第6章　多数据库的监控

在会见DBA应聘者时，我总要向他们提出以下两个问题：

1) 什么是数据库成功的关键因素？

2) 如何监控它们？

看起来这好像是很简单的问题，但许多应聘者马上意识到，他们并不知道自己的数据库成功的关键因素，只是随口答道：“是不是空间问题？”。如果不知道这些因素，他们的监控系统将是不充分的或者过于充分。对于数据库的监控方式，必须考虑到它们特殊的结构和用途。监控的重点应当放在揭示系统执行中的问题上，而不是放在跟踪问题的征兆上。

扑灭宾馆中的一场火灾会引起重视灭火装置。但这并不意味着这个宾馆以后就不会发生火灾。要弄清楚导致系统出现问题的症结；否则，只是看到问题的表面现象，而仍然留下潜在的祸因。

要避免这种DBA灭火模式，需要做好以下4点：

1) 对应用程序如何使用数据库要有定义完好的协议。

2) 一个结构完美的数据库。

3) 一套测定数据库状态的度量标准。

4)进行测量与决定发展趋势的一个系统方法。

前两点在第3～5章中已经讨论。本章着重讨论第3点和第4点，提供测量及监控方法。本章的前一部分通过一个“Command Center”数据库讲述对数据库物理元素的测量，第二部分通过Oracle中的统计脚本文件，讨论对内存对象的监控。

6.1　常见问题

所有Oracle数据库都有一些潜在的问题，它们是：

- 表空间中的自由空间缺乏。
- 临时段的空间不足。
- 回滚段已达到扩展极限。
- 数据段及自由空间碎片。
- SGA区域大小设置得不正确。

本章将详细讨论一个有效的数据库监控系统，该系统能够检测这些方面的每一个不可能令人满意的值。下面几节，在提供一个适合监控的系统之前，将提供一个对所跟踪问题的简介。

6.1.1　表空间中的自由空间缺乏

数据库中的每一个表空间都有一些指定给它的数据文件。如果不使用自动扩展的数据文件，表空间中所有数据文件的空间总和就是表空间中可分配空间的上限。

当在表空间中创建一个段时，将从表空间的有效自由空间中为这个段的初始盘区分配空

间。在初始盘区填满数据时，段会获得另一个盘区。这样的扩展过程会一直继续下去，直到段达到最大的盘区数，或者表空间中的自由空间小于下一个盘区所需要的空间为止。

因此，表空间中的自由空间相当于是一个新段或现有段扩展所能用的未分配空间的缓冲区。如果表空间中的可用自由空间已不足以创建新段或盘区，就必须为此表空间增加另外的空间(通过增加新数据文件或重新设置现有数据文件大小)。

因此，不仅应当对表空间中当前可用的自由空间进行监控，还应对可用空间的变化趋势进行监控 —— 现在的可用自由空间比一星期前的可用空间是多还是少。必须能够确定当前空间分配的效率及对未来的预测。

6.1.2 临时段的空间不足

临时段在排序操作 (如大型查询、创建索引及联合)期间存储临时数据。当通过 create user 或alter user命令创建一个帐户时，每个用户就会被分配一个临时表空间。临时表空间应指向其他某个地方而不是SYSTEM表空间(缺省)。

创建一个临时段时，它使用这个表空间的缺省存储参数作为自己的存储参数。当临时段建立时，其存储参数不会因表空间的缺省存储参数的改变而改变。临时段随需要而自我扩展，当操作完成或遇到错误时就自行丢弃。由于临时段自身能够导致错误 (超过盘区最大值或表空间缺乏)，大型分类查询和操作应当考虑何时调整临时表空间的大小。

在执行一个排序操作时，如果临时表空间用完自由空间，这必将导致操作失败。既然很难准确估算临时段的大小，所以监控用来承载临时段的临时表空间是否足够大就显得非常重要。本章后面将介绍如何监控数据表空间的当前值及其变化趋势。

6.1.3 回滚段达到扩展极限

每一个发生在数据库内的事务都包含着回滚段。它们使数据库能在多事务处理之间保持读的一致性。可用回滚段的数量和大小在数据库创建时由 DBA确定，但以后不能修改。

由于回滚段创建在数据库内，所以它们必须在一个表空间内创建。称为 SYSTEM的第一个回滚段被存在SYSTEM表空间中，接下来的回滚段通常在另外的至少一个独立表空间中创建。因为它们是在表空间中创建的，所以它们的最大尺寸受到表空间中数据文件的空间量限制。

因为它们用与数据段相同的方式分配新空间，所以回滚段存在两个基本问题：超出表空间的可用自由空间和达到盘区的最大允许数量。当其中任一种情况发生时，强制回滚段扩展的事务将会失败。单独一个事务不能跨越多个回滚段。

因此，除了跟踪当前及以后包含回滚段的表空间的自由空间之外，每一个段的盘区数量也必须受到监控。

optimal存储参数仅对回滚段有效。这个参数通常被用来设置回滚段的最佳尺寸；回滚段超过这个尺寸时，会动态地删去未用的盘区以缩减回滚段。这个功能造成很难仅仅通过段空间的利用率来判断回滚段是否在扩展。相反，必须使用动态性能表跟踪每个回滚段扩展和收缩的次数。这里提供的监控系统监控回滚段空间的使用和它们扩展和收缩的次数。

6.1.4 数据段碎片

本章前面已经提到，Oracle通过使一个段能获得多个盘区的方式来管理段的空间。虽然这

使确定表尺寸具有灵活性，但也会导致性能退化。最理想的就是一个段的数据被存在一个可管理的盘区数量内。数据不必全部存储在一个单一的盘区内，但是盘区的数量不应太大，这样反而会影响管理段的能力。

如果一个段有多个盘区，就无法保证这些盘区是相邻存储的。对数据库中单个表的查询可能需要存储在几个不同物理位置上的数据，这可能对性能起负面影响。

数据库中的每一个段都有一个允许的最大盘区数量。自 Oracle7.3起，可以规定段的最大盘区数量不受限制。如果段的最大盘区数量设置为不受限制，则可以由数据库块的大小来决定盘区的最大数量。对于一个2048字节的块，段最多可达121个盘区；对4096字节的块，盘区的最大数量为249。

任何导致一个段试图超过其最大盘区数量的事务都会失败。任何不断扩展的段都会被错误设置尺寸，并且会降低性能。将段压缩在一个盘区内的情况，将在第 8章中叙述。本章中提供的监控工具将跟踪数据库中段的当前扩展及扩展趋势。

6.1.5 自由空间碎片

就像数据段能成为碎片一样，表空间中的可用自由空间也能成为碎片。这个问题在pctincrease缺省值设为0的表空间中最普遍。若要将 pctincrease 为0 的影响降至最低限度，可以在夜间运行一个脚本文件以合并表空间的自由空间。

当删除一个段时，它的盘区被重新分配并且被标为“自由”盘区。然而，这些自由盘区通常不能与相邻的自由盘区再连接。这些自由盘区之间的障碍可能依然存在，对一个新数据盘区可用的自由空间会受其影响。

如果表空间的 pctincrease缺省值不是0，Oracle自动将相邻自由盘区合并为一个大盘区。但是自由盘区可能被其他数据盘区物理地分开，阻碍它们与其他自由盘区合并。

为了检测这些潜在问题，本章提供的工具跟踪每一个表空间中可用自由盘区的大小和数量。

6.1.6 SGA区域大小设置得不正确

用于存放数据库的共享内存对象的 System Global Area(SGA，系统全局区域)的大小通常只设置一次，此后很少被监控或调整，正确的 SGA大小对数据库的性能非常关键。因此，本章后半部分着重介绍SGA的使用方式。

6.2 目标选择

根据提到的普遍问题，以下统计值非常重要：

- 所有表空间中的自由空间。
- 所有表空间中自由空间的变化率。
- 任何时刻由临时段使用的空间总和。
- 回滚段的盘区大小和数量。
- 全部段的盘区数。

这里列出了为系统中每一个数据库应定制的可接受的最低起点。它应该被扩展以包括那些决定系统成败的统计数字。对于每一个统计数字，都应定义高、低限制。一旦定义了可接

受的范围，就可以执行监控系统。注意，这个范围可能与物理测量（例如一个表空间的自由空间)或者它们的变化率有关系。

6.3 最终产品

在描述监控系统的配置和应用之前，首先确定其输出内容。这里列出本章将要叙述的系统的清单例子。如果需要进一步的细节信息或其他信息，你可以修改这个系统。

如图6-1所示，第一份报表是所有数据库表空间的自由空间的趋势报表。它表明了每个数据库(DB_NM列)的每个表空间(TS列)中当前未被分配(自由)空间的百分比。当前自由空间百分比值显示在报表的Today列中，其他列表示前4周中每个表空间的自由空间百分比。Today列中的自由空间百分比值与前4周的值之间的变化显示在Change列中。首先列出了自由空间百分比方面出现最大消极变化的表空间。

Percent Free Trends for Tablespaces

DB_NM	TS	4Wks Ago	3Wks Ago	2Wks Ago	1Wk Ago	Today	Change
CASE	CASE	56	56	55	40	40	-16
	USERS	86	64	75	77	76	-10
	SYSTEM	22	22	22	21	21	-1
	TOOLS	25	25	25	25	25	
	RBS	32	32	32	32	32	
	TEMP	100	100	100	100	100	
CC1	CC	94	94	93	92	92	-2
	TESTS	71	70	70	70	70	-1
	SYSTEM	24	24	24	24	24	
	RBS	32	32	32	32	32	
	CCINDX	51	51	51	51	51	
	TEMP	100	100	100	100	100	

图6-1 表空间的自由空间趋势报表

图6-1示出了每个表空间中自由空间百分比的当前值、过去值和发展趋势。这个报表是本章以后将看到的监控脚本文件输出的一部分。

用于生成这个报表的SQL脚本文件包括一个变量，这个变量可设置成限制报表中只输出那些自由空间百分比的变化超过给定阈值的表空间，也可以通过这个变量限制只显示指定的数据库或表空间。要注意的是，这些用来确定系统是在“控制”之下还是“失控”的阈值范围也可以加入报表中。因此，可以把这个报表用作数据库的一个异常报表。

可以在图6-2所示报表中查看各段的盘区分配。从本章所述的监控脚本文件中生成的这个报表显示了当前包含十个以上盘区的所有段的分配趋势，它也显示了所有回滚段(无论其拥有多少盘区)。

图6-2所示的报表表明每一个段的当前盘区数量(Today列)。列出了段的数据库(DB_NM)、表空间(TS)、拥有者(Owner)、名称(Name)和类型(Type)来确定段的归属，同时也列出了以Oracle块为单位的段的当前尺寸(Blocks列)。除了过去4周的盘区数量的变化(Change列)外，也一并列出了当前和过去的盘区数量。

