**Documentation of the Database overall:**

At the moment, data on endemic Hawaiian species is being kept on spreadsheets but the Biology Department anticipates that as more information is collected the current system will hurt more than help. The database will be cataloging the different endemic Hawaiian species collected as well as when, where and by whom. The database will allow the general public across the world that is interested in endemic Hawaiian species to view information regarding where certain species are found as well as giving the biologists a way to store and research their information efficiently.

The entities/relationships that we added to the EER model were the location\_details, specimen\_location\_code, and upload\_column\_names. The entity upload\_column\_names is used to facilitate the entry of information through the bulkload update. Upload\_column\_names stores the collector’s abbreviation and the name they use as a header for their xml (the format the bulkload accepts) document as primary keys with attribute\_name representing the actual database attribute name and name\_type specifying the taxonomy name. We needed to link a specimen to a location detail because originally the location detail was in the location table which was then linked to the event table. The problem with this is that many specimens were collected at one event and each specimen needed to be linked to their own location detail. Thus, we split off the location detail from the location table and created a relationship between specimen and location detail called specimen\_location\_code.

**Please find the Entity Relationship Model on Attachment 1.**

**Please find the Database Schema on Attachment 2.**

**The create script in order to generate the database:**

CREATE TABLE locality (

locality\_name VARCHAR(100) NOT NULL,

PRIMARY KEY(locality\_name)

);

CREATE TABLE sex (

sex\_code CHAR(6) NOT NULL,

sex\_type VARCHAR(255) NULL,

PRIMARY KEY(sex\_code)

);

CREATE TABLE preservation\_status (

status\_code CHAR(4) NOT NULL,

status\_description VARCHAR(255) NULL,

PRIMARY KEY(status\_code)

);

CREATE TABLE organization (

abbr4\_organization CHAR(4) NOT NULL,

organization\_name VARCHAR(50) NOT NULL,

organization\_street VARCHAR(50) NULL,

organization\_city VARCHAR(50) NULL,

organization\_state VARCHAR(50) NULL,

organization\_zip VARCHAR(10) NULL,

organization\_note VARCHAR(255) NULL,

organization\_country VARCHAR(50) NULL,

PRIMARY KEY(abbr4\_organization)

);

CREATE TABLE media\_type (

file\_format CHAR(4) NOT NULL,

media\_type\_note VARCHAR(255) NULL,

PRIMARY KEY(file\_format)

);

CREATE TABLE meta (

meta\_item CHAR(10) NOT NULL,

PRIMARY KEY(meta\_item)

);

CREATE TABLE location\_details (

location\_detail\_code INTEGER UNSIGNED NOT NULL AUTO\_INCREMENT,

north\_degree DECIMAL(12.7) NOT NULL,

west\_degree DECIMAL(12.7) NOT NULL,

elevation\_foot DECIMAL NULL,

waypoint VARCHAR(25) NULL,

utm\_zone INTEGER UNSIGNED NULL,

utm\_band CHAR(1) NULL,

utm\_nothing INTEGER(7) UNSIGNED NULL,

utm\_easting INTEGER(7) UNSIGNED NOT NULL,

PRIMARY KEY(location\_detail\_code)

)

TYPE=InnoDB;

CREATE TABLE reserve\_permission (

permission\_type CHAR(20) NOT NULL,

PRIMARY KEY(permission\_type)

);

CREATE TABLE life (

short\_scientific\_name CHAR(9) NOT NULL,

scientific\_name VARCHAR(50) NOT NULL,

PRIMARY KEY(short\_scientific\_name),

FULLTEXT INDEX life\_FullText1(scientific\_name),

UNIQUE INDEX life\_Unique1(scientific\_name)

);

CREATE TABLE taxonomy (

rank\_name CHAR(15) NOT NULL,

rank\_name\_2 CHAR(15) NOT NULL,

mandatory ENUM('yes','no') NOT NULL,

PRIMARY KEY(rank\_name),

INDEX taxonomy\_FKIndex1(rank\_name\_2),

FOREIGN KEY(rank\_name\_2)

REFERENCES taxonomy(rank\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE rank\_type (

rank\_type CHAR(25) NOT NULL,

PRIMARY KEY(rank\_type),

FULLTEXT INDEX rank\_type\_FullText1(rank\_type)

);

CREATE TABLE island (

island\_name CHAR(10) NOT NULL,

PRIMARY KEY(island\_name)

);

