5 Advanced SQL

- 从编程语言访问 SQL
  - Dynamic SQL （动态的，传送）
  - Embedded SQL（嵌入式，写代码时将sql语句写入，需要编译）
- **Function and Procedure【函数和存储过程】**
  - 说明
    - 由SQL的处理组件或由外部编程语言定义
    - 此处我们介绍的是SQL标准
  - 声明一个SQL函数
    - **create function** dept_count (dept_name varchar(20))
      returns integer
      begin
      declare d_count **integer**;
      select count (*) into d_count **from** instructor **where** instructor.dept_name = dept_name **return** d_count; **end**
    - 可以直接使用：dept_count(dept_name)
  - **table function**:SQL 标准支持返回一个表的函数
    - **create function** instructor_of (dept_name char(20))
      returns table（ID varchar(5)，name varchar(20), dept_name varchar(20), salary numeric(8,2))
      return table (select ID, name, dept_name, salary from instructor where instructor.dept_name = instructor_of.dept_name)
    - 使用：select* from table(instructor_of('Music'))
  - 函数可以被写作**procegure**
    -
      ```
      create procedure dept_count_proc (in dept_name varchar(20),
                                        out d_count integer)
        begin
          select count(*) into d_count
          from instructor
          where instructor.dept_name = dept_count_proc.dept_name
        end
      ```
    - 可以使用call语句从SQL procedure或嵌入式SQL调用procedure。
      ```
      declare d_count integer;
      call dept_count_proc( 'Physics', d_count);
      ```
    - 也可以从动态SQL调用procedure和函数
    - 只要具有相同名称的procedure的参数数目不同，SQL就允许有多个procedure。

- 名称以及参数的数量用于标识procedure。
- SQL 支持的编程语言
  - 复合语句: **begin** … **end,**
    - 可能包含多个SQL语句 between **begin** and **end**.
    - 局部变量可以在复合语句中声明
  - While and repeat statements:
    - **while** boolean expression **do**
      sequence of statements ;
      end while
    - **repeat**
      sequence of statements ;
      until boolean expression
      end repeat
  - **For** loop
    - Permits iteration over all results of a query
  - Example: Find the budget of all departments

```
declare n integer default 0;
for r as
    select budget from department
    where dept_name = 'Music'
do
        set n = n + r.budget
end for
```

  - 条件语句 (**if-then-else**)

```
if boolean expression
 then statement or compound statement
elseif boolean expression
 then statement or compound statement
 else statement or compound statement
end if
```

- **Trigger【触发器】**
  - 概述
    - 触发器的常用功能
      - 完成比约束更复杂的数据约束。
      - 触发器可以检查T-SQL所做的操作是否被允许。例如：在产品库存表里，如果要对某种产品出库，触发器可以检查该产品最低库存数。
      - 当一个T-SQL语句对数据表进行操作的时候，触发器可以根据该T-SQL语句的操作情况来对另一个数据表进行操作。
      - 触发器也可以调用一个或多个存储过程。
      - 在T-SQL语句执行完之后，触发器可自动调用“数据库邮件”来发送邮件。
    - DML触发器

- AFTER触发器：只有执行某一操作INSERT、UPDATE、DELETE之后触发器才被触发,进行约束检查等动作都在After触发器被激活之前发生。After触发器只能用于表。
- INSTEAD OF触发器：即当对表进行INSERT、UPDATE 或 DELETE 操作时，系统不是直接对表执行这些操作，而是把操作内容交给触发器，让触发器检查所进行的操作是否正确，如正确才进行相应的操作。因此，INSTEAD OF 触发器的动作要早于表约束的处理。可以用于视图

- 创建触发器

DML触发器

```
CREATE TRIGGER <触发器名>
ON  <表名>|<视图名>
FOR|AFTER|INSTEAD OF
[INSERT][,UPDATE][,DELETE]
AS
    <T-SQL语句> | <语句块>
```

  - FOR|AFTER|INSTEAD OF：指定触发器触发的时机。
  - AFTER指定在相应操作（INSERT、UPDATE、DELETE）成功执行后才触发。FOR=AFTER
  - INSTEAD OF指定执行DML触发器用于“代替”引发触发器执行的INSERT、UPDATE或DELETE语句。

- 使用
  - 两个表：inserted 和 deleted，临时，执行之后消失
  - update：先删除后插入

以上内容整理于 幕布文档