Video Game Sales Database

Tae Yong Namkoong, Nate Sackett, Hyuk Joon Yang

CS 564, Summer 2020 : Professor Nguyen

1. Introduction

1.1 Motivation

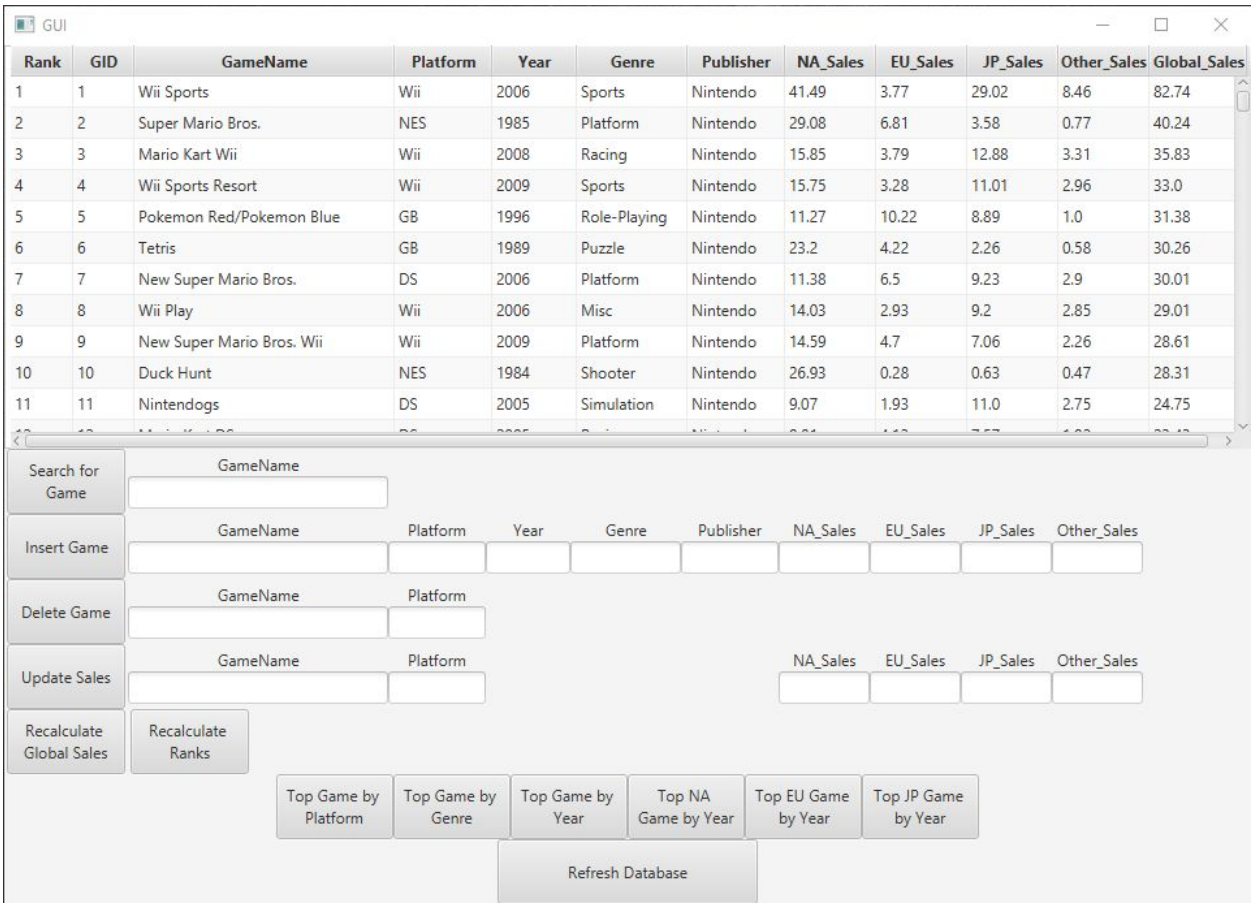
We had several motivations for choosing our domain; all our group members were all interested in gaming. Our primary motivation was to analyze and predict sales trends in the gaming industry. By applying our knowledge that we have learned in DBMS, we thought that this would provide meaningful demographic data in the gaming industry which can provide game designers with consumer demand, buying behavior, and consumption patterns in the gaming market.

There is a great need for DBMS because millions of games are sold every year and the video gaming industry is rapidly growing and expanding. In order to track the success of a particular game, we have to view it in relation/context with various categories such as platform, genre, company, regional sales, and in relation to competitors. By integrating relations between categorical information, we believe that this will provide meaningful information to users through an intuitive and easy-to-use GUI that we have implemented.

1.2 Application Description

Our application is standalone. Our back-end uses stored procedures and queries in MySQL which interfaces with our JavaFX GUI in the front-end. Our application enables users to view the games in the database, sort them by year, genre, etc., and to add, delete, or search for any game. We chose to host the database locally; we had initially planned to host the database online, but changed to make implementation easier.

The front-end is written in Java, with the GUI implemented using JavaFX and SceneBuilder, and the MySQL interface handled using JDBC. The original data is parsed from a csv file and then stored in MySQL. The data can be refreshed at any time by deleting the current tables and refilling them from the CSV file.



The application allows the user to view all 16,000+ games at once. The user can see the game ranking (in terms of overall sales), the ID of the game, its name, the platform it appears on, the year it was released, the genre, publisher, sales data in millions of units sold by region for North America, Europe, Japan, Other, and Global regions (with Global being the sum of all other regions). The user can search for any game using exact or partial match to its name, and all matching games are returned to the current table. The user is also allowed to insert a new game, the Game ID and Rank will automatically be calculated, and the table is automatically refreshed. The user can also Delete a game or update the Sales data for the game using the provided text boxes and the delete or update button. Finally, the user is also given several shortcuts to view the top game by platform, genre, or year, or the top game by year in the NA, EU, or JP regions.

1.3 Task Assignment

- Task assignment for each team member

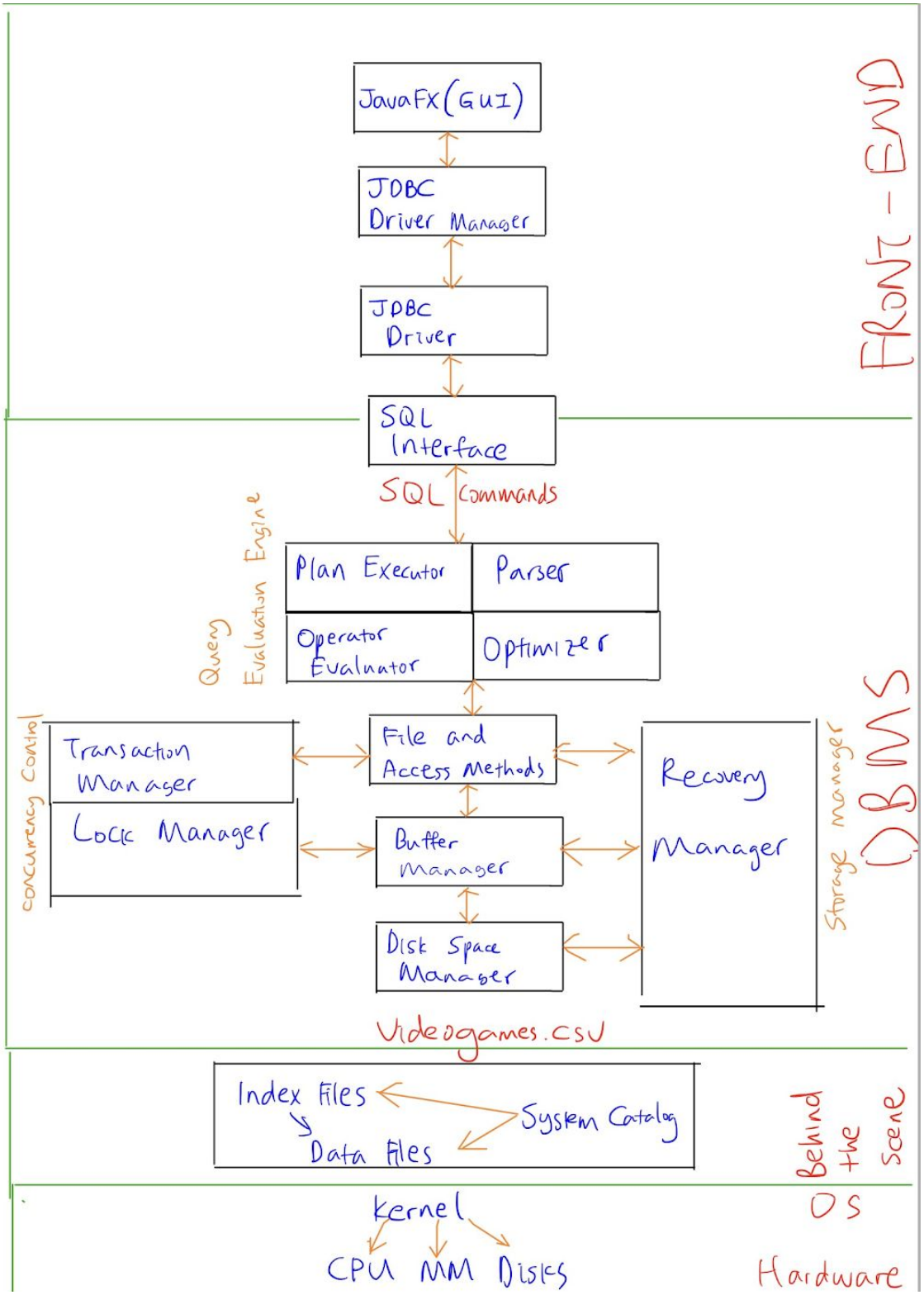
Week	Work	In charge
2	**Checkpoint 1** 1. Motivation & Project Description 2. Dataset Selection & Explanation 3. ER Diagram	 Hyukjoon Nate Tae
3	**Checkpoint 2 (06/28)** 1. Revise ER Diagram 2. Relational Schema 3. Non-trivial Functional Dependencies 4. Update Changes to DB	 Hyukjoon Nate Tae All
4	1. Revise Relational Schema 2. Normalization Process 3. Data Standardization 4. Plan Implementation	Hyukjoon Nate Tae Hyukjoon
5	**Checkpoint 3 (07/12)** 1. Refine Relational Schema 2. Normalization Process 3. Data Standardization 4. Implement Database	 Hyukjoon Nate Tae All
6	**Checkpoint 4 (07/19)** 1. SQL Queries 2. Interface 3. Stored Procedures 4. Complete Implementation 5. Evaluation 6. Write Final Report 7. Prepare Final Presentation 8. Complete Presentation Video	 All All All All All All All All
7	**Presentations**	All

