

Entity-Relationship Modelling

With Real-World Examples

An enhanced exploration of ERM concepts, industry use-cases, and practical application in modern database design



1. ● Introduction to ERM
2. ● What is Entity-Relationship Modelling?
3. ● Why Use ERM? (Academic & Real-World)
4. ● Core Concepts Overview
5. ● Entities: Definition & Types
6. ● Real-World Entity Examples
7. ● Attributes: Types & Illustration
8. ● Real-World Attribute Examples
9. ● Relationships: Definition & Types
10. ● Real-World Relationship Examples
11. ● ER Diagrams: Visual Notation
12. ● Example: University Student Registration
13. ● Case Study: E-Commerce Management
14. ● Case Study: Healthcare Patient Management
15. ● Advanced Concepts: Multiplicity & Cardinality
16. ● Practice Exercise: Banking System
17. ● Key Takeaways & Further Reading

Definition of Entity-Relationship Modelling (ERM), origin, and why it is vital in data-centric organizations.

- ERM is a **conceptual data modeling technique** used to represent data requirements in an organization
- Developed by Peter Chen in 1976 to create unified view of data
- Serves as a **blueprint** for designing databases regardless of the DBMS used
- Focuses on **entities, relationships, and attributes** relevant to the business domain

Key Benefits:

- Bridges communication between business stakeholders and technical teams
- Identifies and resolves data requirements before implementation
- Provides foundation for subsequent database design stages

Visual Element Placeholder



Figure: Description of the visual element

Real-World Application

Example context or case study reference can be included here to provide practical relevance to the theoretical concepts.



What is Entity-Relationship Modelling?

The Entity-Relationship Model is a conceptual schema used to represent the data structure of an organization, independent of how it will be stored in a database.

- **Conceptual Model:** Represents business requirements in terms entities and their relationships
- **Technology Independent:** Designed regardless of specific database technology (Oracle, MySQL, MongoDB)
- **Semantic Modeling:** Capitalizes on knowledge of data meaning to inform database structure
- **Transformation Process:** Later translated into a logical model (e.g., relational) for implementation

Industry Application:

- **E-commerce:** Amazon uses ERMs to model relationships between customers, orders, products, and sellers
- **Banking:** Financial institutions model account holders, transactions, loans, and credit cards
- **Healthcare:** Hospital systems represent patients, doctors, appointments, and medical records

Database Development Lifecycle

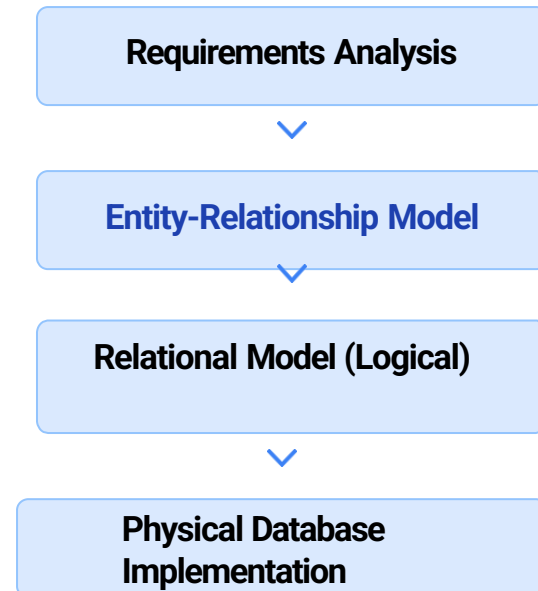


Figure: ERM in the Database Development Process

Real-World Example: Spotify

Spotify uses ERM to model the complex relationships between users, artists, songs, playlists, and albums. This conceptual model helps their engineering team design efficient database structures that support millions of concurrent users.

