**Virginia Tech.** **CS 4604 – Introduction to DBMS**
**Computer Science** **Spring 2014, Prakash**

# Project Assignment 2

**(due April 8th, 2014, 3:30pm, in class—hard-copy please)**

*Reminders*:

a. Out of 100 points. Contains 4 pages.
b. Rough time-estimates: 4-5 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state it clearly.
f. Unless otherwise mentioned, you may use any SQL operator seen in class. Feel free to create intermediate views for SQL.
g. Lead TA for the project: Qianzhou Du.

## Q1. Refining schemas [45 points]

In this question we will see how to refine relational schemas using functional dependencies and normalization. You should also be able see how our original 'recipe' for converting ER-diagrams to relational schemas does in fact do a good job of getting good schemas (assuming your ER diagram is correct of course!). Read the detailed description in Project Assignment 1 once more to remind yourself of the project.

Consider the following nine relations with primary key underlined (Note: this is probably not a good schema and hence we want you to investigate):

`Schema 1`

**Certification** (vid, certification_name, content)

**Video** (vid, title, release_year, producer, color, country, genre)

**Movies** (mid, mtitle, did, dname, age)

**TVEpisode** (tvid, tvtitle, episodeNum, did, dname, dage)

**InSeason** (tvid, seasonNum, Seasontitle)

**InCollection** (seasonNum, Seasontitle, collectionID, collection_title)

**Users** (uid, name, age, location, gender, rate_time, rate_score, vid)

**Friends_request** (user1, user2, request_time, response, response_time)

**Act** (vid, pid, first_name, last_name, age, gender)

*Quick summary:* Each Video has a title, release_year, producer, color, country and a unique vid. A video can have more than one genre. Each video gets a certification. Each certification has content and a unique name. Each actor has a first name, last name, age, gender and a unique pid. Similarly each user has a name, age, location, gender and a unique uid. Each collection also has a unique collectionID and a title. Each movie has a unique mid and a title, the director's unique did, his/her name (dname), and age (dage). The TVEpisode tables stores tv episodes and their directors. Each tv episode has unique id tvid, a title tvtitle and episodeNum it is part of, and also the corresponding director's information (did, dname, dage). In Friends_request, there are user1 id, and user2 id, request time, response (e.g., 'Y', 'N', null, they means agreed, disagreed, and no response respectively. If the response is 'Y', it means user1 and user2 are friends).

Q1.1 (15 points) Using the summary above, for *each* of the nine relations in `Schema 1`, list all completely non-trivial Functional Dependencies (FDs) that apply to that relation. Only list FDs that have one attribute on the right hand side. It is enough to list only the FDs in a minimal basis.
*Hint:* You essentially need to check each line in the summary, if it results in a FD for any relation.

Q1.2 (15 points) Using the FDs you got in Q1.1, for each relation in `Schema 1`, write down if it is in BCNF. Explain each answer. If any of relations are not in BCNF, decompose and normalize them to BCNF.

Q1.3 (10 points) Are the resulting relations from Q1.2 in 3NF?

Q1.4 (5 points) Are the resulting relations you get in Q1.2 the same as the ones we get from the ER conversion in Project Assignment 1? (Our solution of Project Assignment 1 is on the class website) If not, list the differences.

## Q2. Creating and querying the database [55 points]

Although the ER converted schema of Project Assignment 1 may not exactly be same as the decomposed relations above, it is should be of high quality. So to keep things simple, let's just use the ER converted schema from now on.

We have stored all the data you will use in your project in the following **eight (8)** tables (`Schema 2,` with primary key underlined`)` on our PostgreSQL server. Note that `Schema 2` is slightly different than `Schema 1` of Q1, so please read it carefully.

