Table of Contents

Introduction 1.1
Kyligence大数据分析平台 1.2
概念 1.2.1
基本多维分析 1.3
样例数据集 1.3.1
数据导入 1.3.2
创建数据模型 1.3.3
创建Cube 1.3.4
构建Cube 1.3.5
SQL查询 1.3.6
Web界面 1.4
查询 1.4.1
数据展现 1.4.2
REST API 1.5
访问及安全验证 1.5.1
查询 API 1.5.2
模型 API 1.5.3
Cube API 1.5.4
Job API 1.5.5
数据源 API 1.5.6
缓存 API 1.5.7
驱动程序 1.6
JDBC驱动 1.6.1
ODBC驱动 1.6.2
与第三方系统集成 1.7
Tableau8.x集成 1.7.1
Tableau9.x集成 1.7.2
Apache Zeppelin集成 1.7.3
KyAnalyzer 1.8
KAP可视化分析平台KyAnalyzer简介 1.8.1
如何创建Schema 1.8.2
快速实践 1.9
常见问题 1.10
联系我们 1.11

KAP 2.0 User Manual

Kyligence Analytics Platform（简称KAP）是领先的大数据分析平台。

本手册可以在Gitbook浏览或下载。

上海跬智信息技术有限公司

Kyligence大数据分析平台

Kyligence大数据分析平台(Kyligence Analytics Platform, KAP)是基于Apache Kylin实现的、为可伸缩数据集提供分析能力的企业级大数据产品，在Apache Hadoop上为百亿及以上超大规模数据集提供亚秒级标准SQL查询能力，为业务用户、分析师及工程师提供简单、快捷的大数据分析能力。Apache Kylin在业界已经被多个公司采用已解决各种大数据分析挑战。

相较于其它技术，KAP具有以下特点：

- 亚秒级查询

在百亿及以上规模数据集上为业务用户及分析师提供亚秒级的查询速度，并同时支持高并发，使得在大数据平台上对超大规模数据进行交互式分析成为可能；支持ANSI SQL查询标准，使得业务用户及分析师无需重新学习新的技术即可掌握在海量超大规模数据集上快速分析的能力。

- 无缝集成

支持与企业级商业智能(BI)及可视化工具无缝集成，提供标准的ODBC、JDBC驱动及REST API接口等以连接流行的数据分析、展示工具，如Tableau、Microsoft PowerBI、Microsoft Excel、Apache Zeppelin、Saiku等

- 自助服务

Kyligence大数据分析平台使得分析师及用户能以简洁而快速的方式分析海量数据。易于使用的Web界面允许用户自己构建数据集市而无需知晓底层技术 。

- 可扩展架构

全新设计的可扩展架构从根本上解藕了对特定技术的依赖，将计算框架，数据源以及底层存储等扩展到更多的技术领域，为不同的技术栈提供可配置的优化解决方案。

- 非侵入式

Kyligence大数据分析平台的部署不需要在现有Hadoop集群上安装任何新的组件，更不需要在数据节点或其他节点上安装Agent等，所有与集群的操作都通过标准API完成，从而使得对现有集群的影响最小化，也为快速部署带来了可能。

更多信息，请访问Kyligence网站: http://kyligence.io/.

本节介绍了在KAP中使用的基础概念。对于某些领域术语，请参考章节“术语”。

CUBE

- **Table** - 表， 是Cube的数据源；在创建Cube之前，KAP需要从数据源（通常为Hive）同步表的元数据，包含表名、列名、列属性等。
- **Data Model** - 数据模型，定义了由若干张表的一个连接关系。KAP支持星型模型(https://en.wikipedia.org/wiki/Star_schema)的多维分析；在创建Cube之前，用户需定义这么一个数据模型。
- **Cube** - 数据立方体，是一种多维分析的技术，通过预计算，将计算结果存储在某多个维度值所映射的空间中；在运行时通过对Cube的再处理而快速获取结果。
- **Partition** - 分区，用户可以定义一个分区日期或时间列，随后对Cube的构建按此列的值范围而进行，从而将Cube分成多个Segment。
- **Cube Segment** - 每个Cube Segment是对特定时间范围的数据计算而成的Cube。每个Segment对应一张HBase表。
- **Aggregation Group** - 聚合组，每个聚合组是全部维度的一个子集；通过将很多个维度分组，并把常一起使用的维度放在一起，可以有效降低Cube的组合数。

