Database Analysis and Design

Lesson 6: Database Design with Entity Relationship Model

Ms. SEAK LENG

Learning Objectives

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

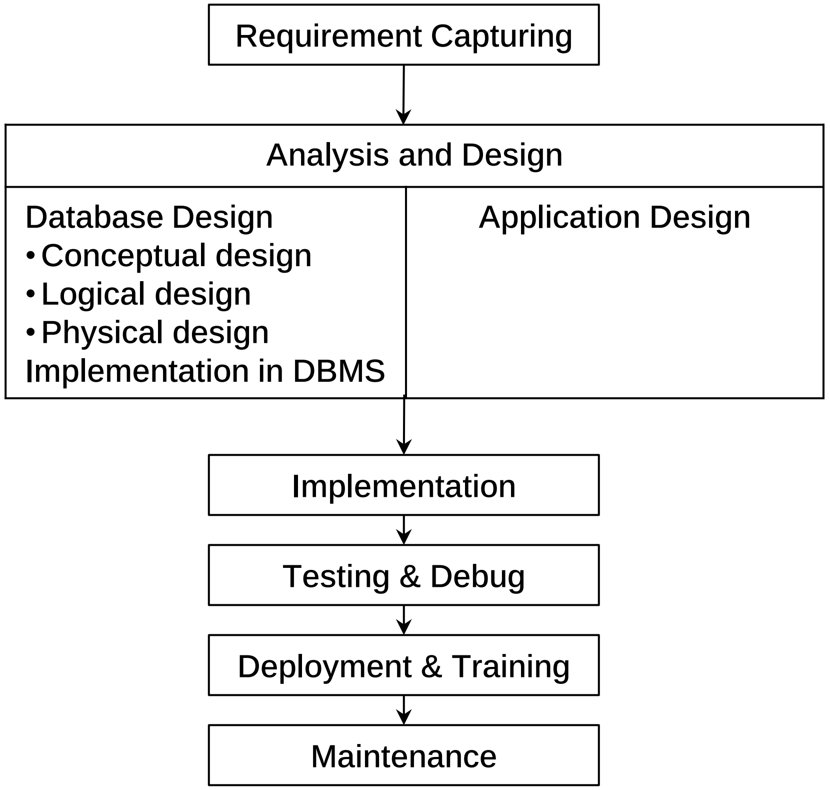
1. Explain the purpose and importance of database design.
2. Identify key elements of the Entity Relationship Model.
3. Distinguish between entity sets, attributes, and relationship sets.
4. Differentiate between binary and ternary relationships.

Overview of Database Design

- **Definition**: Database design is the process of structuring data for efficient storage, retrieval, and management in a database.
- **Purpose of database design**:
  - Ensures data integrity and security.
  - Reduces redundancy.
  - Enhances database performance.

Overview of Database Design

- Overview of database design
  - Database design as a part of application development



Conceptual Design With ER Model

- **Definition of ER Model**: An ER model is a high-level data model that represents data structures using entities, attributes, and relationships.
- **Purpose of ER Model**:
  - Provides a visual representation of data.
  - Facilitates communication between stakeholders by simplifying complex data requirements.

Conceptual Design With ER Model

- Design process
  1. Identity *entity sets* from the requirements specification
  2. Find *attributes* and *key* of entity for each entity
  3. Find *relationship* set between entity sets and constraints

Conceptual Design With ER Model

- A case study used as the example

This case study describes a simple Library Management System. Assume the requirements for the system were written by your client exactly in these terms:

"The system will manage author's and loaner's information, and keep track of books loaned. The borrower's information include name, address, e-mail, and phone. The author's information include name, address and e-mail.

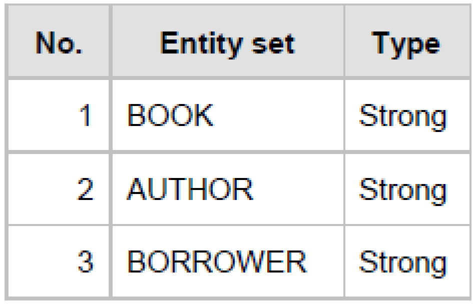
New books, authors and clients are entered into the system. When a client checks out a book, the system will register the date the book was loaned and calculate the days the book can be loaned. It will also calculate the date the book is due to be returned. If the borrower returns the book late, he must pay a fine based on the number of days overdue.“

Conceptual Design With ER Model

1. Identify entity from the requirements specification
   - Highlight all nouns you encounter.
   - Eliminate the *duplicated* and non-relevance noun
   - Noun can be attribute or entity
   - Nouns that depends on other noun should be attributes.
   - Nouns that determines other nouns should be entity
   - Keep only the needed entity.

Conceptual Design With ER Model

1. Identify entity from the requirements specification
   - Nouns from the declaration: books, authors, name, address, e-mail, loaner, client, borrowers, name, address, e-mail, phone, loan date, return date, loan days, fine
   - Relevance noun after duplicate elimination: books, authors, name, address, e-mail, borrowers, phone, loan date, return date, loan days, fine
   - Noun after dependencies elimination: books, authors, borrowers.
   - Entity: BOOK, AUTHOR, BORROWER.



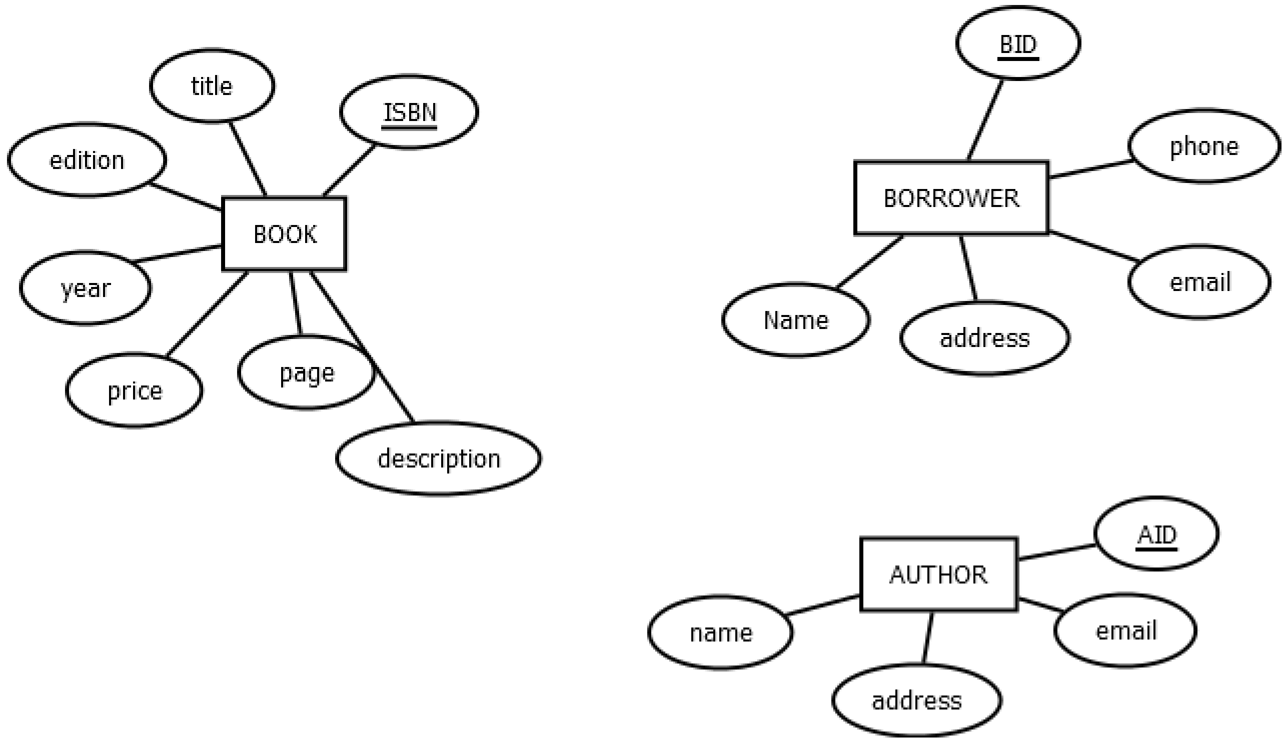
No.	Entity set	Type
1	BOOK	Strong
2	AUTHOR	Strong
3	BORROWER	Strong

Conceptual Design With ER Model

2. Find attributes and key of entity for each entity
   - Identity the attribute from the requirement specification
   - Communicate with client for other needed attributes
   - Eliminate *derived* attributes
   - Decide to keep *composite attributes or decompose* them
   - Move *multi-value* attribute to other entity set
   - Choose a unique identifier

Conceptual Design With ER Model

2. Find attributes and key of entity for each entity
   - Example: Attributes and unique key of entity set from the requirement specification



Conceptual Design With ER Model

2. Find attributes and key of entity for each entity
   - How about other attributes?
     - Loan date exists only when a book is borrowed.
     - Return date exists only when a book is borrowed then returned.
     - Due date is a derived attribute. It can be calculated if the limited loan days is predefined and the loan date is known.
     - Fine exists only when a book is borrowed and returned later than the due date.

Conceptual Design With ER Model

3. Find relationship between entity and constraints
   - Examine the relationships between entities
   - Eliminate the *redundant* relationship
   - Specify weak relationship sets and weak entity
   - Describe the cardinality and constraints of the relationship
   - Example
     - BOOK and BORROWER

     Borrower borrow a copy of book ⇔ A copy of book is borrowed by a borrower.

