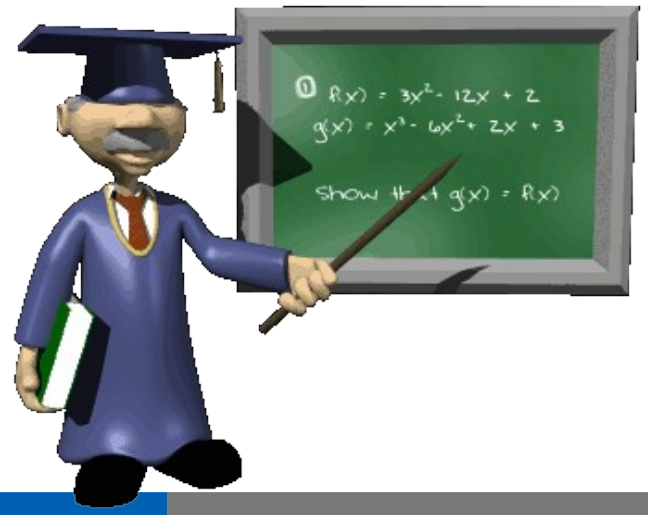
前言

SQL 是非过程化的查询语言，缺少流程控制能力，难以实现应用业务中的逻辑控制。

SQL 编程技术有效克服 SQL 语言实现复杂应用方面的不足，提高应用系统和 DBMS 间的互操作性。

应用系统中使用的 SQL 编程技术有：

- ✓ 嵌入式 SQL
- ✓ 存储过程和自定义函数
- ✓ 过程化 SQL
- ✓ ODBC 、 JDBC 、 OLE DB
- ✓ ……



SQL 编程

01 T-SQL 语言

02 SQL Server 游标

03 SQL Server 触发器

1. SQL Server 变量

① 局部变量：用户定义的变量，以 @ 开头 .

② 全局变量：系统变量，用户只能使用不能定义，以 @@ 开头 .

@@ERROR	当事务成功时为 0，否则为最近一次的错误号
@@ROWCOUNT	返回受上一语句影响的行数
@@FETCH_STATUS	返回最近的 FETCH 语句执行后的游标状态
@@VERSION	返回 SQL Server 当前安装的日期、版本和处理器类型

2. DECLARE 语句

用于定义一个局部变量，变量以 @ 开头 .

DECLARE @ 变量 1　数据类型 1，　@ 变量 2　数据类型 2，……

DECLARE 语句定义局部变量后，可以用 SET 或 SELECT 语句为局部变量赋值 .

- ✓ **SELECT：一次可赋值多个变量，或显示多个表达式的值**
- ✓ **SET：一次仅能给一个变量赋值**

```
DECLARE @sname char(20)，　@avg int，　@cno char(4)
SET @sname=' 李丽 '
SELECT @cno=Cno，　@avg=AVG(Grade) FROM SC GROUP BY Cno
```

3. 运算符

- 算术运算符： + ， - ， * ， / ， %(取余)
- 比较运算符： > ， >= ， < ， <= ， = ， <> ， !=
- 逻辑运算符： AND ， OR ， NOT
- 位运算符： &(按位与) ， |(按位或) ， ~(按位非) ， ^(按位异或)
- 字符串连接运算符： +

4. 函数

SQL Server 函数主要有：

- 数学函数
- 字符串函数
- 日期和时间函数
- 聚合函数
- 系统函数
- ……

① 绝对值函数 abs

② 随机数函数 rand

③ 四舍五入函数 round

④ 上取整函数 ceiling

⑤ 下取整函数 floor

⑥ 指数函数 exp

⑦ 平方根函数 sqrt

4. 函数

SQL Server 函数主要有：

- 数学函数
- 字符串函数
- 日期和时间函数
- 聚合函数
- 系统函数
- ……

表 7-1　部分字符串函数

函数名	函数功能
charindex(*expr*1, *expr*2 [, *start_location*])	返回字符串中指定表达式的起始位置。
left(*character_expr*, *integer_expr*)	返回从字符串左边开始指定字符个数的字符串。
len(*string_expr*)	返回给定字符串的长度(不包含尾随空格)。
lower(*character_expr*)	将大写字符转换为小写字符后返回字符表达式。
ltrim(*character_expr*)	删除起始空格后返回字符表达式。
replicate(*character_expr*, *integer_expr*)	以指定的次数重复字符表达式。
right(*character_expr*, *integer_expr*)	返回字符串中右边的 *integer_expr* 个字符。
rtrim(*character_expr*)	截断所有尾随空格后返回一个字符串。
space(*integer_expr*)	返回由重复的空格组成的字符串。
str(*float_expr* [, *length* [, *decimal*]])	由数字数据转换来的字符数据。
substring(*expr*, *start*, *length*)	提取子串函数。
upper(*character_expr*)	返回将小写字符数据转换为大写的字符表达式。

