**Game Network**

**Database**

**Implementation Report**

Stryder Coleman, Jesse Beach, Zach Baccaro, Emily Hampston, and Brandon Marcellette

Group 3

**CIS 422**

**Physical Design**

**&**

**Implementation of**

**Database Management**

**Systems**

**Table of Contents**

Introduction …………………………...2

Statement of Objectives ……………..2

Requirements …………………....…...2

Design (Revisited) ……………………3

Schema …………………...……..3

Data Dictionary ………………….4

Constraints ……………………....7

Queries ......................................8

Requirements Matrix .................8

Entity-Relationship Diagram ......9

Implementation ................................10

Queries in SQL .........................10

Triggers .....................................13

Test Data ...................................14

Testing Procedure .....................17

**Introduction**

This document attempts to summarize the requirements, analysis and design of a database system designed to support a game network, including a message system, achievements, friend system, game library, and a store.

**Statement of Objectives**

The goal of the project is to design a game network database that can facilitate the functions needed for a large network. The database should be able to host users, games, store listings, and achievements, as well as allowing for users to own games, add friends, and earn achievements. It should be able to display information about all entities within the database.

**Requirements**

The requirements list is as follows:

1. The database must allow users to have profiles.
2. The database must have developers for games.
3. The database must provide functions that allow users to send messages to each other.
4. The database must provide functions that allow users to earn achievements.
5. The database must provide functions that allow users to friend other users.
6. The database must allow users to own games.
7. The database must not allow users under 18 to own M rated games.
8. The database must contain records of games owned by users in their game library.
9. The database must support a store that has games listed to it.
10. The database must allow games to exist, but are not listed to the store.
11. The database must allow users to own games

**Design (Revisited)**

**Schema**

User(userid CHAR(60), password CHAR(50), dateCreated DATE, sex CHAR(1), dateOfBirth DATE)

KEY {userid}

Developer(studioName:CHAR(60))

KEY {studioName}

Game(gameid:CHAR(60), listingID:INTEGER, title:CHAR(100), description:CHAR(255),

size:DOUBLE rating:CHAR(2))

KEY {gameid}

Store(listingID INTEGER, genre CHAR(20), releaseDate DATE)

KEY {listingID}   
 FOREIGN Key listingID REFERENCES Game(listingID)

Message(mId INTEGER, userID CHAR(60), recipient: CHAR(60), text:CHAR(1000))

KEY {mID, recipient}

FOREIGN KEY userID REFERENCES User (userID)

FOREIGN KEY recipient REFERENCES User(userID)

Achievement(achID INTEGER, title CHAR(100), type CHAR(15), gameid INTEGER)

KEY{achID}

FOREIGN KEY gameid REFERENCES Game(gameid)

Earns(userid CHAR(60), achID INTEGER)

KEY{userid, achID}

FOREIGN KEY userid REFERENCES User (userid)

FOREIGN KEY achID REFERENCES Achievement (achID)

IsFriendsWith(userid CHAR(60), friend CHAR(60))

KEY{userid, friend}

FOREIGN KEY userid REFERENCES User (userid)

FOREIGN KEY friend REFERENCES User (userid)

Develops(studioName:CHAR(60), gameid:CHAR(60))

KEY {studioName, gameid}

FOREIGN KEY gameid REFERENCES Game

FOREIGN KEY studioName REFERENCES Developer(studioName)

Owns(userid:CHAR(60), gameid:CHAR(60))

KEY{userid, gameid}

FOREIGN KEY userid REFERENCES User(userid)

FOREIGN KEY gameid REFERENCES Game(gameid)

**Data Dictionary**

User

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| userid | CHAR(60) | Key for a user | user1 |
| password | CHAR(50) | Password to user’s account | 1234 |
| dateCreated | DATE | Date account created (Format: mm-dd-yyyy) | 10/26/2020 |
| sex | CHAR(1) | Sex of user | M |
| dateOfBirth | DATE | Date user was born (Format: mm-dd-yyyy) | 01-27-1999 |

Developer

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| studioName | CHAR(60) | Name of Studio, Key for Developer | Blizzard |

Game

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| listingID | INTEGER | Key for game, used to list to store | 12342 |
| title | CHAR(100) | Title of game | GameName |
| description | CHAR(255) | Description of game | The game is fun |
| size | DOUBLE | Data size of game | 12 |
| rating | CHAR(2) | Rating of game | M |

Store

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| listingID | INTEGER | Key for a game in the store | 12345 |
| genre | CHAR(20) | The genre of the game | Adventure |
| releaseDate | DATE | Release Date of Game | 1/1/0001 |

Message

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| mID | INTEGER | Key of message | 1234 |
| recipient | CHAR(60) | Id of user who will receive message | user2 |
| text | CHAR(1000) | Body of message being sent | Hello, how are you |
| userid | CHAR(60) | User who is sending message | user1 |

