Database Systems

Final Project Report

Student Success Tracker

2.0

Developed by Team P.A.T.

April 27, 2020

Terrence Gaines

James E Camon

Kathy Vy Nguyen

# Introduction

Georgia State University is a well-known institution. The IIT department has decided to consult with a team of Seniors in the Computer Science department to modify the existing Tutor Trac database application to make it more user friendly. The Georgia State IIT current Tutor Trac currently allows student on each campus to login into its tutoring center with the Panther identification number and course that they need a tutor for while at the tutoring center. Some administrative features of the current database allow the admin user to pull a report to show the total number of student users to use the center on each day of the week. The application also provides the admin user with budgeting information that shows the total number of hours that the tutor spends in the facility. It also shows the peak time when the center has the most students. The upgraded database application known as the Student Success Tracker 2.0 will allow the student to find the perfect tutor. The database will collect additional data from the student and additional data about the tutor specialties and work schedule. The database will use the information to match the student with the perfect tutor to match their academic needs. Our motivation is the create a better database that will help to improve the academic success of student’s in the classroom with learning concepts, exam performance, and solving class projects better. Some important features of our design are the database will create a profile from each tutor employed by the university like the Employee ID number, name, specialty, and work schedule. On the student side, the application will collect the student identification number, the course, the classification, problem type.

# Requirement Analysis

## Data Requirements

## Functional Requirements

1. Student Class Schedule screen – the function of this entity will retrieve the student schedule to and determine the student’s availability to get help. The function will record the start and end time as well as assigned a unique identification number the student availability results will be compared with the Employee availability to determine the number of student’s availability.
2. Employee Work Schedule availability Screen - This function allows the employee to input their availability to work in the tutoring center. The employee will input the start time, end time, and the day of the week that he or she is available to work.
3. Student Appointment Screen this function will take the student’s schedule and employee schedule and match the student will the perfect tutor to help them in the course. This function will capture the student identification number, type of course, student classification, type of problem. The results of this data will show the student on the screen the available tutors that are availability to give immediate help or our available for a future date.
4. Employee Appointment Screen – The function allows the tutor to review all tutoring appointments to be able to accept or deny the request. The employee Id is capture, first name, last name, role, and specialty will be recorded.
5. Student-User Login Screen This entity function will capture the student identification number, first name, last name, problem type, and course level (2000,3000,4000) The data from this entity will be compared with the employee entity to determine the ideal tutor match to meet the student or group need.
6. Employee-Tutor Profile Screen – the functionality involved in this entity is to capture the employee identification number, first and last name of the employee, the employee type needs to be recorded because the employee may be permanent staff or a work-study worker. This information is necessary for budgeting purposed because it will help to limit the type of administrative functions that the work study worker can do versus the staff worker. Work study student are temporary workers and may not be eligible to work if they are not approved for financial aid. A staff worker may be hired to work permanent in the tutoring center for part time or full-time availability. This data will help to know the total number of work hours the tutor can work on a weekly basis.
7. Student-Group login Screen This function will have a unique group identification number so we can keep track of all the groups that come to the tutoring center for help. The course type, problem type are important data to track. The name of the person does not need to be stored. The only need for name is to identify the group in the facility. The start and end time need to be recorded. This information will help us to determine the best possible time for the group to get help.
8. Create a Session screen – The function in this session will have a unique session identification number, course type and course level are important because we need to know the demand of specific course type that the tutor is helping the students within the tutoring lab. Course level is also important because the school may have multiple campuses and offer specific course levels on different campus. This will help us to not to match the student to a specific campus that only tutor a specific level of course. The time is important to capture because different campus have different hours of availability.
9. Feedback Screen – this function entity will have a unique feedback identification number and will only capture the comments left by the student users.
10. Campus Screen – This function entity will track with campus is desired for access to a tutor. It will also capture the location city.

We have summarized some basic assumptions based on our high-level introduction regarding the database development of the Student Success Tracker Pro (SST) utility. Our EER-model is designed according to those assumptions.

We will assume the following based on the high-level requirements of the SST:

* The characteristics of a student calendar include a unique calendar identification number, a start time and end time.
* The characteristics of a tutor work schedule calendar include a unique employee identification number, the start of the employee shift, the end of the employee shift, the type of employee that is working at the tutoring center. At the tutoring resource facility, we would like to record the type of employee that is working in our facility. We have full time staff, part time staff, and work-study students.
* The characteristics of a user include a unique student identification number, the courses type of the student, the course level, the specific problem the student need help with at the tutoring center.
* The characteristics of a staff profile include a unique employee identification number, the course that the staff can teach, the subject specialty of the staff and the role of the tutor.
* The characteristics of a group profile include an unique group identification number, the course type that the group of students need assistance with at the tutoring lab, the specific problem is identified by the students, and the name of the group is capture, the desired start time for the session is given and the desired end time.
* The characteristics of a session will be a unique session identification number, the start time, the end time, the type of problem is identified, the type of course is identified, and the course level is captured.
* The characteristics of a feedback will include a unique feedback identification number, it will capture comments the student user. The comments from the staff does not need to be recorded.
* The characteristics of a calendar events (student appointments) will include a unique calendar identification number, the stat time and the end time.
* The characteristics of a calendar events (tutor appointments) will include a unique calendar identification number, the stat time and the end time.
* The characteristics of a report will include a unique report identification number, desired start time, desired end time, the PID (what is this for), the EID (what is this for)

# Conceptual Design

In our Report we are showing our entities (and their attributes), and relationships one by one.

### *Entity 1: User Calendar Events*

*Entity*: Describes the student class schedule that list all the courses that the student is registered for the current semester. The purpose of the entity is to display the student’s schedule name on the screen for the student to select the class that they need academic assistance.

*Attributes*

* Start class time: … This attribute captures the start of the class.
* End class time: …. This attribute captures the end of the class.

*Relationships*

* \*Contains: … This relationship exists between the User Calendar Event entity, calendar days entity.
* \*Has: … This relationship exists between the User Calendar Event entity, User entity.

*Primary Key*

* It is identified by User Calendar Event ID attribute, because it will uniquely identify each course.

*Foreign Key*

* It is identified by User ID attribute, because it will uniquely identify each student user

### *Entity 2: Employee Calendar Events*

*Entity*: Describes the employee’s work schedule that list all the days of the week the employee is available to work in the tutoring center. The purpose of the entity is to allow the database to retrieve the schedule to compare to the student’s selected class time.

*Attributes*

* Employee Calendar Event ID
* Start class time: … This attribute captures the start of the employee’s shift.
* End class time: …. This attribute captures the end of the employee’s shift.

*Relationships*

* \*Have: … This relationship exists between Employee’s Calendar Event, and Employee.

*Primary Key*

* It is identified by Calendar Day ID attribute, because it will uniquely identify each work shift.

*Foreign Key*

* It is identified by Employee ID attribute, because it will uniquely identify an employee in the table.

### *Entity 3: User*

*Entity*: The purpose of the entity is to allow the database to collect the student’s information to match up with the perfect tutor

*Attributes*

* First Name: … This attribute captures the student’s first name.
* Last Name: …. This attribute captures the capture the student’s last name.

*Relationships*

* \*Has: … This relationship exists between the User calendar Event enity and User entity.

*Primary Key*

* It is identified by User ID attribute, because it will uniquely identify each student’s records.

### *Entity 4: Calendar Days*

*Entity*: This entity is used by calendar events to give them a unique calander day to reference when scheduling events

*Attributes*

* Date Key: … This attribute captures an alternative way to say a date. It is a short version of date.
* Date: …. This attribute captures the month, day, and year.
* Day of Month: … This attribute captures the month.
* Day Suffix: … This attribute captures the day.
* Day Name: …

*Relationships*

* \*Contains: … This relationship exists between the User calendar Events and User profile.

