



MongoDB
中文社区

IT大咖说
知识分享平台

teambition

如何有效提升MongoDB开发者的工作效率

E叔



MongoDB
中文社区

IT大咖说
知识分享平台

E叔

周李洋（E叔）

MongoDB Master

Teambition运维总监

MongoDB上海用户会联合发起人

MongoDB在公司或团队中的定位

- 主业务数据库
- 内部平台数据库
- 日志相关数据库

在Teambition

跟随着产品的生命周期来看待
MongoDB相关工作

- 命名规则
- 数据结构设计
- 持续优化与维护数据结构
- 日常query的脱敏、审核、鉴权
- 监控

命名规范

- 数据库名约定为全小写
- 数据库名称不能包含这些字符： / \ . " \$
- 数据库名称长度要小于64个 characters
- 表名称需要以小写字符或者下划线开始
- 表名不得包含 \$， 能为空字符串 (“ ”)
- 不能包含 \0 或空字符, 这个字符表示键的结尾
- 不能以 【system.】 开头
- 表名的最大长度为120bytes包括前缀的 【databse.】 内容 内容

命名规范

- 键不可不能包含\0或空字符,这个字符表示键的结尾
- 不能以\$开头
- 不能包含. (点号)
- key区分大小写并且不能重复

表结构设计

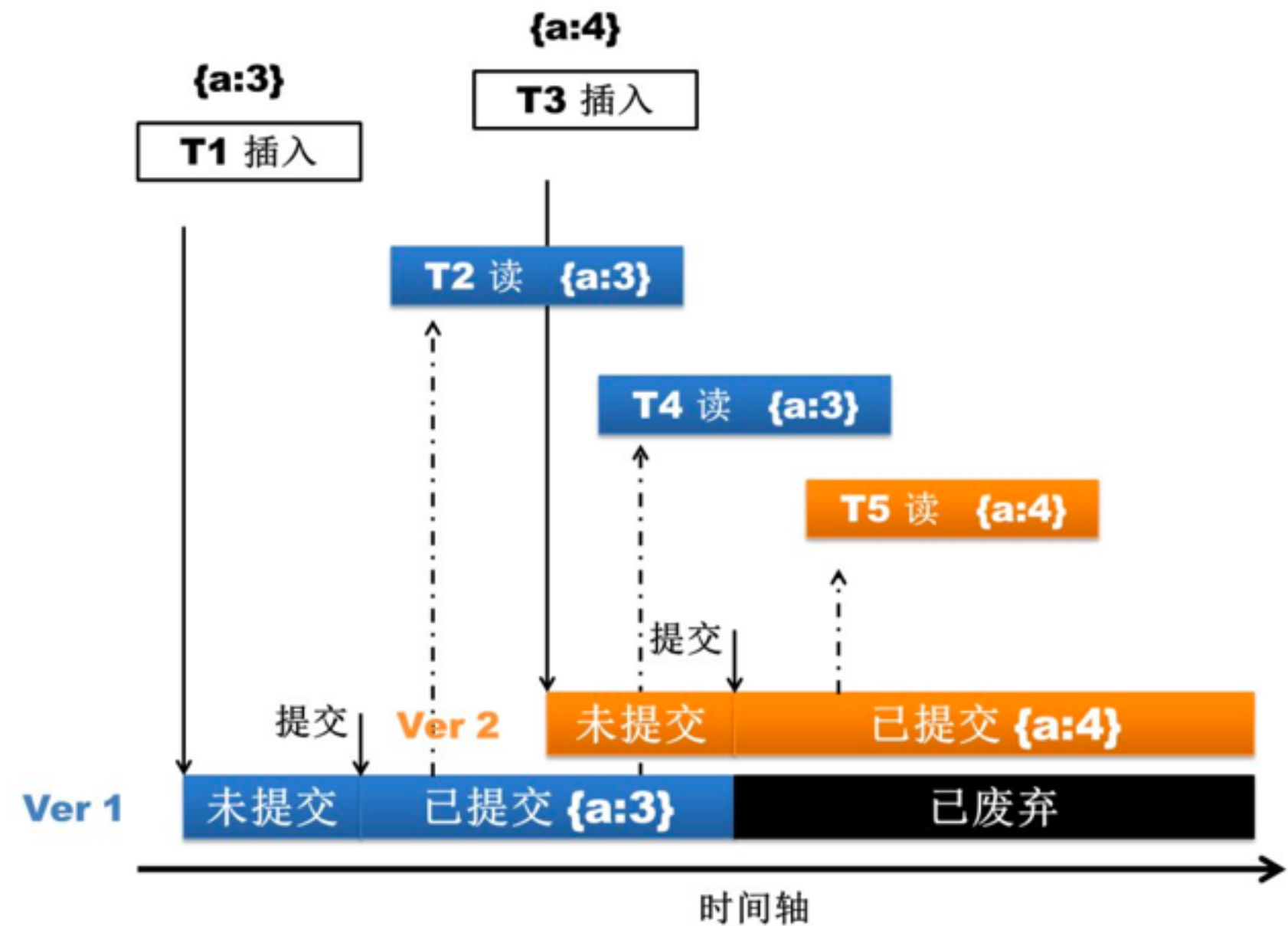
- 优先考虑内嵌
- 数组结构的增长应该是有限的
- 不要害怕应用层的join
- 清楚每个字段的读写比
- 只有贴合业务的数据结构才是最优的

