MyISAM存储引擎的分析与改进*

张 萍

莆田学院电子信息工程学系，福建莆田 351100

摘 要：本文分析了数据存储实现策略中所采用的系统管理表空间（SMS）和数据库管理表空间(DMS)。并基于 MySQL 数据库管理系统的 MyISAM 存储引擎，实现了 DMS 的管理方式。通过实验结果得知，改进后的存储引擎性能有所提高。

关键词：MySQL MyISAM 表空间 系统管理表空间 DMS 管理表空间

1 背 景

Mysql 是一个非常流行的开源数据库管理系统，在市场上占据 25%左右[1]，其主要有两个数据引擎，支持非事务处理的 MyISAM 和支持事务的 Innodb，前者因为没有事务开支，执行更快，存储需要更少的磁盘空间，执行更新需要更少的内存。但是由于使用单文件存储关系数据，文件随着关系数据的存储自动扩展。数据操作的频繁，导致频繁的文件扩展，会浪费时间并且导致文件碎片，增加了寻道时间和旋转延迟。故本文提出修改 MyISAM 引擎原有的数据存储方式，结合系统管理表空间（SMS）和数据库管理表空间(DMS）[2]方式，改进了 MyISAM 存储引擎，实验证明效率提高了。

2 MyISAM 存储引擎

MyISAM 是 Mysql 默认的存储引擎。每个 MyISAM 在磁盘上存储成三个文件[4]。第一个文件的名字以表的名字开始，扩展名指出文件类型。frm 文件存储表定义。数据文件的扩展名为.MYD (MYData)。索引文件的扩展名是.MYI(MYIndex)。

2.1 frm 文件存储表定义

在 MySQL 下使用的任何数据引擎，其产生的每张表都用一个相应的.frm 文件来描述表的信息。表的信息一般包含表采用的数据引擎类型，列表及数据类型等。

2.2 MYD 文件存储表数据

MyISAM 支持三种不同存储格式[5]。其中两个（固定格式和动态格式)根据正使用的列的类型来自动选择。第三个，即已压缩格式，只能使用 **myisampack** 工具来创建。对于固定格式的表，.MYD 存储一个字节的记录头部以及每列数据。对于动态格式的表，.MYD 存储记录头部，数据长度，未使用长度以及数据等。对于压缩格式的表，.MYD 存储的列的数据是经过 Huffman 编码的。

2.3 MYI 文件存储表的索引信息

MYI 文件包含两部分：头部信息和关键字值。头部信息包含四段：state、base、keydef 和 recinfo。主

*作者简介：张萍, 福建莆田学院电子信息工程系.

要有头部长度，空块指针，文件长度，关键字值地址等。关键字值存放在各个块（1024 字节）当中，每块只能属于一个关键字。当关键字值的数据没有足够的空间存放时，则增加一块的文件空间。

3 表 空 间

与其他的数据库引擎不同，MYISAM 数据引擎没有使用页来存储数据。也没有表空间的概念。

表空间是数据库及存储在该数据库中的表之间的逻辑层。表空间在数据库中创建，表在表空间中创建。使用表空间的一个明显的好处是能够把数据合理的分布存储在不同的磁盘上或者存储在磁盘的不同位置上，有助于提高数据存取的效率。

表空间按管理方式分为两种[6]：系统管理空间（System Management Space：SMS）和数据库管理空间（Database Management Space：DMS）。

SMS 每个容器是操作系统的文件空间中的一个目录；DMS 每个容器是一个固定的、预分配的文件，或是物理设备。

SMS 的管理比较简单，由操作系统自动管理，空间的大小随数据量的变化系统自动调整。

DMS 是由数据库管理的，空间大小在创建时确定，空间不够时要手工添加或删除部分数据以释放空间。表 1 对 SMS 和 DMS 的功能特性做了相应的比较[6]：

表 1 MS 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	根据需要自动增长/缩小	预分配空间
易管理性	最好的：几乎不需要优化，空间管理方式简单等	较好：需要做一些调优，如取大小，预取方式，缓存池存放等；空间管理较复杂等
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，因为这种方式可以提供复杂的功能且更好的提高性能。

4 存储结构的改进设计

由于 SMS 和 DMS 各有优势，本文结合两者的特性，在 MyISAM 引擎基础上结合 InnoDB 的存储结构[7]和 Oracle 的存储结构[8]，设计如图 1 的存储结构[9]：

