第六章　第六次作业

1. 关于游标，选项中说法错误的是（）

   A. 游标可以定在查询结果集的特定行，也可以从结果集的当前行检索一行或多行

   B. 通常我们并不使用游标，但是需要逐条处理数据的时候，游标显得十分重要

   C. 游标使用时必须在完成后关闭，以释放资源

   D. 游标的 SELECT 语句中可以使用 INTO 子句来创建新表

2. 下列关于触发器的描述正确的是（）

   A. MySql 的触发器只支持行级出发，不支持语句级触发

   B. 触发器可以调用将数据返回客户端的存储程序

   C. 在触发器中可以使用显式或者隐式方式开始或结束事务的语句

   D. 在 MySql 中，不可使用 new 和 old 引用触发器中发生的记录内容

3. 关于 before 和 after 触发器的说法中，哪一项是正确的？（）

   A. 在 BEFORE 触发器中，可以对 INSERT 和 UPDATE 的 NEW 值进行修改；而在 AFTER 触发器中

   B. 在 AFTER 触发器中，可以修改 NEW 值，但在 BEFORE 触发器中不能修改

   C. BEFORE 触发器不能用于 UPDATE 操作

   D. AFTER 触发器可以对 INSERT 和 UPDATE 的 NEW 值进行修改

4. 关于 MySQL 触发器中 SELECT 语句的使用，以下哪种说法是正确的？（）

   A. 触发器中可以包含 SELECT 语句，并且可以返回结果集

   B. 触发器中可以包含 SELECT 语句，但不能返回结果集

   C. 触发器中必须包含 SELECT 语句，以便执行其他操作

   D. 触发器中可以使用 SELECT 语句来更新表的记录

5. 关于存储过程的说法中，哪一项是正确的？（）

   A. 存储过程可以通过 SELECT 语句直接返回结果集给调用者

   B. 存储过程只能接受一个输入参数，并且不能返回任何值

   C. 存储过程可以使用 BEGIN ... END 块来定义多个 SQL 语句的执行逻辑

   D. 存储过程的定义必须包含 CREATE FUNCTION 关键字

6. 创建一个名为 check_student_age_trigger 的触发器，该触发器用于在 students 表中插入新记录之前进行年龄检查。要求如下：

   触发器在执行 INSERT 操作之前触发。

   如果新插入的学生年龄（age 列）小于 18 岁，触发器应该将 age 设置为 18。

   如果新插入的学生年龄大于 120 岁，触发器应该将 age 设置为 120。

   触发器应该记录所有插入操作的原始年龄值到一个名为 age_log 的表中，表结构如下：

   log_id: INT, 主键，自增

   student_id: INT, 学生学号

   original_age: INT，原始年龄值

   log_time: DATETIME，记录插入的时间

   要求：

   (1) 编写 SQL 语句创建 age_log 表。

   (2) 编写 SQL 语句创建 check_student_age_trigger 触发器。

   提示：

   使用 NEW 关键字访问将要插入的记录中的字段。

   使用 INSERT INTO 语句将原始年龄记录到 age_log 表中。

   示例：

   如果插入一条记录：INSERT INTO students (name, age) VALUES ('Alice', 16);

   触发器将插入 age 为 18，并在 age_log 表中记录 original_age 为 16。

```sql
create table age_log
(
    log_id       int primary key auto_increment,
    student_id   int comment '学生学号',
    original_age int comment '原始年龄值',
    log_time     datetime comment '记录插入的时间'
);

create trigger check_student_age_trigger
    before insert
    on students
    for each row
begin
    insert into age_log(student_id, original_age, log_time) value (new.id,
    ↪  new.age, now());
    case
        when (new.age < 18) then set new.age = 18;
```

```
        when (new.age > 120) then set new.age = 120;
        end case;
end;
```

