Database Systems
Project Ia – index Choices for SQL Queries

Instructor: Sharma Chakravarthy
Description of the IMDb Database and Questions

Made available on: **8/22/2025**
Project Ia due date: **9/18/2025 (11:55 PM)**
Submit on: **Canvas (1 zipped folder containing a file with English questions, SQL queries, and answers obtained)**

Weight: **8% of total**
Total Points: **100**

We have created a large database and populated it with Millions of rows of International Movies and TV episodes information. It is known as the IMDb database by the community (publicly available data set, but not as a relational DBMS) and used by researchers in databases and other fields. The details of the tables are given below. This has all movie and TV episode information from the beginning (1925) until 2020 or so for both US and other countries. You can query this database for looking up certain information of interest to you, finding aggregate and statistical information that you are interested in, and OLAP analysis queries (using CUBE ROLL UP) as well to the extent possible using Oracle SQL.

The IMDb database includes the following information: movie title, year produced, genres a movie belongs to, actors, writers, directors, runtime, adult or non-adult classification, reviews in terms of votes on the movie, average rating, region, language etc. Similarly for TV series.

The purpose of setting up this database and this project is to provide you with an understanding of the differences between a toy DBMS (having 10 to 100 rows) and a reasonably large real-world DBMS, in terms of the kinds of queries you can ask, the response time, and appreciate the technology behind a DBMS (query optimization, concurrency control, simple relational abstraction, easy-to-use, non-procedural query language etc.)

Please make sure you do not write queries that produce large amounts of output, such as list all rows of a table containing Millions of rows. You need to think in terms of aggregate queries so you can extract the sliver of information that you are interested in. Also, as many fields contain strings with some delimiter, you need to be

careful about string comparison and use the LIKE operator of SQL with % and _ for picking out the correct string of interest (can also use string matching). Converting strings into either lowercase (LOWER(attribute) or UPPER(attribute)or upper case for comparison is also a good idea. For example, genres can be matched using LIKE 'Comedy' or LIKE 'Drama'. Note the first letter is capitalized. The other genres present are Horror, Short, Thriller, Sci-Fi, Music, Musical, to name a few. For years, use LIKE '200%' to get values in the range 2000 to 2009. Similarly for others. Some populated field values have a \N as their value. So, it is useful to have NOT LIKE '\N' to exclude those. The same goes for names of actors or directors and using LIKE with % and _ is a better idea.

Use select *
From your_table_name
Fetch first n rows only; //n is an integer
You can also use last or rownum (as in where rownum <= 10; etc.)

Please lookup Oracle SQL documentation

I. The following tables are populated in the database:

We have setup 2 identical databases imdb00 and imdb01. Hence you need to prefix table names with the database name to query ten (e.g., imdb00.title_basics or imdb01.title_basics). We will be using the database imdb01 as we will setup indexes in that database. Imdb00 will have no indexes, and you can run the same queries on both databases by changing the prefix. Imdb00 will act as the baseline for comparison.

Total Number of Tables: 9
Maximum Number of rows in a table: 27 million rows
Maximum Number of attributes in a table: 9; they are self-explanatory.

1. TITLE_BASICS table

SQL> describe TITLE_BASICS

Name	Null?	Type
TCONST	NOT NULL	VARCHAR2(10)
TITLETYPE		NVARCHAR2(500)
PRIMARYTITLE		NVARCHAR2(950)
ORIGINALTITLE		NVARCHAR2(950)
ISADULT		NUMBER(1)

```
STARTYEAR                      NUMBER(4)
ENDYEAR                        NUMBER(4)
RUNTIMEMINUTES                 NUMBER(10)
GENRES                         NVARCHAR2(350)

SQL> select count(*) from TITLE_BASICS;

  COUNT(*)
----------
   Total number of row: 4809386 ( 4.8 million rows)
```

