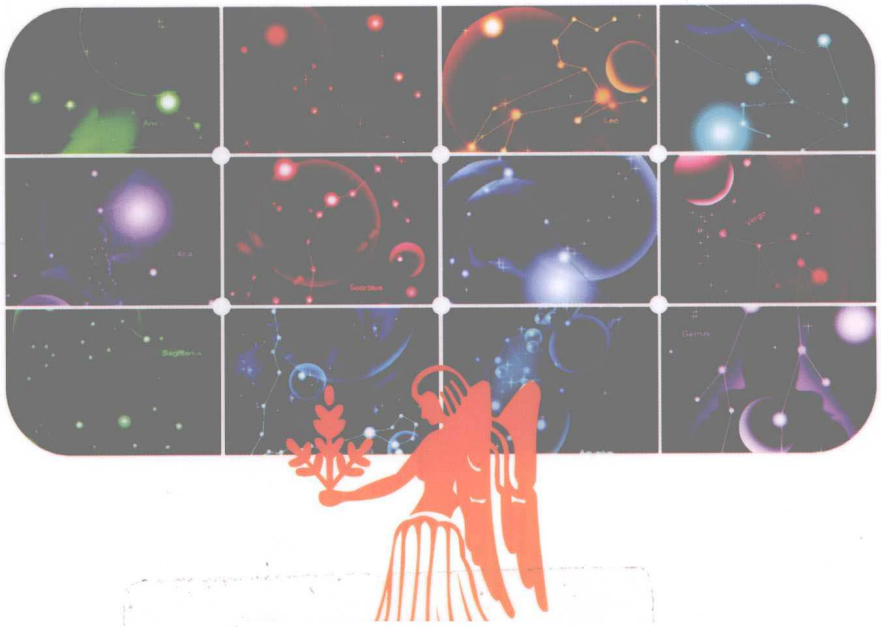
HZ BOOKS
华章科技

数据库技术丛书

多年管理和维护经验总结　案例实用方便读者快速上手



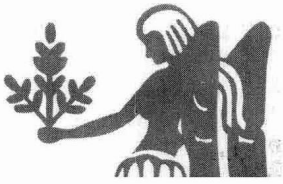
Oracle 11g
数据库管理员指南

刘宪军◎编著

机械工业出版社
China Machine Press

数据库技术丛书

Oracle 11g
数据库管理员指南



刘宪军◎编著

机械工业出版社
China Machine Press

Oracle 11g是Oracle公司最新推出的数据库版本。本书从实用的角度出发，系统地介绍了Oracle 11g的使用和管理，并对它的体系结构和常规管理进行了重点描述。本书对深奥的理论知识不作过多的讨论，重点突出实用性，在每章中都提供了许多实用的例子，力求帮助读者更好地使用Oracle。

从内容组织形式上来看，本书分为四大部分。第一部分介绍了Oracle 11g的使用基础，包括SQL语言基础、SQL*Plus的使用和PL/SQL编程，其中对Oracle特有的PL/SQL进行了比较深入的描述。第二部分是本书的重点，介绍了Oracle数据库管理（DBA）的各个方面，其中对Oracle的体系结构进行了重点介绍。第三部分介绍了Oracle的自动文件管理和自动存储管理。第四部分介绍了数据库的备份与恢复，其中对RMAN进行了重点介绍。

本书不仅可以作为Oracle数据库技术人员的参考手册，还可以作为培训中心的培训教材。

图书在版编目（CIP）数据

Oracle 11g数据库管理员指南 / 刘宪军编著. —北京：机械工业出版社，2010.6
（数据库技术丛书）

ISBN 978-7-111-30935-2

Ⅰ.O… Ⅱ.刘… Ⅲ.关系数据库-数据库管理系统，Oracle 11g Ⅳ.TP311.138

中国版本图书馆CIP数据核字（2010）第106170号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：陈佳媛
北京市荣盛彩色印刷有限公司印刷
2010年8月第1版第1次印刷
186mm × 240mm · 23.75印张
标准书号：ISBN 978-7-111-30935-2
定价：49.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

