Dr Renata Borovica Gajic
David Eccles

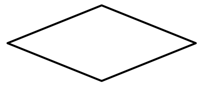
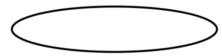
THE UNIVERSITY OF MELBOURNE

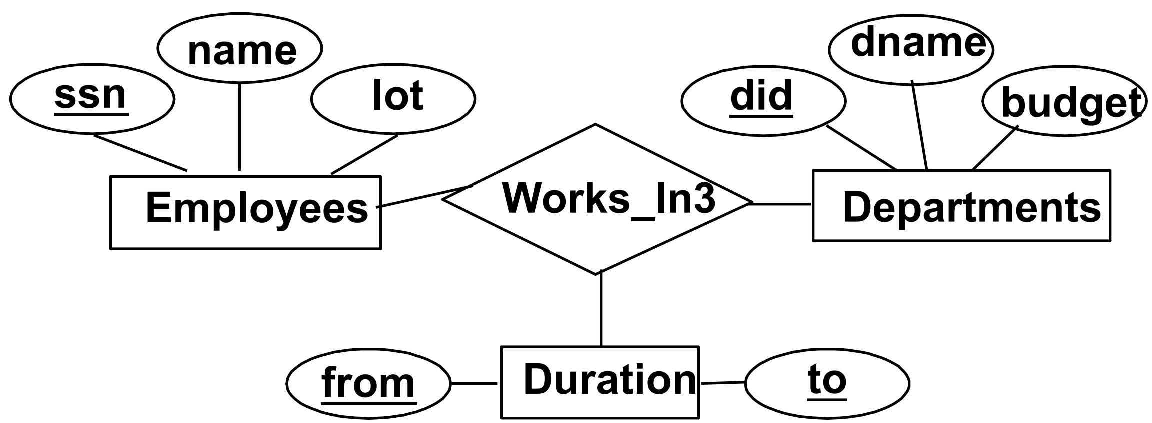
INFO90002 Database Systems & Information Modelling

Lecture 04

Logical & Physical Modelling

Last time: Intro to Modelling

- Basic ER modeling concepts
  - Entities, 
  - Relationships 
  - Attributes (Key Attributes) 
- Constraints
  - Key Constraints M:M 1:M 1:1
  - Participation Constraints
    - Total
    - Partial
- Conceptual Design



L3 Homework

1. A professor teaches at least one subject.
2. A professor can teach one subject (no more, no less).
3. All professors teach exactly one subject (no more, no less), and every subject must be taught by some professor.

L3 Homework Answers

1

2

3

Today

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

Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems

THE UNIVERSITY OF MELBOURNE

Relational Model

Relational Data Model

- **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 (Tuples and Attributes/fields)
  – Keys & Foreign Keys to link Relations

Enrolled

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

Student

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 (attribute).

    Example:

    Student(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)
  - ***Instance*** : a ***table***, with rows and columns.

    #rows = *cardinality*

    #fields = *degree (or arity)*
- You can think of a relation as a *set* of *rows* or *tuples*.
  - all rows are *distinct*
  - *no order* among rows

Example Instance of Students Relation

Students

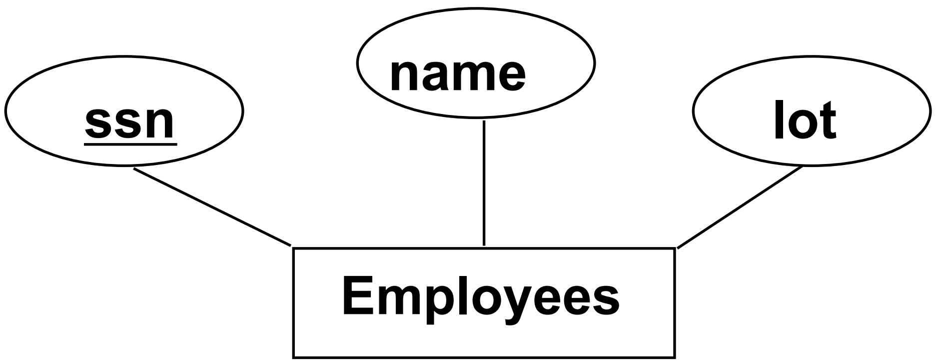
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

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

Logical Design: ER to Relational Model

In logical design **entity** set becomes a **relation.** Attributes become attributes of the relation.

