Nicholas Sosnivka

CSC 423

Project 1 Part 1

1. Entity types
   1. Department
   2. Student
   3. Major
   4. Event
2. Relationship types
   1. Student ------ **Studies** ------> major
   2. Major ------ **AssociatedWith** -----> Department

or (Department ----- **Offers** ------> Major)

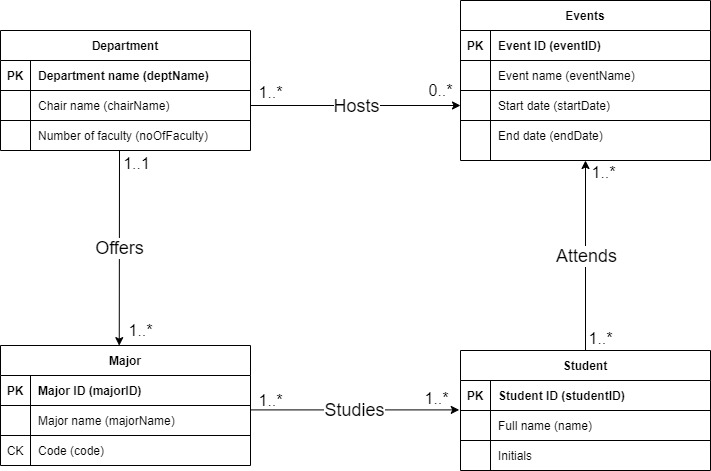
* 1. Department ------ **Hosts** ------> Event
  2. Student ------ **Attends** ------> Event

1. Multiplicity constraints
   1. Student (1..\*) ------ **Studies** ------> (1..\*) major
      1. Majors can be declared by one (1) or more (\*) students
      2. Students … each of them has declared at least one major (1..\*)
   2. Major (1..\*) ------ **AssociatedWith** -----> (1..1) Department

or (Department (1..1) ----- **Offers** ------> (1..\*) Major)

* + 1. A major references one department (1..1), however a department offers one (1) or more (\*) majors
  1. Department (1..\*) ------ **Hosts** ------> (0..\*) Event
     1. An event can be hosted by one (1) or more (\*) departments
     2. Each department has the possibility of hosting events (possibility meaning they don’t have to host an event, so (0..\*))
  2. Student (1..\*) ------ **Attends** ------> (1..\*) Event
     1. Each event will comprise (or will be attended by) one (1) or more (\*) students
     2. Students must attend one (1) or more (\*) events

1. Attributes
   1. Department: department name, chair name, number of faculty
      1. Department(deptName, chairName, noOfFaculty)
   2. Student: student ID, first name, last name, initials
      1. Student(studentID, fName, lName, initials)
   3. Major: major ID, major name, code
      1. Major(majorID, majorName, code)
   4. Event: event ID, eventName, start date, end date
      1. Event(eventID, eventName, startDate, endDate,)
2. Candidate and primary keys
   1. Department
      1. Primary key: department name (deptName)
         1. There are no departments at a university with multiple that have the same department name
   2. Student
      1. Primary key: student ID (studentID)
         1. There are no students at a university that have the same student ID
   3. Major
      1. Primary key: major ID (majorID)
         1. There are no majors at a university that have the same majorID
      2. Candidate key: code (code)
         1. The code attribute is unique since there are no multiple of the same code for a major
   4. Event
      1. Primary key: event ID (eventID).
         1. No events can have the same eventID.



Part 2

1. Derive relations from the conceptual model
   1. Every \*:\* relationship creates a new relation which are
      1. majorStudent(majorID, studentID)
      2. studentAttendance(studentID, eventID)
      3. departmentEvent(deptName, eventID)
   2. The 1:\* relationship between Department and Major causes the primary key from Department (deptName) to be the foreign key in Major.

