OSCAR 中嵌入式 SQL 的设计与实现*

黄晓燕，杨子江，董金祥
(浙江大学 计算机科学与技术学院，浙江 杭州 310027)

摘　要: 实现了一个嵌入式 SQL 系统,支持在 C 语言中嵌入 SQL 语句。该系统是对象关系型数据库管理系统 OSCAR 客户端的重要组成部分,分为两部分实现: 预处理模块和 ESQL 运行库。预处理模块根据 ESQL 语言的语法规则分析和识别 ESQL 语句,并且转换为实现相同功能的 C 语言函数调用; ESQL 运行库实现这些 C 函数,通过网络通信库与服务器端通信,并执行 SQL 语句,返回结果值。

关键词: 嵌入式 SQL; DBMS; SQL-92
中图法分类号: TP311.13　　**文献标识码**: A　　**文章编号**: 1001-3695(2005)01-0137-03

Design and Implementation of the Embedded SQL System in OSCAR

HUANG Xiao-yan, YANG Zi-jiang, DONG Jin-xiang
(College of Computer Science & Technology, Zhejiang University, Hangzhou Zhejiang 310027, China)

Abstract: To enable an embedding of SQL statements in C language, the embedded SQL system is implemented. The system plays a crucial role in the client part of OSCAR, an object relational database management system. It includes two components: preprocess module and ESQL runtime library. The former parses ESQL statements with the syntactic rules of ESQL language, and translates them into corresponding C function calls. The latter implements the C functions that communicate with the server via network communication library. The server then executes the SQL statements and returns results to the client.
Key words: Embedded SQL System; DBMS; SQL-92

1　引言

结构化查询语言(Structured Query Language, SQL)[1]是最普遍使用的关系数据库语言。它具有表达能力强、功能丰富、语言简洁、使用灵活等特点。但是 SQL 语言是面向集合的描述性语言,本身不具有过程性结构。而大多数的数据查询应用都是过程性的,要求程序根据不同的条件执行不同的动作,或者能和一个具有图形化用户界面的高级语言结合起来,因此 SQL 语句必须嵌入到高级语言中,如 C, FORTRAN, COBOL 等。这种技术称为嵌入式 SQL[2,3],它已经成为一种标准,被写入 SQL 标准文档中,如 SQL-92[2], SQL-99[3]等。

本文所描述的嵌入式 SQL 系统(ESQL)的实现允许把 SQL 语句嵌入到 C 语言中,这种语言称为 ESQL 语言。用 ESQL 语言编写的应用程序能够综合 C 语言的过程性结构和 SQL 的高效查询优势对数据库进行操作。而且,它还能够与 C 语言编写的图形界面结合使用,开发用户界面友好的应用软件。几乎所有的关系数据库系统都对 ESQL 提供了某种程度的支持。例如, PostgreSQL 数据库管理系统[4,5]的嵌入式 SQL。但它只基本符合 SQL-92 标准的入门级要求。而本文所讨论的 ESQL 可以达到 SQL-92 标准的中间级要求。

ESQL 是一个对象关系型数据库管理系统(ORDBMS) OSCAR 的组成部分。OSCAR 基于 Client/Server 架构实现,服务器端除涵盖通常数据库管理系统的一般功能外,还支持工程数据管理功能;客户端在提供各种通用应用开发接口的基础上,还具有丰富的连接、操作和配置服务器端的能力。客户端支持 ESQL, JDBC 和 ODBC 等,分别为用户提供通过 ESQL 语言、Java 语言和数据库开放互连标准访问 OSCAR 数据库管理系统的方法。

收稿日期: 2004-03-21; 修返日期: 2004-05-07
基金项目: 国家“863”计划资助项目(2003AA4Z3010)

2　总体设计

ESQL 分为两部分实现: 预处理模块和 ESQL 运行库。预处理模块包括词法分析子模块(Lexical Analysis SubModule, LASM)和语法分析子模块(Syntax Analysis SubModule, SASM),它识别 ESQL 语句并转换为宿主语言。ESQL 运行库实现与服务器端的交互。

预处理模块读入并分析 ESQL 语言编写的源文件,识别语句,并转换为相应功能的 C 语句,输出到 C 源文件。其中, LASM 进行词法分析,把 ESQL 源文件作为输入流,识别关键字、标识符、C 变量等,并将识别出的标识符输出给 SASM。SASM 接收 LASM 识别的标识符,并且根据嵌入式 SQL 语言给出的语法规则对 ESQL 语句进行语法检查与语义分析,将 ESQL 语句转换为 C 语言的变量声明或者对应的函数调用。预处理过程完成后,对于 ESQL 语言的处理工作就结束了。

