

## 第3章 物理数据库的设计、硬件和相关问题

本章要点：

理解应用类型

使用定量评估

使非标准化

理解存储分层结构

理解RAID

理解DBMS中的瓶颈

选择平台

操作系统集成和通用内存 /CPU建议

物理设计原则和常用硬件设计建议

### 3.1 理解应用类型

在学习物理数据库设计以及以后进行的性能调整之前，了解主要的应用类型是很重要的。首先，考虑术语“事务”和“查询”。在数据库理论中，广义地讲，事务（transaction）就是一个单一的、原子的SELECT、INSERT、UPDATE或DELETE语句。但是，考虑到应用类型，事务被更为宽松地定义为一个业务处理过程，可能会包含多条INSERT、UPDATE或DELETE语句。另外，DML实际指的就是SELECT、INSERT、UPDATE和DELETE语句。然而，像文 中的事务一样，DML通常只被用于表示INSERT、UPDATE和DELETE操作。总的来说，DML和事务通常意味着只写或只修改。为区分作为只读的SELECT操作，要用到术语“查询（query）”。

看一下下面这些工业约定，尽管它们十分容易引起混淆，并且实际上和它们真实的定义相冲突。这里指出，在数据库系统应用领域有三个主要的应用类型：

OLTP（联机事务处理）——OLTP系统是一个包含繁重DML的应用，其面向事务的活动主要包括更新，但也包括一些插入和删除。经典的例子是预定系统，例如那些用于航空公司和旅馆的系统。OLTP系统可以允许有很高的并发性（在这种情况下，高并发性通常表示许多用户可以同时使用一个数据库系统）。

DSS（决策支持系统）——DSS系统通常是一个大型的、包含历史性内容的只读数据库，通常用于简单的固定查询或特别查询。DSS常常按某种方式变成一个VLDB、DM或DW，这些方式在第1章“数据库、DBMS 原理和关系型模型”中已经讨论过。DSS一个很好的例子是某个组织的局域网后面的数据库。

批作业处理——批作业处理系统是作用于数据库的非交互性的自动应用。它通常含有繁忙DML语句并有较低的并发性（在这种情况下，较低的并发性通常表示少数几个用户能够同时使用一个数据库系统）。事务查询的比率决定了如何物理地设计它，经典的例子是与DW有关的成品数据库和可操作数据库。

一些不太常用的应用类型包括如下：

OLAP（联机分析处理）——OLAP系统可提供分析服务。这意味着数学、统计学、集合以及大量的计算，一个OLAP系统并不永远适合OLTP或DSS模型，有时它是两者之间的交叉。另外，有些人简单地把OLAP看作是在OLTP系统或DSS之上的一个扩展或一个附加的功能层次。ROLAP代表关系型OLAP，尽管这个术语在分类方法上并不能真正增加什么。一个OLAP工具常被严格地和一个MDD（在第1章讨论过）配对使用，有时它只是简单地放在一个修改的RDBMS上。通常，地理信息系统（Geographic Information Systems，GIS）或有关空间的数据库和OLAP数据库相集成，提供图表的映射能力。用于社会统计的人口统计数据库就是一个很好的例子。

VCDB（可变基数数据库）——这种类型的数据库通常被用作一个处理系统的数据库后端，这样就会导致在数据处理期间，数据库中的表显著地增长或收缩，其相反情况是保持不变或在一定范围变化，例如一个10%的变化。基数是指在一个给定时间里一个表中行的数目。有些表可能是静态的查询表，但多数表的记录数目肯定是高度可变的，一个好的例子就是任何记录短期活动的数据库，例如一个安全授权数据库。

### 3.2 使用定量评估

任何种类的定量评估都是试图量化或测量一些过程或产品，对于数据库来说，定量评估的两种主要类型是事务分析（有时称为量化分析，volume analysis）和筛分分析（sizing analysis）。

#### 1. 事务分析

事务分析是简单地把数字放到数据库系统中。不同的评估方法意味着不同的事情，并应用于特定种类的数据库系统。最典型的度量或尺度，包括下列项目的最小数、平均数和最大数：

并发用户数目。

响应时间。

经过的时间。

事务数目。

并发程序的数目。

读或写的字节数。

还有更多的尺度，例如受一个操作影响的行的数目。

如果这些评估是在一个给定的时间段或动作的背景下表达出来，那么它们通常有着更多的含义。例如，知道每秒钟事务的最大数比知道事务的累积数更有用处。后者只能告诉你很少的关于数据库通常的压力或负载情况。这些数字在数据库为哪种应用服务的背景下也意味着更多的含义。