CREATE TABLE container (

cabinet\_code CHAR(5) NOT NULL,

box\_code CHAR(9) NOT NULL,

section\_code CHAR(9) NOT NULL,

abbr4\_organization CHAR(4) NOT NULL,

PRIMARY KEY(cabinet\_code, box\_code, section\_code, abbr4\_organization),

INDEX container\_FKIndex1(abbr4\_organization),

FOREIGN KEY(abbr4\_organization)

REFERENCES organization(abbr4\_organization)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE collector (

abbr3\_collector CHAR(3) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

middle\_name VARCHAR(50) NULL,

email VARCHAR(50) NOT NULL,

passwd TEXT NULL,

abbr4\_organization CHAR(4) NULL,

user\_name VARCHAR(8) NULL,

PRIMARY KEY(abbr3\_collector),

INDEX collector\_FKIndex1(abbr4\_organization),

FOREIGN KEY(abbr4\_organization)

REFERENCES organization(abbr4\_organization)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE upload\_column\_names (

abbr3\_collector CHAR(3) NOT NULL AUTO\_INCREMENT,

xml\_name VARCHAR NOT NULL,

attribute\_name VARCHAR NULL,

name\_type ENUM('taxon','column') NULL,

PRIMARY KEY(abbr3\_collector, xml\_name),

INDEX upload\_column\_names\_FKIndex1(abbr3\_collector),

FOREIGN KEY(abbr3\_collector)

REFERENCES collector(abbr3\_collector)

ON DELETE NO ACTION

ON UPDATE NO ACTION

)

TYPE=InnoDB;

CREATE TABLE reserve (

reserve\_name CHAR(20) NOT NULL,

permission\_type CHAR(20) NULL,

PRIMARY KEY(reserve\_name),

INDEX reserve\_FKIndex1(permission\_type),

FOREIGN KEY(permission\_type)

REFERENCES reserve\_permission(permission\_type)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE dna (

extraction\_code CHAR(9) NOT NULL,

section\_code CHAR(9) NOT NULL,

box\_code CHAR(9) NOT NULL,

cabinet\_code CHAR(5) NOT NULL,

abbr4\_organization CHAR(4) NOT NULL,

sequence TEXT NOT NULL,

sequence\_name VARCHAR(50) NOT NULL,

primer\_name VARCHAR(50) NOT NULL,

PRIMARY KEY(extraction\_code),

FULLTEXT INDEX dna\_FullText1(sequence),

INDEX dna\_FKIndex1(abbr4\_organization, cabinet\_code, box\_code, section\_code),

FOREIGN KEY(abbr4\_organization, cabinet\_code, box\_code, section\_code)

REFERENCES container(abbr4\_organization, cabinet\_code, box\_code, section\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE media (

media\_id INTEGER UNSIGNED NOT NULL AUTO\_INCREMENT,

file\_format CHAR(4) NOT NULL,

directory\_path VARCHAR(125) NOT NULL,

file\_name VARCHAR(125) NOT NULL,

media\_note VARCHAR(255) NULL,

PRIMARY KEY(media\_id),

INDEX media\_FKIndex1(file\_format),

INDEX media\_Unique1(directory\_path, file\_name, file\_format),

FOREIGN KEY(file\_format)

REFERENCES media\_type(file\_format)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE taxonomy\_rank (

rank\_name CHAR(15) NOT NULL,

rank\_type CHAR(25) NOT NULL,

PRIMARY KEY(rank\_name, rank\_type),

INDEX taxonomy\_rank\_FKIndex1(rank\_type),

INDEX taxonomy\_rank\_FKIndex2(rank\_name),

FOREIGN KEY(rank\_name)

REFERENCES taxonomy(rank\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(rank\_type)

REFERENCES rank\_type(rank\_type)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE dna\_media (

extraction\_code CHAR(9) NOT NULL,

media\_id INTEGER UNSIGNED NOT NULL,

PRIMARY KEY(extraction\_code, media\_id),

INDEX dna\_media\_FKIndex1(extraction\_code),

INDEX dna\_media\_FKIndex2(media\_id),

FOREIGN KEY(extraction\_code)

REFERENCES dna(extraction\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(media\_id)

REFERENCES media(media\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE life\_media (

short\_scientific\_name CHAR(9) NOT NULL,

media\_id INTEGER UNSIGNED NOT NULL,

PRIMARY KEY(short\_scientific\_name, media\_id),

INDEX life\_media\_FKIndex1(short\_scientific\_name),

INDEX life\_media\_FKIndex2(media\_id),

FOREIGN KEY(short\_scientific\_name)

