实验 3 DBMS 的安装和使用

姓名：段皞一
学号：3190105359
专业：计算机科学与技术

一 实验目的

1. 熟悉通过 SQL 进行数据完整性控制的方法。

二 实验平台

1. 操作系统： Windows
2. 数据库管理系统：SQL Server

三 实验内容和要求

3.1 定义若干表，其中包括 primary key, foreign key 和 check 的定义

打开 Microsoft SQL Server Management Studio，连接自己的服务器，建立一个基于教材的详细的大学模式数据库。

在完成数据库University的建立后，在数据库的表选项中添加相应的一系列表。也可以使用SQL语言直接在新建查询中进行表的新建。

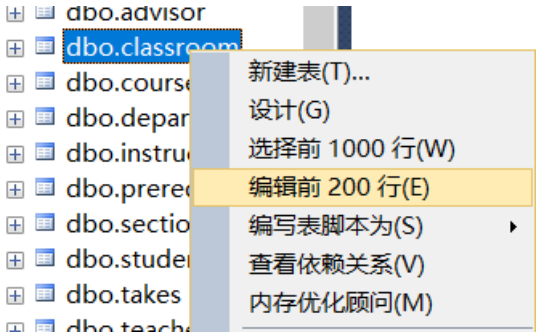
本次实验，采用SQL语句和图形界面结合的方式，进行表的创建。

3.1.1创建表classroom

创建表“classroom”.相关代码如下所示：

```
create table classroom
(
building varchar(15),
room_number varchar(7),
capacity numeric(4,0),
primary key (building, room_number)
);
```

表建成以后，右键dbo.classroom单击选择“编辑前200行”选项，可以使用图形界面进行字段的添加。



完成添加字段。效果图如下：

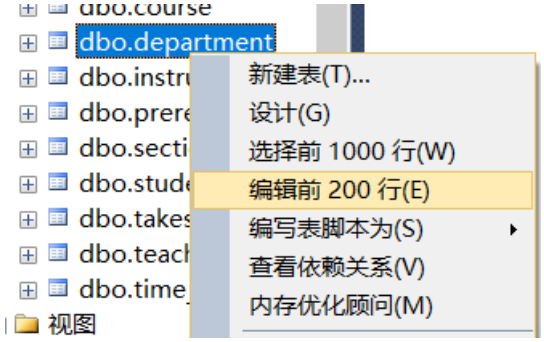
	building	room_nu...	capacity
▶	Packard	101	500
	Painter	514	10
	Taylor	3128	70
	Watson	100	30
	Watson	120	50
*	NULL	NULL	NULL

3.1.2创建表department

创建表”department”,相关SQL语句如下所示：

```
create table department
(
dept_name varchar(20),
building varchar(20),
budget numeric(12,2) check (budget > 0),
primary key (dept_name)
);
```

创建表成功后，可以在表的栏目下查找到”dbo.department”记录。右键以后，在弹出的窗口栏中选择“编辑前200行”选项，即可进行该表相应字段的添加。



该表字段添加完成以后的示意图如下所示：

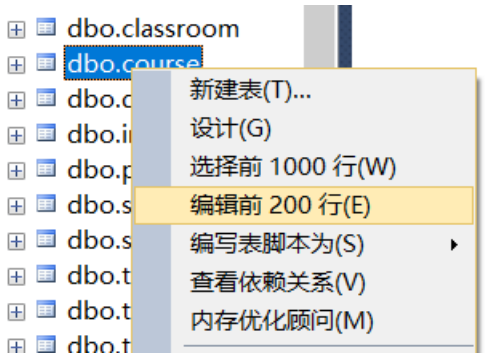
	dept_name	building	budget
	Biology	Watson	90000.00
	Comp.Sci.	Taylor	100000.00
	Elec.Eng.	Taylor	85000.00
	Finance	Painter	120000.00
	History	Painter	50000.00
	Music	Packard	80000.00
	Physics	Watson	70000.00
▸*	NULL	NULL	NULL

3.1.3 创建表 course

该表创建的相关代码如下所示：

```
create table course
(
course_id varchar(8),
title varchar(50),
dept_name varchar(20),
credits numeric(2, 0) check (credits > 0),
primary key (course_id),
foreign key (dept_name) references department
              on delete set null
);
```

