**University of British Columbia, Department of Computer Science**

**CPSC 304**

**2016 Winter Term 1**

**Project Part  \_\_Formal Specification\_\_**

**Group Name: DogeTV DB Team**

**Group Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Student Number** | **Unix ID** | **Email Address** |
| Caiyi Ku | 48449136 | d1b9 | imkcy@qq.com |
| Ziyu Wang | 56125131 | j8h0b | richard.ziyu.wang@gmail.com |
| Zhixuan Xu | 53854121 | f3v9a | xuzhixuanventus@gmail.com |
| NULL | NULL | NULL | NULL |

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 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.

ER diagram update notes:

1. Shows’ attribute theme is removed.
2. Channel is now in total participation many to one relationship with Type.
   1. Each channel must have one type.
   2. Each channel can only have one type.
3. Total participation on Streamer in GrantsPrivilege is removed.
   1. Streamer may not have any moderators.

1.2.

Type (name)

* Represents: the type of a channel
* Primary key: name
* Constraints:
  + a type is either a Game or a Show

Game (name, platform)

* Represents: entity set game
* Primary key: name, platform
* Foreign key: name
  + (name) references Type

Show (name)

* Represents: entity set show
* Primary key: name
* Foreign key: name
  + (name) references Type

Channel (username, ID, Type.name, numOfViewers, status, language, description, title)

* Represents: entity set channel
* Primary key: username, ID
* Foreign key: username , Type.name
  + (username) references Streamer
  + Type.name references Type
* Constraints:
  + Username needs to be unique
  + Username cannot be null
  + Type.name cannot be null
* Status is one of {“online”, “offline”, “banned”}

User (username, gender)

* Represents: entity set user
* Primary key: username
* Constraints:
  + Each user is either a guest or a registered user

Guest (username)

* Represents: entity set guest
* Primary key: username
* Foreign key: username
  + (username) references User

RegisteredUser (username, password, numOfFollowers, regDate)

* Represents: entity set registered user
* Primary key: username
* Foreign key: username
  + Username references User

Moderator (username)

* Represents: entity set moderator
* Primary key: username
* Foreign key: username
  + (username) references RegisteredUser

Streamer (username)

* Represents: entity set streamer
* Primary key: username
* Foreign key: username
  + (username) references RegisteredUser

ChatMessage (sendername, Streamer.username, ID, time, content)

* Represents: entity set chat message
* Primary key: sendername, time
* Foreign key: sendername, username, ID
  + (Sendername (Registered.username)) references RegisteredUser
  + (Streamer.username, ID) references Channel
* Constraints:
  + Username cannot be null
  + ID cannot be null

Watches (User.username, Streamer.username, ID)

* Represents: relationship set watches
* Primary key: User.username, Streamer.username, ID
* Foreign key:
  + (User.username) references User
  + (Streamer.username, ID) references Channel

Follows (RegisteredUser.username, RegisteredUser.username)

* Represents: relationship set follows
  + Follows (Caiyi, IU) means Caiyi follows IU.
* Primary key: RegisteredUser.username, RegisteredUser.username
* Foreign key:
  + (RegisteredUser.username) references RegisteredUser
  + (RegisteredUser.username) references RegisteredUser

Mutes (Moderator.username, RegisteredUser.username, startTime, duration, endTime)

* Represents: relationship set mutes
  + Mutes (Taeyeon, Caiyi, 2016/10/17/1300, 24, 2016/10/18/1300) means that Taeyeon mutes Caiyi at 1PM on 2016/10/17 for 24 hours.
* Primary key: Moderator.username, RegisteredUser.username, startTime
* Foreign key:
  + (Moderator.username) references Moderator
  + (RegisteredUser.username) references RegisteredUser

GrantsPrivilege (Streamer.username, Moderator.username)

* Represents: relationship set grantsPrivilege
  + GrantsPrivilege (Seolhyun, Caiyi) means that Caiyi is a moderator in Seolhyun’s Channel.
* Primary key: Streamer.username, Moderator.username
* Foreign key:
  + (Streamer.username) references Streamer
  + (Moderator.username) references Moderator

Donates (RegisteredUser.username, Streamer.username, transactionNo, amount)

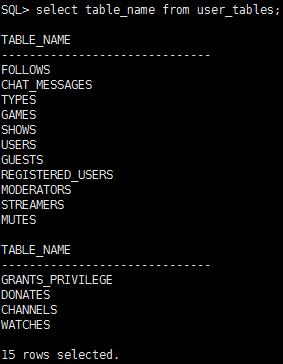
* Represents: relationship set donates
  + Donates (Caiyi, Ventus, 6666, 6.66) means that Caiyi donates 6.66 to Ventus and the transactionNo is 6666.
* Primary key: RegisteredUser.username, Streamer.username, transactionNo
* Foreign key:
  + (RegisteredUser.username) references RegisteredUser
  + (Streamer.username) references Streamer

3.4.5.6.

