学习笔记
    用户管理、权限管理、角色管理
在使用 oracle 的过程中对用户的操作有如下几步:
    创建和管理数据库用户账户
    创建和管理角色
    授权,撤销授权
    控制用户的资源使用

1.用户管理
要对 oracle 数据库用户及用户所使用的系统资源和系统安全进行有效管理第一步就是创建 oracle 用户。

1.1 在创建一个用户之前需要检查的事项:
决定该用户必须存储对象的表空间。
决定每个表空间的配额。
赋予该用户默认的数据表空间和临时表空间。
创建用户。
将该用户所需的系统权限和角色授予用户。
选择一个概要文件 profile.

1.2 创建用户格式
```
CREATE USER 用户名
IDENTIFIED {BY 口令 | EXTERNALLY | GLOBALLY AS external name}
[DEFAULT TABLESPACE 表空间名]
[TEMPORARY TABLESPACE 表空间名]
[QUOTA {正整数 ·[K|M]| UNLIMITED} ON 表空间名]
[QUOTA {正整数 [K|M] | UNLIMITED} ON 表空间名]...]
[PASSWORD EXPIRE]
[ACCOUNT {LOCK |UNLOCK}]
[PROFILE {概要文件名| DEFAULT }]

SQL> CREATE USER DOG IDENTIFIED BY DOG DEFAULT TABLESPACE USERS TEMPORARY
TABLESPACE TEMP QUOTA 10M ON USERS;
不限制指定用户使用的表空间和表空间的大小
SQL> GRANT UNLIMITED TABLESPACE TO USER1
```

1.3 查询系统中的用户
查询当前系统中存在哪些用户
```
SELECT USERNAME,DEFAULT_TABLESPACE,TEMPORARY_TABLESPACE,CREATED FROM DBA_USERS;
```
查询用户指定用户所使用的表空间配额
```
SELECT USERNAME,TABLESPACE_NAME,BYTES/1024/1024 MB,MAX_BYTES/1024/1024 "MAX MB"
FROM DBA_TS_QUOTAS WHERE USERNAME='DOG';
```
注意:如果字节数为-1 则表示不限制用户对表空间的使用.在 oracle11 中在不赋给用户操作表空间的操作权限的时候可以建表.但是不能为表插入数据.

1.4 数据库模式
模式是一个命了名的对象的集合，如表、视图和序列号等。
当一个用户被创建时，一个与之相对应的模式也被创建。
一个用户只能与一个模式相关。
用户名与模式互换。

oracle 数据库中，模式对象包括:
表(tables)
视图(views)
索引(indexes)
约束(constraints)
序列号(sequences)
同义词(synonyms)
触发器(triggers)
存储过程、函数和软件包(stored programs,functions and packets)
用户定义的数据类型(user_defined data types)

1.5 改变用户在表空间上的配额
修改配额大小
```
ALTER USER DOG QUOTA 20M ON USERS;
```
收回用户在表空间上的全部空间
```
ALTER USER DOG QUOTA 0 ON USERS;
```

1.6 删除用户
```
DROP USER DOG CASCADE;
```

1.7 用户的安全控制域
当创建了一个用户的同时，也定义了一个用户的安全控制域(security domain)。
安全控制域除了前面曾经介绍过的安全检测机制(authentication mechanism)、用户的默认表空间(default tablespace)、用户排序所用的临时表空间(temporary tablespace)和表空间配额(tablespace quotas)之外，还包括如下的设置：
    账户上锁(account locking):可以通过对账户上锁来阻止用户登录数据库。上锁可以是自动的也可以是手动指定的。
    资源限制(resource limits): 加载资源使用上的限制，如ＣＰＵ时间，逻辑输入或输出和用户所能打开的会话等。
    直接权限(direct privileges): 控制用户对数据库能做的操作。
    角色权限(role privileges): 通过使用角色所间接授予的权限。

1.8 概要文件(profile)
概要文件是一组命了名的口令和资源限制，它是通过ＤＤＬ语句 CREATE USER 或 ALTER USER 赋予用户
概要文件可以被开启（激活）和关闭（禁止），而且概要文件也可以与默认的概要文件相关。

