

Database Development & Design

ENTITY RELATIONSHIP (ER) MODEL

Outline

- ER Model
- Data Models in DBMS
- Design Process
- Conceptual Model
- ER Model components
- ER Notations
- DBMS Scenario

ER Model

- ▶ **An entity relationship diagram (ERD)** is a diagram that defines the structure of database instances.
- ▶ An **Entity Relationship (ER)** Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.
- ▶ ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.
- ▶ Depending on the levels of data we are modeling, we have divided data models into 3 categories:
 - Conceptual Model
 - View Model (logical model)
 - Physical model

Data Models in DBMS (schema in DBMS)

▶ Conceptual Model - High-level:

- It is high level design. What **entities** should exist and the connections between them. In this phase you recognize the entities in your model and the **relationships** between them.

▶ Logical Model:

- ▶ A logical data model describes the data in as much detail as possible, without regard to how they will be physical implemented in the database. Logical model include:
 - All entities and relationships among them.
 - Resolves M:N relationships.
 - All attributes for each entity are specified.
 - The primary and foreign keys for each entity is specified.
 - Create normalized business rules.

▶ Physical Model - Low-level:

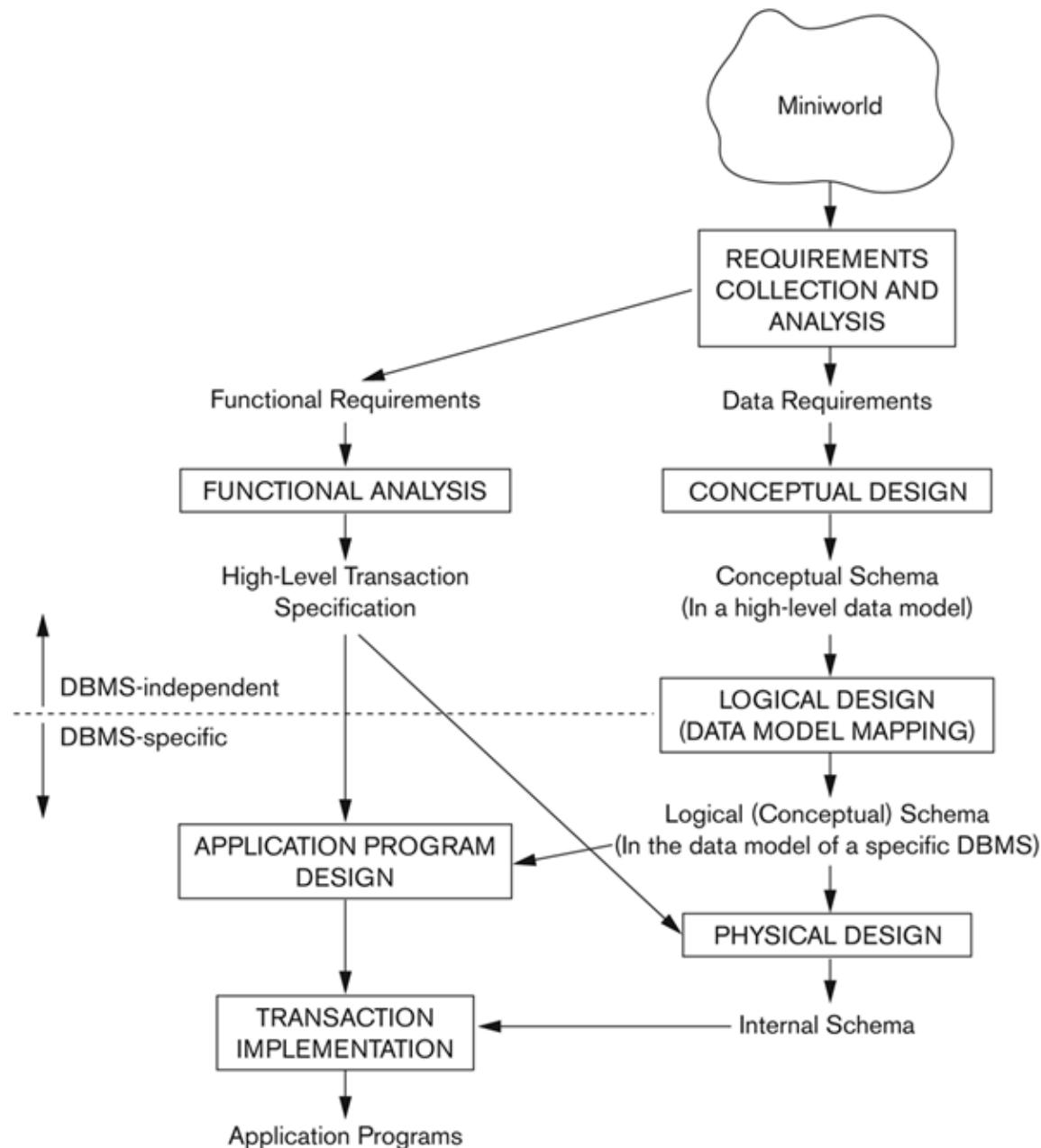
- Represents how the model will be built in the database.
- Shows all entity structures, attribute data type, attribute constraints, primary key, foreign key, and relationships.

