Transport
Vehicle
co2 10km
price 10km

All tables here are constants used by the database to do calculations

The values in these tables are never changed, neither by the server nor by database triggers.

The database only reads from these values to calculate co2/price saved

Food
item
co2
kg

LocProd
name
co2 ratio
co2
price

Constants
item
unit
price
co2

Temp
name
kwh saved hour

SolPan
name
kwh day

To create a new user, server inputs:

(**username**, **password**, bdate, email, nationality, firstname, lastname)

Username, password are the only required fields, the rest are optional

User		
userid	PK	AI
date created	Curdate	
username	NN	UNI
password		NN
bdate		
email		
nationality		
firstname		
lastname		
has solarP		
date added solarP		
friend_count		

Values in blue are generated automatically by the database upon creating a new user, and these values will never change.

All the triggers and functions in the database use userid to identify the table

Values in red are entirely server input when creating a new user. These values do not affect any database triggers, meaning whatever the server puts in these rows here will not change anything else in the database

Values in green change according to input in the 'functional' tables (functional tables are "friend_add" and "insert_history")

Green values are determined only by the database functions/triggers, they are never directly changed by the server

Triggers for records update for each user automatically iff the insert_history input follow all the rules as below. The table itself will accept any insert as long as (userid, act1) are NN, but the triggers will not activate unless the input requirements are met.

Purple letters means trigger will look in the constants tables to see if the input is one of the values in there. Red letters means the trigger will activate only if that exact string is input as act1. X means any value can be inserted and trigger will still activate, but the x values are ignored

	userid	Act1	Pric1	Alt1	Pric2	Hours	Km	
Food	NN	Food	NN	Food	NN	x	x	Trigger updates saved (co2/price) food in rec.
Trans.	NN	Vec.	x	Vec.	x	x	NN	Trigger updates saved (co2/price) transp. in rec.
LocProd	NN	"LocProd"	NN	x	x	x	x	Trigger updates saved co2 localproduce in rec
Temp.	NN	"HeatOff"	x	x	x	NN	x	Trigger updates saved (co2/price) temp. in rec.
SolPan	NN	"SolPan"	x	x	x	x	x	.Trigger saves date_added_solpan. in user table

Universal triggers (will activate for user when **any** insert is made into insert_history with their userid):

- kwh_prod_solapanels update calculation based on the current date and the date the user added the solarpanels (from user table)

- (price & co2)_saved_solarpanels update based on function of kwh_prod_solarpanels and constant for co2/price per kwh

- (co2 & price)_saved_total update as the sum of all sources of price/co2 saved in records

- Achievements will be updated based on records table for that user

The insert_history table also has an insert_id which is auto-incremented integer for every insert made, and a date_of_insert which is also automatically generated by the database. These have no functionality at the moment, but they may be useful if we decide to add more features later.

Database contains no in-built foreign keys whatsoever, to prevent errors from inserts. All records are automatically created/updated by triggers. If server inserts wrong input, database will not reject the input (no errors), but the triggers will simply not activate (and will not change anything)

Current issues:

- Solar panels can be inserted multiple times. Each time 'solarpanels' is inserted with a user, it will update the user table date_added_solarpanels to the date of the latest 'solarpanels' insert, and (price/co2/kwh)_solarpanels is reset to 0, and the total co2/price values will also change accordingly (this issue can be fixed in the database)

- All triggers are stateless and will only update upon insert. Unless there is a new insert, none of the triggers will update (solarpanel production count will lag behind if an insert is not made)

Proposed solution: ('login', userid) into insert_history upon user login will activate universal triggers

Functional table: friend_add, insert only (userid1, userid2) in any order, they will be registered as friends in the friends list

Any insert into friend_add will increment both user's friend_count (user table) by one

userid1	userid2	add_id	add_date

After insert into friend_add, both users will have the other user added to their friends list

Database knows which friend[number] to insert into based on each user's friend count on the user table.

src_user	friend1_id	...	friend25_id

Friend numbers are independent for each user

e.g. User 1 has friend count 7. User 2 has friend count 11. Insert into friend_add (1, 2)

User 1's friend8_id will become 2, since he already has 7 other friends

User 2's friend12_id will become 1, since he already has 11 friends

User can add up to 25 friends. If you already have 25 friends and an insert is made into friend_ad with your userid, it will not cause any errors, but the database will not register your new friend into the friends list.