*Primary Key*

* It is identified by Calendar Day ID attribute, because it will uniquely identify each student’s calendar events.

### *Entity 5: Course Level*

*Entity*: The purpose of the entity is to allow denote the level of a user’s course.

*Attributes*

* Name: … This attribute captures name of the student’s course.
* Value: …. This attribute captures the student’s course number.

*Relationships*

* \*Has: … This relationship exists between the Course Level and Tutoring Sessions.

*Primary Key*

* It is identified by Course Level ID attribute, because it will uniquely identify each student’s records.

### *Entity 6: Employee*

*Entity*: The purpose of the entity is to allow the database to collect the employee’s information to match up with the perfect student

*Attributes*

* First Name: … This attribute captures the employee’s first name
* Last Name: …. This attribute captures the employee’s last name
* Course Type ID: …. This attribute captures the course type ID number
* Role ID: … This attribute captures the assigned role ID number given to the employee
* Specialty Value: … This attribute captures the assigned specialty value
* Course Value: … This attribute captures the assigned course value

*Relationships*

* \*Have: … This relationship exists between the Employee Calendar Events and Employee.

*Primary Key*

* It is identified by Employee ID attribute, because it will uniquely identify each student’s records.

### *Entity 7: Problem Type*

*Entity*: The purpose of the entity is to denote the user’s problem type.

*Attributes*

* Name: … This attribute captures the name of the Course.

*Relationships*

* \*Has: … This relationship exists between the Problem Type and Tutoring Sessions.

*Primary Key*

* It is identified by Problem Type ID attribute, because it will uniquely identify each student’s records.

### *Entity 8: Tutoring Sessions*

*Entity*: The purpose of the entity is to allow the database to describe the session that the student is participate in with the employee.

*Attributes*

* Start Time: … This attribute captures the name of the Course.
* End Time:: …. This attribute captures the capture the course level such as 2000,3000,4000. This will help the database to match the student with the right tutor on the right campus. Different campuses over different course level so this data needs to be recorded.
* Group ID: …. Capture what issue the student have that they need help. The student can select among the list: Teaching, Exam, and Project Assistance
* User ID: … This attribute captures the specific user that is requesting for a session.
* Employee ID: … This attribute captures the specific employee that has been assigned to the session
* Status ID: … this attribute captures an unique status ID number of the user’s session.
* Problem Type ID: … This attribute captures the unique problem type Id number for the user’s session
* Course Type ID: … This attribute captures the unique course type ID number for the user’s session.
* Course Level ID: …This attribute captures the specific course level of the course the user is taken.

*Relationships*

* \*Has & Teaches: … These relationships exists between the Tutoring Sessions, Employee, Campuses, Feedback, and Users

*Primary Key*

* It is identified by Tutoring Session ID attribute, because it will uniquely identify each student’s records.

### *Entity 9: Feedback*

*Entity*: The purpose of the entity is to allow the database to collect the student’s feedback about their session.

*Attributes*

* Comments: … This attribute captures the name of the Course.

*Relationships*

* \*In: … This relationship exists between the User calendar and User profile.

*Primary Key*

* It is identified by Feedback ID attribute, because it will uniquely identify each student’s records.

### *Entity 10: Campuses*

*Entity*: The purpose of the entity is to allow the database to keep track of what campus the user is requesting tutoring services at.

*Attributes*

* Name: … This attribute captures the name of the Course.
* City: …. This attribute captures the capture the course level such as 2000,3000,4000. This will help the database to match the student with the right tutor on the right campus. Different campuses over different course level so this data needs to be recorded.
* State: …. Capture what issue the student have that they need help. The student can select among the list: Teaching, Exam, and Project Assistance

*Relationships*

* \*Takes Place On: … This relationship exists between the Tutoring Sessions and Campuses

*Primary Key*

* It is identified by Campuses ID attribute, because it will uniquely identify each student’s records.

### *Entity 11: Role*

*Entity*: The purpose of the entity is to allow the database to assigned a specific permission to the user given them access to view and perform limited amount of queries.

*Attributes*

* Name: … This attribute captures the name of the Course.
* *Relationships*
* \*Belong to: … This relationship exists between the Role and Employee.

*Primary Key*

* It is identified by Role ID attribute, because it will uniquely identify each student’s records.

### *Entity 12: Specialties*

*Entity*: The purpose of the entity is to allow the database to collect the employee’s information to match up with the right user.

*Attributes*

* Name: … This attribute captures the name of the Course.
* Specialty Value …. This attribute captures a specific value. Each value stand for a specific specialty about the employee. This value is used to match the employee with a perfect user.
* *Relationships*
* \*Have: … This relationship exists between the Specialties and Employee.

*Primary Key*

* It is identified by Specialty ID attribute, because it will uniquely identify each student’s records.

### *Entity 13: User Group*

*Entity*: The purpose of the entity shows what user’s belong to a given group.

*Attributes*

* Group Name: … This attribute captures the name of the group.
* *Relationships*
* \*In: … This relationship exists between the User calendar and User profile.

*Primary Key*

* It is identified by User Group ID attribute, because it will uniquely identify each student’s records.

### *Entity 14: Statuses*

*Entity*: The purpose of the entity is to display the status of a given tutoring session.

*Attributes*

* Name: … This attribute captures the name of the Course.

*Relationships*

* \*In: … This relationship exists between the User calendar and User profile.

*Primary Key*

* It is identified by Statuses ID attribute, because it will uniquely identify each student’s records.

### *Entity 15: Course Type*

*Entity*: The purpose of the entity is to denote the user’s course type.

*Attributes*

* Name: … This attribute captures the name of the Course.
* *Relationships*
* \*In: … This relationship exists between the User calendar and User profile.

*Primary Key*

* It is identified by Course Type ID attribute, because it will uniquely identify each student’s records.

## Relationships

### *Relationship 1: Contains*

*Relation*: Shows the relation between User Calendar Events and Calendar Days

*Attributes*

* SID - the SID attribute will capture the student identification number entered by the student. It is necessary for this attribute to be the primary key because it will help to identify a specific student that is using the database.
* FN (Foreign Key from Entity 2). This attribute will be used for the student’s first name.
* LN – This attribute will be used for the student’s last name.
* PTID – This attribute will assigned a unique problem type identification number for the student.
* ProblemType\_Name – This attribute will be used to capture the desired problem that the student will need help in at the tutoring facility. The student will choose from a variety of options such as: Teaching, Exam Assistance, and, Project Help
* CTID – This attribute will store a course identification number.
* CourseType\_Name – This attribute will store the desired course type that the student will need help in. For example, Computer Science, Math, Science, etc.
* GID – This attribute will be for a group identification number.
* Group\_Name – This attribute will create a name for multiple students that come to the tutoring center together to get help together for the same problem.

**Cardinality Ratios** - The **cardinality ratios** for the relationship between User calendarEvents and Calendar Days specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are many-to-many ratio (M:N). There are many events that have access to many dates in the calendar.

### *Relationship 2: Has*

*Relation*: Shows the relation between User calendar events and User

**Cardinality Ratios** - The **cardinality ratios** for the relationship between User calendar and User specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-many ration (1:N). There is one user that have access to many dates in the calendar to schedule an appointment for academic assistance.

*Relationship 3: Participates*

*Relation*: Shows the relation between Employee Calendar Events and Employee

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Employee calendar events and Employee specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-many ration (1:N). There is one employee that have access to many dates in the calendar to their work schedule when they are available to help students that need academic assistance.

