第9章 数据库安全与审计

创建和实施安全过程有助于保护公司最重要的财富 —— 数据。当存储在数据库的这些数据变得对公司更有用和更有效时，它就更容易受到非法访问。必须能预防和发现这些非法访问企图。

Oracle数据库设有多个安全层，并且可以对各层进行审计。在本章中，可以看到对审计过程中各层的描述，也能看到设置口令和使其失效的方法。

9.1 安全性能

Oracle把数据库管理员可用的安全功能分为以下几个级别：

- 合法用户的帐户安全性。
- 数据库对象的访问安全性。
- 管理全局权限的系统级安全性。

上面这些性能将在以下各节进行描述，本章 9.2节“ 实现安全性 ”将详细介绍使用这些有效选项的情况。

9.1.1 帐户安全性

若要访问一个Oracle数据库中的数据，必须先访问该数据库的一个帐户。这个访问可以是直接访问 —— 通过到一个数据库的用户连接 —— 或间接访问。间接连接访问包括通过在数据库链接中预设权限的访问。每个帐户必须有一个与其相关的口令。一个数据库帐户也可以连接到一个操作系统帐户上。

口令是在创建用户帐户时为每一用户设置的，并可在该帐户创建后对它们进行变更。用户变更帐户口令的能力受他(她)访问工具权限的限制。数据库以加密的形式将口令存储在一个数据字典表中。如果帐户直接与操作系统帐户相关，就可以旁路口令检查。

在Oracle8中，口令可以无效。数据库管理员可以建立能重复使用口令的条件 (通过一个数据库口令历史设置值)。而且，可以使用环境文件为口令制定标准 (例如最小长度)，如果连续多次与帐户连接都不成功，就可以自动锁定帐户。

9.1.2 对象权限

对数据库中一个对象的访问是通过权限 (privilege)来完成的。通过grant命令就可以针对特定的数据库对象使用特定的数据库命令。例如，如果用户 THUMPER拥有一个叫作EMPLOYEE的表并执行命令

```
grant select on EMPLOYEE to PUBLIC;
```

那么全部用户 (PUBLIC)就可以从 THUMPER的EMPLOYEE表中选择记录。可以创建角色 (role) —— 即权限组 —— 来简化权限的管理。对于有大量用户的应用程序，角色将大大减少grant命令的使用量。由于角色可以用口令保护且能被动态地允许和禁止，所以它们给数据库

增加了一个附加安全层。

9.1.3 系统级角色和权限

可以使用角色来管理对用户有效的系统级命令。这些命令包括 create table和alter index。对于每种数据库对象的操作都是由各自的权限授权的。例如，一个用户可以被授予 CREATE TABLE权限，而不是CREATE TYPE权限。可以创建一些自定义的系统级角色，这些角色只向用户授予他们在数据库中所需的权限而不是过多的特权。如第 5章所述，CONNECT和RESOURCE角色分别是最终用户和开发人员所要求的基本系统权限。

具有RESOURCE角色的用户还被授予UNLIMITED TABLESPACE系统权限，使他们在数据库的任何地方都能创建对象。由于有这个附加权限，所以应限制把 RESOURCE角色用于开发和测试环境。

9.2 实现安全性

Oracle中的安全性能包括角色、环境文件和直接授权。 Oracle Enterprise Manager工具集提供了一个Security Manager 工具来管理用户帐户、角色、权限和环境文件。在以下各节中你将看到这些功能部件的使用，其中包括一些非文档性能。

9.2.1 操作系统安全性

若要访问一个数据库，必须首先能够以直接或非直接方式访问正在运行该数据库的服务器。要使数据库安全，首先要使其所在的平台和网络安全。然后就要考虑操作系统的安全性。

Oracle使用大量用户不需要直接访问的文件。例如，数据文件和联机重做日志文件只能通过Oracle的后台进程进行读写。因此，只有要创建和删除这些文件的数据库管理员才需要在操作系统级直接访问它们。导出转储文件和其他备份文件也必须受到保护。

可以把数据复制到其他数据库上，或者是作为复制模式的一部分，或者是提供一个开发数据库。若要保护数据的安全，就要对数据所驻留的每一个数据库及这些数据库的备份进行保护。如果某人能从含有你的数据备份的数据库中带走备份磁带，那么你在数据库中所做的全部保密工作就失去意义。必须防止对全部数据备份的非法访问。

9.2.2 创建用户

创建用户的目标是，建立一个安全、有用的帐户，并且这个帐户要有充分的权限和正确的缺省设置值。可以使用 create user命令来创建一个新的数据库帐户。该帐户创建后，在授权前它没有任何效力，用户甚至不能注册。

所有用户帐户所需的设置值都可以在一个 create user命令中指定。这些设置包括表 9-1列出的所有参数。

表9-1 create user命令的参数

参 数	使 用
Username	用户名
Password	帐户的口令，也可以直接与操作系统主帐户名相连，或者通过一个网络验证服务验证。对于基于主机的验证，使用 identified externally命令。对于基于网络的验证，使用 indentified globally as命令

(续)

参 数	使 用
Default tablespace	缺省表空间用来存储在该模式下创建的对象。这个设置并不授予用户创建对象的权力；而只是设置一个缺省值
Temporary tablespace	这个表空间只用来存储排序事务处理时所用的临时段
Quota[on tablespace]	用户在规定的表空间存储对象，最多可达到这个定额规定的总尺寸
Profile	赋予用户一个环境文件。如果没有指定环境文件，就使用缺省环境文件。环境文件用于限制对系统资源的使用和执行口令管理规则
Default Role	设置缺省角色供用户使用

下面列出一个样例create user命令。在这个例子中，创建一个叫作THUMPER的用户，口令为RABBIT，缺省表空间为USERS，临时表空间为TEMP。没有定额，使用缺省环境文件。

```
create user THUMPER
identified by RABBIT
default tablespace USERS
temporary tablespace TEMP;
```

由于没有指定环境文件，所以必须使用数据库的缺省环境文件。实际的环境文件名为DEFAULT；其初始设置值是设置所有资源消耗的限值为 UNLIMITED。有关环境文件详细情况，请参见本章9.2.5节“用户环境文件”。

由于没有指定定额，所以用户不能在数据库中创建对象。

如下例所示，若要授予资源定额，需要使用 create user或alter user命令的quota参数。在这个例子中，授予THUMPER的定额为USERS表空间中的100MB。

```
alter user THUMPER
quota 100M on USERS;
```

用户THUMPER现在可以在USERS表空间中创建最多100MB的数据段。

注意 用户不需要TEMP表空间中的空间定额来为他们的查询创建临时段。

除用户名外，create user命令中的全部参数都可以由 alter user命令来更改。

可以从OEM Security Manager(OEM 安全管理器)屏幕创建一个新用户或创建一个“像”其他用户那样的用户。这个功能部件使你能创建一个具有与现有用户相同属性的新用户。图9-1示出了初始Security Manager屏幕，屏幕上选中了用户 THUMPER。创建该用户后，可以利用OEM工具授予系统权限、对象权限和定额。在图 9-1的第一个用户屏幕上，口令验证激活特定口令的标识——帐户为全局帐户(用于远程数据库管理)或帐户在外部被识别的标识。这里有使口令提前失效的选项且可以把帐户创建成锁定或解锁的帐户 (有关口令失效和帐户锁定的情况，见本章后面9.2.6节“口令管理”)。

通过选择General选项卡并单击右鼠标按钮，或者在选定屏幕的 User区域时从Object菜单选中Create选项，就可以创建一个新用户。当选中 Create或Create Like选项时，User Creation Wizard(用户创建向导)被激活，这时就可以填充创建用户所需要的角色、权限等。在缺省情况下，把CONNECT角色赋予新用户且为用户的缺省表空间和临时表空间输入 SYSTEM表空间。图9-2示出了Create User窗口，窗口中有一些用于创建一个像用户 THUMPER那样的新用户的选项。由于把THUMPER的缺省表空间选择为USERS且其临时表空间被指定为TEMP，在缺省情况下，新用户有和 THUMPER一样的授权和表空间设置。创建新用户需要填充的信息是用户的名称、口令和与THUMPER的信息不同的任何其他信息。

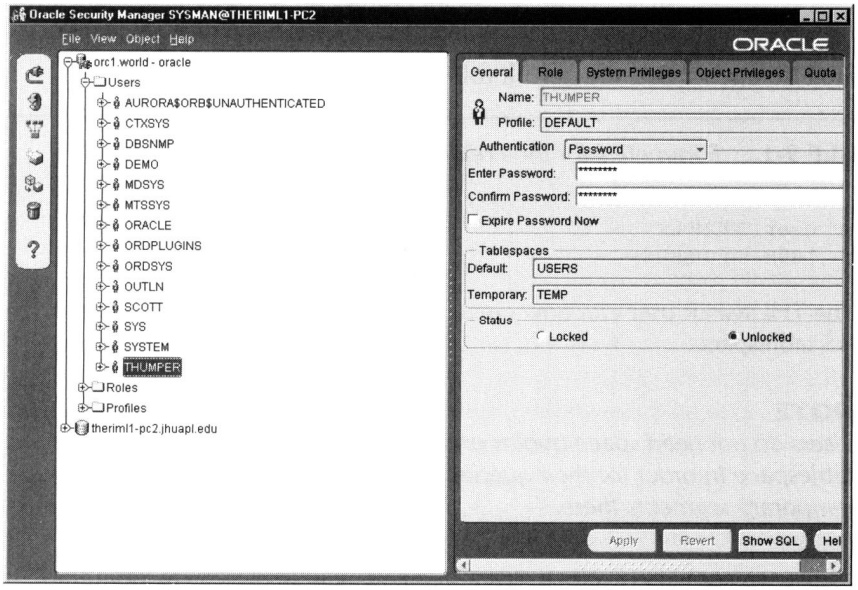

图9-1 口令验证

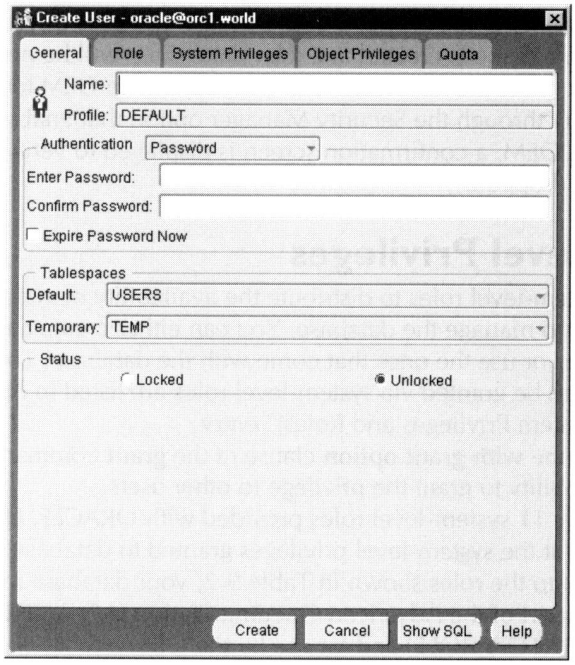

图9-2 Create User窗口，General选项卡

9.2.3 撤消用户

可以用drop user命令从数据库中撤消一个用户。这个命令只有一个参数——cascade，在撤消该用户之前，它撤消用户模式中的所有对象。如果用户拥有对象，就必须指定 cascade以撤消用户。下面示出了一个样本drop user命令：

```
drop user THUMPER cascade;
```