Achievement

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| achID | INTEGER | Key for an achievement | 1234 |
| title | CHAR(100) | Title of achievement | Played 1 game |
| type | CHAR(15) |  |  |

Earns

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| userid | CHAR(60) | ID of user who earned achievement | user1 |
| achID | INTEGER | Key for earns, ID of achievement | 1234 |

isFriendsWith

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| userid | CHAR(60) | Key for a user | user1 |
| friend | CHAR(60) | User that is being friended | user2 |

Develops

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| studioName | CHAR(60) | Key of developer | Blizzard |
| listingid | CHAR(60) | Id of game being developed | 12342 |

Owns

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Defines | Example |
| userid | CHAR(60) | Key of user, person who owns the game | user1 |
| listingid | CHAR(60) | Key of game that is owned | 12342 |

**Constraints**

Domain constraints:

* dateOfBirth and createdDate cannot be passed the current date.
* The value of sex can either be ‘M’, ‘F’, or ‘O’.
* Genres can be ‘Adventure,’ ‘Racing,’ ‘Sports,’ ‘Strategy.’
* Ratings can be ‘E’, ‘T’, or ‘M.’

Key constraints:

* Every User must have a unique userId value.
* In the IsFriendsWith table, the foreign keys userid and friend must reference unique values since a user cannot be friends with themself.
* For IsFriendsWith, the userid and friend values must have a unique combination.
* Every Game must have a unique gameid value and listingId value.
* Every Achievement must have a unique achid value.
* Every Developer has a unique studioName.
* For Earns, the userid and achid must have a unique combination.
* Users under 18 cannot own M rated games

