Lecture 3

E-R Model to Relational Tables

COMP3278A

Introduction to Database Management Systems

Dr. Ping Luo

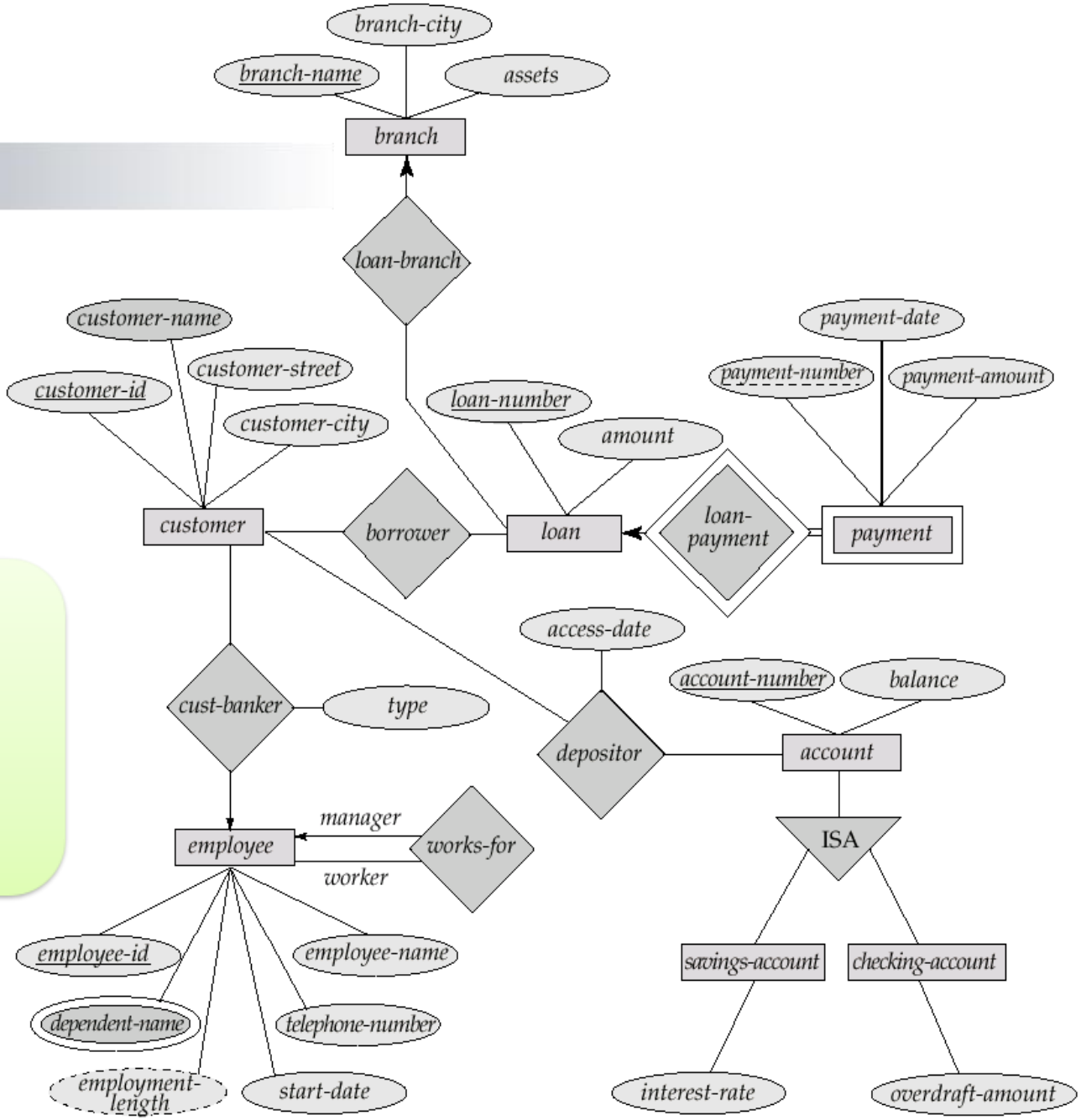
Email : pluo@cs.hku.hk

Department of Computer Science, The University of Hong Kong

Acknowledgement: **Dr. Chui Chun Kit**

Recap

Can you understand the data model captured by this E-R Diagram?



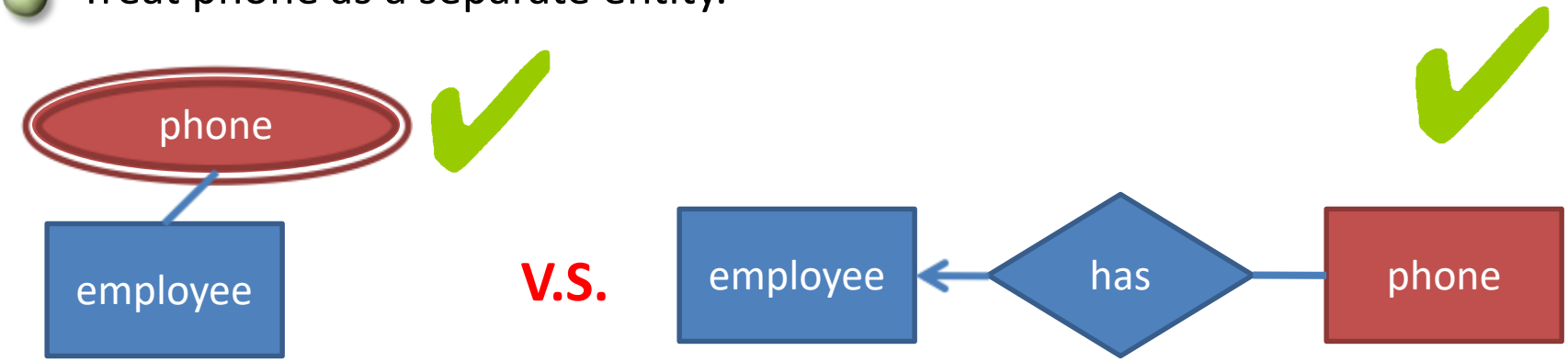
E-R Diagram for a Banking Enterprise

Section 3.1

E-R Design Decision

Entity sets v.s. Attributes

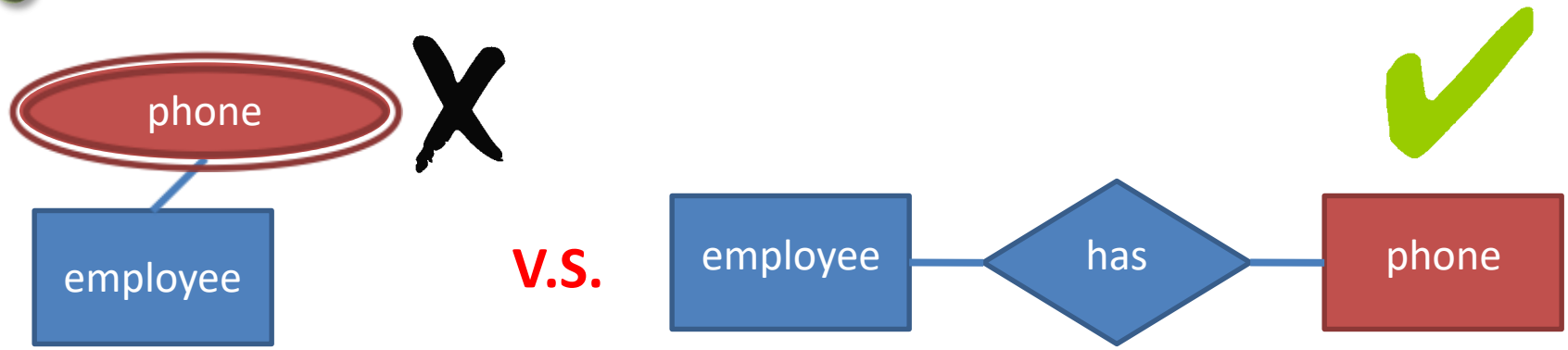
- How do you model an employee and his phone number?
  - Treat phone number as an attribute of an employee.
  - Treat phone as a separate entity.



