

# COMP 3311

# DATABASE MANAGEMENT

# SYSTEMS

## TUTORIAL 1

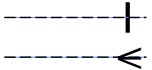
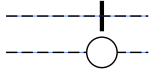

## ENTITY-RELATIONSHIP (E-R) MODEL

## AND DATA BASE DESIGN

# REVIEW: E-R MODEL & DATABASE DESIGN

- Generally, we construct an E-R diagram, by identifying:
  - Entity types  $\Rightarrow$  what should be entities?
    - Strong or weak?
    - Attributes
      - Can they be inherited (generalization or specialization)?
  - Relationship types  $\Rightarrow$  how should entities be related?
    - Participation constraints
      - Total participation or partial participation?
    - Cardinality constraints
      - One to one, one to many or many to many?
    - Any attributes?
    - Need to label roles?
- E-R Diagram Notation
  - Many different notations; no standard E-R notation!

# REVIEW: E-R DIAGRAM LECTURE NOTATION

Entity	➤ entity type
<u>Relationship</u>	➤ relationship type
attribute	➤ single-valued attribute
{attribute}	➤ multivalued attribute
attribute ()	➤ derived attribute
	➤ cardinality constraint (one or many)
	➤ participation constraint (total or partial)
Weak entity	➤ weak entity type
<u>Relationship</u>	➤ identifying relationship for weak entity type
	➤ generalization

## **EXERCISE 1: BANK APPLICATION**

We want to record account and loan information for a bank's customers.

- For each customer we store an id, name, address, which is composed of street, city and state, and one or more phone numbers.
- For each account we store a unique account number and the balance.
- For a saving account we store the interest rate while for a checking account we store whether it has overdraft protection.
- An account can be held by several customers and a customer can hold several accounts.
- For each loan that a customer takes out we record a number and amount.
- A loan may require a guarantor, who must also be a bank customer.
- Each loan can have several payments for which we record a number, date and amount.
- A customer can either hold an account or take out a loan or both.

**Construct an E-R diagram for the bank application.  
Identify all keys of entities and constraints on relationships.**

## EXERCISE 1: BANK APPLICATION—ENTITIES

- For each **customer** we store an id, name, address, which is composed of street, city and state, and one or more phone numbers.
- For each **account** we store a unique account number and the balance.
- For a **saving** account we store the interest rate while for a **checking** account we store whether it has overdraft protection.
- An account can be held by several customers and a customer can hold several accounts.
- For each **loan** that a customer takes out we record a number and amount.
- A loan may require a guarantor, who must also be a bank customer.
- Each loan can have several **payments** for which we record a number, date and amount.
- A customer can either hold an account or take out a loan or both.



# EXERCISE 1: BANK APPLICATION— ATTRIBUTES AND KEYS OF ENTITIES

- For each **customer** we store an id, **name**, **address**, which is composed of **street**, **city** and **state**, and one or more **phone numbers**.
- For each **account** we store a unique account number and the **balance**.
- For a **saving** account we store the **interest rate** while for a **checking** account we store whether it has **overdraft** protection.
- An account can be held by several customers and a customer can hold several accounts.
- For each **loan** that a customer takes out we record a number and **amount**.
- A loan may require a guarantor, who is also a customer of the bank.
- Each loan can have several **payments** for which we record a **number**, **date** and **amount**.
- A customer can either hold an account or take out a loan or both.

Customer
<u>id</u> name address street city state {phoneNo}

Account
<u>accountNo</u> balance

Saving
interestRate

Checking
overdraft

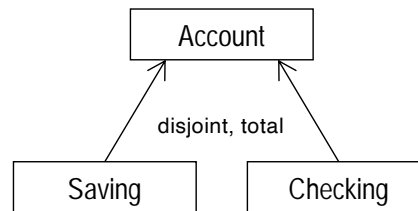
Loan
<u>loanNo</u> amount

Payment
paymentNo date amount



# EXERCISE 1: BANK APPLICATION— ENTITY GENERALIZATION/SPECIALIZATION

- For each **account** we store a unique account number and the balance.
- For a **saving** account we store the interest rate while for a **checking** account we store whether it has overdraft protection.



**What should be the generalization?**  $\Rightarrow$  **Account** superclass;  
Saving, Checking subclasses

**What should be the coverage constraint?**  $\Rightarrow$  disjoint, total

Customer

Account

Saving



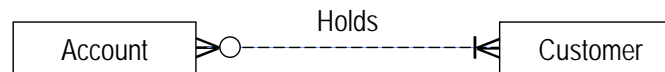
Checking

Loan

Payment

# EXERCISE 1: BANK APPLICATION—RELATIONSHIPS

- An **account** can be **held by** several **customers** and a customer can hold several accounts.
- A customer can either hold an account or take out a loan or both.



