实验 基于 SQL 的数据定义与修改

1. 实验目的

(1) 掌握用 SQL 语句创建数据库、修改数据库属性、删除数据库的方法。

(2) 掌握用 SQL 语句对数据进行增、删、改等操作。

(3) 致敬英雄楷模，立志报效祖国。

2. 实验环境（写清硬件配置和软件版本）

(1) 硬件：

a) 笔记本型号：TUF Gaming FX505GE_FX86FE..

b) 内存：SK Hynix 8G

c) CPU：Inter® Core™ i7-8750H CPU @ 2.20GHz

d) 硬盘：

1. 主硬盘：WDC PC SN520 SDAPNUW-256G-1002
2. 从硬盘：HGST HTS721010A9E630

(2) 操作系统：Windows 10 家庭中文版

(3) 数据库管理系统：PostgreSQL 10

3. 实验内容

(1) 使用 SQL 语句创建数据库、修改数据库属性、删除数据库。

(2) 使用 SQL 语句插入数据、更新数据、删除数据。

4. 实验数据

2019 年 12 月以来，湖北省武汉市部分医院陆续发现了多例有华南海鲜市场暴露史的不明原因肺炎病例，现已证实为2019新型冠状病毒感染引起的急性呼吸道传染病。当地时间 2020 年 3 月 11 日，世界卫生组织总干事谭德塞宣布，根据评估，世卫组织认为当前新冠肺炎疫情可被称为全球大流行（pandemic）。美国约翰斯·霍普金斯大学：截至北京时间 2020 年 9 月 23 日 7 时 23 分，全球新冠确诊病例达 31453048 例，死亡病例为 967347 例；美国是全球疫情最严重的国家，确诊病例达 6890014 例，死亡病例为 200654 例。新型冠状病毒肺炎爆发，面对疫病，英雄的中国人民非但没有被吓倒，反而众志成城、守望相助，树立起必胜的信心，在党和政府的坚强领导下，依赖科学、组织抗击，在短时间内取得抗疫斗争的胜利。学习抗疫英雄，争当时代先锋。本次实验基于

“抗疫英雄数据库”，其中，抗疫英雄表包含英雄编号（aehero_id），名字（aehero_name），性别（aehero_gender）；抗疫措施表包含措施编号（aemeasure_id），措施名称(aemeasure_name)，措施具体内容（aemeasure_detail）；一位抗疫英雄会参与多项抗疫措施的制定，一项抗疫措施会有多位英雄参与制定，措施制定贡献表记录各位英雄参与各项措施制定中的贡献（aehero_deeds）。

5. 实验作业

(1) 在数据库 Antiepidemic 中，使用 SQL 语句创建数据表 aehero(aehero_id, aehero_name, aehero_gender)，设置 aehero_id 为主键。

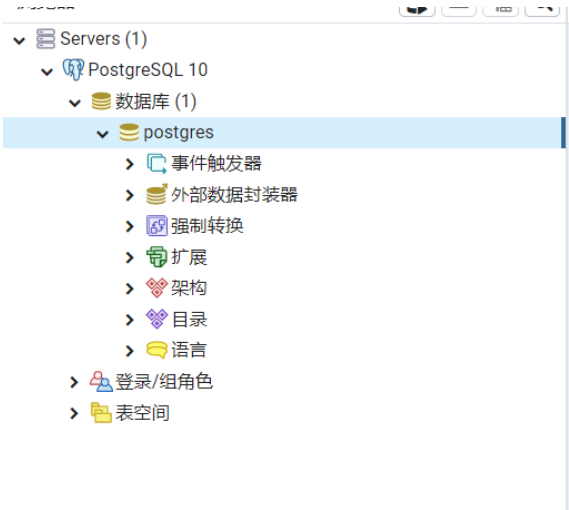
操作步骤：

1) 创建数据库 Antiepidemic：打开 pgAdmin4，在 postgres 中新建查询工具，输入如下的语句：

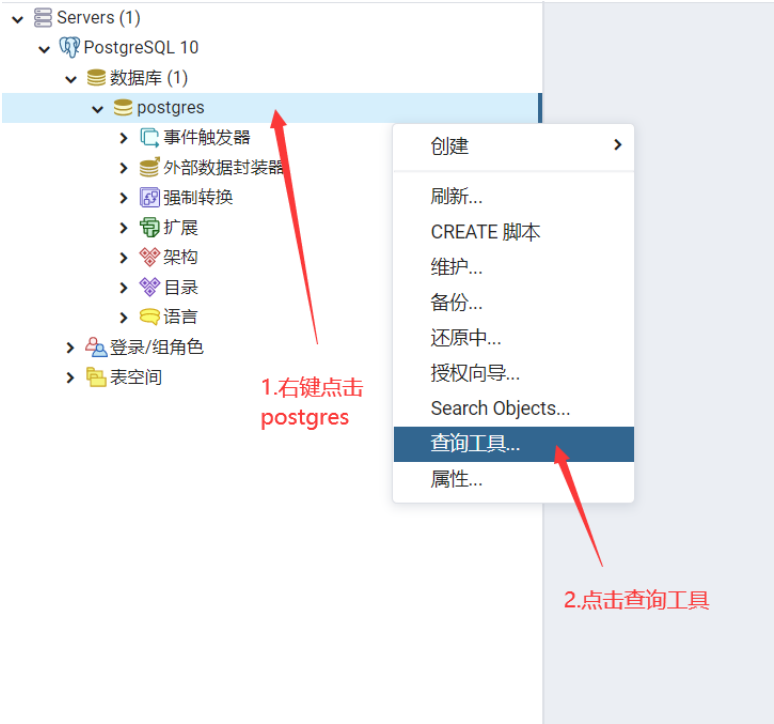
```
create database Antiepidemic
```

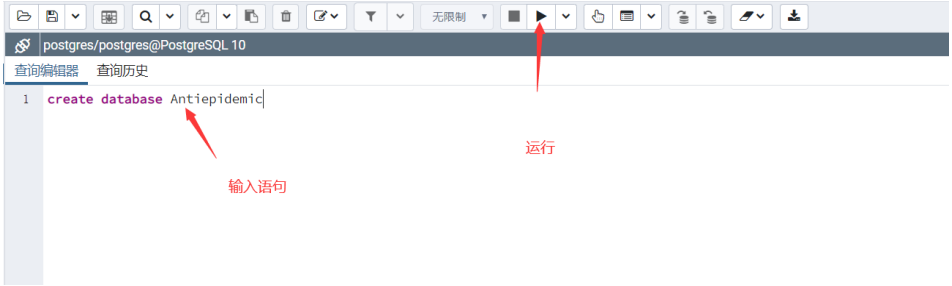
之后点击运行，刷新即可

原始状态：

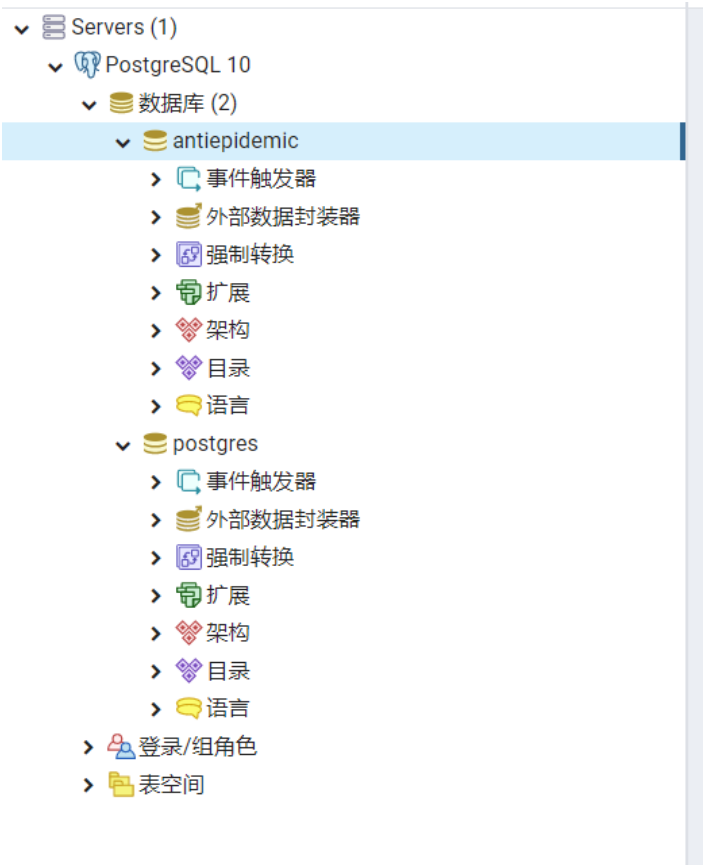


新建查询工具，输入语句：


Servers (1)
PostgreSQL 10
数据库 (1)
postgres
事件触发器
外部数据封装器
强制转换
扩展
架构
目录
语言
登录/组角色
表空间
1.右键点击
postgres
创建
刷新...
CREATE 脚本
维护...
备份...
还原中...
授权向导...
Search Objects...
查询工具...
属性...
2.点击查询工具


无限制
postgres/postgres@PostgreSQL 10
查询编辑器
查询历史
1 create database Antiepidemic
运行
输入语句

