Caribherp Database

Sarah Hanson

Aim

To build a spatially-enabled object relational database to store spatial and non-spatial data about reptiles and amphibians of the Caribbean to support scientific research for Temple's Center for Biodiversity.

Data Types & Sources

The data sources include a mix of shapefiles and CSV tables. These data include:

- Polygon shapefiles which represent the ranges of ~1000 Caribbean reptiles and amphibians
  – created by Hedge's Lab using ArcGIS Desktop 10.4 in accordance with IUCN Red List Range Mapping Standards
  – attribute tables store important information about each population including presence and origin (i.e. extinct vs. extant & native vs. introduced)

- Point data (Museum specimens and/or observation points)
  – Multiple tables that each contain point data. Some are personal collections, others are large public databases, and yet others are points that have been digitized from paper maps. All records with lat, long coordinates will be pushed into a table in the database called observation_points. Records that were digitzed from scanned maps will be coded with a number ‘2’ in the accuracy column to indicate that the accuracy is low.
  – point shapefiles of observations digitized from scanned maps in various literature sources
- Caribbean Basemap
  – Modified version of GADM’s World Basemap
  – Generated new attribute table to include island name, smaller island group, larger island group or archipelago name, and country (i.e. Martinique Island, Martinque Bank, Lesser Antilles, France) to support analysis at different scales
  – Digitized ~50 small islands using satellite imagery basemap from ESRI for islands that were not represented by the basemap but had species records from them
- Trait data - measurements about museum specimens (i.e. snout width, forearm length, etc.)
- Integrated Taxonomic Information System (ITIS) Species List
  – database with reliable, but not always up-to-date information on species names and their hierarchical classification
  – Because our center does a lot of work in taxonomy, discovering and reclassifying species, their species list was not able to be used as a primary key in our species table, but as an attribute instead in case anyone should need to link our species

