ForFit

Never Quit Again

Illya Starikov, Jason Young, Claire Trebing

May 2, 2016

Part I.

The Database Design Manual

1. Problem Statement

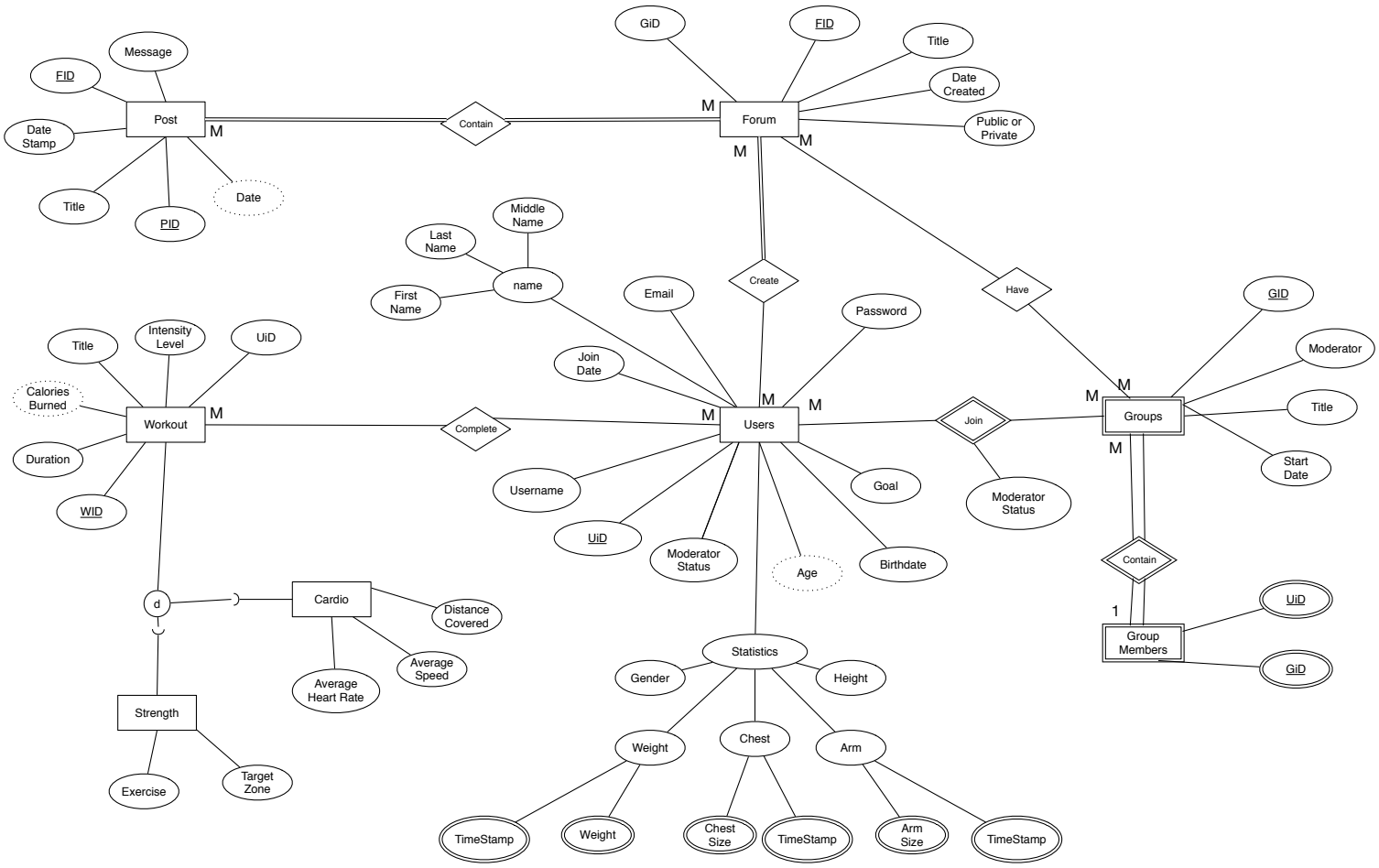
2015 marked a record year for Americans regularly exercising, hitting just over 55%. Keeping that momentum is difficult. As blue-collar jobs continue to decline, it has been more important than ever to keep a weekly regimen of healthy eating and exercising.

