GEOG 897D: Spatial Databases
Professor: Jim Detwiler
April 27, 2013

Final Project: Citizen Science Data Collection Application

The goal of the Final Project is taking the culmination of knowledge gained throughout GEOG 897D and developing a spatial database to solve a problem of our own choosing. This report will discuss the problem the spatial database intends to solve and how it was accomplished. Given the time constraints, the focus is on the back end database rather than the full application.

Problem

The Maine coastline is long and beautiful, but many of the sandy beaches are eroding away. In order to combat erosion, features are constructed to hopefully alleviate the destruction. Sometimes the features work, but at what cost? When erosion is stopped and accretion begins, the sand being deposited is usually coming from another spot along the beach. Essentially, the erosion has been pushed from the original location to another portion of the beach. To keep track of changes in beach erosion and accretion, Maine has established a beach scoring system that will hopefully aid in the management of their sandy shoreline. In what is often referred to as citizen science, the system uses residents as volunteers to make and record manual measurements. The system has been effective so far, but it creates a great deal of paperwork.

Creation of a database to hold the manual measurements will make keeping track of the data easier. It will help to alleviate the delays associated with mailing the paperwork. Finally, it will allow scientists to quickly perform advanced analysis on collected data.

Database Description

While the database is designed to effectively store individual data points, it also takes into account features that needed for both spatial and temporal analysis. The database consists of five (5) tables (Figure 1).

Table	Description
beaches	polygons that represent specific beaches
points	holds data measurements
profiles	profile linestrings of collected measurements
teams	collection team information
volunteers	resident volunteer information

Figure 1: Table Descriptions

Each table has a serial data type primary key (PK) and two (2) tables, i.e. points and teams, have foreign keys (FK). The columns for each table are represented by an entity-relationship diagram (Figure 2).

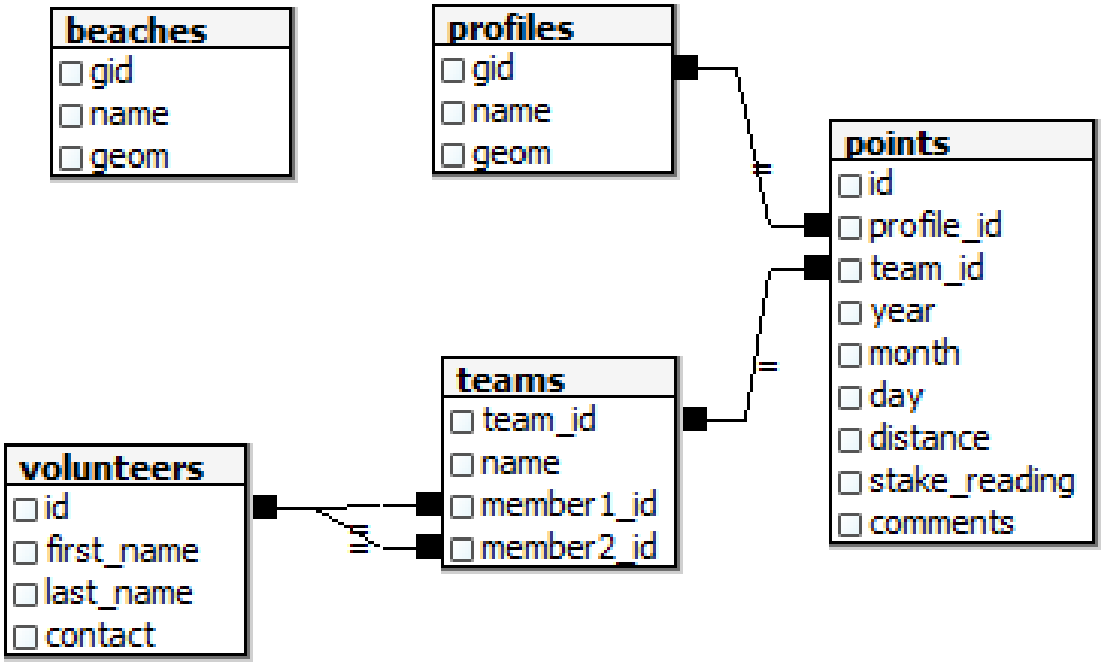

Figure 2: Entity-Relationship Diagram

Both the beaches and profiles tables were created by importing the respective shapefiles using the PostGIS shapefile loader. They were projected as "NAD_1983_UTM_Zone_19N", so a SRID of 26919 was set when loaded into the database. Each row of beaches is a polygon that represents a specific beach. The table is used to make spatial groupings of the profiles. The profiles are linestrings that represent the straight path from the dunes to the water. Profiles are important because it alleviates the need for volunteers to carry GPS units. The volunteers table contains a record for each individual volunteer. Likewise, the teams table contains a record for each team, which consists of two (2) volunteers. Finally, the points table contains the bulk of the volunteer data collection. It contains data on who collected it (team_id), where it was collected (profile_id and distance), when it was collected (year, month, and day), elevation readings (stake_reading), and anything extra the volunteers want recorded (comments). The data within the tables allows for both spatial and temporal queries.

