DB代码说明报告

# 目录结构

项目采用模块化设计，主要目录结构如下：

sql：SQL 引擎，包含 parser（SQL 语句解析）和 optimizier（执行计划优化）

executor：执行引擎，负责执行 SQL 生成的执行计划，包含 operator（执行算子，执行计划的最小单位）

storage：存储引擎，负责数据的存储和组织，包含 transaction（事务管理）

network：网络通信模块

common：公共组件及工具函数

catalog：元数据管理，包含表、索引、函数等元信息

tests：测试用例目录

# 核心模块说明

## 1. SQL 引擎

#### 1.1 解析器（parser）

基于 Python 的 sly 或 ply 库实现词法和语法解析

支持 SELECT、INSERT、DELETE、UPDATE、CREATE TABLE、CREATE INDEX 等 SQL 语句

定义了抽象语法树（AST）节点结构，包括 Select、OrderBy、Join、Identifier 等

#### 1.2 优化器（optimizier）

实现逻辑计划到物理计划的转换

提供多种物理算子选择策略：

扫描算子：优先使用索引，支持覆盖索引扫描（CoveredIndexScan）、索引扫描（IndexScan）和表扫描（TableScan）

排序算子：支持内排序和外排序

聚合算子：基于哈希的聚合操作（HashAgg）

连接算子：嵌套循环连接（NestedLoopJoin）

## 2. 执行引擎（executor）

提供执行计划的执行入口 exec\_plan 函数

实现多种物理算子的执行逻辑：

扫描算子：处理表和索引的扫描操作

排序算子：实现数据排序功能

聚合算子：处理 GROUP BY 相关的聚合计算

连接算子：执行表之间的连接操作

DML 算子：处理插入、删除、更新等数据操作

## 3. 存储引擎（storage）

#### 3.1 数据存储

采用 slotted page 结构管理页内数据，支持记录的插入、删除、查询和更新

页大小固定为 8KB，包含页头（PageHeader）、槽目录（slot\_directory）和记录数据（records）

#### 3.2 索引结构

实现 B + 树索引，支持节点的序列化和反序列化

叶子节点存储键值对，非叶子节点存储键和子节点指针

#### 3.3 缓存管理

实现 LRU（最近最少使用）缓存策略，管理内存中的数据页

支持缓存的 pin/unpin 操作，保证正在使用的数据不被淘汰

#### 3.4 事务管理

提供事务的开始、提交和回滚功能

支持事务范围内的数据操作原子性

#### 4. 元数据管理（catalog）

管理表信息（CatalogTableForm）：包含表名、列名和列类型

管理索引信息（CatalogIndexForm）：包含索引名、关联表名和索引列

管理函数信息（CatalogFunctionForm）：包含函数名、参数数量和回调函数

# 关键功能实现

## 1. 执行计划生成与执行

逻辑计划通过递归方式转换为物理计划

物理计划采用树形结构组织，每个节点为一个执行算子

执行过程通过 open-next-close 接口模式实现，支持流式数据处理

## 2. 事务支持

事务范围内的 DML 操作（插入、删除、更新）具有原子性

支持事务提交（commit）和回滚（abort），保证数据一致性

## 3. 索引使用策略

遵循最左匹配原则选择合适的索引

在存在多个候选索引时，优先选择列数最少的索引

当查询涉及的列都包含在索引中时，使用覆盖索引扫描

# 测试用例

项目包含丰富的测试用例，覆盖：

执行计划生成测试

各种算子功能测试（扫描、排序、聚合、连接等）

事务操作测试

页存储功能测试

LRU 缓存测试