Relational Model

Relational Model

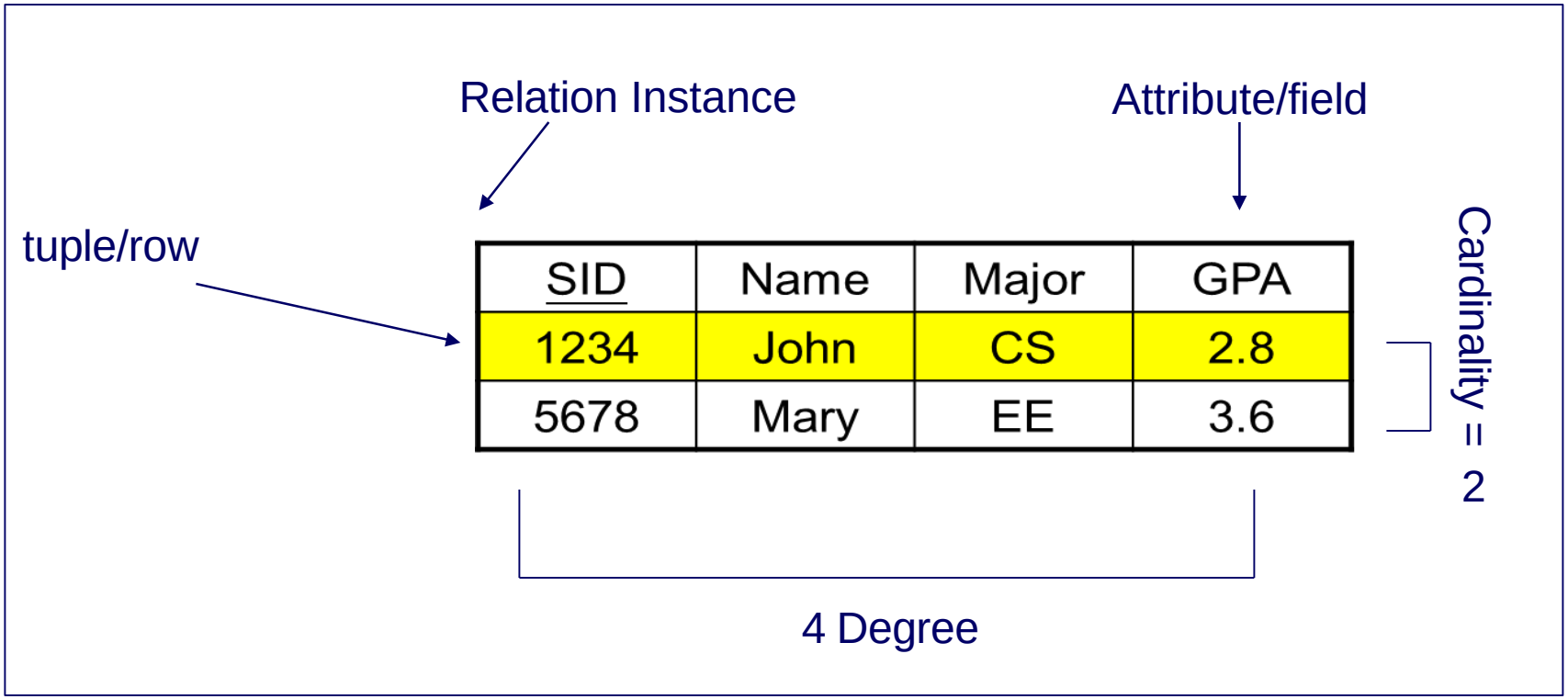
- To represent data in the relational mode -> Relation
- A relation consists of
  - Relation Schema
    - Relation's name, Name of each field/column/attribute, domain of each field

      Student(SID: Integer, Name: String, Major: String, GPA: Number)
  - Relation Instance
    - Set of tuples which has the same number of fields as the schema

SID	Name	Major	GPA
1234	John	CS	2.8
5678	Mary	EE	3.6

Relational Model: Requirements

- Rows be unique (key constraint)
- Order of rows does not matter
- Degree of a relation is the number of fields
- Cardinality of a relation is the number of tuples



Integrity Constraints

- Condition specified on a database schema and restrict the data that can be stored in a DB instance.
- Integrity Constraints must be true for any DB instance
- DBMS enforces the integrity constraints.
- Integrity Constraints
  - Domain Constraint
  - Primary Key Constraint
  - Foreign Key Constraint

Primary Key Constraints

- A set of fields is a *key* for a relation if :
  1. No two distinct tuples can have same values in all key fields

     (SID, Name) -> (1234, John) and (1234, Jack)

     ~~(1234, John) and (1234, John)~~
  2. Primary Key cannot be NULL <- Entity Integrity Constraint

Foreign Keys

- *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'.
- Prevent actions that would destroy links between tables
- Prevent invalid data from being inserted into Foreign Key column.
  - It has to be one of the values contained in the table it points to.
- Foreign Key can be NULL.

