目录

主流关系数据库：MySQL

- 最新版本：`v1.0`
- 更新时间：`20210915`

简介

介绍目前最主流的关系型数据库MySQL。先对MySQL进行概览；再介绍如何安装MySQL；再介绍MySQL的使用方面的内容；其中包括mysql的命令行；创建MySQL的数据库和设计表时的相关内容，比如字段属性、借助于在线网站设计表等；以及MySQL的数据库的备份和恢复，包括专门的工具mysqldump等；以及数据库管理工具Sequel Pro、MySQL Workbench等；以及log日志相关；总结了MySQL的使用心得；包括阿里云RDS；整理Python中如何使用mysql，比如用库pymsql；最后给出参考文档。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/popular_rmdb_mysql: 主流关系数据库：MySQL

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 主流关系数据库：MySQL book.crifan.com
- 主流关系数据库：MySQL crifan.github.io

离线下载阅读

- 主流关系数据库：MySQL PDF
- 主流关系数据库：MySQL ePub
- 主流关系数据库：MySQL Mobi

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

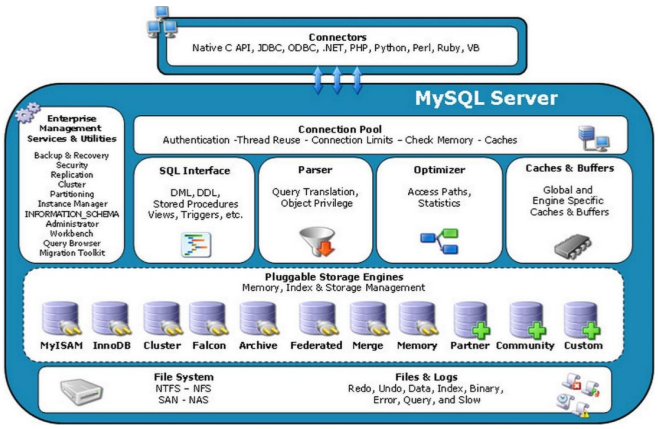
更多其他电子书

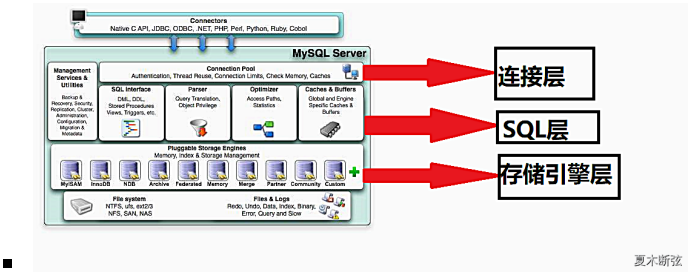
本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

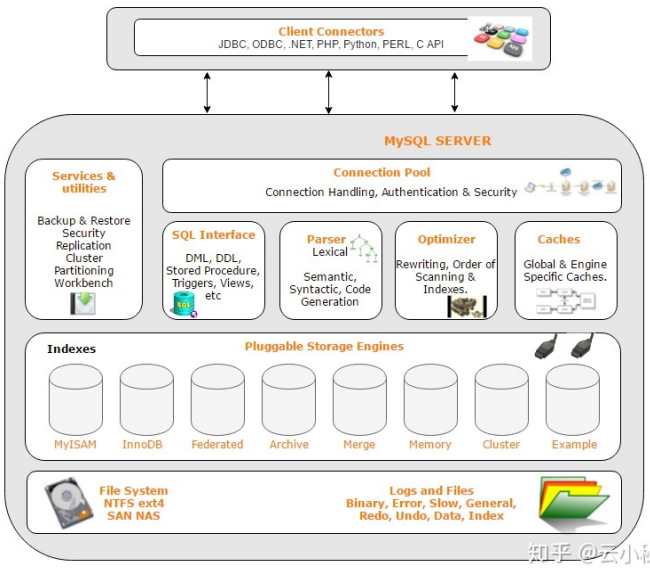
crifan/crifan_ebook_readme: Crifan的电子书的使用说明

MySQL概览

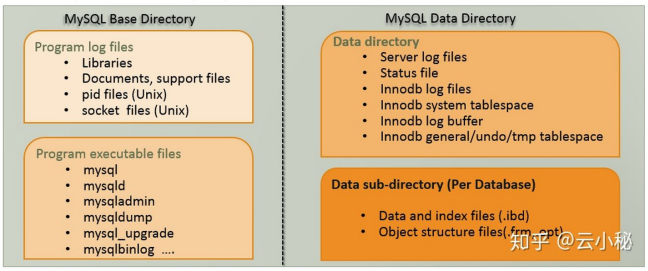
MySQL是目前最主流的关系型数据库。

- `MySQL`
  - 一句话描述：一种开源的关系型数据库管理系统
  - 官网
    - http://www.mysql.com
  - 体系架构
    - 

    - 

    - 

  - 细节
    - 默认端口：`3306`

- 配置文件：`my.cnf`
- 物理结构
  - 

- 文档
  - 主入口
    - MySQL :: MySQL Documentation
      - https://dev.mysql.com/doc/
  - 参考手册
    - MySQL :: MySQL 8.0 Reference Manual
      - https://dev.mysql.com/doc/refman/8.0/en/

安装MySQL

Mac安装MySQL

下载和安装mysql

先去mysql官网下载dmg安装文件，然后安装：

MySQL :: Download MySQL Community Server

->

MySQL :: Download MySQL Community Server

找到此处需要的：

- `mysql-5.7.22-macos10.13-x86_64.dmg`
  - 版本：macOS 10.13 (x86, 64-bit),
  - 格式：DMG Archive
  - 大小：341.1MB

下载后，双击dmg即可安装。

注意期间会有生成一个临时密码。

把mysql加到PATH环境变量

```
vi ~/.bashrc
```

把 `/usr/local/mysql/bin` 加进去：

```
export PATH=/usr/local/mysql/bin:$PATH
```

然后source使得立刻生效

```
source ~/.bashrc
```

通过which，确保能找到

```
which mysql
```

设置mysql密码

Mac中dmg安装的mysql默认不需要密码就可以登录的

所以此处最好去设置对应的密码

且如果用默认的临时密码，可能会导致其他工具无法正常连接

单独开个终端shell去：

```
sudo mysqld_safe --skip-grant-tables
```

然后另外再开个终端去：

```
use mysql;
update user set authentication_string=password('new_passwo
flush privileges;
exit
```

注意：`password('new_password` 中，是 `英文的单引号 = '` ，不是 `中文的单引号 = ‘`

-》我就是从别人网页中拷贝过来，结果发现是单引号，导致无法继续正常执行的。

然后就可以正常使用：

```
mysql -u root
```

去用新密码登录进去了。

设置密码不过期

为了防止其他mysql工具连接出错，还需要设置密码不过期，且重新再去设置一下新密码

```
mysql -u root -p
SET PASSWORD = PASSWORD('your_new_password');
use mysql;
update user set password_expired='N' where user='root';
flush privileges;
exit
```

确保mysql的服务已运行

后续记得用mysql之前，确保mysql（的server）是正常运行了的：

对于mysql的服务的管理是：

```
sudo /usr/local/mysql/support-files/mysql.server status
sudo /usr/local/mysql/support-files/mysql.server start
sudo /usr/local/mysql/support-files/mysql.server stop
sudo /usr/local/mysql/support-files/mysql.server restart
```

MySQL自身

MySQL命令行

用法举例

database=数据库

删除database数据库

```
drop database xxx;
```

table=表

查看当前已打开的表

```
show open tables;
```

查看当前已打开的，正在使用的表

```
show open tables where in_use>0;
```

查看表结构

```
describe table_name;
```

举例

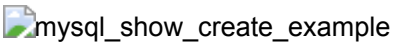
```
MySQL [xxx]> describe keyword;
+------------+-----------+------+-----+---------+----------
| Field      | Type      | Null | Key | Default | Extra
+------------+-----------+------+-----+---------+----------
| id         | int(11)   | NO   | PRI | NULL    | auto_incr
| name       | char(100) | NO   | MUL | NULL    |
| type       | char(20)  | NO   |     | NULL    |
| createTime | datetime  | YES  |     | NULL    |
| modifyTime | datetime  | YES  |     | NULL    |
+------------+-----------+------+-----+---------+----------
5 rows in set (0.00 sec)
```

查看已有表的创建create的命令

```
SHOW CREATE TABLE tbl_name
```

举例

```
MySQL [xxx]> SHOW CREATE TABLE keyword;
+---------+-------------------------------------------------
| Table   | Create Table
+---------+-------------------------------------------------
| keyword | CREATE TABLE `keyword` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` char(100) NOT NULL,
  `type` char(20) NOT NULL,
  `createTime` datetime DEFAULT NULL,
  `modifyTime` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `Multi` (`name`,`type`)
) ENGINE=InnoDB AUTO_INCREMENT=7529 DEFAULT CHARSET=utf8 |
+---------+-------------------------------------------------
1 row in set (0.00 sec)
```


mysql_show_create_example

record=记录=数据=值

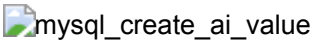
插入数据

对于自增ID的数据插入可以省略ID

对于表结构是：

```
CREATE TABLE `keyword` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` char(100) NOT NULL,
  `type` char(20) NOT NULL,
  `createTime` datetime DEFAULT NULL,
  `modifyTime` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `Multi` (`name`,`type`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

其中 `id 是 AUTO_INCREMENT`


mysql_create_ai_value

插入值时，可以省略ID：

```
INSERT INTO `keyword`(`name`,`type`,`createTime`,`modifyTi
```

更新数据

```
UPDATE keyword SET name="animal" WHERE type="sectorTopic"

UPDATE keyword SET name="ocean animal" WHERE type="topic"
```

删除特定数据

```
SELECT * FROM keyword WHERE type="topic" AND name="basketb
id=7426
DELETE FROM keyword_rel WHERE keyword2=7426;
```

删除多个数据

比如id在 `4` 到 `13` 之间，包括 `4` 和 `13`

```
DELETE FROM user WHERE id>=4 AND id<=13;
DELETE FROM user WHERE BETWEEN 4 AND 13;
```

mysql全局

查看当前进程

查看当前活跃的进程的列表：

```
show processlist;
```

举例

```
mysql> show processlist;
+----+------+-----------------+-----------+---------+------
| Id | User | Host            | db        | Command | Time
+----+------+-----------------+-----------+---------+------
|  2 | root | localhost       | NULL      | Query   |    0
| 28 | root | localhost:53199 | nxxxg     | Sleep   |  404 |
+----+------+-----------------+-----------+---------+------
2 rows in set (0.01 sec)
```

杀死进程

```
kill process_id;
```

查看当前的mysql最大连接

```
show variables like 'max_connections';
```

查看当前active活跃的连接数

```
show status where `variable_name` = 'Threads_connected';
```

查看字符编码

```
SHOW VARIABLES LIKE "character_set_server";

SHOW VARIABLES LIKE "collation_server";
```

创建数据库和表

注意搞清楚字符串编码

常见的是 `utf8` 或 `utf8mb4`

通过mysql的命令：

