北京林业大学

数据库原理与应用

北京林业大学 信息学院

Transact-SQL的变量、注释和运算符

北京林业大学 信息学院

本章目录

CONTENTS

Transact-SQL概述

北京林业大学 信息学院

Transact-SQL概述

Transact-SQL（T-SQL）是 SQL 在 Microsoft SQL Server 上的增强版。

T-SQL不仅提供标准 SQL 的 DDL 和 DML 功能，还提供了一些内置函数和程序设计结构(例如 IF 和 WHILE)，可以让SQL Server的程序设计具有更强的表达能力。

Transact-SQL的变量

北京林业大学 信息学院

Transact-SQL的变量

变量： 变量是可以对其赋值并参与运算的一个实体

全局变量

- 全局变量由系统定义和维护的，只能使用预先说明及定义。
- 全局变量对用户而言是只读的，用户无法对它们进行修改或管理 。

局部变量

- 用户定义的变量。
- 用户根据需要对它们进行修改或管理

Transact-SQL的变量

全局变量

- 在SQL Server中，全部变量使用两个@标记为前缀。可以使用简单的SELECT查询语句检索任意全局变量。

- SELECT @@VERSION AS SQL_SERVER_VERSION

Transact-SQL的变量

全局变量

- 在SQL Server中，一些常用的全局变量

全部变量名	含义
@@connections	服务器启动以来已经创建的连接数
@@ERROR	最后一个T-SQL错误的错误号
@@IDENTITY	最后一次插入的标识
@@SERVERNAME	本机服务器名称
@@ROWCOUNT	上一个执行的SQL语句影响行数

Transact-SQL的变量

变量声明

DECLARE @变量名 变量类型

[，@变量名 变量类型……]

变量赋值

SELECT @变量名=变量值

或

SET @变量名=变量值

Transact-SQL的变量

[例] 声明一个长度为8个字符的变量id，并赋值为'10010001'。

```
DECLARE @id char(8)
SET @id=‘10010001 ‘
```

Transact-SQL的变量

[例] 从表S中查询学号为‘S7’的学生的学号和姓名，并将查询的学号和姓名分别复制给@sno和@sn。

```
DECLARE @sno varchar(10),@sn varchar(10)
SELECT  @sno=SNO,@sn=SN  FROM  S
WHERE Sno='S7'.
```

Transact-SQL 的注释

北京林业大学 信息学院

Transact-SQL的注释

注释符

作用：使用注释进行程序的解释和说明；对暂时不需要执行的语句进行屏蔽。在Transact-SQL中可以使用两类注释符：

（1）ANSI标准的注释符“- -”用于单行注释；

（2）与C语言相同的程序注释符，即“/*……*/”，“/*”用于注释文字的开头，“*/”用于注释文字的结尾，可在程序中标识多行文字为注释。

例子：--这是一个注释
SELECT * FROM STUDENT /*注释内容*/

Transact-SQL的运算符

北京林业大学 信息学院

Transact-SQL的运算符

运算符

运算符是一种符号，用来指定要在一个或多个表达式中执行的操作。
SQL Server提供了如下几种运算符：

- 算术运算符
- 比较运算符
- 赋值运算符
- 逻辑运算符
- 一元运算符
- 字符串连接运算符
- 按位运算符

Transact-SQL的运算符

算术运算符

算术运算符对两个表达式执行数学运算，参与运算的表达式必须是数值数据类型或能够进行算术运算的其他数据类型。

运算符	含义
+	加
-	减
*	乘
/	除
%	求余数

Transact-SQL的运算符

赋值运算符

```
DECLARE @MyCounter INT
SET @MyCounter = 1
```

Transact-SQL的运算符

字符串连接运算符

加号 (+) 是字符串连接运算符，可以用它将字符串连接起来。其他所有字符串操作都使用字符串函数进行处理。

Transact-SQL的运算符

运 算 符	含　义
=	等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于
<>	不等于
!=	不等于（非 SQL-92 标准）
!<	不小于（非 SQL-92 标准）
!>	不大于（非 SQL-92 标准）

Transact-SQL的运算符

运 算 符	含 义
ALL	如果一组比较中都为TRUE，运算结果就为TRUE
AND	如果两个表达式都为TRUE，运算结果就为TRUE
ANY	如果一组的比较中任何一个为TRUE，运算结果就为TRUE
BETWEEN	如果操作数在某个范围之内，运算结果就为TRUE
EXISTS	如果子查询包含一些行，运算结果就为TRUE
IN	如果操作数等于表达式列表中的一个，运算结果就为TRUE
LIKE	如果操作数与一种模式相匹配，运算结果就为TRUE
NOT	对逻辑值取反，即如果操作数的值为TRUE，运算结果为FALSE，否则为TRUE
OR	如果两个布尔表达式中的一个为TRUE，运算结果就为TRUE
SOME	如果一系列操作数中，有些值为TRUE，运算结果为TRUE

Transact-SQL的运算符

按位运算符

按位运算符对两个表达式进行二进制位操作，这两个表达式必须是整型或者整数兼容数据类型。

Transact-SQL的运算符

运算符	含义	运算规则
&	按位与	两个数对应的二进制位上都为1时，该位上的运算结果为1，否则为0
|	按位或	两个数对应的二进制位上有一个为1时，该位上的运算结果为1，否则为0
^	按位异或	两个数对应的二进制位上不同时，该位上的运算结果为1，否则为0

