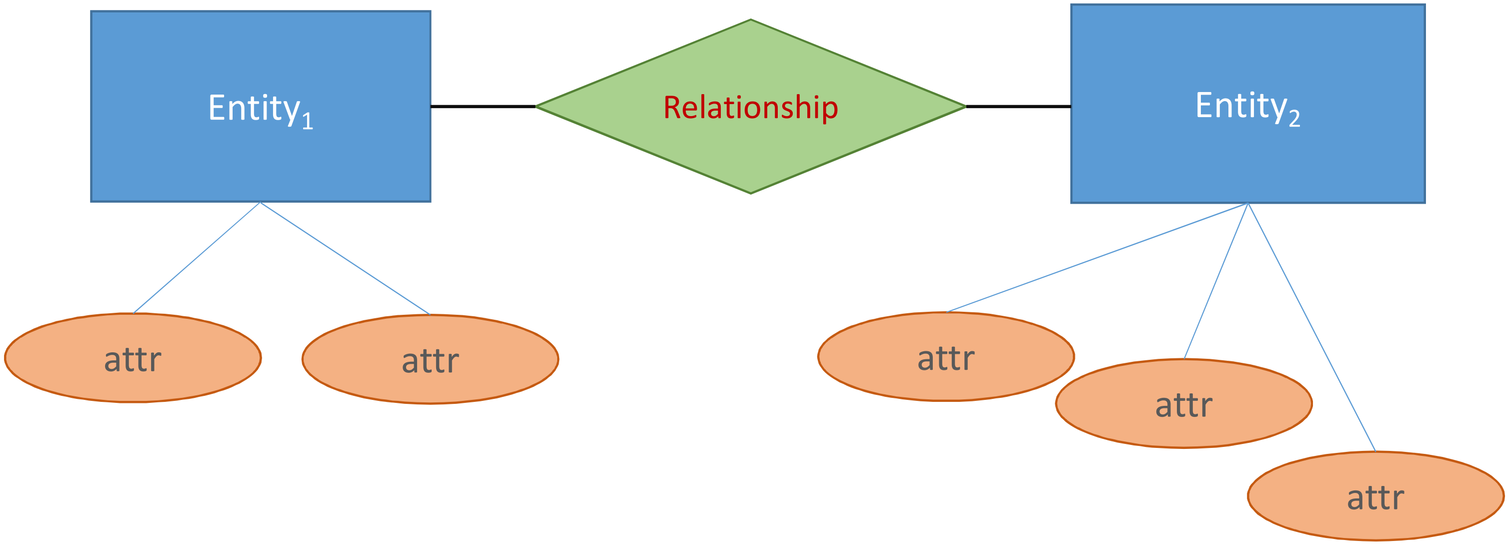
Recap: The Entity-Relationship Model

… provides a framework for thinking about data in terms of **entities** and their **relationships**.



Today’s menu

- How to convert ER Model into relational model (to SQL specifically)
- Our first programming with SQL

ICCS240   Database Management

DB Design/ER

(cont.)

Translation: Principles

- Maps
  - ER Schemas to relational schemas
  - ER instances to relational instances

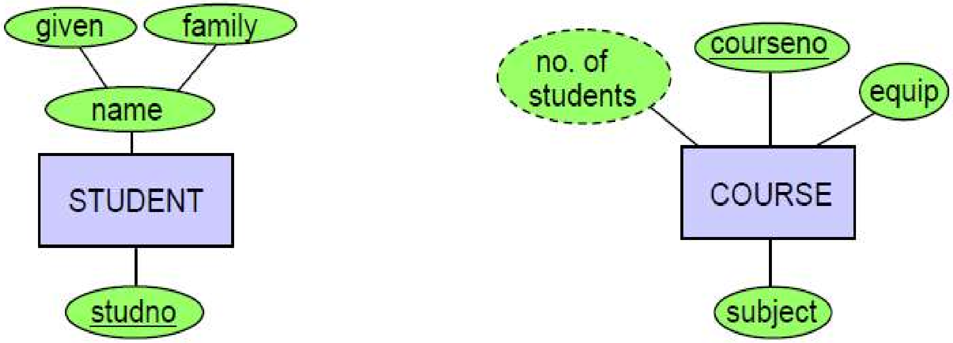
- Ideally, the mapping should
  - Be 1-to-1 in both directions
  - Not lose any information

