目前的拆分思路为以场景驱动的方式，通过监控典型应用场景测试用例的执行，获取方法调用链、SQL语句、执行频率以及执行顺序等信息，形成场景驱动的数据访问轨迹图，在此基础上通过优化算法来推荐微服务拆分方案（含数据库表拆分及相应的代码重构）。整个优化算法将综合考虑数据库表及字段在访问轨迹中的耦合及共享程度、代码重构开销等方面的因素。整体工作内容可分为场景驱动的数据访问轨迹图生成、微服务拆分方案生成、微服务拆分方案可视化及反馈调整、代码重构四个部分。

下面分别介绍各个部分的具体思路。其中的具体示例都来自于开源项目jeesite。

**一、场景驱动的数据访问轨迹图生成**

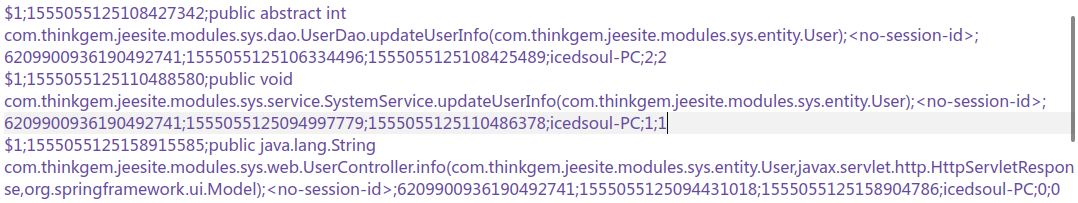
**1. 准备输入**

通过测试驱动的方式获得访问轨迹图。其中，测试既可以通过自动化的方式，也可以通过人工执行的方式实现。需要准备以下信息。

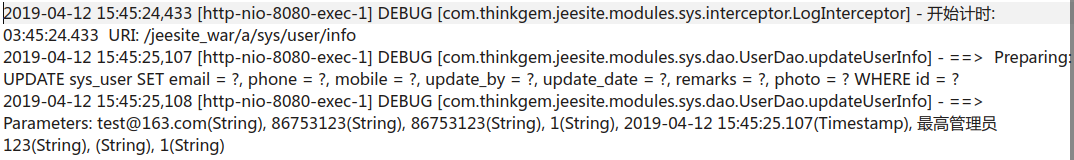
* 测试用例：测试用例为用户使用场景级别，一个场景包含一个或多个请求。例如，一个管理员发通告的场景，包含了查询办公室列表、根据办公室查询用户信息、保存通告内容、刷新页面四个请求。
* 使用频率：每个测试用例对应的用户的使用频率，可以为一个估计的相对比例；可选，没有则默认每个用例执行频率相同。

**2. 执行测试用例，监控代码执行路径和数据库操作**

* 使用监控工具获取运行时的代码执行路径，如“修改个人信息”操作会触发的代码执行路径如下：



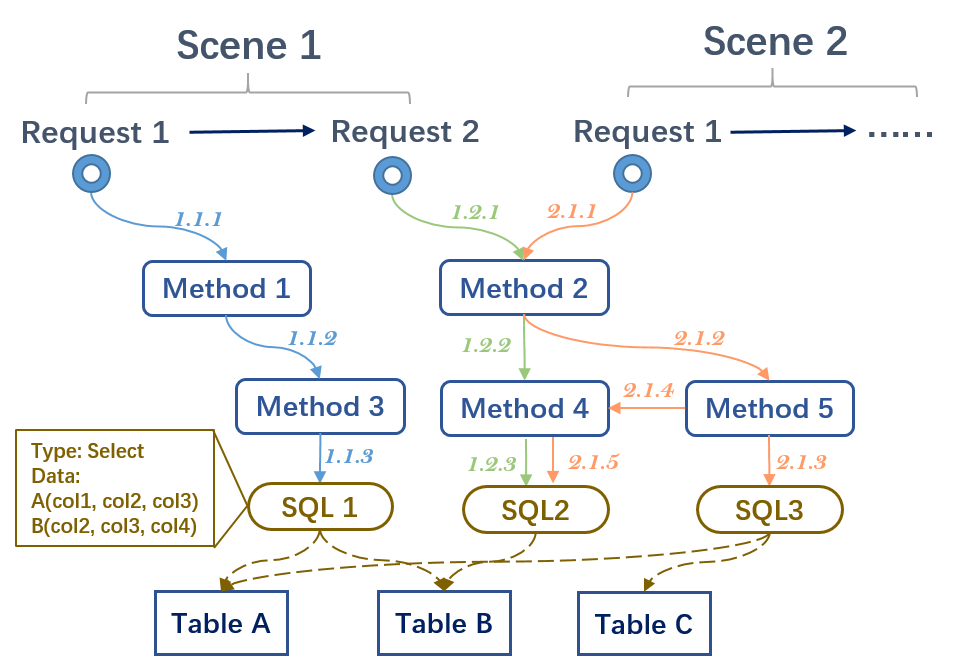
* 同时，获取DAO层方法对应执行的SQL语句，如上图中updateUserInfo方法执行了以下SQL语句：