00000111

& 00000100

= 0000100

Transact-SQL的运算符

一元运算符

一元运算符只对一个表达式进行运算。

运算符	含　义
+	正号，数值为正
-	负号，数值为负
~	按位取反，对操作数进行按二进制位取反运算，即二进制位上原来为1，运算结果为0，否则为1

Transact-SQL的运算符

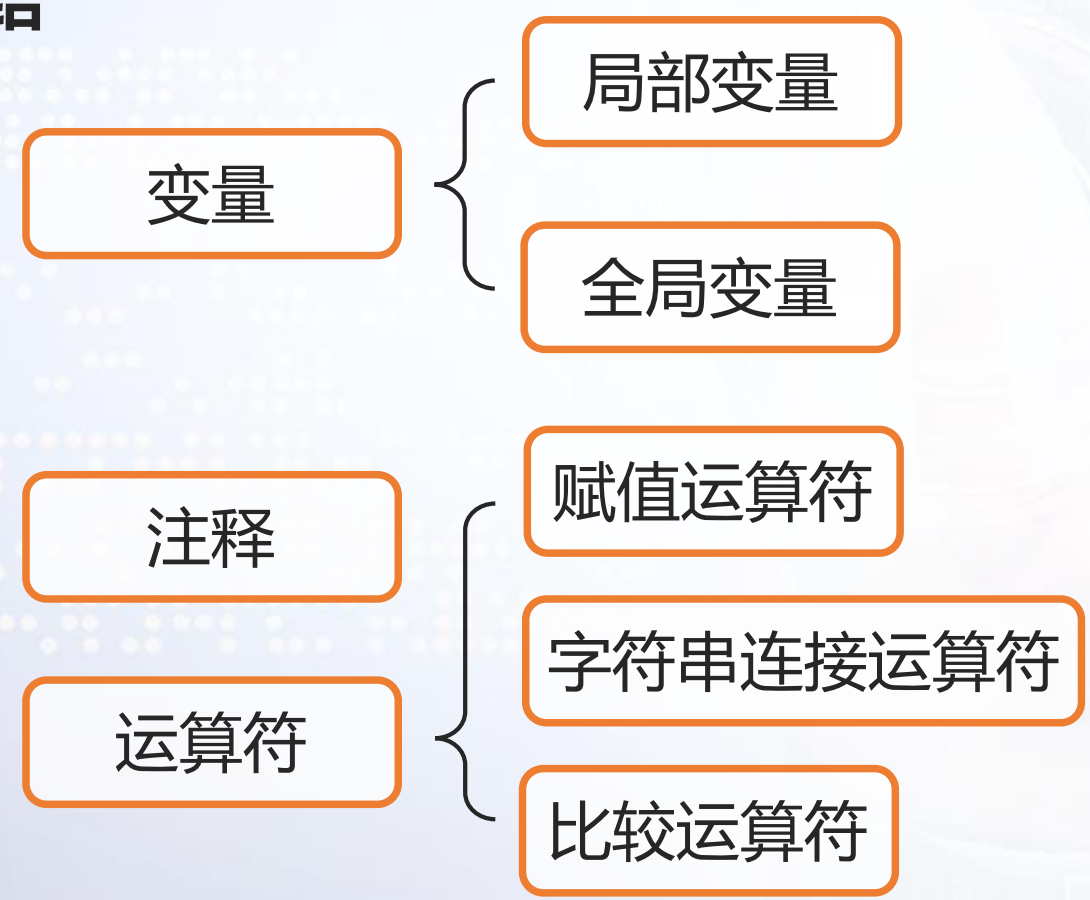
运算符优先级和结合性

优先级（从高到低）	运算符	说明
1	( )	小括号
2	+、-、~	正、负、按位取反
3	*、/、%	乘、除、求余数
4	+、-、+	加、减、字符串连接
5	=、>、<、>=、<=、<>、!=、!>、!<	各种比较运算符
6	^、&、|	位运算符
7	NOT	逻辑非
8	AND	逻辑与
9	ALL、ANY、BETWEEN、IN、LIKE、OR、SOME	逻辑运算符
10	=	赋值运算符

小结

Transact-SQL的变量、注释和运算符

小结



• • •

Transact-SQL
的批处理和流程控制

北京林业大学 信息学院

本节内容

CONTENTS

- 批处理
- 流程控制语句
- 小结

批处理

批处理

批处理是包含一个或多个T-SQL语句的组。
批处理的所有语句被整合成一个执行计划。

批处理

批处理是使用GO语句标识批处理的技术。多个批处理可以用多个GO分开，其中每两个GO之间的SQL语句就是一个批处理单元。

每个批处理被单独地处理，所以一个批处理中的错误不会阻止另一个批处理的运行。

```
USE STUDENT
SELECT * FROM STUDENT
UPDATE STUDENT SET AGE=23 WHERE
SN='1001'
GO
```

流程控制语句

北京林业大学 信息学院

流程控制语句

T-SQL使用的流程控制语句与常见的程序设计语言类似，使其能够产生控制程序执行及流程分支的作用。主要有以下几种控制语句。

- **BEGIN…END语句**
- **IF…ELSE语句**
- **CASE语句**
- **WHILE语句**
- **WAITFOR语句**
- **GOTO语句**
- **RETURN语句**

BEGIN…END语句

使用BEGIN…END语句创建一个由一条语句或多条语句构成的程序块

BEGIN…END的语法格式如下：

```
BEGIN
    <命令行或程序块>
END
```

- BEGIN…END语句经常在条件语句和循环语句中使用。
- 可以在BEGIN…END中嵌套另一个程序块。

IF…ELSE语句

使用IF…ELSE语句创建条件语句

IF…ELSE语句的语法格式如下：

IF <条件表达式>

<命令行或程序块>

[ELSE

<命令行或程序块>]

条件表达式是各种表达式的组合。

ELSE子句是可选择的。

在T-SQL中IF…ELSE可以嵌套，最多嵌套32级。

IF…ELSE语句例子

【例】从数据库Teach中的SC数据表中求出学号S1同学的平均成绩，如果此成绩或等于60分，则输出’ pass’信息，否则，输出’ fail’信息

```
USE Teach
GO
IF (SELECT AVG（SCORE）FROM SC WHERE SNO=‘S1’)>=60
   PRINT ‘pass’
ELSE
   PRINT ‘fail’
GO
```

