Overview

The purpose of this lab is to become familiar with flashback, writing standard operating procedures and the explain plan.

Contents

- ○ Part 1 - Flashback
- ○ Part 2 – Standard Operating Procedures
- ○ Part 3 – Explain Plan

Due Date:

The lab is due no later than Friday Nov 17th. You must **hand in** PDFs of your work to the dropbox on D2L.

Scoring:

Lab is worth **21** marks.

Part 1 – Flashback

Demo

1. Login into Oracle XE with the Lab07 account. In the Lotto649 table, run a query to change the nbrbonus to 1 from 13 for the drawnbr=1 row. Use ***flashback query*** to set it back to 13. *(3 Marks)*

Note: If you don't have the lotto649 table, pick another table in your schema to change a column in the first row.

```
UPDATE LOTTO649 SET NBRBONUS = 1

WHERE DRAWNBR = 1;

SELECT * FROM LOTTO649

WHERE DRAWNBR = 1;

UPDATE LOTTO649 SET NBRBONUS =

(SELECT NBRBONUS FROM LOTTO649

AS OF TIMESTAMP SYSDATE - 11/8

WHERE DRAWNBR = 1)

WHERE DRAWNBR = 1;
```

Demo

2. Login to your babbage account and using SQL Developer, create a new table called temp as a select * from dual. Drop the table and then use ***flashback table*** to recover the table. Purge your recycle bin and then try executing the flashback table command again. *(3 Marks)*

```
CREATE TABLE temp AS (SELECT * FROM dual);

DROP TABLE temp;

FLASHBACK TABLE temp

TO BEFORE DROP;
```

Part 2 – Standard Operating Procedures

Hand In

1. A common excuse for not writing SOPs is that most tasks are "simple enough."' Resetting a password is one of the suggested SOPs.

Describe briefly both points of view as to why a DBA may or may not need an SOP for this. *(4 Marks)*

Depending on how the system is set up a password reset can be forced after x amount of time which would remind / prompt the dba for a new password or reminder that it will expire. If this is not how the system is set up then an SOP would be an asset / reminder of the standards and requirements for said system.

Hand In

2. Write a SOP for resetting a password, following the suggested structure from the slides. Any sections that you feel are not necessary may be omitted. *(7 Marks)*

Note: this should not be more than one page.

SOP – Password Reset

1- Overview
    a. Passwords must be reset every 30 days
    b. Must be at least 8 characters long with at least one number and one capital letter
2- Approvals
    a. The system will allow the new password if requirements are met
3- Notification
    a. The system will notify the user if the password has been accepted or rejected
4- Backup

a. A user that has higher clearance in case account is locked or password is forgotten to release holds on the account

5- Execution
   a. System prompted reset
      i. Login with current password
         1. Prompted with Enter new password:
            a. Type new password hit enter

6- Verification
   a. To confirm log out and log back in with new password

Part 3 – Explain and the Plan Table

Generating and displaying the execution plan of a SQL statement is a common task for most DBAs, SQL developers and performance experts as it provides them information on the performance characteristics of a SQL statement. An execution plan shows the detailed steps necessary to execute a SQL statement. These steps are expressed as a set of database operators that consumes and produces rows. The order of the operators and their implementation is decided by the query optimizer using a combination of query transformations and physical optimization techniques.

The explain plan shows table access methods (FULL, INDEX), table join types (nested loops, hash join, merge join), and sorting.

Execution plans can look very confusing, but reading them is reasonably simple provided you follow three simple rules:

1. The first operation, or starting point, is the first leaf node, when reading from the top to the bottom. That is, the first element without an indented entry below it. You read from that point backwards.
2. Join operations always require two sets. The order you read the sets is top down, so the first set is the driving set and the second is the probed set. In the case of a nested loop, the first set is the outer loop. In the case of a hash join, the first set is used to build the hash table.
3. One join is performed at a time, so you only need to consider two sets and their join operation at any one time.

There are two different methods you can use to look at the execution plan of a SQL statement:

1. **EXPLAIN PLAN command** - This displays an execution plan for a SQL statement without actually executing the statement.
2. **V$SQL_PLAN** - A dictionary view that shows the execution plan for a SQL statement that has been compiled into a cursor in the cursor cache.

