

## 第二部分 Oracle 7.x RDBMS

### 第5章 Oracle数据库体系结构

本章要点：

- 定义数据库

- SYS和SYSTEM的模式

- 数据库组件

- 数据库段

- Oracle数据字典

- 其他数据库对象

#### 5.1 定义数据库

“数据库”这个术语既可以作为整个数据库管理环境的名字，也可以（在 Oracle中）用来描述建立关系型数据库管理系统（RDBMS）的逻辑的和物理的数据结构。作为一名 Oracle专家，你可以把Oracle数据库定义为共同组成数据处理环境的配置文件、数据文件、控制文件以及重做日志文件；表、索引和其他包含在这些对象中的结构。

#### 5.2 SYS和SYSTEM的模式

SYS和SYSTEM是每个Oracle数据库都缺省安装的两个帐户。SYS模式是所有内部数据库表、结构、供给包、过程等等的拥有者，它还拥有所有的 V\$和数据字典视图，并创建所有封装的数据库角色（DBA、CONNECT、RESOURCE等等）。SYS是一个Oracle数据库的根用户或系统管理员；由于它具有全能的性质，你应尽量避免作为 SYS注册到系统中工作。当你以SYS帐户注册到系统中工作，哪怕一个简单的打字错误，就有可能造成毁灭性的灾难。

SYS帐户是唯一能够访问特定内部数据字典表的用户，因为它拥有所有的数据字典结构，为了将数据字典对象明确地授权给其他模式，SYS 也是你必须登录的帐户。当你使用数据字典视图和表编写存储过程或触发器时，也必需使用 SYS帐户。当数据库首次安装时，SYS帐户的缺省口令是CHANGE\_ON\_INSTALL，并且每个称职的数据库系统管理员（他或她）都会立即更改这个口令。

SYSTEM模式也是在数据库创建时安装的，是用于 DBA 任务的缺省帐户。SYSTEM也对所有的数据库对象拥有完全的权限，而且许多第三方工具软件依赖于 SYSTEM模式的存在及特权。SYSTEM 帐户的缺省口令是MANAGER，并且像SYS帐户的口令一样，在数据库创建后应该立即被更改。许多数据库系统管理员使用 SYSTEM模式执行数据库管理任务，但它更适于创建一个特殊的用户来完成数据库管理员的任务，这就确保了一个特殊的帐户与一个特定的人相连，而那个特定的用户则对全部数据库的修改负有责任。

因为这些模式都被大家所熟知，并且存在于每个 Oracle 数据库中，在安装完数据库后立即更改它们的缺省口令是很重要的，这样可以防止对帐户未经授权的访问。如果安全性是个主要的问题，你或许还应该考虑将这些帐户变为不可登录，只在需要登录它们时，才设置合法的口令。

注意 通过发出 ALTER USER  $\times \times \times$  IDENTIFIED BY VALUES '口令'命令，可以禁止对一个帐户的注册；这里'口令'是任何小写字字符串。这就将贮存的口令设置为你给出的实际的值，而不是加密的方式，在加密方式中，Oracle 用一个普通的 ALTER USER  $\times \times \times$  IDENTIFIED BY '口令'命令来存储口令。对 Oracle 来说，产生一个小写加密口令字符串是不可能的，从而使得对该帐户的登录变为不可能。

## 5.3 数据库组件

你可以把数据库对象划分成两种不同的类型：一类是由 RDBMS 内部使用的对象，被称为系统数据库对象（system database object），另一类是可以通过任何程序访问的对象，被称为用户数据库对象（user database object）。

### 5.3.1 系统数据库对象

当提到系统数据库对象时，是指 RDBMS 用于支持内部数据库功能的数据库对象。这些对象是由数据库系统管理员或服务器本身配置和创建的，并且不显式地用于用户数据库事务。

系统数据库对象如下所示：

初始化参数文件。

控制文件。

联机 and 归档重做日志文件。

追踪文件。

ROWID（行内部地址）。

Oracle 块。

下面是对这些对象的逐一解释：

#### 1. 初始化参数文件

初始化参数文件（initialization parameter file）或 init.ora 是 RDBMS 主要的配置点，它是配置键码和数值的集合，每一个配置键码和数值都控制或修改数据库和实例操作的某个方面。它是一个 ASCII 文本文件，可以在 UNIX 服务器上的 \$ORACLE\_HOME/dbs 目录下和 NT 服务器上的 \$ORACLE\_HOME/database 下找到它。默认条件下，文件的名字为 initSID.ora，在这里，SID 相当于它所控制的数据库的标识符。在一个 UNIX 服务器上，如果在命令行上没有显式地指定一个 init.ora 文件，当启动一个数据库时，这就是 Oracle 服务器所要寻找的文件名（这里 SID 等于 \$ORACLE\_SID 环境变量的值）。每个 Oracle 数据库和实例都有它自己唯一的 init.ora 文件。

init.ora 文件可以包含来自其他使用 IFILE 参数的文件的配置值。在 UNIX 环境中，通常将 \$ORACLE\_HOME/dbs/init.ora 文件与其他位置的文件相连，以便使数据库环境安装有更好的控制和结构。非正式的参数（主要是由 Oracle Worldwide Customer Support 使用）使用前缀下划线来命名。

当数据库启动时，在创建实例或读取控制文件之前，先读取 init.ora 文件。init.ora 文件中的值决定着数据库和实例的特性，例如共享池、高速缓存、重做日志缓存分配、后台进程的自动启动、控制文件的读取、自动联机回滚段等等。直到数据库被关闭并重新启动，对 init.ora 文件中参数的更改才被承认。

