# Database Management Systems INFO 210

## Entity Relationships Lecture 3

**Franz Wotawa**
TU Graz, Institut for Software Technologie
Inffeldgasse 16b/2
wotawa@ist.tugraz.at

## Today…

- Last Session:
  - Intro do databases
  - Conceptual Modeling
- Today's Session:
  - Introduction to databases and database systems (*recap*)
  - Main steps involved in designing databases
  - Constructs of the entity relationship (ER) model
  - Integrity constrains that can be expressed in the ER model

# Outline

- A Primer on Databases ✓
- Database Design
- ER Model: Constructs and Constraints

# Some Definitions

- A database is a collection of data which describes one or many real-world enterprises
  - E.g., a university database might contain information about entities like students and courses, and relationships like a student enrollment in a course
- A DBMS is a software package designed to store and manage databases
  - E.g., DB2, Oracle, MS SQL Server, MySQL and **Postgres**
- A database system = (Big) Data + DBMS + Application Programs

# Data Models

- The user of a DBMS is ultimately concerned with some real-world enterprises (e.g., a University)
- The data to be stored and managed by a DBMS *describes* various aspects of the enterprises
  - E.g., The data in a university database describes students, faculty and courses entities and the relationships among them
- A data model is a collection of high-level data description constructs that hide many low-level storage details
- A widely used data model called the entity-relationship (ER) model allows users to pictorially denote entities and the relationships among them

# The Relational Model

- The relational model of data is one of the most widely used models today
- The central data description construct in the relational model is the relation
- A relation is basically a table (or a set) with rows (or records or tuples) and columns (or fields or attributes)
- Every relation has a schema, which describes the columns of a relation
- Conditions that records in a relation must satisfy can be specified
  - These are referred to as integrity constraints

# The Relational Model: An Example

- Let us consider the student entity in a university database

Students Schema

**Students(*sid*: string, *name*: string, *login*: string, *dob*: string, *gpa*: real)**

An attribute, field or column

Integrity Constraint: Every student has a unique *sid* value

A record, tuple or row

| *sid* | *name* | *login* | *dob* | *gpa* |
|---|---|---|---|---|
| 512412 | Khaled | khaled@lzu.cn | 18-9-1995 | 3.5 |
| 512311 | Jones | jones@lzu.cn | 1-12-1994 | 3.2 |
| 512111 | Maria | maria@lzu.cn | 3-8-1995 | 3.85 |

An *instance* of a Students *relation*

# Queries in a DBMS

- The "ease" (*plus* efficiency & effectiveness) with which information can be queried from a database determines its value to users

- A DBMS provides a specialized language, called the query language, in which queries can be posed

- The relational model supports powerful query languages
  - Relational calculus: a formal language based on mathematical logic
  - Relational algebra: a formal language based on a collection of operators (e.g., selection and projection) for manipulating relations
  - Structured Query Language (SQL):
    - Builds upon relational calculus and algebra
    - Allows creating, manipulating and querying relational databases
    - Can be embedded within a host language (e.g., Java)

# Concurrent Execution and Transactions

- An important task of a DBMS is to *schedule* concurrent accesses to data so as to improve performance

| T1 | T2 |
|---|---|
| | R(A) |
| | W(A) |
| R(B) | |
| W(B) | |
| | R(C) |
| | W(C) |

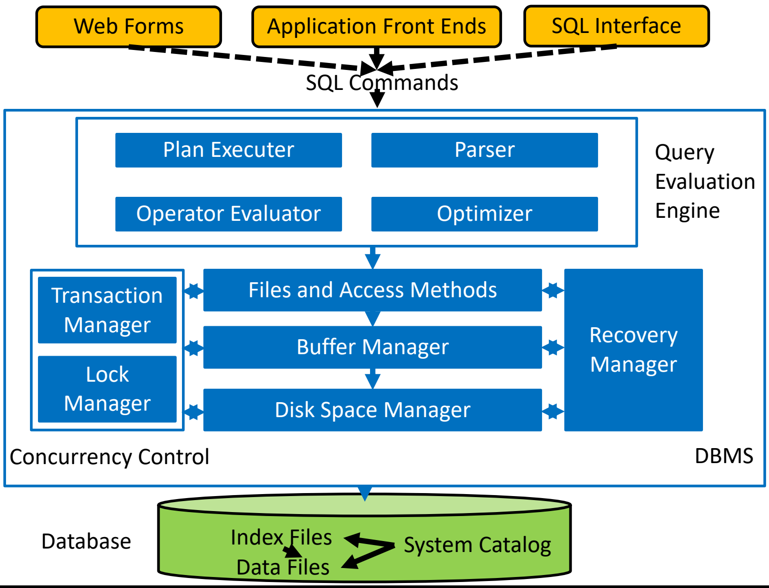
An *atomic* sequence of database actions (read/writes) is referred to as *"transaction"*

