VideoGame Store

A databases project

Francisco José de Caldas District University

Teacher - Carlos Andrés Sierra Virguez

Databases Foundations

Students

- Diego Alejandro Garzón **- 20222020009**

- Daniel Josué Rozo Rodríguez **- 20222020006**

2024-I

Technical document of the project

1. Description of the project

The project consists of designing a database that simulates an online video game store, specifically looking for this to be a replica of the famous desktop application “Steam”. The idea is to provide, in a simple and as objective way as possible, a design that adapts to the proposed needs, maintains a stable and large volume of data (as it usually is), but at the same time simplifies the entire system in the best way possible. We are two members in charge of carrying out the project.

Steam by default is a desktop application, although it is also designed to display the catalog and features in a web browser (web application), but we will not focus much on that. It has an initial interface that receives the data of “username” and the respective password. The business model of the application is simple, sales, since it is a virtual store, handles the purchase of video games, live broadcasts of them by other users (to promote the products), a personal library for each user and a “community” section that at the same time includes its own “mini market”. The application also has a section of free games where, as in the previous section, we can purchase and install them from the application.

Inside its interface it manages a menu with six options and a keyboard search bar. It divides the games into their different categories, prices and other features. It's a complete shop.

The tools we will use are:

- PostgreSQL
- Python
- Draw.IO
- DBeaver CE

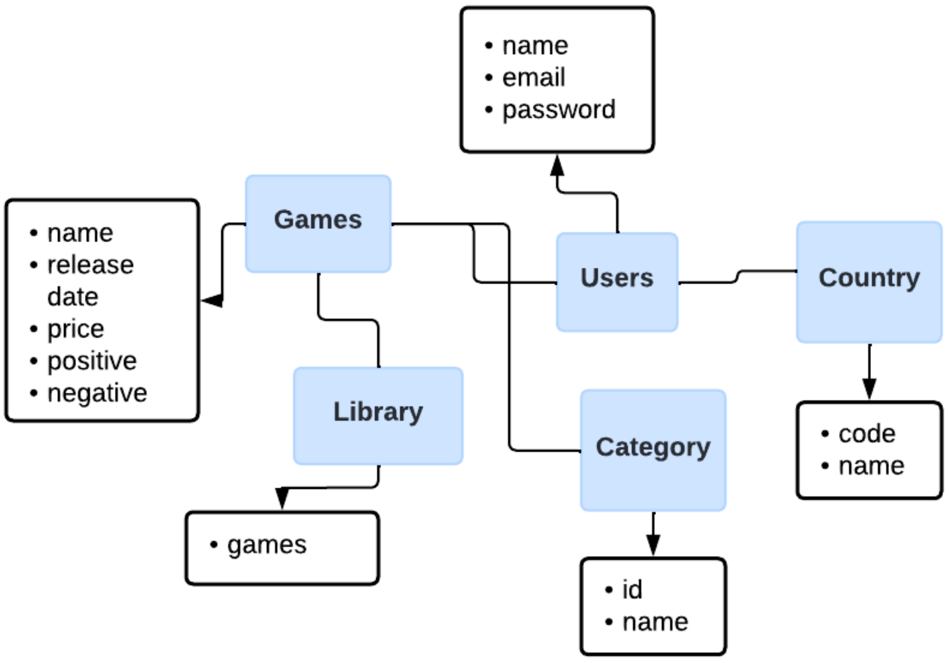
It is important to clarify that the system itself has an immense magnitude and scope, given that it is an application of global character, the idea is to replicate it but for practical purposes soon there will be functions with which application counts that come to be discarded in the final model. Not because we don't want to replicate it well, but because we will take more account of the most vital and important features that make the app one of the most used online stores today.

1. User stories:

- As a user I want to see the catalogs of the games organized by categories
- As a user I want to see the number of positive or negative reviews that a game could have, that to know how much good or bad is a specific game
- As a user I want to see the price of every game in the store
- As a user I want to know the release data of the games in the store
- As a user I want a library, where I can see all the games I have purchased or downloaded
- As a user I want to be shown all the features and details of a game when accessing the download page of the same
- As a user I want to know how many downloads has a specific game

2. Create a conceptual model of your project:

- **Step 0. Define components:**



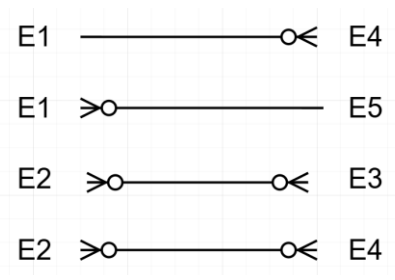
- **Step 1. Define entities:**
- **E1.** User
- **E2.** Games
- **E3.** Category
- **E4.** Library
- **E5.** Country

