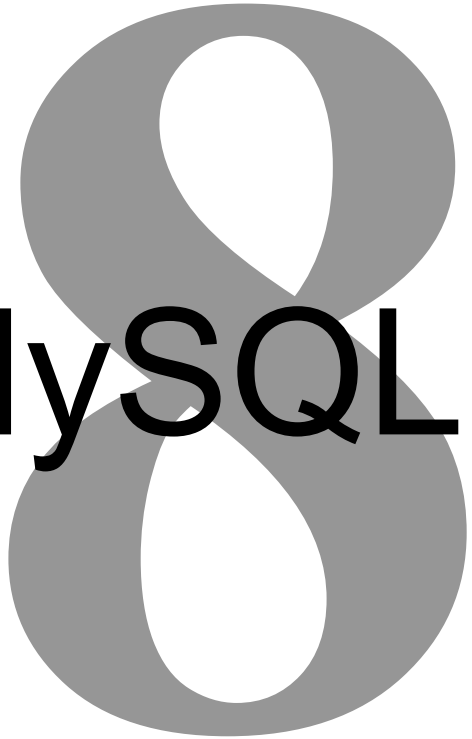


第八讲 MySQL编程基础

教学内容

- 常量和变量
- 运算符和表达式
- 选择结构
- 循环结构**while**
- **LOOP、REPEAT**

8.1 常量和变量

- 常量：固定数据值、字面量、字面值、标量值
- 作用：参与运算或给变量赋值
- 常量类型：5种
  - 字符串型
  - 数值型
  - 日期时间型
  - 布尔值
  - NULL
- 表示方法：因类型而不同

字符串常量

- 用成对单引号“‘”或双引号“"”括起来的字符序列。

  MySQL推荐用单引号表示（区别于其它语言的字符串）。

- 示例
  - **'It\'s a 3.5"disk’**
  - **"I'm from China."**

字符串中的转义字符

字符本义	转义表示
0	\0
换行	\n
回车	\r
制表符	\t
退格符	\b
Ctrl+Z	\Z
'（单引号）	\'
"（双引号）	\"
\（反斜线）	\\
%	\%
_（下划线）	_

- **select 'I\'m a teacher' as a,**

  "you're a student" as b

数值型常量

- 分二进制常量、十进制常量和十六进制常量
- 十进制常量
  - 最常用，直接书写，不需要定界符
  - 例：3，-5，+3.14，123.7E-2（123.7E-2表示123.7×10^{-2}）
- 十六进制常量
  - 需要使用前缀和单引号表示
  - 只加前缀：0x41、0x4d7953514c
  - 结合前缀和单引号：x'41' 、X'41'、x'4d7953514c'

十六进制常量

- 前缀“0x”中的“x”必须小写
- 只能出现0-9、a-f、A-F这16个字符，前缀中的“x”、“X”除外
- 十六进制常量默认是字符串常量的另一种表现形式（每字节2位十六进制数），用CAST(0x41 AS UNSIGNED)可以确保按数值处理。
- 十六进制常量常用于储存图像和电影等格式的数据。
- 函数 HEX(“中国”) 可得到字符串对应的十六进制数值。
- 示例
  - select 0x41, x'41', X'41', 0x4d7953514c, x'4d7953514c’,
  - CAST(0x41 AS UNSIGNED), HEX('中国');
  - #结果为： A A A MySQL MySQL 65D6D0B9FA

二进制常量

- 二进制常量，加前缀“b”、“B”，用单引号括起来
  - b'1011' 、b'0' 、b'1' 、B'1011' 、B'0' 、B'1'
- 二进制常量默认是按字符处理，
  - OCT( )按八进制数值处理
  - BIN( )按二进制数值处理
- 示例
  - select B'111101', BIN(b'111101'+0), OCT(B'111101'+0),HEX(b'111101’),
  - CAST(B'111101' AS UNSIGNED);
  - #结果为： 1111 75 3D 61

日期时间常量

- 格式
  - ‘年-月-日[时:分:秒[.微秒]] ‘ #用单引号括起来
- 事项
  - 年份取值为1000-9999，月份取值1-12，日期取值1-31
  - 省略“时:分:秒[.微秒]”只表示日期，可以精确到微秒级别
  - 给TimeStamp列字段或变量赋值时会根据所处时区自动转换，而且会忽略微秒部分，年份只能取值1970-2037。
  - “年-月-日”之间的分隔符“-”可以换为“\”、“@”、“%”等字符。

