COLLEGE DATABASE PROJECT

A college contains many departments. Each department can offer any number of courses. Many instructors can work in a department, but an instructor can work only in one department. For each department, there is a head, and an instructor can be head of only one department. Each instructor can take any number of courses, and a course can be taken by only one instructor. A student can enroll for any number of courses and each course can have any number of students.

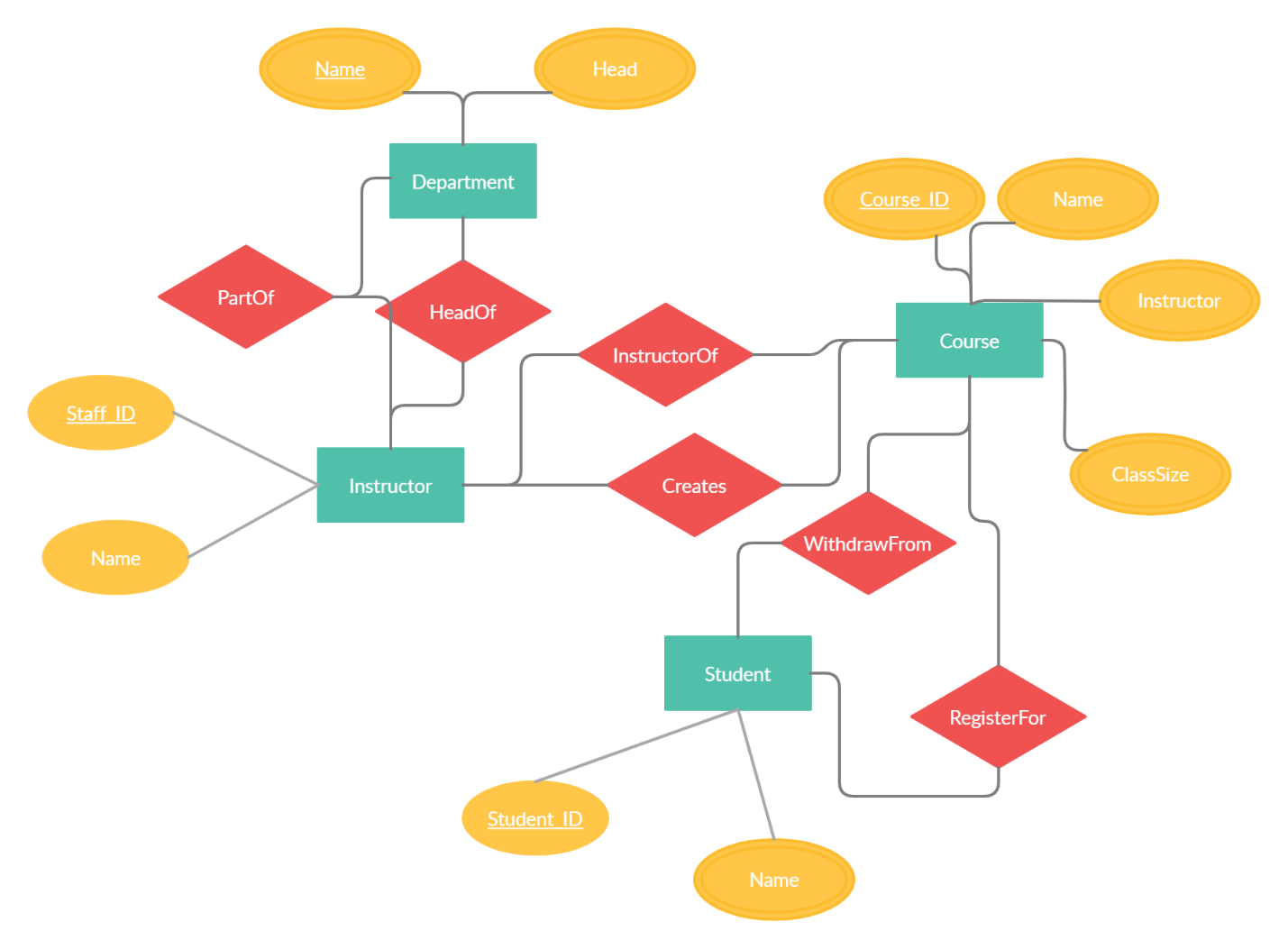
INTRODUCTION

The project focusses on designing and implementing a university database for use by the students and instructors. The application will allow students to register for courses and withdraw from course, as well as view and search for courses. The instructors will be able to create courses and view students in them, as well as remove students and declare themselves department head.

