元数据管理软件框架设计方案

**摘 要**：高能同步辐射光源（HEPS）是我国“十三五”期间优先建设的、为国家的重大战略需求和前沿基础科学研究提供技术支撑平台的国家重大科技基础设施，服务于超高空间分辨、时间分辨、能量分辨的高通量同步辐射实验。其一期建设的十五条光束线实验站，预计平均每天平均产生200TB的原始实验数据，峰值可达每天500TB。这些实验数据需要得到存储、共享。元数据管理软件框架对实验元数据进行组织管理，实现科学数据高效的组织和利用，实现科学数据生命周期的跟踪与管理，提高前沿交叉学科基础研究、基础应用研究成果产出效率。

**关键词**： 高能同步辐射光源；原始实验数据；实验元数据；科学数据生命周期

# 引言

高能同步辐射光源（High Energy Physics Photon Source, HEPS）是我国“十三五”期间优先建设的、为国家的重大战略需求和前沿基础科学研究提供技术支撑平台的国家重大科技基础设施，位于怀柔科学城北部核心区域。建成之后，HEPS将是我国第一台高能量同步辐射光源，也将是世界上亮度最高的第四代同步辐射光源。

HEPS建成之后将是一个开放的大型科学实验平台，年均面向全球用户提供实验机时将在5000小时以上，开展X射线衍射、散射、成像和谱学等同步辐射实验。作为第四代同步辐射装置，HEPS产生的X射线能够或的毫电子伏的能量分辨、纳米的空间分辨率和飞秒的时间分辨[高能同步辐射光源初步设计报告]，相对于现有光源有数量级的提升。同时，随着光学、电子学技术的发展，以及先进的光学仪器、探测器的使用，用户实验过程中的实验数据将呈现爆发性增加[Siniša Veseli, Nicholas Schwarz and Collin Schmitz, *J. Synchrotron. Rad.* **25** 1574-1580 (2018)]，海量的实验原始数据、实验元数据需要高效、安全地进行采集、传输、存储、分析和共享，以满足装置、光束线实验站、用户等各方面的需求，促进依托HEPS的科研实验产出。

HEPS作为我国面向多学科综合研究平台类重大科技基础设施，其运行和科学数据处理处理效率以及用户体验至关重要。HEPS科学数据处理平台作为设施的重要支撑，需要提供依托元数据管理软件框架对实验数据进行组织和管理。