Data Models in DBMS (schema in DBMS)

The three levels of data modeling (**conceptual**, **logical**, **physical**) were discussed in prior slide. The table below compares the different features:

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	✓
Entity Relationships	✓	✓	✓
Attributes		✓	✓
Primary Keys		✓	✓
Foreign Keys		✓	✓
Attribute constraints			✓
Attribute Data Types			✓

Overview of Database Design Process



Database Design Process

Main phases of database design:

1. Requirements collection and analysis:

- Database designers interview prospective database users to understand and document data requirements
- Result:
 - **Data requirements.**
 - **Functional requirements** of the application consist of the user defined operations (or transactions) that will be applied to the database, including both retrievals and updates.

Database Design Process

2. Creating a Conceptual schema for the database, using a high-level conceptual data model.

- This step is called: **Conceptual design**
- Description of data requirements:
 - Includes detailed descriptions of the entity types, relationships, and constraints
 - Because these concepts do not include implementation details, they are usually easier to understand and can be used to communicate with nontechnical users.
- The high-level conceptual schema can also be used as a reference to ensure that all users' data requirements are met and that the requirements do not conflict.
- This approach enables database designers to concentrate on specifying the properties of the data, without being concerned with storage and implementation details.
- This makes it is easier to create a **good conceptual database design**.

Database Design Process

3. Logical design or data model mapping:

- The next step in database design is the actual implementation of the database, using a commercial DBMS. Most current commercial DBMSs use an implementation data model—such as **the relational** or the object-relational database model.
- In this step the conceptual schema is transformed from the high-level data model into the implementation data model.
- Result is: a **database schema** in implementation data model of DBMS

Database Design Process

4. Physical design phase

- Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files specified
- In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

Build the Conceptual Schema

To build the conceptual schema for any database requirements using the ER model, **we need to identify four main concepts (elements)**

- The set of possible **entities**
- The set of possible **attributes** for each entity
- The set of possible **relationships** between entities
- The possible **constraints** for each relationship

ER model in DBMS

► Entity:

- Represents a real-world object or concept that are represented in the database.
- **Rectangles** are used to represent the entity in the diagram. Name of the Entity is written inside the rectangle.
- An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) Or it may be an object with a conceptual existence (for instance, a company, a job, or a university course)

ER model in DBMS

- ▶ **Attribute:** attributes are the describing characteristics or properties that define all items pertaining to a certain category applied to all cells of a column.
 - ▶ **Simple attribute:** are atomic values, which cannot be divided further; For example, SSN or Sex.
 - ▶ **Composite attribute:** are made of more than one simple attribute; For the example the Address attribute of the EMPLOYEE entity can be subdivided into (Street_address, City, State, and Zip)
 - ▶ **Derived attribute:** are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, the value of Age can be determined from the current (today's) date and the value of that person's Birth_date.

ER model in DBMS

- ▶ **Multi-value attribute:** Multi-value attributes may contain more than one values. For the example a College_degrees attribute for a person: one person may not have a college degree, another person may have one, and a third person may have two or more degrees.
- ▶ **Complex attribute:** multi-value attribute and composite attribute.
- ▶ **Primary Key attribute :** An underline to the attribute name is put to represent the primary key.
- ▶ **Relationship attribute:** a relation can have attribute. Most relationship attributes are used with M:N relationships.

