Databases and Information Systems

Introduction
Conceptual Modeling

Dmitriy Traytel
slides mostly by Marcos Vaz Salles

UNIVERSITY OF COPENHAGEN

What is a DBMS?

- Database: A very large, integrated collection of data.
- Models real-world **enterprise**.
  - Entities (e.g., students, professors, courses)
  - Relationships (e.g., “attends”, “teaches”, “is a TA for”)
- A **Database Management System (DBMS)** is a software package designed to store and manage databases.

Database Management Systems (DBMS)

A Database Management System (DBMS) provides…

Database Management Systems (DBMS)

A Database Management System (DBMS) provides…

… **efficient**, **reliable**, **convenient**, and **safe multi-user** storage of and access to **massive** amounts of **persistent** data.

Database Management Systems (DBMS)

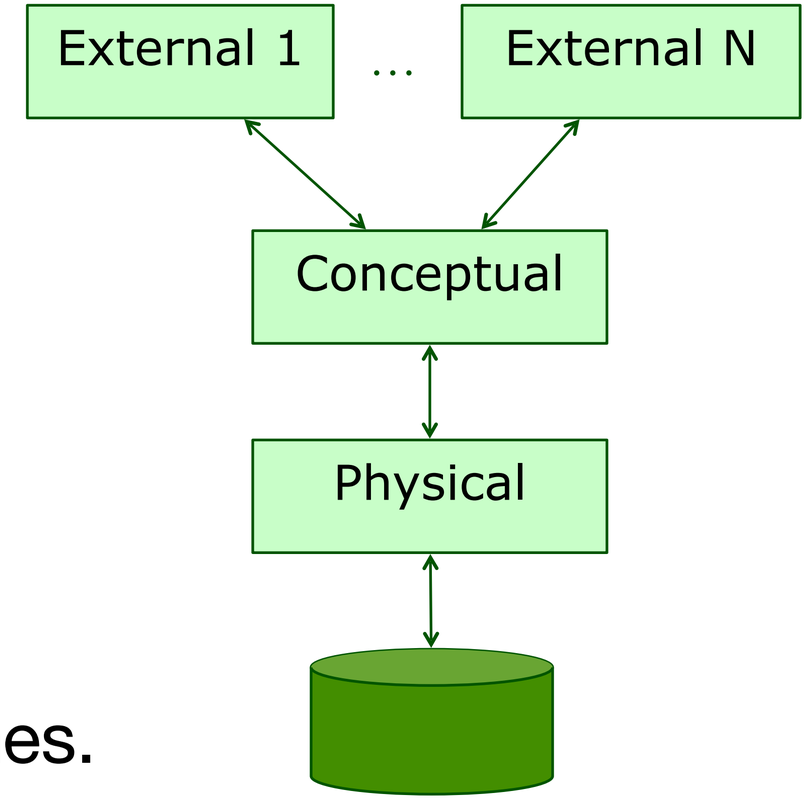
A Database Management System (DBMS) provides…

… **efficient**, **reliable**, **convenient**, and **safe multi-user** storage of and access to **massive** amounts of **persistent** data.

We will focus on Relational DBMS (RDBMS)

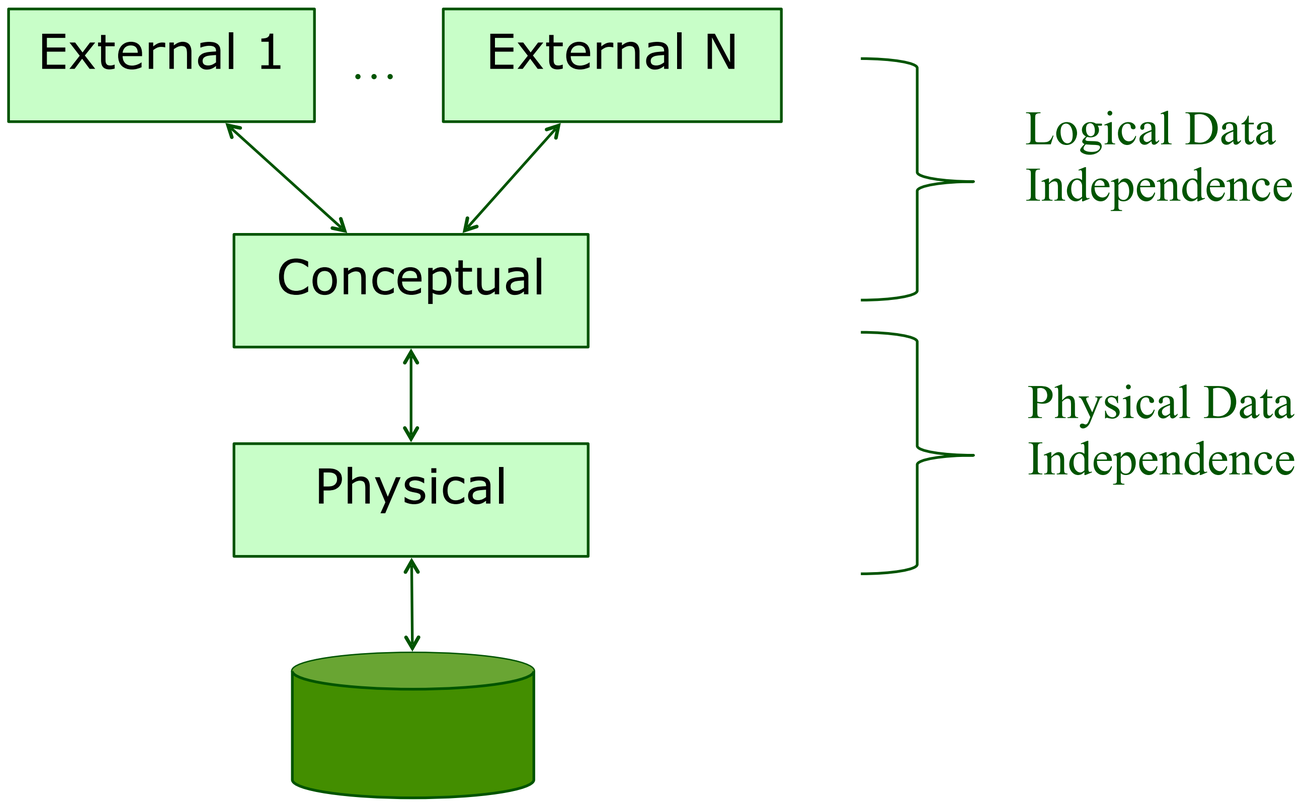
Why Use a DBMS?

- **Data independence** and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access and recovery from crashes.



Source: Ramakrishnan & Gehrke (partial)

Levels of Abstraction



Source: Ramakrishnan & Gehrke (partial)

Databases: Key Concepts and People

- **Key Concepts**
  - Data model
  - Schema versus data
  - Data definition language (DDL)
  - Data manipulation (or query) language (DML)

- **Key People**
  - DBMS implementer
  - Database designer
  - Database application developer
  - Database administrator

Source: Widom (partial)

Databases: Key Concepts and People

- **Key Concepts**
  - Data model
  - Schema versus data
  - Data definition language (DDL)
  - Data manipulation (or query) language (DML)
- **Key People**
  - DBMS implementer
  - Database designer
  - Database application developer
  - Database administrator

What are examples of the key concepts in the case of a real application, e.g., a social network?

