University of British Columbia, Vancouver

Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2
Date: October 20th, 2023
Group Number: 68

Name	Student Number	CS ID	Email
Liam Buchan	75087510	s0d7t	liamjamesbuchan@gmail.com
Eric Lee	44415206	g4g9q	elee1820@gmail.com
Sammi Pau	56010564	q7d9h	sammi.pau@live.com

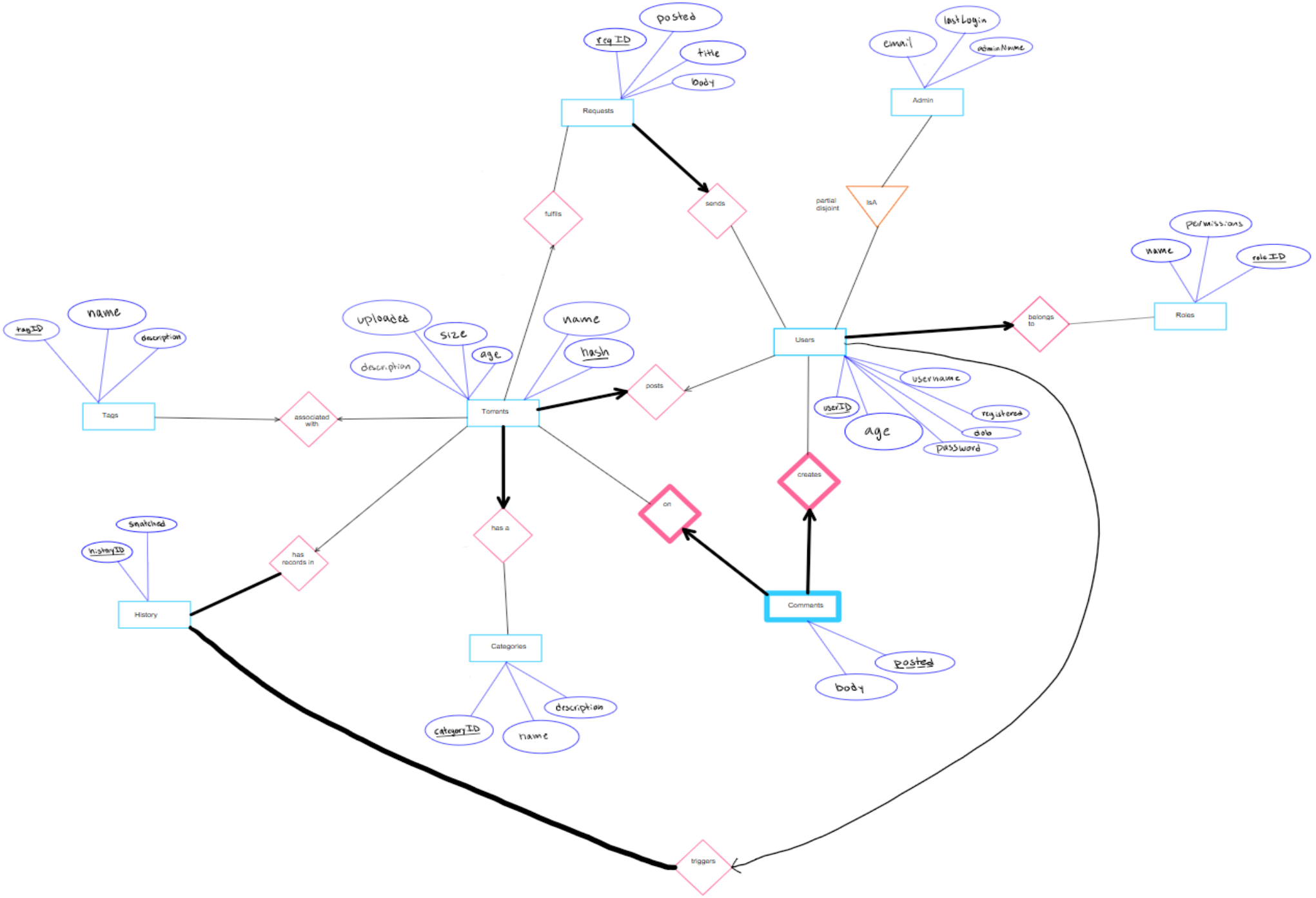
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Project Summary

Our project goal is to create a torrent site to help users index and archive data through the bittorrent protocol. To achieve this, the service allows users to upload bittorrent files which point to the data itself along with metadata and tagging that allows other users to easily find the files. For example, a user could upload an Arch Linux ISO torrent, tag it with the “Arch” tag and place it within the “Linux ISO” category.

