Bài thực hành số 5

VIRTUAL PRIVATE DATABASE (2)

❖ *Tóm tắt nội dung:*

- Quyền EXEMPT ACCESS POLICY
- Giám sát quyền EXEMPT ACCESS POLICY
- Xử lý các Exception về Policy Function
- Column Sensitive VPD

I. Quyền EXEMPT ACCESS POLICY

A. Lý thuyết

- Tuy RLS cung cấp một kỹ thuật bảo mật rất tốt, nhưng nó cũng dẫn đến một khó khăn khi thực hiện các tác vụ quản trị CSDL (ví dụ: tác vụ backup dữ liệu). Như đã biết, ngay cả các DBA và người chủ của các đối tượng đó cũng không thể tránh được các chính sách bảo mật. Nếu người chủ của một bảng nào đó hoặc một DBA thực hiện backup dữ liệu của bảng đó trong khi các chính sách bảo mật trên nó vẫn có tác dụng, rất có thể file backup sẽ không có dữ liệu nào hết. Vì lý do này (và một số lý do khác nữa), Oracle cung cấp quyền EXEMPT ACCESS POLICY. Người được cấp quyền này sẽ được miễn khỏi tất cả các function RLS. Người quản trị CSDL có nhiệm vụ thực hiện backup cần có quyền này để đảm bảo rằng tất cả các dữ liệu sẽ được backup lại.

B. Thực hành

- Để minh họa tác dụng của quyền EXEMPT ACCESS POLICY, trước hết ta tạo ra một chính sách không cho phép delete trên bảng EMP :

```
sec_mgr> CREATE OR REPLACE FUNCTION no_records (
            p_schema   IN  VARCHAR2 DEFAULT NULL,
            p_object   IN  VARCHAR2 DEFAULT NULL)
         RETURN VARCHAR2
         AS
```

```
            BEGIN
               RETURN '1=0';
            END;
            /
            Function created.

sec_mgr> BEGIN
               DBMS_RLS.add_policy
                  (object_schema     => 'SCOTT',
                  object_name        => 'EMP',
                  policy_name        => 'NOT_DELETE',
                  function_schema    => 'SEC_MGR',
                  policy_function    => 'No_Records',
                  statement_types    => 'DELETE');
            END;
            /
PL/SQL procedure successfully completed.
```

- Tạo một user mới và cấp role DBA cho user đó:

```
sec_mgr> GRANT dba TO backup_mgr IDENTIFIED BY backup;
```

- Do BACKUP_MGR bị ảnh hưởng bởi các policy function nên user này không xóa được record nào:

```
backup_mgr> DELETE FROM scott.emp;
0 rows deleted.
```

- Ta cấp cho BACKUP_MGR quyền EXEMPT ACCESS POLICY để user này không bị ảnh hưởng bởi các chính sách RLS nữa:

```
sec_mgr> GRANT EXEMPT ACCESS POLICY TO backup_mgr;
Grant succeeded.
```

- Thực hiện lại lệnh xóa ở trên với user BACKUP_MGR:

```
backup_mgr> DELETE FROM scott.emp;
14 rows deleted.
backup_mgr> ROLLBACK; -- undo lại hành động delete
```

II. Giám sát quyền EXEMPT ACCESS POLICY

A. Lý thuyết

- Do đây là quyền rất mạnh, không chỉ định trên cụ thể một schema hay object nào nên ta cần cẩn trọng trong việc quản lý xem ai được phép nắm giữ quyền này. Mặc định, những user có các quyền SYSDBA sẽ có quyền này (account SYS).
- Ta không thể ngăn cản các user được cấp quyền khỏi việc lạm dụng quyền được cấp. Ta chỉ có thể theo dõi xem họ làm gì với quyền được cấp đó. Auditing là một cách hiệu quả để đảm bảo quyền miễn trừ khỏi các chính sách RLS không bị lạm dụng. Auditing sẽ được trình bày kỹ hơn trong các bài lab về Auditing sau này. Trong phần này sẽ mặc định là sinh viên đã biết và hiểu về auditing. Sinh viên có thể đọc lại phần này sau khi học về Auditing.

