Jose Cisneros
June 9, 2017
CS340

UNIVERSITY INFORMATION DATABASE

Outline

For the final project I created a university information management system. It allows the management of students, instructors, departments, and courses. The management of this information is useful to facilitate the creation and management of a course catalog, which is demonstrated as a feature of the website.

Database Outline

Entities

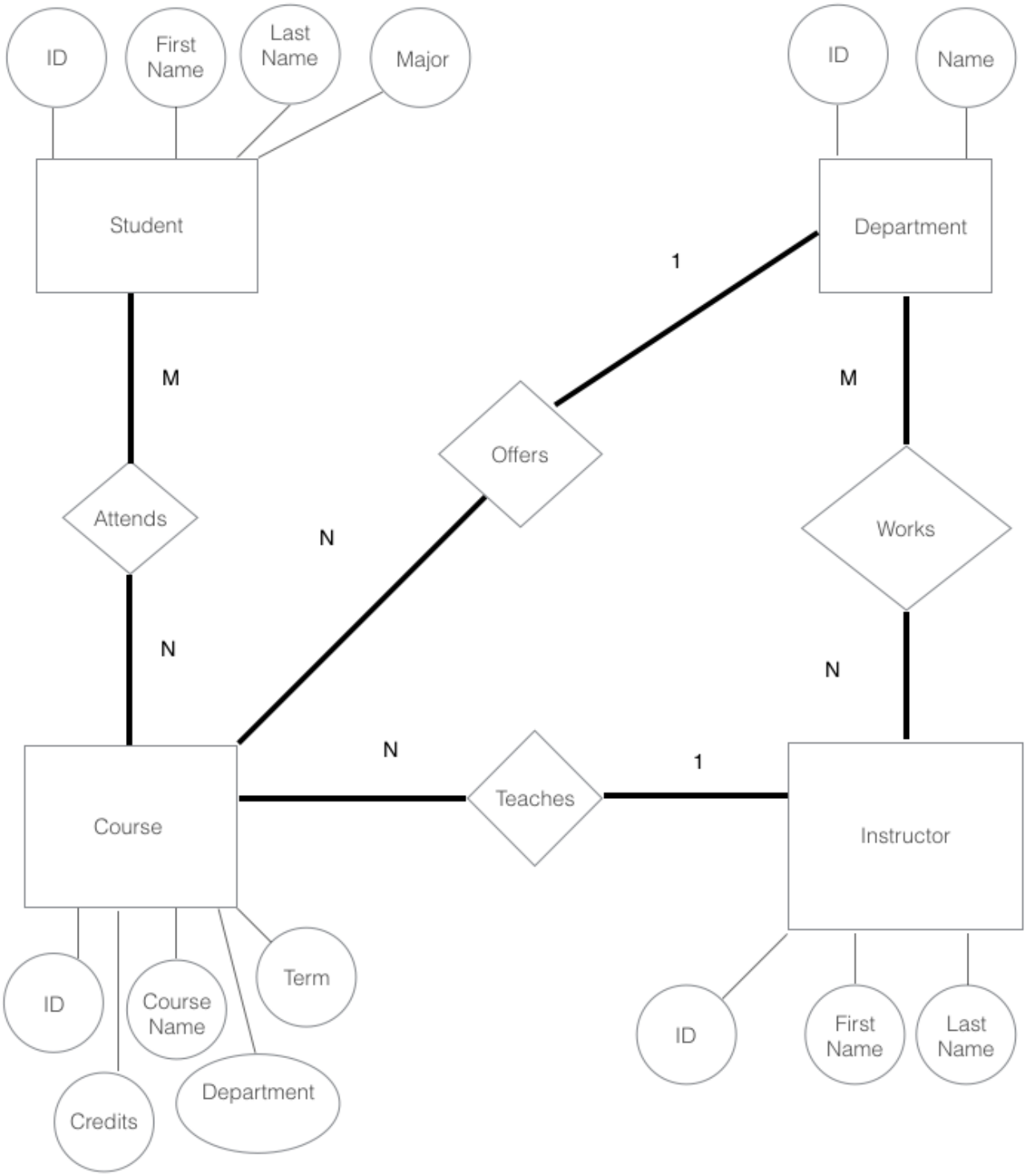
1. Students—They have an id, first name, last name, and a major.
2. Courses—They have an id, course name, credit value of the course, term it is being offered, instructor, and the department that offers the course
3. Instructors—They have an id, first name, and last name.
4. Departments—They have an id and a name.