验证一下

```
drop table if exists students;
drop table if exists age_log;

create table students
(
    id   int primary key,
    name varchar(100) comment '姓名',
    age  int comment '年龄'
);

create table age_log
(
    log_id       int primary key auto_increment,
    student_id   int comment '学生学号',
    original_age int comment '原始年龄值',
    log_time     datetime comment '记录插入的时间'
);

create trigger check_student_age_trigger
    before insert
    on students
    for each row
begin
    insert into age_log(student_id, original_age, log_time) value (new.id, new.age,
    ↪  now());
    case
        when (new.age < 18) then set new.age = 18;
        when (new.age > 120) then set new.age = 120;
        end case;
end;

insert into students(id, name, age)
values (10001, 'Alice', 16),
       (10002, 'Bob', 150);
```

students:

	id	name	age
1	10001	Alice	18
2	10002	Bob	120

age_log:

	log_id	student_id	original_age	log_time
1	1	10001	16	2024-11-01 15:39:52
2	2	10002	150	2024-11-01 15:39:52

7. 假设有以下两个表：

   1. **accounts** 表：

列名	数据类型	描述
account_id	INT, 主键, 自增	账户唯一标识符
account_holder	VARCHAR(100)	账户持有者的姓名
balance	DECIMAL(10, 2)	账户余额

   2. **transactions** 表：

列名	数据类型	描述
transaction_id	INT, 主键, 自增	交易唯一标识符
from_account_id	INT	转出账户的 ID
to_account_id	INT	转入账户的 ID
amount	DECIMAL(10, 2)	转账金额
transaction_time	DATETIME DEFAULT CURRENT_TIMESTAMP	转账时间

   创建一个名为 withdraw_funds 的存储过程，用于从银行账户中提款。要求如下：

   存储过程接受两个参数：

   account_id: INT, 要提款的账户的 ID

   amount: DECIMAL(10, 2), 要提款的金额

   在提款前检查该账户的余额是否足够支付提款金额。

   如果余额不足，存储过程应返回一个错误消息。

   如果提款成功，更新账户余额，并在 transactions 表中记录提款信息（转出账户为提款账户，转入账户为 NULL，金额为提款金额）。

```
create procedure withdraw_funds(in _account_id int, in amount decimal(10, 2))
begin
    start transaction;
    -- 如果账户不存在呢？好像没有交代应该怎么处理
    if (select accounts.balance from accounts where accounts.account_id =
    ↪  _account_id) < amount then
        rollback ;
        signal sqlstate '45000' set message_text = '余额不足';
    end if;
    update accounts set balance = balance - amount where accounts.account_id
    ↪  = _account_id;
    insert into transactions(from_account_id, to_account_id, amount) value
    ↪  (_account_id, null, amount);
    commit;
end;
```

验证一下

```sql
drop table if exists accounts;
drop table if exists transactions;

create table accounts
(
    -- 自增好像是从 1 开始的
    account_id     int primary key auto_increment comment '账户唯一标识符',
    account_holder varchar(100) comment '账户持有者的姓名',
    balance        decimal(10, 2) comment '账户余额'
);

create table transactions
(
    transaction_id   int primary key auto_increment comment '交易的唯一标识符',
    from_account_id  int comment '转出账户的 ID',
    to_account_id    int comment '转入账户的 ID',
    amount           decimal(10, 2) comment '转账金额',
    transaction_time datetime default current_timestamp comment '转账时间'
);

insert into accounts(account_holder, balance)
values ('a', 10),
       ('b', 100);

drop procedure if exists withdraw_funds;
create procedure withdraw_funds(in _account_id int, in amount decimal(10, 2))
begin
    start transaction;
    -- 如果账户不存在呢？好像没有交代应该怎么处理
    if (select accounts.balance from accounts where accounts.account_id =
    ↪  _account_id) < amount then
        rollback ;
        signal sqlstate '45000' set message_text = '余额不足';
    end if;
    update accounts set balance = balance - amount where accounts.account_id =
    ↪  _account_id;
    insert into transactions(from_account_id, to_account_id, amount) value
    ↪  (_account_id, null, amount);
    commit;
end;

call withdraw_funds(2, 10);
call withdraw_funds(1, 10);
call withdraw_funds(1, 10);
```

```
[2024-11-01 16:35:33] [45000][1644] 余额不足
```

