ForFit

Never Quit Again

Illya Starikov, Jason Young, Claire Trebing

April 27, 2016

1 The Database Design Manual

1.1 Revised Problem Statement

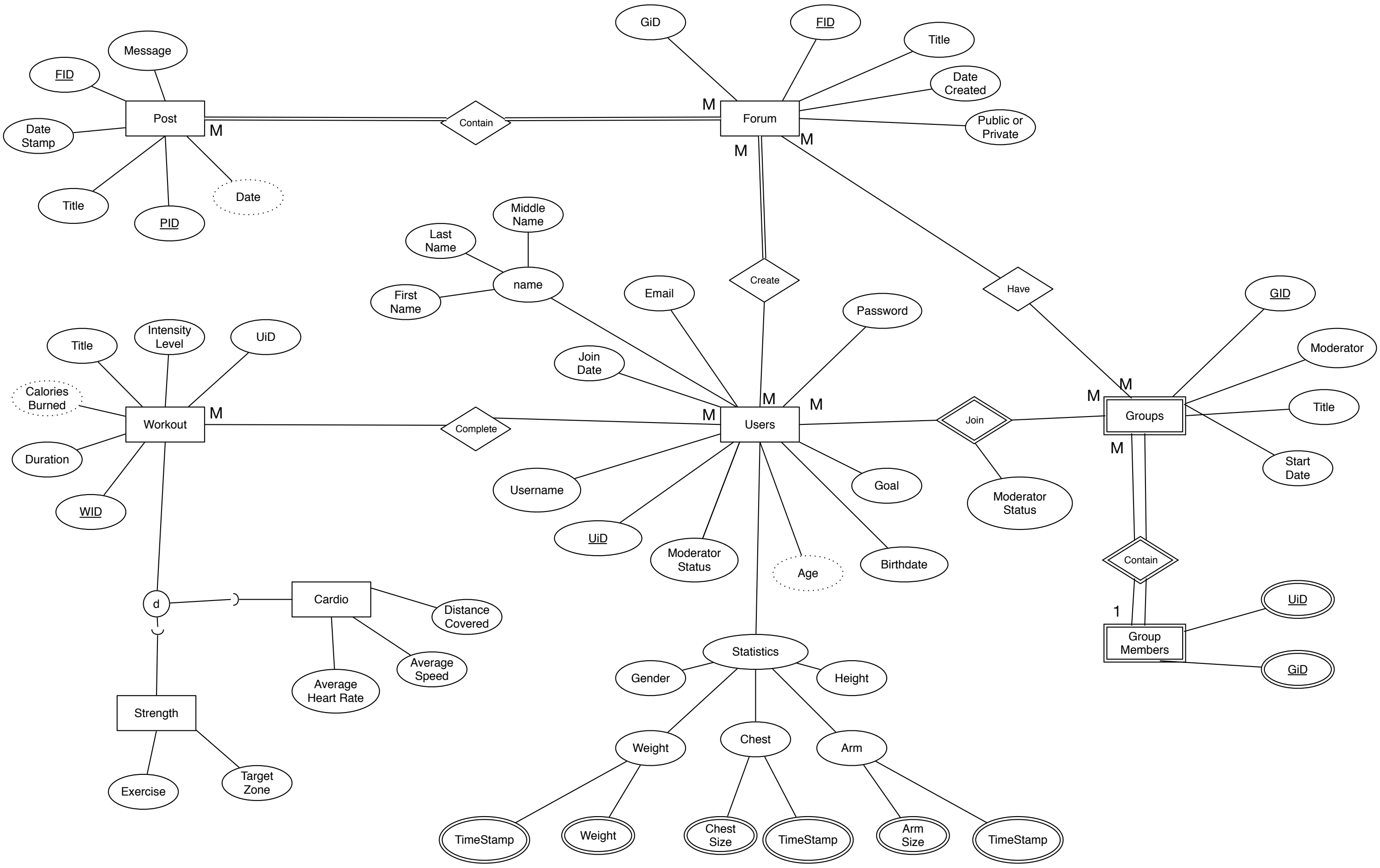
2015 marked a record year for Americans regularly exercising, hitting just over 55%. Keeping that momentum is difficult. As blue-collar jobs continue to decline, it has been more important than ever to keep a weekly regimen of healthy eating and exercising.

Living in the most interconnected generation poses quite a viable idea for social networking: healthy living. We can build a social network that connects users to friends, peers and family to make an online community of healthy living.

Our database will be essential because it will unite something so mundane and uninteresting with familiar faces. This will make exercise more enjoyable and offer a group to hold you accountable. We can shape an entire generation by having motivation a click away.

This database will consist of premade workouts, groups, and reminder emails to help you stay on top of your fitness plan.

1.2 Conceptual Database Design



Message
FID
Post
Date Stamp
Title
PID
Date
M
Contain
GiD
FID
Title
Date Created
Forum
Public or Private
M
M
M
Create
Have
Last Name
Middle Name
First Name
name
Email
Password
Join Date
Users
M
M
M
Title
Intensity Level
UiD
Calories Burned
Workout
Duration
WID
M
Complete
Join
Moderator Status
Groups
GID
Moderator
Title
Start Date
M
M
M
Username
UiD
Moderator Status
Age
Goal
Birthdate
d
Cardio
Distance Covered
Average Speed
Average Heart Rate
Strength
Exercise
Target Zone
Contain
1
Group Members
UiD
GiD
Statistics
Gender
Height
Weight
Chest
Arm
TimeStamp
Weight
Chest Size
TimeStamp
Arm Size
TimeStamp

1.3 Logical Database Design

1.3.1 Relational Set

Please note primary keys are signified by $^{\textbf{PK}}$ and foreign keys are signified by ^{FK}.