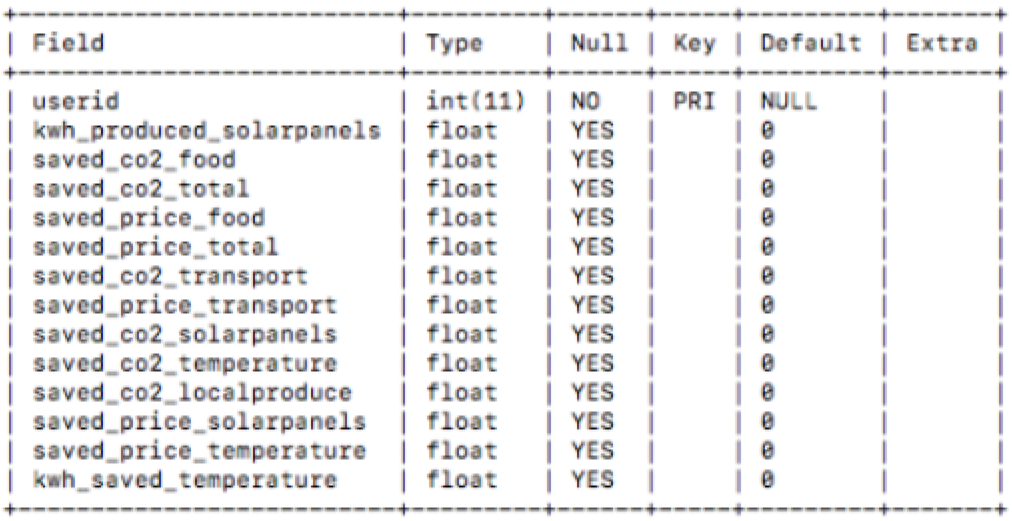
Issues with the friend system:

- You can add yourself as a friend, and you can add the same friend as many times as you want (and it will register them into the friend_list multiple times). You **cannot** make trigger force the database to reject an insert without throwing a mysql exception, and if we start implementing those into the database the server side is going to start messing up a lot. You can set separate triggers to undo the effects of an insert under certain conditions, but those types of triggers are very prone to bugs. Please implement these restrictions on server/client side.

Final words on triggers:

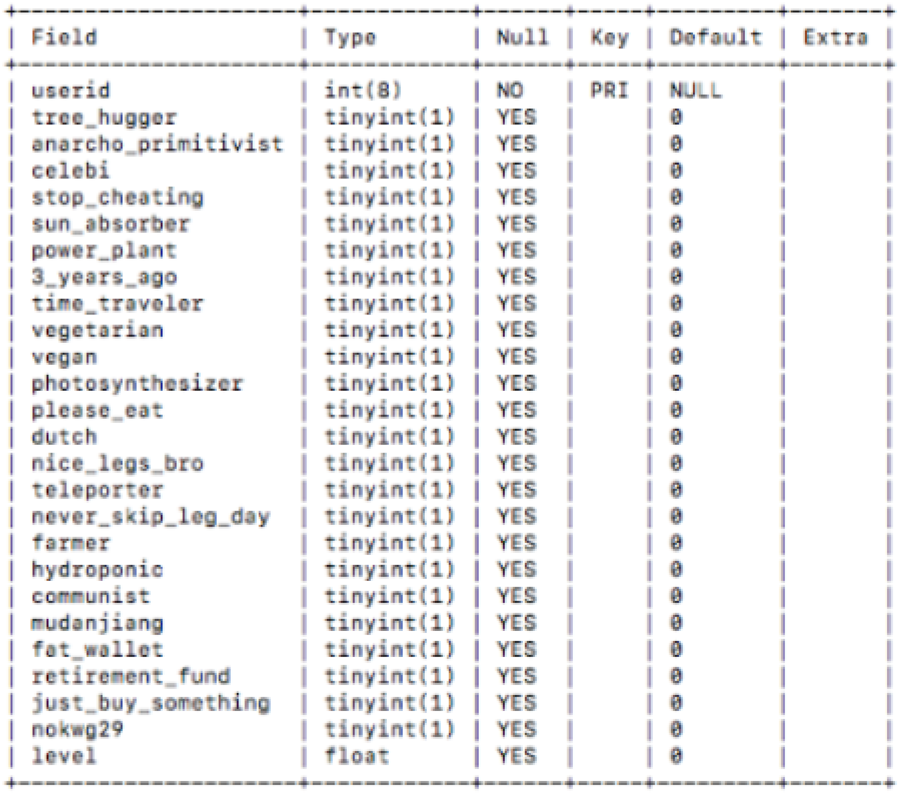
- Triggers can be made to alter/insert/delete any value in a table, but they **cannot** alter a table (they cannot create/delete columns, or create/delete a table). The server side can implement this, but that kind of operation is a true horror to deal with (both for server and the database)

