import ply.lex as lex

import re

from math import \*

import ply.yacc as yacc

global col\_lst

col\_lst=[]

#TOKENS

tokens=('SELECT','FROM','WHERE','ORDER\_BY','GROUP\_BY','HAVING','NAME','AND','OR','COMMA',

'EQUALS','CONDITION','SPACE','LP','RP','JOIN','AVG','BETWEEN','IN','SUM','MAX','MIN','COUNT','UNION','INTERSECT',

'EXCEPT','NUMBER','ATTR','NATURAL\_JOIN','INSERT','INTO','VALUES','DELETE','ANY','ALL','UPDATE','SET','INV','AS','DOT','DISTINCT')

literals = ['=','+','-','\*', '^','>','<' ]

#DEF OF TOKENS

def t\_LP(t):

r'\('

return t

def t\_JOIN(t):

r'JOIN'

return t

def t\_DISTINCT(t):

r'DISTINCT'

return t

def t\_DOT(t):

r'\.'

return t

def t\_AS(t):

r'AS'

return t

def t\_UPDATE(t):

r'UPDATE'

return t

def t\_SET(t):

r'SET'

return t

def t\_ANY(t):

r'ANY'

return t

def t\_ALL(t):

r'ALL'

return t

def t\_DELETE(t):

r'DELETE'

return t

def t\_INSERT(t):

r'INSERT'

return t

def t\_VALUES(t):

r'VALUES'

return t

def t\_INTO(t):

r'INTO'

return t

def t\_UNION(t):

r'UNION'

return t

def t\_INTERSECT(t):

r'INTERSECT'

return t

def t\_EXCEPT(t):

r'EXCEPT'

return t

def t\_SUM(t):

r'SUM'

return t

def t\_MIN(t):

r'MIN'

return t

def t\_MAX(t):

r'MAX'

return t

def t\_COUNT(t):

r'COUNT'

return t

def t\_AVG(t):

r'AVG'

return t

def t\_RP(t):

r'\)'

return t

def t\_BETWEEN(t):

r'BETWEEN'

return t

def t\_IN(t):

r'IN'

return t

def t\_SELECT(t):

r'SELECT'

return t

def t\_FROM(t):

r'FROM'

return t

def t\_WHERE(t):

r'WHERE'

return t

def t\_ORDER\_BY(t):

r'ORDER\_BY'

return t

def t\_GROUP\_BY(t):

r'GROUP\_BY'

return t

def t\_HAVING(t):

r'HAVING'

return t

def t\_OR(t):

r'OR'

return t

def t\_AND(t):

r'AND'

return t

def t\_COMMA(t):

r','

return t

def t\_NATURAL\_JOIN(t):

r'NATURAL\_JOIN'

return t

def t\_CONDITION(t):

r'[a-zA-Z0-9\_]+[\t]\*[=\+-><][\t]\*[a-zA-Z0-9\_]+[\t]\*[AND]\*[OR]\*'

return t

def t\_INV(t):

r'\"'

return t

def t\_NUMBER(t):

r'[0-9]+'

return t

def t\_NAME(t):

r'[A-Za-z]+|[a-zA-Z\_][a-zA-Z0-9\_]\*|[A-Z]\*\.[A-Z]$'

return t

def t\_ATTR(t):

# r'[a-zA-Z0-9\_]\.[a-zA-Z0-9\_]+'

r'([0-9]\*\.[0-9]+|[0-9]+)'

return t

# Ignored characters

t\_ignore = " \t"

def t\_error(t):

print("Illegal character '%s'" % t.value[0])

t.lexer.skip(1)

# LEXICAL ANALYSIS

lex.lex()

#PARSING GRAMMAR

def p\_query(t):

'''query : query1

| union

| groupquery

| LP query RP

| select

| insert

| delete

| update

| joinquery

'''

if len(t)==2:

t[0]=t[1]

else:

t[0]=t[2]

def p\_table(t):

'''table : NAME

| LP query RP

| NAME AS NAME

| table COMMA table'''

if len(t)==2:

t[0]=t[1]

elif t[2]=='AS':

t[0]='[RHO(%s){%s}]'%(t[3],t[1])

elif t[2]==',':

t[0]='%s X %s'%(t[1],t[3])

else :

t[0]=str(t[2])

def p\_query1(t):

'''query1 : SELECT list FROM table where'''

if len(t)==6 and t[2]!='':

t[0]='PI(%s){%s{%s}}'%(t[2],t[5],t[4])

elif len(t)==6 and t[2]=='':

t[0]='SIG(%s){%s{%s}}'%(t[2],t[5],t[4])

else:

t[0]='PI(%s){{%s}}'%(t[2],t[4])

def p\_where(t):

