第一部分　数据库体系结构

第1章　Oracle体系结构

每一种新版本的Oracle，都要增加一些新的性能或对原有性能进行某些改善。在 Oracle8i中，除了增加许多新的性能外，还改进了原版本的许多功能。同时 Oracle增加了许多新的工具以简化数据库管理任务。在本书的第一部分，你不仅能看到对 Oracle体系结构及其实现的综合介绍，而且还将看到创建Oracle数据库所需要采取的步骤。

本书的第二部分介绍管理Oracle数据库的具体方针，例如回滚段的管理及口令的建立。第三部分描述Oracle在网络环境中的使用方法。最后一部分介绍最常用的 SQL命令的语法，并含有一个不同Oracle版本初始化参数变化的导引。

这一部分提供Oracle体系结构的基本框架和创建数据库所要遵循的步骤。第 1章向你介绍Oracle数据库的组件和实现其用途的基本原理。管理 Oracle数据库需要了解这些不同组件之间的相互关系、它们在框架中的位置以及如何以最佳方式定制系统以满足用户需要。总之，本章宛如其他章节深入讨论数据库管理的一个向导。

1.1　数据库概述及实例

要了解Oracle体系结构，必须先了解两个基本概念：数据库和实例。下面两节将详细描述这两个基本概念及其在Oracle中的实现。

1.2　数据库

数据库(database)是一个数据集合。 Oracle能够提供按照一致性方式定义的定义模型 (称作关系模型)存储和访问数据的方法，因此 Oracle被认为是一种关系数据库管理系统 (RDBMS)。对“数据库”一词的大多数引用不仅是指物理的数据，也指本章中描述的物理、内存及进程对象的组合。

数据库中的数据存储在表中。关系表由列 (column)定义，并赋予一个列名。数据在表中以行(row)的方式存储。表可以相互关联，数据库可用来实施这些关联。表结构的一个样例如图1-1所示。

除了按关系格式存储数据外， Oracle(例如Oracle 8)支持面向对象(OO)的结构(如抽象数据类型和方法)。对象既可以与其他对象建立关系，也能包含其他对象。如第 5章中所述，可以用对象视图来启动面向对象的数据接口，而不必对表做任何修改。

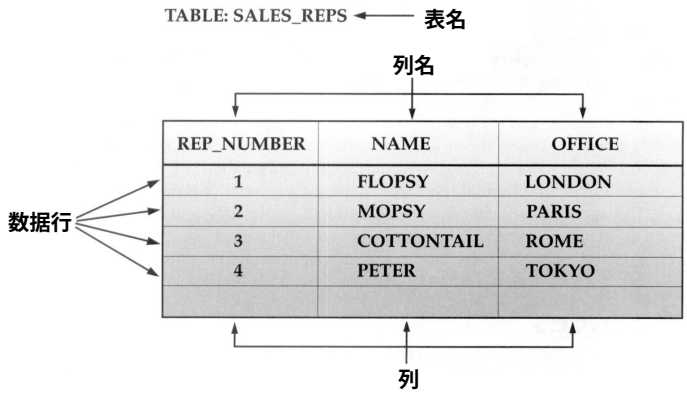

图1-1 表结构样例

无论是采用关系结构还是面向对象的结构， Oracle数据库都将其数据存储在文件中。在其内部，数据库结构提供数据对文件的逻辑映射，允许不同类型的数据分开存储。这些逻辑划分被称作表空间。下面两小节介绍表空间和文件。

1.2.1 表空间

表空间(tablespace)是数据库的逻辑划分，每个数据库至少有一个表空间 (称作SYSTEM表空间)。为便于管理和提高运行效率，可以使用一些附加表空间来划分用户和应用程序。例如：USER表空间供一般用户使用， RBS表空间供回滚段使用(有关这方面的情况在本节后面描述)。一个表空间只能属于一个数据库。

1.2.2 文件

每个表空间由同一磁盘上的一个或多个文件组成，这些文件叫数据文件 (datafile)。一个数据文件只能属于一个表空间。在 Oracle 7.2中，数据文件创建后可改变大小。创建新的表空间需要创建新的数据文件。

数据文件一旦加入到表空间中，就不能从这个表空间中移走，也不能与其他表空间发生联系。

如果数据库对象存储在多个表空间中，那么可以通过把它们各自的数据文件存放在不同磁盘上来对其进行物理分割。在规划和协调数据库处理 I/O请求的方法中，数据分割是一种很重要的工具。数据库、表空间和数据文件之间的关系如图 1-2所示。

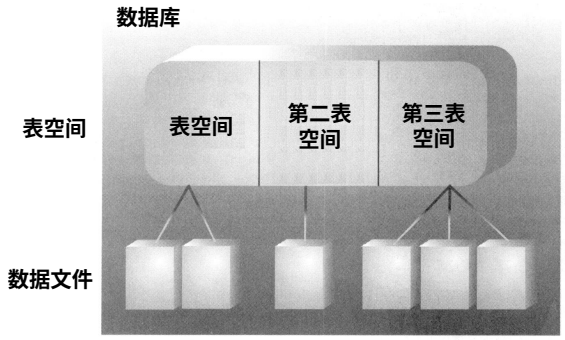

图1-2 数据库、表空间和数据文件之间的关系

1.3 实例

为了访问数据库中的数据，Oracle使用一组所有用户共享的后台进程。此外，还有一些存储结构(统称为System Global Area，即SGA)用来存储最近从数据库查询的数据。数据块缓存区和SQL共享池(Shared SQL Pool)是SGA中的最大部分，一般占分配给SGA的内存95%以上。通过减少对数据文件的I/O次数，这些存储区域可以改善数据库性能。

数据库实例(instance)也称作服务器(server)，是用来访问数据库文件集的存储结构及后台进程的集合。一个数据库可以被多个实例访问(这是Oracle的并行服务器选项)。实例与数据库之间的关系如图1-3所示。

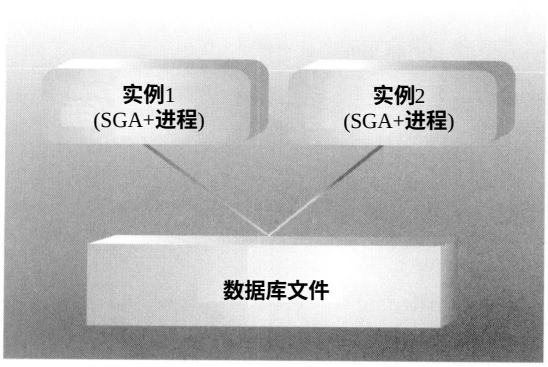

图1-3 Oracle中的实例和数据文件

决定实例的大小及组成的参数存储在init.ora文件中。实例启动时需要读这个文件，并且在运行时可以由数据库管理员修改。对该文件的任何修改都只有在下一次启动时才起作用。实例的init.ora文件名通常包含该实例的名字；如果一个实例名为ORCL，则init.ora文件通常将被命名为initorcl.ora。另一个配置文件config.ora存放一些在数据库创建后就不再改变的变量值(如数据库的块大小)。实例的config.ora文件名通常也包含该实例的名字；如果实例名字为ORCL，则config.ora文件一般将命名为configorcl.ora。为便于使用configorcl.ora设置值，在实例的init.ora文件中，该文件必须通过IFILE参数作为包含文件列出。

1.4 数据库内部结构

通过上面对数据库及实例的介绍，Oracle数据库结构可分为三个范畴：

- 数据库内部的结构(如表)。
- 存储区内部的结构(包括共享存储区和进程)。
- 数据库外部的结构。

以下几小节将对每一范畴内的每个元素进行介绍。这些范畴按上述顺序给出。

下一节将对数据库内部结构的元素进行介绍，这些元素包括：

- 表、列、约束条件、数据类型（含抽象数据类型）。
- 分区与子分区。
- 用户与模式。
- 索引、簇和散列簇。
- 视图。

- 序列。
- 过程、函数、软件包和触发器。
- 同义词。
- 权限及角色。
- 数据库链接。
- 段、盘区和块。
- 回滚段。
- 快照与显形图。

下面对上述每一种元素进行详细介绍。

1.4.1 表、列和数据类型

表是数据在一个Oracle数据库中的存储机制，如图1-1所示，它包含一组固定的列。表中的列描述该表所跟踪的实体的属性，每个列都有一个名字及各自的特性。

列由两部分组成：数据类型(datatype)和长度(length)。对于使用NUMBER数据类型的列，可以指定列的小数位及精度特性，精度决定数值的有效位数，小数位表示数值的小数点位置。说明为NUMBER(9,2)的列表示该列总共有9位数，其中2位数在小数点右边。缺省的数值精度为38位数，这也是Oracle所允许的最大数值精度。

表1-1列出了可用的数据类型。

表1-1 Oracle数据类型

数据类型	描 述
CHAR	固定长度字符域，最大长度可达2 000个字节
NCHAR	多字节字符集的固定长度字符域，长度随字符集而定，最多为2 000个字符或2 000个字节
VARCHAR2	可变长度字符域，最大长度可达4 000个字符
NVARCHAR2	多字节字符集的可变长度字符域，长度随字符集而定，最多为4 000个字符或4 000个字节
DATE	用于存储全部日期的固定长度(7个字节)字符域，时间作为日期的一部分存储其中。除非通过设置init. ora文件的NLS_DATE_FORMAT参数来取代日期格式，否则查询时，日期以DD-MON-YY格式表示，如13-APR-99表示1999.4.13
NUMBER	可变长度数值列，允许值为0、正数和负数。NUMBER值通常以4个字节或更少的字节存储
LONG	可变长度字符域，最大长度可到2GB
RAW	表示二进制数据的可变长度字符域，最长为2 000个字节
LONG RAW	表示二进制数据的可变长度字符域，最长为2GB
MLSLABEL	只用于Trusted Oracle，这个数据类型每行使用2至5个字节
BLOB	二进制大对象，最大长度为4GB
CLOB	字符大对象，最大长度为4GB
NCLOB	多字节字符集的CLOB数据类型，最大长度为4GB
BFILE	外部二进制文件，大小由操作系统决定
ROWID	表示RowID的二进制数据，Oracle 8 RowID的数值为10个字节，在Oracle7中使用的限定RowID格式为6个字节
UROWID	用于数据寻址的二进制数据，最大长度为4 000个字节

