Database Project Writeup

Students: Ryan Ruiz, Johnny Sanchez

Database Outline

The database represents the collection of data for a Gym franchise. The data is intended to track the people in the gym, be it employees or members. We are able to track who works at the gym, what position they hold, and if they teach one of the workout classes that the gym has at one of its many locations.

This database would theoretically allow Gym employees to quickly add and remove other employees (something maybe a hiring manager might do) and assign trainers some classes to teach. On the member side of things, this would allow the registration of new members to the Gym and also be able to see if they are an active member. What makes this a flexible tool for our theoretical users is that you are able to see data across multiple locations and not just one, allowing you to see a bigger picture of the people structure of the gym franchise.

Database Explanation

The database works on the premise of being able to add and delete data. This adheres to the idea that the database serves to manage the people that are part of the gym franchise, whether it be employees or members.

Relationships

Since this database refers to a Gym franchise, then we must have a Gym entity that represents the locations of all of the gyms that are part of the franchise. Next, we have the employees that work for the gym, and they form a separate entity. The employees and the gym are connected by the gym location, which would be set by the ID of that particular gym. It is important to note that employees may work for different gym locations and the relationship is found in a separate table isEmployee.

Next, we have the members of the Gym. This would be denoted by the Member entity. The members are associated with a Gym through the isMember relationship, which links a member to the gym through the member ID and gym ID. Many members can be part of the same gym and members can be part of many gyms. This is a many-to-many relationship.

A Gym may also offer workout classes. This is denoted by the Class relationship and is a one-to-many relationship.

A member may take many workout classes at once. A class may have one or more students. This is a many-to-many relationship.

Constraints

If we modify data, such as deleting data, we need to make sure that rows in other tables will also deleted appropriately.

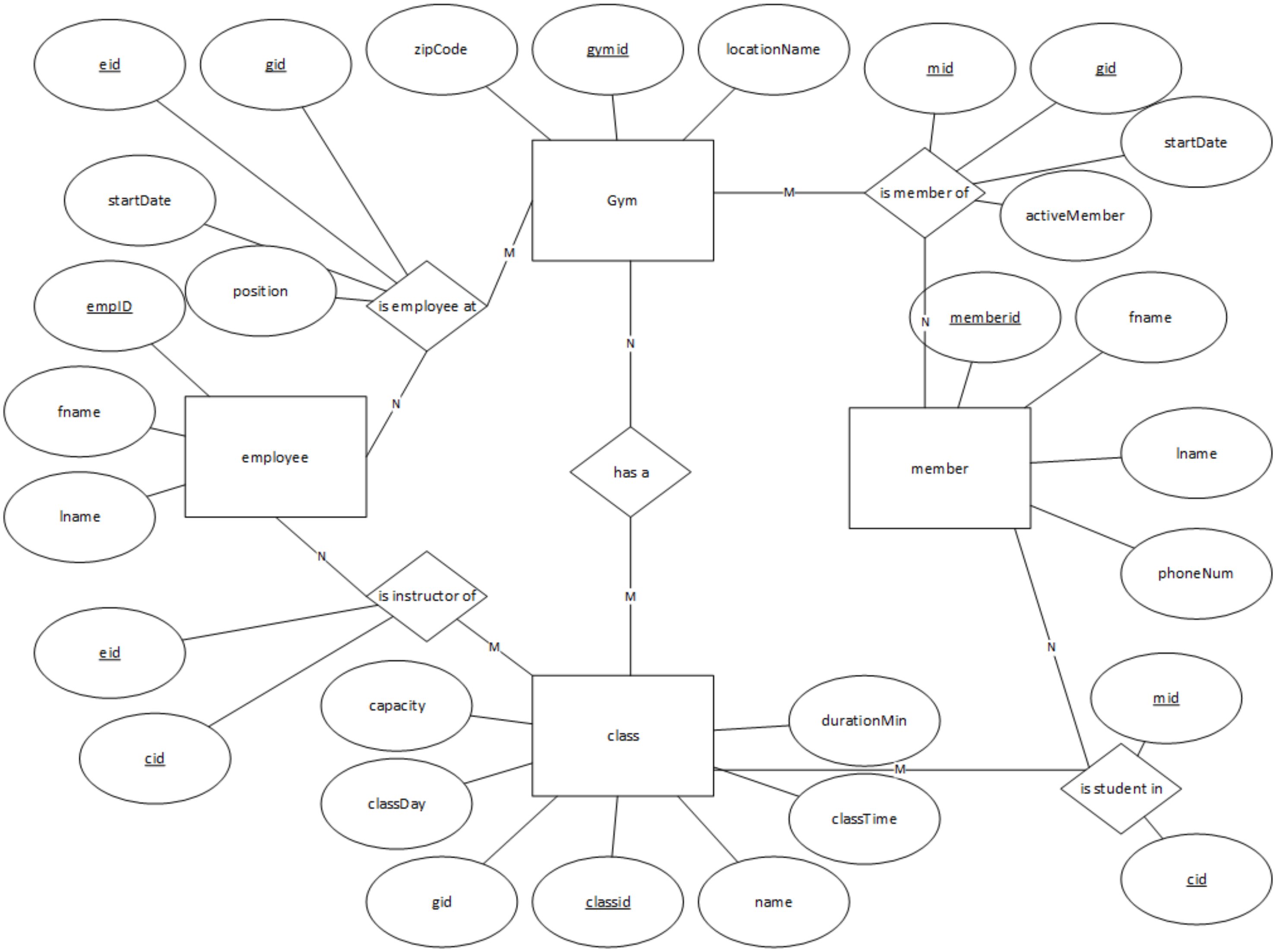
If a gym location is deleted, then all the members, employees and workout classes expressed in the relationship will be deleted from the relationship tables.

