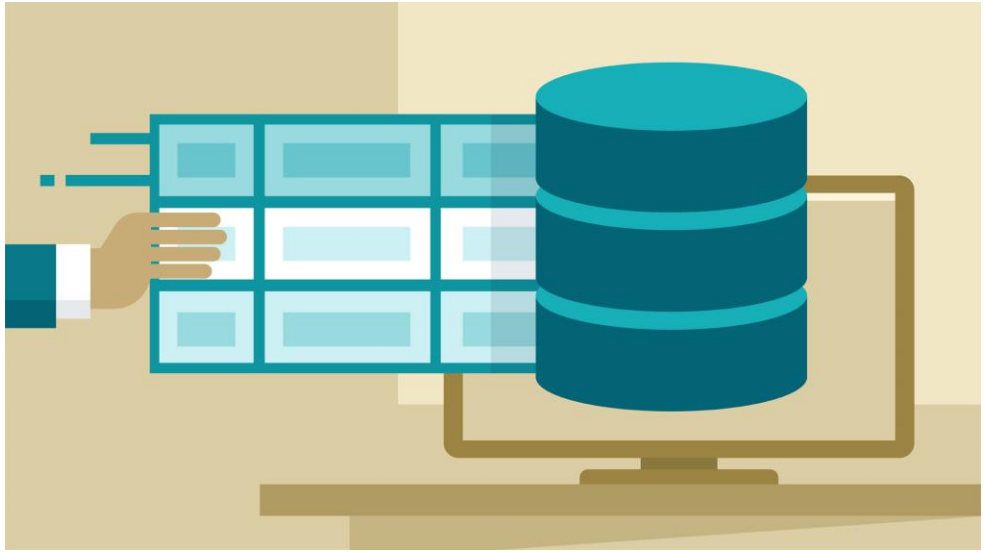
Video Game Sales Database
Group 8

- Tae Yong Namkoong, Hyuk Joon Yang, Nathan Sackett
- CS 564, Summer 2020 : Professor Nguyen



STEP 1 : PICK AN APPLICATION

1. Motivation and Domain

I. Motivation

- Analyze and predict sales trends in game market
- Video game market is rapidly growing and expanding → Requires real-time data access
- Provide meaningful demographic data in gaming industry
- Integrate relations between categorical information through queries

1. Motivation and Domain

II. The Need for Database & DBMS

- Millions of games are sold every year
- To track the success of a game, platform, genre, or company, necessary to look in context of its competition in various categories
- Efficient data storage/retrieval & data processing for users
- Intuitive UI/UX allows users to extract meaningful information from raw data through queries
- Fast, flexible, and real-time data access and data manipulation

2. Application Description

I. The Type of Application

- Web-based application accessible by multiple parties
- Real-time data access/manipulation/user queries.
- Supports concurrent access/modification of data as well as scalability to large datasets (big data)

2. Application Description

- **II. Functionalities**

- Handles User Queries:

SearchGame	SearchBy (category1, category2,…categoryN)
AddGame	getSalesBy
RemoveGame	RefreshDB
UpdateSales	UpdateDB
UpdateRank	GetNumGames
FindNewRank	

2. Application Description

III. Datasets

- Obtained videogamessales.csv from Kaggle.com
- Dataset contains the names of video games, the platform on which they are released, the year of publication, genre, publisher, and sales data for each region.
- **Reason:**

i. 11 columns of data and 16600 rows

ii. Dataset well-organized and clean

iii. Single source → no multiple source integration will be required

3. Project Management

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

STEP 2 : CONCEPTUAL DESIGN

1. ER Diagram

TO DO

STEP 3 : RELATIONAL SCHEMA

1. Entity Sets

TABLE 1. Game (GID: int, GameName: String, Platform: String, Year: String, Genre: String)

- GID – Unique Ranking; ≤ # of columns in table (Primary)
- GameName - Unique String; ≤ 255 Characters
- Platform – Unique String; ≤ 255 Characters
- Year - Date; 1900 ≤ Year ≤ 2200
- Genre - String; (Action, Adventure, Fighting, Misc, Platform, Puzzle, Racing, Role-Playing, Shooter, Simulation, Sports, Strategy)
- Primary Key : GID

1. Entity Sets

TABLE 2. Sales (GID: int, NA_Sales: double, EU_Sales: double, JP_Sales: double, Other_Sales: double, Global_Sales: double)

- GID – Unique Ranking; ≤ # of columns in table (Primary)
- 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)

1. Entity Sets

TABLE 3. Publisher (GID: int, PublisherName: String)

