Technologies for Information Systems

Part I (10 points)

prof. L. Tanca – July 21st, 2017

Available time: 25 minutes

Last Name _____

First Name _____

Student ID _____ **Signature**_____

1) Define the concepts of valid time and transaction time in temporal databases also describing their advantages and disadvantages.

2) Describe flow, level, and unitary measures in the data warehouse context. Provide a set of examples.

- **During this part of the exam, students are not allowed to consult books or notes.**
- **Students should answer the theoretical questions using their own words, in order for the teachers to be able to assess their real level of understanding.**

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – July 21st, 2017

Available Time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliTwitter and ***UniTwitter*** are two social data providers: they collect Twitter data and store them in an internal database. Then, they grant access to this database to their customers. PoliTwitter collects only tweets coming from European users and stores them into a relational database, while UniTwitter collects only tweets coming from American users and stores them into a document-based NoSQL database.

UniPoliTwitter, on the contrary, is a company dealing with social data analysis, and has purchased the access to the data sources provided by PoliTwitter and UniTwitter. Since PoliTwitter and UniTwitter update regularly their data sources, in order to use both of them and at the same time keep them up to date UniPoliTwitter asks you to set up a virtual data integration system. In particular, UniPoliTwitter wants to build a system relying on a relational global schema.

The relational schema employed by **PoliTwitter** is as follows:

CITY (CityName, Country)
USER (Nickname, RealName, BirthDate*, CityName, ImageURL*, ProfileBackgroundColor, ProfileTextColor)
// Nicknames are unique worldwide. Examples: Nickname: '@Alice88', RealName: 'Alice Smith'; Nickname: '@BobBrown', RealName: 'Bob Brown'. The database contains just users having specified their home city.
FOLLOWS (FollowerNickname, FolloweeNickname, StartDate)
TWEET (UserNickname, Timestamp, Text, CityName*, Language*, UserRetweetedTweet*, TimestampRetweetedTweet*) *// Not all the tweets are geolocalized, so for some of them the city is not known. If this tweet is not a retweet, the attributes UserRetweetedTweet and TimestampRetweetedTweet are null.*

UniTwitter uses a NoSQL document-based store. The NoSQL store is composed of a single document collection named Tweet, constituted by a set of JSON documents; each document of the collection represents a tweet.
A JSON document is a set of key-value pairs. Keys are strings, while values may be strings, arrays of strings (delimited by square brackets), or sub-documents (delimited by braces).

A sample JSON document that is part of the collection, representing information related to a tweet, is the following:

```
{
        "TweetId": "1234567",
        "Timestamp": "2017-04-08 20:35:45.143",
        "Language": "English",
        "Location": {
                        "Latitude": "38.899",
                        "Longitude": "-77.085",
                        "City": {
                                        "Name": "Washington",
                                        "Country": "USA"
                                }
                },
        "Text": "A man arrested after Stockholm truck attack is being held on suspicion of terror",
        "User": {
                        "UserId": "555666",
                        "UserName": "@MarkTheNewYorker",
                        "Real Name": "Mark Green",
                        "City": {
                                        "Name": "New York",
                                        "Country": "USA"
                                }
                },
        "RetweetOf": "8769890",
        "FavoredBy": ["121555", "777333", "442137"]
}
```

- Note that the key Location is associated with a sub-document with three keys: Latitude, Longitude and City. City, in its turn, is associated with a sub-document. Also the key User is associated with a sub-document.
- TweetId is a tweet identifier that is univocal worldwide.
- UserId is a user identifier that is univocal worldwide.
- RetweetOf is specified if this tweet is a retweet of another one, and represents the identifier of the tweet that has been retweeted.
- FavoredBy is an array of user identifiers, representing the users who have indicated this tweet as favorite.
- The sample document above contains all the possible keys. However, some of them are optional: Language, Location, RetweetOf, FavoredBy. Within the sub-document associated with Location, Latitude, Longitude and City are all required. Within City, both Name and Country are required. Within User, all the keys are required (the store contains just users having specified their home city).

Another example of JSON document representing a tweet, which does not specify all the available fields, is the following:

```
{
        "TweetId": "5436582",
        "Timestamp": "2017-04-08 20:35:45.143",
        "Text": "Today I'm very happy!",
        "User": {
                        "UserId": "324983",
                        "UserName": "@Susan_87",
                        "Real Name": "Susan White",
```

```
            "City": {
                        "Name": "Los Angeles",
                        "Country": "USA"
                    }

        }
}
```

Notes:

- PoliTwitter and UniTwitter consider disjoint subsets of users, referred to different geographic areas.
- Since the users in the two data sources are disjoint, you can assume that also the tweets are disjoint.

