基础概念：

备份，将当前已有的数据或记录另存一份；

还原，将数据恢复到备份时的状态。

为什么要进行数据的备份与还原？

防止数据丢失；

保护数据记录。

数据备份与还原的方式有很多种，具体可以分为：数据表备份、单表数据备份、SQL备份和增量备份。

数据表备份

数据表备份，不需要通过 SQL 来备份，我们可以直接进入到数据库文件夹复制对应的表结构以及数据；在需要还原数据的时候，直接将备份（复制）的内容放回去即可。

不过想要进行数据表备份是有前提条件的，因为不同的存储引擎之间是有区别的。

对于存储引擎，MySQL 主要使用两种，分别为：InnoDB 和 Myisam，两者均免费。在这里，咱们可以顺便科普一下存储引擎的知识：

| 特点 | Myisam | InnoDB | BDB | Memory | Archive |
| --- | --- | --- | --- | --- | --- |
| 批量插入的速度 | 高 | 低 | 高 | 高 | 非常高 |
| 事务安全 | —— | 支持 | 支持 | —— | —— |
| 全文索引 | 支持 | 5.5版本支持 | —— | —— | —— |
| 锁机制 | 表锁 | 行锁 | 页锁 | 表锁 | 行锁 |
| 存储限制 | 没有 | 64TB | 没有 | 有 | 没有 |
| B树索引 | 支持 | 支持 | 支持 | 支持 | —— |
| 哈希索引 | —— | 支持 | —— | 支持 | —— |
| 集群索引 | —— | 支持 | —— | —— | —— |
| 数据缓存 | —— | 支持 | —— | 支持 | —— |
| 索引缓存 | 支持 | 支持 | —— | 支持 | —— |
| 数据可压缩 | 支持 | —— | —— | —— | 支持 |
| 空间使用 | 低 | 高 | 低 | N/A | 非常低 |
| 内存使用 | 低 | 高 | 低 | 中等 | 低 |
| 外键支持 | —— | 支持 | —— | —— | —— |

其中，Myisam 和 InnoDB 的数据存储方法也有所区别：

Myisam：表、数据和索引全部单独分开存储；

InnoDB：只有表结构，数据全部存储到ibd文件中。

执行如下 SQL 语句，测试 Myisam 的数据存储方式：

-- 创建 Myisam 表

create table my\_myisam(

id int

)charset utf8 engine = myisam;

-- 显示表结构

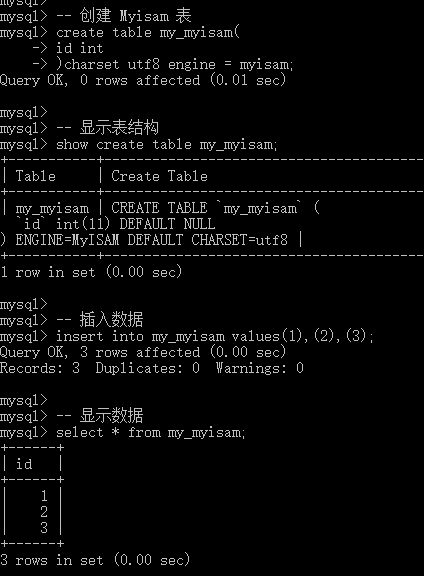
show create table my\_myisam;

-- 插入数据

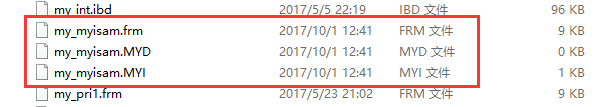
insert into my\_myisam values(1),(2),(3);

-- 显示数据

select \* from my\_myisam;

[](https://camo.githubusercontent.com/5b19f55fe28343ba950a0a147c33d4b52c838058/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313234323131373931)

如上图所示，我们创建了名为my\_myisam，存储引擎为 Myisam 的数据表。为了验证 Myisam 的存储特性，我们可以到data文件夹查看具体的数据存储情况：

[](https://camo.githubusercontent.com/1e0dfe86b8ab15d00fa2acd135c3add25501b94b/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313235303434383137)

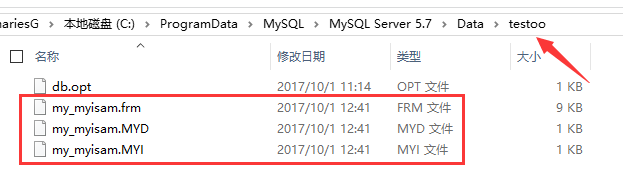
如上图所示，我们仅仅创建了一个表my\_myisam，但是 Myisam 对于会生成三个存储文件，分别为：

my\_myisam.frm：存储表的结构；

my\_myisam.MYD：存储表的数据；

my\_myisam.MYI：存储表的索引。

现在，我们将这三个文件复制到testoo数据库（至于如何找到 MySQL 数据文件的存储位置，可以参考[详述查看 MySQL 数据文件存储位置的方法](https://github.com/guobinhit/mysql-tutorial/blob/master/articles/datafile.md)）：

