

## 第20章 Oracle 8.x调整问题

本章要点：

分区

唯一索引表

并行数据控制语言

大池与非常大的内存

网络增强

数据库限制

第16章适用于任何厂家的 RDBMS 软件或版本，并且第 17~19 章中的大部分适用于 Oracle 7.x 和 Oracle 8.x 的 RDBMS，而本章几乎是专门适用于 Oracle 8.x 的 RDBMS 的。你将了解一下输入/输出、内存和网络关于性能的改进。

### 20.1 分区

当然，在 Oracle 7.x RDBMS 基础上的 Oracle 8.x RDBMS 的输入/输出性能的主要提高是真正的分区。在第10章中讨论了一些 Oracle 7.x 分区视图与 Oracle 8.x 分区的不同之处。比较明显，Oracle 8.x 有了某些性能上的优点。鉴于 Oracle 7.x 分区视图依赖多重表及那些联结起来的表的视图，因为确实没有新结构，所以没有本地语言支持。因此，所有事情（锁、优化等等）中输入/输出的级别仍然在表级。相反，在 Oracle 8.x 中，分区不仅是新的存储结构，而且被认为是本地语言，并且是优化的。

分区为性能带来的一个主要好处是分区消除。如果一个表在某一列（分区键）上被分区了，任何一条给定的查询（也就是 SELECT 语句）都可以被那些分区中仅仅一个分区中的行满足。一个例子是你已经把雇员在薪水等级列上进行了分区。如果你希望检查某个薪水等级，只需要存取一个分区以获得那些所需要的行即可。Oracle 优化程序知道这个，因为它可以把你需要的值域与那些你以前用来分区表行的值域相匹配。如果这里有  $n$  个分区要搜索的话，分区消除可以潜在地把性能加速  $n$  倍。

当然，就像 RAID 和 Oracle “人工”数据条一样，分区是一个分离存放的物理设计策略（参考第3章）。如果有  $n$  个分区需要存取，并且过程并行与数据并行同时发生（例如，并行查询），那么速度的提高可以接近  $n$  倍。然而，全部性能提高中还要减去再结合的总开销。另外，还有 Oracle 过程的某些并行方法、并行查询（Oracle 7.x 和 Oracle 8.x）及本章 20.3 节中的并行数据处理语言（Oracle 8.x）。

另一个需要考虑的分区问题是它在表和行之间增加了一个锁的级别。增加锁的级别几乎总是有帮助的，特别是如果它的级别比表低的话。考虑一下接近行总数的  $1/5$  的多行锁的开销，同一个任务一个单独的分区锁就可以完成。因此，当需要行级锁时，它们被授予。但是当需要一个分区锁而不是一个表锁时，你就不仅节省了许多行锁的系统开销，而且对未锁定分区

提供了更多的并行存取。表锁在语法分析时确定，之后在运行时得到。分区锁都是在运行时确定和得到的，而表锁在任何时间都可以。

在Oracle 8.x中，有一个日志/无日志属性，此属性在表空间、表、索引或者是分区级别的某些SQL语句中是适用的。无日志属性代替 Oracle 7.x中的不可恢复选项。然而，为了以后的兼容性，目前仍然支持不可恢复选项。由于不可恢复选项是与一条语句联系在一起的，因此，在数据库对象中要指定无日志属性。例如，所有的插入（INSERT）操作（串行或并行的）不产生重做日志项，同时不记录实际上要受影响的对象。另一个例子，CREATE TABLE和CREATE INDEX可以使用不记录它们影响的对象，因为它们已经在它们的语句中使用了不可恢复的对象。此外，INSERT...SELECT和CREATE TABLE AS SELECT是可以利用无日志属性的潜在的操作。

直接载入操作作用并行方式支持；然而，它们也可以是串行的。直接载入与 SQL\*Loader直接路径模式类似。直接载入只是对插入（INSERT）操作而言。它们使用临时段，它们将临时段直接追加到高于目标对象的高水位标志（HWM）的地方。对象的HWM被调节到新水平，同时新近插入的行被给定了ROWID。这种类型的载入与通常的插入方法不同，在写入目标对象的数据文件前必须经过与SGA相关的自由表。INSERT/\*+APPEND\*/提供了一个使用直接载入的优化程序提示，这可用于实现对象的直接载入。如果不需要直接载入，那么就不需要做些什么特别的事情，除非由于某些原因你想要完全保护它 那么就用NOAPPEND提示。

