EECS 3421 Winter 2019

Assignment 3

ERD & DB Design

Instructor: Manos Pappagelis


Taswar Karim

Number of Grace Days Used: 3

PART A

Assumptions

- Only users that have joined the group can view, write, reply to posts. That is why we have a entity called GroupMembers.
- We have assumed that users have already signed up into the system and hence they can send and receive numerous couch requests/friend requests which we have shown in the entities Couch Request and FriendRequest
- User can surf and host couch. In our E/R diagram, a user has both functionalities.
- A user may be able to host more than one couch.
- We have an Event Attendees entity, that keeps track of the users attending the event.
- A user can choose to create no or many events
- An Event can be uniquely determined by eventID
- One attendee can invite many users and a user can receive many invites from different attendees.
- We assume an event has at least one attendee (the user who created the event)
- A member of the group can choose to create no or many posts, reply to no or many posts and also report a post to the admin
- Administrators cannot be friends with users and cannot participate in groups or any event activities. We are treating administrators as moderators, who examines groups and reports.
- Each admin can review many reports and similarly each report can be reviewed by many administrators.
- Each user has a search history, each search can be uniquely identified by its historyID and searchID
- We assume once a couch is available, a notification will be sent to the user who has searched for the couch previously.
- Each notification is uniquely identified by its nID in the relation Notification and it also keeps track of users each notification is being sent to.
- Each user has their own profile on while they choose to share further information, each Profile is uniquely identified by its profileID
- Each user profile contains friendlist which may contain no or many friends.
- ProfileID, SurferID,AttendeeID, creatorID, hostID all acts as UserID.
- Friendrequests are uniquely identified by fid and friendships are identified by the two userId's.
- Not all foreign keys are included in our E/R diagram, but our DataBase Schema contains all the primary, foreign and super keys.
- Adresses of user, events and couch can be accessed by City (where city contains the country name too)