在 Oracle RDBMS 中，缺省的 init.ora 文件位于 \$ORACLE\_HOME/dbs 目录下，带有针对小、中、大型数据库所预先设置的基本 init.ora 参数和不同的推荐（任意的）值。当你创建新数据库和实例时，这个文件可被复制和重新命名。

通过查询 V\$PARAMETER 视图，可以从数据库内部观察 init.ora 文件中的配置参数集。V\$PARAMETER 视图把所有的 init.ora 参数和它们的值都列出来，并且每一个值都有一个标记符，用以指明参数值是否为服务器默认值。

在表 5-1 中，解释说明了包含在缺省 init.ora 文件中的参数。要得到一个更为全面的清单，参见服务器文档集中的 Oracle Server Reference manual（Oracle 服务器参考手册）。

表 5-1 常用的 init.ora 参数

参数名称	用 途
audit_trail	允许或禁止写记录到审计追踪文件。注意，这里只是允许审计，审计的动作必须分别配置
background_dump_dest	Oracle 后台进程追踪文件的最终目录，包括 alert.log
compatible	数据库的兼容级。能够防止使用比所使用数据库版本参数值高的数据库特性
control_files	数据库的控制文件
db_block_buffers	包含在高速缓存中的数据库块数目。db_block_buffers × db_block_size = 数据库高速缓存的大小，单位为字节
db_block_size	Oracle 数据库块的大小。在数据库建立起来后，这个值就不能改变了
db_files	能够打开的数据库文件的最大数目
db_name	可选择的数据库的名字。如果使用该参数，它必须与用在 CREATE DATABASE 语句中的数据库名称相一致
db_file_multiblock_read_count	一次 I/O 操作所能读取数据库块的最大数。这是用于顺序搜索的，当调整全表搜索时，是非常重要的
dml_locks	由所有数据库用户用于全部表的 DML 锁的最大数目
log_archive_dest	归档重做日志文件的最终位置
log_archive_start	允许或禁止自动归档。如果允许，当实例启动时，ARCH 进程自动地启动
log_buffer	分配给重做日志缓冲区的字节数
log_checkpoint_interval	触发一个检测点需要填充的重做日志文件块数
max_dump_file_size	Oracle 追踪文件操作系统块的最大尺寸
processes	能与数据库连接的操作系统进程的最大数目，其中包括后台进程。当在 UNIX 服务器上调整共享内存时，这是很重要的
remote_login_passwordfile	确定一个口令文件是否用于远程内部验证，并指定有多少数据库可以使用一个口令文件。它可以被设置为 NONE、SHARED 和 EXCLUSIVE
rollback_segments	在数据库启动时，自动联机回滚段的列表清单
sequence_cache_entries	能够缓存在系统全局区中的序列数。它应当被设置成实例中随时能用的最大序列数
shared_pool_size	共享池的大小，用字节表示
snapshot_refresh_processes	在实例启动时，启动的 SNP 进程数目。SNP 进程负责刷新快照，以及执行由 DBMS_JOB 提交的数据库工作
timed_statistics	允许或禁止收集数据库的实时统计信息。虽然把该参数设置成为真会导致性能开销，但是在数据库调整时有更大的灵活性
user_dump_dest	用户追踪文件的最终目录，包括通过设置 sql_trace 为真，所产生的那些文件

## 2. 控制文件

控制文件是数据库的心脏，它包含以下信息：属于数据库的数据文件和重做日志文件信息、数据库中的数据应该以何种字符集存储的信息、数据库中每个数据文件的状态和版本信息、以及其他的重要信息。包含在控制文件中的大部分参数是在数据库创建过程中设定的，并且相对来说是静态的；也就是说，它们不是经常改变的。控制文件采用二进制格式，并且是不可读或手工编辑的。

控制文件是在数据库创建时创建的。大多数数据库可以操作多个控制文件。特定控制文件的创建是在init.ora参数CONTROL\_FILES中指定的，在CREATE DATABASE子句中指定的数据库创建参数存储在这些文件中。

如果没有正确的控制文件，数据库不能被打开。如果由于某种原因，导致控制文件无效或损坏，数据库将无法启动，并且存储在数据库中的数据信息将无法访问。正是由于这个原因，控制文件的镜像备份功能是被 Oracle 服务器内部支持的，并且也是大力推荐的。如果要将控制文件镜像备份到一个新的数据库中，只需要在发出 CREATE DATABASE 命令之前，为 CONTROL\_FILES 指定一个以上的参数值即可。要将控制文件镜像备份到一个现有数据库中，你必须关闭数据库，将当前的控制文件拷贝到你想要备份的目录当中，编辑 CONTROL\_FILES 参数，以指定新的控制文件的位置，然后启动数据库。

注意 根据经验得来的一个有用的法则是：在四个不同的物理磁盘上，存储至少四份控制文件的备份。

不幸的是，修改控制文件参数不像改变初始化参数并启动数据库那么容易。为了改变任何控制文件参数，你必须重新创建控制文件。重新创建控制文件的步骤如下：

1) 备份数据库。修改控制文件过程中，出现的任何一个错误都会损坏你的数据库，使之无法恢复。没有一个有效的数据库备份，千万不要进行这项操作。

2) 发出 ALTER DATABASE BACKUP CONTROLFILE TO TRACE；命令，它是来自服务器管理器或 SQL\*Plus 的命令。这就建立了一个用户追踪文件（位于 USER\_DUMP\_DEST），该文件带有重建当前控制文件所必须的命令。

