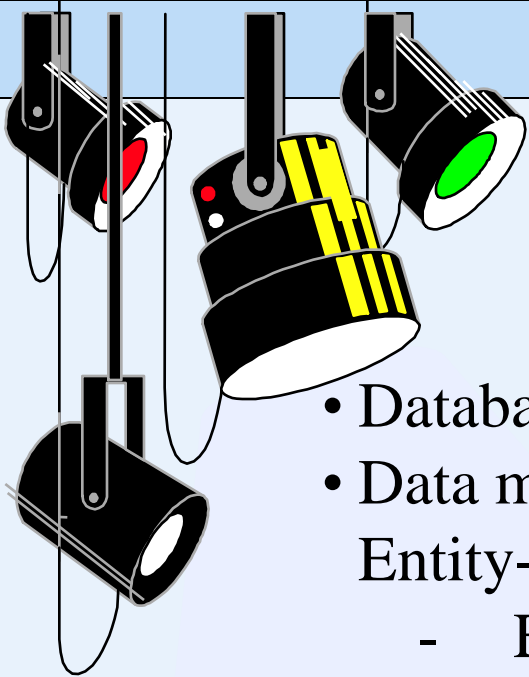
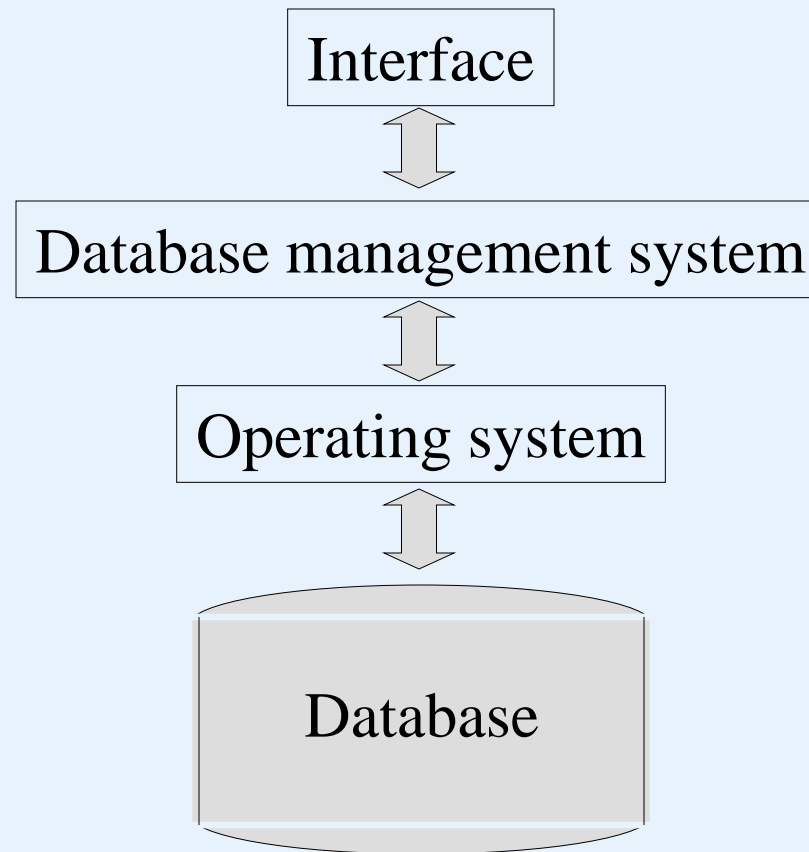


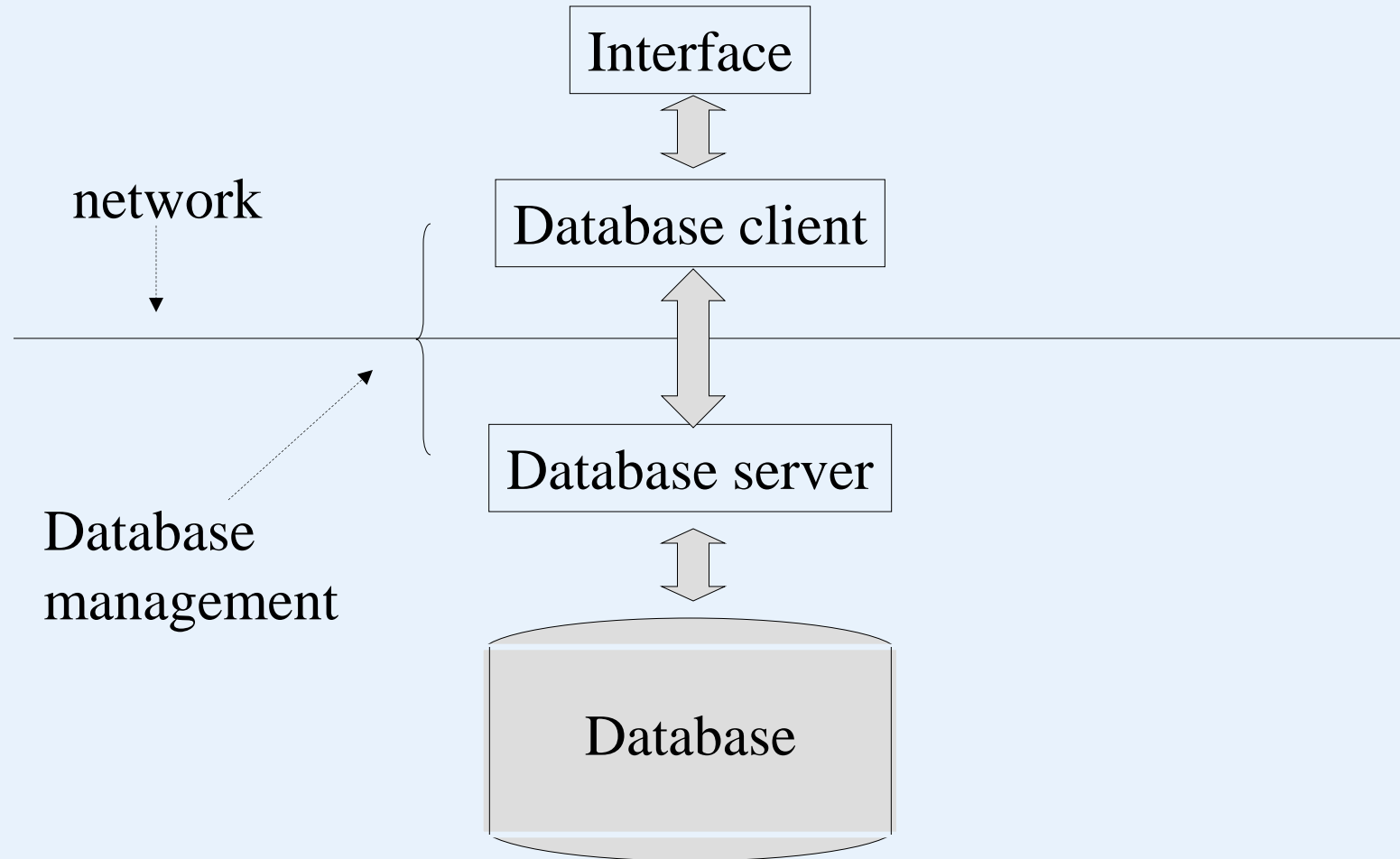
Outline: Database Basics

- 
- Database system architecture
 - Data modeling
 - Entity-relationship model
 - Entity types
 - strong entities
 - weak entities
 - Relationships among entities
 - Attributes - attribute classification
 - Constraints
 - cardinality constraints
 - participation constraints
 - ER-to-Relation-mapping

- **Database system architecture**



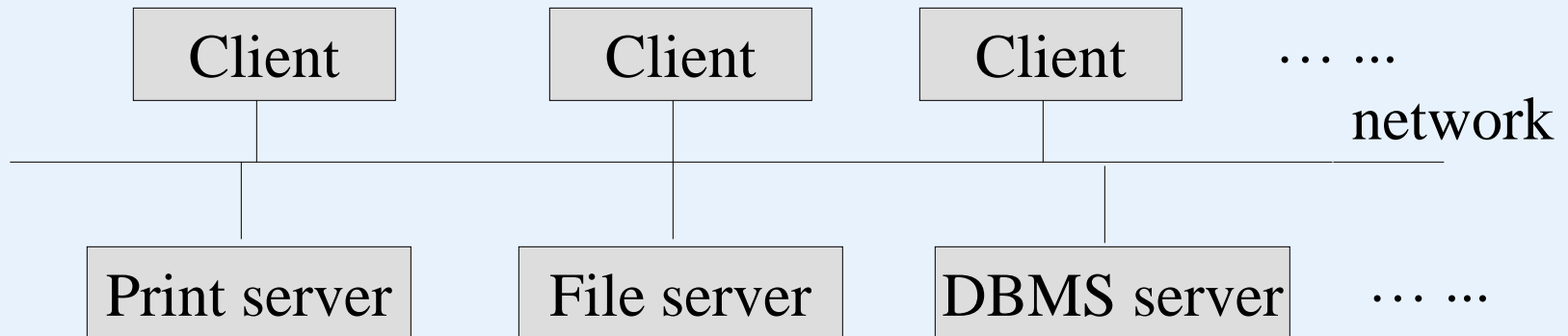
- Client-server database system Architecture



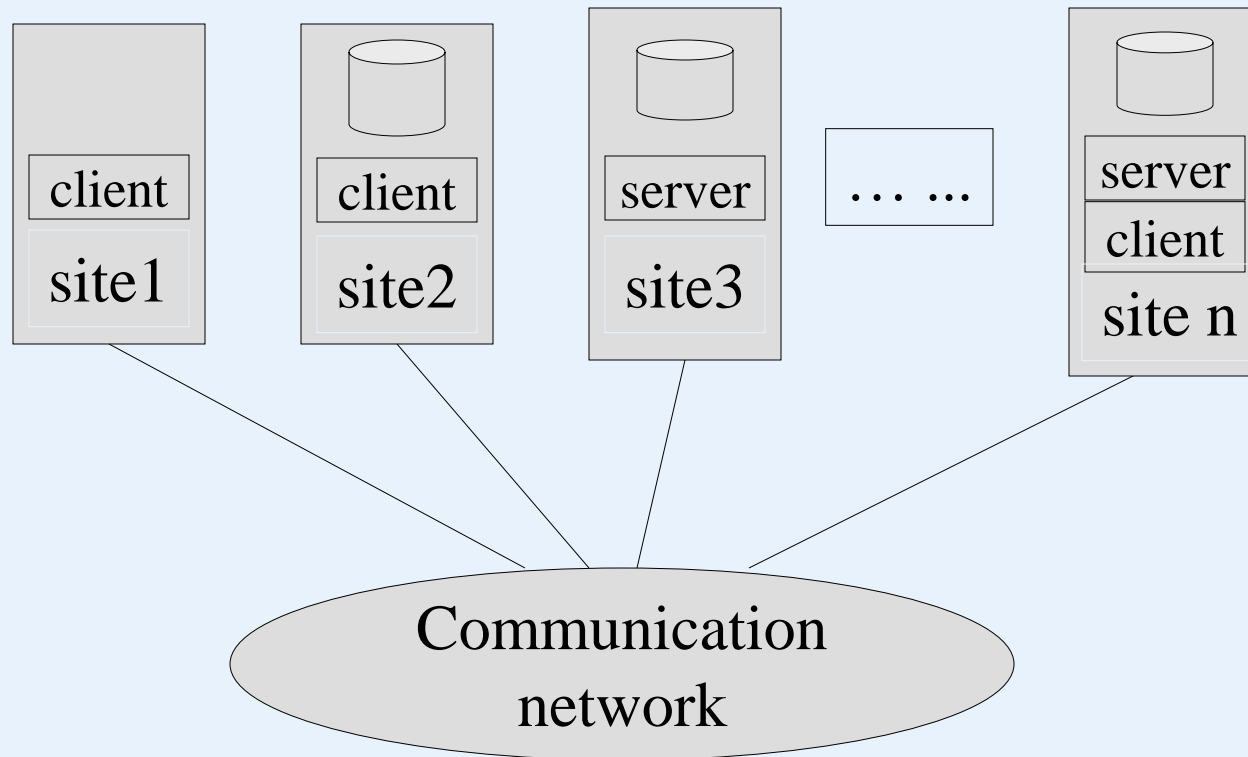
- Client-Server Computer Architecture

- Terminals are replaced with PCs and workstations
- Mainframe computer is replaced with specialized servers (with specific functionalities).

File server, DBMS server, mail server, print server, ...



- **Client-server database System Architectures**



- Client-Server database system architecture
 - database client
 - user interface, application programs
 - database server
 - SQL language, transaction management
 - database connection
 - ODBC - open database connectivity
 - API - application programming interface

- Client-server database system architecture

- database client

user interface, data dictionary functions, DBMS
interaction with programming language compiler, global
query optimization, structuring of complex objects from
the data in the buffers, ...

- database server

data storage on disk, index mechanism, local
concurrency control and recovery, buffering and caching
of disk storage, ...

- Data dictionary – system catalog (meta data)
 - relation names, attribute names, attribute domains (data types)
 - description of constraints
 - primary keys, secondary keys, foreign keys, NULL/NON-NULL, cardinality constraints, participation constraints, ...
 - views, storage structure, indexes
 - security, authorization, owner of each relation

Database Basics

- Catalog is stored as relations. (It can then be queried, updated and managed using DBMS software - SQL.)

REL_AND_ATTR_CATALOG

REL_NAME	ATTR_NAME	ATTR_TYPE	MEMBER_OF_PK	MEMBER_OF_FK	FK_RELATION
EMPLOYEE	FNAME	VSTR15	no	no	
...	...				
EMPLOYEE	SUPERSSN	STR9	no	yes	EMPLOYEE
EMPLOYEE	DNO	INTEGER	no	yes	DEPARTMENT
...	...				

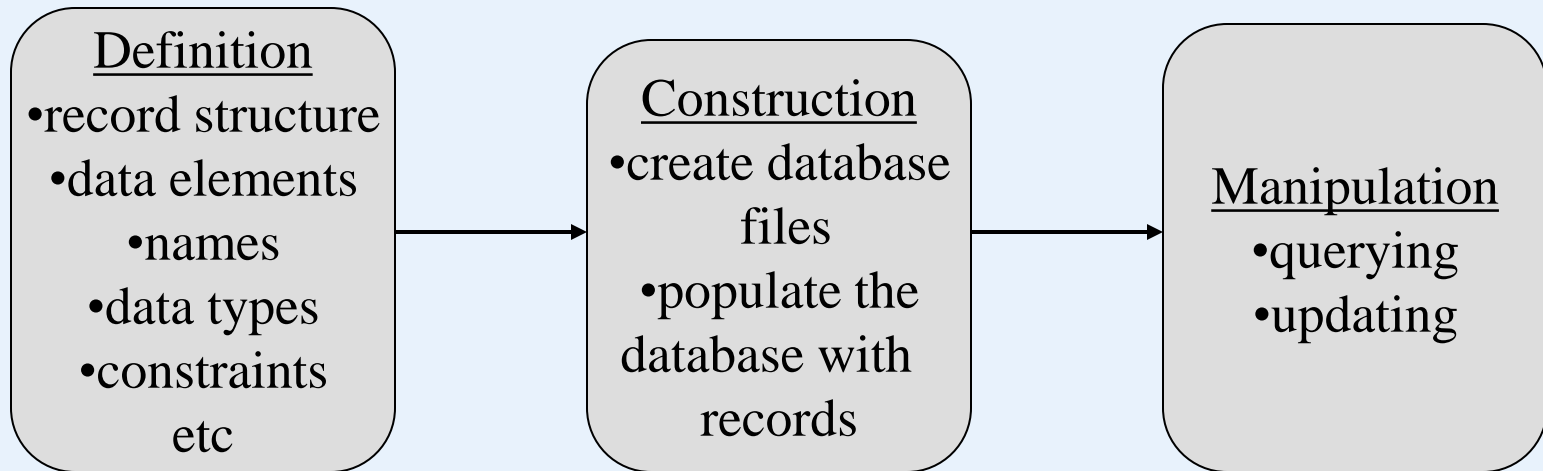
Employee relation schema:

FNAME	LNAME	SSN	SUPERSSN	DNO	...
--------------	--------------	------------	-----------------	------------	-----

Illustration for DBMS interaction with programming language compiler:

```
EXEC SQL DECLARE C1 CURSOR FOR
SELECT au_fname, au_lname FROM authors FOR BROWSE;
EXEC SQL OPEN C1;
while (SQLCODE == 0)
{
    EXEC SQL FETCH C1 INTO :fname, :lname;
}
```