如果你的应用类型是一个OLTP系统，并发用户的数目、每秒钟的事务数目和响应时间尤为重要，因为并发是OLTP系统的首要问题。

如果你有一个批量处理系统，经过的时间和并发的程序数也许最重要。DSS可能要求你除了知道其他项目外，还应知道每个时间单位内可读的字节数。

作为一名DBA，你需要问这些问题并收集尽可能好的答案，因为这会影响到你的物理设

计和运行性能。另外，这些数字基本上就是用于衡量基准工作上的度量措施，那么，需要你努力去做的就是在系统真正建立之前，收集估计的、原型的或典型试验的数字以帮助建立系统。

## 2. 筛分分析

筛分分析也许是一种更广为人知的行为，甚至广泛地被所有的 DBA 采用。在事务或量化分析中，考虑到处理和数据流量，你会问“多久一次？”和“有多少？”；对于筛分分析，考虑到数据的存储，你会问“有多大”。

基本情况是：如果一个表有  $n$  行，每行最多有  $b$  个字节，那么至少需要  $n \times b$  个字节的存储空间。当然，这没有考虑系统的开销，而且这种计算随着所选择的商品软件不同而有很明显的差异。例如，像其他软件厂商一样，Oracle 提供一套相当复杂的步骤，帮助测量一个表、索引或其他结构。建议最好是一次把这个公式放入一个电子表格中，那么你永远不必再做这件事了。每次需要测量其他的数据库对象时，你需要做的就是取出它并插入相关的输入数字，如块的大小、块参数、行数、列大小等等。这样，你可以用表、表集合、索引、索引集合为整个数据库实现部分求和和分类汇总。

另外，一个经验丰富的 DBA 会在估算数值上加上一个裕量因子来对付任何低估错误、意外的疏忽以及未预料到的变化。将有些项目（如最后估计值的大小）乘以 120%，不是没有道理的。

也要记住，你所评估的数字通常是单独基于表和索引的。如同所有的 RDBMS 商用软件一样，有许多更具结构性的因素增加了整个系统的大小（在第 5 章“Oracle 数据库体系结构”和第 6 章“Oracle 实例体系结构”中讨论过）。不管你最后得到的字节数是什么，记住这是你需要的可用空间，而不是裸盘的空间。低级的硬件和一个高级的操作系统会消耗最初磁盘未格式化（裸盘）的大小，剩下的可用空间比你认为的要少。

例如，格式化可占用一个 4GB 磁盘中 15% 的空间，仅留下 85%（即 3.4GB）。如果你最终的大小估算是 20GB，并且没有考虑到这一点，于是你买了  $5 \times 4\text{GB}$  的磁盘，然而实际上你只有  $5 \times 3.4\text{GB}$ （即 17GB）的可用空间，那么你还可能需要其他磁盘。

## 3.3 使非标准化

使非标准化是指由于物理设计、性能调整或其他原因，使表的标准化形式的级别在一定程度上降低。

**提示** 除非你有非常好的理由，否则不要这样做。缺少磁盘存储空间而且又没有买新盘的预算或许就是一个很好理由。没有根据地判定将有一个不良性能，而不是实际存在不良性能，并不是一个好理由。实际上，即使是不良性能本身，在你的逻辑设计上也不会立即表明有放弃的必要。你所要做的第一步工作就是性能调整，非标准化一般是最后的办法。

那么，到底要做什么呢？假定在数据库中，有 30 个临时的表组成数据库的逻辑设计，这些表都是 3NF 或更高形式的。然而，当使用的时候，这些表中的三个表实际上总是要联合到一起。联合操作本身是资源密集的，并且只消耗很少的机器时间。出于对性能改善的要求，又没有其他性能调整的技术可用，那么为了长久地使用，你可能希望把这三个表预先联合成一个表，这样就可避免将来访问时的联合。把两个或多个标准化的表放回到一个表，实际上

