The Game Database Design

Abstract:

We designed our database so that it can accommodate conveniently the essential elements that builds the game up. In this document, we will discuss these different elements and how they are related to each other in a way that assures a consistent user experience

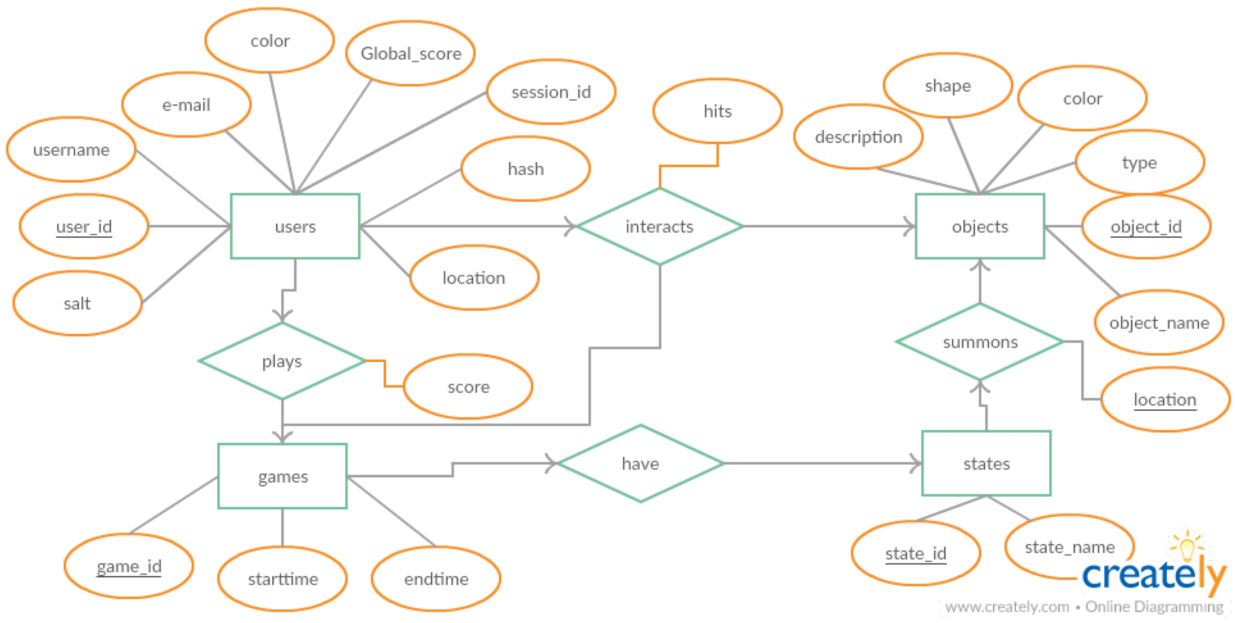
Entity Relationship Model:

In this section, we depict the database design and elaborate more on the entities, attributes and relations we included.

- ***Database Name:*** TheGame
- ***Username:*** admin
- ***Password:*** thegame

E-R Diagram:

The diagram shown below depicts the game data base system. The diagram was created on www.creatly.com



Entities Description:

Users:

Users is the entity that holds the game users, who are the online players registered to play the game. The user entity contains a list of attributes that differentiates between users and helps in managing the game.

- ***Username:*** a 1 to 16 alphanumeric unique name created by each user at sign up. This username differentiates players and it is subjected to user changes which means it is not consistent but it always has to be unique.
- ***E-mail:*** a valid verified e-mail address which should be used for any kind of communication required between the game and its users.
- ***Color:*** a unique default color that will be used by the user to identify themselves from others. The color is subjected to user updates which means it is not constant yet it always has to be a unique color
- ***Global score:*** an accumulative score calculated for each user so that they can follow on their progress and compete with others. The global score is an accumulative score calculated from all the games a user has participated in.
- ***SessionID:*** when a user signs in their session is stored in this session_id. This session_id is usually compared to what the server gets from the client side in a cookie to approve authentication. This field updates every time the user logs out and in by the system.
- ***Hash:*** this field stores the hashed version of the user's created password for later comparison for every login trial. This updates every time the user changes their password.
- ***Location:*** this stores the latest reported location of the user before closing their game session. This is to be used to get the player to their same location when they get back into the game.
- ***Salt:*** this stores the salt used to hash the user's password at the time of creation. It changes every time the user updates their password
- ***User_id:*** this is a 6 digit automatically unique generated number assigned per user. It is used for database management purposes since everything else is changeable, this is the only unique consistent value that can be used as a primary key to address users in the database.