- Working process with DBMS



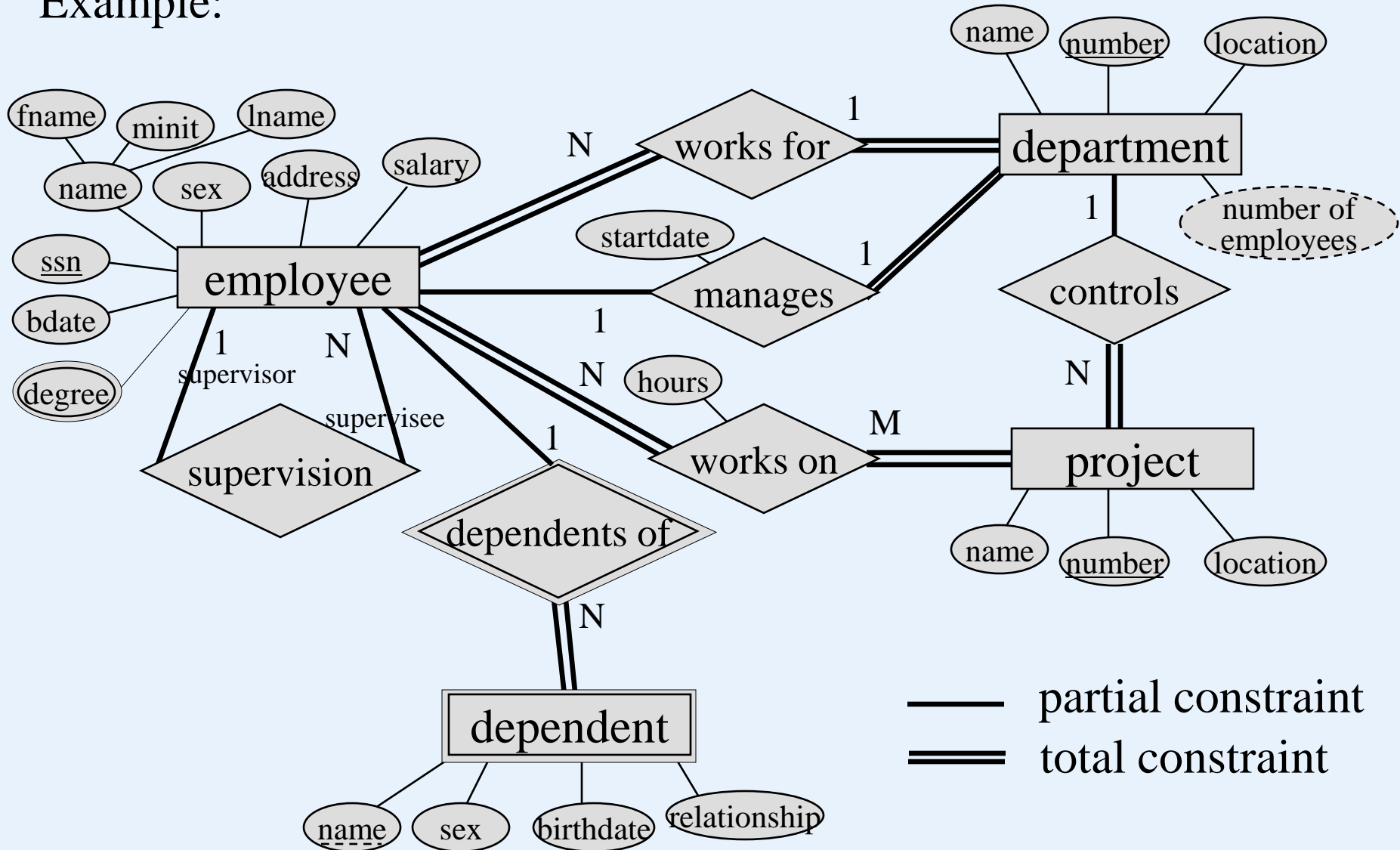
- **Entity-relationship model (ER model)**

ER model:

- is used to create a conceptual data model that reflects all the user data requirements.
- It includes detailed descriptions of
 - entity types,
 - relationships, and
 - constraints
- no implementation details. So it can be used for communication with non-technical users

Database Basics

Example:



Entities

- entity type – logical object (concept), physical object
- strong entity
 - key attribute – uniquely identifies an individual entity
 - entity has a key attribute or a combination of attributes which can be used as a key.
- weak entity

No key attributes. Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with some of their attribute values.

- identifying owner
- identifying relationship
- partial key

The entities:

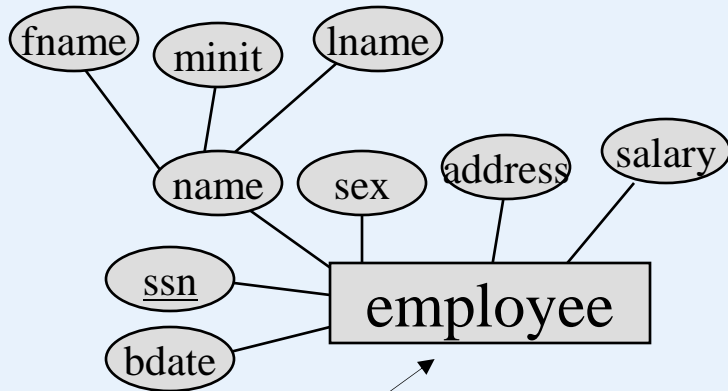
employee

dependent

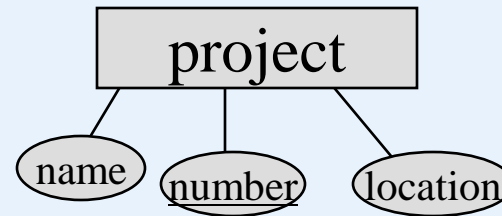
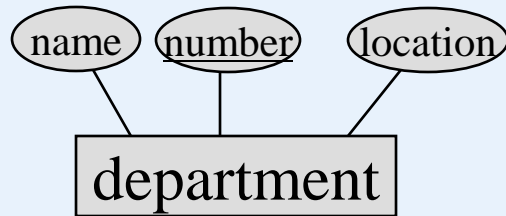
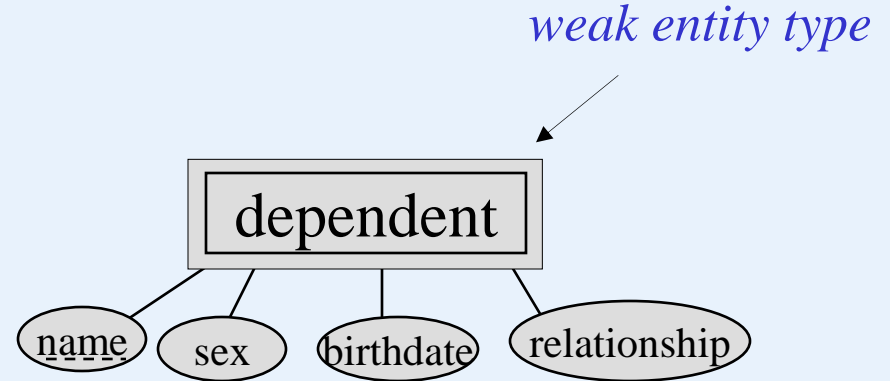
department

project

The entities:



strong entity type



Attribute – property of an entity type

Attribute classification

- atomic attribute
- multivalued attribute
- composite attribute
- complex (nested) attributes
- null values
- not applicable, unknown, missing
- key attribute
- Domain

Attribute storage

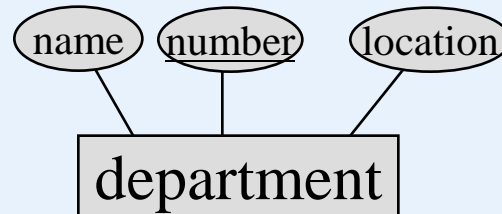
- stored & derived attribute

From a domain, an attribute takes its values.

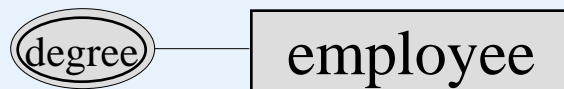
data type

Attribute classification

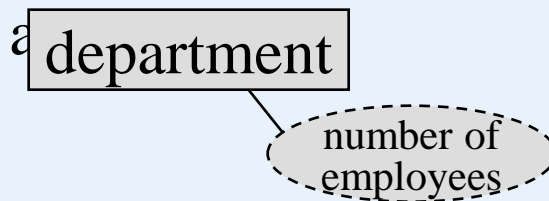
- atomic attribute



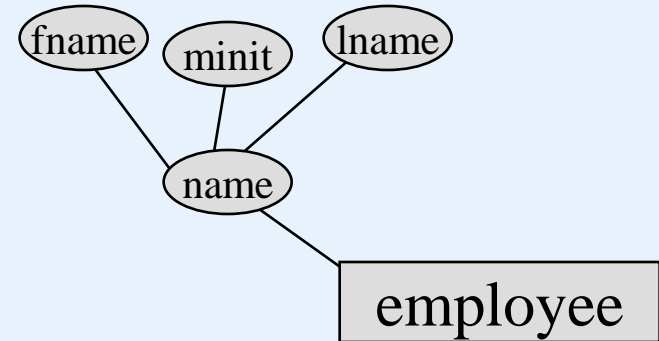
- Multivalued attribute



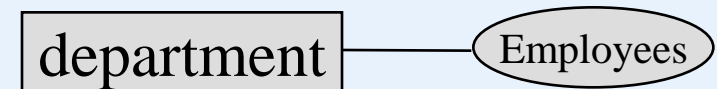
- stored & derived



- composite attribute



- Complex attribute



not often used in practice

- **Relationships**

- degree of a relationship
- recursive relationship
- role names
- constraints

cardinality: $1:1$, $1:n$, $m:n$

participation (existence dependency) :

partial – all the entities take part in a relationship

total – all the entities take part in a relationship

Example

The company database keeps track of a company's employees, departments, and projects:

Requirements:

concerning the department:

1. company is organized into departments
2. a department has a unique name, a unique number, and a specific employee is its' manager
3. we track the start date for the manager function
4. a department may be in several locations
5. a department controls a number of projects

concerning the project:

6. a project has a unique name, a unique number, and is in a single location

example continued

concerning the employee:

7. each employee has a name, social insurance number, address, salary, sex, and birth date
8. an employee is assigned to one department but may work on several projects which are not necessarily controlled by the same department
9. we track the number of hours per week that an employee works on each project
10. we keep track of the direct supervisor of each employee
11. we track the dependents of each employee (for insurance purposes)

concerning the dependent:

12. we record each dependent's first name, sex, birth date, and relationship to the employee

The entities:

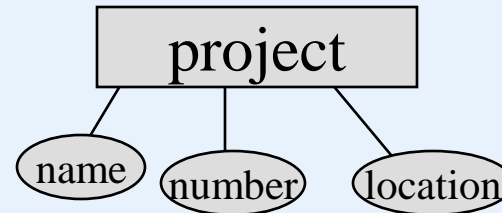
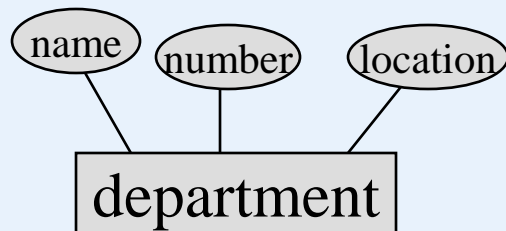
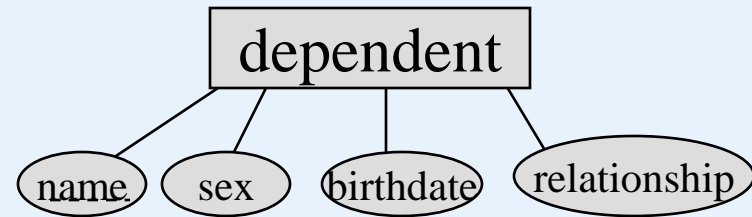
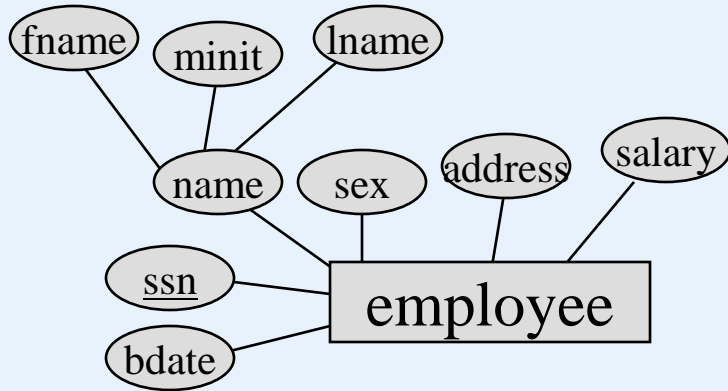
employee

dependent

department

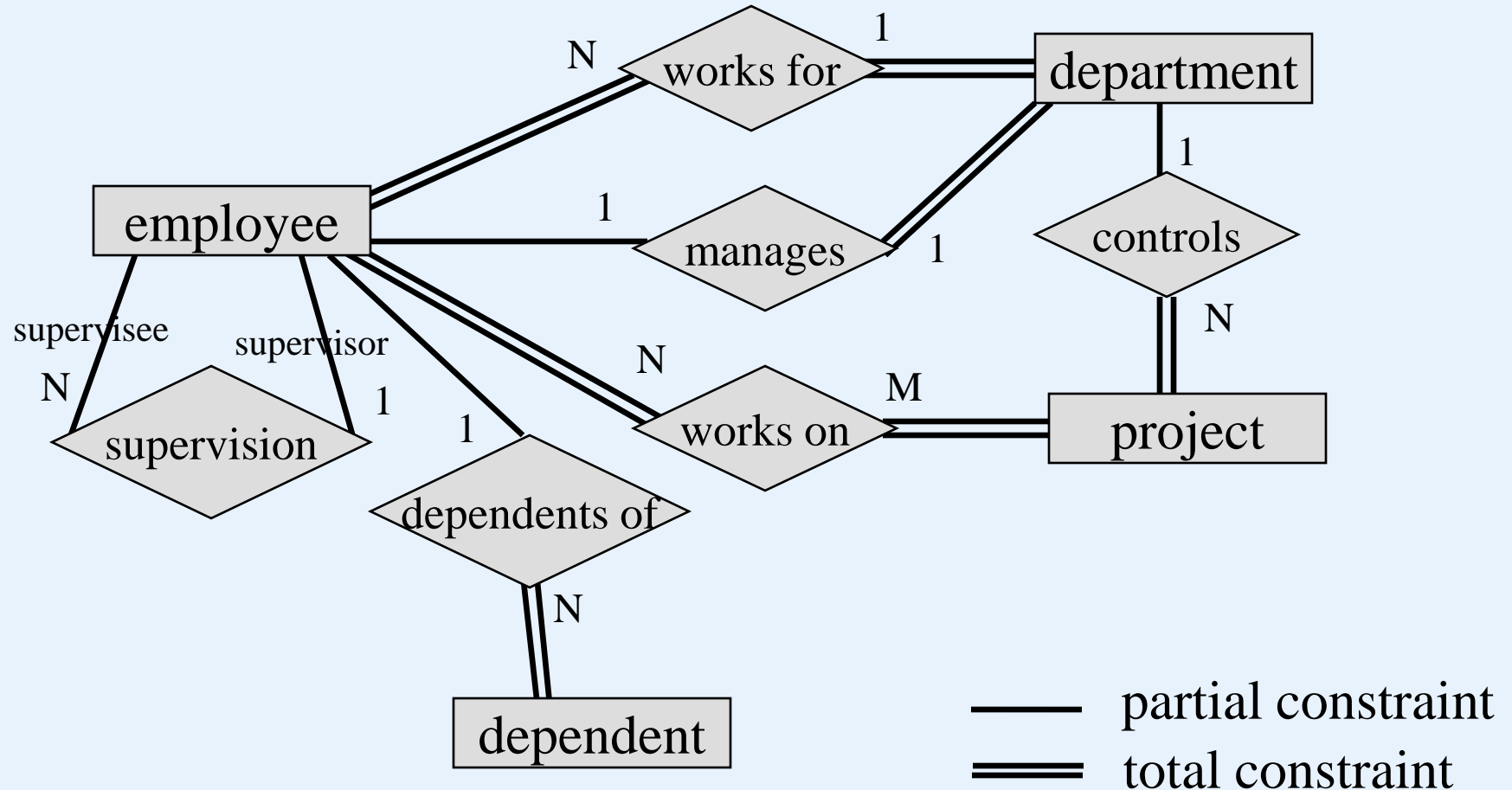
project

The entities:



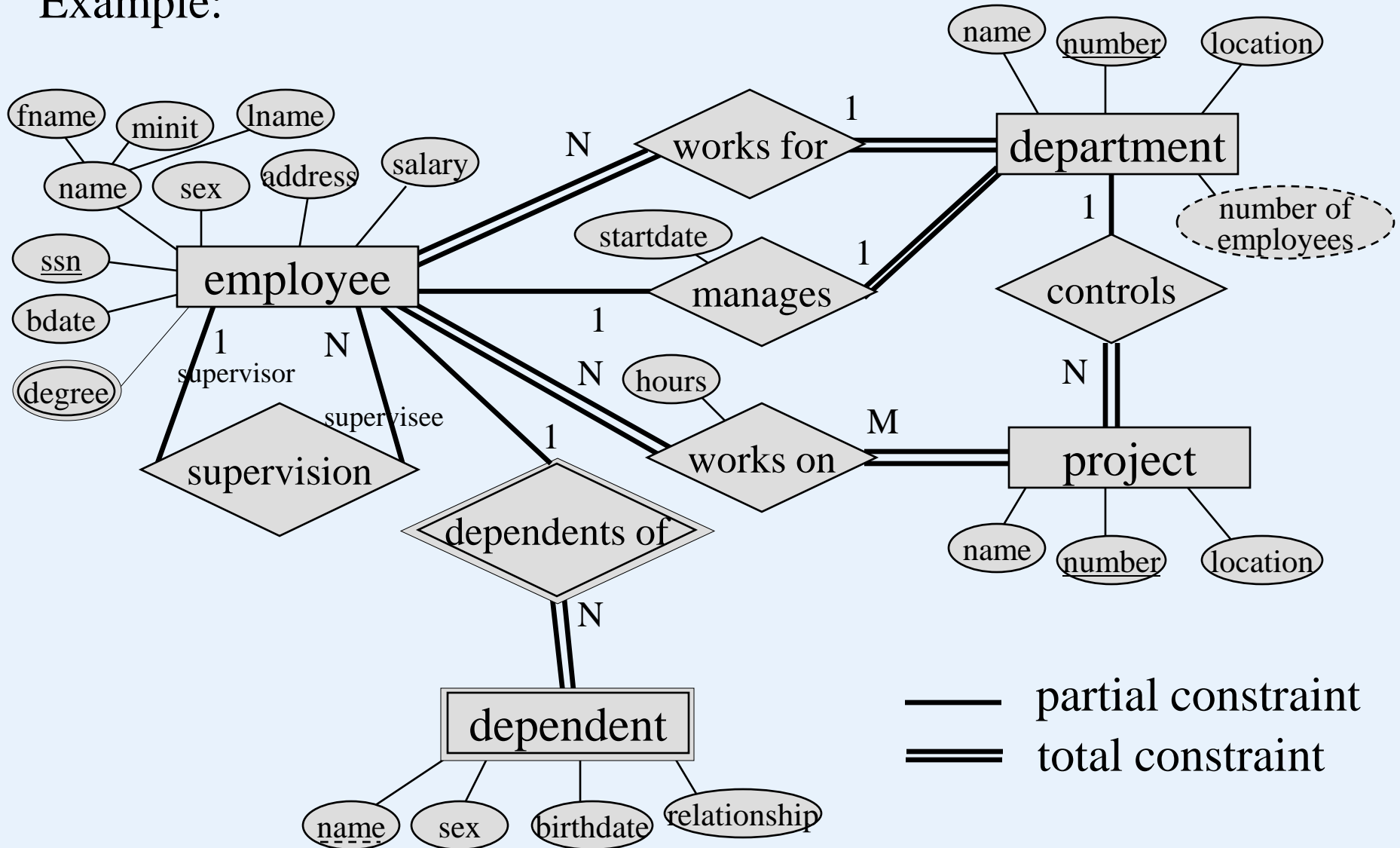
Database Basics

With relationships:

