

CSCI 2141, Fall 2014

Final Project Report

Team member 1, Student ID: B00642710, Student Name: Jason Parsons

Team member 2, Student ID: B00674452, Student Name: Wanru Tian

Submission date: Dec. 2, 2014

Introduction:

This report details the design and implementation of a database system for the storage and organization of data gathered by a theoretical pelagic predator tagging system. This system is modelled after the TOPP Survey (Tagging of Pelagic Predators) run by Stanford University and its partners. The survey involves the radio tagging of a variety of marine species in order to track their movements and data pertaining to depth, temperature, location etc. The raw data for the TOPP Survey can be found at <http://www.gtopp.org/> as well as their sister site <http://oceanview.pfeg.noaa.gov/ATN/#>. It was our original intention to use this data directly; however the server housing the most complete data sets were lost in a server crash at Stanford's end leaving us with limited data available. As a result we have made use of certain data elements provided by TOPP and extrapolated our own test data in place of that which was lost due to the server's crash. The data from TOPP and the data we have extrapolated will be outlined later in this report.

The original data from the gtopp website contained data for numerous marine species. Each data table contained the exact same columns for each species represented. In analyzing their structure we noticed a great deal of redundancy, and the tables were not in 1NF. This makes them hard to read and find the tuples/columns you are interested in observing/analyzing. In an attempt to resolve this redundancy and poor organization we have devised our own database and web application entitled PPTS (Pelagic Predator Tagging System) which uses real world systems such as the TOPP survey as inspiration. In doing so, this project presents one possible solution for making data collected by organizations such as TOPP better organized, indexed and readable, while reducing redundancy and potential insert and deletion errors.

System design and Development:

The data tables from the gtopp website were arranged with columns representing date/time, tag_id, depth, internal temperature, external temperature, light, latitude, and longitude. All of this data was in the same table and was presented the same for every animal. Therefore the site contained an overabundance of data tables that all look the same for every species of animal. This leads to confusion over where and how to access the data you are specifically looking for as you have to scan through a list of tables, and then scan through them to find the info you want to look at. Also, date and time were represented by one column which violates 1NF which states that each column must represent an atomic value. Our first task was to normalize this to 1NF which would further allow us to observe the functional dependencies involved and decomposing it further into 2NF and 3NF forms.

The TOPP system, as well as other similar tracking systems, works by using radio transmitters attached to the animals which then send signals that are received by a system of buoys in the ocean. The buoys receive the signals and transmit the data to a satellite system that then relays them to stations on land that gather and collect them. From there the data is organized, analyzed and published for scientists and the public to view. In our theoretical model we are treating the animal, the buoy, the station on land, and the readings themselves as our entity sets. Each set will have a series of attributes which will be defined below.

Before doing so, it is important to look at how 1NF was achieved. In the TOPP model the animal entity type looked like the following:

animal (date-time, depth, int_temp, ext_temp, light, longitude, latitude)

After TOPP's server went down this was reduced further to just the date-time, latitude and longitude. After their server crashed the columns were reduced further as seen below:

```
Blue Shark 1614001 120689
track:
Date Time, Latitude (N), Longitude (E)
12-Jul-2014 11:40, 33.202, -121.289
15-Jul-2014 10:28, 32.745, -122.611
18-Jul-2014 02:55, 32.847, -123.317
18-Jul-2014 11:21, 32.947, -123.492
18-Jul-2014 05:22, 33.05, -123.484
18-Jul-2014 07:49, 33.099, -123.487
18-Jul-2014 10:44, 33.151, -123.438
21-Jul-2014 12:18, 33.61, -123.199
21-Jul-2014 12:54, 33.605, -123.187
21-Jul-2014 01:17, 33.614, -123.194
```

To normalize this to 1NF we proposed this solution:

animal(date, time, depth, int_temp, ext_temp, light, longitude, latitude)

The TOPP data also contained a ptt number for each animal tagged. The ptt id is the second number after the species name as seen above. The ptt number represents the id of the tag itself for each animal allowing it to be identified. However, in their schema, the ptt is not a part of the regular data table, but is instead part of the label identifying the animal on their website. To better utilize the ptt id number, we have made it a part of our normalized table schema. There is also no mention of which buoy picked up the signal from that particular tag in the animal data set. That information is located in separate files and are represented by graphs with data points representing the ptt tag id's and the x-axis representing the date. Because this data is scattered amongst many different tables, with no defined foreign keys to maintain data integrity, we believe this makes their data hard to read, and even harder to analyze.

