Virtual Private Database

Mathias Weber und Markus Geis, Institut für Notfallmedizin und Medizinmanagement, Klinikum der Universität München

Die Sicherheit von Daten vor unberechtigten Zugriff spielt in der heutigen Zeit eine entscheidende Rolle. Aufgabe von IT und Organisation ist es, Strukturen zu schaffen, die es ermöglichen, Informationen vor unbefugtem Zugriff zu schützen.

Das Institut für Notfallmedizin und Medizinmanagement (siehe „www.inm-online.de“) wurde zum Jahreswechsel 2001/02 als interdisziplinäre klinische Einrichtung am Klinikum der Universität München errichtet. Neben der interdisziplinären Forschung und Lehre in der Notfallmedizin, im Rettungswesen und im Management/Lehrmanagement der Medizin erbringt das INM vor allem Dienstleistungen auf den genannten Gebieten.

Ist-Zustand bei Zugriffsrechten bei Datenbank-Applikationen

Datenbank-Applikationen beruhen meist auf einer unterschiedlichen Anzahl von Tabellen, die die Daten der Applikation beinhalten. Werden für die Applikationen differenzierte Zugriffsrechte benötigt, können auf die Datenbank-Objekte Rechte vergeben werden (Objekt-Privilegien für Tables: „select“, „insert“, „update“ oder „delete“). Diese Privilegien beziehen sich auf die gesamten Objekte. Wenn etwa für eine bestimmte Tabelle das „select“-Privileg an einen Benutzer gegeben wird („grant select on emp to user1“), werden sämtliche Daten dieser Tabelle vollständig angezeigt („select * from emp“).

Wie kann man erreichen, dass ein Benutzer eine eingeschränkte Sicht auf Datensätze hat (etwa nur die Daten der Abteilung 20 sehen darf)? Diese Einschränkungen werden sehr oft über die Applikation realisiert, indem die Abfrage für jeden User vor der Ausführung geändert wird („select * from emp where deptno=20“). Dieser Applikationsaufbau benötigt dann eine selbstgeschriebene Userverwaltung, die als Basis eine vordefinierte Rechtestruktur beinhaltet; diese ermöglicht Einschränkungen über SQL-Statements.

Gibt es Alternativen?

Bietet Oracle auf Datenbank-Ebene Möglichkeiten, die Sichten auf Daten einer Table zeilenweise für unterschiedliche User einzuschränken, ohne dass beispielsweise Statement-Änderungen in Applikationen vorzunehmen sind beziehungsweise diese Einschränkungen schon via SQL-Plus greifen. Neben dem Zugriff über Views bietet Oracle mit Virtual Private Database eine Möglichkeit, den Zugriff auf Zeilen-Ebene zu steuern.

Views werden zur Laufzeit aufgebaut und können daher trotz gleichem Statement-Aufbau unterschiedliche Zeilenanzahlen zurückgeben. Die Nutzung von VIEWS ist für die Applikation transparent. Der „User-Context“ wird genutzt, um den angemeldeten User an die VIEW weiterzugeben (local -/ default Context). *Listing 1* zeigt ein Beispiel.

Folgendes Statement ergibt bei unterschiedlichen eingeloggten Usern andere Ergebnisse, User „SCOTT“ *(siehe Listing 2)* und User „SYSTEM“ *(siehe Listing 3)*.

Diese Möglichkeit ist in allen Oracle-Editionen möglich und kostenfrei. Bei diesem Vorgehen kann allerdings die Performance ein Problem darstellen. Zudem sind „insert“, „update“ und „delete“ bei komplexen Views über PL/SQL-Prozeduren/Funktionen zu realisieren.

Virtual Private Database

Neben der Möglichkeit über VIEWS bietet die Oracle Enterprise Edition mit dem Feature „Virtual Private Database“ (VPD) eine weitere interessante Möglichkeit, den Zugriff von Daten auf Zeilenebene einzuschränken. Als zusätzliche Option wird die Label-Security aufgeführt, die allerdings extra lizenziert werden muss.

VPD ergänzt SQL-Statements aufgrund einer Funktion mit einer zusätzlichen Bedingungszeile, die während der Laufzeit an das Statement angehängt wird und die Einschränkung durchsetzt. Die Statements müssen dadurch nicht geändert oder angepasst werden, sondern sind über den Connect beziehungsweise CONTEXT gesteuert. Synonym werden die Begriffe „Row Level“ (RLS) und „Fine grained access control“ (FGAC) verwendet.