Conceptual Design: → **Logical Design**:

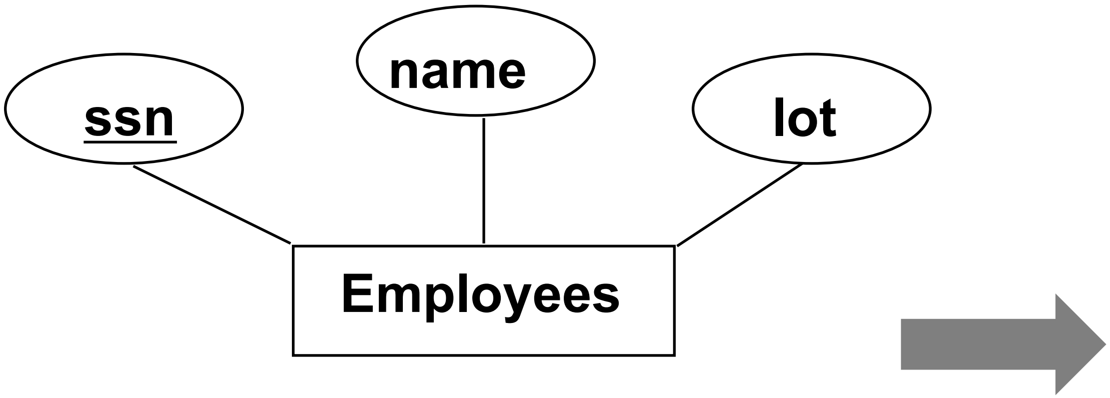


Employees = (<u>ssn</u>, name, lot)

ER to Logical to Physical

In physical design we choose data types

1. Conceptual Design:



3. Physical Design:
Employees
(ssn CHAR(11),
name CHAR(20),
lot INTEGER)

2. Logical Design:
Employees (ssn, name, lot)

The Entire Cycle

1. Conceptual Design

↓

2. Logical Design

↓

3. Physical Design

↓

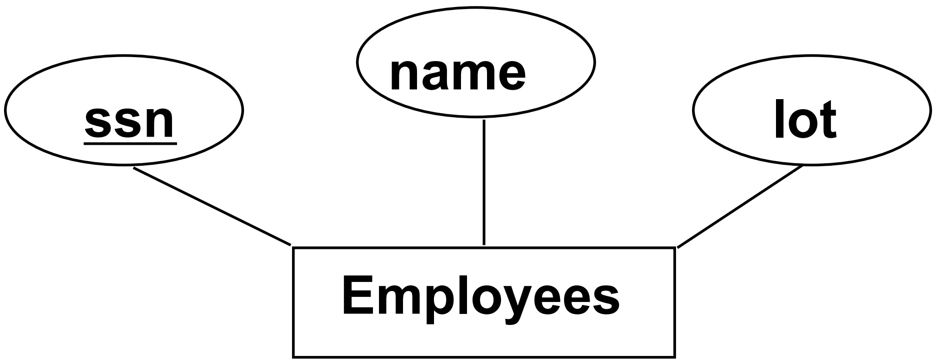
4. Implementation

↓

5. Create Instance

The Entire Cycle

1. Conceptual Design:



2. Logical Design:

Employees (ssn, name, lot)

3. Physical Design:

Employees
(ssn CHAR(11),
 name CHAR(20),
 lot INTEGER)

4. Implementation:

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

5. Instance:

EMPLOYEES

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

THE UNIVERSITY OF MELBOURNE

Key & Integrity Constraints

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

Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems

THE UNIVERSITY OF MELBOURNE

Keys

- Keys are a way to associate tuples in different relations
- Keys are one form of **integrity constraint (IC)**
- **Example:** *Only students can be enrolled in subjects.*

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

FOREIGN Key

PRIMARY Key

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 and no subset of the fields is a superkey (minimal subset)
- Out of all keys *one* is chosen to be the ***primary key*** of the relation. Other keys are called ***candidate*** keys.
- Each *relation* has a *primary key*.

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

- **Your turn:**
  1. *Is sid* a key for Student?
  2. What about *name*?
  3. Is the set {*sid, gpa*} a superkey? Is the set {*sid, gpa*} a key?
  4. Find a primary key from this set {*sid*, *login*}

Selecting the Primary Key

Superkeys

Keys

Candidate Keys

Primary Key

- Superkey – a set of fields that contains the key
- Keys are columns that in combination or alone can uniquely identify the tuple (row)
- Candidate keys are all the possible key combinations that could be the Primary Key
- Of all candidate keys the database designer *identifies* the primary key. The primary key is the *fewest number of columns* that can uniquely identify a key
- N.B.* Not all relations will have a key. In those cases the database designer will add a *surrogate key*

*N.B. means Nota Bena latin for "Note well" or in David speak "This information is important!")

Primary and Candidate Keys in SQL

- There are possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*. Keys must be chosen carefully.

Example:

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*** : A set of fields in one relation that is used to 'refer' to a tuple in another relation. The foreign key must correspond to the primary key of the other relation.

- If all foreign key constraints are enforced in a DBMS, we say a ***referential integrity*** is achieved.