1. In my company, an employee can have multiple phone numbers…

Entity sets v.s. Attributes

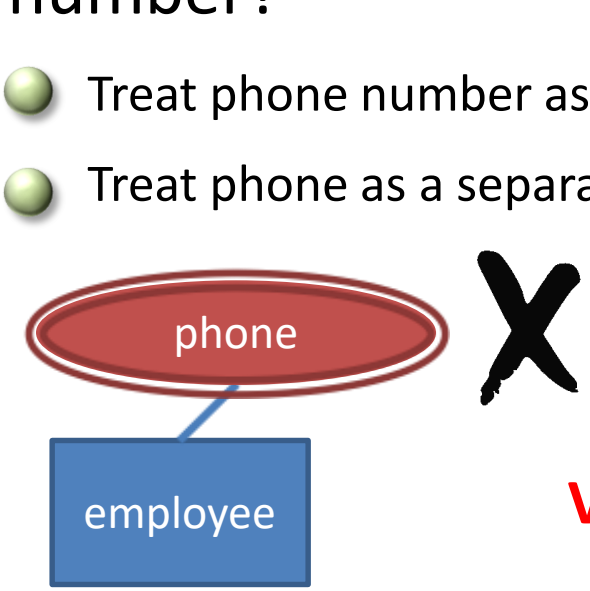
- How do you model an employee and his phone number?
  - Treat phone number as an attribute of an employee.
  - Treat phone as a separate entity.

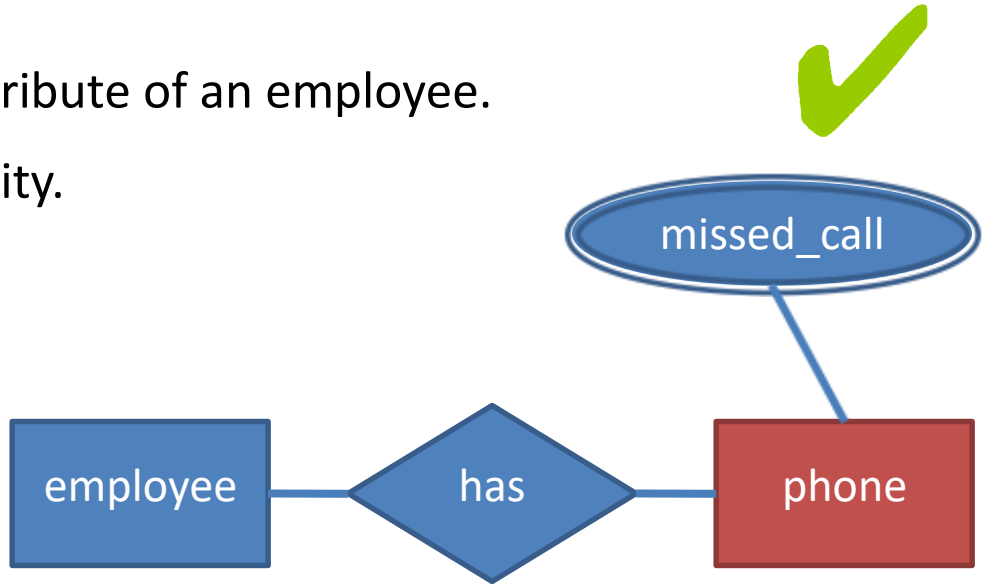


2. In my company, a phone number can be **shared by** multiple employee…

Entity sets v.s. Attributes

- How do you model an employee and his phone number?
  - Treat phone number as an attribute of an employee.
  - Treat phone as a separate entity.





3. In the system, for each phone, I want to keep a list of missed call numbers.

Entity sets v.s. Relationship sets

- Use a relationship set to **describe an action that occurs between entities.**
  - **Hint:** entity sets often have “nouns” as name, and relationship sets have “verbs” as name.

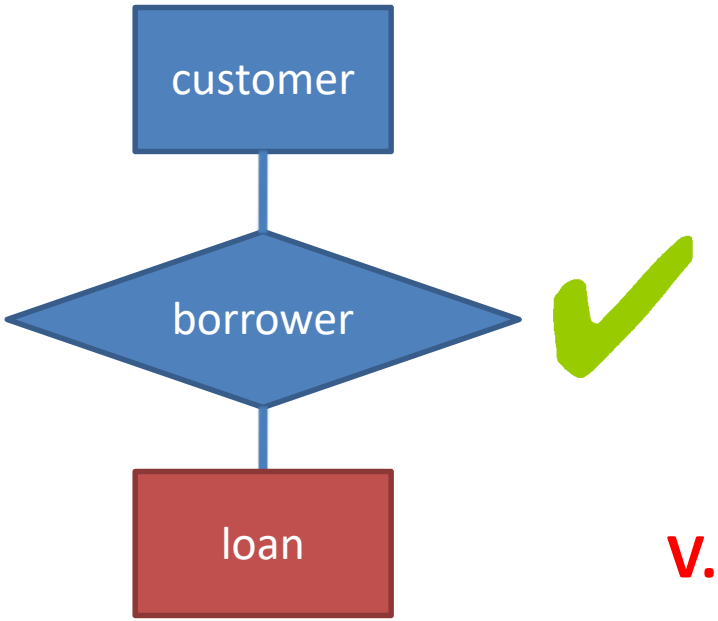
Entity sets v.s. Relationship sets

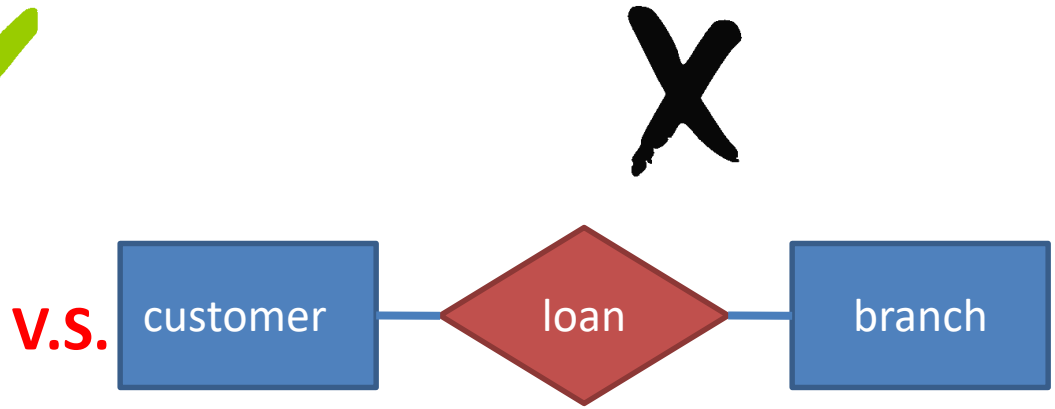
- How to model a loan?
  - 1. As a Loan entity.
  - 2. As a relationship between a customer and a branch.

A loan is an object in this phrase.

Can we have joint loan?
(E.g., Do we need to express something like **"A loan can be associated with multiple customers"**)

Entity sets v.s. Relationship sets





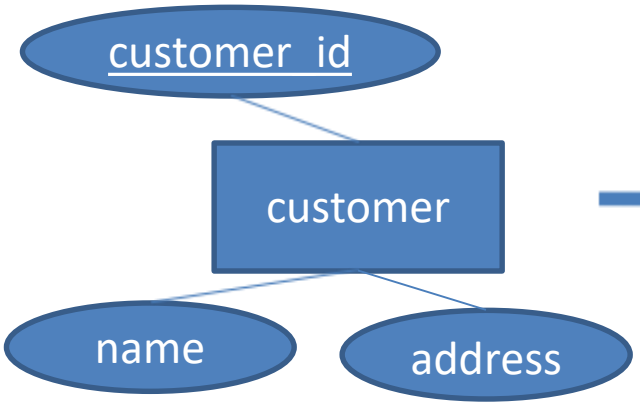
Can we have joint loan?
(E.g., Do we need to express something like **"A loan can be associated with multiple customers"**)

Section 3.2

From E-R Schema to Relational Tables

Entity sets

- An **Entity set** (or another name, **strong entity set**) reduces to **a table** with the same attributes.