Untrennbar mit VPD ist der Begriff „Context“ verbunden. Damit ist eine Anzahl von Variablen bezeichnet, die Informationen zwischen Applikation, Datenbank und Usern abgleichen und setzen. Diese Möglichkeit wurde speziell geschaffen, um VPD mit Drei-Tier-Applikationen zu nutzen, da diese keine Verbindung halten.

```
create or replace view v_daten_current_mitarbeiter
Select * from emp
where ename = SYS_CONTEXT('userenv','current_user');
```

Listing 1

```
select ename, sal from scott.v_daten_current_mitarbeiter;

ENAME             SAL
---------- ----------
SCOTT            3000
```

Listing 2

```
select ename, sal from scott.v_daten_current_mitarbeiter;
Es wurden keine Zeilen ausgewählt
```

Listing 3

```
my_users
USERID     CLASS      DEPTS
---------- ---------- ----------
SYSTEM     ADMIN
SCOTT      DEPTADM    40
BLAKE      DEPTADM    20
MILLER     DEPTADM    30
KING       ADMIN
```

Listing 4

```
(select * from emp where deptno = 40)
CREATE OR REPLACE function benutzerdaten IS
    v_dept   number(6);
    v_class  varchar2(20);
  BEGIN
    select depts, class
    into v_dept, v_class
    from my_users
    where userid = SYS_CONTEXT('USERENV', 'SESSION_USER');

    if v_class = 'ADMIN' then
       return '1=1'
    else
     return 'DEPTNO = '|| v_dept;
    end if;
end;
```

Listing 5

Aufbau einer VPD

VPD ist durch eine Funktion, eine Policy und eine Userverwaltungs-Umgebung definiert. Die Funktion erzeugt einen Rückgabe-String, der das auszuführende SQL-Statement ergänzt. Die Policy hängt die Funktion an die definierte Tabelle, um die zeilenweise Einschränkung der Datenansicht durchzusetzen. Die Userverwaltungs-Umgebung kann zum Beispiel aus einer Tabelle bestehen. Diese enthält die Informationen der Zugriffsrechte. Im folgenden Beispiel soll die Tabelle „my_users" die Sicht auf die Informationen der Mitarbeiter-Tabelle „emp" einschränken *(siehe Listing 4)*.

Die User „SYSTEM" und „KING" sollen alle Daten sehen, alle anderen User jeweils nur die Mitarbeiter-Infos der Abteilung („Depts"), der sie angehören. Um diese Vorgabe durchzusetzen, wird eine Funktion benötigt, die die Infos für die Berechtigungen (Rückgabewert für die Policy) zusammenstellt. Diese Funktion wird mit PL/SQL als „stored procedure/function" erstellt. Sie baut den „Rückgabe-String" auf, der den abgesetzten SQL-Befehl um eine zusätzliche „WHERE"-Bedingung ergänzt. Die Berechtigungskonzepte können beliebig komplex sein *(siehe Listing 5)*. Über eine Policy wird die Funktion „benutzerdaten" an die Tabelle „EMP" gebunden: „SYS.DBMS_RLS.ADD_POLICY" *(siehe Listing 6)*.

Die Policy anlegen

Beim Anlegen der Policy wird diese auch aktiviert („enable=>TRUE"). Über „statements_types" kann definiert werden, wann die Policy greifen soll („SELECT, INSERT, UPDATE, DELETE"). Somit kann ein User keine Zeilen für eine Abteilung anlegen beziehungsweise ändern, wenn ihm über die Policy keine Rechte eingeräumt werden.

Die Einstellung „policy_type=>dbms_rls.dynamic" legt fest, dass die Policy bei jedem Aufruf neu geparst beziehungsweise geprüft wird. Beim Ausführen des Select-Statements („select * from emp;") erhält der User „KING" alle Datensätze der Tabelle „Emp" zurück, während der User „SCOTT" nur die Datensätze der Tabelle „EMP" sieht, die die Abteilungs-Nummer „40" enthalten.

