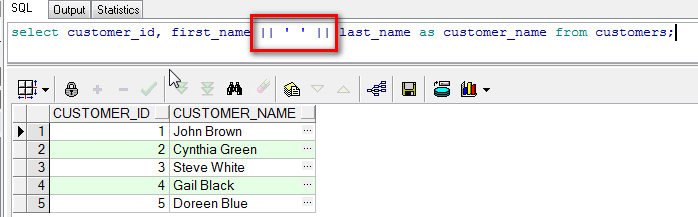
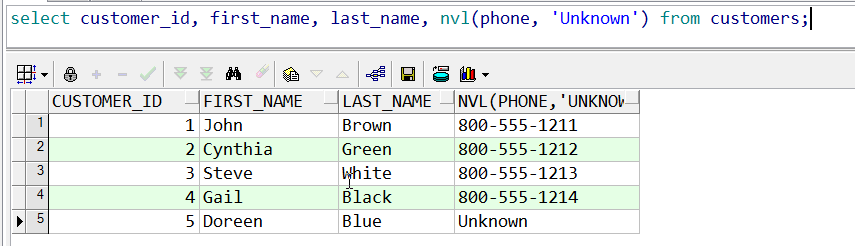
## Ch2 从数据库表中检索信息

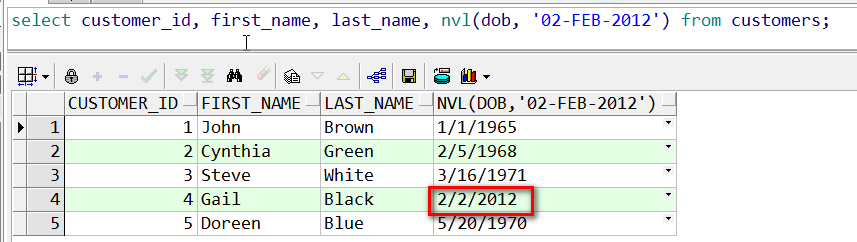
1. 用 || 来合并列的输出结果



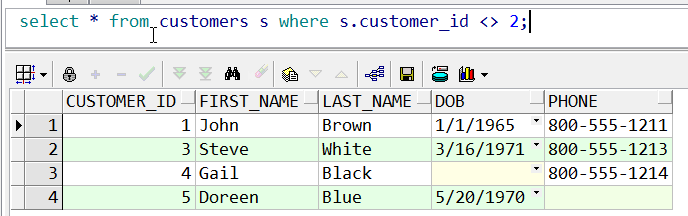
1. NVL()



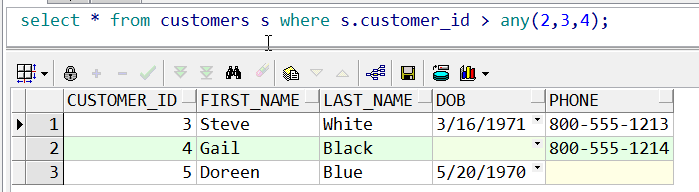
NVL除了可以转换字符串外还可以转换日期



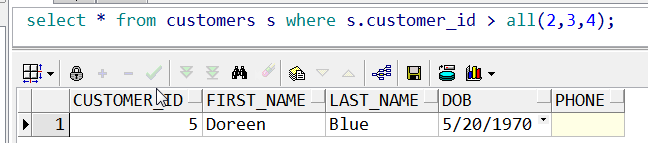
1. 操作运算符之不等于号 <> !=



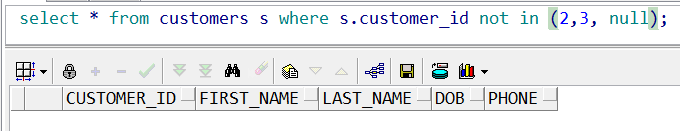
1. 操作运算符之ANY(大于任意一个就成立)



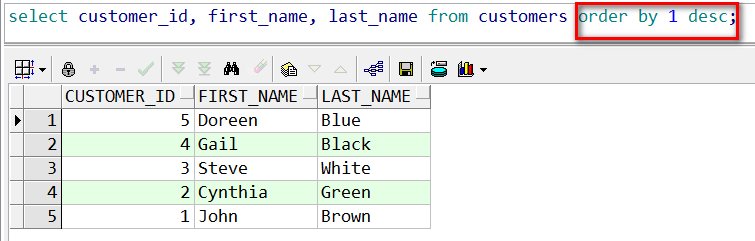
1. 操作运算符之ALL(必须大于所有的才成立)