布尔值和NULL常量

- 布尔值
  - 只能表示True或False
  - False对应数值0，True对应数值1。
- NULL
  - 用于表示“未知”、“待定”、“没有值”、“无数据”等意义，是一种无类型的值，但不同于0或空字符串的含义。

常量小结

- 常量是一个固定值，用于运算或为变量赋值。
- 分字符串常量、数值常量、日期时间常量、布尔值以及NULL
- 不同类型常量的表示方法不同。
- 十进制常量、布尔值以及NULL常量，无需前缀也不需定界符，其它常量需要前缀、定界符或2者结合才能表示
- 十六进制：“0x”前缀
- 字符串和日期时间常量：单引号
- 字符串常量：双引号定界符（不推荐）
- 二进制和十六进制：前缀+单引号
  - b'Value'、B'Value'、x'Value'、X'Value'

MySQL变量

- 变量用于记录或暂时存放某一时段的状态值
- 用户自定义变量
  - 局部变量
  - 会话变量
- 系统变量
  - 会话变量
  - 全局变量
- 变量包括局部变量和用户变量。用户变量包括会话变量和全局变量

SET定义变量

- set定义变量的形式是以”@”开始，如：”@变量名”。
  - 示例
  - SET @t1=0, @t2=1, @t3=2;
  - SELECT @t1:=(@t2:=1)+@t3:=4,@t1,@t2,@t3;

MySQL变量

- 用户变量
  - 以@开始，形式为@变量名。用户变量跟mysql客户端是绑定的，只对当前用户使用的客户端生效；用户变量使用set语句
- 全局变量
  - set GLOBAL 变量名
  - set @@global.变量名，对所有客户端生效。需具有super权限
- 会话变量
  - 只对连接的客户端有效
- 局部变量
  - 作用范围：begin…end语句块之间；declare专门用于定义局部变量
- set语句是设置不同类型的变量，包括会话变量和全局变量

查看系统变量

- 语法
  - **SHOW [GLOBAL | SESSION | LOCAL] VARIABLES [LIKE 模式字符串]**
- 示例
  - **show global variables; #查看所有的系统变量**
  - **show global|【session】 variables like '%char%';**
  - **select @@global|【session】.系统变量名； #查看某个系统变量**

会话用户变量定义

- 形式1：
  - SET @var1 = 值1 [, @var2 := 值2, …]
- 形式2：
  - SELECT 值1 INTO @var1 [, 值2 INTO @var2, …]
- 形式3：
  - SELECT @var1 := 值1 [, @var2 := 值2, …]
- 变量名必须以“@”字符开头
- “:=”是赋值运算符

局部变量定义

- 语法
  - **DECLARE var1 [, var2] ... 数据类型 [DEFAULT 默认值]**
- 说明
  - 局部变量不可以用“@”字符开头。
  - **DECLARE只能用于BEGIN…END语句块在开头部分定义局部变量**
  - 作用范围：只能在**BEGIN…END**语句块中使用。
    不能用于其它封装语句块或别的地方。
  - 函数和存储过程的形式参数也属于局部变量

关于变量名称

- 系统变量名多数都以2个“@”开头
- 用户会话变量必须以1个“@”开头
- 局部变量则不能以“@”开头

SET设置变量值

- 语法
  - SET [GLOBAL | [SESSION | LOACAL| @@ | @]变量名1 = 值1| @@global. | @@ [session. | @@local. | @@ | @]变量名1 = 值1
  - SELECT 值1 INTO [@]变量名1 [, 值2 INTO [@]变量名2, …]
  - SELECT [@] 变量名1 := 值1 [, @] 变量名2 := 值2, …]
- 说明
  - 全局系统变量：GLOBAL或@@global.
  - 会话系统变量：SESSION、LOCAL、@@session.、@@local.或@@前缀。

变量示例

- DECLARE var1, var2, var3 INT;
- SET var1=10, var2=20;
- SET var3=var1+var2;