的确降低了标准化级别或甚至完全变成非标准化的。当然，如果必须这样做，必须写入更多的过程化的和应用级的数据完整性控制，来取代 DBMS 的自动特性。

例如，如果你选择同时保留原始表和新的联合表，那么你将面临一个不麻烦的任务：使它们保持同步。然而，如果更新非常频繁（每小时或更短时间），并且要求联合数据的当前值，就不得不使用其他方法了。

另外一种方法就是在标准化的基表上创建一个非标准化的视图。然而，这个方案的性能与简单地拥有三个原始表相同或更糟。它唯一的好处就是用户接口简单。那么，尽管非标准化可能在一些静态数据事例中有帮助，但常常无法解决许多动态数据问题。在这种情况下，如果非标准化不能工作，Oracle 聚簇这样的方法可能就足够了。这第 5 章“Oracle 数据库体系结构”、第 16 章“性能调整基础”以及第 19 章“调整输入/输出”中将详细讨论。

最后，非常普通的一个非标准化例子出现在 DBA 必须处理时间序列时。时间序列通常必须被非标准化的原因，是因为时间必须总是表主键的一部分。这是正确的，因为大多数存储的数据是与时间相关的。

一个较好的例子是一个贮存卫星反馈数据的数据库，这样的数据带有时间信息，它是数据库中大多数表的主键的一部分。对于任何给定的包括时间信息成分的表，所有组成主键的其他列（如卫星 ID）对于每个不同的时戳来说是重复的。换句话说，如果数据库每小时从一个卫星下载 100 行数据，那么在 8 小时内，你要毫无必要地重复 800 次以存储 800 行数据以及其他的卫星信息，这是对存储的极大浪费，尤其是考虑到可能有多个卫星或更密集的取样频率。

典型的解决办法就是通过把数据从行方式转换为列方式，实行非标准化。现在你有 100 个带有时间信息的列，并且你把行数从 800 减少到 8，或者使用一个取样间隔为 100 的因子。存储的减少，尤其是行的减少，总能有助于提高搜索性能，因为在扫描一个表或一个索引时，扫描的行数减少了。然而，这种类型的非标准化，尽管通常是必要的，结果却导致了一个不标准的表，因为时间信息被作为重复组列贮存。因此，在先前为主键组成部分的列上，你不能用外键约束。相反，你必须依赖于检查约束、过程、触发器以及可能的增补应用完整性来处理有关操作。你应从这个特殊的例子中学到教训是：如果出于性能因素，你必须实行非标准化，那么你必须在完整性管理上付出代价。

### 3.4 理解存储分层结构

数据库系统管理员能够学到的最重要的事情之一就是存储分层，以及与其相关的折衷方案。这比其他任何事情更能帮助你解释许多关于物理设计和性能调整问题。存储分层背后的一个关键思想就是机电设备的不足（electromechanical disadvantage）——即任何有引擎或运动机件的东西，天生就比纯电子设备的速度慢。

图 3-1 表示了存储分层的现代解释。很清楚，内存比磁盘速度快。沿这个金字塔向上，速度提高（存取时间），费用也增加（每单位），存储量降低。

你想贮存的越多，必须付出的也越多，尤其是你希望得到更快速度时更是如此。但是，你可以取一些折衷方案。对于一个历史数据系统，你可以把比较旧的数据存放到联机光盘或至少较慢的磁盘上，而将更老的数据存放到联机或脱机磁带上。如果你特定的存储金字塔能让所有的存储介质自动联机、下载到磁带，并且你有按照要求访问它的软件，那么这就是一个分层存储管理（Hierarchical Storage Management, HSM）系统。有专门研究这类硬件和软

件的厂商。历史数据系统能允许向金字塔底部移动，因为相对于数据流通，速度并不重要。但是，对于实时系统（例如飞机导航等），将存储加快是最优先的考虑因素。许多商务应用的一个合理的方法就是把重要、当前的数据放在 RAID 或快速磁盘上，而将其他东西放在较慢的磁盘上。

