

## 第6章 Oracle 实例结构

本章要点：

- 介绍
- 定义实例
- 创建实例
- Oracle实例的组成
- 事务剖析
- 监控实例

### 6.1 介绍

当有人谈到Oracle数据库时，很可能指的是整个 Oracle数据库管理系统（DBMS）；但是作为一个专业的Oracle数据库人员，你必须认识到数据库和实例二者之间的重大区别——一个经常使非Oracle的系统管理人员混淆的区别。本章将探讨 Oracle实例的结构和构造，在下一章将继续从更深层次探讨 Oracle关系型数据库管理系统（RDBMS）。为了避免混淆，RDBMS这个术语用来描述由Oracle数据库和实例组成的整个数据管理服务器。本章还将详细地讨论实例何时和如何产生。

### 6.2 定义实例

为提供Oracle客户所期望的不同程度的服务、灵活性与性能，数据库的许多工作由实例完成，实例是一系列复杂的内存结构和操作系统进程。除非使用并行 Oracle服务器选项，否则每个Oracle数据库都有一个实例与之相关，一个数据库被唯一的一个实例装载。实例结构允许RDBMS同时对来自多个用户的不同种类事务的请求提供服务，与此同时提供一流的性能、容错性、数据的一致性和安全性。

注意 本章定义的术语“进程”是指在没有用户干预的情况下正在运行的任务。你的操作系统可能将之称作“进程”，或者使用其他术语，例如任务、作业、线程和其他类似的术语。

在UNIX实现多任务操作系统后，实例是松散的结构方式。在一起工作的离散的进程在实现实例的目标的RDBMS中完成指定的任务。每一个进程都有各自的一个内存块，该内存块用于保存私有变量、地址堆栈和其他运行时的信息。进程间使用公共共享区并在公共共享区内完成它们的工作。公共共享区是能够在同一时间内被不同程序和不同进程读写的一块内存区。该内存块称为系统全局区（SGA）。

注意 因为SGA驻留在一个共享内存段中，所以它经常被称作共享全局区。

你可以认为后台进程就像数据库的手，直接处理数据库的组件；你也可以认为 SGA就像大脑，在必要时间间接地调度手处理它们的信息与存储检索。SGA参与发生在数据库中全部的

信息和服务器的处理。

注意 单用户的Oracle配置（例如 Personal Oracle Lite），不使用多进程执行数据库的功能。相反，所有的数据库功能由一个Oracle进程完成。由于这个原因，单用户也称为单进程Oracle。

### 6.3 创建实例

打开一个Oracle数据库包括以下三步：

- 1) 创建一个Oracle实例（非安装阶段）。
- 2) 由实例安装数据库（安装阶段）。
- 3) 打开数据库（打开阶段）。

Oracle实例在数据库启动的非安装阶段创建。当数据库经过非安装阶段时，读取 init.ora 参数文件，启动后台进程，初始化系统全局区（SGA）。init.ora 文件定义了实例的配置，包括内存结构的大小和启动后台进程的数量和类型等。实例名根据环境变量 Oracle\_SID 设置，它不一定要与打开的数据库名称相同（但是习惯上通常如此）。下一阶段称为安装阶段。init.ora 文件中的控制文件参数值决定数据库的安装实例。在安装阶段，读取控制文件并使其成为可访问的，可以对控制文件内存储的数据进行查询和修改。最后的阶段就是打开数据库。在这一阶段，其名字存储在控制文件中的数据库文件以排它使用方式被实例锁定，使数据库能够被普通用户访问。打开是数据库的正常操作状态。在数据库打开之前，只有 DBA 能访问数据库，且只能通过服务器管理器对其进行访问。

为了改变数据库的操作状态，必须作为内部连接到数据库，或拥有 SYSDBA 特权。当数据库从关闭状态到打开状态时，你可以明确地单步调试数据库的每一个操作状态，但当关闭数据库时，只能从当前运行状态转到完全关闭状态。例如，可以在服务器管理器工具中执行 STARTUP NOMOUNT 命令，这将使数据库处在非安装阶段，接下来可以运行 ALTER DATABASE MOUNT 或者运行 ALTER DATABASE OPEN 命令以单步调试到操作阶段。无论是在何种操作状态下，如果执行 SHUTDOWN 命令，将完全关闭数据库。例如，数据库不能从打开状态转到安装状态。

没有安装数据库的实例被称为空闲的——它使用内存，但不做任何工作。一个实例只能唯一地与一个数据库连接，而且除非使用并行服务器，否则对一个数据库也只分配一个实例。实例是数据管理的核心——它做所有的工作，而数据库存储所有的数据。

### 6.4 Oracle实例的组成

图6-1是Oracle实例的可视化表示。以下是对其各不同部分的解释。

许多参数和现成的技术能够帮你配置实例，以最好地支持你的应用和要求。配置实例对象以获得最高性能在多数情况下是一个试验和错误过程——你可以使用可能的参数值启动，但是只有经过一段时间的监控才会得出所有设置和变量的最好的可能组合。

配置实例参数包括改变必要的 init.ora 参数和在数据库状态之间切换（打开、关闭数据库）。在 init.ora 文件中有大量的参数，这些参数有些是无文档说明的。虽然你不应该改变或者增加你不熟悉的参数，但是可以查询内部表 x\$kspspi 以查看数据库所有可能的初始化参数。ksppinm 和 ksppdesc 列分别给出参数名和对参数的简要描述。