```
show CHARACTER SET;
```

可以列出各种编码：

```
mysql> show CHARACTER SET;
+----------+---------------------------------+-------------
| Charset  | Description                     | Default col
+----------+---------------------------------+-------------
| big5     | Big5 Traditional Chinese        | big5_chinese
| dec8     | DEC West European               | dec8_swedish
| cp850    | DOS West European               | cp850_genera
| hp8      | HP West European                | hp8_english_
| koi8r    | KOI8-R Relcom Russian           | koi8r_genera
| latin1   | cp1252 West European            | latin1_swedi
| latin2   | ISO 8859-2 Central European     | latin2_gener
| swe7     | 7bit Swedish                    | swe7_swedish
| ascii    | US ASCII                        | ascii_genera
| ujis     | EUC-JP Japanese                 | ujis_japanes
| sjis     | Shift-JIS Japanese              | sjis_japanes
| hebrew   | ISO 8859-8 Hebrew               | hebrew_gener
| tis620   | TIS620 Thai                     | tis620_thai_
| euckr    | EUC-KR Korean                   | euckr_korean
| koi8u    | KOI8-U Ukrainian                | koi8u_genera
| gb2312   | GB2312 Simplified Chinese       | gb2312_chine
| greek    | ISO 8859-7 Greek                | greek_genera
| cp1250   | Windows Central European        | cp1250_gener
| gbk      | GBK Simplified Chinese          | gbk_chinese_
| latin5   | ISO 8859-9 Turkish              | latin5_turki
| armscii8 | ARMSCII-8 Armenian              | armscii8_ger
| utf8     | UTF-8 Unicode                   | utf8_genera
| ucs2     | UCS-2 Unicode                   | ucs2_genera
| cp866    | DOS Russian                     | cp866_genera
| keybcs2  | DOS Kamenicky Czech-Slovak      | keybcs2_gene
| macce    | Mac Central European            | macce_genera
| macroman | Mac West European               | macroman_ger
| cp852    | DOS Central European            | cp852_genera
| latin7   | ISO 8859-13 Baltic              | latin7_gener
| utf8mb4  | UTF-8 Unicode                   | utf8mb4_gene
| cp1251   | Windows Cyrillic                | cp1251_gener
| utf16    | UTF-16 Unicode                  | utf16_genera
| utf16le  | UTF-16LE Unicode                | utf16le_gene
| cp1256   | Windows Arabic                  | cp1256_gener
| cp1257   | Windows Baltic                  | cp1257_gener
| utf32    | UTF-32 Unicode                  | utf32_genera
| binary   | Binary pseudo charset           | binary
| geostd8  | GEOSTD8 Georgian                | geostd8_gene
| cp932    | SJIS for Windows Japanese       | cp932_japane
| eucjpms  | UJIS for Windows Japanese       | eucjpms_japa
| gb18030  | China National Standard GB18030 | gb18030_chin
+----------+---------------------------------+-------------
41 rows in set (0.05 sec)
```

字段属性

在设计和查看数据库字段时

比如：

- 用 `MySQL Workbench` 设计表时
  -  mysql_workbench_design_table
- sequel中查看字段时
  -  sequel_check_field

会看到数据库字段的属性的缩写：

- PK
- NN
- UQ
- BIN
- UN
- ZF
- AI
- G

这部分都是通用的逻辑。

对应的含义是：

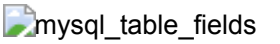
- `PK = Primary Key = 主键`
- `Unsigned` =无符号整数
- `ZF = Zerofill` =用0填充
- `BIN = Binary` =二级制
- `Allow Null` =允许空值
- `Key` =键
- `Default` =默认值
- `UQ` =值唯一
- `AI = Auto Increment` =自增

如何设计表

SQL数据库设计辅助工具

将数据库字段生成表格

想要得到这种：



可以借助于在线网站：

WWW SQL Designer

http://ondras.zarovi.cz/sql/demo/

备份和恢复

此处整理，MySQL数据库的备份和恢复。

mysql 导入/恢复

直接命令行中导入

mysql 导入sql 恢复mysql：

```
mysql -u user_name -p database_name < previous_exported_sq
```

然后输出密码，即可导入。

举例：

```
➜  mysql ll -h
-rw-r--r--@ 1 crifan  staff   5.1K 12 26 14:11 checkpoint_
➜  mysql mysql -u root -p xxx < checkpoint_20181226.sql
Enter password:
```

先进入mysql的命令行，再去导入

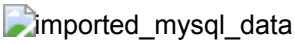
登录sql命令行，去source导入：

```
mysql -u username -p  database_name
```

进入后再 `source` ：

```
> source '/path/to/dumped/sql/file'
```

即可导入数据：



mysqlimport

- `mysqlimport`
  - 不能用来导入 `sql` ，只能导入 `txt` 文本
  - 语法

    ```
    mysqlimport  [OPTIONS]  database  textfile
    ```

- 举例

```
mysqlimport -u root -p --local database_name dump.t
```

注意事项

导入之前，先要创建一个空的数据库

- 方式1：登录sql命令行后去用命令新建

```
CREATE DATABASE database_name DEFAULT CHARACTER SET u
```

- 方式2：用工具去新建
  - `Sequel Pro`
    -  sequel_pro_add_database
    -  sequel_pro_new_db_info
- 才可以正常导入：

```
mysql -u root -p qpy < qpy.sql
Enter password:
```

- 否则会报错：

```
mysql -u root -p qpy < qpy.sql
Enter password:
ERROR 1049 (42000): Unknown database 'qpy'
```

mysqldump

`mysqldump`是MySQL的命令行工具，用于导出数据库，往往用于备份数据库的数据。

- 导出

```
mysqldump -u user_name -p database_name > dump_out_sql_
```

举例

导出数据库：

```
mysqldump -u root -p databaseName > databaseName_20190608.s
```

带压缩 + 整个database数据库：

```
mysqldump --host=xx-xxxx.mysql.rds.aliyuncs.com --port=3306
```

或-u的用户名和-p的密码的参数重去掉空格的写法：

```
mysqldump -h xxx.mysql.rds.aliyuncs.com -P 3306 -B xxx -uro
```

直接导出某个db的某个table表：

```
mysqldump -u user_name -p database_name table_name_in_db >
```

多个database：

```
mysqldump -uroot -proot --databases db1 db2 >/tmp/user.sql
```

举例：

```
mysqldump -h HOST_OR_IP -P 3306 xxx keyword -uroot -pYOUR_P
```

批量导出某个数据库database中多个table表的数据，且带创建table的sql的做法是：

```
mysqldump --host your_mysql_host --port 3306 -u user_name -
```

如果不想要创建table，则可以加上：`--no-create-info`