### *Relationship 4: Teaches*

*Relation*: Shows the relation between User and Employee

**Cardinality Ratios** - The **cardinality ratios** for the relationship between User and Employee specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-one ratio (1:N). An employee can only teach one student a given time for the individual teaching session. Cardinality ratio can change depending on the specified teaching session. If the teaching session is a group, the ratio can be a (1:N) ratio.

### *Relationship 5: Belong to*

*Relation*: Shows the relation between Employee and Role

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Employee and Role specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-one ratio (1:1). An employee can only have one assigned role at a given time. The type of role that an employee has will determine how much access the employee has to perform administrative functions of the database.

### *Relationship 6: Have*

*Relation*: Shows the relation Employee and Specialties

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Employee and Specialties specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-one ratio (1:1). An employee may have a given specialty which allows them to offer better academic teaching strategies to the student. Specialties may be determined by employee’s strength in specific education concentrations, skills, and experience.

### *Relationship 7: Takes Place On*

*Relation*: Shows the relation between Campuses and Tutoring Sessions

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Campuses and Tutoring Sessions specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-many ration (1:N). Many campuses may offered multiple tutoring sessions to a student.

### *Relationship 8: Has*

*Relation*: Shows the relation between Feedback and Tutoring Sessions

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Feedback and Tutoring Sessions specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-one ratio (1:1). A student can provide feedback to the tutoring session that they participated in during his or her interaction with the employee (tutor).

### *Relationship 9: In*

*Relation*: Shows the relation between Users and User Groups

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Users and User Groups specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are many-to-one ratio (M:1). There can be many students that belong to given group (related to a given subject matter) to receive tutoring from a given employee

### *Relationship 10: Has*

*Relation*: Shows the relation between Tutoring Session and Problem Type

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Tutoring Session and Problem Type specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-one ratio (1:1). A student can participate in one tutoring session to get help on one specific type of problem. Problem can type can be teaching, exam, or project understanding.

### *Relationship 11: Has*

*Relation*: Shows the relation between Tutoring Session and Course Type

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Tutoring Session and Course Type specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-one ratio (1:1). A student can participate in one tutoring session to get help on one specific given course such as Computer Science, Math, Biology, Chemistry, Physics, or English.

### *Relationship 12: Has*

*Relation*: Shows the relation between Tutoring Session and Course Level

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Tutoring Session and Course Type specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-one ratio (1:1). A student can participate in one tutoring session to get help on one specific given course level (2000, 3000,4000).

### *Relationship 13: Has*

*Relation*: Shows the relation between Tutoring Session and Statuses

**Cardinality Ratios** - The **cardinality ratios** for the relationship between Tutoring Session and Statuses specifies the maximum number of relationship instance in which the entity can participate in. The **cardinality ratios** are one-to-one ratio (1:1). A student can participate in one tutoring session to get help on challenging subject matter. A student can on have one status participating in a tutoring session. Statuses are divided into either two-year degree program or four-year degree program. Different campuses may have limited resources based upon courses offered on a given campus.

# Teamwork

**Terrence Gaines**

Terrence gathered requirement from different students to get additional ideas about what to include in the database. Terrence also gathered different prospective from other campuses to come up with the ideal database solution. Organized the team project schedule for the team to follow to stay on top of key deadlines for the project. Talking with tutors from different campuses to determine new features to improve the existing database. Terence also developed the initial project design report draft that the team use to modify. In addition, he helped to develop the insert.sql file for the implementation phase of the project. Terrence also work with James on the development of the queries.sql file. Terrence also participated in all the presentation practice session organized by Kathy. Terrence also create the initial draft of the final project report.

**James Camon**

James have been reviewing all the ER Models created by each team member and revising each one to compile one main EER model that best represents the team’s idea that we will take as the high-lever requirements and constraints of the project. He has also been analyzing different case study model to see if the new improved functionality of the database design will work and if it is necessary. James also develop the create.sql file and work closely with Terrence on the final modifications needed for both files (create and insert) to run properly. James and Terrence work together to create the queries file to address all the functional requirements of the database. James also participated in all the presentation practice sessions organized by Kathy. James also reviewed the final project report and help to make corrections.

**Kathy Nguyen**

Kathy also created multiple versions of the Relational Schema (table view and graphical view) with her own idea for the team to review. Kathy is also developing the initial and final draft of the Data Dictionary. She has also been analyzing all the database constraints and making improvements to the functionalities of the database. Kathy also create the team’s in class presentation project demo. She organized practice sessions for all the team members to practice the presentation. Kathy also create the project demo video. Kathy also reviewed the final project report and help to make corrections. Kathy also help to review the functional requirements and made any corrections as required.

# Initial ER Model

A picture containing text, map

Description automatically generated

# Initial Relational Schema

User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PID | FN | LN | ProblemType | Course Lvl | CourseType | GID (fk) | RoleID (fk) |

 Staff

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EID | FN | LN | Specialty | RoleID (fk) |

Roles

|  |  |
| --- | --- |
| RoleID | Name |

User Calendar Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CalE\_ID | StartTime | EndTime | PID (fk) | UCID (fk) |

 Staff Calendar Events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CalE\_ID | StartTime | EndTime | EID (fk) | SCID (fk) |

Calendar

|  |  |  |  |
| --- | --- | --- | --- |
| UCID | DayM | DaySuff | DayName |

Campus

|  |  |  |  |
| --- | --- | --- | --- |
| CampusID | Name | Address | SID (fk) |

 Session

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SID | StartTime | EndTime | ProblemType | CourseType | Course Lvl | FID (fk) | PTID (fk) | CTID (fk) | CID (fk) |

