Database Documentation

The following document has been created keeping in mind the next developer that would be accessing the database.

Contents:

1. Introduction
2. Schema

1.1. EER-Diagram

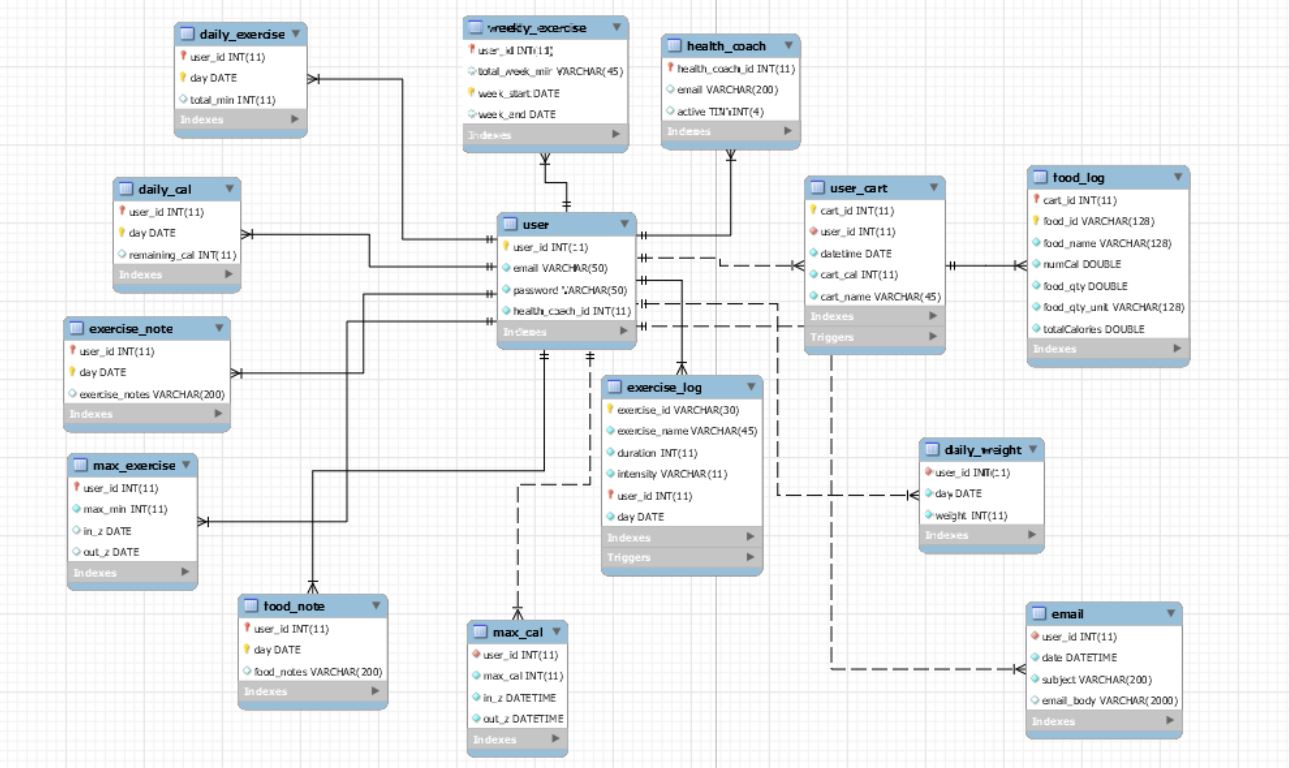
1.2. Description of the entity sets

1. Dataflow
2. Queries
3. Stored Procedures
4. Future work
5. INTRODUCTION

The database has been created to provide an efficient storage system for the Healthy Living app. It has been implemented using MySQL.

