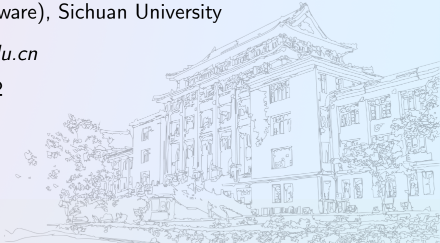
四川大學
SICHUAN UNIVERSITY

Database System Concepts

Database Design Using the E-R Model

伍元凯
College of Computer Science (Software), Sichuan University
wuyk0@scu.edu.cn

2023/04/12



1. **Requirements Analysis:** Gather information about the system's requirements.
2. **Conceptual Design:** Create a high-level conceptual model of the system.
3. **Logical Design:** Refine the conceptual model into a logical model that can be implemented in a database management system.
4. **Physical Design:** Determine the physical storage and access structures for the database.
5. **Implementation:** Implement the database design in a DBMS.
6. **Testing:** Test the database to ensure that it meets the system's requirements.
7. **Maintenance:** Maintain the database to ensure its ongoing reliability and performance.

目录

book_id	title	author
1	The Catcher in the Rye	J.D. Salinger
2	To Kill a Mockingbird	Harper Lee
3	The Great Gatsby	F. Scott Fitzgerald
4	1984	George Orwell

author_id	name	book
1	J.D. Salinger	The Catcher in the Rye
2	J.D. Salinger	To Kill a Mockingbird
3	F. Scott Fitzgerald	The Great Gatsby
4	George Orwell	1984

The book column in the Authors table duplicates information already stored in the Books table. This redundancy can lead to issues such as data inconsistencies and wasted storage. It is important to avoid redundancy in database design to ensure data integrity and efficiency.

customer_id	name	phone
1	John Smith	555-1234
2	Jane Doe	555-5678

order_id	date	customer_id	amount
1	2022-03-15	1	50.00
2	2022-03-18	2	100.00
3	2022-03-20		75.00

Notice that the third row of the Orders table has a missing value in the customer_id column. This incompleteness can lead to issues such as data inconsistencies and incorrect results. It is important to ensure that all required fields are present in a database design to ensure data integrity and accuracy.

目录

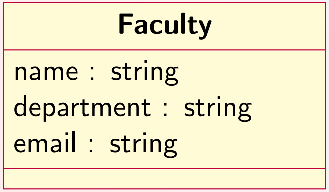
Consider a university that needs to manage its student records, course offerings, and faculty information.
An **entity** is an object or concept in the real world that is represented in the database. It can be a **physical object**, such as a person or a car, or **an abstract concept**, such as an order or a transaction.

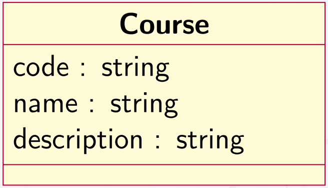
- **Student**: A person enrolled in the university who is taking one or more courses.
- **Course**: An offering by the university that teaches a particular subject.
- **Faculty**: A member of the university staff who teaches one or more courses.

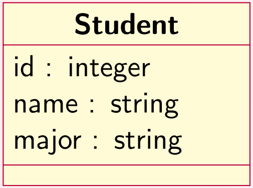
Consider the Student entity mentioned earlier. Each individual student in the university system is an entity, but **the collection of all students in the system is an entity set**.

In the context of the E-R model, the extension of an entity set refers to **the set of all individual entities that belong to that set**.
The Student entity set mentioned earlier. The extension of this entity set would be the set of all individual students in the university system.
Similarly, the extension of the Course entity set would be the set of all individual courses offered by the university, and the extension of the Faculty entity set would be the set of all individual faculty members employed by the university.

Attributes are the properties that describe the characteristics of an entity. Each attribute has a name and a data type, such as string, integer, or date. Attributes can be simple, such as a person's name or age, or composite, such as a person's address, which is made up of multiple sub-attributes.
Values, on the other hand, are the actual data that is stored in the attributes of an entity. For example, the value of the 'name' attribute for a particular 'Student' entity might be "John Smith", while the value of the 'age' attribute might be "22". The values of attributes can be unique to each entity, or they can be shared by multiple entities within an entity set.


Faculty
name : string
department : string
email : string


Course
code : string
name : string
description : string


Student
id : integer
name : string
major : string

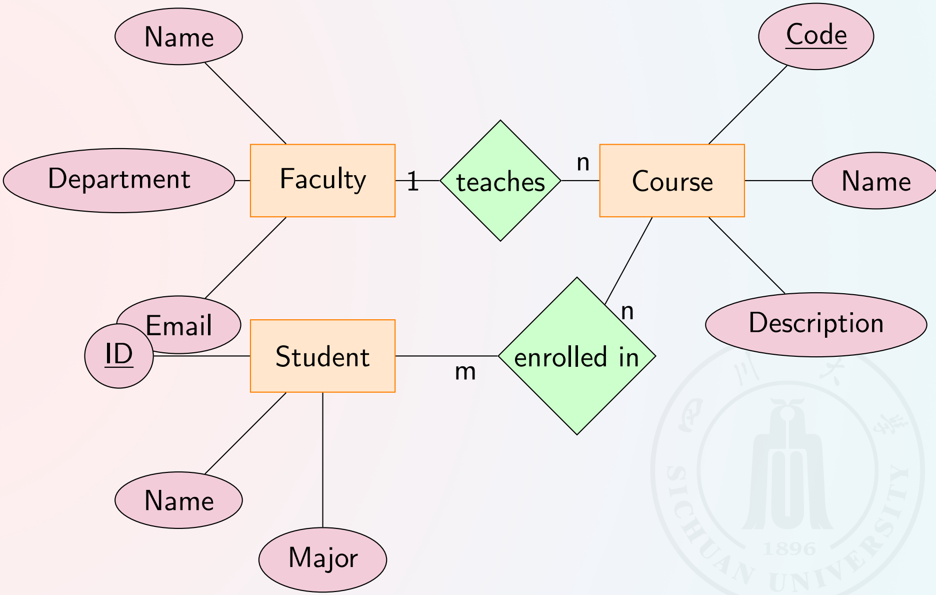
目录

A relationship in ER modeling represents an association between two or more entities. In this example, we can identify two relationships:

- A Faculty member teaches one or more Courses.
- A Course is enrolled in by one or more Students.

A relationship set is a set of relationships of the same type. In this example, we can define two relationship sets:

- The set of all teaches relationships between Faculty and Course.
- The set of all enrolled in relationships between Course and Student.


Name
Code
Department
Faculty
1
teaches
n
Course
Name
Email
ID
Student
m
enrolled in
n
Description
Name
Major

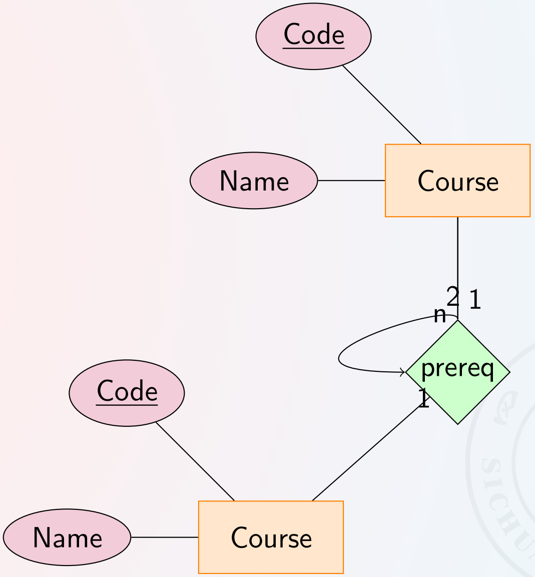
Formally, a relationship set is a mathematical relation on $n \geq 2$ (possibly nondistinct) entity sets. If E_1, E_2, …, E_n are entity sets, then a relationship set R is a subset of

$$\{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

where $(e_1, e_2, \ldots, e_n)$ is a relationship instance.
The association between entity sets is referred to as participation; i.e., the entity sets $E_1, E_2, \ldots, E_n$ participate in relationship set R.

A role is a way of specifying the part that an entity plays in a relationship. A role specifies how an entity participates in a relationship set.
Let's say that we want to capture the fact that a Faculty member can teach multiple courses, and a Course can have multiple Faculty members teaching it, but each Faculty member can only be assigned to one role per course, such as **"primary instructor" or "assistant instructor"**.
Using roles helps us to capture more information about the relationship between entity sets. It can help us to model more complex scenarios and relationships.