1.4 Final Report Table of Contents

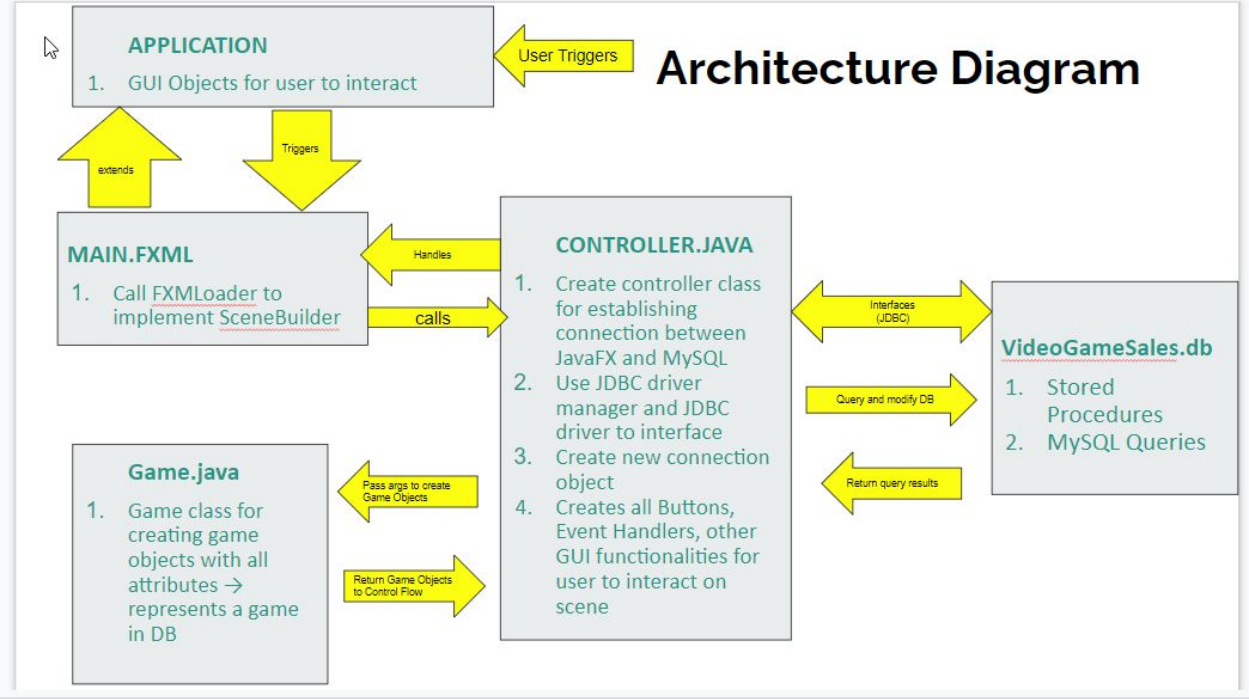
2. Our Implementation

2.1 Description of the system architecture

Our conceptual design:



Our final design:



Backend:

The backend consists of 4 MySQL tables, each describing the games, publishers, rankings, and sales data. We have stored procedures to update the global sales based on the other sales data, the rank based on the global sales, various sortings for the data, and one to view the current state of the database. We have additional stored procedures for the user to add, delete, update sales, or search for a game.

Frontend:

The frontend is written in Java; it consists of four classes and an fxml file. The Main class launches the JavaFX GUI and nothing else. Main.fxml is the xml description of the GUI, its elements, and the functions implemented when a button is clicked; it was automatically generated using SceneBuilder, a 3rd party extension for JavaFX. SceneBuilder allowed for easy development and rapid changes to the GUI layout. The Game class is the Java representation of a tuple (consisting of the joined values of the games, publishers, rankings, and sales tables) from the MySQL database. The SQLConnection class uses JDBC to establish a connection between the MySQL database and this Java project. That connection is then exported to our final class for use in displaying the game data and implementing the various functions allowed by our GUI.

The final Java class is the controller class, and this is where the bulk of the workload is handled. The Controller class initializes the fields in the GUI, the main components being the table used to display the game data, the TextFields that allow the user to input/alter the database, and the buttons used to implement the various stored procedures and queries in MySQL. The Controller class implements a function for each button in the GUI, translating the SQL query into a Statement for MySQL to execute, then handling the returned ResultSet, displaying the results in the GUI's table. This was by far the most time-intensive portion of this project, taking 40-50 hours to implement. (However, much of that time was spent learning JavaFX, SceneBuilder, JDBC, and attempting to implement various failed projects like dynamically renaming and allocating table columns.)

2.2 Description of the dataset

The dataset was obtained from Kaggle.com and is titled videogames.csv. It contains the names of video games, the platform on which the games were released, the year of publication, their genre, publisher, and the sales data for North American, European, Japanese, Other, and Global regions. Our dataset has 16,591 rows and 11 columns, which met the minimum requirement of at least 10 columns and 10,000 columns. We chose this dataset because it was relatively clean and came from a single-source, which required no multiple source integration and minimal cleaning.

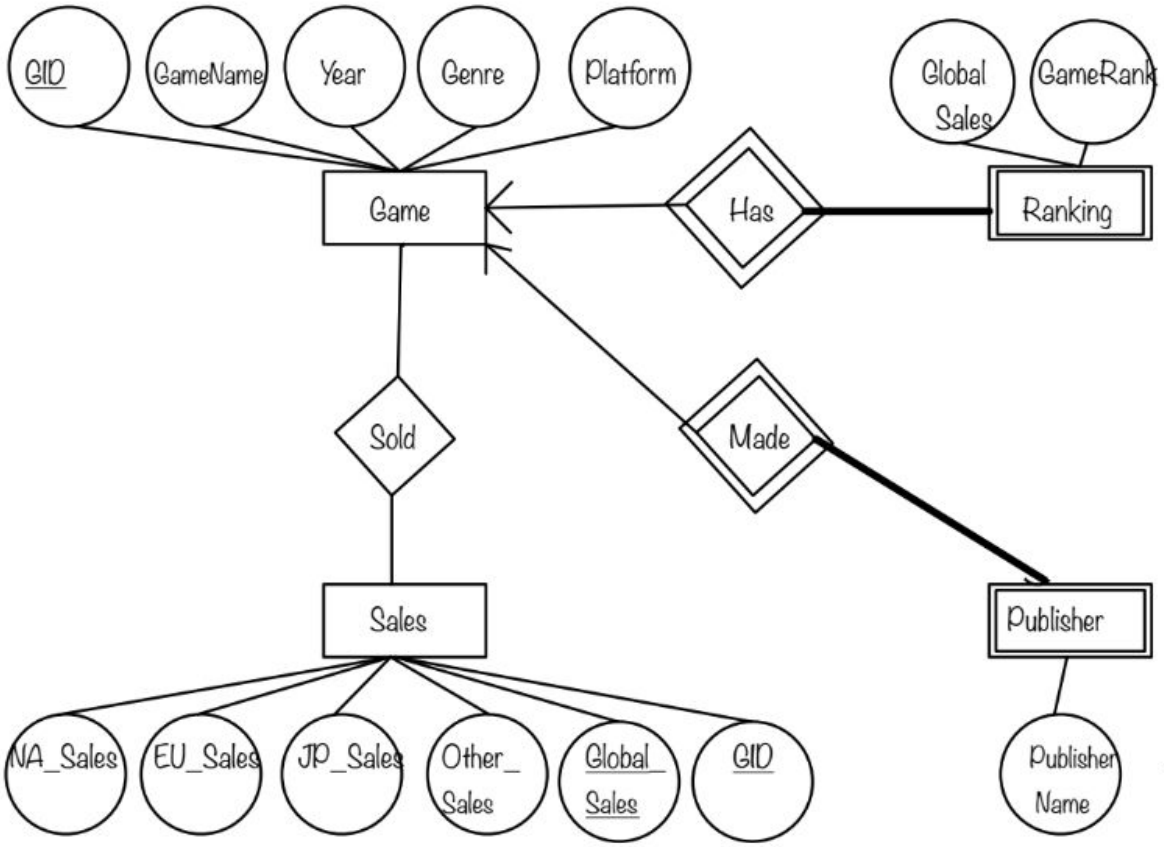
We had a few challenges handling the dataset. The global sales attribute determines the rank for each game, but several ranks were missing, so we knew that we needed to recalculate the game ranks. Next, we realized that global sales was not consistent with regional sales; the dataset had rounding errors or wrong global sales data for some games, so we knew that that needed to be recalculated based on the regional sales data. Lastly, we found that there was no way for the Games Table to be in either BCNF or 3NF without adding something to the table. GameName was a non-unique, games with the same name were released for multiple systems or in multiple years or even by multiple publishers. A candidate key for the Game table would have required a composite key consisting of the name, platform, year, and publisher, which is not in 3NF with the multiple functional dependencies derived from this. The only unique attribute was the game rank, but we knew that the game rank was going to change as the sales data updated.

<u>Solution : Normalization:</u>

To resolve these issues, we created a Game Identification Number, or GID. We had a unique quantity, the game rank, but that value was not static, so we created the GID based on the original, unique game rank. We then made every table 3NF by making all columns functionally dependent on the GID. By letting GID be the primary key, we were able to achieve 3NF and ensure a lossless and dependency

preserving decomposition. For the sales table, since global sales is the sum of all regional sales, we had to make both GID and global sales as the primary key. ER diagram (final version from the previous checkpoint copied here)

2.3 ER Diagram



2.4 Relational Model

i. Game Table

Game (GID: BIGINT, GameName: String, Platform: String, Year: String, Genre: String)

- GID - BIGINT;
- GameName - String; ≤ 135 Characters
- Platform - String; ≤ 15 Characters
- Year - String ≤ 5 characters (Some games has no data for year)
- Genre - String; (Action, Adventure, Fighting, Misc, Platform, Puzzle, Racing, Role-Playing, Shooter, Simulation, Sports, Strategy)
- Primary Key : GID

ii. Sales Table

Sales(GID: int, NA_Sales: double, EU_Sales: double, JP_Sales: double, Other_Sales: double, Global_Sales: double)

- GID – BIGINT
- NA_Sales – Positive double; 0.0 ≤ Sales ≤ 100,000
- EU_Sales – Positive double; 0.0 ≤ Sales ≤ 100,000
- JP_Sales – Positive double; 0.0 ≤ Sales ≤ 100,000
- Other_Sales – Positive double; 0.0 ≤ Sales ≤ 100,000
- Global_Sales – Positive double; 0.0 ≤ Sales ≤ 100,000 (Primary)
- Primary Key : GID, Global_Sales
- Foreign Key : (GID) references Game(GID)

iii. Publisher Table

Publisher (GID: int, PublisherName: String)

- GID – BIGINT; ≤ # of rows in table (Primary)
- PublisherName - String; ≤ 50 Characters
- Primary Key : GID
- Foreign Key : (GID) references Game(GID)

iv. Ranking Table

Ranking (GID: int, GameRank: int, Global_Sales: double)

- GID – Unique Ranking; ≤ # of rows in table (primary)
- GameRank - Unique Ranking; ≤ # of rows in table (Primary)
- Global_Sales – Positive double; 0.0 ≤ Sales ≤ 100,000
- Primary Key : GID
- Foreign Key : (GID, Global_Sales) references Sales(GID, Global_Sales)

Entity Sets Explained:

1. Game entity has attributes GID (Unique id for each game), GameName (Name of each game), Year (Year the game is released), Genre (Genre of the game), and Platform (Platform that the game is based on, such as PC, Nintendo.

