Relational Database

The dominate data model

- IBM’s DB2 family
- Oracle
- Microsoft’s Access and SQLServer
- A multi-billion dollar industry

Relational Database

- Relational database*:* a set of relations
- *Relation:* made up of 2 parts:
  - *Schema* : specifies the name of relations, 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
- Can think of a relation as a *set* of *rows* or *tuples or records* (i.e., all rows are distinct). *
  - *commercial systems allow duplicate rows, but

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 (#rows)= 3
degree (#columns)= 5
all rows are

Does the order Matter?

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

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

- Record (row): no
  - Why: set
- Columns (field): depends

Integrity Constraints (ICs)

- IC: condition that must be true for *any* instance of the database
  - ICs are specified when schemas are 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

Two Common Types of ICs

- Primary Key Constraints (key)
- Referential Integrity Constraint (Foreign key)

Primary Key

- Key: *minimal* set of the fields of a relation that can uniquely identify a tuple
  - {SSN} is a key
  - If there's >1 key for a relation, one of the keys is chosen (by DBA) to be the primary key. The other keys are called candidate keys.
- Superkey: set of the fields of a relation that can uniquely identify a tuple
  - E.g., {SSN, Name, Age} is a superkey.
  - A key must be a super key, but not vice versa

Excise

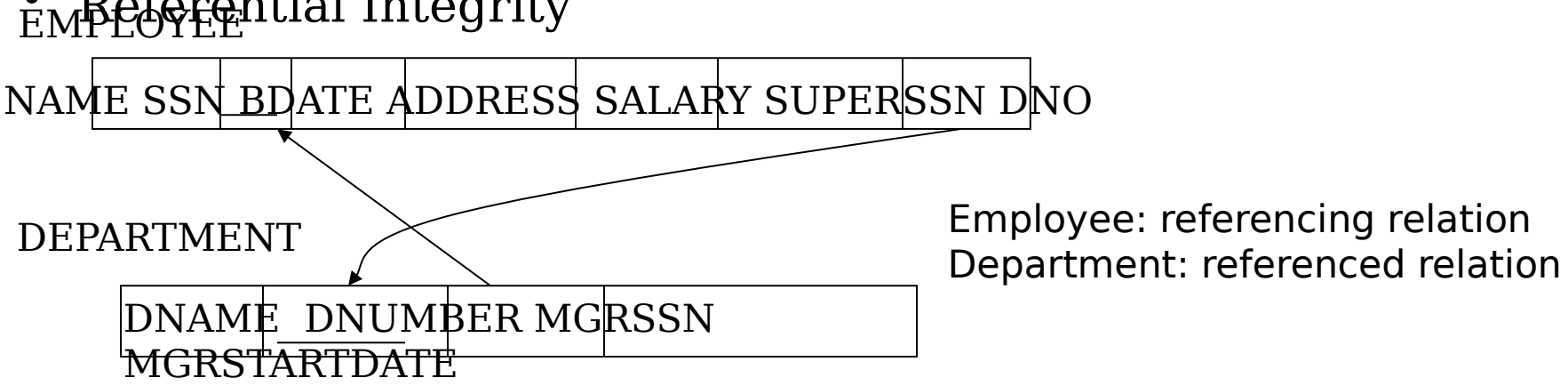
- T/F: every relation is guaranteed to have a key
  - Yes. Because relation is a set of tuples, the set of all fields is always a superkey
- Facebook. What is/are the key/keys?

Foreign Key

Foreign key (FK) : 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.)

- FK is Like a `logical pointer'.

- Referential Integrity



- DNO is a foreign key of EMPLOYEE.

The SQL Query Language

- Data Definition Language (DDL)
  - enables creation, deletion, and modification of definitions for tables and views and integrity constraint specification.
- Data Manipulation Language (DML)
  - allows users to query, to insert, or to delete rows.