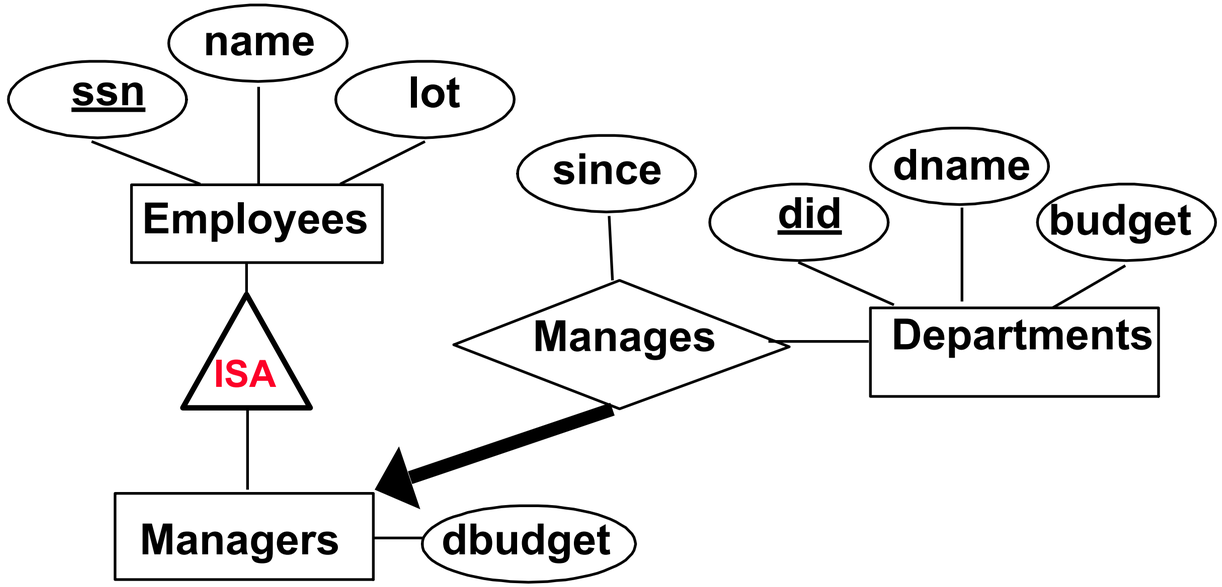
Source: Widom (partial)

What should we learn today?

- Explain the concepts of entity (set), relationship (set) and express these concepts in entity-relationship (E/R) diagrams
- Explain and express constraints (key, uniqueness, and ref. integrity) in E/R diagrams
- Explain and capture in E/R diagrams weak entity sets and ISA hierarchies
- Analyze trade-offs and argue for the advantages and disadvantages of a E/R design

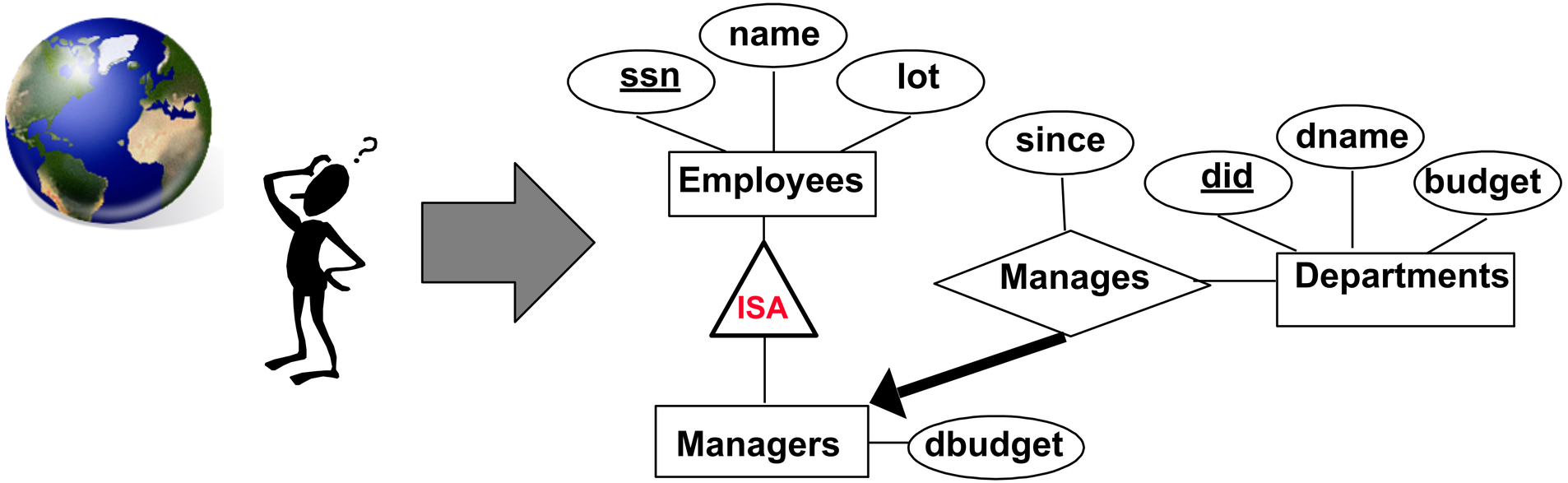
- Explain what a data model is, along with the distinction between schemas and instances
- Define and explain the relational data model
- Define and explain the main classes of integrity constraints in the relational model, in particular key and foreign key constraints
- Model relational schemas and express them in SQL

