第 32 卷 第 16 期 计 算 机 工 程 2006 年 8 月
Vol.32 № 16 Computer Engineering August 2006
·软件技术与数据库· 文章编号：1000—3428(2006)16—0038—03 文献标识码：A 中图分类号：TP311.13

# 表空间存储策略在 PostgreSQL 中的研究与实现

古 锐，亓 伟，叶晓俊
(清华大学软件学院信息系统与工程研究所，北京 100084)

**摘 要：**分析了数据库管理系统中数据存储实现策略所采用的系统管理表空间(SMS)和 DBMS 管理表空间(DMS)这两种主要的表空间管理方式。基于 PostgreSQL 数据库管理系统，给出了在其上实现这两种表空间共存的策略，并描述了 DMS 表空间的实现方法。
**关键词：**PostgreSQL；表空间；系统管理表空间；DBMS 管理表空间

# Research and Implementation of Tablespace Storage Strategy in PostgreSQL

**GU Rui, QI Wei, YE Xiaojun**
(Institute of Information System and Engineering, School of Software, Tsinghua University, Beijing 100084)

**【Abstract】** Tablespace is the main data storage management strategy of DBMS. This paper analyzes the two primary approaches to manage the tablespace: system managed space(SMS) and DBMS managed space(DMS). Then it proposes a strategy to co-exist the two strategies based on PostgreSQL, and especially presents the method for the implementation of DMS.
**【Key words】** PostgreSQL; Tablespace; System manager space(SMS); DBMS manager space (DMS)

## 1 概述

数据库管理系统(DBMS)由一个互相关联的数据的集合和一组用以访问这些数据的程序组成，基本目标是要提供一个可以方便、有效地存取数据库信息的环境。数据库常常需要大量存储空间来存储数据，数据存储策略的好坏直接关系到数据库的性能及可扩展性。目前主流的商业数据库管理系统，如：Oracle，DB2，SQL Server 等，以及一些主要的开源数据库管理系统，如：PostgreSQL，MySQL 等，在其数据存储的实现策略方面均采用了表空间的管理方式。

表空间可以被看作为一个容纳数据库对象的容器，目前表空间的管理方式主要可以分为两种：系统管理表空间(System Managed Space，SMS)的方式和 DBMS 管理表空间(DBMS Managed Space，DMS)的方式。以 DB2 为例，系统管理表空间是由底层操作系统维护的目录或文件系统。DB2 在目录中创建文件对象，并将关系数据分配到每个这样的文件中，每个关系默认对应一个文件。而 DBMS 管理表空间是 DB2 控制的原始设备或预分配的文件，DB2 自行创建分配映射表并管理 DMS 表空间，Oracle 也采用了类似的结构。

表 1 描述了主要的商业及开源数据库管理系统支持的表空间管理方式。

**表 1 数据库管理系统的表空间管理方式**

| DBMS | SMS | DMS |
|---|---|---|
| Oracle | 不支持 | 支持 |
| DB2 | 支持 | 支持 |
| SQL Server | 不支持 | 支持 |
| PostgreSQL | 支持 | 不支持 |

## 2 SMS 方式与 DMS 方式的比较

数据库管理系统在实现时，因为复杂性等原因往往建立在文件系统之上实现数据的存储。但是如果仅采用 SMS 方式，即一个关系对应一个文件，那么若为文件分配的初始空间过小，则对数据进行操作将导致频繁的文件扩展，产生文件碎片，并增加寻道时间和旋转延迟，从而降低效率，所以数据库管理系统往往会申请较大的操作系统文件，然后自行管理其内部空间的分配使用情况。如果初始为文件预分配大量空间，会造成空间浪费，故通常是将多个关系存放在同一个大文件中，达到性能和空间的平衡，即 DMS 的管理方式，其内部策略则借鉴了文件系统管理磁盘时所使用的结构。

表 2 对 SMS 和 DMS 的功能特性做了相应的比较。

**表 2 SMS 和 DMS 的比较**

| | SMS | DMS |
|---|---|---|
| 对象管理 | 操作系统管理，文件名唯一 | 数据库管理系统分配映射表管理 |
| 空间分配 | 依据需要自动增长/缩小 | 预分配空间 |
| 易管理性 | 最好：几乎不需要优化(例如，OS 的预取方式通常很好)；空间管理方式简单 | 较好：需要做一些调优，如取大小，预取方式，缓冲池存放等；空间管理较复杂 |
| 性能表现 | 很好 | 最好：大约可提升 5%~10%的性能。 |
| 应用场景 | 小型应用、临时数据管理 | 大型数据库 |

