芋艿v的博客

愿编码半生，如老友相伴！

微信公众号福利：芋艿的后端小屋

- 0. 阅读源码葵花宝典
- 1. RocketMQ / MyCAT / Sharding-JDBC 详细中文注释源码
- 2. 您对于源码的疑问每条留言都将得到认真回复
- 3. 新的源码解析文章实时收到通知，每周六十点更新
- 4. 认真的源码交流微信群

分类

- Docker [2]
- MyCAT [9]
- Nginx [1]
- RocketMQ [14]
- Sharding-JDBC [17]
- 技术杂文 [2]

分享

Sharding-JDBC 源码分析 —— SQL 解析（四）之插入SQL

2017-07-29 更新日期:2017-07-31 总阅读量:18次

文章目录

1. 1. 概述
2. 2. InsertStatement
3. 3. #parse()
    3.1. 3.1 #parseInto()
    3.2. 3.2 #parseColumns()
    3.3. 3.3 #parseValues()
        3.3.1. 3.4.1 GeneratedKey
        3.3.2. 3.4.2 Condition
    3.4. 3.4 #parseCustomizedInsert()
    3.5. 3.5 #appendGenerateKey()
        3.5.1. 3.5.1 GeneratedKeyToken
4. 666. 彩蛋

分享

> □□□关注**微信公众号：【芋艿的后端小屋】**有福利：
>
> 1. RocketMQ / MyCAT / Sharding-JDBC **所有**源码分析文章列表
> 2. RocketMQ / MyCAT / Sharding-JDBC **中文注释源码 GitHub 地址**
> 3. 您对于源码的疑问每条留言**都**将得到**认真**回复。**甚至不知道如何读源码也可以请教噢**。
> 4. **新的**源码解析文章**实时**收到通知。**每周更新一篇左右**。
> 5. **认真的**源码交流微信群。

- 1. 概述
- 2. InsertStatement
- 3. #parse()
  - 3.1 #parseInfo()
  - 3.2 #parseColumns()
  - 3.3 #parseValues()
  - 3.4 #parseCustomizedInsert()
  - 3.5 #appendGenerateKey()
- 666. 彩蛋

1. 概述

本文前置阅读：

- 《SQL 解析（一）之词法解析》
- 《SQL 解析（二）之SQL解析》

本文分享**插入SQL解析**的源码实现。

不考虑 INSERT SELECT 情况下，插入SQL解析比查询SQL解析复杂度低的多的多。不同数据库在插入SQL语法上也统一的多。**本文分享 MySQL 插入SQL解析器 MySQLInsertParser。**

