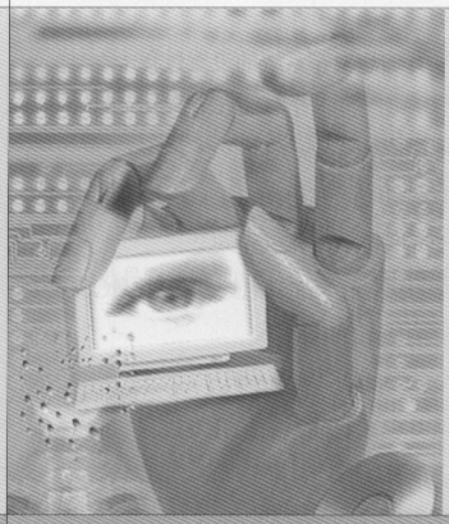

基于Oracle虚拟专用数据库的用户管理的设计

Design of User Management Based on Virtual Private Database of Oracle

李滋堤　宋 音 (北京总后油料研究所自动化室 102300)

摘　要：本文简要介绍了Oracle8i的虚拟专用数据库（Virtual Private Database，VPD），提出了管理信息系统（MIS）用户管理的两个设计模型，详细阐述了应用VPD实现基于“最终用户与数据库用户一一映射”设计模型的用户管理模块的设计方法。

关键词：理信息系统　虚拟专用数据库　细粒度访问控制　应用程序上下文

1 虚拟专用数据库（VPD）

在访问数据库的方式（如在本地、通过Intranet或Internet）日益多样化的今天，仅在应用程序实施安全策略进行访问控制设计是不够的，因为某些软件如SQL*PLUS可以方便地绕过系统的安全机制，直接访问数据库，对数据库造成巨大的安全隐患，如图1所示。因此，为确保数据安全性而在数据库端强制执行“行级”（row-level）的细粒度访问控制变得至关重要。有鉴于此，Oracle数据库在其8.1.5版中引入了虚拟专用数据库的特性，将系统的安全策略（Securiy Policy）直接捆绑在需要保护的数据库表、视图上，用户在访问表、视图中的数据时，自动激活安全策略，对数据实施保护，从而使任何试图绕过安全

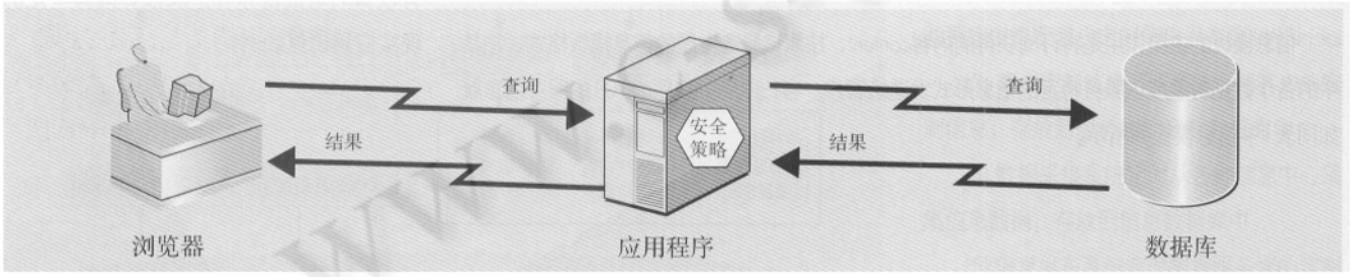

图 1　在应用程序内执行安全策略

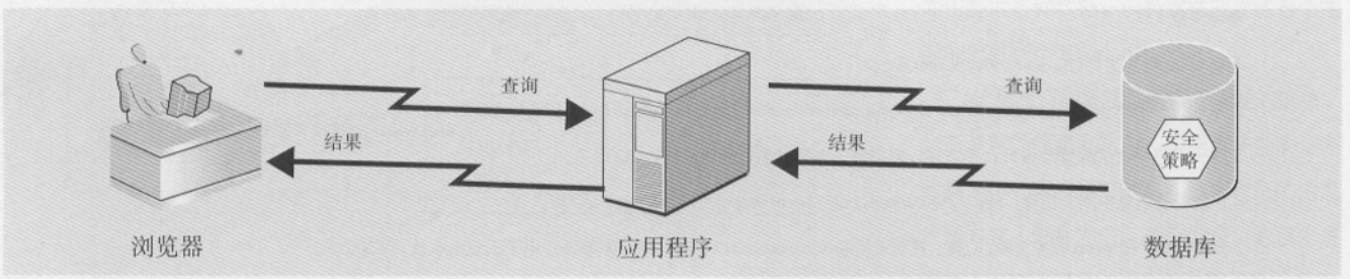

图 2　VPD 允许将应用程序内的安全策略转移到数据库

机制直接访问数据库的努力成为徒劳，如图2所示。这种将数据库根据用户访问权限进行逻辑分割（精确到“行级”粒度），确保用户只能访问与其权限相适应的数据的技术，称为虚拟专用数据库。它实现了虽然物理上只有一个数据库，但对系统最终用户而言，似乎存在着多个专用的数据库。