创建成功以后，可以在表窗口栏下查到 dbo.course.右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



添加成功后的效果图如下：

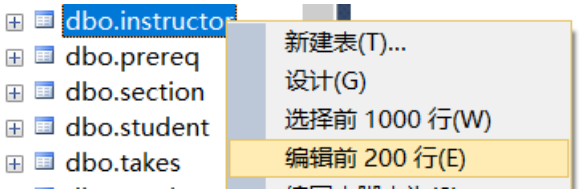
	course id	title	dept name	credits
▶	BIO-101	Intro. to Bi...	Biology	4
	BIO-301	Genetics	Biology	4
	BIO-399	Computati...	Biology	3
	CS-101	Intro. to C...	Comp.Sci.	4
	CS-190	Game Desi...	Comp.Sci.	4
	CS-315	Robotics	Comp.Sci.	3
	CS-319	Image Pro...	Comp.Sci.	3
	CS-347	Database ...	Comp.Sci.	3
	EE-181	Intro. to Di...	Elec.Eng.	3
	FIN-201	Investmen...	Finance	3
	HIS-351	World Hist...	History	3
	MU-199	Music Vid...	Music	3
	PHY-101	Physical Pr...	Physics	4
*	NULL	NULL	NULL	NULL

3.1.4 创建表 instructor

该表的创建代码如下所示：

```
create table instructor
(
ID varchar(5),
name varchar(20) not null,
dept_name varchar(20),
salary numeric (8,2) check (salary > 29000),
primary key(ID),
foreign key (dept_name) references department
on delete set null
);
```

创建成功以后，可以在表窗口栏下查到 dbo. instructor. 右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



字段添加完成以后的效果图如下：

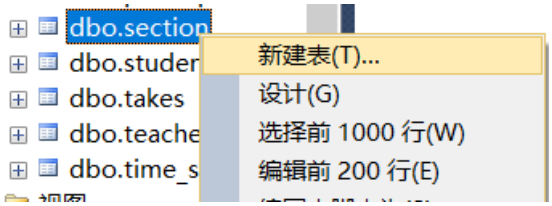
	ID	name	dept_name	salary
	10101	Srinivasan	Comp.Sci.	65000.00
	12121	Wu	Finance	90000.00
	15151	Mozart	Music	40000.00
	22222	Einstein	Physics	95000.00
	32343	El Said	History	60000.00
	33456	Gold	Physics	87000.00
	45565	Katz	Comp.Sci.	75000.00
	58583	Califieri	History	62000.00
	76543	Singh	Finance	80000.00
	76766	Crick	Biology	72000.00
	83821	Brandt	Comp.Sci.	92000.00
	98345	Kim	Elec.Eng.	80000.00
▸*	NULL	NULL	NULL	NULL

3.1.5 创建表 section

该表的创建的代码如下：

```
create table section
(
course_id varchar(8),
sec_id varchar(8),
semester varchar(6) check (semester in
('Fall', 'Winter', 'Spring', 'Summer')),
year numeric(4, 0) check (year > 1701 and year < 2100),
building varchar(15),
room_number varchar(7),
time_slot_id varchar(4),
primary key (course_id, sec_id, semester, year),
foreign key (course_id) references course
on delete cascade,
foreign key (building, room_number) references classroom
on delete set null
);
```

创建成功以后，可以在表窗口栏下查到 dbo.section.右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