****

2. TITLE_CREW_WRITER table

```
SQL> describe TITLE_CREW_WRITER
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 TCONST                         NOT NULL     VARCHAR2(10)
 WRITERS                        NOT NULL     VARCHAR2(10)

 SQL> select count(*) from TITLE_CREW_WRITER;

  COUNT(*)
----------
  Total number of rows: 5297540 ( 5.2 million rows)
```

3. TITLE_CREW_DIR table

```
SQL> describe TITLE_CREW_DIR
 Name                                      Null?          Type
 ----------------------------------------- -------- ----------------------------

 TCONST                         NOT NULL     VARCHAR2(10)
 DIRECTORS                      NOT NULL     VARCHAR2(10)

 SQL> select count(*) from TITLE_CREW_DIR;

  COUNT(*)
----------
  Total number of rows: 3408484 (3.4 million rows)
```

****

4. TITLE_EPISODE table

SQL> describe TITLE_EPISODE

Name	Null?	Type
TCONST	NOT NULL	VARCHAR2(10)
PARENTTCONST	NOT NULL	VARCHAR2(10)
SEASONNUMBER		NUMBER(9)
EPISODENUMBER		NUMBER(9)

SQL> select count(*) from TITLE_EPISODE;

COUNT(*)

Total number of rows:3206322 (3.2 million rows)

****

5. TITLE_PRINCIPALS table

SQL> describe TITLE_PRINCIPALS

Name	Null?	Type
TCONST	NOT NULL	VARCHAR2(10)
ORDERING		NUMBER(4)
NCONST	NOT NULL	VARCHAR2(10)
CATEGORY		VARCHAR2(550)
JOB		VARCHAR2(500)
CHARACTERS		NVARCHAR2(800)

SQL> select count(*) from TITLE_PRINCIPALS;

COUNT(*)

Total number of row: 27054380 (27 million rows)

****

6. TITLE_RATINGS tables

SQL> describe TITLE_RATINGS

Name	Null?	Type
TCONST	NOT NULL	VARCHAR2(10)
AVERAGERATING	NOT NULL	NUMBER(5,2)
NUMVOTES	NOT NULL	NUMBER(15)

```
SQL> select count(*) from TITLE_RATINGS;

  COUNT(*)
----------
Total number of rows:  805011 ( 0.8 million rows)
```

****

7. TITLE_AKAS table

```
SQL> describe TITLE_AKAS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 TITLEID                                   NOT NULL VARCHAR2(10)
 ORDERING                                           NUMBER(10)
 TITLE                                              NVARCHAR2(950)
 REGION                                             NVARCHAR2(550)
 LANGUAGE                                           NVARCHAR2(550)
 TYPES                                              NVARCHAR2(550)
 ATTRIBUTES                                         NVARCHAR2(500)
 ISORIGINALTITLE                                    NUMBER(2)

 SQL> select count(*) from TITLE_AKAS;

  COUNT(*)
----------
   3563547 (3.5 million rows)
```

****

8. NAME_TITLE_MAPPING table

```
SQL> describe NAME_TITLE_MAPPING
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 NCONST                                    NOT NULL VARCHAR2(10)
 TCONST                                    NOT NULL VARCHAR2(10)

SQL> select count(*) from NAME_TITLE_MAPPING;

  COUNT(*)
----------
  14144524 (14 million rows)
```

****

9. NAME_BASICS table

```
SQL> describe NAME_BASICS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 NCONST                                    NOT NULL VARCHAR2(10)
 PRIMARYNAME                               NOT NULL NVARCHAR2(950)
 BIRTHYEAR                                          NUMBER(4)
 DEATHYEAR                                          NUMBER(4)
 PRIMARYPROFESSION                                  VARCHAR2(900)

SQL> select count(*) from NAME_BASICS;

  COUNT(*)
----------
   8424762 (8.4 million rows)
```

