




















极客大学架构师训练营

李智慧

Panthera代码解析

 generator	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 transformer	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 ExplainSession.java	Panthera integrated into apache-hive-release 0.12.0
 HiveParseException.java	Panthera integrated into apache-hive-release 0.12.0
 PantheraConstants.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 PantheraExpParser.java	Panthera integrated into apache-hive-release 0.12.0
 PantheraMap.java	Panthera integrated into apache-hive-release 0.12.0
 ParseError.java	Panthera integrated into apache-hive-release 0.12.0
 SQLMagicUtil.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 SqlASTChecker.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 SqlASTErrorNode.java	Panthera integrated into apache-hive-release 0.12.0
 SqlASTNode.java	Panthera integrated into apache-hive-release 0.12.0
 SqlASTTranslator.java	Panthera integrated into apache-hive-release 0.12.0
 SqlParseDriver.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 SqlParseException.java	Panthera integrated into apache-hive-release 0.12.0
 SqlTextSession.java	Panthera integrated into apache-hive-release 0.12.0
 SqlXlateException.java	transform DISTINCT, ASTERISK/double check GROUP at end/support ' ' a...
 SqlXlateUtil.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 TranslateContext.java	Panthera integrated into apache-hive-release 0.12.0

```
*/  
  
public ASTNode translate(SqlASTNode sqlASTRoot) throws SqlXlateException {  
    ASTNode ret = null;  
    LOG.info("Starting Translation from SQL AST to Hive AST");  
    LOG.info("Original SQL AST : " + sqlASTRoot.toStringTree().replace('(', '[').replace(')', ']'));  
  
    TranslateContext context = new TranslateContext(conf);  
  
    if (transformer == null) {  
        transformer = TransformerBuilder.buildTransformer();  
    }  
    transformer.transformAST(sqlASTRoot, context);  
    LOG.info("Transformed SQL AST : " + sqlASTRoot.toStringTree().replace('(', '[').replace(')', ']'));  
  
    QueryTextGenerator tg = TextGeneratorFactory.getTextGenerator(sqlASTRoot);  
    String recoveredQuery = "";  
    try {  
        recoveredQuery += tg.textGenerateQuery(sqlASTRoot, context);  
    } catch (Exception tgEx) {  
        LOG.error(tgEx.getMessage());  
        recoveredQuery += "Failed.";  
        //throw new SqlXlateException(tgEx);  
    }  
    LOG.info("Rebuilded SQL Query : " + recoveredQuery);  
    SqlTextSession.set(recoveredQuery);  
  
    HiveASTGenerator generator = GeneratorFactory.getGenerator(sqlASTRoot);  
    ASTNode hiveTopNode = new ASTNode();  
  
    // long b = System.currentTimeMillis();  
    // for (int i = 0; i < 1000; i++) {  
    generator.generateHiveAST(hiveTopNode, sqlASTRoot, hiveTopNode, sqlASTRoot, context);  
    // }  
    // long e = System.currentTimeMillis();  
    // System.out.println("-----" + (e - b));
```


