拉勾教育
—互联网人实战大学—

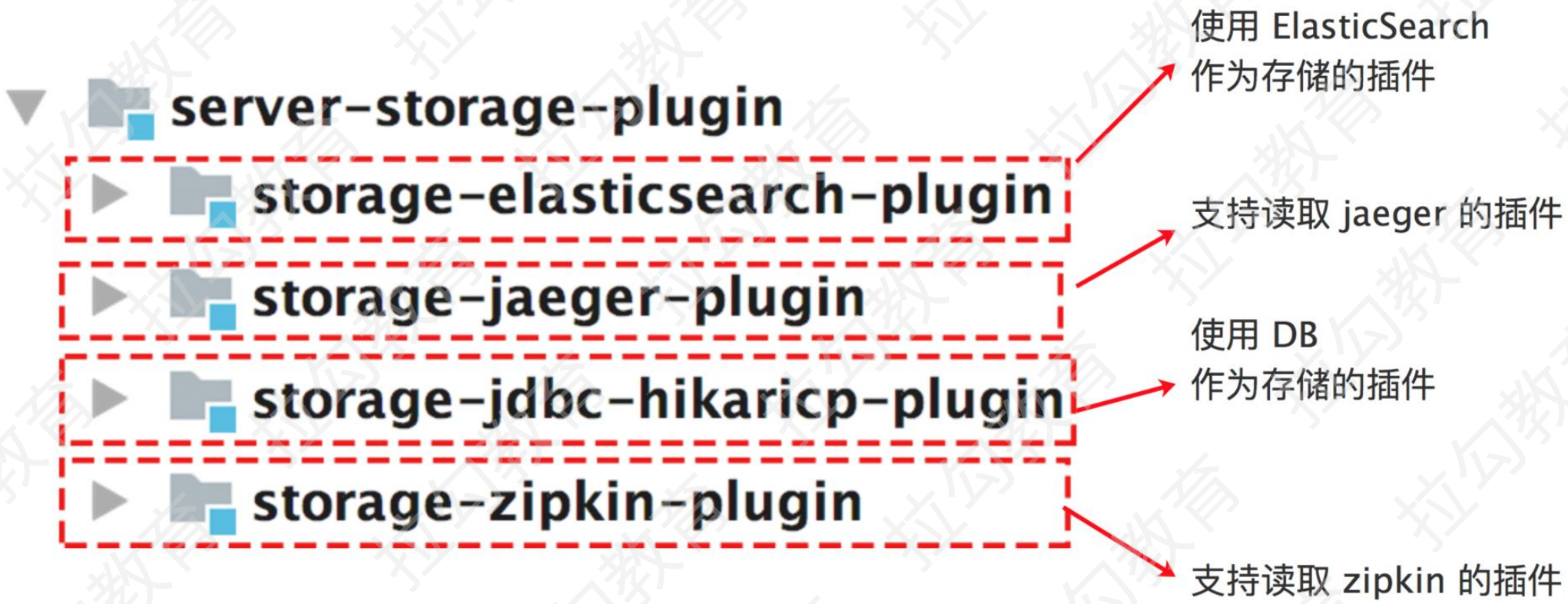
《31 讲带你搞懂 SkyWalking》

徐郡明 资深技术专家

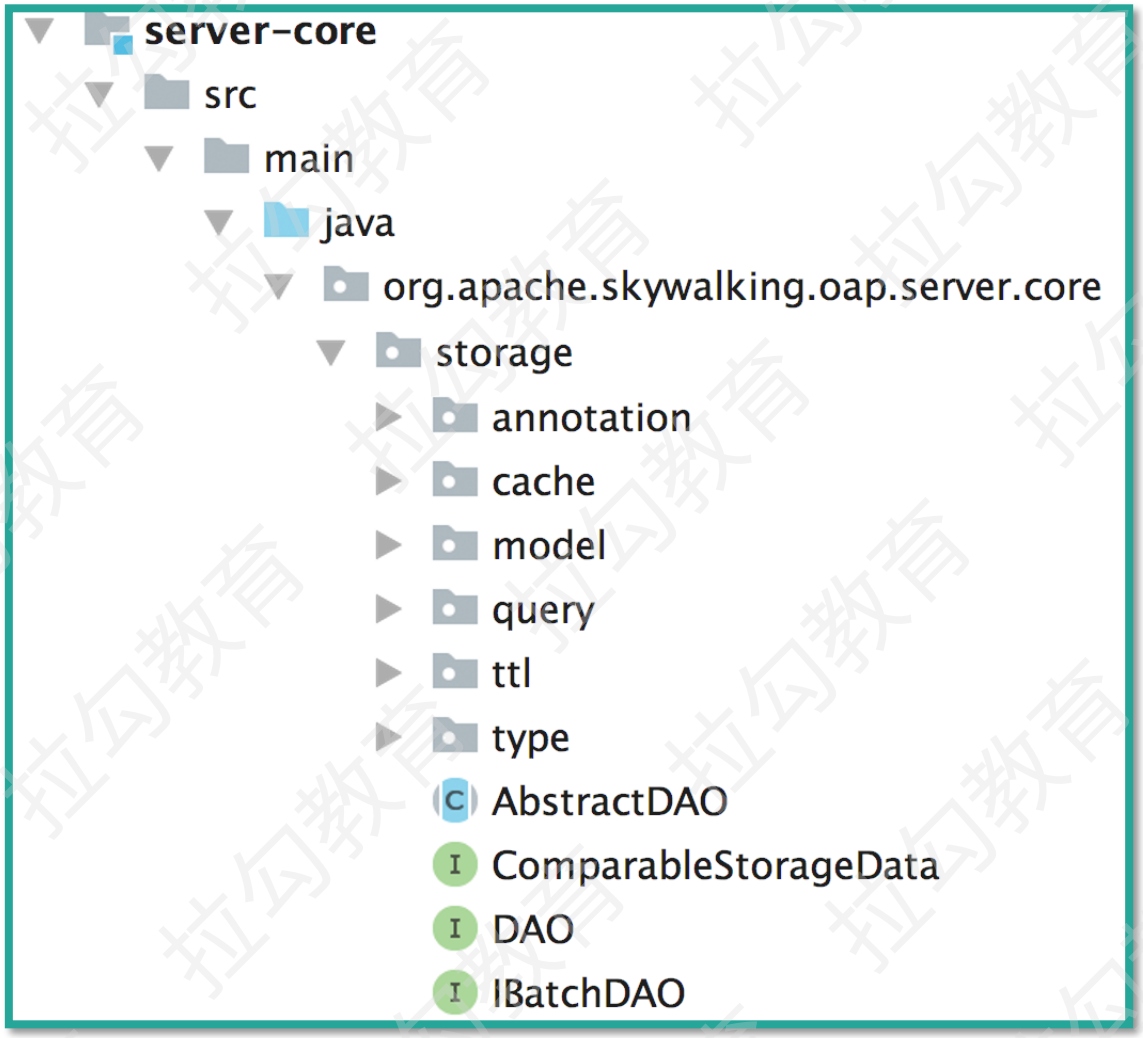
— 拉勾教育出品 —

加餐3：SkyWalking OAP 存储体系剖析

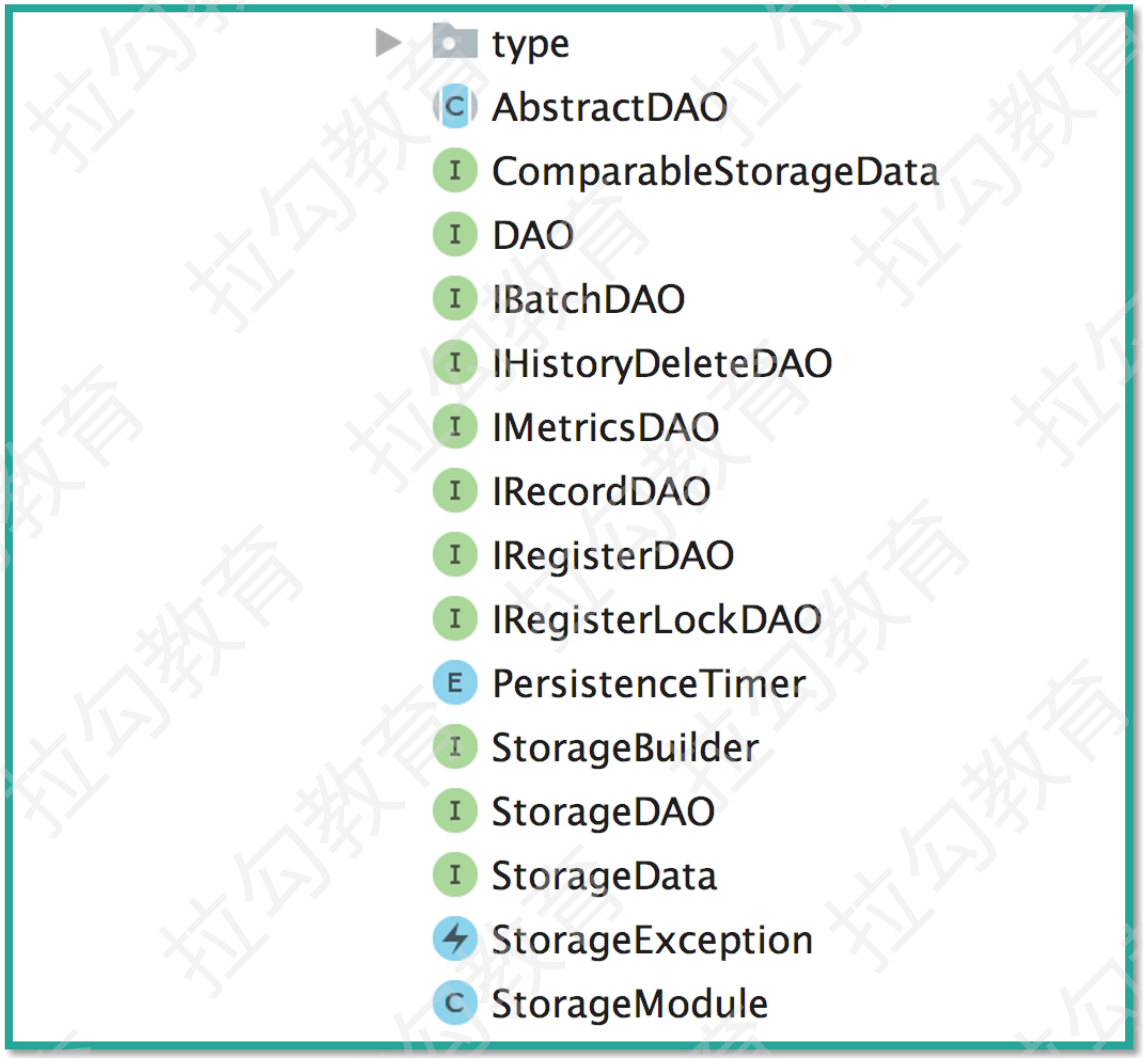
前言



前言



前言



前言

