

Database Analysis and Design

Lesson 4: Entity Relationship Data Model

Ms. SEAK Leng

Plan

- Introduction to ER model
- Notion of Entity
- Notion of Attribute
- Notion of Relationship
- Degree of Relationship
- Cardinality of Relationship
- Participation Constraints
- Weak Entity
- Associative Entity
- ISA hierarchies

Learning Objectives

By the end of this lesson, students will be able to:

- Define and Describe Key Concepts of ER Models
- Identify and Differentiate Types of Attributes
- Explain Advanced ER Concepts: weak entities, associative entities, ISA hierarchies

History of ER Model

- The Entity-Relationship (ER) model was introduced by Peter Chen in 1976 in his seminal paper “The Entity-Relationship Model: Toward a Unified View of Data.”
- The model was developed to address the need for a more intuitive and conceptual approach to database design by allowing database designers to model data at a higher, more abstract level.

History of ER Model

- Before the ER model, databases were typically designed using hierarchical and network models, which were complex, rigid, and difficult to adapt to evolving data needs.
- The ER model revolutionized database design by offering a clear and standardized way to model data relationships, and it continues to be widely used today in both academic and industrial settings.

Why the ER Model?



1. Conceptual Clarity and Visual Representation

- The ER model is a high-level representation of the data and its relationships, making it easier to understand for non-technical stakeholders. It provides an intuitive way to map real-world objects and their associations into a diagram that is easy to read and interpret
- For example, entities like Student and Course are concepts that are easily understood by domain experts, and the relationships between them can be visually represented.

Why the ER Model?

2. Abstraction and Platform-Independence

- The ER model is abstract, meaning it is independent of the physical database technology or platform being used. This independence allows database designers to focus on the structure of the data without worrying about technical implementation details at the initial design stages.
- The ER model is platform-agnostic and can be mapped to various types of database systems, including MySQL, PostgreSQL, Microsoft SQL Server, Oracle.

Why the ER Model?

3. Simplifies Database Design

- The ER model simplifies the process of organizing and representing complex data relationships. It offers a structured approach to handle entities and their relationships into a clear framework.
- Using the ER model, one can define different types of relationships (one-to-one, one-to-many, many-to-many), which helps in visualizing how entities interact within the system.

Why the ER Model?

4. Foundation for Relational Databases

- The ER model has become the conceptual basis for relational databases. The mapping of entities, relationships, and attributes in the ER model directly translates to tables, columns, and foreign keys in relational database systems
- This close alignment with relational database theory makes it easy to convert an ER diagram into a database schema during the implementation phase.

