第1章

Oracle的体系结构

本章内容比较枯燥，但它是理解以后章节的基础。如果读者在开始学习时对有些内容未能完全理解也先不用太着急，可以继续学习后面的内容，等使用了Oracle系统一段时间之后，一些概念就容易理解了。

1.1 Oracle 引入复杂的体系结构的原因

Note

数据库管理系统引入非常复杂的内存和外存体系结构的主要原因是为了有效地管理稀有的系统资源。资源不足不只是数据库管理系统需要面对的。其实，在我们五千年的人类发展历史中，我们的祖先一直在同资源不足作斗争。历史上粮食和土地等一直都是稀有资源，还记得我们的祖先用什么方法来管理这些稀有资源吗？用战争，我们的祖先为粮食而战，为土地而战；当代人类为石油而战，为市场而战，为金钱而战。

那么在 Oracle 数据库中什么是稀有资源呢？又是如何来管理它们的呢？如果读者接触过数据库或读过相关的书籍，应该会有印象，数据库的数据量和输入/输出量都是相当大的，而这些数据一般都存在硬盘（外存）上，因此硬盘为数据库的一类资源。为了方便介绍，图 1-1 给出了硬盘的内部结构示意图。

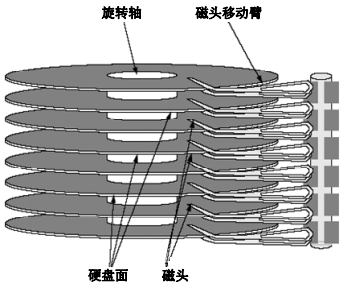


图 1-1

从图 1-1 可以看出，所有硬盘上数据的访问都是通过硬盘的旋转和磁头的移动来完成的，这种旋转和移动是机械运动。因为在计算机中所有数据的修改操作必须在内存中进行，所以内存也是数据库的一类资源。为了帮助读者更好地了解内存与外存的区别，表 1-1 给出了内存和外存的简单比较。

表 1-1

项　　目	内　　存	外存（硬盘）
数据访问速度	很快	很慢
存储的数据	临时	永久
价钱	很贵	相当便宜

从表 1-1 的比较可知，内存的数据访问速度要比外存（硬盘）快得多。这是因为内存的数据访问是电子速度，而硬盘的数据访问主要取决于机械速度。也就是说，如果一个数据库管理系统能够使绝大多数（如 90%以上）数据操作在内存中完成，那么该数据库管理系统的效率将非常高。但是由于内存中的数据在断电或出现系统故障时会消失，所以数据库管理系统还必须保证所有的数据改动都必须及时写到硬盘上，以保障不会丢失数据；即使数据库崩溃之后，所有提交过的数据都能得到完全恢复。尽管可以通过加大内存来提高数据库管理系统的效率，但在大多数情况下信息系统的开发和维护经费都是有限的。

通过以上的讨论，读者应该意识到，在数据库管理系统中最宝贵的稀有资源是内存。为了高效地使用内存这种稀有资源，同时保证不会丢失数据库中的任何数据，Oracle 数据库管理系统引入了一个非常复杂的体系结构。

1.2 Oracle 数据库中常用的术语

Note

为了讲解容易，在详细讨论 Oracle 体系结构之前，先介绍一下相关的名词和术语。在这里只给出实用的解释，并不追求学术上的严谨。

- ☑ 进程（process）：一段在内存中正在运行的程序。如果没有学过计算机操作系统相关课程，可以把进程想象成能够自动完成某些特定任务的任何东西，如训练有素的宠物、跑龙套的演员等。
- ☑ 后台进程（background process）：进程的一种。在内存中运行时，不占显示，而且优先级比前台进程低。需要注意的是，在运行进程中只能有一个前台进程，但可以同时有多个后台进程。
- ☑ 缓冲区（buffer）：一段用来临时存储数据的内存区。
- ☑ 主机（host）：计算机系统的另一个称呼。
- ☑ 服务器（server）：一台在网络中向其他计算机系统提供一项或多项服务的主机。
- ☑ 客户机（client）：一台使用由服务器（server）提供服务的计算机系统。

1.3 Oracle 数据库管理系统的体系结构

为了能使 Oracle 数据库管理系统满足商业用户的要求，Oracle 引入了如图 1-2 所示复杂的体系结构。

上述体系结构主要包括 Oracle 服务器（server），还包括一些其他的关键文件、用户进程和服务器进程等。

Oracle 服务器由 Oracle 实例（instance）和 Oracle 数据库（database）两大部分组成。它是一个数据库管理系统，提供了一致、开放和多样的信息管理的方法和途径。服务器中的一些结构并不在处理 SQL 语句时使用，而是为了改进数据库系统的效率或数据的恢复等而设计的。

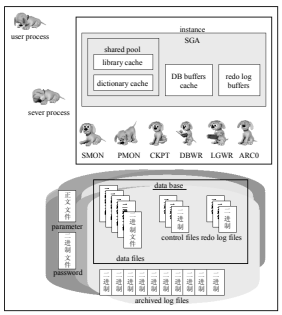


图 1-2

1.4 Oracle 服务器

Oracle 服务器（Oracle Server）实际上是一个逻辑上的概念，一个 Oracle 服务器与一台计算机之间并不存在一一对应的关系。Oracle 服务器 = 实例（instance）+ 数据库（database）。虽然在一台计算机上有时可以安装多个 Oracle 服务器，但是一般情况下都是只安装一个，因为在实际工作中，一台计算机上运行一个 Oracle 服务器有时效率都很难保证。Oracle 服务器一般包括以下三种安装方式。

（1）基于主机方式：在此种配置下，用户可直接在安装了数据库的计算机上登录 Oracle 数据库。

（2）客户端-服务器（client-server）（两层模型）方式：数据库和客户终端分别安装在不同的计算机上，用户可通过网络从个人计算机（客户端）上访问数据库。

（3）客户端-应用服务器-服务器（client-application server-server）（三层模型）：用户首先从自己的个人计算机登录应用服务器，再通过应用服务器访问真正的数据库。