B. Thực hành

- Ta có thể kiểm tra xem ai được cấp quyền EXEMPT ACCESS POLICY bằng câu lệnh sau:

```
sec_mgr> SELECT grantee FROM dba_sys_privs
         WHERE PRIVILEGE = 'EXEMPT ACCESS POLICY';
```

- Sử dụng câu lệnh sau để thiết lập audit quyền EXEMPT ACCESS POLICY:

```
sec_mgr> AUDIT EXEMPT ACCESS POLICY BY ACCESS;
Audit succeeded.
```

- Kiểm tra việc audit bằng cách thực hiện lại tác vụ delete trong account người được cấp quyền:

```
backup_mgr> DELETE FROM scott.emp;
14 rows deleted.
backup_mgr> ROLLBACK;
Rollback complete.
```

- Thực hiện đoạn PL/SQL sau để hiển thị các tác vụ bị audtit:

```
sec_mgr> SET SERVEROUTPUT ON;
sec_mgr>
```

```
BEGIN
    FOR rec IN
        (SELECT * FROM dba_audit_trail
        WHERE username = 'BACKUP_MGR'
        ORDER BY timestamp)
    LOOP
         DBMS_OUTPUT.put_line ('-------------------------');
         DBMS_OUTPUT.put_line ('Who: ' || rec.username);
         DBMS_OUTPUT.put_line ('What: ' || rec.action_name
                               || ' on ' || rec.owner
                               || '.'    || rec.obj_name);
         DBMS_OUTPUT.put_line ('When: '
                    || TO_CHAR(rec.timestamp,'MM/DD HH24:MI'));
         DBMS_OUTPUT.put_line ('Using: ' || rec.priv_used);
    END LOOP;
END;
/
-------------------------
Who:   BACKUP_MGR
What:  LOGON on
When:  04/04 14:22
Using: CREATE SESSION
-------------------------
Who:   BACKUP_MGR
What:  DELETE on SCOTT.EMP
When:  04/04 14:23
Using: DELETE ANY TABLE
-------------------------
Who:   BACKUP_MGR
What:  DELETE on SCOTT.EMP
When:  04/04 14:23
Using: EXEMPT ACCESS POLICY
-------------------------
Who:   BACKUP_MGR
What:  LOGOFF on
```

```
When:  04/04 14:27
Using:
PL/SQL procedure successfully completed.
```

- Audit trail hiển thị 4 record, trong đó 2 record ghi nhận việc log in/log out, 2 record còn lại liên quan đến lệnh delete mà user đã thực hiện. Bởi vì BACKUP_MGR đã sử dụng 2 quyền khi thực hiện lệnh DELETE nên có tới 2 record được ghi nhận cho cùng 1 hành động. Quyền thứ nhất là quyền DELETE ANY TABLE cho phép delete trên tất cả các bảng. Quyền thứ hai là quyền EXEMPT ACCESS POLICY cho phép không bị ảnh hưởng bởi chính sách bảo mật được áp đặt cho bảng EMP. Ta cũng thấy rằng mặc dù đã dùng lệnh ROLLBACK để undo lại hành động delete, nhưng hành động delete vẫn được ghi nhận đầy đủ.

III. Xử lý lỗi cho Policy Function

A. Lý thuyết

- Nhìn chung có 2 loại lỗi (error) thường gặp có thể khiến cho một chính sách RLS không thực hiện được:
  - ✓ Policy function không hợp lệ cho nên nó không được recompile và thực thi. Ví dụ, lỗi này sẽ xảy ra khi policy truy vấn đến một table không tồn tại. Lỗi về chính sách cũng có thể xảy ra nếu policy function không tồn tại (việc này thường do policy function đã bị drop hoặc nó đã được đăng ký không đúng trong thủ tục ADD_POLICY).
  - ✓ Chuỗi trả về của policy function khi được thêm vào câu lệnh SQL truy vấn trên đối tượng được bảo vệ gây ra lỗi câu lệnh SQL không hợp lệ. Có rất nhiều lý do khiến cho việc này xảy ra.
- Phần này sẽ trình bày cách dùng kỹ thuật xử lý ngoại lệ của một hàm để giữ được tính trong suốt của các chính sách RLS trong trường hợp loại lỗi thứ nhất xảy ra.

