Steam Games Query Processing: A Technical Report

Jon Piolo C. Jacinto[1], Marion Jose S. Manipol[2], and Carl Gabriel C. Yap[3]

De La Salle University Manila

[1]jon_jacinto@dlsu.edu.ph, [2]marion_manipol@dlsu.edu.ph, [3]carl_gabriel_yap@dlsu.edu.ph

ABSTRACT

This report details the application of Online Analytical Processing (OLAP) to an instructor-provided dataset, intending to teach students about key concepts in data warehousing and query optimization. First, we design a star schema (Appendix A) to organize the data for efficient querying. Next, we develop an OLAP application that generates analytical reports based on multidimensional queries. After running several queries, we evaluate their performance and identify areas for improvement. Finally, we propose optimization strategies to enhance query efficiency, such as indexing and partitioning. This project provides hands-on experience in building data warehouses, working with OLAP tools, and optimizing queries, helping students gain skills they can apply to real-world data analysis tasks.

Keywords

Data Warehouse, ETL, OLAP, Query Processing, Query Optimization

1. Introduction

Online Analytical Processing (OLAP) plays a pivotal role in enabling the analysis of large datasets, particularly in scenarios where multidimensional queries and trend analyses are required. Leveraging OLAP with a robust data warehousing infrastructure allows for the efficient organization, storage, and retrieval of data for complex business intelligence tasks. The process begins with Extract, Transform, Load (ETL) procedures, where raw data from various sources is consolidated, transformed into a standardized format, and stored within a data warehouse. This structured data can be utilized for comprehensive analyses, providing valuable insights into underlying trends and patterns.

In this study, we explore the application of OLAP to the Steam Games DB, a dataset comprising game metrics, user scores, and categories for over 97,000 titles. This dataset offers a rich source of information on game performance, user engagement, and industry trends. By applying OLAP methodologies to this dataset, we aim to uncover actionable insights to inform recommendation systems, market analyses, and predictive modeling within the gaming industry.

2. Data Warehouse

This section outlines the implementation of the star schema for the data warehouse. The design includes two regular dimensions, discussed in Section 2.1, followed by two special dimensions in Section 2.2. Lastly, Section 2.3 presents the structure and details of the central fact table.

2.1 Company and Game Data

<Text>

Table 1. Columns of Company dimension

Column Name	Data Type
Item #1	N (%)
Item #2	N (%)

2.2 Categories and Genres

<Text>

2.3 Game Metrics

Column Name	Data Type
Item #1	N (%)
Item #2	N (%)

3. ETL Script

Creating the game dimension table

```
Unset
1. Create dim_game table:
   - Define columns:
     - gameID (Primary Key)
```

```
     - name (Not Null)
     - releaseDate (Date)
     - requiredAge
     - aboutTheGame
     - websiteURL
     - supportURL
     - supportEmail
     - windowsSupport, macSupport,
linuxSupport (Boolean)
     - supportedLanguages
     - fullAudioLanguages
     - headerImageHREF
     - categories, tags, genres
(Text)

2. Create Full-Text Indexes:
   - Full-Text index on tags
   - Full-Text index on genres
   - Full-Text index on categories

3. Create temp_game_data table:
   - Define columns:
     - AppID
     - name
     - releaseDate (as String)
     - requiredAge
     - aboutTheGame
     - websiteURL
     - supportURL
     - supportEmail
     - supportedLanguages
     - fullAudioLanguages
     - headerImageHREF
     - categories, tags, genres

4. Import CSV into temp_game_data.

5. Insert Data into dim_game:
   - Insert AppID, name, and other
fields from temp_game_data into
dim_game.
   - Convert releaseDate:

6. Drop temp_game_data table.
```

Creating the company dimension table

```
Unset
1. Create dim_company table:
   - Define columns:
     - companyID (Primary Key, Auto
Increment)
     - developer
     - publisher

2. Create temp_game_data table:
   - Define columns:
     - developer
     - publisher

3. Import CSV into temp_game_data.

4. Insert Data into dim_company:
   - Insert DISTINCT developer and
publisher pairs from temp_game_data.
   - Exclude pairs already present
in dim_company.

5. Drop temp_game_data table.
```

Creating the fact table

```
Unset
1. Create fact_GameMetrics table:
   - Define columns:
     - gameID (INT, FK to dim_game)
     - companyID (INT, FK to
dim_company)
     - osID (VARCHAR(3), FK to
dim_OS)
     - price (FLOAT)
     - peakCCU (INT)
     - achievementCount (INT)
     - averagePlaytimeForever
(FLOAT)
     - medianPlaytimeForever (FLOAT)
     - estimatedOwners
(VARCHAR(255))
     - dlcCount (INT)

2. Populate gameID:
```

```
   INSERT INTO fact_GameMetrics
(gameID, price, peakCCU,
achievementCount,
averagePlaytimeForever,
medianPlaytimeForever,
estimatedOwners, dlcCount)
   SELECT g.gameID, gd.price,
gd.peakCCU, gd.achievementCount,
gd.averagePlaytimeForever,
gd.medianPlaytimeForever,
gd.estimatedOwners, gd.dlcCount
   FROM game_data gd
   JOIN dim_game g ON gd.AppID =
g.gameID
   WHERE gd.AppID IS NOT NULL;

3. Populate companyID:
   CREATE TEMPORARY TABLE
TempCompanyMapping AS
   SELECT gd.AppID, dc.companyID
   FROM game_data gd
   JOIN dim_company dc ON
gd.Developer = dc.developer AND
gd.Publisher = dc.publisher
   WHERE gd.AppID IS NOT NULL;

   UPDATE fact_GameMetrics fgm
   JOIN TempCompanyMapping tcm ON
fgm.gameID = tcm.AppID
   SET fgm.companyID = tcm.companyID
   WHERE fgm.companyID IS NULL;

   DROP TEMPORARY TABLE
TempCompanyMapping;

4. Populate osID:
   UPDATE fact_GameMetrics fgm
   JOIN dim_game dg ON fgm.gameID =
dg.gameID
   JOIN dim_OS dos ON
dg.windowsSupport =
dos.windowsSupport
                    AND
dg.macSupport = dos.macSupport
                    AND
dg.linuxSupport = dos.linuxSupport
   SET fgm.osID = dos.osID;
```

4. OLAP Application

This section

5. Query Processing and Optimization

This section

6. Results and Analysis

This section

7. Conclusion

This section

8. References

[1]

[2]

[3]

[4]

A. Appendix

Some conference papers include an Appendix where authors can place supplementary materials.

For our Technical Report, supplementary materials may include code snippets, SQL statements, sample extract of the dataset, and sample query results. These are used to help your readers better understand your implementation and your discussion.

Note that in the appendix, sections are lettered, not numbered.