Advanced Databases Course Introduction

Nguyễn Thị Hải Bình
nth.binh@hutech.edu.vn

Contents

- Course objectives.
- What will cover in this course? – Detail syllabus.
- Assessment Methods
- Textbook and References.

Course Objectives

- This course provides students background and advanced knowledge about database.
- Students can apply knowledge in this course to:
  - Analyze, design and construct databases for software applications.
  - Plan, organize, and monitor the processes of system deploying such that the system is ensured to be operated smoothly.
- After this course, students are expected to have the following abilities:
  - Implement systems of software, information, computer networks and information security.
  - Join entrepreneurship teams or R&D teams to research and study knowledge more.

What will cover in this course?

- Lesson 1: Database design and the E-R model – 9 periods
  - The entity relationship (E-R) data model for database design.
  - Development of a relational database design from an E-R design.
- Lesson 2: Relational database design – 9 periods
  - This lesson shows pitfalls in database design.
  - And how to design a database schema systematically in a way that avoids those pitfalls.
- Lesson 3: Application design and development – 9 periods
- Lesson 4: Case study 1 - Analyze and Design relational database for ERP (Enterprise Resource Planning) system – 9 periods
- Lesson 5: Case study 2 - Analyze and Design relational database for LMS (Learning Management System) – 9 periods

Assessment Methods and Rubric

Type	Detail	Maximum score
Midterm/Progress Evaluation • *total score = s1 + s2 + bonus* • *maximum score = 10*	(s1) Attendance (individual)	3
	(s2) Exercises (teamwork)	7
	(bonus) Answering in-class questions (individual) – 0.5 point/correct answer.	3
Final Examination (project-based) • *total score = s1 + s2 + bonus – penalty* • *maximum score = 10*	(s1) Written report (submitted before 20/10/2021)	5
	(s2) Presentation + QA (on 25/10/2021)	5
	(bonus) Teams who give good questions or good reviews on other teams' work will get bonus points - 1 point / question or review.	3
	(penalty) Teams who do not give any discussion on other teams' work will be minus 1 points from the total scores.	1

- Final score = (midterm + final project) / 2
- Project topics and guides will be announced in the second week.
- The number of members in each team: 2 – 4 students

Textbook and References

- Avi Silberschatz, Henry F. Korth, S. Sudarshan (2010), Database System Concepts, 7E, McGraw-Hill. ISBN 0-07-352332-1. (Chapter 6-7)
- Lecture notes on Google Classroom
- Database design on course on Datacamp.com
- Tool:
  - https://app.diagrams.net/
  - https://www.lucidchart.com
  - https://www.smartdraw.com/
  - https://creately.com/
  - MS Visio

Class Rules

- All classes are recorded.
- Ask whenever you want.
- Raise your hand to talk.
- Always turn on your camera.
- Mute your mic when you are not talking.
- Be on time, dress properly, do not eat.
- **Enjoy learning.**

Database Design and the Entity Relationship (E-R) Model

Nguyễn Thị Hải Bình
nth.binh@hutech.edu.vn

Contents

- Overview of Database Design
  - Basic steps
- Entity – relationship data model (E-R):
  - Means of identifying entities to be represented in the database and how those entities are related.
  - E-R Diagram

What is Database Design?

- Is a complex task.
- Determines how data is logically stored.
    - This is crucial because it affects how the database will be queried.
- Uses **database models**: high-level specifications for database structure.
    - Most popular: relational model → which is used to make relational databases.
    - Some other options: NoSQL models, object-oriented model, network model.
- Uses **schemas**: blueprint (or implementation) of the database
    - Defines tables, fields, relationships, indexes, and views a database will have.
    - When inserting data in relational databases, schemas must be respected.

Data Modelling

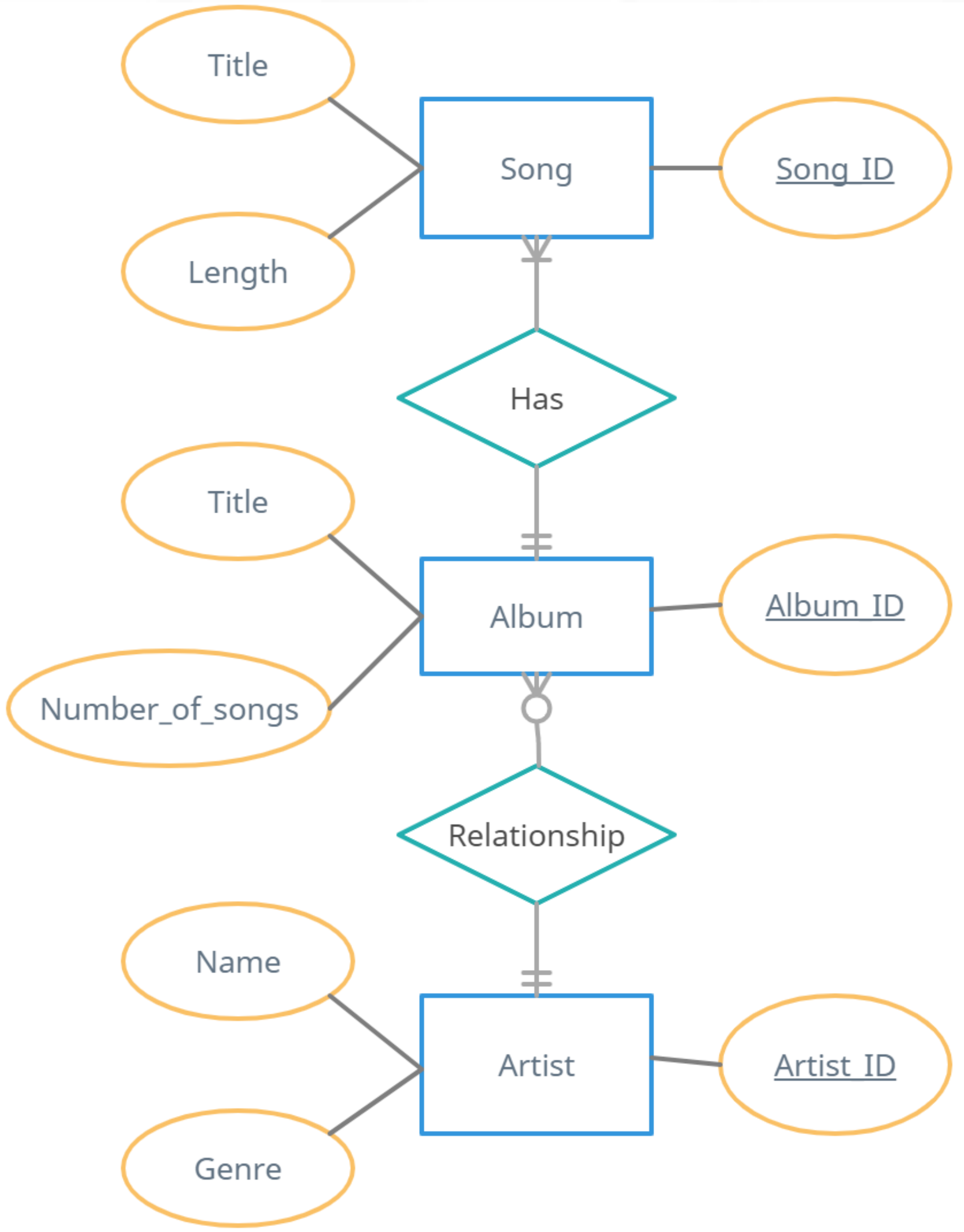
First step in database design.

Process of creating a data model for the data to be stored.

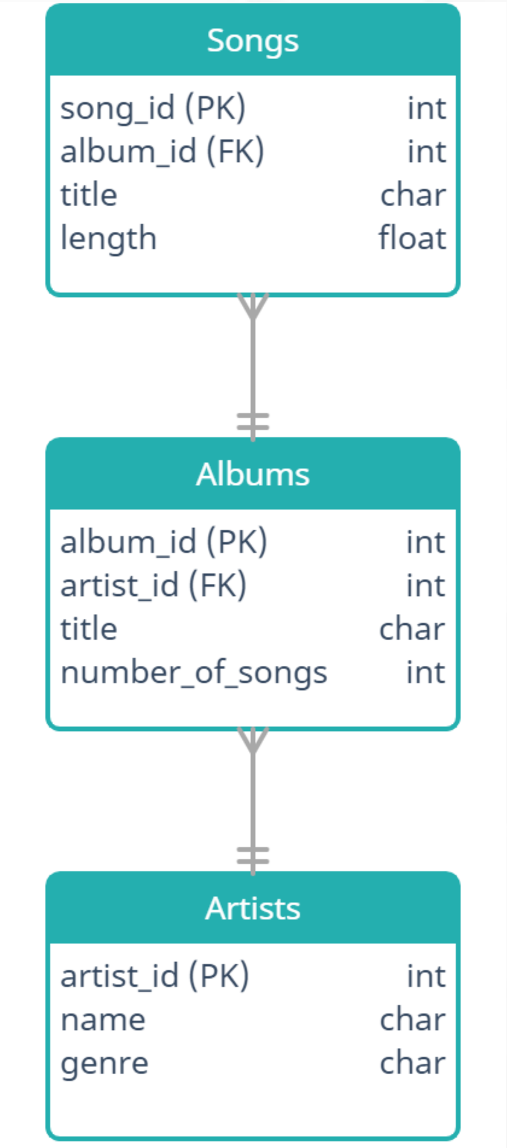
Three levels in data modelling:

1. Conceptual data model: describes what the database contains, such as entities, relationships, and attributes.
    - Tools: data structure diagrams, e.g., entity-relational diagrams and UML diagrams.
2. Logical data model: decides how entities and relationships map to tables.
    - Tools: database models and schemas, e.g., relation model and star schema.
3. Physical data model: how data will be physically stored.
    - Tools: partitions, CPUs, indexes, backup systems, and tablespaces.

Simplified Example



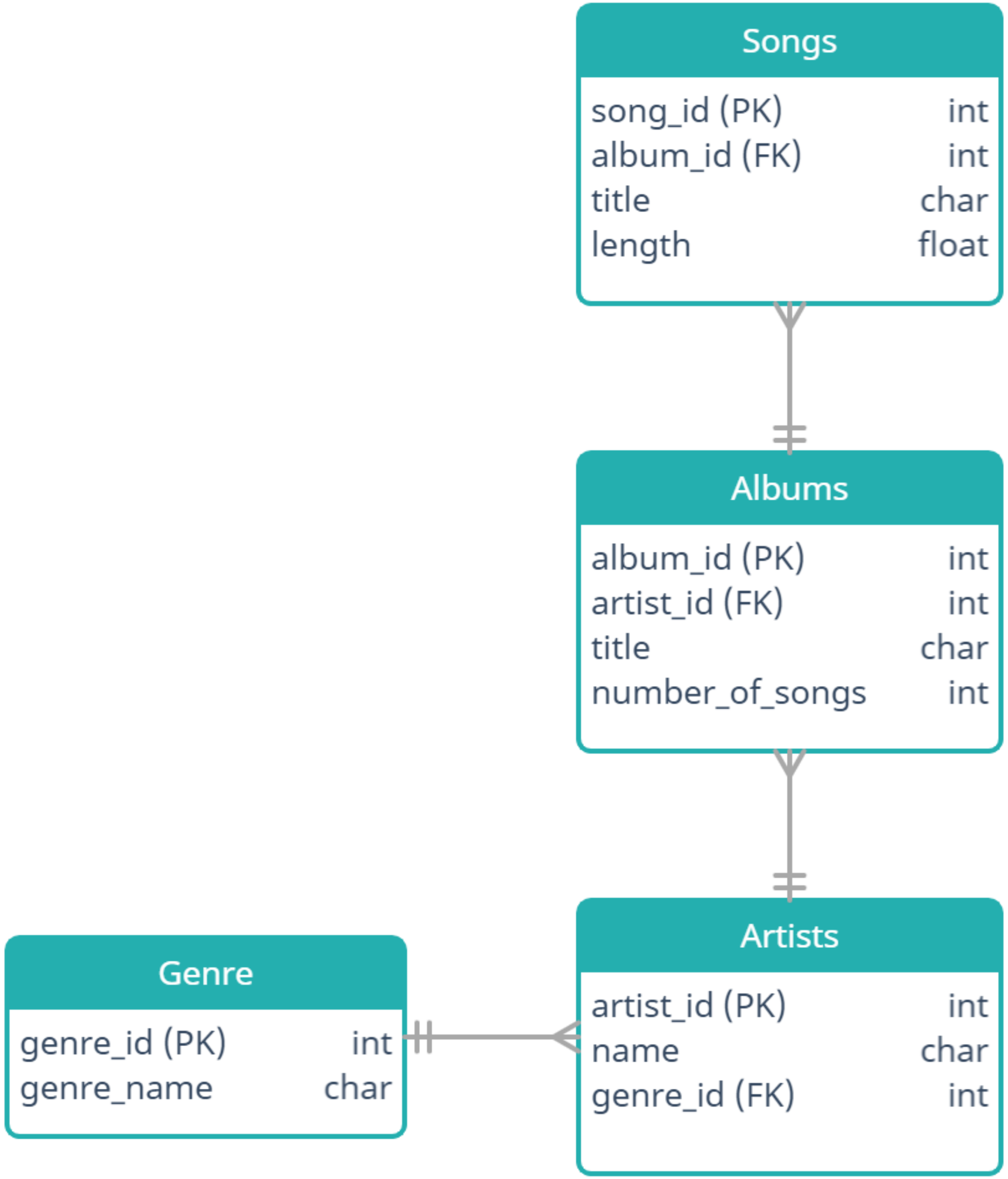
Conceptual – ER Diagram



Logical - Schema

Other database design options

Songs	
song_id (PK)	int
song_title	char
length	float
album_title	char
number_of_songs	int
artist_name	char
genre	char



Exercise:

- We learned about three different levels of data models: conceptual, logical, and physical.
- Each of these cards hold a tool or concept that fits into a certain type of data model. Place the cards in the correct category.

1. File structure of data storage
2. Relational model
3. Gather business requirement
4. Determine tables and columns
5. Entities, attributes, and relationship

Entity Relationship (ER) Model

E-R Model – Conceptual Data Model

- Defined as a logical representation of the data for a business process.
- The entity relationship model is expressed in terms of:
  - Entity set,
  - Relationships among those entities,
  - Attributes of both the entities and their relationships.
- An entity relationship diagram (ERD) is a graphical representation of an entity relationship model.

Entity Sets

- **Entity:**
  - Real world “things” or “objects”, distinguishable from other objects.
  - Described using a set of attributes.
  - Uniquely identified by the values of some set of attributes.
  - Example: each student in a university

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Entity Sets

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

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Example of Entity Sets - instructor and student

76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Attribute Types

- Attributes are properties that characterize or describe entities or relationships.
- Simple and composite attributes.
  - Simple attributes:
    - Cannot be divided into subparts.
    - Example: student_id
  - Composite attributes:
    - Can be divided into subparts (i.e., other attributes).
    - Example: name, address

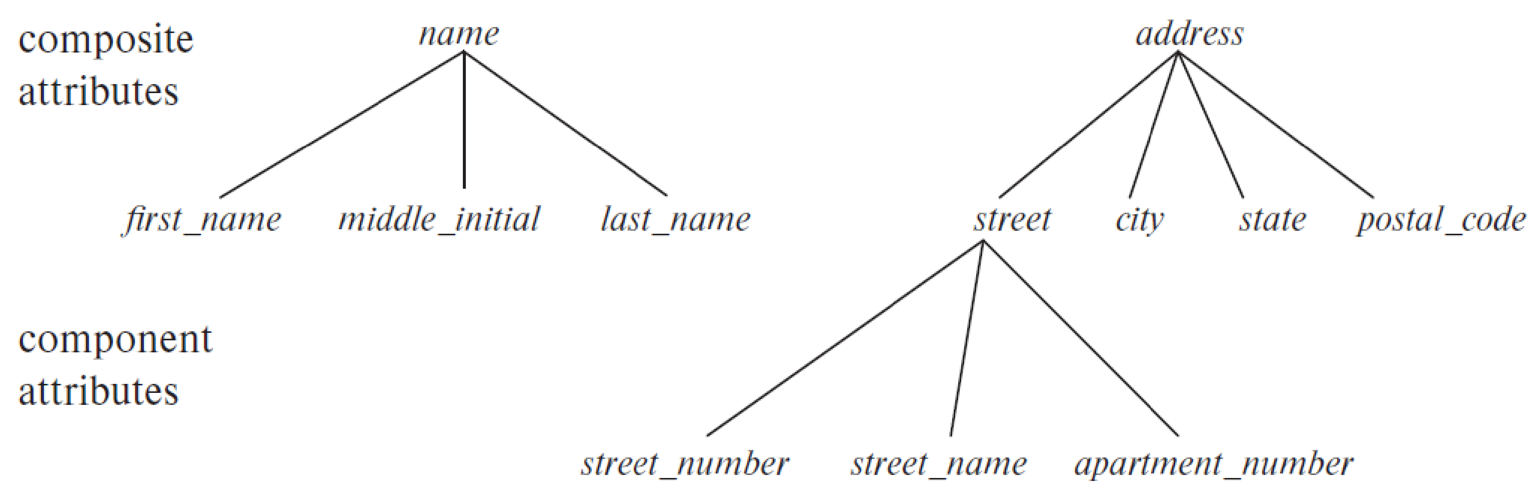


Figure 6.7 Composite attributes instructor *name* and *address*.

Attribute Types

- Single-valued and multivalued attributes
  - Single-valued attributes:
    - The attributes have a single value for a particular entity.
  - Multivalued attributes:
    - The attributes have a set of values for a specific entity.
- Derived attributes
  - Can be computed from other attributes
  - Example: *age*, given *date_of_birth*

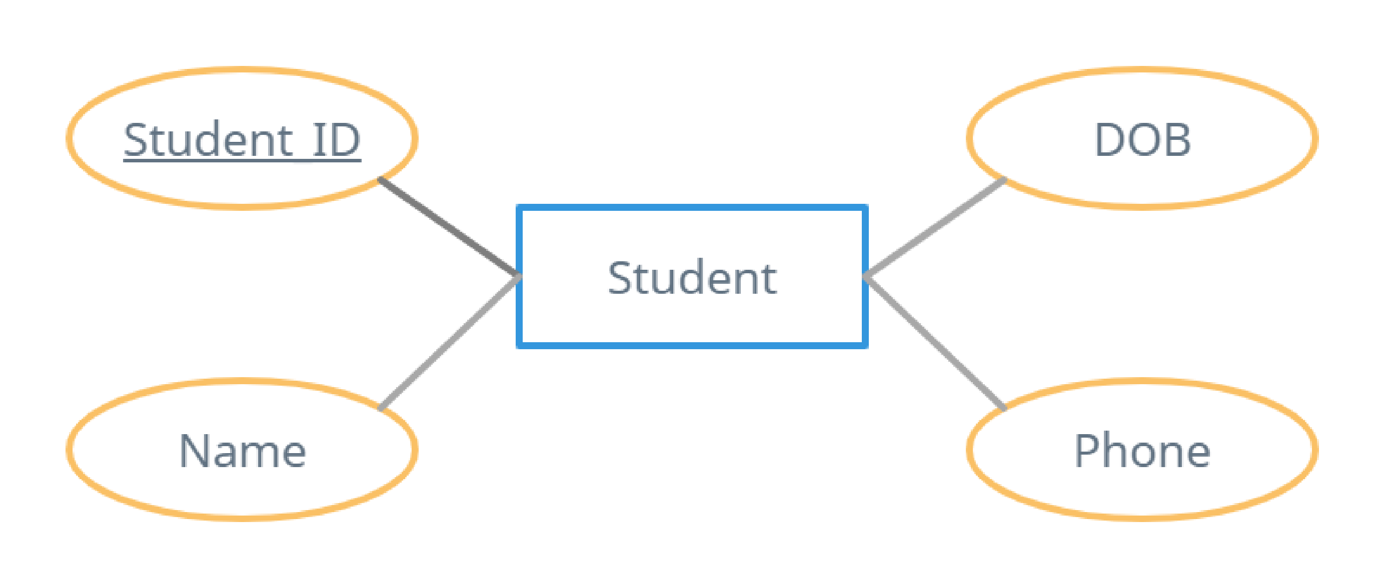
Single-valued attribute

Multivalued attribute

ID	name	DOB	phone_number
76766	Crick	7/7/1978	
45565	Katz	1/6/1980	0109839489, 0119886453
10101	Srinivasan	1/1/1992	0102348484
98345	Kim	3/6/1960	0105827091
76543	Singh	5/5/1977	03287821, 03212822, 01127382822

Represent Entity sets in ER Diagram

- Entity sets can be represented graphically as follows:
  - Rectangles represent entity sets.
  - Ovals represent attributes.
  - Underline indicates primary key attributes.



