

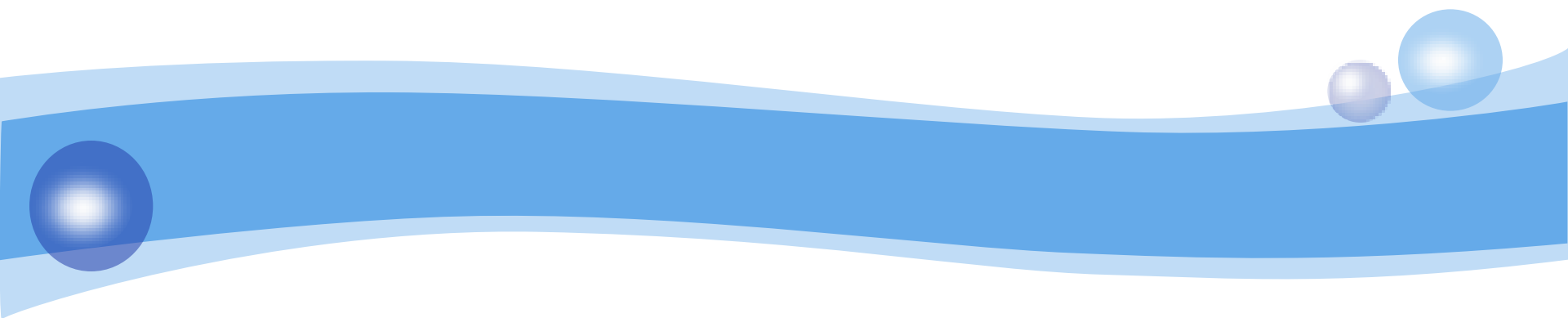


# 数据库原理

Theory of Database

李静

信息科学与技术学院

- 
- 一. 存储过程
- 二. 触发器

# 存储过程概述

- ❖ 存储过程是SQL Server服务器上一组预编译的Transact-SQL语句，用于完成某项任务，它可以接受参数、返回状态值和参数值，并且可以嵌套调用。  
SQL提供了一种方法，它可以将**一些固定的操作集中**起来由数据库服务器来完成，以实现某个任务，这种方法就是存储过程。

# 存储过程概述

- ❖ 存储过程在第一次执行时进行语法检查和编译，之后驻留在系统的内存中，再次调用时不必进行编译。
- ❖ 存储过程可以由应用程序多次激活，提高重复任务的执行性能
- ❖ 存储过程具有输入参数和输出参数，通过这些参数返回结果集。
- ❖ 存储过程可以实现多种功能，如数据表数据的查询、添加记录、修改记录、和删除记录以及复杂的数据处理。

# 存储过程的优点

## 1、模块化程序设计

存储过程只需创建一次并存储在数据库中，即可被应用程序反复调用，用户可以独立于应用程序对存储过程进行修改。

## 2、提高执行速度

存储过程在首次运行时候编译，之后就常驻内存，再次调用时不必进行编译，也不必从磁盘调入内存。

# 存储过程的优点

## 3、降低网络通信量

包括数百行**T-SQL**语句的存储过程，可以通过一次执行过程代码的语句来执行，不需要在网络中发送数百行代码，减少了**T-SQL**语句代码在网络上的传输量。

