第9章　禁止非法使用

安全是人类的基本需求。从表面上看，一个数据库应用程序决不会发生使安全受到威胁的事情，但也必须提供一定程度的安全性以防止被破坏。想达到的安全级别正是应用程序采取的安全策略。

9.1　为什么要求安全

安全性的含义是什么？为什么在这上面要花费时间呢？ Oracle或Oracle Developer能自动地保证数据的安全吗？如果系统是通过因特网提供公众使用的，那又如何保护系统以防止非法入侵呢？

下面是由于某些人的故意行为引发的一些事情：

- 不可应用性：系统可能会拒绝合法用户的使用，这可能是因为某些人破坏或使系统过载造成的。如果应用程序是通过因特网使用的，这种可能性更大。
- 不可统计性：由于无法确定数据项的来源，会使系统不能保证应用程序或数据库中数据的准确性或合法性。例如，当一个应用程序通过 Web页获得数据时，世界上的任何人都可以向其输入数据，这就必须有一种方法知道数据的来源。
- 不可控制性：使系统不能控制应用程序的操作，导致不能保证产生的结果的合法性。
- 窃取：有的人会通过伪造或欺骗的方法窃取数据，甚至货币和商品，这也包括窃取智力情报。然而，因特网使所有这一切变得非常容易。

本书并不是一本关于计算机安全性的书，计算机安全是一个具有很多方面复杂问题的广阔领域。在Oracle Developer软件系统中如果没有保护系统来防止被破坏，则可能会发生以上的一些情况。那么，在这方面如何做得更好一些呢？

首先，必须确定来自于故意破坏软件行为的风险。对每一种风险评估失败的可能性及影响，对于那些认为不可接受的风险，采取减少风险的措施，通过这样的分析就可以确定哪些是安全问题，哪些不是。如果认识到很多问题对系统并没有影响，就可以排除掉很多的“安全问题”。决不要陷入这样的陷阱：为了安全，每件事情必须是安全的。

注意　必须确定一些非故意行为破坏系统所带来的一些风险。这种风险的一个例子是：由于查询语句查询非常巨大数量的数据而使系统瘫痪。数据库及应用程序设计应当考虑到这种技术性失败的风险。安全性风险是另外一种猛兽。

其次，应采取的方法是停止那些对软件的未授权使用，并且停止有可能导致非授权使用的授权使用。如果认为必须通过安全措施保证数据的安全性，决不要欺骗自己：使用安全策略可以保证防止某些人使用软件。根据所处理的问题的不同，有可能使得合法的用户也较难访问和使用软件。在采取这样的方法之前，要判断它是否会降低产品的质量。

第三，可能扩展的选择是在取得审查 (auditing)的事实之后提请法庭调查。审查是数据库或应用程序中的一种跟踪事件及活动。建立一种审查跟踪可以帮助及时地调查及处理各种发生的问题，并且可以防止由试探性入侵者所产生的各种问题。

保证安全性的通用技术是非常广泛的，本章开始所讨论的问题主要着重于 Oracle及Oracle Developer所采用的自主性访问控制和审查。下一节讲述一种更安全的方法，叫做强制访问控制，对于更高安全性要求的应用程序可以使用这种方法。

最后，人们对应用程序的物理访问方法可能决定它真正的安全性。在风险分析及管理计划中必须考虑到物理手段及软件手段。如一个安全性非常敏感的应用程序不能通过公共电话线，当然也不能通过因特网使用。必须限制对运行软件机器的物理访问方法，包括网络连接。

这本书并不要求每个数据库应用程序是绝对安全的，事实上恰恰相反。一个公共的安全模型是限制用户只能访问为完成工作所必须的内容。虽然这样可以产生较好的安全性，但结果是不利于方便使用。如果是为因特网制作程序，这样的方法是不可用的。强加于用户的安全性越少，应用程序将是越易使用的。为满足应用需求牺牲多少易用性及采用多少安全性完全由自己决定。尽可能满足能承受的风险要求，但并不是所有一切。

对安全性访问要求较高的是因特网。如果准备把应用程序发布在因特网上，必须确信应用程序没有违反防火墙的要求，并且不允许黑客 (或任何其他人)进入数据库。 Oracle Developer支持各种防火墙，通过外部防火墙的运行及经过安全的 NET8通道可以访问防火墙另一边的各个服务器。

9.2 自主性访问控制

自主性访问控制(discretionary access control)是控制访问应用程序及数据库的一种能力，它通过给使用应用程序及数据库的主题 (用户及进程)授权的方法实现。这一节概述在 Oracle Developer中可用的各种不同的自主性访问控制方法，给出了标准 SQL安全性控制与Oracle提供的标准之外的安全性控制的区别，并且详细给出了在 Oracle Developer中可用的实现自主性访问控制的不同方法。

注意 下面几节中描述的很多安全性方法是基于服务器的，不是Oracle Developer的内容，然而这些概念很重要，本章在某些细节给出了解释。更详细的内容可以参考关于SQL的书或Oracle文档(参看下面各节中的特殊引用)。

9.2.1 SQL访问控制

ANSI SQL 标准提供了基于Oracle安全系统的最基本的访问控制级别。如果要开发一个使用ODBC的可移植数据库的应用程序，必须使用 ANSI 安全技术加以限制，否则就可以无限制地使用Oracle的各种扩展功能，或任何其他想使用的数据库管理器。

1. 模式及授权

ANSI 安全性方案起始于模式及相对应的授权标识符。在 ANSI SQL中模式(schema)是一个拥有特殊表集、视图集、权限集的对象(视图集及权限集将分别讨论)。任何特殊的表、视图、权限只能属于一个模式。每一个模式有一个名字，称作授权标识符 (authorization identifier)。这个名字也可用于标识一个表，因为不同的模式可以包含具有相同名字的表，也就是说，模式为它所有的对象的名字提供一个范围。在 Oracle中，这些术语等同于用户 (user)及用户名(user name)。Oracle也允许为这些模式提供一个口令(在下面关于Oracle的扩展功能中将会学到更多内容)。

