INFO20003 Database Systems

Dr Renata Borovica-Gajic

Lecture 04

Relational Model & SQL

Translating ER diagrams

Coverage

- Relational Model & SQL overview
- Keys & Integrity Constraints
- Translating ER to Logical and Physical Model

Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems

Databases Model the Real World

- **Data Model** allows us to translate real world things into structures that a computer can store
- Many models: Relational, ER, O-O, Network, Hierarchical, etc.

- **Relational Model**
  - Rows & Columns
  - Keys & Foreign Keys to link Relations

Enrolled

sid	cid	grade
53666	Carnatic101	5
53666	Reggae203	5.5
53650	Topology112	6
53666	History105	5

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	5.4
53688	Smith	smith@eecs	18	4.2
53650	Smith	smith@math	19	4.8

Relational Database: Definitions

- *Relational database:* a set of *relations*.
- *Relation:* made up of 2 parts:
  - *Schema* : specifies name of relation, plus name and type of each column.
    - E.g. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)
  - *Instance* : a *table*, with rows and columns.
    - #rows = *cardinality*
    - #fields = *degree / arity*
- Can think of a relation as a *set* of rows or *tuples*.
  - i.e., all rows are distinct, no order among rows

Example Instance of Students Relation

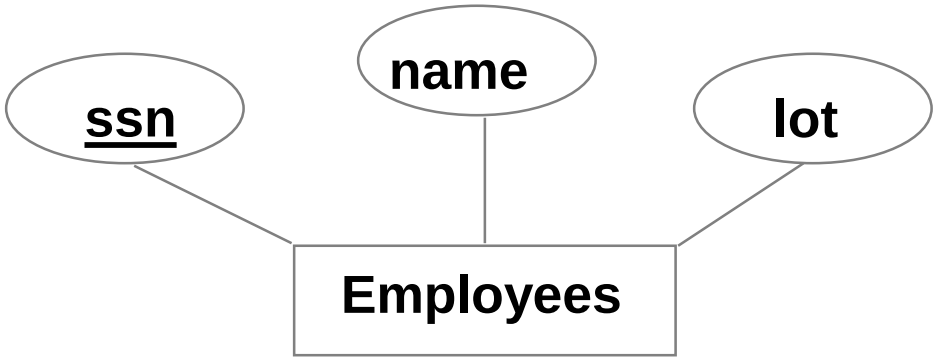
Students

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

Cardinality = 3, degree (arity) = 5, all rows distinct

Logical Design: ER to Relational Model

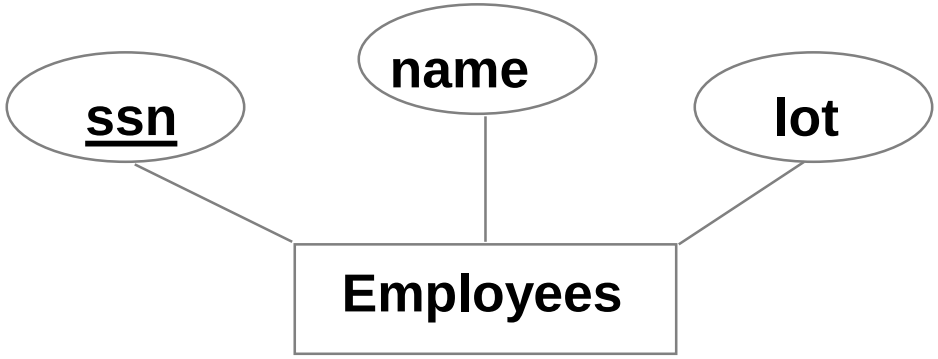
Entity set to **relation**



Logical Design:

Employees = (<u>ssn</u>,
name,
lot)

ER to Logical to Physical Design



EMPLOYEES

ssn	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Logical Design:

Employees = (ssn,
name,
lot)

Physical Design:

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

Structured Query Language (SQL)

SQL - A language for Relational DBMS

- SQL (a.k.a. “Sequel”), standard language
- Data Definition Language (DDL)
  - create, modify, delete relations
  - specify constraints
- Data Manipulation Language (DML)
  - Specify *queries* to find tuples that satisfy criteria
  - add, modify, remove tuples
- Data Control Language (DCL)
  - control access to the database
  - administer users, security, etc.