除了表1-1所列的数据类型外，Oracle还提供可作为ANSI标准数据类型的替代类型。对于ANSI数据类型CHARACTER及CHAR，使用Oracle的CHAR数据类型；对于ANSI数据类型CHARACTER VARYING及CHAR VARYING，使用Oracle的VARCHAR2数据类型；对于

ANSI NUMERIC、DECIMAL、DEC、INTEGER、INT和SMALLINT数据类型，使用Oracle的NUMBER数据类型；对于ANSI标准的FLOAT、REAL和DOUBLE PRECISION数据类型，Oracle支持PL/SQL中的一个FLOAT数据类型。

从Oracle 8起可以创建用户自己的抽象数据类型。也可以使用特定的REF数据类型，这些REF数据类型引用数据库其他地方的行对象(参见第5章)。

用户SYS和SYSTEM所拥有的表被称为数据字典表(data dictionary table)，数据字典表提供一个数据库用来管理自己的系统目录。数据字典由Oracle提供的一组目录脚本文件创建。每当安装或升级一个数据库时，都需要使用创建或修改数据字典表的的脚本文件。当在数据库中安装一个新的选项时，可能要运行另外一些附加的目录脚本文件。

表通过其共用的列相互关联。可以要求数据库通过引用完整性(referential integrity)实施这些关联。如果使用Oracle面向对象的特征，就可以通过称作对象ID(OID)的内部引用使行相互关联。引用完整性通过约束条件在数据库级实施。

1.4.2 约束条件

可以在一个表的列上创建约束条件(constraint)，当一个约束条件应用于一个表时，表中的每一行都必须满足约束条件定义所规定的条件。在下面的create table命令中，表EMPLOYEE就是由多个约束条件创建的。

```
create table EMPLOYEE
(EmpNo              NUMBER(10)     PRIMARY KEY,
 Name               VARCHAR2(40)   NOT NULL,
 DeptNo             NUMBER(2)      DEFAULT 10,
 Salary             NUMBER(7,2)    CHECK (salary<1000000),
 Birth_Date         DATE,
 Soc_Sec_Num        CHAR(9)        UNIQUE,
 foreign key (DeptNo) references DEPT(DeptNo))
tablespace USERS;
```

首先，表被赋予了表名(EMPLOYEE)，表中的每个列都有各自的列名(EmpNo、Name等)，每个列都有指定的数据类型及长度。EmpNO列被定义为NUMBER数据类型，没有小数位(等同于一个整数)。Name列被定义为VARCHAR2(40)，表示Name是一个可变长度列，其最大长度可以达到40个字符。

表中的主键(Primary Key)是表中的一个列或多个列，它们决定表中每个行的唯一性，数据库中的主键列必须是非空值(NOT NULL)，表示存储在表中的每一行必须有这个列的一个值；它不能为空值(NULL)。非空值约束条件可用于表中的其他列(如上例中的Name列)。

一个列可以有一个DEFAULT(缺省)约束条件，这种约束条件使得在往表中插入一个行但没有为列指定值时生成一个值。

CHECK(检查)约束条件确保指定列中的值符合一定的条件(如本例中，Salary列的值小于1000000)。CHECK列约束条件不能引用一个独立表。非空值约束条件被数据库看作是一个CHECK约束条件。

另一个约束条件是UNIQUE(唯一性)，用于保证应具有唯一性而又不是主键的一部分的那些列的唯一性。在这个例子中，Soc_Sec_Num列拥有UNIQUE约束条件，所以表中的每个记录都必须有该列的唯一值。

外键(foreign key)约束条件规定表间的关系性质。一个表的外键引用数据库中此前定义的主键。

例如，如果表DEPT有主键DeptNo，则DEPT中的记录将列出全部有效的DeptNo值。上述程序段中所示的EMPLOYEE表中的DeptNo列，引用了DEPT.DeptNo列。通过指定EMPLOYEE.DeptNo作为DEPT.DeptNO的一个外键，可以保证只有DEPT表中已存在DeptNo值时才能输入到EMPLOYEE表中。

数据库的约束条件有助于确保数据的引用完整性。引用完整性保证数据库中的所有引用都有效且全部约束条件都得到满足。

1.4.3 抽象数据类型

在Orcle8中，你可以定义自己的数据类型，例如，可以创建一个包括个人名字的多个部分——名、姓、中名首字母、后缀等——的数据类型作为一个单独的数据类型。NAME_TY数据类型的创建如下：

```
create type NAME_TY as object
(First_Name      VARCHAR2(25),
Middle_Initial   CHAR(1),
Last_Name        VARCHAR2(30),
Suffix           VARCHAR2(5));
/
```

上面清单中的creat type命令对Oracle8有效。可以用用户定义的数据类型来规范应用程序中对数据的使用。例如，你可以在任何地方像使用其他数据类型一样来使用NAME_TY数据类型。在下面的例子中，对EMPLOYEE表进行重建；这次EMPLOYEE.Name列的数据类型是NAME_TY数据类型：

```
create table EMPLOYEE
(EmpNo             NUMBER(10)     PRIMARY KEY,
 Name              NAME_TY,
 DeptNo            NUMBER(2)      DEFAULT 10,
 Salary            NUMBER(7,2)    CHECK (salary<1000000),
 Birth_Date        DATE,
 Soc_Sec_Num       CHAR(9)        UNIQUE,
 foreign key (DeptNo) references DEPT(DeptNo))
tablespace USERS;
```

如上面NAME_TY创建语句中所示，表EMPLOYEE中的Name列包含四个属性。如果用户对NAME_TY数据类型定义一些方法(定义一些对数据类型属性起作用的程序)，就可以把这些方法应用于EMPLOYEE表中Name列的值。有关抽象数据类型及其他面向对象结构的使用和管理例子，请参见第5章。

1. 构造方法

当你创建一个抽象数据类型时，Oracle就自动创建一个构造方法来支持插入到使用这种数据类型的那个列。如NAME_TY数据类型，其构造方法被命名为NAME_TY，这个方法的参数是数据类型的属性。有关如何使用构造方法的例子，请参见第5章。

2. 对象表

对象表是一种其行完全由对象组成的表，它们全部拥有对象ID(OID)值。可以通过create table命令来创建一个对象表。例如，可以利用下面列出的create table命令创建一个基于

NAME_TY数据类型的NAME表。

```
create table NAME of NAME_TY;
```

然后就可以创建从其他表到NAME对象表中行对象的引用，如果创建对NAME行对象的引用，就可以通过引用来选择NAME行而不是直接查询NAME表。有关通过对象视图使用行对象和模拟行对象的例子，请参见第5章。

3. 嵌套表和可变数组

嵌套表是一个表中的一个列(或多个列)，它含有表中一个行的多个值。例如，如果你有一个人的多个地址，就可以在表中创建一个行，该行含有Address列的多个值，但只有一个值可用于其他列。嵌套表可以含有多个列和无限的行。称作可变数组的另一种集合器在其可含有的行数方面受到限制。

对如何使用嵌套表及可变数组的深入讨论超越了本书的范围；请参见《Oracle8完全参考手册》中有关这些结构及其相关语法的详细例解。通常，嵌套表在数据管理上可提供比可变数组更多的灵活性(特别是在Oracle8.0中，由于只能通过PL/SQL来选择可变数组值)。嵌套表及可变数组都需要修改访问数据所使用的SQL语法。一般来说，可以通过相关的关系表来模拟嵌套表中的数据关系。

1.4.4 分区和子分区

随着表的增大，对它的维护也更加困难。在非常大的数据库中，可以通过把一个大表的数据分成多个小表来大大简化数据库的管理活动。例如，可以根据表中的部门或产品值把一个表分成独立的小表。

在Oracle8中，当把一个大表分成若干小表时，可以规定一些范围供数据库使用。这些称作分区(partition)的小表比大表的管理更加简单。例如，可以截断(truncate)一个分区的数据而不截断其他分区的数据。Oracle将把分区表看作一个大表，但可以把这些分区作为一些独立的对象来管理。

分区还可以改善应用性能。由于优化程序将知道作为分区基础使用的范围值，所以它在表访问时就可以只使用特定的分区直接查询。因为在查询进程中只浏览少量数据，自然就改善了查询性能。

除了表外，也可以对索引进行分区，一个分区索引的分区中的值范围可以与索引表使用的范围相匹配，这种情况的索引叫作局部索引(local index)。如果索引分区不能与表分区的值范围相匹配，则该索引就叫作全局索引(global index)。

在Oracle8i中，能够细分分区，创建子分区。例如，可以先根据一组值分割一个表，然后再根据另一种分割方法分割分区。有关分区、子分区、局部索引和全局索引的管理问题，请参见第12章。

1.4.5 用户

用户帐号虽然不是数据库中的一个物理结构，但它与数据库中的对象有着重要的关系，这是因为用户拥有数据库的对象。用户SYS拥有数据字典表，这些表存储了数据库中其他结构的所有信息；用户SYSTEM拥有访问数据字典表的视图，这些视图供数据库中其他用户使用。

当在数据库中创建对象时，必须是在用户帐号下进行。可以对每一个用户帐号进行自定义，以便将一个特定的表空间作为他的缺省表空间使用。

可以把数据库帐号与操作系统帐号连在一起，使用户能从操作系统直接访问数据库，而不必既输入操作系统的口令，又输入数据库的口令。这样，用户就可以访问其拥有的对象或被授权访问的对象。

1.4.6 模式

用户帐号拥有的对象集称为用户的模式 (schema)。可以创建不能注册到数据库的用户帐号。这样的用户帐号提供一种模式，这种模式可以用来保存一组同其他用户模式分开的数据库对象。

1.4.7 索引

在关系数据库中，一个行的物理位置无关紧要，除非数据库需要找到它。为了能找到数据，表中的每一行均用一个 RowID来标识。 RowID告诉数据库这一行的准确位置 (指出行所在的文件、该文件中的块、该块中的行地址)。

