PostgreSQL 的 database 和 table 与磁盘文件的对应

最近一段时间一直在学习 PostgreSQL 源码中 backend/storage 目录下的源码，学习了 smgr，学习了 buffer，对 PostgreSQL 数据在磁盘的布局，shared buffer 及其缓存替换机制有了一定的了解。本文重点讲述 PostgreSQL 中的数据库和数据库中的表如何在磁盘存储。当然 FSM 和 VM 文件暂不涉及。PostgreSQL 大牛可以一笑而过。

database 对应的磁盘文件：

首先是如何查看我们的 PostgreSQL 有几个数据库 database？

方法有

1 psql -l

我们看到创建任何数据库之前，PostgreSQL 已经存在了 3 个 database。

```
manu@manu:/usr/pgdata$ psql -l
                              List of databases
   Name    | Owner | Encoding |   Collate   |    Ctype    | Access privileges
-----------+-------+----------+-------------+-------------+-------------------
 postgres  | manu  | UTF8     | zh_CN.UTF-8 | zh_CN.UTF-8 |
 template0 | manu  | UTF8     | zh_CN.UTF-8 | zh_CN.UTF-8 | =c/manu          +
           |       |          |             |             | manu=CTc/manu
 template1 | manu  | UTF8     | zh_CN.UTF-8 | zh_CN.UTF-8 | =c/manu          +
           |       |          |             |             | manu=CTc/manu
(3 rows)

manu@manu:/usr/pgdata$ cd base/
```

2 oid2name

oid2name 也可以看到我们当前已经存在的 database

```
manu@manu:/usr/pgdata$ cd base/
manu@manu:/usr/pgdata/base$ ll
总用量 28
drwx------  5 manu manu  4096  5月 19 17:56 ./
drwx------ 13 manu manu  4096  6月  3 21:20 ../
drwx------  2 manu manu 12288  5月 19 17:57 1/
drwx------  2 manu manu  4096  5月 19 17:56 11946/
drwx------  2 manu manu  4096  6月  3 21:20 11954/
manu@manu:/usr/pgdata/base$
manu@manu:/usr/pgdata/base$ od
od          odbcinst    ods-server  odvicopy    odvitype
manu@manu:/usr/pgdata/base$ oid2name
All databases:
    Oid  Database Name  Tablespace
----------------------------------
  11954       postgres  pg_default
  11946      template0  pg_default
      1      template1  pg_default
manu@manu:/usr/pgdata/base$ createdb manu_db
```

那么这三个数据库在磁盘中是如何存储的呢？我们看到/usr/pgdata/base 目录下存在三个目录文件，对应的恰好是三个 database 的 OID.

What is the FUCK OID？

OID 是一个数字，对于 PostgreSQL 的 database 和某个 database 内的 table 都会有一个唯一的 OID 和它对应，对于我这个解释不太满意的，可以看 PostgreSQL 的国内大牛德哥的博文 get PostgreSQL's next oid。从原理到代码解释的都比较清楚。我就不唧唧歪歪了。

新装好的的 PostgreSQL 会有三个默认的 database，对应在 pgdata 目录下的 base 下，每个 database 都有自己的目录。如果我们通过 createdb 新创建一个 database，那么，可以期待在 pgdata/base 目录下会新增一个以 db OID 为名字的目录，我们验证之：

```
manu@manu:/usr/pgdata/base$ ll
总用量 32
drwx------  6 manu manu  4096  6月   3 21:31 ./
drwx------ 13 manu manu  4096  6月   3 21:20 ../
drwx------  2 manu manu 12288  5月  19 17:57 1/
drwx------  2 manu manu  4096  5月  19 17:56 11946/
drwx------  2 manu manu  4096  6月   3 21:20 11954/
drwx------  2 manu manu  4096  6月   3 21:31 16384/
manu@manu:/usr/pgdata/base$
manu@manu:/usr/pgdata/base$ oid2name
All databases:
    Oid  Database Name  Tablespace
----------------------------------
  16384        manu_db  pg_default
  11954       postgres  pg_default
  11946      template0  pg_default
      1      template1  pg_default
manu@manu:/usr/pgdata/base$
```