## 4、保证系统的安全性

系统管理员可以设置用户对存储过程的操作权限，避免非授权用户对数据的访问。

# 存储过程的类型

❖ SQL Server存储过程的类型包括：

- 系统存储过程
- 用户自定义存储过程
- 临时存储过程
- 远程存储过程
- 扩展存储过程

# 存储过程的类型

## 1、系统存储过程

系统存储过程是由系统提供的存储过程，可以作为命令执行各种操作。系统存储过程定义在系统数据库master中，其前缀是sp\_。

系统存储过程允许系统管理员执行修改系统表的数据库管理任务，可以在任何一个数据库中执行。

当创建一个新数据库时，一些系统存储过程会在新数据库中被自动创建。



# 存储过程的类型

## 2、用户自定义存储过程

完成用户指定的某一特定的数据库操作，只能在当前数据库中创建。

其名称不能以sp\_为前缀。

用户存储过程的名称在数据库中必须唯一，可以附带参数，完全由用户创建和维护。

# 存储过程的类型

## 3、临时存储过程

属于本地存储过程。如果本地存储过程的名称前面有一个“#”，该存储过程就称为局部临时存储过程，只能在一个用户会话中使用。

如果本地存储过程的名称前有两个“##”，该过程就是全局临时存储过程，可以在所有用户会话中使用。

使用临时存储过程必须创建本地连接，当SQL Server关闭后，这些临时存储过程将自动被删除。

# 存储过程的类型

## 4、远程存储过程

远程存储过程指从远程服务器上调用的存储过程。

## 5、扩展存储过程

在SQL Server环境之外执行的动态链接库称为扩展存储过程，其前缀是sp\_。

使用时需要先加载到SQL Server 系统中，并且按照使用存储过程的方法执行。

扩展存储过程只能添加到master数据库中。

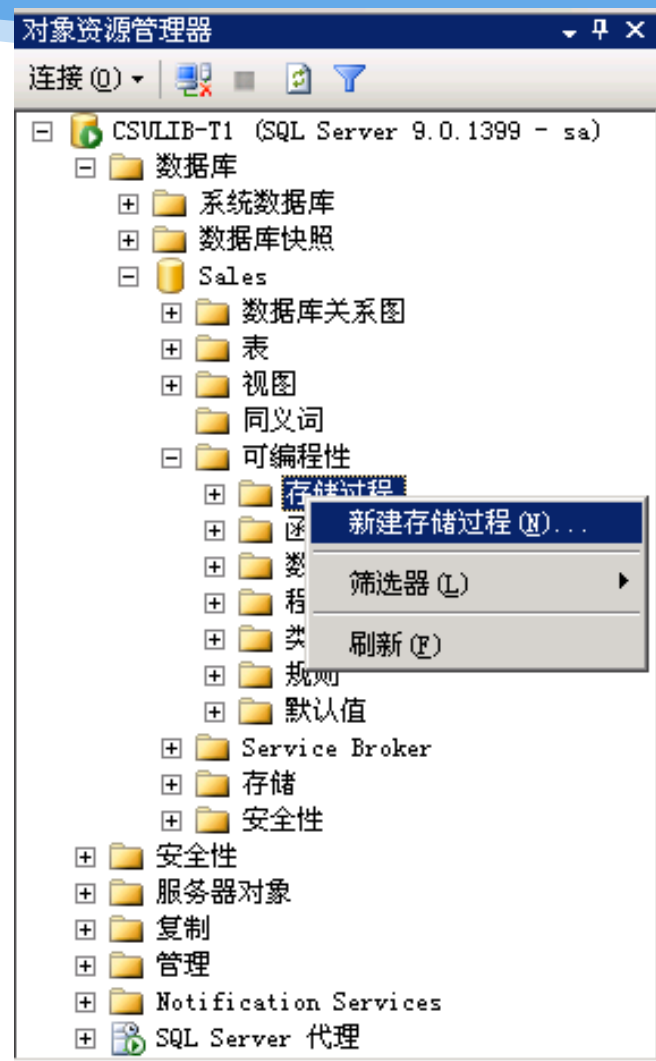
# 创建存储过程

在**SQL Server**中，可以使用二种方法创建存储过程：

- ① 利用**SSMS**图形化界面工具创建存储过程。
- ② 利用**Transact-SQL**语句创建存储过程。

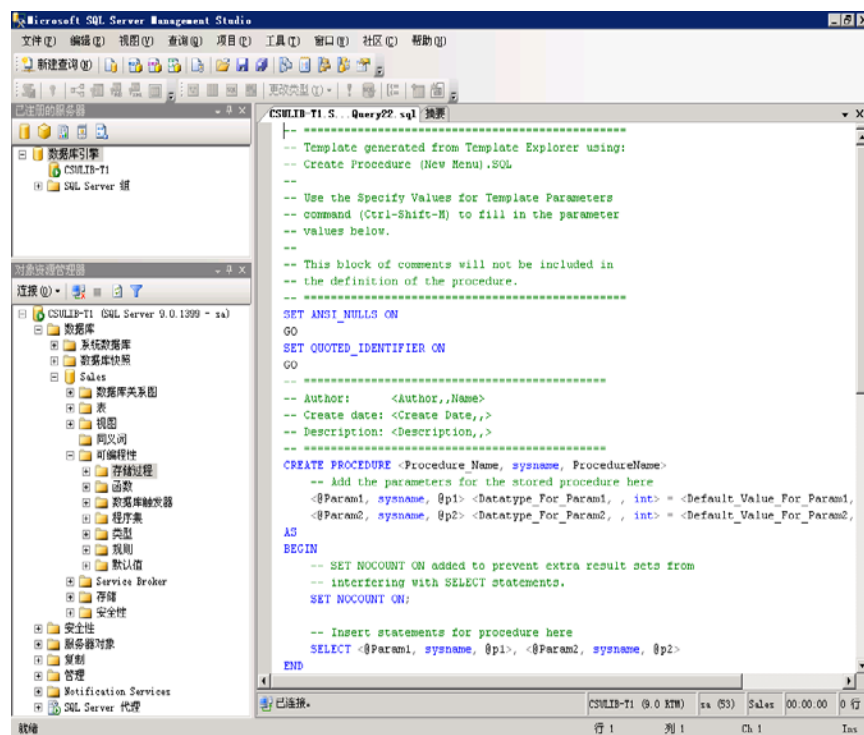
# 创建存储过程—SSMS

- 启动ServerManagement Studio。
- 在“对象资源管理器中”展开要创建存储过程的“数据库”。
- 展开“可编程性”→“存储过程”，在窗口的右侧显示出当前数据库的所有存储过程。
- 单击鼠标右键，在弹出的快捷菜单中选择“新建存储过程”命令。



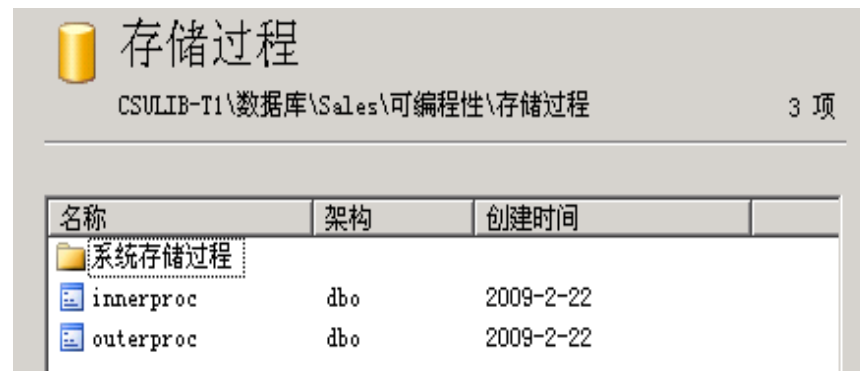
