Design and implement in MySQL (or other RDBMS) a relational database system that models a problem of your choice.

here is an example (**don't do this one**): design a university's course enrollment process, with full support for bitemporal data—tracking both when data is entered into the system (transaction time) and when it is considered valid in the real world (valid time).

Requirements

1. Database Design Define a schema that includes entities such as:

- Students
- Courses
- Enrollments
- Professors
- Departments

Include temporal attributes:

- transaction_start, transaction_end (for transaction time)
- valid_start, valid_end (for valid time) Use appropriate data types (e.g., TIMESTAMP, DATE) and constraints.

2. Data Population Populate the database with at least:

- 50 students
- 5 professors
- 6 courses
- 60 enrollment records

Ensure that enrollment records reflect realistic changes over time (e.g., students dropping or re-enrolling in courses, courses changing instructors).

3. Temporal Querying Write and execute SQL queries (better create views) that demonstrate:

- Retrieving the current valid enrollments for a given student.
- Retrieving the history of changes to a course's instructor.
- Identifying conflicts between transaction time and valid time (e.g., late data entry).
- Showing all enrollments that were valid during a specific semester.
- Retrieving records as they were known at a specific transaction time (e.g., "What did the system know on March 1st?").

4. Documentation

- An ER diagram of the schema.
- SQL scripts for table creation and data insertion. A brief report explaining:
- The approach to modeling bitemporal data.
- Challenges faced and how they were addressed.
- Insights gained about temporal databases.

For the highest grade (1 point each):

- Implement triggers to automatically manage transaction time.
- Implement the solution using temporal extensions in PostgreSQL or Oracle (e.g., PERIOD FOR syntax).

Deadline: 3rd seminar