```
mysqldump --host your_mysql_host --port 3306 -u user_name -
```

log日志

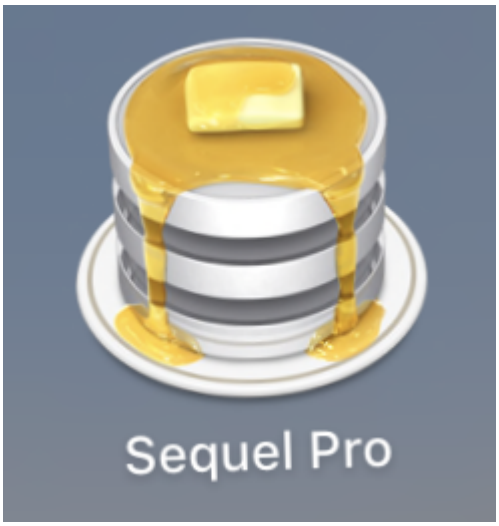
- 背景知识
  - 关系型数据库都有日志
  - MySQL也是关系型数据库的一种，也有日志
- MySQL的日志
  - MySQL中有很多种日志，其中两个是：
    - `Bin Log = 二进制日志`
      - MySQL服务端会把数据库内容的任何改动的事件都记录下来，保存到二进制的文件，即BinLog中。
    - `Slow Log = 慢查询日志`
      - 记录所有执行时间超过 `long_query_time` 秒的所有查询或不使用索引的查询
        - 可以使用 `mysqldumpslow` 命令获得日志中显示的查询摘要来处理慢查询日志
        - 主要用于性能分析，找出性能低，执行语句时间长的地方，便于后续优化，以提升性能
        - 涉及的参数的解释
          - `long_query_time` ：设定慢查询的阀值，超出次设定值的SQL即被记录到慢查询日志，缺省值为 `10s`

MySQL图形界面管理工具

现有很多数据库管理工具，其中有很多支持`MySQL`，比如：

- 跨平台
  - 客户端
    - `MySQL Workbench`
  - Web网页端
    - `phpMyAdmin`
- Mac
  - `Sequel Pro`
- Win
  - `Navicat`

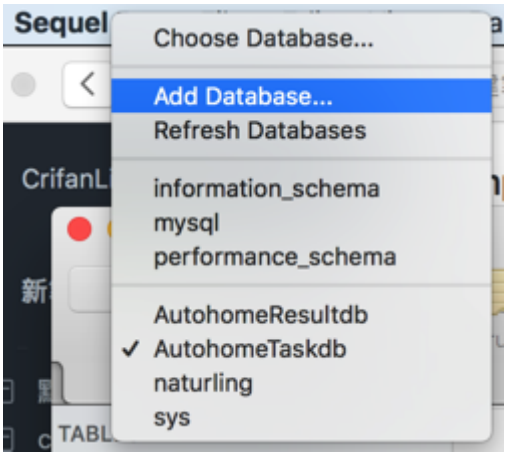
Sequel Pro

- `Sequel Pro`：数据库工具
  - 可用来管理MySQL数据库
  - 

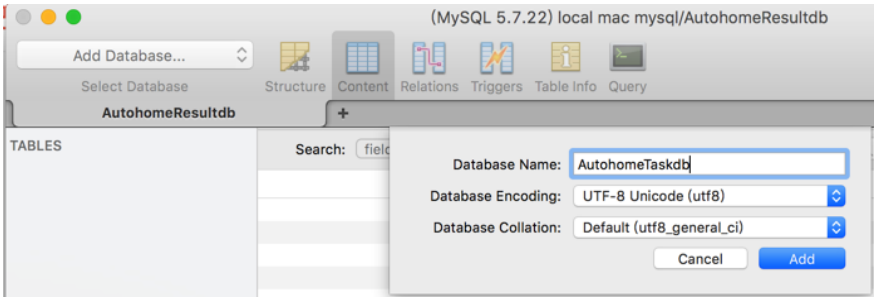
Sequel中新建mysql数据库并新建表和相应字段

sequel中，创建对应的mysql数据库，以及在数据库中创建对应的表，和相应字段

点击左上角的按钮，选择：`Add Database`：

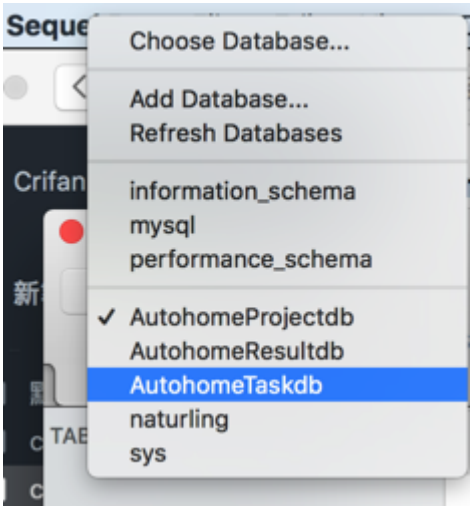


输入数据库信息：

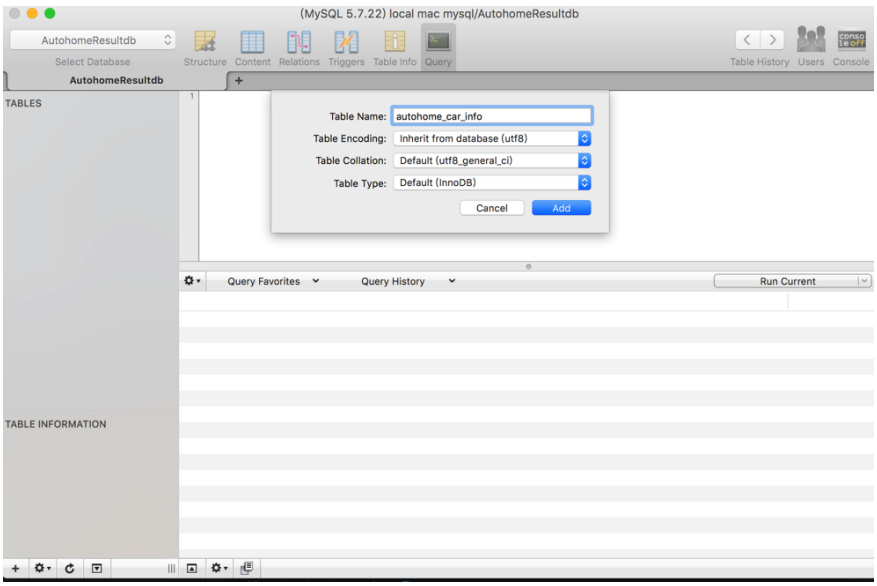


- `Database Name`：`AutohomeProjectdb`
- `Database Encoding`：`UTF-8 Unicode(utf8)`
- `Database Collation`：`Default (utf8_general_ci)`

再去切换到，刚建好的数据库：`AutohomeProjectdb`



再去创建对应的table：

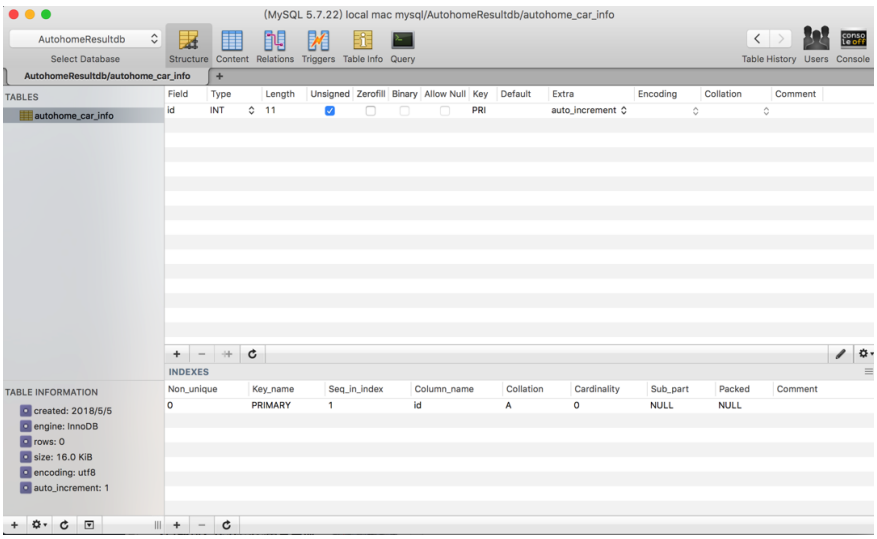


- `Table Name`：`autohome_car_info`
- `Table Encoding`：`inherit from database (utf8)`
- `Table Collation`：`Default (utf8_general_ci)`

- `Table Type : Default (InnoDB)`

点击 `Add`

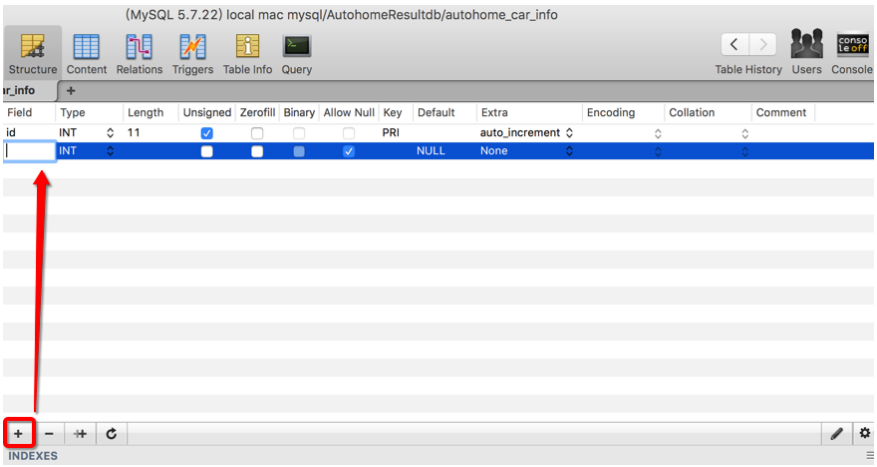
然后有了主键id



且默认已经帮忙配置好了 `auto increment = 自增`

再去添加其他字段：

点击左下角的 `+` ，去新建字段

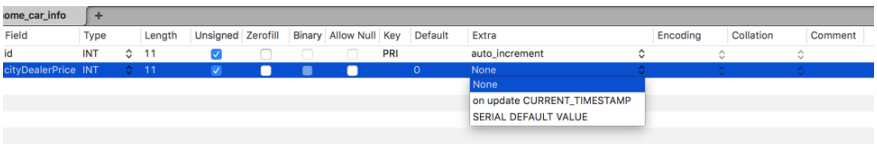


相关属性解释：

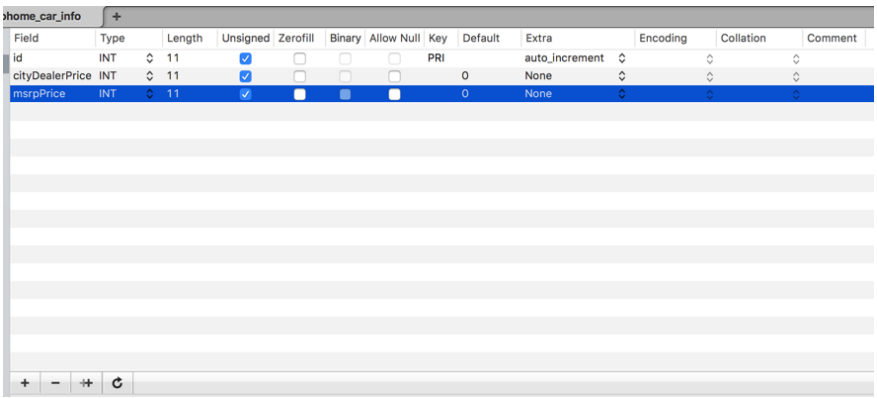
- `PK = primary key = 主键`
- `NN = not null = 非空`
- `UQ = unique = 唯一`
- `BIN = binary = 二进制数据 ： 比text更大`
- `UN = unsigned = 无符号 ： 非负数`
- `ZF = zero fill = 填充0 ： 例如字段内容是 1 int(4) , 则内容显示为 0001`
- `AI = auto increment = 自增`

此处调整字段属性为自己的需求：

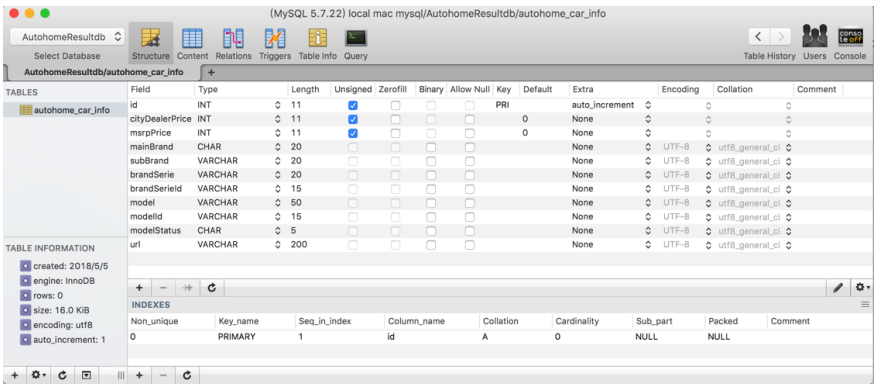
- 没必要 `ZF` ，所以去掉
  - 取消勾选 `ZeroFill`
- 也不允许NULL，默认为0
  - 取消勾选 `Allow Null`
- `Extra` 为空
  - `Extra` 改为 `None`



用类似方式，再去新建其他字段



最后新建的各个字段为：



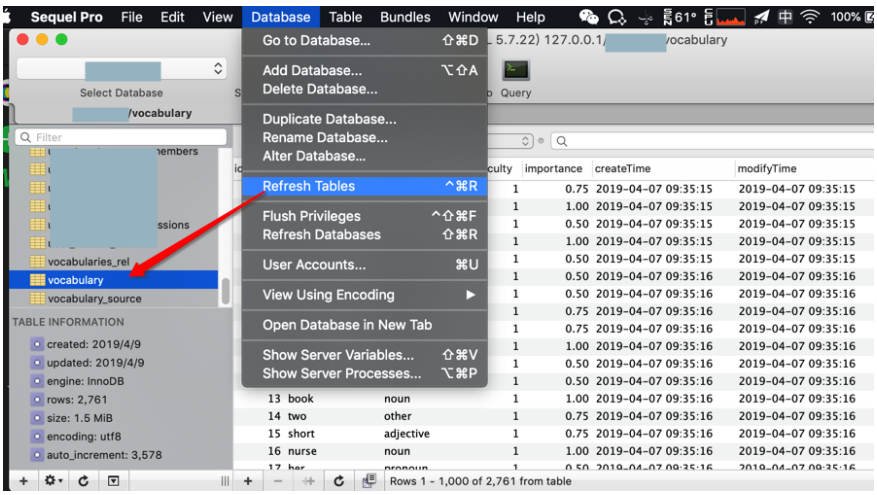
其中：

- 字段 `mainBrand` =主品牌：类型是 `CHAR`
  - 设计考虑点是
    - `char(n)` ：适合保存，基本上确定了不会超过n个字符的，个数变化不大的
      - 此处汽车主品牌的名字，不会太长，所以用 `CHAR` ，暂定最多 `20` 个字符
    - `varchar(n)` ：适合保存，基本上不太会超过n个字符，但是个数可能会变化的

使用心得

刷新

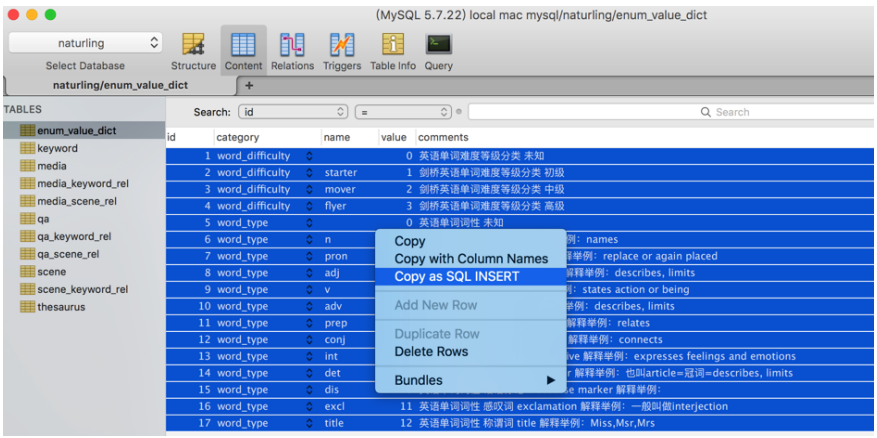
比如 导入了数据到MySQL中之后，需要刷新：



才能看到最新（已导入）的数据。

拷贝已有内容为sql语句

对于之前手动录入的内容，全选后，右击，支持：`Copy as SQL INSERT`

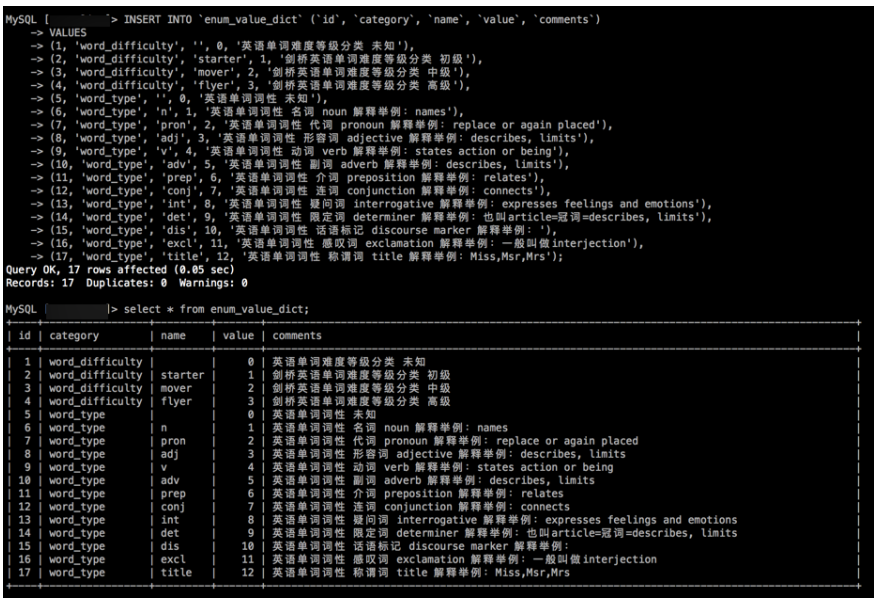


从而拷贝出sql语句：