IF [NOT] EXISTS语句

IF [NOT] EXISTS语句的语法格式如下：

IF [NOT] EXISTS (SELECT 子查询)

<命令行或程序块>

[ELSE

<命令行或程序块>]

- EXISTS后面的查询语句结果不为空，则执行其后的程序块，否则执行ELSE后面的程序块。
- 当采用NOT关键字的时候，与上面的功能相反。

IF [NOT] EXISTS例子

【例】从数据库Teach中的S表中读取学号S1同学记录，如果存在，则输出’存在记录’，否则，输出’不存在记录’。

```
USE Teach
GO
DECLARE @message VARCHAR(255)
IF EXISTS (SELECT * FROM S WHERE SNO='S1')
   SET @message='存在记录'
ELSE
   SET @message='不存在记录'
PRINT @message
GO
```

CASE语句

CASE <表达式>

WHEN <表达式> THEN <表达式>

…

WHEN <表达式> THEN <表达式>

[ELSE <表达式>]

END

当存在多种条件判断时，可以使用CASE语句，通过判断CASE后面表达式的取值，找到对应的WHEN，然后执行THEN后的表达式，执行后跳出CASE。

如果没有ELSE子句，则所有比较失败后，反馈NULL，如果存在ELSE子句，ELSE是没有任何匹配的WHEN时，执行的内容。

CASE语句例子

【例】从数据库Teach中的S表中选取SNO和Sex，如果Sex字段为′男′，则输出′M’；如果为′女′，则输出′F’。

```
USE Teach
GO
SELECT SNO, Sex=CASE sex
WHEN‘男’ THEN ‘M’
WHEN‘女’ THEN ‘F’
END
FROMS
GO
```

CASE语句可以嵌套在SQL语句中。

WHILE…CONTINUE…BREAK语句

当需要循环结构时，可以使用WHILE语句，通过WHILE后条件表达式情况，判读循环是否终止。如果循环未结束，则执行BEGIN和END标注的内容。

WHILE <条件表达式>
BEGIN
<命令行或程序块>
[BREAK]
[CONTINUE]
[命令行或程序块]
END

WHILE语句例子

【例】计算输出1-100之间能够被3整除的数的总和及个数。

```
DECLARE @s SMALLINT,@i SMALLINT,@nums SMALLINT
SET @s=0
SET @i=1
SET @nums=0
WHILE (@i<=100)
   BEGIN
     IF (@i%3=0)
       BEGIN
         SET @s=@s+@i
         SET @nums=@nums+1
       END
     SET @i=@i+1
   END
PRINT @s
PRINT @nums
```

WAITFOR语句

当程序需要阻塞一段时间的时候，可以使用WAITFOR实现。

WAITFOR {DELAY <'时间'> | TIME <'时间'>

| ERROREXIT | PROCESSEXIT |

MIRROREXIT}

WAITFOR语句

- （1）*DELAY*：用来设定等待的时间，最多可达24小时。
- （2）*TIME*：用来设定等待结束的时间点。
- （3）*ERROREXIT*：直到处理非正常中断。
- （4）*PROCESSEXIT*：直到处理正常或非正常中断。
- （5）*MIRROREXIT*：直到镜像设备失败。

WAITFOR语句例子

【例】等待1小时2分零3秒后才执行SELECT语句

```
WAITFOR DELAY '01:02:03'
SELECT * FROM S
```

【例】指定在11:24:00时间点时开始执行SELECT语句

```
WAITFOR TIME '11:24:00'
SELECT * FROM S
```

GOTO语句

将执行流程改变到由标签指定的位置。系统跳过GOTO后边的语句。

```
DECLARE @s SMALLINT,@i SMALLINT
SET @i=1
SET @s=0
BEG:
IF (@i<=10)
  BEGIN
    SET @s=@s+@i
    SET @i=@i+1
    GOTO BEG   /*使程序跳转到标号为BEG的地方执行*/
  END
PRINT @s
```

GOTO 标识符

RETURN语句

RETURN（[整数值]）

可以在任意位置使用RETURN退出，系统将不会执行RETURN后的语句。

RETURN语句不能返回NULL值。SQL Server保留−1～−99之间的返回值作为系统使用。

返回值	含义
0	程序执行成功
-1	找不到对象
-2	数据类型错误
-3	死锁错误
-4	违反权限原则
-5	语法错误
-6	用户造成的一般错误
-7	资源错误
-8	非致命的内部错误
-9	达到系统配置参数极限

小结

Transact-SQL的批处理和流程控制

小结

- 批处理过程
- 流程控制
  - BEGIN-END
  - IF、CASE
  - WHILE
  - GOTO
  - RETURN

Transact-SQL中常用命令和函数

北京林业大学 信息学院

本节内容

CONTENTS

- 常用命令
- 函数
- 小结

常用命令

常用命令

BACKUP

用于将数据库内容或其事务处理日志备份到存储介质上（软盘、硬盘、磁带等）。

常用命令

DBCC

用于验证数据库完整性、查找错误、分析系统使用情况等。

常用命令

DECLARE

DECLARE的语法格式如下：

```
DECLARE {{ @local_variable data_type }
| { @cursor_variable_name CURSOR }
| { table_type_definition }
} [, ...n]
```

常用命令

EXECUTE

EXECUTE或EXEC命令用来执行存储过程 。

KILL

KILL命令用于终止某一过程的执行。

常用命令

PRINT

PRINT的语法格式如下：

PRINT 'any ASCII text' | @local_variable | @@FUNCTION | string_expression

PRINT 命令向客户端返回一个用户自定义的信息，即显示一个字符串、局部变量或全局变量。

常用命令

RAISERROR

用于在SQL Server 系统返回错误信息时，同时返回用户指定的信息。

常用命令

SELECT

SELECT 命令可用于给变量赋值，其语法格式如下：

SELECT {@local_variable = expression }
[，...n]

