第11章 管理Oracle Financials和其他软件包及实用程序

当安装一个与Oracle数据库一起使用的软件包时，通常要修改应用的前端部分以真实反映业务过程。就像业务需求驱使对应用前端进行修改一样，应用后端也需要进行修改。若要获得软件包的最佳应用效果，就要修改软件包使用的缺省数据库结构。本章主要介绍软件包的管理准则。为Oracle财务软件包、Oracle Designer(Oracle的CASE工具)和Oracle实用程序(例如ConText、SQL*Loader)提供特殊准则。

11.1 软件包的通用管理准则

管理软件包需要了解软件包的技术细节及其相应的业务进程。在大多数情况下，封装式软件的缺省安装无法无缝地满足业务进程的需要。因此，软件包安装时可能存在空间管理或软件性能方面的问题。而且随着软件包使用的增长，问题也会相应地不断增多。

如果从不能正确支持软件包的数据库环境着手管理，产品的使用负担很快就会成为问题。产品问题可能包括事务无法完成，查询异常缓慢或者数据段没有足够的空间。所有这些问题都将导致产品服务中断或者严重的性能问题。另一方面，如果从灵活和可支持的应用环境着手管理，就可以提高对产品出现问题的应变能力。软件掌握得越好，越能在软件包产品发布前更好地调整环境。

要想高效地管理封装式软件包，需要提供一个尽可能稳定的数据库环境。这一节给出调整数据库结构、管理事务、监控和其他系统级考虑的准则。在这一节后将介绍Oracle Financials、Oracle Designer、ConText及其他软件包的一些特殊应用准则。

11.1.1 定制数据库结构

如果安装的是非Oracle公司编写的第三方软件，那么就不能指望获益于Oracle的特有特性，如序列、存储过程、触发器及Oracle优化程序等。还应怀疑该软件包的数据分布、空间分配及索引模式。许多软件销售商都支持多个数据库平台，要求他们尽可能保持其产品通用。

如果软件包是Oracle公司开发的(如Oracle Financials、Oracle Designer)，那么就应该把调整数据库结构的注意力集中在支持你的业务进程特性上。例如，如果你的业务支持大量反复的预算处理，就要调整存储预算信息的Oracle Financials表，这完全不同于涉及很少预算过程的业务。

应进行调整的数据库结构包括表空间、索引及回滚段，这些在下列各小节中讨论。

1.表空间

一般来说，非Oracle公司开发的大多数软件包都尽可能少地使用表空间。例如，一个软件包可能希望在数据库中创建以下表空间(除SYSTEM、TEMP和ROLLBACK外)：

APP_TABLES	用于所有的应用程序表
APP_INDEXES	用于所有的应用程序索引

当安装软件包时，所有的表都建在APP_TABLES表空间中，所有的索引都建在

APP_INDEXES表空间中(尽管有些软件包在缺省情况下并不将它们的表和索引存储在不同的表空间中)。虽然这样做可以使表和索引的管理变得简单，但这种设计没有考虑到应用程序中所涉及的不同种类的表。例如，有些表可能小而静态，而另外一些表却大而频繁变动。

在第3章中，有对典型OFA表空间结构进行扩展的推荐形式。DATA_2表空间被标识为用于低使用数据段，INDEXES_2是这些表的索引。下面是修改过的应用程序表空间：

APPS_TABLES	用于高使用(high-use)应用程序表
APPS_STATIC_TABLES	用于静态应用程序表
APPS_INDEXES	用于高使用应用程序索引
APPS_STATIC_INDEXES	用于静态应用程序索引

通过把静态表和索引从主表空间中移走，就可以简化全部表空间的空间管理问题。在修改后的表空间结构中，APPS_STATIC_TABLES和APPS_STATIC_INDEXES表空间需要很少的直接管理，从而能将管理与调整的注意力集中在主数据表空间上。用这种方法，既通过表空间分离了表，也将动态和静态表分离到不同的数据文件，简化了I/O分布的管理。

可能要进一步划分频繁使用的表。例如，如果软件包使用了许多在数据界面处理过程中用到的“临时”表，就可以把这些表同其他高使用表分开。也可以增加表空间来支持特殊临时段的需求和软件包的事务尺寸。

> **注意** 如果要对软件包的表和索引的表空间分配进行修改，可能也要修改供应商提供的供今后升级用的脚本文件，因为这些脚本文件可能假定对象处于其原始表空间中。另外，如果应用程序的模式把APPS_TABLES作为其缺省表空间，执行create table或create index命令的任何升级脚本文件都将把对象创建在APPS_TABLES表空间中；除非该命令显式地命名一个表空间。这可能使已经从APPS_TABLES表空间移走的对象在升级过程中又移回到APPS_TABLES表空间中。

2. 索引

你不可能预想到用户对数据库提出的所有问题；同样一个非本公司的开发者也不可能想到你的用户有什么需求。因此，任何用户可以进行特定查询的软件包都很可能要遇到一些查询的性能问题。尽管这种查询数量很少，但它们消耗大量的系统资源，给数据库的其他用户造成了消极影响。

造成这种资源密集型查询的最通常原因是缺少合适的索引。只有了解了用户在业务处理中如何使用软件包，才能够完全了解所需要的数据访问路径。就是这样也不能完全定义用户所需的所有索引。如果对业务处理过程理解得很透彻，可能会定义很多关键索引，但是由于正在使用封装软件，所以不可能看到软件执行的所有SQL语句，也就不能判断表应如何进行索引；直到用户发出抱怨，才认识到这个问题。

随着时间的推移，索引可能失效。因为从表中删除记录时，也从表的索引中删除相应的条目。表中释放的空间可以重用，但索引中释放的空间不能再用。因此，频繁从中删除行的表应定期重建其索引，以避免在索引中造成空间碎片。这种问题最常出现在装载过程中临时使用的表里。例如，如果软件包向一个临时表插入收到的记录，对记录进行处理，接着从临时表删除这些记录，那么临时表的索引将不断增加；即使表中没有行也是如此。为了避免出现这种问题，当表空的时候应定期截断表，truncate命令是不可回滚的，它删除表中所有行，也删除索引碎片并把表压缩成一个盘区(在缺省情况下，可以用truncate命令的reuse storage选

项保持以前分配的盘区)。

如果表增长得太快，销售商就要提供一个数据归档机制，还要经常重建索引。数据归档进程通常要删除比设置时间早的数据。由于定期进行大删除，所以索引成为碎片且效率低。在每一次归档操作后，应该重建索引。

3. 回滚段

对于数据输入及特制报表，大多数软件包都包括一个联机用户组件；对于大型数据处理操作，则包括一个批处理组件。联机用户需要足够数量的回滚段来支持其生成的大量并发事务。批处理需要足够大的回滚段来支持最大的批处理事务。在第 7章中，已经介绍了如何确定回滚段的数量和大小。如果在软件包中有一个事务明显地比其他事务大，就要为该事务专门设计一个回滚段。这个专用回滚段应存储在它自己的表空间中 (见第3章)，在启动该事务之前，应发出set transaction use rollback segment命令，强制这个事务使用这个回滚段。在每次事务提交之后，都要重复这一个命令。

如果不能变更事务代码使其包含 set transaction use rollback segment命令，就应安排这个事务在没有联机用户在数据库中激活时进行。在开始该事务之前，使数据库中的其他回滚段为脱机；只使SYSTEM回滚段和特定回滚段为联机。用 alter rollback *segment_name* offline命令使回滚段脱机(见第7章关于激活和取消回滚段的 Oracle Enterprise Manager方法)。当事务启动时，它将被迫使用专用回滚段。当事务结束时，再使其他回滚段恢复为联机。

