数据库系统概论
An Introduction to Database System

第八章 数据库编程

中国人民大学信息学院

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

- SQL 语言提供了两种不同的使用方式:
  - 交互式
  - 嵌入式
- 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，在程序设计的环境下， SQL 语句要做某些必要的扩充

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

8.1.3 不使用游标的 SQL 语句

8.1.4 使用游标的 SQL 语句

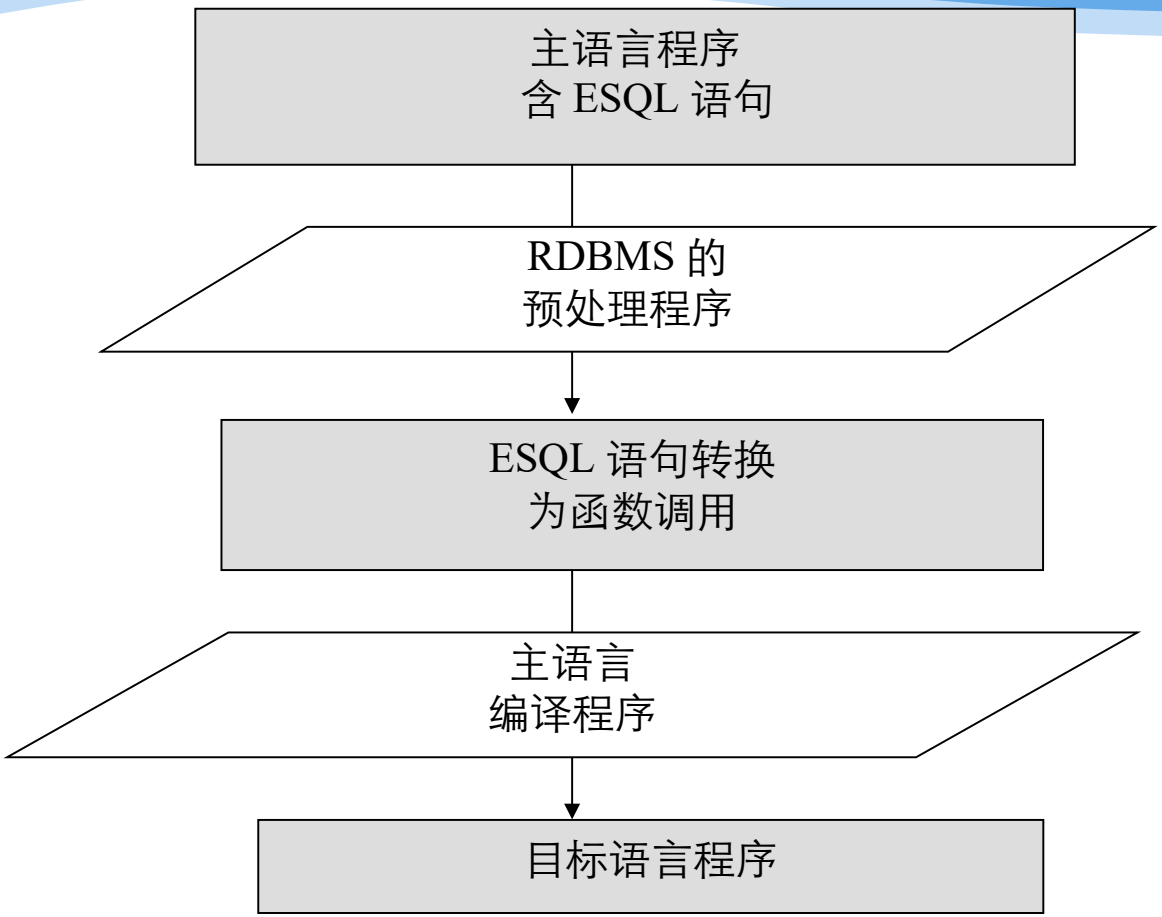
8.1.5 动态 SQL

8.1.6 小结

8.1.1 嵌入式 SQL 的处理过程

- 主语言
  - 嵌入式 SQL 是将 SQL 语句嵌入程序设计语言中，被嵌入的程序设计语言，如 C 、 C++ 、 Java ，称为宿主语言，简称主语言。
- 处理过程
  - 预编译方法

嵌入式 SQL 的处理过程（续）



ESQL 基本处理过程

嵌入式 SQL 的处理过程（续）

- 为了区分 SQL 语句与主语言语句， 所有 SQL 语句必须加前缀 EXEC SQL ， 以 (;) 结束 :

  EXEC SQL <SQL 语句 >;

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 与主语言的通信

8.1.3 不使用游标的 SQL 语句

8.1.4 使用游标的 SQL 语句

8.1.5 动态 SQL

8.1.6 小结

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
  - 它们之间应该如何通信?