SELECT 命令可以一次给多个变量赋值。

常用命令

SET

命令有两种用法。
用于给局部变量赋值。
用于用户执行SQL 命令时，SQL Server 处理选项的设定 。
SET：选项ON；
SET：选项OFF；
SET：选项值。

常用命令

SHUTDOWN

SHUTDOWN [WITH NOWAIT]
SHUTDOWN 命令用于停止SQL Server 的执行。

常用命令

USE

USE {database}
USE命令用于改变当前使用的数据库为指定的数据库。

函数

- 函数是能够完成特定功能并返回处理结果的一组T-SQL语句，处理结果称为“返回值”，处理过程称为“函数体”。
- 函数可以用来构造表达式，可以出现在SELECT语句的选择列表中，也可以出现在WHERE子句的条件中。
- SQL Server的内置函数包括：

统计函数

算术函数

字符串函数

日期函数

自定义函数

统计函数

STDEV函数

STDEV函数返回表达式中所有数据的标准差。

STDEVP函数

STDEVP 函数返回表达式中所有数据的总体标准差。

算术函数

可以在SELECT语句的SELECT和WHERE子句以及表达式中使用。

函　数	功　能
三角函数	
SIN	返回以弧度表示的角的正弦
COS	返回以弧度表示的角的余弦
TAN	返回以弧度表示的角的正切
COT	返回以弧度表示的角的余切
角度弧度转换	
DEGREES	把弧度转换为角度
RADIANS	把角度转换为弧度

算术函数

指数函数	EXP(表达式)	返回以e为底、以表达式为指数的幂值
对数函数	LOG(表达式) LOG10(表达式)	返回表达式的以e为底的自然对数值 返回表达式的以10 为底的对数值
平方根函数	SQRT(表达式)	返回表达式的平方根
取近似值函数	CEILING(表达式) FLOOR(表达式) ROUND(表达式,n)	返回大于等于表达式的最小整数 返回小于等于表达式的最大整数 将表达式四舍五入为指定的精度n
符号函数	ABS(表达式) SIGN(表达式)	返回表达式的绝对值 测试表达式的正负号，返0、1或 - 1
其他函数	PI( ) RAND( )	返回值为π，即3.1415926535897936 返回0 ~ 1之间的随机浮点数

字符串函数

- 字符串转换函数

ASCII（character_expression）

返回字符表达式最左端字符的ASCII 码值

CHAR （integer_expression）

CHAR函数用于将ASCII 码转换为字符

LOWER （character _expression）

LOWER函数用于把字符串全部转换为小写

字符串函数

- 字符串转换函数

UPPER （character _expression）

UPPER函数用于把字符串全部转换为大写

STR（float _expression [， length[， <decimal>]]）

STR函数用于把数值型数据转换为字符型数据

字符串函数

- 去空格函数

LTRIM （character _expression）

LTRIM函数用于把字符串头部的空格去掉。

RTRIM （character _expression）

RTRIM函数用于把字符串尾部的空格去掉 。

字符串函数

◆ 取子串函数

LEFT（character_expression, integer_expression）

LEFT函数返回的子串是从字符串最左边起到第integer_expression 个字符的部分。

RIGHT（character_expression, integer_expression）

RIGHT函数返回的子串是从字符串右边第integer_expression 个字符起到最后一个字符的部分。

字符串函数

- 字符串比较函数

CHARINDEX （substring_expression，expression）

CHARINDEX函数返回字符串中某个指定的子串出现的开始位置。

日期函数

DAY（<date_expression>）

DAY函数返回date_expression 中的日期值。

MONTH（<date_expression>）

MONTH函数返回date_expression中的月份值。

日期函数

YEAR（<date_expression>）

YEAR函数返回date_expression 中的年份值。

DATEADD（<datepart> <number> <date>）

DATEADD函数返回指定日期date加上指定的额外日期间隔number产生的新日期。

日期函数

DATEDIFF（<datepart>，<date1>，<date2>）

DATEDIFF函数返回两个指定日期在datepart方面的不同之处，即date2超过date1的差距值，其结果值是一个带有正负号的整数值。

DATENAME（<datepart>，<date>）

DATENAME函数以字符串的形式返回日期的指定部分，此部分由datepart 来指定。

用户自定义函数

- （1）创建标量值函数
- （2）创建内联表值函数
- （3）多语句表值函数

数值函数返回结果为单个数据值；表值函数返回结果集（table数据类型）

（1）创建标量值函数

（1）创建标量值函数

标量值函数的函数体由一条或多条T-SQL语句组成，这些语句以BEGIN开始，以END结束

```
CREATE  FUNCTION  function_name
([ { @parameter_name [ As ] parameter_data_type
[ = default ] [ READONLY ] }
  [ ,...n ]
  ]
)
RETURNS return_data_type
[ WITH ENCRYPTION ]
[ AS ]
BEGIN
function_body
Return scalar_expression
END
```

创建标量值函数的例子

【例】

自定义一个标量函数Fun1，判断一个整数是否为素数，如果为素数，则函数返回1，否则返回0，待判断的数通过参数传给函数。

创建标量值函数的例子

```
CREATE FUNCTION dbo.Fun1(@n AS INT)
RETURNS INT
AS
BEGIN
   DECLARE @i INT
   DECLARE @sign INT
   SET @sign=1
   SET @i=2
   WHILE  @i<=SQRT(@n)
     BEGIN
       IF @n % @i=0
         BEGIN
            SET @sign=0
            BREAK
         END
        SET @i=@i+1
     END
   RETURN @sign
END
```

调用：SELECT dbo.Fun1(13)

（2）创建内联表值函数

（2）创建内联表值函数

```
CREATE  FUNCTION  function_name
( [ { @parameter_name [ AS ]
parameter_data_type  [ = default ]
[ READONLY ] }
    [ ,...n ]
    ]
)
RETURNS TABLE
[ WITH ENCRYPTION ]
[ AS ]
RETURN （select_statement）
```

