|  |
| --- |
| 360BUY |
| SQL解析优化模块设计文档 |
|  |
|  |
| **秦波** |
| **2013/7/12** |

|  |
| --- |
|  |

目录

[1、 本模块设计目标 2](#_Toc361412706)

[2、 sql解析语法树结构 3](#_Toc361412707)

[2.1 INSERT语法树 3](#_Toc361412708)

[2.2 SELECT语法树 4](#_Toc361412709)

[3、 sql执行计划优化 8](#_Toc361412710)

[3.1 理论基础 8](#_Toc361412711)

[3.2 Select语句优化 8](#_Toc361412712)

[3.3 所依赖的表信息 8](#_Toc361412713)

[3.4 主要处理流程 9](#_Toc361412714)

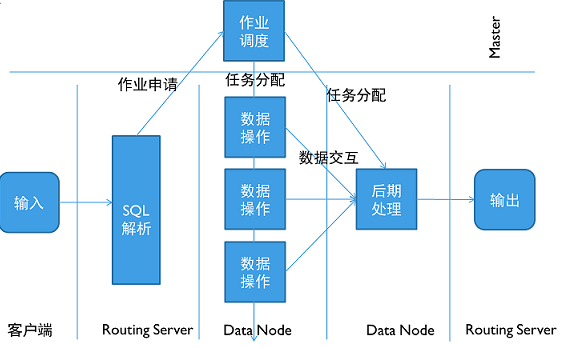
# 本模块设计目标

完善SQL解析，结构化输入输出

工作要求：

1. MySQL查询语句->语法树->目标执行SQL
2. 查询语句合法性检验
3. 定义语法树结构
4. 生成包含并行语义的执行计划
5. 支持附加系统限制（比如自动加查询的limit语句等）

模块所在位置示意图如下：



# sql解析语法树结构

## 2.1 INSERT语法树

insert语句语法树固定有5个子节点，

分别为：

1. 要插入的目的表名
2. 列名
3. 相对应的列值
4. 列值是否来自相应的query查询结果
5. 关键字是否是replacement

**举例1：列值非来自select语句**

INSERT INTO Persons (LastName, Address) VALUES ('Wilson', 'Champs-Elysees')

<<Part 2 : PARSE TREE>>

|-T\_STMT\_LIST

|-T\_INSERT

|-T\_IDENT : persons

|-T\_COLUMN\_LIST

|-T\_IDENT : lastname

|-T\_IDENT : address

|-T\_VALUE\_LIST

|-T\_VALUE\_VECTOR

|-T\_STRING : Wilson

|-T\_STRING : Champs-Elysees

|-NULL

|-T\_BOOL : FALSE

**举例2：列值非来自select语句**

<<Part 1 : SQL STRING>>

INSERT INTO table2(id, name, address) SELECT id, name, address FROM table1

<<Part 2 : PARSE TREE>>

|-T\_STMT\_LIST

|-T\_INSERT

|-T\_IDENT : table2

|-T\_COLUMN\_LIST

|-T\_IDENT : id

|-T\_IDENT : name

|-T\_IDENT : address

|-NULL

|-T\_SELECT

|-NULL

|-T\_PROJECT\_LIST

|-T\_PROJECT\_STRING

|-T\_IDENT : id

|-T\_PROJECT\_STRING

|-T\_IDENT : name

|-T\_PROJECT\_STRING

|-T\_IDENT : address

|-T\_FROM\_LIST

|-T\_IDENT : table1

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-T\_BOOL : FALSE

## 2.2 SELECT语法树

select语句语法树固定有14个子节点，

分别为：

1）记录ALL或者DISTINCT关键词处理

2）select clause

3）from clause(JOIN的处理在此处)

4）where clause

5）group by排序

6）having clause

7）set operation （记录UNION类型 T\_SET\_UNION）

8）记录ALL或者DISTINCT关键词处理

9）former select stmt （UNION关键词前一个sql语句，INTERCEPT和EXECPT关键词不支持）

10）later select stmt （UNION关键词后一个sql语句，INTERCEPT和EXECPT关键词不支持）

11）order by排序

12）limit语句

13）for update 是否需要索

14）hints /\*默认不支持\*/

**举例1：非JOIN select语句**

<<Part 1 : SQL STRING>>

SELECT Customer,SUM(OrderPrice) FROM Orders WHERE Customer='Bush' OR Customer='Adams' GROUP BY Customer HAVING SUM(OrderPrice)>1500