[](https://camo.githubusercontent.com/eba8ccaba54d463af93d5031cab4645b6c95bb4b/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313235383433383435)

执行如下 SQL 语句，进行测试：

-- 切换数据库

use testoo;

-- 查看 testoo 数据库中的表

show tables;

-- 查看表 my\_myisam

select \* from my\_myisam;

[](https://camo.githubusercontent.com/4475cd262fc2d63802d42315d2f9a8ae9bfe1678/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313235373233333037)

如上图所示，显然我们已经通过复制文件的方式，完成了数据表的备份工作。

在这里，有一点需要我们注意，那就是：我们可以将通过 InnoDB 存储引擎产生的.frm和.idb文件复制到另一个数据库，也可以通过show tables命令查看复制过来的表名称，但是却无法获得数据。

[](https://camo.githubusercontent.com/617456d55726678a52271060ff503184c964a81f/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313330353239343831)

执行如下 SQL 语句，进行测试：

-- 查看 testoo 数据库中的表

show tables;

-- 查看表 my\_class

select \* from my\_class;

[](https://camo.githubusercontent.com/1115a1b6ce5305a79e6c6d8fbb1dad4c5545b1e4/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313330363433333135)

通过以上测试，显然数据表备份这种备份方式更适用于 Myisam 存储引擎，而且备份的方式也很简单，直接复制 Myisam 存储引擎产生的.frm、.MYD和.MYI三个存储文件到新的数据库即可。

单表数据备份

单表数据备份，每次只能备份一张表，而且只能备份数据，不能备份表结构。

通常的使用场景为：将表中的数据导出到文件。

备份方法：从表中选出一部分数据保存到外部的文件中，

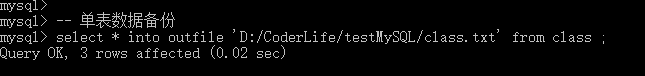
select \*/字段列表 + into outfile + '文件存储路径' + from 数据源;

在这里，使用单表数据备份有一个前提，那就是：导出的外部文件不存在，即文件存储路径下的文件不存在。

执行如下 SQL 语句，进行测试：

-- 单表数据备份

select \* into outfile 'D:/CoderLife/testMySQL/class.txt' from class;

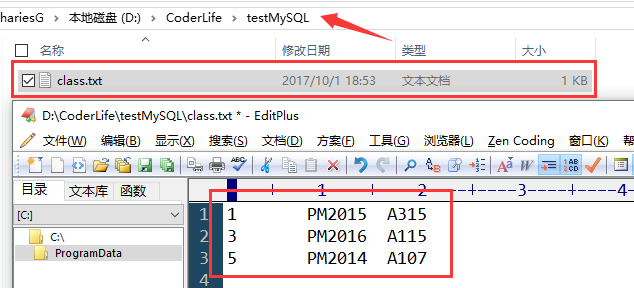
[](https://camo.githubusercontent.com/2aadbdd1ded9d46803956e2c29d6818b1f333c6c/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313835323030383537)

如上图所示，SQL 语句已经执行成功。在这里，如果我们遇到：

ERROR 1290 (HY000): The MySQL server is running with the –secure-file-priv option so it cannot execute this statement.

这个错误，可以通过查看「[详述 MySQL 导出数据遇到 secure-file-priv 的问题](https://github.com/guobinhit/mysql-tutorial/blob/master/articles/secure.md)」进行解决。

为了验证是否真的将class表中的数据导出到指定位置，我们可以到该路径下进行确认：

[](https://camo.githubusercontent.com/9b2af5ee6f80052cfa60e80a2637c6bfeaf88040/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313835363337313630)

如上图所示，显然我们已经将class表中的数据导出到本地啦！不过在这里，有一点需要我们特别注意，那就是：对于从数据库导出的文件，我们最好用EditPlus等编辑工具打开，防止乱码。

此外，对于上述用于导出表中数据的 SQL 语法，其实我们可以颠倒书写顺序，也没有问题，例如：

select \*/字段列表 + from 数据源 + into outfile + '文件存储路径';

执行如下 SQL 语句，进行测试：

-- 单表数据备份

select \* from class into outfile 'D:/CoderLife/testMySQL/class2.txt';