授权标识符是分别访问数据库不同部分的基本方法，应用 SQL的这种特征，根据不同访

问控制可以把数据库分给各个模式。在模块及模式之间创建关系可以构造复杂的访问控制。例如，在一个中心的模式中创建基本应用程序数据，然后为使用该应用程序的不同人员创建不同的用户，为他们授予合适的访问数据级别，这种方法是十分普遍的。

模式及授权标识符通过把数据库分成各个部分，而不是将数据库看作一个整体的方法，可以实现限制访问数据库中的某一部分对象的安全性目标。

2. 权限

ANSI权限(privilege)用一个特殊的授权标识符来表示对表或视图的操作的分类，共有五种操作(actions)：

- INSERT：允许被授权者在表或可修改的视图中插入新行。
- DELETE：允许被授权者在表或可修改的视图中删除行。
- SELECT：允许被授权者从表或视图中的读数据行。
- UPDATE：带有一个可选择的列表，允许被授权者改变表或修改视图中所指定列的值。
- REFERENCES：带有可选择的列表，允许被授权者在一个完整性约束中参照这些列。

可用GRANT语句定义一个权限，格式如下：

```
GRANT <privileges> ON <table name> TO <grantee> [{, <grantee>}...]
  [WITH GRANT OPTION]
```

ANSI安全系统通过把授权标识符与一个称作模块的抽象对象相联系的方法运行，这个模块包含针对数据库执行 SQL。要执行模块中 SQL语句，必须对授权标识符授予必要的权限。这种方法对数据库中的表及视图提供了一个基本的访问控制级别。

WITH GRANT OPTION子句允许被授权者授权给另一个被授权者，否则只有授权标识符的所有者可以为另一个对象授权。

Oracle及1992 SQL标准增加了一条取消权限的 REVOKE语句，用该语句可选择性地逐级取消被授权者的权限。1992标准还将它扩展加入到模式中的所有对象的权限 (范围、字符集等)，对INSERT权限增加了列表，对各列还可以带 Default说明缺省值。1992标准为新的对象增加了一种新权限USAGE，提供使用范围(domain)、字符集、转换(translation)、校勘(collation)的能力。

使用模式，可以达到如下的安全目标：

- 利用对其他授权标识符的授权，可以限制对特定表、列、视图的访问。
- 利用几种类型的权限，可以限制对某些表及视图的访问种类。

3. 视图

视图是一个看起来像表一样的对象，但事实上它是一个定义于该表的 SELECT语句。视图可以从一个或更多基本表或视图中导出数据，某些视图是不可修改的，如带有连接、分组或表达式的SELECT语句。

为了安全性目的可以使用视图，因为它可以压缩地引用基本表，当使用它时不需要有关基本表的任何权限。也就是说，可以在一个模式中创建基本表，创建该表上的一个视图，并仅把该视图的权限授予其他的授权标识符。

使用视图，可以达到下面的安全目标：

- 通过视图定义中的SELECT列表，可以限制访问一个或几个基本表的某些列。
- 通过视图定义中的WHERE子句，可以限制访问一个或几个基本表的特定行。

9.2.2 Oracle 访问控制

为了为数据库服务器提供一个完整的自主性访问控制系统， Oracle采取了几种不同方法扩展ANSI SQL标准。

对于Oracle的访问控制及审查 (auditing)功能的详细描述参看 Oracle Server Concepts Manual(Oracle服务器概念指南)。

1. 认证

认证(Authentication)是验证一个用户是否与其声明内容相符的过程，以便保证能够合法使用授予给该用户的权限。正像前节所提到的， Oracle也为授权标识符提供了一套口令系统，在Oracle中称作用户名。 Oracle为授权标识符提供了更进一步的应用，并在 SQL中增加了CREATE USER命令。该命令可以创建一个带有口令的用户 (同时定义了一个不含任何对象的模式)，同时指定缺省及临时表空间、空间限额及配置文件 (下一节讨论表空间及配置文件)。

Oracle提供两套独立的口令机制，第一套假定操作系统提供了一套口令方案，并且对于Oracle会话不使用口令检查。

```
CREATE USER <user name> IDENTIFIED EXTERNALLY
```

第二套方案将口令以加密的形式存在服务器上数据库数据词典中，无论何时启动一个会话都会检查它。使用CREATE USER语句，在任何数据库中可以使用任何一种方案或同时使用两种方案。

```
CREATE USER <user name> IDENTIFIED BY <password>
```

如果用EXTERNALLY，Oracle使用一个标准的前缀，通常是“ OPS$ ”，加在操作系统用户标识符前。因此在使用 CREATE USER命令时，必须在<user name>中加相同的前缀。对于特定系统参看OS_AUTHENT_PREFIX初始化参数。

无论采用哪一种方案，只有经常改变口令才会好用。用户如果不能很经常地改变口令，至少也应每个月改变一次。口令应当是单词或不具有明显意义 (如名字等)的杂乱的符号，并应包括奇异的符号 (如$、#、_等)。人们不应把口令写在黄色的胶纸上并贴在终端上，也不能把口令告诉街上的过路人，或把口令放在电子公告板上。系统验证安全的强壮性取决于使用该系统的用户的可信程度。

当有人启动一个会话时， Oracle 的验证方案完成验证该用户身份的安全目标。

2. 角色

一个Oracle角色(role)是一个权限集合。可以把多个权限授予一个用户，或者把一个权限集合授予一个角色，然后把该角色授予该用户 (或多个用户)。这种方法提供了多个不同用户管理非常复杂的权限组合的能力。

例如， Talbot 雇了一个新的会计师，则仅仅把他或她的用户名字加到 Accounting Role(记帐角色)中即可。如果一个应用程序需要一个新表及在该表上的权限 ,Talbot 仅需改变角色的权限，并且所有具有该角色的用户都会得到正确的权限。

