Database Systems — CSci 4380
Midterm Exam #1
September 26, 2019

SOLUTIONS

Question 1 (10*3=30 points). Write the following queries using relational algebra using the data model below. The model is described in detail in the back of the exam.

Users(userid, email, name, accountcreationdate,displayname, description, url, city, country, numfollowers)
Posts(postid, postdate, posttime, text, media, userid, numlikes)
Likes(postid, userid, dateliked)
PostHashtags(postid, hashtag, rank)
Hashtags(hashtag, numposts)
Comments(commentid, postid, userid, text, commentdate, commenttime, replyto_commentid)
Follows(userid, followed_userid, followdate)
Bookmarks(userid, postid, bookmarkdate)

(a) Return the `userid, email` and `displayname` of users who have at least one post with 300 or more likes (`numlikes`) and at least one post that is bookmarked by a different user other than themselves. The post with 300 likes and the bookmarked post may be the same post.

Solutions:

$$\begin{aligned} R1 &= \Pi_{userid,email,displayname}(Users \bowtie \sigma_{numlikes>300} Posts) \\ R2(userid1, postid1) &= \Pi_{postid,userid} Bookmarks \\ R3 &= Posts \bowtie_{postid=postid1 \text{ and } userid<>userid1} R2 \\ R4 &= \Pi_{userid,email,displayname}(Users \bowtie R3)) \\ Result &= R1 \cap R4 \end{aligned}$$

Alternatively, we can also use $Result = R1 \bowtie R2$.

(b) Suppose U is a user with `displayname 'warmandfuzzy'`. Return the `postid, text` of posts of users who are followed by U. The returned posts must have `postdate` after `'9/20/2019'` and must have the hashtag `'#northeasterner'`.

Solutions:

$$\begin{aligned} R1 &= \Pi_{userid}(\sigma_{displayname=warmandfuzzy} Users) \\ R2(userid) &= \Pi_{followed\_userid}(R1 \bowtie Follows) \\ R3 &= \sigma_{postdate>'9/20/2019'}(Posts) \bowtie R2 \\ Result &= \Pi_{postid,text}(\sigma_{hashtag=northeasterner}(PostHashtags) \bowtie R3) \end{aligned}$$

(c) Find posts with no likes, no comments and no bookmarks. Return the userid and postid for these posts.

Solutions:

$$\begin{aligned} R1 &= \Pi_{postid}(\sigma_{numlikes>0}(Posts)) \\ R2 &= \Pi_{postid}(Comments) \\ R3 &= \Pi_{postid}(Bookmarks) \\ R4 &= \Pi_{postid}(Posts) - (R1 \cup R2 \cup R3) \\ Result &= \Pi_{postid,userid}(R4 \bowtie Posts) \end{aligned}$$

Also possible to do various other versions, such as:

$$R4 = \Pi_{postid}(Posts) - (\Pi_{postid}(Likes) \cup \Pi_{postid}(Comments) \cup \Pi_{postid}(Bookmarks))$$

One common mistake is to use operations such as: $\Pi_{postid}(Posts \bowtie Bookmarks)$. Note that this finds posts with a bookmark by the same user who created the post, not posts with any bookmark as in $\Pi_{postid}(Bookmarks)$. This is not correct.

Another common mistake is to use an expression like: $\Pi_{postid,userid}(Posts) - \Pi_{postid,userid}(Bookmarks)$. This only excludes people who bookmarked their own post, not users who have posts with no bookmark.

Also incorrect to use intersection or join instead of union in the original solution because that returns users with posts that do not have all of likes/bookmarks/comments at the same time.

Question 2 (12*3=36 points). Suppose you are given the following relations to add to the data model in the appendix. Answer questions regarding each additional relation below.

(a) `Stories(userid, postdatetime, media, rankorder)`

Stories are temporary posts. Each user can have at most one story in the database. For each story, we store its `postdatetime`. A story can have multiple `media`, for each media in a specific story, there is a `rankorder`.

(1) Based on the above information, list all applicable functional dependencies.

(2) What are the key(s)?

(3) Is this relation in BCNF? 3NF? Explain why or why not.

(4) If it is not in BCNF, use BCNF decomposition to get relations that are in BCNF.

Solutions:

(1) Two functional dependencies:
`userid` → `postdatetime`
`userid media` → `rankorder`
Note: Some answers also included a third f.d.: userid rankorder → media. While this was not explicitly mentioned, it is not an unreasonable assumption that for someone's story, there is a unique rankorder (1,2,3, etc.). However, you still need the second fd to cover the statement: for each media in a specific story, there is a rankorder. If this third fd is used, there are two keys: userid, media and userid, rankorder. The remaining parts have the same answer.

(2) Key: userid, media

(3) Not in BCNF or 3NF as userid side is not a superkey and the righthand is not composed of prime attributes.