ER Diagram

There are a few changes in our ER diagram from milestone 1. We added a Tag and History entity. Some of the weak entities were changed to normal entities, due to being falsely classified as weak. The isA relationship has been specified as partial disjoint. Lastly, the Snatch entity was removed as it is not necessary. The updated ER diagram is attached below.


posted
req ID
title
body
Requests
fulfils
sends
lastLogin
email
adminName
Admin
partial
disjoint
IsA
permissions
name
roleID
Roles
belongs
to
name
tagID
description
Tags
associated
with
uploaded
size
age
description
name
hash
Torrents
posts
Users
username
userID
registered
dob
age
password
creates
on
has a
has
records in
snatched
historyID
History
Comments
posted
body
Categories
categoryID
name
description
triggers

Schema

For the following schema, primary keys are underlined and foreign keys are bolded.

Roles(roleID: integer, permissions: bit(10), name: varchar)

- roleID and name are unique
- name is a CK
- All attributes are not null

Tags(tagID: integer, name: varchar, description: varchar)

- tagID and name unique
- name is a CK
- tagID and name are not null

Categories(categoryID: integer, name: varchar, description: varchar)

- categoryID and name are unique
- name is a CK
- categoryID and name are not null

Users(userID: integer, username: varchar(32), password: varchar, registered: timestamp, dob: date, age: interval, **roleID: integer**)

- userID and username are unique
- username is a CK
- All attributes are not null

AdminUsers(userID: integer, username: varchar, password: varchar, registered: timestamp, dob: date, age: interval, email: varchar(254), lastLogin: timestamp, adminName: varchar, **roleID: integer**)

- userID and username are unique
- username is a CK
- All attributes are not null

Torrents(hash: varchar(32), name: varchar(100), size: integer, uploaded: timestamp, description: varchar, age: interval, **categoryID: integer**, **userID: integer**)

- hash is unique
- hash, name, size, uploaded, categoryID, userID are not null

Requests(reqID: integer, posted: timestamp, title: varchar, body: varchar, **hash: varchar(32)**, **userID: integer**)

- reqID is unique
- reqID, posted, title, body, and userID are not null

History(historyID: integer, snatched: timestamp, **hash: varchar(32), userID: integer**)

- historyID is unique
- All attributes are not null

Comments(posted: timestamp, **hash: varchar(32)**, **userID: integer,** body: varchar)

- All attributes are not null

Functional Dependencies & Normalization

(roles)
roleID → permissions, name
name → roleID, permissions

The Role table is already in BCNF, because roleID and name are superkeys.
PK: roleID
CK: name

(tags)
tagID → name, description
name → tagID, description

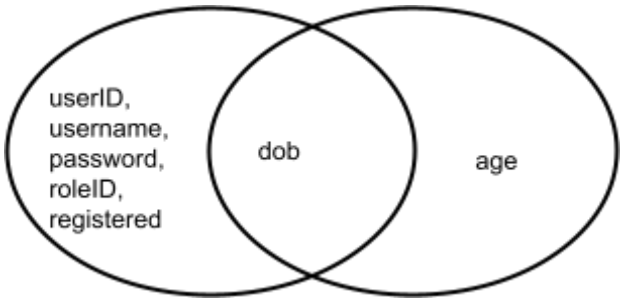
The Tag table is already in BCNF, because tagID and name are superkeys.
PK: tagID
CK: name

(categories)
categoryID → name, description
name → categoryID, description

The Categories table is already in BCNF, because categoryID and name are superkeys.
PK: categoryID
CK: name

(users)
userID → username, password, registered, dob, age, roleID
username → userID, password, registered, dob, age, roleID
dob→ age

The Users table is not in BCNF or 3NF, because dateOfBirth is not a superkey. So, we decompose.
Take dob→ age,

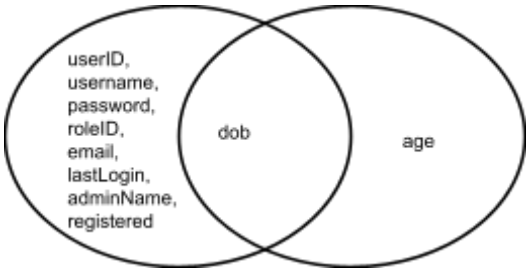


UsersAge(dob, age), Users(dob, userID, username, password, registrationDate, roleID)

UsersAge
PK: dob
Users
PK: userID
CK: username
FK: dob, roleID

(adminUsers)
userID → username, password, registered, dob, age, email, lastLogin, adminName, roleID
username → userID, password, registered, dob, age, email, lastLogin, adminName, roleID
dob→ age

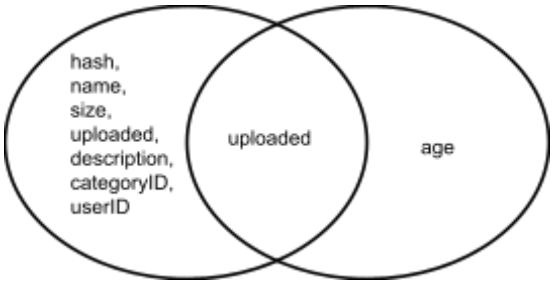
The AdminUser table is not in BCNF or 3NF, because dateOfBirth is not a superkey. So, we decompose.
Take dob → age,