主流的商业数据库管理系统都采用了 DMS 的方式，这种方式可以提供复杂的功能且更好的提高性能。但 SMS 的方式在某些小型应用的情况下也有其自身的优势。

## 3 在 PostgreSQL 上实现 SMS 和 DMS 方式共存的策略

PostgreSQL 采用的是小文件方式的数据存储策略，类似于 DB2 的 SMS 的方式，借助操作系统来完成部分必要的功

**基金项目：**国家“973”计划基金资助项目(2004CB719400)；国家“863”计划基金资助项目(2003AA413230)
**作者简介：**古 锐(1981—)，男，硕士生，主研方向：数据库管理系统的存储策略；亓 伟，硕士；叶晓俊，副教授
**收稿日期：**2005-09-19 **E-mail：**gur0303@mails.tsinghua.edu.cn

能。在 PostgreSQL 中创建一个数据库时，可以在文件系统中指定一个目录或使用缺省目录作为数据库存储数据文件的位置。从 PostgreSQL8.0 开始，可以创建新的数据目录，也称为表空间。一个关系(表、索引)对应一个或多个数据文件，可以指定关系文件所属的表空间。所以从 PostgreSQL8.0 开始，通过数据库 ID、表空间 ID 和文件 OID 定位文件的物理存储位置。

当添加 DMS 表空间管理方式后，变革了 PostgreSQL 存储的体系结构，在表现上如图 1 所示。

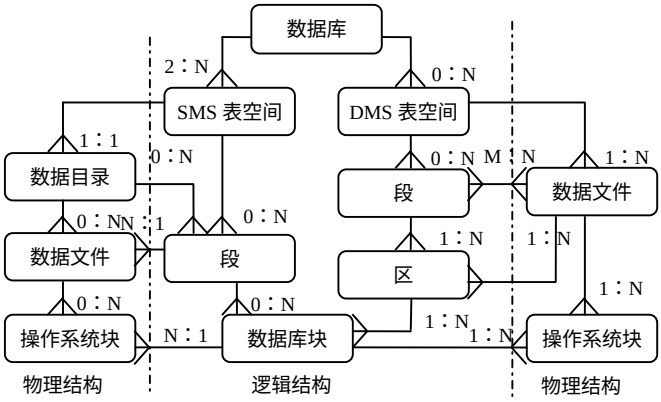


**图 1 变革后的 PostgreSQL 存储结构示意图**

变革后的数据库缺省使用 SMS 方式的表空间，设定至少有两个 SMS 表空间：一个是 SYSTEM 表空间，用来存放系统表；另一个是 DEFAULT 表空间，即系统默认的表空间，如果创建某个关系未指定所属表空间，该关系将被存放在 DEFAULT 表空间中。

DMS 表空间在物理上由一个或多个的数据文件组成，在逻辑上由段组成。段由若干个区组成，区由一个到多个连续的数据库块组成。不同的区可以属于不同的数据文件，因此段可能被分配到多个数据文件中。

SMS 表空间在物理上对应一个数据目录。只有在 SMS 表空间中创建关系时，才实际创建物理文件。关系在存储时对应一个段。在 SMS 表空间中创建关系时，关系的数据量增大到一定程度(缺省为 1G)，会创建第 2 个文件存储关系的数据，属于一个关系的所有文件统称为一个段。

DMS 表空间下的数据文件是预分配空间的，可以存放多个段的数据。SMS 表空间下的数据文件不预分配空间，是随着数据的存储自动进行扩展的。

由于 PostgreSQL8.0 已经支持 SMS 方式，因此 DMS 的实现方式是本文讨论的重点。

## 4 DMS 方式的实现

本节给出一种基于 PostgreSQL 的 DMS 方式的具体实现方法，这种方法借鉴了 Oracle 的分配策略和自动段空间管理的方式以及 InnoDB 和 SQL Server 的分配实现方式。

### 4.1 数据文件结构

属于 DMS 表空间的数据文件，其结构如图 2 所示。

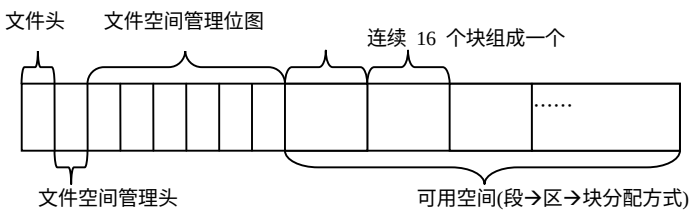


**图 2 文件管理结构**

DMS 表空间的数据文件是预分配空间的，所以要记录预分配的大小、已经使用的大小、文件所属的表空间等基本信息，这些信息保存在文件头中，结构如下：

```
typedef struct fileheaderdata
{
        int magic;          //校验码
        XLogRecPtr pd_lsn;//日志修改号
        Oid spcid;          //所在表空间 ID
        Oid fileid;         //文件 ID
        int size;           //文件大小，以块为单位
        int used_size;//文件已用大小，以块为单位
        int high_water_mark;//区号，此区之后的所有区从来没有
//被用过
        int low_water_mark;//第 1 个可用区区号
}FileHeaderData;
```

