第3章　数据库逻辑设计

数据库的逻辑配置不仅对数据库的性能有重要影响，而且对简化管理有明显效果。本章介绍对Oracle数据库的表空间进行恰当设计的准则。

数据库逻辑对象的高效分布首先由Oracle的Cary Millsap定型，Cary Millsap命名了最终结构——Optimal Flexible Architecture (OFA，优化软结构)。这里所描述的逻辑数据库设计将对OFA进行定义并扩充，因为它与逻辑数据库设计相关。使用这种结构进行布局规划会大大简化数据库管理操作，并在计划和协调物理布局时使数据库管理人员有更大的选择自由。根据选择的安装选项，标准的OFA表空间布局会在Oracle软件安装时自动创建。本章将对该布局和一些有用的标准布局替代方案进行说明。

3.1　最终产品

这里描述的数据库设计目标用于数据库配置，以便数据库对象按对象类型及操作类型进行区分。这种配置会大大降低数据库管理所需的工作量，同时也能够减少所需的监控。从而使一个区域的问题不会影响数据库的其他区域。

规划数据库的物理布局时，用这里描述的方式分布数据库对象还将使数据库管理人员具有更大的灵活性。第4章所讲的练习，在尽可能分布化的逻辑设计中是最易于实现的。

为了在数据库中高效地分布对象，必须首先建立一个分类系统。数据库中的逻辑对象必须根据它们的使用方式及其物理结构对数据库的影响来进行分类。这种分类过程包括将索引与表分开，将低活动性表与高活动性表分开。尽管对象的活动量只能在产品使用时确定，但频繁使用的数据表核心集通常可以分离出去。

3.2　优化软结构

下面几节，可以看到优化软结构(OFA)定义的对象类别，此后，将是建议的OFA扩展。

3.2.1　起点：SYSTEM表空间

虽然并不可取，但是有可能将所有数据库对象存储在一个表空间中；这类似于把所有文件存储在根目录下。相当于根目录的SYSTEM表空间存储了数据字典表(data dictionary tables，属于SYS)。SYSTEM表空间也是SYSTEM回滚段的存储位置，并且在数据库创建时，SYSTEM表空间暂用于存储另一个回滚段(随后，这个回滚段会被休眠或撤消)。

没有理由把除数据字典表和SYSTEM回滚段外的其他东西存储在SYSTEM表空间中。在SYSTEM表空间中存储其他段类型会增加出现空间管理问题的可能性，这些问题可能要求重建表空间。由于唯一能重建SYSTEM表空间的方法是重新创建数据库，所以能移出SYSTEM表空间的任何数据都应该移出。

数据字典表存储了有关数据库中全部对象的所有信息。数据字典段(Data dictionary segment)——数据字典表的物理存储区——存储在SYSTEM表空间中，并且如果不对数据库

应用程序做大的结构改动，就几乎不变。数据字典段在数据库生成时创建并且相当小。所创建的过程对象(例如触发器及过程)越多，所使用的抽象数据类型和面向对象的特征越多，数据字典段就越大。触发器等过程对象把PL/SQL代码存储在数据库中，并且将其定义存储在数据字典表中。

在缺省情况下，在数据库中创建的任何新用户都有一个缺省的SYSTEM表空间。为避免用户在SYSTEM表空间中创建对象，SYSTEM上的任何定额(定额允许在系统中生成对象)都必须被取消。

```
alter user USRE quota 0 on SYSTEM;
```

当创建一个用户(使用create user命令)时，可以指定一个缺省表空间；

```
create user USERNAME identified by PASSWORD
default tablespace TABLESPACE_NAME;
```

一旦创建了用户，就可以使用

```
alter user USERNAME default tablespace TABLESPACE_NAME;
```

命令重新指定用户的缺省表空间。指定用户和开发者的缺省表空间后，无需tablespace子句，就可以把创建的对象存储在SYSTEM表空间之外。

注意 如果一个用户被授予UNLIMITED TABLESPACE系统权限或RESOURCE角色，则这个授权将覆盖该用户的任何定额设置。

下面几节将讨论各种类型的数据库对象、它们的用途以及它们为什么与数据库的其他部分分开存储。根据数据库的使用方式，最终结果将是一个标准的数据库配置，这种配置具有最多19个标准的表空间类型。通过开发一个一致性逻辑结构，就可以简化对数据库一致性物理结构的开发。物理数据库结构越一致，数据库的管理操作就越简单。

