

作品类别：系统软件

作品链接：<https://forgeplus.trustie.net/Relaen/Relaen>

Relaen 框架技术报告

一、 框架说明

Relaen 是一款全新基于 Node.js 环境的 ORM 框架，支持 TypeScript 和 JavaScript 开发使用。框架提供高效的开发方式，帮助开发者简化数据库访问操作，提高开发效率。Relaen 支持 Active Record 和 Data Mapper 模式，可以以灵活的选择编写高质量的、松耦合的、可扩展的、可维护的应用程序。Relaen 参考了很多优秀的 ORM 框架的设计与实现，比如 Hibernate，Sequelize 和 TypeORM。

目前支持 MySQL、Oracle Database 12c+、Microsoft SQL Server 2012+、PostgreSQL 10+、MariaDB、SQLite 数据库。

二、 框架设计方案

一般业务开发中，会编写大量数据访问层的代码，对数据库进行保存、读取、修改和删除信息，而这些代码多大都是重复的。而使用 ORM 框架，则会大大减少重复性代码，提高开发效率。并且避免在代码中嵌入大量 SQL 语句，降低程序的耦合度，提高可维护性。

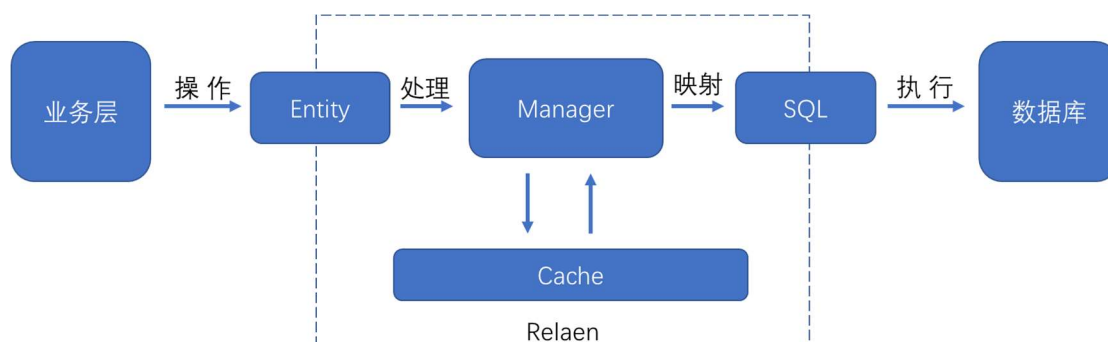


图 1 框架简要流程

业务层基于 Entity 实体进行数据持久化操作，Relaen 对实体操作进行分析处理。如果为数据库 DQL 操作，先查询缓存中是否存储有实体数据，有则返回，没有根据实体注解配置生成对应执行 SQL 语句访问数据库。如果为数据库 DML 操作，则更新缓存，同上生成 SQL 语句访问数据库。

三、框架实现方案

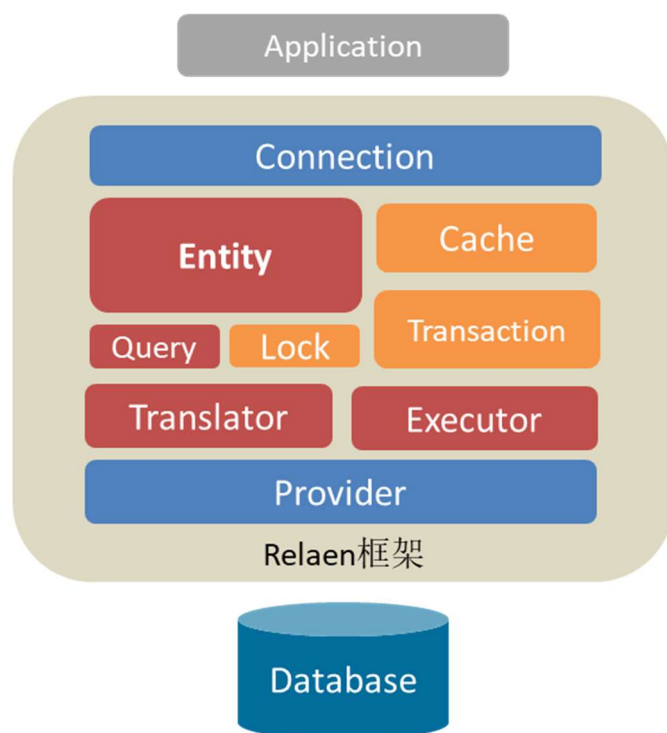


图 2 框架主体模块

3.1 核心模块

1) Entity 实体模型模块: 实体模型是 ORM 框架的核心模块, Relaen 通过定义一个类来建立对象实体和关系型数据库中的表映射关系。实体模型设计基于 TypeScript 的类, 并使用注解开发方式来标识映射关系, 支持 Active Record 和 Data Mapper 模式;