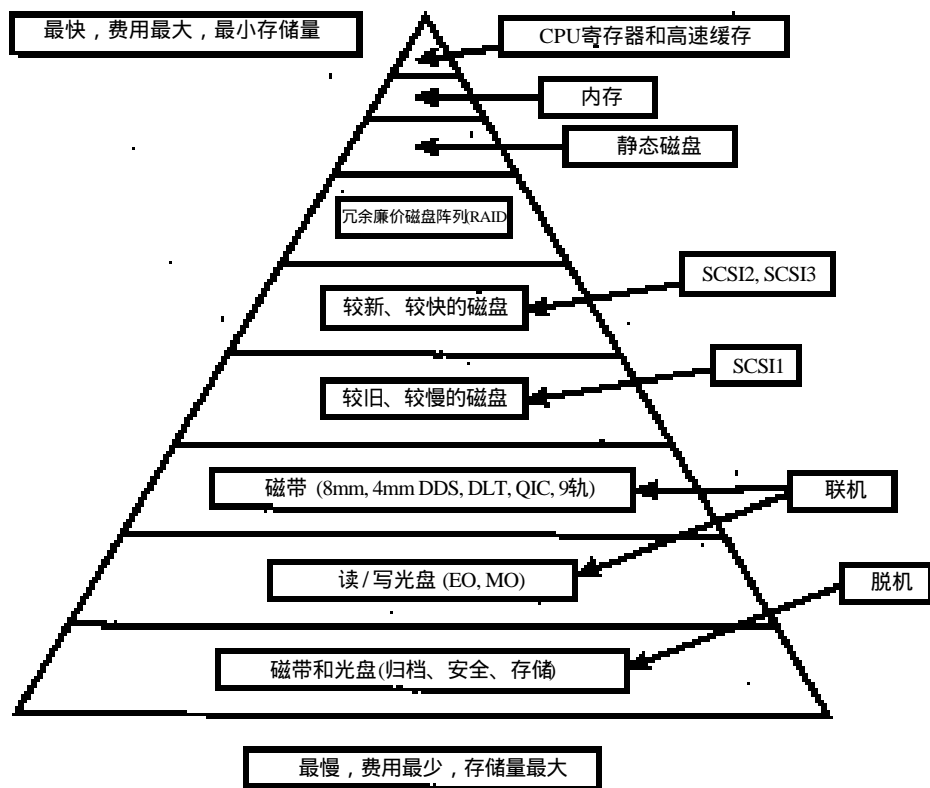


图3-1 存储分层

### 3.5 理解RAID

RAID（冗余廉价磁盘阵列，Redundant Array of Inexpensive Disks）也许是一个误称。从一开始，组成 RAID 的磁盘从未真正地比普通的 SCSI（小型计算机系统接口）盘便宜过。另外，RAID 需要特定的硬件、软件或两者皆有来工作，这都增加了成本。所以 RAID 中的 I（廉价）不十分确切。

RAID 是一组能并行工作的磁盘，能够减少 I/O 时间，其效果主要取决于组成磁盘组的磁盘数量这一因素，对数据库来说这是件重要的事情。RAID 通过人们所熟知数据条（striping）技术来实现并行工作，与普通磁盘存储有所区别，后者是串行工作的。这只是以并行方式跨越多个磁盘使用数据条来写一个文件或文件块。数据条是一个物理数据块的一定倍数，是磁盘 I/O 操作的最小单位。通常，一个数据块由 512 个字节组成，这对于大多数 UNIX、VMS 和 DOS/NT 系统来说都是正确的。

RAID 还提供实时磁盘失效恢复，这是数据库的另外一件重要的事情。RAID 可以通过奇偶信息提供这一级别的可用性，这种信息也被写到一个或多个磁盘中。奇偶校验、校验和、

以及错误纠正是通过一定的数学算法实现的，这些算法能重建在丢失磁盘上的丢失数据。RAID既支持大磁盘组也可以支持小磁盘组，可以有助于不同的数据库应用类型。例如，DSS系统使用较大的磁盘通常运行得更好，而OLTP系统使用较小的磁盘运行得较好。RAID分为很多形式，被称为级别。

当今在工业上最常使用的RAID级是0、1、3和5。所有其他的级别都没有被使用或只是这4个级别的一些变种。供应商提供的RAID级达到了7或更高，但在实际工作中很少看到它们。RAID 0是没有校验的基本数据条。就是说，你得到了性能上的优点但是没有校验。当纯粹为了速度而可用性并不太重要时，这是很好的。

RAID 1就是大家所知的镜像（mirroring），或者有时叫双工（duplexing），在镜像中同样没有校验。实际上，对于偶数数目的磁盘组，其中的一半是保存实际的或主要的数据，另一半基于磁盘级保存数据的备份。实际上，两部分磁盘是同时写入的，并且有时这意味着写入时性能的损失。另一方面，在读的时候，如果软件被设计成能读多个盘并且能将数据流合并使用，那么实际上读操作是可以被加速的。