- Difficulties:
  - What to do with ER-instances that have identical attribute values, but consist of different entities
  - In which way do we want to preserve information?

source: Werner Nutt

Entity to Relation

- For every entity type create a **relation**
- Every *atomic attribute* of the entity type becomes a *relation attribute*
- *Composite attributes*: include *all the atomic attributes*
- *Derived attributes* are not included (but remember their *derivation rules*)
- Relation instances are subsets of the cross product of the domains of the attributes
- Attributes of the *entity key* make up the ***primary key*** of the relation

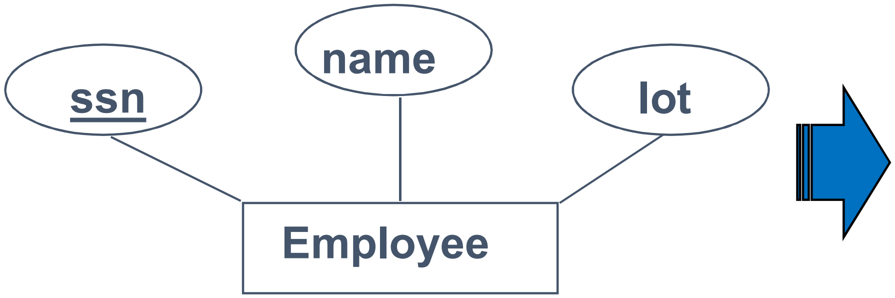


STUDENT (studno, givenname, familyname)

COURSE (courseno, subject, equip)

Adapted from source: Werner Nutt

Entity to Relation (more example)



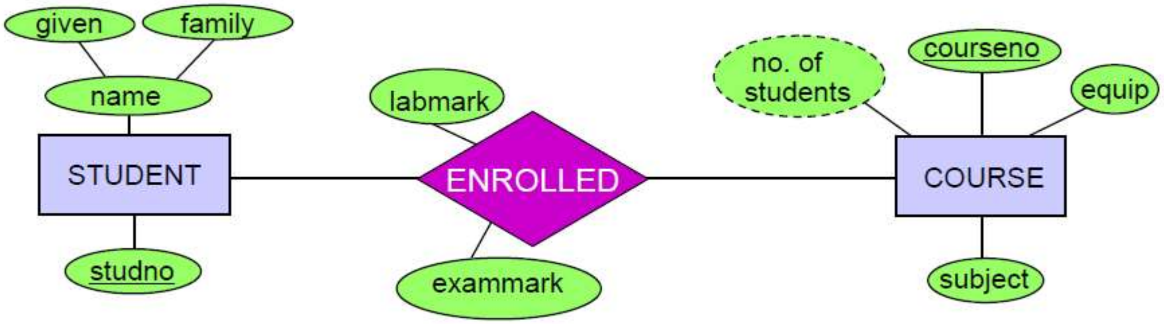
```
CREATE TABLE Employee
 (ssn CHAR(11),
  name CHAR(20),
  lot  INTEGER,
   PRIMARY KEY  (ssn))
```

Relationships to Relations

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

- Relationship sets
  - 1-to-1, 1-to-many, and many-to-many
  - Key/Total/Partial participation

Many-Many Relationships to Relations (example)



```
CREATE TABLE ENROLLED (
 studno INT,
 courseno INT,
 labmark FLOAT,
 exammark FLOAT,
 PRIMARY KEY (studno,courseno),
 FOREIGN KEY(studno)
       REFERENCES STUDENT,
 FOREIGN KEY (courseno)
       REFERENCES COURSE
);
```

