To: Katrina Ward

From: Fish Locale

Subject: Revised Problem Statement

Date: December 4, 2016

**Original Problem Statement**

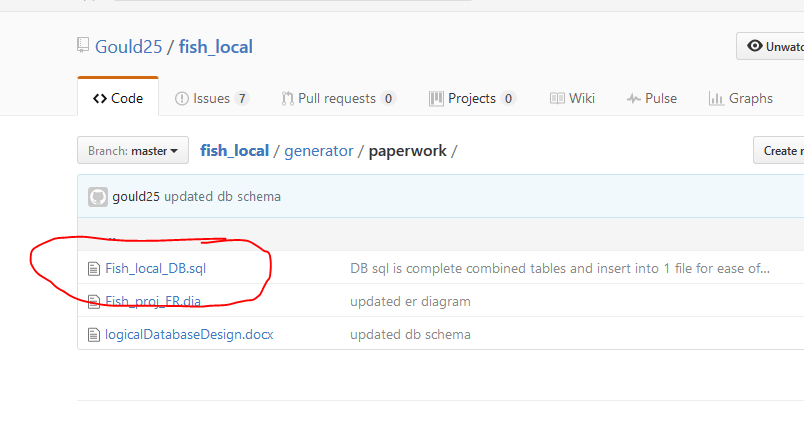
The purpose of this application is to catalog fishing at bodies of water in the state of Missouri. It is designed to catalog what kinds of fish can be caught at a given body of water and what kind of bait, rod and reel is best to be used for catching a certain species of fish at a given body of water. The application will store the latitude and longitude coordinates, name of the body of water, and other related information is outlined in the ER diagram. Additionally, the application will store information about the legal implications of catching a certain type of fish at a given body of water. Cataloging all this information and presenting it in an intuitive consumable way to the end user makes a database an essential component, as without one this endeavor will be futile.

**Final Problem Statement**

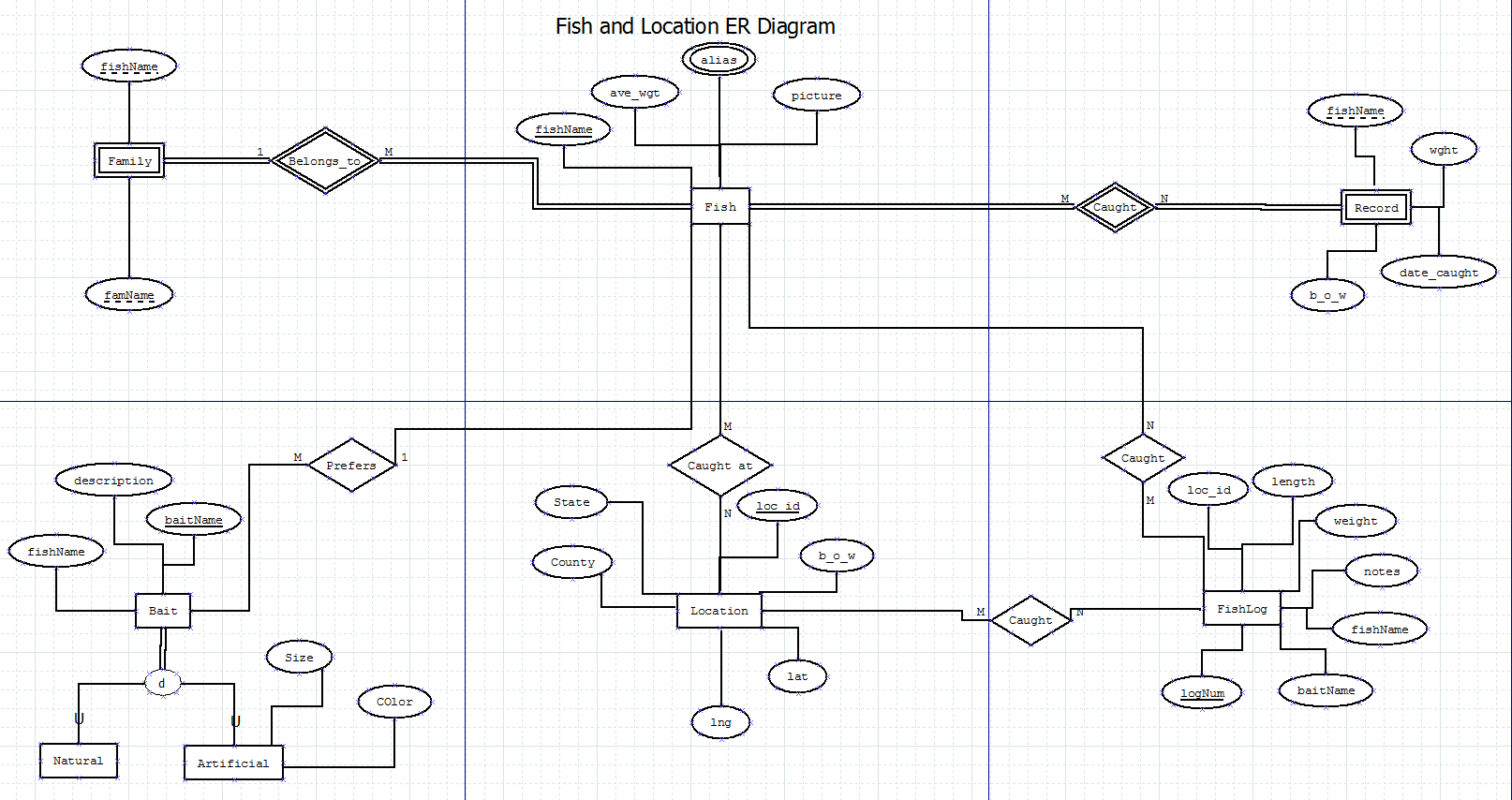
The overall purpose of creating a way for fisherman to store and access their locations, fish, and baits has remained the same. A Missouri records table was added to the DB and updates were also made to both the ER diagram and the Database schema for efficiency and comprehension since the last memo. The revisions are shown in the diagrams at the end of this document. At the time of this memo being written, all main objectives have been completed. The extras that include google maps and pictures of each fish are still unfinished with the hope of completion by the time of the presentation. Only the code from master branch will work.

