

# 数据库 day01 随堂笔记

## 今日内容

- 数据类型和约束
- 通过 SQL 语句实现一系列操作
  - 对数据库的操作
  - 对数据表的操作
  - 对数据的操作(增/删/改)

## 数据类型和约束

### 数据类型

数据类型: 对于填入的数据值本身进行控制, 保证数据准确性

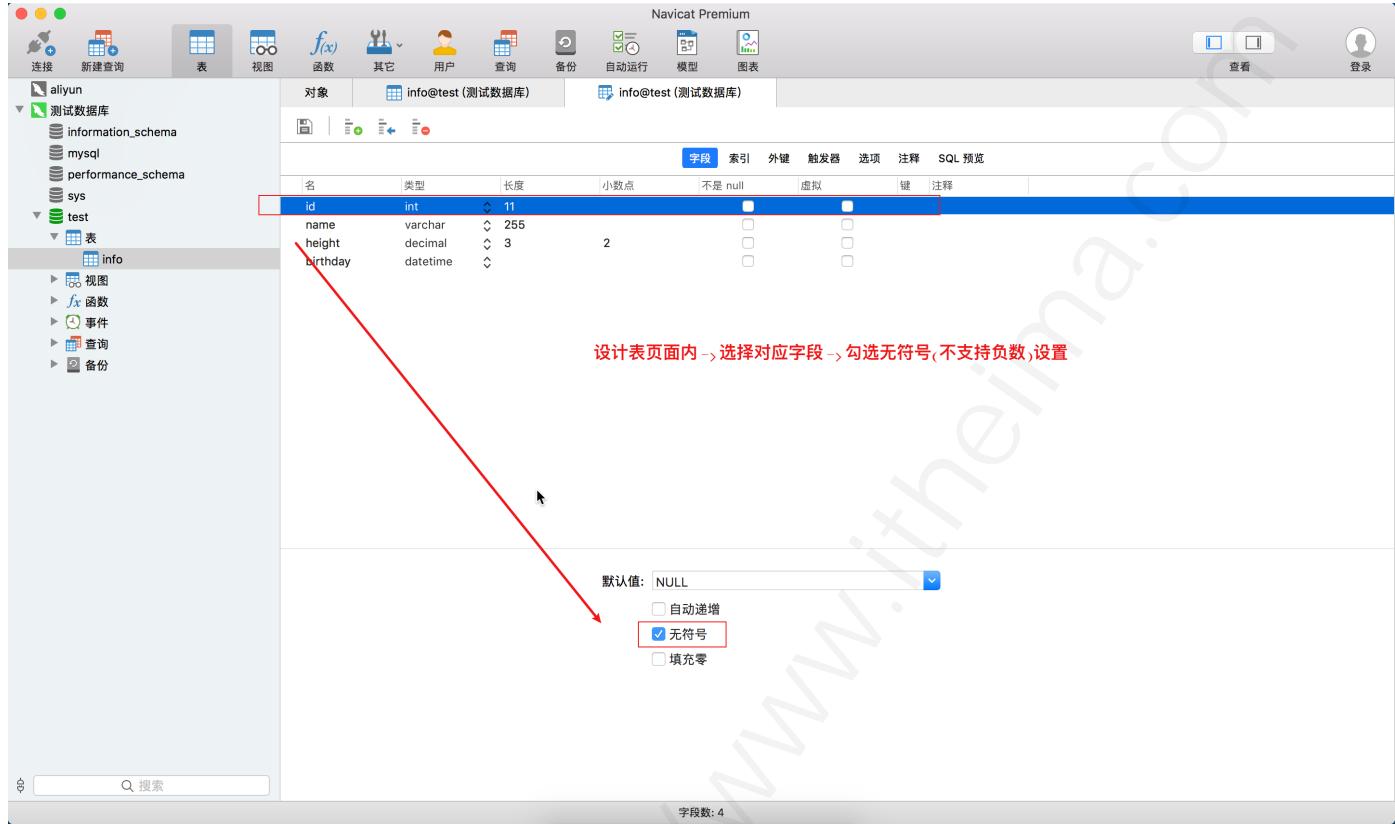
**整数:** int, 有符号范围 (-2147483648 ~2147483647) , 无符号 (unsigned) 范围 (0 ~ 4294967295)

**小数:** decimal, 例如: decimal(5,2) 表示共存5位数, 小数占2位, 整数占3位

**字符串:** varchar, 范围 (0~65535) , 例如: varchar(3) 表示最多存3个字符, 一个中文或一个字母都占一个字符

**日期时间:** datetime, 范围 (1000-01-01 00:00:00 ~ 9999-12-31 23:59:59) , 例如: '2020-01-01 12:29:59'

## 整数无符号设置



## 约束

约束: 对于整张数据表进行限制, 确保对应字段的所有数据符合设计要求

**主键 (primary key)** : 能唯一标识表中的每一条记录的属性组

**非空 (not null)** : 此字段不允许填写空值

**唯一 (unique)** : 此字段的值不允许重复

**默认值 (default)** : 当不填写此值时会使用默认值, 如果填写时以填写为准

**外键 (foreign key)** : 一个表中的一个字段引用另一个表的主键

# 主键

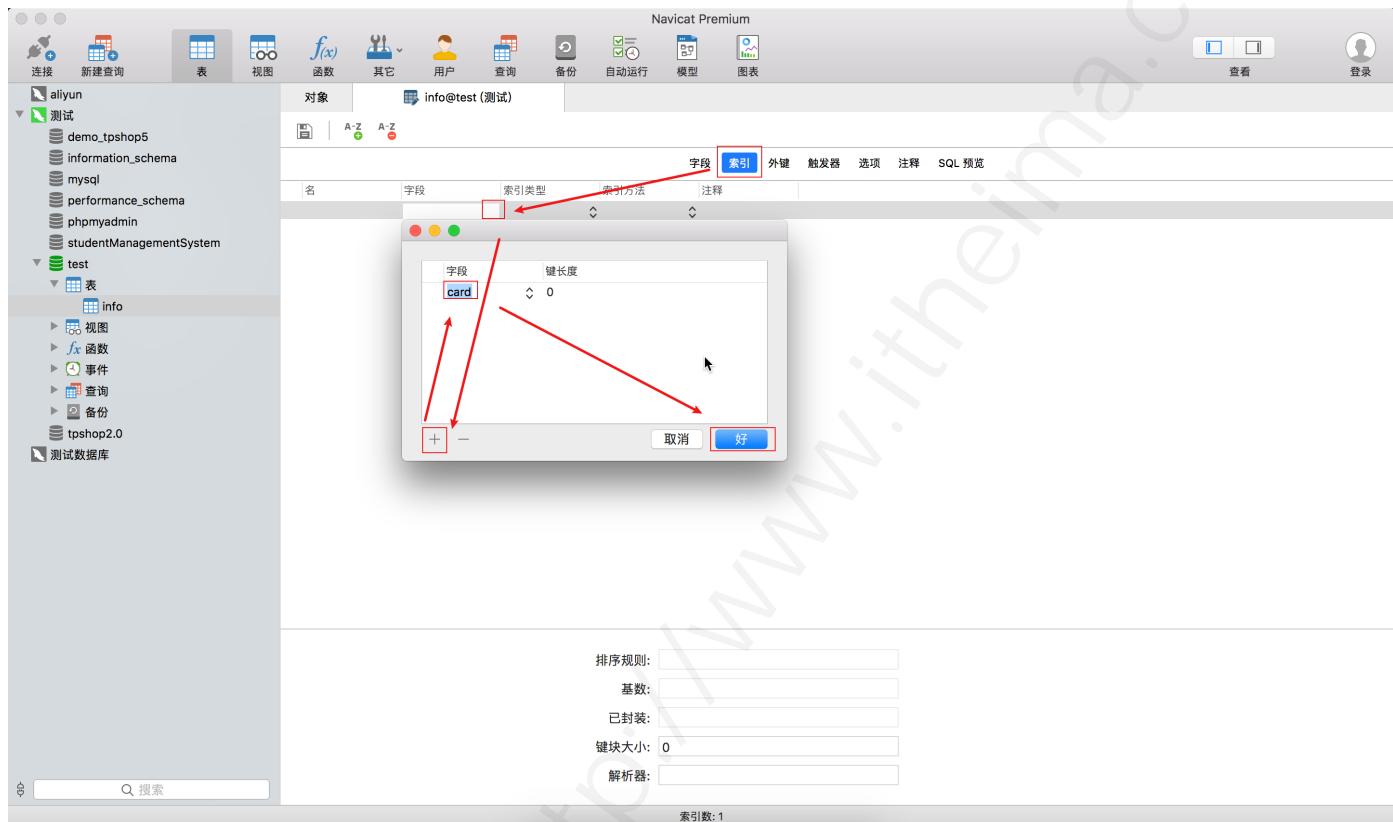
The screenshot shows the Navicat Premium interface for MySQL database management. A new table named 'test' is being created under the 'test' schema. In the table definition window, the 'id' column is selected, and its properties are displayed. The '类型' (Type) is set to 'int'. The '不能为空' (Cannot be Null) checkbox is checked, indicating it is a primary key. Below the table definition, a note states: '主键: 主要用于保证数据表内的数据每一条的顺序是固定的, 不会由于删除或增加数据, 而导致数据乱序!' (Primary key:主要用于保证数据表内的数据每一条的顺序是固定的, 不会由于删除或增加数据, 而导致数据乱序!). Another note says: '注意: 一般的一张数据表中只需要有一个主键!!!!' (Note: Generally, one table only needs one primary key!!!!). At the bottom, there is a section for default values, with checkboxes for '自动递增' (Auto-increment) and '无符号' (Unsigned), both of which are checked.