Post

PiD $^{\textbf{PK}}$	FiD ^{FK}	Message	DateStamp	Title	Date

Forum

FiD $^{\textbf{PK}}$	Title	DateCreated	PublicOrPrivate	GiD ^{FK}

Groups

GiD $^{\textbf{PK}}$	Moderator ^{FK}	Title	StartDate

Workouts

WiD $^{\textbf{PK}}$	Duration	Title	IntensityLevel	CaloriesBurned	UiD ^{FK}

Strength

WiD $^{\textbf{PK}}$	Duration	Title	IntensityLevel	CaloriesBurned	UiD ^{FK}	Exercise	Target Zone

Cardio

WiD $^{\textbf{PK}}$	Duration	Title	IntensityLevel	CaloriesBurned	UiD ^{FK}	AverageHeartRate	AverageSpeed	DistanceCovered

Users

UiD $^{\textbf{PK}}$	Username	Height	Birthdate	Goal	Password	JoinDate	Gender	FirstName	Middle

Weight

UiD $^{\textbf{PK, FK}}$	TimeStamp $^{\textbf{PK}}$	Weight

Arm Size

UiD $^{\textbf{PK, FK}}$	TimeStamp $^{\textbf{PK}}$	Arm Size

Chest Size

UiD $^{\textbf{PK, FK}}$	TimeStamp $^{\textbf{PK}}$	ChestSize

Group Members

UiD $^{\textbf{PK}}$	GiD ^{FK}	ModeratorStatus

1.4 Application Program Design

1.4.1 Create a New User

This function creates a new user, accessing only the `user` table.

Input

`Username, Height, Birthdate, Goal, Password, Gender`.

Steps

1. Check to see if the `username` is available. If available, proceed. If not, display appropriate message to notify the user.
2. Insert appropriate information to the User table. (Username to `username`, height to `height`)
3. Generate a user ID, assign it to the `UiD` attribute.
4. Take a time stamp, assign it to the `JoinDate` attribute.
5. Calculate the age based on the `BirthDate` attribute.

Output

A new `User` entity will be inserted into the table, with appropriate data into proper columns (along with computed properties and derived properties).

Assumptions

`Username, Height, Birthdate, Goal, Password, Gender` are all correct (this will be validated in the sign up form).

1.4.2 Delete A Group

This function deletes a group by updating information in `Forum`, and removing the `Group` and `Group Members` tables.

INPUT

The Group ID (`GiD`) that is to be deleted.

STEPS

1. Check to see if the request is made by the moderator via the `Moderator` column in the `Groups` table. If true, approve the request. If not, cancel the request and notify the user.
2. Remove the row that has a matching `GiD` that was provided for deletion in the `Groups` table.
3. Query the `Group` Members table, removing any row that match the GiD provided for deletion.
4. Query the `Forum` table for any matching Group Ids (`GiD`), setting the `GiD` to `null` if matching.

OUTPUT

The groups are deleted, the group members within that group are deleted, and any reference to the group is deallocated.

ASSUMPTIONS

None.

1.4.3 Modifying User Statistics

Our user statistics have the ability to fluctuate. We would like to accommodate for this fluctuation by allowing users to update their respective statics; specifically, we would like to let users update `Username, Height, Goal, Password, Gender` in the `Users` table.

Input

The specific attribute(s) of the set `Username, Height, Goal, Password, Gender` that would like to be updated with the new value.

Steps

1. Ensure the data is valid (e.g. is not `null` when applicable, in the proper domain). If it is valid, continue. If not, prompt the user with an error message and try again.
2. Modify the attribute to reflect the new value.
3. Repeat for any additional attributes provided.

Output

The attribute(s) should now reflect the new value provided.

Assumptions

The data is within a proper range (will not overflow).

1.4.4 Query Other Users

This function allows for users to query other users; this can be done via `Username` or `FirstName, MiddleName`, and `LastName` from the `Users` table.

Input

Either a `Username` xor any subset of `FirstName, MiddleName`, or `LastName`.

Steps

1. Check to see if input is valid. If is, proceed. If not, display error message to the user.
2. Query the `Users` table to see if the user exists. If the user exists, proceed. If not, display appropriate message to the user.
3. Project the profile.

Output

Either the search user will be projected or an error message is the user does not exist.

Assumptions

The first, middle and last name are all provided. The names are unique (solely for the testing purposes).

1.4.5 User Leaderboards

Generate the leaderboard based on the workouts accomplished; specifically aggregating data from the `Strength` and `Cardio` table. *Note this is the function that requires multiple tables.*

Input | None.

Steps |

1. Merge the `User` and `Workouts` table, call the new table `Merged`.
2. Add up the total duration (call the new property `TotalDuration`) in the `Merged` table based on the `UiD` attribute, making a new table named `Sums`.
3. Sort the `Sums` by the `TotalDuration` attribute.
4. Display the top 10 on the sorted `Sums` table to the user.
5. Display the user their current rank.

Output | An eleven-row table displaying the top 10 leaderboards and the users current rank.

Assumptions | There is a bare minimum of eleven users.