Project 3 - DBMS Models and Implementation

Group: 2

Thanh Cao

Zaid Issa

Nam Nguyen

CSE-5331-001

April 23, 2024

University of Texas at Arlington

I. Overall Status

For task 1 we were able to implement everything correctly, the mapper the reducer, along with the datasheet. For the mapper we basically split the fields into different variables for example Type was in a String var, the rating was stored inside of a double. After we stored the variables we ran an if statement that checked to see if the year was between 1991 and 2020 along with the rating being greater than 7.5 and the type is a movie. We checked what year it belonged to(example [1991-2000)], then we added it to a hashset arrayList. After that we checked to see which genre it belonged to and then wrote it to the file as 1. We used a combiner to group the same year type genre and yeartype together. After that we partitioned the key and value depending on the key itself. We then ran it through the reducer which combined all the results from the map. We also have a block of code that checks if the directory already exists if so we delete it and overwrite it with a new output directory.

For task 2, we were able to use sqlplus on Omega to implement the query necessary to get the correct output (top 5 action, thriller movies with 150000 votes between the years 2001 and 2010) . We also got the query plan from using the EXPLAIN statement and put everything in one output file. The output file contains the query used to get the top 5 movies based on their ratings and the EXPLAIN statement for that query.

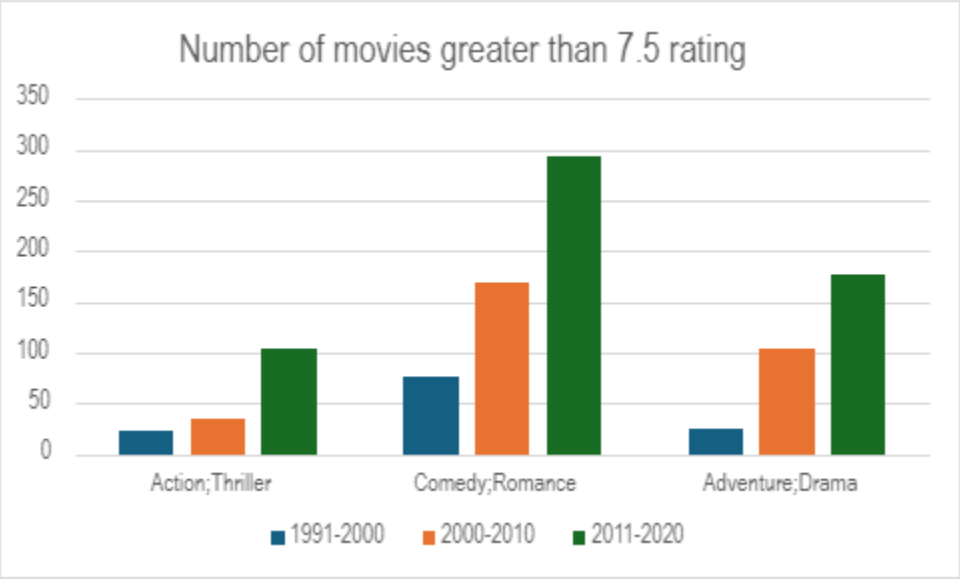
II. Analysis Results

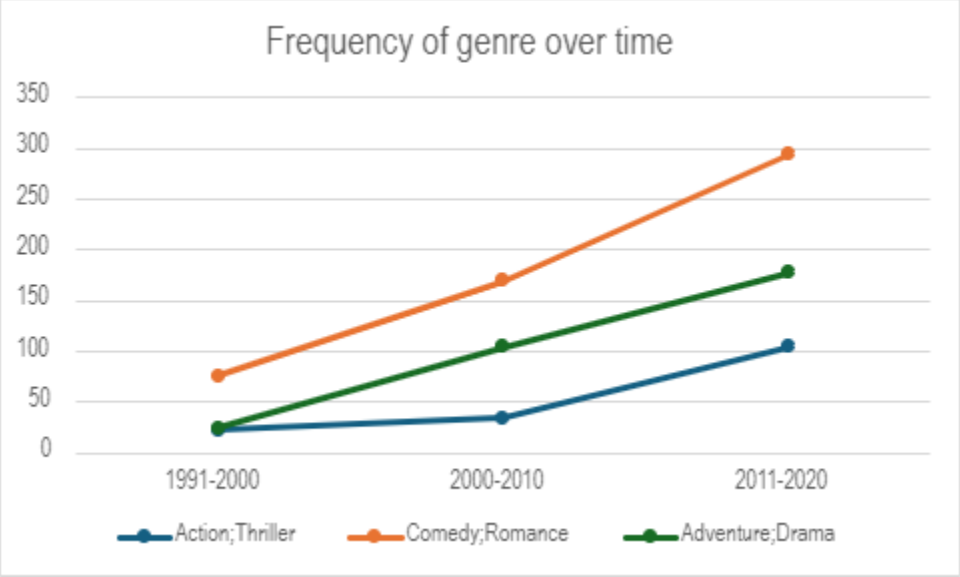
[TASK 1-M/R]

Our results show that there is an increase in popularity and rating for movies throughout the years.

	1991-2000	2000-2010	2011-2020
Action;Thriller	23	34	105
Comedy;Romance	76	170	294
Adventure;Drama	25	105	178

As the results show each genre throughout the years have an increase, we believe this is due to an increase of movie popularity and the access of technology. We also did a line graph and a clustered column.