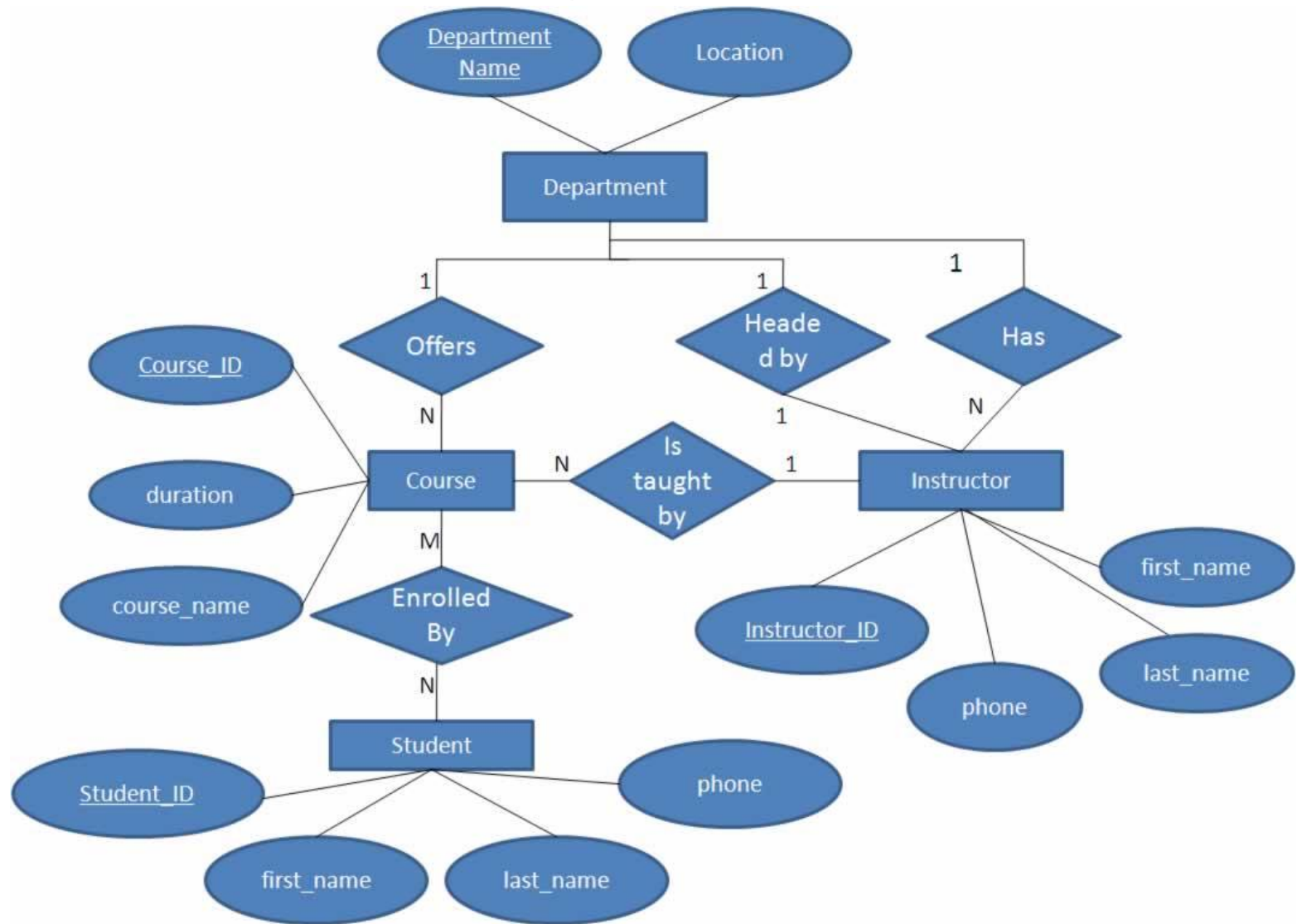
Introduction to ER Data Model

- **The entity-relationship (ER) data model** is a high-level conceptual data model used to describe the structure of a database.
- It helps visualize the relationships between entities in the database and how they interact with each other.
- ER data models serve as a blueprint for designing databases, and they are widely used during the design phase of database development.

Introduction to ER Data Model

- An ER data model **consists** of 3 main components: entities, attributes, and relationships.
- These elements are represented graphically, usually rectangles for entities, ovals for attributes, and diamonds for relationships.
- It shows entity sets and relationship set rather than individual record.

Example: ER Diagram



An ER diagram of a university database

Notion of Entity

- An entity is a real-world object or concept that is distinguishable from other objects.
- In the context of a database, an entity is anything that you want to store information about. Each entity has attributes that define its properties.
- A noun is used to represent an entity.
- An entity might be:
 - An object with physical existence (e.g., a lecturer, a student, a car).
 - An object with conceptual existence (e.g., a course, a job, a position).

Notion of Entity

- **Entity Type:** A collection of entities that share the same properties. It is usually represented as a rectangle in ER diagrams.
- **Entity Instance:** A single occurrence of an entity type.
- **Example:** In a university database, examples of entities are Student, Course, Department, etc:
 - Student: represents a student in the university
 - Course: represents a course offered by the university.

Exercise

Suppose that a bookshop sells different kind of reading book such as philosophy, social, mathematic, etc. The shop owner wants to keep record about detail information of all books in the shop, and he also wants to keep track of his customer. He wants to filters the address of customers, authors and editors of the books. What can be the entity type for this kind of system?