**User’s Manual**

The assumption is being made that the user will have the ability to use a database interface application like PhpMyadmin and be able to use Github.

1. Clone the repository from <https://github.com/Gould25/fish_local> .
2. Navigate to the master/generator/paperwork and open the fish\_local\_DB.sql file.
3. Copy and paste the sql code from inside the file to your DB interface. This is the initial setup for the DB.
4. Open a terminal and type the following “npm start”.
5. In a browser navigate to <http://localhost:3000/> to enjoy the web app.

**Revised ER Diagram**



**Revised Logical Database Design**

Fish (

fishName -VARCHAR (15), NOT NULL, - fish name

ave\_wgt – INTEGER (4), NOT NULL, - average weight of the fish

picture – IMAGE, NOT NULL, - Picture of fish

)

Primary Key fishName

Record (

fishName -VARCHAR (15), NOT NULL, - fish name

b\_o\_w – VARCHAR (25), NOT NULL, - body of water

weight – INTEGER (4) , - weight of record fish

date\_caught – DATE - NOT NULL

)

Foreign Key fishName References fishName(Fish)ON UPDATE CASCADE ON DELETE CASCADE

Family (

fishName -VARCHAR (15), NOT NULL, - fish name

famName – VARCHAR (15), NOT NULL – fish family name

)

Foreign Key fishName References fishName(Fish) ON UPDATE CASCADE ON DELETE CASCADE

Alias (

fishName -VARCHAR (15), NOT NULL, - fish name

aliasName – VARCHAR (15), DEFAULT NULL – UNIQUE - alias’ or nick names for fish

)

Foreign Key fishName References fishName(Fish) ON UPDATE CASCADE ON DELETE CASCADE

Bait (

fishName -VARCHAR (15), NOT NULL, - fish name

baitName – VARCHAR (15), NOT NULL – name of bait

description – TEXT – description of bait

)

Primary Key baitName

Foreign Key fishName references fishName(Fish) ON UPDATE CASCADE ON DELETE SET NULL

Artificial (

baitName – VARCHAR (15), NOT NULL – name of bait

size – VARCHAR – size of bait

color – VARCHAR – color of bait

)

Primary Key baitName

Natural (

baitName – VARCHAR (15), NOT NULL – name of bait

)

Primary Key baitName

Location (

lat - FLOAT (10, 6) -, NOT NULL, - latitude coordinates

lng - FLOAT (10, 6) -, NOT NULL – longitude coordinates

b\_o\_w -VARCHAR (25) -, NOT NULL, body of water

state – VARCHAR (15) – DERIVED FROM lat and lng – state body of water is in

county – VARCHAR (15) – DERIVED FROM lat and lng – county body of water is in

fishName – VARCHAR (15) – NOT NULL – fish name

)