MySQL INSERT 语法一共有 3 种 ：

- 第一种：`INSERT {VALUES | VALUES}`

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

- 第二种：`INSERT SET`

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
```

分享

```
        [, col_name=expr] ... ]
```

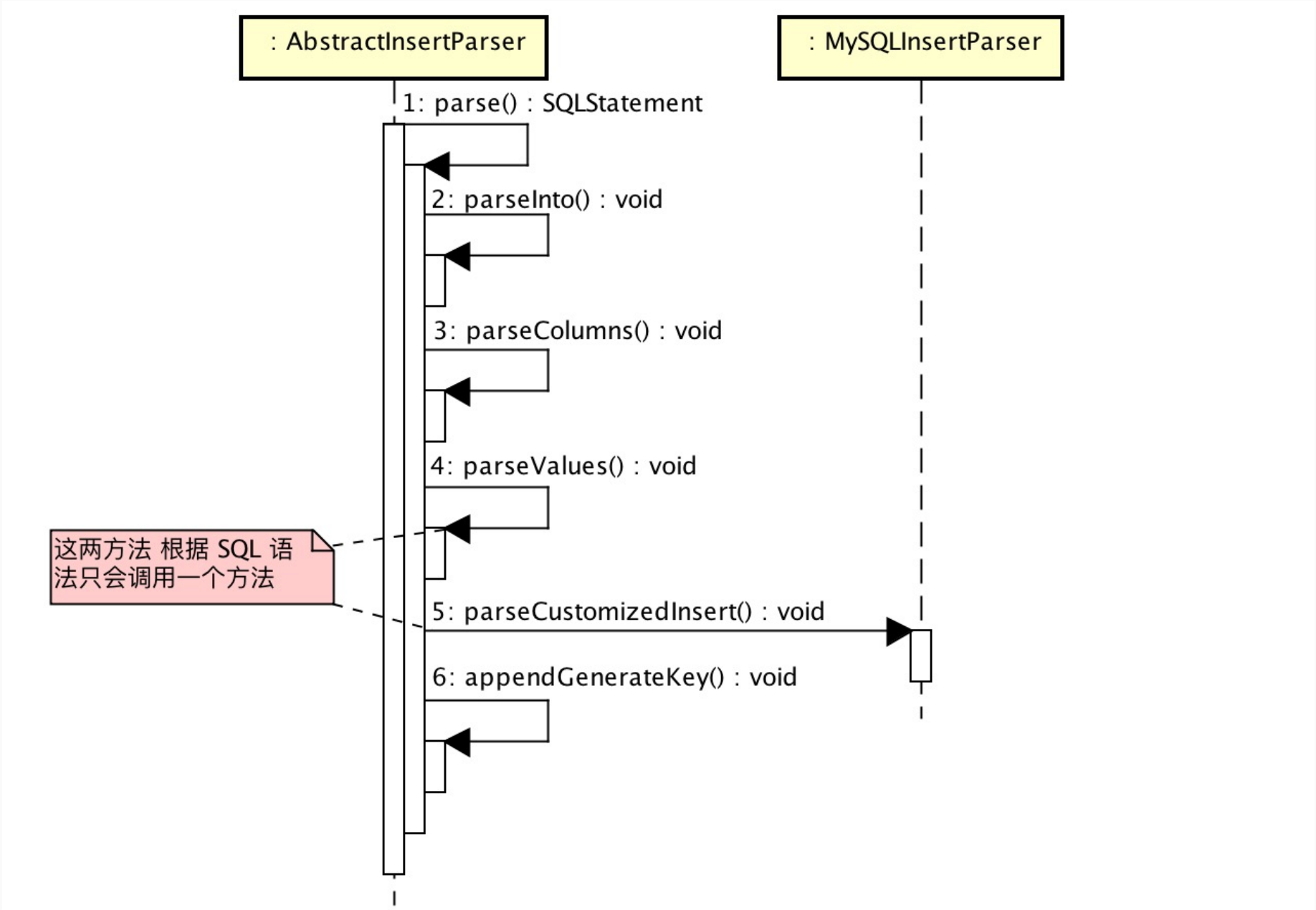
- 第三种：`INSERT SELECT`

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

Sharding-JDBC 目前支持：

- 第一种：`INSERT {VALUES | VALUES}` **单条记录**
- 第二种：`INSERT SET`

Sharding-JDBC 插入SQL解析主流程如下：