要建立一个角色，必须首先确定安全目标。每一个角色应当是表示某一功能的连贯的权限组。例如，每一个应用程序中都会有一个建立好的唯一角色，它包含运行该应用程序所必须的所有权限。对于一个复杂的应用程序，可能需要划分几个使用该程序的不同用户组，每一个用户组可能会得到不同并且是相互交错的权限集，对每一个用户组可以授予不同的角色。这些应当作为应用程序需求分析的一部分，只有划分好了不同角色，才会比较容易地确定这

些角色的访问需求。

也可以把角色授予其他角色，这样可以在层次状的角色中，将一个角色与附加的权限或其他角色进行组合。

可以通过在CREATE ROLE语句中带有IDENTIFIED BY分句的方法，对角色实施口令保护。这种方法使得在获得角色的权限之前，用户需要先输入角色的口令激活该角色。这样对角色增加了更进一层的保护，但显然加大了用户注册的难度，只有那些具有重要风险需要验证的事情，才需进行角色口令保护。

Oracle还提供了一个特殊的关键字 PUBLIC，用于表示数据库当前定义的所有用户，该用户组缺省可以访问数据字典的表。通过给 PUBLIC授权，可以把权限授予所有用户，这在有时是非常有用的。还可以把同义词(下一节描述)和数据库链结创建成PUBLIC形的同义词或链接，这意味着所有用户都可以引用它们。

角色应用于安全目地可以更容易地管理复杂的权限集合，可以完成更精细的访问控制，维护这种安全结构只需做很少的工作，这也达到了限制访问的目标，即限制用户只需访问为完成本职工作所必需的数据。

3. 系统及对象权限

Oracle 在服务器安全系统中增加了几种不同种类的权限，新增权限的主要原因是由于Oracle在数据库标准表和视图之外附加了一个模式 (schema)对象扩展集：

- 同义词(Synonym)：任何对象名字的别名。
- 簇(Cluster)：是一种把共享公共信息的多个表组织在一起存储的存储结构，例如与外部关键字完整性约束相关的表。
- 索引(Index)：对表中的数据提供可选择的访问路径的辅助的存储结构。
- 序列(Sequence)：可以用于产生唯一的整数值作为主关键字的对象。
- 过程(Procedure):一个PL/SQL存储过程或函数。
- 触发器(Trigger):与作用在表上的服务器事件相联系的存储 PL/SQL过程，如在UPDATE或INSERT之前或之后的事件。
- 快照(Snapshot):保存从主表(通常是远程数据库上)查询结果的一个表。

系统权限(system privileges)允许管理自己的模式及它的操作，这通过执行特殊对象类型上的特殊操作完成。获得这种权限后，可以使用CREATE、ALTER及DROP命令从模式中增加、改变及删除任何不同类型的对象。在权限上增加关键字 ANY，意味着不仅可以在自己的任何模式，而且可以在不属于自己的模式上实施权限。

除了管理模式对象的权限以外，还有一些特殊的系统权限，这些权限主要是用于数据库管理，但是偶尔也可以把它们授权给应用程序角色，这在表 9-1中解释。

只有当另一个用户给你授予带有ADMIN OPTION的权限或具有GRANT ANY PRIVILEGE权限时，才可以把系统权限授予别的对象。

对象权限(object privilege)是对一个特殊的、现存对象采取某种动作的权限。表 9-2描述了这组权限。

表9-1 Oracle系统权限

权 限	描 述
BECOME USER	允许从任何模式中输入数据

(续)

权 限	描 述
BACKUP ANY TABLE	允许从任何模式中输出数据
EXECUTE ANY PROCEDURE	允许执行任何函数或过程或引用任何可用的公共程序包
FORCE [ANY] TRANSACTION	对两阶段提交中值得怀疑的事务允许强制执行 COMMIT 或 ROLLBACK
GRANT ANY PRIVILEGE	允许授权给任何系统或对象权限
GRANT ANY ROLE	允许把任何角色授权给任何用户
SELECT/INSERT/UPDATE ANY TABLE	允许查询、插入、修改任何数据库表中的数据
LOCK ANY TABLE	允许对任何数据库加锁
MANAGE TABLESPACE	允许对表空间进行在线及离线备份
READUP	允许用比当前会话高的访问类别查询数据 (看下一节的强制访问控制)
RESTRICTED SESSION	允许在受限服务器访问模式下注册
SELECT ANY SEQUENCE	允许引用顺序值
UNLIMITED TABLESPACE	允许超过给定的空间限制
WRITEDOWN	允许用比当前会话低的访问类别CREATE、ALTER、DROP、INSERT、UPDATE、DELETE对象(参看“强制访问控制”)
WRITEUP	允许用比当前会话高的访问类别CREATE、ALTER、DROP、INSERT、UPDATE、DELETE对象(参看“强制访问控制”)

表9-2 Oracle对象权限

权 限	表	视图	序列	过程、函数、软件包	快照
ALTER	X		X		
DELETE	X	X			
EXECUTE				X	
INDEX	X				
INSERT	X	X			
REFERENCES	X				
SELECT	X	X	X		
UPDATE	X	X			X

自主性访问控制系统假定开始没有权限存在。因此，必须明确地对一个用户或对角色授权，该角色与在解决问题中能够采取行动的某人的用户相对应。

对于给角色及用户赋权限的SQL GRANT和 REVOKE命令语法的详细讨论，参看《Oracle Server SQL language Reference Manual》或有关Oracle SQL的书，如Koch及Loney的Oracle书：《Oracle 8完全参考手册》。

使用Oracle系统及对象权限，可以达到下列安全性目标：

■ 采取给用户或角色授权的方法，可以限制访问各种不同的 Oracle特殊对象。

■ 使用几种类型的权限可以限制某些对象的访问种类。

■ 可以限制对数据库服务器及数据库实际所需的维护能力。

4. 配置文件

Oracle 配置文件(profile)给出了一种控制用户在一次会话中耗费资源的方法。 Oracle建有一个缺省的配置文件，它可以用于所有用户，如果设置初始化参数 RESOURCE_LIMIT为TRUE，就可以定义并且使用附加的配置文件。可以使用 CREATE USER语句为一个新用户指定一个特殊的配置文件，或者用 ALTER USER语句把配置文件指定给一个已存在的用户。