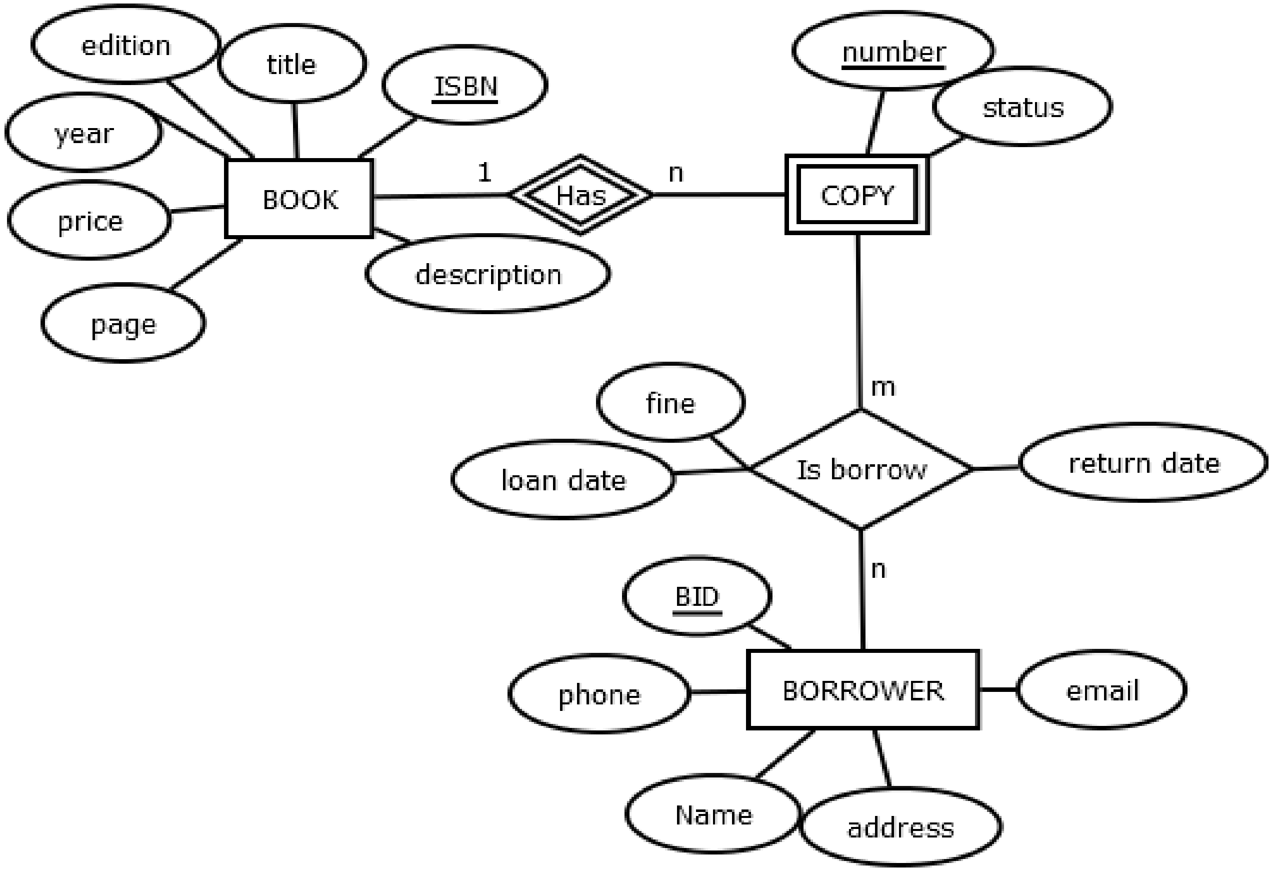
     A new entity set appears. Entity set COPY is a weak entity set which is identifier by BOOK.

     Relationship between BORROWER and COPY is "Is borrowed". Information about the relationship is loan date, return date, and fine.

Conceptual Design With ER Model

3. Find relationship between entity and constraints

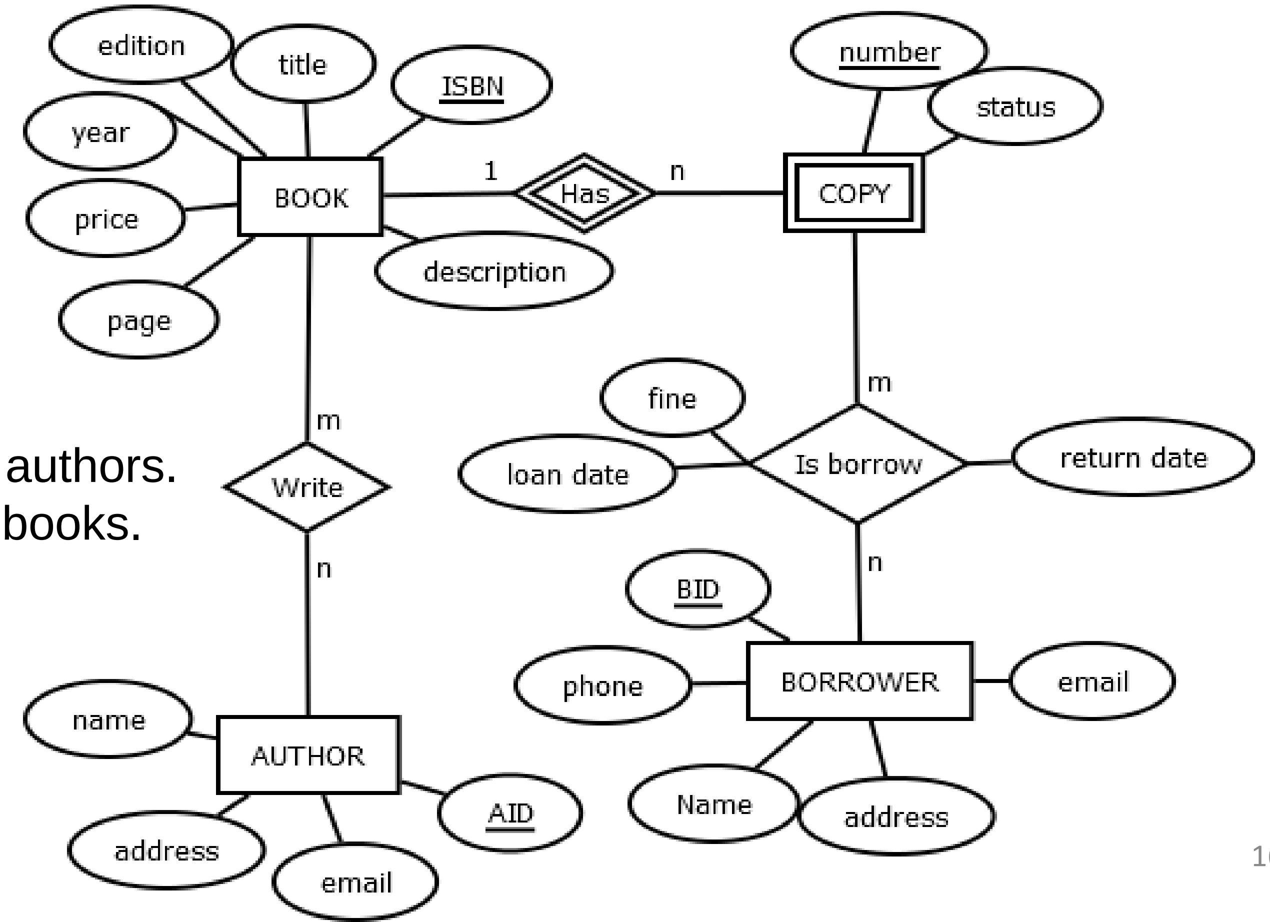
- Example: BOOK and BORROWER



Conceptual Design With ER Model

3. Find relationship between entity and constraints

– Example: BOOK and AUTHOR



A book is written by authors.
⇔ An author writes books.

Conceptual Design With ER Model

- Business rule
  1. Some business rule can be incorporated into Entity Relationship diagram.
     - Ex: a borrower can borrow many books, and a book can be borrowed by many borrowers.
  2. Some others have to implemented by program
     - Ex: If the borrower does not return the book on time, he will be fined 0.1% of the book's price per day.
- Remark
  - ➢ Conceptual modeling is an iterative (repeat) process.
  - ➢ *Many* possible solutions for a single system.

Conceptual Design With ER Model

Developing a conceptual design presents several choices, including the following:

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
- What are the relationship and their participating entity?

Entity or Attribute

Whether a property should be modeled as an attribute or as an entity set depends on how central it is to the system and how much detail or structure it needs.

1. Does it represent a thing or concept with distinct existence?

- Entity: If the object or concept has a distinct, identifiable existence in the real world, it is an entity. Ex: Customer, Reservation, Product, Department.
- Attribute: If it is a description or characteristic of an entity, it is an attribute. Ex: Name, Email, Price.

Entity or Attribute

2. Can it have multiple instances?

- Entity: if there can be multiple instances or occurrences of it in the system, it should be an entity. Ex: A customer can have many instances (Jonh, Jane, Alex, etc)
- Attribute: If it can only exist as a single instance within the system (it doesn’t have meaningful duplicates), you might omit it as an entity. Ex: Country in a Customer profile could be an attribute rather than an entity if only one country is stored per customer.

Entity or Attribute

3. Does it have relationships with other entities?

- Entity: If the object or concept has relationships with other entities in the system, it is likely an entity. Ex: Employee might have a relationship with Department, Project, or Task.
- Attribute: If it doesn’t have a relationships with other entities and is purely descriptive, it is more likely an entity. Ex: Phone_Number is an attribute of a Customer entity.

Entity or Attribute

4. Is it a unique or specific object in the domain?

- Entity: If it represents a unique or specific object in the real world that the system needs to track separately, it should be an entity. Ex: ProductID could be unique identifiers for an entity Product.
- Attribute: If it doesn’t require distinct tracking and only describes another object, it is likely an attribute.

Entity or Attribute

5. Can it have multiple values?

- Entity: If the object can have multiple values in the system (e.g multiple addresses, orders, or payments), it could be an entity.
- Attribute: if the object can have only one value per instance (such as a person's Firstname, or email), it is an attribute.

Entity or Attribute

5. Can it have multiple values?

- Entity: If the object can have multiple values in the system (e.g multiple addresses, orders, or payments), it could be an entity.
- Attribute: if the object can have only one value per instance (such as a person’s Firstname, or email), it is an attribute.