2) Query SQL 构造器: 提供灵活便捷的 SQL 构造方法, 以满足不

同复杂 SQL 的执行需求。主要分为 Query 查询构造器和 Native Query 原生查询构造器，采用链式操作构建查询条件；

3) Translator SQL 翻译器：将基于对象实体的操作转化成对应的 SQL 语句，主要转换数据库 CRUD 操作、事务和锁等；

4) Executor SQL 执行器：调用底层数据库执行转化的 SQL 语句，对数据库返回的原生结果集进行处理，根据映射关系配置转换为实体，再以实体的形式返回，保证业务层中一直操作的实体对象。

3.2 功能模块

1) Transaction 事务模块：基于数据库连接对象，提供统一的事务操作方式，提供事务隔离级的设置；

2) Cache 缓存模块：对执行过的查询操作存入缓存模块，减少相同查询操作对底层数据库的调用执行；

3) Lock 锁机制模块：基于数据库本身锁机制，提供统一的锁机制操作方式，并提供悲观锁和乐观锁模式。

3.3 底层模块

1) Connection 连接管理模块：根据配置建立与数据库的物理连接，并管理连接对象，支持数据库连接池，拟开发支持多数据库源；

2) Provider 数据库底层模块：主要完成底层关系型数据库的驱动接入，封装实现统一的框架调用方法。目前已接入常见关系型数据库：MySQL、Oracle、SQL Server、PostgreSQL、MariaDB 和 SQLite 等；

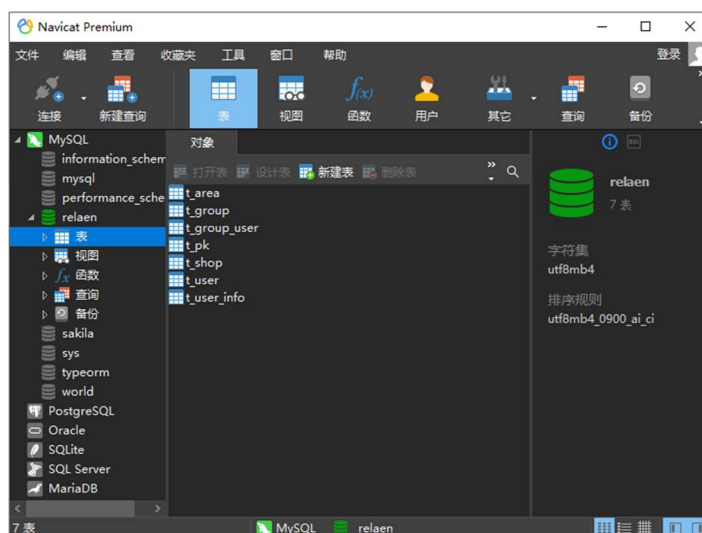
四、 框架特色创新

➤ 支持 JavaScript 和 TypeScript

- 支持 Active Record 和 Data Mapper
- 注解开发方式
- 清晰简洁的对象关系模型——实体
- 事务
- 锁机制
- 查询缓存
- 日志
- 提供自动映射生成实体 cli 工具
- 支持多款数据库, 如: MySQL、Oracle、SQL Server、PostgreSQL、MariaDB、SQLite

