

EECS 484 F15 Project 1

Database Design for Social Network Data

Due: Oct. 1st, 2015, 11:55 PM

(Up to 4-day late with 15% penalty is permitted)

Overview

In Project 1, you will design a relational database for storing information about your Fakebook social network. You will begin with a detailed description of the content. Then, you will need to systematically go through the conceptual and logical database design process you learned about in class. You can do the project either alone or in a group of two. If working in a group, a single submission is required.

Part 1: ER Design

As a starting point, we have done the initial “requirements analysis” for you. The following is a brief description of the data that you will store in your database. (In real life, you would probably begin with much fuzzier information.) **All IDs in the specs below are, of course, unique.**

User Information

There can be an unlimited number of users. Each user has the following information:

- **Profile information**

This includes the following attributes: user ID, first name, last name, year of birth, month of birth, date of birth, gender.

- **Hometown Location**

A user’s hometown includes the following attributes: city, state, country.

- **Current Location**

Exactly the same attributes as hometown location.

- **Education History**

A user's educational history contains information on each college program attended, if any, with each college program attended containing the following attributes: name of the institution (e.g., University of Michigan), year of graduation, concentration (e.g., CS, EE, etc.), and degree (e.g., BS, MS, PhD, etc.).

- **Friendship information**

Each user can have any number of friends. Each friend must also be a Fakebook user.

Photos

“Photos” is an important Fakebook application. It records the following information:

● Album information

Each photo MUST belong to exactly one album. An album has the following attributes: album_ID, owner_ID (this refers to the owner's Facebook ID), album_name, cover_photo_ID (this refers to a photo ID), album_created_time, album_modified_time, album_link and album_visibility.

● Other information

Each photo has the following attributes: photo_ID, photo_caption, photo_created_time, photo_modified_time, and photo_link.

● Photo Tags

Users can also interact by tagging each other. A photo tag identifies a Facebook user in a photo. It has the following associated attributes:

tag_photo_id (a Facebook photo ID), tag_subject_id (a Facebook user ID), tag_x_coordinate and tag_y_coordinate, and tag_created_time

The database does not track who did the tagging.

Note that there can be multiple tags at exactly the same (x, y) location. However, there can be only ONE tag for each subject in the photo; Facebook doesn't allow multiple tags for the same subject in a single photo. For example, you cannot tag Lady Gaga twice in a photo, even if she appears to be at two separate locations in the photo.

Messages

Users can also send private messages to each other.

● Message information

sender_ID (a Facebook user ID), receiver_id (a Facebook user ID), message_content (the text of the message), and sent_time

In this version of Facebook, there are no group messages. A user can, of course, send zero or more messages and to different users.

Events

"Events" is another useful Facebook feature.

● Basic event information

event_ID, event_creator_id (Facebook user who created the event), event_name, event_tagline, event_description, event_host (this is a string, not a Facebook user), event_type, event_subtype, event_location, event_city, event_state, event_country, event_start_time, and event_end_time

● Event participants

Participants in an event must be Fakebook users. Each participant must have a confirmation status value (attending, declined, unsure, or not-replied). The sample data does not have information on Event Participants, so you can leave the information on Participants empty.

Task for Part 1

Your task in Part 1 is to perform “Conceptual Database Design” using ER Diagrams. There are many ER variants, but for this project, we expect you to use the conventions from the textbook and lecture.

Hints for Part 1

You need to identify the entity sets and relationship sets in a reasonable way. We expect there to be multiple correct solutions; ER design is somewhat subjective. Your goal should be to capture the given information using ER constructs that you have learned about in class (participation constraints, key constraints, weak entities, ISA hierarchies and aggregation) as necessary.

For the entity set, relationship set and attribute names, you can use the ones we have provided here, or you may also choose your own names, as long as they are intuitive and unambiguous.

Before you get started, you should also read Part 4 to understand the specifics of the data.

Part 2: Logical Database Design

For the second part of the project, your task is to convert your ER diagrams into relational tables. You are required to write SQL DDL statements for this part. You should turn in two files:

1. createTables.sql
2. dropTables.sql

Hints for Part 2

You should capture as many constraints from your ER diagrams as possible in your createTables.sql file. In your dropTables.sql, you should write the DROP TABLE statements necessary to destroy the tables you have created.

Using Oracle SQL*Plus, you can run your .sql files with the following commands:

```
sqlplus <accountName>/<password> @ dropTables.sql
```

```
sqlplus <accountName>/<password> @ createTables.sql
```

Please double-check that you can run the following sequence without errors. Otherwise, you may fail our auto-grading scripts. Also remember to drop any triggers, constraints, etc., that you created.