Foreign Keys in SQL

Example: *Only students listed in the Students relation should be allowed to enroll in 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 (sid))
```

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*'.)

- Note: 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.

- This is also known as **Schema on Write**
  - The schema table structure is known in advance of the data to be inserted into it

THE UNIVERSITY OF MELBOURNE

Translating ER Models

From Conceptual, to Logical to Physical Entity Relationship Models

Relational Model

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

Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems

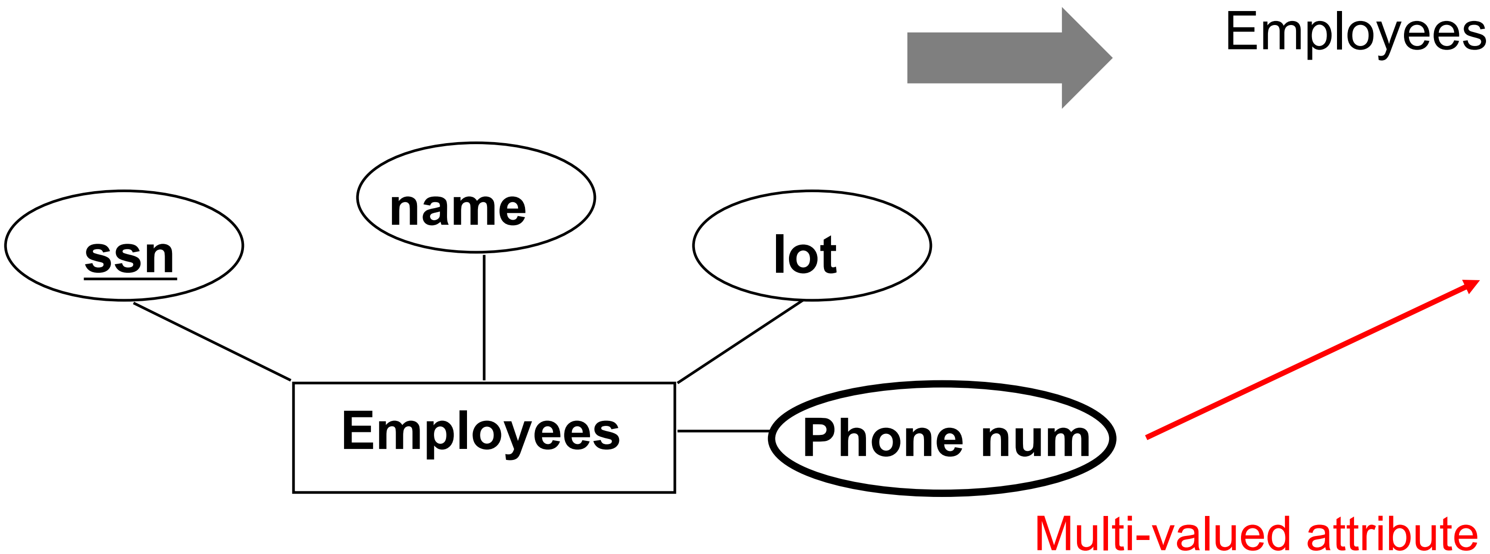
Multi-valued attributes in logical design

- Multi-valued attributes need to be unpacked (flattened) when converting to logical design.

Example:

For employees we need to capture home phone number and work phone number.

Conceptual Design:

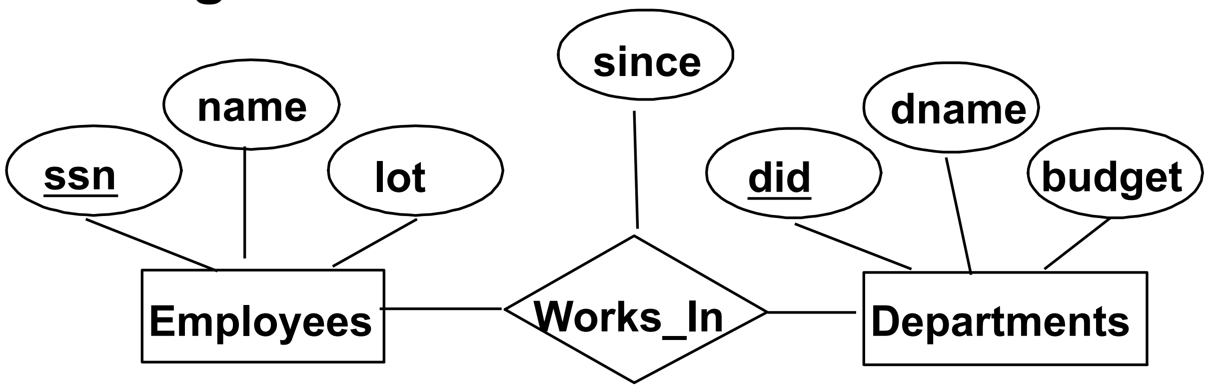


Logical Design:

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

ER to Logical Design

Conceptual Design:



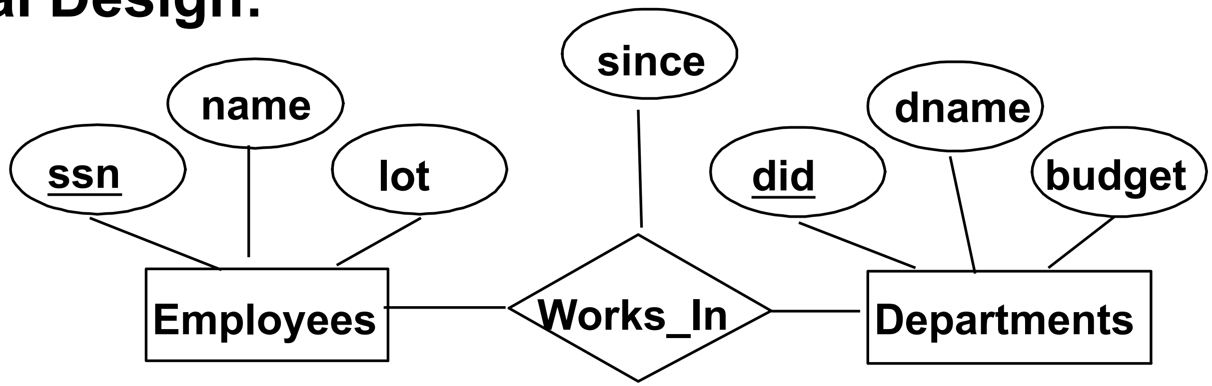
Logical Design:

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

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

ER to Logical Design

Conceptual Design:



Logical Design:

Employees = (ssn,
name
lot)

Departments = (did,
dname,
budget)

Works_In = (*ssn*,
did,
since)

Keys from connecting entities become PFK

Note:
Underline = PK,
Italic = FK,
Underline + Italic = PFK

Logical to Physical Design

Logical Design:

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

Departments = (<u>did</u>, dname, budget)

Works_In = (<u>*ssn*</u>, <u>*did*</u>,since)