`Schema 2` **certify** (vid, certification_name, content)
**video** (vid, title, release_year, producer, color, country, type, genre)

**videosdirect** (vid, did, first_name, last_name, age)

**tvepisode** (tvid, episodenum, seasonnumber, seasontitle)

**incollection** (collectionid, collection_title, seasontitle, seasonnumber)

**users** (uid, first_name, last_name, age, location, gender, rate_time, rate_score, vid)

**friends_request** (user1, user2, request_time, response, response_time)

**act** (vid, pid, first_name, last_name, age, gender)

*Quick Summary:* In certify, there are video id, certification name, and certification content. The table video includes video id, video title, release year, producer, color, country, type (e.g., "movie", "tvepisode", and "collection"), and genre. In videosdirect, it contains video id, director id, director name, and director age. In table tvepisode, there are tvepisode id, episode number, season number, and season title (It means the name of the drama, e.g., "House of Cards"). In incollection, there are collection id, collection title, season number, season title. Table users includes user id, user name, user age, user location, user gender, video id, rating time, and rate score. In friends_request, there are user1 id, and user2 id, request time, response (e.g., 'Y', 'N', null, they means agreed, disagreed, and no response respectively. If the response is 'Y', it means user1 and user2 are friends.). In act table, there are video id, and performers' information, such as performer name, age, and gender.

We have stored the project dataset in `Schema 2` form in the 'cs4604s14_project' database on the server. Similar to HW4, this time you can download all of these tables to your *group database* by using the following at the *command-prompt* of cs4604.cs.vt.edu:

"pg_dump -U YOUR-PID cs4604s14_project | psql -d your-group_database"

Q2.1 (25 points) First create the VTFlix ER converted schema of Project Assignment 1 by following our DDL statements in our solutions. Then we have to insert the right tuples into them: but we want you to insert the data *from* the eight tables in `Schema 2`.

For each table in the VTFlix ER schema, show your DDL statements, and output the number of tuples for each tables.

**Example: VTFlix has the certificates table; which clearly should get data only from the certify table in Schema 2.**

1. Create the table certificates:

```
CREATE TABLE CERTIFICATES (
    Certification_name varchar primary key,
    Content varchar);
```

2. Insert proper data; (this part is easy in this example)
   INSERT INTO CERTIFICATES
   SELECT DISTINCT certification_name, content FROM CERTIFY;

3. Count the number of tuples;
   SELECT COUNT(*) FROM CERTIFICATES;
   Output: 5

*Friendly Note*: When you insert tuples into the VTFlix table Friends, insert mirror pairs for each pair of friends i.e. if A and B are friends, insert both (A, B) *and* (B, A) into the table---it will help in writing future queries.

*Note 2:* Also note that in the users table of `Schema 2,` there may be null values for `vid, rate_time and rate_score` as not all users may have rated a movie. In fact this is one of the reasons intuitively why `Schema 2` is not a good design. But VTFlix is a good design, so you have to take care that you insert only the correct tuples into ratings and userinfo tables of VTFlix (which obey their primary key/foreign key constraints e.g. the vid attribute of ratings table in VTFlix can not be null).

*Important:* After creating your VTFlix tables, please delete *your copy* of the `Schema 2` tables in your group database to save space. Also, please do not use your personal database for the project data (we won't have enough space if everyone copies data to their personal databases).

Q2.2 (30 points) After creating the tables of VTFlix as above, write and run the following SQL queries on your database. These are just sample queries to give you a sense of how SQL can enable some task for your eventual application e.g. Query A below can help find recent movies of Brad Pitt for any user. For each of the problems listed below, show:

- Your SQL query,
- The result you obtained,
- The time your query took (use the \timing command in psql to obtain the time).

While grading your solutions, we will pay attention to the **quality of your queries**, e.g., whether they are correct, the number of tables they reference, and the running time. Please desist from creating massive new tables to support answering these queries!

A. (5 points) List 10 latest videos that the actor "Brad Pitt" participated.

B. (10 points) List the names of directors who have directed at least one movie in each genre.
*Hint*: Think division but in SQL.

C. (15 points) List the user who has the most friends.
*Hint:* First create a view to calculate the number of friends for each user.