# EECS 484 F16 Project 1 Database Design for Social Network Data Due: September 29, 11:55 PM

(Late penalty for this particular project will be 5% (flat), since this is the first project. Up to 4 late days allowed. Start early. You may have to learn some basic SQL to generate the views and to import the datasets  for the last part on your own.  We will go through some SQL in the lectures/discussions, but some self-learning is likely to be required..) Discussion slides have additional important information on how to login to Oracle, etc. Make sure you attend the discussions.

## 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

## You should do this to derive constraints for the relational model and to make sure you can go from requirements to ER model to Relational model.

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 Fakebook 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 Fakebook user in a photo. It has the following associated attributes:

tag_photo_id (a Fakebook photo ID), tag_subject_id (a Fakebook 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; Fakebook 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 Fakebook user ID), receiver_id (a Fakebook user ID), message_content (the text of the message), and sent_time

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

## Events:

“Events” is another useful Fakebook feature.

● **Basic event information**

event_ID, event_creator_id (Fakebook user who created the event), event_name, event_tagline, event_description, event_host (this is a string, not a Fakebook 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 Appendix to understand the specifics of the data. Some of the ER diagram constraints are in Appendix.

## 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

As a starting point, we are giving you a set of tables, along with some columns. Your design must use these tables. But, you will need to add any integrity constraints so that the schema is as close to enforcing the requirements as is practical. You can add additional columns as well. Use the most appropriate types for the fields as well.

The required tables and their schema are given below:

USERS:

USER_ID (NUMBER)

FIRST_NAME (VARCHAR2(100))

LAST_NAME (VARCHAR2(100))

YEAR_OF_BIRTH (INTEGER)

MONTH_OF_BIRTH (INTEGER)

DAY_OF_BIRTH (INTEGER)

GENDER (VARCHAR2(100))

FRIENDS:

USER1_ID (NUMBER)

USER2_ID(NUMBER)

CITIES:

CITY_ID (INTEGER)

CITY_NAME(VARCHAR2(100))

STATE_NAME (VARCHAR2(100))

COUNTRY_NAME (VARCHAR2(100))

USER_CURRENT_CITY:

USER_ID (NUMBER)

CURRENT_CITY_ID (INTEGER)

USER_HOMETOWN_CITY:

USER_ID (NUMBER)

HOMETOWN_CITY_ID (INTEGER)

MESSAGE:

MESSAGE_ID (INTEGER)

SENDER_ID (NUMBER)

RECEIVER_ID(NUMBER)

MESSAGE_CONTENT (VARCHAR2(2000))

SENT_TIME (TIMESTAMP)

PROGRAMS:

PROGRAM_ID (INTEGER)

INSTITUTION (VARCHAR2(100))

CONCENTRATION (VARCHAR2(100))

DEGREE (VARCHAR2(100))

EDUCATION:

USER_ID (NUMBER)

PROGRAM_ID (INTEGER)

PROGRAM_YEAR (INTEGER)

USER_EVENTS:

EVENT_ID (NUMBER)

EVENT_CREATOR_ID (NUMBER)

EVENT_NAME (VARCHAR2(100))

EVENT_TAGLINE (VARCHAR2(100))

EVENT_DESCRIPTION (VARCHAR2(100))

EVENT_HOST (VARCHAR2(100))

EVENT_TYPE (VARCHAR2(100))

EVENT_SUBTYPE (VARCHAR2(100))

EVENT_LOCATION (VARCHAR2(100))

EVENT_CITY_ID (INTEGER)

EVENT_START_TIME (TIMESTAMP)

EVENT_END_TIME (TIMESTAMP)

PARTICIPANTS:

EVENT_ID (NUMBER)

USER_ID (NUMBER)

CONFIRMATION (VARCHAR2(100))

ALBUMS:

ALBUM_ID (VARCHAR2(100))

ALBUM_OWNER_ID (NUMBER)

ALBUM_NAME (VARCHAR2(100))

ALBUM_CREATED_TIME (TIMESTAMP)

ALBUM_MODIFIED_TIME (TIMESTAMP)

ALBUM_LINK (VARCHAR2(2000))

ALBUM_VISIBILITY (VARCHAR2(100))

COVER_PHOTO_ID (VARCHAR2(100))

PHOTOS:

PHOTO_ID (VARCHAR2(100))

ALBUM_ID (VARCHAR2(100))

PHOTO_CAPTION (VARCHAR2(2000))

PHOTO_CREATED_TIME (TIMESTAMP)

PHOTO_MODIFIED_TIME (TIMESTAMP)

