Oracle数据库既包含物理组件，也包含逻辑组件。

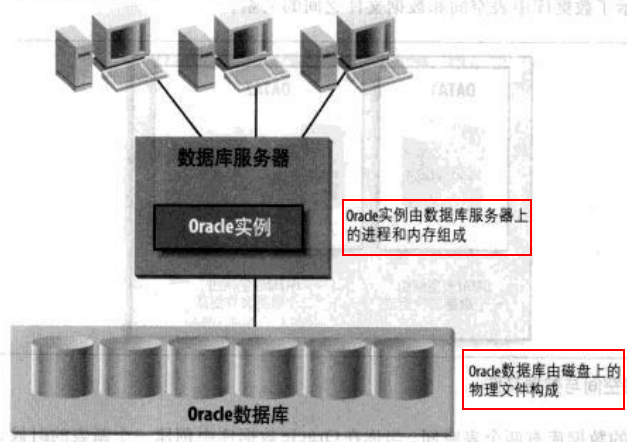
内容：

* Oracle数据库与其实例的区别
* 物理组件
* 实例
* 数据字典

## 1、数据库与实例

实例和数据库是两个不同但又相关的实体。

在Oracle中，数据库指的是信息的物理存储，而实例指的是服务器上运行的软件，它提供了对数据库中信息的访问。实例可以运行在计算机上，也可以运行在服务器上；数据库存储在服务器上的磁盘中。



数据库是物理的：它由存储在磁盘中的文件组成。

实例是逻辑的：它由服务器内存中的结构和进程组成。

例如：Oracle既使用系统全局区域（SGA）的共享内存区，也使用程序安全区域的用于每个进程的私有内存区域。尽管同一个数据库可以由多个实例，但是一个实例属于且仅属于一个数据库。实例时短暂存在的，而数据库只要维护适当可以永久存在。

用户并不能直接访问Oracle数据库中的信息，而是将访问请求发给Oracle实例。

现实世界提供了一个很好的例子来模拟实例和数据库的关系：如果把数据库看作是一个岛，那么实例可以被看作是通往这个数据库的一座桥。出入这个岛必须经过这座桥。如果桥关闭了，尽管岛仍然存在，但出入这个岛已经不可能了。用Oracle的术语来说就是：实例在运行，数据就可以流入或流出数据库。数据库的物理状态可以变化。如果实例宕掉了，尽管数据库在物理上还存在，但用户已经不能再访问这个数据库了。数据库是静态的：它本身不会发生什么变化。数据等待着实例恢复提供业务。

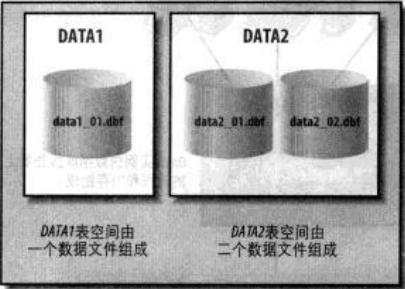
## 2、Oracle数据库中的结构

Oracle数据库中的结构包括表空间、控制文件、redo日志文件、日志档案、块变化跟踪文件、Flashback日志以及恢复备份（RMAN）文件。

### 2.1表空间

Oracle数据库中所有的数据都存在于表空间内。表空间是一个逻辑的结构：你无法通过观察操作系统看到表空间。每一个表空间都由数据文件的结构组成：每个表空间必须包含一个或多个数据文件，每个数据文件仅属于一个表空间。在创建表时，可以在那个表空间中创建表。接着，Oracle会将这个表安放在组成表空间的一个数据文件中。

数据库表空间和数据文件之间的关系：



该图显示的数据库有两个表空间。当你在Oracle数据库中创建一个新表的时候，可以选择将表放在DATA1表空间或者DATA2表空间中。这个表在物理上存在于组成表空间的某个数据文件中。

（就Oracle 10g而言，Oracle为所有类型表指定的默认表空间就是本地管理的表空间。这种表空间管理方式允许创建大文件表空间，这将充分利用64位系统的优势及其超大文件的管理能力。

Oracle 9i引入了Oracle管理文件（OMFs）的概念，这使得数据库可以在适当的时候自动创建、命名和删除组成数据库的所有文件。OMFs降低了命名和跟踪数据库文件的维护成本，同时也避免了在人工执行此类操作时可能导致的问题。从Oracle 10g开始，OMFs和大文件表空间特性使得数据文件对用户完全透明。

Oracle数据库可以部署在高达64000个数据文件中。由于大文件表空间包含的文件比小文件空间的大1024倍，而且大文件表空间在64位操作系统中的块大小为32KB，因而Oracle数据库可以扩展高达8安字节（一个按字节等于一百万兆字节）。大文件表空间设计用于Oracle自动化存储管理（ASM），支持清楚的其他逻辑卷管理工具，以及RAID。