# 不为空

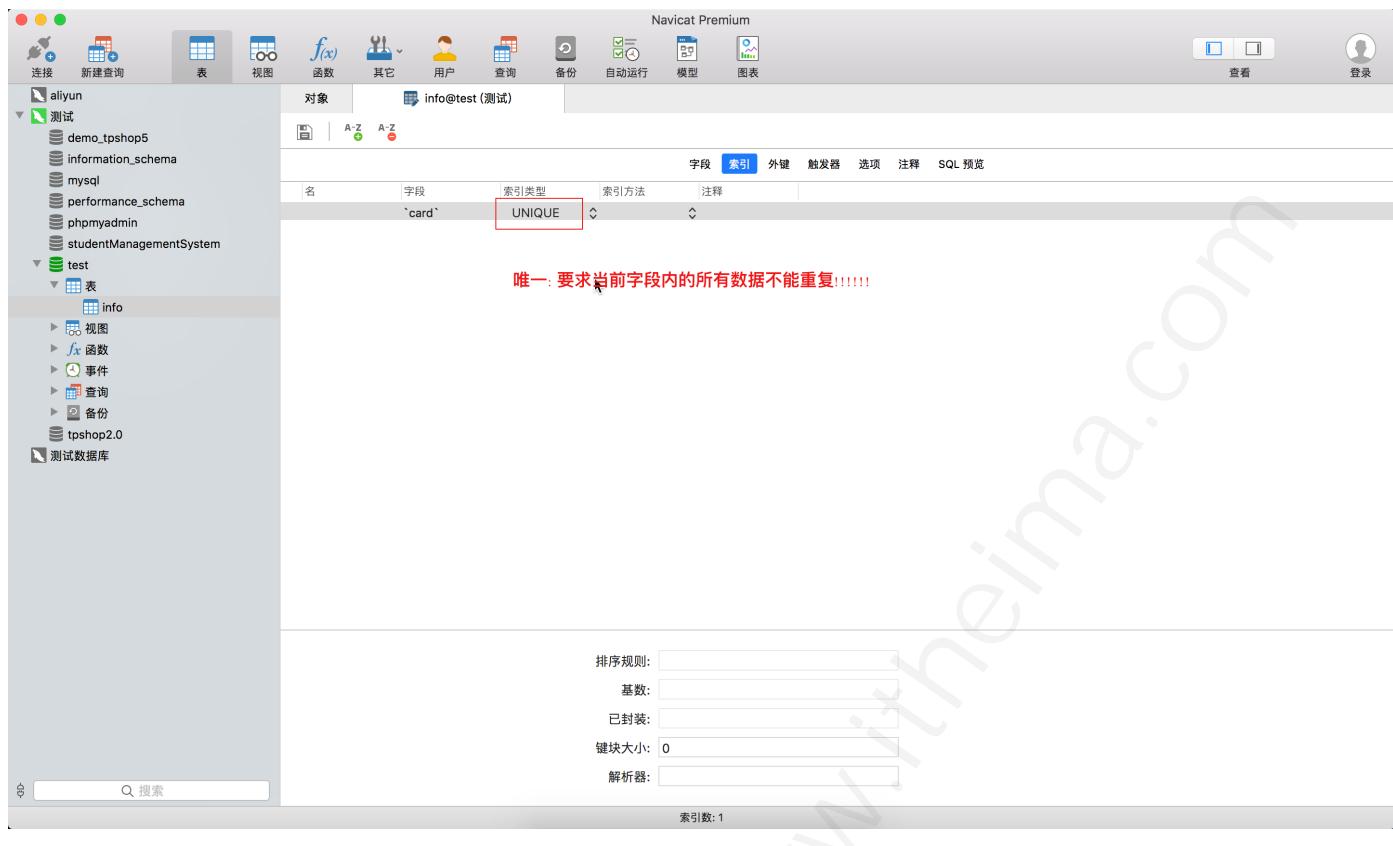
The screenshot shows the Navicat Premium interface for MySQL database management. A new table named 'info' is being created under the 'test' schema. In the table definition window, the 'name' column is selected, and its properties are displayed. The '长度' (Length) is set to 255, and the '不能为空' (Cannot be Null) checkbox is checked, indicating it is a non-null constraint. Below the table definition, a note states: '不为空: 设置字段内数据必须有值, 不能为空!' (Not null: The data in the field must have a value, cannot be empty!). At the bottom, there are settings for default values, character sets ('字符集: latin1'), collations ('排序规则: latin1\_swedish\_ci'), and binary options.

