第2章 硬件配置研究

尽管每个Oracle数据库的组成部分基本相同，但可供选择的选项则随硬件平台及操作系统的不同而不同。对于大多数硬件平台，将有一系列可供选择的选项。本章将介绍可利用的标准体系结构——把各部分组合在一起的方法。由于Oracle支持许多硬件平台，所以本章不可能覆盖所有选项。这里重点介绍最常用的工具。

2.1 结构概述

一个Oracle数据库由物理文件、内存区和进程组成。这些组件的分布随着数据库体系结构的不同而不同。

数据库中的数据存储在磁盘上的物理文件(称为数据文件)中；数据被使用时，则调入内存。Oracle利用内存区来改进性能并管理用户间数据的共享。数据库中的主内存区又称为 System Global Area(SGA，系统全局区)。为了在SGA与数据文件之间读写数据，Oracle采用所有用户共享的一组后台进程来实现。

数据库服务器(也叫做实例)由一组内存结构和访问数据库文件的后台进程组成。图 2-1给出了服务器与数据库之间的关系。

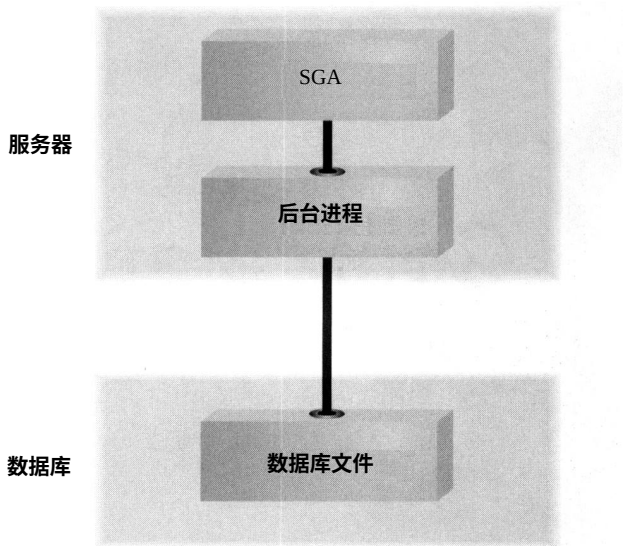

图2-1 Oracle中的服务器和数据库

数据库服务器的特征(例如SGA的大小和后台进程的数量)都在数据库启动时被规定。这些参数存储在 init.ora文件中。数据库的 init.ora 文件通常把数据库名包含在文件名中，称为ORA1的数据库一般具有一个称作initora1.ora的init.ora文件。

init.ora文件又可以调用相应的config.ora文件。如果使用config.ora文件，它通常只为不变的信息(例如数据库块大小和数据库名)存储参数值。初始化文件只在数据库启动时才被读取，

对此文件的修改要直到下一次启动时才起作用。

2.2 独立主机

数据库最简单的概念上的配置是一个服务器访问一个独立的、单磁盘主机上的数据库。在图2-2所示的配置中，所有的文件都存储在服务器的专用设备上，并且服务器上只有一个SGA和一组Oracle后台进程。

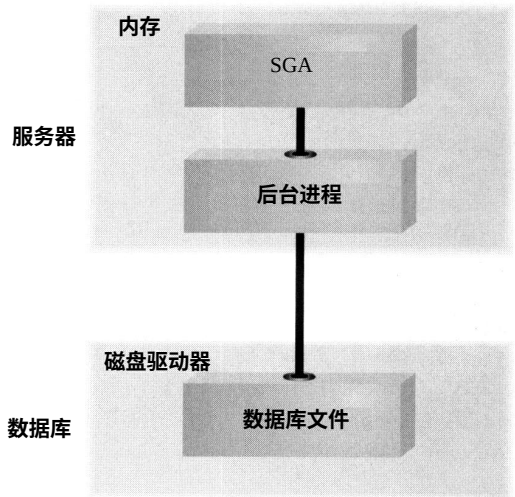

图2-2 独立主机中的单台服务器

图2-2所示的结构是一种最小化的配置。其他所有的数据库配置都是对此基本结构的修改。

磁盘中存储的文件包含数据库数据文件和主机的 init.ora文件 。如图2-2所示，在此数据库中存在两个主要接口点：

- 后台进程与数据库文件之间。
- 后台进程与SGA之间。

调整的重点主要是对这两个接口点的性能进行改善。如果指定给数据库的内存区域足够大，则很少对此数据库文件反复进行读操作。由于在这个配置中，所有文件都存储在专用磁盘设备上，所以要最小化数据文件的访问次数。文件调整的相关主题将分别在第 4章、第8章详述。

2.2.1 磁盘阵列独立主机

如果有多个磁盘，数据库文件便可以分开存储。这样可以减少数据库文件间的连接数量，提高数据库的性能。在数据库操作期间，处理一个事务或查询需要多个文件的信息是非常普遍的。如果不通过多个磁盘分配文件，系统就需要同时从同一个磁盘中读取多个文件。图 2-3展示了文件在多个磁盘间的分配。

数据库可以使用多种文件类型。这些文件类型及其在多磁盘间的优化分配准则将在第 4章讲述。

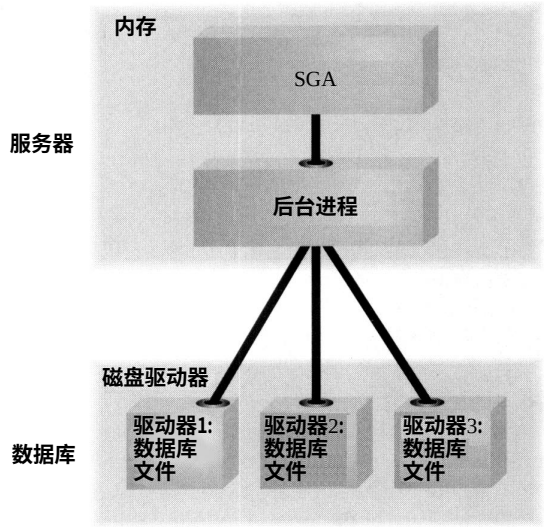

图2-3 多磁盘阵列独立主机上的服务器

1. 控制文件镜像

访问数据库的服务器的init.ora文件存储于Oracle软件目录下，通常是在Oracle软件基目录下的一个子目录中。在缺省的目录配置中，init.ora文件位于/orasw/app/oracle/admin/*instance_name*/pfile目录下。例如，如果实例名是ORA1，则init.ora文件将被命名为initoral.ora，并且存储于/orasw/app/oracle/admin/ORA1/pfile目录下。init.ora文件并不列出数据文件名或数据库的联机重做日志文件；这些文件都存储在数据字典中。不过init.ora文件列出了数据库的控制文件名。在一个多磁盘主机中，控制文件应存储于不同的磁盘中。数据库将保持它们之间的同步。通过把镜像的控制文件存储在多个磁盘中，可以大大降低由于介质故障而引起的数据库问题。