Entity or Attribute

6. How Frequently Will It Be Repeated or Shared?

- Entity: If the concept will appear in multiple records across the database and may need to be referenced or managed independently, it’s usually better as an entity for the consistency. Ex, if many entities share the same address, creating "Address" as an entity allows you to store it once and link it to many people.
- Attribute: If it’s unique to a single record or doesn’t need to be reused, it’s often more efficient to keep it as an attribute.

Entity or Attribute

Example: Address as an Entity or Attribute?

Entity or Attribute

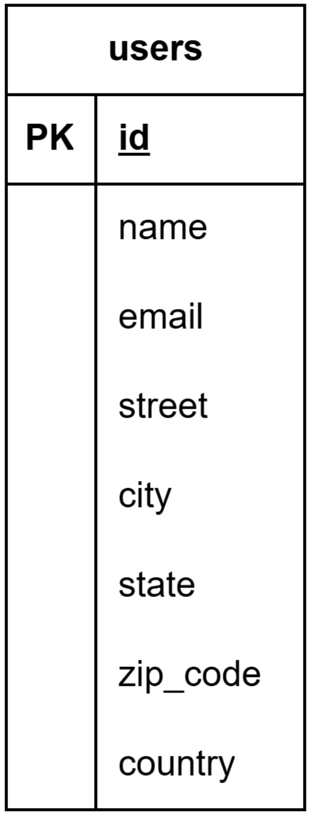
Example: Address as an Attribute

- One instance has only one address value.
- The address data is straightforward and won’t be reused or referenced by other parts of the system.
- There is no need for complex queries on the address data, such as filtering by state or city.
- The address is tightly coupled with the primary entity and isn’t likely to change independently

Entity or Attribute

Example: Address as an Attribute

- **Pros**: Easier to manage for simple applications, fewer joins, straightforward data retrieval.
- **Cons**: Redundant storage of address if multiple users share the same address. Limited ability to update address independently. Limit each user to have at most only one address.



Entity or Attribute

Example: Address as an Entity

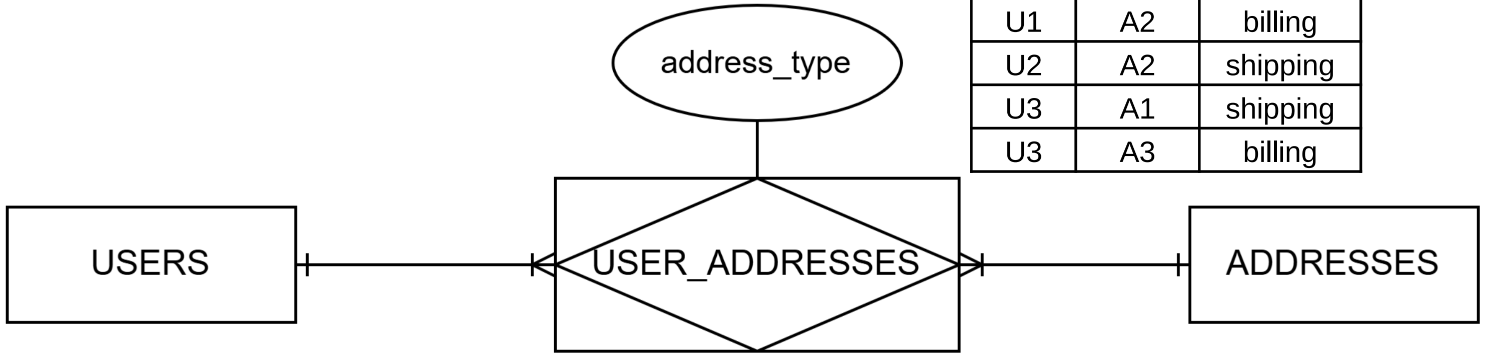
- There is a need to associate multiple addresses (like billing address, shipping address) with an entity.
- The address data needs to be reusable or shared across multiple entities
- The system requires complex address queries or address management features (like updating addresses independently, address verification, or address version history)
- There is a need to minimize redundancy for storage efficiency or to ensure data consistency across related tables.

Entity or Attribute

Example: Address as an Entity

- Pros: Reduced redundancy, addresses are normalized and managed separately, easier to update or modify shared addresses, and better support for complex queries.
- Cons: More complex schema requiring joins, which may impact query performance in larger databases.

U1	A1	shipping
U1	A2	billing
U2	A2	shipping
U3	A1	shipping
U3	A3	billing



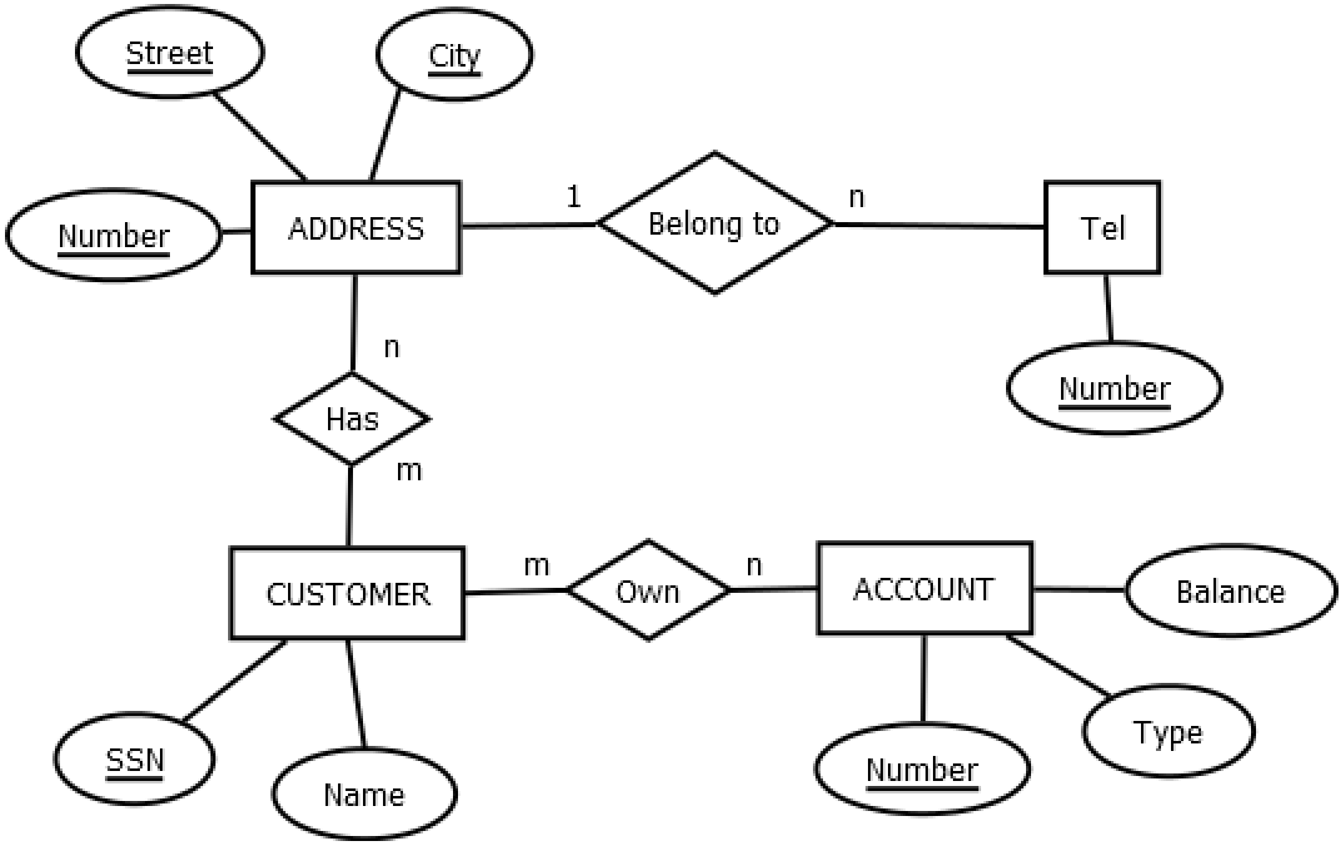
Entity or Attribute

- Exercise

Design a database for a bank, including information about customers and their accounts. Information about a customer includes their name, address(s), phone(s), and Social Security number. *A customer can have many addresses, and for each address, it is possible to have many phones.* Accounts have numbers, types (e.g., savings, checking) and balances. Also record the customer(s) who own an account. A customer can have many accounts in the bank. Draw the E /R diagram for this database.

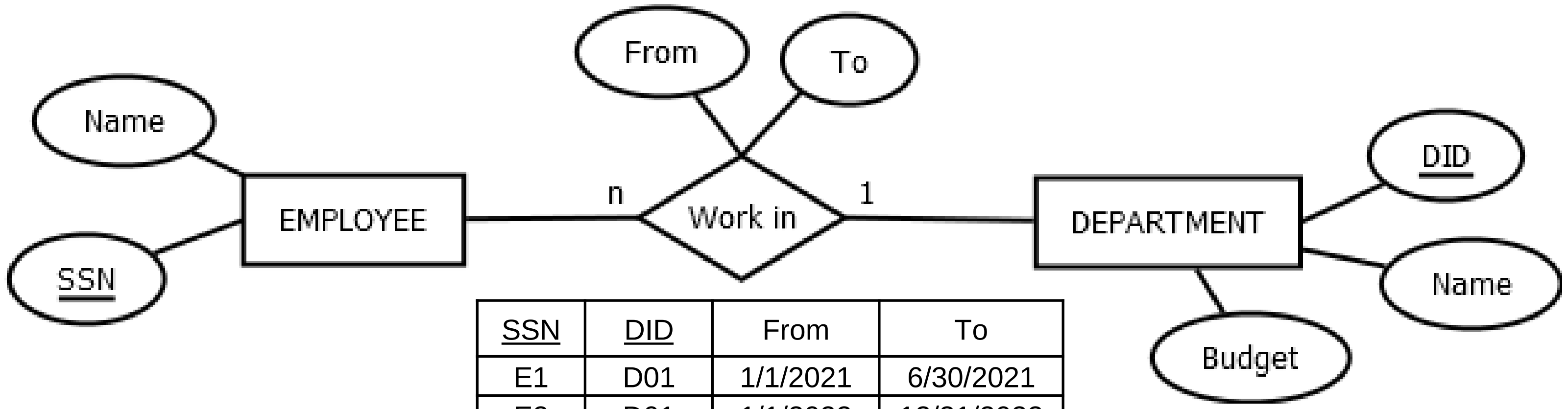
- Workflow
  - Find needed entity sets
  - For each entity set, find attributes. Remember that an attribute has *only one value for an entity*.
  - Find relationship between each entity sets and specify the cardinality according to the declaration of the exercise.