Foreign Keys

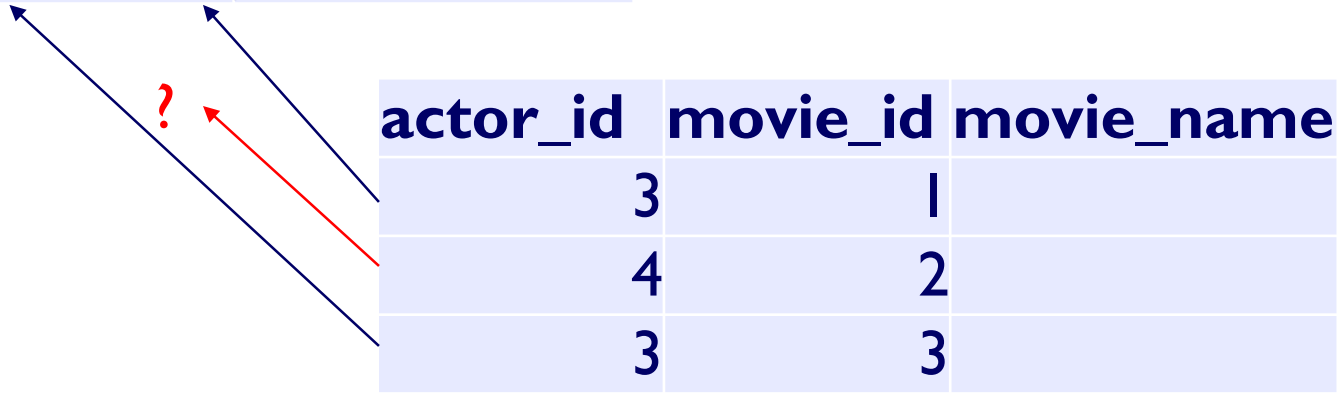
- An arrow originates from each foreign key and points to the related primary key in the associated relation
- Relations R1 and R2:
  - Attributes in foreign key have the same domain as the primary key of R2
  - Foreign Key is equal to primary key in some rows of the primary table, or else have no value (NULL).

Referential Integrity

- If all foreign key constraints are enforced, *referential integrity* is achieved, i.e., no dangling references.
- A tuple in one relation that refers to another relation must refer to an existing tuple in that relation.
- A referential integrity constraint requires that for each row of a table, the value in the foreign key matches a value in the parent key.

Enforcing Referential Integrity

actor_id	actor_name
1	Angelina
2	Brad
3	Jennifer



actor_id	movie_id	movie_name
3	1	
4	2	
3	3	

- Referential integrity NOT enforced.

Enforcing Referential Integrity

- 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.
- 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*.
  - (In SQL, also: Set sid in Enrolled tuples that refer to it to a special value *null*, denoting *`unknown'* or *`inapplicable'*.)

Creating relational tables

- Tables are created using CREATE TABLE statements in SQL
- Each attribute is defined, taking the data type and length from the domain definitions
- For example, the attribute Customer_Name can be defined as a VARCHAR (variable character) type with length 25
- By specifying NOT NULL, each attribute can be constrained from being assigned a null value
- The primary key for each table is specified using the PRIMARY KEY clause at the end of each table definition

Creating relational tables

```
CREATE TABLE CUSTOMER
    (CUSTOMER_ID     VARCHAR(5)     NOT     NULL
    CUSTOMER_NAME   VARCHAR(25)   NOT   NULL
    PRIMARY KEY (CUSTOMER_ID);
```

Creating relational tables

```
CREATE TABLE ORDER
    (ORDER_ID       CHAR(5)              NOT NULL
    ORDER_DATE    DATE                   NOT NULL
    C_ID VARCHAR(5)                      NOT NULL
PRIMARY KEY (ORDER_ID)
FOREIGN KEY (CUSTOMER_ID) REFERENCES
    CUSTOMER(CUSTOMER_ID);
```

Creating relational tables

- In SQL, a FOREIGN KEY REFERENCES statement corresponds to one of these arrows
- The foreign key CUSTOMER_ID references the primary key of CUSTOMER, which is also CUSTOMER_ID
- Foreign Key constraint has DELETE and UPDATE operations
  - Actions:
    - Cascade, No Action, Set NULL, Set Default

Creating relational tables

- ON UPDATE/ON DELETE Actions:
  - CASCADE: Delete or update the row from the parent table, and automatically delete or update the matching rows in the child table.
  - NO ACTION: Prevent the deletion or update of a parent key if there is a row in the child table that references the key (default)
  - SET NULL: Delete or update the row from the parent table, and set the foreign key column or columns in the child table to NULL.
  - SET DEFAULT: The "SET DEFAULT" actions are similar to "SET NULL", except that each of the child key columns is set to contain the columns default value instead of NULL.

Creating relational tables

The ORDER_LINE table illustrates how to specify a primary key when that key is a composite attribute of two foreign keys:

```
CREATE TABLE ORDER_LINE
    (ORDER_ID       CHAR(5)          NOT  NULL
    PRODUCT_ID      CHAR(5)          NOT  NULL
    QUANTITY        INT              NOT NULL
PRIMARY KEY(ORDER_ID, PRODUCT_ID)
FOREIGN KEY (ORDER_ID) REFERENCES ORDER(ORDER_ID)
FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT(PRODUCT_ID);
```

Constraints in Create Table

- Adding constraints to a table enables the database system to enforce data integrity.
- Different types of constraints:

* Not Null	* Default Values
* Unique	* Primary Key
* Foreign Key	* Check Condition

An Example

```
CREATE TABLE Student (
        ID          NUMBER,
        Fname       VARCHAR2(20),
        Lname       VARCHAR2(20),
);
```

Not Null Constraint

```
CREATE TABLE Student (
        ID          NUMBER,
        Fname       VARCHAR2(20) NOT NULL,
        Lname       VARCHAR2(20) NOT NULL,
);
```

Primary Key Constraint

```
CREATE TABLE Student (
        ID          NUMBER    PRIMARY KEY,
        Fname     VARCHAR2(20) NOT NULL,
        Lname     VARCHAR2(20) NOT NULL,
);
```

Primary Key implies: * NOT NULL * UNIQUE.
There can only be one primary key.