另一个服务器配置文件config.ora由init.ora调用。利用config.ora文件，可以为数据库中通常不变动的参数设置数值；控制文件名就在这些参数之中。下面是一个config.ora文件中CONTROL_FILES参数的条目例子。

```
control_files              = (/db01/oracle/ORA1/ctrl1ora1.ctl,
                              /db02/oracle/ORA1/ctrl2ora1.ctl,
                              /db03/oracle/ORA1/ctrl3ora1.ctl)
```

这个条目命名了3个控制文件。如果在数据库创建期间执行这一操作，则数据库便会自动创建这里列出的3个控制文件。若想为已有数据库增加另外的控制文件，可按如下步骤执行：

1) 关闭数据库。

2) 把一个当前控制文件拷贝到新的位置。

3) 编辑config.ora文件，在CONTROL_FILES条目中增加新的控制文件名。

4) 重新启动数据库。

新的控制文件将被激活。

一个init.ora文件可以通过IFILE参数调用多个初始化文件。最通用的IFILE条目包含config.ora参数文件。也可以嵌套初始化文件。例如，init.ora文件可能有一个IFILE条目包含一个tuning.ora文件。tuning.ora文件本身也可能有一个包含disktuning.ora文件的IFILE条目。可

以利用对多个初始化文件的支持来分组相关的参数。不过，必须注意不要因为把一个参数设置用在多个被包含的文件中而造成重复设置。

2. 重做日志文件镜像

如前面所述，数据库会自动镜像控制文件。同样，数据库也可以镜像联机重做日志文件。若要镜像联机重做日志文件，请使用重做日志组 (redo log group)来实现。如果重做日志组已被使用，操作系统就无须执行联机重做日志文件的镜像，它由数据库自动执行。

当使用这一功能时，LGWR后台进程会同时写入当前联机重做日志组中的所有成员。因此，会有重做日志文件组中的循环，而不是文件中的循环。由于组中成员通常存储于不同的磁盘驱动器中，所以在文件间不会存在磁盘冲突，因此 LGWR也几乎不影响性能。有关重做日志文件的布置的更多信息，请参见第 4章。

重做日志组可以通过create database命令创建，也可以在数据库创建后通过 alter database命令把重做日志组添加到数据库中。下面的例子描述了如何在一个现有的数据库中加入一个重做日志组。例子中的组名为“ GROUP 4 ”，使用组号可以简化对它们的管理，通常组号从 1开始，连续递增。例子中的alter database命令由Server Manager执行。

```
> svrmgrl
SVRMGR> connect internal as sysdba
SVRMGR> alter database
     2> add logfile group 4
     3> ('/db01/oracle/CC1/log_1c.dbf',
     4>  '/db02/oracle/CC1/log_2c.dbf') size 5M;
```

若要把一个新的重做日志文件添加到已存在的日志组中，请使用下例中的 alter database命令。和前面的例子一样，这个命令由 Server Manager执行，它在GROUP 4组中增加第3个成员。

```
> svrmgrl
SVRMGR> connect internal as sysdba
SVRMGR> alter database
     2> add logfile member '/db03/oracle/CC1/log_3c.dbf'
     3> to group 4;
```

从Oracle8i开始，可以登录到SQL*Plus和connect internal上以创建相同的日志文件或执行修改操作。

```
> sqlplus internal/<password>
 SQL> alter database
   2  add logfile group 4
   3 ('/db01/oracle/CC1/log_1c.dbf',
   4  '/db02/oracle/CC1/log_2c.dbf') size 5M;
```

若要把一个新的重做日志文件添加到已存在的组中，请使用下面列出的alter database命令。和前面例子一样，这个命令由SQL*Plus执行。该命令在GROUP组中增加第3个成员。

```
> sqlplus internal/<password>
 SQL> alter database
   2  add logfile member '/db03/oracle/CC1/log_3c.dbf'
   3  to group 4;
```

当使用alter database命令中的add logfile member选项时，并不规定文件大小的信息。这是因为组中所有成员必须有同样大小的文件。由于组已存在，所以数据库也就知道生成的新文

件有多大。

3. 归档重做日志文件镜像

对于Oracle8，在写归档重做日志文件时，可以命令数据库对每个文件都写多个拷贝。在init.ora中，LOG_ARCHIVE_DEST参数为归档重做日志文件设置主存储位置。在Oracle8.0中，可以使用LOG_ARCHIVE_DUPLEX_DEST参数为归档重做日志文件规定另一个存储位置。当写归档重做日志文件时，Oracle将写到这两个存储位置上。对主存储位置的写入必须每次都要成功，否则数据库将不可用。

对LOG_ARCHIVE_DUPLEX_DEST规定的第二个归档日志目标区域的写入是可选的。如果把LOG_ARCHIVE_MIN_SUCCEED_DEST设置成1，则只有对一个存储位置(第一个存储位置，由LOG_ARCHIVE_DEST规定)的写入必须成功。即使对另一个存储位置的写操作失败，数据库的可用性也不会受干扰。

从Oracle8i开始，归档日志目标区域的init.ora参数将发生变化。在Oracle8i中，LOG_ARCHIVE_DEST参数已不能用，由LOG_ARCHIVE_DEST_*n*取代。可以规定最多5个归档日志目标区域，用目标号代替*n*。例如，可以通过LOG_ARCHIVE_DEST_1和LOG_ARCHIVE_DEST_2参数规定两个不同的归档重做日志目标区域，如下所示：

```
log_archive_dest_1 = '/db00/arch'
log_archive_dest_2 = '/db01/arch'
```