Conceptual Models



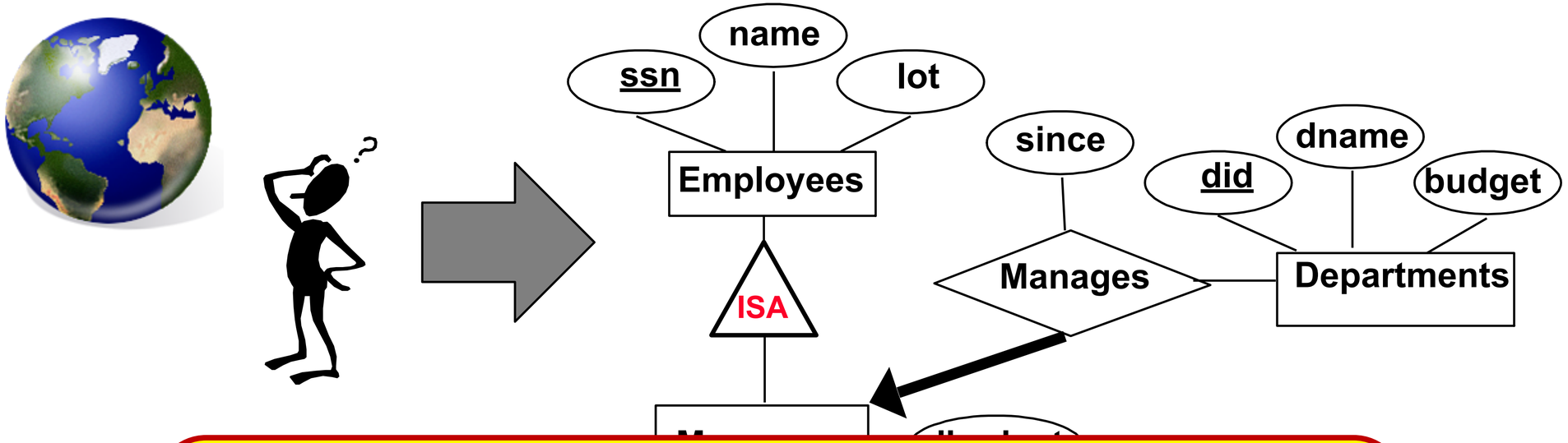
Source: Ramakrishnan & Gehrke

Conceptual Models



Source: Ramakrishnan & Gehrke

Conceptual Models



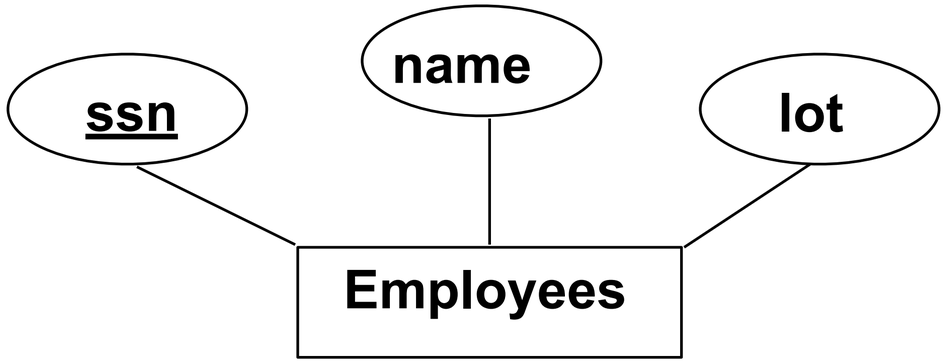
How do we create a conceptual model?

Source: Ramakrishnan & Gehrke

E/R Model Basics

- *Entity:* Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes*

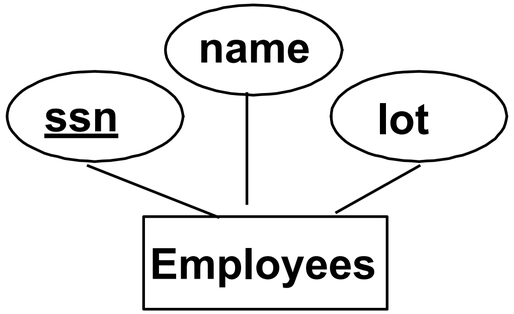
- *Entity Set*: A collection of similar entities. E.g., all employees
  - All entities in an entity set have the same set of attributes
  - Each entity set has a *key*
  - Each attribute has a *domain*

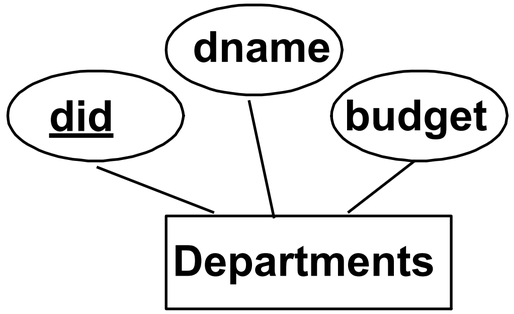


E/R Model Basics

- *Relationship*: Association among two or more entities.
  - E.g., Attishoo works in Pharmacy department.

- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En
  - Each relationship in R involves entities e1 in E1, ..., en in En

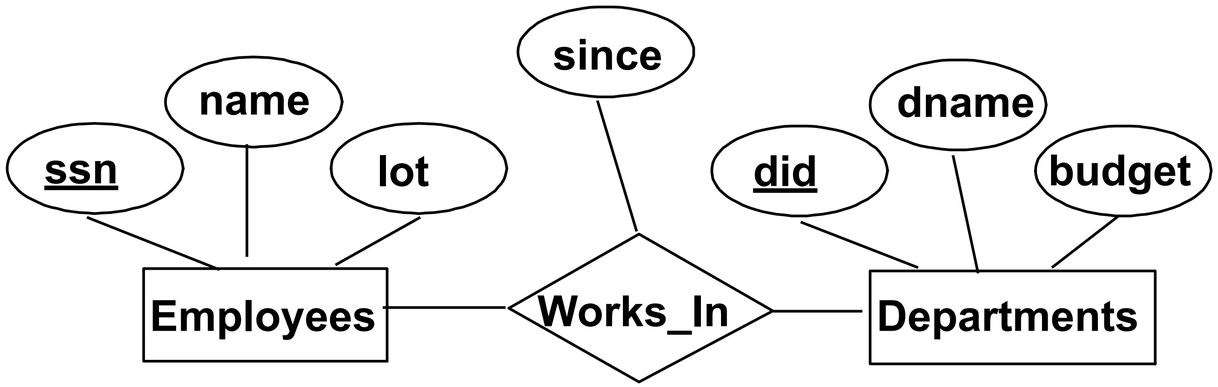




E/R Model Basics

- *Relationship*: Association among two or more entities.
  - E.g., Attishoo works in Pharmacy department.

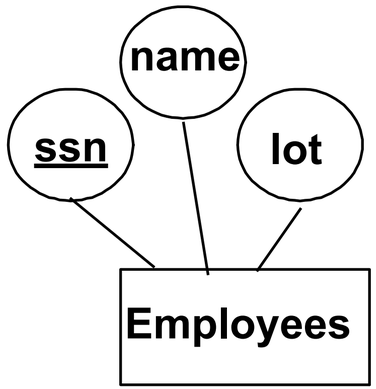
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En
  - Each relationship in R involves entities e1 in E1, ..., en in En



Source: Ramakrishnan & Gehrke (partial)

Relationships (Contd.)

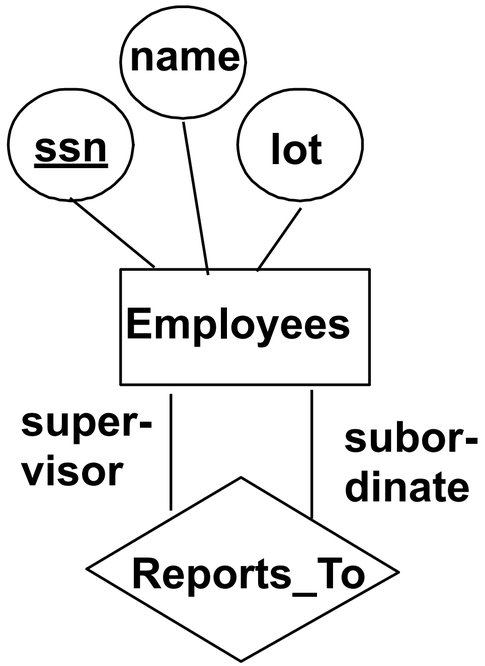
- Relationships can have roles
- For example, if we want to capture supervisor-subordinate relationship:



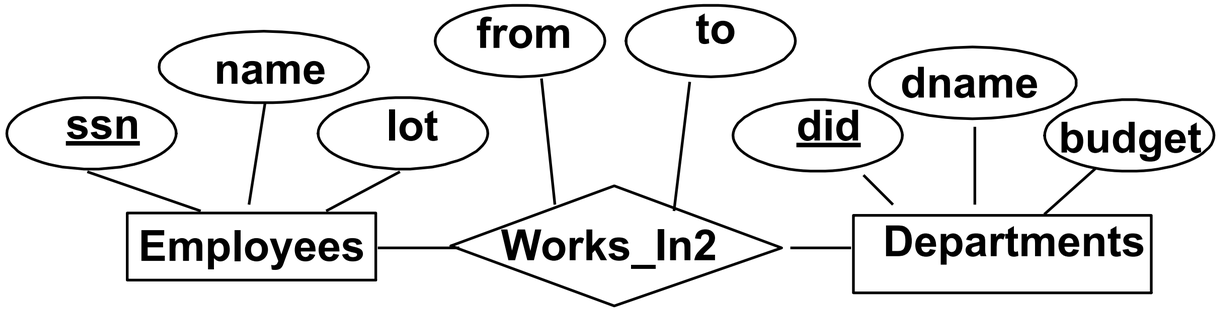
-

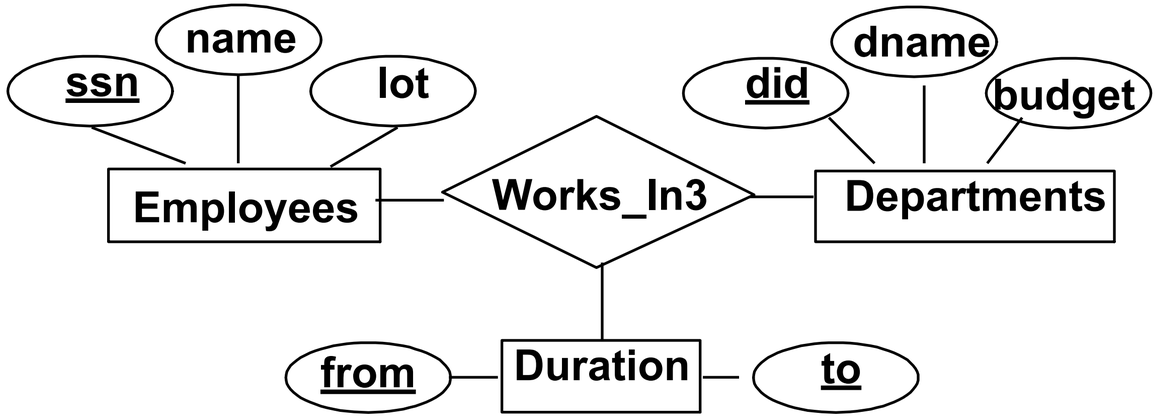
Relationships (Contd.)

- Relationships can have roles
- For example, if we want to capture supervisor-subordinate relationship:



How are these different?

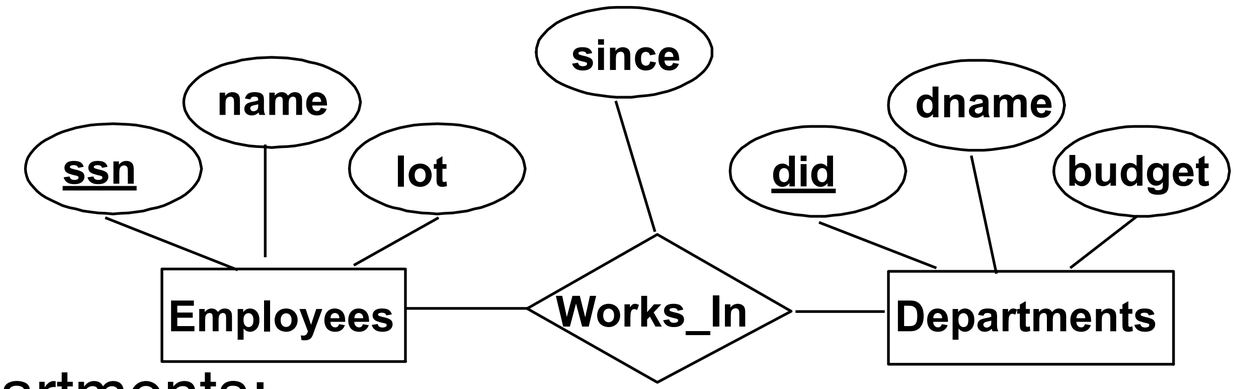




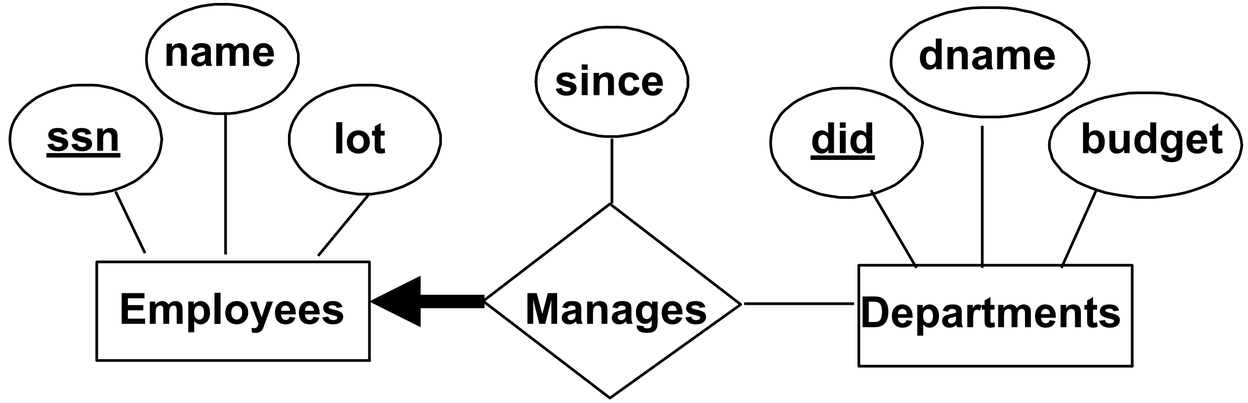
- Attributes on relationship vs. ternary relationship

Source: Ramakrishnan & Gehrke (partial)

Uniqueness Constraints



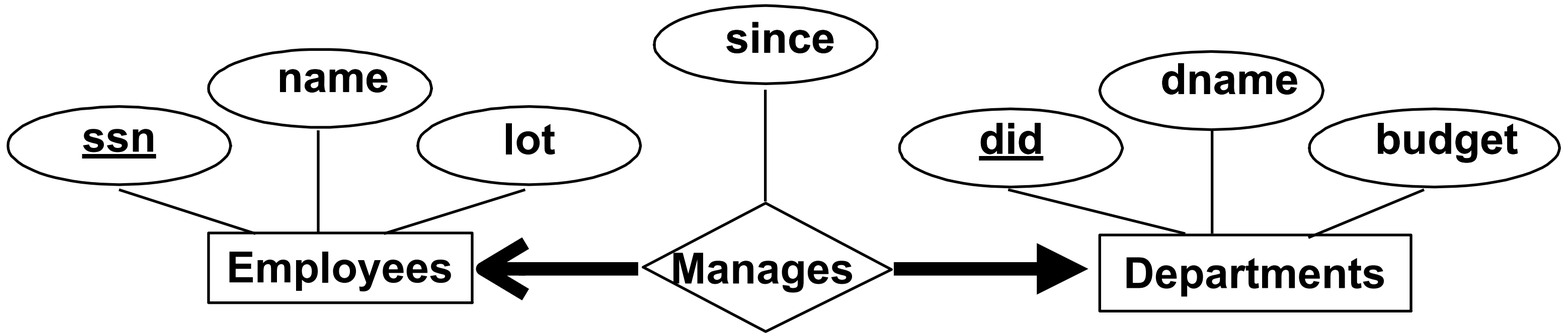
- An employee can work in many departments;
  a department can have many employees (many-many)
- Each department has **at most one** manager,
  according to the *uniqueness constraint* on Manages (many-one)



Source: Ramakrishnan & Gehrke (partial)

Referential Integrity

- A department has **exactly one** manager,
  but an employee may manage **at most one** department:



- Referential-integrity constraint says that employee
  is required to exist in this one-one relationship

Source: Ramakrishnan & Gehrke (partial)

E/R Modeling Exercise

Drugwarehouse.com has hired you to design its database schema.