- BaseSqlASTTransformer.java
- BetweenTransformer.java
- ConditionStructTransformer.java
- CountAsteriskPositionTransformer.java
- CrossJoinTransformer.java
- DistinctTransformer.java
- ExpandAsteriskTransformer.java
- FilterBlockAdjustTransformer.java
- FilterInwardTransformer.java
- GroupElementNormalizeTransformer.java
- InTransformer.java
- IntersectTransformer.java
- Leftsemi2LeftJoinTransformer.java
- MinusIntersectTransformer.java
- MinusTransformer.java
- MultipleTableSelectTransformer.java
- NaturalJoinTransformer.java
- NotEqualJoinTransformer.java
- NothingTransformer.java
- OrderByFunctionTransformer.java

```

public class TransformerBuilder {

    private static SqlASTTransformer tf =
        new RedundantSelectGroupItemTransformer(
        new DistinctTransformer(
        new GroupElementNormalizeTransformer(
        new PrepareQueryInfoTransformer(
        new OrderByTransformer(
        new OrderByFunctionTransformer(
        new MinusIntersectTransformer(
        new PrepareQueryInfoTransformer(
        new UnionTransformer(
        new Leftsemi2LeftJoinTransformer(
        new CountAsteriskPositionTransformer(
        new FilterInwardTransformer(
        // unComment the following line to use leftJoin method to handle not exists for correlated
        //new NotEqualJoinTransformer(
        new CrossJoinTransformer(
        new PrepareQueryInfoTransformer(
        new SubQUnnestTransformer(
        new PrepareFilterBlockTransformer(
        new PrepareQueryInfoTransformer(
        new TopLevelUnionTransformer(
        new FilterBlockAdjustTransformer(
        new PrepareFilterBlockTransformer(
        new ExpandAsteriskTransformer(
        new PrepareQueryInfoTransformer(
        new CrossJoinTransformer(
        new PrepareQueryInfoTransformer(
        new ConditionStructTransformer(
        new MultipleTableSelectTransformer(
        new WhereConditionOptimizationTransformer(
        new PrepareQueryInfoTransformer(
        new InTransformer(
        new TopLevelUnionTransformer(
        new MinusIntersectTransformer(
        new NaturalJoinTransformer(
        new OrderByNotInSelectListTransformer(
        new RowNumTransformer(
        new BetweenTransformer(
        new UsingTransformer(
        new SchemaDotTableTransformer(
        new NothingTransformer())))))))))))))))))))))))))))))))))));

    private TransformerBuilder() {
    }

    public static SqlASTTransformer buildTransformer() {
        return tf;
    }
}

```

```

/**
 * Transformer interface.<br>
 * transformer instance is created as singleton, please keep it thread safety.<br>
 * SqlASTTransformer.
 *
 */
public interface SqlASTTransformer {
    void transformAST(CommonTree tree, TranslateContext context) throws SqlXlateException;
}

```

```

public class UnionTransformer extends BaseSqlASTTransformer {



























    SqlASTTransformer tf;

    public UnionTransformer(SqlASTTransformer tf) {
        this.tf = tf;
    }

    @Override
    public void transform(CommonTree tree, TranslateContext context) throws SqlXlateException {
        tf.transformAST(tree, context);
        trans(tree, context);
    }

    private void trans(CommonTree node, TranslateContext context) throws SqlXlateException {
        int childCount = node.getChildCount();
        for (int i = 0; i < childCount; i++) {
            int index = i;
            if (node.getType() == PantheraExpParser.SUBQUERY) {
                // for multiple UNION
                Integer reduceChildCount = (Integer) context.getBallFromBasket(node);
                if (reduceChildCount != null) {
                    index = index - reduceChildCount;
                }
            }
            CommonTree child = (CommonTree) node.getChild(index);
            trans(child, context);
        }
        if (node.getType() == PantheraExpParser.SQL92_RESERVED_UNION) {
            processUnion((CommonTree) node.getFirstChildWithType(PantheraExpParser.SQL92_RESERVED_ALL),
                node, context);
        }
    }
}

```


 AndGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 AnyElementGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 AsteriskGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 BaseHiveASTGenerator.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 BetweenGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 BooleanGenerator.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 CastGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 CharGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 CharStringGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 ConcatenationOpGenerator.java	transform DISTINCT, ASTERISK/double check GROUP at end/support ' ' a...
 CountGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 CurrentGenerator.java	transform DISTINCT, ASTERISK/double check GROUP at end/support ' ' a...
 DateGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 DateIntervalExpressionGenerator....	Panthera integrated into apache-hive-release 0.12.0
 DecimalGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 DotAsteriskGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 DoubleGenerator.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 EqualsNsGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 EqualsOpGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 ErrorGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 ExplainGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 FalseGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 FloatGenerator.java	change exception type;support multi-line SQL/GROUP BY position/NATRUA...
 FollowingGenerator.java	transform DISTINCT, ASTERISK/double check GROUP at end/support ' ' a...
 FromGenerator.java	Panthera integrated into apache-hive-release 0.12.0
 FunctionEnablingOverGenerator.j...	transform DISTINCT, ASTERISK/double check GROUP at end/support ' ' a...


```

public interface HiveASTGenerator {
    public static final Log LOG = LogFactory.getLog("hive.ql.parse.sql.HiveASTGenerator");

    boolean generateHiveAST(ASTNode hiveRoot, CommonTree sqlRoot, ASTNode currentHiveNode,
        CommonTree currentSqlNode, TranslateContext context) throws SqlXlateException;

    void setHivePreGenerator(HiveASTGenerator preGenerator);

    void setHivePostGenerator(HiveASTGenerator postGenerator);

}