Our solution to the redundancy and integrity issues is to further normalize the 1NF form using four entity types. They are as follows:

animal

buoy

station

reading

Our theoretical model will be running on the assumption that each buoy in the ocean reports to one of three stations on land, and reports to the same station every time. Likewise, in our model we are only looking at three buoys for the sake of scope and time. Because the satellite system was not defined by TOPP or any of the other organizations that do this form of animal tagging, we are treating the satellites as non-entities for the sake of keeping our theoretical model simple and due to a lack of real world data to determine how that system works. So we are assuming that each buoy talks to its designated onshore station directly. In essence the flow of data is as follows:

animal -> buoy -> station

This is one reason we have chosen to represent each as an entity type in our database schema. The other reason we are doing so is in analyzing the decomposition of the 1NF. Each animal is identified by its ptt as well as its species name. Each buoy is represented by a buoy_id, its latitude position, and its longitude position. Each station has a station_id, a name, and a location (identified by city/state/province).

For our reading table, the attributes specific to each is the date, time, depth, int_temp, ext_temp, light, latitude position, longitude position. In order to allow these tables to interact with each other, we require a system of primary and foreign keys. We are proposing that every reading tuple should contain the buoy_id of the buoy that received the signal. This would act as a foreign key for table reading. Also, every reading tuple should contain the ptt id # of the animal that was tracked which will also act as a foreign key to link it to the animal table. Each tuple in the buoy table will use the station_id of the station it reports to as a foreign key to link it to the station table.

We have identified the following primary keys for each table in our database schema:

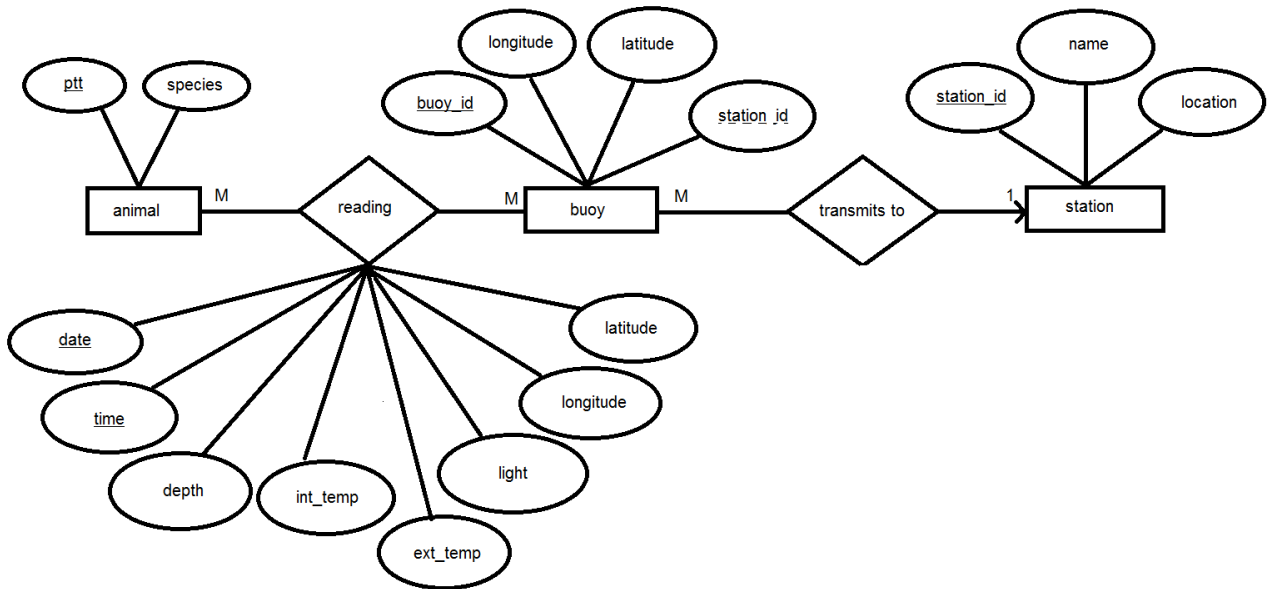
animal - > ptt

buoy -> buoy_id

station -> station id

reading -> date, time, buoy id

Note that buoy_id in table reading also acts as a foreign key for accessing the buoy table. Our proposed ER diagram for this schema is as follows:



From this, our proposed table schema is:

animal

<u>ptt</u>	species
------------	---------

buoy

<u>buoy_id</u>	longitude	latitude	<u>station_id</u>
----------------	-----------	----------	-------------------

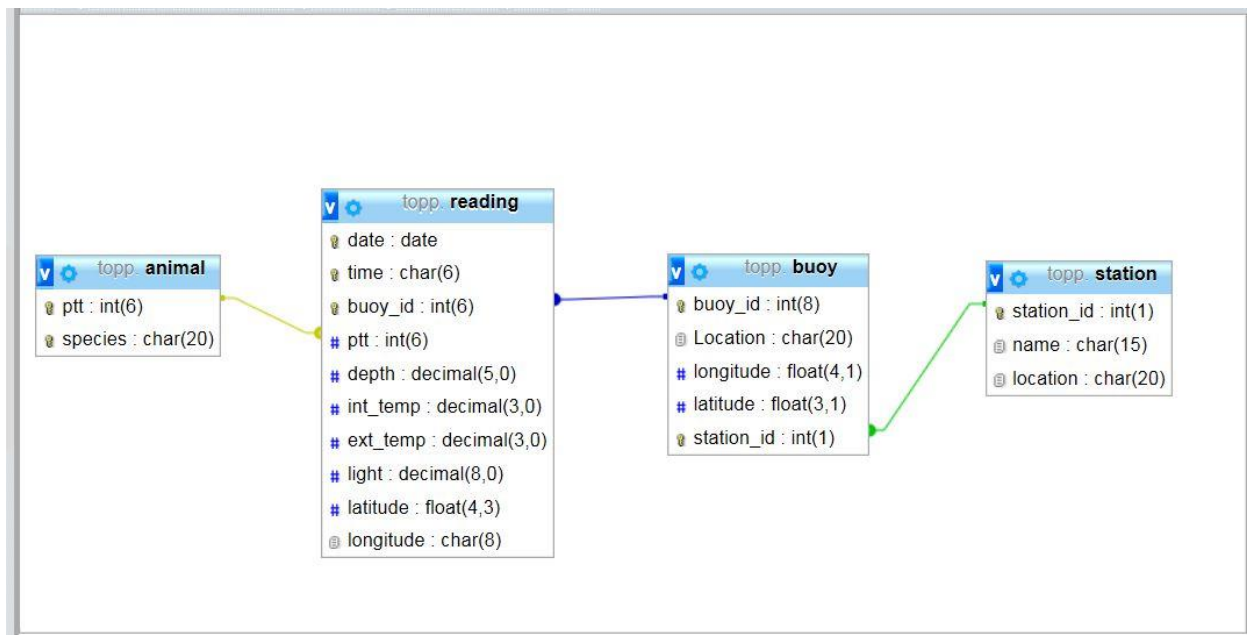
station

<u>station_id</u>	name	location
-------------------	------	----------

reading

<u>date</u>	<u>time</u>	<u>buoy_id</u>	<u>ptt</u>	depth	int_temp	ext_temp	light	longitude	latitude
-------------	-------------	----------------	------------	-------	----------	----------	-------	-----------	----------

Represented another way using phpmyadmin's visualization tool:



Our schema is now deconstructed into a 3NF structure. Each entity set has a primary key, and no non-primary attributes are dependent on each other. The reading table has functional dependencies on the animal and buoy tables through the use of foreign key's: ptt and buoy_id. Likewise the buoy table is functionally dependent on the station table through use of the foreign key: station_id. This structure helps to better organize the system used by TOPP by removing redundancy and ensuring data integrity through the use of primary and foreign key constraints.