运行结果：

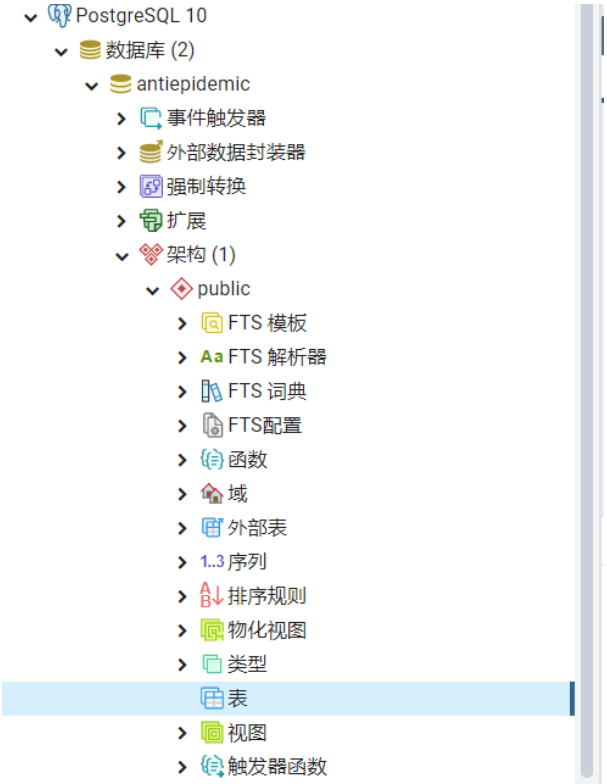


2) 右击 Antiepidemic，新建查询工具，输入 SQL 语句：

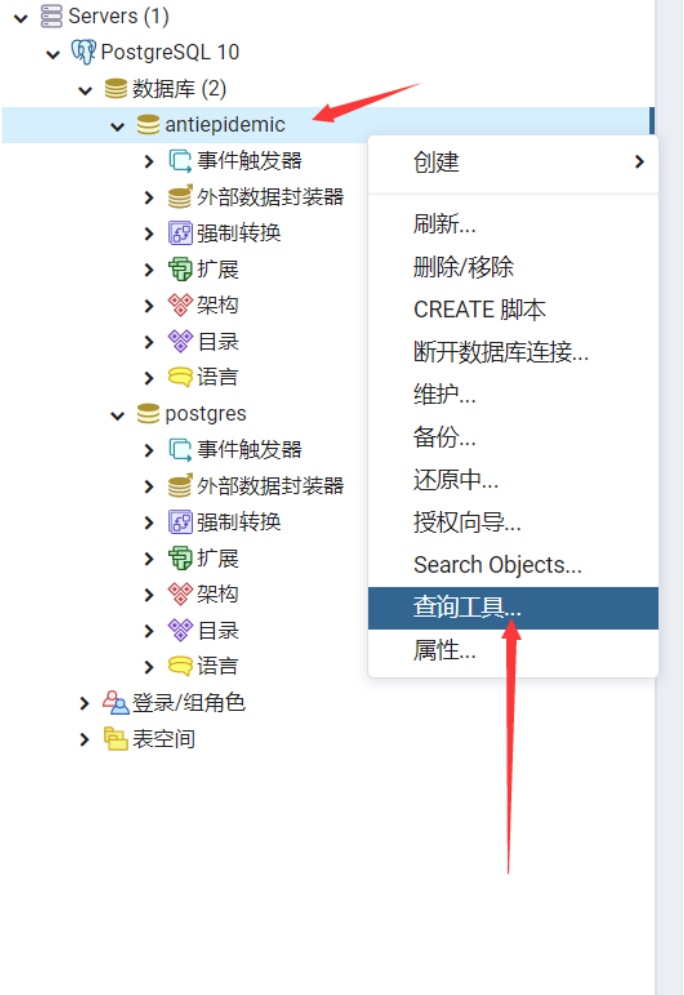
```
1) create table aehero(
2)     aehero_id char(20) not null unique primary key,
3)     aehero_name char(20),
4)     aehero_gender char(20)
5)
6) )
```

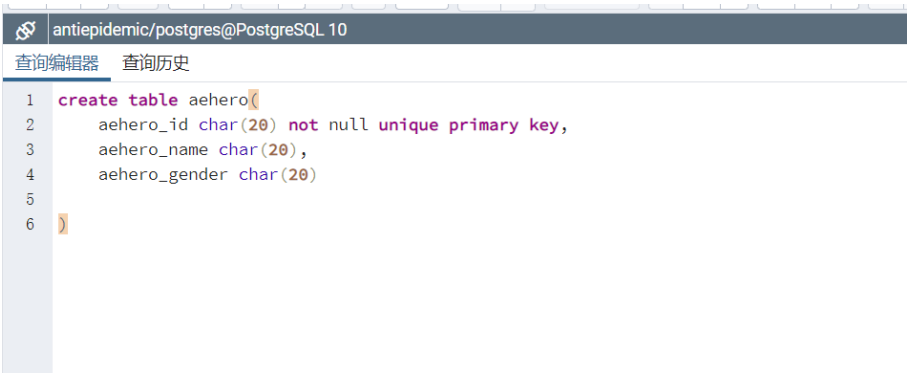
点击运行，再刷新即可。同时在表的属性中查看相关的信息

原始状态：


PostgreSQL 10
数据库 (2)
antiepidemic
事件触发器
外部数据封装器
强制转换
扩展
架构 (1)
public
FTS 模板
FTS 解析器
FTS 词典
FTS配置
函数
域
外部表
序列
排序规则
物化视图
类型
表
视图
触发器函数

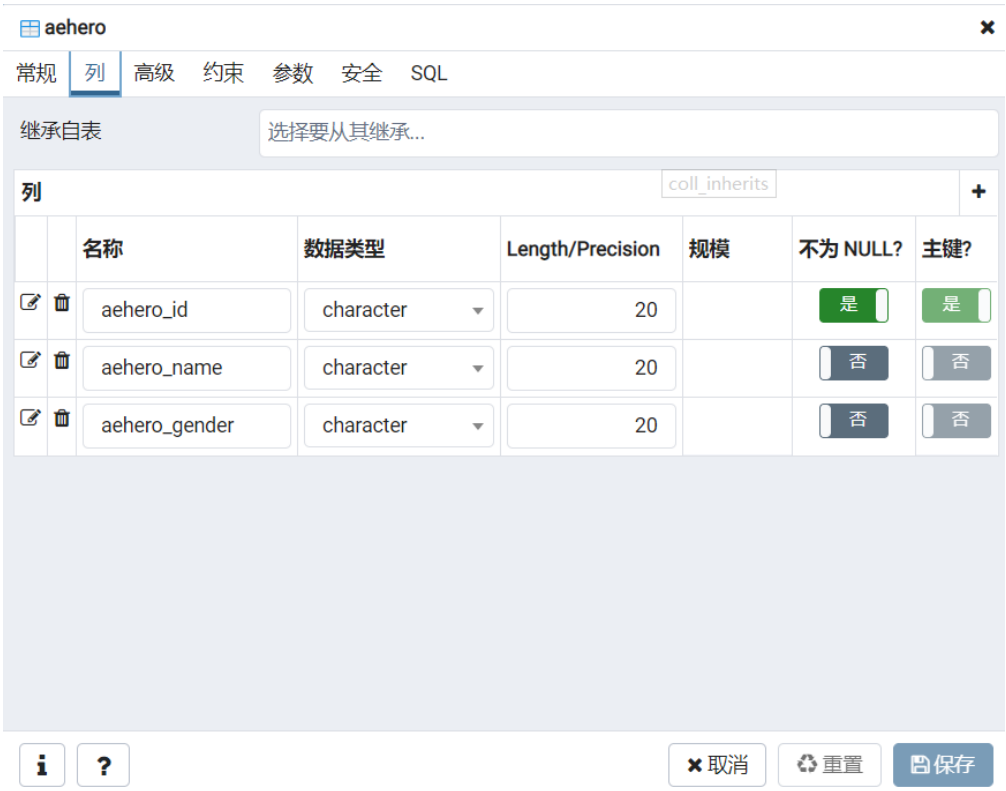
执行过程：


Servers (1)
PostgreSQL 10
数据库 (2)
antiepidemic
事件触发器
外部数据封装器
强制转换
扩展
架构
目录
语言
postgres
事件触发器
外部数据封装器
强制转换
扩展
架构
目录
语言
登录/组角色
表空间
创建
刷新...
删除/移除
CREATE 脚本
断开数据库连接...
维护...
备份...
还原中...
授权向导...
Search Objects...
查询工具...
属性...


antiepidemic/postgres@PostgreSQL 10
查询编辑器
查询历史
create table aehero(
aehero_id char(20) not null unique primary key,
aehero_name char(20),
aehero_gender char(20)
)

运行结果：

点击 aehero->属性->列，查看相关的属性信息



从而完成了对于表 aehero 的创建。

(2) 在数据库 Antiepidemic 中,使用 SQL 语句创建数据表 aemeasure(aemeasure_id, aemeasure_name, aemeasure_detail)，设置 aemeasure_id 为主键。

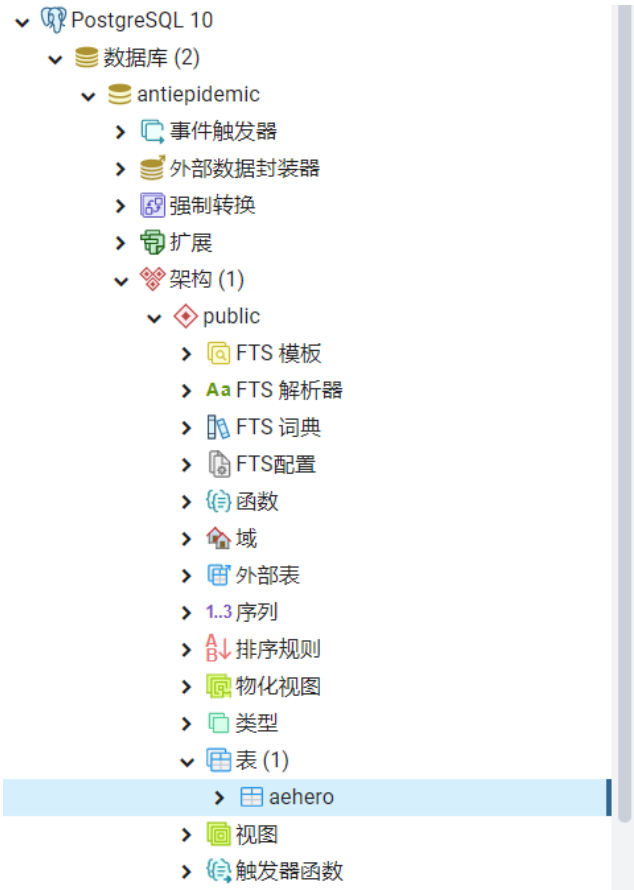
操作步骤：

和问题（1）类似，在数据库 Antiepidemic 中新建查询工具，输入如下的 SQL 语句：

```
create table aemeasure(
    aemeasure_id char(20) not null unique primary key,
    aemeasure_name char(20),
    aemeasure_detail char(20)
)
```