CREATE PROFILE 语句允许定义一个配置文件，在文件中可以设置限制用户在一次会话中所用到的资源为一个或多个，这些限制在表 9-3 中解释。

表9-3 Oracle配置文件资源设置

资源限制	描 述
SESSIONS_PER_USER	限制用户某一特定的并发会话数。例如，如果允许每个人用相同用户名注册到应用程序，就可以使用这个参数来限制对某一应用程序的访问用户数
CPU_PER_SESSION	限制一次会话可以使用的CPU时间数
CPU_PER_CALL	限制一个特定的分析、运行或取数据的调用可以使用的CPU时间数
CONNECT_TIME	限制一次会话的结束时间
IDLE_TIME	在一定的空闲时间后断开会话(在查询执行期间不计时)
LOGICAL_READS_PER_SESSION	限制会话可以从数据库中读数据块的数量
LOGICAL_READS_PER_CALL	限制在一次分析、运行或取数据调用期间，会话可以从数据库中读的数据块数
COMPOSITE_LIMIT	限制在一个复合单元中一次会话总的耗费
PRIVATE_SGA	限制一个会话可以在SGA中分配为私有空间的总数量

可以使用Oracle的审查功能(在本章的“审查”中讨论)及Oracle的进程管理和跟踪功能来收集不同种类用户的资源信息。然后应当分析与这些资源相关的风险，并且决定采取什么样的策略来限制风险。

对服务器资源的交互使用也会有风险，例如由于资源的耗尽会使应用程序过载或破坏应用程序，用户配置文件可以达到限制这种风险的安全目标。

5. 表空间及空间限额

配置文件允许控制交互作用的资源，但对存储资源又怎样呢？在 CREATE USER语句中给出表空间的使用限额，所以有能力控制允许用户在数据库中使用的存储空间数量。

表空间(CREATE TABLESPACE)是一组操作系统文件及在磁盘上创建表、索引、簇的存储规范。

空间限额是用户可以在一个特定的表空间上创建的空间数量。

使用表空间，可以把数据分布在物理数据库的特定部分。使用不同的 CREATE语句，可以在特定的表空间上创建属于各用户的对象。使用空间限额，可以限制某一用户在一特定的表空间上针对它所创建的对象增加数据的数量。

例如，如下语句在缺省的表空间上创建 Talbot用户并指定了空间限额：

```
CREATE USER Talbot IDENTIFIED BY GeorgeDEFAULT
   TABLESPACE LocalTables
   QUOTA 5M ON LocalTables
   PROFILE CentralAdministration;
```

空间限额达到限制与物理存储资源耗费有关的风险的目的。

9.2.3 客户安全

列的更新控制及菜单访问控制为 Oracle Developer提供了一些其他必需的安全形式。前者允许通过Oracle Developer控制对列的修改，后者使用定制菜单特征允许 Oracle角色激活或关闭菜单项。

注意 这两种特征都依赖于Oracle，如果有数据库移植性的要求，任何一种特征都是相

关的。

1. 列的更新修改

在第5章的注册过程节详细给出了开始设计一个表单的步骤，有些步骤是设定列安全性的。

块(block)的Update Allowed(允许更新)属性允许控制是否让用户修改块记录，修改一条记录表示要查询该记录，然后改变一个或多个项。如果设置 Update Allowed为False，则用户就只能查询记录但不能改变它们。

项目(item)的Update Allowed属性用来控制是否可以对记录的一个特定列进行修改。如果块属性设置为真，则可以设置一个或多个列属性为 False，以防止用户改变这些列的值。项目的Update If Null(为空时更新)属性可以防止用户对任何值的修改，除非值为 NULL。这意味着用户可以键入值，但是一旦离开该项目之后就不能再改变它的值。例如，在一个工程管理系统中，可以把任务的结束日期设为 Update If Null，当输入一个结束日期以后，表示任务完成，并且系统不允许以后的用户改变该日期。可能有用户不喜欢这种限制的反映，但有时安全性是件不受人感谢的工作。

这种方法达到了这样一个目的，即防止通过表单的特定的块改变表的行或值。可以在设计表单时设置这些属性，也可以在运行时通过调用 PL/SQL触发器改变属性设置。

Oracle Developer通过列安全性(Column Security)方法可以做一些工作。可以设置表单的列安全属性为True，当一个用户注册到该表单时， Oracle Developer在数据字典中查询表单中块的表及列的修改权限，对于没有 UPDATE权限的用户的所有列，它把Update Allowed属性设为False。用这种方法可以利用服务器上设置的权限。

2. 菜单访问控制

如果创建自己的菜单，可以使用 Oracle Developer的菜单定制特征，通过 Oracle安全性角色关掉或删除菜单项。甚至可以使用 Oracle Developer的对话框设置安全性角色，并把角色授权某个特定用户(利用File|Administration|Database Roles菜单项)。不管怎样，在大多数情况下，应使用SQL脚本，或是通过对Oracle数据库的管理接口完成该功能，这样更容易看到发生了什么。在定义角色的同时，应当在数据字典中安装 FRM50_ENABLED_ROLES视图，并且把视图的SELECT权限授予要用安全菜单开发应用程序的所有用户。为了完成这些工作，必须使用Oracle Developer Build和Admin组中的Grant脚本安装Forms所有数据库表。

一旦定义好角色，可以进入 Object Navigator的菜单模块，在菜单属性列表中选择 Module Roles属性，点击More按钮可以打开对话框。现在就可以输入想对该模块执行控制菜单访问的所有角色。在该对话框中，有本章的 9.5.3节“ Talbot应用程序安全性 ”将要创建的角色 (见图9-1)。