使用概要文件的好处是：
可以将用户按他们的安全控制和资源使用要求分成若干个组，然后为每一组按用户的需求创建一个概要文件，最后再将这些概要文件分别赋予相关的用户。这样可以大大地减轻数据库管理员的工作负担，也提高了工作效率，同时也减少了出错的机会。

oracle 服务器会自动创建一个名为ＤＥＦＡＵＬＴ的默认概要文件。概要文件的所有限制的初始值都是无限的，但是数据库管理员可以根据情况进行修改。
概要文件有如下特性：
    赋予用户的概要文件并不影响当前的会话。
    只能将概要文件赋予用户而不能将概要文件赋予角色或者其他的概要文件。
    如果在创建用户时没有赋予一个概要文件，默认的概要文件将赋予这个用户。

1.9 利用概要文件进行资源管理
利用概要文件来控制资源使用的具体步骤如下：
    1）利用概要ＣＲＥＡＴＥ　ＰＲＯＦＩＬＥ命令创建一个概要文件，在这个概要文件中定义资源和口令的限制。
    2）使用 CREATE USER 和 ALTER USER 命令将概要文件赋予用户。
    3）用以下方法之一来开启资源限制。
    在初始化参数文件中将 RESOURCE_LIMIT 设为 TRUE
    使用 ALTER SYSTEM 命令将 RESOURCE_LIMIT 设为 TRUE　　ALTER SYSTEM SET RESOURCE_LIMIT=TRUE
要想利用概要文件来控制资源的使用必须开启资源控制否则即使在概要文件中已经定义了资源限制也没有用。但是口令限制只要定义了就起作用。

1.10 资源限制的设置
概要文件的资源限制既可以加载会话一级，也可以加在调用一级。
在会话级设置的资源限制是强加在每一个连接上的。

在会话级可以设置的资源限制如下：
    SESSIONS_PER_USER:每个用户所允许的并行会话数。
    CPU_PER_SESSION:总共的ＣＰＵ时间，其单位是 1%.
    IDLE_TIME:没有活动的时间，其单位是分。只计算服务器进程。
    CONNECT_TIME:连接的时间，其单位是分。
    LOGICAL_READS_PER_SESSION:物理（磁盘）和逻辑（内存）读的数据块数。

在调用一级设置的资源限制是强加在每一个执行一条ＳＱＬ语句所做的调用之上的。当超过了调用级的资源限制时：

ｏｒａｃｌｅ系统挂起所处理的语句。
回滚这条语句。
所有之前的语句都完好无损。
用户的会话仍然保持链接状态。

在调用级可以设置的资源限制如下:
CPU_PER_CALL:每个调用所用的ＣＰＵ时间，其单位是1%ｓ。
LOGICAL_READS_PER_CALL:每个调用可以读的数据块数。

1.11 创建资源限制的概要文件
创建语句是:
CREATE PROFILE 概要文件名 LIMIT
[SESSION_PER_USER 最大值]
[CPU_PER_SESSION 最大值]
[CPU_PER_CALL 最大值]
[CONNECT_TIME 最大值]
[IDLE_TIME 最大值]
[LOGICAL_READS_PER_SESSION 最大值]
[LOGICAL_READS_PER_CALL 最大值]

SQL> CREATE PROFILE luck_prof LIMIT
SESSIONS_PER_USER 8 #使用这个概要文件的用户，利用同一个用户名和口令可以同时打开8个会话。
CPU_PER_SESSION 16800 #每个会话最多可以使用ｃｐｕ时间为16800个1%ｓ（168秒）。
LOGICAL_READS_PER_SESSION 23688 #每个会话最多可以读23688个数据块（包括内存读和磁盘读）。
CONNECT_TIME 268 #每个会话的连接时间最多为268分（4小时28分）。
IDLE_TIME 38; 每个会话的没有活动的时间不能超过38分。

1.12 查看已经创建的概要文件的内容
SQL> SELECT * FROM DBA_PROFILES WHERE PROFILE LIKE 'LUCK%';

另外每个ｏｒａｃｌｅ服务器中都有一个叫做ＤＥＦＡＵＬＴ的默认概要文件，如果一个用户没有被显式地赋予一个概要文件，ｏｒａｃｌｅ系统将自动吧ＤＥＦＡＵＬＴ概要文件赋予这个用户。不过ＤＥＦＡＵＬＴ概要文件中的一些设置可能无法满足一些商业数据库的实际需要，还可能留下安全隐患，这是数据库管理员应该进行修改。