2. Sales entity set contains NA_Sales (Sales in North America), EU_Sales (Sales in Europe), JP_Sales (Sales in Japan), Other_Sales (Sales in other regions), Global_Sales (Total sales), and GID (Unique id for game). We can use GID as the primary key but we need the Global_Sales from Sales in the Ranking. As we cannot define foreign key constraint, we set (GID, Global_Sales) as primary key

3. Publisher entity set is a weak entity set associated with Game entity set. It has PublisherName (Name of publisher). All entities in the Publisher entity set participate in the relationship with the Game entity by Many to One relationship constraint. Also, some games do not have

publishers so we decide to define the Publisher entity set as a weak entity set.

4. Ranking entity set contains Global_Sales (Total Sales) and GameRank (Rank of the game). All entities in the set participate in the relationship with the Game and all entities in the Ranking entity set are associated with one entity in the Game entity set. Furthermore, some ranks can have the same global sales while the rank should be constantly updated when there is a change in the global sales. Therefore, we define Ranking entity set as a weak entity set.

We used an ER approach to translate ER diagrams into a Relational model. Therefore, every weak entity set is translated into relations including key attributes of its superclass entity set and all attributes of its entity set.

Non-Trivial Functional Dependencies

1. *Non-Trivial FDs in relation Game*

   GID → GameName, Platform, Year, Genre

   GameName, Platform → GID

2. *Non-Trivial FDs in relation Sales*

   GID → EU_Sales, NA_Sales, Other_Sales, JP_Sales, Global_Sales

3. *Non-Trivial FDs in relation Ranking*

   GID → Ranking

   Note : This is initially a trivial FD but becomes a non-trivial FD as we update rank.

4. *Non-Trivial FDs in relation Publisher*

   GID → Publisher_Name

2.5 Implementation: Description of the prototype

Stored procedures and queries in DBMS

```
/* Update Global_Sales */
delimiter $$
drop procedure if exists updateGlobalSales;
create procedure updateGlobalSales()
begin
    update Sales
    set Global_Sales = round (NA_Sales + EU_Sales + JP_Sales + Other_Sales, 2);
end $$
delimiter ;
```

a. Stored procedures

i. updateGlobalSales

ii. This procedure helps the DBMS to correctly update the Global_Sales of each entity in the Sales entity set according to the sales in North America, Europe, Japan and Other regions. This procedure is called whenever there is a change in the sales of any region.

```
/* SEARCH procedure */
delimiter $$
drop procedure if exists searchGame;
create procedure searchGame (in gName varchar(135))
begin
    select r.gamerank, g2.gid, g2.GameName, g2.platform, g2.year, g2.genre, p.publishername,
        s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
    from Ranking r, sales s, publisher p, game g2,
        (select g.gid, g.GameName, g.platform, g.year, g.genre
        from Game g
        where g.GameName like concat('%',gName,'%'))g1
    where r.gid=s.gid and s.gid=p.gid and p.gid=g2.gid and g2.gid=g1.gid;
end $$
delimiter ;
```

ii.searchGame

This procedure reads user input from Java to return all data of the game the user wants to search. If a user types a certain name of a game, this procedure searches all games that have the user input in their names. If there are several entities that have the string value in their game names, it returns a list of those entities. This procedure helps the user to see the sales of the game containing the name the user wants to search.

```
/* UPDATE RANK procedure */
delimiter //
drop procedure if exists updateRank;
create procedure updateRank()
begin

    drop table if exists temp;
    CREATE TEMPORARY TABLE temp
    select * from
    (select gid,
        row_number() over (
            order by Global_Sales desc
        ) r

    from ranking
    order by global_sales desc) h;

    /* Update ranking*/
    update Ranking
        inner join temp
        on temp.gid = ranking.gid
    set ranking.gamerank= temp.r;

    drop table temp;
end //
delimiter ;
```

iii. updateRank

This procedure helps the DBMS to keep the rank of the game updated according to the changes in the global sales of the game. It creates a temporary table that copies all instances in the Sales relation and sorts them by the Global_Sales. Then the GameRank of Sales relation is updated by the row number where the GID of the instance is placed in the temporary table.

```
/* INSERT procedure*/
delimiter $$
drop procedure if exists insertGame;
create procedure insertGame(in gameName varchar (135), in platform varchar(10), in gameYear varchar (5),
                            in genre varchar(20), in publisherName varchar(40), in NA_Sales double,
                            in EU_Sales double, in JP_Sales double, in Other_Sales double)
begin

    declare gid BIGINT default (select max(GID) from Game);
    declare global_Sales double default 0.0;
    declare insertGID BIGINT;

    /* Assign unique GID for this new data*/
    set gid = gid + 1;
    /* Calculation for global_variables*/
    set global_Sales = round (NA_Sales + EU_Sales + JP_Sales + Other_Sales, 2);

    set insertGID =(
        select GID
        from Ranking r
            natural join
            (select g.gid, g.GameName, g.platform
            from Game g
            where g.GameName = gameName)g1
            natural join
            Sales s
        where g1.GameName = gameName
            and g1.Platform = Platform);

    if insertGid is null then

        /* Insert data into each table*/
        insert into Game (GID, GameName, Platform, Year, Genre)
            values (gid, gameName, platform, gameYear, genre);
        insert into Publisher (GID, PublisherName)
            values (gid, publisherName);
        insert into Sales (GID, NA_Sales, EU_Sales, JP_Sales, Other_Sales, Global_Sales)
            values (gid, NA_Sales, EU_Sales, JP_Sales, Other_Sales, global_Sales);
        insert into Ranking (GID, GameRank, Global_Sales)
            values (gid, 0, global_Sales);

        call updateRank;
    end if;
end $$
delimiter ;
```

iv. insertGame

This stored procedure inserts a new instance(Game) from user input into the database. It automatically calculates the available GID for the new instance. Then it searches if there is an existing game with the same name and platform in the database. If the game does not exist in the database, the instance is inserted into each relation with the values from the parameters. If the game already exists, the game is not inserted into the database.

v. deleteGame

This procedure is called when the user wants to delete a certain instance from the database. The procedure uses the name and the platform of a game as parameters because the attributes are super key in the relation Game. If this procedure finds the GID of a game to be deleted, it deletes the instance from the Game relation. Then all instances from other relations will be deleted by CASCADE operation. If there is no such game, no deletion of an instance is performed.

```
/* Update sales of a game */
delimiter $$
drop procedure if exists updateSales;
create procedure updateSales(in updateGameName varchar(135), in updatePlatform varchar (15),
                             in updateNA double, in updateEU double, in updateJP double, in updateOthers double)
begin
 declare updateGID BIGINT default 0;

    set updateGID =(
        select GID
        from Game
        where GameName = updateGameName
            and Platform = updatePlatform);

    if updateGID is not null then

        update Sales
        set NA_Sales = updateNA,
            EU_Sales = updateEU,
            JP_Sales = updateJP,
            Other_Sales = updateOthers
        where GID = updateGID;

        call updateGlobalSales;
        call updateRank;
    end if;

end $$
delimiter ;
```

vi. updateSales

This stored procedure helps the user to update changes in the sales of a game stored in the database. It first finds the GID of the game user wants to update sales. If there is no such game in the database, no update is performed. If the game is found, then it updates the sales in North America, Europe, Japan and Other regions by the parameters. Then it calls updateGlobalSales to update the Global_Sales of each entity set and updateRank to update

the rank of each game.

b. Queries

```
/* Display game by rank */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre,
    p.publishername, s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
order by r.gamerank;
```

```
/* DELETE procedure*/
delimiter $$
drop procedure if exists deleteGame;
create procedure deleteGame(in deleteGameName varchar (135),
                            in deletePlatform varchar(15))
begin
    declare deleteGid BIGINT default 0;

    set deleteGid =(
        select GID
        from Game
        where GameName = deleteGameName
            and Platform = deletePlatform);

    delete
    from Game
    where gid = deleteGid;

    call updateRank;

end $$
delimiter ;
```

i.Display game sorted by rank

This query displays all games stored in the database sorted by the rank. This query will be called to display all data in our interface.

ii. Display top game by each platform

```
/* Display top game by platform */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre,
    p.publishername, s.na_sales, s.eu_sales, s.jp_sales, s.other_sales,
    s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
group by g.platform
having r.gamerank <= all (select r2.gamerank
                        from Game g2, Ranking r2
                        where g.platform = g2.platform
                        and g2.GID = r2.GID
                        and g.GID <> g2.GID
                        )
order by g.platform;
```

This query is called to display the top game for each platform in the database. This helps the user see the top game by each platform

iii. Display top game by genre

```
/* Display top game by Genre */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
    s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
group by g.genre
having r.gamerank <= all (select r2.gamerank
                        from Game g2, Ranking r2
                        where g.genre = g2.genre
                        and g2.GID = r2.GID
                        and g.GID <> g2.GID
                        )
order by g.genre;
```

This query is called to display the top game by genre.

iv. Display top game per year

```
/* Display top game by Year */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
    s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
group by g.year
having r.gamerank <= all (select r2.gamerank
                          from Game g2, Ranking r2
                          where g.year = g2.year
                          and g2.GID = r2.GID
                          and g.GID <> g2.GID
                          )
order by g.year desc;
```

This query is called to return the best-selling game in each year. From this data, the user can understand the change of consumer's preferences over years.

v. Display top game in North America by year

```
/* Display top game in North America by year */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
    s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
    and s.NA_Sales > all (select s2.NA_Sales
                          from Game g2, Sales s2
                          where g.year = g2.year
                              and g2.GID = s2.GID
                              and g.GID <> g2.GID
                          )
group by g.year
order by g.year desc;
```

This query returns a table that contains the top game sold in North America by each year. The user can understand changes in the sales trend of the game market in North America from this query.