Extent Trends for Segments with 10 or more Extents

DB_NM	TS	Owner	Name	Type	Blocks	4Wks Ago	3Wks Ago	2Wks Ago	1Wk Ago	Today	Change
CASE	CASE	CASEMGR	TEMP_TBL	TABLE	100				20	20	20
			TEMP_IDX	INDEX	80				16	16	16
	RBS	SYSTEM	ROLL1	ROLLBACK	3800	19	19	19	19	19	
			ROLL2	ROLLBACK	3800	19	19	19	19	19	
	USERS	AL1	TEST1	TABLE	120		12	12	12	12	12
			TEST2	TABLE	140		14	14	14	14	14
CC1	RBS	SYSTEM	ROLL1	ROLLBACK	3800	19	19	19	19	19	
			ROLL2	ROLLBACK	3800	19	19	19	19	19	

图6-2 段的盘区发展趋势报表

图6-2中的报表为变成碎片的段示出了盘区数量的当前值、过去值以及发展趋势。也指出了所有回滚段的统计数字。这个报表是本章提供的监控系统输出的一部分。

这个报表也可以定制为显示那些已经变化的值。然而当所有警告(>9个盘区)记录被显示时，该报表则更有意义。此后将这个报表与自由空间发展趋势报表进行比较就能对数据库做出结论。有了这两个报表示例，就可以总结出：

- 所有回滚段显示出合适的尺寸。尽管在数据库中创建了新表，但所有回滚段在尺寸上都没有增加(注意，这里假设回滚段还没有达到最佳尺寸)。
- CASE数据库中的AL1用户在USERS表空间中已经创建了几个表。尽管它们使此表空间的自由空间减少，但这并不是三周前使自由空间降至 64%的原因。这个下降是由过渡表导致的(因为空间已经被恢复)。
- CASE_MGR帐户通过在CASE表空间中创建临时表和索引而影响了产品表空间。如果这些段不是产品应用程序的一部分，就应该将它们从表空间中移走。

本章余下部分描述的应用程序所产生的这两个报表，是测量自由空间、回滚段状态、盘区分配以及它们的发展趋势的有效方式。更重要的是，它们提供了数据库设计的准确信息，给出了应用程序的行为。报表表明回滚段大小恰当，但 CASE_MGR帐户在产品数据库中创建了开发表，这显示产品系统缺乏控制并可能最终导致产品失败。

由于有助于了解每个数据库的概要，所以最后的样例报表统计了 CC1数据库。它列出了所有文件和表空间以及各自的空间细节。 CC1数据库是Command Center数据库，下一节将对此进行介绍。

图6-3所示的样例输出分为两个部分，第一部分列出数据库中的每一个数据文件 (Fi1e nm列)以及被分配的表空间(Tablespace列)，同时也列出了每个数据文件的 Oracle块数(Orablocks)和磁盘块(DiskBlocks)。

报表的另一部分显示了表空间的自由空间统计数字。对于每一个表空间，都显示自由盘区数量(NumFrExts)，这个列指出表空间中的可用自由空间分裂成多少碎片；除了显示表空间中所有自由空间的总和 (SumFrBl)外，MaxFrExt列还显示以Oracle块为单位的最大自由盘区尺寸。PercentFr列显示未分配的表空间所占百分比。

Oracle Tablespaces in CC1
Check Date = 27-MAY-99

Tablespace	File nm	Orablocks	DiskBlocks
CC	/db03/oracle/CC1/cc.dbf	30,720	122,880
CCINDX	/db04/oracle/CC1/ccindx.dbf	20,480	81,920
RBS	/db02/oracle/CC1/rbs01.dbf	5,120	20,480
	/db02/oracle/CC1/rbs02.dbf	5,120	20,480
	/db02/oracle/CC1/rbs03.dbf	5,120	20,480
SYSTEM	/db01/oracle/CC1/sys01.dbf	10,240	40,960
TEMP	/db01/oracle/CC1/temp01.dbf	15,360	61,440
TESTS	/db04/oracle/CC1/tests01.dbf	30,720	122,880

Oracle Free Space Statistics for CC1
(Extent Sizes in Oracle blocks)
Check Date = 27-MAY-99

Tablespace	NumFrExts	MaxFrExt	SumFrBl	PERCENTFR	MaxFrPct	DiskFrBl	DiskBlocks
CC	1	21504	21504	70.00	100	86016	122880
CCINDX	1	15360	15360	75.00	100	61440	81920
RBS	3	2019	2057	13.39	98	8228	61440
SYSTEM	1	6758	6758	66.00	100	27032	40960
TEMP	6	12800	15360	100.00	83	15360	15360
TESTS	1	21504	21504	70.00	100	86016	122880

图6-3 样例空间统计报表

MaxFrPct列显示最大的自由盘区对可用总自由空间的比例，此列若高表明大多数可用自由空间位于一个盘区中。最后两列显示每个表空间的可用自由空间的磁盘块数 (DiskFrBl)、每一个表空间的总磁盘块数 (DiskBlocks)。图6-3所示的样例数据表明这个数据库使用 2KB的数据块、512字节的操作系统块。

图6-3中的报表提供了对数据库中空间使用情况的总览。第一部分表示自由空间从何而来 —— 属于表空间的数据文件，而第二部分表示这些自由空间当前是怎样使用的。这个报表是本章提供的监控系统输出的一部分。

将图6-1、图6-2和图6-3的报表结合起来，就能有效地衡量前面所列出的所有目标。它们都是基于对Command Center数据库的查询所产生的。这个数据库的设计方法在下节介绍，接下来说明数据获取和一系列标准查询以便为报表提供的一个基准。

6.4 建立Command Center数据库

建立一个独立的、仅仅用于监控其他系统的数据库，解决了传统 Oracle数据库监控能力方面存在的三个问题：

- 多数据库之间的监控活动可以联合进行。
- 监控活动不会影响正在监控的系统的空间利用。
- 对于正被监控的参数可以检测其发展趋势。

本节将讨论一个独立监控数据库的创建、构造和实现。

这里描述的系统是一个反应监控器 (reactive monitor)，由于设计良好、应用良好和监控良好的系统基本上安全可靠，所以它不是用来执行系统的灵活、实时监测的。由于仅仅着重于

这个数据库不能影响客户机/服务器或分布式数据库的性能(并且以后把整个功能给予系统管理小组)，所以理想情况下的监控系统应从一个系统或网络监控器调用。

此例中的监控数据库被给予实例名CC1，表示它是系统中的第一个Command Center数据库。根据系统结构情况，可能会希望有多个数据库来执行这个功能。此数据库的结构如下表所示：

SYSTEM表空间	20MB
RBS表空间	30MB
	2个10MB回滚段，10MB自由空间
CC 表空间	30MB
CCINDX 表空间	20MB
TESTS表空间	30MB
TEMP表空间	15MB
重做日志	3个2MB重做日志

存储监控结果的应用程序设计得相当简单。对于每个实例，系统都要存储有关位置和使用的描述信息，同时还要存储每个数据库中每个文件的信息（对于不打开被恢复的数据库时进行恢复，这是非常有用的信息）。也要存储每个表空间的自由空间统计数据，并且检查每个段的多余盘区。图6-4示出了监控表的物理数据库图示。

FILES表的视图被称作FILES_TS_VIEW，该视图通过按实例ID、表空间(TS)和检查日期(Check_Date)来分组FILES表而被创建。在对FILES表(已分配空间)和SPACE表(用过的和未用的空间)中的数据进行比较时需要这个视图。创建这些对象的 DDL在下面的程序清单中给出，对象在表6-1中描述。

表6-1 中央命令数据库中的表和视图

对 象	描 述
DBS	存储实例说明信息的表
FILES	存储文件信息的表
FILES_TS_VIEW	FILES表的视图，按表空间分组
SPACES	存储自由空间信息的表
EXTENTS	存储已用盘区信息的表

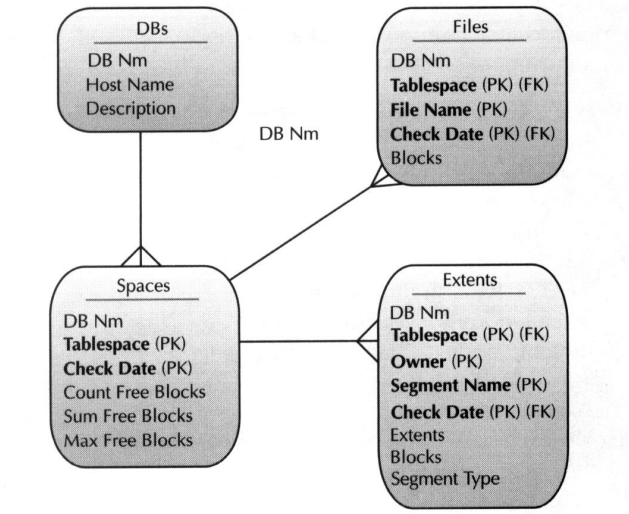

图6-4 Command Center数据库的物理数据库图示

```
drop table dbs;

rem * This table will store information about instances*/

create table DBS
(Db_Nm          VARCHAR2(8),      /*instance name*/
 Host_Nm        VARCHAR2(8),      /*host (server) name*/
 Description    VARCHAR2(80))     /*instance description*/
tablespace CC
storage (initial 64K next 64K pctincrease 0);

drop table FILES;

rem /*This table will store information about datafiles*/

create table FILES
(Db_Nm          VARCHAR2(8),      /*instance name*/
TS              VARCHAR2(30),     /*tablespace name*/
Check_Date      DATE,             /*date entry was made */
File_Nm         VARCHAR2(80),     /*file name*/
Blocks          NUMBER,           /*size of the file*/
primary key(Db_Nm, TS, Check_Date,File_Nm))
tablespace CC
storage (initial 128K next 128K pctincrease 0);

drop view FILES_TS_VIEW;

rem /*This view groups the file sizes by tablespace*/

create view FILES_TS_VIEW as
select
   Db_Nm,                         /*instance name*/
   TS,                            /*tablespace name*/
   Check_Date,                    /*date entry was made */
   SUM(Blocks) Sum_File_Blocks /*blocks allocated for ts*/
from FILES
group by
   Db_Nm,
   TS,
   Check_Date;

drop table SPACES;

rem /*This table will store information about free space*/

create table SPACES
(Db_Nm          VARCHAR2(8),    /*instance name*/
TS              VARCHAR2(30),   /*tablespace name*/
Check_Date      DATE,           /*date entry was made */
Count_Free_Blocks NUMBER,       /*number of free extents*/
Sum_Free_Blocks   NUMBER,       /*free space, in Ora blocks*/
Max_Free_Blocks   NUMBER,       /*largest free extent */
primary key (Db_Nm, Ts, Check_Date))
tablespace CC
storage (initial 128K next 128K pctincrease 0);

drop table EXTENTS;
```

```
rem /*This table will store information about extents */

create table EXTENTS
(Db_Nm    VARCHAR2(8),       /*instance name*/
TS        VARCHAR2(30),      /*tablespace name*/
Seg_Owner     VARCHAR2(30), /*segment owner*/
Seg_Name VARCHAR2(32),       /*segment name*/
Seg_Type VARCHAR2(17),       /*segment type*/
Extents   NUMBER,            /*number of extents allocated*/
Blocks    NUMBER,            /*number of blocks allocated*/
Check_Date     DATE,         /*date entry was made */
primary key (Db_Nm, TS, Seg_Owner, Seg_Name, Check_Date))
tablespace CC
storage (initial 128K next 128K pctincrease 0);
```