SQL Overview

- CREATE TABLE <name> (<field> <domain>, …)
- INSERT INTO <name> (<field names>)
  VALUES (<field values>)
- DELETE FROM <name>
  WHERE <condition>
- UPDATE <name>
  SET <field name> = <value>
  WHERE <condition>
- SELECT <fields>
  FROM <name>
  WHERE <condition>

Creating Relations in SQL

- Creates the Students relation.
  - Note: the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

```
CREATE TABLE Students
       (sid CHAR(20),
        name CHAR(20),
        login CHAR(10),
        age INTEGER,
        gpa FLOAT)
```

Table Creation (continued)

- Another example:
  - the Enrolled table holds information about courses students take.

```
CREATE TABLE Enrolled
    (sid CHAR(20),
     cid CHAR(20),
     grade CHAR(2))
```

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
 VALUES  (‘53688’, ‘Smith’, ‘smith@cs’, 18, 3.2)
```

- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE
  FROM Students
 WHERE name = ‘Smith’
```

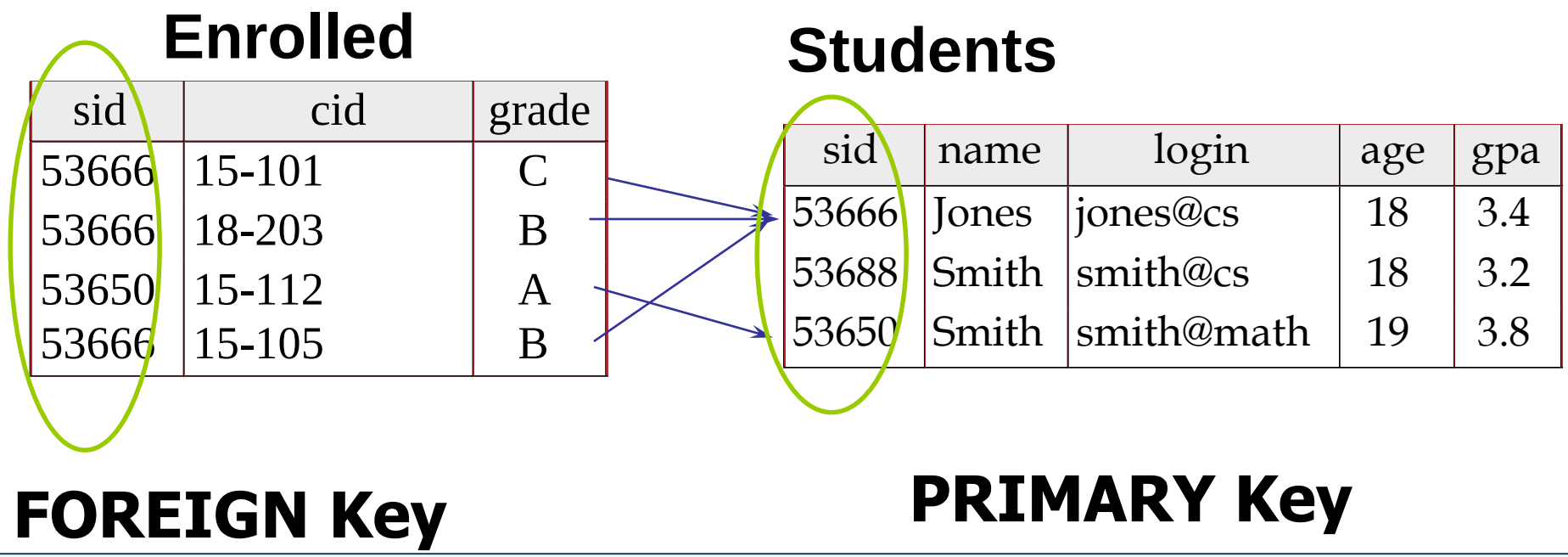
Relational Model

- Relational Model & SQL overview
- **Keys & Integrity Constraints**
- Translating ER to Logical and Physical Model

Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems

Keys

- Keys are a way to associate tuples in different relations
- Keys are one form of integrity constraint (IC)



Primary Keys

- A set of fields is a *superkey* if:
  - No two distinct tuples can have same values in all key fields
- A set of fields is a *key* for a relation if :
  - It is a superkey
  - No subset of the fields is a superkey
- what if >1 key for a relation?
  - one of the keys is chosen (by DBA) to be the ***primary key***. Other keys are called ***candidate*** keys.
- E.g.
  - *sid* is a key for Students.
  - What about *name*?
  - The set {*sid, gpa*} is a superkey.