Primary Key Constraint (Syntax 2)

```
CREATE TABLE Students (
        ID          NUMBER,
        Fname       VARCHAR2(20) NOT NULL,
        Lname       VARCHAR2(20) NOT NULL,
        PRIMARY KEY(ID)
);
```

Needed when the primary key is made up of one or more fields

Logical DB Design: ER to Relational

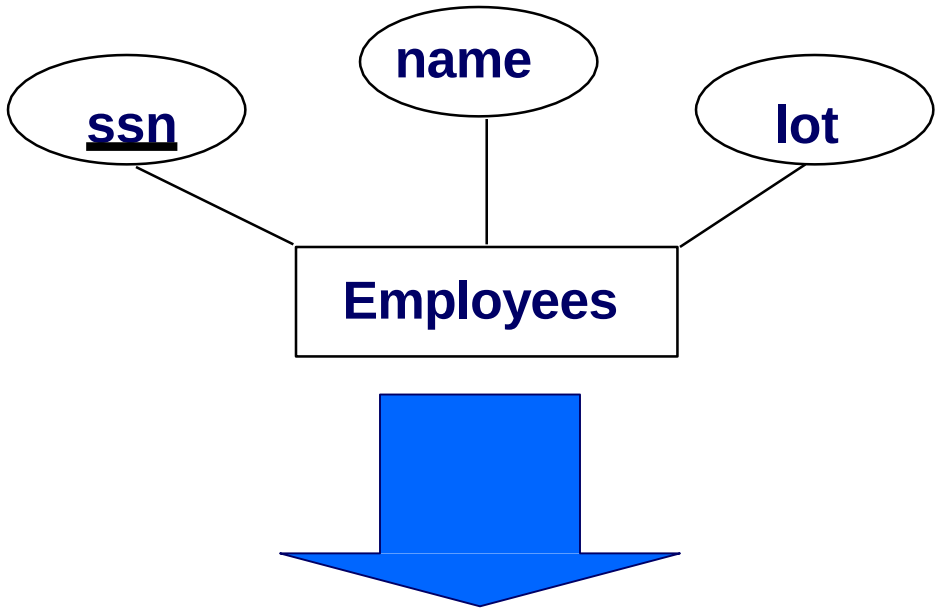
So… how do we convert an ER diagram into a table??

Basic Ideas:

- Build a table for each entity set
- Build a table for each relationship set (if necessary)
- Make a column in the table for each attribute in the entity set
- Primary Key

Logical DB Design: ER to Relational

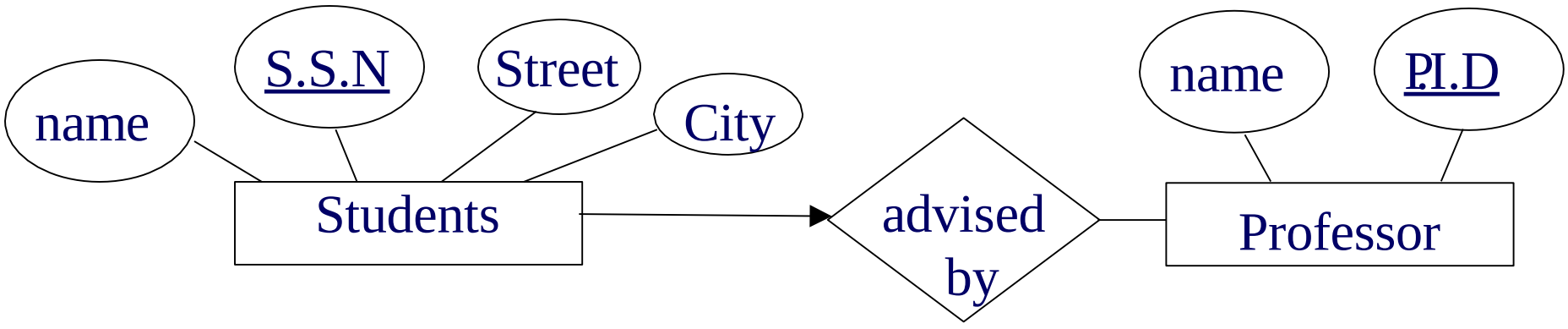
- **Entity sets to tables:**




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

ssn	name	lot
123-22-3666	Smith	48
231-31-5368	Lee	22
131-24-3650	Brown	35

Strong Entity Sets



SSN	Name	Street	City
123-22-3666	Smith	Main St	Santa Clara
231-31-5368	Lee	1st St	San Jose
131-24-3650	Brown	2nd St	San Jose