[TASK 2-SQL]
Iv, v)

```
SQL>
SQL> --Retrieve movie titles from the imdb00.TITLE_BASICS table (alias as tb) and their average ratings from the imdb00.TITLE_RATINGS table (alias as tr)
SQL> SELECT tb.PRIMARYTITLE AS "Movie Title", tr.AVERAGERATING AS "Average Rating"
  2  FROM imdb00.TITLE_BASICS tb
  3  --Join the two tables (based on their shared attribute) where the title type is 'movie', the release year is between 2001 and 2010,
  4  --the number of votes is at least 150000, the genre combination is 'Action' and 'Thriller'
  5  JOIN imdb00.TITLE_RATINGS tr ON tb.TCONST = tr.TCONST
  6  WHERE tb.TITLETYPE = 'movie'
  7  AND tb.STARTYEAR BETWEEN '2001' AND '2010'
  8  AND tr.NUMVOTES >= 150000
  9  AND tb.GENRES LIKE '%Action%'
 10  AND tb.GENRES LIKE '%Thriller%'
 11  --Order the results retrieved by the average rating in a descending order
 12  ORDER BY tr.AVERAGERATING DESC
 13  --Fetch only the top 5 results
 14  FETCH FIRST 5 ROWS ONLY;

Movie Title
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Average Rating
--------------
The Bourne Ultimatum
              8

Casino Royale
              8

Kill Bill: Vol. 2
              8

The Bourne Identity
            7.9

District 9
            7.9


SQL>
SQL> --EXPLAIN statement to explain the query plan
SQL> EXPLAIN PLAN
  2  FOR
  3  SELECT tb.PRIMARYTITLE AS "Movie Title", tr.AVERAGERATING AS "Average Rating"
  4  FROM imdb00.TITLE_BASICS tb
  5  JOIN imdb00.TITLE_RATINGS tr ON tb.TCONST = tr.TCONST
  6  WHERE tb.TITLETYPE = 'movie'
  7  AND tb.STARTYEAR BETWEEN '2001' AND '2010'
  8  AND tr.NUMVOTES >= 150000
  9  AND tb.GENRES LIKE '%Action%'
 10  AND tb.GENRES LIKE '%Thriller%'
 11  ORDER BY tr.AVERAGERATING DESC
 12  FETCH FIRST 5 ROWS ONLY;
```

```
 12  FETCH FIRST 5 ROWS ONLY;

Explained.

SQL>
SQL> --Display the query plan
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 2653010624

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |     5 |  5205 |  3846   (1)| 00:00:01 |
|*  1 |  VIEW                         |               |     5 |  5205 |  3846   (1)| 00:00:01 |
|*  2 |   WINDOW SORT PUSHED RANK     |               |   276 | 31740 |  3846   (1)| 00:00:01 |
|   3 |    NESTED LOOPS               |               |   276 | 31740 |  3845   (1)| 00:00:01 |
|   4 |     NESTED LOOPS              |               |  1380 | 31740 |  3845   (1)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL        | TITLE_RATINGS |  1380 | 23460 |  1084   (2)| 00:00:01 |
|*  6 |      INDEX UNIQUE SCAN        | SYS_C00547784 |     1 |       |     1   (0)| 00:00:01 |
|*  7 |     TABLE ACCESS BY INDEX ROWID| TITLE_BASICS  |     1 |    98 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("from$_subquery$_004"."rowlimit_$$_rownumber"<=5)
   2 - filter(ROW_NUMBER() OVER ( ORDER BY INTERNAL_FUNCTION("TR"."AVERAGERATING") DESC
              )<=5)
   5 - filter("TR"."NUMVOTES">=150000)
   6 - access("TB"."TCONST"="TR"."TCONST")
   7 - filter("TB"."TITLETYPE"=U'movie' AND "TB"."STARTYEAR"<='2010' AND "TB"."GENRES"
              LIKE U'%Action%' AND "TB"."GENRES" LIKE U'%Thriller%' AND "TB"."STARTYEAR">='2001' AND
              "TB"."GENRES" IS NOT NULL AND "TB"."GENRES" IS NOT NULL)

26 rows selected.

SQL>
SQL> SPOOL OFF
```

- EXPLAIN statement output:

- The query started by accessing and filtering rows from the TITLE_BASICS table by certain criterias (the conditions are specified in line 7 of the Predicate Information section)
- For each row from the TITLE_BASICS table, the value of TCONST is used to access rows from the TITLE_RATINGS table using an index unique scan on SYS_C00547784 (both tables share TCONST so they were joined through that condition)

- Each row in the TITLE_RATINGS table is filtered based on the condition that NUMVOTES is greater or equal to 150000. Every row that satisfies the condition will then be accessed.
- The program performs a nested loop join between the rows retrieved from the two tables, where the join matches rows based on the join condition.
- Another nested loop join is performed based on the result of the previous join and the WHERE clause conditions.
- The results are then sorted using the WINDOW SORT PUSHED RANK operation, where they will be ordered by AVERAGERATING in a descending order (specified in line 2 of the Predicate Information section)
- A view is created based on the sorted result where it will only show the top 5 rows based on AVERAGERATING.

III. Assumptions

- Due to some (254) entries not having a value for their year, we had to modify the code to include those cases.

IV. File Descriptions

- query.sql: contains the queries, English queries, and commands to run on Oracle sqlplus that would give the output specified in SPOOL
- top_movies_output.txt: contains the formatted output of the queries ran in query.sql
- Output folder: contains all 9 outputs for the M/R code
- Project3.java: This contains the M/R code
- Project3.xlsx: This contains the datasheets for our M/R analysis
- Javadoc folder: javadoc output from Project3.java

V. Division of Labor

Zaid was responsible for Task 1, he was the one that set up Hadoop and helped write and run the M/R code. Zaid worked for 14 hours.

Nam was responsible for Task 2, he helped build the SQL statement and the Explain statement output, he worked for 10 hours.

Thanh was responsible for working on Task 1 and Task 2, he basically did cooperative coding with Nam and Zaid. He worked for 12 hours.