For more information on the Oracle Optimizer and the Explain Plan see http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-explain-the-explain-plan-052011-393674.pdf

□ 1. Run the following SQL in SQL Developer

```
explain plan for select * from lotto649 where drawdate = '1988-01-01';

select plan_table_output from
table(dbms_xplan.display('plan_table',null,'typical'));
```

□ 2. Create an index on the drawdate column of the lotto649 table.

CREATE INDEX drawdate_idx

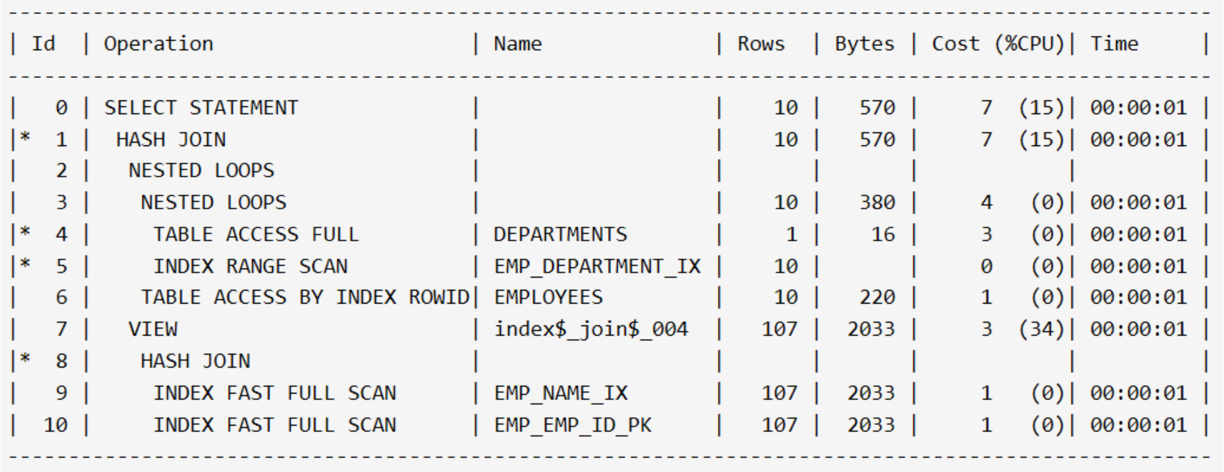
ON LOTTO649(drawdate);

□ 3. Run the two SQL statements from step 1 again.

Hand In 4. What is different in the results? *(2 Marks).*

There is a new operation called INDEX RANGE SCAN

Hand In 5. Given the following explain results, what is the order of operation of the query by **Id**? *(2 Marks).*



```
---------------------------------------------------------------------------------------------
| Id  | Operation                     | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                  |    10 |   570 |     7  (15)| 00:00:01 |
|*  1 |  HASH JOIN                    |                  |    10 |   570 |     7  (15)| 00:00:01 |
|   2 |   NESTED LOOPS                |                  |       |       |            |          |
|   3 |    NESTED LOOPS               |                  |    10 |   380 |     4   (0)| 00:00:01 |
|*  4 |     TABLE ACCESS FULL         | DEPARTMENTS      |     1 |    16 |     3   (0)| 00:00:01 |
|*  5 |     INDEX RANGE SCAN          | EMP_DEPARTMENT_IX |    10 |       |     0   (0)| 00:00:01 |
|   6 |    TABLE ACCESS BY INDEX ROWID| EMPLOYEES        |    10 |   220 |     1   (0)| 00:00:01 |
|   7 |   VIEW                        | index$_join$_004 |   107 |  2033 |     3  (34)| 00:00:01 |
|*  8 |    HASH JOIN                  |                  |       |       |            |          |
|   9 |     INDEX FAST FULL SCAN      | EMP_NAME_IX      |   107 |  2033 |     1   (0)| 00:00:01 |
|  10 |     INDEX FAST FULL SCAN      | EMP_EMP_ID_PK    |   107 |  2033 |     1   (0)| 00:00:01 |
---------------------------------------------------------------------------------------------
```

4 -> 5 -> 3 -> 6 -> 2 -> 9 -> 10 -> 8 -> 7 -> 1 -> 0