4. 临时段

批处理事务在处理过程中需要大型临时段。例如，使用 SQL*Loader Direct Path选项的大型数据装载，在装载数据时需要临时段来存储表索引的数据。如果在索引创建过程中频繁使用Parallel Query Option(PQO)，就需支持大量的临时段；因为每个并行查询服务器进程都可能获得一个临时段。如果在批处理过程中使用了很多并发并行查询服务器进程，临时段空间的需求量很可能超出通常供软件包用户使用的资源。

根据应用程序的情况，可能要将用户 —— 或者应用程序模式的拥有者 —— 划分成不同的临时表空间组。例如，可以创建四个独立的临时表空间，并且把联机用户分成四组。如果用户的临时表空间按这种方法划分，在需要临时段的处理过程中，用户之间的影响就较小。如果一个事务或查询对临时段的需求量远远大于软件包中其他任何事务或查询的需求量，就应该赋予该事务或查询的用户一个缺省临时表空间，该表空间专为支持这类事务而设计。

5. 表和索引的大小

对于小且不变的表，软件包提供的表和索引的缺省大小一般是精确的，尤其是 Oracle公司开发的软件包尤其精确。不过，仍可以按照空间估算来检验缺省的存储参数 (见第5章)。对于大且很活跃的表，应该想到表的缺省存储参数不正确。没有办法使应用程序开发者确切地知道你将如何使用软件包，因此需要调整表和索引大小以支持预计的数据容量。

不能依赖于通过Import/Export整理表碎片的方法来执行所需要的表大小调节工作。例如，假设软件包最开始使用时表的大小不合适。它的 initial盘区大小为100KB，next盘区为100KB，pctincrease值为0。表的每个盘区都是 100KB大小。在使用的第一个月，表达到了 10个盘区。显然，随着更多的用户加入系统，表的空间增长率也增大了。如果用 Import/Export把表的盘区压缩为一个盘区，那么新的存储参数就变为： initial盘区为1MB，next盘区为100KB。尽管Import/Export过程给出一个较好的 initial盘区大小，但对 next盘区大小没有解决任何问题。需

要手工调节表以获得一个更合适的next盘区。

许多销售商都提供有可调整大小的电子表格，可以用它来估计对象的空间使用情况。数据库管理员必须与应用程序组一道工作，以确保在应用程序实现前进行调整练习。根据所涉及的计算，销售商的尺寸计算可能非常费时。

6. 管理pctused设置值

每个Oracle表都有一个pctused存储参数；该参数的缺省值为40。在大多数应用程序中，将pctused设置为40并不会引起问题。不过，移植到Oracle的软件包应用程序可能会造成与pctused设置相关的空间使用问题。

pctused参数设置必须使用的数据库块的最小百分比。如果数据库块使用的空间百分比低于pctused值，则该块就添加到表中自由块的清单上，并且一个新记录可能写入这个块中。如果空间使用率大于pctused，该块将不适合接受任何新记录。

移植到Oracle上的某些应用程序并不利用Oracle的锁定机制。为避免遇到锁定问题，这些应用程序可在插入之后执行删除操作。如果一个应用程序执行许多删除和插入操作，表中的这些块将不能有效地使用其空间，除非你改变了pctused参数。

假设这样一种情况：在pctused参数的缺省值设置为40的表中，一个应用程序创建许多行。然后将大量的行更新，应用程序通过删除旧行和插入新行来执行更新操作。当旧行被删除时，Oracle就计算块的空间使用比率。如果使用的空间百分比大于40，就不能接受新行。执行插入操作时，新行写入另一个块，旧行所使用的空间保持空状态。当重复出现这种情况时，就会发现你的表的块只有40%到50%在使用着，这使空间要求倍增。

为避免出现这种问题，对于任何频繁执行删除/插入事务的表，都要把pctused参数设置为一个高值(70或更高)。必须与包应用程序销售商一起确定这在你的环境中是否是一个问题。

7. 管理SQL共享池

注意 只是在本节中，术语“包”才表示通过create package命令创建的存储PL/SQL对象。

PL/SQL对象在使用时存储在System Global Area(SQA)内SQL共享池的库缓存中，如果它已被一个用户调入内存，其他用户在执行这个包时就会提高性能。把一个包“驻留(pin)”在内存中减少了用户执行过程中的响应时间。

为了提高把大型PL/SQL对象驻留在库缓存中的能力，数据库一打开就要把它们装入SGA中。启动后立即驻留包，可以增加在内存中相邻部分存储PL/SQL对象的可能性。可以用DBMS_SHARED_POOL包把PL/SQL对象驻留在SGA中。若要使用DBMS_SHARED_POOL，首先要引用想驻留在内存的对象。若要把PL/SQL对象装入内存，可以引用包中定义的一个哑过程或重新编译包。

一旦对象被引用，就可以执行DBMS_SHARED_POOL.KEEP过程来驻留对象。如下面清单中所示，DBMS_SHARED_POOL的KEEP过程，以其对象名和对象类型(对于包缺省为“P”)作为输入参数。

```
alter procedure APPOWNER.ADD_CLIENT compile;
execute DBMS_SHARED_POOL.KEEP('APPOWNER.ADD_CLIENT','P');
```

注意 DBMS_SHARED_POOL包并非由catalog.sql文件创建。要创建这个包，可能要运行dbmspool.sql脚本文件(当登录为SYS时)。这个脚本文件位于Oracle软件主目录下

的/rdbms/admin子目录中。

上面的例子表示，将一个对象驻留在内存中要经历两步操作：首先是引用对象(通过编译)，其次是给它做标记。启动之后立即把最常用的对象驻留在内存可以提高在SGA中为它们获取连续空间的机会。

每个数据库都有两组包要驻留：一组是每个数据库都要用的内核，另一组是应用程序特定的。驻留的软件包内核组一般包括SYS拥有的包STANDARD、DBMS_SQL、DBMS_UTILITY和DIUTIL。若要确定为实例驻留哪个软件包，请查询DBA_OBJECT_SIZE。首先应驻留最大的软件包。若要确定要驻留的软件包及其驻留顺序，可使用下面例子所示的脚本文件。这个脚本文件用DBA_OBJECT_SIZE视图列出驻留对象的顺序。

```
select Owner,
       Name,
       Type,
 Source_Size+Code_Size+Parsed_Size+Error_Size  Total_Bytes
  from DBA_OBJECT_SIZE
 where Type in ('PACKAGE BODY','PROCEDURE')
 order by 4 desc;
```

除了驻留包外，还有两个可用于改善SQL共享区管理的附加选项。这两个选项通过init.ora参数SHARED_POOL_RESERVED_SIZE和LARGE_POOL_SIZE控制。

SHARED_POOL_RESERVED_SIZE设置SQL共享区保留部分的大小(以字节为单位)，这个保留部分专用于满足对大置连续内存的请求。在缺省情况下，SHARED_POOL_RESERVED_SIZE将设置为SQL共享区的5%。如果有效地使用SHARED_POOL_RESERVED_SIZE，就不需要驻留许多包。可以通过SHARED_POOL_RESERVED_MIN_ALLOC为保留区域的条目建立最小的尺寸；但Oracle8i不支持这个参数。

LARGE_POOL_SIZE参数并不提供一个驻留大型对象的区域，而是将SQL共享区的一部分用于并行查询操作、多线程服务器内存和备份操作上。在Oracle8.0中，可通过LARGE_POOL_MIN_ALLOC参数为大池的条目设置最小尺寸，但是这个参数在Oracle8i中不被支持。只要使用Parallel Query Option、DBWR I/O从进程或多线程服务器，Oracle就自动为LARGE_POOL_SIZE设置一个值。