在Oracle8i中，Oracle8.0的LOG_ARCHIVE_MIN_SUCCEED_DEST参数已不能用，代替该参数，可以使用LOG_ARCHIVE_DEST_STATE_*n*参数来允许或禁止归档日志目标。例如，为了禁止第二个归档日志目标，可把LOG_ARCHIVE_DEST_STATE_2设置成DEFER。在缺省情况下，这个状态值被设置成ENABLE。

如果不用Oracle8的内部能力镜像归档重做日志文件，就必须在操作系统级镜像它们。可以使用RAID技术(参见第4章)作为硬件镜像方法的一部分。只要硬件系统能支持镜像而无重大性能影响，一般应支持硬件系统对Oracle的镜像解决方案。采用Oracle方法或操作系统方法的镜像解决方案对跨平台操作非常方便，但可能对CPU的使用有所影响。

在恢复期间，不能跳过遗失的归档重做日志文件。有关对备份和恢复的详细情况，请参见第10章。

2.2.2 磁盘镜像独立主机

许多操作系统都提供了文件备份的维护及文件拷贝的同步，这些服务通过磁盘映像(disk shadow)或卷映像(volume shadow)进程来实现这种操作(也称作镜像)。

磁盘映像有两个好处。首先，磁盘映像可以作为磁盘失效时的备份来使用。在大多数操作系统中，磁盘失效会自动引发相应的映像磁盘来取代失效的磁盘。第二个好处是可以改进系统的性能。支持卷映像的大多数操作系统都能引导文件的I/O请求使用映像的文件而不是主文件集，借助文件映像实现对文件的I/O装载。这样就减轻了对主磁盘的I/O负载，增加了I/O的能力。图2-4展示了磁盘映像的使用。

注意 如果操作系统不支持异步写操作，操作系统级的磁盘映像可能影响写操作性能。

图2-4中所示的映像类型称为RAID-1(独立磁盘冗余阵列)映像。在这种映像类型中，主磁盘中的每一个磁盘与映像磁盘中的每一个磁盘一一对应成对。根据操作系统，也可以使用其

他映像选项。在RAID-3和RAID-5映像中，磁盘组被看作是一个逻辑单元，并且每一个文件自动“分布”于每个磁盘。在RAID-3和RAID-5中，一个奇偶校验系统提供了一种对磁盘组的破坏或失效成员恢复功能的方法。

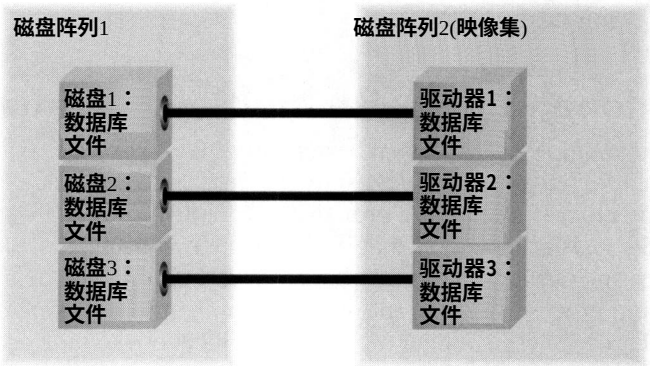

图2-4 磁盘映像

所采用的映像方法将影响文件在设备中的分配情况。例如，存储表的数据文件与存储表索引的数据文件通常分布于不同的磁盘中。然而，如果使用了RAID-3或RAID-5，则磁盘间的区别就变得模糊不清。使用这些选项访问数据文件时，几乎总是要求访问所有磁盘。因此很有可能出现磁盘间的冲突。

不考虑这些，冲突就不会很严重。例如，在RAID-5中，第一个数据块存储在组中的第一个磁盘中，第二个数据块存储在下一个磁盘中。因此，数据库只能在一个磁盘中读取一个块后才移到下一个磁盘中去读另一个数据块。这样，对同一个磁盘的多个访问而造成的任何冲突，也顶多发生在读取一个块时。

2.2.3 多数据库独立主机

在一个主机中，可以创建多个数据库。每一个数据库拥有自己的文件，并且被不同的服务器访问。第4章提供了合适的目录结构的准则。

图2-5展示了支持两个数据库的一个主机。由于每一个服务器要求一个SGA和一组后台进程，所以主机必须能够支持这种配置的内存和进程要求。

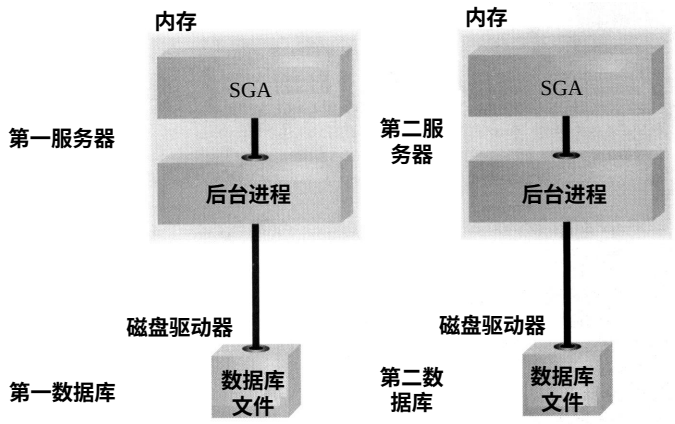

图2-5 多数据库的独立主机

如前所述，这些配置是对基本数据库结构的修改。在这种情况下，你只需模仿第一个数据库结构创建第二个数据库。请注意，尽管这两个数据库拥有同一个主机，但它们彼此并不交流。第一个数据库中的服务器，不能访问第二个数据库中的数据文件。

同一个服务器中的多个数据库一般可以共享相同的 Oracle资源代码目录。图2-5中，两个数据库的init.ora文件存储在不同的目录中，这是因为实例名是目录结构中的一部分。例如，如果两个服务器分别名为 ORA1、ORA2，它们相应的 init.ora文件则分别为 initora1.ora及initora2.ora。前一个文件存储在 /orasw/app/oracle/admin/ORA1/pfile目录下，后一个文件存储在/orasw/app/oracle/admin/ORA2/pfile目录下。像它们的数据文件一样，它们的服务器参数也是完全相互独立的。它们的 config.ora文件应当也在文件名中包含了服务器名 (例如，configora1.ora、configora2.ora)，也应当将其存储在与它们各自的 init.ora文件相同的目录下。