II. In this project, you are given 4 English queries on the IMDb database described above. What you need to do is:

a) Translate these English queries into SQL and make sure they are correct in terms of the answers you get

b) For EACH query, execute it on the imdb00 database where there are no indexes. Record the response time

c) For Each query choose 4 different indexes (each can be multiple indexes separate for each table) from among the indexes available to the tables and run it on the imdb01 database. Remember, more than one index may be available on any table.

d) Record the response time for each index chosen for each query and create a table of response time plus the response time from imdb00 database which has no index as shown below.

e) Do an analysis of the response time based on what you have learnt in the course on the use of indexing and file structure.

f) The table used for analysis should like

Query/index	Imdb00	Index 1	Index 2	Index 3	Index 4
Q1					
Q2					

Q3					
Q4					

If you are using different indexes for different queries, make a separate table for each query. The column header should indicate the index type used along with its description clearly.

Choice of indexes and analysis is the most important aspect of this project. It should clearly indicate/justify the response time you see and your explanation of why that is the case.

*Hint: use of **with** clause makes it easier to write these queries. It is very similar to the subqueries in the FROM clause.*

1. Retrieve by the years (for a 10 year period, such as 2000 to 2009), the count of movies produced in a genre (choose one from the genre list, Comedy, Drama, Horror, Sci-Fi, …) whose rating is greater than the average rating of movies in that genre for that year.

Start Year	Genres	Above_avg
2000	Comedy	134
2001	Comedy	136
2002	Comedy	123
2003	Comedy	153
2004	Comedy	176
2005	Comedy	220
2006	Comedy	224
2007	Comedy	207
2008	Comedy	257
2009	Comedy	267

2. For the above years, retrieve the total number of movies produced in each genre and the average rating

Start Year	Genres	Movies_produced
2000	Comedy	325
2001	Comedy	312
2002	Comedy	314
2003	Comedy	398
2004	Comedy	443
2005	Comedy	539

2006	Comedy	567
2007	Comedy	562
2008	Comedy	689
2009	Comedy	751

3. Choose you favorite actor and an actress and for each fine the number of movies done by them for each year in the entire database

Here is the result for Tom Hanks

```
NB.PRIMARYNAME||','||TB.STARTYEAR||','||COUNT(*)
--------------------------------------------------------------------------------
Tom Hanks,2016,4
Tom Hanks,2004,3
Tom Hanks,1986,3
Tom Hanks,1989,2
Tom Hanks,2013,2
Tom Hanks,2011,2
Tom Hanks,2002,2
Tom Hanks,1998,2
Tom Hanks,1988,2
Tom Hanks,2017,2
Tom Hanks,1985,2
Tom Hanks,1995,2
Tom Hanks,1990,2
Tom Hanks,1999,2
Tom Hanks,1984,2
Tom Hanks,1993,2
Tom Hanks,1992,1
Tom Hanks,1987,1
Tom Hanks,2000,1
Tom Hanks,2006,1
Tom Hanks,1996,1
Tom Hanks,1994,1
Tom Hanks,2010,1
Tom Hanks,2019,1
Tom Hanks,2012,1
Tom Hanks,2009,1
Tom Hanks,2007,1
Tom Hanks,2015,1
Tom Hanks,2008,1

29 rows selected.
```

b) Retrieve the average ratings of the movies by genre for a 5-year period (e.g., 2010 to 2014) for each genre, Comedy, Drama, Horror, Sci-Fi, ….
Should have 24 rows of output, for 5 years and 4 genres.

start year	GENRES	YEARLY_AVG
2010	Drama	6.35474326
2010	Horror	4.93798077
2010	Sci-Fi	5.27391304
2010	Thriller	5.6872
2011	Drama	6.40383653
2011	Horror	4.95151515
2011	Sci-Fi	5.17
2011	Thriller	5.78739496
2012	Drama	6.47727987
2012	Horror	4.98027211
2012	Sci-Fi	4.99583333
2012	Thriller	5.48928571
2013	Drama	6.46666667
2013	Horror	4.8939759
2013	Sci-Fi	5.46666667
2013	Thriller	5.61341463
2014	Drama	6.54582185
2014	Horror	4.94344828
2014	Sci-Fi	5.28
2014	Thriller	5.6625