# 需求分析

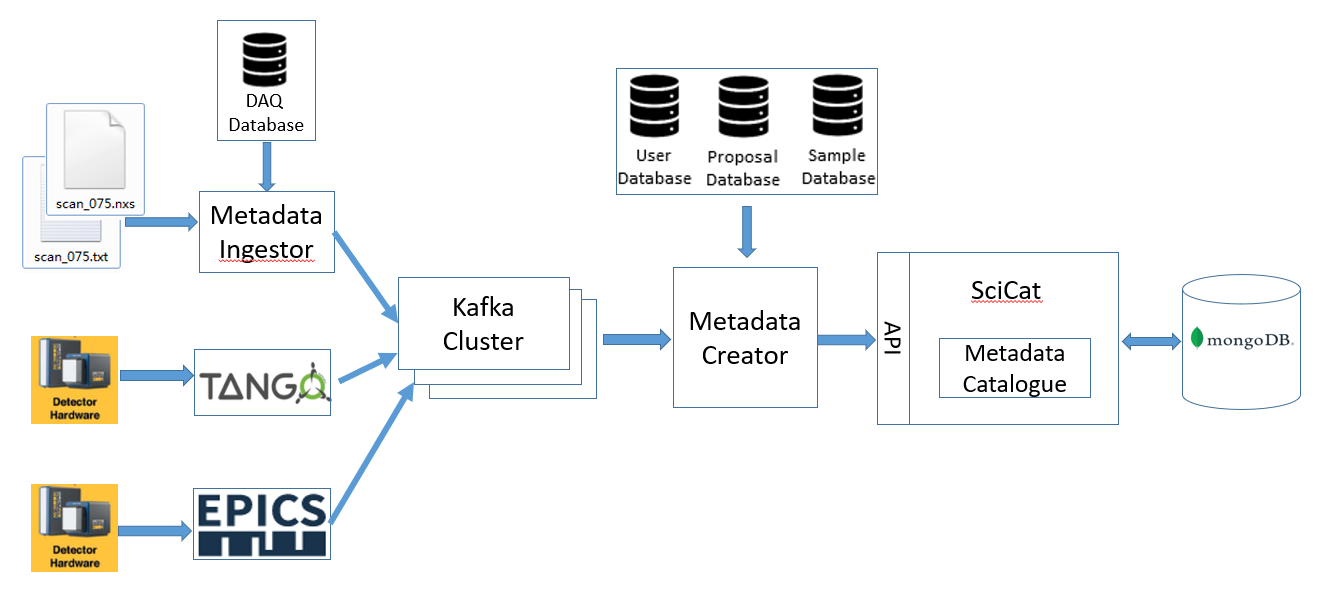
HEPS一期工程包括B1-BE共14个线站和1个测试线站，预计平均每天平均产生200TB的原始实验数据。线站产生的科学实验数据均需要得到及时、快速的处理、分析、存储和共享，同时需要提供实验数据的实时分析和快速反馈，以便为实验站用户提供决策以指导和修正实验过程。如下图1所示，实验数据通过探

图1：HEPS科学实验过程数据处理流程图

测器被收集并存储到DAQ Database、HDFS文件块、磁带等存储介质中，Metadata Creator（元数据提取器）从用户服务系统获取提案、用户、样本相关信息，从实验参数文件获取部分关键实验元数据，并存储到元数据目录数据库MongoDB，用于实验数据的查找、搜索和共享。实验数据的组织和管理需要依赖于实验元数据，将实验元数据存储到元数据目录数据库，能够实现实验数据生命周期的跟踪与管理。

从实验用户角度来说，科学数据是设施实验的核心，贯穿于整个科研活动过程中，在实验过程的不同阶段中会产生相应的元数据和原始数据。实验过程中，用户需要实时查看实验产生的数据，对数据进行快速分拣和提炼，对实验过程中数据质量检查及快速分析；在实验完成后，用户希望根据不同的参数进行快速查找数据，查看数据，下载数据，发布数据。

从IT角度出发，整个科学数据管理流程包括：数据获取、数据格式标准化、数据存储，数据目录索引（catalogue），数据服务。首先从线站获取原始数据和元数据，对数据进行封装，并转化成标准格式HDF5文件，将原始数据和标准化后数据文件传输到数据中心存储。元数据管理软件框架负责存储元数据提取器收集到的实验元数据（如：用户系统的用户提案数据、DAQ系统的实验样本数据、HDF5文件系统的文件存储信息数据）。

# 2目标和任务

需要实现对HEPS实验产生的所有科学数据的在数据获取、传输、存储、分析和数据成果发布各个阶段进行全生命周期的管理。需要实现的目标包括：制定科学数据管理标准与规范；研究和设计科学元数据目录管理架构，实现对科学数据全生命周期的管理，保证科学数据的完整性和可追溯性；实现从实验不同阶段从控制系统、用户服务系统通、数据分析系统获取数据和元数据；提供标准接口，满足其他各系统之间的协作与通讯；提供高效便捷的用户数据服务，实现数据管理制度和规范下对数据的可查看、可下载、可共享和可利用。