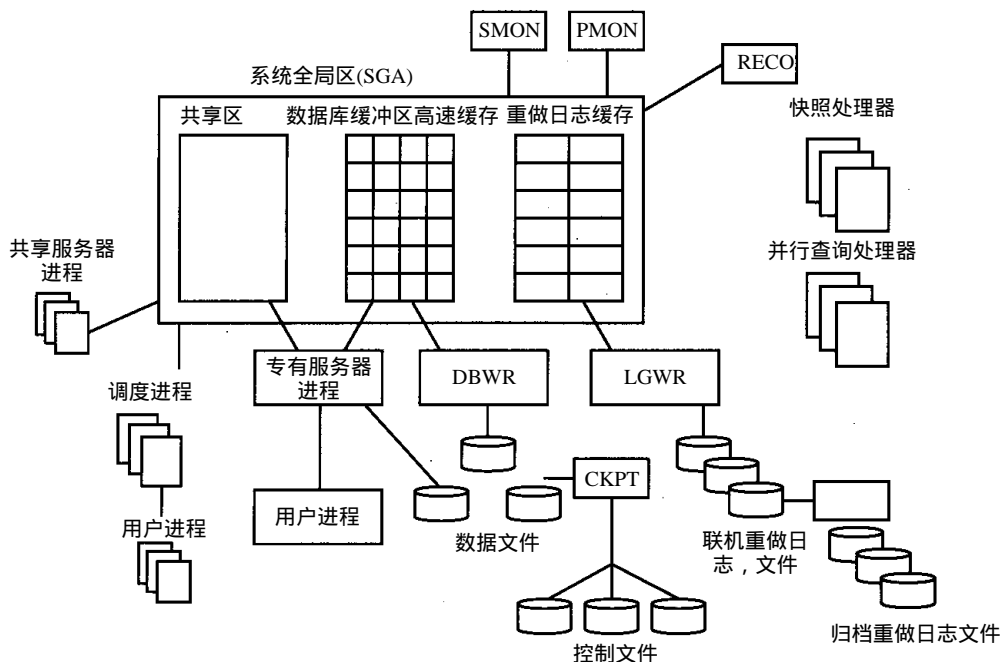


图6-1 Oracle实例是一个复杂的内存与后台进程的交互体

注意 改变初始化参数而没有清楚地理解这种改变可能的后果，是非常危险的。有许多参数纯粹是出于单纯的诊断原因而存在的，改变它能导致你的数据库处于非同步状态或者崩溃状态。无文档说明的参数以下划线开头命名。除非你有足够的把握，否则不要轻易增加或改变init.ora文件的关键字或值。

对于大多数部分，实例的配置主要关心在 SGA 中的对象，你会发现你的数据库配置与调整时间主要花在这些结构上。但是，与后台进程相关的问题和配置选项也需要进行解释。

#### 6.4.1 系统全局区

系统全局区是实例的主要部分。它含有数据维护、SQL 语句分析与重做缓存所必须的所有内存结构。系统全局区的数据是共享的，也就是说，多个进程可以在同一时间对 SGA 中的数据进行访问和修改。所有数据库操作都使用包含在 SGA 中某点上的结构。正如前所述，在数据库非安装阶段，当实例被创建时，分配 SGA；当实例关闭时，释放 SGA。

SGA 组成如下：

共享池。

数据库缓冲区高速缓存。

重做日志缓冲区。

多线程服务器 (MTS) 结构。

在下面几节对它们做出解释。

##### 1. 共享池

共享池 (参见图 6-2) 包括库高速缓存、数据字典高速缓存和服务器控制结构 (例如数据字典字符集)。库高速缓存存储已提交给 RDBMS 的 SQL 语句文本、分析过的格式与执行计划，

以及已被执行的PL/SQL包头与过程等。数据字典高速缓存存储用于分析 SQL语句的数据字典行。

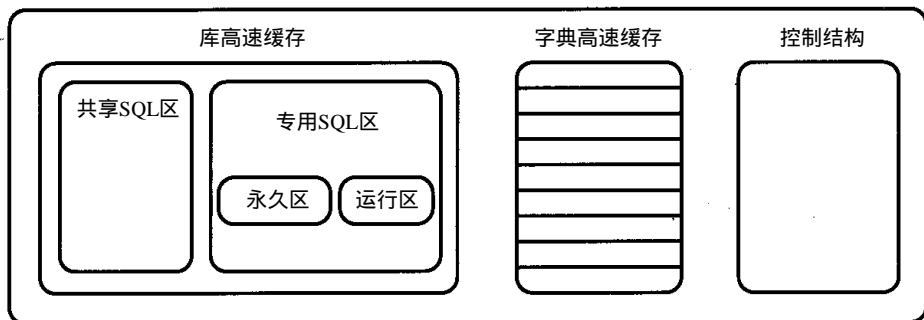


图6-2 当分析和执行SQL语句时使用的共享池高速缓存信息

Oracle服务器用库高速缓存来提高执行 SQL语句的性能。当一条 SQL语句提交时，服务器首先查找高速缓存，查看相同的语句是否已被提交或缓存过。如果有，Oracle使用存储的语法分析树和执行路径来执行该语句，而不是从头重建这些结构。尽管使用这个特性不会影响特定的查询的执行，但是使用存储代码可以获得明显的性能提高。

**注意** 对于使用以前缓存的版本的SQL语句，它必须在所有方面都与缓存版本完全相同，包括标点符号和字母的大小写。Oracle对语句中的文本使用哈希算法识别语句——为了可以使用缓存的版本，语句文本产生的哈希值必须对当前语句和缓存的语句是唯一的。

库高速缓存包括共享和专用 SQL区。共享 SQL区包括 SQL语句语法分析树和执行路径，而专用 SQL区存储特定的会话信息，例如捆绑变量、环境和会话参数、运行堆栈和缓冲区等。专用 SQL区在每个事务初始化时创建，在与专用 SQL区相关的游标关闭时被释放。一个用户会话能够一次打开的专用 SQL区的数量由 init.ora 参数 OPEN\_CURSORS 决定。使用这两个结构，Oracle服务器能够重用一条 SQL语句的所有执行的所有共同信息。与此同时，可以从专用 SQL区中查询执行的特定会话信息。然而值得注意的重要一点是，包含在用户的进程全局区（PGA）中的用户全局区（UGA）的特定会话信息（包括专用 SQL区），只在使用多线程服务器实例时才由 SGA 拥有，否则，它由专用服务器拥有。

