Atitit ast构造法 v5 t0285

Ast 规范可以按照sql和jqpl的。。这里选择sql的。。Jpql的都是普通node

核心原则，只有右括号出现，才把零食stash的节点，挂在到最上面去。

多级stack 每一个零时的node有一个零时的stack node。。。

只有一个方法finish fun()，才把节点挂在上一级。。然后一步步执行。。

左括号增加stash节点。。

逗号，只是增加参数。。

D:\0wkspc\platform-top-run-coffey\platform-top-service\platform-top-service-finance\src\main\java\org\chwin\firefighting\apiserver\QL\astiqlAstBuilder.java

package org.chwin.firefighting.apiserver.QL;  
  
import com.alibaba.fastjson.JSON;  
import com.alibaba.fastjson.serializer.SerializerFeature;  
import com.alibaba.fastjson.serializer.SimplePropertyPreFilter;  
import lombok.extern.slf4j.Slf4j;  
import net.sf.jsqlparser.expression.Expression;  
import net.sf.jsqlparser.expression.Function;  
import net.sf.jsqlparser.expression.NullValue;  
import net.sf.jsqlparser.expression.StringValue;  
import net.sf.jsqlparser.expression.operators.relational.ExpressionList;  
import net.sf.jsqlparser.parser.Token;  
import net.sf.jsqlparser.schema.Column;  
import org.hibernate.criterion.NullExpression;  
import org.jetbrains.annotations.NotNull;  
  
import java.util.ArrayList;  
import java.util.EmptyStackException;  
import java.util.List;  
import java.util.Stack;  
  
@Slf4j  
public class astiqlAstBuilder {  
  
 //private final String str;  
 List<Token> tokenList;  
 int cursor = -1;  
  
 Token cur\_token;  
 Stack<Expression> stack = new Stack();  
  
 public astiqlAstBuilder(List<Token> li) {  
 tokenList = li;  
 }  
  
 public static void main(String[] args) {  
  
 String s = " $(from(tab),where( c=5),and(d=xxx),select(a,b,c))";  
 s = "fun(a,b)";  
  
 s = "fun(fun1(),b)";  
 s = "fun(fun1(c),b)";  
  
 s = "fun(fun1(c),b(c()))";  
 s = "fun(fun1(),b)";  
 char[] a = s.toCharArray();  
 List<Token> li = new atiQlTokiz(s).getListToken(s);  
 System.*out*.println(li);  
  
 Function fun = (Function) new astiqlAstBuilder(li).getFuntionAst();  
  
  
 System.*out*.println(QlExpress.*toJsonString*(fun));  
  
 }  
  
  
 private Expression getFuntionAst() {  
 try {  
 cursor++;  
  
 cur\_token = tokenList.get(cursor);  
  
  
 *log*.info("now char:" + String.*valueOf*(cur\_token));  
 switch (cur\_token.image) {  
 case "(":  
 return leftBracket\_case();  
 case ")":  
 return ritBracket\_case();  
 case ",":  
 return comma\_cash();  
 default:  
 return Normalchar\_case();  
 }  
 } catch (IndexOutOfBoundsException e) {  
   
 return stack.pop();  
  
 }  
  
 }  
  
 //...  
  
 */\*\*  
 \* none ,only add to last push stack  
 \*  
 \** ***@return*** *\*/* private Expression Normalchar\_case() {  
 try {  
 Expression obj = stack.pop();  
 if (obj instanceof NullValue) {  
 //pop ...  
 } else  
 stack.push(obj);  
  
 StringValue aaa = new StringValue("aaa");  
 aaa.setValue(cur\_token.image);  
 Expression e = aaa;  
 stack.push(e);  
 return getFuntionAst();  
  
  
 } catch (EmptyStackException ex) {  
 Expression e = new StringValue(cur\_token.image);  
 stack.push(e);  
  
 return getFuntionAst();  
 }  
  
 }  
  
 // last is Function,or the normal Stringvalue Expression  
 private Expression leftBracket\_case() {  
 Expression obj = stack.pop();  
 // if (obj instanceof StringValue) {  
 Function f = new Function();  
 f.setName(((StringValue) obj).getValue());  
 ExpressionList ExpressionList1 = new ExpressionList();  
 ExpressionList1.setExpressions(new ArrayList<>());  
 f.setParameters(ExpressionList1);  
 // Expression tk = getTokenTmp(obj);  
 stack.push(f);  
 stack.push(new NullValue());  
  
 // }  
  
  
 return getFuntionAst();  
 }  
  
  
 private Expression comma\_cash() {  
 Expression obj = stack.pop();  
 Function f = (Function) stack.pop();  
 f.getParameters().getExpressions().add(obj);  
 stack.push(f);  
  
 return getFuntionAst();  
 }  
  
 // s = "fun(fun1(),b)";  
 private Expression ritBracket\_case() {  
 Expression obj = stack.pop(); //pop cur sub function ,or sub str exp  
 if (obj instanceof NullValue) {  
  
 } else {  
 Function f = (Function) stack.pop();  
 f.getParameters().getExpressions().add(obj);  
 stack.push(f);  
 }  
  
  
 return getFuntionAst();  
 }  
  
  
}