OAP 存储了两种类型的数据：**时间相关的数据**和**非时间相关的数据**（与“时序”这个专有名词区分一下）

注册到 OAP 集群的 Service、ServiceInstance 以及同步的 EndpointName、NetworkAddress

都是非时间相关的数据，一个稳定的服务产生的这些数据是有限的

可以用几个固定的 ES 索引（或数据库表）来存储这些数据

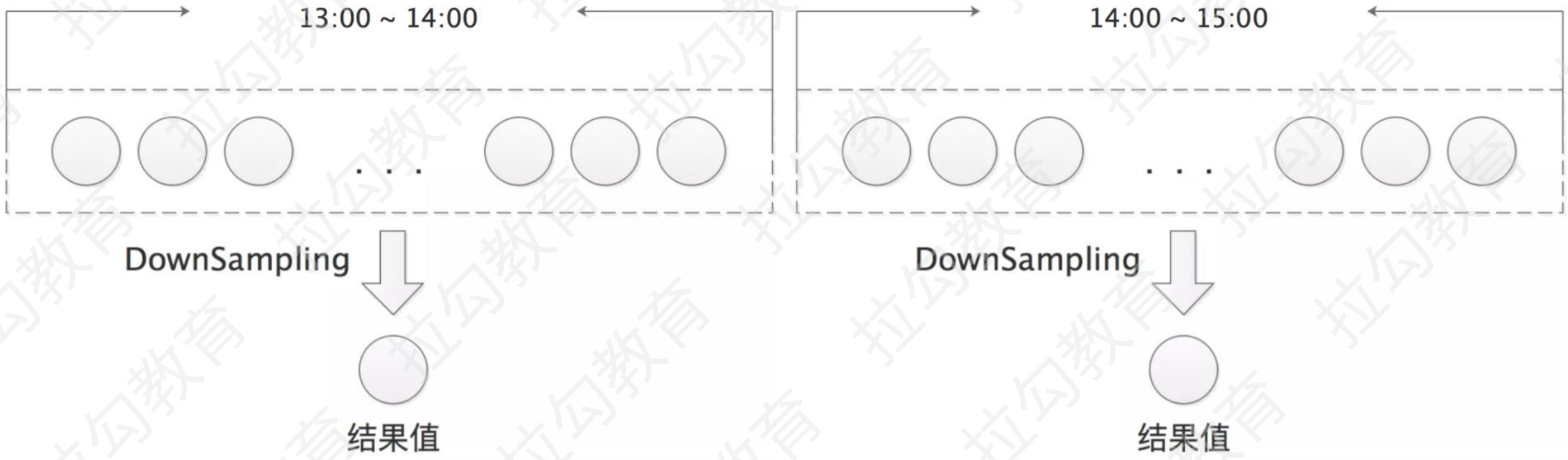
常见的切分方式按照**时间窗口**以及 **DownSampling** 进行切分