- **Step 2. Define attributes per entity:**
- **E1:** user_id**, user_n**ame, email, password
- **E2:** games_id, name, release_date, price, positive, negatives, downloads
- **E3:** category_id, name
- **E4:** library_id, games
- **E5**: country_id, name, acronyms

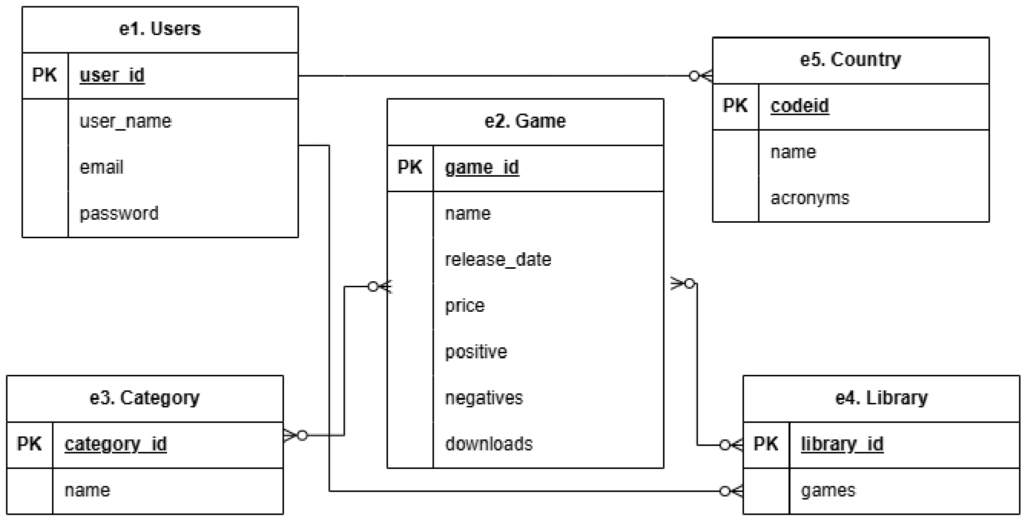
- **Step 3. Define Relationships:**

🟥	**E1**	**E2**	**E3**	**E4**	**E5**
E1	🟥			🟩	🟩
E2		🟥	🟩	🟩	
E3		🟩	🟥		
E4	🟩	🟩		🟥	**3**
E5	🟩				🟥

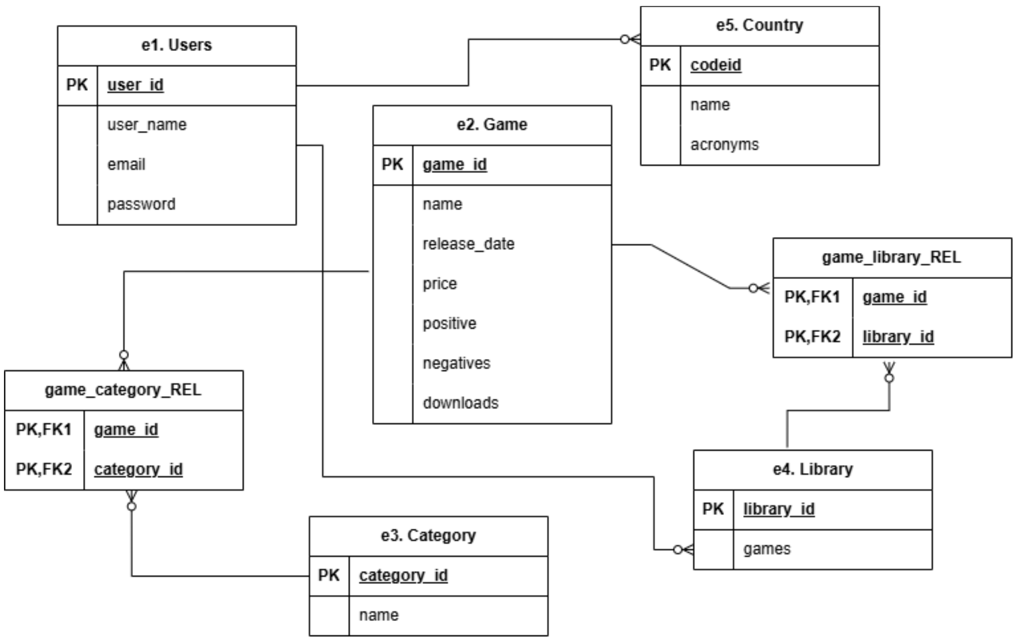
- **Step 4. Define relationships type:**