注意 索引结构表没有传统的Oracle RowID，不过，其主键起一个逻辑 RowID的作用。

索引是一种供服务器在表中快速查找一个行的数据库结构。索引有三种形式：簇索引、表索引和位映射索引。簇索引把簇关键字值存储在簇中；下面一小节将对簇的用途进行详细描述。表索引除了确定行的物理位置 (RowID)外，还存储表的行值。位映射索引是表索引的一种特殊形式，用于支持对大表进行查询 (这些大表很少有不同值的列)。

每一个索引条目都由一个键值和 RowID组成。可以索引一个列或一组列， Oracle用B*树 (B*-tree)机制存储索引条目，以保证用最短路径访问键值。当一个查询访问索引时，就能找到与查询条件相匹配的索引条目。与条目相匹配的 RowID值向Oracle提供相关行的物理位置，以减轻定位数据所需要的I/O负担。

索引既可以改善数据库性能，又可以保证列值的唯一性。当 create table命令中规定有UNIQUE或PRIMARY KEY约束条件子句时， Oracle就自动创建一个索引。也可以通过 create index命令来手工创建自己的索引。有关 create index命令的语法和选项的全部信息，请参见附录A。

可以在表中的一列或多列上创建索引。如果使用前面给出的 EMPLOYEE表例子， Oracle将自动在 EmpNo及Soc_Sec_Num列上创建唯一索引，因为这两列上分别声明了 PRIMARY KEY及UNIQUE。撤消一个索引将不影响先前进行过索引的表中的数据。

在Oracle7.3中，可以创建位映射索引 (bitmap index)。如第12章所述，当数据的可选择性很小(列中极少有不同值时)，位映射索引就非常有用。位映射索引可加快搜索，这样无选择性的列就能用作查询中的约束条件。位映射索引对于完全静态的数据非常有效。

在Oracle8中，可以用与存储顺序相反 (reverse)的数据创建索引。例如，数值是“ 1002 ”的条目可以按“ 2001 ”创建索引。在创建索引之前将数据顺序反转可以使数据在索引中有更好的分布。由于反转索引将数值颠倒，反转顺序索引只能用于执行等值查询，例如：

```
where key_col_value = 1002
```

如果正在执行某一范围内的搜索，如：

```
where key_col_value > 1000
```

那么反转顺序索引就无法有效地满足要求，因为相邻的值并不相近存储。由于连续的行在索引中没有存储在一起，所以若对反转关键字索引的范围进行查询，就必须读取比传统索引要多的块。

在Oracle8中，能够创建索引组织表。在索引组织表(通过create table命令中的organization index子句规定)中，整个表存储在一个索引结构中，数据按表中的键存储。要创建一个索引组织表，就必须为表规定一个主键约束条件。索引组织表没有行的RowID(行标识)，Oracle将把主键值用作逻辑RowID值。在Oracle8i中，可以对一个索引组织表创建二级索引。

1.4.8 簇

经常被同时访问的表在物理位置上可以存储在一起。为了将它们存储在一起，就要创建一个簇(cluster)来管理这些表。表中的数据一起存储在簇中，从而最小化必须执行的I/O次数，改善系统性能。

表中相关的列称为簇键(cluster key)。簇键用一个簇索引(cluster index)来进行索引；对于簇中的多个表，簇键值只存储一次。在把任何行插入簇的表中之前，都必须先创建一个簇索引。

对于经常频繁一起查询的表说，使用簇比较方便。在簇中，来自不同表的行存储在同一个块中；因此同将表分开存储相比，连接这些表的查询就可能执行更少的I/O。不过，与对非簇表的相同操作比较，簇表的插入、更新和删除性能要差很多。在聚簇表之前，要判断共同查询这些表的频率。如果这些表总是一起查询，就要考虑把它们合并成一个表而不是聚簇两个表。

1.4.9 散列簇

散列簇(hash cluster)是簇的另一种形式。为确定行应存储的物理位置，对行的簇键使用散列函数(hashing function)。散列簇可以大大提高等值查询的效率，例如下列查询：

```
select Name
  from EMPLOYEE
 where EmpNo = 123;
```

在这个例子中，按EmpNo列的准确匹配对EMPLOYEE表进行查询。如果EMPLOYEE表是散列簇的一部分，EmpNo就是簇键的一部分，数据库就可以使用散列函数来快速确定数据的物理位置。如果where子句指定了一个值范围，就无法获得预期的效果，如下所示：

```
select Name
  from EMPLOYEE
 where EmpNo > 123;
```

查找标准索引表中的一个行可能需要多次I/O：一次(或多次)查找索引中的键值，另一次从表中读取行。使用散列算法将减少等值查询时返回行所需要的I/O次数。

1.4.10 视图

视图(view)像表一样由列组成，其查询方式与表相同。但是视图中没有数据。从理论上讲，可以把视图看作是覆盖一个或多个表的“蒙版”，视图中的列可以在一个或多个基本表中找到。

所以视图不使用物理存储位置来存储数据。视图的定义 (包括作为基础的查询、列安排、授予的权限)存储在数据字典中。

对一个视图进行查询时，视图将查询其基于的表，并且以视图定义所规定的格式和顺序返回值。由于视图没有直接相关的物理数据，所以不能被索引。

视图经常用于对数据设置行级保密和列级保密。例如，可以授权用户访问一个视图，这个视图只显示该用户可以访问的行，而不显示表中所有行。同样，可以通过视图限制该用户访问的列。

对于Oracle8，可以采用对象视图在表上创建一个面向对象的层。可以使用对象视图模拟抽象数据类型、对象ID和引用。有关对象视图的应用实例，请参见第5章。

对象视图

如果使用抽象数据类型，执行时可能会遇到一致性问题。访问抽象数据类型属性所使用的语法不能用于访问正规列。因此，为了支持抽象数据类型，可能要改变 SQL企业编码标准。在进行事务处理和对表查询时，还要记住哪个表使用了抽象数据。

对象视图为通向抽象数据类型之路架起了一座重要的桥梁。可以使用对象视图为自己的关系数据提供一个对象关系图像。基本表不变化，但视图支持抽象数据类型定义。从数据库管理员的观点来看，发生的是一些微小变化 —— 就像管理数据库中其他任何表一样管理基本表。从开发人员的观点来看，对象视图提供了对表数据对象关系的访问。有关实现和使用对象视图的详细情况，请参见第5章。

1.4.11 序列

序列(sequence)定义存储在数据字典中。序列通过提供唯一数值的顺序表用来简化程序设计工作。

当一个序列第一次被查询调用时，它将返回一个预定值。在随后的每一次查询中，序列将产生一个按其指定的增量增长的值。序列可以是循环的，或者是连续增加的，直到指定的最大值为止。

使用一个序列时，不保证将生成一串连续不断的值。例如，如果查询一个序列的下一个值供insert使用，则该查询是能使用这个序列值的唯一会话。如果未能提交事务处理，则序列值就不被插入表中，以后的 insert将使用该序列随后的值。

1.4.12 过程

过程(procedrure)是一个PL/SQL语句块，它存储在数据字典中并可被应用程序调用。你可以使用过程存储数据库中频繁使用的应用逻辑。当执行一个过程时，其语句被作为一个整体执行。过程不将任何值返回调用程序。

可以使用存储的过程来帮助实施数据的安全性。可以不授权用户直接访问应用程序中的一些表，而是授权用户执行访问这些表的一个过程。当过程被执行时，它将以过程拥有者的权限来执行。除非通过过程，否则用户就不能访问这些表。

1.4.13 函数

与过程一样，函数 (function)是存储在数据库中的代码块。其差别在于函数可以把值返回

调用程序。你可以建立自己的函数，并在SQL语句中调用它们，就像执行Oracle提供的函数一样。

例如，Oracle提供一个SUBSTR函数来执行字符串上的“取子串”操作，如果创建一个叫做MY-SUBSTR的函数来执行一个自定义的取子串操作，就可以在一个SQL命令中调用它：

```
select MY_SUBSTR('text') from DUAL;
```

如果你不拥有MY_SUBSTR函数，则必须被授予对这个函数的EXECUTE(执行)权限。如果用户定义函数不改变数据库的任何行，在SQL语句中就只能使用一个这样的函数。

1.4.14 软件包

可以使用软件包(package)将过程及函数安排在逻辑分组中。包说明和包体都存储在数据字典中。软件包在过程及函数所需要的管理任务中非常有用。

软件包中的不同元素可以定义为“公用”或“私有”。公用元素可以被软件包的用户使用，私有元素则对用户是隐藏的。私有元素可以包括那些被包中其他过程调用的过程。

函数、软件包和过程的源代码被存储在数据字典表中。如果应用程序经常使用软件包，就必须增加SYSTEM表空间的容量以适应数据字典容量的增长。例如在Oracle财务软件(Financial)工具中，就可能需要一个大于250MB的SYSTEM表空间。

软件包的数量及复杂性在使用中直接影响着SGA的SQL共享池(Shared SQL Pool)部分的大小。有关SQL共享池的情况，将在本章的1.5节“内部存储结构”中描述。

1.4.15 触发器

触发器(trigger)是一些过程，当一个特定的数据库事件发生时就执行这些过程。可以使用触发器来扩充引用完整性，实施附加的安全性或增强可用的审计选项。

触发器分为两种类型：

- 语句触发器：对每一个触发语句触发一次。
- 行触发器：对受语句影响的表中的每一个行触发一次。

例如，语句级触发器对删除10000行的delete(删除)命令触发一次；而行级触发器将对相同的事务处理触发10000次。

无论对于哪一种触发器，都能为每种触发事件类型创建BEFORE触发器及AFTER触发器。触发事件包括insert(插入)、update(更新)和delete(删除)。

如果触发器的动作代码不取决于受影响的数据，语句触发器就非常有用。例如，可以在表上创建一个BEFORE INSERT语句触发器，以防止在某些特定期限以外的时间对一个表进行插入。

如果触发器动作取决于受事务处理影响的数据，行触发器就非常有用。例如，可以在表上创建一个AFTER INSERT行触发器来把一个新行插入一个审计表及触发器的基本表中。