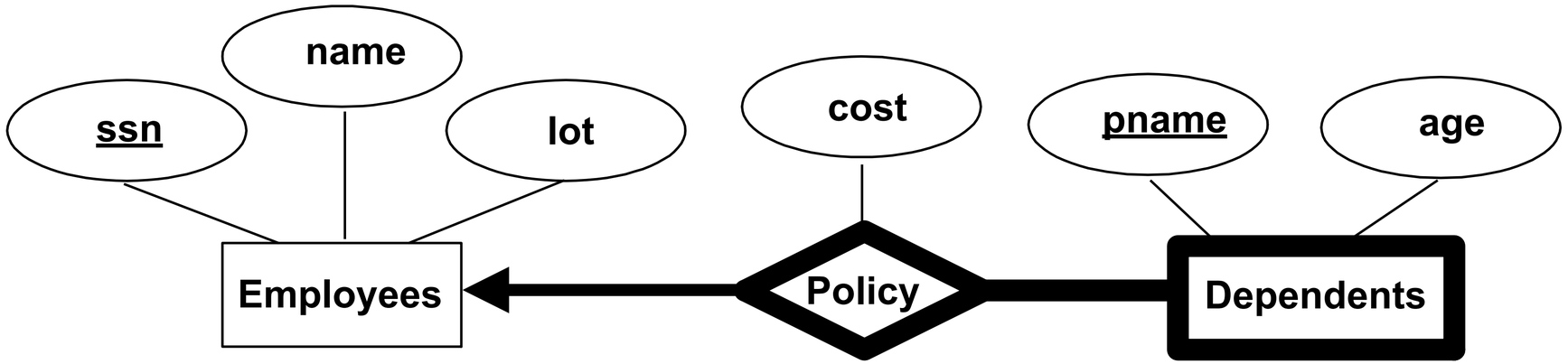
Here is the information that you gathered:

- Patients are identified by their SSN, and we also store their names and age.
- Doctors are identified by their SSN, and we also store their names and specialty.
- Each patient has exactly one primary care physician, and we want to know since when the patient has been with her primary care physician.
- A doctor can be a primary care physician for multiple patients.

Source: Ramakrishnan & Gehrke (partial)

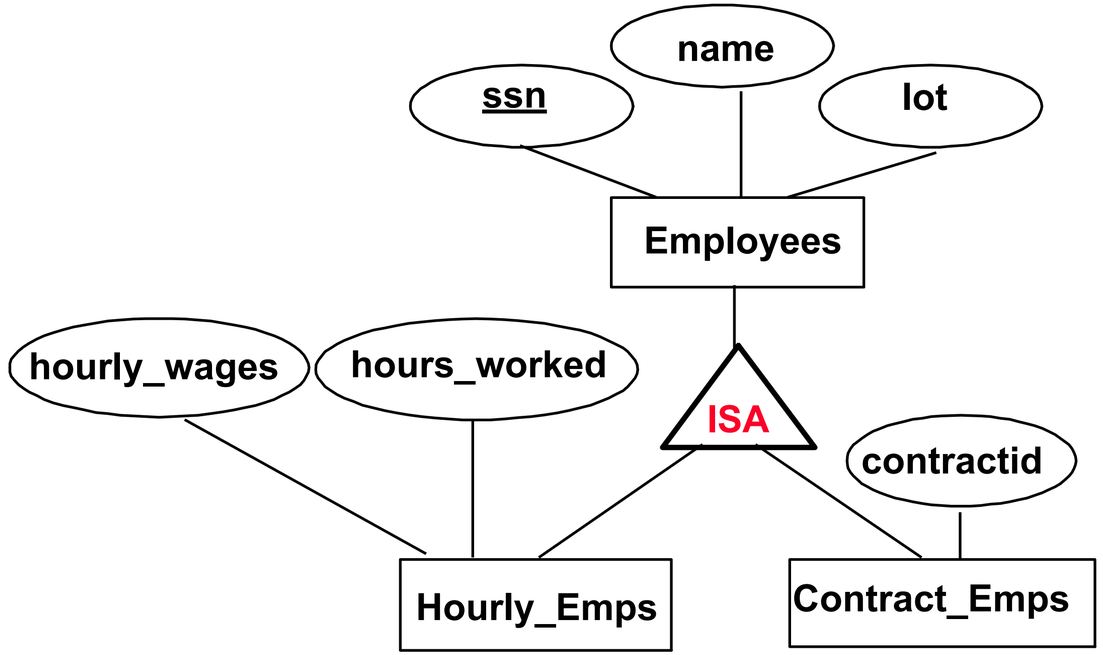
Weak Entities

- 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 (one owner, many weak entities).
  - Keys of weak entity set (partially) composed of attributes from *supporting* entity sets.
-



Source: Ramakrishnan & Gehrke (partial)

ISA (`is a') Hierarchies



- Similar to object-oriented modeling
- In OO, objects are in one class only.
  - Subclasses inherit from superclasses.
- In contrast, E/R entities have representatives in all subclasses to which they belong.
  - Rule: if entity e is represented in a subclass, then e is represented in the superclass. (and recursively up the tree).

Source: Ramakrishnan & Gehrke (partial) / Bulskov (partial)

Conceptual Design Using the E/R Model

- Design principles:
  - Be faithful to the application.
  - Avoid redundancy.
  - Strive for simplicity.

- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary?

- Constraints in the E/R Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in E/R diagrams. (Which for example?)

Source: Ramakrishnan & Gehrke

Entity vs Attribute

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?

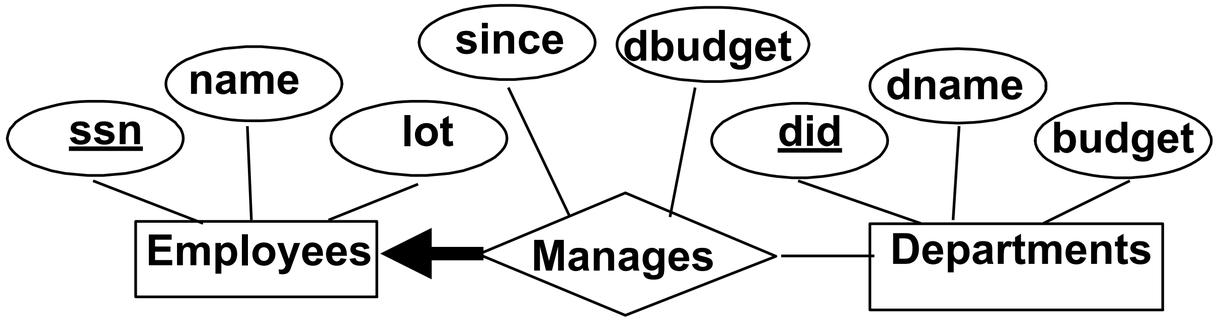
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).