3.2.2 分离应用程序数据段：DATA

数据段(data segment)是一些物理区域，用于存储与表和簇相关的数据。数据段经常被数据库访问，也经常执行数据操作事务。对数据段访问要求的管理是产品数据库的主要目标。

一个典型的DATA表空间包含了与应用程序相关的所有表。对这些表的大量I/O操作使它们更应该分离出来而拥有自己的表空间。如果将应用程序表与DATA表空间分开，便可以从数据库的其他数据文件中区分出这个表空间的数据文件。这种跨磁盘的数据文件分离可能会改善系统性能(可以减少I/O资源的冲突)并且简化文件管理。

DATA表空间中的段有可能成为碎片。如果数据段成为碎片，创建段时就不能准确确定段大小。管理碎片还导致需要把DATA从SYSTEM中分离出来，这个问题将在第4章做详细描述。从数据字典段中分离数据段更有利于隔离和解决碎片问题。

在DATA中，可以有多种表类型。与大的活动表相比，小的静态表有不同的存储特性和管理要求。有关管理不同表类型的详细情况，请参见本章后面的3.3.1节。

3.2.3 分离应用程序索引段：INDEXES

与数据表相关的索引，同样存在I/O和增长/碎片等问题，这些问题促使从SYSTEM表空间移出数据段。索引段不应与其相关表存储在同一个表空间中，因为它们在数据管理和查询时

存在许多I/O冲突。

不正确的大小设置或不可预测的表增长是索引段生成碎片的原因之一。将应用程序索引从表空间中分离出来，可以大大减少消除DATA或INDEX表空间中碎片所需的管理工作。

从数据表中分离出已有索引也可以通过使用 alter index命令中的rebuild选项来实现。如果已在与相关表相同的表空间中创建了一个索引，就可以用一个命令移动它。在下面的例子中，EMPLOYEE$DEPT_NO索引被移到INDEXES表空间，并且为它分配新的 storage(存储)值：

```
alter index EMPLOYEE$DEPT_NO rebuild
tablespace INDEXES
storage (initial 2M next 2M pctincrease 0);
```

如果使用分区，应把分区索引与表分区分开。例如，可以用十个分区创建一个表。如果创建局部分区索引，则将有十个分区索引 —— 每个表分区一个索引。应把这些分区索引与它们所索引的表分区分开存储。如果在一个分区表上创建一个全局索引，这个索引中的条目就可能与任一个表分区相关。如果使用一个全局索引，就应把这个索引与该表的全部表分区分开。

例如，如果SALES表被分区，就可以在 SALES表的分区上创建局部索引。每一个这样的局部索引分区，都应与它们各自的表分区分开存储。如果创建一个跨越整个 SALES表的全局索引，这个索引就应与SALES表的全部分区分开存储。

当创建一个表时，数据库将为指定的所有 PRIMARY KEY或UNIQUE约束条件创建一个索引(除非首先创建了没有约束条件的表，然后再创建索引，再创建约束条件)。如果创建表时没有规定这些约束条件的tablespace和storage参数，数据库将自动在缺省表空间中使用这个表空间的缺省存储参数创建这些索引。为避免出现这个问题，要用 create table命令的using index子句把约束条件索引与表分开。例如：

```
create table JOB
(Job_Code NUMBER,
Description VARCHAR2(35)
constraint JOB_PK primary key (Job_Code )
using index tablespace INDEXES
storage (initial 2M next 2M pctincrease 0) )
tablespace DATA
storage (initial 5M next 5M pctincrease 0);
```

JOB表将在DATA表空间中创建，但是其主键索引 JOB_PK将在INDEXES表空间中创建。注意，JOB表及其主键索引有不同的storage和tablespace子句。

3.2.4 分离工具段：TOOLS

尽管前面几节发出不要在 SYSTEM表空间存储数据段的警告，但还是有许多工具存储在SYSTEM表空间。这并不是因为要特别调用它们的对象来存储在 SYSTEM表空间，而是将它们存储在SYSTEM数据库帐号下，这个帐号通常把 SYSTEM表空间作为其存储对象的缺省区。要避免这种情况，可以将 SYSTEM帐号的缺省表空间改变为 TOOLS表空间，并且取消它在