虚拟专用数据库包括两个组件：细粒度访问控制（Fine-Grained Access Control）和应用程序上下文（Application Context），下面对其进行简要介绍。

1.1 细粒度访问控制

细粒度访问控制是通过SQL语句上的Where子句部分的动态谓词（Dynamic Predicate）而起作用的。动态谓词是动态附加在Where子句上的一个或多个限定条件，即所谓的安全策略。当用户提交的SQL语句查询或操作数据库中的表或视图时，数据库在这些SQL语句执行之前的解析阶段将自动在其Where子句中添加安全策略，实时透明地改写SQL语句，对表或视图中的数据进行保护。

1.2 应用程序上下文

应用程序上下文是内存中的一块暂存区域，用来存储实施安全策略需要的少量的信息，如用户名称、ID号码等信息。这些信息作为应用程序上下文的安全属性被用于数据库访问控制的程序设计，必须通过Oracle内置的专门的程序包才能对其进行设置和调用。应用程序上下文对整个数据库来说是全局的，一旦创建所有用户均能使用其属性。

那么这两个组件是如何协作从而为系统提供安全、一致的访问控制机制的呢？下面以Email信箱的安全设计为例做简要说明。众所周知，Email信箱的基本安全策略是“用户只能查看自己的邮件”。下面假定Email信箱采用Oracle数据库来存储邮件信息，所有邮件信息存储在EMAILS表中，信箱用户同时也是数据库用户。安全策略的实施步骤如下：

(1) 当用户成功登录信箱服务器时，用户名称自动存入到一个Oracle预定义的上下文USERENV的属性SESSION_USER中，同时发出一条查询语句：

```
SELECT * FROM EMAILS
```

(2) Oracle通过细粒度访问控制运行安全策略函数，调用应用程序上下文中的用户名称，生成动态谓词：

```
ID=SYS_CONTEXT('USERENV', 'SESSION_USER')
```

(3) 策略函数自动在查询语句上添加动态谓词从而动态修改用户发出的查询语句：

```
SELECT * FROM EMAILS WHERE ID= ID=SYS_CONTEXT('USERENV',
'SESSION_USER')
```

(4) Oracle运行修改后的查询语句，将该用户的所有邮件返回到客户端。

2 用户管理的两种设计模型

MIS的用户管理是软件安全设计中的核心功能模块，占据着非常重要的地位，对于包含保密信息且应用于广域网的MIS更是如此。目前，基于Oracle等大型关系数据库系统的MIS用户管理设计通常采用两种设计模型。

2.1 最终用户多对一映射到数据库用户

如图3所示，该模型是目前多数MIS的用户管理模块所采用的设计方法，即系统所有最终用户的用户名称和密码等信息存放在一个用户表中，登录时通过比较输入的用户名称/密码与存储在用户表中的用户名称/密码是否相符来进行身份鉴别，通过身份鉴别后，再由一个被高度授权的数据库用户与后台数据库系统建立会话连接，借助该连接实现数据资源的访问。这就是说，系统最终用户与数据库用户是多对一的映射关系，即最终用户不是数据库用户。采用这种用户管理模型的MIS用户认证、访问控制和用户管理等功能都需要在应用程序上通过专门编制的程序模块来实现。前文述及，这种在应用程序上实施的安全策略存在较大安全隐患，因此Oracle推荐采用“最终用户与数据库用户一一映射”用户管理设计模型。

