Logical Operator

> Desinged by ZHU Yueming in April 20th 2025.
>
> The project framework is provided by Zhang Ziyang

Introduce

1. Concept

Logical operators serve as the fundamental abstraction layer in query processing, it can transform declarative SQL into executable operator trees, such as mapping language constructs to operational primitives (eg. SELECT-FROM-WHERE change to Scan -> Filter -> Project). It also serves as intermediate representation between SQL query and physical plans.

1. Find the LogicalPlanner generator from the program entry.
2. SQL statement parsing - lexical analysis (词法分析) and syntax analysis (语法分析), which can convert SQL statements into traversable structures of traversable programming languages.
3. DML executor - metadata management.
4. Construct the Logical plan tree.
5. Query optimization: query optimization process in handleSelect.

2. Important Files

1. JSqlParser

In LogicalPlanner, we first need to parse **SQL statements**, which we can do with `JSqlParser`.

JSqlParser is an open source Java library that is mainly used for parsing and refactoring SQL statements. Its core functions do include **lexical analysis (Lexical Analysis) and syntax analysis (Syntax Analysis)**, which can convert SQL statements into traversable Java object structures (usually abstract syntax trees, ASTs), making it easier for programs to further analyze and process SQL.

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

2. DMLExecutor

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

When generating logical operators, some Executors will be generated, such as the content in the DML package: create table, show database, etc.

The Executor actually does not modify the data, but modifies the structure of the table.

3. Statement

net.sf.jsqlparser.statement; It is an instance obtained by parsing SQL through JsqlParser. Statement is an interface, and its main implementation classes are as follows:

Implement Classes	Describe
Select	select statement
Insert	insert statement
Update	update statement
Delete	delete statement
CreateTable	create table statement
Alter	alter table statement
ExplainStatement	query plan statement
ShowStatement	Show statement, such as `show database;`
ShowTableStatement	such as `show tables;`
DescribeStatement	such as ``` ``describe tables; ```

4. LogicalOperator

`LogicalOperator` is an abstract class used to represent the **Logical Operator** in the database query plan. It is a tree structure and one of the core abstractions in the **database query optimization** and **execution process**. Its main functions include

```java
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`
  Used to store the sub operators of current operator.
- `public LogicalOperator getChild()`

  Get the first sub operator of current operator.
- `public abstract String toString()`

  Visualize the representation of the current operator, which can be used to return the style of the current operator.

3. Execution Plan Tree

The execution plan tree is generated through the operator call logic in user-defined code.

update statement:

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

Plan Tree:

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

select Statement:

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

Plan Tree:

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

Exercises:

1. Complete the design of the logical operator execution plan tree for the drop table operation.
2. Complete the design of the logical operator execution plan tree for the delete operation.