INFS2200/INFS7903 ASSIGNMENT

Semester 2 2018

Marks:	100 marks (15%)
Due Date:	11.59pm 22-Oct-2018
What to Submit:	SQL script file in addition to a short PDF report
Where to Submit:	Electronic submission: Blackboard

The goal of this project is to gain practical experience in applying several database management concepts using the Oracle DBMS.

Your task is to first populate your database with appropriate data, then design, implement, and test the appropriate queries to perform the tasks explained in the next sections.

You must work on this project individually. The standard academic honesty rules apply. Plagiarism will be taken seriously and punished appropriately.

Roadmap: Section 1 describes the database schema for your project and it also provides instructions on downloading the script file needed to create and populate your database. Section 2 describes the tasks to be completed for this project. Finally, Section 3 provides you with all the necessary submission guidelines.

Enjoy your Project!

SECTION 1. THE MOVIES DATABASE

The Database: The MOVIES database (Figure 1) captures information regarding movies and the actors in these movies.. The database includes six tables: **actor**, **film**, **category**, **film_actor**, **film_category**, and **language**. **Actor** stores information about all actors in the industry. **Film** keeps track of film details. **Category** stores information about the different types of film categories. **Language** stores the different languages in which these movies are released. **Film_actor** and **film_category** keeps track of which actors have acted in which films, and which films are classified under which categories, respectively.

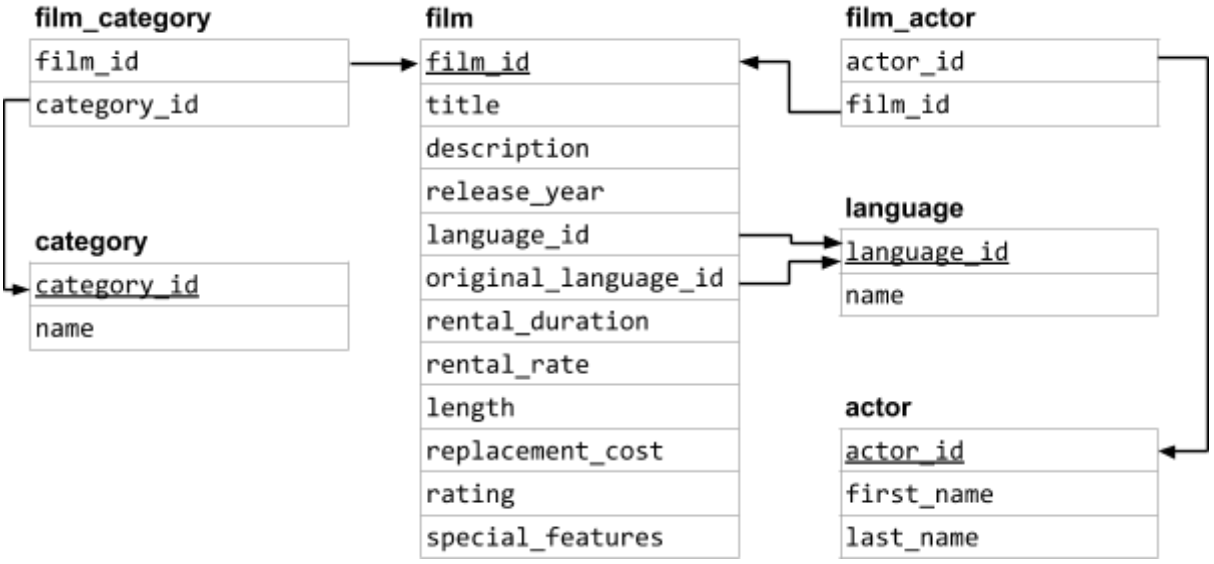

Figure 1 Database schema

The Script File: Please go to Blackboard and download the supplementary script file.

The Database Constraints: The following table lists all the constraints applied to the MOVIES database.