- createTables.sql
1. dropTables.sql
 2. createTables.sql
 3. dropTables.sql

Part 3: Populate Your Database

For this part of the project, you will populate your database with Fakebook data, described in Part 5. You should turn in the set of SQL statements (DML) to load data from the public tables (**PUBLIC_USER_INFORMATION**, etc.) into your tables. You should put all the statements into a file called "loadData.sql".

Hints for Part 3

There will be some variations depending on the schema that you choose. In most cases, however, you can load data into your tables using very simple SQL commands.

Please double-check that you can run the following sequence without errors. Otherwise, you may fail our auto-grading scripts.

- createTables.sql
- loadData.sql
- 4. dropTables.sql
- 5. createTables.sql
- 6. loadData.sql
- 7. dropTables.sql

Your loadData.sql must load from our PUBLIC datasets, not from a private copy. We will be testing your system against hidden datasets and therefore need your loadData.sql to be loading from the specified dataset. Otherwise, you will fail our tests.

Part 4: Create views on your database

As a final thing, you will create some views on your tables. Here is what we would like:

Define views to recreate the same schemas as the PUBLIC tables (see Part 5). The rows in a view do not have to be in exactly the same order as in the corresponding table in the PUBLIC datasets, but the schema must be identical. The columns must have identical names and types. You can check the schema of the PUBLIC tables by using the "DESC TableName" command. For the public dataset, the original data satisfied all the integrity constraints, each view will have the same set of rows as in the corresponding input table. Name your view tables as follows (correspondence to the public tables should be obvious -- See Part 5 later)

- **VIEW_USER_INFORMATION**
- **VIEW_ARE_FRIENDS**
- **VIEW_PHOTO_INFORMATION**
- **VIEW_TAG_INFORMATION**

● VIEW_EVENT_INFORMATION

Turn in the following files that create and drop the views:

- createviews.sql
- dropviews.sql

Hints for Part 4:

1. You should check that the following sequence works correctly (no errors).

- createTables.sql
- loadData.sql
- createviews.sql
- dropviews.sql
- dropTables.sql
- createTables.sql
- loadData.sql
- createviews.sql
- dropviews.sql
- dropTables.sql

2. You should also check for the provided dataset that createviews.sql results in identical tables to the provided tables. For example, the following checks should result in an empty result:

- ☐ SELECT * FROM keykholt.PUBLIC_USER_INFORMATION

MINUS

SELECT * FROM VIEW_USER_INFORMATION;
- SELECT * FROM VIEW_USER_INFORMATION

MINUS
SELECT * FROM keykholt.PUBLIC_USER_INFORMATION;

You should apply the same checks for all the public and view tables.

3. You may also wish to further test your system to make sure it is observing the specified integrity constraints with your own test input tables. Attempting to insert data that violates the specified constraints should fail.

Submission Checklist

Please put all your files in a single zip file called project1.zip and submit a single file via Canvas Assignments (Project 1). The zip file should contain the following files:

1. A PDF document that contains your ER Diagram from Part 1. You may also draw the ER diagram by hand, and submit an electronic version by scanning the drawing. Name the file ER_Diagram.pdf.
2. A short summary report in a file mytables.pdf that contains the following in a tabular form for all your tables: table name, table's schema, and the count of number of rows in that relation. You can retrieve the count for each table as follows (replace *table_name* by the table name):

```
SELECT COUNT (*) FROM table_name;
```
3. Five SQL files
 - a. createTables.sql (Part 2)
 - b. dropTables.sql (Part 2)
 - c. loadData.sql (Part 3, Part 5)
 - d. createViews.sql (Part 4, Part 5)
 - e. dropViews.sql (Part 4, Part 5)

Only **single** submission is required per team (from either one of the member).

How to create a zip file?

Log into a Linux machine. Put all your submission files into one folder

```
% zip -r project1.zip ER_Diagram.pdf mytables.pdf createTables.sql dropTables.sql loadData.sql  
createViews.sql dropViews.sql
```

You MUST create the zip file using the above command as exactly typed. That ensures that you include the correct set of files with exactly the right names. You can add in a README.txt file if you wish as well for any additional information.

To test that your zip file contains everything, email or copy the zip to another machine or folder and unzip it to make sure you are able to extract all the files.

Part 5: Fake data set

This section describes the format of the fake data we will provide you to load into your database

Fake social network data

Everyone will have access to a fake data set, which is designed to emulate a social network dataset. The fake data includes the following five tables:

```
PUBLIC_USER_INFORMATION  
PUBLIC_ARE_FRIENDS  
PUBLIC_PHOTO_INFORMATION  
PUBLIC_TAG_INFORMATION
```

PUBLIC_EVENT_INFORMATION

These tables are stored in the GSI's account (keykholt). You can access the public tables for the fake data using GSI's account name (keykholt). **For example, to access the PUBLIC_USER_INFORMATION table, you need to refer to the table name as keykholt.PUBLIC_USER_INFORMATION.** You can copy the data into your own account with the following command:

```
CREATE TABLE NEW_TABLE_NAME AS (SELECT * FROM keykholt.TABLE_NAME);
```

The data will then be stored into your personal Oracle space. You can login to SQL*Plus to browse the data.