**Note:** MySQL server is running version 5.1, so if you are using Workbench to connect, you need to download an older version (Workbench 6)

2. SCHEMA

EER-Diagram:

Description of the entity sets:

Note: Front-end of the app refers to what the user sets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Information created by/gathered from** | **Table Function** | **Primary Key** | **Foreign Key** |
| user | Health coach | * Stores information about each user. * Assigns them a user\_id, and keeps their password and username. Also assigns each user a health\_coach\_id. * The user\_id from this table has been used consistently to match the user with their details in almost all the other tables. | user\_id | Health\_coach\_ id |
| user\_cart | The front-end of the app | * (On the front end) When the user checks out the cart, it creates a cart\_id for that account. * Each checked out cart stores the following information:’ * Date and time on which it was created * Total calories for that particular cart * Food diary for user notes about the cart that is being added\* * Cart\_name (optional) such as breakfast, lunch, dinner. * All the individual food from the cart is stored in the food\_log table, referencing the cart\_id as a foreign key. | cart\_id | user\_id |
| food\_log | Nutrition IX API + Front-end | * Each item in the checked out cart is stored as a record in the food\_log table with a reference to the cart\_id that it is a part of. * food\_id, food\_name and food\_cal is extracted from Nutrition IX API. * Food\_qty, food\_qty\_unit is extracted from the front-end of the app.   NOTE: The primary key for this table is a combined key to avoid unnecessary multiple record creation such as two records for two bananas from the same cart. | (combined) cart\_id + food\_id | cart\_id |
| exercise\_log | Nutrition IX API + Front-end | * Every time a user enters (on the front-end) an exercise performed, a record is created and stored here. * Exercise\_id and exercise\_name are extracted from NutritionIX API * Duration, intensity and day are extracted from the front-end. | None \* | user\_id |
| max\_cal | Health coach | * The user’s max intake for the day is set by the health coach while creating the user accounts. * This table also comes with an in\_z and out\_z function to keep track of all the past max\_cal values for the user (for research purposes) * In\_z keeps track of when the value was set. * Out\_z for the current value shows a date of 12/31/2099, and for past values shows the date on which it was changed. | None | user\_id |
| daily\_cal | Stored procedure calculation | * This table uses the cart\_cal and the max\_cal from user\_cart and max\_cal to keep track of the remaining calories that can be consumed by the user for that day. \* | (combined) user\_id + day | user\_id |
| daily\_weight | Front-end | * The user can enter their daily weight to keep track of their progress and view past records. | None | user\_id |
| food\_note | Front-end | * The user can add notes each day about what they thought of their food intake.   NOTE: Primary key is a combined key here so the user can enter only one note per day about their intake. | (combined) user\_id + day | user\_id |
| exercise\_note | Front-end | * The user can add notes each day about their exercise. * This can also be used as the weight notes for the day.   NOTE: Primary key is a combined key here so the user can enter only one note per day about their exercise. | (combined) user\_id + day | user\_id |
| health\_coach | health\_coach | * Information is stored about each health\_coach (currently only email) that is a part of the program. * The active attribute can provide the functionality to keep track of whether a health\_coach is currently part of the system or not. | health\_coach\_id | None |
| email | Front-end of the app/ health coach | * Keeps track of the emails sent back and forth between the user and the health coach. | None | user\_id |
| max\_exercise | health\_coach | * The user’s max exercise duration for the **WEEK** is set by the health coach while creating the user accounts. * This table also comes with an in\_z and out\_z function to keep track of all the past max\_cal values for the user (for research purposes) * In\_z keeps track of when the value was set. * Out\_z for the current value shows a date of 12/31/2099, and for past values shows the date on which it was changed. | (combined) user\_id + in\_z | user\_id |
| daily\_exercise\* | Front-end | * Keeps track of daily exercise duration done by each user | (combined) user\_id + day | user\_id |
| weekly\_exercise | Front-end | * Keeps track of weekly exercise duration done by each user. | (combined) user\_id + week\_start | user\_id |

3. DATAFLOW

The dataflow has been described (along with the query being used) for the functions that use join queries, or that access multiple tables at once. The rest of them are pretty straightforward.

1. Add to cart

* First a record is created in the user\_cart table and then using the created cart\_id, the food in the cart is moved to the food\_log table.

|  |
| --- |
| -- /meallog POST a new user cart, where cart\_cal is summed before adding individual food items INSERT INTO user\_cart (user\_id, datetime, cart\_cal, cart\_name)  VALUES (1, NOW(), 200, 'Breakfast');  -- /meallog POST food to cart (for each food to be added to the cart) INSERT INTO food\_log VALUES (12, 10, "Apple", 100, 1, 'serving'); |
|  |
|  |

1. Get all carts

* All the cart and the food in them is pulled using a join query between the user\_cart and the food\_log table.

|  |
| --- |
| -- /meallog GET all carts, as well as the food in each cart, for a specific user on the current day SELECT user\_cart.cart\_id, cart\_name, food\_id, food\_name, food\_cal, food\_qty, food\_qty\_unit  FROM user\_cart, food\_log  WHERE user\_cart.cart\_id = food\_log.cart\_id AND user\_cart.user\_id = 1 AND DATE(user\_cart.datetime) = CURDATE(); |
|  |

1. Calculate remaining calories

* This is done using a stored procedure that uses the cart\_cal from the user\_cart and the max\_cal for that user. Every time a record is created in the user\_cart table, that cart\_cal is used to update the remaining\_cal in the daily\_cal table.