If an employee is deleted, then the appropriate rows will need to be deleted from any of the classes that they have taught as well as being deleted from the appropriate gym location (relationship table). This will be done by cascading any changes through its references.

If a member is deleted, then they will be deleted from the appropriate workout class roster and gym location. This will be done by cascading the changes through its reference.

If a class is deleted, then only the relationship is deleted, the employees and members stay intact.

ER Diagram



Schema

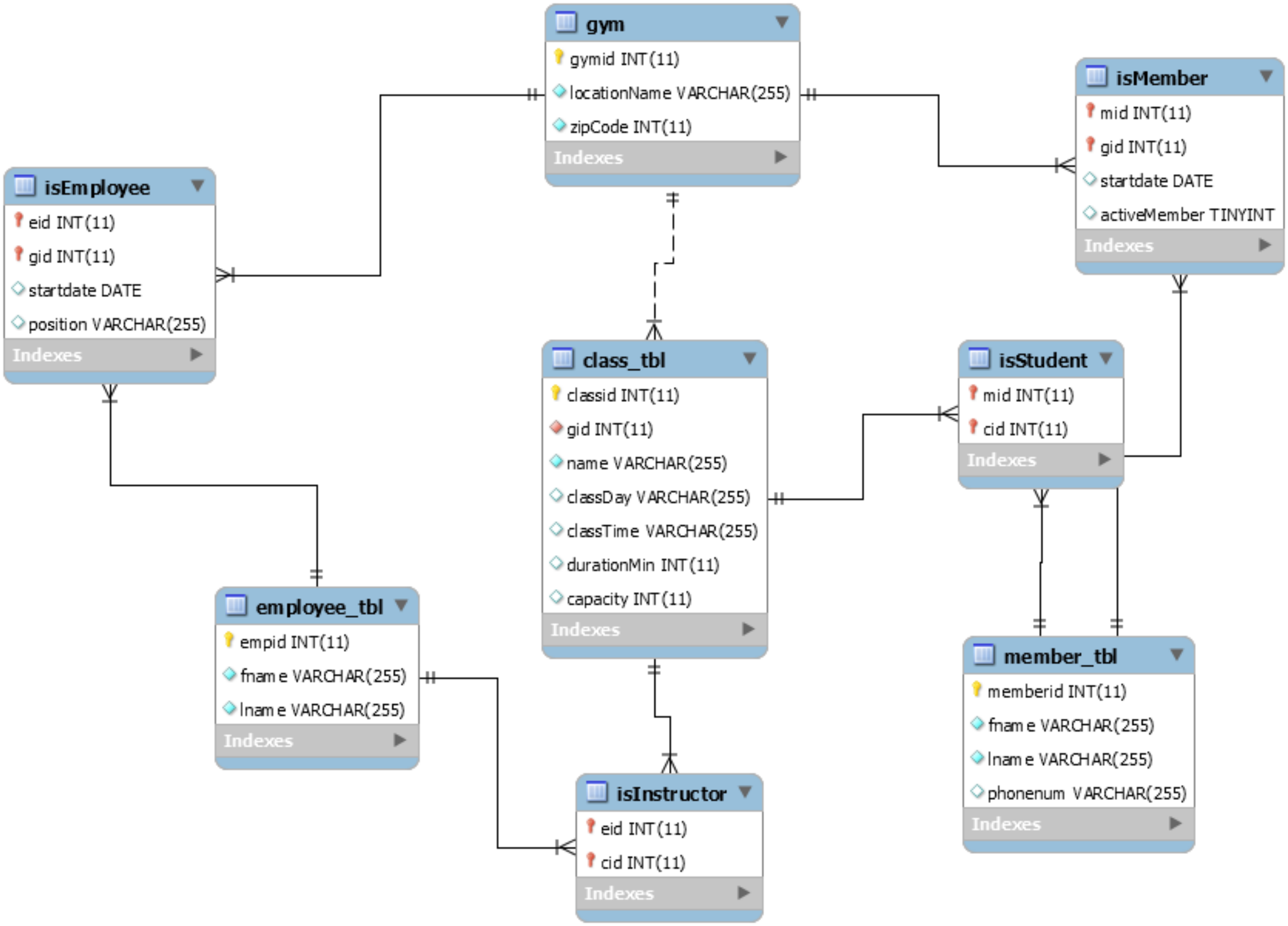


Table Creation Queries

```
CREATE TABLE `member_tbl` (
  memberid int(11) NOT NULL AUTO_INCREMENT,
  fname varchar(255) NOT NULL,
  lname varchar(255) NOT NULL,
  phonenum varchar(255),
  PRIMARY KEY (memberid)
) ENGINE=InnoDB;

CREATE TABLE `gym` (
  gymid int(11) NOT NULL AUTO_INCREMENT,
  locationName varchar(255) NOT NULL,
  zipCode int(11) NOT NULL,
  PRIMARY KEY (gymid)
) ENGINE=InnoDB;

CREATE TABLE `employee_tbl` (
  empid int(11) NOT NULL AUTO_INCREMENT,
  fname varchar(255) NOT NULL,
  lname varchar(255) NOT NULL,
  PRIMARY KEY (empid)
) ENGINE=InnoDB;

CREATE TABLE `class_tbl` (
  classid int(11) NOT NULL AUTO_INCREMENT,
  gid int(11) NOT NULL,
  name varchar(255) NOT NULL,
  classDay varchar(255),
  classTime varchar(255),
  durationMin int(11),
  capacity int(11),