1. 如果指定的值列表中有一个为null，那么not in就会返回false，什么都查不到



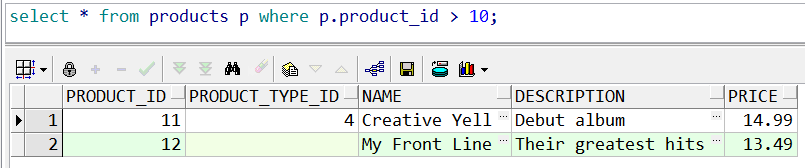
1. Order by 也可以根据列的次序指定对哪 一列进行排序：1表示按第1列排序



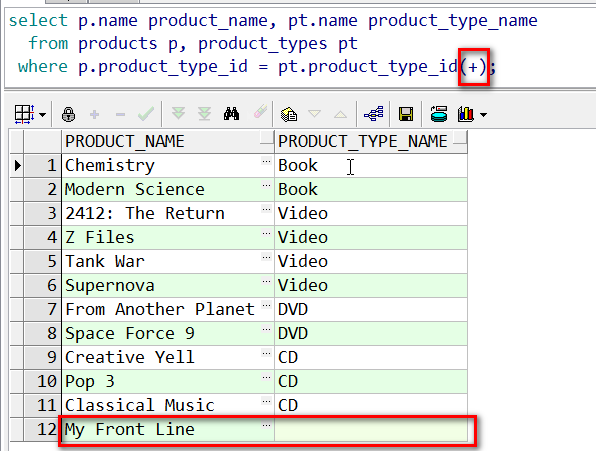
1. 理解外连接

所谓外连接就是指即使连接中的列包含一个空值，外连接也会返回一行。在连接条件中可以使用外连接操作符来执行一个外连接；外连接操作符是使用圆括号括起来的加号：(+)

先来看products表中包含有一个product\_type为null的列my front line：



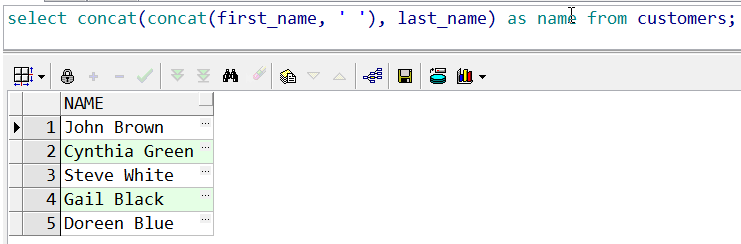
再看下面的检查结果：



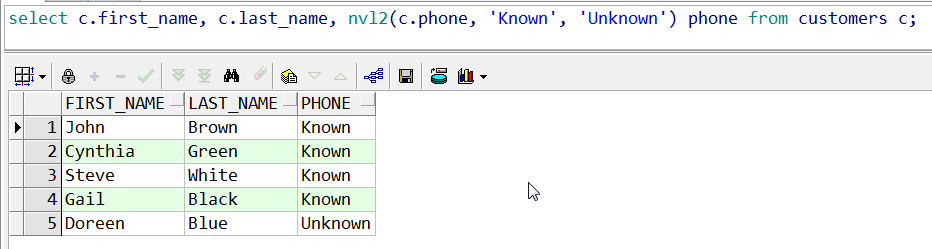
注意外连接操作符(+)位于product表product\_type\_id列（包含空值的列）相反的一边

## Ch3 使用简单函数

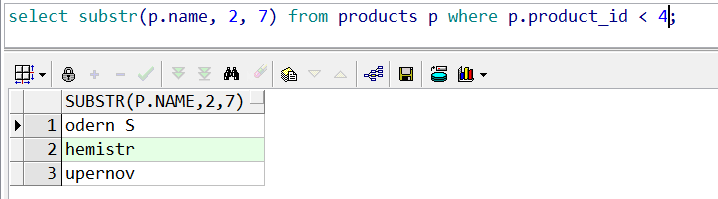
1. Concat(a, b)将b字符串连到a后面



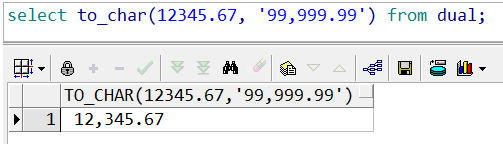
1. NVL2(x, value1, value2)如果X非空，则返回value1，否则返回value2



