Basics of database systems

Project – Database design

Sepideh Soleimani

Lappeenranta-Lahti University of Technology LUT
Software Engineering

Basics of database systems
Spring 2024

TABLE OF CONTENTS

1 DEFINITION

In the project 'Course Database,' the database is designed to meet the needs of an educational institution offering various courses to its members. The institution manages information related to members, mentors (Mentors), courses, enrollments, payments, and equipment. The primary goal is to streamline the administration of courses, memberships, and associated transactions.

The educational institution faces challenges in efficiently managing course enrollments, tracking member information, handling mentor assignments, processing payments, and maintaining equipment associated with classrooms. The need for a centralized database arises to ensure accurate and timely information retrieval, streamline administrative tasks, and provide a seamless experience for both members and mentors.

The database enables administrators to have complete control and oversight, allowing them to manage courses, members, and financial transactions. Staff members, in addition to accessing their personal information, can read details about courses, members, and enrollments. Members, representing students or participants, have the ability to access and modify their personal information. They can also retrieve information about courses, enrollments, and payments, ensuring transparency and ease of use.

Courses may have multiple enrollments, reflecting the dynamic nature of educational offerings. Payment records are linked to courses and members, allowing for a comprehensive overview of financial transactions within the educational system.

In essence, this database not only supports the administration of existing courses and enrollments but also provides a foundation for planning and managing future educational journeys. The system is designed to enhance the user experience, streamline administrative tasks, and ensure accuracy in the representation of course-related data.

2 MODELING

2.1 Conceptual model

In the conceptual model of the COURSE database seen in Figure 1, there are six entities: Member, Mentor, Course, Enrollment, Payment, and Equipment.

The 1:1 relationship between the Member and Enrollment entities signifies that one member can be associated with multiple enrollments, illustrating the flexibility for individuals to enroll in various courses. Also, the 1:1 relationship of the Enrollment-Payment shows that each payment corresponds to a specific enrollment. Mentor-Course relationship follows a one-to-many 1:N relationship, signifying that a mentor can instruct multiple courses, but each course has a singular Mentor. The N:M relationship between Member and Course shows that a member can enroll in multiple courses, and each course can have multiple members enrolled. Hence, it is a many-to-many relationship.

The conceptual model includes the 'YearOfBirth' attribute for members. During implementation, age can be derived from the 'YearOfBirth' attribute and may not need to be stored explicitly. The 'Phone' attribute for both members and mentors is multivalued. Depending on implementation decisions, it might be reduced to a single value, split into multiple fields, or represented by an additional relation.

Attributes of the entities shown in the figure are the data that will be stored in the database. Underlined attributes are key attributes and they identify each data set from one another in the entity.

