Jama Jama
11/27/2020
CSE460
Professor Chuang

Report for TinyHub

User management

First before creating anything, a user account must be created. A user account has four attributes; user_account_id as the primary key, optional unique display_name of type string, unique mandatory email of type string, and a mandatory password of type string. Based on how I developed this database, an account can be of many types which includes students, staff, and professor. There is a table called account_type which holds account_type_id and unique account_type. This table ensures that there can be different unique accounts, but it is not limited to 3. These account types must be created manually by the university database administrators. This is because when an account is made, they must specify the account type they are creating using the id. This ensures that no account is made without a proper designation and assignment.

Department management

In the department table, each department is identified by a unique department id number called dept_id. A department can hire many professors and many staff. The table for Professors and staff have a mandatory attribute dept_id (foreign key). This ensures professors and staffs aren't hired without a department. A department can have different programs in a one to many relationship.

There is a constraint on the programs table program_name_dept_id which stipulates that program name and dept_id should be unique. This ensures that different departments can not have the same programs. Students are connected to department through the major table. A student can major in many departments and pursue many programs. This is captured in the major table which has dept_id and program_id. A student is limited to pursue a program that one is majoring in. This is ensured by having a mandatory department and program number in the major table. The disadvantage with this design is that we may need to delete multiple rows with program numbers from the table if a student no longer wants to major in a department.

Course management

In the course table, there is a foreign key dept_id which allows a department to offer multiple courses. This key is not null which ensures that a course is not created unless it is associated with a department. A course can be offered, but it does not have to be open in every semester. This can be found in the open_course table which holds all the open courses in a semester. Open_course table is connected to semester table and course table and has two mandatory foreign keys for course_id and semester_id which ensures that a course is open.

There is also a sessions number in open_course to indicate which a session for similar courses. This ensures there can be multiple opened courses from the same course in a semester. Open_course is also connected to the student table through a foreign key attribute TA which assigns a TA to the course. The professor id number can be found by joining the open_course table with the instruct table. There is a max attribute for student capacity found in course table. Also I have a connection between course table and course_prerequisite table which holds all the tuples of course of id's with this kind of prerequisite relationship.

Student-Course management

In the enroll table, a student is allowed to enroll in any open courses as long as they are registered in a semester. This is ensured by the course enroll table being connected to registration and open course table; and open course table is connected to semester to check for the semester using the semester id. The student must first register in a semester and then enroll. A student is not allowed to enroll for a course in which a prerequisite is not met. This is insured by the course prerequisite table which holds a tuple of courses with this kind of relationship. Also course enroll table contains all the courses a student has enrolled in throughout his/her stay at the university. So a check can be made for the grades a student received for any course to ensure that he has passed prerequisite courses.

For ensuring that a student doesn't enroll in a course that is at capacity and student is only taking courses that is being offered by a department they are majoring in, some backend/frontend logic would have to be inserted in order to make that decision. Before inserting data into course enroll, one must check if the course is maxed out using the attribute max_student_num in open course table and check if the student is majoring in the department by joining the open course table with major and department table. A grade can be given at the end of the semester in the course enroll table using the attribute grade.

Students are also allowed to give feed back on courses they are enrolled in using the feedback attribute in the course enroll table. In the exam table, there can be a list of exams with different ids and since it is connected to the enroll_student_exam table, a student can receive letter grade using the attribute grade. Enroll_student_exam holds all the exam ids and grades for all the exams a student has taken. Also in the answer table holds all the scores for every problem in the exam.