- When several users access a database *concurrently*, the DBMS must order their requests carefully to avoid conflicts
  - E.g., A check might be cleared while account balance is being computed!
- DBMS ensures that conflicts do not arise via using a locking protocol
  - Shared vs. Exclusive locks

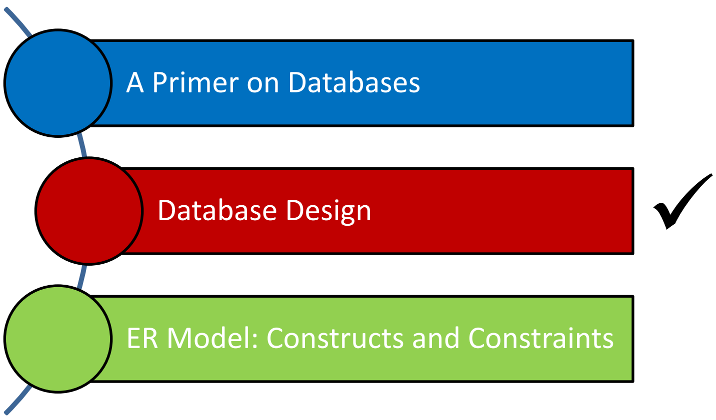
# Ensuring Atomicity

- Transactions can be interrupted before running to completion for a variety of reasons (e.g., due to a system crash)
- DBMS ensures atomicity (all-or-nothing property) even if a crash occurs in the middle of a transaction
- This is achieved via maintaining a log (i.e., history) of all writes to the database
  - *Before* a change is made to the database, the corresponding log entry is forced to a safe location (this protocol is called Write-Ahead Log or WAL)
  - After a crash, the effects of partially executed transactions are *undone* using the log

# The Architecture of a Relational DBMS




# Outline



# Database Design

- Requirements Analysis
  - Users needs
- Conceptual Design
  - A high-level description of the data (e.g., using the ER model)
- Logical Design
  - The conversion of an ER design into a relational database schema
- Schema Refinement
  - Normalization (i.e., restructuring tables to ensure some desirable properties)
- Physical Design
  - Building indexes and clustering some tables
- Security Design
  - Access controls

# Outline

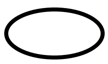
- A Primer on Databases
- Database Design
- ER Model: Constructs and Constraints

# Entities and Entity Sets

- Entity:
  - A real-world object distinguishable from other objects in an enterprise (e.g., University, Students and Faculty)
  - Described 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 (until we consider ISA hierarchies, anyway!)
  - Each entity set has a *key*
  - Each attribute has a *domain*

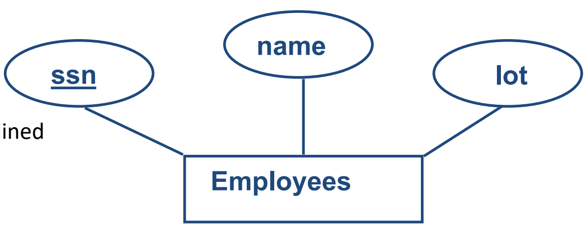
# Tools and An ER Diagram

**Entities ('Entity Sets')**



**Attributes**

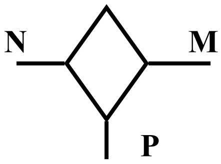
"ssn" is the *primary key*, hence, underlined



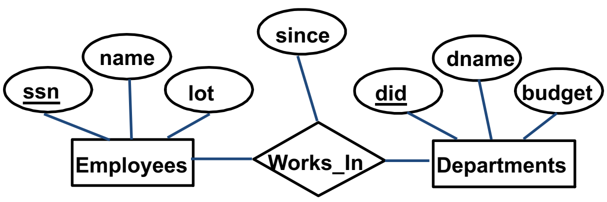
# Relationship and Relationship Sets

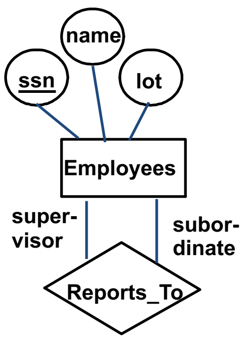
- Relationship:
  - Association among two or more entities (e.g., Rui is teaching INFO210)
  - Described using a set of attributes
- Relationship set:
  - Collection of similar relationships
  - Same entity set could participate in different relationship sets, or in different "roles" in the same set