这些数据库结构允许DBA在所有数据库中跟踪前面列出的全部目标。注意，这里没有针对回滚段的表。如果回滚段的盘区数量超过规定的范围，就通过EXTENTS表跟踪它们。

注意 可以修改存储参数以便与第5章中确定的标准相匹配。

6.4 .1 数据获取

表6-1列出的第一个对象DBS代表有多个数据库的站点。可以使用DBS表输入有关实例的说明信息，它也是此应用程序中唯一需要人工输入数据的表。所有其他数据会自动被装入表中，全部标准数据报表也都是自动产生的，也可以使用可用的特定性能。

填充这些表所需要的数据可通过每个数据库的SYSTEM帐户访问(另一DBA帐户也能用于此目的，这个帐户要求访问DBA权限的表，因此一个指定的系统角色或SELECT_CATALOG_ROLE可用于此目的)。

在CC1数据库中，可以创建一个拥有这个监控应用程序的帐户。这个帐户不需要DBA角色。在这个帐户中，创建到各远程数据库DBA权限帐户的私有数据库链接，这个数据库链接的名称应与它链接的实例名称相同。例如：

```
create database link CASE
connect to system identified by manager
using 'case';
```

这个链接在用服务名“case”识别的数据库中访问SYSTEM帐户。使用时，将会作为用户系统登录到这个数据库中，口令为manager。

要注意的是，如果用户名和口令在远程系统上和本地系统上都相同，就不需要connect to行。有关数据库链接的connect to current user子句的细节，参见第9章。

注意 任何可以对这个监控数据库进行非法访问的人，都可以通过你建立的数据库链接来访问产品数据库。一定要认真保护这个监控数据库。

图6-5示出了数据获取过程的概况。

在数据获取过程中，一个批处理调度程序将用于调用启动进程的命令脚本文件。这项工作应该每天在非峰值工作时间进行。

如图6-5所示，监控应用程序将执行以下几步操作：

1) 命令脚本文件(称为ins_cc1)调用一个列出所有监控数据库的SQL*Plus脚本文件(称为inserts.sql)。

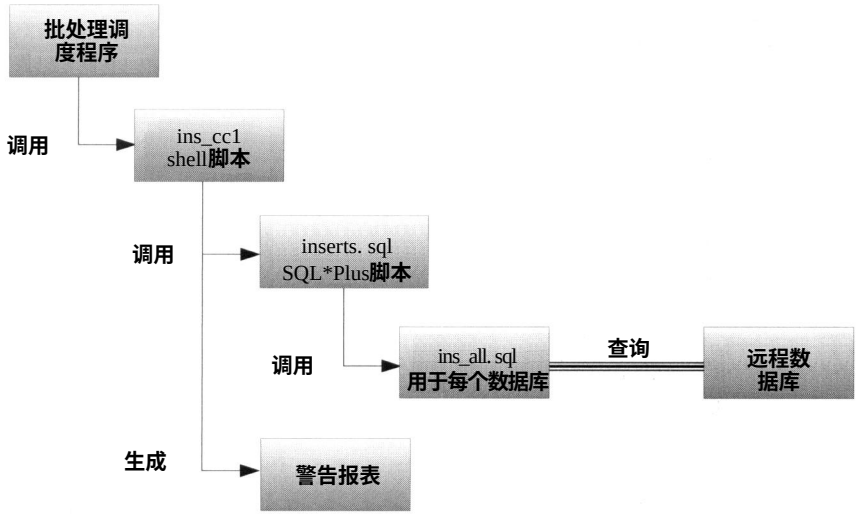

图6-5 数据库监控的过程流程图

2) 通过SQL*Plus脚本文件(称为ins_a1l.sql)检查每一个数据库。

3) 这些查询结果被ins_all.sql存储在Command Center数据库的表中。

4) 然后命令脚本文件在Command Center数据库中产生基于近期值的报警报表。

为了启动这个系统，可构造一个利用操作系统的批处理调度程序运行的命令文件。通过cron调用的UNIX shell文件例子如下所示。对于NT，可使用at命令。下面的脚本文件首先建立指向CC1数据库的环境变量。然后通过 Server Manager(服务管理器)打开CC1数据库。一旦数据库被打开，就执行一条命令来启动insert.sql脚本文件，数据处理完成后，CC1数据库关闭。

```
# file:  ins_cc1
#
# This script is run once daily to insert records into the
# CC1 database recording the space usage of the databases
# listed in inserts.sql file called by this file.  New
# databases need to have new links created in CC1 and have
# to have entries in the # inserts.sql script.
#
ORACLE_SID=cc1; export ORACLE_SID
ORAENV_ASK=NO; export ORAENV_ASK
. oraenv
cd /orasw/dba/CC1
svrmgrl <<EOF
connect internal
startup;
!sqlplus / @inserts
shutdown
EOF
```

注意，这个文件假设当监控没有发生时数据库是关闭的。这使 CC1的SGA和后台进程通常使用的内存在正常工作时间是自由的。假若一天发生一次监控，应在系统轻载时的固定时间进行监控。

前面列出的shell脚本文件调用 /orasw/dba/CC1目录下的inserts.sql文件。下面是一个insert.sql文件的例子。这个脚本文件为每个被监控的数据库调用 ins_all.sql脚本文件，当它完成之后，就执行两个名为space_watcher.sql和extent_watcher.sql的报警报表。

```
rem
rem  file:  inserts.sql
rem  location:  /orasw/dba/CC1
rem  Called from ins_cc1 shell script.
rem  New entries must be made here every time a new database
rem  is added to the system.
rem
set verify off
@ins_all CASE
@ins_all CC1
analyze table FILES compute statistics;
analyze table SPACES compute statistics;
analyze table EXTENTS compute statistics;
@space_watcher
@extent_watcher
```

inserts.sql脚本文件列出了即将收集监控统计数字的所有数据库，当新的数据库加入系统时，就会为它产生一个附加行。这个文件因而提供了一层管理过程，它不需要修改提交的批处理程序或访问远程数据库的SQL语句就可以轻松地变化。

对两个数据库(CASE和CC1)执行ins_all.sql之后，inserts.sql脚本文件对存储监控数据的表进行分析。为保证报表的性能不受监控表的影响，必须对这些表进行分析。由于监控表中的数据变化频繁，所以必须经常分析这些表。

这个脚本文件中的最后两行调用两个名为 space_watcher.sq1和extent_watcher.sql的SQL脚本文件。这两个脚本文件将分别生成图 6-1、图6-2所示的“ Perfect Free by Tablespace ”和“ Segments with over 10 Extents ”报表的版本。因为它们将被用于报警，所以只有变化超极限的行才会在下一个列表中出现。

迄今为止，我们已经见到了前两层数据获取系统，沿着图 6-5中所述的处理路径，下一步将要运行一个把记录插入到所有可应用监控的表的脚本文件 (ins_all.sql)，这些记录基于对远程数据库的查询。这些查询使用前面创建的数据库链接依次搜索实例并统计返回的记录。下面的清单示出了插入记录到FILES、SPACES和EXTENTS表的 ins_all.sql脚本文件。

ins_all.sql的第一部分把记录插入 FILES表中，它查询每个数据库的所有数据文件信息、表空间信息及其尺寸。

第二部分查询每个数据库的自由空间统计数字，它在 SPACES表中存储它的输出，记录每一个表空间的自由盘区数、可用自由空间的总和、最大单个自由盘区的尺寸。

第三部分检查段对空间的使用并将其结果存储在 EXTENTS表中。它记录标识段所需的信息(例如拥有者和名称)，及其当前大小和盘区数。为了限制查询返回的记录数量，只有选择回滚段和大于9个盘区的段。

```
rem
rem  file:  ins_all.sql
rem  location:  /orasw/dba/CC1
rem  Used to perform all inserts into CC1 monitoring
rem  tables.  This script is called from inserts.sql for
rem  each instance.
rem  For best results, name the database links after the
rem  instances they access.
rem
insert into FILES
   (Db_Nm,
   TS,
```

```
    Check_Date,
    File_Nm,
    Blocks)
select
    UPPER('&&1'),      /*insert database link,instance name*/
    Tablespace_Name,   /*tablespace name*/
    TRUNC(SysDate),    /*date query is being performed*/
    File_Name,         /*full name of database file*/
    Blocks             /*number of database blocks in file*/
from sys.DBA_DATA_FILES@&&1
/
commit;
rem
insert into SPACES
    (Db_Nm,
    Check_Date,
    TS,
    Count_Free_Blocks,
    Sum_Free_Blocks,
    Max_Free_Blocks)
select
    UPPER('&&1'),      /*insert database link,instance name*/
    TRUNC(SysDate),    /*date query is being performed*/
    Tablespace_Name,   /*tablespace name*/
    COUNT(Blocks),     /*num. of free space entries */
    SUM(Blocks),       /*total free space in the tablespace*/
    MAX(Blocks)        /*largest free extent in the ts*/
from sys.DBA_FREE_SPACE@&&1
group by Tablespace_Name
/
commit;
rem
insert into EXTENTS
    (Db_Nm,
    TS,
    Seg_Owner,
    Seg_Name,
    Seg_Type,
    Extents,
    Blocks,
    Check_Date)
select
    UPPER('&&1'),     /*insert database link,instance name*/
    Tablespace_Name,  /*tablespace name*/
    Owner,            /*owner of the segment*/
    Segment_Name,     /*name of the segment*/
    Segment_Type,     /*type of segment (ex. TABLE, INDEX)*/
    Extents,          /*number of extents in the segment*/
    Blocks,          /*number of database blocks in segment*/
    TRUNC(SysDate)    /*date the query is being performed*/
from sys.DBA_SEGMENTS@&&1
where Extents>9           /*only record extended segments*/
or Segment_Type = 'ROLLBACK'    /*or rollback segments*/
/
commit;
rem
undefine 1
```

注意，插入到EXTENTS表的只有回滚段和超过9个盘区的段的记录。因此这是数据库中段的不完全清单。由于这两个where子句决定盘区报警报表的阈值，应将盘区范围改变成你所需要的值。

这一系列的脚本文件将执行生成前面所示的报表所需要的所有数据获取功能。这里假设它们以一日为基础运行，如果需要更频繁的监控，表的主键必须修正为包括 Check_Date和Check_Hour(以便每小时报告一次)。

既然数据已经插入CC1监控表，所以Oracle可以自动生成报警报表。这通过从inserts.sql文件中调用space_watcher.sq1和extent_watcher.sql文件来完成。

