管理安全性

——用户与权限管理

单世民

大连理工大学

用户与权限管理概述

- 对于任意一个多用户计算机系统来说，访问和访问安全都是至关重要的。既要允许很多用户访问计算机系统，又要防止未授权的用户访问。

- Oracle提供了三种用户认证方法：
  - **口令认证**
  - 操作系统认证
  - 全局认证

用户管理

- 建立用户帐号

```
CREATE USER username
    IDENTIFIED BY password
    [DEFAULT TABLESPACE default_tablespace]
    [TEMPORARY TABLESPACE temp_tablespace]
    [PASSWORD EXPIRE]
    [ACCOUNT {LOCK|UNLOCK}]
```

```
CREATE USER Michael
       IDENTIFIED BY lincoln
```

用户管理

- 用户与模式
  在Oracle中，数据库用户和模式是安全的最基本的单元。术语“用户”和“模式”经常互换使用，然而它们是有区别的：
  - 数据库模式定义为数据库对象的集合。
  - 模式的名称就是拥有或控制这些数据库对象集合的用户名称。

- **所有的数据库对象**，包括表、视图、索引、触发器、Java存储过程、PL/SQL程序包、函数等，**都归Oracle数据库中的某一个用户所有**。甚至Oracle的数据字典、系统目录也是名称为sys的模式的一部分。

- 在Oracle数据库中，可以存在没有拥有任何数据库对象的用户(不是模式)，但是不会没有命名的模式或数据库对象集合。

用户管理

- 修改用户帐号口令

```
ALTER USER username IDENTIFIED BY password
```

- 删除用户帐号

```
DROP USER username [CASCADE]
```

如果将Cascade关键字用于drop user命令的末尾，则在从数据库中删除用户之前，删除用户的所有对象。该关键字不仅可以删除所有的用户对象，而且还可以删除其他用户模式中对已删除对象表进行引用的约束，使其他用户所拥有的引用了已删除对象的对象无效。

用户管理

- 用户的默认表空间与临时表空间

```
ALTER USER username
DEFAULT TABLESPACE default_tablespace
TEMPORARY TABLESPACE temp_tablespace;
```

如果不进行指定，则用户的默认表空间为USERS表空间；默认的临时表空间是TEMP表空间，如果没有创建TEMP表空间，则SYSTEM表空间为用户临时表空间

用户管理

- 锁定和解锁用户帐号
  被锁定的帐号不能进行数据库访问操作

```
ALTER USER username ACCOUNT {LOCK|UNLOCK};
```

为什么要锁定用户帐号而不是将其删除？

用户帐号关联的模式中含有需要保留的表或其他对象；使用帐号对应的模式作为某应用程序使用的数据库对象的一种组织方式；其他一些原因

用户管理

- 修改用户的磁盘空间配额

```
ALTER USER username
DEFAULT TABLESPACE default_tablespace
QUOTA nn on default_tablespace
```

用户管理

- 默认的数据库用户
  每个Oracle数据库都有两个默认的数据库用户帐号SYS和SYSTEM

  - SYS帐号拥有数据库数据字典对象，除非需要安装属于SYS的额外的数据字典对象，否则不应使用SYS进行数据库操作

  - SYSTEM帐号是默认的数据库管理员帐号，可以使用此帐号启动一个新的数据库

权限管理

- 除非用户具有执行特定的数据库操作权限，否则，用户既不能与数据库服务器连接，也不能做任何事情

- 例如：
  - 除非用户具有CREATE SESSION系统权限，否则用户不能与Oracle数据库连接
  - 除非用户具有CREATE TABLE系统权限，否则用户不能在自己的模式中创建表

权限管理

- 在Oracle数据库中，有两类权限：对象权限和系统权限。
  - 对象级别权限是由用户赋予的访问或操作数据库对象的权限。例如，如果需要向scott.emp表中插入行的数据库用户必须拥有完成这项工作的指定权限。
  - 系统权限不是控制对指定数据库对象的访问，而是用来许可对各种特性的访问，或许可Oracle数据库中的特定任务。

权限管理

- 数据库权限的类型--系统权限

  系统权限（System Privilege）向用户提供了执行某一种或某一类型的数据库操作的能力，有近100种系统权限。

  系统权限不是控制对指定数据库对象的访问，而是用来许可对各种特性的访问，或许可Oracle数据库中的特定任务。

权限管理

- 系统权限的授予和撤销

```
GRANT {sys_priv_1[,sys_priv_2]...|ALL[PRIVILEGES]}
    TO {user_1[,user_2]...|PUBLIC}
    [WITH ADMIN OPTION];
```

```
REVOKE {sys_priv_1[,sys_priv_2]...|ALL[PRIVILEGES]}
  FROM {user_1[,user_2]...|PUBLIC};
```