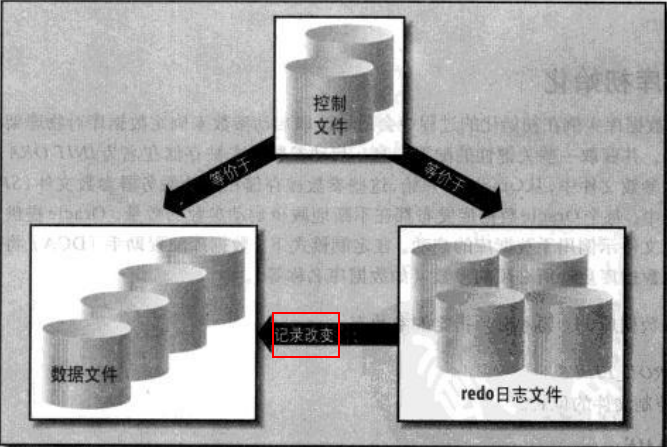
）

### 2.2数据库文件

Oracle数据库中有三种基本类型的物理文件：

1. 控制文件
2. 数据文件
3. redo日志文件

这三种基本类型描述了物理数据库自身。相互关系：



控制文件包含了组成数据库（包括数据文件和redo日志文件）的其他物理文件的位置信息，以及有关数据库内容和状态的关键信息，包括：

* 数据库的名称
* 数据库创建的时间
* 数据文件和redo日志文件的名称和位置
* 表空间信息
* 数据文件离线范围
* 日志历史及当前日志顺序信息
* 归档的日志信息
* 备份集合、数据片、数据文件和redo日志信息
* 数据文件拷贝信息
* 检查点信息

控制文件除了提供启动时的基本信息，还用于删除数据库。从Oracle 10g开始，用户可以使用DROP DATABASE命令来删除控制文件中列出的数据库文件和控制文件本身。

## 3、数据库初始化

Oracle数据库实例在初始化的过程中会通过读取启动参数来确定数据库对物理架构的应用情况，并获取一些关键性的配置信息。启动参数过去被存储在名为INIT ORA的实例初始化参数文件中，从oracle 9i开始，这些参数被存储在名为服务器参数文件（SPFILE）的仓库中。每个Oracle数据库发布都在不断地减少启动参数的数量。Oracle提供了一个初始化文件示例用于数据库的启动。在定制模式下，数据库配置助手（DCA）将提示用户输入数据库启动所必须的参数（如数据库名称等）。

Oracle 11g要求必须制定的参数有：

CONTROL\_FILES——控制文件的位置

DB\_NAME——本地数据库名称

DB\_DOMAIN——数据库域名

LOG\_ARCHIVE\_DEST——日志文档目的位置

LOG\_ARCHIVE\_DEST\_STATE——打开日志归档的参数

DB\_RECOVERY\_FILE\_DEST——数据库迅速恢复区域（目录、文件系统或者ASM磁盘组）的位置

DB\_RECOVERY\_FILE\_DEST\_SIZE——数据顺序恢复区域的最大字节数

DB\_BLOCK\_SIZE——数据块大小，以字节计算

PROCESSES——并发数据库操作系统进程的最大数目

SESSIONS——数据库会话的最大数目

OPEN\_CURSORS——数据库被打开游标的最大数目

SHARED\_SERVERS——数据库共享服务器的最小数目

REMOTE\_LISTENER——远程监听者名称

COMPATIBLE——当特性影响文件格式时需要兼容的数据库版本

MEMORY\_TARGET——自动分配给SGA和实例PGA组件的目标内存大小

DDL\_LOCK\_TIMEOUT——DDL语句等待互斥锁的超时时间，这是针对数据定义语言语句的参数。

NLS\_LANGUAGE——为数据库指定国家支持（NLS）语言

NLS\_YERRITORY——为数据库指定国家语言支持的范围

Oracle 11g在许多方面迈向自动化，其中一个例子就是，UNDO\_MANAGEMENT这个参数的默认值现在被设置成自动化undo管理。在数据库恢复、读一致性、flashback特性中，undo用于事务的回滚。（不过redo记录存储在物理的redo日志中；该日志存储了数据段和undo数据块的变化，同时存储了undo段的事务表。）目前Oracle可以根据undo表空间配置方式自动调整undo的持续周期。

## 4、部署物理组件

Oracle数据库部署的实践性指导。

### 4.1控制文件

数据库通常在不同的物理磁盘中至少包含两个控制文件。缺少控制文件的当前拷贝，系统会有丢失数据库中部分内容的危险。丢失控制文件未必是致命的问题；通常有办法可以重建它们。不过重建控制文件本身比较困难，可能引入新的风险。

控制文件的位置是通过启动参数CONTROL\_FILES来定义的。可以通过在CONTROL\_FILES参数中指定多个位置来指定多个控制文件，例如：

control\_files = (/u00/oradata/control.001.dbf,

/u01/oradata/control.002.dbf,

/u02/oradata/control.003.dbf

)

Oracle将会保证控件文件的所有拷贝都能够保持同步，对控制文件的所有更新将会同时进行。如果没有指定这个参数，Oracle将会使用默认的文件名创建控制文件，或者使用Oracle管理文件特性。

许多Oracle数据库部署的时候都采用了诸如RAID-1或者RAID-5等冗余磁盘解决方案，从而避免因单个磁盘故障导致数据丢失。

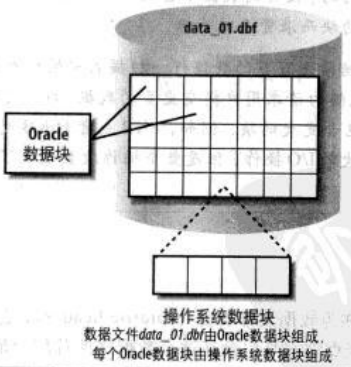
将控制文件存储在磁盘中，需要维护多个拷贝。用户不需要担心写多个控制文件可能会对性能产生影响。在Oracle环境中，相比其他磁盘I/O而言，控制文件的更新操作对性能影响很小。