前言

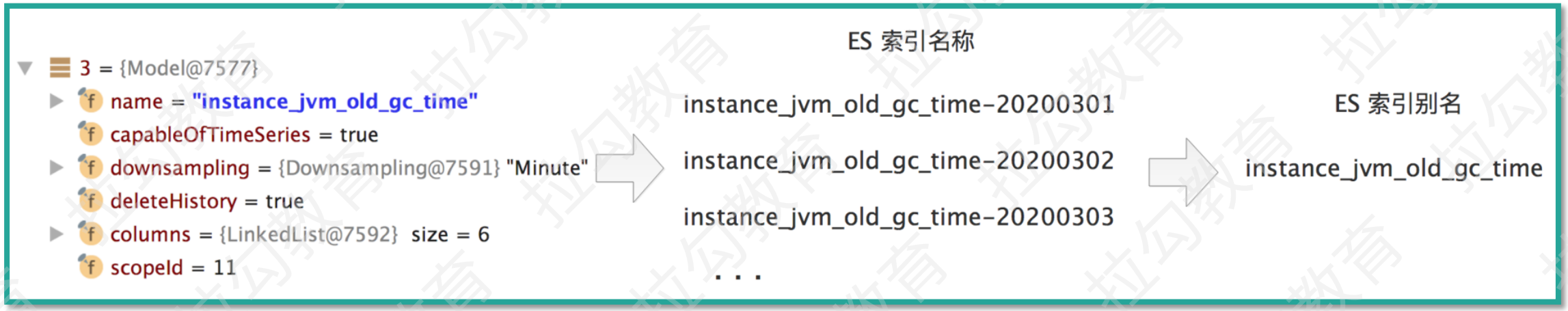
拉勾教育
互联网人实战大学

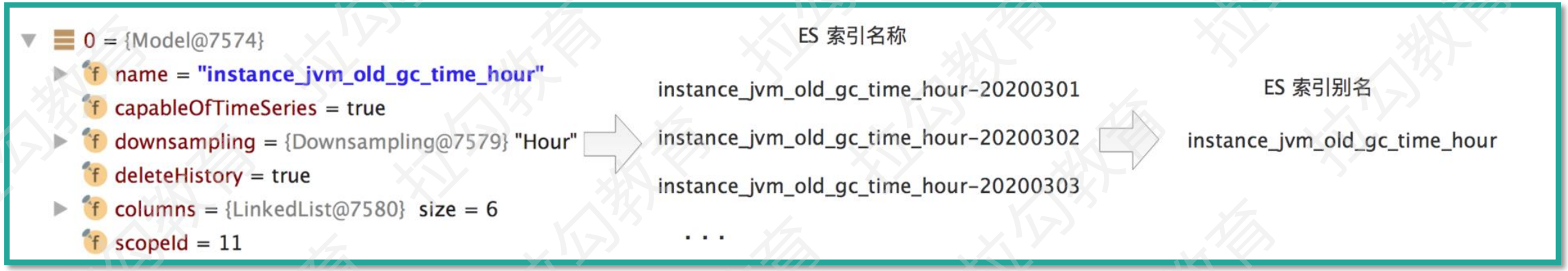
DownSampling（翻译为"向下采样"或是"降采样"）

是降低数据采样率或分辨率的过程

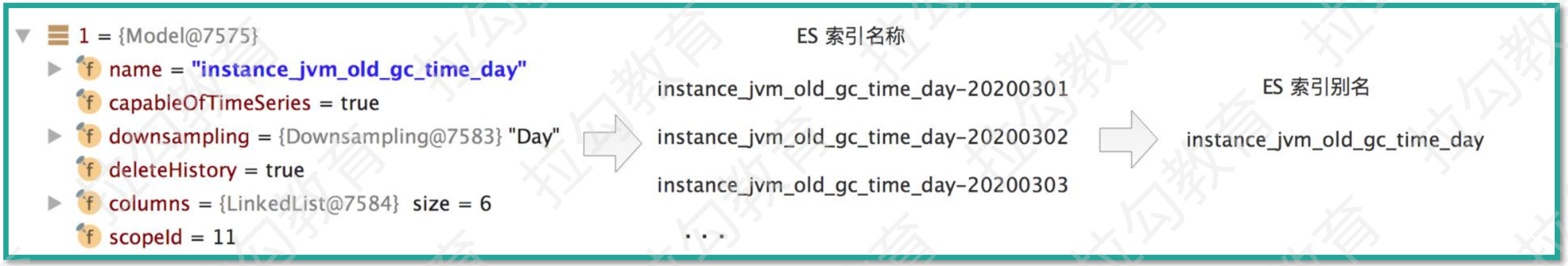


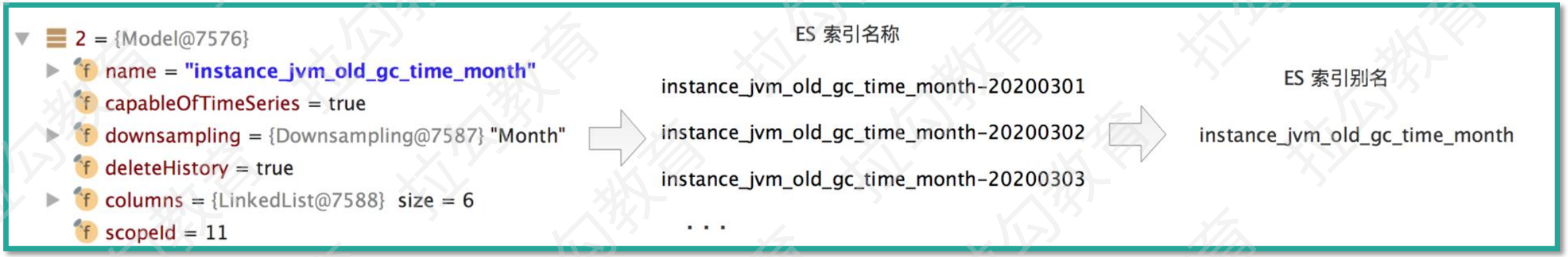
Model 与 ES 索引





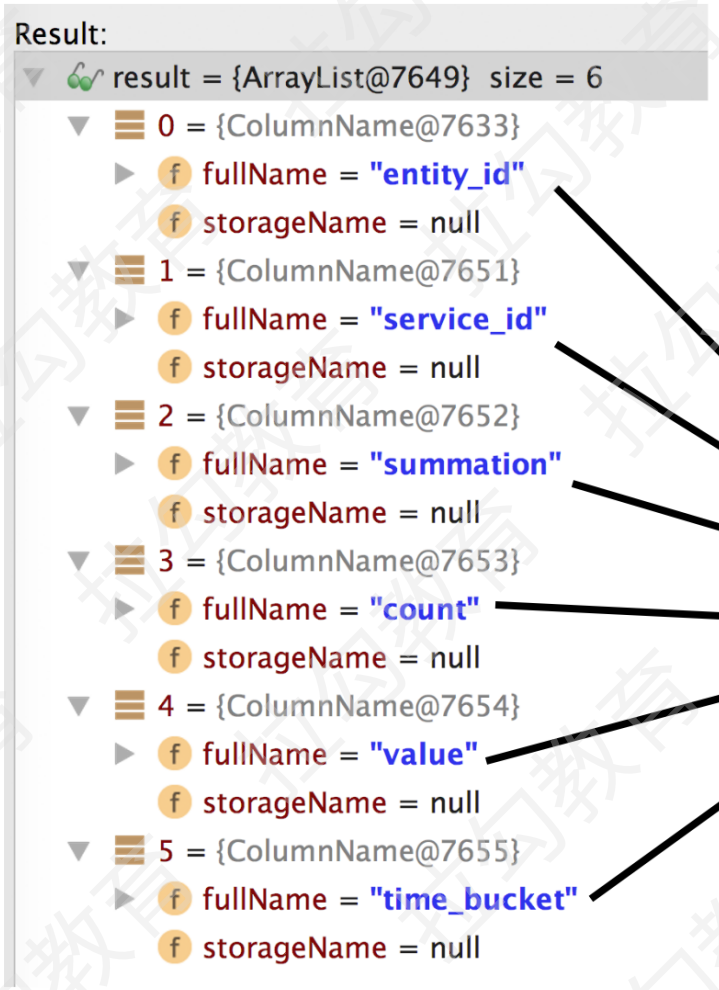
Model 与 ES 索引





Model 与 ES 索引

Model.columns 集合

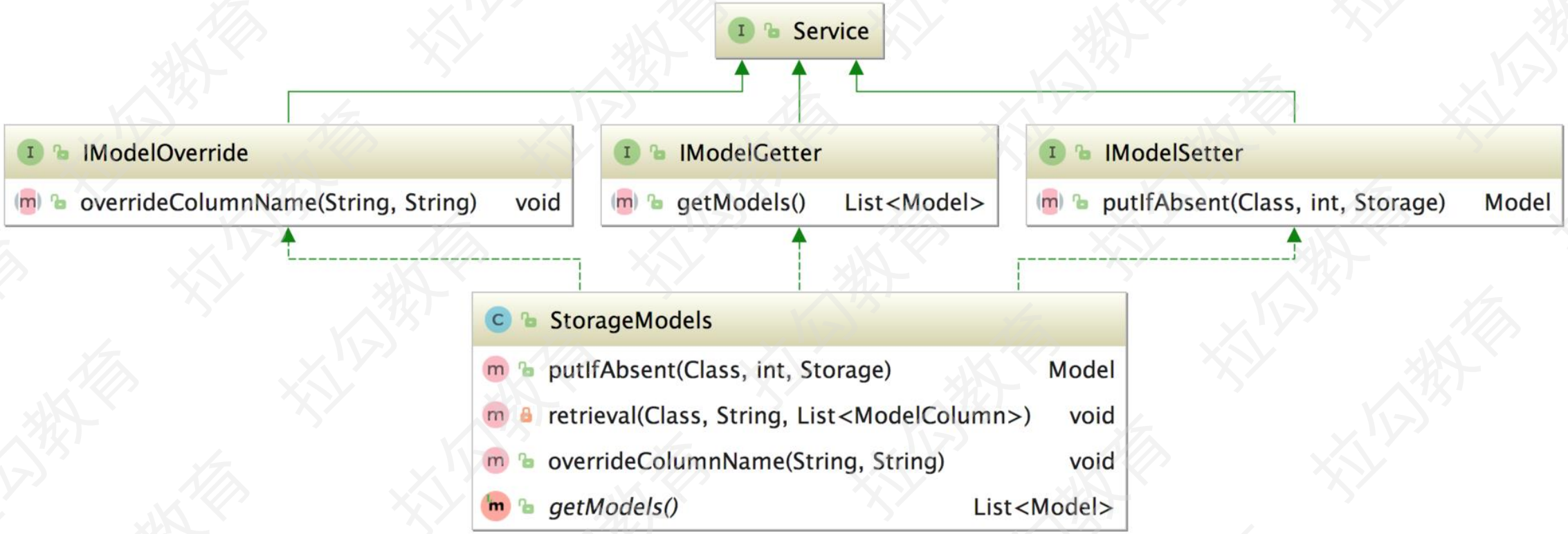


对应 ES 索引中的 Field

Table JSON

t	_id	20200319_93
t	_index	instance_jvm_old_gc_time_day-20200319
#	_score	1
t	_type	type
t	entity_id	93
#	service_id	3
#	summation	97
#	count	685
#	value	0
#	time_bucket	20,200,319