Formally, a recursive relationship set is a relationship set in which the same entity set participates more than once in different roles.
For example, consider the entity set course that records information about all the courses offered in the university. To depict the situation where one course (C_2) is a prerequisite for another course (C_1) we have relationship set prereq that is modeled by ordered pairs of course entities. The first course of a pair takes the role of course C_1, whereas the second takes the role of prerequisite course C_2. In this way, all relationships of prereq are characterized by (C_1, C_2) pairs; (C_2, C_1) pairs are excluded.

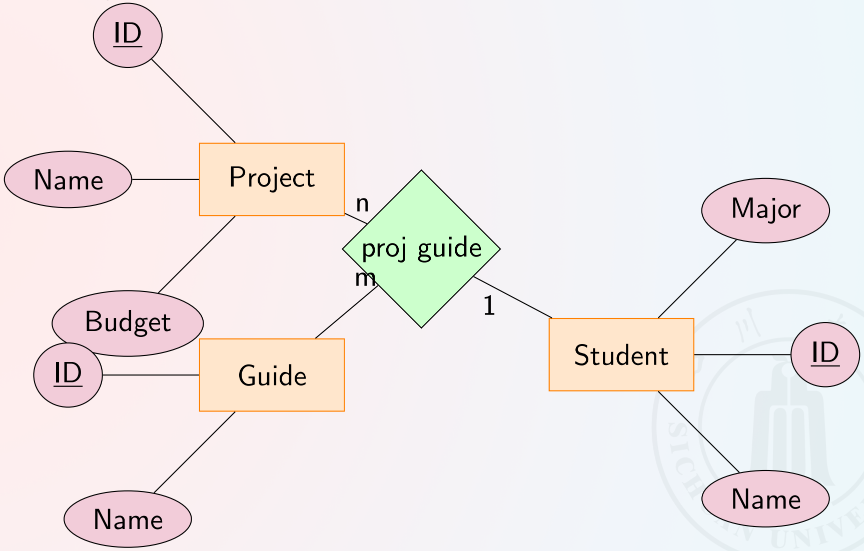

Code
Name
Course
2 1
n
prereq
1
Code
Name
Course

Descriptive attributes are attributes that provide additional information about an entity, beyond the primary key attributes. In the context of the course, student, and faculty entities, we can consider the following descriptive attributes:

- Course entity:
    - Credit hours: the number of credit hours that a course offers.
    - Department: the department that offers the course.
- Student entity:
    - GPA: the grade point average of the student.
    - Address: the mailing address of the student.
- Faculty entity:
    - Rank: the academic rank of the faculty member (e.g., assistant
    - professor, associate professor, full professor).
    - Office: the office number of the faculty member.

The degree of a relation set in a database schema refers to the number of distinct entity sets that participate in the relation.
For example, a relationship with a degree of 1 involves only one entity set, while a concept with a degree of 2 involves two different entity sets. A concept with a degree of 3 involves three entity sets, and so on.

A ternary relationship example:



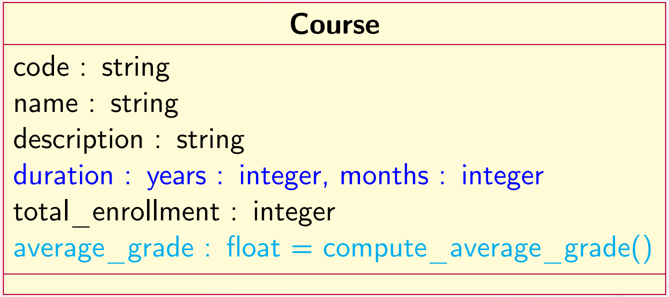
Attribute Types in the E-R Model

An attribute, as used in the E-R model, can be characterized by the following attribute types:

- Simple attribute: an attribute that cannot be further subdivided.
- Composite attribute: an attribute that can be subdivided into smaller parts.
- Derived attribute: an attribute whose value is derived from other attributes or data in the database.
- Key attribute: an attribute that uniquely identifies an entity within an entity set.

Each entity in an entity set has a value for each of its attributes. These values are stored as part of the entity's record in the database.

Complex attributes



Course

code : string
name : string
description : string
duration : years : integer, months : integer
total_enrollment : integer
average_grade : float = compute_average_grade()

- duration is a composite attribute consisting of two sub-attributes (years and months)
- average_grade is a derived attribute with a default value set by the method compute_average_grade()

Null Values in the E-R Model

For example, consider an `Employee` entity with attributes `EmployeeID`, `FirstName`, `LastName`, and `HireDate`. If an employee's hire date is unknown or has not yet been set, the `HireDate` attribute could be set to a null value.

Null values can also be used in relationships. For example, consider a `Project` entity and an `Employee` entity with a `WorksOn` relationship between them. If an employee is not currently working on any project, the relationship instance could have null values for both the `EmployeeID` and `ProjectID` attributes.

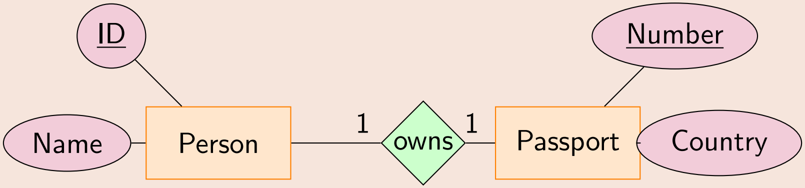
Mapping Cardinality for Binary Relationship Set

For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

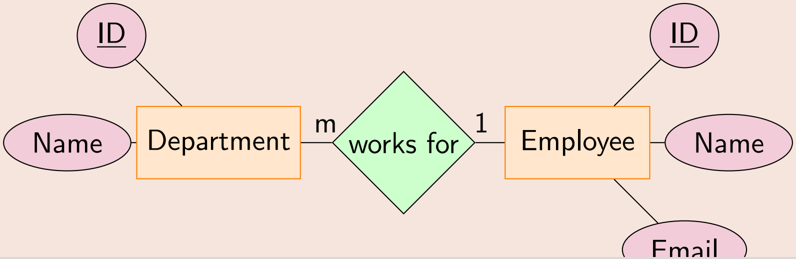
- **One-to-One (1:1)**: Each entity in set A is associated with at most one entity in set B, and vice versa.
- **One-to-Many (1:N)**: Each entity in set A is associated with any number of entities (possibly zero) in set B, but each entity in set B is associated with at most one entity in set A.
- **Many-to-One (N:1)**: Each entity in set A is associated with at most one entity in set B, but each entity in set B is associated with any number of entities (possibly zero) in set A.
- **Many-to-Many (N:M)**: Each entity in set A is associated with any number of entities (possibly zero) in set B, and vice versa.

One-to-one relationship

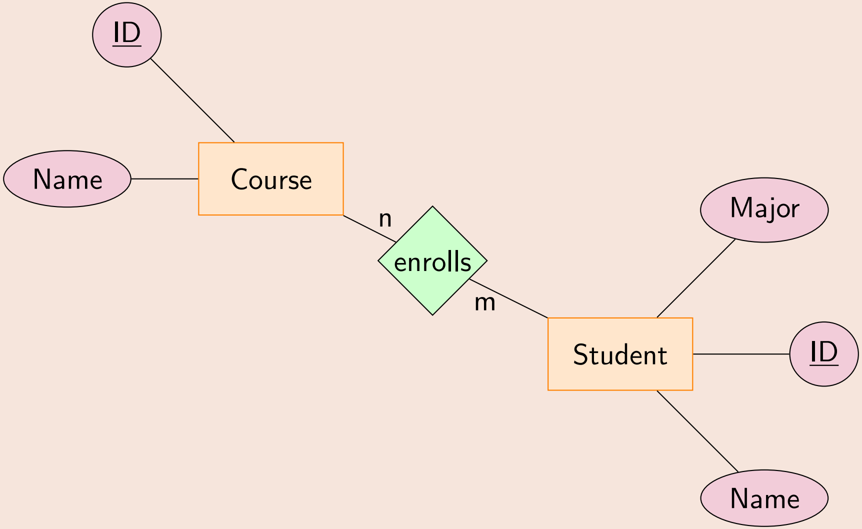
:



One-to-many relationship:



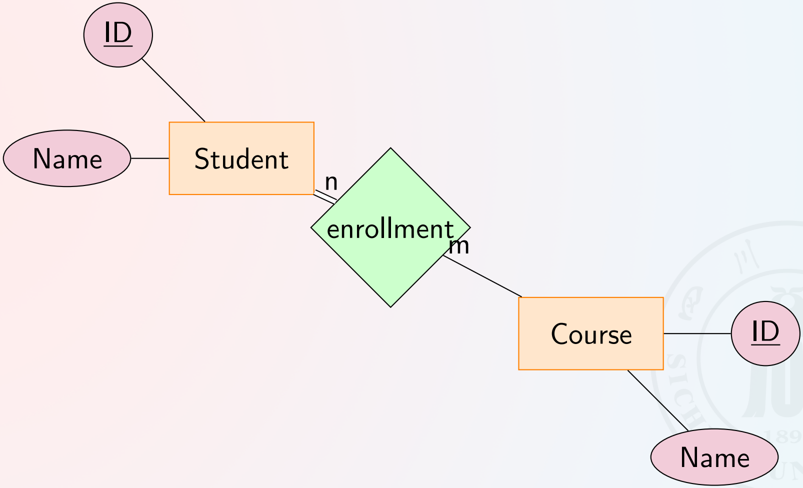
Many-to-many relationship:



Total and Partial Participation in an ER Model

The participation of an entity set E in a relationship set R can be either total or partial. Total participation means that every entity in E must participate in at least one relationship in R. Partial participation means that some entities in E may not participate in any relationships in R.