之后点击运行，刷新即可，可以在 aemeasure 表中查看属性

原始状态：



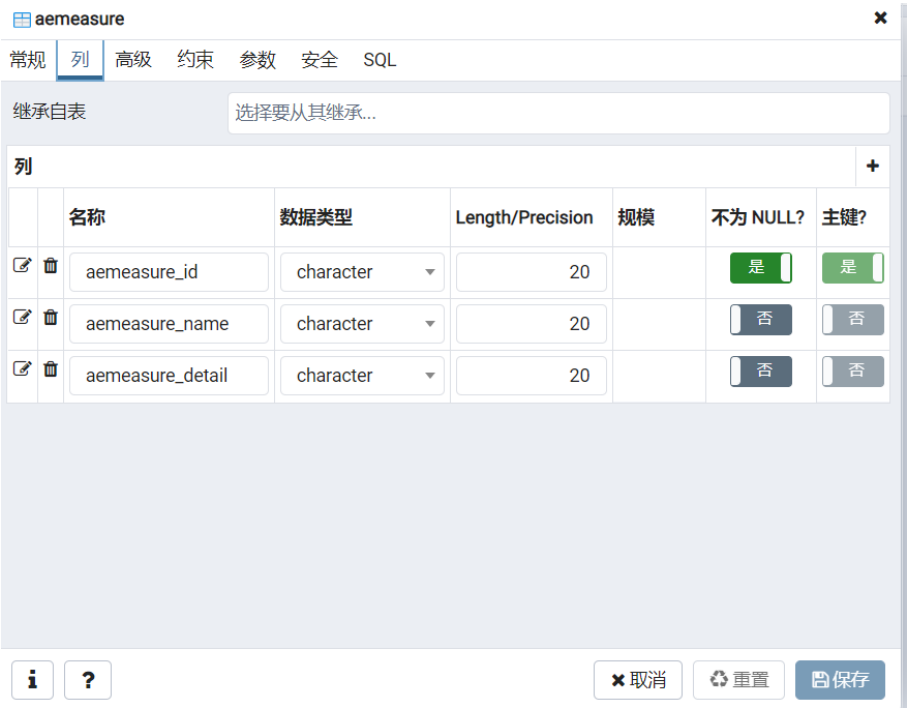
执行过程：

antiepidemic/postgres@PostgreSQL 10

查询编辑器　查询历史

```
create table aemeasure(
    aemeasure_id char(20) not null unique primary key,
    aemeasure_name char(20),
    aemeasure_detail char(20)
)
```

执行结果：



可见该表创建成功

(3) 在数据库 Antiepidemic 中,使用 SQL 语句创建数据表 contribution(aehero_id, aemeasure_id, aehero_deeds)，设置主键、外键。

操作步骤：

和问题（1）（2）类似，采用类似的方式创建表。需要注意的是本问题涉及了外键的建立，因此需要在建立表中增加 constraint 与 references 等涉及外键的语句。根据分析，主键适合设置为(aehero_id,aemeasure_id)，整体的 SQL 语句如下所示：

```
create  table contribution(
    aehero_id char(20),
    aemeasure_id char(20),
    aehero_deeds char(50),
    primary key(aehero_id,aemeasure_id),

    constraint fk_aehero foreign  key(aehero_id)
    references aehero(aehero_id),

    constraint fk_aemeasure foreign key(aemeasure_id)
    references aemeasure(aemeasure_id)
```

```
12. )
```

运行之后刷新，在表中查看结果。

原始状态：

执行过程

antiepidemic/postgres@PostgreSQL 10

查询编辑器　查询历史

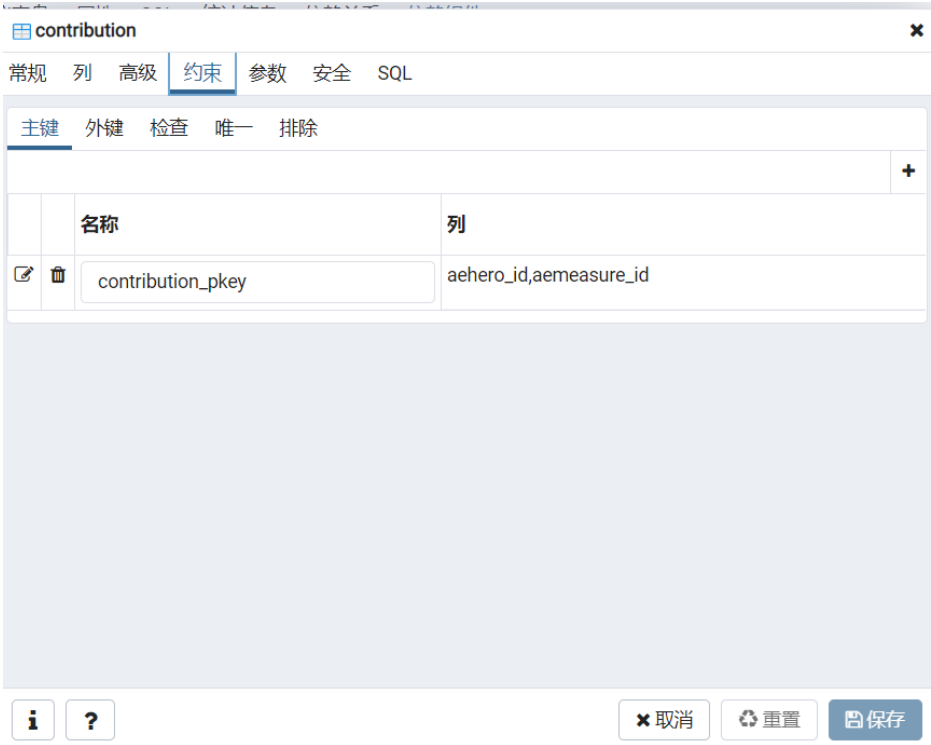
```
create  table contribution(
    aehero_id char(20),
    aemeasure_id char(20),
    aehero_deeds char(50),
    primary key(aehero_id,aemeasure_id),

    constraint fk_aehero foreign  key(aehero_id)
    references aehero(aehero_id),

    constraint fk_aemeasure foreign key(aemeasure_id)
    references aemeasure(aemeasure_id)
)
```

执行结果

类型	名称	限制
Foreign Key	public.contribution.fk_aehero	auto
Foreign Key	public.contribution.fk_aemeasure	auto
Primary Key	public.contribution_pkey	auto



(4) 使用 SQL 语句修改数据表 aehero 的名称为 hero1，并添加字段 aehero_team（抗疫英雄所属医疗队）。

操作步骤：

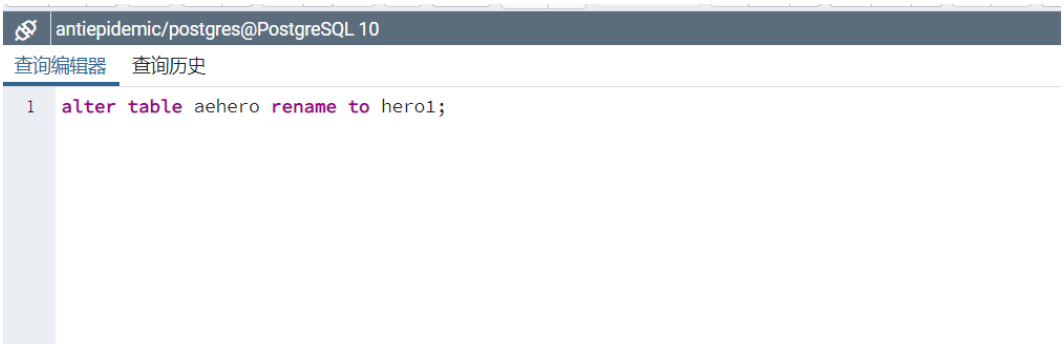
1) 在数据库 Antiepidemic 中新建查询工具，为了修改 aehero 的名称，可以考虑输入如下的 SQL 语句

```
1) alter table aehero rename to hero1;
```