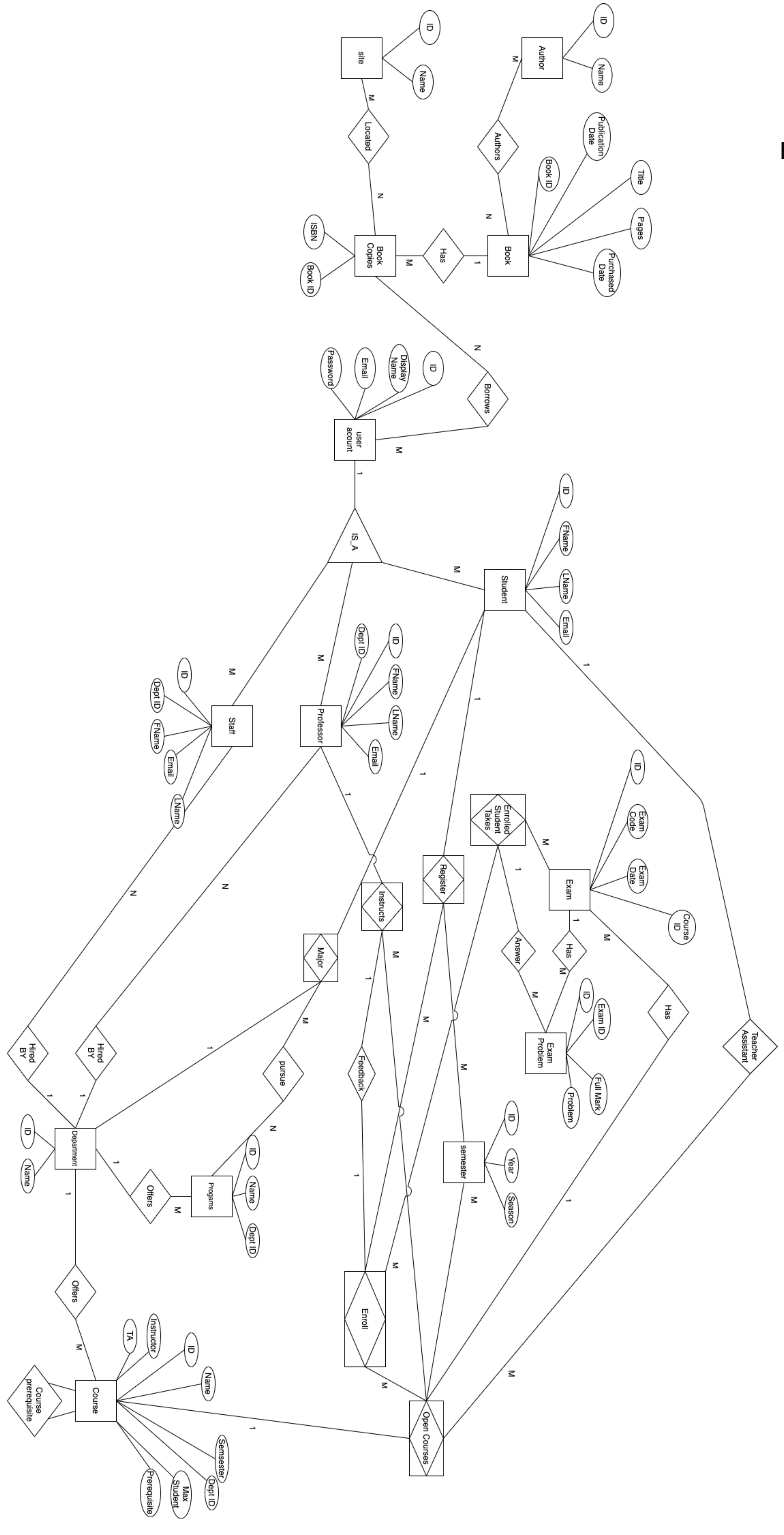
Library management

There are two tables for books. One is for keeping the metadata such as book id, title, ISBN, publication_date, number_of_pages, and purchase_date for all the books and other is for the copies of a book. Since I was unsure if ISBN numbers weren't unique for every copy, in the book_copy table, there is an identifier for all books called book_copy_id which is a primary key. This ensure that every single copy of a book has a unique ID number. I've also included an attribute I have a separate table called for book_authors which holds tuples of book IDs and their author IDs. This table is connected to the authors table which holds the name of the author in the attribute author_name. The book_author table ensures that we can accommodate multiple authors of a book.

