Homework#4

ACS575 Database Systems, Spring 2024

Due Date: April 5

- Submit your homework as a single compressed file, labeled with YourLastName_YourFirstName_ACS575_HW4.zip.
- Your submission should be organized into four parts: Part I, Part II, Part III and Part IV. Please ensure each section is clearly labeled within your file
- Within each section, clearly number your responses to correspond with the question numbers, for example, Q1, Q2(1), and so forth.

- While this homework is designed with Oracle RDBMS in mind, you are free to use other RDBMS platforms, provided they offer similar functionality. If you opt for a different RDBMS, please clearly state which one you've selected and describe any challenges or limitations you encountered that pertain to the homework exercises.

Part I. Indexes

Preparation Instructions: This section uses the movie database from Homework#3. Re-execute the script (Movie_Actor_Casting_Tables.sql) in your database to ensure the movie, actor, and casting tables are set up correctly for this exercise.

Q1. **Creating Indexes on Foreign Keys:** (1) Construct SQL commands to generate indexes for each foreign key in the casting table. Assign each index a meaningful name. (2) Show the executions of these commands.

Q2. **Creating Indexes on Non-key Columns:** (1) Construct SQL commands to create indexes on actor's name, movie's title, and movie's votes. (2) Show the executions of these commands.

Q3. **Retrieving Index Information:** (1) Develop a SQL query to gather detailed information about the indexed on the actor, casting and movie tables. Utilize both USER_INDEXES and USER_IND_COLUMNS dictionary views to include details such as the index name, index type, associated table name, uniqueness status, indexed column name, and column position within the index. Order the query output by table name, followed by index name and column position within the index. (2) Present the results of your query.

For guidance on using the USER_ INDEXES system catalog view, consult the Oracle documentation available at https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/ALL_INDEXES.html#GUID-E39825BA-70AC-45D8-AF30-C7FF561373B6

For the USER_IND_COLUMNS view, refer to https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/ALL_IND_COLUMNS.html#GUID-2A73036B-502E-4D93-9C98-DEDE234872AA

Q4. **Analyzing Index Types**: Based on the index information retrieved in Q3. **(1)** Identify which indexes serve as primary indexes and which ones are secondary. **(2)** Discuss the fundamental difference between primary and secondary indexes, particularly in terms of their roles in physical file organization.

Q5. **Identification and Application of Clustered Indexes:** Given the schemas for 'action' and 'move' tables, discuss whether these tables are ideal candidates for clustered indexes.

Q6. In the context of Oracle RDBMS, explore how data is physically organized and retrieved for tables with primary key constraints, specifically focusing on the 'actor' and 'move' tables, and compare this with SQL Server's explicit clustered index definition. Detail the differences in terminology and concept between the two RDBMS regarding index organization and data storage.

Part II. Query Processing and Optimizer

This section focuses on analyzing how the query optimizer processes SQL statements and the impact of indexes on query performance.

Preparation Instructions: Before proceeding, ensure the database management system has up-to-date statistical information on each table by executing:

```
ANALYZE TABLE movie COMPUTE STATISTICS;
ANALYZE TABLE actor COMPUTE STATISTICS;
ANALYZE TABLE casting COMPUTE STATISTICS;
```

In these questions, you'll delve into query execution plans selected by the Oracle query optimizer. For Oracle SQL Developer users, the "Explain Plan" feature simplifies this process. Focus on the "SELECT STATEMENT" part of the Explain Plan output and omit the "Other XML" section.

Q1. (1) Execute SQL queries S1 and S2 and provide the query plans for each:

- S1: SELECT * FROM movie WHERE votes BETWEEN 0 AND 1000;
- S2: SELECT * FROM movie WHERE votes BETWEEN 0 AND 50000;

(2) Given that S1 and S2 are similar in structure, provide an analysis of why their execution plans may differ. Explain why the two query plans are different.

Q2. (1) Execute the SQL queries S3 and S4 and provide the query plans for each.

- S3: SELECT votes FROM movie WHERE votes < 1000;
- S4: SELECT * FROM movie WHERE votes < 1000;

(2) Discuss the difference between the query plans of S3 and S4, specifically identifying which query benefits from an index-only plan and why.

Q3. (1) Execute the SQL queries Q5, Q6 and Q7 and provide the query plan for each.

- S5: SELECT name FROM actor WHERE name like 'W%';
- S6: SELECT name FROM actor WHERE substr(name, 1, 1) = 'W';
- S7: SELECT name FROM actor WHERE name LIKE '%w';

(2) S5 and S6 yield identical results; however, their query plans differ. Explain the reasons behind the differing query plans.

(3) Delve into why the query plan for S7 does not leverage an available index, explaining the possible rational behind this decision.

Q4. (1) Execute the SQL query S8 and present its query plan.

- S8: SELECT a.name FROM actor a, movie m, casting c
  WHERE m.title = 'Star Wars' and m.id=c.movieid and a.id=c.actorid

(2) Identify the join algorithms used in this query and provide concise explanations of each algorithm mentioned, highlighting their use cases and operational mechanism.

