Bài thực hành số 4

VIRTUAL PRIVATE DATABASE (1)

❖ *Tóm tắt nội dung:*

- Giới thiệu về Virtual Private Database
- Row-level security
- Kỹ thuật làm việc với policy function

I. Giới thiệu chung

- Trong nhiều năm dài, việc áp dụng các chính sách bảo mật cho dữ liệu nằm trong các bảng CSDL được hiện thực bằng việc sử dụng view cùng với các function. Tuy nhiên cách hiện thực này nhiều khi không thể là một giải pháp thực tế cho mục đích trên, đặc biệt khi cần thực hiện bảo mật ở mức độ "dòng dữ liệu" (row-level security). Thấy được nhu cầu ngày càng cao của người dùng, từ Oracle Database 8.1.5, Oracle đã giới thiệu một công nghệ mới rất hiệu quả là Virtual Private Database (từ đây gọi tắt là VPD).
- VPD là sự kết hợp của 2 kỹ thuật:
  - ✓ ***Fine-grained access control (FGAC):*** cho phép người quản trị dùng các function để hiện thực các chính sách bảo mật và liên kết các chính sách bảo mật đó với các table, view hoặc synonym. Việc gán các chính sách như vậy khiến cho những người dùng với quyền hạn khác nhau sẽ thấy được những "khung nhìn" khác nhau đối với đối tượng được bảo vệ. Việc giới hạn khung nhìn này sẽ trong suốt đối với mọi người dùng. Đồng thời chính sách bảo mật đó sẽ được áp dụng cho bất kỳ user nào truy xuất đến table đó mà không cần người quản trị phải gán chính sách cho từng user. Điều này khiến các chính sách bảo mật được hiện thực bằng FGAC dễ quản lý hơn khi hiện thực bằng view.
  - ✓ ***Application Context***: cung cấp một nơi lưu trữ bảo mật cho những giá trị ngữ cảnh ứng dụng. Sử dụng Application Context sẽ nâng cao hiệu quả thực hiện của FGAC (trong chương trình chúng ta không học về Application-context).

  Lưu ý: bởi vì đây là 1 phương pháp hiệu quả và phổ biến để hiện thực việc bảo mật ở

mức dòng dữ liệu trong Oracle, nên người ta thường dùng thuật ngữ *Row-level security* (RLS) để thay cho *Fine-grained access control* hoặc *Virtual Private Database*.

II. Row-level Security

A. Lý thuyết

1. Row-level Security

- Row-level security (RLS) cho phép giới hạn việc truy xuất các hàng (record) dựa trên một chính sách bảo mật (security policy) được hiện thực bằng PL/SQL. Một chính sách bảo mật mô tả các quy định quản lý việc truy xuất các dòng dữ liệu.

2. Cơ chế thực hiện

- Để thực hiện RLS, đầu tiên ta tạo 1 hàm PL/SQL (PL/SQL function) trả về một chuỗi (string). Chuỗi string này chứa các điều kiện của chính sách bảo mật mà ta muốn hiện thực.
- Hàm PL/SQL vừa được tạo ở trên sau đó được đăng ký cho các table, view mà ta muốn bảo vệ bằng cách dùng package PL/SQL DBMS_RLS.
- Khi có một câu truy vấn của bất kỳ user nào trên đối tượng được bảo vệ, Oracle sẽ nối chuỗi được trả về từ hàm nêu trên vào mệnh đề WHERE của câu lệnh SQL ban đầu (nếu trong câu lệnh SQL ban đầu không có mệnh đề WHERE thì Oracle sẽ tự động tạo thêm mệnh đề WHERE để đưa chuỗi điều kiện vào), nhờ đó sẽ lọc được các hàng dữ liệu theo các điều kiện của chính sách bảo mật.

3. Các lưu ý khi làm việc với RLS

- Các hàm PL/SQL được đăng ký cho các table, view hay synonym bằng cách gọi thủ tục DBMS_RLS.ADD_POLICY.
- Thủ tục ADD_POLICY đòi hỏi ít nhất phải có 3 tham số nhập vào: object_name, policy_name, policy_function.

  (Mô tả chi tiết của package DBMS_RLS được chứa trong file **dbmsrlsa.sql** tại ***<thư mục cài Oracle>\product\11.2.0\dbhome_1\RDBMS\ADMIN***).
- Sự kết hợp của object_schema, object_name, và policy_name phải là duy nhất.
- Mặc định, policy sẽ được áp dụng cho tất cả các lệnh DML. Người quản trị có thể dùng tham số STATEMENT_TYPES để chỉ ra policy áp dụng cho loại câu lệnh nào.