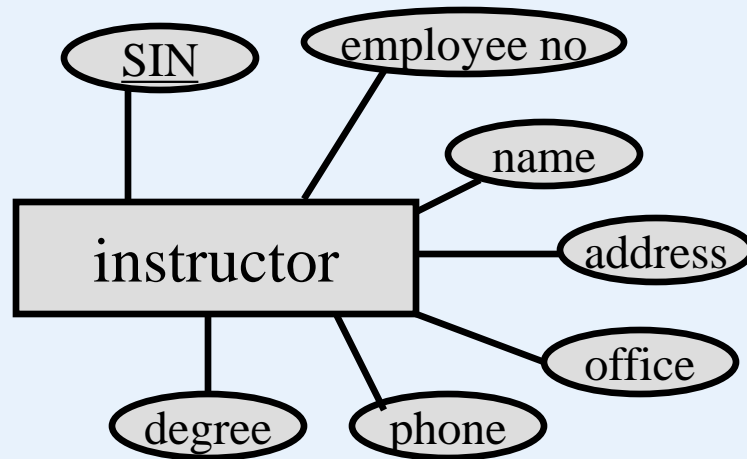


Database Basics

Example:

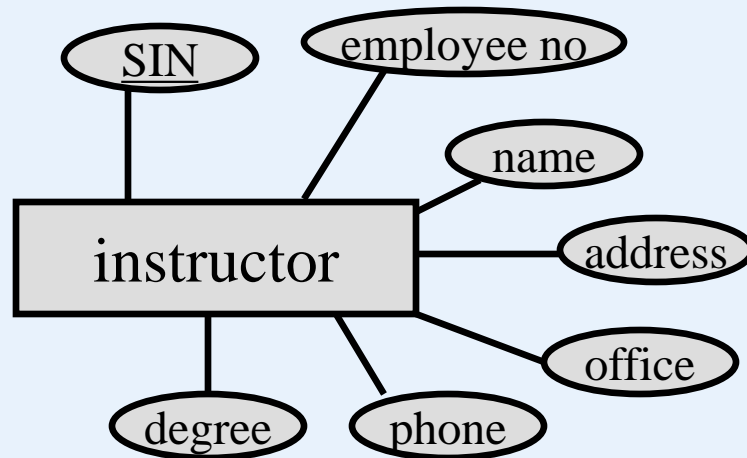


Instructors: let's assume this classification includes instructors, professors, part-time people (at least for now). These people have SIDs, employee numbers, names, addresses, offices, phones, ...



Database Basics

Is there a *key* attribute? What are the *domains*? Can any attribute be *null*? Is any attribute *composite*, *derived*, *complex*, *multivalued*?

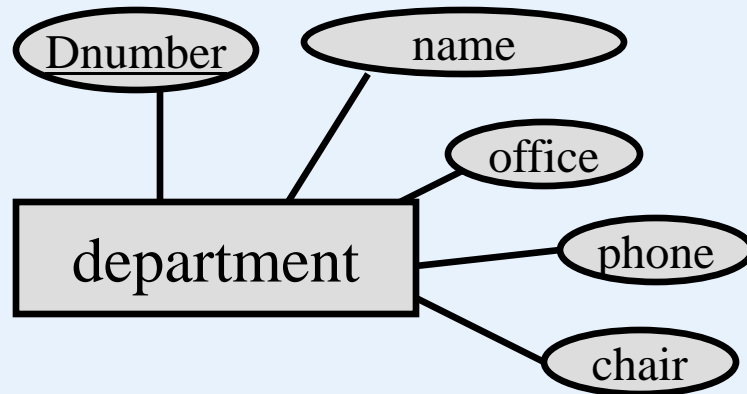


Is this a *weak* entity or a *strong* entity?

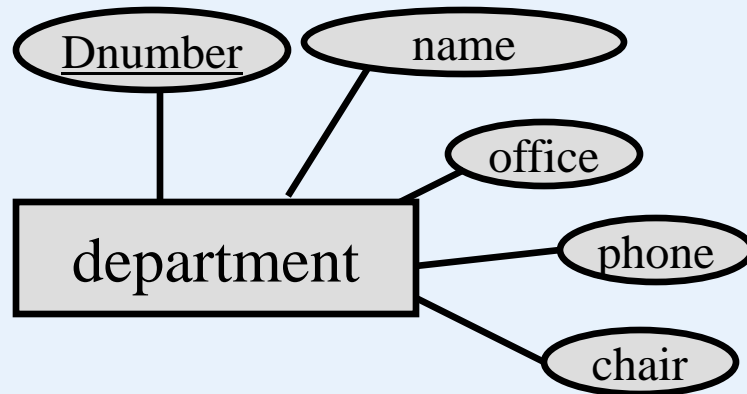
Should department be an attribute?

Departments: obviously instructors are employed by the University and associated with a department

A department has a name, number, office, chair, ...

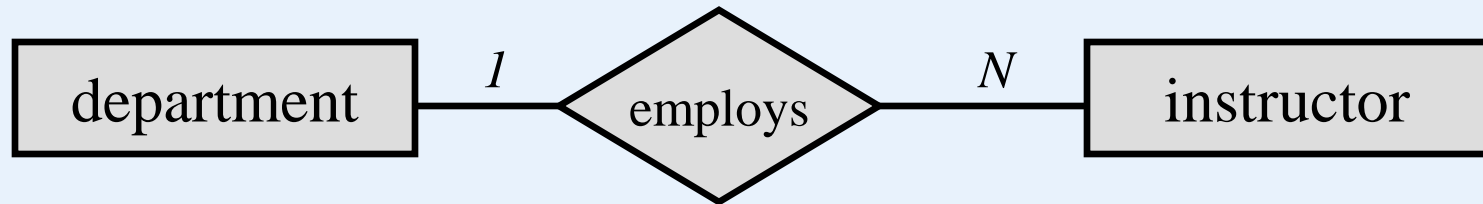


Is there a *key* attribute? What are the *domains*? Can any attribute be *null*? Is any attribute *composite*, *derived*, *complex*, *multivalued*?



Should *chair* be an attribute, or is there a relationship between two entity types?

Employs relationship: If we assume the relationship between department and instructor is $1:N$ then we only associate each department with a single instructor, but we associate any number of instructors with a single department

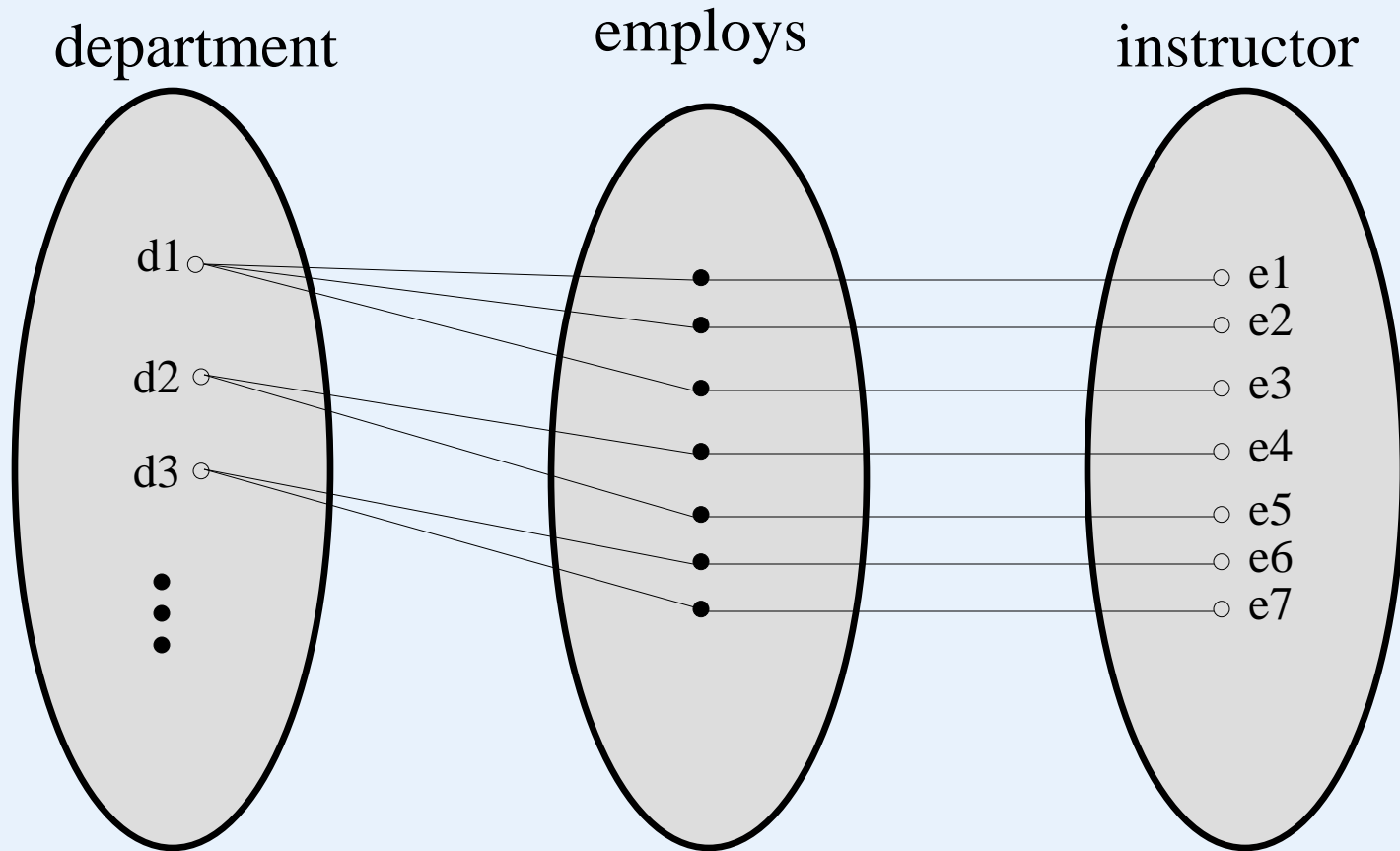


$1:N$ is the *cardinality* of the relationship

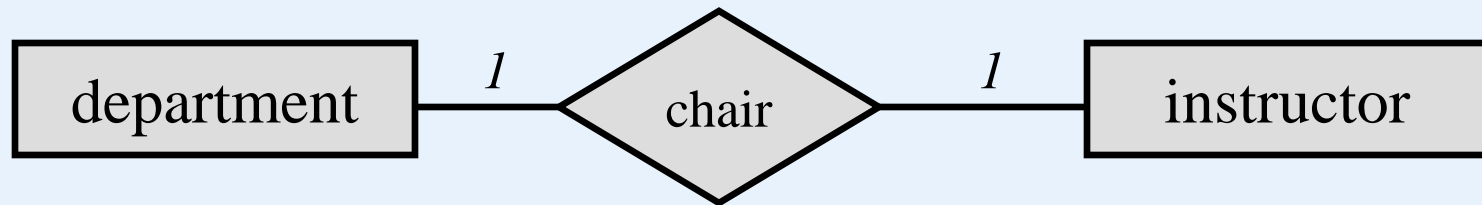
the relationship is of *degree 2*; it is a *binary* relationship

Both entities are considered *strong* entities

Consider some instances



Chair relationship: A department has a *chair* who has special responsibilities. One person (instructor) is designated as such.



1:1 is the *cardinality* of the relationship

the relationship is of *degree 2*; it is a *binary* relationship.

Weak entity types

a weak entity does not have a key of its own - may have a partial key

the identifying relationship will have total participation for the weak entity

e.g. consider courses and sections at UWinnipeg

Consider courses and course sections

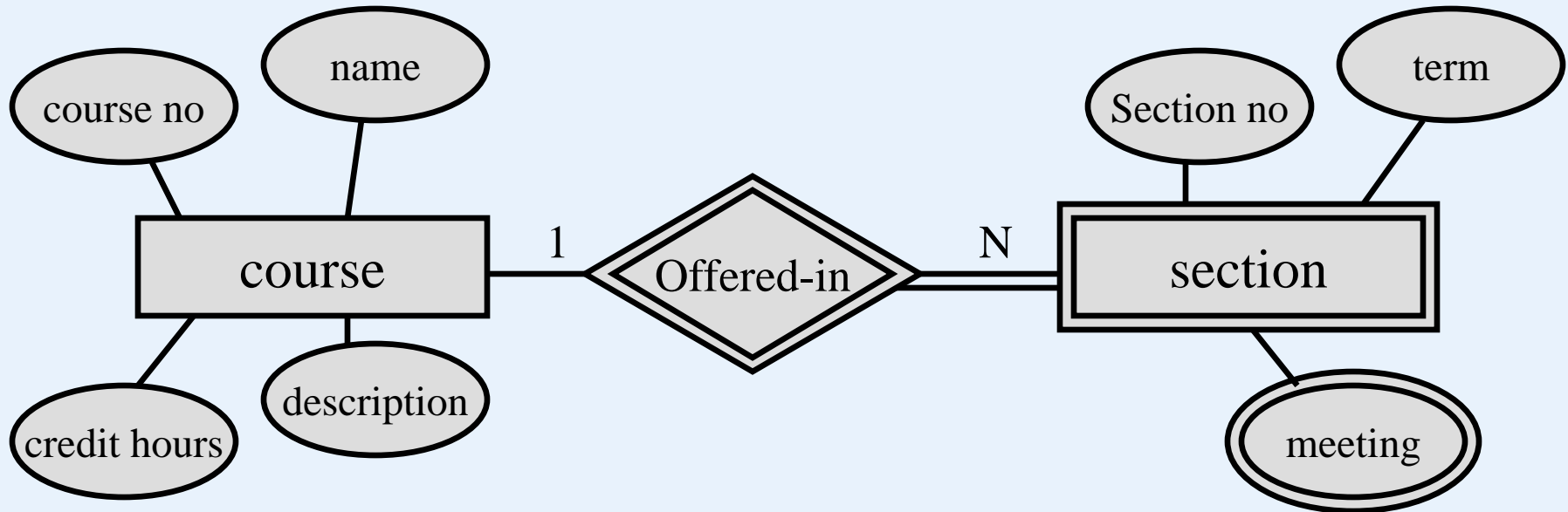
In the fall and winter we have:

```
91.1453/3-001 F Intro Computers staff MW 16:30-17:45 3C13 ..
91.1453/3-002 W Intro Computers staff MW 16:30-17:45 3C13 ..
91.1453/3-050 F Intro Computers staff T 18:00-21:00 3C13 ..
91.1453/3-051 W Intro Computers staff T 18:00-21:00 3C13 ..
```

Section numbers are 001, 002, 050, 051, ...

Sections have a section number, a term, days and times, ...

Consider courses and course sections



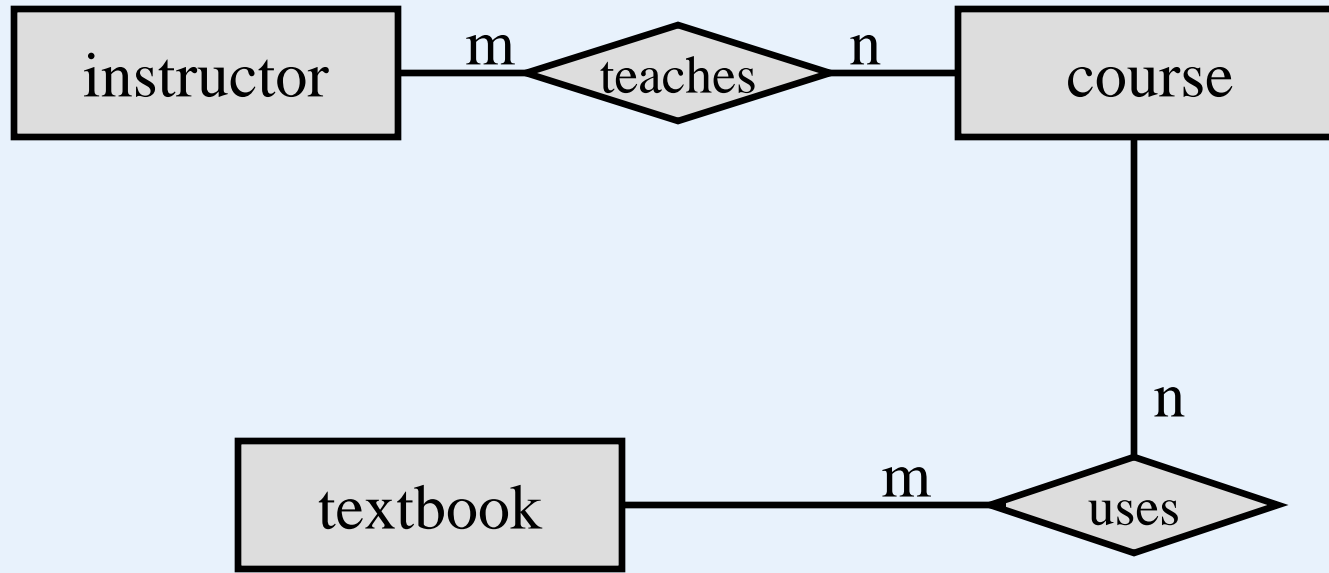
Section is a *weak entity* - it has a *discriminator* (partial key), section number.

