COMP421 Project1: Database of Gaming Platform

Group 60
Kexin Fang, 260773389
Pinghsuan Cheuh, 260709676
Xiaohui Wang, 260719359
Zhenghua Chen, 260783959

Part 1: Requirement analysis

Application Description

This application demonstrates an online platform that supports digital distribution of video games. We specify the data requirements and design a database that satisfies the functional requirements of the application. The digital platform is incorporated with functionalities such as game purchase, discussion board, community activity and friend connection.

The application allows publishers to publish new games. Registered users can buy games, attend activities, make friends and communicate their game experience through participating in various types of communities. Users benefit from a communicative environment through the platform.

We have some calculation expectations listed as we are interested in processing the statistics in the database, for example:
Number of players who bought games
Number of players who bought a specific game
Number of players who bought a specific game over a time interval through time
Number of players who bought a specific game during a specific period
Number of active players of a game
Number of activities hosted about a game
Number of attendees of each activity hosted about a game
Amount of money players spent on a genre of games
Number of posts uploaded on a community platform over time

By carrying out calculations and data processing, our gaming platforms allows effective analysis and future projection regarding the games and the industry.

Entities and their attributes (10)

User: A user is someone who is registered with the online platform. A user is associated with the attributes such as username, password, birthday, district, billing address, bank account, e-mail and a unique primary key called id.
Publisher: A publisher is a subcategory of User, and is someone who publishes games on the platform. It inherits the features as a User and supports its own attributes such as income and billing address.

Player: A player is a subcategory of User, and is someone who buys and/or plays games on the platform. It inherits the features as a User and supports its own attribute recharge.

Game: A game is published by a publisher and played by players. A game is associated with attributes such as price, name and id, where id is the primary key.
Version: A version is a weak entity that supports the updates of a game. A version is associated with the key version number.

Community: A community is a weak entity that serves as a platform, where users' posts are shown. A community has attributes such as name, number of members and create time. It is defined by the community name and the foreign key of the game associated, the name-version combination.
High-end community: High-end community is a subcategory of the community entity. Each high-end community has attributes of maximum member count and minimum level.
VIP community: VIP community is a subcategory of the community entity. Each VIP community has an attribute of minimum recharge. As a privilege to be in the VIP community, members are informed of special messages from the publisher.

Post: A post is a weak entity that stores the messages shared on the community platform of a game. A post has attributes such as time and contents. It is defined by time, and foreign keys: associated community and the game associated with the community.
Activity: An activity is an event which the publishers host and players attend. An activity has attributes such as activity name, host time and rules, where name and version form the primary key.

Relationships (14)

Friend: A user is friends with other users. This is a many-to-many relationship as a users can have multiple friends, and multiple users can be friends with one user.
Publish: A game is published by a publisher. This is a one-to-many relationship as a publisher can publish multiple games, and a game can only be published by one publisher. The publish relationship has the attribute publish date. Each game is associated with one and only one publisher.
Buy: A player buys a game. This is a one-to-many relationship as a player can buy multiple games, and a game can only be bought by a player once. The buy relationship has the attributes of order id and date.
Play: A player plays a game. This is a many-to-many relationship as a game can be played by multiple players, and a player can play multiple games. The play relationship has the attribute that indicates the date when the user last played.

Join: A player joins a community. This is a many-to-many relationship as a player can join in multiple communities, and a community can have multiple players joined in.
Administrate: A player administrates a community. This is a many-to-many relationship as a player can administrate multiple communities, and a community can be administrated by multiple players. A community needs to have at least administrator.

Has: A game has a community. This is a one-to-many relationship as a game can have multiple communities, and each community has only one game associated.
Upload: A post is uploaded by a user. This is a one-to-many relationship as a post can be uploaded only by a single user, and a user can upload multiple posts.
Contain: A post is contained in a community. This is a one-to-many relationship as a post can be contained in only a single community, and a community can have multiple posts.
Update: A game is updated to different versions. This is a one-to-many relationship as a game can be updated in multiple versions, and a version can be updated regarding only one game. The update relationship has the attribute version number.