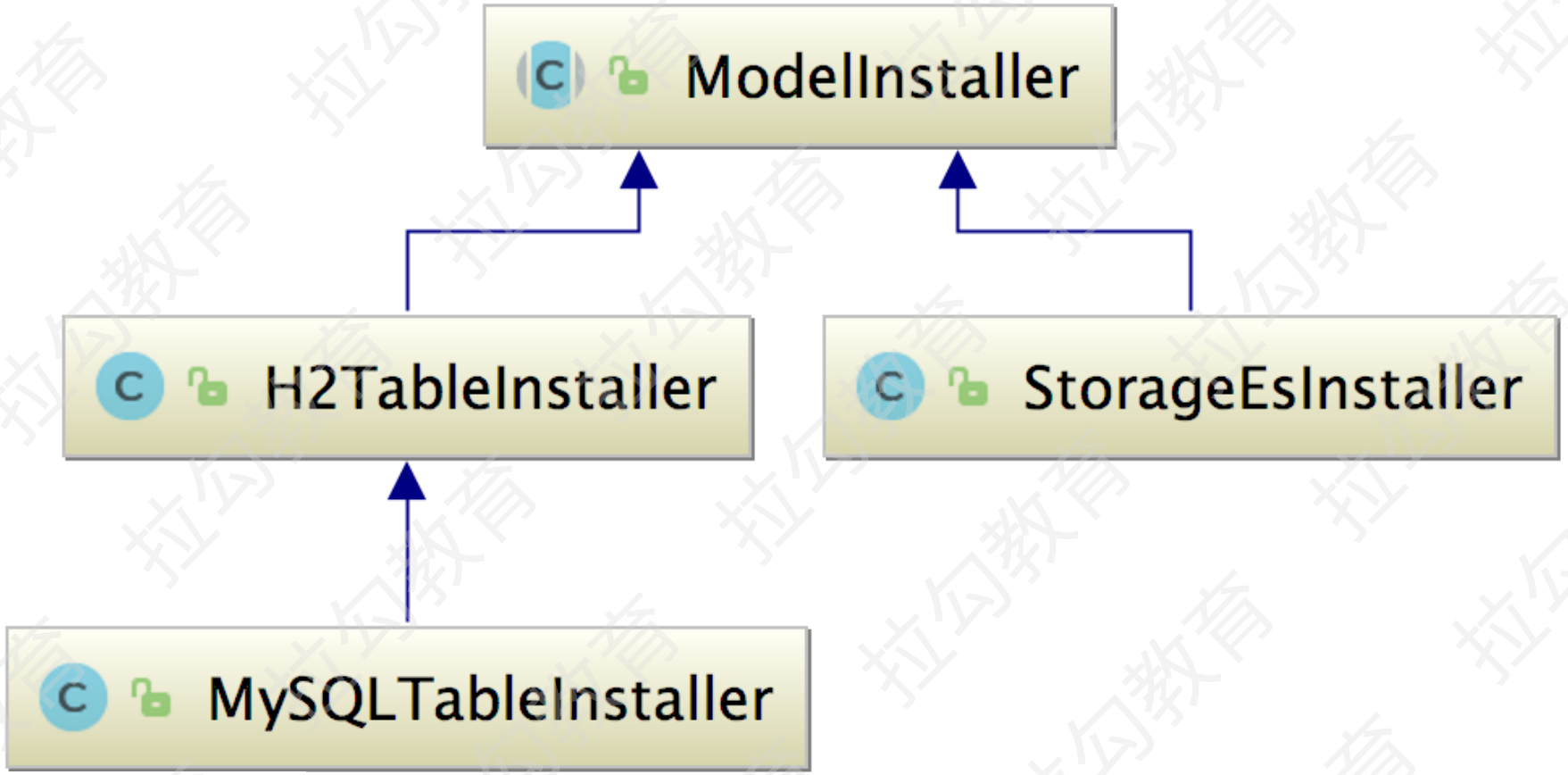
Model 与 ES 索引



初始化存储结构


```
public final void install(Client client) throws StorageException {
    // 获取全部Model对象
    IModelGetter modelGetter = moduleManager.find(CoreModule.NAME)
      .provider().getService(IModelGetter.class);
    List<Model> models = modelGetter.getModels();
    //根据mode这个环境变量决定是否创建底层存储结构
    if (RunningMode.isNoInitMode()) {
      ... // mode环境变量为no-init，则只会输出日志，不会初始化ES索引
    } else { // mode环境变量为非non-init值，走这个分支
      for (Model model : models) {
        //检查Model对应的底层存储结构是否存在，如果不存在则通过
        // createTable()方法进行创建
        if (!isExists(client, model)) {
          createTable(client, model);
        }
      }
    }
}
```

初始化存储结构


ModelInstaller
H2TableInstaller
StorageEsInstaller
MySQLTableInstaller

初始化存储结构


```
public void start() throws ModuleStartException {
    elasticSearchClient.connect();
    StorageEsInstaller installer = new StorageEsInstaller(...);
    installer.install(elasticSearchClient);
}
```

初始化存储结构


```
protected void createTable(Client client, Model model) {
    ElasticSearchClient esClient = (ElasticSearchClient)client;
    //创建settings，其中指定了索引的分片数量、副本数量以及refresh时间间隔
    JsonObject settings = createSetting();
    //创建mapping，其中指定了各个Field的类型等配置
    JsonObject mapping = createMapping(model);
    if (model.isCapableOfTimeSeries()){
      //对于时间相关的索引，先创建Template，其中除了指定settings和mapping
      //之外，还会指定该Template匹配的索引名称(index_patterns)以及
      //别名(aliases)
      if (!esClient.isExistsTemplate(model.getName())){
        boolean isAcknowledged = esClient.createTemplate(
            model.getName(), settings, mapping);
      }
      if (!esClient.isExistsIndex(model.getName())) {
        // ElasticSearch中真正的索引名称会添加时间戳后缀
        String timeSeriesIndexName = TimeSeriesUtils
```

初始化存储结构


```
        boolean isAcknowledged = esClient.createTemplate(
            model.getName(), settings, mapping);
    }

    if (!esClient.isExistsIndex(model.getName())) {
        // ElasticSearch中真正的索引名称会添加时间戳后缀
        String timeSeriesIndexName = TimeSeriesUtils
                .timeSeries(model);
        boolean isAcknowledged = esClient.createIndex(
                timeSeriesIndexName);
    }
  } else {
    //与时间无关的索引只会创建一个索引，直接使用Model.name作为索引名
    //称创建(没有时间戳后缀)，另外，同样会设置settings和mapping
    boolean isAcknowledged = esClient.createIndex(model.getName(),
        settings, mapping);
  }
}
```

初始化存储结构

模板名称	instance_jvm_old_gc_time	instance_jvm_old_gc_time_hour	instance_jvm_old_gc_time_day	instance_jvm_old_gc_time_month
index_patterns	instance_jvm_old_gc_time-*	nstance_jvm_old_gc_time_hour-*	nstance_jvm_old_gc_time_day-*	nstance_jvm_old_gc_time_month-*
匹配索引的示例	instance_jvm_old_gc_time-20200301 instance_jvm_old_gc_time-20200302 …	instance_jvm_old_gc_time_hour-20200301 instance_jvm_old_gc_time_hour-20200302 …	instance_jvm_old_gc_time_day-20200301 instance_jvm_old_gc_time_day-20200302 …	instance_jvm_old_gc_time_month-202002 instance_jvm_old_gc_time_month-202003 …
aliases	instance_jvm_old_gc_time	instance_jvm_old_gc_time_hour	instance_jvm_old_gc_time_day	instance_jvm_old_gc_time_month