The code that creates tables is in createAll.sql.

The docx version of the code and functional dependencies are shown as below.

The diagram below shows the tables we create.



CREATE TABLE Types (

name char(30) NOT NULL,

CONSTRAINT Types\_pk PRIMARY KEY(name)

);

Types (name)

FDs: none

In BCNF

CREATE TABLE Games (

name char(30),

platform char(30),

CONSTRAINT Games\_pk PRIMARY KEY(name,platform),

CONSTRAINT fkGame\_Types FOREIGN KEY (name) REFERENCES Types(name)

);

Games (name, platform)

FDs: none

In BCNF

CREATE TABLE Shows (

name char(30) NOT NULL,

CONSTRAINT Shows\_pk PRIMARY KEY(name),

CONSTRAINT fkShow\_Types FOREIGN KEY(name) REFERENCES Types(name));

Shows (name)

FDs: none

In BCNF

CREATE TABLE Users(

username char(30) NOT NULL,

gender char(10),

CONSTRAINT Users\_pk PRIMARY KEY(username));

Users (username, gender)

FDs: username->gender

In BCNF

CREATE TABLE Guests(

username char(30) NOT NULL,

CONSTRAINT Guests\_pk PRIMARY KEY(username),

CONSTRAINT fkGuests\_Users

FOREIGN KEY(username)

REFERENCES Users(username)

ON DELETE CASCADE

);

Guests (username)

FDs: none

In BCNF

CREATE TABLE Registered\_Users(

username char(30) NOT NULL,

password char(20) NOT NULL,

numOfFollowers Integer,

regDate Date,

CONSTRAINT Registered\_Users\_pk PRIMARY KEY(username),

CONSTRAINT fkRegistered\_Users\_Users

FOREIGN KEY(username)

REFERENCES Users(username)

ON DELETE CASCADE

);

RegisteredUsers (username, password,numOfFollowers,regDate)

FDs: username->password,numOfFollowers,regDate

% Given the username we know all information about the user

In BCNF

CREATE TABLE Moderators(

username char(30) NOT NULL,

CONSTRAINT Moderators\_pk PRIMARY KEY(username),

CONSTRAINT fkModerators\_Registered\_Users

FOREIGN KEY(username)

REFERENCES Registered\_Users(username)

ON DELETE CASCADE

);

Moderators (username)

FDs: none

In BCNF

CREATE TABLE Streamers(

username char(30) NOT NULL,

CONSTRAINT Streamers\_pk PRIMARY KEY(username),

CONSTRAINT fkStreamers\_Registered\_Users

FOREIGN KEY(username)

REFERENCES Registered\_Users(username)

ON DELETE CASCADE

);

Streamers(username)

FDs: none

In BCNF

CREATE TABLE Follows (

follower\_name char(30) NOT NULL,

followee\_name char(30) NOT NULL,

CONSTRAINT Follows\_pk PRIMARY KEY(follower\_name, followee\_name),

CONSTRAINT fkFollows\_follower

Foreign key(follower\_name)

REFERENCES Registered\_Users(username)

ON DELETE CASCADE,

CONSTRAINT fkFollows\_followee

Foreign key(followee\_name)

REFERENCES Registered\_Users(username)

ON DELETE CASCADE);

Follows (RegisteredUser.username, RegisteredUser.username)

FDs: none

In BCNF

CREATE TABLE Mutes (

Moderator\_username char(30) NOT NULL,

RegisteredUser\_username char(30) NOT NULL,

startTime Date NOT NULL,

duration Integer,

CONSTRAINT Mutes\_pk PRIMARY KEY(Moderator\_username, RegisteredUser\_username, startTime),

CONSTRAINT fkMutes\_Moderators

Foreign key(Moderator\_username)

REFERENCES Moderators(username),

CONSTRAINT fkMutes\_Registered\_Users

Foreign key(RegisteredUser\_username)

REFERENCES Registered\_Users(username)

ON DELETE CASCADE);

Mutes (Moderator.username, RegisteredUser.username, startTime, duration)

FDs: Moderator.username, RegisteredUser.username, startTime -> duration

% Given the start time and duration of the muting, we can deduce the endtime.

In BCNF

CREATE TABLE Grants\_Privilege (

Streamer\_username char(30),

Moderator\_username char(30),

CONSTRAINT Grants\_Privilege\_pk PRIMARY KEY(Streamer\_username, Moderator\_username),

CONSTRAINT fkGrants\_Privilege\_Streamers

Foreign key(Streamer\_username)

REFERENCES Streamers(username)

ON DELETE CASCADE,

CONSTRAINT fkGrants\_Privilege\_Moderators

Foreign key(Moderator\_username)

REFERENCES Moderators(username)

ON DELETE CASCADE);

GrantsPrivilege (Streamer.username, Moderator.username)

% A streamer have the right to give a user the privilege as a moderator.