Why Use ERM? (Academic & Real-World Importance)

Entity-Relationship Modelling serves as a vital tool for both academic understanding and practical database implementation in organizations:

- **Communication tool:** Bridges the gap between business stakeholders and technical teams
- **Technology-independent:** Models concepts regardless of DBMS, focusing on business requirements
- **Error prevention:** Identifies data integrity issues before implementation, reducing costly redesign
- **Complexity management:** Breaks down complex systems into understandable components

Business Impact:

- Reduces data redundancy and inconsistency in operational systems
- Improves application performance through optimized database design
- Facilitates scalable systems that can adapt to changing requirements
- Provides documentation for future maintenance and enhancements

ERM Impact Across Industries



Amazon's product catalog system uses ER modeling to handle millions of products, categories, and relationships



Electronic Health Record systems use ERM to model complex patient-provider relationships and medical histories



Financial institutions use ERM to design systems tracking accounts, transactions, and customer relationships

Real-World Case: Netflix

Netflix uses sophisticated ER models to manage relationships between users, viewing histories, content categories, and recommendations. This data model enables personalized content suggestions and improves user engagement.

The Entity-Relationship Model consists of three core components that form the foundation of database design:

- **Entities:** Real-world objects or concepts that can be distinctly identified
Example: In e-commerce, entities include *Customer*, *Product*, and *Order*
- **Relationships:** Associations between entities that represent how they interact
Example: *Customer places Order*, *Order contains Product*
- **Attributes:** Properties that describe entities or relationships
Example: Customer has *name*, *email*, *address*; Product has *price*, *stock level*

Learning Objectives:

- Identify entities, relationships, and attributes in business scenarios
- Understand relationship constraints (cardinality, participation)
- Create Entity-Relationship diagrams using standard notation
- Transform conceptual models into logical database designs

Basic ERM Concepts Visualization

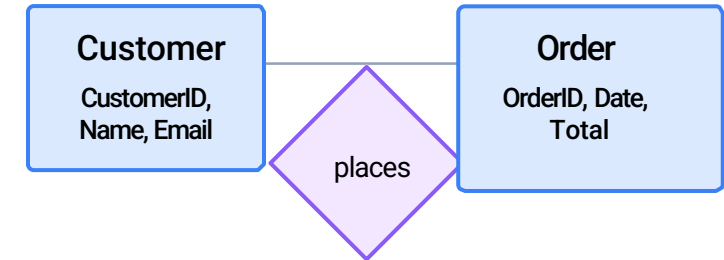


Figure: Simple E-commerce Entity-Relationship example

Real-World Application

Healthcare System Example: A hospital needs to track patients, doctors, and appointments. Each patient can schedule multiple appointments with different doctors, and doctors can see many patients. Patient and doctor details need to be stored along with appointment information.

What is an Entity?

An entity is a distinguishable object or concept in the real world that is important to the organization and needs to be represented in the database.

- **Strong (Regular) Entity:** Has an independent existence and can be uniquely identified by its own attributes
- **Weak Entity:** Depends on another entity for its existence and cannot be uniquely identified by its own attributes alone

Notation Guidelines

- Entity type name uses singular form (e.g., Customer, not Customers)
- Capitalize first letter of each word in multi-word names (e.g., OrderItem)
- Strong entities drawn with solid rectangles; weak entities with double or dashed lines
- Each entity will have attributes (covered in upcoming slides)

Entity Types & Examples

Strong Entities

Customer

Product

Employee

Can exist independently

Weak Entities

OrderItem

Dependent

PaymentInstallment

Cannot exist without owner entity

Real-World Application

In an e-commerce platform like Amazon, **Customer** and **Product** are strong entities that can exist independently, while **Review** is a weak entity that cannot exist without both a customer who wrote it and a product it's about.



Healthcare System Entities

Healthcare information systems track multiple key entities:

- **Patient** - Person receiving medical care
- **Doctor** - Healthcare provider with specialization
- **Appointment** - Scheduled meeting between patient and doctor
- **Medical Record** - History of diagnoses, treatments, etc.

Example: Patient Entity

Patient
PatientID PK
FirstName
LastName
DateOfBirth
ContactNumber
InsuranceDetails

In a hospital management system, each Patient entity has a unique ID and stores personal information essential for treatment and administration.