注意 这个对话框中包含有一个简单的文本列表，在从数据库的模式中取得角色时它不能给予任何帮助，当设置该列表时，可以先写下所有想输入的角色，以免在输入列表时忘记。

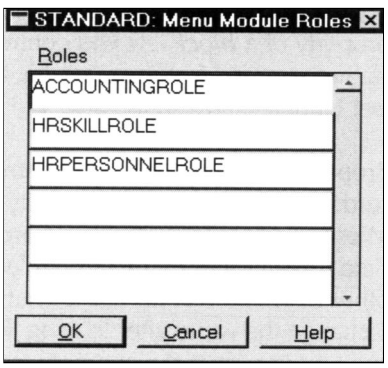

图9-1 Modole Roles对话框

现在，在Object Navigator中找到想进行角色控制的菜

单项，找到该菜单项属性列表中的Item Roles属性，点击More按钮显示一个对话框，在对话框中可以交互选择激活菜单的角色。当单击鼠标选中一个角色时，它表示被授权该角色的用户可以在菜单条中看到该菜单项是激活的，没有被授权该角色的用户则看到该菜单项是被关掉的。当点击鼠标时按住 SHIFT键，可以连续选择几个项目，当点击鼠标时按住CTRL键可以选择或不选择一个特定的项目，而不管其他各项目的状态(见图9-2)。

图9-2 在对话框中选择激活菜单的角色

Display Without Privilege属性控制Oracle Developer是否可以关掉菜单项或完全删除它，值为 Yes表示关掉它， No表示删除它。现在把菜单属性列表的 Use Security标志设为Yes，选择File|Administration|Compile File重新生成菜单，所有一切就完成了。使用Oracle Developer的这个特点，通过 Oracle 安全特征就可以自动实现安全措施。

注意 如果想开发并测试应用程序而不受菜单的安全性影响，让 Use Security值为No，直到准备发布应用程序为止，到时把它的值改为YES并重新生成菜单模块即可。

9.3 审查

审查(Auditing)是调查一个数据库系统及其用户活动的过程。可以监视数据库的活动，跟踪这些活动直到它们的起源，或收集关于活动的统计信息。也可以记录有关历史信息以确保能够重新构造一个事件链。所记录的信息 (不管信息属于什么类型)叫做审查跟踪(audit trail)。

9.3.1 Oracle中的审查

Oracle提供如下扩展审查功能：

- 语句审查(Statement auditing)：跟踪一个或多个用户某一特殊类型语句的执行。
- 权限审查(Privilege auditing)：跟踪一个或多个用户与某一特定系统权限相关的活动。
- 对象审查(Object auditing)：跟踪数据库所有用户对某一特定的模式中对象所执行的所有语句。

所有这些选择允许把审查跟踪存在表 AUD$中，该表属于SYS用户。可以通过数据字典视图访问这些信息。有关数据字典支持审查的详细信息请看《 Oracle Server Administrator's Guide》。

审查是一种检查安全策略是否达到预期效果的方法。如果确认某种风险是非常严重的，就可以审查各种可能引起系统失败的活动。如果系统确实失败了，就可以确定引起系统失败的活动序列，或者可以跟踪到引起失败的可能不合法的根源，并且处理这种根源。通过查找审查跟踪也可以隔离各种错误，如果合适，可以审查到各种成功或不成功的语句。当遇到一个问题并且它几乎没有相关信息时怎么办呢？审查可以提供相关信息。

最好懂得审查的影响，可通过研究不同的选项和进行实验，懂得在应用程序中如何应用于安全性问题。

审查可以达到总的安全目标，它可以保证满足其他各种安全目标。

9.3.2 客户审查跟踪

Oracle可以审查数据库的活动，但它无法扩展到去审查客户应用程序中各种有意义的事件。用户根据自己的安全观点，可能想在应用程序中建立附加的审查。

一个非常广阔的区域就是应用程序审查 (application auditing)。这是把事务历史当作应用程序数据一部分的一种标准办法。如，在一个记帐应用程序中，必须保证能记录所有对数据修改的审查跟踪，在任何一给定时刻，必须能对某一值重新构造出它的正确值，而不管这个值在数据库中已经发生了什么变化。必须能将每一个对象与现实世界的对应物品相联系，如一个货物清单记录必须与货架上的实际货物相对应。所有这些需求在为应用程序设计数据库时必须是明确的。

还可以有一些基于某种特殊安全论点的需求，这叫做论点审查 (issue auditing)。如，可能有一些有关事务的论点对应用程序是非常重要的，想验证一下在所有环境下事务是合法的并且成功，就可以把事务的跟踪存储在分离的审查跟踪中来验证。这些是Oracle提供的大致功能，但它可以应用于应用程序中有意义的事件，而不必用于 Oracle审查已提供支持服务器事件。

一般地，由于审查跟踪的这种直观特性，可能希望把审查及跟踪表封装在独立的数据库模式及过程中，以便客户不必干预审查过程。参看前一节关于模式和授权、认证及过程封装，也可看第13章关于跟踪的进一步讨论，在第 13章可以跟踪SQL执行，可以了解一个应用程序系统在运行时大量的其他信息。

9.4 强制访问控制

强制访问控制(mandatory access control)是一种对于数据库对象实行非常强的访问控制的安全系统，它给每一个对象及会话加一个安全性标号，标号提供了一种分级方案，允许服务器基于分级的级别而不是特定的权限来决定是否授权访问某个对象。

为什么这样的一个方案是必要的呢？在自主性访问控制下，数据仅仅是数据，如果把数据移到另一个地方，数据的权限是由它所在的地方决定的，而不是数据自身。例如，一个用户对某一个表具有 SELECT权限，但不带GRANT选项，该用户仍然可以把数据拷贝到另一个表中，这个表是他事先建好的具有相同结构的一个表，并且可以授权其他人访问这个表。有很多类似的情况，限制了自主性访问控制管理安全风险的能力。

强制访问控制通过把安全约束与数据相联系的方法来处理大多数的情况，当拷贝数据时，也同时拷贝了安全约束。可以使用权限完成这些，但用分组方案的安全级别处理起来更容易些。

