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

分享

Sharding-JDBC 源码分析 —— SQL 解析（六）之删除SQL

2017-08-02 更新日期:2017-07-31 总阅读量:6次

文章目录

> □□□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**

分享

> 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

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**删除SQL解析**的源码实现。

□ 如果你已经理解《SQL 解析（三）之查询SQL》，那本文会是一篇水文，当成一种放松吧。还是跟前文一样，以 MySQL 举例子。我们来一起看看 MySQLDeleteParser。

MySQL DELETE 语法一共有 2 种 ：

- 第一种：**Single-table syntax**

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
    [PARTITION (partition_name,...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

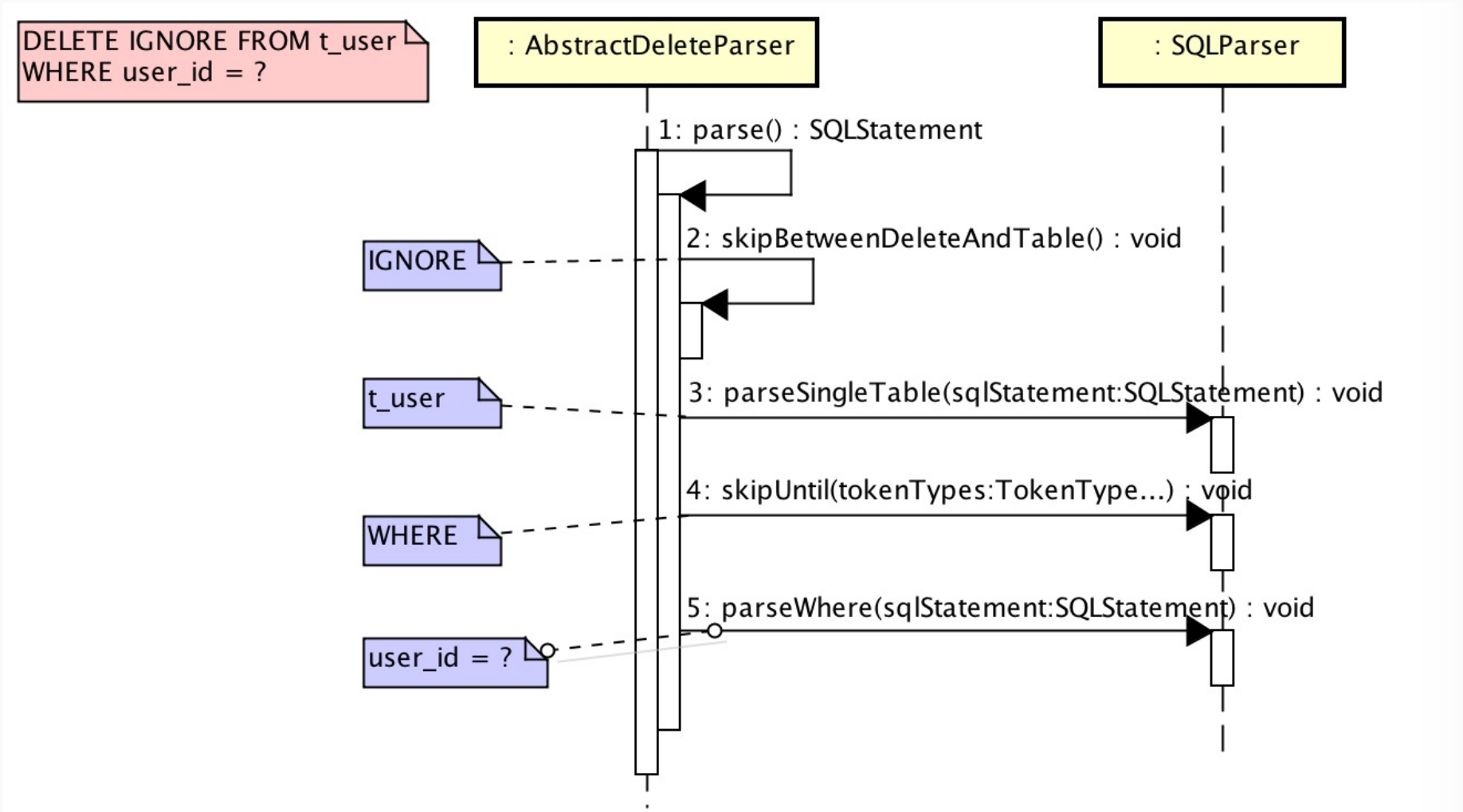
- 第二种：**Multiple-table syntax**

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    tbl_name[.*] [, tbl_name[.*]] ...
    FROM table_references
    [WHERE where_condition]