对于Oracle8，可以创建INSTEAD OF触发器。INSTEAD OF触发器执行一个替代操作来代替触发器的操作。也就是说，如果对表创建一个INSTEAD OF INSERT触发器，将执行触发器的代码且绝不会出现引起触发器执行的insert操作。INSTEAD OF触发器也可以应用于视图中。如果一个视图在查询中涉及到多个表，只要用户试图通过视图更新行，INSTEAD OF触发器就能引导Oracle操作。

对于Oracle8i，可以对系统级事件创建触发器。当发出 create(创建)、alter(变更)或delete(删除)命令时，就可以触发被执行代码。也可以把登录、注销、数据库关闭和启动之类的系统事件用作触发事件。有关命令语法的情况请参见附录 A中的创建触发器(create trigger)条目。

1.4.16 同义词

在一个分布式数据库环境中，为了完全识别一个数据库对象如表或视图，必须规定主机名、服务器(实例)名、对象的拥有者和对象名。具体需要多少参数根据对象的位置确定，一般需要1至4个这样的参数。为了筛选用户的这个进程，开发者可以创建指向适当对象的同义词，这样用户使用时只需知道同义词名即可。公用同义词由一个特定数据库的所有用户共享；私有同义词只被数据库的各个用户帐号所有者所拥有。

例如，前面所述的EMPLOYEE表必须由一个帐号(例如HR)所拥有，对于同一数据库的其他用户帐号，这个表可以作为HR.EMPLOYEE被引用。然而，这种语法需要另一个帐号知道：HR帐号是EMPLOYEE表的所有者。为避免这种情况发生，可以创建一个公用同义词EMPLOYEE来指向HR.EMPLOYEE，无论何时引用同义词EMPLOYEE，它都将指向正确的表。下列SQL语句创建同义词EMPLOYEE：

```
create public synonym EMPLOYEE for HR.EMPLOYEE;
```

同义词提供表、视图、过程、函数、包和序列的指针，这些指针指向本地数据库或远程数据库中的对象。指向远程数据库时需要通过使用数据库链接来完成，本章后面将对数据库链接进行介绍。

不能为抽象数据类型创建同义词。进一步讲，当你创建同义词时，Oracle不检查这个同义词的有效性。在创建同义词后，用户应自行测试，以保证同义词的有效性。

1.4.17 权限及角色

为访问其他帐号所拥有的对象，必须首先被授予访问这个对象的权限。一般情况下，非所有者可以被授予对表、快照或视图的行进行 insert(插入)、select(选择)、update(更新)、delete(删除)的权限，也可以授予用户select(选择)序列值和execute(执行)过程、函数、包和抽象数据类型的权限。由于索引或触发器只能在表操作过程中被数据库访问，所以没有针对索引或触发器访问的权限。也以授予对目录(对于BFILE数据类型)的read(读)权限和对库(由应用代码调用的外部程序)的execute(执行)权限。权限可以授予给某个用户或PUBLIC，PUBLIC把权限授予数据库中的全体用户。

可以创建角色(role)——权限组——来简化权限管理进程。可以把一些权限授予一个角色，而这个角色又可以被授予多个用户。由于把角色授予用户或从用户取消角色是一个简单的事情，所以给应用程序增加一些新用户就成为一个非常容易的管理过程。

权限与角色的关系如图1-4所示，在图1-4a中，把对两个表的select(选择)访问权授予四个用户所需要的权限用线条表示；图 1-4b中的角色用于简化对权限的管理。许多权限被授予一个角色，这个角色又被授予四个用户。在应用程序中角色可以被动态地允许或禁止。

也可以使用授予系统级权限的角色，如 create table (创建表)。有关系统级角色将在第5章和第9章中详细讨论。

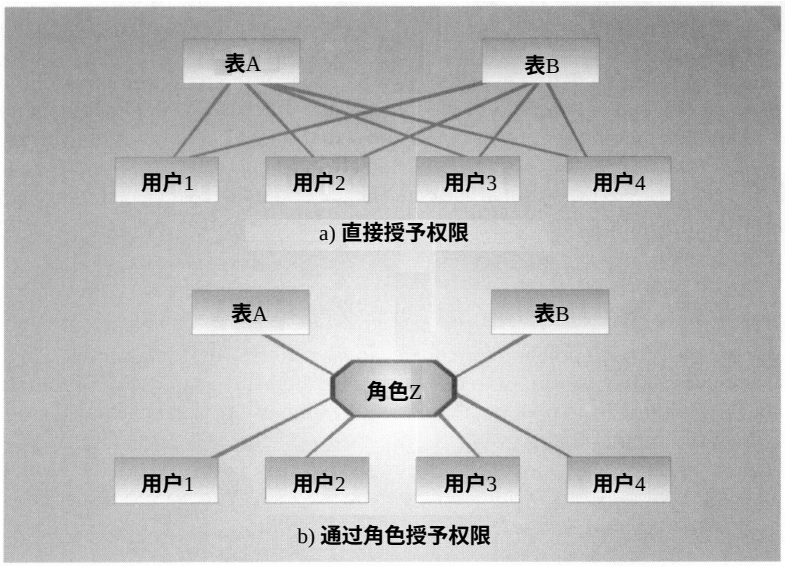

图1-4 权限与角色之间的关系

1.4.18 数据库链接

Oracle数据库具有引用本地数据库以外数据的能力。当需要引用这样的数据时，必须指定远程对象的全限定名。在前面描述的同义词例子中，只是全限定名的两个部分——所有者及表名——被指定。如果表在一个远程数据库中又该怎么办呢？

为了指定对远程数据库中一个对象的访问路径，必须创建一个数据库链接，数据库链接既可以公用(数据库中的所有帐号都可以使用)，也可以私有(只为某个帐号的用户创建)。当创建一个数据库链接时，必须指定与数据库相链接的帐号名、帐号的口令以及与远程数据库相连的服务器名字。如果不指定帐号名，Oracle将使用本地帐号名和口令来建立与远程数据库的链接。下面是创建一个名为MY_LINK的公用链接例子：

```
create public database link MY_LINK
connect to HR identified by PUFFINSTUFF
using 'DB1';
```

这个例子中的链接规定，当使用这个链接时，它将打开由服务(service)DB1指定的数据库中的一个对话。当它在DB1实例中打开一个对话时，将按用户帐号为“HR”、口令为“puffinstuff”来注册。实例的服务名存储在由SQL*Net(最新的SQL*Net版本称为Net8)使用的配置文件中。服务名的配置文件称为tnsnames.ora，它指定与每个服务名相关的主机、端口及实例。

为使用这个链接来访问一个表，链接必须用from子句来指定，如下例所示：

```
select * from EMPLOYEE@MY_LINK;
```

上述查询将通过MY_LINK数据库链接来访问EMPLOYEE表，也可以为该表创建一个同义词，如下面的SQL命令所示：

```
create synonym EMPLOYEE for EMPLOYEE@MY_LINK;
```

请注意，数据库对象的全限定标志已被定义，其中包括通过服务名的主机和实例、通过

数据库链接的拥有者(HR)和表名(EMPLOYEE)。

EMPLOYEE表的位置因而变得对终端用户完全透明。可以把EMPLOYEE表移到一个不同的模式或不同的数据库中。改变数据库链接的定义将把该同义词转移到新的位置。

注意 如果存储过程、包或触发器包含对一个数据库链接的引用，则必须有供PL/SQL编译用的链接。

1.4.19 段、盘区和块

段(segment)是逻辑数据库对象的物理副本。段存储数据。例如，索引段存储与索引相关的数据。对段的有效管理需要数据库管理员了解应用程序将使用的对象、数据如何进入这些对象以及检索数据的方法。

由于段是一个物理实体，所以必须把它分配到数据库中的一个表空间中(放在表空间的一个数据文件上)。段由称作盘区(extent)的一些邻接的Oracle块集合组成。一旦段中的现有盘区不能再容纳新数据，该段将获取另外的盘区。如果需要的话，这种扩展将持续下去，直到表空间的数据文件中没有自由空间或者已达到每个段内部的盘区最大数量为止。如果一个段中有多个盘区，将无法保证这些盘区连接在一起。

对每个段最多能获得多少盘区都有范围限制。尽管在理论上对段中盘区的数量限制为数十亿，但是如果每个表的盘区数量限制在10000以内的话，大多数数据库维护操作就执行得很好。可以在段级指定每个段中盘区的最大数量，也可以用缺省方式表示段的表空间。

当你撤消一个段时，该段所使用的盘区就成为自由盘区。Oracle可以重新把这些自由盘区用于新的段或现有段的扩展。有关对特定段类型的管理方法，请参见第4章和第7章。

1.4.20 回滚段

为了保证数据库中多用户读的一致性和能够回滚事务，Oracle必须拥有一种机制来重新构造一个前映像(before image)数据，为未提交的事务服务。Oracle用数据库中的回滚段来提供一个前映像数据。

事务利用回滚段来记录变化前的数据映像。例如，一个大的delete(删除)操作需要一个大的回滚段来保存被删除的记录。如果delete(删除)事务被回滚，则Oracle将使用回滚段重新构造该数据。

查询也使用回滚段。由于Oracle执行读一致性的查询操作，所以数据库必须能把数据重新构造得与开始查询时存在的数据一样。如果一个事务在查询开始后结束，Oracle将连续使用这个事务的回滚段条目来重新构造更改过的行。通常，应避免安排长时间执行并发查询的事务。

回滚段随它所支持的事务的增大而增大。如何有效地管理回滚段将在第7章中描述。

1.4.21 快照和显形图

可以使用快照(sanpshot)把远程数据的本地拷贝提供给用户。快照基于一个查询，该查询使用数据库链接选择远程数据库中的数据。把快照的数据从源数据库复制到目标数据库，然后用户就可以查询快照。可以把快照设置成只读方式或可更新方式。若要改善性能，可以索引快照使用的本地表。

