数据库系统概论

An Introduction to Database System

第八章 数据库编程

第八章 数据库编程

8.1 嵌入式SQL

8.2 存储过程

8.3 ODBC编程

8.1 嵌 入 式 SQL

8.1.1 嵌入式SQL的处理过程

8.1.2 嵌入式SQL语句与主语言之间的通信（略）

8.1.3 游标的概念和特点

8.1.4 使用游标处理数据

8.1.5 小结

8.1 嵌入式SQL

- SQL语言提供了两种不同的使用方式：
  - 交互式
  - 嵌入式
  - 嵌入式SQL是将SQL语句嵌入到程序设计语言中，被嵌入的程序设计语言，如C，C++，Java称为宿主语言或主语言。
- 为什么要引入嵌入式SQL
  - SQL语言是非过程性语言
  - 事务处理应用需要高级语言
- 这两种方式细节上有差别，在程序设计的环节下，SQL语句要做某些必要的扩充。

8.1.1 嵌入式SQL的处理过程

- 为了区分SQL语句与主语言语句，需要:
  - 前缀：EXEC SQL
  - 结束标志：随主语言的不同而不同
- 以C为主语言的嵌入式SQL语句的一般形式

  EXEC SQL <SQL语句>;

  例：EXEC SQL DROP TABLE Student;
- 以COBOL作为主语言的嵌入式SQL语句的一般形式

  EXEC SQL <SQL语句> END-EXEC

  例： **EXEC SQL DROP TABLE Student END-EXEC**

DBMS处理宿主型数据库语言SQL 的方法

- 预编译
- 修改和扩充主语言使之能处理**SQL**语句

预编译

1. 由DBMS的预处理程序对源程序进行扫描，识别出SQL语句

2. 把它们转换成主语言调用语句，以使主语言编译程序能识别它

3. 最后由主语言的编译程序将整个源程序编译成目标码。

8.1.2 嵌入式SQL语句与主语言之间的通信

将SQL嵌入到高级语言中混合编程，程序中会含有两种不同计算模型的语句

- SQL语句
  - 描述性的面向集合的语句
  - 负责操纵数据库
- 高级语言语句
  - 过程性的面向记录的语句
  - 负责控制程序流程

工作单元之间的通信方式

1. SQL通信区

向主语言传递SQL语句的执行状态信息

主语言能够据此控制程序流程

2. 主变量

1）主语言向SQL语句提供参数

2）将SQL语句查询数据库的结果交主语言进一步处理

3. 游标

解决集合性操作语言与过程性操作语言的不匹配

1. SQL通信区

- **SQLCA： SQL Communication Area**
  - **SQLCA是一个数据结构**
- **SQLCA的用途**
  - **SQL语句执行后，DBMS反馈给应用程序信息**
    - 描述系统当前工作状态
    - 描述运行环境
  - **这些信息将送到SQL通信区SQLCA中**
  - **应用程序从SQLCA中取出这些状态信息，据此决定接下来执行的语句**

SQLCA的内容

- 与所执行的SQL语句有关
- 与该SQL语句的执行情况有关

例：在执行删除语句DELETE后，不同的执行情况，SQLCA中有不同的信息：

- 违反数据保护规则，操作拒绝
- 没有满足条件的行，一行也没有删除
- 成功删除，并有删除的行数
- 无条件删除警告信息
- 由于各种原因，执行出错

3. 游标（cursor）

- 为什么要使用游标
  - SQL语言与主语言具有不同数据处理方式
- SQL语言是面向集合的，一条SQL语句原则上可以产生或处理多条记录
- 主语言是面向记录的，一组主变量一次只能存放一条记录
  - 仅使用主变量并不能完全满足SQL语句向应用程序输出数据的要求
  - 嵌入式SQL引入了游标的概念，用来协调这两种不同的处理方式

8.1.3 游标的概念和特点

- 游标的定义
  - 游标是系统为用户开设的一个数据缓冲区，存放SQL语句的执行结果
  - 用户可以用SQL语句逐一从游标中获取记录，并赋给主变量，交由主语言进一步处理

