# PostgreSQL 12中的小表存储优化

原文：<https://www.2ndquadrant.com/en/blog/optimizing-storage-small-tables-postgresql-12/>

作者：John Naylor 翻译：Nail

## 此前存在的问题

如果你的数据库中有大量的小表，那么你可能浪费了大量的空间。我们来创建一个只有一个属性的表来看看：

CREATE TABLE foo (str text);

INSERT INTO foo VALUES (‘a’);

VACUUM foo;

接下来找到此表的数据文件相对于数据目录($PGDATA)的路径:

SELECT pg\_relation\_filepath(‘foo’);

pg\_relation\_filepath

----------------------

base/16384/16405

(1 row)

（有关PostgreSQL文件和目录的更多信息，请参阅Craig Ringer关于该主题的文章）

我们列出该目录下与此表有相同前缀的文件：

$ ls base/16384/16405\*

base/16384/16405

base/16384/16405\_fsm

base/16384/16405\_vm

这个名字为16405的文件是存储我们表数据的heap文件，那么另外两个呢？他们是存储了一些附加信息的辅助文件，帮助PostgreSQL访问与管理维护表。

* 16405\_fsm是空闲空间映射表。它的作用是记录表中哪些页还有空闲空间可以插入数据。
* 16405\_vm是可见性映射表。通过它我们可以知道哪些页需要被VACUUM或被冻结，也可以知道哪些页在仅索引查询中会被访问。

还有其它一些与此表有关联的文件，通过查询就可以找到：

select

pg\_relation\_filepath(c.reltoastrelid) as toast\_table\_path,

pg\_relation\_filepath(i.indexrelid) as toast\_index\_path

from pg\_class c

left outer join pg\_index i on c.reltoastrelid=i.indrelid

where c.relname = 'foo';

toast\_table\_path | toast\_index\_path

------------------+------------------

base/16384/16408 | base/16384/16410

(1 row)

我们找到了此表对应的toast表和toast表的索引。当我们插入足够大的记录时，就会将数据进行压缩并存储到toast表中。

现在来总结一下，看看存一条记录我们需要多少硬盘空间？

\x

select

pg\_relation\_size(c.oid, 'main') as heap\_size,

pg\_relation\_size(c.oid, 'fsm') as fsm\_size,

pg\_relation\_size(c.oid, 'vm') as vm\_size,

pg\_relation\_size(c.reltoastrelid) as toast\_table\_size,

pg\_relation\_size(i.indexrelid) as toast\_index\_size

from pg\_class c

left outer join pg\_index i on c.reltoastrelid=i.indrelid

where c.relname = 'foo';

-[ RECORD 1 ]----+------

heap\_size | 8192

fsm\_size | 24576

vm\_size | 8192

toast\_table\_size | 0

toast\_index\_size | 8192

存储一条记录，我们需要5个文件总计48KB的空间。你可能会想“所以呢？我的表大部分都在GB级别的，我为什么要在意这几十KB的空间？”。想象一下，假如有一个使用模式(schema)来提供多租户的应用，它在运作时可能会有数千个模式(schema),每个模式都有数百个小表，那么这些小表以及它们的辅助文件数量会急剧增长。

## 有什么办法可以减少空间的消耗吗？

首先，我们可以通过限制列的长度来阻止产生toast表和toast索引：

create table foo\_no\_toast (str varchar(500));

insert into foo\_no\_toast values ('a');

VACUUM foo\_no\_toast;

select

pg\_relation\_size(c.oid, 'main') as heap\_size,

pg\_relation\_size(c.oid, 'fsm') as fsm\_size,

pg\_relation\_size(c.oid, 'vm') as vm\_size,

pg\_relation\_size(c.reltoastrelid) as toast\_table\_size,

pg\_relation\_size(i.indexrelid) as toast\_index\_size

from pg\_class c

left outer join pg\_index i on c.reltoastrelid=i.indrelid

where c.relname = 'foo\_no\_toast';

-[ RECORD 1 ]----+------

heap\_size | 8192

fsm\_size | 24576

vm\_size | 8192

toast\_table\_size |

toast\_index\_size |

因此现在我们有3个文件总计40KB。在Postgres内部机制中，当元组的大小超过某个阈值后就会将元组存放到toast表中。而我们上面这个表，因为可以确定没有元组的大小可以超过阈值，所以Postgres不会为它创建toast表。

然而这样的改进仍然不足。首先，我们仅仅只节约了一点点空间，其次，在创表时我们很难确定一个足够使用又不会触发toast机制的值。我做了很多次试验，试验结果都不同程度的与属性数(列数量)，数据库编码甚至是数据库版本相关。

如果我们能够去掉空闲空间映射表就好了，毕竟它比它跟踪的表要大足足3倍！很快我们就有一个简单的办法来实现。

## 新版本中的改进！

从PostgreSQL 12开始，创建的表只有当它超过4页或者每页8k的情况下超过32k时，才会为其创建空闲空间映射表。

我们在一个新版本上尝试一下：

create table foo\_no\_fsm (str text);

insert into foo\_no\_fsm values ('a');

VACUUM foo\_no\_fsm;

select

pg\_relation\_size(c.oid, 'main') as heap\_size,

pg\_relation\_size(c.oid, 'fsm') as fsm\_size,

pg\_relation\_size(c.oid, 'vm') as vm\_size,

pg\_relation\_size(c.reltoastrelid) as toast\_table\_size,

pg\_relation\_size(i.indexrelid) as toast\_index\_size

from pg\_class c

left outer join pg\_index i on c.reltoastrelid=i.indrelid

where c.relname = 'foo\_no\_fsm';

-[ RECORD 1 ]----+-----

heap\_size | 8192

fsm\_size | 0

vm\_size | 8192

toast\_table\_size | 0

toast\_index\_size | 8192

总共24K，是在PG11上的一半，也没有必须限制元组长度的缺点。顺便说一下，在升级到PG12的过程中就会自动进行检测，将这些小表的空闲空间映射表删除，所以如果你的数据库中有大量的小表，空间的使用量会立即有一个明显的变化。