每张表是由一个数据文件组成和一个索引文件组成，数据文件由文件头部和一系列数据页组成。文件头部需要记录表所属数据库 ID 和名称，表 ID 和名称，文件大小，页大小，每次扩展区数，剩余大小，第一个可用区等基本信息。定义如下的文件头部结构：

```
Typedef struct FileHeader{
Int    magic;//校验码
Int    DatabaseID;
Char* dbname;
Int    TableID;
Char* tablename;
```

Int FileSize;//以页数计算
Int PageSize;//页大小
Int ExBlockNum;//扩展区数
Int FreeBlock;//剩余区数；
Int FreeAddr;//第一个可用区地址
}
以上信息保存在文件的头部，头部之后是一系列页。

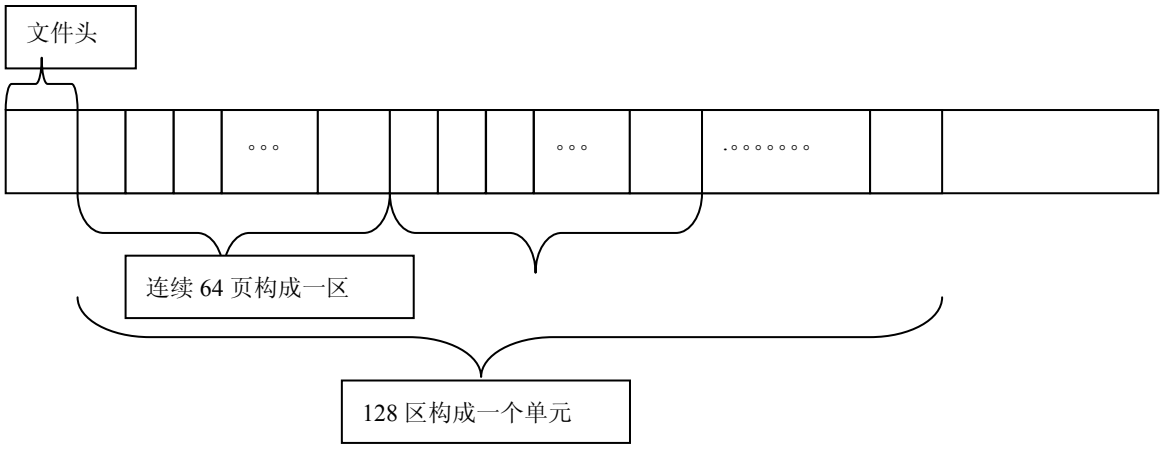

图 1 存储结构

页是文件分配回收的基本单位，每一页默认为 16K 字节。从文件头部后开始为文件的页编号(记为 page num)，第一页的编号为 0，第二页编号为 1，依此类推。每连续的 64 页构成一个区，也就是说所有(page_num-(page_num mod 64))/64 结果相同的页构成一个区。同样每连续的 128 个区构成一个单元。

每一个单元的第一个页是区描述信息页。区描述信息页包括其所属单元所有区的描述信息。区描述信息包含 128 个区描述符。区描述符包括四部分信息：区 ID、区状态、构建链表用节点和页状态位图（每页两位，共 16 字节）。

索引文件采用的是 BTree 聚簇文件组织方式。

索引文件中 B-Tree 的数据分属于两个段，非叶子节点在一个段中，叶子节点数据在一个段中。这样是为了保证叶子节点的数据在磁盘上尽量连续。

B-Tree 的节点对应着数据文件的一个块。其中其根节点记录着两个段的段头信息。根节点对应的页在数据字典中记录着。

B-Tree 的叶子节点存放着记录。叶子节点层次为 0，层次 n>0 的非叶子节点存放着指向 n-1 层的节点。

一个关系，当它有主键时，将生成以主键为唯一标识的聚簇索引。当没有主键时，关系将添加一个 ROWID 字段，唯一标识一条记录，也形成以 ROWID 为唯一标识的聚簇索引。但由于 ROWID 随着插入的递增性，B-Tree 退化为简单的顺序存储的形式。可见，两种情况下，关系都有一个唯一性索引。

每一个节点中我们均定义了一个最小记录字段，最小记录的指针指向第一个子节点的最左边。

B-Tree 在两种情况下分裂：块空间不足时分裂；当连续从同一个方向插入的次数>BTR_PAGE_SEQ_INSERT_LIMIT 时分裂。这是一种预测机制，认为下次可能还会从同样的方向插入，减少分裂次数。

只有当 B-Tree 节点块上没有记录时，删除此页；如果页中的记录大小小于块大小的一半时，尝试与其兄弟页合并；如果无兄弟页，则它是这一层的最后一页，那么减少树的层次。