Primary and Candidate Keys in SQL

- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.
- Keys must be used carefully!
- "For a given student and course, there is a single grade."

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

VS.

```
CREATE TABLE Enrolled
  (sid CHAR(20)
   cid  CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid),
   UNIQUE (cid, grade))
```

"Students can take only one course, and no two students in a course receive the same grade."

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 the primary key of the other relation.
  - Like a `logical pointer'.

- If all foreign key constraints are enforced, *referential integrity* is achieved (i.e., no dangling references.)

Foreign Keys in SQL

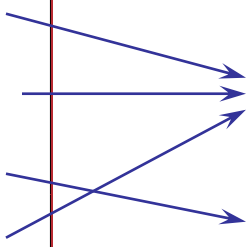
Example: Only students listed in the Students relation should be allowed to enroll for courses.

- *sid* is a foreign key referring to Students:

```
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	15-101	C
53666	18-203	B
53650	15-112	A
53666	15-105	B



Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@cs	18	3.2
53650	Smith	smith@math	19	3.8

Enforcing Referential Integrity

- Consider Students and Enrolled; *sid* in Enrolled is a foreign key that references Students.
- 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*'.)
- Similar issues arise if primary key of Students tuple is updated.

Integrity Constraints (ICs)

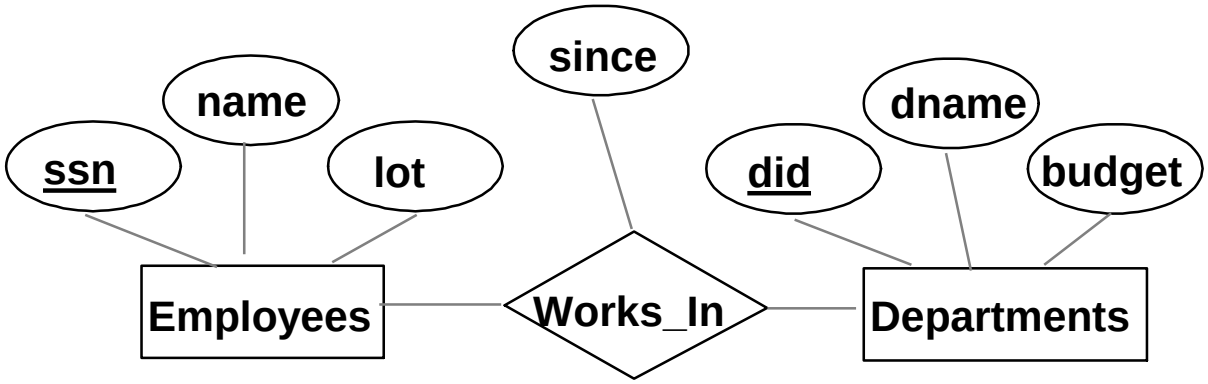
- IC: condition that must be true for *any* instance of the database; e.g., *domain constraints.*
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.
- A *legal* instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!

Relational Model

- Relational Model & SQL overview
- Keys & Integrity Constraints
- **Translating ER to Logical and Physical Model**

Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems

ER to Logical Design



Logical Design:

Employees = (ssn, name, lot)

Departments = (did, dname, budget)

Works_In = (**ssn**, **did**, since)

Note: Underline = PK,
italic and underline = FK,
underline and bold = PFK

Logical to Physical Design

Logical Design:

Employees = (ssn, name, lot)

Works_In = (**ssn**, **did**,since)

Departments = (did, dname, budget)

Note: Underline = PK,
italic and underline = FK,
underline and bold = PFK

Physical Design:

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

```
CREATE TABLE Departments
        (did INTEGER,
         dname CHAR(20),
         budget FLOAT,
         PRIMARY KEY  (did))
```