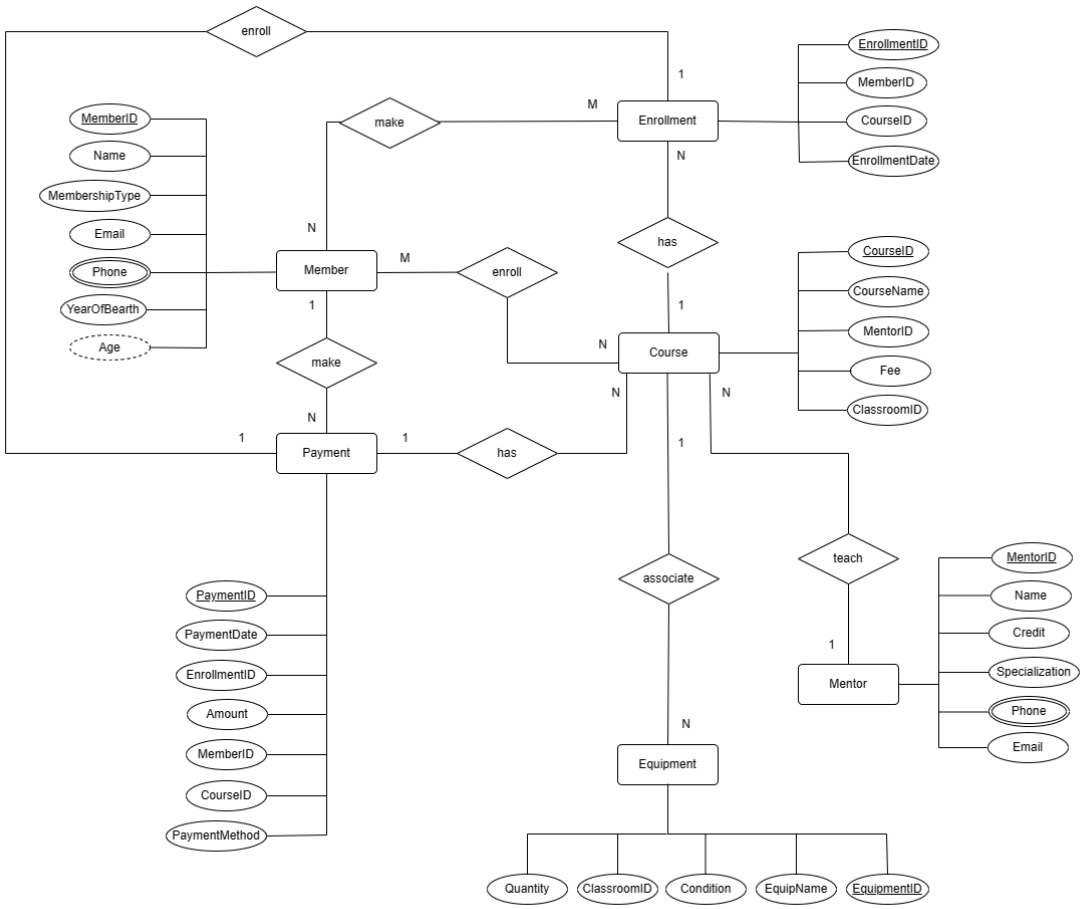

Figure 1: ER model

2.2 Logical model

In Figure 2, the logical model derived from the Course Database's ER model is presented. This logical representation serves as a refined structure that translates the abstract relationships and entities of the conceptual model into a form suitable for database implementation. The logical model emphasizes the organization of data into tables, establishing primary and foreign key relationships, defining data types, and specifying value ranges for attributes.

The tranformation from ER-model to a logical model seen in Figure 2 was done using the transformation rules. First each strong entity was made into a relation and key attributes were used to form primary keys for the entities. Primary keys are presented in the

model with PK next to the data type of the primary key. The transformation of attributes was simple because the ER-model only had a few derived attributes in addition to key and normal attributes. Derived attributes had to be discarded. Finally, the relationships between entites had to be transformed.

The 1:1 relationships meant that one of the relations had to have a foreign key in the relational model. The foreign keys were placed in the relation that made most sense to the function of the database. In 1:N relationships the foreign key was placed on the “many” side of the relationship. In the N:M relationships a linking relation had to be formed, which would store the attributes that were directly related to the relationship and the foreign keys to both of the relations in the relationship. And there is no FK related to Equipment.