总的来说，游标是允许用户能够从select语句查询的结果集中，逐条逐行地访问记录，可以逐行地显示、修改或删除这些记录的数据访问处理机制。游标可以理解为数据表记录逐行访问（移动当前记录和在当前记录上进行访问）的位置指针。

8.1.3 游标的概念和特点

- 游标的优点

✓允许程序对由查询语句SELECT返回的结果集中的每一行记录执行相同或不同的操作，而不是对整个结果集执行同一操作。

✓允许定位结果集的特定行

✓从结果集的当前位置检索一行或多行

✓支持对结果集中当前位置的行进行数据修改

✓游标实际上是作为面向集合的数据库管理系统和面向行的程序设计之间的桥梁，使这两种处理方式通过游标沟通起来

8.1.4 使用游标处理数据

- 使用游标的步骤
- （1）声明游标：Declare 游标名 cursor for select 语句
- （2）打开游标：Open 游标名
- （3）处理数据：

✓ 移动到当前行并读取数据：

Fetch 游标名 [into @变量名]

✓ 删除当前行数据

Delete from 表或视图名 where current of 游标名

✓ 修改当前行数据

Update 表或视图名　　Set 列名=表达式

Where current of 游标名

- （4）关闭游标： Close 游标名
- （5）释放游标：Deallocate 游标名

8.1.4 使用游标处理数据

- 游标的类型
- **Static（静态）：** 静态游标的完整结果集在游标打开时建立在tempdb系统数据库中，一旦打开，就不再变化。静态游标只能是只读的。当一个用户正在逐条访问查询结果时，如果其他用户正使用同一个数据表修改记录，那么该用户将不会看到该修改，所看到的数据记录是他运行open语句时的记录。
- **Dynamic（动态）：** 能反映对结果集中所做的修改。在接收到查询结果之后，记录会不断被更新，以便能够实时看到最新记录信息。该种游标最灵活，但需要较大的系统开销和资源。
- **Forward only（只进）：** 只能前进，只支持游标从前向后一条一条移动记录指针来访问记录。
- **Scroll（滚动）：** 允许向前或向后一条或多条滚动记录指针来访问记录。

8.1.4 使用游标处理数据

- 声明游标
- 格式：

DECLARE 游标名 CURSOR

[FORWARD_ONLY|SCROLL]

[STATIC|DYNAMIC]

FOR SELECT 语句

[FOR UPDATE [OF 列名[,…N]]]

例： **declare db_cursor4 scroll cursor**

for

select * from S

8.1.4 使用游标处理数据

打开游标

- 格式：

OPEN [GLOBAL] 游标

当游标被打开时，行指针会指在第一行之前。

打开游标时，如果**@@error=0**，表示游标打开操作成功。

打开游标后，可用**@@cursor_rows**返回游标记录数

例： **open db_cursor4**

8.1.4 使用游标处理数据

- [例]用游标函数查询记录数

```
Declare c_学生游标 cursor
For
Select * from 学生
Open c_学生游标
If @@error = 0
Print '学生总数：'+convert(char(5),@@cursor_rows)
Close c_学生游标
Deallocate c_学生游标
```

运行结果：

学生总数：9

8.1.4 使用游标处理数据

- 读取游标中的数据

FETCH [NEXT|PRIOR|FIRST|LAST|ABSOLUTE N|RELATIVE N]

FROM 游标名 [INTO @变量名[,…N]]

其中：

First：移动到第一行并将其作为当前行

Next：移动到下一行并将其作为当前行

Prior：移动到上一行并将其作为当前行

Last：移动到最后一行并将其作为当前行

Absolute n：

若n>0，移动从第一行开始到正数第n行，并将其作为当前行。

若n<0，移动从最后一行开始到倒数第n行，并将其作为当前行。

Relative n：

若n>0，移动从当前行开始到正数第n行，并将其作为当前行。

若n<0，移动从当前行开始到倒数第n行，并将其作为当前行。

8.1.4 使用游标处理数据

- [例]

```
Declare db_cursor4 scroll cursor
For
Select * from 药品
Open db_cursor4
Fetch first from db_cursor4
Fetch next from db_cursor4
Fetch last from db_cursor4
Fetch prior from db_cursor4
Fetch absolute 2 from db_cursor4
Fetch relative 2 from db_cursor4
Close db_cursor4
Deallocate db_cursor4
```

