• ER is a model for the **logical level**
  - it describes the structure of the database at a high abstraction level

• A database can be modeled as
  - a collection of **entities**
  - a **relationship** among entities

**An entity** is an object that exists independently and is distinguishable from other objects

- **Examples:**
  • an employee, a company, a car, a student, a class etc.
  • color, age, etc. are not entities
An entity set is a set of entities of the same type

- Examples
  - a set of employees, a set of departments
- also called entity types

Entity Type: Employee

Entity Set:

\[ e_1, e_2, e_3, \ldots \]

A general specification

The actual employees
An attribute is a property of an entity or a relationship

- **Examples:**
  - name, address, weight, height are properties of a *Person* entity
  - date of marriage is a property of a relationship *Marriage*

- **Attribute types:**
  - Simple
  - Composite
  - Multivalued
  - Derived
  - Key
• Simple attribute: contains a single value
- **Composite attribute**: consists of several components (e.g., address)
• Multivalued attribute: contains more than one value
- **Derived attribute:** computed from other attributes (e.g., age can be computed from the date of birth and the current date)
• **Key attribute**: uniquely identifies an entity
More on Key Attributes

• An entity may have **more than one key**
  - A minimal set of attributes that uniquely identifies an entity is called a **candidate key**
    • **Question:** which are possible candidate keys for HKUST students?
  - Only **one candidate key** is selected to be the **primary key**
    • **Question:** which is the primary key for HKUST students?

• Sometimes **artificial keys** maybe created
  - **Example:** assume that we want to store information about the current offering of CSIT5300. We can select a unique number (e.g., 1235) to serve as the key.
  - **Question:** which are possible alternatives for this example, without introducing additional attributes?

• **Composite Key:** contains two or more attributes
E-R diagram for entity **customer**

- **customer-id**
- **first-name**
- **last-name**
- **middle-initial**
- **name**
- **phone-number**
- **date-of-birth**
- **age**
- **street-name**
- **street-number**
- **apartment-number**
- **street**
- **city**
- **state**
- **zip-code**

**customer**
A relationship is an association among several entities

- The **degree** refers to the number of entity sets that participate in the relationship set
  - Relationship sets that involve **two** entity sets are called **binary** (or of degree two)
  - Relationship sets among **more than two** entity sets are called **ternary**
• *borrower* is a relationship between *customer* and *loan*
  - it means that a customer can be associated with one or more loans and vice versa
• **depositor** is a relationship between **customer** and **accounts**
• **access-date** is an attribute of **depositor**
We express **cardinality constraints** by drawing either a directed line \((\rightarrow)\), signifying “one”, or an undirected line \((\rightarrow\rightarrow)\), signifying “many”, between the relationship set and the entity set.

**One-to-one relationship**
- A customer is associated with at most one loan via the relationship *borrower*
- A loan is associated with at most one customer via *borrower*
One-to-many relationship
- a loan is associated with at most one customer via borrower
- a customer is associated with several (including 0) loans via borrower
Many-to-one relationship
- a loan is associated with several (including 0) customers via *borrower*
- a customer is associated with at most one loan via *borrower*
Many-to-many relationship

- A customer is associated with several (possibly 0) loans via borrower
- A loan is associated with several (possibly 0) customers via borrower
Participation of an Entity Set in a Relationship Set

- **Total participation** (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., participation of *loan* in *borrower* is total
    every loan must have at least a customer associated to it via *borrower*

- **Partial participation**: some entities may not participate in any relationship in the relationship set
  - E.g., participation of *customer* in *borrower* is partial
    some customers may not have any loans
Cardinality Limits

- The edge between *loan* and *borrower* has a cardinality limit of 1..1
  - the minimum and the maximum cardinality are both 1
  - each loan must have exactly one associated customer
- The limit 0..* on the edge from *customer* to *borrower* indicates that a customer can have zero or more loans.

- Observe that
  - the relationship *borrower* is one to many from *customer* to *loan*
  - the participation of *loan* in *borrower* is total

- Hence, cardinality limits can express cardinality and participation constraints
• Entity sets of a relationship need not be distinct
• The labels “manager” and “worker” are called roles; they specify how employee entities interact via the works-for relationship set
• Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles
• Role labels are optional, and are used to clarify semantics of the relationship
The combination of primary keys of the participating entity sets forms a super key of a relationship set:
- \((\text{customer-id}, \text{account-number})\) is the super key of \textit{depositor}.
- This means that a pair of entities can have at most one relationship in a particular relationship set.
  - Problem: if we wish to track all access dates to each account by each customer, we cannot assume a relationship for each access.
  - Solution: use a multivalued attribute for access dates.
- Must consider the mapping cardinality of the relationship set when deciding the candidate keys.
• Example:
  - Suppose that employees of a bank have jobs (responsibilities) at multiple branches, with different jobs at different branches
  - Then there is a ternary relationship set between entity sets employee, job and branch
Some relationships that appear to be ternary may be better represented using binary relationships

- **Example:** A ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
  
  - Using two binary relationships allows partial information (e.g. only mother being known)

But there are some relationships that are naturally ternary

- **Example:** *works-on*
In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.