PART B
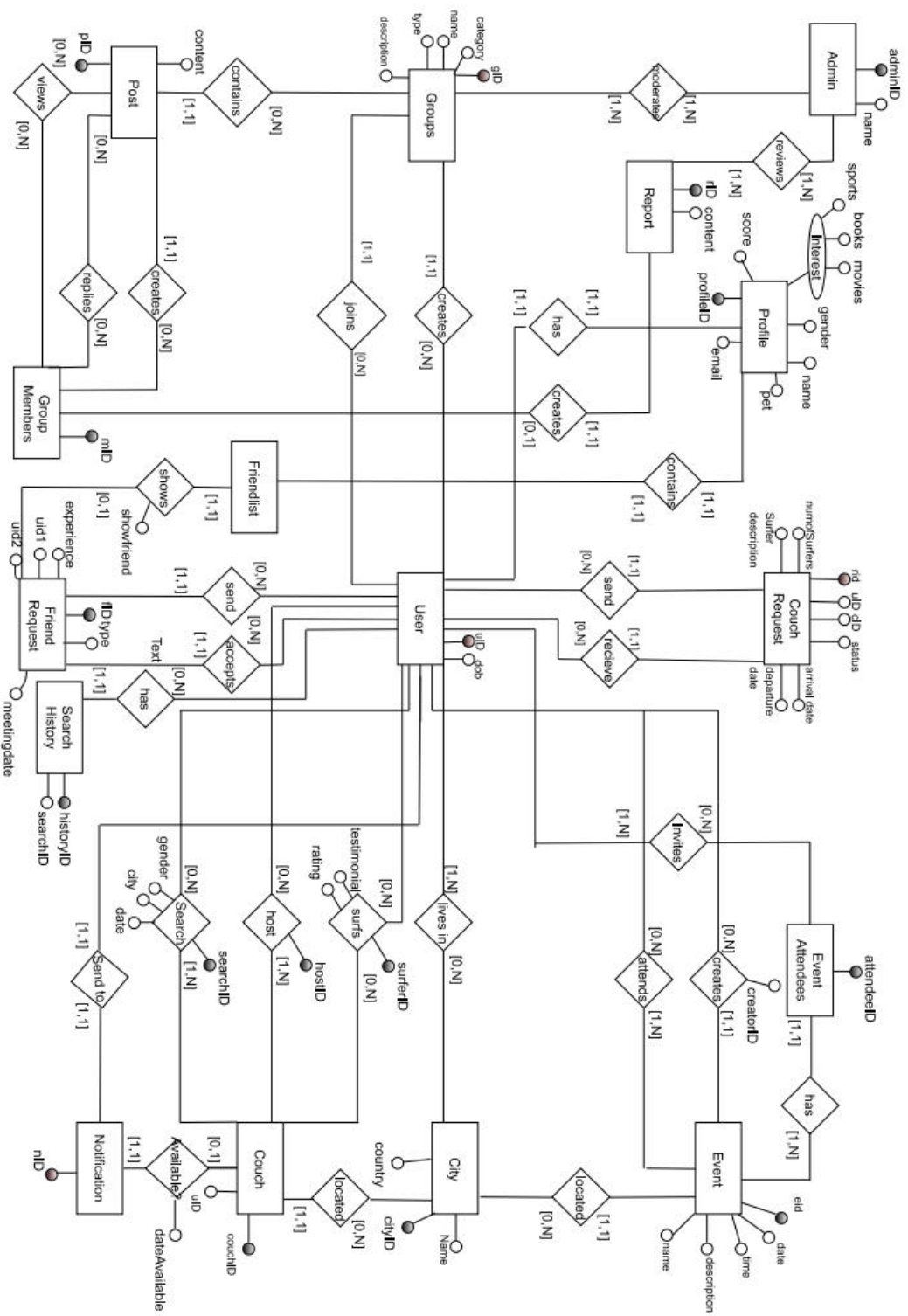
Part C

User (**uID**, dob, e-mail)
Profile(**profileID**, name, gender, score, pet, sports,books,movies)
City(**cityID**, country, name, **uID**)
Event(**eID**,date,time,description,name,creatorID)
Locations(**eID**,cityID,**couchID**)
EventAttendees(**atteneeID**,eID,**cityID**)
Invites(**attendeeID**,**uID**,**eID**)
Surfer(**surferID**,testimonial,rating,**couchID**)
Host(**hostID**,**couchID**)
Couch(**couchID**,**uid**,dateAvailable)
Search(**searchID**,gender,city,date)
Notification(**nID**,**uID**,**couchID**)
CouchRequest(**rID**,**uID**,**couchID**,status,numOfSurfers,surferDescription,arrivalDate,DepartureDate)
FriendRequest(**fID**,friendType,**uID1**,**uID2**,meetingDate,experience)
FriendList(**uID**,**fID**,showfriend)
SearchHistory(**historyID**,**sID**,**uID**)
GroupMembers(**mID**,**gID**)
Post(**pID**,**gID**,**mID**,content)
Groups(**gID**,category,name,type,description)
Report(**rID**,content,**pID**)
Admin(**adminID**,name,**rID**)

