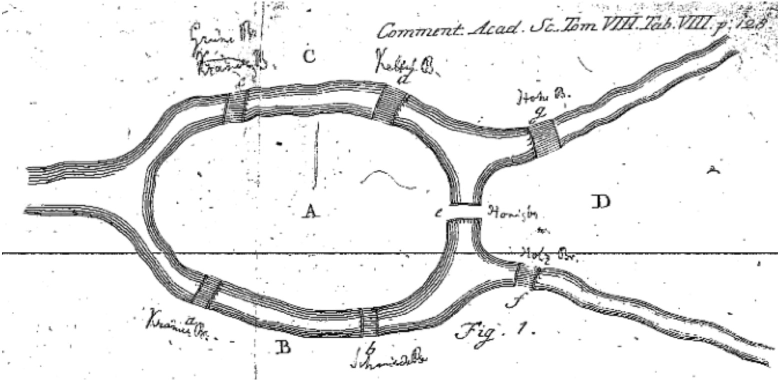
Week 4 Workshop



Housekeeping information

- SQL Assessment (Assignment 1) will be available on Wattle 23:59 tonight, and the submission via Wattle is due 23:59 Sep 3 (Friday, Week 6)
  - **Individual, no group work!**
  - **Do not post any idea/partial solution/result on Wattle.**
  - **Do not wait until the last minute to check/submit your solution.**
  - Sample SQL questions/solutions will be available on Wattle.
  - The correctness of queries does not depend on any database state.
  - Partial marks may be awarded.

Housekeeping information

- SQL Assessment (Assignment 1) will be available on Wattle 23:59 tonight, and the submission via Wattle is due 23:59 Sep 3 (Friday, Week 6)
  - **Individual, no group work!**
  - **Do not post any idea/partial solution/result on Wattle.**
  - **Do not wait until the last minute to check/submit your solution.**
  - Sample SQL questions/solutions will be available on Wattle.
  - The correctness of queries does not depend on any database state.
  - Partial marks may be awarded.
- Drop-in sessions for Assignment 1 (Week 5 and Week 6)
  - Aug 23 (Mon) 2-3 pm (NEW)
  - Aug 24 (Tue) 2-3 pm
  - Aug 25 (Wed) 8-9 pm (NEW)
  - Aug 30 (Mon) 2-3 pm (NEW)
  - Aug 31 (Tue) 2-3 pm
  - Sep 1 (Wed) 8-9 pm (NEW)

Database Design – Four Phases

- The database design process has **four phases**:

  1. **Requirements Collection and Analysis**
  2. **Conceptual Design**
     Entity-Relationship Model
  3. **Logical Design**
     From Entity-Relationship Model to Relation Schemas
  4. **Physical Design**

Phase 2: Conceptual Design

- **Conceptual design** is the process of constructing a conceptual data model that is
  - modeled at a high-level of abstraction;
  - sufficiently simple and often graphical;
  - used to communicate the requirements of a database with nontechnical users.
- A conceptual data model is built using the information in users' requirements specification.

  Note: The conceptual design is based on the **Entity-Relationship Model** in this course.

Model and Modeling

- **What is a model?**

Model and Modeling

- **What is a model?**

  A model is

  - a simplification of reality
  - often a graphical depiction of data
  - associated with a modeling language

Model and Modeling

- **What is a model?**

  A model is

  - a simplification of reality
  - often a graphical depiction of data
  - associated with a modeling language

- **What does modeling do?**

Model and Modeling

- **What is a model?**

  A model is

  - a simplification of reality
  - often a graphical depiction of data
  - associated with a modeling language

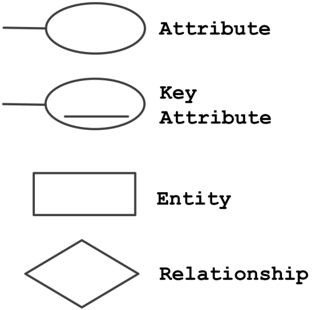
- **What does modeling do?**

  Modeling

  - creates an understanding and relationships of components of a system
  - helps in conceptualising and visualising the structure of a system that we may want to build.
  - facilitates specifications of the behaviour of a system
  - gives rise to a template that guides us in constructing a system
  - . . .

Entity-Relationship (ER) Model

- ER diagrams (Peter Chen in 1976):
  - **Attribute** as *oval*;
  - **Key attribute** with *underlined*;
  - **Entity** as *rectangles*;
  - **Relationship** as *diamonds*.



(Exercise 1) Consider the following data requirements for a university student database that is used to keep track of students' transcripts.

- The university keeps track of each student's name, student number, social security number, address, phone, and birthdate. Both social security number and student number have unique values for each student.
- Each student has exactly one major, and may have a minor (if any) with departments.
- Each department has name, department code, office number, office phone, and college. Both name and code have unique values for each department.
- Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of course number is unique for each course.
- Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.
- A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each student has name, student number, social security number, address, phone and birthdate. Both social security number and student number have unique values for each student.

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each student has name, student number, social security number, address, phone and birthdate. Both social security number and student number have unique values for each student.

Question: What are the entities, relationships and attributes?

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each student has name, student number, social security number, address, phone and birthdate. Both social security number and student number have unique values for each student.

Question: What are the entities, relationships and attributes?

- **Entities:** STUDENT
- **Relationships:**
- **Attributes**: name, student number, social security number, address, phone and birthdate for STUDENT

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Entities, Relationships and Attributes

- **Entities:** “Things” in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each student has exactly one major, and may have a minor (if any) with departments

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each student has exactly one major, and may have a minor (if any) with departments

Question: What are the entities, relationships and attributes?

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each student has exactly one major, and may have a minor (if any) with departments

Question: What are the entities, relationships and attributes?