<<Part 2 : PARSE TREE>>

|-T\_STMT\_LIST

|-T\_SELECT

|-NULL

|-T\_PROJECT\_LIST

|-T\_PROJECT\_STRING

|-T\_IDENT : customer

|-T\_PROJECT\_STRING

|-T\_FUN\_SUM

|-NULL

|-T\_IDENT : orderprice

|-T\_FROM\_LIST

|-T\_IDENT : orders

|-T\_OP\_OR

|-T\_OP\_EQ

|-T\_IDENT : customer

|-T\_STRING : Bush

|-T\_OP\_EQ

|-T\_IDENT : customer

|-T\_STRING : Adams

|-T\_EXPR\_LIST

|-T\_IDENT : customer

|-T\_OP\_GT

|-T\_FUN\_SUM

|-NULL

|-T\_IDENT : orderprice

|-T\_INT : 1500

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

**举例2：带JOIN select语句**

<<Part 1 : SQL STRING>>

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo FROM Persons INNER JOIN Orders ON Persons.Id\_P = Orders.Id\_P ORDER BY Persons.LastName

<<Part 2 : PARSE TREE>>

|-T\_STMT\_LIST

|-T\_SELECT

|-NULL

|-T\_PROJECT\_LIST

|-T\_PROJECT\_STRING

|-T\_OP\_NAME\_FIELD

|-T\_IDENT : persons

|-T\_IDENT : lastname

|-T\_PROJECT\_STRING

|-T\_OP\_NAME\_FIELD

|-T\_IDENT : persons

|-T\_IDENT : firstname

|-T\_PROJECT\_STRING

|-T\_OP\_NAME\_FIELD

|-T\_IDENT : orders

|-T\_IDENT : orderno

|-T\_FROM\_LIST

|-T\_JOINED\_TABLE

|-T\_JOIN\_INNER

|-T\_IDENT : persons

|-T\_IDENT : orders

|-T\_OP\_EQ

|-T\_OP\_NAME\_FIELD

|-T\_IDENT : persons

|-T\_IDENT : id\_p

|-T\_OP\_NAME\_FIELD

|-T\_IDENT : orders

|-T\_IDENT : id\_p

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-NULL

|-T\_SORT\_LIST

|-T\_SORT\_KEY

|-T\_OP\_NAME\_FIELD

|-T\_IDENT : persons

|-T\_IDENT : lastname

|-T\_SORT\_ASC

|-NULL

|-NULL

|-NULL

# sql执行计划优化

## 3.1 理论基础

**Insert和Replace语句的分解**

按照PartitionKey将数据分组；

**Delete, Update和简单Select语句**

分析where子句，运用集合论

1) 结合律：a\_cond OR b\_cond AND c\_cond等价于a\_cond OR (b\_cond AND c\_cond)

2) 分配律：(a\_cond OR b\_cond) AND c\_cond等价于a\_cond AND c\_cond OR b\_cond AND c\_cond

在此过程中按照PartitionKey生成最终的执行计划；

## 3.2 Select语句优化

1）建立AS别名映射关系；

2）解析group by /order by/limit参数，并保存；

3）如果既有group by又有order by，将order by删去；

4）如果没有分布式表，不管单表还是多表，不拆分；

否则包含分布式表，则：

如果是单表查询：没有where语句则发送到所有的片；有where语句则根据where\_condition分析，并结合路由方式，构造发送到相应片的语句；

如果是多表查询：没有where语句则发送到所有的片；有where语句则根据where\_condition分析，并结合路由方式，构造发送到相应片的语句；

如果是select count(distinct ...)，且涉及分布式表，将命令改写成SELECT <field\_name> FROM <table\_name> GROUP BY <field\_name>

5）UNION关键词的作为两个语句进行上述处理，对每个语句单独分析

## 3.3 所依赖的表信息

1. 是否是分布式表
2. 若是分布式表，其划分信息（hash方式，范围方式）
3. 表的所有子片的信息（包括存储子片的数据节点）

## 3.4 主要处理流程

流程图如下所示：