前　言

目前全世界发展势头最猛的IT巨头可能要数Oracle公司了。几年前，Oracle收购了BEA公司，成功地将Weblogic这个优秀的应用服务器软件收归己有。不久前，Oracle又将SUN公司纳入自己旗下，从此有了自己的小型机和UNIX操作系统，这无疑为Oracle插上了腾飞的双翼。然而，Oracle公司最引人注目的还是它的数据库产品。

在15年前，大家可能想不到Oracle公司会有今天的发展成果，当时的数据库市场是Sybase的天下，Oracle只是一个名不见经传的小公司。如今Oracle数据库产品以其出色的安全性、稳定性和优异的性能稳居数据库产品的榜首，占领了数据库市场的大部分份额，在银行、保险、通信、政府等应用领域具有绝对的话语权。

在一个企业应用系统中，核心部分是数据。对一个企业而言，数据就是生命。要管理重要的数据库，就需要高水平的数据库管理员。无论在国内还是国外，数据库管理员的待遇都是很丰厚的。然而Oracle软件毕竟是一个非常复杂的数据库产品，如果只掌握它的皮毛，你只能做一些初级的技术工作，根本无法管理数据库，你就失去了升职、深造的机会。不仅如此，如果不小心把数据搞丢了，还要承担法律责任。

想必大家都听过“庖丁解牛”的故事吧？庖丁在解牛的时候，手所接触的地方，肩所靠的地方，脚所踩的地方，膝盖所顶的地方，都发出皮骨相离声，刀子刺进去时响声更大，这些声音没有不合乎音律的。正当你在欣赏这种音律时，牛肉已经被干净利索地解了下来，而庖丁“提刀而立，为之四顾，为之踌躇满志，善刀而藏之”。究其原因，只有“依乎天理”，才能做到“以神遇而不以目视，官知止而神欲行”，从而“批大郤，道大窾，因其固然”。一般的厨工每月换一把刀，是因为他们用刀子去砍骨头，而庖丁的刀已经用了19年，还像新的一样。

学习Oracle也是一样的，只有掌握Oracle的脉络，那么在解决实际问题的时候才能像庖丁那样，做到得心应手，游刃有余。在客户现场，当几个工程师（可能包括原厂工程师）正在为解决一个问题争得面红耳赤的时候，如果你三下五除二帮客户解决了问题，你也能体会到庖丁那种“踌躇满志”的感觉。

本书的编写目的，就是试图使读者掌握Oracle的脉络，而不是教大家怎样学SQL语句。本书既不是对英文资料的翻译，也不是对Oracle产品用法的直白描述，而是对作者多年从事技术支持和培训工作的经验总结。书中列举了很多实际的例子，都是作者在客户现场遇到过并亲手解决的实际案例，希望这些例子对大家有所帮助。

读者在学习Oracle时，可以先学习怎样在UNIX/Linux系统中安装Oracle软件，然后学习怎样创建数据库，接着应该学习SQL和PL/SQL。接下来要重点学习的内容是Oracle的体系结构，只有掌握了这部分内容，才有可能对数据库进行管理、备份与恢复以及性能优化。以后如果有机会配置RAC集群环境，或者利用DataGuard配置数据库的异地容灾，读者就能体会到，一旦

掌握了Oracle体系结构，对自己的帮助是多么大。

现在的企业应用系统都越来越复杂，涉及好多种技术。要想管理好数据库，只掌握少数的两三种技术是不够的。就数据库而言，目前的运行环境一般都是UNIX/Linux，管理员应该至少掌握一种UNIX，还应该了解Java、网络和一些存储方面的知识。

在本书编写之前不久，Oracle公司发布了Oracle 11g的最新版本的产品11.2。根据以往的情况来看，Oracle每个版本的产品都有两个次版本，其中第二个次版本是最稳定、用户使用时间最长的产品。如Oracle 10g有两个次版本10.1和10.2，目前很多企业用户还在使用10.2这个版本。本书的内容力求体现Oracle的最新特点。