Trusted Oracle

Oracle提供了强制访问控制方案，使得 Oracle与U.S.B1 TCSEC标准及欧洲ITSEC标准的E3安全级相一致。这是一种可选择产品叫做 Trusted Oracle(委托Oracle)，这种系统只能运行在提供类似特征的操作系统上，这种特征与安全标准一致。例如， Windows和Macintosh不能用做Trusted Oracle的服务器，也不能把应用程序放在 WWW(World Wide Web)上，除非浏览器是运行在一种安全的操作系统下。

关于Trusted Oracle中强制访问控制方案的详细信息，请看 Trusted Oracle Server Administrator’s Guide。下面概述一下，它是如何工作的。

在这种方案中，每个标号(label)有分级部分、类别部分和标记部分。

1. 分级组件

分级(classification)是一个层次级制，它确定被标记信息的敏感性。可以在操作系统级定义分级及顺序，例如：TOP SECRET、SECRET、SENSITIVE、UNCLASSIFIED。

2. 分类组件

分类(category)是对分级的一种划分，它由分级区域组成。即使在分级方案中，也允许限制访问一个特殊的区域，例如，可以让每一个工程具有类别，类别是按工程来划分级的，只有标号中含有工程类别的会话才能看到相关工程的信息。

3. 标记组件

标记(marking)是在一个分级中更详细的一组信息，它在不同环境下与数据相联系。例如，可以把一个特殊的对象标记为 PROPRIETARY，应用程序可以保证打印出所有带有 PROPRIETARY的对象，同时带有一条关于传播的特殊警告。

当注册一个会话时，强制访问控制系统是通过将对象的标号与会话的标号进行比较来工作的，会话的标号是通过操作系统的安全通行证建立的。这种比较是通过支配过程工作的：一个标号的分级如果大于或等于另一个标号的分级，它就可以支配 (dominate)另一个标号，并且它的类别是另一个标号类别的超集。如果一个标号支配另一个标号但不匹配，则它是严格的支配另一个标号。如果是由于类别和分级的区别，使得没有标号能支配其他标号，则标号是不可比较的。安全许可证 (security clearance)是被授权读及写信息的标号范围。只有当标号可以支配对象的标号时，才能读对象，只有当标号匹配了对象的标号时，才能写对象。

与这些约束相关的有几个Oracle系统权限：

- READUP ：允许用严格支配用户标号的标号读数据。
- WRITEUP：允许用严格支配用户标号的标号写数据。
- WRITEDOWN：允许用用户标号严格支配的标号写数据。

关于系统及对象权限的说明及利用系统及对象权限进行自主性访问控制的更多信息，参看9.2节“ 自主性访问控制 ”一节。

总之，如果需要非常的安全，政府或其他顾客一定会要求应用程序与 B1或其他安全标准一致。操作系统及Trusted Oracle的结合，能够利用Oracle Developer去开发这样的程序。注意，不管怎样， Oracle Developer必须运行在可靠的操作系统上以与安全标准相一致。

9.5 保证Talbot系统的安全

作为使用Oracle Developer及Oracle安全性的一个例子，本节从头至尾考虑 Talbot安全策略的开发。这个例子很简单，而一个真正的安全策略将是很复杂的。

9.5.1 Talbot的安全风险

对Talbot计算机操作的风险的分析确认了几种可能的安全风险。

对于Talbot来说，系统入侵及智能窃取的主要风险是不存在的。 Talbot农场的工作站主要分布在农场带锁的办公室里，数据库是物理上分离的，除了农场工作站外，没有人能访问到。通过运行在局域网上的因特网有可能入侵网络，但在非安全区域是可接受的，并且分析表明，即使系统完全失败，影响也是很小的。 Intranet有防火墙保护，它会严格控制通过因特网及调

制解调器的外部访问。在这样的情况下，风险评估小组决定，每天把备份数据存贮在一个安全位置，并制定一套灾难恢复策略，足以能够处理由于入侵造成系统失败的风险。他们还决定，透露给竞争者信息的影响是很小的，不需要另外的措施保护数据，以防止未授权访问。

因为Talbot使用的是一个不提供安全保护的操作系统，风险评估小组决定主要依靠数据库服务器的安全：即Oracle用户名及口令。

与数据库内容有关的风险包括如下三种可能性：

- 修正或改变Ledger入口有可能导致审查问题，更会导致税务审查及法人审查问题。
- 要求工人保存好自己的地址必须有规章制度。如果一个地址通过广告信息泄露了或缺乏安全措施，则风险主要是规章制度惩罚或对雇员的法律诉讼。
- Talbot维护数据库的经验建议，给每一个人完全访问权可能导致数据库的不准确，进而是不可用的。

风险评估小组考虑了两种管理Ledger修改的可能方法。第一种是双份记录对数据库 Ledger的修改，以维护一个审查跟踪。这种方法的优点是 Ledger总是尽可能正确，而且审查者可以从审查跟踪重新构造修改的历史。第二种可能的方法是，要求改正独立的 Ledger入口，并要求各入口是正确的。这种方法的优点是减少了复杂性，减少了系统开支，并且审查容易 (第二种方法在实际记帐业务中是一种标准方法)。小组决定，不允许修改 Ledger入口，并且要求所有修改必须通过附加的入口。做为一种附加安全性，小组决定把 Ledger表从其他表中分离出来，放在一个带口令保护的模式中。

风险评估小组注意到计帐工作的一部分是在 Person表里维护支出信息，这不会扩展管理地址(后面将做讨论)。

风险评估小组决定把 Ledger表分离放在单独的模式中，以确保它尽可能的不受入侵，只有授权的人事部的人才能访问这些数据。

其他表(Person、Skill、WorkerHasSkill)都放在一个模式中，人事部具有操作访问权，其他人只有查询访问权。

9.5.2 Talbot的Oracle安全性体系结构

图9-3给出了起初的UML类框图，对于不同类别的附加模式信息用容器型的方框表示，每个模式对应一个特定的Oracle用户名和口令。

每个实际的系统用户将会有一个具有不同口令的 Oracle帐号，安全小组建议采用标准的口令维护过程(每个月改变一次口令等)。

与这些要求相对应的角色是直观的：

- Accounting(记帐)：要求在Ledger表中能查询和插入但不能修改或删除，能对 Person表进行查询、插入修改及删除，但不能删除或修改个人的地址入口。
- Address：拥有并管理私人地址表。
- HR Skills：要求能插入、修改及删除工作能力，以及人员和工作能力之间的关系。
- HR Personnel ：要求能对Person表插入、修改及删除信息。