引用被撤消的用户模式中对象的任何视图、同义词、过程、函数或数据包都被标上INVALID。如果以后用相同的名字创建另一个用户，他从具有相同名称的前任用户继承不到任何东西。OEM工具通过Security Manager选项提供一种remove能力(该选项允许撤消用户)。在OEM中，显示一个确认屏幕，以检验是否真的要撤消用户。

9.2.4 系统级权限

可以使用系统级角色分派以管理数据库的系统级命令。可以创建自定义的系统级角色或使用数据库自带的角色。可通过系统级授予的权限列在附录 A的“GRANT system_priviliges _and_roles”条目下面。

可以用grant命令的with grant option子句把向其他用户授权的能力传递给被授予者。

表9-2列出了Oracle提供的11个系统级角色。使用这些角色就能对授予数据库管理角色的系统级权限进行限制。除了表 9-2所示的角色外，数据库还可能包含有为支持 Advanced Queuing Option 而生成的角色(AQ_USER_ROLE和AQ_ADMINISTRATOR_ROLE)和供EM Intelligent Agents使用的角色(SNMPAGENT 角色)。

表9-2 Oracle8i提供的系统级角色

角 色	授予角色的权限
CONNECT	ALTER SESSION、CREATE CLUSTER、CREATE DATABASE LINK、CREATE SEQUENCE、CREATE SESSION、CREATE SYNONYM、CREATE TABLE、CREATE VIEW
RESOURCE	CREATE CLUSTER、CREATE PROCEDURE、CREATE SEQUENCE、CREATE TABLE、CREATE TRIGGER
DBA	全部系统权限WITH ADMIN OPTION
EXP_FULL_DATABASE	SELECT ANY TABLE、BACKUP ANY TABLE、INSERT、DELETE、AND UPDATE ON THE TABLES SYS.INCVID、SYS.INCFIL、AND SYS.INCEXP
IMP_FULL_DATABASE	BECOME USER
DELETE_CATALOG_ROLE	所有字典软件包上的DELETE权限
EXECUTE_CATALOG_ROLE	所有字典软件包上的EXECUTE权限
SELECT_CATALOG_ROLE	所有类别表和视图上的SELECT权限
CREATE TYPE	CREATE TYPE、EXECUTE、EXECUTE ANY TYPE、ADMIN OPTION、GRANT OPTION
RECOVERY_CATALOG_OWNER	DROP ROLE RECOVERY_CATALOG_OWNER、CREATE ROLE、RECOVERY_CATALOG_ OWNER、CREATE TRIGGER、CREATE PROCEDURE TO RECOVERY_CATALOG_OWNER
HS_ADMIN_ROLE	HS_EXTERNAL_OBJECT、HS_EXTERNAL_USER

注意 除了表9-2列出的权限外，DBA和RESOURCE角色的用户还接受UNLIMITED TABLESPACE系统权限。

CONNECT角色一般是授予最终用户。尽管它具有一些创建对象的能力 (其中包括CREATE TABLE权限)，但它未给用户任何表空间的定额。由于用户没有表空间定额 (除非授

予他们这个定额)，所以也就不能创建表。

RESOURCE角色是授予开发人员的。如第5章所述，RESOURCE角色给予开发人员最适合应用程序开发的权限。

DBA角色包括了所有的系统级权限，并具有将这些权限授予其他用户的选项。

当执行整个数据库的导入或导出操作时 (见第10章)，在导出和导入期间分别使用IMP_FULL_DATABASE和EXP_FULL_DATABASE角色。这些角色是DBA角色的一部分，可以使用这些角色授予用户有限的数据库管理权限。

SELECT_CATALOG_ROLE、EXECUTE_CATALOG_ROLE和DELETE_CATALOG_ROLE角色是Oracle8中新增加的角色。

DELETE_CATALOG_ROLE是这三种角色中最简单的角色。如果授予用户这个角色，该用户就可以从SYS.AUD$表中删除记录。SYS.AUD$表是写审计记录的表。给一个用户授予这个角色，该用户不再需要其他DBA级命令就能从审计跟踪表中删除记录。使用这种角色可以简化审计跟踪管理进程。

SELECT_CATALOG_ROLE和EXECUTE_CATALOG_ROLE角色授予用户选择或执行可导出的数据字典对象的权限。即，不是每个数据库对象在整个系统导出时都要被导出 (见第10章)，例如动态性能视图(见第6章)就不导出。因此，SELECT_CATALOG_ROLE不给予用户从动态性能表中选择的能力，而是给予用户从大多数数据字典中查询的能力。同样，EXECUTE_CATALOG_ROLE给予用户执行数据字典中部分过程和函数的能力。

如果使用Oracle8中的Object Option(对象选项)，就可以激活CREATE TYPE角色。拥有CREATE TYPE角色的用户可以创建抽象数据类型。

如果系统级权限和角色有效，就可能要重新检查帐户创建过程。与数据库备份进程一样，需要DBA级权限来执行帐户创建。不过，可以选择一个创建新用户所需要的权限子集。

例如，可以创建一个新的系统级角色，叫做 ACCOUNT_CREATOR。它只能创建用户，而不能执行任何其他DBA级命令。创建这个角色的命令如下所示：

```
create role ACCOUNT_CREATOR;
grant CREATE SESSION, CREATE USER, ALTER USER
   to ACCOUNT_CREATOR;
```

这里列出的第一个命令，创建一个叫 ACCOUNT_CREATOR的角色。第二个命令授权这个用户注册(CREATE SESSION)、创建和修改帐户(CREATE USER和ALTER USER)的能力。这样ACCOUNT_CREATOR被授权成一个能够处理所有新帐户的创建的解色。通过选择Create Role(创建角色)选项并填充合适的信息，就可以使用 OEM创建这个角色。图9-3示出了使用OEM Security Manager工具创建的ACCOUNT_CREATOR角色，而图9-4显示给该角色授予权限。

集中帐户创建有助于在申请帐户时确保合适的授权过程。在这种情况下，系统级权限和角色的灵活性允许把这种能力给予一个用户，而不给这个用户从数据库查询数据的能力。

在进行软件封装时，创建 ACCOUNT_CREATOR角色的能力特别有用。许多第三方封装应用程序都假设它们具有数据库中的全部 DBA权限，而实际上只需要执行 create user和alter user命令的能力。通过创建一个ACCOUNT_CREATOR角色，就可以限制包模式拥有者在数据库的其余部分的权限。

在缺省情况下，每当注册时就激活角色。可以通过 alter user命令的default role子句来修改一个用户的缺省角色。例如，可以修改用户使其没有缺省角色。

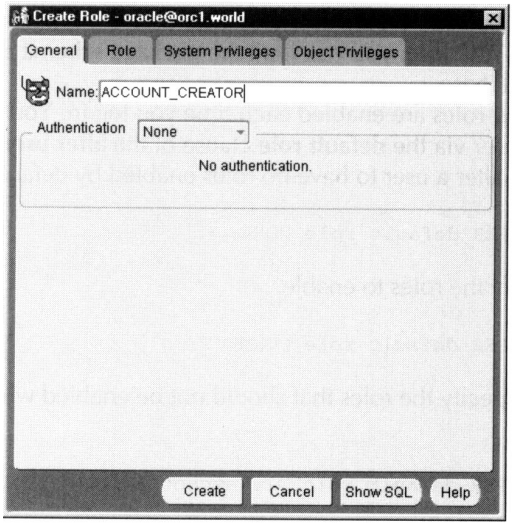

图9-3 创建ACCOUNT_ CREATOR角色

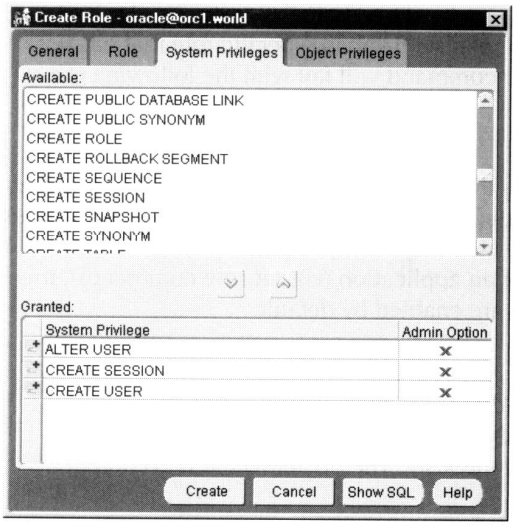

图9-4 将系统权限赋予ACCOUNT_ CREATOR角色

```
alter user THUMPER default role NONE;
```

可以指定角色启用：

```
alter user THUMPER default role CONNECT;
```

可以指定角色在会话开始时禁止：

```
alter user THUMPER default role all except ACCOUNT_CREATOR;
```

如果没有将指定的角色授予用户，alter user命令将不起作用。如果用户没有被授予一个像CONNECT那样的特定系统级角色，那么将这种角色设置成一个用户的缺省角色将造成下述错误：

```
ORA-01919: role 'CONNECT' does not exist
```

如果指定的角色是没有被授予用户的一个数据库特有角色， alter user命令将因下述错误而失败：

```
ORA-01955: DEFAULT ROLE 'ACCOUNT_CREATOR' not granted to user
```

必须在建立用户的缺省角色之前把角色授予用户。

如果使用了default role all子句，当用户的会话开始时，用户的全部角色就会被启用。如果打算在应用程序的不同部分动态地启用和禁止角色 (通过set role命令)，就 应控制被缺省启用的那些角色。