权限管理

- 使用系统权限时，需要注意以下几点：
  - 一般情况下，都应该将CREATE SESSION权限授予用户
  - 用户需要CREATE TABLE权限来在自己的模式中创建、修改、删除或查询任何表
  - 如果要删除其他模式中的表，用户必须具有DROP ANY TABLE系统权限
  - CREATE ANY PROCEDURE允许用户创建、修改、删除或执行任何存储过程、程序包和函数
  - 开发人员一般需要几个系统权限，包括CREATE TABLE,CREATE VIEW,CREATE TYPE等，以创建支持前台应用程序的数据库模式

权限管理

- 数据库权限的类型--对象权限
  对象权限控制用户是否能在特定数据库对象（如表、视图或存储过程）上执行特定类型的操作

对象权限	适用对象	允许的操作
SELECT	表、视图、序列	查询
UPDATE	表、视图或其中的字段	更新
DELETE	表和视图	删除行
INSERT	表、视图或其中的字段	插入行
EXECUTE	存储过程，存储函数与程序包	执行PL/SQL存储对象
READ	目录	读取目录
INDEX	表	在表上建立索引
REFERENCES	表或其中字段	在其他表中创建的外键能引用表或表中的字段
ALTER	表或序列	修改表或序列的结构

权限管理

- 对象权限的授予和撤销

```
GRANT {obj_priv_1[,obj_priv_2]...|ALL[PRIVILEGES]}
   ON {[schema.]object[(column1[,column2])]|DIRECTORY dir}
   TO {user_1[,user_2]...|PUBLIC}
   [WITH GRANT OPTION]
   [WITH HIERARCHY OPTION];
```

```
REVOKE {obj_priv_1[,obj_priv_2]...|ALL[PRIVILEGES]}
    ON {[schema.]object[(column1[,column2])]|DIRECTORY dir}
  FROM {user_1[,user_2]...|PUBLIC}
  [CASCADE CONSTRAINTS]
  [FORCE];
```

权限管理

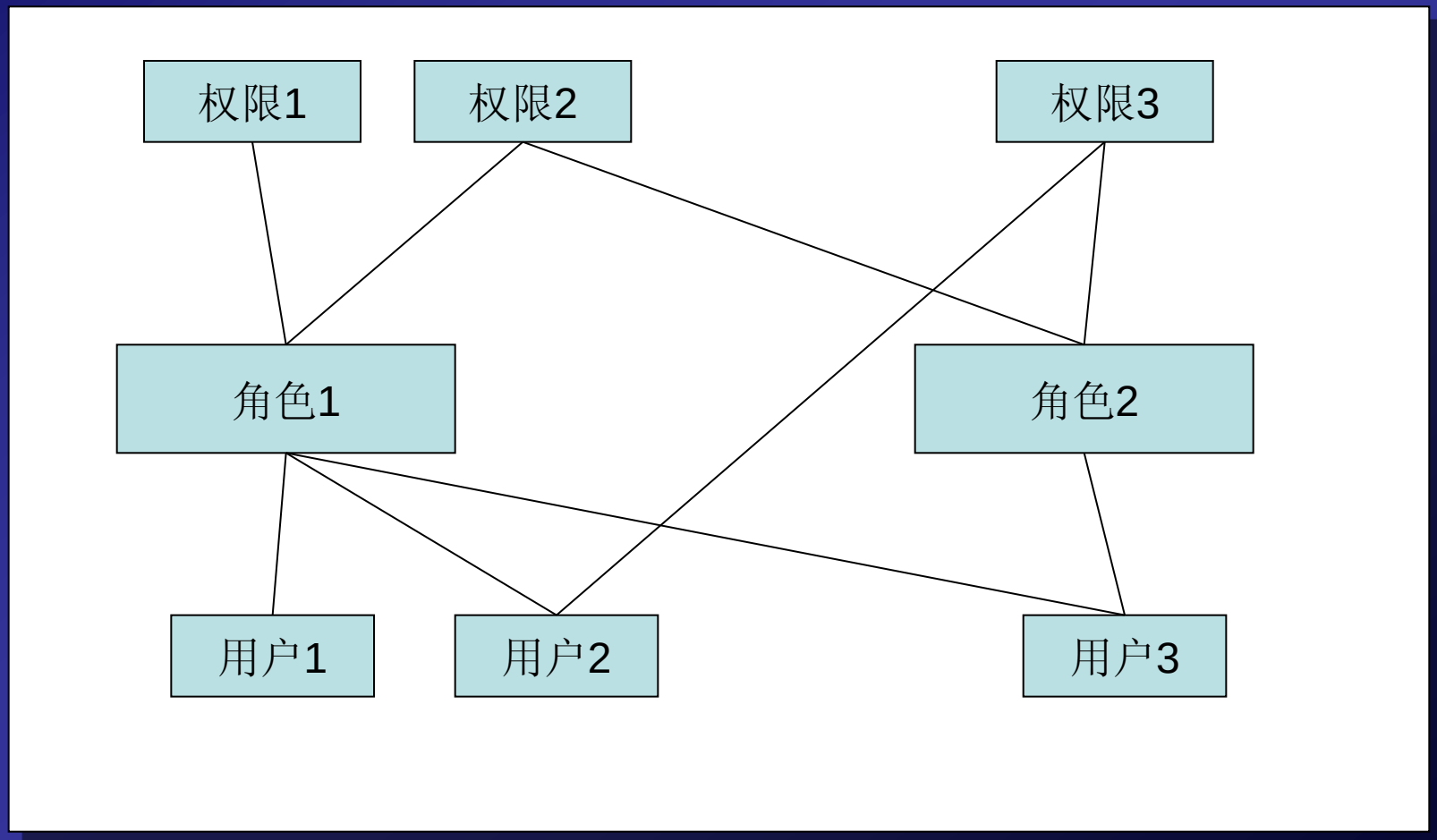
- 使用对象权限时需要注意以下几个问题：
  - 如果一个视图引用了其他模式中的表或视图，则该视图的拥有者必须以WITH GRANT OPTION方式获得这些表或视图的权限，才能将该视图的对象权限授予其他用户
  - 用户只在具有某个表的所有权限时，才能锁定这个表
  - SELECT对象权限只能授予整个表而不能授予表中的字段