3) 编辑在前面步骤中所产生的追踪文件。除了 CREATE CONTROLFILE 语句外，删除追踪文件中的所有行，设置新的参数值。

4) 正常关闭（SHUTDOWN NORMAL）数据库。将旧的控制文件移放到备份目录里，确保 Oracle 在数据库启动时，不能在目录中找到控制文件的任何副本，否则下一步操作将会失败。

5) 执行 STARTUP NOMOUNT 命令启动数据库，运行你所编辑的 CREATE CONTROLFILE 追踪文件，这将重建带有新参数值的控制文件。

6) 执行 ALTER DATABASE OPEN；命令。

在数据库创建时，把数据库参数值设为比你所需要的值高一些，可以避免重建你的控制文件。将数据库参数值设为比需要的高一些，带来的唯一负面影响是在内存和磁盘空间方面微不足道的浪费。

警告 当你创建数据库时，正确地设定 CHARACTERSET 参数是很重要的，改变这个参数需要重新创建整个数据库，不能通过重建控制文件改变它。

可配置的系统控制文件参数见表 5-2

表5-2 包含在控制文件中的可修改的配置参数

参数名称	说 明
MAXLOGFILES	联机重做日志文件的最大数
MAXLOGMEMBERS	每个重做日志文件的最大成员数
MAXDATAFILES	数据文件的最大数
MAXINSTANCES	能够安装到这个数据库上的最大实例数（并行服务器）
MAXLOGHISTORY	用于恢复实例的归档重做日志文件组的最大数（并行服务器）

注意 要改变数据库名，要像前面所描述的那样重新创建控制文件。但是，要在追踪文件中把REUSE DATABASE <老名字>行变为SET DATABASE <新名字>。

V\$CONTROLFILE 视图列出了 Oracle 服务器当前正在读写的控制文件。

提示 你应该经常通过执行语句 ALTER DATABASE BACKUP CONTROLFILE TO TRACE，保持控制文件的一个文本备份（在保持通常的二进制副本的同时），当所有其他的手段都失败的时候，它可以帮助你重新创建数据库。

### 3. 联机重做日志文件

日志写后台进程（LGWR）将重做日志缓冲区的内容写入联机重做日志文件中，重做日志贮存了数据库的所有变化信息，并在数据库的恢复期间被 Oracle 使用。正如图 5-1 所示，联机重做日志文件至少由两组重做日志文件组成，并且按照一种循环的特性写入。

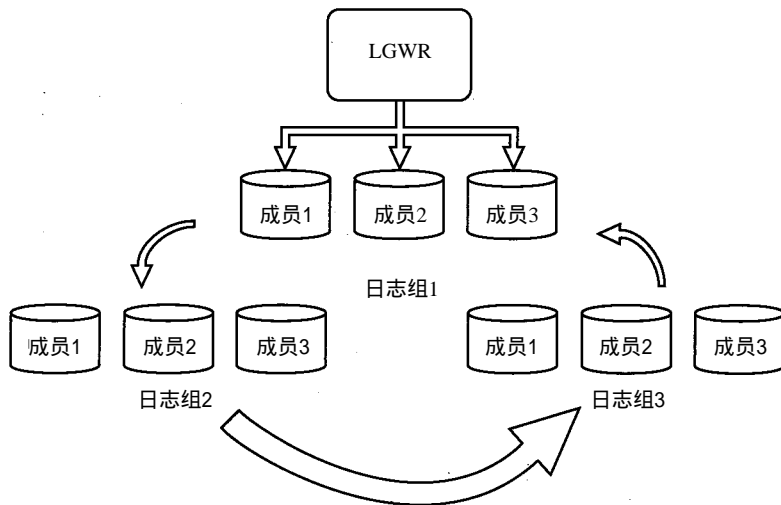


图5-1 带有多个成员的重做日志组

一个重做日志组如果当前正被 LGWR 写入，那么它是当前的。当当前日志组被填满，并且 LGWR 进程停止对它的写入而转到下一个日志组的时候，就发生日志切换。当日志切换发生并且归档日志模式被数据库允许的时候，前面写入的重做日志组被归档（ARCH）进程锁定，并被拷贝到磁盘或磁带上，具体拷贝方向取决于系统配置。如果 LGWR 追上 ARCH 进程，并且需要实时写入由 ARCH 正在写入的组，所有的数据库活动将被挂起，直到 ARCH 进程写完日志。如果你在 alert.log 文件中看到错误说明一个自由的日志组不能被 LGWR 进程锁定，那么说

明你需要添加更多的重做日志组或者调整日志组的大小。

每个日志组可由多个成员组成，日志组的每个成员是其他成员的精确镜像，并且重做日志项被并行地写入每个成员中。如果 LGWR 不能写入组里的成员，它也不失败；相反，它在 alert.log 文件中写入一个数据项。通过在每个组中使用多个成员，你能防止数据库由于丢失重做日志而导致失效。只要组里有一个成员是可用的，数据库就能继续工作。

注意 通过使用 RAID1（镜像）卷镜像备份重做日志组，可以获得同样的功能。当 LGWR 进程必须为每个数据库事务更新多个日志组成员时，这将缓解上述情况造成的系统开销。

V\$LOG 和 V\$LOGFILE 视图保存联机重做日志文件的信息，下列的查询检查当前日志的状态：

```
SELECT b.member, a.bytes, a.members, a.status
FROM v$log a, v$logfile b
WHERE a.group# = b.group#
ORDER BY b.member;
```