### 4.2数据文件

数据文件存储着数据库中的实际数据，包括存储数据的表和索引，维护数据结构信息的数据字典，以及用于实现并发性的回滚段。

数据文件由Oracle数据库数据块组成，而Oracle数据块由磁盘上的操纵系统数据块组成。Oracle数据块的代销为2KB到32KB不等。在Oracle 9i之前，整个数据库中仅包含一种大小的块。之后，用户可为数据库设置默认的块大小；而数据库支持多大五种大小非默认的块（不过每个表空间仅使用一种大小的块）。

Oracle数据块与操作系统数据块之间的关系：



数据文件仅属于一个数据库的一个表空间。数据库实例根据用户的操作和需要并以Oracle数据块为单位将数据读入内存。数据库实例还会根据需要将数据块从存储写入磁盘文件中的数据文件，以保证数据库能够可靠的记录用户所做的改动。

数据文件是位于Oracle数据库与操作系统之间最底层的粒度。当你将数据库安置在I/O子系统之上，你所能放置在任何位置的最小部分就是数据文件。调整I/O子系统以改进Oracle性能的典型操作之一就是将数据文件从一组磁盘移到另一组。从Oracle 10g开始，Oracle提供了自动化存储管理，用于自动清除，从而省去了这种调整任务中的人工操作。

Oracle 9i之后，可以使用实例启动参数DB\_BLOCK\_SIZE来设置数据库默认的块大小。

块大小是一次读或者写操作的最小数据量。在线事务处理（OLTP）系统中，一个事务通常包含数据量不多但已经定义好的若干行数据，这些行可能是特定用户经过排序的产品集合。在这种类型的操作中，对行的访问操作适合通过索引来进行，这与扫描整个表操作是不同的。较大的数据块会包含事务不需要的额外数据，Oracle并不会因为传输这些大的块而浪费系统资源。

数据仓库的工作负荷包括读上百万的数据航，扫描表中的所有数据。对于这类操作，使用较大的数据块能够为请求用户提交更多的数据。为了更好的支持这种类型的操作，数据仓库通常包含更大的块，例如，8KB或者16KB大小的块。较大的数据块可能会稍微延长每次的I/O操作，但是整个操作次数减少了，最终系统的整体性能得到很大改善。

1）数据文件结构

数据文件的第一个数据块称为数据文件首部（datafile header），它包含了维护整个数据库完整的关键信息。

检查点结构是包含在这个数据文件首部中最重要的信息之一。它是一个逻辑时间戳，指示改动写入数据文件的最后时刻。这个时间戳对数据库恢复来说极其关键，Oracle数据的恢复过程就是利用数据文件首部中的这个时间戳来确定应该使用哪个redo日志文件将数据文件及时的恢复到当时那个时间点。

2）数据区间与段

从物理层次来看，一个数据文件是作为操作系统数据库加以存储的。从逻辑层次看，数据文件有三个中间结构层：数据块（block）、数据区间（extent）和段（segment）。数据区间是指一个Oracle数据文件中相邻的数据块的集合。段就是Oracle数据库中占据空间的一个对象，如由一个或多个数据区间组成的表或索引。

当Oracle更新数据的时候，它首先尝试着在同一个数据块中更新数据。如果这个数据块中没有足够的空间用容纳新的数据，那么Oracle将会把数据写入一个新的数据块，这个数据块可能属于另一个数据区间。（如需深入了解有关数据区间与段的信息以及它们对性能的影响，参考“碎片与重组”，Oracle 10g增加了Segment Advisor，该工具可以大大简化回收利用数据库未使用的空间。）

### 4.5redo日志文件

redo日志文件记录了事务和Oracle内部行为对数据库造成的变动。由于Oracle通常会在内存中缓存被修改的数据块，当数据库实例发生故障的时候，这些被修改的数据块可能还未被写入数据文件。这个时候，redo日志文件中记录的变动可以被应用，从而保证事务的一致性。

另外，当ROLLBACK语句被执行的时候，redo日志文件会被用于undo操作。未提交给数据库的变动将会回滚到上次提交后的状态。

禁止记录redo日志：

在默认情况下，Oracle会在日志中记录数据库的所有变动。redo日志的产生会给系统增加一定的开销。用户可以禁止redo日志的产生，以便提高操作的执行速度。这样，相关操作将不会记入redo日志，当故障发生的时候，用户也无法恢复该操作。

如果你确实决定禁止记录某些操作的redo日志，那么可以在这些操作的SQL语句中加入NOLOGGING关键字。以后当操作过程中出现故障，你需要重复该操作。例如，可以为一个表创建但不用生成redo信息。当数据库发生故障，然后由恢复，这个索引不会被重新创建，因为这个操作没有被写入日志。这个时候，你需要简单地执行原先的脚本再次创建索引。