RAID 3是数据条，带有一个单一、专用校验盘。你可以丢失一个数据盘，并且仍然可以用校验盘来恢复它。不过，校验盘是一个单一故障点。如果它丢失，将是致命的，这依赖于软件或硬件以及它是如何写入的来处理这类事件。

RAID 5也是带有校验的数据条，只不过不是用一个单一、专用磁盘，它将校验信息与数据一起保存在所有的磁盘上。校验信息和数据一样受到保护，并且没有单一的故障点。和RAID 3一样，RAID 5能够在单个盘的丢失时恢复数据。然而，3和5都不能恢复两个磁盘的缺失，因为丢失了太多的校验信息。

工业上很少销售或实现RAID级别2、4、6。这些级别的校验信息背后的数学计算量太大而不实用。

关于RAID最后要说明的就是：对于数据条，RDBMS和OS工具及技术手工能做的事，RAID都能做。但是，它通常做得更好，它有精密形式的数据条并提供更多的特性，如分区容量（数据条集合）、可调整大小的数据条等等。另外，尽管RAID的数据条部分可以毫无困难地以一种总体的方式来实现，但你必须管理这些数据条。RAID通常擅长于此，并提供一个虚拟文件系统（Virtual File System，VFS）的单一文件接口，协调OS文件级的命令和工具并且提供了其他特性，如大于物理磁盘尺寸的文件。

最后，尽管你可能可以管理某些形式的数据条，但是除了有标准的备份和恢复机制外，并没有其他保护措施。但是RAID可以容忍丢失一张磁盘并进行恢复，你要试图自己编写这样的软件实在是太复杂了。

提示 当可用性不是一个主要关心的事时，可以使用大致的、手工的RDBMS或OS数据条。否则，当你需要它提供的可用性并希望使用更为精密的数据条以获得更佳性能时，或需要它的其他特性（如巨型文件）的时候，要使用RAID。

### 3.6 理解DBMS中的瓶颈

过去，DBMS常常被指责为I/O限制或硬盘限制。这就等于说一个DBMS由于它对磁盘的读出和写入，导致它在系统中成了瓶颈。研究一下你刚刚学到的存储分层就会发现这几乎是不相关的。访问磁盘的速度要比访问内存或CPU的速度慢得多，这实际上就是过去的数年里

许多数据库应用的情况。对于 DSS、VLDB 和 DW 来说，这尤为正确。因为，对于一个查询来说必须移动大量的数据（GB）。然而，这并不永远是事实。OLTP 和 OLAP（在线分析处理）系统常常是内存限制或 CPU 限制。

如果一个数据库硬件服务器有一个较低的内存对硬盘的比率，意味着应用是内存短缺的，数据库性能降低显然是由于存储数据和缓存代码的内存空间不足。如果一个 OLAP 类型的数据库应用（比如一个科学数据库），那么速度是最重要的事。因此，这种类型的应用如果运行在带有相对较弱 CPU 的服务器上，按照用户的期望（如响应时间）来讲，性能肯定是低劣的。这里需要知道的就是在任何系统（包括数据库系统）上，瓶颈现象是与应用类型相关的。

最后，在客户/服务器数据库系统中，网络系统是整个系统中最慢的组件，甚至比硬盘还要慢。为了有效地发挥一个客户/服务器系统的功能，必须有合理的应用分段并且网络系统一定不能超载。另外，和其他资源一样，网络系统必须被合理设计，以减少竞争。还有，为利用当前网络的能力，网络硬件和软件应该实行现代化。你将在第 16 章“性能调整基础”中再次了解瓶颈问题。