优先考虑内嵌

- 优先考虑内嵌，充分发挥MongoDB的优势;但是也要注意不使用内嵌的情况：
 - 1.需要对内嵌部分数据进行独立查询
 - 2.业务预估内嵌部分的数据会有大量增长；
 - 3.内嵌部分数据已经在外部有遵循范式的表并且会被需改

数组结构的增长应当是有限的

- MongoDB 单条document大小有限
- MVCC机制下数组的变动也等同于多版本写一次；如果数组结构有特别多数据，这时候就不要以内嵌形式设计了
- 为数组建立索引的时候会为每一个element建立一个index key



不要害怕应用层join

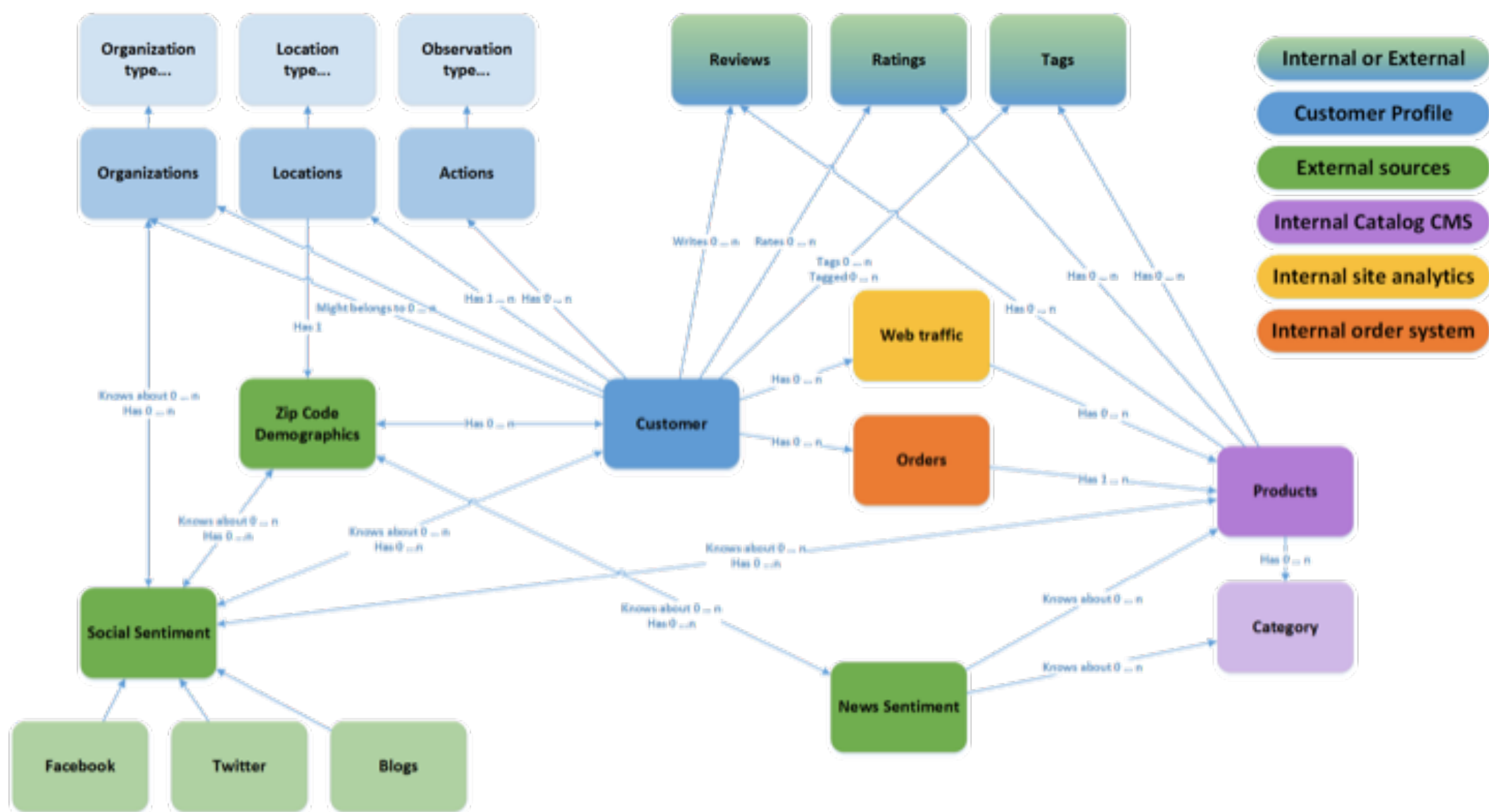
- MongoDB不同于RDBMS的传统Join，通过内嵌解决了很多需求
- 面对跨表查询需求的时候，不要违反前文提到的规则来强行内嵌，只要表结构设计得当并且有合理的index，应用层的多次查询并进行业务逻辑拼接处理也是非常高效的