E-Commerce System Entities

Online shopping platforms require these core entities:

- **Product** - Items available for purchase
- **Customer** - Registered user who can place orders
- **Order** - Transaction initiated by customer
- **Payment** - Financial transaction details

Example: Product Entity

Product
ProductID PK
Name
Description
Price
CategoryID
StockQuantity

In an e-commerce platform like Amazon, each Product entity contains information needed for display, inventory management, and sales.

Attributes describe the properties of entities and relationships in an ERM. They represent the data we want to store.

Simple/Atomic Attributes

Cannot be divided further (e.g., date of birth, product ID)

Composite Attributes

Can be divided into smaller parts (e.g., address → street, city, zipcode)

Multi-valued Attributes

Can have multiple values (e.g., phone numbers, skills)

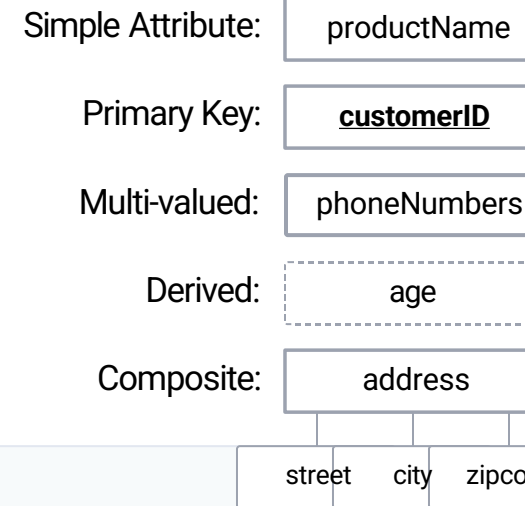
Derived Attributes

Value calculated from other attributes (e.g., age from DOB, total price)

Key Attributes

Uniquely identify an entity instance (e.g., SSN, employee ID)

ER Notation for Attributes



Real-World Example

- **E-commerce Customer Entity:** customerID (key), name (simple), address (composite), phone_numbers (multi-valued), loyalty_points (simple), member_years (derived)
- **Healthcare Patient Entity:** patientID (key), full_name (composite), allergies (multi-valued), date_of_birth (simple), age (derived)



Examining how attributes describe entities in real-world systems across different industries:

Healthcare Example: Patient Entity

- **Simple attributes:** PatientID, FirstName, LastName, Gender
- **Composite attribute:** Address (Street, City, State, ZIP)
- **Multi-valued attribute:** PhoneNumbers [Home, Mobile, Emergency]
- **Derived attribute:** Age (calculated from DateOfBirth)

E-commerce Example: Product Entity

- **Simple attributes:** ProductID, Name, Price, Weight
- **Composite attribute:** Dimensions (Length, Width, Height)
- **Multi-valued attribute:** Categories [Electronics, Accessories, etc.]
- **Derived attribute:** StockValue (calculated from Price × Quantity)

Healthcare vs. E-commerce Attributes

Industry	Entity	Key Attributes	Type
Healthcare	Doctor	DoctorID, Name, Specialty	Simple
Healthcare	Medical Record	Diagnoses [multiple]	Multi-valued
Healthcare	Appointment	Duration (EndTime-StartTime)	Derived
E-commerce	Customer	CustomerID, Name, Email	Simple
E-commerce	Order	ShippingAddress (composite)	Composite
E-commerce	Shopping Cart	TotalAmount (sum of items)	Derived

Real-World Application

Amazon's product database demonstrates all attribute types: simple (price), composite (dimensions), multi-valued (categories), and derived (average rating). These attributes enable search filtering, recommendations, and inventory

A relationship is an association between two or more entities that defines how these entities interact with each other in the real world.