Represent Entity sets in ER Diagram - Example

- Entity set: **Department**
  - Attribute: **ID, name, location**

- Entity set: **Instructor**
  - Attribute: **ID, name, DOB**

In Summary …

- Entity = “thing” or “object”
- Entity set = collection of similar entities.
- Attribute = property of (the entities of) an entity set.
  - Attribute types:
    - Simple and composite attributes.
    - Single-valued and multivalued attributes
    - Derived attributes.
  - Example: name of an employee, color of a car, balance of an account, location of a house,…

Relationship

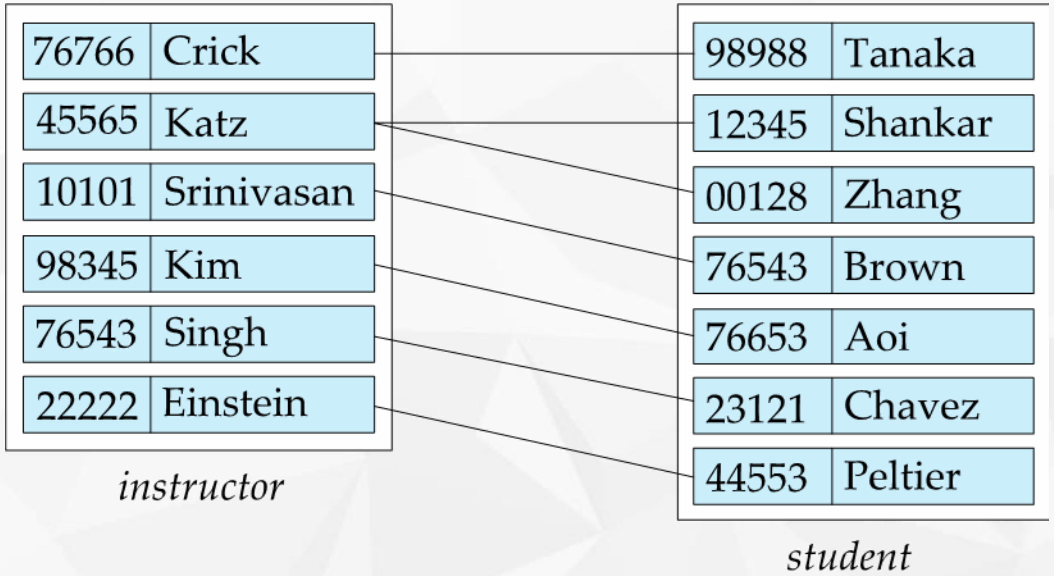
- A **relationship** is an association among several entities.
- Example of relationship:

00128 (Zhang)	advice	98345 (Kim)
student entity	relationship set	*instructor* entity

This relationship specifies that Kim is an advisor to student Zhang.

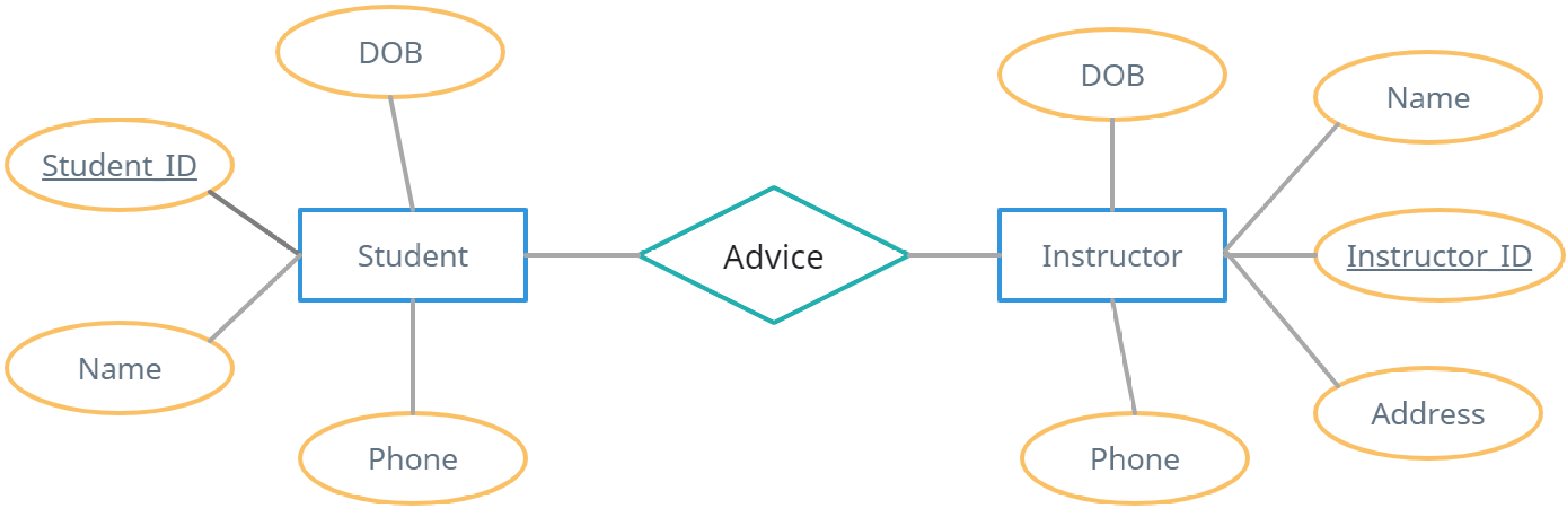
Relationship Sets

- A **relationship set** is a set of relationships of the same type.
- Example of relationship set:
  - Consider two entity sets: *instructor* and *student*.
  - One instructor may be advisors of one or more students → Define the relationship set *advice* to denote the associations between students and instructors who act as their advisors.



Represent Relationship Sets via ER Diagrams

- Diamonds represent relationship sets.



Relationship Sets - Example

- Given two entity sets: *employees* and *departments*

SSN	FName	LName	BDate
111-11-1111	John	Smith	Jan-1-78
222-22-2222	Jane	Doe	Apr-1-76
333-33-3333	Jack	Rabbit	May-4-79

***Employees* entity**

DNum	DName
5	Research
1	Payroll

***Departments* entity**

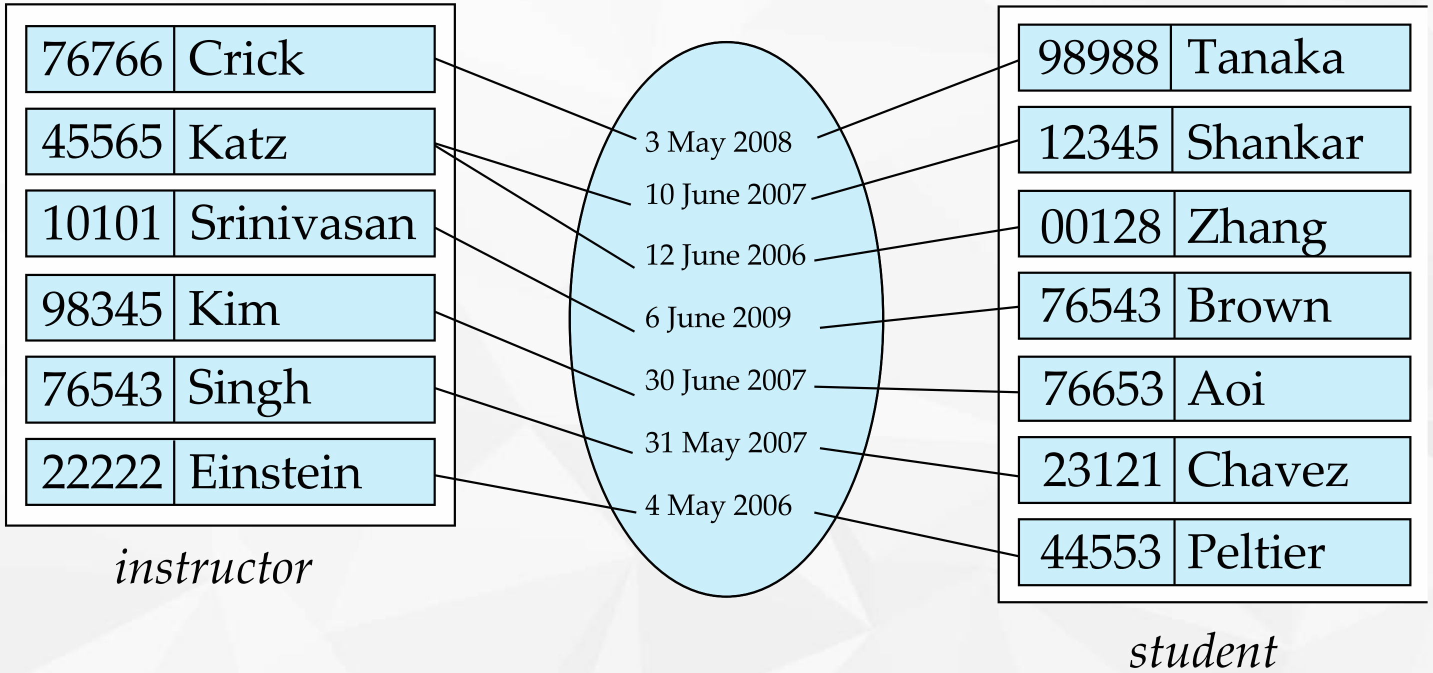
- We know that each employee works for one departments.
- Define the relationship set and draw the simplified ERD.

Relationship Sets - Example

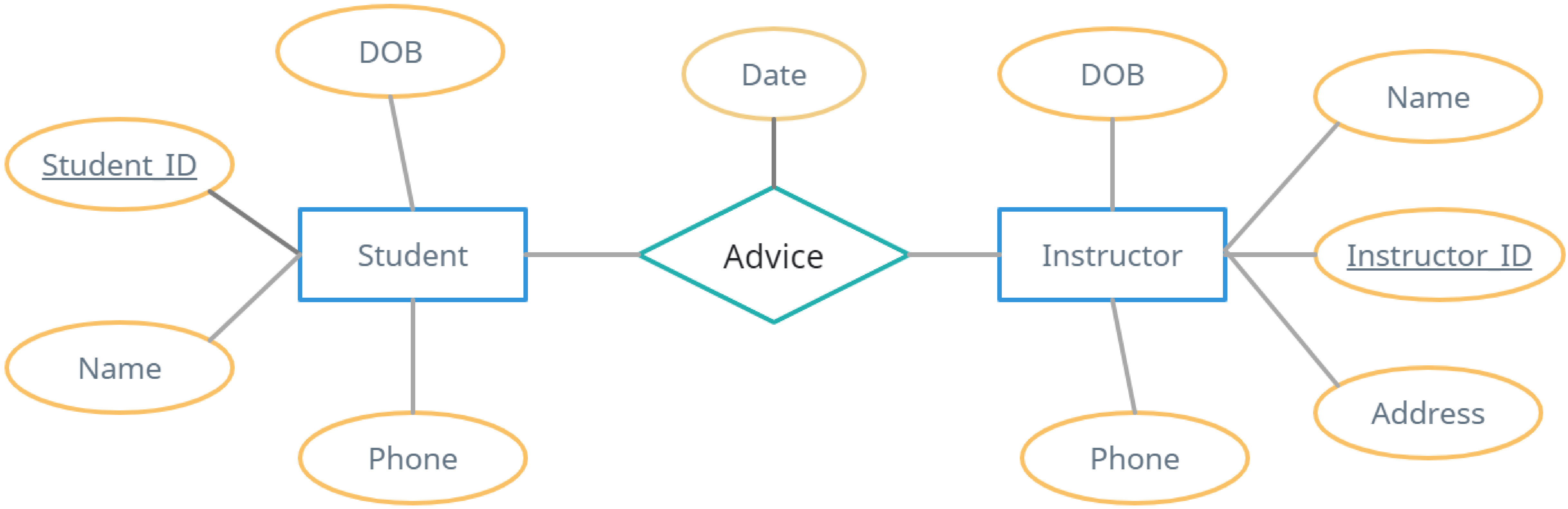
- A university wants to build a database system that stores the following information:
  - ID, name, DOB, address of each student.
  - Code, name, number of credits of each course.
  - Each student take several courses.
- Define entity sets, relationship sets, and draw simplified ERD.

Relationship Sets with Attributes

- An **attribute** can also be associated with a **relationship set.**
- Example:
  - Consider the ***advice*** relationship set between ***instructor*** entity set and ***student*** entity set.
  - The ***advice*** relationship may have the attribute ***date*** which tracks when the instructor become the advisor of a student.



Relationship Sets with Attributes



Relationship Sets with Attributes - Example

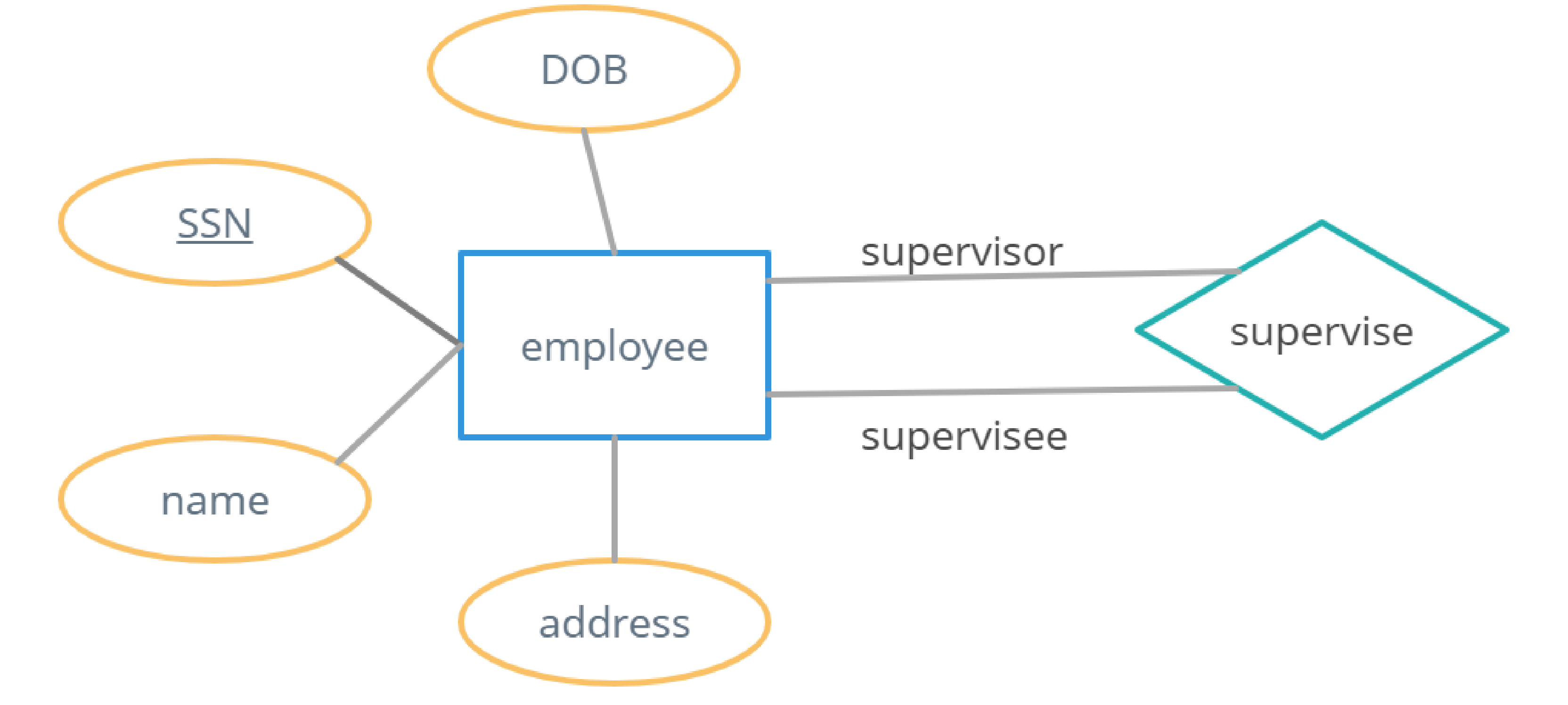
- A company wants to build a database system that stores the following information:
  - ID, name, DOB, address of each employee.
  - Code, name, and location of each branch.
  - Each employee works at a specific branch.
  - The database must record when the employee starts working at the current branch.
- Define entity sets, relationship sets, and draw simplified ERD.

Relationship Sets with Attributes - Example

- Design a database to store the following information:
  - ID, name, DOB, phone number of employees.
  - ID, name, start date, budget of projects.
  - When an employee starts and stops working on a project.
- Define entity sets, relationship sets, and draw simplified ERD.

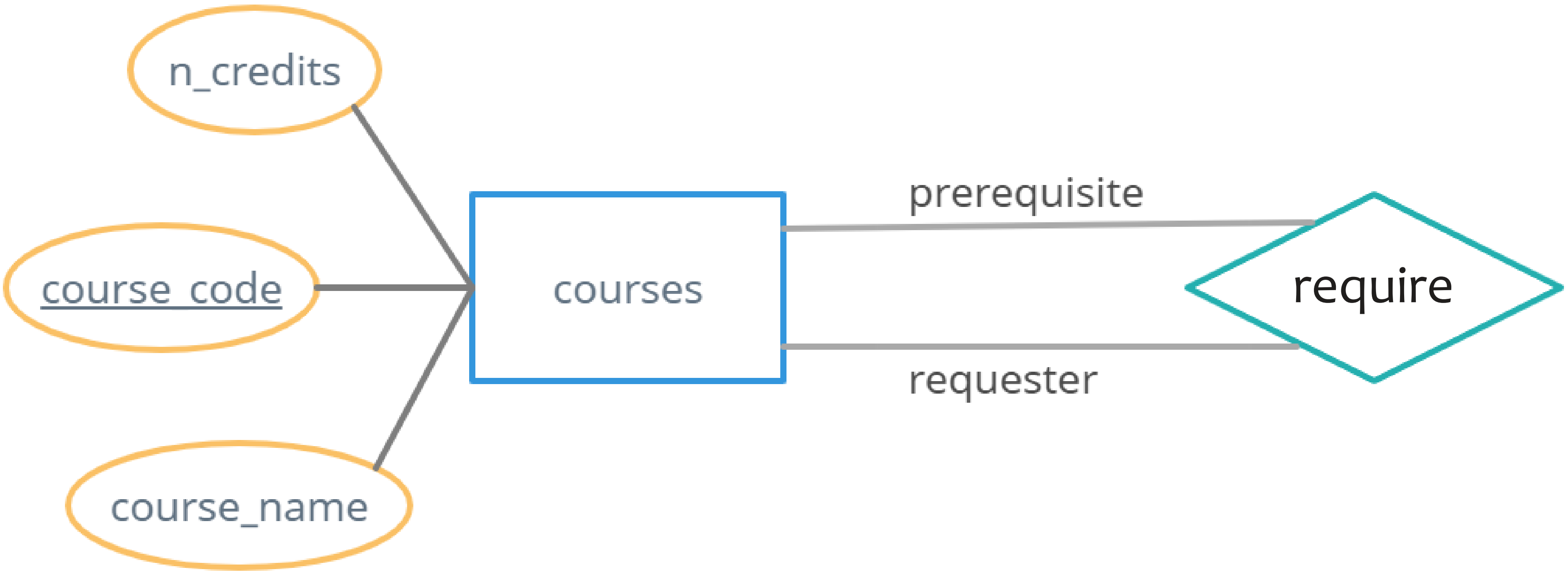
Unary or Recursive Relationship Set

- **Recursive relationship set** happens when the entity sets of a relationship set are not distinct.
- In this case, each occurrence of an entity set plays a "role" in the relationship → must specify role names to know how an entity participates in a relationship instance.



