1. Preamble

- Last modified date: 03/04/2016 - Project: Game Library - Members:
- Sam Lerman, slerman (team captain)
- Cinthia Campos, clagesca
- Rodrigo Santiago, rdeolive
- Yuen Fung Lee, ylee109
- Master directory:
- slerman/public_html

2. Domain descriptions

-A library designed for players to keep track of states, scores and other game stats. This is designed for a developer group, therefore the developers would have access to the database in order to update it with new games and game specifications. The users – the players – would have individual logins and passwords, and they can interact with each other by "friending" them.

3. Primary Entities

Sam:

Subscriptions Platform

Cinthia:

Players States

Rodrigo:

Scores Developer

Lee:

Guild OnlineWallet

4. Relationships and Weak Entities

Weak Entities:

games

Cinthia:

subscribesTo(player_id, subscription_id)
gameStates(player_id, state_id)
usersPlatform(plat_name, plat_version, player_id)
contests (player_id, score, game_id)

Lee:

plays(player_id, game_id)
develops(developer_id, game_id)
IsPlayingAgainst(player_id, player_id, game_id)
owns(player_id, wallet_contents)

Sam:

playerScore(player_id, game_id, score, game_state)
runsOn(game_name, plat_name,plat_version)
stateOf(game_name, game_state)

developsFor(developer_id, plat_name,plat_version)

Rodrigo

IsFriend(player_id, player_id)
IsPartOfASeries(game_id, game_series)
IsOnGuild(player_id, game_guild)
banned(player_id, game_id)
IsAdmin(player_id, guild_id)

5. Attributes

Cinthia:

Players: player_id, player_name, password, friends_count, game_hours,
States: state_id, game_state, currscore, duration, num_players, turn

Rodrigo:

Scores: score_id, score_value, coop_score, versus_score
Developers: developer_id, developer_name, developer_country, year_founded

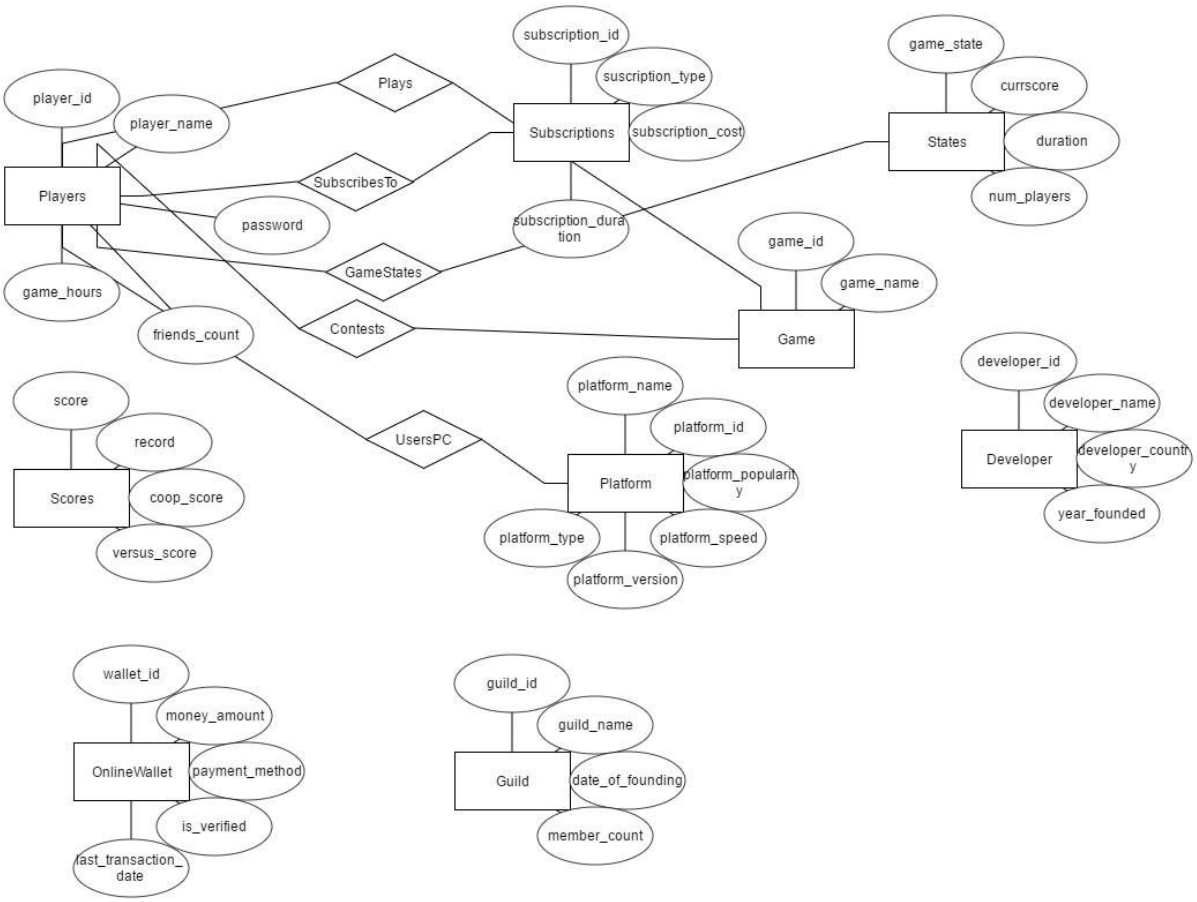
Sam:

Subscriptions(subscription_id, type, subscription_cost, subscription_duration)
Platform(platform_id, plataform_name, popularity, speed, version, type)
Games(game_name)

Lee:

OnlineWallet(id, money_amount,payment_method, is_verified,last_transaction_date)
Guild(member_count, date_of_founding,guild_id,guild_name)

6.



7.

Games
Platform_id
Game_name

States
state_id
game_state
Currscore
Duration
Num_players
turn

Players
Player_id
Player_name
Password
Friends_count
Game_hours

8. Functional Dependencies (M)

players:

id->name,password,friends_count,game_hours

States:

state_id->game_state

game_state->currscore,duration,num_players,turn

Scores:

score_id->score_value

score_value->coop_score, versus_score

Developer:

developer_id->developer_name,developer_country, year_founded

Subscriptions:

subscription_id->subscription_type,subscription_duration

subscription_type->subscription_cost

Platform:

platform_id->platform_name,platform_type,platform_version

platform_type->popularity, speed

OnlineWallet:

onlineWallet_id->money_amount,payment_method, is_verified,last_transaction_date

Guild:

guild_id->guild_name,member_count,date_of_founding

8. Functional Dependencies (M) Decomposed Into 3rd Normal Form

Players:

player_id->player_name,password,friends_count,game_hours

States:

state_id->game_state,currscore,duration,num_players,turn

Scores:

score_id->score_value,coop_score, versus_score

developers:

id->name,country, year_founded

Subscriptions:

subscription_id->subscription_type,subscription_duration

Subscription_Costs:

subscription_type->subscription_cost

Platform:

platform_id->platform_name,platform_type,platform_version, speed, popularity

OnlineWallet:
onlineWallet_id->money_amount,payment_method, is_verified,last_transaction_date

Guild:
guild_id->guild_name,member_count,date_of_founding

9.Proof of Normal Form (M)

For the entity Subscriptions, we had to decompose the second FD creating a new weak entity called Subscription_Costs with the two attributes subscription_type and subscription_cost.