- Objective:
 - To get to know if student understand the notion of entity or not
 - To give students a chance to practice identifying individual entity of a given real domain and classifying them into an entity set. This is a process *in conceptual data modeling*.
- Working flow:
 - Brainstorming: get all possible entities
 - Filtering: pick up only needed entities which are right to the context

Exercise

Answer: Entity Set

1. BOOK: a book is an independent entity. It can be distinctly identified by ISBN. The book here is not the same as the copy of the book. The book may have many copy. All books share the same property, for instance it has ISBN, a title, author(s), publish year, etc., so the entity set BOOK contain all book in the bookshop.
2. CUSTOMER: each person can be distinctly identified, a customer is a person, so they can be identified too. The entity set CUSTOMER contain all customers of the store.
3. ADDRESS: an address can be considered as an entity because all address is a set of number, street, postal code, city, and country. ADDRESS is an entity set containing all address in the system.
4. AUTHOR: you may think that author is just information of a book, but author exist even there is not a book. Thus, author is an independent entity. AUTHOR is an entity set containing all authors of books in the shop.
5. EDITOR: the same as author editor is an independent entity. It can exist without the presence of a book.

Notion of Attributes

- Attributes are properties or characteristics of an entity.
- Each entity will have a set of attributes, and they describe the details or information associated with the entity.
- Attributes have value of any data types (e.g. numbers, character strings, dates, etc).
- All values in attribute have the same data type.
- Each attribute should have **only one value** for each instance of the entity.

Different kinds of attributes:

- **Key attribute or Primary key:** an attribute or a set of attributes which uniquely identifies an instance of an entity, ex: id
- **Simple attribute:** single component, cannot be divided further, ex: Gender, age
- **Composite attribute:** can be divided into smaller subparts, ex: Full name = First name + Last name
- **Derived attributes:** an attribute whose value can be calculated from other attributes, ex: age can be derived from birth_date.
- **Multi-value attribute:** an attribute that can have multiple values, ex: phone number of a person.

Notion of Attribute

- Exercise

Suppose that a bookshop sells different kind of reading book such as philosophy, social, mathematic, etc. The shop owner wants to keep record about detail information of all books in the shop, and he also wants to keep track of his customer. Find **attributes** of entity type you've found in the previous exercise?

- Work flow:

- All entity types are identified in the previous exercise, so the next work is to find all useful attributes for each entity type. There may be numerous data for each entity set, but **only those are needed** for the system are chosen.

Notion of Attribute

Answer:

- BOOK
 - ISBN: the international number of a book. It is the key of entity set BOOK because it is unique for each book in BOOK.
 - Title: the title of book.
 - PublishYear: the year in which the book is published.
 - Edition: the number of edition of the book.
 - Category: this is the choice of design. We may consider category as an attribute if a book has only one category. In contrast, if a book can have many category, we have to design category as entity set.
 - Description: the description of book.

Notion of Attribute

Answer:

- CUSTOMER
 - CustomerNo: the key of each CUSTOMER. This key is created by the system to simplify the identifying of each customer.
 - Name: the name of customer.
 - Tel: the number of telephone of customer. We consider only one number for each customer.
- ADDRESS
 - Number: the number of house or apartment. This attribute is a part of key of entity which is composed of 3 attributes Number, Street, and City.
 - Street
 - City
 - Postalcode.

Notion of Attribute

Answer:

- AUTHOR
 - AuthorNo: the key of AUTHOR.
 - Name: the name of author.
 - Address: the address of author. We consider only one address for each author. On the other hand, the address of customer can be multiple and important because the shop may need to deliver books to customers. Thus, the detail information about the address of customer allows user to find the most efficient deliver method.
- EDITOR
 - EditorNo: the key of EDITOR.
 - Name: the name of editor.
 - Address: the address of editor.

Notion of Relationship

- A relationship represents an association between two or more entities.
- Regularly represented using a **verb**.
 - Ex: John <work in> IT Department. He is related to the IT Department for being the member of the department. Thus, associations between entities are described by their relationship.
- Same entity set could participate in different relationship sets, or in different “roles” in same set.
 - Ex: EMPLOYEE <Work in> DEPARTMENT.
 - EMPLOYEE <Manage> DEPARTMENT.

