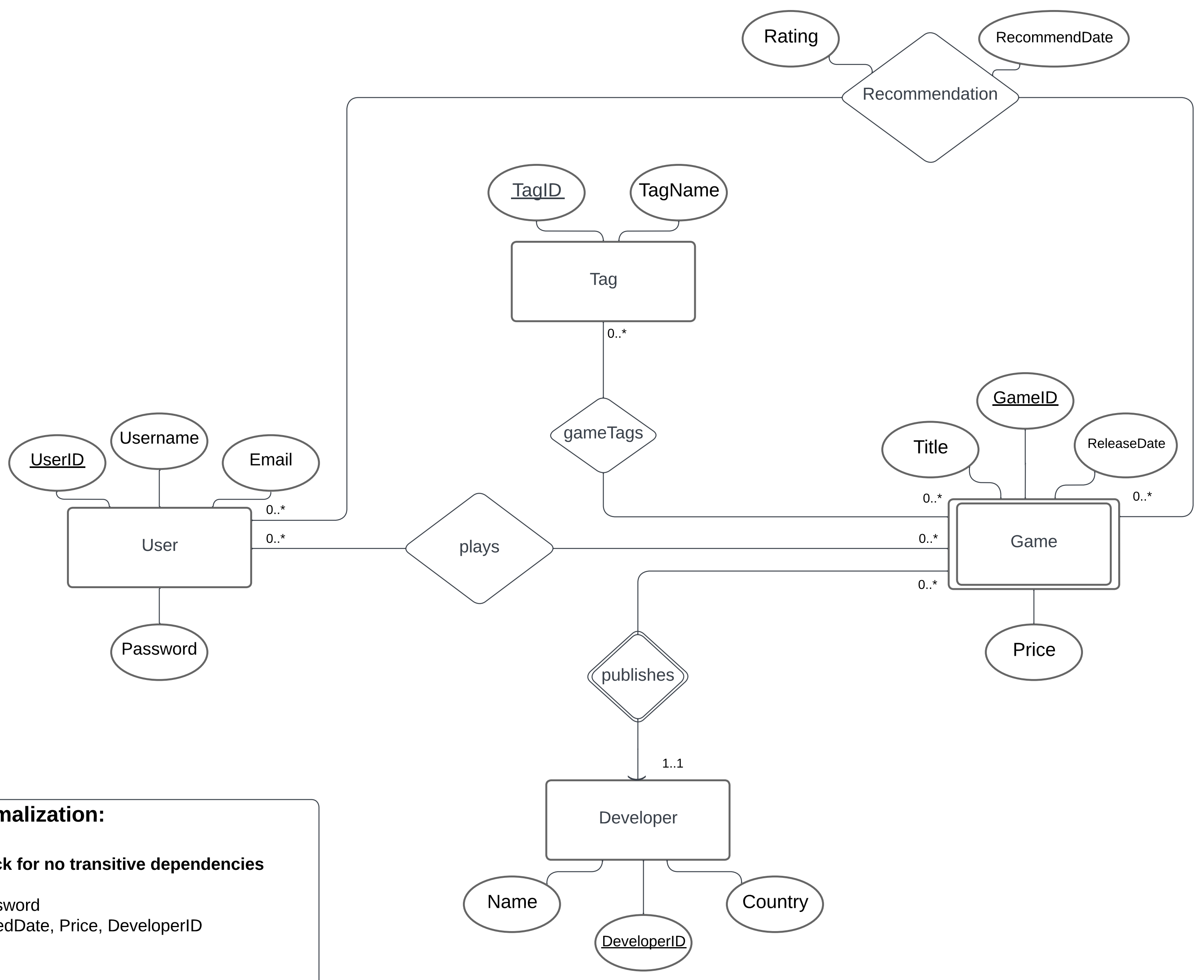

Normalization:

Third Normal Form (3NF) - check for no transitive dependencies

UserID -> Username, Email, Password
GameID -> GameName, ReleasedDate, Price, DeveloperID
DeveloperID -> Name, Country
TagID -> TagName

All attributes go directly to their primary keys
There is no non-key attribute that rely on another non-key attribute
Therefore, no transitive dependencies.

Recommendation - > UserID, GameID, Rating, RecommendationDate

Rating and RecommendDate go directly on (UserID, GameID) the composite key.
There is no non-key attribute that rely on another non-key attribute
Therefore, no transitive dependencies.

gameTags -> TagID, GameID
plays -> UserID, GameID

These tables support the many-to-many relationship between their two entities.
There is no non-key attribute that rely on another non-key attribute
Therefore, no transitive dependencies.

Relational Schema:

User(UserID:Int [PK], Username:VARCHAR(40), Email:VARCHAR(255), Password:VARCHAR(255))

Game(GameID:INT [PK], GameName:VARCHAR(255), ReleasedDate:DATE, Price:FLOAT, DeveloperID: INT[FK to Developer.DeveloperID])

Recommendation(UserID:INT [FK to User.UserID], GameID:INT [FK to Game.GameID], Rating:INT, RecommendDate:DATE)

Developer(DeveloperID:INT[PK], Name:VARCHAR(255), Country:VARCHAR(255))

Tag(TagID:INT [PK], TagName:VARCHAR(225))

gameTags(TagID:INT [FK to Tag.TagID], GameID:INT [FK to Game.GameID], TagID:INT[PK], GameID;INT[PK])

plays(UserID:INT [FK to User.UserID], GameID:INT [FK to Game.GameID], UserID:INT[PK], GameID;INT[PK])

Assumptions

1. User: This represents a person using the application. Each user has unique info and preferences.

2. Game: This represents a game available in the application. Each game has unique details and pricing information.

3. Recommendation: This represents a recommendation made to a user based on their preferences and past interactions.

4. Tag: This represents either a category or genre that a game falls under on.

5. Developer: Represents who or what team help develop a game.

--Explanations--

User and Game as separate entities: These represent fundamental entities with their own distinct attributes and relationships, which justify modeling them separately.
Developer: Modeled as a seperate entity to help give Games more of a category through who or what team is behind the games and its regional origin
Recommendation: Separate entity to track game recommendations to users, allowing for detailed tracking and timestamping.
Tag and gameTags: Since one game can have multiple genres/categories, it is useful to have gameTag as a separate entity. This is a many-to-many relationship.

Relationships

User-Game (Many-to-Many through **plays**)
- A user can play multiple games, and a game can be played by multiple users.

User-Game (Many-to-Many through **Recommendation**)
- A user can receive multiple game recommendations, and a game can be recommended to multiple users.

Game-Tags (Many-to-Many through **gameTags**)
- A game can be tagged to multiple genres, and a tag can be associated to multiple games.

Game-Developer (Many-to-One through **publishes**)
- A game can only be published from one developer group or team, and a a developer group or team can publish multiple games.