**注意** 游标使用过程中并不关闭游标的应用会继续为应用分配越来越多的内存，部分原因是为每个打开的游标分配了专用 SQL区。

库高速缓存中的专用 SQL区可更进一步分为永久区和运行区。永久区中包含合法的信息，并可应用于 SQL语句的多个执行中，而运行区中仅包含正在被执行的 SQL语句的数据。

字典高速缓存含有 RDBMS引擎分析 SQL语句所使用的数据库字典信息。在这个区域中含有段信息、安全性、访问权限和在可用的自由存储空间等信息。

共享池的大小取决于 init.ora 文件参数 SHARED\_POOL\_SIZE，它是以字节为单位的。你必须将这个值设得足够大，以确保有足够的可用空间来装载和存储 PL/SQL块和 SQL语句。共享区经过长期装卸和卸载数据对象会产生许多碎片，如果在共享池中没有一个足够的连续空间用来装载目标数据，会产生错误。解决这个问题的捷径是运行 SQL命令 ALTER SYSTEM FLUSH SHARED\_POOL。但是如果在数据库操作时，经常遇到共享池错误，你必须增大共享

池。

## 2. 数据库缓冲区高速缓存

数据库缓冲区高速缓存是影响整个数据库系统运行的重要因素之一。数据库缓冲区高速缓存是由与 Oracle 块相同大小的内存块组成。所有 Oracle 操作的数据在使用前被装入到数据库缓冲区高速缓存中。数据的更新在内存块中完成。由于这个原因，合适大小的缓冲区高速缓存是至关重要的。访问内存的速度要比访问磁盘快几百倍。在 OLTP 环境下，大多数数据操作应该完全在内存中进行，使用早已被装入高速缓存中的数据库块。

Oracle RDBMS 根据最近最少被使用（LRU）列表将数据清出缓冲区高速缓存。最近最少使用列表追踪记录哪些数据块被访问和访问的频繁程度。当一个块被访问或者被查询写了缓冲区时，它被写入最近最多被使用（MRU）列表的末端，当 Oracle 服务器在缓冲区中需要更多空间来从磁盘读入一个数据块时，它去访问 LRU 列表，以确定可以清出哪些块，在 MRU 端最末端的那些块最先被移走。使用这种方法，保证最频繁使用的块保存在内存中。

注意 LRU 装载规则的例外是，通过全表扫描被访问的数据自动地放在 LRU 列表的底端。

可以通过指明表为 CACHE 忽略这个规则。

被修改过的缓冲块称为脏的，它们被记录在脏列表中。脏列表追踪记录所有在内存中被修改而又尚未写入磁盘中的数据。当 Oracle 接收到修改数据的请求时，对高速缓存中的块进行数据修改，同时写入重做日志中，然后该块被放入脏的列表中，对这些数据的随后访问从高速缓存中读取改变的数据的新的值。

Oracle 服务器使用延迟、多个块写以减缓磁盘 I/O 的冲突。这意味着更新一块数据，并不立即更新数据文件中的数据。RDBMS 等到以下时刻才将改变的数据刷新写入数据文件：预定数量的数据块被改变、要求收回缓冲区中的空间以装入新数据、发生一个检查点或者超过数据库后台写的时间。当数据库后台写进程被通知执行一个缓冲区高速缓存写操作时，它将一组块中的数据写入数据文件中。

配置缓冲区高速缓存的关键是保证分配正确总量的内存，以优化缓存数据。将有可能获得的内存资源都分配给缓冲区高速缓存是不必要的；然而，在多数计算机应用中，内存分配持续增长到某一点后性能增长明显降低。这时增加内存以获得一个增长的更好的缓存命中率方面的投入应减少。分配给缓冲区高速缓存的内存能够被更好地利用在其他地方，例如其他的 Oracle 内存结构中。

DB\_BLOCK\_SIZE 和 DB\_BLOCK\_BUFFERS 是决定缓冲区高速缓存大小的两个初始化参数。DB\_BLOCK\_SIZE 参数用于在数据库创建时，设置 Oracle 块大小。这一点将在第 7 章“探讨 Oracle 环境”中进行详细解释。DB\_BLOCK\_BUFFERS 参数决定分配给缓冲区高速缓存的块的数量。用 DB\_BLOCK\_SIZE 乘以 DB\_BLOCK\_BUFFERS 就得出缓冲区高速缓存的内存总数（以字节为单位）。

## 3. 重做日志缓冲区

重做日志缓冲区用于在内存中存储未被刷新写入联机重做日志文件的重做信息。它是循环使用的缓冲区，这意味着从顶端到底端填充信息，然后又返回到缓冲区的起始点。当重做日志缓冲区填满时，将它的内容写入联机重做日志文件。

重做日志缓冲区的大小是由 LOG\_BUFFER 初始化参数决定，以字节为单位，决定在内存中保留多少空间缓存重做日志项。如果这个值设置得过低，进程之间相互竞争，日志写入

(LGWR) 进程读出和写入缓存, 有可能会导导致性能问题。然而, 这在总数中是很少的, 但却是数据库最活跃的情形, 可以用 V\$SYSSTAT视图进行监控。使域 name等于redo log space requests, 查询V\$SYSSTAT视图value域, 它表明用户进程等待重做日志缓冲区所花费的时间。

为强迫重做日志顺序写入, Oracle服务器使用门控制对缓存的访问。门是一个 Oracle进程对一个内存结构的锁定——在概念上与文件锁定和行锁定类似。一个进程必须持有重做分配门, 才能写入重做日志缓冲区。当一个进程持有分配门时, 其他任何进程都不能使用这个分配门写入重做日志缓冲区。