In an ER diagram, total participation is indicated by placing a double line between the entity set and the relationship set, while partial participation is indicated by a single line:



目录

Entity Sets and Primary Keys

An **entity set** is a collection of entities that have the same attributes. Each entity in an entity set has a set of attribute values that are distinct from the attribute values of all other entities in the set.
To uniquely identify entities in an entity set, we need a **primary key**, which is a set of one or more attributes that collectively identify each entity.

Books: entity set. Book: entity. Primary key: ISBN

目录

Let R be a relationship set involving entity sets $E_1, E_2, \dots, E_n$. Let primary-key(E_i) denote the set of attributes that forms the primary key for entity set E_i. Assume for now that the attribute names of all primary keys are unique. The composition of the primary key for a relationship set depends on the set of attributes associated with the relationship set R.

If the relationship set R has no attributes associated with it, then the set of attributes

$$\text{primary-key}(E_1) \cup \text{primary-key}(E_2) \cup \cdots \cup \text{primary-key}(E_n)$$

describes an individual relationship in set R.

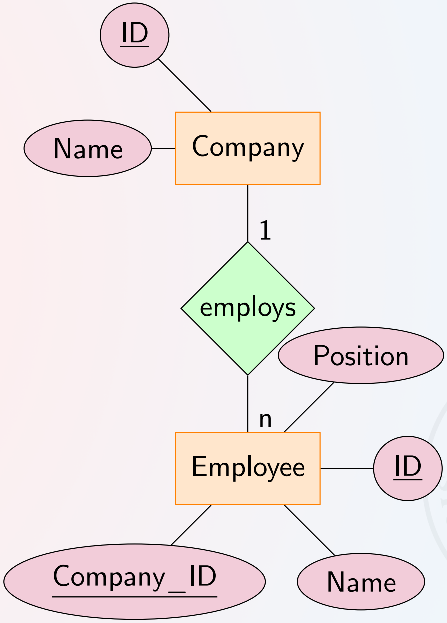
If the relationship set R has attributes $a_1, a_2, \dots, a_m$ associated with it, then the set of attributes

$$\text{primary-key}(E_1) \cup \text{primary-key}(E_2) \cup \cdots \cup \text{primary-key}(E_n) \cup \{a_1, a_2, \dots, a_m\}$$

describes an individual relationship in set R.

The choice of the primary key for a binary relationship set depends on the mapping cardinality of the relationship set.

- **One-to-One Relationship:** In a one-to-one relationship set, either entity can serve as the primary key. The choice of primary key is arbitrary.
- **One-to-Many Relationship:** In a one-to-many relationship set, the primary key of the entity on the one-side becomes a foreign key in the entity on the many-side. The primary key of the entity on the many-side is the combination of its own attributes and the foreign key.
- **Many-to-Many Relationship:** In a many-to-many relationship set, the primary key of the relationship set is the combination of the primary keys of the entities involved in the relationship set.


ID
Name
Company
1
employs
Position
n
Employee
ID
Company_ID
Name

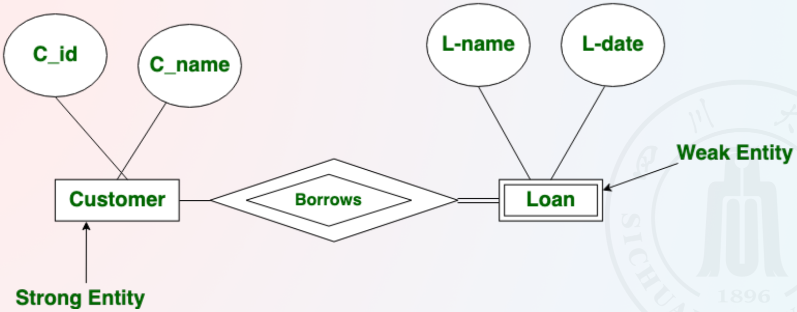
Consider the project guide discussed earlier with entity sets Project, Guide, and Student. A possible superkey for the relationship involving these entity sets would be:

Project_ID, Guide_ID

This superkey includes the primary keys of all two entity sets involved in the relationship.
Chapter 7

目录

A weak entity set is one whose existence is dependent on another entity set, called its identifying entity set; instead of associating a primary key with a weak entity, we use the primary key of the identifying entity, along with extra attributes, called discriminator attributes to uniquely identify a weak entity. An entity set that is not a weak entity set is termed a strong entity set.



Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be existence dependent on the identifying entity set. The identifying entity set is said to own the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the identifying relationship.
A weak entity set can participate in relationships **other than the identifying relationship**.

One way to remove redundancy is **to eliminate one of the redundant attributes, while keeping the other. Another approach is to combine the redundant attributes into a single attribute.** However, it is important to be careful when removing or combining attributes, as doing so can impact the ability to accurately and efficiently query the data.
In some cases, removing a redundant attribute may not be possible without **compromising data integrity**. In these situations, it may be necessary to keep the redundant attribute in order to maintain data accuracy and consistency.

- classroom: with attributes (<u>building</u>, <u>room number</u>, capacity, building name).
- department: with attributes (<u>dept name</u>, building, <u>budget</u>, department name).
- course: with attributes (<u>course id</u>, title, credits, course name).
- instructor: with attributes (<u>ID</u>, name, salary, department name).
- section: with attributes (<u>course id</u>, <u>sec id</u>, <u>semester</u>, <u>year</u>, course name, section name).
- student: with attributes (<u>ID</u>, name, tot cred, department name).
- time slot: with attributes (<u>time slot id</u>, (day, start time, end time), time slot name).

The attributes **"building name", "department name", "course name", "section name", and "time slot name"** are redundant because they can be derived from the primary key attributes.

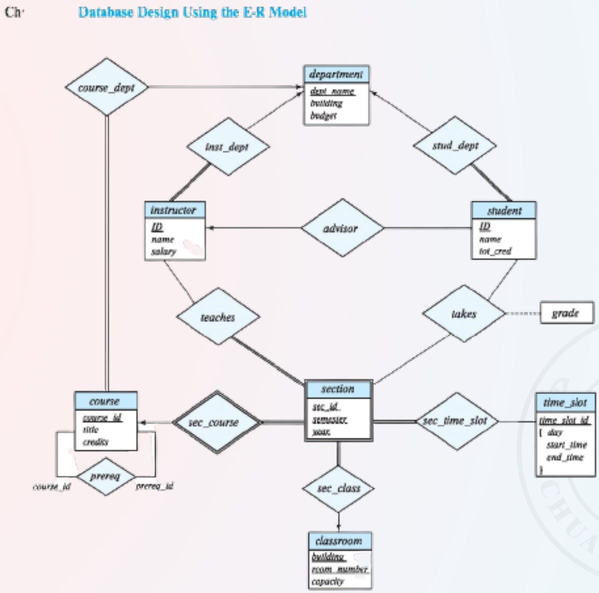
no redundant attribute

- classroom: with attributes (<u>building</u>, <u>room number</u>, capacity).
- department: with attributes (<u>dept name</u>, building, budget).
- course: with attributes (<u>course id</u>, title, credits).
- instructor: with attributes (<u>ID</u>, name, salary).
- section: with attributes (<u>course id</u>, <u>sec id</u>, <u>semester</u>, <u>year</u>).
- student: with attributes (<u>ID</u>, name, tot cred).
- time slot: with attributes (<u>time slot id</u>, (day, start time, end time)).

relationship sets

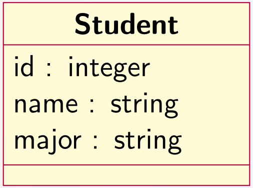
- inst_dept: relating instructors with departments.
- stud_dept: relating students with departments.
- teaches: relating instructors with sections.
- takes: relating students with sections, with a descriptive attribute grade.
- course_dept: relating courses with departments.
- sec_course: relating sections with courses.
- sec_class: relating sections with classrooms.
- sec_time_slot: relating sections with time slots.
- advisor: relating students with instructors.
- prereq: relating courses with prerequisite courses.

