Summer Camp

By:
GROUP 6
Nicholas Yom (nicholas.yom@baruchmail.cuny.edu),
Artem Zinkin (artem.zinkin@baruchmail.cuny.edu),
Isaiah Hong (isaiah.hong@baruchmail.cuny.edu),
Tofajjal Mirza (tofajjal.mirza@baruchmail.cuny.edu)

Course Number: CIS 3400
Course Section: EMWA

Business Description:

Our summer camp primarily caters toward children from grades kindergarten to fifth grade recently considered switching from using spreadsheets to a database management system. The exponential increase in admissions to our summer camp program has led us to sought efficient methods to organize important information. We have been using spreadsheets to record our students, camp capacity, programs, program activities and program instructor. As an alternative of our manual method, we would like to be in favor of a scalable database.

With a database, Students enrolls in one of the many Programs our summer camp offers. The Program must include the following: program name, enrollment count, objective, curriculum, protocols. With every Program, there is a leading Instructor whose information must be recorded such as name, gender, age, phone number, program assigned, and et cetera. Before a Student enrolls into a Program, they are able to see the Activities included into the Program. These activities may be trips, workshops, recreational activities, group exercises.

Among our Instructors, we must keep track of their pay rate, maximum and minimum number of hours required to work, and their supervisor. This is critical to properly conduct payroll for each and every one of our instructors.

List of Entities and Attributes:

Students	**Program**
StudentID - FirstName - LastName - Grade - Gender - PhoneNumber - ParentName - EmergencyContact - StreetName - ZipCode - ApartmentNumber - City - State	ProgramID - ProgramName - EnrollmentCount - Objective - Curriculum - Protocols - CapacityStatus

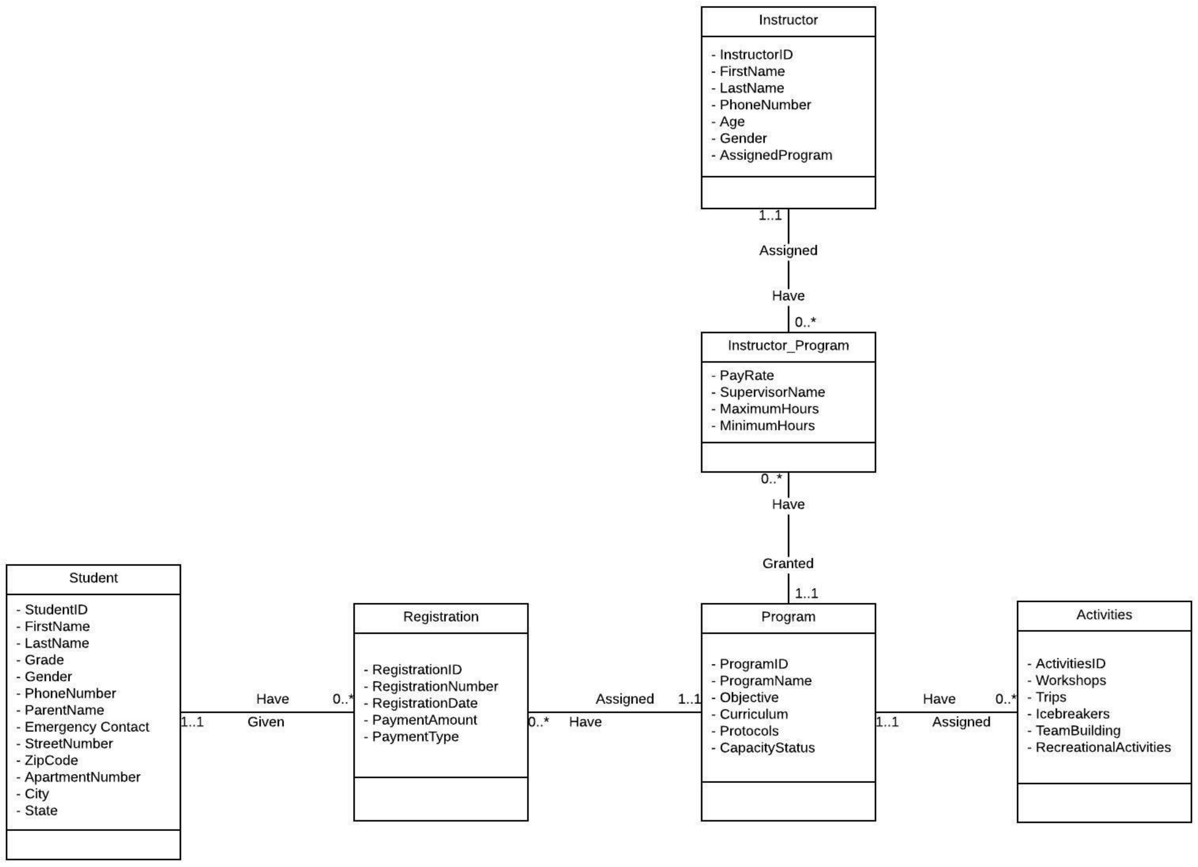
Activities	**Instructor**
ActivitesID - Workshops - Trips - Icebreakers - TeamBuilding - RecreationalActivities	InstructorID - FirstName - LastName - PhoneNumber - Age - Gender - AssignedProgram