4. 函数

SQL Server 函数主要有：

- 数学函数
- 字符串函数
- 日期和时间函数
- 聚合函数
- 系统函数
- ……

表 7-2　日期和时间函数

函数名	函数功能
dateadd(*datepart*, *number*, *date*)	在指定日期上加一段时间，返回新的 datetime 值
datediff(*datepart*, *startdate*, *enddate*)	返回两个指定日期的日期和时间边界数
datename(*datepart*, *date*)	返回指定日期的指定日期部分的字符串
datepart(*datepart*, *date*)	返回指定日期的指定日期部分的整数
day(*date*)	返回指定日期中日(day)的整数
getdate()	返回当前系统日期和时间
getutcdate()	返回世界时间坐标或格林尼治标准时间的 datetime 值
month(*date*)	返回指定日期中月(month)的整数
year(*date*)	返回指定日期中年(year)的整数

4. 函数

SQL Server 函数主要有：

- 数学函数
- 字符串函数
- 日期和时间函数
- 聚合函数
- 系统函数
- ……

表 7-3　系统函数

函数名	函数功能
convert(*data_type*[(*length*)],*expr*[,*style*])	将某种数据类型的表达式转换为另一种数据类型
current_user()	返回当前的用户，等价于 user_name()
datalength(*expr*)	返回任何表达式所占用的字节数
@@*ERROR*	返回最后执行的 SQL 语句的错误代码
isnull(*check_expr*,*replacement_value*)	使用指定的替换值替换 NULL
@@*ROWCOUNT*	返回受上一语句影响的行数
session_user()	返回当前会话的用户名
user_name()	返回给定标识号的用户名
host_name()	返回工作站名称
user()	当前数据库用户名

5. BEGIN…END 语句

该语句用来定义一组 SQL 语句构成的块。

```
BEGIN
        语句块
END
```

6. IF…ELSE 语句

该语句用来定义有条件执行某些语句， ELSE 是可选的。

IF 布尔表达式
　　语句
[ELSE [IF 布尔表达式] 语句]

7. WHILE 语句

该语句用来表示当布尔表达式 (即条件) 成立时重复执行某语句或语句块。

WHILE　布尔表达式
　　语句

8. BREAK 与 CONTINUE 语句

用于提早结束循环：

BREAK

用于结束本次循环，继续下一次循环：

CONTINUE

9. GOTO 语句

用于无条件将语句的执行顺序转到标号处。

GOTO　标号

10. RETURN 语句

用于无条件退出一个查询或一个过程:

RETURN [整型表达式]

这里可选的整数表达式返回给调用者一个状态值 .

11. WAITFOR 语句

用于定义某天中的一个时刻执行一个语句块 .

WAITFOR　DELAY 'time' | TIME 'time'

DELAY 后的时间说明需要等待的时间 ;

TIME 后的时间说明要等到哪个时刻 .

```
waitfor delay '00:00:05'        -- 等待 5 秒钟
```

12. PRINT

PRINT 文本字符串
| @ 字符类型变量
| @@ 返回字符串结果的函数
| 字符串表达式

函数 CONVERT(数据类型，表达式)

将表达式转换成指定数据类型 .

```
PRINT ' 成绩 =' + CONVERT(char(6), @grade)
```

PRINT 后只能是字符串，所以需要将 int 型变量 @grade 转换为字符串类型

SQL 编程

01 T-SQL 语言

02 SQL Server 游标

03 SQL Server 触发器

1. 什么是游标

游标是一种能从包括多条数据记录的结果集中每次提取一条记录的机制。

因为主语言中主变量一次只能存放一条记录，而一条 SQL 语句可产生或处理多条记录。通过使用游标，可逐一获取记录，交由主语言进一步处理。

2. 使用游标的步骤

① 定义游标 DELCARE

② 打开游标 OPEN

③ 逐行提取游标集中的行 FETCH

④ 关闭游标 CLOSE

⑤ 释放游标 DEALLOCATE

定义游标的语法:

declare < 游标名 > cursor

for <SELECT 语句 >

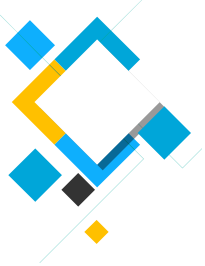
[for [read only| update { of < 属性列名列表 >}]

read only： 当前游标集中元组仅可查询，不可修改。

update： 默认，当前游标集中的元组可更新。

① 定义游标

```
declare mycur cursor for
    select Sname, Grade
    from Student, Course, SC
    where Student.Sno=SC.Sno and SC.Cno=Course.Cno and Cname=' 数学 '
for read only
```



```
open mycur
fetch mycur into @sname, @gra
while (@@fetch_st      =0)
```