【OR】
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    FROM tbl_name[.*] [, tbl_name[.*]] ...
    USING table_references
    [WHERE where_condition]
```

Sharding-JDBC 目前仅支持第一种。业务场景上使用第二种的很少很少。

Sharding-JDBC 更新SQL解析主流程如下：




```
// AbstractDeleteParser.java
@Override
public DeleteStatement parse() {
    sqlParser.getLexer().nextToken(); // 跳过 DELETE
    skipBetweenDeleteAndTable(); // // 跳过关键字，例如：MYSQL 里的 LOW_PRIORITY、IGNORE 和 FROM
    sqlParser.parseSingleTable(deleteStatement); // 解析表
    sqlParser.skipUntil(DefaultKeyword.WHERE); // 跳到 WHERE
```

```
    sqlParser.parseWhere(deleteStatement); // 解析 WHERE
    return deleteStatement;
}
```

> **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**

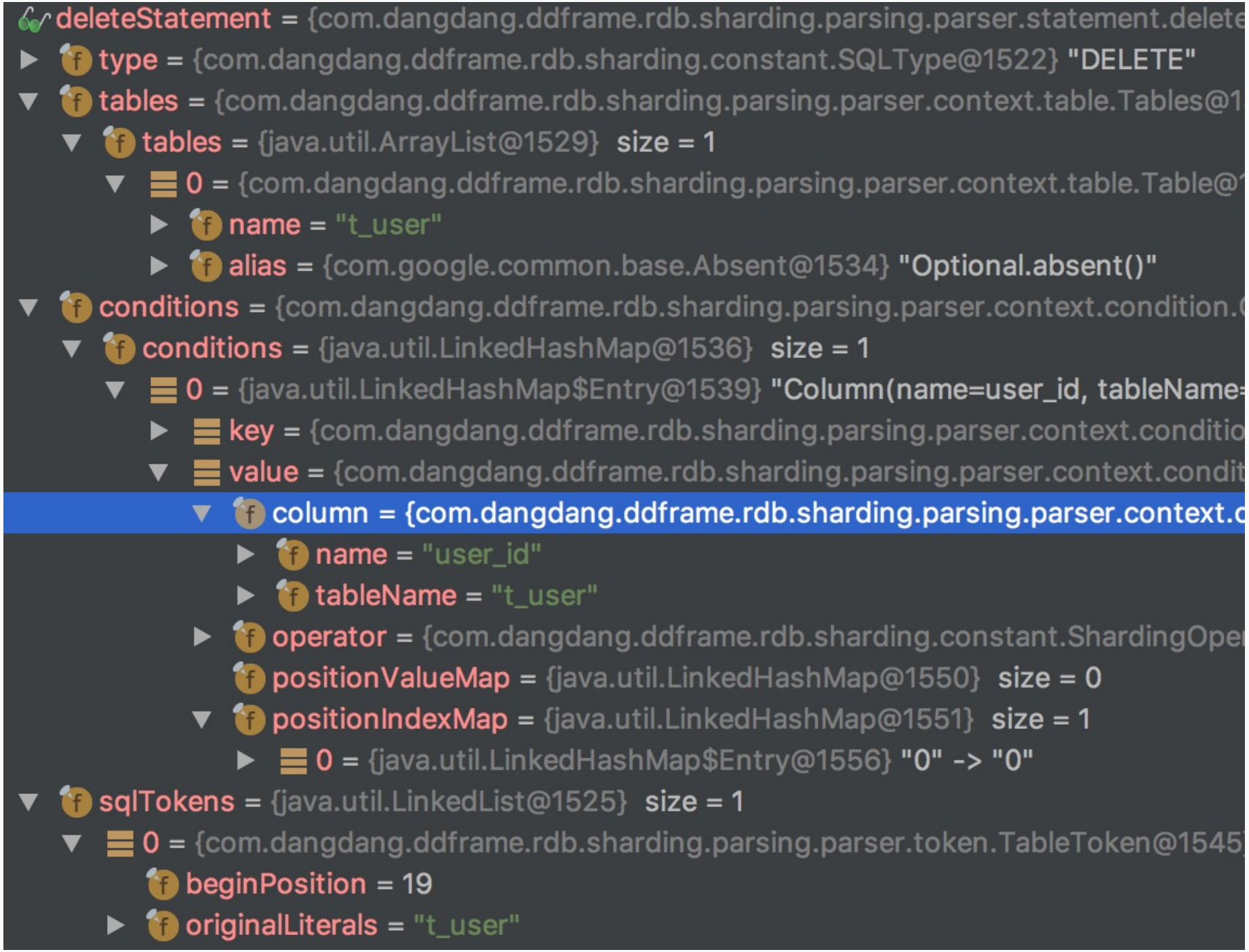
2. DeleteStatement

删除SQL 解析结果。