Creating Relations in SQL

- Creates the Students relation. Observe that 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 REAL
primary key (sid))
```

- As another example, the Enrolled table holds information about courses that students take.

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

Primary and Candidate Keys in SQL

- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.
  - "A student can take a same course only once."

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

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

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

Foreign Keys in SQL

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

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

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

Enrolled

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

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?
  - Not allowed if it appears in Enrolled
  - Delete all Enrolled tuples that refer to it.
  - 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'*.)

Referential Integrity in SQL/92

- SQL/92 supports all 4 options on deletes and updates.
  - Default is NO ACTION (*delete/update is rejected*)
  - CASCADE (also delete all tuples that refer to the deleted tuple)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY (sid,cid),
   FOREIGN KEY (sid)
     REFERENCES Students
     ON DELETE CASCADE)
```

```
CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY (sid,cid),
   FOREIGN KEY (sid)
     REFERENCES Students
     ON DELETE SET NULL)
```

Destroying and Altering Relations

```
DROP TABLE Students
```

- Destroys the relation Students. The schema information *and* the tuples are deleted.

```
ALTER TABLE Students
    ADD firstYear integer
```

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *NULL* value in the new field.

```
ALTER TABLE Students
    drop column firstYear
```

Adding and Deleting Tuples

- Insert a single tuple:

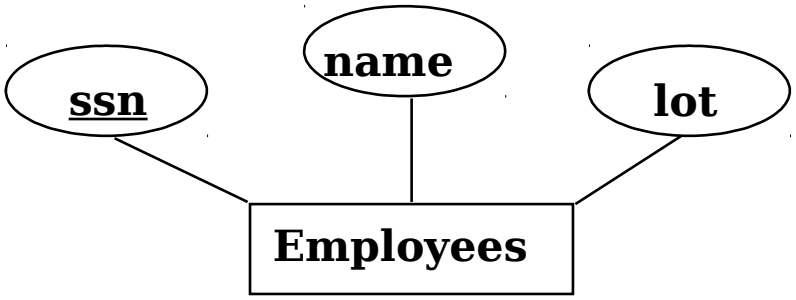
```
INSERT INTO  Students (sid, name, login, age, gpa)
VALUES  ('53688', 'Smith', 'smith@ee', 18, 3.2)
```

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

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

Logical DB Design: ER to Relational

- Entity sets to tables.

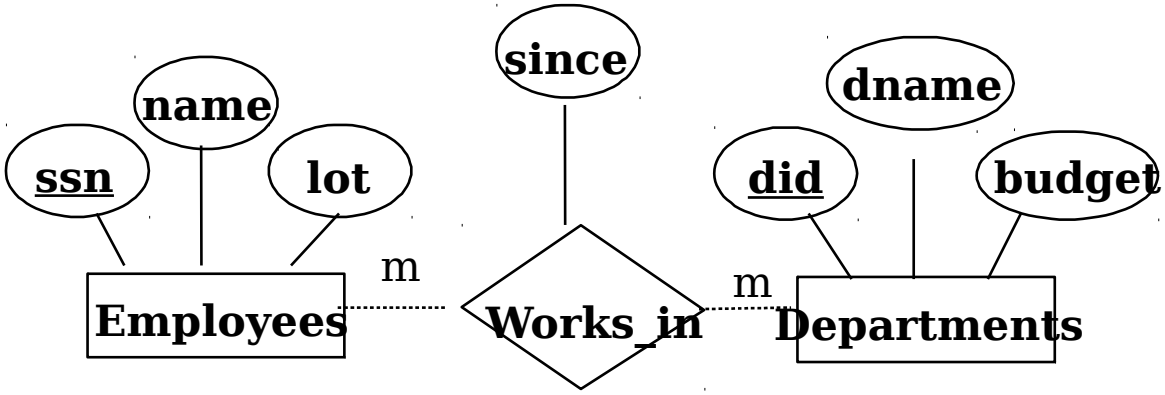



```
CREATE TABLE Employees
    (ssn CHAR(11),
    name CHAR(20),
    lot  INTEGER,
    PRIMARY KEY  (ssn))
```

For each regular entity set E in the ER diagram, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute.