注意 init.ora参数MAX_ENABLED_ROLES限制用户能同时启用的角色数量。缺省值是20。

注意 创建一个角色时，可以用缺省方式启用它。如果创建许多角色，即使不是这些角色的用户，也可能超过MAX_ENABLED_ROLES设置值。

9.2.5 用户环境文件

可以使用环境文件来限制可由用户使用的系统和数据库资源并管理口令限制。如果数据库中没有创建环境文件，将使用缺省环境文件；缺省环境文件指定对于所有用户资源没有限制。

表9-3列出了可以通过环境文件限制的资源。

表9-3 由环境文件限制的资源

资 源	描 述
SESSION_PER_USER	在一个实例中，一个用户可以同时拥有的会话数量
CPU_PER_SESSION	一个会话可以使用的 CPU时间，以百分之一秒为单位
CPU_PER_CALL	语法分析、执行或获取可以使用的CPU时间，以百分之一秒为单位
CONNECT_TIME	一个会话可以连接到一个数据库的分钟数
IDLE_TIME	一个会话可以连接到一个数据库而没有激活使用的分钟数
LOGICAL_READS_PER_SESSION	可以在一个会话中读取的数据库块数
LOGICAL_READS_PER_CALL	在语法分析、执行或获取期间可以读取的数据库块数
PRIVATE_SGA	在SGA的SQL共享池中，一个会话可以分配的私有空间量(对于MTS)
COMPOSITE_LIMIT	一个基于前面的限制的复合限制
FAILED_LOGIN_ATTEMPTS	将引起一个帐户被锁定的连续注册失败的次数
PASSWORD_LIFE_TIME	一个口令在其终止前可以使用的天数
PASSWORD_REUSE_TIME	一个口令在能够被重新使用之前所必须经过的天数
PASSWORD_REUSE_MAX	一个口令在能够被重新使用之前必须改变的次数
PASSWORD_LOCK_TIME	如果超过FAILED_LOGIN_ATTEMPTS设置值，一个帐户将被锁定的天数
PASSWORD_GRACE_TIME	以天为单位的“宽限时间”。在宽限期内，在口令达到PASSWORD_LOGIN_TIME设置值时，仍能对其修改
PASSWORD_VERIFY_FUNCTION	一个函数名，用于判断口令的复杂性；由 Oracle提供一个口令并可以编辑

注意 PASSWORD_REUSE_MAX和PASSWORD_REUSE_TIME互不相容，如果其中一个资源设置成一个值，另一个必须设置成 UNLIMITED。

如表9-3所示，许多资源都可以被限制。但是，所有这些限制都可以重新激活，在用户超过资源限制前不会发生任何动作。因此在用户到达他们所定义的限额前，环境文件在防止超范围查询使用大量系统资源方面不会有太多帮助。一旦到达限值， SQL语句就被停止。

环境文件是通过create profile命令创建的。下例所示的alter profile命令用于修改现有的环境文件。在这个例子中，数据库的 DEFAULT环境文件被修改成允许最大空闲时间为一小时：

```
alter profile DEFAULT
limit idle_time 60;
```

OEM Security Manager工具提供一种创建和管理环境文件的 GUI方法。图9-5示出的缺省

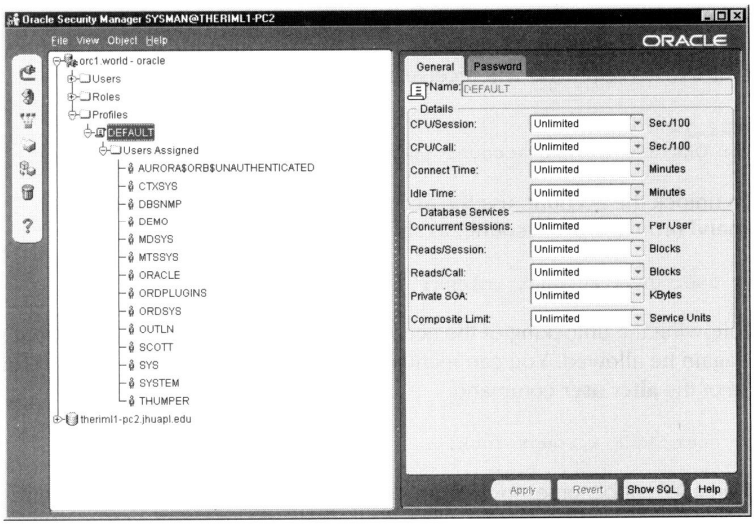

图9-5 DEFAULT环境文件设置

环境文件屏幕上，有一些分配给环境文件的资源。

如下节所述，可以使用环境文件管理口令的复杂性和寿命。

9.2.6 口令管理

在Oracle8中，可以使用环境文件来管理口令的终止、重新使用和复杂性。例如，可以限制一个口令的寿命、锁定口令过旧的帐户。也可以强制一个口令至少有一定程度的复杂性并锁定一个多次注册失败的帐户。

例如，如果设置用户环境文件的FAILED_LOGIN_ATTEMPTS资源为5，该帐户允许连续注册失败5次，第6次就会引起帐户被锁定。

注意 如果第5次尝试时提供正确的口令，则“失败的注册尝试计数”被复位为0；从而在计数被锁定之前，允许另外连续五次未成功的注册尝试。

在下面的例子中，创建一个供用户JANE使用的LIMITED_PROFILE环境文件：

```
create profile LIMITED_PROFILE limit
FAILED_LOGIN_ATTEMPTS 5;

create user JANE identified by EYRE
profile LIMITED_PROFILE;

grant CREATE SESSION to JANE;
```

如果连续5次与JANE帐户的连接失败，该帐户将自动由Oracle锁定。然后当使用JANE帐户的正确口令时，会收到一条错误信息：

```
connect jane/eyre
ERROR: ORA-28000: the account is locked
```

要对帐户解锁，请在数据库管理员帐户中使用alter user命令的account unlock子句，如下所示：

```
alter user JANE account unlock;
```

帐户解锁后，JANE帐户再一次被允许连接。可以通过alter user命令的account lock子句来手动锁定一个帐户。

```
alter user JANE account lock;
```

若一个帐户由于多次连接失败而被锁定，当超过其环境文件的PASSWORD_LOCK_TIME值时将自动解锁。例如，如果PASSWORD_LOCK_TIME设为1，前面例子中的JANE帐户就被锁定一天，过后帐户即被解锁。

可以通过环境文件中的PASSWORD_LIFE_TIME资源建立一个口令的最大寿命。例如，可以强制LIMITED_PROFILE环境文件的用户每30天改变一次口令。

```
alter profile LIMITED_PROFILE limit
PASSWORD_LIFE_TIME 30;
```

在这个例子中，alter profile命令用于修改LIMITED_PROFILE环境文件。PASSWORD_LIFE_TIME值设为30，因此使用这个环境文件的每个帐户在30天后口令就会过期。如果口令过期，就必须在下次注册时修改它，除非环境文件对过期的口令有一特定的宽限期。宽限期参数叫做PASSWORD_GRACE_TIME。如果在宽限期内没有修改口令，帐户就

会过期。

注意 如果打算使用PASSWORD_LIFE_TIME参数，必须向用户提供一种便于改变其口令的方法。

“过期”帐户与“锁定”帐户不同。如本节前面所述，锁定帐户会随着时间的推移自动解锁。而过期帐户需要通过数据库管理员人工干预才能重新激活。

注意 如果使用口令过期特性，就要确保拥有应用程序的帐户具有不同的环境文件设置值；否则它们会被锁定且应用程序变为不能使用。

如前面例子所述，若要重新恢复一个过期帐户，需使用 alter user命令。在这个例子中，用户JANE首先由数据库管理员手工使其口令过期。

```
alter user jane password expire;

User altered.
```

接着，JANE试图连接其帐户。当她输入她的口令时，立即被提示输入帐户的新口令。

```
connect jane/eyre
ERROR: ORA-28001: the account has expired

Changing password for jane
Old password:
New password:
Retype new password:
Password changed
Connected.
SQL>
```

也可以使用create user命令的password expire子句，强制用户在第一次访问他们的帐户时修改口令。不过create user命令不允许对用户设置的新口令设一个限期日期。要那样做的话，必须使用前面例子中的PASSWORD_LIFE_TIME环境文件参数。

若要查看任一帐户的口令限期日期，可查询 DBA_USERS数据字典视图的Expire_Date列。若用户自己想查看，可查询 USER_USERS数据字典视图的Expiry_Date列(通过SQL*Plus或一个基于客户机的查询工具)。

9.2.7 防止口令重新使用

若要防止一个口令重新使用，可以使用两个环境文件参数的其中一个： PASSWORD_REUSE_MAX或PASSWORD_REUSE_TIME。这两个参数互不相容；如果给其中的一个设了值，另一个就必须设为UNLIMITED。

PASSWORD_REUSE_TIME参数规定一个口令可以重新使用前必须经过的天数。例如，如果设置PASSWORD_REUSE_TIME为60天，则在60天内不能使用同一个口令。

PASSWORD_REUSE_MAX参数指定一个口令可以重新使用前必须对其改变的次数。如果试图在这个限制到达前重新使用该口令， Oracle会拒绝口令的修改。

例如，可以为本章前面创建的 LIMITED_PROFILE环境文件设置一个 PASSWORD_REUSE_MAX参数。

```
alter profile LIMITED_PROFILE limit
PASSWORD_REUSE_MAX 3
PASSWORD_REUSE_TIME UNLIMITED;
```

如果用户JANE现在试图重新使用一个最近的口令，修改口令的试图就会失败。例如，假若她如下修改口令：

```
alter user JANE identified by austen;
```

然后再次改变它：

```
alter user JANE identified by eyre;
```

在下次修改口令时，试图重新使用最近的口令，但失败了。

```
alter user jane identified by austen;
alter user jane identified by austen
*
ERROR at line 1:
ORA-28007: the password cannot be reused
```

她不能重新使用任何她的最近口令，必须提供一个新口令。

口令历史被存储在SYS模式下一个叫USER_HISTORY$的表中。在这个表中，Oracle存储了用户资源识别符、加密的口令值和创建该口令的日期/时间标记。当PASSWORD_REUSE_TIME值已过期或口令修改次数超过PASSWORD_REUSE_MAX值时，这个老的口令记录就从SYS.USER_HISTORY$表中删除。如果一个新的密码与现有的密码一样，这个新口令就被拒绝。

由于老口令存储在SYS拥有的一个表中，所以数据存储在SYSTEM表空间中。因此，如果要为频繁修改口令的大量用户保留非常大的口令历史，口令历史表(SYS.USER_HISTORY$)所需的空间就会影响SYSTEM表空间的空间需求。