AdminUsersAge(dob, age), AdminUsers(dob, userID, username, password, registered, email, lastLogin, adminName, roleID)
AdminUsersAge
PK: dob
AdminUsers
PK: userID
CK: username
FK: dob, roleID

(torrents)
hash → name, size, uploaded, description, age, categoryID, userID, tagID
uploaded → age

The Torrent table is not in BCNF or 3NF, because uploaded is not a superkey. So, we decompose.
Take uploaded → age,



TorrentsAge(uploaded, age), Torrents(uploaded, hash, name, size, uploaded, description, categoryID, userID)
TorrentsAge
PK: uploaded
Torrents
PK: hash
FK: uploaded, categoryID, userID

(requests)
reqID → posted, title, body, hash, userID

The Request table is already in BCNF, because reqID is a superkey.
PK: reqID

(history)
historyID → snatched, hash, userID

The History table is already in BCNF, because historyID is a superkey.
PK: historyID

(comments)
posted, userID, hash → body

The Comment table is already in BCNF, because (posted, userID, hash) is a superkey.
PK: (posted, userID, hash)

SQL DDLs

The following DLLs were made using PostgreSQL v11+

```
-- roles
create table Roles (
   roleID serial primary key,
   name varchar unique not null,
   permissions bit(10) not null
);

-- tags
create table Tags (
   tagID serial primary key,
   name varchar unique not null,
   description varchar
);
```

```sql
--  categories
create table Categories (
   categoryID serial primary key,
   name varchar unique not null,
   description varchar
);

-- users
create table UsersAge (
   dob date primary key,
   age interval not null
);

create table Users (
   userID serial primary key,
   username varchar(32) unique not null,
   password varchar not null,
   registered timestamp not null,
   dob date not null,
   roleID integer not null,
   foreign key (dob) references UsersAge(dob),
   foreign key (roleID) references Roles(roleID)
);

-- adminUsers
create table AdminUsersAge (
   dob date primary key,
   age interval not null
);

create table AdminUsers (
   userID serial primary key,
   username varchar(32) unique not null,
   password varchar not null,
   registered timestamp not null,
   dob date not null,
   roleID integer not null,
   email varchar(254) not null,
   lastLogin timestamp not null,
   adminName varchar not null,
   foreign key (dob) references AdminUsersAge(dob),
   foreign key (roleID) references Roles(roleID)
```

```
);

-- torrents
create table TorrentsAge (
   uploaded timestamp primary key,
   age interval not null
);

create table Torrents (
   hash varchar(32) primary key,
   name varchar(100) not null,
   size int not null,
   uploaded timestamp not null,
   description varchar,
   categoryID integer not null,
   userID integer not null,
   foreign key (uploaded) references TorrentsAge(uploaded),
   foreign key (categoryID) references Categories(categoryID),
   foreign key (userID) references Users(userID)
);

-- requests
create table Requests (
   reqID serial primary key,
   posted timestamp not null,
   title varchar not null,
   body varchar not null,
   hash varchar(32),
   userID integer not null,
   foreign key (hash) references Torrents(hash),
   foreign key (userID) references Users(userID)
);

-- history
create table History(
   historyID serial primary key,
   snatched timestamp not null,
   hash varchar(32) not null,
   userID integer not null,
   foreign key (hash) references Torrents(hash),
   foreign key (userID) references Users(userID)
);
```

```
-- comments
create table Comments(
   body varchar not null,
   posted timestamp,
   hash varchar(32) ,
   userID integer,
   primary key (posted, hash, userID),
   foreign key (hash) references Torrents(hash),
   foreign key (userID) references Users(userID)
);

-- junction tables
create table TorrentTags (
   tagID integer not null,
   hash varchar(32) not null,
   primary key (tagID, hash),
   foreign key (tagID) references Tags(tagID),
   foreign key (hash) references Torrents(hash)
);
```

