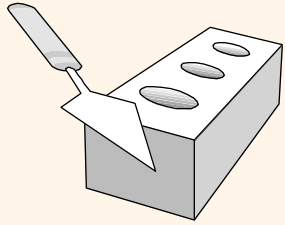
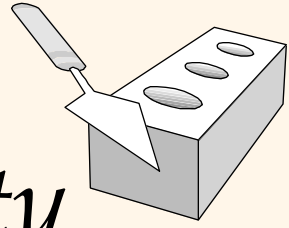


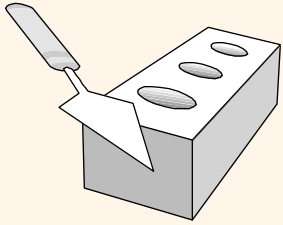
The Relational Model

Chapter 3

Foreign Keys, Referential Integrity



- *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.) Like a `logical pointer'.
- E.g. *sid* is a foreign key referring to Students:
  - Enrolled(*sid*: string, *cid*: string, *grade*: string)
  - If all foreign key constraints are enforced, *referential integrity* is achieved, i.e., no dangling references.



Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses.

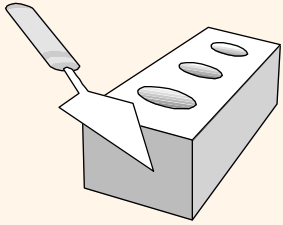
```
CREATE TABLE Enrolled  (sid CHAR(20),
                        cid CHAR(20),
                        grade CHAR(2),
                        PRIMARY KEY  (sid,cid),
                        FOREIGN KEY (sid) REFERENCES Students )
```

Enrolled

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

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

Enrolled

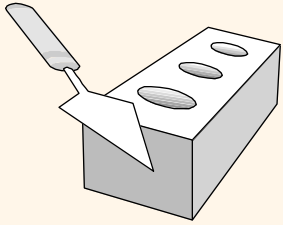
sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

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

- What should be done if an Enrolled tuple with a non-existent student id is inserted?
  - INSERT INTO Enrolled (sid, cid, grade)
    VALUES ('53611', 'History105', 'A')
  - *(Reject it!)*

Enforcing Referential Integrity



Enrolled

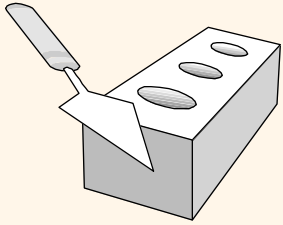
sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

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

- What should be done if a Students tuple is deleted?
  - OPTION 1: Also delete all Enrolled tuples that refer to it.
  - OPTION 2: Disallow deletion of a Students tuple that is referred to.
  - OPTION 3: Set sid in Enrolled tuples that refer to it to a *default sid*.
  - OPTION 4: Set sid in Enrolled tuples that refer to it to *null*.
- Similar if primary key of Students tuple is updated.

Referential Integrity in SQL

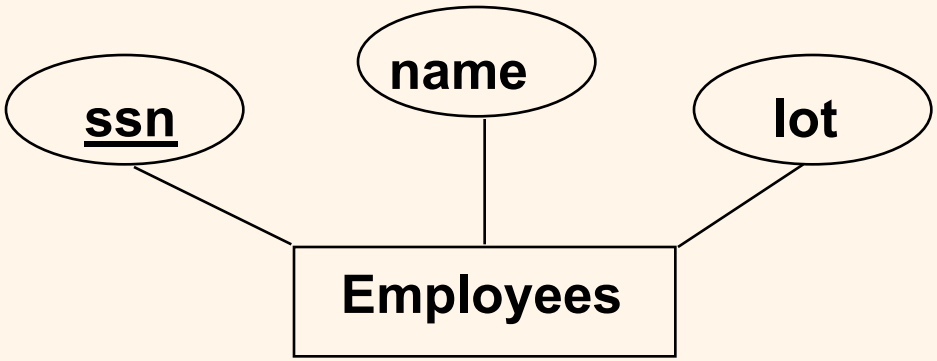


- SQL/92 and SQL:1999 support all four options on deletes and updates.
  - Default is NO ACTION (*delete/update is rejected*)
  - CASCADE (also delete all tuples that refer to deleted tuple)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