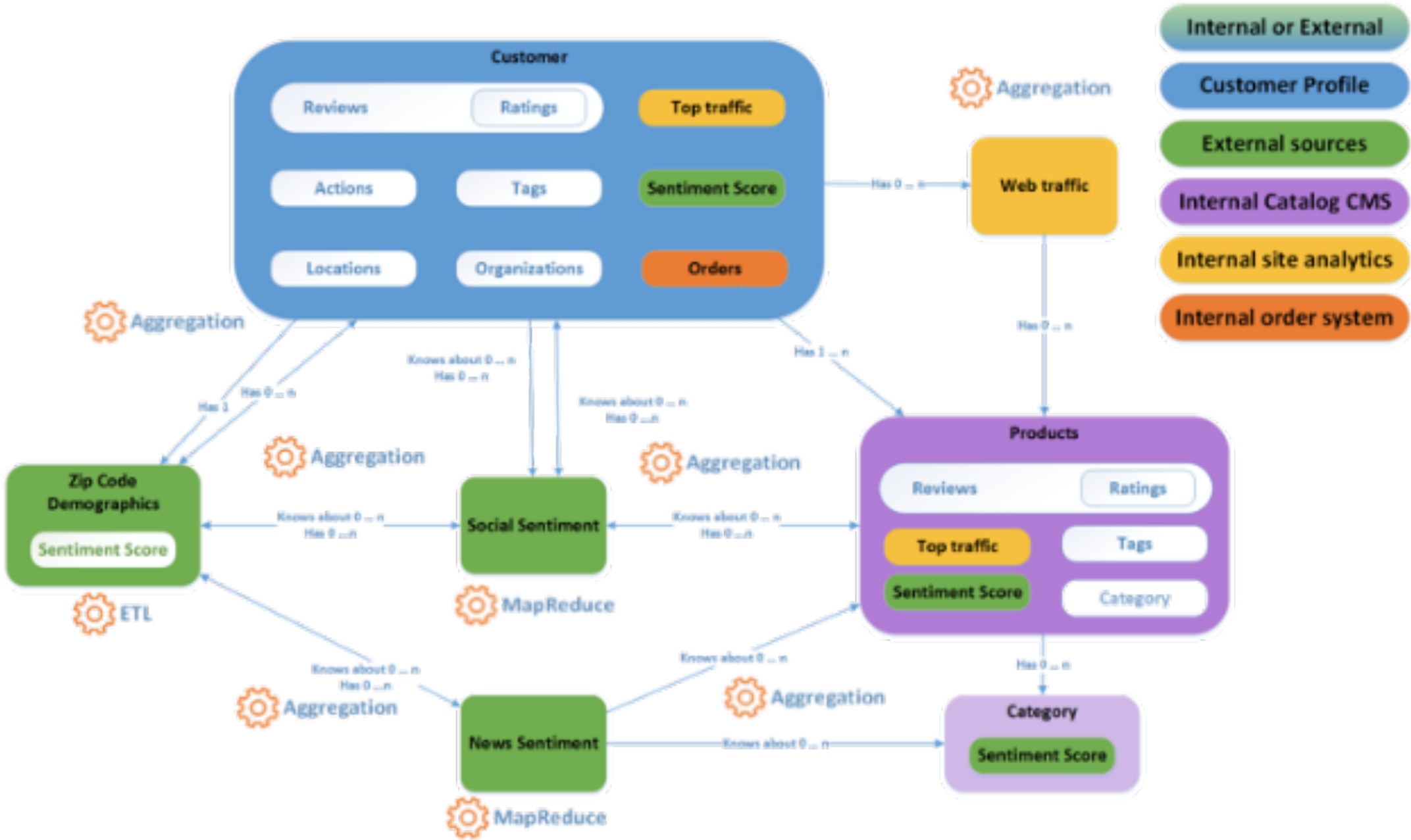
清楚每个字段的读写比

- 当进行反范式设计的时候需要考虑到字段的读写比。
- 如果是一个只被读不会被更新的则是一个“完美”的内嵌字段
- 要充分考虑到业务与数据边界情况时可能带给DB的请求情况



贴合业务的数据结构







看到这里是不是觉得都在加重开发者的负担？

上线后的持续优化

- Schema free but you may need some rules
- Variety
- MongoDB Compass
- Document Validation

Schema free but you may need some rules

- 脏数据带来的字段类型异常所造成的bug
- 有强数据类型依赖的字段因为字段异常带来的性能损失
- 带给code review不必要的麻烦

Document Validation

phone字段为string类型或者email字段要匹配”@mongodb.com”结尾，或者status为”Unknown”或者”Incomplete”

```
db.createCollection( "contacts",  
{ validator: { $or:[  
  { phone: { $type: "string" } },  
  { email: { $regex: /  
    @mongodb.com$/ } },  
  { status: { $in: [ "Unknown",  
    "Incomplete" ] } }  
]}  
]
```

