Database Systems — Week 3 Notes: The Entity Relationship Model

Overview

This week takes a database from an idea to real tables: we model it with an ER diagram, resolve its many-to-many relationships, document each table in a Table Instance Chart (TIC), and translate the TICs into SQL.

References: Elmasri & Navathe, *Fundamentals of Database Systems*, 6th edition, Chapter 7 (ER model) and Chapter 9 (ER-to-relational mapping). The TIC format follows the Oracle design methodology.

Running example: a small UNIVERSITY database. A department offers many courses; every student majors in one department and enrolls in many courses, receiving a grade in each.

By the end of the week, students should be able to:

1. Define entity, attribute and relationship, and classify attributes.
2. Determine the cardinality ratio and participation of a relationship.
3. Read and draw ER diagrams in Chen and crow's foot notation.
4. Resolve an M:N relationship into an intersection entity.
5. Write a TIC for each table, with keys, nulls and foreign keys.
6. Translate TICs into CREATE TABLE statements.

Part 1: The ER model and its building blocks

The Entity–Relationship (ER) model, proposed by Peter Chen in 1976, is a high-level (conceptual) data model. It describes data as users see it, independent of any DBMS, and is used in the conceptual design step: requirements → **conceptual design (ER)** → logical design (tables, TICs) → physical design (indexes, storage).

Entities

- **Entity:** a thing in the miniworld with an independent existence, physical (a student) or conceptual (a course).
- **Entity type:** a set of entities with the same attributes, e.g. STUDENT. One entity, such as Smith, is an instance of it.
- **Entity set:** the entities of one type stored at a given moment; later these become the rows of a table.
- **Strong entity type:** has its own key (STUDENT, key Student_id).
- **Weak entity type:** has no key of its own and is identified through an owner entity plus a *partial key* (DEPENDENT of an EMPLOYEE, identified by the employee's id + the dependent's Name).

Attributes

Kind	Meaning	Example (STUDENT)
Simple (atomic)	Cannot be divided	Student_id

Kind	Meaning	Example (STUDENT)
Composite	Made of smaller parts	Name = First_name + Last_name
Single-valued	One value per entity	Birth_date
Multivalued	Several values per entity	Phone
Stored	Kept in the database	Birth_date
Derived	Computed from other data	Age (from Birth_date)
Key	Unique for each entity	Student_id
NULL value	Unknown or not applicable	Phone of a student with no phone

Relationships

A relationship is an association among entities, such as Smith *enrolls in* Database. A relationship type is named with a verb (ENROLLS, OFFERS, MAJORS_IN).

- **Degree:** the number of entity types involved. Unary or recursive (COURSE is a prerequisite of COURSE), binary (STUDENT enrolls in COURSE, the most common), ternary (SUPPLIER supplies PART to PROJECT).
- **Relationship attributes:** a relationship can have its own attributes. Grade belongs to ENROLLS, because a student has one grade per course, not one grade overall.

Structural constraints

Cardinality ratio is the maximum number of relationship instances an entity can take part in:

Ratio	Meaning	Example
1:1	Each entity relates to at most one on the other side	EMPLOYEE manages DEPARTMENT
1:N	One entity relates to many; each of those relates to one	DEPARTMENT offers COURSE
M:N	Many on both sides	STUDENT enrolls in COURSE

To find the ratio, read the relationship in both directions: "one department offers many courses" and "one course is offered by one department" gives 1:N.

Participation is the minimum:

- **Total (mandatory):** every entity must participate, e.g. every COURSE must be offered by a DEPARTMENT. Shown as a double line in Chen notation; usually becomes NOT NULL on the foreign key.

- **Partial (optional):** some entities may not participate, e.g. not every EMPLOYEE manages a department.

Part 2: Drawing ER diagrams

An ER diagram shows the conceptual schema graphically. Two notations are common; students should be able to read both.