- Bất cứ khi nào 1 user truy xuất một cách trực tiếp hay gián tiếp vào đối tượng được bảo vệ, RLS engine sẽ được gọi một cách trong suốt, hàm PL/SQL đã đăng ký sẽ được thực thi, và rồi lệnh SQL của user sẽ được chỉnh sửa và thực thi.
- Tuy nhiên, *account SYS không bị ảnh hưởng bởi bất kỳ chính sách bảo mật nào.*
- Nhiều policy cũng có thể áp dụng cho cùng 1 đối tượng. Khi đó CSDL sẽ kết hợp tất cả các policy đó lại với nhau theo phép AND.
- Quyền sử dụng package DBMS_RLS không được gán cho mọi người dùng. Những người quản trị cần được gán quyền EXECUTE ON DBMS_RLS để có thể sử dụng được nó.
- Tất cả các policy function mà ta tạo ra đều phải có đúng 2 tham số truyền vào. Tham số đầu tiên là tên của schema sở hữu đối tượng mà chính sách RLS đó bảo vệ. Tham số thứ hai là tên của đối tượng được bảo vệ. Hai tham số này rất hữu ích vì 1 policy function có thể được áp dụng cho nhiều đối tượng khác nhau trong nhiều schema khác nhau. Tên của các tham số có thể được đặt thoải mái nhưng thứ tự của 2 tham số phải tuân thủ theo thứ tự trên. Các tham số sẽ được dùng để xác định đối tượng nào mà chính sách đó được gọi cho nó. Kiểu của 2 tham số truyền vào và của giá trị trả về phải là kiểu VARCHAR2.
- Policy function cần được tạo ra trong schema của người quản trị bảo mật. Điều này quan trọng vì việc truy xuất vào các policy function cần được bảo vệ. Các user khác không nên có quyền thực thi hay quyền alter hoặc quyền drop trên các policy function này.
- Để hiện thực được các chính sách bảo mật phức tạp một cách hiệu quả, thông thường người ta sử dụng kết hợp RLS với Application Context. Nhờ đó các chính sách bảo mật sẽ được áp dụng theo các điều kiện linh hoạt hơn (ví dụ: áp dụng chính sách bảo mật nào là dựa trên người dùng thuộc Department số mấy). Trong chương trình thực hành của chúng ta không học về Application Context, sinh viên tự tìm hiểu thêm về vấn đề này.

B. Thực hành

1. Tạo chính sách bảo mật RLS

- Ta có bảng EMP thuộc schema của user SCOTT[1] với nội dung sau:

```
sec_mgr> SELECT DISTINCT deptno FROM scott.emp;
DEPTNO
----------
20
30
10
```

- Giả sử ta có một chính sách bảo mật (security policy) quy định không một người dùng nào được truy xuất đến các record thuộc Department có deptno là 10 trong bảng EMP. Để chính sách này có thể áp dụng cho CSDL, đầu tiên ta cần tạo 1 PL/SQL function trong schema SEC_MGR[2] có chuỗi trả về là điều kiện của chính sách bảo mật trên:

```
sec_mgr> CREATE OR REPLACE FUNCTION no_dept10 (
             p_schema   IN  VARCHAR2,
             p_object   IN  VARCHAR2)
          RETURN VARCHAR2
          AS
          BEGIN
             RETURN 'deptno != 10';
          END;
          /
Function created.
```

- Sau khi tạo function hiện thực chính sách bảo mật, ta cần đăng ký function đó cho đối tượng mà chính sách đó muốn bảo vệ bằng cách dùng thủ tục ADD_POLICY trong package DBMS_RLS.

[1] Đây là user được tạo sẵn trong Oracle với các bảng dữ liệu để dùng cho các trường hợp chạy demo, hướng dẫn. Nếu quá trình cài đặt chưa tự động tạo ra user này, SV có thể tự thực thi script tạo schema SCOTT: **<thư mục cài Oracle>\product\11.2.0\dbhome_1\RDBMS\ADMIN\scott.sql**

[2] Trong Lab 04 và Lab 05, SV cần chú ý username được đặt bên trái dấu "**>**" tại đầu mỗi câu lệnh để biết câu lệnh đó được thực hiện bằng user account nào. Trong bài lab sẽ không nêu cụ thể các lệnh log in/log out vào các tài khoản, SV cần tự thực hiện các lệnh log in/log out.

```
sec_mgr> BEGIN
            DBMS_RLS.add_policy
            (object_schema     => 'SCOTT',
            object_name        => 'EMP',
            policy_name        => 'quickstart',
            policy_function    => 'no_dept10');
         END;
         /
PL/SQL procedure successfully completed.
```

- Hai bước hiện thực chính sách bảo mật vừa trình bày ở trên nên được thực hiện bởi account chịu trách nhiệm về quản lý bảo mật (trong ví dụ này và cả các ví dụ khác trong bài lab, account chịu trách nhiệm quản lý bảo mật là **SEC_MGR**).