ProblemType

|  |  |
| --- | --- |
| PTID | Name |

CourseType

|  |  |
| --- | --- |
| CTID | Name |

Course Lvl

|  |  |
| --- | --- |
| CID | Name |

Feedback

|  |  |
| --- | --- |
| FID | Comments |

A screenshot of a cell phone

Description automatically generated

Data Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table** | **Attribute** | **Data Type** | **Primary Key** | **Foreign Key** | **Constraints** |
| USERS | UserId | uniqueidentifier | YES |  | Unique |
| USERS | FirstName | nvarchar(50) |  |  |  |
| USERS | LastName | nvarchar(50) |  |  |  |
| USERS | RoleId | uniqueidentifier |  | FK |  |
| USERS | GroupId | uniqueidentifier |  | FK |  |
|  |  |  |  |  |  |
| USERGROUPS | UserGroupId | uniqueidentifier | YES |  | Unique |
| USERGROUPS | GroupName | nvarchar(255) |  |  |  |
|  | | | | | |
| EMPLOYEES | EmployeeId | uniqueidentifier | YES |  | Unique |
| EMPLOYEES | FirstName | nvarchar(50) |  |  |  |
| EMPLOYEES | LastName | nvarchar(50) |  |  |  |
| EMPLOYEES | CourseTypeId | uniqueidentifier |  |  |  |
| EMPLOYEES | RoleID | uniqueidentifier |  | FK |  |
| EMPLOYEES | SpecialtyValue | INT |  |  |  |
| EMPLOYEES | CourseValue | INT |  |  |  |
|  | | | | | |
| ROLES | RoleID | uniqueidentifier | YES |  | Unique |
| ROLES | IsDeleted | bit |  |  |  |
| ROLES | Name | varchar(255) |  |  |  |
|  | | | | | |
| CALENDARDAYS | CalendarDayId | uniqueidentifier | YES |  | Unique |
| CALENDARDAYS | Date | datetime |  |  |  |
| CALENDARDAYS | DayOfMonth | nvarchar(50) |  |  |  |
| CALENDARDAYS | DaySuffix | nvarchar(50) |  |  |  |
| CALENDARDAYS | DayName | nvarchar(50) |  |  |  |
| CALENDARDAYS | DateKey | nvarchar(50) |  |  |  |
|  | | | | | |
| CAMPUSES | CampusId | uniqueidentifier | YES |  | Unique |
| CAMPUSES | Name | nvarchar(255) |  |  |  |
| CAMPUSES | City | nvarchar(100) |  |  |  |
| CAMPUSES | State | nvarchar(50) |  |  |  |
|  |  |  |  |  |  |
| USEREVENTSCALENDAR | UserCalendarEventId | uniqueidentifier | YES |  | Unique |
| USEREVENTSCALENDAR | StartTime | time(7) |  |  |  |
| USEREVENTSCALENDAR | EndTime | time(7) |  |  |  |
| USEREVENTSCALENDAR | CalendarDayId | uniqueidentifier |  | FK |  |
| USEREVENTSCALENDAR | UserId | uniqueidentifier |  | FK |  |
|  | | | | | |
| CAMPUS | CampusID | INT | YES |  | Unique |
| CAMPUS | Name | VARCHAR(25) |  |  |  |
| CAMPUS | Address | VARCHAR(100) |  |  |  |
| CAMPUS | SID | INT |  | SESSION(SID) |  |
|  | | | | | |
| TUTORING SESSIONS | TutoringSessionId | uniqueidentifier | YES |  | Unique |
| TUTORING SESSIONS | StartTime | datetime |  |  |  |
| TUTORING SESSIONS | EndTime | datetime |  |  |  |
| TUTORING SESSIONS | GroupId | uniqueidentifier |  | FK |  |
| TUTORING SESSIONS | UserId | uniqueidentifier |  | FK |  |
| TUTORING SESSIONS | EmployeeId | uniqueidentifier |  | FK |  |
| TUTORING SESSIONS | StatusId | uniqueidentifier |  | FK |  |
| TUTORING SESSIONS | ProblemTypeId | uniqueidentifier |  | FK |  |
| TUTORING SESSIONS | CourseTypeId | uniqueidentifier |  | FK |  |
| TUTORING SESSIONS | CourseLevelId | uniqueidentifier |  | FK |  |
|  | | | | | |
| PROBLEMTYPES | ProblemTypeId | uniqueidentifier | YES |  | Unique |
| PROBLEMTYPES | NAME | varchar(255) |  |  |  |
|  | | | | | |
| COURSETYPE | CourseTypeId | uniqueidentifier | YES |  | Unique |
| COURSETYPE | NAME | varchar(255) |  |  |  |
|  | | | | | |
| COURSELVL | CourseLevelId | uniqueidentifier | YES |  | Unique |
| COURSELVL | NAME | varchar(255) |  |  |  |
| COURSELVL | CourseValue | int |  |  |  |
|  | | | | | |
| FEEDBACK | FeedbackId | uniqueidentifier | YES |  | Unique |
| FEEDBACK | Comments | nvarchar(MAX) |  |  |  |
| FEEDBACK | ScopeId | uniqueidentifier |  |  |  |
|  |  |  |  |  |  |
| EMPLOYEE | EmployeeCalendarEventId |  | YES |  | Unique |
| CALENDAR | StartTime | time(7) |  |  |  |
| EVENTS | EndTime | time(7) |  |  |  |
| ---- | CalendarDayId | uniqueidentifier |  | FK |  |
| ---- | EmployeeId | uniqueidentifier |  | FK |  |
|  |  |  |  |  |  |
| SPECIALTIES | SpecialtyId | uniqueidentifier | YES |  | Unique |
| SPECIALTIES | Name | nvarchar(50) |  |  |  |
| SPECIALTIES | SpecialtyValue | int |  |  |  |
|  |  |  |  |  |  |
| STATUSES | StatusId | uniqueidentifier | YES |  | Unique |
| STATUSES | Name | nvarchar(MAX) |  |  |  |
|  |  |  |  |  |  |
| TEACHES | TeachesId | uniqueidentifier | YES |  | Unique |
| TEACHES | EmployeeId | uniqueidentifier |  | FK |  |
| TEACHES | UserId | uniqueidentifier |  | FK |  |
| TEACHES | GroupId | uniqueidentifier |  | FK |  |
| TEACHES | TutoringSessionId | uniqueidentifier |  | FK |  |
|  |  |  |  |  |  |

# Project Summary

***key findings***

* *current active database only tracks the number of users that attends the center*
* *allows student to receive help from available tutor at that time.*

***challenges***

* *student may login for help but not get their academic performance needs met*
* *tutor schedule at that time are not able to assist the student*

***significance of your database***

* *paints a better actual picture of what the student needs help for the course*
* *tutor can help students that closely match their own individual specialties*

Appendix: (attached Final EER Model, Final Relational Schema, SQL code, Optional screen shots of the DB)

# Final EER Model

A close up of a map

Description automatically generated

# Final Relational Schema

Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | FirstName | LastName | GroupID(fk) | RoleID (fk) |

Employees

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EmployeeId | FirstName | LastName | CourseTypeId | RoleID(fk) | SpecialtyValue | CourseValue |

Roles

|  |  |  |
| --- | --- | --- |
| RoleID | IsDeleted | Name |

UserCalendarEvents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserCalendarEventId | StartTime | EndTime | CalendarDayId (fk) | UserId (fk) |

EmployeeCalendarEvents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EmployeeCalendarEventId | StartTime | EndTime | CalendarDayId (fk) | EmployeeId (fk) |

CalendarDays

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CalendarDayId | DateKey | Date | DayOfMonth | DaySuffix | DayName |

Campuses

|  |  |  |  |
| --- | --- | --- | --- |
| CampusID | Name | City | State |

TutoringSessions

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TutoringSessionId | StartTime | EndTime | GroupId (FK) | UserId (FK) | EmployeeId (FK) | StatusId (FK) | ProblemTypeId (FK) | CourseTypeId (FK) | CourseLevelId (FK) |

ProblemTypes

|  |  |
| --- | --- |
| ProblemTypeId | Name |

CourseTypes

|  |  |
| --- | --- |
| CourseTypeId | Name |

CourseLevels

|  |  |  |
| --- | --- | --- |
| CourseLevelId | Name | CourseValue |

Feedback

|  |  |  |
| --- | --- | --- |
| FeedbackId | Comments | ScopeId |

Specialties

|  |  |  |
| --- | --- | --- |
| SpecialtyId | Name | SpecialtyValue |

Statuses

|  |  |
| --- | --- |
| StatusId | Name |

UserGroups

|  |  |
| --- | --- |
| UserGroupId | GroupName |

Teaches

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TeachesId | EmployeeId (FK) | UserId(FK) | GroupId(FK) | TutoringSessionId(FK) |

A screenshot of a cell phone

Description automatically generated

Normalization

After carefully reviewing each relation of our relational schema, we have determined that our schema fits into a pure 3NF Normalization. Our relational schema was in the 3NF form by performing the following actions. Step one: we made sure to create a new relation with the determinant and everything dependent upon it. Step two: We remove everything dependent upon the determinant from the original relation and the last step we added any foreign key constraint to make sure that our final relational schema is in the 3NF.