Choose one of the key attributes of E as the primary key of R.
If the chosen key is composite, the simple attributes that form the key are taken together as the primary key of R.

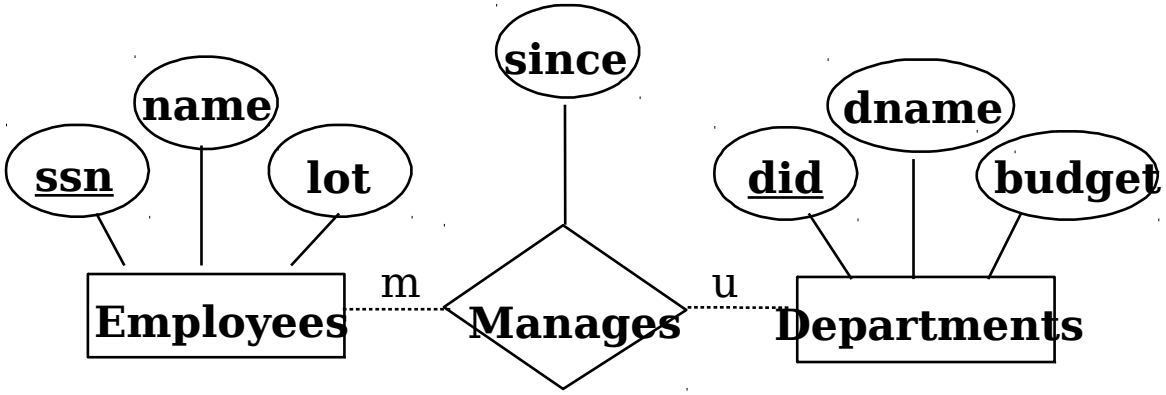
Relationship Sets to Tables



- In translating a 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 *key* for the relation.
  - All descriptive attributes.

```
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)
```

Uni-participation Constraints

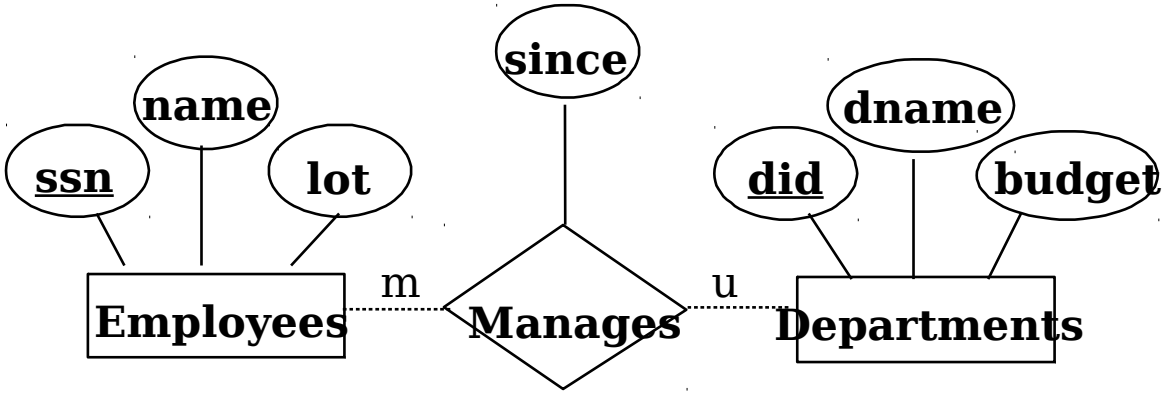


- Map relationship to a table:
  - Note that did is the key now!
  - Separate tables for Employees and Departments.