Our table should now allow for insertion, selection, and deletion of tuple data while maintaining data integrity. By separating the station, buoy, and animal entities from being contained in one table with the reading data we make sure that the deletion of one tuple does not break our database and that any change to a station or buoy cannot be done without a check against which tuples in reading will be affected. We also ensure data integrity through the use of three triggers. All 3 triggers, as well as the sql database itself were created using the xampp/phpmyadmin software package. Our triggers are as follows:

Trigger 1:

Details

Trigger name	Trigger 1
Table	reading
Time	BEFORE
Event	INSERT
Definition	<pre>1 BEGIN 2 DECLARE dummy,baddata INT; SET baddata = 0; 3 IF (NEW.light < 0) or (New.light > 100000) THEN SET baddata = 1; 4 End if; 5 IF baddata = 1 THEN 6 signal sqlstate '45000' set message_text = " Invalid insert!"; 7 End if; 8 END</pre>
Definer	root@localhost

```
CREATE TRIGGER `Trigger 1` BEFORE INSERT ON `reading` FOR EACH ROW BEGIN DECLARE dummy,baddata INT; SET baddata = 0; IF (NEW.light < 0) or (New.light > 100000 ) THEN SET baddata = 1; End if; IF baddata = 1 THEN signal sqlstate '45000' set message_text = " Invalid insert!"; End if; END
```

This trigger ensures that any new entry in the column light in the table reading has a value greater than 0 and less than 100,000. This is to ensure that any new entries conform to the possible values of light measurement (in Lux) which cannot exceed or fall below these values in order to be a true reading of light intensity in the real world.

If this trigger is fired, the sqlstate is set to 45000 which will cause the insert to fail and the user will receive a message notifying them of such. If the trigger is not fired, the insert proceeds normally.

Trigger 2:

Details

Trigger name	Trigger 2
Table	station
Time	BEFORE
Event	INSERT
Definition	<pre>1 BEGIN 2 DECLARE dummy,baddata INT; SET baddata = 0; 3 IF (NEW.location != 'Monterey CA') or (New.location != 'Victoria BC') or (New.location != 'Honolulu HI') THEN SET baddata = 1; 4 End if; 5 IF baddata = 1 THEN 6 signal sqlstate '45000' set message_text = " Invalid insert!"; 7 End if; 8 END</pre>
Definer	root@localhost

```
CREATE TRIGGER `Trigger 2` BEFORE INSERT ON `station` FOR EACH ROW BEGIN DECLARE dummy,baddata INT; SET baddata = 0; IF (NEW.location != 'Monterey CA') or (New.location != 'Victoria BC') or (New.location != 'Honolulu HI') THEN SET baddata = 1; End if; IF baddata = 1 THEN signal sqlstate '45000' set message_text = " Invalid insert!"; End if; END
```

This trigger is set for the attribute location in the station table. As mentioned earlier in the report, there are only three available stations. They are: Monterey CA, Victoria BC, and Honolulu HI. This trigger is used to ensure that any new tuple in station must be located in one of these three cities. If not, the trigger fires, sqlstate 45000 is set, the insert fails, and the user is notified.

Like trigger 1, we have chosen to check this condition before insertion rather than after given that any insertion in this table maintains the integrity of our database structure and that of the tagging survey project itself.