嵌入式 SQL 语句与主语言之间的通信（续）

- 数据库工作单元与源程序工作单元之间的通信：
  - 1. SQL 通信区
    - 向主语言传递 SQL 语句的执行状态信息
    - 使主语言能够据此控制程序流程
  - 2. 主变量
    - 主语言向 SQL 语句提供参数
    - 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

一、 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， RDBMS 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

SQL 通信区

- SQLCA 使用方法：
  - 定义 SQLCA
    - 用 EXEC SQL INCLUDE SQLCA 定义
  - 使用 SQLCA
    - SQLCA 中有一个存放每次执行 SQL 语句后返回代码的变量 SQLCODE
    - 如果 SQLCODE 等于预定义的常量 SUCCESS，则表示 SQL 语句成功，否则表示出错
    - 应用程序每执行完一条 SQL 语句之后都应该测试一下 SQLCODE 的值，以了解该 SQL 语句执行情况并做相应处理

二、主变量

- 主变量
  - 嵌入式 SQL 语句中可以使用主语言的程序变量来输入或输出数据
  - 在 SQL 语句中使用的主语言程序变量简称为主变量（Host Variable）

主变量（续）

- 主变量的类型
  - 输入主变量
  - 输出主变量
  - 一个主变量有可能既是输入主变量又是输出主变量

主变量（续）

- 指示变量：
  - 一个主变量可以附带一个指示变量（Indicator Variable）
  - 什么是指示变量
  - 指示变量的用途

主变量（续）

- 在 SQL 语句中使用主变量和指示变量的方法
  - 1) 说明主变量和指示变量

```
BEGIN DECLARE SECTION
    .....
    .....    （说明主变量和指示变量）
    .....
END DECLARE SECTION
```

主变量（续）

- 2) 使用主变量
  - 说明之后的主变量可以在 SQL 语句中任何一个能够使用表达式的地方出现
  - 为了与数据库对象名（表名、视图名、列名等）区别， SQL 语句中的主变量名前要加冒号（：）作为标志
- 3) 使用指示变量
  - 指示变量前也必须加冒号标志
  - 必须紧跟在所指主变量之后

主变量（续）

- 在 SQL 语句之外 (主语言语句中) 使用主变量和指示变量的方法
  - 可以直接引用，不必加冒号

三、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

游标（续）

- 游标
  - 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
  - 每个游标区都有一个名字
  - 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

四、建立和关闭数据库连接

- 建立数据库连接

  EXEC SQL CONNECT TO *target* [AS *connection-name*] [USER *user-name*];

  target 是要连接的数据库服务器：
  - 常见的服务器标识串，如 <dbname>@<hostname>:<port>
  - 包含服务器标识的 SQL 串常量
  - DEFAULT

  connect-name 是可选的连接名，连接必须是一个有效的标识符

  在整个程序内只有一个连接时可以不指定连接名

- 关闭数据库连接

  EXEC SQL DISCONNECT [*connection*];

- 程序运行过程中可以修改当前连接：

  EXEC SQL SET CONNECTION *connection-name* | DEFAULT;

五、程序实例

[例 1] 依次检查某个系的学生记录，交互式更新某些学生年龄。

```
EXEC SQL BEGIN DEC LARE SECTION;  /* 主变量说明开始 */
    char deptname[64];
    char HSno[64];
    char HSname[64];
    char HSsex[64];
    int    HSage;
    int    NEWAGE;
EXEC SQL END DECLARE SECTION;     /* 主变量说明结束 */
long SQLCODE;
EXEC SQL INCLUDE sqlca;                /* 定义 SQL 通信区 */
```