6.4.2 生成报警报表

前面提到的报警脚本文件的构成，完全依靠被监控系统的特性，它们只是对图 6-1和图6-2所示的常规自由空间和盘区变化趋势报表的修改。它们利用 where和group by子句来消除没有超过某个阈值的条目。这些值应该为每一个站点定制 —— 它们基于目标变量选择过程中定义的控制范围。

由于这些报表在把记录插入Command Center数据库后自动调用，DBA可能希望使它们自动地用电子邮件发送或打印，这样每天早上用户在开始访问数据库之前就可以看到它们。

接下来的脚本文件生成某些表空间的空间趋势报表，这些表空间的自由空间在前 4周至少改变了5%。

```
rem
rem  file:  space_watcher.sql
rem  location:  /orasw/dba/CC1
rem  Called from inserts.sql
rem
rem  ...like some watcher of the skies
rem  when a new planet swims into his ken (Keats)
rem
column Db_Nm format A8
column TS format A20
column Week4 format 999 heading "1Wk|Ago"
column Week3 format 999 heading "2Wks|Ago"
column Week2 format 999 heading "3Wks|Ago"
column Week1 format 999 heading "4Wks|Ago"
column Today format 999
column Change format 999

set pagesize 60
break on Db_Nm skip 2
ttitle center 'Tablespaces whose PercentFree values have -
decreased 5 pct this month' skip 2

select
   SPACES.Db_Nm,
   SPACES.TS,
   MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate-28),
      ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) Week1,
   MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate-21),
      ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) Week2,
   MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate-14),
      ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) Week3,
```

```
    MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate-7),
       ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) Week4,
    MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate),
       ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) Today,
    MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate),
       ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) -
    MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate-28),
       ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) Change
from SPACES, FILES_TS_VIEW FTV
where SPACES.Db_Nm = FTV.Db_Nm          /*same DB name*/
and SPACES.TS = FTV.TS                  /*same TS name*/
and SPACES.Check_Date = ftv.Check_Date  /*same check date*/
and exists                              /*does ts exist?*/
   (select 'x' from spaces x
   where x.db_nm = SPACES.db_nm
   and x.ts = SPACES.ts
   and x.Check_Date = TRUNC(SysDate))
group by
   SPACES.Db_Nm,
   SPACES.Ts
having               /*has percentfree dropped 5 pct?*/
(  MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate),
         ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) -
   MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate-28),
         ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0))
>5    )
or                  /*is percentfree less than 10?*/
( MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate),
      ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) <10)
order by SPACES.Db_Nm,
   DECODE(MAX(DECODE(SPACES.Check_Date,TRUNC(SysDate),
     ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) -
   MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate-28),
     ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)),0,9999,
   MAX(DECODE(SPACES.Check_Date,TRUNC(SysDate),
     ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0)) -
   MAX(DECODE(SPACES.Check_Date, TRUNC(SysDate-28),
     ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0))),
   MAX(DECODE(SPACES.Check_Date,TRUNC(SysDate),
     ROUND(100*Sum_Free_Blocks/Sum_File_Blocks),0))

spool space_watcher.lst
/
spool off
```

如果exists部分不考虑上面的查询，所有表空间都将被显示；即使它们从数据库中删除之后也如此。having子句中的两个限制条件定义报警报表的阈值。在这个例子中，只有那些在过去28天内自由空间百分比下降5%以上的表空间才会通过第一个having条件；第二个having条件识别那些当前自由空间百分比不到10%的表空间，而不管其发展趋势如何。

如果特定数据库中的增值变化是其成功的关键，就可能要在这个报表中加入一个having子句来列出这些特定的数据库，而不管其发展趋势如何。

图6-6示出了样本输出(基于图6-1的自由空间百分比趋势报表)。

```
                Tablespaces whose Percent Free values have
                      decreased 5 pct this month

                                4Wks   3Wks   2Wks   1Wk
DB_NM          TS                Ago    Ago    Ago    Ago   Today  Change
-----------    ----------     ----   ----   ----   ----   -----  ------
CASE           CASE              56     56     55     40      40     -16
               USERS             86     64     75     77      76     -10
```

图6-6 表空间自由空间百分比趋势报警报表样例

注意这些报警报表不显示后来已经被解决的波动（例如，将不显示4周百分比自由空间趋势70-70-20-70-70）。这个报警报表将以一日为基础运行和观察，如果一个表空间的空间问题已经解决，它就不再显示在报警报表中。

第二个报表是盘区监视器(extent_watcher.sql)。与空间监视器报表一样，SQL*Plus查询使用group by子句将多行变成一行的多列。这个报表提供图 6-2所示的常规范围趋势报表的数据子集。

```
rem
rem  file:  ext_watcher.sql
rem  location:  /orasw/dba/CC1
rem  Called from inserts.sql
rem
rem  ...like some watcher of the skies
rem  when a new planet swims into his ken (Keats)
rem
column Db_Nm format A8
column TS format A18
column Seg_Owner format a14
column Seg_Name format a32
column Seg_Type format a8
column Blocks format 99999999
column Week4 format 999 heading "1Wk|Ago"
column Week3 format 999 heading "2Wks|Ago"
column Week2 format 999 heading "3Wks|Ago"
column Week1 format 999 heading "4Wks|Ago"
column Today format 999
column Change format 999

set pagesize 60 linesize 132
break on Db_Nm skip 2 on TS skip 1 on Seg_Owner
ttitle center 'Segments whose extent count is over 10' -
skip 2

select
   EXTENTS.Db_Nm,
   EXTENTS.TS,
   EXTENTS.Seg_Owner,
   EXTENTS.Seg_Name,
   EXTENTS.Seg_Type,
   MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate),
         Blocks,0)) Blocks,
   MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate-28),
         Extents,0)) Week1,
```

```
    MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate-21),
          Extents,0)) Week2,
    MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate-14),
          Extents,0)) Week3,
    MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate-7),
          Extents,0)) Week4,
    MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate),
          Extents,0)) Today,
    MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate),
          Extents,0)) -
    MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate-28),
          Extents,0)) Change
from EXTENTS
where exists  /*did this segment show up today?*/
   (select 'x' from EXTENTS x
   where x.Db_Nm = EXTENTS.Db_Nm
   and x.TS = EXTENTS.TS
   and x.Seg_Owner = EXTENTS.Seg_Owner
   and x.Seg_Name = EXTENTS.Seg_Name
   and x.Seg_Type = EXTENTS.Seg_Type
   and x.Check_Date = TRUNC(SysDate))
group by
   EXTENTS.Db_Nm,
   EXTENTS.TS,
   EXTENTS.Seg_Owner,
   EXTENTS.Seg_Name,
   EXTENTS.Seg_Type
order by EXTENTS.Db_Nm, EXTENTS.TS,
   DECODE(MAX(DECODE(EXTENTS.Check_Date,TRUNC(SysDate),
          Extents,0)) -
   MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate-28),
          Extents,0)),0,-9999,
   MAX(DECODE(EXTENTS.Check_Date,TRUNC(SysDate),
          Extents,0)) -
   MAX(DECODE(EXTENTS.Check_Date, TRUNC(SysDate-28),
          Extents,0))) desc,
   MAX(DECODE(EXTENTS.Check_Date,TRUNC(SysDate),
          Extents,0)) desc

spool extent_watcher.lst
/
spool off
```

这个查询(限制记录显示)的唯一选项是exists子句，它决定是否从当天的查询中返回段。这个报表假定“盘区数”变量的阈值在插入到 EXTENTS表(见数据获取一节)期间被实施。一个数据库中可能有成千上万个段， EXTENTS表是唯一在插入时而不是查询时实施阈值的表。

如果特定段中的增量变化是成功的关键，那么就可能要把 where子句添加到in_all.sql脚本文件部分，这个脚本文件把那些段的行插入到 EXTENTS表中，然后该报表就会列出那些特定段，而不管其趋势如何。

图6-7示出了extent_watcher.sql报表的输出样本，与图 6-2完全一样。这是因为插入到EXTENTS表的记录只是那些已经成为碎片的段，没有产生报警报表所需要的附加限制。

Extent Trends for Segments with 10 or more Extents

DB_NM	TS	Owner	Name	Type	Blocks	4Wks Ago	3Wks Ago	2Wks Ago	1Wk Ago	Today	Change
CASE	CASE	CASEMGR	TEMP_TBL	TABLE	100				20	20	20
			TEMP_IDX	INDEX	80				16	16	16
	RBS	SYSTEM	ROLL1	ROLLBACK	3800	19	19	19	19	19	
			ROLL2	ROLLBACK	3800	19	19	19	19	19	
	USERS	AL1	TEST1	TABLE	120		12	12	12	12	12
			TEST2	TABLE	140		14	14	14	14	14
CC1	RBS	SYSTEM	ROLL1	ROLLBACK	3800	19	19	19	19	19	
			ROLL2	ROLLBACK	3800	19	19	19	19	19	

图6-7 盘区使用趋势的报警报表样例

6.4.3 空间汇总报表

由于每天的统计数据存储在CC1 Command Center数据库中，所以可以为任何数据库、任何指定的日期生成一个汇总报表。应通过批调度程序每周产生一个这种报表，不必当时就打印出来。但是对于DBA来说，它必须有效联机。使报表有效联机将缩短获得报表的延时，因为CC1数据库在每天的批运行之后总是保持关闭状态(像在ins_cc1 shell脚本文件中那样)。

