

编译原理上机报告

《DBMS 的设计与实现》

www.docin.com

完成时间：2012 年 5 月 26 日

目 录

1. 项目概况	3
1.1 基本目标	3
1.2 完成情况	3
1.3 逻辑结构与物理结构（由于语义制导没做，所以这部分没写）	6
1.4 语法结构与数据结构（由于语义制导没做，这里只给出语法结构）	6
1.5 执行流程	7
1.6 功能测试	8
2. 总结与未来工作	9
2.1 未成功能	9
2.2 未来实现方案	9

www.docin.com

1. 项目概况

1.1 基本目标

本次项目主要的目标是 DBMS 的设计与实现，即完成 SQL 语句的语法制导解释，通过这次项目，加深对《编译原理》课程的理解，掌握词法与语法的制导过程，熟悉 Lex 和 Yacc 的使用。

1.2 完成情况

目前完成的功能是 SQL 词法分析器和语法分析器，下面介绍一下具体的实现情况。

1. 完成的语句：

```
CREATE DATABASES x
CREATE TABLE x
SHOW TABLES
DROP TABLES
INSERT INTO x VALUES x
SELECT x FROM x
SELECT x FROM x WHERE x
SELECT x FROM x GROUP BY x
DELETE FROM x WHERE x
UPDATE x SET x WHERE x
```

2. 识别出的关键字如下：

```
ID NUMBER CREATE TABLE CHAR INT SELECT GROUP FROM WHERE AND OR DROP SHOW
INSERT INTO BY VALUES DATABASE TABLES UPDATE DELETE SET OR AND
```

3. 识别出的符号如下：

<	>	=	()
!	,	;	*	.

4. 识别标示符的说明:

[0-9]的 NUMBER, [0-9][a-zA-Z]的字符型。

5. lex 进行的词法分析如下:

```
%{
#include "myparser.h"
}%

char [a-zA-Z_]
digit [0-9]
digits {digit}+
optional_fraction ("."{digits})?
optional_exponent (E[+]?{digits})?

%%

[ ]+ ;
{digits}{optional_fraction}{optional_exponent} {return NUMBER;}
/////SQL 关键字的识别/////
"CREATE"    {return CREATE;}
"DATABASE"  {return DATABASE;}
"TABLE"     {return TABLE;}
"GROUP"     {return GROUP;}
"BY"        {return BY;}
"CHAR"      {return CHAR;}
"INT"       {return INT;}
"SELECT"    {return SELECT;}
"FROM"      {return FROM;}
"WHERE"     {return WHERE;}
"AND"       {return AND;}
"OR"        {return OR;}
"DROP"      {return DROP;}
"SHOW"      {return SHOW;}
"INSERT"    {return INSERT;}
"UPDATE"    {return UPDATE;}
"DELETE"    {return DELETE;}
"TABLES"    {return TABLES;}
"INTO"      {return INTO;}
"VALUES"    {return VALUES;}
"SET"       {return SET;}
{char}({char}|{digit})* {return ID;}
/////对特殊符号的识别
"<"        {return '<';}
```

```

">"    {return '>';}
"="     {return '=';}
"!"     {return '!';}
"("     {return '(';}
")"     {return ')';}
";"     {return ';';}
";"     {return ';';}
"*"     {return '*';}
"."     {return '.';}
%%

```

6. yacc 进行的语法分析如下:

```
statements : statements statement|statement;
```

```
statement : createdatabase | createtable | selectsql | droptable | showtables | insertsql | deletesql | updatesql;
```

对于所给语句分别写出它们的文法如下:

CREATE DATABASE 的文法:

```
createdatabase:CREATE DATABASE database ';' {printf("create database!");};
```

```
database:ID;
```

CREATE TABLE 的文法:

```
createtable : CREATE TABLE table '(' fieldsdefinition ')' ';' {printf("create table!\n");};
```

```
table : ID;
```

```
fieldsdefinition : field_type | fieldsdefinition ',' field_type;
```

```
field_type : field type;
```

```
field : ID;
```

```
type : CHAR '(' NUMBER ')' | INT;
```

SELECT 的文法:

```
selectsql : SELECT fields_star FROM tables ';' {printf("you use SELECT!\n");}
```

```
          | SELECT fields_star FROM tables WHERE conditions ';' {printf("you use SELECT!\n");}
```

```
          | SELECT fields_star FROM tables GROUP BY fields_star ';' {printf("you use SELECT!\n");};
```

```
fields_star : table_fields | '*';
```

```
table_fields : table_field | table_fields ',' table_field;
```

```
table_field : field | table '.' field;
```

```
tables : tables ',' table | table;
```

```
conditions : condition | '(' conditions ')'
```

```
          | conditions AND conditions | conditions OR conditions;
```

```
condition : comp_left comp_op comp_right;
```

```
comp_left : table_field;
```

```
comp_right : table_field | NUMBER;
```

comp_op : '<' | '>' | '=' | '!' | '!=';