# 3 总体架构

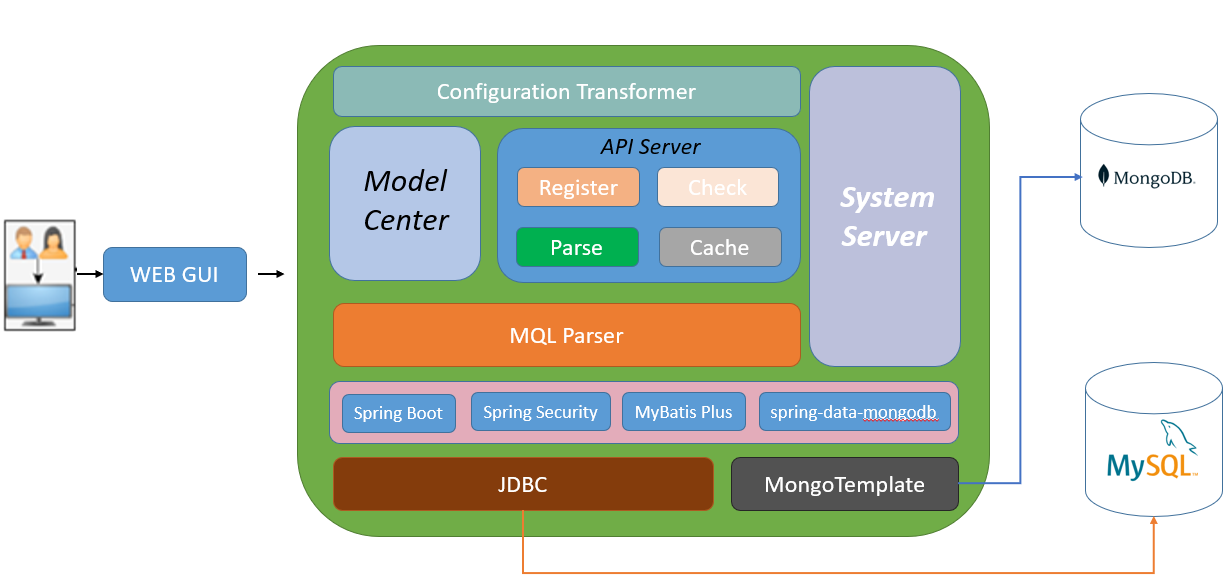
元数据管理软件框架架构（图2）采用了弹性框架，具有可扩展性，不受数据类型和数据量变化的限制；使用json配置文件定义元数据模型，并且可由元数据模型直接生成RESTful API，供其他系统调用；后端采用关系型数据库Mysql和非关系型文档数据库MongoDB结合方式，Mysql负责存储结构化数据，包括系统级别的运行数据、用户登录数据、数据模型定义数据、动态接口对象数据等；MongoDB负责存储业务数据（即实验元数据），它支持高并发元数据读写，同时数据结构灵活，能符合各线站元数据不一致的需求；同时该架构集成基于web的可视化界面，支持元数据检索功能，简化后期web界面开发过程。

图2：元数据管理软件框架架构图

1. Configuration Transformer：配置转换器，负责将WEB GUI交互结果的json配置文件转换为Java实体对象。
2. Model Center：模型中心，负责WEB GUI用户数据模型的定义过程。包括模型的合法校验、字段项及字段树提取、模型关系提取、创建模型、修改模型、删除模型、字段库检索、字段树检索、模型库检索。
3. API Server：动态接口服务，为WEB GUI用户提供可靠的动态接口配置服务，目前仅支持post请求方式的接口。用户在WEB界面为需要传输的数据模型创建动态接口，定义动态接口的过程主要包括接口注册、接口解析、接口校验、接口缓存。
4. System Server：系统服务中心，为整个系统的运行提供可靠的数据服务。系统级别的业务需要依赖System Server，如用户管理、权限管理、登录管理、菜单管理、日志、异常处理、模型管理、接口管理等，这些系统业务所产生的数据存储在关系型数据库Mysql。
5. MQL Parser：MongoDB语句解析器，负责将接口对象解析为可执行的MongoExecutor（MongoTemplate组件能够将其转换为MongoDB语句）。接口被调用时，MQL Parser在解析API对象过程中，会结合API对象的约束条件（如查询条件、分页、排序），将请求体body中的数据作为参数封装为MongoExecutor对象