# 创建存储过程—SSMS

- 在打开的SQL命令窗口中，系统给出了创建存储过程命令的模板。
- 在模板中可以输入创建存储过程的Transact-SQL语句。
- 单击“执行”按钮即可创建存储过程。



# 创建存储过程—SSMS

- 建立存储过程的命令被成功执行后，在“对象资源管理器”→“数据库服务器”→“可编程性”→“存储过程”中可以看到新建立的存储过程。
- 编辑存储过程代码时，可通过“查询”→“指定模板参数的值”，在对话框中完成存储过程的相关定义。



# 创建存储过程—T-SQL

❖ 使用CREATE PROCEDURE语句创建存储过程应该考虑以下几个方面：

- (1) 在一个批处理中，CREATE PROCEDURE语句不能与其他SQL语句合并在一起。
- (2) 数据库所有者具有默认的创建存储过程的权限，它可把该权限传递给其他的用户。
- (3) 存储过程作为数据库对象其命名必须符合标识符的命名规则。
- (4) 只能当前数据库中创建属于当前数据库的存储过程。



# 创建存储过程—T-SQL

```
CREATE PROC [EDURE] [owner.] procedure_name [{@parameter data_type}  
  [VARYING] [=default] [OUTPUT]  
  ] [ ,...n ]  
  [WITH {RECOMPILE|ENCRYPTION|RECOMPILE , ENCRYPTION}]  
AS sql_statement [ ...n ]
```

参数说明：

**Varing**：该关键字用于存储过程的输出参数为游标的情况。

**默认值**：用于输入参数。

**With recompile**：存储过程不驻留在内存中，而是每次执行重新编译。

**With encryption**：对它的SQL语句加密，其他用户无法查询。

**For replication**：存储过程只在复制过程中执行，与上一个不同时用。

**SQL语句**：不适用于创建数据库、索引、表、规则、触发器、过程。

# 创建存储过程—T-SQL

例如：

```
Create procedure pro_stu
```

```
As select * from student
```

# 执行存储过程

❖ 执行存储过程的语法格式:

[[EXEC[UTE]]

{[@return\_status=]

procedure\_name [;number]|@procedure\_name\_var}

[[@parameter=]{value|@variable

[OUTPUT]][DEFAULT]]

[ ,...n ]

[WITH RECOMPILE ]