9.2.8 设置口令复杂度

可以强制用户的口令符合复杂度标准。例如，可以要求口令的最小长度、不是一些简单的词、至少包括一个数字或标点符号。create profile和alter profile命令的PASSWORD_VERIFY_FUNCTION参数指定用于评估口令的函数名。如果用户提出的口令不符合要求，就不会被接受。例如，可以拒绝“austen”和“eyre”作为口令，因为它们未包含任何数字值。

为简化实施口令复杂度的过程，Oracle提供了一个函数VERIFY_FUNCTION。在缺省情况下，不创建这个函数。只有在运行utlpwdmg.sql脚本文件(该文件位于Oracle软件主目录下的/rdbms/admin子目录中)时才创建VERIFY_FUNCTION函数。下面示出了这个文件的简化版本。用黑体示出的内容为本节描述的参数。

```
Rem utlpwdmg.sql
Rem
Rem  Copyright (c) Oracle Corporation 1996. All Rights Reserved.
Rem
Rem    NAME
Rem      utlpwdmg.sql - script for Default Password Resource Limits
Rem
Rem    DESCRIPTION
Rem      This is a script for enabling the password management features
Rem      by setting the default password resource limits.
Rem
Rem    NOTES
Rem      This file contains a function for minimum checking of password
Rem      complexity. This is more of a sample function that the customer
Rem      can use to develop the function for actual complexity checks
Rem      that the customer wants to make on the new password.
Rem
```

```
Rem    asurpur     12/12/96 - Changing the name of
Rem    password_verify_function
-- This script sets the default password resource parameters
-- This script needs to be run to enable the password features.
-- However the default resource parameters can be changed based
-- on the need.
-- A default password complexity function is also provided.
-- This function makes the minimum complexity checks like
-- the minimum length of the password, password not same as the
-- username, etc. The user may enhance this function according to
-- the need.
-- This function must be created in SYS schema.
-- connect sys/<password> as sysdba before running the script

CREATE OR REPLACE FUNCTION verify_function
(username varchar2,
  password varchar2,
  old_password varchar2)
  RETURN boolean IS
   n boolean;
   m integer;
   differ integer;
   isdigit boolean;
   ischar  boolean;
   ispunct boolean;
   digitarray varchar2(20);
   punctarray varchar2(25);
   chararray varchar2(52);

BEGIN
   digitarray:= '0123456789';
   chararray:= 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
   punctarray:='!"#$%&()''*+,-/:;<=>?_';

   - Check if the password is same as the username
IF password = username THEN
raise_application_error(-20001, 'Password same as user');
END IF;

   - Check for the minimum length of the password
IF length(password) < 4 THEN
raise_application_error(-20002, 'Password length less than 4');
   END IF;

   - Check if the password is too simple. A dictionary of words may be
   - maintained and a check may be made so as not to allow the words
   - that are too simple for the password.
   IF password IN ('welcome', 'password', 'oracle', 'computer', 'abcd') THEN
      raise_application_error(-20002, 'Password too simple');
   END IF;

   - Check if the password contains at least one letter, one digit and one
   - punctuation mark.
   - 1. Check for the digit
   isdigit:=FALSE;
   m := length(password);
   FOR i IN 1..10 LOOP
       FOR j IN 1..m LOOP
         IF substr(password,j,1) = substr(digitarray,i,1) THEN
            isdigit:=TRUE;
             GOTO findchar;
         END IF;
      END LOOP;
   END LOOP;
   IF isdigit = FALSE THEN
      raise_application_error(-20003, 'Password should contain at
least one digit, one character and one punctuation');
   END IF;
   -- 2. Check for the character
```

```
<<findchar>>
   ischar:=FALSE;
   FOR i IN 1..length(chararray) LOOP
      FOR j IN 1..m LOOP
         IF substr(password,j,1) = substr(chararray,i,1) THEN
            ischar:=TRUE;

             GOTO findpunct;
         END IF;
      END LOOP;
   END LOOP;
   IF ischar = FALSE THEN
      raise_application_error(-20003, 'Password should contain at least one \
              digit, one character and one punctuation');
   END IF;
   -- 3. Check for the punctuation
<<findpunct>>
   ispunct:=FALSE;
   FOR i IN 1..length(punctarray) LOOP
      FOR j IN 1..m LOOP
         IF substr(password,j,1) = substr(punctarray,i,1) THEN
            ispunct:=TRUE;
             GOTO endsearch;
         END IF;
      END LOOP;
   END LOOP;
   IF ispunct = FALSE THEN
      raise_application_error(-20003, 'Password should contain at least one \
              digit, one character and one punctuation');
   END IF;

   <<endsearch>>
   -- Check if the password differs from the previous password by at least
   -- 3 letters
IF old_password = '' THEN
      raise_application_error(-20004, 'Old password is null');
   END IF;
   - Everything is fine; return TRUE ;
   RETURN(TRUE);
   differ := length(old_password) - length(password);

   IF abs(differ) < 3 THEN
      IF length(password) < length(old_password) THEN
         m := length(password);
      ELSE
         m := length(old_password);
      END IF;
      differ := abs(differ);
      FOR i IN 1..m LOOP
          IF substr(password,i,1) != substr(old_password,i,1) THEN
             differ := differ + 1;
          END IF;

      END LOOP;
      IF differ < 3 THEN
          raise_application_error(-20004, 'Password should differ by at \
            least 3 characters');
      END IF;
   END IF;
   - Everything is fine; return TRUE ;
   RETURN(TRUE);
END;
/

- This script alters the default parameters for Password Management

- This means that all the users on the system have Password Management

- enabled and set to the following values unless another profile is
```

```
- created with parameter values set to different value or UNLIMITED

- is created and assigned to the user.

ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10
PASSWORD_REUSE_TIME 1800
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PASSWORD_VERIFY_FUNCTION verify_function;
```

注意 这个函数应当在SYS模式下创建。

函数中的前三个if子句检查口令是否与用户名相同、是否少于4个字符、是否是一组特定的词之一。可以任意修改这些检查或增加你的要求。例如，安全原则可能要求口令最少有六个字符；运行前要简单地更新部分utlpwdmg.sql文件。

函数的下一个主要部分是对口令字符串内容的三段检查。要通过这些检查，口令中至少要包含一个字符、一个数字和一个标点符号。同前面的检查一样，它们是可以编辑的。例如，可以不要求用户在其口令中使用标点符号，只要简单地绕过那部分口令检查就可以。

函数的下一部分是将新口令与老口令逐字符进行比较。如果它们之间的不相同之处少于三个，新口令将不予接受。

这个脚本文件中最后一条命令不属于该函数，它是一条改变DEFAULT环境文件的alter profile命令。如果改变了DEFAULT环境文件，那么数据库中所有使用DEFAULT环境文件的用户都会受到影响。程序清单中的命令建立以下限制：口令寿命60天，宽限期10天，口令重新使用期1800天，三次连接失败后帐户锁定和一分钟(一天的1/1440)后自动解锁。这些参数可能并不反映你所要求的设置值。最重要的设置是最后一条——由ut1pwdmg.sql脚本文件创建的VERIFY_FUNCTION函数作为PASSWORD_VERIFY_FUNCTION。

注意 这个函数仅用于使用特定环境文件的用户。

要注意的是，VERIFY_FUNCTION函数并不进行任何数据库访问，也不更新任何数据库值。如果修改这个函数，应确保此修改不需要数据库访问或修改。

可以修改缺省的环境文件以便在不修改口令过期参数的情况下使用VERIFY_FUNCTION函数。

```
alter profile DEFAULT limit
PASSWORD_VERIFY_FUNCTION VERIFY_FUNCTION;
```

如果修改DEFAULT环境文件，要确保该环境文件的所有用户都能成功地使用它。例如，如果SYS和SYSTEM用户使用DEFAULT，能否按这里指定的设置值来处理其口令呢？可能要创建一个新的环境文件并将该文件指定给非数据库管理员的用户和非应用程序拥有者的用户来简化环境文件管理。这种方法存在的问题是：要记住给所有新用户指定新的环境文件。用户管理活动越标准化，实现这种处理的机会越好。

口令验证函数的名字不一定是VERIFY_FUNCTION。如前一个程序清单所示，把函数名作为alter profile命令中的一个参数传递。由于VERIFY_FUNCTION的名字几乎每个函数都用到，所以应当将其改为对数据库有意义的一个名字。例如，可以将其改为VERIFY_Oracle_PASSWORD。应当给它一个描述性的和容易记忆的名字，这样做可以增进其他数据库管理员

对程序所执行的函数的理解。

OEM Security Manager工具允许创建环境文件。可以使用这个工具方便地定义常规环境文件控制资源和口令资源的范围。图9-6示出了常规环境文件值，图9-7示出了口令值屏幕。

有关口令的附加管理选项，将在本章后面9.5节“口令加密与技巧”描述。

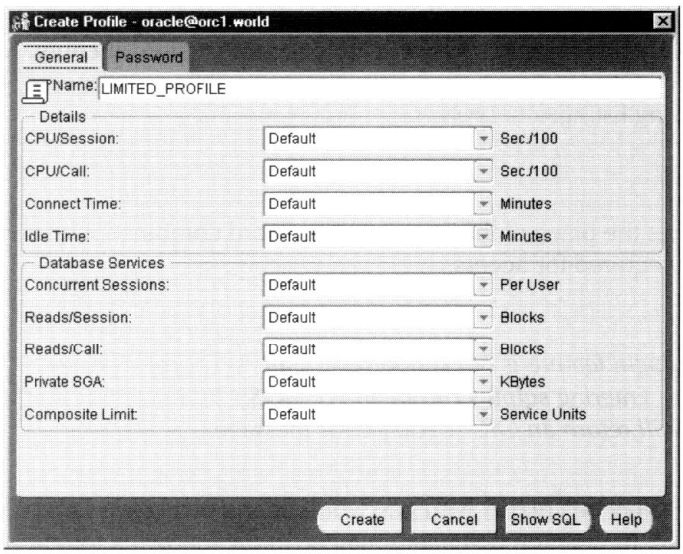

图9-6 LIMITED_PROFILE常规设置

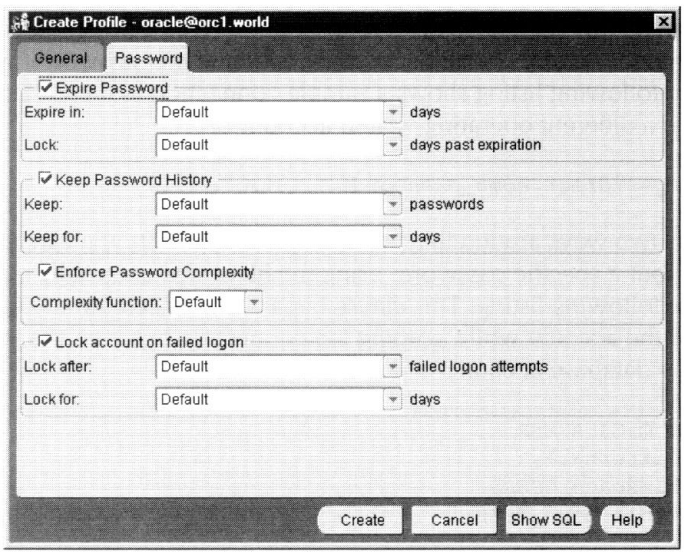

图9-7 LIMITED_PROFILE口令设置

9.2.9 数据库帐户与主机帐户相连

当用户输入有效的用户名和数据库的口令后，就允许访问该数据库。不过，有可能利用操作系统提供一层附加的用户验证。

在同一服务器中，一个数据库帐户可以和一个操作系统帐户配成对。只是在数据库帐户名的前缀部分两个帐户名有所不同。缺省前缀为“ OPS$”，但可以通过数据库init.ora文件的OS_AUTHENT_PREFIX参数改变成另外的值。甚至可以将这个前缀设置成空字符串，以便不用前缀。