索引页包含一个页头，页头信息包括了构建 B+树所需的结构以及组织索引页数据所需的结构。

在页头之后是记录区域，存储着实际的索引记录，所有的记录构成一个按字母表排序的线性表。

在页尾部是一个指针数组，称之为指针目录。指针指向的是记录组，每个记录组平均拥有 6 个记录。指针按指向的记录组的第一条记录的顺序排序。指针 I 指向的记录组的第一条记录，拥有一个字段记录着属于这个记录组的记录的数目，即指针 I 与 I-1 之间的记录，计数时它包括指针 I 指向的记录但不包括 I-1 指向的记录。这个数目保持在 4-8 之间，超过 8 会导致指针目录(slot)分裂，小于 4 导致收缩。

这种结构下的搜索，不减慢速度，与有指针指向每条记录相比，差不多。但在插入时，会大大提高速度。因为插入是将记录放入堆中，每 8 次的插入才会导致指针目录的变动，而这个开销很小。

当记录删除时，仅仅是将其从线性表中移除，同时更新记录组计数，如果计数小于 4 了，将会收缩，更新指针目录。当删除的记录是记录记录组计数的那条记录时，需要特殊处理一下。记录的删除并不立即回收空间，而是加入到空闲记录链表中，在插入时可以根据大小再次利用。当页分裂收缩或是变满了但拥有删除的记录时，页需要重组，整理合并空闲空间碎片。

5 结 果 分 析

根据上面的设计，建立如下的实验平台：

硬件：CPU intel celeron 1G，内存 384M，操作系统 Linux redhat 9.0，MySQL 5.1。由于数据引擎不支持事务处理，本实验只用了常见的 create、insert、update 操作进行测试。建立 20 个表 tblisam1 到 tblisam20，表字段为(id int, name char(10))，插入 1,000,000 条记录的存储过程如下：

```
Create procedure p_create()
Begin
Declare counter int;
Set counter=100,000;
While counter>=1 do
Insert into tblisam1 values(counter,”abcde”);
Insert into tblisam2 values(counter,”abcde”);
……
Insert into tblisam20 values(counter,”abcde”);
Set counter=counter-1;
End while;
End;
```

实验结果如表 2 所示。

表 2 select、insert、update 操作实验结果(单位：秒)

	Create(10,000)	Insert（2,000,000）	Update(100,000)
原 MyISAM 引擎	61.54	192.69	304.57
改进的 MyISAM 引擎	66.87	180.13	283.35

从测试结果对比中可见，改进后的存储引擎在表的建立上性能有所下降，主要是因为，每个数据文件都添加了文件头，增加了额外的读写。但在插入和删除操作有性能有所提高。主要由于使用了预分配空间，虽然增加 IO 读写次数，却减少了寻道时间和旋转延迟。

本文在比较 SMS 和 DMS 两种表空间的管理方式后，对 MySQL 的 MyISAM 数据引擎进行了改进，实现了 DMS 表空间的存储方法。实验结果表明对于多表混合频繁删除，插入的应用，改进后的数据引擎，性能上有所提高。

参 考 文 献

[1] Evans 数据公司. http://www.evans.co.uk/.2009-1-16.
[2] 古锐, 亓伟, 叶晓俊. 表空间存储策略在 PostgreSQL 中的研究与实现[J]. 计算机工程, 2006.8: 38-40.
[3] http://www-inst.eecs.berkeley.edu/~cs186/lectures/lecture4Files.pdf.
[4] MyISAM 数据引擎的相关信息. http://forge.mysql.com/wiki/MySQL_Internals_MyISAM.
[5] Mysql5.1 参考手册. http://dev.mysql.com/doc/refman/5.1/zh/storage-engines.html#myisam-table-formats.
[6] IBM Learning Services,DB2 Universal Database Administration Workshop for Windows NT[D]. IBM Learning Services, 2000.
[7] 亓伟. 关系数据库存储子系统研究与实现[D]. 清华大学, 2005.
[8] OCP, Oracle9i DBA Fundamentals [D]. Oracle University, 2001: 8-11.
[9] H.Gracia-Molina, 等. 数据库系统全书[M]. 北京: 机械工业出版社, 2003.
[10] Silberschatz A, Korth H F, Sudarshan S. Database System Concepts(Fourth Edition)[M]. Beijing: China Machine Press, 2003.

The Analysis and Improvement of MyISAM storage engine

Zhang Ping

Putian Colleage, Department of Electronic Information Engineering, Fujian, 351100

Abstract: This paper analyzes system management table space (SMS) and Database Management table space (DMS) in the data storage implementation strategies. And implement DMS based on the MyISAM storage engine. Through the experimental results, the improved STORAGE engine increased performance.

Keywords: MySQL; MyISAM; tablespace; SMS; DMS