# 4 详细设计方案

## 4.1 元数据模型管理

元数据模型管理通过微服务的架构对科学数据进行全生命周期的管理。元数据管理软件框架使用关系型数据库Mysql记录元数据模型的结构信息，非关系型数据库MongoDB记录元数据。

### 4.1.1 元数据

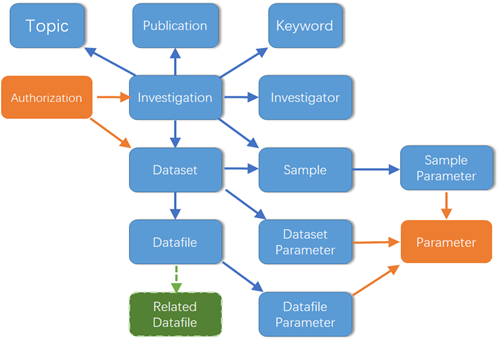
元数据是科学数据管理的基础，在光源类设施的现实应用背景下，它包含了数据在连接科研活动方方面面的信息：从提案到数据发布，贯穿整个实验和数据分析过程。图3从逻辑上展示了科学元数据结构。

图3 科学元数据结构示意图

从上图可以看出，科学元数据的结构围绕着“科学研究”而展开，一项“科学研究”关联着实验主题、实验数据集、实验样本、实验者和关键词、数据发布等信息；实验数据集关联样本信息、数据文件和数据集参数；数据文件关联数据文件参数和其他相关数据文件。

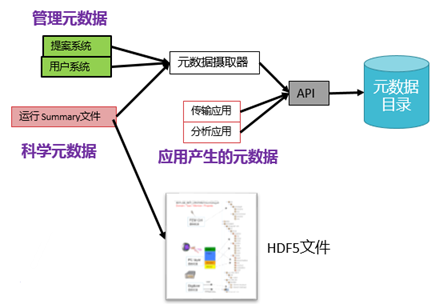
元数据分为三类：管理元数据（administrative metadata）、科学元数据（scientific metadata）、其他元数据（other metadata）。管理元数据包括与数据相关的提案信息、用户信息、实验类型、线站信息等，来自提案系统和用户系统，会被存储到元数据库，用于元数据目录管理。科学元数据包括与数据相关的样本信息和实验环境参数等相关信息，从控制系统获取。由于科学元数据会被用于数据分析和数据目录索引，它需要进行两部分存储：与原始文件一起进行数据封装标准化，生成HDF5文件，同时一部分科学元数据需要被存储到元数据库。其他元数据主要来自传输或者分析应用，包括该数据传输中完整性校验信息（checksum）、分析软件名称版本信息、数据更新时间等，会被存储到元数据库，用于元数据目录管理。各类元数据获取来源详细示意图如下（图4）：