1. **Source schema reverse engineering**. Provide, for each input data source, the reverse engineering from the logical schema to the conceptual model (ER graph). For the NoSQL source *UniTwitter* provide also the relational translation of the ER schema. (5 points)

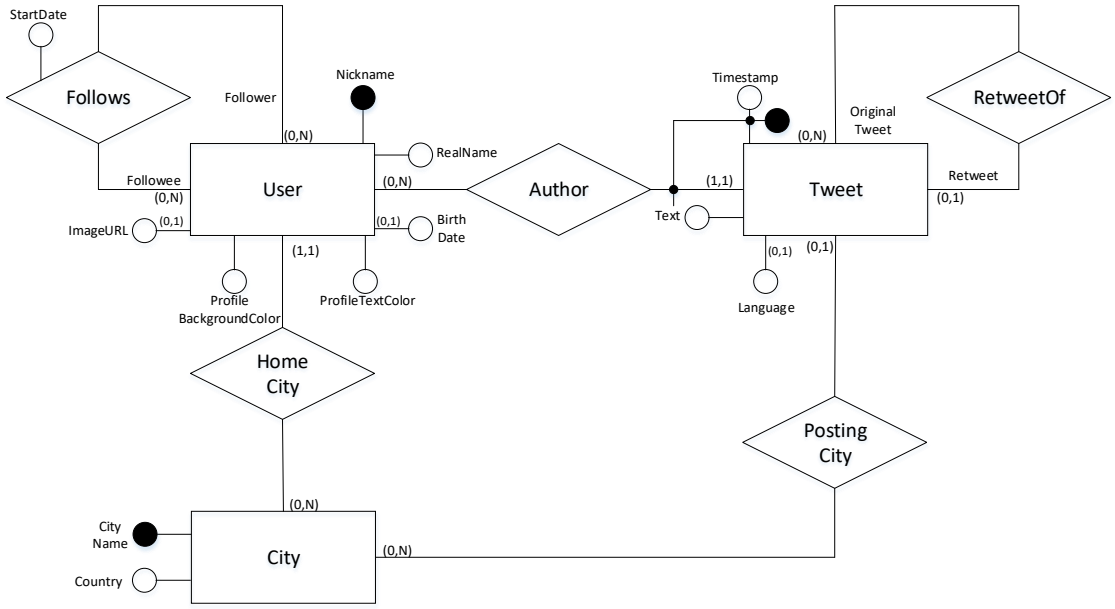
2. **Schema integration**. Design an integrated global conceptual schema (ER graph) for *UniPoliTwitter* capturing all the data coming from both *PoliTwitter* and *UniTwitter*, and provide the corresponding global logical (relational) schema. In more detail, follow these steps:
    a. *Related concept identification and conflict analysis and resolution*. Write a table as shown in the exercise sessions, using the following columns: "PoliTwitter concept", "UniTwitter concept", "Conflict", "Solution". (3.5 points)
    b. *Integrated conceptual schema* (ER graph). (3.5 points)
    c. *Conceptual to logical translation of the integrated schema*. (2 points)

3. **Query answering and mapping definition**. Consider the query Q: "Find the nationalities of the users that have posted from Milan or New York at least one tweet containing the word 'soccer'".
    a. *Query formulation*. Consider query Q posed on the logical schema of *UniPoliTwitter* and write it in SQL. (1.5 points)
    b. *Mapping definition*. Write the GAV mappings between the schema of *UniPoliTwitter* and the two sources using SQL. For *UniTwitter*, write the mappings between the *UniPoliTwitter* integrated relational schema and the *UniTwitter* relational schema defined at point 1. Write the mappings only for the tables used to answer query Q**.** (4 points)
    c. *Query rewriting*. Show the rewriting of Q on the two data sources using SQL. Again, for *UniTwitter* consider the relational schema defined at point 1. (2.5 points)