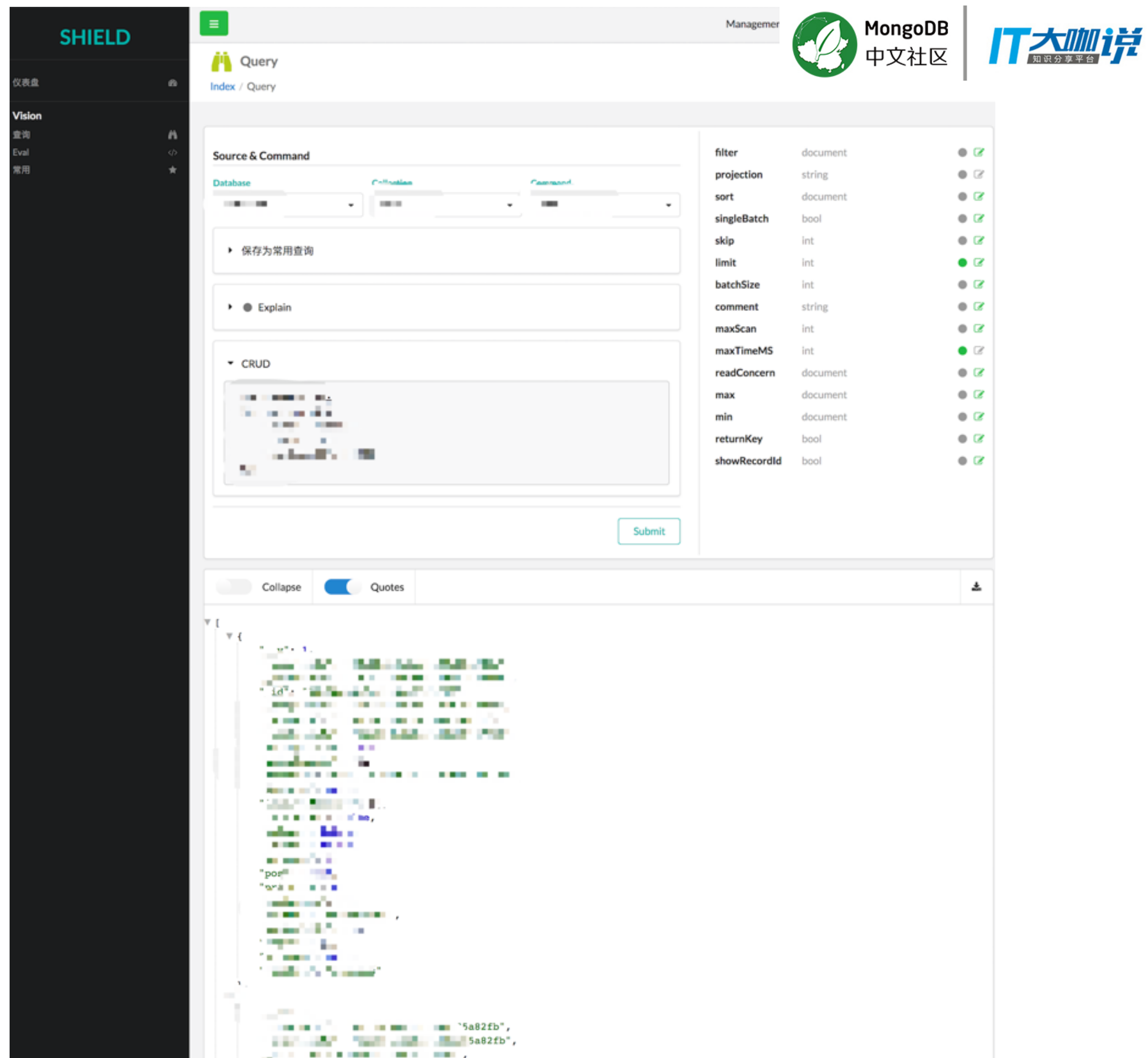
日常工作

- 效率与安全
- 将运维同学从繁琐的日常query中解放
- 让开发同学能够有意识的对自己的query做优化
- 可追溯性、可控与脱敏

[illegible]

Vision

- 账号、角色、赋权
- 脱敏
- 语句审核（非自动）
- Query Explain
- 常用查询
- js脚本
- 管理后台



未来的Vision

希望能做到如inception那么强悍

- 查询语句优化建议（根据explain结果以及现有index，mongoclient语义检测）
- 索引优化建议（根据explain结果以及现有index）
- 全面支持CRUD（预执行库、库的flashback等）
- 更自动与强大的审核（利用mongoclient）
- Flashback?
- 开源



- https://github.com/mongodb/mongo/blob/cc954e9e1d88b30d1ab89ee3bbbd9db0bb15263d/src/mongo/db/query/plan_ranker.cpp#L191
- https://github.com/mongodb/mongo/blob/cc954e9e1d88b30d1ab89ee3bbbd9db0bb15263d/src/mongo/db/query/plan_ranker.cpp#L68

```
// static
double PlanRanker::scoreTree(const PlanStageStats* stats) {
    // We start all scores at 1. Our "no plan selected" score is 0 and we want all plans to
    // be greater than that.
    double baseScore = 1;
    ...
    double productivity =
        static_cast<double>(stats->common.advanced) / static_cast<double>(workUnits);
    ...
    const double epsilon = std::min(1.0 / static_cast<double>(10 * workUnits), 1e-4);
    ...

    double tieBreakers = noFetchBonus + noSortBonus + noIxisectBonus;
    double score = baseScore + productivity + tieBreakers;
    ...
    if (internalQueryForceIntersectionPlans.load()) {
        if (hasStage(STAGE_AND_HASH, stats) || hasStage(STAGE_AND_SORTED, stats)) {
            // The boost should be >2.001 to make absolutely sure the ixisect plan will win due
            // to the combination of 1) productivity, 2) eof bonus, and 3) no ixisect bonus.
            score += 3;
            LOG(5) << "Score boosted to " << score << " due to intersection forcing.";
        }
    }

    return score;
}
```

