Lucidchart link:
https://lucid.app/lucidchart/4b9e25d0-6bb9-43ad-92f4-1f31b8e211b7/edit?viewport_loc=-11%2C-11%2C939%2C1056%2C0_0&invitationId=inv_6da203d0-893a-4c9b-89d5-f3983b55833d

Group 24 Report on Stage 1 ER Diagram and Translation

Group members: Braeden Bertz, Adrian Gottwein, and Cheng Peng

1. Application description

We are building an online video game distribution service similar to Steam Platform and Epic Game Store. Players who want to purchase games, developers who develop new games, and game publishers are the main involved parties. Developers and publishers can sell their game products on our platform where potential players can buy them. After purchasing and playing a game, players can write reviews to share their own experience on the game and rate the game, which serves as reference to people who are deciding whether to buy this game. To provide these main functionalities and keep track of the transactions on our platform, a database must be used to store business data.

2. ER Diagram

1) Diagram (on the last page)
2) Domain Constraints
   a) Every User has can have exactly 1 wallet that they can use to buy games
   b) Every User can setup exactly 1 profile to express themselves to other gamers
   c) Every Review must be done by a user and must be related to a game. Thus, it forms a weak entity set because it depends on two other entities to form its key.
   d) Every game is either an Indie game or a AAA game
   e) Every game is coded by a developer
   f) Every triple A game is published by a verified publisher to ensure that AAA quality is upheld and that the AAA game is available on every platform that we serve. Indie game devs would have too much trouble implementing on every platform, so they have a field that indicates which platform they were built for.

3. Relational Schema

1. USER(UserId, Password, Username, Email)
   Keys:
   - UserID (Primary)
   - Email, Username
2. WALLET(WalletId, Balance)

Keys:
    - WalletID (Primary)
3. PROFILE(Profile ID, Profile Picture URL, Profile Name)
   Keys:
    - Profile ID (Primary)
4. REVIEWS(UserId, GID, Review Content, Date, Score)
   Keys:
    - UserID, GID (Primary)
5. GAME(GID, Gname, Developer, Genre, Expected Playtime)
    - AAA(GID, Pname)
    - INDIE(GID, Platform, Community URL)

   Keys:
    - GID (Primary)
    - Gname, Developer

6. DEVELOPER(Dname, Preferred Game Engine, Email)
   Keys:
    - Dname (Primary)
    - Email
7. PUBLISHER(Pname, Tax Country, Homepage URL)
   Keys:
    - Pname (Primary)

ISA relationship:
We decided to go with the E/R approach. The null approach would condense the entities into a single table, with nulls in the columns, and while this would make for an easy place to find all necessary information about games, it leads to lots of null columns (having harder queries) and forces queries to be slower (since a query for indie games would have to go through all indie and AAA games to find). The OO approach is a viable alternative, however its use case (find the color of ales made by Pete's) is not our use case. The use case of the E/R approach (E/R approach good for queries like "find all beers (including ales) made by Pete's.")
Our use case is most often finding a game by name (in which case we just query GAME) or by finding games based on their quality (AAA vs INDIE) and filtering with usually genre or platform. Thus, "Find all games where gname = ??" and "find all INDIE games where platform = ?? and Developer = ??" can both be applied to the E/R style very easily and fast.

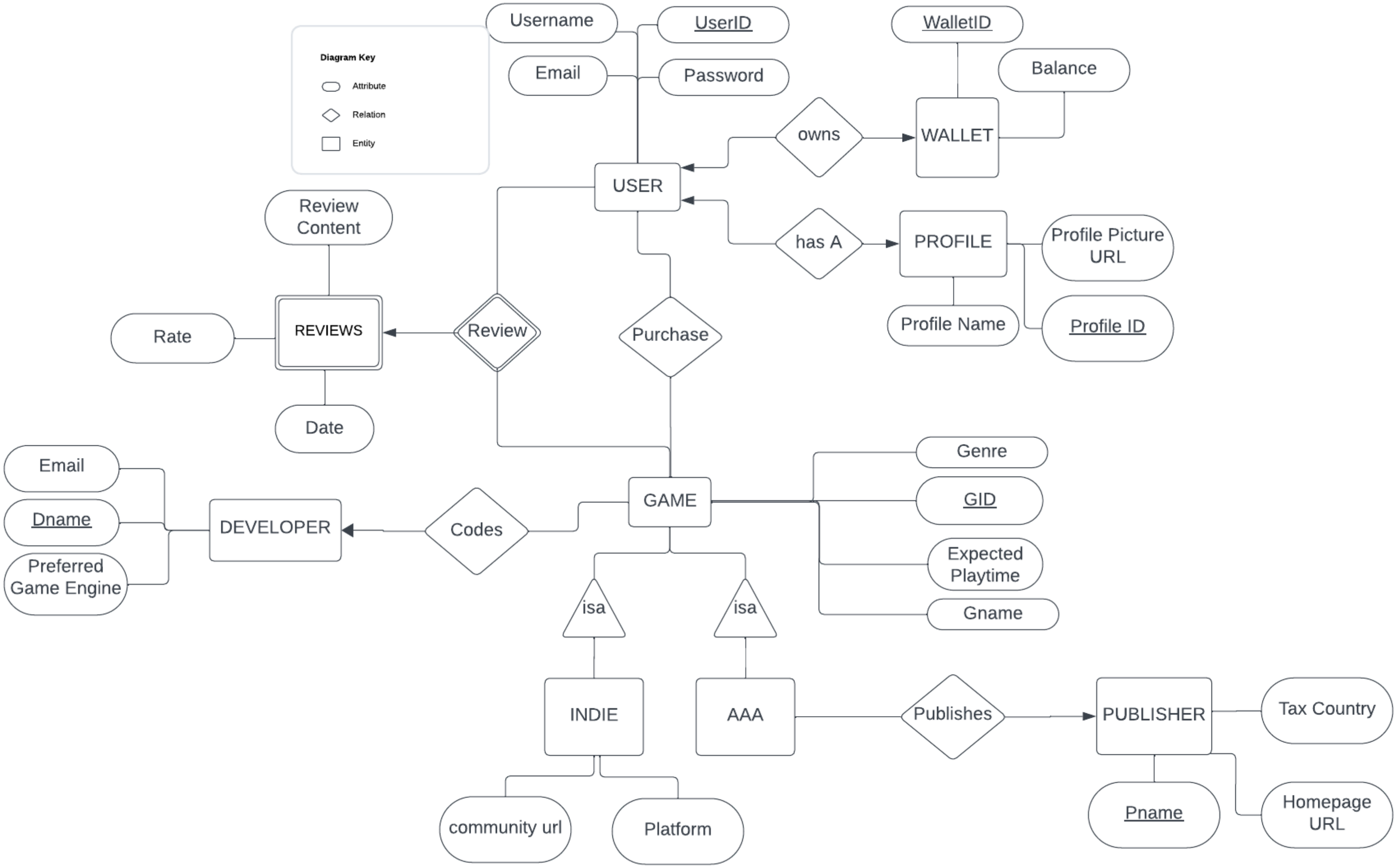
Diagram Key
Attribute
Relation
Entity
Username
UserID
Email
Password
WalletID
Balance
owns
WALLET
USER
has A
PROFILE
Profile Picture URL
Review Content
Profile Name
Profile ID
Rate
REVIEWS
Review
Purchase
Date
Email
Dname
Preferred Game Engine
DEVELOPER
Codes
GAME
Genre
GID
Expected Playtime
Gname
isa
isa
INDIE
AAA
Publishes
PUBLISHER
Tax Country
community url
Platform
Pname
Homepage URL