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.

2. Conceptual Database Design


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

3. Logical Database Design

3.0.1. Relational Set

Please note primary keys are signified by **PK** and foreign keys are signified by *FK*.

Post

PiD **PK**	FiD *FK*	Message	DateStamp	Title	Date

Forum

FiD **PK**	Title	DateCreated	PublicOrPrivate	GiD *FK*

Groups

GiD **PK**	Moderator *FK*	Title	StartDate

Workouts

WiD **PK**	Duration	Title	IntensityLevel	CaloriesBurned	UiD *FK*

Strength

WiD **PK**	Duration	Title	IntensityLevel	CaloriesBurned	UiD *FK*	Target Zone

Cardio

WiD **PK**	Duration	Title	IntensityLevel	CaloriesBurned	UiD *FK*	AverageHeartRate
AverageSpeed	DistanceCovered					

Users

UiD **PK**	Username	Height	Birthdate	Goal	Password	JoinDate	Gender
FirstName	MiddleName	LastName					

Weight

UiD ***PK, FK***	TimeStamp **PK**	Weight

Arm Size

UiD ***PK, FK***	TimeStamp **PK**	Arm Size

Chest Size

UiD ***PK, FK***	TimeStamp **PK**	ChestSize

Group Members

UiD **PK**	GiD *FK*	ModeratorStatus

3.0.2. Summary Table

Attribute	Data Type	Constraints	Meaning
Message	varchar	500 characters	Contents of a post.
FID	int	unique	Forum ID.
DateStamp	timestamp	timestamp of creation	Timestamp of post was creation.
Title	char	35 characters	Title of a post.
PID	int	unique	Post ID.
Date	timestamp	date of creation	Timestamp of post creation.
Title	char	35 characters	Title of a forum.
DateCreated	timestamp	timestamp of creation	Timestamp of forum creation.
PublicOrPrivate	boolean	-	Is forum public.
GID	int	unique	Group ID.
Moderator	char	must match a username	Group owner/moderator.
Title	char	35 characters	Title of the group.
StartDate	timestamp	timestamp of creation	Timestamp of group creation.
IntensityLevel	int	-	Enumerated value, code for different options.
Title	char	35 characters	Title of the workout.
CaloriesBurned	int	num > 0	How many calories burned during exercise.
Duration	timestamp	num > 0	Length of exercise.
WID	int	unique	Workout ID.
Exercise	char	35 characters	Exercise name.
TargetZone	char	35 characters	Area exercise is intended to workout.
AverageHeartRate	int	num > 0	Average heart rate recorded during exercise.
AverageSpeed	int	num > 0	Average speed of cardio exercise.
DistanceCovered	int	num > 0	Distance covered during cardio exercise.
Password	char	-	Password of user.
JoinDate	timestamp	timestamp of creation	Timestamp of users sign up.
Username	char	unique	Users username, used for sign in.
ModeratorStatus	boolean	-	Is a moderator.
Birthdate	date	-	Date of users birth.
Goal	varchar	500 characters	Users intended workout goals.
Gender	char	1 character, M or F	Users gender, male(M) or female(F).
Weight	int	num > 0	Users weight at a certain date.
ChestSize	int	num > 0	Users chest size at a certain date.
ArmSize	int	num > 0	Users arm size at a certain date.
Height	int	num > 0	Users height at a certain date.
UID	int	unique	Users ID.
TimeStamp	date	unique	Date of users weight mesurement.
TimeStamp	date	unique	Date of users chest measurement.
TimeStamp	date	unique	Date of users arm measurement

4. Application Program Design

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).

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.

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).

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).

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.

Part II.

User Manual

5. Functions

5.1. Delete Group

Summary of delete group.

5.1.1. Parameters

groupID The `GiD` of the group that is to be deleted.