# SQL Program Code – create table

USE [Boogs] /\*\*\* CHANGE TO DB NAME PLEASE\*\*\*/

GO

/\*\*\*\*\*\* Object: Table [dbo].[CalendarDays] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CalendarDays](

[DateKey] [nvarchar](50) NOT NULL,

[CalendarDayId] [uniqueidentifier] NOT NULL,

[Date] [datetime] NOT NULL,

[DayOfMonth] [nvarchar](50) NOT NULL,

[DaySuffix] [nvarchar](50) NOT NULL,

[DayName] [nvarchar](50) NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Campuses] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Campuses](

[CampusId] [uniqueidentifier] NOT NULL,

[Name] [nvarchar](255) NOT NULL,

[City] [nvarchar](100) NULL,

[State] [nvarchar](50) NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CourseLevels] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CourseLevels](

[CourseLevelId] [uniqueidentifier] NOT NULL,

[Name] [varchar](255) NOT NULL,

[CourseValue] [int] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CourseTypes] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CourseTypes](

[CourseTypeId] [uniqueidentifier] NOT NULL,

[Name] [varchar](255) NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Employees] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Employees](

[EmployeeId] [uniqueidentifier] NOT NULL,

[FirstName] [nvarchar](50) NOT NULL,

[LastName] [nvarchar](50) NOT NULL,

[CourseTypeId] [uniqueidentifier] NOT NULL,

[RoleID] [nvarchar](50) NOT NULL,

[SpecialtyValue] [int] NOT NULL,

[CourseValue] [int] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[EmployeeCalendarEvents] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[EmployeeCalendarEvents](

[EmployeeCalendarEventId] [uniqueidentifier] NOT NULL,

[StartTime] [time] NOT NULL,

[EndTime] [time] NOT NULL,

[CalendarDayId] [uniqueidentifier] NOT NULL,

[EmployeeId] [uniqueidentifier] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Feedback] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Feedback](

[FeedbackId] [uniqueidentifier] NOT NULL,

[Comments] [nvarchar](max) NOT NULL,

[ScopeId] [nvarchar](50) NULL

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[ProblemTypes] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ProblemTypes](

[ProblemTypeId] [uniqueidentifier] NOT NULL,

[Name] [varchar](255) NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Teaches] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Teaches](

[TeachesId] [uniqueidentifier] NOT NULL,

[EmployeeId] [uniqueidentifier] NOT NULL,

[UserId] [uniqueidentifier] NULL,

[GroupId] [uniqueidentifier] NULL,

[TutoringSessionId] [uniqueidentifier] NOT NULL

) ON [PRIMARY]

/\*\*\*\*\*\* Object: Table [dbo].[Roles] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Roles](

[RoleId] [uniqueidentifier] NOT NULL,

[IsDeleted] [bit] NOT NULL,

[Name] [varchar](255) NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Specialties] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Specialties](

[SpecialtyId] [uniqueidentifier] NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[SpecialtyValue] [int] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Statuses] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Statuses](

[StatusId] [uniqueidentifier] NOT NULL,

[Name] [nvarchar](max) NOT NULL

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[TutoringSessions] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[TutoringSessions](

[TutoringSessionId] [uniqueidentifier] NOT NULL,

[StartTime] [datetime] NOT NULL,

[EndTime] [datetime] NOT NULL,

[GroupId] [uniqueidentifier] NULL,

[UserId] [uniqueidentifier] NULL,

[EmployeeId] [uniqueidentifier] NOT NULL,

[StatusId] [uniqueidentifier] NOT NULL,

[ProblemTypeId] [uniqueidentifier] NOT NULL,

[CourseTypeId] [uniqueidentifier] NOT NULL,

[CourseLevelId] [uniqueidentifier] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[UserCalendarEvents] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[UserCalendarEvents](

[UserCalendarEventId] [uniqueidentifier] NOT NULL,

[StartTime] [time] NOT NULL,

[EndTime] [time] NOT NULL,

[CalendarDayId] [uniqueidentifier] NOT NULL,

[UserId] [uniqueidentifier] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[UserGroups] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[UserGroups](

[UserGroupId] [uniqueidentifier] NOT NULL,

[GroupName] [nvarchar](255) NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Users] Script Date: 4/12/2020 9:01:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Users](

[UserId] [uniqueidentifier] NOT NULL,

[FirstName] [nvarchar](50) NOT NULL,

[LastName] [nvarchar](50) NOT NULL,

[RoleId] [uniqueidentifier] NULL,

[GroupId] [uniqueidentifier] NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[CourseLevels] ADD CONSTRAINT [DF\_Table\_1\_CourseTypeId] DEFAULT (newsequentialid()) FOR [CourseLevelId]

GO

ALTER TABLE [dbo].[CourseTypes] ADD CONSTRAINT [DF\_Table\_1\_ProblemTypeId] DEFAULT (newsequentialid()) FOR [CourseTypeId]

GO

ALTER TABLE [dbo].[ProblemTypes] ADD CONSTRAINT [DF\_Table\_1\_RoleId] DEFAULT (newsequentialid()) FOR [ProblemTypeId]

GO

ALTER TABLE [dbo].[Roles] ADD CONSTRAINT [DF\_Roles\_RoleId] DEFAULT (newsequentialid()) FOR [RoleId]

GO

ALTER TABLE [dbo].[Roles] ADD CONSTRAINT [DF\_Roles\_IsDeleted] DEFAULT ((0)) FOR [IsDeleted]

GO

# SQL Program Code – insert

IF NOT EXISTS ( select \* from users where FirstName = 'Boogey' )