VPD in einer Drei-Tier-Umgebung

Moderne Web-Applikation nutzen für die Datenbank-Verbindung sehr oft einen technischen User („connection-pool"). Die Applikation regelt hier über eine eigene User-Verwaltung die Zugriffe und damit Änderungen von SQL-Statements, um die Einschränkung der Datensicht zu ermöglichen. Diese Applikations-User haben keine echte Verbindung zur Datenbank und können nicht über den „DEFAULT-CONTEXT" ausgelesen werden. Das Statement „SYS_CONTEXT('USERENV', 'SESSION_USER'" ergibt dabei den technischen User. Somit sind diese Applikationen „stateless". Die Connection wird nicht gehalten, sondern über den Connection-Pool realisiert. Wie ist dieses Problem zu lösen?

Oracle bietet die Möglichkeit, einen eigenen „CONTEXT" zu setzen. Diese „CONTEXT"-Variablen werden während der Applikationsausführung von der Policy-Funktion ausgewertet, um die Datensicht einzuschränken. Der „CONTEXT" ist

```
BEGIN
  SYS.DBMS_RLS.ADD_POLICY     (
    ,object_name           => 'EMP'
    ,policy_name           => 'ZUGRIFFSKONTROLLE_EMP'
    ,policy_function       => 'benutzerdaten'
    ,statement_types       => 'SELECT,INSERT,UPDATE,DELETE'
    ,policy_type           => dbms_rls.dynamic
    ,enable                => TRUE );
END;
```

Listing 6

```
CREATE OR REPLACE package body rdb_login_package
is
  procedure set_context(p_userid in varchar2) is
     v_dept number(6);
  begin
     for cl in (select nvl(dept,0) from
      my_users where upper(ename) = upper(p_userid)) loop
        v_dept := cl.id_rdb;
     end loop;
     dbms_session.set_context('rdb_context','ID_DEPT', v_dept);
  end;
```

Listing 7

```
CREATE OR REPLACE function f_dept_policy
(p_schema varchar2, p_object varchar2) return varchar2 is
  v_sql varchar2(32767);
begin
  if sys_context('rdb_context','ID_RDB') = 0 then
     v_sql ='1=1';
   elsif sys_context('rdb_context','ID_RDB') != 0 then
     v_sql:=' depnto =('||sys_context('rdb_context','ID_RDB')||')';
  end if;
return v_sql;
end;
```

Listing 8

im Hinblick auf VPD optimiert, um eine hohe Performance zu erreichen. Der Aufbau eines „CONTEXT“ erfolgt mit: „CREATE OR REPLACE CONTEXT RDB_CONTEXT USING RDB_LOGIN_PACKAGE;“. Das „Create“-Statement legt einen „CONTEXT“ mit dem Namen „RDB_CONTEXT“ an, der nur vom Package „RDB_LOGIN_PACKAGE” geändert werden kann („trusted“). In *Listing 7* wird ein „CONTEXT“ in einem Package erzeugt („dbms_session.set_context“). Er kann beispielsweise von einer Funktion ausgewertet werden, die den Rückgabewert für die Policy liefert *(siehe Listing 8)*.

„SYS.DBMS_RLS.ADD_POLICY“ legt die Policy an. Die aufgebaute Policy-Umgebung kann auf zweierlei Weise genutzt werden:

- Datenbank-User
  über „after-logon-trigger” und „rdb_login_package.set_context('xyz-user');“
- Applikations-User
  Aufruf beispielsweise unter Apex

Das Package wird wie folgt eingebunden: Shared Components -> Security Attributes -> Security/Database Session.

VPD für Spaltenmaskierung

Neben der Durchsetzung zeilenweiser Datensatz-Einschränkung können auch Spalten via VPD maskiert werden („column-level VPD“). Dies wird über die Parameter beim Erstellen der Policy realisiert. Welche Spalten maskiert werden sollen (Spalte „SAL“), wird mit „sec_relevant_cols =>'SAL'“ angegeben. Werte der eingestellten Spalten werden nicht angezeigt, wenn keine Berichtigung über die Policy ermöglicht wurde (etwa „SAL“ bleibt „NULL“) „sec_relevant_cols_opt => DBMS_RLS.ALL_ROWS“. Die Default-Einstellung ist „NULL”. Das heißt, dass Zeilen nicht angezeigt werden, wenn keine Berechtigung über die Policy vorhanden ist und die Spalte angesprochen wird. *Listing 9* zeigt dazu ein Beispiel.