尽管数据库可以共享相同的资源代码目录，但它们的数据文件应存储在各自的目录下；并且如果合适的话，可以存储在各自的磁盘上。数据文件的目录结构样本将在第 4章讨论。如果多个数据库有存储于同一个设备上的数据文件，则没有一个数据库的 I/O统计能够正确反映该设备的I/O负载。这就需要累加每一个数据库对每一个磁盘的 I/O次数。

简化升级过程

如果你的操作系统支持多个版本的 Oracle软件，那么就可以大大简化升级 Oracle数据库版本的过程。例如，如果 Oracle8i的8.1.5版和Oracle8i的8.1.6版都安装在主机上，通过下述操作就可以把一个数据库从 8.1.5升级到8.1.6 ：

1) 关闭数据库。

2) 编辑环境文件(通常为/etc/oratab)，以使Oracle软件主目录指向8.1.6软件目录。

3) 将环境变量(例如Oracle_HOME)置“ 0 ”，以指向数据库新的主软件目录。

4) 把listener.ora(见第三部分中关于 SQL*Net和Net8的说明)编辑成指向数据库新的主软件目录。停止并重新启动监听器。

5) 转到新软件版本之下的/rdbms/doc目录。查找README.doc文件并转到关于升级的部分。根据版本的不同，升级后需要一个或多个“ cat ”脚本文件来更新数据字典目录。彻底阅读README.doc文件以便发现将要应用的“ cat ”脚本文件。

6) 转到新软件版本的/rdbms/admin/子目录下。

7) 安装并打开数据库。例如对于 ORA1实例：

```
SVRMGRL> connect internal as sysdba
SVRMGRL> startup mount ORA1 exclusive;
SVRMGRL> alter database open;
```

8) 应用README.doc文件中规定的目录脚本文件 (见第5步)。

9) 作为可选择的步骤，重新应用整个 catalog.sql、catexp.sql和catproc.sql脚本文件：

```
SVRMGRL> @catalog
SVRMGRL> @catproc
SVRMGRL> @catexp
```

数据库现在已完全升级为新的版本。

注意 像往常一样，在升级之前对数据库做一个全备份。

2.3 网络主机

当支持Oracle数据库的主机通过网络连接时，这些数据库可以借助于 Oracle Net8(称为

SQL*Net)进行通信。如图2-6所示，Net8驱动器依赖于本地网络协议来实现两个服务器间的连接。Net8支持两个服务器上应用层之间的通信。在本书的第三部分将对Net8进行讨论。

图2-6 Oracle Net8结构图

在网络环境中，数据库配置的有效选项取决于网络的配置和选项。下面几节将讨论以下几个主要的体系结构：

- 数据库网络，用于远程查询。
- 分布式数据库，用于远程事务。
- 并行服务器数据库，用于多用户访问同一个数据库。
- 并行查询操作，用于多个CPU为一个操作服务。
- 客户机/服务器数据库。
- 三层体系结构。
- Web可访问的数据库。
- Oracle Transparent Gateway(透明网关)访问。
- 备用数据库。
- 复制的数据库。
- 外部文件访问。

2.3.1 数据库网络

Net8允许Oracle数据库与通过网络可访问的其他数据库进行通信。所涉及的每一个服务器都必须运行Net8。图2-7展示了这种配置。

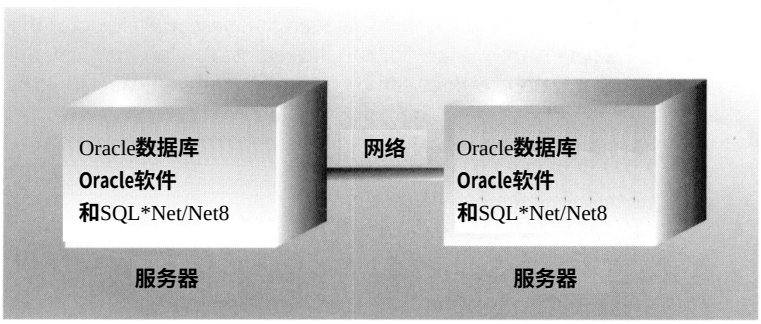

图2-7 数据库网络主机

在图2-7中，展示了两台主机。如前面图2-2、2-3所示，每个主机都能以独立方式操作数据库。这个例子中的每个主机都保存有Oracle软件的拷贝和一个或多个数据库。

对于进行通信的数据库，它们各自的服务器必须可以彼此通信。如图2-6所示，数据库层的通信依赖于网络软件及硬件来建立服务器间的通信链接。一旦建立通信链接，数据库软件便可利用这种链接在远程数据库间传输数据包。

这种用于数据库间数据传输的Oracle软件称为Oracle Net 8。在它的最简单配置中，它由通过特定连接路径等待连接的一个主机进程组成。当检测这些连接时，它执行通过连接传输来的指令并返回请求的数据。本书第三部分将对Oracle Net 8进行全面描述。

对于Net8的通信接收与处理，主机必须运行一个称作listener (监听器)的进程，这个监听器必须在与数据库通信有关的每一个主机上运行。每个服务器必须进行配置以指定此进程连接到一个特定的通信端口(见第14章的相应例子)。

有关使用数据库连接的例子，将在下面几节描述。这些例子包括对远程数据库的查询及远程数据库事务处理。

远程查询

对远程Oracle数据库的查询使用数据库链接(database link)来识别数据查询路径。一个数据库链接直接或间接地指定主机、数据库和用于访问特定对象的帐号。数据库链接通过引用数据库的服务名(service name)来识别进行访问的主机和数据库。当一个数据库链接被一个SQL语句引用时，Oracle就打开指定数据库的会话并且在那里执行该SQL语句。然后返回数据，并且远程会话可以保持打开状态，以备下一次使用。数据库链接可以是公共链接(通过DBA，使此链接对本地数据库中的所有用户都有效)，也可以是私有链接。