- **Entities:** STUDENT, DEPARTMENT
- **Relationships:** has_major_with between STUDENT and DEPARTMENT, has_minor_with between STUDENT and DEPARTMENT
- **Attributes**: name for has_major_with, name for has_minor_with

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each student has exactly one major, and may have a minor (if any) with departments.

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each student has exactly one major, and may have a minor (if any) with departments.

Question: What are the constraints on relationship "**has_major_with**"?

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each student has exactly one major, and may have a minor (if any) with departments.

Question: What are the constraints on relationship "**has_major_with**"?

Cardinality ratios: Every student has at most **one** major and a department may offer **many** majors (to different students)

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each student has exactly one major, and may have a minor (if any) with departments.

Question: What are the constraints on relationship "**has_major_with**"?

Cardinality ratios: Every student has at most **one** major and a department may offer **many** majors (to different students)

Participation constraints: Every student **must** have one major (**total**) and each department **must** (typically) offer one major (**total**).

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each student has exactly one major, and may have a minor (if any) with departments.

Question: What are the constraints on relationship "**has_minor_with**"?

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each student has exactly one major, and may have a minor (if any) with departments.

Question: What are the constraints on relationship "**has_minor_with**"?

Cardinality ratios: Every student has at most **one** minor and a department may offer **many** minor (to different students)

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each student has exactly one major, and may have a minor (if any) with departments.

Question: What are the constraints on relationship "**has_minor_with**"?

Cardinality ratios: Every student has at most **one** minor and a department may offer **many** minor (to different students)

Participation constraints: Every student **may or may not** have one minor (**partial**) and each department **must** (typically) offer one minor (**total**).

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each course has a course name, description, course number, number of semester hours, level, and offering department.

Entities, Relationships and Attributes

- **Entities:** “Things” in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each course has a course name, description, course number, number of semester hours, level, and offering department.

Question: What are the entities, relationships and attributes?

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each course has a course name, description, course number, number of semester hours, level, and offering department.

Question: What are the entities, relationships and attributes?

- **Entities:** course

Australian National University

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

Each course has a course name, description, course number, number of semester hours, level, and offering department.

Question: What are the entities, relationships and attributes?

- **Entities:** course, department
- **Relationships:** offer (between **department** and **course**)
- **Attributes**: course name, description, course number, number of semester hours and level (of the entity **course**)

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each course has a course name, description, course number, number of semester hours, level, and offering department.

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

  Each course has a course name, description, course number, number of semester hours, level, and offering department.

  Question: What are the constraints on relationship "**offer**"?

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each course has a course name, description, course number, number of semester hours, level, and offering department.

Question: What are the constraints on relationship "**offer**"?

Cardinality ratios: Every course is offered by at most **one** department and a department may offer **many** courses

Constraints on Relationships

- **Cardinality ratios**: Specifies the *maximum* number of relationships that an entity can participate in.
- **Participation constraints** (total, partial): Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Each course has a course name, description, course number, number of semester hours, level, and offering department.

Question: What are the constraints on relationship "**offer**"?

Cardinality ratios: Every course is offered by at most **one** department and a department may offer **many** courses

Participation constraints: Every course **must** be offered by some department (**total**) and each department **may (or may not)** offer any courses (**partial**).

Entities, Relationships and Attributes

- **Entities:** “Things” in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

  Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.

  A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

Entities, Relationships and Attributes

- **Entities:** “Things” in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

  Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.

  A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

  Question: What are the entities, relationships and attributes?

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

  Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.

  A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

  Question: What are the entities, relationships and attributes?

  - **Entities:** section, course, student

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

  Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.

  A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

  Question: What are the entities, relationships and attributes?

  - **Entities:** section, course, student
  - **Relationships:** section_taught (between **section** and **course**), grade_record (between **student** and **section**)

Australian National University

Entities, Relationships and Attributes

- **Entities:** "Things" in the real world (with independent existence).
- **Relationships:** Associations between entities.
- **Attributes:** Properties that describe entities and relationships.

  Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.

  A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

  Question: What are the entities, relationships and attributes?

  - **Entities:** section, course, student
  - **Relationships:** section_taught (between **section** and **course**), grade_record (between **student** and **section**)
  - **Attributes**: instructor, semester, year, and section number (of the **weak** entity **section**), final mark and letter grade (of the relationship **grade_record**)

(Exercise 1) Consider the following data requirements for a university student database that is used to keep track of students' transcripts.

- The university keeps track of each student's name, student number, social security number, address, phone, and birthdate. Both social security number and student number have unique values for each student.
- Each student has exactly one major, and may have a minor (if any) with departments.
- Each department has name, department code, office number, office phone, and college. Both name and code have unique values for each department.
- Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of course number is unique for each course.
- Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.
- A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

Constructing an ER or EER Model

- Identify the entities (including weak entity types)

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  student, course, department, section (weak entity)