Section *totally* participates in the *offered in* relationship

PK (primary key) of Section is ... (*offered_in* is an *identifying* relationship)

Is meeting *multivalued*?

Data analysis:

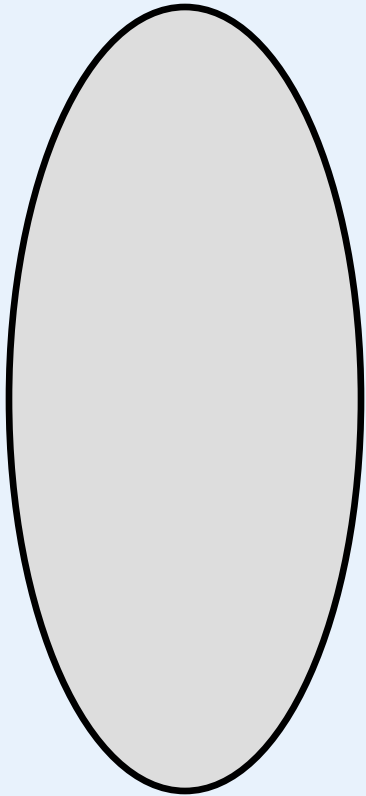


Note that *teaches* and *uses* are both binary relationships:

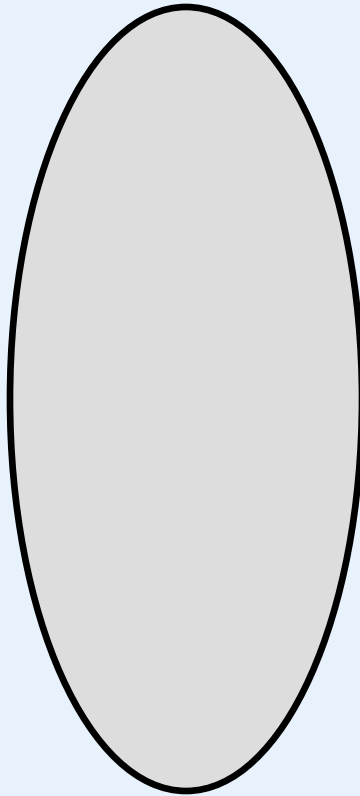
- we expect situations where a specific Instructor *teaches* a specific Course, and where
- a specific Course *uses* a specific text

Consider instances

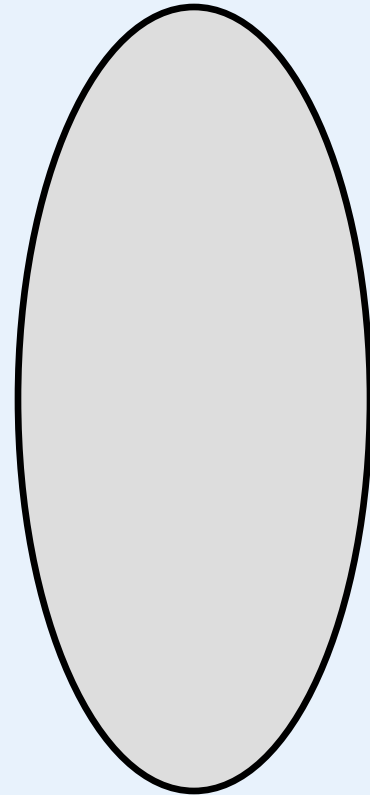
instructor



course

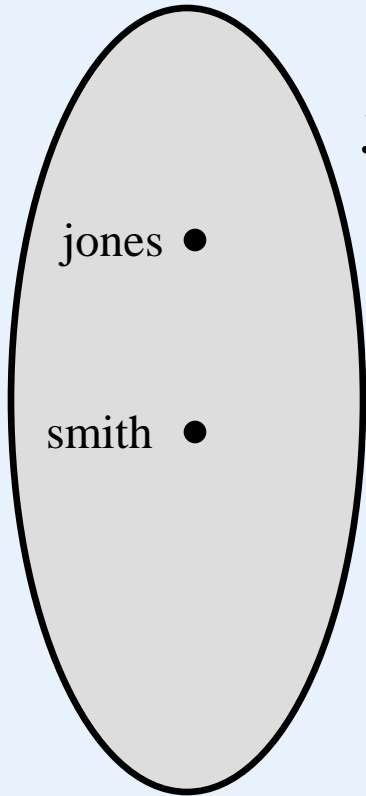


textbook



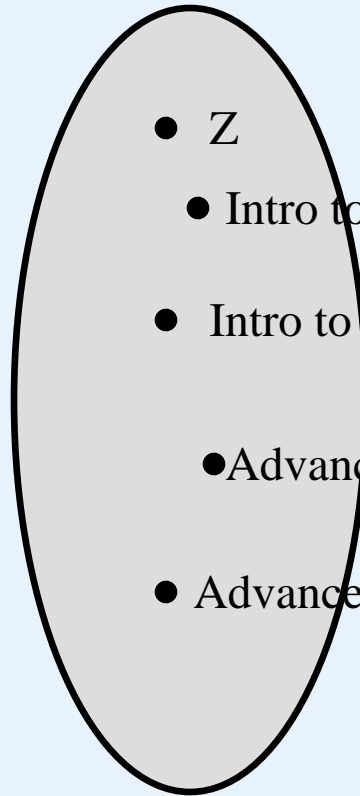
Database Basics

instructor



*Jones and Smith
are Instructors*

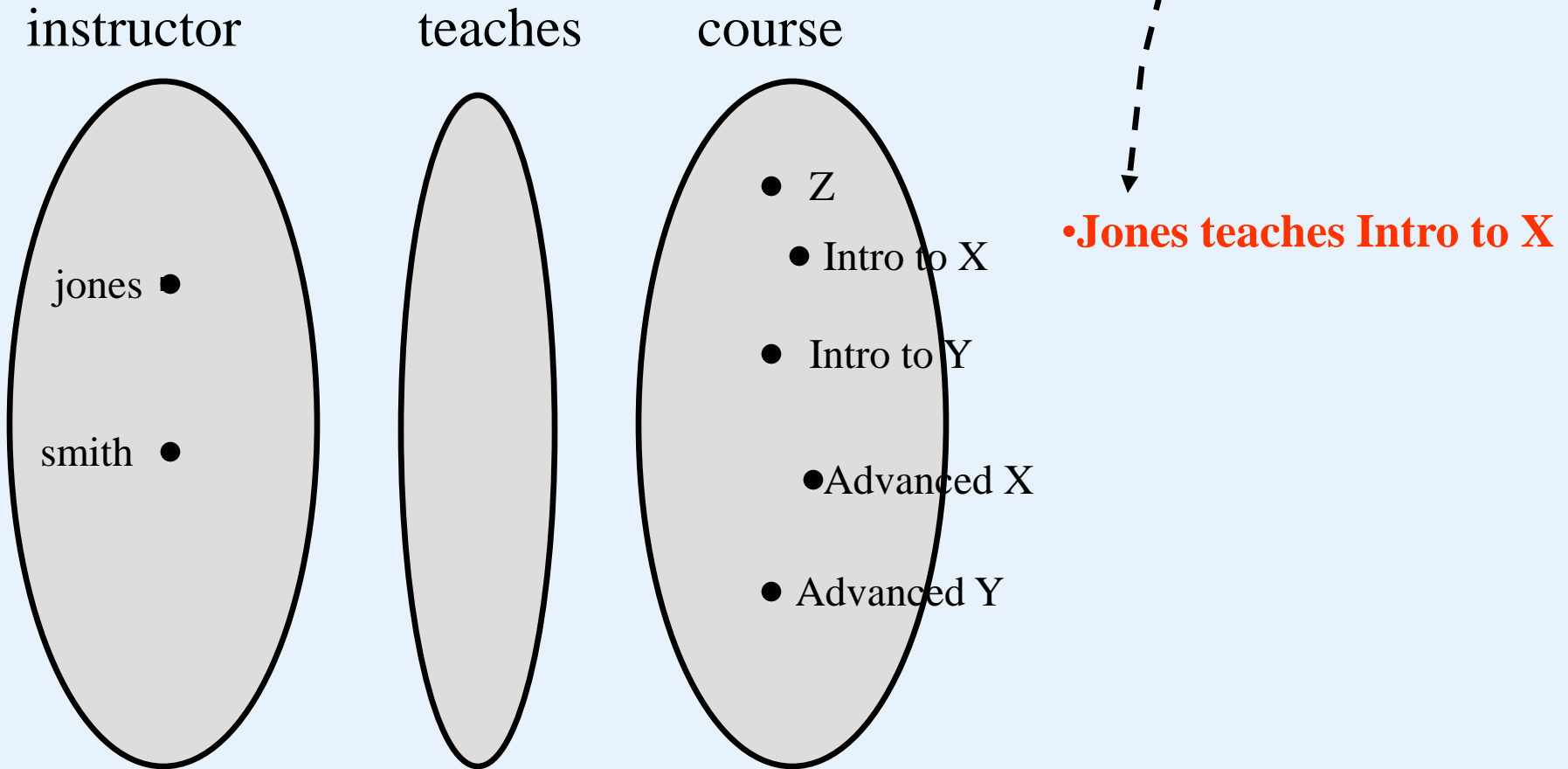
course



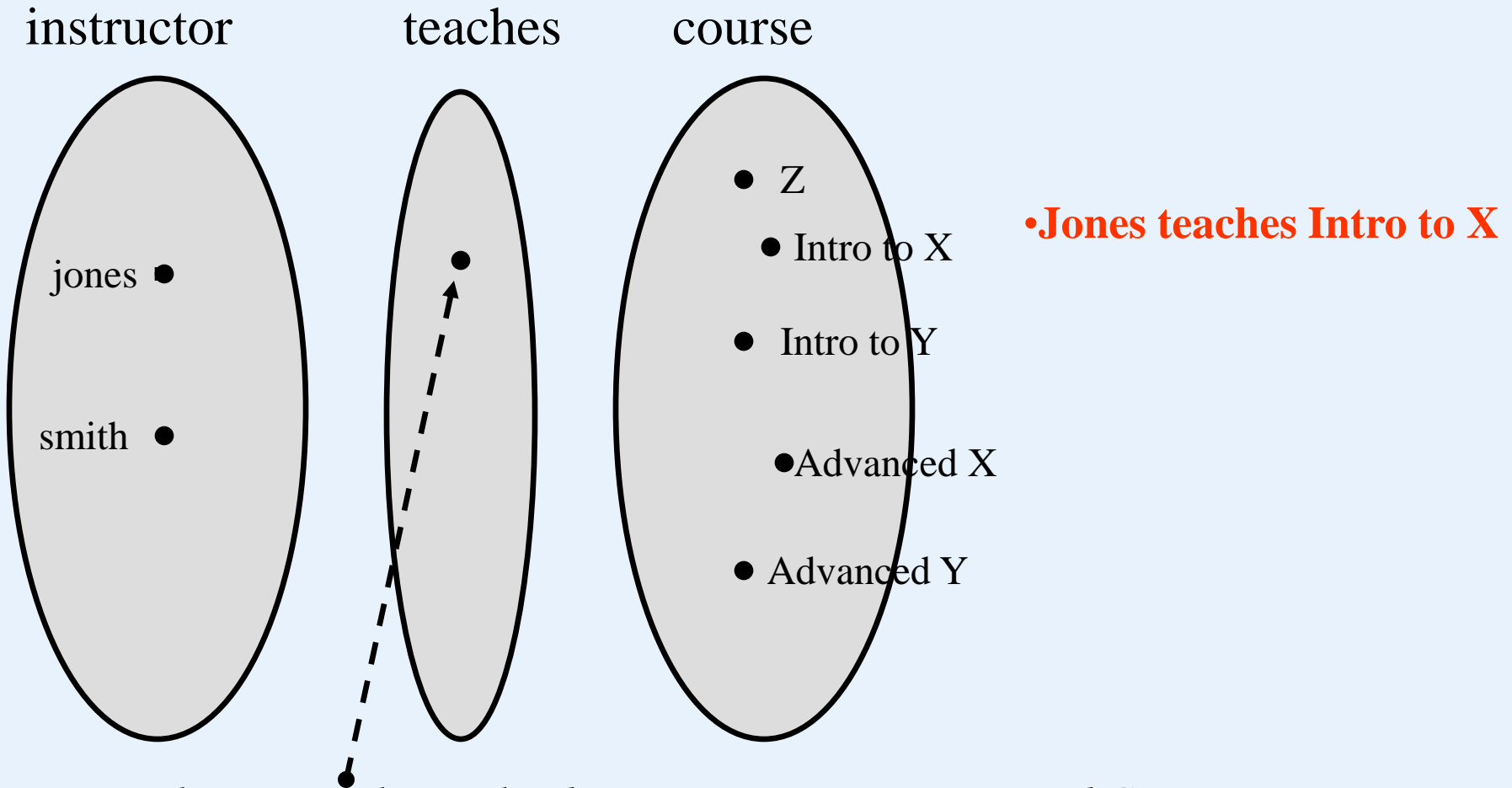
*Courses offered are
Intro to X,
Intro to Y,
Z,
Advanced X,
Advanced Y*