So the new relations are going to be:

Student (studentID, fName, lName, initials)

Major (majorID, majorName, code, deptName)

Department (deptName, chairName, noOfFaculty)

Event (eventID, eventName, startDate, endDate)

majorStudent (majorID, studentID)

studentAttendance (studentID, eventID)

departmentEvent (deptName, eventID)

1. Validate the logical model against user transactions
   1. Functional Dependencies in Student
      1. studentID 🡪 fName, lName, initials (primary key)
      2. fName, lName 🡪 initials (Transitive dependency)
         1. This special dependency adds up a lot of complexity compared to the redundancy it removes, so it is best to keep it in the student table because of the extra calculations joining the two tables would imply. So in this case we are not removing this dependency.
   2. Functional Dependencies in Major
      1. majorID 🡪 majorName, code (primary key)
      2. code 🡪 majorID, majorName (candidate key)
   3. Functional Dependencies in Department
      1. deptName 🡪 chairName, noOfFaculty (primary key)
   4. Functional Dependences in Event
      1. eventID 🡪 eventName, startDate, endDate (primary key)
   5. Functional Dependences in majorStudent
      1. There are no functional dependencies in majorStudent
   6. Functional Dependencies in studentAttendance
      1. There are no functional dependencies in studentAttendance
   7. Functional Dependencies in departmentEvent
      1. There are no functional dependencies in departmentEvent

There are no functional dependencies that exist that are partial or transitive, therefore the E-R diagram is in 3NF.

1. Validate the logical model against user transactions
   1. List all students and their names who attended the homecoming event
      1. In order to do this the user would have to triple join the Student, studentAttendance, and Event tables and would list the studentID, fName, lName where studentAttendance studentID = Student studentID, studentAttendance eventID = Event eventID, and where the Event eventName ~= “homecoming”.
   2. List all students and their names that study computer science.
      1. In order to do this the user would have to triple join the major, majorStudent, and Student tables and would list the studentID, fName, lName where majorStudent majorID = Major majorID, majorStudent studentID = Student studentID, and majorName ~= ‘computer science’ (or the code = ‘CSC’)
   3. List the chair name in the department that oversees the computer science major
      1. In order to do this the user would have to join the Department and Major tables and would list the chairName and majorName where Department deptName = Major deptName and majorName ~= ‘computer science’ (or the code = ‘CSC’)
   4. List the department name that hosts the homecoming event
      1. In order to do this the user would have to join the departmentEvent table and the Event table and would list the deptName where departmentEvent eventID = Event eventID and eventName ~= ‘homecoming’
   5. Find the number of students studying psychology
      1. In order to do this the user would have to join the majorStudent and Major tables and would select the count(studentID) where majorStudent majorID = Major majorID and majorName ~= ‘psychology’ (or the code ‘PSY’)
2. Integrity constraints
   1. Primary key constraints: deptName, studentID, majorID, and eventID all can’t be null because they are assigned as primary keys. The entity integrity constraint states that primary keys can’t be null and they have to be unique.
   2. For the foreign key deptName in Major from Department and the foreign keys in the \*:\*, they must match a same value of some attribute from it’s home relation or else they must be null.
   3. code in Major– alternate key cannot be null and must be unique