Source: Ramakrishnan & Gehrke

Entity vs Relationship

- E/R model OK if a manager gets a separate discretionary budget for each department.

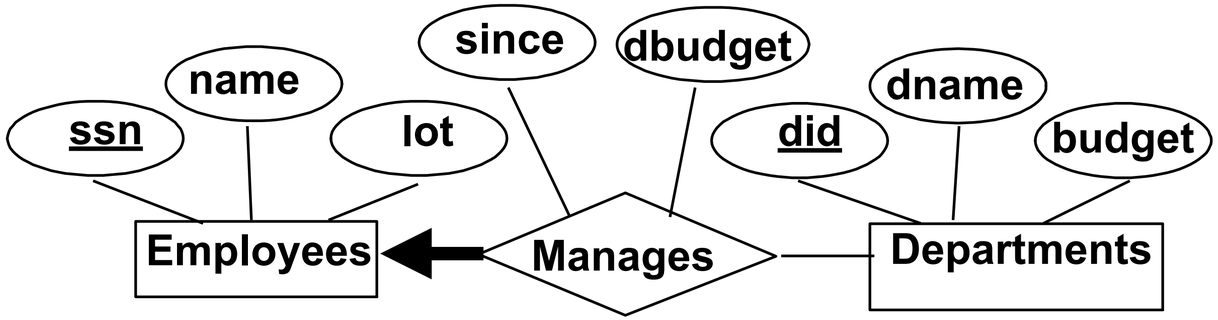
- What if a manager gets a discretionary budget that covers *all* managed departments?
  - Redundancy: *dbudget* stored for each department managed by manager.
  - Misleading: Suggests *dbudget* associated with department-manager combination.



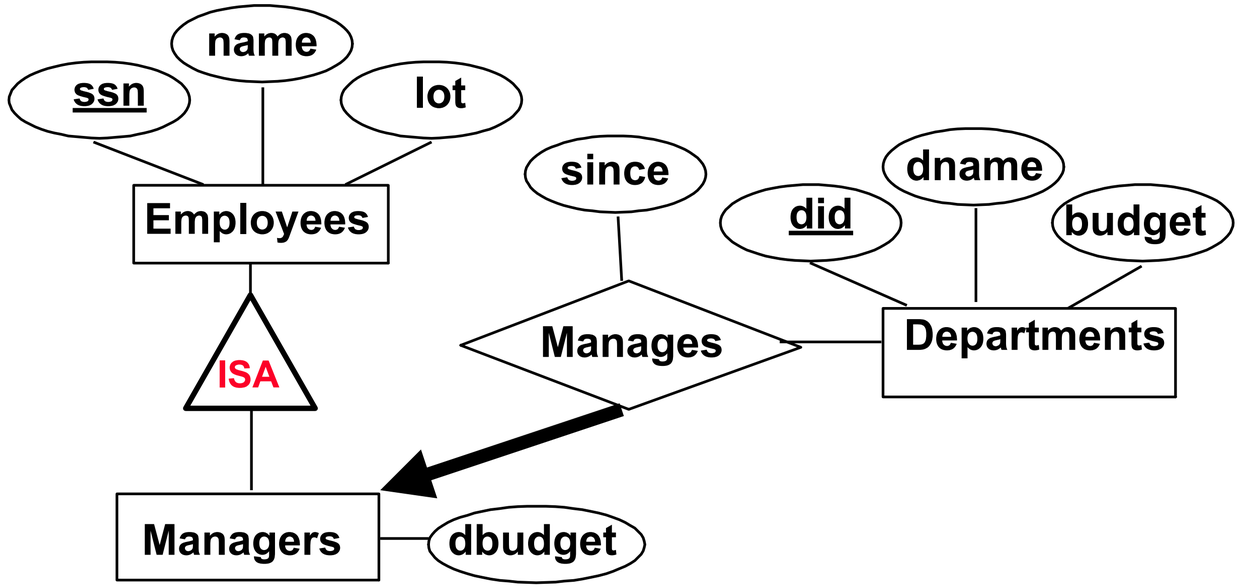
Source: Ramakrishnan & Gehrke (partial)

Entity vs Relationship

- E/R model OK if a manager gets a separate discretionary budget for each department.



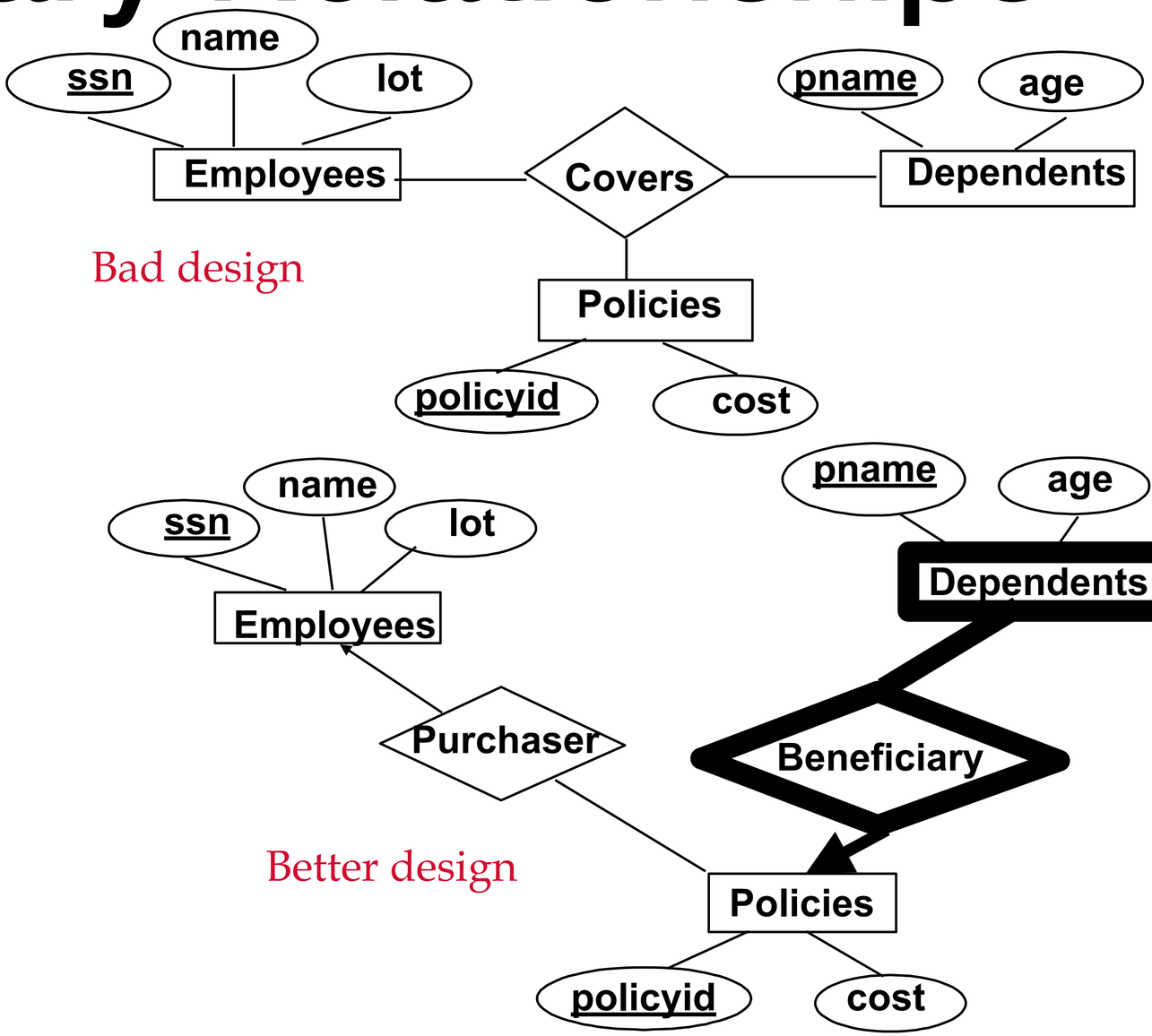
- What if a manager gets a discretionary budget that covers *all* managed departments?
  - Redundancy: *dbudget* stored for each department managed by manager.
  - Misleading: Suggests *dbudget* associated with department-manager combination.