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

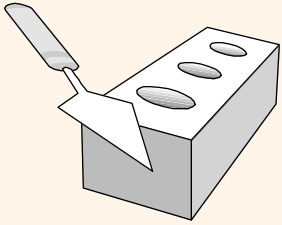
Logical DB Design: ER to Relational

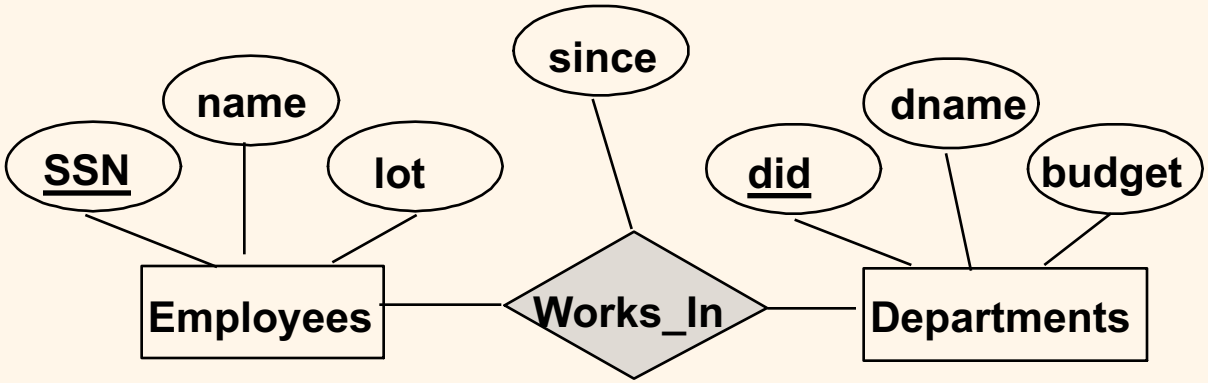
- Entity sets to tables:




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

Relationship Sets to Tables

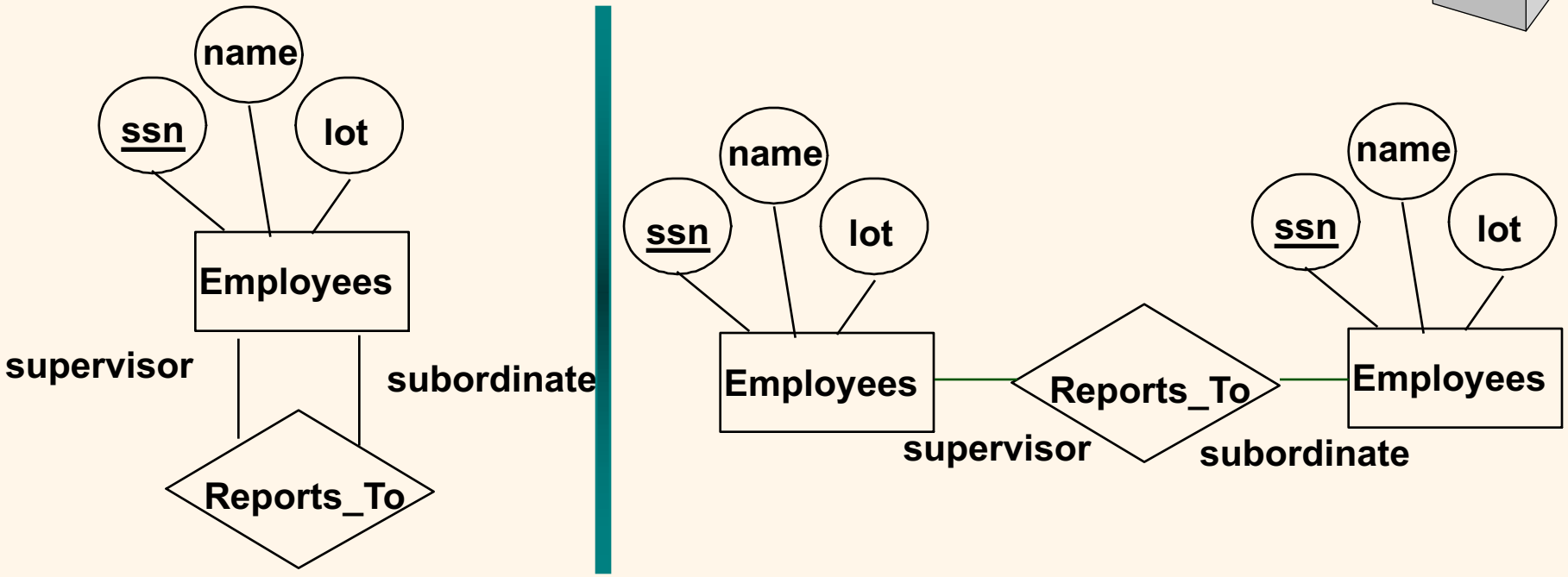




- In translating a relationship set to a relation, attributes of the relation must include:
  - Keys for each participating entity set (as foreign keys).
  - All descriptive attributes.

```
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)
```

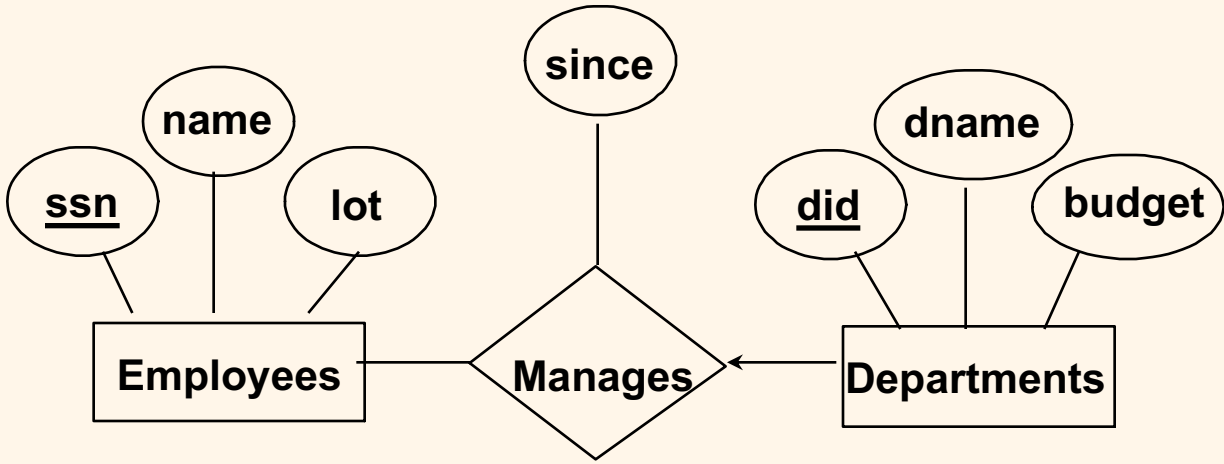
Relationship Sets to Tables

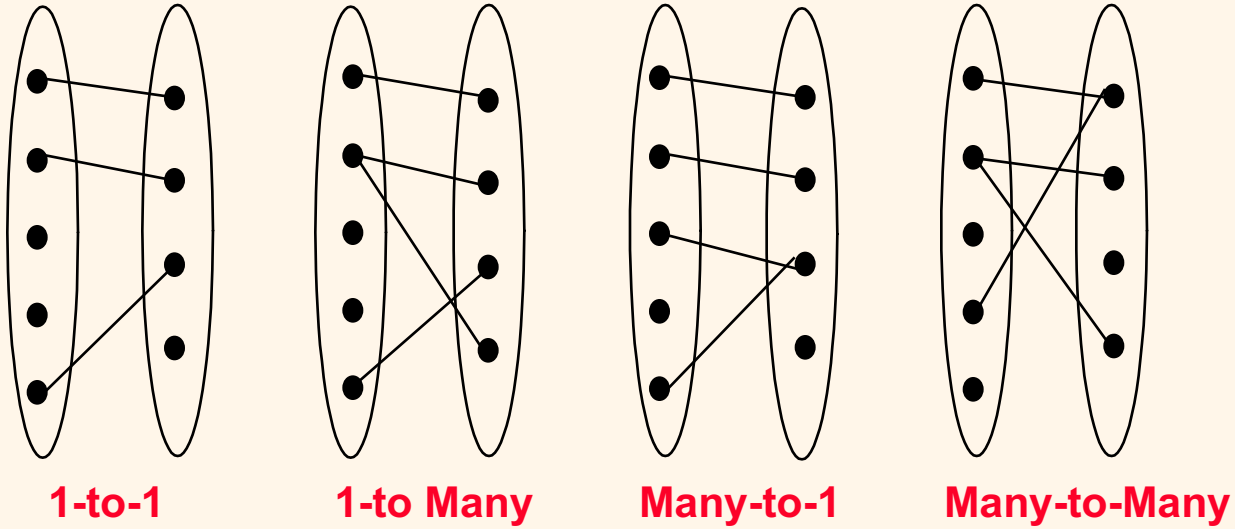



```
CREATE TABLE Reports_To( supervisor_ssn  CHAR(11),
                         subordinate_ssn  CHAR(11),
    PRIMARY KEY (supervisor_ssn, subordinate_ssn),
    FOREIGN KEY (supervisor_ssn)   REFERENCES Employees(ssn),
    FOREIGN KEY (subordinate_ssn) REFERENCES Employees(ssn))
```

