School Management System

by Fatma Alfadhli 77433
Sara Shamouh 70158
Maram Alotaibi 69543

Introduction and Goals Explanation

School Management System Database (SMS Database)

SMS is a comprehensive database application designed to streamline and enhance the administrative and academic management of educational institutions. The core of the SMS is a relational database that integrates all facets of school operations, from student and teacher management to class scheduling and event organization.

Designing Steps & Implementations

Step 1: Understanding Requirements

Primary Tables Identified:

- Student
- Class
- Teacher
- Coordination
- Event

Analysis:

- Information required for each table
- Assumptions about data organization and representation

Example: Student Table:

- Attributes: names, IDs, grades, age, contact information, addresses

Design Considerations:

- Composite attributes
- Multi-valued attributes
- Weak entities
- Balancing efficiency and complexity

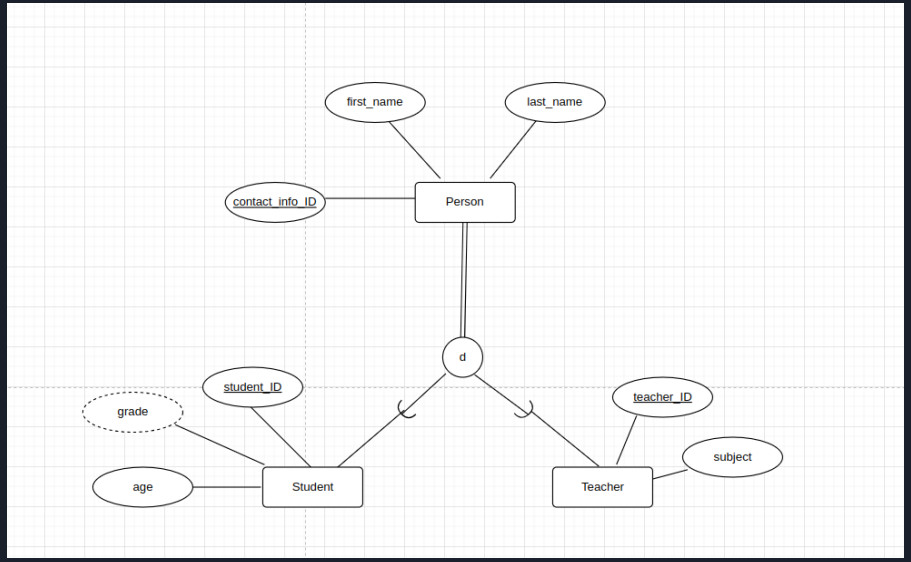
Step 2: EER Diagram

1. Identifying major strong entities:
- Student
- Class
- Teacher
- Coordination (represented as a coordinate relationship)
- Event

Step 2: EER Diagram

Supertypes & Generalization

Noticing that Teachers and Students share similar attributes (first name, last name, contact info), we created a supertype entity, Person, to assign these shared attributes. In our system, a Person must be either a Teacher or a Student (total participation) and cannot be both (disjoint).