**What should be related?** ⇒ Account related to Customer

**What should be the cardinality constraints?**

- ⇒ Account **many** (An account can be held by several customers.)  
Customer **many** (A customer can hold several accounts.)

**What should be the participation constraints?**

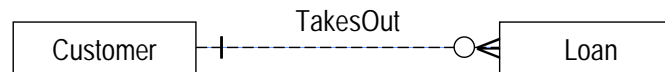
- ⇒ Account **total** (Every account must be held by a customer—common sense.)  
Customer **partial** (A customer may take out a loan only and hold no account.)





# EXERCISE 1: BANK APPLICATION—RELATIONSHIPS

- For each **loan** that a **customer** **takes out** we record a number and amount.
- A customer can either hold an account or take out a loan or both.



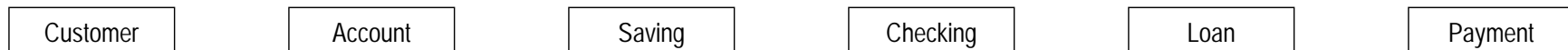
**What should be related?** ⇒ Customer related to Loan

**What should be the cardinality constraints?**

- ⇒ Customer **many** (A customer could take out several loans—common sense.)  
Loan **unknown** (Could be 1 or many—need to verify with client.)

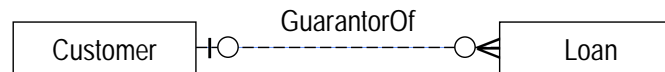
**What should be the participation constraints?**

- ⇒ Customer **partial** (A customer may hold an account only and have no loan.)  
Loan **total** (Every loan must be taken out by a customer—common sense.)



# EXERCISE 1: BANK APPLICATION—RELATIONSHIPS

- A **loan** may require a **guarantor**, who must also be a bank **customer**.



**What should be related?** ⇒ Customer related to Loan

**What should be the cardinality constraints?**

- ⇒ Customer **many** (A customer may be a guarantor for many loans—common sense.)  
Loan **1** (A loan requires only one guarantor—implied by statement.)

**What should be the participation constraints?**

- ⇒ Customer **partial** (A customer may not be a guarantor of any loan.)  
Loan **partial** (Not every loan requires a guarantor—implied by statement.)

Customer

Account

Saving



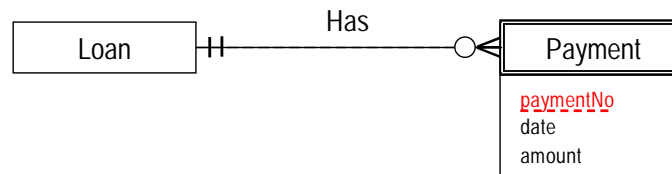
Checking

Loan

Payment

# EXERCISE 1: BANK APPLICATION—RELATIONSHIPS

- Each **loan** can **have** several **payments** for which we record a number, date and amount.



**What should be related?**  $\Rightarrow$  Loan related to Payment

**What kind of entity is Payment?**  $\Rightarrow$  Weak entity dependent on Loan.

**Is there a discriminator for Payment?**  $\Rightarrow$  Yes — paymentNo.

**What should be the cardinality constraints?**

$\Rightarrow$  Loan **many** (Each loan can have several payments.)

Payment **1** (Every payment is for only one loan—common sense.)