- GID – Unique Ranking; ≤ # of columns in table (Primary)
- PublisherName - Unique String; ≤ 255 Characters
- Primary Key : GID
- Foreign Key : (GID) references Game(GID)

1. Entity Sets

TABLE 4. Ranking (GID: int, GameRank: int, Global_Sales: double)

- GID – Unique Ranking; ≤ # of columns in table (primary)
- GameRank - Unique Ranking; ≤ # of columns 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)

2. Relationship Sets

1. On(GameName: String, PlatformName: String)
   - Game — ◆ — Platform (many to many)
2. Sold (GameName: String, Region: String)
   - Game — ◆ — Sales (many to many)
3. Made (GameName: String, PublisherName: String)
   - Game — ◆ —) Publisher (many to many, total participation, referential)

STEP 4 : NORMALIZATION PROCESS

ENSURE 3NF

- Games Table was in neither BCNF nor 3NF initially
- PROBLEM: GameName was a non-unique, non-candidate key because there were duplicate elements with many non-trivial functional dependencies
- ex) There were some games with identical names published in the same year (duplicates & data anomalies)
- SOLUTION : We made all columns (attributes) functionally dependent on the GID which was the unique ranking assigned to each game
- Now GID is primary key
- For Sales table: Global_Sales is sum of all regional sales so we had to make both GID and Global_Sales as primary key
- Guarantees lossless decomposition & dependency preserving

```
/* Update GameRank of Ranking*/
CREATE TEMPORARY TABLE temp
    select *
    from Ranking;

update Ranking
    inner join temp
    on temp.gid = ranking.gid
    set ranking.gamerank= temp.gid;

drop table temp;
```

NON-TRIVIAL FUNCTIONAL DEPENDENCIES

RELATION	FUNCTIONAL DEPENDENCIES	3NF ACHIEVED?
Game	**FDGame** = {GID → (All Attributes of Game)	All attributes depend on GID (primary key)
Sales	**FDSales** = {(GID,Global_Sales) → (All Attributes of Sales)	RHS is prime attribute
Publisher	**FDPublisher** = {GID → (All Attributes of Publisher)	All attributes depend on GID (primary key)
Ranking	**FDRanking** = {GID → (All Attributes of Ranking)	All attributes depend on GID (primary key)

Step 5,6: Pick a DBMS and implement database

1. BACK-END DBMS: MySQL

- Relational DBMS based on Structured Queries
- Scalable through multi-threading ➔ supports big data
- Simple syntax & DDL/DML covered in lecture
- GUI Support (MySQL Workbench) ➔ Easy to Use & high functionality
- Bridges gap between logical model and machine (physical data independence & logical data independence)
- Supports .csv

Step 5,6: Pick a DBMS and implement database

2. FRONT-END GUI: JavaFX

- Team members have experience from CS400
- FXML intuitive & easy to use
- Can be integrated with back-end MySQL
- Enables UI tests
- Built-in functionalities (buttons, frames, panels, etc)

BACK-END DEMONSTRATION & EVALUATION (TESTING)

STORED PROCEDURES:

updateGlobalSales()

```
/*This procedure is used to update the Global_Sales of games*/
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 ;
```

GID	GameRank	Global_Sales
1	1	82.74
2	2	40.24
3	3	35.82
4	4	33
5	5	31.37
6	6	30.26
7	7	30.01
8	8	29.02
9	9	28.62
10	10	28.31
11	11	24.76
12	12	23.42
13	13	23.1
14	14	22.72
15	15	22
16	16	21.82
17	17	21.4
18	18	20.81
19	19	20.61
20	20	20.22
21	21	18.36
22	22	18.14
23	23	17.28
24	24	16.38
25	25	16.15
26	26	15.85

GID	GameRank	Global_Sales
1	1	82.74
2	2	40.24
3	3	35.83
4	4	33
5	5	31.38
6	6	30.26
7	7	30.01
8	8	29.01
9	9	28.61
10	10	28.31
11	11	24.75
12	12	23.43
13	13	23.09
14	14	22.72
15	15	22
16	16	21.82
17	17	21.39
18	18	20.81
19	19	20.62
20	20	20.22
21	21	18.35
22	22	18.14
23	23	17.28
24	24	16.38
25	25	16.15
26	26	15.84
27	27	15.32