Note

1.5 Oracle 实例

Oracle 实例是一种访问数据库的机制，它由内存结构和一些后台进程组成。它的内存结构也称为系统全局区（System Global Area，SGA）。系统全局区是实例的最基本的部件之一。实例的后台进程中有 5 个是必需的，即只要这 5 个后台进程中的任何一个未能启动，则该实例都将自动关闭。这 5 个后台进程分别是 SMON、PMON、DBWR、LGWR 和 CKPT。在 OCP 考题中有时可能会问哪些后台进程是可选的？除了这 5 个都是可选的。实例一旦启动即分配系统全局区和启动所需的后台进程。这里应该指出的是，每个实例只能操作其对应的一个数据库。但是反过来是不成立的，因为一个数据库可以同时被几个实例操作（在 Oracle 集群中）。

系统全局区（SGA）中包含了以下几个内存结构：共享池（shared pool）、数据库高速缓冲区（database buffer cache）、重做日志缓冲区（redo log buffer）和其他的一些结构（如锁和统计数据）等。

1.6 Oracle 数据库

Oracle 数据库是数据的一个集合，Oracle 把这些数据作为一个完整的单位来处理。Oracle 数据库也称为物理（外存）结构，为数据库信息提供了真正的物理存储，它由以下三类操作系统文件组成。

（1）控制文件（control files）：包含了维护和校验数据库一致性所需的信息。

（2）重做日志文件（redo log files）：包含了当系统崩溃后进行恢复所需记录的变化信息。

（3）数据文件（data files）：包含了数据库中真正的数据。

1.7 Oracle 其他的关键文件

除了以上三类数据库文件之外，Oracle 服务器还需要其他的一些文件，这些文件不属于数据库。其中包括以下几种。

- ☑ 初始化参数文件（parameter files）：定义了实例的特性，如系统全局区中一些内存结构的大小、DBWR 的个数等。
- ☑ 密码文件（password files）：包含了数据库管理员或操作员用户在启动和关闭实例时所需的密码。虽然 Oracle 数据库提供了相当完善的安全管理机制，但是在 Oracle 数据

库没有开启时，如何验证要启动数据库的人是真正的数据库管理员还是操作员呢？这就是 Oracle 引入密码文件的原因。

☑ 归档重做日志文件（archived redo log files）：重做日志文件的脱机备份。在系统崩溃后恢复时可能需要这些文件。

Note

1.8 建立与 Oracle 实例的连接

Oracle 实例是用 Oracle 的 STARTUP 命令启动的（该命令将在后面的章节中详细介绍）。它的启动就意味着 SGA 的所有内存结构都已生成，所有必需的后台进程都已在内存中运行。那么此时用户又是如何使用 Oracle 数据库呢？

用户在向 Oracle 数据库发出 SQL 命令之前必须与实例建立连接。用户启动一个工具，如 SQL*Plus，或运行一个利用 Oracle 工具开发的应用程序，如用 Oracle Forms 开发的应用程序时，该工具或应用程序就被作为一个用户进程来执行。

注意

用户进程是不能直接访问数据库的。用户进程是运行在客户端的。

在专用连接的情况下（即默认情况下），当一个用户登录 Oracle 服务器时（如在 SQL*Plus 的提示下输入用户名和密码），如果登录成功（用户名和密码都准确无误），Oracle 就在服务器所运行的计算机上创建了一个服务器进程。在这种连接下，该服务器进程只能为该用户进程提供服务，用户进程与服务器进程是一对一的关系。用户进程向服务器进程发请求，服务器进程对数据库进行实际的操作并把所得的结果返回给用户进程。就好像一个富豪想炒股票，但又不懂股票市场的运作，于是他请了一位股票经纪人。这位富豪就相当于用户进程，而股票经纪人就相当于服务器进程，股票市场就相当于 Oracle 服务器。

一个用户每次登录 Oracle 服务器，如果成功，该用户就与 Oracle 服务器建立了连接，而这种连接的状态被称为会话。一个会话开始于用户成功地登录 Oracle 服务器，终止于用户退出或非正常终止连接。一个数据库用户可能同时有多个会话存在，即用相同的用户名和密码同时登录多次。

提示

虽然连接与会话都与用户进程紧密相关，但是这两者之间还是有很大的不同。连接是指一个用户进程与一个 Oracle 数据库实例之间的通信路径，而会话代表的是一个当前用户登录该数据库实例的状态。

1.9 各种不同的连接方式

连接是用户进程与 Oracle 服务器之间的通信路径。与 Oracle 服务器的三种安装方式相对应，

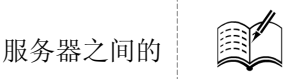
一个数据库用户可以用以下三种方式之一与 Oracle 服务器连接。

（1）基于主机方式：此时的用户进程与服务器进程是在同一台计算机的相同的操作系统上的，用户进程与 Oracle 服务器之间的通信路径是通过操作系统内部进程通信（Inter Process Communication，IPC）机制来建立的。

（2）客户端-服务器（client-server）（两层模型）方式：用户进程与 Oracle 服务器之间的通信是通过网络协议（如 TCP/IP）来完成的。

（3）客户端-应用服务器-服务器（client-application server-server）（三层模型）：用户的个人计算机通过网络与应用服务器或网络服务器通信，而该应用服务器或网络服务器又是通过网络与运行数据库的计算机相连的。例如，用户使用浏览器通过网络运行 Windows 2008 Server 上的应用程序，而 Windows 2008 Server 又从运行在 UNIX 主机上的 Oracle 数据库中提取数据。

以上所介绍的连接是用户进程与服务器进程的一对一的连接，也称为专用服务器连接（dedicated server connection）。除了这种连接外，在联机事务处理（On Line Transaction Processing，OLTP）系统的配置时还有另外一种连接，它在 Oracle 9i 之前的版本中称为多线程（MTS）连接，在 Oracle 9i 或以后的版本中称为共享服务器（shared server）连接。这种连接在 Oracle 的网络和调优的书籍中有相关介绍。



Note

1.10 服务器进程

当 Oracle 创建一个服务器进程的同时要为该服务器进程分配一个内存区，该内存区称为程序全局区（Program Global Area，PGA）。与 SGA 不同，PGA 是一个私有的内存区，不能共享，且只属于一个服务器进程。它随着服务器进程的创建而被分配，随着服务器进程的终止而被回收。在专用服务器进程的配置情况下，程序全局区主要包括以下结构。

