Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Datenbanksysteme · Prof. Dr. Grust

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Datenbanksysteme I

WS 2019/20
Torsten Grust, Christian Duta

Assignment #10

Submission Deadline: January 14, 2020 - 10:00

Exercise 1: Cell Phone Tracking (12 Points)

These days, data preservation, privacy and location-based services are relevant topics. In this excercise you will get a rough idea of what can be easily done using data generated and collected by vour smart phone.

Please have a look at the data provided in `location_data.zip`.

The schema file contains two tables, `celltowers` and `cells`. The table `celltowers` is taken from http://opencellid.org/ and provides a database of so-called *cell towers* along with their GPS postition. Every single cell tower is uniquely identified by its `mcc`, `mnc`, `lac`, `cell_id` and the `net_type`. In table `cells`, a set of measurements collected on a mobile phone is stored. These records contain attributes to identify a particular cell tower as well as a timestamp indicating when the measurement has been taken. However, it is possible to track a phone's movements by combining the timestamp and the position of the *cell tower* that has been in range at this time. Note that a mobile phone can record **one** measurement at a time only! Please find the description of some of the attributes of the table below:

- `mcc`: The mobile country code describes roughly to which country a particular cell phone network belongs
- `mnc`: The mobile network code identifies a particular cell phone carrier
- `lac`: The location area describes a rough estimate of the current position of a mobile phone in the network of a particular cell phone carrier
- `cell_id`: A cell id identifies a *cell tower* uniquely in the network of a partiular cell phone carrier
- `lat`: The latitude of a *cell tower's* position
- `lon`: The longitude of a *cell tower's* position

Please construct the following queries:

1. Calculate the number of *cell towers* that are located within a 10 000m radius from point 48.521460, 9.053271 in the center of Tübingen. To calculate the distance between two points p_1, p_2 we use the *haversine* formula:

$d = 2r \arcsin\left(\sqrt{\sin^2(\frac{radians(\phi_2-\phi_1)}{2}) + \cos(radians(\phi_1))\cos(radians(\phi_2))\sin^2(\frac{radians(\lambda_2-\lambda_1)}{2})}\right)$, where

- d is the distance between p_1 and p_2
- r is the radius of the sphere (here: Earth's radius $= 6371000m$)
- ϕ_1, ϕ_2 is the latitude of p_1 and p_2 respectively
- λ_1, λ_2 is the longitude of p_1 and p_2 respectively
- `radians` converts degrees to radians

We provided you with the *haversine* function in form of a SQL function

```
haversine(lat_p1 float, lon_p1 float, lat_p2 float, lon_p2 float)
```

in `haversine.sql`.

2. For each day occurring in table `cells`, calculate what *cell tower* was used the most.
3. Formulate a query to calculate the approximate distance (in meters) covered per day. The distance can be determined by summing up the distances between the *cell towers* we have taken measurements for in ascending temporal order.

Notes:

- Do **not** use SQL window functions!
- As the queries might get complex in this excercise you should think about breaking the problem down into smaller, manageable chunks. It is a good idea to formulate queries using `WITH` here.
- If you are curious you can have a look at the points given in the database. Simply insert them in the format *latitude*, *longitude* into the google maps search bar.

Exercise 2: Entity Relationship Model (18 Points)

Consider the following description — provided by a health domain expert — for a database that stores information about hospitals. Please read the requirements **completely** before you start the design.

- A hospital has a unique name and an address.
- A ward in a hospital has a ward number (e.g. "A2") as well as a specialty designation (e.g. "delivery ward").
- Every hospital consists of one or multiple wards. Every ward belongs to exactly one hospital. It is uniquely identified by its ward number together with the name of the hospital.
- Each Hospital has employees which are uniquely identified by their Social Security Number (SSN). For each employee, given name, last name, date of birth and salary need to be stored.
- Each employee works at exactly one hospital.
- A ward has an arbitrary number of employees assigned, while an employee might be assigned to multiple wards, too.
- Hospitals also employ doctors which are uniquely identified by the number of their medical license. A doctor may be a specialist of a certain medical field (e.g. "gynecologist") and has an academic title.
- An employee might be a doctor. Every doctor is an employee.

- Patients treated in a hospital are identified by their health insurance number. A patient's personal information should be on record: given name and last name, date of birth and address.
- Every patient is treated by at least one doctor who is primarily responsible for him. Additionally (non-responsible) doctors may be consulted which treat the patient as well. A doctor may treat multiple patients at a time.

Model this scenario as a diagram of the Entity Relationship Model. Identify the entity and relationship types as well as their attributes. Underline key attributes and add cardinality constraints in $(\min, \max)$ notation. Document any additional assumptions you make.

Note: Two entity types may be connected by more than one relationship type.

Exercise 3: Key Derivation (10 Bonus Points)

This exercise is optional but, if completed, counts as bonus points towards your total.

Write a **Python3** function `key(k, u, fds)` to compute the set of candidate keys where parameters `k`, `u` denote sets of column names and `fds` denotes a set of functional dependencies. The pseudo-code formulation of the algorithm is given on slide 20 of slide set 10 "Functional Dependencies".

Implement sets using Python's `frozenset`[1] collection type.

Test your program on the following **test cases** and hand them in together with your program.

- The `instructions` example from the lecture (slide 19, slide set 10 "Functional Dependencies") with all functional dependencies.
- A schema $\mathrm{R}(A_1, \ldots, A_n, B_1, \ldots, B_n)$ with dependencies $A_i \to B_i, B_i \to A_i, i = 1, \ldots, n$. How large can you choose n to obtain the result in reasonable time?
- A schema $\mathrm{R}(A, B, C, D, E)$ with dependencies

$$\begin{aligned} AB &\to E \\ AD &\to B \\ B &\to C \\ C &\to D \end{aligned}$$

[1] https://docs.python.org/3/library/stdtypes.html#frozenset