维度 & 度量

- **Mandotary** - 必需的维度：这种类型用于对Cube生成树做剪枝：如果一个维度被标记为“Mandatory”，会认为所有的查询都会包含此维度，故所有不含此维度的组合，在Cube构建时都会被剪枝（不计算）.
- **Hierarchy** - 层级维度：如果多个维度之间有层级（或包含）的关系，通过设置为“Hierarchy”，那些不满足层级的组合会被剪枝。如果A, B, C是层级，并且A>B>C，那么只需要计算组合A, AB, ABC; 其它组合如B, C, BC, AC将不做预计算。
- **Derived** - 衍生维度：维度表的列值，可以从它的主键值衍生而来，那么通过将这些列定义为衍生维度，可以仅将主键加入到Cube的预计算来，而在运行时通过使用维度表的快照，衍生出非PK列的值，从而起到降维的效果。
- **Count Distinct(HyperLogLog)** - 基于HyperLogLog的Count Distint：快速、精确的COUNT DISTINCT是较难计算的, 一个近似的轻量级算法 - HyperLogLog 为此而发明, 能够在大规模数据集上做去重并保持较低的误差率.
- **Count Distinct(Bitmap)** - 基于Bitmap的COUNT DISTINCT，可以精确去重，但是存储开销较大。目前只支持int的数据类型.
- **Top N** - 预计算最top的某些记录的度量，如交易量最大的1000个卖家。

CUBE 操作

- **BUILD** - 构建：给定一个时间范围，将源数据通过运算而生成一个新的Cube Segment。
- **REFRESH** - 刷新：对某个已经构建过的Cube Segment，重新从数据源抽取数据并构建，从而获得更新。
- **MERGE** - 合并：将多个Cube Segment合并为一个Segment。这个操作可以减少Segment的数量，同时减少Cube的存储空间。
- **PURGE** - 清空：将Cube的所有Cube Segment删除。

JOB 状态

- **NEW** - 新任务，刚刚创建。
- **PENDING** - 等待被调度执行的任务.
- **RUNNING** - 正在运行的任务。
- **FINISHED** - 正常完成的任务（终态）。
- **ERROR** - 执行出错的任务。
- **DISCARDED** - 丢弃的任务（终态）。

JOB 操作

- **RESUME** - 恢复：处于ERROR状态的任务，用户在排查或解决问题后，通过此操作来重试执行。
- **DISCARD** - 丢弃：放弃此任务，立即停止执行并不会再恢复。