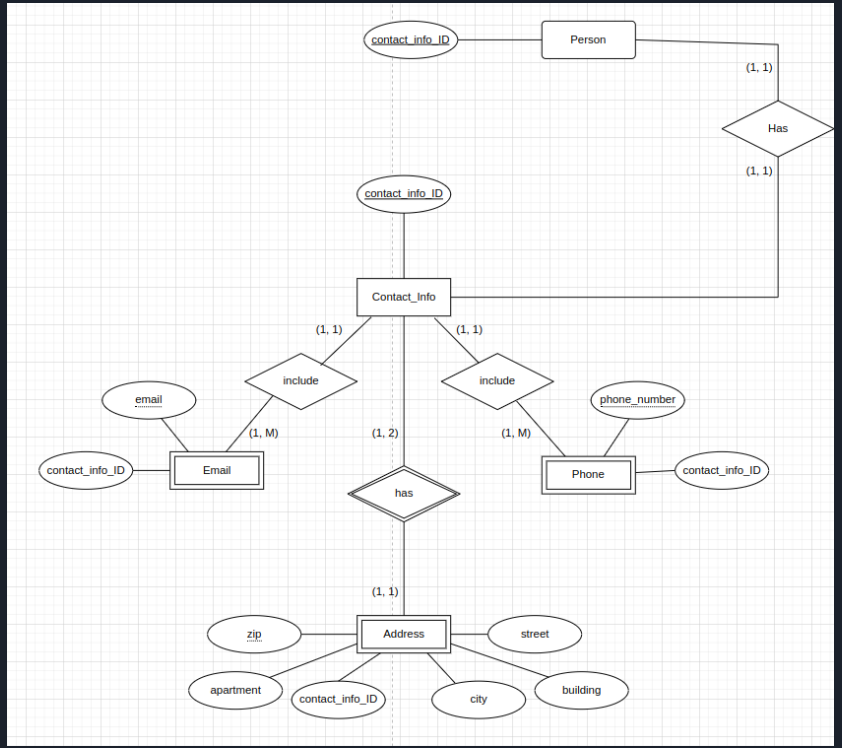


Step 2: EER Diagram

Weak Entities

Users, including teachers and students, are enabled to register multiple contact details such as emails, phones, and addresses. Support is provided for multiple emails and phones per user, and shared addresses, such as parents living in the same house, are allowed. These assumptions are reflected in the EER diagram, showing entities, relationships, and cardinalities, and were later translated into relational schema constraints.

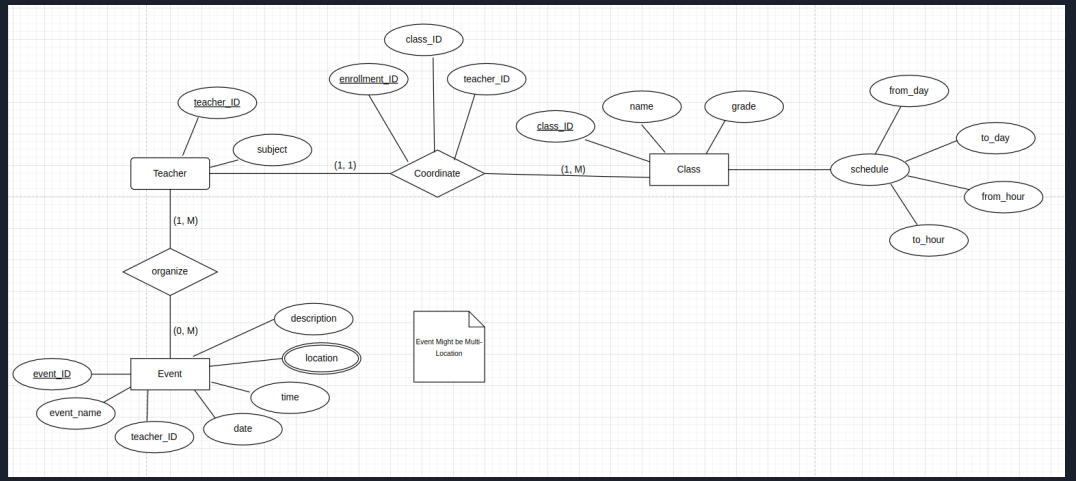
Each user is assigned a unique contact ID to which their contact details are attached. The Contact_Info entity is designed with identifying relationships to weak entities (Email, Phone, Address), which cannot exist independently.



Step 2: EER Diagram continued.

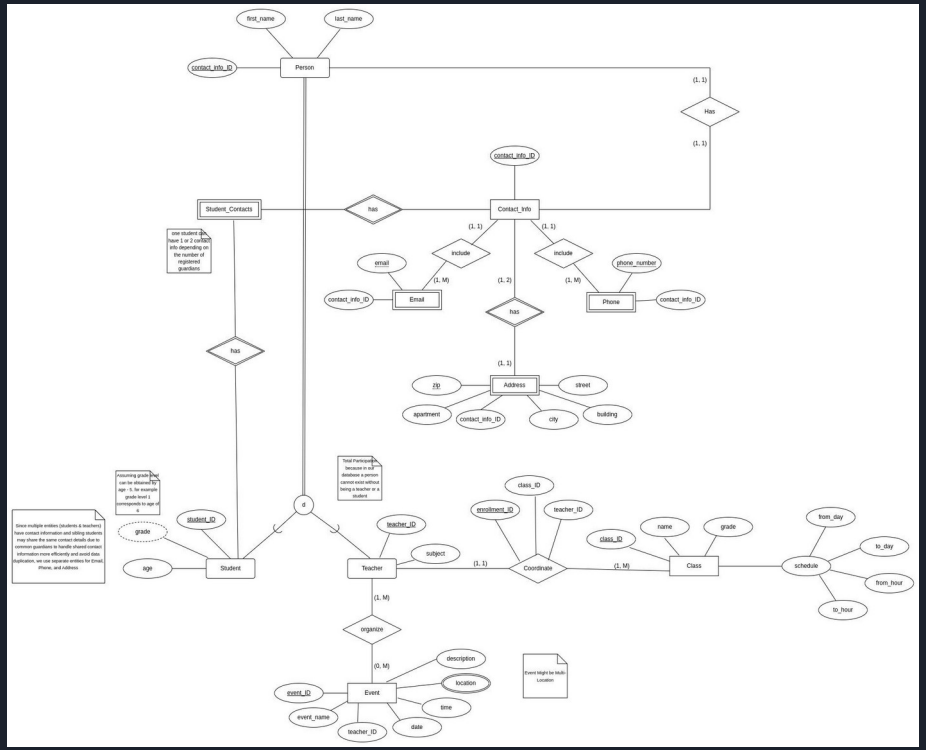
The Class entity was created and related to Teacher through the Coordinate relationship, which was later translated into the Coordination table in the relational schema. The cardinalities indicate that one teacher can teach multiple classes, while one class is taught by one teacher. The schedule attribute of Class, a composite attribute, will be flattened in the relational schema.