BEGIN

/\*Student Users\*/

Insert into Users (UserId, FirstName, LastName) Values

('00000000-0000-0000-0000-000000000001','Alex','Smith'),

('00000000-0000-0000-0000-000000000002','Sara','Dickerson'),

('00000000-0000-0000-0000-000000000003','Terrence','Gaines'),

('00000000-0000-0000-0000-000000000004','Boogey','Camon'),

('00000000-0000-0000-0000-000000000005','Kathy','Nguyen'),

('00000000-0000-0000-0000-000000000006','John','Patterson'),

('00000000-0000-0000-0000-000000000007','Pierette','Barge'),

('00000000-0000-0000-0000-000000000008','Lakeesha','Patterson'),

('00000000-0000-0000-0000-000000000009','Natasha','Mullings'),

('00000000-0000-0000-0000-000000000010','Ramisa','Chowdhury'),

('00000000-0000-0000-0000-000000000011','Alsdex','Smith'),

('00000000-0000-0000-0000-000000000012','Sasdra','Dickerson'),

('00000000-0000-0000-0000-000000000013','Terrasdence','Gaines'),

('00000000-0000-0000-0000-000000000014','Boasdogey','Camon'),

('00000000-0000-0000-0000-000000000015','Kaasdthy','Nguyen'),

('00000000-0000-0000-0000-000000000016','Joasdhn','Patterson'),

('00000000-0000-0000-0000-000000000017','Piasderette','Barge'),

('00000000-0000-0000-0000-000000000018','Lakeasdesha','Patterson'),

('00000000-0000-0000-0000-000000000019','Nataasdsha','Mullings'),

('00000000-0000-0000-0000-000000000020','Ramasdisa','Chowdhury'),

('00000000-0000-0000-0000-000000000021','Alasdex','Smith'),

('00000000-0000-0000-0000-000000000022','Sarasda','Dickerson'),

('00000000-0000-0000-0000-000000000023','Teasdrrence','Gaines'),

('00000000-0000-0000-0000-000000000024','Boasdogey','Camon'),

('00000000-0000-0000-0000-000000000025','Katasdhy','Nguyen'),

('00000000-0000-0000-0000-000000000026','Johasdn','Patterson'),

('00000000-0000-0000-0000-000000000027','Pieasdrette','Barge'),

('00000000-0000-0000-0000-000000000028','Lakeasesha','Patterson'),

('00000000-0000-0000-0000-000000000029','Natasha','Mullings'),

('00000000-0000-0000-0000-000000000030','Ramisa','Chowdhury'),

('00000000-0000-0000-0000-000000000031','Roger','Smith'),

('00000000-0000-0000-0000-000000000032','Jennifer','Smackerson'),

('00000000-0000-0000-0000-000000000033','Melody','Johns'),

('00000000-0000-0000-0000-000000000034','Jasmine','Masterson'),

('00000000-0000-0000-0000-000000000035','Cheryl','Moodyson'),

('00000000-0000-0000-0000-000000000036','Tonisha','Walker'),

('00000000-0000-0000-0000-000000000037','Tawanda','Vikens'),

('00000000-0000-0000-0000-000000000038','Leonard','Njoku'),

('00000000-0000-0000-0000-000000000039','Precious','Ojiku'),

('00000000-0000-0000-0000-000000000040','Lauren','Smitherson'),

('00000000-0000-0000-0000-000000000041','Johanna','Beatty'),

('00000000-0000-0000-0000-000000000042','Daniel','Gaines'),

('00000000-0000-0000-0000-000000000043','Uche Jude','Anyaorah'),

('00000000-0000-0000-0000-000000000044','Kathy-Lee','James'),

('00000000-0000-0000-0000-000000000045','Marcus','Nguyen'),

('00000000-0000-0000-0000-000000000046','Johnny','Verse'),

('00000000-0000-0000-0000-000000000047','Pam','Barnes'),

('00000000-0000-0000-0000-000000000048','Lever','Lilkerson'),

('00000000-0000-0000-0000-000000000049','Nic','Jamal'),

('00000000-0000-0000-0000-000000000050','Ashley','Clarke'),

('00000000-0000-0000-0000-000000000051','Alan','Kann'),

('00000000-0000-0000-0000-000000000052','Sara','Gannson'),

('00000000-0000-0000-0000-000000000053','Timothy','Ness'),

('00000000-0000-0000-0000-000000000054','Brian','Cannon'),

('00000000-0000-0000-0000-000000000055','Kim','Yenson'),

('00000000-0000-0000-0000-000000000056','Jim','Silson'),

('00000000-0000-0000-0000-000000000057','Pamela','Bann'),

('00000000-0000-0000-0000-000000000058','Larry','Barnes'),

('00000000-0000-0000-0000-000000000059','Liz','Aqui'),

('00000000-0000-0000-0000-000000000060','Mohamed','Ali'),

('00000000-0000-0000-0000-000000000061','Alexa','Johnson'),

('00000000-0000-0000-0000-000000000062','Seeway','Vic'),

('00000000-0000-0000-0000-000000000063','Travis','Mullings'),

('00000000-0000-0000-0000-000000000064','Cindy','Camel'),

('00000000-0000-0000-0000-000000000065','Kimberly','Nguyen'),

('00000000-0000-0000-0000-000000000066','Jane','Patel'),

('00000000-0000-0000-0000-000000000067','Bridgette','Mollins'),

('00000000-0000-0000-0000-000000000068','Karen','Rafane'),

('00000000-0000-0000-0000-000000000069','Nakeisha','Donaldson'),

('00000000-0000-0000-0000-000000000070','Marcus','Robbins'),

('00000000-0000-0000-0000-000000000071','Vicki','Vonle'),

('00000000-0000-0000-0000-000000000072','Apple Lee','Camon'),

('00000000-0000-0000-0000-000000000073','Matthew','Culberson'),

('00000000-0000-0000-0000-000000000074','Kirby','Sanderson'),

('00000000-0000-0000-0000-000000000075','Emily','Gagson'),

('00000000-0000-0000-0000-000000000076','Jimmy','valdez'),

('00000000-0000-0000-0000-000000000077','Rome','Eeaton'),

('00000000-0000-0000-0000-000000000078','Spirit','Peoples'),

('00000000-0000-0000-0000-000000000079','Tommy','Banks'),

('00000000-0000-0000-0000-000000000080','Alicia','Carter'),

('00000000-0000-0000-0000-000000000081','Van','Driver'),

('00000000-0000-0000-0000-000000000082','Barney','Kroger'),

('00000000-0000-0000-0000-000000000083','Mark','Morgan'),

('00000000-0000-0000-0000-000000000084','Jennifer','Ransom'),

('00000000-0000-0000-0000-000000000085','Sharon','Gables'),

('00000000-0000-0000-0000-000000000086','Susan','Mulls'),

('00000000-0000-0000-0000-000000000087','Andrea','Peterchily'),

('00000000-0000-0000-0000-000000000088','Sessy','Patterson'),

('00000000-0000-0000-0000-000000000089','Harry','Smith'),

('00000000-0000-0000-0000-000000000090','Vicki','Barnett'),

('00000000-0000-0000-0000-000000000091','Georgia','Lovett'),

('00000000-0000-0000-0000-000000000092','Lorainne','Terrace'),

('00000000-0000-0000-0000-000000000093','Artist','Gann'),

('00000000-0000-0000-0000-000000000094','Jeff','Covid'),

('00000000-0000-0000-0000-000000000095','Ron','Emory'),

('00000000-0000-0000-0000-000000000096','Juan','Luiz'),

('00000000-0000-0000-0000-000000000097','Ranisa','Carter'),

('00000000-0000-0000-0000-000000000098','Clarke','Milsap'),

('00000000-0000-0000-0000-000000000099','Mindy','Chin'),

('00000000-0000-0000-0000-000000000100','Ekwenchi','Chinonye');