面对技术发展的迅猛势头，作者也感觉才疏学浅。本书难免有疏漏和不足的地方，敬请读者朋友批评指正。

作　者

2010年5月

目　录

第二部分 Oracle DBA

第四部分 备份与恢复

第一部分 Oracle使用基础

第1章 SQL语言基础

SQL是结构化查询语言（Structured Query Language）的缩写，它是目前关系数据库系统中通用的标准语言。

SQL最早在20世纪70年代由Boyce和Chamberlin提出，并首先在IBM公司的数据库管理系统System R上实现，随后又在IBM的DB2上实现，并获得了巨大的成功。后来美国标准化组织和国际标准化组织先后将SQL作为关系数据库系统的标准语言，从此，SQL得到了发展的机会。到目前为止，包括Oracle、Sybase、Informix等在内的几乎所有大型数据库系统都支持SQL。

SQL在字面上虽然称为结构化查询语言，实际上它还包括数据操纵、数据定义、事务控制、安全控制等一系列命令。SQL操作的基本对象是表，也就是关系。它可以对表中的数据进行查询、增加、删除、修改等常规操作，还可以维护表中数据的一致性、完整性和安全性，能够满足从单机到分布式系统的各种应用需求。

SQL是一种非过程化的语言，用户在使用SQL操作数据时，只需要告诉系统做什么，而不需要关心怎么做，系统会根据用户的意图自动完成相应的操作。由于SQL的这一特点，它被人们称为“第四代语言”（4GL），以区别于面向过程的高级语言。

用SQL语言编写的SQL语句有两种执行方式，一种是联机交互方式，SQL语句在一定的平台上执行，例如数据库管理系统提供的实用程序。这个执行平台将SQL语句提交给数据库服务器，并将从数据库服务器返回的执行结果显示给用户。另一种方式是嵌入方式，用户在用C/C++、Java等高级语言编写应用程序时，可能需要操作数据库中的数据，这时SQL作为一种嵌入式语言，嵌入到高级语言程序中，通过数据库接口如ODBC、JDBC访问数据库中的数据。

SQL包括一系列命令，可以满足对数据的各种访问。按照通用的分类标准，SQL命令分为以下几种类型：

- 查询命令　包括SELECT命令
- DML命令　包括INSERT、DELETE、UPDATE命令
- DDL命令　包括CREATE、DROP、ALTER、RENAME、TRUNCATE命令
- 事务控制命令　包括COMMIT、ROLLBACK、SAVEPOINT命令
- DCL命令　包括GRANT、REVOKE命令

命令和相关的参数一起构成了SQL语句。下面将对这些命令分别进行详细的介绍。

1.1　查询语句

查询语句是使用最为频繁的数据库访问语句，对应的SQL命令是SELECT。虽然只有一条命令，但是由于它有灵活多样的形式，以及功能强大的子句，可以组成各种复杂的查询语句，能够完成各种复杂的查询。

SELECT语句可以根据用户的要求查询数据库中的数据，并且可以对它们进行简单的计算和统计。最简单的SELECT语句只有一个FROM子句，格式如下：

```
SELECT表达式
FROM表名
```

其中SELECT之后引导一个或多个列名，或者表达式，用来指定需要查询的列，或者对数据所进行的计算。在FROM子句指定一个或多个表名，用来指定本次查询所涉及的表。查询的结果是返回一行或多行数据，每行由一个或多个列的列值组成。

完整的SELECT语句包括WHERE、ORDER、GROUP等子句。格式如下：

```
SELECT 表达式
FROM 表名
WHERE 条件
GROUP BY 列名
HAVING 条件
ORDER BY 表达式
```

SELECT语句最灵活的用法体现在WHERE子句中的查询条件，这个条件用来指定查询什么样的数据。在以下各节中，我们将分别介绍SELECT语句的各个组成部分。

1.1.1　查询语句的基本用法

如果要查询某个表中一个或多个列的数据，需要在SELECT命令之后指定列名，并在FROM子句中指定查询所涉及的表。格式如下：

```
SELECT 列1，列2...
FROM 表名
```

查询的结果是从指定的表中将指定列的数据显示出来。例如，要查询dept表中的deptno和loc列，对应的SELECT语句为：

```
SELECT deptno,loc FROM dept;
```