# 存储过程举例

例： 创建存储过程，从表goods和表goods\_classification的联接中返回商品名、商品类别、单价。

```
CREATE PROCEDURE goods_info AS  
SELECT goods_name, classification_name, unit_price  
FROM goods g JOIN goods_classification gc  
ON g.classification_id = gc.classification_id
```

## 存储过程举例

例：执行上例中存储过程goods\_info

❖ 在SQL查询分析器中输入命令：

EXEC goods\_info

❖ 运行的结果：

	goods_name	classification_name	unit_price
1	IBM R51	笔记本计算机	9999.0000
2	旭日 160-D1.7G	笔记本计算机	9499.0000
3	NEC S3000	笔记本计算机	9900.0000
4	HP1020	激光打印机	1550.0000
5	Canon LBP2900	激光打印机	1380.0000
6	HP3938	喷墨打印机	450.0000
7	LS-106C	交换机	2500.0000

# 创建存储过程注意事项

对于存储过程要注意下列几点：

- (1) 用户定义的存储过程只能在当前数据库中创建。
- (2) 成功执行 CREATE PROCEDURE 语句后，过程名称存储在 sysobjects 系统表中，而 CREATE PROCEDURE 语句的文本存储在 syscomments 中。
- (3) 自动执行存储过程:SQL Server 启动时可以自动执行一个或多个存储过程。这些存储过程必须由系统管理员在 master 数据库中创建，并在 sysadmin 固定服务器角色下作为后台过程执行。这些过程不能有任何输入参数。
- (4) sql\_statement的限制:除了 SET SHOWPLAN\_TEXT 和 SET SHOWPLAN\_ALL外，其它SET 语句均可在存储过程内使用。

# 执行存储过程注意事项

存储过程的执行要注意下列几点：

- (1) 如果存储过程名的前三个字符为 `sp_`，SQL Server 会在 Master 数据库中寻找该过程。如果没能找到合法的过程名，SQL Server 会寻找所有者名称为 `dbo` 的过程。
- (2) 参数可以通过 `value` 或 `@parameter_name = value` 提供。
- (3) 执行存储过程时，若语句是批处理中的第一个语句，则不一定要指定 `EXECUTE` 关键字。

## 存储过程的几种情况——带输入参数

例如：向学生表中添加一行数据的存储过程

```
Create procedure pro_stu1
```

```
@num char(6),@name char(20), @sex char(2)
```

```
As
```

```
Insert into student(sno, sname, ssex)
```

```
Values(@num, @name, @sex)
```

运行：

```
Exec pro_stu1 '200501','张力' , '女'
```



# 存储过程的几种情况——带默认值

例如：向学生表中添加一行数据的存储过程

```
Create procedure pro_stu2
```

```
@num char(6)='200502',
```

```
@name char(20)='李莉' ,@sex char(2)='女'
```

```
As
```

```
Insert into student(sno, sname, ssex)
```

```
Values(@num, @name, @sex)
```

运行：

```
Exec pro_stu2
```

没有指定输入参数的值，则按照默认值去执行。

# 存储过程的几种情况——带输出参数

例如：

创建一个存储过程，由于返回指定学号学生所选课程的总成绩

```
Create procedure pro_stu3
```

```
    @num char(6),    @g int output
```

```
As
```

```
    Select @g=sum(grade)    From sc
```

```
    Where sno=@num and grade is not null
```

运行存储过程：

```
Declare @total int
```

```
Exec pro_stu3 '200502', @total output
```

# 存储过程的返回状态

用变量接受返回值:

返回0: 存取执行;

返回-1: 对象丢失;

.....;

返回-99: .....

例如:

```
declare @status int
```

```
Exec @status=pro_stu1.....
```

## 例： 使用带参数的存储过程

【例1】从XSCJ数据库的三个表中查询某人指定课程的成绩和学分。该存储过程接受与传递参数精确匹配的值。

```
CREATE PROCEDURE student_info1 @name char (8),@cname char(16)
AS
SELECT a.sno, sname, cname, grade, credit
FROM student a INNER JOIN sc b
ON a.sno = b.sno INNER JOIN course t
ON b.cno= t.cno
WHERE a.sname=@name and t.cname=@cname
```

student\_info1 存储过程有多种执行方式，下面列出了一部分：

```
EXECUTE student_info1 '王林', '计算机基础'
```

或者 EXECUTE student\_info1 @name='王林', @cname='计算机基础'

或者 EXECUTE student\_info1 @cname='计算机基础', @name='王林'