2.2 最终用户与数据库用户一一映射

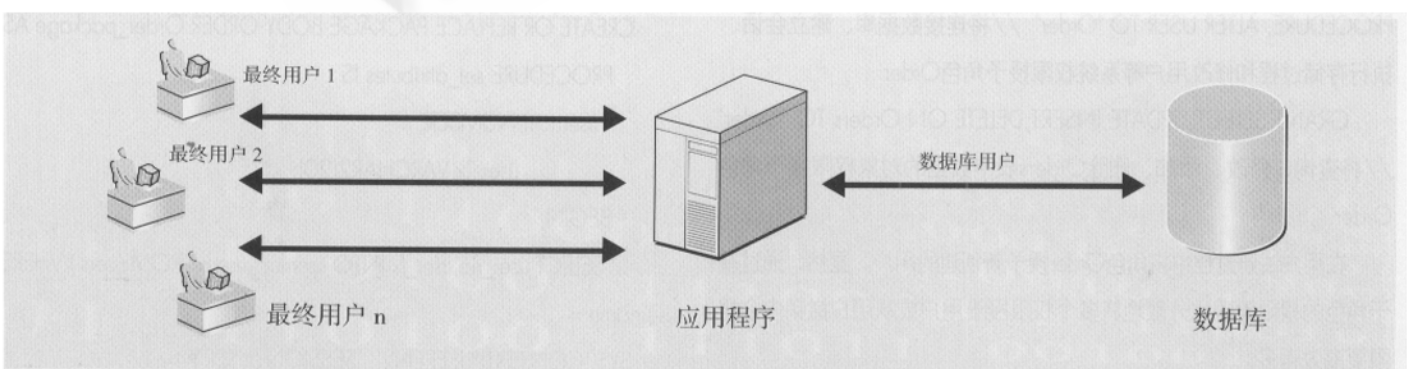

图 3 最终用户多对一映射到数据库用户

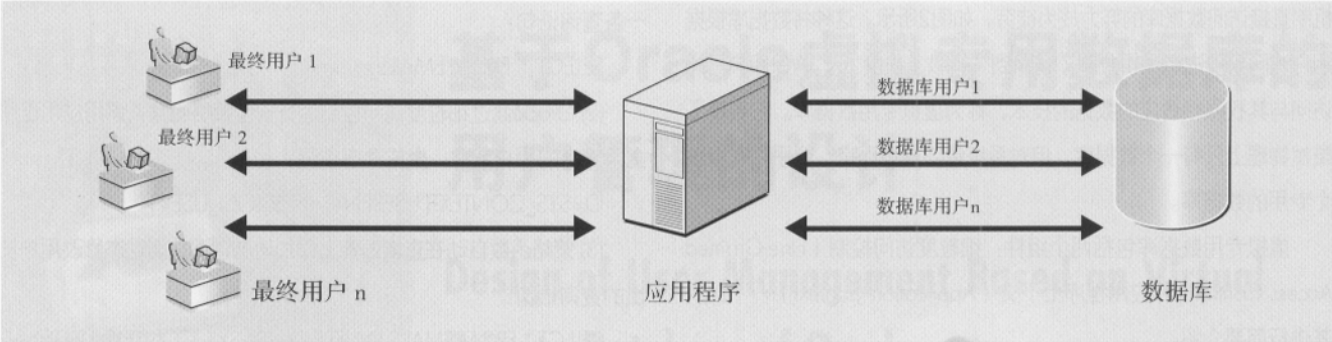

图 4　最终用户与数据库用户一一映射

如图4所示，该用户管理设计模型中系统最终用户同时也是数据库用户，因此能有效利用Oracle等大型数据库内部功能强大的安全机制，如基于角色的用户权限管理、存储过程和安全审计等功能特性。采用“最终用户与数据库用户一一映射”的用户管理设计模型与Oracle VPD安全特性相结合的方法进行用户管理模块设计，将传统的在应用程序中定义和实施的安全策略转移到数据库中，从根本上消除某些软件能绕过应用程序中安全策略而直接访问数据库所带来的安全隐患，因而有效地提高系统的安全性。同时，用户的身份认证、访问控制等以往需要通过编程实现的工作现在由数据库自动实现，因而能显著提高软件开发效率。下面以网上客户订单管理为例，介绍基于Oracle VPD的用户管理的设计方法。

3 基于 Oracle VPD 的用户管理的设计

3.1 创建 Order 角色

角色是一组相关的权限集合，Oracle通过角色（Role）来管理数据库系统权限（如创建数据库表等）和对象权限（如访问数据表等）。客户订单的权限管理是通过Order角色来实现的，创建该角色时，为其授予了连接数据库、操作数据库表等多种必要的权限。

```
CREATE ROLE "Order" NOT IDENTIFIED  //创建Order角色，无密码校验
GRANT CONNECT, CREATE SESSION, EXECUTE ANY PROCEDURE, ALTER USER TO "Order"  //将连接数据库、建立会话、执行存储过程和修改用户等系统权限授予角色Order
GRANT SELECT,UPDATE,INSERT,DELETE ON Orders TO "Order"  //将查询、修改、添加、删除Orders表中数据的对象权限授予角色Order
```

在用户注册过程中将角色Order授予新创建的用户。显然，通过基于角色的权限管理比分散地将多个权限授予用户或从用户撤销多个权限要高效得多。

```
CREATE USER "JOHN" IDENTIFIED BY "SMITH"  //创建用户JOHN，密码为SMITH
GRANT "Order" TO "JOHN"  //将角色Order授予用户JOHN
```