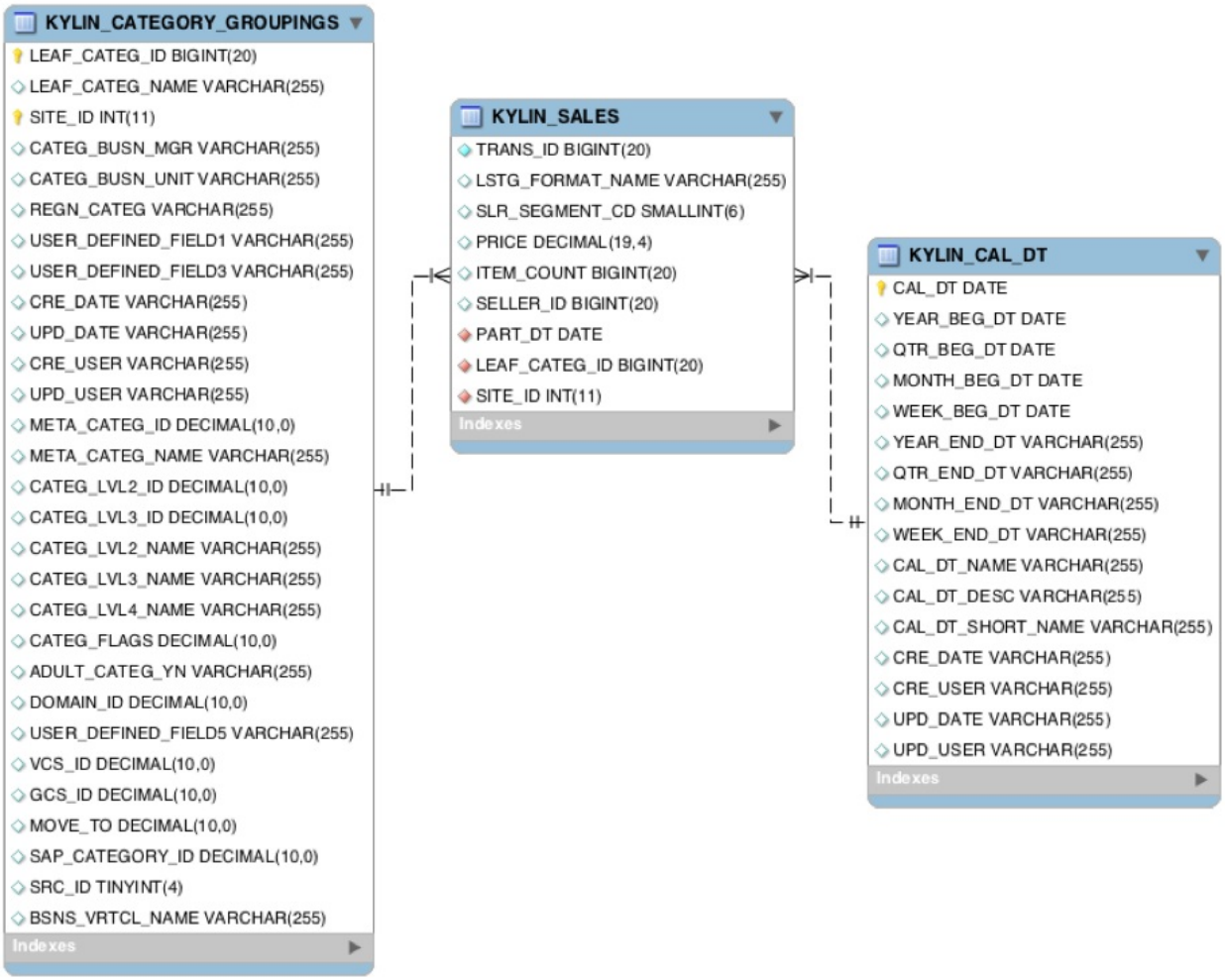
样例数据集

KAP二进制包中包含了一份用于测试的样例数据集，总共大小仅1MB左右，共计3张表，其中事实表有10000条数据。因为数据规模较小，方便在虚拟机中进行快速实践和操作，用户可以自行搭建Hadoop Sandbox的虚拟机并快速部署KAP，然后导入该数据集进行试验。

KAP仅支持星型数据模型，这里用到的样例数据集就是一个规范的星型模型结构，它总共包含了3个数据表：

- KYLIN_SALES 该表是事实表，保存了销售订单的明细信息。每一列保存了卖家、商品分类、订单金额、商品数量等信息，每一行对应着一笔交易订单。
- KYLIN_CATEGORY_GROUPINGS 该表是维表，保存了商品分类的详细介绍，例如商品分类名称等。
- KYLIN_CAL_DT 该表是维表，保存了时间的扩展信息。如单个日期所在的年始、月始、周始、年份、月份等。

这三张表一起构成了整个星型模型的结构，下图是实例-关系图（图中未列出表上的所有列）：



数据表与关系

在介绍完表间关系之后，我们再对一些表中的关键字段进行介绍：

表	字段	意义
KYLIN_SALES	PART_DT	订单日期
KYLIN_SALES	LEAF_CATEG_ID	商品分类ID
KYLIN_SALES	SELLER_ID	卖家ID
KYLIN_SALES	PRICE	订单金额
KYLIN_SALES	ITEM_COUNT	购买商品个数
KYLIN_SALES	LSTG_FORMAT_NAME	订单交易类型
KYLIN_CATEGORY_GROUPINGS	USER_DEFINED_FIELD1	用户定义字段1
KYLIN_CATEGORY_GROUPINGS	USER_DEFINED_FIELD3	用户定义字段3
KYLIN_CATEGORY_GROUPINGS	UPD_DATE	更新日期
KYLIN_CATEGORY_GROUPINGS	UPD_USER	更新负责人
KYLIN_CATEGORY_GROUPINGS	META_CATEG_NAME	一级分类
KYLIN_CATEGORY_GROUPINGS	CATEG_LVL2_NAME	二级分类
KYLIN_CATEGORY_GROUPINGS	CATEG_LVL3_NAME	三级分类
KYLIN_CAL_DT	CAL_DT	日期
KYLIN_CAL_DT	WEEK_BEG_DT	周始日期