运行之后刷新即可显示新的名称。

原始状态：

执行过程



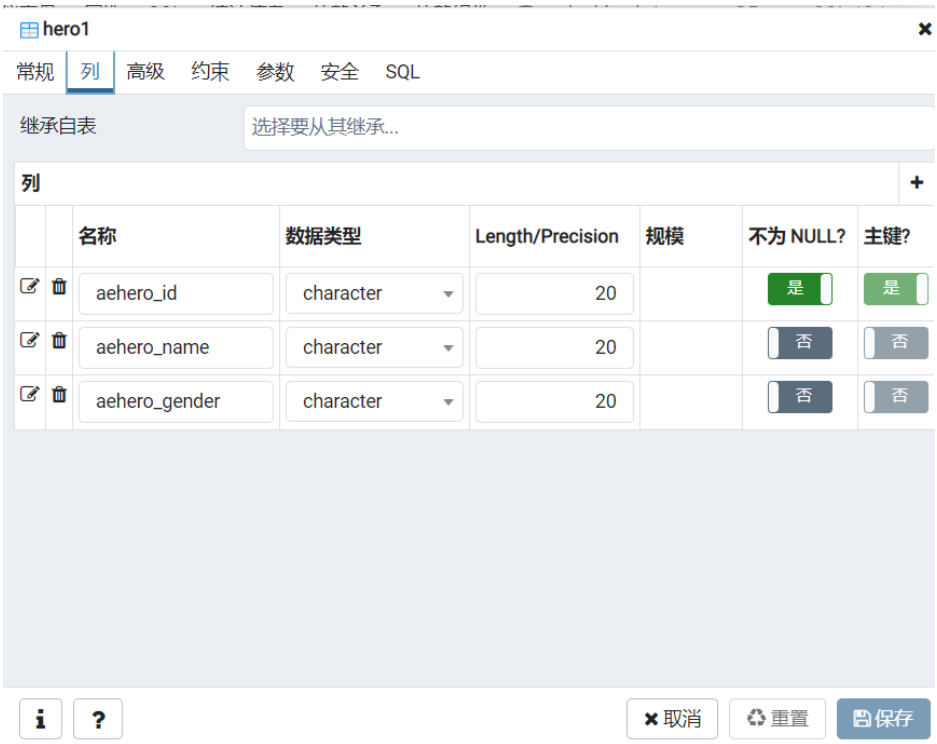
执行结果

2) 为了添加字段 aehero_team，可以考虑在查询工具中输入如下的 SQL 语句：

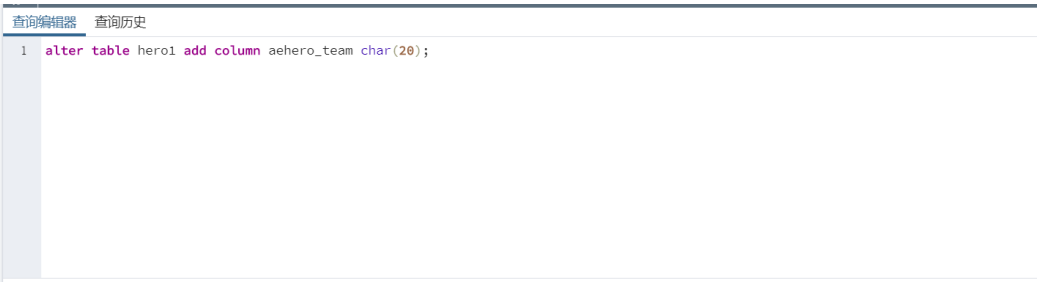
```
1) alter table hero1 add column aehero_team char(20);
```

运行之后，可以在 hero1 中查看属性

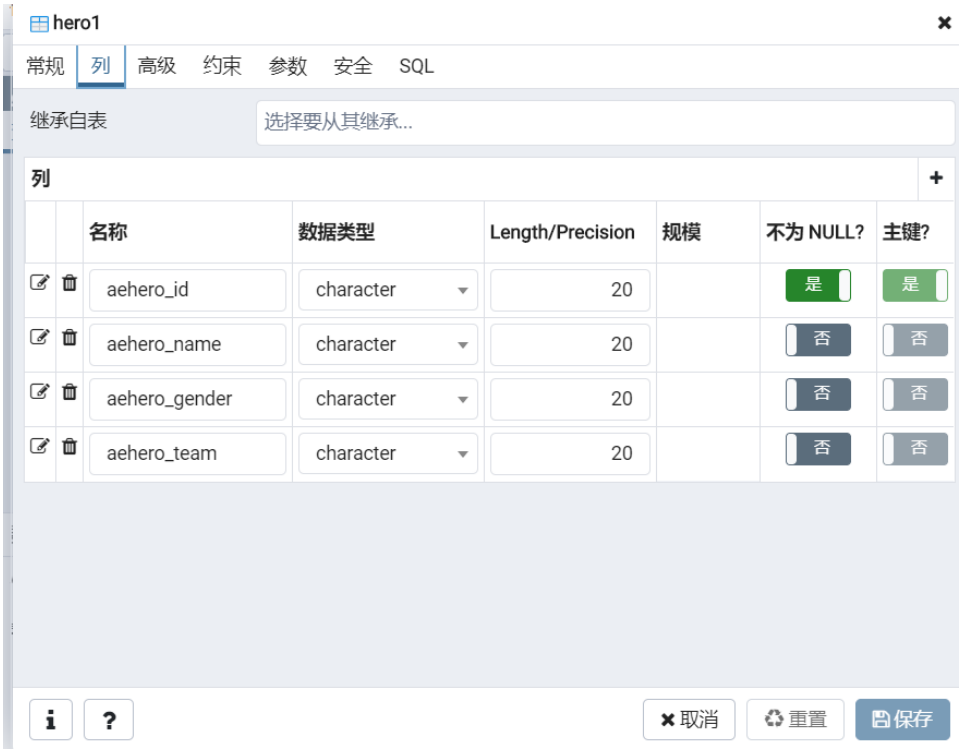
原始状态：


hero1
常规
列
高级
约束
参数
安全
SQL
继承自表
选择要从其继承...
列
名称
数据类型
Length/Precision
规模
不为 NULL?
主键?
aehero_id
character
20
是
是
aehero_name
character
20
否
否
aehero_gender
character
20
否
否
取消
重置
保存

执行过程：


查询编辑器
查询历史
1 alter table hero1 add column aehero_team char(20);

执行结果：



可见 aehero_team 成功添加。

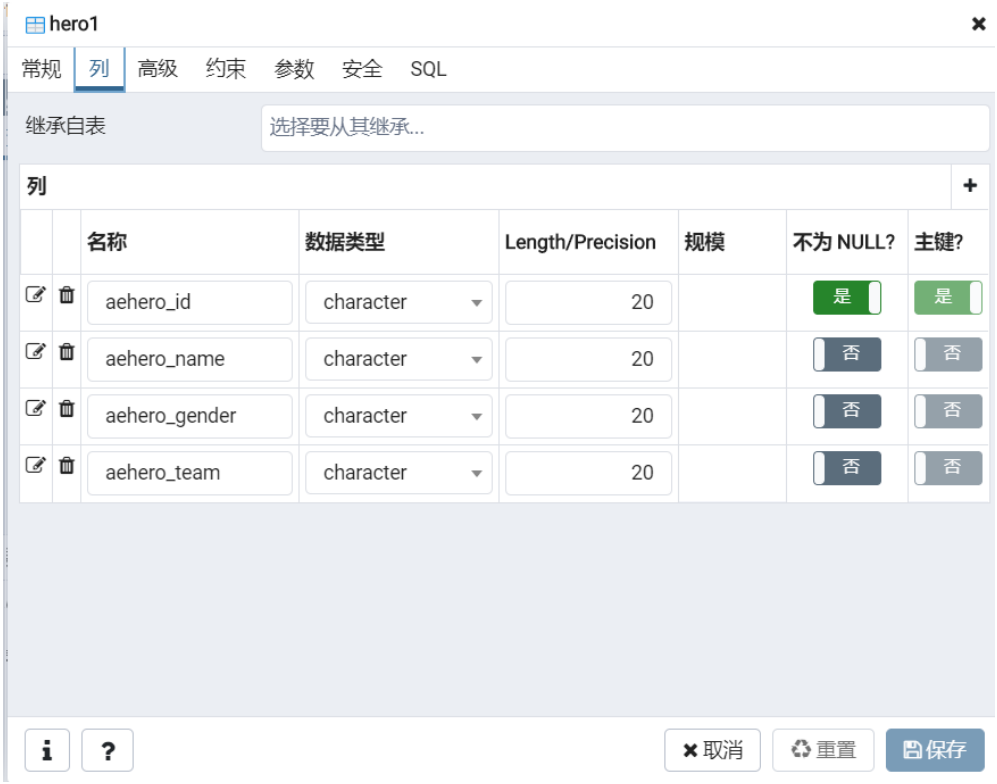
(5) 使用 SQL 语句修改数据表 hero1 中的 aehero_name 字段的类型为文本类型。

操作步骤：

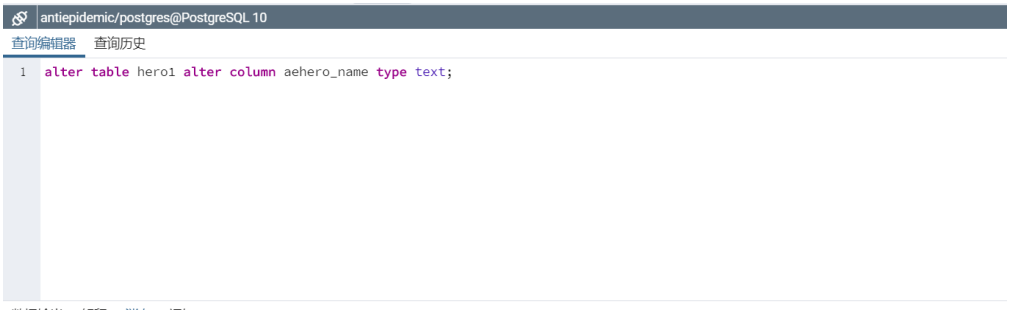
为了修改 hero1 中的 aehero_name 的类型，可以考虑在查询工具中输入如下的 SQL 语句：

```
1.  alter table hero1 alter column aehero_name type text;
```