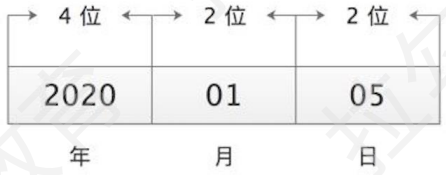
初始化存储结构

DownSampling.Second

DownSampling.Minute

DownSampling.Hour

DownSampling.Day

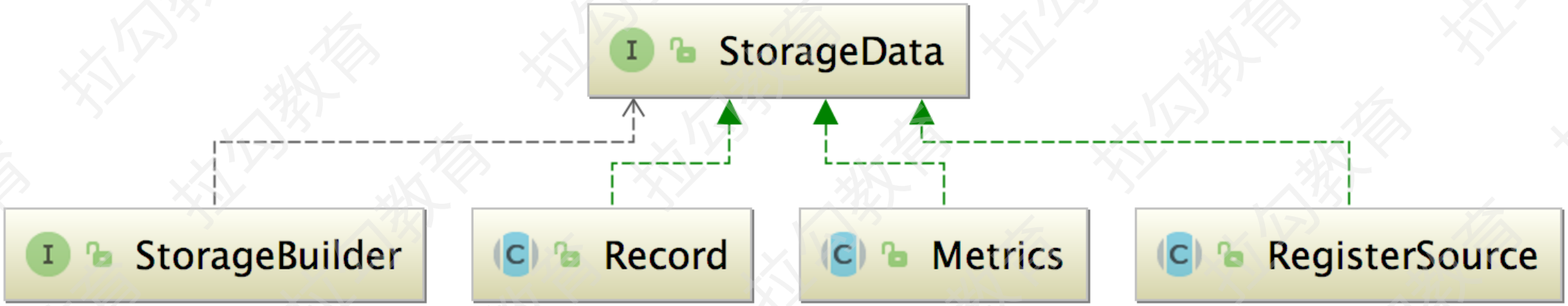


DownSampling.Month

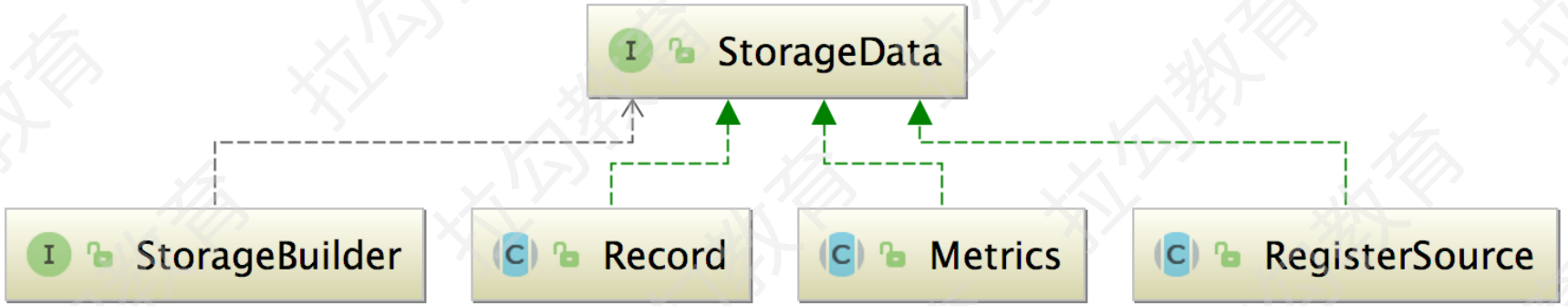
初始化存储结构


```
public static String timeSeries(String modelName,
    long timeBucket, Downsampling downsampling) {
  switch (downsampling) {
    case None:
      return modelName;
    case Hour: //一天内的Hour级别数据存储到同一个索引之中
      return modelName + Const.LINE + timeBucket / 100;
    case Minute: //一天内的分钟级数据存储到一个索引之中
      return modelName + Const.LINE + timeBucket / 10000;
    case Second: //一天内的秒级数据存储到一个索引之中
      return modelName + Const.LINE + timeBucket / 1000000;
    default: //对于Day、Month以及Year不做处理，直接用时间窗口作为后缀
      return modelName + Const.LINE + timeBucket;
  }
}
```

数据抽象



数据抽象

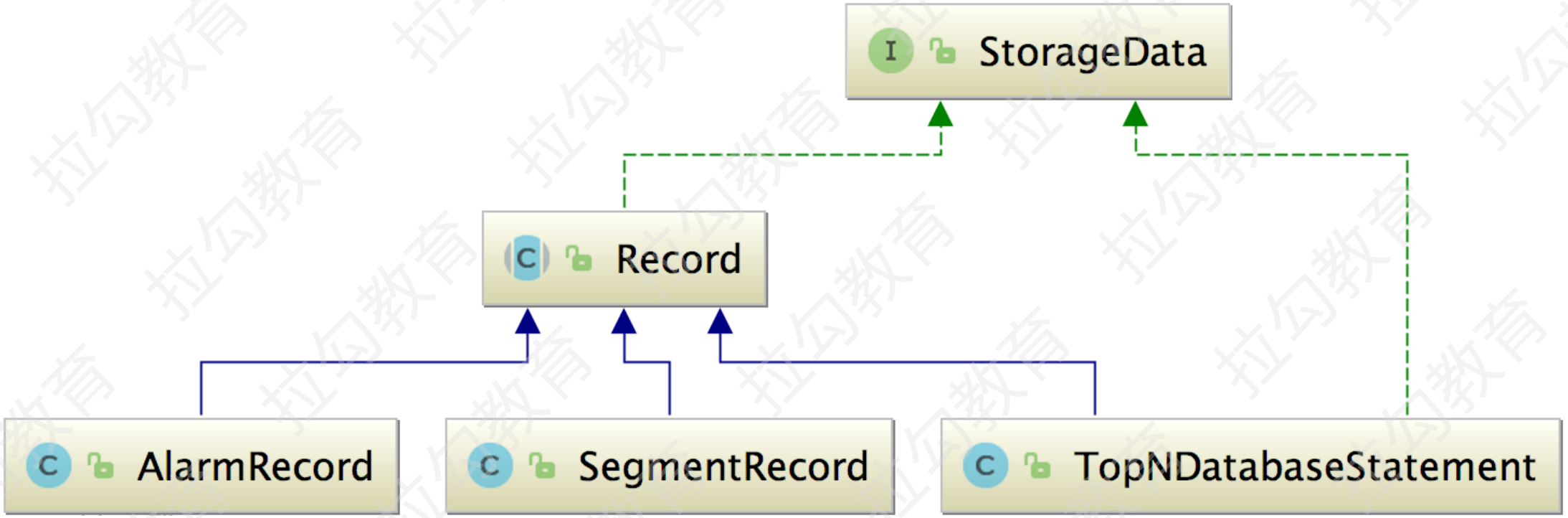


Metrics

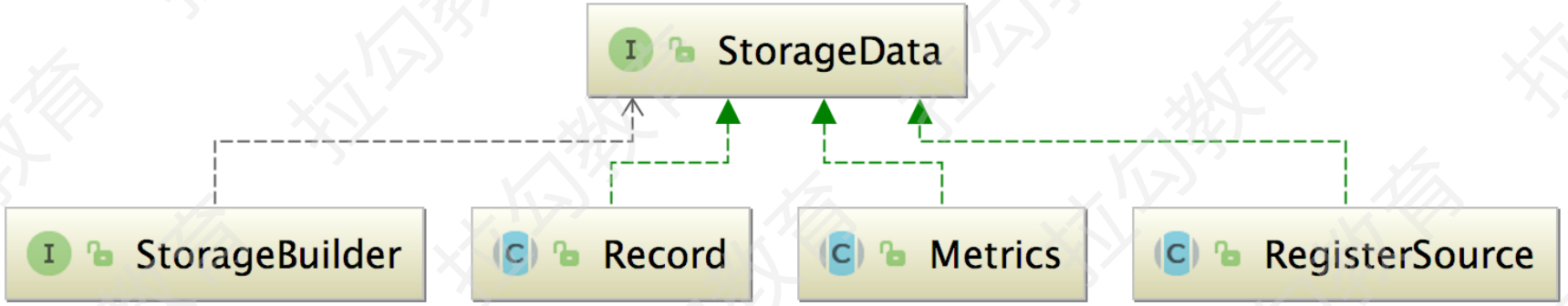
所有监控指标的顶级抽象，其中定义了一个 timeBucket 字段（long类型）