Q5. (1) Execute the SQL queries Q9 and Q10 and present the query plan for each.

- S9: SELECT m.title FROM movie m WHERE m.title= 'Scrooge'
- S10: SELECT   /*+ FULL(m)*/ m.title FROM movie m WHERE m.title= 'Scrooge'

(2) Despite S9 and S10 generating the same result set, their query plans are different. Analyze and explain the differences between these plans, focusing on the underlying factors that contribute this difference.

Part III. Database Privileges and Authorization

Preparation Instructions:

- Ensure you have access to two distinct database accounts (for example, C##USER57501 and C##USER57501_2).
- For students using Oracle in Diamond.pfw.edu, a second account should be provided. If using a different Oracle instance or a different RDBMS, you may need to create your second account manually.
- Open two database sessions; one for each account.
- Download and review the '**emp.sql**' script file provided.
- <u>For each question, execute the given instructions, providing SQL statements where necessary, and include both the execution results and any descriptive answers required.</u>

Q1. Table Creation and Initial Query: In the session linked to your first account, run the **emp.sql** script to create the EMP table. Then, perform a query on this EMP table.

Q2. Cross-Account Table Access: Using the session for your second account, attempt to query the EMP table from your first account (e.g., SELECT * from C## USER USER57501.EMP). Document either the result or the error message received, explaining the latter if applicable.

Q3. Privilege Granting: From your first account, grant SELECT (query) privileges on the EMP table to your second account.

Q4. Verification of Access Post-Privilege Grant: Re-test the accessibility of the EMP table from your second account by querying the table again. Record the result or any error message encountered.

Q5. Insert Operation by Second Account: In your second account, attempt to insert a new record to the EMP table (to which you've been granted access). Provide the outcome or any error encountered, with an explanation if necessary.

Q6. Synonym Creation: Within your second account, create a synonym for the EMP table located in your first account. Use this template:

```
CREATE SYNONYM EMP FOR _____.EMP;
```

Q7. Query Using Synonym: Post-synonym creation, use your second account to query the EMP table via the established synonym.

Q8. Revocation of Privileges: From your first account, revoke the previously granted query privileges on the EMP table from your second account. .

Q9. Final Access Test: Following the revocation, use your second account to attempt a query on the EMPT table again. Report either the successful result or the error message, providing an explanation for the latter.

Part IV. Transaction Management

Understanding transaction isolation levels is crucial for ensuring data integrity and consistency within a database system. The ANSI SQL standard outlines four primary transaction isolation levels: Read Uncommitted, Read Committed, Repeated Read, and Serializable. Oracle's default isolation level is Read Committed, wherein each query within a transaction is guaranteed to see only the data that was committed before the query started, preventing it from reading any uncommitted or "dirty" data.

This exercise explores a common concurrency issue known as a "lost update," which can occur under certain isolation levels. Follow the steps and answer the questions below.

Preparation Instructions:

- **In your primary account,** execute the '**emp.sql**' script to recreate the EMP table.
- Grant all privileges on the EMP table to PUBLIC, allowing other users unrestricted access.
- **In your secondary account,** create a synonym for the EMP table, naming it EMP. This step facilitates easier access to the table across accounts.

<u>For the following exercises, write SQL statements along with their execution results. For questions requiring conceptual answers, provide a clear and concise explanation</u>

Execution order	Using the primary account	Using the secondary account	Questions
1	`SELECT ename, sal FROM emp WHERE ename IN ('SMITH', 'ALLEN', 'PARK');`		**Q1.** Provide the execution result

2	`UPDATE emp SET sal = 3000 WHERE ename='SMITH';`
	
	`SELECT ename, sal FROM emp WHERE ename IN ('SMITH', 'ALLEN', 'PARK');`
			Q2. Provide the execution results

3	`INSERT INTO emp (empno, ename) VALUES (7777, 'PARK');`
	
	`SELECT ename, sal FROM emp WHERE ename IN ('SMITH', 'ALLEN', 'PARK');`
			Q3. Provide the execution result

4		`SELECT ename, sal FROM emp WHERE ename IN ('SMITH', 'ALLEN', 'PARK');`	**Q4.** Provide the execution result and compare it with the result from Q3.
5		`UPDATE emp SET sal = 1000 WHERE ename='SMITH';`	**Q5.** Can your second account execute this update?
6	`COMMIT;`		**Q6.** After commit in the first account session, what happens in the second account session?

7		`SELECT ename, sal FROM emp WHERE ename IN ('SMITH', 'ALLEN', 'PARK');`	**Q7.** Provide the execution result and compare it with the result from Q4.
8		`COMMIT;`	
9	`SELECT ename, sal FROM emp WHERE ename IN ('SMITH', 'ALLEN', 'PARK');`		**Q8.** Provide the execution result and compare it with Q3. What happens in SMITH's salary?