Part D

```
DROP SCHEMA IF EXISTS A3 CASCADE;
CREATE SCHEMA A3;
SET search_path TO A3;

DROP TABLE IF EXISTS User CASCADE;
DROP TABLE IF EXISTS Profile CASCADE;
DROP TABLE IF EXISTS City CASCADE;
DROP TABLE IF EXISTS Event CASCADE;
DROP TABLE IF EXISTS Location CASCADE;
DROP TABLE IF EXISTS EventAtendees CASCADE;
DROP TABLE IF EXISTS Invites CASCADE;
DROP TABLE IF EXISTS Surfer CASCADE;
DROP TABLE IF EXISTS Host CASCADE;
DROP TABLE IF EXISTS Couch CASCADE;
DROP TABLE IF EXISTS Search CASCADE;
DROP TABLE IF EXISTS Notification CASCADE;
DROP TABLE IF EXISTS CouchRequest CASCADE;
DROP TABLE IF EXISTS FriendRequest CASCADE;
DROP TABLE IF EXISTS FriendList CASCADE;
DROP TABLE IF EXISTS SearchHistory CASCADE;
DROP TABLE IF EXISTS GroupMembers CASCADE;
DROP TABLE IF EXISTS Post CASCADE;
DROP TABLE IF EXISTS Groups CASCADE;
DROP TABLE IF EXISTS Report CASCADE;
DROP TABLE IF EXISTS Admin CASCADE;


CREATE TABLE User (
    uID             INTEGER        PRIMARY KEY,
    dob             VARCHAR(15)    NOT NULL,
    email           VARCHAR(15)    NOT NULL);

CREATE TABLE Profile (
    profileID  INTEGER      REFERENCES User(uID) ON DELETE RESTRICT,
    name       VARCHAR(15)  NOT NULL,
    gender     VARCHAR(15)  NOT NULL
    score      INTEGER      NOT NULL,
    pet        BOOLEAN      NOT NULL,
    sports     VARCHAR(50)  NOT NULL,
    books      VARCHAR(50)  NOT NULL,
    movies     VARCHAR(50)  NOT NULL);

CREATE TABLE City (
    cityID     INTEGER      PRIMARY KEY,
    country    VARCHAR(15)  NOT NULL,
    name       VARCHAR(15)  NOT NULL
    uid        INTEGER      REFERENCES User(uID) ON DELETE RESTRICT);
```

```
CREATE TABLE Event (
    eID           INTEGER       PRIMARY KEY,
    creatorID  INTEGER       REFERENCES User(uID) ON DELETE RESTRICT,
    date          VARCHAR(15)   NOT NULL,
    time          INTEGER       NOT NULL
    name         VARCHAR(15)   NOT NULL,
    description  INTEGER       NOT NULL);

CREATE TABLE Location (
    eID           INTEGER       REFERENCES Event(eID) ON DELETE RESTRICT,
    cityID       INTEGER       REFERENCES City(cityID) ON DELETE RESTRICT,
    couchID     INTEGER       REFERENCES Couch(couchID) ON DELETE RESTRICT,
    PRIMARY KEY(eID,cityID)
);

CREATE TABLE EventAttendees (
    attendeeID INTEGER       REFERENCES User(uID) ON DELETERESTRICT,
    eID           INTEGER       REFERENCES Event(eID)ON DELETE RESTRICT,
    cityID      INTEGER       REFERENCES City(cityID) ON DELETE RESTRICT);
    PRIMARY KEY(attendeeID,eID)

CREATE TABLE Invites (
    attendeeID INTEGER       REFERENCES User(uID) ON DELETERESTRICT,
    uID          INTEGER       REFERENCES User(uID)ON DELETE RESTRICT,
    eID          INTEGER       REFERENCES Event(eID) ON DELETE RESTRICT);

CREATE TABLE Surfer (
    surferID   INTEGER       REFERENCES User(uID) ON DELETERESTRICT,
    testimonial VARCHAR(50)   NOT NULL,
    rating       INTEGER       NOT NULL,
    couchID     INTEGER       REFERENCES Couch(couchID) ON DELETE RESTRICT);
    PRIMARY KEY(surferID,couchID)

CREATE TABLE Host (
    hostID     INTEGER       REFERENCES User(uID) ON DELETERESTRICT,
    couchID    INTEGER       REFERENCES Couch(couchID)ON DELETE RESTRICT);
    PRIMARY KEY(hostID,couchID)


CREATE TABLE Couch (
    couchID        INTEGER   PRIMARY KEY,
    uID             INTEGER   REFERENCES User(uID)ON DELETE RESTRICT,
    dateAvailable  VARCHAR(15) NOT NULL);


CREATE TABLE Search (
    searchID   INTEGER       PRIMARY KEY,
    gender     VARCHAR(6)    NOT NULL,
    city        VARCHAR(15)   NOT NULL,
    date        VARCHAR(15)   NOT NULL);

CREATE TABLE Notification (
    nID          INTEGER       PRIMARY KEY,
    uID          INTEGER       REFERENCES User(uID)ON DELETE RESTRICT,
    couchID    INTEGER       REFERENCES Couch(couchID) ON DELETE RESTRICT);
```

```sql
CREATE TABLE CouchRequest (
  rID                 INTEGER    PRIMARY KEY,
  uID                 INTEGER    REFERENCES User(uID)ON DELETE RESTRICT,
  couchID             INTEGER    REFERENCES Couch(couchID) ON DELETE RESTRICT,
  status          VARCHAR(10) NOT NULL,
  numOfSurfers        INTEGER    NOT NULL,
  surferDescription VARCHAR(50) NOT NULL,
  arrivalDate       VARCHAR(15) NOT NULL,
  departureDate.    VARCHAR(15) NOT NULL);


CREATE TABLE FriendRequest (
    fID           INTEGER      PRIMARY KEY,
    friendType   VARCHAR(20)   NOT NULL,
    uID1         INTEGER       REFERENCES User(uID)ON DELETE RESTRICT,
    uID2         INTEGER       REFERENCES User(uID)ON DELETE RESTRICT,
    meetingDate  VARCHAR(15)   NOT NULL,
    experience   INTEGER       NOT NULL);

CREATE TABLE FriendList (
    fID         INTEGER       REFERENCES FriendRequest(fID)ON DELETE RESTRICT,
    uID         INTEGER       REFERENCES User(uID)ON DELETE RESTRICT,
    showFriend BOOLEAN        NOT NULL);
    PRIMARY KEY(fID,uID)

CREATE TABLE SearchHistory(
    historyID  INTEGER        PRIMARY KEY,
    sID        INTEGER        REFERENCES Search(searchID)ON DELETE RESTRICT,
    uID        INTEGER        REFERENCES User(uID)ON DELETE RESTRICT);

CREATE TABLE GroupMembers (
    mID         INTEGER       PRIMARY KEY,
    gID         INTEGER        REFERENCES Group(gID)ON DELETE RESTRICT);

CREATE TABLE Post (
    pID         INTEGER       PRIMARY KEY,
    gID         INTEGER       REFERENCES Group(gID)ON DELETE RESTRICT,
    mID         INTEGER       REFERENCES Member(mID) ON DELETE RESTRICT,
    content   VARCHAR(200)    NOT NULL);

CREATE TABLE Groups (
    gID         INTEGER       PRIMARY KEY,
    category   VARCHAR(10)    NOT NULL,
    name       VARCHAR(20)    NOT NULL,
    type       VARCHAR(5)     NOT NULL,
    description VARCHAR(50)    NOT NULL);

CREATE TABLE Report(
    rID         INTEGER       PRIMARY KEY,
    content   VARCHAR(200)    NOT NULL
    pID         INTEGER       REFERENCES Post(pID) ON DELETE RESTRICT);
```

```sql
CREATE TABLE Admin(
    adminID     INTEGER         PRIMARY KEY,
    name        VARCHAR(20)     NOT NULL,
    rID         INTEGER         REFERENCES Report(rID) ON DELETE RESTRICT);
```