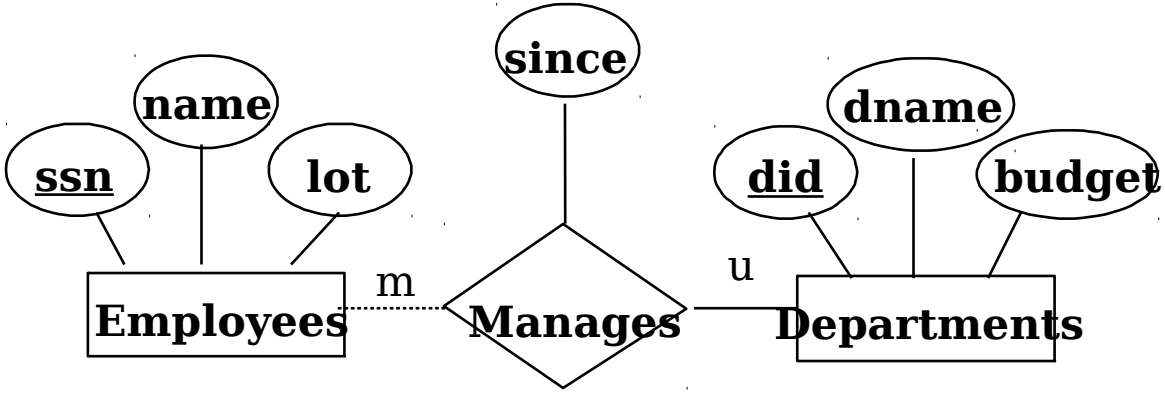
```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```

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

