Chapter 3: Conceptual Database Schema (EA) and Mapping to the Relational Model

Entity–Association Paradigm and Translation Rules

Database Systems – Undergraduate

Department of Computer Science

February 14, 2026

Learning objectives

- Understand the **Entity–Association (EA)** conceptual model: entities, attributes, identifiers, associations, and cardinalities.
- Build a **Conceptual Data Model (CDM)** from requirements.
- Apply standard rules to translate EA/CDM into a **Relational schema** (tables, PK, FK).
- Practice on two case studies: **University** and **Public transport**.

Learning objectives

- Understand the **Entity–Association (EA)** conceptual model: entities, attributes, identifiers, associations, and cardinalities.
- Build a **Conceptual Data Model (CDM)** from requirements.
- Apply standard rules to translate EA/CDM into a **Relational schema** (tables, PK, FK).
- Practice on two case studies: **University** and **Public transport**.

Learning objectives

- Understand the **Entity–Association (EA)** conceptual model: entities, attributes, identifiers, associations, and cardinalities.
- Build a **Conceptual Data Model (CDM)** from requirements.
- Apply standard rules to translate EA/CDM into a **Relational schema** (tables, PK, FK).
- Practice on two case studies: **University** and **Public transport**.

Learning objectives

- Understand the **Entity–Association (EA)** conceptual model: entities, attributes, identifiers, associations, and cardinalities.
- Build a **Conceptual Data Model (CDM)** from requirements.
- Apply standard rules to translate EA/CDM into a **Relational schema** (tables, PK, FK).
- Practice on two case studies: **University** and **Public transport**.

Outline

1. Reminder: Data models
2. Conceptual Data Model: Entity–Association (EA)
3. Mapping EA/CDM to the Relational Model
4. Mapping associations by cardinality
5. Practice: Case studies

Reminder: Database & DBMS

Database

A database is a structured set of data enabling efficient **storage**, **management**, and **manipulation** of information.

DBMS

A DBMS provides services such as:

- DDL (schema definition),
- DML (query/update),
- access control and integrity,
- backup and recovery.

Reminder: Database & DBMS

Database

A database is a structured set of data enabling efficient **storage**, **management**, and **manipulation** of information.

DBMS

A DBMS provides services such as:

- DDL (schema definition),
- DML (query/update),
- access control and integrity,
- backup and recovery.

Reminder: Database & DBMS

Database

A database is a structured set of data enabling efficient **storage**, **management**, and **manipulation** of information.

DBMS

A DBMS provides services such as:

- DDL (schema definition),
- DML (query/update),
- access control and integrity,
- backup and recovery.

Reminder: Database & DBMS

Database

A database is a structured set of data enabling efficient **storage**, **management**, and **manipulation** of information.

DBMS

A DBMS provides services such as:

- DDL (schema definition),
- DML (query/update),
- access control and integrity,
- backup and recovery.

Reminder: Database & DBMS

Database

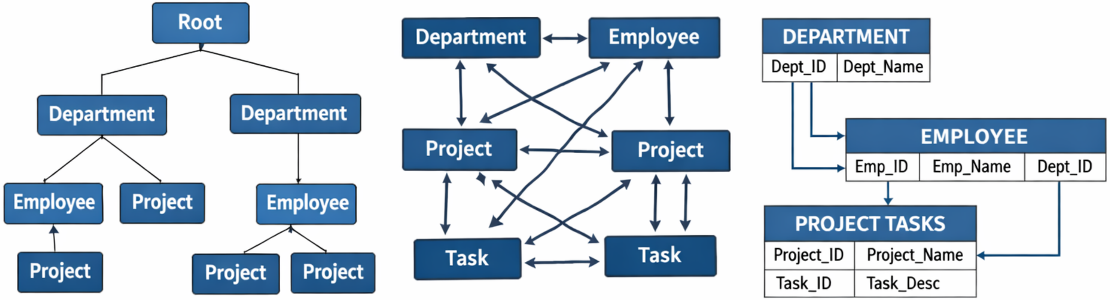
A database is a structured set of data enabling efficient **storage**, **management**, and **manipulation** of information.