根据快照基本查询的复杂性，可以使用快照日志(snapshot log)来提高复制操作的性能。

复制操作根据用户为每个快照的安排自动完成。有关创建和维护快照及快照日志的详细情况，请参见本书的第三部分。

对于Oracle8i，你可以创建显形图(materalized view)。显形图在结构上与快照非常相似。它存储基于一个基本查询的复制数据。快照一般存储来自远程数据库的数据，而显形图通常则存储从当前数据库中复制的数据。在数据库操作期间，如果显形图要返回相同的数据，优化程序就可以动态地选择使用一个可利用的显形图，以代替对一个更大数据表的查询。显形图还可以提供潜在的性能，但是要增加空间占用和维护代价。有关实现显形图的详细情况，请参见第12章。

1.5 内部存储结构

Oracle使用共享内存区和后台进程来管理其内存和文件结构。后台进程将在下节描述，本节重点描述由Oracle数据库全体用户使用的全局存储区。

根据所使用的数据库服务器选项，存储选项的可选择范围是非常广泛的。最通用的实现方法在本章和第2章中描述。

本节内容包括：

- 系统全局区(SGA)。
- 数据块缓存区。
- 字典缓存区。
- 重做日志(redo log)缓冲区。
- SQL共享池。
- 大池。
- Java池。
- 多缓冲区池。
- 环境区。
- 程序全局区(PGA)。

1.5.1 系统全局区

如果有人已经读过本书的某一章，要把这些知识传达给其他人的最快方法是哪一种呢？可以请其他人也自己去读，但这不是最快的方法，最快的方法是将自己头脑中掌握的这方面的全部知识传授给其他人。

Oracle数据库中的系统全局区(SGA，System Global Area)用于同一目的——在用户间有效地传输信息，它也包含有关数据库的最通用结构信息。

系统全局区(SGA)的构成如图1-5所示，图中所示的存储单元的具体描述将在以后几节中进行。

1. 数据块缓存区

数据块缓存区(data block buffer cache)是SGA中的一个高速缓存区域，用来存储从数据库中读取数据段的数据块(如表、索引和簇)。数据块缓存区的大小由数据库服务器init.ora文件中的DB_BLOCK_BUFFERS参数决定(用数据库块的个数表示)。在调整和管理数据库时，调整数据块缓存区的大小是一个重要的部分。

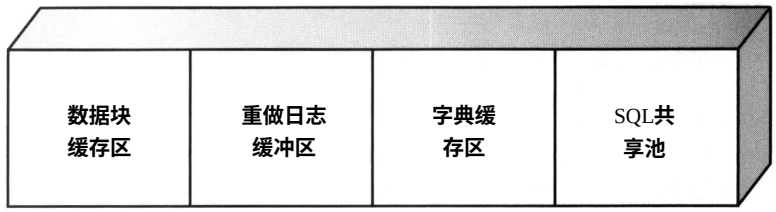

图1-5 SGA结构

因为数据块缓存区的大小固定，并且其大小通常小于数据库段所使用的空间，所以它不能一次装载下内存中所有的数据库段。通常，数据块缓存区只是数据库大小的 1%～2%，Oracle使用最近最少使用 (LRU，least recently used)算法来管理可用空间。当存储区需要自由空间时，最近最少使用块将被移出，新数据块将在存储区代替它的位置。通过这种方法，将最频繁使用的数据保存在存储区中。

然而，如果SGA的大小不足以容纳所有最常使用的数据，那么，不同的对象将争用数据块缓存区中的空间。当多个应用程序共享同一个 SGA时，很有可能发生这种情况。此时，每个应用的最近使用段都将与其他应用的最近使用段争夺 SGA中的空间。其结果是，对数据块缓存区的数据请求将出现较低的命中率，导致系统性能下降。有关如何控制数据块缓存区使用的信息，请参见第6章中的6.5.2节“统计报表解释”。

2. 字典缓存区

数据库对象的信息存储在数据字典表中，这些信息包括用户帐号数据、数据文件名、段名、盘区位置、表说明和权限，当数据库需要这些信息 (如检查用户查询一个表的授权)时，将读取数据字典表并且将返回的数据存储在字典缓存区的 SGA中。

数据字典缓存区通过最近最少使用 (LRU) 算法来管理。字典缓存区的大小由数据库内部管理。字典缓存区是SQL共享池的一部分，共享池的大小由数据库文件 init.ora中的SHARED_POOL_SIZE参数来设置。

如果字典缓存区太小，数据库就不得不反复查询数据字典表以访问数据库所需的信息，这些查询称为循环调用(recuesive call)，这时的查询速度相对字典缓存区独立完成查询时要低。有关如何监控字典缓存区使用的有关信息，请参见第 6章的6.5.2节“统计报表解释”。

3. 重做日志缓冲区

重做日志文件将在本章 1.7.1节“重做日志”中加以描述。重做项描述对数据库进行的修改。它们写到联机重做日志文件中，以便在数据库恢复过程中用于向前滚动操作。然而，在被写入联机重做日志文件之前，事务首先被记录在称作重做日志缓冲区 (redo log buffer)的SGA中。数据库可以周期地分批向联机重做日志文件中写重做项的内容，从而优化这个操作。

重做日志缓冲区的大小(以字节为单位)由init.ora文件中的LOG_BUFFER参数决定。

4. SQL共享池

SQL共享池存储数据字典缓存区及库缓存区 (library cache)，即对数据库进行操作的语句信息。当数据块缓冲区和字典缓存区能够共享数据库用户间的结构及数据信息时，库缓存区允许共享常用的SQL语句。

SQL共享池包括执行计划及运行数据库的 SQL语句的语法分析树。在第二次运行 (由任何用户)相同的SQL语句时，可以利用 SQL共享池中可用的语法分析信息来加快执行速度。有关监控SQL共享池使用的信息，请参见第 6章的6.5.2节“统计报表解释”一节。

SQL共享池通过LRU算法来管理。当SQL共享池填满时，将从库缓存区中删掉最近最少使用的执行路径和语法分析树，以便为新的条目腾出空间。如果SQL共享池太小，语句将被连续不断地再装入到库缓存区，从而影响操作性能。

SQL共享池的大小(以字节为单位)由init.ora文件参数SHARED_POOL_SIZE决定。

5. 大池

大池(Large Pool)是一个可选内存区。如果使用线程服务器选项或频繁执行备份/恢复操作，只要创建一个大池，就可以更有效地管理这些操作。大池将致力于支持SQL大型命令。利用大池，就可以防止这些SQL大型命令把条目重写入SQL共享池中，从而减少再装入到库缓存区中的语句数量。大池的大小(以字节为单位)通过init. ora文件的LARGE_POOL_SIZE参数设置，用户可以使用init.ora文件的LARGE_POOL_MIN_ALLOC参数设置大池中的最小位置。Oracle8i已不用这个参数。

作为使用Large Pool的一种选择方案，可以用init.ora文件的SHARED_POOL_RESERVED_SIZE参数为SQL大型语句保留一部分SQL共享池。

6. Java池

由其名字可知，Java池为Java命令提供语法分析。Java池的大小(以字节为单位)通过在Oracle8i引入的init.ora文件的JAVA_POOL_SIZE参数设置。init.ora文件的JAVA_POOL_SIZE参数缺省设置为10MB。

7. 多缓冲池

可以在SGA中创建多个缓冲池，能够用多个缓冲池把大数据集与其他的应用程序分开，以减少它们争夺数据块缓存区内相同资源的可能性。对于创建的每一个缓冲池，都要规定其LRU锁存器的大小和数量。缓冲区的数量必须至少比LRU锁存器的数量多50倍。

创建缓冲池时，需要规定保存区(keep area)的大小和再循环区(recycle area)的大小。与SQL共享池的保留区一样，保存区保持条目，而再循环区则被频繁地再循环使用。可以通过BUFFER_POOL_KEEP参数规定来保存区的大小。例如：

```
BUFFER_POOL_KEEP=(buffers:200, lru_latches:3)
BUFFER_POOL_RECYCLE=(buffers:50, lru_latches:1)
```

保存和再循环缓冲池的容量减少了数据块缓冲存储区中的可用空间(通过DB_BLOCK_BUFFERS参数设置)。对于使用一个新缓冲池的表，通过表的storage子句中的buffer_pool参数来规定缓冲池的名字。例如，如果需要从内存中快速删除一个表，就把它赋予RECYCLE池。缺省池叫作DEFAULT，这样就能在以后用alter table命令把一个表转移到DEFAULT池。

1.5.2 环境区

在SQL共享区中，分为公用或私有两个区域，用户发出的每一个SQL语句都需要一个SQL私有区，这个区一直保存到与该语句相应的游标关闭为止。对于Oracle8，当使用与对象相关的特征时，同样要使用一个私有对象缓存区。

1.5.3 程序全局区

程序全局区(PGA，Program Global Area)是存储区中的一个区域，由一个Oracle用户进程所使用，PGA中的内存不能共享。

如果正在使用多线程服务器(MTS，Multithreaded Server)，PGA的一部分可以被存储在SGA中。多线程服务器结构允许多个用户进程使用同一个服务器进程，从而减少数据库存储区的需要。如果使用MTS，用户对话信息就存储在SGA中而不是PGA中。如果使用MTS，应增加SQL共享池的容量以适应另外一些共享存储区的要求。

1.6 后台进程

数据库的物理结构和存储结构之间的关系由后台进程来维持。数据库拥有多个后台进程，其数量取决于数据库的配置。这些进程由数据库管理，它们只需要进行很少的管理。

每个后台进程创建一个跟踪文件。在实例操作期间保存跟踪文件。后台进程跟踪文件的命名约定和位置随操作系统和数据库版本而不同。一般来说，跟踪文件含有后台进程名或后台进程的操作系统进程ID。可以设置init.ora文件的BACKGROUND_DUMP_DEST参数来规定后台进程跟踪文件的位置，但是有些版本的Oracle忽略这种设置。排除数据库故障时，跟踪文件就显得非常重要。影响后台进程的严重问题通常记录在数据库的警告日志上。

警告日志通常位于BACKGROUND_DUMP_DEST目录下。一般来说，这个目录是ORACLE_BASE目录下的/admin/*INSTANCE_NAME*/bdump目录。

1. SMON

当启动一个数据库时，SMON(System Monitor，系统监控程序)进程执行所需的实例恢复操作(使用联机重做日志文件)，它也可以清除数据库，取消系统不再需要的事务对象。

