**COVER PAGE**

**Course: CS 4318- Database Systems**

**Date: May 1, 2022**

**Contributors: Fabian Quijano, Bradley Pham**

**Abstract**

The database we worked on is themed around video games. We proposed to work on this theme because it’s one of the top medias in the entertainment industry. Many of us have played video games in some form since it is one of the best entertainments in the media industry. This database application would show the various relationships videogames have from people’s gameplay and within the industry.

**Mission Statement**

Creating a database that showcases video games. Also learn its attributes such as genre, consoles used, development, releases and more.

**Mission Objectives**

Firsthand objectives are to have multiple players register their gamertag, platform they play on, and the videogames they enjoy.

Secondhand objectives are to have the player learn and enjoy playing video games. Additionally, ensure that the player has a great experience while playing. Make sure that video games have little to no bugs and glitches so that they don’t interfere with the player’s enjoyment.

**ER Diagram**

The primary keys are highlighted and boxed in red

**Diagram

Description automatically generated**

****

****

****

****

****

**Relational Model**

**Relations:**

Developer(DeveloperID, CompanyName, Location, NumberofGamesDeveloped, NumberofEmployees, YearFounded)

MediaCompany(MediaID, Name, Location, NumberOfEmployees, YearFounded)

PersonalInfo(FullName, City, State, Country, Region)

Platform(PlatformID, Name, ReleaseDate, NumberOfPlayers)

Players(GamerID, Gamertag)

Publisher(PublisherID, Name, Founders, Location, NumberofGamesPublished, NumberOfEmployees, YearFounded)

VideoGame(VideoGameID, Title, Genre, ReleaseDate)

PlayersBackground(FullName, GamerID, City, State, Country, Region)

PlayersPlays(VideoGameID, Title, Genre, ReleaseDate)

PlayersPlayOn(PlatformID, Name, ReleaseDate, NumberOfPlayers)

PlatformCreatedBy(MediaID, Name, Location, NumberOfEmployees, YearFounded)

VideogameReleasedOn(PlatformID, Name, ReleaseDate, NumberOfPlayers)

VideoGameDevelopedBy(DeveloperID, CompanyName, Location, NumberofGamesDeveloped, NumberofEmployees, YearFounded)

VideoGamePublishedBy(PublisherID, Name, Founders, Location, NumberofGamesPublished, NumberOfEmployees, YearFounded)

**Function Dependencies:**

DeveloperID -> CompanyName, Location, NumberofGamesDeveloped, NumberofEmployees, YearFounded

MediaID -> Name, Location, NumberOfEmployees, YearFounded

FullName -> City, State, Country, Region, GamerID

PlatformID -> Name, ReleaseDate, NumberOfPlayers

GamerID -> Gamertag

PublisherID -> Name, Founders, Location, NumberofGamesPublished, NumberOfEmployees, YearFounded

VideoGameID -> Title, Genre, ReleaseDate

They are all in 3NF because they can’t be broken down into smaller entities. Deleting the IDs for any entities should delete its data in its entirety.

**Actor Use Cases**

These actor use cases are ideal

1. Actor: Ling Xu

Steps: Professor Ling Xu can register a new gamertag. Before registering the gamertag, it requires her personal information. After she completes it, it gets stored in the personalinfo entity and then she can register a gamertag, then the system would assign her a unique gamerID. It then asks her the games she has played or has interest in playing. The system picks this up and stores it in the videogame entity

1. Actor: New Videogame

Steps: A new videogame would have to go under many revisions in the developer entity. After it has been completed, it would go under review in the publisher entity. If the publisher entity approves of the product, it will be published on certain platforms, or in the platform entity. It will generate all the number of sales in the videogame entity and the number of videogames published will increase in the publisher entity

1. Actor: A new platform

Steps: The media company will have a time frame in between platform releases. Companies like Sony and Microsoft are always competing with the releases dates and the specs of their new platform. For example, when the new generations of consoles are released, they would have a brand new platformID corresponding to their medicompanyID. After a few months, the number of players will accumulate, and it will be updated in the platform entity.

1. Actor: A new developer company

Steps: Whenever a startup developer company begins its journey, they are not promoted, so they must create and release their games separately. Although they are independent now, they’re number of games develop still increase to gain attraction. When publishing companies recognize them, the developers will be promoted by the publishers. The publishers will now have more games published in their publisher entity and developers would now have the chance more games at a faster pace.