Source: Ramakrishnan & Gehrke (partial)

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.
- What are the additional constraints in the second diagram?



Source: Ramakrishnan & Gehrke (partial)

Binary vs Ternary Relationships (Contd.)

- Previous example illustrated a case when two binary relationships were better than one ternary relationship.

- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*. No combination of binary relationships is an adequate substitute:
  - S "can-supply" P, D "needs" P, and D "deals-with" S does not imply that D has agreed to buy P from S.
  - How do we record *qty*?
  - However, we can model Contracts as an *entity* instead, and then use binary relationships

Summary of Conceptual Design

- Conceptual design follows requirements analysis
  - Yields a high-level description of data to be stored
- E/R model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships)
- Some additional constructs: *weak entities*, *ISA hierarchies*.
- Note: There are many variations on E/R model.

Source: Ramakrishnan & Gehrke (partial)

Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *uniqueness constraints*, and *referential-integrity constraints*. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for a use case.

Source: Ramakrishnan & Gehrke (partial)

Summary of E/R (Contd.)

- E/R design is **subjective**. There are often many ways to model a given scenario! Analyzing alternatives can be tricky. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, how to define constraints.

- Ensuring good database design: resulting relational schema should be analyzed and refined further.

- Functional dependencies and normalization techniques can be useful but cannot be captured in a E/R model. More on this later in the course ☺

Source: Ramakrishnan & Gehrke (partial)

Questions so far?

Why Study the Relational Model?

- Codd, 1969
- Most widely used model
  - Vendors: IBM, Microsoft, Oracle, Sybase/SAP, etc.
- Query languages
  - Relational calculus
  - Relational algebra
  - SQL
- “Legacy systems” in older models
  - E.G., IBM’s IMS
- Competitors
  - In the early 90s: *object-oriented model*
  - A synthesis: *object-relational model* (Oracle, DB2)
  - *Document-oriented*: XML, JSON
  - Often collected under the term *NoSQL*

Source: Ramakrishnan & Gehrke

Relational Database: Definitions

- ***Relational database:*** a set of ***relations***
- ***Relation:*** made up of two 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* or *arity*.
- Can think of a relation as a ***set*** of rows or ***tuples*** (i.e., all rows are distinct).

Source: Ramakrishnan & Gehrke (partial)

Example Instance of Students Relation

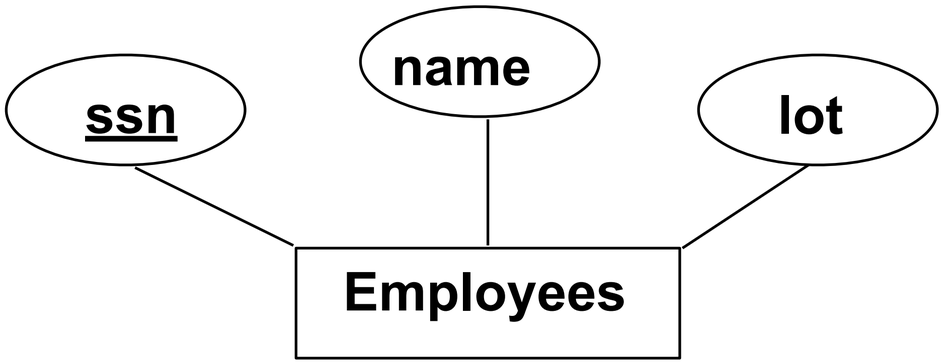
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree = 5, all rows distinct
- Do all columns in a relation instance have to be distinct?

Source: Ramakrishnan & Gehrke

Logical Design: E/R to Relational

- Entity sets to tables.




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

```
Employees
(ssn : string,
 name : string,
 lot : int)
```

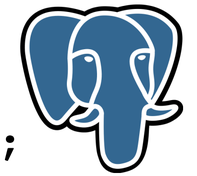
https://traytel.bitbucket.io/rc-eval/

Example Instance

Employees

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

```
INSERT INTO Employees(ssn,name,lot)
VALUES (0983763423, 'John', 10), (9384392483, 'Jane', 10), (3743923483, 'Jill', 20);
```



```
Employees
("0983763423", "John", 10) ("9384392483", "Jane", 10) (“3743923483", "Jill", 20)
```

https://traytel.bitbucket.io/rc-eval/

Integrity Constraints (ICs)

- IC: condition that must be true for *any* instance of the database
  - Domain constraints
  - Key constraints
  - Foreign key constraints (later)
- A *legal* instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances
  - Avoids data entry errors too!

Source: Ramakrishnan & Gehrke

Primary Key Constraints

- A set of fields is a *superkey* for a relation if:
  - No two distinct tuples can have same values in all fields belonging to the superkey
- A set of fields is a *key* if:
  - The set of fields is a superkey
  - No proper subset of the set of fields is a superkey
- If there is >1 key for a relation, one of the keys is chosen (by DBA) to be the *primary key*.
- E.g., {*ssn*} is a key for Employees. (What about {*name*}?) The set {*ssn, name*} is a superkey.

What does this mean?

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

Source: Ramakrishnan & Gehrke

Candidate Keys

- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.

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

- Each student is enrolled in at most one course
- No two students in a course get the same grade

Source: Ramakrishnan & Gehrke

Where do ICs Come From?

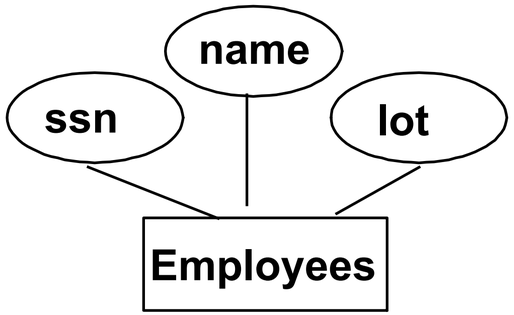
- ICs are based upon the semantics of the real-world enterprise that is being described in the database relations.

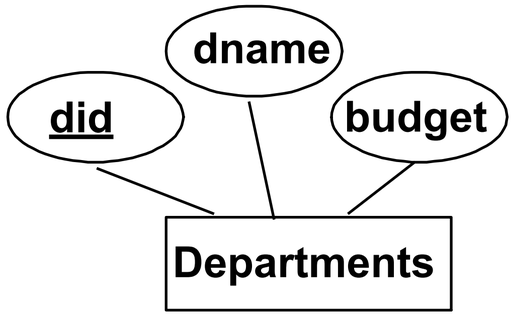
- We can check a database instance to see if an IC is violated, but we can NEVER infer that an IC is true by looking at an instance.
  - An IC is a statement about *all possible* instances!
  - From example, we know *name* is not a key, but the assertion that *sid* is a key is given to us.