[](https://camo.githubusercontent.com/42268a584a60a89bb91f3f43e0c1650abdf63b50/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313930353036323232)

接下来，我们学习一些用于单表数据备份的高级操作，即自己指定字段和行的处理方式。

基本语法：select \*/字段列表 + into outfile + '文件存储路径' + fields + 字段处理 + lines + 行处理 + from 数据源;

字段处理：

enclosed by：指定字段用什么内容包裹，默认是，空字符串；

terminated by：指定字段以什么结束，默认是\t，Tab键；

escaped by：指定特殊符号用什么方式处理，默认是\\，反斜线转义。

行处理：

starting by：指定每行以什么开始，默认是，空字符串；

terminated by：指定每行以什么结束，默认是\r\n，换行符。

执行如下 SQL 语句，进行测试：

-- 指定单表数据备份处理方式

select \* into outfile 'D:/CoderLife/testMySQL/class3.txt'

-- 字段处理

fields

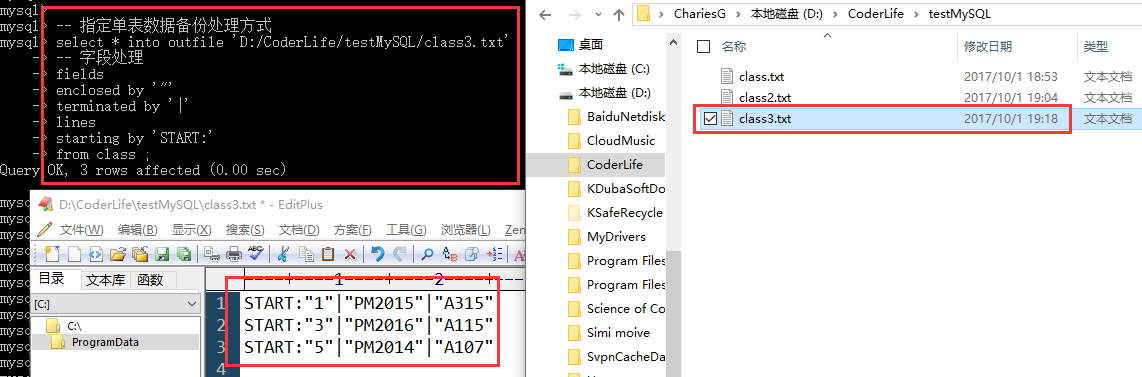
enclosed by '"'

terminated by '|'

lines

starting by 'START:'

from class ;

[](https://camo.githubusercontent.com/cf2b2b48222786c359bca97ef22dd87029b3d464/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313932303331343338)

如上图所示，显然导出文件class3.txt按照我们指定的格式进行输出啦！在前面，我们已经测试了各种单表数据备份的方式，现在我们删除数据，并尝试还原数据，即将保持在外部的数据重新恢复到数据表中。But，由于单表数据备份仅能备份数据，因此如果表结构不存在，则不能进行还原。

基本语法：load data infile + '文件存储路径' + into table + 表名 + [字段列表] + fields + 字段处理 + lines + 行处理;

执行如下 SQL 语句，进行测试：

-- 删除表 class 中的数据

delete from class;

-- 查看表 class 中的数据

select \* from class;

-- 还原表 class 中的数据

load data infile 'D:/CoderLife/testMySQL/class3.txt'

into table class

-- 字段处理

fields

enclosed by '"'

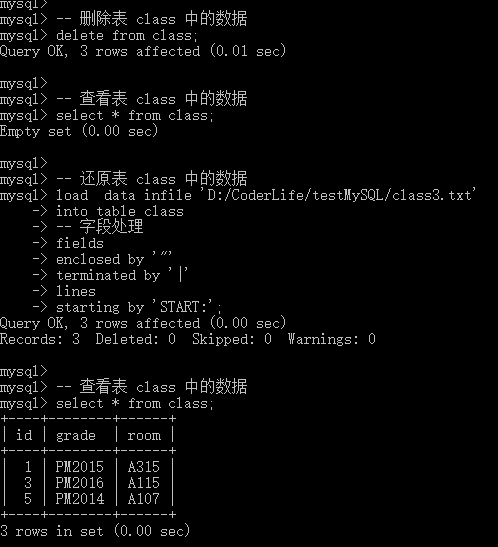
terminated by '|'

lines

starting by 'START:';

-- 查看表 class 中的数据

select \* from class;

[](https://camo.githubusercontent.com/cc723a27f2569b46c0b3dc69d83bd7fd25bba705/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303031313933343435373238)

