Assignment 4 – CIS4301, Fall 2019

Alin Dobra

November 21, 2019

- **Due Date:** Wednesday December 4th, 2019 at 11:50pm
- **Submit via Canvas**

For this assignment, you will be asked to write and test three different programs in SQLite. Each of these will use the database maintained by the imaginary Southern Sierra Wildflower Club (SSWC), an organization whose members are interested in observing wildflowers in their native habitat in the southern part of the Sierra Nevada mountains of California.

The database maintained by the club has four tables:

SIGHTINGS (NAME, PERSON, LOCATION, SIGHTED)
FEATURES (LOCATION, CLASS, LATITUDE, LONGITUDE, MAP, ELEV)
FLOWERS (GENUS, SPECIES, COMNAME)
MEMBERS (NAME, MEMBERSINCE, NUMSIGHTINGS)

SIGHTINGS.PERSON →MEMBERS.NAME
SIGHTINGS.LOCATION →FEATURES.LOCATION
SIGHTINGS.NAME →FLOWERS.COMNAME

The database tables have the following semantics:

- `SIGHTINGS` gives information that describes every time that a member of the club observes one of the wildflowers described in the table. `FLOWERS.NAME` tells the name of the flower observed, `PERSON` describes who saw the flower, `LOCATION` tells the name of a nearby geographical feature where the flower was seen, and `SIGHTED` tells the day when the flower was seen.
- `FEATURES` lists the various locations where flowers have been observed. `LOCATION` is the name of the place, `CLASS` is the type of place (there are several types, such as Summit, Mine, Locale, etc.), `LATITUDE` and `LONGITUDE` describe where on the surface of the earth the locations are found. `MAP` tells the name of the topographic map where the feature can be found and `ELEV` tells the height of the feature.
- `FLOWERS` lists all of the flowers that the members of the `SSWC` try to find. GENUS and SPECIES give the scientific name for the flower, and `COMNAME` gives the non-scientific name.
- `MEMBERS` lists all of the members of the `SSWC`. `NAME` is the member's first name. `MEMBERSINCE` is the date the member joined the club. `NUMSIGHTINGS` is the number of sightings the member has had.

The database with the schema and initial data is attached to this assignment.

The assignment is broken into three separate tasks. The tasks are ordered according to increasing difficulty. For each task, you need to turn in an electronic copy of your code, plus a copy of the tests that you are asked to run, which includes the copied-and-pasted results of the tests.

Finally, please make sure that every time that you run the set of tests associated with a task, you use a "fresh" copy of the database. In other words, re-load the database file between tasks.

You can find a description of the specific trigger language used by `SQLite` here:

- `https://www.w3resource.com/sqlite/sqlite-triggers.php`,
- `http://www.sqlitetutorial.net/sqlite-trigger/`

Task 1

For this task, you need to write a trigger that catches inserts into the `SIGHTINGS` table, and lets the user know when there is a potential foreign key problem. Specifically, the trigger should print an error (hint: use `RAISE`) message if the newly-inserted tuple references a person that does not currently exist in the database.

Error: Insert into the SIGHTINGS table references a person that is not found in the database.

Once you have implemented your trigger, test it by trying the following inserts:

```
INSERT INTO SIGHTINGS VALUES
  ("Douglas dustymaiden", "Jennifer", "Double Mountain", "2008-11-28");

INSERT INTO SIGHTINGS VALUES
  ("Douglas dustymaiden", "Brad", "Shirley Peak", "2007-08-18");

INSERT INTO SIGHTINGS VALUES
  ("Douglas dustymaiden", "Donna", "Grouse Meadow", "2011-11-28");

INSERT INTO SIGHTINGS VALUES
  ("Douglas dustymaiden", "Maria", "Grouse Meadow", "2014-08-16");

INSERT INTO SIGHTINGS VALUES
  ("Douglas dustymaiden", "Joe", "Piute Peak", "2007-02-17");
```

Then run the following query:

```
SELECT *
FROM SIGHTINGS
WHERE PERSON = "Jennifer"
OR PERSON = "Brad"
OR PERSON = "Donna"
OR PERSON = "Maria"
OR PERSON = "Joe";

SELECT *
FROM MEMBERS
WHERE MEMBERSINCE IS NULL;
```

Task 2

Repeat the work in Task 1 but this time, if a tuple references a name not found in the `MEMBERS` table a new tuple is inserted. The new tuple should have the person's name with `MEMBERSINCE` set to NULL and `NUMSIGHTINGS` set to 1.

Re-execute the diagnostic queries in Task 1.

Task 3

Sometimes, when people try to insert into the `SIGHTINGS` table, they accidentally insert the Latin name of the flower instead of its common name. For example, they might use:

```
INSERT INTO SIGHTINGS VALUES
("Chaenactis douglasii", "Person A", "Shirley Peak", "2006-08-18");
```

Instead of:

```
INSERT INTO SIGHTINGS VALUES
("Douglas dustymaiden", "Person A", "Shirley Peak", "2006-08-18");
```

For this task, you need to write a trigger that catches inserts into the `SIGHTINGS` table. If there is not a foreign key problem with the new flower then the flower is inserted normally. Otherwise, if there is a foreign key problem, and the problem is because someone tried to enter in the Latin name rather than the common name, a warning message is printed to the screen and the correct common name is inserted into the database.

So, for example, if someone tries to use:

```
INSERT INTO SIGHTINGS VALUES
 ("Chaenactis douglasii", "Person A", "Shirley Peak", "2006-08-18");
```

Then the modified tuple is inserted:

```
"Douglas dustymaiden", "Person A", "Shirley Peak", "2006-08-18".
```

Disregard other foreign key errors.

Then, test your trigger as follows.

Run the following five insertions with your trigger active.

```
INSERT INTO SIGHTINGS VALUES
  ("Sky pilot", "Person X", "Grouse Meadow", "2006-08-18");

INSERT INTO SIGHTINGS VALUES
  ("Hoar buckwheat", "Person X", "Grouse Meadow", "2006-08-18");

INSERT INTO SIGHTINGS VALUES
  ("Zigadenus venenosus", "Person X", "Grouse Meadow", "2006-08-18");

INSERT INTO SIGHTINGS VALUES
  ("Carex limosa", "Person Y", "Grouse Meadow", "2006-08-18");

INSERT INTO SIGHTINGS VALUES
  ("Draperia", "Person Z", "Grouse Meadow", "2006-08-18");
```

Then, in order to verify that the database was updated correctly, run the following queries:

```
SELECT *
FROM SIGHTINGS
WHERE PERSON = "Person X"
OR PERSON = "Person Y"
OR PERSON = "Person Z";
```

A Word of Caution

Start early! It may take many hours to get all 3 tasks to work.