它是所有监控指标的公共字段，用于表示该监控点所处的时间窗口

数据抽象



数据抽象



RegisterSource

抽象了服务注册、服务实例注册、EndpointName（以及 NetworkAddress）同步三个过程中产生的数据

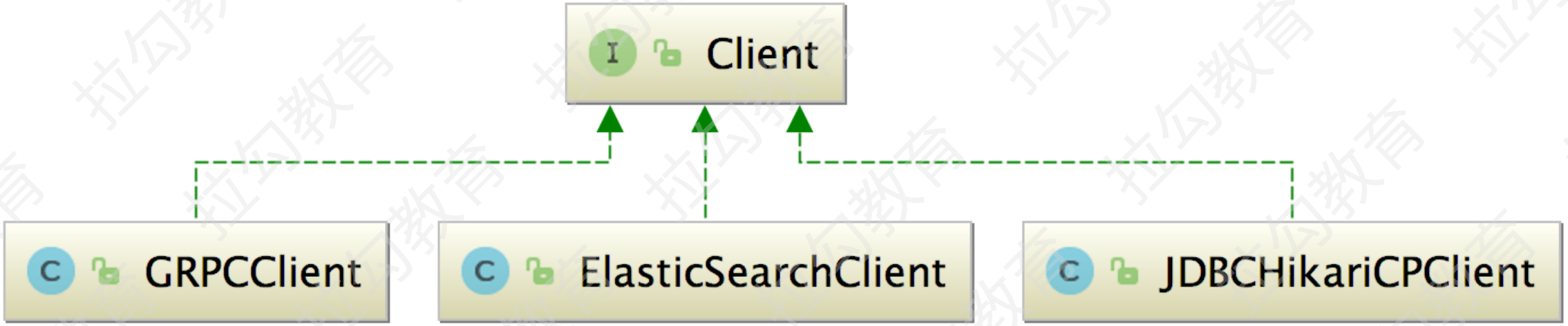
其中定义了三个公共字段，且这三个字段都被 @Column 注解标注

sequence（int 类型）

registerTime（long 类型）

heartbeatTime（long 类型）

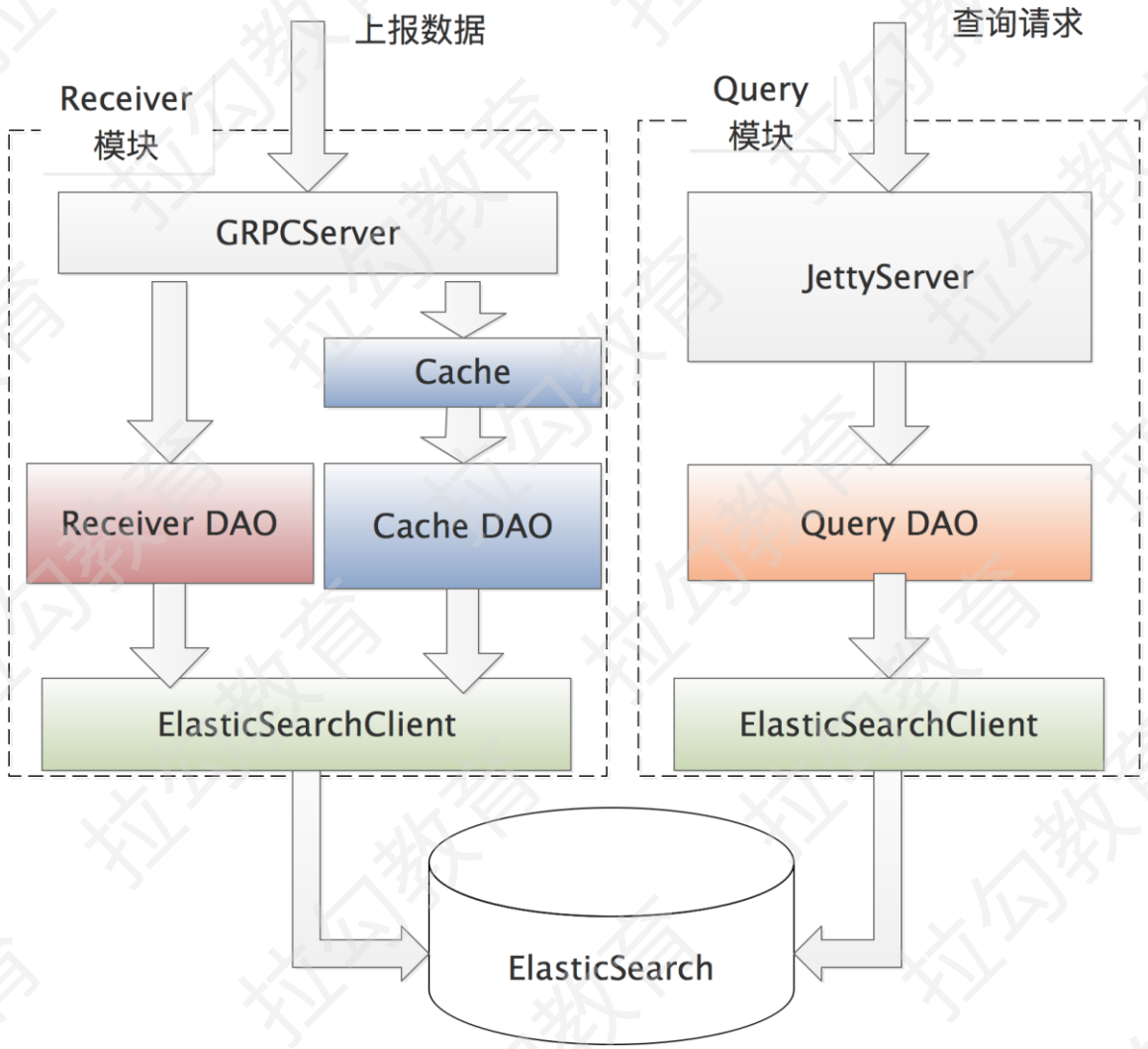
DAO 层架构



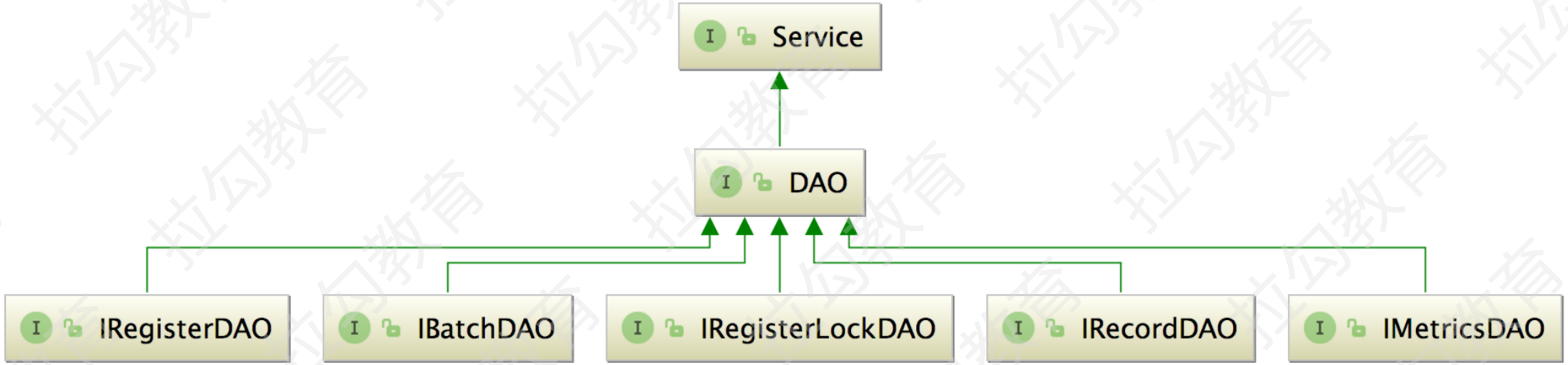
DAO 层架构

DAO 层会完成业务概念与存储概念的转换

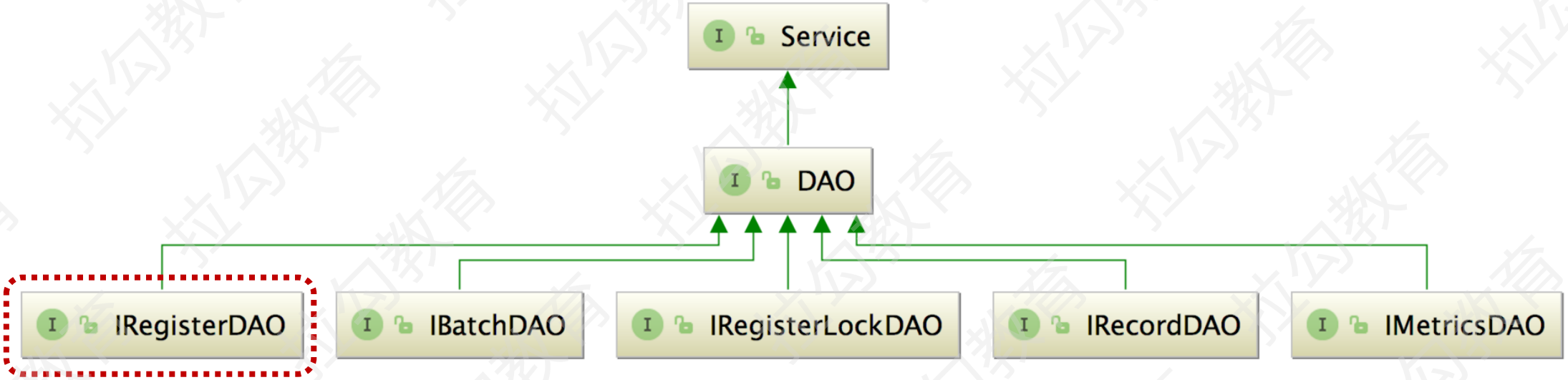
DAO 层架构



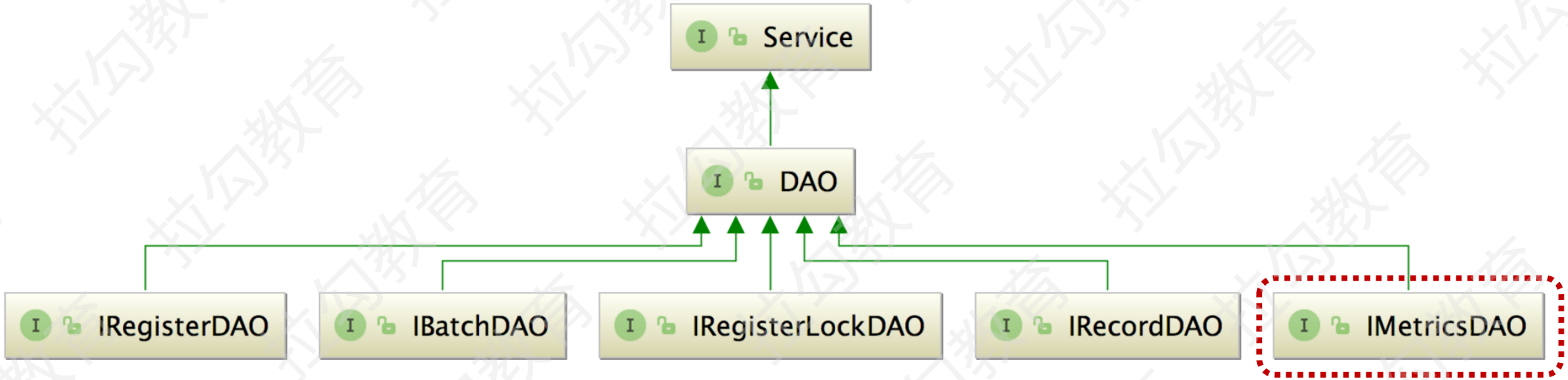
DAO 层架构



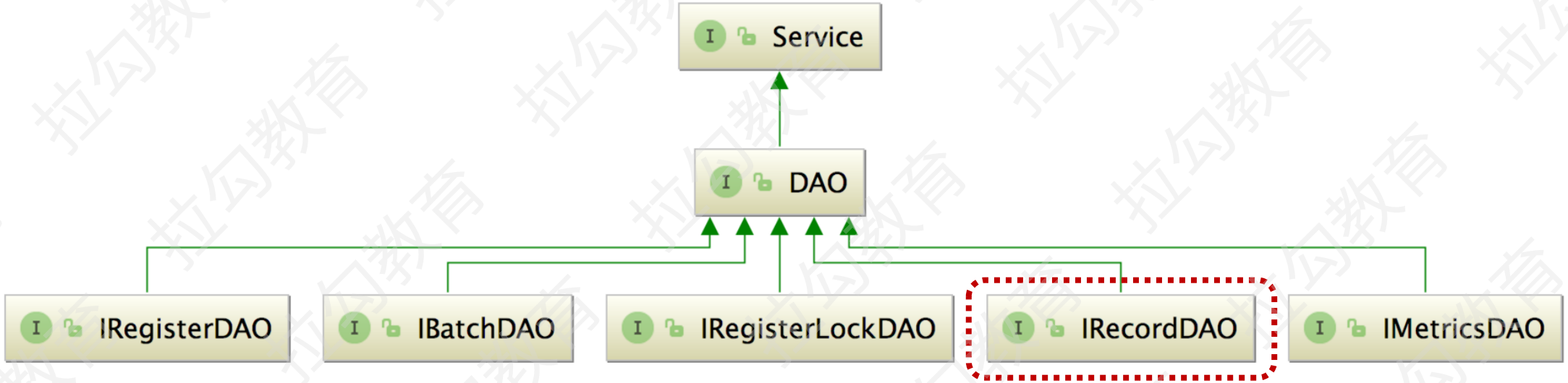
DAO 层架构



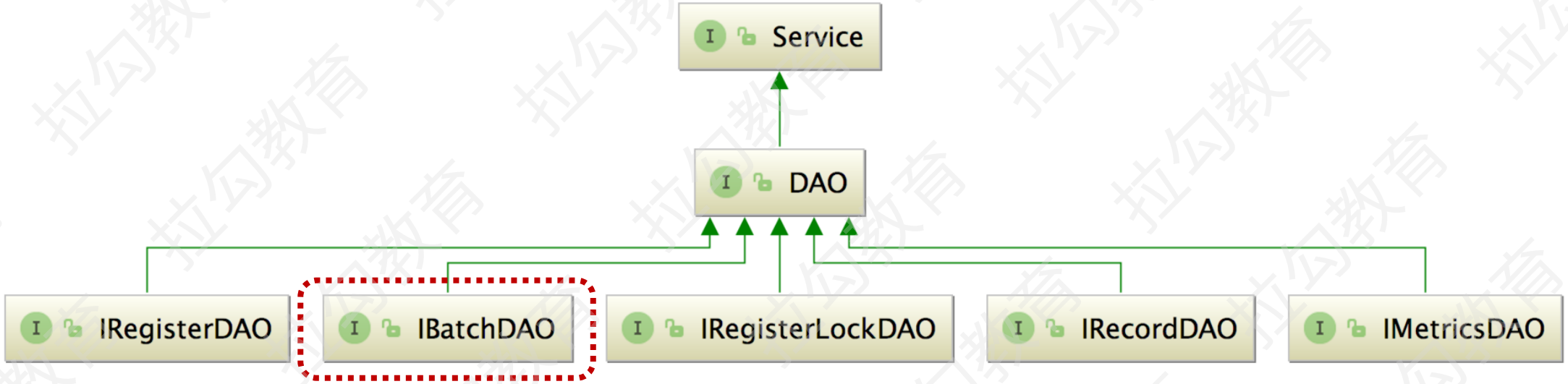