Host: An activity is hosted by a publisher relating to a game. This is a ternary relationship as every time an activity is hosted, there is a publisher who hosts the activity and a game associated, which the publisher published and is what the activity about.
Send_gift: A player sends a gift to another player in a game. This is a ternary relationship as every time a player sends a gift to another play, there are two different players involved and the action happens only with a game.
Inform: A publisher informs the VIP community. This is a many-to-many relationship as a publisher can inform multiple VIP communities, and a VIP community can be informed by multiple publishers. The inform relationship has the attributes of special events and special information. This relationship provides the VIP community with a privileged comparing to the other communities.
Attend: A player attends an activity. This is a many-to-many relationship as a player can attend multiple activities and an activity can be attended by multiple players. The attend relationship has an attribute that indicates the number of attendees.

Part 2: ER Diagram
(please view attached)

Part 3: Relational model

User(id, username, password, birthday, district, billing_address, payment_method, email)
Player(id, level, recharge)
Publisher(id, income)
Game(game_id, gname, price, id_publisher, id_player) (id_publisher Ref User, id_player Ref User)
Version(game_id, version_num) (game_id Ref Game)
Post(id, time,com_name, game_id, content) (id Ref User, com_name Ref Community, game_id Ref Game)
Community(game_id, com_name, num_mem, create_time, max-admis) (game_id Ref Game)
High_end_community(game_id, com_name, max_member, min_level) (game_id Ref Game)
VIP_community(game_id, com_name, min_recharge) (game_id Ref Game)

Activity(activity_name, host_time, rules)

friend(user1, user2) (user1 Ref User(id), user2 Ref User(id)) [*Comment: We are concerned about the duplication of friend relationship in the database. In the future implementation, we wish to remove the duplication in the actual database.]*

play(game_id, id, late_play_date) (game_id Ref Game, id Ref User)
send_gift(player1, player2, send_date) (player1 Ref User(id), player2 Ref User(id))
buy(game_id, id, order_id, date) (game_id Ref Game, id Ref User)
publish(game_id, id, date) (game_id Ref Game, id Ref User)
host(activity_name, host_time, id, game_id) (activity_name Ref Activity, host_time Ref activity, id Ref User, game_id Ref Game)
attend(id, activity_name, host_time, num_attend) (id Ref User, activity_name Ref Activity, host_time Ref activity)
inform(game_id, com_name, id, special_events, special_information) (game_id Ref Game, com_name Ref Community, id Ref User)
administrate(game_id, com_name, id) (game_id Ref Game, com_name Ref Community, id Ref User)
join(game_id, com_name, id) (game_id Ref Game, com_name Ref Community, id Ref User)
contains(id, ptime, com_name, game_id) (id Ref User, ptime Ref Post, game_id Ref Game, com_name Ref Community)

Part 4: Creativity and complexity of design

1) How our application domain stands out:

Our application creates an open and communicative environment for the game publishers and players. The database supports the data flow between the different roles on the platforms, making it effective to gather feedback and analyze trends in video gaming industry. For example, by tracking the number of players who buy a specific game through time intervals, we may analyze the popularity and future growth of a game or a genre of games.

2) Use of inheritance

Weak entity sets:

- Post => Community => Game
- Version => Game

ISA subclasses:

- Community is a high-end community or a VIP community. The community is not covered entirely by these two subclasses.
- User could be a publisher or a player

Ternary relationship:

- Host - publisher, game, activity
- Send gift - player, player, game

3) Application of constraints

There should be a maximum number of players who are administors of a community.

Participation constraint:

- In the “administrates” relation between player and community

Key and participation constraint:

- The post can be posted by only one user, and a user can post more than one post
- The post can be contained by only one community, but a community can contain more than one post
- A game can have zero or more communities, but a community can be only associated with one game
- A publisher can publish multiple games, but a game can only be published by one publisher
- A player can buy multiple games, but a game can only be bought once by each player.
- A version update can be only associated one game, but a game can be updated to different versions