② 打开游标

open < 游标名 >

执行一次 **fetch** 语句，系统将

③ 逐行获取当

f

注意

中的

下移

fetch 语句的执行状态保存在全局变量 *@@fetch_status* 中：

0： fetch 语句成功，已经从游标集中获取了元组值

1： 所指定的位置超出了游标结果集合的范围，读不到数据

2： 所要读取的行已从游标结果集合中删除 .

```
    print ' 平均分 =' + convert (char(4), @sumgrade/@count)
else
    print ' 平均分 =0'
close mycur
deallocate mycur
```

④ 关闭游标

游标名 >

⑤ 释放游标

deallocate < 游标名 >

3. 当前游标集的修改与删除

游标可以放在触发器和存储过程中使用，可对游标集中的当前元组执行删除和修改操作。

① 删除游标集中的当前元组

DELETE FROM < *表名* >

WHERE CURRENT OF < *游标名* >

② 修改游标集中的当前元组

UPDATE < *表名* >

SET < *列名* > = < *表达式* > [, < *列名* > = < 表达式 >...]

WHERE CURRENT OF < *游标名* >

例：创建一个游标，显示选修了“数学”课程且成绩不及格的学生姓名和成绩，并从数据库中删除该选课记录。

```
declare @sname char(20), @grade int
declare mycur cursor for
    select Sname, Grade
    from Student, Course, SC
    where Student.Sno=SC.Sno and SC.Cno=Course.Cno and Grade<60 and Cname=' 数学 '
open mycur
fetch mycur into @sname, @grade
while (@@fetch_status=0)
  begin
    select @sname, @grade
    delete from SC where current of mycur
    fetch mycur into @sname, @grade
  end
close mycur
deallocate mycur
```

① 定义游标，缺省表示可修改

② 打开游标

③ 逐行获取当前游标值□变量

显示学生姓名和成绩
删除当前游标指向的选课记录
获取下一个游标值

④ 关闭游标

⑤ 释放游标

SQL 编程

1. 什么是存储过程

- ✓ 存储过程：为完成特定功能由 SQL 语句和控制流语句构成的语句集合。
- ✓ 存储过程能够通过接收输入参数，并以输出参数的形式返回多个参数给调用存储过程的过程。

2. 使用存储过程的优点

① 运行效率高：存储过程不像 SQL 语句那样在提出操作请求时才进行语法分析和优化工作，因而它提供了在服务器端快速执行 SQL 语句的有效途径。

② 降低了客户机和服务器之间的通信量：应用程序只需要向服务器发出存储过程的名字和参数，就可以让 RDBMS 执行许多条 SQL 语句，并把结果返回给客户端。

③ 方便实施企业的业务操作：可以把企业复杂的业务操作编写存储过程放入数据库服务器，从而便于集中控制和维护 .

④ 实现一定程度的安全性保护：用户对存储过程只有执行权限，没有查看权限。

3. SQL Server 中存储过程的类型

① 系统存储过程

由 SQL Server 提供的存储过程，可作为命令执行。

系统存储过程主要存储在 master 数据库中，并以 sp_ 为前缀，主要是从系统表中获取信息。

② 用户存储过程

由用户创建的完成某一特定功能的存储过程。

4. 创建存储过程

```
CREATE Procedure 过程名 ( [ @参数1 类型1 [output],
                           @参数2 类型2 [output], ...])
AS
    SQL 语句
```

例：账户信息表 ACCOUNT(ACCOUNTNUM， TOTAL)，定义存储过程实现：从账户 a 转账指定金额到账户 b

```
CREATE PROCEDURE TRANSFER(@a INT， @b INT，
AS
BEGIN
    DECLARE @totalDeposit FLOAT
    SELECT @totalDeposit=total FROM ACCOUNT WHERE
    IF @totalDeposit IS NULL
    BEGIN
        PRINT '账户不存在'
        RETURN
    END
    IF @totalDeposit<@amount
    BEGIN
        PRINT '账户存款不足'
        RETURN
    END
    UPDATE account SET total=total-@amount WHERE ACCOUNTNUM=@a
    UPDATE account SET total=total + @amount
END
```

定义存储过程 TRANSFER， 3 个输入参数：
转入账号 b 、转出账号 a 、转账金额 amount

定义局部变量 totalDeposit， 求转出账号 a 的余额并保存 totalDeposit 中 .

若 totalDeposit 为空，则说明该账户不存在并退出 .

若 totalDeposit 小于 amount 则说明该账户余额不足并退出 .

否则修改账户 a 和账户 b 的余额，进行转账

5. 执行存储过程

EXEC 存储过程名 实参 1, 实参 2,…

例 : 调用存储过程 TRANSFER 实现从帐户 01003388 转 10000 元到帐户 01113456 中 .