- Key and foreign key ICs are the most common; more general ICs supported too.

Source: Ramakrishnan & Gehrke

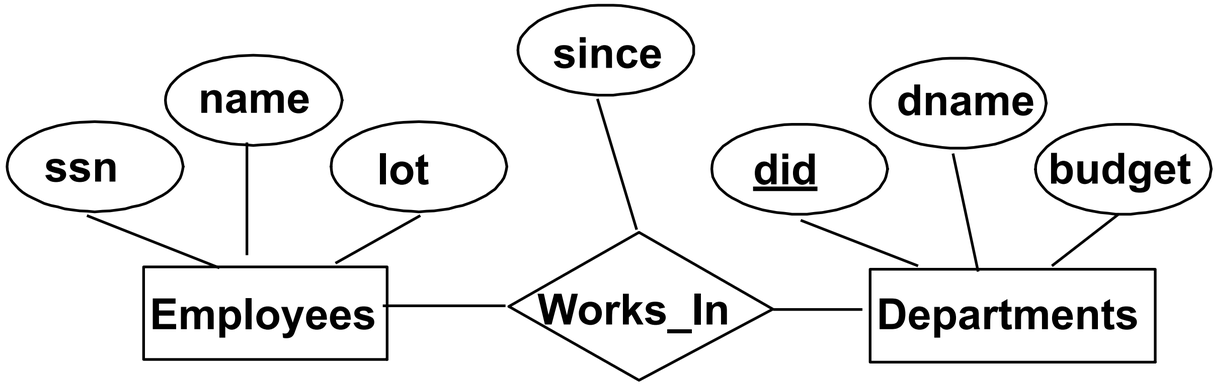
E/R to Relational (contd.)





Source: Ramakrishnan & Gehrke

E/R to Relational (contd.)



Source: Ramakrishnan & Gehrke

Relationship Sets to Tables

```
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))
```

Source: Ramakrishnan & Gehrke

Relationship Sets to Tables

```
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)
```

Source: Ramakrishnan & Gehrke

Example Instance

Employees

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

Departments

did	dname	budget
101	Sales	10K
105	Purchasing	20K
108	Databases	1000K

Works_In

ssn	did	since
0983763423	101	1 Jan 2003
0983763423	108	2 Jan 2003
9384392483	108	1 Jun 2002

Source: Ramakrishnan & Gehrke

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'.

- If all foreign key constraints enforced, *referential integrity* is achieved, i.e., no dangling references.
  - Not like HTML links!

Enforcing Referential Integrity

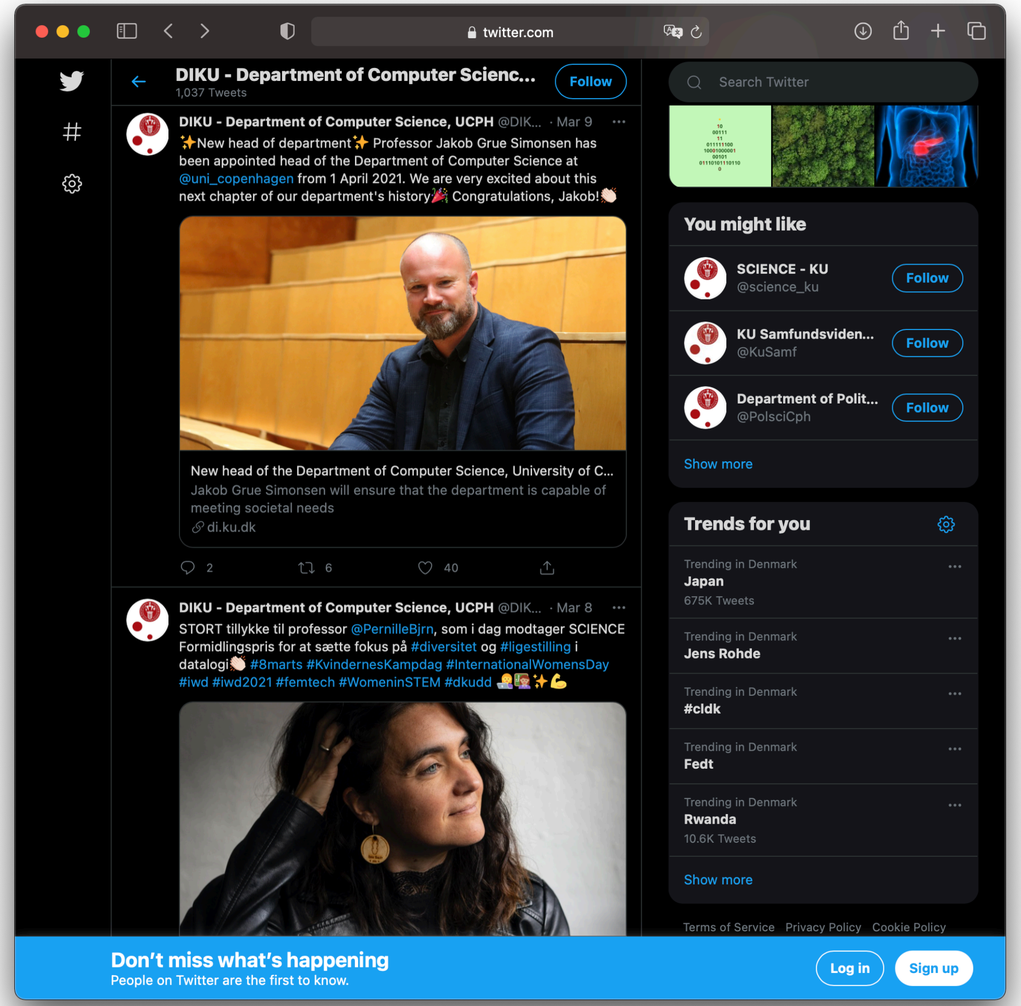
- What if a new “Works_In” tuple is added that references a non-existent employee?
  - Reject it!
- What if an Employee tuple is deleted?
  - Also delete all Works_In tuples that refer to it.
  - Disallow deletion of Employee tuple that is referred to.
  - Set *ssn* to some default value
  - Set *ssn* in Works_In to *null*, denoting *`unknown’*
- Similar if primary key of Employee tuple is updated

Source: Ramakrishnan & Gehrke

Discussion: Relational Twitter

- Let us say you would like to model two tables from Twitter:
  - Users
  - Tweets

- Discuss
  - Which attributes should each table have?
  - What are the keys in these tables? Would you also need foreign keys? If so, which?
  - Do not model Followers for now! But you may think about it. ☺
  - If you have time, try to write the SQL



What should we learn today?

- Explain the concepts of entity (set), relationship (set) and express these concepts in entity-relationship (E/R) diagrams
- Explain and express constraints (key, uniqueness, and ref. integrity) in E/R diagrams
- Explain and capture in E/R diagrams weak entity sets and ISA hierarchies
- Analyze trade-offs and argue for the advantages and disadvantages of a E/R design

- Explain what a data model is, along with the distinction between schemas and instances
- Define and explain the relational data model
- Define and explain the main classes of integrity constraints in the relational model, in particular key and foreign key constraints
- Model relational schemas and express them in SQL