Oracle服务器使用LOG\_SMALL\_ENTRY\_MAX\_SIZE初始化参数限制一次写入的重做的总量。这个参数以字节为单位, 其缺省值随操作系统和硬件不同而不同。对具有多个 CPU的服务器而言, Oracle服务器不允许使用重做分配门所书写的重做日志项所需空间比参数 LOG\_SMALL\_ENTRY\_MAX\_SIZE大。相反, 进程必须持有一个重做复制门。可获得的重做复制门的数量等于LOG\_SIMULTANEOUS\_COPIES初始化参数的值。LOG\_SIMULTANEOUS\_COPIES的缺省值是系统中CPU的数量。使用重做复制门, 多个进程能同时写入重做日志缓冲区。

可以使用V\$LATCH动态性能视图监控重做分配门与重做复制门(参见第18章“调整内存”, 以获得更多调整重做门的信息)。

#### 6.4.2 Oracle 后台进程

在任意瞬间, Oracle数据库可以处理许多行信息、处理几百个同步用户请求、进行复杂的数据操作,与此同时提供最高水平的性能和数据的完整性。为了完成这些任务, Oracle数据库将一项大的工作分散到多个程序中, 其中每个程序的大部分操作都是相互独立的, 并扮演一个特定的角色。这些程序称为 Oracle后台进程, 是有效地处理施加在 Oracle数据库的多个操作的有效方法。完全理解后台进程和它们担负的任务, 将有助于分析性能问题、指出瓶颈和诊断数据库中的故障点。

**注意** 在NT服务器上, 后台进程的执行是作为Oracle Service的多线程实现的, 它允许Oracle进程更有效地使用共享内存地址空间, 从而减少了NT操作系统处理Oracle操作时的环境改变。

Oracle后台进程如下:

系统监控和进程监控进程 (SMON和PMON)。

数据库写进程 (DBWR)。

日志写进程 (LGWR)。

调度进程 (Dnnn)。

归档进程 (ARCH)。

检查点 (CKPT)。

恢复进程 (RECO)。

快照进程 (SNPn)。

锁进程 (LCKn)。

并行查询进程 (Pnnn)。

用户和服务器进程 (Snnn)。

它们之中包括处理数据库上用户事务的用户和服务进程，以及执行对数据库的并行操作的并行查询进程。尽管它们不能归类于 Oracle 后台进程，但是理解它们在 Oracle 环境中所扮演的角色非常重要。下面对这些进程分别进行讨论。

### 1. 系统监控和进程监控

由于种种原因，对 Oracle 数据库的连接可能会发生崩溃、挂起或其他非正常终止。终端用户可能关闭他们的客户机而没有从数据库应用中退出，或者与数据库无关的网络或系统故障可能造成一个数据库自动作业失败。Oracle 服务器必须有能力去处理由于上述原因而引起的失败。

系统监控和进程监控都是自动解决数据库系统问题的后台进程。进程监控（PMON）自动清除中断或失败的进程，包括清除非正常中断的进程留下的孤儿会话、回滚未提交事务、释放被断开连接的进程占有的锁、释放被失败进程占有的系统全局区（SGA）资源，它同时监控服务器和调度进程，如果它们失败则自动重启它们。

系统监控（SMON）担任一个比较小但非常重要的角色。从数据库启动开始，SMON 就作为恢复自动实例的进程。如果最后的数据库关闭不干净，SMON 自动向前回滚正在进行的操作，并回滚尚未提交的事务。SMON 也是管理某些数据库段的进程，收回不再使用的临时段空间，并自动合并和数据文件中相邻的自由空间块。

注意 SMON 仅在当创建表空间或表时的缺省存储参数——pctincrease 不是 0 时，合并表空间中的自由空间。如果想让 SMON 自动地处理这个操作，至少将 pctincrease 设置为 1。

SMON 和 PMON 是两个必需的后台进程。如果它们之中的任意一个在数据库启动时失败，数据库将不能启动。

### 2. 数据库写

数据库写进程（DBWR）负责将缓冲区中脏的数据块写入到数据文件中。DBWR 不是在每一数据块被修改后立即写入数据文件，而是一直等待，直到满足一定标准后，才成批地读脏列表，并将在脏列表中发现的所有块刷新写入数据文件。这提供了高级别的性能，并最小化数据库输入/输出约束的范围。

当下列情况发生时，数据库刷新脏的块：

- 1) 发生一个检查点。
- 2) 脏列表的长度达到 init.ora 文件中 DB\_BLOCK\_WRITE\_BATCH 参数值的一半。
- 3) 使用的缓冲区数量达到 init.ora 参数 DB\_BLOCK\_MAX\_SCAN。
- 4) DBWR 后台进程发生超时（大约每 3 秒）。

配置 DBWR 后台进程是相当简单的，对于小型或中型数据库而言，它的缺省设置值在许多情况下是足够的。然而，对于较大的、经常活动的或经常有专门需求的专用数据库，必须手工配置一些 DBWR 后台进程参数。

在大多数安装中，有一个 DBWR 进程处理所有数据库的所有写入活动。但是当你发现这个 DBWR 进程不能满足数据库要求时，你可以启动一个以上的 DBWR 进程。在 init.ora 文件中 DB\_WRITES 的缺省值是 1，它设置在启动时创建的 DBWR 进程数。在多数情况下，仅当安装的 Oracle 服务器的操作系统不支持同步的 I/O 时，才决定使用多个 DBWR 进程。一旦遇到这种情况，应该创建多个 DBWR 进程。建议你使用与存储数据文件的物理磁盘一样多的 DBWR 进程；另一建议是将这个数量设为与数据库中数据文件的数量相等。增加或减少 DBWR 进程数

进行试验，直到能够最好地满足你的执行。

数据库缓冲区高速缓存也使用门控制对内存结构的访问。LRU门控制在缓冲区高速缓存中缓冲区的替换。有具有多个CPU的极为活动的服务器上，可能会发生对这些门的争用。如果发生这种情况，设置DB\_BLOCK\_LRU\_LATCHES参数值为与为缓冲区高速缓存创建的门的数量相等。这个数值不能超过CPU数量的两倍，并被自动设置为系统中的CPU数量。