B. Thực hành

- Để minh họa cho lỗi của một policy function, trước tiên ta tạo một bảng sẽ được truy xuất bởi policy function:

```
sec_mgr> CREATE TABLE test(id int);
Table created.
sec_mgr> INSERT INTO test VALUES(1);
1 row created.
sec_mgr> CREATE PUBLIC SYNONYM testTable FOR test;
Synonym created.
```

- Tạo ra policy function có lệnh truy xuất đến bảng TEST:

```
sec_mgr> CREATE OR REPLACE FUNCTION pred_function (
            p_schema  IN  VARCHAR2 DEFAULT NULL,
            p_object  IN  VARCHAR2 DEFAULT NULL)
         RETURN VARCHAR2
         AS
            total  NUMBER;
         BEGIN
            SELECT COUNT (*) INTO total FROM testTable;
            RETURN '1 <= ' || total;
         END;
         /
Function created.
```

- Gán policy function trên cho bảng DEPT của SCOTT:

```
sec_mgr> BEGIN
            DBMS_RLS.add_policy
                (object_schema     => 'SCOTT',
                object_name        => 'DEPT',
                policy_name        => 'debug',
                function_schema    => 'SEC_MGR',
                policy_function    => 'pred_function');
         END;
         /
```

- Ta nhận thấy mọi thứ ban đầu đều làm việc tốt:

```
scott> SELECT COUNT(*) FROM dept;
```

```
COUNT(*)
--------
  4
```

- Tuy nhiên, nếu bảng TEST bị xóa đi thì sẽ có lỗi sinh ra do policy function không hợp lệ và không được recompile:

```
sec_mgr> DROP TABLE test;
Table dropped.
scott> SELECT COUNT(*) FROM dept;
SELECT COUNT(*) FROM dept
                     *
ERROR at line 1:
ORA-28110:policy function or package SEC_MGR.PRED_FUNCTION has
error
```

- Để sửa lỗi trên ta chỉ cần recover lại việc xóa bảng bằng lệnh Flashback Drop. Khi đó việc thực thi của chính sách bảo mật cũng sẽ được phục hồi:

```
sec_mgr> FLASHBACK TABLE test TO BEFORE DROP;
Flashback complete.

scott> SELECT COUNT(*) FROM dept;
COUNT(*)
--------
  4
```

- Tuy nhiên, trong ví dụ trên, vấn đề chúng ta quan tâm là khi câu truy vấn được thực hiện sau khi bảng TEST bị xóa và trước khi việc phục hồi bảng TEST được thực hiện. Khi đó CSDL sẽ hiển thị lỗi với nội dung chỉ ra chính xác tại sao câu truy vấn bị lỗi (trong ví dụ trên là hiển thị ra tên policy function và schema mà nó thuộc về). Thông báo tuy rất có ích để ta biết được lỗi xảy ra do đâu nhưng nó lại làm lộ 2 thông tin nhạy cảm mà người dùng bình thường không nên biết. Thứ nhất là nó chỉ ra rằng có một chính sách bảo mật trên bảng đó. Thứ hai nó cho biết tên của policy function bảo vệ bảng đó và schema mà function đó thuộc về.