**Student** (studentID, fName, lName, initials)

**Primary Key** studentID

**Major** (majorID, majorName, code, deptName)

**Primary Key** majorID

**Alternate Key** code

**Foreign Key** deptName **references** Department(deptName) ON UPDATE CASCADE ON DELETE CASCADE

**Department** (deptName, chairName, noOfFaculty)

**Primary Key** deptName

**Event** (eventID, eventName, startDate, endDate)

**Primary Key** eventID

**majorStudent** (majorID, studentID)

**Primary Key** majorID, studentID

**Foreign Key** majorID **references** Major(majorID) ON UPDATE CASCADE ON DELETE CASCADE

**Foreign Key** studentID **references** Student(studentID) ON UPDATE CASCADE ON DELETE CASCADE

**studentAttendance** (studentID, eventID)

**Primary Key**  studentID, eventID

**Foreign Key** studentID **references** Student(studentID) ON UPDATE CASCADE ON DELETE CASCADE

**Foreign Key** eventID **references** Event(eventID) ON UPDATE CASCADE ON DELETE CASCADE

**departmentEvent** (deptName, eventID)

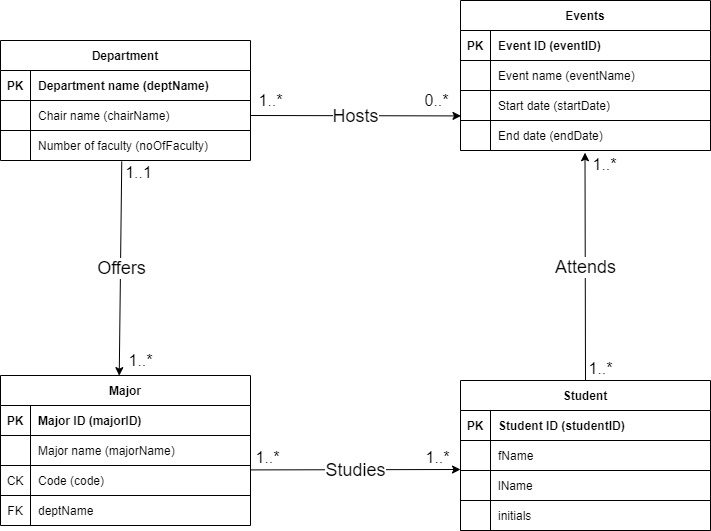
**Primary Key** deptName, eventID

**Foreign Key** deptName **references** Department(deptName) ON UPDATE CASCADE ON DELETE CASCADE

**Foreign Key** eventID **references** Event(eventID) ON UPDATE CASCADE ON DELETE CASCADE

* 1. General constraints
     1. Department names must always start with Department
     2. chairName cannot be NULL in Department
     3. Initials > one character
     4. Code has to be three characters in Major
     5. End date must be after start date in Event
     6. Start date must be after a past date or current date

1. E-R diagram for the logical level



Part 3

1. CREATE TABLE Student

(studentID VARCHAR(4) PRIMARY KEY,

fName VARCHAR(100) NOT NULL,

lName VARCHAR(100) NOT NULL,

initials VARCHAR(4) CONSTRAINT greaterThan1

CHECK (LENGTH(initials) > 1));

CREATE TABLE Major

(majorID VARCHAR(4) PRIMARY KEY,

majorName VARCHAR(100) NOT NULL,

code VARCHAR(4) UNIQUE NOT NULL CONSTRAINT mustBe3

CHECK (LENGTH(code) = 3),

deptName VARCHAR(100),

FOREIGN KEY(deptName) REFERENCES Department ON DELETE CASCADE);

CREATE TABLE Department

(deptName VARCHAR(100) PRIMARY KEY CONSTRAINT startDept

CHECK (deptName LIKE 'Department%'),

chairName VARCHAR(100) NOT NULL,

noOfFaculty INT);

CREATE TABLE Event

(eventID VARCHAR(4) PRIMARY KEY,

eventName VARCHAR(100) NOT NULL,

startDate DATE,

endDate DATE,

CONSTRAINT endDateGreater

CHECK (endDate > startDate),

CONSTRAINT startDateGreater

CHECK (startDate > '08-DEC-2021'));

CREATE TABLE majorStudent

(majorID VARCHAR(4),

studentID VARCHAR(4),

PRIMARY KEY(majorID, studentID),

FOREIGN KEY(majorID) REFERENCES Major ON DELETE CASCADE,

FOREIGN KEY(studentID) REFERENCES Student ON DELETE CASCADE);

CREATE TABLE studentAttendance

(studentID VARCHAR(4),

eventID VARCHAR(4),

PRIMARY KEY(studentID, eventID),

FOREIGN KEY(studentID) REFERENCES Student ON DELETE CASCADE,

FOREIGN KEY(eventID) REFERENCES Event ON DELETE CASCADE);