Fake data raw schema

The fake data tables we provide actually give you some hints on the previous parts of the assignment. However, these tables are highly “denormalized” (poorly designed), and without any table constraints.

As mentioned earlier, the table names are:

```
PUBLIC_USER_INFORMATION  
PUBLIC_ARE_FRIENDS  
PUBLIC_PHOTO_INFORMATION  
PUBLIC_TAG_INFORMATION  
PUBLIC_EVENT_INFORMATION
```

The fields of those tables are as follows:

PUBLIC_USER_INFORMATION table:

1. USER_ID
This is the Fakebook unique ID for users
2. FIRST_NAME
Every user **MUST** have a first name on file
3. LAST_NAME
Every user **MUST** have a last name on file
4. YEAR_OF_BIRTH
Some users may not provide this information
5. MONTH_OF_BIRTH
Some users may not provide this information
6. DAY_OF_BIRTH
Some users may not provide this information
7. GENDER
Some users may not provide this information
8. HOMETOWN_CITY
Some users may not provide this information

9. **HOMETOWN_STATE**
Some users may not provide this information
10. **HOMETOWN_COUNTRY**
Some users may not provide this information
11. **CURRENT_CITY**
Some users may not provide this information
12. **CURRENT_STATE**
Some users may not provide this information
13. **CURRENT_COUNTRY**
Some users may not provide this information
14. **INSTITUTION_NAME**
Some users may not provide this information.
A single person may have studied in multiple institutions (college and above).
15. **PROGRAM_YEAR**
Some users may not provide this information.
A single person may have enrolled in multiple programs.
16. **PROGRAM_CONCENTRATION**
Some users may not provide this information.
This is like a short description of the program.
17. **PROGRAM_DEGREE**
Some users may not provide this information.

PUBLIC_ARE_FRIENDS table

1. **USER1_ID**
2. **USER2_ID**

Both **USER1_ID** and **USER2_ID** refer to the values in the **USER_ID** field of the **USER_INFORMATION** table. If two users appear on the same row, it means they are friends; otherwise they are not friends. A pair of users should only appear once in the table (i.e., a pair should only appear in one of the two possible orders).

PUBLIC_PHOTO_INFORMATION table

All attributes must be present unless otherwise specified

1. **ALBUM_ID**
ALBUM_ID is the Facebook unique ID for albums.
2. **OWNER_ID**
User ID of the album owner.
3. **COVER_PHOTO_ID**
Each album **MUST** have one cover photo, and that photo must be in the album. The values are the Facebook unique IDs for photos.
4. **ALBUM_NAME**
5. **ALBUM_CREATED_TIME**
6. **ALBUM_MODIFIED_TIME**

7. ALBUM_LINK
The URL directly to the album
8. ALBUM_VISIBILITY
It is one of the following values: EVERYONE, FRIENDS_OF_FRIENDS, FRIENDS, MYSELF, CUSTOM
9. PHOTO_ID
This is the Fakebook unique ID for photos.
10. PHOTO_CAPTION
An arbitrary string describing the photo. This field is not necessarily populated.
11. PHOTO_CREATED_TIME
12. PHOTO_MODIFIED_TIME
13. PHOTO_LINK
The URL directly to the photo

PUBLIC_TAG_INFORMATION table

All attributes must be populated.

1. PHOTO_ID
Unique Id of the corresponding photo
2. TAG_SUBJECT_ID
Unique Id of the corresponding user
3. TAG_CREATED_TIME
4. TAG_X_COORDINATE
5. TAG_Y_COORDINATE

PUBLIC_EVENT_INFORMATION table

All required unless otherwise specified

1. EVENT_ID
This is the Fakebook unique ID for events.
2. EVENT_CREATOR_ID
Unique Id of the user who created this event
3. EVENT_NAME
4. EVENT_TAGLINE
Not necessarily provided
5. EVENT_DESCRIPTION
Not necessarily provided
6. EVENT_HOST
7. EVENT_TYPE
Fakebook has a fixed set of event types to choose from a drop-down menu.
8. EVENT_SUBTYPE
Fakebook has a fixed set of event subtypes to choose from a drop-down menu.
9. EVENT_LOCATION
User entered arbitrary string. For example, "my backyard". Not necessarily provided
10. EVENT_CITY

- Not necessarily provided.
- 11. EVENT_STATE
Not necessarily provided.
 - 12. EVENT_COUNTRY
Not necessarily provided.
 - 13. EVENT_START_TIME
 - 14. EVENT_END_TIME

Part 5: Oracle and SQL*Plus

This section describes how to get started using Oracle and SQL*Plus.