# 例：使用带有通配符参数的存储过程

【例2】从三个表的连接中返回指定学生的学号、姓名、所选课程名称及该课程的成绩。该存储过程在参数中使用了模式匹配，如果没有提供参数，则使用预设的默认值。

```
CREATE PROCEDURE st_info    @name varchar(30) = '刘%'
AS
SELECT a.sno, a.sname, c.cname, b.grade
FROM student a INNER JOIN sc b
ON a.sno =b.sno INNER JOIN course c
ON c.cno= b.cno
WHERE sname LIKE @name
```

st\_info 存储过程可以有多种执行形式，下面列出了一部分：

```
EXECUTE st_info    /*参数使用默认值*/
```

或者 EXECUTE st\_info '王%' /\*传递给@name 的实参为'王%'\*/

# 例子：使用带OUTPUT参数的存储过程

【例3】用于计算指定学生的总学分，存储过程中使用了一个输入参数和一个输出参数。

```
CREATE PROCEDURE totalcredit @name varchar(40),  
    @total int OUTPUT  
AS  
SELECT @total= SUM(credit)  
FROM student a INNER JOIN sc b ON a.sno =b.sno  
    INNER JOIN course c ON c.cno= b.cno  
WHERE sname=@name GROUP BY a.学号
```

注意： OUTPUT 变量必须在创建表和使用该变量时都进行定义。

定义时的参数名和调用时的变量名不一定要匹配，不过数据类型和参数位置必须匹。

```
DECLARE @t_credit char(20),@total int  
EXECUTE totalcredit '王林', @total OUTPUT
```

# 用户存储过程的修改

使用ALTER PROCEDURE命令可修改已存在的存储过程并保留以前赋予的许可。

**ALTER** procedure 存储过程名称[; 版本号]  
[{参数 数据类型} [varying] [=默认值] [output].....]  
[with {recompile | encryption |  
recompile,encryption}]  
[for replication]  
As SQL语句

说明： 各参数含义与CREATE PROCEDURE相同。

# 示 例

用 **ALTER PROCEDURE** 更改后，过程的权限和启动属性保持不变。

【例1】对存储过程student\_info1进行修改。

**USE XSCJ**

**GO**

**ALTER PROCEDURE student\_info1**

**@name char(8),@cname char(16)**

**AS**

**SELECT a.sno, sname, cname, grade, credit**

**FROM student a INNER JOIN sc b ON a.sno =b.sno**

**INNER JOIN course c ON c.cno= b.cno**

**WHERE a.sname=@name and t.cname=@cname**

**GO**



# 示 例

【例2】创建名为 `select_students` 的存储过程，默认情况下，该过程可查询所有学生信息，随后授予权限。当该过程需更改为能检索计算机专业的学生信息时，用 `ALTER PROCEDURE` 重新定义该存储过程。

**USE XSCJ**

**GO**

**CREATE PROCEDURE select\_students /\*创建存储过程\*/**

**AS**

**SELECT \* FROM student ORDER BY sno**

**GO**

修改存储过程**select\_students**

**ALTER PROCEDURE select\_students WITH ENCRYPTION**

**AS**

**SELECT \* FROM student**

**WHERE sdept= '计算机' ORDER BY sno**

**GO**

# 用户存储过程的删除

语法格式:

**DROP PROCEDURE { procedure } [ ,...n ]**

说明:

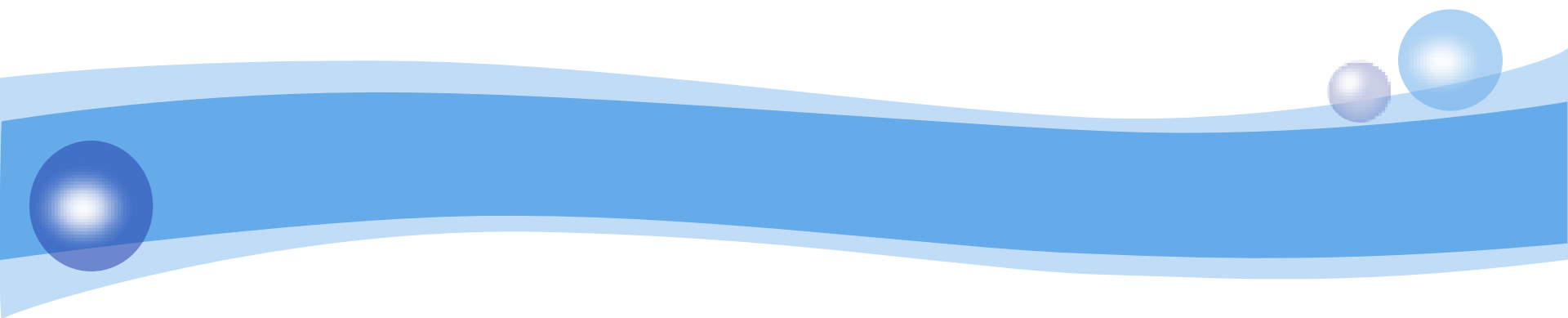
**procedure**指要删除的存储过程或存储过程组的名称; **n**: 表示可以指定多个存储过程同时删除。