ER model in DBMS

- **A key attribute may be composite.**
 - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- **An entity type may have more than one key.**
 - The CAR entity type may have two keys:
 - VehicleIdentificationNumber (popularly called VIN)
 - VehicleTagNumber (Number, State), aka license plate number.

ER model in DBMS

➤ Relationship:

- A **relationship** relates two or more distinct entities with a specific meaning.
 - For example, EMPLOYEE John Smith works on the ProductX PROJECT, or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a **relationship type**.
 - For example, the WORKS ON relationship type in which EMPLOYEES and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEES and DEPARTMENTS participate.

Relationship Type vs. Relationship Set

Relationship Type (role name):

- Is the schema description of a relationship
- Identifies the relationship name and the participating entity types
- Also identifies certain relationship constraints

Relationship Set:

- The current set of relationship instances represented in the database
- The current *state* of a relationship type

Each instance in the set relates individual participating entities – one from each participating entity type

E-R model in DBMS

➤ **Cardinality:**

- cardinality specifies how the number of occurrences of one object are related to the number of occurrences of another object (1:1, 1:N, M:N).

Constraints on Relationships Types

- **Cardinality (*maximum*)**: Specifies maximum number of relationship instances that entity can participate in:
 - One
 - Many
- **Participation (*minimum*)**: Specifies whether existence of entity depends on its being related to another entity via the relationship type
 - zero (optional participation, not existence-dependent, partial)
 - One (mandatory participation, existence-dependent, total)

Recursive Relationship Type

- A relationship type between the same participating entity type in **distinct roles**
- Also called a **self-referencing** relationship type.
- Example: the SUPERVISES relationship
- Example: Each relationship instance relates two distinct EMPLOYEE entities:
 - One employee in supervisor role
 - One employee in supervisee role

Entity vs. Entity Type vs. Entity Set

► Entity type:

- Entities with the same basic properties are grouped or typed into an entity type (the category of a particular entity).
- For example, the entity type employee, department, project.

► Entity:

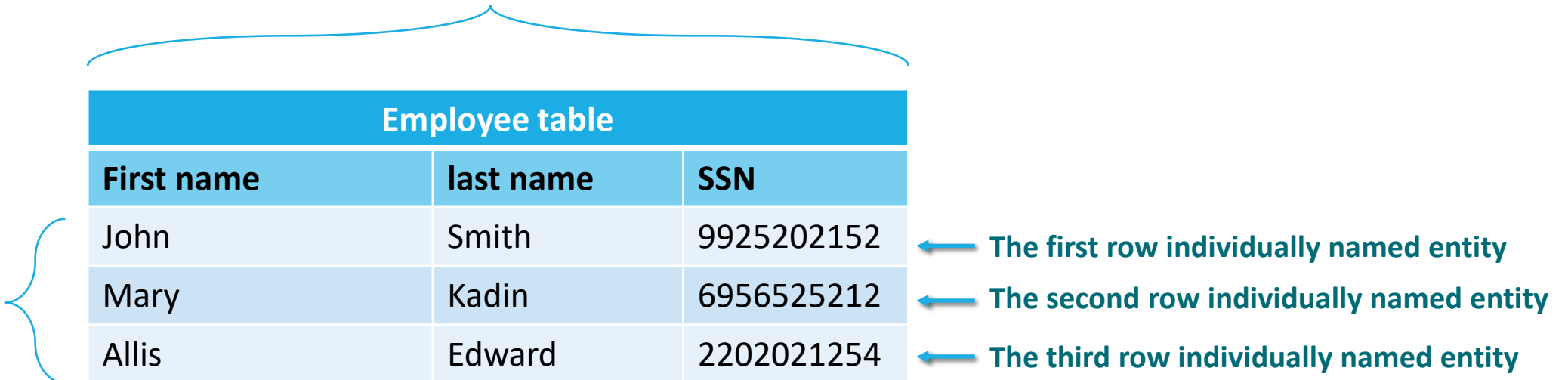
- Entity is a basic concept for the ER model. Entities are specific things or objects in the mini-world that are represented in the database.
- For example, the employee John Smith, the Research department, the ProductX project.