## 20.2 唯一索引表

有了基于B树变体的数据结构的普通 Oracle索引，你基本上有该索引和表本身，这个索引是一个表的排序子集。用 Oracle 8.x，你可以有唯一索引表，它们实际上就是表，这种表本身也有索引。这就是说表和索引的结构是一个而且是相同的。作为比较，Sybase总是在被物理排序的表中聚簇索引，这与被另一个索引结构逻辑排序的表（在 Sybase术语中叫无聚簇索引）形成对比。因此，Oracle唯一索引表模拟Sybase的聚簇索引，普通 Oracle索引模拟Sybase的无聚簇索引。

为什么需要唯一索引表？如果使用正确的话，普通 Oracle索引（B树）对大多数数据库应用程序是非常有效的。然而，它们确实有一些缺点。首先，它们冗余地存储索引列。因此，性能与键大小成反比。其次，B树总是至少需要两个逻辑读（通常更多）：一个用于读索引，一个用于读表。这种情况如何能得到改善呢？一个很自然的扩充就是唯一索引表（IOT）。

在用这些索引表时有一些限制和需要考虑的事情。一个唯一索引表只能被物理排序一次，它被主键限制。一个IOT必须有一个主键。Oracle唯一索引表（和Sybase的聚簇索引一样）是一个倒排索引或倒排列表的RDBMS例子，它们是在3GL和现代DBMS的文件处理系统中使用的数据结构。而且，IOT既不会被分区也不会被复制。也许考虑使用IOT时最重要的问题是它没有ROWID！例如，因为没有ROWID，在IOT上建立其他索引是不可能的（这与Sybase相反，Sybase在一个聚簇索引表上可以有第二个索引）。IOT不能是索引簇或哈希簇的一部分。另外，除了主键外的唯一性限制不能被实施，因为对于 Oracle唯一性限制（例如主键）是通过索引实施的。

清单20-1给你一个建立唯一索引表的例子。

清单20-1 建立一个索引表

```
SQL> create table employee
1 (eid number,
2  ename char(35),
3  eaddr varchar2(80),
4  resume CLOB)
5 primary key (eid)
6 organization index
7 tablespace data1
8 pctthreshold 90
9 overflow tablespace data2;
```

注意CREATE语句的几个事情。在一个给定的表空间（ data1 ）中在主键（ eid ）上创建唯一索引表（ employee ），并通过使用organization index子句指定它是一个IOT。更进一步指定如果任何行超出了Oracle块长度（ db\_block\_size ）的90%，就要把那些行的非键列存储在溢出表空间（ data2 ）中。这个溢出区域函数非常像链接行的溢出区域，该区域包含指向溢出段的溢出指针。在dba\_tables和user\_tables视图中的IOT列有一个新的IOT列，这个列被设置为IOT、IOT\_OVERFLOW 或者是NULL（对常规的或者非IOT表）。

20.3 并行数据控制语言

我们已经在第10章中简要地讨论了分区所具有的并行特性。本章稍微深入一点，但不特别限定在分区的并行性。有三种主要类型的并行粒度，每一种适用于它自己的一组 SQL语句，如表20-1所示。

表20-1 SQL语句的并行粒度

类 型	SQL语句	备 注
ROWID	SELECT	在表中或分区中
分区	UPDATE、DELETE	在分区级
从属	INSERT、SELECT	在表级

并行查询（ PQ ）结构在Oracle 8.x与Oracle 7.x中是一样的。但是，现在用PQ除了可以并行执行SELECT之外，还能够并行执行一些 INSERT、UPDATE和DELETE操作。这个新的DML功能（记住：Oracle用DML指代INSERT、UPDATE和DELETE）被叫做并行数据控制语言（ PDML ）。PQ可以在一个分区内并行，而PDML只能在分区或表级并行。

ROWID级的并行被并行查询用于处理 SELECT语句。表是否被分区了没有关系，因为并行性的粒度级别比分区（ ROWID ）低。这就意味着多个PQ从属可以从一个单独的表或分区中读。但是，没有PQ从属可以跨过多个分区。

分区或从属过程级的并行被PDML用于处理UPDATE、DELETE和INSERT...SELECT操作。首先，考虑UPDATE和DELETE操作。当从属的表被分区时，并行性就是分区级，用每个分区只允许一个从属分区的方法进行 UPDATE和DELETE操作。当从属的表不被分区时， UPDATE和DELETE就不能并行。

