<Game Hub>

Database Design Report

<2019.05.15>

Prepared by Group 8

<Marcus 1730026073

Scott 1730026090

Alan 1730026006

Lawrence 1730026045>

Computer Science and Technology Program

United International College

Contents

1. Introduction

1.1 Background

With the continuous development of video games, video games have gradually become an indispensable part of the lives of many young people. But the quality of games on the market is uneven, and the spread of pirated games is very serious.

1.2 Purpose

This project is aimed to create a platform for all players to buy games and to share the experience of some particular games, helping users communicate and interact with their partners to have more fun.

1.3 Scope

GAMEHUB is supposed to be the bridge between players and game developers. Developers can post their games on the GAMEHUB platform to make them more accessible to users. Moreover, GAMEHUB platform also enable users to request refunds whenever they want (requests need to be checked by administrator) and try our best to give users the best service.

2. Assumptions

2.1 Refund Service

User can send a refund request whenever they want. User can choose particular games of one order to refund, but one game can only have one chance to request a refund.

2.2 User type

There are 2 kinds of users in the GAMEHUB system: Customers and Administrators. One administrator is also a customer and has the attributes normal user has. But the identification of customer and administrator is distinguished by an extra attribute. Since that the number of administrators is small and their rights are vital in the system operation, administrator can only be pre-added to the system manually, instead of being registered by the registration process which customer use.

2.3 Account Balance

We did not design an account balance attribute for users. Given the difficulty of importing real-life payment interfaces in this project, we assume that user always have enough balance in their account to directly check out.

2.4 Download Link

There are buttons named download link designed in the user interface for those games they have purchased. We assume that each game has an attribute named download link which store the unique link to download that game after buying it. However, because of copyright issues, we cannot use real links here and also have no needs to use the same link for downloading simulation to cause redundancy in the database. Therefore, we assume that the downloading buttons can work to download corresponding games by access to corresponding page.

2.5 Records Management

There are 2 types of records in the GAMEHUB system: Order record and Refund record. Every record can be edited but we did not design functions to delete it. Considering the importance of transaction records, we do think that GAMEHUB as an intermediate platform should guarantee the integrity of these transaction records.