REFERENCES life(short\_scientific\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(media\_id)

REFERENCES media(media\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE media\_meta (

meta\_item CHAR(10) NOT NULL,

media\_id INTEGER UNSIGNED NOT NULL,

meta\_numeric\_value INTEGER UNSIGNED NOT NULL,

PRIMARY KEY(meta\_item, media\_id),

INDEX media\_meta\_FKIndex1(meta\_item),

INDEX media\_meta\_FKIndex2(media\_id),

FOREIGN KEY(media\_id)

REFERENCES media(media\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(meta\_item)

REFERENCES meta(meta\_item)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE life\_taxonomy (

rank\_type CHAR(25) NOT NULL,

rank\_name CHAR(15) NOT NULL,

short\_scientific\_name CHAR(9) NOT NULL,

PRIMARY KEY(rank\_type, rank\_name, short\_scientific\_name),

INDEX life\_taxonomy\_FKIndex2(rank\_type, rank\_name),

INDEX life\_taxonomy\_FKIndex1(short\_scientific\_name),

FOREIGN KEY(rank\_type, rank\_name)

REFERENCES taxonomy\_rank(rank\_type, rank\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(short\_scientific\_name)

REFERENCES life(short\_scientific\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE location (

island\_name CHAR(10) NOT NULL,

reserve\_name CHAR(20) NOT NULL,

locality\_name VARCHAR(100) NOT NULL,

location\_detail\_code INTEGER UNSIGNED NOT NULL,

PRIMARY KEY(island\_name, reserve\_name, locality\_name),

*INDEX Location\_FKIndex1(island\_name),*

*INDEX Location\_FKIndex2(reserve\_name),*

*INDEX Location\_FKIndex3(*locality\_name),

INDEX Location\_FKIndex4(location\_detail\_code),

FOREIGN KEY(island\_name)

REFERENCES island(island\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(reserve\_name)

REFERENCES reserve(reserve\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(locality\_name)

REFERENCES locality(locality\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(location\_detail\_code)

REFERENCES location\_details(location\_detail\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE event (

event\_code CHAR(9) NOT NULL,

abbr3\_collector CHAR(3) NOT NULL,

locality\_name VARCHAR(100) NOT NULL,

reserve\_name CHAR(20) NOT NULL,

island\_name CHAR(10) NOT NULL,

date DATE NOT NULL,

event\_note TEXT NULL,

PRIMARY KEY(event\_code),

INDEX event\_FKIndex1(island\_name, reserve\_name, locality\_name),

INDEX event\_FKIndex2(abbr3\_collector),

FOREIGN KEY(island\_name, reserve\_name, locality\_name)

REFERENCES location(island\_name, reserve\_name, locality\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(abbr3\_collector)

REFERENCES collector(abbr3\_collector)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE event\_collector (

event\_code CHAR(9) NOT NULL,

abbr3\_collector CHAR(3) NOT NULL,

PRIMARY KEY(event\_code, abbr3\_collector),

INDEX event\_collector\_FKIndex1(event\_code),

INDEX event\_collector\_FKIndex2(abbr3\_collector),

FOREIGN KEY(event\_code)

REFERENCES event(event\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(abbr3\_collector)

REFERENCES collector(abbr3\_collector)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE collection (

catalog\_code CHAR(9) NOT NULL,

event\_code CHAR(9) NOT NULL,

short\_scientific\_name CHAR(9) NOT NULL,

number\_sample INTEGER UNSIGNED NULL,

number\_male INTEGER UNSIGNED NULL,

number\_female INTEGER UNSIGNED NULL,

collection\_note TEXT NULL,

PRIMARY KEY(catalog\_code, event\_code),

INDEX specimen\_FKIndex1(event\_code),

INDEX collection\_FKIndex3(short\_scientific\_name),

FOREIGN KEY(event\_code)

REFERENCES event(event\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(short\_scientific\_name)

REFERENCES life(short\_scientific\_name)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE specimen (

specimen\_code CHAR(12) NOT NULL,

event\_code CHAR(9) NOT NULL,

catalog\_code CHAR(9) NOT NULL,

cabinet\_code CHAR(5) NOT NULL,

box\_code CHAR(9) NOT NULL,

section\_code CHAR(9) NOT NULL,

abbr4\_organization CHAR(4) NOT NULL,

sex\_code CHAR(6) NULL,

status\_code CHAR(4) NULL,

PRIMARY KEY(specimen\_code),

INDEX specimen\_FKIndex1(abbr4\_organization, cabinet\_code, box\_code, section\_code),