Constructing an ER or EER Model

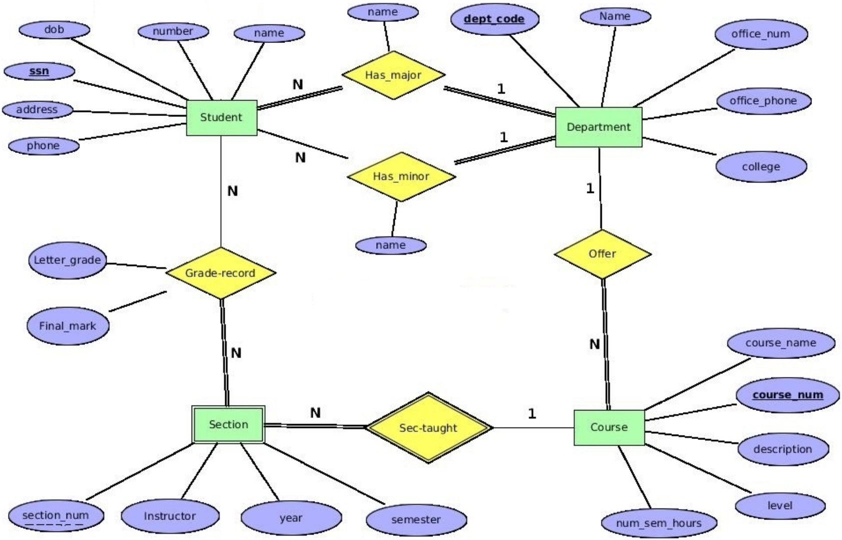
- Identify the entities (including weak entity types)
  student, course, department, section (weak entity)
- Identify the relationships

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  student, course, department, section (weak entity)
- Identify the relationships
  - **has_major** (between **student** and **department**)
  - **has_major** (between **student** and **department**)
  - **offer** (between **department** and **course**)
  - **section_taught** (between **section** and **course**)
  - **grade_record** (between **student** and **section**)

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  student, course, department, section (weak entity)
- Identify the relationships
  - **has_major** (between **student** and **department**)
  - **has_major** (between **student** and **department**)
  - **offer** (between **department** and **course**)
  - **section_taught** (between **section** and **course**)
  - **grade_record** (between **student** and **section**)
- Identify the attributes of entities and relationships and identify a primary key for each entity type
- Identify cardinality ratios and participation constraints on relationships


name
dept_code
Name
office_num
dob
number
name
ssn
N
Has_major
1
address
Student
Department
office_phone
phone
1
N
Has_minor
college
N
1
name
Letter_grade
Grade-record
Offer
Final_mark
N
N
course_name
course_num
N
1
Section
Sec-taught
Course
description
level
section_num
Instructor
year
semester
num_sem_hours

Software tool to draw ER diagram

- We require students to use an academic tool, TerraER, to draw the ER diagrams.
- TerraER allows you to save your ER diagrams into xml files and export your ER diagrams as a JPEG figure.
- You can download the jar file from the following website: `https://github.com/rterrabh/TerraER/releases/download/TerraER3.01/TerraER3.01beta.jar`
- You can double-click that file to execute on Windows/Mac/Linux (assume that the Java Runtime Environment JRE has been installed).
- More information on how to use TerraER will be provided next week.

(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells products in both local shops and webstores on the Internet. Each local shop has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each webstore. Every product has a unique productID, a description, an item price, and a quantity in stock. The database application should also record customers' details such as their name, address and email. Every customer is assigned a unique ID. A customer may place an order that consists of at least one product and each order is from either a shop or a webstore. Customers have three payment options (i.e., cash, paypal, and credit card) but for each order only one payment option can be chosen. A delivery may be requested for each order. After full-payment is received, a delivery would be sent out subject to products' availability. Every delivery has a unique tracking number.

(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells **products** in both local **shops** and **webstores** on the Internet. Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**. Every **product** has a unique productID, a description, an item price, and a quantity in stock. The database application should also record **customers**' details such as their name, address and email. Every **customer** is assigned a unique ID. A **customer** may place an **order** that consists of at least one **product** and each **order** is from either a **shop** or a **webstore**. **Customers** have three payment options (i.e., cash, paypal, and credit card) but for each **order** only one payment option can be chosen. A **delivery** may be requested for each **order**. After full-payment is received, a **delivery** would be sent out subject to **products**' availability. Every **delivery** has a unique tracking number.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**

- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - The company sells products in both local **shops** and **webstores** on the Internet.
  - Each **order** is associated with either a **shop** or a **webstore**.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**

- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - The company sells products in both local **shops** and **webstores** on the Internet.
  - Each **order** is associated with either a **shop** or a **webstore**.

  - subclass **shop**, **webstore**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**

- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - The company sells products in both local **shops** and **webstores** on the Internet.
  - Each **order** is associated with either a **shop** or a **webstore**.

  - subclass **shop**, **webstore**
  - superclass **store**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**

- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - The company sells products in both local **shops** and **webstores** on the Internet.
  - Each **order** is associated with either a **shop** or a **webstore**.

  - subclass **shop**, **webstore**
  - superclass **store**
  - disjoint and complete

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - The company sells products in both local **shops** and **webstores** on the Internet.
  - Each **order** is associated with either a **shop** or a **webstore**.

  - subclass **shop**, **webstore**
  - superclass **store**
  - disjoint and complete
- Identify the relationships

(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells **products** in both local **shops** and **webstores** on the Internet. Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**. Every **product** has a unique productID, a description, an item price, and a quantity in stock. The database application should also record **customers**' details such as their name, address and email. Every **customer** is assigned a unique ID. A **customer** may place an **order** that consists of at least one **product** and each **order** is from either a **shop** or a **webstore**. **Customers** have three payment options (i.e., cash, paypal, and credit card) but for each **order** only one payment option can be chosen. A **delivery** may be requested for each **order**. After full-payment is received, a **delivery** would be sent out subject to **products**' availability. Every **delivery** has a unique tracking number.

