CS2102 Database Systems

Semester 1 2019/2020

Tutorial 10 (*Selected Answers*)

Introduction

This tutorial looks at the logical aspect of SQL. As a language that is rooted on relation (similar to Prolog), it has certain capabilities that can be mapped to Prolog albeit at a weaker strength due to absence of the general *unification* and *resolution* algorithm. However, we can still solve logical puzzle using SQL, in its simpler form.

While the latter part of the tutorial assumes the existence of a solution to an earlier part and thus, can be solved independently, you are advised to attempt all of them to better understand the capability of SQL to solve puzzle.

This puzzle is generated from https://logic.puzzlebaron.com/. It has the following background story.

> *The Winter Olympics have just wrapped up in Norway. Using only the clues that follow, determine the number of gold and silver medals won by each country.*

There are four participating countries:

- Argentina
- Dominica
- Luxembourg
- Malawi

There are four different distribution of silver medals

- One country won 2 silver medals
- One country won 4 silver medals
- One country won 11 silver medals
- One country won 12 silver medals

There are four different distribution of gold medals

- One country won 1 gold medals
- One country won 2 gold medals
- One country won 3 gold medals
- One country won 4 gold medals

You are also given the following clues:

1. The team from Malawi finished with 3 gold medals.
2. The squad that won 2 gold medals was either the team from Malawi or the team that won 4 silver medals.
3. The squad from Dominica finished with 2 gold medals.
4. The four teams were the squad that won 1 gold medal, the squad from Dominica, the team from Luxembourg and the squad that won 2 silver medals.
5. The team that won 4 gold medals was either the squad that won 12 silver medals or the squad that won 4 silver medals.

Tutorial Questions

To solve the problem, we can start by creating 3 different tables (or 2, if you fix one of them). Our 3 tables are described as follows:

```
CREATE TABLE Countries (
  name   varchar(100),
  label  integer
);
```

```
CREATE TABLE Silver (
  total  integer,
  label  integer
);
```

```
CREATE TABLE Gold (
  total  integer,
  label  integer
);
```

For simplicity, we fix the label for gold. If a country won n gold medal, the label will be n. We can therefore, initialize our table with all possibilities as follows:

```
INSERT INTO Countries VALUES ('Argentina', 1);
INSERT INTO Countries VALUES ('Argentina', 2);
INSERT INTO Countries VALUES ('Argentina', 3);
INSERT INTO Countries VALUES ('Argentina', 4);
INSERT INTO Countries VALUES ('Dominica', 1);
INSERT INTO Countries VALUES ('Dominica', 2);
INSERT INTO Countries VALUES ('Dominica', 3);
INSERT INTO Countries VALUES ('Dominica', 4);
INSERT INTO Countries VALUES ('Luxembourg', 1);
INSERT INTO Countries VALUES ('Luxembourg', 2);
INSERT INTO Countries VALUES ('Luxembourg', 3);
INSERT INTO Countries VALUES ('Luxembourg', 4);
INSERT INTO Countries VALUES ('Malawi', 1);
INSERT INTO Countries VALUES ('Malawi', 2);
INSERT INTO Countries VALUES ('Malawi', 3);
INSERT INTO Countries VALUES ('Malawi', 4);

INSERT INTO Silver VALUES (2, 1);
INSERT INTO Silver VALUES (2, 2);
INSERT INTO Silver VALUES (2, 3);
INSERT INTO Silver VALUES (2, 4);
INSERT INTO Silver VALUES (4, 1);
INSERT INTO Silver VALUES (4, 2);
INSERT INTO Silver VALUES (4, 3);
INSERT INTO Silver VALUES (4, 4);
INSERT INTO Silver VALUES (11, 1);
INSERT INTO Silver VALUES (11, 2);
INSERT INTO Silver VALUES (11, 3);
INSERT INTO Silver VALUES (11, 4);
INSERT INTO Silver VALUES (12, 1);
INSERT INTO Silver VALUES (12, 2);
INSERT INTO Silver VALUES (12, 3);
INSERT INTO Silver VALUES (12, 4);
```

```
INSERT INTO Gold VALUES (1, 1);
INSERT INTO Gold VALUES (2, 2);
INSERT INTO Gold VALUES (3, 3);
INSERT INTO Gold VALUES (4, 4);
```