(4) (userid, postdatetime), userid → postdatetime, Key: userid
(userid, media, rankorder), userid media → rankorder, key: userid, media
Both relations are in BCNF.

(b) We store a new relation for tagging other people in the media of a specific post, and its associated functional dependencies are given below:

`PostTag(postid, userid, x, y, usertag)`

`postid x y` $\rightarrow$ `userid`
`userid` $\rightarrow$ `usertag`

Answer each of the following with yes/no and write a sentence to explain your answer.

(1) Can a user be tagged multiple times in a single post?

(2) Can we store two different `usertag` values for a specific `userid`?

(3) What are the keys? Is this relation in 3NF?

Solutions:

(1) Can a user be tagged multiple times in a single post?
YES, for different x,y values.

(2) Can we store two different `usertag` values for a specific `userid`?
No, usertag is implied by userid.

(3) What are the keys? Is this relation in 3NF?
Key: postid, x, y
Not in 3NF because the second f.d. does not have a superkey on the left or a prime attribute on the right.

(c) You are given the following relation for a new functionality we are adding to Instagram allowing people to sell products to each other.

`MarketPlace(buyeruserid, address, postid, price, purchasedate, questionid, questiontext, answer, questionuserid, tag)`

`postid` → `price, buyeruserid, purchasedate`
`buyeruserid` → `address`
`postid questionid` → `questiontext, answer, questionuserid`

(a) List all the keys.

(b) Convert this relation to 3NF using 3NF decomposition. Show your work. Note: You can shorten the attribute names for simplicity.

(c) For each resulting relation, show the key and state simply whether it is in BCNF or not.

Solutions:

Key: postid, questionid, tag

3NF decomposition:

`(postid, price, buyeruserid, purchasedate)`

postid → price, buyeruserid, purchasedate

Key: postid

`(buyeruserid, address)`

buyeruserid → address

Key: buyeruserid

`(postid, questionid, questiontext, answer, questionuserid)`

postid questionid → questiontext, answer, questionuserid

Key: postid, questionid

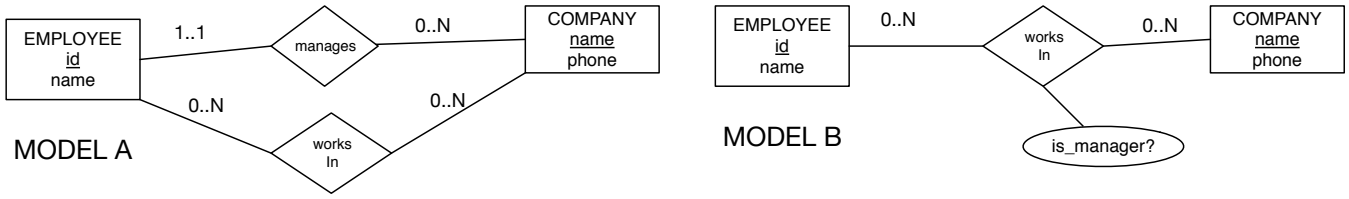
`(postid, questionid, tag)`

Key: all attributes

All relations are in BNCF.

Question 3 (6 points). You are given two alternate models in the following Entity-Relationship diagram. In which ways are these models similar or different? Give a short explanation.

Solutions:



In model A: the manager does not have to work in the company. In Model B, he/she has to.

In model A: a company can only have one manager, in model B, company can have multiple managers.

Question 4 (12 points). You are given relation: R(A,B,C,D,E) and $F = \{AB \to C, C \to D, D \to E\}$

and the decomposition: $R1(A, B, C), R2(C, D), R3(A, B, E)$.

Determine whether this decomposition is dependency preserving or not. Show your work.

Solutions: First, we project the given functional dependencies to the decomposed relations.

$R1(A, B, C) : \{AB \to C\}$

$R2(C, D) : \{C \to D\}$

$R3(A, B, E) : \{AB \to E\}$

The resulting set of functional dependencies is: $F2 = \{AB \to C, C \to D, AB \to E\}$

We note that $D \to E$ is not preserved in $F2$ since $D^+ = \{D\}$ according to $F2$. Hence, this decomposition is not dependency preserving.

Question 5 (16 points). Create an Entity-Relationship diagram for the following database, capturing all the requirements below precisely. Make sure you list all the relevant attributes, underlining the keys. For each relationship, mark the participation constraints clearly (one-to-one, one-to-many, or many-to-many). If you do not find a natural key for an entity, feel free to add an id attribute.

You are creating a database that will help raise money for different charitable or political causes.

In this database, you store users. Each user has an email, name, password, and a phone number. Emails are unique in the database. You also store causes. Each cause has an id, name, description, target dollar amount and deadline. Causes are posted by a single user. Causes may have multiple suggested donation amounts. Causes may be spin-offs of other causes.