```
// AbstractInsertParser.java
```

分享

```
public final InsertStatement parse() {
    sqlParser.getLexer().nextToken(); // 跳过 INSERT 关键字
    parseInto(); // 解析INTO
    parseColumns(); // 解析表
    if (sqlParser.equalAny(DefaultKeyword.SELECT, Symbol.LEFT_PAREN)) {
        throw new UnsupportedOperationException("Cannot support subquery");
    }
    if (getValuesKeywords().contains(sqlParser.getLexer().getCurrentToken().getType())) { // 第一种插入S
        parseValues();
    } else if (getCustomizedInsertKeywords().contains(sqlParser.getLexer().getCurrentToken().getType())
        parseCustomizedInsert();
    }
    appendGenerateKey(); // 自增主键
    return insertStatement;
}
```

> **Sharding-JDBC 正在收集使用公司名单：传送门。**
>
> **你的登记，会让更多人参与和使用 Sharding-JDBC。传送门**
>
> **Sharding-JDBC 也会因此，能够覆盖更多的业务场景。传送门**
>
> **登记吧，骚年！传送门**

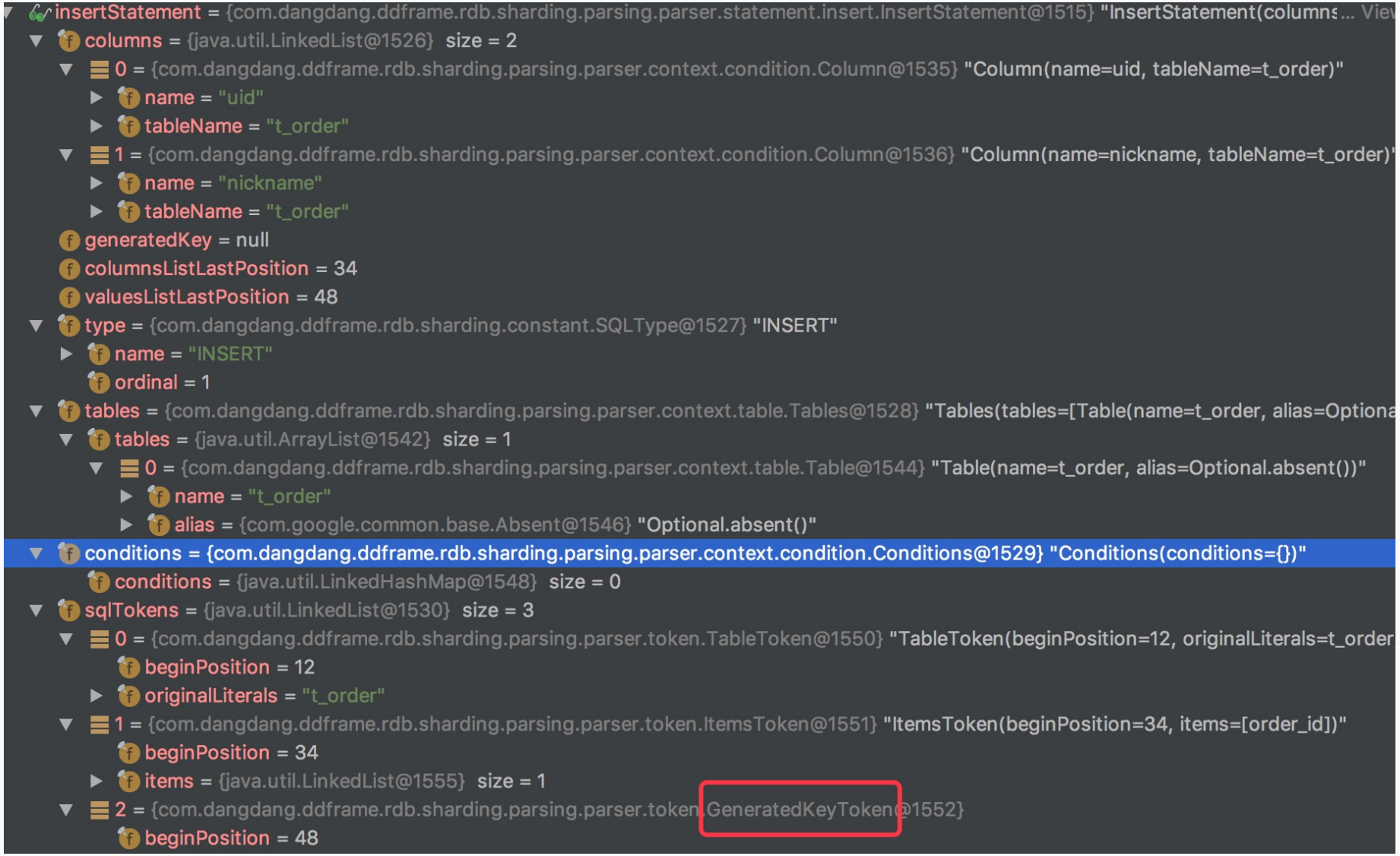
2. InsertStatement

插入SQL 解析结果。

```
public final class InsertStatement extends AbstractSQLStatement {
    /**
```

```
     * 插入字段
     */
    private final Collection<Column> columns = new LinkedList<>();
    /**
     *
     */
    private GeneratedKey generatedKey;
    /**
     * 插入字段 下一个Token 开始位置
     */
    private int columnsListLastPosition;
    /**
     * 值字段 下一个Token 开始位置
     */
    private int valuesListLastPosition;
}
```

我们来看下 `INSERT INTO t_order (uid, nickname) VALUES (?, ?)` 的**解析结果**：



3. #parse()

3.1 #parseInto()

解析**表**。