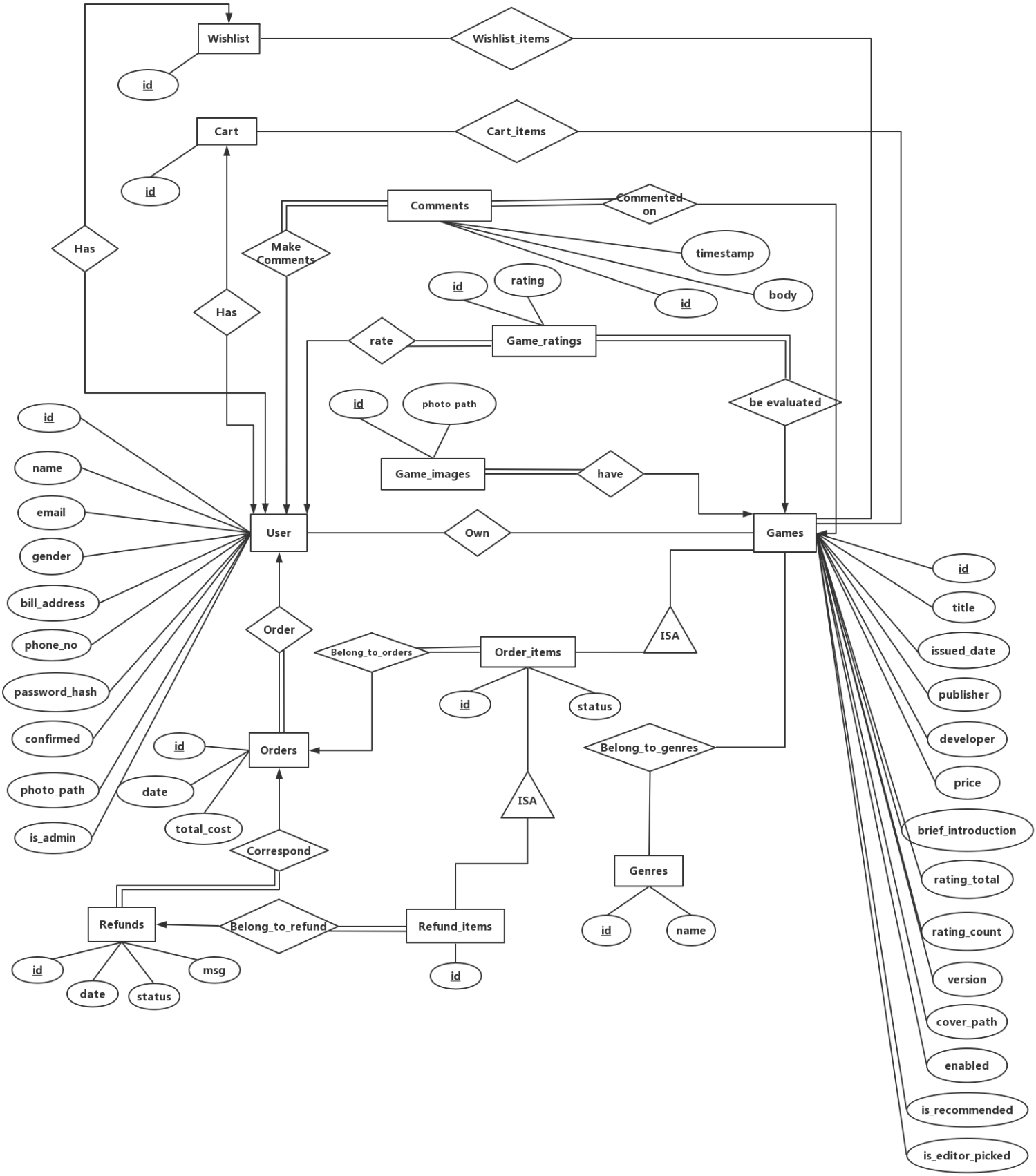
2.6 User Registration

The primary key used to distinguish each user is the User identification which is automatically given by system. But one user account needs one unique corresponding email address to finish the registration. We have no special requests on users' password format.

2.7 Permission

Administrators have the permission to access to or even to edit the information of users, orders, games and refunds (except users' password and id of each entity given by the system), but users can only browse the records of themselves, and they are also not permitted to edit them. (But they can change their personal information except attribute automatically given by the system like id)

3. ER Diagram


Wishlist
Wishlist_items
id
Cart
Cart_items
id
Comments
Commented on
timestamp
body
id
Has
Make Comments
Has
id
rating
Game_ratings
rate
be evaluated
id
photo_path
Game_images
have
id
name
email
gender
bill_address
phone_no
password_hash
confirmed
photo_path
is_admin
User
Own
Games
Order
Belong_to_orders
Order_items
ISA
id
status
id
Orders
Belong_to_genres
date
total_cost
ISA
Correspond
Genres
Refunds
Belong_to_refund
Refund_items
id
name
id
date
status
msg
id
id
title
issued_date
publisher
developer
price
brief_introduction
rating_total
rating_count
version
cover_path
enabled
is_recommended
is_editor_picked

4. Schemas

4.1 User

User (id, name, email, gender, bill_address, phone_no, password_hash, confirmed, photo_path, is_admin)

User represents the real user in GAMEHUB system.

Id: **Primary Key** of user, used to distinguish each user.

Name: The registration name of each user.

Email: The email address user used for registration.

Gender: The gender of the user.

Bill_address: The mailing address of physical bills.

Phone_no: The telephone number of each user.

Password_hash: The password of the user account.

Confirmed: One user account can only operate normally after user clicking the confirming link in the email address he used for registration. Confirmed represents the status whether user has confirmed or not.

Photo_path: The file path of user's profile.

Is_admin: The status indicates whether user is an administrator or not.

4.2 Games

Game (id, title, issued_date, publisher, developer, price, brief_introduction, rating_total, rating_count, version, cover_path, enabled, is_recommended, is_editor_picked)

Games represents the games sold in the GAMEHUB store.

Id: **Primary Key** of game, used to distinguish each game in the system.

