根据您提供的完整代码库（前端 Vue + 后端 Spring Boot/MyBatis），系统已经具备了一个\*\*数据库管理工具（MVP版本）\*\*的核心雏形。整体架构清晰，涵盖了连接管理、元数据浏览、CRUD操作、ER图可视化和基础SQL执行。

为了将其打磨成一个成熟、健壮的生产级工具，我为您梳理了以下**改进建议**和**后续开发路线**。

### 一、 核心功能增强 (High Priority)

#### 1. SQL 编辑器升级 (Monaco Editor)

目前的 SQL 终端使用的是普通的 el-input type="textarea"，体验较差。

* **改进方案**：引入 **Monaco Editor** (VS Code 的核心编辑器) 或 **CodeMirror**。
* **功能**：
  + **语法高亮**：SQL 关键字变色。
  + **智能提示**：输入表名、字段名时自动补全（前端需要从后端获取元数据缓存）。
  + **格式化**：一键美化 SQL。
* **实现思路**：前端引入 monaco-editor 组件，替换 sql-box 中的 el-input。

#### 2. 真·服务端分页 (Performance)

当前逻辑是 LIMIT 1000，然后在前端进行假分页。如果表有 100 万行数据，用户永远无法看到第 1001 行。

* **改进方案**：后端 getData 接口支持 page 和 size 参数。
* **SQL 调整 (达梦/Oracle 语法)**：  
  SQL  
  -- 使用 OFFSET FETCH (达梦较新版本)  
  SELECT \* FROM "SCHEMA"."TABLE" ORDER BY "DB\_INTERNAL\_ID"   
  OFFSET ${(page-1)\*size} ROWS FETCH NEXT #{size} ROWS ONLY;  
    
  -- 或者使用 ROWNUM (通用)
* **前端调整**：el-pagination 的 current-change 事件不再只是切片数组，而是重新调用 loadData。

#### 3. 数据导出/导入功能

目前的表格只能看和改，无法带走。

* **导出**：支持将查询结果（当前页或全量）导出为 **Excel (.xlsx)** 或 **CSV**。建议后端处理流式输出，防止 OOM。
* **导入**：支持上传 Excel/CSV，解析后批量插入数据库。

### 二、 安全性与稳定性 (Critical)

#### 1. SQL 注入防御 (Security)

在 DbMapper.java 中，大量使用了 ${} (字符串替换) 来拼接 schema 和 tableName。虽然这是动态表名场景下的无奈之举，但存在安全风险。

* **改进方案**：
  + **白名单校验**：在 Controller 层接收到 tableName 后，先去 getAllTables 的缓存中校验该表名是否存在。如果不存在，直接拒绝执行。
  + **正则校验**：严格限制 schema 和 tableName 只能包含字母、数字、下划线。

#### 2. 密码安全

目前连接信息是明文保存在前端 localStorage (DMDB\_CONNECTIONS) 中的。任何人拿到浏览器都能看到数据库密码。

* **改进方案**：
  + **后端加密**：前端只存一个 token 或加密后的配置，实际连接信息存在后端 SQLite/H2 数据库中（如果是桌面版）。
  + **最简方案**：前端存储时使用简单的 Base64 或者 AES 加密，虽不能防高手，但能防瞥视。

#### 3. 连接池管理

ConnectionManager.java 中虽然有 removeDataSource，但没有定时清理机制。如果用户打开了很多连接却不关闭，服务器内存会泄露。

* **改进方案**：引入 Guava Cache 或 Caffeine 来管理 targetDataSources，设置过期时间（例如 30 分钟无操作自动断开）。

### 三、 用户体验优化 (UX)

#### 1. DDL 查看 (逆向工程)

目前只能看到列信息，无法看到建表语句。

* **功能**：在左侧树的表上右键 -> "查看 DDL"。
* **后端**：达梦支持 DBMS\_METADATA.GET\_DDL('TABLE', 'TABLE\_NAME', 'SCHEMA')，可以通过 Mapper 调用该存储过程返回建表语句。

#### 2. 更加智能的 ER 图

目前的 ER 图是全量加载该表的一级关联。如果表关系很复杂（例如 ERP 系统），图会非常乱。

* **改进**：
  + **层级控制**：增加参数，允许查看 2 层甚至 3 层深度的关联。
  + **字段筛选**：在连线上只显示外键字段，鼠标悬停时才显示完整字段。

#### 3. 结果集编辑优化

目前的编辑是弹窗式 (handleEdit)。

* **改进**：实现 **行内编辑 (Inline Edit)**。双击单元格直接变为 input 框，焦点移开后变回文本，并记录修改状态（变色），最后点击顶部“保存”按钮一次性提交所有修改。

### 四、 代码结构重构建议

#### 1. 统一异常处理

目前 DbController 里每个方法都有 try-catch 包裹。

* **建议**：使用 Spring Boot 的 @RestControllerAdvice + @ExceptionHandler 全局异常处理。这样 Controller 代码会非常干净，只需关注业务逻辑。

#### 2. Mapper XML 化

DbMapper.java 使用了大量的 @Select("<script>...") 注解。虽然方便，但当 SQL 变得复杂（尤其是涉及动态条件和 <if> 标签嵌套时），Java 文件的可读性会变得很差。

* **建议**：对于复杂的查询（如 getColumnsForTables、queryByConditions），将其抽取到 resources/mapper/DbMapper.xml 中。

### 五、 后续开发路线图 (Roadmap)

| **阶段** | **任务** | **描述** |
| --- | --- | --- |
| **P0 (当前)** | **Bug修复与体验** | 修复视图残留 Bug (已完成)，优化 Loading 状态，添加简单的 SQL 语法高亮。 |
| **P1** | **SQL能力增强** | 引入 Monaco Editor，支持 SQL 历史记录，支持多语句执行 (;分隔)。 |
| **P2** | **数据能力增强** | 实现服务端分页，支持 Excel 导出，实现简单的行内编辑。 |
| **P3** | **结构管理** | 支持查看/修改 DDL，支持“设计表”UI界面（可视化修改字段类型/长度）。 |
| **P4** | **多库支持** | 抽象 Mapper 层，适配 MySQL、PostgreSQL，不仅仅局限于达梦。 |

您现在的代码基础非常好，结构分层合理。如果按照上述路线图迭代，这完全可以发展成一个类似 DBeaver 或 Navicat 的轻量级 Web 版数据库管理工具。加油！