#### 4. 追踪文件

所有的 Oracle 数据库都至少有一个文件用于记录系统信息、错误及主要事件。这个文件，叫做 sidALRT.log（这里 sid 是数据库系统的标识符），存储在由 init.ora 参数 BACKGROUND\_DUMP\_DEST 指定的位置。当调查数据库故障时，这是你应该查看的第一个位置。关键的错误总是在这里被记载，如数据库的启动和关闭信息、日志切换信息以及其他的事件。

后台进程和用户进程也建立它们自己的追踪文件，在其中记载出现问题和故障的信息。后台进程追踪文件贮存在 BACKGROUND\_DUMP\_DEST 所指定的位置，而用户进程追踪文件贮存在由 USER\_DUMP\_DEST 参数设置所指定的目录里。将 USER\_DUMP\_DEST 目录设定为不同于由 BACKGROUND\_DUMP\_DEST 所指定的目录，使得你能够跟踪不同类型的追踪文件。后台进程追踪文件被命名为 sidPROC.trc，这里 sid 是数据库的系统标识符，PROC 是后台进程（DBWR、LGWR、SMON、PMON 等等）的名字。

用户会话追踪文件带有一个 ora 前缀，后面是一系列带有 .trc 文件扩展名的唯一标识数字。当一个用户会话引起无法恢复的问题（例如死锁或一个服务器进程崩溃）或用户会话被显式地告知创建跟踪文件时（例如当 SQL 跟踪被允许，或者发出一个 ALTER DATABASE BACKUP CONTROLFILE TO TRACE 命令），用户会话追踪文件就被生成。如果要允许 SQL 追踪，在 SQL\*Plus 提示符状态下发出 ALTER SESSION SET SQL\_TRACE=TRUE 命令，或把 init.ora 参数 SQL\_TRACE 设置为 TRUE。但是注意，在 init.ora 中将 SQL\_TRACE 设为 TRUE，将会导致把所有与数据库不一致的 SQL 语句写进追踪文件中，这将产生大量的追踪信息。

可以在 V\$PARAMETER 视图中获得 BACKGROUND\_DUMP\_DEST 和 USER\_DUMP\_DEST 参数的当前设置。

#### 5. ROWID

为了提取信息，Oracle 数据库必须能够唯一识别数据库中的每一行。Oracle RDBMS 中用于这种任务的内部结构叫做 ROWID（行内部地址），这是存储数据库中每条记录行的物理位置的两字节数值。ROWID 的格式如下：

```
BBBBBBBB.RRRR.FFFF
```

BBBBBBBB是数据文件中行所在的块号（采用16进制）。RRRR是数据行所在块中的行号（采用16进制），FFFF是块所在的文件号（采用16进制）。

例如：一个表中的一行可能有如下所示的 ROWID 形式：

0000068C.0000.0001

这个 ROWID 是在第一个数据文件（0001）的 68C（16进制）块中，并且是该块的第一行（0000）。

注意 通过查询 DBA\_DATA\_FILES 视图，你可以把前面 ROWID 中的文件号和文件名相匹配。在 Oracle7 中，这些文件号是固定的；而在 Oracle8 中，它们是在数据库启动时决定的。

当每行在第一次创建时，都会被分配一个 ROWID。除非该行被删除或该行所在的段被重组（通过导入/导出、第三方厂商重组工具等等），否则这个 ROWID 保持不变。在数据库中使用 ROWID 是找到一行的最快方法。数据库的每个表都有一个名为 ROWID 的伪列（pseudocolumn），通过查询它可以列出表中每一行的 ROWID。

因为 ROWID 的唯一性，它可以被用来创造性地解决许多不同的难题。例如：ROWID 的一个最常用用途是在数据库中识别那些具有相同值的列。下面的 SQL 脚本将显示这是如何实现的：

```
delete from duplicate_table
where ROWID not in (select MIN (ROWID) from duplicate_table
                    group by a1, a2);
```

在查询时，你也可以使用 ROWID 的实际值，下面的 SQL 脚本显示一个在其中含有行的表的数据库文件数目：

```
SELECT COUNT(DISTINCT(SUBSTR(ROWID, 15, 14))) "Files"
FROM test_table;
```

## 6. Oracle 块

你能操作的最低级的数据库存储单元就是 Oracle 块，这是服务器能够访问的最小存储单元。不应该把它与操作系统块混淆，一个 Oracle 块是由操作系统块组成的，它们并不相同。

提示 Oracle 块是操作系统块的整数倍。例如：在 UNIX 中，操作系统块通常为 8KB，那么你应该将 db\_block\_size 设置为 8192、16384 等等，使它成为 8KB 的整数倍。

所有的数据访问都是以 Oracle 块的形式实现的。Oracle 块的大小是指一次 I/O 过程中，RDBMS 从数据文件中读写的字节数、数据库对象大小和高速缓存中的块也是以 Oracle 块的形式设置的（尽管有些视图用字节显示存储的大小，但这只是为了增加可读性）。

注意 Oracle 块的大小是在数据库创建时为数据库设置的，是不能改变的。如果在创建数据库后，你认为需要大一些的（或小些的）块，那么你不得不从头开始，重新建立整个数据库。

每个 Oracle 块都包含头信息空间、块中数据将来的更新空间以及块中实际存储行所占空间。块头保存着这样的信息，例如在块中有行存在的数据库段、同时可以有多少事务访问块等等。每个块还分配了一定量的空间，用于块中存储行的将来更新，如果升级引起原先的行变大，那么这个自由空间就被使用了。