角色与权限管理

- **数据库角色就是权限的命名集合**。使用角色可以大大降低用户权限的维护负担。角色可以是对象权限或系统权限的命名集合。数据库管理员只需创建特定的数据库角色，使其反映组织或应用的安全权限，就可以将这些角色赋予用户。

角色与权限管理

- 利用角色进行权限管理



角色与权限管理

- 创建角色

```
CREATE ROLE role_name;
```

- 删除角色

```
DROP ROLE role;
```

角色与权限管理

- 将系统权限授予角色

```
GRANT {sys_priv_1[,sys_priv_2]...|ALL[PRIVILEGES]}
TO role_1[,role_2]...
[WITH ADMIN OPTION]
[WITH HIERARCHY OPTION];
```

- 将对象权限授予角色

```
GRANT {obj_priv_1[,obj_priv_2]...|ALL[PRIVILEGES]}
ON {[schema.]object[(column1[,column2])]|DIRECTORY dir}
TO {role_1[,role_2]...|PUBLIC}
[WITH GRANT OPTION]
[WITH HIERARCHY OPTION];
```

角色与权限管理

- 撤销角色的系统权限

```
REVOKE {sys_priv_1[,sys_priv_2]...|ALL[PRIVILEGES]}
FROM {role_1[,role_2]...};
```

- 撤销角色的对象权限

```
REVOKE {obj_priv_1[,obj_priv_2]...|ALL[PRIVILEGES]}
  ON {[schema.]object[(column1[,column2])]|DIRECTORY dir}
FROM {user_1[,user_2]...}
[CASCADE CONSTRAINTS]
[FORCE];
```

角色与权限管理

- 将角色授予用户或其他角色

```
GRANT role_1[,role_2]...
TO {user_1[,user_2]...|PUBLIC|role_1[,role_2]}
[WITH ADMIN OPTION];
```

- 撤销授予用户或其他角色的角色

```
REVOKE role_1[,role_2]...
FROM {user_1[,user_2]...|PUBLIC|role_1[,role_2]};
```

PL/SQL与角色

- 默认情况下，PL/SQL函数、过程、程序包都要使用“定义者”的命名空间和权限执行。需要注意的是，这些**已编译的“PL/SQL程序”对象要使用直接赋予设计用户的权限执行，而不使用用户通过数据库角色得到的对象权限来执行**。当然，这将使让作为特定数据库用户在SQL*Plus中测试特定DML语句的开发人员混淆，他们将发现PL/SQL过程中的相同语句不能够编译。**大多数情况下，造成这种问题的原因是由于对象权限是通过角色授予，而不是直接授予设计用户的。**

PL/SQL与角色

- **通常情况下，应用开发的安全方法是不直接将表和视图上的权限赋予数据库用户，只能够通过PL/SQL过程、函数或程序包访问附属的表和视图，而执行这些已编译对象的权限要通过数据库角色提供。**这种方式的最大优点是：除了通过公开的方法之外，终端用户不能直接操作应用程序中的表和视图。如果将安全直接绑定到用户应用中，而不是数据库中，就意味着安全实现只能够适用于用户应用。然而，如果用户只能够提供通过PL/SQL过程、函数和程序包修改用户应用的表和视图的能力，那么就要为访问Oracle数据库的所有应用维护数据库对象的安全。

小结

- 用户管理
- 权限的基本作用
- 系统权限（with admin option）
- 对象权限（with grant option）
- 角色（with admin option）
- PL/SQL与角色

The End