EXEC TRANSFER 01113456, 01003388, 10000

6. 删除存储过程

DROP PROCEDURE 存储过程名

例：删除用户存储过程 TRANSFER 。

DROP PROCEDURE TRANSFER

例：定义存储过程 out_spro ， 返回选修了指定课程号的学生人数及该课程平均成绩。

```
CREATE PROCEDURE out_spro (@count int output, @g
AS
        select @count=count(Sno), @grade=avg(Grade)
        from SC
        where Cno=@cno
```

output 表示参数为 " 返回参数 ", 其值可以返回给调用它的 EXEC 语句

执行存储过程 out_spro ， 求选修 1 号课程的学生人数及 1 号课程平均成绩。

```
declare @t1 int , @t2 real
EXEC out_spro @t1 output, @t2 output, '1'
print ' 选修了 1 号课程的学生人数为 : '+ str (@t1, 2)+ ', 平均分为 : ' + str(@t2,6,2)
```

SQL 编程

01 T-SQL 语言

02 SQL Server 游标

03 SQL Server 触发器

1. 什么是触发器

触发器 Trigger—— 定义在关系表上的由事件驱动的特殊过程，又称“事件 - 条件 - 动作规则”。当事件（对表的增删改操作或事务结束）发生时，对条件进行检查，若条件满足则执行动作。

用于保证完整性，可以进行更为复杂的完整性检查和违约处理。

触发器的执行是由触发事件（用户对关系表进行的增、删、改操作）激活，并由数据库服务器自动执行。

注意：不同 RDBMS 产品实现触发器的语法各不相同。

2. 定义触发器

CREATE TRIGGER < 触发器名 >

ON {< 表名 >|< 视图名 >}

FOR < 触发事件 >

AS < 触发动作体 >

触发器创建者应为表的拥有者 .
注意 : 数据库中触发器名唯一 .

ON 子句包含了在其上执行触发器的表或视图 .

触发事件:
INSERT 、 DELETE 、 UPDATE

触发器动作体是触发器执行的 SQL 语句 , 这些语句放在 BEGIN…END 块中 .

3. inserted 表与 deleted 表

✓在使用触发器的过程中， SQL Server 的每个触发器有两个特殊的表： inserted 表和 deleted 表 .

✓由系统管理的临时表，存储在内存中而不在数据库中。不允许用户修改，但可引用这两个表的数据，当触发器工作完成，这两个表被删除。

① 若一个表上定义了 **insert** 触发器，当对该表执行插入操作时，所有的插入行都有一个副本存放到 **inserted** 表中 .

② 若一个表上定义了 **delete** 触发器，当对该表执行删除操作时，所有删除行存放在 **deleted** 表中，当需要时可从 **deleted** 表中恢复。

③ 若一个表上定义了 **update** 触发器，当对该表执行更新操作时，在 **deleted** 表中存放旧值，在 **inserted** 表中存放新值。

例 1：创建一触发器 check_score，向 SC 表中插入数据时，检查插入的成绩是否在 0 到 100 之间。

```
create  trigger check_score
on SC for insert
As
    declare @score int
    select @score=grade from inserted
    if @score<0 or @score>100
     begin
       RAISERROR(' 成绩必须在 0 到 100 之间 ',16,1)
       ROLLBACK TRANSACTION
  end
```

- 定义局部整型变量 @score
- 用 SELECT 语句为变量赋值

生成错误消息并启动会话的错误处理

事务撤销，即把刚才所作 insert 修改操作撤销 .

例 2：创建触发器 Check_UpdateGrade，当对 SC 表中 Grade 的值进行修改，就自动在成绩变化表：Grade_Log(Sno，Cno，Oldgrade，Newgrade，Username，Dat) 中增加一条相应记录。

```
create  trigger Check_UpdateGrade
on SC for update
as
 if update(grade)
 begin
    INSERT INTO Grade_Log
        SELECT inserted.Sno，inserted.Cno，inserted.Grade，deleted.Grade，
                    CURRENT_USER，CURRENT_TIMESTAMP
        FROM  inserted，deleted
        WHERE inserted.Sno=deleted.Sno  AND  inserted.Cno=deleted.Cno
end
```

如果 **Grade** 属性的值被修改才插入记录，
而 **Sno** 或 **Cno** 的值被修改则不会插入记录到成绩变化表

将学号、课程号、修改前成绩、修改后成绩、修改操作者、修改时间插入到 **Grade_Log** 表。

4. 激活触发器

- ✓触发器的执行是由触发事件激活的，并由数据库服务器自动执行。
- ✓一个数据表上可以定义多个触发器。

5. 删除触发器

DROP TRIGGER < 触发器名 >

触发器必须是一个已经创建的触发器，并且只能由具有相应权限的用户删除 .

```
DROP  trigger Check_UpdateGrade
```