图4 元数据获取来源

元数据管理软件框架提供基于json格式的元数据项和数据格式接口，分别接收元数据提取器、各类第三方应用中的元数据，具体如下图5所示：

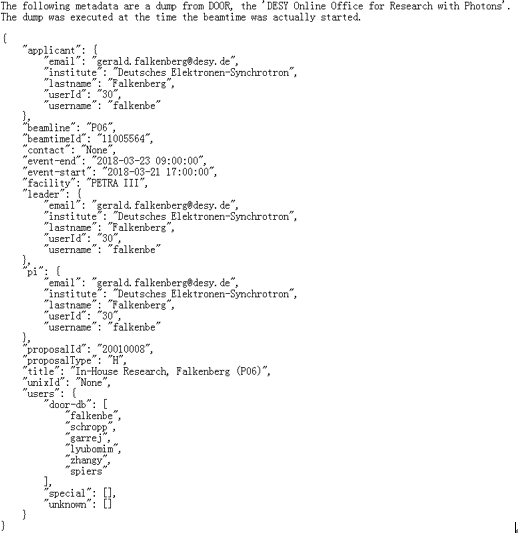


图5 基于json格式的元数据项和数据格式

### 4.1.2 元数据模型结构设计

**逻辑结构**

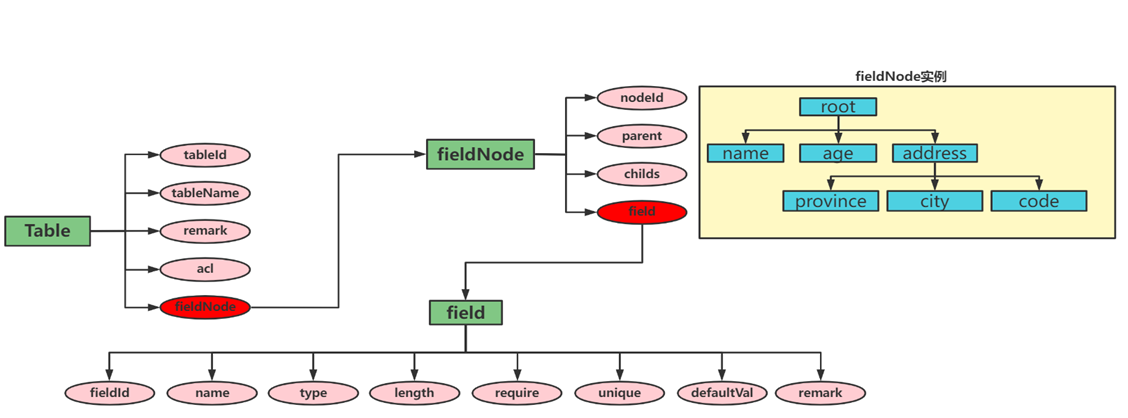
元数据模型是对元数据特征的抽象，是一种面向用户、面向客观世界的模型，主要用来描述元数据的概念化结构。它由模型名称、模型属性、模型之间引用关系等组成。元数据模型逻辑结构如图6所示：

图6 元数据模型逻辑结构

**物理结构**

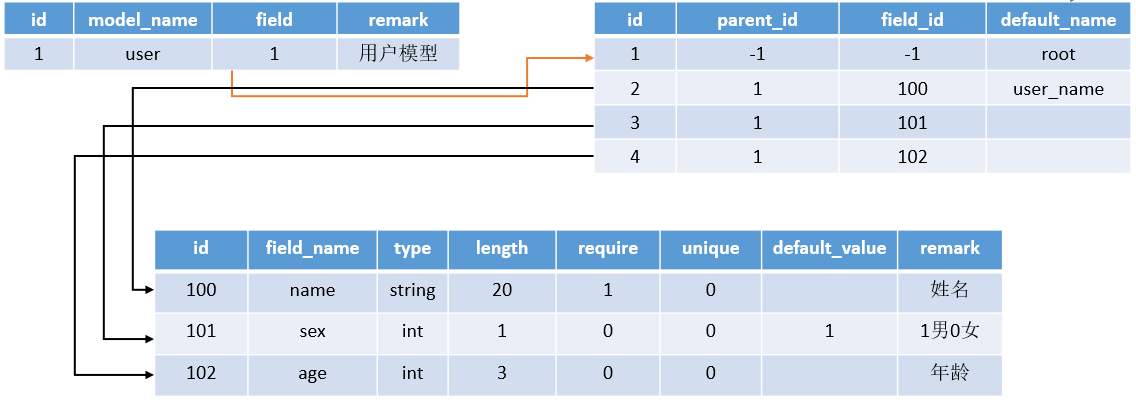
元数据模型的结构信息属于结构化数据，适合用关系型数据库进行组织管理，如图7所示，利用关系型数据库存储元数据模型的结构信息，由模型表、字段项表、字段树表将模型的结构信息进行组织。模型表记录元数据模型的基本信息，字段项表记录每一个字段的详细信息，字段树表记录模型中字段之间的组织结构信息。

