COL362 - Project - Milestone 1

DataBoys - Tanish Gupta, Rishi Jain, Daksh Khandelwal

20th March 2023

1 Introduction

Our project is about building a clone of a subset of StackExchange. For Milestone 1 we have built the basic setup of our database. We defined the underlying schema, key constraints and the associated ER-Diagrams. We computed the functional dependecies, and normalized the relations in our database to BCNF. The major chunk of the milestone was to study the dataset, and derive meanings and logical relationships between the tables.

The dataset has been taken from here, which is the official dump of stackoverflow. The XML files were converted to CSV, thanks to this repository, so that they could be easily read and imported into the Postgresql database. Functional dependencies and BCNF was studied from lecture slides and db-book.

Now we are set up with the basic structure and ready to work for the next stage. In Milestone-2, we aim to write the SQL queries that will drive our application and optimize them to obtain results faster.

2 Functional Dependencies

For the Badges relation, we have the following basic set of non-trivial FDs:

- ID → UserId
- ID → Class
- ID → Name
- ID → TagBased
- ID → Date

For the Users relation, we have the following basic set of non-trivial FDs:

- ID → Reputation
- ID → CreationDate
- ID → DisplayName
- ID → LastAccessDate
- ID → WebsiteUrl
- ID → Location
- ID → AboutMe
- ID → Views
- ID → UpVotes
- ID → DownVotes
- ID → AccountId

For the Comments relation, we have the following basic set of non-trivial FDs:

- ID $\rightarrow$ PostId
- ID $\rightarrow$ Score
- ID $\rightarrow$ Text
- ID $\rightarrow$ CreationDate
- ID $\rightarrow$ UserId
- ID $\rightarrow$ ContentLicense
- ID $\rightarrow$ UserDisplayName

For the Post History relation, we have the following basic set of non-trivial FDs:

- ID $\rightarrow$ PostHistoryTypeId
- ID $\rightarrow$ PostId
- ID $\rightarrow$ RevisionGUID
- ID $\rightarrow$ CreationDate
- ID $\rightarrow$ UserId
- ID $\rightarrow$ Text
- ID $\rightarrow$ ContentLicense
- ID $\rightarrow$ Comment
- ID $\rightarrow$ UserDisplayName

For the Posts relation, we have the following basic set of non-trivial FDs:

- ID $\rightarrow$ PostTypeId
- ID $\rightarrow$ AcceptedAnswerId
- ID $\rightarrow$ CreationDate
- ID $\rightarrow$ Score
- ID $\rightarrow$ ViewCount
- ID $\rightarrow$ Body
- ID $\rightarrow$ OwnerUserId
- ID $\rightarrow$ LastEditorUserId
- ID $\rightarrow$ LastEditDate
- ID $\rightarrow$ LastActivityDate
- ID $\rightarrow$ Title
- ID $\rightarrow$ Tags
- ID $\rightarrow$ AnswerCount
- ID $\rightarrow$ CommentCount
- ID $\rightarrow$ ContentLicense
- ID $\rightarrow$ FavoriteCount

- ID $\rightarrow$ ParentId
- ID $\rightarrow$ OwnerDisplayName
- ID $\rightarrow$ ClosedDate
- ID $\rightarrow$ LastEditorDisplayName
- ID $\rightarrow$ CommunityOwnedDate

For the Votes relation, we have the following basic set of non-trivial FDs:

- ID $\rightarrow$ PostId
- ID $\rightarrow$ VoteTypeId
- ID $\rightarrow$ CreationDate
- ID $\rightarrow$ UserId
- ID $\rightarrow$ BountyAmount

For the Tags relation, we have the following basic set of non-trivial FDs:

- ID $\rightarrow$ TagName
- ID $\rightarrow$ Count
- ID $\rightarrow$ ExcerptPostId
- ID $\rightarrow$ WikiPostId

For the Post Links relation, we have the following basic set of non-trivial FDs:

- ID $\rightarrow$ CreationDate
- ID $\rightarrow$ PostId
- ID $\rightarrow$ RelatedPostId
- ID $\rightarrow$ LinkTypeId

3 FD Preserving Normalizations

Boyce Codd Normal Form (BCNF): A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F^+ (closure of FDs) of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, atleast one of the following holds:

- $\alpha \rightarrow \beta$ is trivial i.e. $\beta \subseteq \alpha$
- α is a superkey for R

We claim that the submitted dataset is in BCNF form. Let's prove the claim for one of the relations say `Badges`.