INDEX specimen\_FKIndex2(status\_code),

INDEX specimen\_FKIndex3(sex\_code),

INDEX specimen\_FKIndex4(catalog\_code, event\_code),

FOREIGN KEY(abbr4\_organization, cabinet\_code, box\_code, section\_code)

REFERENCES container(abbr4\_organization, cabinet\_code, box\_code, section\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(status\_code)

REFERENCES preservation\_status(status\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(sex\_code)

REFERENCES sex(sex\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(catalog\_code, event\_code)

REFERENCES collection(catalog\_code, event\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE specimen\_media (

media\_id INTEGER UNSIGNED NOT NULL,

specimen\_code CHAR(12) NOT NULL,

PRIMARY KEY(media\_id, specimen\_code),

INDEX specimen\_media\_FKIndex1(specimen\_code),

INDEX specimen\_media\_FKIndex2(media\_id),

FOREIGN KEY(specimen\_code)

REFERENCES specimen(specimen\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(media\_id)

REFERENCES media(media\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE specimen\_dna (

extraction\_code CHAR(9) NOT NULL,

specimen\_code CHAR(12) NOT NULL,

PRIMARY KEY(extraction\_code, specimen\_code),

INDEX specimen\_dna\_FKIndex1(specimen\_code),

INDEX specimen\_dna\_FKIndex2(extraction\_code),

FOREIGN KEY(specimen\_code)

REFERENCES specimen(specimen\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(extraction\_code)

REFERENCES dna(extraction\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE specimen\_location\_code (

specimen\_code CHAR(12) NOT NULL,

location\_detail\_code INTEGER UNSIGNED NOT NULL,

PRIMARY KEY(specimen\_code),

INDEX specimen\_location\_code\_FKIndex1(specimen\_code),

INDEX specimen\_location\_code\_FKIndex2(location\_detail\_code),

FOREIGN KEY(specimen\_code)

REFERENCES specimen(specimen\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(location\_detail\_code)

REFERENCES location\_details(location\_detail\_code)

ON DELETE NO ACTION

ON UPDATE NO ACTION

)

TYPE=InnoDB;

**Normalization:**

We have evaluated our database and have found that our tables are in Third Normal Form. This implies that everything in the tables are dependent on their respective primary keys and only their primary keys. Also, the attributes of the tables are dependent on the whole primary key. We did not normalize the tables any further in order to preserve the working database model of the Biology Department’s data.

**Data Dictionary part a:**

upload\_column\_names:

Stores the collector’s abbreviation and the name they use as a header for their xml (the format the bulkload accepts) document as primary keys with attribute\_name representing the actual database attribute name and name\_type specifying the taxonomy name.

location\_detail:

Stores the specific coordinates of locations of where specimens have been collected.

specimen\_location\_code :

Stores the link from each specimen to a specific location\_detail which is where they were collected.

**Please find Data Definition part b in Attachment 3.**

The SQL query used to generate Data Definition part b is as follows:

SELECT TABLE\_NAME, COLUMN\_NAME, DATA\_TYPE FROM information\_schema.COLUMNS where TABLE\_SCHEMA = 'TCBES';

**Queries: (Please find the results attached in Attachment 4)**

**Query 1:** SELECT xml\_name, attribute\_name, name\_type

FROM upload\_column\_names

WHERE abbr3\_collector = "$\_SESSION['uid'] ";

This query allows us to pull down information for the current user as to how their known naming scheme of columns corresponds to the database attributes. Query gives no results as upload\_column\_names is currently empty.

**Query 2:** SELECT box\_code, section\_code, cabinet\_code, catalog\_code

FROM specimen

WHERE specimen\_code = ‘8’;

Finds where specimen 8 is being stored.

**Query 3:** select REFERENCED\_TABLE\_NAME, REFERENCED\_COLUMN\_NAME

from information\_schema.KEY\_COLUMN\_USAGE

where TABLE\_NAME = '%s' and COLUMN\_NAME = '%s' and REFERENCED\_TABLE\_NAME is not null;

Used to query the information\_schema to find dependancies. Results assume that TABLE\_NAME = ‘specimen’ and COLUMN\_NAME = ‘event\_code’.