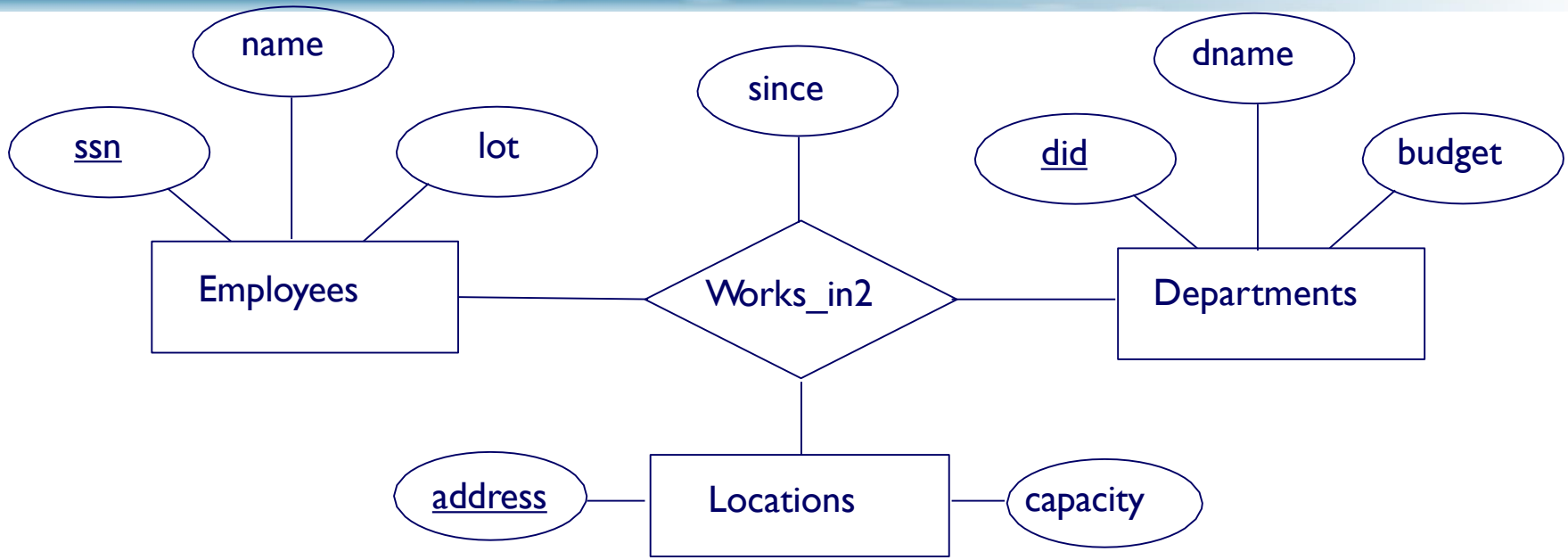
PID	Name
9999	Smith
4444	Lee
3333	Brown

Relationship Sets to Tables

- To represent a relationship, must identify:
  - Each participating entity
    - Keys for each participating entity set (as foreign keys).
    - This set of attributes forms a superkey for the relation.
  - All descriptive attributes of the relationship.

Relationship Sets to Tables

No Constraints



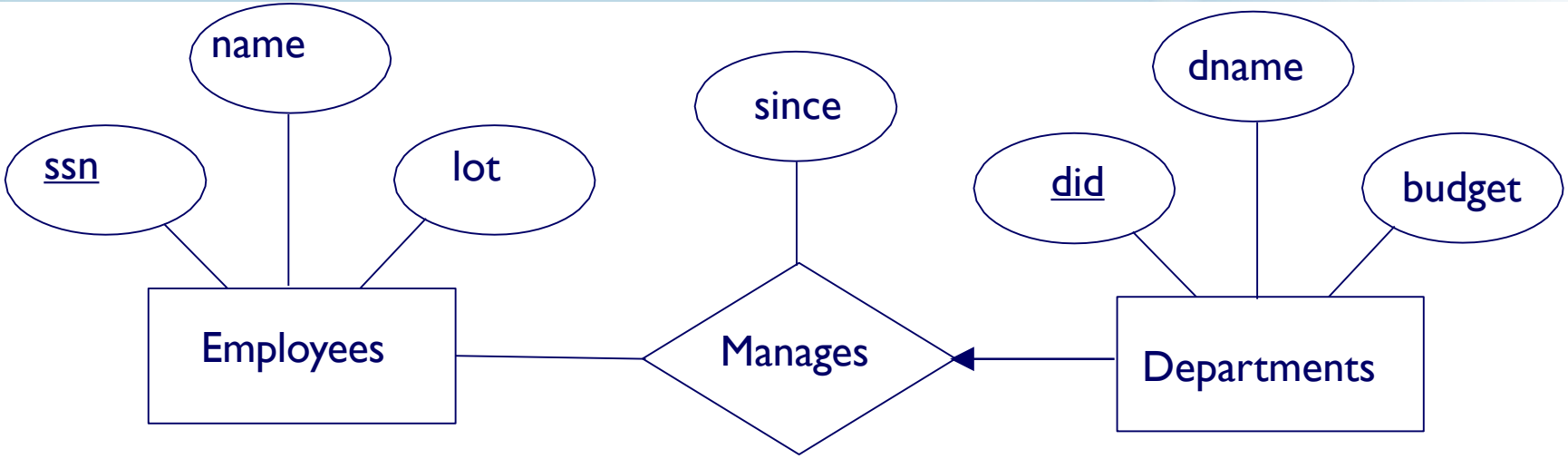
ssn	did	address	since

```
CREATETABLEWorks_In2(
  ssn  CHAR(11),
  did INTEGER,
  address CHAR(20),
  since  DATE,
  PRIMARY KEY (ssn, did,address),
  FOREIGN KEY (ssn)
       REFERENCES Employees,
  FOREIGN KEY (address)
       REFERENCES Locations,
  FOREIGN KEY (did)
       REFERENCES Departments);
```

- ssn is a foreign key referencing employees
- did is a foreign key referencing departments
- address is a foreign key referencing locations