- Từ lý do trên xuất hiện nhu cầu che giấu các exception do policy function gây ra để không hiển thị các thông tin nhạy cảm cho người dùng thấy. Cách tiếp cận tốt nhất cho yêu cầu này là sử dụng câu SQL động (dynnamic SQL) và xử lý ngoài lề. Policy function sẽ vẫn phải return ra một giá trị vì việc return null sẽ có thể dẫn tới việc cho phép người dùng truy xuất đến tất cả các record. Function cần xử lý sao cho nếu có lỗi xảy ra thì không có record nào được trả về. Ví dụ sau minh họa cho cách làm này:

```
sec_mgr> CREATE OR REPLACE FUNCTION pred_function (
            p_schema  IN  VARCHAR2 DEFAULT NULL,
            p_object  IN  VARCHAR2 DEFAULT NULL)
         RETURN VARCHAR2
         AS
            total NUMBER;
         BEGIN
            EXECUTE IMMEDIATE 'SELECT COUNT (*) FROM testTable'
               INTO total;
            RETURN '1 <= ' || total;
         EXCEPTION
            WHEN OTHERS THEN RETURN '1 = 0';
         END;
         /
Function created.
scott> SELECT COUNT(*) FROM dept;
COUNT(*)
--------
  4
```

- Khi xóa bảng TEST, chính sách bảo mật bị lỗi. Không có record nào được trả về và không có thông báo lỗi nào được đưa ra cho user.

```
sec_mgr> DROP TABLE test;
Table dropped.
scott> SELECT COUNT(*) FROM dept;
COUNT(*)
--------
  0
```

IV. Column Sensitive VPD

A. Lý thuyết

- Oracle Database cung cấp thêm 1 tính năng hữu dụng cho VPD gọi là Column Sensitive VPD. Mục đích của tính năng này là thực hiện các chính sách bảo mật khi cột cần bảo vệ được tham khảo.

B. Thực hành

1. Tham số SEC_RELEVANT_COLS

- Giả sử ta có 1 chính sách bảo vệ trên bảng EMP của SCOTT quy định một user có thể xem tất cả thông tin của những nhân viên khác ngoại trừ lương và ngày bắt đầu làm việc của họ và có thể xem tất cả thông tin của bản thân (kể cả lương và ngày bắt đầu làm việc). Điều đó có nghĩa là ta cần hiện thực 1 chính sách bảo mật trên EMP quy định một user chỉ được truy xuất đến record của bản thân người đó nếu trong câu lệnh truy xuất có tham khảo đến một trong hai cột SAL và HIREDATE:

```
sec_mgr> CREATE OR REPLACE FUNCTION user_only (
            p_schema   IN  VARCHAR2 DEFAULT NULL,
            p_object   IN  VARCHAR2 DEFAULT NULL)
          RETURN VARCHAR2
          AS
          BEGIN
             RETURN 'ename = user';
          END;
          /
Function created.
```

(Giả sử username của các account chính là ename được lưu trong bảng EMP).

- Để hiện thực chính sách, ta chỉ áp dụng policy function vừa tạo lên cột SAL và HIREDATE của bảng EMP. Ta sử dụng tham số SEC_RELEVANT_COLS để chỉ định cụ thể các cột được áp dụng chính sách.

```
sec_mgr> BEGIN
            DBMS_RLS.add_policy
                (object_schema       => 'SCOTT',
```

```
                object_name          => 'EMP',
                policy_name          => 'people_sel_sal',
                function_schema      => 'SEC_MGR',
                policy_function      => 'user_only',
                statement_types      => 'SELECT',
                sec_relevant_cols    => 'SAL,HIREDATE');
        END;
        /
PL/SQL procedure successfully completed.
```

- Để kiểm tra việc áp dụng chính sách trên, đầu tiên ta truy vấn bảng EMP bằng account SCOTT nhưng không select các cột SAL và HIREDATE. Do vậy SCOTT có thể thấy được record của các user khác.

```
scott> SELECT ename,job FROM emp WHERE ename >= 'S';
ENAME        JOB
----------   ---------
SMITH        CLERK
SCOTT        ANALYST
WARD         SALESMAN
TURNER       SALESMAN
```

