一、 MySQL 用户自定义函数

1.基本语法

语法：

```
create function <db_name>.<func_name> (param1, param2, ...) returns data_type
[funciton options]
begin
body;
return sentence;
end;
```

- 创建自定义函数时，不能与已有的函数名（包括系统名）重复。此外，建议在自定义函数名中统一添加前缀 `fn_` 或者后缀 `_fn` 。
- 函数的参数无需使用 `declare` 命令定义，但它仍属于局域变量，且必须提供参数的数据类型。
- 函数体内部的变量需要用 `declare` 命令定义。
- 自定义函数如果没有参数，则使用空参数 `()` 即可。
- 函数必须指定返回值的数据类型，且需与 `return` 预计中的返回值的数据类型相近（长度可以不同）。

- 函数选项如下:

- `language sql` 默认选项，用于说明函数体用sql编写。
- `[not] deterministic` 如果对相同输入得到相同输出结果，则为 `deterministic` ，否则为 `not deterministic` (默认值)
- 函数体是否包含SQL
  - `contains sql` ： 表明函数体不包含读或写数据的语句（如 `set` ） （默认值）
  - `no sql` ： 表明函数体中不包含 `sql` 语句
  - `reads sql data` ： 包含 `select` 查询
  - `modifies sql data` ： 包含 `update` 或 `delete` 或 `replace`
- `sql security` 用于指定函数的执行许可
- `definer` 表明函数只能有创建者调用
- `invoker` 表明函数可以被其它创建者调用 （默认值）
- `comment` 注释

```
use purchase;
```

示例1: 定义函数查询行号, 每返回一行，行号+1.

```
drop function if exists row_no_fn;

delimiter $$
create function row_no_fn () returns int
  no sql -- 函数体不涉及sql语句
  begin
    set @row_no = @row_no + 1;
    return @row_no;
```

```
  end;
$$ -- 不要忘记加上去
delimiter ;

set @row_no = 0;
select row_no_fn() as '行号', product_id, product_name, Product_Place
from Product limit 10;

-- 等价于
SELECT @row_no := @row_no + 1 as '行号', product_id, product_name, Product_Place
FROM product, (SELECT @row_no := 0) as r;
```

> 注：在 `MySQL` 命令行的客户端中，服务器处理语句默认以 `;` 作为结束标志，如果有一行命令以分号结束，那么按【ENTER】键后，服务器将执行该命令。在存储程序（如函数、触发器、存储过程、事件）中往往涉及到多个命令，如果仍然以分号作为结束标志，那么执行完第一分号语句后，就会认为程序结束。此时，可以通过重新定义结束符来解决该问题。 `delimiter $$` 将结束符设置为 `$$` 。如此设置之后， `MySQL Server` 将不同 `$$` 之间的语句作一个语句块同时处理。
>
> ```
> delimiter $$
> SELECT * FROM sort LIMIT 10;
> SELECT * FROM subsort LIMIT 10;
> $$
> ```
>
> 以上语句中的两个 `select` 语句将同时发送至服务器，返回执行结果。当然，也可以定义其它的结束符，如 `//` 等。

示例2: 定义函数 `get_product_number_fn` ,输入类别编号(`sort_id`)，输出该类别下的产品数量

```
desc product;

delimiter $$
create function get_sum_product_fn (p_sortid char(4)) returns int
  reads sql data -- 函数体涉及select
  begin
    declare count_number int;
    select count(*) into count_number
    from Product
    where sort_id=p_sortid;
    return count_number;
  end;
$$
delimiter ;

select * from Sort;
set @v_sortid = '11';
select @v_sortid as 类别编号, get_sum_product_fn(@v_sortid) as 产品数量;
```

示例3: 定义函数 `get_sum_sort_fn` ，输入产品编号，返回其产品类别下子类别的数量

```
delimiter $$
create function get_sum_sort_fn (p_productid char(5)) returns int
  -- 若在参数中定义varchar，则需提供具体的长度；return和returns类型要一致
  reads sql data  -- 函数需通过sql读取数据
  begin
    declare v_sortid char(2);
    declare v_subsortid int; -- 定义变量，用于存储子类别数量
    select sort_id into v_sortid
    from Product
    where product_id = p_productid;
    select count(subsort_id) into v_subsortid
    from SubSort
    where sort_id = v_sortid;
    return v_subsortid;
  end;
$$
delimiter ;

drop function if exists get_sum_sort_fn;
select get_sum_sort_fn('1001') as 子类别数量;
```