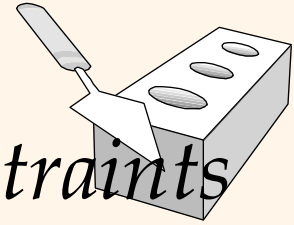
Review: Key Constraints

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





Translation to relational model?



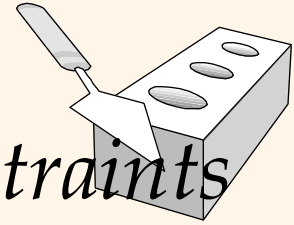
Translating ER Diagrams with Key Constraints

- Map relationship to a table:
  - Separate tables for Employees and 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)
```

- Note that did is the key now!

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

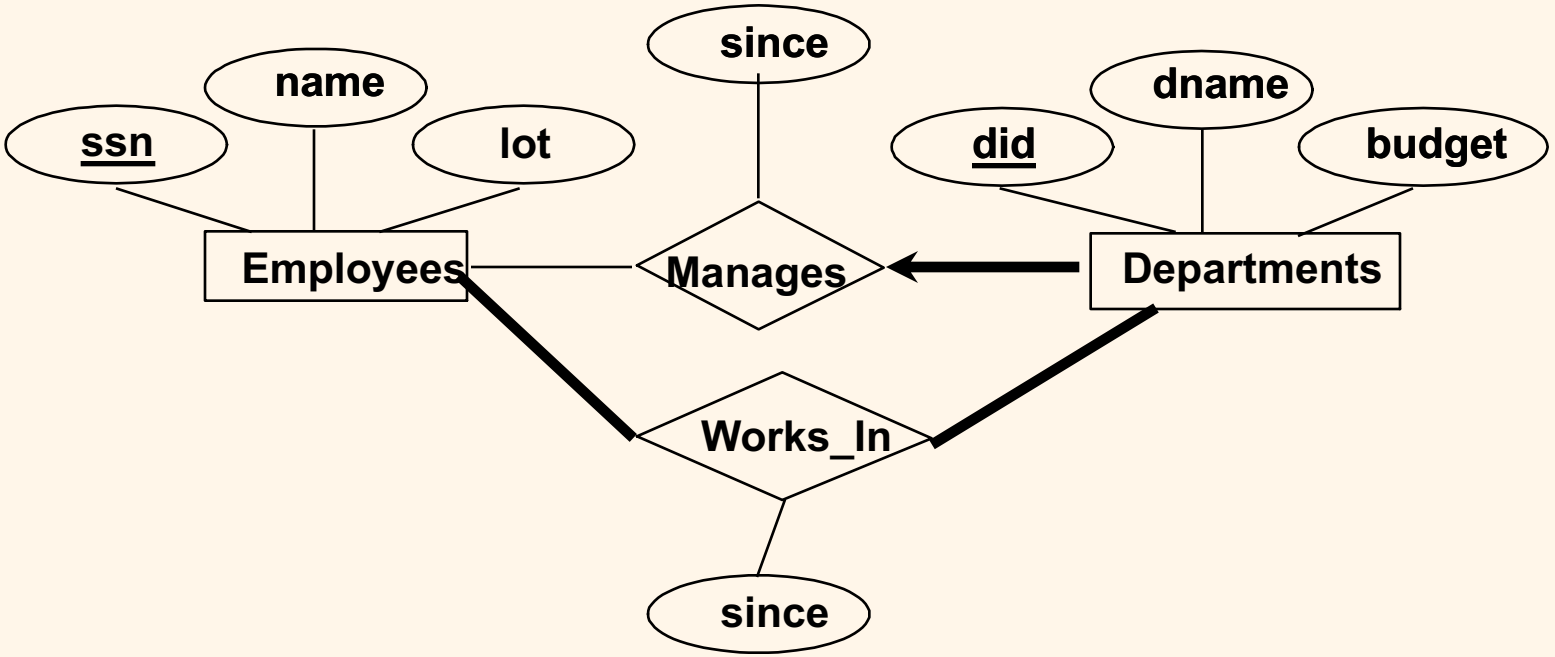


Translating ER Diagrams with Key Constraints

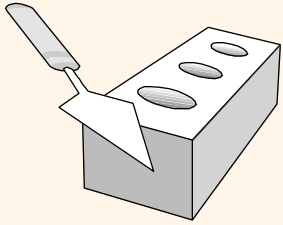
- Since each department has a unique manager, we could instead combine Manages and Departments.