- Triggers can only be activated by an update/insert/delete operation. Queries do not count, meaning it is not possible for the database to trigger anything if the server is only reading values



```
+---------------------------+---------+------+-----+---------+-------+
| Field                     | Type    | Null | Key | Default | Extra |
+---------------------------+---------+------+-----+---------+-------+
| userid                    | int(11) | NO   | PRI | NULL    |       |
| kwh_produced_solarpanels  | float   | YES  |     | 0       |       |
| saved_co2_food            | float   | YES  |     | 0       |       |
| saved_co2_total           | float   | YES  |     | 0       |       |
| saved_price_food          | float   | YES  |     | 0       |       |
| saved_price_total         | float   | YES  |     | 0       |       |
| saved_co2_transport       | float   | YES  |     | 0       |       |
| saved_price_transport     | float   | YES  |     | 0       |       |
| saved_co2_solarpanels     | float   | YES  |     | 0       |       |
| saved_co2_temperature     | float   | YES  |     | 0       |       |
| saved_co2_localproduce    | float   | YES  |     | 0       |       |
| saved_price_solarpanels   | float   | YES  |     | 0       |       |
| saved_price_temperature   | float   | YES  |     | 0       |       |
| kwh_saved_temperature     | float   | YES  |     | 0       |       |
+---------------------------+---------+------+-----+---------+-------+
```

Current records table. All values are calculated by the database triggers based on inserts for each user



```
+----------------------+------------+------+-----+---------+-------+
| Field                | Type       | Null | Key | Default | Extra |
+----------------------+------------+------+-----+---------+-------+
| userid               | int(8)     | NO   | PRI | NULL    |       |
| tree_hugger          | tinyint(1) | YES  |     | 0       |       |
| anarcho_primitivist  | tinyint(1) | YES  |     | 0       |       |
| celebi               | tinyint(1) | YES  |     | 0       |       |
| stop_cheating        | tinyint(1) | YES  |     | 0       |       |
| sun_absorber         | tinyint(1) | YES  |     | 0       |       |
| power_plant          | tinyint(1) | YES  |     | 0       |       |
| 3_years_ago          | tinyint(1) | YES  |     | 0       |       |
| time_traveler        | tinyint(1) | YES  |     | 0       |       |
| vegetarian           | tinyint(1) | YES  |     | 0       |       |
| vegan                | tinyint(1) | YES  |     | 0       |       |
| photosynthesizer     | tinyint(1) | YES  |     | 0       |       |
| please_eat           | tinyint(1) | YES  |     | 0       |       |
| dutch                | tinyint(1) | YES  |     | 0       |       |
| nice_legs_bro        | tinyint(1) | YES  |     | 0       |       |
| teleporter           | tinyint(1) | YES  |     | 0       |       |
| never_skip_leg_day   | tinyint(1) | YES  |     | 0       |       |
| farmer               | tinyint(1) | YES  |     | 0       |       |
| hydroponic           | tinyint(1) | YES  |     | 0       |       |
| communist            | tinyint(1) | YES  |     | 0       |       |
| mudanjiang           | tinyint(1) | YES  |     | 0       |       |
| fat_wallet           | tinyint(1) | YES  |     | 0       |       |
| retirement_fund      | tinyint(1) | YES  |     | 0       |       |
| just_buy_something   | tinyint(1) | YES  |     | 0       |       |
| nokwg29              | tinyint(1) | YES  |     | 0       |       |
| level                | float      | YES  |     | 0       |       |
+----------------------+------------+------+-----+---------+-------+
```

Achievements for each user are based on values from the records table. Specific requirements for each activity can be found in the trigger list in the .xml file