Unary or Recursive Relationship Set - Example

- Consider the entity set ***course*** that records information about all the courses offered in the university.
- One course shall be a prerequisite for another course.

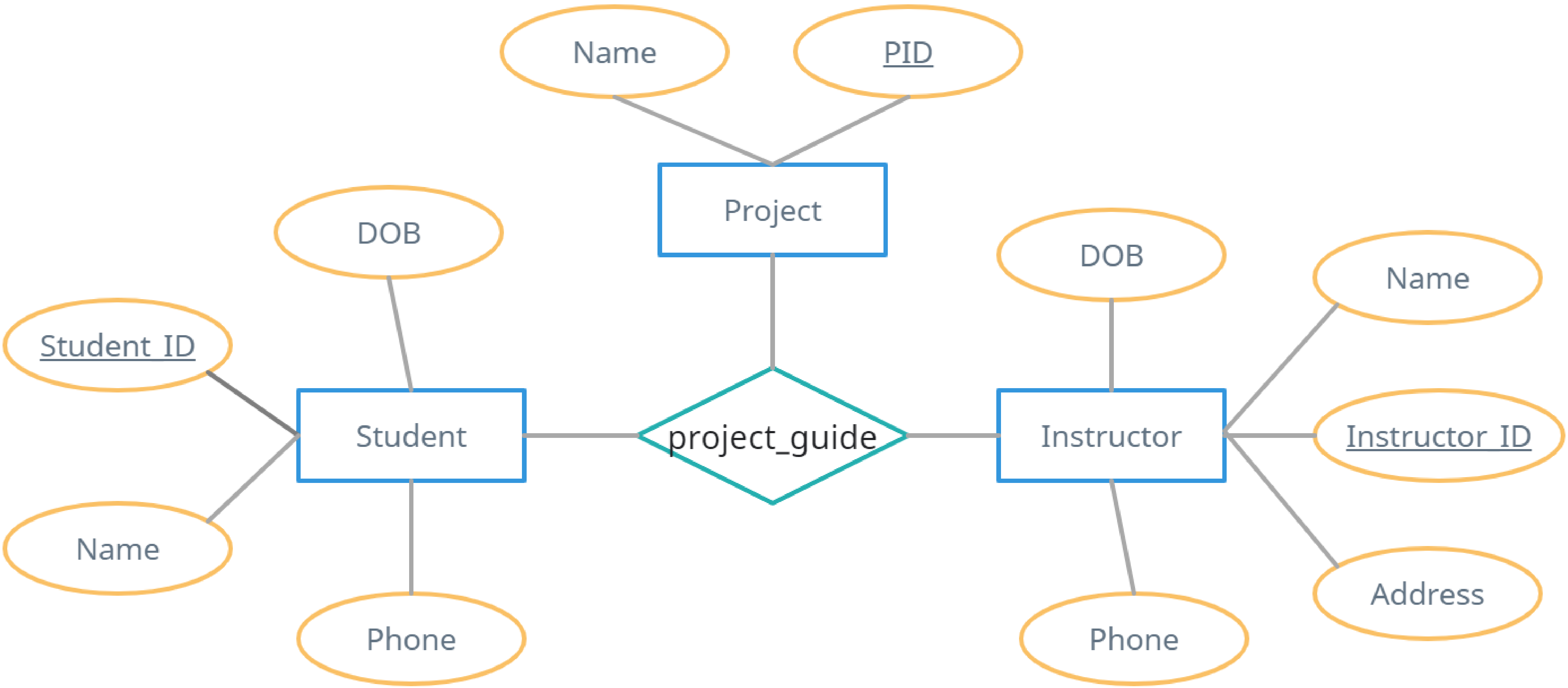

n_credits
course_code
course_name
courses
prerequisite
requester
require

Degree of a Relationship Set

- The number of entity sets that participate in a relationship set is the degree of the relationship set.
- Binary relationship:
  - Involve two entity sets (or degree two).
  - Most relationship sets in a database system are binary.
- Non-binary relationship sets:
  - Involve more than two entity sets.
  - Example: Ternary Relationship.

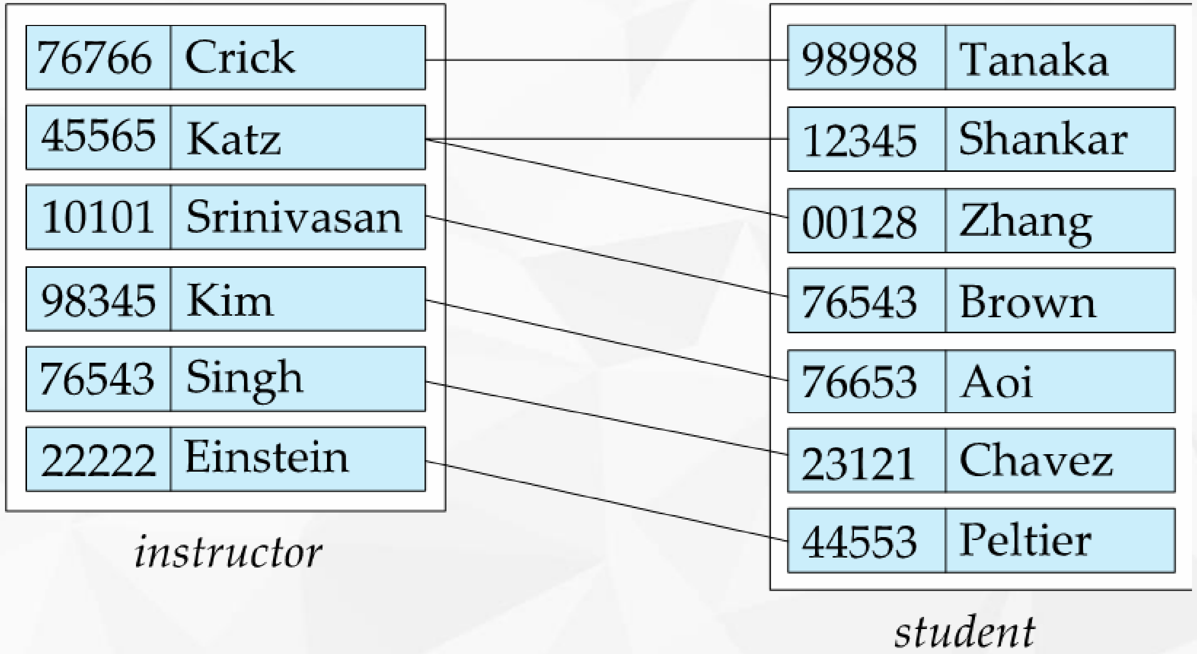
Example: Ternary Relationship

An instance of *project_guide* indicates that a particular student is guided by a particular instructor on a particular project.



Mapping Cardinalities

- Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.
- Mapping cardinalities are most useful in describing binary relationship sets.



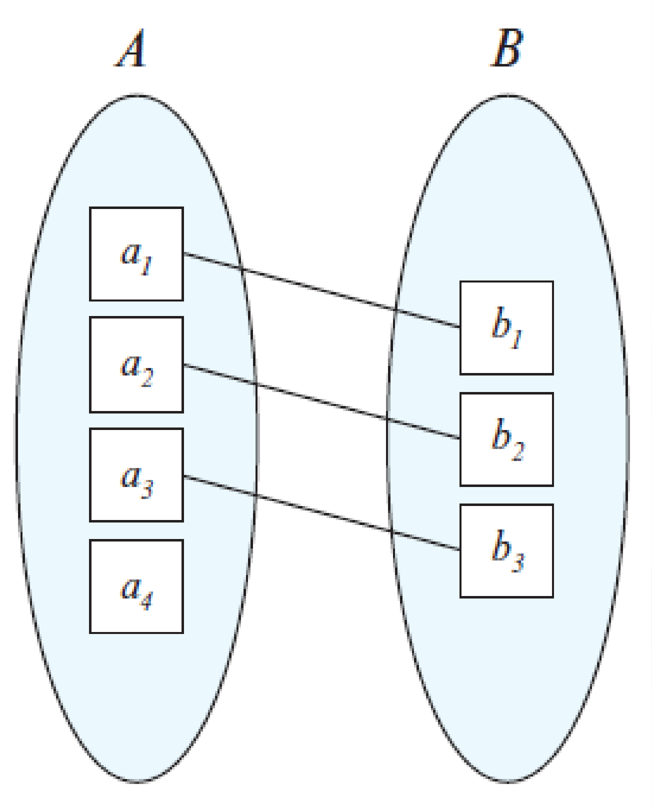
One instructor supervised one or several students.
One student is supervised by one instructor.

Mapping Cardinality Types

- For a binary relationship set, the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinality Types - One to one

- An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



NID	Name	DOB
1111111	A	1/1/2000
1111112	B	3/2/1999
1111113	C	5/5/2005
1111114	D	3/2/1999
1111115	E	1/12/1998

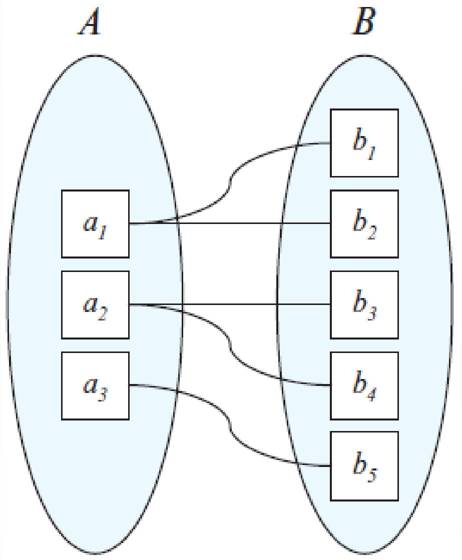
Person entity set

NO	Issue_Date	Expiry_Date
ABA9875413	3/9/2020	3/9/2030
J12393496	5/1/2021	5/1/2031
KF0192332C	1/6/2018	1/6/2023

Passport entity set

Mapping Cardinality Types – One to many

- An entity in A is associated with any number (zero or more) of entities in B.
- An entity in B, however, can be associated with at most one entity in A.



customer_ID	first_name	last_name
001	Jane	Doe
002	John	Doe
003	Jane	Smith
004	John	Smith
005	Jane	Jones

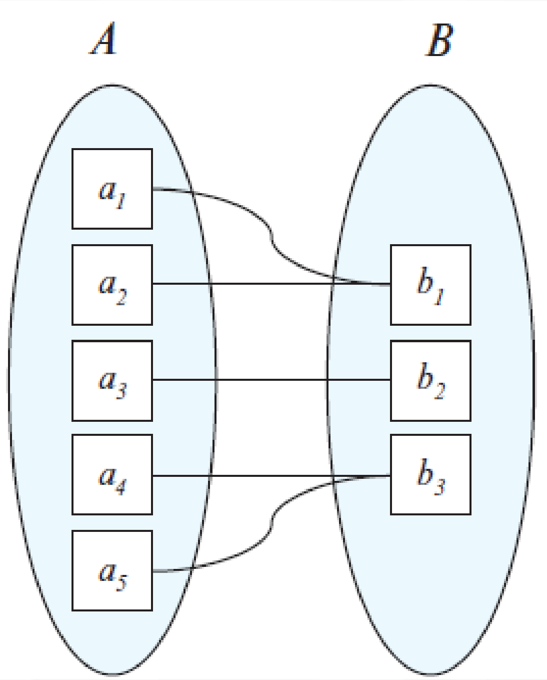
Customer entity set

order_ID	order_date	order_total
1001	10/10/2009	250.85
1002	21/2/2010	125.89
1003	15/11/2009	1567.99
1004	22/11/2009	180.92
1005	15/12/2009	656.00
1006	22/11/2009	25.00
1007	8/10/2009	85.00
1008	29/12/2009	109.12

Order entity set

Mapping Cardinality Types – Many to one

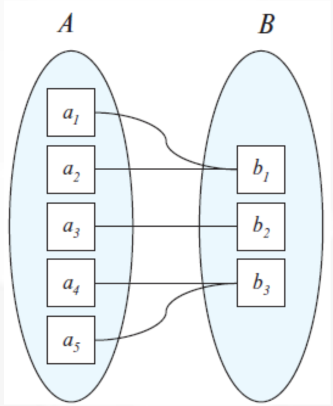
- An entity in A is associated with at most one entity in B.
- An entity in B can be associated with any number (zero or more) of entities in A.



course_id	title	credits
BIO101	Introduction to Biology	4
BIO301	Genetics	4
BIO399	Computational Biology	3
CS101	Introduction to computer science	4
CS190	Game design	4
CS315	Robotics	3
CS319	Image Processing	3
CS347	Database system concepts	3
EE181	Introduction to digital systems	3

dept_id	dept_name	building
D001	Biology	A
D002	Computer Science	T
D003	Electric Engineering	T

Many to one

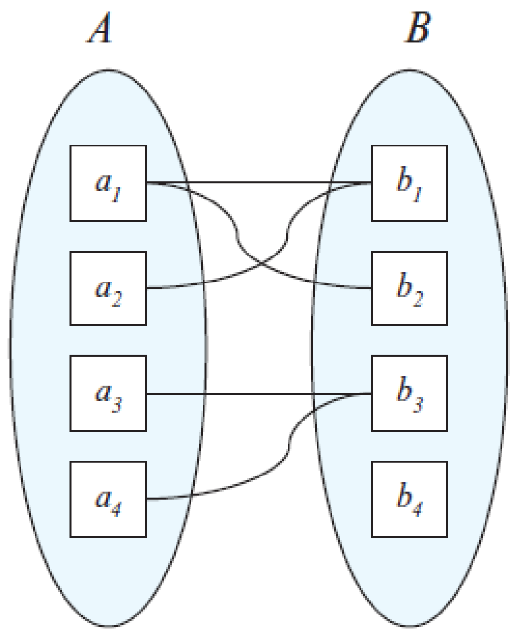


course_id	title	credits
BIO101	Introduction to Biology	4
BIO301	Genetics	4
BIO399	Computational Biology	3
CS101	Introduction to computer science	4
CS190	Game design	4
CS315	Robotics	3
CS319	Image Processing	3
CS347	Database system concepts	3
EE181	Introduction to digital systems	3
FIN201	Investment banking	3
HIS351	World history	3
MU199	Music production	3
PHY101	Physical principles	4

dept_id	dept_name	building
D001	Biology	A
D002	Computer Science	T
D003	Electric Engineering	T
D004	Finance	P
D005	History	P
D006	Music	P
D007	Physics	A

Mapping Cardinality Types – Many to many

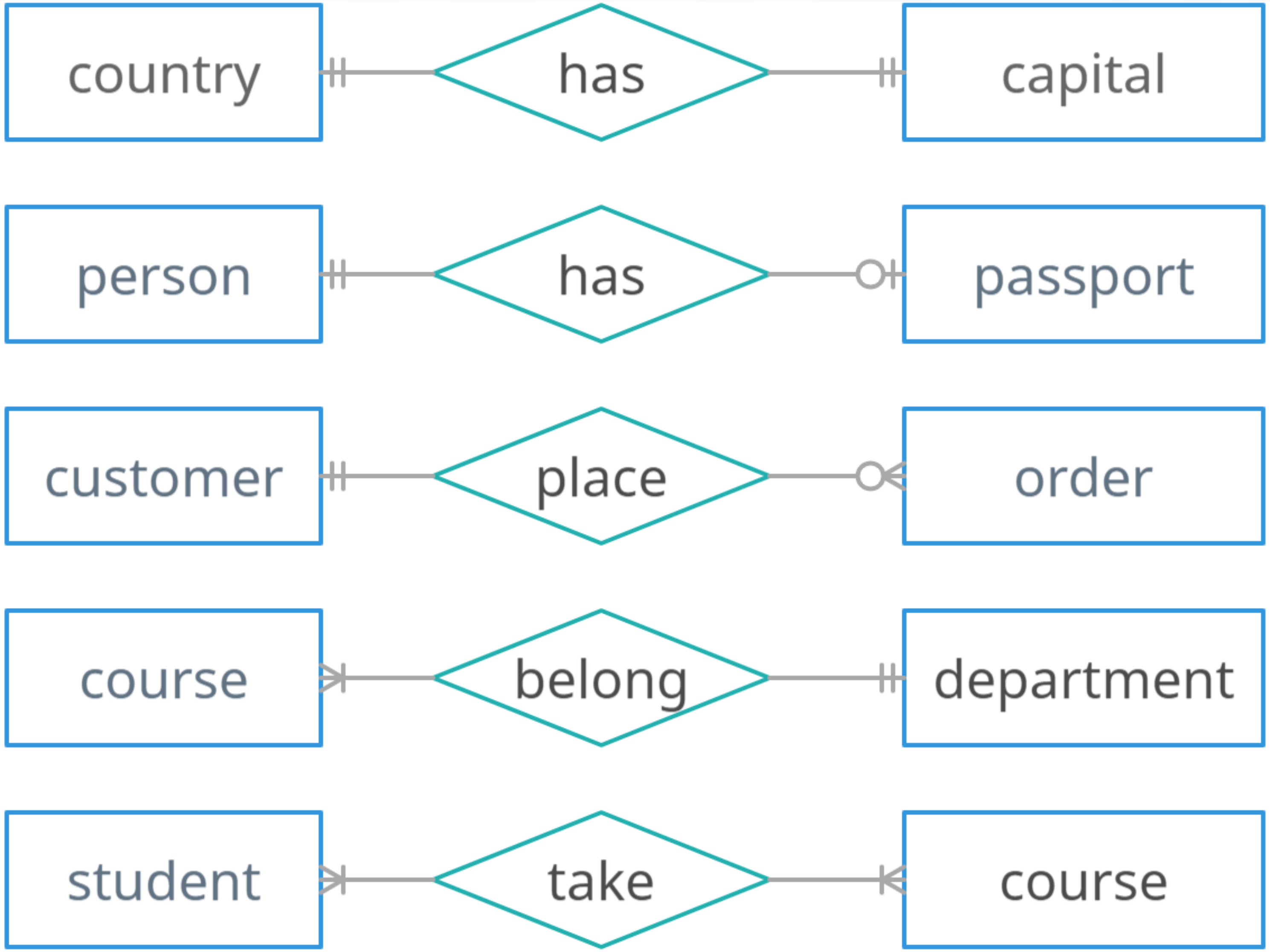
- An entity in A is associated with any number (zero or more) of entities in B.
- An entity in B is associated with any number (zero or more) of entities in A.



SID	name	total_credit
00128	Zhang	102
12345	Shankar	32
19991	Brandt	80
23121	Chavez	110
44553	Peltier	56

course_id	title	credits
BIO101	Introduction to Biology	4
BIO301	Genetics	4
BIO399	Computational Biology	3
CS101	Introduction to computer science	4
CS190	Game design	4
CS315	Robotics	3
CS319	Image Processing	3
CS347	Database system concepts	3

Represent Cardinality Constraints in ER Diagram



A country has one capital.
A capital belongs to one country.

A person has zero or one passport.
A passport belongs to one person.