另一个影响DBWR行为的init.ora参数是DB\_BLOCK\_CHECKPOINT\_BATCH。这个参数设置在每个检查点DBWR写入的块的最大量（参看下面的检查点进程以得到更详细的信息）。增大这个参数，可以减少DBWR刷新缓冲区的次数。然而将这个数值增加得过大，当DBWR最终刷新缓冲区时，可能会产生不能接受的延迟。

要记住的第三个参数是DB\_BLOCK\_CHECKSUM。这是一个布尔参数。启用这个参数时，导致每个数据库块被写入时附加一个校验和值。当随后读取该块时，计算校验和值并与存储在数据库中的数值进行比较，如果值不同，将产生错误。当查找数据损坏问题时，这是一个有价值的参数，但是不应该在全部时间都启用这个值，因为对每一个I/O计算并存储校验和会影响性能。

### 3. 日志写

日志写（LGWR）是第四个也是最后一个必须的后台进程。LGWR是将在系统全局区中重做日志缓冲区重做日志条目写入到联机重做日志文件的进程。LGWR执行写入操作的条件是：发生提交、到达LGWR非活动时限、重做日志缓冲区满度达到三分之一或者DBWR在检查点完成数据缓存块的刷新。如果LGWR在完成代表其他用户的提交刷新缓冲区之前，有一个或多个用户执行提交时，LGWR也能同时处理多用户提交事务。

值得注意的重要一点是：直到Oracle在LGWR完成将重做信息从重做缓冲区刷新到联机重做日志文件之后，Oracle才认为一个事务已完成。在LGWR成功地将重做日志项写入联机重做文件时（并不是改变数据文件中的数据时），将一个成功码返回给服务器进程。

LGWR进程极少能引起数据库操作问题。另外，可以用于配置LGWR进程的选项很少，大多数的配置涉及重做日志缓冲区和支持该缓冲区的内存结构，而不是LGWR进程自身。

然而，有一点例外，LGWR进程处理的次要任务是，执行实施数据库检查点所需要的操作。除非检查点进程被激活，否则LGWR进程完成这一任务。检查点导致LGWR和DBWR都要花费进程和I/O时间。检查点间隔时间越短，发生数据库故障时需要的恢复时间越短，同时减少了必须执行每一检查点所需的工作。当你决定正确的检查点间隔时，你必须权衡所有这些因素。有几个控制发生数据库检查点的参数。

LOG\_CHECKPOINT\_INTERVAL和LOG\_CHECKPOINT\_TIMEOUT是能够改变检查点间隔的两个参数，也是触发数据库检查点所必须的时间或条件。当设置LOG\_CHECKPOINT\_INTERVAL参数时，当一定数量的操作系统块（不是Oracle块）写入重做时，引起数据库中的一个检查点被触发。当设置LOG\_CHECKPOINT\_TIMEOUT参数时，在这个参数指定的时间间隔（以秒为单位）发生一个检查点。

对这些参数的使用必须要小心。如果使用LOG\_CHECKPOINT\_INTERVAL参数，应该设置它以使触发检查点的操作系统块的数量与重做日志组的大小相对。记住，当一个重做日志组写满时，一个检查点被触发。要注意的是不要设置不必要的检查点，或者迫使不需要的检查点发生。例如，如果一个重做日志组大小为3MB，而LOG\_CHECKPOINT\_INTERVAL设置

为2.5MB。当有2.5MB的数据写入重做日志时，参数 LOG\_CHECKPOINT\_INTERVAL 的值导致发生一个检查点。另外，当重做日志组写满时（仅在又写入 0.5MB的数据后），发生另一个检查点。事实上，这两个检查点将相继发生。

你还可以通过相应地设置重做日志组的大小控制检查点发生的频繁程度。如果你设置你的日志大小使得每小时发生一次日志切换，那么在重做日志组切换后，在一小时内，你将只有一个检查点。但是如果你的日志组大小设置为每 5 分钟发生一个检查点，你将浪费大量的进程活动和I/O次数以执行相关的检查点。

最后一个使用的参数是布尔值 LOG\_CHECKPOINTS\_TO\_ALERT。每当检查点发生时，它为数据库在 alert.log 文件中设置一个标记，并用于试图指出确切的检查点间隔。

#### 4. 调度进程

正如前面所提到的，服务器进程既可以是一个用户进程专有的，也可以在多个用户进程之间共享。使用共享服务器要求配置多线程服务器，在第 37 章“安装和配置 OAS”中将讨论这个问题。当使用共享服务器进程时，至少必须存在一个调度进程（Dnnn），在环境需要时也可能有多个调度进程。调度进程将用户请求传送到系统全局区的请求队列，并将服务器的响应信息返回给正确的用户进程。

使用 init.ora 参数的一个数控制调度进程的数量。参数 MTS\_DISPATCHERS 指定调度进程使用的协议及开始使用该协议的调度进程的数量。多协议组可以使用多个 MTS\_DISPATCHERS 行进行配置。一个典型的 MTS\_DISPATCHERS 行可能像这样：

```
MTS_DISPATCHERS = "tcp, 4"  
MTS_DISPATCHERS = "spx, 2"
```

多协议组也可以在相同的 MTS\_DISPATCHERS 参数内中像这样配置：

```
MTS_DISPATCHERS = ("tcp, 4", "spx, 2")
```

参数 MTS\_MAX\_DISPATCHERS 控制 RDBMS 允许的调度进程的最大数量（参见第 34 章“高级安全选项”以获取关于 MTS 服务配置的详细信息）。