**Use Cases Implementation**

1. PersonalInfo:

* Aggregate Query: SELECT COUNT(State) AS AmountOfPeople

FROM PersonalInfo

WHERE State = ‘Texas’;

* Insert Query: INSERT INTO PersonalInfo(FullName, GamerID, City, State, Country, Region)

VALUES(‘Ling Xu’, 10 , ‘Houston’, ‘Texas’, ‘United States’, ‘North America’);

* Delete Query: DELETE FROM PersonalInfo

WHERE GamerID = 10;

* Update Query: UPDATE PersonalInfo

SET City = ‘Los Angeles’, State = ‘California’

WHERE GamerID = 10;

1. Platform:

* Aggregate Query: SELECT SUM(NumberOfPlayers)

FROM Platform;

* Insert Query:  INSERT INTO Platform(PlatformID, Release Date, NumberOfPlayers, Name)

Values(22, ‘March 18, 2022’, 1234567, Ninbox);

* Delete Query: DELETE FROM Platform

WHERE PlatformID = 15;

* Update Query: UPDATE Platform

SET NumberOfPlayers = 2002009;

1. MediaCompany

* Aggregate Query: SELECT MAX(NumberOfEmployees) FROM MediaCompany;
* Insert Query: INSERT INTO MediaCompany VALUES (‘NumberOfEmployees’, ‘Year Founded’, ‘MediaID’, ‘Location’, ‘Name’);
* Delete Query: DELETE FROM MediaCompany WHERE MediaID;
* Update Query: UPDATE MediaCompany

SET NumberOfEmployees = 100;

* Join Query: SELECT \* FROM Publisher, MediaCompany

WHERE Location = Location

1. Developer

* Aggregate Query: SELECT MAX(NumberOfGamesDeveloped)

FROM Developer;

* Insert Query: INSERT INTO Developer VALUES (‘Company Name’, ‘DeveloperID’, ‘Developed By’, ‘NumberOfEmployees’, ‘NumberOfGamesDeveloped’, ‘YearFounded’);
* Delete Query: DELETE FROM Developer WHERE Location;
* Update Query: UPDATE Developer SET NumberOfGamesDeveloped = 10;
* Join Query: SELECT \*

FROM Publisher, Developer

WHERE YearFounded = YearFounded

1. Players

* Aggregate Query: SELECT MIN(GamerID) FROM Players;
* Insert Query: INSERT INTO Players VALUES (‘Plays On’, ‘Plays’, ‘Background’, ‘Gamertag’, ‘GamerID’);
* Delete Query: DELETE FROM Players WHERE Gamertag;
* Update Query: UPDATE Players SET GamerID = 374582;
* Join Query: SELECT \* FROM VideoGame, Players WHERE VideoGameID < GamerID

1. Publisher

* Aggregate Query: SELECT MAX(NumberOfGamesPublished)

FROM Publisher;

* Insert Query: INSERT INTO Publisher VALUES (‘Founders’, ‘YearFounded’, ‘Name’, ‘NumberOfEmployees’, ‘NumberOfGamesPublished’, ‘PublisherID’);
* Delete Query: DELETE FROM Publisher WHERE Location;
* Update Query: UPDATE Publisher

SET NumberOfEmployees = 50;

* Join Query: SELECT \* FROM Publisher, Developer

WHERE NumberOfEmployees = NumberOfEmployees

1. VideoGame

* Aggregate Query: SELECT MAX(NumberOfSales)

FROM Video Game;

* Insert Query: INSERT INTO VideoGame VALUES (‘NumberOfSales’, ‘Title’, ‘Release Date’, ‘Genre’, ‘VideoGameID’);
* Delete Query: DELETE FROM Video Game WHERE VideoGameID;
* Update Query: UPDATE Video Game

SET NumberOfSales = 100,000;

* Join Query: SELECT \*

FROM Video Game, Players

WHERE VideoGameID > GamerID

**Tests**

At the beginning of our rough draft for the database, I was already able to see the issues. There weren’t any foreign keys implemented in our ER diagram. So, I updated our ER diagram, and this was able to solve the issue when doing any kind of query. I was able to do most join queries, but for some reason, the database removed the foreign keys to three relationships. It was already too late for me to add them back. But I was able to showcase the other join queries successfully.

**Sample Query output from each use case:**

1. PersonalInfo: Aggregate Query

Output: Number of Texans in total

1. Platform: Aggregate Query

Output: Total number of players from all platforms

1. MediaCompany: Aggregate Query

Output: Displays the company with the greatest number of employees

1. Developer: Aggregate Query

Output: Displays the company with the greatest number of games developed

1. Players: Aggregate Query

Output: Displays minimum amount of gamers

1. Publisher: Aggregate Query

Output: Displays the company with the most amount of games published

1. VideoGame: Aggregate Query

Output: Displays the videogame who had sold the most

**Conclusion**

In conclusion, we were able to successfully display the contents of the database. We went and modified our ER to best fit our needs. Each table had a set of data corresponding to the theme of videogames. From this class we learned the overall importance form database systems, from its models, schemas, algebraic functions, and implementation. We hope to increase our knowledge in databases soon with our post grad jobs.

**REFERENCES**

[SQL Tutorial (w3schools.com)](https://www.w3schools.com/sql/)

[MySQL :: Download MySQL Workbench](https://dev.mysql.com/downloads/workbench/)