Trigger 3:

Details

Trigger name	Trigger 3
Table	buoy
Time	BEFORE
Event	INSERT
Definition	<pre>1 BEGIN 2 DECLARE dummy,baddata INT; SET baddata = 0; 3 IF (NEW.latitude > 90) or (New.latitude < -90) or (New.longitude > 0) or (New.longitude < -360) THEN SET baddata = 1; 4 End if; 5 IF baddata = 1 THEN 6 signal sqlstate '45000' set message_text = " Invalid insert!"; 7 End if; 8 END</pre>
Definer	root@localhost

This trigger is set for the columns latitude and longitude in the table buoy. It is a compound trigger that ensures that any new tuple in the buoy table must have valid data values for the latitude and longitude attributes. Latitude is measured from -90 to +90 and longitude is measured from 0 to -360 degrees. Again, as with triggers 1 and 2, this condition is checked before insertion and sqlstate 45000 is set and the user notified if the trigger fires.

All three of these triggers are used to ensure that any new data being entered into the PPTS database conforms to specific data ranges as demanded by the administrators as well as the constraints of the natural world.

One example of an insertion causing a trigger to fire is as follows:

```
Error
SQL query:

INSERT INTO `reading` (`date`, `time`, `buoy_id`, `ptt`, `depth`, `int_temp`, `ext_temp`, `light`, `latitude`, `longitude`) VALUES ('2014-01-28', '3:16', '111222', '222111', '20', '25', '2

MySQL said:
#1452 - Cannot add or update a child row: a foreign key constraint fails (`topp`.`reading`, CONSTRAINT `reading_ibfk_1` FOREIGN KEY (`buoy_id`) REFERENCES `buoy` (`buoy_id`))
```

In this case the `buoy_id` we are attempting to insert does not conform to the values specified in the `buoy` table. By firing, this trigger maintains the integrity of the `buoy_id` attribute and the tuples in the `reading` table.

Sample tuples from our various tables:

From table `animal`:

```
SELECT * FROM `animal`
```

Number of rows: 25 Filter rows: Search this table

Sort by key: None

+ Options

			ptt	species
<input type="checkbox"/>	Edit	Copy	Delete	1388 Blue Whale
<input type="checkbox"/>	Edit	Copy	Delete	49101 Hawksbill Turtle
<input type="checkbox"/>	Edit	Copy	Delete	67553 Laysan Albatross
<input type="checkbox"/>	Edit	Copy	Delete	126318 Silky Shark
<input type="checkbox"/>	Edit	Copy	Delete	126347 Blue Marlin
<input type="checkbox"/>	Edit	Copy	Delete	135902 Silvertip Shark










From table buoy:

```
SELECT * FROM `buoy`
```

Number of rows: 25 Filter rows: Search this table

Sort by key: None

+ Options

				buoy_id	Location	longitude	latitude	station_id
<input type="checkbox"/>	 Edit	 Copy	 Delete	200050	Tomalos	-123.0	38.2	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	200075	Avo Nuevo	-122.3	37.1	2
<input type="checkbox"/>	 Edit	 Copy	 Delete	200076	Farallones	-123.0	37.1	3










From table station:

```
SELECT * FROM `station`
```

Number of rows: 25 Filter rows: Search this table

Sort by key: None

+ Options

				station_id	name	location
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	Alpha	Monterey CA
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	Beta	Victoria BC
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	Omega	Honolulu HI

From table reading:

✓ Showing rows 0 - 24 (178 total, Query took 0.0000 seconds.) [date: 2008-01-12 - 2008-01-22]

```
SELECT * FROM `reading` ORDER BY `date` ASC
```

1 Show all > >> | Number of rows: 25 Filter rows: Search this table

Sort by key: None

+ Options

			date	time	buoy_id	ptt	depth	int_temp	ext_temp	light	latitude	longitude
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-12	10:53	200075	1388	2	31	33	4854	8.891 -96.59
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-14	8:55	200076	1388	8	30	34	4863	8.794 -96.698
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-15	10:03	200076	1388	24	31	33	4621	9.110 -96.712
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-15	8:50	200075	1388	1	31	30	4921	9.087 -96.623
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-16	11:54	200075	1388	110	30	24	699	8.952 -96.841
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-16	8:38	200076	1388	61	31	28	4277	8.989 -96.81
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-17	10:39	200050	1388	2	31	32	4931	9.228 -97.389
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-17	11:32	200075	1388	21	31	26	4258	9.182 -97.144
<input type="checkbox"/>	Edit	Copy	Delete	2008-01-17	7:37	200050	1388	36	31	26	4011	9.253 -97.291

Our reading table contains 178 tuples. This is just a small snapshot of that table's content.