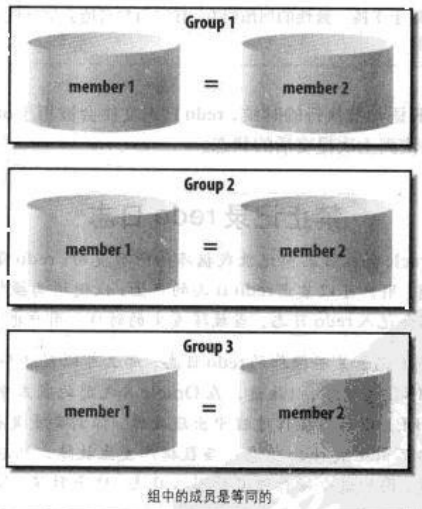
为了简化故障恢复的操作过程，建议对未记入日志的操作进行备份；如果你无法承受操作结果丢失的后果，或者由于某种原因无法重复这个操作，那么更应该对该操作进行备份。除了在某些命令中使用NOLOGGING关键字之外，还可以为表或者整个表空间标记上NOLOGGING属性。这样做会禁止为所有应用在该表或者该表空间上操作记录redo信息。

1）增加redo日志文件的拷贝

Oracle定义了特定术语来描述redo日志的管理方法。一个Oracle实例使用一个redo线程来记录它对数据库的改动。redo线程由若干个redo日志组构成，而redo日志组由一个或多个redo日志成员构成。

从逻辑上讲，可以把redo日志组看成单个redo日志文件。不过，Oracle允许你指定redo日志的多个拷贝，以保护redo日志非常重要的完整性。通过为每个redo日志文件创建多个拷贝，就可以保护redo日志以防磁盘故障和其他类型的灾难。

下图中给出拥有多个组合多个成员的redo线程。每个组两个成员，对应于互为镜像的redo日志。



一个redo日志组有多个成员，表示Oracle为一个redo文件维护了多个拷贝。用于控制文件复制的参数同样在这里适用。然而，对于控制文件来说，如果丢失了，用户可以重新构建控制文件中的静态部分；但对于redo日志文件而言，丢失了，就无法重载。因而，用户应确保redo文件有多个拷贝。由于人为错误而导致的redo日志文件受损或丢失，简单的冗余磁盘保护方案起不到作用。

Oracle会同步地对所有的redo日志成员执行写操作。等到对所有redo日志的写操作都完成之后，Oracle才会认为此次写操作完成。如果日志文件的一个拷贝放在一个快速或者负荷较轻的磁盘上，而另一个拷贝放在一个慢速或者比较繁忙的磁盘上，那么整体的性能将会受限于这个慢速的磁盘。Oracle必须保证redo日志文件所有的拷贝都能够成功地被更新以防止数据丢失。

如果Oracle对redo日志不同拷贝的写操作并不同步，例如，先写主要的日志文件然后在后台更新其他拷贝，那么将发生什么情况。当故障发生导致系统宕机，主要的日志文件被损坏的时候，Oracle可能并没有完成对所有redo日志文件拷贝的更新。这个时候，用户提交的事务可能就会丢失，因为记录着变更的主要日志文件已丢失，而日志的其他拷贝还没有及时更新。

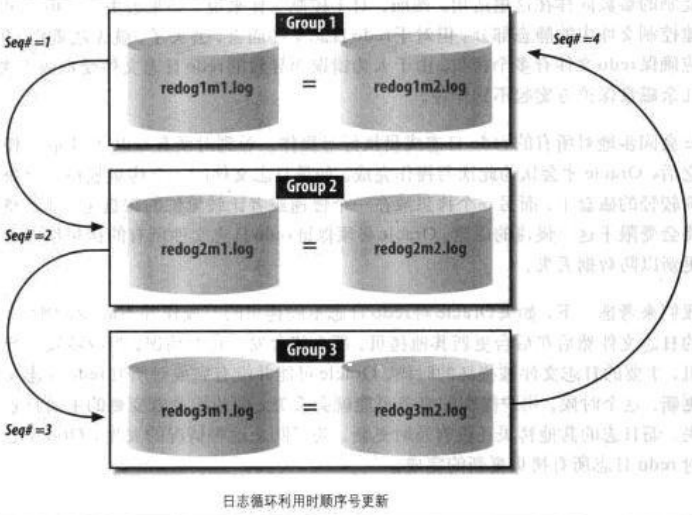
2）Oracle如何使用redo日志

一旦Oracle将一个redo日志文件写满，它将自动开始使用下一个日志文件。当Oracle数据库服务器使用完所有的redo日志文件，它将会返回到第一个，并重新使用它。在这个过程中，Oracle通过redo日志顺序号来跟踪不同的redo日志。该顺序号记录在redo日志文件中。

考虑下面三个redo日志文件：redolog1.log、redolog2.log、redolog3.log。Oracle第一次使用它们的顺序为1、2、3。当Oracle返回到第一个redo日志redolog1.log的时候，它将重新使用这个文件，并将它的顺序号更新为4。当Oracle移到redolog2.log时，它会把该文件的顺序号更新为5。