2. 查看函数定义

示例4: 查看用户定义的函数

```
show function status where definer='root@localhost';
show function status where db='purchase';
show function status like 'get_sum%'
```

示例5: 查询完整的函数定义语句

```
show create function get_sum_sort_fn;
show create function get_sum_product_fn;
```

练习1

1. 定义函数 `get_product_number_fn` ,输入产地名称，输出该产地下的产品类别数量
2. 定义函数 `get_member_num_fn` ，输入省份，返回该省份中的用户数量;然后查看该函数的定义。

二、分支与循环语句

1.分支

1.1 `if` 语句

基本语法

```
If 条件表达式1 then 语句块1;
[elseif 条件表达式2 then 语句块2];
...
[else 语句块n]
end if;
```

示例6: 创建函数f, 输入x，如果$x<0$，计算$y=-x$;如果$x>=0$，计算$y=2x$，返回y.

```
delimiter $$
create function f(x float) returns float
    no sql
    begin
        declare y float;
        if (x < 0) then
            set y = -x;
        else
            set y = x * 2;
        end if;
    return y;
    end;
$$
delimiter ;
```

示例7: 创建函数 `change_price_fn` ,输入产品编号 `product_id` ，根据 `sort_id` 的值计算并返回返利: `sort_id` 为11的产品返利为价格的10%, `sort_id` 为21的产品的产品返利为价格的30%， `sort_id` 为31的产品返利为价格的40%, 其它返回为5%。

```
delimiter $$
create function cal_rebate_fn (v_p_id char(2)) returns decimal(10, 2)
    reads sql data
    begin
        declare v_sortid char(2);
        declare v_rebate decimal(10, 2);
        select sort_id into v_sortid
        from product
        where product_id = v_p_id;
        if (v_sortid='11') then
            select price * 0.1 into v_rebate
            from product
            where product_id = v_p_id;
        elseif (v_sortid='21') then
```

```
            select price * 0.3 into v_rebate
            from product
            where product_id = v_p_id;
        elseif (v_sortid='31') then
            select price * 0.4 into v_rebate
            from product
            where product_id = v_p_id;
        else
            select price * 0.05 into v_rebate
            from product
            where product_id = v_p_id;
        end if;
    return v_rebate;
    end;
$$
delimiter ;
```

查询前5条记录的返利情况

```
select product_id, sort_id, cal_rebate_fn(product_id) as rebate
from product limit 5;
```

1.2. `case` 语句

语法：

```
case 表达式
when value1 then 语句块1;
when value2 then 语句块2;
...
when value(n-1) then 语句块n-1;
else 语句块n;
end case;

-- 或者
case
when 表达式1 then 语句块1;
when 表达式2 then 语句块2;
...
when 表达式n-1 then 语句块n-1;
end case;
```

示例8: 创建 `get_week_fn()` 函数，使用该函数根据 `mysql` 系统时间打印星期几

> `weekday` 可以返回和星期相关的数值

```
delimiter $$
create function get_week_fn (p_date date) returns char(20)
    no sql
    begin
```

```
    declare week_day char(20);
    case weekday(p_date)
        when 0 then set week_day = 'Monday';
        when 1 then set week_day = 'Tuesday';
        when 2 then set week_day = 'Wednesday';
        when 3 then set week_day = 'Thursday';
        when 4 then set week_day = 'Friday';
        else set week_day = 'Weekend';
    end case;
    return week_day;
    end;
$$
delimiter ;

drop function get_week_fn;

select weekday(now());

select product_id, product_name, product_date, get_week_fn(product_date) as week
from Product
limit 10;
```

练习2

1. 创建函数 `func` ,输入x，当$x < -5$时，$y = x^3$;当$-5 <= x < 5$时，$y = x$;当$x >= 5$时，$y = 2x + 1$，返回y.
2. 创建函数 `cal_due_fn` ，输入订单号，计算该订单折扣后对应的应付款 `p_due` ，价格计算方式如下：如果产品数量(`quantity`)小于10，则 `p_due = 产品单价 * 数量 * 0.95` ;如果数量在10和50之间，则 `p_due = 产品单价 * 数量 * 0.9` ;如果数量大于50，则 `p_due = 产品单价 * 数量 * 0.8` .