# More Tools and ER Diagrams




**Relationships ('rel. sets') and mapping constraints**



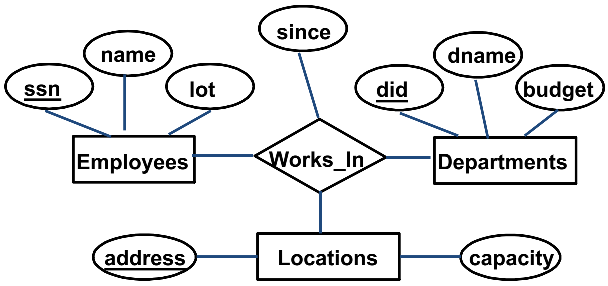




A Binary Relationship

A Self-Relationship

# Ternary Relationships

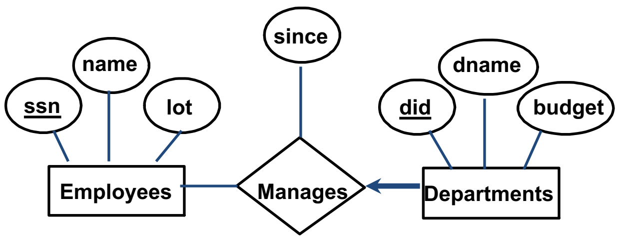
- Suppose that departments have offices at different locations and we want to record the locations at which each employee works
- Consequently, we must record an association between an employee, a department and a location




This is referred to as a “Ternary Relationship” (vs. Self & Binary Relationships)

# Key Constraints

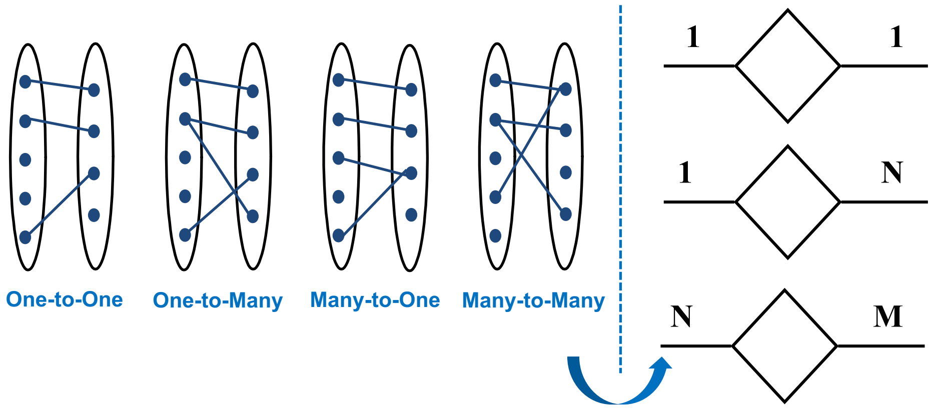
- Consider the “Employees” and “Departments” entity sets with a “Manages” relationship set
  - An employee can work in *many* departments
  - A department can have *many* employees
  - This restriction is an example of a “key constraint”



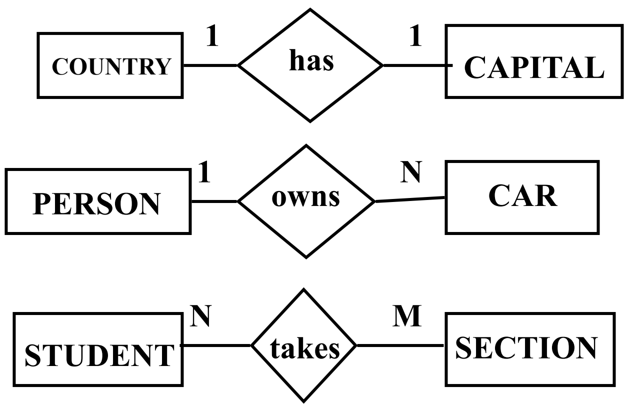

Key constraints are denoted by *thin arrows*

# Cardinalities

- Entities can be related to one another as “one-to-one”, “one-to-many” and “many-to-many”
  - This is said to be the cardinality of a given entity in relation to another