程序实例（续）

```
int main(void)                          /*C 语言主程序开始 */
{
    int    count = 0;
    char  yn;                           /* 变量 yn 代表 yes 或 no*/
    printf("Please choose the department name(CS/MA/IS): ");
    scanf("%s", deptname);                  /* 为主变量 deptname 赋值 */
    EXEC SQL CONNECT TO TEST@localhost:54321 USER
     "SYSTEM" /"MANAGER";          /* 连接数据库 TEST*/
    EXEC SQL DECLARE SX CURSOR FOR   /* 定义游标 */
          SELECT Sno, Sname, Ssex, Sage   /*SX 对应语句的执行结果
    */
          FROM Student
          WHERE SDept = :deptname;
    EXEC SQL OPEN SX;          /* 打开游标 SX 便指向查询结果的第一行
    */
```

程序实例（续）

```
for ( ; ; )                          /* 用循环结构逐条处理结果集中的记录 */
{
    EXEC SQL FETCH SX INTO :HSno, :HSname, :HSsex,:HSage;
                                     /* 推进游标，将当前数据放入主变量 */
    if (sqlca.sqlcode != 0)          /* sqlcode != 0, 表示操作不成功 */
          break;                     /* 利用 SQLCA 中的状态信息决定何时退出循环 */
    if(count++ == 0)                 /* 如果是第一行的话，先打出行头 */
     printf("\n%-10s %-20s %-10s %-10s\n", "Sno", "Sname", "Ssex", "Sage");
     printf("%-10s %-20s %-10s %-10d\n", HSno, HSname, HSsex, HSage);
                                              /* 打印查询结果 */
     printf("UPDATE AGE(y/n)?"); /* 询问用户是否要更新该学生的年龄 */
    do{
          scanf("%c",&yn);
     }
     while(yn != 'N' && yn != 'n' && yn != 'Y' && yn != 'y');
```

程序实例（续）

```
    if (yn == 'y' || yn == 'Y')              /* 如果选择更新操作 */
          {
           printf("INPUT NEW AGE:");
           scanf("%d",&NEWAGE);          /* 用户输入新年龄到主变量中 */
           EXEC SQL UPDATE Student                    /* 嵌入式 SQL*/
                    SET Sage = :NEWAGE
                    WHERE CURRENT OF SX ;
          }              /* 对当前游标指向的学生年龄进行更新 */
}

    EXEC SQL CLOSE SX;          /* 关闭游标 SX 不再和查询结果对应 */
    EXEC SQL COMMIT WORK;                      /* 提交更新 */
    EXEC SQL DISCONNECT TEST;                 /* 断开数据库连接 */
}
```

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

8.1.3 不使用游标的 SQL 语句

8.1.4 使用游标的 SQL 语句

8.1.5 动态 SQL

8.1.6 小结

8.1.3 不用游标的 SQL 语句

- 不用游标的 SQL 语句的种类
  - 说明性语句
  - 数据定义语句
  - 数据控制语句
  - 查询结果为单记录的 SELECT 语句
  - 非 CURRENT 形式的增删改语句

不用游标的 SQL 语句（续）

- 一、查询结果为单记录的 SELECT 语句
- 二、非 CURRENT 形式的增删改语句

一、查询结果为单记录的 SELECT 语句

- 这类语句不需要使用游标，只需要用 INTO 子句指定存放查询结果的主变量

[例 2] 根据学生号码查询学生信息。假设已经把要查询的学生的学号赋给了主变量 givensno 。

```
EXEC SQL SELECT Sno , Sname , Ssex , Sage , Sdept
                INTO :Hsno , :
    Hname , :Hsex , :Hage , :Hdept
                FROM Student
                WHERE Sno=:givensno ;
```

查询结果为单记录的 SELECT 语句（续）

(1) INTO 子句、 WHERE 子句和 HAVING 短语的条件表达式中均可以使用主变量

(2) 查询返回的记录中，可能某些列为空值 NULL 。

(3) 如果查询结果实际上并不是单条记录，而是多条记录，则程序出错， RDBMS 会在 SQLCA 中返回错误信息

查询结果为单记录的 SELECT 语句（续）

[例 3] 查询某个学生选修某门课程的成绩。假设已经把将要查询的学生的学号赋给了主变量 givensno ， 将课程号赋给了主变量 givencno 。

```
EXEC SQL SELECT Sno ， Cno ， Grade
                INTO :Hsno ， :Hcno ， :Hgrade:Gradeid
                                        /* 指示变量 Gradeid*/
                FROM  SC
                WHERE Sno=:givensno AND Cno=:givencno ；
```