Insert into UserGroups (UserGroupId, GroupName ) Values

('50000000-0000-0000-0000-000000000001','CS Group Study 1'),

('50000000-0000-0000-0000-000000000002','CS Group Study 2'),

('50000000-0000-0000-0000-000000000003','CS Group Study 3'),

('50000000-0000-0000-0000-000000000004','CS Group Study 4'),

('50000000-0000-0000-0000-000000000005','CS Group Study 5'),

('50000000-0000-0000-0000-000000000006','CS Group Study 6'),

('50000000-0000-0000-0000-000000000007','CS Group Study 7'),

('50000000-0000-0000-0000-000000000008','CS Group Study 8'),

('50000000-0000-0000-0000-000000000009','CS Group Study 9'),

('50000000-0000-0000-0000-000000000010','CS Group Study 10'),

('40000000-0000-0000-0000-000000000001','Math Group Study 11'),

('40000000-0000-0000-0000-000000000002','Math Group Study 12'),

('40000000-0000-0000-0000-000000000003','Math Group Study 13'),

('40000000-0000-0000-0000-000000000004','Math Group Study 14'),

('40000000-0000-0000-0000-000000000005','Math Group Study 15'),

('40000000-0000-0000-0000-000000000006','Math Group Study 16'),

('40000000-0000-0000-0000-000000000007','Math Group Study 17'),

('40000000-0000-0000-0000-000000000008','Math Group Study 18'),

('40000000-0000-0000-0000-000000000009','Math Group Study 19'),

('40000000-0000-0000-0000-000000000010','Math Group Study 20'),

('30000000-0000-0000-0000-000000000001','Physics Group Study 1'),

('30000000-0000-0000-0000-000000000002','Physics Group Study 2'),

('30000000-0000-0000-0000-000000000003','Physics Group Study 3'),

('30000000-0000-0000-0000-000000000004','Physics Group Study 4'),

('30000000-0000-0000-0000-000000000005','Physics Group Study 5'),

('30000000-0000-0000-0000-000000000006','Physics Group Study 6'),

('30000000-0000-0000-0000-000000000007','Physics Group Study 7'),

('30000000-0000-0000-0000-000000000008','Physics Group Study 8'),

('30000000-0000-0000-0000-000000000009','Physics Group Study 9'),

('30000000-0000-0000-0000-000000000010','Physics Group Study 10'),

('20000000-0000-0000-0000-000000000001','Biology/Chemistry Group Study 11'),

('20000000-0000-0000-0000-000000000002','Biology/Chemistry Group Study 1'),

('20000000-0000-0000-0000-000000000003','Biology/Chemistry Group Study 2'),

('20000000-0000-0000-0000-000000000004','Biology/Chemistry Group Study 3'),

('20000000-0000-0000-0000-000000000005','Biology/Chemistry Group Study 4'),

('20000000-0000-0000-0000-000000000006','Biology/Chemistry Group Study 5'),

('20000000-0000-0000-0000-000000000007','Biology/Chemistry Group Study 6'),

('20000000-0000-0000-0000-000000000008','Biology/Chemistry Group Study 7'),

('20000000-0000-0000-0000-000000000009','Biology/Chemistry Group Study 8'),

('20000000-0000-0000-0000-000000000010','Biology/Chemistry Group Study 9'),

('10000000-0000-0000-0000-000000000001','English Group Study 1'),

('10000000-0000-0000-0000-000000000002','English Group Study 2'),

('10000000-0000-0000-0000-000000000003','English Group Study 3'),

('10000000-0000-0000-0000-000000000004','English Group Study 4'),

('10000000-0000-0000-0000-000000000005','English Group Study 5'),

('10000000-0000-0000-0000-000000000006','English Group Study 6'),

('10000000-0000-0000-0000-000000000007','English Group Study 7'),

('10000000-0000-0000-0000-000000000008','English Group Study 8'),

('10000000-0000-0000-0000-000000000009','English Group Study 9'),

('10000000-0000-0000-0000-000000000010','English Group Study 10');

/\*Employee Users\*/

Insert into Employees(EmployeeId,FirstName,LastName, CourseTypeId, RoleID, SpecialtyValue, CourseValue) Values

('10000000-0000-0000-0000-000000000001','Hannah','Johnson','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000003', 1, 2),

('10000000-0000-0000-0000-000000000002','Kevin','Horning','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000003', 1, 2),

('10000000-0000-0000-0000-000000000003','Kathy','Nguyen','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000002', 1, 2),

('10000000-0000-0000-0000-000000000004','Terrence','Gaines','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000003', 65, 9),

('10000000-0000-0000-0000-000000000005','Boogey','Camon','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000002', 1, 2),

('10000000-0000-0000-0000-000000000006','John','Doe','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000001', 1, 2),

('10000000-0000-0000-0000-000000000007','Natasha','Mullings','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000001', 1, 2),

('10000000-0000-0000-0000-000000000008','Samedy','Yong','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000001', 1, 2),

('10000000-0000-0000-0000-000000000009','Jessica','Smith','12000000-0000-0000-0000-000000000001','A0000000-0000-0000-0000-000000000001', 1, 2);

/\*Database Administrative Roles\*/

Insert into Roles(RoleID,Name) Values

('A0000000-0000-0000-0000-000000000001','Student'),

('A0000000-0000-0000-0000-000000000002','Admin'),

('A0000000-0000-0000-0000-000000000003','Teacher'),

('A0000000-0000-0000-0000-000000000004','Other');

/\*This section identifies the specialty that the tutor can better help the student in\*/

Insert into Specialties(SpecialtyID,Name, SpecialtyValue) Values

('B0000000-0000-0000-0000-000000000001','Projects',15),

('B0000000-0000-0000-0000-000000000002','Exams', 50),

('B0000000-0000-0000-0000-000000000004','Other', 999);

Insert into UserCalendarEvents (UserCalendarEventId,StartTime,EndTime,CalendarDayId,UserId) Values

('C0000000-0000-0000-0000-000000000001','1900-01-01 17:30:00.000','1900-01-01 19:00:00.000','D0000000-0000-0000-0000-000000000001','00000000-0000-0000-0000-000000000004'), -- mon

('C0000000-0000-0000-0000-000000000002','1900-01-02 12:45:00.000','1900-01-01 14:00:00.000','D0000000-0000-0000-0000-000000000002','00000000-0000-0000-0000-000000000004'), -- tues

('C0000000-0000-0000-0000-000000000003','1900-01-02 14:45:00.000','1900-01-01 16:30:00.000','D0000000-0000-0000-0000-000000000002','00000000-0000-0000-0000-000000000004'), -- tues

('C0000000-0000-0000-0000-000000000004','1900-01-03 17:30:00.000','1900-01-01 19:00:00.000','D0000000-0000-0000-0000-000000000003','00000000-0000-0000-0000-000000000004'), -- wend

('C0000000-0000-0000-0000-000000000005','1900-01-04 12:45:00.000','1900-01-01 14:00:00.000','D0000000-0000-0000-0000-000000000004','00000000-0000-0000-0000-000000000004'), -- th

('C0000000-0000-0000-0000-000000000006','1900-01-04 14:45:00.000','1900-01-01 16:30:00.000','D0000000-0000-0000-0000-000000000004','00000000-0000-0000-0000-000000000004'), -- th

('C0000000-0000-0000-0000-000000000007','1900-01-04 18:00:00.000','1900-01-01 18:50:00.000','D0000000-0000-0000-0000-000000000004','00000000-0000-0000-0000-000000000004'); -- th

/\*Record of the employee work availability to help the student user\*/

Insert into EmployeeCalendarEvents (EmployeeCalendarEventId,StartTime,EndTime,CalendarDayId,EmployeeId) Values

('C0000000-0000-0000-0000-000000000001','1900-01-01 09:00:00.000','1900-01-01 13:00:00.000','D0000000-0000-0000-0000-000000000001','10000000-0000-0000-0000-000000000004'), -- mon

('C0000000-0000-0000-0000-000000000002','1900-01-02 08:00:00.000','1900-01-02 12:00:00.000','D0000000-0000-0000-0000-000000000002','10000000-0000-0000-0000-000000000004'), -- tues

('C0000000-0000-0000-0000-000000000003','1900-01-03 08:00:00.000','1900-01-03 12:00:00.000','D0000000-0000-0000-0000-000000000003','10000000-0000-0000-0000-000000000004'), -- wed

('C0000000-0000-0000-0000-000000000004','1900-01-04 08:00:00.000','1900-01-04 11:00:00.000','D0000000-0000-0000-0000-000000000004','10000000-0000-0000-0000-000000000004'), -- thur

('C0000000-0000-0000-0000-000000000005','1900-01-05 08:00:00.000','1900-01-05 12:00:00.000','D0000000-0000-0000-0000-000000000005','10000000-0000-0000-0000-000000000004') -- frid

Insert into CalendarDays (DateKey,CalendarDayId,[Date],[DayOfMonth],DaySuffix,[DayName]) Values

('20000101','D0000000-0000-0000-0000-000000000001','2000-01-01 00:00:00.000','1','1st', 'Sunday'),

('20000102','D0000000-0000-0000-0000-000000000002','2000-01-02 00:00:00.000','2','2nd', 'Monday'),

('20000103','D0000000-0000-0000-0000-000000000003','2000-01-03 00:00:00.000','3','3rd', 'Tuesday'),

('20000104','D0000000-0000-0000-0000-000000000004','2000-01-04 00:00:00.000','4','4th', 'Wednesday'),

('20000105','D0000000-0000-0000-0000-000000000005','2000-01-05 00:00:00.000','5','5th', 'Thursday'),

('20000106','D0000000-0000-0000-0000-000000000006','2000-01-06 00:00:00.000','6','6th', 'Friday'),

('20000107','D0000000-0000-0000-0000-000000000007','2000-01-07 00:00:00.000','7','7th', 'Saturday');

Insert into Campuses (CampusID, Name, City, State) Values

('CA000000-0000-0000-0000-000000000001','Atlanta','Atlanta','GA'),

('CA000000-0000-0000-0000-000000000002','Clarkston','Clarkston','GA'),

('CA000000-0000-0000-0000-000000000003','Dunwoody','Dunwoody','GA'),

('CA000000-0000-0000-0000-000000000004','Newton','Newton','GA'),

('CA000000-0000-0000-0000-000000000005','Decatur','Decatur','GA');