SQL*Plus报表应该为每个数据库运行一次。它产生一个输出文件，这个输出文件的名字包括在查询中用到的数据库链接的名字。这个脚本文件包含以下3个参数：

1) 数据库链接名(因为它将被存储在输出文件名中，所以应该有与它访问的实例相同的名称)。

2) 检查日期(因为报表能对任何一天运行)。

3) Oracle块大小(例如2KB)与主机操作块大小(例如512字节)的比例，对于这两个块，比例为2048/512=4。

此报表分成两部分，第一部分查询FILES表以确定当前文件名和数据库的大小。第二部分对SPACES表的值(自由空间大小)和FILES表的值(通过FILES_TS_VIEW视图)进行比较。由于这些表包括了分配给表空间的空间信息和其中必须分配的数量，所以每个表空间中的剩余自由空间的百分比可以计算出来。

```
rem
rem space_summary.sql
rem  parameter 1: database link name
rem  parameter 2: check date
rem  parameter 3: ratio of Oracle to OS block size
rem
rem  to call this report from within sqlplus:
rem  @space_summary link_name Check_Date block_ratio
rem
rem  Example:
rem  @space_summary CASE 27-MAY-99 4
rem
```

```
rem  Should be called weekly for each database.
rem
set pagesize 60 linesize 132 verify off feedback off
set newpage 0
column TS heading 'Tablespace' format A18
column File_Nm heading 'File nm' format A40
column Blocks heading 'Orablocks'
column Percentfree format 999.99
column Diskblocks format 99999999
column Cfb format 9999999 heading 'NumFrExts'
column Mfb format 9999999 heading 'MaxFrExt'
column Sfb format 9999999 heading 'SumFrBl'
column Dfrb format 9999999 heading 'DiskFrBl'
column Sum_File_Blocks heading 'DiskBlocks'
column Maxfrpct heading 'MaxFrPct' format 9999999

break on TS
ttitle center 'Oracle Tablespaces in ' &&1 skip center -
'Check Date = ' &&2 skip 2 center
spool &&1._space_summary.lst
select
   Ts,                    /*tablespace name*/
   File_Nm,               /*file name*/
   Blocks,                /*Oracle blocks in the file*/
   Blocks*&&3 Diskblocks  /*OS blocks in the file*/
  from FILES
 where Check_Date = '&&2'
   and Db_Nm = UPPER('&&1')
 order by TS, File_Nm
/
ttitle center 'Oracle Free Space Statistics for ' &&1 -
skip center '(Extent Sizes in Oracle blocks)' skip center -
 'Check Date = ' &&2 skip 2
select
   SPACES.TS,                     /*tablespace name*/
   SPACES.Count_Free_Blocks Cfb, /*number of free extents*/
   SPACES.Max_Free_Blocks Mfb,    /*lgst free extent*/
   SPACES.Sum_Free_Blocks Sfb,    /*sum of free space*/
   ROUND(100*Sum_Free_Blocks/Sum_File_Blocks,2)
        Percentfree,              /*percent free in TS*/
   ROUND(100*Max_Free_Blocks/Sum_Free_Blocks,2)
    Maxfrpct,                  /*ratio of largest extent to sum*/
   SPACES.Sum_Free_Blocks*&&3 Dfrb, /*disk blocks free*/
   Sum_File_Blocks*&&3 Sum_File_Blocks /*disk blocks allocated*/
   SPACES.Sum_Free_Blocks*&&3 Dfrb, /*disk blocks free*/
   Sum_File_Blocks*&&3 Sum_File_Blocks /*disk blocks allocated*/
 from SPACES, FILES_TS_VIEW FTV
where SPACES.Db_Nm = FTV.Db_Nm
  and SPACES.TS = FTV.TS
  and SPACES.Check_Date = FTV.Check_Date
  and SPACES.Db_Nm = UPPER('&&1')
  and SPACES.Check_Date = '&&2'
/
spool off
undefine 1
undefine 2
undefine 3
```

图6-8示出了样例输出报表。根据查询中列出的spool命令，输出文件名为CC1_space_summary.lst。这是基于将句点(.)用作SQL*P1us中的连接字符。

Oracle Tablespaces in CC1
Check Date = 27-MAY-99

Tablespace	File nm	Orablocks	DiskBlocks
CC	/db03/oracle/CC1/cc.dbf	30,720	122,880
CCINDX	/db04/oracle/CC1/ccindx.dbf	20,480	81,920
RBS	/db02/oracle/CC1/rbs01.dbf	5,120	20,480
	/db02/oracle/CC1/rbs02.dbf	5,120	20,480
	/db02/oracle/CC1/rbs03.dbf	5,120	20,480
SYSTEM	/db01/oracle/CC1/sys01.dbf	10,240	40,960
TEMP	/db01/oracle/CC1/temp01.dbf	15,360	61,440
TESTS	/db04/oracle/CC1/tests01.dbf	30,720	122,880

Oracle Free Space Statistics for CC1
(Extent Sizes in Oracle blocks)
Check Date = 27-MAY-99

Tablespace	NumFrExts	MaxFrExt	SumFrBl	PERCENTFR	MaxFrPct	DiskFrBl	DiskBlocks
CC	1	21504	21504	70.00	100	86016	122880
CCINDX	1	15360	15360	75.00	100	61440	81920
RBS	3	2019	2057	13.39	98	8228	61440
SYSTEM	1	6758	6758	66.00	100	27032	40960
TEMP	6	12800	61440	100.00	83	15360	15360
TESTS	1	21504	21504	70.00	100	86016	122880

图6-8 空间汇总报表样例

图6-8与本章前面的图6-3相同，该图分成两部分。第一部分列出了数据库中的每一个数据文件(File nm 列)以及它所属的表空间 (Tablespace列)。各个数据文件中的 Oracle块数(Orablocks)和磁盘块数(DiskBlocks)也在这部分描述。

报表的第二部分显示了表空间的自由空间统计数字，对每一个表空间，显示其自由盘区数(NumFrExts)，此列指出表空间的有效自由空间分成多少个碎片。除了表空间中所有自由空间的总和(SumFrB1)外，MaxFrExt列中还显示Oracle块中的最大单个自由盘区。未分配的表空间百分比显示在PercentFr列。

MaxFrPct列显示出一个最大自由盘区与总可用自由空间的比例，此列是一个高值，它表明大多数可用自由空间位于一个盘区中。最后两列显示每一个表空间的可用自由空间的磁盘块数(DiskFrBl列)和总可用磁盘块数(DiskBlocks列)。图6-8所示的样本数据是：一个数据库使用的数据库块大小为2KB，操作系统的块大小为512字节。

6.4.4 数据过滤

由于未被检查，前面提到的表会一直增长到它们占用所有可用表空间中的自由空间为止。为避免出现与数据容量相关的问题，应该定期从 EXTENTS、SPACES和FILES表中删除记录。

数据的保留量依需要而定。如果从来不需要这些表中有超过 60天的旧数据，那么可以使

数据过滤成为数据插入进程的一部分。例如，在执行插入的 ins_all.sql脚本文件的结尾，可以添加以下命令：

```
delete from FILES
 where Check_Date < SysDate-60;

commit;

delete from SPACES
 where Check_Date < SysDate-60;

commit;

delete from EXTENTS
 where Check_Date < SysDate-60;

commit;
```

如果每天执行这个命令，delete事务的大小应可以被回滚段支持。

如果表变大，那么可以通过在 FILES、SPACES和EXTENTS表的Check_Date列上创建索引来改善删除时的性能。

6.5 监控内存对象

Oracle的内存对象(例如SGA和后台进程)也能够被监控，由于大多数对后台进程的监控是在操作系统级进行(并且操作系统专用)，所以这一节将集中讨论调整 SGA。

Oracle不断地更新一系列内部统计数字。当确信在监控检查中数据库不会关闭时，这些统计数字应只被存在跟踪表中(例如本章关于空间监控的各节中所用的表)。每当数据库关闭并重新启动时，这些内部统计数字都要复位。

为了方便监控有关SGA使用的统计，Oracle提供了两个应修改的脚本文件UTLBSTAT.SQL和UTLESTAT.SQL。它们位于Oracle软件主目录的/rdbms/admin子目录下。第一个文件UTLBSTAT(开始统计)创建一系列表并且将它们与当时数据库的统计数字存放在一起；第二个文件 UTLESTAT稍后运行，它创建一系列基于当时数据库中统计数字的表，然后生成一个报表(称为REPORT.TXT)，报表列出统计数字在开始和结束这两个脚本文件运行时间间隔内发生的变化。

6.5.1 UTLBSTAT及UTLESTAT的必要修改

在开始运行Oracle提供的统计脚本文件之前。先对它们进行修改。首先修改脚本文件使其包括数据库链接作为 from子句的一部分 —— 这将允许它们从Command Center数据库运行。然后，添加一个tabl espace子句以便将段存储在一个数据或暂存区表空间中。

为什么要进行这两个修改呢？不这样做就会产生两个问题：第一个问题是，由于这些脚本文件在表中插入记录，所以把表存储在被自动监控的数据库中会使数据库的文件 I/O统计出现偏差(因为监控活动影响I/O操作)。第二个问题是，为了从内部连接的 Server Manager(服务器管理器)中运行，应改写脚本文件。如果没有可选择的表空间被命名， SYSTEM表空间会由于这些脚本文件中的 create table和drop table操作而成为碎片。有关对碎片及其解决方案的说明，请参见第4章。

注意 现在可以在SQL*Plus中进行内部连接；在Server Manager中的Server Manager格式化命令将被忽略。

开始和结束统计表由utlbstat.sql创建。然后在生成report.txt之前，utlestat.sql创建一系列表来存储开始和结束统计之间的变化。由于引用 SYS.FILE$表，这些脚本文件必须在一个DBA帐户下运行，由这些脚本文件引用的其余所有表都可以被有权访问监控视图 (V$视图)的用户访问。

为了从统计脚本文件中引用远程数据库，需要添加一个指向目标数据库的数据库链接。为了访问这些脚本文件所需的全部表，需要创建到目标 SYS帐户的链接或者对脚本文件使用的SYS表授予远程SYSTEM帐户权限。为了使风险最小，可创建一个 SYS链接来规定临时口令。

```
create database link CASE_STAT
connect to SYS identified by only_for_a_minute
using 'case';
```

一旦链接建立，就登录到 CASE数据库，将SYS口令变成only_for_a_minute(或者任意选择)，返回到CC1并运行统计脚本文件。完成后，复位SYS口令。

为了通知查询使用这个链接，给被查询的表名添加上数据库链接名。下面的清单显示修改为使用这个链接的utlbstat.sql的表创建部分。如果管理多个数据库，就用一个变量代替数据库链接名。

对这个例子，表将被存储在CC1 Command Center数据库的CC数据表空间中。每个create table命令都加上一个tablespace CC子句，并且每一个远程表的查询将加上 @case_stat子句，以便使用数据库链接CASE_STAT。