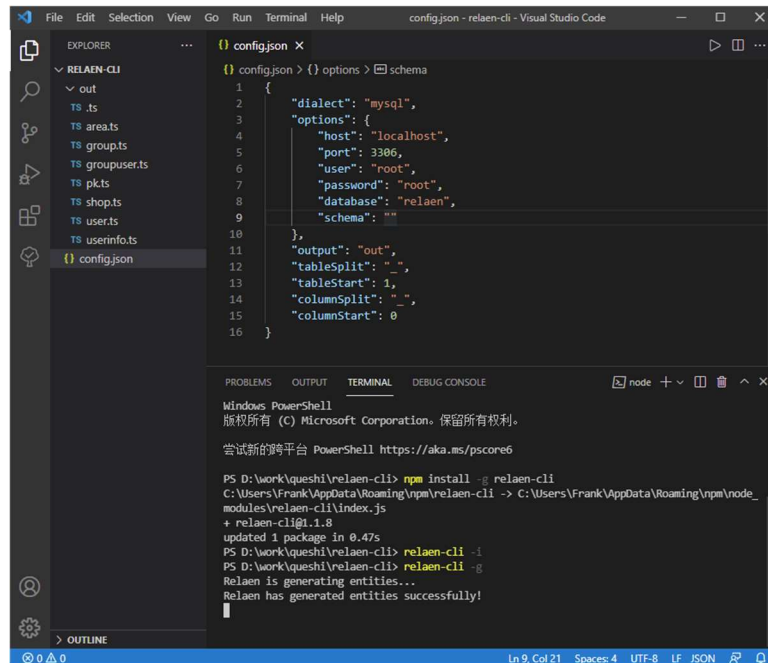
五、框架运行效果

1) 创建项目使用数据库



2) Relae-cli 工具自动生成实体

- 执行下载工具包命令: `npm install -g relaen-cli`
- 执行生成数据库连接配置文件命令: `relaen-cli -i`
- 执行实体映射生成命令: `relaen-cli -g`



```
config.json - relaen-cli - Visual Studio Code
1 {
2   "dialect": "mysql",
3   "options": {
4     "host": "localhost",
5     "port": 3306,
6     "user": "root",
7     "password": "root",
8     "database": "relaen",
9     "schema": ""
10  },
11  "output": "out",
12  "tableSplit": "-",
13  "tableStart": 1,
14  "columnSplit": "-",
15  "columnStart": 0
16 }
```

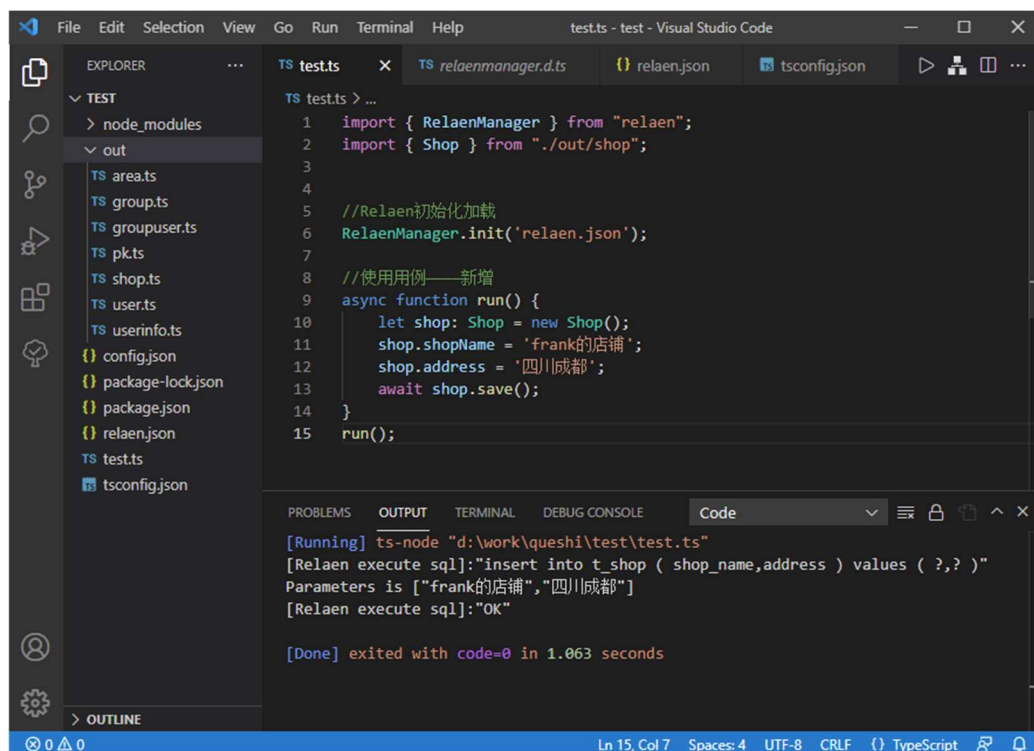
```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/powershell

PS D:\work\qeshi\relaen-cli> npm install -g relaen-cli
C:\Users\Frank\AppData\Roaming\npm\relaen-cli -> C:\Users\Frank\AppData\Roaming\npm\node_modules\relaen-cli\index.js
+ relaen-cli@1.1.8
updated 1 package in 0.47s
PS D:\work\qeshi\relaen-cli> relaen-cli -i
PS D:\work\qeshi\relaen-cli> relaen-cli -g
Relaen is generating entities...
Relaen has generated entities successfully!
```

3) Relaen 运行

- 配置项目运行环境
- 下载 Relaen 包，下载相关数据库包
- 创建数据库连接配置文件进行配置：relaen.json
- 新建 test.ts 文件运行测试



```
test.ts - test - Visual Studio Code
1 import { RelaenManager } from "relaen";
2 import { Shop } from "../out/shop";
3
4
5 //Relaen初始化加载
6 RelaenManager.init('relaen.json');
7
8 //使用用例——新增
9 async function run() {
10   let shop: Shop = new Shop();
11   shop.shopName = 'frank的店铺';
12   shop.address = '四川成都';
13   await shop.save();
14 }
15 run();
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Code
[Running] ts-node "d:\work\qeshi\test\test.ts"
[Relaen execute sql]: "insert into t_shop ( shop_name,address ) values ( ?,? )"
Parameters is ["frank的店铺","四川成都"]
[Relaen execute sql]: "OK"

[Done] exited with code=0 in 1.063 seconds
```