一个块可以用于接受新行的能力是通过 PCTFREE和PCTUSED存储设置来控制的，PCTFREE参数负责分配块空间的百分比，这些空间被保留在一旁，用于数据更新。例如，如果一个块有一个 30%的PCTFREE值，那么该块 70%的空间是用于存储新行的，当该块 70%的块已经被填充时，Oracle把该块从自由列表中取出，利用剩下 30%的空间来处理要求更多的空间的块中行的更新。

在块被重新放回自由列表之前，PCTUSED参数指定块中必须有多少自由空间（通过删除块中的行，或更新减少贮存一行所需空间）。

PCTFREE参数和PCTUSED参数一起使用，确保一个块有足够的空间来满足将来的存贮需要，而且确保了块不会反复摆动，换句话说，不会花费相当多的时间从自由列表进进出出。PCTFREE和PCTUSED的值不应该等于 100%，否则参数值就不起作用了，参见第 21 章“管理数据库存储”，可以得到关于调整Oracle块的更多信息。

因为数据库块决定着在每一次 I/O 操作中能从数据文件中读取的字节数，所以它的大小是非常重要的调整因素。联机事务处理（Online Transaction Processing，OLTP）应用有时可以从较小的块（4KB或8KB）上获益，因为这些应用在每个事务期间，通常要读写少量的数据，因此较小的块在 I/O 性能方面更为有效。

对于仓库式的或决策支持系统（Decision Support System，DSS）数据库来说，较大的数据库块能够极大地提高性能。这些类型的应用在一个事务过程中，常常需要处理大量的数据，并且通常考虑系统的响应时间以分钟来计时，而不以秒来计时。

通过设置数据库块的大小，你可以调整一次 I/O 操作所读入内存缓冲中的数据量。当决定数据库块的最佳大小时，你必须记住你的应用将要执行的特定类型的数据存储，以及你希望贮存在每一行中的数据量。一个过大的数据库块将会把一些不必要的数据读进缓冲区，而一个太小的块则会导致行链接。

### 5.3.2 用户数据库对象

用户数据库对象（User database objects）是指那些不是 Oracle RDBMS 专用的对象。当然，它们是受 Oracle 内部管理的，但是它们能够提供一组用户建造块，利用这些块用户可以创建自己的数据库。用户数据库对象包括数据文件、区间、表空间以及数据库段。本章下面部分对这些术语做了一一解释。

#### 1. 数据文件

Oracle 数据文件作为操作系统文件而存在，每个数据文件被分配给一个表空间，并且拥有存储在那个表空间里的实际数据。数据文件是文件系统中的实际文件，它可以像其他任何操作系统文件一样进行监控和操作。存储在数据文件中的数据采用 Oracle 二进制格式，这样除了 Oracle RDBMS 外，不能被其他任何东西读取。

数据文件是使用 SQL 命令 CREATE TABLESPACE 或 ALTER TABLESPACE 创建的。一个数据文件的大小是根据 CREATE 语句中指定的大小来建立的，而不是由它所贮存的数据量决定的，例如：一个以 10MB 大小创建的数据文件，不管它包含一行或一百万行，它都用满 10MB 的空间。

从 Oracle 7.3 开始，数据库管理员具备动态增加和缩减 Oracle 数据文件的能力。然而，数据文件不能被收缩得小于数据文件的高水位线。数据文件也能被单个脱机，以进行备份或其

他的数据库操作。

你可以访问 DBA\_DATA\_FILES和V\$DATAFILE 视图，来获得数据文件中为数据库定义的有关信息。下列查询展示了表空间与数据文件之间的映射：

```
SELECT tablespace_name, file_name, bytes
FROM dba_data_files
ORDER BY tablespace_name, file_name;
```

## 2. 区间

区间（extent）是一个存储单位，由一个或多个逻辑（就是说，在 Oracle内）连续的 Oracle块组成，每个数据库段都是由一个或多个区间组成，在一个数据库段中的每个区间的大小可以相同或不同。从 Oracle 7.3开始，一个数据库对象所能拥有的区间最大数取决于 Oracle块的大小。

一个数据库段在对象创建时，完全按照 CREATE命令的存储子句中所指定的来分配区间。当一个段再也无法把新数据填充到它当前分配的区间中时，必须为它分配其他的区间。下一个分配区间的大小决定于好几个因素，有些因素并不是显而易见的。新区间的分配管理是数据库空间管理主要涉及的内容。参阅第 21章“管理数据库存储”，可以得到更多信息。

区间信息贮存在DBA\_EXTENTS视图中。

## 3. 表空间

表空间（tablespace）是一个数据结构，用于组合被相似地访问的数据。每个表空间都是由一个或多个数据文件组成的，所有的数据库对象必须被指定一个表空间，它们在那里被创建。于是组成对象的数据就被贮存到分配给指定表空间的数据文件中。

表空间被用于分隔涉及数据访问的 I/O，例如：可以创建一个表空间来存储数据对象，也可以创建另外一个表空间来存储索引对象。通过给驻留在不同物理磁盘上的表空间分配数据文件，你可以确保对索引数据的访问不会影响对索引指向的数据的访问。参阅第 19章“调整输入输出”，可以得到更多信息。

在数据库备份和恢复中，表空间也扮演了一个重要的角色。因为一个表空间直接映射着一个或多个数据文件，备份和恢复数据通常是在表空间（数据文件）级下进行的。当然，备份或恢复操作是应用于整个数据库时是例外的。