REQUIREMENT ANALYSIS

The database application will allow instructors to assign themselves to their specific departments as well as declare themselves the department head if they are the department head. Instructors can also create courses and assign themselves as the instructor for courses. Instructors can also view which students are signed up for their classes, as well as administratively withdraw students by removing them from the course. Students will have the ability to search for courses as well as sign up for courses and withdraw from courses.

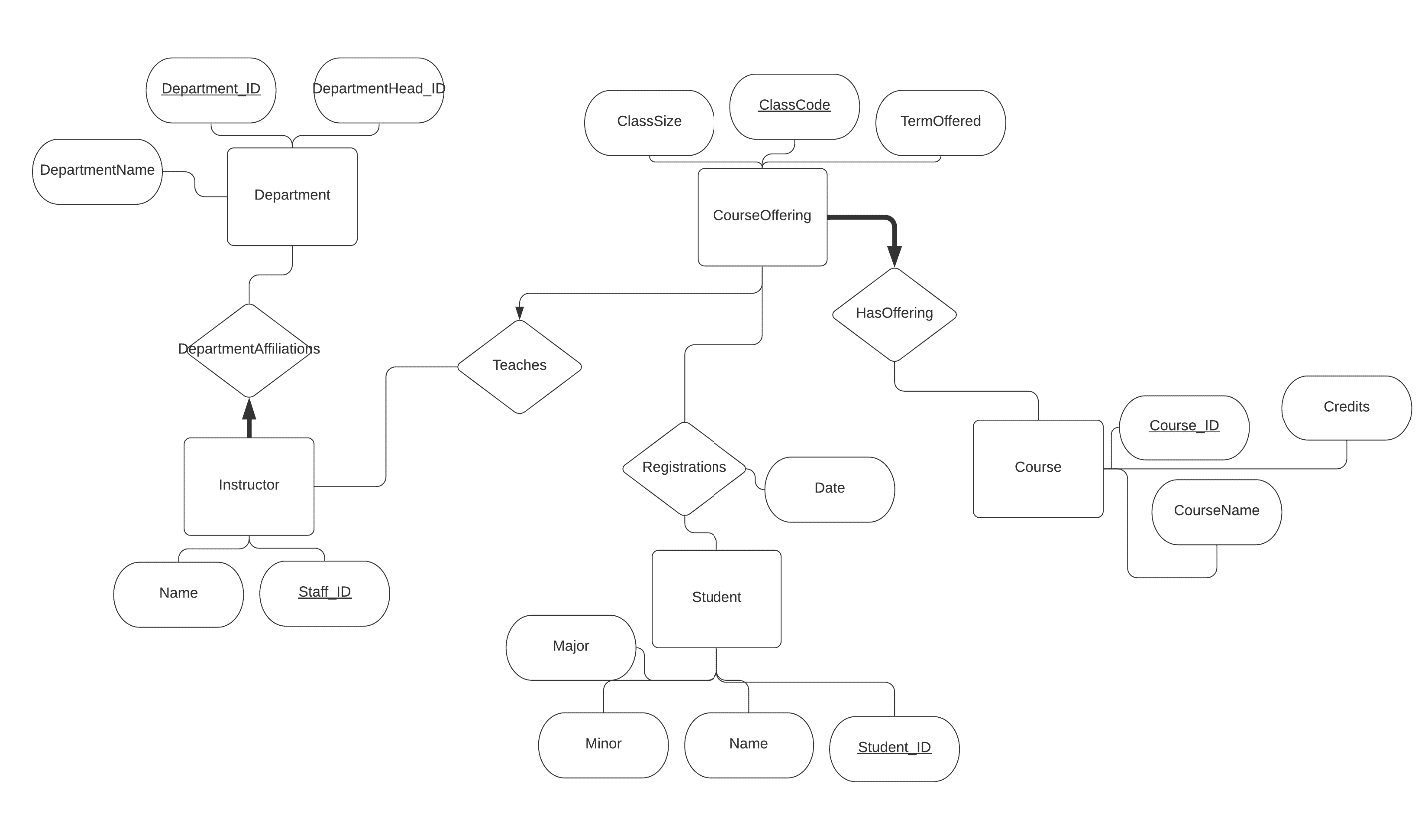
CONCEPTUAL DESIGN



The four main entities in the database will be students, departments, instructors, and courses. Students, instructors, and courses all have unique IDs as primary keys and departments have unique names. Instructors are part of a department and can be the head, as well as create and instruct courses. Students can register for courses if the ClassSize attribute allows it, and can withdraw from courses.