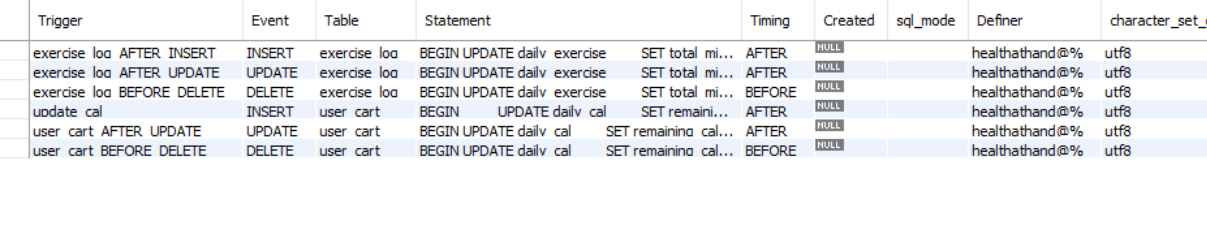
|  |
| --- |
| DELIMITER //  CREATE EVENT IF NOT EXISTS test\_remaining\_calories  ON SCHEDULE EVERY 1 MINUTE STARTS '2018-11-8 00:00:00'  ON COMPLETION PRESERVE ENABLE  DO  UPDATE daily\_cal, max\_cal  SET daily\_cal.remaining\_cal = max\_cal.max\_cal  WHERE daily\_cal.user\_id = max\_cal.user\_id; //  DELIMITER ; |

4. QUERIES

|  |
| --- |
| -- /mealnotes POST update notes for a specific user on the current day UPDATE daily\_cal SET notes = "" WHERE user\_id = 1 AND DATE(datetime) = CURDATE();   -- /exernotes POST update notes for a specific user on the current day UPDATE daily\_weight SET notes = "" WHERE user\_id = 1 AND DATE(datetime) = CURDATE();   -- /report POST weight for a specific user INSERT INTO daily\_weight (user\_id, day, weight) VALUES (1, CURDATE(), 200);   -- /meallog POST a new user cart, where cart\_cal is summed before adding individual food items INSERT INTO user\_cart (user\_id, datetime, cart\_cal, cart\_name) VALUES (1, NOW(), 200, 'Breakfast');   -- /meallog POST food to cart (for each food to be added to the cart) INSERT INTO food\_log VALUES (12, 10, "Apple", 100, 1, 'serving');   -- /exerlog POST exercise info for a user on a specific day INSERT INTO exercise\_log VALUES (5, 'Jumping Jacks', 60, 'light', 1, CURDATE()); |

5. STORED PROCEDURES

List of triggers: (SHOW triggers;)



Event scheduler:

|  |
| --- |
| DELIMITER //  CREATE EVENT IF NOT EXISTS test\_remaining\_calories  ON SCHEDULE EVERY 1 DAY STARTS '2018-12-0 00:00:00'  ON COMPLETION PRESERVE ENABLE  DO  UPDATE daily\_cal, max\_cal  SET daily\_cal.remaining\_cal = max\_cal.max\_cal  WHERE daily\_cal.user\_id = max\_cal.user\_id; //  DELIMITER ; |

6. FUTURE WORK

Food Diary

- The food\_diary attribute from the user\_cart table can be moved to the food\_note table.

- To move it, the food\_note table needs an additional foreign key attribute, cart\_id, for reference.

- After moving it, food\_diary can be deleted from the user\_cart table.

Exercise Log

- Create primary key, if needed by combining the exercise\_id and day.

- Add calories burnt.

Event Scheduling

- The code is working, but scheduling is not enabled on the server. To enable, root access to the server is needed.

-Change working directory to /etc

-‘[sudo] nano my.cnf’

-Add line “GLOBAL event\_scheduler=ON”

Daily\_exercise

Also useful for future daily exercise analytics functionality implementation.