- DECLARE Sname VARCHAR(100);
- SELECT studname INTO Sname FROM StudInfo WHERE StudNo='20181152001';

常量变量举例

- #系统变量SQL_SELECT_LIMIT返回SELECT最大行数
- Set @@SQL_SELECT_LIMIT = default;
- Select @@local.SQL_SELECT_LIMIT,
  @@session.SQL_SELECT_LIMIT, @@SQL_SELECT_LIMIT,
  @@global.SQL_SELECT_LIMIT into @x1, @x2, @x3, @x4;
- Set @@SQL_SELECT_LIMIT := 10

常量变量举例

```
DELIMITER $$
CREATE PROCEDURE sp_proc( )
BEGIN
    DECLARE x1, x2, x3 int default 0;
    SET x2=@x1;
    select 'p1:' as pos, x1, x2, x3, @x1;
    BEGIN
        DECLARE x1, x2 int default 1;
        select 'p2:' as pos, x1, x2, x3, @x1;
    END;
  select 'p3:' as pos, x1, x2, x3, @x1;
END $$
DELIMITER ;
```

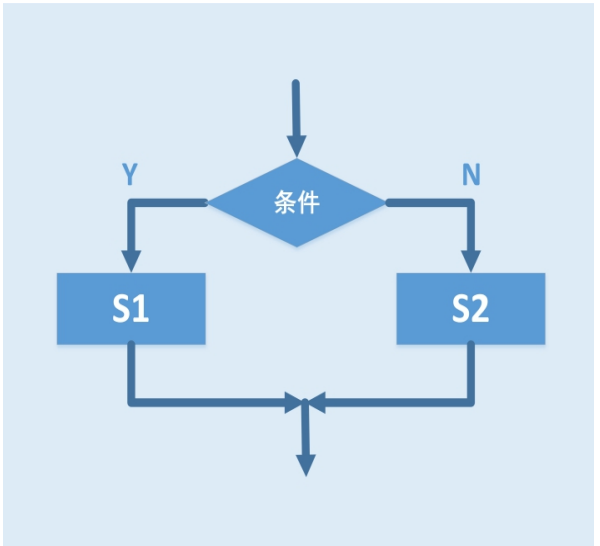
MySQL 分支结构

- 顺序结构
  - 程序从上而下执行
- 分支结构
  - 程序从两条或多条路径中选择一条去执行
- 循环结构
  - 程序在满足一定条件的基础上，重复执行一段代码

IF函数

- IF(expr1,expr2,expr3)
- 示例
  - SELECT *,if(studgender='男', 'Male','Female')
  - FROM StudInfo;