2. 循环

2.1. `while`

当条件表达式为 `true` 时，反复执行循环体，直到表达式值为 `false` 。语法结构：

```
[循环标签:] while 条件表达式 do
循环体;
end while [循环标签];
```

示例9：创建函数 `get_sum_fn` ，输入 `n` ，返回整数1到n的累加和

```
delimiter $$
create function get_sum_fn (n int) returns int
    no sql
```

```
    begin
        declare accum_sum int default 0;
        declare start_num int default 0;
        set accum_sum = 0;
        set start_num = 0;
        while (start_num < n) do
            set start_num = start_num + 1;
            set accum_sum = start_num + accum_sum;
        end while;
    return accum_sum;
    end;
$$
delimiter ;

select get_sum_fn(100);
```

2.2. leave [1]

用于跳出当前的循化语句。语法：

```
leave 循环标签;
```

示例10：基于 `leave` 语句，创建函数 `get_sum_fn2`，输入 `n`，返回整数 `1` 到 `n` 的累加和

```
delimiter $$
create function get_sum_fn2 (n int) returns int
    no sql
    begin
        declare accum_sum int default 0;
        declare start_num int default 0;
        add_num : while true do
            set start_num = start_num + 1;
            set accum_sum = start_num + accum_sum;
            if (start_num = n) then
                leave add_num;
            end if;
        end while add_num;
    return accum_sum;
    end;
$$
delimiter ;

select get_sum_fn2(100);
```

2.3. iterate [2]

用于跳出本次循环，继续下次循环。语法：

```
iterate 循环标签;
```

示例11：基于 `iterate` ，输入 `n` ，对 `1~n` 内所有能被3整除的数加和

```
delimiter $$
create function get_sum_fn3 (n int) returns int
    no sql
    begin
        declare accum_sum int default 0;
        declare start_num int default 0;
        add_num : while true do
            set start_num = start_num + 1;
            if start_num <= n then
                if (start_num % 3) = 0 then
                  set accum_sum = accum_sum + start_num;
                else iterate add_num; -- continue
                end if;
            else
                leave add_num;  -- break
            end if;
        end while add_num;
    return accum_sum;
    end;
$$
delimiter ;

select get_sum_fn3(100);
```

2.4. `repeat`

当条件表达式的值为 `false` 时，反复执行循环，直到条件表达式的值为 `true` 。语法：

```
[循环标签:] repeate
循环体;
until 条件表达式
end repeat[循环标签];
```

示例12：基于 `repeat` 定义函数，输入 `n` ,实现从 `1` 到 `n` 的累加和

```
delimiter $$
create function get_sum_fn4 (n int) returns int
    no sql
    begin
        declare accum_sum int default 0;
        declare start_num int default 0;
        repeat
            set start_num = start_num + 1;
            set accum_sum = start_num + accum_sum;
        until (start_num = n)
        end repeat;
    return accum_sum;
    end;
$$
```

```
delimiter ;

drop function get_sum_fn4;

select get_sum_fn4(100);
```

2.5. loop

loop通常借助 `leave` 语句跳出循环。语法：

```
[循环标签:]loop
循环体;
if 条件表达式 then
    leave [循环标签];
end if;
end loop;
```

示例13：基于 `loop` ，输入 `n` ，实现从 `1` 加到 `n` 的累加和

```
delimiter $$
create function get_sum_fn5 (n int) returns int
    no sql
    begin
        declare accum_sum int default 0;
        declare start_num int default 0;
        add_sum : loop
            set start_num = start_num + 1;
            set accum_sum = start_num + accum_sum;
            if start_num = n then
                leave add_sum;
            end if;
        end loop;
    return accum_sum;
    end;
$$
delimiter ;

drop function get_sum_fn5;

select get_sum_fn5(100);
```

练习3

1. 选用一种循环结构，编写函数get_prod_fn，实现从m到n的累乘
2. 选择一种循环结构，编写程序求三位水仙花数之和 `3` 。

补充：`case...when...then...else`在`SQL`中的应用

```
use mis;
```

1. 在select中使用case...when...

```
case 表达式
when 值1 then 结果1
when 值2 then 结果2
...
when 值n then 结果n
else 结果n+1 end

--
case
when 表达式1 then 结果1
when 表达式2 then 结果2
...
when 表达式n then 结果n
else 结果n+1 end
```