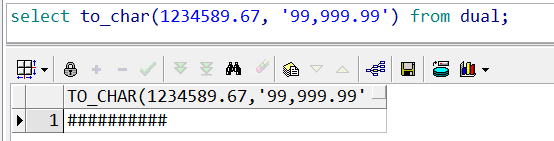
1. Substr



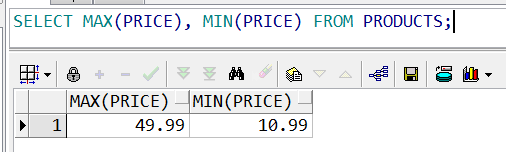
1. to\_char(value, pattern)根据pattern来格式化value，并将结果作为字符串返回



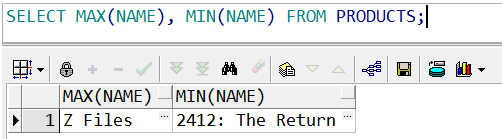
当要处理的数包含的数字个数多于格式中指定的数的个数，那么试图进行这种转换成，就会有如下的结果：



1. Count(column)，，使用此函数时，应该尽量避免使用星号（\*）,因为这样count()返回结果时所需要的时间会更长。
2. MAX()和MIN()函数



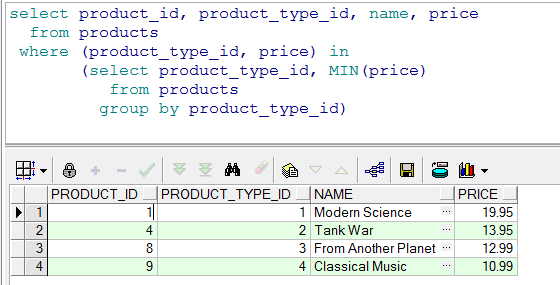
MAX和MIN函数可以用于任何类型，包括字符串和日期。在对字符串使用MAX时，字符串按照字符串的顺序进行排序，字符串的最大值们于列表的底部，最小值位于顶部。



## Ch6 子查询

1. 多列子查询

查询每种产品类型中价格最低的产品



1. EXISTS和IN的比较

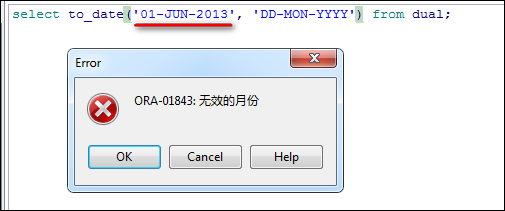
EXISTS和IN不同：EXISTS只是检查行的存在性，而IN则要检查实际值的存在性。通常来讲，EXISTS的性能要比IN高，因此应该尽可能的使用EXIST而不是IN。

在编写使用NOT EXIST和NOT IN的查询时必须谨慎。当一个值列表包含一个空值时NOT EXIST就返回true,而NOT IN则返回false。考虑下面的例子，本例使用了NOT EXISTS，查过那些在products表中没有任何产品的产品类型

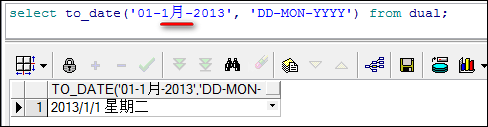
## Ch7 高级查询

to\_date(dateStr, pattern):

使用to\_date函数时，我们必须要注意第一个参数的值必须符合当前oracle的格式，如在中文系统下，如果想要进行如下的转换，则会报错：



正确的做法是：



[What is the difference between UNION and UNION ALL](http://stackoverflow.com/questions/49925/what-is-the-difference-between-union-and-union-all)

UNION removes duplicate records (**where all columns in the results are the same**), UNION ALL does not.

There is a performance hit when using UNION vs UNION ALL, since the database server must do additional work to remove the duplicate rows, but usually you do not want the duplicates (especially when developing reports).

The implication of this, is that union is much less performant as it must scan the result for duplicates

In ORACLE: UNION does not support BLOB (or CLOB) column types, UNION ALL does.

You should avoid of unnecessary UNIONs they are huge performance leak. As a rule of thumb use UNION ALL if you are not sure which to use.