SMON的另一个用途是：将邻接的自由盘区组成一个较大的自由盘区。自由空间碎片进程在第4章中描述。对于某些表空间，数据库管理员必须手工执行自由空间合并；第8章将对执行这个任务的指令进行描述。SMON只合并表空间中的自由空间，这些表空间的缺省pctincrease存储值为非零。

2. PMON

PMON(进程监控程序)后台进程清除失败用户的进程，释放用户当时正在使用的资源。当一个持有锁的进程被取消时，其效果是显而易见的，PMON负责释放锁并使其可以被其他用户使用。同SMON一样，PMON周期性地唤醒检测它是否需要被使用。

3. DBWR

DBWR(数据库写入程序)后台进程负责管理数据块缓存区及字典缓存区的内容。它以批方式把修改块从SGA写到数据文件中。

尽管每一个数据库实例只有一个SMON和一个PMON进程在运行，但是根据平台和操作系统的不同，用户可以同时拥有多个DBWR进程。使用多个DBWR进程有助于在进行大的操作期间减少DBWR中的冲突。所需DBWR进程的数量由数据库的init.ora文件中的DB_WRITER_PROCESSES参数决定。如果系统支持异步I/O，可以用多个DBWR I/O从(slave)进程创建一个DBWR进程。DBWR I/O从进程的数量由init.ora文件的DBWR_I/O_SLAVES参数设置。

如果创建多个DBWR进程，这些进程就不叫做DBWR，它们将有一个数字分量。例如，如果创建5个DBWR进程，进程的操作系统名就可能是DBW0、DBW1、DBW2、DBW3和DBW4。

4. LGWR

LGWR(日志写入程序)后台进程负责把联机重做日志缓冲区的内容写入联机重做日志文

件。LGWR分批将日志条目写入联机重做日志文件。重做日志缓冲区条目总是包含着数据库的最新状态，这是因为DBWR进程可以一直等待到把数据块缓冲区中的修改数据块写入到数据文件中。

LGWR是数据库正常操作时唯一向联机重做日志文件写入内容并从重做日志缓冲区直接读取内容的进程。与DBWR对数据文件执行的完全随机访问相反，联机重做日志文件以序列形式写入。如果联机重做日志文件是镜像文件， LGWR同时向镜像日志文件中写内容。有关这种镜像性能的详细情况，请参见第2章。

对于Oracle8，可以创建多个LGWR I/O从进程以改善向联机重做日志文件的写入性能，其个数由数据库的init.ora文件的LGWR_IO_SLAVES参数决定。

在Oracle8i中，这个参数已不能用， LGWR I/O从进程由DBWR_IO_SLAVES设置值派生而来。

5. CKPT

CKPT(检查点进程)用来减少执行实例恢复所需的时间。检查点使DBWR把上一个检查点以后的全部已修改数据块写入数据文件，并更新数据文件头部和控制文件以记录该检查点。当一个联机重做日志文件被填满时，检查点进程会自动出现。可以用数据库实例的 init.ora文件中的LOG_CHECKPOINT_INTERVAL参数来设置一个频繁出现的检查点。

CKPT后台进程把早期数据库版本中LGWR的两个功能(向检查点发信号及复制日志内容)分成两个后台进程。当数据库实例的init.ora文件中的CHECKPOINT_PROCESS参数被设置为TRUE时，就可以建立CKPT后台进程。

6. ARCH

LGWR后台进程以循环方式向联机重做日志文件写入；当填满第一个日志文件后，就开始向第二个日志文件写入；第二个日志文件填满后，再向第三个日志文件写入。一旦最后一个重做日志文件填满，LGWR就开始重写第一个重做日志文件的内容。

当Oracle以ARCHIVELOG(归档日志)模式运行时，数据库在开始重写重做日志文件之前先对其进行备份。这些归档的重做日志文件通常写入一个磁盘设备中。也可以直接写入磁带设备中，但是这往往要增加操作员的劳动强度。

这种归档功能由ARCH(归档进程)后台进程完成，利用该性能的数据库在处理大数据事务时将遇到重做日志磁盘冲突问题，这是因为当 LGWR准备写入一个重做日志文件时， ARCH正准备读取另一个。如果归档日志目标磁盘写满，数据库还将遇到数据库锁定问题。此时，ARCH冻结，禁止LGWR写入；从而禁止在数据库中出现进一步的事务处理；这种情况一起延续到归档重做日志文件的空间清空为止。

对于Oracle8，可以创建多个ARCH I/O从进程以改善对归档重做日志文件的写入功能。在Oracle8.0中，ARCH I/O从进程的个数由数据库的init.ora文件中的ARCH_IO_SLAVES参数决定。在Oracle8i中，这个参数已不能用， ARCH_IO_SLAVES设置值由DBWR_IO_SLAVES设置值派生。

有关归档进程及数据库后台进程的管理细节，请参见第10章。

7. RECO

RECO(恢复进程)后台进程用于解决分布式数据库中的故障问题。 RECO进程试图访问存在疑问的分布式事务的数据库并解析这些事务。只有在平台支持 Distributed Option(分布式选

项)且init.ora文件中的DISTRIBUTED_TRANSACTIONS参数大于零时才创建这个进程。

8. SNP*n*

Oracle的快照刷新及内部作业队列调度依赖于它们执行的后台进程 (快照进程)。这些后台进程的名字以字母SNP开头，以数字或字母结束。为一实例所创建的 SNP进程的个数由数据库的init.ora文件中的 JOB_QUEUE_PROCESSES参数决定 (在Oracle7中，该参数名为SNAPSHOT_REFRESH_PROCESSES)。

9. LCK*n*

当采用Oracle并行服务器选项时，多个LCK(锁定进程)后台进程(命名为LCK0~LCK9)用于解决内部实例的锁定问题。 LCK进程的个数由GC_LCK_PROCS参数决定。

10.D*nnn*

Dnnn(调度程序进程)是MTS结构的一部分；这些进程有助于减少处理多重连接所需要的资源。对于数据库服务器支持的每一个协议必须至少创建一个调度程序进程，调度程序进程根据SQL*Net(或Net8)的配置在数据库启动时创建，在数据库打开后可以创建或取消。

11. S*nnn*

创建S*nnn*(服务器进程)来管理需要专用服务器的数据库连接。服务器进程可以对数据文件进行I/O操作。

12. P*nnn*

如果启动数据库中的并行查询选项，一个查询的资源要求可以分布在多个处理器中。当实例启动由init. ora文件的PARALLEL_MIN_SERVERS参数确定时，指定数目的并行查询服务器进程就启动。每一个这样的进程都将出现在操作系统级。需要并行操作的进程越多，启动的并行查询服务器进程就越多。每个并行查询服务器进程在操作系统级将有一个 P000、P001、P002这样的名字。并行查询服务器进程的最大数量由 init. ora文件的PARALLEL_MAX_SERVERS参数确定。

并行查询服务器进程不生成跟踪文件。有关执行并行查询选项的详细情况，请参见第 12章。

1.7 外部结构

如本章前面1.2.2节“文件”部分所述，数据库的数据文件为数据库的数据提供物理存储区。因此，它们既是“内部”结构 (因为它们直接与表空间相连接)，又是“外部”结构 (因为它们是物理文件)。跨设备的分布式规划进程将在第 4章描述。

下面是从数据文件中分离出来的、与数据库相关连的文件类型。这些文件包括：

- 重做日志。
- 控制文件。
- 跟踪文件及警告日志。

1.7.1 重做日志

Oracle保存所有数据库事务的日志。这些事务被记录在联机重做日志文件 (online redo log file)中。当数据库被破坏时，这些日志文件能够以正确顺序恢复数据库事务。重做日志文件信息存储在数据库数据文件的外部。

重做日志文件也可以让Oracle优化向磁盘写入数据的方式。当数据库中出现一个事务时，就把该事务输入到重做日志缓冲区；同时受该事务影响的数据块不会立即写入磁盘。

每个Oracle数据库都有两个或更多的联机重做日志文件。 Oracle以循环方式向联机重做日志文件写入：第一个日志文件被填满后，就向第二个日志文件写入，然后依次类推。当所有联机重做日志文件都被填满时，就又回到第一个日志文件，用新事务的数据对其进行重写。如果数据库正以ARCHIVELOG模式运行，在重写联机重做日志文件前，数据库将先对其进行备份。任何时候都可以用这些归档重做日志文件来恢复数据库的任何部分 (参见第10章)。

重做日志文件可以被数据库镜像 (复制)。镜像联机重做日志文件不依赖操作系统或操作环境的硬件性能就可以对重做日志文件进行镜像。有关镜像功能的详细情况，请参见第 2章。

1.7.2 控制文件

数据库的全局物理结构由其控制文件 (control file)维护。控制文件记录数据库中所有文件的控制信息。控制文件维护内部的一致性并引导恢复操作。

由于控制文件对数据库至关重要，所以联机存储着多个拷贝。这些文件一般存储在各个不同的磁盘上，以便将因磁盘失效引起的潜在危险降至最低限度。创建数据库时，同时就提供与之对应的控制文件。

数据库控制文件的名字通过 init. ora文件的CONTROL_FILES参数规定。尽管这是一个init.ora参数，但是CONTORL_FILES参数通常用config.ora文件规定，因为它很少变化。如果需要给数据库添加一个新的控制文件，可关闭实例，把已存在的一个控制文件复制到新的地址，把新的地址添加到CONTROL_FILES参数设置值上，并重新启动这个实例。

1.7.3 跟踪文件与警告日志

在实例中运行的每一个后台进程都有一个跟踪文件与之相连。跟踪文件记载后台进程遇到的重大事件的信息。除了跟踪文件外， Oracle还有一个称作警告日志 (alert log)的文件，警告日志记录数据库文件运行中主要事件的命令及结果。例如，表空间的创建、重做日志的转换、操作系统的恢复、数据库的建立等信息都记录在警告日志中。警告日志是数据库每日管理的重要资源，当需要查找主要失败原因时，跟踪文件就非常有用。