**Query 4:** select ‘yes’ as collection

from collection

where catalog\_code = %s and event\_code = %s

This query checks to see if there is something in the collection table that has a certain catalog code and event code combination. If there is the query returns yes otherwise it will be blank. This query is used to peruse the database for existing data with the same key values. Results assume that catalog\_code = 01 and event\_code = m0001.

**Query 5:** select collection.event\_code, collection.catalog\_code, collector.first\_name,

collector.last\_name, collector.middle\_name, collector.abbr3\_collector, life.scientific\_name, life.scientific\_name, event.island\_name, event.reserve\_name, event.locality\_name, event.date,

round (location\_details.north\_degree,4) as north\_degree, round(location\_details.west\_degree,4) as west\_degree,

specimen.cabinet\_code, specimen.box\_code, specimen.section\_code,

specimen.abbr4\_organization

from collection, event, location, collector, life, specimen, location\_details

where collection.event\_code = event.event\_code and

location.location\_detail\_code = location\_details.location\_detail\_code and

event.island\_name = location.island\_name and

event.reserve\_name = location.reserve\_name and

event.locality\_name = location.locality\_name and

event.abbr3\_collector = collector.abbr3\_collector and

collection.short\_scientific\_name = life.short\_scientific\_name and

collection.event\_code = specimen.event\_code and

collection.catalog\_code = specimen.catalog\_code

order by collection.event\_code;

**Query 6:** INSERT INTO specimen(specimen\_code,event\_code, catalog\_code, cabinet\_code, box\_code, section\_code, abbr4\_organization, sex\_code, status\_code)

VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s);

Inserts into specimen a new tuple defined by the variables (used in bulkload). Results assume the tuple is (‘8’, ‘m0300’, ‘02’, ‘01’,’16’, ‘2’, ‘UHH’, ‘f’, NULL). The results state that foreign key constraints stop the update.

**Query 7:** SELECT specimen\_code, north\_degree, west\_degree

FROM specimen\_location\_code, location\_details

WHERE specimen\_code = 1 AND specimen\_location\_code.location\_detail\_code = location\_details.location\_detail\_code;

Finds out the north degree and west degree of specimen 1.

The results return 0 results due to specimen\_location\_code being empty.

**Query 8:** SELECT first\_name, last\_name, organization\_name

FROM collector, organization

WHERE collector.abbr4\_organization = organization. abbr4\_organization

ORDER BY organization\_name;

Lists the first and last name of every collector as well as their corresponding organization ordered by organization name.

**Query 9:** SELECT first\_name, last\_name, event\_code

FROM event, collector

WHERE date = “2009-05-20” AND event.abbr3\_collector = collector. abbr3\_collector;

Lists the first and last name of every collector and event code present at any event on 5/20/2009.

**Query 10:** SELECT specimen\_code, sex\_type

FROM specimen, sex

WHERE sex.sex\_code = specimen. sex\_code;

Lists all the specimen’s specimen codes and their respective sex types.

**Query 11:** SELECT specimen\_code, sequence

FROM specimen\_dna, dna

WHERE specimen\_dna.extraction\_code = dna. extraction\_code;

Lists all the specimen’s specimen codes and their respective dna sequences.

This returns no results since specimen\_dna and dna are empty.

**Query 12:** SELECT first\_name, last\_name

FROM collector

WHERE abbr4\_organization = ‘UHH’;

Lists the first name and last name of all the collector’s from the organization UHH.

**Query 13:** DESCRIBE $variable

Assuming $variable is the name of a table this query is used to describe a table in order to determine what elements also need to be added when we insert data.

Results supposing $variable = event.

**Query 14:** SELECT extraction\_code

FROM dna, specimen

WHERE dna.section\_code = specimen.section\_code and specimen\_code = ‘8’;

The result is the extraction code for all dna that share a section\_code with specimen 8. The result returns 0 rows.