3.1 Badges Table

For the Badges relation, we have the following basic set of non-trivial FDs:

- ID $\rightarrow$ UserId
- ID $\rightarrow$ Class
- ID $\rightarrow$ Name
- ID $\rightarrow$ TagBased
- ID $\rightarrow$ Date

There are other trivial dependencies that hold here.

Note that Id is a superkey here, since Id can functionally determine all other attributes of the relation. So, the above FDs satisfy the conditions for BCNF.

Now consider any non trivial dependency $\alpha \to \beta$. Here, $\beta \subseteq \alpha$. The closure may include new FDs only by one of the following rules:

- Reflexive Rule: All FDs formed using reflexive rule satisfy the first condition of BCNF.
- Augmentation Rule: If the original FD is a trivial FD to which this rule is applied, then the new FD is also trivial (i.e. if $\alpha \to \beta$ is trivial, then so is $\gamma\alpha \to \gamma\beta$ since $\gamma\beta \subseteq \gamma\alpha$ follows from $\beta \subseteq \alpha$). In the other case, if original FD is not a trivial FD, then it must contain ID on its LHS i.e. ID $\subseteq \alpha$ (This can be proven by induction on the length of derivation of FD, since we have only 3 rules for FD closure). Thus, α is a superkey. So, both cases satisfy BCNF conditions.
- Transitivity Rule: Again, we can proceed on case by case basis. If both the original FDs are trivial, then the transitive FD is also trivial. If either of them is non-trivial, then that particular FD should have ID in LHS (Again, this can be proven by induction), and hence, α will be a superkey. Thus, BCNF conditions are satisfied.

Therefore, every FD satisfies one of the BCNF condition, and hence this relation is in BCNF.

Another way to show that all the FDs satsify the BCNF conditions is as follows:

We can compute the closure of FDs using the closure for each subset of attributes i.e. we can use the following algorithm to compute the α^+ for any $\alpha \subseteq R$

```
result := $\alpha$;
repeat
    for each functional dependency $\beta \to \gamma$ in F do
        if $\beta \subseteq$ result then result := result $\cup\ \gamma$;
until (result does not change)
```

Now, there are 2 cases for α:

- If ID $\subseteq \alpha$, then it easily follows, since all FDs formed using these subsets will have ID on LHS, and since ID is a superkey, the 2nd condition of BCNF is satsified.
- For the other case, if ID $\not\subseteq \alpha$, then, since all the non-trivial functional dependencies have ID on their LHS (can be proven by induction for the ones not listed above), the if condition in the for loop will not be true for any of the non-trivial dependencies. The only possible unions to the result can be through trivial FDs (which do not contain ID on LHS since otherwise $\beta \not\subseteq$ result), and hence any new FDs will also be trivial. Thus, 1st condition of BCNF will be satisfied.

Hence, the relation is in BCNF.

Since in all the relations, the non-trivial FDs are of similar form (ID - primary key on LHS), it follows that all the relations are in BCNF. Similar arguments may follow for their proof as well.

4 ER Diagram

A few notations in order to understand the ER diagram attached:

- Entity sets are represented by rectangles and Relationship sets are represented with diamonds.
- `PK` stands for Primary Key and `FK` stands for Foreign key.
- We indicate cardinality constraints on a relationship by drawing either a directed line ($\to$) or an undirected line (—) between the relationship set and the entity set in question.

- One-to-one: We draw a directed line from the relationship set to both entity sets.
- One-to-many: We draw a directed line from the relationship set to the “one” side of the relationship.
- Many-to-one: We draw a directed line from the relationship set to the “one” side of the relationship.
- Many-to-many: We draw an undirected line from the relationship set to both entity sets.