DBMS

A DBMS provides services such as:

- DDL (schema definition),
- DML (query/update),
- access control and integrity,
- backup and recovery.

Examples of Data Models



DEPARTMENT	
Dept_ID	Dept_Name

EMPLOYEE		
Emp_ID	Emp_Name	Dept_ID

PROJECT TASKS	
Project_ID	Project_Name
Task_ID	Task_Desc

Hierarchical Model **Network Model** **Relational Model**

Interpretation

- Hierarchical model: tree structure ; parent → child
- Network model: graph with multiple links; many-to-many navigation
- Relational model: tables connected by keys

Interpretation

- Hierarchical model: tree structure ; parent → child
- Network model: graph with multiple links; many-to-many navigation
- Relational model: tables connected by keys

Interpretation

- Hierarchical model: tree structure ; parent → child
- Network model: graph with multiple links; many-to-many navigation
- Relational model: tables connected by keys

EA model: definition

Entity–Association (EA) model

A graphical representation of the data structure of a system.

It is based on:

- **Entities**: real-world objects (Student, Course, Patient, ...)
- **Attributes**: properties of entities (Name, BirthDate, ...)
- **Identifiers**: attributes that uniquely identify an entity occurrence
- **Associations (relationships)**: links between entities

EA model: definition

Entity–Association (EA) model

A graphical representation of the data structure of a system.

It is based on:

- **Entities**: real-world objects (Student, Course, Patient, . . .)
- **Attributes**: properties of entities (Name, BirthDate, . . .)
- **Identifiers**: attributes that uniquely identify an entity occurrence
- **Associations (relationships)**: links between entities

EA model: definition

Entity–Association (EA) model

A graphical representation of the data structure of a system.

It is based on:

- **Entities**: real-world objects (Student, Course, Patient, . . .)
- **Attributes**: properties of entities (Name, BirthDate, . . .)
- **Identifiers**: attributes that uniquely identify an entity occurrence
- **Associations (relationships)**: links between entities

EA model: definition

Entity–Association (EA) model

A graphical representation of the data structure of a system.

It is based on:

- **Entities**: real-world objects (Student, Course, Patient, . . .)
- **Attributes**: properties of entities (Name, BirthDate, . . .)
- **Identifiers**: attributes that uniquely identify an entity occurrence
- **Associations (relationships)**: links between entities

EA model: definition

Entity–Association (EA) model

A graphical representation of the data structure of a system.

It is based on:

- **Entities**: real-world objects (Student, Course, Patient, . . .)
- **Attributes**: properties of entities (Name, BirthDate, . . .)
- **Identifiers**: attributes that uniquely identify an entity occurrence
- **Associations (relationships)**: links between entities

Key notions (self-contained)

- **Identifier (conceptual)**: attribute(s) that uniquely identify an entity occurrence.
- **Primary key (relational)**: column(s) that uniquely identify a tuple (row).
- **Foreign key (relational)**: column(s) referencing a primary key in another table.
- **Referential integrity**: each FK value must exist in the referenced table.

Key notions (self-contained)

- **Identifier (conceptual)**: attribute(s) that uniquely identify an entity occurrence.
- **Primary key (relational)**: column(s) that uniquely identify a tuple (row).
- **Foreign key (relational)**: column(s) referencing a primary key in another table.
- **Referential integrity**: each FK value must exist in the referenced table.

Key notions (self-contained)

- **Identifier (conceptual)**: attribute(s) that uniquely identify an entity occurrence.
- **Primary key (relational)**: column(s) that uniquely identify a tuple (row).
- **Foreign key (relational)**: column(s) referencing a primary key in another table.
- **Referential integrity**: each FK value must exist in the referenced table.