ESQL 运行库通过远程过程调用(Remote Procedure Call, RPC)[6]机制实现应用程序和服务器端的信息交互,其中应用程序作为客户端。ESQL 运行库实现客户端通过 RPC 调用服务器端过程的各类函数。这些函数的功能包括: 通过网络通信

库建立和服务器端的连接; 根据 ESQL 语句构造请求, 并按照客户端和服务器端的预定义协议把纯 SQL 语句发送到服务器端; 服务器端完成请求的执行后, 通过网络通信库把执行结果返回给客户端。ESQL 系统信息流如图 1 所示。

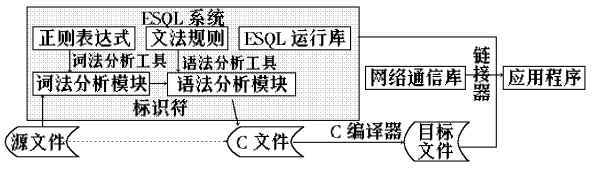

图 1 ESQL 系统信息流

图 2 说明了 ESQL 应用程序和服务器端之间的客户端和服务器端(C/S) 关系。其中向右的箭头代表 ESQL 应用程序取得客户请求并通过 ESQL 运行库和网络通信库发送给服务器端的过程; 向左的箭头代表应用程序从服务器端收取执行结果的过程。

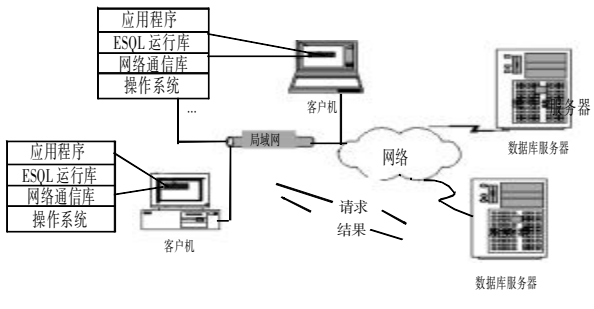

图 2 ESQL 系统与服务器端交互的物理架构

3 预处理模块设计

预处理模块提供嵌入式 SQL 预编译器, 它将含有 ESQL 语句的 ESQL 源程序转换为仅含有纯 C 语句的 C 文件。预处理模块识别 C 语句和 ESQL 语句, 对 C 语句不做改动; 对 ESQL 语句, 转换为完成 ESQL 语句功能的 C 语句。ESQL 处理两类语句: 一类是与 SQL 语句对应的语句; 另一类是 ESQL 语言独有的, 它们是为了与宿主语言或者服务器端交互而加入的语句。预处理模块对 ESQL 语句进行语法分析, 记录输入和输出主变量信息, 最后输出对 ESQL 运行库接口函数的调用。

3.1 LASM 的设计

借助词法分析工具, 能够灵活、高效地设计标识符解析程序。LASM 采用了 Flex[7] 词法分析工具。根据 ESQL 语言给出的正则表达式, 词法分析工具生成解析 ESQL 源文件的程序, 它把 ESQL 源文件作为输入流, 按顺序读取并识别源文件中的各种标识符, 并对它们进行归类。通过词法分析, ESQL 源文件被分解成独立的标识符流。

LASM 实现了完成标识符识别工作的确定有限自动机(Deterministic Finite Automation, DFA) [7], 它包括两个核心状态: C 状态和 SQL 状态, 分别处理 C 语言和 SQL 语言的语句。有限自动机初始处于 C 状态, 接收到 SQL 起始符号, 则进入 SQL 状态。在 SQL 状态下读到 SQL 语句的结束符号, 则返回 C 状态。此外, 还有一些状态用于识别 ESQL 语言中若干类别的标识符, 或对词法分析过程进行控制。所有的标识符主要分成以下类别: 关键字、常量字符串、标识符、操作符、主变量等。

图 3 中箭头表示读入特定的字符串之后, 进入或者退出相应的状态, 指向自身的箭头表示读入本状态可接收的字符。例如, 在 SQL 状态下接收到字符串{ b |B} '(比特流起始符号) , 则进入 xb 状态。它用来识别 SQL 中的比特流(Bit Stream) 。接下来的非单引号字符都将作为比特流, 直到读入下一个单引号字符(比特流结束符号) , 才返回到 SQL 状态。状态 xd, xh 用于识别 SQL 中的字符串和十六进制数。状态 xq, xd, xdc 用于识别 SQL 或者 C 中的常量字符串。