以上信息存储在文件的第 1 块上。文件的第 2 块，用于存放该文件头部 6 个文件位图页的日志情况。在这之后的 6 个位图页，与 Oracle 不同，Oracle 文件头部的位图页记录的是块的分配情况，而这里记录的是区的分配情况。位图的操作较为复杂，故用单个字节记录一个区的分配情况来简化操作，为 1 表示该区已被分配出去，0 表示该区未被分配。区的组织借鉴 InnoDB 的方式，大小固定，由 16 个连续的块组成一个区。一个文件块的大小是 8kB，6 个位图页可管理的文件大小为 8 192×6×16×8kB=6GB。在大多数 32 位的操作系统上，文件大小被限制在 4GB，所以 6 个位图页够用了。

假定记录区 E 的分配情况的字节在第 N(范围：1~6)个位图页的 OFFSET(0~8 191)位置上，那么区 E 的区号被认为是

$$E = (N-1)\times 8192+OFFSET \quad (1)$$

区 E 的首块块号为

$$BLN= ((N-1)\times 8192 + OFFSET)\times 16 + 8 \quad (2)$$

### 4.2 段结构

PostgreSQL 采用了多版本的方式来实现并发控制，多版本信息与关系的数据混杂存储在一起，并不像 Oracle 的实现，多版本数据统一存储在回退段中，所以目前的实现仅需要支持数据段和索引段两种类型。鉴于数据段和索引段存储的数据都是记录的形式，仅仅是组织结构不同，所以采用相同的段结构。注意，通常一个关系对应一个段，而一个数据文件可能包含多个段，一个段也可能存在于多个数据文件当中。

段的结构借鉴了 Oracle 的自动段空间管理的策略。已经分配给段的所有区的首页均是 1 个一级位图页(即区首页)，该页记录属于该区的每一个块的状态、总的可用块的数目、从未使用过的块的数目、第一个可用块等信息。其中状态主要包括每个块的可用级别，以利于对各个块进行空间管理。

属于段的第 1 个区的第 2 页是一个二级位图页，记录管理着 512 个一级位图页的位置及其可用级别。第 3 页是段头页，记录着所有二级位图页的位置、高水位线、当前段的区数目等信息。

根据如上的结构设计，在具体实现中，一个段仅在其首个区中包含一个段头页，它最多可记录管理 1 016 个二级位图页。而每个二级位图页设定管理 512 个一级位图页，因此一个段最大为 1 016×512 = 520 192 个区，所以段的最大大小为 520 192×16×8k=63.5G，能够满足一般的应用需求。段的第 1 个区中的二级位图页管理区号为 0～511 的 512 个区的一级位图页信息，在区号为 512 的区中，其第 2 页被用来作为该段的第 2 个二级位图页，

管理区号为 512～1 023 的 512 个区的一级位图页信息，以下雷同。

### 4.3 段创建算法

通常一个关系对应一个段，段是属于表空间的。创建段，按照表空间区分配算法为段分配一个段首区，并初始化前 3 页，即对如上所述的段头页、二级位图页、一级位图页分别进行初始化。

### 4.4 段扩展算法

当段空间不够用时，需要进行段扩展，调用表空间区分配算法新分配一个区。初始化区首页，即该区的一级位图页。分配区成功后，修改段头记录相应字段的值，并在二级位图页中记录一级位图页的位置。

需要注意的是，当新分配区的区号为 512 的倍数时，记录一级位图页的二级位图页需要扩展一页，此时新分配区的第 2 页便被用作二级位图页，那么该区实际存放关系数据的块数为 14。

### 4.5 表空间区分配算法

表空间区分配算法借鉴 SQL Server，采用比例填充算法，使用这一算法，数据文件按照大小成比例地被填充，使得所有的文件按相同的使用率被使用。如果文件 1 的大小为 400MB，文件 2 的大小为 100MB，将在文件 1 中分配的 4 个区，而在文件 2 中分配的 1 个区。通过这种方式，文件 1 不会先于文件 2 填满；反之亦然，数据将按比例在文件之间分布。在实现中，我们通过计算文件头部记录的已用大小 used_size 与目前大小 size 的比率来选用文件。即新分配区时选择 used_size/size 值最小的文件。