记住操作系统是使用redo日志文件来表示对应的物理文件，而Oracle使用redo日志文件的顺序号来确定循环使用日志文件的顺序。由于Oracle会自动重用redo日志文件，因而redo日志文件的名称并不一定能够指示它在redo日志文件序列中的位置。

redo日志文件的循环使用：



3）Redo日志的命名规则

操作系统对组成数据库的各种文件的命名相当重要。至少对用户来说，是这样的，因为用户是通过文件的名称来认识这些文件的。如果没有使用Oracle管理文件（OMF），那么应该遵循这个命名惯例，从而保证文件名称能够体现该文件本身的目的及某些关键细节。下面给出了一种可行的redo日志文件命名规则：

redog1m1.log，redog1m2.log

redo前缀和.log后缀表明这是一个redo日志文件。g1m1和g1m2字符串给出了组号和成员号。

4）归档redo日志

Oracle循环使用以前用过的日志文件，就会覆盖这些文件中以前的日志；那么eOracle是如何避免丢失这些日志文件中的重要信息？

对于这个问题实际上有两种解决方法：

1. 因为你没有采用避免信息丢失的方法，所以错误发生时肯定会遭受一定的损失。当redo日志文件被覆盖后，存在其中的历史信息就丢失了。如果系统发生错误并损坏了数据文件，你必须将这个数据库恢复到前一次备份时的状态；因为前一次备份后数据库变动的历史信息已经被冲走，所以无法再产生这些变动，从而丢失这些变动给数据库带来的影响。由于不能将数据库恢复到错误发生的地方，必然会导致损失，这种方法难以被用户接受，几乎没人采用这种方法。
2. 在日志文件被重写之前，将日志文件归档。不过，为了更好的理解日志文件归档，首先应该了解Oracle日志文件的两种类型：
   1. 在线redo日志：Oracle正在循环使用的操作系统文件，用于记录数据库的变动。
   2. 归档redo日志：被写满的在线redo日志文件的拷贝，用于避免因在线redo日志被覆盖而带来的redo数据丢失。

关于redo日志的归档，Oracle数据库可以运行下面两种不同的模式：

1. NOARCHIVELOG：就是不对redo日志归档。
2. ARCHIVELOG：当Oracle指向一个新的redo日志，它将会对上一个redo日志归档。为了防止在备份的时间上出现间隙，只有当一个redo日志文件被成功归档之后，它才能够被重用。归档redo日志加上在线redo日志记录了数据库变动的完整历史信息。借助它们，Oracle将所有提交的事务恢复到故障发生的准确时间点。Oracle运行在这种模式中才能支持表空间和数据文件的备份。

Oracle在利用redo日志和归档redo日志恢复数据库的过程中，前面提到的内部顺序号将会起到指导作用。

5）ARCHIVELOG模式和自动化归档

从Oracle 10g开始，通过下面的SQL语句就可以打开Oracle数据库的自动归档功能：

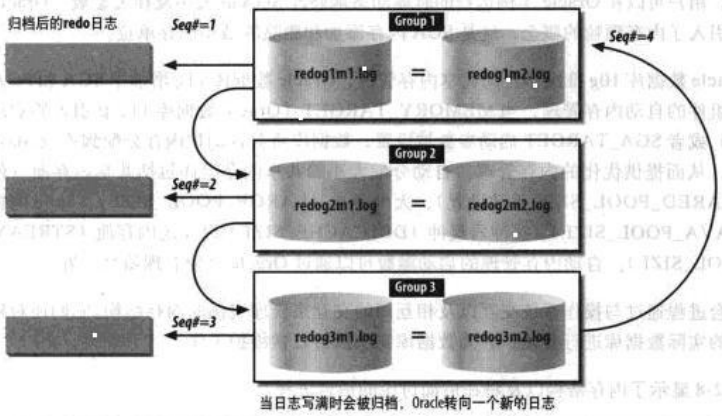
ALTER DATABASE ARCHIVELOG

如果数据库工作在ARCHIVELOG模式中，当redo日志文件被写满以后，Oracle会把它们标记出来。被写满的日志文件在被重用之前必须被归档。ALTER DATABASE ARCHIVELOG命令将会默认打开自动归档功能，启动归档程序。

归档日志目录以及归档日志名称的格式可通过LOG\_ARCHIVE\_DEST和LOG\_ARCHIVE\_FORMATA参数来指定。

由于初始文件是在Oracle实例启动的时候被读取，因而参数修改后并不会立即生效，必须重启Oracle实例。另外，打开自动归档功能并不会是数据库运行在ARCHIVELOG模式下。同样，将数据库运行在ARCHIVELOG模式下并不会打开自动归档功能。

同时，还需要确保归档日志的目的目录必须由足够的空间可用Oracle向里面自动写日志。如果指定的目录满了，Oracle会因为无法归档redo日志文件而被挂起。



归档redo日志对数据库的恢复至关重要。一个在线redo日志可以有多个拷贝，同样用户可以为归档日志指定多个目的地。Oracle会把鞋面的redo日志拷贝到这些目录中。另外，用户还可以指定是否要所有的拷贝都必须成功。完成上述功能的参数如下所示：

LOG\_ARCHIVE\_DUPLEX\_DEST

为redo日志指定另外一个归档目录，已实现归档redo日志的冗余。

LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST

指定redo日志是否必须写入一个或者所有的归档位置。如果是两个，有效值为1或2，如果两个以上，有效值为1~10。

## 5、实例包含的内存和进程

Oracle实例可以定义为一个共享的内存区域以及一个后台进程集合。实例的这个共享内存区域叫做系统全局区域（SGA）。SGA并不是一个不做区分的大内存段——它由不同的部分组成。一个实例的所有进程（包括系统进程和用户进程）共享SGA。