CREATE TABLE departmentEvent

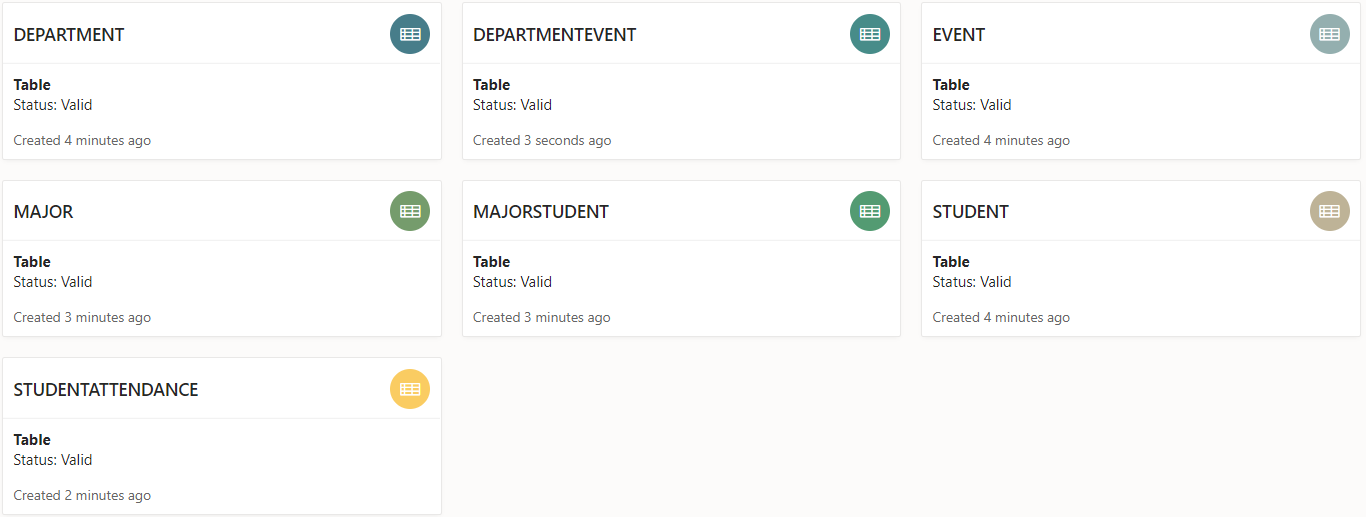
(deptName VARCHAR(100),

eventID VARCHAR(4),

PRIMARY KEY(deptName, eventID),

FOREIGN KEY(deptName) REFERENCES Department ON DELETE CASCADE,

FOREIGN KEY(eventID) REFERENCES Event ON DELETE CASCADE);



1. INSERT INTO Student (studentID, fName, lName, initials)

WITH tempTable AS (

SELECT 1, 'S001', 'Nick', 'Sosnivka', 'NVS' FROM dual UNION ALL

SELECT 2, 'S002', 'Lionel', 'Messi', 'LM' FROM dual UNION ALL

SELECT 3, 'S003', 'Barry', 'Allen', 'BA' FROM dual UNION ALL

SELECT 4, 'S004', 'Julia', 'Sosnivka', 'JSS' FROM dual UNION ALL

SELECT 5, 'S005', 'Walter', 'White', 'WW' FROM dual

)

SELECT \* FROM tempTable

INSERT INTO Student

Values

('S001', 'Nick', 'Sosnivka', 'NVS');

INSERT INTO Student

Values

('S002', 'Lionel', 'Messi', 'LM');

INSERT INTO Student

Values

('S003', 'Barry', 'Allen', 'BA');

INSERT INTO Student

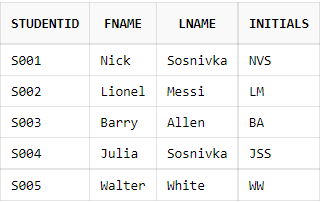
Values

('S004', 'Julia', 'Sosnivka', 'JSS');

INSERT INTO Student

Values

('S005', 'Walter', 'White', 'WW');



INSERT INTO Department

VALUES

('Department of Physics', 'Albert Einstein', 15);

INSERT INTO Department

VALUES

('Department of Mathematics', 'Isaac Newton', 10);

INSERT INTO Department

VALUES

('Department of Engineering', 'Vanessa Aguiar', 26);