- Thêm cột SAL và HIREDATE vào câu truy vấn, chính sách bảo mật được áp dụng nên SCOTT chỉ còn thấy được thông tin của chính user này. Ta cũng thấy rằng chỉ cần ít nhất một trong 2 cột trên xuất hiện trong câu truy vấn, chính sách bảo mật đã được áp dụng.

```
scott> SELECT ename,sal FROM emp;
ENAME         SAL
---------- --------
SCOTT        3000

scott> SELECT ename,hiredate FROM emp;
ENAME      HIREDATE
---------- ----------
SCOTT      19-APR-87
```

```
scott> SELECT ename,hiredate,sal FROM emp;
ENAME      HIREDATE     SAL
---------- ---------- --------
SCOTT      19-APR-87    3000
```

- Nếu một trong hai cột SAL và HIREDATE xuất hiện trong mệnh đề WHERE, chính sách bảo mật cũng được áp dụng. Ta so sánh kết quả truy vấn sử dụng cùng một câu select của 2 user SCOTT và BACKUP_MGR (user đã được gán quyền miễn trừ đối với tất cả các chính sách RLS trong phần I):

```
backup_mgr> SELECT ename FROM scott.emp
            WHERE ename >= 'S' AND sal > 1000;
ENAME
----------
SCOTT
WARD
TURNER

scott> SELECT ename FROM emp
       WHERE ename >= 'S' AND sal > 1000;
ENAME
----------
SCOTT
```

2. Tham số SEC_RELEVANT_COLS_OPT

- Có một nhu cầu thực tế là người quản trị mong muốn cho dù người dùng có tham khảo đến cột được bảo vệ, kết quả trả về sẽ chứa đầy đủ các record giống như khi không có tham khảo đến cột đó và các giá trị của cột đó ở những record được bảo vệ sẽ có giá trị null. Điều này không chỉ giúp cho dữ liệu trả về cho người dùng được đầy đủ mà còn giúp cho chính sách bảo mật trở nên trong suốt đối với người dùng. Cách tiếp cận này được gọi là "**column masking**".

- Xét tiếp ví dụ trong phần 1. Điều ta mong đợi là khi trong câu truy vấn có tham khảo đến cột SAL hoặc HIREDATE, tất cả các record thỏa câu truy vấn đều được

trả về và giá trị tại 2 cột này của những record của các user khác sẽ có giá trị null.

```
sec_mgr> BEGIN
            -- Xóa chính sách hiện tại
            DBMS_RLS.drop_policy
               (object_schema     => 'SCOTT',
                object_name       => 'EMP',
                policy_name       => 'people_sel_sal');
            -- Tạo lại chính sách với thay đổi ở
            -- tham số SEC_RELEVANT_COLS_OPT
            DBMS_RLS.add_policy
               (object_schema     => 'SCOTT',
                object_name       => 'EMP',
                policy_name       => 'people_sel_sal',
                function_schema   => 'SEC_MGR',
                policy_function   => 'user_only',
                statement_types   => 'SELECT',
                sec_relevant_cols => 'SAL,HIREDATE',
                sec_relevant_cols_opt => DBMS_RLS.all_rows);
         END;
         /
PL/SQL procedure successfully completed.

scott> SELECT ename,job,sal,hiredate
       FROM emp WHERE ename >= 'S' ;
ENAME      JOB           SAL  HIREDATE
---------- --------- ------ ------------
SMITH      CLERK
SCOTT      ANALYST     3000  19-APR-87
WARD       SALESMAN
TURNER     SALESMAN
```

*Lưu ý: tham số **sec_relevant_cols_opt** chỉ có thể áp dụng cho câu lệnh **SELECT**.*

V. Bài tập

1. Hoàn thiện chính sách bảo mật ở bài thực hành số 6 (HolidayControl) để đảm bảo khi exception xảy ra sẽ không bộc lộ thông tin nhạy cảm của chính sách bảo mật này.

2. Chỉnh sửa lại chính sách bảo mật ở câu 1, cho phép An xem được thông tin EmpNo và Name của các nhân viên khác trong bảng EmpHoliday nhưng chỉ xem được ngày nghỉ của chính mình.

3. Từ chính sách HolidayControl ở câu 2, thiết lập quyền không ảnh hưởng và đảm bảo sự giám sát đối với chính sách này cho user Han. Thực thi một số thay đổi dữ liệu và xem kết quả.