5.1.2. Results

The group who's ID matches the given `groupID` will be deleted. If the group does not exist, nothing is displayed.

```
rMBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select delete_group(42);
 delete_group
--------------
(0 rows)

postgres=#
```

5.2. Modify User

Summary of what modify user does.

5.2.1. Parameters

userID The `UiD` of the user to who's statistics that will be modified. If this is not provided, there is no way of changing the user.

newUsername The new username property of the user. If this statistic will not change, pass a `'0'` to the function. Otherwise, proceed with changing.

newHeightFeet See above.

newHeightInches See above.

newGoal See above.

newPassword See above.

newGender See above.

5.2.2. Results

For all attributes that do not have the value of `'0'`, the value of said attribute will be updated (provided it upholds all constraint).

```
rMBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select modify_user(1, 'IllyaStarikov', 0, 0, 42, '0', 'F');
 modify_user
-------------
(0 rows)

postgres=# select * from USERS where UiD = 1;
 uid |   username    |          email           | heightfeet | heightinches | birthdate  | goal |   password    |  joindate  | gender | fname | minit |  lname
-----+---------------+--------------------------+------------+--------------+------------+------+---------------+------------+--------+-------+-------+----------
   1 | IllyaStarikov | CasieScarlett1@gmail.com |          6 |            2 | 1967-03-31 |   42 | EtZEdlnfNiBAI | 1999-12-31 | F      | Casie | M     | Scarlett
(1 row)
```

5.3. New User

Summary of what modify user does.

5.3.1. Parameters

id, username, email, heightFeet, heightInches, birthdate, goal, password All the attributes that can be inserted into user table.

joinDate, gender, firstName, mInitial, lastName See above.

5.3.2. Results

Provided all the attributes meet database constraints, a new user will be inserted into the database. If a constraint is not meant, there will be a message outputting the error and the insert will be rejected.

```
postgres=# select new_user(404, 'IllyaStarikov', 'IllyaStarikov@iCloud.com', 5, 10, '07-20-1994', 165, '42', '07-21-1994', 'M', 'Illya', 'A', 'Starikov');
 new_user
----------
(0 rows)

postgres=# select * from USERS where UiD = 404;
 uid |   username    |          email           | heightfeet | heightinches | birthdate  | goal | password |  joindate  | gender | fname | minit |  lname
-----+---------------+--------------------------+------------+--------------+------------+------+----------+------------+--------+-------+-------+----------
 404 | IllyaStarikov | IllyaStarikov@iCloud.com |          5 |           10 | 1994-07-20 |  165 | 42       | 1994-07-21 | M      | Illya | A     | Starikov
(1 row)
```

5.4. Query User

Summary of what query user does.

5.4.1. Parameters

firstName, lastName The first and last name of the user to be queried.

5.4.2. Results

If the user exists, the column of the user will be returned. If the user does not exist, not such column will be returned.

```
rMBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select query_user('Illya', 'Starikov');
                                                                                                  query_user
-------------------------------------------------------------------------------------------------------------------------------------------
 (117,IllyaStarikov,IllyaStarikov@iCloud.com,5,10,1994-07-20,165,42,1994-07-21,M,Illya,A,Starikov)
(1 row)
```

5.5. Total Users

Summary of what total users does.

5.5.1. Parameters

None.

5.5.2. Results

Returns the total count of users, a maintenance function used to compare to the total count in the database.

```
rMBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select total_users();
 total_users
-------------
          52
(1 row)
```

5.6. User Leaderboards

Summary of what user leaderboards does.

5.6.1. Parameters

None.

5.6.2. Results

Returns the top 10 users (calculated via count of workouts done) in the database.

```
rMBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select user_leaderboards();
    user_leaderboards
-----------------------
 (Gregory,Patty,6)
 (Agueda,Heather,4)
 (Bertram,Gerardoy,4)
 (Gay,Arthury,4)
 (Garry,Shanony,4)
 (Brad,Lana,4)
 (Francisco,Bruce,4)
 (Edgar,Dennisy,4)
 (Alvina,Carter,4)
 (Darlene,Salvatore,3)
(10 rows)
```