```
CREATE TABLE  Dept_Mgr(
   did  INTEGER,
   dname  CHAR(20),
   budget  REAL,
   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!)

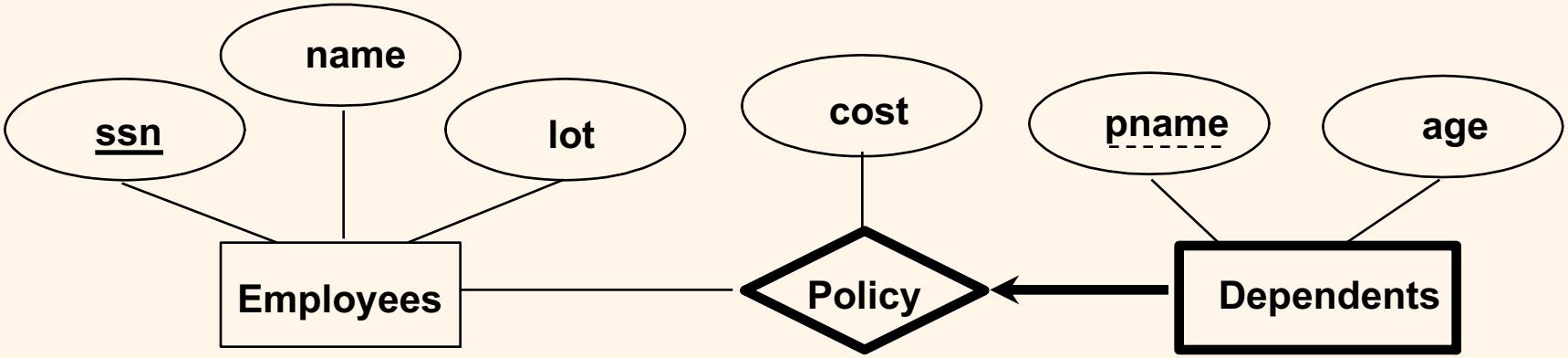


Participation Constraints in SQL

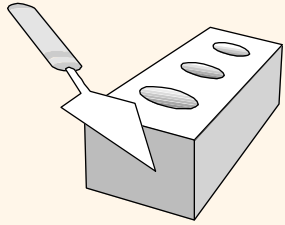
- We can capture participation constraints involving one entity set in a binary relationship.

```
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)
```

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.

