Question-Answer Platform Database

Nevaan M. Perera
Khang D. Le

CS-345: Database Systems
12/21/2017

Functional Dependency Analysis

- We start our decomposition process by starting with a mega relation.
- Our mega relation is as follows…

Table (u_id, first, last, password, u_name, following, count_topics, position, num_q, num_a, friend_id, data_start, q_id, user_ans, date_ques, type_q, count_ans, count_tags, q_text, a_id, date_ans, count_upvotes, count_comment_ topic_id, date_following_topic, topic_id, exp_id, year_of_exp, name_exp, group_id, name_group, est_est_date, count_member, leader_id, date_enter, comment_id, u_comment, date_comment, tag_id, tags_name, q_booked, date_booked, reputation)

Our list of functional dependencies are as follows.

1)
U_id → u_id, first, last, password, u_name, following, count_topics, position, num_q, num_a, friend_id, data_start)

2)
Q_id → U_id
As Q_id determines u_id, it is implied that q_id determines everything u_id determines.

3)
a_id → q_id, user_answer, date_answer, count_upvotes

4)
u_id →→ topic_id

5)
topic_id → topic_name

6)
q_id →→ tags_id

7)
tag_id → tag_name

8)
u_id →→ expertise_id

9)
expertise_id → expertise_name

10)
u_id →→ organization_id

11)
org_id → group_id, name_group, est_date

12)
comment_id → a_id, user_comment, date_comment

13)
num_a → reputation

Tables

- All tables are in alphabetical order.
- Only non-trivial functional dependencies are included.

1)
Answer (a_id, q_id, u_id, date_ans, count_upvotes, count_comments)

- A → B where,
  - $A \subseteq$ {a_id, q_id, u_id, date_ans, count_upvotes, count_comments}
  - $A \cap$ {a_id} = {a_id}
  - $B \subseteq$ {a_id, q_id, u_id, date_ans, count_upvotes, count_comments}
- Since a_id uniquely identifies all rows in the table, A is a superkey, so BCNF

2)
Answer Ruputation (num_a)

- There is only one column in this table; therefore, it is instrincly in BCNF.

3)
Comments (comment_id, a_id, u_comment, date_comment)

A → B where,

A ⊆ {comment_id, a_id, u_comment, date_comment}

A ∩ {comment_id} = {comment_id}

B ⊆ {comment_id, a_id, u_comment, date_comment}

- All functional dependencies contain a superkey or are trivial, so this table is in BCNF.

4)
Expertise (expertise_id, u_id)

{expertise_id, expertise_type} → {expertise_id, expertise_type}

{expertise_id, expertise_type} is the key for this table, therefore BCNF

- Tables with two attributes are in BCNF

5)
Expertise_type (expertise_id, name)

expertise_id → B or {expertise_id, name} → B

where – B ⊆ {expertise_id, name}

- These are both superkeys because they uniquely identify a tuple, so BCNF

6)
Following (u_id, follower_id, start_date)

A → B where the following hold:

A ⊆ {u_id, follower_id, start_date}

A ∩ {u_id, follower_id} = {u_id, follower_id }

B ⊆ {u_id, follower_id, start_date}

- All functional dependencies contain a superkey or are trivial, so this table is in BCNF.

7)
Group_Membership (org_id, u_id)

{org_id, u_id} → {org_id, u_id}

{org_id, u_id} is the key for this table, therefore BCNF

Tables with two attributes are in BCNF

All functional dependencies contain a superkey or are trivial, so this table is in BCNF.

8)
Organization (org_id, name, est_date)

A ⊆ {org_id, name, est_date}

A ∩ {org_id} = {org_id}

B ⊆ {org_id, name, est_date}

- All functional dependencies contain a superkey or are trivial, so this table is in BCNF.

9)
Question (q_id, u_id, date_ques, type, question_text)

A → B where,

A ⊆ {q_id, u_id, date_ques, type, question_text}

A ∩ {q_id} = {q_id}

B ⊆ {q_id, u_id, date_ques, type, question_text}

- Since q_id uniquely identifies all rows in the table, A is a superkey, so BCNF

10)
Question_tag (tag_id, q_id)

{tag_id, q_id} → {tag_id, q_id}

{tag_id, q_id} is the key for this table, therefore BCNF

Tables with two attributes are in BCNF

11)
Tag (tag_id, name)

code → B or {tag_id, name} → B where,

– B ⊆ {tag_id, name}

These are superkeys, so BCNF

12)
Topic (topic_id, name)

code → B or {topic_id, name} → B where,

– B ⊆ {topic_id, name}

These are superkeys, so BCNF

13) User_topic (u_id, topic_id, date_following)

A → B where the following hold:

A ⊆ {u_id, topic_id, date_following}

A ∩ {u_id, topic_id} = {u_id, topic_id}

B ⊆ {u_id, topic_id, date_following}

- All functional dependencies contain a superkey or are trivial, so this table is in BCNF.