应经常监控警告日志。警告日志的条目将通知你数据库操作期间遇到的任何问题，其中包括出现的任何ORA_0600内部错误。为使警告日志便于使用，最好是每天能自动对其重新命名。例如，如果警告日志称作alert_orcl.log，可以对它重新命名，以便其文件名包括当前日期。下次Oracle要写该警告日志时，将找不到具有 alert_orcl.log文件名的文件，因此数据库将创建一个新的文件名。这样，除了有以前的警告日志外，还有一个当前的警告日志 (alert_orcl.log)。用这种方式区分警告日志条目就可以使对警告日志条目的分析更有效。

1.8 基本数据库的实现

一个Oracle数据库的最简单形式由下列组件构成：

- 一个或多个数据文件。
- 一个或多个控制文件。
- 两个或多个联机重做日志文件。

数据库内部结构包括：

- 多用户/模式。
- 一个或多个回滚段。
- 一个或多个表空间。
- 数据字典表。
- 用户对象(表、索引、视图等)。

访问数据库的服务器的最小构成如下：

- 一个SGA(其中包括数据块缓存区、重做日志缓存区、 SQL共享池)。
- SMON后台进程。
- PMON后台进程。
- DBWR后台进程。
- LGWR后台进程。
- CKPT后台进程。
- 与PGA相关联的用户进程。

以上是Oracle的基本配置；其他内容是可选择的，或者取决于所使用的 Oracle版本及选项。

除了Oracle数据库的典型逻辑及物理设计外，本节后面还将对数据库的恢复及安全功能做一概要介绍。

1.8.1 备份/恢复功能

Oracle数据库提供一系列备份及恢复选项。第10章将对每一种选项进行详细介绍。下面几节介绍可利用的选项。

1. 导出/导入

导出(export)实用程序查询数据库并将其输出的内容存储在一个二进制文件中。你可以选择要导出的数据库部分。可以导出整个数据库、一个用户模式或用户模式集合、一个特定的表集合。导出还具有一些选项，这些选项允许只导出上一次导出 (称作增量导出)或上一次全系统导出(称作累积导出)后变化了的那些表。

全系统导出(full system export)读取全部数据字典表。由于数据字典跟踪用户、数据文件和数据库对象，所以可以用全系统导出来完整地重建一个数据库。全系统导出通常用于消除数据库中的碎片(见第8章)。

导出实用程序执行数据库的一个逻辑读。若要读取由导出创建的二进制转储文件的信息，必须使用导入(import)实用程序。导入实用程序将向数据库插入数据 (而不是重写已存在的记录)。

导出和导入是大多数数据库备份和恢复计划的一部分。由于导出是一次一个地方地读取数据，所以只能使用导出的数据把数据恢复到其当时所处的状态。因此对于不是最容易变动的数据备份，导出操作最有效。例如，当正在执行诸如把表传送到不同模式的数据库维护操作时，数据不应发生变化。对于这类操作，导出和导入是最有效的工具。对于频繁的事务处理环境，导出就不是最佳的备份方法，一般是采用联机备份。

2. 脱机备份

除了对数据库进行逻辑备份外，也能够对数据库文件进行物理备份。对数据库进行物理备份有两种方式可供选择：联机备份(online backup)和脱机备份(offline backup)。首先关掉数据库后再执行脱机备份，将构成数据库的文件备份到存储设备上(通过磁盘对磁盘的拷贝或磁带写入方式)。备份完成后，数据库就能够重新打开了。

尽管脱机备份方式不是主要采用的备份与恢复方式，但是周期性地进行脱机备仍然是一个好主意(例如，当其驻留的主机进行例行程序维护时)。

3.联机备份

在数据库以ARCHIVELOG模式运行时，可以对这些数据库进行联机备份。在数据库打开的状态下，可以用联机备份对数据库进行物理备份。在联机备份期间，先将表空间暂时置于一个备份状态，当文件备份完毕后，再将表空间恢复为正常状态。第10章将详细介绍联机备份与脱机备份的执行情况。

4.恢复管理器

对于Oracle8，可以使用RMAN(Recorvery Manager，恢复管理器)工具对数据库进行物理备份。RMAN可以对数据文件进行物理增量备份而不备份整个数据文件。当进行全部(level 0)数据文件备份时，数据文件中使用过的数据块都要进行备份。当进行累积(level 1)数据文件备份时，上一次全部数据文件备份后所使用的全部数据块都要进行备份。增量(level 2)数据文件备份只是备份那些上一次累积或全备份后发生变化的数据块。可以对增量备份所使用的级别进行定义。

通过一个恢复目录或者把需要的信息放置在被备份数据库的控制文件内，恢复管理器就可以对备份进行跟踪。通过init. ora文件的CONTROL_FILE_RECORD_KEEP_TIME参数来设置存储在控制文件中的RMAN记录的天数值。

如果数据库非常大且有隔离的事务操作区域，对数据文件执行增量和累积备份可大大提高备份的能力。有关实现RMAN和其他备份方法的细节，请参见第10章。

1.8.2 安全性能

Oracle中与安全有关的全部性能将在第9章中详细描述。本节只介绍Oracle安全性能的概要。

1. 帐号安全性

可以用口令来保护数据库帐号。也可以建立启用自动注册性能，使访问过一个主机帐号的用户不需要输入数据库口令就可以访问相关的数据库帐号。用户在一个数据库中拥有帐号及权限并不代表他在其他数据库中拥有帐号或权限。

2.系统级权限

可以从全部系统级权限集(例如CREATE TABLE、CREATE INDEX、SELECT ANY TABLE)中创建系统级角色以扩展系统级角色的基本集，CONNECT、RESOURCE及DBA就是分别提供给用户、开发者及数据库管理员的标准角色。

3.对象安全性

创建对象的用户可以通过grant命令把这些对象的权限授予其他用户。他们也可以授权一个用户带授权选项访问表。在这种情况下，这个用户(被授予者)可以授权其他用户访问这些表。

4. 审计

可以通过audit命令来审计涉及数据库对象的用户活动。审计操作包括表访问、注册企图及数据库管理员的特权操作。所有这些审计结果存储在数据库内的审计表中。除了提供的审计功能外，还可以创建数据库触发器来记录数据值的变化。

1.8.3 典型数据库逻辑设计

Oracle数据库的逻辑设计对数据库管理员所拥有的管理选项有很大的影响。基于本书第 3章的设计准则所设计的表空间如表 1-2所示。这种设计的目的是根据数据库段的用途和特性来隔离数据库段。

表1-2 数据库表空间的逻辑分布

表 空 间	使 用
SYSTEM	数据字典
DATA	标准操作表
DATA_2	标准操作时使用的静态表
INDEXES	标准操作表索引
INDEXES_2	静态表索引
RBS	标准操作回滚段
RBS_2	用于数据装入的特定回滚段
TEMP	标准操作临时段
TEMP_USER	由特定用户创建的临时段
TOOLS	RDBMS工具表
TOOLS_1	频繁使用的RDBMS工具表索引
USERS	用户对象，在开发数据库中
USERS_1	用户索引，在测试数据库中
SNAPS	快照表
SNAPS_1	快照表索引
AGG_DATA	聚集表和显形图
AGG_DATA_1	聚集表及显形图的索引
PARTITIONS	表或索引段的分区，为它们创建多个表空间
PARTITIONS_1	分区的局部及全局索引
TEMP_WORK	数据装入处理时使用的临时表

1.8.4 典型数据库物理设计

对数据库文件进行恰当地物理设计是数据库的一大特点；然而可应用一些常用规则来正确分开那些I/O请求将发生冲突的数据库文件。有关这方面的详细情况，请参见第 4章。表1-3所示的配置是12磁盘产品系统的典型配置。第 4章给出的过程将确定如何分布数据文件以满足需要。

表1-3 12磁盘数据库文件配置

磁 盘	内 容
1	ORACLE软件
2	SYSTEM表空间，控制文件1
3	RBS表空间，RBS_2表空间，控制文件2
4	DATA表空间，控制文件3

(续)

磁 盘	内 容
5	INDEXES表空间
6	TEMP表空间，TEMP_*USER*表空间
7	TOOLS表空间，INDEXES_2表空间
8	联机重做日志1、2和3
9	应用软件
10	DATA_2表空间
11	归档重做日志目标磁盘
12	导出转储文件目标磁盘

1.9 逻辑模型约定

在上一节介绍的表约束条件中，对一些数据模型的术语做了介绍。本节用图示方法来说明这些术语之间的关系。本节内容将有助于数据库管理员解释应用程序的数据模型（本书只采用其中的一部分）。

主键(PK，primary key)是确定表记录唯一性的列集或一个列。外键(FK，foreign key)是引用一个存在的主键的列集。

表之间可以通过三种关系类型相互关联：一对一、一对多和多对多。在一对一(1:1)关系中，这些表共享一个通用的主键；在一对多(1:M)关系中，一个表中的一个记录与另一表中的多个记录相关联；在多对多(M:M)关系中，一个表中多个记录与另一表中多个记录相关联。下面各节对每一种关系提供一个样例。

描绘不同关系类型的这些图示方式将贯穿全书使用。

1.9.1 一对一关系

除非出于性能或安全的原因而专门设计，否则两个表拥有同一个主键的情况很少见。例如，当一个表中用到LONG数据类型时，Oracle建议将该数据类型存储在一个单独的表中以提高系统性能，此时就采用1:1的模式使两个表相互关联。

现在考察本章前面图1-1的SALES_REPS表。如果在存储的数据中加上一个具有LONG数据类型的列RESUME，那么将会如何呢？既然每一个销售报表都有一个RESUME，当然应该将此列存储在SALES_REPS表中。不过，每当需要这个表时，都要迫使数据库读取这个LONG值；即便只是为了读取一个NAME字段也是如此。

为了改善执行性能，需创建另一个称为SALES_REPS_RESUME的表。这个表拥有同样的主键(REP_NUMBER)和一个附加的列(RESUME)。因此这两个表具有图1-6所示的1:1关系。