Key notions (self-contained)

- **Identifier (conceptual)**: attribute(s) that uniquely identify an entity occurrence.
- **Primary key (relational)**: column(s) that uniquely identify a tuple (row).
- **Foreign key (relational)**: column(s) referencing a primary key in another table.
- **Referential integrity**: each FK value must exist in the referenced table.

Types of relationships (cardinalities)

- **1:1 (one-to-one)**: each A is linked to exactly one B and vice versa.
- **1:N (one-to-many)**: one A can be linked to many B; each B linked to one A.
- **M:N (many-to-many)**: many A linked to many B and vice versa.

Why cardinalities matter?

They determine how we translate relationships into tables and keys.

Types of relationships (cardinalities)

- **1:1 (one-to-one)**: each A is linked to exactly one B and vice versa.
- **1:N (one-to-many)**: one A can be linked to many B; each B linked to one A.
- **M:N (many-to-many)**: many A linked to many B and vice versa.

Why cardinalities matter?

They determine how we translate relationships into tables and keys.

Types of relationships (cardinalities)

- **1:1 (one-to-one)**: each A is linked to exactly one B and vice versa.
- **1:N (one-to-many)**: one A can be linked to many B; each B linked to one A.
- **M:N (many-to-many)**: many A linked to many B and vice versa.

Why cardinalities matter?

They determine how we translate relationships into tables and keys.

Types of relationships (cardinalities)

- **1:1 (one-to-one)**: each A is linked to exactly one B and vice versa.
- **1:N (one-to-many)**: one A can be linked to many B; each B linked to one A.
- **M:N (many-to-many)**: many A linked to many B and vice versa.

Why cardinalities matter?

They determine how we translate relationships into tables and keys.

Types of relationships (cardinalities)

- **1:1 (one-to-one)**: each A is linked to exactly one B and vice versa.
- **1:N (one-to-many)**: one A can be linked to many B; each B linked to one A.
- **M:N (many-to-many)**: many A linked to many B and vice versa.

Why cardinalities matter?

They determine how we translate relationships into tables and keys.

Relational model: quick reminder

- Data is stored in **relations** (tables).
- Each row is a **tuple** and each column is an **attribute**.
- A **primary key (PK)** uniquely identifies rows.
- A **foreign key (FK)** links tables and enforces integrity.

Relational model: quick reminder

- Data is stored in **relations** (tables).
- Each row is a **tuple** and each column is an **attribute**.
- A **primary key (PK)** uniquely identifies rows.
- A **foreign key (FK)** links tables and enforces integrity.

Relational model: quick reminder

- Data is stored in **relations** (tables).
- Each row is a **tuple** and each column is an **attribute**.
- A **primary key (PK)** uniquely identifies rows.
- A **foreign key (FK)** links tables and enforces integrity.

Relational model: quick reminder

- Data is stored in **relations** (tables).
- Each row is a **tuple** and each column is an **attribute**.
- A **primary key (PK)** uniquely identifies rows.
- A **foreign key (FK)** links tables and enforces integrity.

General principle: Relationship mapping

Key idea

The translation strategy depends on the **relationship cardinality**: 1:1, 1:N, M:N (and sometimes ternary).

We illustrate each case with:

- an EA diagram (figure),
- the relational schema produced.

General principle: Relationship mapping

Key idea

The translation strategy depends on the **relationship cardinality**: 1:1, 1:N, M:N (and sometimes ternary).

We illustrate each case with:

- an EA diagram (figure),
- the relational schema produced.