14) User (u_id, first, last, password, username, position, sex)

A → B where,

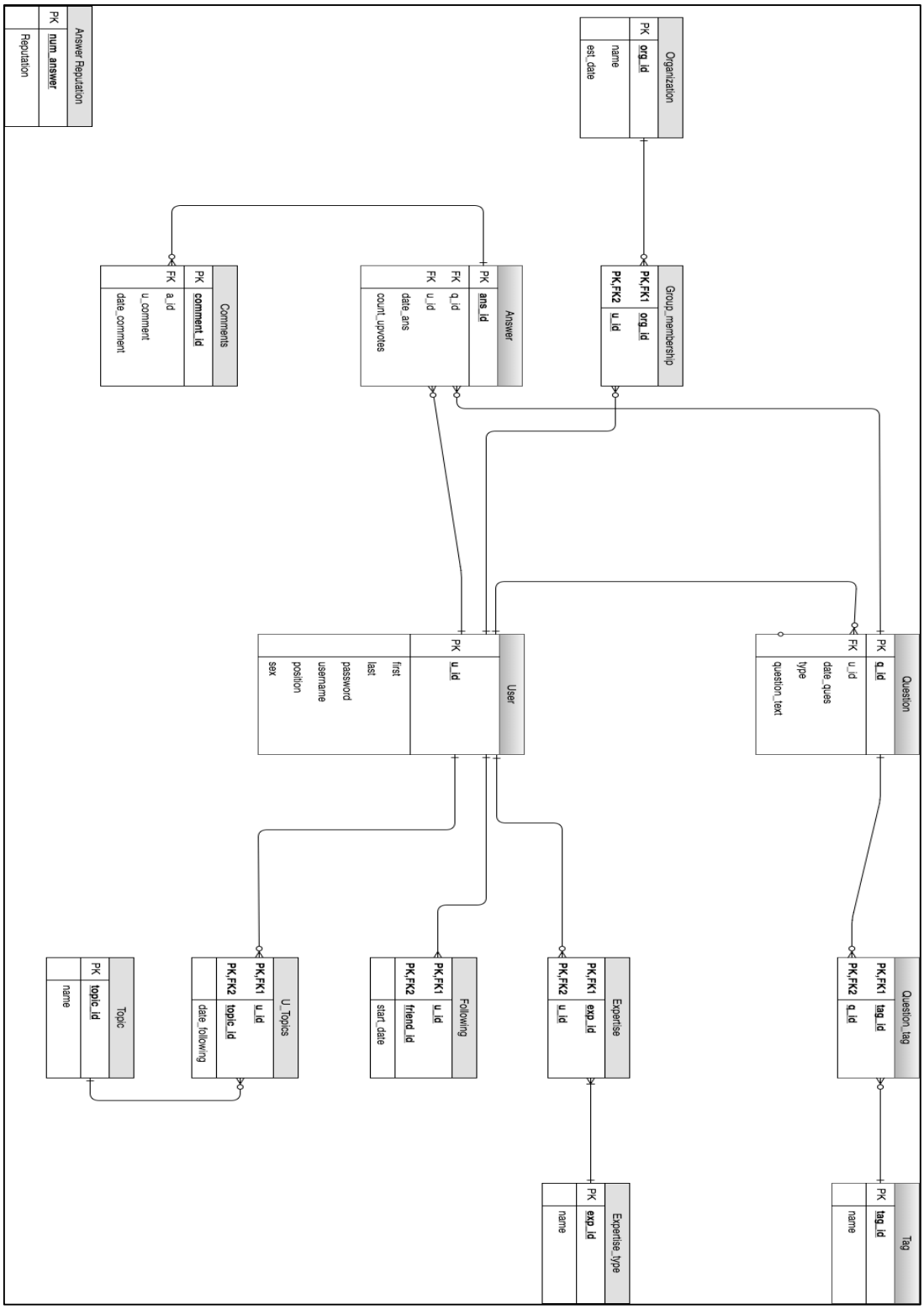
A ⊆ {u_id, first, last, password, username, position, sex}

A ∩ {u_id} = {u_id}

B ⊆ {u_id, first, last, password, username, position, sex}

- Since u_id uniquely identifies all rows in the table, A is a superkey, so BCNF

ER Diagram


Answer Reputation
PK num_answer
Reputation
Organization
PK org_id
name
est_date
Comments
PK comment_id
FK a_id
u_comment
date_comment
Answer
PK ans_id
FK q_id
FK u_id
date_ans
count_upvotes
Group_membership
PK,FK1 org_id
PK,FK2 u_id
User
PK u_id
first
last
password
username
position
sex
Question
PK q_id
FK u_id
date_ques
type
question_text
Topic
PK topic_id
name
U_Topics
PK,FK1 u_id
PK,FK2 topic_id
date_following
Following
PK,FK1 u_id
PK,FK2 friend_id
start_date
Expertise
PK,FK1 exp_id
PK,FK2 u_id
Question_tag
PK,FK1 tag_id
PK,FK2 q_id
Expertise_type
PK exp_id
name
Tag
PK tag_id
name

