基于您提供的完整前后端源码，我为您整理了当前**Web版达梦数据库管理系统**的系统设计文档。

# 系统设计与架构文档

## 1. 系统概述

本系统是一个轻量级、基于Web的**达梦数据库（Dameng DB）管理工具**。它允许用户通过浏览器连接多个数据库实例，进行可视化的数据浏览、增删改查（CRUD）、表结构管理、SQL执行以及ER图关系可视化。

## 2. 技术栈 (Tech Stack)

| **领域** | **技术/框架** | **用途** |
| --- | --- | --- |
| **前端** | **Vue.js 2.x** | 核心前端框架 |
|  | **Element UI** | UI 组件库 (布局, 表格, 弹窗, 表单) |
|  | **AntV G6** | 图可视化引擎 (用于绘制 ER 关系图) |
|  | **Axios** | HTTP 客户端 (处理 API 请求与拦截器) |
| **后端** | **Spring Boot 2.7+** | 核心后端框架 |
|  | **MyBatis** | ORM 框架 (处理动态 SQL) |
|  | **HikariCP** | 数据库连接池 |
|  | **Dameng JDBC** | 达梦数据库驱动 |
| **架构** | **AbstractRoutingDataSource** | 实现多数据源动态切换 |
|  | **ThreadLocal** | 实现请求级别的数据库上下文隔离 |

## 3. 整体架构设计

系统采用典型的 **B/S 架构**，后端采用无状态（Stateless）设计，通过请求头携带连接标识来实现多数据源切换。

### 3.1 架构图 (逻辑视图)

代码段

graph TD  
 User[用户浏览器] -->|HTTP Request + Conn-Id| Nginx[Nginx (反向代理/静态资源)]  
 Nginx -->|API Request| Controller[Spring Boot Controller]  
   
 subgraph "后端核心层 (Backend)"  
 Controller -->|Set Context| DbContext[ThreadLocal Context]  
 Controller -->|Call| Mapper[MyBatis Mapper Interfaces]  
 Mapper -->|Get Connection| ConnMgr[ConnectionManager (RoutingDataSource)]  
 ConnMgr -->|Read Context| DbContext  
 ConnMgr -->|Switch DS| Pool{连接池 Map}  
 end  
   
 subgraph "数据层 (Data Layer)"  
 Pool -->|JDBC| DB1[达梦数据库 A]  
 Pool -->|JDBC| DB2[达梦数据库 B]  
 Pool -->|JDBC| DB3[达梦数据库 ...]  
 end

### 3.2 核心设计模式

1. **动态数据源路由 (Dynamic Routing DataSource)**:
   * **实现原理**: 继承 AbstractRoutingDataSource。
   * **机制**: 每一个用户建立的连接（Host/Port/User）在后端都会生成一个独立的 HikariDataSource 实例，存储在 ConcurrentHashMap 中。
   * **切换**: 每次 HTTP 请求头中携带 Conn-Id。拦截器或 Controller 将其存入 ThreadLocal。MyBatis 执行 SQL 获取连接时，determineCurrentLookupKey() 从 ThreadLocal 读取 ID，从而找到对应的数据源。
2. **动态 SQL 拼接**:
   * 由于是通用数据库工具，表名、列名无法预知。系统大量使用了 MyBatis 的 ${} 语法进行 Schema 和 Table 的动态拼接。
   * *注：当前设计依赖后端对参数的基本校验，存在 SQL 注入风险（待优化项）。*
3. **前端视图互斥与缓存**:
   * 利用 Vue 的 :key 和 v-if 强制刷新 DOM，解决 G6 Canvas 残留问题。
   * 利用 keep-alive 或 el-tabs 保持多窗口状态。

## 4. 功能模块详解

系统主要包含以下四大核心模块：

### 4.1 连接与元数据管理 (Sidebar)

* **多连接管理**: 支持创建、编辑、删除多个数据库连接信息。
* **懒加载树形目录**:
  + 一级：连接 (Root)。
  + 二级：模式 (Schemas)。
  + 三级：分类文件夹 (数据表、视图、触发器)。
  + 四级：具体对象 (Table/View/Trigger)。
* **状态展示**: 在树形菜单中直接展示触发器的启用/禁用状态（颜色标识）。

### 4.2 表数据管理 (Data View)

* **数据浏览**: 分页展示数据（目前为前端分页，后端限制 Limit 1000）。
* **高级筛选**: 支持多条件组合（AND/OR）、模糊查询 (LIKE)、空值查询 (IS NULL) 等。
* **CRUD 操作**:
  + **新增**: 弹窗表单录入。
  + **编辑**: 弹窗表单修改。
  + **删除**: 单行删除。
* **智能冲突检测**:
  + 在删除或修改主键时，后端会自动递归查询 ALL\_CONSTRAINTS。
  + 如果存在外键引用，会拦截操作并返回具体的**冲突表名、关联字段和冲突记录数**。
  + 前端提供“去处理”按钮，支持一键跳转到冲突表并自动筛选相关数据。
* **级联更新**: 支持模拟级联更新（插入新记录 -> 更新子表引用 -> 删除旧记录）。

### 4.3 可视化与分析 (Advanced Views)

* **ER 关系图**:
  + 基于 AntV G6 引擎。
  + 自动分析当前表的一级外键关联（父表和子表）。
  + 可视化展示表结构、主键、外键连线。
  + **交互**: 双击节点可强制跳转到该表的数据视图。
* **SQL 终端**:
  + 提供基础的 SQL 输入框。
  + 支持执行任意 SQL（SELECT 返回表格，DDL/DML 返回执行结果）。

### 4.4 结构定义管理 (DDL)

* **可视化建表**: 提供 GUI 界面添加列、设置类型、长度、主键和非空约束。
* **创建视图/触发器**: 提供 SQL 模板编辑器。
* **对象删除**: 支持右键或按钮删除表、视图和触发器。

## 5. 当前数据流转示例

**场景：用户点击“删除某一行数据”**

1. **前端**: 用户点击删除 -> TableDetail.vue 获取行 ROWID -> 发送 DELETE /api/db/delete 请求（携带 Conn-Id）。
2. **后端入口**: DbController 接收请求 -> DbContext.setConnId() 设置当前线程上下文。
3. **数据源路由**: MyBatis 请求数据库连接 -> ConnectionManager 从 DbContext 获取 ID -> 返回对应的 HikariDataSource。
4. **业务逻辑**:
   * DbMapper 执行删除。
   * **异常捕获**: 如果数据库抛出 引用约束 异常。
   * **冲突分析**: catch 块中调用 getAllChildTables 查询谁引用了该表，并循环 countReference 统计引用数量。
5. **响应**:
   * 成功：返回 200 OK。
   * 冲突：返回 503 状态码 + 冲突详情列表。
6. **前端**: 根据 503 状态码弹出“冲突警告框”，展示引用表列表。

## 6. 系统现状总结

* **完成度**: 高。已经具备了数据库管理工具的核心闭环功能。
* **兼容性**: 深度适配达梦数据库（系统视图查询做了针对性调整，如 SYSOBJECTS, ALL\_tab\_comments）。
* **体验**: 解决了复杂的视图切换残留 Bug，实现了人性化的外键冲突引导。

这是一个结构清晰、功能实用的垂直领域数据库管理工具雏形。