```

zhihuili/project-panthera-ase/blob/master/ql/src/java/org/apache/hadoop/hive/ql/parse/sql/generator/BaseHiveASTGenerator.java

```

@Override
public boolean generateHiveAST(ASTNode hiveRoot, CommonTree sqlRoot, ASTNode currentHiveNode,
    CommonTree currentSqlNode, TranslateContext context) throws SqlXlateException {
    if (this.preGenerator != null
        && !this.preGenerator.generateHiveAST(hiveRoot, sqlRoot, currentHiveNode, currentSqlNode,
            context)) {
        return false;
    }
    if (!this.generate(hiveRoot, sqlRoot, currentHiveNode, currentSqlNode, context)) {
        return false;
    }
    if (this.postGenerator != null
        && !this.postGenerator.generateHiveAST(hiveRoot, sqlRoot, currentHiveNode, currentSqlNode,
            context)) {
        return false;
    }
    return true;
}

```

```
public class LikeGenerator extends BaseHiveASTGenerator {

    @Override
    public boolean generate(ASTNode hiveRoot, CommonTree sqlRoot, ASTNode currentHiveNode,
        CommonTree currentSqlNode, TranslateContext context) throws SqlXlateException {
        return super.baseProcess(HiveParser.KW_LIKE, "like", hiveRoot, sqlRoot, currentHiveNode, currentSqlNode, context);
    }

}
```

[zhihuili/project-panthera-ase/blob/master/ql/src/java/org/apache/hadoop/hive/ql/parse/sql/generator/BaseHiveASTGenerator.java](#)

```
boolean baseProcess(int ttype, String text, ASTNode hiveRoot, CommonTree sqlRoot,
    ASTNode currentHiveNode,
    CommonTree currentSqlNode, TranslateContext context) throws SqlXlateException {
    ASTNode ret = this.newHiveASTNode(ttype, text);
    this.attachHiveNode(hiveRoot, currentHiveNode, ret);
    return this.generateChildren(hiveRoot, sqlRoot, ret, currentSqlNode, context);
}
```

```

public abstract class BaseHiveASTGenerator implements HiveASTGenerator {

    private HiveASTGenerator preGenerator;
    private HiveASTGenerator postGenerator;

    boolean generateChildren(ASTNode hiveRoot, CommonTree sqlRoot, ASTNode currentHiveNode,
        CommonTree currentSqlNode, TranslateContext context) throws SqlXlateException {
        if (currentSqlNode.getChildren() == null) {
            return true;
        }
        Iterator i = currentSqlNode.getChildren().iterator();
        if (i == null) {
            return true;
        }
        while (i.hasNext()) {
            Object o = i.next();
            CommonTree node = (CommonTree) o;
            HiveASTGenerator generator = GeneratorFactory.getGenerator(node);
            if (generator == null) {
                throw new SqlXlateException(node, "Untransformed sql AST node:" + node.getText());
            }
            if (!generator.generateHiveAST(hiveRoot, sqlRoot,
                currentHiveNode, node, context)) {
                return false;
            }
        }
        return true;
    }
}

```


<https://github.com/zhihuili/project-panthera-ase/blob/master/ql/src/java/org/apache/hadoop/hive/ql/parse/sql/>

THANKS! |  极客大学