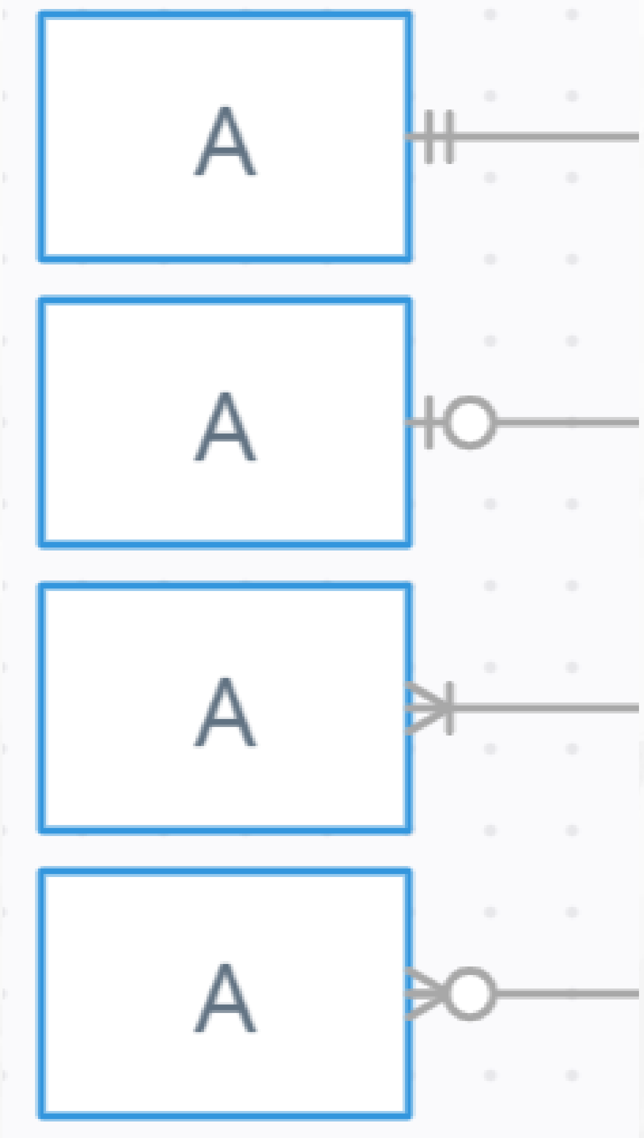
A customer places zero or many orders.
An order is placed by one customer.

A course belongs to one department.
A department has one or many course.

A student takes one or more course.
A course is taken by one or more students.

*** Entities as shown without attributes for simplicity.*

Represent Cardinality Constraints in ER Diagram



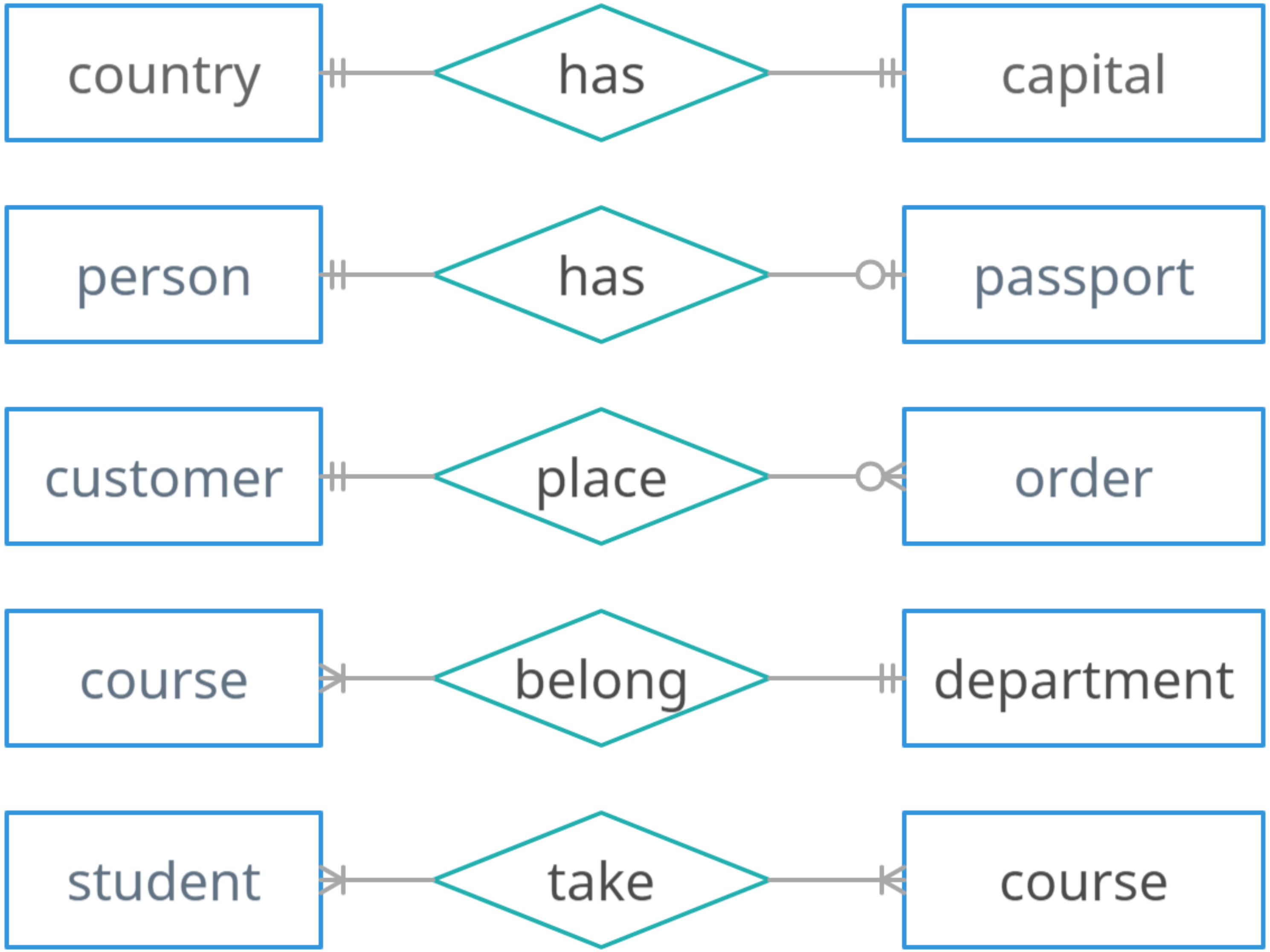
One and only one

Zero or one

One or many

Zero or one or many

Exercise: Identify mapping cardinality type

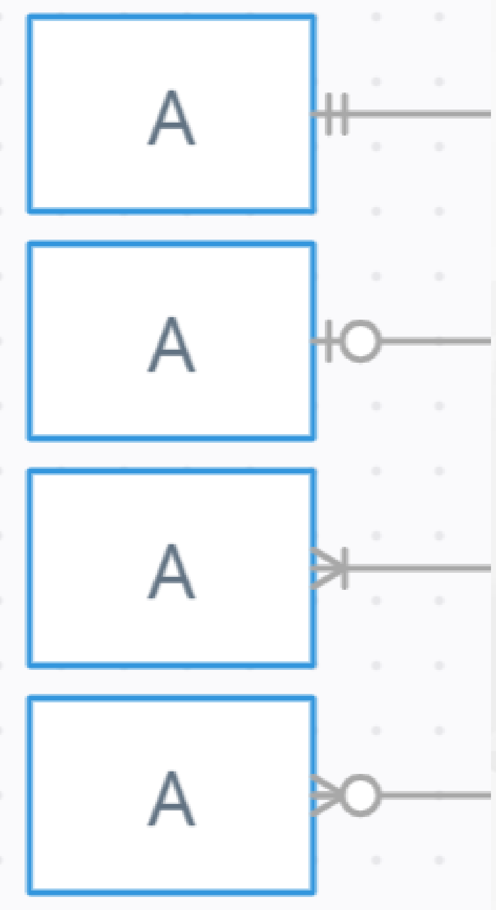


In summary ...

- Relationship = an association among several entities.
- Relationship set = a set of relationships of the same type.
- Recursive relationship set happens when the entity sets of a relationship set are not distinct.
- Binary relationship set: involve two entity sets.
- Non-binary relationship set: involve more than two entity sets.
- Relationship sets may associate with attributes.

In summary ...

- Mapping cardinalities (cardinality ratios) express the number of entities to which another entity can be associated via a relationship set.
  - One to one
  - One to many
  - Many to one
  - Many to many



One and only one

Zero or one

One or many

Zero or one or many

Primary Key

- Primary keys provide a way to specify how entities and relations are distinguished.
- This class will consider:
  - Primary key for Entity sets
  - Primary key for Relationship sets
  - Weak entity sets

Primary key for Entity Sets

- A key for an entity is a set of attributes that can distinguish entities from each other.
- The concepts of superkey, candidate key, and primary key are applicable to entity sets just as they are applicable to relation schemas.
- Example: Consider *student* entity set
  - Attributes: student_id, first_name, last_name, date_of_birth, phone_number, address.
  - Key: student_id

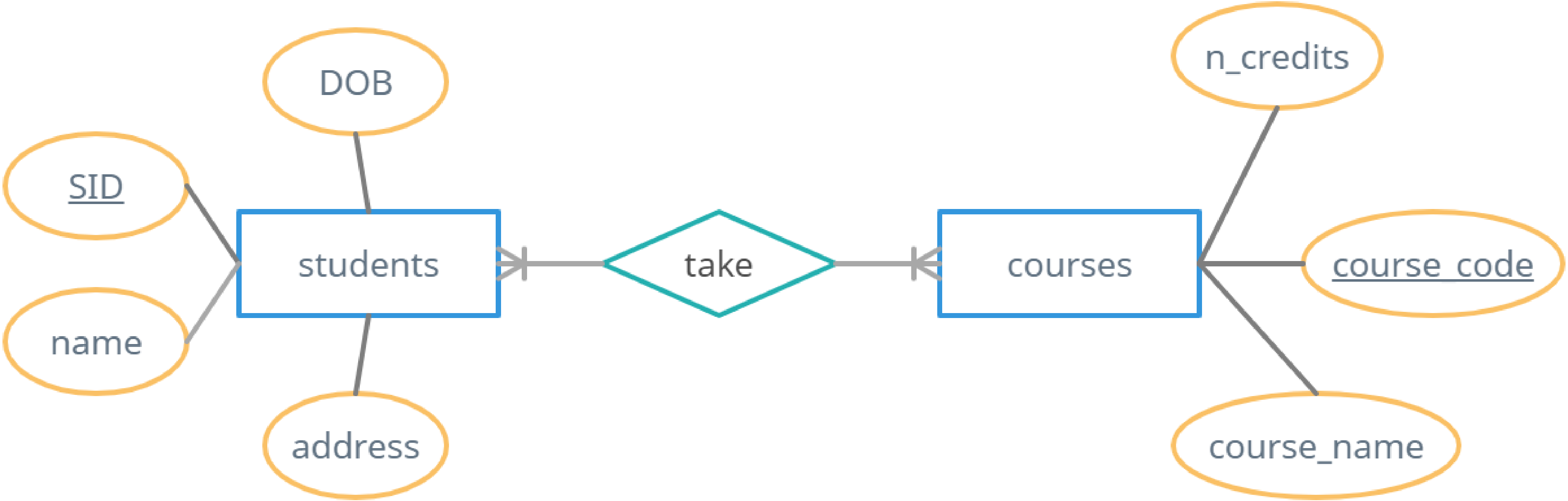
Primary Key for Relationship Sets

- Let R be a relationship set involving entity sets E1, E2, .. En
- In general,
  - The primary key for R is consists of the union of the primary keys of entity sets E1, E2, ..En
  - If the relationship set R has attributes a1, a2, .., am associated with it, then the primary key of R also includes the attributes a1, a2, .., am
- The choice of the primary key for a relationship set depends on the mapping cardinality of the relationship set.

Choice of Primary key for Binary Relationship

Many-to-Many relationships:

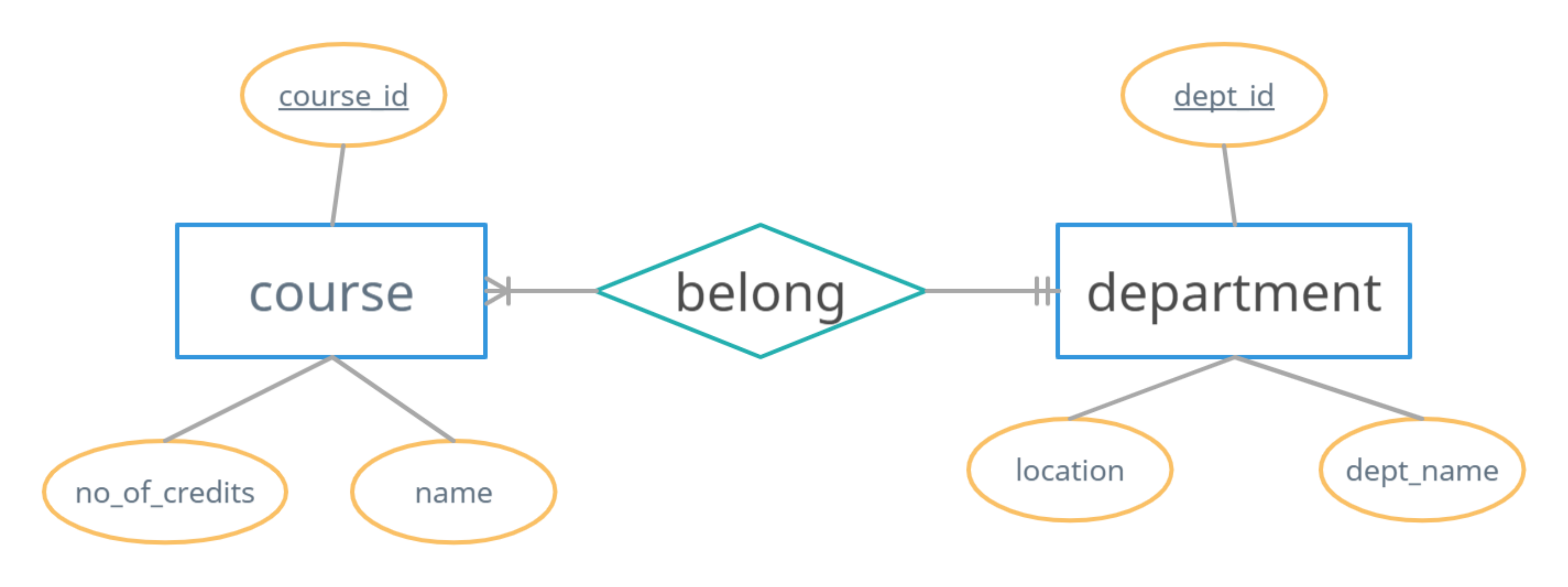
- The union of the primary keys of two entity sets is chosen as the primary key.
- Example: primary key for *take* = {SID, course_code}



Choice of Primary key for Binary Relationship

One-to-Many relationships and Many-to-one relationships

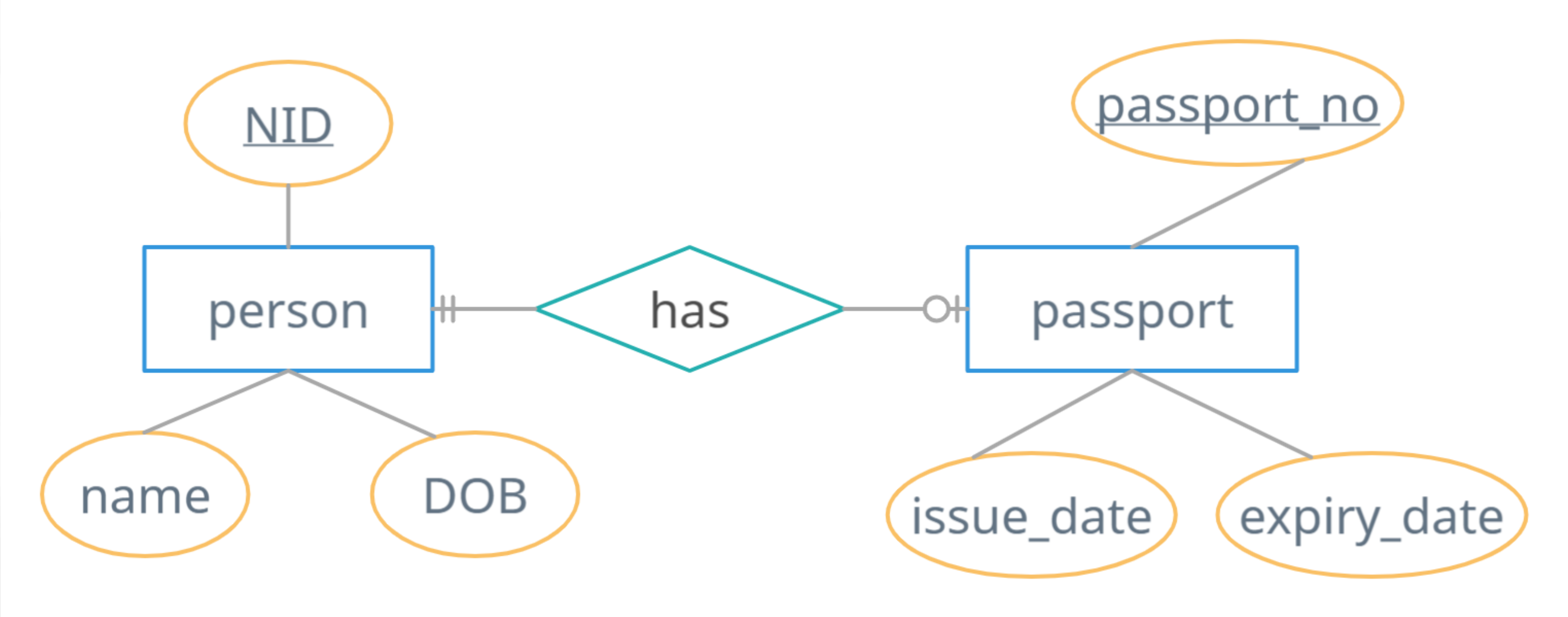
- The primary key of the "many" side is chosen as the primary key.
- Example: primary key for *belong* = {course_id}



Choice of Primary key for Binary Relationship

One-to-one relationships

- The primary key of either one of the participating entity sets is chosen as the primary key.
- Example: primary key for *has* = {NID} or {passport_no}



Weak Entity Sets

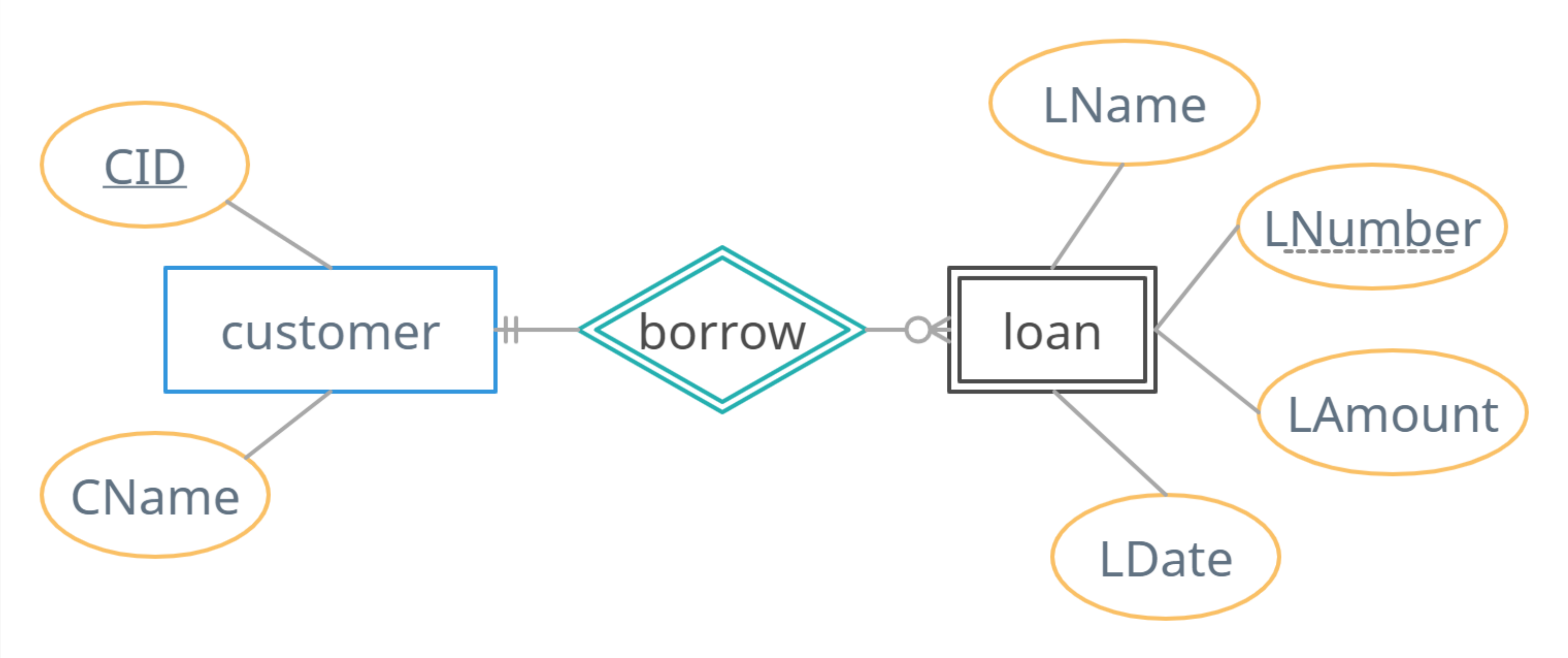
- A weak entity is an entity whose instances cannot exist in the database without the existence of an instance of another entity.

