NYU Tandon School of Engineering
Computer Science and Engineering
CS-UY 3083, Introduction to Database Systems, Fall 2017 Prof Frankl

COURSE PROJECT

Project Overview

The course project this semester is PriCoSha, a system for privately sharing content items among groups of people. PriCoSha gives users somewhat more privacy than many content sharing sites by giving them more detailed control over who can see which content items they post and more control over whether other people can tag content items with a user's personal information. You may choose the kind of content your version of PriCoSha focuses on, e.g. photos, video clips, announcements about social or professional events, etc. The focus of the project will be on storing data about the content, allowing users to specify who can see and/or modify what, and providing ways for users to find content of interest. In Part 1, you'll assume that the content is photos, but in the remaining parts, you are encouraged to have other kinds of content, along with customized operations for users to work with that content.

Users will be able to log in, post content items, view (some of) the content items posted by others (public content and content posted by people who have given them access via a ``friend'' mechanism, detailed below; tag content items with usernames of people referred to in the content items (if they have permission to do so), etc. In part 1 of the project, you will design an ER diagram for the database. In part 2, you will convert my E-R diagram (which I will post later) to a relational schema, write table definitions in SQL, and write some queries. Part 3 is a milestone to guarantee that you're making progress: you will hand in some of your source code along with evidence that it is working. Part 4 will be the most work: using my schema from part 2, you will revise your queries, if necessary, and write the rest of the application code for the system. A detailed specification will be provided shortly after part 2 is due.

PART 1: Due Oct 17, 2017

PriCoSha allows users to define groups of friends (called FriendGroups), and to share content items with specific FriendGroups. For example, suppose Ann has a "family" FriendGroup and a "besties" friend group. She might share content about her cat with "family" and content about a wild party with "besties".

Draw an ER diagram modeling the data needed for the system. It should include entity sets (possibly including one or more weak entity sets) representing Person, Item, and FriendGroup and several relationship sets.

Each Person has a unique username, a password, and a name, consisting of a first name and a last name.

Each content item has a unique item ID, a date, and other pertinent data. For example, depending on the nature of your content, you might want to include a location, consisting of latitude, longitude, and/or location name, data about the size of the item, etc. Each content item is posted by exactly one person. *For Part 1, you can limit the content items to just photos, which should include a unique item ID, a date, a file path, and a name.*

A FriendGroup has a name and a description. Different people may have FriendGroups with the same name (and description), but an individual user cannot have more than one FriendGroup with the same name. For example, Ann and Bob can each have FriendGroups named “family” and they can even have the same description, but Ann cannot have two friend groups named “family”.

Each FriendGroup is owned by exacly one person (the person who is defining a group of his/her friends). Each FriendGroup has (two or more) members in the group. For example Ann can create a FriendGroup called “family” with members Ann, Cathy, David, and Ellen. A content item can be shared with multiple FriendGroups. For example, Ann can post a content item of her cat and share it with her family FriendGroup, which allows them to see the content item.

A person can comment on a content item. A comment includes a timestamp and the text of the comment. A person can make multiple comments about the same content item.

A person can tag a content item with someone’s username. For example, if David is mentioned in Ann’s cat content item, Ann or David or someone else can tag it with David’s username. We want to keep track of who added the tag (the tagger) as well as who is tagged (in this case, David), and when the tag was added. In addition PriCoSha will only display tags that have been approved by the person who is tagged (e.g. David), so we need to keep track of the status of a tag.

What You Should Do

Design an ER diagram for PriCoSha. When you do this, think about: which information should be represented as attributes, which as entity sets or relationship sets? Are any of the entity sets weak entity sets? If so, what is the identifying strong entity set? What is the primary key (or discriminant) of each entity set? What are the cardinality constraints on the relationship sets? Draw the ER diagram neatly. You may draw it by hand or use a drawing tool.

Partners, etc

You may work alone or team of up to 4 people. if your want to work on a 4 person team, you must check with me and propose some substantial extra features. You may work on part 1 alone then team up for parts 2, 3, 4. Teams will be expected to Note that each partner is expected to contribute roughly equally and each partner is responsible for understanding the entire system. The quiz or exam question about the project will test each person individually.

The total project grade will be 25% of your course grade. Part 1 counts for about 15% of the project grade. Part 2 counts for about 20% of the project grade. Part 3 counts for about 15% of the project grade. There may also be a quiz or exam question(s) based on the project.

PART 2 Due Nov Sunday 11/5

A. Using my solution to Part 1 (posted), use the procedures we studied to derive a relational database schema, then write SQL CREATE TABLE statements and execute them in your database system. Remember to include primary key and foreign key constraints. (You don't have to hand in the schema diagram, only the create table statements, but you may find it useful to draw a schema diagram.) Use the following data types:

- Dates and times are Date, DateTime, or Timestamp data types
- IDs are integers. You may use autoincrement so they'll be assigned automatically
- Passwords will be stored using cryptographic hashes of the passwords users supplies. For now, we'll just use an insecure md5 hash of the passwords. More details and further requirements on how to do this better, coming later.
- Other attributes are VARCHAR.
- Add one more attribute, is_pub, to content. This should be Boolean and will indicate whether the content is public, i.e. can be viewed by everyone, regardless of whether they're in particular user groups.