III. Grading Scheme — 100 points

1. Correct SQL and output (4 queries) — 40 (10 each)
   A file containing English queries, its SQL version
   That can be submitted for execution
   A separate fie containing the above And result from execution on Oracle on Omega

2. A word fie containing analysis explaining the choice of Indexed for each query and analysis of response time — 40 (10 each)

3. SQL coding style and completeness of submission
   Each clause of SQL should be on a separate
   line and properly indented 20

Please pay attention to what you need to submit. All files should be named appropriately and included for submission.

All the above files/directories should be placed in a single zipped folder named as - 'proj1a_team_<no>_final'. Only one zipped folder per team should be uploaded using canvas. We will ONLY grade the Latest version submitted.

- **Y**our .sql file shoud be executable (without any modification by us) on Omega Oracle using the @filename command. So, please test it for that before submitting it. However, the source code files can be created and/or edited on any editor that produces an ASCII text file. As I mentioned in the class, an IDE is not necessary for this and subsequent projects. If you decide to use it, please learn it on your own and make sure your code compiles and executes with appropriate package information.

- You can submit your zip file at most 3 times. The latest one (based on timestamp) will be used for grading. So, be careful about what you turn in and when!

- Five days after the due date, the late submission upload link (separate from the on-time submission link) will be closed. Penalty for each day is 20% of the earned grade. No partial penalties per day!

Oracle Information

1. If you want to use a client (e.g., Oracle Development workbench, https://docs.oracle.com/cd/E99083_01/PDF/Open_Dev_Tools/03-Development_WorkBench_Getting_Started.pdf), you are welcome to use. However, you need to learn on your own. However, submission should be done as indicated in this project description.

2. Oracle provides extensive documentation on its web site. We are using Oracle 19c. You may want to use that for

Oracle SQL:
https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/

Oracle Hints:

Oracle SQL hints are instructions embedded within SQL statements to influence the Oracle optimizer's execution plan. They are placed within a comment block immediately following
the SELECT, UPDATE, MERGE, INSERT, or DELETE keyword, and are distinguished by a plus sign (+) immediately after the comment delimiter.

SELECT /*+ *hint [hint ...]* */ column_list FROM table_name;

documentation: https://docs.oracle.com/en/middleware/bi/analytics-server/datamodel-oas/use-hints-sql-statements.html#GUID-14FCF9E8-B592-4336-82CC-E16C24068D68

Purpose:
Hints override the optimizer's default processing plan, allowing developers to guide the optimizer towards a more efficient plan based on their knowledge of data characteristics or specific performance goals. This can be useful in scenarios where the optimizer might not choose the optimal plan automatically.

Types of Hints:
Oracle offers various types of hints to control different aspects of query execution:

- **Optimization Approaches and Goals:**
  - ALL_ROWS: Optimizes for best throughput (retrieving all rows).
  - FIRST_ROWS(n): Optimizes for best response time (retrieving the first n rows quickly).
- **Access Paths:**
  - FULL(table_alias): Forces a full table scan.
  - INDEX(table_alias index_name): Forces the use of a specific index.
- **Join Orders and Operations:**
  - ORDERED: Joins tables in the order they appear in the FROM clause.
  - USE_NL(table_alias): Forces a nested loop join.
  - USE_HASH(table_alias): Forces a hash join.
- **Parallel Execution:**
  - PARALLEL(table_alias degree_of_parallelism): Enables parallel execution for a table.
- **Query Transformations:**
  - STAR_TRANSFORMATION: Forces a star transformation for star schemas.

Considerations:

While hints can improve performance in specific cases, they should be used cautiously. They can make SQL statements less portable and require maintenance as data or database environments change. Oracle recommends using tools like SQL Tuning Advisor and SQL Plan Management for addressing performance issues, as these tools provide more dynamic and adaptable solutions compared to static hints.