（1）排序区（sort area）：用于处理 SQL 语句所需的排序。

（2）Cursor 状态区（cursor state）：用于指示会话当前所使用的 SQL 语句的处理状态。

（3）会话信息区（session information）：包括会话的用户权限和优化统计信息。

（4）堆栈区（stack space）：包括其他的会话变量。

如果是共享服务器进程或多线程的配置，以上这些结构除了堆栈区外大部分都将存在于 SGA 中。如果有 large pool，它们就会存在于 large pool 中，否则它们就存在于共享池中。

提示

在许多中文的 Oracle 书中将 Cursor 一词翻译成游标（或光标），实际上这是一个误会。虽然 Cursor 的英文字面意思确实是游标（或光标），但是在这里与游标（或光标）毫无关系。实际上，Cursor 是 Current set of rows 的缩写。引入 Cursor 这一数据结构的目的是减少 I/O——将磁盘操作变成内存操作。当要对一个或几个表中的一批数据进行反复处理时，为了避免重复访问这些表（访问硬盘），Oracle 可以将这批数据一次性装入内存。而这些数据行被称为活动集（Active set），Oracle 使用一个指针指向活动集的第一行（第一个记录），如图 1-3 所示。程序可以利用这个指针来处理活动集中的所有数据行，而不需要重复访问硬盘。有关 Cursor 的详细介绍属于 PL/SQL 程序设计的课程。

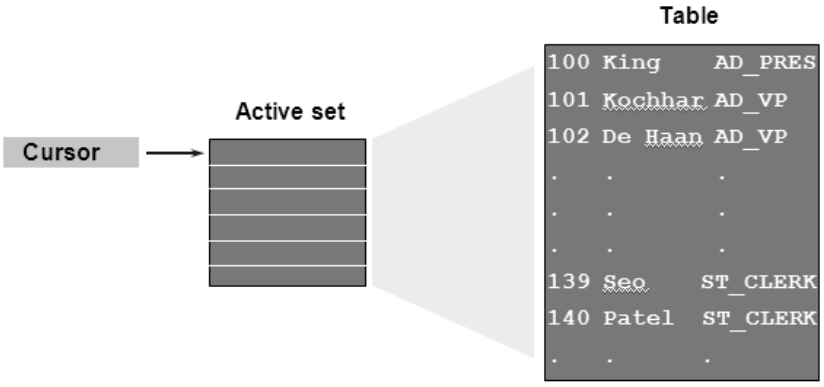


图 1-3

1.11 Oracle 执行 SQL 查询语句的步骤

如果用户在 SQL*Plus 下输入了如下查询语句：SELECT * FROM dept;，那么 Oracle 将如何来处理这个语句？SQL 语句的执行主要由用户进程与服务器进程来完成，其他的一些进程可能要辅助完成这一过程。查询语句与其他的 SQL 语句有所不同，如果一个查询语句执行成功，就要返回查询的结果，而其他的 SQL 语句只是返回执行成功或失败的信息。查询语句的处理主要包括以下三个阶段：编译（parse）、执行（execute）和提取数据（fetch）。

- ☑ 编译(parse)：在进行编译时，服务器进程会将 SQL 语句的正文放入共享池(shared pool)的库高速缓存（library cache）中并将完成以下处理。
  - ✓ 首先在共享池中搜索是否有相同的 SQL 语句（正文），如果没有就进行后续的处理。
  - ✓ 检查该 SQL 语句的语法是否正确。
  - ✓ 通过查看数据字典来检查表和列的定义。
  - ✓ 对所操作的对象加编译锁（parse locks），以便在编译语句期间，这些对象的定义不被改变。
  - ✓ 检查所引用对象的用户权限。
  - ✓ 生成执行该 SQL 语句所需的优化的执行计划（执行步骤）。
  - ✓ 将 SQL 语句和执行计划装入共享的 SQL 区。

以上的每一步操作都是在处理正确时才进行后续的处理。如果不正确，就返回错误。

- ☑ 执行（execute）：Oracle 服务器进程开始执行 SQL 语句是因为它已获得了执行 SQL 语句所需的全部资源和信息。
- ☑ 提取数据（fetch）：Oracle 服务器进程选择所需的数据行，并在需要时将其排序，最后将结果返回给用户（进程）。

1.12 共 享 池

Note

SGA 中的共享池（shared pool）是由库高速缓存（library cache）和数据字典高速缓存（data dictionary cache）两部分所组成。服务器进程将 SQL（也可能是 PL/SQL）语句的正文、编译后的代码（parsed code）以及执行计划都放在共享池的库高速缓存中。在进行编译时，服务器进程首先会在共享池中搜索是否有相同的 SQL 或 PL/SQL 语句（正文），如果有就不进行任何后续的编译处理，而是直接使用已存在的编译后的代码和执行计划。

提示

库高速缓存包含共享 SQL 区和共享 PL/SQL 区两部分，它们分别存放 SQL 和 PL/SQL 语句以及相关的信息。在 Oracle 11g 中，库高速缓存还可以包括 SQL 或 PL/SQL 的执行结果，这样其他使用相同 SQL 或 PL/SQL 语句的进程就可以直接共享这些结果了，其效率将更高。

要想共享 SQL 或 PL/SQL 语句：第一，库高速缓存要足够大，因为只有这样，要共享的 SQL 或 PL/SQL 语句才不会被很快地淘汰出内存；第二，SQL 或 PL/SQL 语句是能共享的通用代码（generic code），因为 Oracle 是通过比较 SQL 或 PL/SQL 语句的正文来决定两个语句是否相同的，只有当两个语句的正文完全相同（字母大小写 Oracle 会自动转换，而且多余的空格或制表键 Oracle 也会自动压缩）时，Oracle 才重用已存在的编译后的代码和执行计划。这里通过以下的实例来进一步解释这一点。读者猜猜如下的两个 SQL 语句是否相同？

```
select * from emp where sal >= 1500;
select * from emp where sal >= 1501;
```

