第八章 数据库编程

8.1 嵌入式 SQL

SQL 语言的使用方式：嵌入式和交互式

8.1.1 嵌入式 SQL 的处理过程

主语言：即宿主语言，嵌入式 SQL 是将 SQL 语句嵌入到程序设计语言中

处理过程：预编译

具体过程：主语言程序->预编译->程序编译->目标语言程序

主语言为 C 语言时语句格式：EXEC SQL<SQL 语句>

8.1.2 嵌入式 SQL 语句与主语言之间的通信

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，负责控制逻辑流程

数据库工作单元与源程序工作单元之间的通信

（1） 向主语言传递 SQL 语句执行状态，用 SQL 通信区实现

（2） 主语言向 SQL 提供参数，用主变量实现

（3） 将 SQL 语句查询数据库结果交给主语言，由主变量和游标实现

1、SQL 通信区

即 SQLCA

工作流程：SQL 语句执行后，数据库反馈系统当前工作状态及运行环境，这些信息存在 SQL 通信区，应用程序从 SQL 通信区取出信息。

用法：

定义 SQLCA：EXEC SQL INCLUDE SQLCA

使用 SQLCA：SQL 每次执行完返回 SQLCODE，若为 SUCCESS 则表示成功，否则出错。应用程序每执行完一条 SQL 语句后都应测试 SQLCODE 的值

2、主变量

主语言程序变量

主变量类型：（1）输入主变量 应用程序赋值，SQL 引用（2）输出 SQL 赋值或设置状态信息，放回给应用程序

指示变量：整型变量，用来“指示”主变量的值或条件，一个主变量可带一个指示变量

指示变量用途：指示输入变量是否为空值，输出变量是否为空值，值是否被截断

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

SQL 中主变量名前加（：）作为标志，指示变量也，必须紧跟在所指主变量之后

在 SQL 语句之外引用可以直接引用不用加冒号

3、游标

原因：主语言一次存放一条记录，SQL 语句一次可以产生或处理多条记录

定义：游标是一个数据缓冲区，存放 SQL 语句执行结果

4、建立和关闭数据库连接

（1）建立数据库连接

EXEC SQL CONNECT TO target[AS connenction-name][USER user-name]

target 是要连接的数据库服务器（<daname>@<hostname>:<port>）

EXEC SQL SET CONNECTION connection-name|DEFAULT

（2）关闭数据库连接

EXEC SQL DISCONNECT[connection];

8.1.3 不使用游标的 SQL 语句

语句种类：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

1、查询结果为单记录的 SELECT 语句

不用游标，只需用 INTO 子句指定存放查询结果的主变量

INTO, WHERE, HAVING 短语都可以使用主变量

UPDATE 的 SET 子句和 WHERE 子句可用主变量，SET 子句还可以使用主变量

8.1.4 使用游标的 SQL 语句

必须使用游标的 SQL 语句：

1、查询结果为多条记录的 SELECT 语句

（1）说明游标

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

说明性语句，系统不执行 SELECT 语句

（2）打开游标

EXEC SQL OPEN<游标名>

即执行 SELECT 语句，查询结果放到缓冲区中，游标指针指向结果集第一条记录

（3）推进游标指针并取当前记录

EXEC SQL FETCH<游标名>INTO

<主变量>[<指示变量>][,<主变量>[<指示变量>]]...;

指定方向推动游标指针，同时将缓冲区中的记录取出来送至主变量

（4）关闭游标

EXEC SQL CLOSE<游标名>

2、CURRENT 形式的 UPDATE 和 DELETE

修改或删除最近一次取出的记录，即缓冲区中最新存的记录，要用

WHERE CURRENT OF<游标名>

当游标定义中的 SELECT 语句带有 UNION 或 ORDER BY 子句不能使用游标，因为 SELECT 语句相当于定义了一个不可更新视图

8.1.5 动态 SQL

静态嵌入式 SQL：无法到执行时才能够确定要提交的 SQL 语句（如查询条件等）

动态————————：SQL 语句可以临时组装，支持动态组装 SQL 语句和动态参数

1、使用 SQL 语句主变量

程序主变量包含的内容是 SQL 语句的内容，而不是输入输出变量

EXEC SQL BEGIN DECLARE SECTION;

const char *stmt="CREATE TABLE test(a int);";

/*SQL 语句主变量，内容是创建表的 SQL 语句*/

EXEC SQL END DECLARE SECTION;

……

EXEC SQL EXECUTE IMMEDIATE :stmt;

2、动态参数

SQL 语句的可变元素，使用“？”表示该位置的数据

动态参数不是编译时绑定，而是通过 PREPARE 语句准备主变量和 EXECUTE 绑定

数据或主变量来完成

使用步骤：

（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>；

3、执行准备好的语句

[INTO<主变量表>]