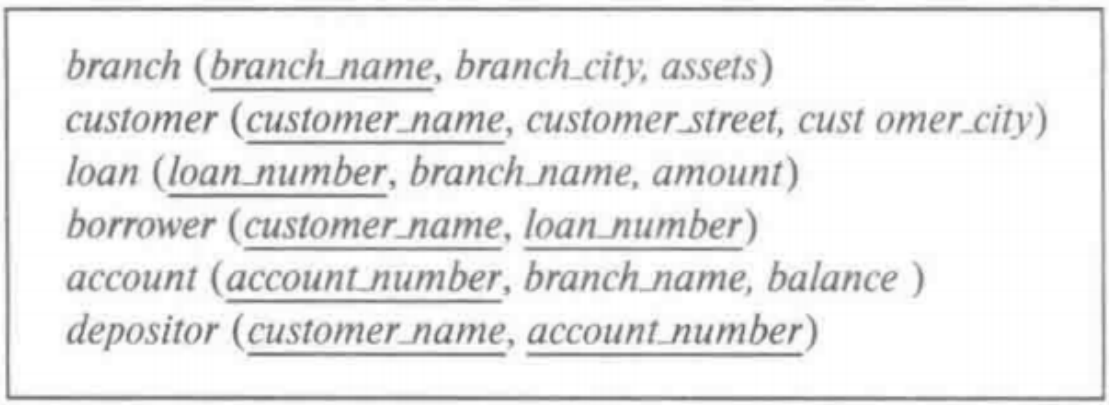
accounts:

	account_id	account_holder	balance
1	1	a	0.00
2	2	b	90.00

transactions:

	transaction_id	from_account_id	to_account_id	amount	transaction_time
1	1	2		10.00	2024-11-01 16:35:33
2	2	1		10.00	2024-11-01 16:35:34

8. 请根据下图所示的银行数据库，编写一个触发器来执行下列操作：在删除一个账户时，检查该账户的拥有者是否还有其他账户，如果没有，则将其从 depositor 关系中删除。



这里为什么要检测是否有其他账户，删除一个账户时不管这个人有没有其他账户都需要从 depositor 里删除 account_number 对应的行吧，那删除最后一个账户时 depositor 里自然也删完了。

```
create trigger delete_account
    after delete
    on account
    for each row
begin
    delete from depositor where account_number = old.account_number;
end;
```

验证一下

```
drop table if exists account;
drop table if exists depositor;

create table account
(
    account_number int primary key,
    branch_name    varchar(100),
```

```
    balance        int
);

create table depositor
(
    customer_name  varchar(100),
    account_number int references account.account_number
);

insert into account
values (1, 'a', 100),
       (2, 'b', 200);

insert into depositor
values ('c', 1),
       ('c', 2);

create trigger delete_account
    after delete
    on account
    for each row
begin
    delete from depositor where account_number = old.account_number;
end;

delete from account where account_number = 1;
select * from depositor;
```

	customer_name	account_number
1	c	2

```
delete from account where account_number = 2;
select * from depositor;
```

	customer_name	account_number

9. 假设有一个名为 employees 的员工表和一个名为 departments 的部门表，结构如下：

1. **employees** 表：

列名	数据类型	描述
employee_id	INT, 主键, 自增	员工唯一标识符
employee_name	VARCHAR(100)	员工姓名
department_id	INT	部门 ID
salary	DECIMAL(10, 2)	员工工资

2. **departments** 表：

列名	数据类型	描述
department_id	INT, 主键, 自增	部门唯一标识符
department_name	VARCHAR(100)	部门名称

（1）存储过程：名为 increase_salary，用于根据部门名称增加员工工资。要求：

接受两个参数：dept_name（部门名称）和 percentage（增加的百分比）。

如果部门不存在，返回错误消息。

如果部门存在，更新该部门所有员工的工资，并返回更新的员工数量。

```
create procedure increase_salary(in dept_name varchar(100), in percentage
↪  int)
begin
    if (select count(*) from departments where dept_name =
    ↪  departments.department_name) <= 0 then
        signal sqlstate '45000' set message_text = '部门不存在';
    end if;
    update employees join departments using (department_id)
    set salary = salary * (1 + percentage / 100)
    where department_name = dept_name;
    -- 存储过程怎么返回更新的员工数量
    -- 30 ms 中有 2 行受到影响 返回的影响行数应该就能反映更新的员工数量了
end;
```