Games:

Games is the entity that holds the information related to the history of each player. One game is a time session where the user played the game, each game consists of different states, which means the user has encountered a number of challenges. The user entity contains a list of attributes that helps in managing the game.

- ***Startime:*** stores the starting time of a game session started by a player.
- ***Endtime:*** stores the ending time of a game session.
- ***Game_id:*** a unique automatically generated 10 digit number that identifies each game session for each different user. (Primary key)

States:

States is the entity that holds the different states of the game. One state of a game defines what the challenges that the game will have at this particular time, this means adding or removing different objects and changing or keeping objects locations. The state entity contains a list of attributes that helps in managing the game.

- ***State_id:*** is a unique 10 digit id that differentiates between different states.
- ***State_name:*** which is a unique name given to each state for differentiation

Objects:

Objects is the entity that holds all other objects in the game that the players will deal with. Objects are everything except players and buildings which are hard coded in the map. The objects entity contains a list of attributes that helps in managing the game.

- ***Description:*** is a brief description that is associated with every object.
- ***Object_name:*** is a unique name that identifies each object
- ***Shape:*** describes how each object should look like
- ***Color:*** is the color of the given object
- ***Type:*** each group of objects will have a certain type. Each type indicates which kind of actions should entail or impose to the game and what kind of reaction is expected when a player runs into the object.
- ***Object_id:*** is a 10 digits unique identifier that differentiates between different objects.

Relations definitions:

Plays:

Plays is a relationship that ties users to games. Each play action will be connecting one user to one game, throughout this game the user will be collecting score. Score is the only attribute included in this relationship

Have:

Have is the relation between games and states where each user game session will be toggling between different sessions. No attributes are required per relation

Summons:

Summons is the relation between states and objects where each state will be summoning a group of objects to be present for the state. Each summons tuple will be given a unique location attribute that indicates objects location in a certain state.
All of the attributes together should be unique, this guarantees that even though an object can appear more than once in state, it will never be in the exact same location.

Interacts:

Interacts is the relationship between users and objects in games where each user will be interacting with the objects in the game. Interactions will store the hits of each user to the number of objects, this helps in having statistics and setting scores for each user per game session.

Database Triggers

Users entity:

Incident: Before Insert:

- User_id is a 6 digit in form of (000111) assigned automatically by the system.
- Username can only be alphanumeric
- No 2 users can have the same username
- No 2 users can have the same color at a time
- email should be in the form of %@%.%

Incident: Before Update:

- No 2 users can have the same username
- No 2 users can have the same color at a time
- No 2 users can have the same session_id
- email should be in the form of %@%.%

Incident: Before Delete:

After taking user permission

- Delete all games played with this user
- Clear all scores tables
- Clear all objects interactions

Games entity:

Incident: Before Insert:

- Game_id is a 10 digit in form of (0000000111) assigned automatically by the system.
- Game_id, user_id combination should always be unique.
- Starttime is assigned automatically to the current timestamp
- Endtime should be null till the next update

Incident: After Insert:

- Populate a new data tuple in the plays table with the new combination of user_id, game_id.

Incident: Before Update:

- Game_id, user_id combination should always be unique.
- Starttime is not to be changed
- Endtime should be greater than the Starttime

Objects entity:

Incident: Before Insert

- Object_id is a 10 digit in form of (0000000111) assigned automatically by the system
- Object_name should be unique
- Object color should also be unique

Incident: Before update

- Object_name should be unique
- Object color should also be unique

Incident: After delete

- Delete object from interacts table
- Delete object from all containing states

States entity:

Incident: Before Insert

- state_id is a 10 digit in form of (0000000111) assigned automatically by the system
- state_name should be unique

Incident: Before Update

- state_name should be unique

Plays Relation:

Incident: Before Insert:

- Game score should be initiated at zero

Incident: After Update:

- Update users global score with the new computed score

Summons Relation

Incident: Before Insert:

- Combination of state, object and location should be unique

Incident: Before update:

- Combination of state, object and location should be unique

Interacts Relation

Incident: Before Insert:

- Hits should be set to zero

Incident: After update:

- Update accumulative hits per user

Database Procedures:

Start game:

This procedure should be called everytime a user joins the game. This procedure initiates all required data points in the database like a game start time and a ready scores field.

End Game:

This procedure is called everytime a user ends a game by any means including direct logging out, jumping to main menu, closing the connection or losing the connection.
This procedure stores the number of hits the user made with different objects and calculates the game score and global score based on the number of hits and type of objects

Reset Score:

This procedure is called when the user decides to reset all his activity in the game. It should be called only by the user. It deletes all game-related information played by the user while keeping their own info

Delete Game:

This procedure is called when the user decides to remove a certain game from his playing list. This procedure deletes this game and recalculate their score accordingly.