► Entity Set:

- Each entity type will have a collection of entities stored in the database.
- Entity set is the current state of the entities of that type that are stored in the database.

Entity vs. Entity Type vs. Entity Set

The whole table called entity type



Employee table		
First name	last name	SSN
John	Smith	9925202152
Mary	Kadin	6956525212
Allis	Edward	2202021254

The three rows
together named
entity set

← The first row individually named entity

← The second row individually named entity

← The third row individually named entity

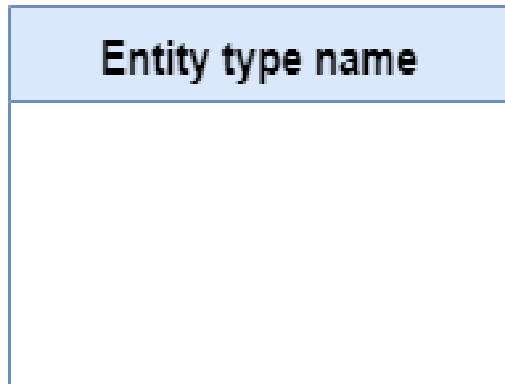
ERD Notations in Data Modeling

Different Notations

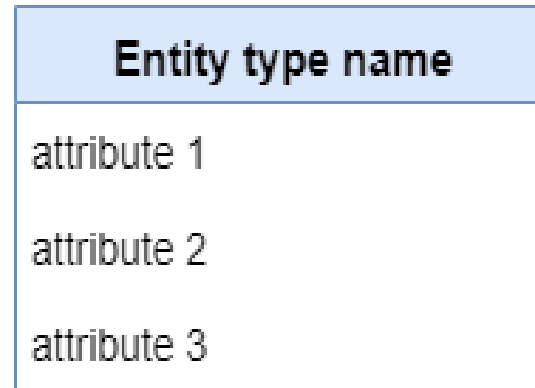
- Barker's Notation
- Chen Notation
- IDEF1X Notation
- Arrow Notation
- UML Notation
- **Crow's Foot Notation; we will use in this course.**

Notations for ER Diagrams

► Entity type:



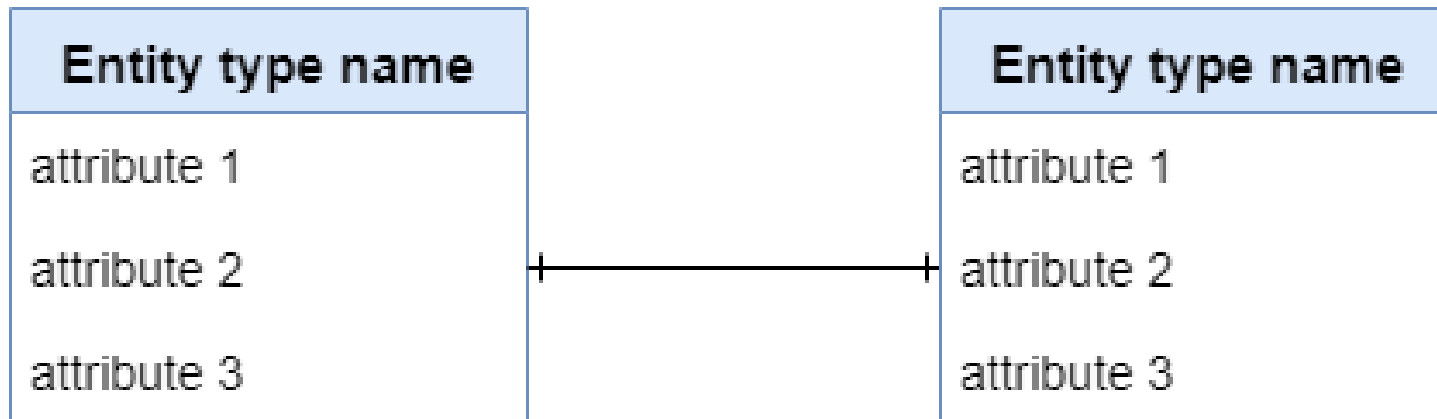
► Attributes



Notations for ER Diagrams

Relationship:

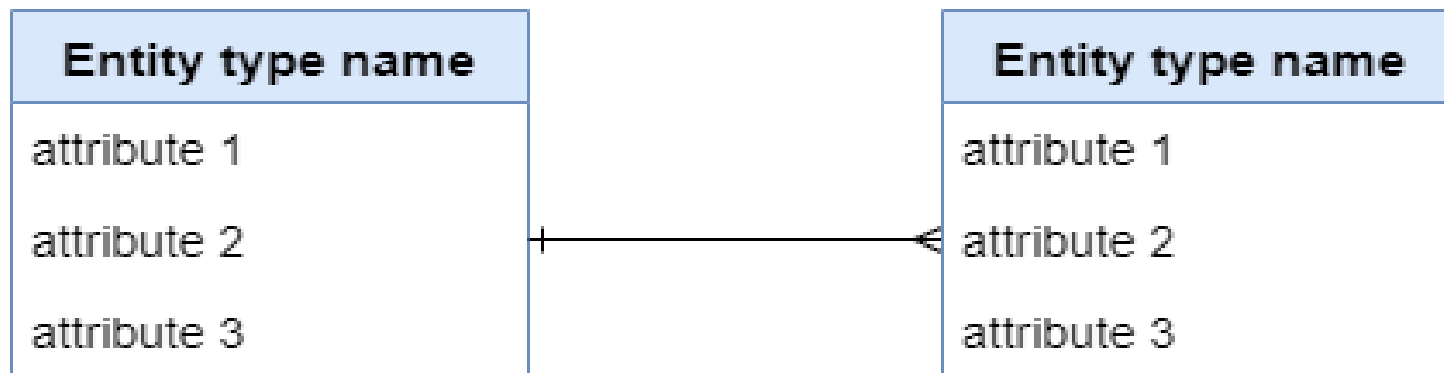
- One-to-one relation (1:1)



ERD Notations

► Relationship:

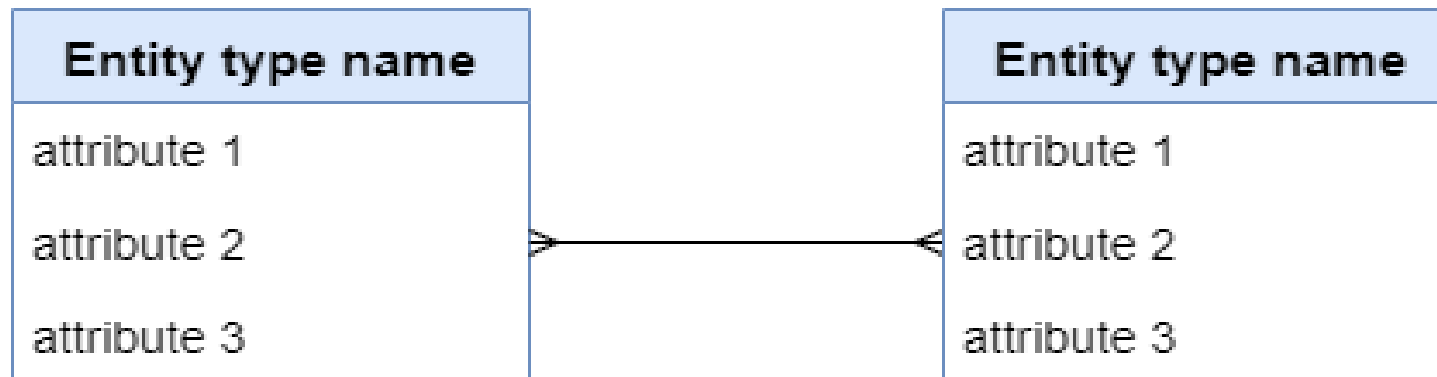
- One-to-many relation (1:M)