to other tables that ITIS provides

Entity Relationship Diagram | Data Model

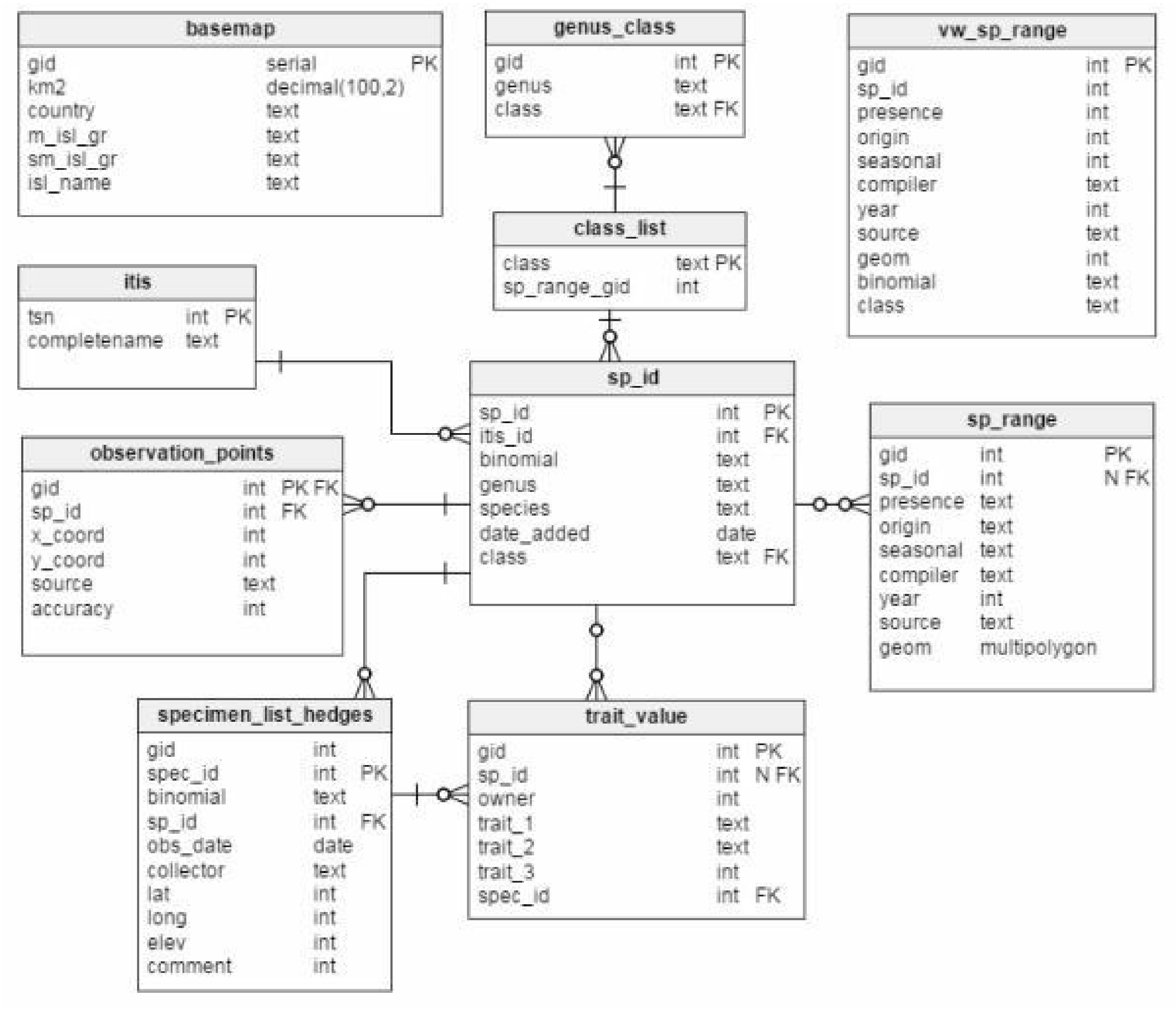

Figure 1. Entity Relationship Diagram (ERD) created using Vertabelo.

Table Explanations

The above ERD does not show the staging tables for clarity. When importing new trait data and range data, we use staging tables to clean the data and assign their respective sp_id values before pushing them into their final resting place (trait_data and sp_range).

sp_id

The most central table to the database is the sp_list table, which stores each species name along with a serial id number, called sp_id, which is the primary key. Also in this table is the corresponding ITIS id number, if applicable, the date it was added to the database, and the class to which it belongs (i.e. amphibia or reptilia) which is a foreign key from the class table. The class is determined by matching the first part of the species name, called the genus, to the genus_class table which has a list of genera with their corresponding class (class is a foreign key to a small table called class to make sure there aren't spelling errors). For this reason, we parsed the species name field into two columns called genus and epithet. This table (genus_class) was created manually to start, and moving forward will probably need to be updated manually, adding new genera and assigning their appropriate classes as well. The sp_id is used as a foreign key in the range table (sp_range), observation table (specimen_list), and the trait table (trait_data).

itis

longnames table from the ITIS database. Includes just their species name and their id number that they assign to it. This id number is stored as a foreign key in the sp_list table.

class_list

Includes just one column which list the classes of life. Used as a FK in the genus_class table and sp_id table to reinforce referential integrity.

genus_class

List of genera and their respective classes. Class is a FK from the class_list table. This table was created manually from all of the genera in the sp_range table currently in the database and will likely need to be updated manually to add new genera to the table. This table is used to determine the class stored in the sp_id table. In the future, we may drop this table and have database users manually add the class name to the sp_id table, that or upload a much larger table in its place from another database like NCBI Taxonomy.