DROP 语句的文法:

droptable : DROP table ';' {printf("you drop table!\n");};

SHOW TABLES 语句的文法:

showtables : SHOW TABLES ';' {printf("show all tables!\n");};

INSERT 语句的文法:

insertsql : INSERT INTO table '(' fieldname ')' VALUES '(' fieldvalues ')' ';' {printf("insert data successfully!\n");};

 | INSERT INTO table VALUES '(' fieldvalues ')' ';' {printf("insert data successfully!\n");};

fieldname : field | fieldname ',' field;

fieldvalues : fieldvalue | fieldvalues ',' fieldvalue;

fieldvalue : ID | NUMBER;

DELETE 语句的文法:

deletesql : DELETE FROM table WHERE conditions ';' {printf("delete table!\n");};

UPDATE 语句的文法:

updatesql : UPDATE table SET condition WHERE conditions ';' {printf("update table!\n");};

7. 语义制导:

这部分目前还没有完成。

1.3 逻辑结构与物理结构（由于语义制导没做，所以这部分没写）

1.4 语法结构与数据结构（由于语义制导没做，这里只给出语法结构）

逐条一一说明增加 SQL 语句的语法结构与数据结构。语法结构用产生式说明，其中非终结符的属性用结构体表示，所用数据结构需要说明，并且最后用图形说明整个 SQL 语句的数据结构。

CREATE 语句的产生式语法结构:

createsql: CREATE TABLE table '(' fieldsdefinition ')' ';' ;

SELECT 语句的产生式语法结构:

```
selectsql : SELECT fields_star FROM tables ';'
          | SELECT fields_star FROM tables WHERE conditions ';'
          | SELECT fields_star FROM tables GROUP BY fields_star ';'

```

DROP 语句的产生式语法结构:

```
droptable : DROP table ';'

```

SHOW TABLES 语句的产生式语法结构:

```
showtables : SHOW TABLES ';'

```

INSERT 语句的产生式语法结构:

```
insertsql : INSERT INTO table '(' fieldname ')' VALUES '(' fieldvalues ')' ';'
          | INSERT INTO table VALUES '(' fieldvalues ')' ';'

```

DELETE 语句的产生式语法结构:

```
deletesql : DELETE FROM table WHERE conditions ';'

```

UPDATE 语句的产生式语法结构:

```
updatesql : UPDATE table SET condition WHERE conditions ';'

```

1.5 执行流程

逐条一一说明增加 SQL 语句的执行流程，并且配以流程图。流程中涉及到的函数需要对其名称、说明、输入参数、输出参数、执行流程进行一一说明。

函数名称: CreateValueIndex(int datasetID, int docID)

函数说明: 结合 XML 文档的 Schema 信息对一个给定文档上所有节点及属性值建立值索引

输入参数: datasetID dataset 编号, docID doc 编号

输出参数: 0 表示正确, 其他表示错误

执行流程: ? ? ?

1.6 功能测试

测试所实现的 SQL 语句的基本功能，框架如下：

测试 1

输入：CREATE DATABASE mydb;

输出：create database!

测试 2

输入：CREATE TABLE Student

(Sno CHAR(9), Sname CHAR(20), Ssex CHAR(2), Sage INT);

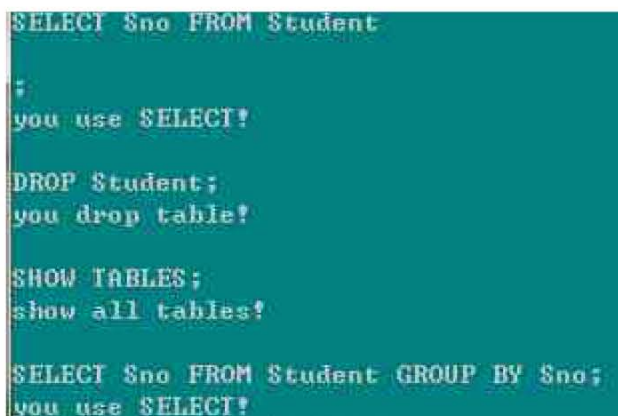
输出：create table!

测试 3

输入：SELECT Sno FROM Student;

输出：you use SELECT!

测试部分语句截图如下所示：



```
SELECT Sno FROM Student;
;
you use SELECT!

DROP Student;
you drop table!

SHOW TABLES;
show all tables!

SELECT Sno FROM Student GROUP BY Sno;
you use SELECT!
```


2. 总结与未来工作

2.1 未成功能

应该可以实现，但由于时间关系没有实现的功能。

2.2 未来实现方案

未成功能的预期设计实现方案概述。

www.docin.com