ERD Notations

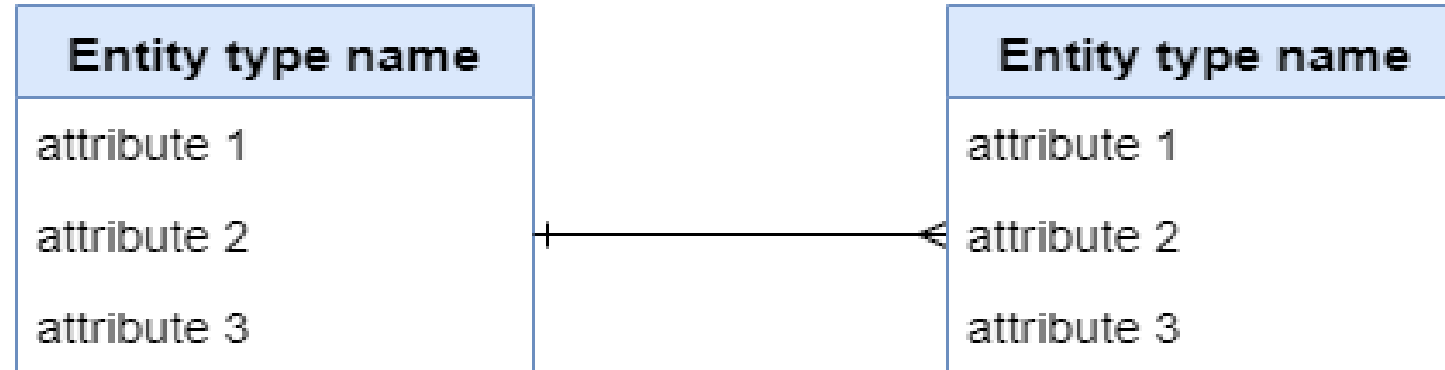
► Relationship:

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

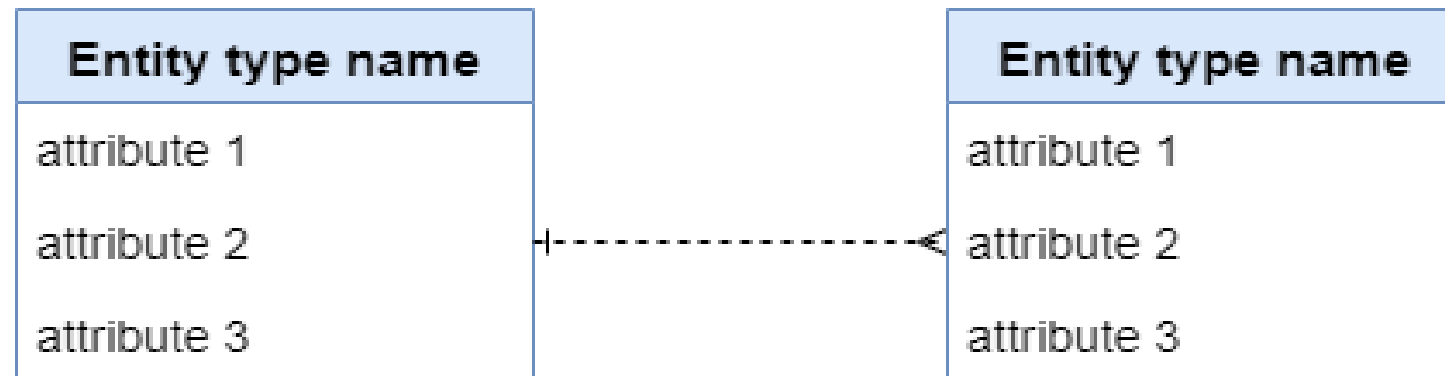


Shapes of E-R Diagram

► Weak relationship



► Strong relationship



Shapes of E-R Diagram

Optional relationship

The symbol indicating whether the relationship is mandatory or optional.



Customer may have many sales. It is permissible to have a customer with no associated sale records.



Bank Management System Example

Scenario:

The bank has many branches. Each branch consists of multiple departments, and each department includes many employees. The employee should work in one department. An employee can make many transactions. Each transaction is performed on one account. Note: the account may have multiple transactions. In addition, the account is owned by one or more customers and customers can have multiple accounts. There are many account types like (saving, checking, and money market accounts). For each account, there is one account type. Customers can visit multiple branches.