# 唯一

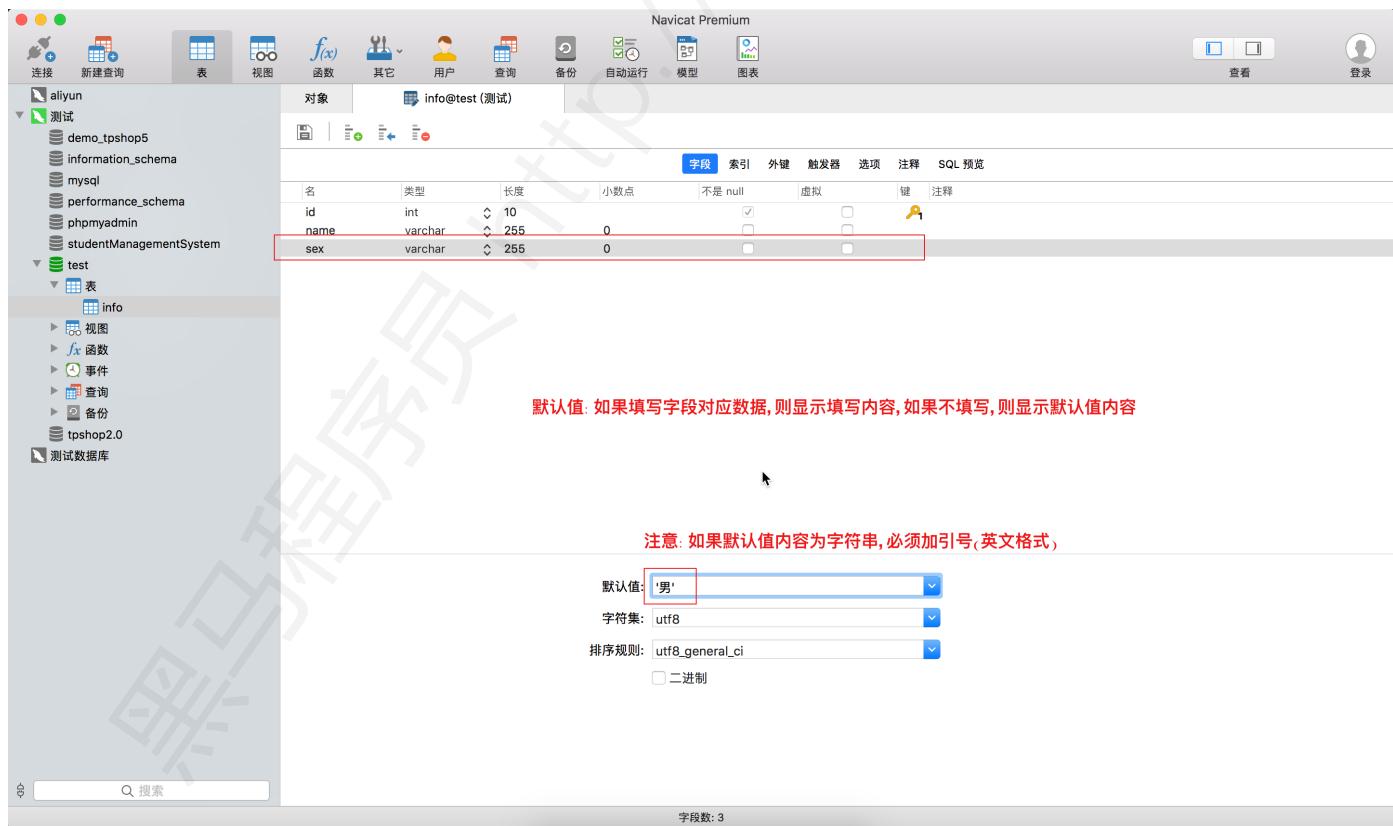
索引 -> 字段



选择索引类型为: unique(唯一)