```
INSERT INTO `enum_value_dict` (`id`, `category`, `name`, `
VALUES
    (1, 'word_difficulty', '', 0, '英语单词难度等级分类 未知'),
    (2, 'word_difficulty', 'starter', 1, '剑桥英语单词难度等级
    (3, 'word_difficulty', 'mover', 2, '剑桥英语单词难度等级分
    (4, 'word_difficulty', 'flyer', 3, '剑桥英语单词难度等级分
    (5, 'word_type', '', 0, '英语单词词性 未知'),
    (6, 'word_type', 'n', 1, '英语单词词性 名词 noun 解释举例:
    (7, 'word_type', 'pron', 2, '英语单词词性 代词 pronoun 解
    (8, 'word_type', 'adj', 3, '英语单词词性 形容词 adjective
    (9, 'word_type', 'v', 4, '英语单词词性 动词 verb 解释举例:
    (10, 'word_type', 'adv', 5, '英语单词词性 副词 adverb 解释
    (11, 'word_type', 'prep', 6, '英语单词词性 介词 prepositi
    (12, 'word_type', 'conj', 7, '英语单词词性 连词 conjuncti
    (13, 'word_type', 'int', 8, '英语单词词性 疑问词 interroga
    (14, 'word_type', 'det', 9, '英语单词词性 限定词 determine
    (15, 'word_type', 'dis', 10, '英语单词词性 话语标记 discou
    (16, 'word_type', 'excl', 11, '英语单词词性 感叹词 exclama
    (17, 'word_type', 'title', 12, '英语单词词性 称谓词 title
```

后续可用于其他地方，比如：

去另外一个在线mysql中插入同样内容



```
MySQL [          > INSERT INTO `enum_value_dict` (`id`, `category`, `name`, `value`, `comments`)
    -> VALUES
    -> (1, 'word_difficulty', '', 0, '英语单词难度等级分类 未知'),
    -> (2, 'word_difficulty', 'starter', 1, '剑桥英语单词难度等级分类 初级'),
    -> (3, 'word_difficulty', 'mover', 2, '剑桥英语单词难度等级分类 中级'),
    -> (4, 'word_difficulty', 'flyer', 3, '剑桥英语单词难度等级分类 高级'),
    -> (5, 'word_type', '', 0, '英语单词词性 未知'),
    -> (6, 'word_type', 'n', 1, '英语单词词性 名词 noun 解释举例: names'),
    -> (7, 'word_type', 'pron', 2, '英语单词词性 代词 pronoun 解释举例: replace or again placed'),
    -> (8, 'word_type', 'adj', 3, '英语单词词性 形容词 adjective 解释举例: describes, limits'),
    -> (9, 'word_type', 'v', 4, '英语单词词性 动词 verb 解释举例: states action or being'),
    -> (10, 'word_type', 'adv', 5, '英语单词词性 副词 adverb 解释举例: describes, limits'),
    -> (11, 'word_type', 'prep', 6, '英语单词词性 介词 preposition 解释举例: relates'),
    -> (12, 'word_type', 'conj', 7, '英语单词词性 连词 conjunction 解释举例: connects'),
    -> (13, 'word_type', 'int', 8, '英语单词词性 疑问词 interrogative 解释举例: expresses feelings and emotions'),
    -> (14, 'word_type', 'det', 9, '英语单词词性 限定词 determiner 解释举例: 也叫article=冠词=describes, limits'),
    -> (15, 'word_type', 'dis', 10, '英语单词词性 话语标记 discourse marker 解释举例: '),
    -> (16, 'word_type', 'excl', 11, '英语单词词性 感叹词 exclamation 解释举例: 一般叫做interjection'),
    -> (17, 'word_type', 'title', 12, '英语单词词性 称谓词 title 解释举例: Miss,Msr,Mrs');
Query OK, 17 rows affected (0.05 sec)
Records: 17  Duplicates: 0  Warnings: 0

MySQL [          ]> select * from enum_value_dict;
+----+---------------+---------+-------+-----------------------------------------------------------------------------------------------------------------------+
| id | category      | name    | value | comments                                                                                                              |
+----+---------------+---------+-------+-----------------------------------------------------------------------------------------------------------------------+
|  1 | word_difficulty |         |     0 | 英语单词难度等级分类 未知                                                                                             |
|  2 | word_difficulty | starter |     1 | 剑桥英语单词难度等级分类 初级                                                                                         |
|  3 | word_difficulty | mover   |     2 | 剑桥英语单词难度等级分类 中级                                                                                         |
|  4 | word_difficulty | flyer   |     3 | 剑桥英语单词难度等级分类 高级                                                                                         |
|  5 | word_type       |         |     0 | 英语单词词性 未知                                                                                                     |
|  6 | word_type       | n       |     1 | 英语单词词性 名词 noun 解释举例: names                                                                                |
|  7 | word_type       | pron    |     2 | 英语单词词性 代词 pronoun 解释举例: replace or again placed                                                           |
|  8 | word_type       | adj     |     3 | 英语单词词性 形容词 adjective 解释举例: describes, limits                                                             |
|  9 | word_type       | v       |     4 | 英语单词词性 动词 verb 解释举例: states action or being                                                               |
| 10 | word_type       | adv     |     5 | 英语单词词性 副词 adverb 解释举例: describes, limits                                                                  |
| 11 | word_type       | prep    |     6 | 英语单词词性 介词 preposition 解释举例: relates                                                                       |
| 12 | word_type       | conj    |     7 | 英语单词词性 连词 conjunction 解释举例: connects                                                                      |
| 13 | word_type       | int     |     8 | 英语单词词性 疑问词 interrogative 解释举例: expresses feelings and emotions                                           |
| 14 | word_type       | det     |     9 | 英语单词词性 限定词 determiner 解释举例: 也叫article=冠词=describes, limits                                           |
| 15 | word_type       | dis     |    10 | 英语单词词性 话语标记 discourse marker 解释举例:                                                                      |
| 16 | word_type       | excl    |    11 | 英语单词词性 感叹词 exclamation 解释举例: 一般叫做interjection                                                        |
| 17 | word_type       | title   |    12 | 英语单词词性 称谓词 title 解释举例: Miss,Msr,Mrs                                                                      |
+----+---------------+---------+-------+-----------------------------------------------------------------------------------------------------------------------+
```

常见问题

Unable to connect to host 127.0.0.1, or the request timed out

- 现象

在Mac中，虽然之前用mysql的命令行把安装时的临时密码修改过了，且终端中也可以用：

```
mysql -u root -p
```

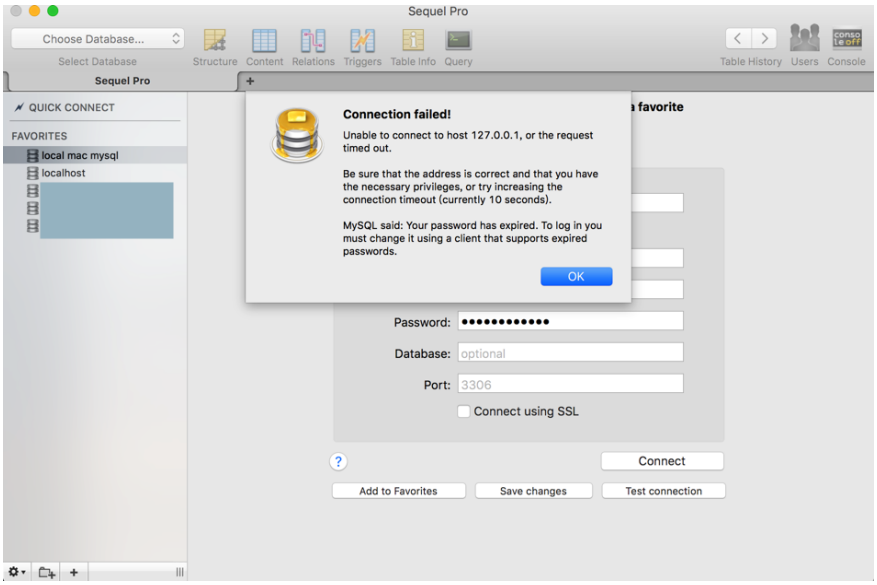
再输入密码，而进去mysql了。

但是此处工具 `Sequel Pro` 中，却无法连接：

- host：`127.0.0.1`
- user：`root`
- password：xxx
- port：`3306`

而报错：

```
Connection Failed
Unable to connect to host 127.0.0.1, or the request timed
```



而再去找具体提示中的原因是：

```
Your password has expired. To log in you must change it us
```

而具体解决办法是：

再去终端中进去mysql：

```
mysql -u root -p
```

然后，先选择对应的mysql数据库：

```
>use mysql;
```

会提示你：

```
ERROR 1820 (HY000): You must reset your password using ALTE
```

然后才能设置新密码：

```
SET PASSWORD = PASSWORD('yourNewPassword');
```

(如果需要)再去设置密码不过期：

```
use mysql;
update user set password_expired='N' where user='root';
```

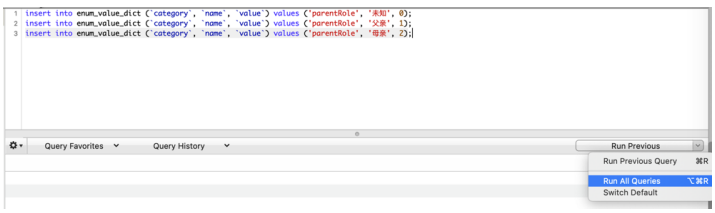
最后记得保存/同步/更新后再退出：

```
flush privileges;
quit
```

然后 `Sequel Pro` 中就可以用新密码去正常连接了。

运行多行mysql语句

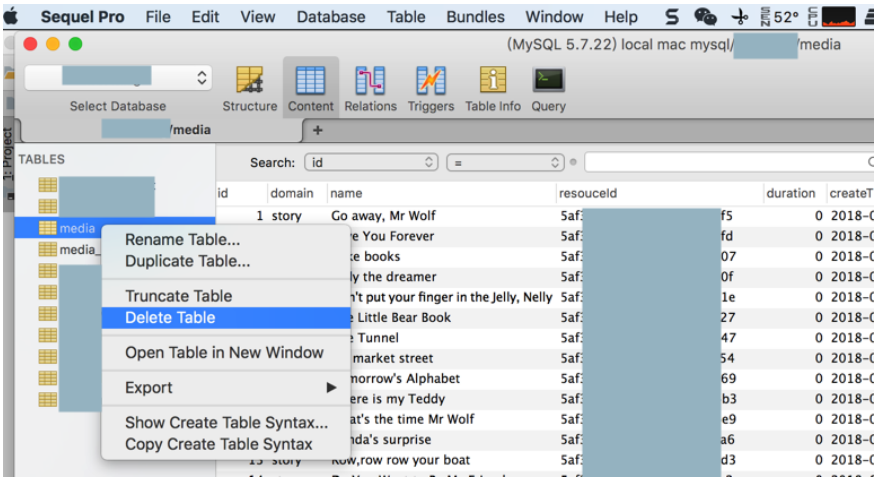
sequel中，要是运行多行sql语句，则

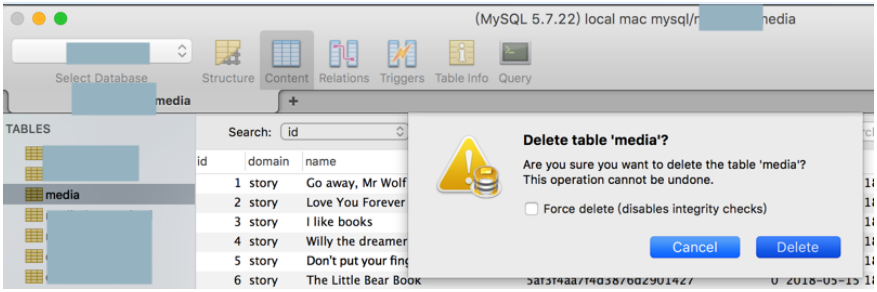
- 应该是：`Run All Queries`
  - 图
    - 

  - 底部提示的执行结果中，才能看到想要的：`N rows affected`
- 而不是：`Run Previous`
  - 否则多行语句，只能执行最后一行，之前的行，不会执行
  - 底部提示的执行结果中，也只能看到：`1 rows affected`

删除表出错：MySQL said: Cannot delete or update a parent row: a foreign key constraint fails

Sequel中，去右键删除某个表：

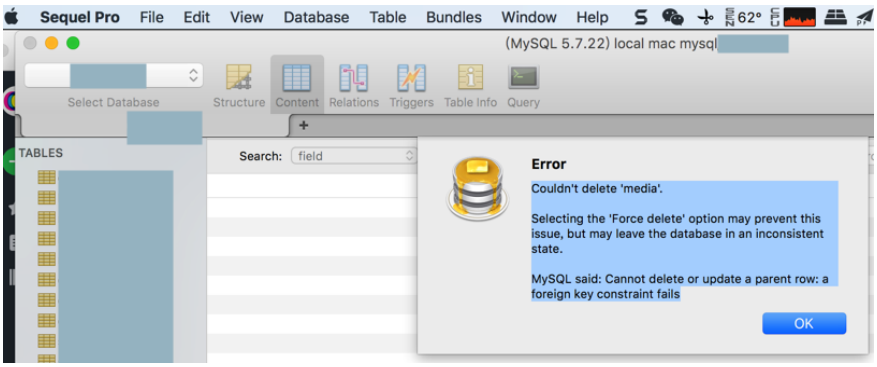




结果报错：

```
Error
Couldn't delete 'media'.

