UNIVERSITY
LIBRARY MANAGEMENT SYSTEM
Assignment # 2

GROUP MEMBERS
Efrata Bayle
Muhammad Owais Suhail
Ahmed Abasimel

Database used:sowais_db

Mapping ER Diagram to Relational Model for University Library Management System

For the **University Library Management System**, we adopt an approach to convert **entities**, **relationships**, and **attributes** into tables (relations). Below provides is a detailed explanation of how we map the system's elements and handle ISA hierarchies:

1. Mapping Regular Entities

Each entity in the ER diagram, such as **Author**, **Publisher**, **Vendor**, **Library**, **Admin**, **Employee**, and **Member**, is mapped directly into a table. Each table has a **primary key** that uniquely identifies its rows (tuples). The entity's attributes become columns in the corresponding table.

Example:
The `Author` entity with attributes `author_code`, `author_name`, and `author_subject` is mapped to a relation as:

```
Author(author_code, author_name, author_subject)
```

Here, `author_code` is the **primary key**.

2. Mapping Relationships

Depending on the cardinality and participation constraints, relationships are mapped differently:

One-to-Many (1 : M) Relationships:
In a 1 : M relationship, a **foreign key** is placed in the table on the "many" side, referencing the primary key of the "one" side.
Example:
The `Admin` entity has a one-to-many relationship with `Employees`. Thus, `admin_id` is added as a foreign key in the `Employees` table:

```
Employees(admin_id) REFERENCES Admin(admin_id)
```

Many-to-One (M:1) Relationships:

These relationships are mapped similarly to 1 : M, where a foreign key from the "one" side is placed in the "many" side.

Example:

The `Books` table has a many-to-one relationship with the `Library`. Therefore, `library_code` is added to `Books` as a foreign key:

```
Books(library_code) REFERENCES Library(library_code)
```

Many-to-Many (M: M) Relationships:

M : M relationships are represented using a **join table** that contains foreign keys referencing both related entities.

Example:

The `Employee_Member` table is created to handle the many-to-many relationship between `Employees` and `Members`:

```
Employee_Member(employee_id, member_id)
```

One-to-One (1:1) Relationships:

A 1:1 relationship is implemented by placing a **foreign key** in either table, usually where the entity must always exist.

Example:

The `Admin` manages the `Library`, so `admin_id` is added to the `Library` table as a foreign key with a unique constraint:

```
Library(admin_id UNIQUE) REFERENCES Admin(admin_id)
```

3. Handling ISA Hierarchies

ISA hierarchies involve inheritance-like relationships where a parent entity has sub-entities with distinct attributes.

- **Approach 1: Single Table Inheritance (All in One Table)**:
  One option is to store all attributes, including those specific to sub-entities, in a single table with a "type" column to distinguish between sub-entities.

- **Why We Didn't Use This Approach**:
  This approach can lead to **redundancy** and **inefficiencies**, as many NULL values would appear in rows for attributes that don't apply to certain subtypes (e.g., `contract_id` for a student assistant).

Approach 2: Class Table Inheritance (One Table per Class):
This method creates separate tables for each subclass, with each subclass table holding only its specific attributes. The subclass tables use a foreign key to reference the base class.

Why We Used This Approach:
In our system, the `Employees` table is divided into **Librarian** and **StudentAssistant**, and `Members` is divided into **StudentMember** and **FacultyMember**. Each subclass has distinct attributes, like `contract_id` for Librarian and `hourly_wage` for StudentAssistant. This avoids redundancy and preserves the hierarchical structure.

Example:
The `Employee` table holds common attributes (`employee_id`, `employee_name`, etc.). The `Librarian` and `StudentAssistant` tables hold their specific attributes (`contract_id` and `hourly_wage`) and reference `Employee`:

```
Librarian(employee_id PRIMARY KEY, contract_id)
StudentAssistant(employee_id PRIMARY KEY, hourly_wage)
```

4. Integrity Constraints

While mapping the ER diagram to relations, we enforce the following integrity constraints:

- **Primary Keys**: Uniquely identify each row in a table.
- **Foreign Keys**: Ensure referential integrity between related tables.

SQL Table Creation and Insertion Examples

The following code below is just 1 example of creating a table using SQL Queries. The rest of the SQL Queries you can find in the following link SQL Queries.

```
-- Example for Author Table
CREATE TABLE Author (
    author_code INT PRIMARY KEY,
    author_name VARCHAR(255),
    author_subject VARCHAR(255)
);
```

Conclusion

This approach of mapping the ER diagram into a relational model ensures that the relationships, ISA hierarchies, and constraints are implemented efficiently and maintain data integrity. Each entity is represented by a table, relationships are managed using foreign keys, and inheritance is handled with class table inheritance.

As we didn't get the access to the new MySQL server team credentials, we have used **sowais_db** database to do the tasks.

The Following is the link to the Updated ER Diagram:-

Updated ER diagram