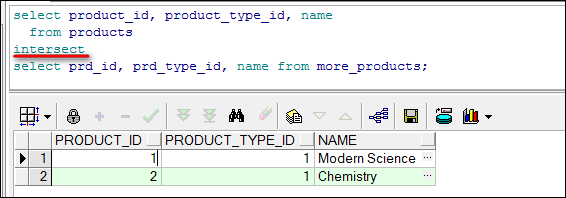
You can avoid duplicates and still run much faster than UNION DISTINCT (which is actually same as UNION) by running query like this:

SELECT \* FROM mytable WHERE **a=X** UNION ALL SELECT \* FROM mytable WHERE b=Y AND **a!=X**

Notice the AND a!=X part. This is much faster than UNION.

intersect（相交，交集）

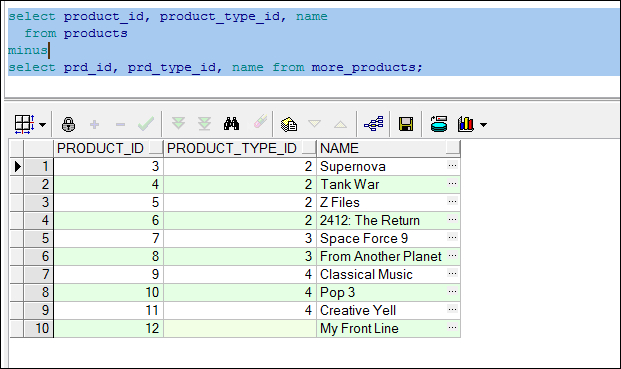
该操作符返回两个查询所检索出的共有行。下面这个例子使用了intersect，注意只检索出了那些products和more\_products共有的行



minus(减去，负数)

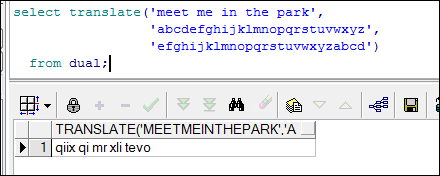
返回第二个查询检索出的行从第一个查询检索出的行中减去之后剩余的行。

下面的例子为从products中返回的行中先减去从more\_products中返回的行，然后再返回剩余的行。（即是排除同时在两个表中都出现的行）



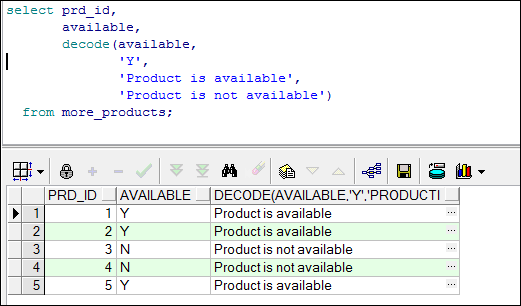
translate(x, from\_string, to\_string)

在x字符串中查找from\_string中的字符（是字符，不是字符串，即不需要x中的某一部分完全匹配from\_string才转换，看下面的例子），并将其转换成to\_string中对应的字符

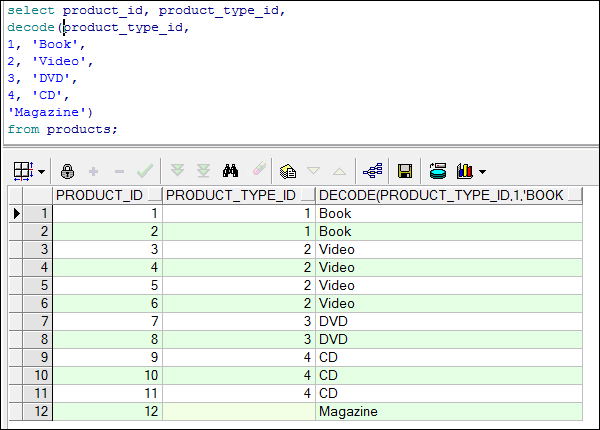


Decode(value, search\_value, result, default\_result)

对Value与Search\_value比较，如果两个值相等，则返回result，否则返回default\_result。Decode函数提供了if-then-else的功能而不需要使用PL/SQL