Selecting the 'Force delete' option may prevent this issue,
MySQL said: Cannot delete or update a parent row: a foreign
```

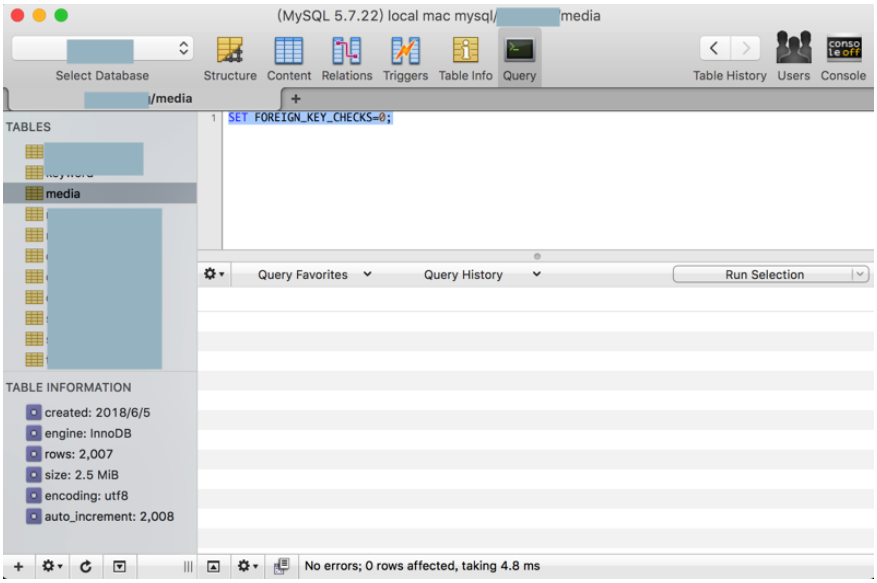


原因：此表某字段有外键和依赖关系

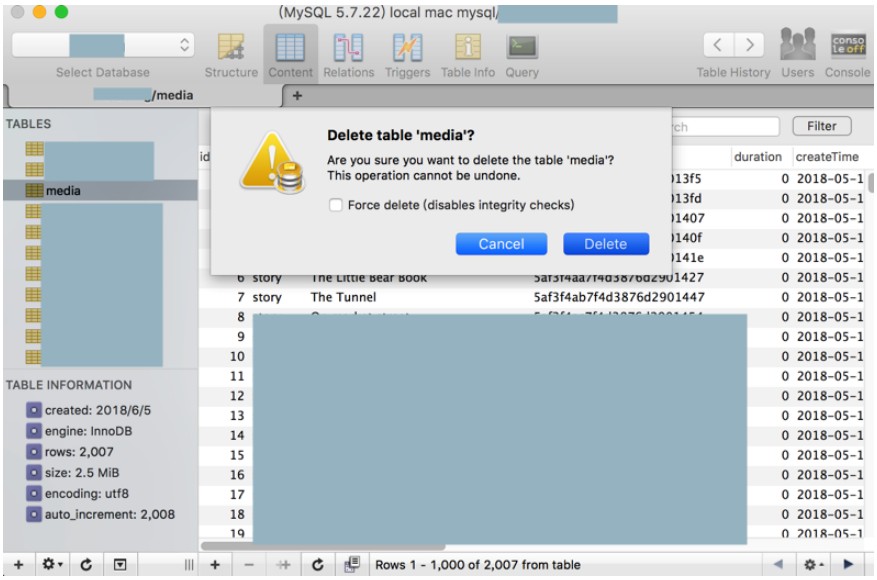
解决办法：

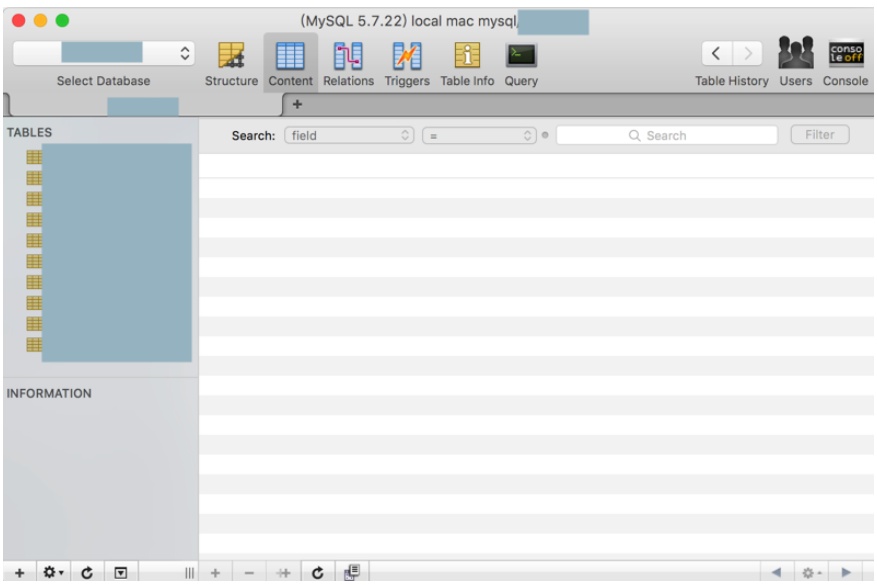
- 关闭（外键检查的）限制

```
SET FOREIGN_KEY_CHECKS=0;
```


(MySQL 5.7.22) local mac mysql/ media
Select Database
Structure
Content
Relations
Triggers
Table Info
Query
Table History
Users
Console
/media
TABLES
media
SET FOREIGN_KEY_CHECKS=0;
Query Favorites
Query History
Run Selection
TABLE INFORMATION
created: 2018/6/5
engine: InnoDB
rows: 2,007
size: 2.5 MiB
encoding: utf8
auto_increment: 2,008
No errors; 0 rows affected, taking 4.8 ms

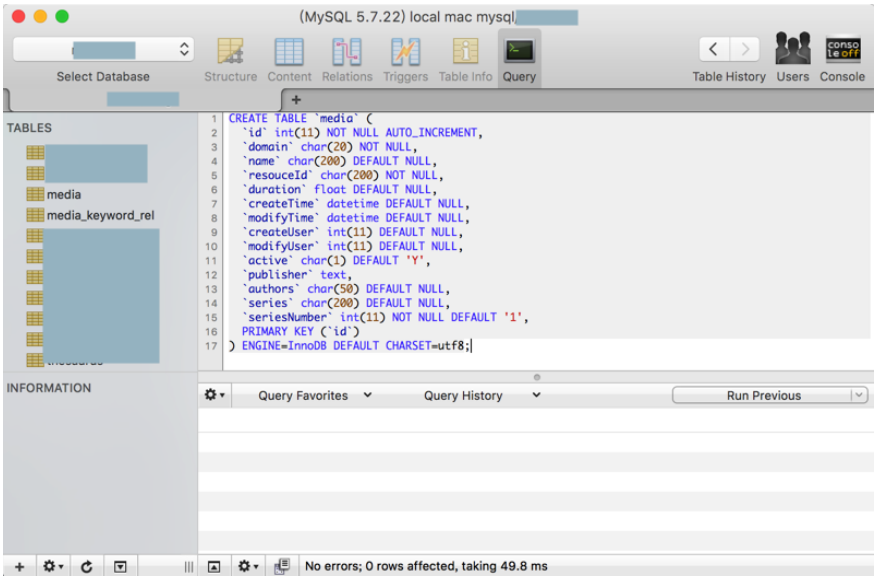
- 去操作，删除表


(MySQL 5.7.22) local mac mysql/
Select Database
Structure
Content
Relations
Triggers
Table Info
Query
Table History
Users
Console
/media
TABLES
media
Delete table 'media'?
Are you sure you want to delete the table 'media'?
This operation cannot be undone.
Force delete (disables integrity checks)
Cancel
Delete
Filter
duration
createTime
7 story The Tunnel 5af3f4ab7f4d3876d2901447
TABLE INFORMATION
created: 2018/6/5
engine: InnoDB
rows: 2,007
size: 2.5 MiB
encoding: utf8
auto_increment: 2,008
Rows 1 - 1,000 of 2,007 from table


(MySQL 5.7.22) local mac mysql
Select Database
Structure
Content
Relations
Triggers
Table Info
Query
Table History
Users
Console
TABLES
Search:
field
Search
Filter
INFORMATION

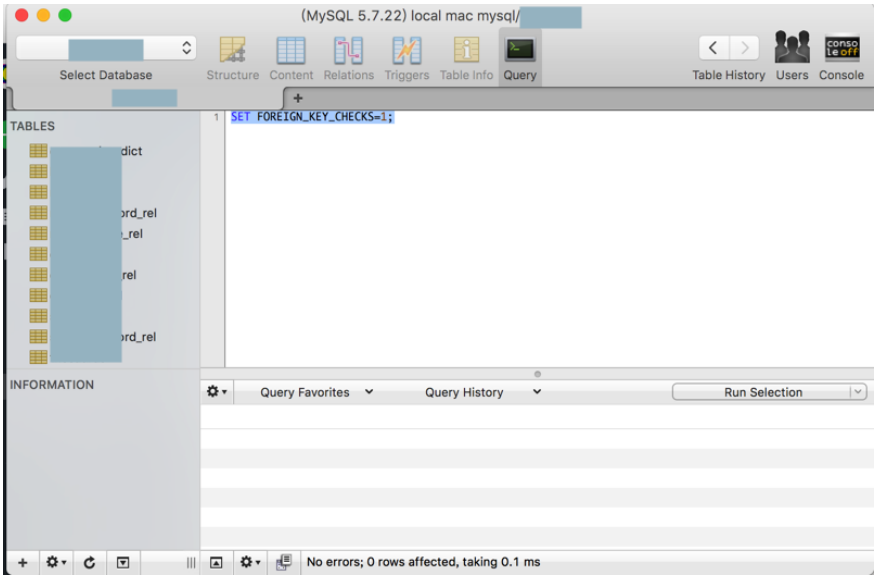
- （更新了个别字段后，重新）创建之前的表

```
CREATE TABLE `media` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `domain` char(20) NOT NULL,
  `name` char(200) DEFAULT NULL,
  `resouceId` char(200) NOT NULL,
  `duration` float DEFAULT NULL,
  `createTime` datetime DEFAULT NULL,
  `modifyTime` datetime DEFAULT NULL,
  `createUser` int(11) DEFAULT NULL,
  `modifyUser` int(11) DEFAULT NULL,
  `active` char(1) DEFAULT 'Y',
  `publisher` text,
  `authors` char(50) DEFAULT NULL,
  `series` char(200) DEFAULT NULL,
  `seriesNumber` int(11) NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```



- 重新开启（外键检查的）限制

```
SET FOREIGN_KEY_CHECKS=1;
```



如此即可满足需求。

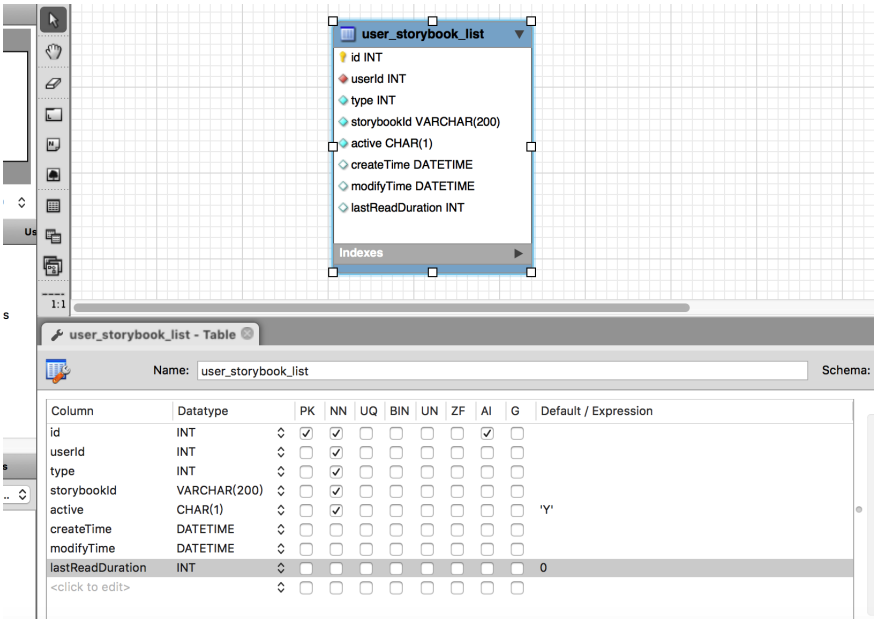
且之前的两个表（ `media_keyword_rel` 和 `media_scene_rel` ）的逻辑结构也还是没有变的，正常的外键依赖。