Entity or Attribute



Entity or Attribute

- Ex: An employee can work in a department during a given period. It is possible for an employee to work in a given department over more than one period.
  - First model: Violate the concept that relationship can only be identified by participating entity. The problem is that we want to record several values for the descriptive attributes for each instance of the <Work in> relationship.

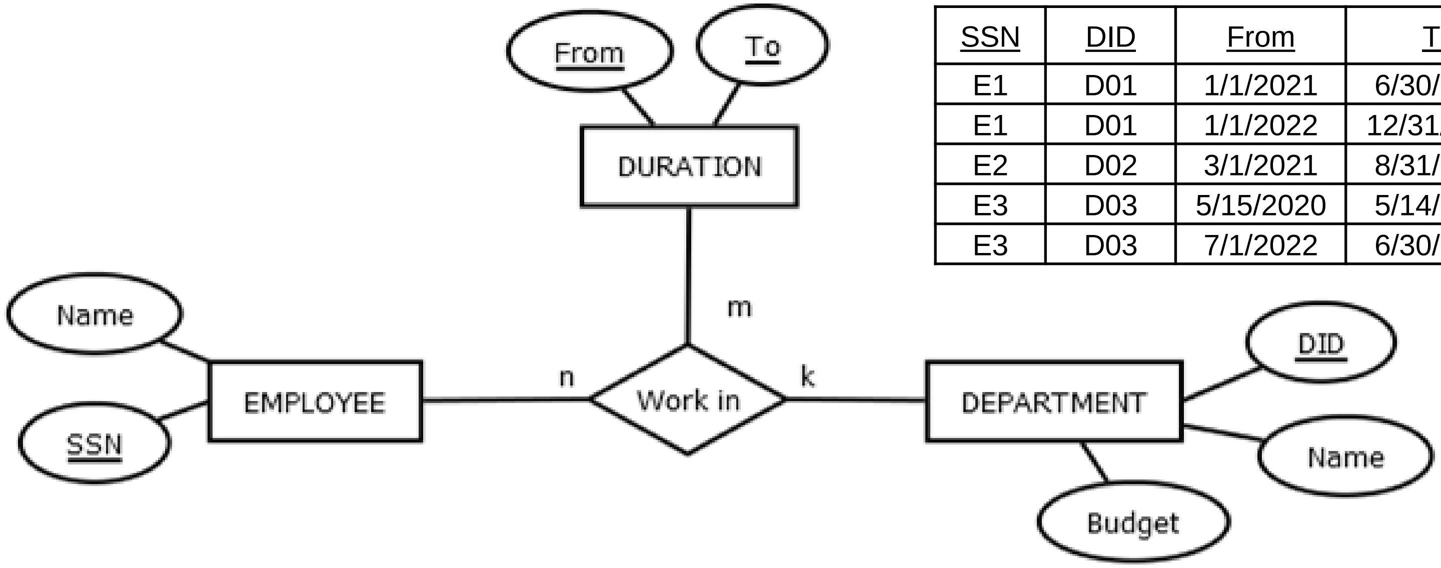


SSN	DID	From	To
E1	D01	1/1/2021	6/30/2021
E2	D01	1/1/2022	12/31/2022
E3	D02	3/1/2021	8/31/2021

Entity or Attribute

- Ex: An employee can work in a department during a given period. It is possible for an employee to work in a given department over more than one period.
  - Second model: Attributes of relationship is modeled as an **entity set**. We can address the previous problem by introducing an entity set DURATION with attribute *From* and *To*.

SSN	DID	From	To
E1	D01	1/1/2021	6/30/2021
E1	D01	1/1/2022	12/31/2022
E2	D02	3/1/2021	8/31/2021
E3	D03	5/15/2020	5/14/2021
E3	D03	7/1/2022	6/30/2023

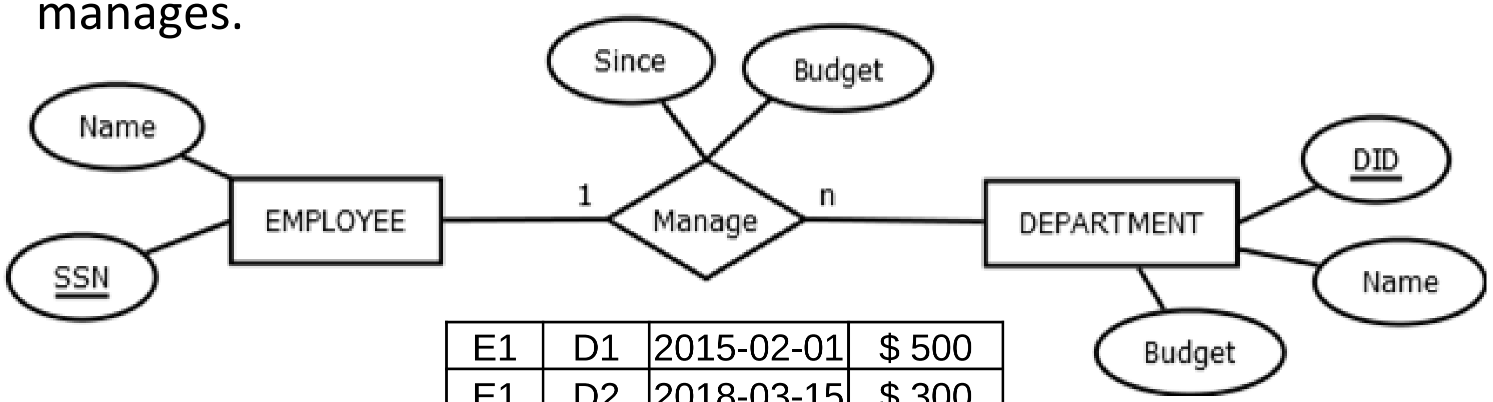


Entity Set or Relationship Set

- A case study

In a company, there are several departments. Each department is managed by an employee which plays the role as the manager of the department. An employee can manage *many* departments and is given a discretionary budget.

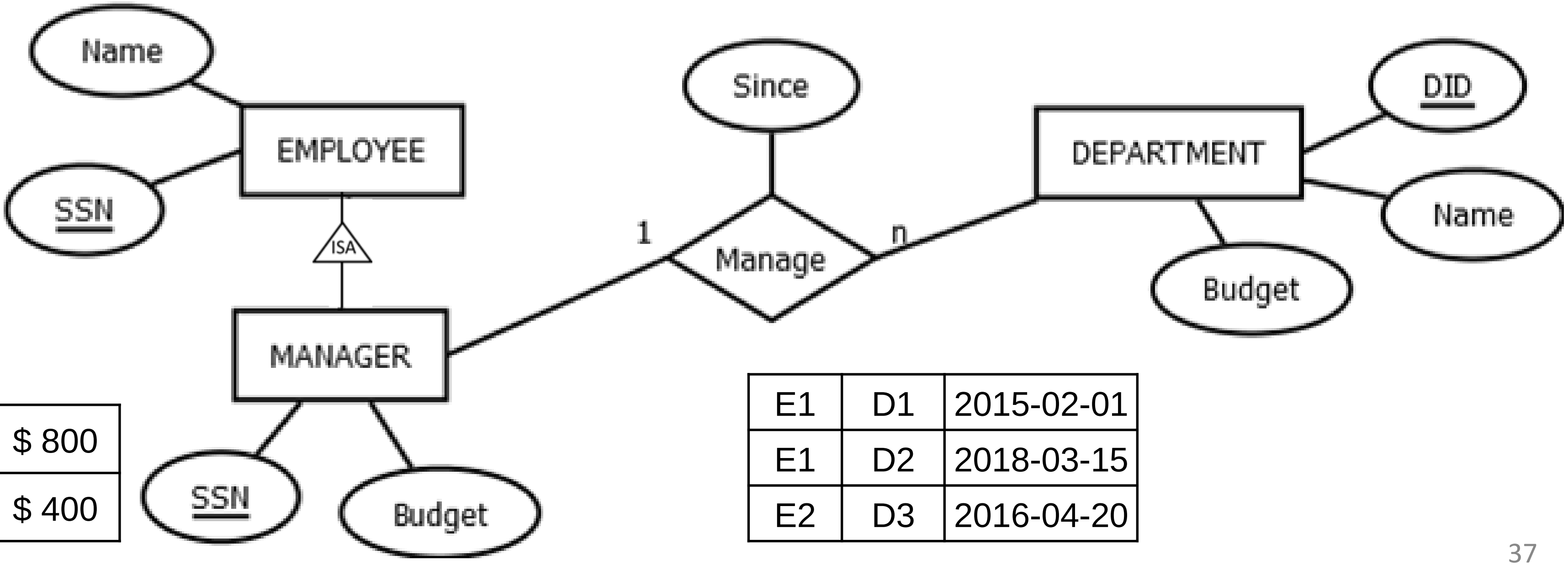
- First model: ER diagram 1
  - In case manager gets a separate budget for each department s/he manages.



