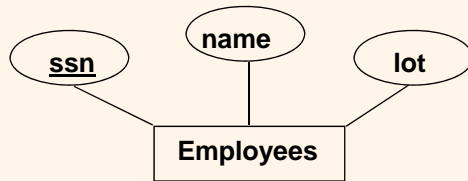


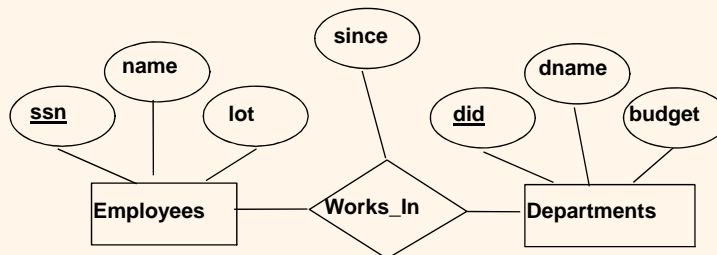
Logical DB Design: ER to Relational

❖ Entity sets to tables:

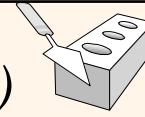


```
CREATE TABLE Employees
(ssn CHAR(11),
 name CHAR(20),
 lot INTEGER,
 PRIMARY KEY (ssn))
```

Relationship Sets to Tables



Relationship Sets to Tables (Cont.)

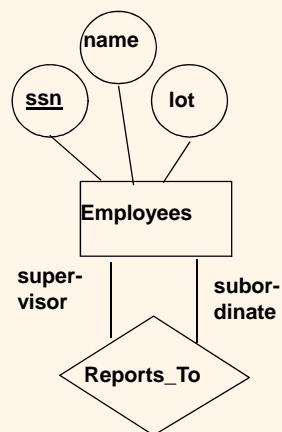
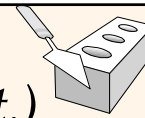


❖ In translating a relationship set to a relation, attributes of the relation must include:

- Keys for each participating entity set (as foreign keys).
- 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)
```

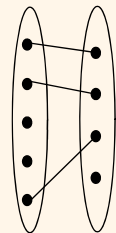
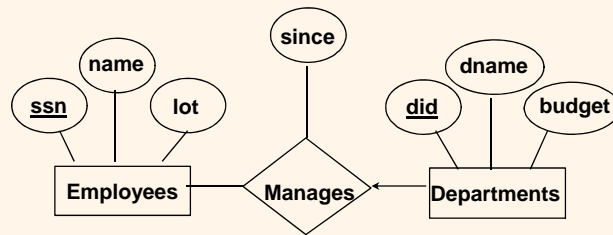
Relationship Sets to Tables (Cont.)



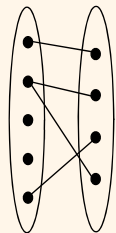
```
CREATE TABLE Reports_To (  
  supervisor_ssn CHAR(11),  
  subordinate_ssn CHAR(11),  
  PRIMARY KEY (supervisor_ssn,  
  subordinate_ssn),  
  FOREIGN KEY (supervisor_ssn)  
    REFERENCES Employees (ssn),  
  FOREIGN KEY (subordinate_ssn)  
    REFERENCES Employees (ssn))
```

Review: Key Constraints

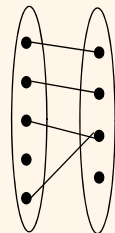
- Each dept has at most one manager, according to the key constraint on Manages.



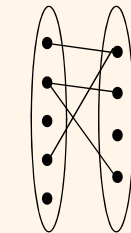
1-to-1



1-to Many



Many-to-1



Many-to-Many

Translation to relational model?

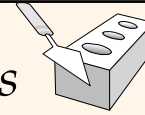
Translating ER Diagrams with Key Constraints

- Map relationship to a table:
 - Note that **did** is the key now!
 - Separate tables for Employees and Departments.
- Since each department has a unique manager, we could instead combine Manages 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)
```

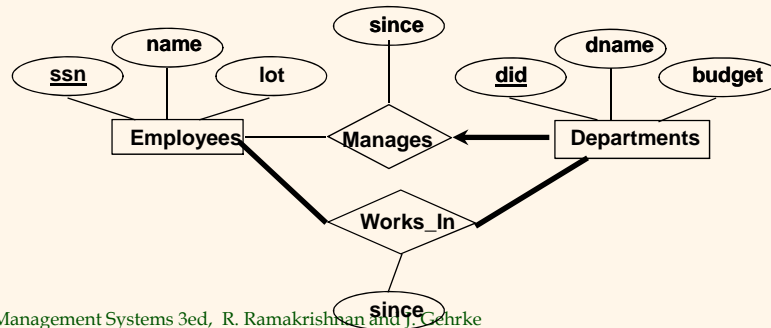
```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```

Review: Participation Constraints

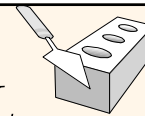


❖ Does every department have a manager?

- 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

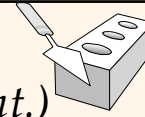


- ### ❖ We can capture participation constraints involving **one** entity set in a binary relationship, but little else.