答案是不相同的（在 Oracle 默认的配置下，Oracle 8i 和 Oracle 9i 以及更高的版本可以通过重新设置 CURSOR_SHARING 参数来修改默认配置，有兴趣的读者可参阅 Oracle 调优方面的书籍）。

可以通过使用绑定变量的方式来重写以上的 SQL 语句，代码如下：

```
SELECT * FROM emp WHERE sal >= :g_sal;
```

上述语句就是可以共享的通用代码，因为变量不是在编译阶段而是在运行阶段赋值的。引入库高速缓存的目的是共享 SQL 或 PL/SQL 代码。那么 Oracle 又是怎样有效地管理库高速缓存的呢？Oracle 是使用一个称为 LRU（Least Recently Used）的队列（list）或算法（algorithm）来实现对库高速缓存的管理的。LRU 队列的算法大致如下：刚使用的内存块（的地址）放在 LRU 队列的头上（最前面），当一个服务器进程需要库高速缓存的内存空间而又没有空闲的内存空间时，该进程就从 LRU 队列的尾部（最后面）获得所需的内存块，这些内存块一旦被使用，它们（的地址）就立即放在 LRU 队列的最前面。这样那些长时间没使用过的内存块将自然地移到 LRU 队列的尾部而被最先使用。

从以上的讨论可知，为了能够共享 SQL 或 PL/SQL 的代码，库高速缓存必须足够大，因为这样，那些可以共享的 SQL 或 PL/SQL 代码才不会被很快地淘汰出内存。不过 Oracle 并没有给出直接设置库高速缓存大小的方法，只能通过设置共享池的大小来间接设置库高速缓存的大小。

介绍完库高速缓存，接下来将介绍数据字典高速缓存（data dictionary cache）。当 Oracle 在执行 SQL 语句时，服务器进程将把数据文件、表、索引、列、用户和其他的数据对象的定义和权限的信息放入数据字典高速缓存。如果在这之后，有进程（用户）需要同样的信息，如表和列的定义，那么所有这些信息都将从数据字典高速缓存中获得。因为以上所说的这些信息都存在于 Oracle 数据库的数据字典中，这也可能就是将该部分内存称为数据字典高速缓存的原因。

Note

表和列的定义等重用的机会要比 SQL 语句大，因此为了能达到共享这些信息的目的，数据字典高速缓存应该尽可能设置得大一些。不过与库高速缓存一样，Oracle 并没有给出直接设置数据字典高速缓存大小的方法，只能通过设置共享池的大小来间接地设置数据字典高速缓存的大小。在 Oracle 9i 之前的版本，可以通过修改参数文件中的 SHARED_POOL_ SIZE 的值来改变共享池的大小，但一定要重新启动 Oracle 数据库。在 Oracle 9i 或以后的版本中，也可以使用类似于例 1-1 的命令来改变共享池的大小。

例1-1

```
SQL> ALTER SYSTEM SET SHARED_POOL_SIZE = 250M;
系统已更改。
```

但是所改变共享池的大小受限于 SGA_MAX_SIZE 参数。该参数将在以后的章节中详细介绍。

注意

本书中采用如下的约定:

- ☑ SQL>为 SQL * Plus 的提示符。
- ☑ 没有阴影的内容为要输入的 SQL 语句或 SQL * Plus 命令等，如在例 1-1 中要输入 ALTER SYSTEM SET SHARED_POOL_SIZE=250M;。
- ☑ 阴影中的内容为系统显示（输出）的结果，如在例 1-1 中的“系统已更改。”。

提示

即使把共享池设置得足够大且所使用的 SQL 或 PL/SQL 语句是能共享的代码，Oracle 也并不能一定使用内存（共享池）中的代码。例如，当有用户修改了某个对象的定义之后，所有使用这个对象的内存（共享池）中的代码全部被 Oracle 设置为无效，因此在使用时必须重新编译。

1.13 数据库高速缓冲区

如果用户发出了以下 SQL 语句：SELECT * FROM emp，那么 Oracle 怎样提取数据库中的数据呢？服务器进程将首先在数据库高速缓冲区（database buffer cache）中搜寻所需的数据，如果找到了就直接使用而不进行磁盘操作；如果没找到就将进行磁盘操作，把数据文件中的数据读入到数据库高速缓冲区中。

从以上的讨论可知，为了能够共享数据库中的数据，数据库高速缓冲区要足够大，因为只有这样那些可以共享的数据才不会被很快地淘汰出内存。Oracle 也是使用 LRU 的队列（list）或算法（algorithm）来实现对数据库高速缓冲区的管理。在早期的版本中可以使用参数文件中的 DB_BLOCK_SIZE 和 DB_BLOCK_BUFFERS 两个参数来设置数据库高速缓冲区的大小。其中 DB_BLOCK_SIZE 为 Oracle 数据块（内存缓冲区）的大小，在 Oracle 数据库中内存和外存的数据块的大小是相同的。DB_BLOCK_BUFFERS 为内存缓冲区的个数。数据库高速缓冲区大小为这两个参数的乘积。但是 DB_BLOCK_SIZE 的值是在创建数据库时设定的，如果要改变该参数的值一般需要重建数据库。因此，多数情况下只能通过改变 DB_BLOCK_BUFFERS 的值来调整数据库高速缓冲区的大小，但此时必须重新启动 Oracle 数据库。在 Oracle 9i 或以后的版本中，Oracle 引入了另一个参数 DB_CACHE_SIZE，这个参数是一个动态参数，即可以在数据库运行时动态地改变该参数。可以使用类似于例 1-2 的命令来改变数据库高速缓冲区的大小。

Note

例1-2

```
SQL> ALTER SYSTEM SET DB_CACHE_SIZE = 250M;
系统已更改。
```

1.14 内存缓冲区顾问

Oracle 9i 或以后的版本还提供了一个称为内存缓冲区顾问（v$db_cache_advice）的工具来帮助获得调整数据库高速缓冲区的统计信息。内存缓冲区顾问一共有三种状态。