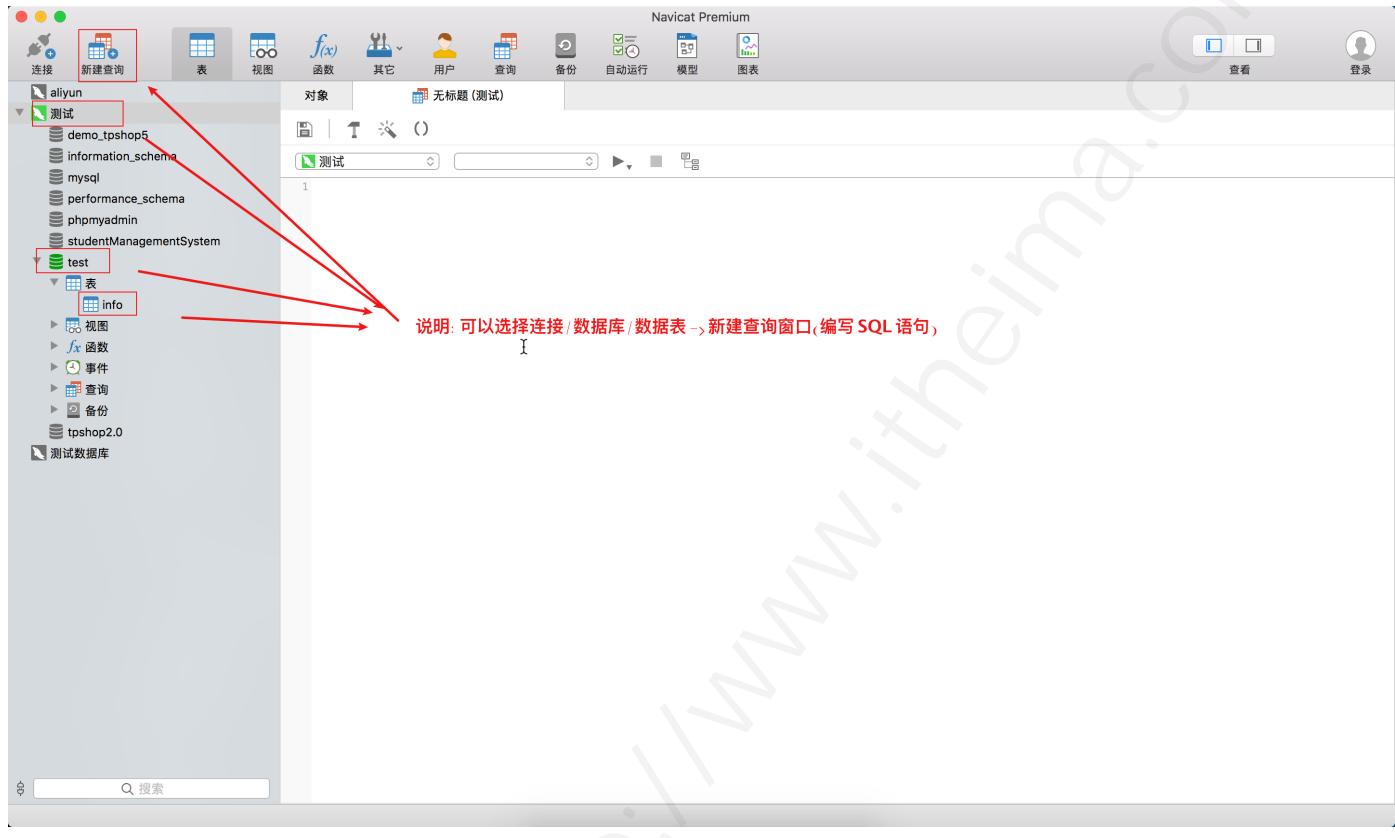


## 默认值

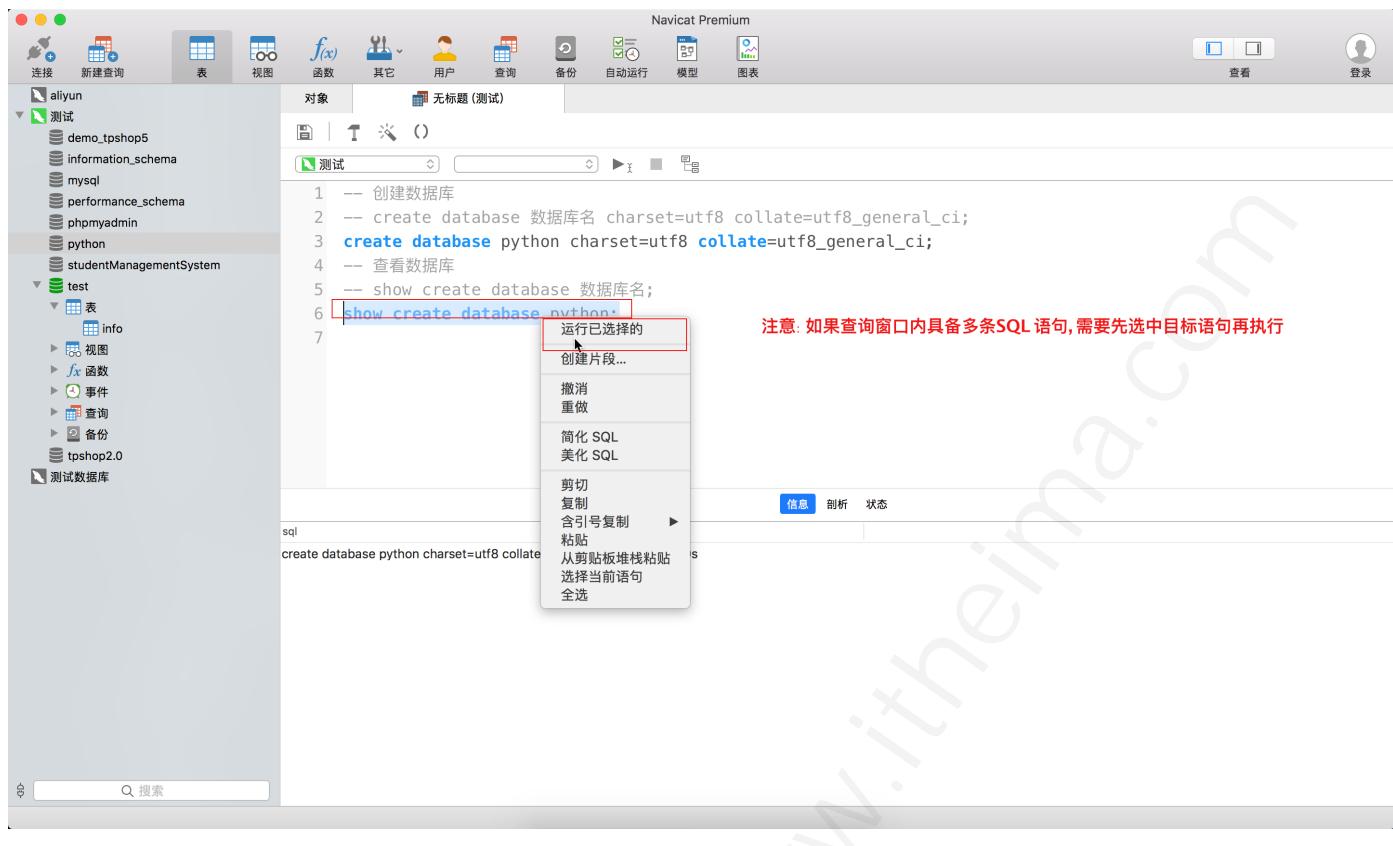


# SQL 语句

## 查询窗口的开启方法



## 单语句运行方法



## 数据库操作

### 创建数据库

```
-- 创建数据库
-- create database 数据库名 charset=utf8 collate=utf8_general_ci;
create database python charset=utf8 collate=utf8_general_ci;
-- 查看数据库
-- show create database 数据库名;
show create database python;
```

### 使用数据库

```
-- 使用数据库(切换数据库)
-- use 数据库名;
use python;
-- 查看当前数据库: database() 是 SQL 的内置函数, 括号不能省略!
select database();
```

## 修改数据库

```
-- 修改数据库
-- 创建
create database testpython charset = gb2312;
-- 修改
-- alter database 数据库名
-- default character set 编码格式
-- default collate 排序规则;
alter database testpython
default character set utf8mb4
default collate utf8mb4_general_ci;
```

## 删除数据库和查看所有数据库

```
-- 删除数据库
-- drop database 数据库名;
drop database python;
-- 查看所有数据库
show databases;
```

# 重点: 数据库备份

## 应用场景

**说明:** 在测试工作中,为了防止对数据库产生错误操作,或产生垃圾数据,都需要在操作前,适当对数据库进行备份操作.

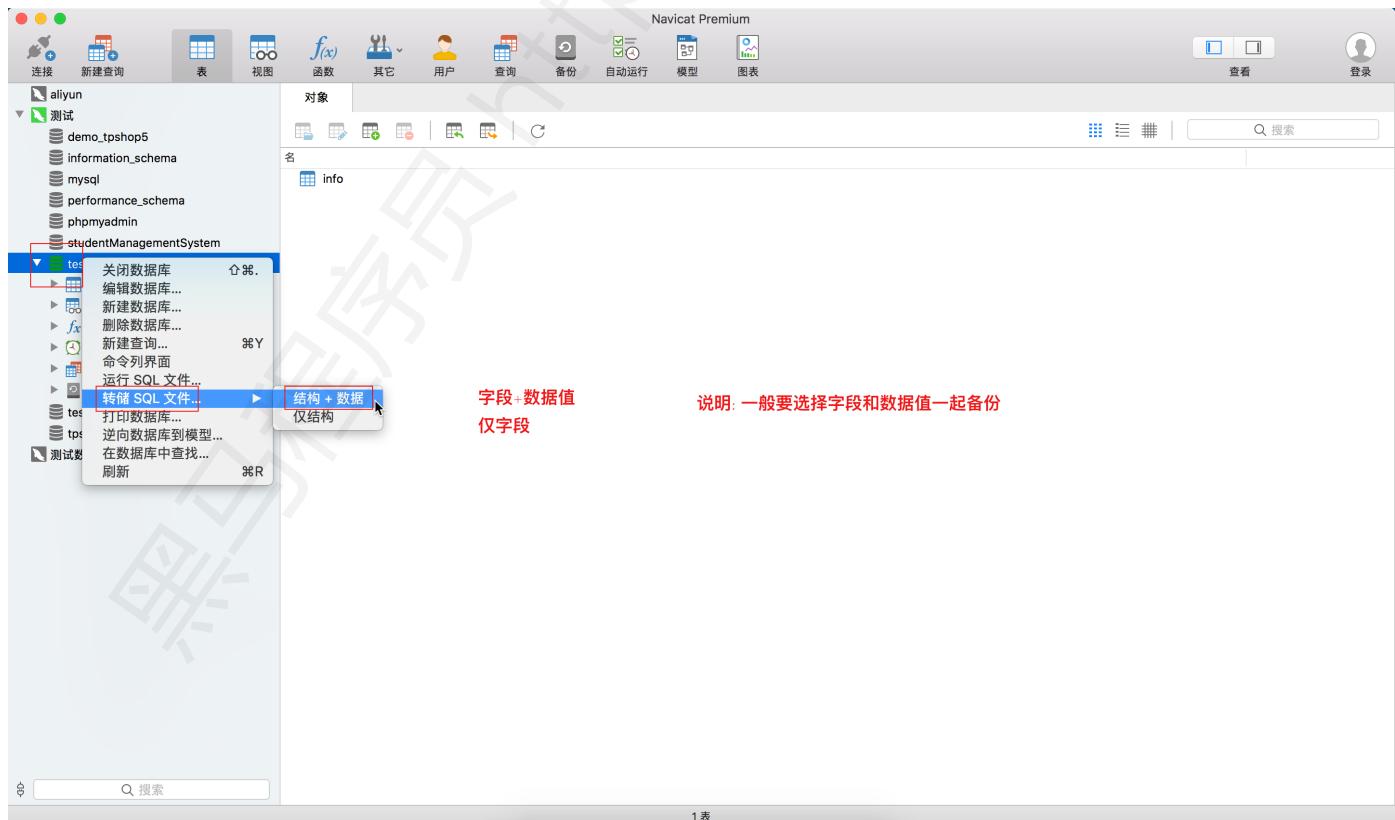
**垃圾数据:** 例如在自动化测试中,对注册模块操作生成的所有数据,属于典型的垃圾数据,应该清理