- Để kiểm tra xem chính sách này có làm việc không, ta lần lượt log on vào các account SEC_MGR và SCOTT truy xuất bảng EMP bằng lệnh DML đã sử dụng lúc đầu. Câu lệnh sau sẽ yêu cầu hiển thị ra tất cả các Department có trong bảng. Tuy nhiên, cho dù log on vào account nào ta cũng sẽ thấy rằng các Department có deptno bằng 10 sẽ không xuất hiện trong kết quả câu truy vấn, bởi vì chính sách RLS đã tự động lọc ra những record đó. Bên cạnh đó, kết quả được trả về bình thường nên user thực hiện câu lệnh cũng không biết được mình bị ảnh hưởng bởi 1 chính sách RLS.

```
sec_mgr> SELECT DISTINCT deptno FROM scott.emp;
DEPTNO
----------
20
30

scott> SELECT DISTINCT deptno FROM emp;
DEPTNO
----------
20
30
```

- Một ưu điểm của RLS nữa là ta có thể thay đổi nội dung của 1 chính sách bảo mật bằng cách viết lại function hiện thực chính sách đó mà không cần phải đăng ký lại chính sách đó cho đối tượng cần bảo vệ. Để thấy được ưu điểm này, ta trở lại với ví dụ trên, thay đổi nội dung của function no_dept10:

```
sec_mgr> CREATE OR REPLACE FUNCTION no_dept10 (
            p_schema  IN  VARCHAR2,
            p_object  IN  VARCHAR2)
         RETURN VARCHAR2
         AS
         BEGIN
            RETURN 'USER != ''SCOTT''';
         END;
         /
```

- Chính sách vừa được sửa đổi quy định không cho người dùng SCOTT truy xuất nội dung của bảng được bảo vệ (USER trong chuỗi là một hàm của Oracle trả về tên người dùng hiện tại của session đó). Ta kiểm tra lại xem việc áp dụng chính sách đã được thay đổi chưa bằng cách lần lượt log on vào hệ thống bằng 2 account SEC_MGR, SCOTT và truy xuất bảng EMP:

```
sec_mgr> SELECT COUNT(*) Total_Records FROM scott.emp;
TOTAL_RECORDS
-------------
14
scott> SELECT COUNT(*) Total_Records FROM emp;
TOTAL_RECORDS
-------------
0
```

2. Kiểm tra nội dung chuỗi trả về

- Sau khi tạo các policy function, nếu muốn ta có thể kiểm tra chuỗi trả về của function vừa tạo bằng cách thực hiện câu lệnh sau:

```
sec_mgr> col predicate format a50;
sec_mgr> SELECT no_dept10('SCOTT','EMP') predicate FROM DUAL;
```

```
PREDICATE
--------------------------------------------------
USER != 'SCOTT'
```

- Nếu trong câu lệnh tạo policy function ta quy định các tham số có giá trị mặc định là null thì câu lệnh để kiểm tra chuỗi trả về vừa nêu ở trên có thể được viết ngắn gọn lại:

```
sec_mgr> CREATE OR REPLACE FUNCTION no_dept10 (
            p_schema   IN  VARCHAR2 DEFAULT NULL,
            p_object   IN  VARCHAR2 DEFAULT NULL)
          RETURN VARCHAR2
          AS
          BEGIN
             RETURN 'USER != ''SCOTT''';
          END;
          /
Function created.
sec_mgr> col predicate format a50;
sec_mgr> SELECT no_dept10 predicate FROM DUAL;
PREDICATE
--------------------------------------------------
USER != 'SCOTT'
```

3. Tham số STATEMENT_TYPES

- Giả sử ta có chính sách bảo mật quy định các user chỉ được insert và update trên các dòng dữ liệu của các Department có deptno < 30. Khi đó ta dùng tham số `statement_types` để quy định loại câu lệnh nào mới áp dụng chính sách bảo mật này. Nếu không đề cập đến tham số này, mặc định chính sách sẽ áp dụng cho tất cả các loại câu lệnh. Policy function được hiện thực như sau:

```
sec_mgr> CREATE OR REPLACE FUNCTION dept_less_30 (
            p_schema   IN  VARCHAR2 DEFAULT NULL,
            p_object   IN  VARCHAR2 DEFAULT NULL)
          RETURN VARCHAR2
```

```
            AS
            BEGIN
               RETURN 'deptno < 30';
            END;
            /

sec_mgr> BEGIN
               DBMS_RLS.add_policy
               (object_schema     => 'SCOTT',
               object_name        => 'EMP',
               policy_name        => 'EMP_IU',
               function_schema    => 'SEC_MGR',
               policy_function    => 'dept_less_30',
               statement_types    => 'INSERT,UPDATE');
         END;
         /
```