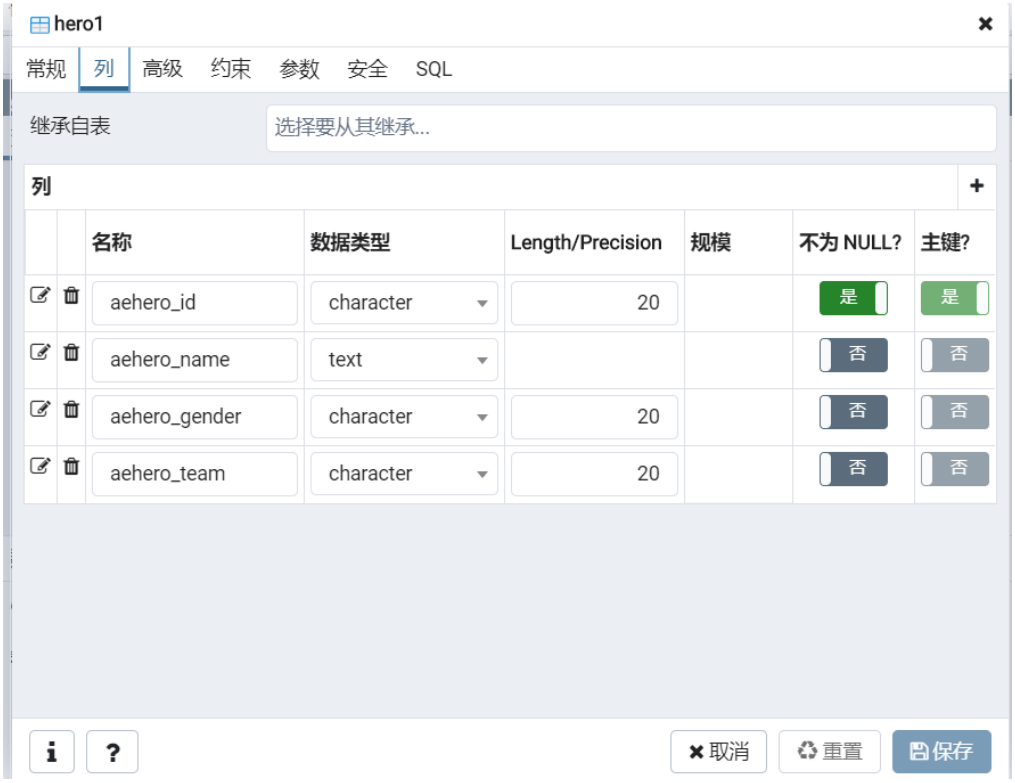
原始状态：


hero1
常规
列
高级
约束
参数
安全
SQL
继承自表
选择要从其继承...
列
名称
数据类型
Length/Precision
规模
不为 NULL?
主键?
aehero_id
character
20
是
是
aehero_name
character
20
否
否
aehero_gender
character
20
否
否
aehero_team
character
20
否
否
取消
重置
保存

执行过程：


antiepidemic/postgres@PostgreSQL 10
查询编辑器
查询历史
1 alter table hero1 alter column aehero_name type text;

执行结果：



可见 aehero_name 的数据类型确实改变了。

(6)使用 SQL 语句把数据表 aemeasure 中的 aemeasure_name 改为 aemeasurename。

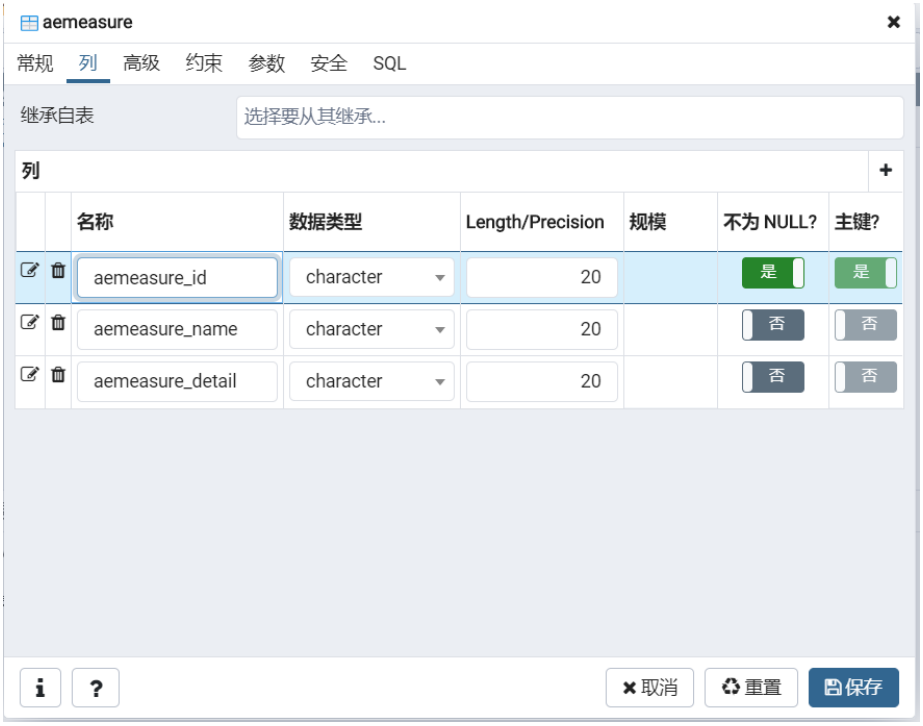
操作步骤：

为了修改 aemeasure 中的 aemeasure_name 的名称，可以考虑在查询工具中输入如下的 SQL 语句：

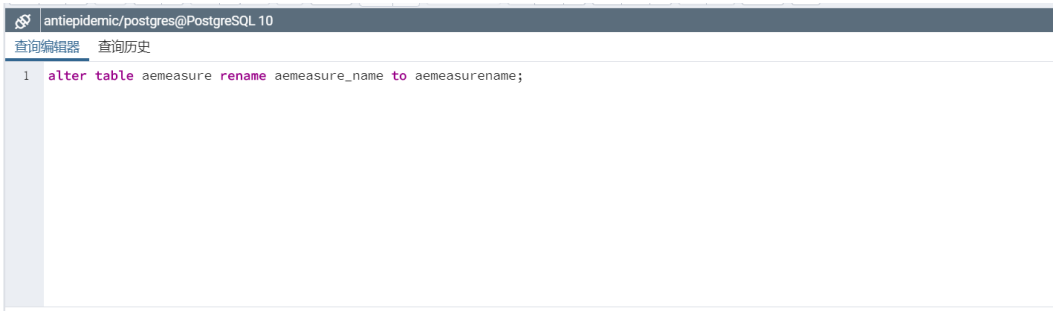
```
1.  alter table aemeasure rename aemeasure_name to aemeasurename;
```

运行之后可以在 aemeasure 中查看相关的属性

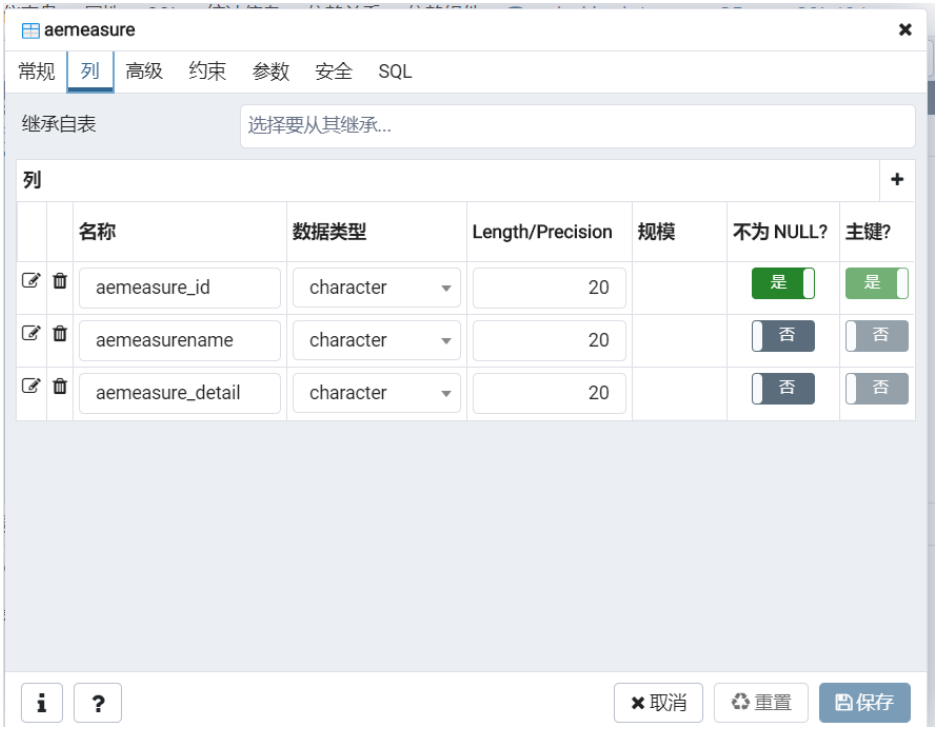
原始状态：


aemeasure
常规
列
高级
约束
参数
安全
SQL
继承自表
选择要从其继承...
列
名称
数据类型
Length/Precision
规模
不为 NULL?
主键?
aemeasure_id
character
20
是
是
aemeasure_name
character
20
否
否
aemeasure_detail
character
20
否
否
取消
重置
保存

执行过程：


antiepidemic/postgres@PostgreSQL 10
查询编辑器
查询历史
1 alter table aemeasure rename aemeasure_name to aemeasurename;