''' where : WHERE lst order

| order

| '''

if len(t)==3 and t[1]=='WHERE' :

t[0]='SIG(%s)'%(t[2])

elif len(t)==4:

#t[0]='SIG(%s){%s}'%(t[2],t[3])

t[0]='SIG(%s)'%(t[2])

else:

t[0]=''

def p\_select(t):

'''select : SELECT list from '''

t[0]='PI(%s){%s{%s}}'%(t[2],t[3],str(table))

def p\_from(t):

'''from : FROM list where

| FROM LP select RP where

'''

global table

if(len(t)==4):

t[0]=t[3]

table=t[2]

else:

t[0]=""+t[5]

table=t[3]

def p\_order(t):

''' order : ORDER\_BY list

|

'''

if(len(t)==3):

t[0]='ORDER\_BY(%s)'%(t[2])

else:

t[0]=""

def p\_groupquery(t):

''' groupquery : SELECT list FROM table where group '''

t[0]='{%s}GROUP\_BY[(%s){%s(%s)}]'%(t[6],t[2],t[5],t[4])

def p\_union(t):

''' union : query UNION query

| query INTERSECT query

| query EXCEPT query'''

if t[2]=='UNION':

t[0]='(%s) U (%s)'%(t[1],t[3])

elif t[2]=='INTERSECT':

t[0]='(%s) /\ (%s)'%(t[1],t[3])

else:

t[0]='(%s) - (%s)'%(t[1],t[3])

def p\_lst(t):

''' lst : condition

| condition AND condition

| condition OR condition

| NAME BETWEEN NUMBER AND NUMBER

| NAME IN LP query RP

| NAME '>' LP query RP

| NAME '<' LP query RP

| NAME '>' ANY LP query RP

| NAME '>' ALL LP query RP

| NAME '<' ANY LP query RP

| NAME '<' ALL LP query RP

| NAME '<' agg

| NAME '>' agg

| agg '>' NUMBER

| NAME '=' agg

| agg '=' NUMBER

| agg '<' NUMBER

'''

if len(t)==2:

t[0]=t[1]

elif t[2]==',':

t[0]='%s,%s'%(t[1],t[3])

elif t[2]=='AND':

t[0]='%s /\ %s'%(t[1],t[3])

elif t[2]=='BETWEEN':

t[0]='%s >= %s /\ %s <= %s'%(t[1],str(t[3]),t[1],str(t[5]))

elif t[2]=='IN':

t[0]='%s IN %s'%(t[1],str(t[4]))

elif t[2]=='<' and len(t)==4:

t[0]='%s < %s'%(str(t[1]),str(t[3]))

elif t[2]=='=' and len(t)==4:

t[0]='%s = %s'%(str(t[1]),str(t[3]))

elif t[2]=='>' and len(t)==4:

t[0]='%s > %s'%(str(t[1]),str(t[3]))

elif t[2]=='>' and t[3]=='ANY':

t[0]='%s > %s'%(t[1],str(t[5]))

elif t[2]=='>' and t[3]!='ANY':

t[0]='%s > %s'%(t[1],str(t[4]))

elif t[2]=='<' and t[3]!='ANY':

t[0]='%s < %s'%(t[1],str(t[4]))

elif t[2]=='<' and t[3]!='ANY':

t[0]='%s < %s'%(t[1],str(t[4]))

elif t[2]=='>' and t[3]=='ALL':

t[0]='%s > %s'%(t[1],str(t[5]))

elif t[2]=='>' and t[3]!='ALL':

t[0]='%s > %s'%(t[1],str(t[4]))

elif t[2]=='<' and t[3]!='ALL':

t[0]='%s < %s'%(t[1],str(t[4]))

elif t[2]=='<' and t[3]!='ALL':

t[0]='%s < %s'%(t[1],str(t[4]))

else:

t[0]='%s V %s'%(t[1],t[3])

def p\_condition(t):

''' condition : NAME '>' NUMBER

| NAME '>' agg

| NAME '<' NUMBER

| NAME '<' agg

| NAME '=' NUMBER

| NAME '=' agg

| NAME '>' NAME

| NAME '<' NAME

| NAME '=' NAME

| list '>' list

| list '<' list

| list '=' list

| NAME '=' INV NAME INV '''

if t[2]=='>':

t[0]='%s>%s'%(t[1],str(t[3]))

elif t[2]=='<' :

t[0]='%s<%s'%(t[1],str(t[3]))

elif t[2]=='=' and len(t) <= 4:

t[0]='%s=%s'%(t[1],str(t[3]))

else:

t[0]='%s=\"%s\"'%(t[1],str(t[4]))

def p\_group(t):

''' group : GROUP\_BY listg having

|

'''

if len(t)==3:

t[0]='(%s)'%(t[2])

elif len(t)==4:

t[0]='(%s {SIG(%s)})'%(t[2],t[3])

else:

t[0]=" "

def p\_agg(t):

