NYU Tandon School of Engineering
Computer Science and Engineering
CS-UY 3083, Introduction to Database Systems, Fall 2018 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 e-mails 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 mid October details to be determined

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 "roommates" friend group. She might share a content item about her cat with "family" and content item about the need to clean the fridge with "roommates". In addition, users who have permission to view a content item may annotate it in various ways, noted below.

For Part 1, you will 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, ContentItem, and FriendGroup, as well as several relationship sets.

Details of the data model: (With a few small changes for Parts 3 and 4, highlighted in red)
Each Person has a unique e-mail, 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 and/or format 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 own FriendGroups with the same name (and description), but an individual user cannot own 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 exactly one person (the person who is defining a group of his/her friends). Each FriendGroup has one or more members, i.e. People who *BelongTo* the FriendGroup. For example Ann can create a FriendGroup called "family" with members Ann, Cathy, David, and Ellen (each of whom *BelongsTo* Ann's family FriendGroup).

A content item can be *Shared* with ~~any number of the owner's~~ FriendGroups to which the poster belongs. For example, Ann can post a content item about her cat and share it with her family FriendGroup and with Cathy's roommates FriendGroup, provided that she's a member of Cathy's roommate FriendGroup. (In **later** parts of the project, you will implement a mechanism that will allow members of the of groups with whom a content item is shared to see the content item, while preventing people who are not in any of those groups from seeing the content item, unless the item is designated as public.)

A person can *Rate* a content item. A rating includes a timestamp and the ASCII or unicode symbol for an emoji. :-) . A person can rate the same content item multiple times. We want to keep track of who rated what and when they did so. For now (at the ER design stage) **we will not worry about** restricting the ability to rate a content item to those people who belong to groups with which the content is shared, but we may add that constraint later.)

A person can *Tag* a content item with someone's e-mail. For example, if David is mentioned in Ann's cat content item, Ann or David or someone else can tag it with David's e-mail. 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. Later, we will require that 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.

PART 2 Due Thurs Nov 1, 2018, 11:59 pm

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.
- status (tag) should be a boolean. True implying that the tag has been accepted, and false implying that the tag is pending approval from the taggee.
- Passwords will be stored using cryptographic hashes of the passwords users supplies. Use type CHAR(64) for SHA-2 256 bit hashes. More details 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.

Note that several tables will have People's emails as attributes and some tables will have more than one email attribute (if you translate directly following the rules.) Rename them to things like poster_email, group_owner_email, etc to disambiguate.

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

- Add the following users into the Person table. Ann Anderson, Bob Baker, Cathy Chang, David Davidson, Ellen Ellenberg, Fred Fox, Gina Gupta, Helen Harper

  Use their initials @nyu.edu as their e-mail, and sha256 of their initials as the password. For example, Ann is ('AA@nyu.edu', sha2('AA',256), 'Ann', 'Anderson').

- 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 “roommates” with users Ann, Gina, and Helen.
- Ann posted a content item with item_ID=1, item_name = “Whiskers”, is pub = False, and shared it with her “family” FriendGroup.
- Ann posted a content item with item_ID=2, item_name = “leftovers in fridge”, is pub = False, and shared it with her “roommates” FriendGroup.
- Bob posted a content item with item_ID=3, item_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).

Later, 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

In parts 3 and 4 of the project, you will use the table definitions I will post (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 (Due 11/21) is a milestone in which you must show that you’ve written code for at least two of the “use cases” shown below. (At least one if you’re working alone). Everyone on your team should help write this code and should understand it.

Part 4 is the completed project, due on **Dec 7 (small penalty until midnight Dec 9; increasing penalties thereafter)**. You will hand in your code, a short write up (details coming soon), explanation of your extra features (see part 8 below). You will also schedule a demo session in which one of the TAs will run through a series of tests with you.

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 e-mails 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 SHA-2 hash (optionally, with 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

Remember that FriendGroups are identified by their name along with the e-mail of the owner of the group. We will assume that in the initial database, the owner of a FriendGroup is a member of that FriendGroup. When modifying the database, PriCoSha should maintain that.

PriCoSha should support the following use cases:

1. **View public content**: PriCoSha shows the user the item_id, email_post, post_time, file_path, and item_name of public content items that were posted within the last 24 hours. (If you prefer, you can include this with use case (3) below).
2. **Login:** The user enters e-mail and password. PriCoSha checks whether the hash of the password matches the stored password for that e-mail. If so, it initiates a session, storing the e-mail 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 e-mail (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.

3. **View shared content items and info about them:** PriCoSha shows the user the item_id, email_post, post_time, file_path, and item_name of content items that are visible to the her, arranged in reverse chronological order. (Optionally, include a link to an actual content item or a way to display it.)
   Along with each content item there should be a 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. Ratings of the content item
4. **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.)
5. **Post a content item:** User enters the relevant data (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 belongs to with which the Photo is shared.
6. **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 e-mail y
   a. If the user is self-tagging (y == x), PriCoSha adds a row to the Tag table: (item_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.
7. **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 Belong 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.
8. **Your choice:** You must add **2(*n - 1) f*eatures, where *n i*s the number of people in your group.** So … working alone: 0 extra feature; two person team 2 extra features, etc. *These individual features should be designed and implemented by an individual team member who will get an individual grade for this work.* You may help your teammates understand what they should do and you may review and test their code, but they should have primary responsibility for planning and implementing their features. Each additional feature must involve interactions with the database and must be relevant to this particular project. *Generic features like a registration page are not acceptable*. For each feature you must write a description including:
    a. The name of the team member who is primarily in charge of implementing this feature
    b. What the feature is (in the style of the 6 requirements above)
    c. A sentence or two on why this is a good feature to add to PriCoSha
    d. A clear explanation of any changes to the database schema that are needed (including additional tables, additional attributes, or additional constraints)
    e. The queries (and any other SQL statements) used in your implementation of the feature
    f. A clear indication of where to find the application source code for the feature. Include plenty of comments and/or a few sentences explaining the code.
    g. One or more screenshots demonstrating the feature, showing how it appears in the browser; Also show the relevant data that's in the database when you execute this demonstration (before and after if the feature changes the data), either as screenshots or as text.

Note: *If your extra features require extensive modification to the database schema, please develop and test them as a separate branch of your project, so that you'll still be able to demonstrate the basic features with data we will supply.*

Here are a few ideas for extra use features:

9. **Optional (this can be one of your extra features:)** Include data that is relevant to the particular kind of content you are sharing (e.g. format data for photos, length and format for videos, location data for location-dependent content, …) and use this data in some interesting way.

10. **Optional (this can be one of your extra features:) Add comments:** Users can add comments about content that is visible to them. Decide what data about the comments should be stored and displayed and how you want to restrict visibility of comments.
11. **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.

12. **Optional (this can be one of your extra features:) Tagging a group:** Include an additional tagging mechanism, where multiple people in a content item can be tagged together, but which will only be visible if all of them agree to make it visible.
13. **Optional (this can be one of your extra features:)** Be creative. Propose something in part 3, check that it's approved, then implement it.

Teams, etc

You may work alone or in a team of up to 5 people. You may work on parts 1 and 2 alone then team up for parts 3, 4. We will post a form for teams to sign up. **Final team formation must be done by Nov. 14.** Note that each team member is expected to contribute roughly equally, each team member is the lead for two additional features, and each team member is responsible for understanding the entire system. There may be some quiz or exam questions about the project.

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 10% of the project grade and Part 4 counts for 65% of the project grade. At around the time when part 4 is due, we will distribute some data that you will use in a demo/test session to be scheduled with the TAs. There may also be a quiz or exam question(s) based on the project.