执行结果：



可见成功实现了属性更名的操作。

(7) 使用 SQL 语句删除数据表 hero1 中的字段 aehero_gender。

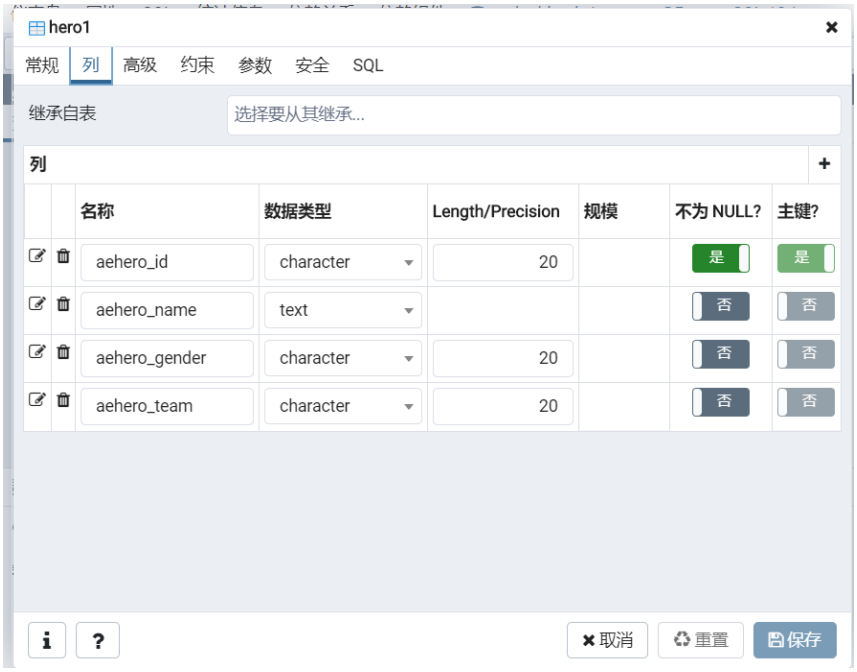
操作步骤：

为了实现删除字段的功能，可以在查询工具中输入如下的 SQL 语句：

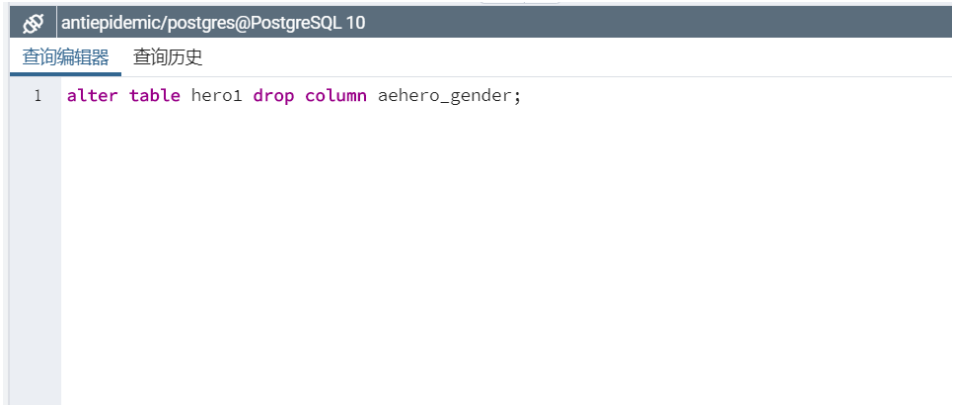
```
1. alter table hero1 drop column aehero_gender;
```

运行之后，可以在 hero1 表内查看相关的字段属性

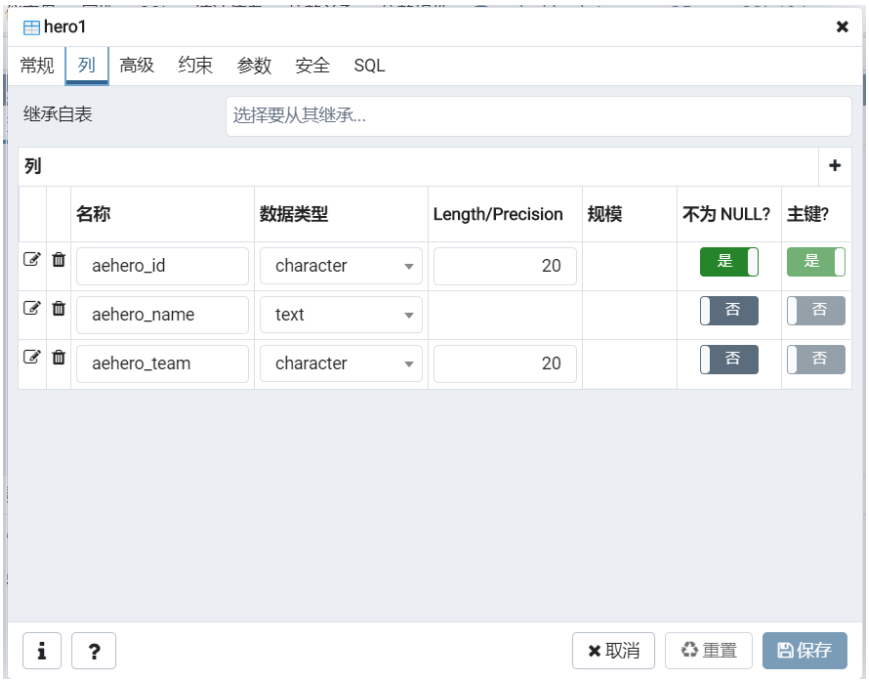
原始状态：


hero1
常规
列
高级
约束
参数
安全
SQL
继承自表
选择要从其继承...
列
名称
数据类型
Length/Precision
规模
不为 NULL?
主键?
aehero_id
character
20
是
是
aehero_name
text
否
否
aehero_gender
character
20
否
否
aehero_team
character
20
否
否
取消
重置
保存

执行过程：


antiepidemic/postgres@PostgreSQL 10
查询编辑器
查询历史
1 alter table hero1 drop column aehero_gender;

执行结果：



可见字段已经删除

(8) 使用 SQL 语句删除数据表 hero1，这时会发生什么情况，截图说明。

操作步骤：

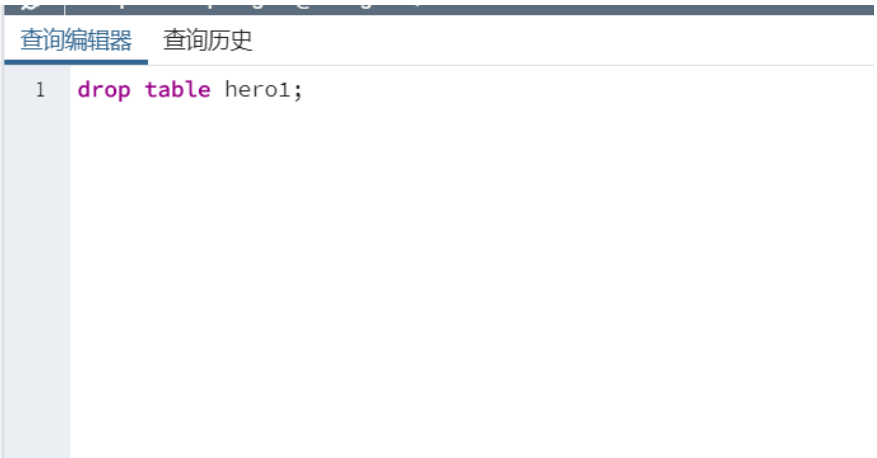
为了删除数据表 hero1，在查询工具中输入如下的语句：

```
1. drop table hero1;
```

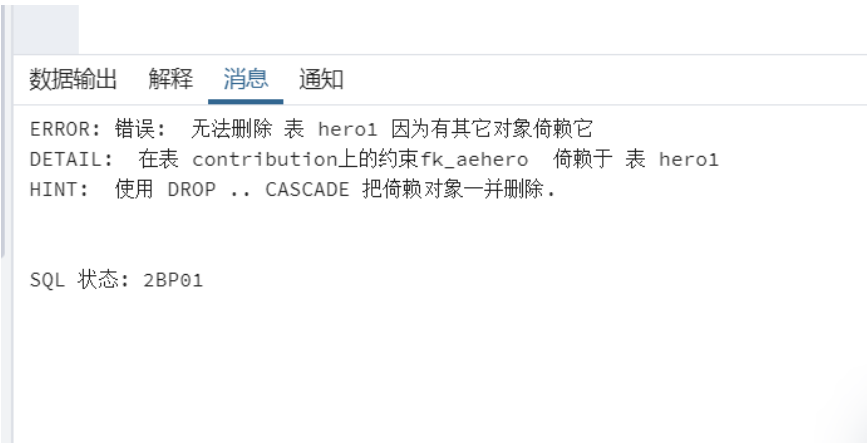
原始状态：

数据库 (2)
antiepidemic
事件触发器
外部数据封装器
强制转换
扩展
架构 (1)
public
FTS 模板
FTS 解析器
FTS 词典
FTS配置
函数
域
外部表
序列
排序规则
物化视图
类型
表 (3)
aemeasure
contribution
hero1

执行过程：


查询编辑器
查询历史
1 drop table hero1;

执行结果：


数据输出
解释
消息
通知
ERROR: 错误: 无法删除 表 hero1 因为有其它对象倚赖它
DETAIL: 在表 contribution上的约束fk_aehero 倚赖于 表 hero1
HINT: 使用 DROP .. CASCADE 把倚赖对象一并删除.
SQL 状态: 2BP01

原因：hero1 和 contribution 之间存在着外键约束 fk_aehero，这个约束会阻止关系的删除。

(9) 使用 SQL 语句删除数据表 hero1，aemeasure。

操作步骤：

1) 在删除之前，需要先解除外键的约束 fk_aehero,fk_aemeasure

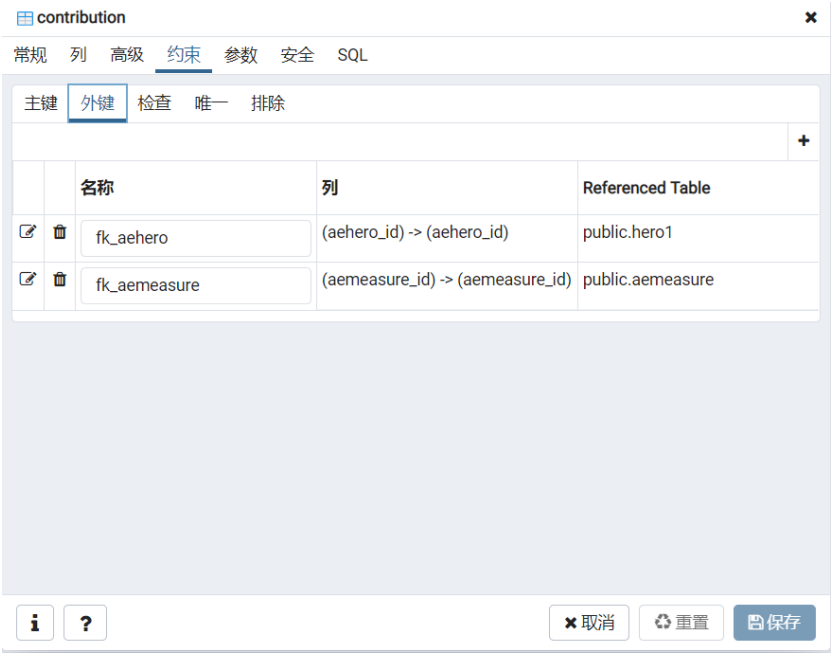
考虑在查询窗口输入如下的 SQL 语句：

```
alter table contribution drop constraint fk_aehero;
alter table contribution drop constraint fk_aemeasure;
```