  FOREIGN KEY (gid) REFERENCES gym (gymid)
    ON DELETE CASCADE ON UPDATE CASCADE,
  PRIMARY KEY (classid)
) ENGINE=InnoDB;


CREATE TABLE `isStudent` (
  mid int(11) NOT NULL,
  cid int(11) NOT NULL,
  PRIMARY KEY  (mid, cid),
  FOREIGN KEY (mid) REFERENCES member_tbl (memberid)
    ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (cid) REFERENCES class_tbl (classid)
    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE `isMember` (
  mid int(11) NOT NULL,
  gid int(11) NOT NULL,
  startdate date,
  activeMember boolean,
  PRIMARY KEY  (mid, gid),

  FOREIGN KEY (mid) REFERENCES member_tbl (memberid)
    ON DELETE CASCADE ON UPDATE CASCADE,

  FOREIGN KEY (gid) REFERENCES gym (gymid)
    ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `isInstructor` (
  eid int(11) NOT NULL,
  cid int(11) NOT NULL,
  PRIMARY KEY  (eid, cid),
  FOREIGN KEY (eid) REFERENCES employee_tbl (empid)
    ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (cid) REFERENCES class_tbl (classid)
    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE `isEmployee` (
  eid int(11) NOT NULL,
  gid int(11) NOT NULL,
  startdate date,
  position varchar(255),
  PRIMARY KEY  (eid, gid),

  FOREIGN KEY (eid) REFERENCES employee_tbl (empid)
    ON DELETE CASCADE ON UPDATE CASCADE,

  FOREIGN KEY (gid) REFERENCES gym (gymid)
    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;
```