Removing Redundant Attributes in Entity Sets



目录

The schema representation of a strong entity set includes the entity set name, its attributes, and its primary key. The primary key is typically underlined to indicate its uniqueness and importance in identifying each entity instance in the entity set.



```sql
CREATE TABLE students (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    major VARCHAR(255)
);
```

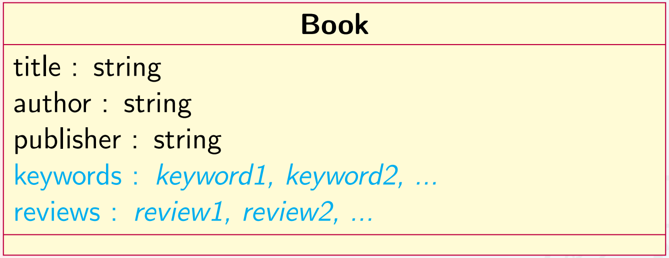
When a strong entity set has nonsimple attributes, things are a bit more complex. We handle composite attributes by creating a separate attribute for each of the component attributes; we do not create a separate attribute for the composite attribute itself.

Course
code : string
name : string
description : string
duration : years : integer, months : integer
total_enrollment : integer
average_grade : float = compute_average_grade()

```
CREATE TABLE Course (
code VARCHAR(255),
name VARCHAR(255),
description VARCHAR(255),
duration_years INT,
duration_months INT,
total_enrollment INT,
PRIMARY KEY (code)
);
```

Note that the complex attribute ''duration'' has been represented as two separate attributes: ''duration_years'' and ''duration_months''. Also, the attribute ''average_grade'' has been defined with a default value of ''compute_average_grade()'', assuming that there is a function in the system to compute the average grade of the course.

Multivalued attributes:


Book

title : string

author : string

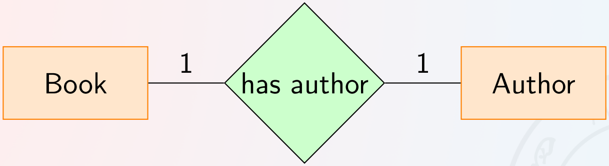
publisher : string

keywords : *keyword1*, *keyword2*, ...

reviews : *review1*, *review2*, ...

```
CREATE TABLE Books (
title VARCHAR(255) PRIMARY KEY,
author VARCHAR(255),
publisher VARCHAR(255)
);

CREATE TABLE BookKeywords (
title VARCHAR(255),
keyword VARCHAR(255),
FOREIGN KEY (title) REFERENCES Books(title)
);

CREATE TABLE BookReviews (
title VARCHAR(255),
review VARCHAR(255),
FOREIGN KEY (title) REFERENCES Books(title)
);
```

For a multivalued attribute M, we create a relation schema R with an attribute A that corresponds to M and attributes corresponding to the primary key of the entity set or relationship set of which M is an attribute. The primary key of R is the combination of all these attributes. For example, consider the following E-R diagram:



E-R diagram for a book and author database.

Suppose the multivalued attribute M is the set of genres for each book. We can create a new relation schema R with attributes "Book_ISBN" and "Genre". The primary key of R is the combination of "Book_ISBN" and "Genre".

```sql
CREATE TABLE Book_Genres (
Book_ISBN VARCHAR(20),
Genre VARCHAR(20),
PRIMARY KEY (Book_ISBN, Genre),
FOREIGN KEY (Book_ISBN) REFERENCES Book (ISBN)
);
```

In the case that an entity set consists of only two attributes—a single primary-key attribute B and a single multivalued attribute M —the relation schema for the entity set would contain only one attribute, namely, the primary-key attribute B.

time_slot

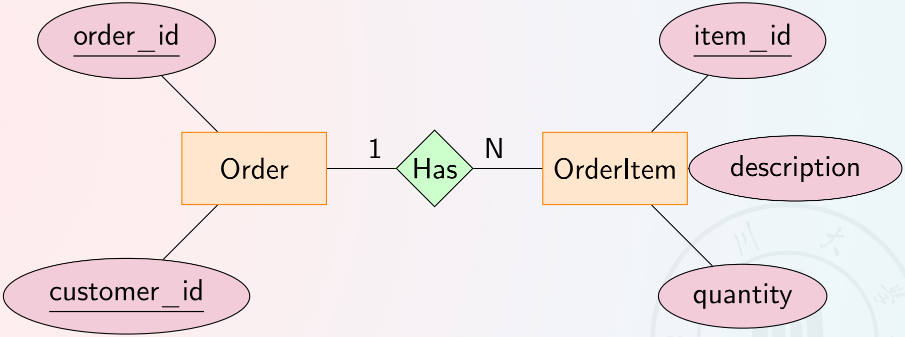
time_slot_id : string
time : *day: string, start_time: string, end_time: string*

```sql
CREATE TABLE time_slot (
    time_slot_id VARCHAR(255) PRIMARY KEY,
    day VARCHAR(255),
    start_time VARCHAR(255),
    end_time VARCHAR(255)
);
```

目录

Let A be a weak entity set with attributes $a_1, a_2, \dots, a_m$. Let B be the strong entity set on which A depends. Let the primary key of B consist of attributes $b_1, b_2, \dots, b_n$. We represent the entity set A by a relation schema called A with one attribute for each member of the set:

$$\{a_1, a_2, \dots, a_m\} \cup \{b_1, b_2, \dots, b_n\}$$


order_id
customer_id
Order
1
Has
N
OrderItem
item_id
description
quantity

```
CREATE TABLE OrderItem (
item_id INT,
description VARCHAR(255),
quantity INT,
order_id INT,
customer_id INT,
PRIMARY KEY (item_id, order_id, customer_id),
FOREIGN KEY (order_id, customer_id) REFERENCES Order(order_id,
    customer_id)
);
```

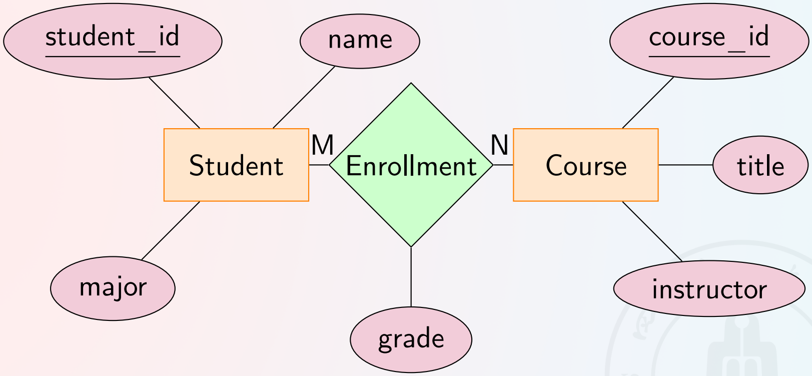
In this example, the entity set OrderItem is weak because it depends on the strong entity set Order, and its primary key includes attributes from both itself and the Order entity set.

目录

Let R be a relationship set, let $a_1, a_2, \dots, a_m$ be the set of attributes formed by the union of the primary keys of each of the entity sets participating in R, and let the descriptive attributes (if any) of R be $b_1, b_2, \dots, b_n$. We represent this relationship set by a relation schema called R with one attribute for each member of the set:

$$\{a_1, a_2, \dots, a_m\} \cup \{b_1, b_2, \dots, b_n\}$$

We also create foreign-key constraints on the relation schema R as follows: For each entity set E_i related by relationship set R, we create a foreign-key constraint from relation schema R, with the attributes of R that were derived from primary-key attributes of E_i referencing the primary key of the relation schema representing E_i.


student_id
name
course_id
Student
M
Enrollment
N
Course
title
major
grade
instructor

```
CREATE TABLE Student (
  student_id INTEGER PRIMARY KEY,
  name TEXT,
  major TEXT
);

CREATE TABLE Course (
  course_id INTEGER PRIMARY KEY,
  title TEXT,
  instructor TEXT
);

CREATE TABLE Enrollment (
  student_id INTEGER,
  course_id INTEGER,
  grade TEXT,
  PRIMARY KEY (student_id, course_id),

  FOREIGN KEY (student_id) REFERENCES Student(student_id),
  FOREIGN KEY (course_id) REFERENCES Course(course_id)
);
```