如上图所示，显然在我们删除表class中的数据之后，还原数据成功。

SQL 备份

SQL 备份，备份的是 SQL 语句。在进行 SQL 备份的时候，系统会对表结构以及数据进行处理，变成相应的 SQL 语句，然后执行备份。在还原的时候，只要执行备份的 SQL 语句即可，此种备份方式主要是针对表结构。

不过，MySQL 并没有提供 SQL 备份的指令，如果我们想要进行 SQL 备份，则需要利用 MySQL 提供的软件mysqldump.exe，而且mysqldump.exe也是一种客户端，因此在操作服务器的时候，必须进行认证。

基本语法：mysqldump.exe -hPup + 数据库名字 + [表名1 + [表名2]] > 备份文件目录

其中，-hPup分别表示

h：IP 或者localhost；

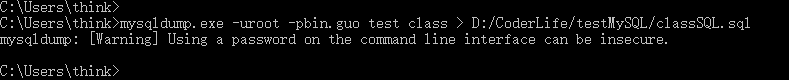
P：端口号；

u：用户名；

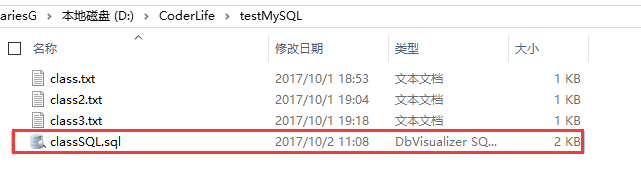
p：密码。

由于mysqldump.exe也是客户端，因此想执行上述命令，我们需要先退出 MySQL 客户端，然后在命令行窗口执行如下命令：

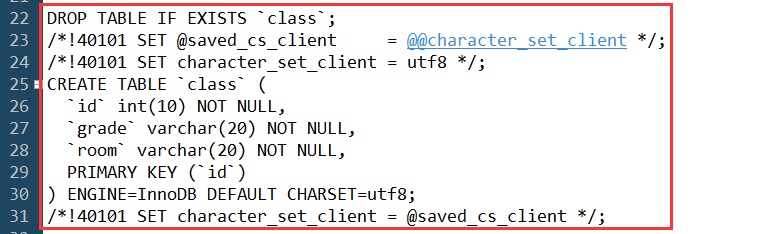
mysqldump.exe -uroot -pbin.guo test class > D:/CoderLife/testMySQL/classSQL.sql

[](https://camo.githubusercontent.com/6b287a31cf4901208ebe524f67011679ac2efd4c/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303032313131383431373634)

如上图所示，虽然mysqldump给出了警告（在命令行中输入密码是不安全的），但是我们输入的命令已经成功执行啦！在这里，如果执行上述命令不成功的话，很有可能是我们没有配置环境变量的问题。

[](https://camo.githubusercontent.com/8427ba3bbbe29f2e181d0160865e01b47783cd8c/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303032313132303234313437)

如上图所示，在testMySQL目录下，已经生产了对表class的 SQL 备份，至于 SQL 备份的内容到底是什么，我们可以打开classSQL.sql文件进行查看：

[](https://camo.githubusercontent.com/d8f060a136ce0c248ee9753c4b628ecc449c04c6/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303032313132333030383730)

如上图所示，通过 SQL 备份的文件，包含了各种 SQL 语句，如创建表的语句以及插入数据的语句等等。

此外，在上面给出的执行 SQL 备份的基本语法中，我们可以看到表名都用[]括了起来，这表示可选项，如果不输入表名，则默认备份整个数据库。执行过程和上面一样，因此我们就不予演示啦！

接下来，我们演示通过 SQL 备份的文件还原数据，有两种方式：

方式 1：使用mysql.exe客户端还原数据

基本语法mysql.exe/mysql -hPup 数据库名称 + 数据库名字 + [表名1 + [表名2]] < 备份文件目录

在命令行窗口执行如下命令，进行测试：

-- 登录 MySQL 客户端

mysql -uroot -p

-- 输入密码，切换数据库

use test;

-- 删除表 class 中的数据

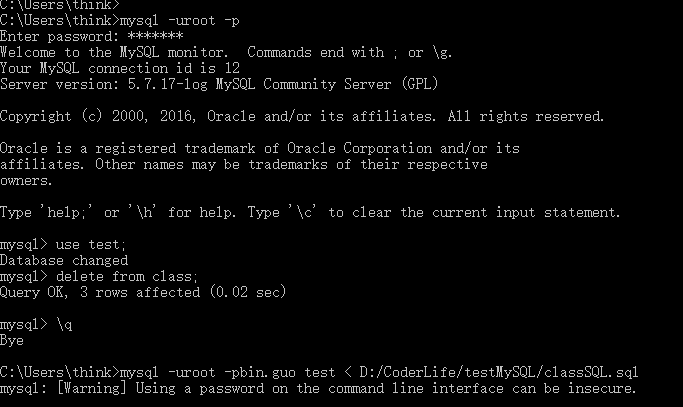
delete from class;