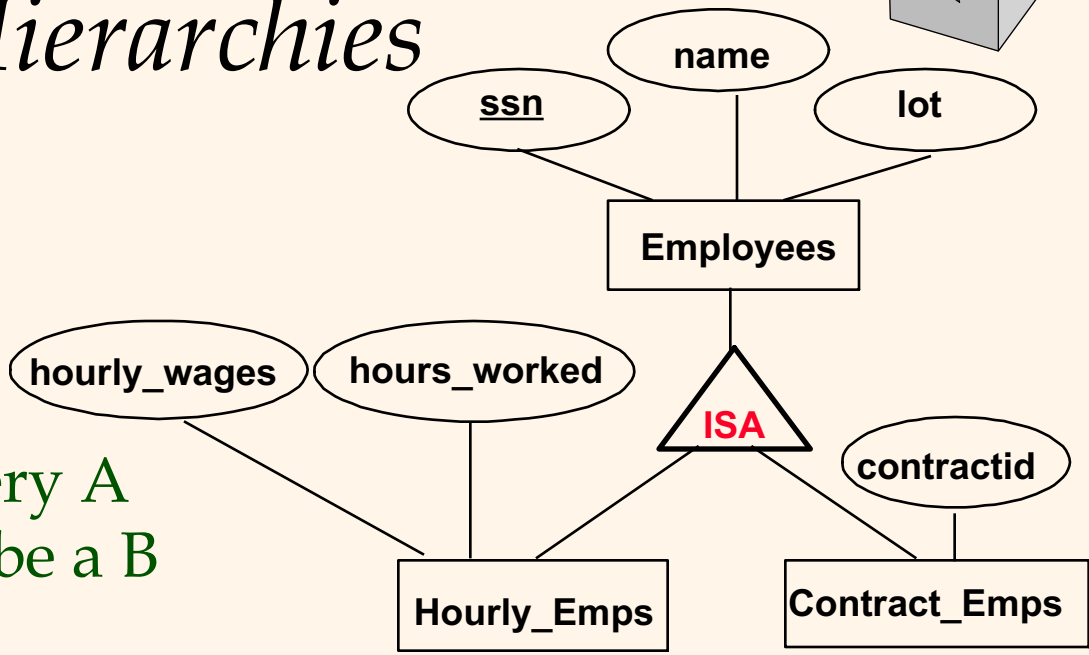


Translating Weak Entity Sets

- 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.

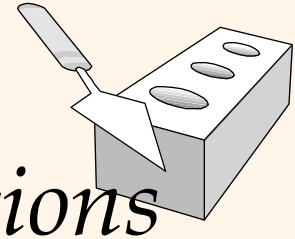
```
CREATE TABLE Dep_Policy (
   pname  CHAR(20),
   age  INTEGER,
   cost  REAL,
   ssn  CHAR(11) NOT NULL,
   PRIMARY KEY  (pname, ssn),
   FOREIGN KEY  (ssn) REFERENCES Employees
      ON DELETE CASCADE)
```

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.

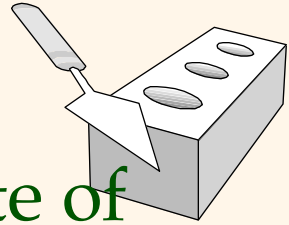
- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)
  - Hourly_Emps OVERLAPS Senior_Emps
- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (*Yes/no*)
  - Hourly_Emps AND Contract_Emps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



Translating ISA Hierarchies to Relations

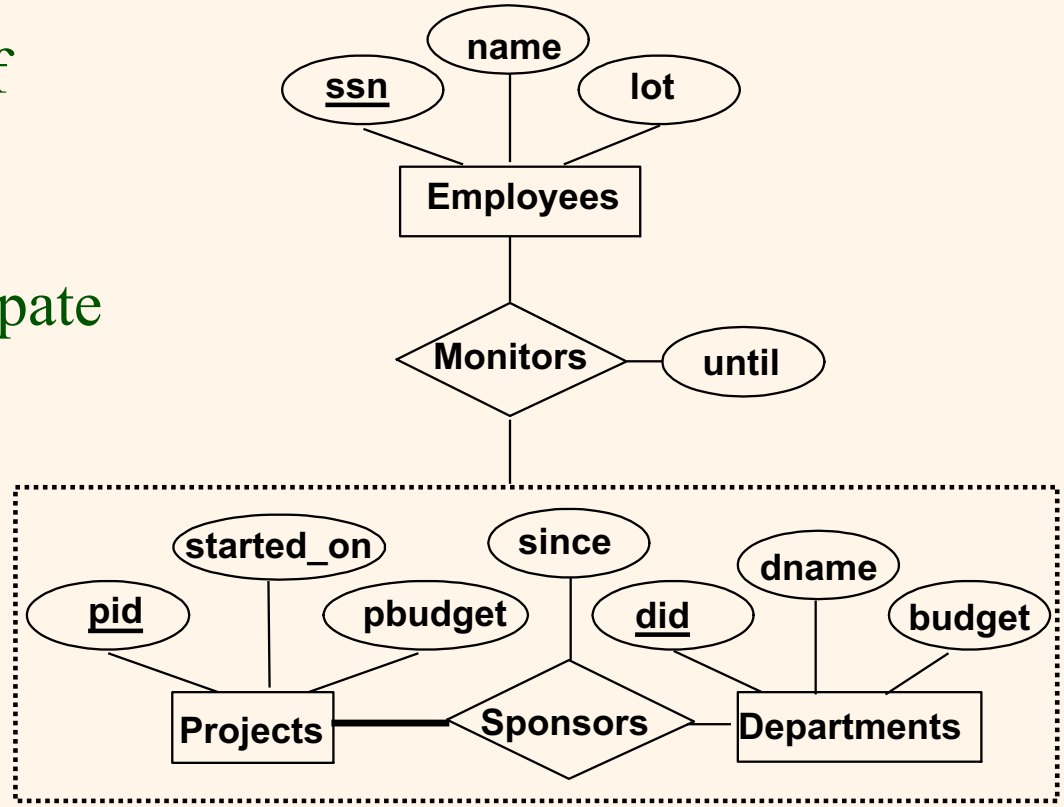
- ***General approach:***
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
    - *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages*, *hours_worked*, *ssn*); must delete Hourly_Emps tuple if referenced Employees tuple is 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.