Database Demonstration

The database makes it possible to do spatial queries. The following SQL will select the profiles contained with a specific beach, in this example Old Orchard Beach (OOB):

```
SELECT profiles.gid, profiles.name, beaches.name
FROM final_project.profiles, final_project.beaches
WHERE final_project.beaches.name = 'OOB' AND st_contains(final_project.beaches.geom,
final_project.profiles.geom);
```

The following SQL will select all points associated with a specific profile:

```
SELECT id, profile_id, team_id, distance, stake_reading
FROM final_project.points
WHERE profile_id = 1;
```

The following SQL will select all elevation readings within a specific distance band, which could be useful for seeing the elevation across all profiles at a single distance band:

```
SELECT year, month, day, profiles.name, team_id, distance, stake_reading
FROM final_project.points
JOIN final_project.profiles ON final_project.points.profile_id = final_project.profiles.gid
WHERE distance = 6;
```

The following SQL will select elevation readings within a specific distance band of a specific profile, which could be useful for visualizing how the locations elevation has changed over time:

```
SELECT year, month, day, profiles.name, team_id, distance, stake_reading
FROM final_project.points
JOIN final_project.profiles ON final_project.points.profile_id = final_project.profiles.gid
WHERE profile_id = 1 AND distance = 6;
```

The database also allows for temporal queries. The following SQL will select all elevation readings associated with a specific profile for one year and show which team took the readings, which could be useful for creating profile graphs that show month to month changes:

```
SELECT profiles.name, teams.name, year, month, day, distance, stake_reading
FROM final_project.points
JOIN final_project.profiles ON final_project.points.profile_id = final_project.profiles.gid
JOIN final_project.teams ON final_project.points.team_id = final_project.teams.team_id
WHERE profile_id = 1 AND year = 2012;
```

The data outputted by the above query can easily be exported into a *.csv file and graphed in Excel. As an example, the above query produced Figure 3 below.

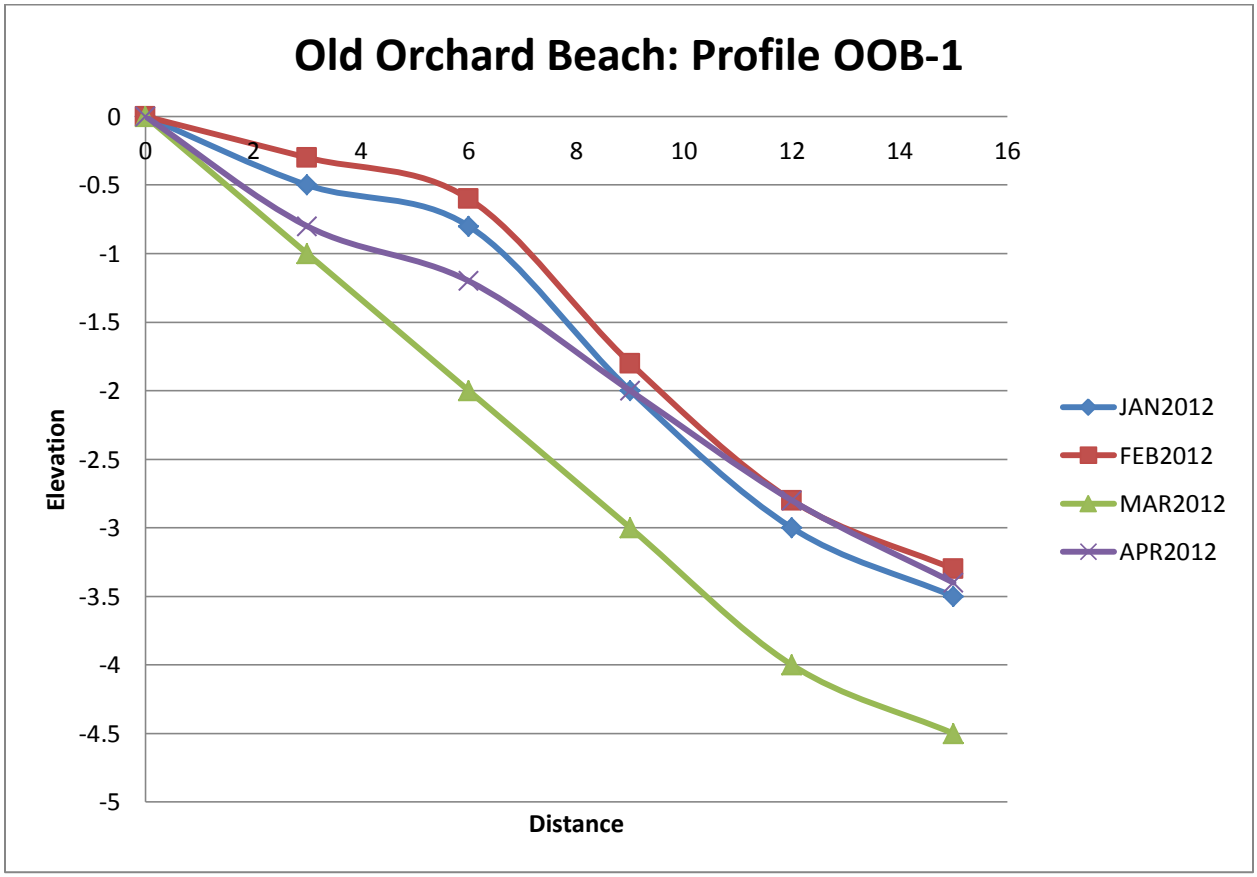

Figure 3: JAN-APR 2012 Beach Erosion/Accretion for Profile OOB-1

Summary

The database was designed to allow easy input form volunteers, but it was also designed to allow scientists and geographers to easily manipulate the data. The scatterplot outputs are a quick and easy way to visualize the erosion and accretion trends on individual profiles or entire beaches. The database represents just the back-end, but if the database were to be placed on a server or in the cloud, it could be made web accessible. Ultimately, it would be nice to implement data visualizations using D3.js. The database in its current state eliminates much of the paperwork and opens up a web accessible option.