For Third Normal Form, each functional dependency must satisfy:

- No non-prime attribute is transitively dependent on prime key attribute.□
- For any non-trivial functional dependency x->a, x is a superkey or a is a prime attribute□

For BCNF, for any non-trivial FD x->a, x must be a superkey □

Database Implementation I - Schema (M)

Lee:

```
CREATE TABLE onlineWallets (
   id       char(5),
   payment_method       varchar(40),
   money_amount         integer,
   last_transaction_date   date,
   is_verified boolean
);

CREATE TABLE guilds (
   id       char(5),
   name       varchar(40),
   member_count         integer,
   date_of_founding   date
);
```

Cinthia:

```
CREATE TABLE players (
   id       char(7) PRIMARY KEY,
   name       varchar(40),
   password         varchar(40) ,
   friends_count  integer,
   game_hours integer
);

CREATE TABLE states (
   id       char(7) PRIMARY KEY,
   state       varchar(40),
   currscore        integer,
   duration   integer,
```

```
    num_players integer,
    turn integer
);
```

Rodrigo:

```
CREATE TABLE scores (
        id                      char(5) PRIMARY KEY,
        value     integer,
        coop_score          integer,
        versus_score        integer
);
```

Sam:

```
CREATE TABLE subscriptions (
    id        integer,
    type       varchar(40),
    duration        integer,
    PRIMARY KEY(subscription_id),
    FOREIGN KEY(subscription_type) REFERENCES subscription_costs(subscription_type)
);

CREATE TABLE subscription_costs (
        subscription_type               varchar(40),
        subscription_cost               integer,
        PRIMARY KEY(subsciption_type)
);

CREATE TABLE platforms (
    name       varchar(40),
    version       varchar(40),
    type       varchar(40),
    speed        integer,
    popularity     integer,
    CONSTRAINT name_version PRIMARY KEY(name,version)
);

CREATE TABLE games(
  name VARCHAR(40)
);

CREATE TABLE players (
    name       varchar(40) PRIMARY KEY,
    password          varchar(40) ,
    friends_count  integer,
    game_hours integer
);
```

```
CREATE TABLE developers (
        name                varchar(40) PRIMARY KEY,
        country   varchar(40),
        year_founded       integer
);

CREATE TABLE runsOn(
  game_name VARCHAR(40),
  plat_name VARCHAR(40),
plat_version VARCHAR(40),
    PRIMARY KEY (game_name, plat_version, player_name),
    FOREIGN KEY(game_name) REFERENCES games(name)  ON DELETE CASCADE UPDATE CASCADE,
    FOREIGN KEY(plat_name, play_version) REFERENCES platforms(name, version) ON DELETE CASCADE
UPDATE CASCADE
);

CREATE TABLE developsFor(
  developer_name VARCHAR(40),
  plat_name VARCHAR(40),
plat_version VARCHAR(40),
    PRIMARY KEY (developer_name, plat_version, player_name),
    FOREIGN KEY(developer_name) REFERENCES developers(name) ON DELETE CASCADE UPDATE
CASCADE,
    FOREIGN KEY(plat_name, play_version) REFERENCES platforms(name, version) ON DELETE CASCADE
UPDATE CASCADE
);

CREATE TABLE usersPlatform(
    plat_name VARCHAR(40) NOT NULL,
    plat_version VARCHAR(40) NOT NULL,
    player_name INT NOT NULL,
    PRIMARY KEY (plat_name, plat_version, player_name),
    FOREIGN KEY(plat_name, play_version) REFERENCES platforms(name, version) ON DELETE CASCADE
UPDATE CASCADE,
    FOREIGN KEY player_name references players (name) ON DELETE CASCADE UPDATE CASCADE
);
```

PROJECT PHASE 4

Note: Sam Lerman did almost the entire project by himself. He stayed up days and night getting it done and making it as perfect as it could be. He absolutely deserves 100%. His team did not help much despite his begging, and that is NOT his fault. He worked extremely hard and did both M1 and M2. Not only that, but he went above and beyond and made a beautiful looking website, with full search/insert/update/delete/relationships/list/show functionality.

11. **Database Implementation - whole team:**

This is the only part of the project that Sam received help on from his teammates. Together, we compiled a list of sql statements in a .sql file and executed the file, creating our entities, relationships, and populating our tables with data.