- Ta kiểm tra việc áp dụng chính sách bảo mật trên. Đầu tiên ta SELECT trên bảng EMP một số record, ta nhận thấy rằng ràng buộc ‘deptno < 30’ không ảnh hưởng đối với câu lệnh SELECT:

```
scott> SELECT ename, deptno
       FROM emp  WHERE ename < 'F';
ENAME       DEPTNO
---------- ----------
ALLEN       30
BLAKE       30
CLARK       10
ADAMS       20
```

- Tuy nhiên khi ta update trên bảng EMP, chính sách bảo mật sẽ được áp dụng. Trong ví dụ bên dưới, ALLEN có deptno = 30. Khi thực thi, Oracle sẽ nối thêm điều kiện ‘deptno < 30’ với điều kiện ename = ‘ALLEN’ của câu lệnh Update và kết quả là không tìm thấy hàng nào thỏa đồng thời cả 2 điều kiện, do vậy không có hàng nào được update, chứ không thông báo lỗi. Nhờ vậy, chính sách RLS đã được

áp dụng một cách trong suốt đối với người dùng.

```
scott> UPDATE emp SET deptno = 10 WHERE ename = 'ALLEN';
0 rows updated.
```

4. Tham số UPDATE_CHECK

- Ở ví dụ sau, tuy giá trị update là 30, vi phạm chính sách đã được tạo trong mục 3, nhưng vẫn update được. Đó là do Oracle chỉ kiểm tra điều kiện các chính sách với các giá trị trước khi update/insert, không quan tâm giá trị sau khi update/insert. Việc này sẽ làm mất tính trong suốt của việc áp dụng chính sách khi user SELECT lại bảng đã UPDATE và không thấy giá trị mình vừa update.

```
scott> UPDATE emp SET deptno = 30 WHERE ename = 'ADAMS';
1 row updated.
```

- Việc Oracle có kiểm tra lại các giá trị sau khi được insert/update hay không phụ thuộc vào tham số *update_check*. Đây là tham số tùy chọn cho các loại lệnh INSERT và UPDATE. Nó có giá trị mặc định là FALSE. Nếu *update_check* có giá trị TRUE, sau khi câu lệnh SQL đã được chỉnh sửa theo điều kiện của chính sách bảo mật và được thực thi, Oracle sẽ thực hiện việc kiểm tra lại các giá trị vừa được UPDATE/INSERT xem nó có vi phạm chính sách bảo mật không. Nếu có vi phạm thì việc thực thi câu lệnh SQL đó sẽ thất bại và thông báo lỗi sẽ được xuất ra. Điều này sẽ được thấy rõ ở phần tiếp theo. Trước tiên ta tạo lại một chính sách khác với tùy chọn tham số *update_check* có giá trị TRUE:

```
sec_mgr> BEGIN
            DBMS_RLS.add_policy
            (object_schema     => 'SCOTT',
            object_name        => 'EMP',
            policy_name        => 'EMP_IU_edit',
            function_schema    => 'SEC_MGR',
            policy_function    => 'dept_less_30',
            statement_types    => 'INSERT,UPDATE',
            update_check       => TRUE);
         END;
         /
```

- Tiếp theo ta update trên bảng EMP deptno của CLARK:

```
scott> UPDATE emp SET deptno = 30 WHERE ename = 'CLARK';
update emp
    *
ERROR at line 1:
ORA-28115: policy with check option violation
```

Trong câu lệnh trên, đầu tiên Oracle tìm được 1 hàng thỏa điều kiện ename = 'CLARK' và điều kiện của chính sách bảo mật là deptno < 30 nên nó đủ điều kiện để thực hiện lệnh update. Nhưng do ta đã thiết lập tham *số update_check = TRUE* nên Oracle sẽ kiểm tra những giá trị kết quả của việc update và nhận thấy rằng giá trị sau khi update vi phạm chính sách bảo mật deptno < 30 (câu lệnh trên đã thay đổi giá trị deptno từ 10 thành 30). Do có vi phạm này nên câu lệnh update trên bị thất bại và có thông báo lỗi xuất hiện.