注意 如果将OS_AUTHENT_PREFIX改变为非“OPS$”的任意值，数据库帐户就可以被用作自动登录帐户或通过用户名/口令访问，两者只能取其一。如果将“OPS$”用作验证前缀，就可以作为自动登录帐户和通过用户名/口令组合进行访问帐户。大多数安装程序使用“OPS$”。

例如，假设一个操作系统帐户名为 FARMER。这个用户的相应数据库帐户名则为OPS$FARMER。当用户FARMER登录到其操作系统帐户后，就可以访问OPS$FARMER帐户而不需要指定一个口令，如下面清单所示：

```
> sqlplus /
```

符号“/”代替通常访问所要求的用户名和口令组合。

注意 不是所有平台都支持自动登录性能。从NT DOS提示输入sqlplus/将返回一个ORA-01004错误消息。

帐户可以用口令创建。再以OPS$FARMER帐户为例，其创建命令可以是以下格式：

```
create user OPS$FARMER
identified by SOME_PASSWORD
default tablespace USERS
temporary tablespace TEMP;
```

即使口令不被使用，也仍然可以设定。由于帐户有一个口令，所以如果知道数据库帐户的这个口令，就可以从另一个操作系统帐户访问数据库的OPS$FARMER帐户。下面的清单显示从另一个操作系统帐户到OPS$FARMER帐户的连接例子：

```
> sqlplus ops$farmer/some_password
```

对于这个问题有两种选择。第一种选择是，若采用identified externally子句，不使用指定的口令就可以创建帐户(如下列清单所示)。这个子句在保持主机帐户名和数据库帐户名连接时不要求显式的帐户口令。

```
create user OPS$FARMER
identified externally
default tablespace USERS
temporary tablespace TEMP;
```

使用identified externally子句时，强制数据库允许操作系统帐户对其访问。操作系统帐户名和数据库帐户名必须一致(除了数据库帐户名的前缀外)。

第二种选择是，用一个不存在的口令创建帐户。如本章9.5.2节“设置不存在的口令”所述，这种方法防止用户不通过与其相关的操作系统帐户就登录到任一数据库帐户。

有一种情况是，可能允许用户有一个带可用口令的OPS$帐户。如果用户通过SQL*Net直接从操作系统或远程帐户登录。使用具有远程访问口令的帐户可能有利。如果开发人员正在操作系统级连接数据库，她将不想显示其口令来测试一个脚本文件，OPS$自动登录功能部件支持这种需要。如果正在通过远程访问(且REMOTE_OS_AUTHENT在init.ora文件中未设置成TRUE)，她将需要一个能访问数据库的口令。

9.2.10 用口令文件进行验证

在大多数情况下，DBA用户可以由操作系统验证。例如在UNIX系统上，/etc/group文件中DBA组的成员可以内部连接。如果DBA用户不能由操作系统验证，就要创建并保存一个口令文件。

若要创建一个口令文件，请进行下述几步操作：

1) 使用ORAPWD实用程序创建口令文件。

```
> ORAPWD FILE=filename  PASSWORD=password  ENTRIES=max_users
```

ORAPWD是一个生成口令文件的 Oracle实用程序。执行 ORAPWD时，除了 SYS和INTERNAL访问的口令外，还规定要创建的口令文件的名称。 ENTRIES参数通知Oracle，要在口令文件中创建多少条目。由于不能在以后扩展该文件，因此要把 ENTRIES值设得高一些。如果超出口令文件条目数的范围限额，就收到一个 ORA-1996错误。重建该口令文件时，需要重新授予SYSDBA和SYSOPER权限。

2) 将init.ora文件中的REMOTE_LOGIN_PASSWORDFILE初始参数设置成EXCLUSIVE，关闭并重新启动数据库，以便变更的参数起作用。

3) 如下面例子所示，将SYSOPER和SYSDBA权限授予需要进行数据库管理的每一个用户。SYSDBA授予用户数据库管理员的权限；SYSOPER使用户能执行数据库操作支持活动。为了授予用户SYSOPER或SYSDBA权限，必须在内部连接。被受权用户现在应能通过使用一个与下述命令类似的命令与数据库连接：

```
connect george/mch11@PROD.world AS SYSDBA
```

如下面例子所示，可以用revoke命令撤消一个用户的SYSDBA或SYSOPER权限：

```
revoke SYSDBA from George;
```

若要查看具有SYSDBA或SYSOPER系统权限的用户，可以查询 V$PWFILE_USERS。如果用户拥有SYSDBA权限，V$PWFILE_USERS在其SysDBA列中将有一个TRUE值；如果拥有SYSOPER权限，将在其SysOper列中有一个TRUE值。

9.2.11 口令保护

帐户和角色都可以用口令保护。它们的口令在创建时设置，也可以通过 alter user和alter role命令修改。

如下面程序清单所示，帐户的初始口令通过 create user命令设置。在这个例子中，帐户THUMPER创建时的初始口令为RABBIT：

```
create user THUMPER
identified by RABBIT;
```

帐户口令的改变应通过alter user命令进行。下面列出一个alter user命令的例子：

```
alter user THUMPER identified by NEWPASSWORD;
```

在Oracle8中，可以使用SQL*Plus的password命令来改变用户的口令。 password命令将提示你输入一个旧口令、一个新口令和对新口令的确认。输入的口令值不在显示屏上显示。

如下所示，若用SQL*Plus改变自己的口令，可键入password命令：

```
password
```

若要改变另一个用户的口令，可以使用其后跟有用户名的 password 命令。

```
password JANE
```

提示要求输入JANE的新口令和一个确认。password 命令对最终用户非常有用，因为它极大地简化了改变口令时所需要的命令。如果不使用 password 命令，将需要使用如下命令：

```
alter user USERNAME identified by NEWPASSWORD;
```

注意 alter user命令并不完全实施本章前面所述的口令验证函数。改变口令时，Oracle推荐使用password命令。

password命令简化了口令修改进程，这在 Oracle8中显得尤为重要；因为你可以强制用户改变其命令。

角色的口令在角色创建时通过 create role命令设置。不一定非要给角色指定一个口令。如果指定了一个口令，角色被用户启用时必须输入口令。

```
create role ACCOUNT_CREATOR identified by HELPDESK_ONLY;
```

可以使用 alter role 命令来改变与角色相关的口令。同用户口令一样，角色也可以是 identified externally，从而也可以执行主帐帐户名与角色名的连接。与用户帐户不同，角色可能没有口令(缺省)。可以通过not identified子句从一个角色中删除口令，如下例所示：

```
alter role ACCOUNT_CREATOR not identified;
```

这个命令执行后，ACCOUNT_CREATOR角色将不受口令保护。

角色可与操作系统权限相连。如果这个性能在操作系统中可用，就可以使用 alter role命令的identified externally子句来调用它。当该角色被启用时，Oracle将检查操作系统来检验访问。修改一个角色以使用这种安全特性的例子如下所示：

```
alter role MANAGER identified externally;
```

在VMS中，检验进程使用操作系统权限识别符。在大多数 UNIX系统中，检验进程都是使用/etc/group文件。为使这个文件用于所有操作系统，必须将 init.ora文件中的OS_ROLES数据库启动参数设置为TRUE。

下面这个检验进程的例子用于 UNIX系统上一个叫作 Local的数据库实例。服务器的/etc/group文件可以包含下列条目：

```
ora_local_manager_d:NONE:1:dora
```

这个条目是将MANAGER角色授予一个帐户名 Dora。后缀_d表示在Dora登录时这个角色缺省授权。后缀_a表示该角色有with admin option选项。如果该角色也是用户的缺省角色，则后缀将是_ad。如果授予该角色的用户不止一个，则要向 /etc/group条目附加用户名。例如：

```
ora_local_manager_d:NONE:1:dora,judy
```

如果使用这个选项，数据库中所有角色都将通过操作系统来启用。

9.2.12 对象级权限

对象级权限(object-level privilege)使用户可以访问不属于自己的数据。可以使用角色来减少权限的管理。显式权限也可以使用，并且在一些情况下是必须的。

权限通过grant命令创建，存于数据字典中。对表、视图、序列(以及它们的同义词)的访问，加上执行过程、函数、软件包及类型的能力都可以授权给用户。表 9-4列出了可以授予对象的权限。

表9-4 允许的对象权限

权 限	授予能力
SELECT	可查询对象
INSERT	可以把行插入对象中，该权限可授予对象的特定列
UPDATE	可更新对象的行，该权限可授予对象的特定列
DELETE	可从对象中删除行
ALTER	可修改对象
INDEX	可以在表上创建索引
REFERENCES	可以创建引用表的外键
EXECUTE	可以执行函数、软件包、过程、库或类型
READ	可访问目录

可以使用with grant option子句向授与用户传递授权能力，以便在基对象上进一步授权。下面的SQL*Plus清单就示出了一个这样的例子。在这个例子中，用户 THUMPER用with grant option选项向用户MCGREGOR授予在EMPLOYEE表上SELECT和部分UPATE访问的权限。这个用户又将其中一个权限授予另一用户 JFISHER。

```
grant select, update (Employee_Name, Address)
on EMPLOYEE to MCGREGOR
with grant option;

connect MCGREGOR/FARMER
grant select on THUMPER.EMPLOYEE to JFISHER;
```