你可以在DBA\_TABLESPACES查看表空间的信息。

## 5.4 数据库段

数据库段（database segment）是贮存在数据库中的用户建立的对象。这在很大程度上包括了组成模式的数据（表）和索引。

除了用户创建的数据和索引段外，通常还有两种类型的段被称为系统段或管理员创建段，它们是临时段和回滚段。当然，那些拥有合法权限的用户可以创建这些段。但是，最好由数据库系统管理员来创建，然后让应用用户和程序共享这些段。

### 5.4.1 表

表（table）是存储数据的数据库段。每个表是由一个或多个列组成的，每个列都被指定一个名字和数据类型。每个列的数据类型为贮存在表中的数据定义了类型和精度，合法的 Oracle列数据类型列于表 5-3中。

表5-3 有效的Oracle 7.x 列数据类型

数据类型	说 明	最大长度
CHAR	固定长度的字符域，尾部使用空格填充	255 字节
VARCHAR	变长字符域	2KB
VARCHAR2	变长字符域	2KB
LONG	变长字符数据	2GB
NUMBER	变长数字数据	$1 \times 10^{130} \sim 9.99 \times 10^{125}$
DATE	固定长度的日期和时间域，从公元前到公元后 4712年12月31日	4712年1月1日
RAW	变长原始二进制数据	255字节
LONG RAW	变长原始二进制数据	2GB
ROWID	行内部地址变量类型	6字节

注意 为预防ANSI规定的改变，Oracle建议所有的变长字符域定义为VARCHAR2，而不是VARCHAR。Oracle保证永远不对VARCHAR2做出导致需要对应用程序修改才能使它们向上兼容的功能性变化。因为VARCHAR的功能是由ANSI标准委员会授权管理的，Oracle不能保证这些功能在新版本里不发生巨大的变化。

DBA\_TABLES和DBA\_TAB\_COLUMNS包含数据库中关于表的信息。

#### 5.4.2 索引

索引（indexe）是为了加速对特定表数据的访问而创建的数据段。一个索引拥有表的一列或多列的值以及与这些列值相对应的行内部地址（ROWID）。当Oracle服务器需要在表中查找某一指定行时，它在索引中查找ROWID，然后直接从表中提出数据。

在Oracle RDBMS中有几种可用的索引类型。到目前为止，最常用的索引类型是 B\*-Tree（B-树）索引。这是执行标准的 CREATE INDEX 语句时，需要使用的索引类型。B\*-Tree索引是标准的搜索树算法的一个变异，其中通过遍历索引树，你保证能在相同数量的树遍历过程中找到所有的叶子结点。每一个叶子结点指向下一个和前一个叶子结点的位置，这样就能在索引扫描范围内进行快速索引搜索及类似操作。B\*-Tree索引能确保维持平衡状态，并且每个结点的四分之三都是空的，以便为更新提供可用空间。

对B\*-Tree索引结构的深层解释不属于本书范围。但是，你可以在大部分现代数据库教科书中看到有关该主题的大量信息，例如 Elmasri和Navathe的Fundamentals of Database Systems。

簇（cluster）索引是在簇中被表共享的列的索引。在本章稍后会解释簇表。不同于常规的索引，簇索引在索引中只存储一次索引键值，而不管索引键在表中重复多少次。在簇上能够执行任何数据操作语言（DML）前，必须在簇上创建簇索引。

最新的索引类型是位映射（bitmap）索引，在一个位映射索引中，位映射是从索引表的列值中创建而来的，并存储在索引中，而不是由实际的列值创建的。换句话说，在键上索引为每一行保存一个位映射，键对于表中的每一行包含一位信息。如果值包含在行里，位值是 1，否则是 0。在基数较低（不同值数量较小）的列中，位映射索引可以比传统的 B\*-Tree索引更小，更有效。表5-4是一个关于汽车表的位映射例子，位映射的键是表中的颜色列。

在这个例子中，表中共有 20 行。汽车表中行与颜色相匹配处，位映射将为 1，即：第 4、7、11、15 和 20 行是红色汽车，而第 2、6、和 12 行是黑色汽车。位映射索引结构创造了比传统

B\*-Tree索引更小的索引，但正如讨论的那样，位映射索引只适用于特定类型数据。

DBA\_INDEXES 和 DBA\_IND\_COLUMNS 含有关于数据库中全部索引的信息。

表5-4 汽车表位映射索引

汽车颜色	位 映 射
红色	00010010001000100001
绿色	10000001010001001010
银色	00100000100010010000
白色	00001000000000000100
黑色	01000100000100000000

### 5.4.3 回滚段

回滚段 (rollback segment) 是存储在数据库事务中发生改变的原始数据块的数据库对象。它们用于提供数据的已经改变但尚未提交的读一致性视图。当做出数据改变时，原始数据就被拷到回滚段中，而且在内存缓冲区中对数据块做出更改。如果其他用户会话要求同样的数据，那么存储在回滚段中的原始数据就会被返回（这就叫读一致性）。当做出更改的会话被提交时，回滚段项将被标明为无效。

多个用户会话可以共享一个回滚段，每个回滚段至少由两个区间组成。当一个事务开始时，用户会话在一个可用回滚段中得到一个可用区间的专用锁，于是事务信息被写入回滚段中，如果事务填满了第一个区间，它会被分配另一个区间。如果另一个区间不可用，回滚段会自动地给自己分配其他的区间，这正是用户会话所要获取的，这叫回滚段扩展 (rollback segment extension)。因为区间的分配影响性能，因此，你的目标应该是在没有分配新区间的情况下，使所有事务能够执行。

如果回滚段不能分配其他的区间（或许是因为已经达到回滚段的最大区间数，或者在回滚段表空间上没有更多的自由区间），那么就会出现错误，而且事务将被回滚。这通常发生在装载大量数据时，在那时联机回滚段无法为事务提供足够的空间来存储全部的回滚信息。