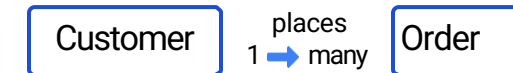
- **Binary relationship:** Between two entity types (most common)
Example: Customer → places → Order
- **Unary/Recursive relationship:** Entity relates to itself
Example: Employee → supervises → Employee
- **Ternary relationship:** Between three entity types
Example: Doctor → prescribes → Medication → to → Patient

Relationship Constraints:

- **Cardinality:** The number of instances of an entity that can be associated with instances of another entity
One-to-One (1:1), One-to-Many (1:M), Many-to-Many (M:N)
- **Participation:** Whether all or only some entity instances participate in a relationship
Total/Mandatory vs. Partial/Optional

Common Relationship Types

Binary (1:M)



Recursive



Figure: Visual representation of relationship types


Real-World Application

In a healthcare system, a **Doctor** can treat **many Patients** (1:M), and a **Patient** can be treated by **many Doctors** (M:1), forming a **many-to-many** relationship. The relationship has attributes like "appointment date" and "diagnosis".




Healthcare Examples

- **Doctor treats Patient** (1:M) - One doctor treats many patients; each patient has one primary doctor
- **Patient schedules Appointment** (1:M) - One patient can schedule multiple appointments over time
- **Nurse assists Doctor** (M:M) - Nurses assist multiple doctors; doctors work with multiple nurses

 **Real Impact:** These relationships allow hospitals to track patient history, manage staff scheduling, and ensure proper patient care coordination.

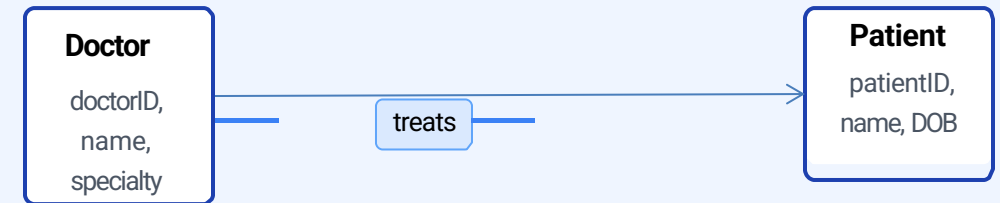
E-Commerce Examples

- **Customer places Order** (1:M) - A customer can place many orders; each order belongs to one customer
- **Order contains Product** (M:M) - An order contains multiple products; a product appears in many orders
- **Product belongs to Category** (M:1) - Many products belong to one category

 **Real Impact:** Amazon uses these relationship models to power product recommendations, shopping cart functionality, and inventory management.

Simplified ER Diagrams

Healthcare Relationship Example



E-Commerce Relationship Example

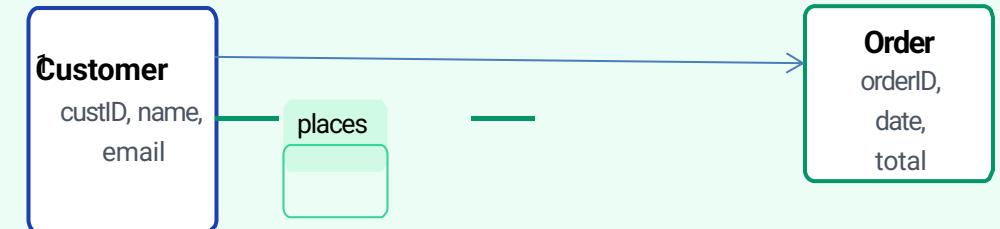







Figure: Binary relationships with different cardinalities in real-world contexts

Implementation Note

In database implementation, relationships translate to foreign keys (1:M), junction tables (M:M), or direct references (1:1). For example, the Patient table would include a doctorID foreign key referencing the Doctor table.

ER diagrams use standardized symbols to visually represent database structure, making it easier to understand complex data relationships.