注意 若给PUBLIC授权，则使授权对数据库中所有用户有效。

如果EMPLOYEE是一个分区表，不允许只对它的一个分区授予 SELECT访问权。不过，可以创建只从一个分区中选择数据的视图，然后把对这个视图的 SELECT访问权授予用户。该视图是一个附加的管理对象，但是可以用它实现分区级数据安全。

权限的管理很快会变成一个费时的任务。对于数据库应用程序中的每一个对象，要授予每个用户适当的权限。一个小的应用程序有 30个用户，每个用户有20张表，这样就要管理600个权限(20张表乘以30个用户)。

随着角色的出现，这些权限的管理就容易了。角色是成组的权限，角色可授给用户，这样就大大简化了权限管理进程。

下面的清单中示出了角色的使用例子。在这个例子中创建了两个角色。第一个角色是APPLICATION_USER，被给予系统级权限 CREATE SESSION；被授予该角色的用户可以在数据库登录。第二个角色是DATA_ENTRY_CLERK，被授予表上的权限。

```
create role APPLICATION_USER;
grant CREATE SESSION to APPLICATION_USER;

create role DATA_ENTRY_CLERK;
grant select, insert on THUMPER.EMPLOYEE to DATA_ENTRY_CLERK;
grant select, insert on THUMPER.TIME_CARDS to DATA_ENTRY_CLERK;
grant select, insert on THUMPER.DEPARTMENT to DATA_ENTRY_CLERK;
```

角色也可以被授予其他角色。例如，可以把 APPLICATION_USER角色授予DATA_ENTRY_ CLERK角色。如下例所示：

```
grant APPLICATION_USER to DATA_ENTRY_CLERK;
```

角色可以被授给一个用户。这个角色可以在用户会话期间通过 set ro1e命令动态地启用和

禁止。

```
grant DATA_ENTRY_CLERK to MCGREGOR;
```

从OEM Security Manager工具中，选择要授予对象权限的特定用户或角色，并使用 Object Privileges窗口选择模式和权限。图 9-8示出了被授予 LOCATION 表上SELECT、INSERT和UPDATE权限的用户 THUMPER。

角色和系统权限 (如CREATE TABLE)可以由接受的用户传递给其他用户。对于角色，可使用with admin option子句。在下面清单中，将前面创建的 DATA_ENTRY_CLERK角色与管理这个角色的权限一起授给用户 BPOTTER：

```
grant DATA_ENTRY_CLERK to BPOTTER with admin option;
```

有了这个权限，用户 BPOTTER就可以给其他用户授予这个角色或从其他用户处撤销这个角色，也可以删除这个角色。

注意 通过角色拥有表权限的用户，不能创建基于这些表的视图或过程。由于仅当用户登录和角色启用时通过角色授权才有效，所以需要这个限制。创建无拥有者的视图要求表中有显式权限。

角色的动态特性对于限制用户权限非常有用。如果在用户启动一个应用程序时启用角色 (通过set role命令)；离开应用程序时就要禁止。这样，除了使用应用程序外，用户不能利用该角色的权限。

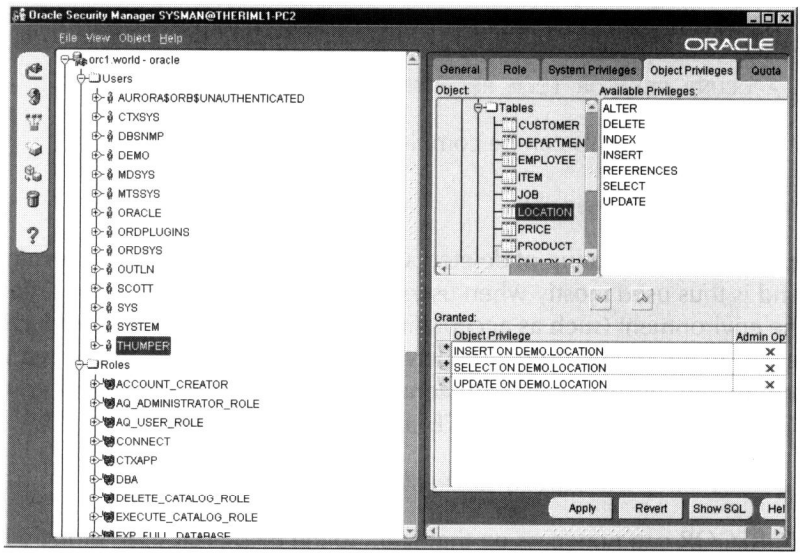

图9-8 对象权限授予用户

例如，当MCGREGOR登录到一个应用程序时，就可以执行命令

```
set role DATA_ENTRY_CLERK;
```

当该用户离开该应用程序时，执行命令

```
set role NONE;
```

就会禁止所有通过角色授予的权限。

可以用revoke命令从用户那里撤销权限和角色。可以撤销一个用户的某些权限 (显式列出)

或全部权限(使用all关键字)。在下例中，从一个用户撤销EMPLOYEE表的一个指定权限，而另一个用户的权限被全部撤销：

```
revoke delete on EMPLOYEE from PETER;
revoke all on EMPLOYEE from MCGREGOR;
```

在下例中，从用户帐户HELPDESK中撤销ACCOUNT_CREATOR角色：

```
revoke ACCOUNT_CREATOR from HELPDESK;
```

由于用户帐户可以通过

```
drop user USERNAME cascade;
```

命令被全部删除，所以删除帐户时不要求清除权限。因此，当用户改变状态或者当应用程序从一个环境(如验收测试)转移到另一个环境(如产品)时，revoke命令使用得最多。

授予with grant option权限的revoke和授予with admin option权限的revoke之间有一个重要的差别。假若THUMPER将with grant option权限授予对EMPLOYEE表访问的MCGREGOR：

```
grant SELECT on EMPLOYEE to MCGERGOR with grant option;
```

除了with grant option 权限外，MCGREGOR现在还可以把这个授权传递给其他用户：

```
grant SELECT on THUMPER.EMPLOYEE to BPOTER with grant option;
```

如果THUMPER现在撤消以前对MCGREGOR的授权：

```
revoke SELECT on EMPLOYEE from MCGREGOR;
```

BPOTTER已通过MCGREGOR接受对EMPLOYEE表的访问，现在他有什么权限呢？由于MCGREGOR不能再访问这个EMPLOYEE表，所以BPOTTER就不能再访问THUMPER的这个表。

撤消授予with admin option的权限所起的作用不同。如果授予MCGREGOR一个系统权限with admin option，则MCGREGOR就可以把这个系统权限授予 BPOTTER。如果此后撤消MCGREGOR的权限，BORTTER仍保持新的系统权限。

9.2.13 列表权限

有关已经授予的权限的信息存储在数据字典中。这个数据可以通过数据字典视图进行访问。

可以使用表9-5中列出的数据字典视图来列出数据库中已授予的权限。也可以使用用户级视图。

表9-5 与权限相关的数据字典视图

数据字典视图	内 容
DBA_ROLES	角色名及其口令状态
DBA_ROLES_PRIVS	已被授予角色的用户
DBA_SYS_PRIVS	已被授予系统权限的用户
DBA_TAB_PRIVS	已被授予表中权限的用户
DBA_COL_PRIVS	已被授予列中权限的用户
ROLE_ROLE_PRIVS	已被授给其他角色的角色
ROLE_SYS_PRIVS	已被授给角色的系统权限
ROLE_TAB_PRIVS	已被授给角色的表权限

例如，可能想要显示哪些系统权限已被授给哪些角色。在这种情况下，下列查询将会显示这些信息：

```
select
      Role,              /*Name of the role*/
      Privilege,         /*System privilege*/
      Admin_Option       /*Was admin option granted?*/
 from ROLE_SYS_PRIVS;
```

若要检索授予用户的表权限，需找出两种类型的授权：给用户的显式授权和通过角色的授权。如下面清单所示，若要查看显式授予的权限，可查询 DBA_TAB_PRIVS视图：

```
select
      Grantee,           /*Recipient of the grant*/
      Owner,             /*Owner of the object*/
      Table_Name,        /*Name of the object*/
      Grantor,           /*User who made the grant*/
      Privilege,         /*Privilege granted*/
      Grantable          /*Was admin option granted?*/
 from DBA_TAB_PRIVS;
```

若要查看通过角色授予的表权限，可在 DBA_ROLE_PRIVS中查找用户的记录并与角色的表权限(列在ROLE_TAB_PRIVS中)进行比较。

```
select
      DBA_ROLE_PRIVS.Grantee,        /*Recipient of the grant*/
      ROLE_TAB_PRIVS.Owner,          /*Owner of the object*/
      ROLE_TAB_PRIVS.Table_Name,     /*Name of the object*/
      ROLE_TAB_PRIVS.Privilege,      /*Privilege granted*/
      ROLE_TAB_PRIVS.Grantable       /*Was admin option granted?*/
 from DBA_ROLE_PRIVS, ROLE_TAB_PRIVS
where DBA_ROLE_PRIVS.Granted_Role = ROLE_TAB_PRIVS.Role
  and DBA_ROLE_PRIVS.Grantee = 'some username';
```

这个查询将针对一个特定用户，检索由角色授予的表权限。

若要查看当前会话的环境文件限制，可以查询 USER_RESOURCE_LIMITS。它的列是：

Resource_Name	资源名(如，SESSIONS_PER_USER)
Limit	对这个资源的限制

USER_PASSWORD_LIMITS描述了用户的口令环境文件参数。它和 USER_RESOURCE_LIMITS具有相同的列。

USER_RESOURCE_LIMITS视图中没有“ DBA ”类型，它严格地受用户当前会话的限制。若要查看与各可用资源相关的开销，可以查询 RESOURCE_COST视图。数据库管理人员可以访问DBA_PROFILES视图来查看所有环境文件的资源限制。 DBA_PROFILES的Resource_Type列表示资源环境文件是一个PASSWORD环境文件，还是一个KERNEL环境文件。

除了这些视图外，还有两个视图。每个视图各有一个列，分别列出了当前会话中启用的权限和角色。它们是：

SESSION_PRIVS	Privilege列列出了会话中启用的所有系统权限，不论是直接授予还是通过角色授予。
SESSION_ROLES	Role列列出了当前会话启用的所有角色。

SSESSION_PRIVS和SESSION_ROLES对所有用户都是可用的。

9.3 限制可用的命令：产品用户环境文件

在SQL*Plus中，提供了一个附加的安全级 —— 对于指定用户禁止使用各个命令。这样，就可以防止表上有更新权限的用户在失控情况下利用 SQL*Plus命令行接口更新这个表。

这个性能允许数据库管理员防止用户从 SQL*Plus内访问操作系统(通过host命令)。当应用程序包含一个访问SQL*Plus的选项并且你不想让用户访问操作系统时，这种预防是很有用的。