Aggregation



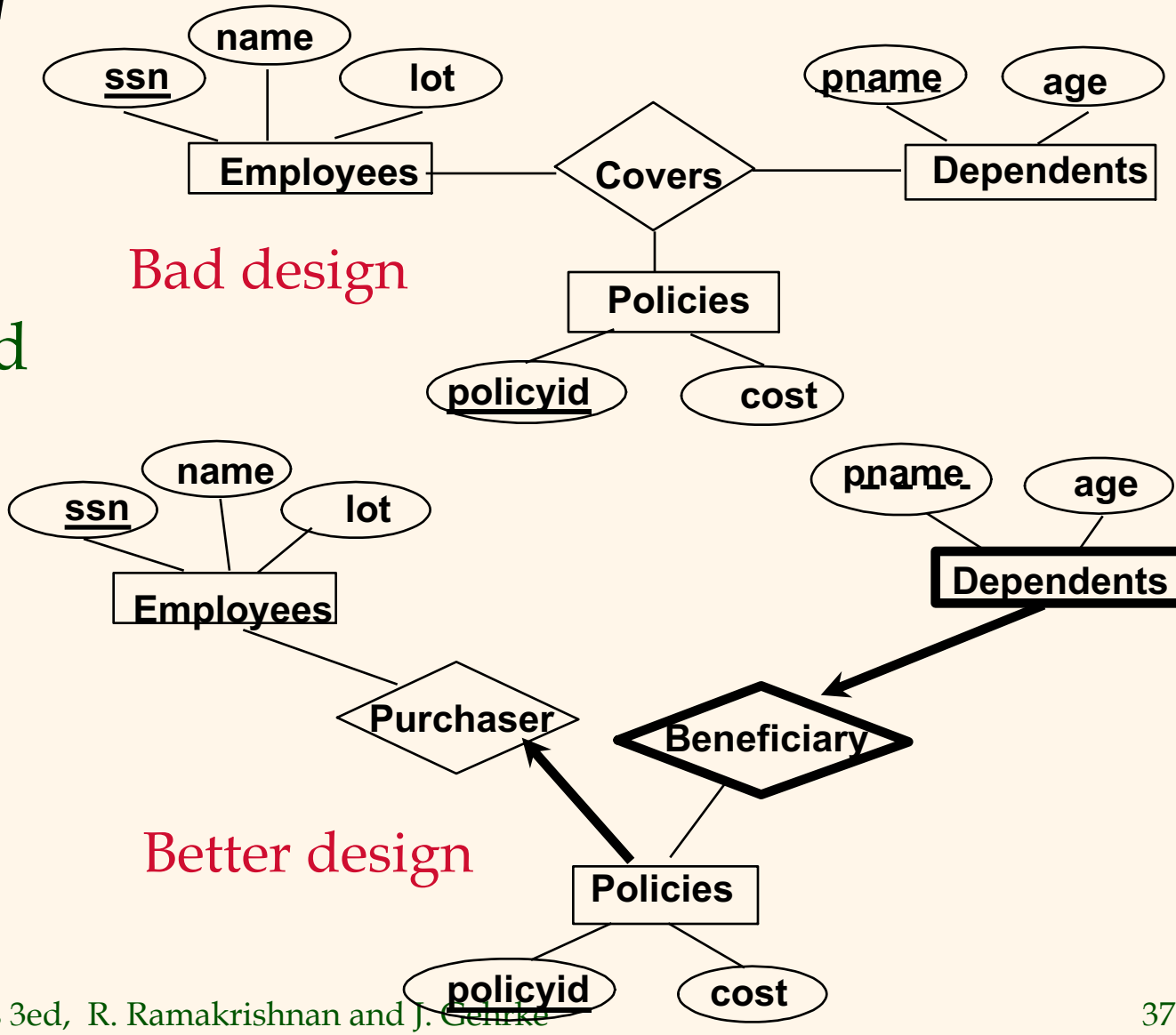
- *Monitors:* Create a relation with the key attribute of
  - "Employee" (ssn)
  - "Sponsors" (did,pid), and
  - Attribute of "Monitors" (until)