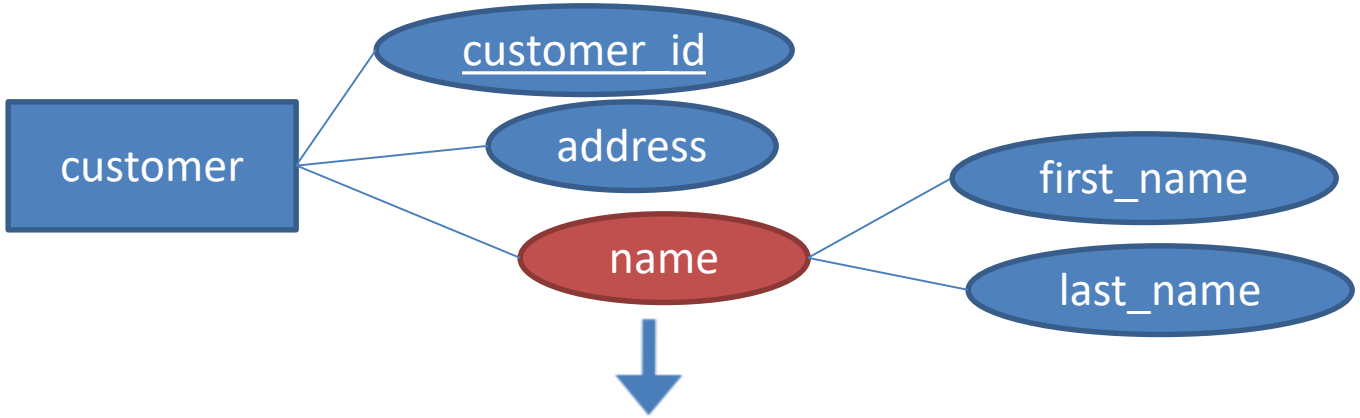


customer_id	name	address
…	…	…
…	…	…
…	…	…

Customer (customer_id, name, address)

Attributes

- **Composite attributes** are **flattened out** by creating a separate attribute for each component attribute.
  - e.g, **name** becomes **name.first_name** and **name.last_name**.

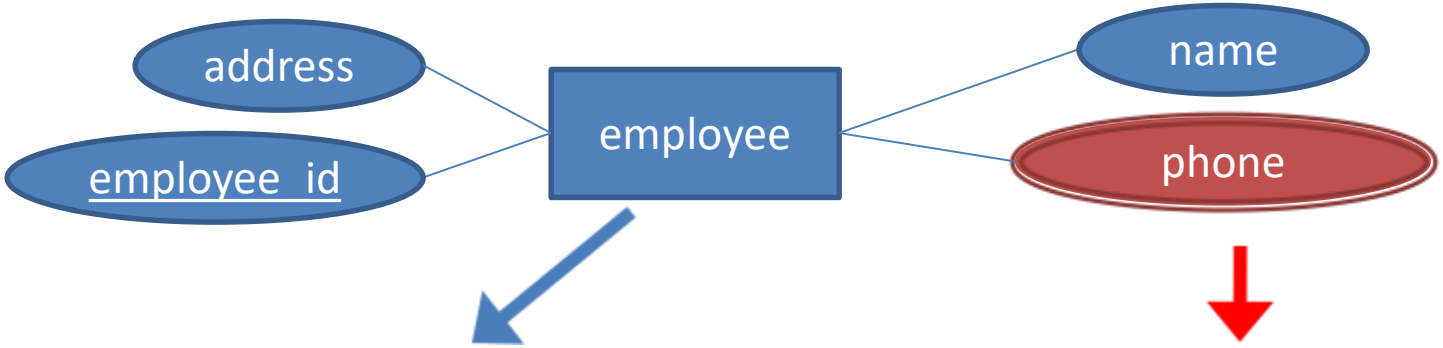


customer_id	name.first_name	name.last_name	address
…	…	…	…
…	…	…	…

Customer (customer_id, name.first_name, name.last_name, address)

Attributes

- A **multi-valued attribute** M of an entity set E is represented by **a separate table EM**, with the primary key of E as one of EM's attribute.



employee_id	name	address
1	Kit	…
2	Ben	…

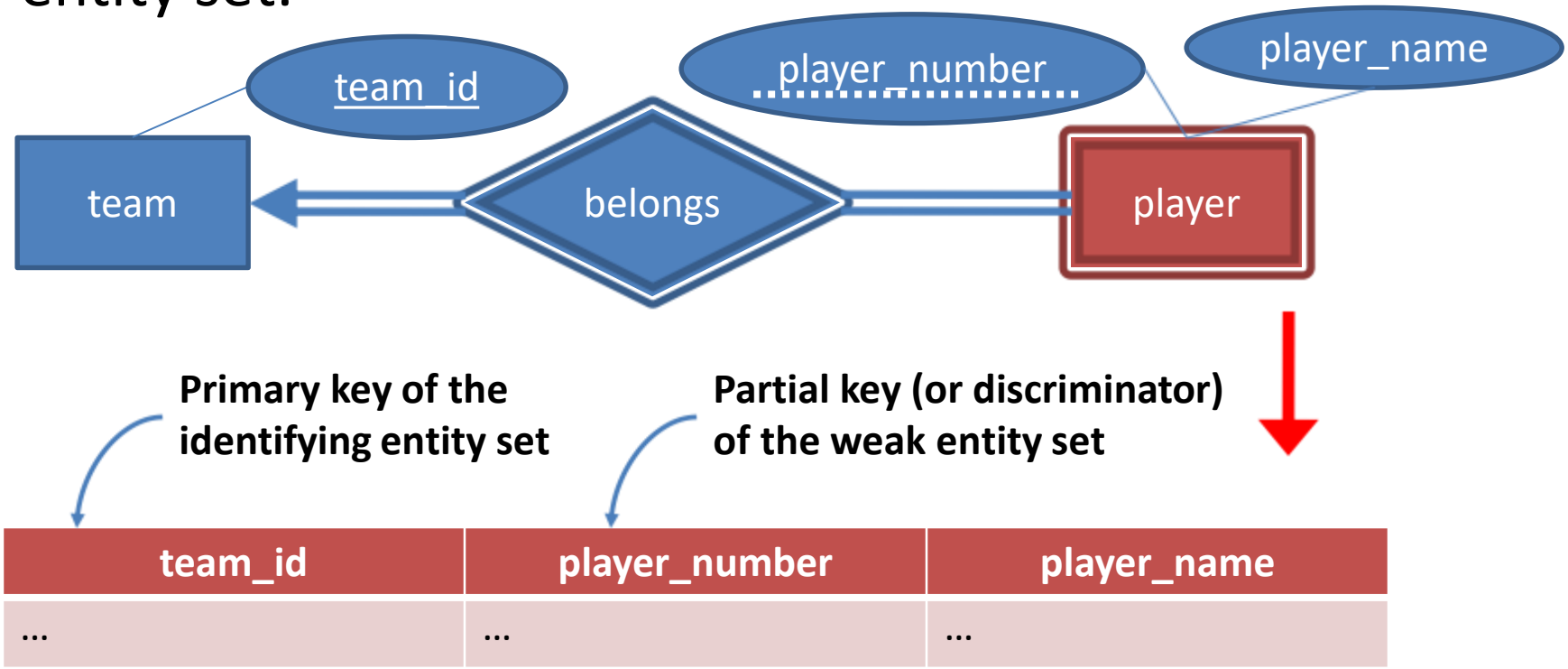
employee_id	phone
1	9123 4567
1	2987 6543

Employee(employee_id, name, address)

EmployeePhone(employee_id, phone)

Weak entity sets

- A **weak entity set** becomes a **table** that includes the columns for the primary key of the identifying strong entity set.



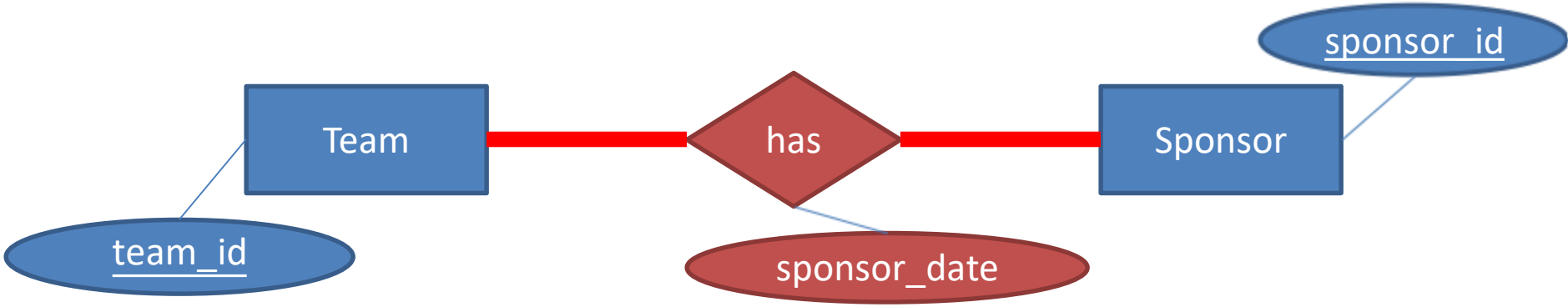
Player (teamID, player_number, player_name)

Relationship sets

- The reduction depends on their mapping cardinalities.
  - Many to many
  - One to many / many to one
  - One to one

Relationship sets

- A **many-to-many** relationship set is a table with columns for the primary keys of the participating entity sets, and any attributes of the relationship set.



team_id	...
1	...
2	...