PHASE TWO

Revised ER Diagram



RELATIONS

CREATE TABLE Instructor

(Staff\_ID INTEGER,

Name CHAR(20),

PRIMARY KEY (Staff\_ID)

);

CREATE TABLE Department

(Department\_ID INTEGER,

DepartmentHead\_ID INTEGER,

DepartmentName CHAR(20),

PRIMARY KEY (Department\_ID),

FOREIGN KEY (DepartmentHead\_ID)

REFERENCES Instructor(Staff\_ID)

ON DELETE SET NULL

);

CREATE TABLE Course

(Course\_ID INTEGER,

Credits INTEGER,

CourseName CHAR(20),

PRIMARY KEY (Course\_ID)

);

CREATE TABLE CourseOffering

(ClassCode INTEGER,

TermOffered CHAR(20),

ClassSize INTEGER,

PRIMARY KEY (ClassCode)

);

CREATE TABLE Student

(Major CHAR(20),

Minor CHAR(20),

Name CHAR(20),

Student\_ID INTEGER,

PRIMARY KEY (Student\_ID)

);

CREATE TABLE DepartmentAffiliations

(Department\_ID INTEGER NOT NULL,

Staff\_ID INTEGER UNIQUE,

PRIMARY KEY (Staff\_ID, Department\_ID),

FOREIGN KEY (Staff\_ID)

REFERENCES Instructor(Staff\_ID)

ON DELETE CASCADE,

FOREIGN KEY (Department\_ID)

REFERENCES Department(Department\_ID)

ON DELETE CASCADE

);

CREATE TABLE Teaches

(ClassCode INTEGER UNIQUE,

Staff\_ID INTEGER,

PRIMARY KEY (Staff\_ID, ClassCode),

FOREIGN KEY (Staff\_ID)

REFERENCES Instructor(Staff\_ID)

ON DELETE CASCADE,

FOREIGN KEY (ClassCode)

REFERENCES CourseOffering(ClassCode)

ON DELETE CASCADE

);

CREATE TABLE Registrations

(ClassCode INTEGER,

Student\_ID INTEGER,

Date DATE,

PRIMARY KEY (Student\_ID, ClassCode),

FOREIGN KEY (Student\_ID)

REFERENCES Student(Student\_ID)

ON DELETE CASCADE,

FOREIGN KEY (ClassCode)

REFERENCES CourseOffering(ClassCode)

ON DELETE CASCADE

);

CREATE TABLE HasOffering

(ClassCode INTEGER UNIQUE,

Course\_ID INTEGER NOT NULL,

PRIMARY KEY (Course\_ID, ClassCode),

FOREIGN KEY (Course\_ID)

REFERENCES Course(Course\_ID)

ON DELETE CASCADE,

FOREIGN KEY (ClassCode)

REFERENCES CourseOffering(ClassCode)

ON DELETE CASCADE

);

RATIONALE FOR REFINED ER DIAGRAM AND RELATIONS

The new ER diagram introduces a new entity ‘course’ which will allow for less redundant entries for the ‘courseoffering’ entity. This also solves the functional dependency issue between a course ID and its name and credits if these attributes were all listed in the ‘courseoffering’ entity. There was also an ID attribute added to the ‘department’ entity to better identify departments. The actual relationships on the diagram have been modified to decrease the number of relationships needed to connect all of the entities. The relationship between ‘student’ and ‘courseoffering’ now includes a date and acts as a record for registrations. The teacher assignment for each course is kept track of by the ‘Teaches’ relationship, and the list of official courses being offered is kept track of by the ‘hasoffering’ entity. Lastly, the department that each instructor is a part of is now part of the ‘departmenaffiliation’ entity. The diagram also now shows the cardinality constraints of the database with the keywords UNIQUE and NOT NULL enforcing one to many relationships and full participations. For the actual definitions of the tables, all tables of entities have primary key constraints, with ‘departments’ having a foreign key denoting the staff member that is a department head. All of the relationships have a combination key as a primary key and also have foreign key constraints which generally chooses to CASCADE when a key is deleted. The reason for choosing to CASCADE is because all of the tuples made by the relationships have no meaning without both foreign keys. All updates to primary keys have the default action of being rejected. The tables and ER diagram appear to have no redundancies or functional dependency anomalies.

RESULTS

Using JDBC, a database was constructed in MySQL and connected to two Java applications. One application populates an empty database, the second allows for the user to view the results of certain queries. There were many struggles to figure out how to create the database and figure out how to use the DBMS. After that, it took a lot of time to connect Java to the database. Next was lots of time spent creating data. After this, most effort went towards creating a working text interface with basic functionalities. The front end was not reached, but if the back end were stronger, the front would be built with Java gui. There was also the issue of not knowing anything about front end development. Overall the project is facing hurdles, but has a vision. The text ui will be replaced with an actual gui and there needs to be more operations in the back end that can add and delete from the database.