Insert into ProblemTypes (ProblemTypeId,Name) Values

('11000000-0000-0000-0000-000000000001','Tutor'),

('11000000-0000-0000-0000-000000000002','Exam'),

('11000000-0000-0000-0000-000000000003','Project');

Insert into CourseTypes (CourseTypeId,Name) Values

('12000000-0000-0000-0000-000000000001','Computer Science'),

('12000000-0000-0000-0000-000000000002','Math'),

('12000000-0000-0000-0000-000000000003','Biology'),

('12000000-0000-0000-0000-000000000004','Chemistry'),

('12000000-0000-0000-0000-000000000005','Physics'),

('12000000-0000-0000-0000-000000000006','English');

Insert into CourseLevels(CourseLevelId, Name, CourseValue) Values

('13000000-0000-0000-0000-000000000001', '2000', 2),

('13000000-0000-0000-0000-000000000002', '3000', 15),

('13000000-0000-0000-0000-000000000003', '4000', 40);

Insert into Feedback(FeedbackId,Comments) Values

('F0000000-0000-0000-0000-000000000001','tutor - great job'),

('F0000000-0000-0000-0000-000000000002','tutor - poor job'),

('F0000000-0000-0000-0000-000000000003','not enough tutors to help'),

('F0000000-0000-0000-0000-000000000004','front desk attendant not friendly'),

('F0000000-0000-0000-0000-000000000005','front desk attendant gave great service');

/\*StatusTypes for TutoringSessions\*/

Insert into Statuses(StatusId, Name) Values

('90000000-0000-0000-0000-000000000001','Pending'),

('90000000-0000-0000-0000-000000000002','Accpeted'),

('90000000-0000-0000-0000-000000000003','Denied')

Insert into TutoringSessions (TutoringSessionId,StartTime,EndTime,GroupId,UserId,EmployeeId,StatusId,ProblemTypeId,CourseTypeId,CourseLevelId) Values

('70000000-0000-0000-0000-000000000001','2018-08-07 11:00:00.000','2018-08-07 13:00:00.000',Null,'10000000-0000-0000-0000-000000000004','10000000-0000-0000-0000-000000000002','90000000-0000-0000-0000-000000000002','11000000-0000-0000-0000-000000000001','12000000-0000-0000-0000-000000000001','13000000-0000-0000-0000-000000000001');

Insert into Teaches (TeachesId,EmployeeId,UserId,GroupId,TutoringSessionId) Values

('60000000-0000-0000-0000-000000000001','90000000-0000-0000-0000-000000000002','10000000-0000-0000-0000-000000000004','00000000-0000-0000-0000-000000000004','70000000-0000-0000-0000-000000000001')

/\*Groups for Users\*/

Insert into UserGroups(UserGroupId,GroupName) Values

('00000000-0000-0000-0000-000000000004','Cool Group Name')

--Add Boogey to group

update users

set GroupId = '50000000-0000-0000-0000-000000000001'

where UserId = '00000000-0000-0000-0000-000000000004'

END

# SQL Program Code – Queries

-- TO set the scence for number 1 the goal is to analyze every student schedule to process how much time they will have throughout their day for availbale tutoring times..

-- This can help rid away of massive amounts of stress from being overwhlemd wtih calsses and will point out where students have breaks.. It will gauge thier schedule and with a static

-- buffer zone of walking to and from the center will prompt tutoring sessiosn ( this would be permanatley dismissable of course ).

1---- Case 1: Student has oppurtunity to go to to tutoring before his/her classes starts,

-- this will be cross refrenced with a tutors schedule to try and pair up with the perfect tutor

select (DATEDIFF( minute , MAX(U.EndTime) , '19:00:00.0000000') ) AS TimeAfterClass

from UserCalendarEvents as U

where UserId = '00000000-0000-0000-0000-000000000004'

and U.CalendarDayId = 'D0000000-0000-0000-0000-000000000004'

1---- Case 2: Student has time after class to stop by and get tutoring if need be

select

t1.UserId,

t1.StartTime,

t1.EndTime,

datediff(minute, max(t2.EndTime), t1.StartTime) as minutes

from UserCalendarEvents t1

left join UserCalendarEvents t2

on t1.UserId = t2.UserId

and t2.EndTime < t1.StartTime

group by t1.UserId, t1.StartTime, t1.EndTime

1---- Case 3: This query determines if the student would have suffiecent time in

-- order to seek tutoring help.. pairing this with front end calculations.

select (DATEDIFF( minute , MAX(U.EndTime) , '19:00:00.0000000') ) AS TimeAfterClass

from UserCalendarEvents as U

where UserId = '00000000-0000-0000-0000-000000000004'

and U.CalendarDayId = 'D0000000-0000-0000-0000-000000000004'

2-- To view employee availabilty this query will display the start and end times of thier shifts as well as the day.. This query will work will a partial search implementation as shown below.

select StartTime, EndTime, DaySuffix, Date

from EmployeeCalendarEvents as E, CalendarDays as C

where E.CalendarDayId = C.CalendarDayId

and EmployeeId = ( select EmployeeId from Employees where FirstName like 'terr%' )

3 Skipped /\*(refer to notes at the top\*/

2-- To view users availabilty this query will display the start and end times of thier shifts as well as the day.. This query will work will a partial search implementation as shown below.

select StartTime, EndTime, DaySuffix, Date

from UserCalendarEvents as E, CalendarDays as C

where E.CalendarDayId = C.CalendarDayId

and UserId = ( select UserId from Users where FirstName like 'Booge%' )

4-- To view a list of all the appointments and the status of them use this query

-- (#8) this also shows specific sessions between a user/group a particualr employee the time and the course info the student selected

select \* from TutoringSessions

5--If a student selects that they needa a tutor that excels on exam preperation within in 4000 level classes regarding a certain major specific course

-- this will be the query that is run in the backend

select \*

from Employees

Where SpecialtyValue >= 65 -- Exams

and SpecialtyValue <> 999

and CourseValue = 9 -- up to 4k level classes

and RoleID = 'A0000000-0000-0000-0000-000000000003' -- teacher

6-- To view the basic information that gets loaded in within each emplyee profile run this query

select top 1 \* from Employees

6-- To view the basic information that gets loaded in within each user profile run this query

select top 1 \* from users

7-- Users are allowed to join/create groups to all get tutored aroud the same principal.. To view the groups that are created you can use this query

select \* from users where GroupId = '50000000-0000-0000-0000-000000000001'

8 -- Above

9-- Users are allowed to submit feedback about the tutoring session to further enhance the program.. To view the feed back use this query which

-- can be used in a where clause in tutoring sessions to find out more info

select \* from Feedback

10-- A session is held on one of many campuses and to display a list of all the campuses our system supports use this query

select \* from Campuses

Database Change Log

From our original database design to now we have changed a lot of minor things. We started off with very minimal entities somewhere between 10-11 and with our current design we branched out to 15 to accommodate a more dynamic and versatile tutoring program to allow for expansion. One of our biggest challenges was deciding and implementing an efficient solution to storing scheduling information given to us by students and employees (would probably use some sort of db or api to pull this info. Not actually input by users). We managed to do this in what we think is a very efficient way by creating calendar days and assigning a calendar event to each student class marking it with its specific calendar day. We also adapted to the possibility that multiple campuses will be working on this at the same time, also we made it available to multiple campuses so they could all run the same program. Storing student and employee schedules may seem expensive however when you consider that scripts would not run every day you get more of a sense of speed that its let on to have. These were the main obstacles we had to face when it came to the design of the database where the more smaller tasks were the naming scheme and the dependency steps.

# Demo Video Access Link

<https://drive.google.com/file/d/1LYnvtIwSVLR4CWZefBoRZG51npYX9NVp/view>