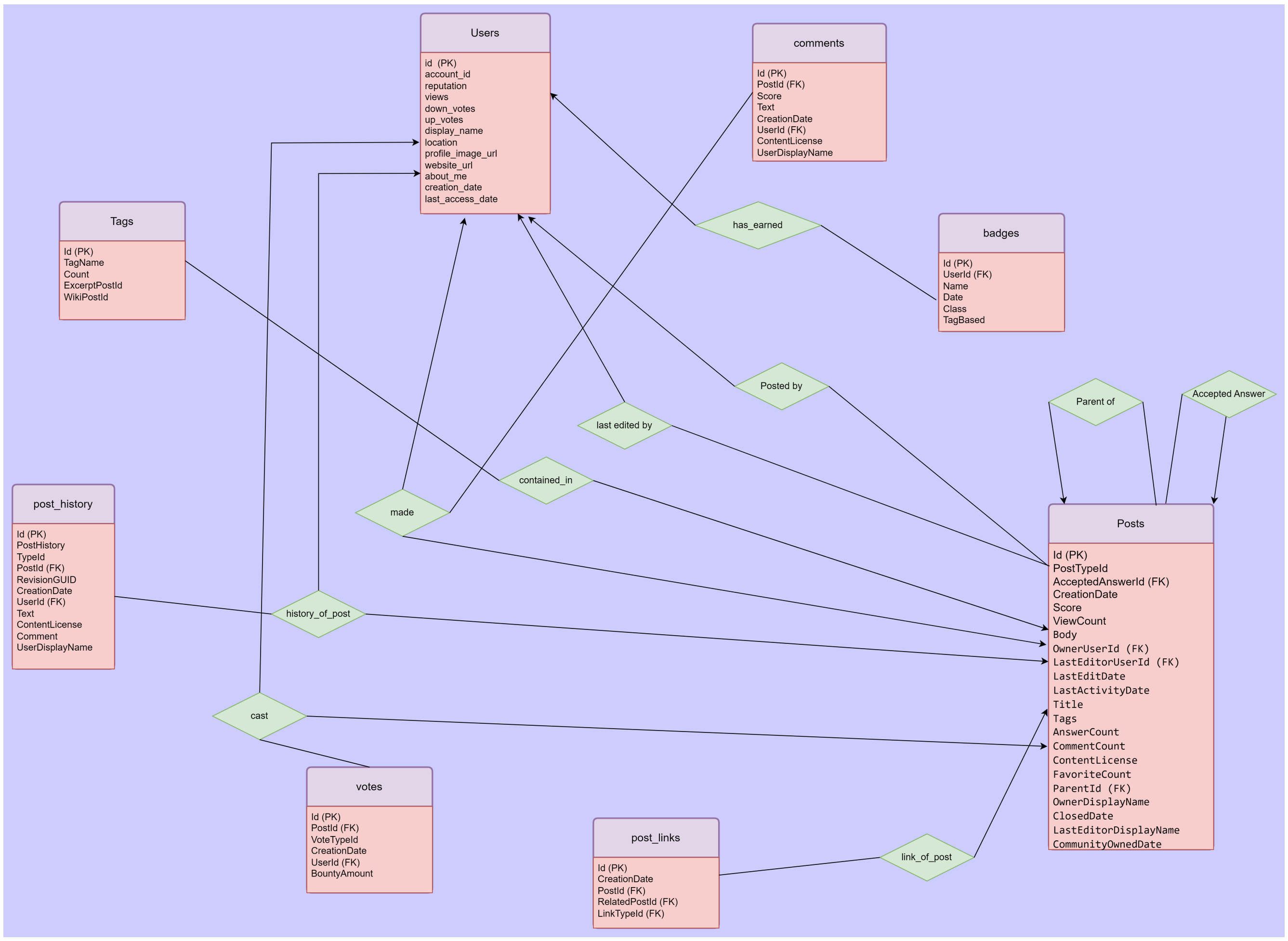
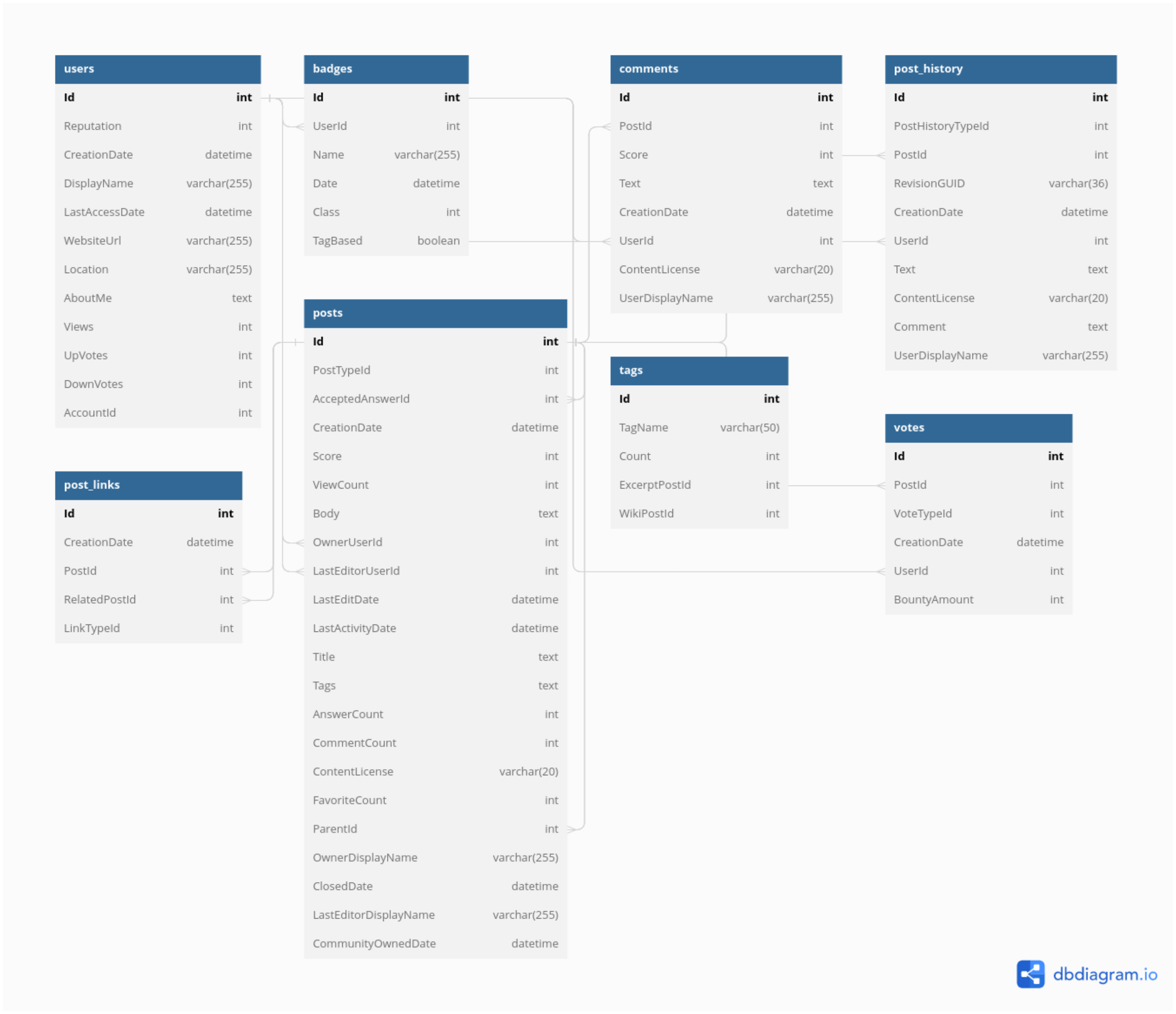