- The descriptive attribute of "sponsors" is not included
- Not every (pid,did) in the "Sponsors" relation participate in the "Monitors" relation

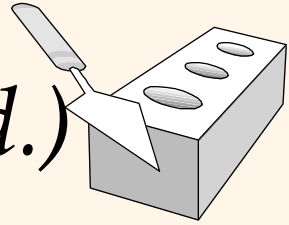


Review: Binary vs. Ternary Relationships

- If each policy is owned by just one employee, and each dependent is tied to the covering policy, first diagram is inaccurate.



Binary vs. Ternary Relationships (Contd.)

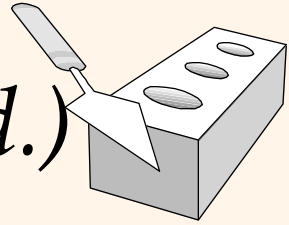


- The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.
- Participation constraints lead to NOT NULL constraints.

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

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

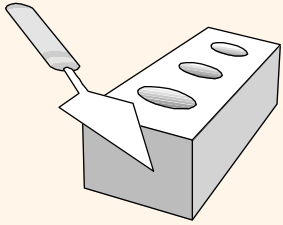
Binary vs. Ternary Relationships (Contd.)



- What if Policies is a weak entity set?
  - Cascaded weak entities

```
CREATE TABLE Policies (
    policyid  INTEGER,
    cost  REAL,
    ssn  CHAR(11),
    PRIMARY KEY (policyid,ssn).
    FOREIGN KEY (ssn) REFERENCES
    Employees ON DELETE CASCADE)
```

```
CREATE TABLE Dependents (
    pname  CHAR(20),
    ssn  CHAR(11),
    age  INTEGER,
    policyid  INTEGER,
    PRIMARY KEY (pname, policyid, ssn)
    FOREIGN KEY (policyid, ssn) REFERENCES Policies
    ON DELETE CASCADE)
```

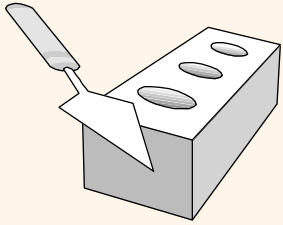


Views

- 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?
  - A couple of options

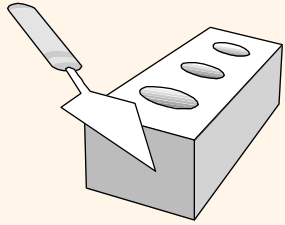


Updates on Views

- SQL-92 allows updates to be specified only on views defined on a single table

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

- Deleting a tuple from GoodStudent would delete the corresponding tuple from the original table
- What would happen if "sid" is replaced by "sname" in the view

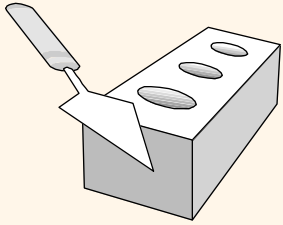


Updates on Views

- A tuple can be inserted into the view

```
INSERT INTO GoodStudents (sid, gpa)
VALUES (34261,3.5)
```

- What would happen to the other fields ➔ NULL
- Can we insert a GPA less than 3 ➔ YES
- To prevent this ➔ Add "WITH CHECK OPTION" to the view definition



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