vi. Display top game in Japan by year

```
/* Display top game in Japan by year */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
    s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
    and s.JP_Sales >= all (select s2.JP_Sales
                        from Game g2, Sales s2
                        where g2.GID = s2.GID
                            and g.year = g2.year
                            and g.GID <> g2.GID
                        )
group by g.year
order by g.year desc;
```

This query returns a table that contains the best selling game in Japan each year. The user can refer to this query to understand the changes in the sales trend of the game market in Japan.

vii. Display top game in Europe by year

```
/* Display top game in Europe by year */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
    s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
    and s.EU_Sales >= all (select s2.EU_Sales
                        from Game g2, Sales s2
                        where g2.GID = s2.GID
                            and g.year = g2.year
                            and g.GID <> g2.GID
                        )
group by g.year
order by g.year desc;
```

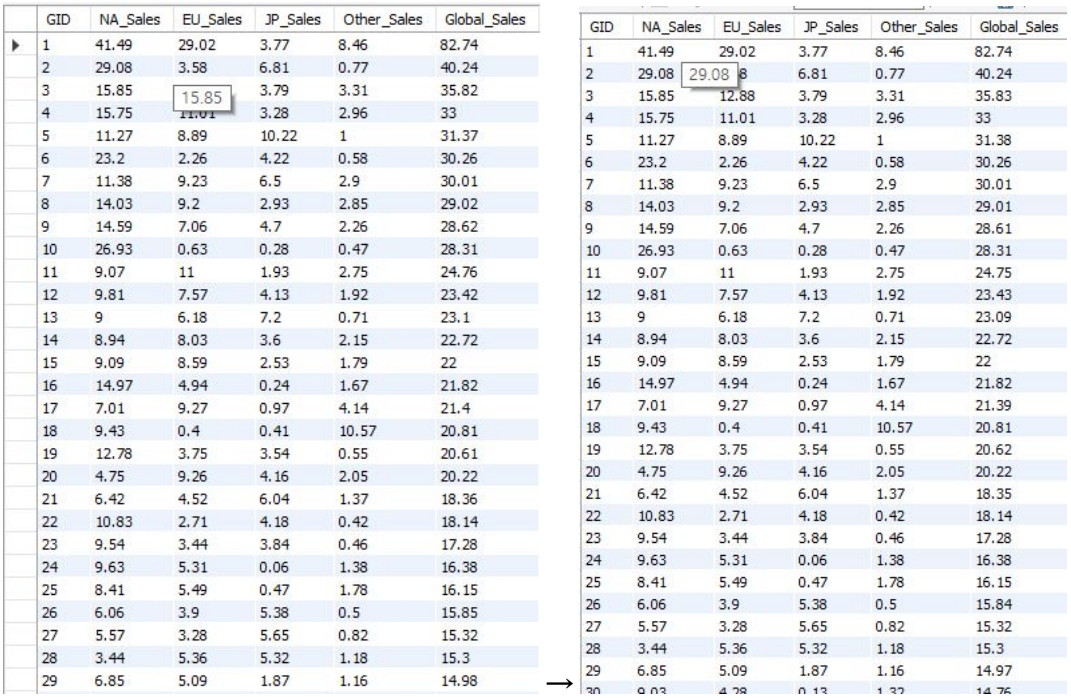
This query returns a table that contains the best selling game in Europe each year. The user can refer to this query to understand the changes in the sales trend of the game market in Europe.

2.6 Evaluation:

We evaluate our project by checking to see if our application can successfully and correctly add, remove, modify, and search games based on user input and event triggers, and checking to see if attributes such as rank and global sales for each game are updated consistently.

Below, we have tested each stored procedure that reads user input and updates the rank in our database. We test our stored procedure by comparing the expected result based on the given CSV file with the actual result returned after calling each stored procedure.

1. updateGlobal Sales



GID	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
1	41.49	29.02	3.77	8.46	82.74
2	29.08	3.58	6.81	0.77	40.24
3	15.85	15.85	3.79	3.31	35.82
4	15.75	11.01	3.28	2.96	33
5	11.27	8.89	10.22	1	31.37
6	23.2	2.26	4.22	0.58	30.26
7	11.38	9.23	6.5	2.9	30.01
8	14.03	9.2	2.93	2.85	29.02
9	14.59	7.06	4.7	2.26	28.62
10	26.93	0.63	0.28	0.47	28.31
11	9.07	11	1.93	2.75	24.76
12	9.81	7.57	4.13	1.92	23.42
13	9	6.18	7.2	0.71	23.1
14	8.94	8.03	3.6	2.15	22.72
15	9.09	8.59	2.53	1.79	22
16	14.97	4.94	0.24	1.67	21.82
17	7.01	9.27	0.97	4.14	21.4
18	9.43	0.4	0.41	10.57	20.81
19	12.78	3.75	3.54	0.55	20.61
20	4.75	9.26	4.16	2.05	20.22
21	6.42	4.52	6.04	1.37	18.36
22	10.83	2.71	4.18	0.42	18.14
23	9.54	3.44	3.84	0.46	17.28
24	9.63	5.31	0.06	1.38	16.38
25	8.41	5.49	0.47	1.78	16.15
26	6.06	3.9	5.38	0.5	15.85
27	5.57	3.28	5.65	0.82	15.32
28	3.44	5.36	5.32	1.18	15.3
29	6.85	5.09	1.87	1.16	14.98

→

GID	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
1	41.49	29.02	3.77	8.46	82.74
2	29.08	29.08	6.81	0.77	40.24
3	15.85	12.88	3.79	3.31	35.83
4	15.75	11.01	3.28	2.96	33
5	11.27	8.89	10.22	1	31.38
6	23.2	2.26	4.22	0.58	30.26
7	11.38	9.23	6.5	2.9	30.01
8	14.03	9.2	2.93	2.85	29.01
9	14.59	7.06	4.7	2.26	28.61
10	26.93	0.63	0.28	0.47	28.31
11	9.07	11	1.93	2.75	24.75
12	9.81	7.57	4.13	1.92	23.43
13	9	6.18	7.2	0.71	23.09
14	8.94	8.03	3.6	2.15	22.72
15	9.09	8.59	2.53	1.79	22
16	14.97	4.94	0.24	1.67	21.82
17	7.01	9.27	0.97	4.14	21.39
18	9.43	0.4	0.41	10.57	20.81
19	12.78	3.75	3.54	0.55	20.62
20	4.75	9.26	4.16	2.05	20.22
21	6.42	4.52	6.04	1.37	18.35
22	10.83	2.71	4.18	0.42	18.14
23	9.54	3.44	3.84	0.46	17.28
24	9.63	5.31	0.06	1.38	16.38
25	8.41	5.49	0.47	1.78	16.15
26	6.06	3.9	5.38	0.5	15.84
27	5.57	3.28	5.65	0.82	15.32
28	3.44	5.36	5.32	1.18	15.3
29	6.85	5.09	1.87	1.16	14.97
30	9.03	4.28	0.13	1.32	14.76

Left table displays the initial values of Global_sales for each instance in the dataset. In the initial table, we found there are miscalculations of global_sales in many games. For example, GID: 3, 5, 8, 9, 11, 12, 13, 14 had different results from our calculations. So we implemented this procedure to check if the procedure correctly updates the values of Global_sales for each instance by comparing with our calculations. In fact, the procedure updates all the values exactly the same as our calculations. Since the calculation method in the procedure is the same as our calculation method while the tests are successful, we can evaluate this procedure to be successful.

2. updateRank()

GID	GameRank	GameName
1525	1525	Digimon World
1526	1526	Rise of the Tomb Raider
1527	1527	Need for Speed Carbon
1528	1528	Yoshi's Woolly World
1529	1529	Mega Man Battle Network 3...
1530	1530	FIFA Street
1531	1531	Resident Evil Zero
1533	1533	Tom Clancy's Rainbow Six: ...
1534	1534	Pirates of the Caribbean: T...
1535	1535	Transformers
1536	1536	Guitar Hero: Aerosmith
1537	1537	Mystery Case Files: MillionHeir
1538	1538	Valkyria Chronicles
1539	1539	Midnight Club 3: DUB Editio...
1540	1540	Champions of Norrath
1541	1541	Watch Dogs
1542	1542	Imagine: Wedding Designer
1543	1543	Metal Gear Solid V: Ground ...
1544	1544	Yakuman
1545	1545	Assassin's Creed Syndicate
1546	1546	Army Men 3D
1547	1547	WWE '13

→

GID	GameRank	GameName
1525	1524	Digimon World
1526	1525	Rise of the Tomb Raider
1527	1526	Need for Speed Carbon
1528	1527	Yoshi's Woolly World
1529	1528	Mega Man Battle Network 3...
1530	1529	FIFA Street
1531	1530	Resident Evil Zero
1533	1531	Tom Clancy's Rainbow Six: ...
1534	1532	Pirates of the Caribbean: T...
1535	1533	Transformers
1536	1534	Guitar Hero: Aerosmith
1537	1535	Mystery Case Files: MillionHeir
1538	1536	Valkyria Chronicles
1539	1537	Midnight Club 3: DUB Editio...
1540	1538	Champions of Norrath
1541	1539	Watch Dogs
1542	1540	Imagine: Wedding Designer
1543	1541	Metal Gear Solid V: Ground ...
1544	1542	Yakuman
1545	1543	Assassin's Creed Syndicate
1546	1544	Army Men 3D
1547	1545	WWE '13
1548	1546	Twisted Metal 4
1549	1547	Manhunt
1550	1548	Kid Icarus: Uprising
1551	1549	Silent Hill 2
1552	1550	Tomb Raider: Underworld
1553	1551	Castlevania: Symphony of ...

Left table is the initial table when the dataset is first imported into the database. We can find the game rank is not consistent because the next rank of 1531 is 1533. 1532 is missing in the rank. Furthermore, we imported 16591 instances so we expect the last rank to be 16591 but the last rank was bigger than our expected rank. After implementation of the stored procedure, we confirmed that the rank was successfully updated because the rank became consistent while the last rank became 16591 as we expected.