[USING<主变量或常量>]

8.2 过程化 SQL

8.2.1 过程化 SQL 的块结构

基本的 SQL 是高度非过程化的语言；过程化 SQL 程序的基本结构是块。

1、定义部分

DECLARE 变量、常量、游标、异常等（常量、变量等只能在基本块中使用）

2、执行部分

BEGIN

EXCEPTION

END

8.2.2 变量和常量的定义

1、变量定义

变量名 数据类型 [[NOT NULL]:=初值表达式] 或者

变量名 数据类型 [[NOT NULL] 初值表达式]

2、常量定义

常量名 数据类型 CONSTANT :=常量表达式

常量必须要给一个值，且不能改变。如果修改它，将返回一个异常

3、赋值语句

变量名称 := 表达式

8.2.3 流程控制

1、条件控制语句

（1）IF condition THEN

Sequence_of_statements;

END IF;

（2）IF condition THEN

Sequence_of_statements1;

ELSE

Sequence_of_statements2;

END IF;

（3） 在 THEN 和 ELSE 子句中还可以再包含 IF 语句，即 IF 语句可以嵌套

2、循环控制语句

（1）简单的循环语句 LOOP

LOOP

Sequence_of_statements;

END LOOP;（可以使用 EXIT, BREAK, LEAVE）

（2）WHILE-LOOP

WHILE condition LOOP

Sequence_of_statements;

END LOOP;

（3）FOR-LOOP

FOR count IN [REVERSE] bound1 … bound2 LOOP

Sequence_of_statements;

END LOOP;

3、错误处理

若出现异常，就在产生异常处停下来，根据异常类型执行异常处理语句

8.3 存储过程和函数

过程化 SQL 有两种：

匿名块：每次执行都编译，不存储在数据库中，不能在过程化 SQL 块中被调用

命名块：编译后即保存在数据库中，称为持久性存储模块，可反复调用

8.3.1 存储过程

存储过程是由过程化 SQL 语句书写的过程，这个过程经编译和优化后存储在数据库服务器中，使用时只要调用即可

1、优点：运行效率高；降低了客户机和服务器之间的通信量；方便实施企业规则

2、存储过程的用户接口

（1）创建存储过程

CREATE OR REPLACE PROCEDURE 过程名([参数 1,参数 2,...])

AS <过程化 SQL 块>；

过程名；参数列表；过程体（声明部分和可执行语句部分）例子：P256

（2）执行存储过程

CALL/PERFORM PROCEDURE 过程名([参数 1,参数 2,...]); 例子：P257

过程体可调用其他存储过程

（3）修改存储过程（重命名）

ALTER PRODUCE 过程名 1 RENAME TO 过程名 2

ALTER PROCEDURE 过程名 COMPILE;（重新编译）

（4）删除

DROP PRODUCE 过程名()

8.3.2 函数

函数与存储过程：持久性存储模块，函数必须指定返回的类型

1. 函数的定义语句格式

CREATE OR REPLACE FUNCTION 函数名 ([参数 1,参数 2,…]) RETURNS <类型>

AS <过程化 SQL 块>;

2. 函数的执行语句格式

CALL/SELECT 函数名 ([参数 1,参数 2,…]);

3. 修改函数（重命名/重新编译）

ALTER FUNCTION 过程名 1 RENAME TO 过程名 2;

ALTER FUNCTION 过程名 COMPILE;