### 3.7 选择平台

当选择适合于应用的硬件数据库服务器和操作系统的类型时，有许多事情需要考虑，包括如下：

应用类型——正如已讨论过的，OLTP、DSS、批任务处理或其他东西。

定量估计——正如已讨论过的，关于事务的、量化的和筛分的数值。

当前环境——主要指你当前的主要平台是什么。

趋势——对于你当前的环境，工业发展的前进方向是什么，以及你正在考虑什么。

处理的需要——需要是实时的、对时间要求不太严格的还是周期的？用你的事务和量化数值来说明并发及加载。

存储需要——用你的估算数值来确定对裸盘的需要，用事务和量化数值来确定你是否需要 RAID。

人员能力——你认为哪些人员能力要求与你的人员的基本能力相差甚远。

时间限制——在给定时间里，你能移植或开发出系统吗。

衔接和合伙——RDBMS 供应商是否与 OS 及硬件供应商有融洽的伙伴关系。

集成——与前一条相似，但是如果不考虑他们之间的业务关系，那么将所有的工作放在一起会有多好呢？

目前主要的 RDBMS 供应商是 Oracle、Sybase 和 Informix。CA 在 Ingres 上做的多一些，微软只提供 NT 环境的 SQL 服务器。这里不考虑小型、桌面数据库。现在，只考虑 Oracle，UNIX 和 Windows NT 是它的两个主要平台。对 UNIX 来说，平台主要是 Sun Solaris。当然，Oracle 也可移植到其他形式的 UNIX 平台。Oracle 在 MVS 和 VMS 系统上也有一些大的安装数量。然而，写本书时，Solaris 和 Windows NT 是它目前最优先选用的系统。

至于硬件，不算 Intel 公司的机器，Sun 公司的 Solaris 系统是唯一的真正选择。Sun 公司的机器已经经历了几代变化，但是它们均基于 RISC 的 SPARC 芯片。Windows NT 主要使用在 Compaq 机器上，但也应用在 DEC 和 HP 的机器上。在 DEC 上，关于 Windows NT 最重要的事就是 DEC 支持 Alpha 和 Intel 机器。Compaq 最近对 DEC 的购买，简化了软件提供商这种进退两难

的局面。Solaris和Windows NT都可以在多处理器系统上运行。但是，Windows NT只能处理有限数量的处理器（14），并且用于处理Oracle的处理器能力大约为4个。Solaris能非常好地扩增，可以超过20个处理器，即使对于Oracle也是如此。虽然Sun公司比Compaq、HP公司更容易提供更大的选择范围，但是磁盘存储选择几乎是相同的，当然DEC公司也能提供和Sun一样大的存储。

不必更深一步地比较硬件，这里有一个建议：作为一套常规原则，在准备选择一个平台之前，列出它们的各项指标，并充分考虑这些指标。如果你的主要环境是DEC/VMS，并且你有可能保留这种环境达几年时间，那么，你的下一个平台最有可能还是DEC/VMS，特别是如果你只打算增加一台机器时更是如此。如果你准备为整个部门购买机器而且风险也不是关键问题，这是考虑选用Windows NT或者UNIX的最佳时机。

如果你只需一台机器，且服务于相对较小的低于10GB的数据库，考虑选用NT较好。相反，如果你需要一台机器，但是对存储要求较大，或者对性能要求较高，考虑选用UNIX较好。最后，如果你没有一个真正的主导环境，或者你正在启动一个新计划（或这个新计划在某个方面影响你的购买），务必既要考虑Windows NT也要考虑UNIX，还要考虑除了环境因素之外的所有因素。

当前，最后一个可供考虑的是Linux。它起源于UNIX，当前得到一些支持，是Windows NT的一个直接竞争者，因为它既是一个操作系统，又是一个网络操作系统（Network Operating System，NOS）。因为它还有待于去建立硬件、应用和围绕它的技术支持，所以还没达到大多数工业销售的可信级；但是，它的普及程度还没有达到巅峰，决不可小视。

### 3.8 操作系统集成与通用内存/CPU建议

除了辅助存储、硬盘和RAID外，当讨论存储分层时，需要强调的是：你需要考虑其他的硬件问题和操作系统组件，如内存和CPU。

注意 内存通常是指核心内存、物理内存、主存或随机存取内存（RAM），这些都是同一事物的不同名词，所以这里只说内存。

本质上说，内存是一种非常快的电子存储器，它贮存指令和数据。对于一个DBMS来说，最重要的事情就是操作系统能够让出一些内存给它，那么，DBMS用它来做自己要做的事。因此DBMS有时被称为微型操作系统、操作系统中的操作系统，或一个建立在操作系统上的操作系统。实质上，虽然要服从操作系统并与之协作，但DBMS在管理和获取资源需求方面，能够照顾自己。这些事情通常是通过一个叫做共享内存功能来实现的，尤其是在UNIX环境中（详情参见附录A“在Solaris系统上的Oracle”）。