DAO 层架构



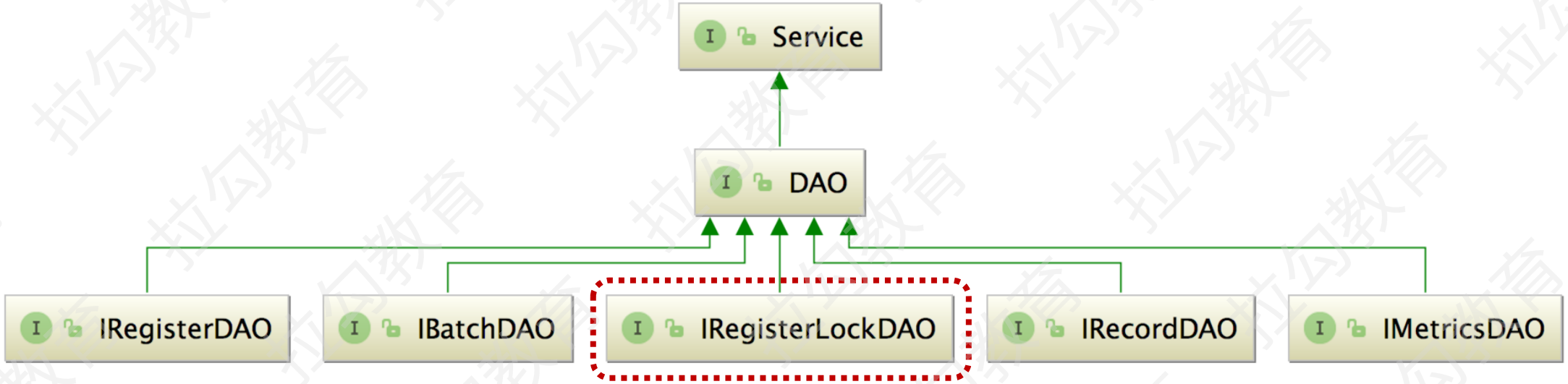
DAO 层架构



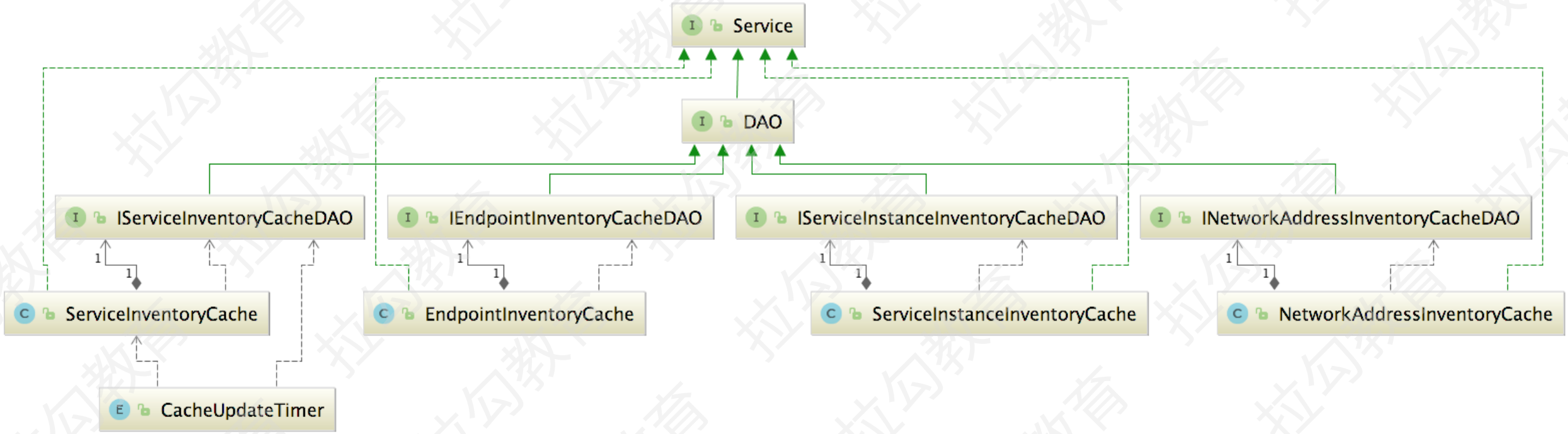
DAO 层架构



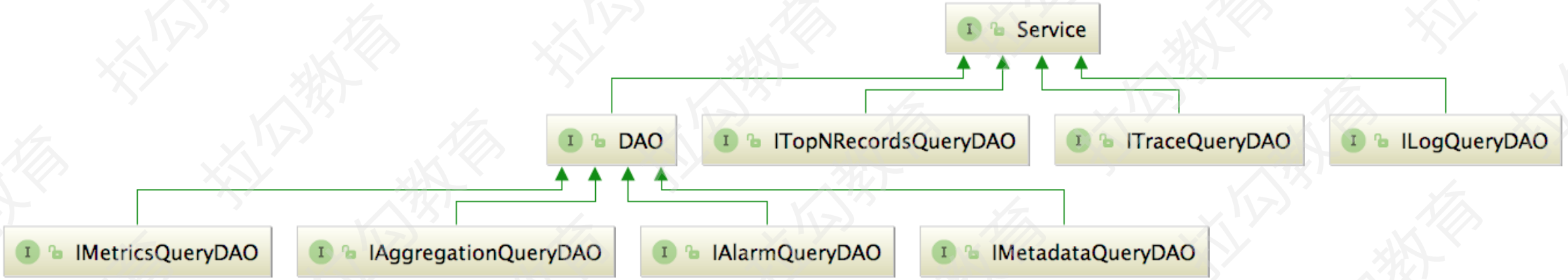
DAO 层架构



DAO 层架构



DAO 层架构



数据 TTL

Metrics、Trace 等（时间相关的数据）对应的 ES 索引都是按照时间进行切分的

随着时间的推移，ES 索引会越来越多

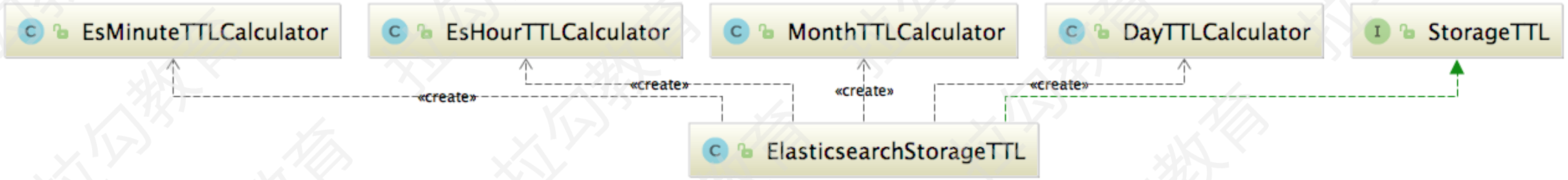
为了解决这个问题，SkyWalking 只会在 ES 中存储一段时间内的数据

CoreModuleProvider 会启动 DataTTLKeeperTimer 定时清理过期数据

数据 TTL

在 DataTTLKeeperTimer 中会启动一个后台线程，每 5 分钟执行一次清理操作，具体执行步骤如下：

1. 通过 ClusterNodesQuery 查询到当前 OAP 集群中全部节点列表，如果当前节点为列表中第一个节点才能继续执行后续的清理操作。这就保证不会出现多个 OAP 节点并发执行清理任务
2. 从 IModelGetter 中拿到全部 Model 对象，根据 Model 数据的 DownSampling 计算每个 Model 保留 ES 索引的范围。这里使用到了 StorageTTL 接口，其中会根据每个 Model 不同的 DownSampling 返回相应的 TTLCalculator，如下图所示：



数据 TTL

删除 ES 索引的过程成还有一些细节：

1. 首先根据模板别名（也就是 Model.name）拿到该 Model 对应的全部 ES 索引
2. 根据 ES 索引的时间后缀，以及 TTLCalculator 计算得到的时间，确定哪些索引会要被删除
3. 如果该 Model 对应的全部索引都要被删除，则会创建一个新索引，为后面写入数据做准备
4. 循环调用 ElasticSearchClient.delete() 方法删除索引

总结

- 介绍 Model 与底层 ES 索引之间的映射关系
- 介绍 ModelInstaller 是如何在 OAP 启动时初始化 ES 索引的
- 介绍 OAP 对不同持久化数据的抽象，以及对应的 DAO 层设计
- 介绍定期删除过期数据的核心原理

Next：第22讲《深入剖析 register-receiver-plugin 插件（上）》

拉勾教育

—互联网人实战大学—

关注拉勾「教育公众号」
获取更多课程信息