We also have an extra table set up called 'users' which was created to allow us to set up a login system on our web application. It contains three attribute columns: id, username, and password. This information is passed to our web application's php code which will be detailed later in this report. The users table looks as follows:

✓ Showing rows 0 - 0 (1 total, Query took 0.0000 seconds.)

```
SELECT * FROM `users`
```

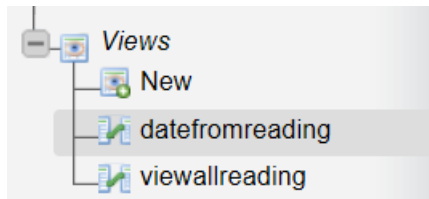
Number of rows: 25 Filter rows: Search this table

+ Options

id	username	password
0	jasonzoe	jasonzoe

We've left the username and password visible in this report so the professor and TA's marking our project will be able to access the website. They will also be provided in a Read Me file in our project submission folder along with other instructions.

Our database also has two test views set up to ensure that that functionality works. They are as follows:



datefromreading -> which selects the date column to display from the table reading.

```
SELECT * FROM `datefromreading`
```

viewallreading -> which displays all tuples in table reading.

```
SELECT * FROM `datefromreading`
```

Each of these views was set up to test their functionality in phpmyadmin and ensure they work. We did not, however, use them as part of our web interface application. This will be detailed later in the report.

Result Demonstration:

The goal of this project was to make data collected by projects like the TOPP survey more useable and readable for various stakeholders such as scientists and the general public. In doing so the data itself becomes of much greater use given that it can be searched and indexed to narrow in on exactly the tuples and columns they are interested in viewing. This makes analysis both easier, and faster which leads to greater efficiency and ease of use, therefore making the data that much more valuable to those looking to make use of it.

For the purposes of our project, and due to time constraints and the high learning curve of learning to use php, we have restricted our web application to the parsing and viewing of data from the 'reading' table. Our web application also forces any user to have to log in and log out

of the system in order to make use of it. Once logged in, the user is presented with a screen that allows the user to pick which columns of data they want to look at from reading through a series of 10 checkboxes. It also allows them to order the data by a variety of attribute types and to sort the results in either ascending or descending order. After the user selects which options they'd like to search by and hits the submit button a dynamic php query is run which presents the results in a table format.

The login page:



Enter your login and password

Username:

Password:

The path for reaching this screen is: <http://localhost/2141/login.php>

For the purposes of testing, presentation, and marking by you the professor and teaching assistants, we have set the username and password each to 'jasonzoe'. You may use that to log yourself in if needed during marking.

The logo at the top contains the abbreviation for our project, PPTS (Pelagic Predator Tagging System). Once past the login screen, the logo can be clicked on at any time to return you to the main search page.

If the login credentials passed to the username and/or password fields fail, the user will be presented with a message that their login failed. If successful, the user will be brought to the main search screen which looks as follows:



[LOG IN/OUT](#) [CONTACT](#)

Sort By Sort Direction

Include:

☐ Date ☐ Depth ☐ Time ☐ Light ☐ Int-Temp ☐ Ext-Temp ☐ Buoy ID ☐ PTT ☐ Latitude ☐ Longitude

This is a standard php form using a series of if statements to run the series of checkboxes in a file called topp.php which will be included in our project submission. As stated earlier, once the user selects their options, they then click on the submit button to process their query. The user also has the option of logging out or visiting the contact page using the navigation strip just below the project logo. Also, as stated before, the user can click on the logo at any time from any screen to return to this search page.