11.1.2 安全与数据访问控制

除了要调整数据库结构外，还应检查实现软件包安全性的方法。开发者对安全性所做的设想在环境中并不一定合适。因为开发者开发软件包时对你的数据库和你的公司的安全策略没有访问权。因此可能会发现，应用程序要求所有用户都要有DBA权限，或用户共享账户，或者在正常使用软件包时能容易看到用户口令。应该同软件包提供者一起决定软件包是否能按照公司的安全准则来实现。

在数据库中，应决定应用程序是否使用了大量不同的角色，哪些角色是用户的缺省角色。如果在软件包中有许多角色，或者数据库支持多种应用程序，就要限制每个用户所允许的角色数(通过init.ora文件中的MAX_ENABLED_ROLES参数设置)，尽可能减少授予角色的系统权限数量(例如ALTER ANY TABLE)。在缺省情况下，软件包将经常使用比需要更高的访问级别，比如，当用户只需要ALTER USER权限以便可以修改用户的口令时，软件包需要DBA角色的访问权。有关用户配置命令的情况，请参见第9章。

有些应用程序在数据库中不提供或提供很少的安全性。如果能与数据库连接，就可以访问大多数数据。为了提供常规安全界面，销售商可能必须创建其自己的用户和口令表。根据这个表对到应用程序的连接进行验证。尽管这种模式可以给销售商一个灵活、通用的安全解决方案，但它可能是数据库中安全脆弱性的一个根源。为避免对数据库恶意或无意的损害，要对用户名和口令严加保护。还应使用 PRODUCT_USER_PROFILE表来限制SQL*Plus中的访问，这些情况已在第9章介绍过。

11.1.3 事务管理

软件包的事务管理需要理解应用程序的批处理和联机部分，并且要把这些处理分离到尽可能最大的盘区。应用程序中与事务相关的大多数问题，都是由于同时安排长时间进行的批处理和联机事务引起的。有关不同类型事务对回滚段的需求，请参见第 7章。如本章前面所述，需要足够数量的回滚段来支持联机用户，同时需要特殊的回滚段来处理大型批处理事务的特别需求。

除了设置大小合适的回滚段外，还要判断软件包如何实现锁定。如果软件包从其他RDBMS移植到Oracle，该软件包就不能充分利用 Oracle中锁定的粒度(granularity)。需要发现什么时间记录被锁定 —— 当用户第一次查询这些记录时它们是否锁定 for update(更新)，如果锁定，那么当另一用户试图查询相同的记录集时，将被迫等待第一个用户查询结束后释放锁。应知道软件包如何处理数据并发问题，当它是一个客户机 /服务器模式的应用程序时尤其如此。例如，如果两个用户分别对同一行进行查询和更新，则哪一个更新被接受呢？不同的软件包有不同的答案，必须知道软件包的实现方法。一旦知道所使用的锁定模式，就能帮助应用程序组用一种最佳的技术与资源使用方式来实现软件包。

管理三层或基于网络的应用程序时，就要反复考虑锁定问题，如何访问数据？何时由应用程序锁定记录？如果该结构的中间层关闭，对用户会话和锁会发生什么情况？

在基于服务器的应用程序中，可以轻松地使服务器上的一个用户与数据库中的活动相联系。当把越来越多的层添加到应用程序结构上时，识别问题根源的过程就非常困难。在一个多层环境中，依赖于每个层正确地管理用户权力及其获得的锁信息。作为把应用程序移植到产品数据库之前执行测试的一部分，应测试数据库如何处理缺乏的技术结构。

11.1.4 文件定位

在开发和测试环境中，应监控软件包以判断哪些文件是频繁使用的文件 (见第4章)。在产品环境中，要移动这些文件以便它们不会彼此争用服务器上的 I/O资源。应清楚磁盘布局，以便可以跨磁盘和跨磁盘控制器分布数据库。对于提交的应用程序产品，要监控软件包的使用情况，以判断实际的使用情况是否符合预想的使用情况，并进行相应的文件分布调整。有关物理设计的I/O分布情况，请参见第4章。

11.1.5 监控

除了软件包需要的特殊监控外，监控软件包还涉及第 6章描述的基本环境监控。例如，软件包的批处理事务在大小上有相当大的差别，这就需要仔细监控它们的时间安排和资源使用情况。应使监控过程尽可能自动化，减少在观察数据库活动上所花费的时间。达到这一目标

的最好办法是将非标准活动(如批量装载或用户数量陡增)隔离出来。在非标准活动阶段，严密监控数据库。在其余时间，数据库操作应进入正常监控范围。如果数据库超出正常监控范围——如一天中的总命中率比阈值还低，你就应该被告知并检查问题的原因。

11.1.6 版本考虑

在支持软件包现有版本的前提下，如何开发软件包新版本呢？软件包的第二个拷贝需要第二个实例，还是软件包的两个拷贝存在于同一个实例中？为了正确规划支持软件包新版本的实现环境，需要对这些问题做出回答。

例如，假设软件的对象通过公共同义词来访问，当用户查询 EMPLOYEE时，实际上用户正在对由公共同义词EMPLOYEE定义的对象进行访问。在这种环境下，可以在同一个实例中创建第二组应用程序表。例如，主软件包对象拥有者 (称为APP_SYS)将拥有第一组表。公共同义词都将指向APP_SYS对象。第二组表将由第二个用户 APP_SYS_TWO创建。所有应对第二组表进行访问的用户都应有指向 APP_SYS_TWO表的私有同义词。当数据库分析对象名时，私有同义词将覆盖公用同义词。

如果软件包要求它的表存储在特定用户名下，就不能在同一数据库中创建软件包的另一个拷贝，而是要给软件包的第二个拷贝创建一个独立的实例。因此，如果有一个开发数据库、一个测试数据库和一个产品数据库，当软件包升级到一个新版本时，就需要第二个开发数据库、第二个测试数据库和第二个产品数据库。在升级时不能使用第一个开发数据库，因为需要有一个数据库支持对当前产品数据库的修改开发。一旦知道软件包如何处理版本问题，就可以规划支持实例所需要的资源。

除了考虑应用程序版本外，还要了解软件包对不同版本 Oracle RDBMS和工具的依赖性；只可以认证一个应用程序运行在 Oracle RDBMS版本7.2.3.2、7.3.4.3和8.0.5.1下。这种限制可能会影响整个版本的移植策略。例如，可能限制磁盘空间的使用，影响能保持联机状态的Oracle软件版本数量。如果只认证应用程序运行于少量的 Oracle RDBMS版本，就要选择一种支持版本作为全部数据库的基本版本。

如果有其他数据库与应用程序表相互作用，在每次升级软件包的数据库时，都要评估这些依赖性。因此，可能要把多个数据库的升级版本有机地“粘合”起来。调整升级可能减少相关依赖性问题，但将增加预升级测试的复杂性和重要性。

将应用程序从一个版本转移到下一个版本时，要始终确保销售商提供有数据迁移实用程序。在执行产品环境中的迁移之前，总要在一个隔离的测试环境中对数据迁移实用程序进行测试。把应用数据从一个版本迁移到另一个版本与数据相关，很少有准确预测成功与失败的通用测试。如有可能，用整个产品数据集测试迁移。若数据迁移在测试环境中成功之后在产品中失败，一般是由下述两种原因之一引起的：

1) 数据量非常大　在整个数据集上迁移事务，需要大型回滚段或临时段。

2) 修改产品系统　产品系统可能含有专用于对产品应用程序进行修改的数据。销售商的迁移实用 程序可能没有考虑到你的修改。

数据迁移工作不可能提供一种简单方法，重建迁移开始前那样的产品环境。因此，应在迁移前就对数据进行联机备份；也可以用实例将数据隔离起来，以防止造成一种不可恢复情况。

11.1.7 DBA的角色

如果正在执行一个自定义应用程序，开发过程中的每一步都要涉及到数据库管理员（见第5章）。对于开发小组来讲，数据库管理员扮演着一个技术顾问的角色，他协助应用程序的规划、测试及实现。