E1	D1	2015-02-01	$ 500
E1	D2	2018-03-15	$ 300
E2	D3	2016-04-20	$ 400

Entity Set or Relationship Set

- Second model: ER diagram 2
  - In case manager gets the budget, which is the sum of budget of each department.
  - New entity set MANAGER can be placed below EMPLOYEE in an ISA hierarchy. While every MANAGER has a Budget, each manager may have a different starting date for each different department.

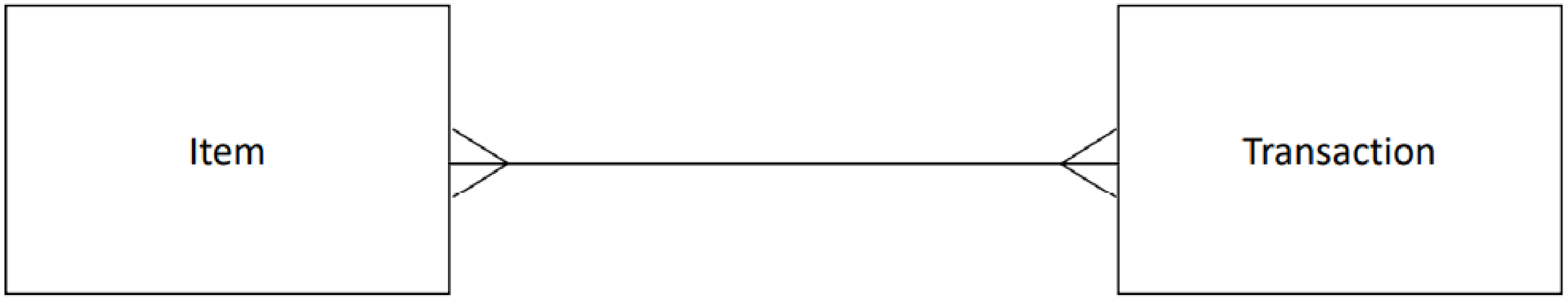


E1	$ 800
E2	$ 400

E1	D1	2015-02-01
E1	D2	2018-03-15
E2	D3	2016-04-20

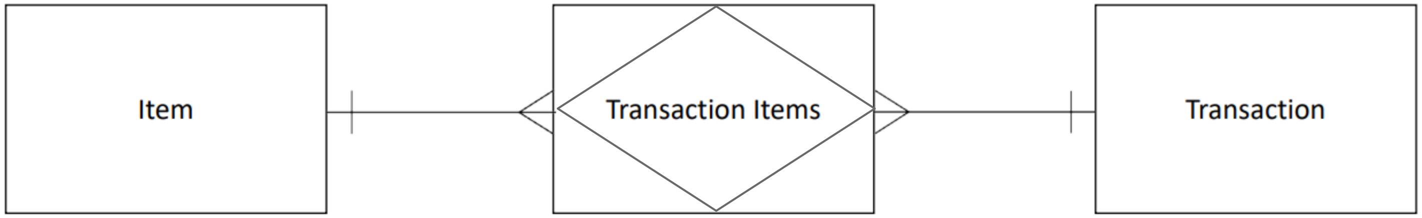
Many-to-Many relationship

- Conceptually, a many to many relationship can be represented using crow's foot notation.
- For example, an item can appear on several transactions, and a transaction can contain several items.



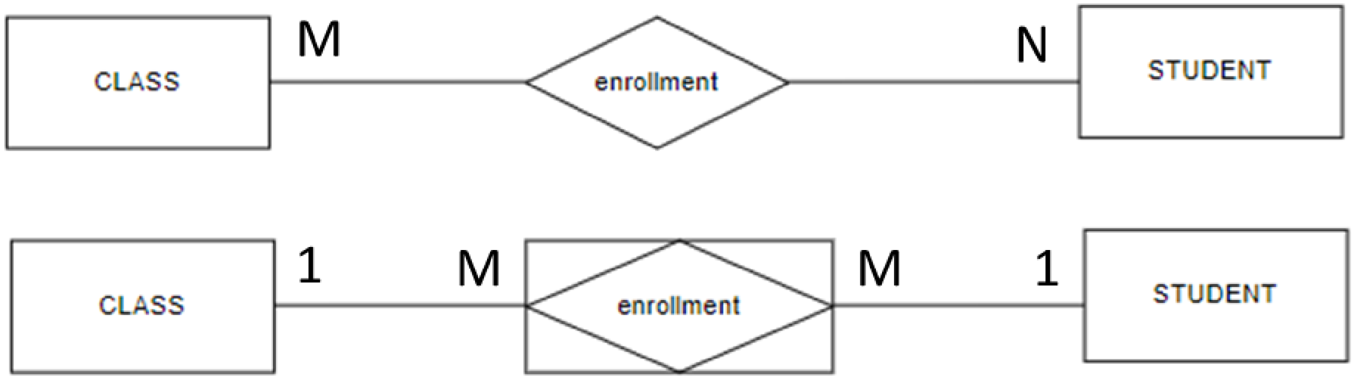
Many-to-Many relationship

- M:M is strictly conceptual. M:M relationships are *not supported* by the relational model and must be resolved by splitting the original M:M relationship set into two 1:N relationship sets.
- In actual design, a many to many relationship requires a separate table, called an associative table, to capture the relationship between the two entities.



Many-to-Many relationship

- So, When finalizing ER, you should not leave many to many relationships in the diagram.
- Ex: 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.



One-to-One relationship

- 1:1 relationships are not as common as 1:N or M:M relationships in data modeling. Often, this type of relationship can be captured sufficiently within a single entity.
- We split one entity into two separate entities with a 1:1 relationship when certain attributes are best stored in a separate table for reasons such as organization, security, or performance.

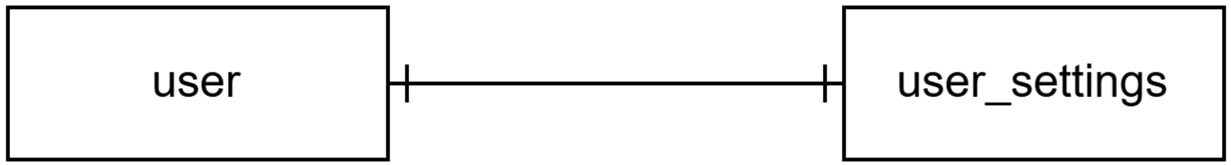
One-to-One relationship

When to Use One-to-One Relationships

1. Avoiding Nulls in a main entity

If some attributes in an entity are optional, null or only apply to specific entities, moving them to a separate entity can help reduce the number of NULL values in the main entity.

- Example: User entity stores general user information. User_Settings entity stores optional settings like theme and autologin setting.

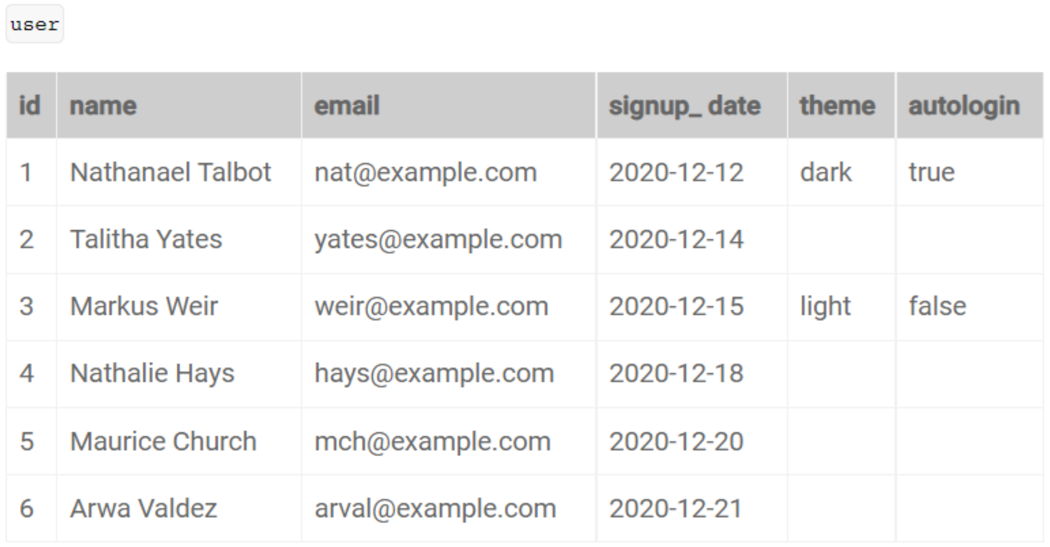


One-to-One relationship

When to Use One-to-One Relationships

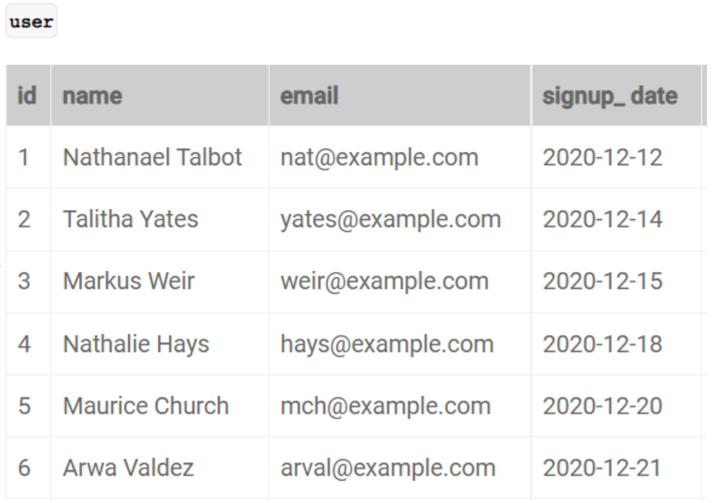
1. Avoiding Nulls in a main entity

- Example:



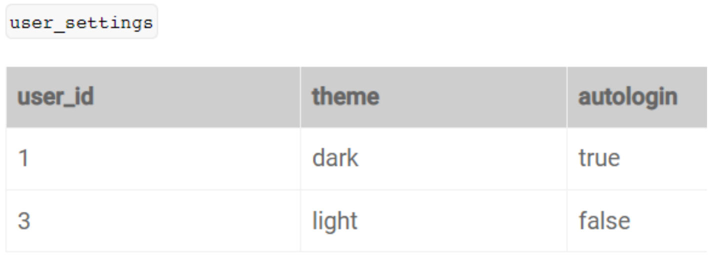
user

id	name	email	signup_ date	theme	autologin
1	Nathanael Talbot	nat@example.com	2020-12-12	dark	true
2	Talitha Yates	yates@example.com	2020-12-14		
3	Markus Weir	weir@example.com	2020-12-15	light	false
4	Nathalie Hays	hays@example.com	2020-12-18		
5	Maurice Church	mch@example.com	2020-12-20		
6	Arwa Valdez	arval@example.com	2020-12-21		



user

id	name	email	signup_ date
1	Nathanael Talbot	nat@example.com	2020-12-12
2	Talitha Yates	yates@example.com	2020-12-14
3	Markus Weir	weir@example.com	2020-12-15
4	Nathalie Hays	hays@example.com	2020-12-18
5	Maurice Church	mch@example.com	2020-12-20
6	Arwa Valdez	arval@example.com	2020-12-21



user_settings

user_id	theme	autologin
1	dark	true
3	light	false

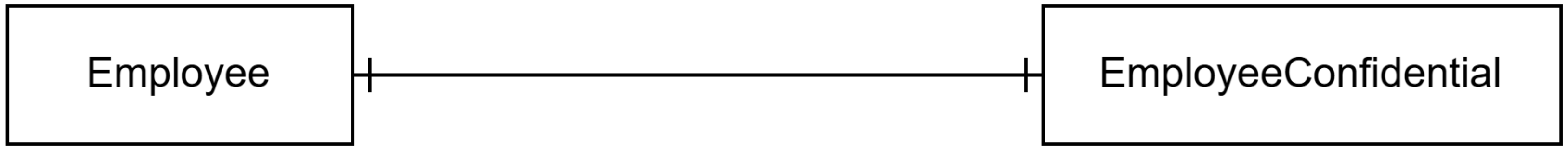
One-to-One relationship

When to Use One-to-One Relationships

2. Security or Privacy

Sensitive data that shouldn't be accessed as frequently can be stored separately and secured.

- Example: Employee entity contains general employee details like id, name, department. EmployeeConfidential entity stores social security numbers and salary details or passwordHash.



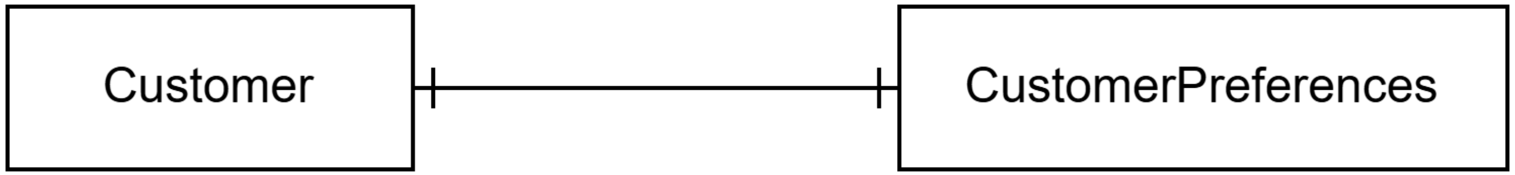
One-to-One relationship

When to Use One-to-One Relationships

3. Performance Optimization

Splitting large or infrequently accessed data into separate tables can reduce the size of frequently queried tables, improving performance.

- Example: Customer entity for common customer data like id , name, email. CustomerPreferences for rarely accessed user settings like preference.



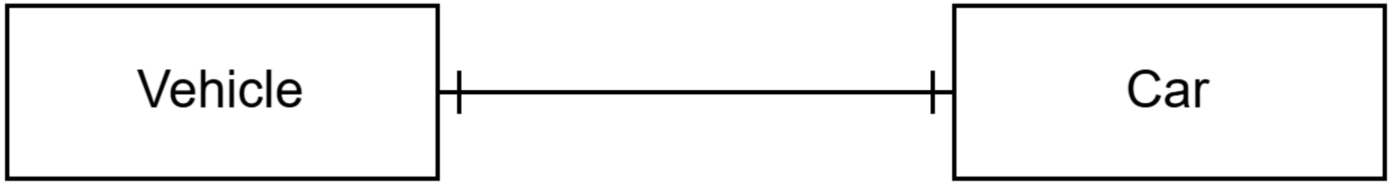
One-to-One relationship

When to Use One-to-One Relationships

4. Class hierarchy (ISA)

A one-to-one relationship can model entities with a specialized subset of attributes.

- Example: Vehicle entity for general information, like id, type, model, manufacturer. Car entity for vehicle-specific details, like engineCapacity, fuelType.



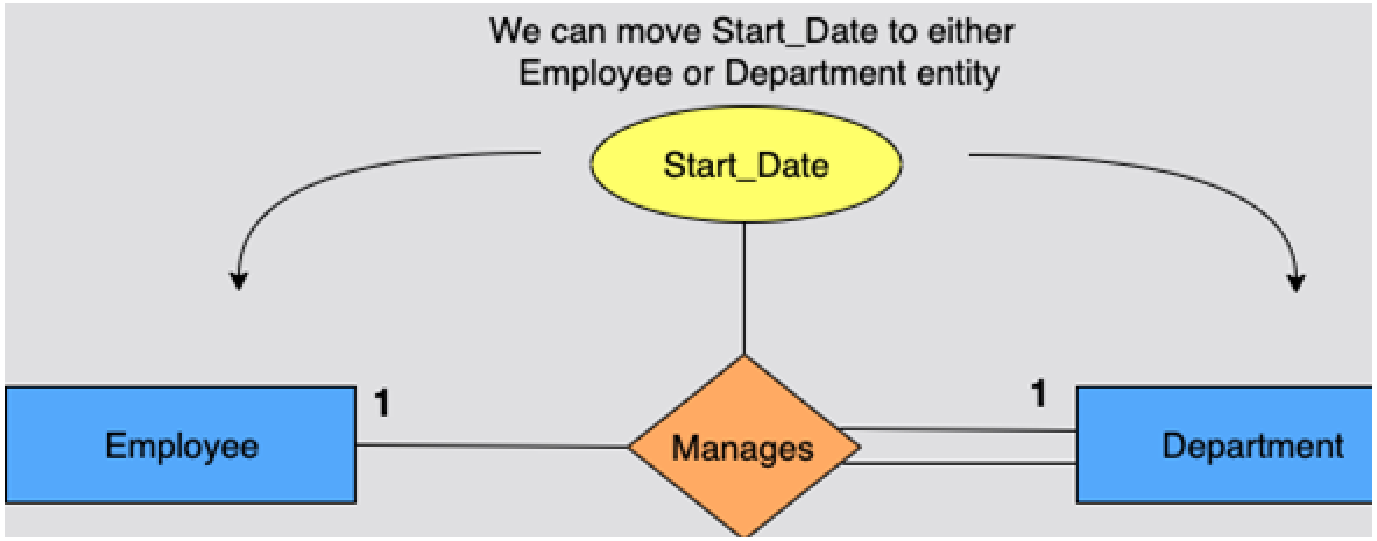
Attributes of Relationship Types

- Generally, it is not recommended to give attributes to relationships if it’s not required because when we represent this ER diagram in the database. We don’t want to create a separate table for each relationship with attributes as this will create complexity.
- So we want to move the attributes from the relationship to either of the two entities joined by the relationship (in the 1:1 and 1:N relationship).

Attributes of Relationship Types

Case 1: In the 1:1 relationship

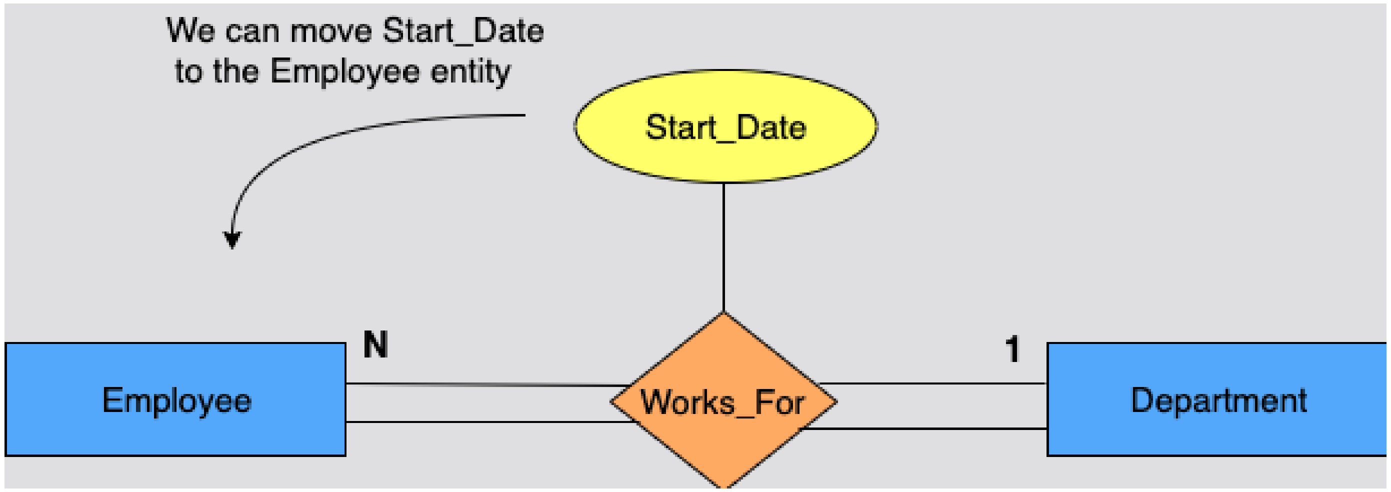
Ex: In the company database, an employee manages a department and each department is managed by an employee. Now, if we want to store the Start_Date from which the employee started managing the department.


