课程名称: 数据库系统

第10讲:(第5章)

高级SQL

一 动态SQL简介

- 许多应用的完成都需要采用某种程序语言和利用SQL完成基于数据库的数据处理

1）**动态SQL**

调用数据库厂商提供的通过函数/方法；

SQL语句-以字符串方式在运行时动态构建和提交；

2）**嵌入式SQL**

将SQL语句嵌入到程序中，但与动态SQL不同的是：

SQL语句-在程序编译时完全确定；

这样的应用程序在编译前，需要进行预编译和优化；

（这种预编译器由提供数据库厂商）

预编译将程序中的SQL代码转换成相应的代码和函数；

数据库厂商提供的库函数供程序编译(链接)时使用；

JDBC(for Java) p. 89

(ODBC for C++, C#, VB)

Java(JDBC)应用示例

```
public static void JDBCexample(String userid, String passwd)
{
    try
    {
        Class.forName ("oracle.jdbc.driver.OracleDriver");
        Connection conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@db.yale.edu:1521:univdb",
                userid, passwd);
        Statement stmt = conn.createStatement();
        try {
            stmt.executeUpdate(
                "insert into instructor values('77987', 'Kim', 'Physics', 98000)");
        } catch (SQLException sqle)
        {
            System.out.println("Could not insert tuple. " + sqle);
        }
        ResultSet rset = stmt.executeQuery(
                "select dept_name, avg (salary) "+
                " from instructor "+
                " group by dept_name");
        while (rset.next()) {
            System.out.println(rset.getString("dept_name") + " " +
                    rset.getFloat(2));
        }
        stmt.close();
        conn.close();
    }
    catch (Exception sqle)
    {
        System.out.println("Exception : " + sqle);
    }
}
```

图 5-1 JDBC 代码示例

载入Oracle for JDBC数据库驱动器

连接ORACLE数据库(指定协议、主机名、端口、数据库名、用户、口令)

动态生成SQL空语句句柄

动态生成SQL插入语句，并执行插入

判定SQL插入语句成功？否则输入错误信息

动态生成SQL查看语句，并执行

一次从查询结果中提取1个元组，并显示

查询输入完成时，释放资源

动态SQL语句可以包含(4个，四个？号)多个参数变量

为每个变量赋值

将该元组插入库

输入新的变量值后再插入1元组

```
PreparedStatement pStmt = conn.prepareStatement(
                "insert into instructor values(?,?,?,?)");
pStmt.setString(1, "88877");
pStmt.setString(2, "Perry");
pStmt.setString(3, "Finance");
pStmt.setInt(4, 125000);
pStmt.executeUpdate();
pStmt.setString(1, "88878");
pStmt.executeUpdate();
```

图 5-2 JDBC 代码中的预备语句

二 嵌入式SQL p. 95 应用于主语言：C, C++, Cobol, Pascal, Java, PL/I, Fortran

1. 程序编写的基本要素

预编译器通过标识识别嵌入在主语言中的SQL语句

```
Include files;
……
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
    char branchname[15] = "";
        char szServerDatabase[100];
        char szLogin[20];
        char szPassword[20];
    int  iassets= 0;
EXEC SQL END DECLARE SECTION;
…(读取并为前4个宿主变量赋值)…
EXEC SQL CONNECT TO :szServerDatabase
            USER :szLogin using :szPassword;
EXEC select assets into iassets
         from branch
        where branch_name = :branchname;
……
```

SQLCODE分量:返回执行状态

SQL通信区:SQLCA

申明SQL语句中将使用的宿主变量（将专门用于SQL语句中）

连接到数据库

查看部门的资产

将该单个属性值存放到iassets

加：---表示宿主变量

2. 游标(cursor)与返回结果集 p. 98

由于select查询结果为元组的集合；
必须建立SQL与主语言的数据交换区；
主语言通过游标指针，一次取一元组；

1. 定义游标

2. 打开游标
（执行语句）

3. 提取当前元组
并移到下一元组
（并放入变量si和sn）

4. 关闭游标
（释放资源）

```
……
…(申明宿主变量)…
…(读取和对宿主变量credit_amount赋值)…
……
EXEC SQL
    declare c cursor for
    select ID, name
    from student
    where tot_cred > :credit_amount;
EXEC SQL open  c;
while ... do{
EXEC SQL fetch c into :si, :sn;
...(打印/显示变量si和sn中的当前值)...
}
EXEC SQL close c;
……
```

2. 游标与数据更新 p. 98

为修改数据库中记录，
该语句定义一个游标。

```
EXEC SQL
  declare c cursor for
     select *
     from instructor
    where dept_name = 'Music'
    for update;
```

```
……
while .. do{
EXEC SQL fetch c;
……
```

该语句利用定义的游标，
对当前记录进行修改。

```
EXEC SQL
    update instructor
    set salary = salary + 100
    where current of c;
```

```
……
}
EXEC SQL close c;
```

需要的话，还可以添加语句：
EXEC SQL COMMIT-提交事务/
EXEC SQL ROLLBACK-回滚事务

三 SQL函数与过程(统称存储过程)

1. SQL函数p. 98

- SQL提供一些常用的内建函数(聚集、日期、字符串转换等)
- 还可编写存储过程(业务逻辑),存于库中，可在SQL/应用代码中调用！

定义(声明)一个函数：有一个输入参数，有一个输出参数(返回值)，begin-end为函数体.