1.13 口令管理
为了使ｏｒａｃｌｅ数据库更加安全，数据库管理员可以通过概要文件来控制ｏｒａｃｌｅ数据库的口令管理。

可以使用的一些口令管理的特性:
账户加锁(account locking): 当一个用户试了规定的次数仍不能登录数据库系统时，开始对这个账户自动上锁。
口令衰老和过期(password aging and expiration): 使口令具有生命周期，过了这段时间之后口令就作废了并且必须改变这一口令。
口令历史(password history): 检查新的口令以确保旧的口令在指定的时间内不会重用，或在指定的变化次数之内不会重用。
口令复杂性检验(password complexity verification): 对口令进行复杂性检验以保证口令足够复杂而不易被黑客通过猜口令来攻破口令防线。

口令管理和资源管理一样，都是通过建立概要文件并将它们赋予用户来进行口令管理。也是通过使用CREATE USER和ALTER USER命令来将概要文件赋予用户和对用户进行加锁、解锁和使（用户）帐号作废等操作的。要注意的是与资源管理不同，口令限制总是开启的。为了开启口令复杂性检验功能，要在ｓｙｓ用户下运行utlpwdmg.sql脚本文件。

完成口令管理要使用的参数:
口令账户加锁是通过以下两个参数来实现的。
FAILED_LOGIN_ATTEMPTS:在账户被锁住之前可以尝试登录失败的次数。
PASSWORD_LOCK_TIME:在尝试登录指定的次数失败后，账户将被锁住的天数。
口令衰老和过期是通过以下两个参数实现的。
PASSWORD_LIFE_TIME:口令的生命周期（可以使用的天数）。在此日志之后口令作废。
PASSWORD_GRACE_TIME: 口令过期后第一次成功的使用原口令登录后要改变口令的宽限天数。

口令历史是通过以下两个参数实现的。
PASSWORD_REUSE_TIME: 在一个口令可以重用之前的天数。
PASSWORD_REUSE_MAX:在一个口令可以重用之前的最大变化数。
注意: 以上两个参数的任何一个被设为某一值而不是ＤＥＦＡＵＬＴ或ＵＮＬＩＭＩＴＥＤ时，另一个参数必须设为ＵＮＬＩＭＩＴＥＤ。
口令复杂性检验是通过以下这个参数来实现的。
PASSWORD_VERIFY_FUNCTION:在一个新的口令赋予一个用户之前，要验证口令的复杂性是否满足安全要求的一个ＰＬ／ＳＱＬ函数。
CONNECT_TIME 每个连接最多可以连接多长时间，单位为分钟.
IDLE_TIME 闲置多长时间后被断开，单位为分钟.
CPU_PER_SESSION 每个会话占用ＣＰＵ的时间.
CPU_PER_CALL 每个调用占用ＣＰＵ的时间.

1.14 口令验证函数
口令复杂性检验函数名为 VERIFY_FUNCTION。该函数必须在ｓｙｓ用户中通过运行 utlpdmg.sql 脚本文件来生成。

导入 utlpwdmg.sql 脚本创建认证函数。
```
SQL> @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql
```
此脚本创建完成后生成一个认证函数 verify_function

修改ＤＥＦＡＵＬＴ默认的概要文件:
```
SQL> ALTER PROFILE DEFAULT LIMIT
  2  PASSWORD_LIFE_TIME 60
  3  PASSWORD_GRACE_TIME 10
  4  PASSWORD_REUSE_TIME 1800
  5  PASSWORD_REUSE_MAX UNLIMITED
  6  FAILED_LOGIN_ATTEMPTS 3   (着重注意，失败次数等于 3 次后账户被锁定. )
  7  PASSWORD_LOCK_TIME 1/1440
  8  PASSWORD_VERIFY_FUNCTION verify_function;
```

当加入认证函数后，ｏｒａｃｌｅ服务器就要对用户提供的口令进行如下检查:
口令的最小长度为 4 个字符。
口令不应该与用户名相同。
口令应该包含至少一个字符、一个数字和一个特殊字符。
口令应该至少有 3 个字母与以前的口令不同。