CREATING OUR DATABASE

Create Tables

- Provided below are some examples how we created our database tables.
- Not all of the tables are provided below; the entire query could be found in the .sql file in our dropbox

```
--1
Drop table if exists Users;
Create table Users (
        u_id varchar(50) not null,
        first varchar not null,
        last varchar not null,
        password varchar not null,
        username varchar not null,
        sex varchar not null,
        position varchar not null,
        primary key(u_id)
        );

--2
Drop table if exists Question;
Create table Question (
        q_id varchar not null,
        u_id varchar(50) not null,
        date_ques varchar not null,
        type_q varchar not null
                check (type_q in ('Multiple Choice', 'Poll', 'Text')),
        text varchar not null,
        primary key(q_id),
        foreign key(u_id) references Users
        );

--3 (topics of interests of the Users)
Drop table if exists U_Topics;
Create table U_Topics (
        u_id varchar(50) not null,
        topic_id varchar not null,
        date_following varchar not null,
        primary key(u_id, topic_id),
        foreign key(u_id) references Users(u_id),
        foreign key(topic_id) references Topics(topic_id)
        );
```

Populating Our Database with Data (Insert Queries)

- Again, this is not an exhaustive list of all the queries; The entire list is provided in an .sql file in our dropbox.

Insert into users

```
insert into users values
        ('kdle15@stlawu.edu','Khang','Le','123',
        'khang_le', 'male', 'students');

insert into users values
        ('nmpere15@stlawu.edu','Nevaan','Perera','1244',
        'nevaan_champ', 'male', 'students');
```

insert into following

```
insert into following values
        ('kdle15@stlawu.edu', 'ehar@stlawu.edu', '10/10/2017');

insert into following values
        ('kdle15@stlawu.edu', 'nmpere15@stlawu.edu', '20/10/2017');
```

topics

```
insert into topics values
        ('3', 'HarryPotter');

insert into topics values
        ('4', 'Hockey');
```

Organization

```
insert into organization values ('1', 'CS-140', '09/17/2017');

insert into organization values ('2', 'IT', '09/02/2015');
```

Question Tags

```
insert into Question_tags values ('1', '2');

insert into Question_tags values ('1', '4');

insert into Question_tags values ('2', '3');
```

Delete tuples

- When the user wants to delete a question, answer or comment he made.
- Also, if the user wants to unfollow other users or topics.
- To delete tuples we add a constraint to all our tables, so that a referenced foreign key is deleted at the same time.
- Provided below are some examples.

```
Create table Question (
        q_id varchar not null,
        u_id varchar(50) not null,
        date_ques varchar not null,
        type_q varchar not null
                check (type_q in ('Multiple Choice', 'Poll', 'Text')),
        text varchar not null,
        primary key(q_id),
        constraint u_id
        foreign key(u_id) references Users
        on delete cascade
        );

Create table Answer (
        q_id varchar not null,
        a_id varchar not null,
        u_id varchar(50) not null,
        date_ans varchar not null,
        count_upvotes int not null check (count_upvotes >= 0),
        primary key(a_id),
        constraint q_id
        foreign key(q_id) references Question on Delete cascade,
        foreign key(u_id) references Users(u_id)
        );

Create table Comments (
        a_id varchar not null,
        comment_id varchar not null,
        u_id varchar (50) not null,
        date_comment varchar not null,
        primary key(comment_id),
        constraint a_id
        foreign key(a_id) references Answer on delete cascade,
        foreign key(u_id) references Users(u_id)
        );
```

- As shown above, when a user deletes a question. The answers, and comments to those answers get deleted as well.

Example Queries

SELECT QUERIES

- When the user wants to retrieve information based on certain criteria

1) Find the male-student who has asked the most number of questions

```
select first, last from users
      where u_id = (Select u_id from (select count (*) as num_q, u_id
      from question natural join users
      where sex = 'male' and position = 'students' group by u_id order by num_q DESC
Limit 1) as foo);
```

2) Get all the questions asked from Nevaan about about academic experience and womens resource center

```
with x as
      (select q_id from question_tags natural join tags where tags_name = 'woman
      resource center' or tags_name = 'Academic')
select * from question natural join x where u_id = 'nmpere15@stlawu.edu';
```

3) List (in descending order) the number of individuals in each position (students, faculty and staff) who have asked questions; list only of the group consists of more than 2 individuals.

```
select position, count (*) from users natural join question group by position having
count(*)>1;
```

4) Get the most up-voted answer based on the question

```
select first, last, count_upvotes from users
      natural join (select a_id, u_id, count_upvotes from answer where q_id = '9'
      order by count_upvotes DESC) as foo;
```