General principle: Relationship mapping

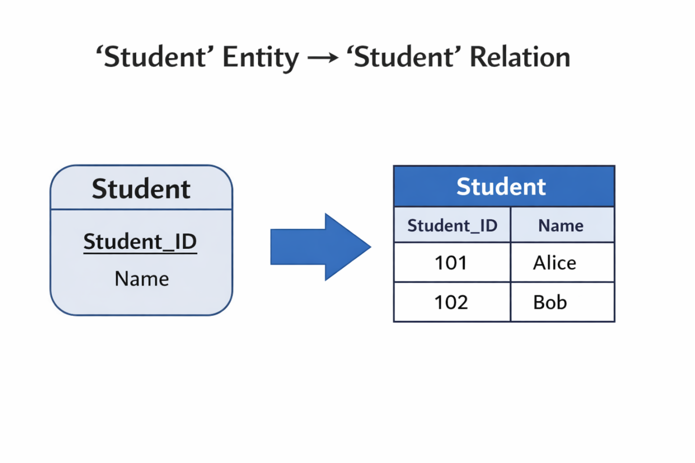
Key idea

The translation strategy depends on the **relationship cardinality**: 1:1, 1:N, M:N (and sometimes ternary).

We illustrate each case with:

- an EA diagram (figure),
- the relational schema produced.

Rule 1: Entity → Table



‘Student’ Entity → ‘Student’ Relation

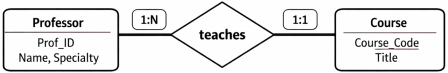
Student	
Student_ID	Name
101	Alice
102	Bob

Translation rule

- Each entity becomes a table
- Attributes → columns
- Identifier → primary key

Association 1:N — Example and mapping

EA diagram (1:N)



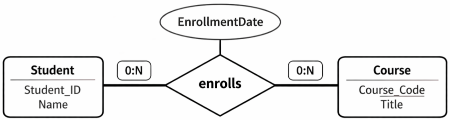
Rule (1:N)

- Put the PK of the **1-side** as an FK in the **N-side**.

Professor(Prof_ID, Name, Specialty)
Course(Course_Code, Title, Prof_ID *FK*)

Association M:N — Example and mapping

EA diagram (M:N) with an attribute



Relational mapping rule

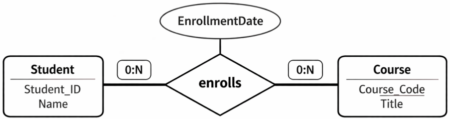
Rule (M:N)

- Create an **intersection table**.
- Include both PKs as FKs.
- Composite PK = (FK1, FK2).
- Add relationship attributes.

Enrollment($\underline{\text{Student\_ID}}$ *FK*, $\underline{\text{Course\_Code}}$ *FK*, EnrollmentDate)

Association M:N — Example and mapping

EA diagram (M:N) with an attribute



Relational mapping rule

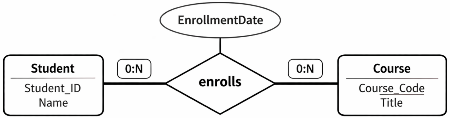
Rule (M:N)

- Create an **intersection table**.
- Include both PKs as FKs.
- Composite PK = (FK1, FK2).
- Add relationship attributes.

Enrollment(Student_ID *FK*, Course_Code *FK*, EnrollmentDate)

Association M:N — Example and mapping

EA diagram (M:N) with an attribute



Relational mapping rule

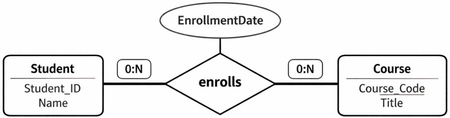
Rule (M:N)

- Create an **intersection table**.
- Include both PKs as FKs.
- Composite PK = (FK1, FK2).
- Add relationship attributes.

Enrollment(Student_ID *FK*, Course_Code *FK*, EnrollmentDate)

Association M:N — Example and mapping

EA diagram (M:N) with an attribute



Relational mapping rule

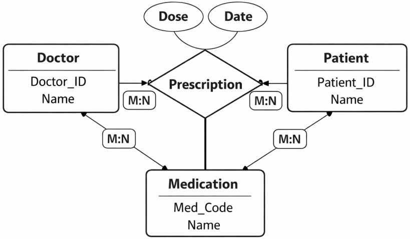
Rule (M:N)

- Create an **intersection table**.
- Include both PKs as FKs.
- Composite PK = (FK1, FK2).
- Add relationship attributes.