staging_table

Table used to clean the range data as it is imported into the database. Once edited, only a subset of the columns get pushed into the sp_range table. In particular, we add a sp_id column, determine the sp_id by matching the binomial column (species name) to the species column in the sp_id table, add a class column, parse the binomial column into species and epithet, and then determine the class by matching the genus name from the staging table to the genus column in the genus_class table. Once cleaned, all ranges with a sp_id will be pushed into the sp_range table and then dropped from the staging_table. The remaining entries will need to be edited to have a correct, matching species name (binomial), or if it is determined that the ranges are of a new species to the database, they will need to be added to the sp_id table using a SELECT INTO statement.

sp_range

This table stores the ranges of all species. It includes the geometry of a range, the sp_id of the species range it represents, attributes about that range like whether the species was introduced or native to that area, whether or not it is still present or thought to be extirpated, who created the range, and the source of the range information.

specimen_list_hedges

This is a table of information about specimens of Dr. Blair Hedges. These specimens include specimen_id value, which is the primary key. The attributes stored in this table include things like date collected, location collected (x, y), collector name, among other details about when and where it was found and the condition in which it was found in. In this table, spec_id is the primary key because each individual animal should only be entered once and must all be unique.

trait_data

This table stores measurements about particular specimens (a specific lizard). The measurements can be linked to a species by joining the specimen_id number to the specimen_list_hedges table which stores the sp_id. In this table, we use a serial primary key, called gid, because many times a specimen is measured many times over by different people. Important attributes in this table include the trait measurements (like snout width, body length, etc.) as well as who measured it and on what date.

observation_points

This table is a table stores data about locations where species were observed, including attributes like sp_id, lat/long coordinates, elevation, collector name, specimen_id, as well as source. Any record from the specimen_list_hedges table that has a lat, long coordinate is stored in this table, along with other collection data with coordinates including public record information from GBIF, as well as point data that was digitized from scanned maps in Schwartz & Henderson (1991) and Rodríguez Schettino (2013). This table will not be able to be edited by users, because it is actually a compilation of all of the individual tables of point data (not currently listed - forthcoming).

vw_sp_range

This is a table was created by saving the output of a join between the sp_range table and the sp_id table. It is a copy of the sp_range table with the binomial and class values added from the sp_id table so that a user can query a single range table using the species name, rather than id, and class. While this de-normalized our database, it is very useful for non-SQL experts. This table also will not be able to be edited by any user to preserve the integrity of the database.

basemap

This table is a basemap of the world, which was generated from a polygon shapefile. While the geometry is accurate at the global scale, it only has detailed information about the identification of the land area, such as island name, island group, country etc. for the Caribbean. Outside of the Caribbean, the only information stored about the land feature, in addition to it’s geometry, is the country name.

Data Visualization

This database can easily be queried to show ranges of species by many features including their location in space, by joining the range table top the basemap using a spatial intersection, or by the species’ attributes, like their name, class, origin, or presence. To see an example of one data visualization possibility, see below.

```
FROM vw_sp_range AS r
JOIN basemap AS b
ON ST_Intersects (r.geom, b.geom)
WHERE b.sm_isl_gr = 'Hispaniola Banormal
nk' AND r.class ILIKE 'amphibia' AND r.presence = 1 AND r
.origin != 3;
```

Figure 2. SQL Query on vw_sp_range table to yield all extant, native amphibians from the island of Hispaniola.