下面的例子创建一个名为HR_LINK的公共数据库链接：

```
create public database link HR_LINK
connect to HR identified by PUFFINSTUFF
using 'hq';
```

其中的create database link命令有以下几个参数：

- 可选关键字public，使DBA可以为数据库中的所有用户创建链接。
- 链接名(本例中为HR_LINK)。
- 要连接的帐户(如果没有指定，就在远程数据库中使用本地用户名和口令)。
- 服务名(hq)。

要使用这种链接，只需简单地将其作为一个后缀添加到命令的表名中。下面便是一个使用数据库链接HR_LINK查询远程数据表的例子：

```
select * from EMPLOYEE@HR_LINK
where  Office='ANNAPOLIS';
```

注意 数据库链接不能用于从LONG数据类型字段返回数值。

数据库链接允许查询访问远程数据库，也使有关数据的物理位置的信息——它的主机、数据库、模式——对用户透明。例如，如果本地数据库用户根据有关数据库链接创建一个视图，则任何对这个本地视图的访问都将自动查询相应的远程数据库。执行查询的用户不需要知道数据的位置。

下面的例子解释了这一点。在这个例子中，使用前面所定义的HR_LINK数据库链接创建一个视图。可以把对这个视图的访问权授予本地数据库的用户，如下例所示：

```
create view LOCAL_EMP
as select * from EMPLOYEE@HR_LINK
where Office='ANNAPOLIS';

grant select on LOCAL_EMP to PUBLIC;
```

当一个用户查询这个LOCAL_EMP视图时，Oracle将使用为HR_LINK数据库链接指定的

连接信息打开一个会话。这样就会执行这个查询并把远程数据返回给本地用户。LOCAL_EMP的本地用户将不知道数据是来自一个远程数据库。

2.3.2 远程更新：高级复制选项

使用Advanced Replication Option(高级复制选项)的数据库不但能查询远程数据库的数据，还能更新远程主机上的数据库内容。对远程数据库的更新可以与对本地数据库的更新合并成一个逻辑工作单元：或者一起执行提交，或者一起执行回滚。

图2-8展示了一个事务处理示例。一个事务针对远程主机数据库，另一个事务针对本地主机。在这个例子中，本地的 EMPLOYEE表被更新，远程的 EMPLOYEE表位于HR_LINK数据库链接指定的数据库中，并且作为同一个事务的一部分也被更新。只要一个更新失败，两个事务都要进行回滚。这是通过执行 Oracle的TWO-Phase Commit(双阶段提交)来实现的，此命令在第三部分详述。

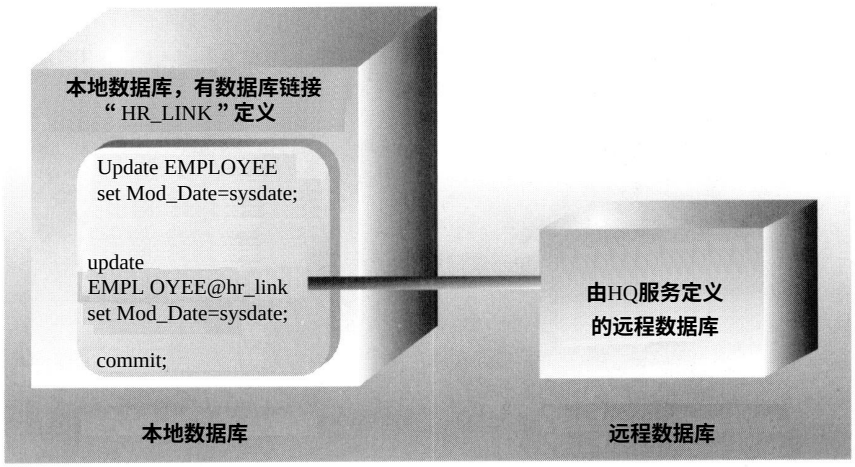

图2-8 分布式事务处理示例

此远程更新所涉及的数据库是在功能上相互分开的。它们中的每一个都拥有自己的数据文件和内存区域。它们都必须运行 Advanced Replication Option。所涉及的主机必须运行 Net8且必须配置成允许主机-主机通信。

在每一个主机都安装了 Net8之后，必须正确配置相关的所有文件。这些配置文件允许数据库解释本章前面create database link命令所示的服务名。

运行Net8的每一个主机都必须保留一个名为 tnsnames.ora的文件。该文件为可从主机访问的服务名定义连接描述符。例如，下面的例子展示了 HR_LINK使用的“HQ”服务名的tnsnames.ora文件条目：

```
HQ =(DESCRIPTION=
      (ADDRESS=
            (PROTOCOL=TCP)
            (HOST=HQ)
            (PORT=1521))
      (CONNECT DATA=
            (SID=loc)))
```

这个例子展示了连接进程的不同方面(其参数专用于TCP/IP，但基本的连接需求对所有平台来说都相同)。首先，存在一个硬件寻址信息——协议、主机名和使用的接口。第二个部分定义实例名——本例中为“loc”。由于tnsnames.ora文件可以向数据库提供连接远程数据库需要的所有信息，所以要保持这个文件的内容跨主机的一致性。有关影响位置透明性管理的情况，请参见第16章。

用于分布式事务处理的逻辑工作单元通过执行Oracle的Two-Phase Commit(2PC)来处理。如果网络或服务器失效，导致工作单元无法顺利完成，受事务处理影响的数据库中的数据就可能不同步。这时，就会自动执行一个后台进程来检查未完成的事务并且在所需的所有资源有效时解决存在的问题。

一个数据库所允许的最大并行分布事务量可以通过其init.ora文件中的DISTRIBUTED_TRANSACTIONS参数来设置。如果此参数设置为0，则不允许分布式事务处理，并且在实例开始时也不会启动后台的恢复进程。

下面两个数据字典视图有助于诊断未完成的分布式事务处理：DBA_2PC_NEIGHBORS视图含有关于未决事务的输入和输出连接的信息。DBA_2PC_PENDING含有关于等待恢复分布式事务的信息。如果分布式事务遇到错误，请详细检查DBA_2PC_NEIGHBORS和DBA_2PC_PENDING。

