



Project 1 - TinyHub

11.30.2020

Vamsi Krishna Velivela - vvelivel

Suhash Bollu - suhashbo

ER-Schema:

The requirements in the universe of discourse are analysed and converted to an ER Model. Aggregation is used extensively in the mapping in order to have a relation with a relationship entity.

User: It is an entity set with attributes *user_id*, *email_id*, *display_name* and *password* where the key attribute is *user_id* and *email_id* can also be used as a unique identifier.

Student: It is a specialization of **User** and maintains ISA relationship with **User** by having a key attribute *student_id* which references the *user_id* in **User** entity set. It participates in the following relations.

- many-to-many relationship(students pursue programs) with the **Program**.
- many-to-many relationship(students enroll in courses) with the **Course**.
- many-to-many relationship(students register in semesters) with the **Semester**.

Professor: It is a specialization of **User** and maintains ISA relationship with **User** by having a key attribute *professor_id* which references the *user_id* in **User** entity set. It participates in the following relations.

- many-to-one relationship(professors work for department) with the **Department**.
- one-to-many(professors teach courses) relationship with the **Course**.

Staff: It is a specialization of **User** and maintains ISA relationship with **User** by having a key attribute *staff_id* which references the *user_id* in **User** entity set. It participates in the following relations.

- many-to-one relationship with the **Department**.

Program: It is an entity set with attributes *program_id*, *program_name*, *department_id*. It participates in the following relations.

- many-to-one relationship with the **Department**.
- many-to-many relationship with the **Student**.

Department: It is an entity set with attributes *department_id*, *department_name*. It participates in the following relations.

- one-to-many relationship with the **Professor**.
- one-to-many relationship with the **Staff**.
- one-to-many relationship with the **Student**.
- one-to-many relationship with the **Program**.
- one-to-many relationship with the **Course**.

Course: It is an entity set with attributes *course_id*, *course_name*. It participates in the following relations.

- many-to-one relationship with the **Department**.
- many-to-many relationship with the **Student**.
- many-to-many relationship with the **Semester**.
- many-to-one relationship with the **Professor**.
- one-to-many relationship with the **Course_session**.

Course_session: It is an entity set with attributes *session_id*, *course_id*, *semester_id*, *capacity*. It participates in the following relations.

- many-to-one relationship with the **Course**.

Semester: It is an entity set with attributes *semester_year*, *semester_id*, *season*. It participates in the following relations.


- many-to-many relationship with the **Course**.
- many-to-many relationship with the **Student**.

Exam: It is an entity set with attributes *exam_id*, *course_id*. It participates in the following relations.

- many-to-one relationship with the **Course**.
- one-to-many relationship with the **Problem**.
- many-to-many relationship with the **Student**.

Problem: It is an entity set with attributes *problem_id*, *course_id*, *description*. It participates in the following relations.

- many-to-one relationship with the **Exam**.
- many-to-many relationship with the **Student**.



Book: It is an entity set with attributes *book_isbn*, *author_id*, *page_count*, *publication_date*, *title*. It participates in the following relations.

- one-to-many relationship with the **Book_unit**.
- many-to-many relationship with the **Author**.

Author: It is an entity set with attributes *author_id*, *author_name*. It participates in the following relations.

- many-to-many relationship with the **Book**.

