

# MySQL 规范

# 目录

1

单机规范

2

集群规范

# 单机规范 - 库名和表名全小写 + 无中文

1. WHY

2. `lower_case_table_names` 默认值 0

3. MySQL 8.0 以后只允许在初始化的时候修改

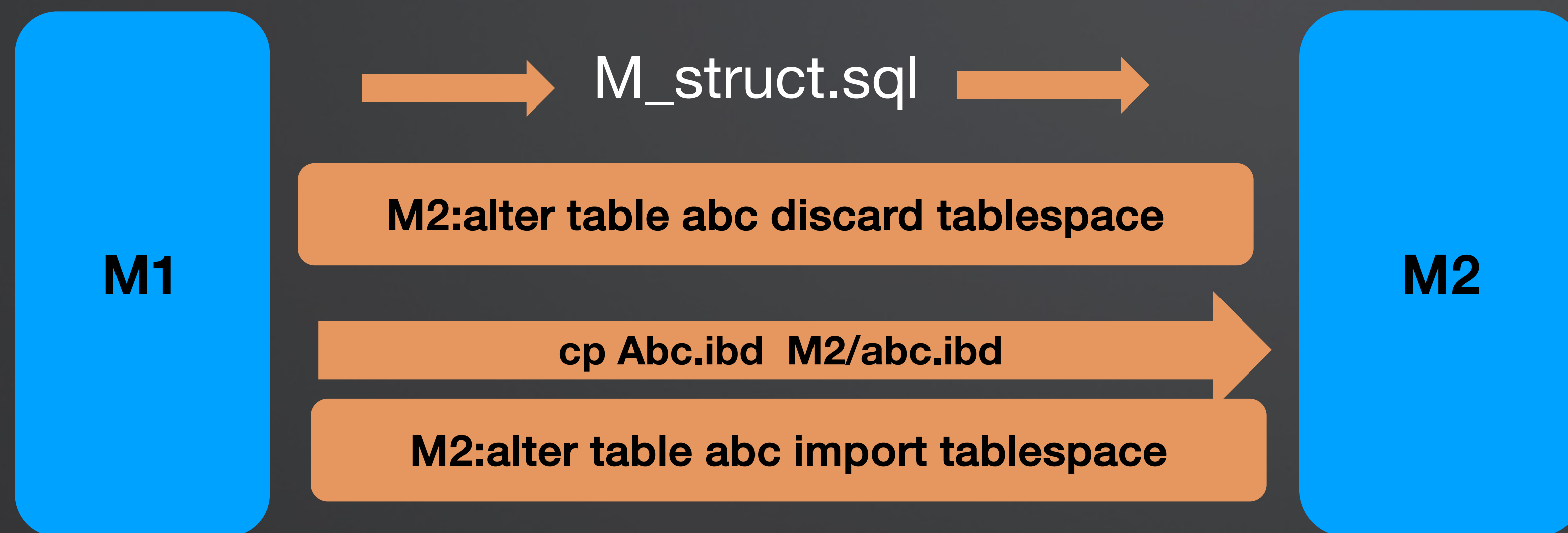
# 讨论：运行一段时间后怎么修改 lower\_case\_table\_names

1. 创建一个空实例 M2, lower\_case\_table\_names = 1
2. mysqldump from M1
3. restore to M2



# 讨论：运行一段时间后怎么修改 lower\_case\_table\_names

1. 创建一个空实例 M2, lower\_case\_table\_names = 1
2. mysqldump 表结构,source 到 M2
3. 利用可传输表空间 import



# 讨论：运行一段时间后怎么修改lower\_case\_table\_names

```
mysql> create table TTT(id int);
Query OK, 0 rows affected (0.08 sec)

mysql> select * from ttt;
ERROR 1146 (42S02): Table 'test.ttt' doesn't exist
```

1. 停机维护时间开始

2. 把这个实例里的所有表名改成纯小写,如 rename table TtT to ttt;

```
mysql> rename table TTT to ttt;
Query OK, 0 rows affected (0.07 sec)
```