作用：给定系名返回系人数

```
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
```

```
select dept_name, budget
from department
where dept_count (dept_name ) > 1
```

SQL查询语句中，可以使用用户定义的函数，就像使用系统固有函数一样。

```
create function instructors_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 = instructors_of.dept_name)
```

使用该函数的参数

定义的函数甚至可返回一个表，需仔细描述返回值(各属性)类型。这里函数体仅一个SQL语句，故函数体不需用begin-end界定。

```
select *
from table (instructors_of ('Music'))
```

函数返回表可以用在SQL查询中允许表出现的位置！

2. SQL过程p. 98-99

过程：一段SQL语句程序；
函数：有处理还有返回值

- 该过程的作用与前面的dept_count函数类似！

```
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
```

```
declare d_count integer;
call dept_count_proc( 'Physics', d_count);
```

存储过程可在SQL过程中或嵌入式SQL中通过call命令调用，还可在动态SQL中用

- 编写存储过程可以使用这些常规控制语句

```
If n < 0
          then
       return(-1)
else
  then return(1)
endif
```

```
declare n integer default 0;
while n < 10 do
          set n = n + 1
end while
```

```
repeat
          set n = n  – 1
  until n = 0
end repeat
```

```
begin … end
```

3. 存储过程示例** p. 100

```
——在确保教室能容纳下的前提下注册一个学生
——如果成功注册，返回0，如果超过教室容量则返回-1
create function registerStudent(
        in s_id varchar(5),
        in s_courseid varchar (8),
        in s_secid varchar (8),
        in s_semester varchar (6),
        in s_year numeric (4, 0),
        out errorMsg varchar(100)
returns integer
begin
    declare currEnrol int;
    select count( * ) into currEnrol
    from takes
    where course_id = s_courseid and sec_id = s_secid
          and semester = s_semester and year = s_year;
    declare limit int;
    select capacity into limit
    from classroom natural join section
    where course_id = s_courseid and sec_id = s_secid
          and semester = s_semester and year = s_year;
    if (currEnrol < limit)
      begin
          insert into takes values
              (s_id, s_courseid, s_secid, s_semester, s_year, null);
          return(0);
      end
    ——否则，已经达到课程容量上限
    set errorMsg = 'Enrollment limit reached for course ' || s_courseid
        || ' section ' || s_secid;
    return( -1);
end;
```

输入课程信息参数

输出错误参数

统计课程人数，暂存于*currEnrol*

查看课程人数限制，暂存于*limit*

满足人数限制时，注册一个学生
（且注册成功时，返回0）

否则，输出错误信息到*errorMsg*
（且注册不成功时，返回-1）

图 5-7　学生注册课程的过程

四 触发器

1. 定义触发器

```
create trigger credits_earned after update of takes on (grade)
    referencing new row as nrow
    referencing old row as orow
    for each row
    when nrow.grade <> 'F' and nrow.grade is not null
        and (orow.grade = 'F' or orow.grade is null)
    begin atomic
        update student
        set tot_cred = tot_cred +
            (select credits
             from course
             where course.course_id = nrow.course_id)
        where student.id = nrow.id;
    end;
```

当属性值改变时

将对改变前后的记录比较

有新记录成绩且非F
旧记录成绩为F或空

begin-end可有多个语句
atomic-表示为一个事务

将根据takes的课程成绩记录，
重新计算相应学生的总学分数.

图 5-9 使用触发器来维护 *credits_earned* 值
(student表的属性)

```
create trigger timeslot_check1 after insert on section
    referencing new row as nrow
    for each row
    when (nrow.time_slot_id not in (
            select time_slot_id
            from time_slot)) /* time_slot 中不存在该 time_slot_id */
    begin
      rollback
    end;
```

撤销新插入的记录！

当time_slot中不存在时，拒绝在section表上插入。

```
    create trigger timeslot_check2 after delete on time slot
    referencing old row as orow
    for each row
    when (orow.time_slot_id not in (
            select time_slot_id
            from time_slot) /* 在 time_slot 中刚刚被删除的 time_slot_id */
    and orow.time_slot_id in (
            select time_slot_id
            from section)) /* 在 section 中仍含有该 time_slot_id 的引用 */
    begin
      rollback
    end;
```

撤销删除的记录！

当section中还有参照记录存在时，拒绝删除time_slot中被参照记录。

图 5-8　使用触发器来维护参照完整性