加入认证函数后再给用户制定口令非常麻烦，如果取消认证函数:
```
SQL>  ALTER PROFILE DEFAULT LIMIT
  2  PASSWORD_VERIFY_FUNCTION NULL;

Profile altered.
```

系统管理员自己可以写一个口令函数，每当将一个新的口令函数加入到ｏｒａｃｌｅ数据库系统时，系统管理员必须考虑如下的一些限制:
函数必须使用上面所介绍的说明（接口）
函数的返回值ＴＲＵＥ为成功，ＦＡＬＳＥ为失败。
如果口令函数产生异常，系统将返回出错信息，并且相应的 CREATE USER 或 ALTER USER 语句被终止。
如果口令函数变为无效，系统也将返回出错信息，并且相应的 CREATE USER 或 ALTER USER 语句也会被终止。
ｓｙｓ用户拥有口令函数。

1.15 创建口令限制的概要文件
创建口令概要文件的语法格式为
```
CREATE PROFILE 概要文件名 LIMIT
[FAILED_LOGIN_ATTEMPTS 最大值]
[PASSWORD_LIFE_TIME 最大值]
[{PASSWORD_REUSE_TIME | PASSWORD_REUSE_MAX} 最大值]
[PASSWORD_LOCK_TIME 最大值]
[PASSWORD_GRACE_TIME 最大值]
[PASSWORD_VERIFY_FUNCTION { 函数名 | NULL | DEFAULT }]
```

其中，在设置口令参数小于 1 天时，
    1 小时，PASSWORD_LOCK_TIME=1/24
    1 分钟，PASSWORD_LOCK_TIME=1/1440

示例:
```
SQL> CREATE PROFILE UNLUCK_PROF LIMIT
  2  FAILED_LOGIN_ATTEMPTS 7
  3  PASSWORD_LOCK_TIME UNLIMITED
  4  PASSWORD_LIFE_TIME 44
  5  PASSWORD_REUSE_TIME 24
  6  PASSWORD_GRACE_TIME 4;
```

1.16 修改和删除概要文件
修改概要文件
```
SQL> ALTER PROFILE UNLUCK_PROF LIMIT
    FAILED_LOGIN_ATTEMPTS 4
    PASSWORD_LIFE_TIME 74
    PASSWORD_GRACE_TIME 14;
```

查看概要文件
```
SQL> SELECT * FROM DBA_PROFILES WHERE PROFILE LIKE 'UNLUCK%' AND
RESOURCE_TYPE='PASSWORD';
```

删除概要文件
```
SQL> DROP PROFILE LUCK_PROF;
```

把概要文件分给用户
```
SQL> ALTER USER BFOA PROFILE UNLUCK_PROF;
```

2.权限管理
2.1 常用的系统权限

2.1.1 有关用户的系统权限
CREATE USER:创建其他的用户（需要具有 D B A 角色的权限）
ALTER USER:修改其他用户的设置。
DROP USER:删除其他的用户。

2.1.2 有关表的系统权限
SELECT ANY TABLE:查询任何用户的表中的数据和视图中的数据的能力。
UPDATE ANY TABLE:修改任何用户的表中的数据和视图中的数据的能力。
DELETE ANY TABLE:删除任何用户的表中的数据和视图中的数据的能力。
CREATE ANY TABLE:在任何模式中创建表。
DROP ANY TABLE:删除任何模式中创建的表。
ALTER ANY TABLE:修改任何模式中创建的表。
CREATE TABLE:在用户自己的模式中创建表。

2.1.3 有关表空间的系统权限
CREATE TABLESPACE:创建表空间的权限。
DROP TABLESPACE:删除表空间的权限。
ALTER TABLESPACE:修改表空间的权限。
UNLIMITED TABLESPACE:使用全部表空间的权限。

2.1.4 有关索引的系统权限
CREATE ANY INDEX:在任何模式中创建索引的权限。
DROP ANY INDEX:在任何模式中删除索引的权限。
ALTER ANY INDEX:在任何模式中修改索引的权限。