3.2 应用 VPD 的准备工作

3.2.1 创建应用程序上下文

客户订单管理的基本安全策略是“客户只能查看自己的订单，店员可以查看所有客户的订单”。针对该安全策略需要创建一个包含用户ID号和用户类型（如客户、店员等）属性的应用程序上下文Order_context，它是基于会话的，只能由合法连接到数据库的用户通过被信任的程序包Order_package来设置用于安全设计的属性，因而可有效防止非法用户对属性值的恶意修改。在会话结束之前，除非用户重新设置属性，否则上下文中设置的属性值将保持不变。

```
CREATE OR REPLACE CONTEXT ORDER.Order_context USING ORDER.Order_package  //创建上下文
```

3.2.2 设置应用程序上下文

应用PL/SQL来创建程序包Order_package，其功能是从用户表中获取用户ID号和用户类型信息以设置上下文属性用户ID号（USER_ID）和用户类型（USER_TP），这两项属性将在下面创建的策略函数中使用。

```
CREATE OR REPLACE PACKAGE ORDER.Order_package AS
  PROCEDURE set_attributes ;
END;  //创建包头
CREATE OR REPLACE PACKAGE BODY ORDER.Order_package AS
  PROCEDURE set_attributes IS
   usernum NUMBER;
        user_lx VARCHAR2(20);
  BEGIN
   SELECT user_no,user_lx INTO usernum,userlx FROM users WHERE username =
SYS_CONTEXT('USERENV', 'SESSION_USER');
DBMS_SESSION.SET_CONTEXT('Order_context', 'USER_ID',
```

```
usernum);
    DBMS_SESSION.SET_CONTEXT('Order_context', 'USER_TP', userlx);
      END set_attributes;
    END; //创建包体
```

3.3 实现细粒度访问控制

3.3.1 创建用户登录触发器

设置上下文初始属性时必须考虑的一个问题是设置的时机，无论用户如何连接数据库，在用户成功登录数据库的同时进行属性设置显然是一个最恰当的时机。由于Oracle8i引入了登录触发器的新特性，通过登录触发器进行初始属性设置成为最好的选择。下面创建登录触发器user_logon，该触发器调用上面创建的程序包Order_package的过程set_attributes来设置上下文的属性。

```
CREATE OR REPLACE TRIGGER user_logon AFTER LOGON ON
DATABASE CALL ORDER.Order_package.set_attributes
```

3.3.2 创建安全策略函数

订单管理的安全策略包括两部分：一是客户只能查看自己的订单；二是店员可以查看所有的订单。对于客户发出的查询（包括添加、修改和删除）语句，需要通过策略函数生成动态谓词进行限制，以确保只能查看自己的订单；对于店员发出的查询，则不需要进行查询限制。采用PL/SQL编写的查询策略函数如下：

```
CREATE OR REPLACE PACKAGE ORDER.Order_policy IS
    FUNCTION select_limits (object_schema IN VARCHAR2,
object_name IN VARCHAR2) RETURN VARCHAR2;
END Order_policy; //安全策略包头
CREATE OR REPLACE PACKAGE BODY ORDER.Order_policy IS
    FUNCTION select_limits (object_schema IN VARCHAR2,
object_name IN VARCHAR2) RETURN VARCHAR2
    IS
      predicate VARCHAR2(200);
    BEGIN
      IF SYS_CONTEXT('Order_context','USER_TP')= '0' THEN
//如果用户类型为客户
            predicate := 'user_id=SYS_CONTEXT
("Order_context", 'USER_ID")' ; //动态谓词为用户ID号码
          ELSIF SYS_CONTEXT('Order_context','USER_TP')= '1'
THEN //如果用户类型为店员
            predicate := ' '; //动态谓词为空字符串，可
以查看所有数据
          END IF ;
      RETURN predicate;
    END;
END Order_policy; //安全策略包头
```

3.3.3 为Orders表添加安全策略

应用Oracle提供的PL/SQL程序包DBMS_RLS.ADD_POLICY为需要保护的数据库表Orders添加安全策略。

```
EXECUTE DBMS_RLS.ADD_POLICY ('ORDER','Orders','Order_select',
'ORDER','Order_policy.select_limits', 'SELECT', TRUE)
```

现在任何客户发出的订单查询语句只能返回自己的订单，换句话说，动态谓词能将查询语句：

```
SELECT * FROM Orders
```

动态修改为：

```
SELECT * FROM Orders WHERE user_id=SYS_CONTEXT
('Order_context', 'USER_ID')
```

对于店员发出的订单查询语句则不作任何修改。

4 结束语

采用“最终用户与数据库用户一一映射”的用户管理设计模型，结合基于“行级”安全策略的虚拟专用数据库的用户管理设计方法，不仅能显著提高MIS软件的开发效率，而且能够利用Oracle等大型数据库系统强大的安全机制，增加MIS的安全性，是一种值得推广的设计方法。