我们通过 createdb manu_db 创建了一个名字为 manu_db 的 database，我们通过 oid2name 看到 database 的 OID=16384,而在 base 目录下的确新增了一个 16384 的目录，目录下已经有一些文件了，哪怕目前 manu_db 还是个空的 db。

```
-rw------- 1 manu manu  8192 6月  3 21:31 11906
-rw------- 1 manu manu 16384 6月  3 21:31 11907
-rw------- 1 manu manu 24576 6月  3 21:31 11907_fsm
-rw------- 1 manu manu  8192 6月  3 21:31 11907_vm
```

如果你关心这些个文件都是干啥的可以用 oid2name -d manu_db -f 11907 来查看文件是干啥的

```
manu@manu:/usr/pgdata/base/16384$
manu@manu:/usr/pgdata/base/16384$ oid2name -d manu_db -f 11907
From database "manu_db":
  Filenode    Table Name
------------------------
     11907  pg_collation
manu@manu:/usr/pgdata/base/16384$
```

我们看到 11907 文件对应的 table 是 pg_collation，作为初学者，我表示不懂这个表是干啥的。对于这些 database 默认创建的文件我们按下不表。

Table 对应的磁盘文件：

目前 database 对应的磁盘文件基本解决，那么如果我在 manu_db 中创建一个 table，磁盘上会发生那些变化呢？

```
manu@manu:~$ psql manu_db
psql (9.1.9)
Type "help" for help.

manu_db=# create TABLE friends (
manu_db(# firstname varchar(20) not NULL,
manu_db(# lastname  varchar(20) not NULL,
manu_db(# cellphone varchar(20) default NULL,
manu_db(# city      varchar(40) default NULL,
manu_db(# extra     text default NULL);
CREATE TABLE
manu_db=#
```

我们通过 create TABLE 创建了一个名字叫 friends 的 table，这个 table 中没有 ID 这种适合做 key 的字段是我的失误，我就懒得改了。

如何查看名为 friends 的 table 对应的磁盘文件。table 在 PostgreSQL 对应叫做 relfile，是 relation file 的意思。我们可以查询 pg_class 这张表看到：

```
manu@manu:/usr/pgdata/base/16384$ oid2name -d manu_db
From database "manu_db":
  Filenode  Table Name
----------------------
     16385     friends
manu@manu:/usr/pgdata/base/16384$ psql -d manu_db  -c "\d+"
                    List of relations
 Schema |  Name   | Type  | Owner |    Size    | Description
--------+---------+-------+-------+------------+-------------
 public | friends | table | manu  | 8192 bytes |
(1 row)

manu@manu:/usr/pgdata/base/16384$ psql manu_db
psql (9.1.9)
Type "help" for help.

manu_db=# select oid,relfilenode,relname from pg_class where relname = 'friends';
  oid  | relfilenode | relname
-------+-------------+---------
 16385 |       16385 | friends
(1 row)

manu_db=# \q
manu@manu:/usr/pgdata/base/16384$ ll 16385
-rw------- 1 manu manu 0  6月   5 22:27 16385
manu@manu:/usr/pgdata/base/16384$
```

我们从上图中可以看到有多种方法可以看到 table 和 磁盘文件的映射关系：

1 oid2name -d manu_db 会列出 manu_db 中的所有 table 和磁盘上 file 的对应关系

2 select oid, relfilenode, relname from pg_class where relname = 'friends';

推荐使用第二方法，目标很明确就是查找 table 名为 friends 的 relfilenode。

找到 relfilenode 之后，我们可以看到，在 base/16384 目录下新增一个文件 16385,这个文件对应的就是 table friend 对应的文件。

值得注意的事情有 2

1 table 的 OID 和 table 对应的磁盘文件名是相同的。一般如此，也不尽然，对表进行一个操作可以改变文件存储的名称而不改变表的 OID，从而导致两者不一致。greg smith 说 TRUNCATE REINDEX 和 CLUSTER 可能引起这种不一致，我还是菜鸟，不能深刻理解。

2 随着表的不断插入新的条目，这个磁盘文件越来越大，当超过 1G 的时候，这个表会分文件存储，PostgreSQL 叫做分 Segment 存储。会生成一个 16385.1。