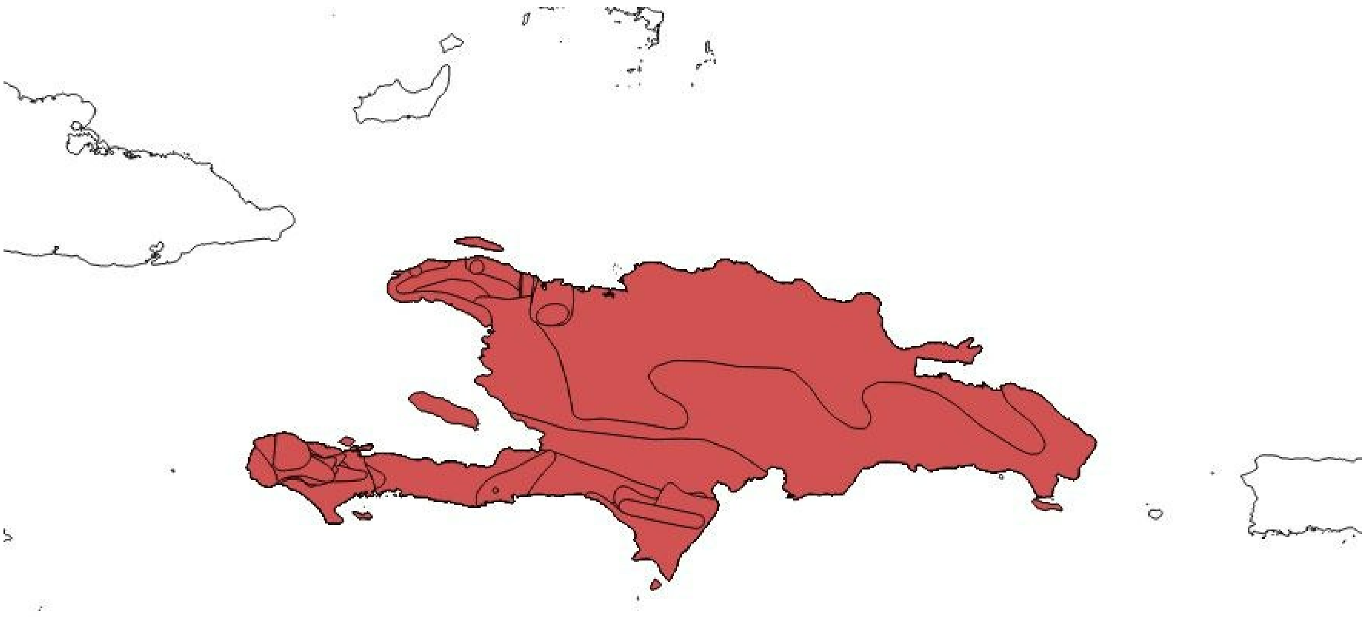

Figure 3. Layer imported to QGIS using the DB Manager and query shown above.

Conclusion

In short, this database enables trait data about reptiles and amphibians to be analyzed spatially using their respective range maps which is a significant achievement. In addition, both ranges and traits can be analyzed using countless other spatial datasets, like environmental data on climate variables like temperature, rainfall, and land use. In the future, this database will serve as the back-end to a program which researchers and students will use to enter new measurements about specimens into the trait_data table of the database.

Appendix

First, existing datasets (spatial and non-spatial) were imported into the Caribherp database using the DB Manager in QGIS. These datasets included:

– shapefile containing of all Caribbean range maps (table called shapefile_upload)

– basemap (polygon shapefile)

– observation_points (point shapefile)

– specimen_list_hedges and trait_data (CSV tables)

– SET search_path = caribherp2,public;

Push data from shapefile_upload (ranges) into a staging table for range data

```
CREATE TABLE staging_table
(
  id serial PRIMARY KEY,
  sp_id int,
  geom geometry(MultiPolygon,4326),
  id_no int,
  binomial text,
  presence integer,
  origin integer,
  compiler text,
  year integer,
  citation text,
  source text,
  dist_comm text,
  island text,
  subspecies text,
  subpop text,
  tax_commen text,
  data_sens text,
  sens_comm text,
  legend text,
  seasonal integer,
  class text,
  genus text
)
;
```

```
INSERT INTO staging_table (
geom
, id_no
, binomial
, presence
, origin
, compiler
, year
, citation
, source
, dist_comm
, island
, subspecies
, subpop
, tax_commen
, data_sens
, sens_comm
, legend
, seasonal
)
SELECT
geom
, id_no :: integer
, binomial
, presence
, origin
, compiler
, year
, citation
, source
, dist_comm
, island
, subspecies
, subpop
, tax_commen
, data_sens
, sens_comm
, legend
, seasonal
FROM shapefile_upload;
```