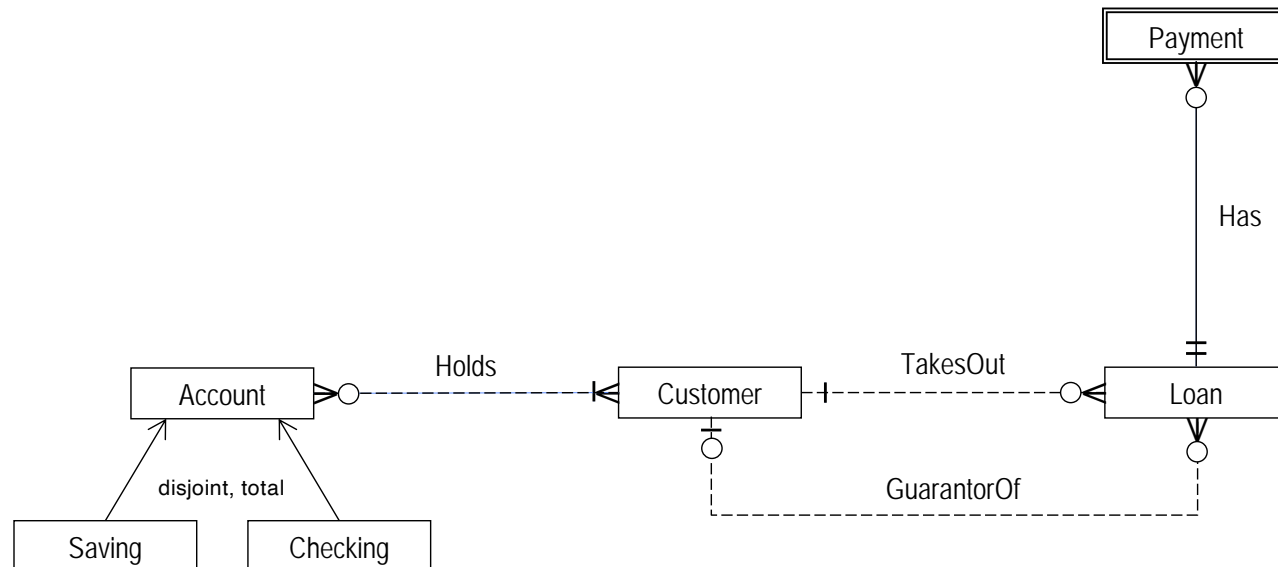
**What should be the participation constraints?**

$\Rightarrow$  Loan **partial** (A loan may not have any payment yet—common sense.)

Payment **total** (Every payment must be for a loan—common sense.)



# EXERCISE 1: BANK APPLICATION—E-R DIAGRAM



Customer
<u>id</u>
name
address
street
city
state
{phoneNo}

Account
<u>accountNo</u>
balance

Saving
interestRate

Checking
overdraft

Loan
<u>loanNo</u>
amount

Payment
<u>paymentNo</u>
date
amount



## EXERCISE 2: FACTORY APPLICATION

We want to record information about products that a factory manufactures.

- The factory has a number of employees. For each employee we store the employee id, name and salary.
- Each employee must be an admin staff or a worker, but not both.
- Admin staff must take seminars. For each seminar we store its id, name and date. For the admin staff, we store the grade received for each seminar taken.
- The factory manufactures a number of products and each product is identified by a product id and has a name.
- A worker is assigned to work on exactly one product; a product has multiple (one or more) workers assigned to it.
- A large number of items are manufactured for each product. Each item has a serial number and a color. Different items of the same product have different serial numbers. However, two items that belong to different products may have the same serial number.

**Construct an E-R diagram for the factory application.  
Identify all keys of entities and constraints on relationships.**

## EXERCISE 2: FACTORY APPLICATION— ENTITIES AND ATTRIBUTES

- The factory has a number of **employees**. For each employee we store the employee id, **name** and **salary**.
- Each employee must be an **admin staff** or a **worker**, but not both.
- Admin staff must take **seminars**. For each seminar we store its id, **name** and **date**. For the admin staff, we store the grade received for each seminar taken.
- The factory manufactures a number of **products** and each product is identified by a product id and has a **name**.
- A worker is assigned to work on exactly one product; a product has multiple (one or more) workers assigned to it.
- A large number of **items** are manufactured for each product. Each item has a serial number and a **color**. Different items of the same product have different serial numbers. However, two items that belong to different products may have the same serial number.

Employee
<u>empld</u> name salary

AdminStaff
------------

Worker
--------

Seminar
<u>id</u> name date

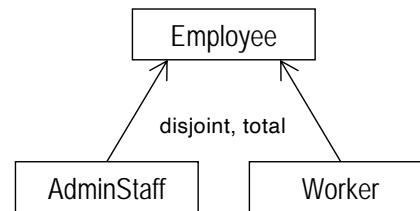
Product
<u>id</u> name

Item
<u>serialNo</u> color



## EXERCISE 2: FACTORY APPLICATION— RELATIONSHIPS

- The factory has a number of **employees**. For each employee we store the **employee id**, **name** and **salary**.
- Each employee must be an **admin staff** or a **worker**, but not both.

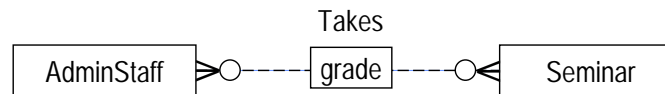


**How to relate the entities?** ⇒ use generalization

**What should be the coverage constraint?** ⇒ disjoint, total

## EXERCISE 2: FACTORY APPLICATION— RELATIONSHIPS

- **Admin staff** must take **seminars**. For each seminar we store its **id**, **name** and **date**. For the admin staff, we store the **grade** received for each seminar taken.