Fazit

Die Nutzung von VPD und „CONTEXT“ kann in unterschiedlichen Umgebungen eingesetzt werden:

- Connection-Pool / PROXY_USER
- LDAP
- Oracle Internet Directory
- Apex-User

Die Nutzung ist dadurch sehr flexibel. VPD ist komplett transparent für jegliche Art von Applikation. Dies betrifft ODBC, OLE, SQL-Plus etc. Der Rechte-Aufbau ist dabei überall gleich und zieht sich durch alle Oracle-Tools (wie „export/import“). Es können nur Daten ausgelesen werden, für die der User über die Policy Berechtigungen hat. Die eingestellten Policies werden konsequent durchgesetzt.

NEWSTICKER

Das neue Flaggschiff der Exadata-Familie

Ein gutes halbes Jahr nach der Vorstellung der ersten Systeme der fünften Exadata-Generation hat Oracle mit dem Modell „Exadata Database Machine X4-8“ das Angebot nach oben vervollständigt. Es ersetzt die bisherige Acht-Sockel-Maschine und richtet sich an Betreiber sehr großer Data-Warehouse-Lösungen, höchst anspruchsvoller OLTP-Systeme oder umfangreicher konsolidierter Oracle-Umgebungen.

```
select name, sal from emp;

NAME       SAL
-----------------
SMITH
MILLER    2000
WARD
JONES
```

Listing 9

Markus Geis
markus.geis@med.uni-muenchen.de

Mathias Weber
mathias.weber@med.uni-muenchen.de

VPD ist einfach umzusetzen, es sind nur PL/SQL-Kenntnisse notwendig. Applikationen bleiben bei Änderungen von Rechtestrukturen aufgrund neuer betrieblicher Anforderungen unverändert. SQL-Statements müssen nicht geändert werden.

VPD nutzt den Oracle-Optimizer (CBO) und baut den „EXPLAIN-PLAN" aufgrund der statistischen Informationen/Histogramme optimal auf. VPD hat daher so gut wie keinen negativen Einfluss auf die Performance einer Applikation. Es entsteht nur ein sehr kleiner Overhead. Einzige Einschränkung: Der User „SYS" umgeht jegliche Policy.

Der Einsatz von VPD ist sehr zu empfehlen, da Sicherheit und Flexibilität durch den Einsatz zunehmen. Applikation werden dadurch zukunftsfähiger, weil bei einer betrieblichen Änderung von Rechte-Strukturen (Abteilungen werden zusammengefasst; andere Sichten sollen ermöglicht werden etc.) keine Anpassungen auf Applikationsseite durchgeführt werden müssen.

Security Guide – Eine Checkliste für den Datenbank-Administrator

Der Datenbank-Administrator trägt aufgrund seiner Tätigkeit eine besondere Verantwortung für den Datenschutz und die Datensicherheit. Oft scheint er sich zwar dessen bewusst, kann aber die Situation häufig nicht richtig einschätzen. Diese Checkliste mit typischen Fallbeispielen soll ihm dabei helfen, ein besseres Gefühl für den Datenschutz zu entwickeln, damit er nicht selbst ins Fadenkreuz der Datenschützer gerät.

In Zusammenarbeit mit den Mitgliedern der DOAG Datenbank Community und dem Competence Center Security – vertreten durch Oliver Pyka, Tilo Metzger und André Lutermann sowie dem Rechtsanwalt Sascha Schoor – ist dieser Ratgeber entstanden. Anhand von praxisbezogenen Fallbeispielen, mit denen jeder Datenbank-Administrator in seinem beruflichen Alltag konfrontiert sein kann, wird das Thema „Datenschutz" und der Umgang mit personenbezogenen Daten beleuchtet, beispielsweise das Bereitstellen von Testdaten, dem Umziehen/Kopieren von Datenbanken, der Anonymisierung von Daten, die Gesetzeslage rund um den Datenschutz sowie mögliche Konsequenzen bei einer Verletzung des Datenschutzgesetzes.

Dieser Ratgeber soll auf keinen Fall Angst verbreiten oder den Datenbank-Administrator einschüchtern. Er soll aber zum Nachdenken anregen, sodass ein sicherer und korrekter Umgang mit personenbezogenen Daten im Unternehmen erfolgt. Er steht für DOAG-Mitglieder unter „http://www.doag.org/pdf/securityguide.php" zum Download bereit.

Tilo Metzger
cc-security@doag.org