The Event entity was created and related to Teacher through the Organize relationship. The location is multi-valued, as one event can be held in multiple halls simultaneously, assuming events can be multi-location.



Step 2: EER Diagram

Connecting Pieces Together.

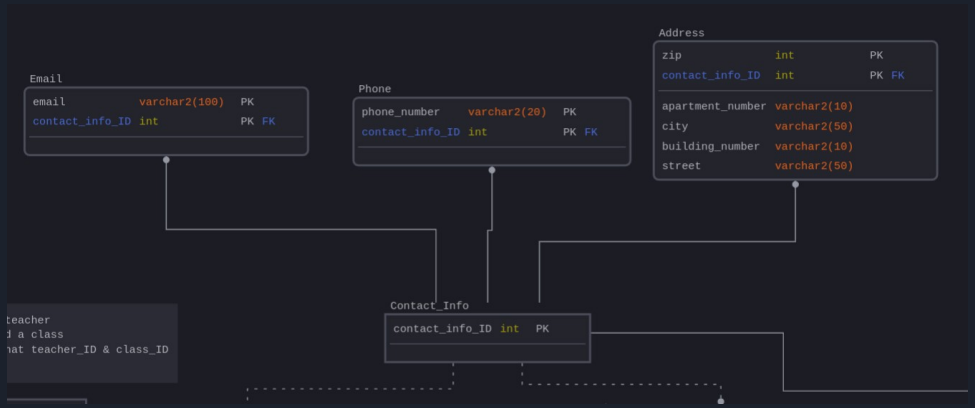


Step 3: Relational Schema

We employed conventional methods for translating EER models into relational schemas, including entity mappings to tables, relationships corresponding to new tables or columns, flattening composite attributes, and appropriately setting primary and foreign keys, along with constraints.

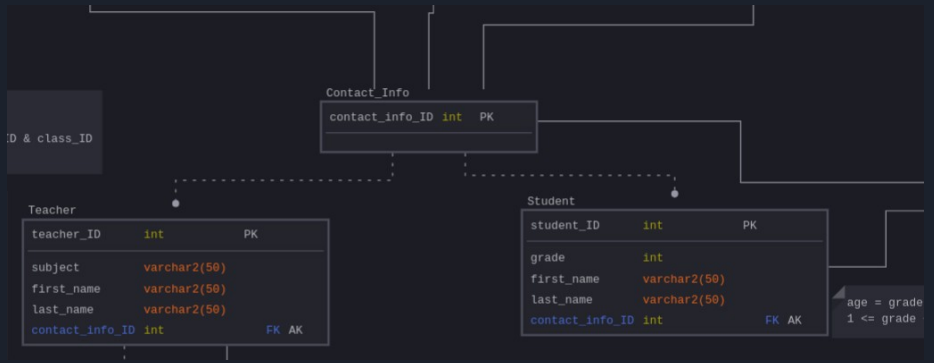
Step 3: Relational Schema

We created the Contact_Info table with one attribute as the primary key. From this, we established three identifying relationships to the Email, Phone, and Address tables by setting each table's primary key as a combination of the Contact_Info primary key and the corresponding partial keys.