目录

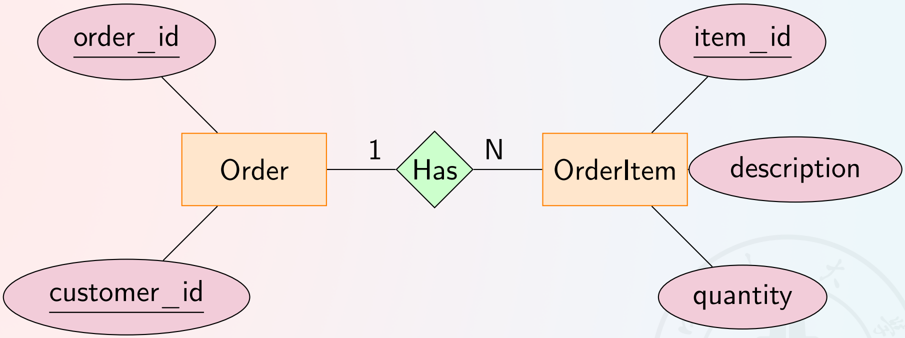
Redundancy in E-R Diagrams

- One common source of redundancy in E-R diagrams is through the use of multiple relationships between two entity sets.
- Another source of redundancy in E-R diagrams is through the use of redundant attributes. If an attribute can be derived from other attributes in the database, it should not be included as a separate attribute.
- The schema for the relationship set linking a weak entity set to its corresponding strong entity set is redundant and does not need to be present in a relational database design

目录

Redundancy in E-R Diagrams

Consider a many-to-one relationship set AB from entity set A to entity set B. Using our relational-schema construction algorithm outlined previously, we get three schemas: A, B, and AB.
Suppose further that the participation of A in the relationship is total; that is, every entity a in the entity set A must participate in the relationship AB. Then we can combine the schemas A and AB to form a single schema consisting of the union of attributes of both schemas. The primary key of the combined schema is the primary key of the entity set into whose schema the relationship set schema was merged.



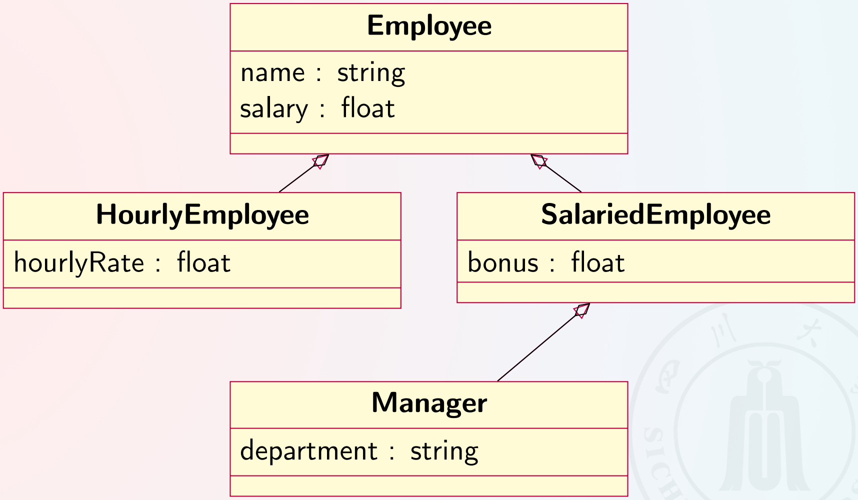
Suppose that every order must have at least one order item, so the participation of Order in the Has relationship is total.

```sql
CREATE TABLE OrderItem (
  item_id INTEGER PRIMARY KEY,
  order_id INTEGER,
  description TEXT,
  quantity INTEGER,
  -- other OrderItem attributes
);

CREATE TABLE OrderHas (
  order_id INTEGER,
  customer_id INTEGER,
  item_id INTEGER,
  description TEXT,
  quantity INTEGER,
  PRIMARY KEY (order_id, item_id),
  FOREIGN KEY (order_id) REFERENCES Order(order_id),
  FOREIGN KEY (item_id) REFERENCES OrderItem(item_id)
);
```

目录

The process of designating subgroupings within an entity set is called specialization.

- Disjoint specialization, on the other hand, requires an entity to belong to exactly one subtype. For example, in a company database, an employee entity may belong to either a full-time or a part-time subtype, but not both.
- Overlapping specialization allows an entity to belong to more than one subtype simultaneously. For example, a person entity may belong to both a student and a staff subtype in a university database.


Employee
name : string
salary : float
HourlyEmployee
hourlyRate : float
SalariedEmployee
bonus : float
Manager
department : string

Generalization | 目录

The refinement from an initial entity set into successive levels of entity subgroupings represents a top-down design process in which distinctions are made explicit. The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features.

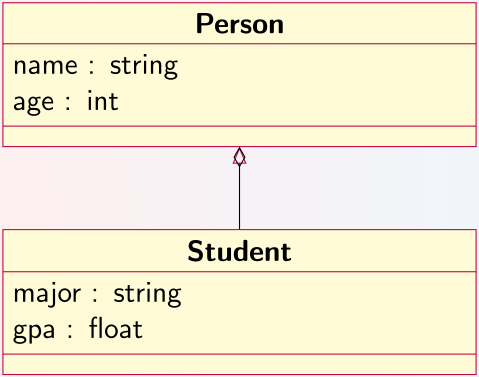
This commonality can be expressed by generalization, which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets.

Higher- and lower-level entity sets also may be designated by the terms superclass and subclass, respectively.

Inheritance

●○○ | 目录

A crucial property of the higher- and lower-level entities created by specialization and generalization is attribute inheritance. The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets.



The Student class inherits **the attributes name and age** from the Person class. The Student class also has **additional attributes major and gpa**.

Entity sets in this diagram have only single inheritance. If an entity set is a lower-level entity set in more than one inheritance relationship, then the entity set has multiple inheritance.

目录

A completeness constraint is a type of constraint on a specialization/generalization in addition to the disjointness or overlapping constraints. This constraint specifies whether an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/specialization.

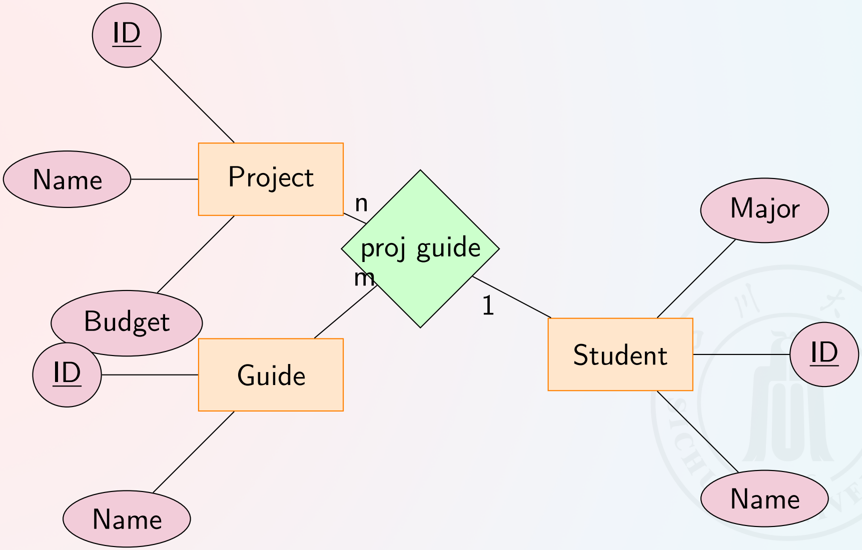
- A total specialization implies that every entity in the higher-level entity set must be a member of one of the lower-level entity sets.
- A partial specialization implies that some entities in the higher-level entity set may not belong to any of the lower-level entity sets.

```sql
-- Create the Vehicle table
CREATE TABLE Vehicle (
    VIN INT PRIMARY KEY,
    Make VARCHAR(50),
    Model VARCHAR(50),
    Year INT
);
-- Create the Car table, which is a specialization of Vehicle
CREATE TABLE Car (
    VIN INT PRIMARY KEY,
    NumDoors INT,
    FOREIGN KEY (VIN) REFERENCES Vehicle(VIN)
);
-- Create the Motorcycle table, which is another specialization of
    Vehicle
CREATE TABLE Motorcycle (
    VIN INT PRIMARY KEY,
    EngineSize INT,
    FOREIGN KEY (VIN) REFERENCES Vehicle(VIN)
);
-- Insert a new entity into the Vehicle entity set, which must also
     be inserted into either the Car or Motorcycle entity set
INSERT INTO Vehicle (VIN, Make, Model, Year) VALUES (1, 'Toyota', '
    Corolla', 2021);
INSERT INTO Car (VIN, NumDoors) VALUES (1, 4);
-- OR
INSERT INTO Motorcycle (VIN, EngineSize) VALUES (1, 1000);
```

When a total completeness constraint is in place, **an entity inserted into a higher-level entity set must also be inserted into at least one of the lower-level entity sets.**
When we insert a new entity into the Vehicle table, we must also insert it into either the Car or Motorcycle table, depending on the type of vehicle it is.