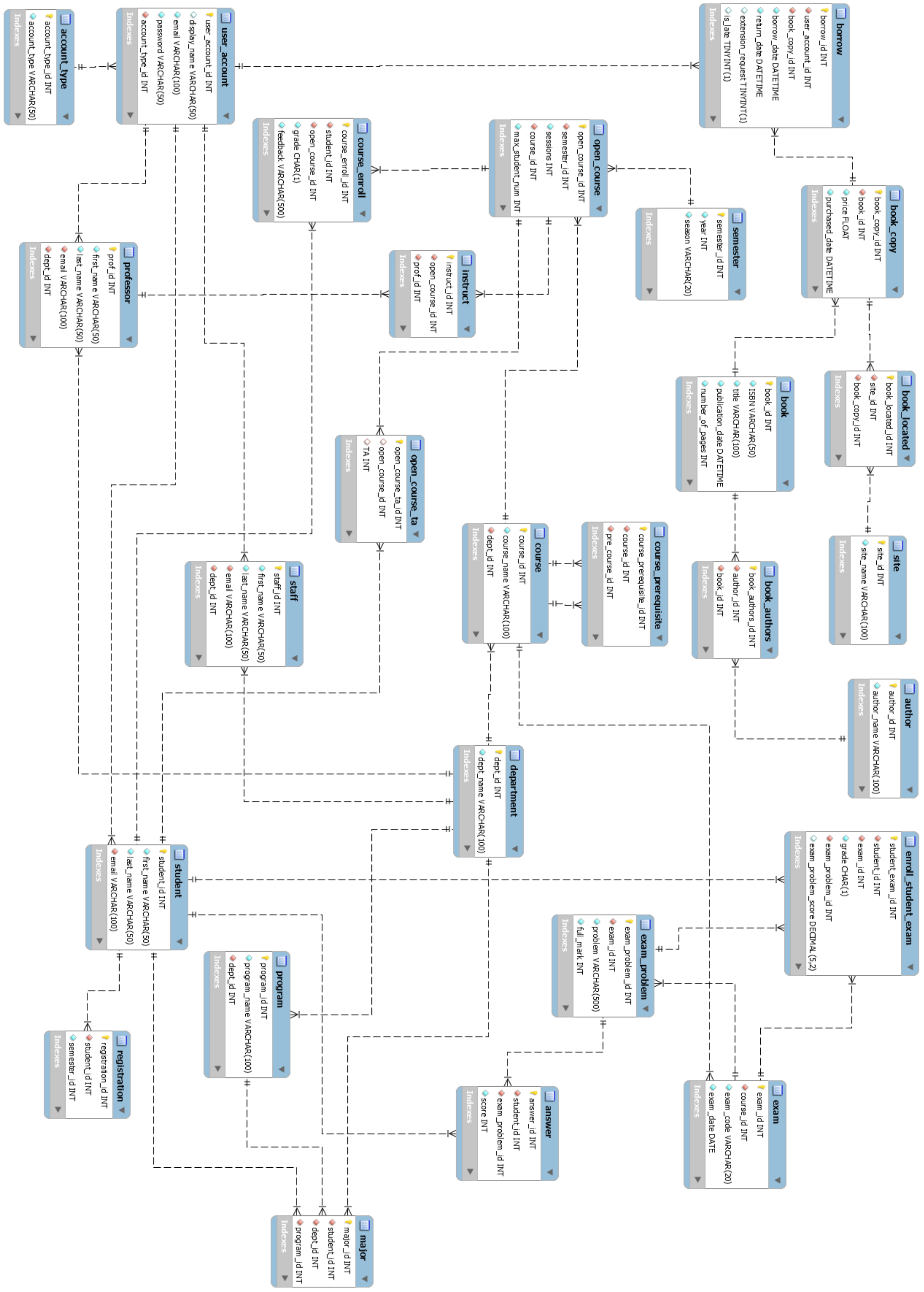
Jama Jama
11/27/2020
CSE460
Professor Chuang

I have a table called site which holds all the sites on campus and it is connected to book_located table which holds the ISBN number of the book and the site ID where it is located. This ensures that all copies and their locations are accounted for. I have created a borrow table which has data on who borrowed the book using the user_account_id as well as a is_late boolean flag which is triggered after the return_date has passed. This will also ensure that users can't borrow any books as long as their are overdue books. However, the logic on the backend/frontend should reflect the appropriate time limit since my implementation doesn't restrict users to two weeks. Also I have a boolean attribute called extension_request which can be triggered to increase the return_date. Additionally, borrow is linked to book_copy_id using a foreign key in case a user tries to return a different copy of the same book.