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



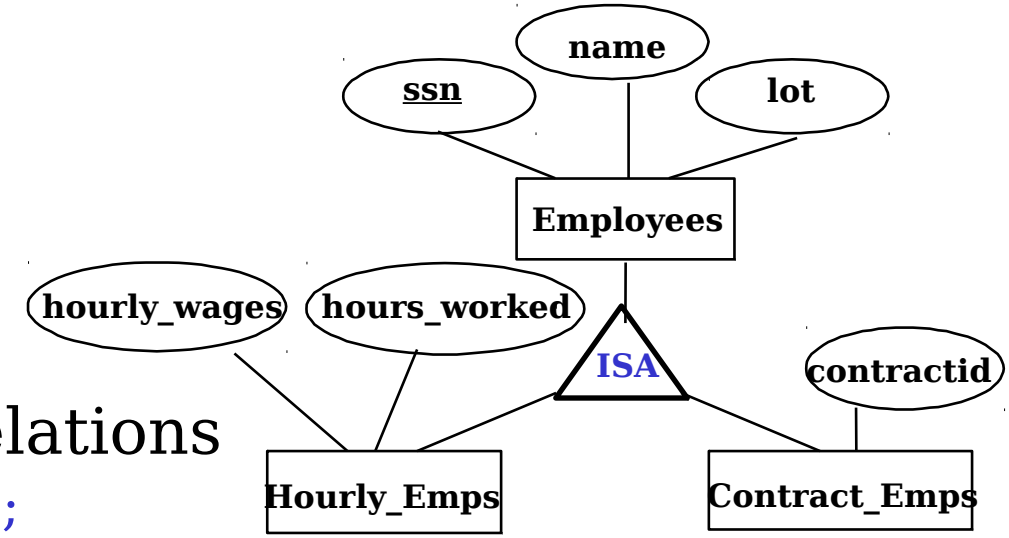
Total and Uni-Participation Constraints



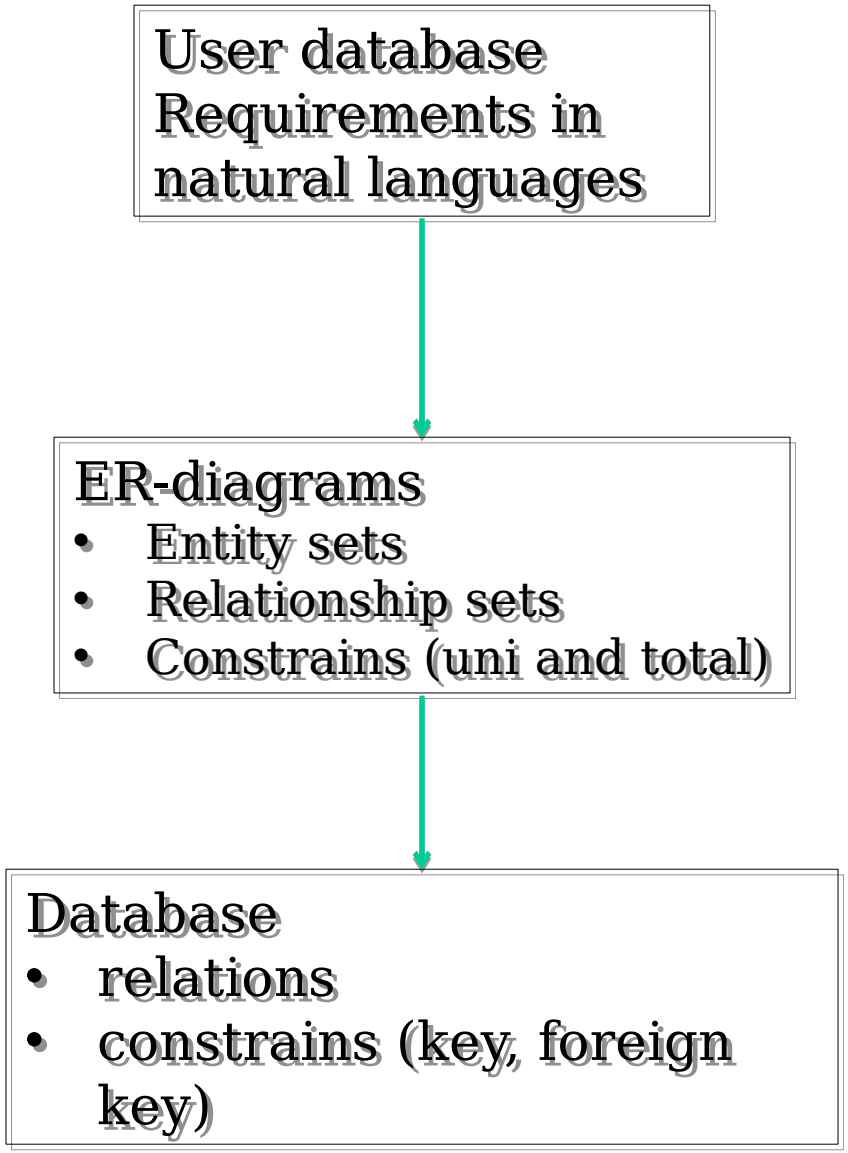

