Bài thực hành số 7

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 sự khó chịu 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

- Ví dụ sau sẽ minh họa việc cấp quyền này và ảnh hưởng của nó trên các function RLS:

```
system@KNOX10g> CONN system/manager
Connected.
system@KNOX10g> -- SYSTEM bị ảnh hưởng bởi các policy function
system@KNOX10g> -- Nên không có record nào được delete
system@KNOX10g> DELETE FROM scott.emp;
0 rows deleted.
```

```
system@KNOX10g> -- Cấp quyền để SYSTEM kg bị ảnh hưởng bởi
policy function
system@KNOX10g> GRANT EXEMPT ACCESS POLICY TO SYSTEM;

Grant succeeded.

system@KNOX10g> -- SYSTEM không còn bị ảnh hưởng bởi các policy
function
system@KNOX10g> -- Tất cả các record bị xóa
system@KNOX10g> DELETE FROM scott.emp;

14 rows deleted.

system@KNOX10g> ROLLBACK;
```

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@KNOX10g> SELECT grantee
                 FROM dba_sys_privs
                 WHERE PRIVILEGE = 'EXEMPT ACCESS POLICY';

GRANTEE
------------------------------
SYSTEM
```

- Ví dụ sau sẽ trình bày cách để audit quyền này. Đồng thời mỗi khi áp dụng việc này, ta cũng cần kiểm tra xem việc audit đã được thực hiện như mình nghĩ chưa.

```
sec_mgr@KNOX10g> -- Audit quyền EXEMPT ACCESS POLICY
sec_mgr@KNOX10g> AUDIT EXEMPT ACCESS POLICY BY ACCESS;

Audit succeeded.

sec_mgr@KNOX10g> -- Kiểm tra việc audit bằng cách thực hiện tác
sec_mgr@KNOX10g> -- vụ trong account người được cấp quyền
sec_mgr@KNOX10g> CONN system/manager
Connected.
system@KNOX10g> DELETE FROM scott.emp;

14 rows deleted.

system@KNOX10g> -- Rollback để undo lệnh delete
system@KNOX10g> -- Lệnh Rollback sẽ không xóa được audit record
mà Oracle
system@KNOX10g> -- vừa tạo ra khi SYSTEM thực hiện lệnh DELETE
system@KNOX10g> ROLLBACK ;
Rollback complete.

system@KNOX10g> CONN sec_mgr/oracle10g
Connected.
sec_mgr@KNOX10g> -- Hiển thị tác vụ vừa được audit
```

```
sec_mgr@KNOX10g>
BEGIN
    FOR rec IN (SELECT * FROM dba_audit_trail)
    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 ('How: "'|| rec.sql_text || '"');
         DBMS_OUTPUT.put_line ('Using: ' || rec.priv_used);
    END LOOP;
END;
/
-------------------------
Who:   SYSTEM
What:  DELETE on SCOTT.EMP
When:  04/04 14:22
How:   "DELETE FROM scott.emp"
Using: DELETE ANY TABLE
-------------------------
Who:   SYSTEM
What:  DELETE on SCOTT.EMP
When:  04/04 14:22
How:   "DELETE FROM scott.emp"
Using: EXEMPT ACCESS POLICY
PL/SQL procedure successfully completed.
```

- Audit trail hiển thị 2 record bởi vì SYSTEM đã sử dụng 2 quyền khi thực hiện lệnh DELETE. 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.

III. Xử lý các Exception về Policy Function

A. Lý thuyết

- Nói chung có 2 loại error có thể khiến cho một chính sách RLS bị thất bại:
  - ✓ 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.

B. Thực hành

- Trong 2 loại lỗi trên, loại thứ nhất có thể làm mất đi tính trong suốt của VPD. Ta xem ví dụ dưới đây:

```
scott@KNOX10g> -- Tạo ra table sẽ được gọi bởi policy function
scott@KNOX10g> CREATE TABLE t AS SELECT * FROM DUAL;
Table created.

scott@KNOX10g> -- Tạo ra policy function phụ thuộc vào table t
scott@KNOX10g> CREATE OR REPLACE FUNCTION pred_function (
               p_schema  IN  VARCHAR2 DEFAULT NULL,
               p_object  IN  VARCHAR2 DEFAULT NULL)
                  RETURN VARCHAR2
               AS    l_total_recs  NUMBER;
               BEGIN
                     SELECT COUNT (*) INTO l_total_recs
                     FROM t;
                     RETURN '1 <= ' || l_total_recs;
               END;
          /
```

```
Function created.
scott@KNOX10g> CONN sec_mgr/oracle10g
Connected.

sec_mgr@KNOX10g> -- Gán policy function trên cho bảng EMP;
sec_mgr@KNOX10g> BEGIN
                 DBMS_RLS.add_policy
                     (object_schema     => 'SCOTT',
                     object_name        => 'EMP',
                     policy_name        => 'debug',
                     function_schema    => 'SCOTT',
                     policy_function    => 'pred_function');
                 END;
                 /
PL/SQL procedure successfully completed.
```

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

```
sec_mgr@KNOX10g> CONN scott/tiger
Connected.
scott@KNOX10g> SELECT COUNT(*) FROM emp;
               COUNT(*)
----------
        14
```

- Tuy nhiên, nếu bảng T bị xóa đi thì sẽ có lỗi sinh ra:

```
scott@KNOX10g> DROP TABLE t;
Table dropped.

scott@KNOX10g> -- Policy function không hợp lệ và không được
recompile
scott@KNOX10g> SELECT COUNT(*) FROM emp;
SELECT COUNT(*) FROM emp
                *
ERROR at line 1:
```

```
ORA-28110:policy function/package SCOTT.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:

```
scott@KNOX10g> -- Phục hồi bảng
scott@KNOX10g> FLASHBACK TABLE t TO BEFORE DROP;
Flashback complete.

scott@KNOX10g> SELECT COUNT(*) FROM emp;
  COUNT(*)
----------
        14
```

- Tuy nhiên, trong ví dụ trên, vấn đề chúng ta quan tâm là khi câu truy vấn được thực hiện sau khi bảng T bị xóa và trước khi việc phục hồi bảng T được thực hiện. Khi đó CSDL sẽ đưa ra lỗi và có chỉ ra chính xác tại sao câu truy vấn bị lỗi. 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 table đó. Thứ hai nó cho biết tên của chính sách bảo mật bảo vệ table đó.

- Từ đó xuất hiện nhu cầu che dấu 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 1 câu SQL động (dynnamic SQL) che dấu các ràng buộc trên đối tượng CSDL. 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:

```
scott@KNOX10g> CREATE OR REPLACE FUNCTION pred_function (
                    p_schema   IN   VARCHAR2 DEFAULT NULL,
                    p_object   IN   VARCHAR2 DEFAULT NULL)
               RETURN VARCHAR2
               AS
                    l_total_recs   NUMBER;
               BEGIN
               -- Câu SQL động để che dấu sự phụ thuộc trên T
                    EXECUTE IMMEDIATE 'SELECT COUNT (*) FROM t'
                         INTO l_total_recs;
                    RETURN '1 <= ' || l_total_recs;
               EXCEPTION
               WHEN OTHERS THEN
-- Loại bỏ tất các record khi chính sách bảo mật không
-- thực hiện được
                    RETURN '1=0';
               END;
               /
               Function created.

scott@KNOX10g> SELECT COUNT(*) FROM emp;
  COUNT(*)
----------
        14

scott@KNOX10g> DROP TABLE t;
Table dropped.

scott@KNOX10g> -- Chính sách bảo mật bị lỗi. Không có record nào
scott@KNOX10g> -- được trả về và không có thông báo lỗi nào được
scott@KNOX10g> -- đưa ra cho user
scott@KNOX10g> SELECT COUNT(*) FROM emp;
  COUNT(*)
----------
```

0

IV. Column Sensitive VPD

A. Lý thuyết

- Oracle Database 10g cung cấp thêm 1 tính năng mới 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

- Giả sử ta có một chính sách bảo vệ trên bảng EMP của SCOTT quy định một user có thể thấy tất cả các thông tin của những nhân viên khác ngoại trừ lương của họ. Khi đó 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 cột salary:

```
sec_mgr@KNOX10g> CREATE OR REPLACE FUNCTION user_only (
                     p_schema   IN  VARCHAR2 DEFAULT NULL,
                     p_object   IN  VARCHAR2 DEFAULT NULL)
                  RETURN VARCHAR2
                  AS
                  BEGIN
               -- Giả sử username của các account
               -- chính là username được lưu trong bảng EMP
                  RETURN 'username = user';
                  END;
                  /
Function created.
```

```
sec_mgr@KNOX10g> 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    => 'SALARY');
END;
/
PL/SQL procedure successfully completed.
```

- Kiểm tra lại việc áp dụng chính sách trên:

```
scott@KNOX10g> -- Người dùng có thể thấy tất cả các record khi
scott@KNOX10g> -- salary không được truy vấn.
scott@KNOX10g> -- Câu lệnh sau chỉ xuất 5 record đầu tiên
scott@KNOX10g> SELECT username FROM emp
               WHERE ROWNUM <= 5;
USERNAME
----------
SMITH
ALLEN
WARD
JONES
MARTIN

scott@KNOX10g> -- Thêm cột salary vào câu truy vấn, chính sách
scott@KNOX10g> -- bảo mật được áp dụng
scott@KNOX10g> SELECT username, salary FROM emp;

USERNAME       SALARY
---------- ----------
```

```
SCOTT                3000
```

- 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 trên. Điều ta mong đợi là khi trong câu truy vấn có tham khảo đến cột salary, tất cả các record thỏa câu truy vấn đều được trả về và giá trị tại cột salary của những record của các user khác sẽ có giá trị null.

```
sec_mgr@KNOX10g> BEGIN
          -- Xóa chính sách hiện tại
                    DBMS_RLS.drop_policy
                     (object_schema    => 'SCOTT',
                      object_name      => 'EMP',
                      policy_name      => 'people_sel_sal');
          -- Thêm 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 => 'SALARY',
                          sec_relevant_cols_opt      =>
                                    DBMS_RLS.all_rows);
                END;
                /
PL/SQL procedure successfully completed.
scott@KNOX10g> SELECT username, salary FROM emp
               WHERE deptno = 20;
```

```
USERNAME        SALARY
----------      ----------
SMITH
JONES
SCOTT            3000
ADAMS
FORD
```

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 Annu xem được các thông tin trong bảng Holiday 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 Hann. Thực thi một số thay đổi dữ liệu và xem kết quả.