图7 元数据模型物理结构

上图可以看出，三张表可以确定一个数据模型的结构，字段项表记录了每一个字段的信息，字段树表的字段field\_id可以关联字段项表的字段id，parent\_id可以组织字段项之间的结构，从而形成字段树，一棵字段树有且仅有一个根节点，模型表字段field只需要记录这个根节点id，能够实现模型和字段集的一一对应关系。

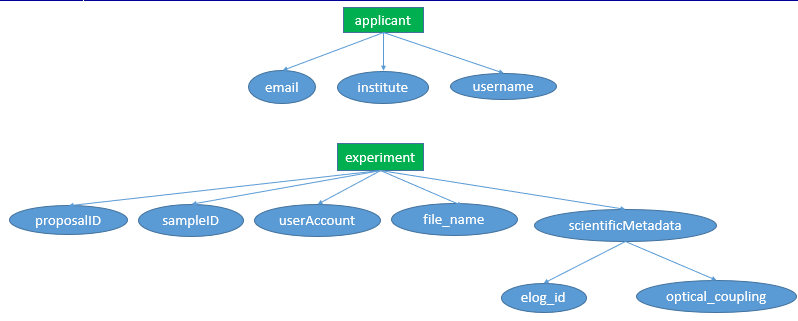
 科学元数据按照json格式传输到元数据管理软件框架，不同类型的元数据结构存在较大差异。如下图8所示，字段集结构可分为两类：线性结构的字段集、树形结构的字段集，元数据模型的字段集如果是线性结构，可以采用同级树形节点存储字段结构（即兼容关系型数据库的字段结构）；如果是树形结构，可以采用多级树形节点存储字段结构（即兼容非关系型数据库的字段结构）

图8 元数据模型的两类字段集结构

### 4.1.3 元数据模型的定义

元数据模型的定义过程本质上是一个创建实体表的过程。用户通过WEB页面创建模型信息，前端负责将用户交互结果组织为json格式的模型定义数据（如图9所示），后端接收到模型json数据后会依次进行配置校验、字段及字段树的提取、字段及字段树的存储、模型关系提取、模型创建。



图9 元数据模型的定义

**配置校验**

配置校验负责校验元数据模型定义配置是否具备合法性，如下图所示，校验的规则包含：①模型名称校验，是否命名合法、已经存在重名模型 ②引用字段项校验，字段集中引用的字段项ID是否真实存在 ③字段树结构校验，配置中组织的字段结构是否符合树形结构 ④模型关系校验，模型关系中引用的模型ID、字段ID是否存在。

**字段及字段树提取**

在元数据模型配置信息中，fileds内容包含了数据项和数据项之间的结构信息，字段间的结构关系通过前端组件的序号方式进行组织，字段和字段树是模型建立的基础。因此，在框架创建模型前需要从配置信息中提取出所有字段项和仅有的一棵字段树。这个过程负责将json配置中的fields内容转换为内存中的字段项对象、字段树对象。

**字段及字段树的存储**

从元数据模型的物理结构可以看出，模型依赖于字段树、字段树依赖于字段项。因此，框架将字段项和字段树信息提取并转换为内存中的对象，为模型创建做好准备工作。为了组织字段项、字段树、模型之间的引用关系，必须先存储字段项，在字段项存储成功后返回对应的字段项ID，再创建字段树，字段树需要引用字段项ID，最后，字段树存储成功后会将其ID返回给模型，并与模型建立对应关系。