SYSTEM表空间中的定额。如果SYSTEM被授予UNLIMITED TABLESPACE，也可以取消这个权限以避免有非数据字典对象存储在SYSTEM表空间中。不过，如果从SYSTEM中取消UNLIMITED TABLESPACE，则务必给表空间授予定额，因为SYSTEM可能需要这一定额来创建回滚段之类的对象。

许多Oracle和第三方工具都创建属于SYSTEM的表。如果在数据库中已经创建了这些表，则可以通过导出数据库、撤消工具的表、取消SYSTEM表空间帐号中的定额、仅授予SYSTEM用户TOOLS表空间中的定额、导入表等一系列操作来移动它们的对象。

下面的程序段是这个进程的一部分。第一步从SYSTEM用户取消定额，第二步把TOOLS表空间的定额授予SYSTEM用户。

```
alter user SYSTEM quota 0 on SYSTEM;
alter user SYSTEM quota 50M on TOOLS;
```

3.2.5 分离回滚段：RBS

在数据库中，回滚段维护语句级和事务级读的一致性。为了创建非SYSTEM表空间，必须首先在SYSTEM表空间中创建另一个回滚段。若从数据字典中分离回滚段(会招致数据库中事务的I/O)，就需要创建一个只包含回滚段的回滚段表空间。用这种方式将它们进行分离，也同样可以大大简化对它们的管理（见第7章）。

一旦创建了RBS表空间，并且在此表空间中激活一个回滚段，就可以撤消SYSTEM表空间中的第二个回滚段。在RBS表空间出现问题时你会发现，保持SYSTEM中的第二个回滚段为SYSTEM非激活状态（仍然可用）是有作用的。

回滚段动态扩展到大型事务的大小，然后再将收缩为指定的最佳大小(见第7章)。回滚段的I/O通常与DATA和INDEXES表空间的I/O同时发生。将回滚段与数据段分隔开有助于避免发生I/O冲突，从而简化对它们的管理。

3.2.6 分离临时段：TEMP

临时段(temporary segment)是数据库中动态创建的对象，用以存储大型排序操作（如select distinct、union、create index等操作）中的数据。由于它们的动态性，所以临时段不应与其他类型的段一起存储。在第4章中，将讨论临时段的正确结构。

对于Oracle7.3，可以通过create tablespace和alter tablespace命令把一个表空间指定为临时表空间。如果把一个表空间指定为临时表空间，将不能在这个表空间内创建表和索引之类的永久性段。此外，当相关的命令结束时，不会撤消这个表空间中的临时段，只是空间管理量有所减少。

除非把一个表空间指定为临时表空间，否则一旦临时段支持的命令结束，临时段就被撤消。因此，当一个TEMP表空间处于“休息”状态时，该表空间中并不存储任何段。从SYSTEM中分离临时段，会消除数据字典区域中的潜在问题，同时创建一个易于管理的表空间。

可以使用create user命令来指定非SYSTEM临时表空间，如下所示：

```
create user USERNAME identified by PASSWORD
default tablespace DATA
temporaray tablespace TEMP;
```

如果已有帐号，可以使用

```
alter user USERNAME temporary tablespace TEMP;
```

命令重新分配临时表空间。 alter user命令将使以后为这个用户帐号创建的所有临时段都在TEMP中创建。

注意 通常，把SYSTEM和SYS用户的临时表空间设置值改变成非SYSTEM表空间比较合适。

3.2.7 分离用户：USERS

即使在产品数据库中用户不具备创建对象的权限，但他们可能在开发数据库中拥有这些权限。用户对象本质上是暂时性的，他们的开发工作通常并不彻底。其结果便是这些对象应与数据库其他部分分开。这将使用户操作对数据库功能的影响最小化。

要分离用户的对象，需要取消用户在其他表空间中的定额，并且将缺省表空间设置值改为USERS表空间。可以用create user命令规定一个替代的缺省表空间，如下所示：

```
create user USERNAME identified by PASSWORD
default tablespace USERS
temporary tablespace TEMP;
```

在用户帐号创建后，可以用下面的命令来重新指定缺省表空间。重新指定缺省表空间将使不用tablespace子句创建的对象存储在USERS中。

```
alter user USERNAME default tablespace USERS;
```

