逻辑算子

> 由朱悦铭于2025年设计
>
> 项目框架贡献者张子阳

1. 概念

逻辑算子是数据库查询优化和执行的基础模块，将SQL语句的声明式语义转化为可操作的树形结构，例如将SELECT-FROM-WHERE 转化为 Scan -> Filter -> Project 的算子链。它也是SQL语句与物理计划的中间转化的桥梁。其主要分为一下几个过程：

1. 从程序入口找到LogicalPlanner逻辑计划生成器。
2. SQL语句解析- 词法分析及语法分析，能够将 SQL 语句转换为可遍历的编程语言的可遍历结构。
3. DDL执行器 - 元数据管理。
4. 逻辑计划树构建。
5. 查询优化：handleSelect里查询优化过程。

2. 关键库文件

1. JSqlParser

在LogicalPlanner中，首先需要解析SQL语句，这里我们通过`JSqlParser`完成。

JSqlParser是一个开源的 Java 库，主要用于 SQL 语句的解析（Parsing）和重构。它的核心功能确实包括 **词法分析（Lexical Analysis）和语法分析（Syntax Analysis）**，能够将 SQL 语句转换为可遍历的 Java 对象结构（通常是抽象语法树，AST），从而方便程序进一步分析和处理 SQL。

创建方式：

```
JSqlParser parser = new CCJSqlParserManager();
```

2. DDLExecutor

```
public interface DDLExecutor {
    void execute() throws DBException;
}
```

生成逻辑算子的时候，会生成一些Executor， 例如DDL package里面的内容：create table ， show database 等。

Executor实际上不对数据进行修改，对表的结构进行修改。

3. Statement

net.sf.jsqlparser.statement; 是通过JsqlParser解析SQL后得倒的实例。 Statement作为一个interface，其主要的实现类如下：

实现类	解释
Select	查询语句
Insert	插入语句
Update	更新语句
Delete	删除语句
CreateTable	建表语句
Alter	修改表语句
ExplainStatement	执行计划分析语句
ShowStatement	Show 相关语句， 例如 show database;
ShowTableStatement	例如： show tables;
DescribeStatement	例如：describe tables;

4. LogicalOperator

`LogicalOperator` 是一个抽象类，用于表示数据库查询计划中的 **逻辑算子（Logical Operator）**是一个树状结构，它是数据库查询优化和执行过程中的核心抽象之一。 它的主要作用包括

```
import java.util.List;

public abstract class LogicalOperator {
    protected List<LogicalOperator> childern;

    public LogicalOperator(List<LogicalOperator> children) {
        this.childern = children;
    }

    public List<LogicalOperator> getChildren() {
        return childern;
    }

    public LogicalOperator getChild() {
        if (childern != null && !childern.isEmpty()) {
            return childern.get(0);
        }
        return null;
    }
```

```
    public abstract String toString();
}
```

- `protected List<LogicalOperator> children`
  由于存储当前算子的子算子。
- `public LogicalOperator getChild()`
  获取当前算子的第一个子算子。
- `public abstract String toString()`
  可视化当前算子的表现形式，可以用来返回当前算子的样式。

3. 执行计划树

通过我们自定义代码内部的算子调用逻辑，来生成执行计划树。

update 语句：

```
update t set t.name = 'hel' , t.gpa = 3.85 where t.age = 19;
```

计划树：

```
UpdateOperator(table=t, columns= (name:'hel')  (gpa:3.85) , expressions=t.age =
19)
 ├── TableScanOperator(table=t)
```

select 语句：

```
select t.id, t.name from t where t.age >19;
```

计划树：

```
ProjectOperator(selectItems=[t.id, t.name])
└── LogicalFilterOperator(condition=t.age > 19)
        └── TableScanOperator(table=t)
```

实践练习：

1. 完成并设计drop table操作的逻辑算子执行计划树。
2. 完成并设计delete操作的逻辑算子执行计划树。