****字段项存储时需要考虑字段重复冗余问题，按照框架的设计要求，字段名称、类型、长度、是否必填、是否唯一、默认值均不同，才能保证两个字段项不同。如图10所示，字段名称都为name，但由于字段长度不同，字段1和字段2属于两个不同字段项。

图10 不同字段项的比较

**模型创建**

字段和字段树创建成功后会返回字段树根节点的ID，模型只需要关联字段树根节点ID，即可对应本模型的数据结构信息。

**模型关系提取**

元数据模型之间存在主外键关系，用户在创建模型时会考虑是否本模型存在外键，主外键关系通过模型配置中的relations内容进行约束，约束内容包括：关系类型（1：表示主外键关系 2：表示继承关系，暂时只考虑主外键关系）、被参照模型的ID、被参照模型的字段ID、本模型的参照字段ID。模型创建成功后会返回模型ID、字段ID，框架从配置中提取出relations内容，将模型ID、字段ID绑定到模型关系对象，存入模型关系表。

### 4.1.4 元数据模型创建过程

图11为元数据模型的创建过程：

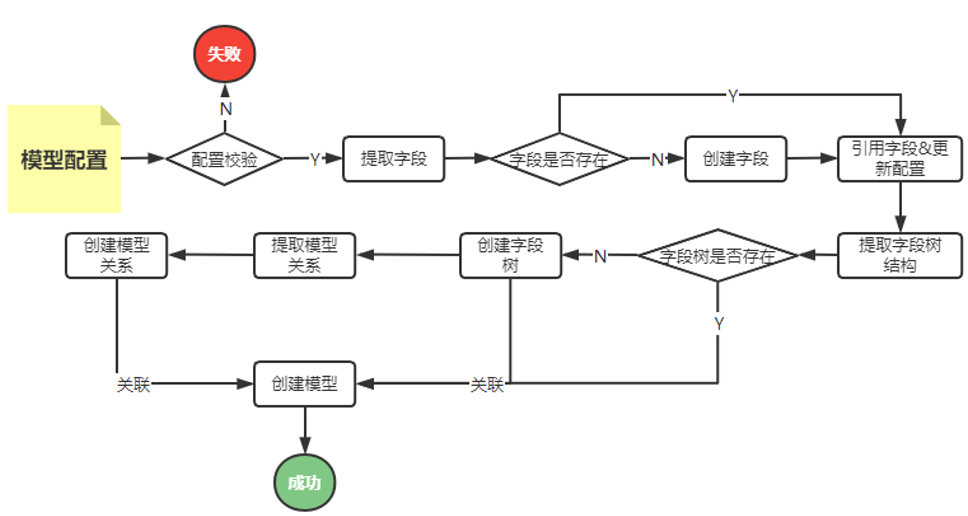


图11 元数据模型创建过程

## 4.2 接口管理

元数据管理软件框架是数据处理平台的中枢大脑，平台内所有系统将元数据按照json格式转发到元数据管理软件框架，框架对所有元数据进行组织、管理，从而提高科学实验数据的利用率，提供更加高效、便捷的数据管理服务。框架基于微服务架构对外提供接口服务，由于数据源种类多、元数据项具有多样不确定性，传统的静态接口（硬编码方式创建固定接口）难以适应用户需求，用户希望基于配置、零代码的方式去创建接口，为数据管理过程提供高效、便捷的接口服务。

**静态接口与动态接口**

目前WEB应用基本上都是采用前后端分离的方式，前端负责在页面渲染后端交互的数据，后端负责为前端提供数据接口，数据接口是需要后端开发人员进行编码完成，重启服务后才能生效，即静态接口；动态接口是基于用户对框架需求提出的新概念，即用户可以在前端仅通过页面交互方式创建一个接口，接口的创建过程由框架自动完成，不再需要开发人员进行编码实现。