3.3 扩展OFA

USERS表空间是最后一个由传统OFA调用的主要部分。然而，还有一些扩展有可能对你的数据库有益。这些扩展将在下面几节讨论，它们会在处理例外情况时，根据使用环境帮助进一步分离对象，而不会影响产品的安装。

3.3.1 分离低使用的数据段：DATA_2

回顾数据表列表时会发现，可以根据数据表的特征将它们合并为两个或多个组：一些含有动态数据，另一些含有静态数据；静态数据表可以包含一个状态列表。例如，当进行查询时，对静态数据表的访问通常与对动态数据的访问并发进行。

通过把全部静态数据表放置在一个专用表空间中，就可以将这种并发的I/O在多个文件间(以及相应的多个磁盘间，为了减少I/O冲突)进行分配。对DATA表空间的管理功能(如清除碎片)，现在就只出现在那些需要协助的表中。同时，静态数据表的表空间——DATA_2应当是静态的且便于维护。

根据数据库的大小和使用的特征，可以有多种类型的DATA表空间。除了具有静态表的DATA_2表空间外，你还可以有下述的多种DATA表空间：

- Aggregation(聚合) 如果你有一个数据仓库，就很可能把聚合存储在不同的表中。因为这些表基于导出数据并可能经常撤消或重建，应将它们与主事务表分开。
- Snapshot(快照) 与聚合一样，快照也基于导出的数据，并且以比应用程序的事务表更

频繁的频率撤消和重建。

- Temporary work table(临时工作表) 如果频繁地把其他系统的数据装入你的数据库，可以先把这个数据装入临时表，然后再把它移到你的事务表。这样的临时工作表应与数据库的其他表分开。
- Partition(分区) 如果广泛地使用分区，就应将分区跨表空间(也跨数据文件)分开。把它们跨表空间分开将使你能把表中的当前数据与表中的归档数据分开。

如何能知道一个表是否是静态表？假若对应用程序不熟悉，就可能需要对表的访问进行审计，以确定哪些表最常使用。除了所执行的访问类型(插入、选择等等)外，每次访问一个表时审计(见第9章)都能进行记录。对许多表的访问进行审计可以生成大量的审计记录，所以执行一天的审计后就禁止审计。应用程序使用一天应该能充分确定使用情况。当分析审计结果时，除了对每个表最常见的访问类型外，还应查找最频繁访问的那些表。正常使用时，如果没有针对一个表的insert、update或delete操作，就应考虑把该表从DATA表空间转移到一个更合适的表空间。

3.3.2 分离低使用的索引段：INDEXS_2

低使用的静态数据表的索引也有可能是低使用和静态的。要简化INDEXES表空间的管理操作，可以将这些静态表的索引移到一个单独的INDEXES_2表空间中。这样有助于改善性能调整选项，因为索引间的并发I/O现在可以在多个磁盘驱动器间分配。

如果低使用的索引已经在INDEXES表空间中创建，那么必须关闭并且在INDEXES_2表空间中重新创建它们。这通常与把低使用表移到DATA_2表空间的操作同步进行。如果索引是通过PRIMARY KEY或UNIQUE约束定义创建的，则可能需要修改这种约束条件。

下面给出了一个自动创建索引的表空间说明例子。这个例子在EMPLOYEE_TYPE静态表的Description列中有一个UNIQUE约束条件。数据库为这个约束条件创建的UNIQUE索引将直接存储在INDEXES_2表空间中。

```
alter table EMPLOYEE_TYPE
add constraint UNIQ_DESCR  unique (DESCRIPTION)
using index tablespace INDEXES_2;
```

如果索引已经存在，就可以使用alter index命令中的rebuild子句把索引从当前表空间转移到一个新的表空间。请参见本章前面的3.2.3节。

如果为聚合、快照、临时工作表和分区创建另外的DATA表空间，则应创建另外的INDEX表空间以支持这些表的索引。

3.3.3 分离工具索引：TOOLS_1

如果数据库对TOOLS表空间有较多操作，则这些TOOLS表的索引可以移到另一个表空间中。这在将TOOLS表空间看作是DATA表空间的环境中是很合适的。这些表空间中的表会导致大量的数据库I/O。