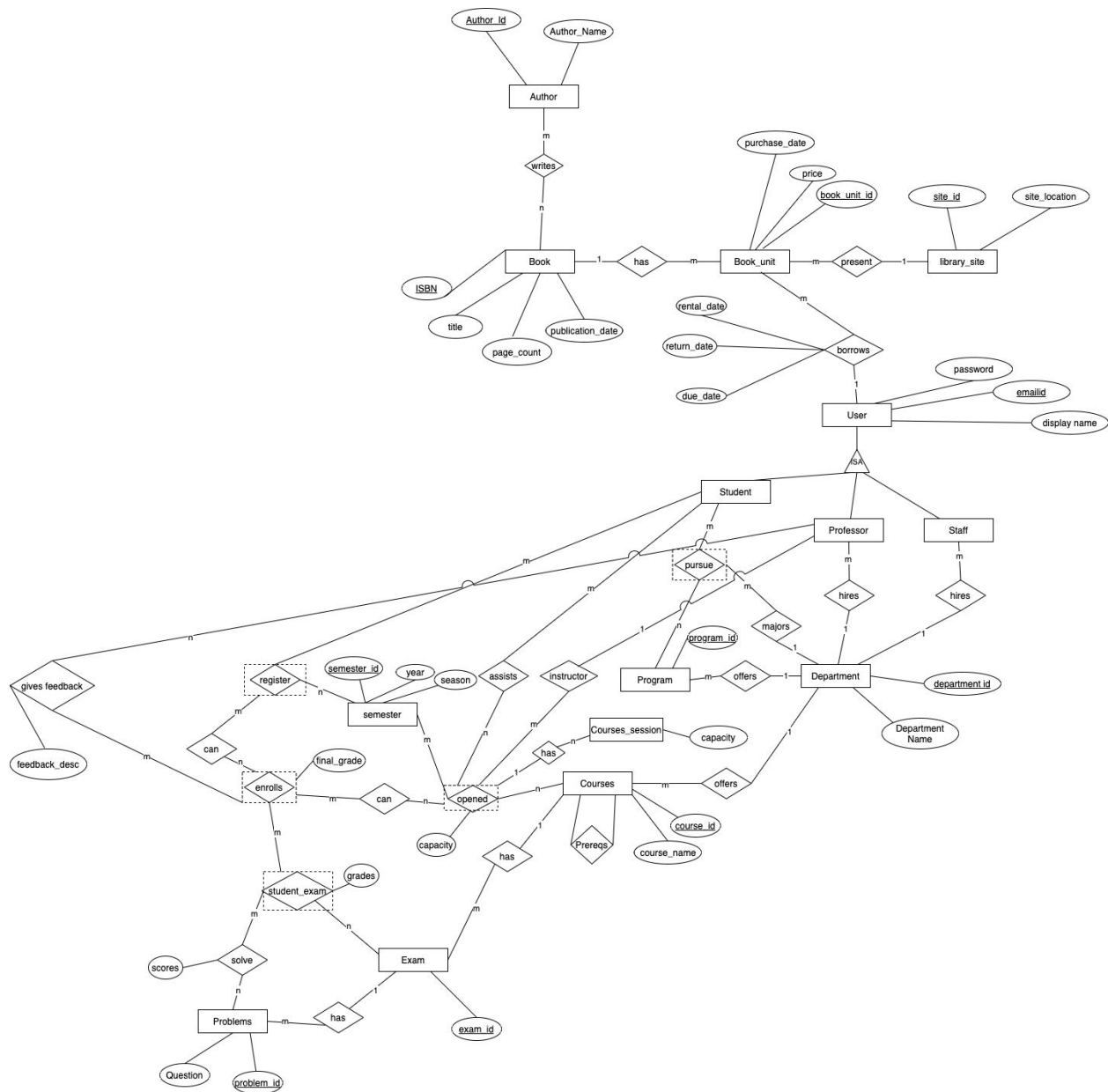
Book_unit: It is an entity set with attributes *book_unit_id*, *price*, *purchase_date*, *book_isbn*, *site_id*, *user_id*, *rental_date*, *due_date*, *return_date*. It participates in the following relations.

- many-to-one relationship with the **Book**.
- many-to-one relationship with the **User**.
- many-to-one relationship with the **Library_site**.

Library_site: It is an entity set with attributes *site_id*, *site_name*. It participates in the following relations.

- one-to-many relationship with the **Book_unit**.

The following diagram depicts the Entity-Relation Schema for the universe of discourse given in project 1.



Relational Database Schema:

The above ER Model is converted to Relational schema using the following rules:

- Each entity set is turned into a relation.

ISA Hierarchies

ISA hierarchies are handled using ER-style conversion strategy. For every entity E, the key attributes from the root entity are included, all other attributes of E are added.

Combining Relations

- For every many-to-one relationship or one-to-many relationship, the attributes of the relation are moved to many side.
- For every many-to-many relationship, all the attributes of a relationship set is translated into a new relation.

User Management:

User(user_id, email_id, password, display_name)

PK(User.user_id)

Student(student_id)

PK(Student.sid)

FK(Student.sid, User.user_id)

Professor(professor_id, department_id)

PK(Professor.pid)

FK(Professor.professor_id, User.user_id)

FK(Professor.department_id, Department.department_id)

Staff(staff_id, department_id)

PK(Staff.staff_id)

FK(Staff.staff_id, User.user_id)

FK(Staff.department_id, Department.department_id)

Department Management:

Program(program_id, program_name, department_id)

PK(Program.program_id)

FK(Program.department_id, Department.department_id)

Department(department_id, department_name)

PK(Department.department_id)

Course Management:

Course(course_id, course_name, department_id)

PK(Course.course_id)

FK(Course.department_id, Department.department_id)

Prerequisite(pre_id, course_id)

PK(Prerequisite.pre_id, Prerequisite.course_id)

FK(Prerequisite.pre_id, Course.course_id)

Semester(semseter_year, semester_id, season)

PK(Semester.semester_id)

Open_courses(course_id, capacity, professor_id, semester_id)