目录

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



Regard the relationship set proj guide (relating the entity sets guide, student, and project) as a higher-level entity set:

```
CREATE TABLE ProjectGuideStudent (
ProjectID int PRIMARY KEY,
GuideID int PRIMARY KEY,
StudentID int,
FOREIGN KEY (ProjectID) REFERENCES Project(ID),
FOREIGN KEY (GuideID) REFERENCES Guide(ID),
FOREIGN KEY (StudentID) REFERENCES Student(ID)
);
```

目录

There are two different methods of designing relation schemas for an E-R diagram that includes generalization:

1. Create a schema for the higher-level entity set. For each lower-level entity set, create a schema that includes an attribute for each of the attributes of that entity set plus one for each attribute of the primary key of the higher-level entity set.
2. if the generalization is disjoint and complete—that is, if no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher-level entity set is also a member of one of the lower-level entity sets. Here, we do not create a schema for the higher-level entity set.

```sql
CREATE TABLE Undergrad (
  PersonId int PRIMARY KEY,
  Name varchar(255),
  Major varchar(255)
);

CREATE TABLE Faculty (
  PersonId int PRIMARY KEY,
  Name varchar(255),
  Department varchar(255)
);
```

目录

The use of **the primary key of an entity set as an attribute of another entity set**, instead of using a relationship.

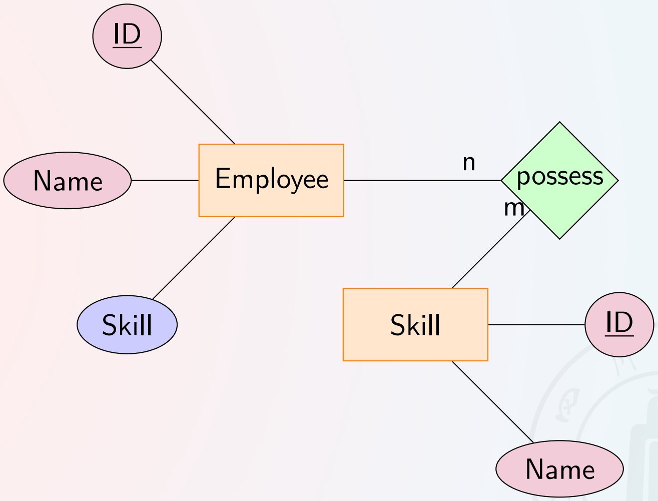
```sql
CREATE TABLE Student (
  student_id int PRIMARY KEY,
  name varchar(255),
  major varchar(255)
);

CREATE TABLE Course (
  course_id int PRIMARY KEY,
  title varchar(255),
  department varchar(255),
  student_id int
);
-- It implies that each course is taken by only one student, which
    is not true.
```

Designate the primary key attributes of the related entity sets as **attributes of the relation**.

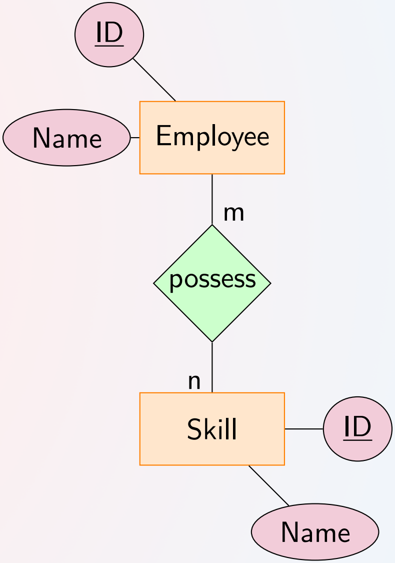
```
CREATE TABLE Employee (
  EmployeeID int PRIMARY KEY,
  Name varchar(255),
  Salary int,
  DepartmentID int
);

CREATE TABLE Department (
  DepartmentID int PRIMARY KEY,
  Name varchar(255)
);

CREATE TABLE WorksIn (
  EmployeeID int,
  DepartmentID int,
  PRIMARY KEY (EmployeeID, DepartmentID),
  FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),

  FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
```

The Employee table has a foreign key that references the Department table, indicating the department in which the employee works, without the need for an additional relationship table.

```
CREATE TABLE Employee (
  EmployeeID int PRIMARY KEY,
  Name varchar(255),
  Salary int,
  DepartmentID int REFERENCES Department(DepartmentID)
);

CREATE TABLE Department (
  DepartmentID int PRIMARY KEY,
  Name varchar(255)
);
```



The above diagram mistakenly uses a **single-valued attribute (employeeskill)** to represent the employee's skills.

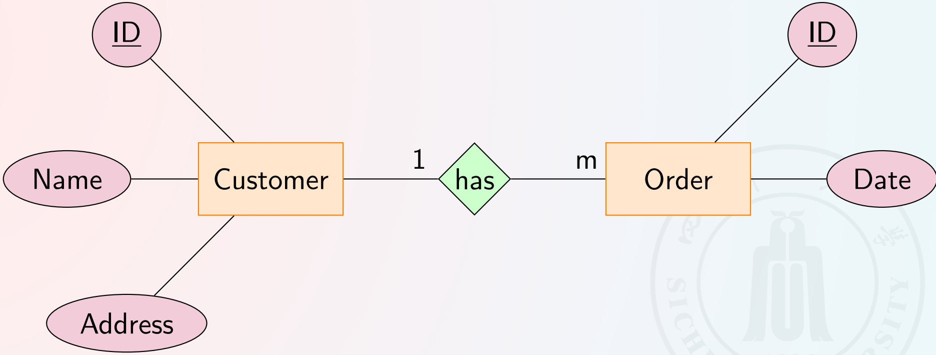


ID
Name
Employee
m
possess
n
Skill
ID
Name

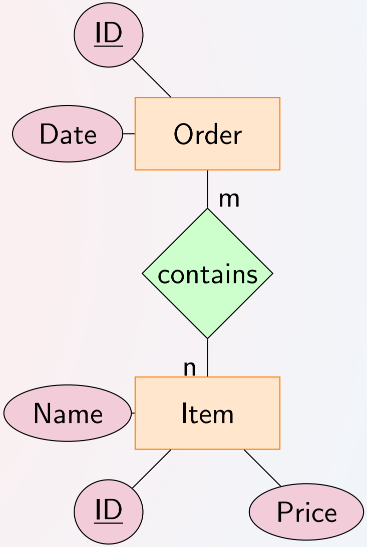
```
CREATE TABLE Employee (
  ID int PRIMARY KEY,
  Name varchar(255)
);

CREATE TABLE Skill (
  ID int PRIMARY KEY,
  Name varchar(255)
);

CREATE TABLE Possess (
  EmployeeID int,
  SkillID int,
  FOREIGN KEY (EmployeeID) REFERENCES Employee(ID),
  FOREIGN KEY (SkillID) REFERENCES Skill(ID),
  PRIMARY KEY (EmployeeID, SkillID)
);
```

This schema separates the multi-valued Skill attribute into its own table and creates a separate Possess table to represent the many-to-many relationship between Employee and Skill

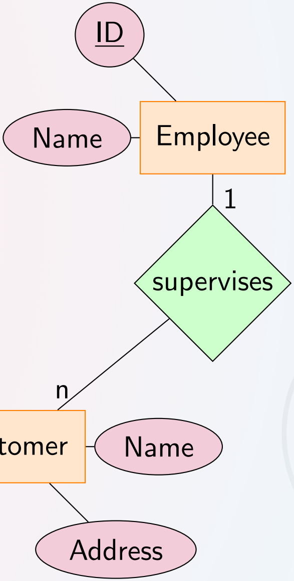
When an E-R diagram becomes too big to draw in a single piece, it makes sense **to break it up into pieces, each showing part of the E-R model.**



Part 2:

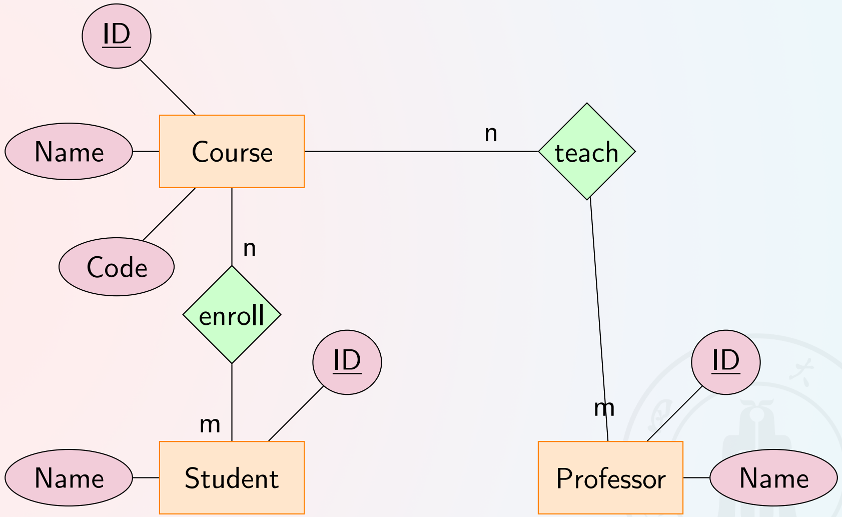


Part 3:



目录

Attributes represent properties or characteristics of an entity.
Entity sets are collections of similar entities. An entity set represents a set of objects that share the same attributes, or in other words, a set of objects that can be described using the same set of properties.
The decision about what constitutes an attribute and what constitutes an entity set is often based on the semantics of the data and the needs of the application. In general, it is better to err on the side of defining more entity sets and fewer attributes, as this can help to improve the accuracy and flexibility of the model.



if we later decide to add additional properties or relationships to courses or students, we can do so without disrupting the entire model

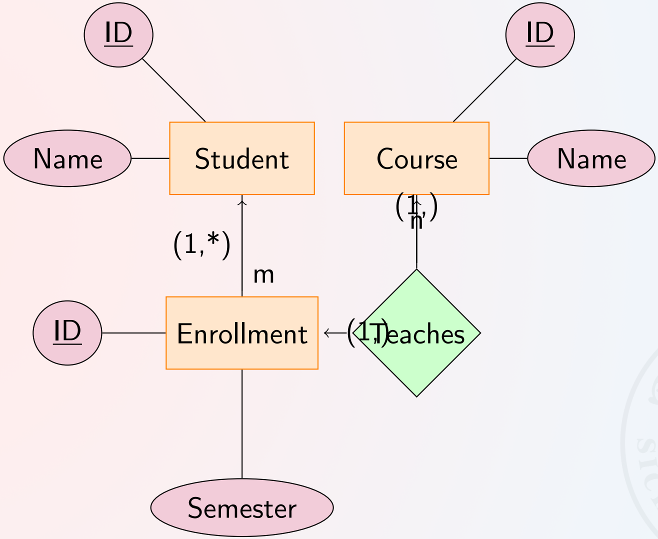
```sql
CREATE TABLE Course (
  ID INT PRIMARY KEY,
  Name VARCHAR(255),
  Code VARCHAR(10)
);

CREATE TABLE Student (
  ID INT PRIMARY KEY,
  Name VARCHAR(255)
);

CREATE TABLE Professor (
  ID INT PRIMARY KEY,
  Name VARCHAR(255)
);

CREATE TABLE Enrollment (
  CourseID INT,
  StudentID INT,
  PRIMARY KEY (CourseID, StudentID),
  FOREIGN KEY (CourseID) REFERENCES Course(ID),
  FOREIGN KEY (StudentID) REFERENCES Student(ID)
);

CREATE TABLE Teaching (
  CourseID INT,
  ProfessorID INT,
  PRIMARY KEY (CourseID, ProfessorID),
  FOREIGN KEY (CourseID) REFERENCES Course(ID),
  FOREIGN KEY (ProfessorID) REFERENCES Professor(ID)
);
```

目录

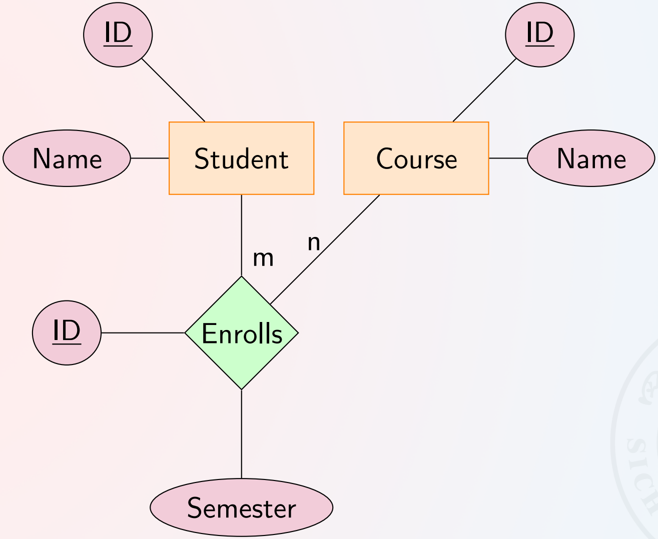
Use of Entity Sets versus Relationship Sets

An entity set represents a collection of objects or things in the real world that share common attributes or characteristics. On the other hand, **a relationship set** represents a connection or association between two or more entity sets.

In general, it is better to err on the side of **defining more entity sets and fewer relationship sets**, as this can help to improve the accuracy and flexibility of the model. However, there may be cases where a relationship set is more appropriate or necessary, and in those cases it should be used.



Incorrect one

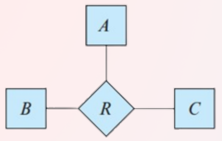


Correct one

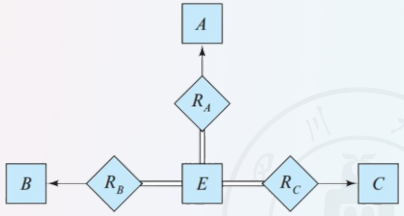
目录

We replace the relationship set 3-ary R with an entity set E, and we create three relationship sets:

- R_A, a many-to-one relationship set from E to A.
- R_B, a many-to-one relationship set from E to B.
- R_C, a many-to-one relationship set from E to C.



(a)



(b)

If the relationship set R had any attributes, these are assigned to entity set E; further, a special identifying attribute is created for E.

For each relationship (a_i, b_i, c_i) in the relationship set R, we create a new entity e_i in the entity set E. Then, in each of the three new relationship sets, we insert a relationship as follows:

- (e_i, a_i) in R_A.
- (e_i, b_i) in R_B.
- (e_i, c_i) in R_C.

Restriction:

- An identifying attribute may have to be created for the entity set created to represent the relationship set. **increases the complexity of the design and overall storage requirements.**
- An n-ary relationship set shows **more clearly** that several entities participate in a single relationship.
- There may not be a way to **translate constraints** on the ternary relationship into constraints on the binary relationships.

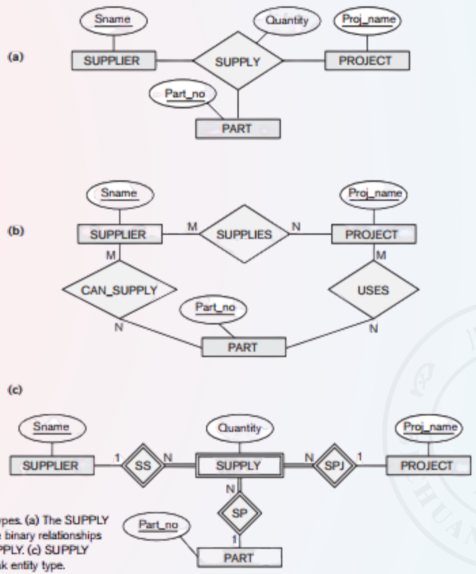
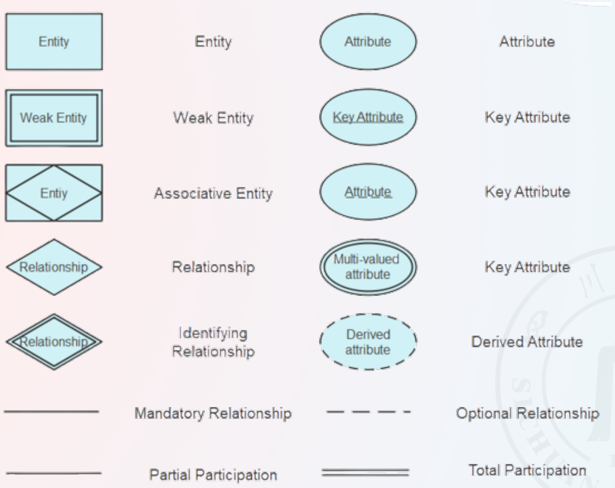
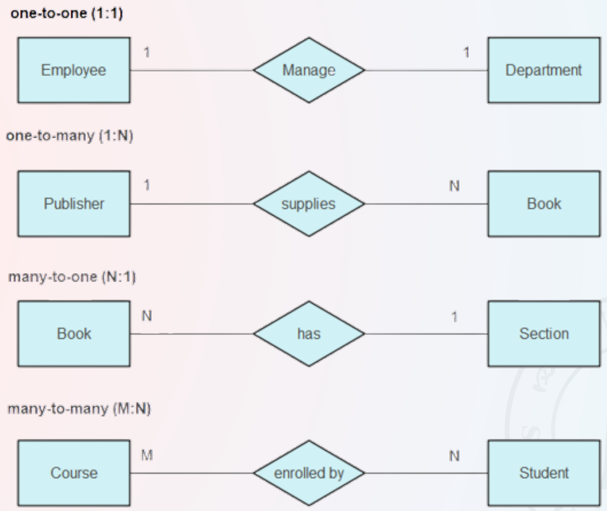