如果是在购买软件的基础上二次开发应用程序，数据库管理员的作用就更大。除了要和开发小组的人员直接接触外，数据库管理员还要和软件包供应商的技术开发人员直接打交道。由于软件包不是自己的开发人员开发的，所以还要和供应商建立联系才能了解应用程序如何执行锁定功能，或如何管理批处理事务。还要决定软件包支持哪一个版本的 RDBMS，以及它如何影响其他应用的数据库软件升级路径。

与所有数据库应用工具一样，数据库管理员还要和应用程序小组人员一起制定创建和监控数据库的计划。任何特殊考虑（如大型批处理事务或软件包频繁使用的表）都应在创建数据库之前提出。在没有建表之前更改表空间的设计比较简单，一旦表在数据库中建起来，就要通过很多步骤来管理它（在产品环境中一点也没有对服务中断进行安排，所以可以移动表）。

数据库管理员还要弄清楚获得软件包支持所需要的过程。如果通过软件包销售商购买Oracle许可证，就不能直接打电话给 Oracle技术支持。相反，可能要联系软件包销售商，销售商再来联系Oracle技术支持。购买软件包时必须明确定义支持过程，必须清楚销售商对你接受答复和解决问题能力的影响。如果不能直接联系 Oracle技术支持，销售商支持部门的反应性和能力决定你获得对任一 Oracle特定问题的答复所需要的时间。

下面几节将介绍管理Oracle Financials、Oracle Designer、ConText和其他软件包及实用程序的特殊准则。前面几节所讲的准则适用于所有软件包。

11.2 管理Oracle Financials软件包的特殊准则

Oracle记帐软件包 —— Financials软件包 —— 是Oracle公司开发和销售的一种应用程序，用于管理公司帐务。它包括 General Ledger(GL)模块、 Accounts Payable(AP) 模块、Purchase Orders(PO) 模块、Fixed Assets(FA) 模块和其他一些主要财务类别的模块。它是用大量 Oracle开发工具开发的。这一章集中介绍 Oracle Financials版本10和11的相关管理问题，也适用于以前的版本。

从数据库管理员的角度看，即使没有安装所有的模块， Oracle Financials软件也是一个复杂的应用软件。以下几节介绍应注意的数据库关键特征及管理 Oracle Financials软件包的特殊准则。

11.2.1 数据库结构

由于 Oracle Financials软件很大程度上依赖于软件包和触发器这类存储对象，因此SYSTEM的空间至少要达到 150MB来存储代码。代码所需的空间与数据库剩余部分的大小没有关系；即使在Oracle Financials表中没有任何记录，也需要那么多空间。 Oracle Financials数据库通常有一个最多达 300MB的SYSTEM表空间。

Oracle Financials软件的每一个模块都对应一个 Oracle帐号。例如用于General Ledger模块的“ GL ”帐号拥有 GL模块所有的表和索引，而“ FND ”帐号拥有应用软件用来生成前端窗体

的所有表，使用窗体时由应用软件对FND表进行查询。

由于每个模块的对象都是在其自己的帐号下创建，所以为每个帐号创建两个表空间：一个用于表(如GL_TABLES)，另一个用于索引(如GL_INDEXES)。可以对创建应用软件的脚本文件进行编辑，以便把表和索引分别放在不同的表空间。还可以通过把频繁使用的表放入不同的表空间来分离表。例如，如果频繁使用GL的日志条目表，就可以把GL_JE_LINES表移入一个单独的表空间，使其对其他GL表的影响降至最小程度。

即使不使用Financials软件的所有模块，也必须安装所有模块的表。因Oracle Financials软件中的很多视图都是对多个模块的表进行查询，也就是对多个拥有者的表进行查询。未能安装不使用的模块将造成视图无法使用。

在数据库重建过程中，这些视图可能会带来一些问题。如果开发者创建自己视图的话，则更是如此。由于导入结构的顺序可能先于导入其基本表，所以可能无法通过导入操作来创建视图。要解决这一问题，可进行多次导入操作：首先只用被导入的结构(使用rows=N标记)，接着导入数据(使用rows=Y和ignore=Y标记)。第二次导入操作将创建第一次导入操作时遇到错误的任何一个视图。这两次导入操作的命令如下：

```
imp system/manager file=export.dmp rows=N
imp system/manager file=export.dmp full=y buffer=64000
    commit=Y ignore=Y
```

第一个命令导入所有数据库结构，但没有任何行记录(rows=N)。第二次导入结构和数据，且每导入64000字节数据进行一次提交。即使这些对象已经存在，ignore=Y标记也能使第二次导入操作成功。有关导入操作的用法，请参见第10章。

1. 调节表和索引的大小

Financials系统管理员负责调整应用程序对象的大小。所需数据库的大小依赖于同应用程序实现相关的很多因素。有关确定正确尺寸所需的计算方法，请参考Oracle Financials Installtion Guide。

2. 管理回滚段

Oracle Financials软件包括联机和批处理模块。应监控联机的使用来决定所需的回滚段数量，详见第7章。对于批处理模块，要创建特殊回滚段来支持运行的特殊事务。可使用本章前面介绍的方法在大型批处理事务过程中使用特殊的回滚段。

3. 管理临时段

Oracle Financials软件包中的GL用户对空间处理的需求超出数据库中其余全部用户的总和，它需要有一个可用于其处理的大型临时表空间。因为它的需求较特别，所以通过创建一个只用于GL临时段的表空间来把它的临时表空间与其余用户分开。初始表空间的大小在50MB和100MB之间。如下面程序那样创建它，使其有20～50个盘区，以便能在表空间中测量实际使用情况。然后使用下面程序所示的命令把GL用户赋予这个表空间。

```
create tablespace TEMP_GL
datafile '/db01/oracle/FIN/temp_gl.dbf' size 100m
default storage
(initial 5m next 5m pctincrease 0);

alter user GL temporary tablespace TEMP_GL;
```

在上面的程序中，创建表空间的起始值为100MB，它的缺省存储参数将使用5MB盘区。因此，当一个大型GL事务正在运行时，就可以通过查询DBA_SEGMENTS来决定创建一个多

大的临时段。例如：

```
select
   Extents,   /*how many extents does the segment have?*/
   Bytes,     /*how large is the temp segment, in bytes? */
   Blocks     /*how large is it, in Oracle blocks? */
from DBA_SEGMENTS
where Segment_Type = 'TEMPORARY'
  and Tablespace_Name = 'TEMP_GL';
```

上面程序中的查询将提供有关数据库中当前使用的每个 GL用户临时段的信息。查询结果显示每个段的盘区数量及总大小。

如果TEMP_GL表空间未含有任何永久对象 ,就可以将其创建为一个临时表空间。有关临时表空间的情况，请参见第 3章。

4. 驻留软件包

为了便于在SQL共享池中驻留软件包， Oracle Financials软件提供了一个驻留最常用软件包的脚本文件，这个名为 ADXCKPIN.sq1的脚本文件，通常存储在 $AD__TOP环境变量定义的目录下的sql子目录中。可以编辑这个脚本文件，以便从驻留清单中去掉那些认为在实现中通常不会使用的软件包。每次数据库启动之后都应立即执行该脚本文件。

作为驻留软件包的一种替代方式，可以使用init.ora参数 SHARED_POOL_RESERVED_SIZE把SQL共享区的一部分分配给大型对象。为了减少启动数据库所需的时间，许多Financials软件安装都使用SHARED_POOL_RESERVED_SIZE参数来代替驻留。

5.管理中间表

Oracle Financials软件要维护一系列的“中间”表，用于处理通过界面获得的数据。例如，预算装载要用到GL_BUDGET_INTERIM表。因为这些表只用于转换的数据 —— 装载后又卸载的数据 —— 所以表和表的索引不断扩大。如本章前面所述，索引不能重新使用被删除的条目释放的空间。因此，即使中间表的大小保持不变，其索引也不断增大。