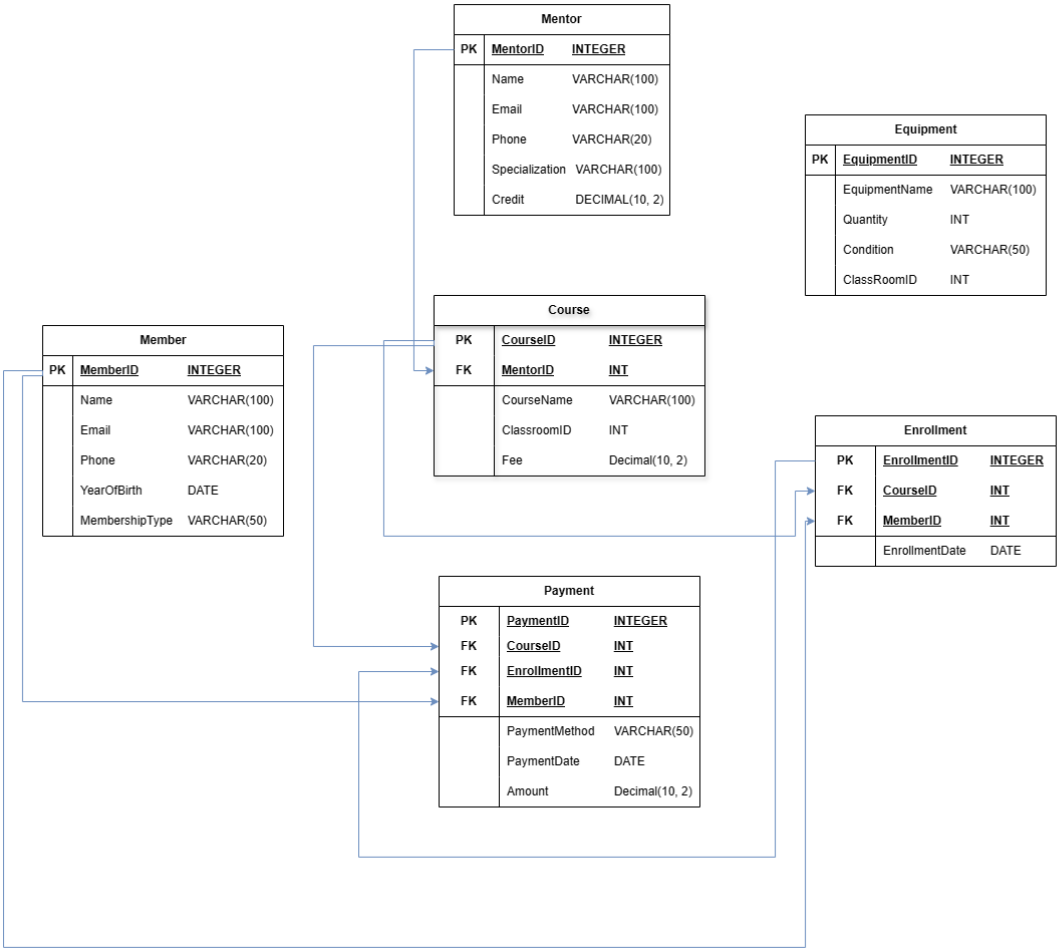

Figure 2: Logical model from the conceptual model

3 DATABASE IMPLEMENTATION

During the implementation of the Course Database, the following constraints and indices have been established to ensure data integrity and optimize query performance:

Member Table:

- **MemberID** is the primary key, and it is an auto-incrementing integer.
- **Name** is a required field (NOT NULL).
- **Email**, **Phone**, **YearOfBirth**, and **MembershipType** are optional fields.

Mentor Table:

- **MentorID** is the primary key, and it is an auto-incrementing integer.
- **Name** is a required field (NOT NULL).
- **Email**, **Phone**, **Specialization**, and **Credit** are optional fields.

Course Table:

- **CourseID** is the primary key, and it is an auto-incrementing integer.
- **CourseName** is a required field (NOT NULL).
- **MentorID** is a foreign key referencing **MentorID** in the Mentor table with CASCADE on DELETE and UPDATE.
- **ClassRoomID** is an optional field.
- **Fee** is a decimal field representing the course fee.

Enrollment Table:

- **EnrollmentID** is the primary key, and it is an auto-incrementing integer.
- **MemberID** is a foreign key referencing **MemberID** in the Member table with CASCADE on DELETE and UPDATE.
- **CourseID** is a foreign key referencing **CourseID** in the Course table with CASCADE on DELETE and UPDATE.
- **EnrollmentDate** is a date field.

Payment Table:

- **PaymentID** is the primary key, and it is an auto-incrementing integer.
- **CourseID**, **MemberID**, and **EnrollmentID** are foreign keys referencing their respective tables with CASCADE on DELETE and UPDATE.
- **PaymentDate** is a date field.
- **Amount** is a decimal field representing the payment amount.
- **PaymentMethod** is a string field representing the payment method.

Equipment Table:

- **EquipmentID** is the primary key, and it is an auto-incrementing integer.
- **EquipmentName** is a required field (NOT NULL).
- **Quantity** is an integer field.
- **Condition** is a string field.
- **ClassRoomID** is a foreign key referencing **ClassRoomID** in the Course table with CASCADE on DELETE and UPDATE.

In addition to these integrity constraints, two indices have been implemented:

1. An index on the **Course** table based on the **CourseName** field to facilitate quick searches for courses by name.
2. An index on the **Enrollment** table based on the **EnrollmentDate** field to optimize queries related to enrollment dates.

A Python interface has been developed to interact with the Course Database, providing functions for querying, inserting, updating, and deleting records. This interface abstracts the complexity of SQL queries and ensures a user-friendly experience for interacting with the database.

Note: in the database the MentorID is mentioned as InstructorID.