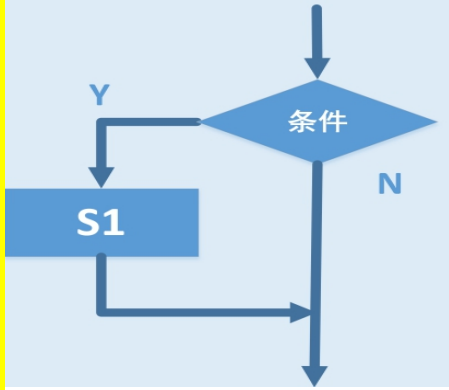
二项选一



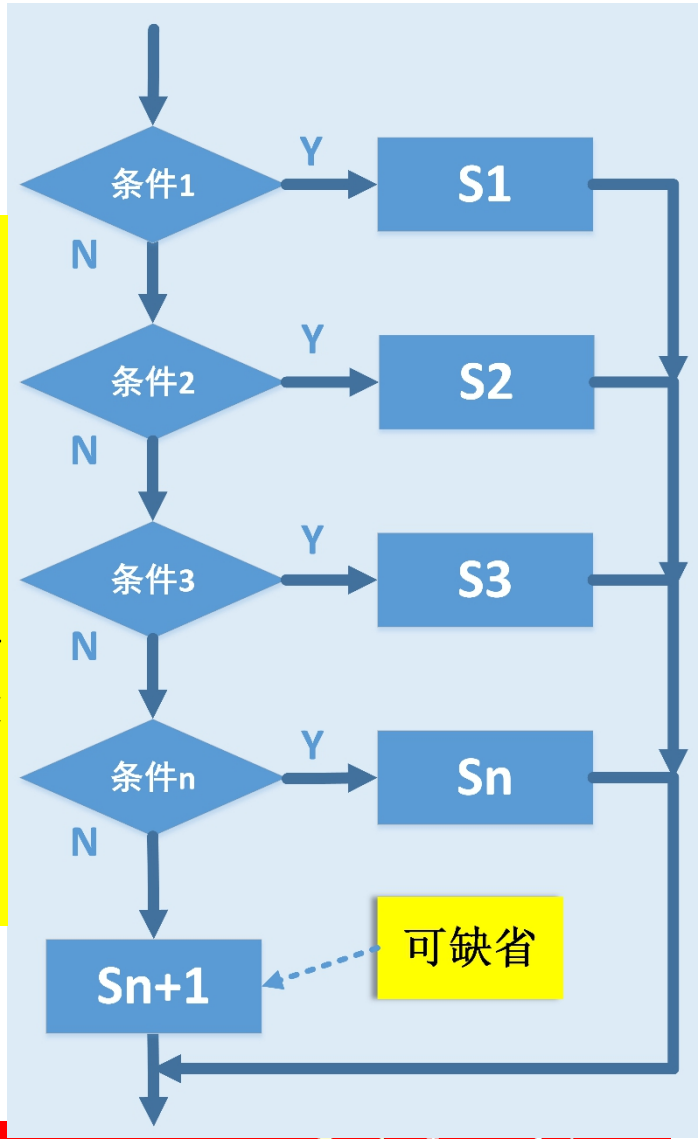
IF多分支

- IF search_condition THEN
- statement_list
- [ELSEIF search_condition THEN]
- statement_list ...
- [ELSE
- statement_list]
- END IF
- #只能应用在 begin end 中

单项选择



多项选择



IF示例

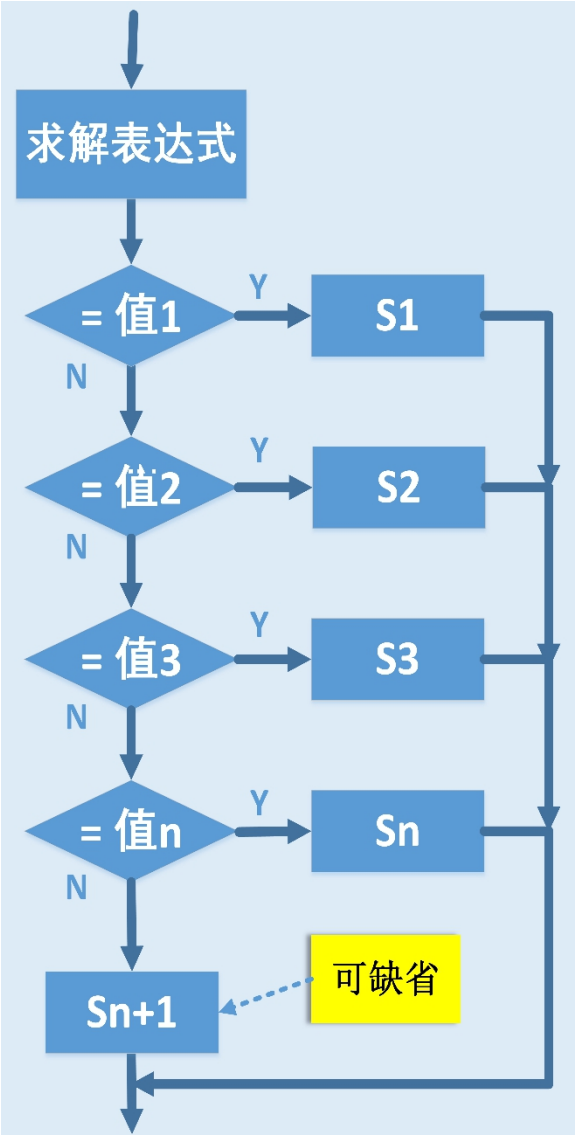
```
create procedure proc_IF()
BEGIN
  Declare I int;
  Set I=5;
  If I>5 then
    set I=I*2;
  else
    set I=I mod 2;
  END if;
  select I;
end;
call proc_if;
```