要解决中间表所造成的空间消耗问题，必须确保在记录从表中删除后定期截断表。truncate命令将从表中删除任何剩余记录 (这个操作是不可回滚的)，该命令还将截断表索引，释放索引中未使用的空间。在缺省情况下， truncate命令也把表缩减为一个盘区。

应安排对所有的中间表进行定期截断。当 Financials软件工具数量增加且使用复杂时，软件包的界面数量就越来越多，中间表的数量也随之增加。为避免这些表造成的持久空间使用问题，必须主动管理它们。

11.2.2 数据库访问

对Financials应用软件的访问控制在应用程序范围内。用户自己登录到应用程序，经验证后以模块拥有者的身份登录到他们使用的模块。也就是说， GL的所有用户都是以GL身份登录。

使用公用数据库帐号从数据库管理员的任务中取消了对用户帐号的管理功能。只有产品数据库中的帐号才是属于模块拥有者的帐号。 Financials系统管理员负责对所有用户的授权。

从帐号管理的角度看，数据库管理员只涉及到设置模块拥有者帐号 (如GL和PO)和创建从外部数据库访问Financials数据库的帐号；典型的是只能查询访问 Financials数据的有限视图。

DBA还必须通过下面程序所示的命令授予 Financials软件的所有用户，对 V$SESSION和

V$PROCESS表的SELECT访问权限。使用 with grant option子句向APPLSYS用户授予对V$PROCESS和V$SESSION直接进行SELECT访问的权限。

```
svrmgrl
SVRMGR> connect internal as sysdba;
SVRMGR> grant select on V$PROCESS to applsys with grant option;
SVRMGR> grant select on V$SESSION to applsys with grant option;
SVRMGR> grant select on V$PROCESS to public;
SVRMGR> grant select on V$SESSION to public;
```

APPLSYS用户需要显式地对V$PROCESS和V$SESSION视图授权，因为这个用户根据这些系统视图来创建视图。 APPLSYS拥有的视图称为FND_V$PROCESS和FND_V$SESSION。如果不授予上面命令所示的权限，就不能创建 APPLSYS视图。

这些权限在数据库重建过程中可能引起问题。在创建一个新数据库后，在从一个Financials数据库导入数据之前，先创建 APPLSYS用户。在执行一个导入操作之前，将V$PROCESS和V$SESSION权限授予APPLSYS。如果不这样做，APPLSYS就不能在这些系统视图的基础上创建自己的视图。在导出/导入操作过程中不维护对 V$PROCESS和V$SESSION的权限。

11.2.3 并发管理器

Oracle Financials软件有一个称为Concurrent Manager(并发管理器)的内部作业队列管理器。一个Financials数据库可以运行几个 Concurrent Manager(通常少于5个)。这些管理器由Financials SysAdmin创建并作为后台进程运行，以指定的时间间隔被唤醒来检测等待执行的作业。

Concurrent Manager队列中的作业存储在 FND帐号拥有的 FND_ CONCURRENT_ REQUESTS表中。这些作业具有以下状态值。

状　态	含　义
Q	作业在队列中等待处理
R	作业正在运行
C	作业已完成

一旦一个请求完成运行，记录就保留在 FND_CONCURRENT_REQUESTS表中。这些旧记录提供对请求提交的审计跟踪，但这样做以消耗数据库空间为代价。完成的作业记录应按时间表定期归档到平面文件中，然后把它们从表中删除。 Financials系统管理员负责完成这项任务，因为Concurrent Manager此刻应关掉。如果FND_CONCURRENT_REQUESTS表开始变大，数据库管理员应建议Financials系统管理员缩短数据归档操作的时间间隔。

如本章前面所述，一般对记录的删除操作会在索引中产生碎片。因此，对记录归档后要对FND_CONCURRENT_REQUESTS表定期进行截断操作，或者重建索引。

Financials系统管理员可以根据需要调整 Concurrent Manager。可以限制 Concurrent Manager运行某种特殊类型的作业或特殊用户运行的某些作业，也可以根据运行的程序对它进行限制。这些约束使运行时间很长的报告在非高峰时间段运行。通过将非激活但又在使用的回滚段数据减少至最低限度，用这种方法分离作业将大大减少所需要的回滚段空间。

尽管可以配置Concurrent Manager，使其处理以比较低的操作系统优先级运行，但这不是一个最佳选择。较好的解决方案是在适当的时候以同一优先级运行作业。这样对每个人都能

产生更好的性能。数据库管理员应与 Financials系统管理员一道研究系统的使用周期，共同决定如何更有效地安排作业。

11.2.4 演示数据库

Oracle把一个演示数据库同Financials软件一起打包发行。这实际上是一个导出转储文件，可以被导入到一个用户帐号，该帐号应指向自己的表空间。除了相关的视图和索引外，演示数据库还将拥有Financials部分的1000多个表。由于在教学或做演示过程中有可能改动了演示数据库，所以数据库管理员应定期重新导入这些数据。

11.2.5 版本管理

Financials软件是一个可修改的软件包，因此一定要建立一个开发区域。数据对一个公司的财务至关重要，所以通常还要用到测试区域。有关开发过程的更多信息，请参见第 5章。可能需要6个实例，分别对应开发、测试、产品和新版本 Financials软件的第二个开发、测试、转出。

由于Oracle Financials软件使用Oracle工具进行数据输入和报告，因此开发人员可以创建新的报表，并把它们加入应用程序中。这种对财务软件的增值开发能力必须通过第 5章中介绍的准则来管理。

用于访问Financials软件的工具在执行过程中频繁地使用数据库，因此开发工作必须在产品实例之外进行。由于需要一个独立的开发数据库，所以演示数据库在产品实例中已没有必要，应将其装入测试实例中用于系统测试或教学。

11.2.6 文件定位

Financials数据库使用的文件的最佳文件定位取决于具体的应用程序实现。不过，有一些可用于建立基准配置的通用准则。可以使用第 4章中介绍的解决方案和准则，进一步改进文件的定位。

当在有效的服务器硬件上设计文件布局时，应遵循以下准则：

- Oracle软件和财务应用软件应存储在不同的硬盘上，以避免并发 I/O冲突。
- 联机重做日志文件要相当大 —— 每个至少5到10MB。至少要有6个这样的文件才能对应Financials软件的每一个实例。这些文件使用很频繁，因此要把它们存储在任务不重的硬盘上。
- FND帐号要有两个表空间，分别用于表和索引。这些表和索引存储应用程序使用的数据，因此这些表空间通常有很大的 I/O量(见第4章)，应该把它们存储在不同的设备上，以便将并发I/O冲突减少到最低限度。
- 每个模块(如General Ledger)都需要两个表空间：一个用于表，另一个用于索引。这些表空间应存储在不同的设备中以减少并发 I/O冲突。在站点上安装的主模块 (例如GL、FA和AP)也要存储在不同的硬盘上。也就是说，不能把 GL和AP表存储在同一硬盘上，因为向一个模块的表输入会影响另一模块的表，从而触发并发的 I/O冲突。
- 至少要有两个回滚段表空间：一个用于产品数据输入，另一个 (RBS_2)在大型数据装载和月末结帐时使用。它们可以存储在同一设备上，因为它们很少在同一时间使用。