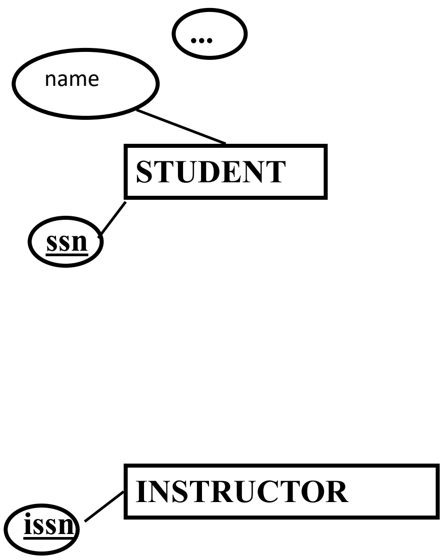



# Cardinalities: Examples

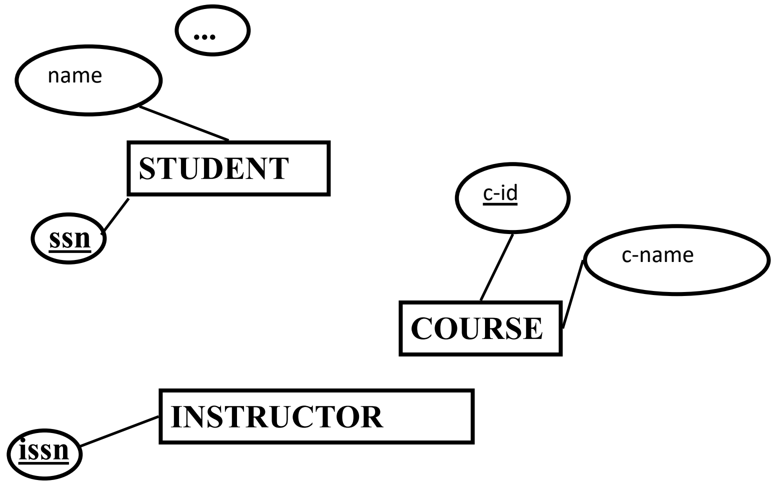


# A Working Example

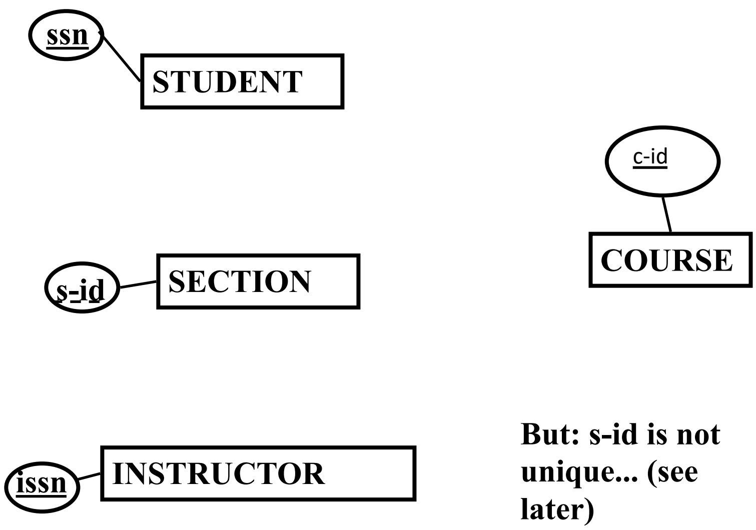
- Requirements: Students take courses offered by instructors; a course may have multiple sections; one instructor per section
- How to start?
  - Nouns -> entity sets
  - Verbs -> relationship sets