加锁是DBMS的一个关键组成部分，它也是通过内存结构来控制的。共享的资源没有进程竞争的危险，因为它们被依次使用。DBMS控制它自己的锁，部分工作和操作系统一起来做，或将加锁任务交给操作系统完成。

当操作系统将它的部分内存出让给构成DBMS的一个进程时，DBMS从那里得到内存，并在这个内存空间里贮存它自己的结构（代码缓存）和数据（数据缓冲）。在第5、6章中将讨论Oracle的结构，没有深入了解Oracle结构的人需要看一看在那里究竟讨论了些什么。Oracle的内存存取基于它的系统全局区（System Global Area，SGA）的资源分配。SGA包含一个叫做数据库高速缓冲（高速数据缓存）的结构和共享池。共享池包含库缓存（高速代码缓存）以

及数据字典缓存。撤销（回滚）块存储在数据块缓冲内，重做块存储在它自己的重做日志缓冲（日志缓冲）部分。通过 Oracle 的参数文件 ‘init.ora’，这些组成部分都是可以配置的，后面将会详细介绍。

关于 CPU 的使用，RDBMS 已走过了一段长路。像前面提到的那样，多数过去的数据库系统趋向于 I/O 限制。然而，有了 VLDB 和 OLAP/MDD 系统，越来越多的数据库变为内存限制或 CPU 限制。有了 VLDB，由于内存数量太小难以容纳大量数据，内存成为瓶颈。对于有大量的分析或科学的系统或者 DW 系统，由于庞大的、并行的计算要求，CPU 可能会成为瓶颈。

在近十年中，随着多处理器机器的出现和改进，许多事情发生了变化。现在，非常大的内存（几十 GB）已成为可能，而且，CPU 的体系结构和速度也大大提高。在写本书的时候，目前的字长是 32 位和 64 位，时钟速度大约是 500MHz，并且很快将达到 1GHz。这些 CPU 拥有密集的管线体系结构，允许每个指令周期（CPU 步）执行多条指令。最重要的是新 RDBMS 软件也跟着推出。

Oracle 和其他主要的 RDBMS 供应商一样，为充分利用这些先进硬件，有可能重新编写它们的过时代码。除了共享内存和非常大的内存之外，多处理器是近年来主要的提高和改进，目前存在着两种主要的多处理器：

对称的多处理器（Symmetric MultiProcessor，SMP）

大规模并行处理器（Massively Parallel Processor，MPP）

在 SMP 机器中（例如那些 Sun 公司提供的），CPU 使用共享内存和其他内部硬件项目（如总线）。现在 SMP 已达到 64 个处理器。MPP 机器中有一个完全不共享体系结构，并且类似于微型局域网（micro-LAN）或一个盒子中的局域网。MPP 可以拥有数百个或数千个处理器。

现在，RDBMS 软件或是完全多线程的或是伪多线程的，以便利用多处理器机器的处理能力。对一个操作系统来说，RDBMS 就是一个或多个进程。多线程是软件的一种能力，在相同的父进程环境下，能够运行多个子进程或线程。例如，Sybase 和 Informix 就是完全多线程的。当使用多线程服务器（MultiThreaded Server，MTS）选项时，Oracle 是伪多线程的；否则，它是单线程的。数据库系统管理员只需要知道 CPU 的数目，并且能够配置 Oracle MTS 就可以了。其他的 Oracle 参数受 CPU 数目的影响（例如，被称为门锁的特定加锁结构）。

### 3.9 物理设计原则与常规硬件设计建议

物理数据库设计的主要原则是什么？物理数据库设计实际上就是预调整，或者说是调整的第二个阶段（逻辑数据库设计是第一阶段）。因此，主要的物理数据库设计原则基本上与主要的性能调整原则相同，只不过你正在处理的数据库是在数据库创建之前或期间而不是创建之后。有许多设计原则，但主要包括如下几条：

分而治之——分区、分段和并行是分而治之的算法设计方法的扩展。如果一个处理时间被分成数块并能够同时运行，那么就说它是可并行的。对此的主要要求是处理的每个部分必须是数据无关的；即：不管任何其他程序块是否已结束，某一程序块都可以开始。例如把一个求总数的运行时间很长的查询程序分割成两块，在不同的 CPU 上运行它们，然后把它们的部分和相加，得到最后的结果。实际上，这就是 Oracle 的并行查询功能所提供的东西。