**【例】**删除 XSCJ数据库中的student\_info1 存储过程。

**USE XSCJ**

**GO**

**DROP PROCEDURE student\_info1**

- 
- 一. 存储过程
  - 二. 触发器

# 为什么需要触发器

- ❑ 为什么需要触发器(TRIGGER)呢？典型的应用就是银行的取款机系统

帐户信息表bank

	customerName	cardID	currentMoney
1	张三	1001 0001	1000 0000

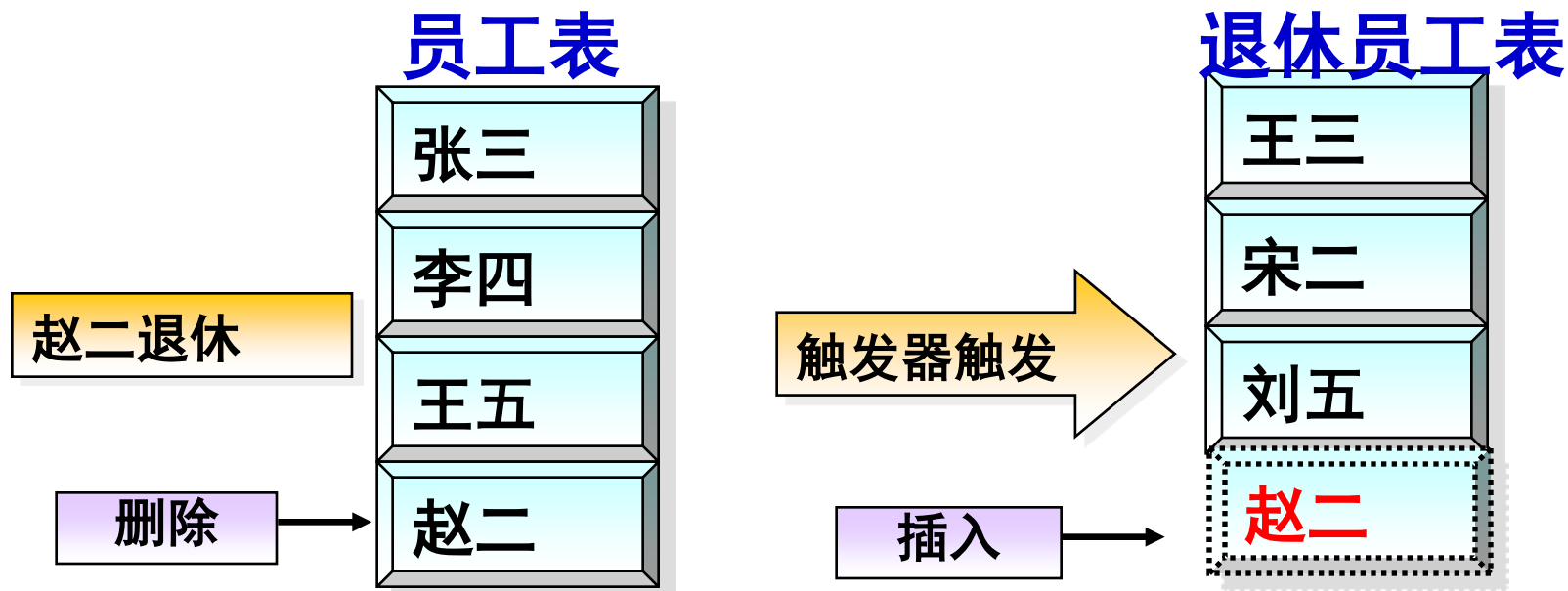
张三开户1000元,  
李四开户1元

最优的解决方案就是采用**触发器**：

- 它是一种特殊的存储过程
- 也具备事务的功能
- 它能在多表之间执行特殊的业务规则

演示： 为什么需要触发器.sql

# 什么是触发器



# 触发器

- ❖ **触发器**是一种特殊的存储过程,其特殊性在于它并不需要由用户直接调用,当对表进行插入、删除、修改等操作时自动执行。
- ❖ **触发器**可以用来实施复杂的完整性约束,以防止对数据的不正确修改。
- ❖ **触发器**不允许带参数、也不允许被调用。
- ❖ **触发器**不能返回任何结果。