STORED PROCEDURES:

makeTempUpdate()

```
/*
makeTempUpdate creates a temp table used to find the new rank of a game
*/
delimiter $$
drop procedure if exists makeTempUpdate;
create procedure makeTempUpdate()
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;
end $$
delimiter ;
```

	gid	r
▶	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	10	10
	11	11
	12	12
	13	13
	14	14
	15	15
	16	16
	17	17
	18	18
	19	19
	20	20
	21	21
	22	22
	23	23
	24	24
	25	25
	26	26

STORED PROCEDURES:

updateRank()

```
/*
updateRank updates the gamerank of a game based on its global sales data
*/
delimiter //
drop procedure if exists updateRank;
create procedure updateRank()
begin
    /*Make temp table*/
    call makeTempUpdate;

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

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

GID	GameRank	Global_Sales
1	1	82.74000000000001
2	2	40.24
3	3	35.830000000000005
4	4	33
5	5	31.380000000000003
6	6	30.259999999999998
7	7	30.009999999999998
8	8	29.009999999999998
9	9	28.61
10	10	28.31
11	11	24.75
12	12	23.43
13	13	23.09
14	14	22.72
15	15	22
16	16	21.82
17	17	21.39
18	18	20.810000000000002
19	19	20.62
20	20	20.220000000000002
21	21	18.35
22	22	18.14
23	23	17.28
24	24	16.380000000000003
25	25	16.150000000000002
26	26	15.84

STORED PROCEDURES:

searchGame()

```
/* Search function that searches all sales of all platforms of the game */
delimiter $$
drop procedure if exists searchGame;
create procedure searchGame (in gameName varchar(135))
begin
    select gameName as Game_Name, g1.platform, s.NA_Sales, s.EU_Sales, s.JP_Sales, s.Other_Sales, s.Global_Sales
    from Sales s, (select g.gid, g.platform
        from Game g
        where g.GameName = gameName)g1
    where s.GID = g1.GID;
end $$
delimiter ;
```

```
124 •  call searchGame('Super Mario Land');
125
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

gid	platform	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
22	GB	10.83	2.71	4.18	0.42	18.14

```
80 •  call searchGame('Call of Duty: Black Ops');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Game_Name	platform	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
Call of Duty: Black Ops	X360	9.67	3.73	0.11	1.13	14.64
Call of Duty: Black Ops	PS3	5.98	4.44	0.48	1.83	12.73
Call of Duty: Black Ops	PC	0.58	0.81	0	0.24	1.63
Call of Duty: Black Ops	Wii	0.82	0.47	0	0.12	1.41
Call of Duty: Black Ops	DS	0.54	0.05	0	0.04	0.63

STORED PROCEDURES:

insertGame()

```
/* Store procedure that inserts user inputs into this database*/
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;

    /* 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);

    /* 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 $$
delimiter ;
```

GameName	GameRank
Test1	2
Test1	1
Spirits & Spells	16599
Know How 2	16598
SCORE International Baja 1000: The Official Game	16600
Men in Black II: Alien Escape	16597
Woody Woodpecker in Crazy Castle 5	16596
Plushees	16595
Myst IV: Revelation	16594
Eiyuu Densetsu: Sora no Kiseki Material Collecti...	16593
Chou Ezaru wa Akai Hana: Koi wa Tsuki ni Shiru...	16592
Mega Brain Boost	16591
Mezase!! Tsuri Master DS	16590
Secret Files 2: Puritas Cordis	16589
Breach	16588
Bust-A-Move 3000	16587
Carmageddon 64	16586
Planet Monsters	16585
Fit & Fun	16584

STORED PROCEDURES:

removeGame()

TO DO

LEVEL 1 QUERIES:

Total # of Games for each platform

```
/* Total number of Games per each platform */
select platform, count(*) as Total
from Game
group by platform
order by Total desc;
```

platform	Total
DS	2163
PS2	2161
PS3	1329
Wii	1325
X360	1265
PSP	1213
PS	1196
PC	960
XB	824

LEVEL 1 QUERIES:

Games published per year

```
/* Games published in per year*/
select Year, GameName, Platform
from Game
where Year <> 'N/A'
order by year desc;
```

Year	GameName	Platform
2020	Imagine: Makeup Artist	DS
2017	Phantasy Star Online 2 Episode 4: Deluxe Package	PS4
2017	Brothers Conflict: Precious Baby	PSV
2017	Phantasy Star Online 2 Episode 4: Deluxe Package	PSV
2016	Assassin's Creed Chronicles	PSV
2016	Psycho-Pass: Mandatory Happiness	PSV
2016	Attack on Titan (KOEI)	PSV
2016	Assassin's Creed Chronicles	XOne

LEVEL 2 QUERIES:

Sales By Publisher

