CSC 455: Database Processing for Large-Scale Analytics

Assignment 2

Akhil Kumar,Ramasagaram(1488438)

Part 1

You are given a following schema in 1NF:
(License Number, Renewed, Status, Status Date, Driver Type, License Type, Original Issue Date, Name, Sex, Chauffer City, Chauffer State, Record Number) and the following functional dependencies:

Chauffeur City → Chauffeur State (both of these are a single column, not two columns)
Record Number → License Number, Renewed, Status, Status Date, Driver Type, License Type, Original Issue Date, Name, Sex, Chauffer City, Chauffer State

The table is based on a real data set original taken from City of Chicago data portal (located here: https://data.cityofchicago.org/Community-Economic-Development/Public-Chauffeurs/97wa-y6ff) However, the data has been cleaned and reduced to approximately one thousand rows. We will revisit a non-clean version of that data later.

Decompose the schema to make sure it is in Third Normal Form (3NF).

Write SQL DDL to create the 3NF tables you created. Remember to declare primary and foreign keys as necessary in your SQL code.

Ans:

The above schema can be decomposed into 3NF as following.

Record Number → License Number, Renewed, Status, Status Date, Driver Type, License Type, Original Issue Date, Name, Sex, Chauffer City.

Chauffeur City → Chauffeur State

Let's name the first table as "Information" which stores all the values data regarding record number, license number, name, sex and other details. The second stores the chauffeur city and state details. Below is the relational schema.

INFORMATION (Record_Number, License_Number, Renewed, Status, Status_Date, Driver_Type, License_Type, Original_Issue_Date, Name, Sex, Chauffer_City)
LOCATION (chauffer_city, chauffer_state)

The chauffer_city attribute in the Location table is a foreign key referencing from the chauffer_city in the Information table.

CODE:

```
CREATE TABLE INFORMATION(
Record_Number VARCHAR2(15) NOT NULL,
License_Number NUMBER(15),
Renewed DATE,
Status VARCHAR2(15),
Status_Date DATE,
Driver_Type VARCHAR2(20),
License_Type VARCHAR2(20),
Original_Issue_Date DATE,
Name VARCHAR(20),
Sex CHAR(7),
Chauffer_City CHAR(20),

CONSTRAINT Rec_NumPK
  PRIMARY KEY(Record_Number)
);

CREATE TABLE LOCATION(
Chauffer_City VARCHAR2(30),
Chauffer_State VARCHAR2(15),

CONSTRAINT City_FK
  FOREIGN KEY(Chauffer_City)
        REFERENCES INFORMATION(Chauffer_City)
);
```

Part 2

Write a python script that is going to create your tables from Part 1 in SQLite and populate them with data automatically. The data file is posted the Dropbox folder on D2L and is also available at this link: http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/Public_Chauffeurs_Short.csv
Use sqlite3 database as shown in class but remember to make data type changes to your tables from Part 1 (i.e., NUMBER(5,0)→INTEGER, NUMBER(5,2)→REAL). SQLite is very forgiving regarding data types, but most databases are not.

I have created some sample code that connects to a SQLite database, loads comma-separated student data and prints the contents of the loaded table. You can find it here:
http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/loadStudentData.py
Naturally you would have to populate however many tables you have created in Part 1, not just 1 table.

For this assignment only, if you run into primary key conflict when loading data

(i.e., “sqlite3.IntegrityError: column ID is not unique” error), you may use INSERT OR IGNORE instead of INSERT when loading data. This will cause INSERT to skip over duplicate inserts without causing an error.

Remember to load NULLs properly (i.e. not as string) and make sure you do not load the very first line that contains column names.

Ans:

I have decomposed the given data set according to my schema in the part1. So I have now two tables one called Information & Location. Below is the code

CODE:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Oct 07 23:09:43 2014

@author: Akhilkumar
"""
import csv
import sqlite3

# Creating a table called Information.
Information_Table = '''CREATE TABLE Information(
License_Number NUMBER(15),
Renewed DATE,
Status VARCHAR2(15),
Status_Date DATE,
Driver_Type VARCHAR2(20),
License_Type VARCHAR2(20),
Original_Issue_Date DATE,
Name VARCHAR(20),
Sex CHAR(7),
Chauffer_City CHAR(20),
Record_Number VARCHAR2(15) NOT NULL,

CONSTRAINT Rec_NumPK
PRIMARY KEY(Record_Number)
);'''

# Creating a table called Location.
Location_Table = '''CREATE TABLE Location(
Chauffer_City VARCHAR2(30),
Chauffer_State VARCHAR2(15),

CONSTRAINT City_FK
```

```
FOREIGN KEY(Chauffer_City)
REFERENCES INFORMATION(Chauffer_City)
);'"


# To open a connection to database
conn = sqlite3.connect("Assignment2.db")

#Request a cursor from the database
cursor = conn.cursor()