Instructor_Program	**Registration**
- PayRate - SupervisorName - MaximumHours - MinimumHours	- RegistrationID - RegistrationNumber - RegistrationDate - PaymentAmount - PaymentType

Responsibilities of each member:

Member	**Primary Role**
Nicholas Yom	Final Report Write Up / Documentation Writer / Modelling
Artem Zinkin	Systems Analysis / Logical and Physical Modeling
Isaiah Hong	Database / Application Implementation
Tofajjal Mirza	Modelling and Application Implementation

ER Model:



Relationship Sentences:

- A STUDENT may *have* many REGISTRATIONS.
- A REGISTRATION must be *given* to one and only one STUDENT.
- A PROGRAM may *have* many REGISTRATIONS.
- A REGISTRATION must be *assigned* to one and only one PROGRAM.
- ACTIVITIES must be *assigned* to one and only one PROGRAM.
- A PROGRAM may *have* many ACTIVITIES.
- An INSTRUCTOR_PROGRAM must be *granted* to one and only one PROGRAM.
- A PROGRAM may *have* many INSTRUCTOR_PROGRAMS.
- An INSTRUCTOR_PROGRAM is *assigned* to one and only one INSTRUCTOR.
- An INSTRUCTOR may *have* many INSTRUCTOR_PROGRAMS.

Relational Model:

Student (**StudentID(KEY)**, FirstName, LastName, Grade, Gender, PhoneNumber, ParentName, EmergencyContact, StreetNumber, City, State, Zipcode, ApartmentNumber)

Registration (**RegistrationID(KEY)**, RegistrationNumber, RegistrationDate, PaymentAmount, PaymentType, **StudentID(KEY)**, **ProgramID(KEY)**)

Program (**ProgramID(KEY)**, ProgramName, Objective, Curriculum, Protocols, CapacityStatus)

Activities (**ActivitiesID(KEY)**, Workshops, Trips, Icebreakers, Teambuilding, RecreationalActivities, **ProgramID(KEY)**)

Instructor (**InstructorID(KEY)**, FirstName, LastName, PhoneNumber, Age, Gender, AssignedProgram)

Instructor_Program (**InstructorID(KEY), ProgramID(KEY)**, PayRate, SupervisorName, MaximumHours, MinimumHours)

Normalization:

Student (**StudentID(PK)**, FirstName, LastName, Grade, Gender, PhoneNumber, ParentName, EmergencyContact, StreetNumber, City, State, Zipcode, ApartmentNumber) Key: StudentID

FD1: StudentID -> FirstName, LastName, Grade, Gender, PhoneNumber, ParentName, EmergencyContact, StreetNumber, City, State, Zipcode, ApartmentNumber

FD2: Zipcode -> City, State

1NF: Yes, this is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: No, there is a transitive dependency in FD2

Solution: Break Student into two new relations, A and B
A (Zipcode, city, state) Key: Zipcode
FD1: Zipcode -> City, State
FD2: City -> State

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: No, there is a transitive key dependency in FD2

Solution:
Break A into C and D
C (city, state) Key: city
FD1: city -> state

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: Yes, there are no transitive key dependencies.

D (zipcode, city) Key: zipcode
FD1: zipcode -> city

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: Yes, there are no transitive key dependencies.

B(**StudentID(PK)**, FirstName, LastName, Grade, Gender, PhoneNumber, ParentName, EmergencyContact, StreetNumber, Zipcode, ApartmentNumber) Key: StudentID

FD1: StudentID -> FirstName, LastName, Grade, Gender, PhoneNumber, ParentName, EmergencyContact, StreetNumber, Zipcode, ApartmentNumber

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: Yes, there are no transitive key dependencies.

Solution Set : B, C and D

Registration (**RegistrationID(PK)**, RegistrationNumber, RegistrationDate, PaymentAmount, PaymentType, **StudentID(FK)**, **ProgramID(FK)**) Key: Registration, StudentID, ProgramID

FD1: RegistrationID, StudentID, ProgramID -> RegistrationNumber, RegistrationDate, PaymentAmount, PaymentType

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: Yes, there are no transitive key dependencies.