2.1.5 有关会话的系统权限
CREATE SESSION:连接数据库的权限。
ALTER SESSION:发 ALTER SESSION 语句的权限。

2.1.6 其他的系统权限
CREATE VIEW:在用户自己的模式中创建视图的权限。
CREATE SEQUENCE:在用户自己的模式中创建序列号的权限。
CREATE PROCEDURE:在用户自己的模式中创建过程的权限。

2.1.7 DML 对表中行或列的授权
查询　　行
插入列
更新列
删除行

2.2 特殊的系统权限

2.2.1SYSOPER 系统权限
执行 STARTUP 和 SHUTDOWN 操作
ALTER DATABASE OPEN | MOUNT | BACKUP
ARCHIVELOG 和 RECOVERY
CREATE SPFILE
包括 RESTRICTED SESSION 权限。

2.2.2SYSDBA 系统权限
SYSOPER 权限 WITH ADMIN OPTION
CREATE DATABASE
ALTER TABLESPACE BEGIN/END BACKUP
RECOVER DATABASE UNTIL

2.3 系统权限的限制
从 oracle 8 开始引入一个名为 O7_DICTIONARY_ACCESSIBILITY 的参数，oracle 系统利用这一参数来控制 SELECT AND TABLE 权限访问系统的方式。
如果这个参数被设为真（TRUE）就表示具有 SELECT AND TABLE 权限的用户可以查询数据字典，即允许访问 SYS 模式中的对象。
如果这个参数被设为假（FALSE）就表示具有 SELECT AND TABLE 权限的用户不可以查询数据字典，即确保具有访问任何模式权限的用户不能访问 SYS 模式中的对象。
SQL> SHOW PARAMETER DICTIONARY

```
NAME                    TYPE   VALUE
------------------------------------ ----------- ------------------------------
O7_DICTIONARY_ACCESSIBILITY      boolean   FALSE
```

2.4 授予系统权限

为 TEST1 用户授予指定权限。
SQL> GRANT CREATE SESSION,SELECT ANY TABLE,CREATE TABLE,CREATE VIEW TO TEST1;

为 TEST1 用户授予指定权限，并且 TEST1 继承了可以分配这些权限的能力。
SQL> GRANT CREATE SESSION,SELECT ANY TABLE,CREATE TABLE,CREATE VIEW TO TEST1
WITH ADMIN OPTION;

查看 TEST1 用户具有那些权限
SQL> SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE='TEST1';

```
GRANTEE               PRIVILEGE            ADM
------------------------------ ---------------------------------------- ---
TEST1                 CREATE VIEW              YES
TEST1                 SELECT ANY TABLE         YES
TEST1                 CREATE TABLE             YES
TEST1                 CREATE SESSION               YES
```

SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE IN ('TEST1');制定多个用户时可以用英文半角逗号隔开。

2.5 回收系统权限

取消ＴＥＳＴ１用户指定权限。
SQL> REVOKE CREATE SESSION,SELECT ANY TABLE,CREATE TABLE,CREATE VIEW FROM TEST1;
如果授权的时候加上了 WITH ADMIN OPTION 选项，则取消授权的时候应该反向取消。

2.6 对象权限
对象权限是维护数据库中的对象的权利（能力）。
oracle 系统中一共有 8 种对象权限，它们分别是:
EXECUTE
ALTER
SELECT
INSERT
UPDATE
DELETE
INDEX
REFERENCES

对象与对象权限对应关系:

对象权限	procedure	sequence	view	table
EXECUTE	Y			
ALTER		Y	$	Y
SELECT		Y	Y	Y
INSERT			Y	Y
UPDATE			Y	Y
DELETE			Y	Y
INDEX			$	Y
REFERENCES				Y

视图以$号表示的两种权限，即 ALTER 和 INDEX 权限。

2.7 对象权限的授权和回收
对象的主人（拥有者）自动具有该对象的一切权限。可以把一些对象的权限授予其他用户。
如果具有 GRANT AND OBJECT PRIVILEGE 系统权限，就可以像对象拥有者一样将对象的权限赋予（收回从）其他用户。

语法格式:
GRANT 对象的权限 |ALL [（列名[, 列名...]）]
ON 对象名
TO [用户名|角色名|PUBLIC]
[WITH GRANT OPTION]
其中:
对象的权限: 要授予的对象的权限。
ＡＬＬ: 所有对象的权限。
列名: 该列上的对象的权限（要授予其他用户的）。
ＯＮ对象名: 该对象上的对象全乡将授予其他用户。
ＴＯ [用户名|角色名]: 指明对象权限要授予谁。
ＰＵＢＬＩＣ: 指明对象权限要授予系统的所有用户。
WITH GRANT OPTION:允许被授权的用户再将这些对象权限授予其他用户。