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

Physical Design:

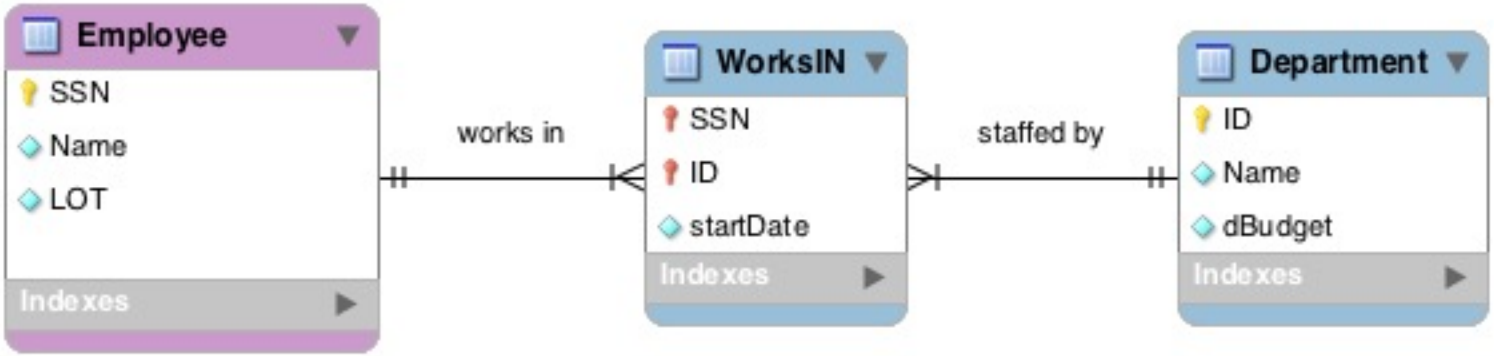
Employees
(<u>ssn</u> CHAR(11),
name CHAR(20),
lot INTEGER)

Departments
(<u>did</u> INTEGER,
dname CHAR(20),
budget FLOAT)

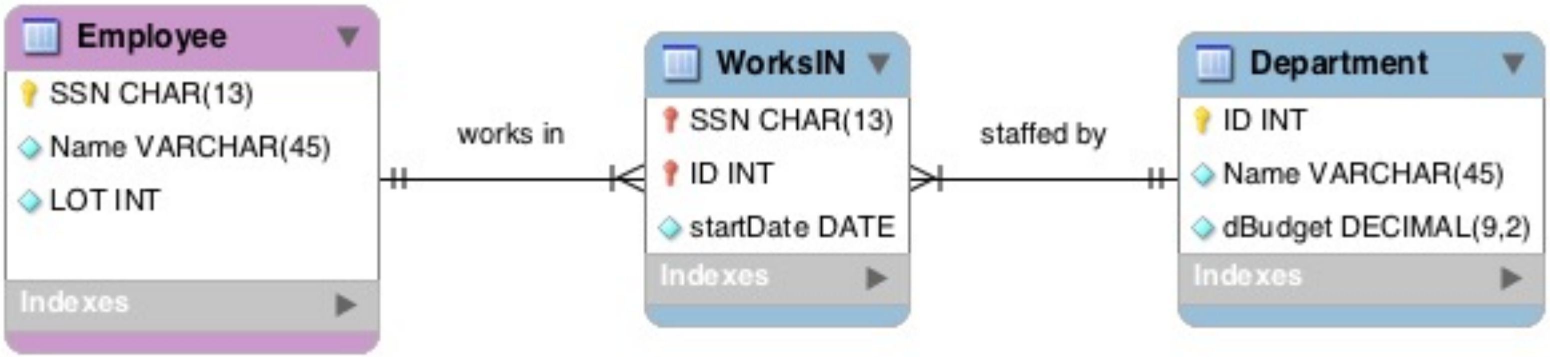
Works_In(
<u>*ssn*</u> CHAR(11),
<u>*did*</u> INTEGER,
since DATE)

Logical to Physical Model

- Logical Model



- Physical Model



Implementation (CREATE TABLE)

Logical Design:

Employees = (ssn, name, lot)

Departments = (did, dname, budget)

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

Note: Underline = PK
Italic = FK,
Underline and Italic = PFK

Implementation:

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

Example Instances

Employees

<u>ssn</u>	name	lot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Departments

<u>did</u>	dname	budget
101	Sales	10K
105	Purchasing	20K
