University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: Nov, 29, 2024

Group Number: 89

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Charlotte Du	86020567	p4n7e	chardudev@outlook.com
Ivy Lin	37133345	y4e4v	ivy.lin128@gmail.com
Haider Khan	46046181	r1t2l	khanh3777@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

MS4:

Description:

Our final project involved designing a relational database to manage video games and their related studios, platforms, and awards—accessible through a web-based interface. The database (backend) also includes information on many other topics relating to each game such as genre tags, game ratings, system requirements, events, and event organizers. Our web based interface supported CRUD operations and featured 10 queries such as searching for studios via specific values, filtering games by platform, displaying the top-rated studios, award-winning games, and more. Moreover, our final project ensures data integrity through well-defined relationships and processes user interactions safely through input sanitization and error handling. We were also able to design a user-friendly interface that allows seamless interaction with the data through query forms and results that are easy to understand and navigate as well as user notification messages. This project demonstrated proficiency in database design, query optimization, and full-stack development. It achieved the goal of providing an interactive platform for managing and analyzing game-related data.

Schema

- Change relation name “On” to GameOnPlatform because “On” is a preserved keyword in Oracle.
- Got rid of NULL constraint for priceCAD in GameOnPlatform (NULL now indicates game is free)
- We no longer normalize the EVENT relation into BCNF because it in fact is lossy (we don't have a relation with all the attributes of a key). We do 3NF instead, and the EVENT relation is already in 3NF.
- Prequel relation: TA said it is not necessary to have -1 as default value because it means nothing to have a null relation with another attribute in a 2-attribute relation.

A list of all SQL queries

2.1.1 INSERT
In: appService.js
Line: 152-253

```
// Inserting Studio
INSERT INTO Studio (studioID, studioName, country, studioType,
yearFound)
VALUES (:studioID, :studioName, :country, :studioType,
:yearFound)`

// Insert Game
`INSERT INTO Game (gameID, gameTitle, crossplay)
 VALUES (:gameID, :gameTitle, :crossplay)`

// Insert into Creates table
`INSERT INTO Creates (gameID, studioID)
 VALUES (:gameID, :studioID)`
```

2.1.2 UPDATE

In: appService.js
Line: 255 - 283

```
UPDATE Game SET ${updateFields.join(", ")}
WHERE gameID = :gameID
```

2.1.3 DELETE

In: appService.js
Line: 285 - 297

```
DELETE FROM Game WHERE gameID = :gameID
```

2.1.4 SELECTION

In: appService.js
Line: 413 - 438

```
SELECT studioID, studioName, studioType, country, yearFound FROM
Studio

queryConditions.push(`${attribute} ${comparisonOperator}
:param${index}`);
queryParams.push(value);

selectionQuery += ' WHERE ' + queryConditions.join(' ');
connection.execute(selectionQuery, queryParams);
```

2.1.5 PROJECTION

In: appService.js
Line: 378 - 410

```
SELECT ${processedCols}
FROM Game g, AwardEvent ae, AwardInfo ai
WHERE ae.gameID = g.gameID AND ai.awdTitle = ae.awdTitle
ORDER BY ae.awdYear DESC

processedCols = columns.map(col => {
   if (['gameID', 'gameTitle'].includes(col)) {
       return `g.${col}`;
   } else if (['awdTitle', 'awdYear'].includes(col)) {
       return `ae.${col}`;
   } else if (['awdDescription'].includes(col)) {
       return `ai.${col}`;
   }
   return col;
}).join(", ");
```

2.1.6 JOIN

In: appService.js
Line: 299 - 326

```
SELECT g.gameID, g.gameTitle, p.platformName, p.platformType,
gop.releaseDate, gop.priceCAD
FROM Game g, GameOnPlatform gop, Platform p
WHERE g.gameID = gop.gameID AND p.platformName = gop.platformName
```

2.1.7 AGGREGATION W/ GROUP BY

In: appService.js
Line: 328 - 344

Description: find the names of all studios and the number of games they have created, sorted in descending order of the game count.

```
SELECT s.studioName, COUNT(c.gameID) AS gameCount
FROM Studio s
      LEFT JOIN Creates c ON s.studioID c.studioID
GROUP BY s.studioName
ORDER BY gameCount DESC
```

2.1.8 AGGREGATION W/ HAVING

In: appService.js
Line: 347 - 364

Description: find the titles of games that have won at least one award. For each game, include the total count of awards it has received and sort in descending order.

```
SELECT g.gameTitle, COUNT(ae.awdTitle) AS awardCount
FROM Game g, AwardEvent ae
WHERE g.gameID = ae.gameID
GROUP BY g.gameTitle
HAVING COUNT(ae.awdTitle) > 0
```

2.1.9 NESTED AGGREGATION W/ GROUP BY

In: appService.js
Line: 464 - 481

Description: find the name, type, country and founding year of studios whose average game rating (across all released games) is higher than the overall average of all studios. For each studio, display their average rating and sort in descending order.

```
SELECT st.studioName, st.studioType, st.country, st.yearFound,
AVG(gr.ratingScore) AS AverageRating
FROM Studio st, Creates cr, Game g, AvgRating gr
WHERE st.studioID = cr.studioID AND cr.gameID = g.gameID AND
g.gameID = gr.gameID
GROUP BY st.studioName, st.studioType, st.country, st.yearFound
HAVING AVG(gr.ratingScore) > (
    SELECT AVG(gr2.ratingScore)
    FROM AvgRating gr2)
```

2.1.10 DIVISION

In: appService.js
Line: 441 - 461

Description: find the game titles, a concatenated list of all platform names, and crossplay status of the games that are available on ALL platform types listed in the Platform table.

```
SELECT g.gameTitle, LISTAGG(gp.platformName, ', ') WITHIN GROUP
(ORDER BY gp.platformName) AS platforms, g.crossPlay
FROM Game g, GameOnPlatform gp
WHERE g.gameID = gp.gameID AND
```

```
NOT EXISTS (SELECT pt.platformType
      FROM Platform pt
      MINUS
      SELECT p.platformType
      FROM GameOnPlatform gp2, Platform p
      WHERE gp2.gameID = g.gameID AND gp2.platformName =
      p.platformName)
GROUP BY g.gameTitle, g.crossplay
```