## 备份方法

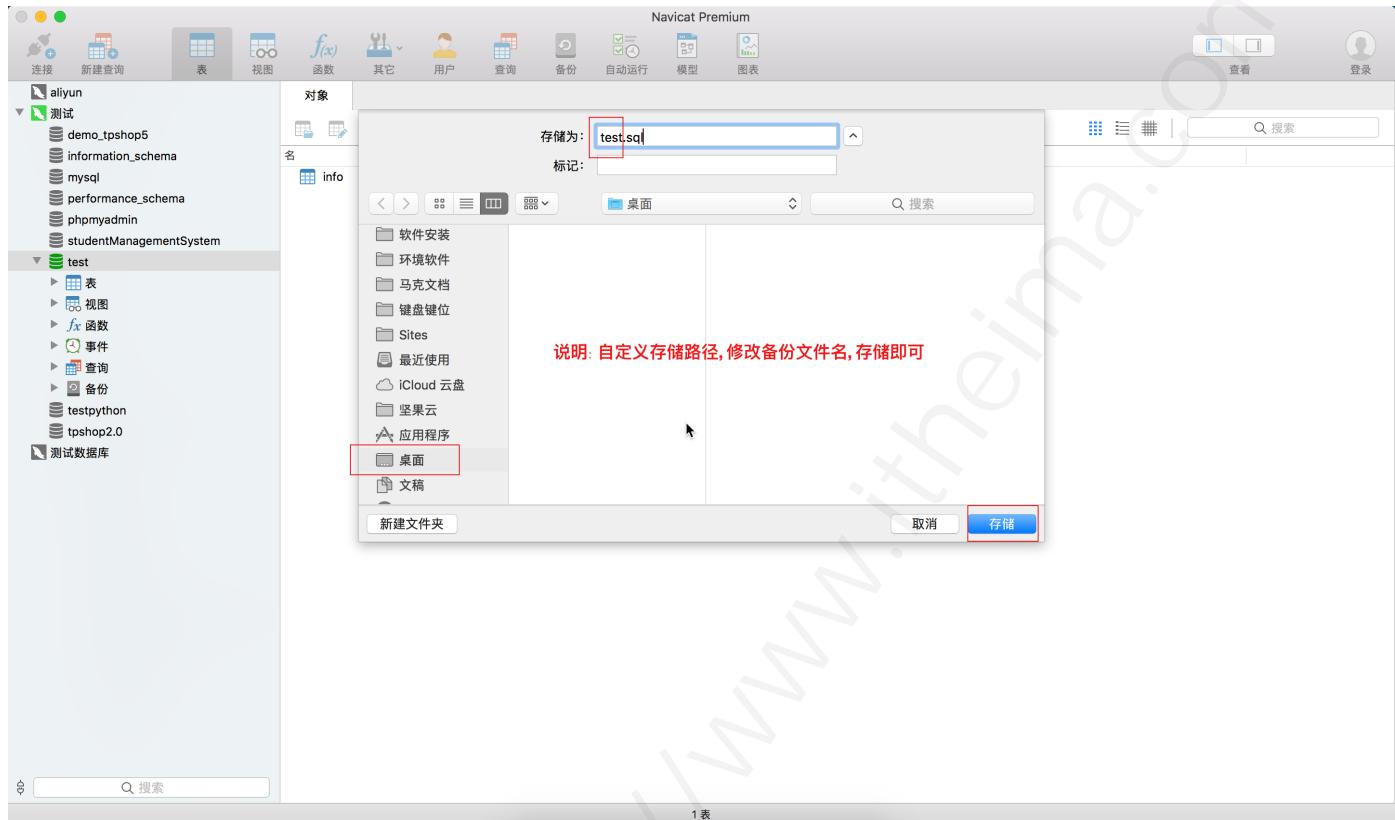
### 利用工具

#### 备份步骤

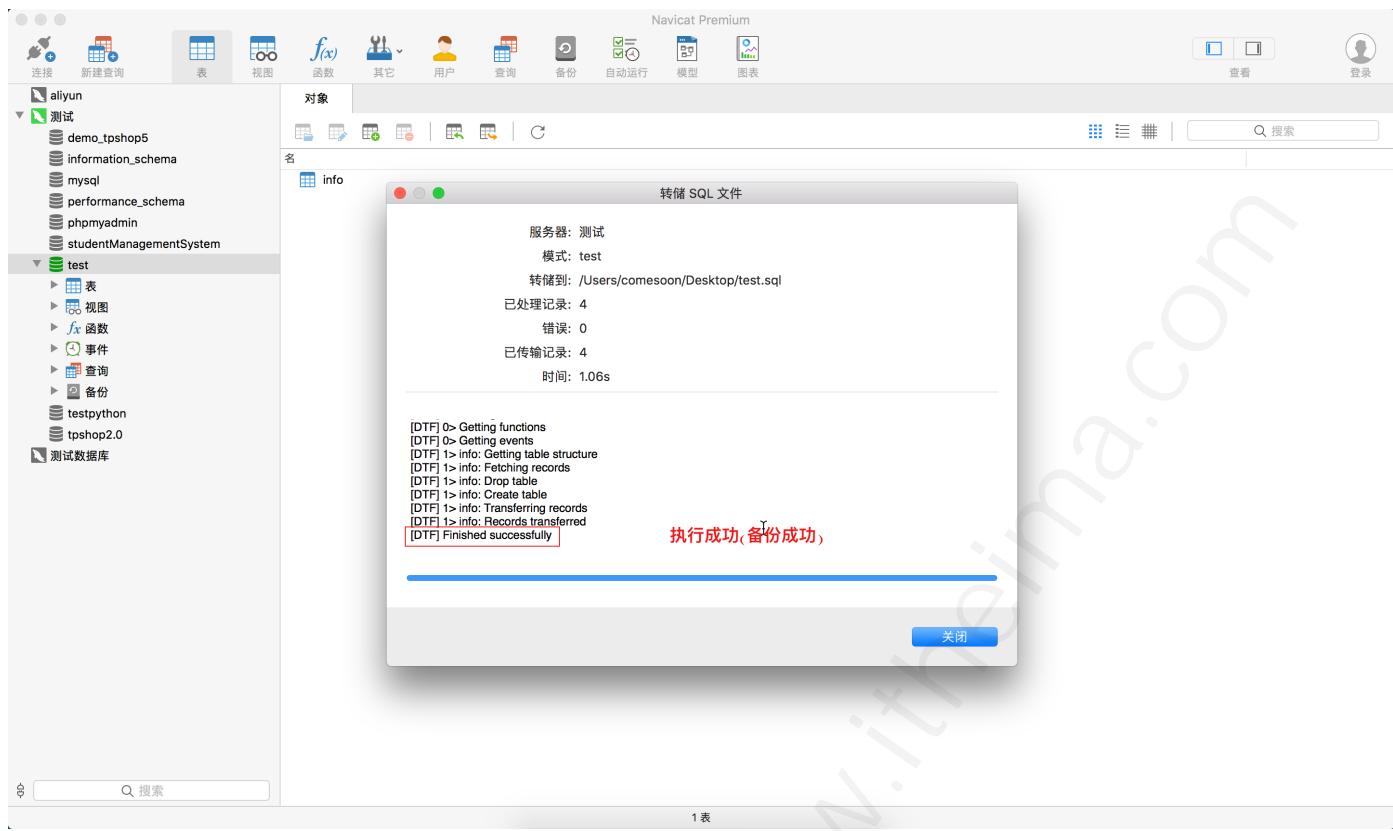
数据库 -> 转储 SQL 文件 -> 结构+数据



## 自行选择存放位置

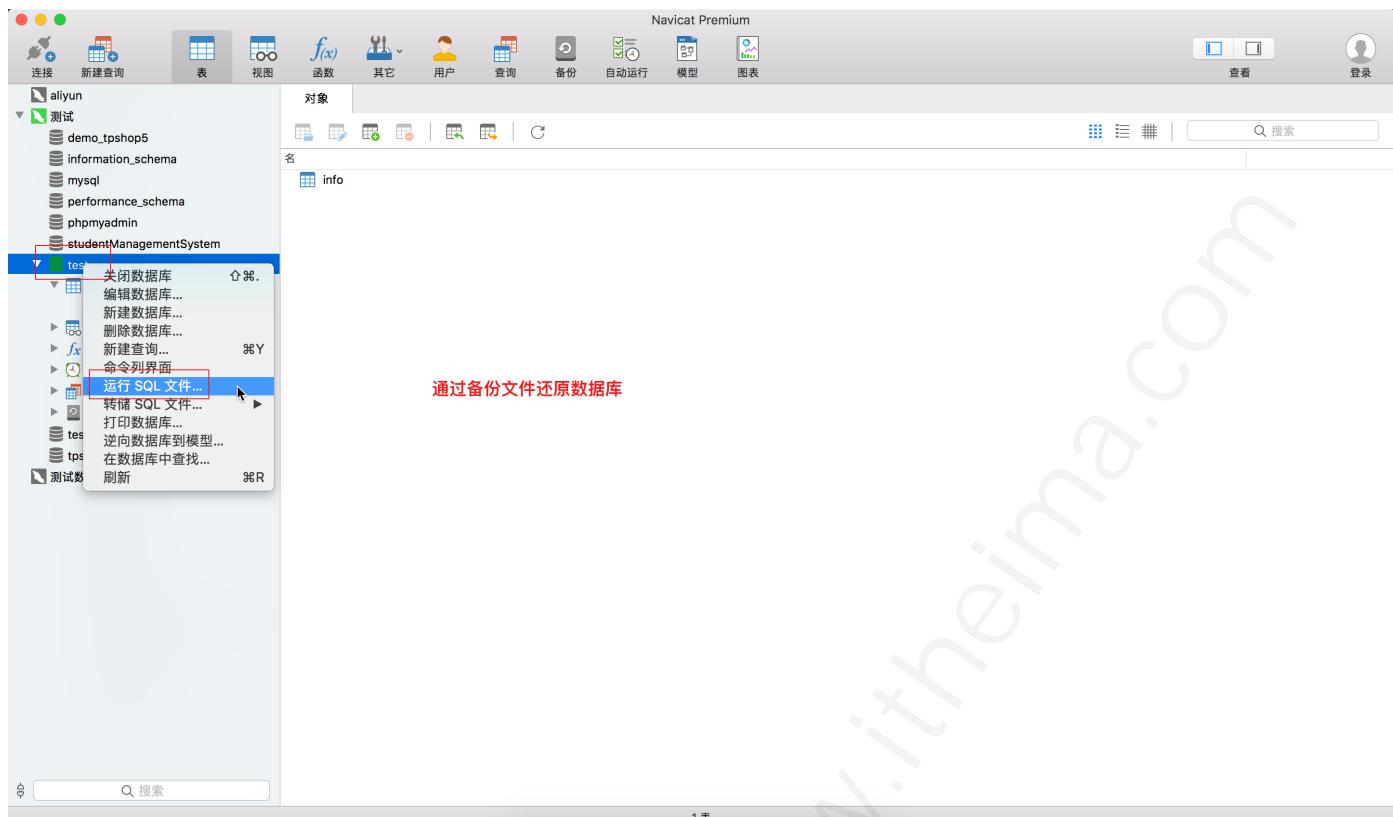


备份结束

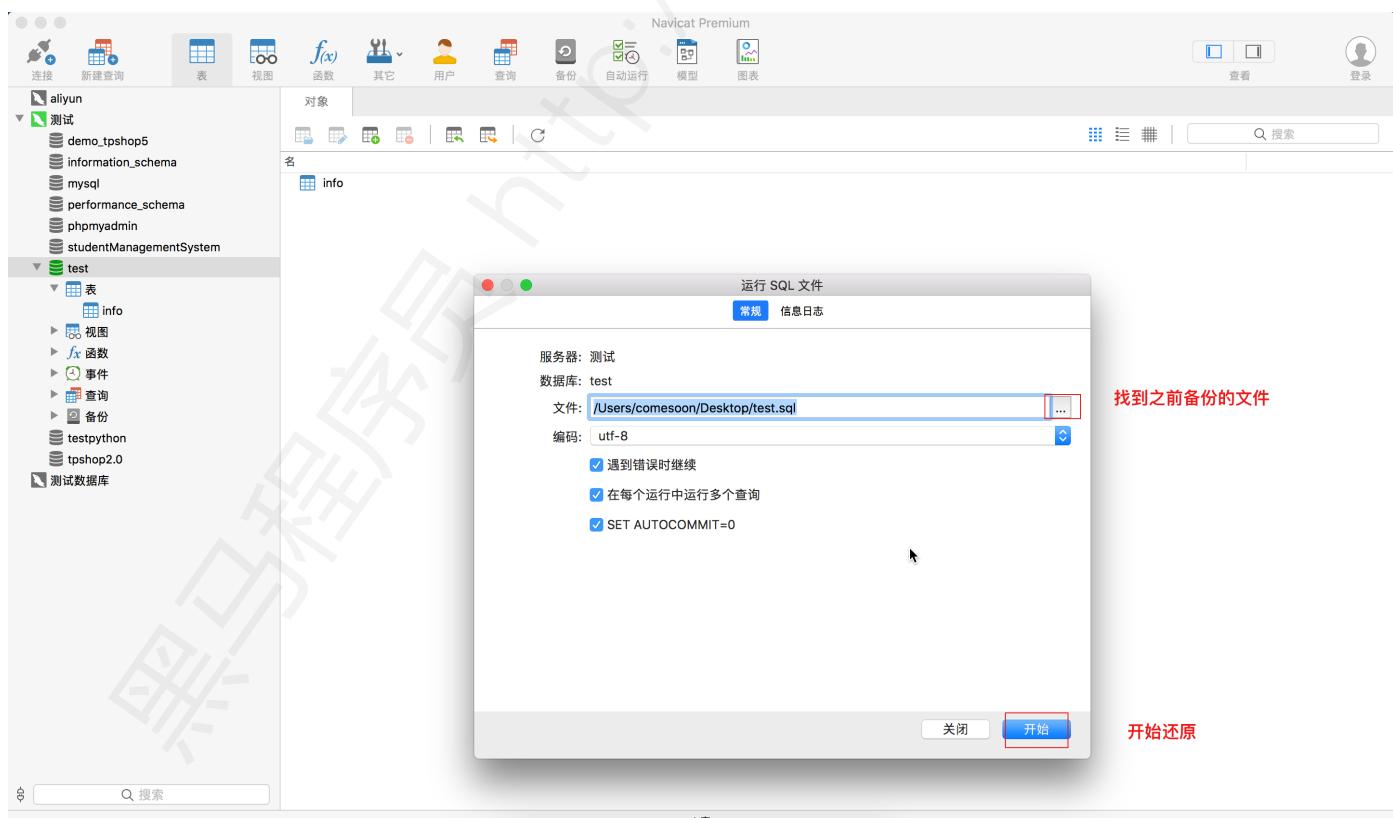


## 还原操作

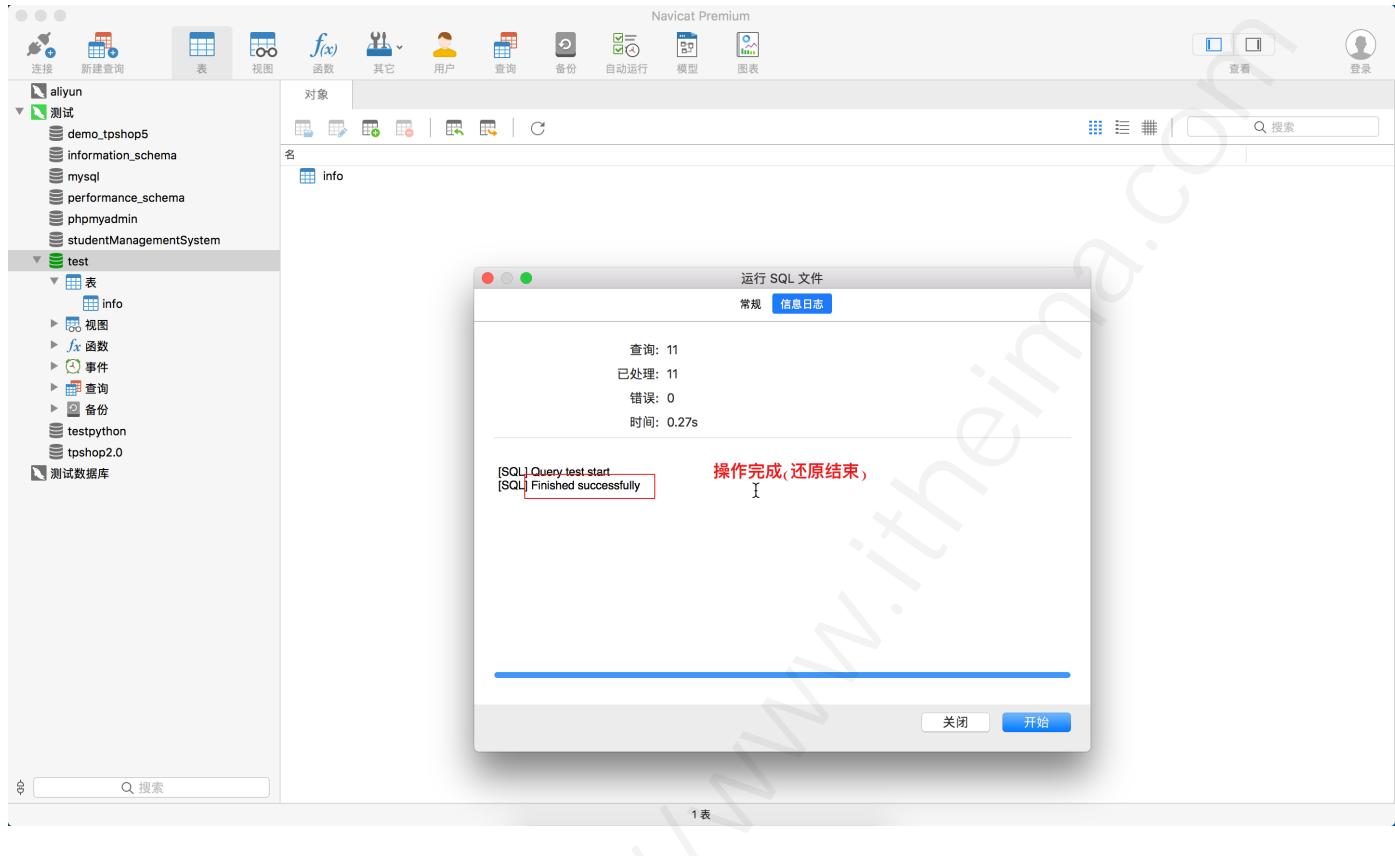
数据库 -> 运行 SQL 文件



## 选择备份文件



## 还原结束



## 扩展: 使用命令备份

**注意:** 命令是不需要连接到数据库以后执行的!(非 mysql> 模式)

mysql> 为 SQL 语句编写模式, 非 Linux 命令行模式

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[admin@localhost ~]$ mysql -uroot -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

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> ls
->;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
for the right syntax to use near 'ls' at line 1
mysql>
```

mysql>  
此模式仅支持 SQL 语句!

## 命令备份与还原数据库操作

```