### What should be the cardinality constraints?

- ⇒ Seminar **many** (A seminar can be taken by many admin staff—common sense.)
- AdminStaff **many** (An admin staff can take many seminars—common sense.)

### What should be the participation constraints?

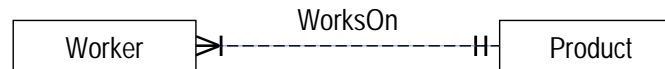
- ⇒ Seminar **partial** (A Seminar instance should be able to exist before any admin staff take it.)
- AdminStaff **partial** (Although admin staff must take seminars, an AdminStaff instance should be able to exist before having taken any seminar.)

**Anything else?** ⇒ Add the attribute **grade** to the **Takes** relationship.



## EXERCISE 2: FACTORY APPLICATION— RELATIONSHIPS

- The factory manufactures a number of **products** and each product is identified by a product **id** and has a **name**.
- A **worker** is assigned to **work on** exactly one **product**; a product has multiple (one or more) workers assigned to it.



**What should be the participation constraint for Worker?**

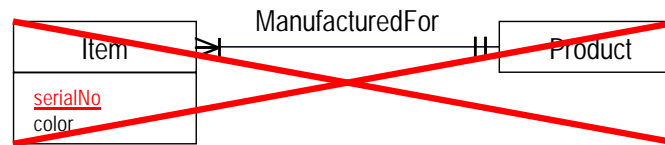
⇒ **total** (A worker is assigned to work on **exactly one** product.)

**What should be the participation constraint for Product?**

⇒ **total** (A product has multiple (**one** or more) workers assigned to it.)

## EXERCISE 2: FACTORY APPLICATION— RELATIONSHIPS

- A large number of **items** are **manufactured for** each **product**. Each item has a **serial number** and a **color**. Different items of the same product have different serial numbers. However, two items that belong to different products may have the same serial number.

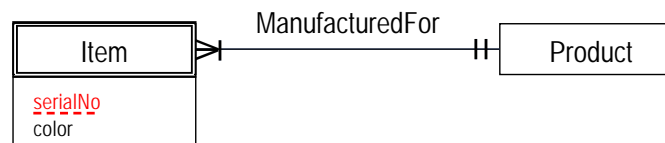


**Is this representation correct?**

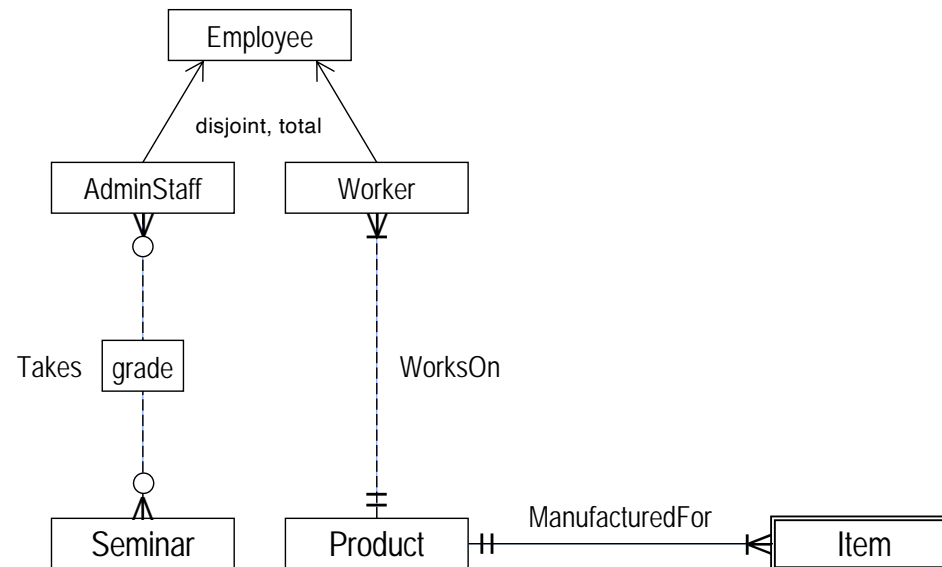
**Consider:** two items that belong to **different products** may have the **same serial number**.

⇒ Item is a weak entity dependent on Product.

⇒ serialNo is a discriminator attribute.



# EXERCISE 2: FACTORY APPLICATION— E-R DIAGRAM



Employee
<u>empld</u>
name
salary

AdminStaff
------------

Worker
--------

Seminar
<u>id</u>
name
date

Product
<u>id</u>
name

Item
<u>serialNo</u>
color