(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells **products** in both local **shops** and **webstores** on the Internet. Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**. Every **product** has a unique productID, a description, an item price, and a quantity in stock. The database application should also record **customers**' details such as their name, address and email. Every **customer** is assigned a unique ID. A **customer** may **place** an **order** that **consists of** at least one **product** and each **order** **is from** either a **shop** or a **webstore**. **Customers** have three payment options (i.e., cash, paypal, and credit card) but for each **order** only one payment option can be chosen. A **delivery** may be requested **for** each **order**. After full-payment is received, a **delivery** would be sent out subject to **products**' availability. Every **delivery** has a unique tracking number.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
  - **customer** **place** **order**
  - **order** **consists of** **product**
  - each order **is from** **store**(superclass) (either subclass **shop** or subclass **webstore**)
  - **delivery** **is for** **order**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - Every **product** has a unique productID, a description, an item price, and a quantity in stock.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - Every **product** has a unique productID, a description, an item price, and a quantity in stock.
  - Attributes for **product**: **productID, description, item price, quantity**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - Every **product** has a unique productID, a description, an item price, and a quantity in stock.
  - Attributes for **product**: **productID, description, item price, quantity**
  - Primary key for **product**: **productID**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - The database application should also record **customers**' details such as their name, address and email. Every **customer** is assigned a unique ID.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - The database application should also record **customers**' details such as their name, address and email. Every **customer** is assigned a unique ID.
  - Attributes for **customer**: **name, address, email, CustomerID**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - The database application should also record **customers**' details such as their name, address and email. Every **customer** is assigned a unique ID.
  - Attributes for **customer**: **name, address, email, CustomerID**
  - Primary key for **customer**: **CustomerID**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**.
  - Attributes for superclass **store**: **name**, **location**/**URL**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**.
  - Attributes for superclass **store**: **name**, **location/URL**
  - Primary key for superclass **store**: <u>**location/URL**</u>

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - subclass **shop**, **webstore**
  - superclass **store**
- Identify the relationships
- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**.
  - Attributes for superclass **store**: **name**, **location/URL**
  - Primary key for superclass **store**: <u>**location/URL**</u>
  - Attributes for subclass **shop**: **phone number, email**
  - Attributes for subclass **webstore**: **last updated date**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **customer** **place** **order**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **customer** **place** **order**
  - ...
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - A **customer** may **place** an **order**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **customer** **place** **order**
  - ...
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - A **customer** may **place** an **order**
  - Cardinality ratios: A customer may **place** **many** orders and an order **is placed by** **one** customer.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **customer** **place** **order**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - A **customer** may **place** an **order**
  - Cardinality ratios: A customer may **place** **many** orders and an order **is placed by** **one** customer.
  - Participation constraints: A **customer** **may or may not** **place** any orders (**Partial**). An **order** **must** **be placed by** one customer (**Total**).

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **delivery** **is for** **order**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **delivery** **is for** **order**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - A **delivery** may be requested **for** each **order**.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **delivery** **is for** **order**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - A **delivery** may be requested **for** each **order**.
  - Cardinality ratios: A delivery **is for** at most **one** order and an order **has** at most **one** delivery.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **delivery** **is for** **order**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - A **delivery** may be requested **for** each **order**.
  - Cardinality ratios: A delivery **is for** at most **one** order and an order **has** at most **one** delivery.
  - Participation constraints: A **delivery** **must** **be for** an order (**Total**). An **order** **may or may not** **have** a delivery (**Partial**).

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **order** **consists of** **product**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **order** **consists of** **product**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - Each order **consists of** at least one **product**

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **order** **consists of** **product**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - Each order **consists of** at least one **product**
  - Cardinality ratios: An order may **contain** **many** products and an product may **be contained** in **many** orders.

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
  - **order** **consists of** **product**
  - . . .
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships
  - Each order **consists of** at least one **product**
  - Cardinality ratios: An order may **contain** **many** products and an product may **be contained** in **many** orders.
  - Participation constraints: A **order** **must** **contain** some product (**Total**). A **product** **may or may not** **be contained** in an order (**Partial**).

Constructing an ER or EER Model

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships

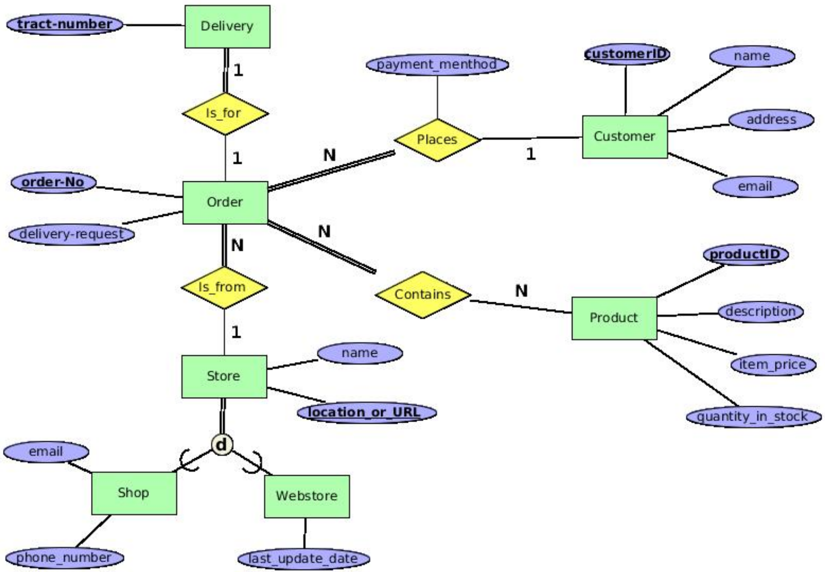
Constructing an ER or EER Model

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships

- **Not all the constraints can be expressed in the ER model**

(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells products in both local shops and webstores on the Internet. Each local shop has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each webstore. Every product has a unique productID, a description, an item price, and a quantity in stock. The database application should also record customers' details such as their name, address and email. Every customer is assigned a unique ID. A customer may place an order that consists of at least one product and each order is from either a shop or a webstore. **Customers have three payment options (i.e., cash, paypal, and credit card) but for each order only one payment option can be chosen.** A delivery may be requested for each order. **After full-payment is received, a delivery would be sent out subject to products' availability**. Every delivery has a tracking number.


