方案模板表结构（solution_templates）

核心架构设计

表名: solution_templates
存储位置: database/init.sql:574-629

字段分类

1️⃣ 基础字段（关系型）

- id - 主键
- template_code - 模板编码（唯一）
- template_name - 模板名称
- category - 方案类别（枚举：hydration补水/whitening美白/anti_aging抗衰/repair修复/hair_care护发/other其他）

2️⃣ 组织与权限（关系型）

- org_id - 所属机构ID（NULL表示全局模板）
- scope - 共享范围（枚举：global全局/org机构/private私有）
- 外键关联到 organizations 表

3️⃣ 动态数据（JSON存储）⭐

- suitable_skin_types - 适用肤质（JSON数组）
- suitable_problems - 适用问题（JSON数组）
- steps - 步骤说明（JSON数组）
- products - 推荐产品（JSON数组）
- services - 包含服务（JSON数组）
- case_photos - 案例照片URLs（JSON数组）

4️⃣ 描述性字段（关系型）

- target_group - 目标人群
- course_duration - 疗程时长
- treatment_frequency - 治疗频次
- expected_effects - 预期效果（TEXT）
- precautions - 注意事项（TEXT）

5️⃣ 价格与展示

- estimated_price_min/max - 预估价格区间
- cover_image - 封面图URL
- usage_count - 使用次数统计

6️⃣ 审计字段

- created_at/updated_at - 时间戳
- created_by/updated_by - 操作人（外键到users表）
- is_deleted - 软删除标记

📐 存储架构特点

混合存储模式：

- ✅ 结构化数据 → 使用MySQL关系型字段
- ✅ 半结构化/动态数据 → 使用JSON字段
- ✅ 多对多关系 → JSON数组存储（如products、services）

索引设计：
UNIQUE KEY `uk_org_code` (`org_id`, `template_code`) -- 防止重复
INDEX `idx_category` (`category`) -- 按分类查询
INDEX `idx_org_id` (`org_id`) -- 按机构查询
INDEX `idx_scope` (`scope`) -- 按范围查询
INDEX `idx_status` (`status`) -- 按状态过滤

🔗 关联关系

solution_templates
├── organizations (org_id) - 所属机构
├── users (created_by) - 创建人
└── users (updated_by) - 更新人

💡 设计优势

1. 灵活性: JSON字段存储动态数据，无需频繁修改表结构
2. 性能: 核心查询字段使用关系型+索引，保证查询效率
3. 扩展性: JSON数组可灵活添加新字段
4. 范围控制: 通过scope+org_id实现全局/机构/私有三级权限控制

这种架构既保持了关系型数据库的事务性和查询性能，又利用JSON的灵活性处理变化频繁的业务数据，是典型的现代混合存储模式

🏗️ 设计架构说明

✅ 单表设计原因

1. JSON字段替代子表

```
- steps (步骤) - JSON数组
- products (产品) - JSON数组
- services (服务) - JSON数组
- suitable_skin_types (适用肤质) - JSON数组
- suitable_problems (适用问题) - JSON数组
```

传统设计可能需要：
solution_templates (主表)
├── solution_template_steps (步骤子表)
├── solution_template_products (产品关联表)
├── solution_template_services (服务关联表)
└── solution_template_conditions (适用条件表)

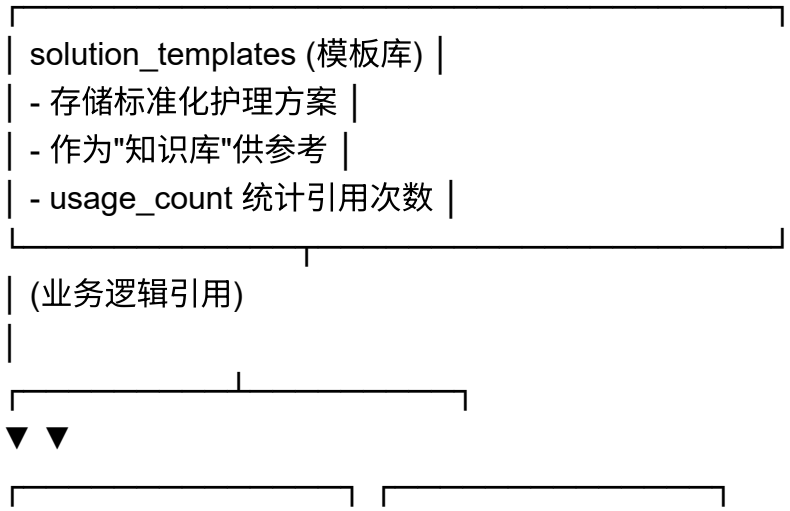
但本系统选择了 JSON存储 → 减少表数量，提高灵活性
2. 业务逻辑层关联（无外键约束）
solution_templates → 客户案例（customer_cases）
└── 通过业务逻辑引用，案例中的 treatment_plan 字段记录方案内容（文本）

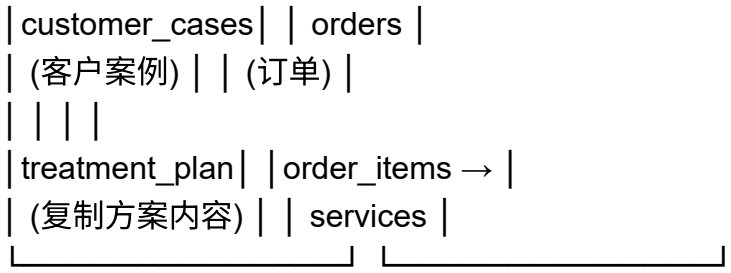
solution_templates → 订单（orders）
└── 订单明细引用具体的 services，不直接引用模板

📊 实际关联关系

● Read(api\models\SolutionTemplate.js)
⎿ Read 30 lines

● 🩹 方案模板使用流程





💡 总结

当前设计：

- ✅ 单表架构 - 仅 solution_templates 一张表
- ✅ JSON字段 - 替代传统的多表关联
- ✅ 软引用 - 通过业务逻辑（如复制模板内容到案例）而非外键
- ✅ 使用统计 - usage_count 字段记录模板被引用次数

优点：

- 简化表结构，减少JOIN查询
- 方案内容独立，修改模板不影响历史案例
- 灵活扩展JSON字段

缺点：

- 无法通过外键追溯哪些案例使用了某个模板
- 数据冗余（案例中复制了模板内容）

这是一种 "模板快照"模式 - 使用模板时创建快照副本，而不是实时引用，确保历史数据稳定性。