# Entry-task 简易mysql编译器设计文档

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 修订人 | 说明 |
| V 1.0 | 2020.10.27 | 张小帅 | 拟制 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[Entry-task 简易mysql编译器设计文档 1](#_Toc54696522)

[1、文档介绍 2](#_Toc54696523)

[1.1文档目的 2](#_Toc54696524)

[1.2 项目背景 2](#_Toc54696525)

[1.3 需求概述 2](#_Toc54696526)

[1.4 文档范围 2](#_Toc54696527)

[1.5 读者对象 2](#_Toc54696528)

[1.6 参考文档 2](#_Toc54696529)

[2、运行环境&架构 3](#_Toc54696530)

[2.1 运行环境 3](#_Toc54696531)

[2.2 系统架构 3](#_Toc54696532)

[3、功能设计 3](#_Toc54696533)

[3.1存储功能设计 3](#_Toc54696534)

[3.2 词法&语法解析设计 4](#_Toc54696535)

[3.3 TreeWalker设计 4](#_Toc54696536)

[3.4 业务逻辑设计 4](#_Toc54696537)

[3.4.1 表抽象设计 4](#_Toc54696538)

[3.4.2 where clause 抽象设计 5](#_Toc54696539)

[3.4.3 select clause 抽象设计 5](#_Toc54696540)

[3.5 输入输出设计 5](#_Toc54696541)

# 1、文档介绍

## 1.1文档目的

以技术角度解释系统实现的逻辑与思路，通过该文档，能够更方便的与需求人员交流，验证需求的满足程度，也方便后续的维护人员更容易理解系统。

## 1.2 项目背景

SQL简易编辑器为入司的Entry Task，更多的意义在于通过项目了解一些知识以及代码熟练。SQL简易编辑器给予Antlr4方便的进行语法解析，借此了解一些编译原理知识以及通用语言模式。

## 1.3 需求概述

完成sql的建表逻辑与查询逻辑。支持的字段有int，double，string三种。支持嵌套查询，支持SUM聚合函数。

## 1.4 文档范围

文档仅包含针对本项目的设计思路，不包括具体的实现逻辑以及部署情况。

## 1.5 读者对象

本文档期待的读者对象为项目开发人员，以及维护人员。

## 1.6 参考文档

<https://github.com/antlr/antlr4/blob/master/doc/index.md>

# 2、运行环境&架构

## 2.1 运行环境

软件环境：MacOS + JRE1.8以上

硬件环境：个人Mac

## 2.2 系统架构

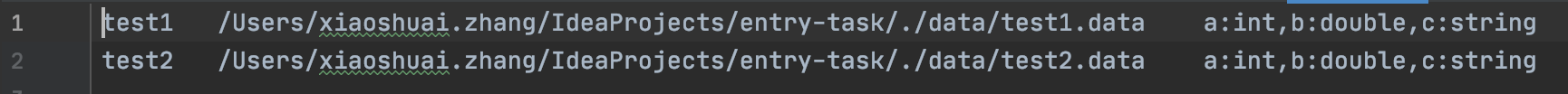
图形用户界面

描述已自动生成

# 3、功能设计

## 3.1存储功能设计

对于系统表数据，采用字符文本的形式进行存储。文件目录为{$projectPath}/data/字符文本分两种类型，meta文件负责存储元数据。Meta文件只每一行存储一个表的元数据信息，格式如下：



每行内容以‘\t’分割，固定有3部分内容，第一部分为表名，第二部分为表内容存储文件，第三部分为字段定义。字段定义部分格式为columnName:columnType[,columnName:columnType]\*

表内容文件存储表的具体数据，每行对应表中的row，字段之间以符号‘\t’分割，不存储columnName。示例如下：

图片包含 户外, 街道, 城市, 标志

描述已自动生成

## 3.2 词法&语法解析设计

词法&语法通过antlr4进行解析，antlr4的基础语法规则设计文档《ETSql.g4》

## 3.3 TreeWalker设计

语法树的遍历，采用Listener，而不是采用Visitor。因为功能需求比较简单，采用Listener将会减少系统实现的复杂度。TreeWalker将会采用深度优先遍历语法树。对每一个规则，都会提供enterRuleName()和exitRuleName()方法，并提供子树信息的Context对象。

## 3.4 业务逻辑设计

### 3.4.1 表抽象设计

采用Stream<String>作为表的抽象表示。这种设计方法有以下几处优点：

* 屏蔽物理表和子查询结果表的区别，物理表和子查询都可以抽象为Stream<String>结构
* 简化where Clause和select clause的实现操作，根据Stream的特性，where clause可以抽象为filter操作，select clause可以抽象为map操作。
* 有利于子查询性能，在子查询中没有聚合函数时，子查询没有单独计算，节约了子查询时的计算时间和存储空间。

### 3.4.2 where clause 抽象设计

Where clause 中的条件语句，全部抽象为一个Predicate<String>类型的functional interface对象，对象接收一个String line并返回boolean值。

### 3.4.3 select clause 抽象设计

Select子句可以抽象为Function<String,String>的接口对象完成列的选择，但是select语句中有可能有聚合函数SUM的存在，因此分为3种情况。

1. Select clause中没有聚合函数

这种情况下，select clause转化为Function<String,String>对象完成列选择

1. Select clause中仅有聚合函数

这种情况下，select clause 转化为Function<String,Int/Double>,并且需要在stream上执行reduce(sum)方法得到聚合值，然后重新进行stream

1. Group by的情况

这种情况下，select clause 转化为 Function<String,string[]>,String[] 中有两个字段，前一个为group by条件字段，后一个为sum字段。转化后，需要在stream对象上调用foreach，将结果转移到map中进行计算，（转移与计算同步进行，因此不会占用太大内存）。最后将map结果进行stream。

## 3.5 输入输出设计

虽然没有并发需求，但是每一条输入的sql都采用了独立的线程。Sql执行输入会终止线程，但是主程序不会推出。输入符号’\q’推出程序。