Program (**ProgramID(PK)**, ProgramName, Objective, Curriculum, Protocols, CapacityStatus)
Key: ProgramID
FD1: ProgramID -> ProgramName, Objective, Curriculum, Protocols, CapacityStatus

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: Yes, there are no transitive key dependencies.

Activities (**ActivitiesID(PK)**, Workshops, Trips, Icebreakers, Teambuilding, RecreationalActivities, **ProgramID(fk)**) Key: ActivitiesID, ProgramID

FD1: ActivitiesID ProgramID->Workshops, Trips, Icebreakers, Teambuilding, RecreationalActivities

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: Yes, there are no transitive key dependencies.

Instructor (**InstructorID(PK)**, FirstName, LastName, PhoneNumber, Age, Gender, AssignedProgram) Key: InstructorID

FD1: InstructorId -> Instructor, FirstName, LastName, PhoneNumber, Age, Gender, AssignedProgram

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: Yes, there are no transitive key dependencies.

Instructor_Program (**InstructorID(FK), ProgramID(FK)**, PayRate, SupervisorName, MaximumHours, MinimumHours) key: InstructorID, ProgramID

FD1: InstructorID, ProgramID -> PayRate, SupervisorName, MaximumHours, MinimumHours

1NF: Yes, it is a relation.
2NF: Yes, there are no partial key dependencies.
3NF: Yes, there are no transitive key dependencies.

Creating Database Schema:

```
CREATE TABLE Students
(
     StudentID           VARCHAR(10) NOT NULL,
     FirstName           VARCHAR(35),
     LastName            VARCHAR(35),
     Grade                    NUMBER,
     Gender              VARCHAR(2),
     PhoneNumber         VARCHAR(15),
     ParentName          VARCHAR(35),
     EmergencyContact    VARCHAR(15),
     StreetName          VARCHAR(35),
     ZipCode             VARCHAR(12),
     ApartmentNumber     NUMBER,
     City                VARCHAR(36),
     State                    VARCHAR(4)
     CONSTRAINT Students
          PRIMARY KEY (StudentID)
)

CREATE TABLE Program
(
     ProgramID           VARCHAR(10) NOT NULL,
     ProgramName         VARCHAR(30),
     EnrollmentCount     NUMBER,
     Objective           VARCHAR(55),
     Curriculum          VARCHAR(55),
     Protocols           VARCHAR(55),
     CapacityStatus      VARCHAR(25),
     CONSTRAINT pk_Program
          PRIMARY KEY (ProgramID)
)

CREATE TABLE Activities
(
     ActivitesID         VARCHAR(10) NOT NULL,
     Workshops           VARCHAR(55),
     Trips               VARCHAR(55),
     Icebreakers         VARCHAR(55),
     TeamBuilding        VARCHAR(55),
     RecreationalActivities   VARCHAR(55),
     CONSTRAINT pk_Activities
          PRIMARY KEY (ActivitiesID)
)

CREATE TABLE Instructor
(
     InstructorID        VARCHAR(10) NOT NULL,
     FirstName           VARCHAR(35),
     LastName            VARCHAR(35),
     PhoneNumber         VARCHAR(15),
     Age                 NUMBER,
     Gender              VARCHAR(2),
     AssignedProgram     VARCHAR(30),
     CONSTRAINT pk_Instructor
          PRIMARY KEY (InstructorID)
)
```

```
CREATE TABLE Instructor_Program
(
     PayRate              INTEGER,
     SupervisorName       VARCHAR(35),
     MaximumHours         NUMBER,
     MinimumHours         NUMBER,
)

CREATE TABLE Registration
(
     RegistrationID       VARCHAR(10) NOT NULL,
     RegistrationNumber   NUMBER,
     RegistrationDate     DATE,
     PaymentAmount        NUMBER,
     PaymentType          VARCHAR(20),
     CONSTRAINT pk_Registration
          PRIMARY KEY (RegistrationID)
)
```

ADDING FOREIGN KEYS
The following SQL codes add foreign keys constraints to link the tables together:

```
ALTER TABLE Registration
     ADD CONSTRAINT fk_registration_student
          FOREIGN KEY (StudentID)
               REFERENCES Student (StudentID)

ALTER TABLE Registration
     ADD CONSTRAINT fk_registration_program
          FOREIGN KEY (ProgramID)
               REFERENCES Program (ProgramID)

ALTER TABLE Activities
     ADD CONSTRAINT fk_activities_program
          FOREIGN KEY (ProgramID)
               REFERENCES Program (ProgramID)
```