8.1.4 使用游标处理数据

- 可以用@@fetch_status返回执行的fetch操作后，当前游标指针的状态：0表示行已成功读取；-1表示读取操作已经操作了结果集；-2表示行在表中不存在。

```
Declare c scroll cursor
For
Select * from S
Open c
Fetch next from c
While @@fetch_status = 0
    Fetch next from c
Close c
Deallocate c
```

8.1.4 使用游标处理数据

- **利用变量输出游标中的字段值:**可以将查询到的数据写入局部变量，但必须先声明该局部变量的类型和宽度，并且必须与对应**select**语句中指定的列的类型和宽度相同。
- **Declare db_cursor5 scroll cursor**
- **For**
- **Select 职工号,姓名,工资 from 职工**
- **Declare @t varchar(10)**
- **Declare @t1 varchar(10)**
- **Declare @t2 int**
- **Open db_cursor5**
- **Fetch absolute 4 from db_cursor5 into @t,@t1,@t2**
- **Select @t as 职工号,@t1 as 职工名,@t2 as 工资**
- **Close db_cursor5**
- **Deallocate db_cursor5**

8.1.4 使用游标处理数据

- 利用游标修改数据表中的数据记录

```
Declare db_cursor6 scroll cursor
For
Select * from 职工备份
Open db_cursor6
Fetch first from db_cursor6
Update 职工备份 set 仓库号='modifybycursor'
 where current of db_cursor6
Close db_cursor6
Deallocate db_cursor6
```

8.1.4 使用游标处理数据

- 利用游标删除数据表中的数据记录

```
Declare db_cursor7 scroll cursor
For
Select * from 职工备份
Open db_cursor7
Fetch absolute 3 from db_cursor6
Delete from 职工备份 where current of db_cursor7
Close db_cursor7
Deallocate db_cursor7
```

8.1.4 使用游标处理数据

- 利用游标判断记录是否存在

```
Declare db_cursor8 scroll cursor
For
Select * from 职工 where 职工号='007'
Open db_cursor8
If @@fetch_status = 0
  print '存在该记录'
Else
 print '不存在该记录'
Close db_cursor8
Deallocate db_cursor8
```

```
DECLARE @XH char(9),@JSH Char(6)
DECLARE @KCH char(4),@CJ tinyint
DECLARE    Scorecur CURSOR
    LOCAL SCROLL
    FOR
       SELECT *       FROM SelectCourse      WHERE Score<60
    FOR UPDATE
OPEN Scorecur
FETCH NEXT FROM Scorecur
    INTO @XH,@JSH,@KCH,@CJ
WHILE @@FETCH_STATUS=0
    BEGIN
       DELETE FROM SelectCourse
       WHERE CURRENT OF    Scorecur
       FETCH NEXT FROM    Scorecur
          INTO @XH,@JSH,@KCH,@CJ
    END
CLOSE    Scorecur
DEALLOCATE    Scorecur
SELECT      count(*)
     FROM SelectCourse       WHERE Score<60
GO
```

结果：0

```
DECLARE @XH char(9),@JSH Char(6)
DECLARE @KCH char(4),@CJ tinyint
DECLARE    Scorecur CURSOR
    LOCAL SCROLL
    FOR
       SELECT *       FROM SelectCourse      WHERE Score>90
    FOR UPDATE
OPEN Scorecur
FETCH NEXT FROM Scorecur
    INTO @XH,@JSH,@KCH,@CJ
WHILE @@FETCH_STATUS=0
    BEGIN
       UPDATE SelectCourse SET Score=90
       WHERE CURRENT OF    Scorecur
       FETCH NEXT FROM    Scorecur
          INTO @XH,@JSH,@KCH,@CJ
    END
CLOSE    Scorecur
DEALLOCATE    Scorecur
SELECT     max(Score) ,min(Score)
     FROM SelectCourse       WHERE Score>=90
GO
```