Chen notation (textbook)

Symbol	Meaning
Rectangle	Entity type
Double rectangle	Weak entity type
Diamond	Relationship type
Double diamond	Identifying relationship (owner to weak entity)
Oval	Attribute
Oval with underlined name	Key attribute
Double oval	Multivalued attribute
Dashed oval	Derived attribute
Oval connected to smaller ovals	Composite attribute
Single / double line	Partial / total participation
1, N, M written on the lines	Cardinality ratio

Crow's foot notation (industry tools and TICs)

Entities are boxes with their attributes listed inside; the relationship is a line, and symbols at each end show the cardinality. A bar (|) means one, a crow's foot means many, and a circle (O) means zero (optional). Two bars mean "one and only one". This notation is used by tools such as Oracle Data Modeler and MySQL Workbench, and in the Oracle methodology that uses TICs.

The UNIVERSITY ER diagram

```
erDiagram
  DEPARTMENT ||--|{ STUDENT : "majors in"
  DEPARTMENT ||--|{ COURSE : offers
  STUDENT }o--o{ COURSE : "enrolls in"
  DEPARTMENT {
    int Dept_id PK
    string Dept_name
  }
  STUDENT {
    int Student_id PK
```

```
    string Name
    string Phone "multivalued"
  }
  COURSE {
    string Course_id PK
    string Title
    int Credits
  }
```

Reading the diagram: every student majors in exactly one department, and a department has many students (N:1, total for STUDENT). A department offers many courses, and every course belongs to one department (1:N, total for COURSE). A student enrolls in many courses and a course has many students (M:N); the enrollment carries its own attributes, Grade and Semester.

How to draw an ER diagram from requirements

1. Find the **entities**: the important nouns (department, course, student).
2. Find the **relationships**: the verbs that connect them (offers, majors in, enrolls in).
3. List the **attributes** of each entity and choose a **key**.
4. Add the **cardinality ratio** and **participation** to each relationship.
5. Review with users, then **resolve M:N** relationships before moving to tables.

Not every noun is an entity: "grade" is a noun in the requirements but becomes an attribute of the ENROLLS relationship.

Part 3: Resolving M:N relationships

A many-to-many relationship cannot be stored directly in relational tables, so we replace it with a new intersection entity and two 1:N relationships.

Why M:N cannot go straight into a table

Suppose we try to store enrollments inside STUDENT:

Student_id	Name	Courses	Grades
17	Smith	CS3380, CS3320	B, C
8	Brown	CS3380	A

A cell now holds a list, which breaks the rule that every value in a relation is atomic. We cannot join or index on a single course, updating one grade is awkward, there is no place for the semester, and a foreign key cannot hold a list. Putting the list in COURSE has the same problem from the other side.

The resolution

```
erDiagram
  STUDENT ||--o{ ENROLLMENT : has
  COURSE ||--o{ ENROLLMENT : has
  ENROLLMENT {
    int Student_id PK, FK
    string Course_id PK, FK
    string Semester
```

```
    string Grade
  }
```

The new entity is called an *intersection*, *associative*, *junction* or *bridge* entity.

Resolution rules

1. Create a new entity named after the event or fact: ENROLLMENT, ASSIGNMENT, ORDER_LINE.
2. Connect it to each original entity with a 1:N relationship; the new entity is the N side.
3. Its identifier combines the keys of both parents (Student_id + Course_id).
4. Move the attributes of the M:N relationship (Grade, Semester) into the new entity.
5. Check the key: if the same pair can repeat (a student retakes a course), add Semester to the key.
6. The new entity participates totally in both relationships: every enrollment belongs to a student and a course.

More examples

M:N relationship	Intersection entity	Key	Own attributes
EMPLOYEE works on PROJECT	ASSIGNMENT	Emp_id + Project_id	Hours
ORDER contains PRODUCT	ORDER_LINE	Order_id + Product_id	Quantity, Unit_price
AUTHOR writes BOOK	AUTHORSHIP	Author_id + ISBN	Author_order

Part 4: Table Instance Charts (TICs)

A Table Instance Chart is a design worksheet for one table. It sits between the ER diagram and the SQL code, and lets you check keys, nulls and foreign keys before writing any code.

