# 嵌入式数据库中利用Lex,Yacc设计SQL编译器

夏 铭,陆 阳,盛业兴,李大勇
(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

**摘 要**:随着嵌入式数据库系统的不断发展,对其数据查询的能力提出了新的要求。基于自行设计的嵌入式数据库系统,利用Lex和Yacc工具实现了嵌入式SQL编译器的设计,进一步对所生成的语法树进行了初步的优化研究。结果表明此编译器大大增强支持SQL查询语言的能力。

**关键词**:嵌入式数据库; Lex; Yacc; SQL
**中图分类号**:TP314 **文献标识码**:A **文章编号**:1673-629X(2007)11-0121-04

# Design of SQL Compiler with Lex and Yacc Based on Embedded Database

XIA Ming, LU Yang, SHENG Ye-xing, LI Da-yong
(College of Computer and Information, Hefei University of Technology, Hefei 230009, China)

**Abstract**: Subsequent with the development of embedded database system, improvements in data query are urgently required. Based on the embedded database system have designed, the SQL complier is successfully implemented using tools of Lex and Yacc. Furthermore, preliminary optimization researches have been proposed on the generated syntax analysis tree. The result indicates that the compiler remarkably reinforces the SQL query capacities.

**Key words**: embedded database; Lex; Yacc; SQL

## 0 引 言

评价嵌入式数据库的关键是它支持SQL语言的程度,一个高效安全的SQL编译器是一个优秀嵌入式数据库的核心和基础。SQL作为关系数据库语言的国际标准,向用户提供了一个易于使用的、非过程化的、描述性的语言来存取数据库中的数据。一般的数据库系统应该能提供一套查询系统,或支持SQL92标准的子集,查询处理实现数据库系统中最基本、最常用的一些查询操作,并且具备较高的执行效率。但对于SQL这样复杂庞大的语法,在嵌入式数据库中为之构建完善且可靠的词法分析器和语法分析器,具有一定的难度和工作量,即使是SQL的子集。Lex和Yacc作为优秀的编译器自动生成工具,适合于各种编译器的实现。通过利用Lex和Yacc工具,可以方便地写出SQL的词法和语法分析脚本,生成一个集所有SQL信息的语法树。这样就可以生成一个SQL编译器。

收稿日期:2007-02-09

作者简介:夏 铭(1979-),女,安徽庐江人,合肥工业大学硕士研究生,安徽工业学院教师,研究方向为嵌入式系统应用;陆 阳,教授,博士生导师,研究方向为计算机控制、嵌入式系统应用、现场总线技术。

探讨了如何在嵌入式ARM[1]开发环境中,以Lex和Yacc语言为基础,利用编译程序构造工具FLex和Bison Yacc自动生成完成嵌入式数据库中SQL查询的词法和语法分析器。与直接写出SQL查询编译器相比,自动生成的SQL编译器具有功能完善、简洁明了、易于修改和扩充等突出特点。下面将具体介绍Lex和Yacc的基本原理以及实现嵌入式SQL编译器的词法语法分析,最后针对生成的SQL语法树提出适用于嵌入式系统的优化策略。

## 1 编译器设计原理

### 1.1 Lex和Yacc的基本原理

1)Lex词法分析工具是根据正规式[2]和确定有限状态自动机原理(DFA),对所输入的一些字符进行线性扫描分析,产生一个个单词符号,并返回单词符号的类型码。一个确定有限状态自动机$M$是一个五元组

$$M=(S,\sum,\delta,S0,F)$$

其中

(1)$S$是一个由有限个状态组成的集合;

(2)$\sum$是一个有穷字母集,它的每个元素称为输入字符;

(3)$\delta$ 是一个从 $SX\sum$ 至 $S$ 的单值部分映射，如 $\delta(s,a)=S'$，其中 $s\in S, a\in\sum$，意味着当现在状态为 $s$，输入字符为 $a$ 时，将转换到状态 $S'$；

(4)$S0\in S$ 为初始状态；

(5)$F\in S$ 是一个终止状态。

有限自动机对输入的字符进行扫描，根据读入进行状态转移，从而分析出所有单词符号。用 Lex 工具生成词法分析器，首先要写出其输入文件，经过 Lex 编译程序生成词法分析器，然后利用所生成的词法分析器对所输入的串进行词法分析，输出单词符号和类型码，供 Yacc 使用。