Team(team_id, ...)

team_id	sponsor_id	sponsor_date
1	1	2013-1-1
2	1	2013-9-1

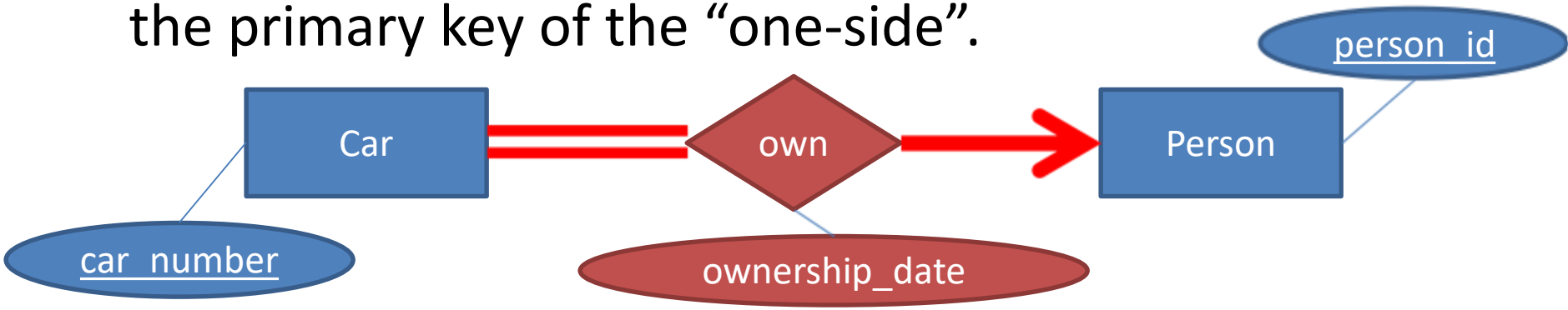
Team_asoc_sponsor
(team_id, sponsor_id, sponsor_date)

sponsor_id	...
1	...
2	...

Sponsor(sponsor_id, ...)

Relationship sets

- **Many-to-one** and **one-to-many** relationship sets that are **total on the many-side** can be represented by adding extra attributes to the “many-side”, containing the primary key of the “one-side”.



car_number	ownership_date	person_id	…
HV 2299	2013-10-1	1	…
HW 2149	2013-12-4	1	…

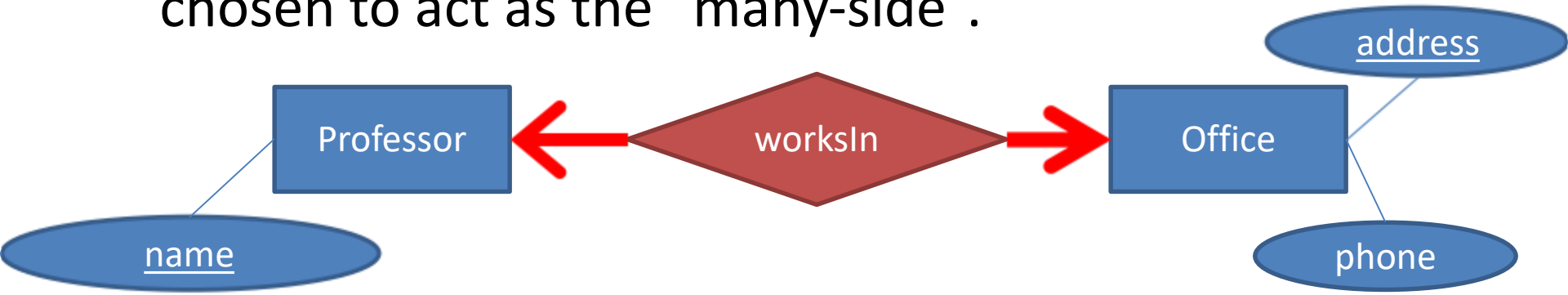
person_id	…
1	…
2	…

Car (car_number, ownership_date, person_id,…)

Person (person_id, …)

Relationship sets

- For **one-to-one** relationship sets, either side can be chosen to act as the "many-side".



name	office.address	…
Professor Kao	CB312	…

Professor(name, **office.address**,…)

address	phone	…
CB312	21234567	…

Office(address, phone, …)

OR

name	…
Professor Kao	…

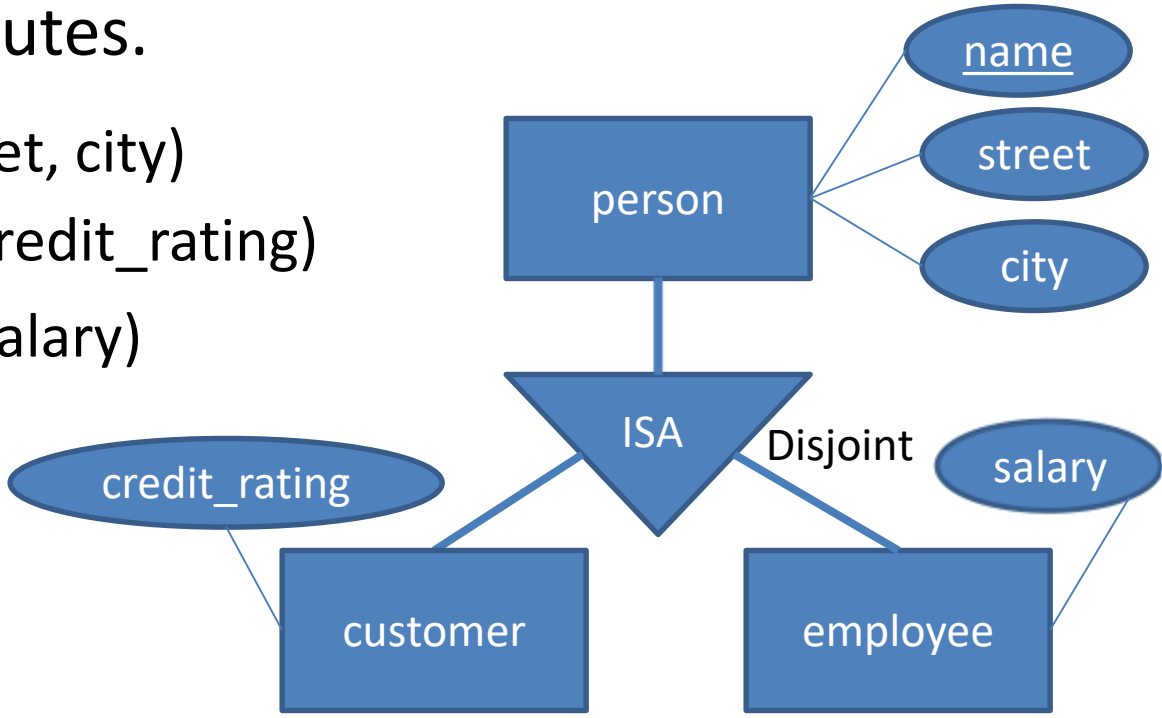
Professor(name, …)

address	professor.name	phone	…
CB312	Professor Kao	21234567	…

Office(address, **professor.name**, phone, …)

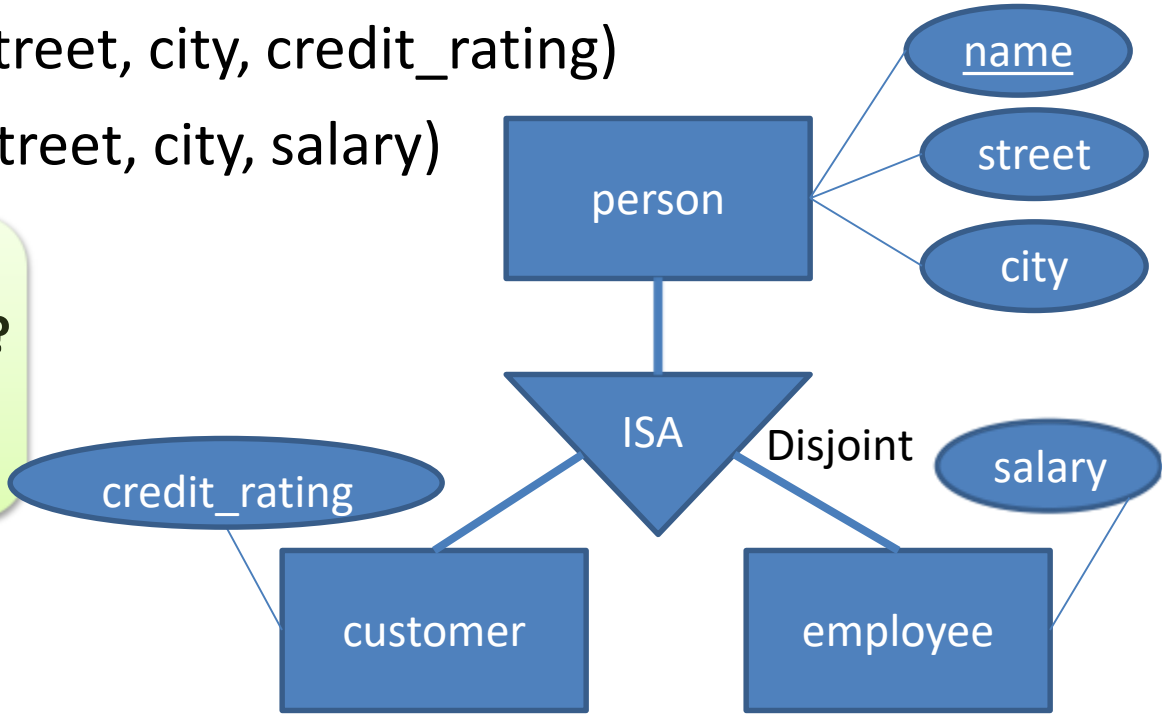
Specialization (method 1)