**Procedure (degree 3)**
- Replace \( R \) between entity sets \( A, B \) and \( C \) by an entity set \( E \), and three relationship sets: \( R_A \) relating \( E \) and \( A \), \( R_B \) relating \( E \) and \( B \) and \( R_C \) relating \( E \) and \( C \).
- Create a special identifying attribute for \( E \).
- Add any attributes of \( R \) to \( E \).
- For each relationship \((a_i, b_i, c_i)\) in \( R \), (i) create a new entity \( e_i \) in the entity set \( E \), (ii) add \((e_i, a_i)\) to \( R_A \), (iii) add \((e_i, b_i)\) to \( R_B \), (iv) add \((e_i, c_i)\) to \( R_C \).
### Example

Converting Ternary Relationships to Binary Form

<table>
<thead>
<tr>
<th></th>
<th>a₁</th>
<th>b₁</th>
<th>c₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>c₁</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>c₂</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>b₂</td>
<td>c₂</td>
<td></td>
</tr>
</tbody>
</table>

Constructing Binary Relations:

**R_a**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>e₁</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>e₂</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>e₃</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>e₄</td>
<td></td>
</tr>
</tbody>
</table>

**R_b**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>e₁</td>
<td></td>
</tr>
<tr>
<td>b₁</td>
<td>e₂</td>
<td></td>
</tr>
<tr>
<td>b₁</td>
<td>e₃</td>
<td></td>
</tr>
<tr>
<td>b₂</td>
<td>e₄</td>
<td></td>
</tr>
</tbody>
</table>

**R_c**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁</td>
<td>e₁</td>
<td></td>
</tr>
<tr>
<td>c₁</td>
<td>e₂</td>
<td></td>
</tr>
<tr>
<td>c₂</td>
<td>e₃</td>
<td></td>
</tr>
<tr>
<td>c₂</td>
<td>e₄</td>
<td></td>
</tr>
</tbody>
</table>
Weak Entity Sets

• An entity set that does not have a primary key is referred to as a weak entity set

• The existence of a weak entity set depends on the existence of an identifying entity set
  – it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  – this relationship is called identifying relationship

• The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes the entities in the weak entity set that depend on a particular strong entity set

• The primary key of a weak entity set is formed by
  – the primary key of the (strong) identifying entity set
  – the weak entity set’s discriminator
Weak Entity Sets

• A weak entity set is depicted by a double rectangle.
• An identifying relationship is depicted by a double diamond.
• We underline the discriminator of a weak entity set with a dashed line.
• Example:
  - discriminator of payment: payment-number
  - Primary key for payment: (loan-number, payment-number)
Another example:

- A child may not be old enough to have a HKID number
- Even if he/she has a HKID number, the company may not be interested in keeping it in the database.
ISA (‘is a’) Hierarchies

• If we declare $A$ ISA $B$, every $A$ entity is also considered to be a $B$ entity
• As in C++ (and other program languages), attributes are inherited

• Reasons for using ISA:
  - To add descriptive attributes specific to a subclass
  - To identify entities that participate in a relationship
ISA (‘is a’) Hierarchies

- **Overlap constraints:**
  - Can Joe be both an `Hourly_Emps` and a `Contract_Emps` entity?

- **Covering constraints:**
  - Does every `Employees` entity also have to be an `Hourly_Emps` or a `Contract_Emps` entity?
Summary of Symbols

- **Many to Many Relationship**: \( R \)
- **Many to One Relationship**: \( R \)
- **One to One Relationship**: \( R \)
- **Cardinality Limits**: \( R \rightarrow E \) with limits 1..h
- **Role Name**: \( R \rightarrow E \)
- **Role Indicator**: \( R \rightarrow E \)
- **ISA (Specialization or Generalization)**
  - Total Generalization: \( ISA \)
  - Disjoint Generalization: \( ISA_{disjoint} \)
• **Example**
  - For each employee we want to store the office number, location of the office (e.g., Building A, Floor 6), and telephone.
  - Several employees share the same office

**Office as attribute**

**Office as entity**
• **Example**
  - Can you see some differences?
  - E.g., can you have accounts without a customer?

• **A possible guideline**
  - A relationship set describes an *action* that occurs between entities

Account as an entity

![Account as an entity diagram](image)

Account as relationship

![Account as relationship diagram](image)
Example:
- We want to record the period that an employee works for some department
- Differences below?
• Example:
  - an account must be associated with exactly one branch
  - two different branches are allowed to have accounts with the same number