- 需要一个小型TOOLS表空间来支持与Financials软件一同安装的Oracle工具。这个表空间相对于数据库文件有很少的I/O。
- 如前所述，需要两个临时表空间：TEMP用于除GL之外的所有用户，TEMP_GL专供GL帐号使用。它们可以存储在同一设备上。在这两者中，TEMP_GL将经历相当大的I/O，在数据库中，TEMP_GL表空间的相对I/O量取决于应用程序的使用方法。如果同时打开很多周期(period)，那么对记录(post)的处理需求就大量增加，也相应加大了由GL创建的临时表。
- 在开发和测试数据库中，需要用DEMO表空间存储为演示帐号创建的对象。这个表空间在教学时间之外应是静止的。

11.2.7 init.ora参数

可以设置数据库初始化参数(通过init.ora初始化参数文件)来提高Oracle Financials软件的性能。下面几小节列出了可以设置的主要参数。

1. SGA

System Global Area(SGA)是该实例可以使用的内存区域。从数据库中读出的数据存储在SGA中加快了其他用户的检索速度。数据库的结构信息和事务返回的数据存储在SGA中。另外一个内存区域(称为SQL共享池)，存储对数据库运行过的语句的语法分析版本。

对于Financials软件，在服务器的可用内存中应给SGA留出得尽可能大一些的空间。Oracle Financials Installtion Guide给出了对最低应使用的SGA尺寸的计算量。一般来讲，要使效率更高，Financials产品SGA中的数据缓存区至少要80MB。为了支持驻留软件包，SQL共享池也至少为80MB。通过init.ora参数SHARED_POOL_SIZE设置SQL共享池的大小(以字节为单位)，数据块缓冲区的大小以数据库块为单位由DB_BLOCK_BUFFERS参数设置。

2. 打开的游标

init.ora参数OPEN_CURSORS对每个用户进程同时可拥有的打开的游标(环境区域)数量进行限制。这个参数的最大值依操作系统的不同而不同。设置该参数值为其最大值，尽管文档中这个值可能为255，但在大多数操作系统上可以超过这个值。

3. 修改优化程序

在Oracle7版本之前，通过修改内部Oracle优化程序来改变优化程序处理嵌套子查询的方法。这种改变的副作用将会降低由以前的优化程序创建的Financials报表的执行效率。

要弥补这种变化，必须使用一个特定的init.ora参数。该参数的第一个字符为下划线(_)，具体格式如下：

```
_optimizer_undo_changes     = TRUE
```

启动后该参数被检测时，查询过程中对处理子查询所做的改动将被优化程序忽略。

4. 数据块大小

要想最大程度地提高Financials软件的运行效率，就要增大所使用的Oracle块。每块不是2048字节，而要把它加大到4096字节或8192字节。增加数据库的块尺寸，降低了每一块额外消耗的百分比。因而每个I/O读入更多的数据。由于每个I/O读入更多的数据，一次查询也就只需要很少的几次I/O，因而提高了效率。这个参数的init.ora格式设置如下：

```
db_block_size     = 4096
```

数据库建起来后就不能改变块大小。要想改变它，就只能彻底重建数据库及其所有数据文件(通过当前数据的导出/导入)。

11.2.8 最活跃的表和索引

哪些表和索引最可能扩展取决于安装的模块以及使用它们的方法。一般情况下，表及其索引的突增很快会引起碎片问题。由于 Financials软件高度索引化，因此索引首先就要遇到问题。应受监控的表如下所示。它们的索引最有可能扩展。

拥有者	表 名
AR	AR_CUSTOMER_PROFILES
AR	AR_PAYMENT_SCHEDULES
AP	AP_INVOICHE
AP	AP_PAYMENT_SCHEDULES
GL	GL_BALANCES
GL	GL_BUDGET_INTERIM
GL	GL_JE_HEADERS
GL	GL_JE_LINES
GL	GL_SUMMARY_INTERIM
PO	PO_VENDOR_SITES
PO	PO_VENDORS

在Financials数据库中遇到扩展问题的实际表可能与此不同。创建并运行 Command Center监控数据库(见第6章)将大大简化确定问题区域的过程。这里要注意的是，由于大多数 FND表定义应用程序的窗体界面，所以相当稳定且容量很大。其他表 (如GL.GL_CODE_COMBINATIONS)在初始数据库装载和建立过程中有很大的扩展，然后又非常稳定。

11.2.9 优化程序

Oracle Financials软件依靠基于规则的优化程序来确定执行查询所使用的路径。不应分析Oracle Financials表。

为了增强Oracle Financials软件的性能，开发了一个基于成本的常规优化程序。这个优化程序填充 GL.GL_SEGMENT_RATIOS表，该表将决定用于每个 Financial Statement Generator(FSG，Financials语句生成程序)报表的索引。当FSG报表运行时，优化程序为这个报表抑制除了最具选择性的索引之外的所有索引。

由于FSG报表依靠GL.GL_SEGMENT_RATIOS表，所以只有当该表为最新时， FSG报表才能选择到合适索引。因此在以下情况下要运行优化程序以重新填充该表：当通用分类帐(ledger)的帐户结构变化时，当汇总模板结构改变时，在阶段性使用之后 (如月终结帐之后)。若要运行GL优化程序，Financials系统管理员要从Financials Navigate|Setup|System菜单中选择优化程序选项。尽管数据库管理员不直接参与这项任务，但 GL优化程序直接影响到性能和合适的时间安排，因此应在数据库管理员和财务系统管理员之间进行协商。

11.3 管理Oracle Designer的特殊准则

Oracle Designer是其较早期Oralce *CASE产品的后继产品。Oracle Designer是一种客户机

/服务器应用程序，数据资料档案库在服务器端而应用程序的显示在客户机端 (通常是一个基于Windows的PC机)。客户机软件通常要有300MB以上的安装空间。

在服务器上，要按照第4章介绍的准则设置数据库和应用程序。以下几小节介绍设置和管理Oracle Designer软件包的准则。 Oracle Designer数据库结构的大多数管理工作都可通过Oracle Designer提供的Repository Administration Utility(资料档案库管理实用程序)来完成。

11.3.1 数据库结构

Oracle Designer资料档案库应存储在一个单独的实例中，与所有活动的应用开发数据库分开。必须创建多个Oracle Designer资料档案库的实例 —— 分别对应开发、测试和产品。当执行一个新版本工具时，如果测试新版本的同时要提供对当前产品版本的开发支持，就必须提供第二组实例。

由于Oracle Designer频繁使用存储过程和软件包，所以SYSTEM表空间要大得足以容纳下存储的代码。开始时SYSTEM表空间在100MB到150MB之间，并监控其可用的剩余空间。

在应用程序中，可以用Respository Administration Utility的Repository Management屏幕，以特定的形式来处理数据库对象(如索引、视图、触发器和软件包)。这个屏幕可以简化在大量删除数据后撤消并重建索引的进程。可以用这一屏幕上的 Pre-Check(预检查)选项在执行SQL之前对它进行检查。由于要执行的SQL命令(如create view和create trigger)需要DDL(数据字典语言)锁，所以应只在没有用户使用数据库时执行这些命令。

1. 调整表和索引的大小

Oracle Designer资料档案库中的大多数记录存储在两个表中： SDD_ELEMENTS和SDD_STRUCTURE_ELEMENTS。确定这些表及其索引的大小要格外小心，因为它们大小的确定很大程度上受模型版本化(model versioning)策略和模型共享(model sharing)数量的影响，下一节中将介绍的抽取频率也影响资料档案库所需的空间。

模型版本化允许开发人员在Oracle Designer资料档案库中为同一个模型保存不同的版本。应用程序设计的数据存储在数据库中，这表明同一应用程序的两个版本将使应用程序在资料档案库中需要的空间加倍。必须和Oracle Designer用户一起为所有应用程序小组人员确认一个统一的版本策略。如果对一个模型创建多个版本是规则的例外，那么就必须对这种例外发生的频率进行估算，以便计算资料档案库可能需要的额外空间量。