Figure 1: ER Diagram

w

5 Relational Schema

The Relational Schema is as follows :-



users	
Id	int
Reputation	int
CreationDate	datetime
DisplayName	varchar(255)
LastAccessDate	datetime
WebsiteUrl	varchar(255)
Location	varchar(255)
AboutMe	text
Views	int
UpVotes	int
DownVotes	int
AccountId	int

badges	
Id	int
UserId	int
Name	varchar(255)
Date	datetime
Class	int
TagBased	boolean

comments	
Id	int
PostId	int
Score	int
Text	text
CreationDate	datetime
UserId	int
ContentLicense	varchar(20)
UserDisplayName	varchar(255)

post_history	
Id	int
PostHistoryTypeId	int
PostId	int
RevisionGUID	varchar(36)
CreationDate	datetime
UserId	int
Text	text
ContentLicense	varchar(20)
Comment	text
UserDisplayName	varchar(255)

posts	
Id	int
PostTypeId	int
AcceptedAnswerId	int
CreationDate	datetime
Score	int
ViewCount	int
Body	text
OwnerUserId	int
LastEditorUserId	int
LastEditDate	datetime
LastActivityDate	datetime
Title	text
Tags	text
AnswerCount	int
CommentCount	int
ContentLicense	varchar(20)
FavoriteCount	int
ParentId	int
OwnerDisplayName	varchar(255)
ClosedDate	datetime
LastEditorDisplayName	varchar(255)
CommunityOwnedDate	datetime

tags	
Id	int
TagName	varchar(50)
Count	int
ExcerptPostId	int
WikiPostId	int

post_links	
Id	int
CreationDate	datetime
PostId	int
RelatedPostId	int
LinkTypeId	int

votes	
Id	int
PostId	int
VoteTypeId	int
CreationDate	datetime
UserId	int
BountyAmount	int