#### 5. 归档进程

归档进程（ARCH）负责将全部联机重做日志复制到归档重做日志文件。这仅在数据库运行在归档模式（ARCHIVELOG）下才发生。归档模式要求时间点恢复，它也允许“热”备份。当 ARCH 正在复制归档重做日志时，没有其他进程能够写入这个重做日志。记住这一点非常重要，因为重做日志是按顺序循环使用的。如果数据库需要转换重做日志，但是 ARCH 还正在按其顺序复制下一个日志，所有数据库的活动将终止，直到 ARCH 完成。还要注意如果归档由于某些原因不能完成复制日志，它将等待直到引起不能写入的错误得到解决为止。

非常值得注意的是在 init.ora 文件中 ARCHIVE\_LOG\_START 参数必须设置为 TRUE，当数据库启动时，才会自动开始归档。设置数据库处于归档模式并不足以导致 ARCH 自动启动。如果你确实设置了归档模式，但不自动启动 ARCH，当所有联机重做日志写满时，数据库将会挂起，等待你手工归档联机日志。

#### 6. 检查点进程

检查点进程（CKPT）是可选的后台进程，执行 LGWR 进程通常会执行的检查点任务——即用当前版本信息更新数据文件和控制文件头。当有经常性的检查点发生、频繁的日志切换或在数据库中有多个数据文件时，启用这个进程来减少 LGWR 的工作量。

设置CHECKPOINT\_PROCESS参数为TRUE可以启用CKPT进程。当CKPT进程运行时，所有与说明检查点有关的其他参数也必须为TRUE。

**警告** 在Oracle 8.x中，CHECKPOINT\_PROCESS参数已被废弃，因为它已被集成入RDBMS中并设置为TRUE。如果在你的Oracle8.x的init.ora文件中包括这个参数，你的实例启动将会失败。

### 7. 恢复进程

恢复进程 (RECO)，负责在分布式数据库中恢复失败的事务。当分布式事务配置数据库时 (也就是，设置init.ora参数DISTRIBUTED\_TRANSACTIONS值大于0)，RECO会自动启动。当分布式数据库中有悬而未决的事务时，RECO在很少或没有DBA干预的情况下运行。当数据库连接成功时，RECO试图连接远程数据库并解决悬而未决事务 (参见第40章“分布式数据库管理”了解关于RECO和两阶段提交的详细信息)。

### 8. 快照进程

快照进程 (SNPn) 处理数据库快照的自动刷新，并通过DBMS\_JOB包运行预定的数据库过程。init.ora参数JOB\_QUEUE\_PROCESS设置启动的快照进程数，参数JOB\_QUEUE\_INTERVAL决定快照进程在被唤醒以处理挂起的作业或事务之前休眠的时间 (以秒为单位)。

### 9. 锁进程

在并行服务器环境中，多个实例安装在同一个数据库上，锁进程 (LCKn) 负责管理和协调每个实例占有的锁。可以分配给安装在并行服务器上的每个实例1~10个锁进程，每个实例必须具有相同的数目。在非并行服务器环境中，这个进程不起作用。参见第39章“并行服务器管理”以获得锁进程的详细信息。

### 10. 并行查询进程

并行查询进程称为Pnnn。根据数据库的活动和你的并行查询选项的配置，Oracle服务器启动和停止查询进程。这些进程涉及并行索引的创建、表的创建及查询。启动的进程的数量与参数PARALLEL\_MIN\_SERVERS指定的数量相同，但决不能超过参数PARALLEL\_MAX\_SERVERS指定的进程数。

配置并行查询进程的详细信息参见第38章“并行查询管理”。

### 11. 用户和服务器进程

应用和工具通过用户进程访问RDBMS。用户进程连接到服务器，既可使用专用方式，也可以在多个用户进程之间使用共享方式 (在多线程服务器中)。服务器进程分析和执行用户提交的SQL语句，并将结果集返回给用户进程。这个进程也从数据文件将数据块读入数据库缓冲区。

每个用户进程被分配一部分内存区，称为进程全局区 (PGA)，PGA的内容根据所连接的数据库模式而有所不同。当一个用户进程通过专用服务器方式连接数据库时，用户的会话数据、堆栈空间和游标状态信息存储在进程全局区中。用户的会话数据包括安全和资源使用信息；堆栈空间含有为用户会话指定的本地变量；游标状态区包括运行时的游标信息、返回的行和游标返回的代码。但是如果用户进程通过共享服务器进程方式进行连接，游标和会话信息被存储在系统全局区 (SGA) 中。尽管对整个数据库而言，这并不增加对内存空间的要求，但是它需要一个更大的系统全局区来存放这些附加的会话信息。

## 6.5 事务剖析

为更好地理解前面讨论的实例组件是如何相互交互的，看一个典型的事务在实例结构中的移动。

当一个用户会话使用 SQL\*Net 驱动程序连接到服务器会话时，开始一个事务。这个连接可以使用它自己的一个服务进程进行专用连接，或通过调度进程处理的一个共享连接。服务器会话对传递给它的 SQL 语句进行哈希构造，并将该语句的哈希数与已经保存在共享 SQL 区中的语句的哈希数进行比较，如果在共享池中发现有完全一样的语句，使用该语句早已存储的做过语法分析的形式与执行计划。如果在共享池中没有发现匹配语句，服务器进程对这些语句进行语法分析。

接下来，服务器会话查看在数据库缓冲区中是否已经存储了完成该事务所必须的数据块。如果在缓冲区中没有，服务器会话就从数据文件中读取必要的数据库块，将它们拷贝到缓冲区中。如果事务是一个查询，服务器会话将查询的结果返回给用户进程（执行必要次数的数据库块读和拷贝以返回所有数据）。

对一个修改数据的事务，有更多的步骤需要做。例如，假设事务是一个更新。在将必要的数据库块读入缓冲区高速缓存中之后，修改内存中的数据库块。修改的缓存块标记为脏的，并将它们放入脏列表中。还产生重做信息，并将重做信息存储在重做日志缓存中。