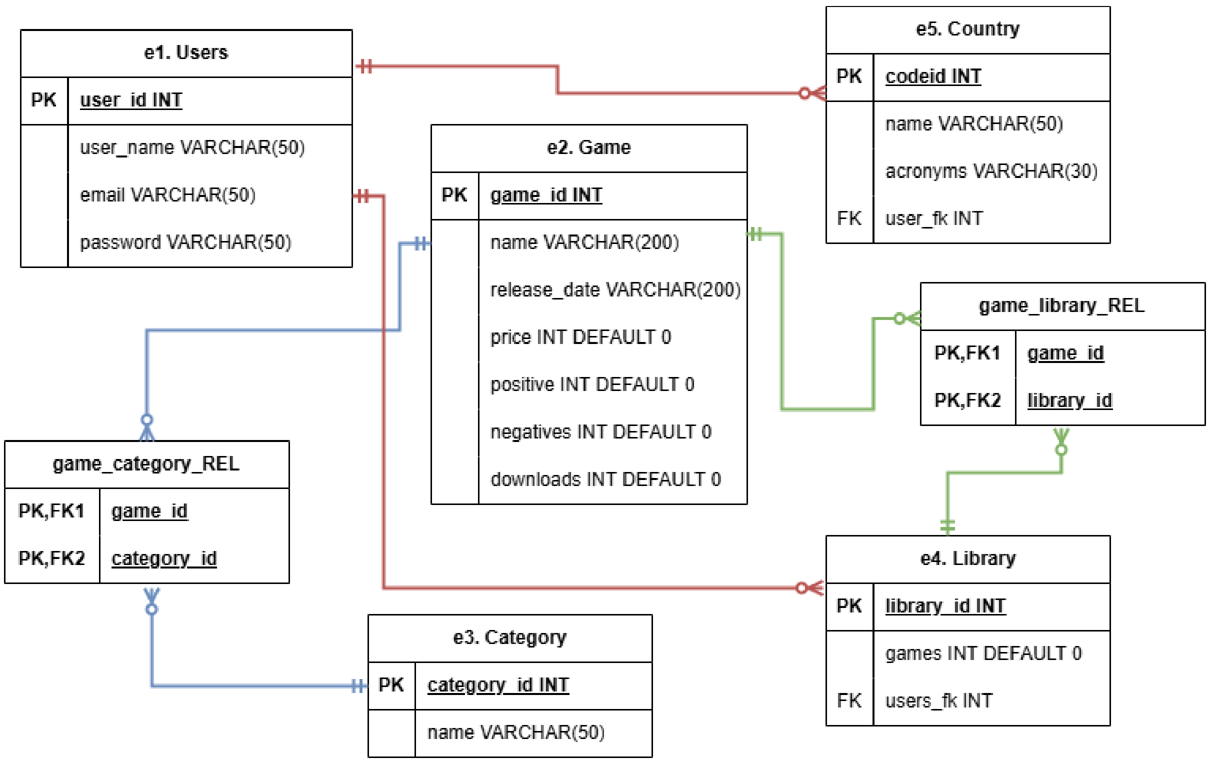
- **Step 5. First Entity-Relationship Draw:**



- **Step 6-8. First Split Many-to-Many Relationships and the Second Entity-Relationship Draw:**



- **Step 9-10. Get Data-Structure E-R M and Define Constraints and Properties of Data:**



- **Data Dictionary:**

Table	Field	Data type	Size	Primary key	Foreign key	Description	Example
Users	user_id	INT		X		Contains each user's unique type id	2233456
Users	user_name	VARCHAR	50			Contains each user's own account name	jhprimo23
Users	email	VARCHAR	50			Contain the user's email addresses	dlmunirl@mail.com
Users	password	VARCHAR	50			Contains the user's password	jhjbu..IU23
Game	game_id	INT		X	X	Contains the characteristic id of each game	4568792
Game	name	VARCHAR	200			Contains the name of the game	Call of Duty
Game	release_date	VARCHAR	200			Contains the game's release date	14/07/2012
Game	price	INT				Contains the price or cost of the game	US 58
Game	positive	INT				Contains the number of positive reviews the game has	1345
Game	negatives	INT				Contains the number of negative reviews the game has	349
Game	downloads	INT				Contains the number of downloads the game has	1432
Category	category_id	INT		X	X	Contains the specific id of each category	45
Category	name	VARCHAR	50			Contains the name of the category	"Sports"
Library	library_id	INT		X	X	Contains the specific id of each library	1234567
Library	games	INT				Contains the number of games the library has	12
Library	users_fk	INT			X	Contains the foreign key that relates the entity to the user table	Users Data
Country	codeid	INT		X		Contains the code of each country	34
Country	name	VARCHAR	50			Contains the name of each country	"Germany"
Country	acronyms	VARCHAR	30			Contains the acronyms of the name of each country	"USA"
Country	users_fk	INT			X	Contains the foreign key that relates the entity to the user table	Users Data