2)字面上理解 Yacc 是一个编译程序的编译程序，但严格说它不是编译程序自动产生器，它不能产生一个完整的编译程序。Yacc 根据上下文无关文法产生式，即 BNF 范式，利用自下而上的语法分析方法，自动构造一个语法分析器，能够利用 Lex 词法分析的结果，将匹配的表达式转化后放到堆栈。词法分析是一种简单、线性分析，而语法分析是一种层次结构的分析。用 Yacc 工具来生成语法分析器，首先要根据 BNF 范式编写 Yacc 的输入程序，然后经过 Yacc 程序，生成语法分析器，最后利用词法分析工具，通过调用 yylex( )，进行词法分析的同时，进行语法分析和生成语法树。

语法规则部分是 Yacc 输入文件的核心部分，它直接关系到语法规则的定义，其格式采用 BNF 范式。子程序部分主要是要编写一些 C 程序可以供动作调用，另外还提供一个语法分析控制器 yyparse( )，该程序是语法分析的入口。

### 1.2 嵌入式 SQL 编译器原理

编译器主要划分成四个部分：词法分析、语法分析、目标代码生成和出错处理。编译器输入嵌入式数据库 SQL 查询语句，输出目标代码，在本系统中就是提供调用数据库处理函数的参数及数据。目标代码是一个存放这些数据的树结构。本系统采用一遍扫描方式，以词法分析程序和语法分析程序为核心，出错处理作为一个独立的过程。

编译器最重要的部分就是词法分析程序和语法分析程序，常用的有 UNIX 系统的实用工具 Lex&Yacc，运行于 MS－DOS 或 OS/2 环境的 MKS 发布的 Lex&Yacc，以及基于 Windows 操作平台的 Parser Generator。这些工具在嵌入式环境中使用有所限制。嵌入式数据库在基于 ARM 的开发板 EasyARM2200 上实现，支持 μClinux[3] 或 μC－OSⅡ操作系统，由于 μClinux 在嵌入式设备中的广泛应用以及还没有能在 μC－OSⅡ系统中适用的词法语法分析程序，因此采用 GNU 发布的 FLex&Bison。GNU 是自由软件基金会的工程，它将 Yacc 的名称替换成 Bison，由于它是免费的，每个人都可以使用 Bison。

本编译器中这两个模块的设计采用 GNU 中的构造工具 FLex 和 Bison 完成。首先根据符合 SQL 标准的程序字和词法、语法规则编制 Lex 和 Yacc 源文件，通过 FLex 和 Bison 自动生成以 C 语言为代码的词法分析程序 yylex 和语法分析程序 yyparse，然后调用输入的 SQL 查询语句，执行两个分析程序，得到目标代码即语法树[4]供数据库函数调用。整个编译器设计原理如图 1 所示。

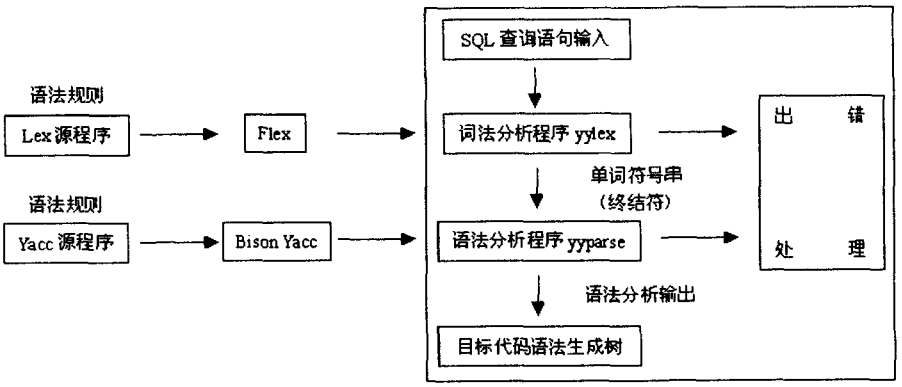



图 1 编译器设计原理图

### 1.3 嵌入式 SQL 编译器设计

(1) 词法分析。

词法分析的主要任务是从左到右按字符逐个读入 SQL 查询语句，对其中的字符流进行扫描和分解，从而识别出各个单词符号。当词法分析程序在分析过程中发现词法错误，则转入出错处理。词法分析程序实现的基础是 Flex，它接受以 Lex 语言为基础的表示某种语言单词集的正规式描述，生成一个能够识别该语言的词法分析程序。以下给出简要的 Lex 输入程序，取名为 sql_scanner.l。

```
%{
digit [0-9]
alpha [a-zA-Z]
alnum [a-zA-Z0-9]
%}
%%
SELECT {yylval.str=yytext;return(SELECT);}
";" {yylval.str=yyetxt;return(ENDMARK);}
.....
{alpha} {alnum}* {yylval.str=yytext;return (ID);}
```

```
{digit}+{yylval.str=yytext;return(ICON);}
{alpha}{alnum}*[.]{alpha}{alnum}*{yylval.str=yytext;return
(FID);}
{alnmu}+{yylval.str=yytext;return(STRI);}
[/t/n];
%%
```