这就牵扯到了 buffer size 和 Segment Size，relfile 在 PostgreSQL 中存在 shared buffer，shared buffer 的页面大小为 8192B，所以，会将 buffer 中的页面 flush 到磁盘文件中，所以是 8K 整数倍。我们插入一条记录看一下表 friends 对应的磁盘文件：

```
manu_db=# INSERT INTO friends  values ('Lee','Bean','158XXXXXXX','Nan Jing','');
INSERT 0 1
manu_db=# select * from friends ;
 firstname | lastname | cellphone  |   city   | extra
-----------+----------+------------+----------+-------
 Lee       | Bean     | 158XXXXXXX | Nan Jing |
(1 row)

manu_db=# \q
manu@manu:/usr/pgdata/base/16384$ ll 16385
-rw------- 1 manu manu 8192  6月   5 23:48 16385
manu@manu:/usr/pgdata/base/16384$
```

果然是 8K，我们可以查看这个 block size 和对于大的 relation file 多少个 block 开始分 segment

```
manu@manu:/usr/pgdata/base/16384$
manu@manu:/usr/pgdata/base/16384$ pg_controldata /usr/pgdata/ |grep "Database block size"
Database block size:                  8192
manu@manu:/usr/pgdata/base/16384$ pg_controldata /usr/pgdata/ |grep "Blocks per segment"
Blocks per segment of large relation: 131072
manu@manu:/usr/pgdata/base/16384$
```

第一个值是 8K，单位是 Byte，第二个值是 128K，单位是个，表示 128K 个 block 组成一个 segment，relation file 再增大的话，就分成另一个 segment，比如我们 friends 这个 relation 插入的 item 越来越多，文件 16385 大小超过 128K×8KB=1G 的时候，就要新增一个磁盘文件 16385.1.

那么如何查看那个 relation file 占据最多的磁盘空间呢。这个内容有点超前，毕竟我们才刚刚创建自己的 table，但是没关系 ，这个内容很实用，我们会很好奇，自己哪个 table 会占用最多的磁盘空间，又占用多少磁盘空间：下面这条命令从一个老外的博客中习得（向他致谢，可惜找不到地址了，原谅我没给链接）：

```
SELECT
schemaname,
tablename,
pg_size_pretty(pg_relation_size(schemaname || '.' || tablename)) AS size_p,
pg_total_relation_size(schemaname || '.' || tablename) AS siz,
pg_size_pretty(pg_total_relation_size(schemaname || '.' || tablename)) AS total_size_p,
pg_total_relation_size(schemaname || '.' || tablename) - pg_relation_size(schemaname || '.' || tablename)
AS index_size,
```

```
(100*(pg_total_relation_size(schemaname || '.' || tablename) - pg_relation_size(schemaname || '.' ||
tablename)))/CASE WHEN pg_total_relation_size(schemaname || '.' || tablename) = 0 THEN 1 ELSE
pg_total_relation_size(schemaname || '.' || tablename) END || '%' AS index_pct
FROM pg_tables
ORDER BY siz DESC LIMIT 50;
```

schemaname	tablename	size_p	siz	total_size_p	index_size	index_pct
pg_catalog	pg_depend	328 kB	753664	736 kB	417792	55%
pg_catalog	pg_proc	448 kB	745472	728 kB	286720	38%
pg_catalog	pg_attribute	304 kB	548864	536 kB	237568	43%
pg_catalog	pg_rewrite	96 kB	483328	472 kB	385024	79%
pg_catalog	pg_description	224 kB	376832	368 kB	147456	39%
pg_catalog	pg_statistic	112 kB	245760	240 kB	131072	53%
pg_catalog	pg_operator	104 kB	212992	208 kB	106496	50%
pg_catalog	pg_type	56 kB	147456	144 kB	90112	61%
pg_catalog	pg_class	56 kB	139264	136 kB	81920	58%
pg_catalog	pg_amop	24 kB	139264	136 kB	114688	82%
pg_catalog	pg_constraint	8192 bytes	114688	112 kB	106496	92%
pg_catalog	pg_conversion	16 kB	98304	96 kB	81920	83%
information_schema	sql_features	56 kB	98304	96 kB	40960	41%
pg_catalog	pg_index	24 kB	90112	88 kB	65536	72%
pg_catalog	pg_opclass	16 kB	81920	80 kB	65536	80%
pg_catalog	pg_collation	16 kB	81920	80 kB	65536	80%

因为我的 db 没有啥数据，所以都是系统表占用的空间多，我在我公司的项目中用了这条 sql，就是我们自己的 relation 占据排行榜的前面。

参考文献：

1 PostgreSQL 性能调校

2 一些博客，没有保存地址，十分抱歉，向前辈致敬。