```
// AbstractInsertParser.java
/**
* 解析表
*/
private void parseInto() {
    // 例如，Oracle，INSERT FIRST/ALL 目前不支持
    if (getUnsupportedKeywords().contains(sqlParser.getLexer().getCurrentToken().getType())) {
        throw new SQLParsingUnsupportedException(sqlParser.getLexer().getCurrentToken().getType());
    }
    sqlParser.skipUntil(DefaultKeyword.INTO);
    sqlParser.getLexer().nextToken();
    // 解析表
    sqlParser.parseSingleTable(insertStatement);
    skipBetweenTableAndValues();
}
/**
* 跳过 表 和 插入字段 中间的 Token
* 例如 MySQL : [PARTITION (partition_name,...)]
*/
private void skipBetweenTableAndValues() {
    while (getSkippedKeywordsBetweenTableAndValues().contains(sqlParser.getLexer().getCurrentToken().ge
        sqlParser.getLexer().nextToken();
        if (sqlParser.equalAny(Symbol.LEFT_PAREN)) {
            sqlParser.skipParentheses();
        }
    }
}
```

其中 `#parseSingleTable()` 请看《SQL 解析（二）之SQL解析》的 `#parseSingleTable()` 小节。

3.2 #parseColumns()

解析**插入字段**。

```
// AbstractInsertParser.java
private void parseColumns() {
    Collection<Column> result = new LinkedList<>();
    if (sqlParser.equalAny(Symbol.LEFT_PAREN)) {
        String tableName = insertStatement.getTables().getSingleTableName();
        Optional<String> generateKeyColumn = shardingRule.getGenerateKeyColumn(tableName); // 自动生成键
        int count = 0;
        do {
            // Column 插入字段
            sqlParser.getLexer().nextToken();
            String columnName = SQLUtil.getExactlyValue(sqlParser.getLexer().getCurrentToken().getLiter
            result.add(new Column(columnName, tableName));
            sqlParser.getLexer().nextToken();
            // 自动生成键
            if (generateKeyColumn.isPresent() && generateKeyColumn.get().equalsIgnoreCase(columnName))
                generateKeyColumnIndex = count;
            }
            count++;
        } while (!sqlParser.equalAny(Symbol.RIGHT_PAREN) && !sqlParser.equalAny(Assist.END));
        //
        insertStatement.setColumnsListLastPosition(sqlParser.getLexer().getCurrentToken().getEndPositio
```

```
        //
        sqlParser.getLexer().nextToken();
    }
    insertStatement.getColumns().addAll(result);
}
```

3.3 #parseValues()

解析**值字段**

```
/**
* 解析值字段
*/
private void parseValues() {
    boolean parsed = false;
    do {
        if (parsed) { // 只允许INSERT INTO 一条
            throw new UnsupportedOperationException("Cannot support multiple insert");
        }
        sqlParser.getLexer().nextToken();
        sqlParser.accept(Symbol.LEFT_PAREN);
        // 解析表达式
        List<SQLExpression> sqlExpressions = new LinkedList<>();
        do {
            sqlExpressions.add(sqlParser.parseExpression());
        } while (sqlParser.skipIfEqual(Symbol.COMMA));
        //
```

```
            insertStatement.setValuesListLastPosition(sqlParser.getLexer().getCurrentToken().getEndPosition
            // 解析值字段
            int count = 0;
            for (Column each : insertStatement.getColumns()) {
                SQLExpression sqlExpression = sqlExpressions.get(count);
                insertStatement.getConditions().add(new Condition(each, sqlExpression), shardingRule);
                if (generateKeyColumnIndex == count) { // 自动生成键
                    insertStatement.setGeneratedKey(createGeneratedKey(each, sqlExpression));
                }
                count++;
            }
            sqlParser.accept(Symbol.RIGHT_PAREN);
            parsed = true;
        }
        while (sqlParser.equalAny(Symbol.COMMA)); // 字段以 "," 分隔
    }
    /**
    * 创建 自动生成键
    *
    * @param column 字段
    * @param sqlExpression 表达式
    * @return 自动生成键
    */
    private GeneratedKey createGeneratedKey(final Column column, final SQLExpression sqlExpression) {
        GeneratedKey result;
        if (sqlExpression instanceof SQLPlaceholderExpression) { // 占位符
            result = new GeneratedKey(column.getName(), ((SQLPlaceholderExpression) sqlExpression).getIndex
        } else if (sqlExpression instanceof SQLNumberExpression) { // 数字
            result = new GeneratedKey(column.getName(), -1, ((SQLNumberExpression) sqlExpression).getNumber
```

```
    } else {
        throw new ShardingJdbcException("Generated key only support number.");
    }
    return result;
}
```

3.4.1 GeneratedKey

自动生成键，**属于分片上下文信息**。