Figure 2: Relational Schema

6 Code snippet

The following code was used to generate the relational schema :-

```sql
BEGIN TRANSACTION;

DROP TABLE IF EXISTS users;
CREATE TABLE users (
   Id INT NOT NULL PRIMARY KEY,
   AccountId INT ,
   Reputation INT NOT NULL,
   Views INT NOT NULL,
   DownVotes INT NOT NULL,
   UpVotes INT NOT NULL,
   DisplayName VARCHAR(255) NOT NULL,
   Location VARCHAR(255),
   WebsiteUrl VARCHAR(255),
  AboutMe TEXT,
  CreationDate TIMESTAMP NOT NULL,
  LastAccessDate TIMESTAMP NOT NULL
);

DROP TABLE IF EXISTS badges;
CREATE TABLE badges (
  Id INT NOT NULL PRIMARY KEY,
  UserId INT NOT NULL REFERENCES users(Id),
  Class INT NOT NULL,
  Name VARCHAR(255) NOT NULL,
  TagBased BOOLEAN NOT NULL,
  Date TIMESTAMP NOT NULL
);


DROP TABLE IF EXISTS comments;
CREATE TABLE comments (
  Id INT NOT NULL PRIMARY KEY,
  PostId INT NOT NULL REFERENCES posts(Id),
  UserId INT REFERENCES users(Id),
  --If userid doesn't exist, then the user commented as a guest.
  --She needs name and email, but email is not included in public data for privacy reasons.
  Score INT NOT NULL,
  ContentLicense VARCHAR(20),
  UserDisplayName VARCHAR(255) ,
  Text TEXT NOT NULL,
  CreationDate TIMESTAMP NOT NULL
);

DROP TABLE IF EXISTS post_history;
CREATE TABLE post_history (
  Id INT NOT NULL PRIMARY KEY,
  PostId INT NOT NULL REFERENCES posts(Id),
  UserId INT REFERENCES users(Id),
  PostHistoryTypeId INT NOT NULL,
  UserDisplayName VARCHAR(255) ,
  ContentLicense VARCHAR(20),
  RevisionGUID VARCHAR(36) NOT NULL,
  Text TEXT ,
  Comment TEXT,
  CreationDate TIMESTAMP NOT NULL

);

DROP TABLE IF EXISTS post_links;
CREATE TABLE post_links (
  Id INT NOT NULL PRIMARY KEY,
  RelatedPostId INT NOT NULL REFERENCES posts(Id),
  PostId INT NOT NULL REFERENCES posts(Id),
  LinkTypeId INT NOT NULL,
  CreationDate TIMESTAMP NOT NULL

);

DROP TABLE IF EXISTS posts;
CREATE TABLE posts (
  Id INTEGER NOT NULL PRIMARY KEY,
  OwnerUserId INTEGER REFERENCES users(Id),
  LastEditorUserId INTEGER REFERENCES users(Id),
  PostTypeId INTEGER NOT NULL,
  AcceptedAnswerId INTEGER REFERENCES posts(Id),
  Score INTEGER NOT NULL,
  ParentId INTEGER REFERENCES posts(Id),
  ViewCount INTEGER,
  AnswerCount INTEGER,
  CommentCount INTEGER,
  OwnerDisplayName TEXT,
  LastEditorDisplayName TEXT,
  Title TEXT,
  Tags TEXT,
  ContentLicense TEXT,
  Body TEXT,
  FavoriteCount INTEGER,
  CreationDate TIMESTAMP NOT NULL,
  CommunityOwnedDate TIMESTAMP,
  ClosedDate TIMESTAMP,
  LastEditDate TIMESTAMP,
  LastActivityDate TIMESTAMP NOT NULL
);

DROP TABLE IF EXISTS tags;
CREATE TABLE tags (
    Id INTEGER NOT NULL PRIMARY KEY,
    ExcerptPostId INTEGER ,
    WikiPostId INTEGER ,
    TagName VARCHAR(50) NOT NULL,
    Count INTEGER NOT NULL
);

DROP TABLE IF EXISTS votes;
CREATE TABLE votes (
    Id INT NOT NULL PRIMARY KEY,
    UserId INT  REFERENCES users(Id),
    PostId INT NOT NULL REFERENCES posts(Id),
    VoteTypeId INT NOT NULL,
    BountyAmount INT ,
    CreationDate TIMESTAMP NOT NULL
);

END TRANSACTION;
```