结果：90,90

8.1.5 小结

- 游标在关系数据库中是一个非常重要的概念。游标提供了一种对表中检索出的数据进行操作的灵活手段，就本质而言，游标实际上是一种能从包括多条数据记录的结果集中每一次只提取一条记录的机制。游标总是与一条SQL选择语句相关联。游标由结果集和结果集中指定特定记录的游标位置组成。

8.2 存储过程

- 8.2.1 存储过程概述
- 8.2.2 创建并执行存储过程
- 8.2.3 修改及删除存储过程

8.2.1 存储过程概述

- 什么是存储过程

应用程序与SQL Server数据库交互执行某些操作有两种方法：

- 一种是存储在本地的应用程序记录操作命令，应用程序向SQL Server发送每一个命令，并对返回的数据进行处理；
- 另一种是在SQL Server中定义某个过程，其中记录了一系列的操作，每次应用程序只需调用该过程就可完成该操作。

这种在SQL Server中定义的过程被称为存储过程。

8.2.1 存储过程概述

- 定义：存储过程（Stored Procedure）是一组为了完成特定功能的SQL语句集合，经编译后存储在服务器端数据库中，利用存储过程可以加速SQL语句的执行。
- 分类：

系统存储过程：在SQL server安装成功后已经存储在系统数据库master中，以sp_为前缀命名。主要从系统表里获取信息。

自定义存储过程：由用户创建并能完成某一特定功能的存储过程。虽然既可以有参数又有返回值，但返回值只能指明执行是否成功，不能像函数那样被直接调用，只能用Exec来执行存储过程。

8.2.1 存储过程概述

- 存储过程的功能

SQL Server中的存储过程类似于编程语言中的过程和函数，它具有以下功能：

- 接受输入参数并返回多个输出值。
- 包含T-SQL语句用以完成特定的SQL Server操作。
- 返回一个指示成功与否及失败原因的状态代码给调用它的程序。

存储过程是一组预编译的Transact-SQL语句，主体构成是标准SQL命令，同时包括SQL的扩展：语句块、结构控制命令、变量、常量、运算符、表达式、流程控制等，所有这些组合在一起用于构造存储过程。

8.2.1 存储过程概述

- 存储过程的优点
  - 允许模块化编程，增强代码的重用性和共享性
  - 使用存储过程可以加快运行速度
  - 使用存储过程可以减少网络流量。
  - 存储过程可以作为安全性机制。

8.2.2 创建并执行存储过程

利用Transact-SQL命令创建存储过程的语法结构如下：

```
CREATE PROC[EDURE] 存储过程名
[ @参数 参数的数据类型 [= default][OUTPUT] ] [，. . . n]
[ WITH {RECOMPILE | ENCRYPTION| RECOMPILE，ENCRYPTION} ]
 AS  SQL语句
```

- ✓ **存储过程名：**遵守 SQL Server 标识符命名规则，最长为 128个字符。
- ✓ **@参数：**创建存储过程时可以声明一个或多个参数，最多为1024个。
- ✓ Default：参数的默认值，可以为NULL，也可以包含通配符(%或_)。
- ✓ OUTPUT：表明参数为一个输出参数，当使用EXECUTE执行时作为返回值，不能是TEXT类型
- ✓ WITH RECOMPILE：每次执行存储过程时重新编译，产生新的执行计划。
- ✓ WITH ENCRYPTION：将syscomments表中的存储过程文本进行加密，使用户不能利用sp_helptext查看存储过程内容。
- ✓ SQL语句： 作为存储过程主体部分的Transact-SQL内容。

8.2.2 创建并执行存储过程

创建简单的存储过程（可以带有嵌套、分组和聚合函数）

例1:

```
Create procedure proc_sql1
As select * from 职工 where 工资>2000
```

```
Exec proc_sql1
```