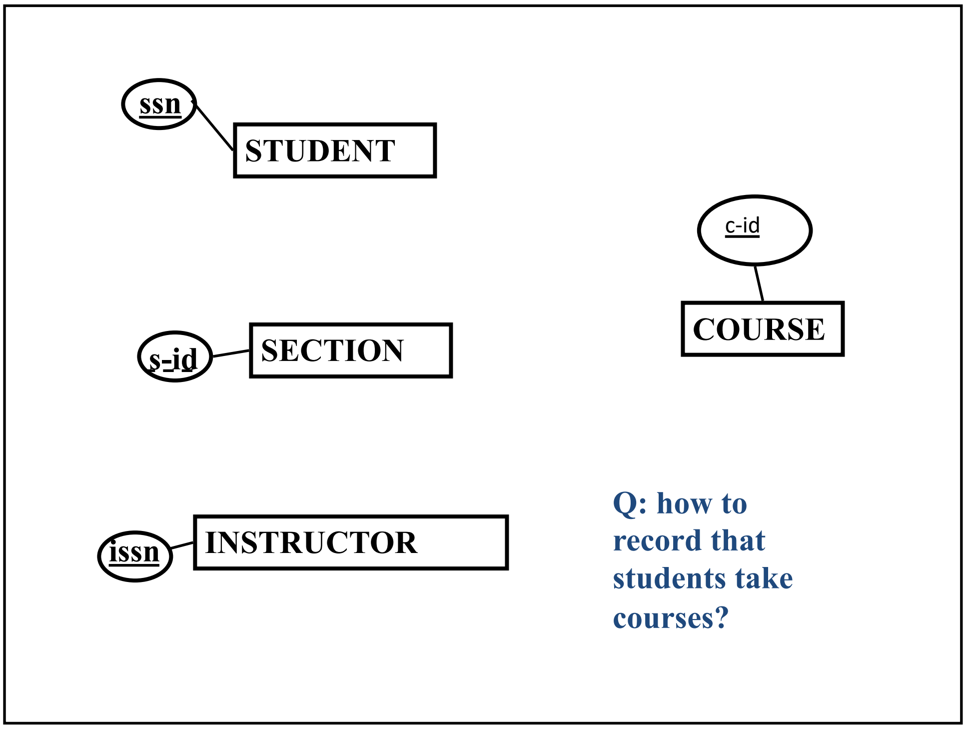

**Primary key = unique identifier → <u>underline</u>**

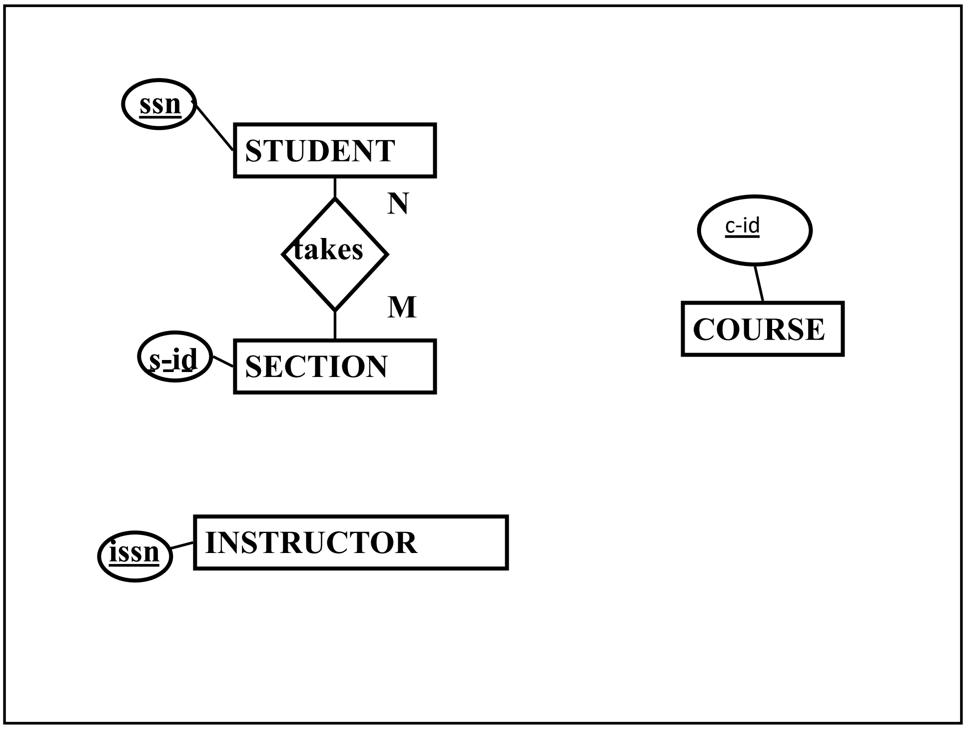


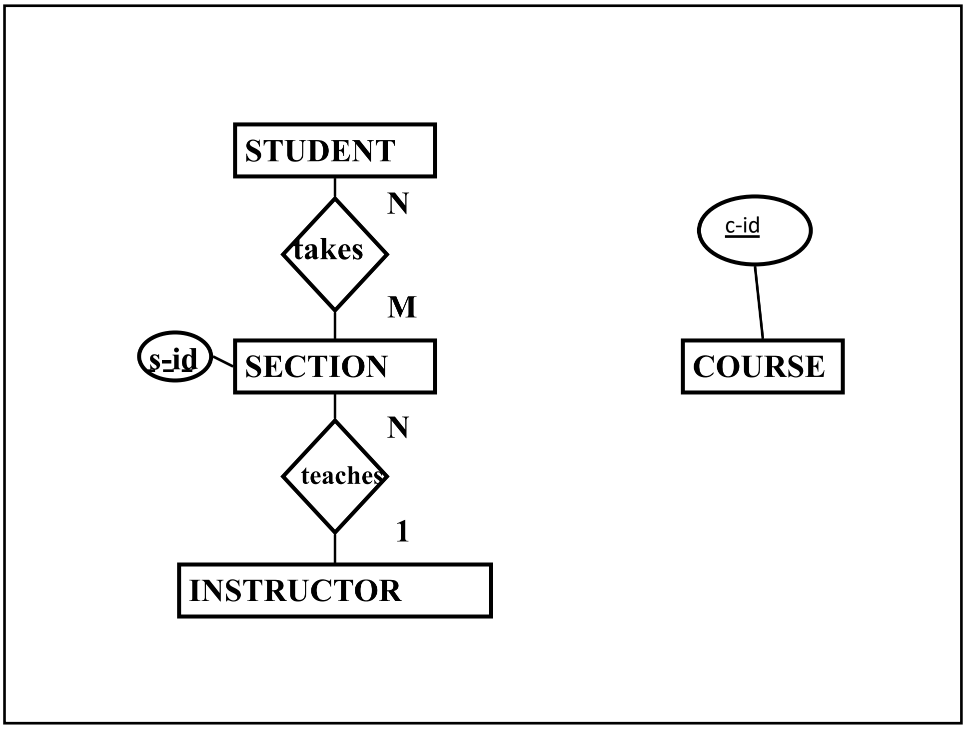

**But: sections of a course (with different instructors)?