[admin@localhost ~]$ mysql -uroot -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

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> ls
->;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
for the right syntax to use near 'ls' at line 1
mysql> exit;
Bye
[admin@localhost ~]$ mysqldump -uroot -p test > test.sql
Enter password:
[admin@localhost ~]$ ls
demo dump.rdb test.sql 公共 模板 视频 图片 文档 下载 音乐 桌面
[admin@localhost ~]$ mysql -uroot -p test < test.sql
Enter password:
[admin@localhost ~]$
```

备份命令: mysqldump -u 数据库用户名 -p 目标数据库名 > 备份文件名.sql

还原命令: mysql -u 数据库用户名 -p 目标数据库名 < 备份文件名.sql

注意: 需要根据提示输入数据库密码, 执行数据库备份与还原操作!

## 数据表操作

### 创建表

```
-- 创建表
-- create table 表名(
-- 字段名 类型 约束,
-- 字段名 类型 约束
-- ...
-- )
-- 简单创建
create table stu(
    name varchar(5)
```

```
);

-- 完整创建
-- unsigned : 无符号
-- primary key : 主键
-- auto_increment : 自动增长
create table students(
    id int unsigned primary key auto_increment,
    name varchar(20),
    age int unsigned,
    height decimal(5,2)
);
```

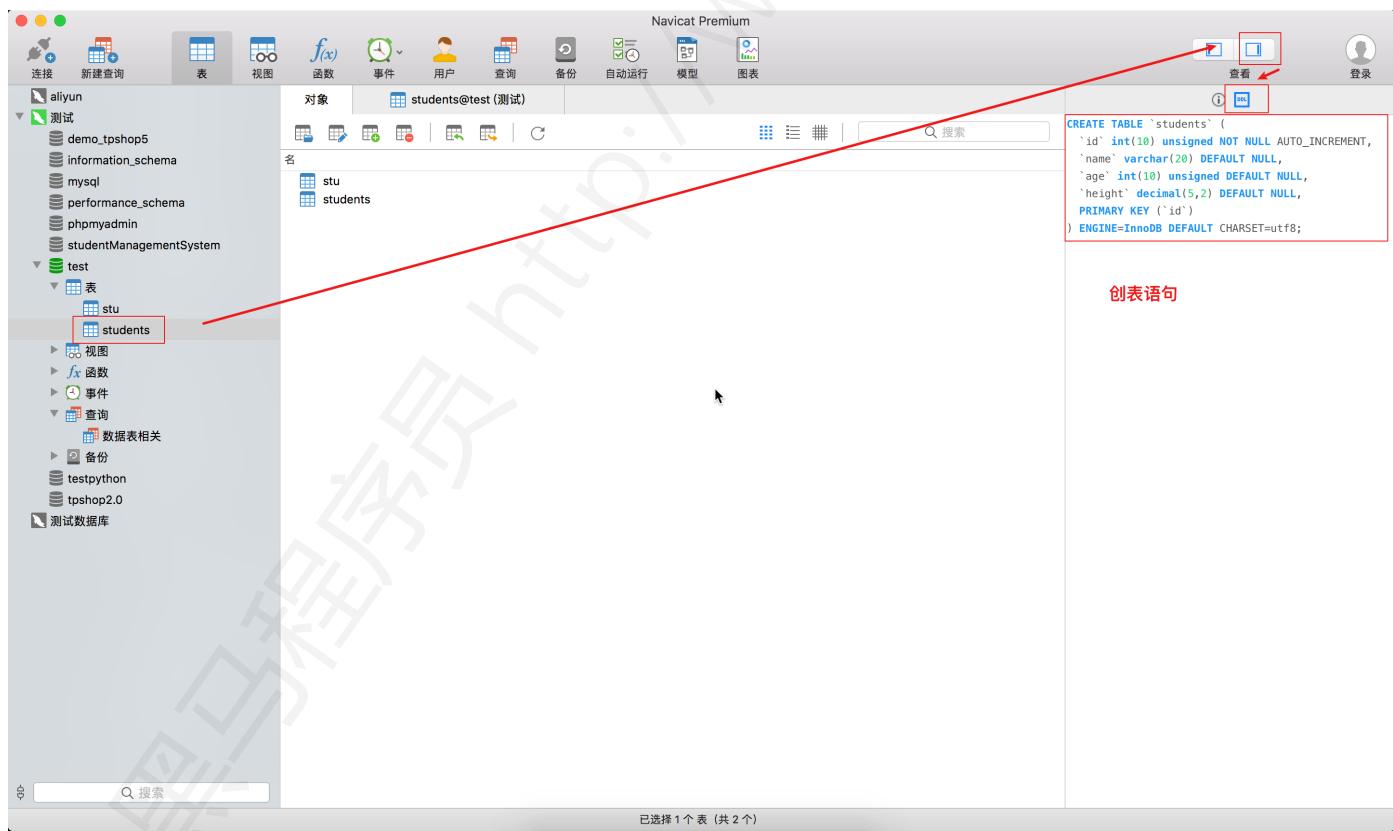
## 查看表

```
-- 查看表信息
-- show create table 表名;
show create table students;
-- 执行结果
-- CREATE TABLE `students` (
--   `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
--   `name` varchar(20) DEFAULT NULL,
--   `age` int(10) unsigned DEFAULT NULL,
--   `height` decimal(5,2) DEFAULT NULL,
--   PRIMARY KEY (`id`)
-- ) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

## 扩展：判断表是否存在移除再创建

```
-- 扩展：判断表是否存在，存在时先删除再创建
-- drop table : 删除表
-- if exists students : 如果 students 存在
drop table if exists students;
create table students(
    id int unsigned primary key auto_increment,
    name varchar(20),
    age int unsigned,
    height decimal(5,2)
);
```