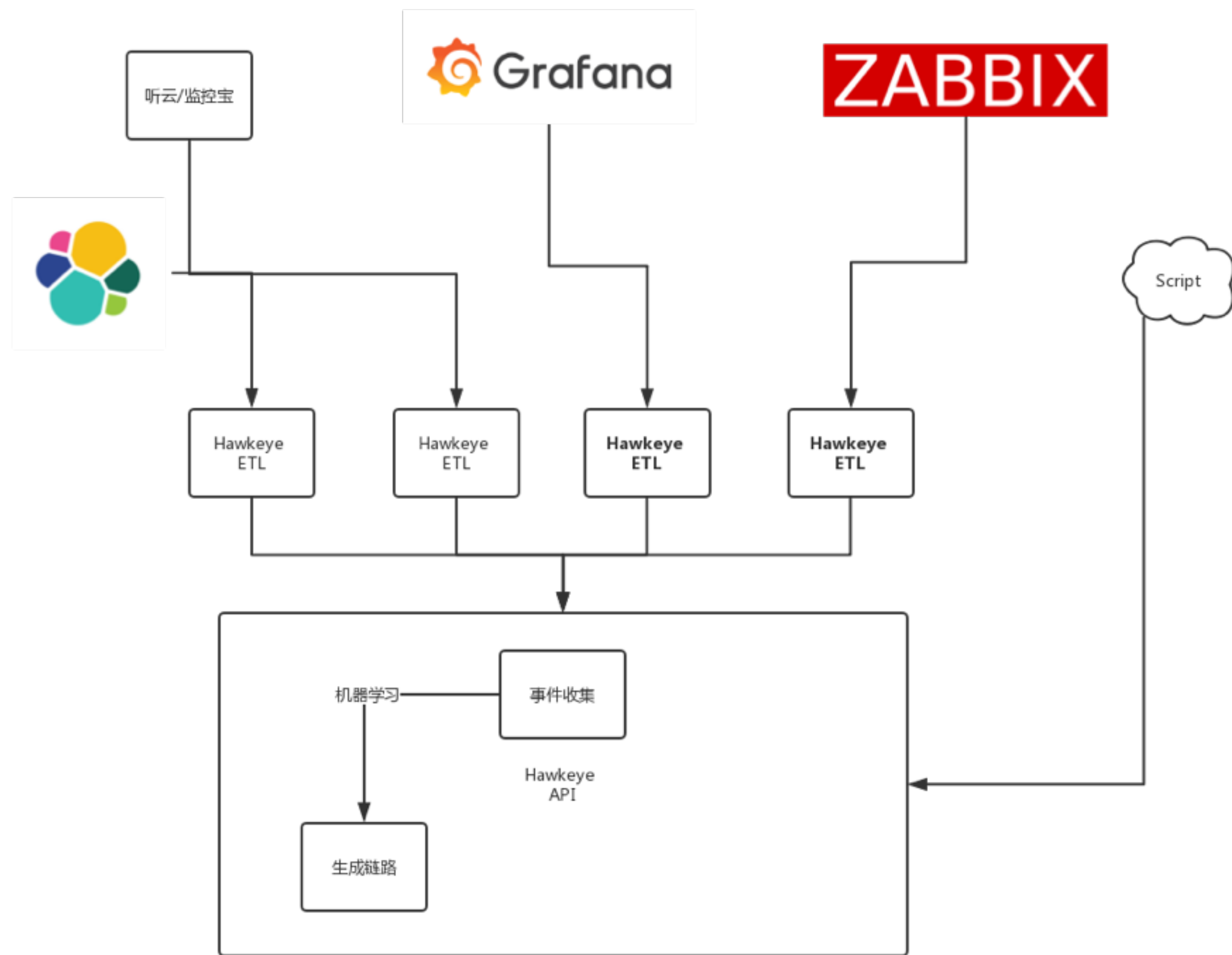
- 默认的plan会有一个baseScore为1
- 获取该plan的works unit以及advanced, 并计算 $productivity = advanced / worksUnits$
- 定义了一个epsilon来防止tie
- 执行计划而言, covered projection 会更优势, 所以有projectstionstage并且有fetch stage的时候 noFetchBounus为0
- 不在内存中sort的plan优先级更高, 所以有sort stage的 noSortBonus为0
- 当出现tie的时候, 单个index比交叉index 优先级更高, 所以使用intersection index的时候 nolxisectBonus为0
- 如果强制指定使用交叉index, 则并且为hash或者sorted stage。则score+3
- isEOF为true的score+1, 但是如果plancache的时候这个isEOF的score要-1避免cache的影响



- works是指query execution阶段的每个工作单元，比如一次index扫描。
- Advanced值可以理解为最终返回的documents的work units
- $works = advanced + iseof$ (每个stage层的) + needTime (每个stage层的)