Jama Jama
11/27/2020
CSE460
Professor Chuang



Jama Jama
11/27/2020
CSE460
Professor Chuang



Jama Jama
11/27/2020
CSE460
Professor Chuang

Account_type:
account_type(account_type_id, account_type)

PK(account_type.account_type_id)

Semester:
semester(semester_id, year, season)

PK(semester.semester_id)

User_account:
user_account(user_account_id, display_name, email, password, account_type_id)

PK(user_account.user_account_id)

FK(user_account.account_type_id, account_type.account_type_id)

Department:
department(dept_id, dept_name)

PK(department.dept_id)

Student:
student(student_id, first_name, last_name, email)

PK(student.student_id)

FK(student.email, user_account.email)

Professor:
Professor(prof_id, first_name, last_name, email, dept_id)

PK(Professor.prof_id)

FK(Professor.email, user_account.email)

FK(Professor.dept_id, department.dept_id)

Staff:
Staff(staff_id, first_name, last_name, email, dept_id)

PK(Staff.staff_id)

FK(Staff.email, user_account.email)

FK(Staff.dept_id, department.dept_id)

Program:
Program(program_id, program_name, dept_id)

PK(Program.program_id)

FK(Program.dept_id, department.dept_id)

Major:
Major(major_id, student_id, dept_id, program_id)

PK(Major.major_id)

FK(Major.student_id, student.student_id)

FK(Major.dept_id, department.dept_id)

FK(Major. program_id, program. program_id)

Registration:
Registration(registration_id, student_id, semester_id)

PK(Registration.registration_id)

FK(Registration.student_id,student.student_id)

Course:
Course(course_id, course_name, dept_id)

PK(Course.course_id)

FK(Course.dept_id, department.dept_id)

Open_course:
open_course(open_course_id, semester_id, sessions, course_id, max_student_num)

PK(open_course.open_course_id)

FK(open_course.semester, semester.semester_id)

FK(open_course.course_id, course.course_id)

Teacher_assistant:
teacher_assistant(teacher_assistant_id, open_course_id, TA)

PK(teacher_assistant.teacher_assistant_id)

FK(teacher_assistant.open_course_id, open_course.open_course_id)

FK(teacher_assistant.TA,student.student_id)

Instruct:
Instruct(course_prerequisite_id, course_id, pre_course_id)

PK(Instruct.course_prerequisite_id)

FK(Instruct.course_id, course.course_id)

FK(Instruct.pre_course_id, course.course_id)

Course_enroll:
course_enroll(course_enroll_id, student_id, open_course_id, grade, feedback)

PK(course_enroll.course_enroll_id)

FK(course_enroll.student_id,student.student_id)

FK(course_enroll.open_course_id, open_course.open_course_id)

Exam:
Exam(exam_id, course_id, exam_code, exam_date)

PK(Exam.exam_id)

FK(Exam.course_id course.course_id)

Exam_problem:
exam_problem(exam_problem_id, exam_id, problem, full_mark)

PK(exam_problem.exam_problem_id)

FK(exam_problem.exam_id, exam.exam_id)

Enroll_student_exam :
enroll_student_exam(student_exam_id, student_id, exam_id, grade)

PK(enroll_student_exam.student_exam_id)

FK(enroll_student_exam.student_id,student.student_id)

FK(enroll_student_exam.exam_id, exam.exam_id)

Answer:
answer(answer_id, student_id, exam_problem_id, score)

PK(answer.answer_id)

FK(answer.student_id, student.student_id),

Jama Jama
11/27/2020
CSE460
Professor Chuang

FK(answer.exam_problem_id, exam_problem.exam_problem_id)

Author:
author(author_id, author_name)

PK(author.author_id)

Book:
book(book_id, ISBN, title, publication_date, number_of_pages)

PK(book.book_id)

Book_copy:
book_copy(book_copy_id, book_id, price, purchased_date)

PK(book_copy.book_id)

FK(book_copy.book_id, book.book_id)

Book_authors:
book_authors(book_authors_id, author_id, book_id)

PK(book_authors.book_authors_id)

FK(book_authors.book_id, book.book_id),

FK(book_authors.author_id, author.author_id)

Site:
Site(site_id, site_name)

PK(Site.site_id)

Book_located:
book_located(book_located_id, site_id book_copy_id)

PK(book_located.book_located_id)

FK(book_located.site_id, site.site_id)

FK(book_located.book_copy_id,book_copy.book_copy_id)

Borrow:

Jama Jama
11/27/2020
CSE460
Professor Chuang

borrow(borrow_id, user_account_id, book_copy_id, borrow_date, return_date, extension_request, is_late)

PK(borrow.borrow_id)

FK(borrow.user_account_id, user_account.user_account_id)

FK(borrow.book_copy_id, book_copybook_copy_id)