## 扩展：通过 Navicat 工具获取创表语句的方法



## 查看表结构和删除表

```
-- 查看表结构(字段)
-- desc 表名;
desc students;

-- 删除表
-- drop table 表名;
drop table students;
```

## 数据操作

### 增加数据

#### 增加一行数据

```
-- 增加数据
-- 增加一行数据
-- insert into 表名 values(...)
-- 注意:
-- 1. 数据值需要和表的字段一一对应(数据个数及数据类型)
-- 2. 主键列是自动增长, 插入时需要占位, 通常使用 0 或者 default 或者
null 来占位, 插入成功后以实际数据为准
insert into students values(0, '张三', 28, 1.78);

-- 增加部分值
-- insert into 表名(字段1,...) values(值1,...)
-- 注意: 值的顺序与给出的字段顺序对应
insert into students(name, height) values('李四', 1.68);
```

## 增加多行数据

```
-- 插入多行数据
-- 方式1：将单行插入语句，多句执行，每句分号隔开
insert into students values(0, '王五', 28, 1.78);
insert into students(name, height) values('赵六', 1.68);
-- 方式2：在插入单行数据的语法基础上，将 value 后边的数据进行多组化处理
-- insert into 表名 values(...),(...)... 
-- insert into 表名(列1,...) values(值1,...),(值1,...)...
insert into students values(0, '王五1', 29, 1.78),(0, '王五2',
30, 1.78);
insert into students(name, height) values('赵六1', 1.78),('赵六
2', 1.88);
```

## 修改数据

```
-- 修改数据
-- update 表名 set 列1=值1,列2=值2... where 条件
-- 注意：where 不能省略，否则会修改整列数据
update students set age=48 where id=9;
```

语句作用对应

The screenshot shows the Navicat Premium interface with a database connection named 'aliyun'. In the left sidebar, under the '测试' (Test) schema, there is a 'test' database which contains a 'stu' table and a 'students' table. The 'students' table is currently selected and displayed in the main grid. The grid has columns: id, name, age, and height. The data shows 9 rows of student information. A specific row, where id=9 and name='王五', is highlighted with a red box. Below the grid, a SQL update statement is entered into the query editor: 'update students set age=48 where id=9;'. Three red arrows point from the text '定表' (Table), '定列' (Column), and '定行' (Row) to the corresponding parts of the SQL statement: 'students' (table), 'age=48' (column), and 'id=9' (row).

## 删除数据

```
-- 删除数据
-- delete from 表名 where 条件;
-- 注意: where 不能省略, 否则会删除全部数据
delete from students where id=6;
```

## 扩展 1: 逻辑删除

**逻辑删除：**对于重要的数据，不能轻易执行 `delete` 语句进行删除。因为一旦删除，数据无法恢复，这时可以进行逻辑删除。

- 1、给表添加字段，代表数据是否删除，一般起名 `isdelete`，0代表未删除，1代表删除，默认值为0
- 2、当要删除某条数据时，只需要设置这条数据的 `isdelete` 字段为1
- 3、以后在查询数据时，只查询出 `isdelete` 为0的数据

## 第一步

The screenshot shows the Navicat Premium interface. On the left, the database structure is visible, including the 'test' schema which contains a 'students' table. The 'students' table has columns: id, name, age, height, and isdelete. The 'isdelete' column is highlighted with a red box. Its properties are being edited: type is set to int, length is 1, and the default value is set to 0. A note at the bottom states: "说明: 所谓逻辑删除,就是通过某一特定字段的特定值表示数据是删除(1)或未删除(0)状态".

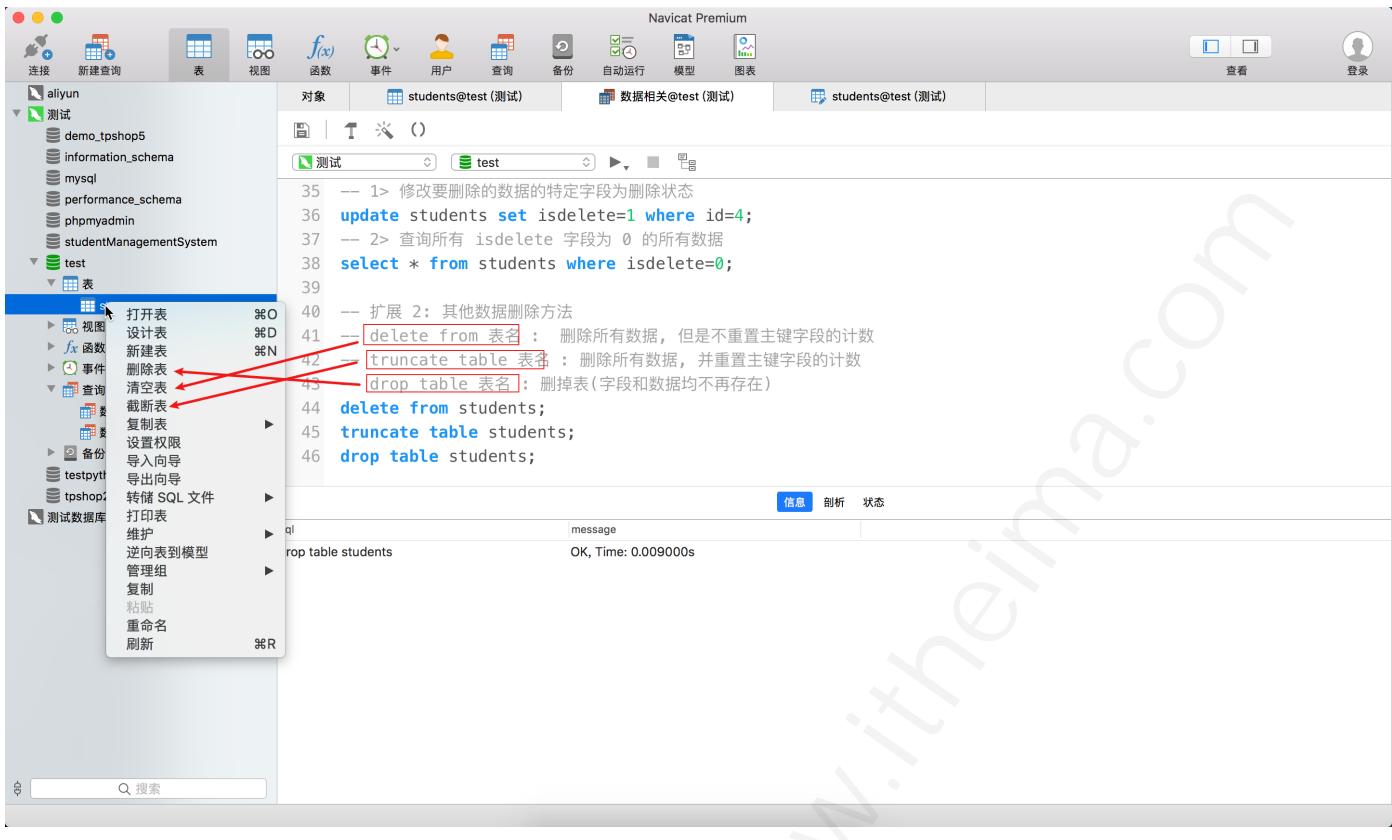
## 第二和第三步

```
-- 扩展 1: 逻辑删除(假删/标记删除)
-- 1> 修改要删除的数据的特定字段为删除状态
update students set isdelete=1 where id=4;
-- 2> 查询所有 isdelete 字段为 0 的所有数据
select * from students where isdelete=0;
```

## 扩展 2: 其他删除数据的方法

```
-- 扩展 2: 其他数据删除方法
-- delete from 表名 : 删除所有数据, 但是不重置主键字段的计数
-- truncate table 表名 : 删除所有数据, 并重置主键字段的计数
-- drop table 表名 : 删掉表(字段和数据均不再存在)
delete from students;
truncate table students;
drop table students;
```

与图形化页面菜单的对应关系



## 今日任务

- 整理今日内容(思维导图/笔记) -> 将来可用为准[优先级最低]
- 预习数据查询内容!!!!!![优先级最高] -> 截止到分页查询