Core Notation Elements:

- **Entity**
Rectangle represents a person, place, object, event or concept
- **Relationship**
Diamond indicates how entities interact with each other
- **Attribute**
Oval represents a property or characteristic of an entity
- **Weak Entity**
Double-outlined rectangle; depends on another entity for identification
- **Key Attribute**
Underlined oval; uniquely identifies an entity instance

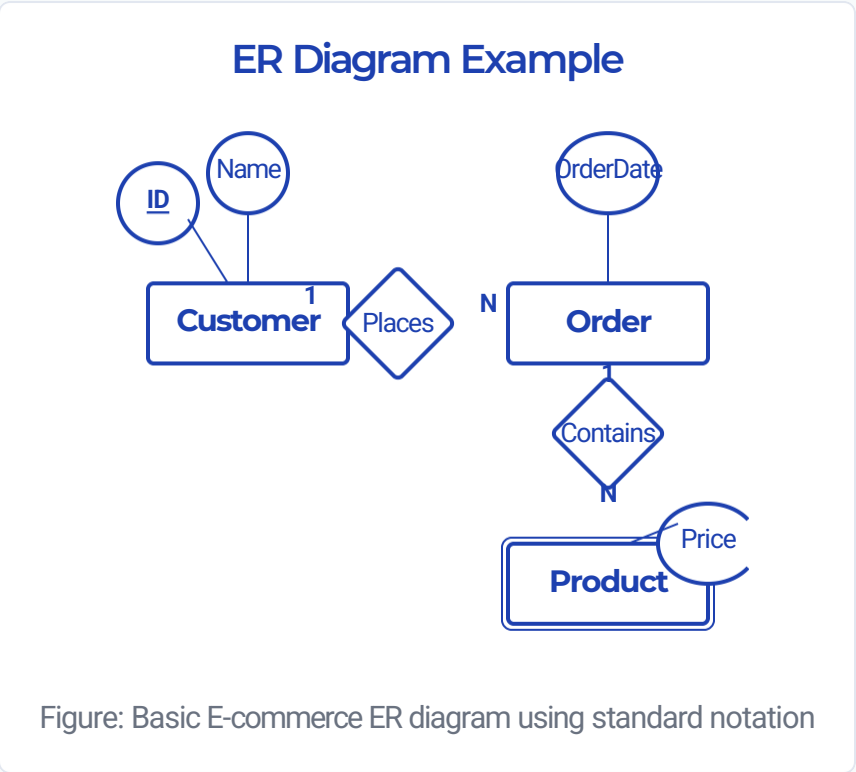


Figure: Basic E-commerce ER diagram using standard notation

Industry Standard

While different tools may use slightly different visual representations, the core concepts remain the same across industries. E-commerce companies like Amazon use ER diagrams with these standard notations as foundation for their complex database systems.

Example: University Student Registration System

A university registration system represents a common real-world application of ER modeling:

- **Students** register for multiple courses each semester
- Each **Course** belongs to a specific department
- **Instructors** teach courses and assign grades
- **Departments** offer courses and employ instructors

Key Relationships:

- Student **registers for** Course (M:N)
- Course **belongs to** Department (N:1)
- Instructor **teaches** Course (1:N)
- Registration has **attributes**: semester, year, grade