(2) 语法分析。

词法分析程序把 SQL 查询语句转换为记号,即所有的关键字、操作符、分隔符、标志符和变量。但这些记号的序列是否符合语法,由语法分析程序来确定。语法分析程序比较复杂,这里只给出构造程序的步骤。

a. 根据 Lex 返回字声明终结符。

```
%token ALL ANY AS BETWEEN BY
……
%token IS KEY LIKE OF ON
……
```

b. 确定识别的语法规则。

```
Sql_list:
    Sql ';'
  | sql_list sql ';'
  ;……
sql: schema
;
  ……/*模块语言*/
sql: module_def
  ;
  ……/*操纵语句*/
sql: manipulative_statement
  ;
```

c. 根据识别的语法规则给出相应的语义动作,在本系统中语义动作有两类:一是当查询语句中的指令与语法规则不符时,给出语法错误信息并转入错误处理;二是识别到各指令的信息时,将这些信息值转换输出为目标代码[5]。

(3)目标代码的语法树生成。

语法树是在对语句进行语法分析的同时生成的,所以在 Yacc 输入文件中必须要确定语法树的结构、生成结点程序的说明。

语法树结点结构:

| 结点类型 | 结点字符 | 左指针 | 中间指针 | 右指针 |
|---|---|---|---|---|

```
struct sstree
{
int ss_type;
char *ss_char;
struct sstree *ss_left;
struct sstree *ss_mid;
struct sstree *ss_right;
}
```

生成语法树结点的子程序如下:

```
struct sstree *builtnode(sstype,sschar,ssleft,ssmid,ssright)
{
struct sstree *sspoint;
sspoint=(struct sstree*)malloc(sizeof(struct sstree));
if(sspoint==NULL)yyerror();
sspoint->ss_char=sschar;
sspoint->ss_type=sstype;
sspoint->ss_left=ssleft;
sspoint->ss_right=ssright;
return(sspoint);
}
```

利用查询语句的 BNF 范式和调用语法树结点生成子程序,可以很方便地写出 Yacc 的输入程序,取名为 sql_yacc.y,Flex 读取 sql_scanner.l 自动构造词法分析器,Bison 读取 sql_yacc.y 自动构造语法分析器,接受用户命令请求提交给分析模块,进行词法、语法分析和预处理即可生成语法树。

## 2　SQL 语法树的优化

高效率、低消耗是每个数据库系统[6]的基本要求。为了提高数据库系统的性能,必须在执行查询计划之前对语法树优化。在关系数据库[7]中,能够将语法树转化成一个表达式,这个表达式可用一个扩展的关系代数操作符表示。如果将关系代数中一些有用的代数定律应用到嵌入式数据库中,通常能够有效地提高查询计划。下面的方法是根据关系代数定律理论[8]以及相关实践得到的:

①查询尽可能的转换为单个的 SELECT 语句,如果有条件查询则把查询条件独立出来形成新的查询,避免两层嵌套;

②在转换查询的时候,考虑先做选择投影,再做连接等其他二元操作;

③连接时,应先做小关系的连接,再做大关系的连接。

如有关系:

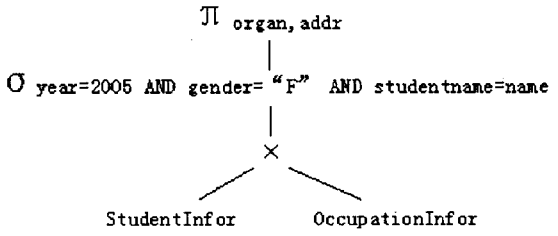
StudentInfor(name,birthdate,addr,gender)

OccupationInfor(organ,year,studentname)

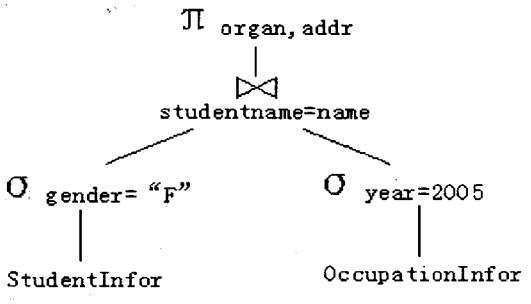
在这两个关系中查询在 2005 年毕业的女学生的就业单位和家庭出生地;

```
SELECT organ,addr
FROM StudentInfor, OccupationInfor
WHERE year=2005 AND
  gender="F" AND
  studentname=name;
```