Hawkeye

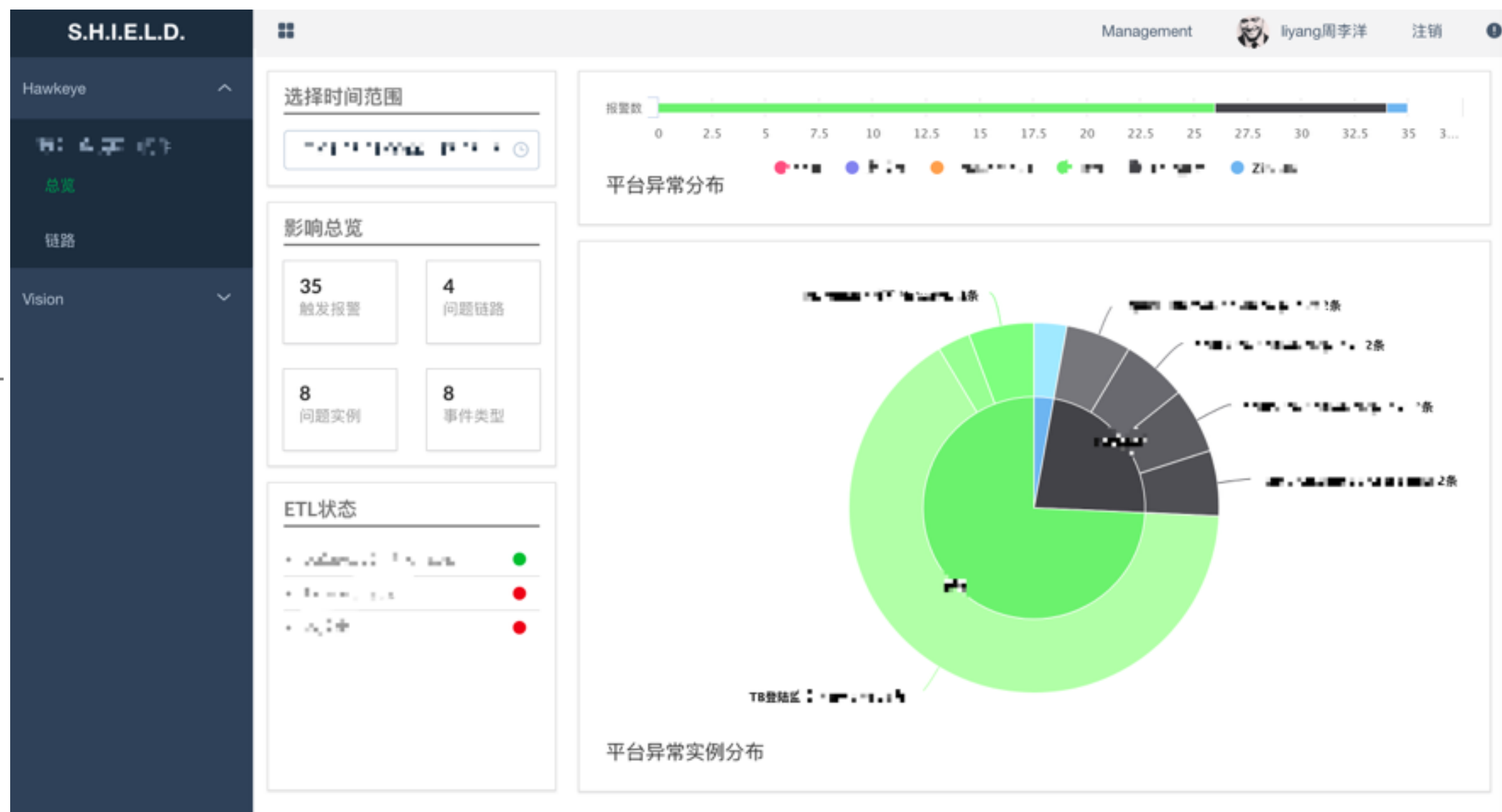
- MMS
- Zabbix
- Influxdb
- 脚本
- Dapper





Hawkeye

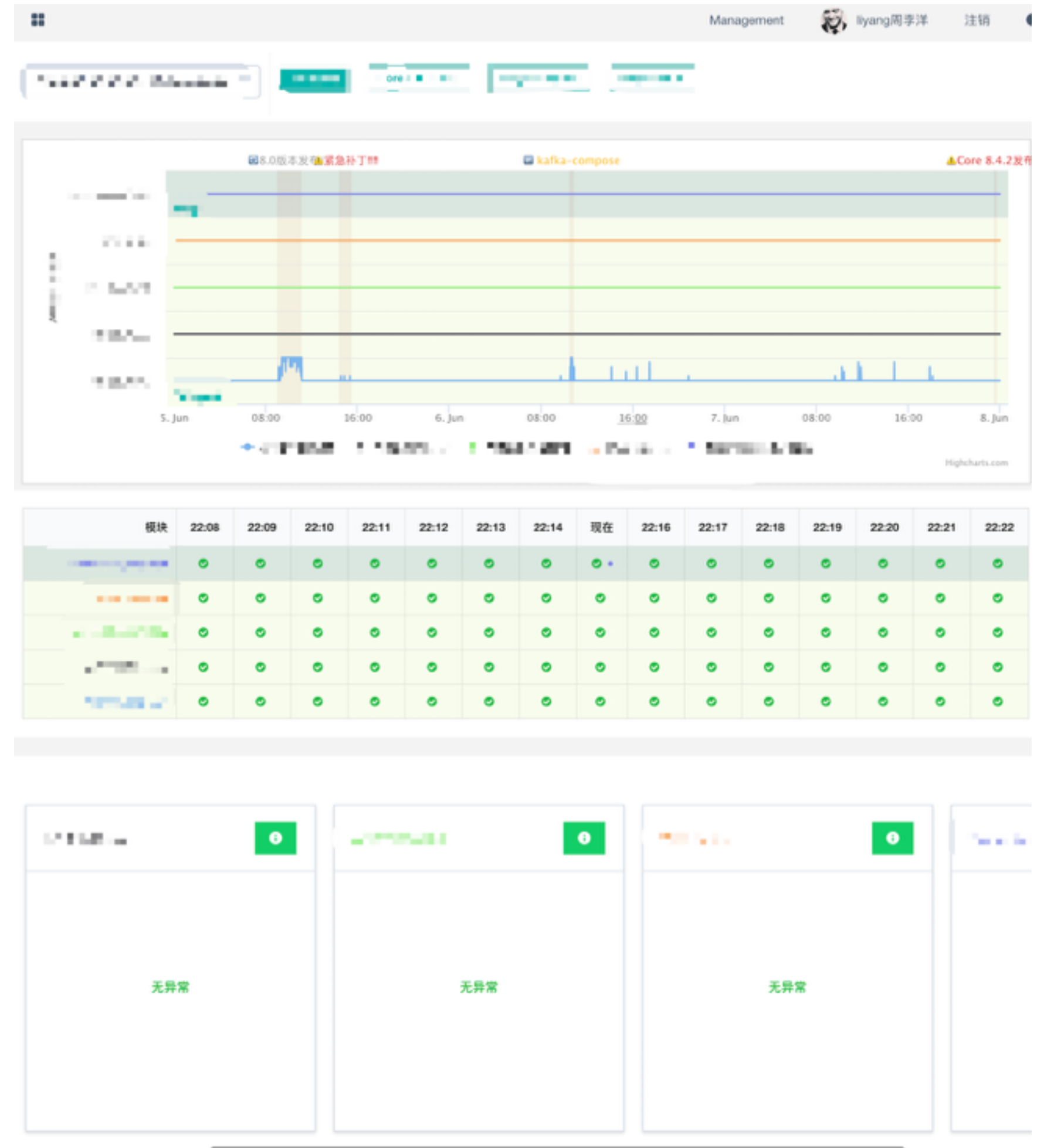
- 监控数据统一展示
- 对应链路计算获取端到端细节
- 加速问题定位





Hawkeye

- 当出现slow query的时候，通过检测程序将相关信息写入Hawkeye
- 此时定义的链路中可能包括了dbquery，主要业务状态，主要业务所在服务的状态，zipkin中程序调用情况，各个事件（如版本发布、等）



未来的Hawkeye

- 非事件的接入（如主动监测数据的接入）
- 通过机器学习将收集到的事件学习成更准确的链路
- 更多的开放API
- 开源



- 12条命名规则
- 5条数据结构设计准则
- 3个持续优化与维护数据结构的工具
- 2个自研运维工具



MongoDB
中文社区

IT大咖说
知识分享平台

问答



MongoDB
中文社区

IT大咖说
知识分享平台

teambition
团队协作工具创导者

www.teambition.com