在这个脚本文件运行后，所有远程数据库的动态性能表的统计数据都被存储在本地数据库的CC表空间中。

```
rem
rem  Modified version of $ORACLE_HOME/rdbms/admin/utlbstat.sql
rem  Note the addition of tablespace clauses and database links
rem
rem ************************************************************
rem                  First create all the tables
rem ************************************************************
drop table stats$begin_stats;
create table stats$begin_stats
TABLESPACE CC
as select * from v$sysstat@CASE_STAT where 0 = 1;
drop table stats$end_stats;
create table stats$end_stats
TABLESPACE CC
as select * from stats$begin_stats;

drop table stats$begin_latch;
create table stats$begin_latch
TABLESPACE CC
as select * from v$latch@CASE_STAT where 0 = 1;

drop table stats$end_latch;
create table stats$end_latch
```

```
TABLESPACE CC
as select * from stats$begin_latch;

drop table stats$begin_roll;
create table stats$begin_roll
TABLESPACE CC
as select * from v$rollstat@CASE_STAT where 0 = 1;

drop table stats$end_roll;
create table stats$end_roll
TABLESPACE CC
as select * from stats$begin_roll;

drop table stats$begin_lib;
create table stats$begin_lib
TABLESPACE CC
as select * from v$librarycache@CASE_STAT where 0 = 1;

drop table stats$end_lib;
create table stats$end_lib
TABLESPACE CC
as select * from stats$begin_lib;

drop table stats$begin_dc;
create table stats$begin_dc
TABLESPACE CC
as select * from v$rowcache@CASE_STAT where 0 = 1;

drop table stats$end_dc;
create table stats$end_dc
TABLESPACE CC
as select * from stats$begin_dc;

drop table stats$begin_event;
create table stats$begin_event
TABLESPACE CC
as select * from v$system_event@CASE_STAT where 0 = 1;

drop table stats$end_event;
create table stats$end_event
TABLESPACE CC
as select * from stats$begin_event;

drop table stats$begin_bck_event;
create table stats$begin_bck_event
  (event varchar2(200),
   total_waits number,
   time_waited number)
TABLESPACE CC;
drop table stats$end_bck_event;
create table stats$end_bck_event
as select * from stats$begin_bck_event;

drop table stats$dates;
create table stats$dates (stats_gather_times varchar2(100))
TABLESPACE CC;
```

```
drop view stats$file_view;
create view stats$file_view
as                  /*NOTE:  Have to change the FROM clause here*/
  select ts.name     ts,
         i.name      name,
         x.phyrds pyr,
         x.phywrts pyw,
         x.readtim prt,
         x.writetim pwt,
         x.phyblkrd pbr,
         x.phyblkwrt pbw,
         ROUND(i.bytes/1000000) megabytes_size
  from   v$filestat@CASE_STAT x,
         ts$@CASE_STAT ts,
         v$datafile@CASE_STAT i,
         file$@CASE_STAT f
 where i.file#=f.file#
   and ts.ts#=f.ts#
   and x.file#=f.file#;

drop table stats$begin_file;
create table stats$begin_file    /*No link needed here*/
TABLESPACE CC
as select * from stats$file_view where 0 = 1;

drop table stats$end_file;
create table stats$end_file
TABLESPACE CC
as select * from stats$begin_file;

drop table stats$begin_waitstat;
create table stats$begin_waitstat
TABLESPACE CC
as select * from v$waitstat@CASE_STAT where 1=0;
drop table stats$end_waitstat;
create table stats$end_waitstat
TABLESPACE CC
as select * from stats$begin_waitstat;
```

对Oracle8**版本的**utlbstat**的修改导致一个数学错误。在创建** STATS$FILE_VIEW**视图的脚本文件中，为**Oracle8**添加下面这个列：**

```
ROUND(i.bytes/1000000) megabytes_size
```

但是，1MB**不只有**1000000**字节，**1MB(1024*1024)**中有**1048576**个字节，必须将其修正为：**

```
ROUND(i.bytes/1048576) megabytes_size
```

由于创建表和插入基于远程表查询的数据， utlestat.sql**脚本文件也需要进行改变。在下面的程序清单中，**utlestat.sql**的统计收集部分使用了恰当的数据库链接。**

```
insert into stats$end_latch select * from v$latch@CASE_STAT;
insert into stats$end_stats select * from v$sysstat@CASE_STAT;
insert into stats$end_lib select * from v$librarycache@CASE_STAT;
update stats$dates set end_time = sysdate;
insert into stats$end_event select * from v$system_event@CASE_STAT;
insert into stats$end_bck_event
  select event, sum(total_waits), sum(time_waited)
```

```
    from v$session@CASE_STAT s, v$session_event@CASE_STAT e
    where type = 'BACKGROUND' and s.sid = e.sid
    group by event;
insert into stats$end_waitstat select * from v$waitstat@CASE_STAT;
insert into stats$end_roll select * from v$rollstat@CASE_STAT;
insert into stats$end_file select * from stats$file_view; /*no link*/
insert into stats$end_dc select * from v$rowcache@CASE_STAT;
```

接下来的程序清单显示了 utlestat.sql**的表创建部分，假设原始表已经在** CC1 Command Center**数据库的**CC**表空间中创建。**

在此例中，表将也被存储在 CC1 Command Center**数据库的**CC**数据表空间中。每一个** create table**命令都将有** tablespace CC **子句，每一个远程表的查询有一个** @case_stat**子句来使用数据库链接**CASE_STAT**。**

```
create table stats$stats
TABLESPACE CC
as select  e.value-b.value change , n.name
   from v$statname n ,  stats$begin_stats b , stats$end_stats e
   where n.statistic# = b.statistic# and n.statistic# = e.statistic#;

create table stats$latches
TABLESPACE CC
as select e.gets-b.gets gets,
   e.misses-b.misses misses,
   e.sleeps-b.sleeps sleeps,
   e.immediate_gets-b.immediate_gets immed_gets,
   e.immediate_misses-b.immediate_misses immed_miss,
   n.name
   from v$latchname n ,  stats$begin_latch b , stats$end_latch e
   where n.latch# = b.latch# and n.latch# = e.latch#;

create table stats$event
TABLESPACE CC
as select  e.total_waits-b.total_waits event_count,
          e.time_waited-b.time_waited time_waited,
          e.event
    from  stats$begin_event b , stats$end_event e
    where b.event = e.event
  union all
  select  e.total_waits event_count,
          e.time_waited time_waited,
          e.event
    from  stats$end_event e
    where e.event not in (select b.event from stats$begin_event b);

create table stats$bck_event tablespace CC_ as
  select  e.total_waits-b.total_waits event_count,
          e.time_waited-b.time_waited time_waited,
          e.event
    from  stats$begin_bck_event b , stats$end_bck_event e
    where b.event = e.event
  union all
  select  e.total_waits event_count,
          e.time_waited time_waited,
          e.event
    from  stats$end_bck_event e
```

```
    where e.event not in (select b.event from stats$begin_bck_event b);

update stats$event e
  set (event_count, time_waited) =
    (select e.event_count - b.event_count,
            e.time_waited - b.time_waited
       from stats$bck_event b
          where e.event = b.event)
   where e.event in (select b.event from stats$bck_event b);

create table stats$waitstat as
select  e.class,
        e.count - b.count count,
        e.time - b.time time
  from stats$begin_waitstat b, stats$end_waitstat e
   where e.class = b.class;

create table stats$roll
TABLESPACE CC
as select  e.usn undo_segment,
        e.gets-b.gets trans_tbl_gets,
   e.waits-b.waits trans_tbl_waits,
   e.writes-b.writes undo_bytes_written,
   e.rssize segment_size_bytes,
        e.xacts-b.xacts xacts,
   e.shrinks-b.shrinks shrinks,
        e.wraps-b.wraps wraps
   from stats$begin_roll b, stats$end_roll e
        where e.usn = b.usn;

create table stats$files
TABLESPACE CC
as select b.ts table_space,
       b.name file_name,
       e.pyr-b.pyr phys_reads,
       e.pbr-b.pbr phys_blks_rd,
       e.prt-b.prt phys_rd_time,
       e.pyw-b.pyw phys_writes,
       e.pbw-b.pbw phys_blks_wr,
       e.pwt-b.pwt phys_wrt_tim,
       e.megabytes_size
  from stats$begin_file b, stats$end_file e
       where b.name=e.name;

create table stats$dc
TABLESPACE CC
as select b.parameter name,
       e.gets-b.gets get_reqs,
       e.getmisses-b.getmisses get_miss,
       e.scans-b.scans scan_reqs,
       e.scanmisses-b.scanmisses scan_miss,
       e.modifications-b.modifications mod_reqs,
       e.count count,
       e.usage cur_usage
  from stats$begin_dc b, stats$end_dc e
       where b.cache#=e.cache#
        and  nvl(b.subordinate#,-1) = nvl(e.subordinate#,-1);
```

```
create table stats$lib
TABLESPACE CC
as select e.namespace,
       e.gets-b.gets gets,
       e.gethits-b.gethits gethits,
       e.pins-b.pins pins,
       e.pinhits-b.pinhits pinhits,
       e.reloads - b.reloads reloads,
       e.invalidations - b.invalidations invalidations
  from stats$begin_lib b, stats$end_lib e
       where b.namespace = e.namespace;
```

这些统计表的前、后“快照”将提供关于在Oracle中能够监控的所有内存对象的信息。这些信息包括字典高速缓存、命中率和基于文件的I/O统计。同时还包括回滚段的使用和锁存的使用信息。

6.5.2 统计报表解释

utlbstat/utlestat脚本文件创建一个名为report.txt的报表，它列出了数据库所有部分的信息。Oracle8的report.txt包括表6-2列出的13个部分。

表6-2 report.txt部分

报表部分
库缓存统计
全面统计
“脏”缓冲区写队列的平均长度
系统范围的等待事件
后台进程的系统范围等待事件
锁存器统计
锁存器的非等待获取
缓冲区忙等待统计
回滚段
init.ora值
字典缓存
按表空间汇总的文件I/O
文件I/O

下面按这些部分在报表中的出现顺序进行描述。

1. 库缓存统计

库缓存(Library Cache, LC)含有SQL和PL/SQL共享区域。报表的这一部分的统计有助于判定是否由于分配给LC的内存不够充足而使SQL共享语句被重新分析。图6-9示出了样例数据。

LIBRARY	GETS	GETHITRATI	PINS	PINHITRATI	RELOADS	INVALIDATI
BODY	0	1	0	1	0	0
SQL AREA	89	.843	282	.879	5	0
TABLE/PROCED	106	.83	96	.802	1	0
TRIGGER	0	1	0	1	0	0

Note: Sum of Pins column = 378. Sum of Reloads column = 6.