（2）创建内联表值函数

内联表值函数没有函数体

```
CREATE FUNCTION dbo.Fun2()
RETURNS TABLE
AS
RETURN SELECT SNo,SN FROM S
```

调用：SELECT * FROM Score_Table('S2')

用户自定义函数的步骤

与内联表值函数不同的是，多语句表值函数在返回语句之前还有其他的Transact-SQL语句

```
CREATE  FUNCTION  function_name
( [ { @parameter_name [ AS ] parameter_data_type
[ = default ] [ READONLY ] }
    [ ,...n ]
    ]
)
RETURNS @return_variable Table
<table_type_definition>
[ WITH ENCRYPTION ]
[ AS ]
BEGIN
function_body
RETURN
END
```

（3）多语句表值函数

```
CREATE FUNCTION Score_Table
(@student_id CHAR(6))
RETURNS @T_score TABLE
(Cname VARCHAR(20),
Grade INT
)
AS
BEGIN
   INSERT INTO @T_score
   SELECT CN,Score
   FROM SC,C
   WHERE  SC.CNo=C.CNo and SC.SNo=@student_id and
Score<60
  RETURN
END
```

调用：SELECT * FROM Score_Table('S2')

小结

小结

- 常用命令
- 函数
  - 常用函数
  - 自定义函数

北京林业大学

数据库原理与应用

北京林业大学 信息学院

存储过程

本节目录

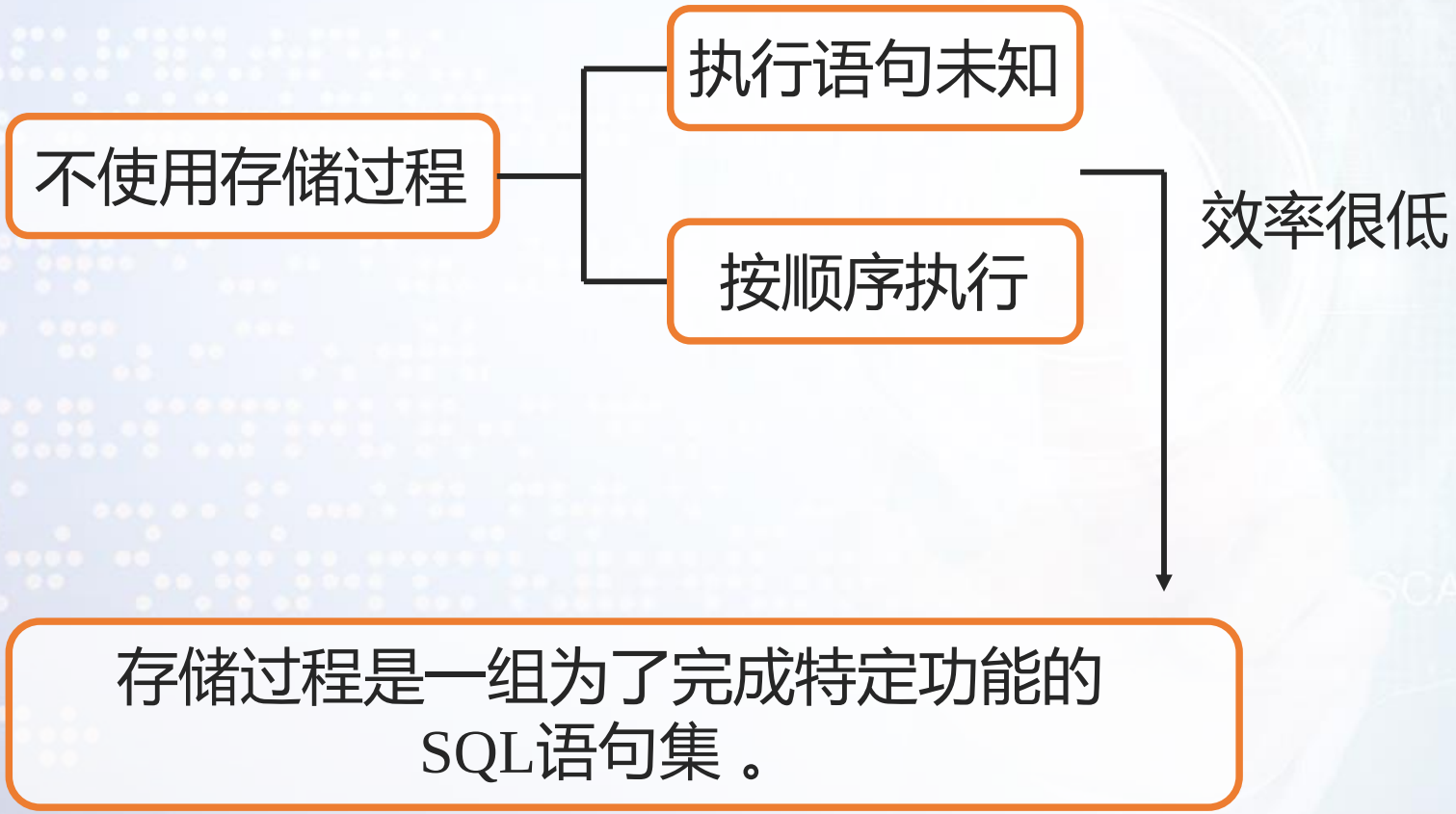
CONTENTS

- 存储过程的概念、优点及分类
- 创建、查看、重命名、删除、执行、修改存储过程

存储过程的概念、优点及分类

北京林业大学 信息学院

存储过程的概念



存储过程的优点

存储过程的优点：

- 模块化的程序设计，独立修改。
- 高效率的执行，一次编译。
- 减少网络流量 。
- 可以作为安全机制使用。

存储过程的分类

存储过程的分类：

- 系统存储过程，master数据库，sp前缀。
- 用户自定义存储过程。
- 扩展存储过程，xp前缀。

创建、查看、重命名、删除、执行、修改存储过程

创建存储过程

当创建存储过程时，需要确定存储过程的三个组成部分：

- 所有的输入参数以及传给调用者的输出参数。
- 被执行的针对数据库的操作语句，包括调用其他存储过程的语句。
- 返回给调用者的状态值以指明调用是成功还是失败。

用CREATE PROCEDURE命令创建存储过程

创建存储过程

```
CREATE PROCEDURE procedure_name [ ; number ]