MySQL Workbench

- `MySQL Workbench`：MySQL数据库工具
  - 可以用来创建MySQL表

用法举例

比如去创建一个表，设计字段：



代码操作mysql

Python中使用mysql

此处介绍如何在Python中使用和操作MySQL数据库。

- 先选择合适的mysql连接器/适配器
  - `PyMySQL`
    - https://github.com/PyMySQL/PyMySQL
      - star: `3000+`
  - `mysqlclient`
    - https://github.com/PyMySQL/mysqlclient-python
      - star: `1000+`
  - Oracle的官方的 `MySQL Connector`
    - https://github.com/mysql/mysql-connector-python
      - star: `300+`
      - 基本用法
        - 安装

          ```
          pipenv install mysql-connector-python
          ```

        - 导入

          ```
          import mysql
          ```

  - 结论
    - 选用的star最多的且支持Python3的： `PyMySQL`

心得

mysql中操作表的字段名时是否一定要用反引号括起来

MySQL中的一些对象，比如：

- database
- table
- index
- column
- alias
- view
- stored procedure
- partition
- tablespace
- resource group

- other object

这些对象的名字，被叫做 标识符 identifier

而标识符的所允许的字符是有一定的限制的

有些标识符是在某些情况下是区分大小写的

对于一个标识符来说，可能被引起来quoted，也可能是没有被引起来unqoted

- 如果包含特殊字符，后者是系统保留字，则必须要被quote
  - 例外是：当这个标识符后面有个点，表示引用字段，则可以不用quote
  - 而系统保留字的话，比如，`select`，`interval`，`from`等等
    - 详见：MySQL :: MySQL 8.0 Reference Manual :: 9.3 Keywords and Reserved Words
- 对于普通的标识符
  - 如果不quote的话，所允许的字符有：
    - `0-9`数字，`a-Z`的字母，美元符号`$`，下划线`_`
    - 扩展的Unicode字符：`U+0080` .. `U+FFFF`
  - 对于quote的标识符的话，所允许的字符有：
    - 基本的ASCII字符：`U+0001` .. `U+007F`
    - 扩展的Unicode字符：`U+0080` .. `U+FFFF`
- 不论是quote还是unquote，都不允许：
  - ASCII `NUL`（`U+0000`）
  - supplementary characters（`U+10000` and higher)
- 标识符中可以以数字开头，但是要quote，且不能只是数字
- `Database`，`table`，`column`的名字最后不能以空格结尾

对于quote的字符：

- 标准的来说是：`反引号` == ` == `backticks`

举例：

```
mysql> SELECT * FROM `select` WHERE `select`.id > 100;
```

- 如果启用了`ANSI_QUOTES`的SQL模式的话，则也可以用`双引号` == `""`去quote

举例：

```
mysql> CREATE TABLE "test" (col INT);
ERROR 1064: You have an error in your SQL syntax...
mysql> SET sql_mode='ANSI_QUOTES';
mysql> CREATE TABLE "test" (col INT);
Query OK, 0 rows affected (0.00 sec)
```

`ANSI_QUOTES` 模式的话，会使得系统把双引号内的字符，作为标识符

同时也意味着，对于普通的字符串来说（本来是用双引号括起来的），现在只能用 `单引号` == `'` 去括起来了。

关于 `ANSI_QUOTES` ，详见：

MySQL :: MySQL 8.0 Reference Manual :: 5.1.10 Server SQL Modes

下面再来说一些，更加特殊的情况：

quote字符，比如 `反引号` ， `双引号` ，也是可以被括起来，放在标识符里的，不过是：

如果被quote起来的字符，包含了quote字符本身，就需要写两次

举例：

```
mysql> CREATE TABLE `a``b` (`c"d` INT);
```

解释：

此处的table的标识符名字是：

```
a`b
```

由于外部的quote引起来的字符用的是 `反引号` `

所以a和b中间的反引号，要写成2遍，变成：

```
a``b
```

被quote引起来就是：

```
`a``b`
```

而对于另外的一个（只有启用了ANSI_QUOTES）quote字符：双引号 `"` 来说，由于外部的quote字符不是双引号，所以不用写2遍，所以可以写成：

```
`c"d`
```

对比标识符的命名来说：

不建议用数字开头，比如：`Me` `MeN`，其中`M`和`N`是数字

总之：

为了避免其他副作用，尽量只用普通的字符：`0-9`，`a-Z`，`下划线`

去命名，作为标识符

- 一，省去了quote的考虑
  - 避免其他不必要的麻烦
- 二，也是好的编程和设计的原则和规范

而对于此处的：

问：写mysql的代码中，是否一定要用别人那种escape的写法？

答：

- 如果自己的标识符变量命名，都是很规则，比如我的`cityDealerPrice`，`modelStatus`等等，没有包含特殊的容易导出解析错误的单词或字符，那就不需要用`去quote
- 如果不能确保标识符是普通的字符，比如标识符中可能出现空格或其他特殊字符，或者是系统保留字，则最好是用上`去quote起来，以防万一。

pymysql

安装:

```
pip install pymysql

pipenv install pymysql
```

Python中用pymysql操作MySQL数据库

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# Project: autohomeBrandData
# Function: demo how to use python to operate mysql
# Author: Crifan Li
# Date: 20180527
# Note:
#   If you want to modify to your mysql and table, you neec
#   (1) change change MysqlDb config to your mysql config
#   (2) change CurrentTableName to your table name
#   (3) change CreateTableSqlTemplate to your sql to create
#   (4) run py file to execute testMysqlDb to init db and c
#   (5) if your table field contain more type, edit insert


import pymysql
import pymysql.cursors

CurrentTableName = "tbl_autohome_car_info"
CreateTableSqlTemplate = """CREATE TABLE IF NOT EXISTS `%s`
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT COMMENT '自
  `cityDealerPrice` int(11) unsigned NOT NULL DEFAULT '0' C
  `msrpPrice` int(11) unsigned NOT NULL DEFAULT '0' COMMENT
  `mainBrand` char(20) NOT NULL DEFAULT '' COMMENT '品牌',
  `subBrand` varchar(20) NOT NULL DEFAULT '' COMMENT '子品牌
  `brandSerie` varchar(20) NOT NULL DEFAULT '' COMMENT '车系
  `brandSerieId` varchar(15) NOT NULL DEFAULT '' COMMENT '
  `model` varchar(50) NOT NULL DEFAULT '' COMMENT '车型',
  `modelId` varchar(15) NOT NULL DEFAULT '' COMMENT '车型ID
  `modelStatus` char(5) NOT NULL DEFAULT '' COMMENT '车型状态
  `url` varchar(200) NOT NULL DEFAULT '' COMMENT '车型url',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;"""


class MysqlDb:
    config = {
        'host': '127.0.0.1',
        'port': 3306,
        'user': 'root',
        'password': 'crifan_mysql',
        'database': 'AutohomeResultdb',
        'charset': "utf8"
    }

    defaultTableName = CurrentTableName
    connection = None

    def __init__(self):
        """init mysql"""
```

```
        # 1. connect db first
        if self.connection is None:
            isConnected = self.connect()
            print("Connect mysql return %s" % isConnected)

        # 2. create table for db
        createTableOk = self.createTable(self.defaultTableN
        print("Create table %s return %s" %(self.defaultTab

    def connect(self):
        try:
            self.connection = pymysql.connect(**self.config
            print("connect mysql ok, self.connection=", se
            return True
        except pymysql.Error as err:
            print("Connect mysql with config=", self.config
            return False

    def quoteIdentifier(self, identifier):
        """
            for mysql, it better to quote identifier xxx us
            in case, identifier:
                contain special char, such as space
                or same with system reserved words, like se
        """
        quotedIdentifier = "`%s`" % identifier
        # print("quotedIdentifier=", quotedIdentifier)
        return quotedIdentifier

    def executeSql(self, sqlStr, actionDescription=""):
        print("executeSql: sqlStr=%s, actionDescription=%s"

        if self.connection is None:
            print("Please connect mysql first before %s" %
            return False

        cursor = self.connection.cursor()
        print("cursor=", cursor)

        try:
            cursor.execute(sqlStr)
self.connection.commit()
            print("+++ Ok to execute sql %s for %s" % (sql
            return True
        except pymysql.Error as err:
            print("!!! %s when execute sql %s for %s" % (er
            return False

    def createTable(self, newTablename):
        print("createTable: newTablename=", newTablename)
```

```
        createTableSql = CreateTableSqlTemplate % (newTable
        print("createTableSql=", createTableSql)

        return self.executeSql(sqlStr=createTableSql, actio

    def dropTable(self, existedTablename):
        print("dropTable: existedTablename=", existedTablen

        dropTableSql = "DROP TABLE IF EXISTS %s" % (existed
        print("dropTableSql=", dropTableSql)

        return self.executeSql(sqlStr=dropTableSql, actionD

    # def insert(self, **valueDict):
    def insert(self, valueDict, tablename=defaultTableName)
        """
            inset dict value into mysql table
            makesure the value is dict, and its keys is the
        """
        print("insert: valueDict=%s, tablename=%s" % (value

        dictKeyList = valueDict.keys()
        dictValueList = valueDict.values()
        print("dictKeyList=", dictKeyList, "dictValueList="

        keyListSql = ", ".join(self.quoteIdentifier(eachKey
        print("keyListSql=", keyListSql)
        # valueListSql = ", ".join(eachValue for eachValue
        valueListSql = ""
        formattedDictValueList = []
        for eachValue in dictValueList:
            # print("eachValue=", eachValue)
            eachValueInSql = ""
            valueType = type(eachValue)
            # print("valueType=", valueType)
            if valueType is str:
                eachValueInSql = '"%s"' % eachValue
            elif valueType is int:
                eachValueInSql = '%d' % eachValue
            # TODO: add more type formatting if necessary
            print("eachValueInSql=", eachValueInSql)
            formattedDictValueList.append(eachValueInSql)

        valueListSql = ", ".join(eachValue for eachValue in
        print("valueListSql=", valueListSql)

        insertSql = """INSERT INTO %s (%s) VALUES (%s)""" %
        print("insertSql=", insertSql)
        # INSERT INTO tbl_car_info_test (`url`, `mainBrand`

        return self.executeSql(sqlStr=insertSql, actionDesc
```

```
    def delete(self, modelId, tablename=defaultTableName):
        """
            delete item from car model id for existing tab
        """
        print("delete: modelId=%s, tablename=%s" % (modelId

        deleteSql = """DELETE FROM %s WHERE modelId = %s"""
        print("deleteSql=", deleteSql)

        return self.executeSql(sqlStr=deleteSql, actionDesc

def testMysqlDb():
    """test mysql"""

    testDropTable = True
    testCreateTable = True
    testInsertValue = True
    testDeleteValue = True

    # 1.test connect mysql
    mysqlObj = MysqlDb()
    print("mysqlObj=", mysqlObj)

    # testTablename = "autohome_car_info"
    # testTablename = "tbl_car_info_test"
    testTablename = CurrentTableName
    print("testTablename=", testTablename)

    if testDropTable:
        # 2. test drop table
        dropTableOk = mysqlObj.dropTable(testTablename)
        print("dropTable", testTablename, "return", dropTab

    if testCreateTable:
        # 3. test create table
        createTableOk = mysqlObj.createTable(testTablename)
        print("createTable", testTablename, "return", creat

    if testInsertValue:
        # 4. test insert value dict
        valueDict = {
            "url": "https://www.autohome.com.cn/spec/5872/#
            "mainBrand": "宝马", #品牌
            "subBrand": "华晨宝马", #子品牌
            "brandSerie": "宝马3系", #车系
            "brandSerieId": "66", #车系ID
            "model": "2010款 320i 豪华型", #车型
            "modelId": "5872", #车型ID
            "modelStatus": "停售", #车型状态
            "cityDealerPrice": 325000, #经销商参考价
```