PK(Open_courses.course_id, Open_courses.semester_id)

FK(Open_courses.course_id, Course.course_id)

FK(Open_courses.professor_id, Professor.professor_id)

FK(Open_courses.semester_id, Semester.semester_id)

Student_Program(sid, program_id)

PK(Student_Program.sid, Student_Program.program_id)

FK(Student_Program.sid, Student.sid)

FK(Student_Program.program_id, Program.program_id)

Course_Session(session_id, capacity, course_id, semester_id)

PK(Course_Session.session_id)

FK(Course_Session.course_id, Open_Courses.course_id)

FK(Course_Session.semester_id, Open_Courses.semester_id)

teaching_assistant(student_id, course_id, semester_id)

PK(teaching_assistant.student_id)

FK(teaching_assistant.course_id, Open_Courses.course_id)

FK(teaching_assistant.semester_id, Open_Courses.semester_id)

registered_student(student_id, semester_id)

PK(registered_student.student_id, registered_student.semester_id)

FK(registered_student.student_id, Student.student_id)

FK(registered_student.semester_id, Semester.semester_id)

Student-Course Management:

course_enrolled_students(student_id, course_id, semester_id, final_grade)

PK(course_enrolled_students.student_id, course_enrolled_students.course_id,
course_enrolled_students.semester_id)

FK(course_enrolled_students.student_id, registered_student.student_id)

FK(course_enrolled_students.course_id, Open_Courses.course_id)

FK(course_enrolled_students.semester_id, registered_student.semester_id)



Exam(exam_id, course_id)

PK(Exam.exam_id)

FK(Exam.course_id, Course.course_id)

Problem(problem_id, description, exam_id)

PK(Problem.problem_id)

FK(Problem.exam_id, Exam.exam_id)

student_exam(student_id, course_id, exam_id, semester_id, grade)

PK(student_exam.student_id, student_exam.course_id, student_exam.exam_id,
student_exam.semester_id)

FK(student_exam.student_id, course_enrolled_students.student_id)

FK(student_exam.course_id, course_enrolled_students.course_id)

FK(student_exam.semester_id, course_enrolled_students.semester_id)

FK(student_exam.exam_id, Exam.exam_id)

student_problem(student_id, course_id, semester_id, email_id, problem_id)

PK(student_problem.student_id, student_problem.course_id,
student_problem.semester_id, student_problem.email_id, student_problem.problem_id)

FK(student_problem.student_id, student_exam.student_id)

FK(student_problem.course_id, student_exam.course_id)

FK(student_problem.semester_id, student_exam.semester_id)

FK(student_problem.email_id, student_exam.email_id)

FK(student_problem.problem_id, Problem.problem_id)

feedback(course_id, student_id, semester_id, professor_id)

PK(feedback.course_id, feedback.student_id, feedback.semester_id, feedback.professor_id)

FK(feedback.course_id, course_enrolled_students.course_id)

FK(feedback.semester_id, course_enrolled_students.semester_id)

FK(feedback.student_id, course_enrolled_students.email_id)

FK(feedback.professor_id, Professor.professor_id)

Library Management:

Book(isbn, title, page_count, publication_date)

PK(Book.isbn)

Author(author_id, author_name)

PK(Author.author_id)

Book_Author(book_isbn, author_id)

PK(Book_Author.book_isbn, Book_Author.author_id)

FK(Book_Author.book_isbn, Book.isbn)

FK(Book_Author.author_id, Author.author_id)

library_site(site_id, site_name)

PK(site_name.site_id)

book_unit(book_id, price, purchase_date, book_isbn, site_id, user_id, rental_date, due_date, return_date)

PK(book_unit.book_id)

FK(book_unit.book_isbn, Book.isbn)

FK(book_unit.site_id, library_site.site_id)


FK(book_unit.user_id, User.user_id)

Further discussion:

Advantages:

1. We used aggregation instead of connecting more than 2 tables with a single relationship. It leads to easy updation of tables and avoids unnecessary actions.
2. In 1-M relationships, we stored the reference of 1 side in the M side table without creating an additional relation-table.

Example:- We stored the department_id in the Staff table without creating an additional relation table.

- 
3. Multiple aggregations are used to keep the schema more dynamic and less tightly coupled.
 4. ISA-hierarchy is used to reduce the data redundancy and increase the ease of access.
 5. We made sure in our design that any update in one table is least affected on other tables.
 6. We used Integrity constraints like ON DELETE CASCADE to make sure that when a row is deleted in one table, corresponding rows in the referenced table are also deleted.

Disadvantages:

1. The aggregation takes extra tables, hence takes additional space.
2. Storing the 1-M relationships on the M side is tight coupling.