除了撤销用户从SQL*Plus中使用host命令的能力外，也可以撤销他们使用 connect命令的能力。消除对这一命令的访问可以迫使用户只能留在其自己的帐户内。下列程序示出了设置这个安全级后这些命令的结果：

```
SQL> host
invalid command: host
SQL> connect system/manager
invalid command: connect
```

各种情况下都返回“ invalid command ”信息。用户必须留在自己的帐户中。

若要创建这个安全级，必须先创建 Product User Profile(产品用户环境文件)表。创建它们的脚本文件叫作pupbld.sql，可以在Oracle软件主目录的/sqlplus/admin子目录中找到它。这个脚本文件创建一些表和视图，需要从 SYSTEM帐户中运行。

对于SQL*Plus，其最重要的表可以通过一个叫 PRODUCT_USER_PROFILE的同义词来访问。表9-6列出了用于安全目的的关键列。通过向该表插入记录来创建所希望的安全级。

也可以使用Product User Profile表来禁止角色。若要禁止一个角色，应将 Attribute列设为ROLES，把角色名放入Char_Value列中。禁止角色通常与禁止 set命令相结合(见表9-6)。

表9-6 PRODUCT_USER_PROFILE中的列

列 名	描 述
PRODUCT	设置为SQL*Plus。如这里所示，名字必须是混合方式
USERID	被禁止命令的用户名，要求大写。指定多用户时可以使用通配符%，只使用 %则适用于所有用户
ATTRIBUTE	被禁止的命令名，要求大写。在 SQL*Plus 中禁止SET命令也就禁止了 set role和set transaction
CHAR_VALUE	用大写设为DISABLED

9.4 登录期间的口令安全

当从一个客户机连接到数据库服务器，或者通过数据库链接从一个数据库连接到另一个数据库时，除非指定其他形式，否则 Oracle将以非加密的形式传输输入的口令。对于 Oracle8，可以设置参数来强制Oracle在传输前将口令值加密。若要启用口令加密，需设置以下参数：

- 对于客户机，把sqlnet.ora文件中的ORA_ENCRYPT_LOGIN参数设为TRUE。
- 对于服务器， 把init.ora文件中的DBLINK_ENCRYPT_LOGIN参数设为TRUE。

一旦这些参数被设置(并且关闭和重新启动数据库)，口令将以加密的形式在客户机到服务器和服务器到服务器之间传送。

9.5 口令加密与技巧

了解如何进行数据库加密和设置口令，可以使数据库管理员能执行一系列用其他方法不

可能执行的事务。如后面几节所述，这些事务包括设置不存在的口令和变成其他用户的能力。

9.5.1 如何存储口令

当为一个用户帐户或角色规定口令时，数据库就把该口令的加密 (encrypted)版本存入数据字典中。为两个不同的帐户设置相同的口令会形成不同的密码。对所有口令，加密的值为 16 个字符长，含有数字和大写字母。

怎样验证口令呢？在用户验证期间输入一个口令，将这个口令加密，将生成的密码与数据字典中该帐户的密码进行比较，如果一致，口令就正确，验证成功。

9.5.2 设置不存在的口令

了解数据库如何存储口令很重要，因为它给帐户安全增加了新的选项。如果可以指定一个口令的密码而不是口令本身，那会发生什么情况呢？如果生成的密码没有遵循加密口令的格式规则，那又会怎样呢？结果将是永远也登录不上的一个帐户。这是因为没有一个口令会生成无效的密码。

假设帐户和加密的口令由下面的查询选择。该查询从 DBA_USERS视图选择Username(用户名)和Password(口令)字段。

```
select
      Username,          /*Username*/
      Password           /*Encrypted password*/
from DBA_USERS
where Username in ('MCGREGOR','THUMPER','OPS$FARMER');

USERNAME          PASSWORD
----------------- -----------------
MCGREGOR          1A2DD3CCEE354DFA
THUMPER           F3DE41CBB3AB4452
OPS$FARMER        4FF2FF1CBDE11332
```

要注意的是，每个加密口令的输出都是 16位字符长。

由于口令不存储在数据字典 (但其密码存储在数据字典中)，导入是如何知道哪些是口令呢？当由输出转储文件执行一个全导入应用操作时，口令同时被导入。

导入执行SQL命令。在一个全导入操作期间，导入执行 create user命令的一个非文档形式。从上一段程序所示的数据库中导入 MCGREGOR用户会生成下面的create user命令：

```
create user MCGREGOR identified by VALUES '1A2DD3CCEE354DFA';
```

换句话说，导入使用 identified by子句中的非文档 values子句为正在创建的用户指定加密的口令。

导入不能用得很随意。可以用同样的命令为任意帐户设置一个密码。只要设置的密码违背加密规则(16个字符，全部是大写字母)，用户验证时就不可能匹配。结果就成为一个只能从服务器上正确操作系统帐户访问的帐户。在下列程序中，密码设置为短语 ‘ no way ’。然后查询DBA_USERS视图。

```
alter user OPS$FARMER identified by VALUES 'no way';

select
      Username,          /*Username*/
```

```
      Password          /*Encrypted password*/
from DBA_USERS
where Username in ('MCGREGOR','THUMPER','OPS$FARMER');

USERNAME          PASSWORD
----------------- -----------------
MCGREGOR          1A2DD3CCEE354DFA
THUMPER           F3DE41CBB3AB4452
OPS$FARMER        no way
```

现在已不可能访问OPS$FARMER帐户，除非通过服务器上的FARMER帐户。尽管那样，也只能通过“/”自动注册来进行访问。不存在的口令对于锁定非OPS$帐户也很有用，这些帐户从来不应直接登录(如SYS)。

9.5.3 变成另一个用户

由于可以设置加密的口令，所以可以暂时接管任何帐户，并将其设置回原来的口令而无需知道帐户口令。这种能力使用户能变成另一个用户(在测试应用程序或查找产品中的问题时，这种能力非常有用)。

暂时变成另一个用户需要以下几步操作：

1) 查询DBA_USERS以确定帐户的当前加密口令。

2) 生成alter user命令，工作完成后需要用这个命令把加密的口令恢复到其当前值。

3) 把alter user命令放进一个文件中。

4) 改变用户的口令。

5) 访问用户的帐户并执行测试。

6) 测试结束时，执行含有alter user命令的文件，以便将用户的加密口令恢复到其初始值。

这个处理通过SQL*Plus脚本文件自动完成。在完成测试后，这个脚本文件自动生成恢复用户帐户口令所必需的命令。

```
REM*  become_another_user.sql
REM*
REM*  This script generates the commands necessary to allow
REM*  you to temporarily become another user.
REM*
REM*  It MUST be run from a DBA account.
REM*
REM*  Input variable: The username of the account to be taken
REM*  over.
REM*
REM*  Steps 1, 2, and 3: Query DBA_USERS. Generate the ALTER USER
REM*  command that will be necessary to reset the password to its
REM*  present value.
REM*
set pagesize 0 feedback off verify off echo off termout off
REM*
REM*  Create a file called reset.sql to hold the commands
REM*  generated
REM*
spool reset.sql
REM*
REM*  Select the encrypted password from DBA_USERS.
REM*
SELECT 'alter user &&1 identified by values '||''''||
password||''''||' profile '||profile||';'
FROM dba_users WHERE username = upper('&&1');
```

```
prompt 'host rm -f reset.sql'
prompt 'exit'
spool off
exit
```

注意 在select语句中，有两组4个单引号。

这个脚本文件生成一个叫作reset.sql的输出文件。这个文件有三行。第一行含有 alter user命令，在values子句后面有一个加密的口令。第二行含有一个删除 reset.sql文件的host命令(因为做完第6步操作后已不需要该文件)。第三行含有一个退出SQL*Plus的exit命令。下面所示的是reset.sql样本文件：

```
alter user MCGREGOR identified by values '1A2DD3CCEE354DFA' profile DEFAULT;
host rm -f reset.sql
exit
```

第二行的rm -f命令应被你的操作系统中适当的文件删除命令替代。

现在可以处理第4和第5步了，它涉及到改变用户口令(通过alter user命令)和访问帐户。这两步的操作如下所示：

```
alter user MCGREGOR identified by MY_TURN;
connect MCGREGOR/MY_TURN
```

现在就可以登录到MCGREGOR帐户。完成测试后，登录到 SQL*Plus并运行上面所示的reset.sql脚本文件。reset.sql脚本文件的执行命令如下所示：

```
sqlplus system/manager @reset
```

如果正在同时测试多个用户，就应将用户名嵌入 reset.sq1文件名中；否则第一个 reset.sql文件会被新的文件重写。如果正在为 OPS$帐号执行这个操作，要小心才是。因为在一些操作系统中(如UNIX中)，“ $ ”是一个特殊字符。

这时，帐户被恢复为其初始的加密口令 —— 即原始口令。帐户测试不需要知道其口令也不会毁坏其口令。

取消“ 变成另一个用户 ”操作期间时的口令约束

在Oracle8中，可以防止用户重新使用旧的口令 (见本章前面 9.2.7节“ 防止口令重新使用 ”)。防止旧口令重用会影响变成另一个用户的能力。如前面所示，变成另一用户的标准步骤如下：

1) 查询DBA_USERS，确定帐户的当前加密口令。

2) 之后，生成alter user命令，以便将加密的口令恢复为其当前值。

3) 将alter user命令保存进一个文件。

4) 改变用户的口令。

5) 访问用户帐户并执行测试。

6) 测试结束时，运行含有alter user命令的文件，以便将用户的加密口令恢复为其原来的值。

如果不允许用户重用旧口令，将是什么情况呢？步骤 6将失败，不能把用户的口令恢复为其原来的值。必须为用户设置一个新口令，从而大大降低“ 变成另一个用户 ”过程的效果。

为避免这种情况，创建一个不执行口令历史约束条件的环境文件。在变成这个用户之前，将新的环境文件分配给它。当完成作为这个用户的会话时，将原始环境文件分配给用户。在前面提供的脚本文件中，获取用户的 profile设置值并将其集成到恢复用户口令的 alter user命令中。其步骤如下：

1) 检查用户的环境文件设置值。

2) 把用户分配给一个环境文件，该环境文件没有口令历史限制、没有口令验证函数，并有无限口令重用次数。例如，环境文件创建命令可能是：

```
create profile temp_profile limit
password_verify_function    null
password_reuse_time         unlimited
password_reuse_max          unlimited;
```