University Registration System ER Diagram

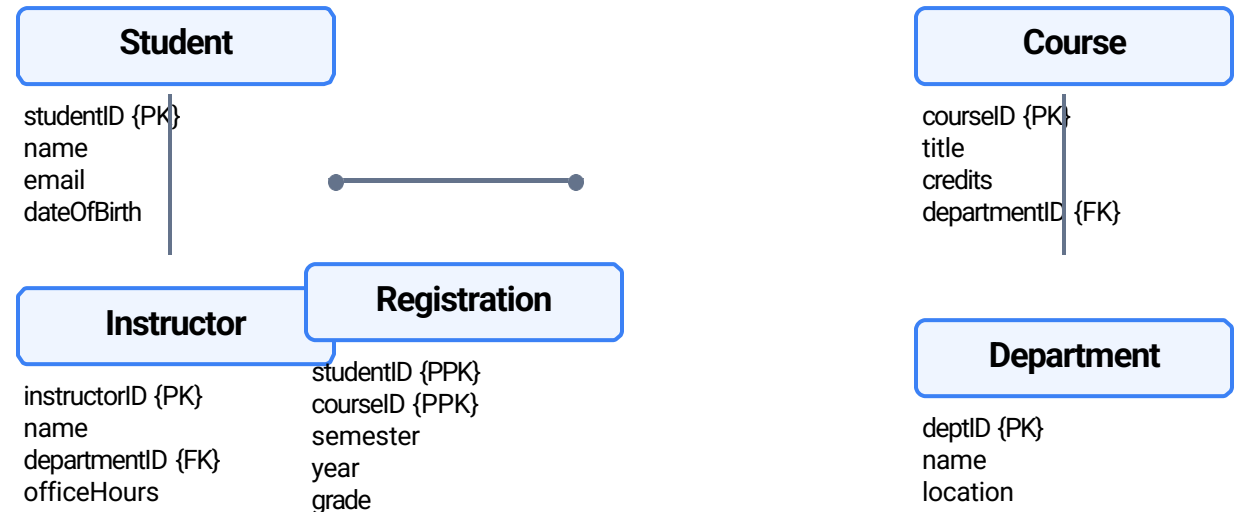


Figure: University Student Registration System ER Diagram

Real-World Application

This model forms the foundation for systems like Canvas, Blackboard, and university portals where students register for courses, instructors manage class rosters, and administrators track department offerings.

Case Study: E-Commerce Inventory Management

This case study demonstrates how ERM helps design a robust database for an e-commerce platform's inventory management system.

- **Key Entities:** Product, Category, Supplier, Order, Customer, Warehouse, Inventory
- **Critical Relationships:** Products are supplied by Suppliers; Products belong to Categories; Orders contain Products; Inventory tracks Products in Warehouses
- **Important Attributes:** Product (SKU, name, price, description); Supplier (ID, name, contact); Inventory (quantity, reorder level)
- **Business Rules:** One product may belong to multiple categories; One order contains multiple products; Each product has stock levels tracked across multiple warehouses

Real-World Benefits:

- Optimized inventory levels reducing storage costs by 23% at ACME Online Retail
- Streamlined order fulfillment decreasing processing time from 3 days to 4 hours
- Enhanced supplier management enabling better negotiation and reduced costs
- Improved reporting capabilities for sales trends and inventory forecasting

E-Commerce Inventory ER Diagram (Simplified)

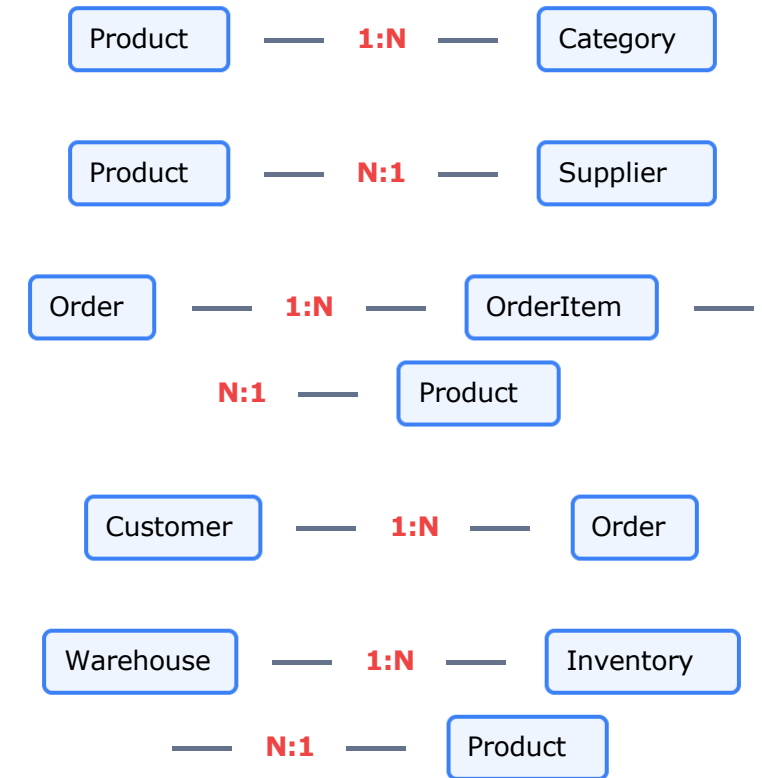


Figure: Simplified ER diagram for an e-commerce inventory management system

Case Study: Healthcare Patient Management

This case study demonstrates how ERM helps design efficient healthcare information systems that manage patient data, medical staff, and clinical processes.