这样的语句可以在Oracle提供的实用工具SQL*Plus中执行，也可以在其他实用工具或应用程序中执行。SQL*Plus的提示符是“SQL>”，在本书中介绍的SQL语句基本上是在SQL*Plus中执行的。在操作系统的终端窗口中输入sqlplus命令，并指定用户名和口令，即可登录数据库服务器。例如（其中$为UNIX/Linux系统的SHELL提示符）：

```
$ sqlplus scott/tiger
```

SQL语句中除字符串外，各个部分是大小写不敏感的。如果在SQL*Plus中执行SQL语句，还要在语句末尾加上一个分号。分号并不是SQL语句的一部分，只是语句结束的标志。一条句可以在一行中书写，也可以分行书写。关于SQL*Plus的详细用法，请参阅第2章。这条命令的

执行结果为：

```
DEPTNO  LOC
    10  NEW YORK
    20  DALLAS
    30  CHICAGO
    40  BOSTON
```

如果要查询表中的所有的列，可以用“*”符号代替所有的列名。例如：

```
SQL> SELECT * FROM dept;
```

如果不了解表的结构，可以在SQL*Plus中执行命令DESCRIBE（简写为DESC），查看表的结构。这个命令的参数是表名，或者其他对象名。注意，这条命令不是SQL命令，而是SQL*Plus中的命令。

为了演示SQL语句的用法，在本章中大部分SQL语句都以Oracle提供的模板模式中的表emp和dept为操作对象。只要以用户名scott和口令TIGER登录数据库服务器，就能直接访问这两个表。其中emp表是员工的信息表，dept是部门信息表，这两个表的结构如下所示（其中各列的意义是作者标注的）：

```
SQL> DESC emp
名称                    是否为空        类型            意义
EMPNO                   NOT NULL        NUMBER(4)       员工编号
ENAME                   NULL            VARCHAR2(10)    姓名
JOB                     NULL            VARCHAR2(9)     职务
MGR                     NULL            NUMBER(4)       经理编号
HIREDATE                NULL            DATE            受聘时间
SAL                     NULL            NUMBER(7,2)     工资
COMM                    NULL            NUMBER(7,2)     奖金
DEPTNO                  NULL            NUMBER(2)       所在的部门号
SQL> DESC dept
名称                    是否为空        类型            意义
DEPTNO                  NOT NULL        NUMBER(2)       部门编号
DNAME                   NULL            VARCHAR2(14)    部门名称
LOC                     NULL            VARCHAR2(13)    部门所在的城市
```

在默认情况下，在显示数据时，各列的标题就是列的名称。在SELECT语句中可以定义列的别名，这样在显示数据时，列的标题就是这个别名，在整个SQL语句中都可以使用这个别名。使用别名的SELECT语句格式为：

```
SQL> SELECT 列1 AS 别名1, 列2 AS 别名2...
```

或者在列名后直接指定别名，省略AS关键字。例如：

```
SQL> SELECT deptno AS 部门编号,loc 地址 FROM dept;
```

这条命令的执行结果为：

```
部门编号 地址
      10  NEW YORK
      20  DALLAS
```

```
30  CHICAGO
40  BOSTON
```

在查询结果中如果有重复行，可以使用DISTINCT关键字去掉重复行的显示。重复行是指在SELECT语句中涉及的所有列的列值完全相同的行。例如，要查询员工所分布的部门，可以用DISTINCT关键字去掉其中的重复行：

```
SQL> SELECT DISTINCT deptno AS 部门编号 FROM emp;
```

实际上多名员工在同一部门上班的情况是存在的，但是这条命令执行的结果去掉了重复的部门编号的显示。命令执行的结果为：

```
部门编号
    10
    20
    30
```

SELECT语句不仅可以进行简单的查询，还可以对查询的列进行简单的计算，也可以在两个列之间进行计算，或者将某个列与其他表达式，或者两个表达式进行计算。表1.1中列出了在SELECT语句可以使用的运算符。

表1.1　查询语句中可以使用的运算符

运算符	意义
-	取相反数
* /	乘法和除法
+ -	加法、减法
||	字符串连接

使用||运算符可以将两个数据连接起来。无论是数字型还是日期型数据，在进行这种运算时，都可以看做是字符型数据。通过||运算符，用户可以设计自己喜欢的数据显示方式，如将两个列的值连接起来，也可以将列的值与其他文字连接起来。连接以后所得的数据可以当做一个列来显示。例如，可以将dept表中的deptno和loc列以及其他文字连接起来，相应的SELECT语句为：

```
SQL> SELECT '部门'||deptno||'的地址为：'||loc AS 部门地址 FROM dept;
```