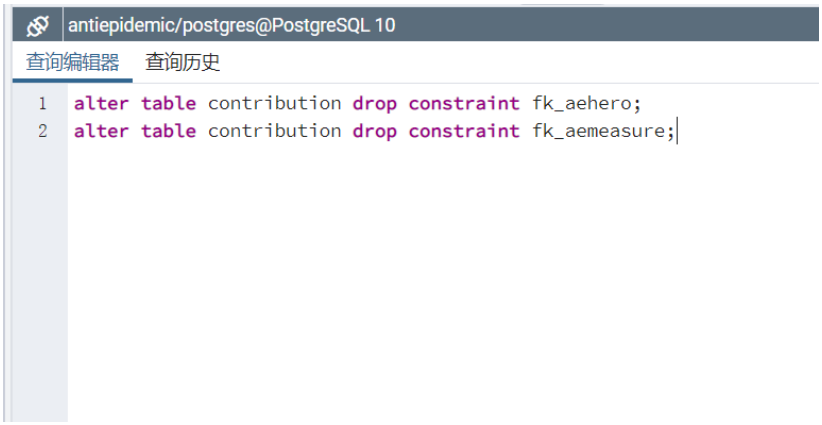
从而将外键的约束进行删除

原始状态:

右键点击 contribution->属性->约束->外键，查看 contribution 相关的约束

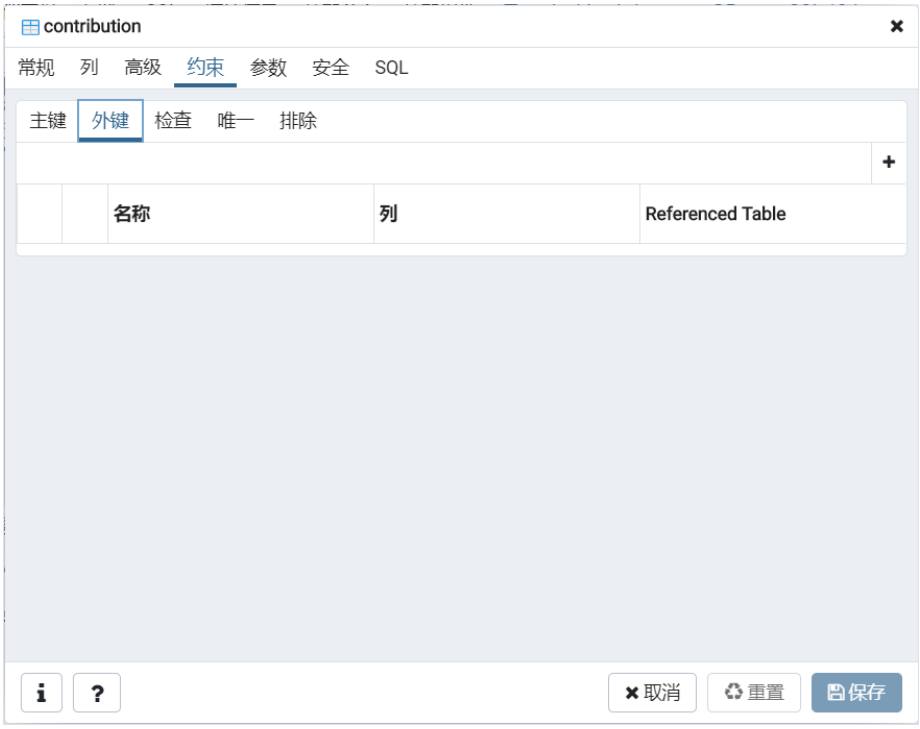


执行过程：



执行结果：

查看外键约束相关的信息，结果如下：



可见约束关系已经删除

2) 在查询工具中输入如下的 SQL 语句

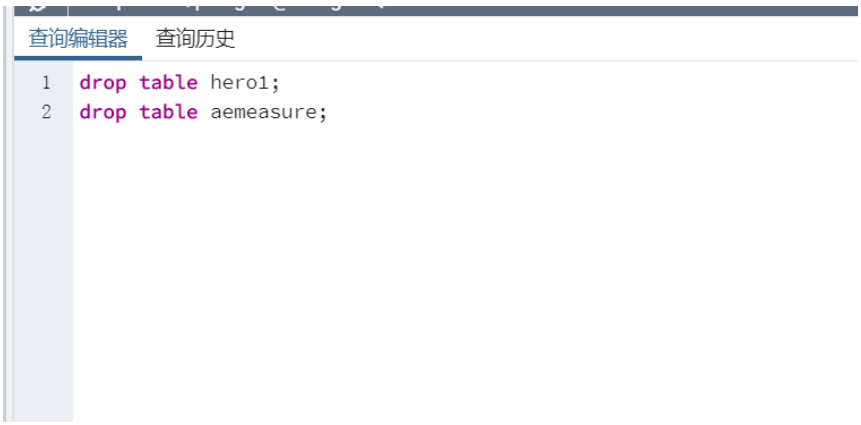
```
1) drop table hero1;
2) drop table aemeasure;
```

运行之后刷新即可。

原始状态：

架构 (1)
public
FTS 模板
FTS 解析器
FTS 词典
FTS配置
函数
域
外部表
序列
排序规则
物化视图
类型
表 (3)
aemeasure
contribution
hero1

执行过程：


查询编辑器
查询历史
1 drop table hero1;
2 drop table aemeasure;

执行结果：

可见两个表均已经删除。

(10) 创建如下 aehero 表，使用 SQL 语句向 aehero 表中插入数据:

aehero_id	aehero_name	aehero_gender	aehero_team
1	李兰娟	女	中国工程院
2	钟南山	男	中国工程院
3	张文宏	男	上海医疗队
4	张继先	男	武汉医疗队
5	张定宇	男	武汉医疗队

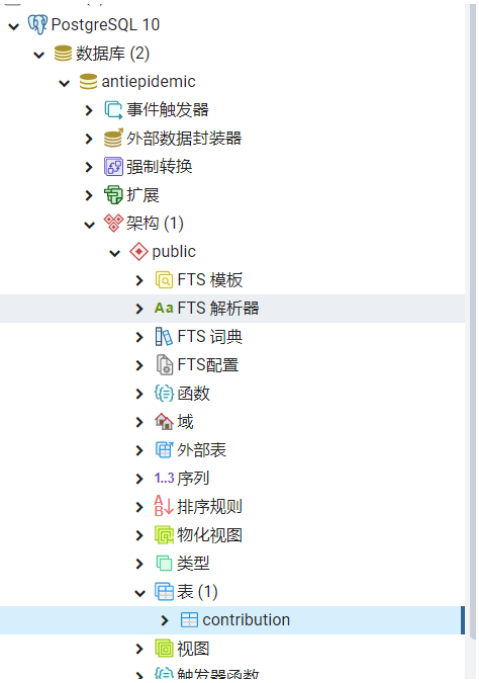
操作步骤：

1) 创建 aehero 表，在查询工具中输入如下的 SQL 语句：

```
create table aehero(
    aehero_id smallint not null unique primary key,
    aehero_name char(20),
    aehero_gender char(20),
    aehero_team char(20)
)
```

之后的过程与问题（1）完全相同，这里不再赘述

原始状态：



执行过程：

```
create table aehero(
    aehero_id smallint not null unique primary key,
    aehero_name char(20),
    aehero_gender char(20),
    aehero_team char(20)
)
```

执行结果：

2) 为了向数据库输入数据，在查询工具中输入如下的 SQL 语句：

```
insert into aehero values
(1,'李兰娟','女','中国工程院'),
(2,'钟南山','男','中国工程院'),
(3,'张文宏','男','上海医疗队'),
(4,'张继先','男','武汉医疗队'),
(5,'张定宇','男','武汉医疗队');
```

运行之后输入

```
select * from aehero
```

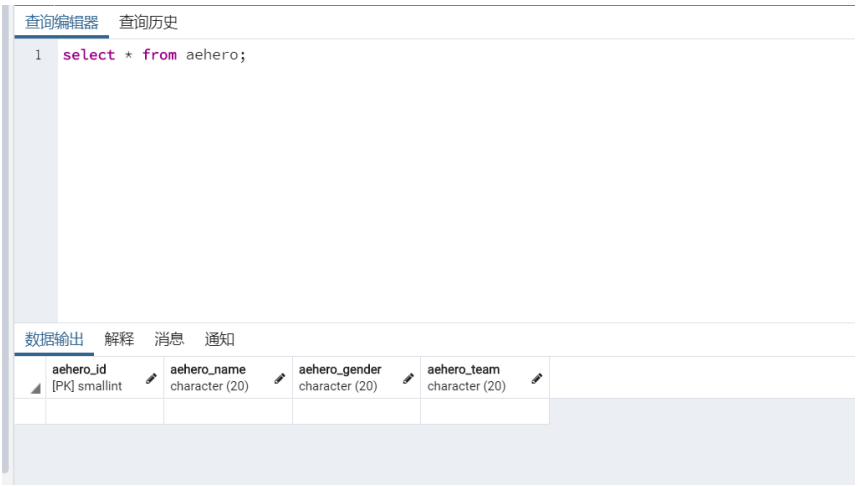
查看数据即可

原始状态：

输入

```
select * from aehero
```

进行查看数据



可见此时表内没有数据

执行过程：

查询编辑器　查询历史

```
insert into aehero values
(1,'李兰娟','女','中国工程院'),
(2,'钟南山','男','中国工程院'),
(3,'张文宏','男','上海医疗队'),
(4,'张继先','男','武汉医疗队'),
(5,'张定宇','男','武汉医疗队');
```