将 DBMS_SPACE_ADMIN 软件包的执行权限赋予 system 用户
SQL> GRANT EXECUTE ON DBMS_SPACE_ADMIN TO SYSTEM;

查看赋予的对象权限。
```
SQL> SELECT * FROM DBA_TAB_PRIVS WHERE PRIVILEGE LIKE 'EX%'
  2  AND TABLE_NAME LIKE 'DBMS_SPACE_ADMIN%';

GRANTEE               OWNE TABLE_NAME             GRAN PRIVILEGE    GRA HIE
----------------------------- ---- ------------------------------ ----
------------ --- ---
SYSTEM                SYS  DBMS_SPACE_ADMIN          SYS EXECUTE      NO  NO
```

将scott.emp表的ＳＥＬＥＣＴ权限授予所有用户
```
SQL> GRANT SELECT ON EMP TO PUBLIC;
SQL> GRANT UPDATE(SAL) ON EMP TO TEST1;
SQL> GRANT UPDATE(JOB) ON EMP TO TEST1 WITH GRANT OPTION;
SQL> GRANT UPDATE(SAL) ON EMP TO TEST1,TEST2;
```

查看授权情况
```
SQL> SELECT * FROM USER_TAB_PRIVS_MADE;
```

取消授权
```
SQL> REVOKE UPDATE ON EMP FROM TEST1;
```

3.角色管理
在用户量非常打的时候如果为每个用户去分配权限的话非常麻烦，这时就用到了角色这个概念，角色里可以预定义一些权限。它相当于操作系统中的权限组.
角色是：一组命名的相关权限，这组权限可以通过这个名字授予用户或其他的角色。
一个角色不能授予自己，也不能循环授予。
每个角色在系统中是唯一的，即不能与任何现有的用户名和角色名重名。
角色不属于任何用户，也不存在与任何用户模式中。角色的描述存放在数据字典中。

系统预定义角色
CONNECT: 包含权限CREATE SESSION
RESOURCE:包含权限CREATE TABLE,CREATE PROCEDURE,CREATE SEQUENCE,CREATE TRIGGER,CREATE TYPE,CREATE CLUSTER,CREATE INDEXTYPE,CREAT OPERATOR
DBA: 绝大多数系统权限.

使用角色的好处:
- 比较容易的权限管理
- 动态的权限管理
- 可以提高系统的效率
- 可以通过操作系统授权
- 可以有选择的使用权限

3.1角色的创建
创建一个不需要口令标识的角色ＣＬＥＲＫ
```
SQL> CREATE ROLE CLERK;
```

创建一个需要口令标识的角色ＳＡＬＥＳ
```
SQL> CREATE ROLE SALES IDENTIFIED BY MONEY;
```

创建一个需要使用外部标识（如操作系统）的角色manager
```
SQL> CREATE ROLE MANAGER IDENTIFIED EXTERNALLY;
```

查看创建的角色和口令信息
```
SQL> SELECT * FROM DBA_ROLES WHERE ROLE IN ('CLERK','SALES','MANAGER');

ROLE                  PASSWORD
------------------------------ --------
CLERK                 NO
SALES                 YES
MANAGER               EXTERNAL
```

3.2 角色的修改
修改角色ＣＬＥＲＫ的验证方法为外部（如操作系统）标识
```
SQL> ALTER ROLE CLERK IDENTIFIED EXTERNALLY;
```

取消角色ＳＡＬＥＳ的认证标识
```
SQL> ALTER ROLE SALES NOT IDENTIFIED;
```

角色的认证标识改为ＶＡＭＰＩＲＥ
```
SQL> ALTER ROLE MANAGER IDENTIFIED BY VAMPIRE;
```

3.3 角色的授权
将 CREATE SESSION,CREATE TABLE,CREATE VIEW 权限赋予角色ＣＬＥＲＫ
```
SQL> GRANT CREATE SESSION,CREATE TABLE,CREATE VIEW TO CLERK;
```