如果 Gradeid < 0 ， 不论 Hgrade 为何值， 均认为该学生成绩为空值 。

二、非 CURRENT 形式的增删改语句

- 在 UPDATE 的 SET 子句和 WHERE 子句中可以使用主变量， SET 子句还可以使用指示变量

[例 4] 修改某个学生选修 1 号课程的成绩。

```
EXEC SQL UPDATE SC
                SET Grade=:newgrade       /* 修改的成绩已赋给主变量 */
                 WHERE Sno=:givensno ;    /* 学号赋给主变量
givensno*/
```

非 CURRENT 形式的增删改语句（续）

[例 5] 将计算机系全体学生年龄置 NULL 值。

```
Sageid=-1 ;
EXEC SQL UPDATE Student
                SET Sage=:Raise :Sageid
                WHERE Sdept= ‘CS’ ;
```

将指示变量 Sageid 赋一个负值后，无论主变量 Raise 为何值，RDBMS 都会将 CS 系所有学生的年龄置空值 。

等价于：

```
EXEC SQL UPDATE Student
                SET Sage=NULL
                WHERE Sdept= 'CS' ;
```

非 CURRENT 形式的增删改语句（续）

[例 6] 某个学生退学了，现要将有关他的所有选课记录删除掉。假设该学生的姓名已赋给主变量 stdname 。

```
EXEC SQL DELETE
            FROM SC
            WHERE Sno=
                (SELECT Sno
                 FROM Student
                 WHERE Sname=:stdname) ;
```

非 CURRENT 形式的增删改语句（续）

[例 7] 某个学生新选修了某门课程，将有关记录插入 SC 表中。假设插入的学号已赋给主变量 stdno ， 课程号已赋给主变量 couno 。

```
gradeid=-1 ;                /* 用作指示变量，赋为负值 */
EXEC SQL INSERT
        INTO SC(Sno ， Cno ， Grade)
        VALUES(:stdno ， :couno ， :gr :gradeid) ;
```

由于该学生刚选修课程，成绩应为空，所以要把指示变量赋为负值

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
- 8.1.3 不使用游标的 SQL 语句
- 8.1.4 使用游标的 SQL 语句
- 8.1.5 动态 SQL
- 8.1.6 小结

8.1.4 使用游标的 SQL 语句

- 必须使用游标的 SQL 语句
  - 查询结果为多条记录的 SELECT 语句
  - CURRENT 形式的 UPDATE 语句
  - CURRENT 形式的 DELETE 语句

使用游标的 SQL 语句（续）

- 一、 查询结果为多条记录的 SELECT 语句
- 二、 CURRENT 形式的 UPDATE 和 DELETE 语句

一、 查询结果为多条记录的 SELECT 语句

- 使用游标的步骤
  1. 说明游标
  2. 打开游标
  3. 推进游标指针并取当前记录
  4. 关闭游标

1. 说明游标

- 使用 DECLARE 语句
- 语句格式

  EXEC SQL DECLARE < 游标名 > CURSOR
  FOR <SELECT 语句 >;

- 功能
  - 是一条说明性语句，这时 DBMS 并不执行 SELECT 指定的查询操作。

2. 打开游标

- 使用 OPEN 语句
- 语句格式

  EXEC SQL OPEN < 游标名 >;

- 功能
  - 打开游标实际上是执行相应的 SELECT 语句，把所有满足查询条件的记录从指定表取到缓冲区中
  - 这时游标处于活动状态，指针指向查询结果集中第一条记录

3. 推进游标指针并取当前记录

- 使用 FETCH 语句
- 语句格式

EXEC SQL FETCH [[NEXT|PRIOR|
FIRST|LAST] FROM] < 游标名 >
INTO < 主变量 >[< 指示变量 >][,< 主变量 >[< 指示变量 >]]...;

推进游标指针并取当前记录（续）

- 功能
  - 指定方向推动游标指针，然后将缓冲区中的当前记录取出来送至主变量供主语言进一步处理
  - NEXT|PRIOR|FIRST|LAST： 指定推动游标指针的方式
    - NEXT： 向前推进一条记录
    - PRIOR： 向回退一条记录
    - FIRST： 推向第一条记录
    - LAST： 推向最后一条记录
    - 缺省值为 NEXT

4. 关闭游标

- 使用 CLOSE 语句
- 语句格式

  EXEC SQL CLOSE < 游标名 >;

- 功能
  - 关闭游标，释放结果集占用的缓冲区及其他资源
- 说明
  - 游标被关闭后，就不再和原来的查询结果集相联系
  - 被关闭的游标可以再次被打开，与新的查询结果相联系