（1）ON：该工具打开，为该工具分配内存并进行统计信息的收集，要有一定的内存和 CPU 开销。

（2）READY：该工具关闭，为该工具分配内存但不进行统计信息的收集，因此没有 CPU 开销。

（3）OFF：该工具关闭，不为该工具分配内存，也不进行统计信息的收集，因此既没有内存的开销，也没有 CPU 开销。

可以通过修改初始化参数 DB_CACHE_ADVICE 的值来改变该工具的状态。该参数是一个动态参数，因此可以使用 ALTER SYSTEM 命令来修改。例如，可以利用类似于例 1-3 的 SQL 语句来查看内存缓冲区顾问的状态。

例1-3

```
SQL> select id, name, block_size, advice_status
  2  from v$db_cache_advice;
        ID NAME                                BLOCK_SIZE ADV
---------- ---------------------------------- ---------- ---
         3 DEFAULT                                  4096 ON
         3 DEFAULT                                  4096 ON
         3 DEFAULT                                  4096 ON
         3 DEFAULT                                  4096 ON
         3 DEFAULT                                  4096 ON
…
已选择 20 行。
```

此时，例 1-3 的显示结果表明内存缓冲区顾问在开启状态。

提示

显示结果中的“…”表示省略了一些行的显示。

Note

之后可以使用类似于例 1-4 的命令将内存缓冲区顾问工具关闭。

例1-4

```
SQL> alter system set db_cache_advice = off;
系统已更改。
```

这时可以再使用类似于例 1-5 的 SQL 语句来查看它的状态。

例1-5

```
SQL> select id, name, block_size, advice_status
  2    from v$db_cache_advice;
        ID NAME                                     BLOCK_SIZE ADV
---------- ---------------------------------------- ---------- --------
         3 DEFAULT                                        4096 OFF
         3 DEFAULT                                        4096 OFF
         3 DEFAULT                                        4096 OFF
         3 DEFAULT                                        4096 OFF
         3 DEFAULT                                        4096 OFF
…
已选择 20 行。
```

此时，例 1-5 的显示结果表明已成功地关闭了内存缓冲区顾问（详细地介绍该工具的使用已超出了本书的范围，有兴趣的读者可参阅 Oracle 调优方面的书籍）。

1.15 重做日志缓冲区

从理论上来讲，如果数据库不会崩溃，则根本没有必要引入重做日志缓冲区（redo log buffer）。引入重做日志缓冲区的主要目的（在 Oracle 8i 之前的版本中也是唯一的目的）就是数据的恢复。Oracle 在使用任何 DML 或 DDL 操作改变数据之前都将恢复所需的信息，即在写数据库高速缓冲区之前，先写入重做日志缓冲区。

与执行查询语句有所不同，Oracle 在执行 DML 语句时只有编译（parse）和执行（execute）两个阶段。以下是 Oracle 执行 UPDATE 语句的步骤：

（1）如果数据和回滚数据不在数据库高速缓冲区中，则 Oracle 服务器进程将把它们从数据文件中读到数据库高速缓冲区中。

（2）Oracle 服务器进程在要修改的数据行上加锁（行一级的锁，而且是在内存的数据行上加锁）。

（3）Oracle 服务器进程将数据的变化信息和回滚所需的信息都记录在重做日志缓冲区中。

（4）Oracle 服务器进程将回滚所需的原始值和对数据所做的修改都写入数据库高速缓冲区。之后在数据库高速缓冲区中，所有的这些数据块都将被标为脏缓冲区，因为此时内外存的

数据是不同的（不一致的）。

Oracle 处理 INSERT 或 DELETE 语句的步骤与处理 UPDATE 语句的步骤大体相同。

注意

有关回滚数据在本书的后面章节中还要介绍，读者也可以参阅笔者的《从实践中学习 Oracle SQL 培训教程——从实践中学习 Oracle SQL 及 Web 快速应用开发》12.20 节的 268～269 页。

Note

1.16 大池和 Java 池

除了以上所介绍的内存结构之外，SGA 中还有可能包含大池（large pool）和 Java 池（Java pool）两个可选的内存结构。

引入 large pool 的主要目的应该是提高效率。large pool 是一个相对比较简单的内存结构，与 shared pool 不同的是它没有 LRU 队列。在多线程（MTS）或共享服务器（shared server）连接时，Oracle 服务器进程的 PGA 的大部分区域（也称为 UGA）将放入 large pool（stack space 除外）。另外，在大规模 I/O 及备份和恢复操作时可能使用该区。可以通过设置参数 LARGE_POOL_SIZE 的值来配置 large pool 的大小。该参数也是一个动态参数。

引入 Java pool 的目的是能够编译 Java 语言的命令。如果要使用 Java 语言就必须设置 Java pool。Java 语言在 Oracle 数据库中的存储与 PL/SQL 语言几乎完全相同。可以通过设置参数 JAVA_POOL_SIZE 的值来配置 Java pool 的大小。其数字的单位是字节（B）。

1.17 内存缓冲区大小的设定

在 Oracle 9i 之前的版本中，只能通过设置初始化参数文件中的一些参数来间接设置 SGA 的大小，如 DB_BLOCK_BUFFERS、LOG_BUFFER、SHARED_POOL_SIZE 等。而且所有的这些参数都是静态的，即当修改完初始化参数文件中这些参数的值之后，必须重新启动 Oracle 数据库。

在 Oracle 9i 以后的版本中，SGA 为动态的。SGA 中的内存缓冲区，如数据库高速缓冲区和共享池等都可以动态地增加和减少。Oracle 是利用所谓的区组（granule）来管理 SGA 的内存的。区组就是一片连续的虚拟内存区，是 Oracle 分配和回收内存区的基本单位。

区组的大小取决于所估计的 SGA 的大小。如果 SGA 的尺寸小于 128MB，则区组的大小即为 4MB；如果 SGA 的尺寸大于或等于 128MB，则区组的大小就为 16MB。Oracle 数据库一旦启动，SGA 中的每个内存缓冲区就会获得所需的区组。SGA 中至少包括三个区组：一个是 SGA 固定区（其中包含了重做日志缓冲区）；一个是数据库高速缓冲区；一个是共享池。

提示

Oracle 11g 和 Oracle 10g 较高的版本对 SGA 的尺寸进行了重新定义，当 SGA 的尺寸小于或等于 1GB 时，区组的大小为 4MB；当 SGA 的尺寸大于 1GB 时，区组的大小就为 16MB。实际上，这种改进主要得益于内存价格的暴跌和容量的稳步增加。