Primary Key (lat, lng)

Foreign Key fishName – References fishName(Fish)ON UPDATE CASCADE ON DELETE SET NULL

fishLog (

logNum – INTEGER (5) – NOT NULL AUTO INCREMENT– Log number

fishName – VARCHAR (15) – NOT NULL – fish name

loc\_id – INTEGER – NOT NULL – Location ID

baitName – VARCHAR(25) – NOT NULL – bait name

weight – INTEGER (4) – DEFAULT NULL – weight of fish

length – INTEGER (4) – DEFAULT NULL – length of fish

notes – TEXT (255) – DEFAULT NULL – User notes

)

Primary Key logNum

Foreign Key fishName – References fishName(Fish)ON UPDATE CASCADE ON DELETE SET NULL

Foreign Key loc\_id – References loc\_id(Location) ON UPDATE CASCADE ON DELETE SET NULL

Foreign Key baitName – References baitName(bait) ON UPDATE CASCADE ON DELETE SET NULL

**Function Description High Level**

No changes were necessary from last report.

/\*

\* Adds a fish entity to the database

\* fish - fish entity to be added

\* post: Updates database with new fish entity

\*/

function add\_fish($fish)

{

   If $fish already exists in the database return

   Otherwise create a new fish record in the database and populate its fields

   based on the values supplied by the $fish object

   Check to make sure the create operation did not fail

   If it did return an error stating why otherwise return nothing

}

/\*

\* Updates a fish entity in the database

\* fish - fish object with updated fields

\*     fish object should contain an id value

\* post: Updates record with new fields in database

\*/

function update\_fish($fish)

{

   Get a fish record with an id matching the one on the $fish object

   If this record does not exist return with an error stating as such

   Copy the fields from the $fish object to the entity and update the database

   If there are any referential constraint violations because foreign keys

   don't exist return an error stating this otherwise return nothing

}

/\*

\* Removes a fish entity in the database

\* id - record to be removed

\* post:  Updates fish entity to remove fish from database

\*/

function remove\_fish($id)

{

   Check if a fish record with an ID of $id exists.  If not return an error

   stating as such

   Delete fish record from the database

   If there are any referential integrity errors return an error stating this

   otherwise return nothing

}

/\*

\* Gets a fish entity in the database

\* id - record to be retrieved

\* post:  returns a fish object populated with the values from the database

\*      fields

\*/

function get\_fish($id)

{

   Check if a fish record with an ID of $id exists.  If not return an error

   stating as such

   Instantiate a new fish object

   Copy the values from the database record fields to the fields on the

   fish object

   return the fish object

}

/\*

\* Adds a record entity to the database

\* record - record entity to be added

\* post: Updates database with new record entity

\*/

function add\_record($record)

{

   If $record already exists in the database return

   Otherwise create a new record entity in the database and populate its fields

   based on the values supplied by the $record object

   Check to make sure the create operation did not fail

   If it did return an error stating why otherwise return nothing

}

/\*

\* Updates a record entity in the database

\* record - record object with updated fields

\*     record object should contain an id value

\* post: Updates record with new fields in database

\*/

function update\_record($record)

{

   Get a record entity with an id matching the one on the $record object

   If this record does not exist return with an error stating as such

   Copy the fields from the $record object to the entity and update the database

   If there are any referential constraint violations because foreign keys

   don't exist return an error stating this otherwise return nothing

}

/\*

\* Removes a record entity in the database

\* id - record to be removed

\* post:  Updates record entity to remove record from database

\*/

function remove\_record($id)

{

   Check if a record entity with an ID of $id exists.  If not return an error

   stating as such

   Delete record entity from the database

   If there are any referential integrity errors return an error stating this

   otherwise return nothing

}

/\*

\* Gets a record entity in the database

\* id - record to be retrieved

\* post:  returns a record object populated with the values from the database

\*      fields

\*/

function get\_record($id)

{

   Check if a record entity  with an ID of $id exists.  If not return an error

   stating as such

   Instantiate a new record object

   Copy the values from the database record fields to the fields on the

   record object

   return the record object

}

/\*

\* Adds a alias entity to the database

\* alias - alias entity to be added

\* post: Updates database with new alias entity

\*/

function add\_alias($alias)