```
public final class GeneratedKey {
    /**
     * 字段
     */
    private final String column;
    /**
     * 第几个占位符
     */
    private final int index;
    /**
     * 值
     */
    private final Number value;
}
```

3.4.2 Condition

条件对象，**属于分片上下文信息**。在**插入SQL解析**里存储**影响分片的值字段**。后续《SQL 路由》 会专门分享这块。

```
public final class Condition {
    /**
     * 字段
     */
    @Getter
    private final Column column;

    // ... 省略其它属性
}
public final class Column {
    /**
     * 列名
     */
    private final String name;
    /**
     * 表名
     */
    private final String tableName;
}
```

3.4 #parseCustomizedInsert()

解析**第二种插入SQL**：`INSERT SET`。例如：

```
INSERT INTO test SET id = 4  ON DUPLICATE KEY UPDATE name = 'doubi', name = 'hehe';
INSERT INTO test SET id = 4, name = 'hehe';
```

```
private void parseInsertSet() {
    do {
        getSqlParser().getLexer().nextToken();
        // 插入字段
        Column column = new Column(SQLUtil.getExactlyValue(getSqlParser().getLexer().getCurrentToken().
        getSqlParser().getLexer().nextToken();
        // 等号
        getSqlParser().accept(Symbol.EQ);
        // 【值】表达式
        SQLExpression sqlExpression;
        if (getSqlParser().equalAny(Literals.INT)) {
            sqlExpression = new SQLNumberExpression(Integer.parseInt(getSqlParser().getLexer().getCurre
        } else if (getSqlParser().equalAny(Literals.FLOAT)) {
            sqlExpression = new SQLNumberExpression(Double.parseDouble(getSqlParser().getLexer().getCur
        } else if (getSqlParser().equalAny(Literals.CHARS)) {
            sqlExpression = new SQLTextExpression(getSqlParser().getLexer().getCurrentToken().getLitera
        } else if (getSqlParser().equalAny(DefaultKeyword.NULL)) {
            sqlExpression = new SQLIgnoreExpression();
        } else if (getSqlParser().equalAny(Symbol.QUESTION)) {
            sqlExpression = new SQLPlaceholderExpression(getSqlParser().getParametersIndex());
            getSqlParser().increaseParametersIndex();
        } else {
            throw new UnsupportedOperationException("");
        }
        getSqlParser().getLexer().nextToken();
        // Condition
        if (getSqlParser().equalAny(Symbol.COMMA, DefaultKeyword.ON, Assist.END)) {
            getInsertStatement().getConditions().add(new Condition(column, sqlExpression), getShardingR
```

```
        } else {
            getSqlParser().skipUntil(Symbol.COMMA, DefaultKeyword.ON);
        }
    } while (getSqlParser().equalAny(Symbol.COMMA)); // 字段以 "," 分隔
}
```

3.5 #appendGenerateKey()

当表设置**自动生成键**，并且插入SQL**没**写自增字段，增加该字段。例如：

```
// 主键为user_id
INSERT INTO t_user(nickname, age) VALUES (?, ?)
```

后续 SQL 改写会生成该自增编号，并改写该 SQL。后续《SQL 改写》 会专门分享这块。

```
private void appendGenerateKey() {
    // 当表设置自动生成键，并且插入SQL没写自增字段
    String tableName = insertStatement.getTables().getSingleTableName();
    Optional<String> generateKeyColumn = shardingRule.getGenerateKeyColumn(tableName);
    if (!generateKeyColumn.isPresent() || null != insertStatement.getGeneratedKey()) {
        return;
    }
    // ItemsToken
    ItemsToken columnsToken = new ItemsToken(insertStatement.getColumnsListLastPosition());
    columnsToken.getItems().add(generateKeyColumn.get());
    insertStatement.getSqlTokens().add(columnsToken);
    // GeneratedKeyToken
```

```
    insertStatement.getSqlTokens().add(new GeneratedKeyToken(insertStatement.getValuesListLastPosition(
}
```

3.5.1 GeneratedKeyToken

自增主键标记对象。

```
public final class GeneratedKeyToken implements SQLToken {
    /**
     * 开始位置
     */
    private final int beginPosition;
}
```

666. 彩蛋

😈 是不是比《SQL 解析（三）之插入SQL》简单很多。

道友，可否分享一波【本文】到朋友圈。

继续加油更新！

Sharding-JDBC

PREVIOUS:
« Sharding-JDBC 源码分析 —— SQL 解析（五）之更新SQL
NEXT:
» Sharding-JDBC 源码分析 —— SQL 解析（三）之查询SQL