108	Databases	1000K

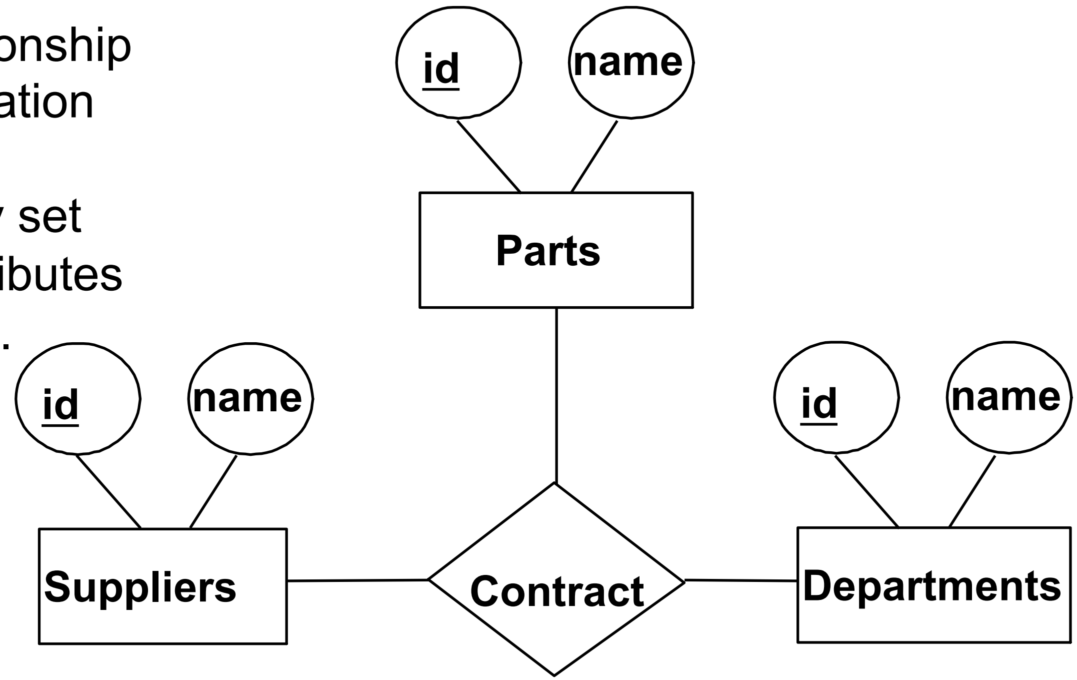
Works_In

<u>ssn</u>	<u>did</u>	since
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 (

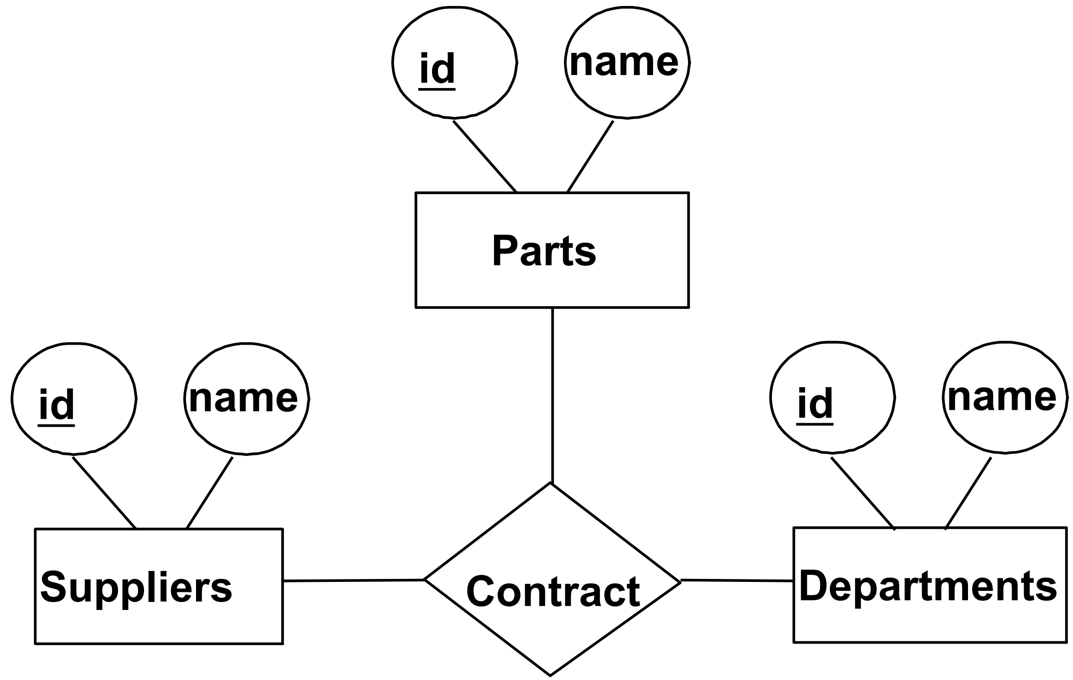
supplier_id,

part_id,

department_id)

Note:
Underline = PK,
Italic = FK,
Underline and italic = PFK

ER to Logical to Implementation Example 2



Logical Design:

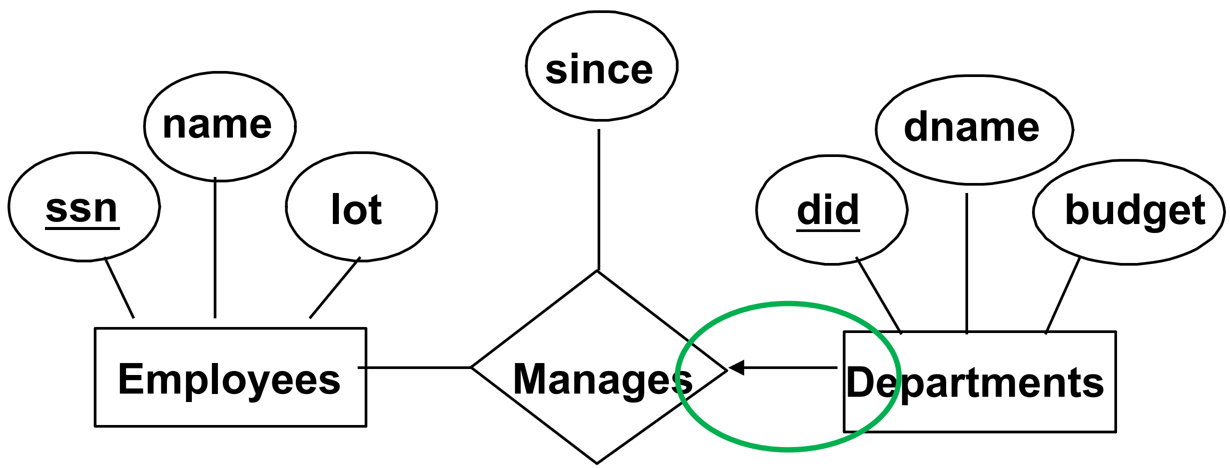
Contracts (
    supplier_id,
    part_id,
    department_id)

Implementation:

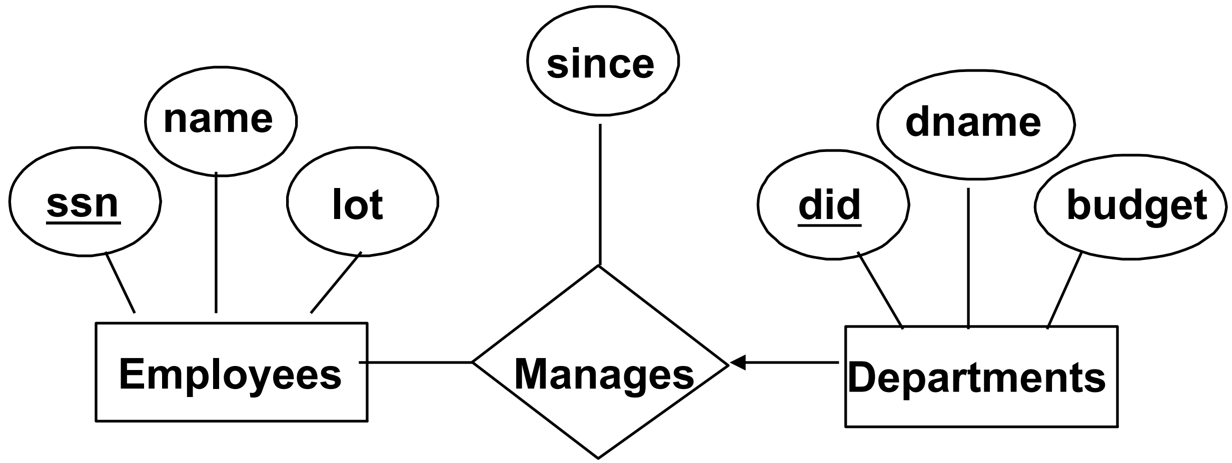
```
CREATE TABLE Contracts (
        supplier_id  INTEGER,
        part_id  INTEGER,
        department_id  INTEGER,
        PRIMARY KEY (supplier_id, part_id, department_id),
        FOREIGN KEY (supplier_id)  REFERENCES Suppliers(id),
        FOREIGN KEY (part_id)      REFERENCES Parts(id),
      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

