

# MySQL 的数据目录

标签： MySQL 是怎样运行的

---

## 数据库和文件系统的关系

我们知道像InnoDB、MyISAM这样的存储引擎都是把表存储在磁盘上的，而操作系统用来管理磁盘的那个东东又被称为文件系统，所以用专业一点的话来表述就是：像 *InnoDB*、*MyISAM* 这样的存储引擎都是把表存储在文件系统上的。当我们想读取数据的时候，这些存储引擎会从文件系统中把数据读出来返回给我们，当我们想写入数据的时候，这些存储引擎会把这些数据又写回文件系统。本章就是要唠叨一下InnoDB和MyISAM这两个存储引擎的数据如何在文件系统中存储的。

## MySQL数据目录

MySQL服务器程序在启动时会到文件系统的某个目录下加载一些文件，之后在运行过程中产生的数据也都会存储到这个目录下的某些文件中，这个目录就称为数据目录，我们下边就要详细唠唠这个目录下具体都有哪些重要的东西。

### 数据目录和安装目录的区别

我们之前只接触过MySQL的安装目录（在安装MySQL的时候我们可以自己指定），我们重点强调过这个安装目录下非常重要的bin目录，它里边存储了许多关于控制客户端程序和服务器程序的命令（许多可执行文件，比如mysql, mysqld, mysqld\_safe等等等等好几十

个)。而数据目录是用来存储MySQL在运行过程中产生的数据，一定要和本章要讨论的安装目录区别开！**一定要区分开！一定要区分开！一定要区分开！**

## 如何确定MySQL中的数据目录

那说了半天，到底MySQL把数据都存到哪个路径下呢？其实数据目录对应着一个系统变量`datadir`，我们在使用客户端与服务器建立连接之后查看这个系统变量的值就可以了：

```
mysql> SHOW VARIABLES LIKE 'datadir';
+-----+-----+
| Variable_name | Value                               |
+-----+-----+
| datadir       | /usr/local/var/mysql/             |
+-----+-----+
1 row in set (0.00 sec)
```

从结果中可以看出，**在我的计算机上**MySQL的数据目录就是`/usr/local/var/mysql/`，你用你的计算机试试呗～

## 数据目录的结构

MySQL在运行过程中都会产生哪些数据呢？当然会包含我们创建的数据库、表、视图和触发器吧啦吧啦的用户数据，除了这些用户数据，为了程序更好的运行，MySQL也会创建一些其他的额外数据，我们接下来细细的品味一下这个数据目录下的内容。

## 数据库在文件系统中的表示

每当我们使用`CREATE DATABASE 数据库名`语句创建一个数据库的时候，在文件系统上实际发生了什么呢？其实很简单，**每个数据库都对应数据目录下的一个子目录，或者说对应一个文件夹**，我们每当我

们新建一个数据库时，MySQL会帮我们做这两件事儿：

1. 在数据目录下创建一个和数据库名同名的子目录（或者说是文件夹）。
2. 在该与数据库名同名的子目录下创建一个名为`db.opt`的文件，这个文件中包含了该数据库的各种属性，比方说该数据库的字符集和比较规则是个啥。

比方说我们查看一下[在我的计算机上](#)当前有哪些数据库：

```
mysql> SHOW DATABASES;
+-----+
| Database                |
+-----+
| information_schema      |
| charset_demo_db        |
| dahaizi                 |
| mysql                   |
| performance_schema     |
| sys                     |
| xiaohaizi               |
+-----+
7 rows in set (0.00 sec)
```

可以看到在我的计算机上当前有7个数据库，其中`charset_demo_db`、`dahaizi`和`xiaohaizi`数据库是我们自定义的，其余4个数据库是属于MySQL自带的系统数据库。我们再看一下[我的计算机上](#)的数据目录下的内容：

```
.
├─ auto.cnf
├─ ca-key.pem
├─ ca.pem
├─ charset_demo_db
├─ client-cert.pem
├─ client-key.pem
├─ dahaizi
├─ ib_buffer_pool
├─ ib_logfile0
├─ ib_logfile1
├─ ibdata1
├─ ibtmp1
├─ mysql
├─ performance_schema
├─ private_key.pem
├─ public_key.pem
├─ server-cert.pem
├─ server-key.pem
├─ sys
├─ xiaohaizideMacBook-Pro.local.err
├─ xiaohaizideMacBook-Pro.local.pid
└─ xiaohaizi
```

