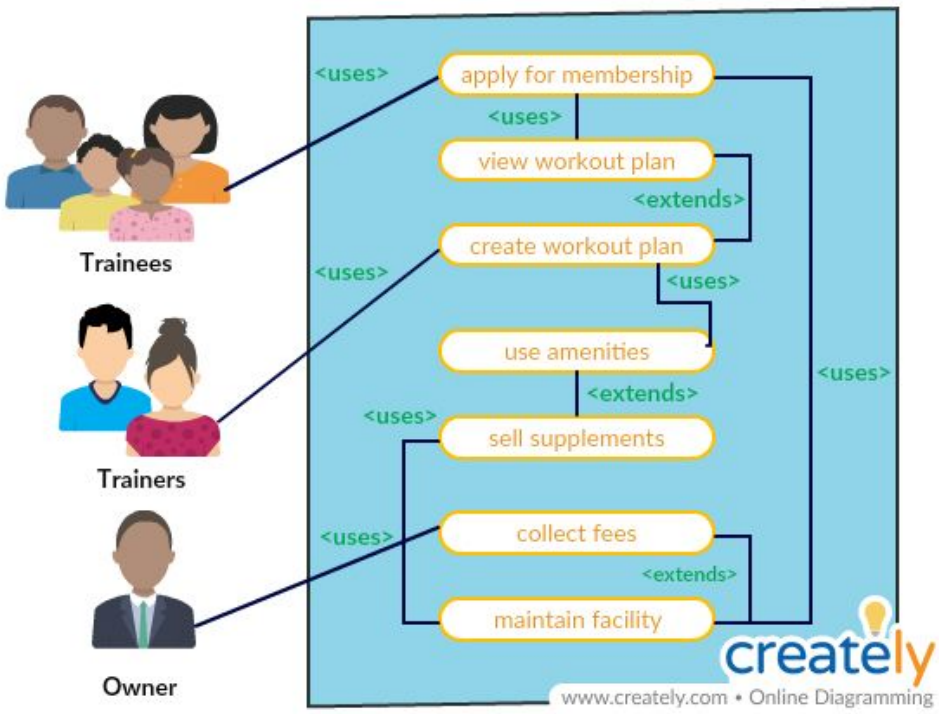
**24-Hour Database**

Created by Neil Soriano and Nathan Huizar

**System Description:**

Our database was designed so users can easily navigate through and eliminate the hassle of finding the right gym. The tables found in our database consist of Trainers, Trainees, Workout Plan, Amenities, Facility, Equipment and Owner. Tuples in Trainer are trainee id, membership id, trainer id, facility id, and age. The Trainee table has weight, age, membership type, trainee id, monthly fee and membership id. The different types of membership types found at this gym range from active to sport to super sport. Monthly fees are lowest for active and highest for super sport. Workout Plan tuples consist of trainee id, membership type, fitness class, trainer id and facility id. The fitness classes in our data are cycling, CrossFit, Zumba, Pilates and yoga. The Amenities table has comfort, sport, personal and facility id. The comfort options found in the 24-Hour DB are massage, spa and sauna and the personal options are daycare or smoothie. The gym also has sports one can play so we included tennis, basketball, swimming and rock climbing as data that can be found in the database. Tuples in Facility are facility id, facility cost, membership type and city. Lastly, the Owner table has facility cost, owner id, facility id, city and name.