Each attribute gets a column; fixed rows describe it:

Row	What you write
Column Name	The column's name
Key Type	PK (primary key), FK (foreign key), UK (unique key)
Nulls/Unique	NN = not null, U = unique
FK Table	The table the foreign key references
FK Column	The column it references
Sample Data	Two or three realistic rows

Sample data is invented for illustration. It often reveals mistakes, such as a "key" whose values are not really unique.

TIC: DEPARTMENT

	Dept_id	Dept_name
Key Type	PK	
Nulls/Unique	NN, U	NN, U
FK Table		
FK Column		
Sample Data	1	Computer Engineering
	2	Electrical Engineering

TIC: STUDENT

	Student_id	Name	Dept_id
Key Type	PK		FK
Nulls/Unique	NN, U	NN	NN
FK Table			DEPARTMENT
FK Column			Dept_id
Sample Data	17	Smith	1
	8	Brown	1

The foreign key Dept_id comes from the 1:N relationship MAJORS_IN and goes on the N side. It is NN because every student must have a major (total participation). The multivalued Phone gets its own table, STUDENT_PHONE.

TIC: COURSE

	Course_id	Title	Credits	Dept_id
Key Type	PK			FK
Nulls/Unique	NN, U	NN	NN	NN
FK Table				DEPARTMENT
FK Column				Dept_id
Sample Data	CS3380	Database	3	1
	CS3320	Data Structures	4	1

TIC: ENROLLMENT (the resolved M:N)

	Student_id	Course_id	Semester	Grade
Key Type	PK, FK	PK, FK		
Nulls/Unique	NN	NN	NN	
FK Table	STUDENT	COURSE		
FK Column	Student_id	Course_id		
Sample Data	17	CS3380	Fall 2026	B
	8	CS3380	Fall 2026	A

Both Student_id and Course_id are part of the primary key and are also foreign keys; together they form a composite primary key. They are not marked U individually, because only the pair is unique. Grade may be NULL until the exam is marked.

Part 5: Translating TICs into relational tables

Each TIC becomes one CREATE TABLE statement, following fixed mapping rules from the ER construct to the relational schema.

ER-to-relational mapping rules

ER construct	Becomes in the relational schema
Strong entity type	A table; simple attributes become columns; the key becomes the PRIMARY KEY
Composite attribute	Only its simple parts become columns (First_name, Last_name)
Multivalued attribute	A new table: the owner's key + the value, together as the primary key
Derived attribute	Not stored; computed in queries (Age from Birth_date)
Weak entity type	A table whose key = owner's key + partial key, with an FK to the owner
1:1 relationship	FK in one of the tables, preferably the total-participation side, plus UNIQUE
1:N relationship	FK in the table on the N side
M:N relationship	A new intersection table; key = both FKs; relationship attributes become columns

This is a simplified form of Elmasri & Navathe's mapping algorithm (Chapter 9). A ternary relationship follows the M:N rule with three foreign keys.

From TIC entries to SQL

TIC entry	SQL
PK	PRIMARY KEY
NN	NOT NULL
U	UNIQUE
FK + FK Table + FK Column	FOREIGN KEY (col) REFERENCES table(column)
Sample Data	INSERT INTO … VALUES (…)