- Form a table for the **higher-level entity set**.
- Form a table for each **lower-level entity set**, which contains the **primary key** of the higher-level entity set and local attributes.
  - Person(<u>name</u>, street, city)
  - Customer(<u>name</u>, credit_rating)
  - Employee(<u>name</u>, salary)



Specialization (method 2)

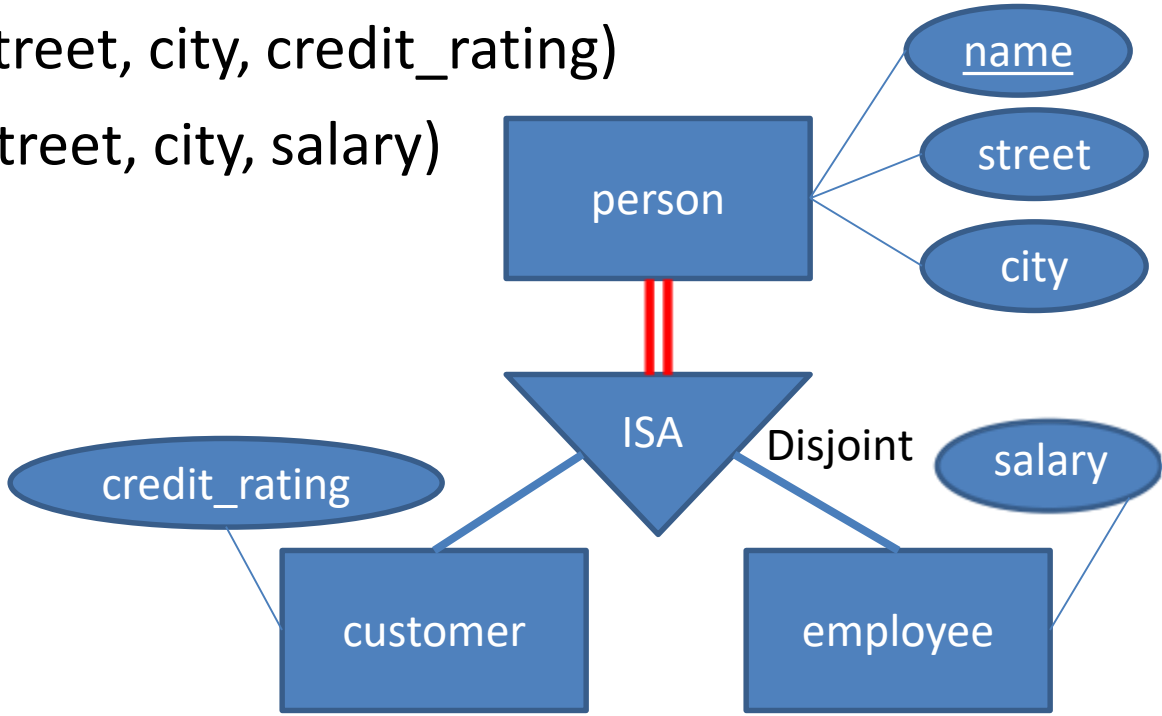
- Form a table for each entity set with all local and inherited attributes.
  - Person(<u>name</u>, street, city)
  - Customer(<u>name</u>, street, city, credit_rating)
  - Employee(<u>name</u>, street, city, salary)



What are the advantage and disadvantage of method 1 and 2?
1. Storage redundancy?
2. Efficiency in retrieving data?

Specialization

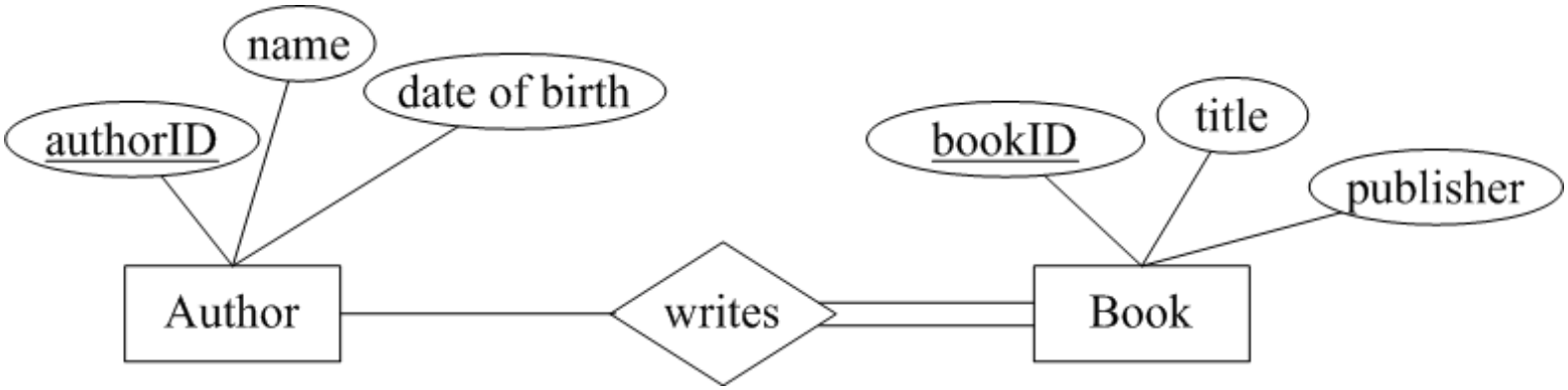
- **Observation:** If the specialization is **total**, the generalized entity set may not require a table!
  - ~~Person(name, street, city)~~
  - Customer(name, street, city, credit_rating)
  - Employee(name, street, city, salary)