We can move Start_Date to either Employee or Department entity
Start_Date
Employee
1
Manages
1
Department

Attributes of Relationship Types

Case 2: In the 1:N relationship

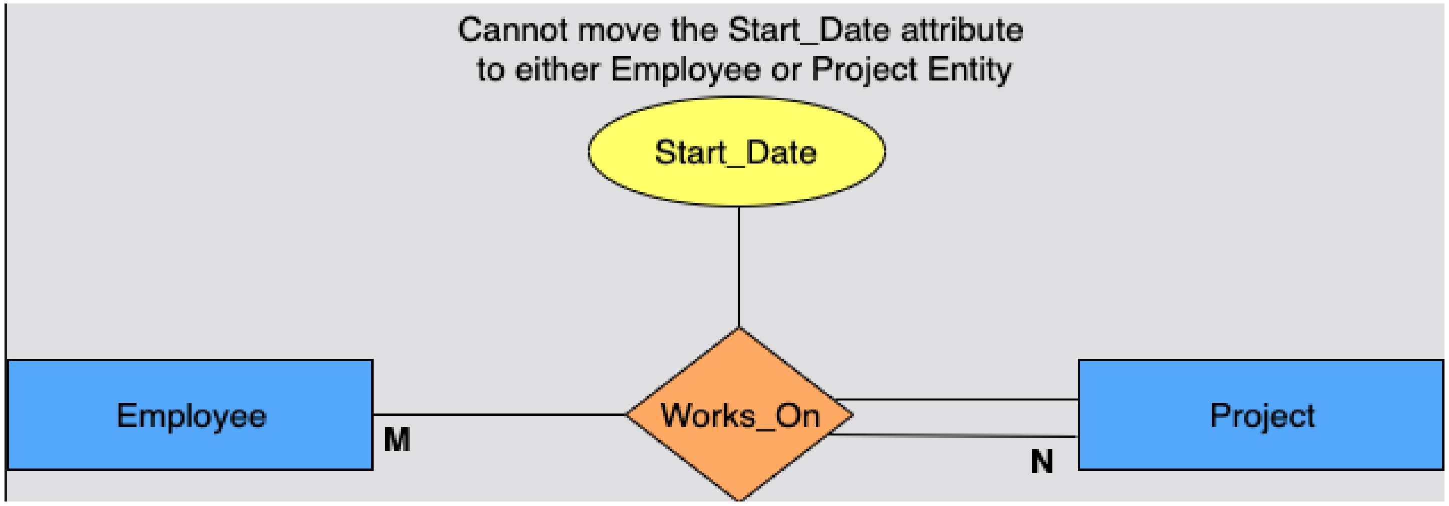
Ex: In the company database, many employees can work for a department but each employee can work only for a single department. So, there is a 1:N relationship between these entities. We want to store the Start_Date when the employee started working for the dept.



Attributes of Relationship Types

Case 3: In the M:M relationship

Ex: in the company database, an employee can work on many projects simultaneously and each project can have many employees working on it. We want to store Start_Date when the employee started working on the project.

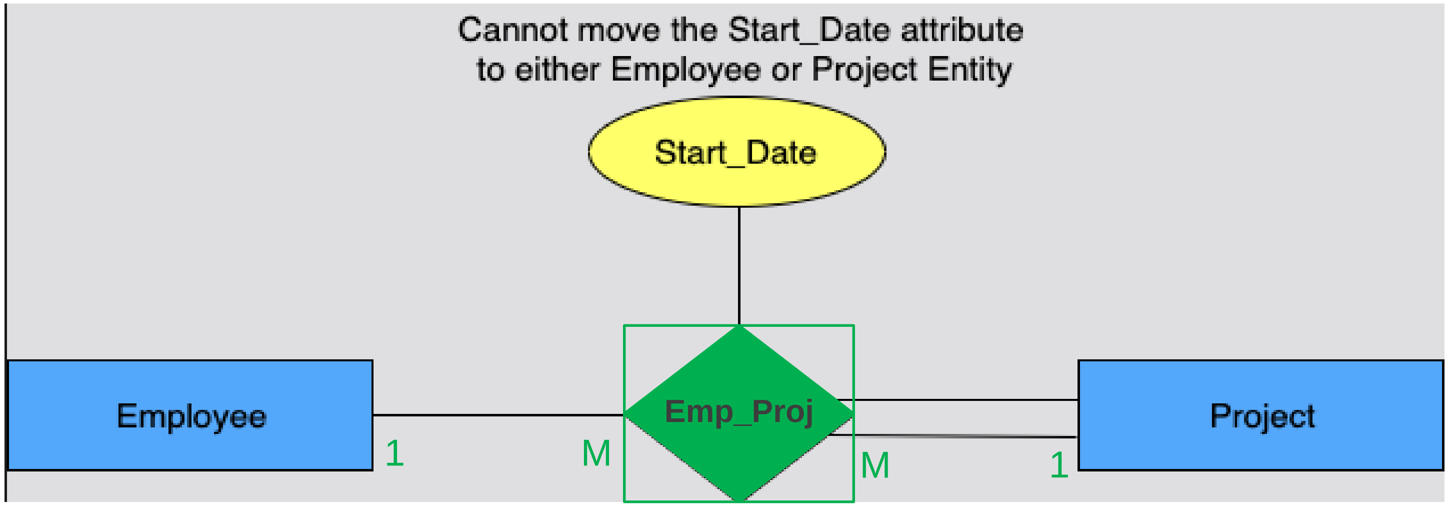


Cannot move the Start_Date attribute to either Employee or Project Entity
Start_Date
Employee
M
Works_On
N
Project

Attributes of Relationship Types

Case 3: In the M:M relationship

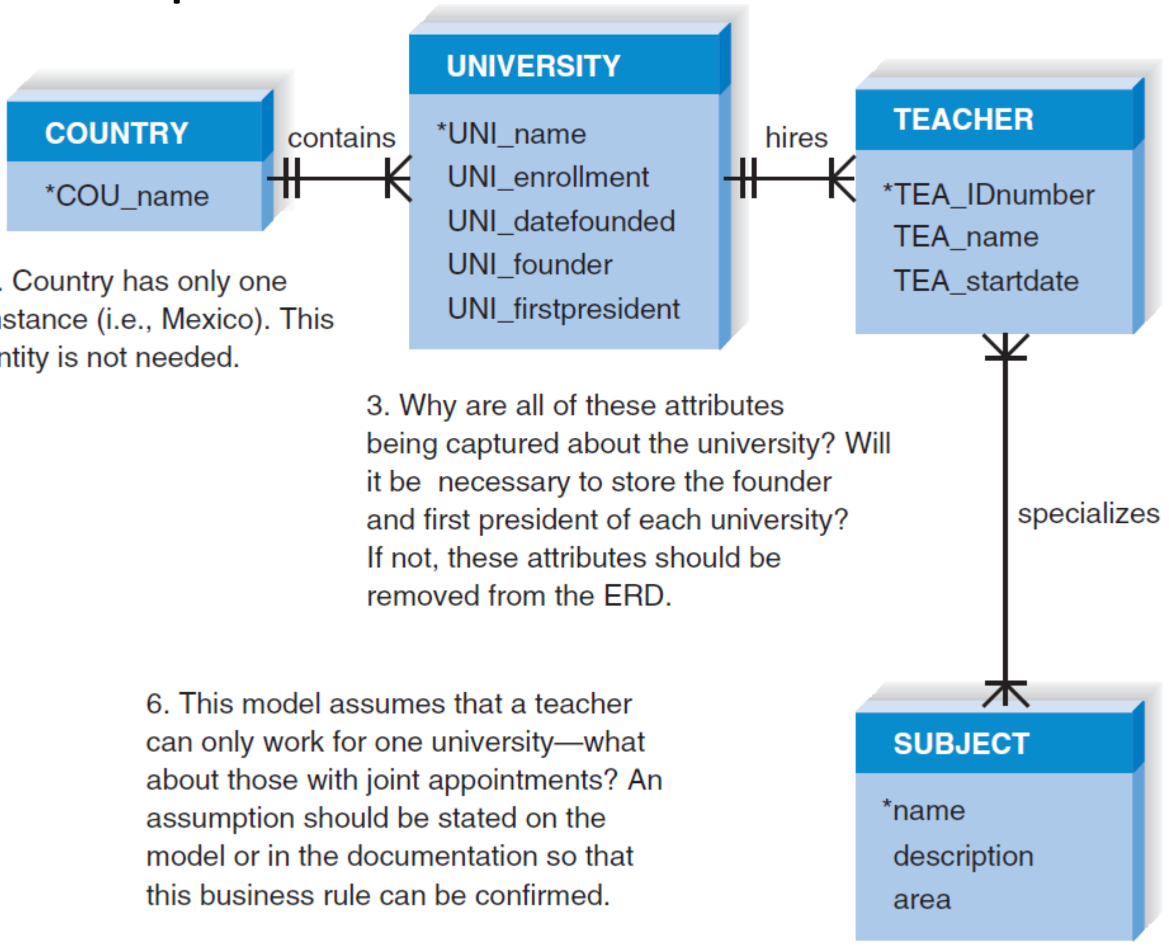
Ex: in the company database, an employee can work on many projects simultaneously and each project can have many employees working on it. We want to store Start_Date when the employee started working on the project.


Cannot move the Start_Date attribute to either Employee or Project Entity
Start_Date
Employee
1
M
Emp_Proj
M
1
Project

Evaluate Your Data Model

Example:

2. If teachers are called “Professors,” then the ERD should contain an entity called “Professor,” to remain consistent.



5. The name attribute really should be broken down into last name and first name—otherwise, there would be no way to manipulate names in the system. For example, there would be no way to sort by last name if it were combined with the first name.