除了这些角色外，还有标准的DBA角色，主要是Talbot用于创建模式、用户、角色、权限等等。

为了创建数据库，Talbot用SQL*Plus以SYSTEM的身份注册到Oracle,并执行如下六个语句：

```
CREATE USER Human_Resources IDENTIFIED BY FT3R$27SD;
GRANT CONNECT, RESOURCE TO Talbot;
CREATE USER Accounting IDENTIFIED BY TYB##5X$5;
GRANT CONNECT, RESOURCE TO Accounting;
CREATE USER Lodging IDENTIFIED BY M45C$BHI_9Z;
GRANT CONNECT, RESOURCE TO Lodging;
```

然后，Talbot就可以连接到每个帐号，通过运行带有 CREATE SCHEMA语句的脚本为记帐程序创建模式。当创建外键并引用另一个用户创建表时，必须确信在那个用户中已经创建了这些表。因此必须以一个特定的顺序创建所有表，以允许外键定义。另外也可以创建所有表以后，用ALTER TABLE添加约束的语法来增加所有外键。

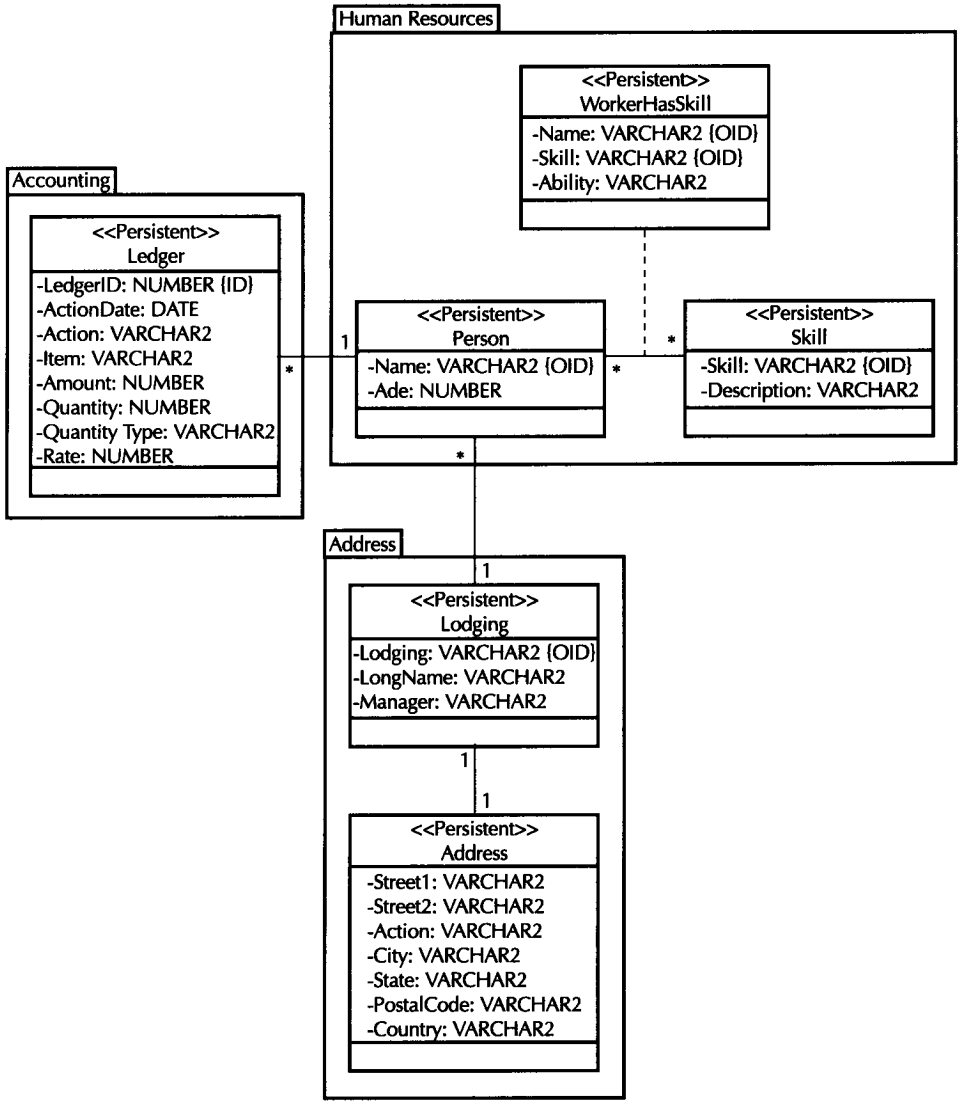

图9-3 Talbot数据库的安全性模式

首先按如下方法创建Lodging模式：

```
CREATE SCHEMA AUTHORIZATION Lodging
```

```
  CREATE TABLE Lodging (
    Lodging   VARCHAR(15) PRIMARY KEY, /* short name for lodging */
    LongName  VARCHAR(40),             /* complete name */
    Manager   VARCHAR(25),             /* manager's name */
    Address   VARCHAR(30)              /* address of the lodging */);
GRANT REFERENCES ON Lodging to Human_Resources;
```

上面最后一个语句把 REFERENCES权限授予了 Human_Resources用户，允许该用户在Person表中建立参照Lodging表的外键。下面创建Human_Resources模式：

```
CREATE SCHEMA AUTHORIZATION Human_Resources
  CREATE TABLE Skill (
    Skill         VARCHAR(25) PRIMARY KEY, /* name of a capability */
    Description   VARCHAR(80)              /* description of the skill */
  )
  CREATE TABLE Person (
    Name          VARCHAR(25) PRIMARY KEY,       /* worker's name */
    Age           INTEGER,                       /* age in years */
    Lodging       VARCHAR(15) REFERENCES Lodging.Lodging
                       /* reference to short name of lodging */
  )
  CREATE TABLE WorkerHasSkill (
    Name          VARCHAR(25) REFERENCES Person, /* worker's name */
    Skill         VARCHAR(25) REFERENCES Skill,  /* capability name */
    Ability       VARCHAR(15),          /* how skilled is the worker? */
    PRIMARY KEY (Name, Skill)
  );
GRANT REFERENCES ON Person TO Accounting;
```