在查询编译阶段完成后的语法树如下:




经过优化后的查询语法树为：




优化过程中，首先在可以进行选择时，倾向于连接而非乘积；其次 WHERE 子句的其他两个条件被拆分到两个选择 σ 操作中，操作被下推到各自树的相应关系上。对比发现，优化后的语法树能够节约使用的存储空间。

## 3 结束语

讨论了 Lex 和 Yacc 的基本工作原理。将它们应用到嵌入式系统中，使得支持 SQL 语言的程度大大提高，并针对嵌入式数据库的特点，作了初步的优化工作。但要真正构造一个实用高效的 SQL 编译器，还有包括 SQL 子集的详细分析、二义性的处理、语法树的进一步优化等。

**参考文献：**

[1] 周立功. ARM 嵌入式系统基础教程[M]. 北京：北京航天航空大学出版社，2005：132－188.
[2] Levine J R. Lex 与 Yacc[M]. 第 2 版. 杨作梅译. 北京：机械工业出版社，2003：122－163.
[3] 周立功，陈明计，陈 渝. ARM 嵌入式 Linux 系统构建与驱动开发范例[M]. 北京：北京航空航天大学出版社，2005：71－101.
[4] 姚泽勤，柏又青，马建峰. 利用 Lex 及 Yacc 实现嵌入式 SQL 分析器[J]. 航空计算技术，2002(3)：55－58.
[5] 沙智华，葛研军，施志辉. 基于 Lex&Yacc 数控代码编译技术研究[J]. 组合机床与自动化加工技术，2002(11)：49－51.
[6] 王晓东，曹庆华，王 卓. DB2 数据库查询优化策略[J]. 现代电子技术，2006(10)：92－95.
[7] 谷震离. 关系数据库查询优化方法研究[J]. 微计算机信息，2006(22)：161－164.
[8] 王振辉，吴广茂. SQL 查询语句优化研究[J]. 计算机应用，2005(12)：207－209.

(上接第 117 页)

**参考文献：**

[1] 冯俐俐，李昌禧. 指纹中心点的定位和特征匹配方法[J]. 华中科技大学学报：自然科学版，2002(10)：78－80.
[2] 谭台哲，宁新宝，尹义龙，等. 一种基于指纹中心点的匹配算法[J]. 南京大学学报：自然科学版，2003(4)：483－490.
[3] 朱 宁，施荣华，吴科桦. 一种新的点模式指纹匹配方法[J]. 计算机工程与应用，2006(5)：74－76.
[4] 赵 娟，王典洪. 指纹图像匹配的算法研究及其实现[J]. 计算机工程与应用，2005(13)：66－69.
[5] 张洪光，刘雪梅. 指纹识别中的一种向量匹配算法[J]. 计算机工程，2002，28(4)：106－108.
[6] 李志敏，彭志刚. 基于动态全局特征的指纹匹配算法的研究[J]. 沈阳化工学院学报，2000，14(4)：292－295.
[7] 罗西平，田 捷. 自动指纹识别中的图像增强和细节匹配算法[J]. 软件学报，2002，13(5)：942－956.

(上接第 120 页)

系数、多项式核的指数以及核函数是值得进一步研究的问题。

**参考文献：**

[1] Kim K I, Park S H, Kim H J. Kernel Principal Component Analysis for Texture Classification[J]. IEEE signal processing letters, 2001, 8(2)：39－41.
[2] Haykin S. 神经网络原理[M]. 叶世伟，史忠植，译. 北京：机械工业出版社，2004.
[3] Vapnik V N. 统计学习理论的本质[M]. 张学工译. 北京：清华大学出版社，2000.
[4] 邓乃扬，田英杰. 数据挖掘中的新方法：支持向量机[M]. 北京：科学出版社，2004.
[5] 张敏贵，潘 泉，张洪才，等. 基于支持向量机的人脸分类[J]. 计算机工程，2004，30(11)：110－112.
[6] 何国辉，甘俊英. 基于核主元分析和支持向量机的人脸识别[J]. 计算机工程与设计，2005，26(5)：1190－1193.
[7] 黄 勇，郑春颖，宋忠虎. 多类支持向量机算法综述[J]. 计算技术与自动化，2005，24(4)：61－63.
[8] 武方方，赵银亮. 一种基于 Morlet 小波核的约简支持向量机[J]. 控制与决策，2006，21(8)：848－852.