# 触发器的类型

- ❖ DELETE 触发器
- ❖ INSERT 触发器
- ❖ UPDATE 触发器

# 触发器工作原理

## ❑ 触发器触发时：

- ❑ 系统自动在内存中创建deleted表或inserted表

- ❑ 只读，不允许修改；触发器执行完成后，自动删除

## ❑ inserted表

- ❑ 临时保存了插入或更新后的记录行

- ❑ 可从inserted表中检查插入的数据是否满足需求

- ❑ 如不满足，向用户报告错误消息，回滚插入操作

## ❑ deleted表

- ❑ 临时保存了删除或更新前的记录行

- ❑ 可从deleted表检查被删除的数据是否满足业务需求

- ❑ 如不满足，向用户报告错误消息，并回滚插入操作



# inserted和deleted表存放的信息

修改操作	inserted表	deleted表
增加(INSERT)记录	存放新增的记录	-----
删除(DELETE)记录	-----	存放被删除的记录
修改(UPDATE)记录	存放更新后的记录	存放更新前的记录

**inserted表和deleted表存放的信息**

# 触发方式

❖ 接触发器被激活的时机：**前/后触发**和**替代触发方式**

❖ **前/后触发：**

执行修改语句→各种约束检查→执行**后触发器**。

执行**前触发器**→执行修改语句→各种约束检查。

前/后触发只能创建在表上，不能创建在视图上。

❖ **替代触发：**

引起触发器执行的修改语句停止执行，仅执行触发器，这种触发方式称为替代触发。

替代触发器可以创建在表或视图上。

# 触发方式

- ❖ 引起触发器执行的修改语句若违反了某种约束：
  - 后触发方式不会激活触发器。前触发器会被激活。
  - 替代触发器方式会激活触发器。

## 原因：

后触发必须在修改语句成功执行后才会激活触发器，当修改语句违反约束，而停止执行，所以不会激活触发器。

替代触发是用触发器的执行代替修改语句的执行。修改语句没执行，也不存在约束的检查，所以使用替代触发方式会激活触发器。

# 触发器的分类

## ❖ DML触发器

数据的更改操作引发的触发器。UPDATE、DELETE、INSERT

## ❖ DDL触发器

对数据库对象的操作引发执行的触发器。CREATE、ALTER、DROP、GRANT、DENY、REVOKE

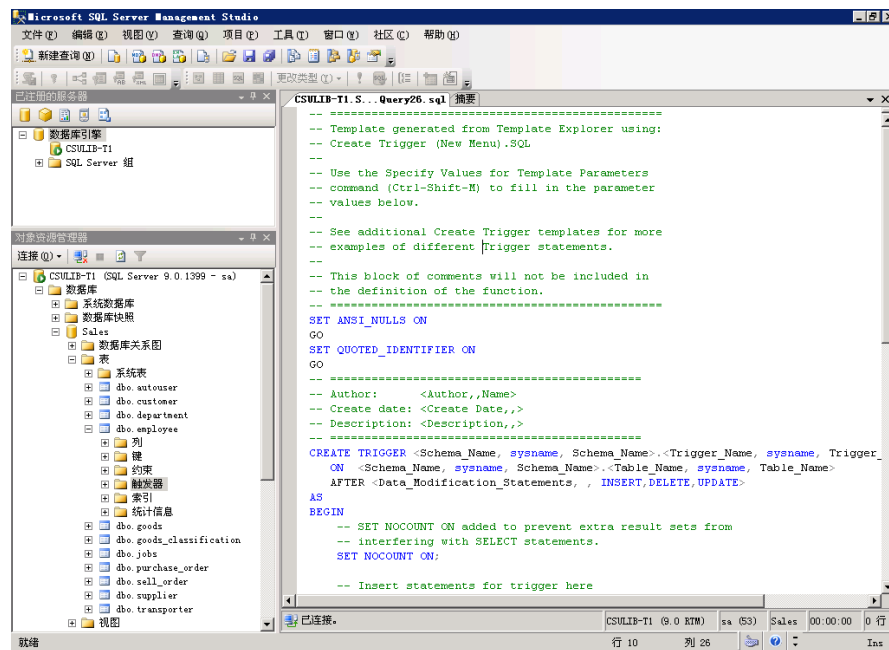
# 利用SSMS图形化界面命令创建触发器

- 启动ServerManagement Studio。
- 在“对象资源管理器中”展开要创建存储过程的“数据库”。
- 展开“数据库”的“表”节点。
- 单击要创建触发器的数据表节点；
- 右键“触发器”节点，在弹出的快捷菜单中选择“新建触发器”命令。