Database Basics

Consider instances of *Instructor teaches Course*

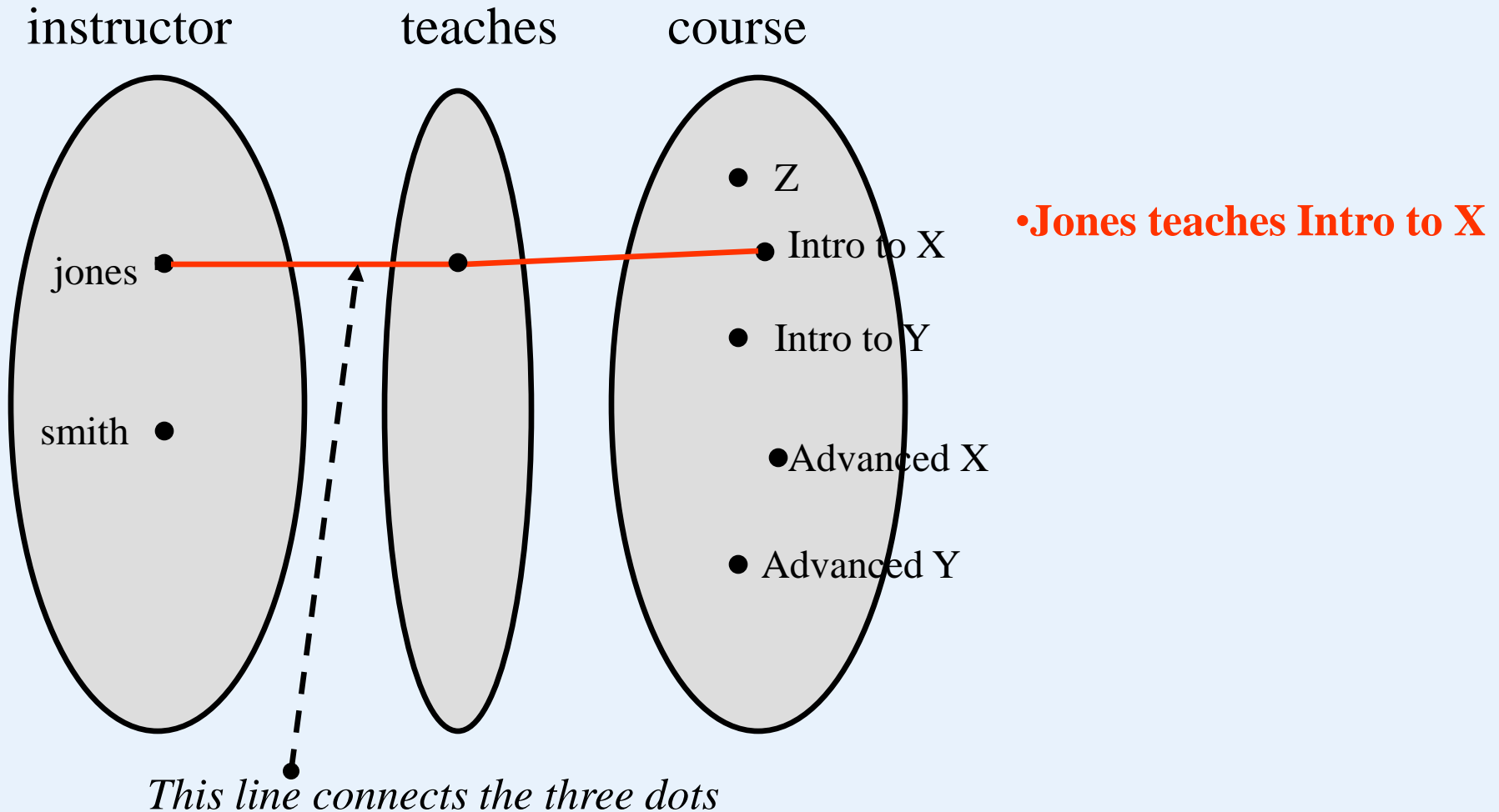


Database Basics

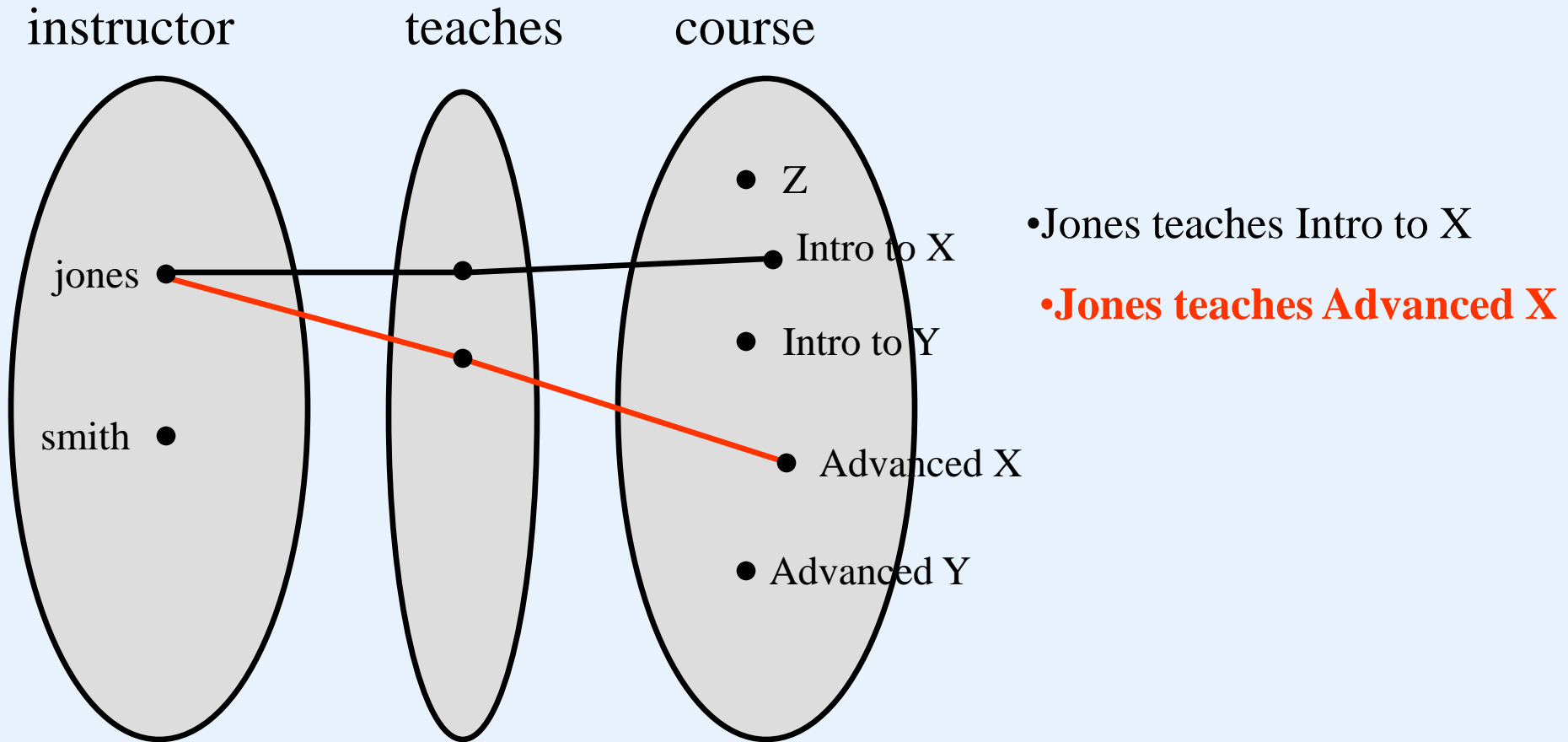


There is a relationship between Instructor Jones and Course Intro to X

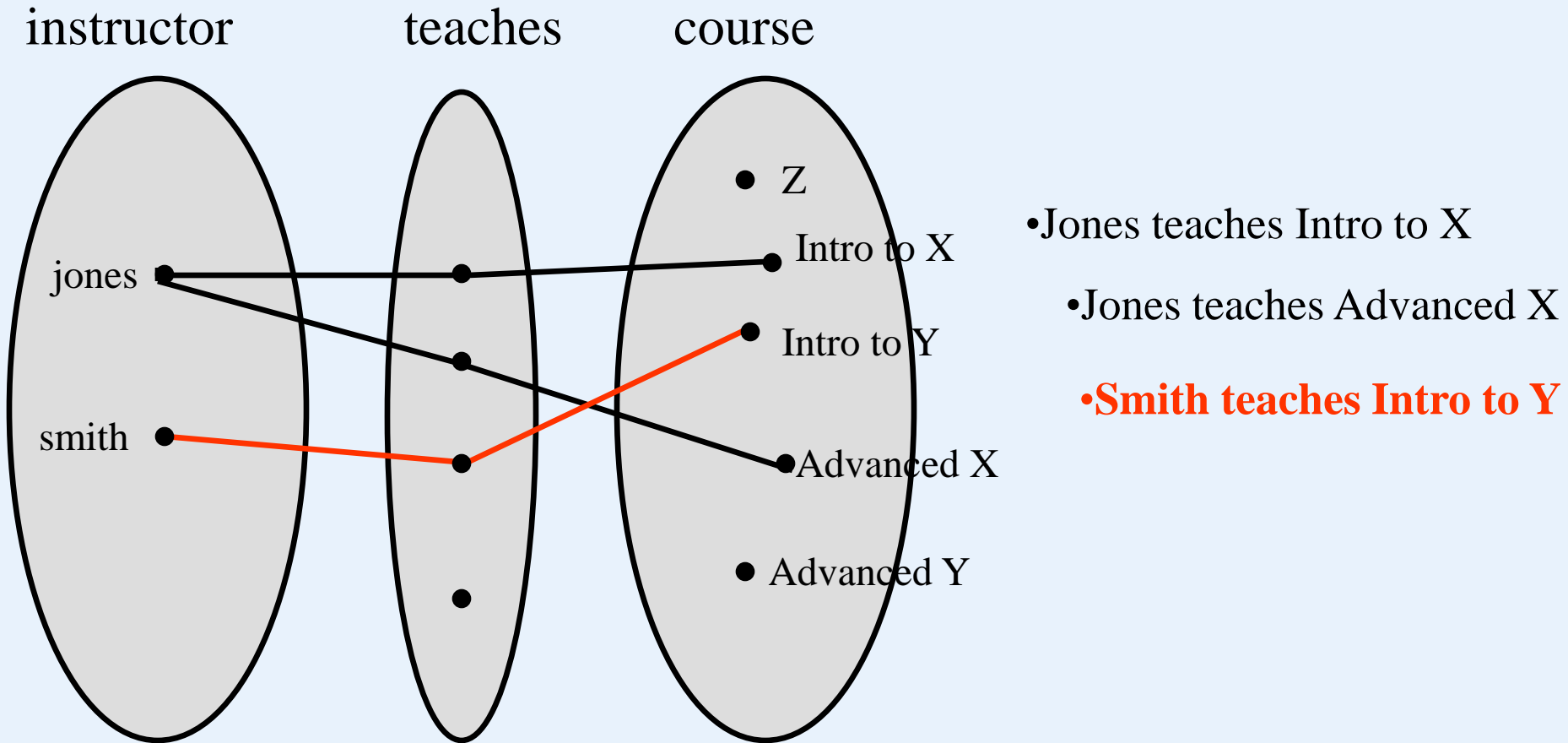
Database Basics



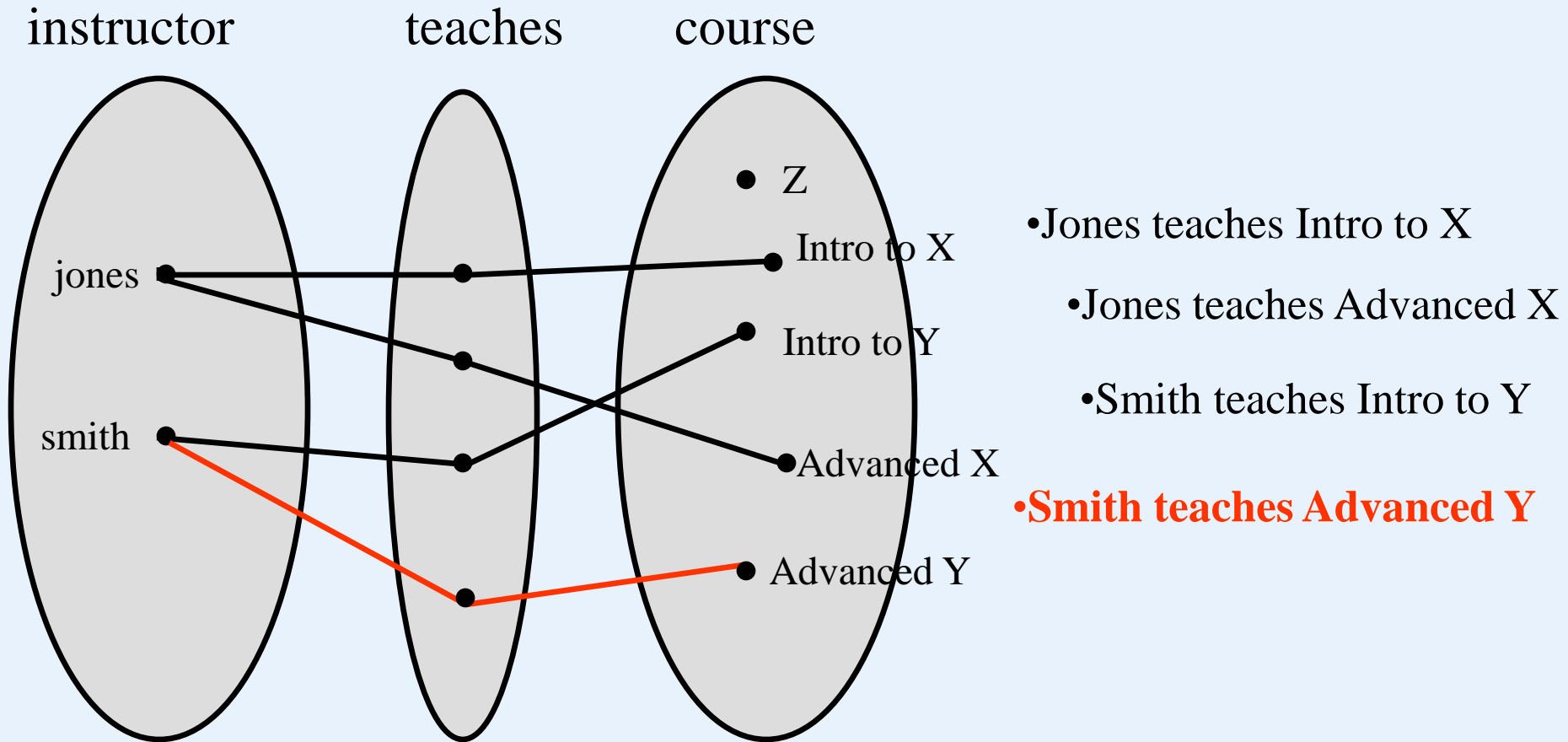
Database Basics



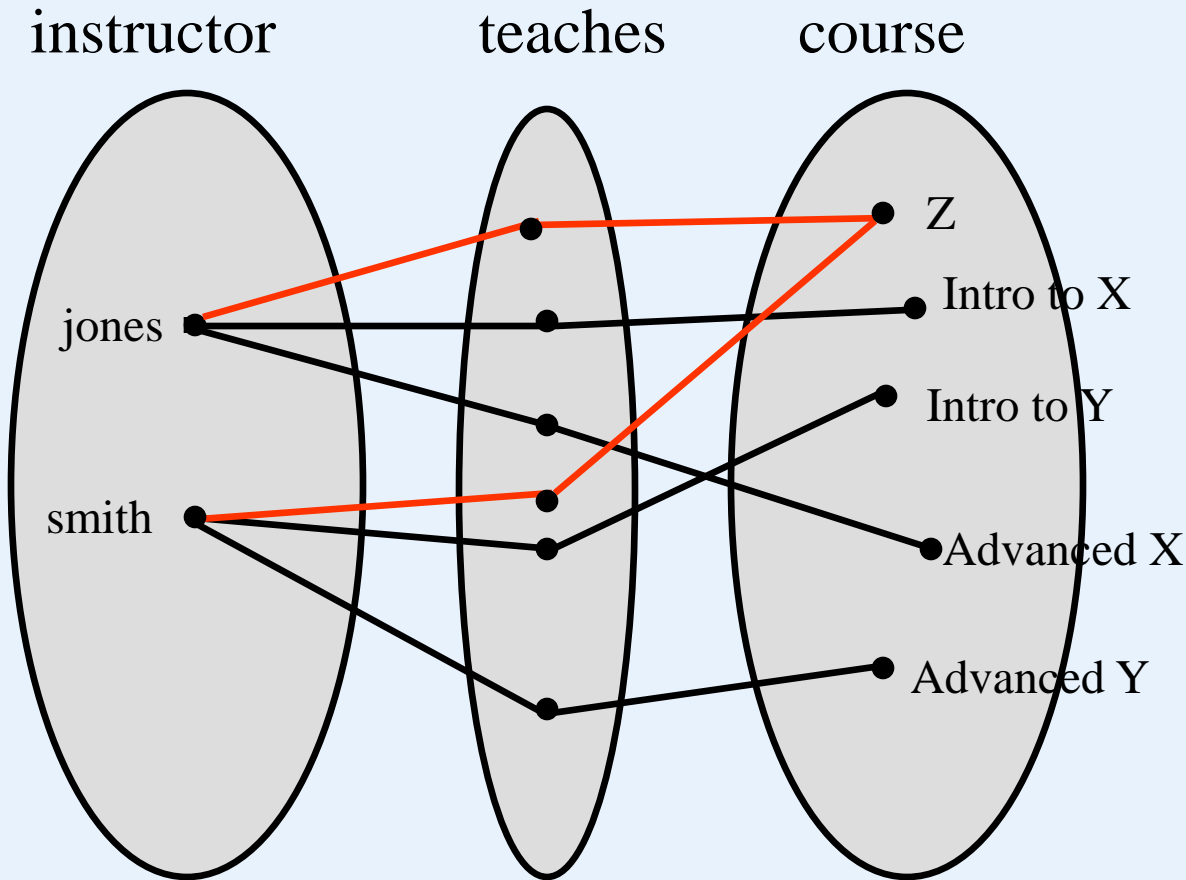
Database Basics



Database Basics



Database Basics

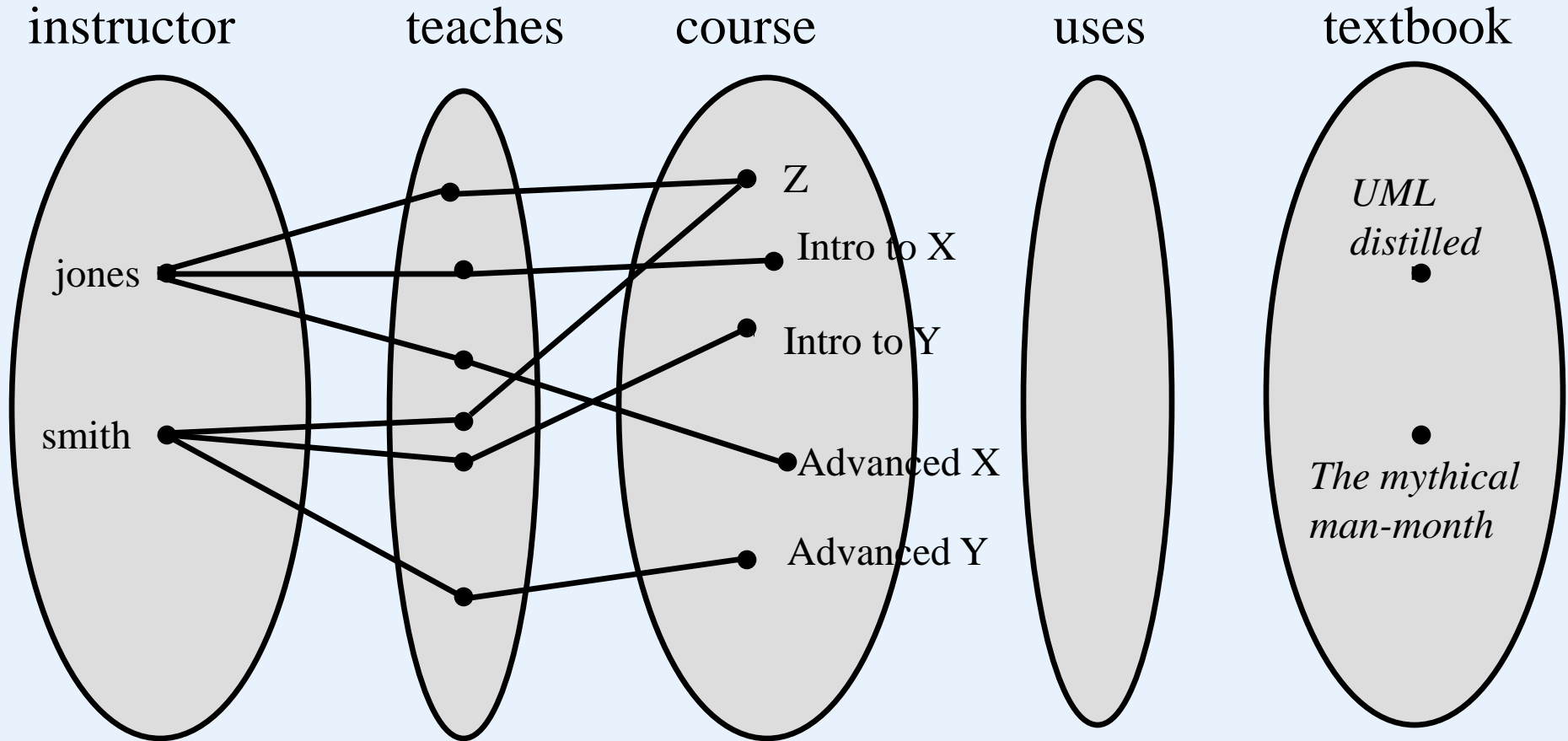


- Jones teaches Intro to X
- Jones teaches Advanced X
- Smith teaches Intro to Y
- Smith teaches Advanced Y