Bank Management System Example

We should follow the below steps to be done with the ERD:

- 1. Identify the entities**
- 2. Identify the relationships and cardinalities**
- 3. Conceptual design**
4. Identify the attributes
5. Schema and mapping
6. Normalization
7. Logical design
8. Physical design

Bank Management System Example

Scenario:

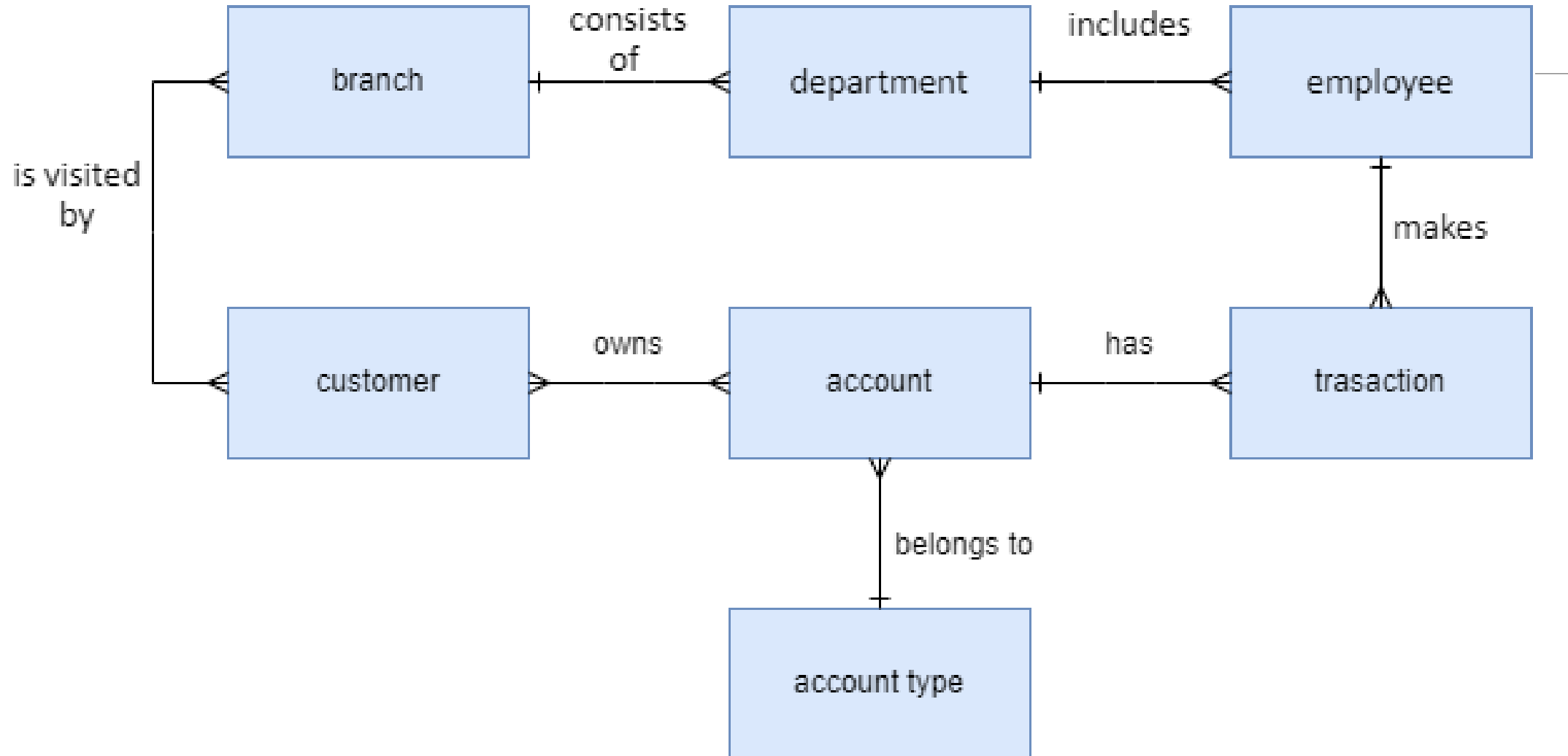
The bank has many **branches**. Each branch consists of multiple **departments**, and each department includes many **employees**. The employee should work in one department. An employee can make many **transactions**. Each transaction is performed on one **account**. Note: the account may have multiple transactions. In addition, the account is owned by one or more **customers** and customers can have multiple accounts. There are many **account types** like (saving, checking, and money market accounts). For each account, there is one account type. Customers can visit multiple branches.