[ { @parameter data_type }
[ VARYING ] [ = default ] [ OUTPUT ]] [ ,...n ]
[ WITH
{ RECOMPILE | ENCRYPTION | RECOMPILE ,
ENCRYPTION } ]
[ FOR REPLICATION ]
AS sql_statement [,...n ]
```

创建存储过程

【例】 在Teach数据库中，创建一个名称为MyProc的不带参数的存储过程，该存储过程的功能是从数据表S中查询所有男同学的信息。

```
USE Teach
GO
CREATE PROCEDURE MyProc AS
SELECT * FROM S WHERE Sex='男'
```

创建存储过程

【例】　定义具有参数的存储过程。在Teach数据库中，创建一个名称为InsertRecord的存储过程，该存储过程的功能是向S数据表中插入一条记录，新记录的值由参数提供。

```
USE Teach
GO
CREATE PROCEDURE InsertRecord
(   @sno VARCHAR(6),
    @sn NVARCHAR(10),
    @sex NCHAR(1),
    @age INT,
    @dept NVARCHAR(20)
)
AS
INSERT INTO S VALUES(@sno,@sn,@sex,@age,
@dept)
```

创建存储过程

【例】 定义具有参数默认值的存储过程。在Teach数据库中，创建一个名称为InsertRecordDefa的存储过程，该存储过程的功能是向S数据表中插入一条记录，新记录的值由参数提供，如果未提供系别Dept的值时，由参数的默认值代替。

创建存储过程

```
USE Teach
GO
CREATE PROCEDURE InsertRecordDefa
(   @sno VARCHAR(6),
    @sn NVARCHAR(10),
    @sex NCHAR(1),
    @age INT,
    @dept NVARCHAR(20)= '无')
AS
INSERT INTO S VALUES(@sno, @sn, @sex, @age, @dept)
```

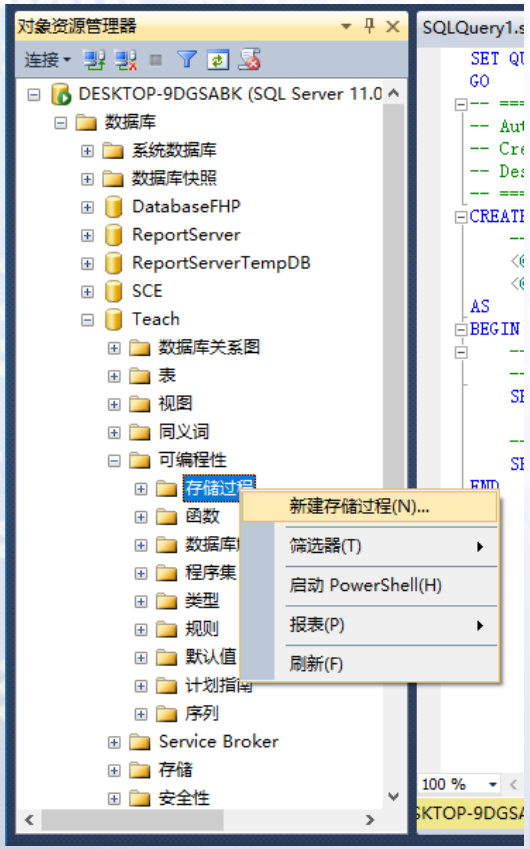
创建存储过程

【例】 定义能够返回值的存储过程。在Teach数据库中，创建一个名称为QueryTeach的存储过程。该存储过程的功能是从数据表S中根据学号查询某一同学的姓名和系别，查询的结果由参数@sn和@dept返回。

```
USE Teach
GO
CREATE PROCEDURE QueryTeach
(   @sno VARCHAR(6),
    @sn NVARCHAR(10) OUTPUT,
    @dept NVARCHAR(20) OUTPUT)
AS
SELECT @sn=SN,@dept=Dept
FROM S
WHERE SNo=@sno
```