不管从属表是否被分区， INSERT...SELECT都将由INSERT的从属过程通过按卷分配新行（没有ROWID）并行。多个从属以直接载入方式插入到目前高水位标志之上，这就象你在本章前面20.1节中直接载入性能的讨论中看到的一样。每个表或分区中都可能多个从属。

你可以在会话级使用 PDML，而PQ一般是在实例级使用的，尽管在会话级也是可控的。

SQL代码中的并行提示按对 PQ所做的那样，对 PDML用同样的方法做。同时，优先权规则仍然应用于确定并行度。除了要考虑系统的 CPU数和要被存取的分度数（如果被分区了）外，PDML跟PQ一样，只需选择下面项目中的一个来确定执行从属 SQL语句的并行度：

- 1) 提示指定的度。
- 2) 表缺省的度。
- 3) 一个连结中有最大度（提示或缺省）的表。

清单 20-2给你一个巧妙的有提示的 INSERT...SELECT例子。假定表 T1和T2缺省度都为 8。

Oracle执行这个语句时选择的并行度会是什么呢？

清单 20-2 在SELECT语句中使用提示

```
SQL> INSERT /*+ parallel(T1,4) */  
INTO T1  
SELECT /*+ parallel (T2,6) */ column1, column2, column3  
FROM T2;
```

总并行度是4，因为首先考虑 INSERT提示，并且不论是表缺省度（8）还是SELECT提示（6）都被忽略了。与它有关的具有最大度的表（这里，T1和T2都是8）不适用，因为没有涉及连结。

最后，当正常处理 PQ时，如何监控 PDML活动以帮助你调整并行度？PDML\_ENABLED列已经被加入到V\$SESSION视图中，并且将详细说明是 YES还是NO，与对那个会话是否使用 PDML相对应。DML Parallelized行已被加入到V\$PQ\_SESSTAT视图中，使你能看到有多少用于那个会话的DML操作已经被累积并行放置。当然，V\$PQ\_SYSSTAT现在包含一个称为DML Initiated的新行，它表示所有累积的跨所有会话启动的 DML实例。注意，在V\$PQ\_SYSSTAT中，所有其他的“老”行针对 PQ和PDML活动的和。例如，“服务器高水位”表示曾经被 PQ或者PDML启动的最多从属进程数。由于本节的讨论主要集中在 PDML上，请参考第38章以对 PQ进行更深入地学习。

为什么PDML能够增强 Oracle 8.x的性能？n路数据并行能够为将速度提高几倍打下基础，进程并行（PQ和PDML）实际上也一样。因此，伴随着 RAID、条或者特别是 Oracle 8.x的分区，PQ和PDML可以使速度提高n倍。

## 20.4 大池与非常大的内存

同在8.0.2中一样，大池是 SGA中一个可选的新内存区。它可以用于通过多线程服务器（MTS）支持UGA（会话特定的）内存再分配，甚至XA支持输入/输出从属的内存分配，例如那些被DBWR和LGWR（db\_writer\_processes、dbwr\_io\_slaves和lgwr\_io\_slaves）使用的内存，以及由备份和恢复操作引起的内存分配。

通过建立一个与共享池分开的大池，MTS不必在共享池中增长，因此要缩小库（SQL）高速缓存的可用空间。这是因为共享池不是动态再分配的，并且MTS和SQL高速缓存（与数据字典高速缓存一起）都竞争相同的内存资源。然而，在静态分配的共享池中，MTS和SQL高速缓存是被动态地分配的内存结构。为帮助减轻这种状况，可以分配一个主要由MTS使用的大池，并让共享池主要处理SQL高速缓存。这就打破了共享池的瓶颈，并防止了它内部的增大-缩小循环的争夺。

同备份和恢复进程一样，输入/输出从属能够分配成百KB的内存。在繁重的竞争下，共享

池可能跟不上这些进程，它们可能因此而缺乏内存。基本上，它们仍然处于长等待状态，没有进展或者每次进展非常少。大池也可以通过从共享池中除去一些竞争负担来缓解这种状况。因为大池没有LRU列表，正如你希望在高速缓存状况中看到的，它们的额外开销很少，仅仅充当一个大的轻量缓冲区。