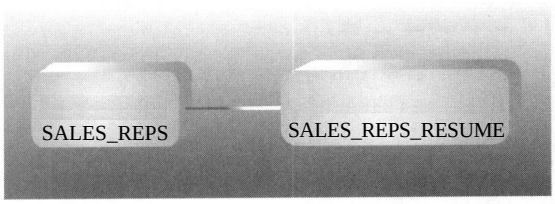

图1-6 一对一关系的实体关系图

两个实体间的实线表明两者间的关系是强制性的，如果关系是可选择的，则使用虚线。

Oracle8中的LOB存储区隐式采用1:1关系存储LOB数据。如果LOB数据大小超过一个阈值，它就与基表分开存储。

1.9.2 一对多关系

一对一关系极为稀少，而一对多 (1：M)关系却非常普遍。在一对多关系中，一个表中的一个记录与另一表中的多个记录相关联。

再来考察SALES_REPS表，图1-1给出的记录只考虑一个办公室只有一个销售报表。然而数据分析显示，同一个办公室拥有多个销售报表是完全可能的。为支持这种需要，应创建一个新的实体OFFICE。SALES_RFPS表中的office列是这个新表的一个外来关键字。

既然一个办公室(OFFICE表中的记录)能够同时拥有多个销售报表(SALES_REPS表中的记录)，那么表间就存在着一个一对多关系。图 1-7示出了这种一对多关系。注意在连线上有两点与图1-6不同：在关系的“多”方加上了一个爪形脚，并且用虚线与“一”方相连。虚线表明这些关联不是强制性的，一个办公室可能连一个销售报表也没有。

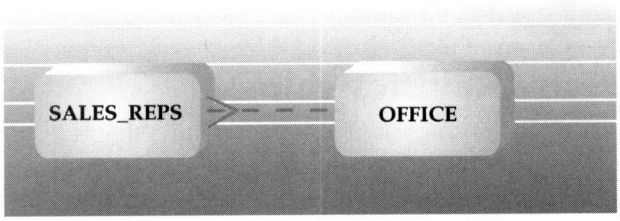

图1-7 一对多关系的实体关系图

1.9.3 多对多关系

有时可能需要一个表中的多个记录与另一表中的多个记录相关联。再来看一下图 1-1中的SALES_REPS表。对于这个例子，假设数据分析显示，一个销售报表包含多个公司。而且，一个公司可能被多份销售报表提及。因此，在 SALES_REPS实体和COMPANIES实体中存在一个多对多的关系。要了解这种关系，就要注意到一个销售报表 (SALES_REPS表中的记录)可能与多个公司 (COMPANIES表中的记录)有关，反之亦然。图 1-8示出了这种关系。

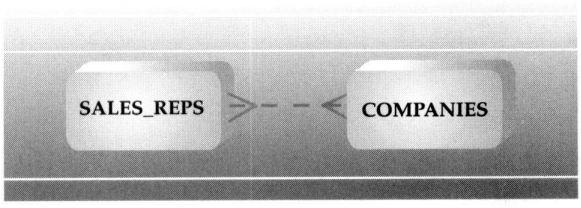

图1-8 多对多关系的实体关系模型

大多数多对多关系都有一个相交表 (intersection talbe)。例如，你可能希望记录每个销售代表与每个客户相互联系的信息。这个数据的主关键字应是 SALES_REPS表与COMPANIES表主关键字的组合。按照图示，相交表将位于 SALES_REPS表和COMPANIES表之间。

1.10 创建数据库

创建数据库时，应致力于表空间及文件在逻辑和物理分布上相互协调。并且应使用一致的脚本文件创建数据库。这样每个数据库都有一个类似的结构，从而简化了以后的数据库管理问题。

最简单的方法是通过Oracle安装程序来生成一个create database(创建数据库)脚本文件。安装Oracle时，安装程序将提供创建数据库的选项。如果使用这个选项，安装程序就创建一个小型数据库。该数据库并不重要，如果不打算使用它，就可以删除它的文件。 Oracle安装程序的重要产品是它所生成的创建数据库的脚本文件集。由安装程序生成的 create database脚本文件为你自己的create database脚本文件提供了一个模板。

1.10.1 修改创建模板的脚本文件

应对Oracle生成的create database脚本文件进行修改以适合自己的要求。对脚本文件的修改应包括以下内容：

- 在create database命令中，将MAXDATAFILES设置成其操作系统可使用的最大值。
- 在config.ora文件中，将DB_BLOCK_SIZE参数设置成操作系统支持的最高值。
- 在init.ora文件中，为 DB_BLOCK_BUFFERS、SHARED_POOL_SIZE、LOG_ARCHIVE_DEST_n、DB_WRITER_PROCESSES和希望自定义的其他init.ora参数设置恰当的数值。
- 在创建数据库的脚本文件的 create tablespace部分，实施所选择的逻辑和物理数据库设计。

除了DB_BLOCK_SIZE设置外，在创建数据库后还可以对其他全部设置进行更改。不过，在创建数据库之前更改则更方便一些。创建用于数据库创建的模板，并对其进行修改以适合创建的每一个新数据库。有关逻辑和物理表空间的分配准则，请参见第 3章和第4章。

1.10.2 创建数据库后修改MAXDATAFILES

MAXDATAFILES设置值作为create database命令的一部分被指定。 MAXDATAFILES为数据库中所能拥有的文件数量设置上限；它取代了 init.ora文件的DB_FILES参数的任何设置。如果需要增加新的数据文件 (例如，给数据库增加一个新的表空间)，但已达到MAXDATAFILES的极限，则文件分配就将失败。

若要更改MAXDATAFILES设置，就要重新创建控制文件。由于数据库打开时不能重新创建控制文件，因此应限制这种操作的次数。重新创建一个控制文件应遵循以下操作步骤：

1) 数据库打开时，进入Server Manager(服务器管理器)并执行下列命令：

```
alter database backup controlfile to trace;
```

这个命令将在用户转储文件的目标目录中生成一个跟踪文件。该跟踪文件含有可删除的头部信息，后面跟一个数据库的create controlfile命令。

2) 使用shutdown或shutdown immediate命令关闭实例。

3) 编辑第1步生成的create controlfile脚本文件。把MAXDATAFILES值设置成永远达不到的一个值。 MAXDATAFILES设置值越大，可创建的控制文件就越多。

4) 重新命名现有的控制文件。此时不要删除它们，因为要作为一个回退位置使用。

5) 通过Server Manager内的startup mount *instance_name* exclusive命令来安装数据库。

6) 运行第3步生成的已编辑的create controlfile脚本文件。使用init.ora或config.ora文件中的CONTROL_FILES设置值，Oracle将为数据库创建新的控制文件。

7) 可能需要使用alter database open命令的resetlogs选项打开数据库。

8) 一旦成功地打开数据库，就可以删除旧的控制文件。

有关shutdown和startup方面的问题

在重建控制文件过程的第 2步，关闭数据库。关闭数据库时，可以使用 shutdown、shutdown immediate或shutdown abort命令。如果使用shutdown abort命令关闭实例，Oracle在执行一个后续的alter database open 命令期间，需要在重新打开实例之前执行恢复操作。不应在执行一个数据库脱机备份之前使用 shutdown abort命令。

不过，有时可能要使用 shutdown abort命令。例如，如果在数据库应用程序中经常有死锁现象，如果使用一个 shutdown immediate命令，Oracle不能关闭数据库。如果需要在这样的环境中执行文件系统备份，应遵循以下操作步骤：

1) 执行一个shutdown abort命令。

2) 立即启动实例。

3) 执行一个shutdown命令。

现在，将以一致的方式关闭实例并准备备份操作。

1.10.3 使用OEM

Oracle7.3之后的版本提供了 Oracle Enterprise Manager(OEM，Oracle企业管理器)——图形用户界面(GUI)工具，这样数据库管理员用一台个人计算机就可以管理他们的数据库。随着Oracle8i的发布，2.0.4版本的OEM Toolset为远程数据库管理提供了相当健全的界面。在所有OEM的1.x版本中，一次只能有一个数据库管理员与 OEM资源库连接。因此，如果一个数据库管理员正在使用资源库时你也想管理一个数据库，就要么等待他完成任务后再开始进行，要么另外创建一个资源库。在这种配置中，数据库的变化有时可能会被另一数据库管理员重写。在OEM版本2的情况下，所有数据库管理员都可以使用同一个中央资源库执行其工作。除了这些变化外，OEM版本2还包含有任务调度和赋值功能，能连续不停地进行数据库覆盖。

对于OEM的所有版本，在安装和配置工具之前必须做出一些重要决定。要确定在什么地方创建OEM资源库，何时并怎样执行备份以保护这个资源库。由于可以把 OEM用作Oracle Recovery Manager(RMAN，恢复管理器)的界面，所以能够把恢复信息存储在 OEM资源库中。尽管没有任何东西阻止你在产品数据库中创建 OEM资源库，但是如果丢失这个数据库的话，将在何处获取恢复信息呢？因此，可能要单独创建一个存储 OEM资源库的小型数据库。应确保这个资源库经常备份，以便保证能恢复这个资源库。

如果你是唯一与OEM工具集打交道的数据库管理员，就不必考虑谁将管理指定的数据库。然而，如果这个站点有若干个数据库管理员，就要确定任务定义、数据库响应性和调度。在使用OEM的情况下，可以根据任务情况把一些访问级别和权限授予组中的每个数据库管理员。可以对OEM进行配置，以便能把电子邮件请求和赋值发送给其他数据库管理员，或者使问题尽快得到解决。

如果系统上有以前版本的OEM，就要把那个资源库移植到最新版本的OEM上，以便利用其最新的性能。如果系统上有不止一个资源库，就要采取预防措施，以确保在移植每一个版本的资源库时不破坏当前存储的信息。

Oracle支持SNMP(Simple Network Management Protocol，简单网络管理协议)。通过支持SNMP，就可以把Oracle产品轻易地集成到系统和网络的监控工具中。由于有了SNMP的支持，Oracle可以用企业网络上的现有工具进行监控。SNMP的主要优点如下：

- 容易集成到由SNMP工具管理的企业网络中。
- 对数据库访问所需要的全部服务进行集中监控。
- 支持对紧急情况自动报警。
- 支持对报警条件自动做出反应。

尽管不必使用OEM来管理数据库，但它对数据库提供一个通用界面。当企业在规模上(在数据库和数据库管理员的数量方面)增大时，数据库管理人员界面的一致性将支持变化控制和产品控制进程实现的一致性。