创建存储过程

利用对象资源管理器创建存储过程



◆ 在选定的数据库下打开“可编程性”节点。

◆ 找到“存储过程”节点，单击鼠标右键，在弹出的快捷菜单中选择“新建存储过程”。

◆ 在新建的查询窗口中可以看到关于创建存储过程的语句模板，在其中添上相应的内容，单击工具栏上的“执行”按钮即可。

查看存储过程

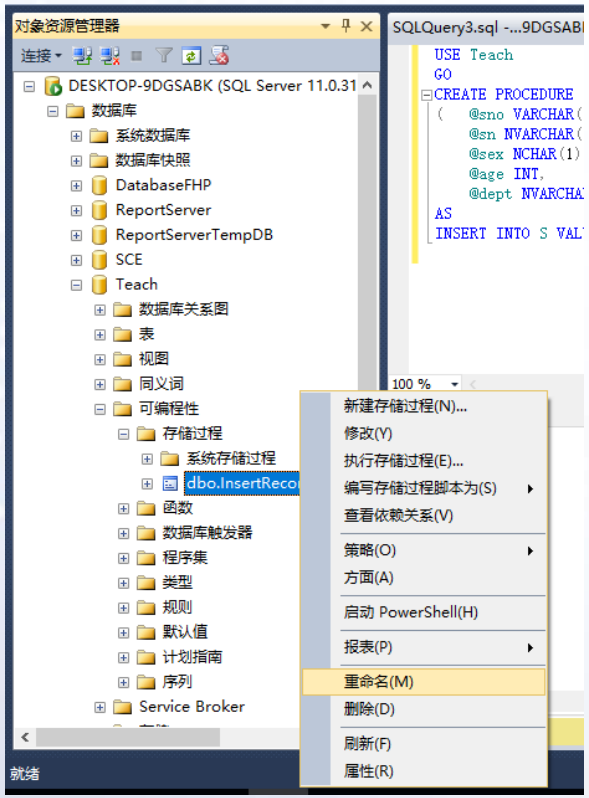
sp_helptext 存储过程名称

【例】 查看数据库Teach中存储过程MyProc的源代码。

```
USE Teach
GO
EXEC sp_helptext MyProc
```

查看存储过程

- 在选定的数据库下打开“可编程性”节点。
- 找到“存储过程”节点，展开。

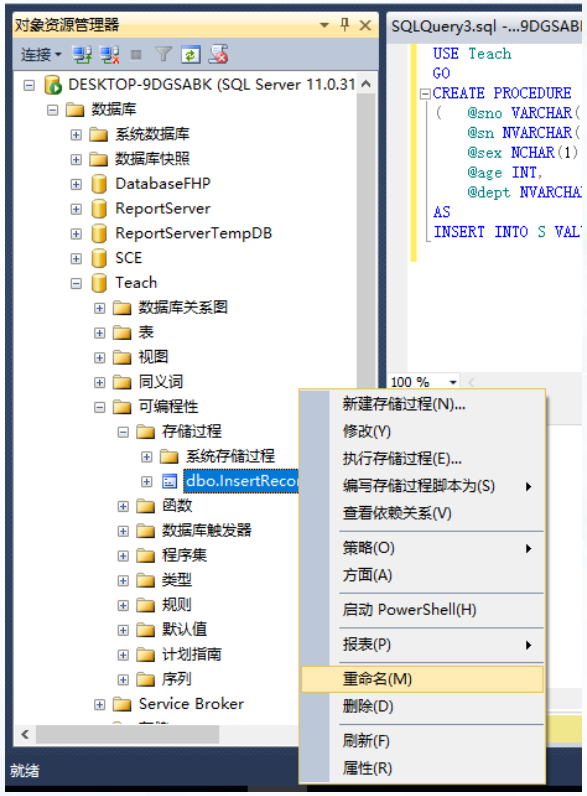


重新命名存储过程

通过对象资源管理器

◆ 在选定的数据库下打开“可编程性”节点。

◆ 找到“存储过程”节点，展开。右键单击要重命名的存储过程名称，在弹出的快捷菜单中选择“重命名”。



删除存储过程

DROP PROCEDURE {procedure} [,…n]

利用对象资源管理器删除存储过程

- 在选定的数据库下打开“可编程性”节点。
- 找到“存储过程”节点，展开。右键单击要删除的存储过程名称，在弹出的快捷菜单中选择“删除”。

执行存储过程

EXEC MyProc

【例】 执行数据库Teach中已定义的不带参数的存储过程MyProc。

```
USE Teach
GO
EXEC MyProc
```

修改存储过程

ALTER PROCEDURE
procedure_name [;number]

利用对象资源管理器修改存储过程

- 在选定的数据库下打开“可编程性”节点。
- 找到“存储过程”节点，展开。右键单击要修改的存储过程名称，在弹出的快捷菜单中选择“修改”。
- 对存储过程代码进行修改，单击工具栏“执行”按钮，即可完成。

北京林业大学

数据库原理与应用

北京林业大学 信息学院

触发器

触发器概述

触发器是一种特殊类型的存储过程。

触发器主要有以下优点：

- 触发器是在某个事件发生时自动激活而执行的。
- 触发器可以实现比约束更为复杂的完整性要求。
- 触发器可以根据表数据修改前后的状态采取相应的措施。
- 触发器可以防止恶意的或错误的INSERT、UPDATE和DELETE操作。

触发器的种类

DML触发器

DML触发器是在执行数据操纵语言（DML）事件时被激活而自动执行的触发器。

DDL触发器

DDL触发器是在执行数据定义语言（DDL）事件时被激活而自动执行的触发器。

登录触发器

登录触发器是由登录（LOGON）事件而激活的触发器。

触发器的工作原理

SQL Server在工作时为每个触发器在服务器的内存上建立两个特殊的表：插入表和删除表。

对表的操作	Inserted表	Deleted表
增加记录（INSERT）	存放增加的记录	无
删除记录（DELETE）	无	存放被删除的记录
修改记录（UPDATE）	存放更新后的记录	存放更新前的记录

触发器的工作原理

（1）INSERT触发器的工作原理

当对表进行INSERT操作时，INSERT触发器被激发，新的数据行被添加到创建触发器的表和Inserted表。

触发器的工作原理

（2）DELETE触发器的工作原理

对表进行DELETE操作时，DELETE触发器被激发，系统从被影响的表中将删除的行放入Deleted表中。

触发器的工作原理

（3）UPDATE触发器的工作原理

当执行UPDATE操作时，UPDATE触发器被激活。触发器将原始行移入Deleted表中，把更新行插入到Inserted表中。

创建触发器

创建DML触发器

使用CREATE TRIGGER创建DML触发器的语法格式为：

```
CREATE TRIGGER trigger_name
ON {table | view}
[With Encryption]
{For | After | Instead Of}
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE] }
AS sql_statement [;]
```

创建触发器

创建DML触发器

【例】设计一个触发器，在学生表S中删除某一个学生时，在选课表SC中该学生的选课记录也全部被删除。

```
USE Teach
GO
CREATE TRIGGER del_S
ON S
AFTER DELETE
AS
        DELETE FROM SC
        WHERE SC.SNo
        IN (SELECT SNo FROM DELETED)
GO
```

