Report 7.1

Hivilake Class Diagram Sql Wrapper Revision

Le Nhu Chu Hiep

September 8, 2020

1. Introduction

This report is part of Hivilake project. It clarifies the Sql Wrapper structure of the storage layer. The SqlWrapper brings query for data storage. Therefore, the developer can discover the storage by the query language and leverage the query power to work with data. In previous version, the SqlWrapper has a poor designing, so in this revision, it was re-designed to remove the ambiguous design as well as allow the easier implementation.

2. Objective

The SqlWrapper is not the sql storage engine. It provides the storage wrapper into the data store so that the user can access tabular data as query style. It can be considered as a discovery tool allowing the convenient access for the tabular data in storage blog.
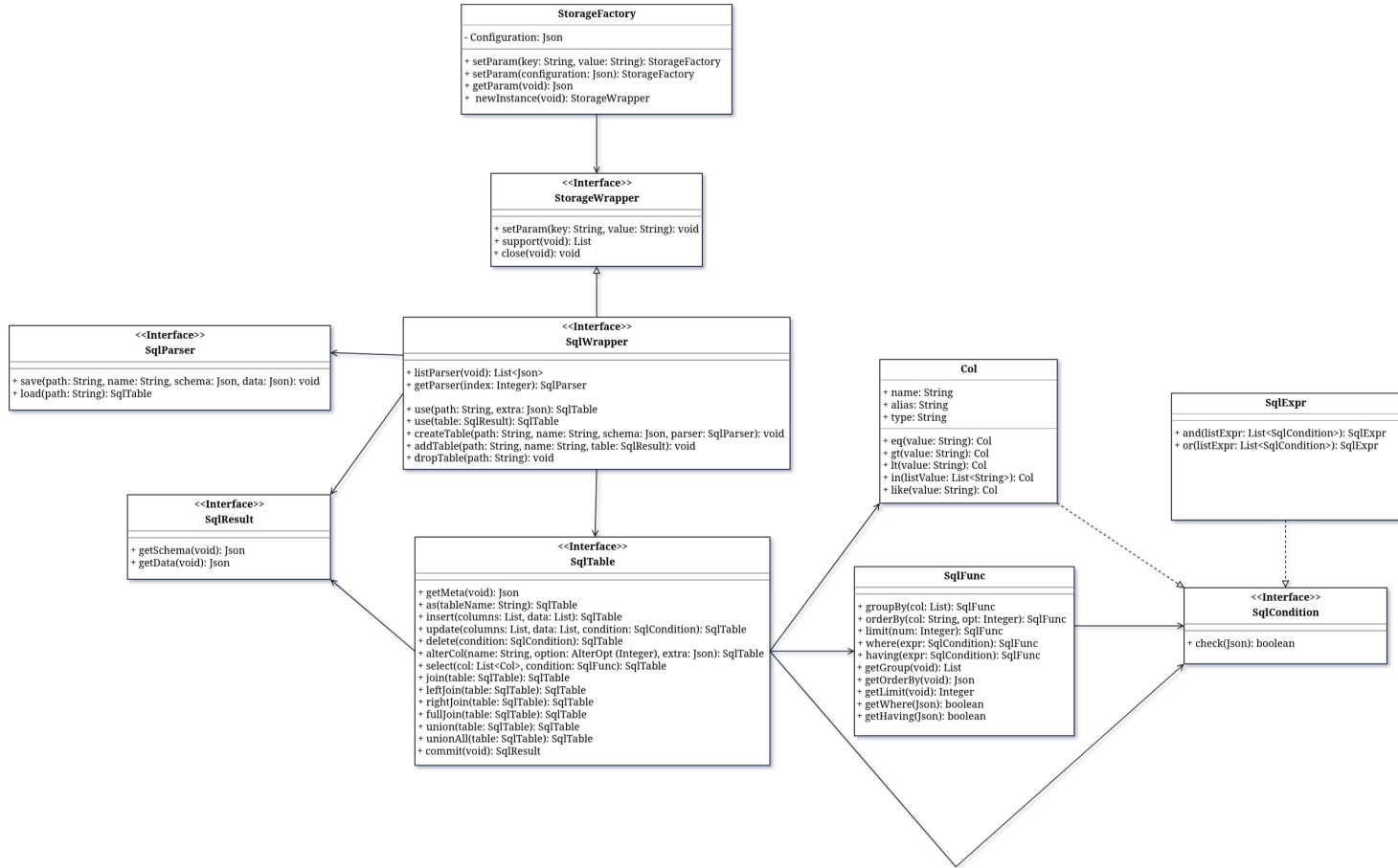
3. Methodology

StorageFactory

- Configuration: Json

+ setParam(key: String, value: String): StorageFactory
+ setParam(configuration: Json): StorageFactory
+ getParam(void): Json
+ newInstance(void): StorageWrapper

**<<Interface>>
StorageWrapper**

+ setParam(key: String, value: String): void
+ support(void): List
+ close(void): void

**<<Interface>>
SqlParser**

+ save(path: String, name: String, schema: Json, data: Json): void
+ load(path: String): SqlTable

**<<Interface>>
SqlWrapper**

+ listParser(void): List<Json>
+ getParser(index: Integer): SqlParser

+ use(path: String, extra: Json): SqlTable
+ use(table: SqlResult): SqlTable
+ createTable(path: String, name: String, schema: Json, parser: SqlParser): void
+ addTable(path: String, name: String, table: SqlResult): void
+ dropTable(path: String): void

**<<Interface>>
SqlResult**

+ getSchema(void): Json
+ getData(void): Json

Col

+ name: String
+ alias: String
+ type: String

+ eq(value: String): Col
+ gt(value: String): Col
+ lt(value: String): Col
+ in(listValue: List<String>): Col
+ like(value: String): Col

SqlExpr

+ and(listExpr: List<SqlCondition>): SqlExpr
+ or(listExpr: List<SqlCondition>): SqlExpr

**<<Interface>>
SqlTable**

+ getMeta(void): Json
+ as(tableName: String): SqlTable
+ insert(columns: List, data: List): SqlTable
+ update(columns: List, data: List, condition: SqlCondition): SqlTable
+ delete(condition: SqlCondition): SqlTable
+ alterCol(name: String, option: AlterOpt (Integer), extra: Json): SqlTable
+ select(col: List<Col>, condition: SqlFunc): SqlTable
+ join(table: SqlTable): SqlTable
+ leftJoin(table: SqlTable): SqlTable
+ rightJoin(table: SqlTable): SqlTable
+ fullJoin(table: SqlTable): SqlTable
+ union(table: SqlTable): SqlTable
+ unionAll(table: SqlTable): SqlTable
+ commit(void): SqlResult

SqlFunc

+ groupBy(col: List): SqlFunc
+ orderBy(col: String, opt: Integer): SqlFunc
+ limit(num: Integer): SqlFunc
+ where(expr: SqlCondition): SqlFunc
+ having(expr: SqlCondition): SqlFunc
+ getGroup(void): List
+ getOrderBy(void): Json
+ getLimit(void): Integer
+ getWhere(Json): boolean
+ getHaving(Json): boolean

**<<Interface>>
SqlCondition**

+ check(Json): boolean

Figure 1: SqlWrapper