INSERT INTO Department

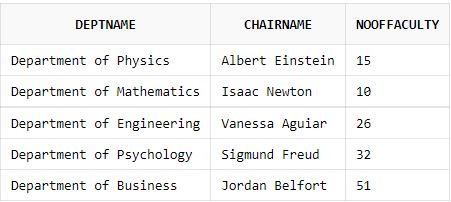
VALUES

('Department of Psychology', 'Sigmund Freud', 32);

INSERT INTO Department

VALUES

('Department of Business', 'Jordan Belfort', 51);



INSERT INTO Event

VALUES

('E101', 'Homecoming', '01-NOV-2022', '07-NOV-2022');

INSERT INTO Event

VALUES

('E102', 'Parents Weekend', '23-SEP-2022', '26-SEP-2022');

INSERT INTO Event

VALUES

('E103', 'Welcome Week Festival', '26-AUG-2022', '27-AUG-2022');

INSERT INTO Event

VALUES

('E104', 'Canes Games', '04-OCT-2022', '06-OCT-2022');

INSERT INTO Event

VALUES

('E105', 'Easter Weekend', '15-APR-2022', '17-APR-2022');



INSERT INTO Major

VALUES

('M001', 'Computer Science', 'CSC', 'Department of Engineering');

INSERT INTO Major

VALUES

('M002', 'Computer Engineering', 'CEG', 'Department of Engineering');

INSERT INTO Major

VALUES

('M006', 'Mathematics', 'MTH', 'Department of Mathematics');

INSERT INTO Major

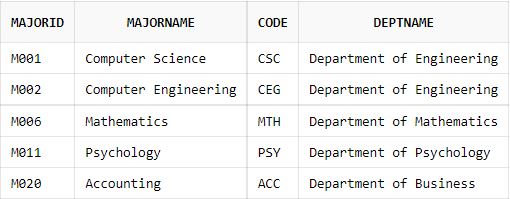
VALUES

('M011', 'Psychology', 'PSY', 'Department of Psychology');

INSERT INTO Major

VALUES

('M020', 'Accounting', 'ACC', 'Department of Business');



INSERT INTO majorStudent

VALUES ('M002', 'S003',);

INSERT INTO majorStudent

VALUES ('M001', 'S001';

INSERT INTO majorStudent

VALUES ('M006', 'S004');

INSERT INTO majorStudent

VALUES ('M011', 'S002');

INSERT INTO majorStudent

VALUES ('M020', 'S005');



INSERT INTO studentAttendance

VALUES ('S004', 'E102');

INSERT INTO studentAttendance

VALUES ('S002', 'E104');

INSERT INTO studentAttendance

VALUES ('S001', 'E101');

INSERT INTO studentAttendance

VALUES ('S003', 'E101');

INSERT INTO studentAttendance

VALUES ('S005', 'E101');



INSERT INTO departmentEvent

VALUES ('Department of Business', 'E101');

INSERT INTO departmentEvent

VALUES ('Department of Business', 'E105');

INSERT INTO departmentEvent

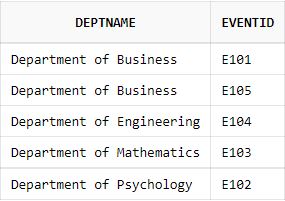
VALUES ('Department of Psychology', 'E102');

INSERT INTO departmentEvent

VALUES ('Department of Engineering', 'E104');

INSERT INTO departmentEvent

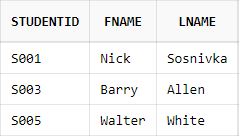
VALUES ('Department of Mathematics', 'E103');



1. Embedded queries
   1. List all students and their names who attended the homecoming event
      1. SELECT y.studentID, y.fName, y.lName

FROM studentAttendance x, Student y, Event z

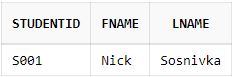
WHERE x.studentID = y.studentID AND x.eventID = z.eventID AND z.eventName LIKE '%Homecoming%';



* 1. List all students and their names that study computer science.
     1. SELECT z.studentID, z.fName, z.lName