Logging in to your Oracle Account

First, connect to login.engin.umich.edu using SSH with your UMich account (username and Kerberos password).

Then execute:

```
module load oracle/muscle
sqlplus
```

NOTE: if you add the “module load” command to your ~/.profile, it will always be executed when you log in to your CAEN account. Then, to connect to the Oracle server, you will just have to enter the sqlplus command.

Enter the user name and password for your Oracle account to login. The default password is eecsclass. When you log in the first time, you will be prompted to change your password. Oracle passwords can contain any alpha numeric characters and underscore (_), dollar (\$), and number sign (#). **Do not use quotation marks or the @ symbol in your new password.** If you do, and find that you cannot log in, email one of the instructors to reset your password. After that, you can type SQL commands to interact with the database system. Note that you must end every statement you want to execute with a semicolon.

To disconnect from Oracle you can execute:

```
EXIT
```

Try this early! If you have trouble accessing your Oracle account, please speak to the GSI.

Troubleshooting

If you run into trouble accessing your account, first look at the google doc listed in the next section. This can help if you improperly disconnect from the server. If you get locked out, forget your password, or are otherwise unable to access your account after looking at the hints doc, email one of

the instructional staff, and we will handle it for you. You do NOT need to go through ITS for this, and we will respond faster.

General Hints on Using SQLPlus Effectively

We have posted some hints on using SQLPlus effectively for this project at this link:

https://docs.google.com/a/umich.edu/document/d/1Qqr-XgauEb8ow4JP1_vmLQ1vM8MWYxbIZi24jxoBI-l/edit?pli=1

The above document is not a tutorial on sqlplus and may not make sense right away. However, it contains many useful tips that are worth looking at (such as how to get command line history). Glance over the first few pages as you get started, and then refer to it as you run into problems during the project. It may have the answers.

Here are some basic commands to browse your data.

- View all the existing tables:

```
SELECT TABLE_NAME FROM USER_TABLES;
```

- View the schema of a table:

```
DESC TABLE_NAME;
```

This includes both field names and datatypes, which will inform the datatypes you use in your own database.

- Browse all the data in a table:

```
SELECT * FROM TABLE_NAME;
```

- Browse the first n rows in a table:

```
SELECT * FROM TABLE_NAME WHERE ROWNUM<N;
```

To change the output format of table columns you can use the “COLUMN” command with the “FORMAT” option. For example, the following two commands can be used to display the first 20 characters of USER1_ID and USER2_ID.

```
COLUMN USER1_ID FORMAT A20;
```

```
COLUMN USER2_ID FORMAT A20;
```

Then, the output of the following “SELECT” statement will be displayed as a table in user-friendlier format.

```
SELECT * FROM ARE_FRIENDS WHERE ROWNUM < 3;
```

Triggers

Suppose (whether or not it is a good design) that you created a table LOCATION, which contains the attributes LOC_ID, CITY, STATE, and COUNTRY. Suppose that you want this table to contain a listing of all the different locations, without duplicates. You might load data into the table using the following command (UNION eliminates duplicates):

```
INSERT INTO LOCATION (CITY, STATE, COUNTRY)
SELECT DISTINCT HOMETOWN_CITY, HOMETOWN_STATE, HOMETOWN_COUNTRY FROM
PUBLIC_USER_INFORMATION
UNION
SELECT DISTINCT CURRENT_CITY, CURRENT_STATE, CURRENT_COUNTRY FROM
PUBLIC_USER_INFORMATION
UNION
SELECT DISTINCT EVENT_CITY, EVENT_STATE, EVENT_COUNTRY FROM
PUBLIC_EVENT_INFORMATION;
```

You may also find yourself in a situation where it would be useful to construct an internal key (i.e., a key whose value is meaningless outside the database), such as the LOC_ID mentioned above. You can do this in Oracle by declaring a sequence variable and a trigger. For example:

```
CREATE SEQUENCE loc_sequence
START WITH 1
INCREMENT BY 1;

CREATE TRIGGER loc_trigger
BEFORE INSERT ON LOCATION
FOR EACH ROW
BEGIN
SELECT loc_sequence.nextval into :new.LOC_ID from dual;
END;

.
RUN;
```

Whenever you insert a row into LOCATION, the above will automatically set the value of LOC_ID to the next integer in the sequence.

As a useful additional reference, you may also want to look at a more extensive guide maintained by Jeff Ullman at Stanford: <http://infolab.stanford.edu/~ullman/fcdb/oracle.html>

There may be some pieces of data that we have asked you to represent in your database schema (ER diagram and relational tables), but for which we have given you no data. Please do represent these items in your schemas (ER diagrams and CREATE TABLE DDL). However, when you load the data from the provided schema, don't worry about populating these fields. That is, you should have either empty tables, or null values, depending on how you have designed the schema.