No	Constraint Name	Table.Column	Description
1	PK_ACTORID	actor.actor_id	actor_id is the primary key of **actor**
2	PK_CATEGORYID	category.category_id	category_id is the primary key of **category**
3	PK_FILMID	film.film_id	film_id is the primary key of **film**
4	PK_LANGUAGEID	language.language_id	language_id is the primary key of **language**
5	UN_DESCRIPTION	film.description	Film description values are unique
6	CK_FNAME	actor.first_name	Actor's first name must not be empty (not null)
7	CK_LNAME	actor.last_name	Actor's last name must not be empty (not null)
8	CK_TITLE	film.title	Film title must not be empty (not null)
9	CK_CATNAME	category.name	Category name must not be empty (not null)
10	CK_RENTALRATE	film.rental_rate	Film rental rate must not be empty (not null)
11	CK_RATING	film.rating	Rating type must be one of the following: 'G','PG','PG-13','R','NC-17'
12	CK_SPLFEATURES	film.special_features	Special Features type must be empty or one of the following: 'Trailers', 'Commentaries', 'Deleted Scenes', 'Behind the Scenes'
13	FK_LANGUAGEID	film.language_id and language.language_id	film.language_id refers to language.language_id
14	FK_ORLANGUAGEID	film.original_language_id and language.language_id	film.original_language_id refers to language.language_id
15	FK_ACTORID	film_actor.actor_id and actor.actor_id	film_actor.actor_id refers to actor.actor_id
16	CK_RELEASEYR	film.release_year	film.release_year is less than or equal to current year (Hardcode the current year 2018)

Table 1. Constraints

SECTION 2. ASSIGNMENT TASKS

Task 0 – *Database*

1. You need to execute the script file to create and populate your database before working on the following tasks. Wait until you see the message "`DONE! All data has been inserted.`" It should only take a minute. The script will also drop related tables.

Task 1 –*Constraints*

1. After running the script file, you will notice that only some of the constraints given in Table 1 were created. Write the necessary SQL statements to find out which constraints have been created on the tables. Some table names may need to be in capitals eg. 'FILM' instead of 'film'.
2. Write the necessary SQL statements to create all the missing constraints.

Task 2 –*Triggers*

1. Write a trigger, named **`BI_FILM_ID`** that automatically populates the film_id when a new film is added. The sequence, named **`FILM_ID_SEQ`**, should start from 22,000 and increment by 2.
2. Write an SQL trigger, that should be named **`BI_FILM_LANG`**, to append text to the description of every new film inserted into the database. It is based on the language (`language_id`) and the original language (`original_language_id`) of the film.
   The format of the text you append should be (replacing tokens):
   `Originally in <original language>. Re-released in <Language>.`
   Original language and language should be the name of the language from the language table.

For example:
If the following query was run:

```
INSERT INTO FILM (title, description, language_id,
original_language_id ) VALUES
('B Movie', 'Movie about wasps.', 1, 2);
```

It should produce the following when the following select statement is run (based on the script file provided to you and assuming B Movie's id is 9999999):

```
SQL> SELECT description FROM FILM WHERE film_id = 9999999;
description
------------------------------------------------------------
Movie about wasps.Originally in Italian. Re-released in
English.
```

Notes for Question 2.2:

- This trigger should only fire upon new rows inserted.
    - You do not need to update existing rows.
- The text should be added to the end of the additional description.
    - You must preserve existing text in additional description.
    - You do not need to handle cases where the resulting text after the trigger exceeds the description length. Let the trigger fail.
- If either language or original language is null, then the trigger should not do anything.
- Your trigger should handle other film languages beyond those provided to you
    - For example if the language ‘SQL’ was added to the language table, then the trigger should be able to handle a movie in ‘SQL’.
- Description must match expected output **exactly** in order to receive marks. Based on the same query in the example above, the following description below **is incorrect** as there is a space between the original movie description and the appended text.

```
Movie about wasps. Originally in English.
Re-released in English.
```

- Do not append a space in the text you are adding(do not add a space before).
    - Note in the example provided there is no space after the first full stop
- Do not add any line breaks (new lines).
- Do not alter the name of the languages in any way. Leave as is.
    - Do not change capitalisation.

Task 3 – *Views*

1. Write a SQL statement to find the 'Comedy' films with the longest running time. Your query should output the titles and lengths of the films.

Notes for Question 3.1:

- 'Running time' refers to the length of the film, and 'Comedy' is a film category.

2. Write a SQL statement to create a (virtual) view called **MAX_COMEDY_ACTORS** that contains all the actors that have acted in the films that you obtained in Task 3.1. The view should include the columns actor id, first name and last name. Note: Actors may act in multiple films, but should only appear once in the view.
3. Write a SQL statement to create a (virtual) view called **V_COMEDY_ACTORS_2008** that lists the ids, first names and last names of any actors that starred in a Comedy film released in the year 2008.
4. Write a SQL statement to create a materialized view called **MV_COMEDY_ACTORS_2008** that lists the same information as in Task 3.3.

Notes for Question 3.3 & 3.4:

- Similar to Task 3.2, there should be no duplicate rows in the query output.

5. Execute the following two SQL statements and report their query execution time. Is there any difference between the reported execution times of Q1 and Q2?
   Please give the reasons for that.

