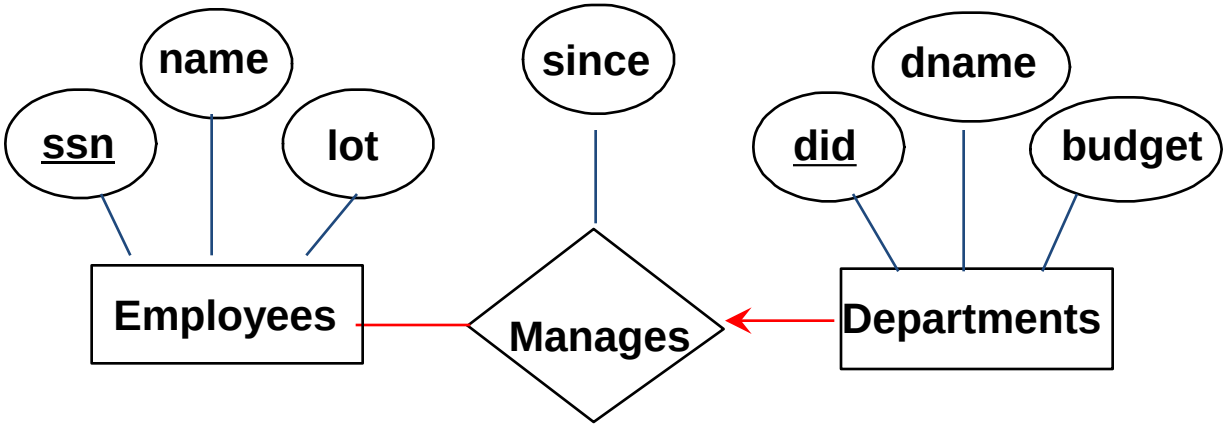
Works-in Relationship
(Many-to-many relationship)
Key: (SSN, did)

An Alternative to Translation of 1-many Relationship



- Do not construct a table for R1
- Instead, add a1, c1, …., ck to E2
  - E2=($\underline{b_1}$, …., b_m, a_1, c_1, …, c_k)
  - Foreign key of E2: a1(reference E1).

Example of the Alternative Translation Scheme



Option 1: Dept_Mgr (ssn, did, since)

Departments (did, dname, budget)

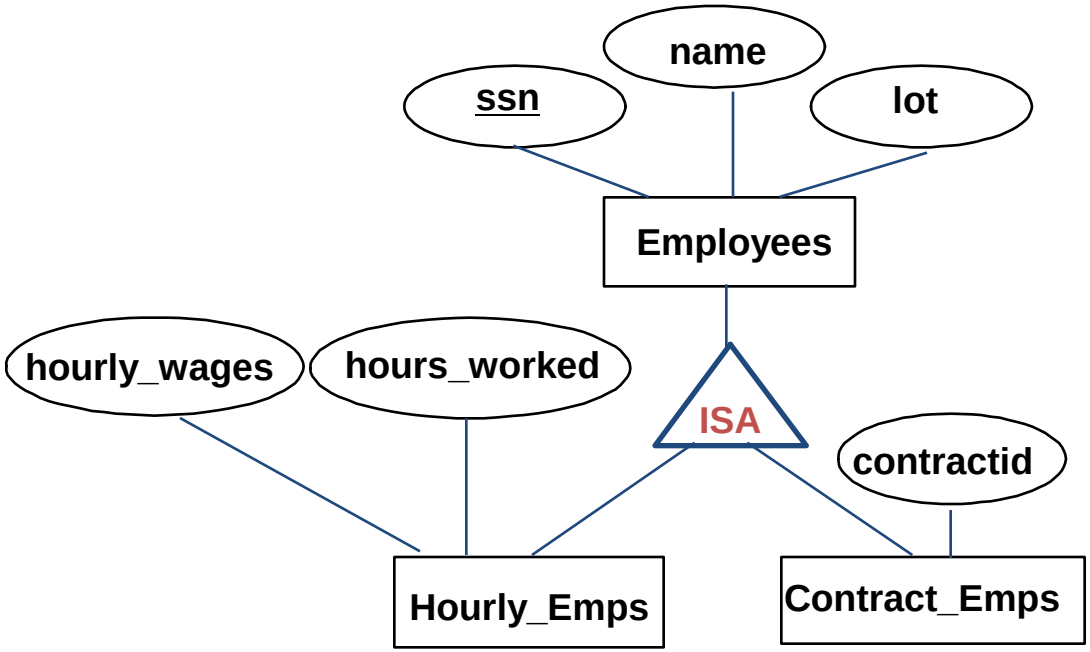
Option 2: Departments (did, dname, budget, ssn, since)