```
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)
```

Example Instance

Employees

0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Departments

101	Sales	10K
105	Purchasing	20K
108	Databases	1000K

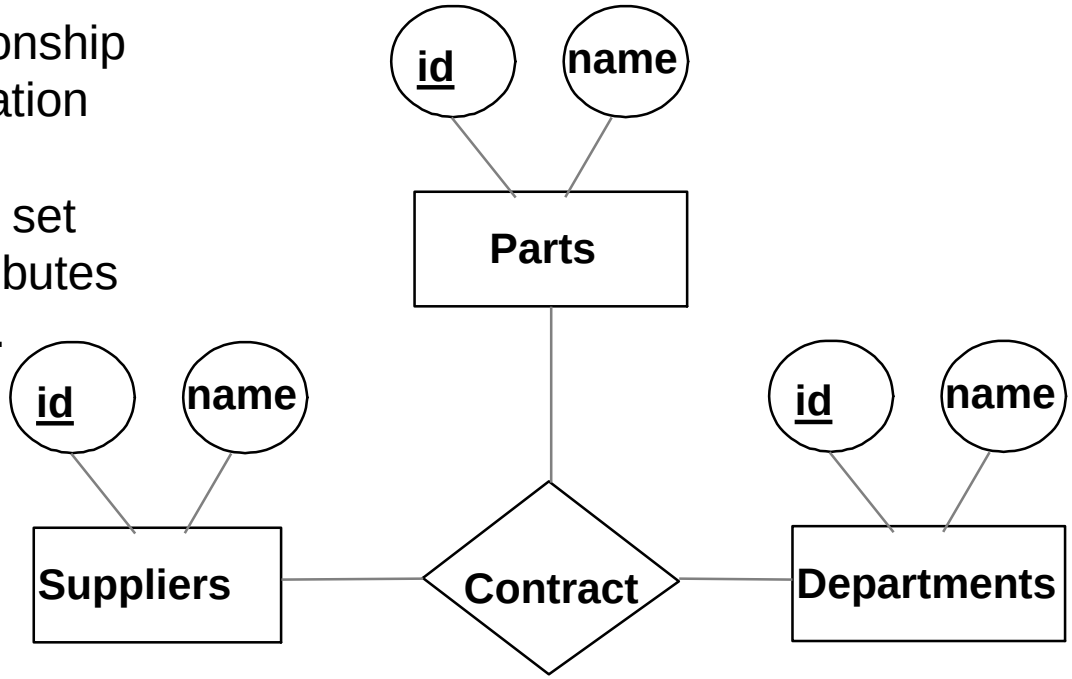
Works_In

0983763423	101	1 Jan 2003
0983763423	108	2 Jan 2003
9384392483	108	1 Jun 2002

ER to Logical Design Example 2

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

- Keys for each participating entity set (as foreign keys). This set of attributes forms a *superkey* for the relation.
- All descriptive attributes.



Logical Design:

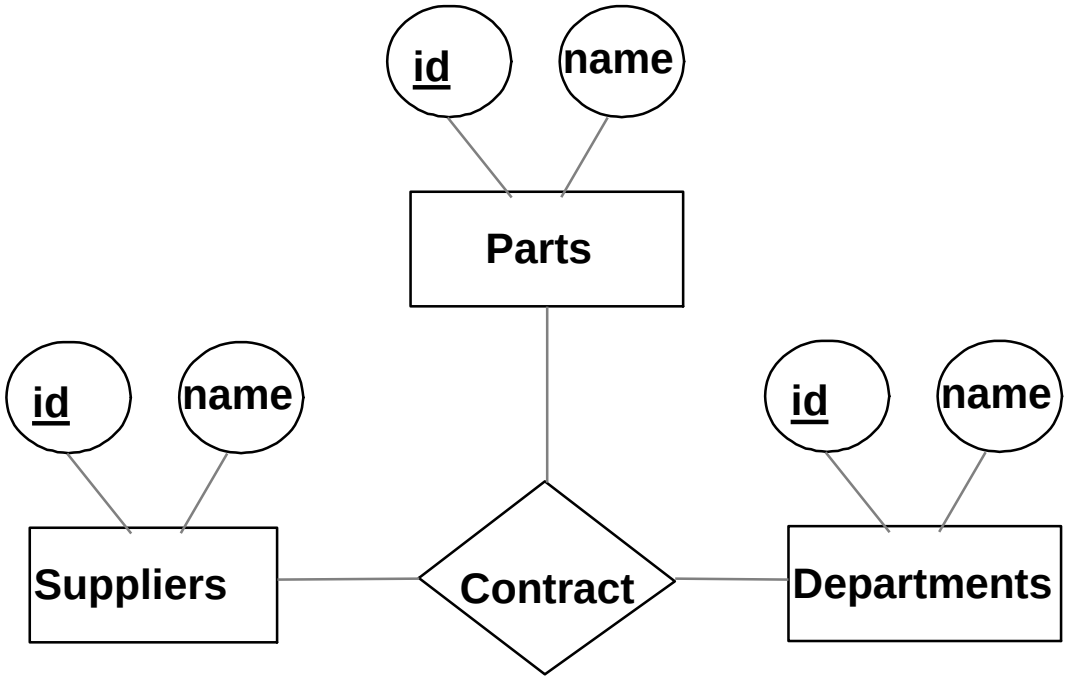
Contracts (
supplier_id,
part_id,
department_id)

Note: Underline = PK, italic and underline = FK, underline and bold = PFK