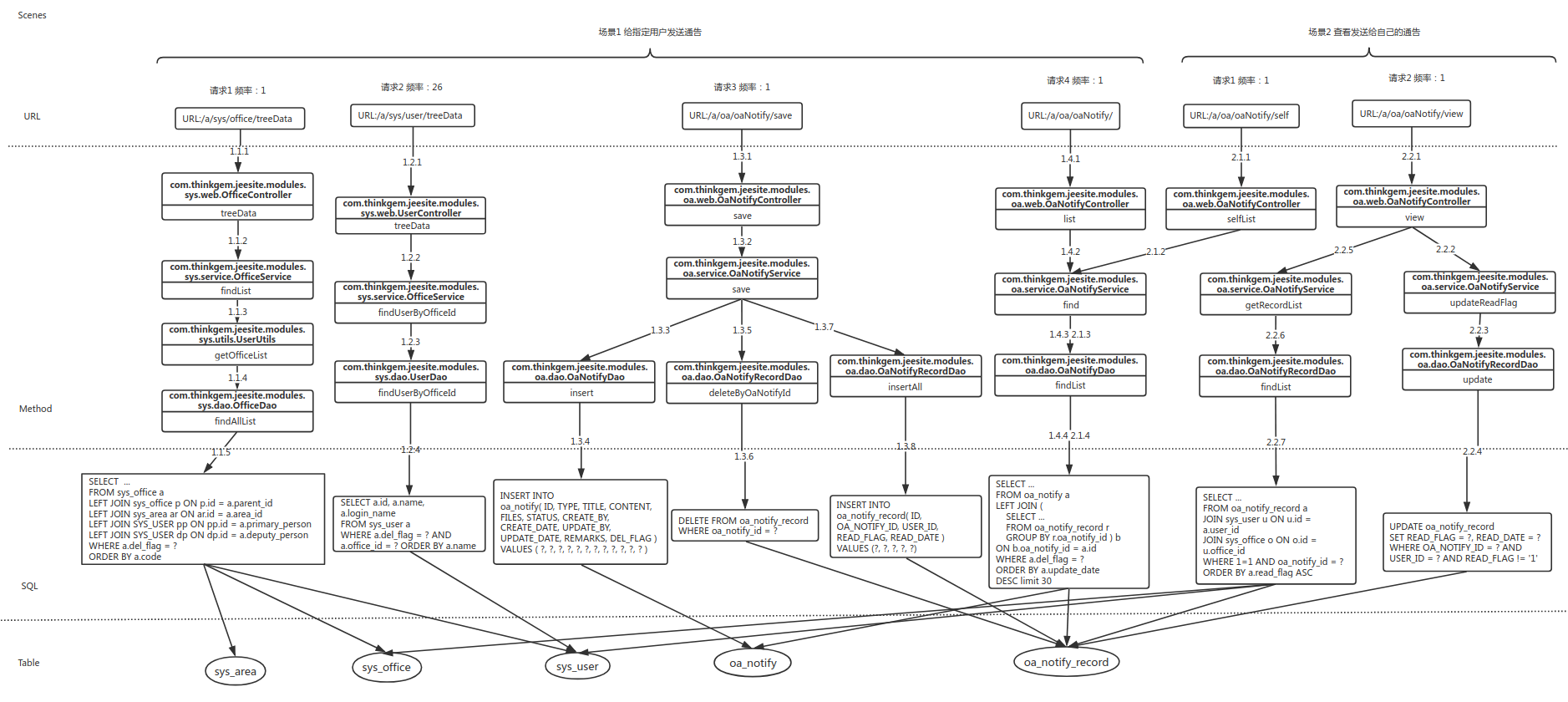
* 根据时间、方法名等信息关联代码执行路径和SQL语句，得到每个请求完整的代码执行和数据操作路径。
* 同时，也会扫描静态SQL和代码结构，作为监控数据的补充。