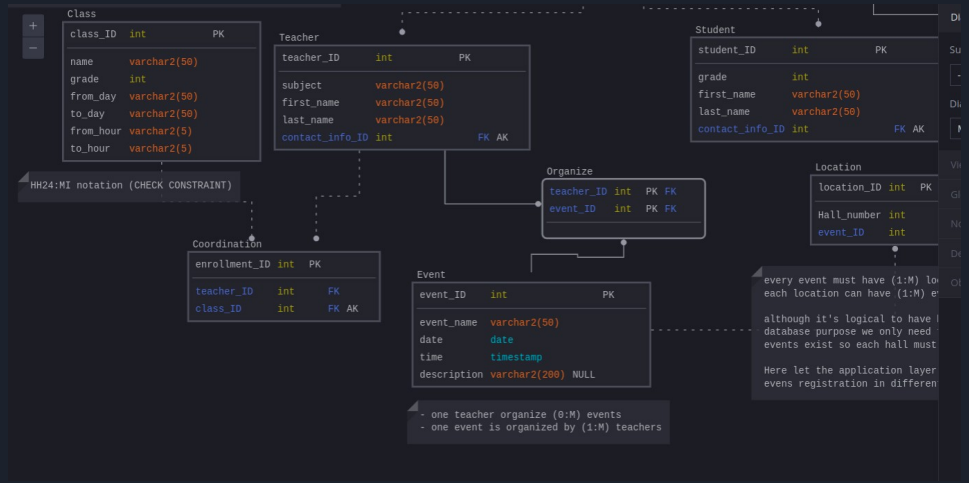
Step 3: Relational Schema

The Teacher and Student tables have non-identifying relationships with the Contact_Info table, referencing the Contact_Info primary key using foreign keys. Additionally, the Student_Contacts table depends on both Contact_Info and Student, allowing two different contact IDs for the same student (e.g., if a student has more than one guardian). Since age is a derived attribute (assumed to be derived from grade), we did not explicitly create a column for it.