• Smith and Jones teach Z together

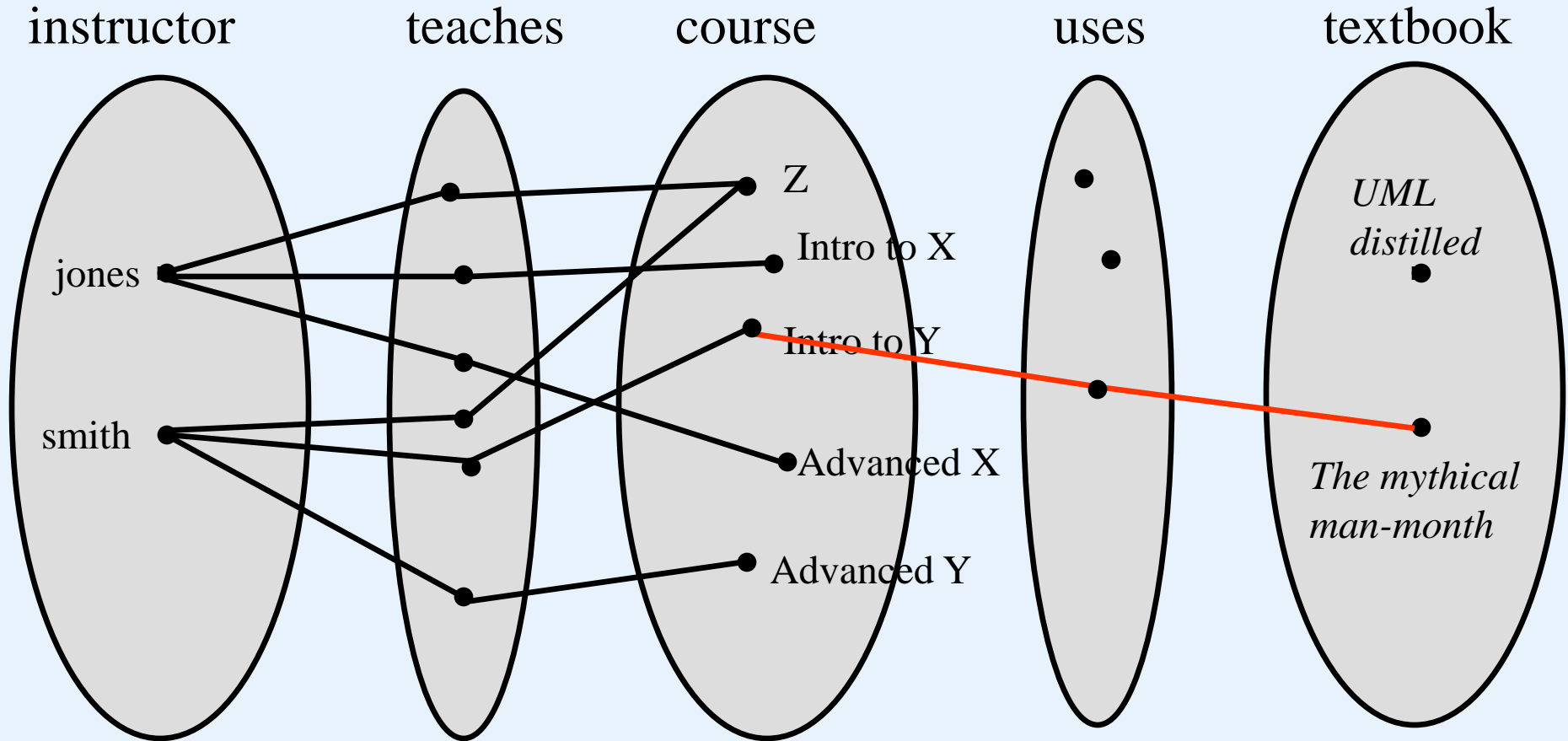
*There are two relationships:
one between Jones and Z; the
other between Smith and Z*

Now let us examine *Course uses Textbook*



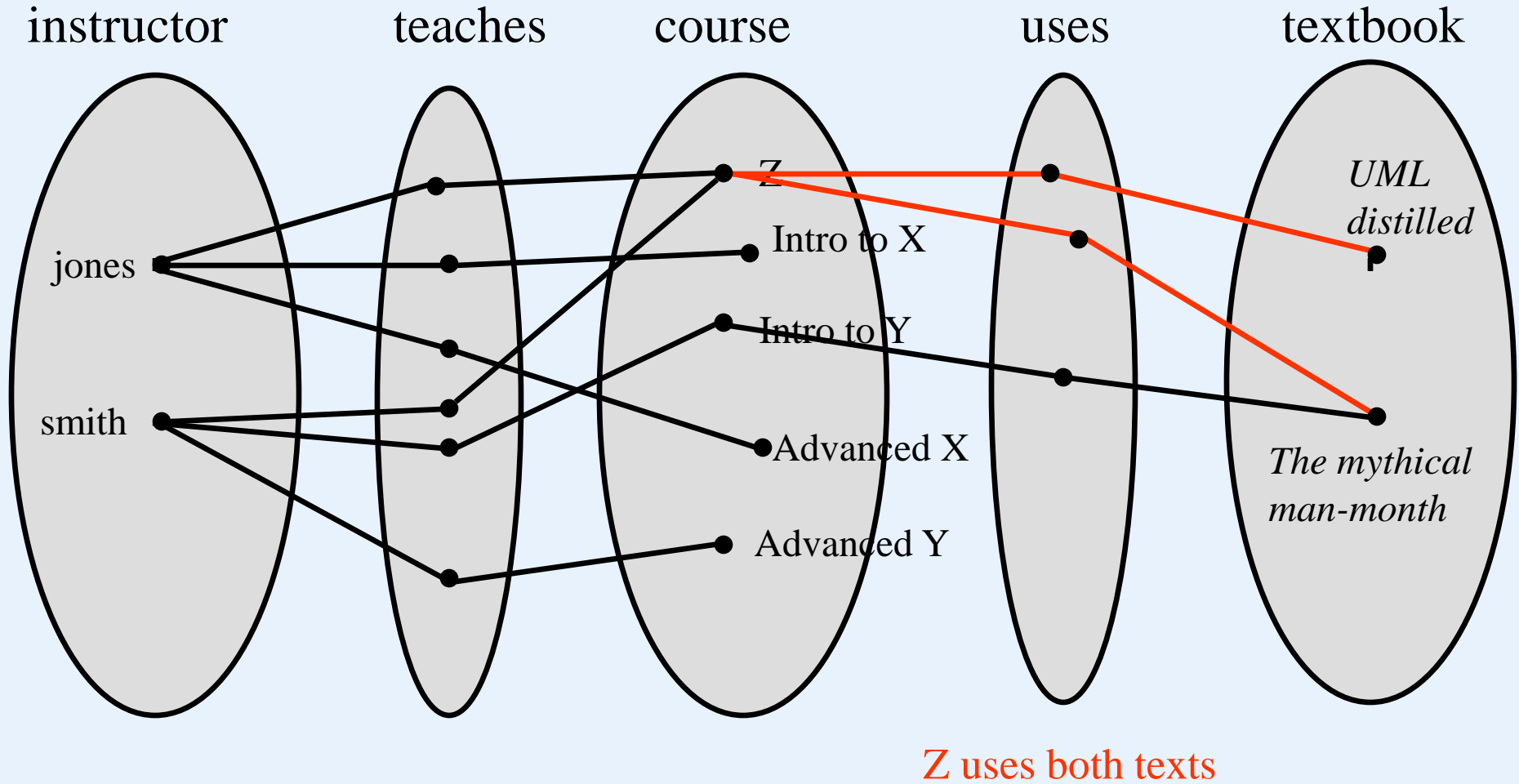
Suppose we have two textbooks: *The mythical man-month*, and *UML distilled*

Database Basics



Intro to Y uses The mythical man-month

Database Basics



- **ER-to-Relational mapping**

1. Create a relation for each strong entity type

- For each atomic attribute associated with the entity type, an attribute in the relation will be created.
- Composite attributes are not included. However the atomic attributes comprising the composite attribute must appear in the pertinent relation.

2. Create a relation for each weak entity type

- include primary key of owner (an FK - foreign key)
- owner's PK + partial key becomes PK

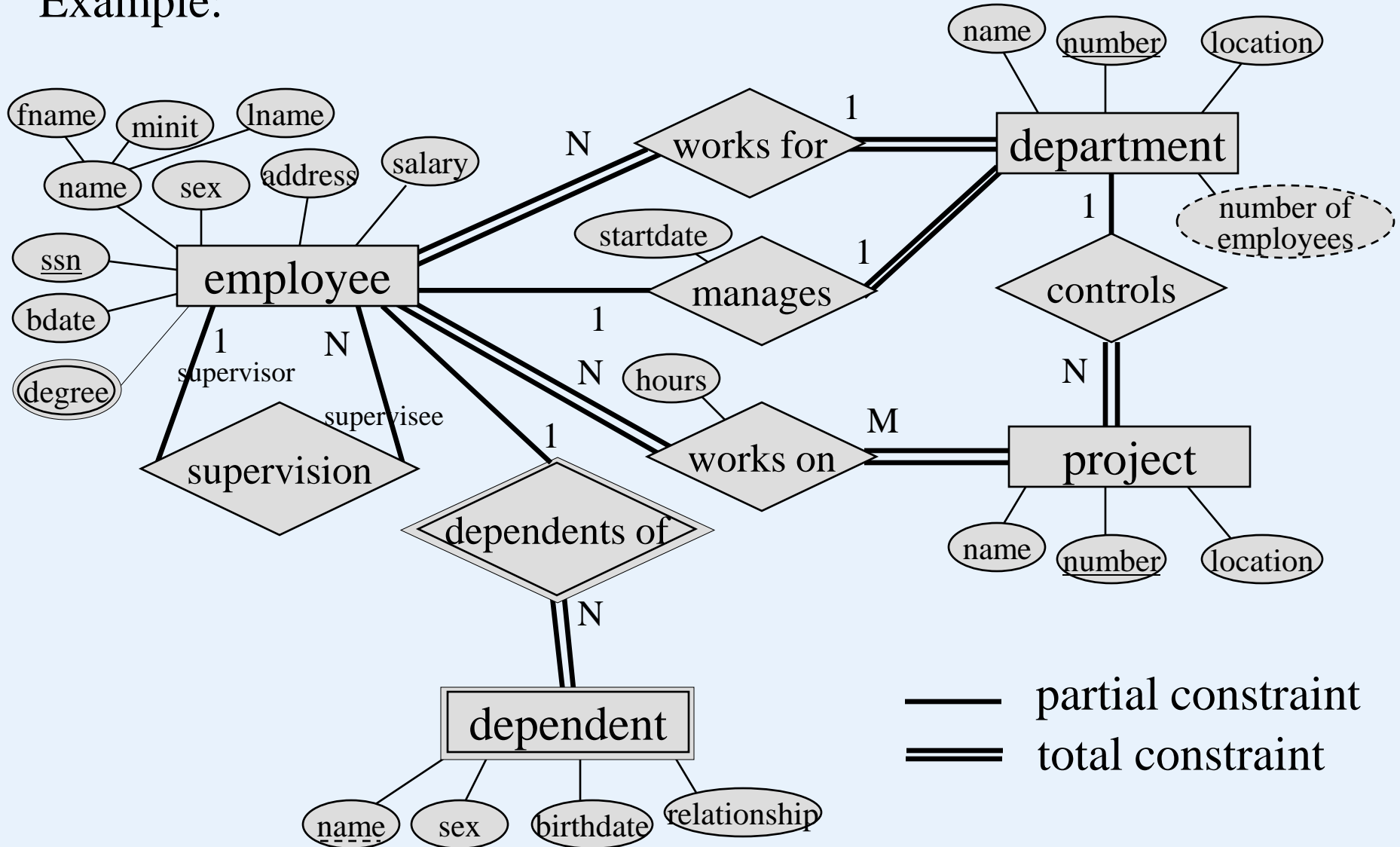
3. For each binary *1:1* relationship choose an entity and include the other's PK in it as an FK. Include any attributes of the relationship

4. For each binary $1:n$ relationship, choose the n -side entity and include an FK with respect to the other entity. Include any attributes of the relationship
5. For each binary $M:N$ relationship, create a relation for the relationship
 - include PKs of both participating entities and any attributes of the relationship
 - PK is the concatenation of the participating entity PKs
6. For each multivalued attribute create a new relation
 - include the PK attributes of the entity type
 - PK is the PK of the entity type and the multivalued attribute

7. For each n -ary relationship, create a relation for the relationship
 - include PKs of all participating entities and any attributes of the relationship
 - PK is the concatenation of the participating entity PKs

Database Basics

Example:



Database Basics

EMPLOYEE

fname, minit, lname, ssn, bdate, address, sex, salary, superssn, dno

DEPARTMENT

Dname, dnumber, mgrssn, mgrstartdate

Dnumber, dlocation

DEPT_LOCATIONS

PROJECT

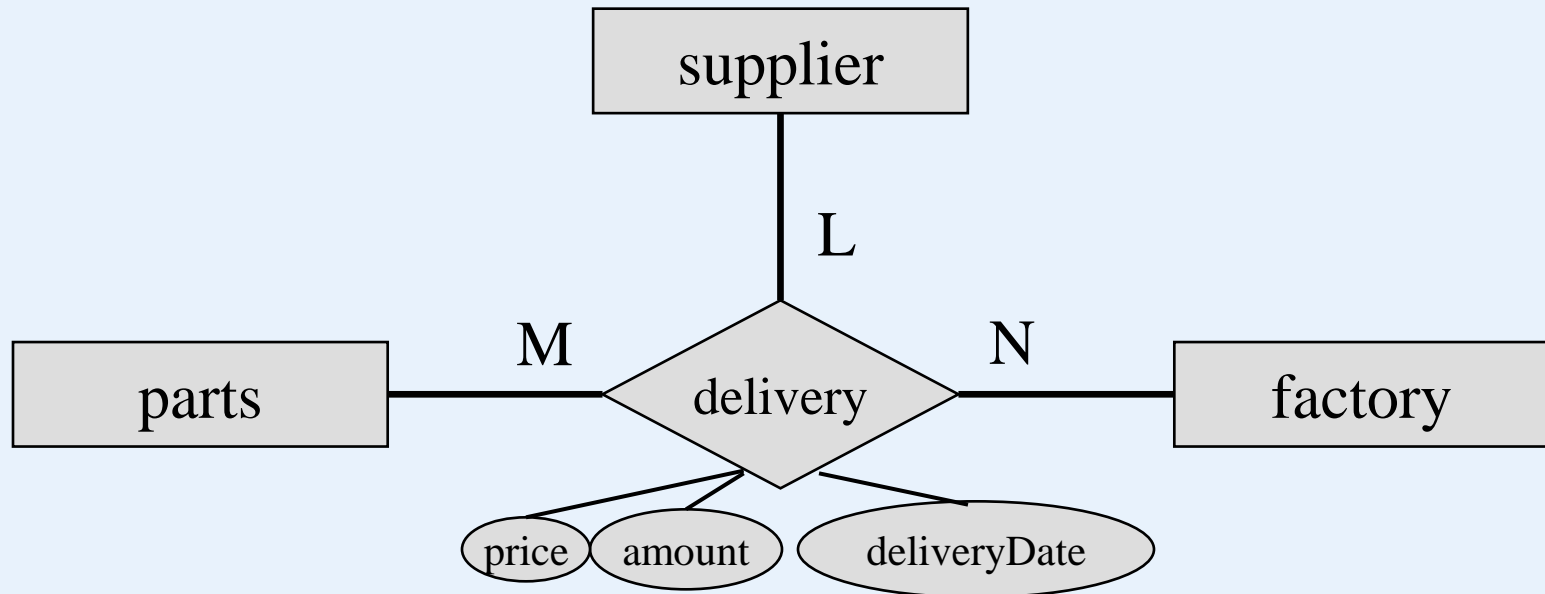
Pname, pnumber, plocation, dnum

Essn, pno, hours

WORKS_ON

DEPENDENT

Essn, dependentname, sex, bdate, relationship

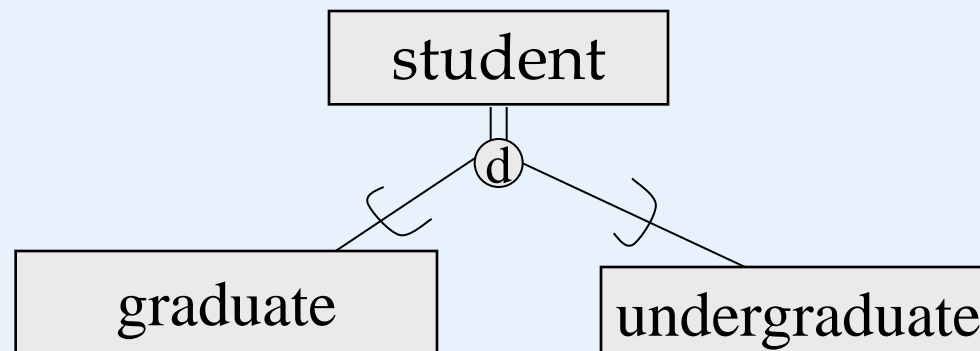


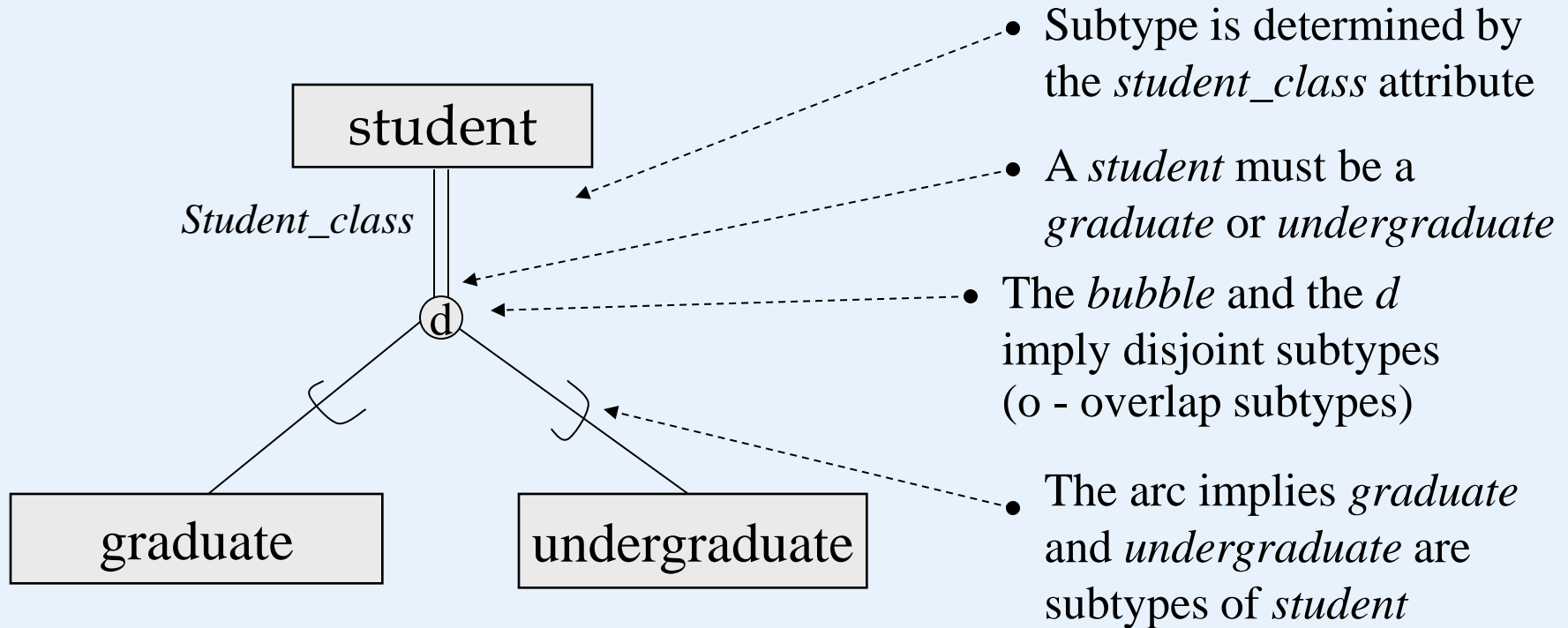
Delivery

sNum, pNum, fNum, price, amount, deliveryDate

• Specialization and Generalization

- Specialization is the process of defining a set of sub-entities of some entity type. Generalization is the opposite approach/process of determining a supertype based on certain entities having common characteristics.
 - e.g. employees may be paid by the hour or a salary (part vs full-time)
 - e.g. students may be part-time or full-time; graduate or undergraduate
- these are similar to 1:1 relationships, but they always involve entities of one (super)type
- these are *'is-a'* relationships





- Participation of supertype may be mandatory or optional
- Subtypes may be disjoint or overlapping
- a predicate (on an attribute) determines the subtype: e.g. attribute *Student_class*

Student_class = 'graduate'; *Student_class* = 'undergraduate'

- **Mapping to a relational database**

4 choices:

1. Create separate relations for the supertype and each of the subtypes.

2. Create relations for the subtypes only - each contains attributes from the supertype.

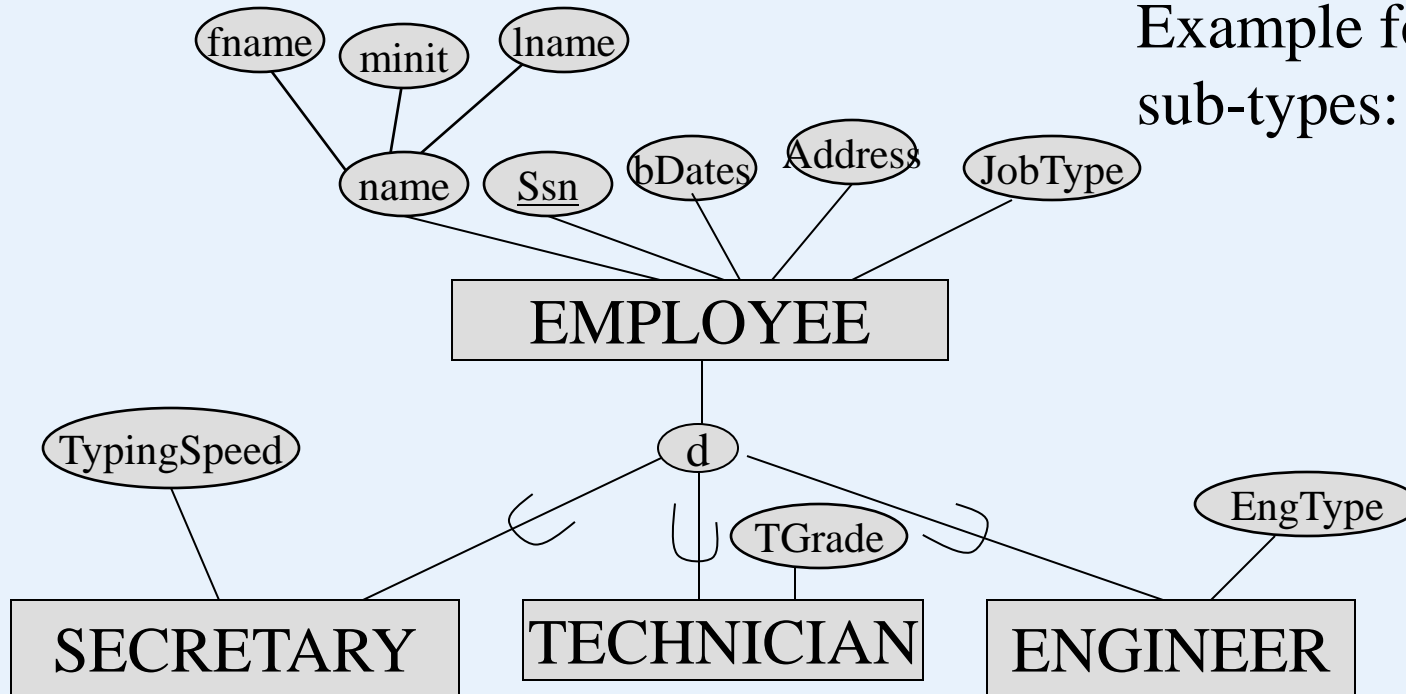
3. (**disjoint** subtypes) Create only one relation - includes all of the attributes for the supertype and all for the subtypes, and one discriminator attribute.

4. (**overlapping** subtypes) Create only one relation - includes all of the attributes for the supertype and all for the subtypes, and one logical discriminator attribute per subtype.

PK is always the same - determined from the supertype

Database Basics

Example for super- & sub-types: choice 1



EMPLOYEE

fname, minit, lname, ssn, bdate, address, JobType

SECRETARY

Essn, TypingSpeed

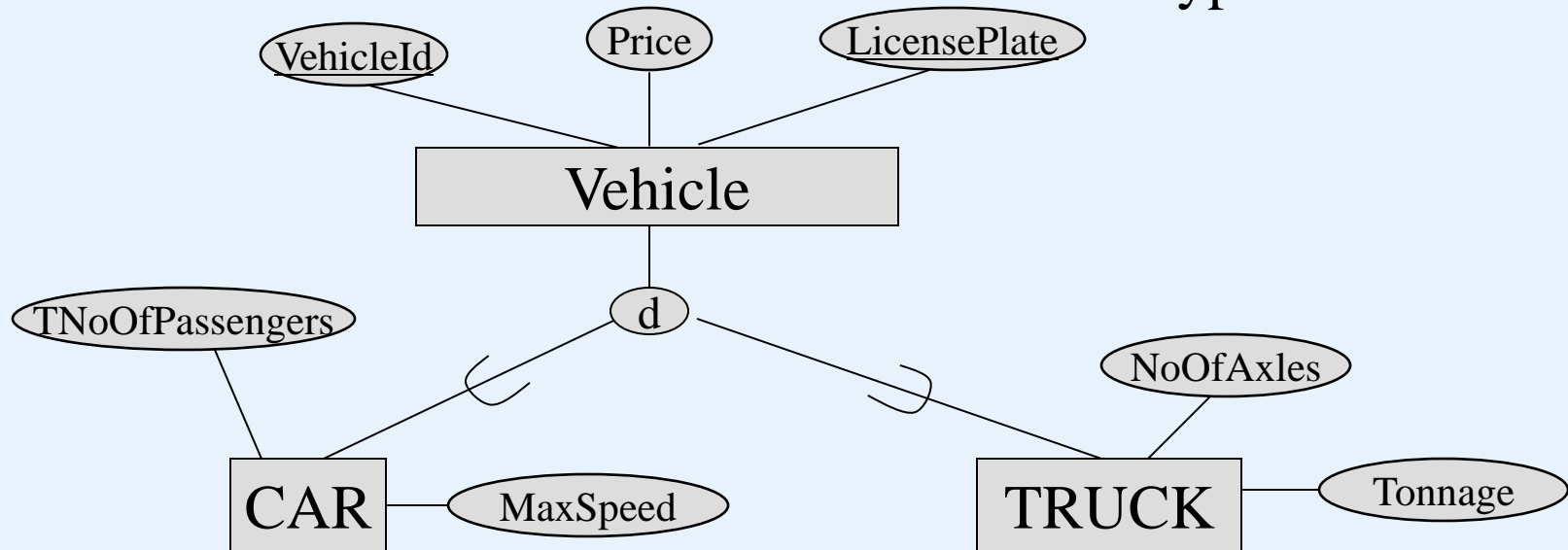
TECHNICIAN

Essn, TGrade

ENGINEER

Essn, EngType

Example for super- & sub-types: choice 2



CAR

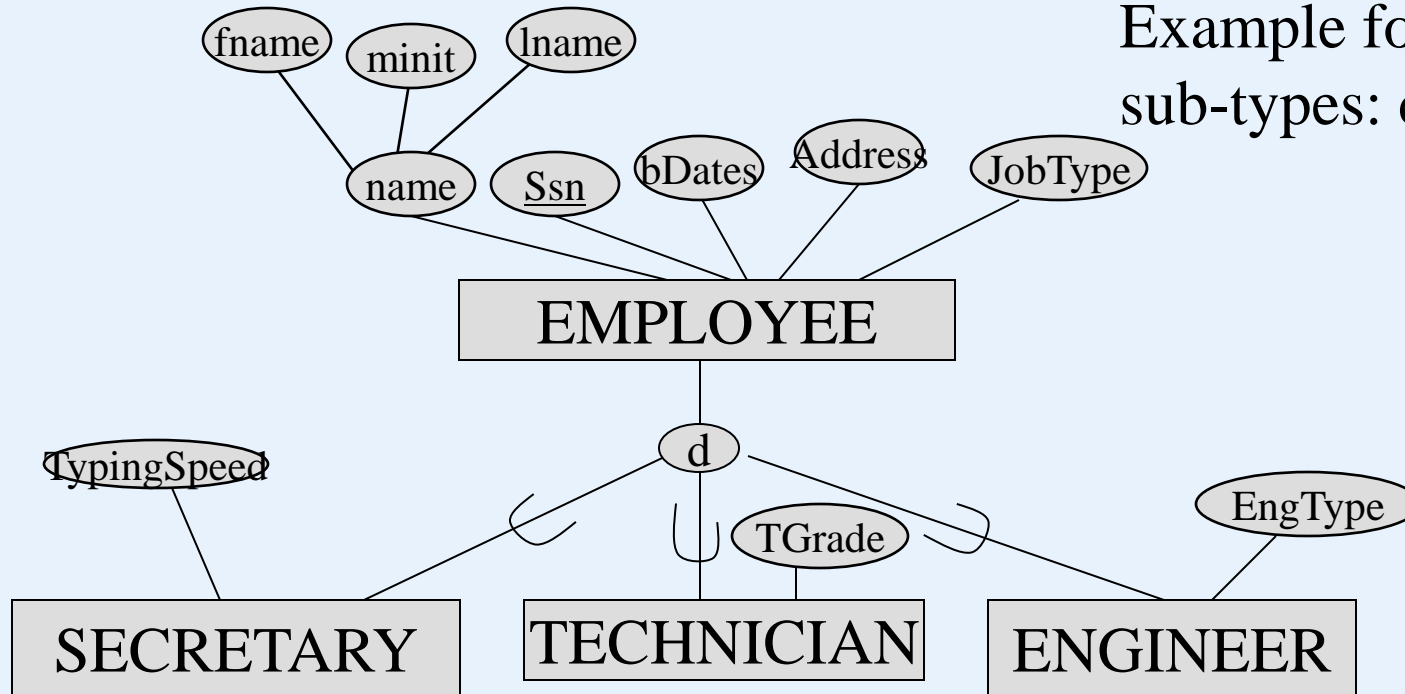
VehicleId, LicensePlate, Price, MaxSpeed, NoOfPassenger

TRUCK

VehicleId, LicensePlate, Price, NoOfAxles, Tonnage

Database Basics

Example for super- & sub-types: choice 3

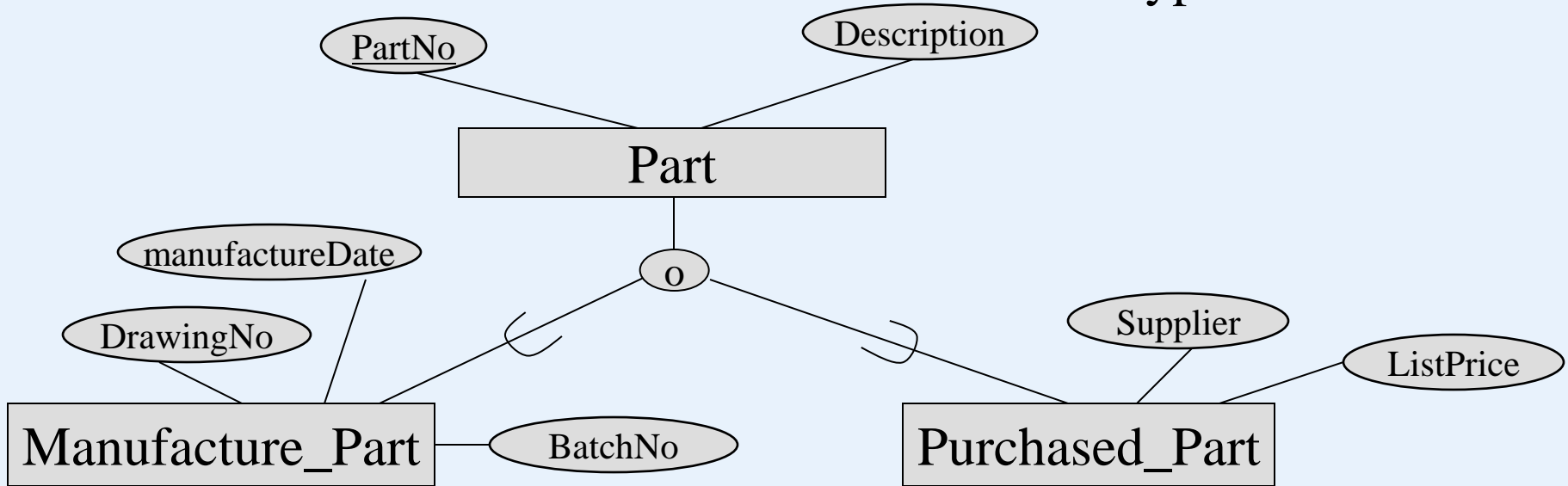


EMPLOYEE

fname, minit, lname, ssn, bdate, address, JobType, TypingSpeed, Tgrade, EngType

12345	1		
56463	2		
55554	...		3		

Example for super- & sub-types: choice 4



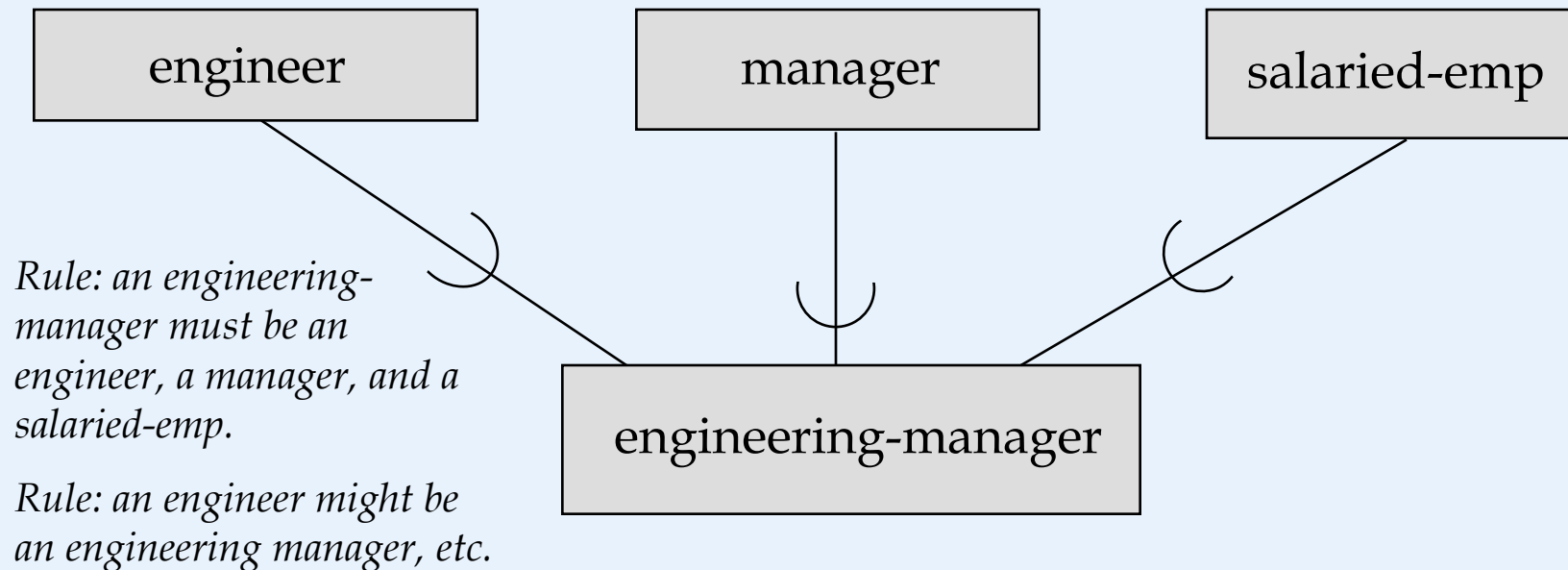
Part

PartNo, Description, MFlag, Drawing, ManufactureDate, BatchNo, Pflag, Supplier, ListPrice