Cleaning

```
VACUUM ANALYZE staging_table;
```

Create class_list table

```
CREATE TABLE class_list (
class text PRIMARY KEY
);
```

Cleaning

```
VACUUM ANALYZE class_list;
```

```
VACUUM ANALYZE basemap;
```

Adding class values to class table

```
INSERT INTO class (class_list)
VALUES ('amphibia');
```

```
INSERT INTO class (class_list)
VALUES ('reptilia');
```

```
INSERT INTO class (class_list)
VALUES ('mammalia');
```

```
INSERT INTO class (class_list)
```

```
VALUES ('aves');
```

```
INSERT INTO class (class_list)
VALUES ('agnatha');
```

```
INSERT INTO class (class_list)
VALUES ('chondrichthyes');
```

```
INSERT INTO class (class_list)
VALUES ('osteichthyes');
```

Create sp_id table

```
DROP TABLE IF EXISTS sp_id;
CREATE TABLE sp_id AS
    SELECT DISTINCT lower(binomial) AS binomial
    FROM shapefile_upload;
```

Add columns to sp_id table

```
ALTER TABLE sp_id
ADD COLUMN sp_id serial PRIMARY KEY;
```

```
ALTER TABLE sp_id
ADD COLUMN genus text;
```

```
ALTER TABLE sp_id
```

```
ADD COLUMN epithet text;
```

```
ALTER TABLE sp_id
ADD COLUMN class text;
```

Check for duplicates

```
SELECT *
FROM sp_id
WHERE binomial IN (SELECT binomial
              FROM (SELECT binomial,
                             ROW_NUMBER() OVER (partition
 BY binomial ORDER BY sp_id) AS rnum
                     FROM sp_id) t
              WHERE t.rnum > 1);
```

Adding constraints to binomial field in sp_id

```
ALTER TABLE sp_id
ALTER COLUMN binomial SET NOT NULL;
```

```
ALTER TABLE sp_id
ADD CONSTRAINT constraint_name UNIQUE (binomial);
```

Partitioning binomial field to genus and epithet

```
UPDATE sp_id
    SET genus = split_part(binomial, ' ', 1),
    epithet = split_part(binomial, ' ', 2);
```

Add ITIS ID to the master sp_list where applicable

```
UPDATE sp_id
    SET itisid = itis.longnames.tsn
    FROM itis.longnames
WHERE lower(itis.longnames.completename) = sp_id.binomial
;
```

Add sp_id value to the range data staging table

```
UPDATE staging_table
    SET sp_id = sp_id.sp_id
    FROM sp_id
WHERE lower(staging_table.binomial) = sp_id.binomial;
```

Check for null values

```
SELECT *
FROM staging_table
WHERE sp_id IS NULL;
```

Add NOT NULL constraint to the sp_id column of the range data staging table

```
ALTER TABLE staging_table
ALTER COLUMN sp_id SET NOT NULL;
```

Cleaning

```
VACUUM ANALYZE sp_id;
```

Partition binomial field into two parts, copying the first part (genus) to a newly created column

```
ALTER TABLE shapefile_upload
ADD COLUMN genus text;
```

```
UPDATE shapefile_upload
    SET genus = split_part(binomial, ' ', 1);
```

Create table of genera names and their respective classes from the shapefile_upload table to be used in the future to determine the appropriate class when a range map is imported, as this information is not traditionally stored in the attribute table

```
DROP TABLE IF EXISTS genus_class;
CREATE TABLE genus_class AS
SELECT DISTINCT lower (genus) AS genus, class
FROM shapefile_upload;
```