- **Key Entities:** Patient, Doctor, Appointment, MedicalRecord, Medication, Department
- **Critical Relationships:** Doctor treats Patient, Patient schedules Appointment, Doctor works in Department
- **Important Attributes:** Patient (patientID, name, DOB, insurance), Doctor (doctorID, specialization)

Real-World Applications:

- Appointment scheduling systems require Patient-Doctor-Appointment relationships
- Electronic Health Records (EHR) rely on Patient-MedicalRecord entity relationships
- Medication tracking systems leverage the Patient-Medication prescription relationship
- Hospital resource allocation based on Department-Doctor assignments

Implementation Benefits

Healthcare ER Diagram

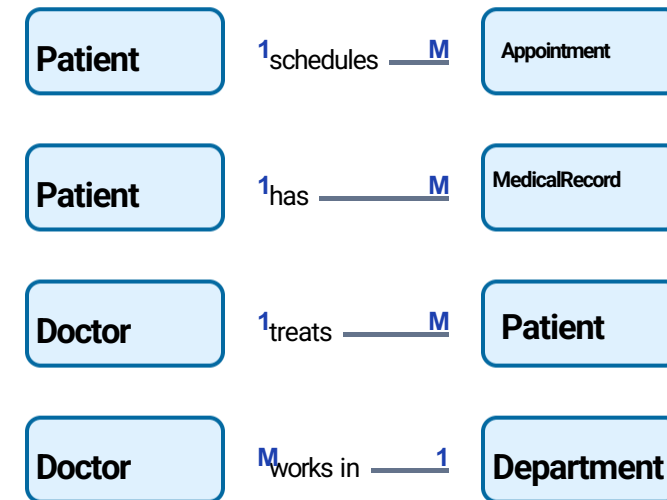


Figure: ER Diagram for Healthcare Patient Management System

Sample Attributes

Patient: patientID (PK), name, DOB, gender, contact, insurance

Relationship Multiplicity Types

- **One-to-One (1:1):** Each entity in A is associated with exactly one entity in B, and vice versa
Example: A Person has one Passport, a Passport belongs to one Person
- **One-to-Many (1:M):** Each entity in A can be associated with multiple entities in B, but each entity in B is associated with only one entity in A
Example: A Department employs many Employees, but each Employee works in only one Department
- **Many-to-Many (M:N):** Each entity in A can be associated with multiple entities in B, and vice versa
Example: A Student enrolls in many Courses, and each Course has many Students

Participation Constraints

- **Total/Mandatory (double line):** Every entity in the set must participate in the relationship
Example: Every Order must contain at least one Product
- **Partial/Optional (single line):** Some entities in the set may not participate in the relationship
Example: A Customer may or may not have placed an Order yet

Relationship Examples in Banking

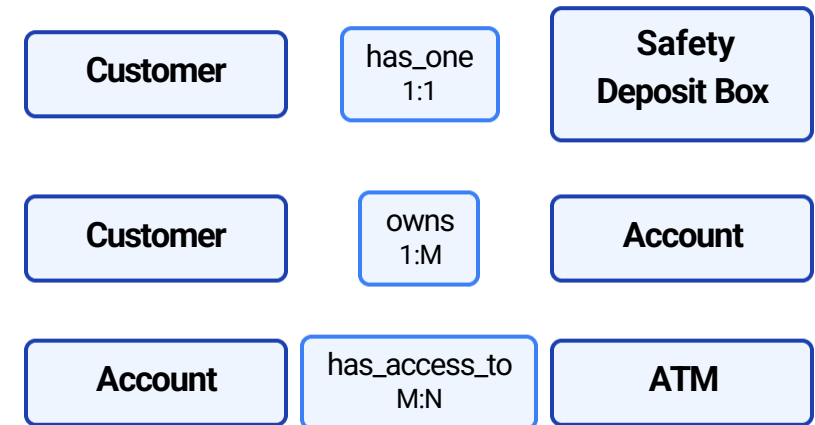


Figure: Banking system relationship examples showing different multiplicities

Notation Variations

Different notations exist across ER modeling standards, including Chen notation, Crow's Foot, IDEF1X, and UML. Each uses different symbols to represent the same multiplicity and participation constraints.