Relationship Sets to Tables

With Key Constraints



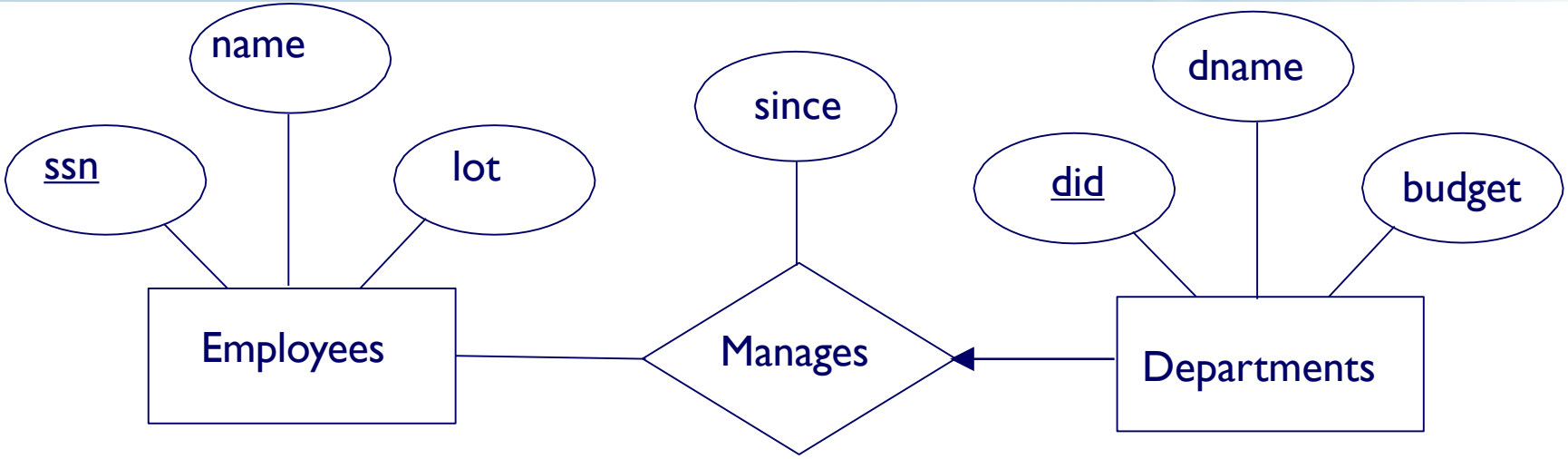
- Method 1
  - Create a separate relation for Manages
  - did is the key
  - Separate relations for Employees and Departments.