```
CREATE TABLE  Dept (
   did  INTEGER,
   dname  CHAR(20),
   budget  REAL,
   mgrssn  CHAR(11), NOT NULL
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY (ssn) REFERENCES Employees)
```

By specifying “NOT NULL”, mrgssn must have a value, thus satisfying total participation constraint

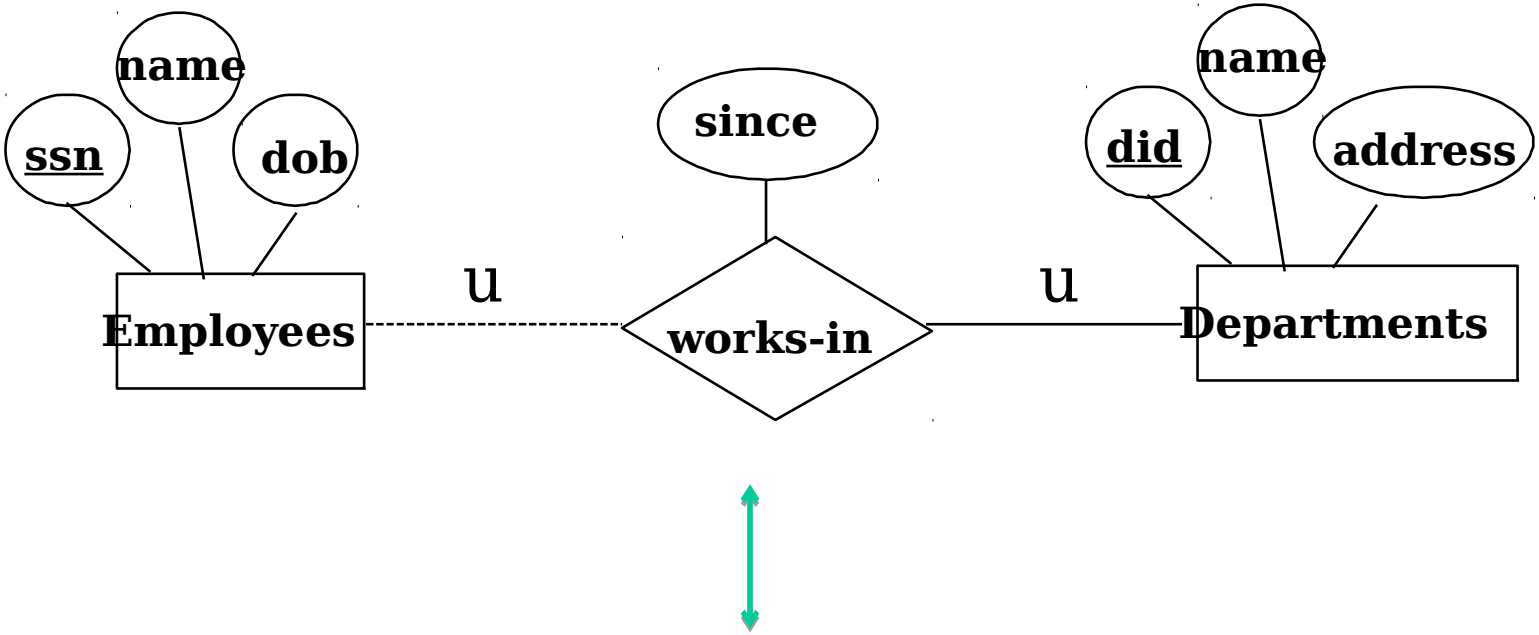
Translating ISA Hierarchies to Relations



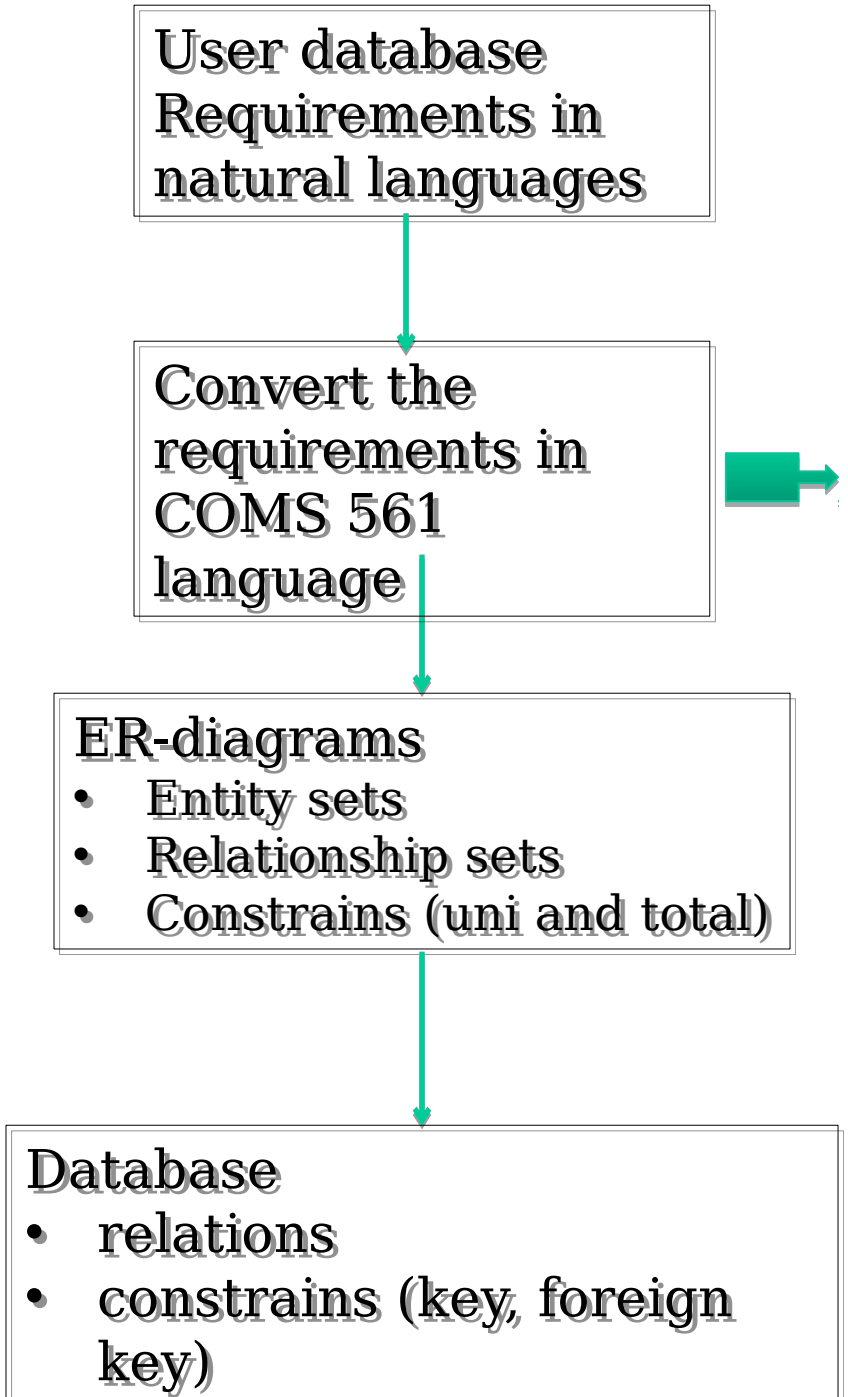
- General approach: 3 relations
  - Employee(ssn,name,lot);
  - Hourly_Emps (ssn, hourly_wages, hours_worked)
  - Contract_Emps(ssn,contractid)
    - Must delete Hourly_Emps tuple if referenced Employees tuple is deleted
    - Queries involving all employees are easy
    - Queries involving just Hourly_Emps require a join to get some attributes
- Alternative: Just Hourly_Emps and Contract_Emps
  - Hourly_Emps (ssn, name, lot, hourly_wages, hours_worked)
  - Contract_Emps (ssn,name, lot, contractid)
    - Each employee must be in one of these two



1. Which one is easier, less likely to make mistake?
   - From requirements to ER
   - From ER to database
2. Why and what can we do?



1. There are a number of Employees, each of which has a unique SSN, a name, a dob
2. There are a number of Departments, each of which has a unique did, a name, an address
3. Some employees works in some departments
   1) An employee can work in at most one department
   2) Every department must have one



Entity Sets

- There are a number of E_1, each of which has a_1, a_2, ..., and a_n1, where a_x is unique, a_y is unique, and ..., so on
- There are a number of E_2, each of which has a_1, a_2, ..., a_n2, where attr1_x is unique
- .....

Relationship Sets

- E_x has to do with E_y
- .....

Constraints

- Each one in E_x can do with someone in E_y, at most one time, and/or at least one time
- .....