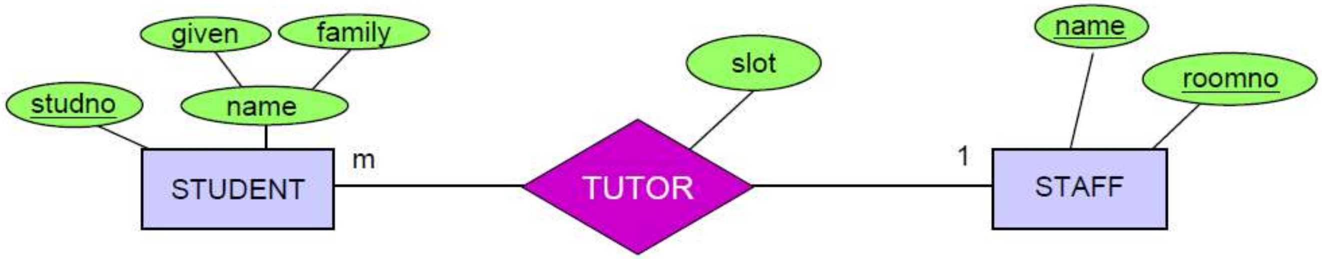
ENROL(studno, courseno, labmark, exammark)

Foreign Key ENROL(studno) references STUDENT(studno)

Foreign Key ENROL(courseno) references COURSE(courseno)

Adapted from source: Werner Nutt

Many-One Relationships to Relations (example)



The relation

STUDENT(studno, givenname, familyname)

is extended to

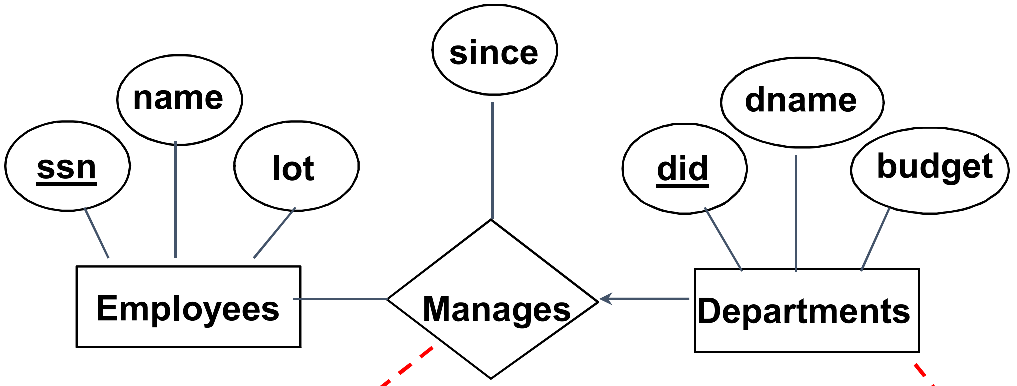
STUDENT(studno, givenname, familyname, tutor, roomno, slot)

and the constraint

Foreign Key STUDENT(tutor,roomno) references STAFF(name,roomno)

Adapted from source: Werner Nutt

Many-One Relationships to Relations
- Approach #1: separate tables

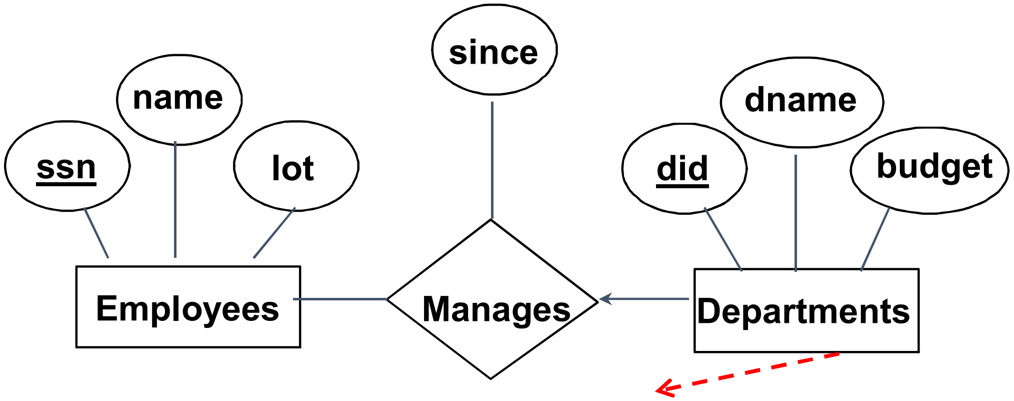


```
CREATE TABLE  Manages(
ssn     CHAR(11),
did     INT,
since   DATE,

PRIMARY KEY  (did),
FOREIGN KEY (ssn) REFERENCES Employees,
FOREIGN KEY (did) REFERENCES Departments)
```

```
CREATE TABLE  Departments(
 did      INT,
 dname    CHAR(20),
 budget   REAL,
 PRIMARY KEY  (did),
)
```