- Any entity that is not weak entity is called a strong entity.
  - Instances of a strong entity can exist in the database independently.

- The weak entity's identifier is a combination of the identifier of the owner entity and the partial key of the weak entity.

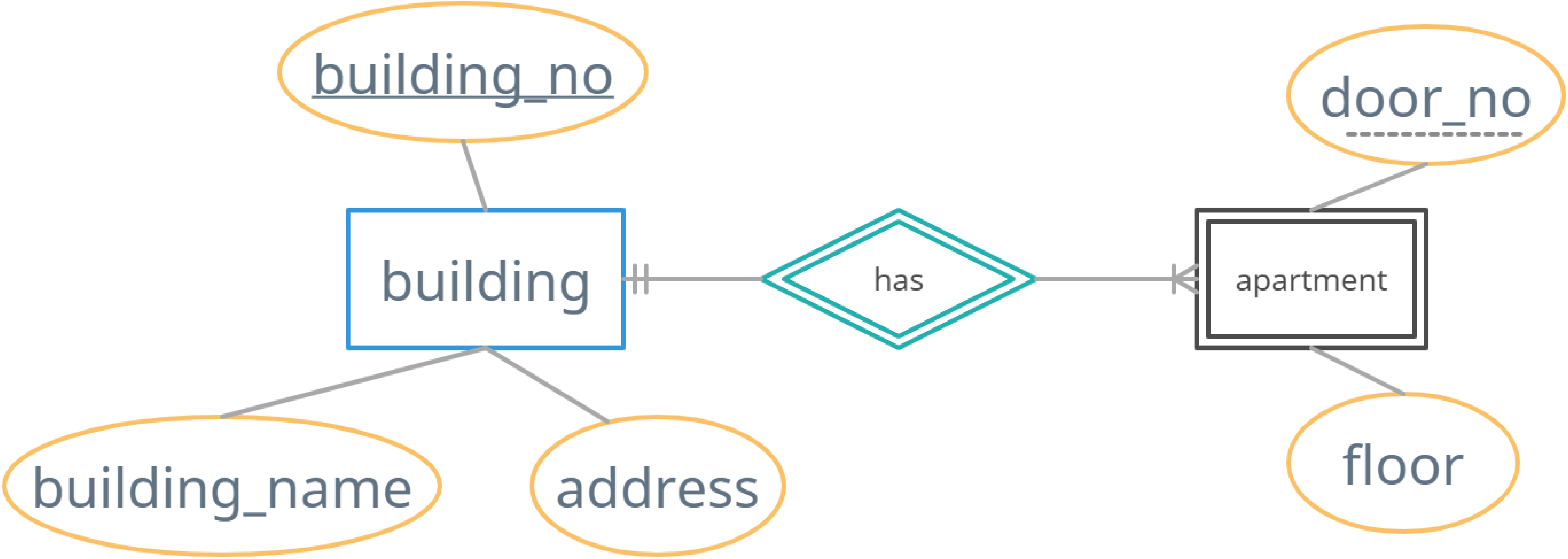
Represent Weak Entity Sets in E-R Diagram

- A weak entity set is depicted via a double rectangle.
- The discriminator of a weak entity set is underlined with a dashed line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond.



Weak Entity Sets - Example

- Two entity sets: Building and Apartment
- Attributes of building entity set: building_no, building_name, address.
- Attributes of building entity set: door_no, floor.
- Identify the primary keys, the relationship set, the strong and weak entity set, and draw the E-R diagram.


building_no
building
has
apartment
door_no
building_name
address
floor

Aggregation

- Consider the following ternary relationship set:
  - An instance of *proj guide* indicates that a particular student is guided by a particular instructor on a particular project.
  - A student could have different instructors as guides for different projects, which cannot be captured by a binary relationship between students and instructors.

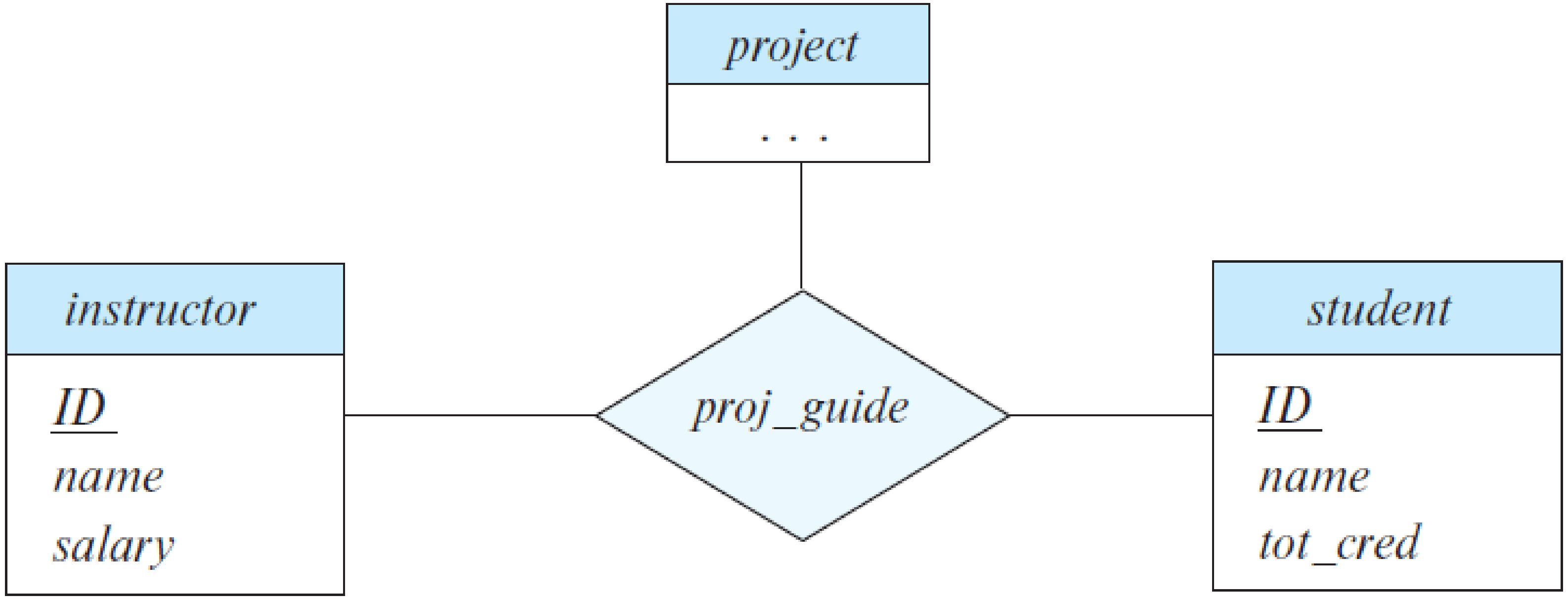
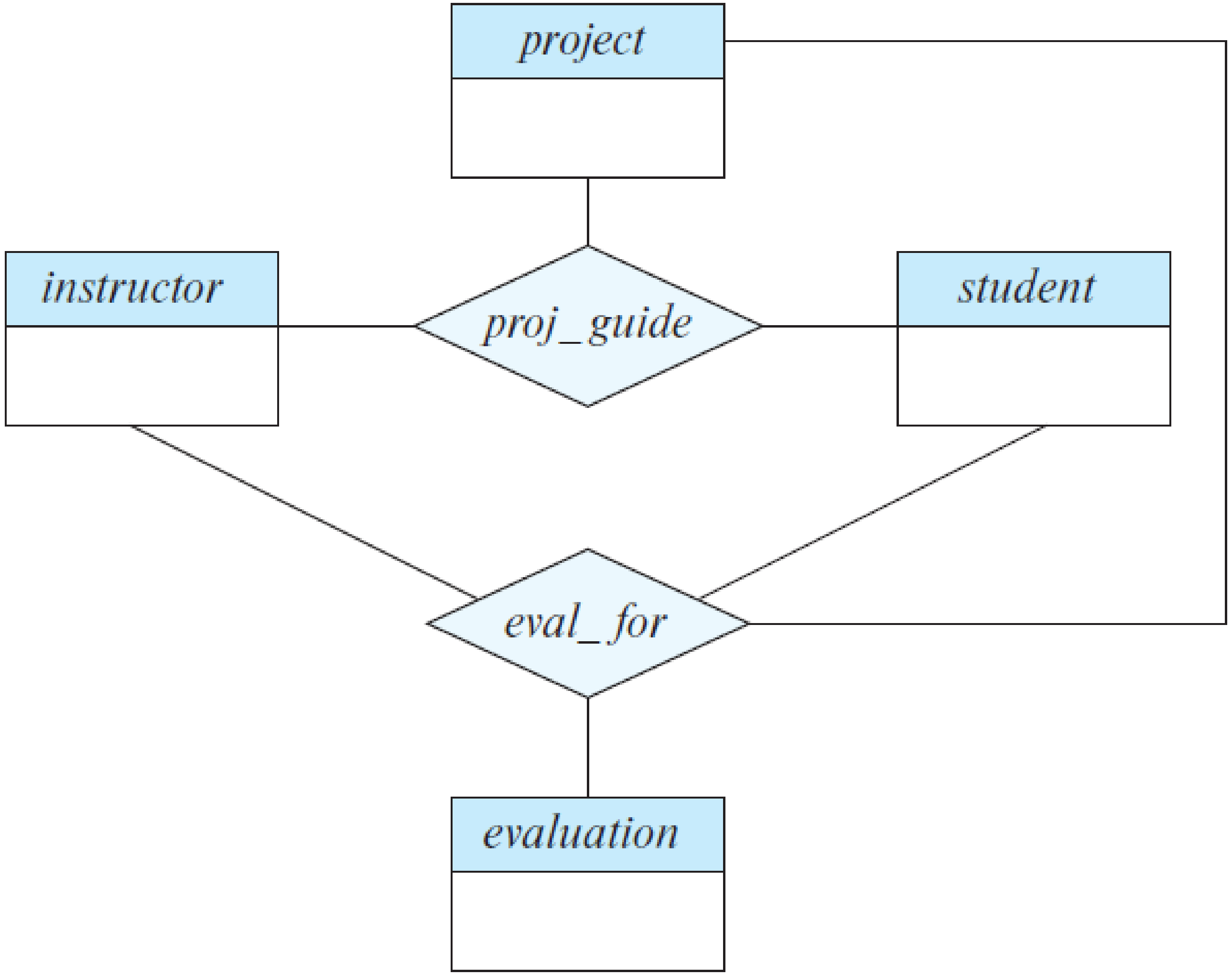


Figure 6.6 E-R diagram with a ternary relationship *proj_guide*.

Aggregation

- Suppose that each instructor guiding a student on a project is required to file a monthly evaluation report.



Aggregation

- **Aggregation is an abstraction through which relationships are treated as higher level entities.**

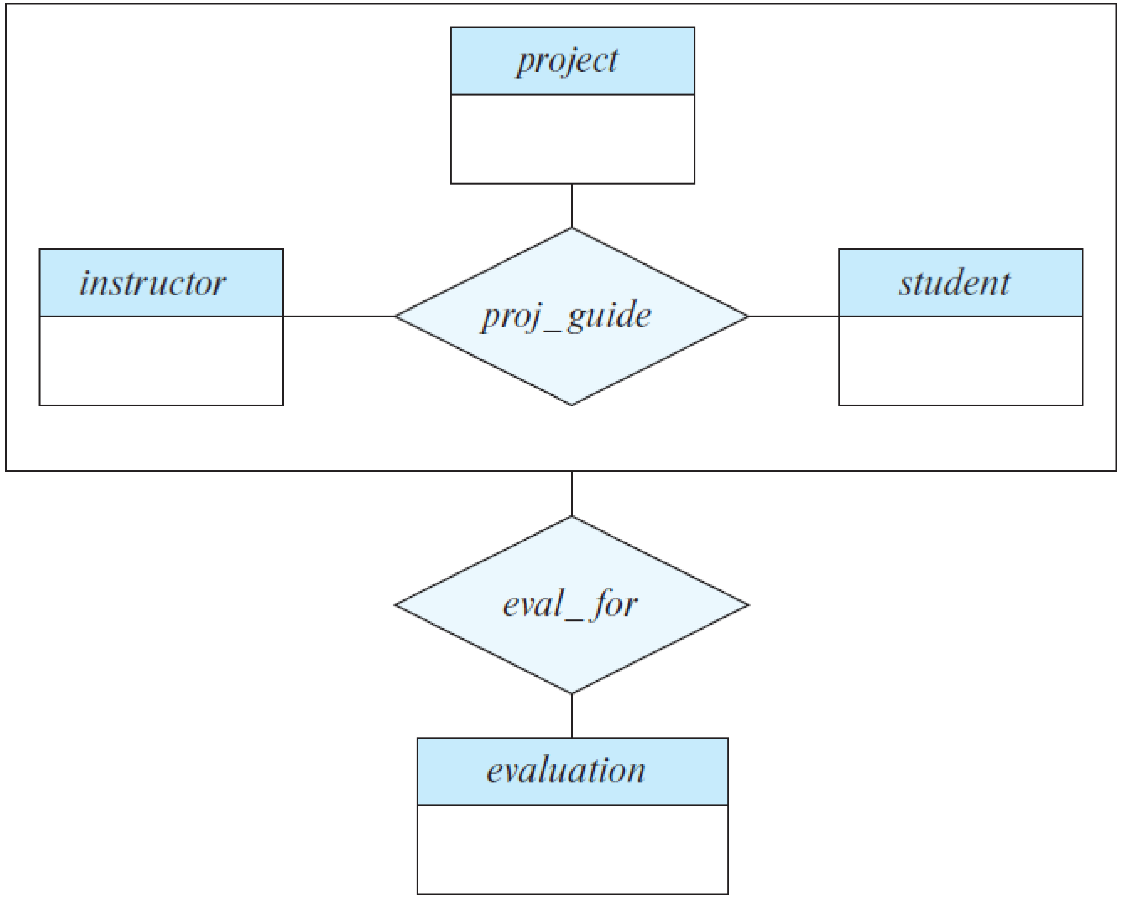
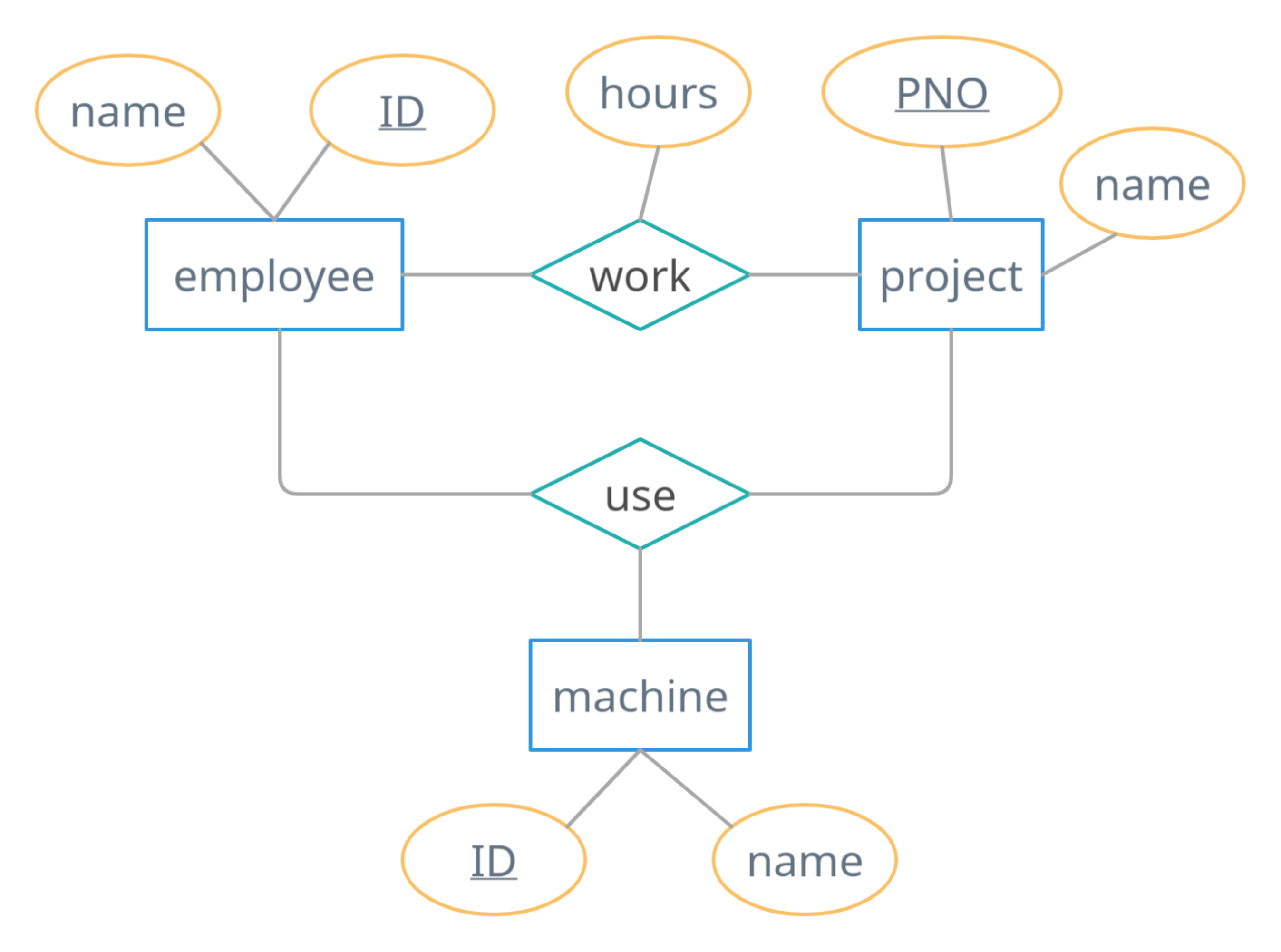


Figure 6.20 E-R diagram with aggregation.

Aggregation - Example

- Consider a DB with information about employees who work on a particular project and use a number of machines doing that work.