FROM majorStudent x, Major y, Student z

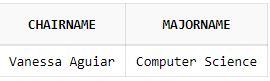
WHERE x.majorID = y.majorID AND x.studentID = z.studentID AND y.majorName LIKE '%Computer Science%';



* 1. List the chair name in the department that oversees the computer science major
     1. SELECT d.chairName, m.majorName

FROM Department d, Major m

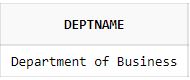
WHERE d.deptName = m.deptName AND m.code LIKE 'CSC';



* 1. List the department name that hosts the homecoming event
     1. SELECT d.deptName

FROM departmentEvent d, Event e

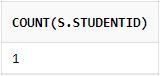
WHERE d.eventID = e.eventID AND e.eventName LIKE '%Homecoming';



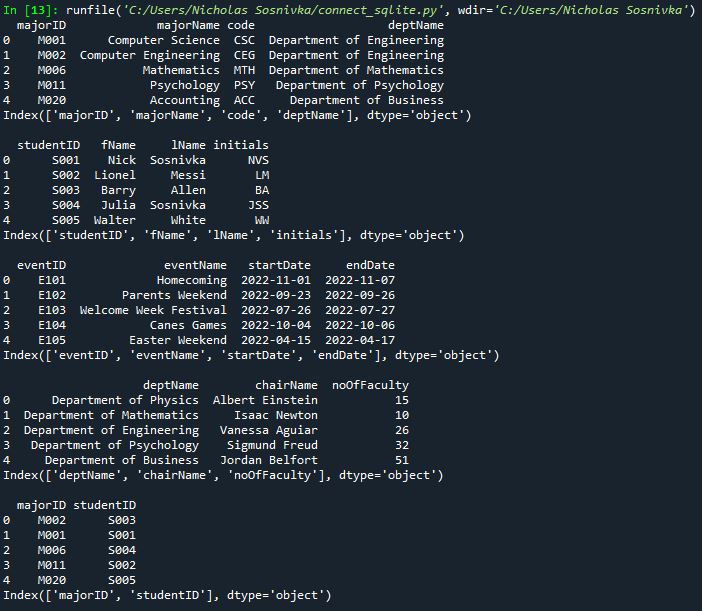
* 1. Find the number of students studying finance
     1. SELECT count(s.studentID)

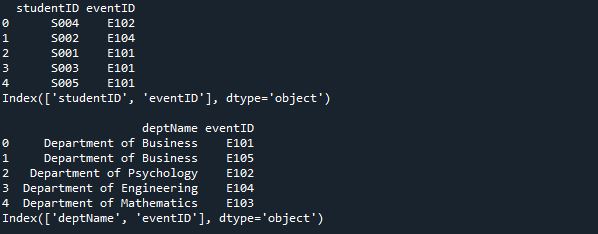
FROM majorStudent s, Major m

WHERE s.majorID = m.majorID AND m.code LIKE 'PSY';

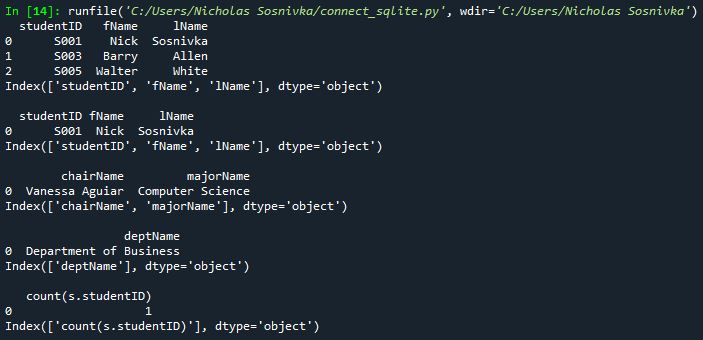


1. Upload all code and documentation to GitHub





^^^^^^^ All tables of the relations



^^^^^^^ Output of Part c (the embedded queries)

<https://github.com/nicksosnivka/CSC423/blob/main/connect_sqlite.py>