Notion of Relationship

- Relationship can be read in both direction (passive or active). Ex:
 - EMPLOYEE <Work in> PROJECT. The relationship between entity sets EMPLOYEE and PROJECT is <Work in>.
 - PROJECT <Is done by> EMPLOYEE.
- Both of them are semantically equivalent. Thus, relationship is normally read in both direction (passive or active).

Notion of Relationship

- Exercises

Suppose that a bookshop sells different kind of reading book such as philosophy, social, mathematic, etc. The shop owner wants to keep record about detail information of all books in the shop. He also wants to keep track of his customers and the books they have bought. Find **relationship** between entity sets you've found in the previous exercise?

- Work flow

- List all entity sets found
- For each entity set, find relationship with other entity and itself.
- Choose only the relationship which are right to our context.

Answer

- AUTHOR <Write> BOOK
- EDITOR <Publish> BOOK
- CUSTOMER <Purchase> BOOK
- CUSTOMER <Has> ADDRESS

Attribute of Relationship

- Attribute of relationship
 - A relationship can also have **descriptive attributes**.
Descriptive attribute used to *record information about relationship*, not about participating entities.
 - Can be an attribute or a set of attributes.
 - Ex: we may wish to record that Sok works in the pharmacy department **since** January 1991.
 - Sok <works> department : *since*

Attribute of Relationship Set

- Why descriptive attribute?
 1. The information of relationship is interesting and exists only in a given context.
 2. The attribute is semantically belonged to the relationship
 - The value of attribute **Since** exists only when the employee starts working in a department, so its existence depends on the existence of the relationship between employee and department.

Degree of Relationship

- Degree of relationship is the number of participating entities in a relationship.
 - $N=1$, Unary relationship or recursive
 - $N=2$, Binary relationship
 - $N=3$, Ternary relationship, ...

Degree of relationship type

1. Unary / Recursive Relationship: A relationship where an entity is related to itself (e.g. an employee supervises other employees).
2. Binary Relationship: The most common type, between two entities (e.g., Teacher teaches Course).
3. Ternary Relationship: Involves three entities (e.g., Doctor, Patient, and Hospital are related in a healthcare system).

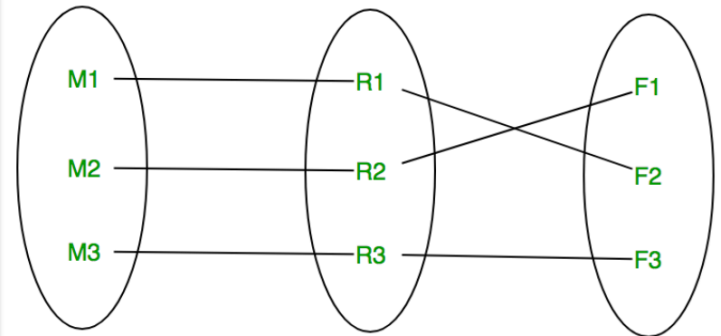
Cardinality of Relationship

- **Cardinality of relationship** defines the *maximum* number of possible relationship for every entity.
- It is used to restrict the relationship between entity according to business rule of a company.
- 3 different types of cardinality of relationship:
 - One-to-one (1:1)
 - One-to-many (1:n)
 - Many-to-many (m:n)

Cardinality of Relationship

- **One-to-one (1:1):** One instance of an entity is related to one instance of another entity.

An entity in E can have relationship with only one entity from F, and an entity in F can only have the relationship with only one entity from E. There is one-to-one mapping between E and F.



Ex: An employee can have only one driving license.
A driving license can belong to only one employee.

Cardinality of Relationship

- **One-to-one (1:1)**

Ex: Each person has exactly one passport, and each passport is issued to exactly one person.

