Laboratory 12: intermediate MariaDB

This set of exercises is designed build on laboratory 11 by using the same data set (Obiang et al. 2019) stored in a more conventionally designed relational database. You will replace the monolithic table with six tables: (1) 'protocols' to record the sampling protocols, (2) 'families' to store plant family names, (3) 'genera' to store generic names and link them to families, (4) 'specificEpithets' to store specific epithets and authors, (5) 'scientificNames' to link specific epithets to genera, and (6) 'observations' to store unique data for each observation and link to protocols and scientific names. The design of the database will require variations on the JOIN syntax to be used for most queries.

Tasks

(1) Begin by logging into MariaDB as the root user by typing `sudo mariadb -u root mysql` in the terminal. Enter your password when prompted.

(2) Create a database for this laboratory exercise by typing `CREATE DATABASE lab12;` in the terminal.

(3) Provide your 'working' user with an additional privilege by typing `` GRANT CREATE TEMPORARY TABLES ON `lab12`.* TO `working`@`localhost`; `` in the terminal.

(4) To make the new privileges active, type `FLUSH PRIVILEGES;` in the terminal.

(5) Logout of the root user, by typing `EXIT;` in the terminal.

(6) Login using your 'working' user and the 'lab12' database by typing `mariadb -u working -p lab12` in the terminal.

(7) Create and populate the 'protocols' table:

   (a) Create the table structure by typing the following in the terminal:
```
CREATE TABLE `protocols` (
`protocolID` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
`plotShape` ENUM('circular', 'rectangular'),
`minimumDBHcm` TINYINT UNSIGNED NOT NULL,
PRIMARY KEY (`protocolID`),
UNIQUE KEY (`plotShape`, `minimumDBHcm`)
) ENGINE=InnoDB DEFAULT CHARSET=UTF8;
```
   (b) Insert the data by typing the following in the terminal:
```
INSERT INTO protocols (plotShape, minimumDBHcm) VALUES
('circular', 5),
('circular', 10),
('rectangular', 1),
('rectangular', 10);
```
   (c) Check your work by typing `SELECT * FROM protocols;` in the terminal.

(8) Create and populate the 'families' table:

   (a) Create the table structure by typing the following in the terminal:
```
CREATE TABLE `families` (
`familyID` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
`family` VARCHAR(32) NOT NULL UNIQUE,
PRIMARY KEY (`familyID`)
) ENGINE=InnoDB DEFAULT CHARSET=UTF8;
```

   (b) Populate the table using a query from the 'lab11' database by typing `INSERT INTO families (family) SELECT DISTINCT family FROM lab11.gabon ORDER BY family;` in the terminal.

   (c) Inspect your work by typing `SELECT * FROM families;` in the terminal.

(9) Create and populate the 'genera' table:

   (a) Create the table structure by typing the following in the terminal:

```
CREATE TABLE `genera` (
`genusID` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
`genus` VARCHAR(32) NOT NULL UNIQUE,
`familyID` TINYINT UNSIGNED NOT NULL,
PRIMARY KEY (`genusID`),
FOREIGN KEY (`familyID`) REFERENCES families (`familyID`) ON DELETE CASCADE ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=UTF8;
```

   (b) Populate the table using a query from the 'lab11' database by typing `INSERT INTO genera (familyID, genus) (SELECT DISTINCT familyID, genus FROM (SELECT family, SUBSTRING_INDEX(SUBSTRING_INDEX(scientificName, ' ', 1), ' ', -1) AS genus FROM lab11.gabon) AS taxa, families WHERE taxa.family = families.family ORDER BY genus) ON DUPLICATE KEY UPDATE genera.familyID = families.familyID;` in the terminal. Because some *Oncoba* species were placed in Achariaceae others in Salicaceae, the query explicitly updates all *Oncoba* to Salicaceae.

   (c) Check your work by typing `SELECT * FROM genera;` in the terminal. Answer question (1).

(10) Create and populate the 'specificEpithets' table:

   (a) Create the table structure by typing the following in the terminal:

```
CREATE TABLE `specificEpithets` (
`specificEpithetID` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
`specificEpithet` VARCHAR(64) NOT NULL UNIQUE,
PRIMARY KEY (`specificEpithetID`)
) ENGINE=InnoDB DEFAULT CHARSET=UTF8;
```

   (b) Add an entry for undetermined species by typing `INSERT INTO specificEpithets SET specificEpithet = 'indet.';` in the terminal.