图6-9 库缓存统计样例

Pins列表示一个条目被执行的次数， Reloads列是遗漏的次数。 Reloads与Pins的比例表明执行重新分析的百分比。对于这个样本数据，该比例为 6/378(或1.6%)。这意味着在执行前需要再次分析语句的次数占总次数的 1.6%。理想的比例值是 0，如果比例大于 1%(如本例)，Oracle建议增加SQL共享池的内存。通过init.ora 参数SHARED_POOL_SIZE来增加SQL共享池的内存。

如果应用程序大部由动态 SQL构成，每当执行它们时就对 SQL命令重新进行分析。即使以前执行过这些SQL命令，也要重新分析。在这样的应用程序中，增加SHARED_POOL_SIZE值不会对LC统计数据产生什么影响。由于连续重分析语句， Reloads与Pins的比率会较大。

2.全面统计

报表的全面统计 (Overall Statistics)部分显示许多系统统计的总变化，同时给出每个事务(per transcation)和每个登录(per logon)值(但是，由于运行utlestat时需要登录到数据库，所以每个登录数值总是比实际发生的登录次数要多一次)。只显示非零值变化。报表的这一部分对判定总命中率和指出数据库设置中可能有的问题很有帮助。

若要确定命中率，需使用公式： (逻辑读-物理读)/逻辑读。逻辑读是“一致性获取”(consistent gets)和“数据块获取”(db block gets)之和，物理读在报表中作为“ physical reads”显示。根据图6-10所显示的统计，命中率是93.4%((1358+214)-103)/(1358 + 214))。

Statistic	Total	Per Transact	Per Logon
consistent gets	1358	1358	226.33
db block gets	214	214	35.67
physical reads	103	103	17.17

图6-l0 命中率统计样例

分析其他统计数字时，最佳的结果是其中的大多数值应为 0或很低。这包括循环调用 (它们也应该显示在报表的字典缓存部分)、表扫描(长表)和队列输出时间；若这些统计值较高，说明所使用的数据库及应用系统需要修改以改善性能。

3.“脏”缓冲区写队列的平均长度

报表的这部分查询重新访问全面统计部分使用的统计表。它比较两个条目，计算“脏队列长度总和”(summed dirty queue length)记录中变化与“写请求”(write request)记录中变化的命中率。如果平均长度 (由查询返回的值)大于init.ora参数db_block_buffers(见“init.ora值”部分)的0.25倍时，可能存在两种原因：一是数据库 I/O不均匀地分布在数据文件中；二是参数db_file_simultaneous_writes设置得过低。无论哪一种情况，数据库写操作的效率都极低。

4. 系统范围的等待事件

报表的这部分列出了一系列系统事件的计数、汇总时间和平均时间，报表中没有文件用来建立所示值的范围。由于这部分 report.ext的查询计算每个事件所需的时间，初始化参数——TIMED_STATISTICS——应该在init.ora中设为TRUE以获得非零值。

5. 后台进程的系统范围等待事件

报表的这一部分列出了事件、等待总数和等待后台进程的总时间。由于 report.ext这一部

分的查询计算每个事件所需的时间，所以初始化参数——TIMED_STATISTICS——应该在init.ora中设置为TRUE以获得非零值。经历长等待的后台进程可能使用 Oracle8的可用选项。DBWR进程可有多个I/O从进程，其数量由init.ora文件中的DBWR_IO_SLAVES参数值决定，LGWR与ARCH进程也可以有I/O从进程(可以通过init.ora参数DBWR_PROCESSES启动多个DBWR进程)。

6. 锁存器统计

可以利用报表的这一部分确定适合数据库的重做日志分配锁存器及重做日志拷贝锁存器的数量。

重做日志拷贝锁存器用在多CPU服务器上来分配进程，这些进程通常由重做日志分配锁存器来完成。在这样的设置中，拷贝锁存器用于把进程的重做信息拷贝到 SGA中的重做日志缓冲区。如果不使用拷贝锁存器，那么分配锁存器必须管理锁存器分配和执行拷贝，从而降低了事务日志的处理速度。

如果重做分配的Misses值超过Gets列的10%，并且服务器上有多个处理器，就要考虑增加重做日志拷贝锁存器。要做到这一点，请减少 init.ora参数LOG_SMALL_ENTRY_SIZE并增大LOG_SIMULTANEOUS_COPIES和LOG_ENTRY_PREBUILD_ THRESHOLD值。这些参数决定锁存器的数量和用分配锁存器拷贝的重做条目的最大尺寸 (其他全部继续传给拷贝锁存器)。修改这些值后，重新生成统计报表，以查看是否要进一步修改锁存器。

7. 锁存器的非等待获取

报表的这一部分计算非等待锁存器请求立即得到满足的百分比。如图 6-11所示的样例数据，所有非等待命中率为100%。

LATCH_NAME	NOWAIT_GETS	NOWAIT_MISSES	NOWAIT_HIT_RATIO
cache buffers chai	60643	0	1
cache buffers lru	1021	0	1
library cache	96	0	1
library cache pin	14	0	1
row cache objects	11	0	1

图6-11 非等待锁存器获取统计样例

8. 缓冲区忙等待

报表的这一部分识别发生冲突的块的类型。如果回滚段很少，就能看到“ undo segment headers ”的高值(>2000)。“ data block ”等待的高值(>10000)表明应该把DBWR的I/O从进程添加到附加的DBWR进程中。

9. 回滚段

报表的这一部分显示关于回滚段使用的统计数字，应该考虑对报表的这一部分进行若干修改以加强它的作用：

- 取消表示活动事务数量的 XACTS列，由于这个统计不是累积统计，所以它对确定开始值和结束值的差别没有多少用处。
- 添加Extends列，此列是累积列且能够添加到 utlbstat查询的回滚段查询上。 Extends列列出了回滚段扩展次数。

这两列跟踪的操作在第7章中叙述。

通过查询V$ROLLNAME表，就可以用Undo_Segment列的值来确定回滚段的名称。

```
select Name from V$ROLLNAME
where USN = &UNDO_SEGMENT;
```

如报表的Trans_Tbl_Waits列中所示，回滚段等待表明数据库中可能需要更多的回滚段，Shrinks和Wraps的非零值表示回滚段动态扩展或收缩(回到其最佳设置)。这些活动表示需要重新设计回滚段以反映在数据库中执行的事务种类。

l0.init.ora值

这一部分显示init.ora参数的非缺省设置。可能要在这个查询中去掉where子句，以便列出全部参数及其值。

11.字典缓存

report.txt的字典缓存(Dictionary Cache，DC)部分反映SQL共享池中字典缓存的设置(Count)和当前使用情况(Usage)。此报表只显示那些在报告的时间间隔内有非零值的参数。遗漏率应比较低(通常小于10%)。图6-12示出了report.txt的字典缓存部分。

NAME	GET_REQS	GET_MISS	SCAN_REQ	SCAN_MIS	MOD_REQS	COUNT	CUR_USAG
dc_free_extents	246	0	0	0	0	97	82
dc_segments	2	1	0	0	0	128	126
dc_rollback_seg	36	0	0	0	0	17	7
dc_users	46	0	0	0	0	14	13
dc_user_grants	32	0	0	0	0	43	10
dc_objects	60	10	0	0	0	221	218
dc_tables	116	5	0	0	0	195	190
dc_columns	446	27	53	5	0	1880	1871
dc_table_grants	54	24	0	0	0	1626	764
dc_indexes	17	1	37	2	0	261	140
dc_constraint_d	1	0	9	0	0	396	13
dc_synonyms	3	0	0	0	0	18	17
dc_usernames	15	0	0	0	0	20	15
dc_sequences	5	0	0	0	0	7	1

图6-12 字典缓存统计样例

12. 按表空间汇总的文件I/O

除了在表空间级汇总而不是在文件级汇总外，报表的这一部分提供的信息与文件I/O部分相同，请参见本章6.5.3节“统计报表的扩展”。

13. 文件I/O

这一部分记录了对数据库中数据文件的物理I/O和逻辑I/O，见本章6.5.3节“统计报表的扩展”。

6.5.3 统计报表的扩展

利用生成report.txt文件的查询，可以产生在特定方式下运行的非常有用的报表。这些查询将针对数据库中的当前统计值运行，而不是针对由utlbstat和utlestat脚本文件创建的表运行。这些查询所产生的统计值反映最近一次启动数据库以来数据库的所有活动。

1. 文件I/O

下列SQL*Plus脚本文件按磁盘生成全部数据库文件的一个清单并计算出每一个磁

盘的I/O活动的总和。它的输出有助于说明如何在可用设备之间很好地分布当前的文件I/O。

注意 这两个查询都假设驱动器名称为5个字符长(例如/db01)，如果驱动器名称不是5个字符长，就要修改查询中的SUBSTR函数和Drive列的格式化命令。

```
clear columns
clear breaks
column Drive format A5
column File_Name format A30
column Blocks_Read format 99999999
column Blocks_Written format 99999999
column Total_IOs format 99999999
set linesize 80 pagesize 60 newpage 0 feedback off
ttitle skip center "Database File I/O Information" skip 2
break on report
compute sum of Blocks_Read on report
compute sum of Blocks_Written on report
compute sum of Total_IOs on report

select substr(DF.Name,1,5) Drive,
       SUM(FS.Phyblkrd+FS.Phyblkwrt) Total_IOs,
       SUM(FS.Phyblkrd) Blocks_Read,
       SUM(FS.Phyblkwrt) Blocks_Written
  from V$FILESTAT FS, V$DATAFILE DF
 where DF.File#=FS.File#
 group by substr(DF.Name,1,5)
 order by Total_IOs desc;
```

样本输出如下所示：

```
DRIVE TOTAL_IOS BLOCKS_READ BLOCKS_WRITTEN
----- --------- ----------- --------------
/db03     57217       56820            397
/db01     39940       27712           6228
/db04     15759       14728           1031
/db02      1898          10           1888
      --------- ----------- --------------
sum      108814       99270           9544
```

第二个文件I/O查询显示了属于每个数据文件的I/O：

```
clear breaks
clear computes
break on Drive skip 1 on report
compute sum of Blocks_Read on Drive
compute sum of Blocks_Written on Drive
compute sum of Total_IOs on Drive
compute sum of Blocks_Read on Report
compute sum of Blocks_Written on Report
compute sum of Total_IOs on Report
ttitle skip center "Database File I/O by Drive" skip 2

select substr(DF.Name,1,5) Drive,
       DF.Name File_Name,
       FS.Phyblkrd+FS.Phyblkwrt Total_IOs,
       FS.Phyblkrd Blocks_Read,
       FS.Phyblkwrt Blocks_Written
  from V$FILESTAT FS, V$DATAFILE DF
 where DF.File#=FS.File#
 order by Drive, File_Name desc;
```

这个查询的样本输出如下所示：

```
DRIVE FILE_NAME                          TOTAL_IOS BLOCKS_READ BLOCKS_WRITTEN
----- ------------------------------ ---------- ----------- --------------
/db01 /db01/oracle/CC1/sys.dbf            29551       27708           1843
      /db01/oracle/CC1/temp.dbf            4389           4           4385
***** ------------------------------ ---------- ----------- --------------
sum                                       33940       27712           6228

/db02 /db02/oracle/CC1/rbs01.dbf           1134           3           1131
      /db02/oracle/CC1/rbs02.dbf            349                        349
      /db02/oracle/CC1/rbs03.dbf            415           7            408
***** ------------------------------ ---------- ----------- --------------
sum                                        1898          10           1888

/db03 /db03/oracle/CC1/cc.dbf             57217       56820            397
***** ------------------------------ ---------- ----------- --------------
sum                                       57217       56820            397

/db04 /db04/oracle/CC1/ccindx.dbf         15759       14728           1031
      /db04/oracle/CC1/tests01.dbf
***** ------------------------------ ---------- ----------- --------------
sum                                       15759       14728           1031

***** ------------------------------ ---------- ----------- --------------
sum                                      108814       99270           9544
```

上面列表中的数据示出了查询输出的格式。报表的第一部分是各个驱动器中数据文件的数据库I/O比较。它表示名为/db03的设备是数据库I/O最频繁的设备。第二部分表示设备/db03上的I/O属于一个文件，因为CC1数据库的驱动器上没有其他文件存在。它还表明对/db03上文件的访问以读为主。