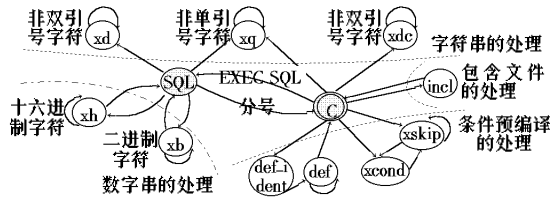

图 3 用于 ESQL 词法分析的有限自动机

在状态 C 下, 接收到字符串 EXEC SQL INCLUDE, 词法分析程序将打开指定的文件, 并切换到该文件继续进行词法分析。所包含的文件分析完成后, 退回到原文件继续分析过程。在状态 C 下, LASM 识别并处理 ESQL 语言的条件预编译相关语句。根据相关布尔表达式的真值决定忽略(转移到状态 xskip) 还是分析预编译语句后的内容(转移到状态 C) 。在状态 xskip 下, 忽略所有的输入, 直到下一个条件预编译语句。包含文件和条件预编译的处理可以嵌套。

3.2 SASM 的设计

借助语法分析工具, 能够灵活、高效地设计语法解析程序。SASM 采用了 BISON[7] 语法分析工具, 它所对应的文法是 LALR(1) 文法, 即向前看一个标识符的左递归文法。SASM 从 LASM 接收标识符, 根据 ESQL 语言给出的语法规则对 ESQL 语言的 ESQL 语句和 C 语句进行语法检查, 并进行语义分析。它将 ESQL 语句转换为 C 语言的变量申明或者对 ESQL 库函数的调用, 并且输出到目标 C 文件中; 将 C 语句原样输出到目标 C 文件中。ESQL 设计的难点在于 SQL 语句和 C 语句的信息交互。交互的内容包括主变量[2, 3] (SQL 语句和 C 语句交互的中介) 的值、SQL 语句执行返回的 SQL CODE 和 SQL STATE。

SASM 构造的状态机识别 ESQL 语言支持的主变量定义语句、可执行语句、连接语句、描述符语句、大对象语句、C 语句等。除了可执行语句之外, 其他语句只需简单地根据语法规则识别语句, 并输出对应的处理函数到目标 C 文件中。而可执行语句则需要更复杂的操作。

3.2.1 主变量定义

在程序中, C 语句与 SQL 语句通过主变量进行信息交互, 如通过主变量得到服务器端执行 SELECT 语句的结果值。主变量定义在 DECLARE 区间中, SASM 将它们的信息保存在符号表中。对于未定义在 DECLARE 区间的 C 变量, 语法分析模块作为一般 C 语句对待, 不存储 C 变量信息。

主变量定义语句的语法分析关键是获取变量的数据类型信息。在匹配变量定义语法规则的过程中, LALR(1) 文法从左到右移入标识符, 匹配语法规则, 进行归约, 同时进行语义分析。因为变量定义时类型总是在变量名的左边, 所以语法分析函数首先获得类型信息, 包括类型名、类型长度等。语义分析模块必须暂存类型信息。在读入右边的变量名时, 根据该信息构造变量的信息结构, 记录变量名、类型名、聚集类型变量的大小、类型尺寸、数组维数等信息。变量类型分为三种, 分别是结构体/联合体类型、数组类型和简单数据类型。创建变量的信

息结构后, 如果变量所在的结构层次不为 0(结构层次初始化为 0, 每次进入结构定义就增 1, 退出则减 1) , 那么说明当前解析的变量是结构体或者联合体的成员, 把它添加到结构变量的成员链表中; 否则说明是一个新定义的变量, 把它作为一个主变量保存到符号表中。

在语法解析过程中遇到对主变量的访问时, 首先确认该变量已经定义, 并将它添加到访问该变量的 ESQL 语句的参数列表中。

3.2.2　可执行语句

可执行语句的处理关键在于如何与 C 语句进行信息交互, 例如, 向 SQL 语句传递输入主变量值, 把执行结果从结果集赋值给输出主变量, 根据 SQL 语句的执行状态(SQL Code 和 SQL State) 判断程序的执行分支等。前两个功能由输入、输出主变量完成。前者从 C 语言向 ESQL 语言传递主变量值; 后者反之。第三个功能通过 SQL 通信区完成, 见第 5 节。

输出主变量的作用是存储 SQL 语句执行的结果, 在处理含输出主变量的语句时, 只需按它们的出现顺序记录语句引用的输出主变量; 输入主变量向 ESQL 运行库传递 SQL 语句中的输入值, 所以处理含输入主变量的语句时, 除了按它们的出现顺序记录语句引用的输入主变量外, 还需要把语句中的输入主变量用主变量占位符替换(即记录输入主变量所在的位置) , 以便 ESQL 运行库根据输入主变量的值构造纯 SQL 语句。在目标 C 文件中, 除了输出 SQL 命令(含主变量占位符) 外, 还需要把记录的输入和输出主变量信息作为可执行语句处理函数 ESQLdo( ) 的参数输出。ESQL 运行库从输入主变量地址读取输入主变量的值, 并按顺序在 SQL 语句的主变量占位符处代入这些值, 构造正确的可执行语句; 而在接收到执行结果时, 把结果值依次保存在输出主变量中。