模型共享可以共享资料档案库中不同模型之间的对象。例如，一个模型包含有CUSTOMER实体，第二个模型可以共享第一个模型的 CUSTOMER实体。共享资料档案库中对象的能力方便了企业范围实体和对象的开发，进而方便了包含所有应用程序的企业模型开发。

当共享属于另一模型的一个对象且然后又创建你的模型的一个新版本时，资料档案库将对所有拥有共享对象的模型进行拷贝。如果共享了属于另一模型的 CUSTOMER实体，然后又创建了模型的一个新版本，那么资料档案库将对你的模型和含有 CUSTOMER实体的模型创建一个新版本。由此可见，模型版本化和模型共享策略将严重影响资料档案库中所需的空间量。如果有5个模型，而且彼此之间共享对象，那么创建一个模型的新版本就要同时创建 5个新模型(一个是模型的拷贝，另外4个用于共享)。尽管只创建了一个模型的新版本，但在资料档案库中加倍使用了空间。

2. 管理回滚段

Oracle Designer应用程序有联机组件和批处理组件。联机组件由开发者用于应用程序设计与开发。批处理由资料档案库管理员用于大型数据处理事务 (例如抽取或模型版本化)。

应设计回滚段以支持联机用户。与联机应用程序相比，资料档案库中联机操作的数量要少一些。因为一般没有一致性的事务量被输入到这个资料档案库中。

若要支持模型版本化或抽取这样的大型事务，至少要有一个专用于这些活动的大型回滚段。抽取不是把数据从资料档案库写入数据库外部的平面文件中，而是通过以下命令建立资料档案库表的拷贝表：

```
create table EXTRACT_TABLE
as select * from REP_TABLE;
```

上面的命令把一个create table命令与一个隐式insert命令合并在一起。insert命令创建一个单一事务，该事务拥有所有来自 REP_TABLE表的数据，用来插入到抽取表。由于事务不能跨回滚段，必须有一个足够大的回滚段来处理一个事务。所以要有一个大容量回滚段用于资料档案库管理员的抽取工作。不过，这个大型回滚段不能反映应用程序的联机需求。 Oracle Designer环境的特点是低事务量。若要支持这样的事务量， 10MB的回滚段就足够了。可为每4～6个活跃事务创建一个回滚段。有关如何监控回滚段数据写入量和每个回滚段的事务数的信息，请参见第7章。

抽取表的存储空间需求镜像了应用程序设计数据的存储空间需求。因此在开始抽取之前，在资料档案库的表空间中至少有 50%的空间必须是未用的。抽取表使用的空间只是临时的，应用程序假定这些表要么导出到平面文件上，要么读入另一个资料档案库中。因此，资料档案库表空间中的额外空间不是用于联机用户，而是短时间地用于资料档案库管理员的抽取工作。尽管如此，这个额外空间对抽取过程至关重要。

3. 驻留软件包

可以使用Repository Administration Utility中的Repository Management屏幕来驻留Oracle Designer资料档案库中的任意软件包。不过要以批处理方式执行驻留操作，而不能通过特定的管理实用程序。应编写一个从 DBA_OBJECTS表中查询所有资料档案库软件包和过程的脚本文件(这里Object_Type= ‘ PACKAGE ’)，并修改数据库的启动脚本文件，以便自动包括驻留脚本文件。如果驻留是数据库启动的一部分，那么当使用应用程序时，数据库总是要驻留它的软件包。

11.3.2 init.ora参数

可以设定许多数据库初始化参数 (通过init.ora初始化参数文件)来提高Oracle Designer的性能。下面几小节列出了可以设定的主要参数：

1. SGA大小

System Global Area(SGA)是实例可以使用的内存区。从数据库中读出的数据保存在 SGA中加快了其他用户的检索速度。数据库的结构信息和事务返回的数据存储在 SGA中。另外一个称为SQL共享池的存储区，存储对数据库进行过操作且进行过语法分析的语句版本。

对于Oracle Designer来说，SGA中数据缓冲区至少要有 30MB才能高效地起作用。 SQL共享池也至少要有30MB以便支持驻留软件包。 SQL共享池的大小以字节为单位，通过 init.ora参

数SHARED_POOL_SIZE设置。数据块缓冲区的大小以数据库块为单位由 init.ora参数DB_BLOCK_BUFFERS设置。

2.打开的游标

init.ora参数OPEN_CURSORS限制每个用户进程可以同时拥有的打开的光标（环境区域）数量。这个参数的最大值依操作系统的不同而不同。设置该参数为最大值，尽管文档中这个值可能为255，但在大多数操作系统上可以超过这个值。

3.块大小

要想最大程度地提高Oracle Designer的效率，就要增加使用的Oracle数据块的大小。每个数据块不是使用2048字节，而要把它加大到4096字节或8192字节。增加数据库的块大小，将降低每一数据块额外开消的百分比。结果是每次I/O读更多的数据。由于每次I/O操作时读更多的数据，所以一次查询也就只需较少的几次I/O，从而也就提高了效率。这个参数的init.ora输入项如下：

```
db_block_size     = 4096
```

不能修改已存在的数据库的数据库块大小。

11.3.3 最活跃的表和索引

如本章前面所述，SDD_ELEMENTS表和SDD_STRUCTURE_ELEMENTS表是资料档案库中访问频率最高的表。如果从资料档案库中删除元素，就要从这两个表中删除记录。当记录被删除时，这些表的索引就会产生碎片，要定期重新创建这些索引来解决由索引造成的性能和空间管理问题。应该监控资料档案库中表和索引的段扩展情况，以确定Oracle Designer工具中有其他哪些表正频繁使用着。有关盘区分配趋势的监控情况，请参见第6章。

11.3.4 优化程序

Oracle Designer依靠基于规则的优化程序来确定所使用的查询执行路径。不应分析Oracle Designer资料档案库的表。

11.4 管理其他软件包和实用程序

以下几节，将介绍一些通用软件包(ConText、SQL*Loader和程序设计界面)的管理指南。

11.4.1 ConText

Oracle的ConText Cartridge依赖一系列后台服务器来处理查询中的文本部分。因此如下例所示，可以执行将关系和文本准则合并起来作为限制条件的查询。该例子假设PROSPECT表中有Resume列，该列可能含有文本数据，或指向包含数据的外部文件。

```
select Name
  from PROSPECT
 where Name like 'B%'
   and contains (Resume, 'digging') > 0;
```

上面的查询中含有一个contains子句，因此将调用ConText来决定哪一个Resume值含有单词“digging”，如果contains搜索的得分大于0且Name值以B字母开始，将由查询返回这个行。

也可以用ConText来执行“模糊匹配”、近似搜索和通配符搜索。当执行上面的查询时，

文本部分通过ConText处理，关系部分由RDBMS处理，结果合并在一起并返给用户。

要处理查询的文本部分，ConText使用数据库中的一系列后台进程和队列。数据库管理员应和开发基于ConText应用程序的工作组成员一道决定所需服务的数量和种类（DML、DDL和Document）。每次启动数据库时都必须启动这些服务器。

可以使用CTXSRV实用程序来启动ConText服务器。启动ConText服务器时要确定服务器的特性（personality），服务器的特性定义服务器可以处理的命令类型。在下面的例子中，用一个特性启动的服务器可以支持3种命令：表示DDL的D、表示DML的M和表示Document Services的S。当执行下面的命令时，*ctxsys_pass*要用CTXSYS帐号的口令来替换：

```
ctxsrv -user ctxsys/ctxsys_pass -personality DMS
```

如下面查询所示，可以通过CTX_SERVERS视图查看ConText服务器的状态：

```
column Ser_Name format A32

select Ser_Name,
       Ser_Status,
       Ser_Started_At
from CTX_SERVERS;
```

CTX_SERVERS的输出样例如下所示：

```
SER_NAME                         SER_STAT SER_START
-------------------------------- -------- ---------
DRSRV_42736                      IDLE     03-MAY-99
```