Check for duplicates, just to be sure

```
SELECT *
FROM genus_class
WHERE genus IN (SELECT genus
              FROM (SELECT genus,
                             ROW_NUMBER() OVER (partition
 BY genus ORDER BY genus) AS rnum
                     FROM genus_class) t
              WHERE t.rnum > 1);
```

Cleaning

```
VACUUM ANALYZE genus_class;
```

Add class to sp_id table based on genus portion of the binomial name

```
UPDATE sp_id
    SET class = genus_class.class
    FROM genus_class
WHERE  lower(sp_id.genus) = lower(genus_class.genus);
```

Create final range table which all range data will be pushed into once cleaned

```
DROP TABLE IF EXISTS sp_range;
CREATE TABLE sp_range
(
  gid serial PRIMARY KEY,
  sp_id int,
  geom geometry(MultiPolygon,4326),
  presence integer,
  origin integer,
  compiler text,
  year integer,
  citation text,
  source text,
  subspecies text,
  legend text,
  seasonal integer
)
;
```

Insert entries with a sp_id value into sp_range from staging_table

```
INSERT INTO sp_range (
sp_id
, geom
, presence
, origin
, compiler
, year
, citation
, source
, subspecies
, legend
, seasonal
)
SELECT
sp_id
, geom
, presence
, origin
, compiler
, year
, citation
, source
, subspecies
, legend
, seasonal
FROM staging_table
WHERE sp_id NOT NULL;
```

Add NOT NULL constraint to sp_id column in sp_range table

```
ALTER TABLE sp_range
ALTER COLUMN sp_id SET NOT NULL;
```

Cleaning

```
VACUUM ANALYZE sp_range;
```

```
VACUUM ANALYZE trait_data;
```

```
VACUUM ANALYZE specimen_list_hedges;
```

Adding contraints to the genus_class table

```
ALTER TABLE genus_class
ALTER COLUMN genus SET NOT NULL;
ALTER TABLE genus_class
ALTER COLUMN class SET NOT NULL;
```

```
##### Add primary key to genus_class table
ALTER TABLE genus_class
ADD PRIMARY KEY (genus, class);
```

Add foreign key constraints to tables

```
ALTER TABLE trait_data ADD CONSTRAINT "spec_id"
    FOREIGN KEY ("spec_id") REFERENCES specimen_list_hedg
es ("spec_id");
```

```
ALTER TABLE genus_class ADD CONSTRAINT class
    FOREIGN KEY (class) REFERENCES class_list (class);
```

```
ALTER TABLE sp_id ADD CONSTRAINT itisid
    FOREIGN KEY (itisid) REFERENCES itis.longnames (tsn);
```

```
ALTER TABLE sp_id ADD CONSTRAINT class
    FOREIGN KEY (class) REFERENCES class_list (class);
```

```
ALTER TABLE sp_range ADD CONSTRAINT sp_id
    FOREIGN KEY (sp_id) REFERENCES sp_id (sp_id);
```

Join range data to sp_table to get class and create view

```
DROP VIEW IF EXISTS vw_sp_range;
CREATE VIEW vw_sp_range AS
SELECT a.*, b.class, b.binomial
FROM sp_range as a
JOIN sp_id as b
ON a.sp_id = b.sp_id;
```

Create table from view

```
CREATE TABLE view_sp_range
AS SELECT * FROM vw_sp_range;
```

```
ALTER TABLE view_sp_range
ADD PRIMARY KEY (gid);
```

Adding sp_id column to trait_data

```
ALTER TABLE trait_data
ADD COLUMN sp_id int;
```

Inserting appropriate sp_id value into the sp_id field of the trait data table

```
UPDATE trait_data
    SET sp_id = sp_id.sp_id
    FROM sp_id
WHERE lower(trait_data.species) = sp_id.binomial;
```

Adding accuracy indicator column to the observation_point table (specifically so one can exclude points digitized from a scanned map since they will be coded as 2 for this field)

```
ALTER TABLE observation_points
ADD COLUMN accuracy int;
```

THE END (for now)