A DATABASE DESIGN FOR GAMEREVIEW.COM

Charlie Ropes

December 2, 2013

TABLE OF CONTENTS

EXECUTIVE SUMMARY

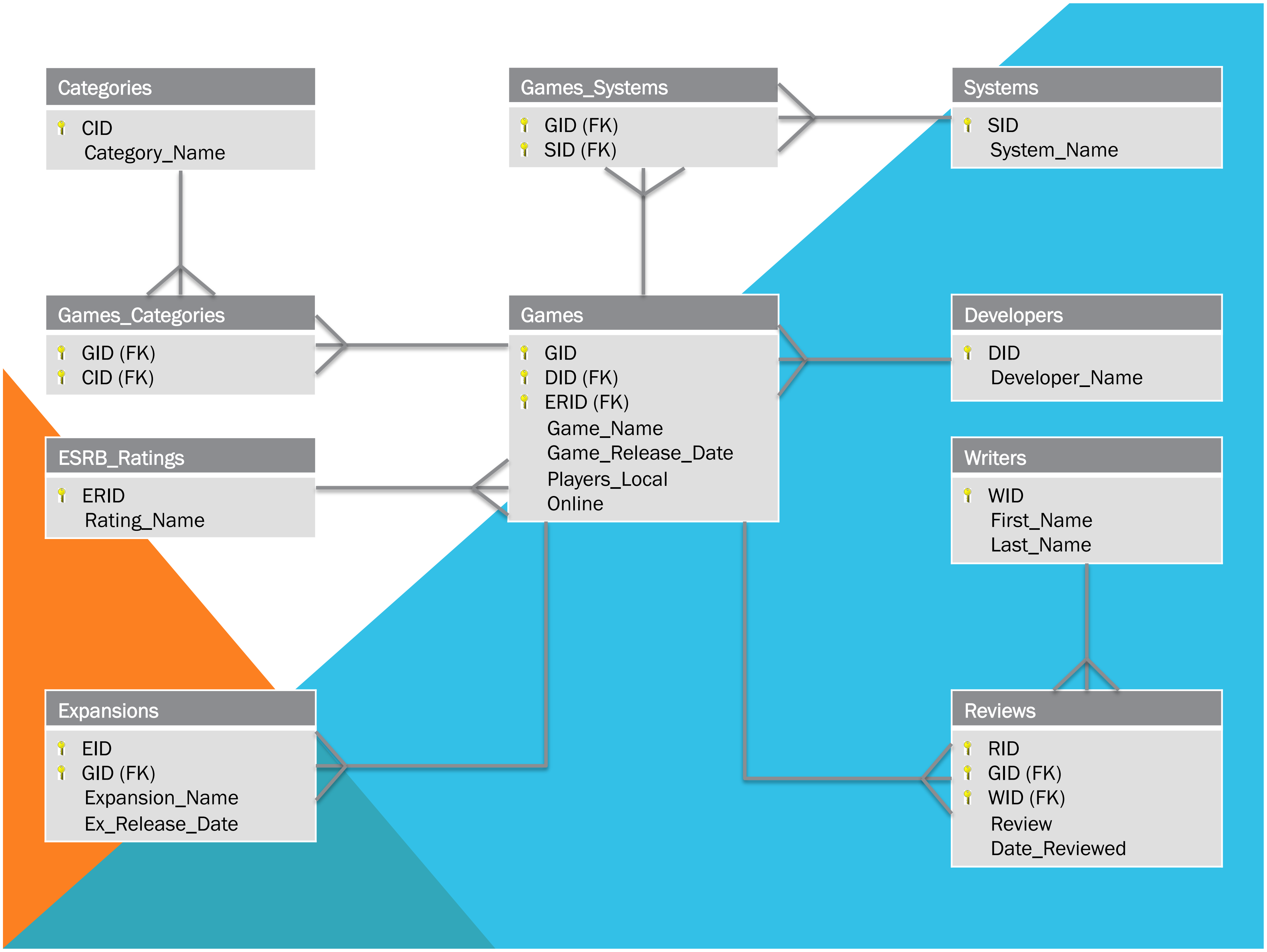
GameReview.com is a new and upcoming website in need of a database that will store information about the games they review. Before they launch and finish building their site they must have a place where they can store and organize all their data. The database must be able to separate games based off the systems they are played on and the genres that they can be classified under. GameReview.com would also like the database to keep track of game expansions and store all of their reviews.