8.4 ODBC 编程

优点：移植性好；能同时访问不同的数据库，共享多个数据资源

8.4.1 ODBC 概述

ODBC 规范应用开发，规范关系 DBMS 应用接口

8.4.2 ODBC 工作原理概述

体系结构：用户应用程序，ODBC 驱动程序管理器，数据库驱动程序，数据源

1、用户应用程序

2、ODBC 驱动程序管理器

装载 ODBC 驱动程序，选择并连接正确的驱动程序，管理数据源，检查参数合法性，记录函数调用

3、数据库驱动程序

单束：驱动程序和应用程序在一台机器上，直接读取数据

多束：驱动程序通过接口读取数据

4、ODBC 数据源

最终访问的数据，包含数据库位置和数据库类型等信息

8.4.3 ODBC API 基础

ODBC 应用程序编程几口的一致性

API 一致性：包含核心机、扩展 1 级、扩展 2 级

语法一致性：最低限度/核心 SQL 语法级/扩展语法级

1、函数概述

ODBC3.9 标准提供了 76 个函数接口

2、句柄及其属性

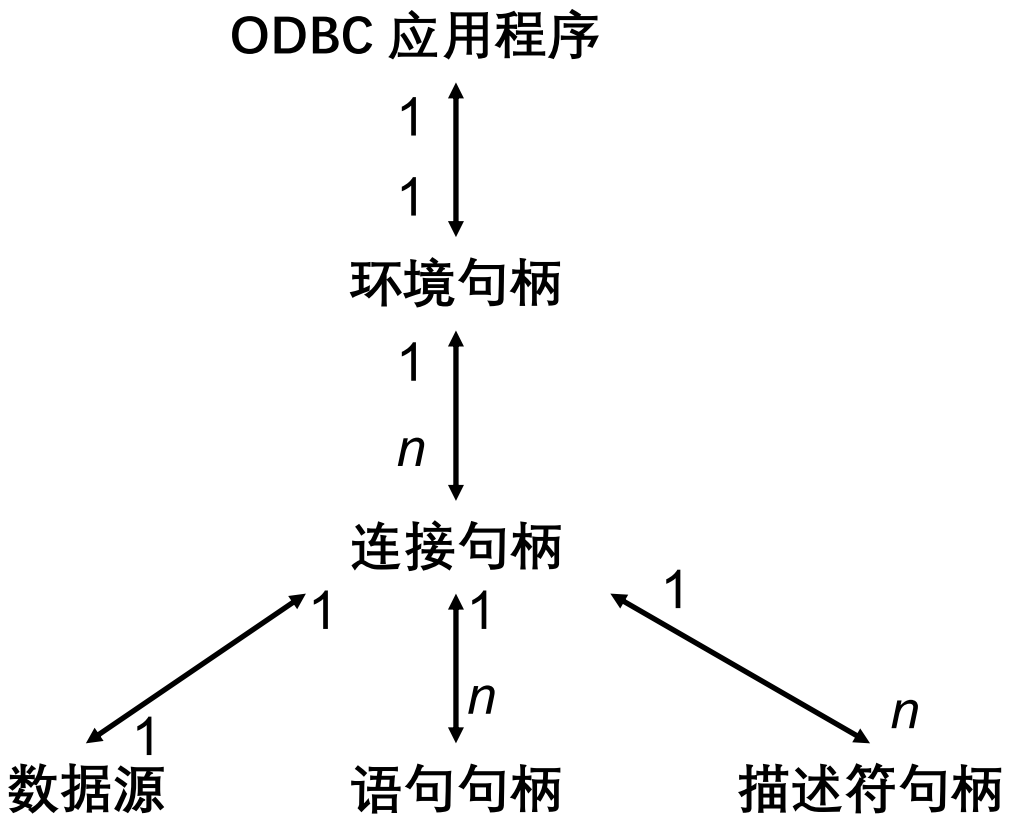
句柄分类：环境句柄、连接句柄、语句句柄、描述符句柄

环境句柄：存取数据的全局性背景，如环境状态。当前环境状态诊断、连接句柄

一个环境可以建立多个连接句柄，每个连接句柄连接一个数据源

一个连接中可建立多个语句句柄，不只是 SQL 语句，还包括 SQL 语句产生的结果集及相关信息

描述符句柄：描述 SQL 语句的参数、结果集列的元数据集合



3、数据类型

SQL 数据类型：用于数据源 C 数据类型：用于应用程序的 C 代码

8.4 ODBC 的工作流程

配置数据源 初始化环境 建立连接 分配语句句柄 执行 SQL 语句 处理结果集 中止处理

1、配置数据源

运行数据源管理工具来配置

使用 Driver Manager 提供的 ConfigDsn 来增删改数据源

2、初始化环境

由 DriverManager 进行控制，此时没有分配环境句柄的数据结构

3、建立连接

调用 SQLAllocHandle 分配连接句柄，通过 SQLConnect、SQLDriverConnect 或 SQLBrowseConnect 与数据源连接

SQLConnect 连接函数的输入参数为：

配置好的数据源名称、用户 ID、口令

4、分配语句句柄

先用 SQLAllocHandle 分配语句句柄

还可通过 SQLtStmtAttr 设置语句属性

5、执行 SQL 语句

预处理（SQLPrepare、SQLExecute）

直接执行（SQLExecdirect）

若含参，用 SQLBindParameter，把每个参数绑定至应用程序变量（每参一个）

6、结果集处理

可通过 SQLNunResultCols 获取结果集中列数

通过 SQL DescribeCol 或是 SQLColAttrbute 获取结果集每一列的名称、数据类型、精度和范围（剩下看书）

7、中止处理

释放语句句柄、断开与数据库连接、断开与服务器连接、释放 ODBC 环境