### 4.6 文件区分配算法

选择哪一个文件分配区由表空间区分配算法决定，选定了文件后，只需要分配一个区，返回区首块的地址即可。

首先判断该文件是否有足够的空间分配 1 个区，即判断 size-used_size 是否大于 16，如果不满足则分配区失败。

然后根据文件头中记录的 low_water_mark，读入相应的位图页。并从 low_water_mark 记录的位置开始搜索，找到 1 个为 0 的字节，按照式(1)和式(2)计算区号和块号返回。并设置 low_water_mark 为对应区号加 1。

### 4.7 文件区回收算法

区的回收非常简单，只需要根据区首块的块号计算出区号，从而定位它在记录区分配情况的文件头部位图页中的位置，修改其值为 0。实际上是式(1)和(2)的逆运算。

当然，文件头 used_size 和 low_water_mark 等字段要做相应的修改。

### 4.8 段释放算法

首先读取段头页，按顺序遍历二级位图页，根据每个二级位图页中记录的一级位图页的位置，也即区首块的位置。得到了每个区首块的位置，然后调用文件区回收算法，依次回收整个段所包含的区，由此来释放整个段所占用的空间。

### 4.9 页分配算法

虽然按区预分配空间，但使用时仍是按页使用。当已使用的页的空间不够用时，需要再分配新的页。

段头页中高水位线 water_mark 之前的页均为已分配使用的页，从高水位线开始的页未被使用。所以页分配时，只需要返回高水位线指向的页面位置即可。高水位线的结构如下：

```
typedef struct highwatermark
{
    int extent;     //段内区号，从 0 递增
    int block;      //区内块号(1~15)
}HighWaterMark; //记录在段头页中
```

water_mark 的 extent 记录的是属于段的相对区号，由此可以从段头页中找到记录这个区的二级位图页位置。读取相应二级位图页，根据偏移地址得到区首页的位置。根据 water_mark 的 block 值，可以知道分配出去的是该区的哪个页。修改区首页中该区的相关信息。

页分配之后，water_mark.block 值置为分配出去的页相对区首页的位置加 1，当 water_mark.block =16 时置其为 1，water_mark.extent 加 1。

当 water_mark.extent 等于段头页中记录的区数目时，说明预分配的空间已经使用完毕，段需要扩展了。首先调用段扩展算法扩展段，再分配页。

## 5 结果分析

数据库管理系统应用在不同的领域会有不同的负载特征，比较重要的负载类型可分为 3 类：OLTP(On-Line Transaction Processing)，OLAP(On-Line Analytical Processing)以及二者的组合。我们选择 TPC-C 基准性能测试和 MySQL 发布的软件包中所包含一组基准测试来分别模拟 OLTP 和 OLAP 环境。

所得的实验数据较多，因为篇幅所限，这里不一一列出。从综合测试结果来看，DMS 表空间的方式在 OLAP 的测试即单客户端大数据量的情况下，绝大多数测试项目在表现上都比 SMS 方式好；在 OLTP 测试中，数据量和并发事务少的情况下，表现也比 SMS 方式稍好；但在数据量大，并发事务多的情况下，表现与 SMS 方式基本持平。

就当前的表现来看，DMS 方式仍然有待提高：(1)PostgreSQL 本身的存储方式仍需改进，比如非覆盖式的存储方式以及检查点的算法；(2)对 DMS 表空间的具体实现可以作进一步的改进。比如目前对位图页的操作存在较严重的锁争用问题，拟在将来的工作中采用对其分段加锁的办法予以改善。另外，一个好的文件组织方式也可以显著地提高系统性能。总的来说，DMS 方式在更多方面要比 SMS 方式占有优势。

## 6 小结

系统管理表空间和 DBMS 管理表空间是两种主要的表空间管理方式。本文在简要比较了这两种表空间管理方式的特点之后，基于 PostgreSQL 数据库管理系统，给出了在其上实现这两种表空间的策略，并重点阐述了 DMS 表空间的实现方法。由于不同表空间管理方式的实现与诸多模块均具有相关性，因此针对 PostgreSQL 本身的特征，对数据字典、数据字典缓存、SQL 语句解析、缓冲池管理、统计信息搜集等相关模块均做了相应的改动。

### 参考文献


1 Silberschatz A, Korth H F, Sudarshan S. Database System Concepts (Fourth Edition)[M]. Beijing: China Machine Press, 2003.

2 Oracle®.Database Administrator's Guide 9i[Z]. Release1(9.0.1)., Part Number B10739-01, 2001.

3 Stonebraker M. The Design of the POSTGRES Storage System[C]. Proceedings of the 13th VLDB Conference, Brighton, 1987.

4 Microsoft®. SQL Server™ 2000 文档[Z]. 2000.