```
            "msrpPrice": 375000 # 厂商指导价
        }
        print("valueDict=", valueDict)
        insertOk = mysqlObj.insert(valueDict=valueDict, tab
        print("insertOk=", insertOk)

    if testDeleteValue:
        toDeleteModelId = "5872"
        deleteOk = mysqlObj.delete(modelId=toDeleteModelId,
        print("deleteOk=", deleteOk)

def testAutohomeResultWorker():
    """just test for create mysql db is ok or not"""
    autohomeResultWorker = AutohomeResultWorker(None, None)
    print("autohomeResultWorker=%s" % autohomeResultWorker)

if __name__ == '__main__':
    testMysqlDb()
    # testAutohomeResultWorker()
```

Python的Flask中pymysql中mysql返回分页查询结果

```
userId = parsedArgs["userId"]
if userId is None:
    return genRespFailDict(BadRequest.code, "Empty userId"

if userId <= 0:
    return genRespFailDict(BadRequest.code, "Invalid userI

pageNumber = parsedArgs["pageNumber"]
if pageNumber < 1:
    return genRespFailDict(BadRequest.code, "Invalid pageNu

pageSize = parsedArgs["pageSize"]
pageIndex = pageNumber - 1
offset = pageSize * pageIndex
orderBy = "modifyTime"

totalCount = -1
totalPageNum = -1

totalCountSql = "SELECT COUNT(*) from `user_storybook_list`
totalCountOk, resultDict = sqlConn.executeSql(totalCountSq
log.debug("%s -> %s, %s", totalCountSql, totalCountOk, resu
if totalCountOk and resultDict["data"]:
    totalCount = resultDict["data"][0]["COUNT(*)"]

if totalCount > 0:
    totalPageNum = int(totalCount / pageSize)

    if (totalCount % pageSize) > 0:
        totalPageNum += 1

    if pageNumber > totalPageNum:
        return genRespFailDict(BadRequest.code, "Current pa
        pageNumber, totalPageNum))

hasPrev = False
if pageNumber > 1:
    hasPrev = True

hasNext = False
if totalPageNum > 0:
    if pageNumber < totalPageNum:
        hasNext = True

searchByUserSql = "SELECT * FROM `user_storybook_list` WHER
searchByUserOk, resultDict = sqlConn.executeSql(searchByUse
log.debug("%s -> %s, %s", searchByUserSql, searchByUserOk,
if searchByUserOk and resultDict["data"]:
    foundItemList = resultDict["data"]
```

```
    respData = {
        "evaluationList": foundItemList,
        "curPageNum": pageNumber,
        "numPerPage": pageSize,
        "totalNum": totalCount,
        "totalPageNum": totalPageNum,
        "hasPrev": hasPrev,
        "hasNext": hasNext,
    }

    respDict["data"] = respData
    return jsonify(respDict)
```

返回数据：

```
{
    "code": 200,
    "data": {
        "curPageNum": 1,
        "evaluationList": [
            {
                "active": "Y",
                "createTime": "Wed, 23 Jan 2019 08:40:58 GM
                "id": 5,
                "lastReadDuration": 0,
                "modifyTime": "Wed, 23 Jan 2019 08:40:58 GM
                "storybookId": "5bd7bd31bfaa44fe2c73736b",
                "type": 0,
                "userId": 28
            },
            {
                "active": "Y",
                "createTime": "Wed, 23 Jan 2019 08:45:04 GM
                "id": 6,
                "lastReadDuration": 0,
                "modifyTime": "Wed, 23 Jan 2019 08:45:04 GM
                "storybookId": "5bd7bd31bfaa44fe2c73736c",
                "type": 0,
                "userId": 28
            },
            {
                "active": "Y",
                "createTime": "Wed, 23 Jan 2019 08:45:32 GM
                "id": 7,
                "lastReadDuration": 0,
                "modifyTime": "Wed, 23 Jan 2019 08:45:32 GM
                "storybookId": "5bd7bd31bfaa44fe2c73736e",
                "type": 0,
                "userId": 28
            }
        ],
        "hasNext": true,
        "hasPrev": false,
        "numPerPage": 3,
        "totalNum": 9,
        "totalPageNum": 3
    },
    "message": "Get user storybook list ok"
}
```

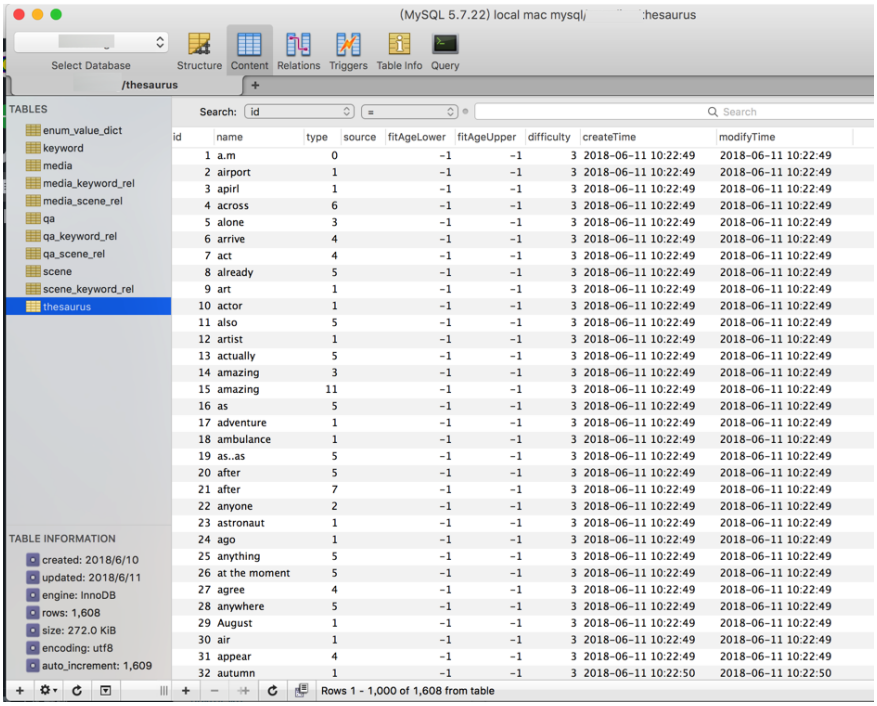

Postman测试效果：

 pymysql_resp_list_postman

MySQL心得

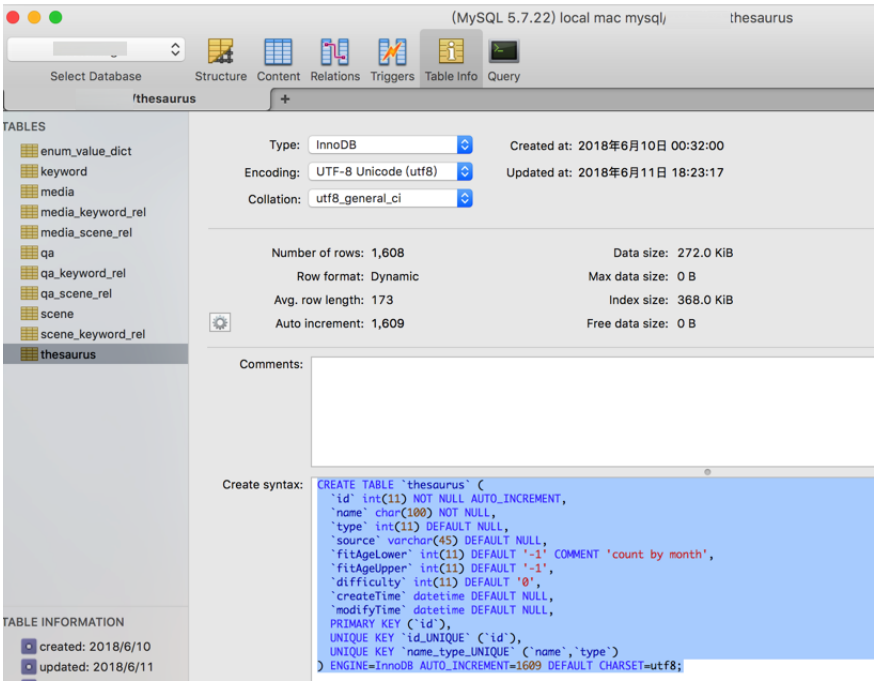
`create table` 的sql语句中，注意新建表时，去掉 `AUTO_INCREMENT=xxx`

对于一个mysql的某个表，本地已经调试完毕，插入了很多数据（1608条）：



需要在在线数据库中新建该表，然后就参考了 `table info` 中的sql语句：

```
CREATE TABLE `thesaurus` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` char(100) NOT NULL,
  `type` int(11) DEFAULT NULL,
  `source` varchar(45) DEFAULT NULL,
  `fitAgeLower` int(11) DEFAULT '-1' COMMENT 'count by mont
  `fitAgeUpper` int(11) DEFAULT '-1',
  `difficulty` int(11) DEFAULT '0',
  `createTime` datetime DEFAULT NULL,
  `modifyTime` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `name_type_UNIQUE` (`name`,`type`)
) ENGINE=InnoDB AUTO_INCREMENT=1609 DEFAULT CHARSET=utf8;
```



然后去粘贴到在线sql命令行中去创建：

```
MySQL          > CREATE TABLE `thesaurus` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `name` char(100) NOT NULL,
    ->   `type` int(11) DEFAULT NULL,
    ->   `source` varchar(45) DEFAULT NULL,
    ->   `fitAgeLower` int(11) DEFAULT '-1' COMMENT 'count by month',
    ->   `fitAgeUpper` int(11) DEFAULT '-1',
    ->   `difficulty` int(11) DEFAULT '0',
    ->   `createTime` datetime DEFAULT NULL,
    ->   `modifyTime` datetime DEFAULT NULL,
    ->   PRIMARY KEY (`id`),
    ->   UNIQUE KEY `id_UNIQUE` (`id`),
    ->   UNIQUE KEY `name_type_UNIQUE` (`name`,`type`)
    -> ) ENGINE=InnoDB AUTO_INCREMENT=1609 DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.20 sec)
```

然后才注意到，其实此处不应该加上那个：

`AUTO_INCREMENT=1609`

-> 否则新建的表的后续插入的值的id，都是从1609开始了。。

->所以应该去掉 `AUTO_INCREMENT=1609` ：

```
CREATE TABLE `thesaurus` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` char(100) NOT NULL,
  `type` int(11) DEFAULT NULL,
  `source` varchar(45) DEFAULT NULL,
  `fitAgeLower` int(11) DEFAULT '-1' COMMENT 'count by mont
  `fitAgeUpper` int(11) DEFAULT '-1',
  `difficulty` int(11) DEFAULT '0',
  `createTime` datetime DEFAULT NULL,
  `modifyTime` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `name_type_UNIQUE` (`name`,`type`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

-> 这样新插入的值，`id`才是从`1`开始的

用mysql保存枚举类型的字符串的值

- 为了（以后数据量大时的性能考虑），建议用`int`保存
- 为了简单方便好用，可以用`ENUM`保存

Python操作mysql

Error 1064 You have an error in your SQL syntax

有时候会insert会遇到语法错误：

`Error 1064 You have an error in your SQL syntax`

其中一种解决办法是：

字符串的如何扩起来，包裹起来，去保存：

- 用`单引号 = '`
- 还是`双引号 = "`

最常见的做法是：

```
'contain single quote inside sql use double single quote: \
```

相关的，能正常运行的Python代码：

```
dialogA = dialogAB[0]
dialogB = dialogAB[1]
dialogA = dialogA.replace("'", "''")
dialogB = dialogB.replace("'", "''")
if len(dialogA) > 0 and len(dialogB) > 0:
    #insertMediaSql = "INSERT INTO qa(id,question,answer,c
    insertMediaSql = """
      INSERT INTO qa(`id`,`question`,`answer`,`createTime`,
      VALUES(%d,'%s','%s','%s','%s',%d)"""
    qaid = curId
    executeSql = insertMediaSql % (qaid, dialogA, dialogB,
```

mysql的命令行

会自动重连

在mysql的console命令行中，（由于超时等原因）断开🔗 后，直接输入命令，会自动重连，然后再执行命令：

```
mysql> SELECT * FROM keyword WHERE type="topic" AND name="a
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    2109
Current database: xxx

+------+-------------------+-------+---------------------+-
| id   | name              | type  | createTime          |
+------+-------------------+-------+---------------------+-
| 7423 | american football | topic | 2018-07-05 15:38:20 |
+------+-------------------+-------+---------------------+-
1 row in set (0.20 sec)

mysql>
```