相关的init.ora参数列在表20-2中。

表20-2 init.ora 参数

参 数	描 述
large_pool_size	大池的大小
large_pool_min_alloc	大池的最小分配
shared_pool_size	共享池的大小
shared_pool_reserved_size	大对象的保留列表的大小
shared_pool_reserved_min_alloc	保留列表的最小分配

#### Oracle 8.x新增加的参数

记住，把共享池作为由 Oracle 分配给 SGA 的所有内存的一部分。

尽管非常大的内存（VLM）是 7.3.x 版本的功能，但是它被用户们相对地忽视了并没被充分利用。在 Oracle 8.x 中它会有更多的用处，这就是它包含在本章而不是包含在第 18 章中的原因。

Oracle 非常大的内存功能由于 Windows NT 操作系统内存寻址的性能而被过大地想象了。尽管 Windows NT 可以运行在 64 位的机器上（例如 Alpha 服务器），但它仍然是一个 32 位的操作系统，受到 32 位地址的限制。例如，可以想象一个 32 位的机器有  $2^{32}$  次幂（即 4GB）的内存要被寻址。然而，Windows NT 保留了 2GB 给它自己（“系统可寻址能力”），2GB 给用户线程（“用户可寻址能力”）。请注意，企业版的 NT 只需要保留 1GB 内存，因此有效地将最大用户寻址能力增加了 1GB 内存（达到 3GB）。

然而，不是所有的 NT 都有 4GB 或者更多的物理内存。那么，Oracle 如何在一个只有 2GB 物理内存（或者更少）的 NT 机器上运行呢？实际上，NT 从最高的内存地址区向下增长，允许诸如 Oracle 的用户线程向上增长。问题变成了 NT 和 Oracle 是否能全部得到它们动态需要的内存而不受到对方的牵制。实际上，这种系统中通常发生的事情是 NT 或者 Oracle 开始发生同普通的内存分页形成对比的交换与系统失效，在这些情况下，通常由于内存（缺乏）资源错误引起致命的系统崩溃。

VLM 是如何工作的？这个问题的另一种问法是用用户过程如何存取已存在的、不直接由操作系统寻址的物理内存？答案是：间接地。Lotus、Intel 和 Microsoft 许多年以前建立了一个规范，这个规范是为了突破所谓的 DOS 物理内存访问的“640KB 或者 1MB 的屏障”。他们成功地建立了现在熟知的扩充内存的 LIM 规范。他们所做的是使用 DOS 的某些高端内存（在第一个 640KB 上方的 384KB），它们是可直接寻址的，存储超过 1MB 的内存地址。本质上，把可寻址的内存作为间接地址，或者换句话说，作为内存分页表。LIM 规范可用于诸如 Lotus 1-2-3 电子表格的应用以便存取几兆字节的内存。尽管随着扩展内存以及后来的 Microsoft Windows 内存地址的出现，此技术现在已经陈旧了，但这在当时是非常创新的。

LIM 如何与 VLM 相比？Oracle VLM 是基于间接地扩展内存存取的完全相同的概念。尽管实际的实现不同，但是概念在功能上是相同的。内存给予者进程是虚拟用户线程，Oracle VBLM 使用该进程在 4GB 的 NT 机器上扩展对超过 2GB 的内存的寻址能力。有了 Oracle VLM 内

存，可寻址能力的极限是安装的物理内存减去 NT所需的 1GB或2GB内存。Oracle的检测已经显示性能提高了一两个数量级；换句话说，速度提高 10倍到100倍（这是非常好的，如果没有额外开销，理论上的最大收益大概有三个数量级的提高）。

如何让 VLM工作？可以使用 init.ora文件中的一个参数：use\_indirect\_data\_buffers。把该参数设置为TRUE，Oracle不仅能在通常的SGA分配之外使用内存，也可以在超过由 NT的4GB的可寻址区或者任何其他的用户线程中使用内存。换句话说，这使 Oracle VLM成为可能。另外init.ora参数可能需要调整。请参考 Oracle 8服务器调整指南或者 Oracle 8调整指南，其中包含更具体的实现细节。下面是一些有用的 Oracle VLM信息的小结：

允许间接存取超过NT管理范围的物理内存。

与Oracle的企业版捆绑在一起。

在NT 4.0、NT 4.0 企业版和NT 5.0上透明地运行。

应用程序代码不需要改变。

已经显示了2~3个数量级的性能增加。

## 20.5 网络增强

和网络有关的主要增强功能能用积极的方法影响性能。被称为外部过程的结构现在是可用的。外部过程是 PL/SQL或者SQL的外部代码段。它是有用的，因为它总是完成一些非常复杂或者独特的处理，例如某些科学的程序设计。换句话说，许多以前传下来的执行得非常好的例行程序，除了Oracle集成外，不必重写它们。重新使用已有的工作代码是一个比重新编写代码更快的开发方法。