导入Hive数据源

目前，KAP支持Hive作为默认的输入数据源。为了使用KAP中自带的样例数据，需要把数据表导入Hive中。在KAP安装目录的bin文件夹中，有一个可执行脚本，可以把样例数据导入Hive：

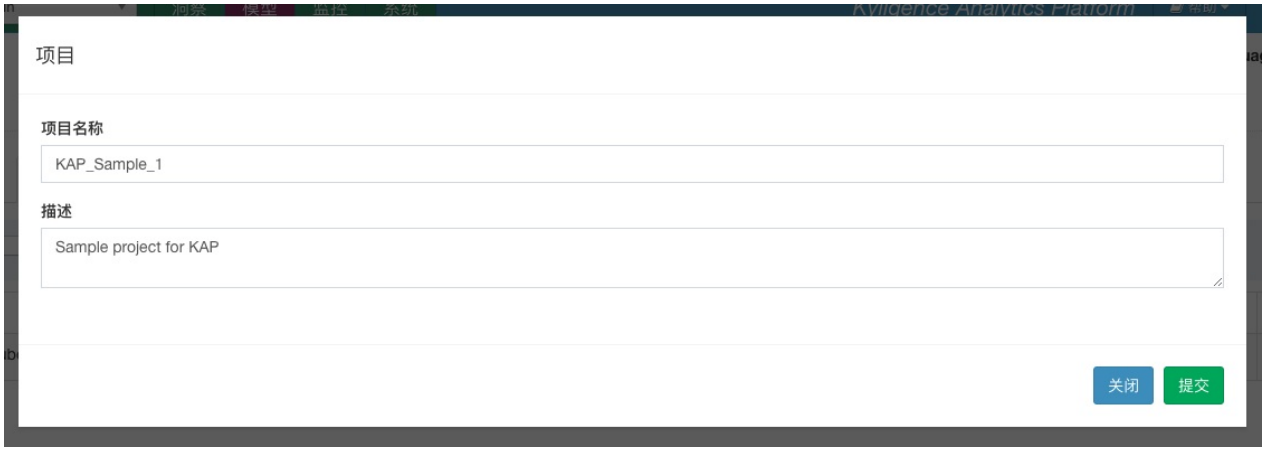
```
$KYLIN_HOME/bin/sample.sh
```

脚本执行成功之后，在服务器执行Hive命令行，确认这些数据已经导入成功，命令如下：

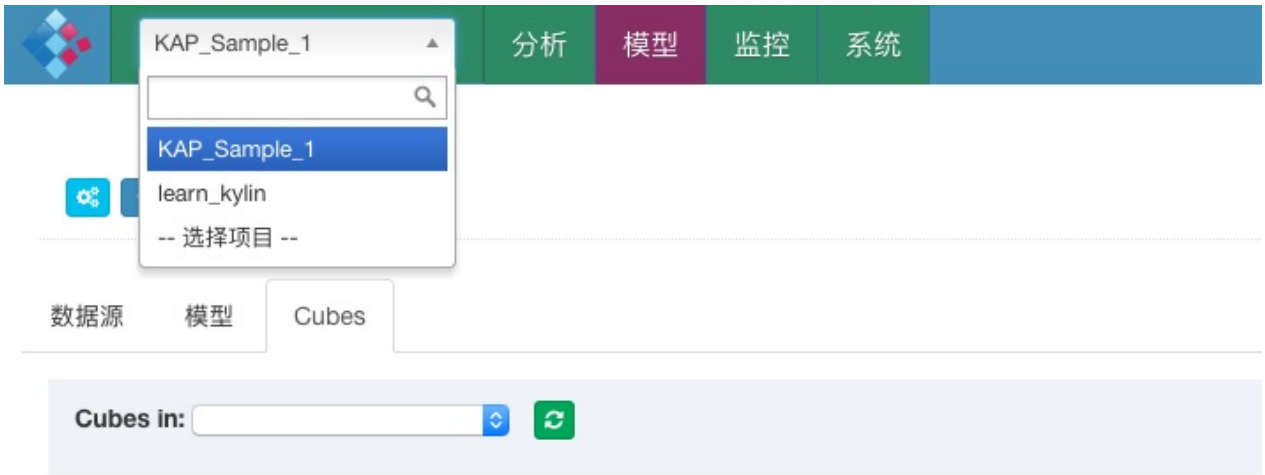
```
hive
hive> show tables;
OK
kylin_cal_dt
kylin_category_groupings
kylin_sales
Time taken: 0.127 seconds, Fetched: 3 row(s)
hive> select count(*) from kylin_sales;
Query ID = root_20160707221515_b040318d-1f08-44ab-b337-d1f858c46d7d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1467288198207_0129, Tracking URL = http://sandbox.hortonworks.com:80
88/proxy/application_1467288198207_0129/
Kill Command = /usr/hdp/2.2.4.2-2/hadoop/bin/hadoop job  -kill job_1467288198207_0129
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-07-07 22:15:11,897 Stage-1 map = 0%,  reduce = 0%
2016-07-07 22:15:17,502 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.64 sec
2016-07-07 22:15:25,039 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.37 sec
MapReduce Total cumulative CPU time: 3 seconds 370 msec
Ended Job = job_1467288198207_0129
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.37 sec   HDFS Read: 505033 HDFS Wr
ite: 6 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 370 msec
OK
10000
Time taken: 24.966 seconds, Fetched: 1 row(s)
```