二、 CURRENT 形式的 UPDATE 语句和 DELETE 语句

- CURRENT 形式的 UPDATE 语句和 DELETE 语句的用途
  - 面向集合的操作
  - 一次修改或删除所有满足条件的记录

CURRENT 形式的 UPDATE 语句和 DELETE 语句（续）

- 如果只想修改或删除其中某个记录
  - 用带游标的 SELECT 语句查出所有满足条件的记录
  - 从中进一步找出要修改或删除的记录
  - 用 CURRENT 形式的 UPDATE 语句和 DELETE 语句修改或删除之
  - UPDATE 语句和 DELETE 语句中的子句：

    WHERE CURRENT OF < 游标名 >

    表示修改或删除的是最近一次取出的记录，即游标指针指向的记录

CURRENT 形式的 UPDATE 语句和 DELETE 语句（续）

- 不能使用 CURRENT 形式的 UPDATE 语句和 DELETE 语句 ：
  - 当游标定义中的 SELECT 语句带有 UNION 或 ORDER BY 子句
  - 该 SELECT 语句相当于定义了一个不可更新的视图

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
- 8.1.3 不使用游标的 SQL 语句
- 8.1.4 使用游标的 SQL 语句
- 8.1.5 动态 SQL
- 8.1.6 小结

8.1.5 动态 SQL

- 静态嵌入式 SQL
  - 静态嵌入式 SQL 语句能够满足一般要求
  - 无法满足要到执行时才能够确定要提交的 SQL 语句
- 动态嵌入式 SQL
  - 允许在程序运行过程中临时“组装” SQL 语句
  - 支持动态组装 SQL 语句和动态参数两种形式

动态 SQL 简介（续）

- 一、使用 SQL 语句主变量
- 二、动态参数

一、使用 SQL 语句主变量

- SQL 语句主变量：
  - 程序主变量包含的内容是 SQL 语句的内容，而不是原来保存数据的输入或输出变量
  - SQL 语句主变量在程序执行期间可以设定不同的 SQL 语句，然后立即执行

使用 SQL 语句主变量（续）

[例 9]　创建基本表 TEST

```
EXEC SQL BEGIN DECLARE SECTION;
const char *stmt = "CREATE TABLE test(a int);";
                /* SQL 语句主变量  */
EXEC SQL END DECLARE SECTION;
... ...
EXEC SQL EXECUTE IMMEDIATE :stmt;
                /*  执行语句  */
```

二、动态参数

- 动态参数
  - SQL 语句中的可变元素
  - 使用参数符号 (?) 表示该位置的数据在运行时设定
- 和主变量的区别
  - 动态参数的输入不是编译时完成绑定
  - 而是通过 (prepare) 语句准备主变量和执行 (execute) 时绑定数据或主变量来完成

动态参数（续）

- 使用动态参数的步骤：

1. 声明 SQL 语句主变量。

2. 准备 SQL 语句 (PREPARE) 。

EXEC SQL PREPARE < 语句名 > FROM <SQL 语句主变量 >;

动态参数（续）

- 使用动态参数的步骤（续）：

3. 执行准备好的语句 (EXECUTE)

EXEC SQL EXECUTE < 语句名 > [INTO < 主变量表 >]
[USING < 主变量或常量 >];

动态参数（续）

[例 10] 向 TEST 中插入元组。

```
EXEC SQL BEGIN DECLARE SECTION;
const char *stmt = "INSERT INTO test VALUES(?);";
                    /* 声明 SQL 主变量  */
EXEC SQL END DECLARE SECTION;
... ...
EXEC SQL PREPARE mystmt FROM :stmt;   /*  准备语句  */
... ...
EXEC SQL EXECUTE mystmt USING 100; /*  执行语句  */
EXEC SQL EXECUTE mystmt USING 200; /*  执行语句  */
```

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

8.1.3 不使用游标的 SQL 语句

8.1.4 使用游标的 SQL 语句

8.1.5 动态 SQL

8.1.6 小结

8.1.6 小 结

- 在嵌入式 SQL 中， SQL 语句与主语言语句分工非常明确
  - SQL 语句
    - 直接与数据库打交道，取出数据库中的数据。
  - 主语言语句
    - 控制程序流程
    - 对取出的数据做进一步加工处理

小结（续）

- 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