Ex1_LuisZüttel_GionRubitschung_D1P

February 25, 2024

[ ]:
```
%load_ext sql
```

[ ]:
```
%sql sqlite://
```

1 Exercise 1

Sie möchten in einer Tabelle in einer Datenbank speichern, an welchem Datum Sie sich mit welcher Person an welchem Ort getroffen haben.

a) Geben Sie das Schema dieser Datenbank

[ ]:
```
%%sql

DROP TABLE IF EXISTS MEETINGS;
CREATE TABLE MEETINGS (
    NAME VARCHAR(100),
    DATE DATE
);
```

```
 * sqlite://
Done.
Done.
```

[ ]:
```
[]
```

b) Geben Sie ein Beispiel für eine Instanz dieses Schemas.

[ ]:
```
%%sql

INSERT INTO MEETINGS (NAME, DATE) VALUES ('Max Muster', '2024-01-01 15:00:00');
SELECT * FROM MEETINGS;
```

```
 * sqlite://
1 rows affected.
Done.
```

[ ]:
```
[('Max Muster', '2024-01-01 15:00:00')]
```

2 Exercise 2

a) Erweitern Sie das Beispiel aus der letzten Aufgabe: Sie wollen in der selben Tabelle auch noch die (aktuelle) Telefonnummer der Personen speichern.

[ ]:
```
%%sql

ALTER TABLE MEETINGS ADD COLUMN PHONE_NUMBER VARCHAR(100) NULL;
UPDATE MEETINGS SET PHONE_NUMBER = '1234567890' WHERE NAME = 'Max Muster';
```

```
 * sqlite://
Done.
1 rows affected.
```

[ ]:
```
[]
```

b) Erklären Sie anhand dieses Beispiels Redundanz in einer Datenbank

An sich können keine redundanten Treffen mit einer Person entstehen. Man kann die Person nicht zweimal am exakt gleichen Zeitpunk treffen. Trifft man die Person jedoch mehrmals wird der Name und die Telefonnummer mehrfach hinterlegt, was zu redundanten Einträgen führt. In unserer Datenbank haben wir Max Muster bisher einmal getroffen, wenn wir Max Muster nocheinmal treffen, ist Max Muster ein redundanter Eintrag.

[ ]:
```
%%sql

INSERT INTO MEETINGS (NAME, DATE, PHONE_NUMBER) VALUES ('Max Muster',␣
 ↪'2024-01-01 16:00:00', '1234567890');
SELECT * FROM MEETINGS;
```

```
 * sqlite://
1 rows affected.
Done.
```

[ ]:
```
[('Max Muster', '2024-01-01 15:00:00', '1234567890'),
 ('Max Muster', '2024-01-01 16:00:00', '1234567890')]
```

Wie wir sehen können ist Max Muster zweimal hinterlegt, was zu einer Redundanz führt.

c) Erklären Sie, wie aus dieser Redundanz Inkonsistenz entstehen kann.

Nehmen wir mal an, wir treffen einen anderen Max Muster der eine andere Telefonnummer hat. Mit zwei Max Mustern ist dies noch einigermassen übersichtlich. Ist es jedoch eine Firma die alle Meetings managed wird dies sehr schnell unübersichtlich.

[ ]:
```
%%sql

INSERT INTO MEETINGS (NAME, DATE, PHONE_NUMBER) VALUES ('Max Muster',␣
 ↪'2024-02-01 16:00:00', '9234567890');
SELECT * FROM MEETINGS;
```

```
 * sqlite://
1 rows affected.
Done.
```

[ ]:
```
[('Max Muster', '2024-01-01 15:00:00', '1234567890'),
 ('Max Muster', '2024-01-01 16:00:00', '1234567890'),
 ('Max Muster', '2024-02-01 16:00:00', '9234567890')]
```

3 Exercise 3

Gegeben seien die Tabellen instructor und department als CSV-Dateien. Implementieren Sie folgende Abfragen in einer Ihnen genehmen Programmiersprache oder Pseudocode:

a) Gegeben eine instructor ID, finde den Namen des instructors mit dieser ID.

[ ]:
```
import pandas as pd

given_id = 10101
instructor_data = pd.read_csv('files/instructor.csv', header=0)

instructor_data[instructor_data['ID'] == given_id]['name']
```

[ ]:
```
0    Srinivasan
Name: name, dtype: object
```

b) Gegeben einen Namen eines instructors, finde das Gebäude in dem er/sie arbeitet.

[ ]:
```
instructor_name = 'Einstein'

department_data = pd.read_csv('files/department.csv', header=0)

instructor_department = pd.merge(instructor_data, department_data,␣
 ↪on='dept_name')
instructor_department[instructor_department['name'] ==␣
 ↪instructor_name]['building']
```

[ ]:
```
2    Watson
Name: building, dtype: object
```

Vergleichsweise ist der Aufwand, um diese Abfragen zu machen, deutlich grösser. Wäre diese Tabellen in einer SQL Datenbank hinterlegt, könnte man die SQL Queries ausführen und hätte keine weitere Dependencies.

4 Exercise 4

Gegeben sei ein Array in der globalen Variable a mit Kontoständen in nichtflüchtigem Speicher. Es soll die Integritätsbedingung gelten, dass die Summe aller Werte des Arrays gleich bleibt. Nun soll das erste Konto geleert und auf das zweite Konto überwiesen werden. Dazu gibt es die Funktion:

```
void transact(int[] a)
    a[1] = a[1] + a[0];
    a[0] = 0;
```

Erklären Sie, wie bei einem Systemabsturz die Integritätsbedingung verletzt werden kann.

Wenn unter Umständen das System abstürzt, während die `transact`-Methode ausgeführt wird, kann es sein, dass die Summe nicht gleich bleibt. Das geschieht dadurch, dass es unterbrochen wird und nur der erste Schritt ausgeführt wird und der zweite Schritt nicht, welches die Integritätsbedingung verletzt.

Erweitern Sie die Funktion auf geeignete Weise, um sicherzustellen, dass die Integritätsbedingung zu jedem Zeitpunkt gilt.

```
void transact(int[] a)
    int[] new_values = a.clone();
    new_values[1] = new_values[1] + new_values[0];
    new_values[0] = 0;
    a = new_values.clone();
```