预分配和预编译——静态分配和固定分配都表示相同的意思，即预分配。换句话说，

提前分配你的资源，而不是动态地由软件为你做这件事。这通常导致额外的计算和 I/O 开销，而这总是不受欢迎的。预编译程序比解释程序执行起来要节省实际时间。虽然 DBMS 缓存控制许多事情，但是 DBA 应该注意可以做些什么以进行补充，例如编写一些通用、可重用的过程并把它们驻存在内存中。Oracle 的 KEEP 操作可实现后者，KEEP 操作将在后面的第 18 章“调整内存”中讨论。

前摄——预测主要的问题。遵循帕拉图规则：解决可能引起 80% 麻烦的 20% 的问题，这也被称为 20/80 规则。用统计的方式来说就是所有的性能问题并不是由各种原因平均引起的。你应尽力预测最主要的问题，并且把它们设计在系统外或至少补偿并围绕它们进行设计。考虑一个大型串行运行的批处理作业系统，只有几个主要程序，每个都可以插入、更新或删除大量数据。这里潜在的问题与事务日志有关，尤其是撤销日志增长得非常快，甚至有可能耗光存储空间。对于 Oracle，撤销日志就是一组可用的回滚段。对于一个 DBA 来说，合理的设计将至少有一个非常大的回滚段，而不是能够控制产生的最大数量回滚数据的 SYSTEM 表空间。

批量、块和批处理——使用大量传送。成批的事务在一起时进行批量处理才有意义。这就意味着对硬盘和网络 I/O 这样的事物来说，常用的最好方法就是大量传送：将有着相同的起源和终点的 I/O 操作组合在一起。这项工作通常是 DSS 和批量处理系统而做的。例如，用户可能经常从一个极大的表中选择多个行。因为这些选择返回如此多行，因此他们从来不用任何可用的索引。如果这时没有得益于任何并行硬件或软件，那么表应该连续地存放在一个单独的、大的磁盘上。然后，你可以增加数据库逻辑缓存的尺寸，以同时读许多物理数据块。例如在 Oracle 中，你可以把 DB-BLOCK-SIZE 设到平台所支持的最大值（如在一个 Windows NT 机器上设到 8K），这样一次读请求便读到尽可能多的数据块。这些都源于一个简单的事实，即在几乎所有机电系统中，启动耗费是相当大的。一个类似的网络情形是在每一个包里发送尽可能多的信息，因为这会提高成本效率。

合理地分割应用——这可以看作是“分而治之”原则的小主题，它的区别在于所强调的东西是整个的环境和应用程序，而不只是它背后的数据库（后端）。在分布应用前，考虑客户、网络和服务器的相对性能能力。把功能放在逻辑上能被执行的地方，例如：在一个交互式的应用中显示和表现的任务显然是属于客户的；而作为数据库对象的数据库事件应在数据库中处理（通过 DBMS），而不是由一些前端的软件处理。在 Oracle 中，使用了触发器和存贮过程进行这种设计。

物理设计和调整的主要目标是消除或最大限度地减少竞争，竞争（contention）就是两个或多个软件争夺同一个资源。常识告诉你有些东西必须等待。为了解决竞争，实践下列的规划建议：

- 把表和索引分开。

- 把大的表和索引放到它们自己的盘上。

- 把经常联合的表放在单独的盘上，或把它们聚合。

- 必要时，把不常联合的表放在相同的盘上（就是说，如果你的磁盘不多）。

- 把 DBMS 软件与表和索引分开。

- 把数据字典与表和索引分开。

如果可能，把撤销（回滚）日志 和重做日志分别放到它们自己的盘上。

为撤销日志或重做日志使用 RAID 1（镜像）。

为表数据使用 RAID 3或5（带有校验的数据条）。

为索引使用 RAID 0（没有校验的数据条）。

正如前面所讨论的，在某些情况下，如果 RAID是不可用的，那么手工数据条可以在某种程度上满足需要。但是，一般来讲，它不够灵活、安全或快速。另外，当前面的列表提到要进行分离时，那是表示把它们放到单独的盘上；但是，这又可能进一步意味着把它们放到单独的磁盘控制器上。控制器越多，性能和安全性就越理想。