FDs: None

In BCNF

CREATE TABLE Donates (

RegisteredUser\_username char(30) NOT NULL,

Streamer\_username char(30) NOT NULL,

transactionNo Integer NOT NULL,

amount Integer NOT NULL,

CONSTRAINT Donates\_pk PRIMARY KEY(RegisteredUser\_username, Streamer\_username, transactionNo),

CONSTRAINT fkDonates\_Registered\_Users

Foreign key(RegisteredUser\_username)

REFERENCES Registered\_Users(username),

CONSTRAINT fkDonates\_Streamers

Foreign key(Streamer\_username)

REFERENCES Streamers(username)

);

Donates (RegisteredUser.username, Streamer.username, transactionNo, amount)

FDs: RegisteredUser.username, Streamer.username, transactionNo -> amount

In BCNF

% Give the username and streamername and transctionNo, we know the amount of this donation.

CREATE TABLE Channels (

Streamer\_name char(30) NOT NULL,

ID Integer NOT NULL,

numOfViewers Integer,

status char(10),

language char(30),

description char(200),

title char(100) NOT NULL,

type\_name char(30) NOT NULL,

CONSTRAINT Channels\_pk PRIMARY KEY(Streamer\_name, ID),

CONSTRAINT Channels\_unique UNIQUE(Streamer\_name),

CONSTRAINT fkChannel\_Streamers

FOREIGN KEY(streamer\_name)

REFERENCES Streamers(username)

ON DELETE CASCADE,

CONSTRAINT fkChannel\_Types

FOREIGN KEY(type\_name)

REFERENCES Types(name)

ON DELETE SET NULL

);

Channels (streamer\_name, ID, numOfViewers, status, language, description, title)

FDs: streamer\_name, ID->numOfViewers, status, language, description, title

% Given streamer name and channel ID, we can enter the channel and see the number of current viewers, status, language, description and title.

In BCNF

CREATE TABLE Watches (

User\_username char(30) NOT NULL,

Streamer\_username char(30) NOT NULL,

ID Integer NOT NULL,

CONSTRAINT Watches\_pk PRIMARY KEY(User\_username,Streamer\_username,ID),

CONSTRAINT fkWatches\_Registered\_Users

Foreign key(User\_username)

REFERENCES Users(username),

CONSTRAINT fkWatches\_Channels

Foreign key(streamer\_username,ID)

REFERENCES Channels(streamer\_name,ID)

ON DELETE CASCADE);

Watches (User.username, Streamer.username, ID)

FDs: None

In BCNF

CREATE TABLE Chat\_Messages (

sendername char(30) NOT NULL,

streamer\_username char(30) NOT NULL,

ChannelID Integer,

time Date NOT NULL,

content char(400),

CONSTRAINT Chat\_Messages\_pk PRIMARY KEY(sendername,time),

CONSTRAINT fkChat\_Msg\_RUsers

FOREIGN KEY(sendername)

REFERENCES Registered\_Users(username),

CONSTRAINT fkChat\_Msg\_Channels

Foreign key(streamer\_username,ChannelID)

REFERENCES Channels(Streamer\_name,ID)

);

ChatMessages (sendername, Streamer.username, ChannelID, time, content)

FDs: Sendername, time -> Streamer.username, ChannelID, content

% A user can only send one message to one streamer at a time.

In BCNF

7. instances

The code that insert values into tables is in insertAll.sql.

The results are in instances.txt.

8.9.10. Functionality Specifications

* Guest and general registered users (not a moderator or streamer) will have the same interface.
* One interface for moderators
* One interface for streamers
* One interface for admins
* **registered users**:
  + join a channel
  + make donations
  + follow other registered users
  + check for followers
  + send chat messages
  + check donations made/received
  + search for channels of specific type
  + search for specific user
* **guest**:
  + join a channel
  + register
  + search for channels of specific type
  + search for specific user
* **moderator**:
  + mute other viewers for a given duration
* **streamer**:
  + start/close stream
  + grant moderator privileges
  + change channel info (i.e. type, description, title and language)
* **admin**:
  + add/delete users
  + create channels for a streamer
  + allow a registered user to become a streamer
  + ban channels which violates the term of service
  + reactivate banned channels
  + get top channels
  + get top channel types

11. Platform to Use

We are going to use the CS Ugrad Oracle installation and provided PHP/Apache.

12. Data

Please see schema and SQL DDL above.

13. Division of Labour

We are going to do the project together and hopefully each member will equally contribute to the project. This may seem inefficient, but every member will learn more.

14. Extra Feature

Every user will receive a prediction list of channels that he/she might prefer to watch in the user interface. This list only consists of a certain number of channels. We will decide how to approach this later in the actual implementation.

15. Connection and Screenshot of Query

The code that creates the connection to the database is in demo.php.