3. SearchGame()

```
call searchGame('call of duty: modern warfare 3');
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
30	30	Call of Duty: Modern Warfare 3	X360	2011	Shooter	Activision	9.03	4.28	0.13	1.32	14.76
38	38	Call of Duty: Modern Warfare 3	PS3	2011	Shooter	Activision	5.54	5.82	0.49	1.62	13.46
1038	1039	Call of Duty: Modern Warfare 3	PC	2011	Shooter	Activision	0.41	0.98	0	0.33	1.72
2350	2352	Call of Duty: Modern Warfare 3	Wii	2011	Shooter	Activision	0.6	0.21	0	0.08	0.89
5916	5920	Call of Duty: Modern Warfare 3: Defiance	DS	2011	Shooter	Activision	0.21	0.07	0	0.02	0.3

```
call searchGame('call of duty:');
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
30	30	Call of Duty: Modern Warfare 3	X360	2011	Shooter	Activision	9.03	4.28	0.13	1.32	14.76
32	32	Call of Duty: Black Ops	X360	2010	Shooter	Activision	9.67	3.73	0.11	1.13	14.64
34	34	Call of Duty: Black Ops 3	PS4	2015	Shooter	Activision	5.77	5.81	0.35	2.31	14.24
35	35	Call of Duty: Black Ops II	PS3	2012	Shooter	Activision	4.99	5.88	0.65	2.52	14.03
36	36	Call of Duty: Black Ops II	X360	2012	Shooter	Activision	8.25	4.3	0.07	1.12	13.73
37	37	Call of Duty: Modern Warfare 2	X360	2009	Shooter	Activision	8.52	3.63	0.08	1.29	13.51
38	38	Call of Duty: Modern Warfare 3	PS3	2011	Shooter	Activision	5.54	5.82	0.49	1.62	13.46
41	41	Call of Duty: Black Ops	PS3	2010	Shooter	Activision	5.98	4.44	0.48	1.83	12.73
56	56	Call of Duty: Modern Warfare 2	PS3	2009	Shooter	Activision	4.99	3.69	0.38	1.63	10.69

```
call searchGame('no game exist');
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales

First scenario is to search for a game that has only one instance that has the same game name. We found there are 5 different platforms for ‘Call of Duty: Modern Warfare 3’ in the dataset so we tested the procedure if it returns 5 instances. As shown in the top most table, the procedure returns the instances we expected.

Second scenario is to search for a game that has the string in its name. We found there are 48 games containing ’Call of Duty:’ in their names so we tested the procedure if it returns 48 instances when we call the procedure with ‘Call of Duty’ as input. The procedure returns all 48 instances that we expected to be returned.

Last scenario is to search for a game that does not exist. We expect no instance to be returned from the procedure, and the procedure returned no instance.

Since 3 different scenarios confirmed the procedure returns expected instances, the procedures are tested to be correct.

4. insertGame()

GameRank	GID	GameName	Platform	PublisherName	Year	Genre	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
1	1	Wii Sports	Wii	Nintendo	2006	Sports	82.74	41.49	29.02	3.77	8.46
2	2	Super Mario Bros.	NES	Nintendo	1985	Platform	40.24	29.08	3.58	6.81	0.77
3	3	Mario Kart Wii	Wii	Nintendo	2008	Racing	35.83	15.85	12.88	3.79	3.31
4	4	Wii Sports Resort	Wii	Nintendo	2009	Sports	33	15.75	11.01	3.28	2.96

```
call insertGame('Test', 'Test', '2020', 'Action', 'CS', 50, 50, 50, 50);
```

GameRank	GID	GameName	Platform	PublisherName	Year	Genre	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
1	16601	Test	Test	CS	2020	Action	200	50	50	50	50
2	1	Wii Sports	Wii	Nintendo	2006	Sports	82.74	41.49	29.02	3.77	8.46
3	2	Super Mario Bros.	NES	Nintendo	1985	Platform	40.24	29.08	3.58	6.81	0.77
4	3	Mario Kart Wii	Wii	Nintendo	2008	Racing	35.83	15.85	12.88	3.79	3.31

```
call insertGame('Test1', 'Test1', '2020', 'Sports', '564', 20, 23.02, 21.01, 0.07);
```

GameRank	GID	GameName	Platform	PublisherName	Year	Genre	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
1	16601	Test	Test	CS	2020	Action	200	50	50	50	50
2	1	Wii Sports	Wii	Nintendo	2006	Sports	82.74	41.49	29.02	3.77	8.46
3	16602	Test1	Test1	564	2020	Sports	64.1	20	23.02	21.01	0.07
4	2	Super Mario Bros.	NES	Nintendo	1985	Platform	40.24	29.08	3.58	6.81	0.77

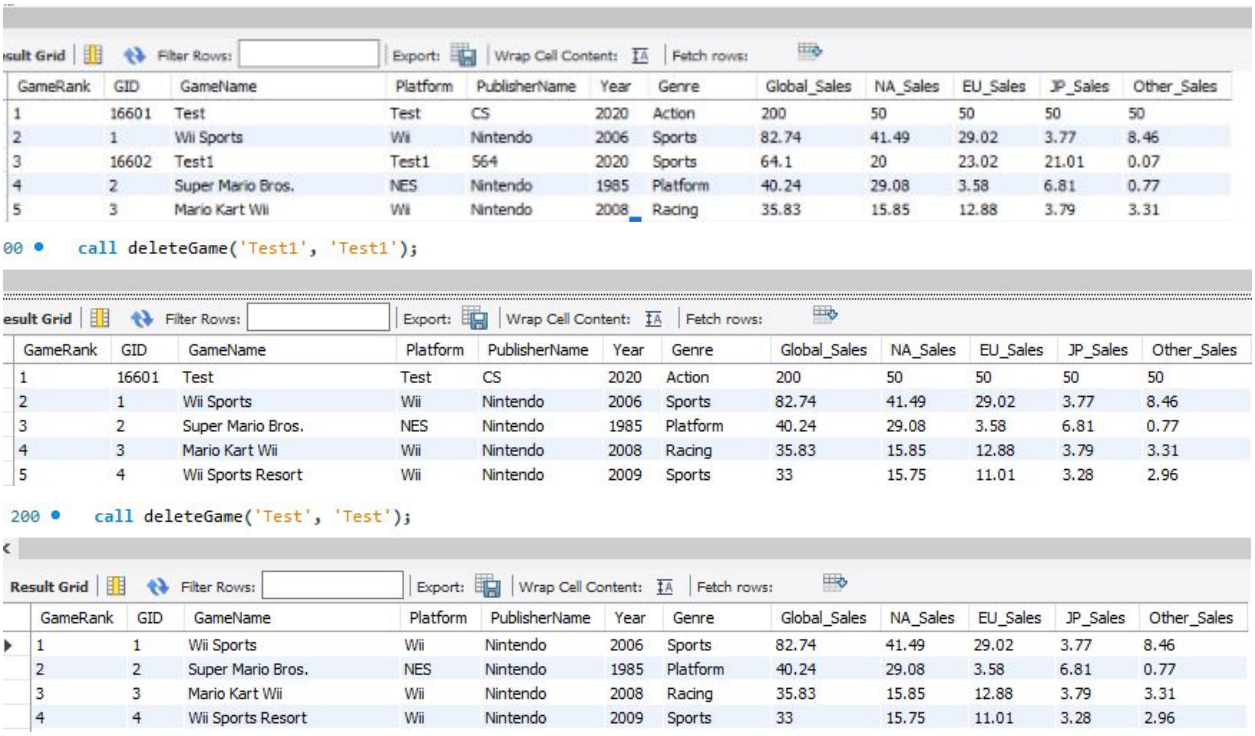
```
call insertGame('Test1', 'Test1', '2019', 'Racing', '564', 60, 100, 21.01, 0.07); /* Existing game cannot be inserted */
```

GameRank	GID	GameName	Platform	PublisherName	Year	Genre	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
1	16601	Test	Test	CS	2020	Action	200	50	50	50	50
2	1	Wii Sports	Wii	Nintendo	2006	Sports	82.74	41.49	29.02	3.77	8.46
3	16602	Test1	Test1	564	2020	Sports	64.1	20	23.02	21.01	0.07
4	2	Super Mario Bros.	NES	Nintendo	1985	Platform	40.24	29.08	3.58	6.81	0.77
5	3	Mario Kart Wii	Wii	Nintendo	2008	Racing	35.83	15.85	12.88	3.79	3.31

The first table is the initial data. We first tested if the procedure inserts correct values of a game with the updates of rank.We expected the game ‘Test’ to be ranked as first place by giving largest values for all sales in regions. After implementing the procedure, we found the game is ranked as first with the correct value for Global_Sales.

Then we tested to insert a new game ‘Test1’ that should be ranked as 3rd. This scenario is to test if the procedure can insert a new game that is in the middle of ranks in the database. The procedure is confirmed to insert the game ‘Test1’ successfully.
Last scenario is to test inserting a game that already exists in the database. We expected there should be no change in the database. As the last table shows, there was no change in the database.
Since the procedure passes 3 different scenarios, the test performs successfully.

5. removeGame()



GameRank	GID	GameName	Platform	PublisherName	Year	Genre	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
1	16601	Test	Test	CS	2020	Action	200	50	50	50	50
2	1	Wii Sports	Wii	Nintendo	2006	Sports	82.74	41.49	29.02	3.77	8.46
3	16602	Test1	Test1	564	2020	Sports	64.1	20	23.02	21.01	0.07
4	2	Super Mario Bros.	NES	Nintendo	1985	Platform	40.24	29.08	3.58	6.81	0.77
5	3	Mario Kart Wii	Wii	Nintendo	2008	Racing	35.83	15.85	12.88	3.79	3.31

```
00 •  call deleteGame('Test1', 'Test1');
```

GameRank	GID	GameName	Platform	PublisherName	Year	Genre	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
1	16601	Test	Test	CS	2020	Action	200	50	50	50	50
2	1	Wii Sports	Wii	Nintendo	2006	Sports	82.74	41.49	29.02	3.77	8.46
3	2	Super Mario Bros.	NES	Nintendo	1985	Platform	40.24	29.08	3.58	6.81	0.77
4	3	Mario Kart Wii	Wii	Nintendo	2008	Racing	35.83	15.85	12.88	3.79	3.31
5	4	Wii Sports Resort	Wii	Nintendo	2009	Sports	33	15.75	11.01	3.28	2.96

```
200 •  call deleteGame('Test', 'Test');
```

GameRank	GID	GameName	Platform	PublisherName	Year	Genre	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
1	1	Wii Sports	Wii	Nintendo	2006	Sports	82.74	41.49	29.02	3.77	8.46
2	2	Super Mario Bros.	NES	Nintendo	1985	Platform	40.24	29.08	3.58	6.81	0.77
3	3	Mario Kart Wii	Wii	Nintendo	2008	Racing	35.83	15.85	12.88	3.79	3.31
4	4	Wii Sports Resort	Wii	Nintendo	2009	Sports	33	15.75	11.01	3.28	2.96

The first table is the initial table before removeGame is implemented. We tested if the rank of a given game is correctly updated after deleting a game, and if the game is successfully deleted. Deleting a game in the middle of rank and deleting a game in top rank returns expected instances. Then we test deleting a game that does not exist in the database and the procedure returns all instances with no change, which we expected to be returned. Therefore, the procedure performs correctly.

Evaluation of queries

For queries, we tested queries by checking if the instances of the result table are correctly provided by the purpose of the queries.

1. Display Game by Rank

```
/* Display game by rank */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre,
    p.publishername, s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
order by r.gamerank;
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
1	1	Wii Sports	Wii	2006	Sports	Wii Sports	41.49	29.02	3.77	8.46	82.74
2	2	Super Mario Bros.	NES	1985	Platform	Super Mario Bros.	29.08	3.58	6.81	0.77	40.24
3	3	Mario Kart Wii	Wii	2008	Racing	Mario Kart Wii	15.85	12.88	3.79	3.31	35.82
4	4	Wii Sports Resort	Wii	2009	Sports	Wii Sports Resort	15.75	11.01	3.28	2.96	33
5	5	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Pokemon Red/Pokemon Blue	11.27	8.89	10.22	1	31.37

This query displays all games sorted in descending order of their rank. We tested by sorting the games by their global_sales since a game with the highest global_sales should also be ranked at the top of our list and then checking if the games ranked higher do have higher global_sales than games ranked below the list. The image below shows the output when the SQL query is run. Clearly, the output matches our expected output since the games with higher global_sales are ranked higher than the games with lower global_sales for all games.