Note

Oracle 数据库管理员可通过 ALTER SYSTEM SET 命令来分配和回收区组。但总的内存大小不能超过参数 SGA_MAX_SIZE 所设定的值。

动态分配和回收内存的最大优点是在调整内存缓冲区大小时不需要重新启动数据库，这对 24 小时运营、7 天营业的商业数据库是至关重要的。

1.18 内存缓冲区信息的获取

可以使用例 1-6 的命令来获得参数 SGA_MAX_SIZE 的值。

例1-6

```
SQL> show parameter SGA_MAX_SIZE
NAME                                 TYPE                              VALUE
------------------------------------ --------------------------------- ------------
sga_max_size                         big integer                       744M
```

该命令的显示表明这个系统目前 SGA 总的内存大小为 744MB，另外也可以使用例 1-7 的命令来获得 SGA 的相关信息。

例1-7

```
SQL> show sga
Total System Global Area      778387456 bytes
Fixed Size                      1374808 bytes
Variable Size                 260048296 bytes
Database Buffers              511705088 bytes
Redo Buffers                    5259264 bytes
```

也可以先使用例 1-8 和例 1-9 的 SQL*Plus 命令来格式化显示输出。之后，利用数据字典 v$parameter，使用例 1-10 的 SQL 查询语句来获得参数 SGA_MAX_SIZE 的值。

例1-8

```
SQL> col name for a20
```

例1-9

```
SQL> col value for a25
```

例1-10

```
SQL>    select name, type, value
  2   from v$parameter
  3   where name = 'sga_max_size';
NAME                                 TYPE VALUE
-------------------------------- -----------------------------------------
```

sga_max_size	6	780140544

介绍完 SGA 的各个部分内存缓冲区之后，下面将详细讨论 Oracle 的主要后台进程。

Note

1.19 重做日志写进程及快速提交

重做日志写进程（LOG writer，LGWR）负责将重做日志缓冲区的记录顺序地写到重做日志文件中。为了更好地理解 LGWR 的操作原理，在这里先介绍 Oracle 提交（commit）语句是如何工作的。

Oracle 服务器使用了一种称为快速提交（fast commit）的技术，该技术既能保证 Oracle 系统的效率，又能保证在系统崩溃的情况下所有提交的数据可以得到恢复。为此 Oracle 系统引入了系统变化数（System Change Number，SCN）。无论任何时候，只要某个事务（transaction）被提交，Oracle 服务器都将产生一个 SCN（号码）并将其赋予该事务的所有数据行。在同一个数据库中 SCN 是单调递增的并且是唯一的。为了避免在进行一致性检验时操作系统时钟可能引发的问题，Oracle 服务器将 SCN 作为 Oracle 的内部时间戳来保证数据文件中数据的同步和数据的读一致性。

当在 SQL*Plus 中发了 commit 语句之后，Oracle 的内部操作步骤如下：

（1）服务器进程将把提交的记录连同所产生的 SCN（号码）一起写入重做日志缓冲区中。

（2）重做日志写进程将把重做日志缓冲区中一直到所提交的记录（包括该记录）的所有记录连续地写到重做日志文件中。在此之后，Oracle 服务器就可以保证即使在系统崩溃的情况下所有提交的数据也可以得到恢复。

（3）Oracle 通知用户（进程）提交已经完成。

（4）服务器进程将修改数据库高速缓冲区中相关数据的状态并释放资源和打开锁等。

此时可能这些数据并未被写到数据文件中，这些数据缓冲区被标为脏缓冲区，因为相同的数据在内外存中为不同的版本。数据库高速缓冲区中的数据是由 DBWR 写到数据文件中的。

曾有不少学生问过这样一个问题：“为什么不同时写两个数据文件呢？”Oracle 的这种解决方案的最大优点是在保证不丢失数据的同时，数据库的效率不会受到很大影响。因为重做日志文件中的记录是以最紧凑的格式存放的，所以它的 I/O 量要比对数据文件的操作少得多。另外 LGWR 是顺序地将重做日志缓冲区中的记录写到重做日志文件中的，这样其 I/O 速度要比将数据块写到数据文件中快得多。

重做日志写进程要在下列情况下将重做日志缓冲区的记录（内存）顺序地写到重做日志文件（外存）中：

- ☑ 当某个事务被提交时。
- ☑ 当重做日志缓冲区中所存的记录已超过缓冲区容量的 1/3 时。
- ☑ 在 DBWR 将数据库高速缓冲区中修改过的数据块写到数据文件之前（如果需要）。
- ☑ 每 3 秒钟。

因为在进行数据库恢复时需要重做日志数据，所以重做日志写进程只有在重做日志数据写到重做日志文件（磁盘）上时才能确定提交已经完成。在 Oracle 8i 之前的版本中，重做日志数据的唯一目的和用处就是数据库恢复。Oracle 在 Oracle 8i 的版本中引入了一个叫做重做日志挖掘器（logminer）的工具。该工具可以将重做日志文件或归档重做日志文件中的数据转换成用户

能理解的正文信息。在 Oracle 8i 中，该工具只有命令行操作方式。Oracle 9i 加强了此工具的功能并引入了一个称为日志挖掘浏览器（logminer viewer）的图形界面工具。

Note

1.20 数据库写进程

在本章开始时曾介绍过数据库的典型操作就是大规模地输入/输出（I/O），因此为了提高 Oracle 系统的效率，一要减少 I/O 量，这可能是 Oracle 引入 LGWR 的原因之一；二要减少 I/O 次数，这可能是 Oracle 引入数据库写进程（DBWR/DBWn）的主要原因。

提示

在 Oracle 的英文书中，有些将“数据库写进程”用 DBWR 表示，有些用 DBWn 表示。这是因为在一个 Oracle 实例中可以启动多个数据库写进程，特别是在要进行大规模输入/输出并且运行在多 CPU 计算机上的 Oracle 数据库系统。Oracle 早期的版本允许在一个实例上最多启动 10 个数据库写进程，它们分别是 DBW0～DBW9。Oracle 11g 对此进行了扩充，允许在一个实例上最多启动 36 个数据库写进程，它们分别是 DBW0～DBW9 和 DBWa～DBWz。Oracle 是使用参数 DB_WRITER_PROCESSES 来设置数据库写进程的个数的，如果在实例启动时没有说明数据库写进程的个数，Oracle 将根据 CPU 的个数来决定参数 DB_WRITER_PROCESSES 的设置。