重建时，可以使用alter index命令中的rebuild子句将已有的索引移到另一个表空间中。在下面的例子中，JOB_PK索引被移到INDEXES表空间中并为它赋予新的storage值：

```
alter index JOB_PK rebuild
```

```
tablespace INDEXES
storage (initial 2M next 2M pctincrease 0) ;
```

3.3.4 分离特殊回滚段：RBS_2

RBS表空间中的回滚段大小和数量必须合适，以支持应用程序的产品使用（见第7章）。但经常会出现一种事务（一般是批事务），其大小因超过产品回滚段配置而不被支持。当执行这种事务时，这个批事务会占用一个产品回滚段并进行大量扩展，不断占用自由空间，直到事务处理成功或失败为止。

这种情况是可以避免的。产品回滚段应由产品用户使用。特殊的事务需求（例如大型数据装入、聚合或删除）应由另一个单独的回滚段来处理。若要规定这个回滚段，应用程序必须在执行事务处理之前使用下述命令：

```
set transaction use rollback segment SEGMENT_NAME
```

应用程序中这类代码的详细设置需求，可能会对应用小组中的数据库管理人员造成极大的困难（见第5章）。然而，set transaction命令只能解决部分问题，因为所选择的回滚段占用了RBS产品表空间的空间。

创建一个单独的回滚段表空间唯一地支持这种类型的事务。当事务完成时，回滚段应当设置成非激活状态或者撤消（其表空间也撤消，从而节省磁盘空间）。另外，根据功能要求将逻辑对象进行分离，可以大大简化它们的管理。

3.3.5 分离用户特殊临时段：TEMP_*USER*

同RBS_2一样，最后一个重要的表空间是一个特殊表空间，用于满足应用程序用户的特殊要求。某些用户，例如Oracle Application数据库中的GL（the General Ledger模式），可能需要比其他用户更多的临时段。在这种情况下，可把这些临时段从标准TEMP表空间分离出来。由于通过TEMP_*USER*既可以设计系统的普通使用性能，又可以处理例外情况，从而简化了管理操作。实施时，以用户名为后缀命名这种表空间，例如TEMP_GL或TEMP_SCOTT。

也可以使用create user命令来规定用户的临时表空间：

```
create user USERNAME identified by PASSWORD
default tablespace TABLESPACE_NAME
temporary tablespace temp_USER;
```

如果已经创建用户，可以用下面的命令变更临时表空间的设置：

```
alter user USERNAME temporary tablespace TEMP_USER;
```

alter user命令将使以后为这个用户帐号创建的所有临时段都在用户自定义的TEMP_*USER*表空间中被创建。

如果应用程序使用一个用于全部用户的数据库登录，你的选项将更受限制。在这种环境中，对临时表空间的分配和调整会影响全部用户。在这种配置中你有两种主要选择：

1) 除了较小的事务外，调整TEMP临时表空间的大小以支持非常大的事务。不要创建另一个临时表空间。

2) 除了TEMP外，还创建一个TEMP_*USER*表空间。在工作时间以外安排大型事务运行。

在执行大型事务之前，更改用户的临时表空间设置以指向 TEMP_*USER*。当事务处理完成时，将用户的临时表空间设置变回TEMP。

3.3.6 附加的应用特殊OFA扩展

可以利用应用程序在数据库中增加一些对象类型。每一种对象类型都应当存储在各自的表空间中，这样可以使对象彼此之间的影响最小化。这些类型是：

1. SNAPS

用于快照 (见第16章)。快照表和索引的管理与数据库中其他大多数表不同。

2. SNAPS_1

用于对快照的索引。

3. AGG_DATA

用于显形图和聚合表。与快照一样，显形图和聚合基于导出的数据。

4. AGG_DATA_1

用于对显形图和聚合表的索引。

5. PARTITIONS

用于分区(见第12章)。可以使用分区来分配I/O负荷并简化大型表的管理。应该确定最常使用的分区，并且把它们作为单独表来管理。

6. PARTITIONS_1

用于与分区相关的局部和全部索引。

7. TEMP_WORK

用于大型数据装载(见第12章)。临时工作段以数据删除或表截断之后的大型批处理装载为特征。

如果不把复制、显形图、分区或临时工作表用于数据装载，就可以不需要这些额外的表空间。如果使用分区，只要分区存在，就可以将它们转移到不同的表空间 (参见第12章和附录A中的alter table命令)。如果使用局部分区索引 (参见第12章)，应当将局部索引从它们各自的表分区中分离出来。