例2：

```
Create procedure proc_sql2
As
Begin
Select * from 职工
where 工资>(select max(工资) from 职工 where 仓库号='wh1'
   and 工资>all（select avg(工资) from 职工 group by (仓库
   号)))
End
```

```
Exec proc_sql2
```

8.2.2 创建并执行存储过程

创建简单的存储过程（可以带有多个select语句）

例3：

```
Create procedure proc_sql3
As
Begin
Select * from 职工 where 姓名 like ‘张%’
Select * from 仓库 where 仓库号 in(
Select 仓库号 from 职工 where 姓名 like ‘张%’
)
End
```

```
Exec proc_sql3
```

8.2.2 创建并执行存储过程

- 创建简单的存储过程（可以多表连接查询）

```
CREATE PROCEDURE borrowed_books1
As
SELECT r.编号,r.姓名,b.图书编号,k.书名,b.借期
FROM readers r inner join borrowinf b
    ON r.编号=b.读者编号 INNER JOIN books k
    ON b.图书编号=k.编号
WHERE 姓名='刘超'
```

8.2.2 创建并执行存储过程

创建带有输入参数的存储过程： SQL Server中存储过程的参数包括输入参数和输出参数。参数扩展了SQL Server的功能，通过储存过程每次执行时不同的参数值，实现其灵活性。

- 输入参数

  输入参数用于把值传入存储过程。

例4：

```
Create procedure proc_sql4
@mingz int,
@maxgz int
As
Select * from 职工 where 工资 between @mingz and @maxgz
```

```
Exec proc_sql4 1500,1800
```

8.2.2 创建并执行存储过程

创建带有输入参数的存储过程

例5：

```
Create procedure proc_sql5
@stu_id varchar(7),
@stu_name varchar(8),
@sex char(2),
@birthday smalldatetime,
@address varchar(30)
As
Insert into 学生 values(@stu_id, @stu_name, @sex,
  @birthday, @address )
```

```
Exec proc_sql5 '2011101','张三','女','1990-1-1','北京'
```

创建带有输入参数的存储过程

[例]

```
CREATE PROCEDURE borrowed_books2
  @name varchar(10)
  As
  SELECT r.编号,r.姓名,b.图书编号,k.书名,b.借期
  FROM readers r ， borrowinf b, books k
  WHERE r.编号=b.读者编号 AND
       b.图书编号=k.编号 AND 姓名=@name
```

创建带有输入参数的存储过程

将值传入存储过程有以下几种方法：

1)直接将值传入

如 **EXEC borrowed_books2 '张刚'**。

2)利用与声明时相同类型的变量来传递

如 **EXEC borrowed_books2 @templ (此处，@templ为已声明的字符类型变量，且已赋值)**。

- 利用下面程序查询0603001号学生选课信息_____处是（ ）
- CREATE PROCEDURE P3 @XH CHAR(9)
- AS
- SELECT S.StuName AS 姓名,C.CourseName AS 课程名,SC.Score AS 成绩
- FROM Student S,Course C,SelectCourse SC
- WHERE S.StuNo=SC.StuNo AND S.StuNo=@XH AND C.CourseNo=SC.CourseNo
- GO
- _____
- GO EXECUTE P3 '0603001'

8.2.2 创建并执行存储过程

创建使用参数默认值的存储过程

例6：

```
CREATE PROCEDURE proc_sql6
      @name varchar(10)=NULL
  As
  IF @name IS NULL
    SELECT r.编号,r.姓名,b.图书编号,k.书名,b.借期
    FROM readers r ， borrowinf b, books k
    WHERE r.编号=b.读者编号 AND  b.图书编号=k.编号
  ELSE
    SELECT r.编号,r.姓名,b.图书编号,k.书名,b.借期
     FROM readers r ， borrowinf b, books k
     WHERE r.编号=b.读者编号 AND
          b.图书编号=k.编号 AND 姓名=@name
```

```
Exec proc_sql6
```