6 directories, 16 files

当然这个数据目录下的文件和子目录比较多哈，但是如果仔细看的话，除了information\_schema这个系统数据库外，其他的数据库在数据目录下都有对应的子目录。这个information\_schema比较特殊，设计MySQL的大叔们对它的实现进行了特殊对待，没有使用相应的数据库目录，我们忽略它的存在就好了哈。

## 表在文件系统中的表示

我们的数据其实都是以记录的形式插入到表中的，每个表的信息其实可以分为两种：

1. 表结构的定义
2. 表中的数据

表结构就是该表的名称是啥，表里边有多少列，每个列的数据类型是啥，有啥约束条件和索引，用的是啥字符集和比较规则吧啦吧啦的各种信息，这些信息都体现在了我们的建表语句中了。为了保存这些信息，InnoDB和MyISAM这两种存储引擎都在数据目录下对应的数据库子目录下创建了一个专门用于描述表结构的文件，文件名是这样：

表名.frm
--------

比方说我们在dahaizi数据库下创建一个名为test的表：

```
mysql> USE dahaizi;
Database changed

mysql> CREATE TABLE test (
    ->     c1 INT
    -> );
Query OK, 0 rows affected (0.03 sec)
```

那在数据库dahaizi对应的子目录下就会创建一个名为test.frm的用于描述表结构的文件。值得注意的是，**这个后缀名为.frm是以二进制格式存储的，我们直接打开会是乱码的～**你还不赶紧在你的计算机上创建个表试试～

描述表结构的文件我们知道怎么存储了，那表中的数据存到什么文件中了呢？在这个问题上，不同的存储引擎就产生了分歧了，下边我们分别看一下InnoDB和MyISAM是用什么文件来保存表中数据的。

## InnoDB是如何存储表数据的

我们前边重点唠叨过InnoDB的一些实现原理，到现在为止我们应该熟悉下边这些东东：

- InnoDB其实是使用页为基本单位来管理存储空间的，默认的页大小为16KB。
- 对于InnoDB存储引擎来说，每个索引都对应着一棵B+树，该B+树的每个节点都是一个数据页，数据页之间不必要是物理连续的，因为数据页之间有双向链表来维护着这些页的顺序。
- InnoDB的聚簇索引的叶子节点存储了完整的用户记录，也就是所谓的**索引即数据，数据即索引**。

为了更好的管理这些页，设计InnoDB的大叔们提出了一个表空间或者文件空间（英文名：table space或者file space）的概念，这个表空间是一个抽象的概念，它可以对应文件系统上一个或多个真实文件（不同表空间对应的文件数量可能不同）。每一个表空间可以被划分为很多很多很多个页，我们的表数据就存放在某个表空间下的某些页里。设计InnoDB的大叔将表空间划分为几种不同的类型，我们一个一个看一下。

### 系统表空间（system tablespace）

这个所谓的系统表空间可以对应文件系统上一个或多个实际的文件，默认情况下，InnoDB会在数据目录下创建一个名为ibdata1（在你的数据目录下找找看有木有）、大小为12M的文件，这个文件就是对应的系统表空间在文件系统上的表示。怎么才12M？这么点儿还没插多少数据就用完了，哈哈，那是因为这个文件是所谓的自扩展文件，也就是当不够用的时候它会自己增加文件大小～

当然，如果你想让系统表空间对应文件系统上多个实际文件，或者仅仅觉得原来的ibdata1这个文件名难听，那可以在MySQL启动时配置对应的文件路径以及它们的大小，比如我们这样修改一下配置文件：

```
[server]
innodb_data_file_path=data1:512M;data2:512M:autoextend
```

这样在MySQL启动之后就会创建这两个512M大小的文件作为系统表空间，其中的autoextend表明这两个文件如果不够用会自动扩展data2文件的大小。

我们也可以把系统表空间对应的文件路径不配置到数据目录下，甚至可以配置到单独的磁盘分区上，涉及到的启动参数就是innodb\_data\_file\_path和innodb\_data\_home\_dir，具体的配置逻辑挺绕的，我们这就不多唠叨了，知道改哪个参数可以修改系统表空间对应的文件，有需要的时候到官方文档里一查就好了。

需要注意的一点是，在一个MySQL服务器中，系统表空间只有一份。从MySQL5.5.7到MySQL5.6.6之间的各个版本中，我们表中的数据都会被默认存储到这个 **系统表空间**。