Relationships

1. Students take courses
    - This is a many to many relationship. Many students can take many courses.
    - A student takes at least 1 course
    - A course has at least 1 student.
2. Instructors teach courses
    - This will be a 1 to many relationship.
    - A class has exactly 1 instructor, but an instructor can teach multiple classes.
    - An instructor teaches at least 1 course.
3. Departments have instructors
    - This is a many to many relationship
    - Many instructors can work for many colleges.
    - A department has at least 1 instructor and an instructor works for at least 1 department.
4. Departments offer courses
    - This is a 1 to many relationship.
    - A course is offered by exactly 1 department, but a department can offer many courses.
    - A department offers at least 1 course.

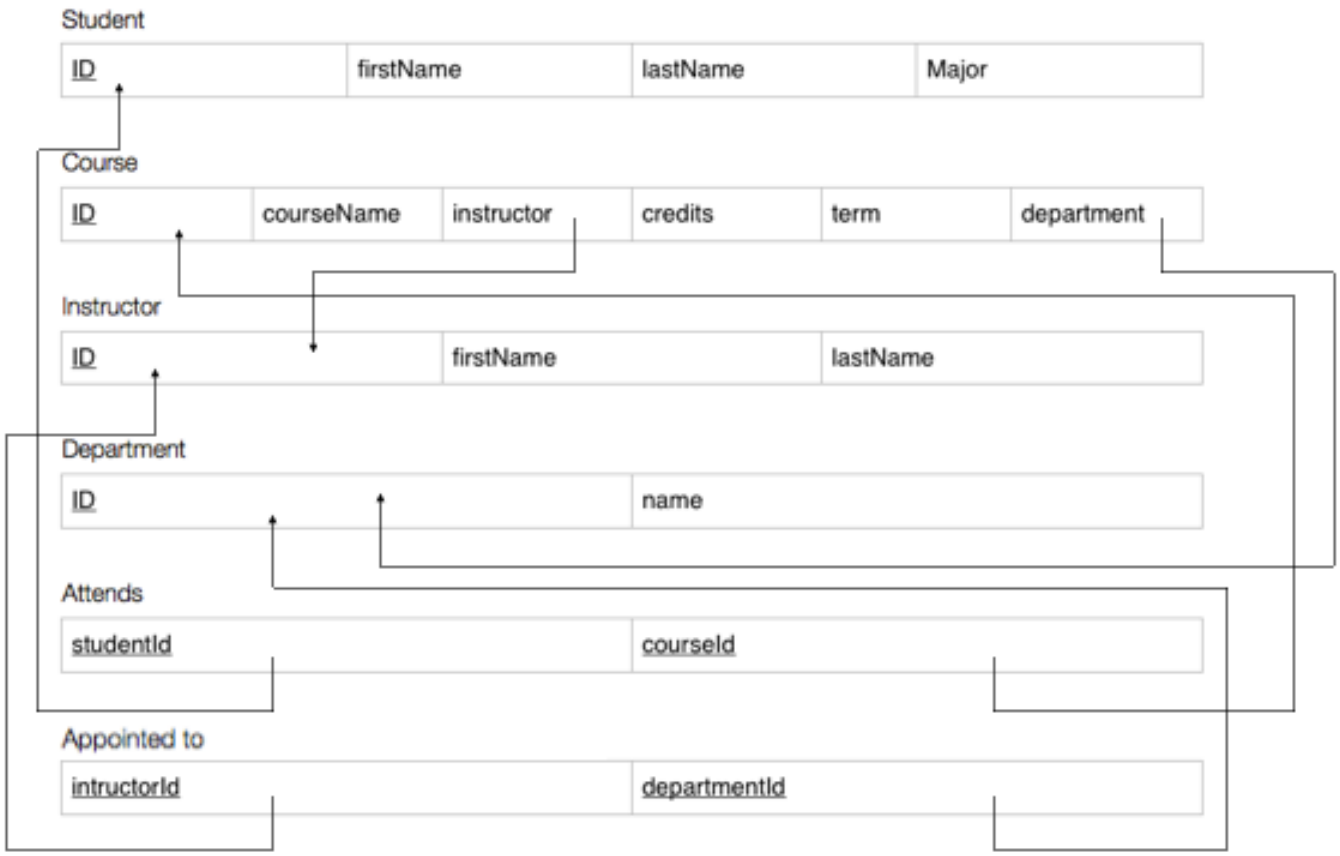
Jose Cisneros
June 9, 2017
CS340

ER Diagram



Jose Cisneros
June 9, 2017
CS340

Schema



Jose Cisneros
June 9, 2017
CS340

Data Definition Queries

```
CREATE TABLE student (
  id int NOT NULL AUTO_INCREMENT,
  firstName varchar(255) NOT NULL,
  lastName varchar(255) NOT NULL,
  gpa double NOT NULL,
  major varchar(255),
  PRIMARY KEY (id)
);

CREATE TABLE department (
id int NOT NULL AUTO_INCREMENT,
departmentName varchar(255) NOT NULL,
PRIMARY Key (id),
UNIQUE (departmentName)
);

CREATE TABLE instructor (
id int NOT NULL AUTO_INCREMENT,
firstName varchar(255) NOT NULL,
lastName varchar(255) NOT NULL,
PRIMARY Key (id)
);

CREATE TABLE course (
id int NOT NULL AUTO_INCREMENT,
courseName varchar(255) NOT NULL,
instructor int,
credits int NOT NULL,
term varchar(255) NOT NULL,
department int NOT NULL,
PRIMARY Key (id),
CONSTRAINT FOREIGN KEY(instructor) REFERENCES instructor(id) ON DELETE SET NULL ON UPDATE
CASCADE,
CONSTRAINT FOREIGN KEY(department) REFERENCES department(id) ON UPDATE CASCADE
);

CREATE TABLE studentcourse (
```

Jose Cisneros
June 9, 2017
CS340

```
  studentId int NOT NULL,
  courseId int NOT NULL,
  PRIMARY KEY (studentId, courseId),
  CONSTRAINT FOREIGN KEY(studentId) REFERENCES student(id) ON DELETE CASCADE ON UPDATE
CASCADE,
  CONSTRAINT FOREIGN KEY(courseId) REFERENCES course(id) ON DELETE CASCADE ON UPDATE
CASCADE
);

CREATE TABLE instructordepartment (
  instructorId int NOT NULL,
  departmentId int NOT NULL,
  PRIMARY KEY (instructorId, departmentId),
  CONSTRAINT FOREIGN KEY(instructorId) REFERENCES instructor(id) ON DELETE CASCADE ON
UPDATE CASCADE,
  CONSTRAINT FOREIGN KEY(departmentId) REFERENCES department(id) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