Oracle 9之前的版本中，SGA的大小都是在Oracle实例启动之前设置的。修改SGA大小或者SGA中其他属性的唯一方法就是修改启动参数，然后重启实例。从Oracle 9i开始，用户可以在Oracle实例运行的时候动态的修改SGA的大小及相关参数。Oracle 9i引入了内存颗粒的概念（SGA内存添加和删除的最小内存单位。）

Oracle 10g推出了自动共享内存管理，Oracle 11g增加了SGA和PGA实例组件的自动内存管理。当MEMORY\_TARGET或者SGA\_TARGET启动参数被设置，数据库将会自动把内存分配到不同SGA组件，从而提供优化的内存管理。自动分配大小的共享内存组件包括：

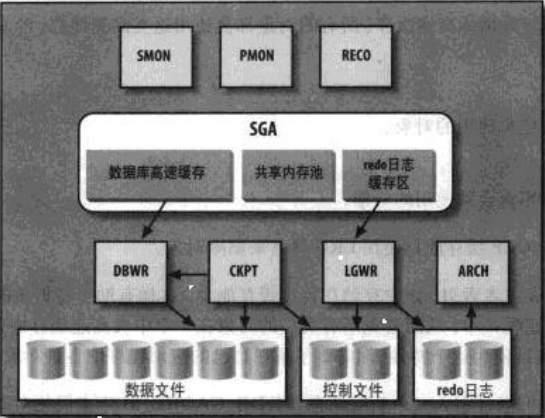
* 共享内存池（使用SHARED\_POOL\_SIZE手动设置）
* 大内存池（LARGE\_POOL\_SIZE）
* Java内存池（JAVA\_POOL-SIZE）
* 缓冲器缓冲（DB\_CACHE\_SIZE）
* 流内存池（STREAMS\_POOL\_SIZE）。

自动内存管理的启动参数可以通过Oracle企业管理器来设置。

后台进程通过与操作系统相互间的交互来管理实例的内存结构。它们还对磁盘上的实际数据库进行管理，并为数据库实例执行日常维护工作。

### 5.1实例的内存结构

如图，系统全局区域由多个内存区组成，它包括图中所示的数据库高速缓存、共享内存池和redo日志缓存，也可能包括Java内存池、大内存池以及流内存池。



1）数据库高速缓存

数据库高速缓存中存放着从数据库中获取的数据。这个缓存位于用户请求和实际的数据文件之间，用于改进Oracle数据库的性能。如果一组数据可以在高速缓存中找到（例如，最近查询过这个组数据，它们就会被存储在高速缓存中），那么用户就可以直接从存储中获取这组数据，从而节省了访问磁盘的开销。Oracle使用最近最少使用（LRU）算法管理高速缓存。（如果用户请求最近被使用过的数据，这些数据很可能已经存储在数据库缓存中，如果没有读磁盘的操作需要执行，那么高速缓存中的数据可能会被立即释放。）

当用户期望读一块高速缓存中不存在的数据时，这块数据必须从数据库中读出来，并写入高速缓存。当用户对这块数据做了修改，这些变化会被应用到高速缓存中的这块数据中。在随后的某个时间，这些变化将会被写入数据库所属的数据文件。这样，用户就不需要等待Oracle把被修改的数据块写入磁盘。

只有在完全必要的情况下才会等待I/O的执行，这一点对于整个Oracle而言都是适用的。磁盘是计算机系统中执行速度最慢的组成部分，因此I/O执行得越少，系统的运行速度越快。通过延迟非关键性的I/O操作而不是立即执行，可以使Oracle数据库得到更好的性能。

从Oracle 8开始，数据库高速缓存可以配置下面几种类型的缓存池：

DEFAULT

标准Oracle数据库高速缓存。所有的对象都会使用这类高速缓存，除非通过其他方式指出。

KEEP

用于存放经常使用的对象。

RECYCLE

用于存放不再会被使用的对象。

KEEP和RECYCLE缓存池都是由LRU算法来删除对象。

可以标记一个表或者索引，让它存储在特定缓存池中。这样有助于将更多的合适对象缓存在同一个高速缓存中，从而避免所有类型的对象在一个中央高速缓存中争夺空间。当然，为了更正确地使用这个特性，需要了解在自己的应用程序中各种对象的访问模式。

Oracle 10g通过引入一个新的动态参数DB\_CACHE\_SIZE来简化对高速缓存容量的管理。这个参数用于指定高速缓存的内存大小。如果MEMORY\_TARGET或者SGA\_TARGET这两个参数其中之一被设置，那么Oracle将自动确定DB\_CACHE\_SIZE的大小。其他启动参数包括DB\_KEEP\_CACHE\_SIZE和DB\_RECYCLE\_CACHE\_SIZE，必须手动设置大小。

2）共享内存池

共享内存池缓存着用户间共享的各种结构。例如，用户提交的SQL查询和查询中的部分代码以及查询结果都会被缓存起来，如果以后相同的语句再次被提交，那么缓存中的东西可以被重用。PL/SQL函数也会被加载到共享内存池执行，这些函数及其结果都将会被缓存，这里使用的算法也是LRU。就Oracle 11g而言，PL/SQL函数也可以被标记下来，指定它的结果应该被缓存，当用户使用相同的参数再次调用它的时候，Oracle数据库不需要重新计算，只需查找共享内存池。共享内存池还可以用于缓存Oracle数据字典中的信息；这些一般都是些描述数据库本身结构和内容的元数据。