tract-number
Delivery
1
Is_for
1
payment_menthod
Places
N
1
customerID
name
Customer
address
email
order-No
Order
delivery-request
N
N
Is_from
Contains
N
productID
Product
description
item_price
quantity_in_stock
1
Store
name
location_or_URL
d
email
Shop
Webstore
phone_number
last_update_date

Phase 3: Logical Design

- **Logical design** is the process of constructing a logical data model (e.g. relational or object-oriented).
- A conceptual data model is translated onto a logical data model, which can be further refined (e.g., normalisation) to meet the data requirements. For example,
  - **From:** An ER model
  - **To:** Relations with their primary and foreign keys, which facilitates SQL to deal with retrieving, updating and deletion.

  Note: The logical design is based on the **relational data model** in this course.

ER-to-Relations Algorithm

- 7-step algorithm to convert the basic ER model into relations, and more steps for the EER model.

  Step 1: Mapping of Regular Entity Types

  Step 2: Mapping of Weak Entity Types

  Step 3: Mapping of Binary 1:1 Relationship Types

  - Foreign key approach
  - Merged relation approach
  - Cross-reference approach

  Step 4: Mapping of Binary 1:N Relationship Types

  Step 5: Mapping of Binary M:N Relationship Types

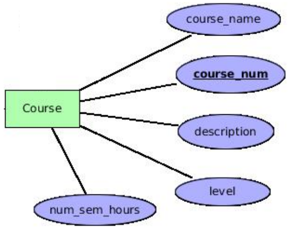
  Step 6: Mapping of Multi-valued Attributes

  Step 7: Mapping of N-ary Relationship Types

  Step 8: Mapping of Superclass/Subclass

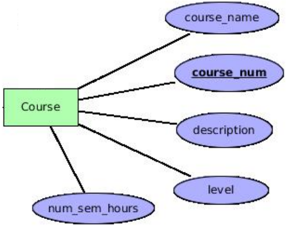
Step 1: Regular Entity types

- For each regular entity type *E*, **create a relation schema** with the attributes of *E* (ignore multi-valued attributes until Step 6), where
  - **PK:** the key attributes of *E*



Step 1: Regular Entity types

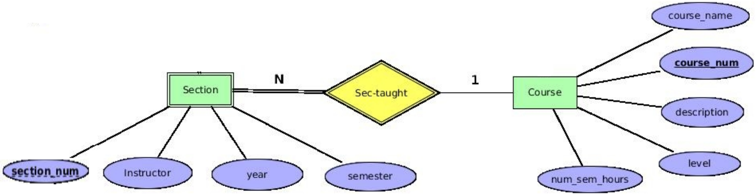
- For each regular entity type *E*, **create a relation schema** with the attributes of *E* (ignore multi-valued attributes until Step 6), where
  - **PK:** the key attributes of *E*



- COURSE(course_num, course_name, description, num_sem_hours, level) with PK: {course_num}
- **Note:** This is not necessarily the final relation schema of COURSE.

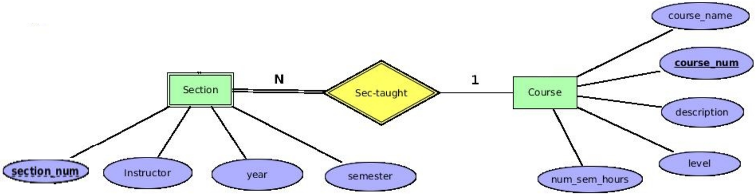
Step 2: Weak Entity Types

- For each weak entity type E_w, **create a relation schema** with the attributes of E_w plus the PK of its identifying entity type, where
  - **PK:** the partial key attributes of E_w plus the PK of its identifying entity type
  - **FK:** references the PK of its identifying entity type



Step 2: Weak Entity Types

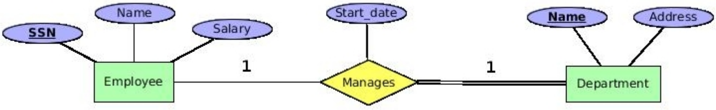
- For each weak entity type E_w, **create a relation schema** with the attributes of E_w plus the PK of its identifying entity type, where
  - **PK:** the partial key attributes of E_w plus the PK of its identifying entity type
  - **FK:** references the PK of its identifying entity type



- SECTION(section_num, instructor, semester, year, course_num)
  with PK: {section_num, course_number}
  with FK: [course_num]$\subseteq$COURSE[course_num]

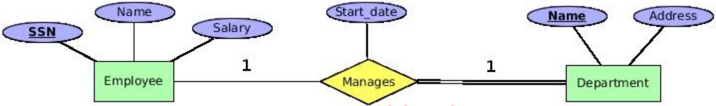
Step 3: Binary 1:1 Relationship Types - (Foreign key)

- For a 1:1 relationship type *R* with one total participation, **extend the relation schema of the total-side entity type** by the attributes of *R* and the PK of the partial-side entity type, where
  - **PK:** still the PK of the total-side entity type
  - **FK:** references the PK of the partial-side entity type



Step 3: Binary 1:1 Relationship Types - (Foreign key)

- For a 1:1 relationship type *R* with one total participation, **extend the relation schema of the total-side entity type** by the attributes of *R* and the PK of the partial-side entity type, where
  - **PK:** still the PK of the total-side entity type
  - **FK:** references the PK of the partial-side entity type



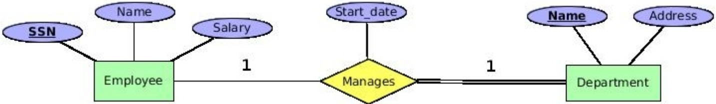
- DEPARTMENT(Name, Address, Mgr_SSN, Start_date) with

  PK: {Name}

  FK: [Mgr_SSN]$\subseteq$EMPLOYEE[SSN].