**

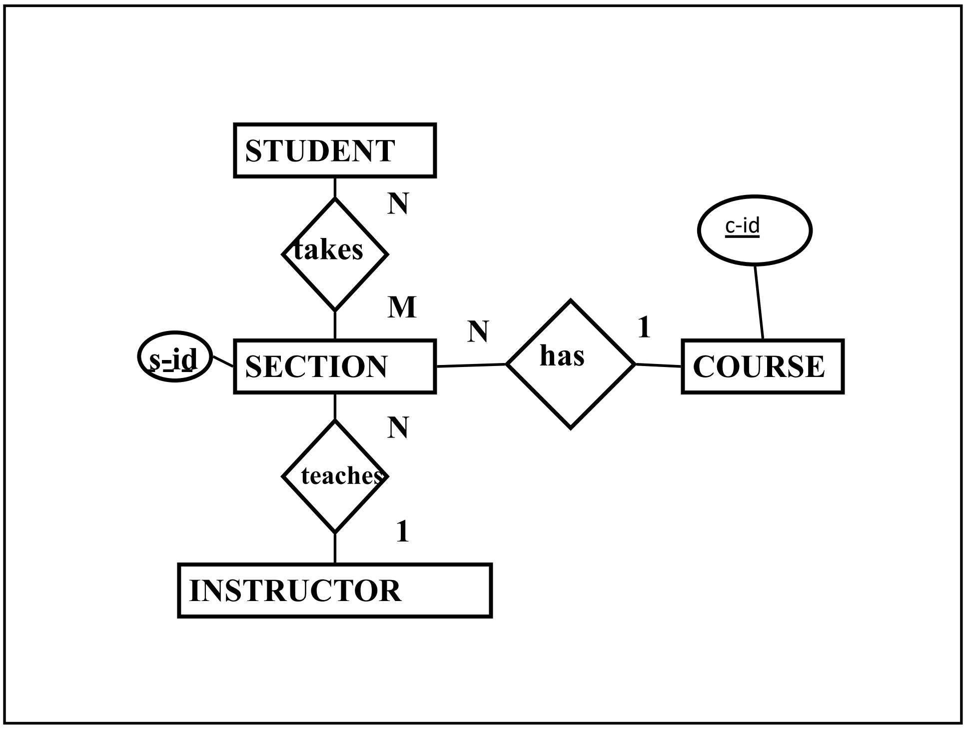



**But: s-id is not unique... (see later)**


ssn
STUDENT
c-id
COURSE
s-id
SECTION
issn
INSTRUCTOR
Q: how to record that students take courses?