第二个最重要的设备是/db01，它有两个数据库文件。这个盘上的大多数活动都是访问SYSTEM表空间文件，TEMP表空间需要的I/O很少。由于不是在一个特定的间隔而是在数据库打开的全部时间查看I/O，所以SYSTEM表空间在查询时的读取频率很高。由于这个原因，与数据库启动和SGA初始化有关所有的I/O都在这里显示。

这个报表对于检测文件I/O可能发生的冲突非常有效。根据这个报表和设备的I/O能力，可以正确地分配数据库文件以达到最小的I/O冲突和最大的流量。关于最小化I/O冲突的更多信息，请参见第4章。

2. 最大扩展的段

本章的报警报表使用了建立在系统级的控制限制准则(例如，每段10个盘区)。然而，通过maxextents存储参数，就可以根据为此段特别定义的限度来评估当前段盘区的使用。

下面的SQL*Plus报表查询远程数据库以检查每一个在其指定的最大扩展倍数内的段。它通过数据库链接访问那些数据库，并且链接名被用作输出文件名的一部分。倍数值应该总大于1——当实际盘区数与最大盘区数进行比较时应该乘以这个值。为了确定哪些段达到了它们的最大扩展的80%，应把这个倍数值设为1.2。

这个查询检查4个不同类型的段：簇、表、索引和回滚段。对于每一种类型的段，它都判断当前的盘区数是否已达到此段的最大盘区数(例如通过maxextents参数设置)，“倍数”用于监控已接近最大扩展的段。如果段达到其最大扩展，就返回其拥有者、名字及当前空间的使用信息。

注意 从Oracle7.3开始，可以设置maxextents值为unlimited。当设置一个段的

maxextents值为unlimited时，Oracle指定此段的maxextents值为2 147 483 645。因此，一个设置maxextents值为unlimited的段理论上能达到其最大扩展，尽 管此值非常大。

```
rem
rem  file:  over_extended.sql
rem  parameters:  database link name (instance name), multiplier
rem
rem  The "multiplier" value should always be greater than 1.
rem  Example:  To see segments that are within 20 percent of
rem  their maximum extension, set the multiplier to 1.2.
rem
rem  Example call:
rem  @over_extended CASE 1.2
rem

select
   Owner,                  /*owner of segment*/
   Segment_Name,           /*name of segment*/
   Segment_Type,           /*type of segment*/
   Extents,                /*number of extents already acquired*/
   Blocks                  /*number of blocks already acquired*/
from DBA_SEGMENTS@&&1 s
where                      /*for cluster segments*/
(S.Segment_Type = 'CLUSTER' and exists
(select 'x' from DBA_CLUSTERS@&&1 c
where C.Owner = S.Owner
and C.Cluster_Name = S.Segment_Name
and C.Max_Extents <= S.Extents*&&2))
or                         /*for table segments*/
(s.segment_type = 'TABLE' and exists
(select 'x' from DBA_TABLES@&&1 t
where T.Owner = S.Owner
and T.Table_Name = S.Segment_Name
and T.Max_Extents <= S.Extents*&&2))
or                         /*for index segments*/
(S.Segment_Type = 'INDEX' and exists
(select 'x' from DBA_INDEXES@&&1 i
where I.Owner = S.Owner
and I.Index_Name = S.Segment_Name
and I.Max_Extents <= S.Extents*&&2))
or                         /*for rollback segments*/
(S.Segment_Type = 'ROLLBACK' and exists
(select 'x' from DBA_ROLLBACK_SEGS@&&1 r
where R.Owner = S.Owner
and R.Segment_Name = S.Segment_Name
and R.Max_Extents <= S.Extents*&&2))
order by 1,2

spool &&1._over_extended.lst
/
spool off
undefine 1
undefine 2
```

这个报表的输出文件在标题中将含有数据库链接名。输出文件名用点（.)作为SQL*Plus的连接字符。

3. 不能扩展到可用自由空间的段

除了监控盘区范围附近的段外，应定期查询数据库以确定是否还有其他段不能扩展到表空间的可用空间中。这个查询不必是 Command Center数据库的一部分。它是一个报警表，由查询返回的记录应立即被寻址。

这个报表有两个独立查询。如下面所示，第一个查询确定表空间中是否有足够的空间供段的下一个盘区使用：

```
select Owner, Segment_Name, Segment_Type
  from DBA_SEGMENTS
 where Next_Extent >
(select SUM(Bytes) from DBA_FREE_SPACE
  where Tablespace_Name = DBA_SEGMENTS.Tablespace_Name);
```

上面的查询可能报告没有问题的段。例如，表空间的数据文件可能被设置成autoextend(自动扩展)；在这种情况下，表空间的可用自由空间受最大数据文件尺寸的限制。

如果由上面的查询返回一个表名，则表示即使表空间中的全部空间都合并在一起，表空间中也没有足够的空间来存储这个表的下一个盘区。这是一种报警状态；可以创建一个单独的查询，在表达到报警条件之前将其识别出来。只要段的下一个盘区不符合表空间中的最大单个自由盘区，以下的查询就返回这个段的名字：

```
select Owner, Segment_Name, Segment_Type
  from DBA_SEGMENTS
 where Next_Extent>
(select MAX(Bytes) from DBA_FREE_SPACE
  where Tablespace_Name = DBA_SEGMENTS.Tablespace_Name);
```

合并自由空间可以解决一些空间问题。不过，如果表的自由空间在数据文件中间分布或被数据段分开，就不能增加最大的可用自由盘区尺寸。

如果不能支持常用段的另一个盘区，可能要给表空间添加空间或改变段的存储参数。

4. 对象中的空间分配率

如第4章和第5章所述，通过正确设置数据库对象的大小就可以简化数据库的空间管理活动。不过，即使可能出现问题，正确设置大小的对象也不会由 Command Center数据库及类似的报警报表来报告。

例如，可以创建一个 SALES表，估计这个表每年需要 1GB。如果为这个表分配一个 1GB的盘区，它许多月内都不需要获取一个新盘区，因此它将不出现在 Command Center报表中。如果表没有获得任何新的盘区，表空间中的自由盘区就可能不发生变化，因而空间监测器(watcher)报表将不列出这个表空间。然而，表中的空间使用也可能会出现问题。

如果表中的空间使用率比预期的要大，表就会比预期扩展得快。对于一个大型事务表，事先预期的扩展可能对可用空间产生重大影响。如果使用前一节提供的脚本文件，应能预先发现即将遇到空间问题的段扩展。可以使用本节提供的查询来确定哪些段可能要扩展。

下面的查询使用数据字典视图中的统计列，因此它只对已分析过的对象有效。此外，查询的输出值直接与分析对象的频率和执行的分析类型相关。最好的结果是，在执行查询之前使用一个全compute statistics命令进行分析。该查询将报告数据块使用占其当前分配使用的95%以上的任一对象。

如下面清单所示，union查询的第一部分为数据库的表选择数据。 column命令对数据进行

格式化，以使它适合于宽度为80列的屏幕。

```
column Owner format A12
column Table_Name format A20
column Empty format 99999
column Pctusd format 999.99

select T.Owner,
       T.Table_Name,
       S.Segment_Type,
       T.Blocks Used,
       T.Empty_Blocks Empty,
       S.Blocks Allocated,
       T.Blocks/S.Blocks Pctusd
  from DBA_TABLES T, DBA_SEGMENTS S
 where T.Owner = S.Owner
   and T.Table_Name = S.Segment_Name
   and S.Segment_Type = 'TABLE'
   and T.Blocks/S.Blocks > 0.95
 order by 7  desc
```

上面查询的样本数据如下所示：

OWNER	TABLE_NAME	SEGMENT	USED	EMPTY	ALLOCATED	PCTUSD
APPOWN	REGION	TABLE	46	0	47	.98
APPOWN	MANUFACTURER	TABLE	210	4	215	.98
APPOWN	VOUCHER_HEADER	TABLE	251	5	257	.98
APPOWN	PO_HEADER	TABLE	459	10	470	.98
APPOWN	VOUCHER_LINE	TABLE	869	5	890	.98
APPOWN	PO_LINE	TABLE	491	0	505	.97

如输出中所示，数据库中的一些表使用了分配给它们的数据库块的95%以上。如果具有最高百分比空间使用的表是活动的事务表，它们就可能扩展。应对表进行判断，以确定这些表是否会扩展。例如，除非你频繁地添加新区域，否则输出清单中的第一个表REGION可能不会扩展。不过，VOUCHER_HEADER和VOUCHER_LINE表是事务表且可能要扩展。

如下面清单所示，这个查询报告出接近其当前空间分配的表和索引：

```
column Owner format A12
column Table_Name format A20
column Empty format 99999
column Pctusd format 999.99

select T.Owner,
       T.Table_Name,
       S.Segment_Type,
       T.Blocks Used,
       T.Empty_Blocks Empty,
       S.Blocks Allocated,
       T.Blocks/S.Blocks Pctusd
  from DBA_TABLES T, DBA_SEGMENTS S
 where T.Owner = S.Owner
   and T.Table_Name = S.Segment_Name
   and S.Segment_Type = 'TABLE'
   and T.Blocks/S.Blocks > 0.95
union all
select I.Owner,
       I.Index_Name,
       S.Segment_Type,
```

```
        I.Leaf_Blocks Used,
        S.Blocks-1-I.Leaf_Blocks Empty,
        S.Blocks Allocated,
        I.Leaf_Blocks/S.Blocks Pctusd
  from DBA_INDEXES I, DBA_SEGMENTS S
 where I.Owner = S.Owner
   and I.Index_Name = S.Segment_Name
   and S.Segment_Type = 'INDEX'
   and I.Leaf_Blocks/S.Blocks > 0.95
order by 7 desc, 2, 1
```

上面的查询将在同一报表中显示表和索引的数据。数据将按所使用的分配空间的百分比排序，首先列出百分比最高的。

可以通过建立历史表并存储过去的使用百分比来扩展这个报表。但是，此历史表可能不是很有用。例如，当段获得一个新盘区时，已使用的百分比将减少，因此历史数据可能没有时间上的可比性。无论是作为对全部段的报警还是跟踪特定段的空间利用，这个报表都非常有用。

6.6 良好管理的数据库

任何系统的有效管理都需要策略规划、质量控制和解决系统中失控部分的操作。本章描述的数据库管理系统为所有数据库监控提供了一个基础。各个数据库可能需要执行附加的监控，或者可能有特定的阈值。可以把这些轻松地加在这里所示的例子上。

建立Command Center数据库可以使系统中的其他数据库在不影响测量效果的情况下被监控。也允许将新数据库简单地增加到监控系统上和对所有统计进行趋势分析。 Command Center数据库应与现有的操作系统监控程序紧密相连，以便与报警消息的分发和解决相配合。

与任何系统一样，必须规划 Command Center数据库。本章中的例子设计成处理数据库中最常被监控的对象，每一个都要设定阈值。在完全定义好要监控什么及其阈值之前，不要创建CC1数据库。一旦完成这些工作，就创建数据库 —— 在递归的大例子中，用监控数据库来监控其自身。