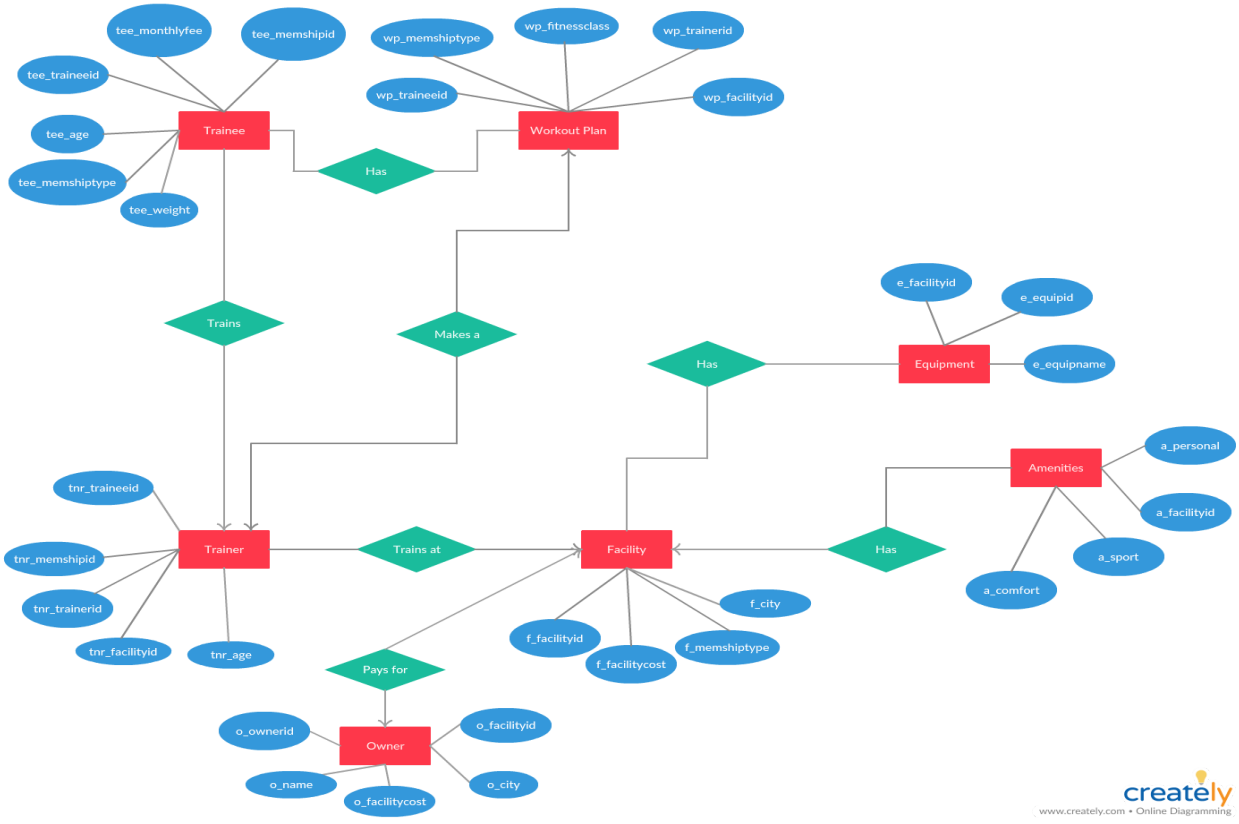
**Use Case Description:**




There are several important reactions that we wanted to display in from our UML Diagram. It was important to us understand what membership types Trainees have since that will narrow down their permission to use certain features of the gym. Active will have the least number of features, sport is in the middle and super sport has the most features. It was also important to know what Trainers have what Trainees. Since a Trainer can have more than one Trainee, we thought it would be important to keep track of who would be paired with who. Knowing what equipment was available at each facility is important as well since not all gyms have the same workout equipment. This may be a deciding factor when people are picking a membership type since super sport is designed to have the most amount of equipment available. Additionally, knowing what amenities and specific workout plans a facility offers alos factors in to a client signing up for a membership. The Trainers create the workout plans for the Trainees so we thought that would be helpful for Trainers to know the workout plan before meeting up with the Trainees. On the more administrative side, owners can see the cost of their respective facilities to see if the maintenance costs are being met.

**E/R Diagram:**

The E/R Diagram we made has seven tables: Trainer, Trainee, Workout Plan, Facility, Amenities, Equipment and Owner. The relationship between a Trainee and Workout Plan is a “has” and a one to one relationship. A Trainee can is assigned to one workout plan and vice versa. The same logic also applies for Trainer and Workout Plan. Trainer and Trainee is a “Trains” and a one to many relationship. A Trainer can have many Trainees, but a Trainee can only have one Trainer. Trainer has a “Trains at” and a one to many relationship with Facility. A facility can have many trainers, but a Trainer can only be assigned to one facility. The Owner and Facility has a “Pays for” and a one to one relationship. The Owner pays for one facility and the costs from the facility go to one owner. Facility and Equipment have a “has” and many to many relationship.The facility can have a lot of equipment and equipment can be present at many facilities. Facility and Amenities have a “has” and one to many relationship. A facility can have many amenities.



**Relational Schema:**

The tuple, membership type, is present in Trainer, Trainee, Workout Plan and Facility. Trainee id can be connected in Trainer, Trainee and Workout Plan. Facility cost is only connected through Owner and Facility. The most common tuple present in almost all the tables is facility id.

**Implementation Details**

We primarily built our project on C++ with sqlite as the database. C++ is a language we were very familiar with and with using sqlite over the semester, we were also very familiar with it. Our program first loads up a menu in which we ask the user what they would like more information about. All of the tables are displayed with the option to exit the program if the user does not need any more information. According to the option that the user chooses, the correct function will be called to display another menu that goes into specific information about the tables. When the user chooses an option here, the correct function is called and a sql statement is called. Our functions that have queries are constructed by first checking to see if the database has been opened(error checking). We then declare a char pointer called sql which will be set equal to the respective sql statement that will be executed. We set an int rc equal to the function sqlite3_exec which will have the char sql as a parameter. This function will execute the sql statement. We have additional if statements for deeper error checking. We then close the database and ask the user if they would like to return to the menu. Y will take you back to the title screen and N will stop the program. We took our program a step further in displaying information because we wanted to have a separate menu for owners of each facility. Owners would be able to check the monthly cost of the facility and monthly costs of the trainees by first “logging in” with their respective owner_id. Owners are the ones with the flexibility to update/delete/insert information into any of the tables. An array is created with all owner ids which is then iterated through to check if the owner id entered is indeed an existing owner id. When finished, we again ask the user if they would like to proceed further.