可以使用例 1-11 的 SQL*Plus 的 show parameter 命令列出系统目前所启动的数据库写进程的个数。

例1-11

```
SQL> show parameter DB_WRITER_PROCESSES
NAME                 TYPE        VALUE
-------------------------------------------------------------------------
db_writer_processes  integer     1
```

数据库写进程负责将数据库高速缓冲区中脏缓冲区中的数据写到数据文件上。为了提高效率，数据库写进程并不是数据库高速缓冲区中的数据一有变化就写数据文件，而是积累了足够多的数据一次写一大批内存数据块到数据文件上。

数据库写进程将在下列事件之一发生时把数据库高速缓冲区中的数据写到数据文件上：

- ☑ 当脏缓冲区的数量超过了所设定的限额时。
- ☑ 当所设定的时间间隔已到时。
- ☑ 当有进程需要数据库高速缓冲区却找不到空闲的缓冲区时。
- ☑ 当校验（检查）点发生时。
- ☑ 当某个表被删除（drop）或被截断（truncate）时。
- ☑ 当某个表空间被设置为只读状态（read only）时。
- ☑ 当使用类似于 ALTER TABLESPACE users BEGIN BACKUP 的命令对某个表空间进行联机备份时。
- ☑ 当某个表空间被设置为脱机状态（offline）或重新设置为正常状态（normal）时等。

1.21 系统监督进程

Note

从前面的论述中可以知道，由于某种原因（如断电）Oracle 系统崩溃了，SGA 中任何没有来得及写到磁盘中的信息都将丢失，如有些已经提交的数据还没有真正地被写到数据文件中就丢失了。在这种情况下，当数据库重新开启时，系统监督进程（SMON）将自动地执行 Oracle 实例的恢复工作。其步骤如下：

（1）执行前滚（roll forward），即将已经写到重做日志文件中但还没写到数据文件中的提交数据写到数据文件中（Oracle 是用 SCN 号码来识别提交记录的）。

（2）在前滚完成后立即打开数据库，此时用户就可以登录并使用数据库了。这时在数据文件中可能还有一些没有提交的数据。之所以这样安排，主要是为了提高系统的效率。

（3）回滚没有提交的事务（数据）。除了 SMON 进程之外，服务器（server）进程也可能进行回滚没有提交的事务，但该进程只回滚它所用到的加锁的数据行。

除此之外，SMON 进程还要执行如下的磁盘空间的维护工作：

- ☑ 回收或合并数据文件中相连的空闲区。
- ☑ 释放临时段（在执行 SQL 语句时用作排序的磁盘区），将它们还给临时文件以作为空闲区使用。

1.22 进程监督进程

当某个进程崩溃时（如在没有正常退出 Oracle 的情况下重新启动了所用的 PC），进程监督进程（PMON）将负责它的清理工作，包括：

- ☑ 回滚用户当前的事务。
- ☑ 释放用户所加的所有表一级和行一级的锁。
- ☑ 释放用户所有的其他资源等。

1.23 校验（检查）点和校验点进程

Oracle 系统为了提高系统的效率和数据库的一致性，引入了一个称为校验点的事件。该事件是在当 DBWR 进程把在 SGA 中所有已经改变了的数据库高速缓冲区中的数据（包括提交的和没提交的数据）写到数据文件上时产生的。从理论上讲，校验点（checkpoint）和校验点进程可以完全不需要，因为 Oracle 系统利用重做日志数据和 SCN 号是能够保证数据库的完全恢复的。引入校验点可能是为了提高系统的效率。因为所有到校验点为止的变化了的数据都已写到了数据文件中，在实例恢复时校验点之前的重做日志记录已经不再需要，这样实例恢复速度就加快了。

在校验点事件发生时，Oracle 要将校验点号码（Oracle 系统自动产生的）写入所有相关的数据文件的文件头中，还要将校验点号码、重做日志序列号、归档日志名称和最低、最高 SCN 号

都写入控制文件中。

尽管经常产生校验点可以加快实例恢复的速度，但是由于在产生校验点时 Oracle 系统要进行大量的 I/O 操作，所以过于频繁地产生校验点会使数据库正常的联机操作受到冲击。数据库管理员要在实例恢复的速度和联机操作之间进行折中。一般的生产或商业数据库的校验点间隔是在 20 分钟或以上。

Note

1.24 归档日志进程

以上 5 个后台进程都是必需的，即它们中的任何一个停止后实例都将自动关闭。在可选后台进程中，归档日志（ARCH/ARCn）进程可能是最重要的一个可选后台进程，因为如果 Oracle 数据库的数据文件丢失或损坏，一般数据库要进行完全恢复，Oracle 数据库应运行在归档方式。

在 Oracle 数据库中，重做日志文件被划分为若干个组。当一组重做日志的文件被写满后，Oracle 就开始写下一组重做日志，这被称为日志切换。切换是以循环的方式进行的，即当最后一组写满后，又开始写第一组。因此，如果只有重做日志文件，即 Oracle 数据库运行在非归档方式下，当遇到数据文件丢失或损坏时，Oracle 系统很难保证完全恢复数据库中的数据。因为此时所需的重做记录可能因重做日志循环使用而被覆盖了。

在归档方式下，ARCn 进程将把切换后的重做日志文件复制到归档日志文件中。可以把归档日志文件看成是重做日志文件的备份，但归档日志文件是脱机的，即除了在进行（复制）时，Oracle 数据库在正常运行时是不会关注归档日志文件的。Oracle 系统确保在一组重做日志的归档操作完成之前不会重新使用该组重做日志。在 Oracle 数据库中归档操作一般是自动执行的。利用这些归档日志文件，Oracle 系统就能确保在遇到数据文件丢失或损坏后可以完全恢复数据库中的数据。