Many-One Relationships to Relations
- Approach #2: combined tables



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

One-Table vs. Two-Table Approaches

- The one-table approach:
  - (+) Eliminates the need for a separate table for the involved relationship set (e.g., Manages)
  - (+) Queries can be answered without combining information from two relations
  - (-) Space could be wasted!
    - What if several departments have no managers?

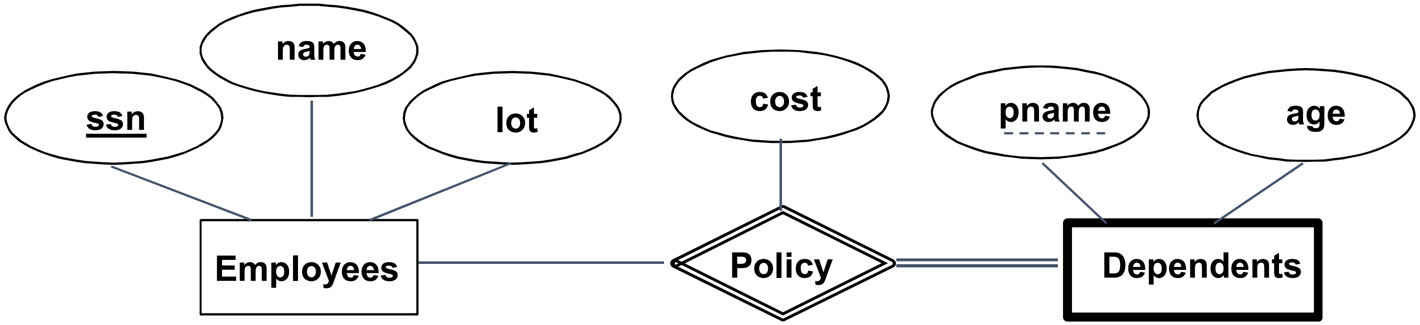
- The two-table approach:
  - The opposite of the one-table approach

For 1-to-1 relationship, can all be combined into one table.

Translating Weak Entities

A weak entity set always:

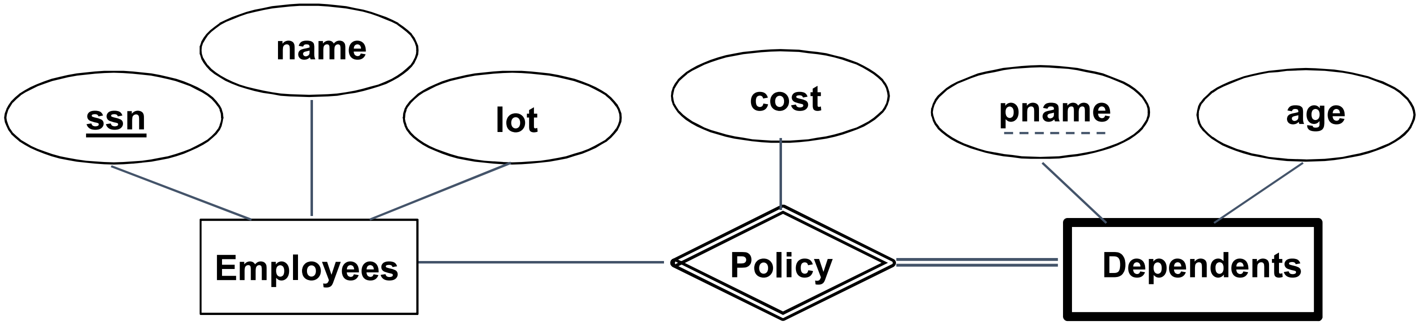
- Participates in a one-to-many binary relationship
- Has a key constraint and total participation



Which approach is ideal for that?

- The one-table approach

Example



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

A foreign key with **cascade delete** means that if a record in the parent table is **deleted**, then the corresponding records in the child table will automatically be **deleted**.