重新使用外部的例行程序而不是重写它们，这不仅是最快的开发方法，而且实际上可能比你用PL/SQL编写的任何程序还要快。例如，它们可以用一些精炼的快速 3GL语言（比如C）编写。此外，C也可以“包含”汇编程序代码，假定该代码被正确编写的话，那么它几乎同任何应用程序代码一样快。

在PL/SQL中调用外部进程的步骤如下所示：

- 1) PL/SQL使用别名库（alias library）调用外部进程。
- 2) PL/SQL把这个需求传送给Net 8监听器。
- 3) 监听器开始extproc专用进程。
- 4) extproc载入外部共享库。
- 5) extproc代表PL/SQL执行外部进程。

别名库是一个内部的Oracle库，它映射到一个外部（操作系统）共享库。要建立（联系）别名库，只需像下面这样使用扩展的DDL：

```
SQL> create or replace library libm  
2 is '/usr/share/libm.so';
```

用户需要什么特权来建立或者使用这些别名库呢？要建立一个别名库，管理员或者用户需要CREATE LIBRARY或者CREATE ANY LIBRARY。外部过程用户需要别名库自身的EXECUTE特权。CREATE LIBRARY语句的准确度只能在运行时检查，而不能在建立时检查。因此，需要确保共享库实际上同指定的绝对文件名（包括路径）正好相同。你可以在USER\_LIBRARIES或者ALL\_LIBRARIES中检查库的状态。

由监听器启动的 extproc 进程只能在会话限制一个外部进程调出的期间内持续。每个会话使用一个不同的 extproc 进程执行外部进程请求。你必须配置监听器以便与实例进行联系。建议使用独立的监听器，以避免由于存储器泄漏而造成的 SGA 损毁不致于波及外部程序或者它们同 PL/SQL 的交互。TNSNAMES.ORA 中的 extproc 项同任何其他项一样。但是 extproc 别名的名称必须是 extproc\_connection\_data，这在安装中已经预定义了。简单地将它的注释标志去掉，如果需要的话就修改它，然后使用。下面是一个 extproc 项的例子：

```
extproc_connection_data =
(
  DESCRIPTION =
  (ADDRESS = (PROTOCOL = ipc) (KEY = mysid) )
  (CONNECT_DATA = (SID = mysid) )
)
```

20.6 数据库限制

数据库限制周期性地增加，无论何时，只要一个新的 RDBMS 软件出现，它们就趋向于增加。历史上，许多 RDBMS 厂家都有它们自己的缺点，不同的 RDBMS 有不同的弱点。然而，Oracle 8.x 的一些上限不太可能被大部分应用程序（甚至是大部分 VLDB）很快达到。

表20-3给出了新的数据类型的最大值。

表20-3 数据类型的最大值

数据类型	最大字节
CHAR ( n )	n 2000
VARCHAR(n)	n 4000
VARCHAR2(n)	n 4000
NCHAR	n 4000

表20-4包括了其他数据库对象的最大值。

表20-4 数据库对象的最大值

对 象	最 大 值
数据库容量	512PB ( 千兆兆字节 )
每个数据库的表空间	~ 2GB
每个表空间的数据文件	1022
每个表的列	1000
每个表索引列	32

20.7 小结

现在复习一下你已经学过的概念以及它们与性能的关系。你已经学习了 Oracle 8.x 的增强概念和结构，在表 20-5 中列出了与性能有关的细节及说明。

表20-5 Oracle增强

增 强	说 明	关 联 性
分区	数据并行	像RAID那样提高速度
IOT	反向表	逻辑输入 / 输出减半
PDML	进程并行	速度按因子提高

(续)

增 强	说 明	关 联 性
大池	卸载共享库	更大的可伸缩性
VLM	间接内存存取	在NT上存取超过4GB的内存
extproc	外部进程	调用C
上限	数据类型和对象	主要对于VLDB

所有这些增强都是对 Oracle 7.x 的显著改进。这样，Oracle 使用比以前更高的本地并行（数据和代码）能力向 VLDB 和数据仓库系统存储和检索数据。