```
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)
```

Not CASCADE?

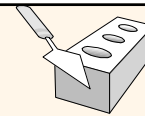
Participation Constraints in SQL (Cont.)



❖ Can we capture participation constraints in the Works_In relationship?

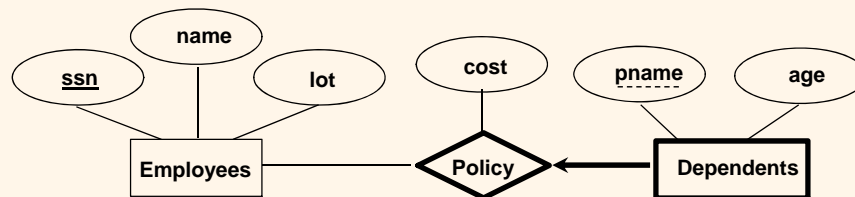
- No. We need to use *table constraints* or *assertions*.
- Table constraints and assertions can be specified using the full power of the SQL query language and are very expressive but also very expensive to check and enforce (more details in Chapter 5).

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.



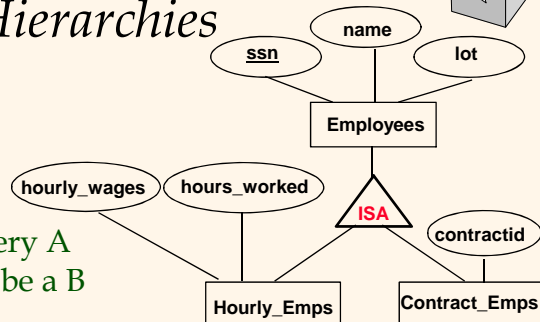
Translating Weak Entity Sets

- ❖ 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)  
  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*)

They are usually expressed in SQL by using assertions.

Translating ISA Hierarchies to Relations

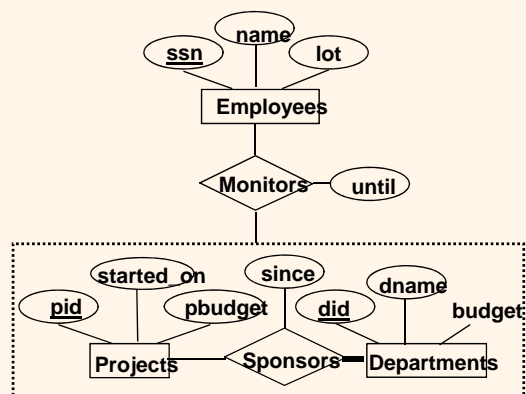
❖ **General approach:**

- 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.**

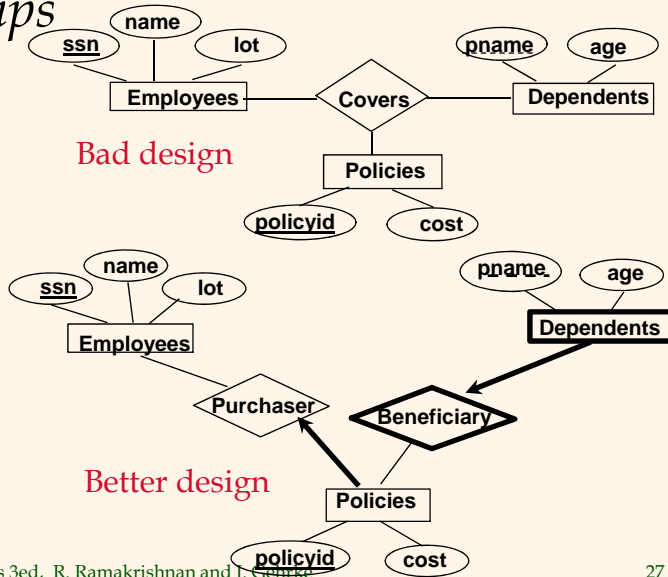
- **Hourly_Emps**: *ssn*, *name*, *lot*, *hourly_wages*, *hours_worked*.
- Each employee must be in one of these two subclasses.

Translating ER Diagrams with Aggregation



For the **Monitors** relationship set, we create a relation with the following attributes: *ssn* (**Employees**), *did*, *pid* (**Sponsors**), and the descriptive attribute of **Monitors** (*until*).

Review: Binary vs. Ternary Relationships



Binary vs. Ternary Relationships (Contd.)

- ❖ The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.

```
CREATE TABLE Policies (
  policyid INTEGER,
  cost REAL,
  ssn CHAR(11) NOT NULL,
  PRIMARY KEY (policyid),
  FOREIGN KEY (ssn) REFERENCES Employees,
  ON DELETE CASCADE)
```

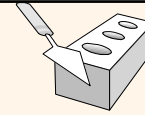
- ❖ Participation constraints lead to NOT NULL constraints.

```
CREATE TABLE Dependents (
```

- ❖ What if Policies is a weak entity set?

```
  pname CHAR(20),
  age INTEGER,
  policyid INTEGER,
  PRIMARY KEY (pname, policyid),
  FOREIGN KEY (policyid) REFERENCES Policies,
  ON DELETE CASCADE)
```

Views

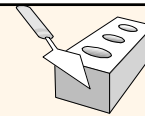


- ❖ A view is just a relation, but we store a *definition*, rather than a set of tuples.

```
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
```

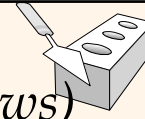
- ❖ Views can be dropped using the **DROP VIEW** command.
 - How to handle **DROP TABLE** if there's a view on the table?
 - DROP TABLE command has options to let the user specify this.

Views and Security



- ❖ Views can be used to present necessary information (or a **summary**), while hiding details in underlying relation(s).
 - Given YoungActiveStudents, but not Students or Enrolled, we can find students who have enrolled, but not the *cid*'s of the courses they are enrolled in.

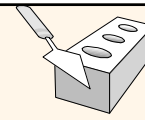
Updates on Views (updatable views)



```
CREATE VIEW GoodStudents (sid, gpa)
AS SELECT S.sid, S.gpa
FROM Students S
WHERE S.gpa > 3.0
```

```
CREATE VIEW GoodStudents2 (sname, gpa)
AS SELECT S.name, S.gpa
FROM Students S
WHERE S.gpa > 3.0
```

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 *always* have domain constraints.
- ❖ Powerful and natural query languages exist.
- ❖ Rules to translate ER to relational model