3. All the relational algebra queries you think will be useful in your project

- Give me a game, for example a game called "KLAUS", that has 20,0000 downloads and is free

$$\sigma_{name = \text{KLAUS} \wedge download=20000 \wedge price=0}(Game)$$

- Give me the price and number of downloads of a video game, for example the video game "WARSAW"

$$\pi_{price,\, downloads}(\sigma_{name=\text{WARSAW}}(Game))$$

- Give me the name and number of the positive and negative reviews of all game's worth less than 15 dollars

$$\pi_{name,\, positive,\, negative}\left(\sigma_{price<15}(Game)\right)$$

- I need to know all the games that are free and have more than 30,000 positive reviews

$$\pi_{name}\left(\sigma_{price=0 \wedge positive>30000}(Game)\right)$$

- I need to know which is the most expensive game in the store

$$\pi_{name}\left(\sigma_{price=\max(price)}(Game)\right)$$

- I need to know which is the cheapest paid game

$$\pi_{name}\left(\sigma_{price=\min(price) \wedge price>0}(Game)\right)$$

- Give me the 10 best games that the store has in rating (most positive reviews and least negative at the same time)

$$\pi_{name}\left(\delta_{10}\left(\sigma_{positive=\max(positive) \wedge negative=\min(negative)}(Game)\right)\right)$$

- Give me the 10 worst games the store has

$$\pi_{name}\left(\delta_{10}\left(\sigma_{negative=\max(negative)}(Game)\right)\right)$$

- I want to know which games came out in 2014 and what price they have

$$\pi_{name,\, price}(\sigma_{releasedate\, like\, '2014'}(Game))$$

- I want to know which users of the application live in Colombia

$$\pi_{username,\,country}\left(\sigma_{countryfk\,like\,'Colombia'}(Users)\right)$$

- I want to know if there are Sudan users

$$\pi_{username,\,country}\left(\sigma_{countryfk\,like\,'Sudan'}(Users)\right)$$

- I want to know how many users there are from Germany

$$\gamma_{count}\left(\sigma_{country=\text{Germany}}(Users)\right)$$

- I want to know which are the 30 categories of games with the most downloads and how many downloads each one has

$$\delta_{30}(\gamma_{categoryid,sum(downloads)\rightarrow totaldownloads}(category \bowtie game) orderbytotaldownloads\ desc)$$

- I want to know which is the category of games with the least downloads

$$\pi_{c.name}\left(\sigma_{downloads=\min(downloads)}(Category)\right)$$

- I want to know which is the category with the most games

$$\pi_{c.name}\left(\sigma_{downloads=\max(downloads)}(Category)\right)$$

- I want to see the price of the least downloaded game in the entire store

$$\pi_{price}\left(\sigma_{downloads=\min(downloads)}(Game)\right)$$

- I want to see the price of the most downloaded game in the entire store

$$\pi_{price}\left(\sigma_{downloads=\max(downloads)}(Game)\right)$$

- I want to know the email of all current users of the application

$$\pi_{email}(Users)$$

- I want to know how many game categories exist in the application

$$\gamma_{count(\cdot)}(Category)$$

- I want to know if there are games from 2023 and 2024, know what price they have, how many positive reviews they have and what the exact release date was

$$\pi_{name, price, positive, releasedata}\left(\sigma_{releasedata\ like\ '2024'\ or\ releasedata\ like\ '2024'}(Game)\right)$$

- I want to know how many games are current or how many came out in 2023

$$\gamma_{count(*)}\left(\sigma_{releasedata\ like\ '2023'}(Game)\right)$$

- I want to know if there is a game that has 0 positive reviews within the entire store

$$\pi_{name}\left(\sigma_{positive=0}(Game)\right)$$

- I want to know if there are games that have less than 30 negative reviews, more than 500 positive reviews and that were released in 2016

$$\pi_{name, positive, negative, releasedata}\left(\sigma_{negative<30 \wedge positive>500 \wedge releasedata\ like\ '2016'}(Game)\right)$$