```
public final class UpdateStatement extends AbstractSQLStatement {
}
```

😈 对，没有其他属性。

我们来看下 `DELETE IGNORE FROM t_user WHERE user_id = ?` 的**解析结果**：


deleteStatement = {com.dangdang.ddframe.rdb.sharding.parsing.parser.statement.delete
type = {com.dangdang.ddframe.rdb.sharding.constant.SQLType@1522} "DELETE"
tables = {com.dangdang.ddframe.rdb.sharding.parsing.parser.context.table.Tables@1
tables = {java.util.ArrayList@1529} size = 1
0 = {com.dangdang.ddframe.rdb.sharding.parsing.parser.context.table.Table@
name = "t_user"
alias = {com.google.common.base.Absent@1534} "Optional.absent()"
conditions = {com.dangdang.ddframe.rdb.sharding.parsing.parser.context.condition.
conditions = {java.util.LinkedHashMap@1536} size = 1
0 = {java.util.LinkedHashMap$Entry@1539} "Column(name=user_id, tableName=
key = {com.dangdang.ddframe.rdb.sharding.parsing.parser.context.conditio
value = {com.dangdang.ddframe.rdb.sharding.parsing.parser.context.condit
column = {com.dangdang.ddframe.rdb.sharding.parsing.parser.context.c
name = "user_id"
tableName = "t_user"
operator = {com.dangdang.ddframe.rdb.sharding.constant.ShardingOper
positionValueMap = {java.util.LinkedHashMap@1550} size = 0
positionIndexMap = {java.util.LinkedHashMap@1551} size = 1
0 = {java.util.LinkedHashMap$Entry@1556} "0" -> "0"
sqlTokens = {java.util.LinkedList@1525} size = 1
0 = {com.dangdang.ddframe.rdb.sharding.parsing.parser.token.TableToken@1545
beginPosition = 19
originalLiterals = "t_user"

3. #parse()

3.1 #skipBetweenDeleteAndTable()

在 `DELETE` 和 表名 之间有些词法，对 SQL 路由和改写无影响，进行跳过。

```
// MySQLDeleteParser.java
@Override
protected void skipBetweenDeleteAndTable() {
    getSqlParser().skipAll(MySQLKeyword.LOW_PRIORITY, MySQLKeyword.QUICK, MySQLKeyword.IGNORE);
    getSqlParser().skipIfEqual(DefaultKeyword.FROM);
}
// OracleDeleteParser.java
@Override
protected void skipBetweenDeleteAndTable() {
    getSqlParser().skipIfEqual(DefaultKeyword.FROM);
    getSqlParser().skipIfEqual(OracleKeyword.ONLY);
}
```

3.2 #parseSingleTable()

解析**表**，请看《SQL 解析（二）之SQL解析》的 `#parseSingleTable()` 小节。

3.3 #parseWhere()

解析 WHERE 条件。解析代码：《SQL 解析（二）之SQL解析》的#parseWhere()小节。

666. 彩蛋

道友，帮我分享一波怎么样？

后面 SQL 路由和改写会更加有趣哟！

Sharding-JDBC

PREVIOUS:
« Sharding-JDBC 源码分析 —— SQL 路由（一）之分库分表配置
NEXT:
» Sharding-JDBC 源码分析 —— SQL 解析（五）之更新SQL

分享