Person_ID	Name	DOB	Address
1	Alice Smith	1985-02-15	123 Elm St, Springfield
2	Bob Johnson	1990-08-22	456 Oak St, Springfield
3	Carol White	1978-03-10	789 Pine St, Springfield
4	David Brown	1965-11-05	321 Maple St, Springfield
5	Eva Green	1992-07-18	654 Birch St, Springfield

Passport_ID	Person_ID	IssueDate	ExpiryDate	Nationality
A1234567	1	2015-01-15	2025-01-15	USA
B9876543	2	2016-06-20	2026-06-20	USA
C1928374	3	2017-03-12	2027-03-12	USA
D5647382	4	2018-09-29	2028-09-29	USA
E1112233	5	2019-11-15	2029-11-15	USA

Cardinality of Relationship

- **One-to-one (1:1)**

Ex: An Order has one Payment, but a Payment is associated with one Order.

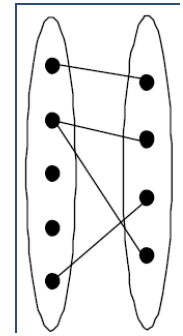
OrderID	OrderDate	UserID	TotalAmount	PaymentStatus
1	2024-10-01 10:00:00	1	919.98	Paid
2	2024-10-05 14:30:00	2	19.99	Paid
3	2024-10-10 09:15:00	1	119.98	Pending

PaymentID	OrderID	PaymentDate	Amount	PaymentMethod	Status
1	1	2024-10-01 10:05:00	59.97	Credit Card	Completed
2	2	NULL	29.99	PayPal	Pending
3	3	2024-10-03 14:20:00	39.99	Credit Card	Completed

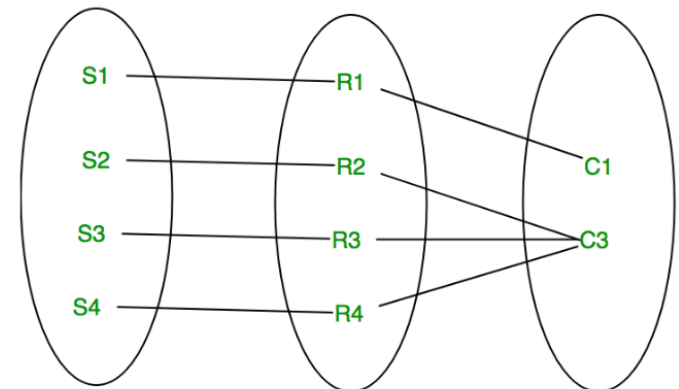
Cardinality of Relationship

- **One-to-many (1:M):** One instance of an entity is related to multiple instances of another entity. It can be applied in both directions.

An entity in E can have relationship with many entities from F, and an entity in F can only have the relationship with only one entity from E. There is one-to-many mapping between E and F.



Ex: A course can be attended by many students. A student can attend at most one course.



Cardinality of Relationship

- **One-to-many (1:M):**

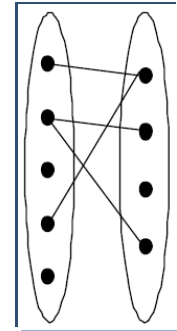
Ex: A User can place multiple Orders, but an Order is placed by one User.

OrderID	OrderDate	UserID	TotalAmount	PaymentStatus
1	2024-10-01 10:00:00	1	919.98	Paid
2	2024-10-05 14:30:00	2	19.99	Paid
3	2024-10-10 09:15:00	1	119.98	Pending

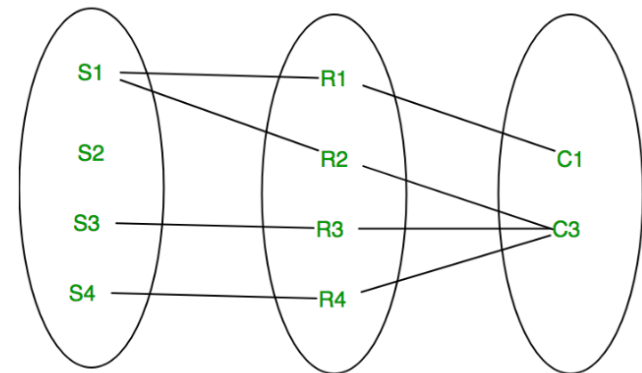
Cardinality of Relationship

- **Many-to-many (M:M):** Multiple instances of an entity are related to multiple instances of another entity.