**3. 图生成**

根据第2步中收集到的信息作图：一个用户场景级别的测试用例（Scene）包含若干个请求（Request），每个请求会经过若干方法（Method），其中有些方法会执行SQL语句、对数据表中的某些字段进行查询或修改操作。



对于

下图为OA模块中的两个场景。左边为“给指定用户发送通告”的场景，包含四个请求，从左到右依次为查询办公室列表、根据办公室查询用户、保存通告内容、刷新页面获取所有通告。右边为“查看发送给自己的通告”场景，包含获取通告、更新通告状态两个请求。

**二、微服务拆分方案生成**

依据第一部分生成的场景驱动的数据访问轨迹图，综合考量数据间的耦合程度、代码拆分开销等多个维度，构建合适的多目标优化模型，给出多个可选的拆分方案。

**1. 维度考量**

* 关联程度：
  + 出现在同一个场景/请求/SQL中的数据（例如表oa\_notify和oa\_notify\_record，两个表经常一起被查询，且向oa\_notify插入一条新的记录后会向oa\_notify\_record插入N条记录），关联度增大。出现频率越高，关联度越大；
* 共享程度：
  + 同时被多个场景访问的数据（例如表sys\_user，大部分场景中都会出现查询系统用户信息的请求），共享度增大。共享的类型可依据数据操作类型（查询、修改等）的不同而不同；
* 拆分开销：
  + 从数据拆分向上回溯代码拆分，分析变更影响、计算拆分开销，一般情况下的开销：方法拆分>方法的移动>类的移动
* ……

**2. 图分割**

综合上述多种维度为数据表之间的边进行加权，采用多种可选的图分割算法（如聚类、最小割等）进行划分。同时，我们会调整不同维度对于权重的影响，生成多个具有不同倾向性的可选方案。以下为常见的几种图分割算法，会在此基础上进行实现和改进；

* 聚类算法：
  + 谱聚类算法：

谱聚类将带权无向图划分为两个或两个以上的最优子图，使子图内部尽量相似，而子图间尽量距离较远，以达到聚类的目的。通过计算图的拉普拉斯矩阵前k个特征值对应的特征向量进行降维，再使用K-Means聚类得到最终的分类。

* + 图团体检测：

用于社交网络中检测联系比较紧密的团体，通过定义模块性(modularity)的计算方式，每次尝试融合两个聚类、再重新计算整个网络的模块性、使得模块性增长最大。最后检查整个聚类过程汇总的所有记录，找到其中模块性最高的聚类模式。

* 最小割算法：
  + Stoer Wagner算法：

从任一顶点出发，以Prim算法扩展最大生成树，记录每次最后扩展的顶点的切割值，合并最后扩展到的两个顶点。不断重复直到所有顶点合并成一个，取记录中的最小切割方案做一次切割。对切割过程中生成的多个图反复运用Stoer Wagner算法得到最终分割结果。

**3. 多目标优化**

构建合适的多目标优化模型，如最大化数据间的内聚程度、最小化代码拆分开销等，对图分割算法得出的多个具有不同倾向性的方案进行筛选，最终得到一个帕累托最优解集合，包含一个或多个在某些维度具有优势的拆分方案。