3. shutdown 并修改 my.cnf , 增加 lower\_case\_table\_names=1

4. 修改 mysql.ibd 中的 LCTN 的值为1

```
python3 modify_lower_case_table_names.py $datadir/mysql.ibd /tmp/mysql.ibd 1
```

```
cp /tmp/mysql.ibd $datadir/mysql.ibd
```

5. 重新启动 mysqld

```
mysql> select * from TTT;
Empty set (0.00 sec)

mysql> create table TTT(id int);
ERROR 1050 (42S01): Table 'ttt' already exists
```

<https://github.com/ddcw/ibd2sql/archive/refs/heads/main.zip>



# 单机规范 - 注释尽量不要用中文字符

WHY

课堂练习: 建表的时候用 GBK, 但是表中 varchar 定义是utf-8

1. 写入数据的时候是应该写 utf-8 还是 GBK
2. mysqldump的时候用 utf-8 还是 GBK
3. 会不会导致建表语句无法导出

## 相关源码 - mysqldump.cc

```
snprintf(buff, sizeof(buff), "show create table %s", result_table);

if (switch_character_set_results(mysql, "binary") ||
    mysql_query_with_error_report(mysql, &result, buff) ||
    switch_character_set_results(mysql, default_charset))
    return 0;
```



# 课堂练习

```
mysql> create table sample(a varchar(32) DEFAULT NULL COMMENT '中' );
```

```
Query OK, 0 rows affected (0.12 sec)
```

```
mysql> insert into sample values('中');
```

```
Query OK, 1 row affected (0.01 sec)
```

```
mysqldump -uroot -h -P --default-character-set=latin1 test sample
```

输出结果中,建表语句的“中”和表数据的“中”, 以下说法正确是:

- A. 都不是乱码
- B. 都是乱码
- C. 建表语句正常, 数据里的乱码
- D. 建表语句乱码, 数据里的正常

# 单机规范 - 注释尽量不要用中文字符

会导致建表语句无法导入!

```
COMMENT ' 主键 ' PRIMARY KEY (`id`)
```

讨论：怎么处理？

# 课堂练习：找 bug

```
03025:     }
03026:   else
03027:   {
03028:     const char *comment_keyword = "COMMENT";
03029:     char *output_sql = (char *)calloc(strlen(row[1]) + 1, sizeof(char));
03030:     if (!output_sql) {
03031:       fprintf(stderr, "Memory allocation failed\n");
03032:       exit(-1);
03033:     }
03034:     int output_index = 0;
03035:     int skip_next = 0;
03036:     for (int i = 0; i < strlen(row[1]); i++) {
03037:       if (strncmp(&row[1][i], comment_keyword, strlen(comment_keyword)) == 0) {
03038:         if (row[1][i-1] != 0x60) { // 反向匹配 ` 字符
03039:           skip_next = 1; // 找到COMMENT, 设置跳过标记
03040:           i += strlen(comment_keyword); // 跳过COMMENT
03041:           // 跳过comment后面的一个空格或者等号
03042:           while (row[1][i] == 0x20 || row[1][i] == 0x3D) {
03043:             i++;
03044:           }
03045:           continue; // 继续循环
03046:         }
03047:       }
03048:
03049:       // 如果设置了跳过标记, 跳过COMMENT后的内容
03050:       if (skip_next) {
03051:         // 遇到下一个分词符号, 分词符号可以是逗号/换行/空格
03052:         if (row[1][i] == 0x20 || row[1][i] == 0x0A || row[1][i] == 0x2C) {
03053:           output_sql[output_index++] = row[1][i]; // 需要复制
03054:           skip_next = 0; // 重置跳过标记
03055:         }
03056:         continue; // 继续跳过
03057:       }
03058:
03059:       // 复制当前字符到输出
03060:       output_sql[output_index++] = row[1][i];
03061:     } ? end for inti=0;i<strlen(row[1... ?
03062:
03063:     output_sql[output_index] = '\0'; // 结束字符串
03064:
03065:
03066:     fprintf(sql_file,
03067:             "/*!40101 SET @saved_cs_client      = @@character_set_client *;\n"
03068:             "/*!40101 SET character_set_client = utf8 *;\n"
03069:             "%s%s;\n"
03070:             "/*!40101 SET character_set_client = @saved_cs_client *;\n",
03071:             (is_log_table || is_replication_metadata_table) ?
```