2. Display Top Game By Platform

```
/* Display top game by platform */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre,
    p.publishername, s.na_sales, s.eu_sales, s.jp_sales, s.other_sales,
    s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
group by g.platform
having r.gamerank <= all (select r2.gamerank
                          from Game g2, Ranking r2
                          where g.platform = g2.platform
                          and g2.GID = r2.GID
                          and g.GID <> g2.GID
                          )
order by g.platform;
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
90	90	Pac-Man	2600	1982	Puzzle	Pac-Man	7.28	0.45	0	0.08	7.81
12637	12637	Policenauts	3DO	1995	Adventure	Policenauts	0	0	0.06	0	0.06
33	33	Pokemon X/Pokemon Y	3DS	2013	Role-Playing	Pokemon X/Pokemon Y	5.17	4.05	4.34	0.79	14.35
638	638	Sonic Adventure	DC	1998	Platform	Sonic Adventure	1.26	0.61	0.46	0.08	2.42
7	7	New Super Mario Bros.	DS	2006	Platform	New Super Mario Bros.	11.38	9.23	6.5	2.9	30.01

We tested this query by first grouping the games by their platform. Then for each distinct platform, we use a subquery to find the top rank for that platform. The image below shows the output of our SQL query. If our query runs correctly, an expected output will display distinct platform names and the top game for each unique platform. Clearly, the platform names are all unique and each platform has the highest global_sales, since we only display the top game for each platform. Therefore, the output of our query corresponds to our expected output.

3. Display Top Game By Genre

```
238     /* Display top game by Genre */
239     select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
240         s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
241     from Ranking r, Game g, Publisher p, Sales s
242     where g.GID = r.GID
243         and g.GID = p.GID
244         and g.GID = s.GID
245     group by g.genre
246     having r.gamerank <= all (select r2.gamerank
247                               from Game g2, Ranking r2
248                               where g.genre = g2.genre
249                               and g2.GID = r2.GID
250                               and g.GID <> g2.GID
251                               )
252     order by g.genre;
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
17	17	Grand Theft Auto V	PS3	2013	Action	Grand Theft Auto V	7.01	9.27	0.97	4.14	21.4
51	51	Super Mario Land 2: 6 Golden Coins	GB	1992	Adventure	Super Mario Land 2: 6 Golden Coins	6.16	2.04	2.69	0.29	11.18
40	40	Super Smash Bros. Brawl	Wii	2008	Fighting	Super Smash Bros. Brawl	6.75	2.61	2.66	1.02	13.04
8	8	Wii Play	Wii	2006	Misc	Wii Play	14.03	9.2	2.93	2.85	29.02
2	2	Super Mario Bros.	NES	1985	Platform	Super Mario Bros.	29.08	3.58	6.81	0.77	40.24

We tested this query by first grouping the games by their genre. Then for each distinct genre, we use a subquery to find the top rank for that genre. The image below shows the output of our SQL query. If our query runs correctly, an expected output will display distinct genres and the top game for each unique genre. Clearly, the genres are all unique and each genre has the highest global_sales, since we only display the top game for each genre. Therefore, the output of our query corresponds to our expected output.

4. Display Top Game By Year

```
254     /* Display top game by Year */
255     select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
256         s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
257     from Ranking r, Game g, Publisher p, Sales s
258     where g.GID = r.GID
259         and g.GID = p.GID
260         and g.GID = s.GID
261     group by g.year
262     having r.gamerank <= all (select r2.gamerank
263                               from Game g2, Ranking r2
264                               where g.year = g2.year
265                               and g2.GID = r2.GID
266                               and g.GID <> g2.GID
267                               )
268     order by g.year desc;
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
180	180	Madden NFL 2004	PS2	N/A	Sports	Madden NFL 2004	4.26	0.26	0.01	0.71	5.23
5959	5959	Imagine: Makeup Artist	DS	2020	Simulation	Imagine: Makeup Artist	0.27	0	0	0.02	0.29
222	222	FIFA 17	PS4	2016	Sports	FIFA 17	0.28	3.75	0.06	0.69	4.77
34	34	Call of Duty: Black Ops 3	PS4	2015	Shooter	Call of Duty: Black Ops 3	5.77	5.81	0.35	2.31	14.24
45	45	Grand Theft Auto V	PS4	2014	Action	Grand Theft Auto V	3.8	5.81	0.36	2.02	11.98

We tested this query by first grouping the games by their year. Then for each distinct year, we use a subquery to find the top rank for that year. The image below shows the output of our SQL query. If our query runs correctly, an expected output will display distinct years and the top game for each unique year. Clearly, the year values are all unique and each year has the highest global_sales, since we only display the top game for each year. Therefore, the output of our query corresponds to our expected output.

5. Display top game in North America by year

```
/* Display top game in North America by year */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
    s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
    and s.NA_Sales > all (select s2.NA_Sales
                          from Game g2, Sales s2
                          where g.year = g2.year
                              and g2.GID = s2.GID
                              and g.GID <> g2.GID
                          )
group by g.year
order by g.year desc;
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
180	180	Madden NFL 2004	PS2	N/A	Sports	Madden NFL 2004	4.26	0.26	0.01	0.71	5.23
5959	5959	Imagine: Makeup Artist	DS	2020	Simulation	Imagine: Makeup Artist	0.27	0	0	0.02	0.29
272	272	Uncharted 4: A Thief's End	PS4	2016	Shooter	Uncharted 4: A Thief's End	1.3	2.07	0.18	0.65	4.2
34	34	Call of Duty: Black Ops 3	PS4	2015	Shooter	Call of Duty: Black Ops 3	5.77	5.81	0.35	2.31	14.24
24	24	Grand Theft Auto V	X360	2013	Action	Grand Theft Auto V	9.63	5.31	0.06	1.38	16.38

We tested this query by first sorting the game by the highest NA_Sales which corresponds to the subquery. Then, we grouped the sorted games by each distinct year and ordered it in descending order of NA_Sales. The image below shows the output of our SQL query. If our query runs correctly, an expected output will display tuples with games that have the highest NA_Sales for each year. One way we tested this was to see if the returned year values were unique and then checking which game in each year actually had the highest NA_Sales. Clearly, the year values are all unique, and each year had the corresponding highest NA_Sales for that year, which matched the expected output.

6. Display top game in Europe by year

```
/* Display top game in Europe by year */
select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
    s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
from Ranking r, Game g, Publisher p, Sales s
where g.GID = r.GID
    and g.GID = p.GID
    and g.GID = s.GID
    and s.EU_Sales >= all (select s2.EU_Sales
                           from Game g2, Sales s2
                           where g2.GID = s2.GID
                               and g.year = g2.year
                               and g.GID <> g2.GID
                           )
group by g.year
order by g.year desc;
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
378	378	FIFA Soccer 2004	PS2	N/A	Sports	FIFA Soccer 2004	0.59	2.36	0.04	0.51	3.49
5959	5959	Imagine: Makeup Artist	DS	2020	Simulation	Imagine: Makeup Artist	0.27	0	0	0.02	0.29
16441	16441	Brothers Conflict: Precious Baby	PSV	2017	Action	Brothers Conflict: Precious Baby	0	0	0.01	0	0.01
222	222	FIFA 17	PS4	2016	Sports	FIFA 17	0.28	3.75	0.06	0.69	4.77
78	78	FIFA 16	PS4	2015	Sports	FIFA 16	1.11	6.06	0.06	1.26	8.49

Result 15

We tested this query by first sorting the game by the highest EU_Sales which corresponds to the subquery. Then, we grouped the sorted games by each distinct year and ordered it in descending order of EU_Sales. The image below shows the output of our SQL query. If our query runs correctly, an expected output will display tuples with games that have the highest EU_Sales for each year. One way we tested this was to see if the returned year values were unique and then checking which game in each year actually had the highest EU_Sales. Clearly, the year values are all unique, and each year had the corresponding highest EU_Sales for that year, which matched the expected output.

7. Display Top Game in Japan