- 示例1：在student表中查询学生出生在周几。

```
-- 使用if()
SELECT s_id, s_name, if(weekday(birthday) = 0,  'Monday',
            if(weekday(birthday) = 1, 'Tuesday',
              if(weekday(birthday) = 2, 'Wednesday',
                if(weekday(birthday) = 3, 'Thursday',
                  if(weekday(birthday) = 4, 'Friday',
                    if(weekday(birthday) = 5, 'Saturday', 'Sunday'))))) as week_day
FROM student
LIMIT 10;

-- 使用case
SELECT s_id, s_name, case weekday(birthday)
                       when 0 then 'Monday'
                       when 1 then 'Tuesday'
                       when 2 then 'Wednesday'
                       when 3 then 'Thursday'
                       when 4 then 'Friday'
                       when 5 then 'Saturday'
                       else 'Sunday' end as week_day
FROM student
LIMIT 10;
```

- 示例2: 查询各门课程男生和女生的平均分

```sql
select * from student limit 10;
select * from course limit 10;
select * from takes limit 10;

-- 普通解法
SELECT c_id, c_name, avg(score) 男生平均分
from course natural join takes natural join student
where gender = '男'
group by c_id;

SELECT c_id, c_name, avg(score) 女生平均分
from course natural join takes natural join student
where gender = '女'
group by c_id;

select *
from (SELECT c_id, c_name, avg(score) 男生平均分
    from course natural join takes natural join student
     where gender = '男'
  group by c_id) a
natural join (SELECT c_id, c_name, avg(score) 女生平均分
    from course natural join takes natural join student
     where gender = '女'
  group by c_id) b
order by c_id;


-- 使用case when
SELECT c_id, c_name, avg(case gender when '男' then score else null end) 男生平均分,
  avg(case gender when '女' then score else null end) 女生平均分
from course natural join takes natural join student
group by c_id
order by c_id;

-- 使用if()
SELECT c_id, c_name, avg(if(gender='男', score, null)) 男生平均分,
  avg(if(gender='女', score, null)) 女生平均分
from course natural join takes natural join student
group by c_id;
```

- 示例3：查询各门课程优、良、中、及格、不及格、缺考（若没有分数，则视为缺考）的人数

```sql
select c_id, c_name, case when score >= 0 and score < 60 then '5-不及格'
                          when score >= 60 and score < 70 then '4-及格'
                          when score >= 70 and score < 80 then '3-中'
                          when score >= 80 and score < 90 then '2-良'
                          when score >= 90 and score <= 100 then '1-优'
                          else '6-缺考' end as 级别,
    count(s_id) 人数
from student natural join takes natural join course
group by c_id, 级别
```

```
order by c_id, 级别;

select c_id, c_name, case when score < 60 then '5-不及格'
                          when score < 70 then '4-及格'
                          when score < 80 then '3-中'
                          when score < 90 then '2-良'
                          when score <= 100 then '1-优'
                          else '6-缺考' end as 级别,
                     count(s_id) 人数
from student natural join takes natural join course
group by c_id, 级别
order by c_id, 级别;
```

2. 更新update

```
-- 准备工作
create table employee(emp_id int primary key auto_increment,
  emp_name varchar(50) not null,
    salary decimal(7, 2));

truncate employee;
insert into employee(emp_name, salary)
values('Miles', 30000), ('Jack', 27000), ('Lucy', 22000), ('Mike', 29000);
```

- 示例4: 完成下述更新

> 假设现在需要根据以下条件对该表的数据进行更新
>
> 1. 对当前工资为30000以上的员工，降薪10%
> 2. 对当前工资为25000以上且不满28000的员工，加薪20%

```
-- 错误方法：分批执行更新
-- 先执行条件1
update employee
set salary = 0.9*salary
where salary >= 30000;

-- 再执行条件2
update employee
set salary = 1.2*salary
where salary >= 25000 and salary < 28000;

-- 正确方法：同批执行更新
-- 条件1和条件2在1个语句中执行
update employee
set salary = case
              when salary >= 30000 then 0.9*salary
              when salary >= 25000 and salary < 28000 then 1.2*salary
              else salary end;
```

1. 相当于 `python` 中的 `break` ↩

2. 相当于 `python` 中的 `continue` 语句. ↩

3. 水仙花数: 对应三位数i，如果它的百位、十位和各位立方之和等于它本身，则这个数为水仙花数。例如153，由于$1^3+5^3+3^3$=155，则153水仙花数。 ↩