给数据库改名

```
mysqldump -u username -p -v olddatabase > olddbdump.sql
mysqladmin -u username -p create newdatabase
mysql -u username -p newdatabase < olddbdump.sql
```

清空某个表的数据内容

	delete	truncate
语法	`delete from 表名;`	`truncate table 表名;`
返回值	被删除的记录数	`0`
作用解释	不带where参数的delete语句可以删除mysql表中所有内容	也可以清空mysql表中所有内容
过程/效果	所有记录一条一条删除到删完	相当于保留mysql表的结构，重新创建了这个表 -》 相当于新表
速度/效率	慢	相对快一些
日志	有日志记录-〉允许Rollback回滚操作	truncate删除后不记录mysql日志 -> 不可以恢复数据
其他说明	支持加上WHERE参数，删除特定内容： `delete FROM table1 WHERE your_condition;`	
结论	想要删除部分内容时，选delete 想要有日志记录时，选delete	想要速度快，相当于直接清空某个表时，选truncate

阿里云的RDS

此处整理关于阿里云的RDS的mysql数据库的内容。

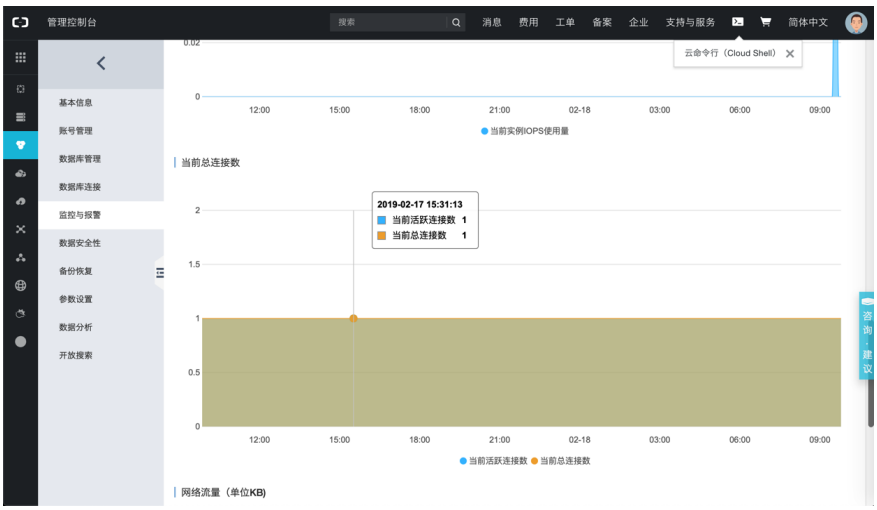
`RDS = Relational Database Service = 关系型数据库服务 = 云数据库`

心得

mysql监控

阿里云的RDS的mysql中，有自己的监控工具：

其中包括当前mysql的连接数：



备份阿里云的RDS的mysql数据库

此处去把，只能内网访问的阿里云的RDS的mysql数据库中某个表的数据，去备份到本地。

其实就是：

- 登录另外一台阿里云的ECS服务器
  - 两者同属一个内网，本身有互相访问的权限
- 然后去执行 `mysqldump` 去导出备份即可
  - 且（由于数据量很大，所以）用了 `gzip` 压缩以减小文件体积

具体命令是：

```
mysqldump --host=xxx.mysql.rds.aliyuncs.com --port=port --u
```

或 `-u` 的用户名和 `-p` 的密码的参数重去掉空格的写法：

```
mysqldump -h xxx.mysql.rds.aliyuncs.com -P port -B dbname -
```

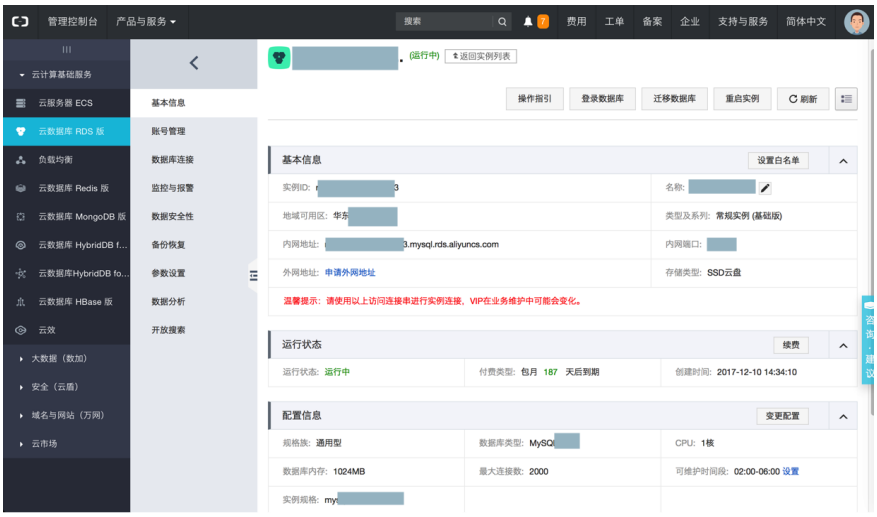
常见问题

无法连接RDS

之前用

```
mysql -h xxx.mysql.rds.aliyuncs.com -P xxx -B xxx -uroot -p
```

结果：无法连上阿里云的RDS的MySQL数据库



原因：给参数时 `-uroot` 是错误的。

根本原因：给的参数是之前参考了 `mysqldump` 的写法

解决办法：把 `-uroot` 改为 `-u root` ，即可。

附录：

完整命令行是：

- 只登录，不选择指定哪个数据库

```
mysql -h xx-xxx.mysql.rds.aliyuncs.com -u root -p
```

- 登录，且指定使用具体的数据库

```
mysql -h xx-xxx.mysql.rds.aliyuncs.com -u root -p dbr
```

连接报错：pymysql.err.OperationalError 2003 Can't connect to MySQL server on mysql.rds.aliyuncs.com Errno 61 Connection refused

此处Flask的python中pymysql去连接远程mysql数据库 `xxx.mysql.rds.aliyuncs.com` 出错：

```
  File "/Users/crifan/.local/share/virtualenvs/robotDemo-HX
    raise exc
pymysql.err.OperationalError: (2003, "Can't connect to MyS(
```

原因：

这个mysql `xxx.mysql.rds.aliyuncs.com` 是阿里云的RDS数据库专用服务器，该名字是阿里云内网才能识别的名字，外网无法访问，只能内网（同处一个区域的服务器）访问。

解决办法：

- 开通外网访问权限
  - 此处，为了更加安全考虑，就不去开放外网访问权限了
- 改为内网（甚至RDS本机）访问
  - 阿里云后台可以针对RDS数据库申请外网IP，再去加上IP白名单，也可以实现很好的安全控制
  - 所以采用：把服务器中的mysql，去用（登录阿里云ECS，和RDS数据库服务器同属于一个内网）mysqldump去导出子表，然后导入到Mac本地刚建mysql数据库，然后再去把配置改为：

```
config = {
    #'host': 'xxx.mysql.rds.aliyuncs.com',
    'host': "127.0.0.1",
    'port': 3306,
    'user': 'root',
    'password': 'xxxx',
    'db': 'yourDB',
    'charset': 'utf8',
}
```

- 即可正常连接

附录

下面列出相关参考资料。

参考资料

- 【已解决】用Sequel给远程mysql中新增字段导致子表卡死
- 【已解决】mysql中批量导出某个数据库的多个table表的数据
- 【已解决】mysql工具sequel中mysql运行insert但部分语句失效
- 【已解决】mysql中一次性删除多个记录
- 【未解决】数据库设计中用到的实体关系
- 【已解决】mysql中如何保存字符串枚举类型的值
- 【记录】下载安装试用mysql workbench
- 【已解决】pymysql中如何用select和where查询到对应的数据
- 【已解决】mysql中操作表的字段名时是否一定要用反引号括起来
-
- 【已解决】把阿里云上只能内网访问的mysql数据库备份到Mac本地
- 【已解决】pymysql连接出错：pymysql.err.OperationalError 2003 Can't connect to MySQL server on mysql.rds.aliyuncs.com Errno 61 Connection refused
- 【已解决】pyspider中运行result_worker出错：ModuleNotFoundError No module named mysql
- 【已解决】mysql中支持的字符编码字符集以及utf8的写法
- 【记录】Sequel中新建mysql数据库并新建表和相应字段
- 【已解决】Sequel去删除mysql的表出错：MySQL said: Cannot delete or update a parent row: a foreign key constraint fails
- 【已解决】Mac中启动mysql出错：ERROR The server quit without updating PID file
- 【已解决】Mac中mysql中创建数据库出错：ERROR 1006 HY000 Can't create database errno 230174752
- 【已解决】Mac中mysqld去重新初始化mysql出错：mysqld Can't create directory /usr/local/mysql-5.7.22-macos10.13-x86_64/data/ Errcode 17 File exists
- 【已解决】Mac中如何彻底卸载之前通过dmg安装的mysql
- 【已解决】Mac本地安装5.7版的MySQL数据库并正常登录和访问mysql
- 【已解决】Mac本地搭建mysql数据库并导入已备份出的sql文件
- 【已解决】Python的Flask中pymysql中mysql返回分页查询结果
- 【已解决】pyspider中pymysql中insert失败且except部分代码没有执行
- 【已解决】pymysql中connect异常时有哪些错误类型
- 【已解决】python中mysql异常时获取Exception的code
- 【已解决】Mac中Sequel连接本地Mysql出错：Unable to connect to host 127.0.0.1, or the request timed out
- 【已解决】PySpider中保存数据到mysql
- 【已解决】Python3中选择合适的mysql的连接器或驱动

- 【已解决】Python中如何操作mysql
- 【调研】Python的合适本地存取和查看的数据库
- 【已解决】Mac中dmg安装mysql后登录出错：ERROR 2002 HY000 Can't connect to local MySQL server through socket /tmp/mysql.sock 61
- 【基本解决】如何恢复Mac中mysql中的mysql表
- 【已解决】阿里云RDS的Mysql数据库连接不上没反应
- 【已解决】RDS数据库中的BINLOG和SLOWLOG
- 【已解决】Mac本地给mysql数据库改名
- 【已解决】mysql中清空某个表的内容
- 【已解决】Mac本地搭建mysql数据库并导入已备份出的sql文件
- 【基本解决】如何恢复Mac中mysql中的mysql表
-
- MYSQL使用mysqldump导出某个表的部分数据 - CSDN博客
- mysqldump 导出某几张表-逆风草-51CTO博客
- 用mysqldump备份和恢复指定表的方法*Mysql*脚本之家
- MySQL mysqldump数据导出详解 - pursuer.chen - 博客园
- 数据库 — flask mega-tutorial 1.0 documentation
- MySQL UPDATE 查询 | 菜鸟教程
- mysql 修改表中某一列的值 - CSDN博客
- MySQL更新某个字段的值为原来的值加1 (非auto_increament) - CSDN博客
- Mysql命令update set：修改表中的数据_C语言中文网
- SHOW CREATE TABLE - MariaDB Knowledge Base
- MySQL :: MySQL 8.0 Reference Manual :: 13.7.6.10 SHOW CREATE TABLE Syntax
- mysql - How to generate a create table script for an existing table in phpmyadmin? - Stack Overflow
- mysql从mysqldump备份中恢复某张表内容 - CSDN博客
- mysql从全库备份中恢复某个表 - CSDN博客
- MySQL单表恢复方法 - billy鹏 - 博客园
- MySQL :: MySQL 8.0 Reference Manual :: 9.3 Keywords and Reserved Words
- MySQL :: MySQL 8.0 Reference Manual :: 5.1.10 Server SQL Modes
- 最棒的10款MySQL GUI工具-阿里云开发者社区
- 最好的MySQL客户端推荐
- MySQL体系结构与存储引擎（一）
- MySQL体系结构和存储引擎概述 - WilburXu - 博客园
- [玩转MySQL之一]MySQL体系架构简介 - 知乎
- MySQL :: MySQL 8.0 Reference Manual
-