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

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 ᅳ♦ᅳ Platform (many to many) 2. Sold (GameName: String, Region: String)

- Game ᅳ♦ᅳ Sales (many to many) 3. Made (GameName: String, PublisherName: String)

- Game ᅳ♦ᅳ) 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

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()

STORED PROCEDURES:

makeTempUpdate()

STORED PROCEDURES:

updateRank()

STORED PROCEDURES:

searchGame()

STORED PROCEDURES:

insertGame()

STORED PROCEDURES:

removeGame()

TO DO

LEVEL 1 QUERIES:

Total # of Games for each platform

LEVEL 1 QUERIES:

Games published per year

LEVEL 2 QUERIES:

Sales By Publisher

LEVEL 2 QUERIES:

Sales By Genre

LEVEL 2 QUERIES:

Sales By Platform

LEVEL 2 QUERIES:

Distribution of Sales for each Game

Architecture Diagram

FRONT-END DEMONSTRATION

COPY PASTE GUI HERE