可以手动指定SHARED\_POOL\_SIZE启动参数，也可以通过指定MEMORY\_TARGET参数或者SGA\_TARGET参数自动为该参数设值数值。

3）redo日志缓存区

redo日志缓存区缓存着redo信息，直到这些信息被写入磁盘中物理的redo日志文件。这个缓存区还可以提高系统性能。Oracle可以缓存着redo信息，一直到某个合适的时候，再将它写入磁盘，这样可以避免频繁地写入磁盘所带来的开销。

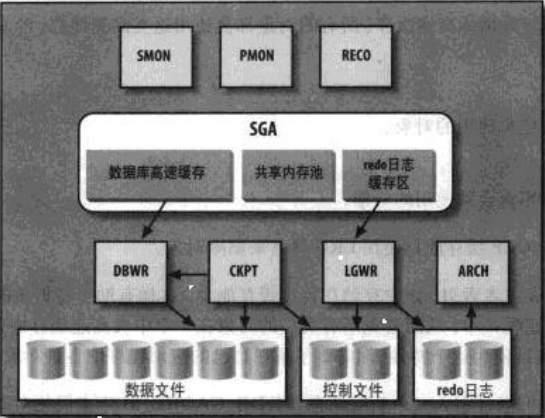
4）SGA中的其他内存池

* 大内存池（LARGE\_POOL\_SIZE）：用于为各种I/O服务器进程、数据库备份以及数据库恢复提供内存分配；另外还提供会话内存，用于共享服务器和Oracle XA的事务处理。
* Java内存池（JAVA\_POOL\_SIZE）：用于为Java对象和Java程序运行（包括数据库中Java虚拟机中数据）提供内存分配。
* 流内存池（STREAMS\_POOL\_SIZE）：用于为Oracle Streams在SGA中的消息队列提供内存分配，但并不为Oracle Streams在数据库表格中的消息队列提供内存分配，另外，它还为捕捉和应用提供内存。

### 5.2自动化PGA管理

Oracle能够自动管理分配给实例程序全局内存区（PGA）的内存。PGA由会话内存和私有SQL内存区组成。这个内存区的大小可以通过PGA\_AGGREGATE\_TARGET初始化参数来设置。自动化PGA管理，从Oracle 10g开始就正式可用，极大地简化了对SQL工作区的管理，使用自动化PGA就无需再设置前面提到的几个初始化参数。在Oracle 11g中可以通过设置MEMORY\_TARGET参数来自动调整PGA的内存分配以及SGA的内存分配。

### 5.3数据库实例的后台进程



如图显示了最常见的后台进程。不同的Oracle发布，它们的后台进程也会不同。Oracle 11g有这样一些后台进程：

* 数据库写进程（DBWR/DBWn）：该进程负责将SGA中数据库高速缓存里的数据写入磁盘中的多个数据文件。一个Oracle实例中，用于处理写文件的DBW进程多达20个，因而用DBWn来标识每一个进程。不过，大多数数据库实例仅运行一个DBW。在下面两种情况下，DBW会把数据块从缓存写入磁盘：
  + 当Oracle需要执行一个检查点的时候（例如，更新数据文件的数据块以便于redo日志保证一致）。Oracle在事务提交的时候为该事物写redo日志，之后才会写实际的数据块。Oracle周期性地执行检查点，以保证数据文件的内容与事务redo日志保持一致。
  + 当Oracle需要将用户请求的数据块读入高速缓存但是此时高速缓存并没有可用空间的时候。这个时候，从缓存中移除并写入磁盘的数据块是最近最少访问的数据块。根据此算法执行写操作可以降低由此带来的性能损耗。
* 日志写进程（LGWR）：该进程负责将SGA日志缓存区中的redo信息写入磁盘里redo日志文件当前的各个拷贝中。在事务处理的过程中，相关的redo信息被记录在SGA的redo日志缓存区中，当事务处理结束，Oracle的日志写进程将会把这些redo信息永久地写入磁盘。
* 系统监视进程（SMON）：该进程负责维护Orcale实例的整体稳定性和可靠性。当数据库实例故障重启后，SMON进程将会执行宕机恢复，在实时应用程序集群（RAC）中，有多个数据库实例访问同一个数据库，SMON进程能够执行故障实例恢复。SMON还可以吧数据文件中相邻的空闲内存合并为一块，并且可以将以前用于记录排序而现在不再需要的空间释放掉。
* 进程监视进程（PMON）：该进程负责监视访问数据库的用户进程。当某个用户进程异常退出的时候，PMON进程将负责清除其遗留下来的资源（如内存），并释放故障进程保持的任何锁。
* 归档进程（ARCH/ARCn）：该进程负责将写满redo日志文件拷贝到指定的归档日志目录。Oracle实例可运行多达10个归档进程，因而每个归档进程用ARCn标识。LGWR进程将会根据负载需要启动额外的归档进程。最多可启动的归档进程数据受限于初始化参数LOG\_ARCHIVE\_MAX\_PROCESSES。在默认情况下，这个参数的默认值为2，很少会改变。
* 检查点进程（CKPT）：该进程负责在执行检查点的时候更新数据文件头。
* 恢复进程（RECO）：该进程负责自动清除障碍的或者被挂起的分布式事务。
* 派遣进程：该进程为可选的后台进程，用于部署了共享服务器配置的环境。
* 全局高速缓存服务（LMS）：该进程负责管理实时应用程序集群资源以及实例间的访问。
* 作业队列：该进程负责批量调度用户PL/SQL语句或过程。
* 队列监控进程（QMNn）：该进程负责监控Oracle Streams消息队列，最多可支持10个监控进程。
* 自动化存储管理（ASM）相关进程：RBAL进程负责协调磁盘组活动量的重新平衡。ORBn进程负责实际执行这个重新平衡的工作。ASMB进程负责在数据库和ASM实例间提供通信渠道。