Data Manipulation Queries

Add a Course

```
INSERT INTO course(courseName, instructor, credits, term, department) VALUES ( [ courseNameInput],
[instructorInput],[creditsInput],[termInput],[departmentInput] );
```

Add Department

```
INSERT INTO department(departmentName) VALUES ( [departmentNameInput] );
```

Add Instructor

```
INSERT INTO instructor(firstName, lastName) VALUES ( [firstNameInput],[lastNameInput ] );
```

Add Student

```
INSERT INTO student(firstName, lastName, gpa, major) VALUES ( [firstNameInput],[lastNameInput],
[gpaInput],[majorInput] );
```

Assign Instructor

```
INSERT INTO instructordepartment(instructorId, departmentId) VALUES ( [instructorIdInput],
[departmentIdInput] );
```

Delete Student

```
DELETE FROM student WHERE id = [idInput];
```

Enroll Student

```
INSERT INTO studentcourse(studentId, courseId) VALUES ( [studentIdInput],[courseIdInput] );
```

Filter Student Schedule

```
SELECT table1.student, table2.courseName FROM studentcourse
INNER JOIN (SELECT id, CONCAT_WS(" ", firstName, lastName) AS student FROM student
```

Jose Cisneros
June 9, 2017
CS340

```
) AS table1 ON studentcourse.studentId = table1.id
INNER JOIN (SELECT id, courseName FROM course
) AS table2 on studentcourse.courseId = table2.id WHERE table1.id = [idInput];

-- student table
SELECT firstName, lastName, gpa, major FROM student;

-- department table
SELECT departmentName FROM department;

-- instructor table
SELECT firstName, lastName FROM instructor;

-- course table
SELECT table1.courseName, table2.instructor, table1.credits, table1.term, table1.departmentName FROM
(SELECT courseName, instructor, credits, term, departmentName FROM course
INNER JOIN department ON course.department = department.id) as table1
INNER JOIN (SELECT id, CONCAT_WS(" ", firstName, lastName) AS instructor FROM instructor) as table2
ON table1.instructor = table2.id;

-- student course
SELECT table1.student, table2.courseName FROM studentcourse
INNER JOIN (SELECT id, CONCAT_WS(" ", firstName, lastName) AS student FROM student
) AS table1 ON studentcourse.studentId = table1.id
INNER JOIN (SELECT id, courseName FROM course
) AS table2 on studentcourse.courseId = table2.id;

-- instructor department
SELECT table1.instructor, table2.departmentName FROM instructordepartment
INNER JOIN (SELECT id, CONCAT_WS(" ", firstName, lastName) AS instructor FROM instructor
) AS table1 ON instructordepartment.instructorId = table1.id
INNER JOIN (SELECT id, departmentName FROM department
) AS table2 on instructordepartment.departmentId = table2.id;
```