General Use Queries

```
-- Show Table Contents
-- gym
SELECT gym.gymid, gym.locationName, gym.zipCode FROM gym

-- employee_tbl
SELECT employee_tbl.empid, employee_tbl.fname, employee_tbl.lname, isEmployee.position, gym.locationName, isEmployee.startdate FROM employee_tbl
LEFT JOIN isEmployee ON employee_tbl.empid=isEmployee.eid
LEFT JOIN gym ON gym.gymid=isEmployee.gid

-- class_tbl
SELECT class_tbl.classid, class_tbl.gid, class_tbl.name, class_tbl.classDay, class_tbl.classTime, class_tbl.durationMin, class_tbl.capacity, employee_tbl.fname, employee_tbl.lname
FROM class_tbl
LEFT JOIN isInstructor ON class_tbl.classid=isInstructor.cid
LEFT JOIN employee_tbl ON employee_tbl.empid=isInstructor.eid

-- isStudent / class_tbl
SELECT class_tbl.name, member_tbl.fname, member_tbl.lname FROM isStudent INNER JOIN member_tbl ON isStudent.mid = member_tbl.memberid INNER JOIN class_tbl ON isStudent.cid = class_tbl.classid WHERE isStudent.cid = [userSelect];

-- member_tbl
SELECT member_tbl.memberid, member_tbl.fname, member_tbl.lname, gym.locationName, isMember.startdate, isMember.activeMember FROM member_tbl LEFT JOIN isMember ON member_tbl.memberid=isMember.mid LEFT JOIN gym ON isMember.gid=gym.gymid

-- Add/Assign Queries
-- gym
INSERT INTO gym(locationName, zipCode) VALUES ([userInput],[userInput]);

-- employee_tbl / isEmployee / isInstructor
INSERT INTO employee_tbl(fname, lname) VALUES ([userInput],[userInput]);
INSERT INTO isEmployee(eid, gid, startdate, position) VALUES ([userSelect],[userSelect],NOW(),[userInput]);
INSERT INTO isInstructor(eid, cid) VALUES ([userSelect],[userSelect]);

-- member_tbl / isMember / isStudent
INSERT INTO member_tbl(fname, lname, phonenum) VALUES ([userInput],[userInput],[userInput]);
INSERT INTO isMember(mid, gid, startdate, activeMember) VALUES ([userSelect],[userSelect],[userInput],[userInput]);
INSERT INTO isStudent(mid, cid) VALUES ([userSelect],[userSelect]);
```

```sql

-- class_tbl
INSERT INTO class_tbl(gid, name, classDay, classTime, durationMin, capacity)
VALUES ([userSelect],[userInput],[userInput],[userInput],[userInput],[userInput]);


-- Delete / Manage Queries
-- gym
DELETE FROM gym WHERE gymid= [userSelect];

-- employee_tbl / isEmployee / isInstructor
DELETE FROM employee_tbl WHERE empid= [userSelect];
DELETE FROM isEmployee WHERE eid= [userSelect] AND gid= [userSelect];
DELETE FROM isInstructor WHERE eid= [userSelect] AND cid= [userSelect];

-- member_tbl / isMember / isStudent
DELETE FROM member_tbl WHERE memberid= [userSelect];
DELETE FROM isMember WHERE mid= [userSelect] AND gid= [userSelect];
DELETE FROM isStudent WHERE mid= [userSelect] AND cid= [userSelect];

-- class_tbl
DELETE FROM class_tbl WHERE classid= [userSelect];

-- Drop-down Menu Queries
-- These queries used in while loop to generate options for <select> element
-- gym
SELECT gymid, locationName FROM gym;

-- employee
SELECT empid, fname FROM employee_tbl;
SELECT empid, fname FROM employee_tbl WHERE empid= [userSelect];

-- member
SELECT memberid, fname FROM member_tbl;
SELECT memberid, fname FROM member_tbl WHERE memberid= [userSelect];

-- class
SELECT classid, name FROM class_tbl;

-- isInstructor / class_tbl
SELECT cid, name FROM isInstructor
INNER JOIN class_tbl ON class_tbl.classid=isInstructor.cid
WHERE eid= [userSelect];

-- isEmployee / gym
SELECT gid, locationName FROM isEmployee INNER JOIN gym ON gym.gymid=isEmployee.gid WHERE eid= [userSelect];

-- isStudent / class_tbl
SELECT cid, name FROM isStudent INNER JOIN class_tbl ON class_tbl.classid=isStudent.cid WHERE mid= [userSelect];

-- isMember / gym
SELECT gid, locationName FROM isMember INNER JOIN gym ON gym.gymid=isMember.gid WHERE mid= [userSelect];
```