## 课堂练习：看代码分析输出

```
snprintf(buff, sizeof(buff), "show create table %s", result_table);

if (switch_character_set_results(mysql, "latin1") ||
    mysql_query_with_error_report(mysql, &result, buff) ||
    switch_character_set_results(mysql, default_charset))
    return 0;
```

```
CREATE TABLE `sample` (
  `a` varchar(32) DEFAULT NULL COMMENT '?\''?'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

不会出现 ' 转 latin 时被转成其他字符的情况，因为其他情况乱码是由于编码方式不对导致的，以前可能是对两个字节编码，然后改成对三个或者四个字节编码，这样就可能出现 ' 跟前面的字符的字节一起被转码的情况，但是 latin 就是一个字节一个字节的转码，因此不会出现这个问题。

# 单机规范 - not NULL

每个字段都要声明成not null,并有默认值

WHY

# 讨论: insert怎么写

```
insert into t(id,a,b) values(...)
```

```
insert into t values(...)
```

变更加字段

```
alter table sample add t int INVISIBLE;
```



# 单机规范 - not NULL

每个字段都要声明成not null,并有默认值

1. insert 时应该明确指定列
2. 对于未指定的列, 可以使用默认值

```
insert into ()() on duplicate update set  
c1=value(c1)
```

```
replace into (..)(..)(..)
```

# 单机规范 - PK

1. bigint unsigned auto\_increment
2. 业务不依赖

# 单机规范 - PK

课堂练习: begin; insert into t(id) values(null); rollback;

自增 id 会改变吗?

```
mysql> show create table t2;
+-----+-----+
| Table | Create Table |
+-----+-----+
| t2    | CREATE TABLE `t2` (
  `id` int NOT NULL AUTO_INCREMENT,
  `e` int DEFAULT NULL,
  `f` int DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CH. |
+-----+-----+
1 row in set (0.00 sec)

mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t2(id) values(null);
Query OK, 1 row affected (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> show create table t2;
```

# 单机规范 - 所有表建议使用 InnoDB 存储引擎

WHY

# 单机规范 - 使用 `super_read_only` 代替 `read_only`

WHY

# 单机规范 - 尽量不要使用视图/禁止使用触发器

WHY



课堂练习: mysqldump 同一个库里的表, 是按照什么顺序生成建表语句的?

```
show tables;
```

课堂练习: 如果出现前面的表外键依赖后面的表, 怎么解决导入错误的问题?

```
SET FOREIGN_KEY_CHECKS=0
```

课堂练习: 如果出现前面的视图依赖后面的表, 怎么解决导入错误的问题?

## 例子

```
create view_a as select * from z where ...;
```

# 视图依赖

## 1. 循环过程中先创建一个假的视图

```
/*!50001 CREATE VIEW `view_a` AS SELECT  
1 AS `id`,  
1 AS `c`,  
1 AS `d`*/;
```

## 2. 到最后,再创建真实的图

```
/*!50001 DROP VIEW IF EXISTS `view_a`*/;  
/*!50001 SET @saved_cs_client      = @@character_set_client */;  
/*!50001 SET @saved_cs_results    = @@character_set_results */;  
/*!50001 SET @saved_col_connection = @@collation_connection */;  
/*!50001 SET character_set_client  = utf8mb4 */;  
/*!50001 SET character_set_results = utf8mb4 */;  
/*!50001 SET collation_connection  = utf8mb4_0900_ai_ci */;  
/*!50001 CREATE ALGORITHM=UNDEFINED */  
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */  
/*!50001 VIEW `view_a` AS select `z`.`id` AS `id`,`z`.`c` AS `c`,`z`.`d` AS `d` from `z` */;  
/*!50001 SET character_set_client      = @saved_cs_client */;  
/*!50001 SET character_set_results    = @saved_cs_results */;  
/*!50001 SET collation_connection    = @saved_col_connection */;  
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```