Review: Key Constraints in ER

- Each department 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.

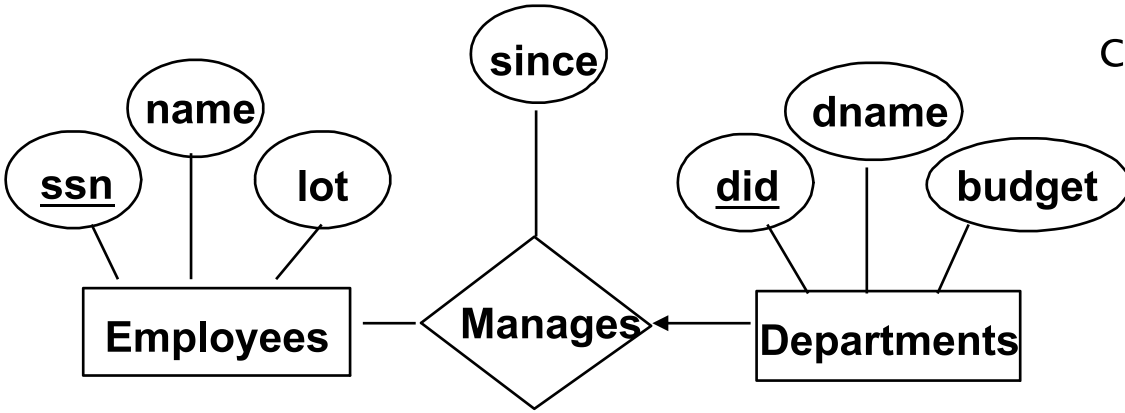
Employees = (ssn, name, lot)

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

Note: Underline = PK,
Italic= FK,
Underline and italic = PFK

Key Constraints in SQL

Implementation:




```
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  (ssn, 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 (ssn))
```

Which one is better?