The contact screen looks as follows:



[LOG IN/OUT](#) [CONTACT](#)

Contacts:

Jason Parsons

Email: Jason.Parsons@dal.ca

Wanru, Tian

Email: tn608865@dal.ca

We will now present a sample query using the form. In this example the user wishes to see the date, time, buoy_id and light sorted by buoy_id in descending order. The search screen is as follows:



LOG IN/OUT

CONTACT

Sort By Sort Direction

Include:

☒ Date ☐ Depth ☒ Time ☒ Light ☐ Int-Temp ☐ Ext-Temp ☒ Buoy ID ☐ PTT ☐ Latitude ☐ Longitude

Search

The results after hitting submit are:



LOG IN/OUT

CONTACT

Connection established

Query: SELECT * FROM reading ORDER BY buoy_id DESC executed

Date	Time	buoy_id	light
2008-01-14	8:55	200076	4863
2008-01-15	10:03	200076	4621
2008-01-16	8:38	200076	4277
2008-01-19	7:15	200076	580
2008-01-20	10:42	200076	596
2008-01-22	10:21	200076	4993
2008-01-23	10:09	200076	4882
2008-01-24	8:54	200076	4610
2008-01-25	11:42	200076	4112

The resulting table is 178 tuples long so we have chosen to abbreviate our screenshot. Due to the limitations of time, and our unfamiliarity with php and the steep learning curve involved, we have had to limit our web application to these operations. Further expansion of this web app could include the ability to enter specific attribute search criteria such as searching buoy_id by allowing the user to enter a specific one in to the form and have the results only pertain to that particular buoy. It could also be expanded to display graphs based on the search menu to show the locations of animals and buoys in relation to one another in 2D space. Each of these was out of the scope of our project, but there is quite a bit of expansion that could be done if interested parties are willing to proceed.

Conclusions:

It was our goal to provide a proof of concept for projects such as Stanford University's Topp Survey to show that the data in its current form can be of much greater use to researchers and the public alike if it is organized in such a way as to allow faster, more efficient, and narrower targeted searching capabilities. What we have provided can be considered a first step in that direction. Further research in this area could include the addition of greater searching power to our web application to even further narrow the user's search to exactly the records, tuples, columns and tables they want to look at. It could also be expanded to include imaging capability similar but further expanded to that currently available on the gtopp and oceanview websites.

In doing this project we have gained a better understanding of how relational databases are designed and created, as well as first-hand experience with some of the tools used to do so (mysql, bluenose, xampp, phpmyadmin, php scripting, html, etc.). We also gained a better understanding of the needs and challenges faced by researchers involved in the tagging and study of wild animals with specific attention paid to marine species.

The main challenges for our project group concerned the design of our project and the database as well as becoming familiar with the tools used to create it. Particular struggle was found in the use of php to create our web application. A large percentage of our time in the latter half of the project was spent learning php and dealing with the difficulties therein. For future classes in 2141, we would recommend more time spent (either in class or in tutorial) preparing students to use php and other web development tools so that more time can be spent on the database side of the project instead of getting bogged down in the php/web programming side.

Acknowledgements:

We would like to thank Dr. Randy Kochevar from Stanford University for his generous help in understanding how the TOPP system is administrated and the various terms and tolls used. Dr. Kochevar helps maintain the gtopp.org website and is one of the researchers involved in the TOPP Survey. The server that was used to house the most robust datasets suffered a failure roughly 3 weeks before our project came due and despite Dr. Kochevar's best efforts he was unable to recover the data for us to use. This is why we have had to extrapolate some of the column data as outlined earlier in the report to fill in the gaps.

The main TOPP Survey website can be found at:

<http://www.gtopp.org/>

Their sister site which serves as home to the less robust datasets we made use of can be found at:

<http://oceanview.pfeg.noaa.gov/ATN/#>