# 利用SSMS图形化界面命令创建触发器

- 在打开的“新建触发器”的查询界面，系统给出了创建触发器命令的模板。
- 在模板中，可以手动编辑触发器代码。
- 单击“执行”按钮即可创建触发器。



# 利用SSMS图形化界面命令创建触发器

- 右击“对象资源管理器”的“触发器”节点，在弹出的快捷菜单中选择“刷新”命令，可以看到新建立的触发器。
- 编辑触发代码时，可通过“查询”→“指定模板参数的值”，在对话框中完成参数的设置。

# 利用SQL命令创建触发器

语法格式:

Create trigger 触发器名

On 表名|视图名

对SQL语句加密

With encryption

触发器的类型，for和after等价

{for after | ~~before~~ | instead of } {delete| update |insert }

[not for replication]

在数据库复制过程中不激活  
触发器

[for each row]

As .....SQL语句

触发后执行的操作

For each row参数说明：行级触发器（否则是语句级触发器）

一个更新操作涉及多少行记录，触发器就被执行多少次。



## 示 例

【例1】对学生表更新后激活，显示共更新了多少行数据

**Create trigger tr\_stu on student**

**For after update**

**As print @rowcount**

# 示 例

【例2】对于XSCJ数据库，如果在student表中添加或更改数据，则将向客户端显示一条信息。

/\*使用带有提示消息的触发器\*/

USE XSCJ

GO

CREATE TRIGGER reminder ON student

FOR after INSERT, UPDATE

AS RAISERROR (4008, 16, 10)

GO

消息 4008 是sysmessages 中的用户定义消息。

## 示 例

【例3】创建一触发器，当向sc表插入一记录时，成绩默认为空值。

```
CREATE TRIGGER check_sc
```

```
ON sc FOR after INSERT
```

```
AS
```

```
update sc set grade=null
```

```
where sno=(select sno from inserted)
```

```
and cno=(select cno from inserted)
```

## 示 例

【例4】定义触发器实现学生表所在系属性为“计算机系”的缺省。

**Create trigger dept1**

**ON student FOR after INSERT, UPDATE**

**AS**

**update student set sdept="计算机系"**

**where sno =**

**(select sno from inserted where sdept is null)**

## 示 例

【例5】 建立一个学生与选课表间来维护参照完整性而使用的级联删除触发器、级联修改触发器和受限插入触发器。

**Create trigger ex51**

**ON student FOR after DELETE**

**AS**

**delete from sc where sno=(select sno from deleted )**

**Create trigger ex52**

**ON student FOR after update**

**AS**

**update sc set sno=(select sno from inserted )  
where sno=(select sno from deleted )**

# 示 例

受限插入触发器:

**Create trigger ex53**

**ON sc FOR after INSERT**

**AS**

**delete from sc where sno=**

**(select sno from inserted where sno not in  
(select sno from student ))**

**or cno=**

**(select cno from inserted where cno not in  
(select cno from course ))**

# 受限插入触发器

- ❖ 利用触发器来保证学生选课库中选课表的参照完整性，以维护其外码与参照表中的主码一致。

# 触发器的修改

## 1、利用SQL命令修改触发器:

**ALTER trigger 触发器名**

**On 表名|视图名**

**With encryption**

**{for after | before | instead of }{delete| update  
|insert }**

**[not for replication]**

**As .....SQL语句**



# 触发器的修改

**【例6】** 修改XSCJ数据库中在student表上定义的触发器reminder。

```
ALTER TRIGGER reminder ON student  
    FOR UPDATE  
    AS RAISERROR (“执行的操作是修改”, 16, 10)  
GO
```

## 2、通过SSMS修改触发器

进入SSMS，修改触发器的步骤与创建的步骤相同。

# 触发器的删除

## 1、利用SQL命令删除触发器

**DROP TRIGGER { trigger } [ ,...n ]**

**trigger:** 指要删除的触发器名称，包含触发器所有者名。

**n:** 表示可以指定多个触发器。

**【例7】** 删除触发器reminder。

**IF EXISTS (SELECT name FROM sysobjects**

**WHERE name = 'reminder' AND type = 'TR')**

**DROP TRIGGER reminder**

**GO**

## 2、通过SSMS修改触发器

# 作业



复习本章内容



实验