添加字段以后的示意图如下：

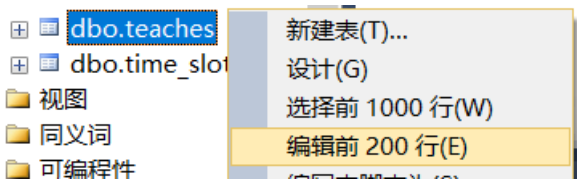
	course id	sec id	semester	year	building	room nu...	time slot...
	BIO-101	1	Summer	2009	Painter	514	B
	BIO-301	1	Summer	2010	Painter	514	A
	CS-101	1	Fall	2009	Packard	101	H
	CS-101	1	Spring	2010	Packard	101	F
	CS-190	1	Spring	2009	Taylor	3128	E
	CS-190	2	Spring	2009	Taylor	3128	A
	CS-315	1	Spring	2010	Watson	120	D
	CS-319	1	Spring	2010	Watson	100	B
	CS-319	2	Spring	2010	Taylor	3128	C
	CS-347	1	Fall	2009	Taylor	3128	A
	EE-181	1	Spring	2009	Taylor	3128	C
	FIN-201	1	Spring	2010	Packard	101	B
	HIS-351	1	Spring	2010	Painter	514	C
	MU-199	1	Spring	2010	Packard	101	D
	PHY-101	1	Fall	2009	Watson	100	A
▸*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3.1.6创建表teaches

创建表的相关代码如下：

```
create table teaches
(
ID varchar(5),
course_id varchar(8),
sec_id varchar(8),
semester varchar(6),
year numeric(4, 0),
primary key (ID, course_id, sec_id, semester, year),
foreign key (course_id, sec_id, semester, year) references section
on delete cascade,
foreign key (ID) references instructor
on delete cascade
);
```

创建成功以后，可以在表窗口栏下查到 dbo. teaches. 右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



添加字段完成以后的示意图如下：

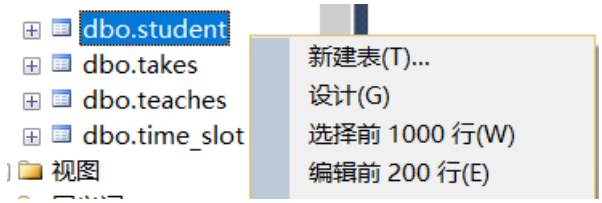
	ID	course id	sec id	semester	year
	10101	CS-101	1	Fall	2009
	10101	CS-315	1	Spring	2010
	10101	CS-347	1	Fall	2009
	12121	FIN-201	1	Spring	2010
	15151	MU-199	1	Spring	2010
	22222	PHY-101	1	Fall	2009
	32343	HIS-351	1	Spring	2010
	45565	CS-101	1	Spring	2010
	45565	CS-319	1	Spring	2010
	76766	BIO-101	1	Summer	2009
	76766	BIO-301	1	Summer	2010
	83821	CS-190	1	Spring	2009
	83821	CS-190	2	Spring	2009
	98345	EE-181	1	Spring	2009
▸*	NULL	NULL	NULL	NULL	NULL

3.1.7创建表student

创建相关表的相应代码如下所示：

```
create table student
(
ID varchar(5),
name varchar(20) not null,
dept_name varchar(20),
tot_cred numeric(3, 0) check(tot_cred >= 0),
primary key (ID),
foreign key (dept_name) references department
on delete set null
);
```

创建成功以后，可以在表窗口栏下查到 dbo. student. 右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



编辑字段成功后，可以看到相应的示意图如下：

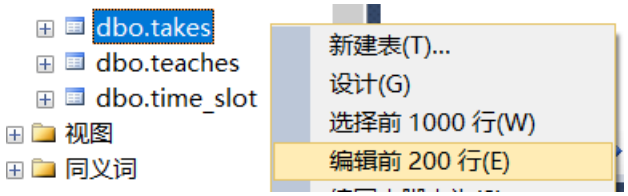
ID	name	dept name	tot cred
00128	Zhang	Comp.Sci.	102
12345	Shankar	Comp.Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp.Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp.Sci.	58
76653	Aoi	Elec.Eng.	60
98765	Bourikas	Elec.Eng.	98
98988	Tanaka	Biology	120
NULL	*NULL*	*NULL*	*NULL*

3.1.8创建表takes

该表创建的相关代码如下所示：

```
create table takes
(
ID varchar(5),
course_id varchar(8),
sec_id varchar(8),
semester varchar(6),
year numeric(4, 0),
grade varchar(2),
primary key (ID, course_id, sec_id, semester, year),
foreign key (course_id, sec_id, semester, year) references section
on delete cascade,
foreign key(ID) references student
on delete cascade
);
```