''' agg : SUM LP NAME RP

| AVG LP NAME RP

| COUNT LP NAME RP

| MIN LP NAME RP

| MAX LP NAME RP

| COUNT LP '\*' RP

'''

t[0]='%s(%s)'%(t[1],t[3])

def p\_having(t):

''' having : HAVING lst

|

'''

if(len(t)==3):

t[0]=(t[2])

else:

t[0]=""

def p\_list(t):

''' list : '\*'

| NAME

| NAME DOT NAME

| list COMMA list

| list AND NAME

| list OR NAME

| LP NAME NATURAL\_JOIN NAME RP

| LP NAME NATURAL\_JOIN query RP

| LP query NATURAL\_JOIN query RP

| LP NAME JOIN query RP

| LP query JOIN query RP

| LP NAME JOIN NAME RP

| NAME NATURAL\_JOIN NAME

| NAME NATURAL\_JOIN query

| query NATURAL\_JOIN query

| NAME JOIN query

| query JOIN query

| NAME JOIN NAME

| agg

'''

if len(t)==2 and t[1]!='\*':

t[0]=(t[1])

elif t[1]=='\*':

t[0]=''

elif t[3]=='NATURAL\_JOIN':

t[0]='N\_JOIN{%s,%s}'%(t[2],t[4])

elif t[3]=='JOIN':

t[0]='{%s X %s}'%(t[2],t[4])

elif t[2]=='NATURAL\_JOIN':

t[0]='N\_JOIN{%s,%s}'%(t[1],t[3])

elif t[2]=='JOIN':

t[0]='{%s X %s}'%(t[1],t[3])

elif t[2]==',':

t[0]='%s,%s'%(t[1],t[3])

else:

t[0]='%s.%s'%(t[1],t[3])

def p\_listg(t):

''' listg : NAME

| listg COMMA NAME

| listg AND NAME

| listg OR NAME

'''

if len(t)==2:

t[0]=t[1]

else:

t[0]='%s,%s'%(t[1],t[3])

def p\_error(t):

print("Syntax error at '%s'" % t.value)

def p\_join(t):

'''joinquery : query JOIN query

| query JOIN joinquery

'''

t[0]=t[1]+' JOIN ' + t[3]

# insert

def p\_insert(t):

''' insert : INSERT INTO NAME query

| INSERT INTO NAME VALUES LP data RP

| INSERT INTO NAME LP data RP VALUES LP data RP'''

if t[4]=='VALUES' :

t[0]='%s <- %s U [%s]' %(t[3],t[3],t[6])

elif len(t) > 7 and t[7]=='VALUES' :

t[0]='%s <- %s U [%s]' %(t[3],t[3],t[9])

else:

t[0]='%s <- %s U [%s]' %(t[3],t[3],t[4])

def p\_data(t):

''' data : NUMBER

| NAME

| INV NAME INV

| data COMMA data

'''

if len(t)==2:

t[0]=t[1]

elif t[2]==',':

t[0]=t[1]+','+t[3]

else:

t[0]=t[1]+t[2]+t[3]

def p\_delete(t):

''' delete : DELETE NAME query

| DELETE NAME WHERE condition

| DELETE NAME FROM table WHERE condition'''

if t[3]=='WHERE':

t[0]='%s = %s - {SIG(%s)}(%s)' %(t[2],t[2],t[4],t[2])

elif t[3]=='FROM':

t[0]='%s = %s - {SIG(%s)}(%s)' %(t[2],t[2],t[4],t[2])

else:

t[0]='%s = %s - [%s]' %(t[2],t[2],t[3])

def p\_update(t):

''' update : UPDATE NAME SET NAME '=' expression

| UPDATE NAME SET NAME '=' expression WHERE condition

'''

if len(t)>7 and t[7]=='WHERE':

t[0]='${%s=%s}(SIG{%s}%s)' %(t[4],t[6],t[8],t[2])

else:

t[0]='${%s=%s}(%s)' %(t[4],t[6],t[2])

def p\_expression(t):

'''expression : NAME '+' NUMBER

| NAME '-' NUMBER

| NAME '\*' NUMBER

| NAME '/' NUMBER

| NAME

| NUMBER'''

if t[2] == '+' : t[0] = '%s + %s' %(t[1],t[3])

elif t[2] == '-': t[0] = '%s - %s' %(t[1],t[3])

elif t[2] == '\*': t[0] = '%s \* %s' %(t[1],t[3])

elif t[2] == '/': t[0] = '%s / %s' %(t[1],t[3])

yacc.yacc()

while 1:

try:

s = raw\_input('-> ')

pass

except EOFError:

break

a=yacc.parse(s)

print a