ssn	did	since

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

Relationship Sets to Tables

With Key Constraints



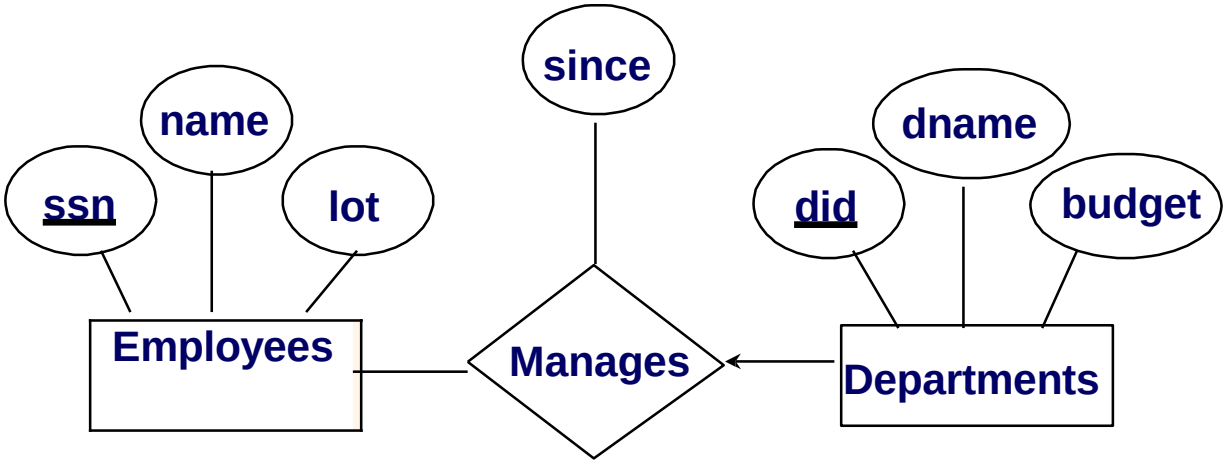
did	dname	budget	ssn	since

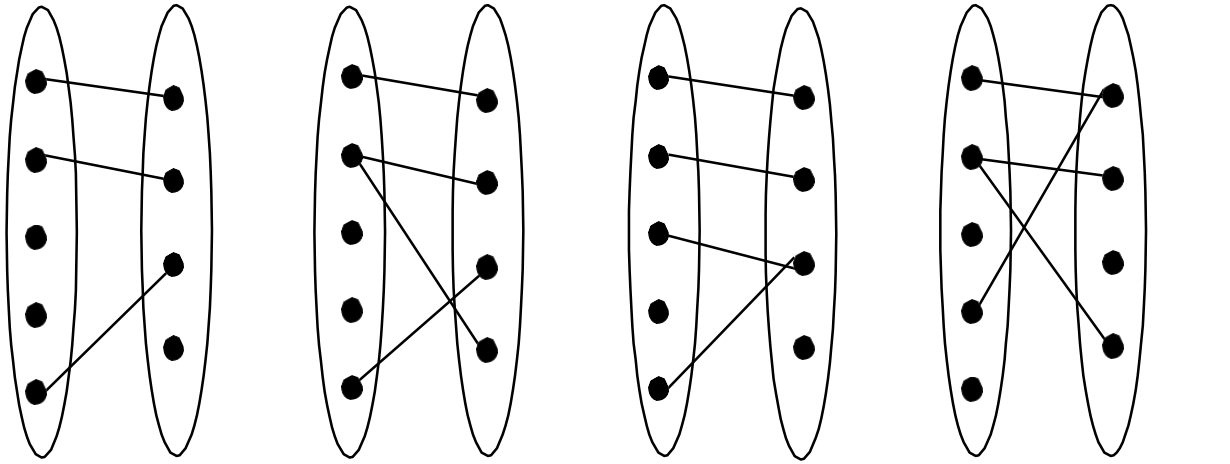
- Method 2
  - Each department has a unique manager, we could instead combine Manages and Departments.

```
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: Key Constraints

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



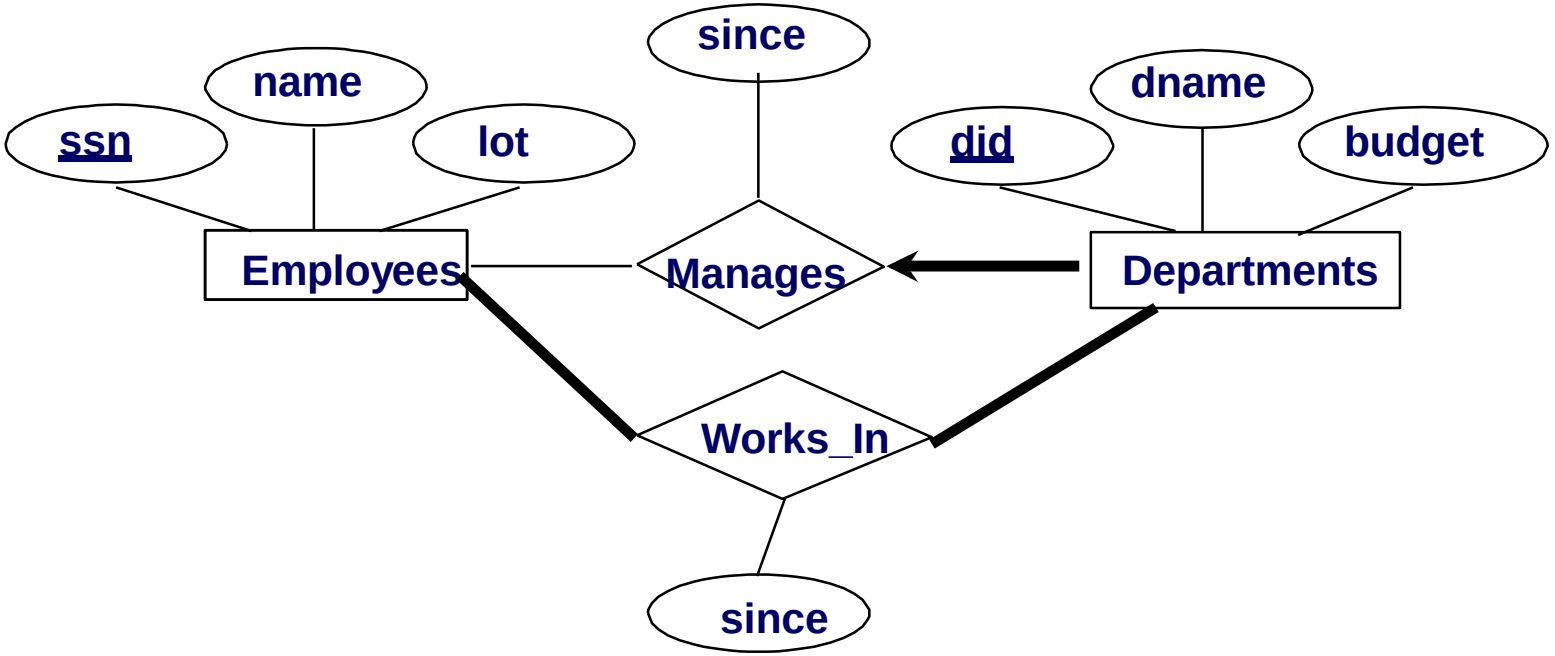


1-to-1 | 1-to Many | Many-to-1 | Many-to-Many

Translation to relational model?

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!)



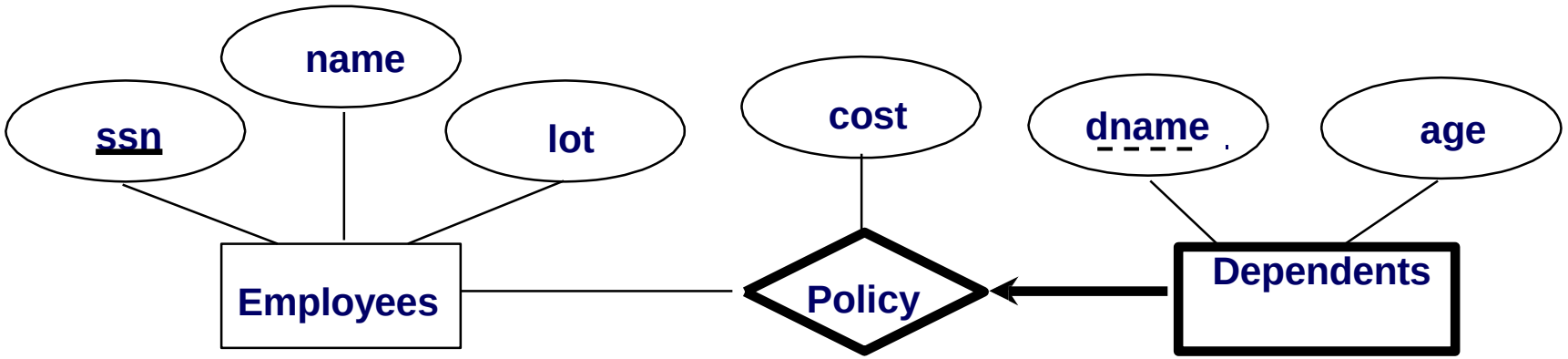
Participation Constraints in SQL

- 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.



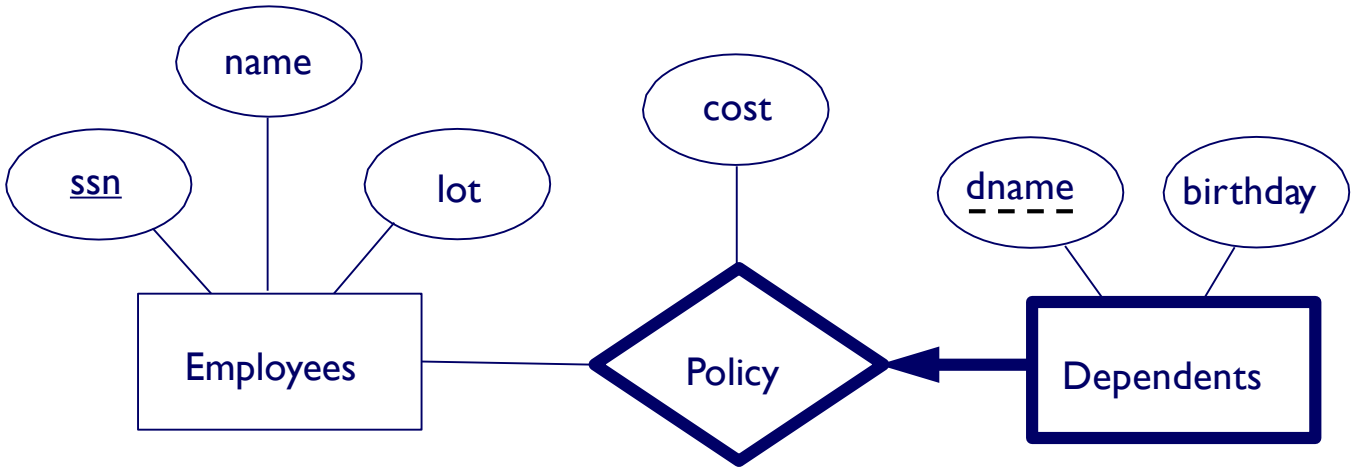