# Incase the tables is already present, then drop the tables.
cursor.execute("DROP TABLE IF EXISTS Information")
cursor.execute("DROP TABLE IF EXISTS Location")
# If there is no table with such name, then create Assignment2.db
cursor.execute(Information_Table)
cursor.execute(Location_Table)

# Let's open a file to populate the Information table in python
with open('Public_Chauffeurs_Short.csv', mode='r') as infile:
reader = csv.reader(infile)
with open('Public_Chauffeurs_Short1.csv', mode='w') as outfile:
writer = csv.writer(outfile)
mydict =
{(rows[0],rows[1],rows[2],rows[3],rows[4],rows[5],rows[6],rows[7],rows[8],rows[9],rows[10]) for rows
in reader}
cursor.executemany('''INSERT OR IGNORE INTO Information
(License_Number,Renewed,Status,Status_Date,Driver_Type,
License_Type,Original_Issue_Date,Name,Sex,Chauffer_City,Record_Number) VALUES
(?,?,?,?,?,?,?,?,?,?,?);''',mydict)

# Let's open a file to populate the Location table in python
with open('City.csv', mode='r') as infile1:
reader1 = csv.reader(infile1)
with open('City1.csv', mode='w') as outfile1:
writer1 = csv.writer(outfile1)
mydict1 = {(rows[0],rows[1]) for rows in reader1}
cursor.executemany('''INSERT OR IGNORE INTO Location (Chauffer_City,Chauffer_State) VALUES
(?,?);''',mydict1)

conn.commit()
conn.close()
```

Part 3

You were hired to do some data analysis for a local zoo. Below is the data table, including the necessary constraints and all the insert statements to populate the database.

```
-- Drop all the tables to clean up
DROP TABLE Animal;

-- ACategory: Animal category 'common', 'rare', 'exotic'.  May be NULL
-- TimeToFeed: Time it takes to feed the animal (hours)
CREATE TABLE Animal
(
  AID       NUMBER(3, 0),
  AName     VARCHAR2(30) NOT NULL,
  ACategory VARCHAR2(18),

  TimeToFeed NUMBER(4,2),

  CONSTRAINT Animal_PK
    PRIMARY KEY(AID)
);


INSERT INTO Animal VALUES(1, 'Galapagos Penguin', 'exotic', 0.5);
INSERT INTO Animal VALUES(2, 'Emperor Penguin', 'rare', 0.75);
INSERT INTO Animal VALUES(3, 'Sri Lankan sloth bear', 'exotic', 2.5);
INSERT INTO Animal VALUES(4, 'Grizzly bear', NULL, 2.5);
INSERT INTO Animal VALUES(5, 'Giant Panda bear', 'exotic', 1.5);
INSERT INTO Animal VALUES(6, 'Florida black bear', 'rare', 1.75);
INSERT INTO Animal VALUES(7, 'Siberian tiger', 'rare', 3.75);
INSERT INTO Animal VALUES(8, 'Bengal tiger', 'common', 2.75);
INSERT INTO Animal VALUES(9, 'South China tiger', 'exotic', 2.25);
INSERT INTO Animal VALUES(10, 'Alpaca', 'common', 0.25);
INSERT INTO Animal VALUES(11, 'Llama', NULL, 3.5);
```

Since none of the managers in the zoo know SQL, it is up to you to write the queries to answer the following list of questions.

1. Find all the animals (their names) that take less than 1.5 hours to feed.

Ans:

```
SELECT AName
FROM Animal
WHERE TimeToFeed < 1.5;
```

2. Find all the rare animals and sort the query output by feeding time (any direction)

Ans:

```
SELECT AName, TimeToFeed
FROM Animal
WHERE ACategory = 'rare'
```

```
ORDER BY TimeToFeed;
```

3. Find the animal names and categories for animals related to a bear (hint: use the LIKE operator)

Ans:

```
SELECT AName, ACategory
FROM Animal
WHERE AName LIKE '%bear';
```

4. Return the listings for all animals whose rarity is not specified in the database

Ans:

```
SELECT AName
FROM Animal
WHERE ACATEGORY IS NULL;
```

5. Find the rarity rating of all animals that require between 1 and 2.5 hours to be fed

Ans:

```
SELECT AName
FROM ANIMAL
WHERE ACategory = 'rare' AND TimeToFeed BETWEEN 1 and 2.5;
```

6. Find the names of the animals that are related to the tiger and are not common

Ans:

```
SELECT AName
FROM Animal
WHERE ACategory != 'common' AND AName LIKE '%tiger';
```

7. Find the minimum and maximum feeding time amongst all the animals in the zoo (single query)

Ans:

```
SELECT MIN(TimeToFeed),MAX(TimeToFeed) FROM Animal;
```

8. Find the average feeding time for all the rare animals

Ans:

```
SELECT AVG(TimeToFeed)
FROM Animal
WHERE ACATEGORY = 'rare';
```

9. Find listings for animals with ID less than 10 and also require more than 2 hours to feed.

Ans:

```
SELECT Aname
FROM Animal
WHERE aid < 10 AND TimeToFeed > 2;
```