事务继续进行，直到发生以下几个事情中一个。如果事务是相对短期的（例如，对一行销售数据的修改），事务完成，用户提交，发出信号给 LGWR 进程让它将重做日志缓冲区刷新到联机重做日志文件。如果事务相对持续时间很长和很复杂，下列事情有可能发生：

- 产生的重做引起重做日志缓冲区写满三分之一的空间，这会触发 LGWR 进程刷新重做日志缓冲区。
- 放入脏列表中的块数达到门限的长度，这会触发 DBWR 进程将数据库缓冲区所有脏列表项刷新写入到数据文件中，它反过来也使 LGWR 进程向磁盘刷新重做日志缓冲区。
- 遇到一个数据库检查点，这将触发数据库缓冲区高速缓存和重做日志高速缓存刷新。
- 可获得的缓冲区高速缓存中的自由缓存空间下降到规定的门限以下，这也会引起数据库高速缓存缓冲区的刷新。
- 产生一个不可恢复的数据库错误，这迫使该事务中断、回滚并向服务器会话报告错误。

当事务正在处理向重做缓存生成的重做并刷新时，联机重做日志逐渐被填满。当前日志被填满后，LGWR 进程开始写入下一个日志组；与此同时，归档进程将重做日志复制到磁盘或磁带。因为直到所有重做日志信息从重做缓冲区写入到联机重做日志之后，事务才被记录为执行成功的，所以 LGWR 和 ARCH 必须各自有能力在无错误的情况下完成自己的任务。

## 6.6 监控实例

在大多数情况下，系统全局区和后台进程在没有系统管理员干预的情况下运行。然而，有时会发生问题，这时必须能诊断和修复发生的问题。对 DBA 来说，有几种方法可以监控和追踪实例及其相关的结构。

### 6.6.1 使用追踪文件

查找关于实例的问题最好的地方是在进程自己的追踪文件中。根据特定的进程和所遇到

的错误，这些追踪文件被写在由参数 USER\_DUMP\_DEST 或者 BACKGROUND\_DUMP\_DEST 所指定的位置。当一个后台进程被终止或者非正常中断一个操作时，通常产生一个追踪文件，包含导致失败的错误信息、当前进程堆栈的转储、当前执行的游标和与问题有关的其他信息。尽管有些信息对 DBA 而言是有用的，但更重要的是收集这些追踪文件，并将它们提供给 Oracle 全球范围客户支持顾问，他们可能会帮助你诊断问题。他们具有可用的工具，能确切指出问题发生的位置。后台进程失败经常在数据库的 alert.log 文件中写入一项，或者写到它们各自的追踪文件，这些文件的位置由 init.ora 参数 BACKGROUND\_DUMP\_DEST 指定。

### 6.6.2 通过操作系统追踪

后台进程也能通过使用操作系统命令进行追踪。在 UNIX 环境中，每一个后台进程是一个分立的任務，因此可以分立追踪。通过查看操作系统的进程的内存和 CPU 的使用（通过使用 sar、ps、vmstat 和 top 等工具）指明性能问题和失控的查询通常是十分有价值的。有时解决挂起、中断服务器进程或用户进程的唯一方法是在操作系统级中断它们。然而，在试图修改或中断任何其他 Oracle 后台进程时，使用这种方法要非常小心，许多后台进程将会由于非正常中断而使整个数据库崩溃。

在 NT 服务器环境中，追踪后台进程很容易，这是因为全部的 Oracle 实例在 NT 操作系统作为一个称为服务的后台进程实现。各个后台进程作为属于这个服务的线程实现。尽管有大量工具可以用于在 NT 中追踪和监控进程的行为，但是线程管理工具还是相当不常见的。一个解决办法是使用与 NT OS 一起到货的 Performance Monitor 工具进行监控，它监控许多事情，包括属于这个服务的线程的内存的消耗和环境的切换。通过以下查询将 SPID 列从十进制转换到十六进制，可以将 NT 线程 ID 与 Oracle 端的后台进程进行匹配。

```
SELECT spid, name FROM V$process, V$bgprocess WHERE addr = paddr;
```

附录 B “Windows NT 上的 Oracle” 中有更详细的调整和追踪 NT 后台线程的信息。

### 6.6.3 使用 V\$ 表监控实例结构

DBA 可以使用许多动态性能视图显示实例的信息。当试图发现当前数据库实例的状态和与实例相关的故障问题时，这些视图有很大的价值。

#### 1. 监控数据库的连接

所有连接到实例的用户和后台进程都能够使用 V\$ 视图被监控。V\$process 视图显示所有连接到数据库的进程的信息，包括后台进程和用户进程；V\$bgprocess 含有所有可能的后台进程的列表和一个附加的 PADDR 列，PADDR 列包含后台运行进程的十六进制地址（对于那些没有运行的进程使用 00）。

在 V\$process 表中你可能感兴趣的列如表 6-1 所示。

表6-1 V\$process表的列

列	用 途
ADDR	进程的Oracle地址
PID	Oracle进程的ID号
SPID	操作系统进程ID号
USERNAME	操作系统进程拥有者

(续)

列	用 途
SERIAL#	Oracle进程序列号#
TERMINAL	操作系统终端标识符
PROGRAM	操作系统程序连接
BACKGROUND	1是后台进程，NULL是用户进程

V\$bkgprocess表中你可能感兴趣的列如表 6-2所示。

表6-2 V\$bkgprocess表中的列

列	用 途
PADDR	Oracle进程地址（与V\$process表的ADDR列相同）
NAME	后台进程名
DESCRIPTION	后台进程的描述
ERROR	错误状态代码（0是无错误）

你可以通过连结 V\$process表和V\$bkgprocess表显示所有运行的后台进程的地址和名称，如下所示：

```
SELECT spid, name
FROM V$process, V$bkgprocess
WHERE paddr(+) = addr;
```