2.3.3 集群服务器：Oracle并行服务器

到目前为止，讨论的所有配置都是针对一个服务器访问的数据库。然而，根据硬件配置，有可能使用多个服务器访问一个数据库。这种配置称为Oracle Parallel Server(OPS，并行服务器)，如图2-9所示。

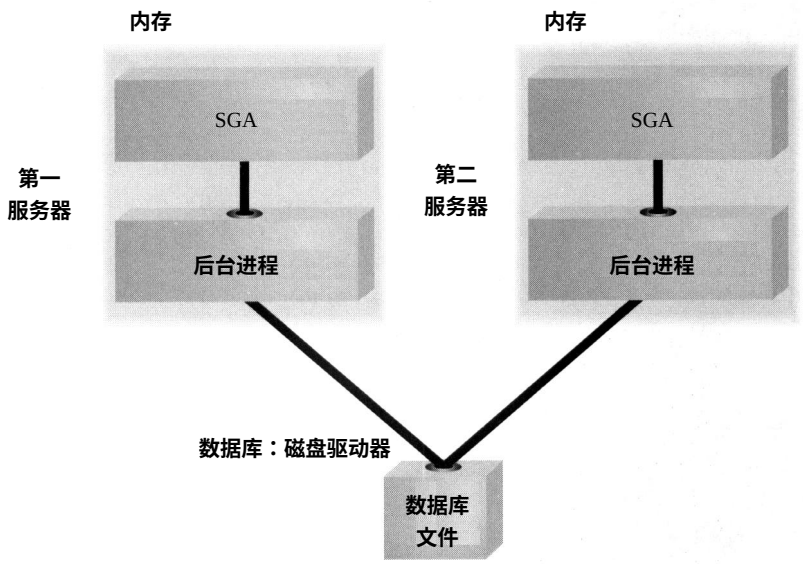

图2-9 Oracle并行服务器

如图2-9所示，两个独立的服务器共享同一套数据文件。通常，这些服务器位于一个硬件集群的不同主机中。集群一般是由各个主机组成的一个组，这些主机与一个高带宽、低等待

时间的互连结构连接在一起，相互传递消息，同时它们可以作为一个单独的实体进行操作。使用这种集群配置会有如下益处：

- 更多的有效内存资源，这是因为使用了两个设备。
- 如果一个主机停机，另一个仍可以访问数据文件，因此提供了一种对灾难进行恢复的手段。
- 可以根据执行的进程的类型对用户进行分组，并且大量占用 CPU的用户可以留在与常规联机处理事务分离的一个主机上。

除了这些优点之外，这一配置也存在一个明显的潜在问题 —— 如何解决两个服务器对同一个记录的更新。进行一个事务处理后， Oracle并不立即把修改块从 SGA写回数据文件。当这些数据块位于 SGA时，另一个实例有可能请求它们。要支持这种请求， Oracle将把这些块写到磁盘上，然后再从磁盘写入到另一个 SGA。这样就导致I/O非常频繁的数据库请求。

解决这一潜在问题的最佳方法是规划一个用户的分布方案，这一方案应当是根据用户的数据使用而不是 CPU使用制定的。这样，要求更新同一表的用户应使用同一个实例来访问数据库。

当设置一组使用OPS的服务器时，必须把一些数据库结构及参数指定给 OPS。

首先，必须配置用于处理不同服务器的中心数据库。它的基本要求是每个服务器都可以使用的一组回滚段。要最佳地管理它们，可以为每个服务器创建一个单独的回滚段表空间，并将服务器名作为表空间名的一部分。例如，如果服务器名为 ORA1和ORA2，则回滚段表空间名为RBS_ORA1和RBS_ORA2。

每个服务器必须在各自的 init.ora文件中指定自己使用的回滚段。第 7章将讨论这种激活回滚段的方法。还有一些必须为并行服务器设置的其他数据库初始化参数。由于它们中的许多参数对每一个服务器必须相同，所以必须使用 init.ora 文件中的IFILE参数。这就可以指定一个包含文件(include file)，该文件列出附加参数的值。如果有两个实例都引用同一个 IFILE参数，则不必为那些通用值而担心。

在一个OPS环境中，某些init.ora参数(如INSTANCE_NAME和ROLLBACK_SEGMENTS)必须对所有实例是唯一的，然而另一些参数 (如DB_BLOCK_SIZE)必须对所有实例是相同的。

有关这些参数的详细情况，请参见《 Oracle并行服务器管理员指南》。请注意，这些参数的数量受所使用的服务器数量的限制。由于这种关系，每当把一个新服务器添加到数据库的一组服务器时，都必须重新估算通用 init.ora参数。

2.3.4 多处理器：并行查询和并行装载选项

可以利用多个处理器来执行事务处理和查询。一个数据库的查询工作可以通过多个相互配合的处理器来完成。这种多处理器间的工作量分配能改进事务处理及查询操作的性能。

Parallel Query Option(PQO，并行查询选项)结构允许所有的数据库操作。可以利用 PQO的操作包括create table as select、create index、全表扫描、索引扫描、排序、 insert、update、delete和大多数查询。

数据库所使用的并行性程度由这些命令中使用的 prarallel关键字的degree和instances参数决定(参见附录 A)。其中degree参数规定给操作的并行程度 —— 所使用的查询服务器的数量。instance参数规定在一个OPS设备的实例中如何分割一个操作。可以在实例的 init.ora文件中规

定实例的并行性规则，如查询服务器的最小数量。 init.ora文件中的 PARALLEL_MAX_SERVERS参数设置了可用的并行查询服务器的最大数量，最小数量则由PARALLEL_MIN_SERVERS参数设置。存储表数据的磁盘数量及服务器上可用的处理器的数量用于为查询生成缺省并行性。

对于Oracle版本7.3，可以用PARALLEL_AUTOMATIC_TUNNING init.ora参数来自动设置许多与并行查询相关的 init.ora参数。只要数据库中有多个活动用户，自动调整参数PARALLEL_ADAPTIVE_MULTI_USER就将减少操作的并行性。因为对并行查询服务器进程的可用数量进行了限制，从而减少一个操作的并行性会防止其使用全部可用资源。如第 5章所述，也可以使用Database Resource Manager(数据库资源管理器)来限制并行性。