3.4 合理的逻辑设计

数据库逻辑设计的结果应当符合下述准则：

- 以相同方式使用的段类型应当存储在一起。
- 应当按照最通常的使用情况(事务大小、用户数量、事务数量等)来设计系统。
- 应有用于例外情况的单独区域。
- 应使表空间冲突最小化。
- 应将数据字典分离开。

要符合这些规则，需要数据库管理人员知道应用程序的执行情况：使用什么工具，什么表最活跃，什么时候出现数据装载，什么用户会有特殊的资源要求以及标准事务如何执行等。这就需要数据库管理人员在应用程序开发进程中较全面地介入 (参见第5章)。第三方软件商所提供的应用程序也需要数据库管理人员介入 (参见第11章)。

符合这些准则的系统，其变化的段类型不会干扰彼此间的需求。这样将简化对数据库的

管理，隔离并解决性能问题。如果数据库按这种方式设计，当出现段或自由空间碎片时 (参见第4章和第8章)，就可以大大简化解决问题的办法。

在这种配置中，唯一可能存在多个段类型的非 STSTEM表空间是USERS表空间。如果开发环境也被当作一个测试环境，则应当将用户的索引分隔进一个 USER_1表空间。

将一个合理的逻辑数据库布局与一个完善的物理数据库布局 (见第4章)结合起来构成的系统，在第一次后期产品检查后，几乎不需要调整 。前期计划阶段的结果在数据库的灵活性和性能方面都得到了体现。执行这种设计所花的代价很小；它可以自动在你的数据库脚本文件中生成，并通过安装Oracle软件自动成为脚本文件文件的一部分。系统最终的整体设计应当是表3-1所示的逻辑分类的组合。

一旦建立标准的配置，无论什么时候创建一个新数据库都可以使用它。运用标准配置将简化管理工作，因为每个数据库都遵循相同的表空间使用准则。这些准则按类型和特性将段分离，以便查出大多数问题并简化解决方案。

表3-1 优化的数据库中段的逻辑分布

表 空 间	用 途
SYSTEM	数据字典
DATA	标准操作表
DATA_2	标准操作时使用的静态表
INDEXES	标准操作表的索引
INDEXES_2	静态表的索引
RBS	标准操作的回滚段
RBS_2	用于数据装载的特定回滚段
TEMP	标准操作的临时段
TEMP_*USER*	由特定用户创建的临时段
TOOLS	RDBMS工具表
TOOLS_1	频繁使用的RDBMS工具表的索引
USERS	开发数据库中的用户对象
USERS_1	测试数据库中的用户索引
SANPS	快照表
SANPS_1	快照表上的索引
AGG_DATA	聚合表和显形图
AGG_DATA_1	聚合表和显形图上的索引
PARTITIONS	表或索引段的分区，为它们创建多个表空间
PARTITIONS_1	分区上的局部和全局索引
TEMP_WORK	数据装载时使用的临时表

3.5 解决方案

可以使用表3-1推荐的表空间类型来设计最适合自己需要的配置。表 3-2列出了数据库应用程序中最常见的配置。

表3-2 常用表空间配置

数据库类型	表 空 间
小型开发数据库	SYSTEM
	DATA
	INDEXES

(续)

数据库类型	表 空 间
	INDEXES
	RBS
	TEMP
	USERS
	TOOLS
产品OLTP数据库	SYSTEM
	DATA
	DATA_2
	INDEXES
	INDEXES_2
	RBS
	RBS_2
	TEMP
	TEMP_USER
	TOOLS
具有历史数据的产品OLTP数据库	SYSTEM
	DATA
	DATA_2
	DATA_ARCHIVE
	INDEXES
	INDESEX_2
	INDEXES_ARCHIVE
	RBS
	RBS_2
	TEMP
	TEMP_USER
	TOOLS
数据仓库	SYSTEM
	DATA
	DATA_2
	INDEXES
	INDEXES_2
	RBS
	RBS_2
	TEMP
	TEMP_USER
	TOOLS
	PARTITIONS
	PARTITIONS_1
	AGG_DATA
	AGG_DATA_1
	SNAPS
	SNAPS_1
	TEMP_WORK
	TEMP_WORK_1