The database stores multiple stories. Each story has a title, text, multiple photos each with a caption. Stories are written by one or more users and are for a specific cause.

The database also stores donations. For each donation, we store the name of the person making the donation (which is a simple string, not a user), credit card number, the amount, date/time of donation and a text comment. Each donation is for a specific cause.

Solutions:

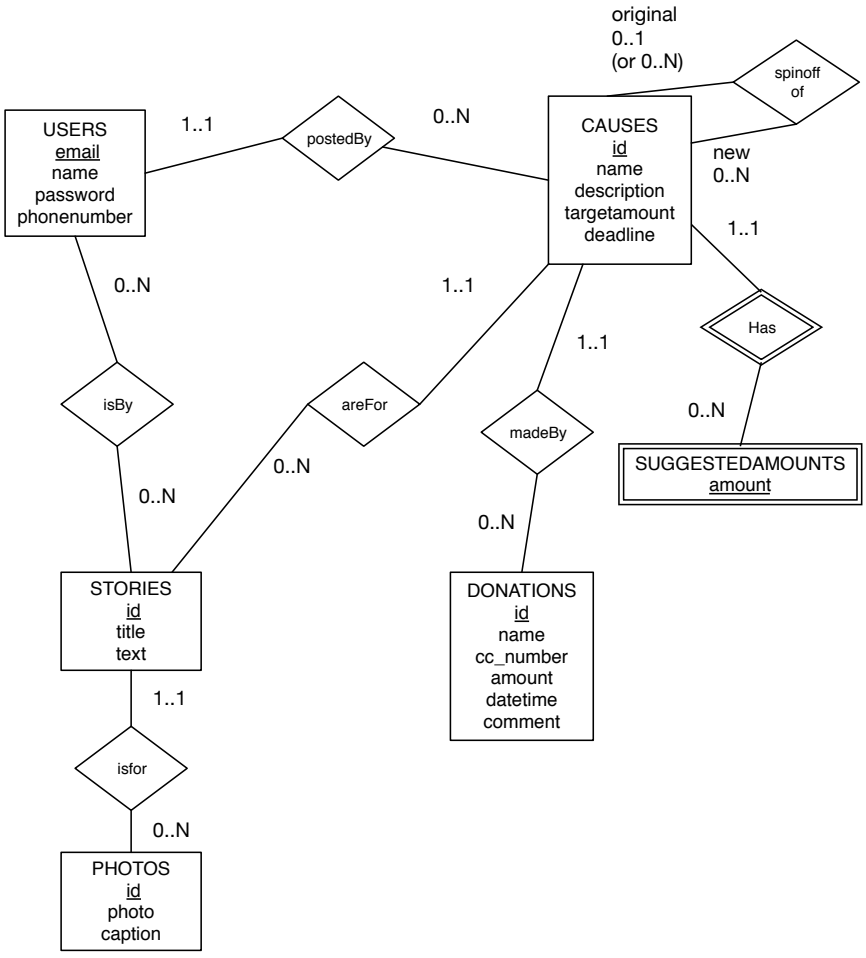


Figure 1: ER Diagram for Question 5

Data model to be used in Exam #1

This is a data model loosely based on data stored in Instagram. Note that a post in this model can only have a single media, photo or video. In this model, we have a number of derived attributes, each is a count of certain type of tuples. These attributes are redundant because theoretically they can be obtained from the remaining data. They are stored explicitly for performance reasons. You can assume these attributes all have accurate and up-to-date values and you can use them in queries.

Users(userid, email, name, accountcreationdate, displayname, description, url, city, country, numfollowers)

For each user, we store a number of attributes including a derived attribute `numfollowers` which is the count of followers for this user.

Posts(postid, postdate, posttime, text, media, userid, numlikes)

Each user potentially creates multiple posts. For each post, there is a date, time, a media (photo or video), the userid of the user who created the post and total number of likes (which is a derived attribute, obtained by counting the total likes.)

Likes(postid, userid, dateliked)

Each post may get likes. A user can like a post only once.

PostHashtags(postid, hashtag, rank)

Posts have many hashtags, each hashtag has a rank order.

Hashtags(hashtag, numposts)

For each hashtag used in the database, we store the total number of posts using that hashtag in the derived attribute `numposts`.

Comments(commentid, postid, userid, text, commentdate, commenttime, replyto_commentid)

We store comments made for a specific post (`postid` by a user (`userid`) with the given comment `text`, its date and time. A comment may be in reply to another comment; if so, we store the id of the comment that was in reply to (`replyto_commentid`).

Follows(userid, followed_userid, followdate)

Users follow other users. We store a tuple indicating that a user (`userid`) follows another user (`followed_userid`) which started on a specific date (`followdate`).

Bookmarks(userid, postid, bookmarkdate)

Users can bookmark posts. For each bookmarking action given by a specific user and post, we store the date of the bookmarking action.