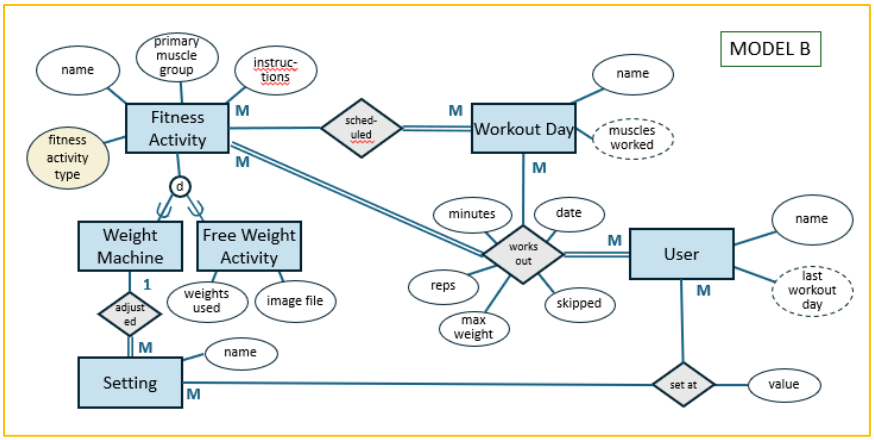
Fitness Application Project
Database Programming - MySQL

What you need to know for this assignment

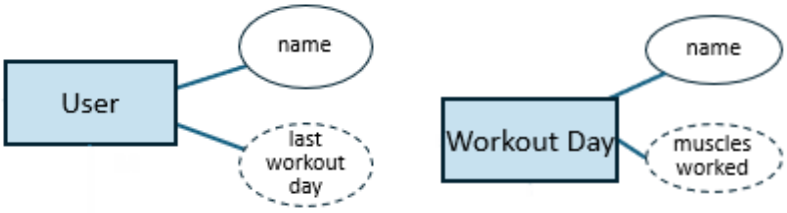
- How to create stored procedures and functions
- How to access a database using PHP PDO
- How to write aggregate queries
- How to write subqueries

This was your EER diagram, from which you built the physical data model:



Instructions:

1. **Database Functions:** One of the most common use of functions is to create a program to calculate derived fields. Note these derived fields in the Fitness Data Model:



   a. Create a function for **last_workout_day**. It should examine the workout records and return the most recent workout day for a given user as an integer (e.g., 1 for Day One, 2 for Day Two, etc.). You will want to pass the function the primary key of the user table. If a user has never worked out, return a 0.

   b. Create a function for **muscles_worked**. This will be a bit more complicated. Take Day 2 for instance.

workout_day	muscles_worked	number_of_activities	activities
Day Two	back/biceps	9	Assisted Pull Ups, Concentration Curls, Dead Lifts, Elliptical, EZ Bar Curls, Hammer Curls, Seated Cable Row, Seated Shoulder Press, Treadmill

   Split training days (which we are using for this application) are typically "back/biceps" or "legs/core" or "chest/triceps". Thus we want to ignore the primary muscle for the Seated Should Press (shoulders) as well as the Treadmill that works out multiple muscles. So this function will have to examine the nine records for Day Two (using the integer input for the primary key, i.e., 2) and return the string either "back/biceps" or "biceps/back". Use your SQL skills to retrieve that string. (No points awarded if you do not use SQL aggregation to generate that string.)

   c. Now that you've created a function for last_workout_day, create one last function for **next_workout_day**. If a user has not worked out at all, that day should be Day 1. If a user's last workout day was Day 6, their next one should be Day 1 to repeat the cycle. Like the last_workout_day function, it should take in the primary key of user, and return the integer corresponding to the next workout day.

2. MySQL Procedures

a. **Build Workout Records for a User.** Write a MySQL procedure to generate a workout for a user. Consider user "venkat" who might be user_id #3. The last day he worked out was August 9th, 2024. After a long absence, on October 3rd, he was able to resume workouts right where he left off. The procedure should take advantage of your **next_workout_day** function and figure out which workout day it is (in this case, Day Six).

Syntax of the Call Procedure command: `CALL build_workout(3, next_workout_day(3));`
where 3 is Venkat's id.

Note that the Day Six fitness activities were added into the workout table – along with null values for the properties Venkat will have to add when he does his workout, e.g., the number of minutes on the treadmill, or the amount of weight used in the Seated Leg Curl.

name	workout_day	name	date
Venkat	Day Five	Treadmill	8/9/2024
Venkat	Day Five	Elliptical	8/9/2024
Venkat	Day Five	Dead Lifts	8/9/2024
Venkat	Day Five	Assisted Pull Ups	8/9/2024
Venkat	Day Five	Seated Cable Row	8/9/2024
Venkat	Day Five	EZ Bar Curls	8/9/2024
Venkat	Day Five	EZ Preacher Curls	8/9/2024
Venkat	Day Five	Zottman Curls	8/9/2024
Venkat	Day Six	Treadmill	10/3/2024
Venkat	Day Six	Elliptical	10/3/2024
Venkat	Day Six	Standing Calf Raise	10/3/2024
Venkat	Day Six	Seated Leg Press	10/3/2024
Venkat	Day Six	Prone Lying Leg Curl	10/3/2024
Venkat	Day Six	Seated Leg Curl	10/3/2024
Venkat	Day Six	Torso Rotation	10/3/2024
Venkat	Day Six	Crunches	10/3/2024
Venkat	Day Six	Planks	10/3/2024

b. Fetch the current workout for a user. Now write a procedure that will fetch the current workout for a user. It should at least the name of the fitness activity, the date of the workout, and all columns that will hold statistics for the activity (e.g., maximum weight lifted, number of minutes, number of reps).

Often, the current workout will be on today's date (assuming it was built today with procedure `build_workout`) but you cannot assume that. A user might have built the current workout the night before, and is trying to fetch it the next morning. How will you query the correct workout records?

Here is the way to call that procedure for user 3, Venkat: `CALL fetch_current_workout(3);`

Please submit:

1. The code for the functions and procedures