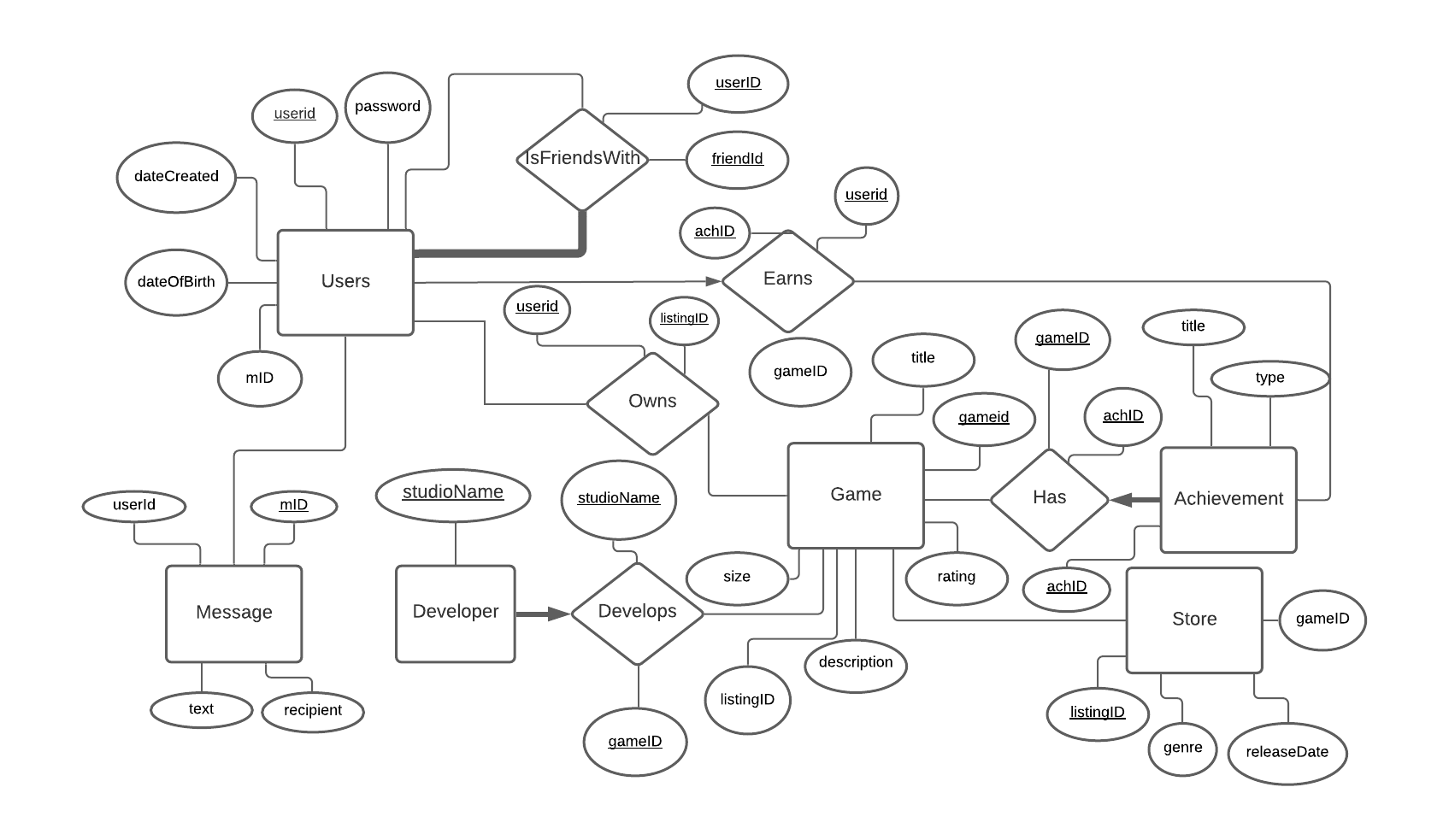
**Queries**

1. Create a user using the attributes from the User table.
2. Add two unique userid values to a record in the IsFriendsWith table, making two users friends.
3. Create a game using the attributes from the Game table.
4. Create an achievement for a game using the attributes of the Achievement table.
5. Add userid and achID to a record in the Earns table, signifying a user has earned an achievement.
6. Add two userid values to a record in the Messages table, allowing a user to message another user.
7. Add a game’s listingID to a record in the Store table, showing that store sells the designated game.
8. Create a developer using the attributes from the Developer table.
9. Create a Develops relation using the attributes in the Develops table, showing that a developer is developing a certain game.
10. Create an Owns relation using the attributes in the Owns table, showing a user owns a certain game.

**Requirements Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Query** | **req1** | **req2** | **req3** | **req4** | **req5** | **req6** | **req7** | **req8** | **req9** | **req10** | **req11** |
| **1** | **x** |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  | **x** |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  | **x** |  |
| **4** |  |  |  | **x** |  |  |  |  |  |  |  |
| **5** |  |  |  | **x** |  |  |  |  |  |  |  |
| **6** |  |  | **x** |  |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  |  |  |  | **x** |  |  |
| **8** |  | **x** |  |  |  |  |  |  |  |  |  |
| **9** |  |  |  |  |  |  |  |  |  |  | **x** |
| **10** |  |  |  |  |  | **x** | **x** | **x** |  |  |  |

**Entity Relational Diagram**



**Implementation**

**Queries in SQL**

CREATE TABLE Users(

userid CHAR(60),   
 password CHAR(50),

dateCreated date default getDate() check(dateCreated < getDate()),

sex CHAR(1) check (sex in('M','F', 'O')),

dateOfBirth DATE check(dateOfBirth < getDate()),

PRIMARY KEY(userid));

CREATE TABLE Developer(

studioName CHAR(60),

PRIMARY KEY (studioName))

CREATE TABLE Game(

listingid INTEGER,

title CHAR(100),

description CHAR(255),

size DOUBLE,

rating CHAR(1) check(rating in('E', 'T', 'M')),

PRIMARY KEY(listingid))

CREATE TABLE Store(

listingid INTEGER,

genre CHAR(20) check(genre in('Action', 'Sports', 'Strategy', 'RPG', 'Adventure')),

releaseDate DATE,

PRIMARY KEY(listingid),

FOREIGN KEY(listingid) REFERENCES Game(listingid)

ON DELETE RESTRICT)

CREATE TABLE Message(

mId INTEGER,

userid CHAR(60),

recipient CHAR(60),

text CHAR(255),

PRIMARY KEY (mId),

FOREIGN KEY(userid) REFERENCES User(userid),

FOREIGN KEY(recipient) REFERENCES User(userid))

CREATE TABLE Achievement(

achID INTEGER,

title CHAR(100),

type CHAR check(type IN(‘Bronze’, ‘Silver’, ‘Gold’)),

PRIMARY KEY(achID))

CREATE TABLE Has(

achID INTEGER,

listingid INTEGER,

PRIMARY KEY(achID, listingid),

FOREIGN KEY(achID) REFERENCES Achievement(achID),

FOREIGN KEY(listingid) REFERENCES Game(listingid))

CREATE TABLE Owns(

userid CHAR(60),

listingid INTEGER,

PRIMARY KEY(userid, listingid),

FOREIGN KEY (userid) REFERENCES User(userid),

FOREIGN KEY (listingid) REFERENCES Game(listingid))

CREATE TABLE Earns(

userid CHAR(60),

achID INTEGER,

PRIMARY KEY(userid, achID),

FOREIGN KEY(userid) REFERENCES User (userid),

FOREIGN KEY(achID) REFERENCES Achievement (achID))

CREATE TABLE isFriendsWith(   
 userid CHAR(60),   
 friend char(60),   
 PRIMARY KEY(userid, friend),   
 FOREIGN KEY(userid) REFERENCES Users (userid),   
 FOREIGN KEY(friend) REFERENCES Users(userid),   
 check(userid <> friend))

