Database Fundamentals

Cybersecurity Department

Course Code: ***CBS 213***

Practical Lecture 4: Implementing Relationships in MySQL

Halal Abdulrahman Ahmed

Outlines

- Introduction to database relationships
- Primary key vs. foreign key refresher
- One-to-One relationship (example + SQL)
- One-to-Many relationship (example + SQL)
- Many-to-Many relationship & junction table (example + SQL)
- Self-Referencing relationship (example + SQL)
- Hands-on practice in MySQL

Learning Outcomes

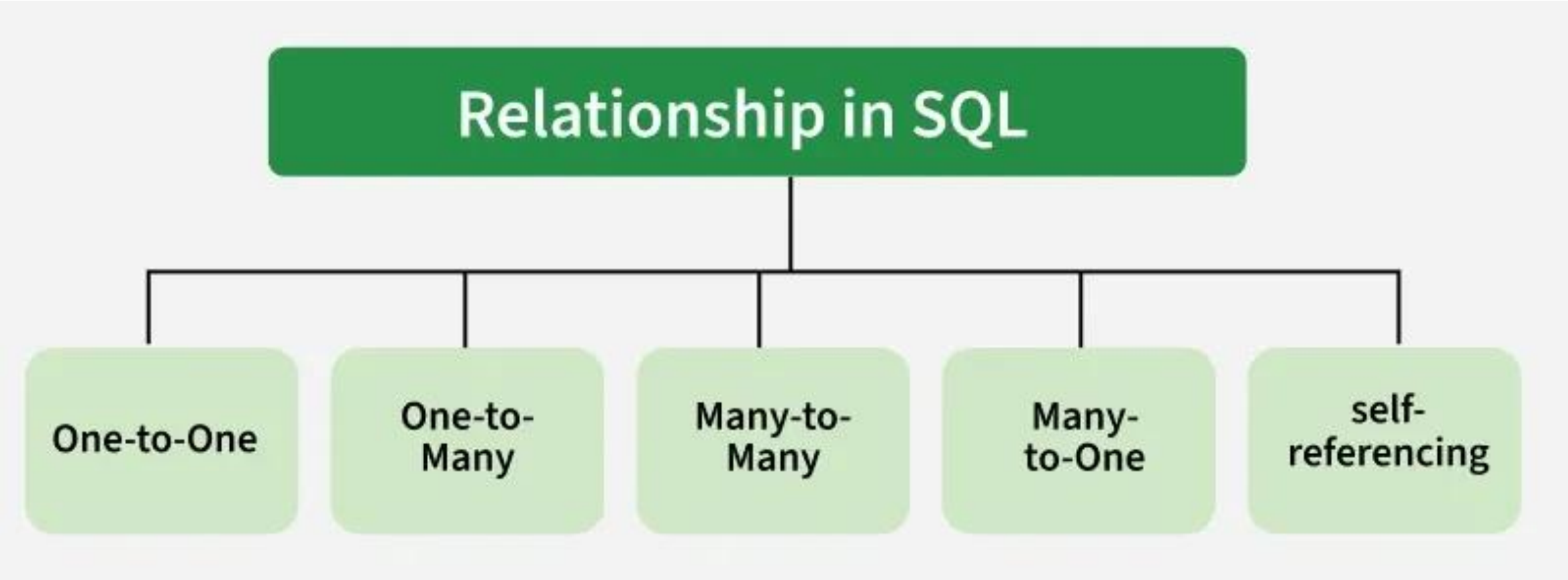
By the end of this lecture, students will be able to:

- Define database relationships and explain why they are used.
- Identify One-to-One, One-to-Many, Many-to-Many, and Self-Referencing relationships.
- Use primary keys and foreign keys to connect tables in MySQL.
- Create SQL tables that implement different types of relationships.
- Build a junction table with a composite key for many-to-many relationships.
- Apply SQL queries to test and validate table relationships.

What is a Relationship in Databases?

- A **relationship** connects data in two or more tables using keys.
- Relationships prevent data duplication and maintain logical links.
- They rely on **foreign keys** that reference a **primary key** in another table.

Types of Relationships



1. One-to-One Relationship

- Each record in Table A is associated with one and only one record in Table B, and vice versa.
- Setup: Include a foreign key in one of the tables that references the primary key of the other table.
- For example: Tables users (A) and user_profiles (B), where each user has a **single** corresponding profile.