Title: The name of the game.

Issued_date: The date when the game is issued.

Publisher: The publisher of the game.

Developer: The developer of the game.

Price: The price of the game.

Brief_introduction: Introduction to the content and characteristics of the game.

Rating_total: The cumulative ratings the game has received.

Rating_count: The number of ratings the game has received.

Version: The current version of the game.

Cover_path: The file path of the cover of the game.

Enabled: The status of whether the game can be searched and sold in the store.

Is_recommended: The status of whether the game is recommended to the home page.

Is_editor_picked: The status of whether the game is picked by editor to the home page.

4.3 Game_ratings

Game_ratings (id, game_id, user_id, rating)

Game_ratings represents the ratings user made for particular games.

Id: **Primary Key** of game_ratings. Id of game_ratings used to distinguish each rating.

Game_id: **Foreign Key referencing Games.** The id of the game be rated.

User_id: **Foreign Key referencing User.** The id of the user makes the rating.

Rating: The value of the rating.

4.4 Game_images

Game_images (id, game_id, photo_path)

Game_images represents the images of the particular games.

Id: **Primary Key** of game_images**.** The id is used to distinguish each image from each other.

Game_id: **Foreign Key referencing Game.** The id of the game the image belongs to.

Photo_path: The file path of the game image.

4.5 Belong_to_genres

Belong_to_genres (genre_id, game_id)

The n to n relationship set represents that one game can belong to many genres and one genre can have many games.

Genre_id and game_id: **Primary Key.**

Genre_id: **Foreign Key referencing Genre.** The id used to distinguish each genre.

Game_id: **Foreign Key referencing Game.** The id of the game.

4.6 Genres

Genres (id, name)

Genres represents the types of games in the system.

Id: **Primary Key** of Genres. The id used to distinguish each genre.

Name: The name of the genre corresponding to each genre id.

4.7 Orders

Orders (id, user_id, total_cost, date)

Orders represents the orders users make when they buy games.

Id: **Primary Key** of Orders. The id used to distinguish each order.

User_id: **Foreign Key referencing User.** The id of the user makes the order.

Total_cost: The total cost of the items in particular order.

Date: The particular date the order is made.

4.8 Order_items

Order_items (id, order_id, game_id, status)

Order_items represents the particular products one user has purchased in one particular order.

Id: **Primary Key** of Order_items. This id is used to distinguish each item of the order.

Order_id: **Foreign Key referencing Order.** The order_id represents the id of the order the item belongs to.

Game_id: **Foreign Key referencing Game.** The id of the game the item actually is.

Status: The status indicates whether the order item is available. (if admin accept the refund request, the game will be no more available for the user to play)

4.9 Refunds

Refunds (id, order_id, date, status, msg)

Refunds represents the refund requests user made.

Id: **Primary Key** of Refunds. This id is used to distinguish each refund record.

Order_id: **Foreign Key referencing Order.** This represents the corresponding order of the refund request. (One order can correspond to many refunds since that there may be many order items).

Date: The specific date the refund is requested.

Status: The process status of the refund request.

Msg: The message administrator sent to the user when process the refund request.

4.10 Refund_items

Refund_items (id, refund_id, order_item_id)

Refund_items represents the particular products one user has request a refund in one particular order.

Id: **Primary Key** of Refund_items. This id is used to distinguish each refund item.

Refund_id: **Foreign Key referencing Refund.** The id of the refund the refund item belongs to.

Order_item_id: **Foreign Key referencing Order_items.** The id of the order item(game) the refund item actually is.

4.11 User_games

User_games (user_id, game_id)

User_games models the relationship set between entity user and entity games.

User_id and game_id: **Primary Key.**

User_id: **Foreign Key referencing User.** The id of the user.

Game_id: **Foreign Key referencing Game.** The id of the game.

4.12 Comments

Comments (id, game_id, user_id, timestamp, body)

Comments represents the comments user made for the particular games.

Id: **Primary Key** of Comments. This id is used to distinguish each comment.

Game_id: **Foreign Key referencing Game.** The id of the game be commented.

User_id: **Foreign Key referencing User.** The id of the user made the comment.