{

   If $alias already exists in the database return

   Otherwise create a new alias record in the database and populate its fields

   based on the values supplied by the $alias object

   Check to make sure the create operation did not fail

   If it did return an error stating why otherwise return nothing

}

/\*

\* Updates a alias entity in the database

\* alias - alias object with updated fields

\*     alias object should contain an id value

\* post: Updates record with new fields in database

\*/

function update\_alias($alias)

{

   Get a alias record with an id matching the one on the $alias object

   If this record does not exist return with an error stating as such

   Copy the fields from the $alias object to the entity and update the database

   If there are any referential constraint violations because foreign keys

   don't exist return an error stating this otherwise return nothing

}

/\*

\* Removes a alias entity in the database

\* id - record to be removed

\* post:  Updates alias entity to remove alias from database

\*/

function remove\_alias($id)

{

   Check if a alias record with an ID of $id exists.  If not return an error

   stating as such

   Delete alias record from the database

   If there are any referential integrity errors return an error stating this

   otherwise return nothing

}

/\*

\* Gets a alias entity in the database

\* id - record to be retrieved

\* post:  returns a alias object populated with the values from the database

\*      fields

\*/

function get\_alias($id)

{

   Check if a alias record with an ID of $id exists.  If not return an error

   stating as such

   Instantiate a new alias object

   Copy the values from the database record fields to the fields on the

   alias object

   return the alias object

}

/\*

\* Adds a bait entity to the database

\* bait - bait entity to be added

\* post: Updates database with new bait entity

\*/

function add\_bait($bait)

{

   If $bait already exists in the database return

   Otherwise create a new bait record in the database and populate its fields

   based on the values supplied by the $bait object

   Check to make sure the create operation did not fail

   If it did return an error stating why otherwise return nothing

}

/\*

\* Updates a bait entity in the database

\* bait - bait object with updated fields

\*     bait object should contain an id value

\* post: Updates record with new fields in database

\*/

function update\_bait($bait)

{

   Get a bait record with an id matching the one on the $bait object

   If this record does not exist return with an error stating as such

   Copy the fields from the $bait object to the entity and update the database

   If there are any referential constraint violations because foreign keys

   don't exist return an error stating this otherwise return nothing

}

/\*

\* Removes a bait entity in the database

\* bait\_name - record to be removed

\* post:  Updates bait entity to remove bait from database

\*/

function remove\_bait($bait\_name)

{

   Check if a bait record with an ID of $bait\_name exists.  If not return an error

   stating as such

   Delete bait record from the database

   If there are any referential integrity errors return an error stating this

   otherwise return nothing

}

/\*

\* Gets a bait entity in the database

\* bait\_name - record to be retrieved

\* post:  returns a bait object populated with the values from the database

\*      fields

\*/

function get\_bait($bait\_name)

{

   Check if a bait record with an ID of $bait\_name exists.  If not return an error

   stating as such

   Instantiate a new bait object

   Copy the values from the database record fields to the fields on the

   bait object

   return the bait object

}

/\*

\* Adds a location entity to the database

\* location - location entity to be added

\* post: Updates database with new location entity

\*/

function add\_location($location)

{

   If $location already exists in the database return

   Also make sure latitude and longitude are valid if not return an error

   stating as such

   Otherwise create a new location record in the database and populate its fields

   based on the values supplied by the $location object

   Check to make sure the create operation did not fail

   If it did return an error stating why otherwise return nothing

}

/\*

\* Updates a location entity in the database

\* location - location object with updated fields

\*     location object should contain an id value

\* post: Updates record with new fields in database

\*/

function update\_location($location)

{

   Get a location record with an id matching the one on the $location object

   If this record does not exist return with an error stating as such

   Copy the fields from the $location object to the entity and update the database

   If there are any referential constraint violations because foreign keys

   don't exist return an error stating this otherwise return nothing

}

/\*

\* Removes a location entity in the database

\* lat and lng - coordinates to be removed

\* post:  Updates location entity to remove location from database

\*/

function remove\_location($lat, $lng)

{

   Check if a location record with a latitude of $lat, and longitude of $lng

   exists.  If not return an error  stating as such

   Delete location record from the database

   If there are any referential integrity errors return an error stating this

   otherwise return nothing

}

/\*

\* Gets a location entity in the database

\* lat and lng - coordinates to be retrieved

\* post:  returns a location object populated with the values from the database

\*      fields

\*/

function get\_location($lat, $lng)