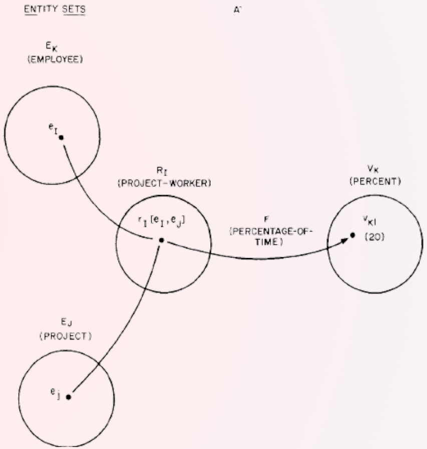
Figure 7.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

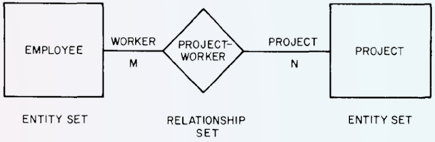
目录

Peter Chen (陳品山) was the one responsible for coming up with the Chen ERD notation.




one-to-one (1:1)
Employee
1
Manage
1
Department
one-to-many (1:N)
Publisher
1
supplies
N
Book
many-to-one (N:1)
Book
N
has
1
Section
many-to-many (M:N)
Course
M
enrolled by
N
Student

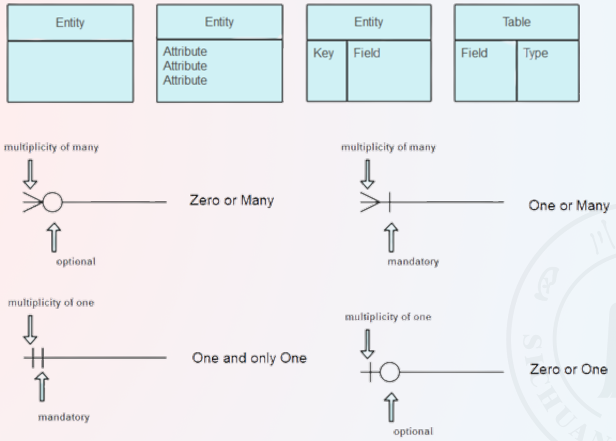




Chen, Peter Pin-Shan. "The entity-relationship model—toward a unified view of data." ACM transactions on database systems (TODS) 1.1 (1976): 9-36.

one of the 38 most influential papers in Computer Science

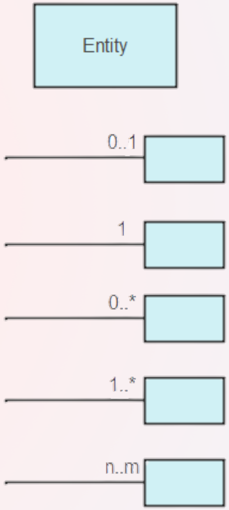
The U.S. **National Institute for Standards and Technology** defined a standard called IDEF1X in 1993. IDEF1X uses the **crow' s-foot notation**.



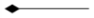
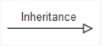
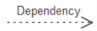
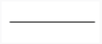
UML Notation ●○○○○○ 目录

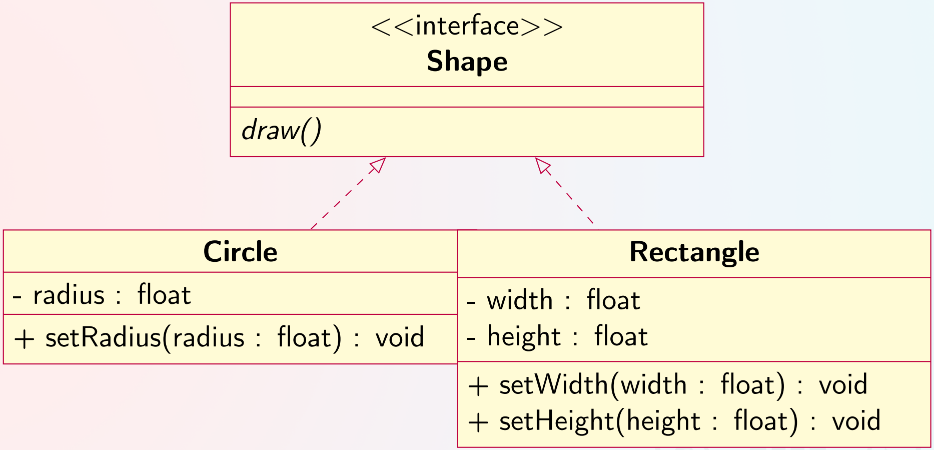
The Unified Modeling Language (UML) is a standard developed under the auspices of the Object Management Group (OMG) for creating specifications of various components of a software system. Some of the parts of UML are

- **Class diagram**. A class diagram is similar to an E-R diagram.
- **Use case diagram**. Use case diagrams show the interaction between users and the system, in particular the steps of tasks that users perform (such as withdrawing money or registering for a course).
- **Activity diagram**. Activity diagrams depict the flow of tasks between various components of a system.
- **Implementation diagram**. Implementation diagrams show the system components and their interconnections, both at the software component level and the hardware component level.

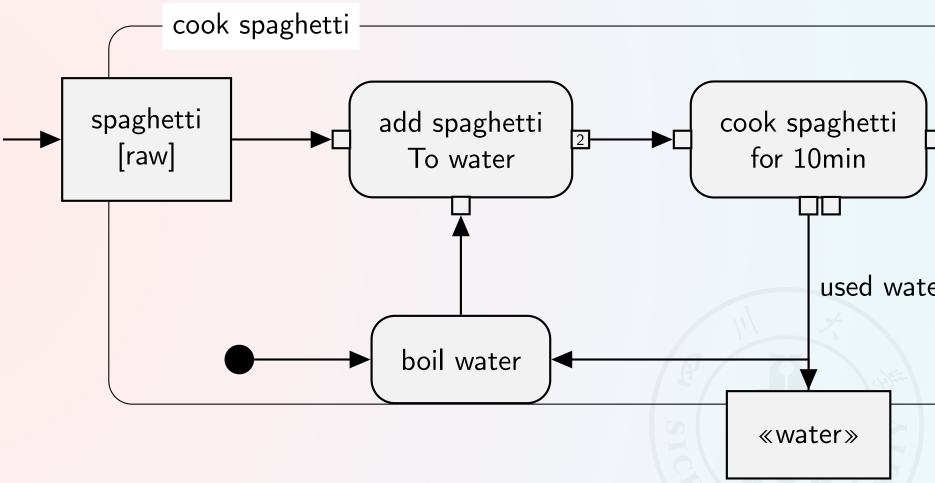


Entity	Entity
0..1	Zero or One
1	One and only One
0..*	Zero or More
1..*	One or More
n..m	The range is specified

Symbols	Description
	**Composition relationship** It was named the composition association relationship in UML 1.4, represents whole-part relationships and is a form of aggregation. A composition relationship specifies that the lifetime of the part classifier is dependent on the lifetime of the whole classifier.
	**Inheritance relationship** It helps people to communicate structure and inheritance of an object model. Inheritance relationships organize classes into generalization-specialization (superclass-subclass) hierarchies; they provide a basic re-use mechanism for sharing attributes and operations.
	**Dependency relationship** It is a relationship in which one element, the client, uses or depends on another element, the supplier.
	**Association relationship** It is a relationship between two classifiers, such as classes or use cases, that describes the reasons for the relationship and the rules that govern the relationship.


<<interface>>
Shape
draw()
Circle
- radius : float
+ setRadius(radius : float) : void
Rectangle
- width : float
- height : float
+ setWidth(width : float) : void
+ setHeight(height : float) : void

Class diagram of the Shape hierarchy


cook spaghetti
spaghetti [raw]
add spaghetti To water
2
cook spaghetti for 10min
used wate
boil water
«water»

目录

All enterprises have rules on what kinds of functionality are to be supported by an enterprise application.

- Transactions that update the data.
- Queries to view data in a desired fashion.
- An authorization mechanism is very important for any enterprise application

目录

The term **workflow** refers to the combination of data and tasks involved in processes like those of the preceding examples. Workflows interact with the database system as they move among users and users perform their tasks on the **workflow**.

目录

- A good database design anticipates future needs of an organization and ensures that the schema **requires minimal changes as the needs evolve**.
- It is important to distinguish between **fundamental constraints** that are expected to be permanent and constraints that are anticipated to change.
- All of the **people** involved with the data have needs and preferences that should be taken into account in order for a database design and deployment to succeed within the enterprise.

阅读"Chen, Peter Pin-Shan. "The entity-relationship model—toward a unified view of data." ACM transactions on database systems (TODS) 1.1 (1976): 9-36."，并写一篇读后感。

Thanks

End of Chapter 6