IF多分支示例

```
DELIMITER $
CREATE FUNCTION testCase(score int) returns char
BEGIN
    IF score>=90 and score<=100 then return 'A';
      ELSEIF score>=80 and score<90 then return 'B';
      ELSEIF score>=70 and score<80 then return 'C';
      ELSEIF score>=60 and score<70 then return 'D';
      else return 'E';
   end if;
END $
DELIMITER ;
#调用：select testCase(23);
```

简单CASE语句

- CASE case_value
- WHEN when_value THEN statement_list
- [WHEN when_value THEN statement_list]
- ...
- [ELSE statement_list]
- END CASE

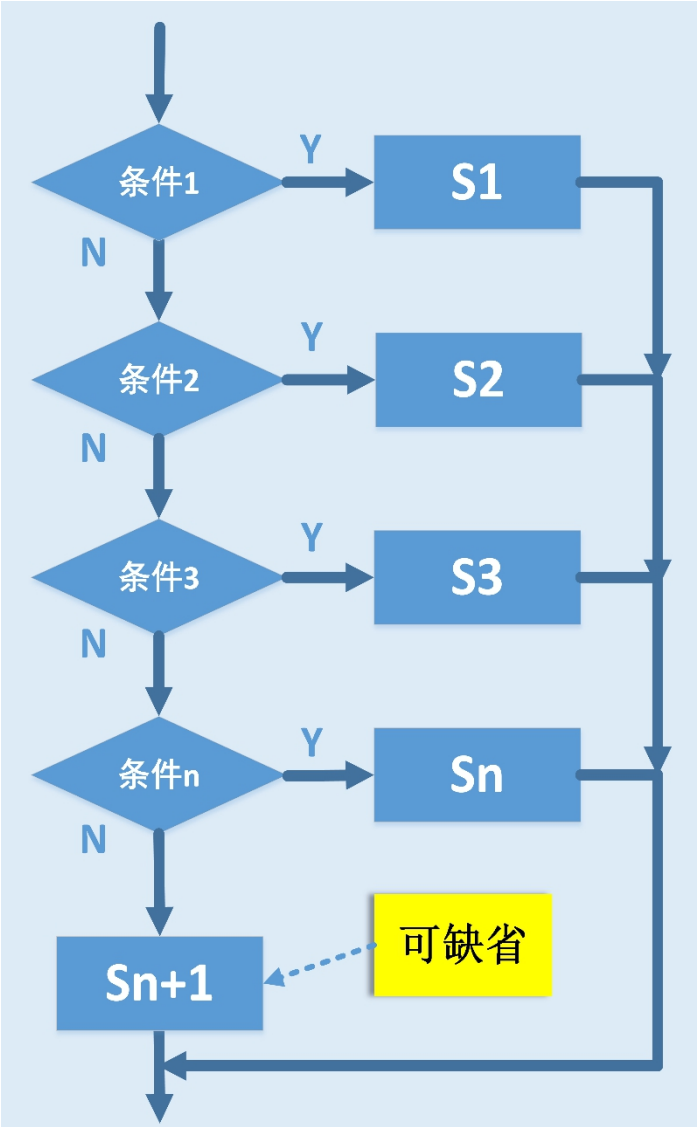


简单CASE示例

```
create procedure proc_case2()
BEGIN
  Declare a int;
  Declare answer varchar(10);
  set a= round(rand()*10);
  Case a
   When 1 then set answer= 'A';
   When 2 then set answer= 'B';
   When 3 then set answer= 'C';
   else
       set answer= 'No';
  End case;
  select answer ;
end;
```

搜索CASE语句

- CASE
- WHEN search_condition THEN statement_list
- [WHEN search_condition THEN statement_list]
- ...
- [ELSE statement_list]
- END CASE