Insert Statements

The following inserts were made using PostgreSQL v11+

```
-- roles
INSERT INTO Roles (name, permissions) VALUES
   ('Admin', B'1111111111'),
   ('Super Moderator', B'1111111000'),
   ('Moderator', B'1111100000'),
   ('User', B'1100000000'),
   ('Banned', B'0000000000');

-- tags
INSERT INTO Tags (name, description) VALUES
   ('Action', 'Movies or games with lots of action'),
   ('Comedy', 'Funny and entertaining'),
   ('Drama', 'Serious and emotional'),
   ('Sci-Fi', 'Science fiction content'),
   ('EDM', 'Electronic Dance Music genre');
```

```sql
-- categories
INSERT INTO Categories (name, description) VALUES
   ('Movies', 'Various movie genres'),
   ('Games', 'Different types of video games'),
   ('ISOs', 'Linux ISOs'),
   ('Programs', 'Programs in ELF format'),
   ('Music', 'All kinds of music albums');

-- users
INSERT INTO UsersAge (dob, age) VALUES
   ('1990-01-15', age(timestamp '1990-01-15')),
   ('1985-06-28', age(timestamp '1985-06-28')),
   ('2000-03-10', age(timestamp '2000-03-10')),
   ('1972-11-03', age(timestamp '1972-11-03')),
   ('1998-09-21', age(timestamp '1998-09-21'));

INSERT INTO Users (username, password, registered, dob, roleID) VALUES
   ('user1', 'pw1', '2020-10-20', '1990-01-15', 1),
   ('user2', 'pw2', '2021-04-20', '1985-06-28', 2),
   ('user3', 'pw3', '2021-10-20', '2000-03-10', 3),
   ('user4', 'pw4', '2022-04-20', '1972-11-03', 4),
   ('user5', 'pw5', '2022-10-20', '1998-09-21', 5);

-- adminUsers
INSERT INTO AdminUsersAge (dob, age) VALUES
   ('1990-01-15', age(timestamp '1990-01-15')),
   ('1985-06-28', age(timestamp '1985-06-28')),
   ('2000-03-10', age(timestamp '2000-03-10')),
   ('1972-11-03', age(timestamp '1972-11-03')),
   ('1998-09-21', age(timestamp '1998-09-21'));

INSERT INTO AdminUsers (username, password, registered, dob, roleID, email,
lastLogin, adminName) VALUES
   ('admin1', 'adminpassword1', '2023-01-15', '1990-01-15', 1,
'admin1@gmail.com', '2023-01-15', 'AdminName1'),
   ('admin2', 'adminpassword2', '2023-02-28', '1985-06-28', 1,
'admin2@gmail.com', '2023-02-28', 'AdminName2'),
   ('admin3', 'adminpassword3', '2023-03-10', '2000-03-10', 1,
'admin3@gmail.com', '2023-03-10', 'AdminName3'),
   ('admin4', 'adminpassword4', '2023-04-25', '1972-11-03', 1,
'admin4@gmail.com', '2023-04-25', 'AdminName4'),
```

```
   ('admin5', 'adminpassword5', '2023-05-30', '1998-09-21', 1,
'admin5@gmail.com', '2023-05-30', 'AdminName5');

-- torrents
insert into TorrentsAge (uploaded, age) values
   ('1990-01-15', age(timestamp '1990-01-15')),
   ('1985-06-28', age(timestamp '1985-06-28')),
   ('2000-03-10', age(timestamp '2000-03-10')),
   ('1972-11-03', age(timestamp '1972-11-03')),
   ('1998-09-21', age(timestamp '1998-09-21'));

insert into Torrents (hash, name, size, uploaded, description, categoryID,
userID) values
   ('hash1', 'name1', 1, '1990-01-15', 'description1', 1, 1),
   ('hash2', 'name2', 2, '1985-06-28', 'description2', 2, 2),
   ('hash3', 'name3', 3, '2000-03-10', 'description3', 3, 3),
   ('hash4', 'name4', 4, '1972-11-03', 'description4', 4, 4),
   ('hash5', 'name5', 5, '1998-09-21', 'description5', 5, 5);

-- requests
insert into Requests (posted, title, body, hash, userID) values
   ('1990-01-15', 'title1', 'body1', 'hash1', 1),
   ('1985-06-28', 'title2', 'body2', 'hash2', 2),
   ('2000-03-10', 'title3', 'body3', 'hash3', 3),
   ('1972-11-03', 'title4', 'body4', 'hash4', 4),
   ('1998-09-21', 'title5', 'body5', 'hash5', 5);

-- history
insert into History (snatched, hash, userID) values
   ('1990-01-15', 'hash1', 1),
   ('1985-06-28', 'hash2', 2),
   ('2000-03-10', 'hash3', 3),
   ('1972-11-03', 'hash4', 4),
   ('1998-09-21', 'hash5', 5);

-- comments
insert into Comments (body, posted, hash, userID) values
   ('body1', '1990-01-15', 'hash1', 1),
   ('body2', '1985-06-28', 'hash2', 2),
   ('body3', '2000-03-10', 'hash3', 3),
   ('body4', '1972-11-03', 'hash4', 4),
   ('body5', '1998-09-21', 'hash5', 5);
```

```sql
-- junction tables
insert into TorrentTags (tagID, hash) values
   (1, 'hash1'),
   (2, 'hash2'),
   (3, 'hash3'),
   (4, 'hash4'),
   (5, 'hash5');
```