创建触发器

创建DDL触发器

创建DDL触发器的CREATE TRIGGER语句的语法格式为：

CREATE TRIGGER trigger_name

On {All Server | Database}

[With Encryption]

{FOR | AFTER} {event_type | event_group } [,...n]

AS sql_statement [;]

创建触发器

创建DDL触发器

【例】创建一个DDL触发器safety，禁止修改和删除当前数据库中的任何表。

```
USE Teach
GO
CREATE TRIGGER safety
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS  PRINT '不能删除或修改数据库表！'
ROLLBACK
GO
```

查看触发器

查看表中触发器

执行系统存储过程sp_helptrigger查看表中触发器的语法格式如下：

EXEC sp_helptrigger 'table'[,'type']

查看触发器

查看触发器的定义文本

利用系统存储过程sp_helptext可查看某个触发器的内容，语法格式为：

```
EXEC sp_helptext 'trigger_name'
```

查看触发器

查看触发器的所有者和创建时间

系统存储过程sp_help可用于查看触发器的所有者和创建日期，语法格式如下：

```
EXEC sp_help 'trigger_name'
```

修改触发器

利用ALTER TRIGGER语句修改触发器

（1）修改DML触发器的语法格式如下：

```
ALTER TRIGGER schema_name.trigger_name
ON ( table | view )
[ WITH ENCRYPTION ]
{ FOR | AFTER | INSTEAD OF }
{ [ DELETE ] [ , ] [ INSERT ] [ , ] [ UPDATE] }
AS  sql_statement [ ; ]
```

修改触发器

利用ALTER TRIGGER语句修改触发器

（2）修改DDL触发器的语法格式如下：

```
ALTER TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH ENCRYPTION ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS  sql_statement  [ ; ]
```

修改触发器

使触发器无效

在有些情况下，用户希望暂停触发器的作用，但并不删除它，这时就可以通过DISABLE TRIGGER语句使触发器无效，语法格式如下：

```
DISABLE TRIGGER { [ schema.] trigger_name [ ,...n ]
| ALL }
ON object_name
```

修改触发器

使触发器重新有效

要使触发器重新有效，可使用ENABLE TRIGGER语句，语法格式如下：

```
ENABLE TRIGGER {[schema_name.] trigger_name
[ ,...n ] | ALL }
ON object name
```

删除触发器

使用DROP TRIGGER语句删除触发器

删除DML触发器的DROP TRIGGER语句的语法格式为：

DROP TRIGGER trigger_name [,...n] [;]

北京林业大学

数据库原理与应用

北京林业大学 信息学院

备份和还原

备份和还原概述

- 备份是对SQL Server数据库或事务日志进行复制，数据库备份记录了在进行备份这一操作时数据库中所有数据的状态，如果数据库因意外而损坏，这些备份文件可以用来还原数据库。
- 还原就是把遭受破坏、丢失的数据或出现错误的数据库恢复到原来的正常状态。

数据库备份的类型

数据库完整备份	是指对数据库内的所有对象都进行备份 。
事务日志备份	只备份数据库的事务日志内容。
差异备份	差异备份是完整备份的补充，只备份自从上次数据库完整备份后数据库变动的部分。
文件和文件组备份	是针对单一数据库文件或者是文件组做备份。

备份和还原的策略

SQL Server 提供了几种方法来减少备份或还原操作的执行时间。

- 使用多个备份设备来同时进行备份处理。
- 综合使用完整数据库备份、差异备份或事务日志备份来减少每次需要备份的数据量。
- 使用文件或文件组备份以及事务日志备份，这样可以只备份或还原那些包含相关数据的文件，而不是整个数据库。

在SQL Server 中的数据库还原模式：

简单还原　完全还原

创建备份设备

使用系统存储过程sp_addumpdevice创建备份设备

```
sp_addumpdevice  [ @devtype = ] 'device_type' ,
[ @logicalname = ] 'logical_name' ,
[ @physicalname = ] 'physical_name'
```

【例】为数据库Teach创建一个磁盘备份设备。

```
USE Teach
GO
EXEC sp_addumpdevice
'disk','pubss','c:\backdev\backdevpubs.bak'
```

创建备份设备

使用sp_dropdevice来删除备份设备

```
sp_dropdevice [@logicalname =] 'device' [,
[@delfile =] 'delfile']
```

【例】删除数据库Teach中创建的备份设备pubss。

```
USE Teach
GO
EXEC sp_dropdevice 'pubss', ' c:\backdev\backdevpubs.bak'
```

备份数据库

（1）在“对象资源管理器”中，找到“数据库”节点展开，选择要备份的数据库，单击鼠标右键，在弹出的快捷菜单中选择“任务”→“备份”命令，出现“备份数据库”对话框。

（2）在“备份数据库”对话框中，对“源”、“备份集”和“目标”中的内容进行设置。

（3）设置完成之后，单击“确定”按钮，即可备份成功。

还原数据库

（1）在“对象资源管理器”中，用鼠标右键单击“数据库”节点，在弹出的快捷菜单中选择“还原数据库”命令，打开“还原数据库”对话框。

（2）在“源”选项对应的“设备”栏右侧，单击“浏览”按钮，打开“选择备份设备”对话框。在“备份介质”列表框中，从列出的设备类型中选择一种，或者单击“添加”按钮可以将一个或多个备份设备添加到“备份位置”列表框中，单击“确定”按钮返回到“还原数据库”对话框。

还原数据库

（3）在“还原数据库”对话框中，在“目标”对应的“数据库”列表中，输入目标数据库的名称。

（4）如果要查看或选择高级选项，可以单击“选择页”中的“选项”，将切换到“选项”选项卡进行有关设置。

（5）以上的设置完成之后，单击“确定”按钮，系统将按照所选的设置对数据库进行还原，如果没有发生错误，将出现还原成功的对话框。