CREATE TABLE Develops(

studioName CHAR(60),

listingid INTEGER,

PRIMARY KEY(studioName, listingid),

FOREIGN KEY(listingid) REFERENCES Game (listingid),

FOREIGN KEY(studioName) REFERENCES Developer(studioName))

Queries

1. INSERT INTO User(userid, password, dateCreated, sex, dateOfBirth) VALUES('user1', 'password', '2020-11-23', ‘F’, 2000-1-5');
2. INSERT INTO `isFriendsWith`(`userid`, `friend`) VALUES ('User1', 'user2');
3. INSERT INTO `Game`(`listingid`, `title`, `description`, `size`, `rating`) VALUES ('1', 'COD', 'FPS', '50', 'M');
4. INSERT INTO `Achievement`(`achID`, `title`, `type`,) VALUES ('1', 'Legendary', ‘Gold’);
5. INSERT INTO `Earns`(`userid`, `achID`) VALUES ('User1', '1');
6. INSERT INTO `Message`(`mId`, `userid`, `recipient`, `text`) VALUES ('1', 'User1', 'User2', 'Hello');
7. INSERT INTO `Store`(`listingid`, `genre`, `releaseDate`) VALUES ('1', 'Action', '2020-12-31');
8. INSERT INTO `Developer`(`studioName`) VALUES ('Studio1');
9. INSERT INTO `Develops`(`studioName`, `listingid`) VALUES ('Studio1', '1');
10. INSERT INTO `Owns`(`userid`, `listingid`) VALUES ('User1', '1');

**Triggers**

CREATE TRIGGER ageCheck ON Owns

AFTER INSERT

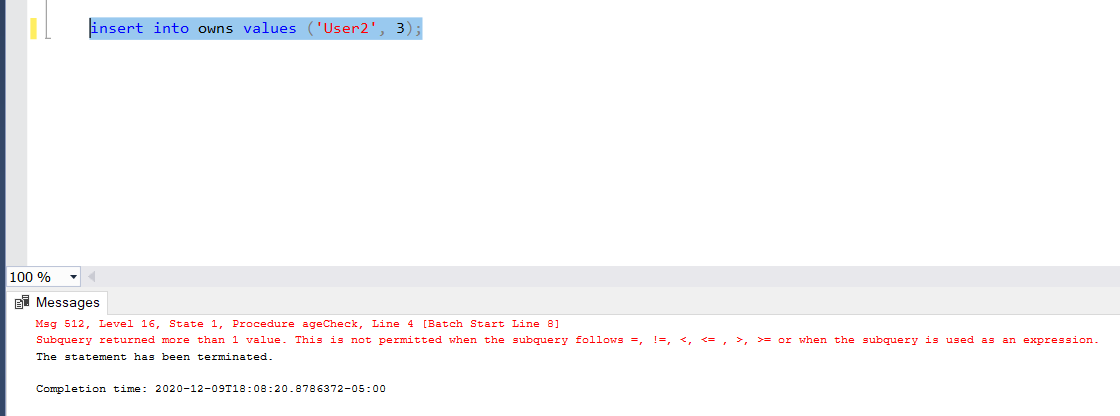
AS

IF(((select rating from game where listingid = NEW.listingid) = 'M') AND ((select dateOfBirth from user where userid = NEW.userid) >= getdate()-18))

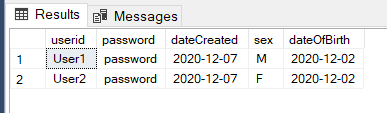
BEGIN

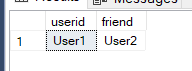
ROLLBACK;

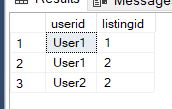
END;

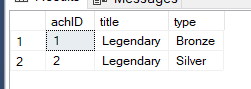
Trigger Working:

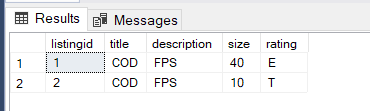
**Test Data**

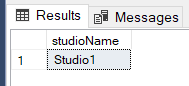
User Table:

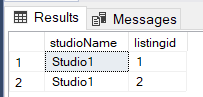
IsFriendsWith Table:

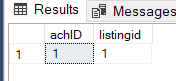
Owns Table:

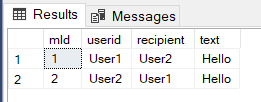
Achievement Table:

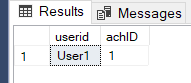
Game Table:

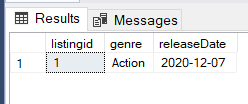
Developer Table:

Develops Table:

Has Table:

Message table:

Earns Table:

Store table:

**Testing Procedure**

