Introduction & Goals

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. Further elaboration on SMS's main functionality is found in the technial report.

Understanding the Database Requirements

From the project description, we identified five primary tables necessary for our Primary School Management System: Student, Class, Teacher, Coordination, and Event. We then analyzed the information each table should record and made assumptions about the organization and representation of this information. For example, the Student table includes students' names, IDs, grades, age, contact information, and addresses. We assumed that contact information and addresses are not single-valued but composed of multiple parts. This led us to consider composite attributes, multi-valued attributes, and weak entities, balancing efficiency and complexity in our design.

EER Model Design

Our process for designing the EER model involved the following steps:

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

2. 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).

3. Our system allows users (Teachers or Students) to register contact information, including email, phone, and addresses. Each user can have multiple emails and phones. Different users can share the same address, for example, if two guardians/parents of a student live in the same house. These assumptions were incorporated into our EER diagram as entities, relationships, and their cardinalities, and later translated into corresponding constraints in our relational schema.

4. Each user has exactly one contact ID, to which contact details are attached. We designed the Contact_Info entity to have an identifying relationship with the weak entities (Email, Phone, and Address). These entities cannot exist independently without the Contact_Info entity. While an address might exist independently in other contexts, within our school management system, it is represented as a weak entity. The cardinalities in the relationships encode our assumptions; for example, one contact ID can include multiple phone numbers. This separation ensured normalization and avoidance of data duplication.

5. We created the Class entity, which relates to Teacher through the Coordinate relationship, later translated into the Coordination table in the relational schema. The cardinalities show 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 when translated into the relational schema.

6. Finally, we created the Event entity, which relates to Teacher through the Organize relationship. The location is multi-valued, as one event can be held in multiple halls simultaneously. We assumed that events could be multi-location.

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.

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.

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.

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.

Constraints:
Apart from foreign key constraints ensuring referential integirty illustrated in the relational schema screenshot we also set up the following:

- Grade must be between 1 and 6
- Time must be entered in HH24:MI Oracle format
- Unique Constraint on the class_ID in the coordination table.

We also chose appropriate data types for all attributes.

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:

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

Results and database demonstration are inside the screenshots pdf file.

Reflection on Learning Outcomes and Concluding Remarks

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.