8.2.2 创建并执行存储过程

创建带有输入和输出参数的存储过程

例7：

```
Create procedure proc_sql7
@cangkuhao varchar(50),
@maxgz int output,
@avggz real output
As
Begin
Select * from 职工 where 仓库 = @cangkuhao
Select @maxgz = max(工资) from 职工 where仓库号 = @cangkuhao
Select @avggz = avg(工资) from职工 where仓库号 = @cangkuhao
End
```

```
Declare @x1 int, @x2 real
Exec proc_sql7 'wh1',@x1 output, @x2 output
Select @x1 as 'wh1仓库最大工资', @x2 as 'wh1仓库职工平均工资'
```

8.2.2 创建并执行存储过程

创建带有输入和输出参数的存储过程

输出参数

输出参数用于把返回值赋予变量并传给调用它的存储过程或应用程序。声明输出参数时需在声明参数的后面加上**OUTPUT**，以表明此参数为输出参数。

执行语句语法格式：

DECLARE @输出参数 数据类型。。。

EXEC 存储过程名 输入参数值，@输出参数 **OUTPUT**

SELECT @输出参数。。。

8.2.2 创建并执行存储过程

创建加密存储过程

存储过程一旦加密，其定义即无法解密，任何人（包括存储过程的所有者或系统管理员）都无法查看存储过程的定义

例8：

```
Create procedure proc_sql8
With encryption
As
Select * from S
```

8.2.3 修改及删除存储过程

重命名存储过程

sp_rename 原存储过程名，新存储过程名

删除存储过程

drop proc 存储过程名

修改存储过程

alter proc 存储过程名

查看存储过程内容

sp helptext 存储过程名

8.2.3 函数

在用存储过程操作数据库中的数据时，往往会用到函数，以便在处理过程时更加方便、合理，还可以重复利用函数简化代码量。

自定义函数有两种：

（1）表值函数：返回一个表

（2）标量值函数：返回一个标量值

表值函数

具体函数结构如下：

<1>表值函数

```
create function dbo.funTblTest[注：函数名]
(
[注:参数]
)
returns @tmpTable[注:表格变量名] table
(
[注:表格变量定义]
)
as
begin
[注:sql语句]
end
```

8.2.3 函数

```
USE CollegeMIS
CREATE FUNCTION StuNo_Score(@StuNo char(9)) RETURNS table
AS
begin
    return
     (SELECT S.StuNo AS 学号,S.StuName AS 姓名,
        C.CourseNo AS 课程号,C.CourseName AS 课程名,
        SC.Score AS 成绩
      FROM Student S,Course C,SelectCourse SC
      WHERE S.StuNo=@StuNo AND
          S.StuNo=SC.StuNo AND SC.CourseNo=C.CourseNo  )
end
```

调用时：**select * from StuNo_Score('0204111')**

标量值函数

<2>标量值函数

```
create function[函数名]
(
[参数]
)
returns  [标量值类型]
as
begin
[注:sql语句]
end
```

```
CREATE FUNCTION [dbo].[testGetSubNodes]
(
 @nodeId int
)
RETURNS int
AS
BEGIN
    declare @nodeCount int
    select @nodeCount=5 from MenuTree
    return @nodeCount
END
```

调用：select dbo.testGetSubNodes

存储过程	函数
用于在数据库中完成特定的操作或者任务（如插入、删除等）	用于特定的数据（如选择）
程序头部声明用procedure	程序头部声明用function
程序头部声明时不需描述返回类型	程序头部声明时要描述返回类型，而且PL/SQL块中至少要包括一个有效的return语句
可以使用in/out/in out 三种模式的参数	可以使用in/out/in out 三种模式的参数
可作为一个独立的PL/SQL语句来执行	不能独立执行，必须作为表达式的一部分调用
可以通过out/in out 返回零个或多个值	通过return语句返回一个值，且改值要与声明部分一致，也可以是通过out类型的参数带出的变量
SQL语句(DML 或SELECT)中不可调用存储过程	SQL语句(DML 或SELECT)中可以调用函数