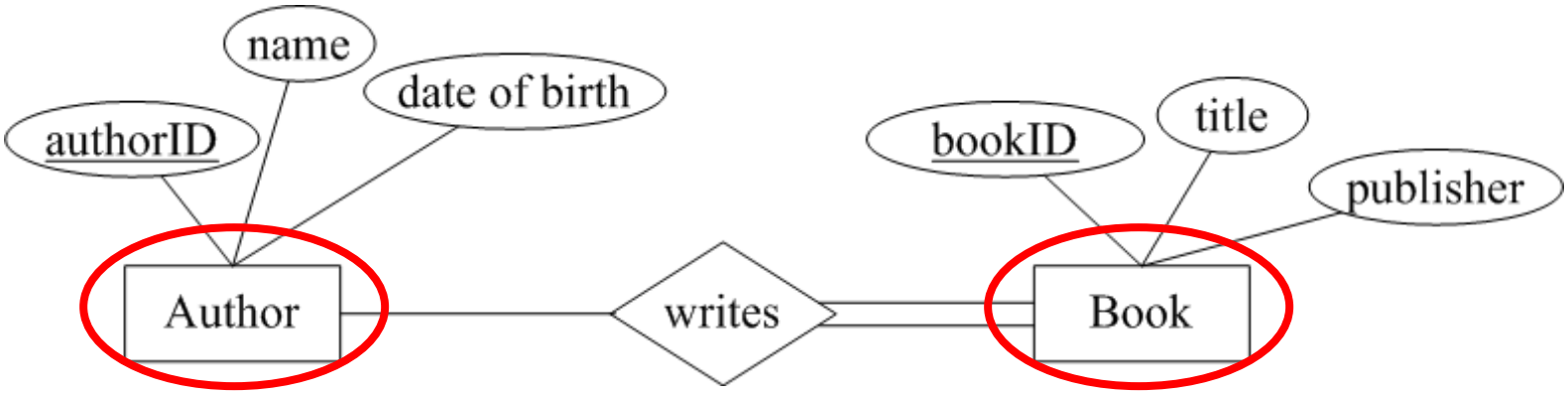
Section 3.3

Foreign Key

Example 1


name
date of birth
authorID
bookID
title
publisher
Author
writes
Book

Example 1



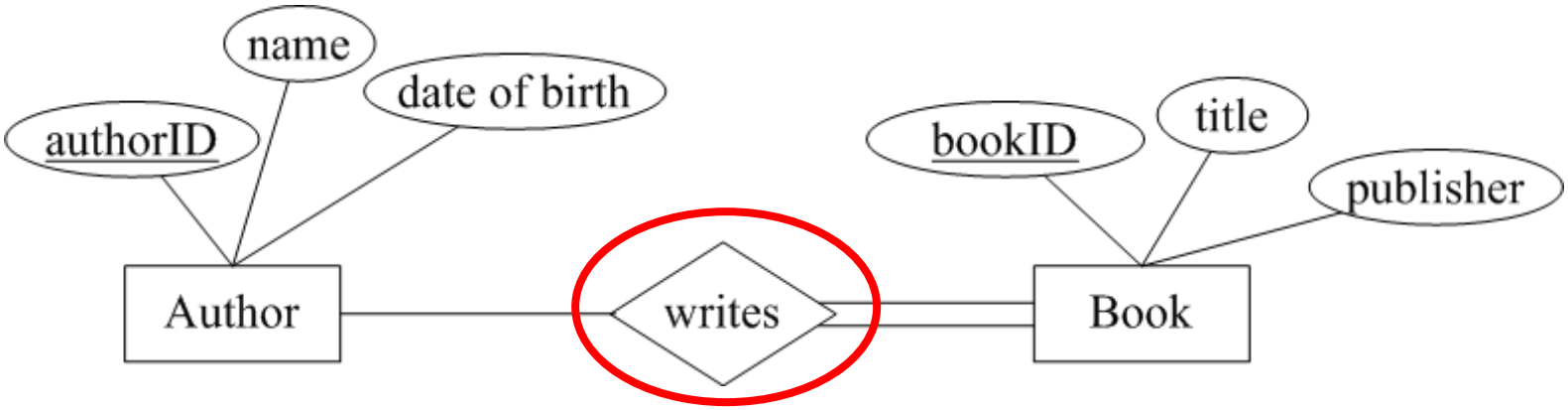
- Author (authorID, name, date of birth)
- Book (bookID, title, publisher)

To transform an ER model to relational tables…

Step 1. Entity set -> table

Each **entity set** becomes a **table**.
Each **attribute** becomes a **column**.
Each **entity** is a **tuple** in the table.

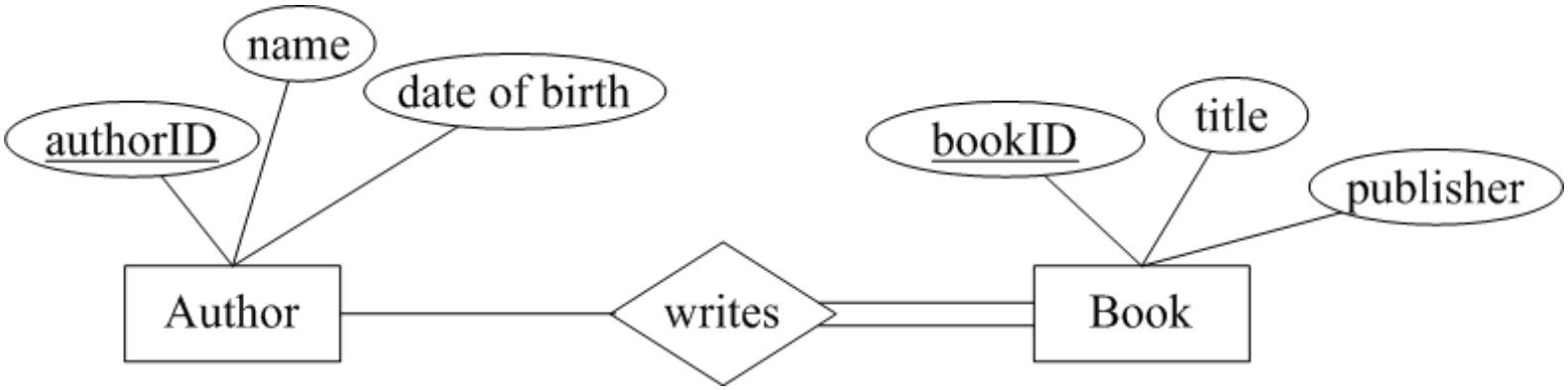
Example 1



- Author (authorID, name, date of birth)
- Book (bookID, title, publisher)
- Writes ( )

Step 2. Relationship set
Whether a relationship set becomes a table or not depends on the **mapping cardinality** of the relationship.
(many to many) , a table.

Example 1



- Author (authorID, name, date of birth)
- Book (bookID, title, publisher)
- Writes (**authorID, bookID**)

Step 3. Identify the key
What is the **primary key** of each table? Any **foreign keys**?

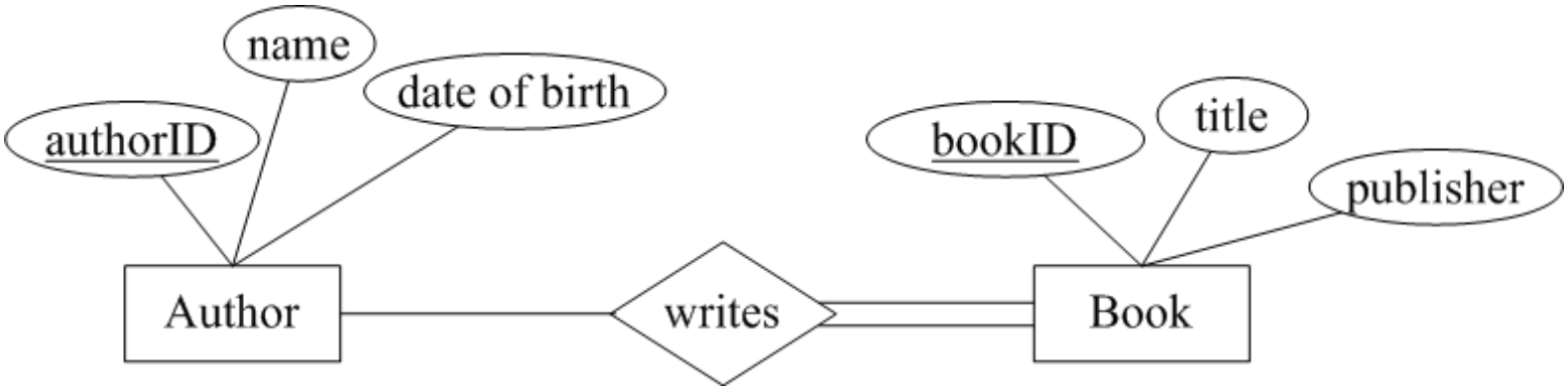
bookID is a **Foreign key** , this key is referencing the column bookID in the Book table.

authorID is another **Foreign key** , this key is referencing the column authorID in the Author table

Foreign key

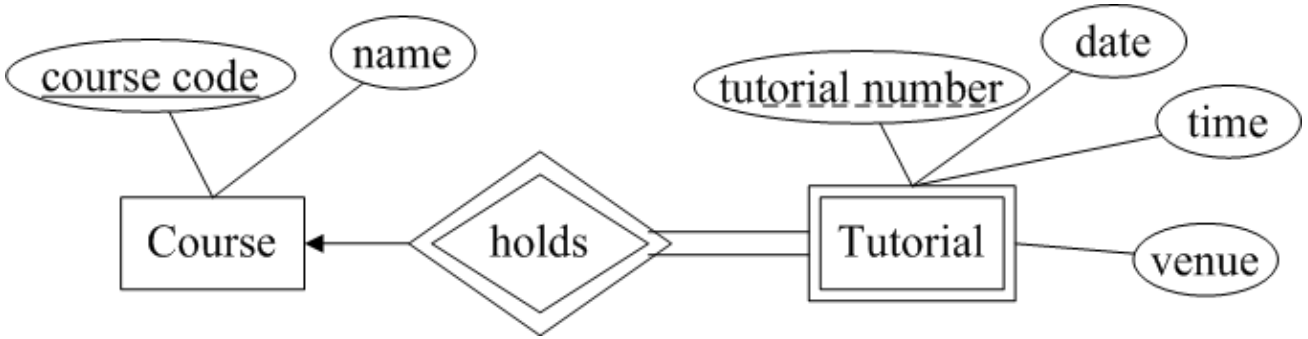
- A **foreign key** is a **referential constraint** between two tables.
- A foreign key is a field in a relational table that matches a candidate key of another table.
- **The foreign key can be used to cross-reference tables.**
  - It is used to link information together.
  - An essential part of ***database normalization*** (To be discussed in Chapter 5).