Overview

The following document outlines all the tables that would be needed to make a database like this work. It gives all the SQL statements that would be needed to generate the database and provides sample data to illustrate how the system would store data. It also provides an ER-Diagram that will help show the relationships between different tables. It also provides sample views, reports, and stored procedures. It also explains all the designs and reasons for them.

Objective

The Objective is that the proposed database will cover all the websites needs. We aim to give them a database that is fully functional with as few bugs as possible.


Categories
CID
Category_Name
Games_Systems
GID (FK)
SID (FK)
Systems
SID
System_Name
Games_Categories
GID (FK)
CID (FK)
Games
GID
DID (FK)
ERID (FK)
Game_Name
Game_Release_Date
Players_Local
Online
Developers
DID
Developer_Name
ESRB_Ratings
ERID
Rating_Name
Writers
WID
First_Name
Last_Name
Expansions
EID
GID (FK)
Expansion_Name
Ex_Release_Date
Reviews
RID
GID (FK)
WID (FK)
Review
Date_Reviewed

CREATE TABLE STATEMENTS

This section will demonstrate and explain the creation of the 10 tables from the ER-Diagram. Each table will be explained with a short summary of what it does followed by the SQL code that creates the table in the database. Each table will also be accompanied by a list of its functional dependencies and a table of sample data to illustrate what the table will look like in the database.

<u>Tables:</u>

- Categories
- Systems
- Developers
- ESRB_Ratings
- Writers
- Games
- Games_Categories
- Games_Systems
- Expansions
- Reviews

CATEGORIES

A Categories table is necessary to list all the different categories that games can fit under. The names must be unique to insure that there is no duplicate data if someone accidentally enters a category name into the database twice.

```
CREATE TABLE Categories(
CID SERIAL NOT NULL,
Category_Name varchar (32) NOT NULL UNIQUE,
PRIMARY KEY (CID)
);
```

Functional Dependencies

CID $\rightarrow$ Category_Name

Sample Data

CID	Category_Name
1	Action
2	Adventure
3	Simulation
4	Strategy
5	Platform
6	Shooter
7	Puzzle
8	RPG

SYSTEMS

A Systems table is needed to list all the different systems that games can be played on. The system's name must be unique to insure that there is no duplicate data if someone accidentally enters a system's name into the database twice.

```
CREATE TABLE Systems(
SID SERIAL NOT NULL,
System_Name varchar (32) NOT NULL UNIQUE,
PRIMARY KEY (SID)
);
```

<u>Functional Dependencies</u>

SID $\Rightarrow$ System_Name

<u>Sample Data</u>

SID	System_Name
1	Play Station 1
2	Play Station 2
3	Play Station 3
4	Play Station 4
5	PS Vita
6	GameCube
7	Wii
8	Wii U
9	DS
10	3DS
11	Xbox
12	Xbox360
13	Xbox One

DEVELOPERS

A Developers table is needed to track the different game developers. The developer's name must be unique so that the same name is not inputted twice.

```
CREATE TABLE Developers(
DID SERIAL NOT NULL,
Developers_Name varchar (32) NOT NULL UNIQUE,
PRIMARY KEY (DID)
);
```

Functional Dependencies

$\text{DID} \rightarrow \text{Developer\_Name}$

Sample Data

DID	Developer_Name
1	UBISOFT
2	Nintendo
3	SQUARE ENIX
4	Sucker Punch
5	LUCASARTS
6	BANDI NAMCO GAMES
7	Koei
8	ACTIVISION
9	EA Sports
10	Bethesda
11	Microsoft