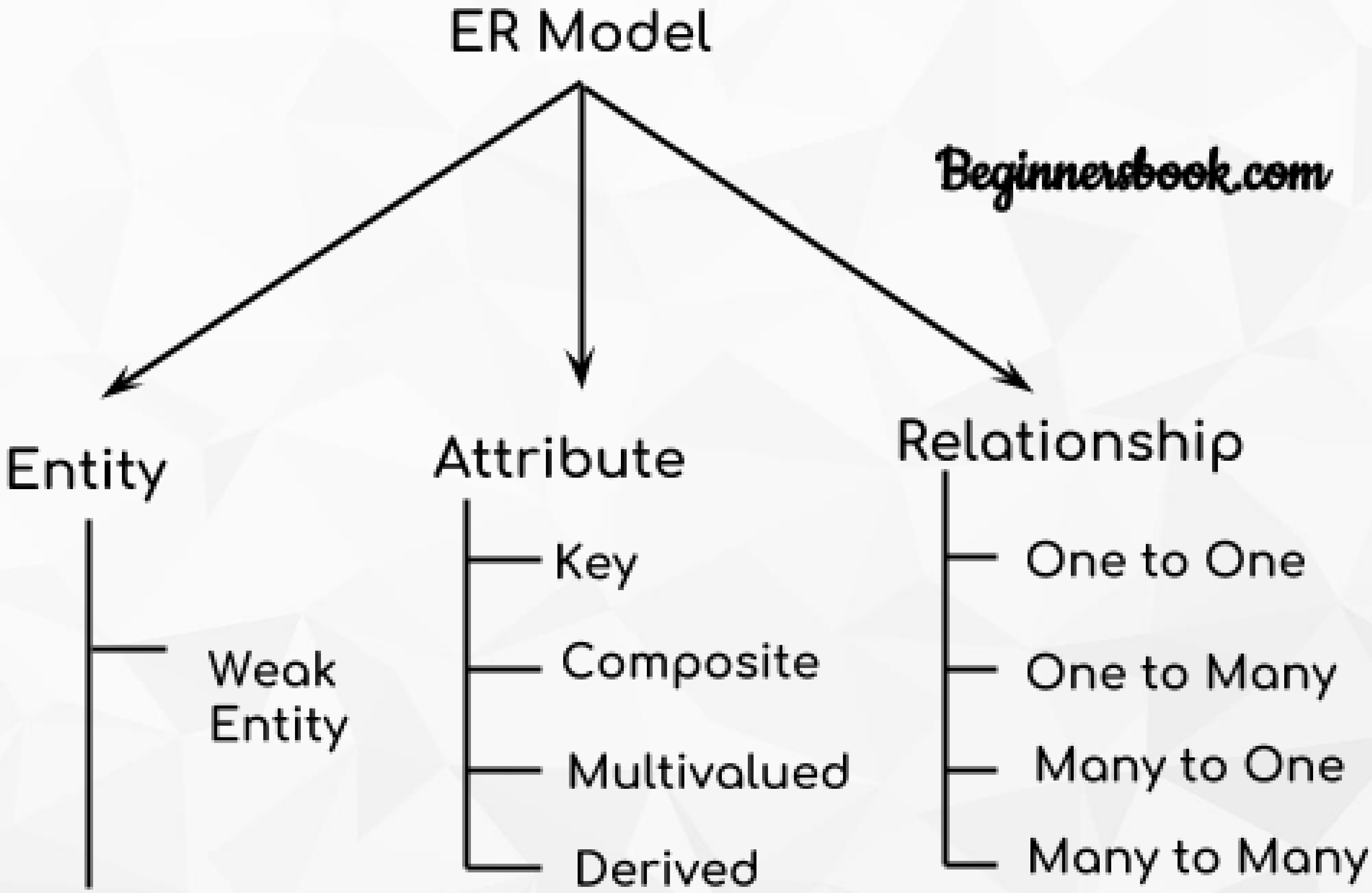


In summary …

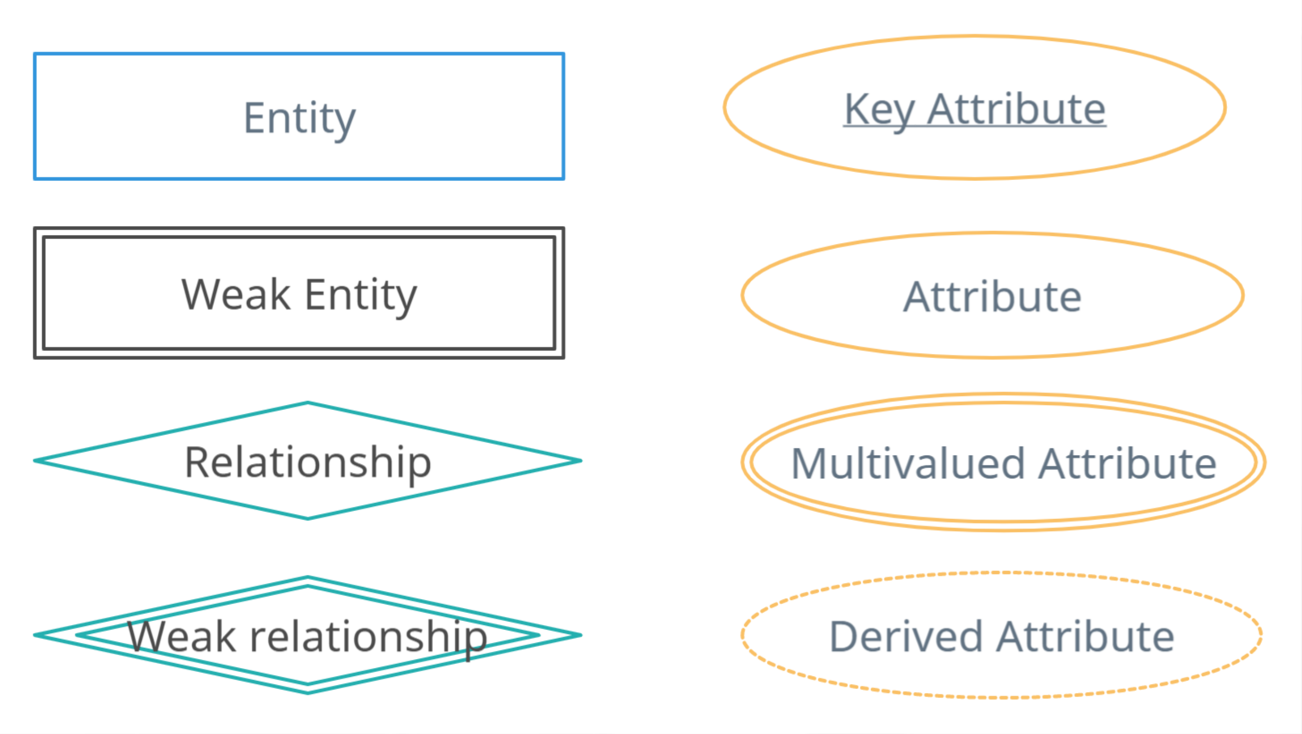
- Primary key for Entity sets = a set of attributes that can distinguish entities from each other.
- Primary key for Relationship sets = a set of attributes that can distinguish among the various relationships of a relationship set.
- Weak entity sets = an entity whose instances cannot exist in the database without the existence of an instance of another entity.
- Aggregation = an abstraction through which relationships are treated as higher level entities.

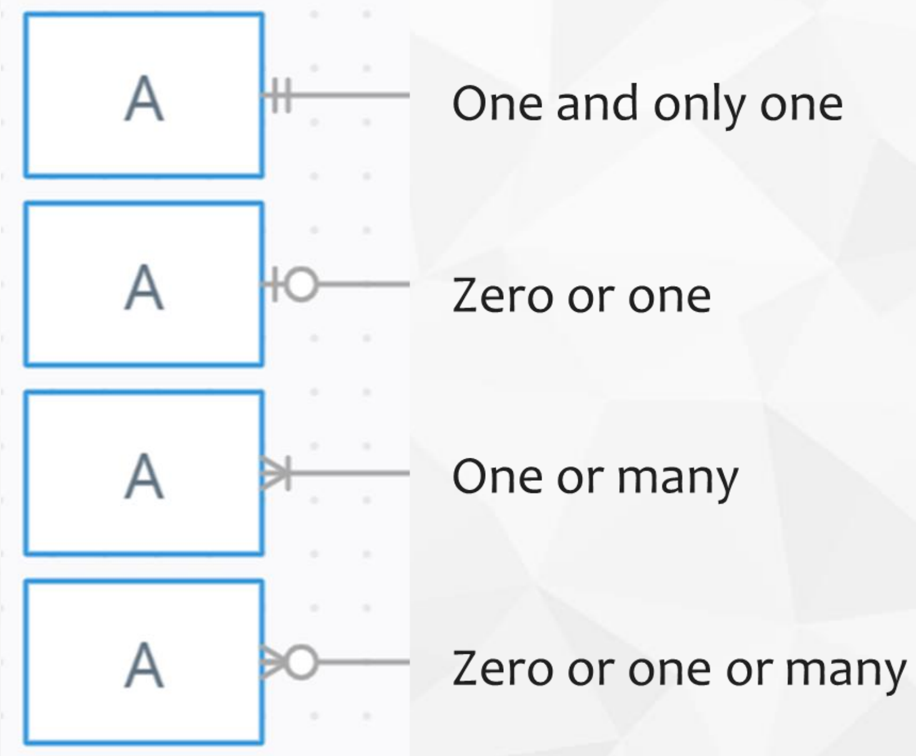
Exercises: Draw E-R Diagram

Summary: Components in ER Diagram

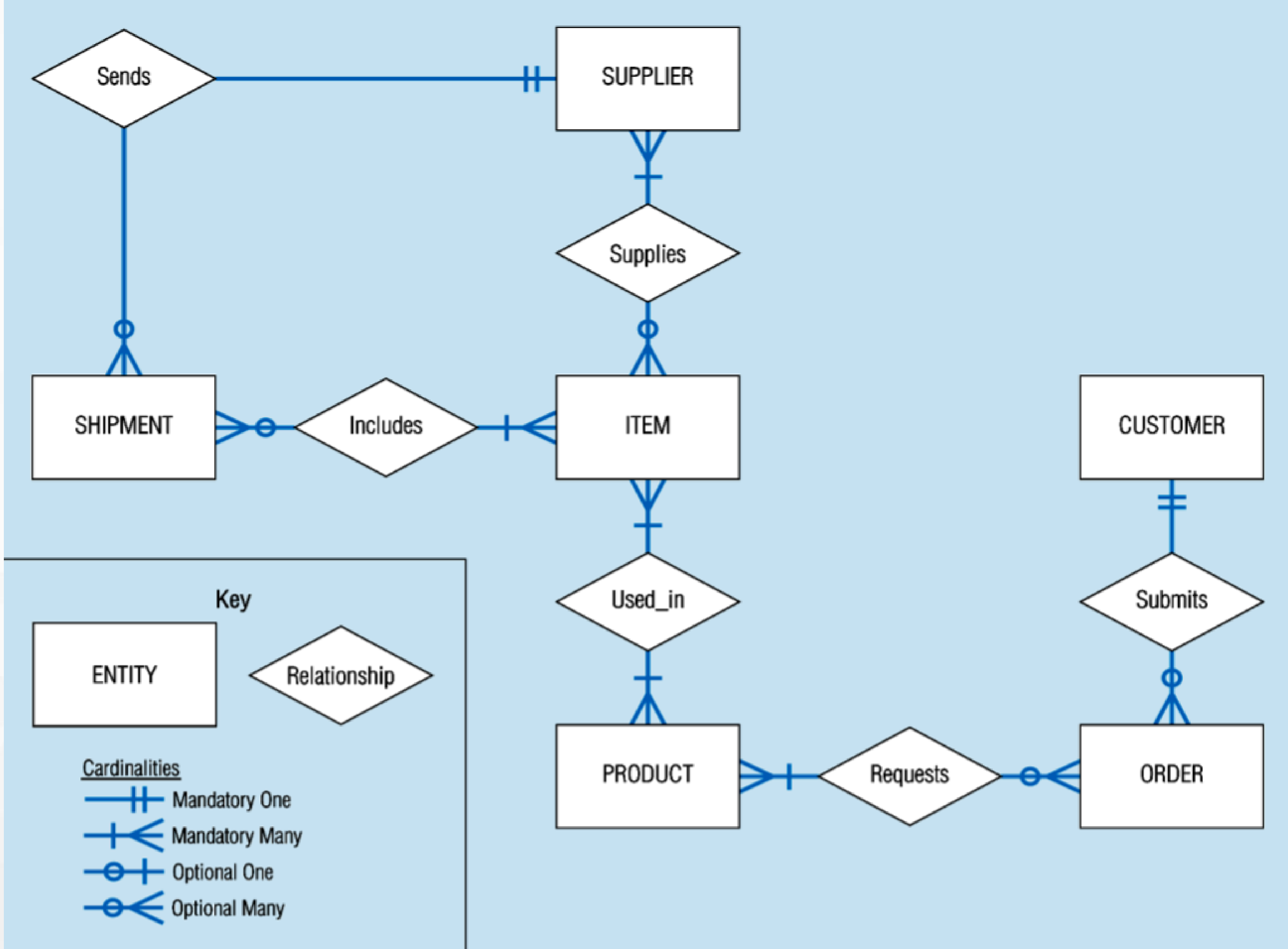


Summary: ER Diagram Symbols





Explain the below E-R model



Exercise

- In a university, a Student enrolls in Courses.
- A student must be assigned to at least one or more Courses.
- A course must have at least one student.
- Each course is taught by a single Professor.
- To maintain instruction quality, a Professor must deliver one and only one course.
- The database should store the ID and name of students, professors, and courses.

How to start? What to do?

– Find the basic entity types
– Find the attributes of entities
    - Decide to which entity an attribute should be assigned.
    - Which attributes are key attributes?
    - Some attributes are better modeled as own entities, which ones?
– Define the relationship types
    - Which role do entities play?
    - Do relationships require additional entity types?
    - Are the relationships total? Identifying? Are weak entities involved?
    - What are the cardinalities of the relationship type?

Exercise: University database

- The university database stores details about university students, courses, the semester a student took a particular course (and his mark and grade if he completed it), and what degree program each student is enrolled in.
- Consider the following requirements list:
  - The university offers one or more programs.
  - A program is made up of one or more courses.
  - Each course in a program is sequenced into a year (for example, year 1) and a semester (for example, semester 1).
  - A program has a name, a program identifier, the total credit points required to graduate, and the year it commenced.
  - A course has a name, a course identifier, a credit point value, and the year it commenced.
  - A student must enroll in a program.
  - A student takes the courses that are part of her program.
  - When a student takes a course, the year and semester he attempted it are recorded. When he finishes the course, a grade (such as A or B) and a mark (such as 60 percent) are recorded.
  - Students have a full name, a student identifier, a date of birth, and the year they first enrolled.

Exercise: Flight Database

- The flight database stores details about an airline's fleet, flights, and seat bookings.
- Consider the following requirements list:
  - The airline has one or more airplanes.
  - An airplane has a model number, a unique registration number, and the capacity to take one or more passengers.
  - An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time.
  - Each flight is carried out by a single airplane.
  - A passenger has given names, a surname, and a unique email address.
  - A passenger can book a seat on a flight.

RELATIONAL DATABASE DESIGN

Nguyễn Thị Hải Bình
nth.binh@hutech.edu.vn

What did we study before?

Features of Good Relational Design

SSN	FName	LName	BDate	DNum
111-11-1111	John	Smith	Jan-1-78	5
222-22-2222	Jane	Doe	Apr-1-76	5
333-33-3333	Jack	Rabbit	May-4-79	1

Employee Relation

DNum	DName	MgrSSN
5	Research	123-45-6789
1	Payroll	777-77-7777

Department Relation

Combine two relations

SSN	FName	LName	BDate	DNum	DName	MgrSSN
111-11-1111	John	Smith	Jan-1-78	5	Research	123-45-6789
222-22-2222	Jane	Doe	Apr-1-76	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	May-4-79	1	Payroll	777-77-7777

What are the problems of this combination?

Features of Good Relational Design

Problem?

1. **Redundancy**

→ For example, the information about the Research department is included in the two tuples.

SSN	FName	LName	BDate	DNum	DName	MgrSSN
111-11-1111	John	Smith	Jan-1-78	5	Research	123-45-6789
222-22-2222	Jane	Doe	Apr-1-76	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	May-4-79	1	Payroll	777-77-7777

Features of Good Relational Design

Problem?

2. **Inconsistency (Update Anomaly)**
   → It is important that the first and second tuples agree as to the manager SSN (MgrSNN).
   → But some user might update the manager SNN in one tuple but not all, and thus create inconsistency.

SSN	FName	LName	BDate	DNum	DName	MgrSSN
111-11-1111	John	Smith	Jan-1-78	5	**Research**	888-88-8888
222-22-2222	Jane	Doe	Apr-1-76	5	**Research**	123-45-6789
333-33-3333	Jack	Rabbit	May-4-79	1	Payroll	777-77-7777

Features of Good Relational Design

Problem?

3. **Insert Anomaly**
   - We insert a new department: dnumber = 6, dname = ‘Human Resources’.
   - This new department does not have any employees yet → We must use NULL values for employee information.

SSN	FName	LName	BDate	DNum	DName	MgrSSN
111-11-1111	John	Smith	Jan-1-78	5	Research	123-45-6789
222-22-2222	Jane	Doe	Apr-1-76	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	May-4-79	1	Payroll	777-77-7777
NULL	NULL	NULL	NULL	**6**	**Hum Resources**	NULL

Features of Good Relational Design

Problem?

4. **Delete Anomaly**
   - Suppose we delete the employee "Jack Rabbit" (who is the only employee in Payroll).
   - We have lost the all information on the Payroll department in the database.

   → The delete operation has deleted additional information than was not intended !!!!

SSN	FName	LName	BDate	DNum	DName	MgrSSN
111-11-1111	John	Smith	Jan-1-78	5	Research	123-45-6789
222-22-2222	Jane	Doe	Apr-1-76	5	Research	123-45-6789

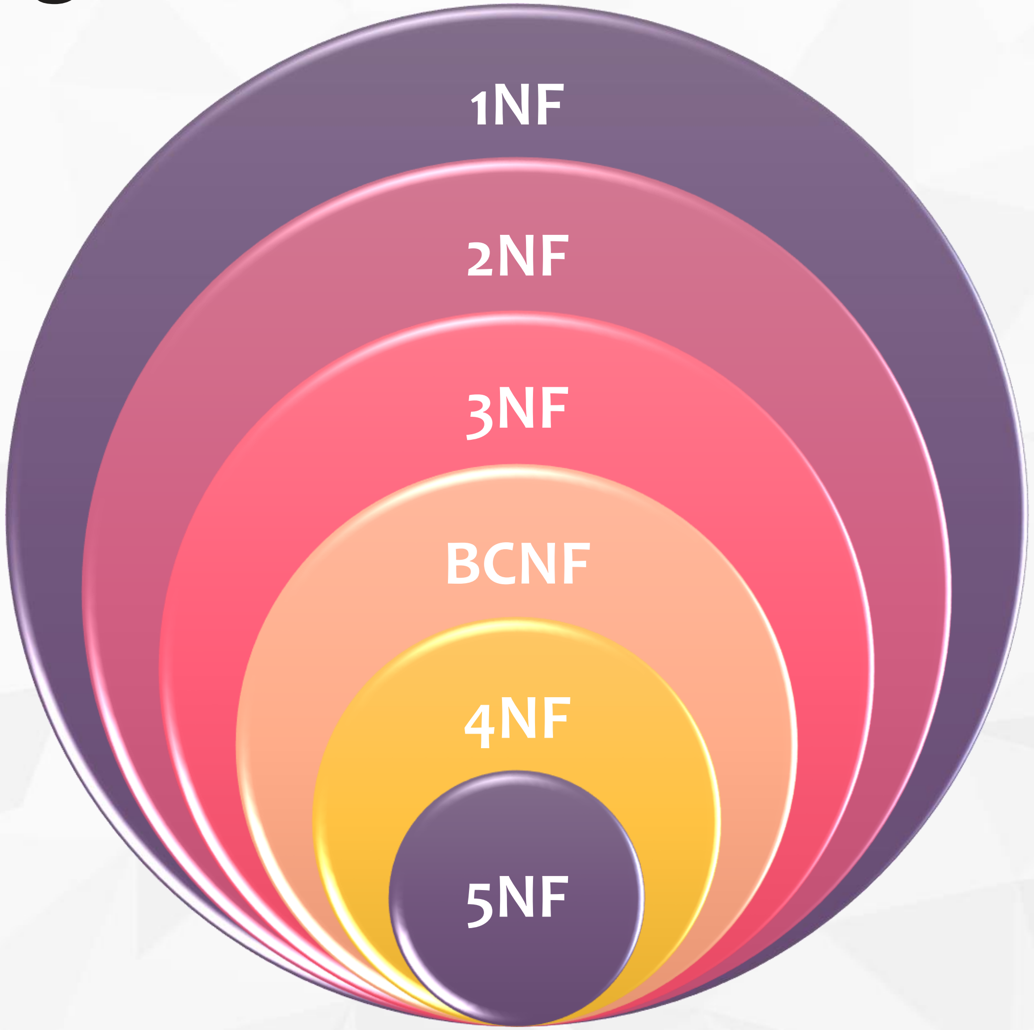
Design a good database

A bit of history

- The relational database model was introduced by E.F. Codd in 1970.
- After publishing the relational data model, he described the anomalies in his paper.
- Codd defined 3 "normal forms" in 1972 to solve the database anomalies.
- Codd's 3 NFs are all based on the concept of functional dependency.
- In 1974, Boyce & Codd introduced Boyce-Codd Normal Form or BCNF.
- 4NF and 5NF are then introduced by Fagin in 1977 and 1979, respectively.