该函数也支持传递多个搜索和结果参数，如下例所示



题外话：从上面的函数的支持可以分析出，解析这个方法的时候，应当是除了第一和最后一个参数之外的参数看作是多组pairs，从而可以支持这个多参数的类型了

Case表达式

Case与decode类似，提供了if-then-else而不需要使用PL/SQL的方式，但相比之下，我们应当使用case，因为它与ANSI兼容

简单case表达式：

Case search\_expression

When expression1 then result1

When expression2 then result2

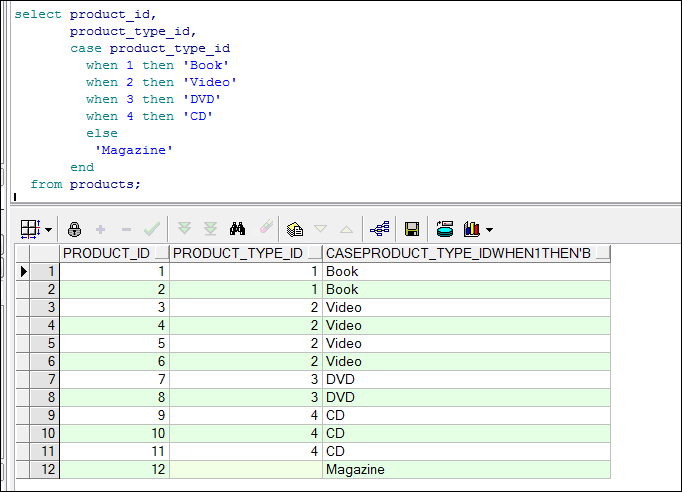
…..

When expressionN then resultN

Else

Default\_value

End



使用搜索case表达式

Case

When condition1 then result1

When condition 2 then result2

…..

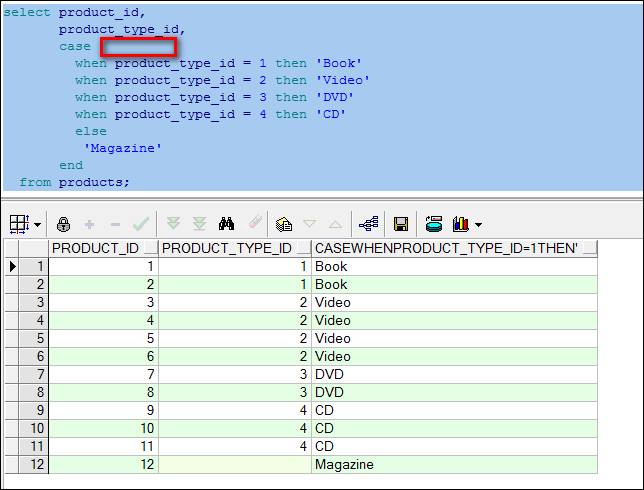
When condition N then resultN

Else

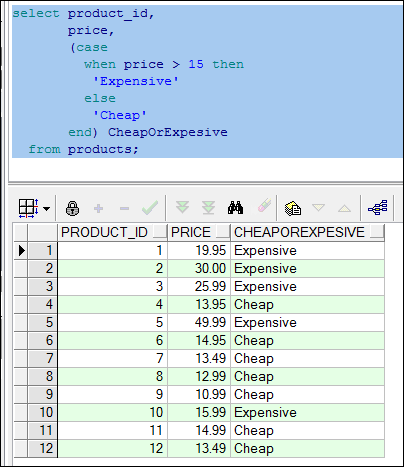
Default\_value

End

（注意这种形式下的case表达式不需要在case反而跟着search\_expression）



当然也支持其它类型的操作符，并且也支持对case表达式返回的结果重命名：



## CH8 修改表的内容

可以在SQL中定义一个变量：

Variable average\_product\_price number

保存点：

Savepoint save1;

然后就可以调用下面的语句来回滚到这个保存点：

Rollback to savepoint save1;

而直接使用rollback表示回滚到事务开始的位置处

事务的ACID特性：

* 原子性(atomicity): 一个事务中包括的所有SQL语句是一个不可分割的单元
* 一致性(consistency)： 事务必须确保数据库的状态保持一致，这就是说事务开始时，数据库的状态是一致的，在事务结束后，数据库的状态也是一致的
* 隔离性(isolation)： 多个事务可以独立运行，而不会彼此产生影响
* 持久性(durability)： 一旦事务被提交之后，数据库的变化就会被永远保留下来，即使运行数据库软件的机器后来崩溃也是如此

并发事务：

Oracle数据库支持多个用户同时与数据库进行交互，每个用户都可以同时运行自己的事务，这种事务就称为并发事务(concurrent transaction)

在默认的情况下，如果两个用户同时进行查询操作，a用户后来执行了一个insert语句，但并没有commit，b用户再进行一次查询操作，那么b看到的结果和一开始时看到的结果是一致的，而只有当b执行commit命令后，执行的查询操作才会看到更新后的结果，这就是oracle默认的隔离级别，而这个隔离级别是可以修改的。