数据输出　解释　消息　通知

```
INSERT 0 5

耗时65 msec 成功返回查询。
```

插入成功！

执行结果：

查询编辑器　查询历史

```
select * from aehero;
```

数据输出　解释　消息　通知

	aehero_id [PK] smallint	aehero_name character (20)	aehero_gender character (20)	aehero_team character (20)
1	1	李兰娟	女	中国工程院
2	2	钟南山	男	中国工程院
3	3	张文宏	男	上海医疗队
4	4	张继先	男	武汉医疗队
5	5	张定宇	男	武汉医疗队

可见数据成功插入。

(11) 创建如下 aemeasure 表，使用 SQL 语句向 aemeasure 表中插入数据:

aemeasure_id	aemeasure_name	aemeasure_detail
1	封锁城市	武汉市关闭离汉通道，暂停市内交通，取消社会聚集活动，严格佩戴口罩。

2	分离病毒	当人类拥有了分离后的毒株，才能对人类进行临床实验，尽可能筛选出合适的治疗药物和治疗手段。
3	研制中医药剂	面对疫情，抗疫英雄们不分日夜，潜心研究，治愈患者成果显著。

操作步骤：

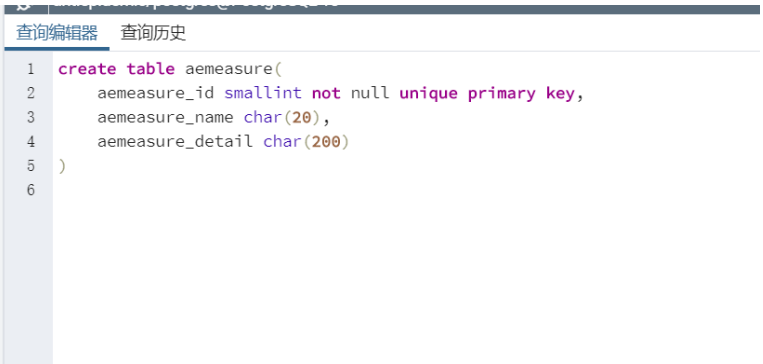
1) 创建 aemeasure 表，方法同问题（2），在查询工具中输入如下的 SQL 语句：

```sql
create table aemeasure(
    aemeasure_id smallint not null unique primary key,
    aemeasure_name char(20),
    aemeasure_detail char(200)
)
```

运行后刷新即可

原始状态：

执行过程：



执行结果：

2) 向查询工具中输入如下的 SQL 语句用于插入数据：

```
insert into aemeasure values
(1,'封锁城市','武汉市关闭离汉通道，暂停市内交通，取消社会聚集活动，严格佩戴口罩。'),
(2,'分离病毒','当人类拥有了分离后的毒株，才能对人类进行临床实验，尽可能筛选出合适的治疗药物和治疗手段。'),
(3,'研制中医药剂','面对疫情，抗疫英雄们不分日夜，潜心研究，治愈患者成果显著。');
```

在运行之后输入

```
select * from aemeasure
```

查看表内的数据。

原始数据：

查询编辑器　查询历史

```
select * from aemeasure;
```

数据输出　解释　消息　通知

aemeasure_id [PK] smallint	aemeasure_name character (20)	aemeasure_detail character (200)

可见此时 aemeasure 中并没有数据

执行过程：

查询编辑器　查询历史

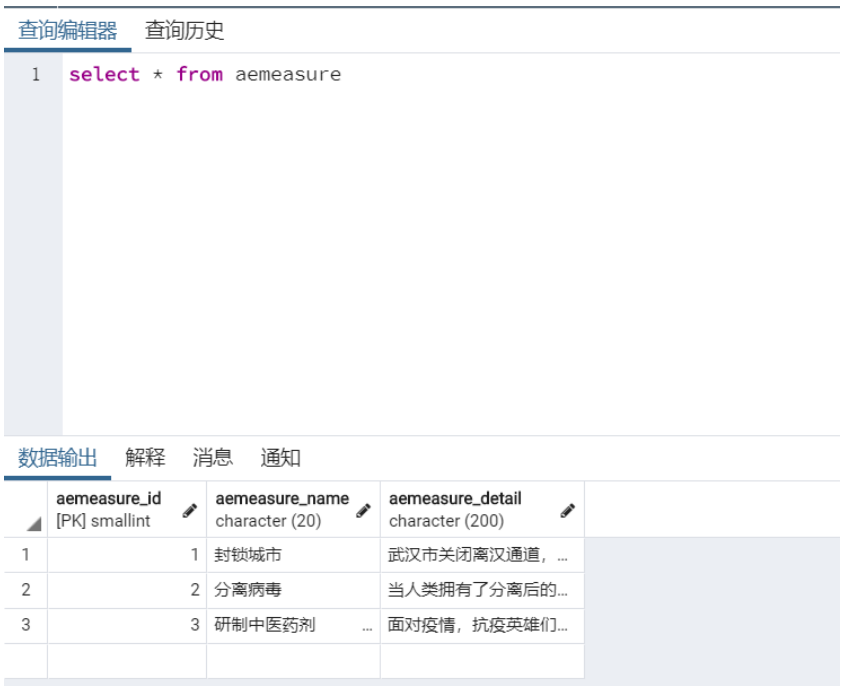
```
insert into aemeasure values
(1,'封锁城市','武汉市关闭离汉通道，暂停市内交通，取消社会聚集活动，严格佩戴口罩。'),
(2,'分离病毒','当人类拥有了分离后的毒株，才能对人类进行临床实验，尽可能筛选出合适的治疗药物和治疗手段。'),
(3,'研制中医药剂','面对疫情，抗疫英雄们不分日夜，潜心研究，治愈患者成果显著。');
```

数据输出　解释　消息　通知

```
INSERT 0 3

耗时101 msec 成功返回查询。
```

执行结果：



可见成功插入了数据。

(12) 使用 SQL 语句向 contribution 表中插入如下指定字段的数据：

aehero_id	aemeasure_id	aehero_deeds
1	3	2 月 4 日，武汉传来好消息。李兰娟院士团队公布重大研究成果，阿比朵尔、达芦那韦两种药物能有效抑制新型冠状病毒。

操作步骤：

在查询工具中输入如下的 SQL 语句

```
1.  insert into contribution values
2.  (1,3,'2 月 4 日，武汉传来好消息。李兰娟院士团队公布重大研究成果，阿比朵尔、达芦那韦两种药物能有效抑制新型冠状病毒。');
```

运行之后输入如下语句

```
2.  select * from contribution
```

即可显示结果

原始状态：

查询编辑器　查询历史

```
select * from contribution
```

数据输出　解释　消息　通知

aehero_id [PK] character (20)	aemeasure_id [PK] character (20)	aehero_deeds character (100)

执行过程：

查询编辑器　查询历史

```
insert into contribution values
(1,3,'2月4日，武汉传来好消息。李兰娟院士团队公布重大研究成果，阿比朵尔、达芦那韦两种药物能有效抑制新型冠状病毒。');
```

数据输出　解释　消息　通知

```
INSERT 0 1

耗时187 msec 成功返回查询。
```

执行结果：

可见成功插入了数据

(13) 使用 SQL 语句将 aemeasure 表中 aemeasure_id=1 的措施名称改成“封城”。

操作步骤：

为了更新数据，可以考虑进行选择操作和更新操作的相关语句，在查询工具中输入如下的 SQL 语句：

```
1.   update aemeasure set aemeasure_name='封城' where aemeasure_id=1;
```

运行前后都可以通过

```
1.   select * from aemeasure
```

进行查看关系内部的数据

原始状态：

查询编辑器　查询历史

```
1  select * from aemeasure
```

数据输出　解释　消息　通知

	aemeasure_id [PK] smallint	aemeasure_name character (20)	aemeasure_detail character (200)
1	1	封锁城市	武汉市关闭离汉通道，...
2	2	分离病毒	当人类拥有了分离后的...
3	3	研制中医药剂 ...	面对疫情，抗疫英雄们...

执行过程：

antiepidemic/postgres@PostgreSQL 10

查询编辑器　查询历史

```
1  update aemeasure set aemeasure_name='封城' where aemeasure_id=1;
```

数据输出　解释　消息　通知

```
UPDATE 1

耗时99 msec 成功返回查询。
```

执行结果：

名称修改成功！

(14) 使用 SQL 语句将 contribution 表中所有数据删除。

操作步骤：

考虑在查询工具中输入如下的 SQL 语句：

```
1.   delete from contribution;
```

运行前后通过查看表内的数据查看修改的情况。

原始状态：

查询编辑器 查询历史

```
1 select * from contribution;
```

数据输出 解释 消息 通知

	aehero_id [PK] character (20)	aemeasure_id [PK] character (20)	aehero_deeds character (100)
1	1	3	2月4日，武汉传来好...

执行过程：

查询编辑器 查询历史

```
1 delete from contribution;
```

数据输出 解释 消息 通知

```
DELETE 1

耗时135 msec 成功返回查询。
```

执行结果：

查询编辑器 查询历史

```
1 select * from contribution;
```

数据输出 解释 消息 通知

aehero_id [PK] character (20)	aemeasure_id [PK] character (20)	aehero_deeds character (100)

数据成功删除!