有关创建和管理回滚段的详细信息，参见第 21 章。

DBA\_ROLLBACK\_SEGS 视图中含有关于回滚段的信息。

### 5.4.4 表簇

表簇 (table cluster) 是一个数据库对象，它可以将那些经常在相同数据块中一起使用的表进行物理分组。当你处理那些经常连接在一起进行查询的表时，表簇是特别有效的。一个表簇存储簇键（用于将表连接到一起的列），以及簇表中的列值。因为簇中的表都被贮存在相同的数据库块中，所以使用簇工作时，I/O 操作就减少了。

### 5.4.5 哈希簇

哈希簇 (hash cluster) 是数据库存储的最后选项。在一个哈希簇中，表是基于哈希值组织的，在表的主键值上使用哈希函数可以得到这个哈希值。在从哈希簇中提取数据时，哈希函数被用于要求的键值上，结果值给出 Oracle 哈希簇中的存储数据的块。

使用哈希簇能明显减少从表中提取数据行的 I/O 操作，但是使用哈希簇也有一些缺点。有

关创建和管理哈希簇的详细信息参见第 21 章。

## 5.5 Oracle数据字典

数据字典 (data dictionary) 是存储在数据库中的所有对象信息的知识库, Oracle RDBMS 使用数据字典获取对象信息和安全信息, 而用户和数据库系统管理员用它来查阅数据库信息。它保存着数据库中数据库对象和段的信息, 例如表、视图、索引、包以及过程, 它还保存着关于如用户、权限、角色、审计和约束等的信息。数据字典是只读的, 你永远不要尝试对任何数据字典表中的任何信息进行手工更新或改动。它由四部分组成: 内部 RDBMS (X\$) 表、数据字典表、动态性能 (V\$) 视图和数据字典视图。

### 5.5.1 内部RDBMS (X\$) 表

Oracle数据库的心脏就是即所谓的内部 RDBMS (X\$) 表, 这些表被 Oracle RDBMS 用于跟踪内部数据库信息。X\$表是加密命名的、非文献性的表, 并且几乎是无法解密的。大多数 X\$表被设计成不能被数据库系统管理员或用户直接使用。尽管如此, 它们仍包含着有价值的信息。许多非文献性的或内部的统计和配置只能在 X\$表中找到。

要解密在某个 X\$表中贮存的信息, 最简单的方法是从一个已知的数据字典表倒推寻找。SQL\*Plus 自动跟踪特性对这项工作是非常有用的。例如, 要想知道 V\$SGASTAT 中的信息到底存放在哪里, 你可以执行下面的分析步骤:

1) 作为 SYS 用户 (或者一个对 X\$ 和 V\$ 表有明确访问权限的帐户) 登录到 SQL\*Plus。如果你所登录的模式中不存在 PLAN\_TABLE, 你可以通过运行 \$ORACLE\_HOME/rdbms/admin/UTLXPLAN.sql 语句来建立一个。

2) 执行下列 SQL\*Plus 命令: SET AUTOTRACE ON。

3) 针对有你感兴趣内容的表执行一条查询, 把 WHERE 子句设为一个永远不会为真的值, 以便没有任何行能够返回: SELECT \* FROM V\$sgastat WHERE 0=1;

在其他信息中, 自动追踪返回类似下列的输出:

```
Execution Plan
0  SELECT STATEMENT Optimizer=CHOOSE
1  0      FILTER
2  1      FIXED TABLE (FULL) OF 'X$KSMSS'
```

从 SQL 追踪的输出, 你可以解密数据字典基表, 从基表中抽取视图信息。查询建立在这种形式上的 X\$ 表, 常常可以产生令人吃惊的信息。

### 5.5.2 数据字典表

数据字典表 (data dictionary table) 存储表、索引、约束以及所有其他数据库结构的信息。它们属于 SYS, 通过运行 SQL.BSQ 脚本来创建 (在数据库创建时自动发生)。通过它们名字后面的美元 (\$) 符号 (tab\$, seg\$, cons\$ 等等), 可以很容易地将它们辨认出来。在数据字典视图中可以找到数据字典表中的大部分信息, 但是一些应用或查询也可以从使用包含在基表中的信息中获益。

数据字典的列和表在 SQL.BSQ 文件中被很好地归档, 这个文件在 \$ORACLE\_HOME/dbs 目录中可以找到。当你熟悉了 SQL.BSQ 的内容后, 你就可以更好地理解 Oracle RDBMS 实际

上是怎样贮存数据字典和数据库信息的了。

### 5.5.3 动态性能视图

动态性能 (V\$) 视图 (dynamic performance (V\$) view) 是 Oracle 数据库系统管理员的主要依靠, 这些视图包含了大量数据库函数运行时的性能和统计信息。它们还具有相当的可读性 (与 X\$ 表相反), 也就是说能够被数据库系统管理员用于诊断和解决问题。关于大多数 V\$ 视图的文档能够在 Oracle Reference Manual 中查到。

注意, V\$ 视图实际上是由 SYS 拥有的 V\$ 视图的公共同义词, 当编写读取 V\$ 表的存储过程或函数时, 这是很值得注意的。引用或授权给基表 V\$ 视图 (而不是 V\$ 公共同义词) 常常是更必要的。