```
/* Sales by Publisher*/
select p.PublisherName as Publisher, ROUND(sum(s.NA_Sales), 2) as NA_Sales,
    round((s.EU_Sales), 2)as EU_Sales, round(sum(JP_Sales), 2) as JP_Sales,
    round(sum(Other_Sales),2) as Other_Sales, round(sum(Global_Sales),2) as Global_Sales
from Publisher p, Sales s
where p.GID = s.GID
group by p.PublisherName;
```

Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
Nintendo	816.87	29.02	455.42	95.33	1786.36
Microsoft Game Studios	155.35	4.94	3.26	18.56	245.78
Take-Two Interactive	220.49	9.27	5.83	55.24	399.70
Sony Computer Entertainment	265.22	5.09	74.10	80.45	607.49
Activision	429.70	4.28	6.54	75.34	727.11
Ubisoft	253.43	3.15	7.50	50.26	474.51
Bethesda Softworks	39.72	2.86	1.45	10.16	82.06
Electronic Arts	595.07	6.06	14.04	129.77	1110.15
Sega	109.40	3.90	57.03	24.52	272.95
SquareSoft	11.06	1.72	40.13	1.54	57.68
Atari	110.04	0.45	10.71	9.01	156.88
505 Games	31.83	2.64	2.05	5.44	55.75

LEVEL 2 QUERIES:

Sales By Genre

```
/* Sales by genre*/
select g.Genre as Genre, ROUND(sum(s.NA_Sales), 2) as NA_Sales,
    round((s.EU_Sales), 2)as EU_Sales, round(sum(JP_Sales), 2) as JP_Sales,
    round(sum(Other_Sales),2) as Other_Sales, round(sum(Global_Sales),2) as Global_Sales
from Game g, Sales s
where g.GID = s.GID
group by g.Genre;
```

Genre	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
Sports	683.35	29.02	135.37	134.97	1330.54
Platform	447.05	3.58	130.77	51.59	831.04
Racing	359.42	12.88	56.69	77.27	731.77
Role-Playing	327.28	8.89	352.31	59.61	927.26
Puzzle	123.78	2.26	57.31	12.55	244.42
Misc	410.24	9.20	107.76	75.32	809.30
Shooter	582.60	0.63	38.28	102.69	1036.84
Simulation	183.31	11.00	63.70	31.52	391.91
Action	877.83	9.27	159.95	187.38	1750.16
Fighting	223.59	2.61	87.35	36.68	448.94
Adventure	105.80	2.04	52.07	16.81	238.81
Strategy	68.70	1.24	49.46	11.36	174.86

LEVEL 2 QUERIES:

Sales By Platform