### 独立表空间(file-per-table tablespace)

在MySQL5.6.6以及之后的版本中，InnoDB并不会默认的把各个表的数据存储到系统表空间中，而是为每一个表建立一个独立表空间，也就是说我们创建了多少个表，就有多少个独立表空间。使用独立表空间来存储表数据的话，会在该表所属数据库对应的子目录下创建一个表示该独立表空间的文件，文件名和表名相同，只不过添加了一个.ibd的扩展名而已，所以完整的文件名称长这样：

```
表名.ibd
```

比方说假如我们使用了独立表空间去存储xiaohaizi数据库下的test表的话，那么在该表所在数据库对应的xiaohaizi目录下会为test表创建这两个文件：

```
test.frm  
test.ibd
```

其中test.ibd文件就用来存储test表中的数据和索引。当然我们也可以自己指定使用系统表空间还是独立表空间来存储数据，这个功能由启动参数innodb\_file\_per\_table控制，比如说我们想刻意将表数据都存储到系统表空间时，可以在启动MySQL服务器的时候这样配置：

```
[server]  
innodb_file_per_table=0
```

当innodb\_file\_per\_table的值为0时，代表使用系统表空间；当innodb\_file\_per\_table的值为1时，代表使用独立表空间。不过innodb\_file\_per\_table参数只对新建的表起作用，对于已经分配了表空间的表并不起作用。如果我们想把已经存在系统表空间中的表转移到独立表空间，可以使用下边的语法：

```
ALTER TABLE 表名 TABLESPACE [=]  
innodb_file_per_table;
```

或者把已经存在独立表空间的表转移到系统表空间，可以使用下边的语法：

```
ALTER TABLE 表名 TABLESPACE [=] innodb_system;
```

其中中括号扩起来的=可有可无，比方说我们想把test表从独立表空间移动到系统表空间，可以这么写：

```
ALTER TABLE test TABLESPACE innodb_system;
```

其他类型的表空间



随着MySQL的发展，除了上述两种老牌表空间之外，现在还新提出了一些不同类型的表空间，比如通用表空间（general tablespace）、undo表空间（undo tablespace）、临时表空间（temporary tablespace）吧啦吧啦的，具体情况我们就不细唠叨了，等用到的时候再提。

## MyISAM是如何存储表数据的

好了，唠叨完了InnoDB的系统表空间和独立表空间，现在轮到MyISAM了。我们知道不像InnoDB的索引和数据是一个东东，在MyISAM中的索引全部都是二级索引，该存储引擎的数据和索引是分开存放的。所以在文件系统中也是使用不同的文件来存储数据文件和索引文件。而且和InnoDB不同的是，MyISAM并没有什么所谓的表空间一说，**表数据都存放对应的数据库子目录下**。假如test表使用MyISAM存储引擎的话，那么在它所在数据库对应的xiaohaizi目录下会为test表创建这三个文件：

```
test.frm
test.MYD
test.MYI
```

其中test.MYD代表表的数据文件，也就是我们插入的用户记录；test.MYI代表表的索引文件，我们为表创建的索引都会放到这个文件中。

## 视图在文件系统中的表示

我们知道MySQL中的视图其实是虚拟的表，也就是某个查询语句的一个别名而已，所以在存储视图的时候是不需要存储真实的数据的，**只需要它的结构存储起来就行了**。和表一样，描述视图结构的文件也会被存储到所属数据库对应的子目录下边，只会存储一个视图名.frm的文件。

## 其他的文件

除了我们上边说的这些用户自己存储的数据以外，数据目录下还包括为了更好运行程序的一些额外文件，主要包括这几种类型的文件：

- 服务器进程文件。

我们知道每运行一个MySQL服务器程序，都意味着启动一个进程。MySQL服务器会把自己的进程ID写入到一个文件中。

- 服务器日志文件。

在服务器运行过程中，会产生各种各样的日志，比如常规的查询日志、错误日志、二进制日志、redo日志吧啦吧啦各种日志，这些日志各有各的用途，我们之后会重点唠叨各种日志的用途，现在先了解一下就可以了。

- 默认/自动生成的SSL和RSA证书和密钥文件。

主要是为了客户端和服务端安全通信而创建的一些文件，大家看不懂可以忽略～

## 文件系统对数据库的影响

因为MySQL的数据都是存在文件系统中的，就不得不受到文件系统的一些制约，这在数据库和表的命名、表的大小和性能方面体现的非常明显，比如下边这些方面：

- 数据库名称和表名称不得超过文件系统所允许的最大长度。

每个数据库都对应数据目录的一个子目录，数据库名称就是这个子目录的名称；每个表都会在数据库子目录下产生一个和表名同名的.frm文件，如果是InnoDB的独立表空间或者使