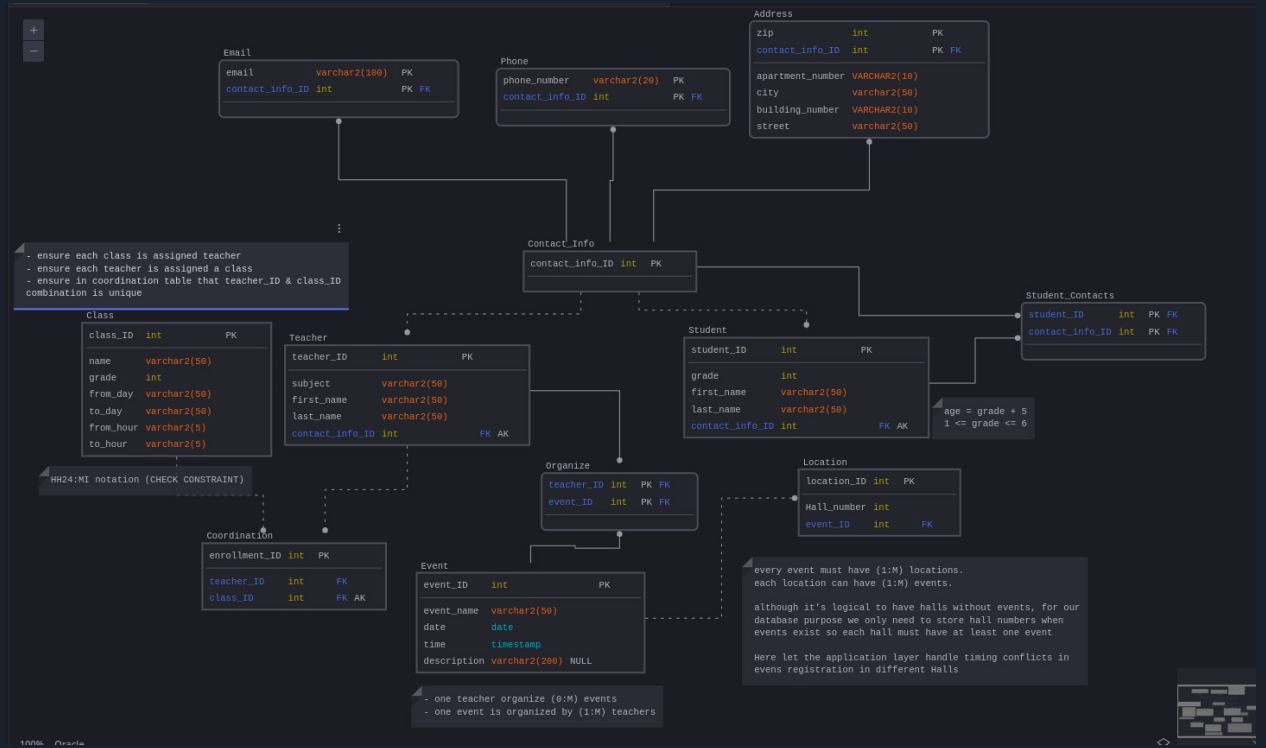
Step 3: Relational Schema

The Class and Coordination tables manage the assignment of different teachers to classes. In the Coordination table, class_ID is an alternate key with a unique constraint, ensuring that one class can only be taught by one teacher.
Lastly, we created the Event, Organize, and Location tables. The Organize table tracks which teacher is responsible for which event, with its primary key being a combination of teacher_ID and event_ID to avoid duplication. The Location table accommodates the assignment of one event to multiple locations simultaneously.



Step 3: Relational Schema

Connecting Pieces Together.



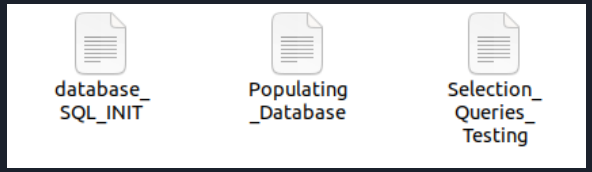
SQL Translation and Database Testing

We translated this relational schema into SQL code, populated our database with information, and tested it using the following three queries:

- Retrieve all teachers and their contact information:
- Find all events organized by each teacher:
- Count the number of students with more than one contact information entry:

The creation, population, and testing of the database is divided into 3 different files in the code folder.

- database_SQL_INIT (Tables initialization)
- Populating_Database (Populating the tables with data)
- Selection_Queries_Testing (Testing)



SQL Translation and Database Testing

Results and database demonstration are inside the screenshots pdf file.

Queries Resutls:

First SELECT Query:

TEACHER_ID	FIRST_NAME	LAST_NAME	SUBJECT	CONTACT_INFO_ID	EMAIL	PHONE_NUMBER	STREET	CITY	ZIP
77433	Fatma	Alfadhli	Mathematics	1	john.doe@example.com	1234567890	123 Main St	Springfield	12345
70158	Sara	Shamouh	English	2	jane.smith@example.com	9876543210	456 Oak Ave	Rivertown	54321
69543	Maram	Alotaibi	Geography	5	-	-	-	-	-

Note: There is no constraint enforcing that each `contact_ID` must have all details filled. However, there is a constraint enforcing the opposite: all contact details for any record must be indexed by a `contact_ID`. If we wish to ensure that contact information is provided for each person, it would be more effective to implement this requirement at the application layer by preventing users from submitting forms unless all required contact information is completed.

Second SELECT Query:

TEACHER_ID	FIRST_NAME	LAST_NAME	EVENT_ID	EVENT_DATE	EVENT_TIME	DESCRIPTION
77433	Fatma	Alfadhli	1	15-JUL-24	14:00	School Open House
70158	Sara	Shamouh	2	01-AUG-24	10:00	Parent-Teacher Conference
77433	Fatma	Alfadhli	3	15-SEP-24	15:00	Science Fair
77433	Fatma	Alfadhli	4	01-OCT-24	09:30	Annual Sports Day

Third SELECT Query

NUM_STUDENTS_WITH_MULTIPLE_CONTACTS
2

Conclusion & Reflections

Learning Outcomes

1. Database Requirements: Extracted key requirements, identified necessary tables and attributes, and made assumptions for a comprehensive design.
2. EER Model Design: Identified major entities and relationships, handled shared, composite, and multi-valued attributes, and incorporated these into the EER model.
3. Normalization and Schema Translation: Applied normalization principles, established primary and foreign keys, and ensured referential integrity in the relational schema.
4. SQL Implementation and Testing: Translated the schema into SQL, populated the database, and verified its integrity through specific queries.

Concluding Remarks

This project enhanced our skills in database design and implementation. We moved from high-level requirements to a detailed EER model, translated it into a relational schema, and successfully implemented and tested the database. The process emphasized careful planning, efficient design, and thorough testing, ensuring a robust and scalable system. This experience has provided a solid foundation for tackling future database challenges.