ESRB_RATINGS

The ESRB_Ratings table is used to store the different ratings that the ESRB can rate a game with. The Rating_Name is unique so the same rating cannot be added twice. This table should only have to have data inputted into it once unless the ESRB adds or removes a rating level.

```
CREATE TABLE ESRB_Ratings(
ERID SERIAL NOT NULL,
Rating_Name varchar (32) NOT NULL UNIQUE,
PRIMARY KEY (ERID)
);
```

Functional Dependencies

ERID $\rightarrow$ Rating_Name

Sample Data

ERID	Rating_Name
1	EC
2	E
3	E10+
4	T
5	M
6	AO

WRITERS

The Writers table will store the names of all authors that have written a review on the site. This table will only hold the writers WID number and their first and last name.

```
CREATE TABLE Writers(
WID SERIAL NOT NULL,
First_Name varchar (32) NOT NULL,
Last_Name varchar (32) NOT NULL,
PRIMARY KEY (WID)
);
```

Functional Dependencies

WID $\rightarrow$ First_Name, Last_Name

Sample Data

WID	First_Name	Last_Name
1	Charlie	Ropes
2	Mark	Vuono
3	Pat	Shea

GAMES

The Games table will store most of the information about a game. It will use foreign keys from the Developers Table and ESRB_Ratings Table for the DID of the game and the ERID of the game. It must us foreign keys because Developers can develop many games while games can only have one main developer. Also a ESRB rating can have many games while a game can only have one rating. Also game names must be unique so duplicate data does not occur. A constraint is also placed on ‘Online’ to insure that the entered data is either a ‘Yes’ or ‘No’ response.

```
CREATE TABLE Games(
GID SERIAL NOT NULL,
DID integer NOT NULL references Developers(DID),
ERID integer NOT NULL references ESRB_Ratings(ERID),
Game_Name varchar (64) NOT NULL UNIQUE,
Game_Release_Date date NOT NULL,
Players_Local integer NOT NULL,
Online varchar(8) NOT NULL
CONSTRAINT Valid_Online CHECK(Online = 'Yes' OR Online = 'No'),
PRIMARY KEY (GID)
);
```

GAMES

Functional Dependencies

GID $\rightarrow$ DID, ERID, Game_Name, Game_Release_Date, Player_Local, Online

Sample Data

GID	DID	ERID	Game_Name	Game_Release_Date	Players_Local	Online
1	4	4	inFamous	2011-0607	1	No
2	8	4	Spider-Man Shattered Dimensions	2010-09-03	1	No
3	2	2	Pokémon Pearl	2007-04-22	1	Yes
4	10	5	Skyrim	2011-11-11	1	No
5	2	3	Fire Emblem: Radiant Dawn	2007-11-11	1	No
6	11	5	Halo 4	2012-11-06	4	Yes

GAMES_CATEGORIES

The Games_Categories table is an associate table. An associate table is needed because the relationship between Games and Categories is a Many to Many relationship. A game can belong in many categories and a category can contain many games.

```
CREATE TABLE Games_Categories(
GID integer NOT NULL references Games(GID),
CID integer NOT NULL references Categories(CID),
PRIMARY KEY (GID, CID)
);
```

Functional Dependencies

None

Sample Data

GID	CID
1	1
1	2
2	1
2	2
3	1
4	1
4	2
5	4
5	8
6	1
6	6

GAMES_SYSTEMS

The Games_Systems table is an associate table. An associate table is needed because the relationship between Games and Systems is a Many to Many relationship. A game can have many systems it is designed for and a system can have many games that are designed for it.

```
CREATE TABLE Games_Systems(
GID integer NOT NULL references Games(GID),
SID integer NOT NULL references Systems(SID),
PRIMARY KEY (GID, SID)
);
```