那么，怎样才能知道 Oracle 目前到底启动了多少个后台进程呢？您可以使用例 1-14 的查询语句获取这方面的准确信息，其中“where background = '1'”子句保证只显示后台进程。为了使显示清晰易读，您最好先使用例 1-12 和例 1-13 的 SQL*Plus 命令格式化一下显示输出结果。

例1-12

```
SQL> col program for a30
```

例1-13

```
SQL> set pagesize 35
```

例1-14

```
SQL> select pid, username, program
  2  from v$process
  3  where background = '1'
  4  order by program;
```

PID	USERNAME	PROGRAM
36	SYSTEM	ORACLE.EXE (CJQ0)
12	SYSTEM	ORACLE.EXE (CKPT)
6	SYSTEM	ORACLE.EXE (DBRM)
10	SYSTEM	ORACLE.EXE (DBW0)
8	SYSTEM	ORACLE.EXE (DIA0)

Note

5 SYSTEM	ORACLE.EXE (DIAG)
4 SYSTEM	ORACLE.EXE (GEN0)
11 SYSTEM	ORACLE.EXE (LGWR)
9 SYSTEM	ORACLE.EXE (MMAN)
16 SYSTEM	ORACLE.EXE (MMNL)
15 SYSTEM	ORACLE.EXE (MMON)
2 SYSTEM	ORACLE.EXE (PMON)
7 SYSTEM	ORACLE.EXE (PSP0)
19 SYSTEM	ORACLE.EXE (Q000)
25 SYSTEM	ORACLE.EXE (Q002)
20 SYSTEM	ORACLE.EXE (QMNC)
14 SYSTEM	ORACLE.EXE (RECO)
40 SYSTEM	ORACLE.EXE (SMCO)
13 SYSTEM	ORACLE.EXE (SMON)
3 SYSTEM	ORACLE.EXE (VKTM)
24 SYSTEM	ORACLE.EXE (W000)
已选择 21 行。	

例 1-14 查询显示的结果不但包括了我们所介绍过的 5 个必需的后台进程，而且还包括了许多其他的后台进程，但是并未包括归档后台进程 ARCn，这是因为我们的数据库目前是运行在非归档方式（模式）。其中，RECO 后台进程是用于分布式数据库的。

1.25 小　　结

在本章即将结束时请读者考虑一个问题：在数据库（数据文件）中所存的数据是否一致？也可以说成是数据库（数据文件）中所存的数据是否被提交？

要回答这个问题，首先要知道数据库当前的状态。如果数据库是处在正常关闭状态，数据库所存的数据当然是一致的。如果数据库是非正常关闭状态，则数据库中应该会存在不一致的数据。另外，如果数据库处在正常运行（开启）状态，则数据库中可能既存在一致的数据又存在不一致的数据。

数据库处在正常运行（开启）状态时，数据库中所存的数据是一致的，这一点很容易理解，怎么可能有不一致的数据呢？设想一下有某个用户发了如下的 DML 语句：UPDATE emp SET sal = sal * 0.9;（您知道这个 DML 语句的商业含义吗？可能是公司长期亏损，为了避免最终倒闭的厄运，公司要求全体员工“同舟共济”，集体减薪 10%），进一步假设 emp 表中有几十万条记录，而且该用户还有个坏习惯，他每次发了 DML 语句后既不提交也不回滚。可以想象经过一段时间在数据库高速缓冲区中的这些数据块就会自动地排到 LRU 队列的尾部。如果此时有一个 SQL 语句需要从数据文件中读入大量的数据到内存，而此时数据库高速缓冲区中已没有空闲的内存块（缓冲区）可用，因此 DBWR 进程要把在 LRU 队列尾部的没有提交的数据写到数据文件上。

另一个类似的问题是，数据库写进程是提交之前把在数据库高速缓冲区中的数据写到数据文件上还是在提交之后写？答案是可能在之前也可能在之后写。读者只要仔细回忆一下本章所介绍的有关内容就不难理解这一点了。

在 OCP 考试中有人统计过，与 SGA 和后台进程有关的问题大约占考试题的 20%以上。虽

然这些题变化多端，但是只要能真正地理解 SGA 和后台进程以及它们之间的关系是不难回答的。

本章主要是讲解与 Oracle 数据库管理系统相关的基本概念和原理。基本上没有实际操作，可能有些读者读起来比较乏味，但本章中的许多内容对理解以后章节的内容是至关重要的，希望读者把本章弄懂。

Note

1.26 您应该掌握的内容

在学习第 2 章之前，请检查一下您是否已经掌握了以下内容：

- ☑ 在数据库系统中什么是稀有资源。
- ☑ Oracle 服务器（server）的组成。
- ☑ Oracle 服务器的三种安装方式。
- ☑ Oracle 体系结构的轮廓。
- ☑ Oracle 实例（instance）。
- ☑ Oracle 引入实例的目的。
- ☑ Oracle 数据库（database）。
- ☑ Oracle 其他的几个关键文件。
- ☑ 怎样建立与实例（instance）的连接。
- ☑ 服务器进程和程序全局区（Program Global Area，PGA）。
- ☑ Oracle 执行 SQL 查询语句的主要步骤。
- ☑ Oracle 实例的系统全局区。
- ☑ 共享池（shared pool）的组成。
- ☑ 库高速缓存（library cache）的工作原理。
- ☑ 数据字典高速缓存（data dictionary cache）的工作原理。
- ☑ 怎样设置共享池。
- ☑ 数据库高速缓冲区（database buffer cache）的工作原理。
- ☑ 重做日志缓冲区（redo log buffer）的工作原理。
- ☑ Oracle 执行 UPDATE 语句的步骤。
- ☑ 怎样设置内存缓冲区的大小。
- ☑ 怎样获取内存缓冲区信息。
- ☑ 重做日志写进程的工作原理。
- ☑ 快速提交（fast commit）技术。
- ☑ 数据库写进程（DBWR/DBWn）的工作原理。
- ☑ 系统监督进程（SMON）的工作原理。
- ☑ 进程监督进程（PMON）的工作原理。
- ☑ 引入校验点（checkpoint）事件和校验点进程的原因。
- ☑ 校验点进程的工作原理。
- ☑ 引入归档日志文件和归档日志（ARCH/ARCn）进程的原因。
- ☑ 归档日志进程的工作原理。