对每一个主变量, 可以相应定义一个指示变量。输入主变量的指示变量可以给主变量赋值为 NULL。输出主变量的指示变量可以用来监测返回结果, 如是否为空、字段是否被截断等。指示变量只能是整型变量, 而且同样需要在 Declare 区间申明。

4　ESQL 运行库设计

经过 LASM 和 SASM 的处理, 用 ESQL 语言写的应用程序已经转换为纯 C 应用程序, 它调用 ESQL 系统的 ESQL 运行库实现的函数。这些函数实现网络连接、事务处理、构造 SQL 语句并发送给服务器端、从服务器端得到执行结果等功能。

ESQL 运行库实现的函数分为两类: ①处理服务器端支持的大部分 SQL 语句的构建和发送, 并接收和解析执行结果; ②处理 ESQL 系统相关的语句, 如连接管理、描述符处理和大对象处理等。连接管理主要维护连接信息: 在创建一个新的连接时加入新连接的信息; 断开连接时, 删除连接信息。连接通过系统的网络通信库实现, 网络通信库按照预定义的协议进行通信。

SASM 把大多数可执行 SQL 语句转换为 ESQLdo( ) 函数, 并输出到目标 C 文件中。ESQL 运行库的核心工作就是实现该函数的功能。

在执行语句前, 必须确认与服务器端已建立相关连接, 然后初始化 SQL 通信区(用于存储执行结果状态) , 根据 ESQL-do( ) 函数的参数列表中 SQL 语句内容(含主变量占位符) 和输入主变量列表构造纯 SQL 语句。

构造纯 SQL 语句的过程如下: ①在 SQL 语句(含主变量占位符) 中查找主变量占位符; ②从参数列表中输入主变量地址处取出输入主变量的值, 按照参数列表的顺序替换主变量占位符; ③重复第①步和第②步, 直至最后一个主变量占位符处理完, 并且输入主变量列表刚好耗尽。

同时, 按照在参数列表中出现的顺序存储输出主变量列表, 用于得到 SQL 语句的执行结果时存储结果值。

完成纯 SQL 语句的构造后, 应用程序调用网络通信库的接口函数发送该语句到服务器端。服务器端执行 SQL 语句, 并返回执行结果。ESQL 运行库在输出主变量中存储得到的查询结果。

存储结果值时, 有一个特殊的输出变量——描述符变量, 它不是 C 语言支持的输出主变量, 而是根据应用程序的 ESQL 语句请求在 ESQL 系统内部分配。该变量指向服务器端返回的结果值存放的内存空间地址, 代表一个结果集合。应用程序不能按照与一般的输出主变量相同的方式读取, 所以需要提供 GET DESCRIPTOR 语句从描述符变量得到结果值, 根据描述符变量中请求的格式获取结果表的属性信息, 如属性名、属性值、属性类型等, 最后存储到 GET DESCRIPTOR 语句指明的输出主变量中。这些输出主变量必须已经在 DECLARE 区间声明。

5　错误处理机制

ESQL 具备一个精巧的错误处理机制, 对 ESQL 语句进行过程性的控制。根据执行 SQL 语句返回的 SQLSTATE, 应用程序可以按照用户需求设置不同的控制流。该机制需要预处理模块和 ESQL 运行库协同工作实现。

SASM 识别一个 WHENEVER 语句时, 记录错误条件和相应的错误处理动作信息。在输出语句时, 对那些根据执行结果的错误码进行处理的语句, 设置特殊的输出模式, 于是, 在输出 ESQLdo( ) 函数到目标文件时, 预处理程序会在该语句后附加判断错误码的语句, 若符合相关错误条件, 就执行相应的错误处理动作。WHENEVER 语句的作用域从该语句出现之处到下一次 WHENEVER 语句用相同的错误条件设置不同的错误处理动作之处。在其作用域内, 所有相关 SQL 语句后都会附加错误码判断和错误处理语句。

在 ESQL 运行库中定义了一个 SQL 通信区(SQLCA) , 用来存放 SQL 语句执行的结果信息, 如错误码、错误消息、处理元组标识、处理元组数等。在语句执行出现异常时, 根据 SQL 语句执行结果设置 SQLCA 的相应域, 并记录 SQL CODE 和 SQL STATE。SASM 附加的谓词就会为真, 应用程序转而执行 WHENEVER 指定的错误处理动作。

6　结论

本文给出了一个嵌入式 SQL 系统的设计和实现。该系统达到 ANSI/ISO 定义的 SQL-92 标准的中间级, 通过了 SQL-92 测试用例集[8] 的测试, 并且已经作为对象关系型数据库管理系统 OSCAR 客户端的重要组成部分, 成功地应用 (下转第 218 页)