3) 查询DBA_USERS，确定帐户的当前加密口令及环境文件。

4) 之后，生成alter user命令，以便以后将加密的口令恢复为其当前值。

5) 将alter user 命令保存进一个文件。

6) 改变用户口令。

7) 访问用户的帐户并进行测试。

8)测试结束时，运行含有alter user命令的文件，以便将用户的加密口令和环境文件恢复为其原始值。

9.6 审计

数据库具有审计发生在其内部的所有操作的能力。审计记录可以写入 SYS.AUD$表或操作系统的审计跟踪中。使用操作系统审计跟踪的能力依赖于操作系统。

有三种不同的操作类型可以被审计：登录企图、对象访问和数据库操作。这些类型将在下面几节分别描述。当执行审计时，数据库的缺省功能记录下成功和不成功的命令；可以在设置审计类型时对其进行修改。

若要允许在一个数据库中进行审计，数据库的 init.ora文件必须含有AUDIT_TRAIL参数的条目。AUDIT_TRAIL值为：

NONE	禁止审计
DB	启用审计，写入SYS.AUD$表
OS	启用审计，写入操作系统的审计跟踪 (依赖于操作系统)

无论是否设置AUDIT_TRAIL参数，都可以发出以下几节描述的 audit命令。除非使用启用审计的AUDIT_TRAIL值启动数据库，否则它们将不被激活。

如果选择把审计记录存储在 SYS.AUD$表中，这个表的记录就应当定期归档，然后表也应被截断。由于它是在数据字典中，所以该表在 SYSTEM表空间中。如果其记录不进行定期清理就会引起空间问题。可以给用户授予 DELETE_CATALOG_ROLE权限，使其能在SYS.AUD$表中进行删除操作。

9.6.1 登录审计

每个连接数据库的企图都可被审计。开始审计登录企图的命令为：

```
audit session;
```

若只是审计成功或失败的连接企图，可使用下列命令之一：

```
audit session whenever successful;
audit session whenever not successful;
```

如果审计记录存储于SYS.AUD$表中，这时就可以通过DBA_AUDIT_SESSION数据字典视图来查看该表。

下面示出的查询从DBA_AUDIT_SESSION视图中检索登录审计记录。它列出了使用的操作系统帐户(OS_Username)、Oracle帐户名(Username)和使用的终端ID(Terminal)。对Returncode列进行检查：如果为0，连接企图就成功；否则就检查两个常用错误号，确定失败的原因。登录和注销的时间也要显示。

```
select
  OS_Username,            /*Operating system username used.*/
  Username,               /*Oracle username of the account used.*/
  Terminal,               /*Terminal ID used.*/
  DECODE(Returncode,'0','Connected',
              '1005','FailedNull',
              '1017','Failed',Returncode),      /*Failure check*/
  TO_CHAR(Timestamp,'DD-MON-YY HH24:MI:SS'),    /*Login time*/
  TO_CHAR(Logoff_Time,'DD-MON-YY HH24:MI:SS')   /*Logoff time*/
from DBA_AUDIT_SESSION;
```

检查的两个错误号为ORA-1005和ORA-1017。这两个错误代码覆盖了常发生的大多数登录错误。当用户输入一个用户名但无口令时就返回ORA-1005。当用户输入一个无效口令时就返回ORA-1017。

如下例所示，若要禁止会话审计，可使用noaudit命令：

```
noaudit session;
```

9.6.2 操作审计

影响数据库对象(如一个表、数据库链接、表空间、同义词、回滚段、用户或索引)的任何操作都可被审计。影响对象的可能操作(如create、alter和drop)可以在审计时编成组。这些命令组可以减少建立和维护审计设置值所需管理工作量。

可以审计所有系统级命令并提供命令组。例如，若要审计影响角色的所有命令，可输入命令：

```
audit role;
```

若要禁止这个设置值，可输入命令：

```
noaudit role;
```

审计的SQL命令组列于附录A的“AUDIT sql_statements”项下。每个组都可用来审计影响它的所有SQL命令(见表9-2中相关权限的详细列表)。例如，前面讲到的audit role命令可审计create role、alter role、drop role和set role命令。

除了附录A所示的主要审计选项外，还可以审计所涉及到的独立命令(例如create table)。Oracle还提供下列语句组选项：

CONNECT	审计Oracle登录和注销
DBA	审计需要DBA权限的命令，如grant、revoke、audit、noaudit、create或alter tablespace；以及create或drop public synonym
RESOURCE	审计表、簇、视图、索引、表空间、类型和同义词的Create和drop命令
ALL	审计所有这些命令
ALL PRIVILEGES	上面的所有命令加上delete、insert、update和其他一些命令；见附录A

所有能被审计的操作在数据库中都被指定一个数字代码。这些代码可通过 AUDIT_ACTIONS视图来访问。下列查询显示你的数据库中可用的代码：

```
select
      Action,        /*Action code.*/
      Name           /*Name of the action, such as ALTER USER.*/
from AUDIT_ACTIONS;
```

只要操作代码是已知的，就可以使用 DBA_AUDIT_OBJECT视图来确定对象是如何受操作影响的。下面所示的查询是从 DBA_AUDIT_OBJECT视图中检索登录审计记录。其中列出了所用的操作系统帐户 (OS_Username)、Oracle帐户名(Username)和所用的终端ID(Terminal)。除了执行操作的操作码(Action_Name)外，还可选择对象拥有者(Owner)和对象名(Obj_Name)。对Returncode(返回代码)列进行检查：若是 0，则连接企图成功；否则就报告一个错误数值。登录及注销的时间也同时显示。

```
select
  OS_Username,           /*Operating system username used.*/
  Username,              /*Oracle username of the account used.*/
  Terminal,              /*Terminal ID used.*/
  Owner,                 /*Owner of the affected object.*/
  Obj_Name,              /*Name of the affected object.*/
  Action_Name,           /*Numeric code for the action.*/
  DECODE(Returncode,'0','Success',Returncode),    /*Failure check*/
  TO_CHAR(Timestamp,'DD-MON-YYYY HH24:MI:SS')      /*Timestamp*/
from DBA_AUDIT_OBJECT;
```

如下例所示，也可以用 audit命令的by *username*子句指定审计个别用户。在这个例子中，用户MCGREGOR的所有更新操作都要被审计：

```
audit update table by MCGREGOR;
```

9.6.3 对象审计

除了系统级的对象操作外，还可以审计对象的数据处理操作。这些操作可能包括对表的选择、插入、更新和删除操作。这种操作类型的审计方式与前节所描述的操作审计非常相似。唯一不同之处在于，audit命令中增加了一个新子句。

对象审计附加的子句是 by session或by access子句。这个子句指定一个审计记录是为每个会话(by session)写入一次，还是每次访问对象 (by access)时都写入一次。例如，如果一个用户对同一个表执行四次不同的更新语句操作，那么 by access审计就会出现四个审计记录 —— 每次访问表都审计一次。而用 by session审计同样的情况，只会有一个审计记录。

by access审计会动态地增加审计记录。所以，它通常只是有限地用在判断在一个指定时间间隔中所发生的独立操作数量；测试做完后，审计应恢复到 by session状态。

这些选项的例子在下面的程序清单中列出。在第一个命令中，对 EMPLOYEE表的所有insert命令都要进行审计。在第二个命令中，对影响TIME_CARDS表的每个命令都要进行审计。在第三个命令中，对DEPARTMENT表的所有delete操作都要进行审计。

```
audit insert on THUMPER.EMPLOYEE;
audit all on THUMPER.TIME_CARDS;
audit delete on THUMPER.DEPARTMENT by session;
```

通过对前节所示的DBA_AUDIT_OBJECT视图进行查询，就可以看到最终的审计记录。

9.7 保护审计跟踪

由于数据库的审计跟踪表SYS.AUD$是存储在数据库中，所以任何写入这里的审计记录都必须得到保护。否则，用户就可能通过非法操作来删除其审计跟踪记录。

向操作系统审计跟踪写审计记录的能力，有助于避免向数据库外部存储记录。但是，这个选项并不是对所有操作系统都有效。

如果必须将审计跟踪信息存储在SYS.AUD$表上，就必须先保护这个表。首先，对这个表的审计操作通过下面命令执行：

```
audit all on SYS.AUD$ by access;
```

若所有操作都是对SYS.AUD$表进行的(通过其他表审计生成的插入不算在内)，这些操作都将被记录在审计跟踪中。不仅如此，对SYS.AUD$的操作也只能由具有CONNECT INTERNAL能力的用户来删除(例如，在DBA组中)。作为INTERNAL连接的任何操作都会自动写入审计跟踪中。

只要有可能，就要调整数据库审计和操作系统审计。这样可以更容易地跟踪问题和调整两种环境的安全策略。由于系统管理员几乎不会去看大量的审计跟踪条目，所以还要促使数据库管理员去精确地分析哪个最关键的操作要审计。其目标应放在每个记录都有意义的审计跟踪上。如果不对，则使用本章给出的命令来修改审计选项，以反映感兴趣的操作。

9.8 分布式环境的安全性

打开一个数据库供其他服务器来访问，同时也使它面对着来自这些服务器的潜在安全威胁。由于这种访问由Net8(和SQL*Net)发出，对Net8参数进行修改就可以防止大多数未经授权的远程访问。有关Net8方面的安全问题细节，请见本书的第三部分。作为一个指导性原则，所有数据访问都必须基于“按需知密”原则。扩充后的原则是：所有对服务器、操作系统及网络的访问都必须基于“按需知密”原则。应该周期性地检查数据库及操作系统级的当前访问权限，与系统管理组一道评估当前网络访问权限。

9.9 解决方案

在Oracle数据库中进行有效的安全管理时，需要解决全部已知的安全问题并审计破坏安全的企图。安全计划至少应包括下述措施：

1) 变更缺省值中SYS和SYSTEM帐户的口令。
2) 定期为全部数据库管理员授权的帐户变更口令。
3) 撤消创建的所有演示帐户(例如SCOTT/TIGER)。
4) 变更DBSNMP帐户上的口令，将新口令放在snmp.ora文件中。
5) 为全部数据库文件设置恰当的保护级别。
6) 如果开发数据库含有产品数据库的数据，要确保为产品数据库制定的安全规则也用于开发数据库。
7) 审计对SYS.AUD$的全部访问。
8) 审计全部失败的连接企图。
9) 审计全部数据库管理员活动。

10) 定期生成审计清单报告并从SYS.AUD$中清除旧记录。

11) 保护数据库备份安全。

12) 保护存储数据库服务器和备份的物理设备的安全。

只要遵循这12条规定，就可以保护数据库的安全。如本章前面所述，还需要与系统管理组和网络管理组一道清除对服务器操作系统的任何非法访问。