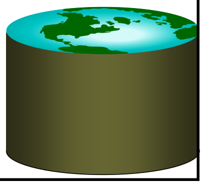
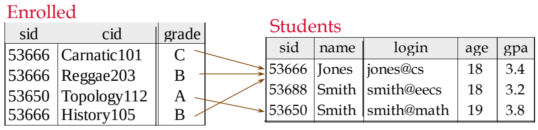
ER & Relational: Digging Deeper

R &G - Chapters 2 & 3



Databases Model the Real World

- **"Data Model" allows us to translate real world things into structures computers can store**
- **Many models: Relational, E-R, O-O, Network, Hierarchical, etc.**
- **Relational**
  - Rows & Columns
  - Keys & Foreign Keys to link Relations



Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

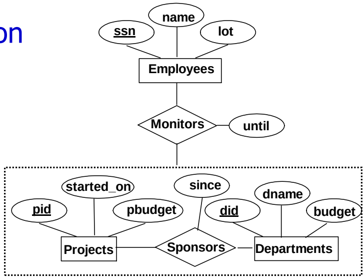
Students

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

Aggregation

Used to model a relationship involving a *relationship set*.

Allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.



Aggregation vs. ternary relationship?

❖ Monitors is a distinct relationship, with a descriptive attribute.

❖ Also, can say that each sponsorship is monitored by at most one employee.

Conceptual Design Using the ER Model

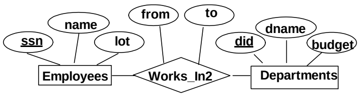
- **ER modeling *can* get tricky!**
- **Design choices:**
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?
- **Note constraints of the ER Model:**
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.
    - We'll refine things in our logical (relational) design

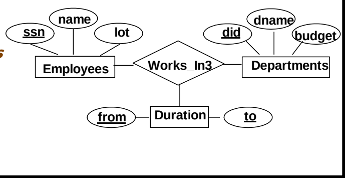
Entity vs. Attribute

- **Should *address* be an attribute of Employees or an entity (related to Employees)?**
- **Depends upon how we want to use 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, *address* must be modeled as an entity (since attribute values are atomic).

Entity vs. Attribute (Cont.)

- **Works_In2 does not allow an employee to work in a department for two or more periods.**
- **Similar to the problem of wanting to record several addresses for an employee: we want to record *several values of the descriptive attributes for each instance of this relationship.***



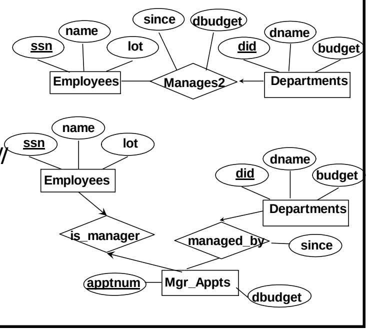


Entity vs. Relationship

OK as long as a manager gets a separate discretionary budget (*dbudget*) for each dept.

What if manager's *dbudget* covers *all* managed depts?

(can repeat value, but such redundancy is problematic)



Now you try it

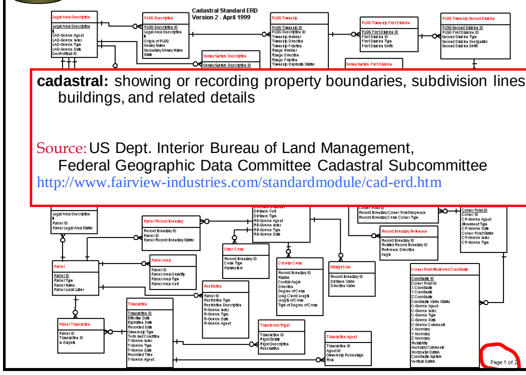
Try this at home - Courses database:

- **Courses, Students, Teachers**
- **Courses have ids, titles, credits, …**
- **Courses have multiple sections that have time/rm and exactly one teacher**
- **Must track students' course schedules and transcripts including grades, semester taken, etc.**
- **Must track which classes a professor has taught**
- **Database should work over multiple semesters**

These things get pretty hairy!