   (c) Populate the table using a query from the 'lab11' database by typing `INSERT INTO specificEpithets (specificEpithet) SELECT DISTINCT specificEpithet FROM (SELECT TRIM(SUBSTR(scientificName, LOCATE(' ', scientificName))) AS specificEpithet FROM lab11.gabon) AS epithets WHERE LENGTH(specificEpithet) > 1 ORDER BY specificEpithet;` in the terminal.

   (d) Inspect your work by typing `SELECT * FROM specificEpithets;` in the terminal. Answer question (2).

(11) Create and populate the 'scientificNames' table:

(a) Create the table structure by typing the following in the terminal:
```
CREATE TABLE `scientificNames` (
`scientificNameID` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
`genusID` SMALLINT UNSIGNED NOT NULL,
`specificEpithetID` SMALLINT UNSIGNED NOT NULL,
PRIMARY KEY (`scientificNameID`),
FOREIGN KEY (`genusID`) REFERENCES genera (`genusID`) ON DELETE CASCADE ON UPDATE
CASCADE,
FOREIGN KEY (`specificEpithetID`) REFERENCES specificEpithets (`specificEpithetID`)
ON DELETE CASCADE ON UPDATE CASCADE,
UNIQUE KEY (`genusID`, `specificEpithetID`)
) ENGINE=InnoDB DEFAULT CHARSET=UTF8;
```
(b) Populate the table using a query from the 'lab11' database by typing `INSERT INTO scientificNames (genusID, specificEpithetID) SELECT DISTINCT genusID, specificEpithetID FROM (SELECT genus, IF(LENGTH(specificEpithet) > 1, specificEpithet, 'indet.') AS specificEpithet FROM (SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(scientificName, ' ', 1), ' ', -1) AS genus, TRIM(SUBSTR(scientificName, LOCATE(' ', scientificName))) AS specificEpithet FROM lab11.gabon) AS renamedTaxa) AS taxa, genera, specificEpithets WHERE taxa.genus = genera.genus AND taxa.specificEpithet = specificEpithets.specificEpithet;` in the terminal.

(c) Inspect your work by typing `SELECT * FROM scientificNames;` in the terminal.

(12) The taxonomic data are a bit messy and inconsistent with the database structure.

(a) For example, type `SELECT DISTINCT family, genus, specificEpithet FROM scientificNames, specificEpithets, genera, families WHERE scientificNames.specificEpithetID = specificEpithets.specificEpithetID AND scientificNames.genusID = genera.genusID AND genera.familyID = families.familyID AND genus = 'Carpolobia';` in the terminal. One row should be returned.

(b) To change 'G.Don' to 'indet.' type `UPDATE scientificNames SET specificEpithetID = (SELECT specificEpithetID FROM specificEpithets WHERE specificEpithet = 'indet.') WHERE specificEpithetID = (SELECT specificEpithetID FROM specificEpithets WHERE specificEpithet = 'G.Don');` in the terminal.

(c) Remove the orphaned 'G.Don' by typing `DELETE FROM specificEpithets WHERE specificEpithet = 'G.Don';` in the terminal.

(d) Rerun the query from step (12)(a) to check your work.

(e) To update the observations that have not been identified to genus type `UPDATE genera SET genus = CONCAT(genus, ' genus indet.') WHERE genus IN (SELECT DISTINCT genus FROM (SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(scientificName, ' ', 1), ' ', -1) AS genus, TRIM(SUBSTR(scientificName, LOCATE(' ', scientificName))) AS specificEpithet FROM lab11.gabon) AS taxa WHERE LENGTH(specificEpithet) < 1);` in the terminal.

(f) Type `SELECT DISTINCT family, genus, specificEpithet FROM scientificNames, specificEpithets, genera, families WHERE scientificNames.specificEpithetID = specificEpithets.specificEpithetID AND scientificNames.genusID = genera.genusID AND genera.familyID = families.familyID AND (family = 'Asteraceae' OR family = 'Malvaceae');` to check your work. Answer question (3).