ER to Logical to Physical Design Example 2

Logical Design:

Contracts (
supplier_id,
part_id,
department_id)

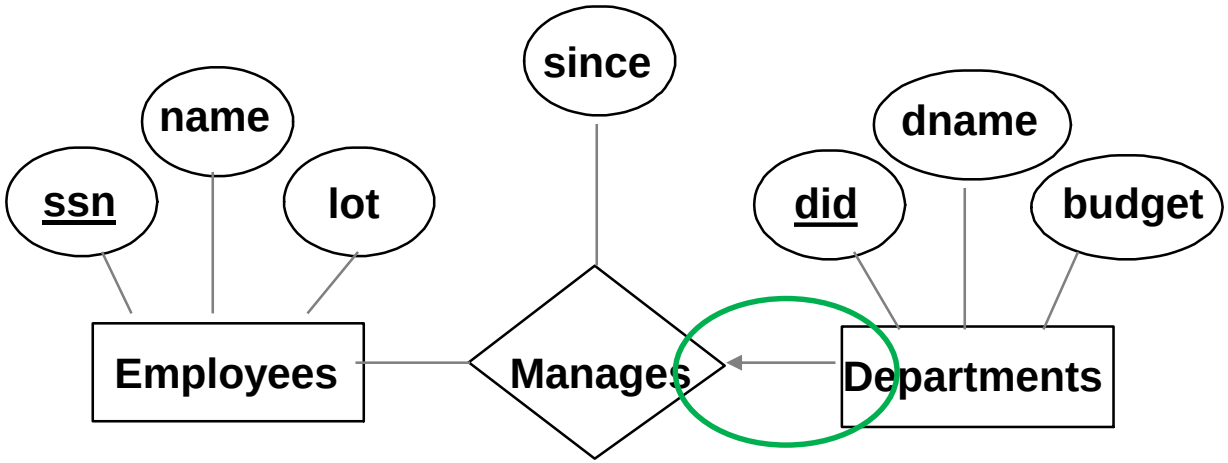


Physical Design:

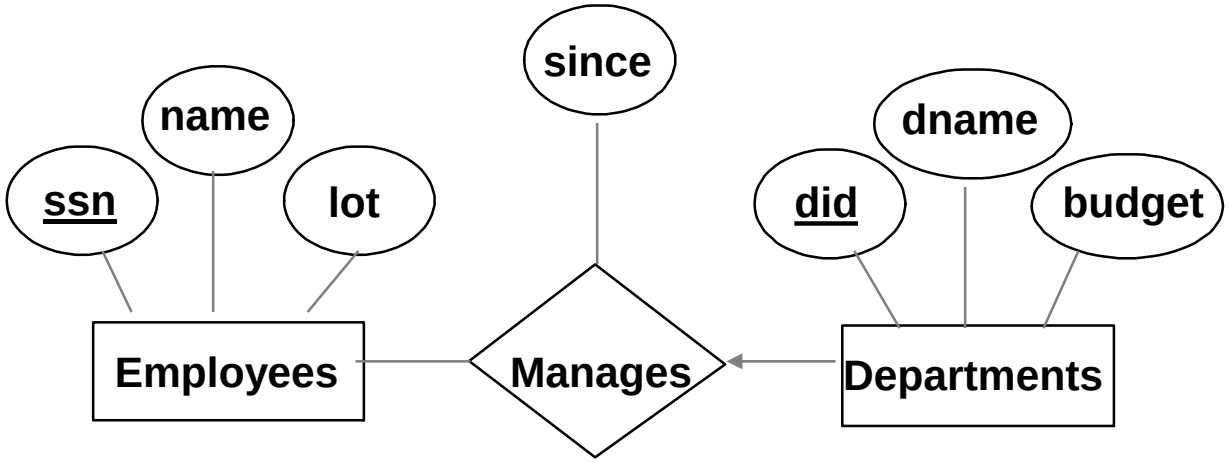
```
CREATE TABLE Contracts (
        supplier_id  INTEGER,
        part_id  INTEGER,
        department_id  INTEGER,
        PRIMARY KEY (supplier_id, part_id, department_id),
        FOREIGN KEY (supplier_id) REFERENCES Suppliers,
        FOREIGN KEY (part_id) REFERENCES Parts,
        FOREIGN KEY (department_id) REFERENCES Departments)
```

Review: Key Constraints in ER

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



Key Constraints: Logical design



Logical Design:

Employees = (ssn,name, lot)

Departments = (did,dname, budget)

Manages = (*ssn*, did, since)

