Tipps & Tricks: Row Level Security

Bereich:	DBA	Erstellung:	09/2008 MP
Versionsinfo:	10.1, 10.2, 11.1	Letzte Überarbeitung:	07/2009 MP

Row Level Security

DBMS_RLS Package
Es besteht die Möglichkeit, auf Views oder Tables eine Funktion zu legen, die abhängig von bestimmten Faktoren nur die Sicht auf die Teile der Tabelle bzw. View freigibt. Man könnte dies mit einem SELECT-Trigger (den es nicht gibt) vergleichen. Dadurch können viele statische Views eingespart werden, die für einzelne Benutzer sonst angelegt werden müssten.

Beispiel:

```
BEGIN
  DBMS_RLS.ADD_POLICY ( 'SCOTT', 'EMP', 'EMP_PRUEF', 'SYSTEM', 'EMP_SEC', 'select');
END;
/
```

Jedes Mal, wenn die EMP-Tabelle mit einem SELECT gelesen wird, prüft die EMP_SEC Funktion des Users SYSTEM und gibt dann nur Teile der Tabelle frei. Dies wird durch Hinzufügen einer WHERE-Klausel durchgeführt.

Der Benutzer setzt folgenden SELECT ab:

```
SELECT * FROM scott.emp;
```

Abgesetzt wird aber durch die Policy der folgende Befehl:

```
SELECT * FROM scott.emp WHERE P1;
```

P1 ist hier die Erweiterung des SELECTS um eine WHERE - Bedingung, die von der Funktion emp_sec zurückgeliefert wird.
Dies ist auch für SQL-Subqueries möglich. Der Benutzer benötigt keine Execute Rechte auf der Policy-Funktion. In den meisten Fällen wird er die Policy-Funktion nicht kennen. Wenn mehrere Policies existieren werden sie mit AND verknüpft.

1. Schritt: Policy Function erstellen

```
CONNECT system/manager;
CREATE OR REPLACE FUNCTION emp_sec (schema IN varchar2, tab IN varchar2)
RETURN VARCHAR2 AS
BEGIN
RETURN 'ename='''|| sys_context('userenv','session_user') ||'''';
```

```
END emp_sec;
/
```

2. Schritt: Policy Funktion für SELECTS einschalten

```
BEGIN
   dbms_rls.add_policy(
     object_schema=>'scott',
     object_name=>'emp',
     policy_name=>'emp_policy',
     function_schema=>'system',
     policy_function=>'emp_sec',-
     statement_types => 'select',
     update_check => TRUE,
     enable => TRUE);
END;
/
```

Anmerkung: Wenn der Parameter UPDATE_CHECK gesetzt wurde, wird auch INSERT/UPDATE mit geprüft

3. Schritt: Testen Policy Funktion

```
CONNECT scott/tiger;
SELECT empno,ename,sal,deptno FROM scott.emp;
```

```
EMPNO     ENAME     SAL     DEPTNO

7788     SCOTT     3000    10
```

Anmerkung: Der SELECT Befehl wird umgewandelt in:

```
SELECT empno,ename,sal,deptno
FROM scott.emp
WHERE ename='SCOTT';
```

```
CONNECT sys/manager;
SELECT empno,ename,sal,deptno FROM scott.emp
```

```
EMPNO     ENAME     SAL     DEPTNO

7369      SMITH     800     20
7499     ALLEN     1600    10
7521     WARD     1250    30
7566     JONES     2975    10
...
```

Anmerkung: Für SYS werden keine Policies angewendet!

```
CONNECT system/manager;
SELECT * FROM scott.emp;
no rows selected
```

Anmerkung: Der SELECT wird umgewandelt in:

```
SELECT empno,ename,sal,deptno
FROM scott.emp
WHERE ename='SYSTEM';
```

4. Schritt: Löschen/deaktivieren der Policy Funktion

```
CONNECT system/manager;
```

Löschen der Policy

```
execute dbms_rls.drop_policy(object_schema=>'scott', object_name=>'emp',
policy_name=>'emp_policy');
```

Deaktivieren der Policy

```
execute dbms_rls.enable_policy(object_schema=>'scott', object_name=>'emp',
policy_name=>'emp_policy', enable=>'FALSE');
```

Tipps & Tricks

Informationen über die Policies finden Sie in der View DBA_POLICIES

```
OBJECT_OWNER    OBJECT_NAME    POLICY_GROUP    POLICY_NAME    PF_OWNER    PACKAGE
   FUNCTION
SCOTT         EMP          SYS_DEFAULT     EMP_POLICY     SYSTEM
 EMP_SEC

SEL           INS          UPD           DEL            CHK        ENA       STA
YES            NO            NO              NO              YES        YES        NO
```

Erweiterungen in 10g

```
EMPNO    ENAME    HIREDATE    SAL    DEPTNO
7369    SMITH    17.12.80    2000   10
7499    ALLEN    20.02.81          20
7521    WARD    20.04.81          30
7566    JONES    01.03.80          20
7654    MARTIN   01.05.81          10
```

Hier werden bei der Ausgabe der Spalten sal und comm nur die Werte für die Zeilen angezeigt, die die Funktion erfüllen:

Die Policy dazu lautet:

```
BEGIN
   DBMS_RLS.ADD_POLICY (
      object_schema => 'SCOTT',
      object_name => 'EMP',
      policy_name => 'SP_JOB',
      function_schema => 'SYSTEM',
      policy_function => 'PF_JOB',
      sec_relevant_cols => 'sal,comm',
      sec_relevant_cols_opt => DBMS_RLS.ALL_ROWS );
END;
/
```

Hinweise:
Das Recht EXEMPT ACCESS POLICY erlaubt die Policy zu umgehen
DIRECT PATH Export umgeht auch Policies

Weitere Filterbeispiele:
Bei einer Return-Klausel 1=2 werden keine Zeilen, bei 1=1 werden alle Zeilen zurückgegeben

```
IF (....) THEN
RETURN ('1=2');
ELSE
RETURN ('1=1');
END;
```

Bestimmte IP-Adresse zulassen:

```
sys_context('userenv','IP_ADDRESS') LIKE '192.168.1.%'
```

Tageszeit / Wochentag:

```
to_number(to_char(sysdate,'HH24'))  between 6 and 20
to_char(sysdate,'D') NOT IN (6,7)
```

Bestimmte Benutzer

```
user IN ('SCOTT','SYSTEM','MARCO')
```