- Ta kiểm tra tiếp trường hợp khi ta INSERT vào bảng EMP:

```
scott> INSERT INTO emp (empno, ename, deptno)
       VALUES (20,'KNOX',10);
1 row created.
scott> INSERT INTO emp (empno, ename, deptno)
       VALUES (21,'ELLISON',30);
insert into emp (empno, ename, deptno)
            *
ERROR at line 1:
ORA-28115: policy with check option violation
```

Tương tự như trường hợp khi ta kiểm tra với các lệnh Update, tác vụ insert 1 record có deptno >= 30 bị thất bại và sinh ra lỗi. Tác vụ này thất bại do vi phạm policy function và do ta đã thiết lập *UPDATE_CHECK=TRUE* khi gọi thủ tục ADD_POLICY. Nếu ta không thiết lập TRUE, việc insert trên sẽ thành công (nghĩa là sẽ có thêm 1 dòng có deptno = 30 được tạo ra).

5. Kỹ thuật ngăn truy xuất tất cả các hàng

- Một trong những cách hiệu quả nhất để ngăn không cho bất kỳ hàng nào bị truy

xuất bằng phương pháp RLS là tạo ra 1 policy function có chuỗi trả về chứa một điều kiện không bao giờ có thể xảy ra (ví dụ: chuỗi "1 = 0"). Cần lưu ý rằng trả về 1 chuỗi null hoặc chuỗi có độ dài bằng 0 thì sẽ cho kết quả ngược lại: ***tất cả các record sẽ được phép truy xuất***.

- Sẽ rất có lợi nếu ta tạo một policy function có tác dụng ngăn chặn tất cả các record. Mỗi khi cần khóa lại một bảng nào đó một cách nhanh chóng ta có thể sử dụng nó:

```
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.
```

- Đây cũng là một cách giúp ta có thể biến một bảng thành bảng chỉ được phép đọc (Read Only table). Ta chỉ cần đăng ký policy function trên cho bảng đó với lựa chọn áp dụng cho các câu lệnh INSERT, UPDATE, DELETE:

```
sec_mgr> BEGIN
            DBMS_RLS.add_policy
               (object_schema     => 'SCOTT',
               object_name        => 'EMP',
               policy_name        => 'PEOPLE_RO_IUD',
               function_schema    => 'SEC_MGR',
               policy_function    => 'No_Records',
               statement_types    => 'INSERT,UPDATE,DELETE',
               update_check       => TRUE);
         END;
         /
```

- Kiểm tra lại việc áp dụng chính sách bảo mật trên:

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

scott> UPDATE emp  SET ename = NULL;
0 rows updated. ----- Không thể update bất kỳ record nào

scott> DELETE FROM emp;
0 rows deleted. ------ Không thể delete bất kỳ record nào

scott> -- Không thể insert thêm bất kỳ record nào
scott> INSERT INTO emp (empno,ename) VALUES (25,'KNOX');

INSERT INTO emp (empno,ename) VALUES (25,'KNOX')
            *
ERROR at line 1:
ORA-28115: policy with check option violation
```

6. Xóa chính sách bảo mật

- Để xóa bỏ 1 chính sách bảo mật, ta dùng thủ tục DROP_POLICY của package DBMS_RLS. Ví dụ:

```
sec_mgr> BEGIN
            DBMS_RLS.drop_policy
               (object_schema     => 'SCOTT',
               object_name        => 'EMP',
               policy_name        => 'PEOPLE_RO_IUD');
         END;
         /
```

- Lưu ý là đoạn lệnh trên chỉ xóa bỏ chính sách bảo mật, chứ không xóa hàm đã dùng cho chính sách đó (policy function).

III. Bài tập

Tạo cấu trúc bảng sau:

EMPHOLIDAY

(EmpNo NUMBER (5),
Name VARCHAR2 (60),
Holiday DATE)

EmpNo	Name	Holiday
1	Han	2/1/2010
2	An	12/5/2010
3	Thu	26/8/2009

Xây dựng một policy HolidayControl cho các trường hợp sau đây:

1. An chỉ được xem và chỉnh sửa thông tin cá nhân của riêng mình.
2. Thu không được xem hay chỉnh sửa bất kỳ thông tin nào.
3. Han được quyền xem tất cả các thông tin nhưng chỉ chỉnh sửa (Insert, Update, Delete) được Holiday nào mà ngày lớn hơn hay bằng ngày hiện tại (không chỉnh sửa được ngày ở quá khứ).

   Ghi chú:

   1/ Giả thiết trường Name có các dữ liệu phân biệt nhau và tương ứng với Username của các account trong hệ thống.

   2/ Schema chứa đối tượng cần được bảo vệ là SCOTT.

   3/ Schema chứa policy function được sử dụng là SEC_MGR.