Functional Dependencies

None

Sample Data

GID	SID
1	3
2	3
3	9
4	3
4	12
5	7
6	22

EXPANSIONS

The Expansions table will keep track of all the expansions that are released for a game. It will store a release data for the expansion and a unique name to insure duplication doesn't happen. It will also use GID as a foreign key from the Games table. It needs to reference Games because a game can have many expansions but an expansion can only be for one game.

```
CREATE TABLE Expansions(
EID SERIAL NOT NULL,
GID integer NOT NULL references Games(GID),
Ex_Name varchar (64) NOT NULL UNIQUE,
Ex_Release_Date date NOT NULL,
PRIMARY KEY (EID)
);
```

Functional Dependencies

EID $\rightarrow$ GID, Ex_Name, Ex_Release_Date

Sample Data

EID	GID	Ex_Name	Ex_Release_Date
1	4	Dragonborn	2013-02-12
2	4	Hearthfire	2013-02-19
3	4	Dawnguard	2013-02-26

REVIEWS

The Reviews table is where all the reviews will be stored. The database will store the review along with the writers WID. WID will be a foreign key from the table Writers. This needs to be done because a Writer can have multiple reviews but a review can only have one main author. Also GID will be a foreign key from Games because a game can have multiple reviews but a review can only be written about one game.

```
CREATE TABLE Reviews(
RID SERIAL NOT NULL,
GID integer NOT NULL references Games(GID),
WID integer NOT NULL references Writers(WID),
Review text NOT NULL,
Date_Reviewed date NOT NULL,
PRIMARY KEY (RID)
);
```

Functional Dependencies

RID $\rightarrow$ GID, WID, Review, Date_Reviewed

Sample Data

RID	GID	WID	Review	Date_Reviewed
1	1	1	inFamous is a great game…	2013-11-30
2	3	1	Pokémon Pearl is a wonderful…	2013-11-30

VIEWS

The following section will cover some views that have been created for the database. These views will be explained with a short summary followed by the SQL code to create that view. These views will also come with a SQL statement that will be used to query the view. A table of sample data will also be provided to illustrate what data the view will return when used.

<u>Views</u>

- Game_Info
- Newest_Game

GAME_INFO

The view ‘Game_Info’ will return all the basic information that the Database has on a game.

```
CREATE VIEW Game_Info AS
SELECT g.Game_Name, g.Game_Release_Date, g.Players_Local, g.Online,
          er.Rating_Name, d.Developer_Name, s.System_Name
FROM Games AS g, ESRB_Ratings AS er, Developers AS d, Systems AS s,
          Games_Systems AS gs
WHERE g.DID = d.DID
  AND g.ERID = er.ERID
  AND g.GID = gs.GID
  AND gs.SID = s.SID;
```

<u>SQL Statement</u>

```
SELECT * FROM Game_Info
```

<u>Sample Data</u>

Game_Name	Game_Release_Date	Players_Local	Online	Rating_Name	Developer_Name	System Name
inFamous	2011-06-07	1	No	T	Sucker Punch	Play Station 3
Spider-Man Shattered Dimensions	2010-09-03	1	No	T	ACTIVISION	Play Station 3
Pokémon Pearl	2007-04-22	1	Yes	E	Nintendo	DS
Skyrim	2011-11-11	1	No	M	Bethesda	Play Station 3
Skyrim	2011-11-11	1	No	M	Bethesda	Xbox360
Fire Emblem: Radiant Dawn	2007-11-11	1	No	E10+	Nintendo	Wii
Halo 4	2012-11-06	4	Yes	M	Microsoft	Xbox360

NEWEST_GAME

The view ‘Newest_Game’ will return a games name and release date. It will order the games by most recent release.

```
CREATE VIEW Newest_Game AS
SELECT g.Game_Name, g.Game_Release_Date
FROM Games AS g
ORDER BY g.Game_Release_Date DESC;
```

<u>SQL Statement</u>

```
SELECT * FROM Newest_Game
```

<u>Sample Data</u>

Game_Name	Game_Release_Date
Halo 4	2012-11-06
Skyrim	2011-11-11
inFamous	2011-06-07
Spider-Man Shattered Dimensions	2010-09-03
Fire Emblem: Radiant Dawn	2007-11-11
Pokémon Pearl	2007-04-22

REPORTS

The following section covers a few reports. Reports are useful because they will give the user a list of data from the database that calculates things such as the total number of games related to a gaming system.