把 SELECT AND TABLE 系统权限和 CLERK 据色一起赋予角色ＭＡＮＡＧＥＲ
```
SQL> GRANT SELECT ANY TABLE,CLERK TO MANAGER;
```

查看据色分配情况
```
SQL> SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE IN ('CLERK','SALES','MANAGER');

ROLE                  PRIVILEGE    ADM
------------------------------ ------------ ---
CLERK                 CREATE TABLE NO
CLERK                 CREATE VIEW  NO
MANAGER               SELECT ANY T NO
                  ABLE

CLERK                 CREATE SESSI NO
                  ON
```

查看角色ＣＬＥＲＫ到底授予了哪些用户和角色的信息。
```
SQL> SELECT * FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE LIKE 'CL%';

GRANTEE               GRANTED_ROLE            ADM DEF
------------------------------ ------------------------------ --- ---
MANAGER               CLERK                NO  YES
SYS               CLERK               YES YES
```

带有 WITH ADMIN OPTION 选项的授权语句将ｍａｎａｇｅｒ角色赋予ＴＥＳＴ１用户
```
SQL> GRANT MANAGER TO TEST1 WITH ADMIN OPTION;
```

查询用户ＴＥＳＴ１所授予的角色信息
```
SQL> CONN TEST1/TEST1
Connected.
SQL> SELECT * FROM USER_ROLE_PRIVS;

USERNAME              GRANTED_ROLE            ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
TEST1                 MANAGER                 YES YES NO
```

查看用户具有的所有权限
```
SQL>  SELECT * FROM SESSION_PRIVS;

PRIVILEGE
------------------------------
CREATE SESSION
CREATE TABLE
SELECT ANY TABLE
CREATE VIEW
```

3.4 建立默认角色
将授予ＴＥＳＴ１用户的所有角色都设为非默认角色，即当用户ＴＥＳＴ１登录系统时以角色赋予的任何系统权限都是不能使用的。
SQL> ALTER USER TEST1 DEFAULT ROLE NONE;

将授予ＴＥＳＴ１用户所有的角色重新设置为默认角色，即当用户ＴＥＳＴ１登录系统时所有的角色赋予的系统权限都有效。
SQL> ALTER USER TEST1 DEFAULT ROLE ALL;

将授予ＴＥＳＴ１用户所有的据色重新设为默认角色但ｓａｌｅｓ据色除外，即当用户 TEST1 登录系统时除了 sales 角色之外的所有角色赋予的系统权限都有效。
SQL> ALTER USER TEST1 DEFAULT ROLE ALL EXCEPT sales;

3.5 激活和禁止角色
连接到指定用户中，在用户环境下临时禁止角色
SQL> CONN TEST1/TEST1;
Connected.
SQL> SET ROLE NONE;

Role set.

激活角色，由于创建角色的时候使用了口令标识，所以激活角色也要用到标识。
SQL> SET ROLE MANAGER IDENTIFIED BY SALES;

3.6 角色的回收和删除
回收指定用户的权限和角色
SQL> REVOKE MANAGER,SELECT ANY TABLE FROM TEST1;

删除角色
SQL> DROP ROLE MANAGER;

选择激活角色
SQL> SET ROLE ALL EXCEPT MANAGER;

3.7 创建和使用角色指南
无

3.8oracle 预定义的角色
oracle 系统中比较常见的预定义角色如下:
EXP_FULL_DATABASE:导出数据库的权限
IMP_FULL_DATABASE:导入数据库的权限
SELECT_CATALOG_ROLE:查询数据字典的权限
EXECUTE_CATALOG_ROLE:数据字典上的执行权限
DELETE_CATALOG_ROLE:数据字典的删除权限
DBA,CONNECT,RESOURCE:这 3 个角色为了不同版本兼容而设置的。

****
注意: 如果给一个用户分配了 RESOURCE 角色后，oralce 自动给这个用户授予了 UNLIMITED TABLESPACE 权限. 这样存在系统安全隐患.
角色可以动态调整.
如果一个用户同时具有两个角色，如果其中一个角色激活后另外的角色就被关闭了. 这个用户就不能使用角色中的权利.
****

查询预定义角色的权限
SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE IN ('CONNECT','RESOURCE');