搜索CASE示例

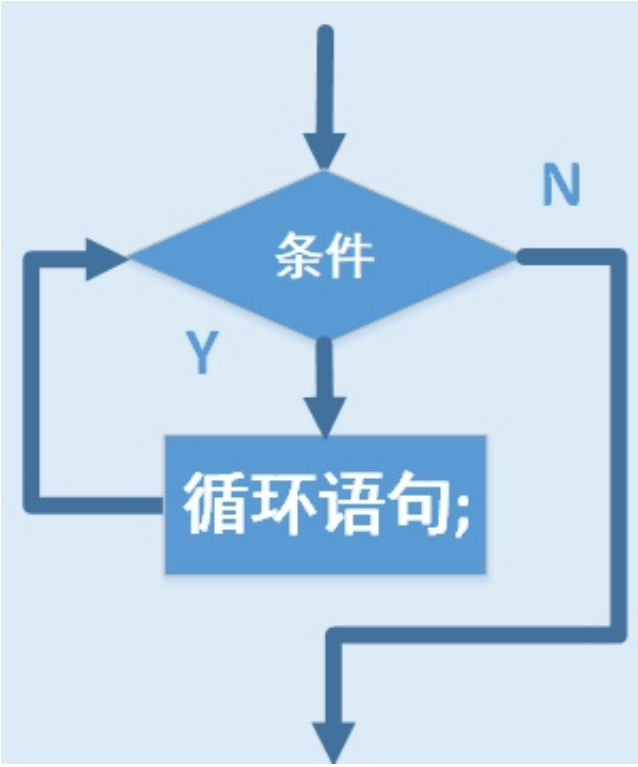
```
create procedure proc_case3()
BEGIN
  Declare a int;
  Declare answer varchar(10);
  set a= round(rand()*10);
  Case When a=1 then set answer= 'A';
   When a<=2 then set answer= 'B';
   When a>3 then set answer= 'C';
   When a<5 then set answer= 'D';
   When a=5 then set answer= 'E';
   else
       set answer= 'Others';
  End case;
  select a,answer ;
end;
CALL proc_case3;
```

循环语句

- WHILE……DO……END WHILE
- REPEAT……UNTIL END REPEAT
- LOOP……END LOOP

while 循环

- 语法
  - while 条件 do
  - 循环体;
  - end while;



While示例1

```
delimiter $$
drop procedure if exists while_loop;
create procedure while_loop()
  begin
    declare i int default 2;
    declare R int default 5;
    while i>0 do
        set R=R+power(i,2);
        set i=i-1;
    end while;
     select R;
  end $$
delimiter;
call while_loop;
```

While示例2

```
delimiter $$
drop procedure if exists while_loop;
create procedure while_loop()
  begin
    declare i int default 3;
    declare R int default 10;
    while i<R do
            set i=i+R MOD 3;
            set R=R- i mod 4;
    end while;
    set R=i+R;
     select R;
  end $$
delimiter;
call while_loop;
```

S=1+2+3+…+100

```
delimiter $$
 drop procedure if exists proc_add;
create procedure proc_add(in m int)
begin
  declare s bigint default 0;
  declare i int default 1;
  while i <=m DO
    set s=s+i;
    set i = i +1;
  end while;
  select s;
end $$
delimiter ;
call proc_add(100);
```

S=N!

```
delimiter $$
 drop procedure if exists proc_jc;
create procedure proc_jc(in n int)
begin
  declare i int;
  declare k bigint;
  set s=0,i=1,k=1;
  while i <=n DO
    set k=k*i;
    set i = i +1;
  end while;
 select k;
end $$
delimiter ;
call proc_jc(5);
```

S=1!+2!+3!+…+N!

```
delimiter $$
 drop procedure if exists proc_jc;
create procedure proc_jc(in n int)
begin
  declare s bigint default 0;
  declare i int default 1;
  declare k bigint default 1;
  while i <=n DO
    set k=k*i;
    set s=s+K;
    set i = i +1;
  end while;
 select s;
end $$
delimiter ;
call proc_jc(5);
```

S=1!+3!+5!…+N!

```
delimiter $$
 drop procedure if exists proc_jc;
create procedure proc_jc(in n int)
begin
  declare s bigint;
  declare i int;
  declare k bigint;
  set s=0,i=1,k=1;
  while i <=n DO
    set k=k*i;
    if i mod 2=1 then
      set s=s+K;
    end if;
    set i = i +1;
  end while;
 select s;
end $$
delimiter ;
call proc_jc(5);
```

自动编号表

```
CREATE TABLE Auto_Test(
   AutoID int auto_increment primary key,
   AutoName varchar(20)
);
```