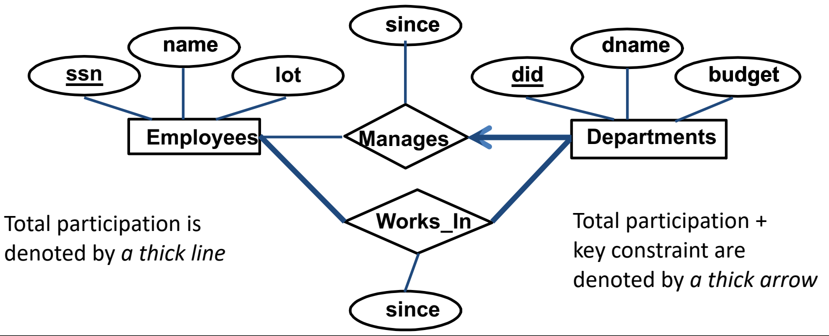

ssn
STUDENT
N
takes
M
c-id
COURSE
s-id
SECTION
issn
INSTRUCTOR


STUDENT
N
takes
M
s-id
SECTION
N
teaches
1
INSTRUCTOR
c-id
COURSE

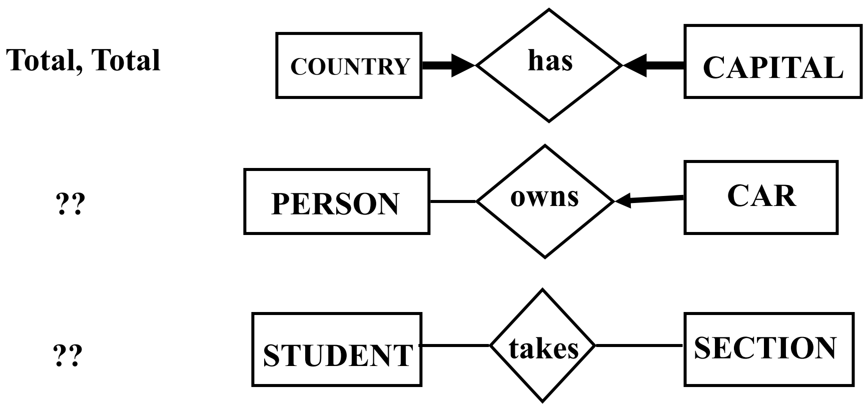

STUDENT
N
takes
M
s-id
SECTION
N
has
1
c-id
COURSE
N
teaches
1
INSTRUCTOR

# Participation Constraints

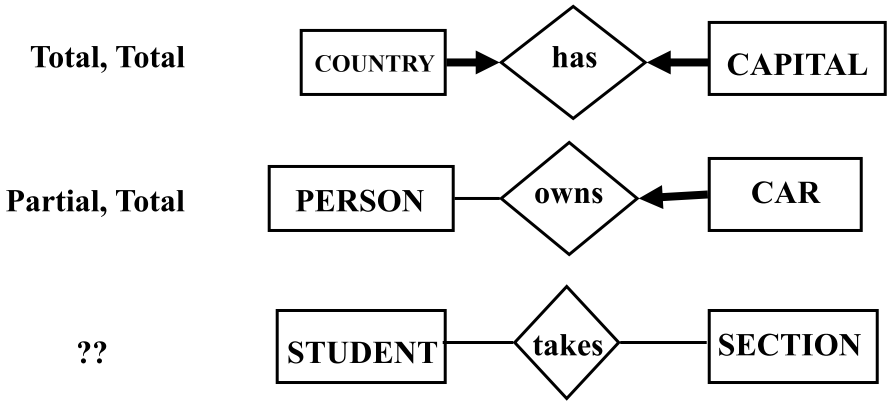
- Consider again the “Employees” and “Departments” entity sets as well as the “Manages” relationship set
  - Should every department have a manager?
  - If so, this is a participation constraint
  - Such a constraint entails that every Departments entity must appear in an instance of the Manages relationship
  - The participation of Departments in Manages is said to be total (vs. partial)



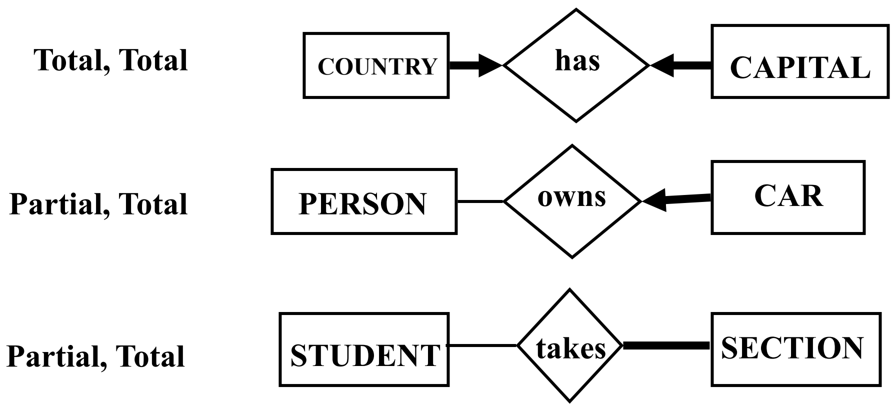

# Total vs. Partial Participations



## Total vs. Partial Participations




## Total vs. Partial Participations



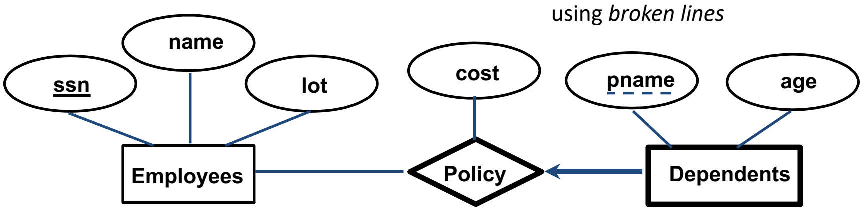
# Weak Entities

- 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)
  - Weak entity set must have total participation in this identifying relationship set
- The set of attributes of a weak entity set that uniquely identifies a weak entity for a given owner entity is called partial key

# Weak Entities: An Example

- "Dependents" has no unique key of its own
  - "Dependents" is a weak entity with partial key "pname"
  - "Policy" is an identifying relationship set
  - "pname" + "ssn" are the primary key of "Dependents"