# 单机规范 - 禁用float 和 double

## WHY

float:  $-3.402823466E+38$ 到 $1.175494351E-38$

改用decimal

课堂练习:

```
root@devops_db 10:56: [test]> select cast('9' as decimal(2,2)), cast('9' as decimal(3,2)) ;
+-----+-----+
| cast('9' as decimal(2,2)) | cast('9' as decimal(3,2)) |
+-----+-----+
|                0.99 |                9.00 |
+-----+-----+
```

# 单机规范 - 禁止在数据库中存储图片、文件等二进制数据

WHY

一般怎么做？



# 单机规范 - 尽量不使用 JSON 类型

WHY

一般怎么做？

# 单机规范 - 可以用text的地方尽量不使用blob

# 课堂练习:slow\_log表的语句字段为什么不可读?

```
mysql> select * from mysql.slow_log\G
***** 1. row *****
  start_time: 2024-12-08 08:06:03.483049
  user_host: root[root] @ [127.0.0.1]
  query_time: 00:00:02.004882
  lock_time: 00:00:00.000006
  rows_sent: 1
  rows_examined: 1
        db: test
last_insert_id: 0
  insert_id: 0
  server_id: 1
   sql_text: 0x73656C65637420736C656570283229
  thread_id: 76
1 row in set (0.00 sec)
```

# 原因:

```
mysql> show create table mysql.slow_log\G
***** 1. row *****
      Table: slow_log
Create Table: CREATE TABLE `slow_log` (
  `start_time` timestamp(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE CURRENT_TIMESTAMP(6),
  `user_host` mediumtext NOT NULL,
  `query_time` time(6) NOT NULL,
  `lock_time` time(6) NOT NULL,
  `rows_sent` int NOT NULL,
  `rows_examined` int NOT NULL,
  `db` varchar(200) NOT NULL,
  `last_insert_id` int NOT NULL,
  `insert_id` int NOT NULL,
  `server_id` int unsigned NOT NULL,
  `sql_text` mediumblob NOT NULL,
  `thread_id` bigint unsigned NOT NULL
) ENGINE=CSV DEFAULT CHARSET=utf8mb3 COMMENT='Slow log'
1 row in set (0.00 sec)
```

1. mysql.slow\_log 的 sql\_text 字段是 mediumblob 格式
2. 从 8.0.19 开始 MySQL 客户端 binary-as-hex 默认值是 on

# 处理&验证:

```
banji@debian-vm:~/mysql-8.0.40/runtime_output_directory$ ./mysql -uroot -h127.0.0.1 -P11717 test --binary-as-hex=off
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 107
Server version: 8.0.37-debug Source distribution

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select sql_text from mysql.slow_log;
+-----+
| sql_text          |
+-----+
| select sleep(2)    |
+-----+
1 row in set (0.00 sec)
```



# 处理&验证:

```
mysql> select convert(sql_text using utf8mb4) from mysql.slow_log;
+-----+
| convert(sql_text using utf8mb4) |
+-----+
| select sleep(2)                  |
+-----+
1 row in set (0.00 sec)
```

不停机分库分  
表

```
mysql> select cast(sql_text as char) from mysql.slow_log;
+-----+
| cast(sql_text as char) |
+-----+
| select sleep(2)        |
+-----+
1 row in set (0.01 sec)
```



# 处理&验证:

```
banji@debian-vm:~/mysql-8.0.40/runtime_output_directory$ echo "select sql_text from mysql.slow_log" | ./mysql -uroot -h127.0.0.1 -P11717 test  
sql_text  
select sleep(2)
```

# Q&A

THANKS