进程还是线程：

以上否是进程，那么Oracle在实现这些服务的时候操作系统这一层到底是使用了线程还是使用了进程。

通常使用进程这个术语来指示Oracle执行的功能。例如，DBW进程或者LGWR进程。对于运行在Windows上的Oracle而言，每个“Oracle进程”其实就是同一进程中的一个线程。而对于运行在UNIX上的Oracle而言，这些“进程”就是实际的操作系统进程，不是线程。因而，在UNIX中，DBW和LGWR是特定的操作系统进程，而在Windows中，它们是同一进程中的两个线程。

不过也有例外，数据库在这一层的具体实施方式也视数据库的版本和操作系统的不同而有所不同。不论什么平台，Oracle企业管理器始终保持了界面的一致性，从而让最终的用户和管理员可以忽略这些实施细节。

## 6、数据库字典

每个Oracle数据库都包含了一组元数据，用于描述包括表定义和完整性约束在内的各种数据结构。本章所讨论的所有组件在数据字典中都有对应的系统表和视图来描述它们的特性。用户可以使用标准的SQL语句来查询这些表和视图。如下表显示了在数据字典中如何查找每个组件的可用信息：

|  |  |
| --- | --- |
| 组件 | 数据字典表和视图 |
| 数据库 | V$DATABASE、V$VERSION、V$INSTANCE |
| 共享服务器 | V$QUEUE、V$DISPATCHER、V$SHARED\_SERVER |
| 连接池 | DAB\_CPOOL\_INFO、V$CPOOL\_STAT、V$CPOOL\_CC\_STATS |
| 表空间 | USER\_FREE\_SPACE、DBA\_FREE\_SPACE、V$TEMPFILE、DBA\_USERS、DBA\_TS\_QUOTAS |
| 控制文件 | V$CONTROLFILE、V$PARAMETER、 V$CONTROLFILE\_RECORD\_SECTION |
| 数据文件 | V$DATAFILE、V$DATAFILE\_HEADER、DBA\_DATA\_FILES、DBA\_EXTENTS、USER\_EXTENTS |
| 段 | DBA\_SEGMENTS、USER\_SEGMENTS |
| 数据区间 | DBA\_EXTENTS、USER\_EXTENTS |
| redo日志 | V$THREAD、V$LOG、V$LOGFILE、V$LOG\_HISTORY |
| undo | V$UNDOSTAT、V$ROLLSTAT、V$TRANSACTION |
| 归档状态 | V$DATABASE、V$LOG、V$ARCHIVED\_LOG、V$ARCHIVE\_DEST |
| 数据库实例 | V$INSTANCE、V$PARAMETER、V$SYSTEM\_PARAMETER |
| 内存结构 | V$SGA、V$SGASTAT、V$SGAINFO、V$SGA\_DYNAMIC\_COMPONENTS、V$SGA\_DYNAMIC\_FREE\_  MEMORY、V$SGA\_RESIZE\_OPS、V$SGA\_RESIZE\_CURRENT\_OPS、V$MEMORY\_TARGET\_  ADVICE、V$SGA\_TARGET\_ADVICE、V$PGA\_TARGET\_ADVICE |
| 工作区内存 | V$PGASTAT、V$SYSSTAT |
| 进程 | V$PROCESS、V$BGPROCESS、V$SESSION |
| 告警 | DBA\_THRESHOLDS、DBA\_OUTSTANDING\_ALERTS、DBA\_ALERT\_HISTORY、V$ALERT\_TYPES、V$METRIC |
| 性能监控 | V$LOCK、DBA\_LOCK、V$SESSION\_WAIT、V$SQLAREA、V$LATCH |
| RMAN恢复 | V$RECOVER\_FILE |
| 用户密码 | V$PWFILE\_USERS |
| 表 | DBA\_TABLES、ALL\_TABLES、USER\_TABLES |
| 索引 | DBA\_INDEXES、ALL\_INDEXES、USER\_INDEXES |
| 数据字典 | DBA\_OBJECTS、ALL\_OBJECTS、USER\_OBJECTS |

SYSTEM表空间总是包含数据字典表。以V$或者GV$为前缀的数据字典表是动态表，它们会持续被更新以记录Oracle数据库的当前状态。静态数据字典表通常以DBA\_、ALL\_或者USER\_为前缀，用于指示该视图中列出的对象范围。

--数据文件的基本信息

desc v$datafile;

select \* from v$datafile;

desc dba\_data\_files;

select file\_name,file\_id,tablespace\_name,bytes,blocks,

status,online\_status

from dba\_data\_files;