1. Country has only one instance (i.e., Mexico). This entity is not needed.

3. Why are all of these attributes being captured about the university? Will it be necessary to store the founder and first president of each university? If not, these attributes should be removed from the ERD.

7. M:M is only conceptual. In logical or final design, M:M relationship must be resolved into two 1:N relationship by using an associative entity.

6. This model assumes that a teacher can only work for one university—what about those with joint appointments? An assumption should be stated on the model or in the documentation so that this business rule can be confirmed.

4. The attributes in the subject entity are poorly labeled. For one, we have no way of knowing to which entity they belong if they stood alone—it would be helpful to begin each attribute with SUB_. Also, what is “area”? A term like “department” or “field of research” would be more descriptive.

Summary

- Conceptual modeling is an iterative (repeat) process.
- To design ER, identity entity sets, attributes, relationship and cardinality by using the requirement specification and direct communication with the client. The resulting diagram should be coherent with the real world, and only needed concepts should be included in the diagram.
- Many possible solutions for a single system. There may be various valid approaches to an ER diagram. Just make sure that the ER diagram supports all the data you need to store.
- Developing an ER diagram presents several choices, whether a concept should be modeled as an attribute or as an entity set, entity or relationship... It depends on what information we want to get from the concept.

Some **best practices** for developing effective and clear ER diagrams:

1. Understand the Requirements Clearly

- Analyze the problem: Gather and understand all the requirements before starting the design.
- Identify entities and relationships: Determine what entities, attributes, and relationships exist in the system.
- Verify all assumptions about business rules so that our data model is correct. When business rules change, the relationships or other components will have to be altered.

Some **best practices** for developing effective and clear ER diagrams:

2. Use Consistent Naming Conventions

- Entity names: Use singular nouns (e.g. Patient, not Patients).
- Attribute names: Use descriptive names (e.g. AppointmentDate rather than Date).
- Avoid abbreviations: Unless they are universally understood in the context (e.g. ID for identifier).

Some **best practices** for developing effective and clear ER diagrams:

3. Minimize Redundancy

- Avoid duplicating data unless absolutely necessary.
- Use normalization techniques to prevent redundancy.
  - Ensure each piece of information is stored in only one place
  - Break down many-to-many relationships with associative entities or junction tables.

Some **best practices** for developing effective and clear ER diagrams:

4. Prioritize Clarity Over Complexity

- Use surrogate key to simplify composite primary keys
- Keep it simple: Focus on readability by organizing entities and relationships cleanly
- Avoid overcomplicating the diagram with too many attributes in the first iteration
- Add only the attributes and relationships relevant to the domain being modeled.

Some **best practices** for developing effective and clear ER diagrams:

5. Use Appropriate Cardinality

- One-to-Many (1:N) : represent with a clear line connecting the entities.
- Many-to-Many (M:N): split into two 1:N relationship using a new associative entity.
- One-to-One (1:1): verify that the relationship 1:1 is better than being captured within a single entity for some reasons.

Some **best practices** for developing effective and clear ER diagrams:

6. Identify the key attribute for each entity

- Clearly identify or add key attributes for each entity (E.g. StudentID).
- In logical and physical design, add foreign keys into related entities to enforce referential integrity.

Some **best practices** for developing effective and clear ER diagrams:

7. Plan for Data Traceability

- Add fields such as createdBy, createdAt, modifiedBy, modifiedAt, deletedBy, and deletedAt for systems needing:
  - Change tracking: who created or updated the data
  - Soft deletions: When records need to be marked as inactive instead of being permanently deleted
  - Audit logs: For compliance with industry regulations (e.g. GDPR, HIPAA).
- Consider a separate audit table for complex use cases
  - For applications requiring detailed change histories, implement a dedicated AuditLog table instead of individual entities.

Some **best practices** for developing effective and clear ER diagrams:

8. Avoid common mistakes

- Overlapping entities: Merge entities if their attributes significantly overlap
- Redundant relationships: avoid representing the same relationship in multiple ways.
- Never connect relationships to each other.

Some **best practices** for developing effective and clear ER diagrams:

9. Clearly annotate the diagram

- Add proper descriptions/labels for entities, attributes, and relationships
- Use cardinality notation to clarify the nature of relationships.
- Use visual markers to indicate weak entities or mandatory/optional relationships.

Some **best practices** for developing effective and clear ER diagrams:

9. Clearly annotate the diagram

- Add descriptions for entities, attributes, and relationships
- Use cardinality notation to clarify the nature of relationships
- Use visual markers to indicate weak entities or mandatory/optional relationships
- Use a diagramming tool like Draw.io, drawsql.app, LucidChart, dbdiagram.io, MySQL Workbench, or Microsoft Visio help create professional and clean Diagrams
- Choose tools that support notation standards like Chen, Crow's Foot, or UML.

Some **best practices** for developing effective and clear ER diagrams:

10. Validate the design

- Walk through the diagram with stakeholders to ensure it aligns with business rules.
- Test and evaluate the design by simulating queries or use cases (E.g. Which customer made this order? What if customers can order many products in one order?)

Case Study

A book shop owner wants to start up an online book shop. Here is his requirements:

"I would like my customers to be able to browse my catalog of books and place orders over the Internet. Currently, I take orders over the phone. I have mostly corporate customers who call me and give me the ISBN number of a book and a quantity; they often pay by credit card. I then prepare a shipment that contains the books they ordered. If I don't have enough copies in stock, I order additional copies and delay the shipment until the new copies arrive; I want to ship a customer's entire order together. My catalog includes all the books I sell. For each book, the catalog contains its *ISBN number, title, author, purchase price, sales price, and the year* the book was published. Most of my customers are regulars, and I have records with their *names and addresses.* New customers have to call me first and establish an account before they can use my website. On my new website, customers should first identify themselves by their unique *customer identification number*. Then they should be able to browse my catalog and to place orders online."

Design an entity relationship diagram according to the requirement of the shop owner.

Case Study

Add new requirements:

- Customers should be able to purchase several different books in a single order. If a customer wants to purchase 3 copies of English Teacher and 2 copies of The Character of Physical Law, the customer should be able to place a single order for both books.
- Shipping policies: As soon as we have enough copies of an ordered book, we ship it first, even if an order contains several books. It is possible to ship differently the books in the orders. It is possible to determine qty and ship_date for the shipped books in the order.
- Customers could place more than one order per day, and they must be able to distinguish between several orders placed the same day.

Case Study

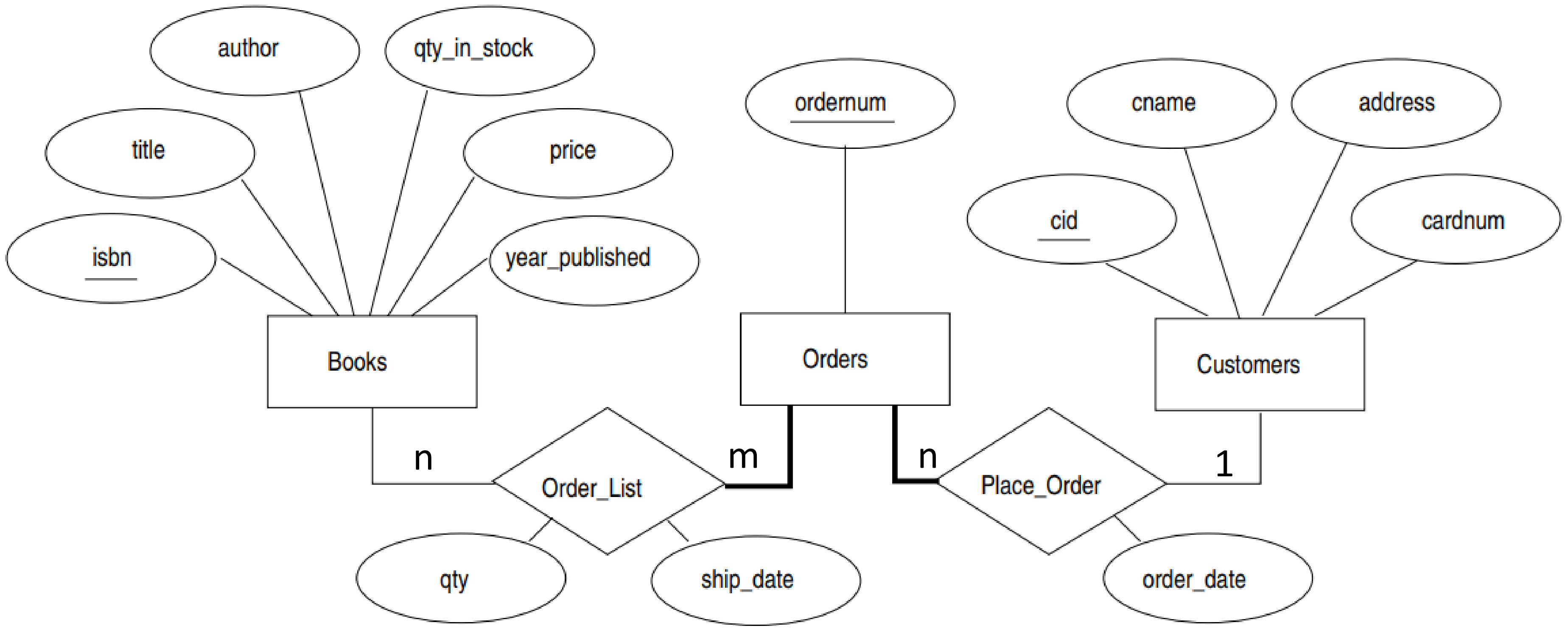


Figure - ER diagram

Assignment

- Create the conceptual design with ERD to organize the information of a company that has employees (each with Social Security Number, surname, first name and date of birth), and subsidiaries (each with code, branch and has one director, who is an employee). Each employee works for a subsidiary. Each subsidiary has many employees. Each subsidiary has only one director. One director manages only one subsidiary.