Enrollment($\underline{\text{Student\_ID}}$ *FK*, $\underline{\text{Course\_Code}}$ *FK*, EnrollmentDate)

Ternary association (3 entities) — Example and mapping

EA diagram (ternary)



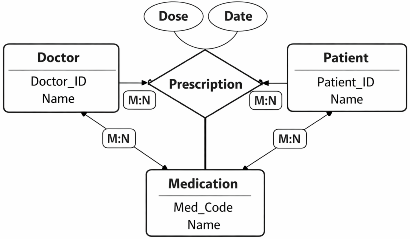
Relational mapping

- Create one table containing the 3 PKs as FKs.
- Composite PK uses the 3 FKs.
- Add ternary attributes (e.g., Dose, Date).

Prescription(Doctor_ID *FK*, Patient_ID *FK*, Med_Code *FK*, Dose, PrescDate)

Ternary association (3 entities) — Example and mapping

EA diagram (ternary)



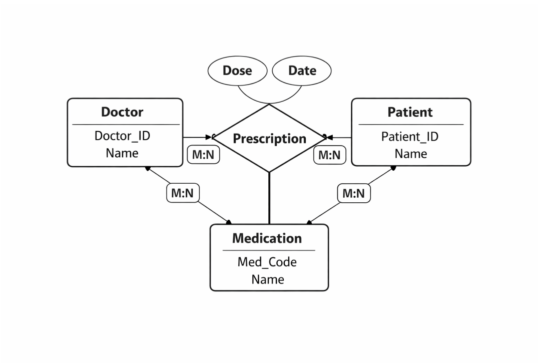
Relational mapping

- Create one table containing the 3 PKs as FKs.
- Composite PK uses the 3 FKs.
- Add ternary attributes (e.g., Dose, Date).

Prescription(Doctor_ID *FK*, Patient_ID *FK*, Med_Code *FK*, Dose, PrescDate)

Ternary association (3 entities) — Example and mapping

EA diagram (ternary)



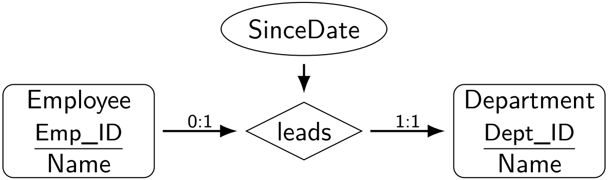
Relational mapping

- Create one table containing the 3 PKs as FKs.
- Composite PK uses the 3 FKs.
- Add ternary attributes (e.g., Dose, Date).

Prescription(Doctor_ID *FK*, Patient_ID *FK*, Med_Code *FK*, Dose, PrescDate)

Association 1:1 — Example and mapping

EA diagram (1:1): Department has one Chief



Relational mapping (1:1) — two possibilities

- **Option A (recommended):** add FK in one table (prefer total participation side).
- **Option B:** merge entities if they are strongly linked (often 1:1 + total participation).

Option A (FK in Department):
Employee(Emp_ID, Name)
Department(Dept_ID, Name, Chief_Emp_ID *FK* → Employee(Emp_ID), SinceDate)

Option B (merge into one relation):
DeptChief(Dept_ID, DeptName, Emp_ID, EmpName, SinceDate)

Summary: association mapping rules

Type	Relational mapping rule
1:1	UNIQUE FK (one side) **or** merge both entities
1:N	FK in the N-side referencing the 1-side PK
M:N	Intersection table: FK1, FK2, composite PK(FK1,FK2)
Ternary	Table with 3 FKs, composite PK(FK1,FK2,FK3), + attributes

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

Exercise 1: University (statement)

A university wants to model:

- Students: ID, Name, BirthDate.
- Teachers: ID, Name, Specialty.
- Courses: Code, Title.
- Each course is taught by exactly one teacher.
- Students attend many courses and courses have many students.
- For each student-course enrollment, store the **Grade**.

Tasks:

- Build the EA model.
- Translate it into a relational schema (tables + PK + FK).

End

Questions?