用MyISAM引擎还会有别的文件的名称与表名一致。这些目录或文件名的长度都受限于文件系统所允许的长度～

- 特殊字符的问题

为了避免因为数据库名和表名出现某些特殊字符而造成文件系统不支持的情况，MySQL会把数据库名和表名中所有除数字和拉丁字母以外的所有字符在文件名里都映射成 @+编码值的形式作为文件名。比方说我们创建的表的名称为'test?'，由于?不属于数字或者拉丁字母，所以会被映射成编码值，所以这个表对应的.frm文件的名称就变成了test@003f.frm。

- 文件长度受文件系统最大长度限制

对于InnoDB的独立表空间来说，每个表的数据都会被存储到一个与表名同名的.ibd文件中；对于MyISAM存储引擎来说，数据和索引会分别存放到与表同名的.MYD和.MYI文件中。这些文件会随着表中记录的增加而增大，它们的大小受限于文件系统支持的最大文件大小。

## MySQL系统数据库简介

我们前边提到了MySQL的几个系统数据库，这几个数据库包含了MySQL服务器运行过程中所需的一些信息以及一些运行状态信息，我们现在稍微了解一下。

- mysql

这个数据库是核心，它存储了MySQL的用户账户和权限信息，一些存储过程、事件的定义信息，一些运行过程中产生的日志信息，一些帮助信息以及时区信息等。

- information\_schema

这个数据库保存着MySQL服务器维护的所有其他数据库的信息，比如有哪些表、哪些视图、哪些触发器、哪些列、哪些索引吧啦吧啦。这些信息并不是真实的用户数据，而是一些描述性信息，有时候也称之为元数据。

- performance\_schema

这个数据库里主要保存MySQL服务器运行过程中的一些状态信息，算是对MySQL服务器的一个性能监控。包括统计最近执行了哪些语句，在执行过程的每个阶段都花费了多长时间，内存的使用情况等等信息。

- sys

这个数据库主要是通过视图的形式把information\_schema和performance\_schema结合起来，让程序员可以更方便的了解MySQL服务器的一些性能信息。

啥？这四个系统数据库这就介绍完了？是的，我们的标题写的就是简介嘛！如果真的要唠叨一下这几个系统库的使用，那怕是又要写一本书了... 这里只是因为介绍数据目录里遇到了，为了内容的完整性跟大家提一下，具体如何使用还是要参照文档～

## 总结

1. 对于InnoDB、MyISAM这样的存储引擎会把数据存储到文件系统上。
2. 数据目录和安装目录是两个东西！
3. 查看数据目录位置的两个方式：
  - 服务器未启动时（类Linux操作系统）：

```
mysqld --verbose --help | grep datadir
```

- 服务器启动后：

```
SHOW VARIABLES LIKE 'datadir';
```

4. 每个数据库都对应数据目录下的一个子目录。

5. 表在文件系统上表示分两部分

- 表结构的定义

不论是InnoDB还是MyISAM，都会在数据库子目录下创建一个和表名同名的.frm文件。

- 表中的数据

针对InnoDB和MyISAM对于表数据有不同的存储方式。

6. 对于InnoDB存储引擎来说，使用表空间来存储表中的数据，表空间分两种类型：

- 系统表空间

默认情况下，InnoDB将所有的表数据都存储到这个系统表空间内，它是一个抽象的概念，实际可以对应着文件系统中若干个真实文件。

- 独立表空间

如果有需要的话，可以为每个表分配独立的表空间，只需要在启动服务器的时候将innodb\_file\_per\_table参数设置为1即可。每个表的独立表空间对应的文件系统中的文件是在数据库子目录下的与表名同名的.ibd文件。

7. 由于MySQL中的数据实际存储在文件系统上，所以会收到文件系统的一些制约：

- 数据库名称和表名称不得超过文件系统所允许的最大长度。
- 会把数据库名和表名中所有除数字和拉丁字母以外的所有字符在文件名里都映射成 @+编码值的形式作为文件名。
- 文件长度受文件系统最大长度限制。
- 如果同时访问的表的数量非常多，可能会受到文件系统的文件描述符有限的影响。