这条命令的执行结果为：

```
部门地址
----------------------------------------
部门10的地址为：NEW YORK
......
```

如果在SQL语句中使用了字符串，必须用一对单引号将字符串限定，并且字符串中的字符是大小写敏感的。

加减乘除四则运算在SELECT语句中比较简单，需要注意的是空值的计算。空值与其他数据进行四则运算时，结果将得到空值，而不管它与什么样的数据运算。例如，要在emp表中查询员工的工资与奖金之和，由于部分员工的奖金为空，致使查询的结果与我们希望的结果不符。查询语句为：

```
SQL> SELECT sal+comm AS 总收入 FROM emp;
```

这条语句的执行结果为：

```
  总收入
----------
```

```
    1900
    1750
    ......
已选择12行。
```

在emp表中共有12名员工，每名员工都有工资。如果奖金为空，对应的计算结果就为空。空值与0或者空格是不同的。空值就是没有数据，而0或者空格是实实在在的数据，就像考试没有成绩和得了0分是不一样的。为了解决空值的计算问题，SQL提供了一个函数，这个函数是NVL，它的功能是把空值转换为其他可以参加运算的数据。这个函数的调用格式是：

```
NVL(表达式，替代值)
```

当表达式的结果为空时，这个函数就把表达式的值用指定的值代替。有了这个函数，我们就可以在奖金为空时把它用0或者其他数据代替。改进后的查询工资和奖金之和的语句为：

```
SQL> SELECT sal+nvl(comm,0) AS总收入 FROM emp;
总收入
----------
    800
   1900
   1750
   2975
......
已选择12行。
```

SELECT命令还可以用来计算一个普通表达式的值，这个表达式可能与表没有任何关系，如3*5这样的表达式。例如：

```
SQL> SELECT 3*5, 3+5 FROM dept;
```

不过这样的查询语句所得的结果却不是我们希望的。这条SELECT语句执行的结果为：

```
3*5        3+5
---------- ----------
 15          8
 15          8
 15          8
 15          8
```

原来，查询的结果是把表达式的值重复了若干次。因为SELECT语句必须通过FROM子句指定一个或多个表，而表达式与这些表是无关的，所以，SELECT语句简单地根据可以查询到的行数，将表达式的值重复若干次。为了解决这个问题，Oracle提供了一个特殊的表dual，这个表的结构为：

```
SQL> DESC dual
 名称                      是否为空? 类型
 ------------------------- -------- ----------------------------
 DUMMY                              VARCHAR2(1)
```

通过查询这个表，发现表中只有一行数据：

```
SQL> SELECT * FROM dual;
```

```
D
-
X
```

可见，dual表只有一个列，而且中有一行数据。所以，在进行与具体的表无关的运算时，可以在FROM子句中指定dual表，这样可以保证计算的结果只显示一次。例如：

```
SELECT 3*5, 3+5 FROM dual;
```

1.1.2　查询语句中的条件

在前面所列举的查询中，由于没有限制条件，所以查询的结果是将表中的所有行都显示出来。如果希望只查询一部分行，那么可以通过WHERE子句指定条件。WHERE子句的作用是通过指定条件，使SELECT语句仅仅查询符合条件的行，如部门10的员工数据，或者工资大于2000元的员工数据等。在更多情况下，都需要根据指定的条件对数据进行查询。

WHERE子句指定的条件是一个关系表达式，如果关系表达式的结果为真，则条件成立，否则条件不成立。关系表达式用于比较两个表达式的大小，或者进行模糊匹配，或者将一个表达式的值与一个集合中的元素进行匹配。表1.2列出了常用的关系运算符。