Key Constraints rule

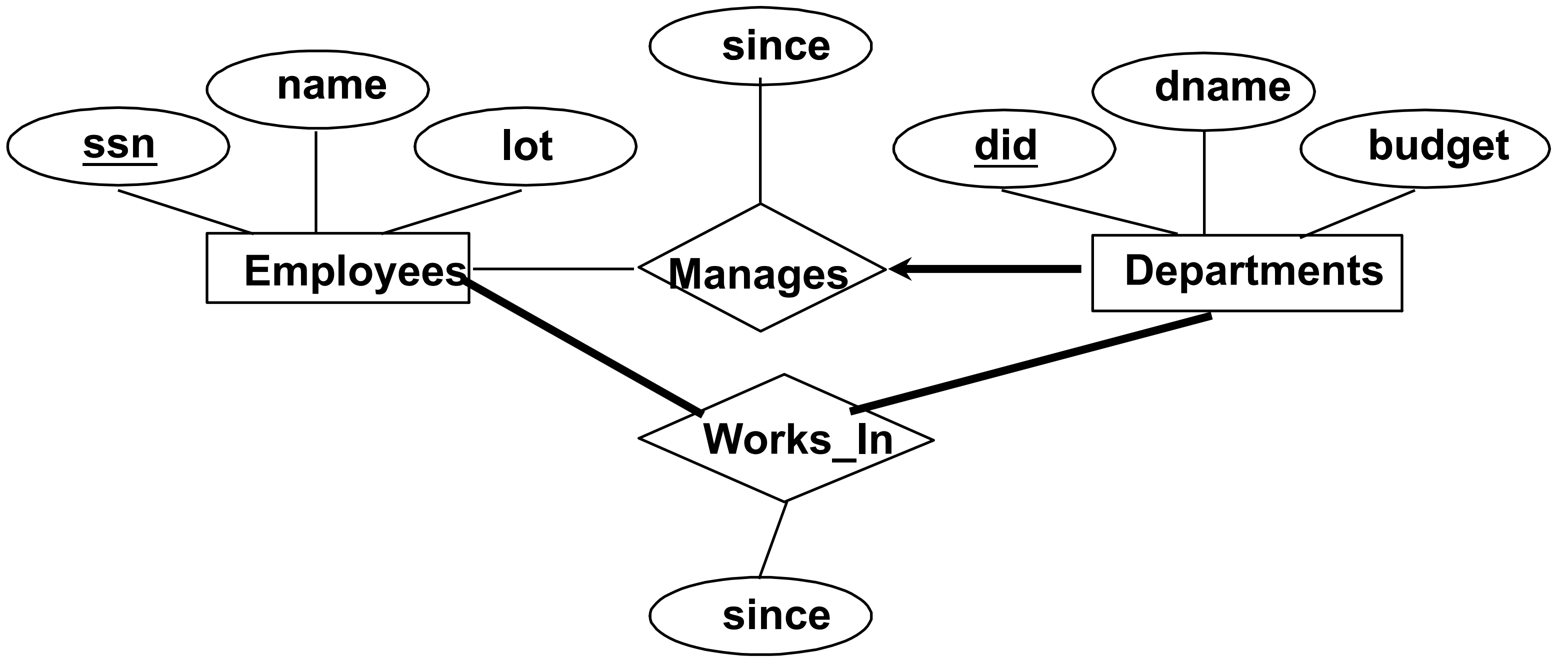
- RULE: Primary key from the one side becomes a foreign key on the many side
- This is the way to ensure that the key constraint holds

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

Each department will have a *single* manager

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*).



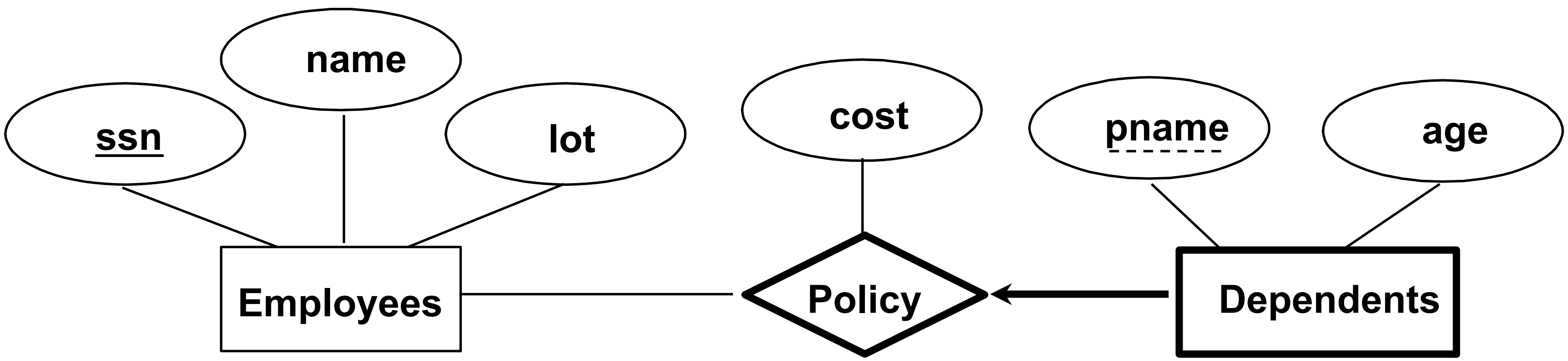
Participation Constraints in SQL

- We specify total participation with key words NOT NULL
  - NOT NULL = this field cannot be empty

```
CREATE TABLE Departments(
   did  INTEGER NOT NULL,
   dname  CHAR(20),
   budget  REAL,
   ssn  CHAR(11) NOT NULL,
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY  (ssn) REFERENCES Employees(ssn),
      ON DELETE NO ACTION)
```

Review: Weak Entities

- A ***weak entity*** can be identified uniquely only by considering the primary key of another (*owner*) entity.



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

Implementation:

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

Relational Model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used.
- Integrity constraints can be specified 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?*

- Be able to model a case study from conceptual to instance and all stages in between (conceptual, logical, physical, implementation and instance)
- Translate conceptual (ER) into logical & physical design
- Understand integrity constraints
- Use DDL of SQL to create tables with constraints

* All material is examinable – these are the suggested key skills you would need to demonstrate in an exam scenario