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

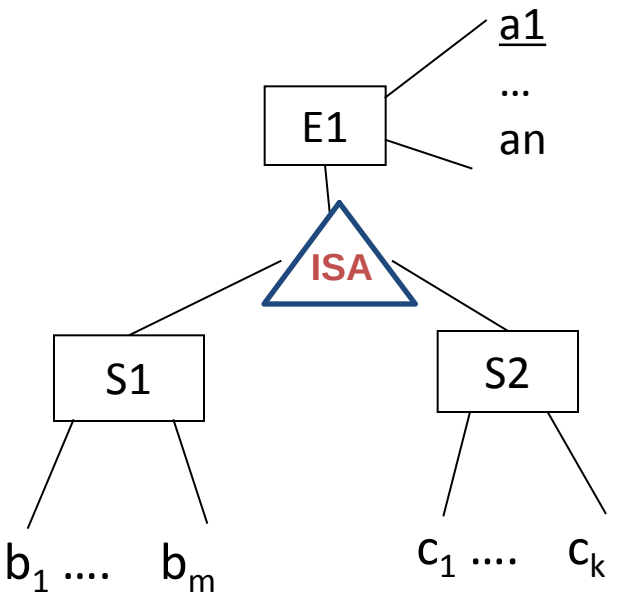
did	dname	budget	SSN	Since
1	HR	20000	12345678	01/01/2010
2	Marketing	400000	12345678	02/03/2010
3	IT	300000	13452121	01/01/2010

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.



Translating ISA Hierarchies to Relations



Method 1:

$E = (\underline{a_1}, \ldots, a_n)$

$S1 = (\underline{a_1}, b_1, \ldots, b_m)$

$S2 = (\underline{a_1}, c_1 \ldots, c_k)$

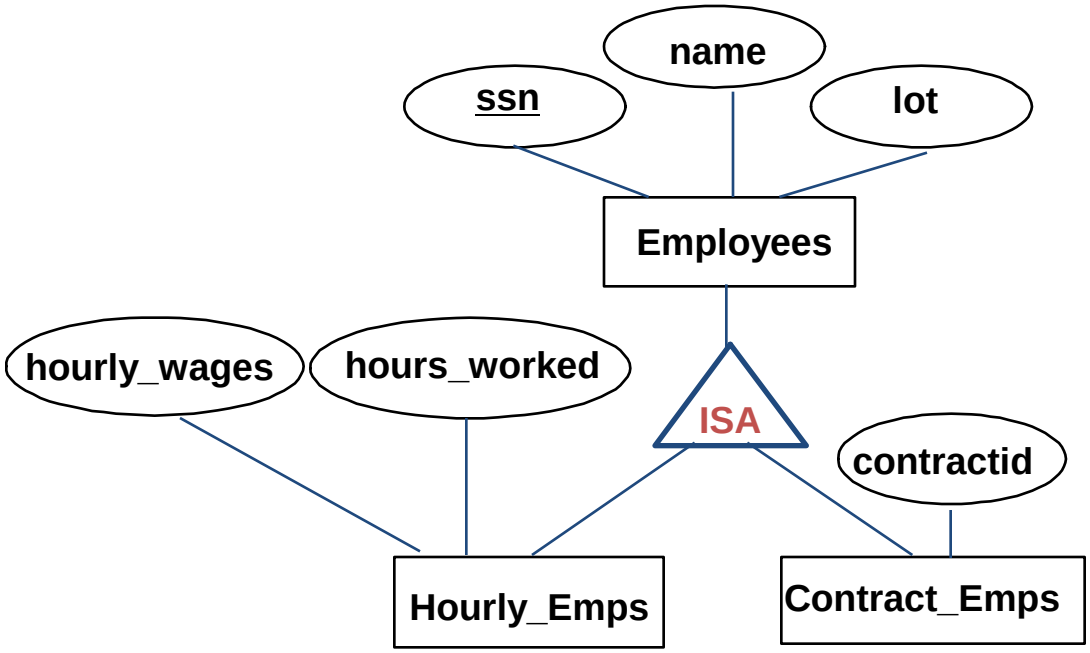
Method 2:

$S1 = (\underline{a_1}, \ldots, a_n, b_1, \ldots, b_m)$

$S2 = (\underline{a_1}, \ldots, a_n, c_1 \ldots, c_k)$

Q: When method 2 is wrong?
(tip: think about the ***covering constraint***)

Example: Translating ISA Hierarchies to Relations



- *Approach 1:*
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
- Approach 2: Just Hourly_Emps and Contract_Emps.