连接到数据库的用户会话的信息存储在 V\$session视图中，V\$session视图包括许多列和大量能够进行访问的有价值信息。

V\$session视图中你可能感兴趣的列如表 6-3所示。

表6-3 V\$session视图的列

列	用 法
SID	会话标识符
SERIAL#	会话序列号
PADDR	父会话的地址
USER#	Oracle用户标识符（来自SYS.USER\$表）
USERNAME	Oracle用户名
COMMAND	这个会话正在处理的当前命令。要了解命令事务的号，应查看 sys.audit_actions表
STATUS	会话的状态（ACTIVE、INACTIVE、KILLED）
SERVER	会话具有的服务器连接的类型（DEDICATED、SHARED、PSEDUO或NONE）
OSUSER	进行连接的操作系统用户名
PROGRAM	进行到数据库连接的操作系统程序
TERMINAL	进行连接的数据库终端类型
TYPE	会话的类型（BACKGROUND或USER）SQL_HASH_VALUE和SQL_ADDRESS。 用于唯一标识当前正在执行的SQL语句

下面这个查询描述了已连接进程的重要信息。它也说明了进程视图之间相关的方式。

```
col bgproc format a6 heading 'BGProc'
col action format a10 heading 'DB Action'
col program format a10
col username format a8
col terminal format a10

SELECT
  b.name bgproc, p.spid, s.sid, p.serial#, s.osuser,
  s.username, s.terminal,
```

```
DECODE(a.name, 'UNKNOWN', '-----', a.name) action
FROM
  V$process p, V$session s, V$bgprocess b,
  sys.audit_actions a
WHERE
  p.addr=s.paddr(+) AND b.paddr(+) = s.paddr AND
  a.action = NVL(s.action, 0)
ORDER BY
  sid;
```

通过查询V\$access视图，能显示当前用户正在进行访问的数据库对象的信息，这对断定是第三方应用或无文档说明的过程正在做什么是有用的，对解决安全问题也是有用的。使用DBA帐户运行一个涉及到安全性问题的应用或过程，你能决定安全性应该给予哪个对象。

最后，V\$mmts视图含有共享服务器进程的追踪信息。这个视图包含的列有最大的连接数、服务器启动、服务器中止和服务器的最高水位。

## 2. 监控共享SQL区

有时能够查看RDBMS引擎内部和正在被执行SQL语句是有用的。视图V\$sqlarea包含共享SQL区中的SQL语句的信息，包括被执行SQL语句的文本、访问该语句的用户数、执行语句时访问的磁盘块和内存块以及其他信息。

注意 在V\$sqlarea中disk\_reads和buffer\_gets列追踪从磁盘中读入的块数和从缓冲区中读入的块数。这两列是找出使用大量数据库资源的查询的简便方法。

V\$open\_cursor视图对研究尚未关闭的游标是有用的。下面的查询显示一个给定的用户SID打开的所有游标：

```
SELECT b.piece, a.sql_text
FROM V$open_cursor a, V$sqltext b
WHERE
  a.sid = &SID and
  a.address = b.address and
  a.hash_value = b.hash_value
ORDER BY
  b.address, b.hash_value, b.piece asc;
```

视图V\$sqltext也能用于确定传递给数据库引擎的SQL语句。V\$sqlarea视图仅存储SQL语句的前80个字符，而视图V\$sqltext保存完整的SQL语句。除了在SQL语句中适时加入换行符之外，V\$sqltext\_with\_newlines视图与V\$sqltext视图是完全相同的。

注意 在V\$sqltext视图中，SQL语句是分块存储的。要获取整个语句，你必须按照PIECE列排序获取SQL语句的所有部分。

## 3. 监控系统全局区

有两个V\$视图提供了对系统全局区操作的信息。V\$sga视图显示每一个SGA主要组成部分的大小（以字节为单位），包括重做日志缓冲区、数据库缓冲区高速缓存和共享池。V\$sgastat视图包括更多的相关信息，在这个视图中你能查找在系统全局区包含的每个单独的内存结构的大小，包括为堆栈空间、PL/SQL变量和堆栈预留的内存。你也可以查询这个视图，找出SGA区中可获得的自由内存的总数：

```
SELECT bytes FROM V$sgastat WHERE name = 'free memory';
```

## 4. 监控库和字典缓冲区

有两个视图包含库和数据字典缓冲区的信息。V\$librarycache视图中包含库缓冲区中的每

种类型对象的性能信息；V\$rowcache视图中包含数据字典缓冲区的性能信息（参见第 38章以获得这些视图和它们所包含的信息的更多知识）。

#### 5. 监控并行查询进程

视图V\$parallel\_sysstat和V\$parallel\_qstat包含并行服务器进程及其行为的信息。查询视图 V\$parallel\_sysstat显示并行查询服务器的当前运行信息，例如忙的和空闲的查询服务器的数量与动态服务器创建和终止的统计数。V\$parallel\_qstat视图包含以前使用并行查询服务器运行的查询的信息（参见第38章以获得追踪并行服务器的详细信息）。

#### 6. 监控归档进程

归档进程活动信息存储在 V\$archivedata视图中，可以从这个视图中查询由 ARCH进程写入的归档日志信息（各列的定义参见第 39章）。

#### 7. 监控多线程服务器进程

视图V\$mmts、V\$dispatcher和V\$shared\_server包含多线程服务器（MTS）和内存结构的状态信息。V\$mmts视图包含共享服务器进程的追踪信息，例如启动和终止的服务器数量和运行服务器的最高水位；V\$dispatcher视图包含运行的调度进程的信息，从这个视图中，可以查询调度进程的名字、支持的协议、处理的字节数、处理的消息数、当前的状态和与调度进程相关的其他运行信息；V\$shared\_server视图为运行的共享服务器进程提供相同类型的信息（参见第33章“Oracle网络基础”，以获取关于设置和调整共享服务器和调度进程的详细信息）。