Step 3: Binary 1:1 Relationship Types - (Foreign key)

- For a 1:1 relationship type *R* with one total participation, **extend the relation schema of the total-side entity type** by the attributes of *R* and the PK of the partial-side entity type, where
  - **PK:** still the PK of the total-side entity type
  - **FK:** references the PK of the partial-side entity type



- DEPARTMENT(Name, Address, Mgr_SSN, Start_date) with

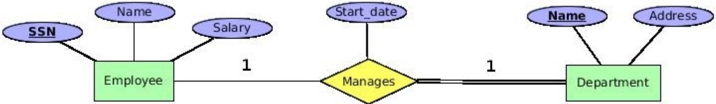
  PK: {Name}

  FK: [Mgr_SSN]$\subseteq$EMPLOYEE[SSN].
- How can we model the total participation?

Australian National University

Step 3: Binary 1:1 Relationship Types - (Foreign key)

- For a 1:1 relationship type *R* with one total participation, **extend the relation schema of the total-side entity type** by the attributes of *R* and the PK of the partial-side entity type, where
  - **PK:** still the PK of the total-side entity type
  - **FK:** references the PK of the partial-side entity type



- DEPARTMENT(Name, Address, Mgr_SSN, Start_date) with

  PK: {Name}

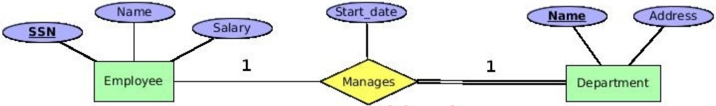
  FK: [Mgr_SSN]$\subseteq$EMPLOYEE[SSN].

- How can we model the total participation?

  Add `NOT NULL` constraint to Mgr_SSN for total participation.

Step 3: Binary 1:1 Relationship Types - (Foreign key)

- For a 1:1 relationship type *R* with one total participation, **extend the relation schema of the total-side entity type** by the attributes of *R* and the PK of the partial-side entity type, where
  - **PK:** still the PK of the total-side entity type
  - **FK:** references the PK of the partial-side entity type



- DEPARTMENT(Name, Address, Mgr_SSN, Start_date) with

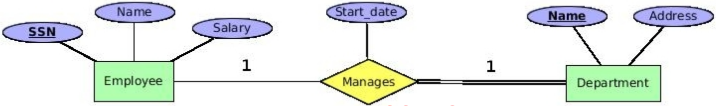
  PK: {Name}

  FK: [Mgr_SSN]$\subseteq$EMPLOYEE[SSN].

- Why don't we extend the relation schema of the partial-side entity type?

Step 3: Binary 1:1 Relationship Types - (Foreign key)

- For a 1:1 relationship type *R* with one total participation, **extend the relation schema of the total-side entity type** by the attributes of *R* and the PK of the partial-side entity type, where
  - **PK:** still the PK of the total-side entity type
  - **FK:** references the PK of the partial-side entity type



- DEPARTMENT(Name, Address, Mgr_SSN, Start_date) with

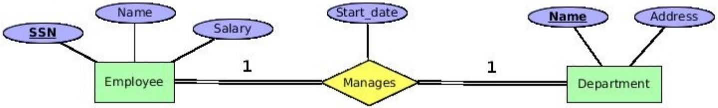
  PK: {Name}

  FK: [Mgr_SSN]⊆EMPLOYEE[SSN].
- Why don't we extend the relation schema of the partial-side entity type?

  This may cause many `NULL` values.

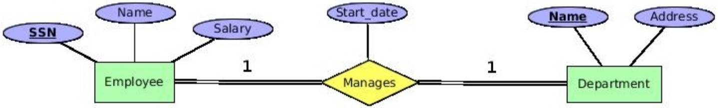
Step 3: Binary 1:1 Relationship Types - (Merged relation)

- How can we translate the following kind of 1:1 relationship type?



Step 3: Binary 1:1 Relationship Types - (Merged relation)

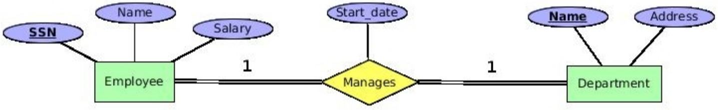
- How can we translate the following kind of 1:1 relationship type?



- If participation on both sides is total, we may **merge the relation schemas of both entity types and the attributes of the relationship type into a single relation**.
- EMPLOYEE-DEP(SSN, Name, Salary, Start_date, Dname, Address) with PK: {SSN} or {Dname}

Step 3: Binary 1:1 Relationship Types - (Merged relation)

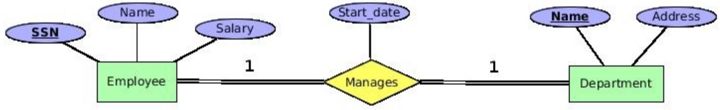
- How can we translate the following kind of 1:1 relationship type?



- If participation on both sides is total, we may **merge the relation schemas of both entity types and the attributes of the relationship type into a single relation**.
- EMPLOYEE-DEP(SSN, Name, Salary, Start_date, Dname, Address) with PK: {SSN} or {Dname}
- How can we model the total participations?

Step 3: Binary 1:1 Relationship Types - (Merged relation)

- How can we translate the following kind of 1:1 relationship type?