### 5.5.4 数据字典视图

数据字典视图是在 X\$ 和数据字典表上创建的视图, 也就意味着它们能被终端用户和数据库系统管理员使用和查询, 它们被分成三类: DBA\_、ALL\_ 和 USER\_ 视图。DBA\_ 视图包含了数据库所有对象的信息。例如, DBA\_TABLES 包含所有已创建表的信息, ALL\_ 视图包含了用户查询表时可以访问的所有对象的信息, USER\_ 视图包含了用户查询表时表所拥有的全部对象的信息。

## 5.6 其他数据库对象

在数据库中还贮存着一些其他的对象, 这些对象没有被准确地按段分类, 尽管如此还是应该讨论一下它们。它们包括视图、序列、同义词、触发器、数据库链以及存储包、过程和函数。下面几节对它们进行说明。

### 5.6.1 视图

视图是存储的能够被查询的 SQL 语句。由于安全原因, 视图用于隐藏一些数据 (例如一个 HR 视图只显示姓、名和地址信息, 而没有显示社会保险号码和工资数据), 并使复杂的查询变得易于理解和使用。通过在远程数据库表上创建视图, 视图也能被用于隐藏分布式数据库对象。任何可以做为 SQL 查询而执行的语句, 都能被创建成为一个视图。

注意 在设计应用程序时, 视图是非常有用的, 因为它们能够以表的形式隐藏复杂的查询逻辑, 使得更易于查询。它们可以和嵌在它们内部的优化器提示一起被创建, 以确保最佳的查询性能。

THE DBA\_VIEWS 视图存有在数据库中创建的视图的信息。

### 5.6.2 序列

序列是用于产生唯一数码的数据库对象。序列创建时带有初始值、增量值以及最大值。每次从序列中返回一个数字, 当前序列值是一个一个递增的, 每个序列产生的数字可达到 38 位长度。

可以通过选择来自序列的 NEXTVAL 或 CURRVAL 伪列使用序列。例如: 你有一个名为

EMP\_SEQ的序列，执行“SELECT EMP\_SEQ.NEXTVAL FROM DUAL；”语句，返回序列的下一个整数值，并且当前序列值加1；你可以使用“SELECT EMP\_SEQ.CURRVAL FROM DUAL；”语句，返回当前序列的整数值。注意，要使用 CURRVAL，你必须通过执行一个序列在NEXTVAL 伪列上的查询，来预先初始化用户会话的序列。

序列最通常的用途是提供唯一的数字作为表主键列的代用品。有关序列的信息贮存在 DBA\_SEQUENCES 视图中。

### 5.6.3 触发器

触发器（trigger）是存储过程，当针对一个表发生特定的动作时，就会激活它。触发器可以被编码。当针对一个表进行插入、更新、删除或三种操作的结合时，激活触发器，也可以在某行被影响或某条语句出现时被激活。触发器经常用于加强数据完整性约束和业务规则，这些业务规则对于内建的 Oracle 引用完整性约束来说实在是太复杂了。关于数据库触发器的信息可以在 DBA\_TRIGGERS 视图找到。

### 5.6.4 同义词

同义词（synonym）是指向其他数据库表的数据库指针。当你创建一个同义词时，你就指定了一个同义词名字和同义词所引用的对象。当你引用同义词名字时，Oracle 服务器会自动地用同义词定义的对象名字来代替同义词的名字。

同义词有两种类型：私有（private）和公共（public）。私有同义词是在指定的模式中创建的，并且只允许拥有它的模式访问。公共同义词由 PUBLIC 模式所拥有，所有的数据库模式都可以引用它们。

理解在 SQL 语句中对象名字的解析顺序是十分重要的。如果执行 SQL 语句“SELECT \* FROM EMP\_SALARY；”，Oracle 服务器将按照如下方式决定 EMP\_SALARY 对象：

- 1) 首先，服务器查看在发出命令的用户模式中是否有名为 EMP\_SALARY 的表或视图。
- 2) 如果这个表或视图不存在，Oracle 检查名为 EMP\_SALARY 的私有同义词是否存在。
- 3) 如果这个私有同义词存在，这个同义词所引用的对象将取代 EMP\_SALARY。
- 4) 如果这个私有同义词不存在，检查名为 EMP\_SALARY 的公共同义词是否存在。
- 5) 如果这个公共同义词不存在，Oracle 返回消息“ORA-00942, table or view does not exist”。

公共同义词应小心使用，因为所有的模式都可以用公共同义词决定对象名字，以致产生无法预料的结果。

关于公共同义词的信息贮存在 DBA\_SYNONYMS 中。注意，在这个视图中，公共同义词的拥有者作为 PUBLIC 被列出来。

### 5.6.5 数据库链

数据库链（database link）是与远程数据库连接的存储定义，它们用于查询分布式数据库环境中的远程表。因为它们存贮在 Oracle 数据库中，它们被归入数据库对象类别。关于数据库链的详细信息可以在第 40 章“分布式数据库管理”和 DBA\_DB\_LINKS 数据字典视图中得到。

注意 如果使用特定的 UserID 和口令与远程数据库连接，DBA\_DB\_LINKS 是一种能够

以明码存储口令的视图。当允许终端用户访问这个数据库视图时，一定要小心。

#### 5.6.6 包、过程和函数

存储包、过程和函数与它们的源代码一起存在数据字典中。一个存储过程是一个操作的代码单元，可以被传递参数，并能够返回值。一个存储函数是一个代码单元，可以被传递参数，并能够返回一个值。一个包是一个过程、变量和函数的集合，包按照功能被逻辑地分组。参见第26章“管理员使用的SQL \* Plus”可得到更多的信息。

你可以通过DBA\_OBJECTS和DBA\_SOURCE视图查看有关存储包、过程和函数的有关信息。