事务锁：

要支持并发事务，oracle必须确保表中的数据一直有效。这可以通过锁(lock)来实现。考虑下面的例子：两个事务t1和t2修改customers表中的顾客1的记录：

* T1执行一条update语句修改顾客1的记录，但t1并没有执行commit语句。此时就称为t1对该行“加锁”了
* T2也试图执行一条update谗来修改顾客1的记录，但是由于该行已经被t1加锁了，因此t2现在就不能获得该行的锁，t2的update语句就必须一直等待，直到t1结束并释放该行的锁为止
* t2获得该行上的锁，并执行update语句。t2获得该行上的锁后一直持有，直到t2结束为止

以上的过程可以归纳为：当一个事务已经拥有某一行上的锁的时候，另外一个事务不能获得该行上的锁。

注意：理解默认的加锁机制的最简单的方法如下：读程序不会阻塞读程序；写程序不会阻塞读程序；只有在试图对相同的行进行修改的时候，写程序才会阻塞写程序

事务的隔离级别：

首先要明白事务处理中可能存在的三种问题：

* 幻像读取（phantom read）事务t1读取一条指定where子句所返回的结果集，然后事务t2新插入一条记录，这行记录恰好可以满足t1所使用的查询中的where子句的条件。然后t1又使用相同的查询再次对表进行检索，但是此时却看到了事务t2新插入的新行，这个新行就称为“幻像“。因为对于t1来说这一行就像是突然出现的一样
* 不可重复读取(nonrepeatable read): 事务t1读取一行记录，紧接着事务t2修改了t1刚才读取的那一行记录的内容。然后t1又再次读取了这一行的记录，发现它与刚才读取到的数据不一致了。这种现象称为“不可重复“读，因为t1原来读取的记录发生了变化
* 脏读（dirty read）事务t1更新了一行记录的内容，但是并没有提交所做的修改，事务t2读取更新后的行。然后t1执行回滚操作，取消了刚才所做的修改。现在t2所读取到的行也就是无效的（也称为脏数据）了，因为在t2读取这行记录的时候，t1所做的修改并没有提交

为了处理这些问题，数据库实现了不同级别的事务隔离性，以防止并发事务会相互影响。SQL标准定义了以下几种事务隔离级别，按照隔离性级别从低到高依次为：

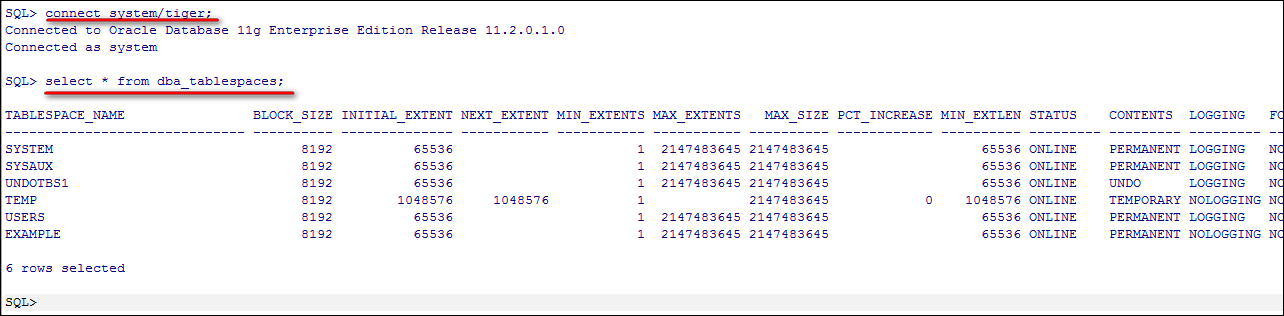
* read uncommitted: 幻像读，不可重复读和脏读都允许
* read committed: 允许幻像读和不可重复读，但是不允许脏读
* repeatable read: 允许幻像读，但是不允许不可重复读和脏读
* serializable: 幻像读，不可重复读和脏读都不允许

Oracle数据库支持read committed和serializable两种事务隔离级别，不支持另外的两种。

虽然SQL标准所定义的默认隔离级别是serializable，但oracle数据库默认使用的是read committed， 这几乎对于所有的应用程序来说是可以接受的。

## CH9 数据库安全性

查询数据库里有多少表空间的方法：



修改用户密码：