Design a good database

Relationship among the normal forms



Design a good database

- Database designer should aim for relations in the "ultimate" 5NF.
- However, typically, designers stop the decomposition at 3NF or BCNF.
- Designing good database (for a large system) is a complex task.
- Normalization is useful in designing good database. But It is not a panacea.
- Indeed, sometimes , normal forms are violated deliberately to achieve better performance (less join operations)

    → Take precaution to make sure "bad" things (= data inconsistency, data loss, etc) do not happen with the data stored in the database.

Consider two methods to store the same information:

1. Method 1: Using one relations

SSN	FName	LName	BDate	DNum	DName	MgrSSN
111-11-1111	John	Smith	Jan-1-78	5	Research	123-45-6789
222-22-2222	Jane	Doe	Apr-1-76	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	May-4-79	1	Payroll	777-77-7777

2. Method 2: Using two relations

SSN	FName	LName	BDate	DNum
111-11-1111	John	Smith	Jan-1-78	5
222-22-2222	Jane	Doe	Apr-1-76	5
333-33-3333	Jack	Rabbit	May-4-79	1

DNum	DName	MgrSSN
5	Research	123-45-6789
1	Payroll	777-77-7777

What storage method is more efficient when we try to find the department name of the employee John Smith?

What storage method is more efficient when we try to find the department name of the employee John Smith?

Using storage method 1:

- Access only one relation to find department number.

Using storage method 2:

We must join 2 relations before we can find department number

This way is more difficult and slower.

→ Method 1 provide a more efficient (= faster) way to access the information.

Functional Dependencies

- The **cause** of the **anomalies** in **relations** is:
  - Duplication of information due to dependencies between different attributes in the relation.

SSN	FName	LName	BDate	DNum	DName	MgrSSN
111-11-1111	John	Smith	Jan-1-78	5	Research	123-45-6789
222-22-2222	Jane	Doe	Apr-1-76	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	May-4-79	1	Payroll	777-77-7777

- The normal forms 2NF, 3NF and BCNF are based on the concept of Functional Dependency.

→ We will study functional dependencies and their applications before discussion the 2NF and 3NF.

Functional Dependencies

- Let X and Y be two attributes in a relation R.
- We say that Y is functionally dependent on X (notation: **X** $\rightarrow$ **Y**) iff:

$$X \rightarrow Y \iff \begin{array}{c}\textbf{for any two tuples } t_1 \textbf{ and } t_2 \textbf{ of the relation } R\textbf{:} \\ \textbf{if } t_1[X] = t_2[X] \textbf{ then: } t_1[Y] = t_2[Y]\end{array}$$

- Y is functionally dependent on X = X functionally determines Y.

Example of Functional Dependencies

- Consider the following relation: Employee(SSN, FName, LName, PNumber, PName, Hours)
- Employee relation stores information about the employees AND the projects that an employee works on.
- The key of this relation is: (SSN, PNumber)
- Sample content of the Employee relation:

SSN	FName	LName	PNumber	PName	Hours
111-11-1111	John	Smith	pj1	DBApplet	20
111-11-1111	John	Smith	pj2	WebServer	10
111-22-3333	Jane	Doe	pj1	DBApplet	5

Example of Functional Dependencies

Employee(SSN, FName, LName, PNumber, PName, Hours)

SSN	FName	LName	PNumber	PName	Hours
111-11-1111	John	Smith	pj1	DBApplet	20
111-11-1111	John	Smith	pj2	WebServer	10
111-22-3333	Jane	Doe	pj1	DBApplet	5

Some functional dependencies in Employee relation:

- SNN → fname, lname
- PNumber → PName
- SSN, PNumber → Hours

Example of Functional Dependencies

Employee(SSN, FName, LName, PNumber, PName, Hours)

SSN	FName	LName	PNumber	PName	Hours
111-11-1111	John	Smith	pj1	DBApplet	20
111-11-1111	John	Smith	pj2	WebServer	10
111-22-3333	Jane	Doe	pj1	DBApplet	5

Is the following functional dependency valid?

- SSN, PNumber → fname, lname

Example of Functional Dependencies

Employee(SSN, FName, LName, PNumber, PName, Hours)

SSN	FName	LName	PNumber	PName	Hours
111-11-1111	John	Smith	pj1	DBApplet	20
111-11-1111	John	Smith	pj2	WebServer	10
111-22-3333	Jane	Doe	pj1	DBApplet	5

Is the following functional dependency valid?

- SSN, PNumber → PName

Example of Functional Dependencies

Employee(SSN, FName, LName, PNumber, PName, Hours)

SSN	FName	LName	PNumber	PName	Hours
111-11-1111	John	Smith	pj1	DBApplet	20
111-11-1111	John	Smith	pj2	WebServer	10
111-22-3333	Jane	Doe	pj1	DBApplet	5

Are the following functional dependencies valid?

- PNumber → FName, LName
- Pnumber → Hours
- SSN → Hours

The natural functional dependency

- A key of a relation will always functionally determine every attributes in the relation.
- Example:

```
SSN is key in the relation Employee:

+-----------+--------+---------+----------+
| ssn       | fname  | lname   | salary   |
+-----------+--------+---------+----------+
| 123456789 | John   | Smith   | 30000.00 |
| 333445555 | Frankl | Wong    | 40000.00 |
| 999887777 | Alicia | Zelaya  | 25000.00 |
| 987654321 | Jennif | Wallace | 43000.00 |
| 666884444 | Ramesh | Narayan | 38000.00 |
| 453453453 | Joyce  | English | 25000.00 |
| 987987987 | Ahmad  | Jabbar  | 25000.00 |
| 888665555 | James  | Borg    | 55000.00 |
+-----------+--------+---------+----------+
```

```
Notice that:

    SSN → fname    (If you know SSN, you also know fname)
    SSN → lname
```

The natural functional dependency

- We call the functional dependency:

  Key $\rightarrow$ attribute in relation

a natural functional dependency.

- **Example:** Consider Employee relation
  - Employee(SSN, FName, LName, PNumber, PName, Hours)
  - Keys: (SSN, PNumber)
  - Natural function dependency: SSN, PNumber → Hours
  - The following function dependencies are not natural:

    SSN → FName, Lname

    PNumber → PName

"Good" and "bad" Functional Dependencies

- A natural (or trivial) functional dependency is a "good" functional dependency.

- All other kinds of functional dependencies are “bad” functional dependencies.
  - These bad functional dependencies will cause anomalies in relations.

- **Example**: Consider Employee relation
  - Employee(SSN, FName, LName, PNumber, PName, Hours)
  - Keys: (SSN, PNumber)

  - “Good” functional dependency: SSN, PNumber → Hours

  - “Bad” functional dependencies :

    SSN → FName, Lname

    PNumber → PName

"Good" and "bad" Functional Dependencies

Why a "bad" functional dependencies will cause anomalies?

- Consider the Employee relation:
  - Employee(SSN, FName, LName, PNumber, PName, Hours)
  - Keys: (SSN, PNumber)

- An instance of Employee relation:

SSN	FName	LName	PNumber	PName	Hours
111-11-1111	John	Smith	pj1	DBApplet	20
111-11-1111	John	Smith	pj2	WebServer	10
111-22-3333	Jane	Doe	pj1	DBApplet	5

- Example of a bad functional dependency:

  PNumber → PName

"Good" and "bad" Functional Dependencies

Why PNumber → PName will cause anomalies?

- Since PNumber is not a key, you can have multiple tuples in the database with the same PNumber value.

SSN	FName	LName	PNumber	PName	Hours
111-11-1111	John	Smith	pj1	DBApplet	20
111-11-1111	John	Smith	pj2	WebServer	10
111-22-3333	Jane	Doe	pj1	DBApplet	5

- Since PNumber → Pname, tuples with the same value for PNumber will have the same PName value.

  → **This functional dependency has caused duplication of information.**

  → **When we update the PName from "DBApplet" to "DatabaseApplet", there will be multiple updates in the table.**

"Good" and "bad" Functional Dependencies

What can we do to remove anomalies from "bad" relations?

- We will break up the relation into multiple relations.
- Each (smaller) relation will only have "good" functional dependencies.

Exercises: Identify functional dependencies

- Consider the relation: R(A, B, C)
- Suppose R contains 4 tuples:

A	B	C
1	2	2
1	3	2
1	4	2
2	5	2

- For each of the following functional dependencies, state whether or not the dependency is satisfied by this relation instance.
  - (a) $A \rightarrow B$
  - (b) $A \rightarrow C$
  - (c) $B \rightarrow A$
  - (d) $B \rightarrow C$
  - (e) $C \rightarrow A$
  - (f) $C \rightarrow B$
  - (g) $AB \rightarrow C$
  - (h) $AC \rightarrow B$
  - (i) $BC \rightarrow A$

Exercises: Identify functional dependencies

- Consider the relation: R(A, B, C)
- Suppose R contains 5 tuples:

A	B	C
1	2	3
1	3	2
1	2	2
3	2	1
3	2	3

- For each of the following functional dependencies, state whether or not the dependency is satisfied by this relation instance.
  - (a) $A \rightarrow B$
  - (b) $A \rightarrow C$
  - (c) $B \rightarrow A$
  - (d) $B \rightarrow C$
  - (e) $C \rightarrow A$
  - (f) $C \rightarrow B$
  - (g) $AB \rightarrow C$
  - (h) $AC \rightarrow B$
  - (i) $BC \rightarrow A$

Exercises: Identify functional dependencies

- Consider the relation: inst_dept (ID, name, salary, dept name, building, budget)
- An instance of inst_dept relation is given below:

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Exercises: Identify functional dependencies

- Consider the relation: classroom(building, room_number, capacity)
- An instance of classroom relation is given below:

building	*room_number*	*capacity*
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Exercises: Identify functional dependencies

- Consider the relation: Student (Student_ID, Student_Address, Lecture, Teaching_Assistant)
- An instance of classroom relation is given below:

Student_Id	Student_Address	Lecture	Teaching_Assisant
1234	Rämistrasse 72	Data Modelling and Databases	Bob
1280	Rennweg 19	Concepts of Concurrent Computation	Scott
1234	Rämistrasse 72	Visual Computing	Sarah
1299	Börsenstrasse 42	Concepts of Concurrent Computation	Benjamin
1356	Klusplatz 45	Concepts of Concurrent Computation	Benjamin

Exercises: Identify functional dependencies

- Consider the database:

  Hotel (HotelNo, HotelName, City)

  Room(RoomNo, HotelNo, type, price)

  Booking(HotelNo, GuestNo, DateFrom, DateTo, RoomNo)

  Guest(GuestNo, GuestName, GuestAddress)

- The same guest cannot have overlapping bookings at the same hotel at the same time.

Exercises: Identify functional dependencies

- Consider the database:

  product(model, maker, price)

  pc(model, speed, ram, hd)

  laptop(model, speed, ram, hd, screen)

  printer(model, color, type)

Exercises: Identify functional dependencies

- Consider the following relational schema: Car(make, model, year, color, dealer)
- Each tuple in relation Car specifies that one or more cars of a particular make, model, and year in a particular color are available at a particular dealer. For example, the tuple (Honda, Civic, 2010, Blue, Fred's Friendly Folks) indicates that 2010 Honda Civics in blue are available at the Fred's Friendly Folks car dealer.
- For each of the following English statements, write one functional dependency that best captures the statement.
- (a) The model name for a car is trademarked by its make, i.e., no two makes can use the same model name.
- (b) Each dealer sells only one model of each make of car.

Exercises: Identify functional dependencies

- Consider the relation: Order (Product_Id, Product_Name, Customer_Id, Customer_Name, Order_Date, Item_Price, Amount)

Importance of the Key

- The "natural" functional dependencies arise from the fact that:

the **key** of a relation **functionally determines *all* *attributes* in the relation.**

- To determine if a functional dependency is natural (beneficial, "harmless") , we must therefore first find all keys of a relation.

- However, the problem of find all keys of a relation. is NP-complete, i.e. the problem is very hard to solve efficiently.

How to recognize a key of a relation R

- **A set of attributes S is a super key of a relation R if:**
  - S functionally determines all attributes in R
- **A set of attributes K is a key of a relation R iff:**
  - K functionally determines all attributes in R
  - K is minimal

primary key candidate key

Student

ID	SSN	Name	Address	Phone	GPA
12345	111-11..	John D	123 My Ln	123-4567	3.5
56789	123-21...	Jane D	321 YourWay	876-5678	3.8

primary key candidate key

Course

CID	CourseNo	Name	Hours
2342	CS457	Database	4
1235	CS255	Comp Arch I	4

foreign key key

GradeReport

ID	CID	Semester	Grade
12345	2342	Fall 02	W
12345	2342	Fall 03	B+

Closure set X^+ of a set of attribute X

- Let X be some set of attributes of a R.
- The closure set X^+ is:
  - set of attributes that are functionally determined by X

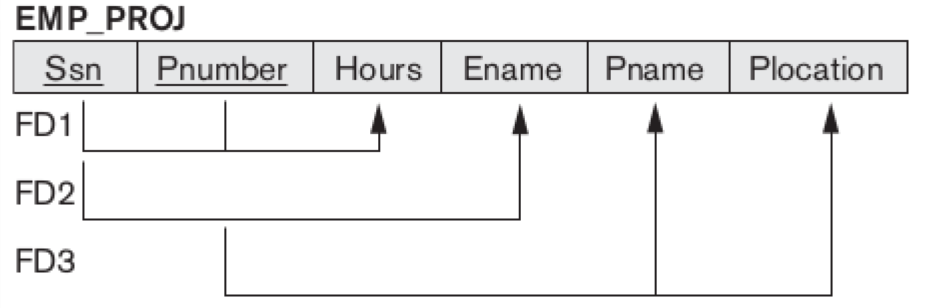
- Clearly: $X \subseteq X+$

The importance of the closure set X^+

- Let R be a relation
- Let X be a set of attributes of R
- If $X^+ = R$ then X is a superkey
- I.e. The closure can tell us if the set of attribute X is a super key

Examples of Closure set

- Given the relation Emp_Proj (SSN, Pnumber, Hours, Ename, Pname, Plocation)



- Functional dependency (FD) set:
  - FDs = {???}

- Closure set?

Examples of Closure set

- Given the relation Student (ID, Name, Age, City, State)
- Functional dependency (FD) set:
  - FDs = {???}
- Closure set?

Algorithm to find the closure set X^+

Given relation: R
Given functional dependency set: FDs
Given: $X \subseteq Attr(R)$
Compute X^+ -- *the closure of X under F*

Algorithm:

$X^+ := X$
do
    $X^+_{old} := X^+$
    foreach functional dependency $Y \rightarrow Z$ in F:
        if $X^+ \supseteq Y$ **then** $X^+ := X^+ \cup Z$
until $X^+ = X^+_{old}$

Finding the closure set X^+: Example

- **Given the relation EmpProj:**

EmpProj(SSN, FName, LName, PNumber, PName, PLocation, Hours)

- **Functional dependency set:**

FDs = {SSN $\rightarrow$ FName, Lname,

PNumber $\rightarrow$ PName, PLocation,

SSN,PNumber $\rightarrow$ Hours}

- **Compute $(SSN)^+$, $(Pnumber)^+$, $(SSN, Pnumber)^+$**

Observation from the example

Summary of the results:

- $(SSN)^+ = \{SSN, FName, LName\}$
- $(PNumber)^+ = \{PNumber, PName, Plocation\}$
- $(SSN,PNumber)^+ = \{SSN, PNumber, FName, LName, PName, PLocation, Hours\}$

Observation 1:

- (SSN,PNumber) is a super key because $(SSN,PNumber)^+ = R$

Observation 2:

- (SSN,PNumber) is minimal because
  - $\{(SSN,PNumber) - (SSN)\}^+ = (PNumber) \neq R$
  - $\{(SSN,PNumber) - (PNumber)\}^+ = (SSN) \neq R$
- So every attribute in (SSN,PNumber) is necessary to form the super key.
- Therefore, (SSN,PNumber) is minimal.
- In other words: (SSN,PNumber) is a candidate key .

How to find a key after computing X^+

- If $X^+ \neq R$ then X is not a super key.

- If $X^+ = R$ then X is a super key.
  - However, X may not be a key because X may not be minimal.

<u>How to check for minimality:</u>

- Remove one or more attributes A from X
- Compute the closure of X-A:
  - If $(X\text{-}A)^+$??? R then X is not minimal.
- X is a candidate key if:
  - For every attribute A: $(X–A)^+ \neq R$

Finding the closure set X^+: Example

- Given R(A, B, C, D, E, F)
- FDs = { $A \rightarrow BC$, $BD \rightarrow EF$, $F \rightarrow A$ }
- Compute $\boldsymbol{A^+}$, $\boldsymbol{BD^+}$, $\boldsymbol{AD^+}$

Find all keys: Example

- Given the relation R(A, B, C, D, E, F)

  and FDs = {A → BC, BD → EF, F → A}

- Algorithm to find all keys:
  - Find the irreplaceable attributes (An attribute is replaceable if it appears in the right-hand side of some functional dependency)
  - A key must include every irreplaceable attribute
  - The base set is the set of all irreplaceable attributes
  - Add other attributes to the base set until you have a key

Find all keys: Exercises

1. Given R(A, B, C, D, E) and FDs = {$A \rightarrow C$, $E \rightarrow D$, $B \rightarrow C$}
2. Given R(A, B, C, D, E) and FDs = {$A \rightarrow BE$, $C \rightarrow BE$, $B \rightarrow D$}
3. Given R(A, B, C, D, E, F) and FDs = {$A \rightarrow B$, $B \rightarrow D$, $C \rightarrow D$, $E \rightarrow F$}
4. Given R(A, B, C, D) and FDs = {$AB \rightarrow C$, $BC \rightarrow D$, $CD \rightarrow A$}
5. Given R(A, B, C, D) and FDs = {$A \rightarrow BCD$, $C \rightarrow A$}
6. Given R(E-ID, E-NAME, E-CITY, E-STATE) and FDs = { E-ID $\rightarrow$ E-NAME, E-ID $\rightarrow$ E-CITY, E-ID $\rightarrow$ E-STATE, E-CITY $\rightarrow$ E-STATE }
7. Given the relation schema R(A, B, C, D, E) and FDs = {$A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$}

The First Normal Form (1NF)

A relation is in 1NF (first normal form) if every attribute of the relation has atomic values.