Timestamp: The specific time when user made the comment.

Body: The content of the comment.

4.13 Cart

Cart (id, user_id)

Cart represents the cart which can store the items which users plan to buy.

Id: **Primary Key** of Cart. This id is used to distinguish each cart.

User_id: **Foreign Key referencing User.** The user_id represents the particular user the cart belongs to.

4.14 Cart_items

Cart_items (cart_id, game_id)

Cart_items represents the game product included in a particular cart.

Cart_id and game_id: **Primary Key** of Cart_items.

Cart_id: **Foreign Key referencing Cart.** The id of the cart the item should belongs to.

Game_id: **Foreign Key referencing Game.** The id of the game product in the cart which the item actually is.

4.15 Wishlist

Wishlist (id, user_id)

Wishlist represents the wish list which can store the items which users wish to buy.

Id: **Primary Key** of Wishlist. This id is used to distinguish each wish list.

User_id: **Foreign Key referencing User.** The user_id represents the particular user the cart belongs to.

4.16 Wishlist_item

Wishlist_item (wishlist_id, game_id)

Wishlist_item represents the game product included in a particular wish list.

Wishlist_id and game_id: **Primary Key** of Wishlist_item.

Wishlist_id: **Foreign Key referencing Wishlist.** The id of the wish list the item should belongs to.

Game_id: **Foreign Key referencing Game.** The id of the game product in the wish list which the item actually is.

5. Functional dependency

5.1

User_id -> name, email, gender, bill_address, phone_no, password_hash, confirmed, photo_path, is_admin

5.2

Game_id -> title, issued_date, publisher, developer, price, brief_introduction, rating_total, rating_count, version, cover_path, enabled, is_recommended, is_editor_picked

5.3

Rating_id -> game_id, user_id, rating

5.4

Image_id -> game_id, photo_path

5.5

Genre_id -> name

5.6

Order_id -> user_id, total_cost, order_date

5.7

Order_item_id -> order_id, game_id, status

5.8

Refund_id -> order_id, refund_date, status, msg

5.9

Refund_item_id -> refund_id, order_item_id

5.10

Comment_id -> game_id, user_id, timestamp, body

5.11

Cart_id -> user_id

5.12

Wishlist_id -> user_id

6. Improvement proposals

The content above is a bit different from the actual database design in GAMEHUB system. (Content above is in normal form and has been processed to eliminate redundancy as possible as we can) This part is to show the changes we have made in the database design.

6.1

The original schema of Order is:

Order (id, user_id, bill_address, total_cost, date)

There are 2 functional dependencies in this schema:

- ***Order_id -> user_id, bill_address, total_cost, date***

- ***User_id -> bill_address***

Order_id can be regarded as a super key, but ***User_id -> bill_address*** *violates BCNF.*

Therefore, the decomposition of it should be:

Order (id, user_id, total_cost, date)

R2 (user_id, bill_address)

Since that we already have schema ***User (id, name, email, gender, bill_address, phone_no, password_hash, confirmed, photo_path, is_admin)***, this could be a redundancy if we still keep schema ***R2 (user_id, bill_address)*** in the database. Therefore, out strategy is to directly abandon ***R2 (user_id, bill_address)*** and use:

Order (id, user_id, total_cost, date)

6.2

The original schema of Refunds is:

Refunds (id, order_id, user_id, date, status, msg)

There are 2 functional dependencies in this schema:

- ***Refund_id -> order_id, user_id, date, status, msg***
- ***order_id -> user_id***

Refund_id can be regarded as a super key, but ***order_id -> user_id*** *violates BCNF.*

Therefore, the decomposition of it should be:

Refunds (id, order_id, date, status, msg)

R2 (order_id, user_id)

Since that we already have schema ***Orders (id, user_id, total_cost, date)***, this could be a redundancy if we still keep schema ***R2 (order_id, user_id)*** in the database. Therefore, our strategy is to directly abandon ***R2 (order_id, user_id)*** and use**:**

Refunds (id, order_id, date, status, msg)

7. Workload

Marcus 25%

Scott 25%

Alan 25%

Lawrence 25%