除了并行查询外，如果使用SQL*Loader Direct Path装入器的话，还可以并行地装入数据。有关协调成批数据插入的详细情况，请参见第 8章。

2.3.5 客户机/服务器数据库应用

如本章前面所述，在一个主机 -主机配置中，每个主机中都有一个 Oracle数据库，该数据库通过Net8进行通信。然而，没有数据库的主机可以访问远程数据库。一般是一个主机中的应用程序去访问第二个主机中的数据库。在这种配置中，运行应用程序的主机称为客户机(client)，而另一个则称为服务器(server)，图2-10演示了这种配置。

如图2-10所示，客户机必须有能力与服务器进行网际交流。应用程序在客户端运行，因此，这种数据库主要用于I/O。运行应用程序的CPU开销取决于客户机而不是服务器。

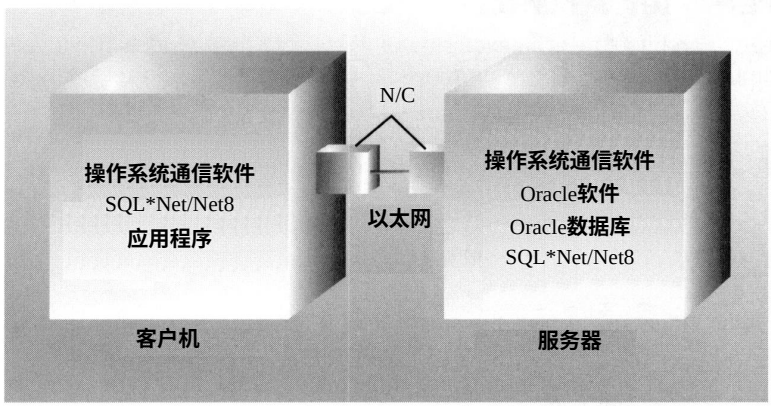

图2-10 客户机/服务器配置

要运行这样的配置，客户机必须运行 Net8或SQL*Net V2。当客户机应用程序提示用户输入有关数据库连接信息时，必须指定服务名。这样应用程序可以打开远程数据库中的会话。

使用客户机/服务器配置有助于减少服务器的工作量。然而，将一个应用程序移到客户机 /服务器配置，并不能自动改进系统性能，原因有两点：

- 以前，CPU资源可能不是个问题。通常， CPU资源的使用经常发生在工作时间而不常发生在非工作时间。可以通过在非工作时间运行批处理进程或者大型程序来改变 CPU使用的分布。
- 应用程序可以不再重新设计。对客户机 /服务器环境的设计，要考虑对每一个数据库访

问所需要的跨网络传输的数据量。在服务器应用程序中，这是没有问题的。而在客户机/服务器应用程序中，必须在计划和协调阶段考虑网络通信量问题。

实现客户机/服务器配置有不同的方法，这取决于硬件系统。图 2-10是一种通常的情况；它可通过一个特定查询工具在一台 PC机上运行，实现对一个服务器上运行的 Oracle数据库的访问。

2.3.6 三层体系结构

三层结构是客户机/服务器模型的一种扩展。每一层的功能都视你的实现而定，通常这三层的安排如下：

- 客户机，用于提供应用程序。
- 应用程序服务器 ，用于应用程序的业务逻辑处理。
- 数据库服务器，用于数据的存储和检索。

在三层结构中，应用程序的处理请求从客户机层传送到应用程序服务器层。通常，应用程序服务器比客户机的性能更强，所以应用程序的执行可能好一些。也可以通过减少数据库和客户机之间的通信量来提高性能。由于发送到客户机的数据较少，所以多数会话可能在应用程序服务器和数据库服务器之间进行。

三层配置的长远好处是可以简化对客户机的维护。因为应用程序保存在应用程序服务器上，大多数升级都将集中在升级应用程序服务器上。如果在你的客户机上没有过多的配置控制，三层结构就会减少应用程序的维护费用。

不过，从客户机/服务器结构移植到三层结构并非那么简单明了。如果你使用一个很好配置的网络、受控的客户机和良好协调的应用程序，转移到三层结构就可能没什么利益，而且会带来潜在的费用问题。例如，需要重新设计自己的应用程序以减少客户机和应用程序服务器之间的网络通信量；如果不能重新设计，就会影响应用程序的性能。进一步来说，如果升级应用程序(而不是升级客户机)的进程发生变化，就要升级被许多客户机使用的应用程序服务器。因为应用程序服务器的使用集中管理，在测试应用程序变化而不影响全部应用程序用户方面，你将遇到很多困难。不过，当更改应用程序时，只需一次更改一个服务器上的应用程序。

图2-11展示了一个Oracle应用程序的常规三层配置。应用程序服务器和数据库服务器通过SQL*Net进行通信。客户机和应用程序服务器之间的通信协议随操作环境而定。它既可以使用SQL*Net，也可以使用HTTP之类的互联网协议。

图2-11 三层配置

无论何种配置，应用程序都应避免客户机和数据库之间不必要的通信量。在三层应用程

序中，应特别注意数据库和客户机之间的通信量，因为它们之间有更多的组件。客户机和数据库之间的附加组件反而可能影响每一个数据库查询的性能。在许多三层应用程序中，客户机端几乎都不执行数据验证，从而减少了事务处理时所需的数据库查找数量。通过使用数值的下拉列表框或应用程序中的多选择复选框，就可能弥补缺少的数据验证查询。

Web可访问的数据库

可以把你的数据库配置成能够通过World Wide Web(万维网)进行访问。外部客户机既与防火墙服务器(中间层)进行通信，又将数据请求传送到数据库服务器。从结构上看，Web可访问的数据库基于图2-11所示的三层模型。例如，客户机层可能是你的个人计算机运行一个Web浏览器。浏览器通过HTTP与中间层(防火墙服务器)进行通信。如果数据库中使用Java，中间层可通过SQL*Net/Net8或通过互联网Inter-ORB协议(IIOP)与数据库进行通信。

有关Web配置(其中包括Web服务器的使用)情况，将在第17章中描述。

2.3.7 Oracle 透明网关访问

可以从你的数据库中访问非Oracle数据库。即，可以创建对一个非Oracle数据库的服务的数据库链接。然后可以通过这个数据库链接查询数据，就像源数据是一个Oracle数据库那样查询数据。