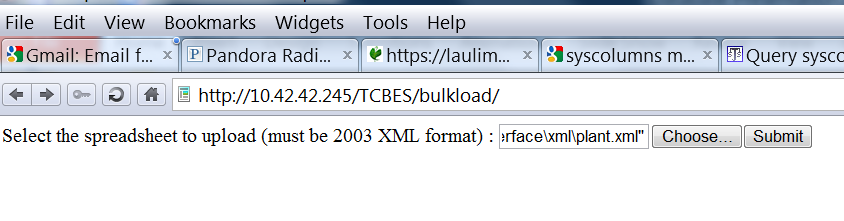
**Indexing:**

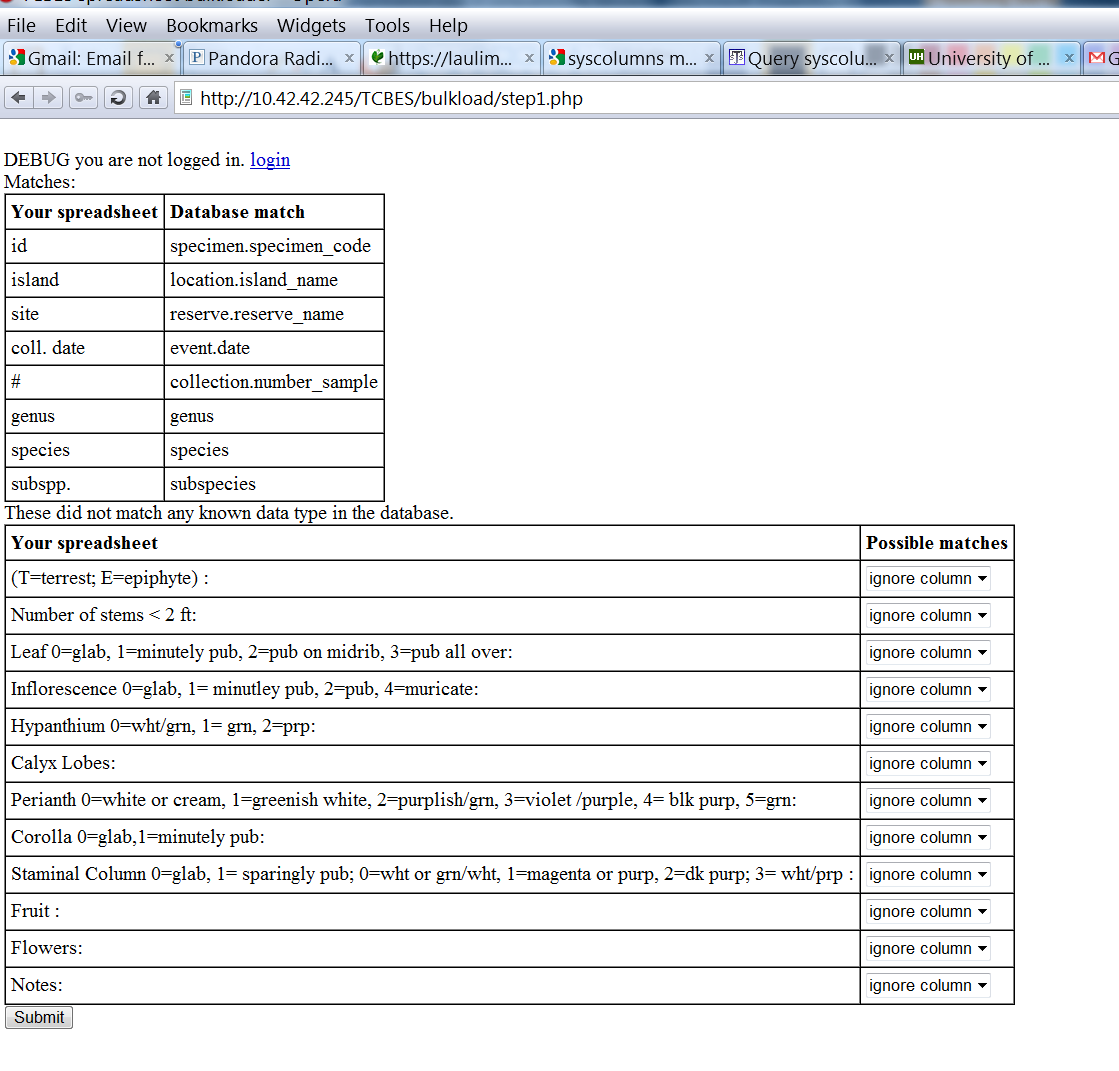
CREATE INDEX longitude ON location\_details (north\_degree);

CREATE INDEX latitude ON location\_details (west\_degree);

This is because when most people would like to search the location\_details table they will refer to the north\_degree and west\_degree.

**Forms**:

We have a form that the user will submit their file to be uploaded and another form to check whether or not we missed any matching columns respectively. 



**Reports**:

We have a report that shows the user what columns in their xml file matched the database and another in \_readDev.php that reports information on different events and the specimens collected respectively. 