Example 1

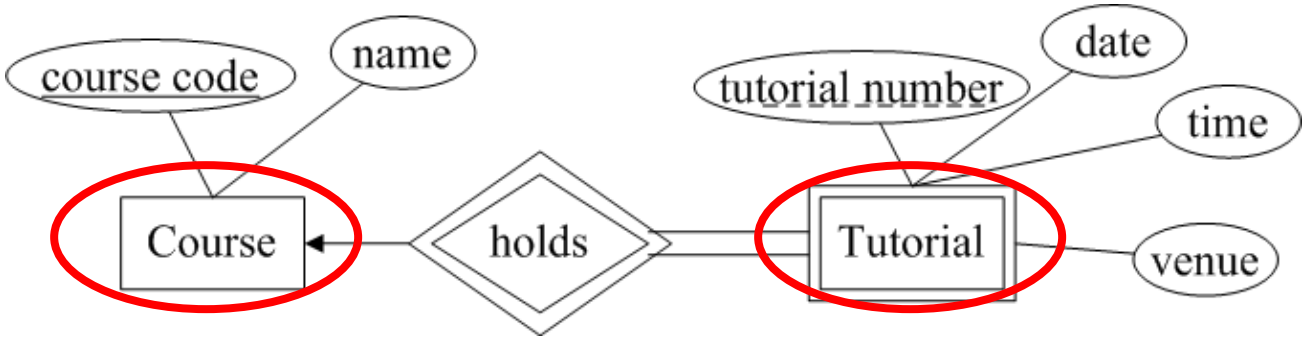


- Author (<u>authorID</u>, name, date of birth)
  - Foreign key: none
- Book (<u>bookID</u>, title, publisher)
  - Foreign key: none
- Writes (<u>authorID</u>, <u>bookID</u>)
  - Foreign keys: {authorID} referencing Author
    {bookID} referencing Book

Example 2


course code
name
tutorial number
date
time
Course
holds
Tutorial
venue

Example 2



To transform an ER model to relational tables…

Step 1. Entity set -> table

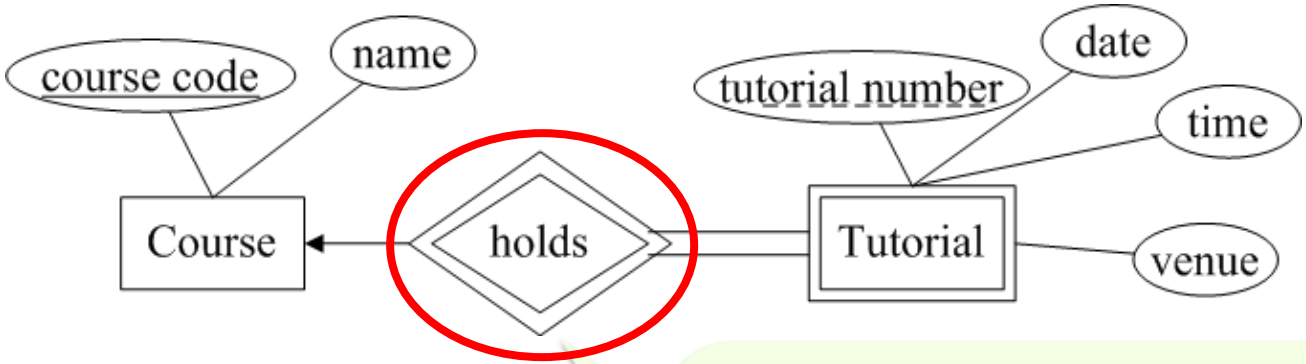
Each **entity set** becomes a **table**.

Each **attribute** becomes a **column**.

Each **entity** is a **tuple** in the table.

- Course (course code, name)
- Tutorial (tutorial number, date, time, venue , course code)

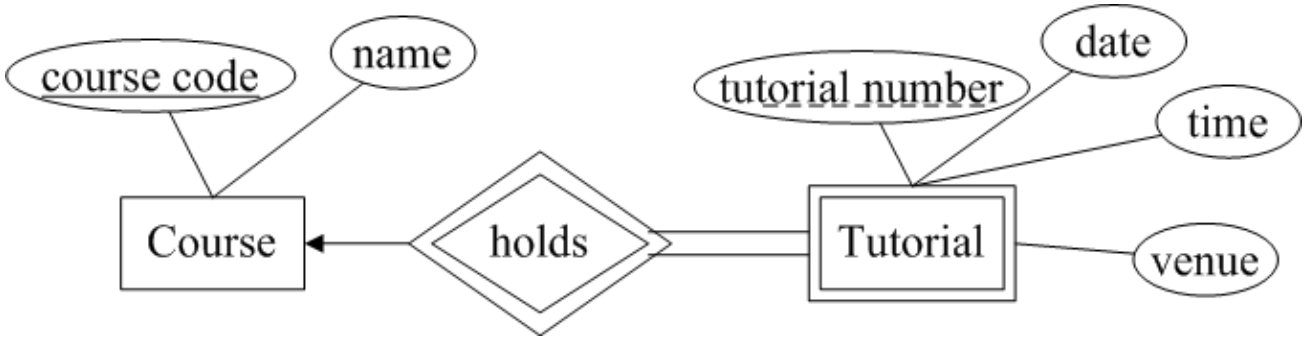
Example 2



Step 2. Relationship set
Whether a relationship set becomes a table or not depends on the mapping cardinality of the relationship.
(**one to many** or **many to one**), attributes go to **"many" side.**

- Course (course code, name)
- Tutorial (tutorial number, date, time, venue, **course_code**)

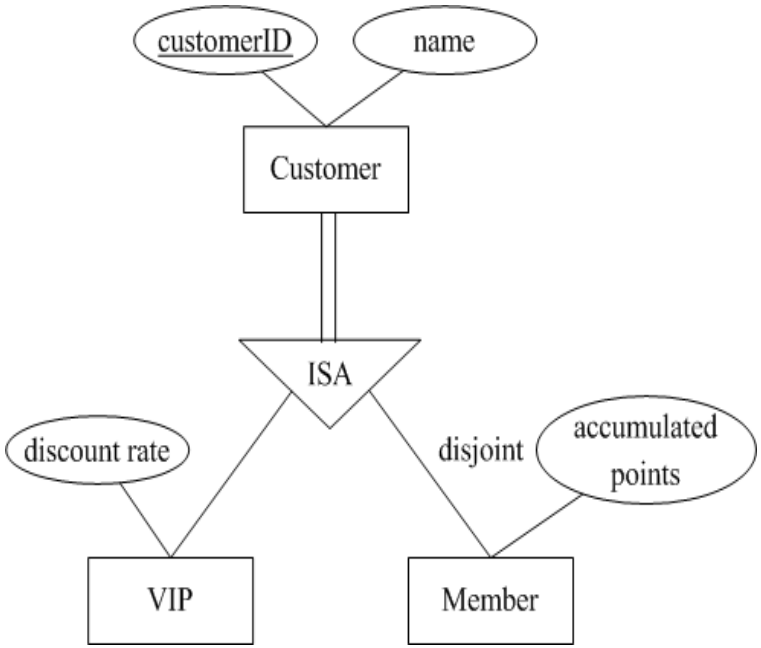
Example 2



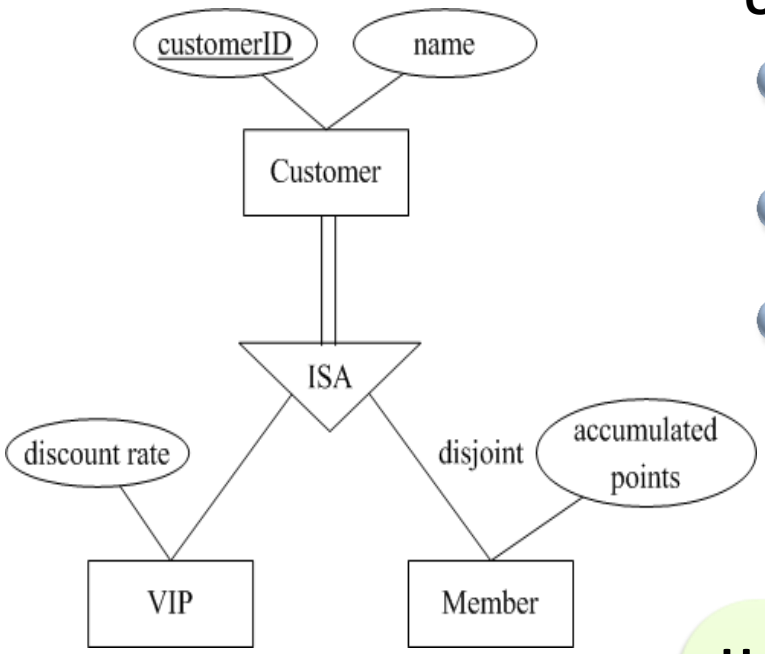
Step 3. Identify the key
What is the **primary key** of each table? Any **foreign keys**?