创建成功以后，可以在表窗口栏下查到 dbo.takes.右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



完成字段的添加以后，相关示意图如下：

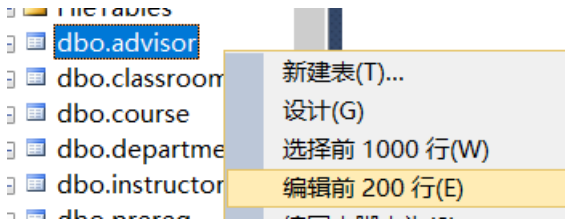
	ID	course id	sec id	semester	year	grade
	00128	CS-101	1	Fall	2009	A
	00128	CS-347	1	Fall	2009	A-
	12345	CS-101	1	Fall	2009	C
	12345	CS-190	2	Spring	2009	A
	12345	CS-315	1	Spring	2010	A
	12345	CS-347	1	Fall	2009	A
	19991	HIS-351	1	Spring	2010	B
	23121	FIN-201	1	Spring	2010	C+
	44553	PHY-101	1	Fall	2009	B-
	45678	CS-101	1	Fall	2009	A-
	45678	CS-319	1	Spring	2010	B
	54321	CS-101	1	Fall	2009	A-
	54321	CS-190	2	Spring	2009	B+
	55739	MU-199	1	Spring	2010	A-
	76543	CS-101	1	Fall	2009	A
	76543	CS-319	2	Spring	2010	A
	76653	EE-181	1	Spring	2009	C
	98765	CS-101	1	Fall	2009	C-
	98765	CS-315	1	Spring	2010	B
	98988	BIO-101	1	Summer	2009	A
	98988	BIO-301	1	Summer	2010	*NULL*
▶*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

3.1.9创建表advisor

该表创建的相关代码如下：

```
create table advisor
(
s_ID varchar(5),
i_ID varchar(5),
primary key (s_ID),
foreign key (i_ID) references instructor(ID)
on delete set null,
foreign key (s_ID) references student(ID)
on delete cascade
);
```

创建成功以后，可以在表窗口栏下查到 dbo.advisor.右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



完成字段的添加以后，相关示意图如下：

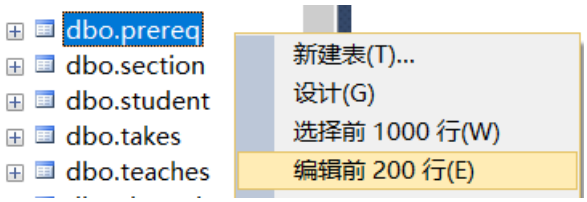
	s_ID	i_ID
	00128	45565
	12345	10101
	23121	76543
	44553	22222
	45678	22222
	76543	45565
	76653	98345
	98988	76766
▶*	NULL	NULL

3.1.10创建表prereq

创建表的相关代码如下：

```
create table prereq
(
course_id varchar(8),
prereq_id varchar(8),
primary key (course_id, prereq_id),
foreign key (course_id) references course
on delete cascade,
foreign key (prereq_id) references course
);
```

创建成功以后，可以在表窗口栏下查到 dbo.prereq.右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



完成字段的添加以后，可以看到示意图如下：

	course_id	prereq_id
	BIO-301	BIO-101
	BIO-399	BIO-101
	CS-190	CS-101
	CS-315	CS-101
	CS-319	CS-101
	CS-347	CS-101
	EE-181	PHY-101
▶*	NULL	NULL

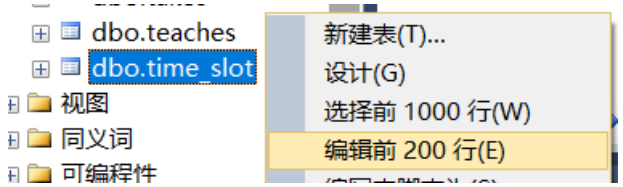
3.1.11创建表time_slot

该表创建的相关代码如下：

```
create table time_slot
```

```
(
time_slot_id varchar(4),
day varchar(1) check (day in ('M', 'T', 'W', 'R', 'F', 'S', 'U')),
start_time time,
end_time time,
primary key(time_slot_id, day, start_time)
);
```