```
/* Sales by platform */
select g.platform as platform, ROUND(sum(s.NA_Sales), 2) as NA_Sales,
    round((s.EU_Sales), 2)as EU_Sales, round(sum(JP_Sales), 2) as JP_Sales,
    round(sum(Other_Sales),2) as Other_Sales, round(sum(Global_Sales),2) as Global_Sales
from Game g, Sales s
where g.GID = s.GID
group by g.platform;
```

platform	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
Wii	507.71	29.02	69.35	80.61	926.05
NES	125.94	3.58	98.65	5.31	251.05
GB	114.32	8.89	85.12	8.20	255.46
DS	390.71	9.23	175.57	60.53	821.46
X360	601.05	4.94	12.43	85.54	979.60
PS3	392.26	9.27	79.99	141.93	957.89
PS2	583.84	0.40	139.20	193.44	1255.77
SNES	61.23	3.75	116.55	3.22	200.04

LEVEL 2 QUERIES:

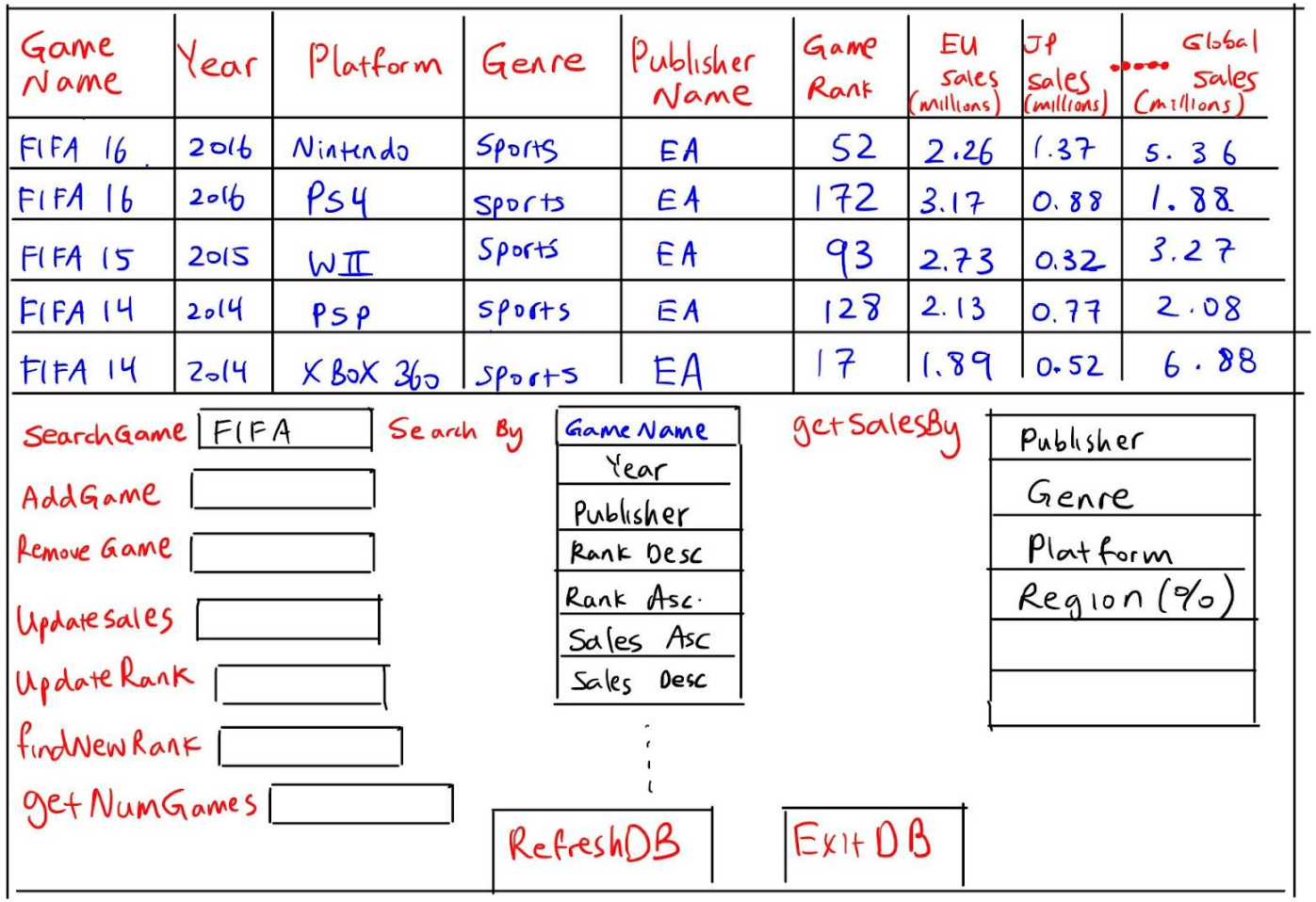
Distribution of Sales for each Game

```
/* Distribution of sales per game*/
select g.GameName, g.Platform, concat(round(s.NA_Sales/s.Global_Sales, 2), '%') as '% NA',
    concat(round(s.EU_Sales/s.Global_Sales, 2), '%') as '% EU',
    concat(round(s.JP_Sales/s.Global_Sales, 2), '%') as '% JP',
    concat(round(s.Other_Sales/s.Global_Sales, 2), '%') as '% Other'
from Game g, Sales s
where g.GID = s.GID;
```

GameName	Platform	% NA	% EU	% JP	% Other
Wii Sports	Wii	0.50%	0.35%	0.05%	0.10%
Super Mario Bros.	NES	0.72%	0.09%	0.17%	0.02%
Mario Kart Wii	Wii	0.44%	0.36%	0.11%	0.09%
Wii Sports Resort	Wii	0.48%	0.33%	0.10%	0.09%
Pokemon Red/Pokemon Blue	GB	0.36%	0.28%	0.33%	0.03%
Tetris	GB	0.77%	0.07%	0.14%	0.02%
New Super Mario Bros.	DS	0.38%	0.31%	0.22%	0.10%
Wii Play	Wii	0.48%	0.32%	0.10%	0.10%
New Super Mario Bros. Wii	Wii	0.51%	0.25%	0.16%	0.08%

Architecture Diagram

FRONT-END DEMONSTRATION



COPY PASTE GUI HERE