- I want to know how many games belong to a specific category, for example the "Aliens" category

$$\gamma_{count(*)}\left(\sigma_{name = \text{Aliens}}(Category \bowtie game \bowtie gamecategory)\right)$$

- I want to know the number of games that a user has who has the letters am in their username and their library id is 1

$$\gamma_{count(*)}\left(\sigma_{username\ like\ \text{am} \wedge libraryid=1}(Library \bowtie Users)\right)$$

4. **Search for data on the web to figure out free data sources to populate any of your entities**.

Within the context of the application and the database that was sought to be replicated, it was key to have the "Game" and "Category" entities with extensive and precise information; To complete these tables, the Kaggle tool was used. **Kaggle** is an online platform that offers datasets, data science competitions, and educational resources to help data scientists collaborate, learn, and compete on data analysis projects. In it, three quite useful sets of data were found, one of them was a data with all the categories that a video game can have, the other was a data with a lot of video games and their respective characteristics and the last one corresponded to a data with a wide variety of passwords that could be assigned to each specific user. These three sets of data were entered respectively in "Category", "Game" and "Users" according to the characteristics of each entity that had been previously defined.

On the other hand, a tool that was also very useful for user data was found, this was **Mockaroo,** which is a versatile online tool designed to assist users in generating realistic test data for various purposes. It offers a user-friendly interface that allows individuals to create customized datasets tailored to their specific needs. The tool provides a wide range of data types, including names, addresses, phone numbers, email addresses, dates, and more. Users can define the format, constraints, and other characteristics of each data field, ensuring that the generated data aligns with their testing requirements. One of the key features of Mockaroo is its flexibility. Users can specify the number of rows of data they need, making it suitable for generating small sample datasets for testing individual components or large datasets for comprehensive system testing. This was used to obtain two sets of data, one with many usernames and the other with a large variety of email addresses for all users. These were entered into the "Users" entity respectively, allowing all that missing information to be completed.

The "country" entity that defines the country to which each application user belongs was completed with csv files reused from two **GitHub repositories** cited below

- 1. https://gist.github.com/brenes/1095110 /4422fd7ba3a388f31a9a017757e21e5df23c5916
- 2.https://github.com/mmonterroca/paises/blob/master/paises.csv

Additionally, the page **"datosmundial.com"** and its section of .csv files were reviewed, which also contained a wide variety of information about the names, codes and references of all countries worldwide. With this, all that information was entered, classified into three columns that we consider relevant when describing the country of origin, which are name, abbreviation and country code.

These three were the main sources of information that were used throughout the project to complete the entire database with the initial conditions that were raised. Among the strategies used to fill out the information, two were mainly used; the use of csv files and the famous "inser into" when the table required it.

The csv files previously extracted and mentioned are inserted into the tables already created within the database without any type of problem, but to give an example, there was data from the csv file that contained the countries, which had to be entered manually due to adjustments and problems import in the tool, in this case the "insert into" command was quite useful to enter the missing information in the country table.

5. Web services that will be used and what tools we going to use for it

Web services in our case are useful to the extent that they provide us with a slightly standardized interface accessible through the web to interact with our database. Allowing queries, operations and information visualizations; On the other hand, they also allow us to verify that the information to be worked on is or is not in the application's database, for example, if the user enters and it does not exist in the database, they will be asked to create a account and this will be stored in the data so that you can enter from then on. In general terms, we use it because it is a way to access and operate the database easily from the web.

To satisfy each of the user stories raised above, web services for information consultation were used. These services allowed us to access and manipulate the database to obtain the requested information. We sought to make all the user queries that we considered possible. important, so through web services we access each one of the "queries" raised.

The web services tool used was **Flask**, which is a minimalist framework for creating web applications in Python. It was developed with the intention of being lightweight and easy to extend, so the tool only gives us the essential components to work with. Flask follows the WSGI (Web Server Gateway Interface) pattern and uses Jinja2 as a templating engine and Werkzeuge as a WSGI utility library. This tool allowed the creation of a simple web application that would consult the entire database and bring the requested information about the video games and other hosted data.

The tests carried out on the web services range from testing the correct communication and insertion of information from the web application with the database, to performance tests that demonstrated that the queries were carried out correctly and that the communication between the app and the database was adequate.