表1.2　常用的关系运算符

关系运算符	用　法	说　明
= != > >= < <=		比较两个表达式的大小及是否相等
LIKE	LIKE '字符串'	字符串的模糊匹配
IN	IN (元素1, 元素2...)	与集合中的元素进行匹配
BETWEEN	BETWEEN a AND b	检查表达式的值是否在a和b之间
AND OR	条件1 AND 条件2	两个条件的连接
	条件1 OR 条件2	
NOT	NOT 条件	条件取反
IS NULL		判断表达式的值是否为空

例如，要在表dept中查询部门10的员工姓名和工资信息，对应的SELECT语句为：

```
SQL> SELECT ename,sal FROM emp WHERE deptno=10;
```

下面的SELECT语句用于查询员工KING的基本情况：

```
SQL> SELECT empno,ename,sal,comm FROM emp WHERE ename='KING';
```

LIKE运算符通常用来进行字符串的模糊匹配，而“=”运算符只能对字符串进行精确比较。在LIKE指定的关系表达式中可以使用两个通配符：%和_，其中%可以代替多个字符，_可以代替一个字符。例如，要查询包含字符串“AR”的员工姓名，构造的SELECT语句为：

```
SQL> SELECT ename FROM emp WHERE ename LIKE '%AR%';
```

又如要查询这样的员工，姓名中第一个字符是任意字符，第二个是“A”，然后是若干任意字符，这时构造的SELECT语句为：

```
SQL> SELECT ename FROM emp WHERE ename LIKE '_A%';
```

注意，%用来代替多个连续的字符，包括空字符串，而_只能用来代替一个字符，不包括空

字符。

IN运算符用来与一个集合中的元素进行比较。SELECT语句将指定的表达式与集合中的元素一一比较，只要与其中一个相等，则条件成立。如果没有任何一个元素与表达式的值相等，则条件不成立。例如，下面的SELECT语句用于查询其姓名在指定集合之中的员工：

```
SQL> SELECT ename FROM emp WHERE ename IN ('SMITH','FORD','HELLO');
```

BETWEEN运算符用于将表达式的值与两个指定数据进行比较，如果表达式的值在这两个数据之间，则条件成立。这两个数据和表达式必须能够比较大小，而且后一个数据必须大于前一个数据。例如，下面的SELECT语句用于查询工资在1000到2000之间的员工：

```
SQL> SELECT ename FROM emp WHERE sal BETWEEN 1000 AND 2000;
```

如果用包含“>”等运算符的表达式改写上述SQL语句，则对应的SELECT语句为：

```
SQL> SELECT ename FROM emp
      WHERE sal>=1000 AND sal<=2000;
```

在复杂的查询语句中，可能需要多个条件，这些条件通过AND或OR运算符连接。多个条件表达式连接起来以后，就构成一个逻辑表达式。逻辑表达式的结果要么为真，要么为假，它是与两个关系表达式的值和所使用的连接运算有关的。假设X和Y是两个关系表达式，表1.3列出了两个关系表达式的运算规则。

表1.3 关系表达式的运算规则

X	Y	X AND Y	X OR Y
真	真	真	真
真	假	假	真
假	真	假	真
假	假	假	假

例如，要查询在部门10工作，且工资在1000和2000之间的员工姓名，相应的SELECT语句为：

```
SQL> SELECT ename FROM emp
     WHERE deptno=10 AND sal BETWEEN 1000 AND 2000;
```

NOT运算符的作用是对关系表达式的值取反。它的用法是在关系表达式之前加上NOT运算符。例如，要查询工资不大于1000的员工姓名，相应的SELECT语句为：

```
SQL> SELECT ename FROM emp WHERE NOT sal<1000;
```

这条语句等价于：

```
SQL> SELECT ename FROM emp WHERE sal>=1000;
```

在默认情况下，NOT运算符只对最近的一个关系表达式取反，如果要对已经通过AND或OR连接的多个关系表达式同时取反，则要用一对圆括号将多个关系表达式限定。例如，要对下列SELECT语句中的两个条件同时取反：

```
SELECT ename FROM emp WHERE sal>1000 AND sal<2000;
```

对两个条件同时取反以后的SELECT语句为：

```
SELECT ename FROM emp WHERE NOT (sal>1000 AND sal<2000);
```

这条语句等价于：

```
SELECT ename FROM emp WHERE sal<=1000 OR sal>=2000;
```