像前面一样，最后一个语句允许 Accounting用户在Ledger中参照Person表建立外键。还应该注意到，在Person表中的REFERENCES子句，使用了认证标识符 Lodging作为Lodging表名字的前缀。最后，创建 Accounting模式，并一起创建用做Ledger主关键字值的LedgerSequence顺序值：

```
CREATE SEQUENCE LedgerSequence /* sequence numbers for Ledger */;
CREATE SCHEMA AUTHORIZATION Accounting
  CREATE TABLE Ledger (
    LedgerID       INTEGER
                     PRIMARY KEY,/* sequence number, primary key */
    ActionDate     DATE,         /* when */
    Action         VARCHAR(8),   /* bought, sold, paid, received */
    Item           VARCHAR(30),  /* what */
    Quantity       INTEGER,      /* how many */
    QuantityType   VARCHAR(10),  /* quantity: lbs, bushels, etc. */
    Rate           NUMERIC(9,2), /* how much per quantity type  */
    Amount         NUMERIC(9,2), /* rate * quantity */
    Person         VARCHAR(25)
                     REFERENCES Human_Resources.Person /* who */
  );
```

当所有用户及表创建以后，现在就可以设置角色的权限了。 SYSTEM用户是DBA，已经具有所有的权限。连接到SYSTEM并创建各角色。

```
CREATE ROLE AccountingRole;
CREATE ROLE  HRSkillRole;
CREATE ROLE HRPersonnelRole;
```

现在连接到Accounting用户，把Ledger表授权予Accounting角色：

```
GRANT SELECT, INSERT ON Ledger TO AccountingRole;
```

连接到Human_Resources用户，对Person、Skill及WorkerHasSkill表授权限。

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Person to AccountingRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON Skill to HRSkillRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON WorkerHasSkill to HRSkillRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON Person to HRPersonnelRole;
```

注意 自主性访问控制不能用权限来强行限制Accounting角色修改有地址的用户，而必须用数据库触发器来实行这种约束。也可以用创建一个独立的视图，用WHERE子句去除掉那些数据行的方法来实现这些。然而这种方法不允许应用程序看到那些名字，这与要求查询到所有名字的其他需求相违备。也可以用这个视图进行修改而用基本表进行查询，但这要告诉Oracle Developer数据块如何工作，是非常难的。可能需要两个独立的块，一个用于查询目的，另一个用于查询/修改目的。

创建了所有的角色并授予了所需要的权限以后，现在就可以创建单个用户并授权他们角色。例如为了建立记帐用户和全体人员的用户，用 SQL*PLUS连接SYSTEM，执行下列语句：

```
CREATE USER User1 IDENTIFIED BY JU8##NMY78$;
GRANT CONNECT TO User1;
GRANT AccountingRole TO User1;
GRANT SELECT ON FRM50_ENABLED_ROLES TO User1;
CREATE USER User2 IDENTIFIED BY H#17_TH$5;
GRANT CONNECT TO User2;
GRANT HRPersonnelRole TO User2;
GRANT SELECT ON FRM50_ENABLED_ROLES TO User2;
```

提示 也可以用Oracle企业管理员程序包中的Security Manager，使用一个友好的用户界面创建用户及授权。

注意 给用户授予FRM50_ENABLED_ROLES的SELECT权限是必要的，以便能用Oracle Developer的菜单安全性特征。没有此授权，会看到FRM-10256这样的Forms错误信息：“ 用户没有被授权运行Form Builder菜单 ”。这里假定已经从Administration程序组安装了表单的数据库表，这些表也是必需的。

9.5.3 Talbot应用程序安全性

Ledger表单必须满足要求，不能修改或删除 Ledger表中数据行。 Oracle数据库通过AccountingRole安全角色执行这种限制。如果修改 Ledger表中的值或删除一行，当向数据库保存所做的修改时会出现错误，因为 SQL语句的执行失败了。

尽管这样解决了问题，但这并不怎么好。接口友好的原则是，当出现错误时要立即给出反馈。而按照数据库的安全性，只有在存盘时而不是在实际改变值或删除记录时给出反馈。为了在程序接口中实行立即约束，需要在应用程序中放一些额外的代码。

为了实现这种方法，所需要做的就是在 Ledger块中关掉Update Allowed和Delete Allowed属性。这将停止用户对从数据库查询到的值的修改或删除，并且会立即给出一个标准的 Forms错误信息反馈。

注意 可以把模块中Column Security标志设为True，但在这个例子中，用块的Update Allowed标志似乎更直接。产生错误提示的一种选择就是关掉删除记录的菜单及键盘操作方法(即Record | Remove菜单项及标准的 [Delete Record] 按键)。这样从根本上不允许执行，将产生一个较好的接口，但是涉及到对工作(创建定制的菜单等)的影响似乎要高

于所带来的好处。如果已经有自己的菜单及这样菜单项的按键定义，应当修改它们。

在开发其他应用程序时，在基于 Oracle角色的不同应用程序中， Talbot可能会用菜单安全系统配置各菜单。如，在Ledger表单中可能会通过AccountingRole安全角色允许数据操作菜单项(如创建记录或删除记录)，通过其他角色关闭菜单项。也可能允许 AccountingRole角色的用户查看Ledger总结报告，而其他用户不允许。关于设置菜单安全性的例子及细节参看 9.2.3节中“菜单访问控制”。

总之， Oracle Developer协同Oracle 或其他DBMS一起工作，会提供强有力的安全工具。应用这些工具要花一些心思并仔细准备，但完成后可以安然入睡，因为知道应用程序已经达到了所能做到的安全级别要求。