**接口注册**

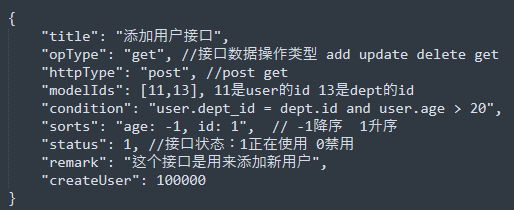
接口是操作数据的工具，数据需要依赖于接口才能进入数据库，在注册接口时必须先确定当前接口需要操作的数据模型对象。接口注册是将接口配置信息（如图12所示）存入系统数据库的过程，即创建接口。接口创建成功后会返回接口ID，接口ID能唯一确定一个接口对象。下图13是一个创建动态接口的过程，通过前端配置即可迅速创建动态接口。

图12 接口配置文件

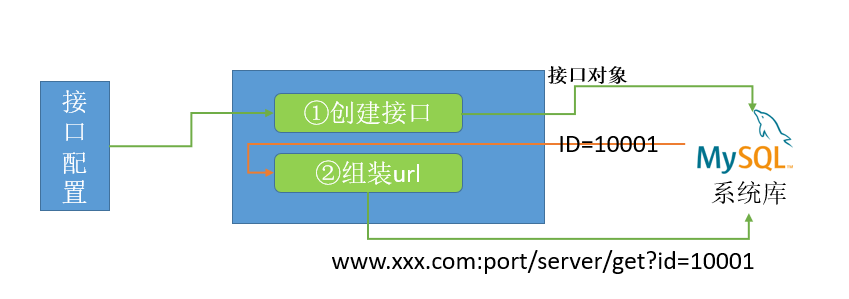
****

图13 动态接口注册过程

**接口校验**

接口校验是对接口配置文件的合法性进行校验，对接口配置进行的合法性验证过程。触发时机：创建或者修改接口对象

**接口解析**

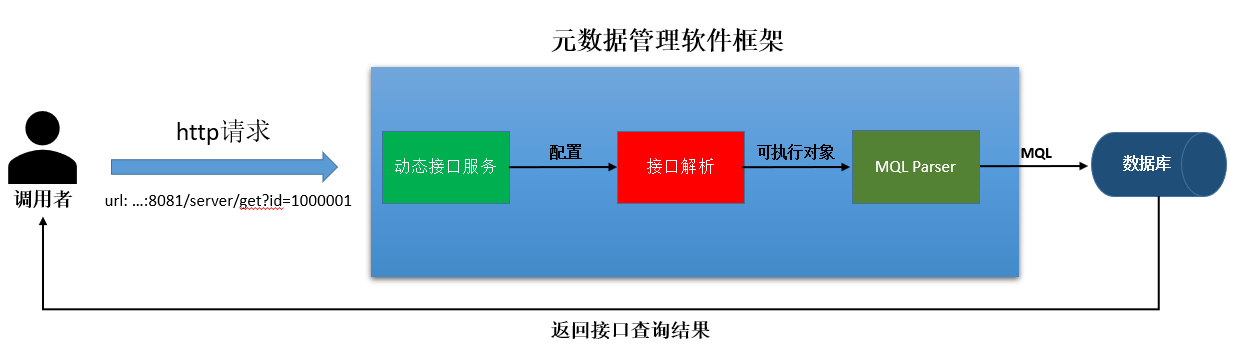
接口注册成功会立即生成url地址，供调用者使用，调用者按照接口的说明（数据开发管理人员约定）进行接口调用。下图所示，接口解析是将配置信息转换为可执行对象的过程，可执行对象本质上是存储配置信息的数据结构（从数据库中把配置信息封装到内存对象中），它存储在内存中。接口被频繁调用的情况下，只需在创建接口时解析接口，生成可执行对象，调用接口时不需要执行接口解析操作。过程如下图13所示：

图13 动态接口解析过程

**数据校验**

接口按照数据操作方式可以分为增加、删除、修改、查询四类，对于查询和删除类型的接口，数据模型对应的MongoDB集合不会受到脏数据的污染；对于增加和修改类型的接口，元数据需要校验通过后才能入库，否则会破坏数据的一致性。数据校验规则如下图14所示：