VS.

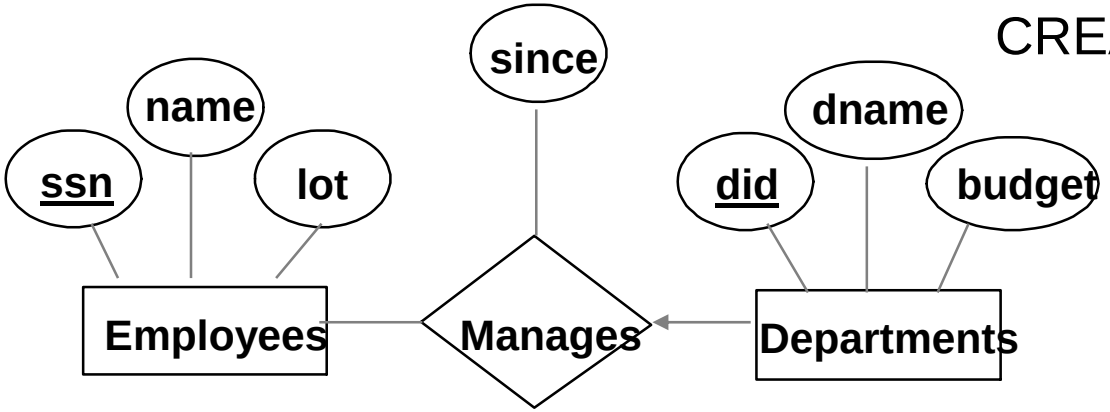
Employees = (ssn,name, lot)

Departments = (did,dname, budget, *ssn*, since)

Note: Underline = PK, italic and underline = FK, underline and bold = PFK

Key Constraints in SQL

Physical Design:




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

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

vs.

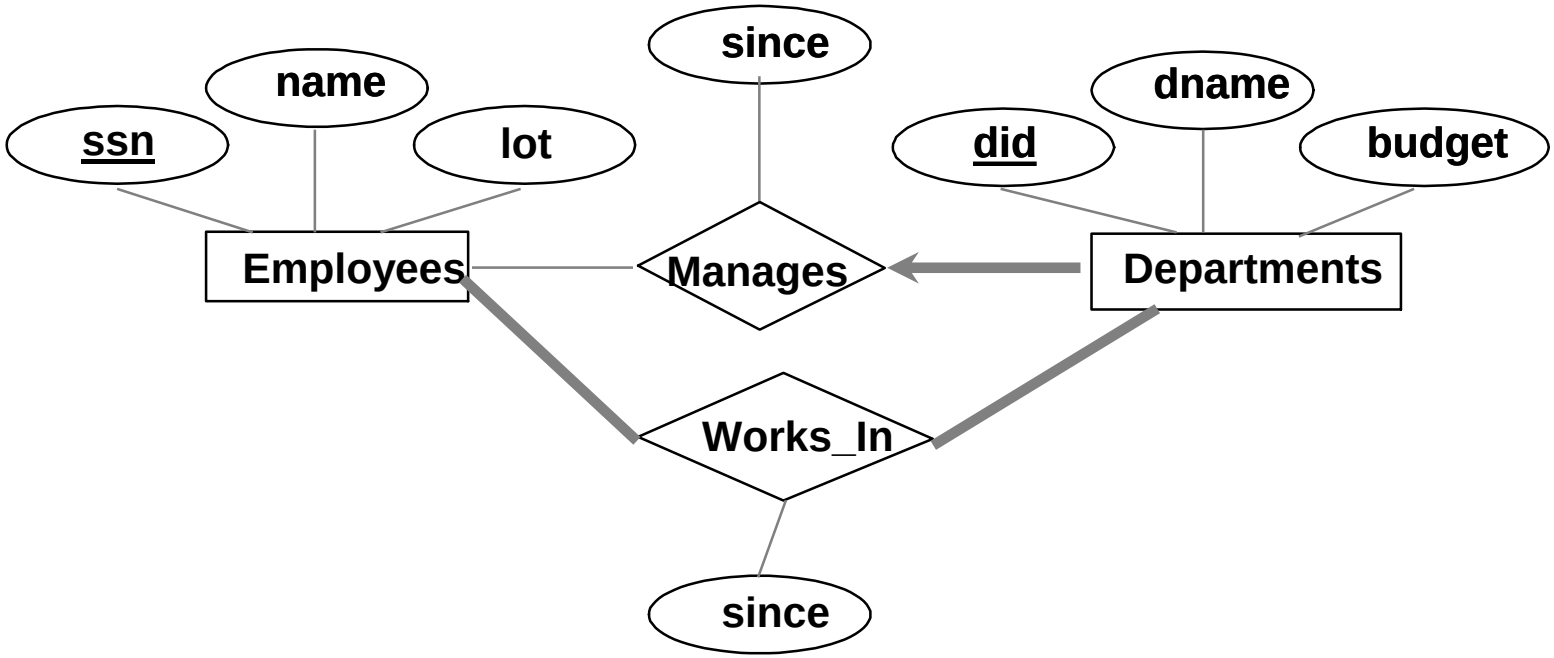
```
CREATE TABLE Departments
       (did INTEGER,
        dname CHAR(20),
        budget FLOAT,
        ssn CHAR(11),
        since DATE,
        PRIMARY KEY  (did)
        FOREIGN KEY (ssn)
        REFERENCES Employees)
```

Which one is better?

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!)



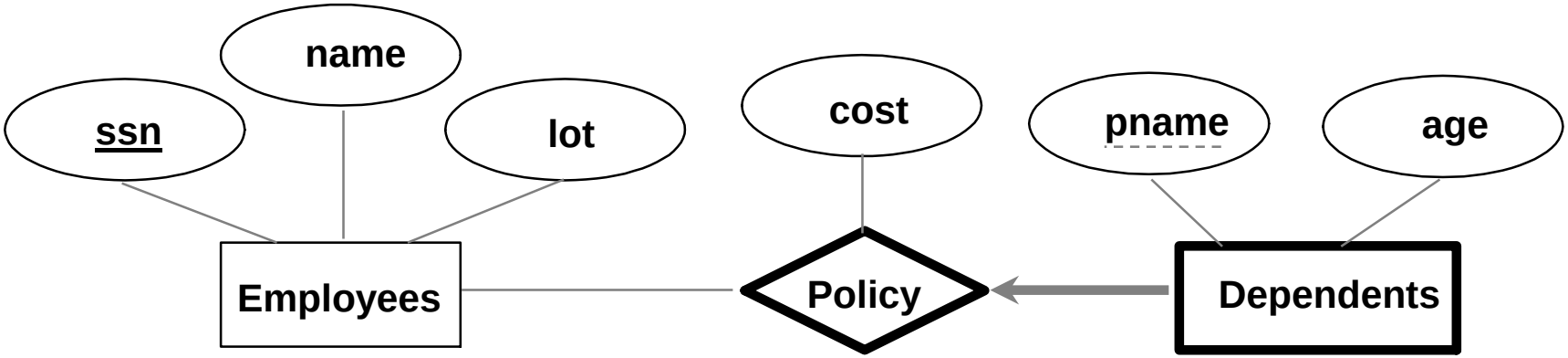
Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to CHECK constraints).

```
CREATE TABLE Departments(
    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.

Logical Design:

Dependents = (<u>pname</u>, age, cost, <u>**ssn**</u>)

Note: Underline = PK, italic and underline = FK, underline and bold = PFK

Physical Design:

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

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.
- Rules to translate ER to logical design (relational model)

What's examinable

- Translate conceptual (ER) into logical & physical design
- Understand integrity constraints
- Use DDL of SQL to create tables with constraints

Next Lecture

- ER Modelling Example with MySQL Workbench
  - You will need this for workshops/labs (and assessment)