{

   Check if a location record with a latitude of $lat, and longitude of $lng

   exists.   If not return an error stating as such

   Instantiate a new location object

   Copy the values from the database record fields to the fields on the

   location object

   return the location object

}

/\*

\* Adds a legality entity to the database

\* legality - legality entity to be added

\* post: Updates database with new legality entity

\*/

function add\_legality($legality)

{

   If $legality already exists in the database return

   Otherwise create a new legality record in the database and populate its fields

   based on the values supplied by the $legality object

   Check to make sure the create operation did not fail

   If it did return an error stating why otherwise return nothing

}

/\*

\* Updates a legality entity in the database

\* legality - legality object with updated fields

\*     legality object should contain an id value

\* post: Updates record with new fields in database

\*/

function update\_legality($legality)

{

   Get a legality record with an id matching the one on the $legality object

   If this record does not exist return with an error stating as such

   Copy the fields from the $legality object to the entity and update the database

   If there are any referential constraint violations because foreign keys

   don't exist return an error stating this otherwise return nothing

}

/\*

\* Removes a legality entity in the database

\* regnum - record to be removed

\* post:  Updates legality entity to remove legality from database

\*/

function remove\_legality($regnum)

{

   Check if a legality record with a registration number of $regnum exists.

   If not return an error stating as such

   Delete legality record from the database

   If there are any referential integrity errors return an error stating this

   otherwise return nothing

}

/\*

\* Gets a legality entity in the database

\* regnum - record to be retrieved

\* post:  returns a legality object populated with the values from the database

\*      fields

\*/

function get\_legality($regnum)

{

   Check if a legality record with a registration number of $regnum exists.

   If not return an error stating as such

   Instantiate a new legality object

   Copy the values from the database record fields to the fields on the

   legality object

   return the legality object

}

/\*

\* Gets all locations coordinates for Google maps

\* post: returns array of tuples for latitude and longitude

\*/

function show\_locations()

{

   Get all the location records from the database

   Copy latitude and logitude for each location into a tuple which will then

   be stored in an array

   return the array

}

/\*

\* Gets all locations coordinates for Google maps for a type of fish

\* fish\_name - name of fish to be returned

\* post: returns array of tuples for latitude and longitude

\*/

function show\_locations($fish\_name)

{

   Make sure $fish\_name actually exists if not return an error stating

   as such

   Join fish records to location records and return retrieved records

   Copy latitude and logitude for each location into a tuple which will then

   be stored in an array

   return the array

}

/\*

\* Gets all locations coordinates for Google maps for a type of fish

\* latitude - latitude, lng - longitude

\* post: returns anonymous type from join

\*/

function get\_fish\_at\_location($lat, $lng)

{

 Make sure a location record exists at latitude $lat and longitude

 $lng if not return an error stating as such

 Join fish, location, bait, and legalities for latitude $lat and longitude

 $lng as an anonymous type

 return the resulting objects

}

/\*

\* Adds a family entity to the database

\* family - family entity to be added

\* post: Updates database with new family entity

\*/

function add\_family($family)

{

   If $family already exists in the database return

   Otherwise create a new family record in the database and populate its fields

   based on the values supplied by the $family object

   Check to make sure the create operation did not fail

   If it did return an error stating why otherwise return nothing

}

/\*

\* Updates a family entity in the database

\* family - family object with updated fields

\*     family object should contain an id value

\* post: Updates record with new fields in database

\*/

function update\_family($family)

{

   Get a family record with an id matching the one on the $family object

   If this record does not exist return with an error stating as such

   Copy the fields from the $family object to the entity and update the database

   If there are any referential constraint violations because foreign keys

   don't exist return an error stating this otherwise return nothing

}

/\*

\* Removes a family entity in the database

\* fish\_name - fish name to be removed, family\_name - family name to be

\*      removed

\* post:  Updates family entity to remove family from database

\*/

function remove\_family($fish\_name, $fam\_name)

{

   Check if a family record with a fish name of $fish\_name, and family name

   of $fam\_name exists.  If not return an error stating as such

   Delete family record from the database

   If there are any referential integrity errors return an error stating this

   otherwise return nothing

}

/\*

\* Gets a family entity in the database

\* fish\_name - fish name to be retrieved, family\_name - family name to be

\*      retrieved

\* post:  returns a family object populated with the values from the database

\*      fields

\*/

function get\_family($fish\_name, $fam\_name)

{

   Check if a family record with a fish name of $fish\_name, and family name

   of $fam\_name exists.  If not return an error stating as such

   Instantiate a new family object

   Copy the values from the database record fields to the fields on the

   family object

   return the family object

}