Practice Exercise: ER Diagram for a Banking System

Design an ER diagram for a banking system with the following requirements:

- **Entities:** Customer, Account, Transaction, Loan, Branch
- **Attributes:** Define primary keys and important attributes for each entity
- **Relationships:** Establish appropriate relationships between entities (consider cardinality)
- **Constraints:** A customer may have multiple accounts but an account belongs to only one customer
- **Constraints:** A transaction must be associated with at least one account

Questions to Consider:

- What attributes would make sense as unique identifiers (PKs)?
- How will you model the relationship between Customer and Loan?
- Should Branch be related to Customer, Account, or both?
- What attributes would you include for Transaction entity?



Hint: Consider the participation constraints between entities

Banking System ER Diagram (Starter)

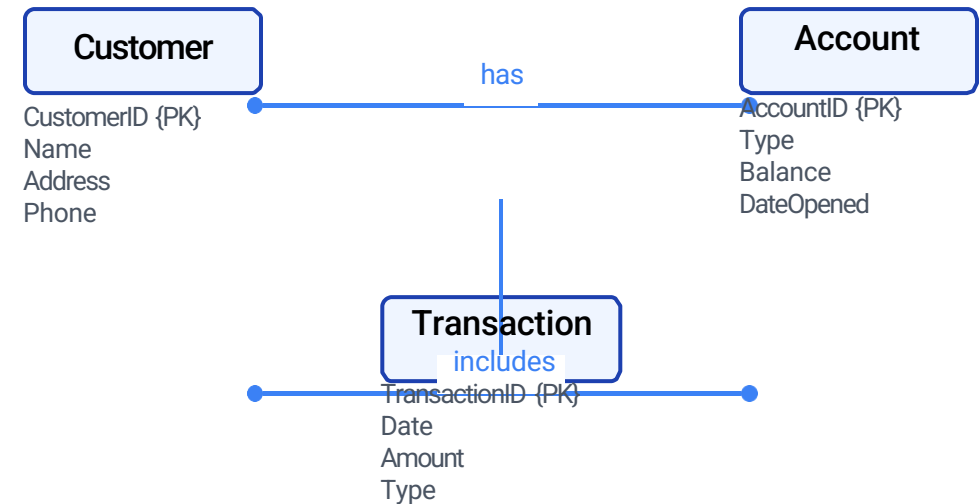


Figure: Basic entities and relationships for a banking system

Task

Complete this diagram by adding the Branch and Loan entities with their attributes. Define all relationships with proper cardinality notation. Consider how customers interact with branches and how loans are processed.

Key ERM Concepts Recap:

- **Entities:** Distinguishable things (strong/weak) like Products, Patients
- **Attributes:** Properties that describe entities (simple, composite, multi-valued)
- **Relationships:** Associations between entities with constraints (cardinality, participation)
- **ER Diagrams:** Visual representation using standardized notation
- **Real-world application:** Critical for healthcare systems, e-commerce platforms, banking applications, and university systems

Recommended Reading:

- Connolly/Begg "Database Systems" (4th ed.) - Chapter 11
- Connolly/Begg "Database Solutions" - Chapter 7
- Online resources: lucidchart.com/pages/er-diagrams, vertabelo.com/blog
- Industry case studies at edrawmax.com/database-tips

Database Design Process

Conceptual Model (ERM)

Business requirements, entities, relationships



Logical Model

Tables, columns, relationships, normalization



Physical Model

Specific DBMS implementation, indexes, storage

Remember

Good database design starts with thorough understanding of business requirements and translating them into accurate ER models before implementation.