Translating Weak Entity Sets

- Weak Entity Set cannot exists alone
  - When the owner entity is deleted, all owned weak entities must also be deleted.
- To build a table/schema for weak entity set
  - Construct a table with one column for each attribute in the weak entity set
  - Remember to include partial key
  - Include primary key of the Strong Entity Set (the entity set that the weak entity set is depending on) as foreign key
  - Primary Key of the weak entity set = partial key+ foreign key

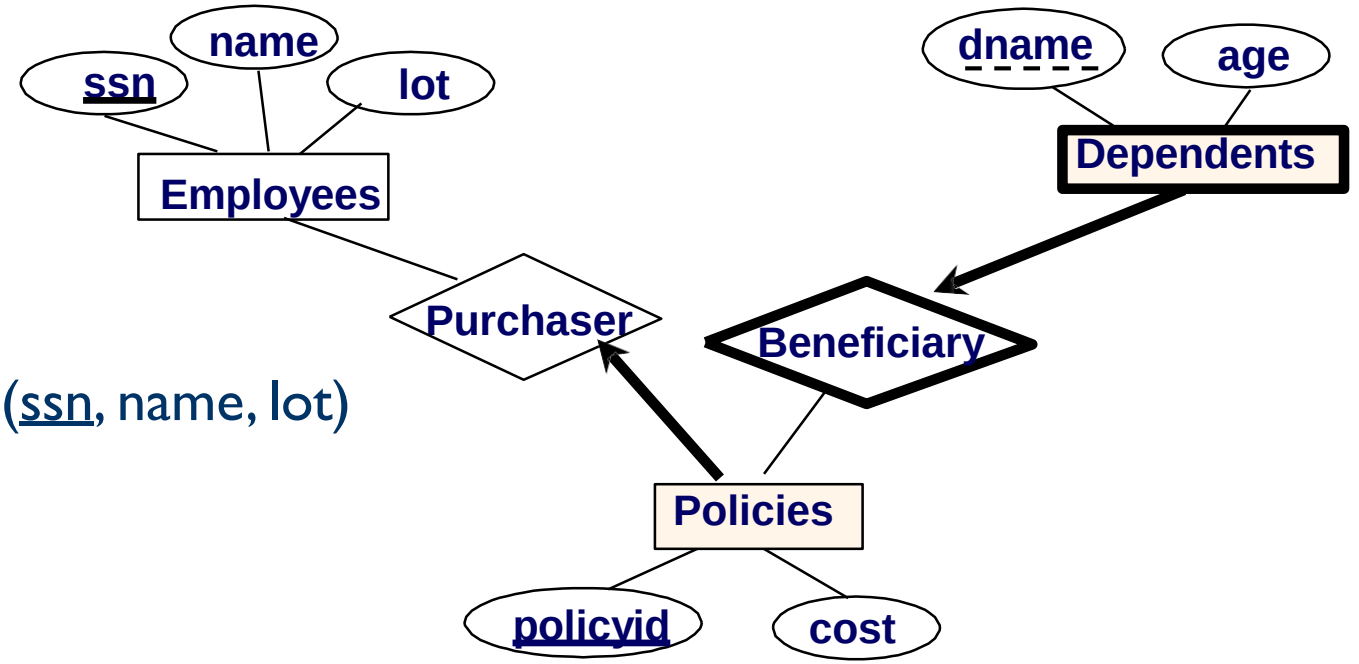
Translating Weak Entity Sets

- 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 (
dname CHAR(20),
birthday DATE,
cost  REAL,
ssn CHAR(11) NOT NULL,
PRIMARY KEY (dname, ssn),
FOREIGN KEY (ssn) REFERENCES Employees
ON DELETE CASCADE);
```