PHOTO_LINK (VARCHAR2(2000))

TAGS:

TAG_PHOTO_ID (VARCHAR2(100))

TAG_SUBJECT_ID (NUMBER)

TAG_CREATED_TIME (TIMESTAMP)

TAG_X (NUMBER)

TAG_Y (NUMBER)

Keep the table and field names exactly as written above. Also, make sure you use the correct field types (e.g., number or integer) as specified above. Failure to do so may result in failing the autograder since the database is case and type sensitive. (Note: The ID types for various fields would normally be INTEGERs in practice, but they are not in this project for reasons other than technical, primarily, that the input data sets we are importing contains non-integer types for keys -- use it as a learning moment to deal with IDs of different types!)

You need to decide what fields will be primary keys and what fields will be foreign keys(if necessary). Use the smallest candidate keys when possible for primary keys.

## 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. (Also, for your own learning, it is good to check if your ER diagrams map to the tables we gave you. If not, you may want to discuss this in the discussions, piazza, or office hours to determine whether the fault lies in your ER diagrams or the tables we gave you).

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 in a single sql script. Otherwise, you may fail our auto-grading scripts. Also remember to drop any triggers, constraints, etc., that you created.

- createTables.sql
- dropTables.sql
- createTables.sql
- dropTables.sql
-

## Part 3: Populate Your Database

For this part of the project, you will populate your database with Fakebook data (please see Appendix 1 on where to find the dataset and its description). 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 in a single sql script. Otherwise, you may fail our auto-grading scripts. Also remember to drop any triggers, constraints, etc., that you created.

- createTables.sql
- loadData.sql
- dropTables.sql

**Your loadData.sql must load from our PUBLIC dataset given in the Appendix, 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.

One concern you might have is how to handle the constraint on regard Friend data. For this project, when loading the data, ensure that only the necessary data is loaded. For example, if the original data contains (2,7) and (7,2), only load one of these two values. Loading both or neither would be incorrect. After the data has been loaded, you only need to ensure that any insertion of new data does not break the no duplication constraint. This can either be done by rejecting any insert or batch insert which would violate the constraint or only accepting valid data and rejecting the rest. The first option tends to be easier.

## Part 4: Create views on your database

As a final task, 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 Appendix). 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 Appendix 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 in a single script (no errors).

● 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 two statements check if the public dataset is the same as your created view.

○ SELECT * FROM keykholt.PUBLIC_USER_INFORMATION

MINUS

SELECT * FROM VIEW_USER_INFORMATION;

● SELECT * FROM VIEW_USER_INFORMATION

MINUS SELECT * FROM keykholt.PUBLIC_USER_INFORMATION;

If both queries return no results, then the rows in both are the same.

3. It is not necessary to exactly recreate the **PUBLIC_ARE_FRIENDS** table since it is not guaranteed that for every (x,y) row entry, there is a corresponding (y,x) entry. For the **VIEW_ARE_FRIENDS**, the requirement is that for every (x,y) entry in the public dataset, it either has a (x,y) or (y,x) entry, but not both. For example, if the public dataset has both (2,7) and (7,2), your view should contain only (2,7) or (7,2).

4. **DOUBLE CHECK YOUR SCHEMA!** Many students forget to check their schema and fail the autograder because of it.

## Submission Checklist

Please put all your files in a single zip file called project1.zip and submit a single file. We will post instructions later.

1. partner.txt: List the partners who worked together. If you worked alone, just list your name. Remember to follow the Engineering Honor Code and Course policies in Lecture00.

2. ER Diagram. Filename should be er.pdf.

3. Five SQL files

a. createTables.sql (Part 2)

b. dropTables.sql (Part 2)

c. loadData.sql (Part 3)

d. createViews.sql (Part 4)

e. dropViews.sql (Part 4)

Both partners should submit separately, even if the submission is identical. Late policy applies individually (even if submissions are identical).

## How to create a zip file?

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

```
% zip -r  project1.zip partner.txt er.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.

We will update the instructions on how precisely submit your zip file to us by next week.

---

## Appendix: Description of the Fake data set for Part 3

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.

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 Fakebook 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 the photo must be in the album. The values are the

Fakebook unique IDs for photos.

4. ALBUM_NAME

5. ALBUM_CREATED_TIME

6. ALBUM_MODIFIED_TIME

7. ALBUM_LINK

    The unique 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 unique 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

## 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 (uniqname and Kerberos password).

Then execute:

**module load eecs484/f16**

NOTE: if you add the “module load” command to your ~/.bash_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.