SSN	FName	LName	BDate	DNum
111-11-1111	John	Smith	Jan-1-78	{5, 1}
222-22-2222	Jane	Doe	Apr-1-76	5
333-33-3333	Jack	Rabbit	May-4-79	1

This relation is not in 1NF

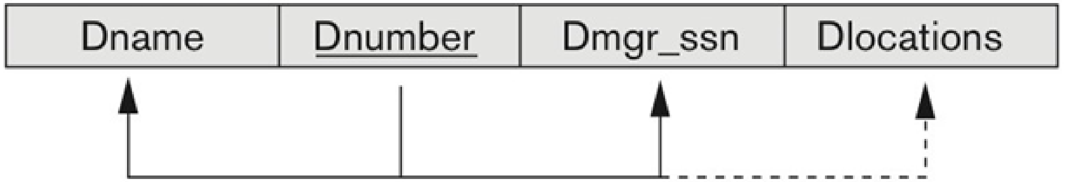
SSN	FName	LName	BDate	DNum
111-11-1111	John	Smith	Jan-1-78	5
111-11-1111	John	Smith	Jan-1-78	1
222-22-2222	Jane	Doe	Apr-1-76	5
333-33-3333	Jack	Rabbit	May-4-79	1

This relation is in 1NF

1NF - Example

(a)

DEPARTMENT



(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(a) A relation schema that is not in 1NF.

(b) Sample state of relation DEPARTMENT.

How to normalize the relation to 1NF?

1NF - Example

(a)
EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

(b)
EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, Alicia J.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(a) A relation schema that is not in 1NF.
(b) Sample state of relation EMP_PROJ.

How to normalize the relation to 1NF?

Second Normal Form (2NF)

- Uses the concepts of **FDs, primary key**
- **Definitions:**
  - **Prime attribute:** An attribute that is member of the primary key K
  - **Full functional dependency:** a FD Y -> Z is full functional dependency if removal of any attribute from Y means the FD does not hold any more.

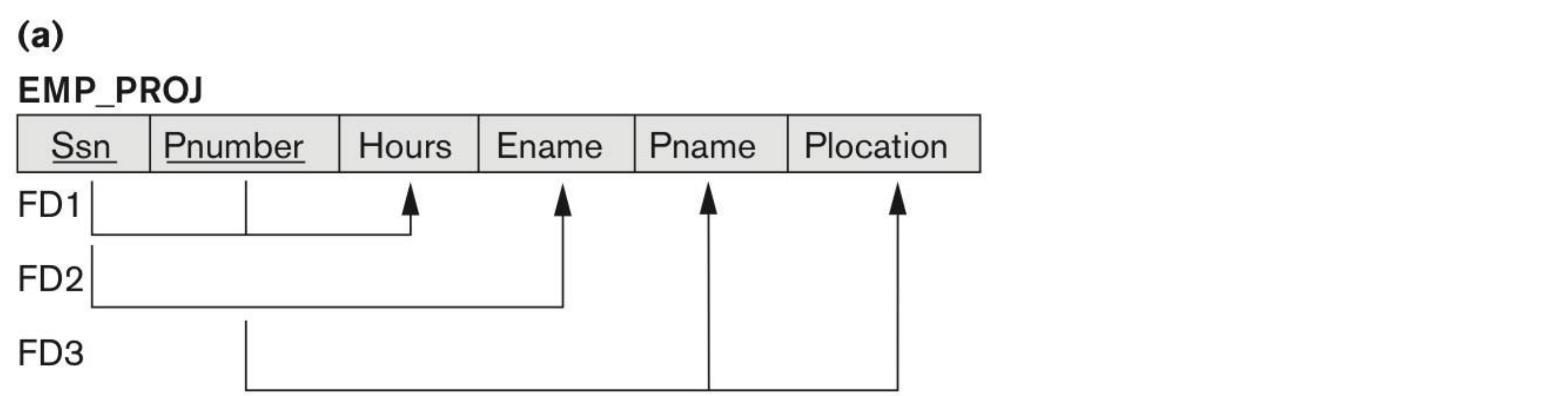
- **Examples:**
  - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold

  - {SSN, PNUMBER} -> ENAME is not a full FD (it is called a partial dependency) since SSN -> ENAME also holds

Second Normal Form (2NF)

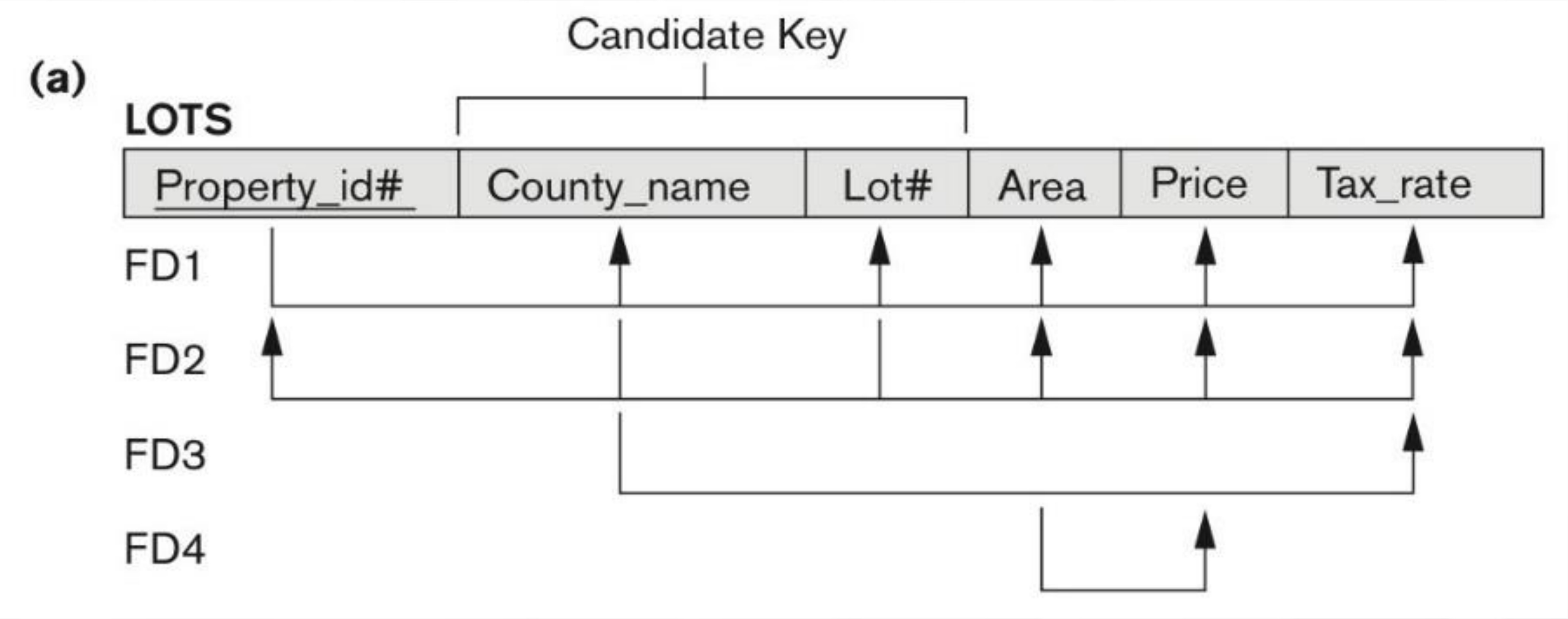
- A relation schema R is in **second normal form (2NF)** if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the keys.

- R can be decomposed into 2NF relations via the process of 2NF normalization or "second normalization"



(a) A relation schema that is not in 2NF.

2NF - Example



Candidate Key
(a)
LOTS
Property_id#
County_name
Lot#
Area
Price
Tax_rate
FD1
FD2
FD3
FD4

Third Normal Form (3NF)

- Definition of Transitive functional dependency:
  - **Transitive functional dependency:** a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z

- Examples:
  - SSN -> DMGRSSN is a **transitive** FD
    - Since SSN -> DNUMBER and DNUMBER -> DMGRSSN hold

  - SSN -> ENAME is **non-transitive**
    - Since there is no set of attributes X where SSN -> X and X -> ENAME

SSN	FName	LName	BDate	DNum	DName	MgrSSN
111-11-1111	John	Smith	Jan-1-78	5	Research	123-45-6789
222-22-2222	Jane	Doe	Apr-1-76	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	May-4-79	1	Payroll	777-77-7777

Third Normal Form (3NF)

- A relation schema R is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the keys.
- R can be decomposed into 3NF relations via the process of 3NF normalization
- NOTE:
  - In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is not a candidate key.
  - When Y is a candidate key, there is no problem with the transitive dependency .
  - E.g., Consider EMP (SSN, Emp#, Salary).
    - Here, SSN -> Emp# -> Salary and Emp# is a candidate key.

Third Normal Form (3NF)

- A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y:
  - X is a super key.
  - Y is a prime attribute (each element of Y is part of some candidate key).

- The following relation is not in 3NF

```
Employee1(SSN, FName, LName, PNumber, PName, Hours)

Functional Dependencies:

    SSN             →  FName, LName
    PNumber         →  PName
    SSN, PNumber    →  Hours
```

Third Normal Form (3NF)

- The following relation is not in 3NF

```
Employee1(SSN, FName, LName, PNumber, PName, Hours)

Functional Dependencies:

    SSN            →  FName, LName
    PNumber        →  PName
    SSN, PNumber   →  Hours
```

SSN	FName	LName	PNumber	PName	Hours
111-11-1111	John	Smith	pj1	DBApplet	20
111-11-1111	John	Smith	pj2	WebServer	10
111-22-3333	Jane	Doe	pj1	DBApplet	5

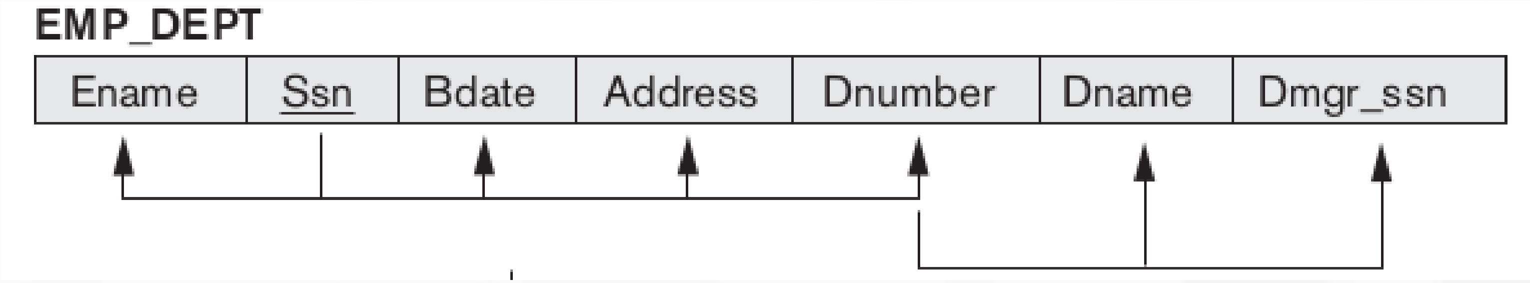
Third Normal Form (3NF)

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

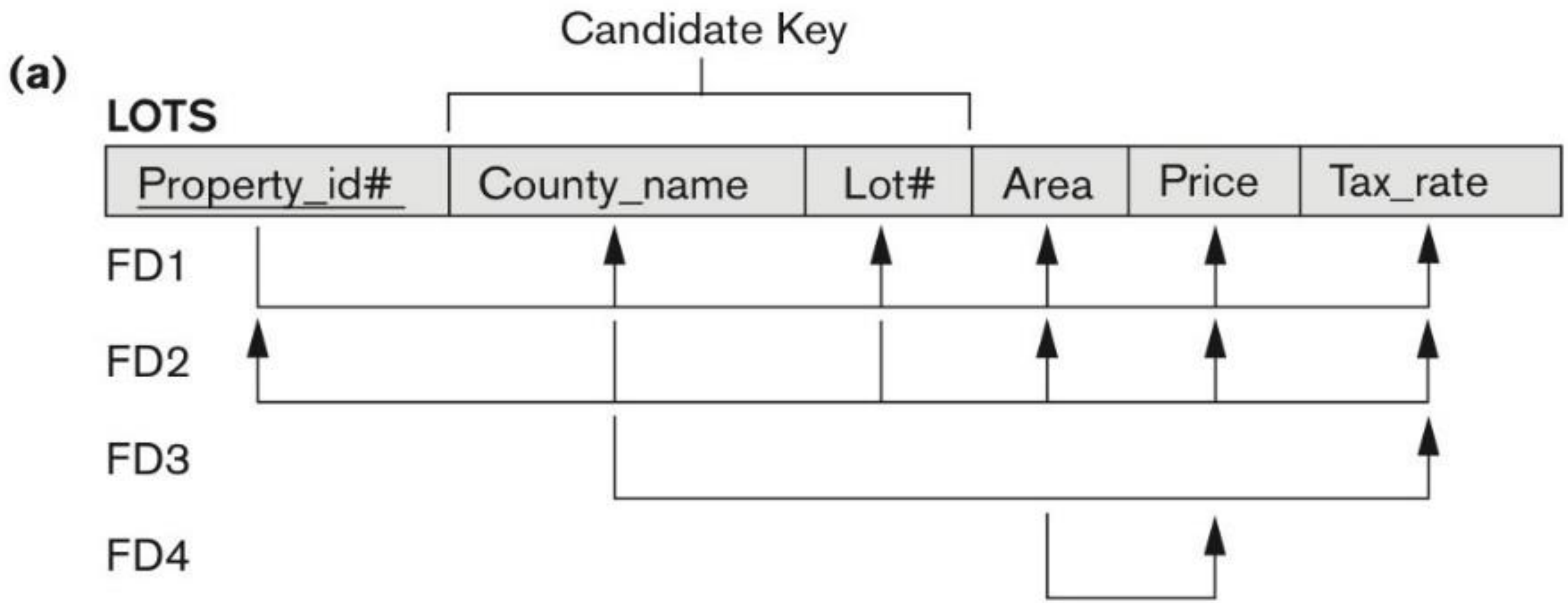
The relation TEACH that is in 3NF.

Third Normal Form (3NF)

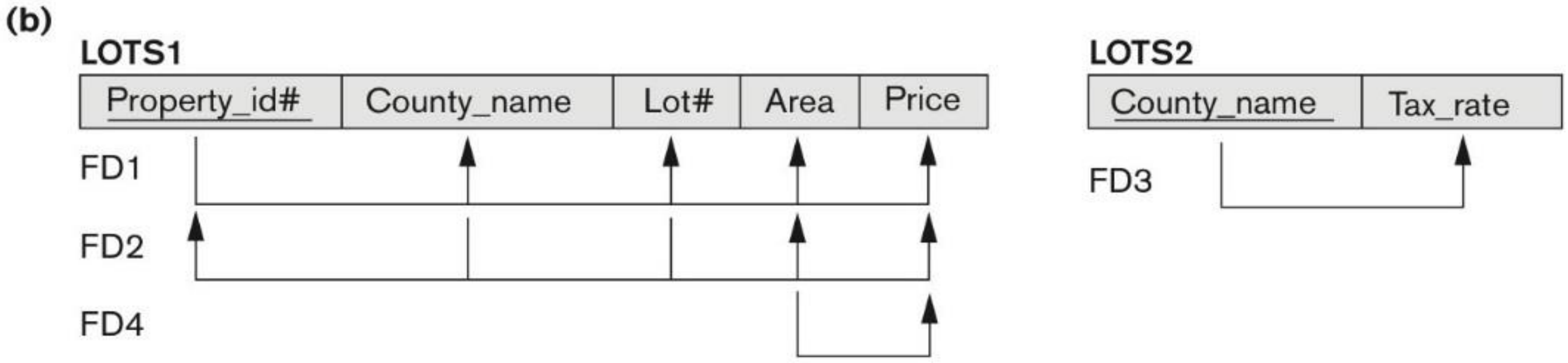


The relation EMP_DEPT is not in 3NF.

3NF - Example

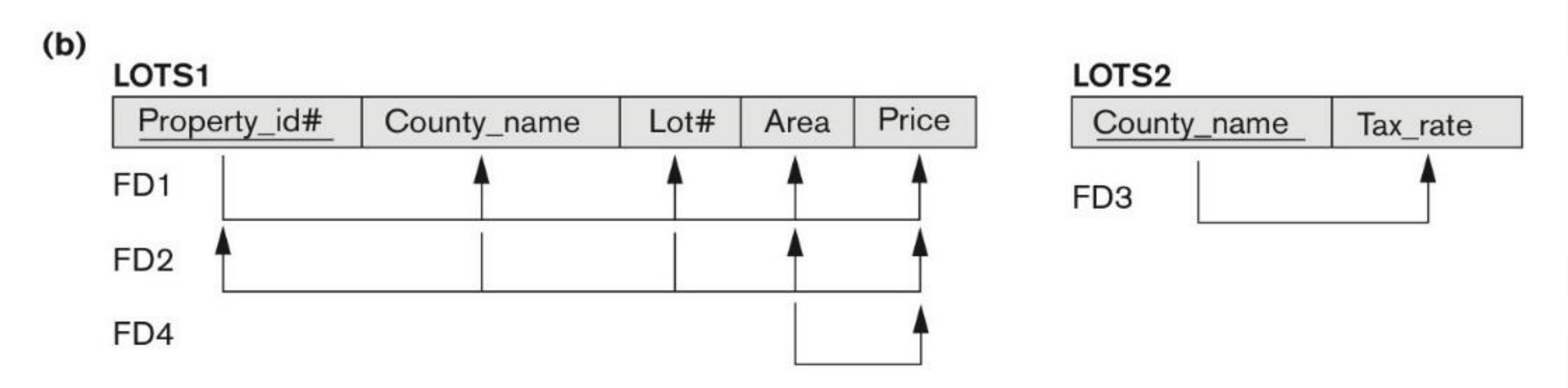


Candidate Key
(a)
LOTS
Property_id#
County_name
Lot#
Area
Price
Tax_rate
FD1
FD2
FD3
FD4



(b)
LOTS1
Property_id#
County_name
Lot#
Area
Price
FD1
FD2
FD4
LOTS2
County_name
Tax_rate
FD3

3NF - Example


(b)
LOTS1
Property_id#
County_name
Lot#
Area
Price
FD1
FD2
FD4
LOTS2
County_name
Tax_rate
FD3

Is the following relation in 3NF?

- Consider relation R(A, B, C, D, E)
- FDs = {A → BC, CD → E, B → D, E → A}

Normal Forms Defined Informally

- 1st normal form
  - All attributes depend on **the key**
- 2nd normal form
  - All attributes depend on **the whole key**
- 3rd normal form
  - All attributes depend on **nothing but the key**

Decomposition into 3NF

```
for ( each relation R )
{
        if ( R is not in 3 NF )
        {
                Let X → B be a violating function dependency;
                Decompose R into:
                        (1) R1 = X⁺;
                        (2) R2 = R - R1 ∪ X;
        }
}
```

Decomposition into 3NF - Example

- Given: Employee(SSN, Fname, LName, PNumber, PName, Hours)

- Functional dependency set = {SSN $\rightarrow$ FName,

  Lname, PNumber $\rightarrow$ Pname,

  SSN, PNumber $\rightarrow$ Hours}

- The key of Employee is (SSN, PNumber)

- Let decompose Employee into 3NF

Decomposition into 3NF - Example

- Given R = (SSN, FName, LName, DNumber, DName, MgrSSN)

- Functional Dependencies:
  - SSN → FName, LName, DNumber
  - Number → DName, MgrSSN

- Let decompose R into 3NF

Decomposition into 3NF - Example

- Given