Review: Binary vs. Ternary



- Employee (ssn, name, lot)
- Dependents(dname, policyid age)
  - policyid is a foreign key referencing Policies
- Policies (policyid, cost, ssn)
  - ssn is a foreign key referencing Employees
  - ssn can not ne NULL

Review: Binary vs. Ternary

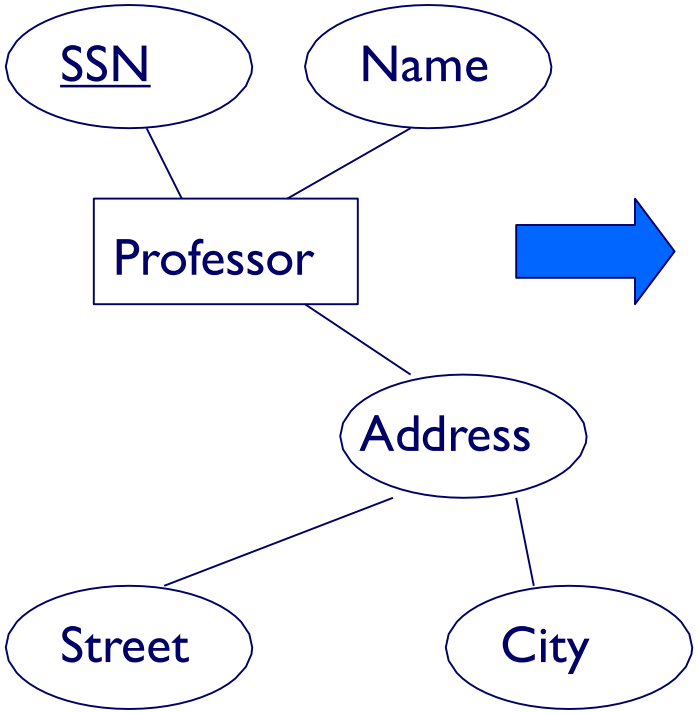
- The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.
- Participation constraints lead to NOT NULL constraints.

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

Representing Composite Attribute

- One column for each component attribute
- NO column for the composite attribute itself

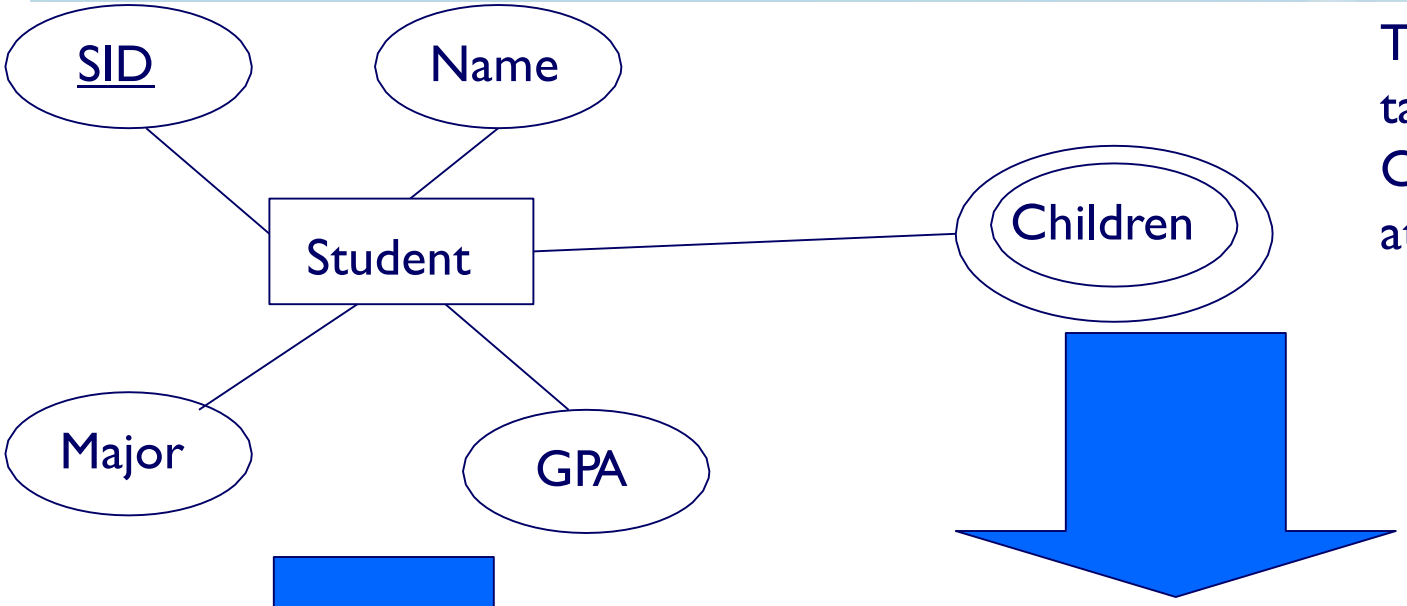


SSN	Name	Street	City
9999	Dr. Smith	50 1st St.	Cupertino
8888	Dr. Lee	1 B St.	San Jose

Representing Multivalue Attribute

- For each multivalue attribute in an entity set/relationship set
  - Build a new relation schema with two columns
  - One column for the primary keys of the entity set/relationship set that has the multivalue attribute
  - Another column for the multivalue attributes. Each cell of this column holds only one value. So each value is represented as an unique tuple
  - Primary key for this schema is the union of all attributes

Example – Multivalue attribute



The primary key for this table is Student_SID + Children, the union of all attributes

SID	Name	Major	GPA
1234	John	CS	2.8
5678	Homer	EE	3.6

Stud_SID	Children
1234	Johnson
1234	Mary
5678	Bart
5678	Lisa
5678	Maggie