创建成功以后，可以在表窗口栏下查到 dbo.time_slot.右键以后将弹出窗口栏，选择“编辑前 200 行”选项，即可进行相应字段的添加。



完成该表字段的添加以后，可以看到相关的示意图如下：

time_slot...	day	start_time	end_time
A	M	08:00:00	08:50:00
A	W	08:00:00	08:50:00
A	F	08:00:00	08:50:00
B	M	09:00:00	09:50:00
B	W	09:00:00	09:50:00
B	F	09:00:00	09:50:00
C	M	11:00:00	11:50:00
C	W	11:00:00	11:50:00
C	F	11:00:00	11:50:00
D	M	13:00:00	13:50:00
D	W	13:00:00	13:50:00
D	F	13:00:00	13:50:00
E	T	10:30:00	11:45:00
E	R	10:30:00	11:45:00
F	T	14:30:00	15:45:00
F	R	14:30:00	15:45:00
G	M	16:00:00	16:50:00
G	W	16:00:00	16:50:00
G	F	16:00:00	16:50:00
H	W	10:00:00	12:30:00
NULL	*NULL*	*NULL*	*NULL*

3.2 让表中插入数据，考察 primary key 如何控制实体完整性

新建查询，书写下面指令，加入学号为 10101 的 Mike 同学字段。相关代码如下:

```
insert student(ID, name, dept_name, tot_cred)
    values (10101, 'Mike', 'Biology', 4)
```

在下表中出现了新的字段 Mike，新建字段成功。

	10101	Mike	Biology	4

在书写以下代码，尝试加入学号同样为 10101 的 Shan 同学。相关代码如下：

```
insert student(ID, name, dept_name, tot_cred)
    values (10101, 'Shan', 'Physics', 5)
```

执行语句后，发现存在以下错误：

消息 2627，级别 14，状态 1，第 1 行
违反了 PRIMARY KEY 约束“PK__student__3214EC2712D27239”。不能在对象“dbo.student”中插入重复键。重复键值为 (10101)。
语句已终止。

3.3 删除被引用表中的行，考察 foreign key 中 on delete 子句如何控制参照完整性

尝试删除表 course 中的 course_id 为'BIO-101'的字段，输入以下代码：

```
delete from course
where course_id = 'BIO-101'
```

可以看到报出以下的错误，

消息 547，级别 16，状态 0，第 1 行
DELETE 语句与 REFERENCE 约束"FK__prereq__prereq_i__32E0915F"冲突。该冲突发生于数据库"University"，表"dbo.prereq"，column 'prereq_id'。
语句已终止。

示意图如下：

```
delete from course
where course_id = 'BIO-101'
```

```
%
消息
消息 547，级别 16，状态 0，第 1 行
DELETE 语句与 REFERENCE 约束"FK__prereq__prereq_i__32E0915F"冲突。该冲突发生于数据库"University"，表"dbo.prereq"，column 'prereq_id'。
语句已终止。
```

3.4 修改被引用表中的行的 primary key，考察 foreign key 中 on update 子句如何控制参照完整性

为了后续实验的方便，首先定义几个新的表。


```
CREATE TABLE customer
(customer_name char(20),
```

```
customer_street char(30),
customer_city char(30),
primary key (customer_name));

CREATE TABLE branch
(branch_name char(15),
branch_city char(30),
assets integer,
primary key (branch_name),
CHECK (assets>=0));

CREATE TABLE account
(account_number char(10),
branch_name char(15),
balance integer,
primary key (account_number),
foreign key (branch_name) references branch,
CHECK(balance>=0));

CREATE TABLE depositor
(customer_name char(20),
account_number char(10),
primary key (customer_name, account_number),
foreign key (account_number) references account,
foreign key (customer_name) references customer);
```

account 表中的外码没有加入 on update 子句进行约束，所以书写以下语句进行 account 中 branch_name 的修改是可以的。

但是当我们定义一下 account_new 表示，注意加入了 on update 子句：

```
CREATE TABLE account
(account_number char(10),
branch_name char(15),
balance integer,
primary key (account_number),
foreign key (branch_name) references branch
on update cascade,
CHECK(balance>=0));
```