若访问非Oracle数据，就要使用Oracle Transparent Gateway(透明网关)产品。每种被访问的数据库引擎需要一个独立的网关。网关在被访问数据的源主机上运行。例如，如果源数据存储在一个AS/400数据库上，则AS/400的Oracle Transparent Gateway软件就安装在AS/400服务器上。执行时，网关软件在源服务器上创建一个监听器，其作用与SQL*Net/Net8监听器一样。如果有一个用户名和这个数据库的口令，就可以访问AS/400数据库中的特定数据对象。

Oracle Transparent Gateway是图2-7所示的服务器/服务器配置的一个扩展，不同之处在于：主机的数据库和数据库软件不是Oracle。

2.3.8 备用数据库

在Oracle7中，可以构造另一个数据库作为一个主数据库的“备用”拷贝。备用数据库是服务器/服务器配置的一种特殊情况。每个服务器都有一个Oracle软件的完整拷贝，并且数据库文件结构相同(如果不同，就要为备用拷贝建立一个独立的控制文件)。两个主机应使用同样的操作系统版本和数据库软件版本。

在产品数据库出现灾难事件时，可以打开几乎没什么数据损失的备用数据库——通常会丢失联机重做日志文件的内容。由于在Oracle8i中归档重做日志文件通过产品实例生成，所以必须把它们复制到备用系统中。备用数据库在未使用之前，一直保持恢复模式。一旦备用数据库打开，它就成为主数据库且不能轻易被重新配置为备用数据库。

若要把归档重做日志文件自动传送到备用数据库，可以使用Oracle8i中引入的LOG_ARCHIVE_DEST_*n*参数。用一个服务名代替目录名来作为归档重做日志文件的备用数据库拷贝的目标。

例如，产品数据库的init.ora文件可能包含下述内容：

```
log_archive_dest_1 = '/db00/arch'
log_archive_dest_state_1 = enable
```

第一个归档日志目标区是新的归档重做日志文件的主目标区。第二个 init.ora参数集通知数据库，把归档重做日志文件写入备用数据库的目标区。对于这个例子，备用数据库有一个STBY的服务名：

```
log_archive_dest_2 = "service=stby.world mandatory reopen=60"
log_archive_dest_state_2 = enable
```

LOG_ARCHIVE_DEST_2设置规定备用数据库的服务名。STBY实例必须可通过Net8访问。如果存在连通性问题，60秒后ARCH进程将试图重新打开连接。

有关建立和维护备用数据库的详细说明，请参见《 Oracle服务器管理员指南》。

2.3.9 复制型数据库

可以利用Oracle的复制特性在数据库之间拷贝数据。可以选择表、列和行并将它们从源数据库复制到其副本中。在这种结构中，复制环境是服务器 /服务器配置，如图2-7所示。

复制型数据库一般用于下述情况：

- 支持OLTP访问的多个站点。远程位置的大型数据输入可以从复制中得受益，因为全世界的多个站点都能同时执行数据输入。可以在数据库之间发送事务，以便全部相关数据库的内容都相当新。
- 创建只读或报表数据库及数据仓库。可以利用复制把自己的 OLTP数据库从所使用的数据库中分出来，以支持自己的报表需求。

上述第一个选项是多母版复制 (multimaster replication)选项：多个数据库可以对该数据进行更改，并且这些更改必须传播到网络中的其他数据库中。多母版数据库系统必须考虑在事务处理期间可能引起冲突的解决方案。

上述第二个选项是只读复制 (read-only replication)选项。在这个选项中，数据只按一个方向复制：从一个源数据库复制到一个或多个目标数据库。一般来说，只读复制的管理和构造比多母版复制要简单得多。可以把制表数据库的只读拷贝用作一个数据仓库的基础，或者用作一个测试数据库。由于这是一个独立的数据库，所以用户可以在副本数据库中创建不同的索引模式、表和进程。

有关配置和管理复制型数据库的详细情况，请参见本书第三部分。

2.3.10 外部文件访问

Oracle提供了许多与外部文件会话的方法。可以把外部文件用作数据源、脚本文件源或输出文件。最通用的外部文件的用途如下：

1) 用作脚本文件的源代码，写入 SQL*Plus、SQL或PL/SQL中。

2) 用作SQL*Plus脚本文件的输出，通过 spool命令生成。

3) 用作PL/SQL程序的输入或输出，通过 UTL_FILE软件包访问。

4) 用作一个PL/SQL脚本文件的输出，通过DBMS_OUTPUT软件包生成。

5) 用作通过BFILE数据类型在数据库中引用的外部数据。 BFILE数据类型含有一个指向外部二进制文件的指针。用户必须首先使用 create directory命令在Oracle中创建一个目录指针，指向存储文件的目录。

6) 用作通过DBMS_PIPE访问的外部程序。该程序必须以 Oracle支持的3GL来编写，例如

C、Ada或COBOL。

当应用程序使用外部文件时，应关注安全问题。 安全问题包括以下几个方面：

1) 文件中有硬编码的口令吗？反复执行的脚本文件必须有某种注册到数据库中的方法。

2) 这些文件保密吗？其他用户能读取它们吗？

3) 如果使用UTL_FILE，UTL_FILE使用的目录(通过UTL_FILE_DIR init.ora参数识别)保密吗？用户将可以看到目录中的全部文件。

4) 如果使用BFILE数据类型，相关目录保密吗？文件能抗修改吗？

如果应用程序依赖于外部文件，就必须确保用与数据库备份相同的操作方式备份这些文件。应用程序管理员和数据库管理员在备份其数据库文件时经常忽略备份外部文件。如果数据库因灾难而丢失，在恢复期间就可以使用数据库备份。但是如果没有备份相关的外部文件，应用程序就可能无用。

无论是否使用这种配置，都必须保证其可用性、可恢复性和安全性。当你将组件添加到自己的环境中时，就对环境添加了潜在的故障点。备份和恢复计划 (见第10章)、安全计划(见第9章)、监控计划(见第6章)和协调计划(见第8章)必须把硬件配置的所有组件都考虑进去。