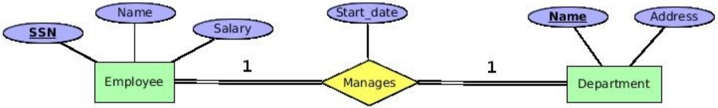


- If participation on both sides is total, we may **merge the relation schemas of both entity types and the attributes of the relationship type into a single relation**.
- EMPLOYEE-DEP(SSN, Name, Salary, Start_date, Dname, Address) with PK: {SSN} or {Dname}
- How can we model the total participations?

  Add `NOT NULL` constraint to both SSN and Dname for total participations.

Step 3: Binary 1:1 Relationship Types - (Merged relation)

- How can we translate the following kind of 1:1 relationship type?

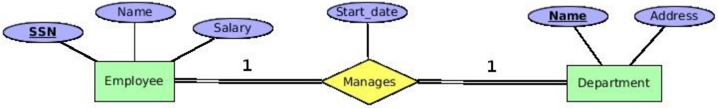


- If participation on both sides is total, we may **merge the relation schemas of both entity types and the attributes of the relationship type into a single relation**.
- EMPLOYEE-DEP(SSN, Name, Salary, Start_date, Dname, Address) with PK: {SSN} or {Dname}
- How can we model the total participations?

  Add `NOT NULL` constraint to both SSN and Dname for total participations.
- **Is merging them always a good solution?**

Step 3: Binary 1:1 Relationship Types - (Merged relation)

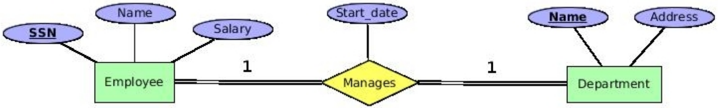
- How can we translate the following kind of 1:1 relationship type?



- If participation on both sides is total, we may **merge the relation schemas of both entity types and the attributes of the relationship type into a single relation**.
- However, **merging them is not always a good solution**. Why?

Step 3: Binary 1:1 Relationship Types - (Merged relation)

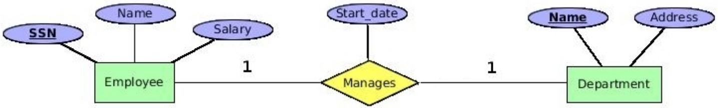
- How can we translate the following kind of 1:1 relationship type?



- If participation on both sides is total, we may **merge the relation schemas of both entity types and the attributes of the relationship type into a single relation**.
- However, **merging them is not always a good solution**. Why?
  (1) The two entity types represent different entities in the real world.
  (2) The two entity types participate in different relationship types.
  (3) Having separate relation schemas for two entity types often leads to more efficient updates than a single relation schema.
  (4) ...

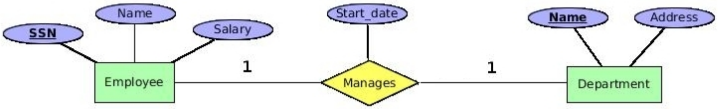
Step 3: Binary 1:1 Relationship Types - (Cross-reference)

- How can we translate the following kind of 1:1 relationship type?



Step 3: Binary 1:1 Relationship Types - (Cross-reference)

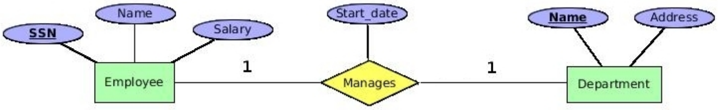
- How can we translate the following kind of 1:1 relationship type?



- If both sides are partial, we may **create a (new) relation schema** which cross-references the PKs of the relation schemas of the two entity types.

Step 3: Binary 1:1 Relationship Types - (Cross-reference)

- How can we translate the following kind of 1:1 relationship type?



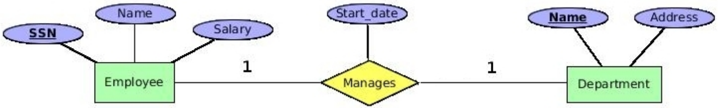
- If both sides are partial, we may **create a (new) relation schema** which cross-references the PKs of the relation schemas of the two entity types.
- MANAGES(SSN, Dname, Start_date) with

  PK: {SSN} or {Dname}

  FKs: [SSN]$\subseteq$EMPLOYEE[SSN] and [Dname]$\subseteq$DEPARTMENT[Name]

Step 3: Binary 1:1 Relationship Types - (Cross-reference)

- How can we translate the following kind of 1:1 relationship type?



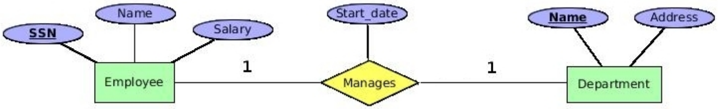
- If both sides are partial, we may **create a (new) relation schema** which cross-references the PKs of the relation schemas of the two entity types.
- MANAGES(SSN, Dname, Start_date) with

  PK: {SSN} or {Dname}

  FKs: [SSN]$\subseteq$EMPLOYEE[SSN] and [Dname]$\subseteq$DEPARTMENT[Name]
- Can we still merge them into a single relation using previous approaches?

Step 3: Binary 1:1 Relationship Types - (Cross-reference)

- How can we translate the following kind of 1:1 relationship type?



- If both sides are partial, we may **create a (new) relation schema** which cross-references the PKs of the relation schemas of the two entity types.
- MANAGES(SSN, Dname, Start_date) with

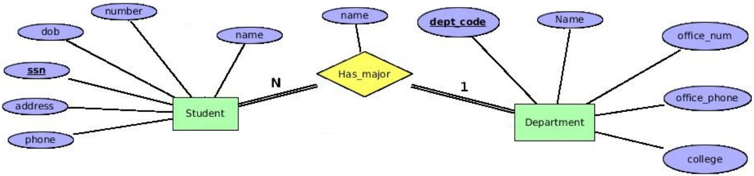
  PK: {SSN} or {Dname}

  FKs: [SSN]⊆EMPLOYEE[SSN] and [Dname]⊆DEPARTMENT[Name]
- Can we still merge them into a single relation using previous approaches?