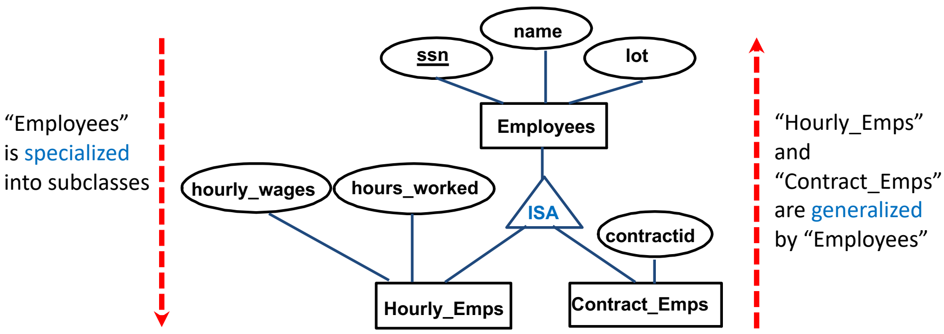
Partial keys are underlined using *broken lines*




Weak entities and identifying relationships are drawn using *thick lines*

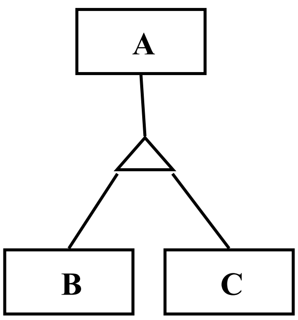
# ISA (`is a') Hierarchies

- Entities in an entity set can sometimes be classified into subclasses (this is "kind of similar" to OOP languages)
- If we declare B **ISA** A, every B entity is also considered to be an A entity



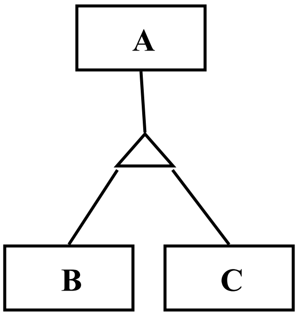

# Overlap and Covering Constraints

- Overlap constraints
  - Can an entity belong to both 'B' and 'C'?
- Covering constraints
  - Can an 'A' entity belong to neither 'B' nor 'C'?



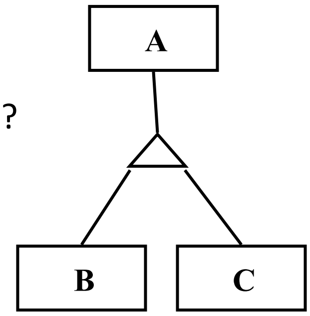
# Overlap Constraints: Examples

- Overlap constraints
  - Can John be in Hourly_Emps and Contract_Emps? Intuitively, *no*
  - Can John be in Contract_Emps and in Senior_Emps? Intuitively, *yes* ➔ "Contract_Emps OVERLAPS Senior_Emps"




# Covering Constraints: Examples

- Covering constraints
  - Does every one in Employees belong to a one of its subclasses? Intuitively, *no*
  - Does every Motor_Vehicles entity have to be either a Motorboats entity or a Cars entity? Intuitively, *yes* ➔ "Motorboats AND Cars COVER Motor_Vehicles"

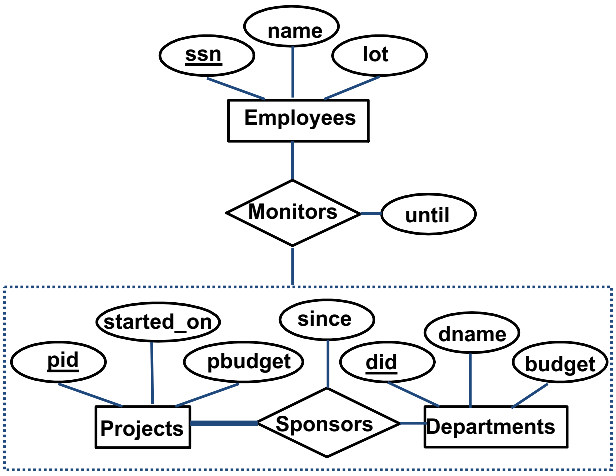


# More Details on ISA Hierarchies

- Attributes are *inherited* (i.e., if B ISA A, the attributes defined for a B entity are the attributes for A *plus* B)
- We can have ***many*** levels of an ISA hierarchy
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass
  - To identify entities that participate in a relationship

# Aggregation

- Aggregation allows indicating that a relationship set (identified through a *dashed box*) participates in another relationship set



# Next Class

Continue the ER Model and
Start with the relational Model