上面的例子显示在实例中已启动了一个ConText服务器，系统赋予服务器的名字是DRSRV_42736。若要关闭一个实例中的所有服务器，可以使用CTX_ADM软件包中的SHUTDOWN过程，命令如下：

```
execute CTX_ADM.SHUTDOWN;
```

如下面的例子所示，若要关闭一个ConText服务器，在执行CTX_ADM.SHUTDOWN过程时应将ConText服务器名指定为一个参数。服务器名在CTX_SERVERS的Ser_Name列中列出。

```
execute CTX_ADM.SHUTDOWN('DRSRV_42736');
```

CTX_ADM软件包的SHUTDOWN过程有第二个参数SDMode，它缺省为NULL。SDMode的值为0或NULL将造成正常关闭ConText服务器。SDMode的值为1表示立即关闭，值为2则强制中断。

若要启用ConText搜索功能（即使用contains子句查询），应在init.ora文件中设置以下参数：

```
text_enable = TRUE
```

在数据库中，CTXSYS用户拥有ConText数据字典。CTXSYS用户是唯一具有CTXADMIN角色的用户。其他与ConText相关的角色包括CTXAPP（应用程序拥有者）和CTXUSER（应用程序用户）。应向所有开发ConText应用程序的用户授予CTXAPP角色。当用户执行一个ConText相关查询时，查询记录存在中间表中。当查询结束时，数据库截断中间表，从而消除了由于使用中间表而可能引起的任何空间管理问题。

在Oracle8i中，Oracle对ConText的管理进行了重大改动。在以前的ConText版本中，先创建策略，然后创建基于这些策略的文本索引。在Oracle8i中，通过create index命令的一个扩展版本来创建文本索引。

在创建一个文本索引之前，首先必须规定优先权。优先权通过 CTX_DDL软件包的CREATE_PREFERENCE过程规定。

```
ctx_ddl.create_preference(preference_name  in varchar2,
                          object_name      in varchar2);
```

如果没有规定优先权，Oracle将使用系统缺省(见CTX_PARAMETERS视图)。一旦创建了优先权，就通过CTX_DDL软件包的SET_ATTRIBUTE过程为这些优先权创建属性。

```
ctx_ddl.set_attribute(preference_name in varchar2,
                      attribute_name  in varchar2,
                      attribute_value in varchar2);
```

例如，下面的命令在PROSPECT表的Resume列上创建一个文本索引Resume_Index：

```
create index Resume_Index on PROSPECT(Resume)
indextype is context
parameters('stoplist MY_STOP');
```

在上面的例子中，indextype is context子句通知Oracle，这是一个文本索引。只有一个参数(禁用词表)被规定；其他参数将使用缺省系统优先权。

在Oracle8i之前，重新组织一个文本索引需要使用 CTX_DDL.OPTIMIZE_INDEX过程。在Oracle8i中，可以用alter index 和drop index命令管理一个文本索引。例如，下面的命令将重建Resume_Index文本索引：

```
alter index Resume_Index rebuild;
```

若要改进重建文本索引的性能，请使用 ConText专用的optimize fast参数。如下面的例子所示，当使用optimize fast参数时，重建期间不从文本索引中取消被删除的值。

```
alter index Resume_Index rebuild parameters('optimize fast');
```

也可以使用alter index命令重新命名一个文本索引，改变其存储参数或改变文本索引的优先权。如下所示，若要撤消一个文本索引，可使用 drop index命令：

```
drop index Resume_Index;
```

11.4.2 SQL*Loader

SQL*Loader是将数据从外部文件装入 Oracle表的一个实用程序。作为数据库管理人员，在使用时应注意以下两方面的情况：

1)表和索引的大小是否适合期望的数据装载？

2)是否使用Direct Path选项？

表和索引大小的计算(见第5章)应考虑从平面文件中读入的数据。一般来说，已对数据的长度和数量进行了恰当的定义，因此对大小的估算应该是精确的。如果是对首次所提供数据的一次性装入，那么只有当数据全部读入且表的大小已经核实之后才创建索引。

Direct Path选项是将数据插入表中的一种快速方法。它绕过 insert语句的正常处理方法，直接写到表的数据块中。当使用Direct Path选项时，平面文件中的数据应按表中的索引列进行预排序。对于大型数据装载，效率的提高是可想而之的。

当然，这种效率的提高是有代价的。在这种情况，代价就是用于临时段的表空间。启动Direct Path选项时，在装载期间表的索引处于无效状态。在数据装入表时，新索引的键值写入临时段。装入结束后，旧的索引值与新值合并在一起，索引又重新有效。

Direct Path装载对临时段存储空间的需求是可想而之的。SQL*Loader需要临时表空间中有足够的有效空间来存储，最少是要装入表上所有索引的初始盘区。因为Direct Path选项通常用于大型数据装载，所以通常对临时表空间的空间需求量很大。对于未排序的数据装载，临时表空间的大小需求量是索引大小需求量的两倍。

如下面所示，若要确定索引的状态，可以查询DBA_INDEXS视图。有效的Status(状态)值为DIRECT LOAD和VALID。

```
select Owner,          /*Owner of the index*/
       Index_Name,     /*Name of the index*/
       Status          /*Either DIRECT LOAD or VALID*/
  from DBA_INDEXES;
```

如果索引在数据装载后还处于DIRECT PATH状态，装载就不符合索引的准则。例如，可能把备份记录装载到一个具有唯一索引的表中。当装载结束时，索引不能重新用于该表并且保持在DIRECT PATH状态。结果就要删除索引，纠正数据后重建索引。

1. SQL*Loader Direct Path选项的高水位标志问题

如第4章所述，Oracle为数据库中的每一个表都保存一个高水位标志。高水位标志是曾写有数据的最高数据块数；如果以后删除该数据，高水位标志则不变化。若要复位高水位标志，必须截断该表。

由于Direct Path装载数据而不是装载各个记录，所以高水位标志对SQL*Loader Direct Path用户很有意义。Direct Path把数据库装到表中高水位标志以上的部分。因此，如果表有一个没有反映其数据实际使用情况的高水位标志，就可能浪费宝贵的空间量。

例如，如果把100MB的数据装入一个表中并在以后删除它，高水位标志将一直设置在100MB处。如果以后对这个表执行Direct Path装入，将在100MB处开始添加新装载的数据；表存储空间的前100MB将是空的。

为避免浪费空间，必须用truncate命令使表的高水位标志返回其原始值。不过，truncate命令是删除表中全部记录的一个DDL命令，没有办法回滚删除。因此，如果使用Direct Path装载选项，应考虑使用Oracle的表分区选项。可以截断一个分区而不影响表中任何其他分区。若要截断一个分区，请使用alter table命令的truncate partition子句。如果正在使用Oracle8i引入的子分区选项，就可以给truncate命令指定一个要截断的子分区。如果不指定子分区，Oracle将截断分区的全部子分区。

在Oracle8i中，Oracle Call Interface(OCI)现在包括一个SQL*Loader使用的直接装载工具，销售商可以开始使用这种新工具通过OCI调用来执行批处理装载。与SQL*Loader Direct Path装载的情况一样，这些批处理装载将不重复使用表的高水位标志之下的任何自由空间。

11.4.3 程序设计接口

对Oracle的程序设计接口使应用开发人员能用第三代GL程序设计语言(例如C语言)编写可以访问数据库的程序。数据库管理员对程序设计接口最关心的是数据库和操作系统软件的升级问题。由于用户的程序使用Oracle核心软件库和操作系统软件库，所以对任何库的修改都要求对用户的程序重新链接。当数据库软件升级(如从版本8.0.5升级到8.1.5版本)或操作系统升级时，就会发生重新链接。