此时当输入以下语句，视图进行 account_new 表中的 branch_name 的修改：

```
update account_new
set branch_name = 'huawei'
where branch_name = 'wahaha'
```

系统将会报出以下的错误：

消息 547，级别 16，状态 0，第 1 行
UPDATE 语句与 FOREIGN KEY 约束"FK__account_n__branc__1CF15040"冲突。该冲突发生于数据库"lab"，表"dbo.branch"，column 'branch_name'。

语句已终止。

```
update account_new
set branch_name = 'huawei'
where branch_name = 'wahaha'
```

```
消息 547，级别 16，状态 0，第 1 行
UPDATE 语句与 FOREIGN KEY 约束"FK__account_n__branc__1CF15040"冲突。该冲突发生于数据库"lab"，表"dbo.branch"，column 'branch_name'。
语句已终止。
```

3.5 修改或插入表中数据，考察 check 子句如何控制校验完整性

```
INSERT INTO account
VALUES('006','Pudong',-10)
```

消息 547，级别 16，状态 0，第 1 行

INSERT 语句与 CHECK 约束"CK__account__balance__164452B1"冲突。该冲突发生于数据库"lab"，表"dbo.account"，column 'balance'。

语句已终止。

```
INSERT INTO account
VALUES('006','Pudong',-10)
```

```
消息 547，级别 16，状态 0，第 1 行
INSERT 语句与 CHECK 约束"CK__account__balance__164452B1"冲突。该冲突发生于数据库"lab"，表"dbo.account"，column 'balance'。
语句已终止。
```

3.6 定义一个 assertion，并通过修改表中数据考察断言如何控制数据完整性

下面是一个断言定义的示例：

```
CREATE ASSERTION assertion_bal
CHECK (not exists(SELECT * FROM account
WHERE balance>5000));
```

但是遗憾的是，本实验使用的 SQL Server 貌似不支持 ASSERSION 功能，所以不能在服务器上进行实验的验证。

消息 343，级别 15，状态 1，第 1 行

CREATE、DROP 或 ALTER 语句中使用了未知的对象类型 'ASSERTION'。
消息 102，级别 15，状态 1，第 3 行
“)”附近有语法错误。

```
CREATE ASSERTION assertion_bal
CHECK (not exists(SELECT * FROM account
WHERE balance>5000));
```

```
%
结果
消息 343，级别 15，状态 1，第 1 行
CREATE、DROP 或 ALTER 语句中使用了未知的对象类型 'ASSERTION'。
消息 102，级别 15，状态 1，第 3 行
“)”附近有语法错误。
```

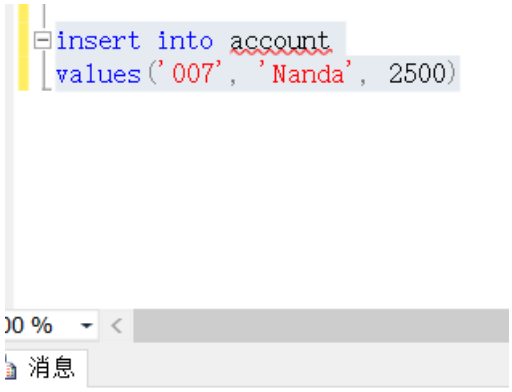
3.7 定义一个 trigger，并通过修改表中数据考察触发器如何起作用

定义一个 trigger，并通过修改数据考察触发器是如何起作用的。

```
CREATE TRIGGER trig
ON account
AFTER INSERT AS
IF(SELECT count(*)FROM account)=7
BEGIN
UPDATE account
SET balance=1.1*balance
WHERE account_number=(SELECT account_number FROM INSERTED)
END
```

定义好触发器以后，当插入新的一个字段，且字段总数已经有 7 个的时候，将触发触发器，对新插入的字段进行 balance 乘上 1.1 的操作，输入下面的代码，作为该表的第八个输入，

```
insert into account
values('007', 'Nanda', 2500)
```



可以发现插入的字段，其 balance 增加了 10%，从输入端的 2500 到输出端的 2750。

▶	007	Nanda	2750