**三、微服务拆分方案可视化及反馈调整**

对第二部分输出的拆分方案进行可视化（包含数据划分结果、代码拆分开销等）。用户根据需求选择最合适的一个方案，可进一步手动调整拆分结果，确定最终的拆分方案。

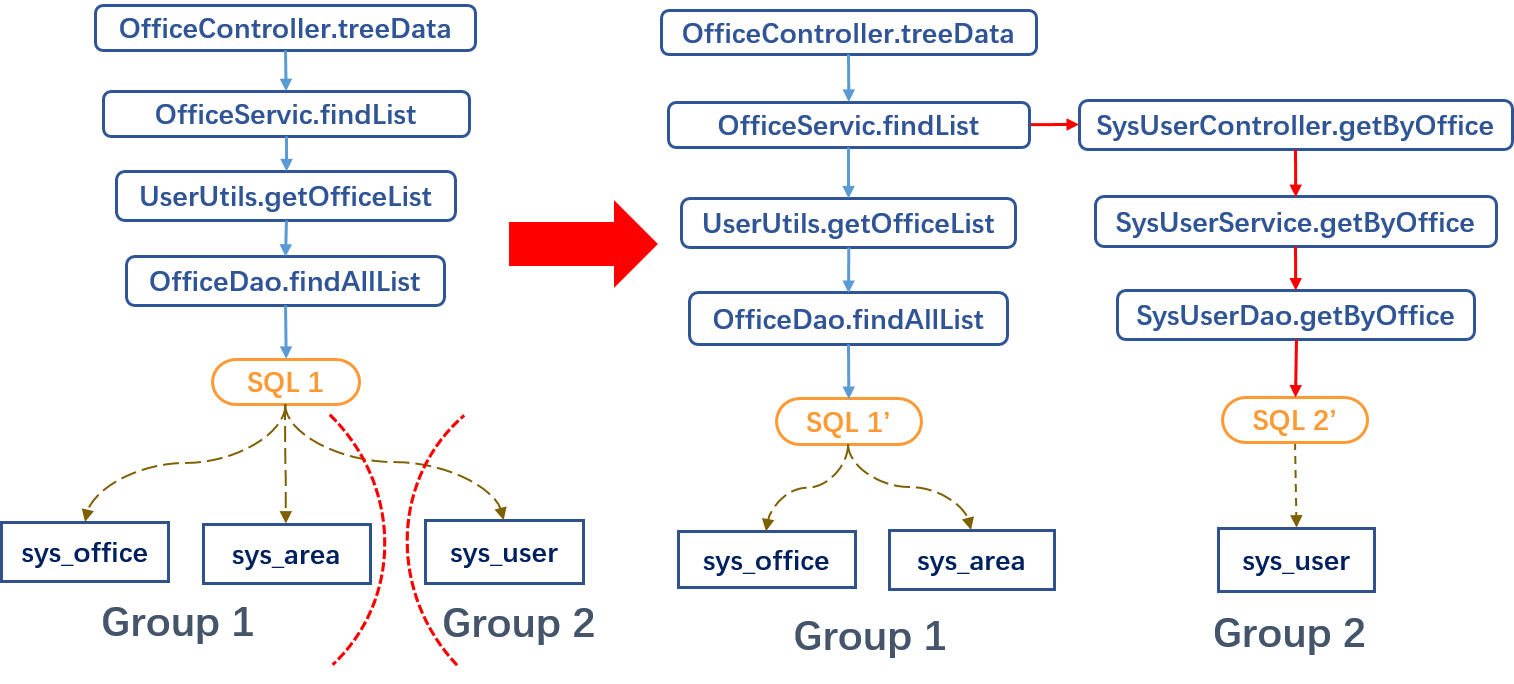
**四、代码重构**

我们目前的工具支持三种类型的原子重构操作：

* 移动方法：将一个方法从一个类移动到另一个类中；
* 提取类：将一个类分解成多个类；
* 提取属性/方法：从多个类中提取相同的属性或方法到一个已有的或新创建的抽象类或接口中。

代码重构过程基于数据划分结果、回溯代码调用路径、自底向上进行分析、逐步调用原子重构操作：

* 简单情况下，一个类中所有方法操作的数据都属于同一个微服务，则这个类被划分到该微服务；
* 如果一个类操作的数据不属于同一个微服务，则以方法为单位进行分析：
  + 如果这个方法操作的数据属于同一个微服务，则这个方法也被划分到该微服务中，视情况为该方法创建新的类（提取方法）或加入已存在的类中（移动方法）；
  + 如果一个方法操作的数据不属于同一个微服务，则涉及方法内部语句的拆分、为拆出的语句创建一个新的方法；
* 调用转化：将原本的方法调用或语句的执行替换成跨网络的服务调用

下图为拆分一条SQL语句后的代码重构示意图：