B. Write SQL INSERT statements corresponding to the following situation:

- Add the following users into the Person table. Use their initials as their username, and md5 of their initials as the password. For example, Ann is ('AA', md5('AA'), 'Ann', 'Anderson').
    - Ann Anderson, Bob Baker, Cathy Chang, David Davidson, Ellen Ellenberg, Fred Fox, Gina Gupta, Helen Harper
- Ann owns FriendGroup called "family" with users Ann, Cathy, David, and Ellen.
- Bob owns FriendGroup called "family" with users Bob, Fred, and Ellen.
- Ann owns FriendGroup called "besties" with users Ann, Gina, and Helen.
- Ann posted a content item with ID=1, caption = "Whiskers", is pub = False, and shared it with her "family" FriendGroup.
- Ann posted a content item with ID=2, name = "My birthday party", is pub = False, and shared it with her "besties" FriendGroup.
- Bob posted a content item with ID=3, name = "Rover", is pub = False, and shared it with his "family" FriendGroup.
- You can make up data for the other attributes or make them NULL.

C. Write a query to show the ID and name of each content item that is shared with David (i.e the content items David can view).

In part 3, you will specify in more detail the particular kind of content you'll be dealing with and propose some specialized features for that kind of content. You should start thinking about this in parallel with doing part 2, but you won't have to hand it in yet. This would be a good time to ask questions about specialized features you're thinking about.

PART 3 and 4
Part 3 Due: 11/21
Part 4 (completed project) Due: 12/10

In parts 3 and 4 of the project, you will use the table definitions I posted (solution to part 2) along with any additional table definitions you need, to implement application code for PriCoSha as a web-based application. You may use Python, PHP, Java, node.js, Go, or C#. If you'd like to use some other language, check with me by 11/12. You must use prepared statements if your application language supports them. Part 3, is a milestone in which you must show that you've written code for the login function and at least *n -1* others, where *n* is the number of people on your team; Part 4 is the completed project.

Your PriCoSha implementation should allow users to log in, post content items, view content items that are public or that are shared with FriendGroups to which they belong, and propose to tag content items with usernames of other users, provided the content items are visible to both the user (the tagger) and the person being tagged (the taggee), and manage tag proposals. Assume each user has already registered and created a password, which is stored as an md5 hash. Better yet, you may use a SHA-1 or SHA-2 hash and/or add salt. If you'd like, you may display actual content items, but this is not required. When we test your application, we'll check data about the content items rather than actual content items.

A content item is visible to a user U if either
- The content item is public, or
- The content item is shared with a FriendGroup to which U belongs, ~~where the FriendGroup is owned by the poster of the content item~~.

(Remember that FriendGroups are identified by their name along with the username of the owner of the group.) We will assume that in the initial database, if (x,y,z) is in SharedWith(ID,name,username) then z is the poster of content item x. We will also assume that the owner of a FriendGroup is a member of that FriendGroup. (When modifying the database, PriCoSha should enforce these constraints.)

Specifically, PriCoSha should support the following use cases:
1. **Login:** The user enters username and password. PriCoSha checks whether the hash of the password matches the stored password for that username. If so, it initiates a session, storing the username and any other relevant data in session variables, then goes to the home page (or provides some mechanism for the user to select their next action.) If the password does not match the stored password for that username (or no

such user exists) PriCoSha informs the user that the the login failed and does not initiate the session. The remaining features require the user to be logged in.

2. **View content items and info about them:** PriCoSha shows the user a list of pids,posters, pdates, and captions of content items that are visible to the her, arranged in reverse chronological order. (Optionally, include a link to an actual content item.) Along with each content item there is way for the user to see further information, including
    a. first name and last name of people who have been tagged in the content item (taggees), provided that they have accepted the tags (Tag.status == true)
    b. comments about the content item
3. **Manage tags:** PriCoSha shows the user relevant data about content items that have proposed tags of this user (i.e. user is the taggee and status false.) User can choose to accept a tag (change status to true), decline a tag (remove the tag from Tag table), or not make a decision (leave the proposed tag in the table with status == false.)
4. **Post a content item:** User enters the title (and optionally, a link to a real content item) and a designation of whether the content item is public or private. PriCoSha inserts data about the content item (including current time, and current user as owner) into the Content table. If the content item is private, PriCoSha gives the user a way to designate FriendGroups (that the user owns) with which the Photo is shared.
5. **Tag a content item:** Current user, who we'll call x, selects a content item that is visible to her and proposes to tag it with username y
    a. If the user is self-tagging (y == x), PriCoSha adds a row to the Tag table: (ID, x, x, TIMESTAMP, true)
    b. else if the content item is visible to y, PriCoSha adds a row to the Tag table: (ID, x, x, TIMESTAMP, false)
    c. else if content item is not visible to y, PriCoSha doesn't change the tag table and prints some message saying that it cannot propose this tag.
6. **Add friend:** User selects an existing FriendGroup that they own and provides first_name and last_name. PriCoSha checks whether there is exactly one person with that name and updates the Member table to indicate that the selected person is now in the FriendGroup. Unusual situation such as multiple people with the same name and the selected person already being in the FriendGroup should be handled gracefully.
7. **Your choice:** You must add ***2 * n - 1 f*****eatures, where *n i*****s the number of people in your group.** So … working alone: 1 extra feature; two person team 3 extra features, etc. If your group has more than 3 people, these must include some really interesting features. Each additional feature must involve intersactions with the database.

8. **Optional (this can be one of your extra features:) Defriend:** Think about what should be done when someone is defriended, including some reasonable approach to tags that they posted or saw by virtue of being in the friend group. Write a short summary of how you're handling this situation. Implement it and test it.