   Q1: `SELECT * FROM V_COMEDY_ACTORS_2008;`
   Q2: `SELECT * FROM MV_COMEDY_ACTORS_2008;`

Note: For any task mentioning execution time, please run the queries on a computer with a HDD, rather than an SSD, so that any timing differences are noticeable. All lab computers have HDD's and are appropriate for this.

Task 4 – *Indexes*

1. Construct a query to select the **first 200 film titles** (in ascending alphabetical order) where the film takes place in a 'Boat'. You may assume that the film location is contained within the film table and attribute, description. Additionally, it is always after the first occurrence of the word 'in' and finishes at the end of the sentence. For example in the following film description:

   "*A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies*"

The location is “The Canadian Rockies”.

Notes for Question 4.1:

- ‘Boat’ can be any boat, U-Boat, JET Boat etc, as long as the location includes the word ‘Boat’.
- You do not need to consider the film description output from the trigger you created for question 2.2.

- You should avoid using LIKE and instead use string manipulation functions such as **INSTR** and **SUBSTR**

- When selecting the top 200 film titles in alphabetical order, the exact requirement is for your query to **select all occurrences** of films that take place on a boat, **order those films in ascending alphabetical order** and **return the top 200.** As such, your query must use the **ORDER BY** clause.

2. Create a function-based index called **`IDX_SEARCH_LOCATION`** that could potentially increase the speed of the query written in 4.1. Write a SQL statement to create an index that best fits that task and explain your choice.

3. Report the execution time of the query statement you wrote in Task 4.1 before and after creating the index in Task 4.2. Explain any difference. If there is no difference, explain why.

Task 5 – *Execution Plan*

1. Write a SQL statement to list all information for the films with a film_id value that is greater than 1000. Report the **rule-based** execution plan chosen by the Oracle optimizer for executing your query. Explain the query processing steps taking place in this plan.

2. Re-execute the query you wrote in Task 5.1 and report the **cost-based** execution plan chosen by the Oracle optimizer for executing your query. Explain the query processing steps taking place in this plan. Comment on the efficiency of this plan.

3. Now write a SQL statement to list the information for the film with a film_id of 1000. Report the **cost-based** execution plan chosen by the Oracle optimizer for executing your query.

4. In your opinion, what are the main differences between the plans you obtained in Task 5.1, and 5.2?

5. In your opinion, what are the main differences between the plans you obtained in Task 5.2 and 5.3?

6. Write SQL statements to gather the following statistics from the database:
    a. The height of the system generated B+ tree index for FILM_ID
    b. The number of leaf blocks in the system generated B+ tree index for FILM_ID
    c. The number of block accesses needed for a direct full table scan of the FILM table.

You may find the following documentation from Oracle to answer Task 5 helpful:

https://docs.oracle.com/cd/B28359_01/server.111/b28320/statviews_5119.htm#REFRN29025

https://docs.oracle.com/cd/B19306_01/server.102/b14237/statviews_4473.htm#REFRN26286

SECTION 3. MARKING SCHEME

Tasks	Marks
0	5
1.1	5
1.2	8
2.1	4
2.2	12
3.1	5
3.2	7
3.3	6
3.4	2
3.5	5
4.1	10
4.2	5
4.3	3
5.1	3
5.2	2
5.3	2
5.4	4
5.5	4
5.6	3
Presentation & Readability	5
Total	**100**

SECTION 4. DELIVERABLES

The project is due **11:59PM, 22nd Oct. 2018**. No late submission is allowed.

You are required to turn in two files (use studentID to name your files):

1. studentID.pdf: (rename studentID)

A report answering all the questions in Section 2 including all the necessary SQL statements and their outputs (please be sensible when including query output, any output close to the size of a page can be shown by just including the first 10 lines and the last 10 lines – reports including pages of query output will lose presentation marks).

2. studentID.sql: (rename studentID)

A script file that includes all your SQL statements.

Your **report** file should include the following content:

- Answers to all the questions in Section 2.
- If you are asked to write SQL statements, you need to include those statements in your report.
- When you execute an SQL statement, if Oracle produces any output (e.g. query results, query execution time, query plan, etc), you need to include the output as well.

Your **script** file is in plain text format. You must make sure that your script file can be executed on the ITEE lab computers using the "@" command. The same SQL statements in your script file should also be copied and pasted into your report file (as explained above). Even though the script file does not introduce any new information in comparison to the report file, it is intended to help the lecturer/tutors to quickly check the correctness of your SQL statements before checking the details in your report file.

Enjoy your project!