```
284     /* Display top game in Japan by year */
285     select r.gamerank, g.gid, g.GameName, g.platform, g.year, g.genre, p.publishername,
286         s.na_sales, s.eu_sales, s.jp_sales, s.other_sales, s.global_sales
287     from Ranking r, Game g, Publisher p, Sales s
288     where g.GID = r.GID
289         and g.GID = p.GID
290         and g.GID = s.GID
291         and s.JP_Sales >= all (select s2.JP_Sales
292                               from Game g2, Sales s2
293                               where g2.GID = s2.GID
294                                   and g.year = g2.year
295                                   and g.GID <> g2.GID
296                               )
297     group by g.year
298     order by g.year desc;
```

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
2297	2297	Rhythm Heaven	Wii	N/A	Misc	Rhythm Heaven	0.13	0	0.77	0.01	0.9
5959	5959	Imagine: Makeup Artist	DS	2020	Simulation	Imagine: Makeup Artist	0.27	0	0	0.02	0.29
1570	1570	Yokai Watch 3	3DS	2016	Action	Yokai Watch 3	0	0	1.27	0	1.27
415	415	Monster Hunter X	3DS	2015	Action	Monster Hunter X	0.25	0.19	2.78	0.04	3.26
420	420	Yokai Watch 2 Ganso/Honke	3DS	2014	Role-Playing	Yokai Watch 2 Ganso/Honke	0.03	0	3.18	0	3.22

We tested this query by first sorting the game by the highest JP_Sales which corresponds to the subquery. Then, we grouped the sorted games by each distinct year and ordered it in descending order of JP_Sales. The image below shows the output of our SQL query. If our query runs correctly, an expected output will display tuples with games that have the highest JP_Sales for each year. One way we tested this was to see if the returned year values were unique and then checking which game in each year actually had the highest JP_Sales. Clearly, the year values are all unique, and each year had the corresponding highest JP_Sales for that year, which matched the expected output.

<u>Final evaluation of the application</u>

- Objective
    - Make sure that games can be added, removed, searched for, global sales correctly. In each test case, we check if running each query increments regional sales, and updates rank consistently after changing sales data.
- Test case 1: Test that Global sales increases when the sales data in any region is incremented
gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
4	4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33

      We decided to use ‘Wii Sports Resort’ to test if the application correctly updates

changes made by user in the database. This is the initial data of the game before we change the value of sales in NA, EU, JP and other regions.

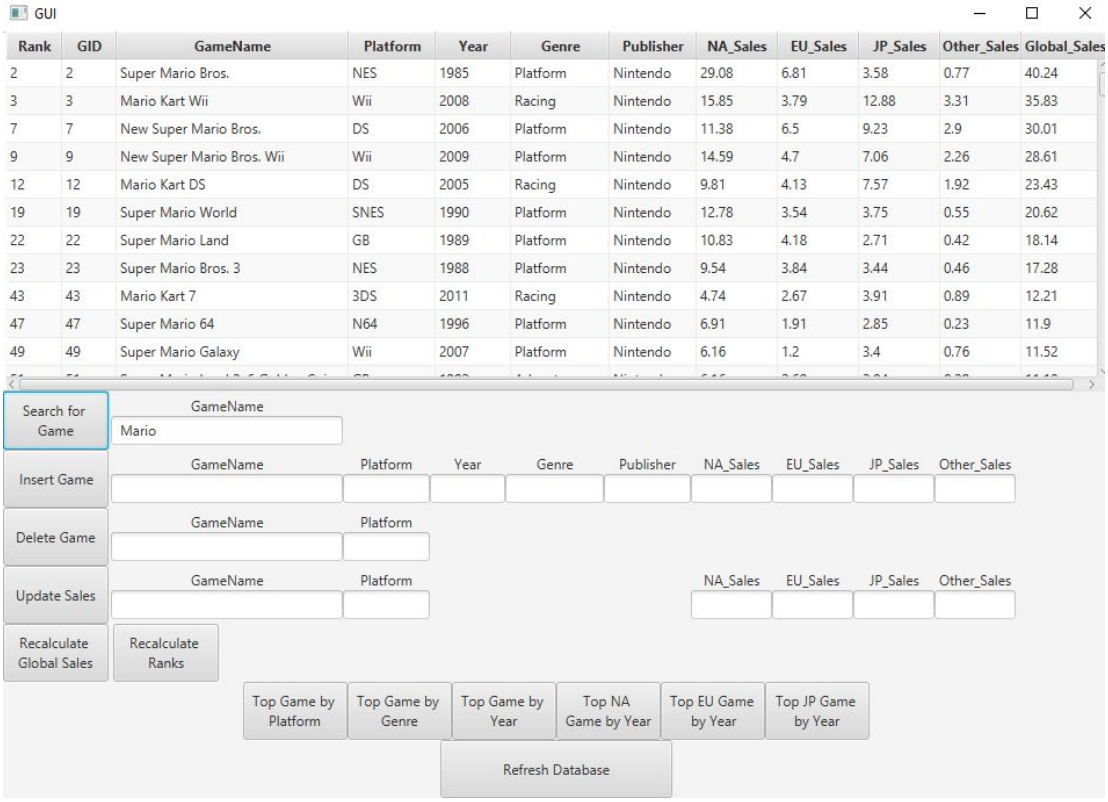
gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
3	4	Wii Sports Resort	Wii	2009	Sports	Nintendo	16.75	12.01	4.28	3.96	37

We incremented all sales in all regions by 1.00. Then we check if all the sales are correctly updated according to the changes. We expected the global_sales to be increased by 4, and we found the changes we made in the data are correctly updated in the database.

gamerank	gid	GameName	platform	year	genre	publishername	na_sales	eu_sales	jp_sales	other_sales	global_sales
4	4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33

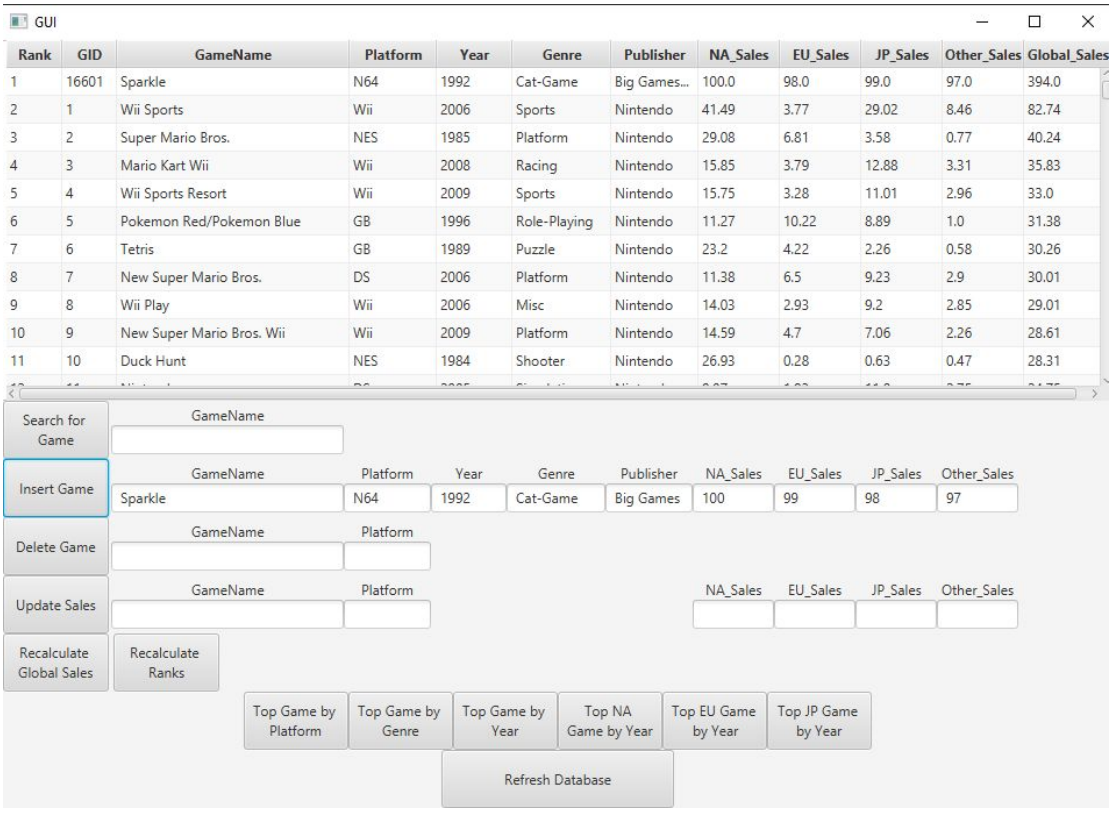
Then we decremented all sales in all regions by 1.00. We expected the data of the game should be exactly the same as the data before any change is made. The table above indicates the changes of the sales are correctly updated because it proves the data after decrementing 1.0 in sales per region

GUI Examples: Search, Insert, Delete, and Update Sales

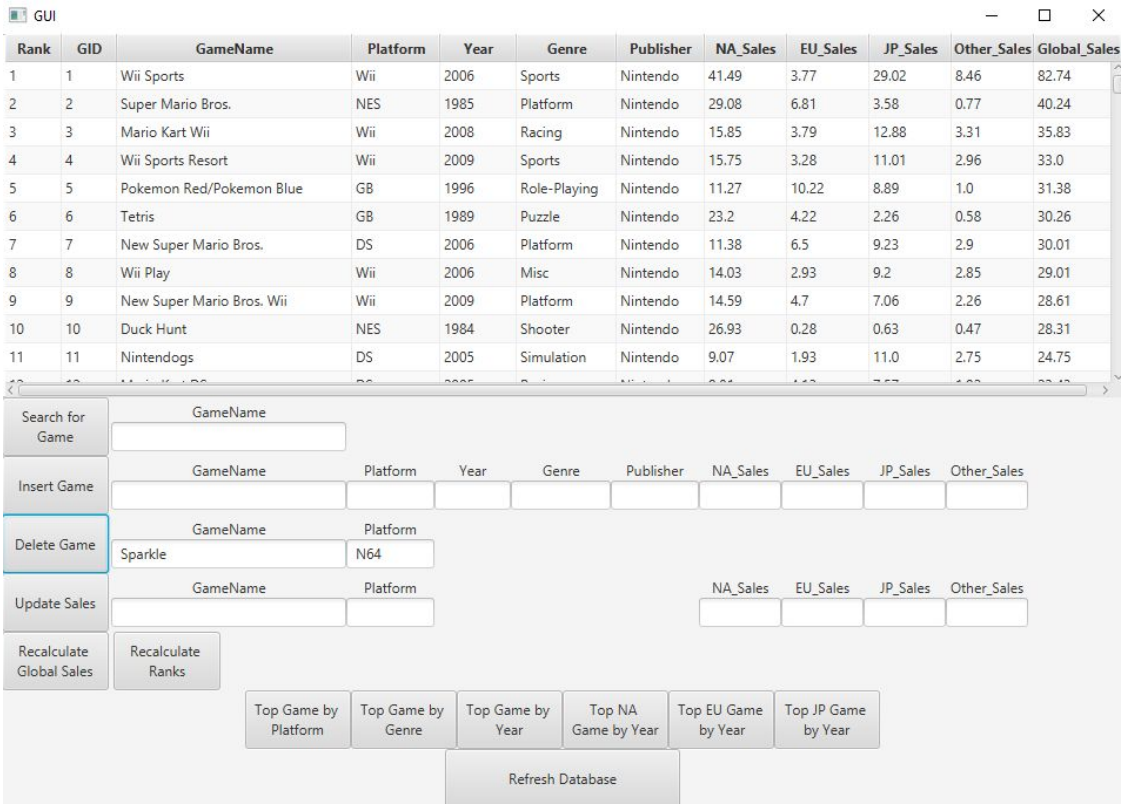


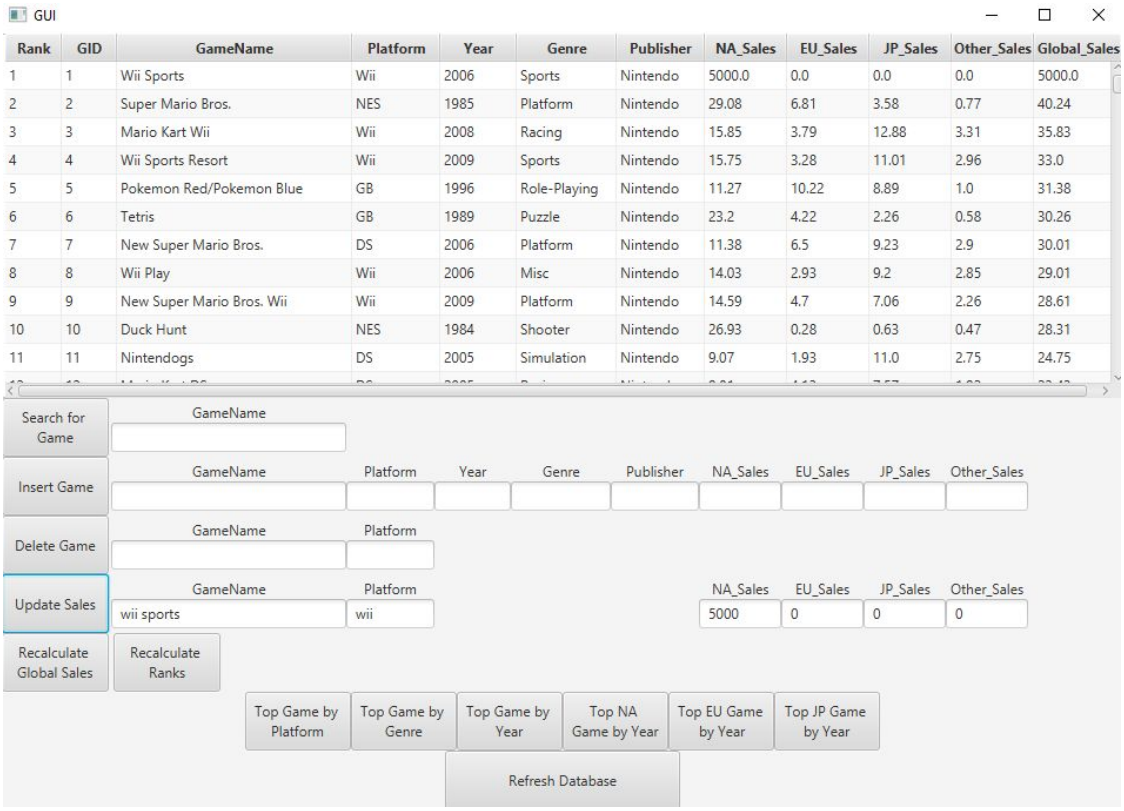
Rank	GID	GameName	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
2	2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	6.81	3.58	0.77	40.24
3	3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	3.79	12.88	3.31	35.83
7	7	New Super Mario Bros.	DS	2006	Platform	Nintendo	11.38	6.5	9.23	2.9	30.01
9	9	New Super Mario Bros. Wii	Wii	2009	Platform	Nintendo	14.59	4.7	7.06	2.26	28.61
12	12	Mario Kart DS	DS	2005	Racing	Nintendo	9.81	4.13	7.57	1.92	23.43
19	19	Super Mario World	SNES	1990	Platform	Nintendo	12.78	3.54	3.75	0.55	20.62
22	22	Super Mario Land	GB	1989	Platform	Nintendo	10.83	4.18	2.71	0.42	18.14
23	23	Super Mario Bros. 3	NES	1988	Platform	Nintendo	9.54	3.84	3.44	0.46	17.28
43	43	Mario Kart 7	3DS	2011	Racing	Nintendo	4.74	2.67	3.91	0.89	12.21
47	47	Super Mario 64	N64	1996	Platform	Nintendo	6.91	1.91	2.85	0.23	11.9
49	49	Super Mario Galaxy	Wii	2007	Platform	Nintendo	6.16	1.2	3.4	0.76	11.52

-



Rank	GID	GameName	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
1	16601	Sparkle	N64	1992	Cat-Game	Big Games...	100.0	98.0	99.0	97.0	394.0
2	1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	3.77	29.02	8.46	82.74
3	2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	6.81	3.58	0.77	40.24
4	3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	3.79	12.88	3.31	35.83
5	4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	3.28	11.01	2.96	33.0
6	5	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	11.27	10.22	8.89	1.0	31.38
7	6	Tetris	GB	1989	Puzzle	Nintendo	23.2	4.22	2.26	0.58	30.26
8	7	New Super Mario Bros.	DS	2006	Platform	Nintendo	11.38	6.5	9.23	2.9	30.01
9	8	Wii Play	Wii	2006	Misc	Nintendo	14.03	2.93	9.2	2.85	29.01
10	9	New Super Mario Bros. Wii	Wii	2009	Platform	Nintendo	14.59	4.7	7.06	2.26	28.61