(13) Create and populate the ‘observations’ table:

(a) Create the table structure by typing the following in the terminal:

```
CREATE TABLE `observations` (
`gbifID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
`occurrenceID` VARCHAR(16) NOT NULL,
`individualCount` SMALLINT UNSIGNED NOT NULL,
`eventDate` DATE NOT NULL,
`protocolID` TINYINT UNSIGNED NOT NULL,
`location` POINT,
`scientificNameID` SMALLINT UNSIGNED NOT NULL,
PRIMARY KEY (`gbifID`),
FOREIGN KEY (`protocolID`) REFERENCES protocols (`protocolID`) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (`scientificNameID`) REFERENCES scientificNames (`scientificNameID`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=UTF8;
```

(b) To simplify the query that will populate the ‘observations’ table, create a temporary (in volatile memory) names table:

(1) Begin by typing `CREATE TEMPORARY TABLE names SELECT DISTINCT scientificNameID, CONCAT(genus, ' ', specificEpithet) AS scientificName FROM scientificNames, specificEpithets, genera, families WHERE scientificNames.specificEpithetID = specificEpithets.specificEpithetID AND scientificNames.genusID = genera.genusID AND genera.familyID = families.familyID;` in the terminal.

(2) Modify the *temporary* data to match the ‘lab11’ version by typing `UPDATE names SET scientificName = 'Carpolobia G.Don' WHERE scientificName = 'Carpolobia indet.';` in the terminal.

(3) Further modify the *temporary* data to match the ‘lab11’ version by typing `UPDATE names SET scientificName = 'Asteraceae' WHERE scientificName LIKE 'Asteraceae%'; UPDATE names SET scientificName = 'Malvaceae' WHERE scientificName LIKE 'Malvaceae%';` in the terminal.

(4) Inspect your work by typing `SELECT * FROM names;` in the terminal.

(c) A temporary ‘samplingProtocol’ table will also simplify the query that will populate the ‘observations’ table.

(1) Create and populate the table by typing `CREATE TEMPORARY TABLE samplings SELECT DISTINCT 4 AS protocolID, samplingProtocol FROM lab11.gabon;` in the terminal.

(2) Fix the ‘protocolID’ field for the ForestGEO plot by typing `UPDATE samplings SET protocolID = 3 WHERE samplingProtocol LIKE '%ForestGEO%';` in the terminal.

(3) Fix the ‘protocolID’ field for circular plots by typing `UPDATE samplings SET protocolID = 2 WHERE samplingProtocol LIKE '%circular%' AND samplingProtocol LIKE '%5 cm'; UPDATE samplings SET protocolID = 1 WHERE samplingProtocol LIKE '%circular%' AND samplingProtocol LIKE '%10 cm';` in the terminal.

(4) Type `SELECT * FROM samplings;` to inspect your work.

(d) Populate the table by typing `INSERT INTO observations SELECT gbifID, occurrenceID, individualCount, eventDate, protocolID, PointFromText(CONCAT('POINT(', decimalLatitude, ' ', decimalLongitude, ')')) AS location, scientificNameID FROM lab11.gabon, names,` `samplings WHERE lab11.gabon.scientificName = names.scientificName AND lab11.gabon.samplingProtocol = samplings.samplingProtocol;` in the terminal. This query may take a while to run.

(e) Inspect your work by typing `SELECT gbifID, occurrenceID, individualCount, eventDate, plotShape, minimumDBHcm, AsText(location) AS location, family, genus, specificEpithet FROM observations, protocols, scientificNames, specificEpithets, genera, families WHERE observations.protocolID = protocols.protocolID AND observations.scientificNameID = scientificNames.scientificNameID AND scientificNames.specificEpithetID = specificEpithets.specificEpithetID AND scientificNames.genusID = genera.genusID AND genera.familyID = families.familyID;` in the terminal.

Questions (https://forms.gle/zTtYpaaf2oM5ZFsG8)

(1) For task (9)(c):

(a) How many rows did the query return?

(b) Identify at least two entries that are not valid generic names.

(2) For task (10)(d):

(a) How many rows did the query return?

(b) Identify an entry that is a not valid specific epithet.

(3) For task (12)(f), identify the two rows modified by your query.

Literature cited

Obiang, N. E., A. Ngomanda & D. Kenfack. 2019. Vegetation assessment and forest dynamic study of various areas in Gabon from 2000 to 2018. Herbier National du Gabon (https://doi.org/10.15468/i8fwlf).

Due at the start of class April 25.