-- 退出数据库

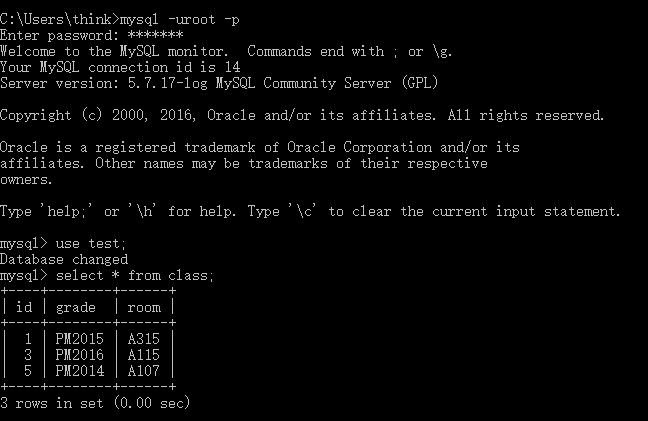
\q

-- 通过 SQL 备份的文件还原数据

mysql -uroot -pbin.guo test < D:/CoderLife/testMySQL/classSQL.sql

[](https://camo.githubusercontent.com/e730ee91e2df0f5a635fe9fc7761616d1d8c2d07/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303032313133393339363835)

如上图所示，上述命令全部执行成功。下面，我们检查还原结果，

[](https://camo.githubusercontent.com/93e7e28c77c12e238213bb6ada3e87cd1245e6cf/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303032313134303330333930)

如上图所示，显然表class的数据在删除之后，我们通过 SQL 备份的文件还原了数据。

方式 2：使用 SQL 命令还原数据

基本语法source + 备份文件目录;

执行如上 SQL 语句，进行测试：

-- 查看表 class 数据

select \* from class;

-- 删除表 class 数据

delete from class;

-- 查看表 class 数据

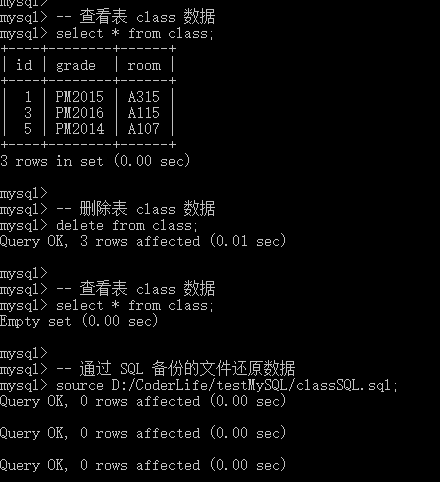
select \* from class;

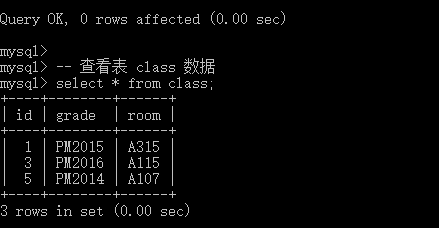
-- 通过 SQL 备份的文件还原数据

source D:/CoderLife/testMySQL/classSQL.sql;

-- 查看表 class 数据

select \* from class;

[](https://camo.githubusercontent.com/63de483a45e029f010c95b194a9b736b74bb8346/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303032313134383037323833)

[](https://camo.githubusercontent.com/e39ec8849ea2d96f91d55feaf62f72ad3cd13928/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731303032313135303032363431)

如上图所示，显然表class的数据在删除之后，我们通过 SQL 备份的文件利用第二种方式还原了数据。

通过上面的学习及测试，我们可以知道 SQL 备份的优缺点：

优点：可以备份表结构；

缺点：增加额外的 SQL 命令，会浪费磁盘空间。

增量备份

增量备份，不是针对数据或者 SQL 进行备份，而是针对 MySQL 服务器的日志进行备份，其日志内容包括了我们对数据库的各种操作的历史记录，如增删改查等。此外，增量备份是指定时间段进行备份，因此备份的数据一般不会出现重复的情况，常用于大型项目的数据备份。在此，我们就不详细的进行介绍了，至于这部分的内容，以后会单独写一篇关于如何进行增量备份的博文。