10. （2）函数：名为 get_average_salary，用于获取指定部门的平均工资。要求：

接受一个参数：dept_id（部门 ID）。

返回该部门员工的平均工资，如果没有员工，则返回 0。

```
create function get_average_salary(dept_id int) returns decimal(10, 2)
begin
    declare result decimal(10, 2);
```

```
    if (select count(*) from employees where department_id = dept_id) <= 0
    ↪  then
        set result = 0;
    else
        select avg(employees.salary) into result from employees where
        ↪  department_id = dept_id;
    end if;
    return result;
end;
```

验证一下

```
drop table if exists employees;
drop table if exists departments;
drop procedure if exists increase_salary;
drop function if exists get_average_salary;

create table employees
(
    employee_id   int primary key auto_increment comment '员工唯一标识
    ↪  符',
    employee_name varchar(100) comment '员工姓名',
    department_id int comment '部门 ID',
    salary        decimal(10, 2) comment '员工工资'
);
create table departments
(
    department_id   int primary key auto_increment comment '部门唯一标识
    ↪  符',
    department_name varchar(100) comment '部门名称'
);

create procedure increase_salary(in dept_name varchar(100), in percentage
↪  int)
begin
    if (select count(*) from departments where dept_name =
    ↪  departments.department_name) <= 0 then
        signal sqlstate '45000' set message_text = '部门不存在';
    end if;
    update employees join departments using (department_id)
    set salary = salary * (1 + percentage / 100)
    where department_name = dept_name;
```

```
    -- 存储过程怎么返回更新的员工数量
    -- 30 ms 中有 2 行受到影响 返回的影响行数应该就能反映更新的员工数量了
end;

create function get_average_salary(dept_id int) returns decimal(10, 2)
begin
    declare result decimal(10, 2);
    if (select count(*) from employees where department_id = dept_id) <=
    ↪  0 then
        set result = 0;
    else
        select avg(employees.salary) into result from employees where
        ↪  department_id = dept_id;
    end if;
    return result;
end;

insert into departments
values (101, 'aaa'),
       (102, 'bbb');

insert into employees(employee_name, department_id, salary)
values ('a', 101, 10),
       ('b', 101, 100),
       ('c', 102, 1000);

select get_average_salary(101);
```

	get_average_salary(101)
1	55.00

```
select get_average_salary(103);
```

	get_average_salary(103)
1	0.00

```
call increase_salary('aaa', 50);

[2024-11-02 11:13:13] 30 ms 中有 2 行受到影响

select * from employees;
```

	employee_id	employee_name	department_id	salary
1	1	a	101	15.00
2	2	b	101	150.00
3	3	c	102	1000.00

```
call increase_salary('ccc', 100);
```

```
[2024-11-02 11:22:51] [45000][1644] 部门不存在
```

11. 存储过程、函数和触发器的区别?

存储过程不能使用 return 语句返回结果，只能通过传入参数设置为 out 来返回结果；而函数可以直接使用 return 语句返回结果。 触发器没有传入参数也不能返回结果，只能通过 old 和 new 获取更新前后的值。

12. 触发器的作用？Mysql 表中允许有多少个触发器?

触发器定义了一系列操作，这一系列操作称为触发程序，当触发事件发生时，触发程序会自动运行。 触发器主要用于监视某个表的插入（insert）、更新（update）和删除（delete）等更新操作，这些操作可以分别激活该表的 insert、update 和 delete 类型的触发程序运行，从而实现数据的自动维护。

在 5.7.2 版本以前，同一个表不能创建两个相同触发时间、触发事件的触发程序，那么就是 6 个触发器： before insert、after insert、before update、after update、before delete、after delete。

在 5.7.2 版本之后没有此限制，那么对触发器的数量就没有限制了。