1. As with any logic puzzle, we assume that all countries have different label. Write the three `VIEW` below.
   a) Create a `VIEW` called `AllDiffCountries` that creates a permutation of labels for countries such that the labels between two countries are different. It will generate a tuple $(l_{arg}, l_{dom}, l_{lux}, l_{mal})$ where all the values of $l_{arg}, l_{dom}, l_{lux}, l_{mal}$ are different. We assume that $l_{arg}, l_{dom}, l_{lux}, l_{mal}$ are the label for Argentina, Dominica, Luxembourg, and Malawi respectively. Use the following `CREATE VIEW` template.
   ```
   CREATE VIEW AllDiffCountries (arg,dom,lux,mal) AS
   ```
   b) Create a `VIEW` called `AllDiffSilver` that creates a permutation of labels for silver medal count such that the labels between two silver medal count are different. It will generate a tuple $(l_{s2}, l_{s4}, l_{s11}, l_{s12})$ where all the values of $l_{s2}, l_{s4}, l_{s11}, l_{s12}$ are different. We assume that $l_{s2}, l_{s4}, l_{s11}, l_{s12}$ are the label for 2 silver medal count, 4 silver medal count, 11 silver medal count, and 12 silver medal count respectively. Use the following `CREATE VIEW` template.
   ```
   CREATE VIEW AllDiffSilver (s2,s4,s11,s12) AS
   ```
   c) Create a `VIEW` called `AllDiffGold` that creates a permutation of labels for gold medal count such that the labels between two gold medal count are different. It will generate a tuple $(l_{g1}, l_{g2}, l_{g3}, l_{g4})$ where all the values of $l_{g1}, l_{g2}, l_{g3}, l_{g4}$ are different. We assume that $l_{g1}, l_{g2}, l_{g3}, l_{g4}$ are the label for 1 gold medal count, 2 gold medal count, 3 gold medal count, and 4 gold medal count respectively. Use the following `CREATE VIEW` template.
   ```
   CREATE VIEW AllDiffGold (g1,g2,g3,g4) AS
   ```
   NOTE: This view is trivial as we fix the label to match exactly the gold medal won. This view can be excluded from your attempt or simply use `SELECT * FROM Gold;`.

CHECK: Check that each of your `VIEW` have 24 values inside with the exception of `AllDiffGold`.

2. Given the `VIEW` representing all different constraint in Question 1, we can now solve the question of *"How many gold did a country won"*. The idea here is to find a labeling that "makes sense" given the constraints given at the start. Put it another way, if we represent the constraints as a condition on `WHERE`-clause, it should match exactly one row on `AllDiffCountries`. There are 5 constraints listed above. To start, I will illustrate how you can convert the constraints above into `WHERE`-clause condition by converting the first constraint.
   ```
   CREATE VIEW SolCountries AS
     SELECT C.arg, C.dom, C.lux, C.mal
     FROM   AllDiffCountries C, AllDiffSilver S, AllDiffGold G
     WHERE  C.mal = G.g3 -- Constraint (1)
   ```
   a) Write the condition for Constraint (2)
   b) Write the condition for Constraint (3)
   c) Write the condition for Constraint (4)
   d) Write the condition for Constraint (5)

CHECK: Check that your `VIEW` have only 1 value inside. You should be able to answer the question *"How many gold did Luxembourg won"* and the answer should be 4.

3. Create the following VIEW to answer the *"How many silver medals did a country with n gold medal won"*.

```
CREATE VIEW SolSilver AS
  SELECT S.s2, S.s4, S.s11, S.s12
  FROM   AllDiffCountries C, AllDiffSilver S, AllDiffGold G
```

❖ **BONUS:** Which country won 11 silver medal?