An entity in E can have relationship with many entities from F, and an entity in F can have the relationship with many entities from E. There is many-to-many mapping between E and F.



Ex: A course can be attended by many students. A student can attend many course.



Cardinality of Relationship

- **Many-to-many (M:M):**

Ex: an Employee can opt for multiple policies, and one policy can be opted by multiple employees.

EmpID	Name	Address	PolicyID	PolicyName	Description
1	Emp1	New Delhi	P1	Policy 1	Medical Policy
1	Emp1	New Delhi	P2	Policy 2	Subsidize Food Policy
2	Emp2	Mumbai	P1	Policy 1	Medical Policy
3	Emp3	Uttar Pradesh	P1	Policy 1	Medical Policy
3	Emp3	Uttar Pradesh	P2	Policy 2	Subsidize Food Policy
3	Emp3	Uttar Pradesh	P3	Policy 3	Subsidize Transport Policy

Cardinality of Relationship

- **Many-to-many (M:M):**
- Many-to-many relationships are *not supported* by the relational model and must be resolved by splitting the original M:M relationship set into two 1:M relationship sets.
- Usually, the unique identifiers of the two entity sets participate in building the unique identifier of the third entity set.
- Example: Consider the M:M relationship between Student and Course. To model this, we split into 1:M relationship between Student and Enrollment, and 1:M relationship between Course and Enrollment.

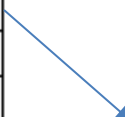
Cardinality of Relationship

- **Many-to-many (M:M):**

EmpID	Name	Address
1	Emp1	New Delhi
2	Emp2	Mumbai
3	Emp3	Uttar Pradesh

PolicyID	PolicyName	Description
P1	Policy 1	Medical Policy
P2	Policy 2	Subsidize Food Policy
P3	Policy 3	Subsidize Transport Policy

EmpID	PolicyID	Date
1	P1	1/1/2024
1	P2	1/2/2024
2	P1	1/3/2024
3	P1	1/4/2024
3	P2	1/5/2024
3	P3	1/6/2024



This is a bridge table used to implement many to many relationship.

Associative Entity

- An associative entity (also called a bridge entity or junction table) is used to model many-to-many relationships by breaking them down into two one-to-many relationships.
- It often has its own attributes.
- Example: Consider the M:M relationship between Student and Course. To model this, associative entity called Enrollment can be created with additional attributes like grade, enrollment_date, etc.

Participation Constraint (Optionality)

- A Participation Constraint (optionality) is a rule in an ER model that defines whether all instances of an entity must participate in a given relationship.
- This constraint specifies the **minimum** number of times an entity instance is required to participate in a relationship.
- There are two main types of Participation Constraint (optionality) : total participation and partial participation.

Participation Constraint (Optionality)

1. Total or Mandatory Participation

- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- Each entity set must participate in a relationship and it cannot exist without that participation; the participation is compulsory.

Ex: If each student must enroll in a course, the participation of student will be total.

Participation Constraint (Optionality)

2. Partial or Optional Participation

- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- There is no requirement for every instance of the entity to be associated with the other entity.

Ex: If some courses are not enrolled by any of the student, the participation of course will be partial.

Participation Constraint (Optionality)

Example: EMPLOYEES <Manages> DEPARTMENTS.

- Every department is required to have a manager. This requirement is an example of a Participation constraint (optionality);
- The participation of the entity Departments in the relationship <Manages> is said to be *total*.
- A participation that is not total is said to be *partial*.
- The participation of the entity set Employees in <Manages> is *partial*, since not every employee gets to manage a department.

Participation Constraint (Optionality)

Example: EMPLOYEES <work in> DEPARTMENTS.

- On the <Work_In> relationship set, it is natural to expect that each employee works in at least one department and that each department has at least one employee.
- This means that the participation of both Employees and Departments in <Work_In> is *total*.

Cardinality and Optionality

- Optionality = must or may? Min 1 or 0?
- Cardinality = max one or more?