- **Many E-R diagrams cover entire walls!**
- **A modest example:**

A Cadastral E-R Diagram



cadastral: showing or recording property boundaries, subdivision lines, buildings, and related details

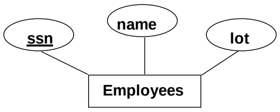
Source: US Dept. Interior Bureau of Land Management, Federal Geographic Data Committee Cadastral Subcommittee
http://www.fairview-industries.com/standardmodule/cad-erd.htm

Converting ER to Relational

- **Fairly analogous structure**
- **But many simple concepts in ER are subtle to specify in relations**

Logical DB Design: ER to Relational

- Entity sets to tables.



ssn	name	lot
123-22-3666	Attishoo	48
231-31-5368	Smiley	22
131-24-3650	Smethurst	35

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

Relationship Sets to Tables

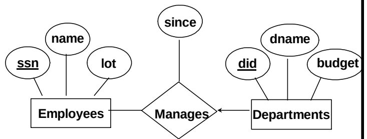
- In translating a many-to-many relationship set to a relation, attributes of the relation must include:
  1) Keys for each participating entity set (as foreign keys). This set of attributes forms a *superkey* for the relation.
  2) All descriptive attributes.

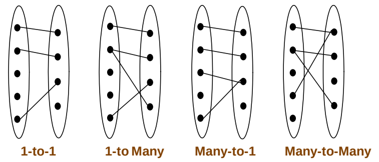
```
CREATE TABLE Works_In(
  ssn  CHAR(1),
  did  INTEGER,
  since  DATE,
 PRIMARY KEY (ssn, did),
 FOREIGN KEY (ssn)
    REFERENCES Employees,
 FOREIGN KEY (did)
  REFERENCES Departments)
```

ssn	did	since
123-22-3666	51	1/1/91
123-22-3666	56	3/3/93
231-31-5368	51	2/2/92

Review: Key Constraints

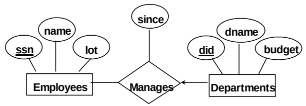
- **Each dept has at most one manager, according to the *key constraint* on Manages.**





Translation to relational model?

Translating ER with Key Constraints



- Since each department has a unique manager, we could instead combine Manages and Departments.

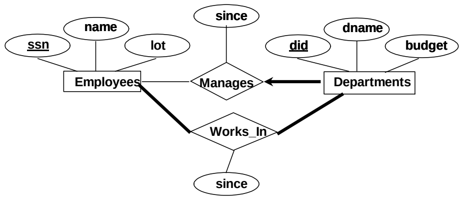
```
CREATE TABLE  Manages(
 ssn  CHAR(11),
 did  INTEGER,
 since  DATE,
 PRIMARY KEY  (did),
 FOREIGN KEY (ssn)
REFERENCES Employees,
   FOREIGN KEY (did)
REFERENCES Departments)
```

Vs.

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

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!)



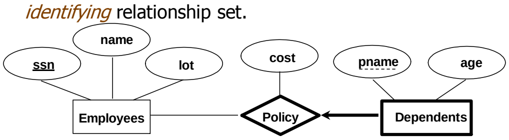
Participation Constraints in SQL

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

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

Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
  - Note: There are many variations on ER model
    - Both graphically and conceptually
- Basic constructs: *entities, relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities, ISA hierarchies* (see text if you're curious), and *aggregation*.

Summary of ER (Cont.)

- Several kinds of integrity constraints:
  - *key constraints*
  - *participation constraints*
- Some *foreign key constraints* are also implicit in the definition of a relationship set.
- Many other constraints (notably, *functional dependencies*) cannot be expressed.
- Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Cont.)

- ER design is *subjective*. There are often many ways to model a given scenario!
- Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further.
  - Functional Dependency information and normalization techniques are especially useful.