产生记录

```
DELIMITER $$
CREATE PROCEDURE pro_insert(IN maxNum INT)
BEGIN
    DECLARE i INT DEFAULT(1);
    while i<=maxNum DO
        INSERT INTO auto_Test(AutoName) VALUES (CONCAT('sc',i));
        SET i=i+1;
    END WHILE;
END $$
DELIMITER ;
CALL pro_insert(20);
```

结构标识符

- 语法
  - 标识名字:**While 条件 do**
  - **If 条件判断 then**
  - 循环控制;
  - **Iterate/leave 标识名字;**
  - **End if;**
  - 循环体
  - **End while [标识名字];**
- 说明
  - **iterate：迭代，就是以下的代码不执行，重新开始循环（continue）.**
  - **leave：离开，整个循环终止（break）.**

While示例

```
delimiter $$
drop procedure if exists lopp;
create procedure lopp()
  begin
    declare i int default 1;
    lp1:while i<10 do
       set i=i+1;
            if i=3 then
                  Iterate lp1;
            end if;
            if i > 5 then
                  leave lp1;
            end if;
            select i;
   end while;
  end $$
delimiter;
call lopp;
```

While示例1

```
delimiter $$
drop procedure if exists while_loop;
create procedure while_loop()
  begin
    declare i int default 10;
    wp:while i>0 do
       set i=i-1;
            if i =8  then
                  Iterate wp;
            end if;
            if i=2 then
                  leave wp;
            end if;
            if i mod 2= 0 then
              select i;
            end if ;
   end while;
  end $$
delimiter;
call while_loop;
```

While示例2

```
delimiter $$
drop procedure if exists while_loop;
create procedure while_loop()
  begin
    declare i int default 10;
    declare R int default 4;
    wp:while i>=0 do
            set i=i-2;
          if i=2 then
              iterate wp;
          end if;
            if i mod 3=0 then
                set R=R+i;
       end if;
    end while;
    select R;
  end $$
delimiter;
call while_loop;
```

loop 循环

- loop_name:loop
- if 条件 THEN -- 满足条件时离开循环
- leave loop_name; -- 和 break 差不多
- end if;
- end loop;

Loop示例 S=1+2+3+…+100

```
drop procedure if exists  sums;
create procedure sums(a int)
begin
        declare sum int default 0;
        declare i int default 1;
        loop_name:loop -- 循环开始
            if i>a then
                leave loop_name;
            end if;
            set sum=sum+i;
            set i=i+1;
        end loop;  -- 循环结束
        select sum; -- 输出结果
end;
call sums(100);
```

LOOP示例

```
delimiter $$
drop procedure if exists lopp;
create procedure lopp()
  begin
    declare i int default 1;
    lp1:LOOP
          set i=i+1;
            if i=3 then
                  Iterate lp1;
            end if;
            if i > 5 then
                  leave lp1;
            end if;
            select i;
   end LOOP;
  end $$
delimiter;
call lopp;
```

repeat 循环

- repeat
- 循环体
- until 条件 end repeat;

Repeat示例 S=1+2+3+…+100

```
drop procedure if exists sum55;
 create procedure sum55(a int)
 begin
     declare sum int default 0;
    declare i int default 1;
     repeat -- 循环开始
        set sum=sum+i;
        set i=i+1;
     until i>a end repeat; -- 循环结束
     select sum; -- 输出结果
 end;
call sum55(100);
```

Repeat循环

```
delimiter //
drop procedure if exists looppc;
create procedure looppc()
begin
   declare i int default 1;
   repeat
   INSERT INTO auto_Test(AutoName) VALUES (CONCAT('sc',i));
   set i = i + 1;
   until i >= 20
   end repeat;
end //
call looppc();
```

下次课内容

- 9.1 系统内置函数
  - 9.1.1 系统函数
  - 9.1.2 字符串函数
  - 9.1.3 日期函数
  - 9.1.4 数学函数
  - 9.1.5 CASE函数
  - 9.1.6 系统内置函数应用
- 9.2 自定义函数
  - 9.2.1 自定义函数简介
  - 9.2.2 创建和使用自定义函数
  - 9.2.3 自定义函数示例