  We cannot; otherwise what would be the primary key for the merged relation schema?

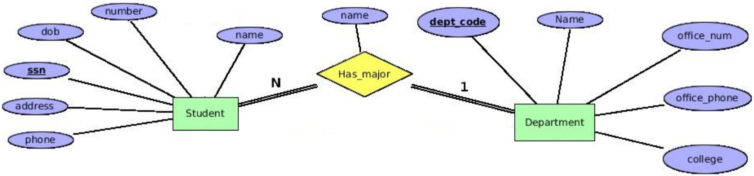
Step 4: Binary 1:N Relationship Types

- For each 1:N relationship type *R*, **extend the relation schema of the N-side entity type** by the attributes of *R* and the PK of the 1-side entity type, where
  - **PK:** still the PK of the N-side entity type
  - **FK:** references the PK of the 1-side entity type



Step 4: Binary 1:N Relationship Types

- For each 1:N relationship type *R*, **extend the relation schema of the N-side entity type** by the attributes of *R* and the PK of the 1-side entity type, where
  - **PK:** still the PK of the N-side entity type
  - **FK:** references the PK of the 1-side entity type



- STUDENT(SSN, Name, Number, DoB, address, phone, major_dept, major_name) with

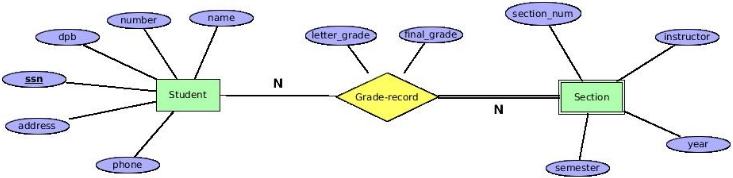
  PK: {SSN}

  FK: [major_dept]$\subseteq$DEPARTMENT[dept_code]

Australian National University

Step 5: Binary M:N (N:N) Relationship Types

- For each M:N (N:N) relationship type *R*, **create a relation schema** with the attributes of *R* plus the PKs of the participating entity types, where
  - **PK:** the combination of the PKs of the participating entity types
  - **FKs:** references the PKs of the participating entity types



- GRADE_RECORD(ssn, section_num, course_num, letter_grade, final_grade)

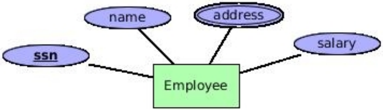
  PK: {ssn, section_num, course_num}

  FK: [ssn] ⊆ STUDENT[ssn]

  FK: [section_num, course_num] ⊆ SECTION[section_num, course_num].

Step 6: Multi-valued Attributes

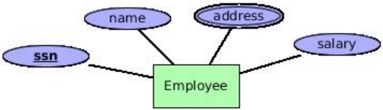
- For each multi-valued attribute *A*, **create a relation schema** with an attribute corresponding to A plus the PK of the entity/relationship type that has *A* as an attribute, where
    - **PK:** the combination of *A* and the PK of the entity/relationship type that has *A*
    - **FK:** references the PK of the entity/relationship type that has *A*



Australian National University

Step 6: Multi-valued Attributes

- For each multi-valued attribute *A*, **create a relation schema** with an attribute corresponding to A plus the PK of the entity/relationship type that has *A* as an attribute, where
  - **PK:** the combination of *A* and the PK of the entity/relationship type that has *A*
  - **FK:** references the PK of the entity/relationship type that has *A*



- EMPLOYEE_ADDRESS(SSN, Address) with

  PK: {SSN, Address}

  FK: [SSN]$\subseteq$EMPLOYEE[SSN]

ER-to-Relations Algorithm (Recall)

- The algorithm to first convert the basic ER model into relations, and then convert superclass/subclass from the EER model into relations.

  Step 1: Mapping of Regular Entity Types

  Step 2: Mapping of Weak Entity Types

  Step 3: Mapping of Binary 1:1 Relationship Types

  - Foreign key approach
  - Merged relation approach
  - Cross-reference approach

  Step 4: Mapping of Binary 1:N Relationship Types

  Step 5: Mapping of Binary M:N Relationship Types

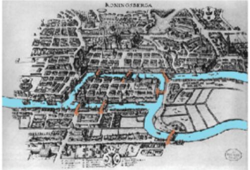
  Step 6: Mapping of Multi-valued Attributes

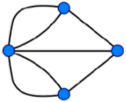
  Step 7: Mapping of N-ary Relationship Types

  Step 8: Mapping of Superclass/Subclass

(Credit Cookie) Graph Model and ER Diagram

Carl Gottlieb Ehler
(1685-1753)



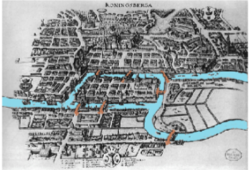


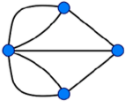
Seven Bridges of Königsberg

Euler
(1707-1783)

(Credit Cookie) Graph Model and ER Diagram

Carl Gottlieb Ehler
(1685-1753)





Seven Bridges of Königsberg

Euler
(1707-1783)

- 1st paper in ACM Transactions on Database Systems in 1976
- 1st international conference on very large data bases (VLDB) in 1975

The Entity-Relationship Model—Toward a Unified View of Data

PETER PIN-SHAN CHEN
Massachusetts Institute of Technology