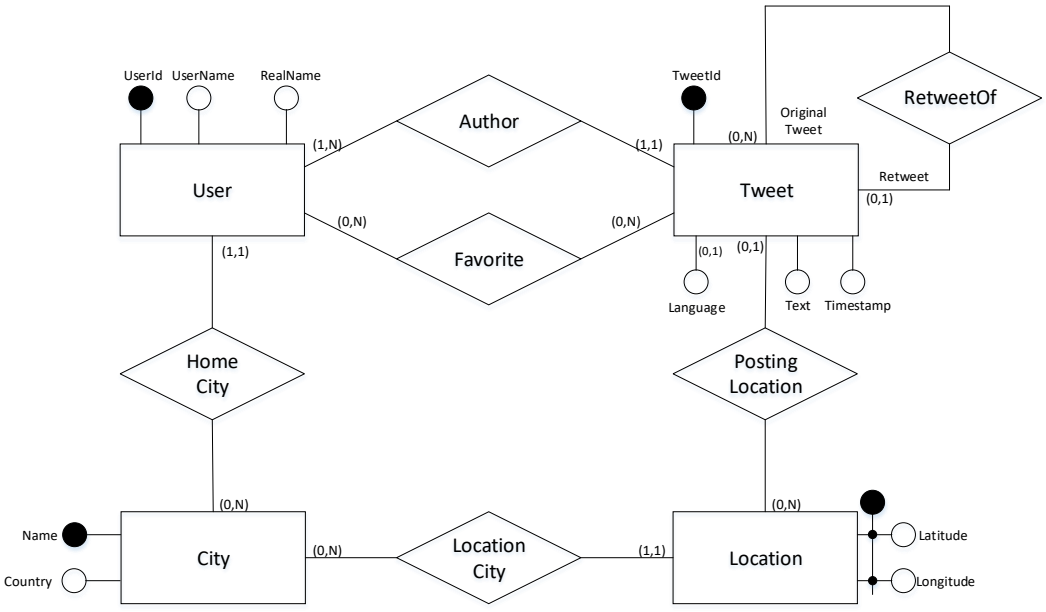
SOLUTION

1. Source schema reverse engineering

PoliTwitter



UniTwitter



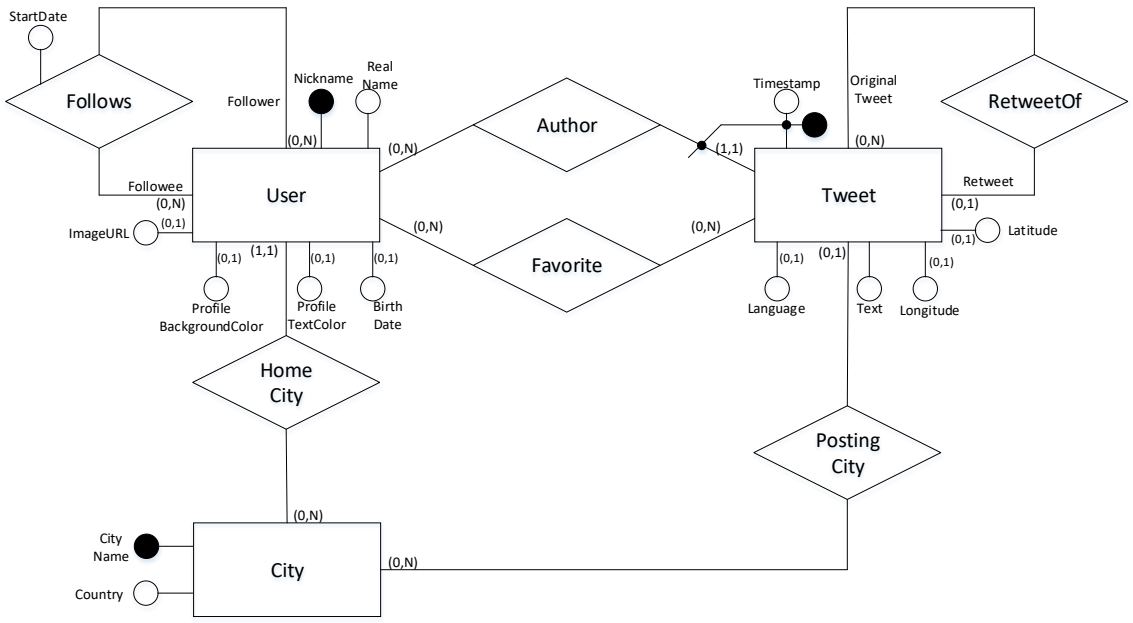
Relational schema
City (Name, Country)
User (UserId, UserName, RealName, CityName)
Location (Latitude, Longitude, CityName)
Tweet (TweetId, UserId, Timestamp, Text, Latitude*, Longitude*, Language*, RetweetOf*)
Favorite (TweetId, UserId)