1	screw	1
2	bolt		1
3	axes	1	1		

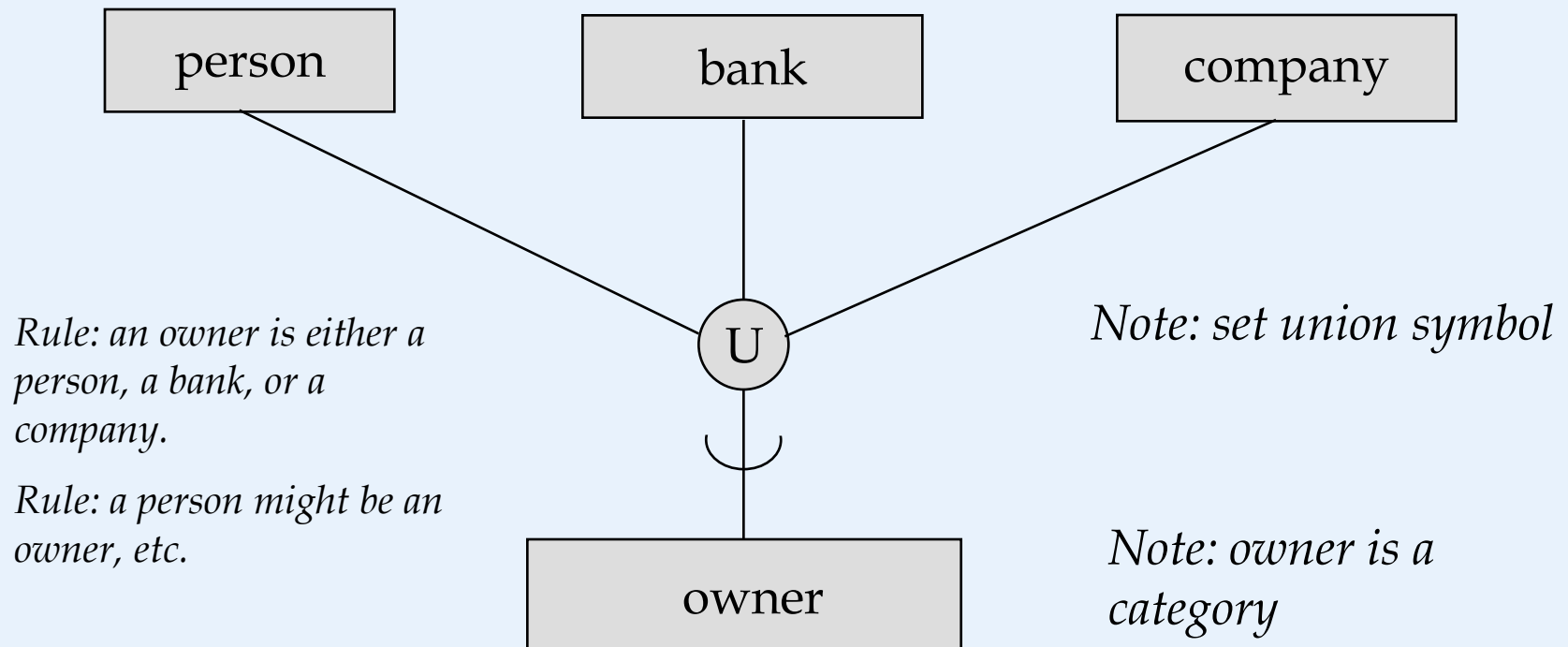
☞ Shared SubClass

- a subclass with more than one superclass
- leads to the concept of multiple inheritance: engineering manager inherits attributes of engineer, manager, and salaried employee



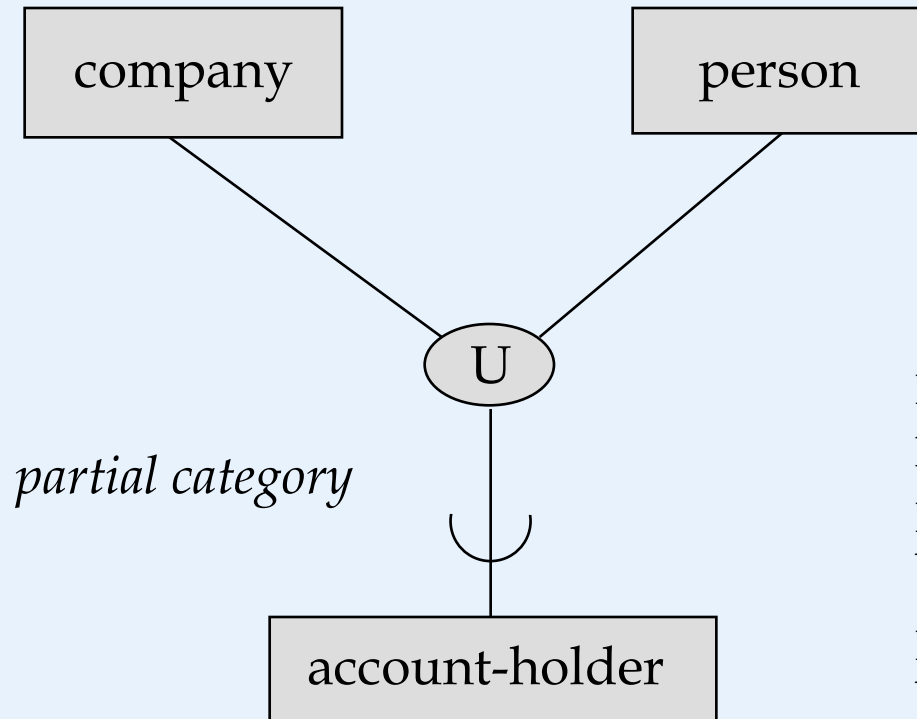
Categories

- ☞ Models a single class/subclass with more than one super class of different entity types



Categories

☞ A category can be either total or partial



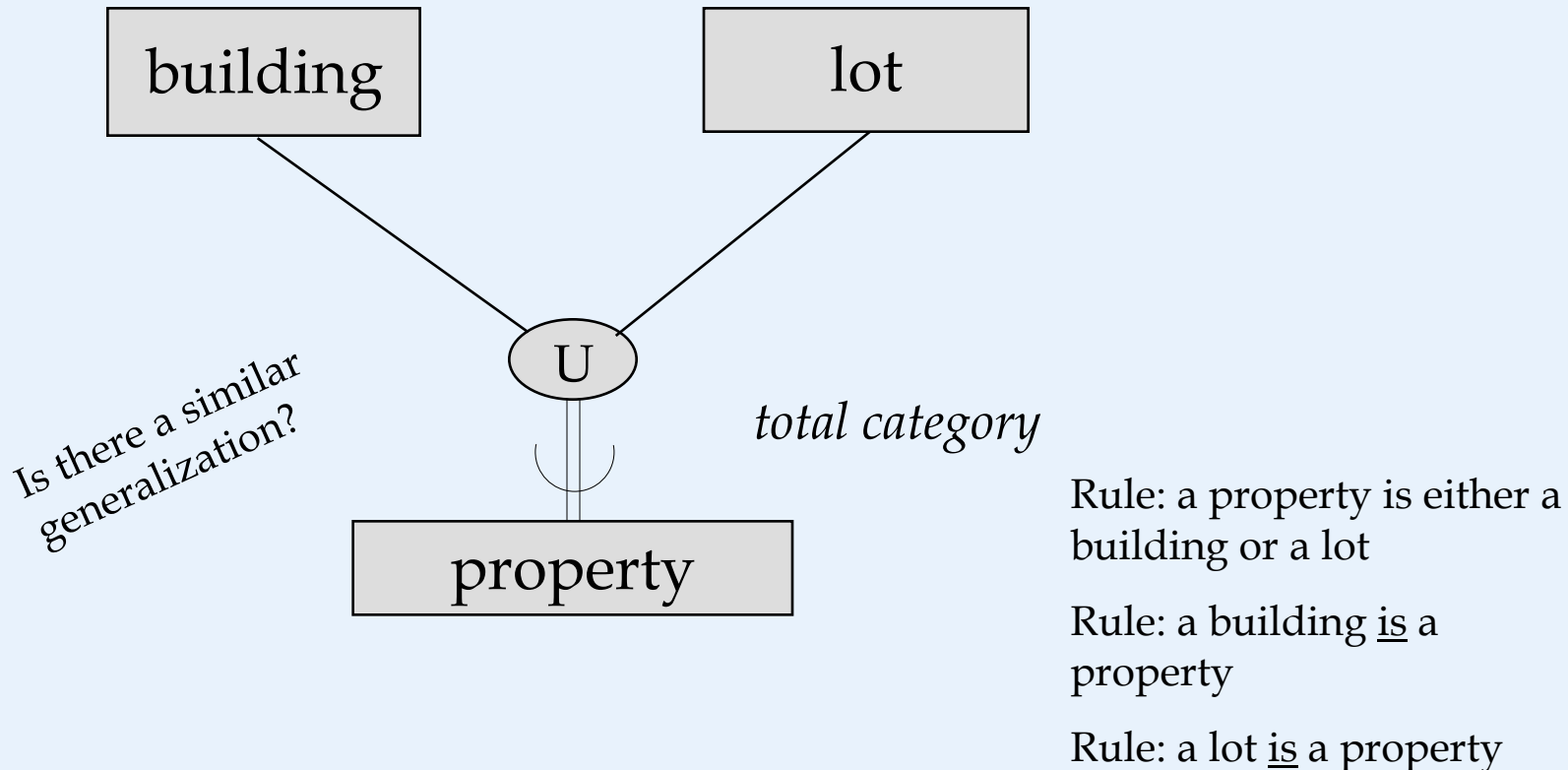
Rule: an account holder is either a person or a company.

Rule: a person may, or may not, be an account owner

Rule: a company may, or may not, be an account holder

Categories

☞ A category can be either total or partial

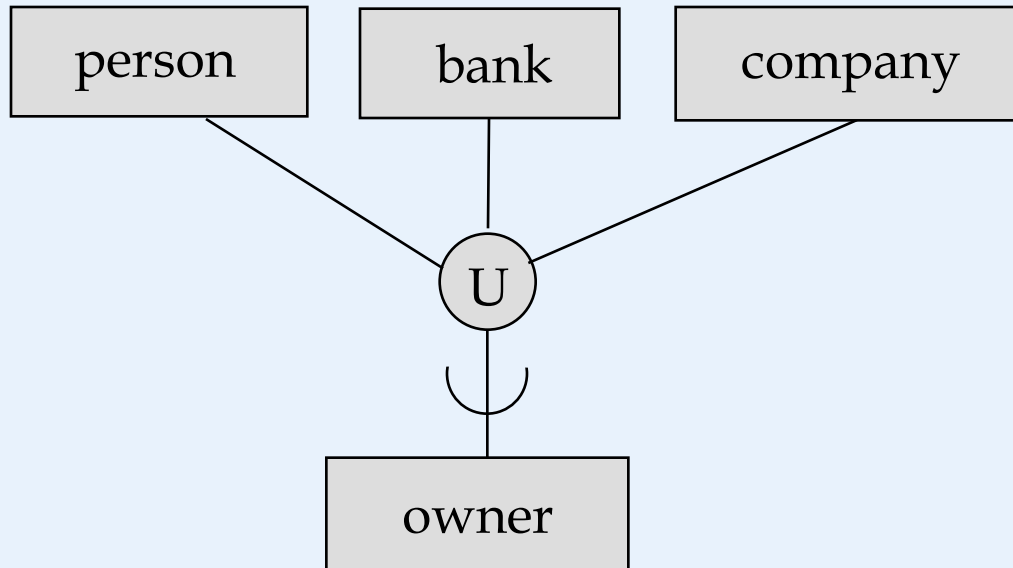


☞ Mapping of Categories

- ☞ Generate a table for each entity type involved
- ☞ Superclasses with different key
- ☞ Specify a new key called surrogate key for the category, which will also be included in the tables for the superclasses as foreign keys

- ☞ Superclasses with the same keys
 - No need of a surrogate key

☞ Categories - Superclasses with different keys



Person (SSN, DrLicNo, Name, Address, Ownerid)

Bank (Bname, BAddress, Ownerid)

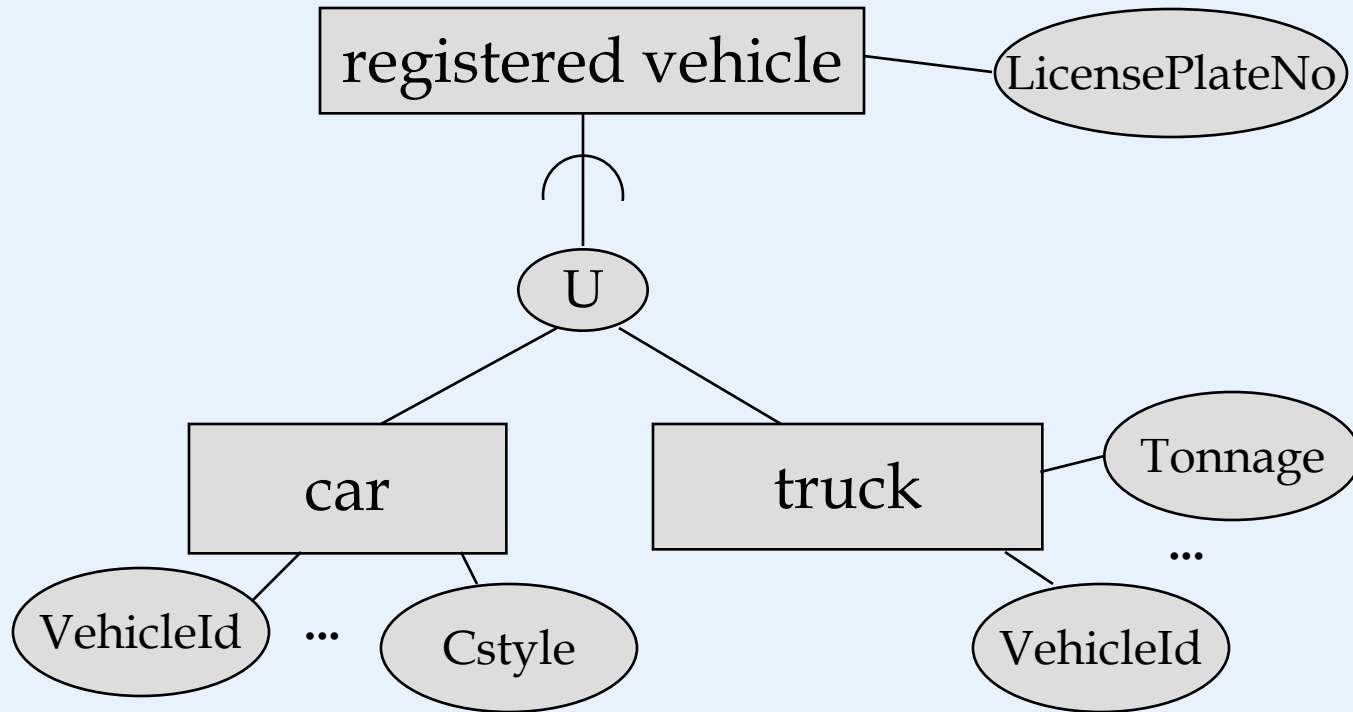
Company (CName, CAddress, Ownerid)

Owner (Ownerid)

← Surrogate key

Note the Foreign Keys

☞ Categories - Superclasses with the same keys



Registered Vehicle (VehicleID, LicensePlateNo,)

Car (VehicleID, Cstyle, CMake, CModel, CYear)

Truck (VehicleID, TMake, TModel, TYear, Tonnage)

Note there are no Foreign Keys