- Course (course code, name)
  - Foreign key: none
- Tutorial (tutorial number, date, time, venue, course_code)
  - Foreign key: {course code} referencing Course

Example 3


customerID
name
Customer
ISA
discount rate
disjoint
accumulated points
VIP
Member

Example 3



Option 1

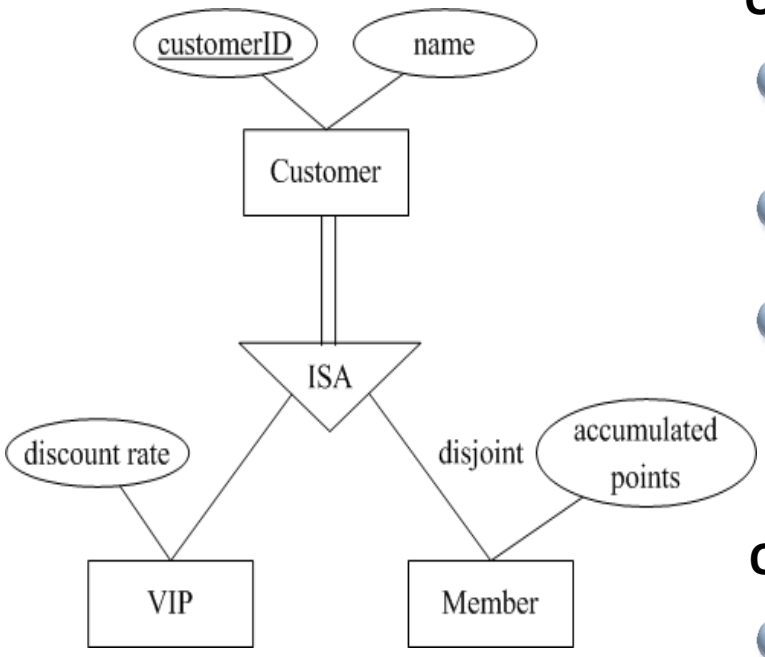
- Customer (customerID, name)
  Foreign key: none
- VIP (customerID, discount_rate)
  Foreign key: {customerID} referencing Customer
- Member (customerID, accumulated_points)
  Foreign key: {customerID} referencing Customer

Handling ISA relationship

Option 1 :

Form a table for higher-level entity set.
Form a table for each lower-level entity set, which contains the primary key of the higher-level entity set and local attributes.

Example 3



Option 1

- Customer (<u>customerID</u>, name)
  Foreign key: none
- VIP (<u>customerID</u>, discount_rate)
  Foreign key: {customerID} referencing Customer
- Member (<u>customerID</u>, accumulated_points)
  Foreign key: {customerID} referencing Customer

Option 2

- Customer (<u>customerID</u>, name)
  Foreign key: none
- VIP (<u>customerID</u>, name, discount_rate)
  Foreign key: {customerID} referencing Customer
- Member (<u>customerID</u>, name, accumulated_points)
  Foreign key: {customerID} referencing Customer

Handling ISA relationship

Option 2 :

Form a table for each entity set with all local and inherited attributes

Example 3

[Storage] Option 1 has less storage redundancy.
[Efficiency] Accessing data (e.g, retrieving the name and discount_rate of a VIP) in option 1 requires accessing two tables (not as efficient as option 2, which requires accessing one table only)!

What are the **advantage** and **disadvantage** of these

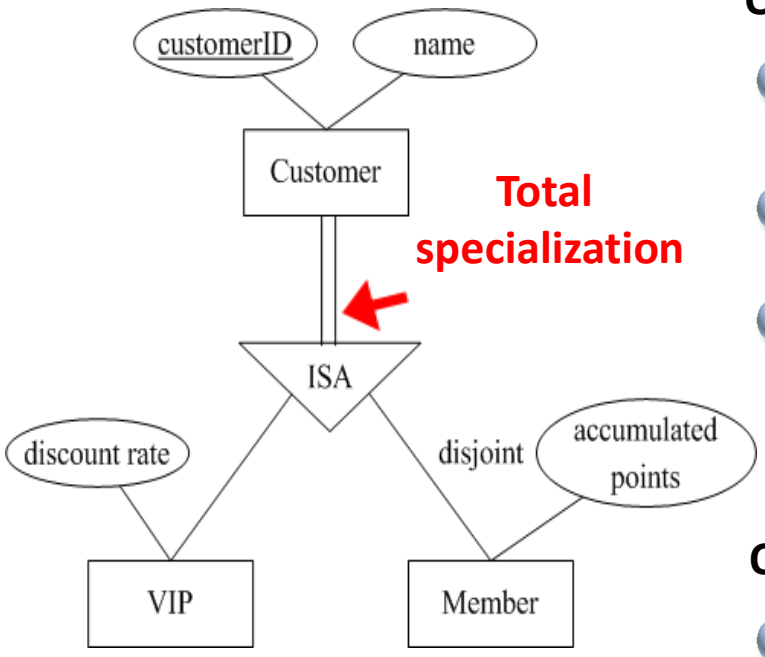
Option 1

- Customer (<u>customerID</u>, name)
  Foreign key: none
- VIP (<u>customerID</u>, discount_rate)
  Foreign key: {customerID} referencing Customer
- Member (<u>customerID</u>, accumulated_points)
  Foreign key: {customerID} referencing Customer

Option 2

- Customer (<u>customerID</u>, name)
  Foreign key: none
- VIP (<u>customerID</u>, name, discount_rate)
  Foreign key: {customerID} referencing Customer
- Member (<u>customerID</u>, name, accumulated_points)
  Foreign key: {customerID} referencing Customer

Example 3



Option 1

- Customer (customerID, name)
  Foreign key: none
- VIP (customerID, discount_rate)
  Foreign key: {customerID} referencing Customer
- Member (customerID, accumulated_points)
  Foreign key: {customerID} referencing Customer

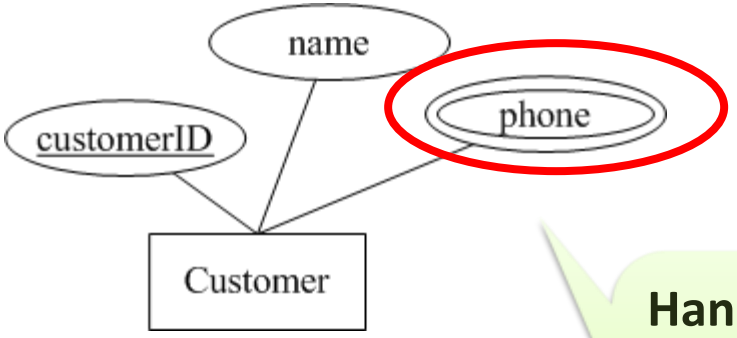
Option 2

- Customer (customerID, name)
  Foreign key: none
- VIP (customerID, name, discount_rate)
  Foreign key: {customerID} referencing Customer
- Member (customerID, name, accumulated_points)
  Foreign key: {customerID} referencing Customer

Option 3

- VIP (customerID, name, discount rate)
  Foreign key: none
- Member (customerID, name, accumulated points)
  Foreign key: none

Example 4



Handling Multivalue attributes
Multivalue attribute becomes a table.

- Customer (customerID, name)
  - Foreign key: none
- CustomerPhone (customerID, phone)
  - Foreign key: {customerID} referencing Customer

Lecture 3

END

COMP3278A

Introduction to Database Management Systems

Dr. Ping Luo

Email : pluo@cs.hku.hk

Department of Computer Science, The University of Hong Kong