2. Schema integration

2a) Related concept identification + conflict analysis and resolution

PoliTwitter	UniTwitter	Conflict	Solution
City	City	Name conflicts	
		- CityName → Name	CityName
User	User	Name conflicts	
		- Nickname → UserName	Nickname
		Key conflict	
		- Nickname → UserId	Nickname (available in both the data sources)
Tweet	Tweet	Key conflict	
		- Nickname+Timestamp → TweetId	Use Nickname + Timestamp (available in both the data sources)
		Structure conflicts	
		- City directly connected to the Tweet → City connected to the Tweet through the Location	Move Latitude and Longitude on the Tweet entity and connect the City directly to the Tweet

2b) Global conceptual schema



2c) Conceptual to logical translation

City (CityName, Country)
User (Nickname, RealName, BirthDate*, CityName, ImageURL*, ProfileBackgroundColor*, ProfileTextColor*)
Follows (FollowerNickname, FolloweeNickname, StartDate)
Tweet (UserNickname, Timestamp, Text, CityName*, Language*, UserRetweetedTweet*, TimestampRetweetedTweet*, Latitude*, Longitude*)
Favorite (NicknameTweet, TimestampTweet, Nickname)

3. Query answering and mapping definition

3a) Query formulation

Find the nationalities of the users that have posted from Milan or New York at least one tweet containing the word 'soccer'.

```
SELECT DISTINCT C.Country
FROM Tweet AS T, User AS U, City AS C
WHERE T.UserNickname=U.Nickname AND U.CityName=C.CityName AND T.Text LIKE '%soccer%' AND
      (T.CityName='Milan' OR T.CityName='New York')
```

3b) GAV mapping definition

```
CREATE VIEW UniPoliTwitter.Tweet (UserNickname, Timestamp, Text, CityName, Language,
UserRetweetedTweet, TimestampRetweetedTweet, Latitude, Longitude) AS (

      SELECT UserNickname, Timestamp, Text, CityName, Language, UserRetweetedTweet,
            TimestampRetweetedTweet, null, null
      FROM PoliTwitter.Tweet

      UNION

      SELECT U.UserName, T.Timestamp, T.Text, L.CityName, T.Language, U-Ret.Nickname,
            Ret.Timestamp, L.Latitude, L.Longitude
      FROM UniTwitter.Tweet AS T JOIN UniTwitter.User AS U ON T.UserId=U.UserId LEFT JOIN
            UniTwitter.Location AS L ON T.Latitude=L.Latitude AND T.Longitude=L.Longitude LEFT JOIN
            UniTwitter.Tweet AS Ret ON T.RetweetOf=Ret.TweetId LEFT JOIN UniTwitter.User AS U-Ret
            ON Ret.UserId=U-Ret.UserId
)

CREATE VIEW User (Nickname, RealName, BirthDate, CityName, ImageURL, ProfileBackgroundColor,
      ProfileTextColor) AS (
```

```
        SELECT Nickname, RealName, BirthDate, CityName, ImageURL, ProfileBackgroundColor,
                ProfileTextColor
        FROM PoliTwitter.User

        UNION

        SELECT UserName, RealName, null, CityName, null, null, null
        FROM UniTwitter.User
)


CREATE VIEW City (CityName, Country) AS (

        SELECT CityName, Country
        FROM PoliTwitter.City

        UNION

        SELECT Name, Country
        FROM UniTwitter.City
)
```

3c) Query rewriting

```
SELECT C.Country
FROM PoliTwitter.Tweet AS T, PoliTwitter.User AS U, PoliTwitter.City AS C
WHERE T.UserNickname=U.Nickname AND U.CityName=C.CityName AND T.Text LIKE '%soccer%' AND
        (T.CityName='Milan' OR T.CityName='New York')

UNION

SELECT C.Country
FROM UniTwitter.Tweet AS T, UniTwitter.User AS U, UniTwitter.City AS C, Location AS L
WHERE T.UserId=U.UserId AND U.CityName=C.Name AND T.Latitude=L.Latitude AND
        T.Longitude=L.Longitude AND T.Text LIKE '%soccer%' AND (L.CityName='Milan' OR
        L.CityName='New York')
```