Examples:

1. Each student may ask from one or more teachers.
2. Each teacher may educate one or more students.
3. An employee may manage one or more employees.
4. An employee must be managed by one employee.

Weak Entity

- **A weak entity** is an entity that cannot be **uniquely identified** by its own attributes alone and needs a foreign key from another entity (called *strong*, or *owner*, or *identifying owner entity*).
- It **usually has** a **partial key** (an attribute that uniquely identifies instances of a weak entity relative to its related strong entity) and a foreign key that links to the primary key of the owner entity.

Weak Entity

Example: Hotel rooms are sub unit of a hotel.

- ROOM is an entity set containing all hotel room, and HOTEL is an entity set containing all hotel of a given context.
- Many hotels may have the room number 201, so the rooms in ROOM can't be identified by their number. It can be identified by considering the identifier of the hotel and the number of the room.
- ROOM is considered as a weak entity, and HOTEL is a strong entity.

Weak Entity

Example:

- Room in a Hotel – Room Relationship
 - Owner Entity: Hotel
 - Weak Entity: Room
 - Attributes of Dependent: room_num, capacity, type
 - Partial key: room_num
 - Foreign key: hotel_number (from Hot entity)

Weak Entity

Examples:

- Dependent in an Employee-Dependent Relationship
 - Owner Entity: Employee
 - Weak Entity: Dependent
 - Attributes of Dependent: dependent_name, relation
 - Partial key: dependent_name
 - Foreign key: employee_id (from Employee entity)

Weak Entity

Examples:

- Class Section in a Course – Class Section Relationship
 - Owner Entity: Course
 - Weak Entity: Class Section
 - Attributes of Dependent: section_number, semester
 - Partial key: section_number
 - Foreign key: course_id (from Course entity)

Weak Entity

The following restrictions must hold:

1. The owner entity set and the weak entity set must participate in a **one-to-many** relationship set. (One owner entity is associated with one or more weak entities, but each weak entity has a single owner). This relationship set is called the **identifying relationship** set of the weak entity set.
2. The weak entity set must have **total participation** in the identifying relationship set.

Supertype and Subtype

- When a group of instances has special properties such as attributes or relationship sets that exist only for that group, it makes sense to subdivide an entity set into subsets.
- Example: Employees have id, name, contract_id (for only contract employees) and hourly wage (for only hourly employees).
- We can classify Employee into Contract_Emps and Hourly_Emps.
- The attributes of Employees are inherited by the Contract_Emps and Hourly Employee and that Hourly_Emps is a (**ISA**) Employees.

Class/ISA Hierarchies

- In this example, we have an Employees entity as a supertype and two subtypes: Contract_Emps and Hourly_Emps. Hourly_Emps and Contract_Emps are subclasses of Employees.
- The entity set is a superset called a parent or superclass or supertype. Each group is a subset called a child, or subclass or subtype.
- A subset consists in all attributes of the superset and takes over all relationship sets of the superset. A subset exists only along with other subsets and may have subsets of its own.

Class/ISA Hierarchies

The 2 main reasons for identifying subclasses:

1. To add *attributes* that makes sense only for the entities in the subclasses.

Ex: hourly_wage does not make sense for Contract_Emps entity.

2. To identify the *specific set of entities* that participates in some relationship.

Ex: we want to define <Manages> relationship so that the participating entity sets are Senior_Emps and Departments, to ensure that only senior employees can be managers.

Assignment

A company database needs to store information about employees (identified by ssn, with salary and phone as attributes), departments (identified by dnu, with dname and budget as attributes), and children of employees (with name and age as attributes). Employees work in departments; each department is managed by an employee; a child must be identified uniquely by name when the parent (who is an employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.

- *Find entity sets in the above database.*
- *For each entity set list attributes defining the entity set.*
- *Find relationship set between entity sets in the database.*

Summary

- ER model
- Entity
- Attribute
- Relationship
- Degree of Relationship
- Cardinality of Relationship
- Participation Constraints
- Weak Entity
- Associative Entity
- ISA hierarchies