12. **List and Show - Sam Lerman:**

Sam created this. I used PHP's for loop to echo a Bootstrap (CSS framework) table filled with fetched data from Platforms, and I limited them to 15 by order of most popular. For the show page, I similarly loopped through all attributes in a given entity, displaying their names and values.

List: http://betaweb.csug.rochester.edu/~slerman/list.php?who=Sam

Show: http://betaweb.csug.rochester.edu/~slerman/details.php?who=Sam&name=windows&version=10

13. **Home page - Sam Lerman:**

Sam created this. I used Bootstrap, jQuery, and LESS to style the page. You'll find links on the top and right, with a link to this document at the bottom.

Home Page: http://betaweb.csug.rochester.edu/~slerman/index.php

14. **Complex (M1) - Sam Lerman:**

Sam created this. I used a separate relationships.php page to display a table of related entities. An example query is:

```
$users_platform = $db->prepare('SELECT players.name AS name FROM usersPlatform, players WHERE usersPlatform.plat_name=:name AND usersPlatform.plat_version=:version AND usersPlatform.player_name=players.name');
```

Relationships:
http://betaweb.csug.rochester.edu/~slerman/relationships.php?who=Sam&name=xbox&version=one

15. **Search (M2) - Sam Lerman:**

Sam created this. I made sure to take into account the majority of possible search variations. For example, whether your search is lowercase or uppercase, or whether it is purely the search term or contains the search term, the results will display the appropriate entities. Here was my somewhat elaborate query:

```
$table = $db->prepare("SELECT name, version, type, speed, popularity FROM platforms WHERE (LOWER(:search) ILIKE '%' || LOWER(name) || '%') OR (LOWER(:search) ILIKE '%' || LOWER(version) || '%') OR (LOWER(:search) ILIKE '%' || LOWER(type) || '%') OR (LOWER(:search) ILIKE LOWER(speed::text)) OR (LOWER(:search) ILIKE LOWER(popularity::text)) ORDER BY popularity DESC");
```

I made sure to sanitize/validate all input. I used a POST on my list.php to return only the list posted by the search query if there is one.

Search: http://betaweb.csug.rochester.edu/~slerman/list.php?who=Sam

16. **Insert (M1) - Sam Lerman:**

Sam created this. Click the green plus button to insert an entity. I made sure to sanitize/validate all input. I used a POST on my details.php page to check which entity to show; in this case, it displays the newly created one. I used this query within detail.php:

```
if($who = 'Sam') {$sql = "INSERT INTO platforms( name, version, type,  speed, popularity) VALUES ( :name,  :version, :type, :speed, :popularity)"; $stmt = $db->prepare($sql);
$stmt->bindParam(':name', test_input($_POST['name']), PDO::PARAM_STR); $stmt->bindParam(':version', test_input($_POST['version']), PDO::PARAM_STR);$stmt->bindParam(':type', test_input($_POST['type']), PDO::PARAM_STR);$stmt->bindParam(':speed', test_input($_POST['speed']), PDO::PARAM_STR);
$stmt->bindParam(':popularity', test_input($_POST['popularity']), PDO::PARAM_STR);
$stmt->execute();}
```

Insert: http://betaweb.csug.rochester.edu/~slerman/list.php?who=Sam

17. **Update (M2) - Sam Lerman:**
Sam created this. Much like insert, using POST on detail.php page and input validation, but with this query:

$sql = "UPDATE platforms SET name = :name, version = :version, type = :type, speed = :speed, popularity = :popularity WHERE name=:nameOriginal AND version=:versionOriginal";

Update: http://betaweb.csug.rochester.edu/~slerman/details.php?who=Sam&name=windows&version=10

18 **Delete - Sam Lerman:**
Sam created this. Click the red x button to delete. This is part of a POST on link. Here's the query:

$sql = "DELETE FROM platforms WHERE name=:name AND version=:version";

Delete: http://betaweb.csug.rochester.edu/~slerman/list.php?who=Sam