Create parent tables before child tables, because a foreign key must reference a table that already exists.

```
CREATE TABLE DEPARTMENT (
  Dept_id   INT PRIMARY KEY,
  Dept_name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE STUDENT (
  Student_id INT PRIMARY KEY,
  Name       VARCHAR(50) NOT NULL,
  Dept_id    INT NOT NULL,
  FOREIGN KEY (Dept_id) REFERENCES DEPARTMENT(Dept_id)
);

CREATE TABLE STUDENT_PHONE (
  Student_id INT,
  Phone      VARCHAR(15),
  PRIMARY KEY (Student_id, Phone),
  FOREIGN KEY (Student_id) REFERENCES STUDENT(Student_id)
);

CREATE TABLE COURSE (
  Course_id CHAR(8) PRIMARY KEY,
  Title     VARCHAR(60) NOT NULL,
  Credits   INT NOT NULL,
  Dept_id   INT NOT NULL,
  FOREIGN KEY (Dept_id) REFERENCES DEPARTMENT(Dept_id)
);

CREATE TABLE ENROLLMENT (
  Student_id INT NOT NULL,
  Course_id  CHAR(8) NOT NULL,
  Semester   VARCHAR(12) NOT NULL,
  Grade      CHAR(1),
  PRIMARY KEY (Student_id, Course_id),
  FOREIGN KEY (Student_id) REFERENCES STUDENT(Student_id),
  FOREIGN KEY (Course_id)  REFERENCES COURSE(Course_id)
```

```
);

INSERT INTO ENROLLMENT VALUES (17, 'CS3380', 'Fall 2026', 'B');
```

If students may retake a course, change the ENROLLMENT key to PRIMARY KEY (Student_id, Course_id, Semester).

The final relational schema

Bold here (underlined on the board) = primary key; → = foreign key and the table it references.

- DEPARTMENT (**Dept_id**, Dept_name)
- STUDENT (**Student_id**, Name, Dept_id → DEPARTMENT)
- STUDENT_PHONE (**Student_id → STUDENT, Phone**)
- COURSE (**Course_id**, Title, Credits, Dept_id → DEPARTMENT)
- ENROLLMENT (**Student_id → STUDENT, Course_id → COURSE**, Semester, Grade)

Three entities, one multivalued attribute and one M:N relationship give five tables. The two 1:N relationships became foreign keys, not tables.

Common mistakes

Mistake	Fix
Foreign key on the 1 side of a 1:N	Put it on the N side (COURSE gets Dept_id)
Several phones in one column	A separate table for the multivalued attribute
M:N left unresolved (a list of courses in STUDENT)	An intersection table with a composite key
Grade placed in STUDENT or COURSE	Relationship attributes go in the intersection table
Storing Age	Store Birth_date and compute Age
A separate table for every 1:N relationship	A 1:N needs only a foreign key

Summary and exercise

Question	Answer in one line
ER model	A conceptual model built from entities, attributes and relationships, with cardinality and participation.
ER diagrams	Chen: rectangles, diamonds, ovals. Crow's foot: boxes with symbols at the line ends.
M:N resolution	Replace M:N with an intersection entity and two 1:N relationships; key = both parent keys.
TICs	A worksheet per table: column names, key type,

Question	Answer in one line
	nulls/unique, FK table and column, sample data.
TIC to tables	PK → PRIMARY KEY, NN → NOT NULL, U → UNIQUE, FK → FOREIGN KEY … REFERENCES.

Exercise: library database

A library has members and books. Each book has one publisher; a publisher publishes many books. A book can have several authors, and an author writes many books. A member borrows many books over time, and a book is borrowed by many members; record the borrow date and the return date.

1. Draw the ER diagram with cardinalities and keys.
2. Resolve every M:N relationship.
3. Write a TIC for each table, with two rows of sample data.
4. Translate the TICs into CREATE TABLE statements.

Model answer (for the instructor): entities MEMBER, BOOK, PUBLISHER, AUTHOR. BOOK holds Publisher_id as a foreign key (1:N). AUTHOR–BOOK resolves into AUTHORSHIP (Author_id, ISBN). MEMBER–BOOK resolves into BORROWING (Member_id, ISBN, Borrow_date, Return_date), with Borrow_date in the key because the same member can borrow the same book again.