11	10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.28	0.63	0.47	28.31


GUI
Rank GID GameName Platform Year Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales
1 1 Wii Sports Wii 2006 Sports Nintendo 41.49 3.77 29.02 8.46 82.74
2 2 Super Mario Bros. NES 1985 Platform Nintendo 29.08 6.81 3.58 0.77 40.24
3 3 Mario Kart Wii Wii 2008 Racing Nintendo 15.85 3.79 12.88 3.31 35.83
4 4 Wii Sports Resort Wii 2009 Sports Nintendo 15.75 3.28 11.01 2.96 33.0
5 5 Pokemon Red/Pokemon Blue GB 1996 Role-Playing Nintendo 11.27 10.22 8.89 1.0 31.38
6 6 Tetris GB 1989 Puzzle Nintendo 23.2 4.22 2.26 0.58 30.26
7 7 New Super Mario Bros. DS 2006 Platform Nintendo 11.38 6.5 9.23 2.9 30.01
8 8 Wii Play Wii 2006 Misc Nintendo 14.03 2.93 9.2 2.85 29.01
9 9 New Super Mario Bros. Wii Wii 2009 Platform Nintendo 14.59 4.7 7.06 2.26 28.61
10 10 Duck Hunt NES 1984 Shooter Nintendo 26.93 0.28 0.63 0.47 28.31
11 11 Nintendogs DS 2005 Simulation Nintendo 9.07 1.93 11.0 2.75 24.75
Search for Game
GameName
Insert Game
GameName Platform Year Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales
Delete Game
GameName Platform
Sparkle N64
Update Sales
GameName Platform NA_Sales EU_Sales JP_Sales Other_Sales
Recalculate Global Sales
Recalculate Ranks
Top Game by Platform
Top Game by Genre
Top Game by Year
Top NA Game by Year
Top EU Game by Year
Top JP Game by Year
Refresh Database


GUI
Rank GID GameName Platform Year Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales
1 1 Wii Sports Wii 2006 Sports Nintendo 5000.0 0.0 0.0 0.0 5000.0
2 2 Super Mario Bros. NES 1985 Platform Nintendo 29.08 6.81 3.58 0.77 40.24
3 3 Mario Kart Wii Wii 2008 Racing Nintendo 15.85 3.79 12.88 3.31 35.83
4 4 Wii Sports Resort Wii 2009 Sports Nintendo 15.75 3.28 11.01 2.96 33.0
5 5 Pokemon Red/Pokemon Blue GB 1996 Role-Playing Nintendo 11.27 10.22 8.89 1.0 31.38
6 6 Tetris GB 1989 Puzzle Nintendo 23.2 4.22 2.26 0.58 30.26
7 7 New Super Mario Bros. DS 2006 Platform Nintendo 11.38 6.5 9.23 2.9 30.01
8 8 Wii Play Wii 2006 Misc Nintendo 14.03 2.93 9.2 2.85 29.01
9 9 New Super Mario Bros. Wii Wii 2009 Platform Nintendo 14.59 4.7 7.06 2.26 28.61
10 10 Duck Hunt NES 1984 Shooter Nintendo 26.93 0.28 0.63 0.47 28.31
11 11 Nintendogs DS 2005 Simulation Nintendo 9.07 1.93 11.0 2.75 24.75
Search for Game
GameName
Insert Game
GameName Platform Year Genre Publisher NA_Sales EU_Sales JP_Sales Other_Sales
Delete Game
GameName Platform
Update Sales
GameName Platform NA_Sales EU_Sales JP_Sales Other_Sales
wii sports wii 5000 0 0 0
Recalculate Global Sales
Recalculate Ranks
Top Game by Platform
Top Game by Genre
Top Game by Year
Top NA Game by Year
Top EU Game by Year
Top JP Game by Year
Refresh Database

III. Conclusion

What do you learn from this project (both interesting and uninteresting points)? Have you found any relevant database knowledge you have learned in this course helpful and have you encountered any database relevant issues that have been discussed in this course?

3.1 Lessons Learned and Knowledge Gained

Back-end:We thought that the project gave us a broad/comprehensive overview of all the topics we had learned throughout the semester. We learned how to develop relational schema from a given dataset and learned how to normalize our data to eliminate data anomalies and duplicate data. As database implementers, we learned how to implement queries and stored procedures using MySQL. we learned how data is internally indexed/organized in the back-end by B+trees. Even though some of our functions did not work, we were able to get an idea on functionalities such as search, insert, and delete work on B+ trees

Front-end: At the front-end, we learned how to use the JDBC driver manager to connect our app with DB and interface between MySQL and the Java IDE. We also learned how to create a new cursor class for constructing objects which calls queries and stored procedures in MySQL. We also learned how to think from the user's perspective, by thinking about what functionalities a typical user might need in the front-end and implementing it through the JavaFX GUI to enhance UI/UX. Lastly, we learned communication skills and how to work together as a team by modularization and incremental design.

3.2 DB Relevant Issues

We had a few challenges handling the dataset. Initially, we thought that the dataset was clean, but it was not entirely clean. There were some "gaps" in the data, and some of the data had anomalies, rounding errors, and duplicates. For example, the global sales attribute determines the rank for each game, but several ranks were missing, so we had to recalculate the game ranks. Another issue we encountered was with normalization. Next, we realized that global sales was not consistent with regional sales; the dataset had rounding errors or wrong global sales data for some games, so we knew that that needed to be recalculated based on the regional sales data. Lastly, we found that there was no way for the Games Table to be in either BCNF or 3NF without adding something to the table. GameName was a non-unique, games with the same name were released for multiple systems or in multiple years or even by multiple publishers. A candidate key for the Game table would have required a composite key consisting of the name, platform, year, and publisher, which is not in 3NF with the multiple functional dependencies derived from this. The only unique attribute was the game rank, but we knew that the game rank was going to change as the sales data updated.