Reports

- Total Reviews Per Writer
- Total Games Per System

TOTAL REVIEWS PER WRITER

This report will display all the writers in the database that have written a review for a game. It will also keep track of the number of reviews written by each writer and display the total number.

```
SELECT w.First_Name AS "First Name", w.Last_Name AS "Last Name", COUNT(r.Review)
          AS "Total Reviews"
FROM Writers AS w, Reviews AS r
WHERE w.WID = r.WID
GROUP BY w.First_Name, w.Last_Name
ORDER BY "Total Reviews" DESC;
```

Sample Data

First Name	Last Name	Total Reviews
Charlie	Ropes	2

TOTAL GAMES PER SYSTEM

This report will display all the systems in the database that have games related to them. It will also keep track of the number of games related to each individual system and display the total number.

```
SELECT s.System_Name AS "System", COUNT(g.Game_Name) AS "Total Games"
FROM Systems AS s, Games AS g, Games_Systems AS gs
WHERE g.GID = gs.GID
  AND gs.SID = s.SID
GROUP BY "System"
ORDER BY "Total Games" DESC;
```

Sample Data

System	Total Games
Play Station 3	3
Xbox360	2
Wii	1
DS	1

STORED PROCEDURE

The following section gives an example of a stored procedure that the database will use regularly. It will be accompanied by the SQL code to generate it, the SQL code to use it, and sample data.

<u>Stored Procedure</u>

- Category_Sort()

CATEGORY_SORT

This Stored Procedure will allow a user to look up which games are part of which category in the database. It does this by changing the inputted string in the function Category_Sort().

```
CREATE OR REPLACE FUNCTION Category_Sort(CatName text)
RETURNS TABLE (Category text, Game text) AS $$
    SELECT c.Category_Name, g.Game_Name
    FROM Categories AS c, Games AS g, Games_Categories AS gc
    WHERE g.GID = gc.GID
      AND gc.CID = c.CID
      AND c.Category_Name = CatName;
$$ LANGUAGE SQL;
```

<u>SQL Statement</u>

```
SELECT Category_Sort('Action');
```

<u>Sample Data</u>

Category_Sort
(Action ,inFamous)
(Action, "Spider-Man Shattered Dimensions”)
(Action, "Pokémon Pearl”)
(Action, Skyrim)
(Action, "Halo 4”)

SECURITY

<u>Administers</u>

Administers of the site will have all the privileges needed to update all the tables. These administers will be chosen by the owner of the site.

```
CREATE ROLE Admin
       GRANT SELECT, INSERT, UPDATE, ON ALL TABLES IN GameDatabase TO Admin
```

<u>Writers</u>

Writers will only be able to write and submit reviews. They will not be allowed to delete reviews from the database.

```
CREATE ROLE Writer
       GRANT SELECT, INSERT, ON REVIEW IN GameDatabase TO Writer
```

IMPLEMENTATION

<u>Implementation</u>

- This database will be easy to implement because the site is still being worked on. This early implementation will allow us to test the database further and correct and improve it as need be.
- The database will also come with a SQL file of the sample data which will allow the clients to experiment with the database.
- Also since the site is still being developed implementation will be easier because they can write code tailored to the database instead of having to rework what they already have.

KNOWN PROBLEMS

Current Problems

- It is not possible to write a review on an expansion for a game. If a writer wishes to do this they must write a review and use the GID of the game that the expansion is for.
- It is not possible to specify which system an expansion is created for.
- Reviews can only have one author. In the event that two people work on a review together only one of the names will be stored.
- It is not possible to specify a developer for an expansion.
- If a game has two main developers only one of the developers names can be related to the game.

FUTURE ENHANCEMENTS

Enhancements

- Allow reviews to be written for expansions.
- Create a table to store usernames and emails allowing people who are not Writers to leave comments on reviews.
- Allow reviews to be co-authored.
- Allow games to have multiple developers.
- Create a separate table to store a Review Rating of a game or expand the current Review table