Bank Management System Example

Entities:

1. Branch
2. Employee
3. Department
4. Transaction
5. Account
6. Customer
7. Account type

Conceptual Model



Company Example

Scenario:

The company is organized into departments. Each department has an employee who manages the department. Each department controls a number of projects. Each employee works for one department but may work on several projects. Each employee may have a number of dependents.

Company Example

Scenario:

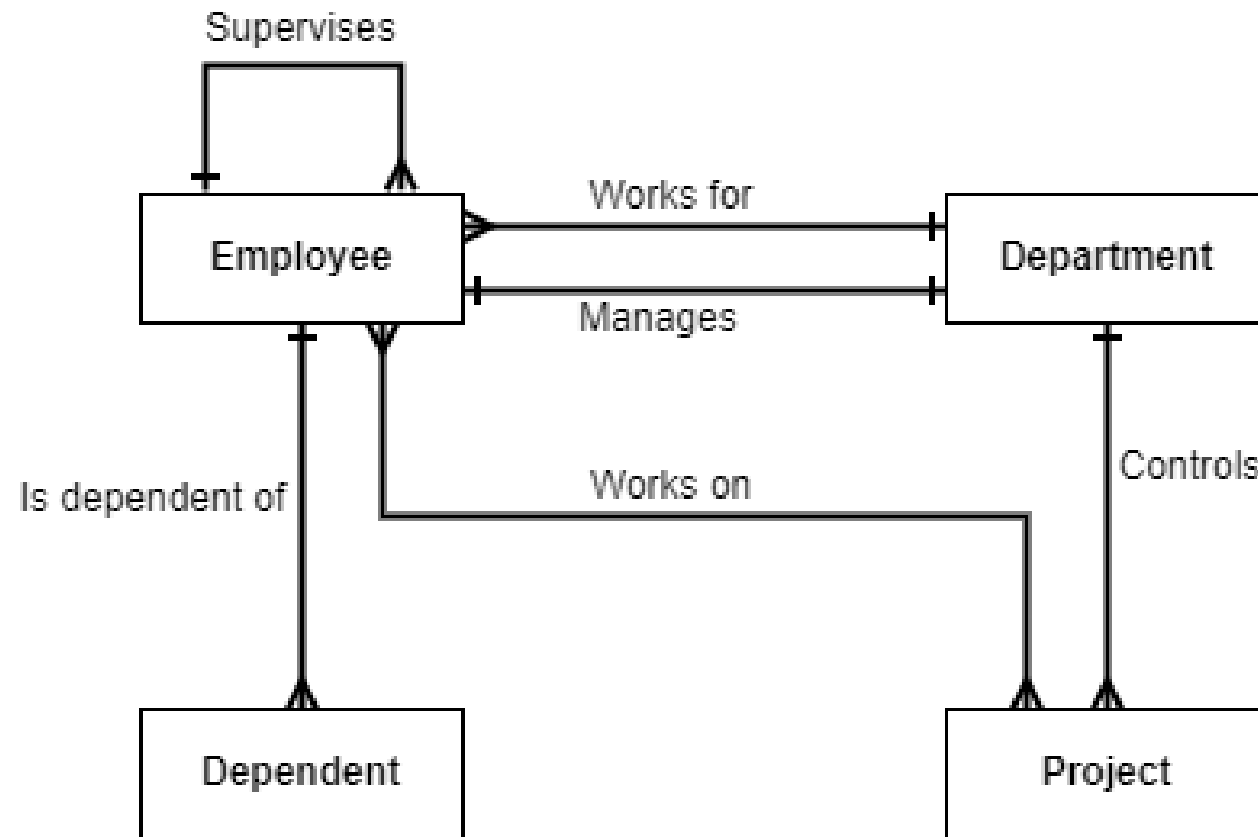
The company is organized into **departments**. Each department has an **employee** who manages the department. Each department controls a number of **projects**. Each employee works for one department but may work on several projects. Each employee may have a number of **dependents**.

Company Example

Entities:

1. Department
2. Employee
3. Project
4. Dependent

Conceptual Model



References

- Elmasri, R., & Navathe, S. (2017). Fundamentals of database systems (Vol. 7). Pearson
- Eng. Lina's slides
- Dr. Raneem's slides.