TABLE A			TABLE B		
user_id	username		profile_id	user_id	profile_data
1	ramesh		p01	1	xyz
2	riya		p02	2	abc
3	akhil		p03	3	gfg
			foreign key		
			primary key		

```sql
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(50)
);

CREATE TABLE user_profiles (
    profile_id INT PRIMARY KEY,
    user_id INT UNIQUE,
    profile_data VARCHAR(255),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

Creates a new table in the database

Constraint that prevents duplicate values

2. One-to-Many Relationship

- **Each** record in Table A (Departments) can be associated with **multiple** records in Table B (Employees), but **each** record in Table B is associated with **only one** record in Table A.
- Setup: Include a foreign key in the "many" side table (Table B) that references the primary key of the "one" side table (Table A).
- For example: Tables departments and employees, where each department can have multiple employees, but each employee belongs to one department.

Departments

department_id	department_name
d1	technical
d2	accounts
d3	pr
d4	product management

Employees

employee_id	employee_name	department_id
e01	Ramesh	d3
e02	Riya	d1
e03	Neha	d2
e04	Mayank	d1
e05	Kritila	d4
e06	Anuj	d4
e07	Sam	d1
e08	Gurpreet	d2

foreign key

primary key

```sql
CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50)
);

CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

Column in departments table, it must match the department ID there

This column in the employees table

The table we are connecting to

3. Many-to-Many Relationship

- **Each** record in Table A (Students) can be associated with **multiple** records in Table B (Courses), and vice versa.
- Setup: Create an intermediate table (also known as a junction or linking table) that contains foreign keys referencing both related tables. Linking table is a table that contains **two foreign keys**, each one coming from a different table, to link them together.
- For example: Tables students and courses, where each student can enroll in multiple courses, and each course can have multiple students.

STUDENTS

student_id	student_name
1	Alice
2	Bob
3	Charlie

COURSES

course_id	course_name
101	Mathematics
102	History
103	Computer Science

STUDENT_COURSES

student_id	course_id
1	101
1	102
2	102

foreign key

primary key

both primary and foreign key

STUDENTS

student_id	student_name
1	Alice
2	Bob
3	Charlie

COURSES

course_id	course_name
101	Mathematics
102	History
103	Computer Science

foreign key

primary key

both primary and foreign key

STUDENT_COURSES

student_id	course_id
1	101
1	102
2	102

Together they form a Composite Key

```sql
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50)
);

CREATE TABLE courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(50)
);

CREATE TABLE student_courses (
    student_id INT,
    course_id INT,
    PRIMARY KEY (student_id, course_id),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

Column names

Linking table

Composite key

Foreign Keys

Try this code

```sql
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50)
);

CREATE TABLE courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(50)
);

CREATE TABLE student_courses (
    student_id INT,
    course_id INT,
    PRIMARY KEY (student_id, course_id),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

4. Many-to-One Relationship

- **Multiple** records in table B (Teachers) can be associated with **one** record in table A (Courses).
- Setup: Crate a Foreign key in "Many Table" that references to Primary Key in "One Table".
- Example: Table Teachers and Courses, many courses can be taught by single teacher.

Teachers

teacher_id	first_name	last_name
101	Ben	Johnson
102	Harish	Patel

Courses

course_id	course_name	teacher_id
201	Math 101	101
202	Computer Science	102
203	Physics Lab	101

Primary Key

Foreign Key

```sql
CREATE TABLE Teachers (
    teacher_id INT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255)
);

CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(255),
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);
```

5. Self-Referencing Relationship

- A table has a foreign key that references its primary key. A **Self-Referencing Relationship** (also called **Self-Join** or **Recursive Relationship**) is when a table has a relationship with itself.
- **Setup:** Include a foreign key column in the same table that references its primary key.
- **For example :** A table employees with a column manager_id referencing the same table's employee_id. It shows one table (employees) where one employee can be another employee's manager.

employees

employee_id	employee_name	manager_id
1	Alice	NULL
2	Bob	1
3	Charlie	1
		foreign key
		primary key

```sql
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
    manager_id INT,
    FOREIGN KEY (manager_id) REFERENCES employees(employee_id)
);
```

References

Simmons, S., & Teyzal, A. (2022). *MySQL cookbook: Solutions for database developers and administrators* (4th ed.). O'Reilly Media.

GeeksforGeeks. (n.d.). *Relationships in SQL: One-to-one, one-to-many, many-to-many*. Retrieved November 2, 2025, from https://www.geeksforgeeks.org/sql/relationships-in-sql-one-to-one-one-to-many-many-to-many/


Any
Question