创建项目

打开KAP的Web UI，如下图所示的操作创建一个新的项目（Project），并命名为KAP_Sample_1。



在Web UI的左上角选择刚刚创建的项目，表示我们接下来的全部操作都在这个项目中，在当前项目的操作不会对其他项目产生影响。



同步Hive表

需要把Hive数据表同步到KAP当中才能使用。为了方便操作，我们通过“使用图形化树状结构方式加载Hive表”进行同步，如下图所示：

在弹出的对话框中展开default数据库，并选择需要的三张表，如图所示

导入后系统会自动计算各表各列的维数，以掌握数据的基本情况。稍等几分钟后，我们可以通过数据源表的详情页查看这些信息。

数据源 模型 Cubes

表

- DEFAULT
  - KYLIN_CAL_DT
  - KYLIN_CATEGORY_GROUPINGS
  - KYLIN_SALES
    - TRANS_ID(bigint)
    - PART_DT(date)
    - LSTG_FORMAT_NAME(varchar(256))
    - LEAF_CATEG_ID(bigint)
    - LSTG_SITE_ID(integer)
    - SLR_SEGMENT_CD(smallint)
    - PRICE(decimal(19,4))
    - ITEM_COUNT(bigint)
    - SELLER_ID(bigint)

表结构:KYLIN_SALES

列

列

过滤器 ...

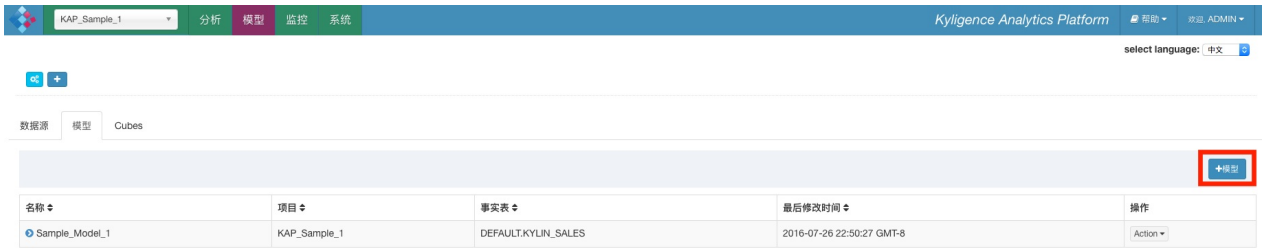
ID	名称	数据类型	基数
1	TRANS_ID	bigint	1
2	PART_DT	date	736
3	LSTG_FORMAT_NAME	varchar(256)	5
4	LEAF_CATEG_ID	bigint	136
5	LSTG_SITE_ID	integer	7
6	SLR_SEGMENT_CD	smallint	8
7	PRICE	decimal(19,4)	10000
8	ITEM_COUNT	bigint	1
9	SELLER_ID	bigint	955

创建数据模型

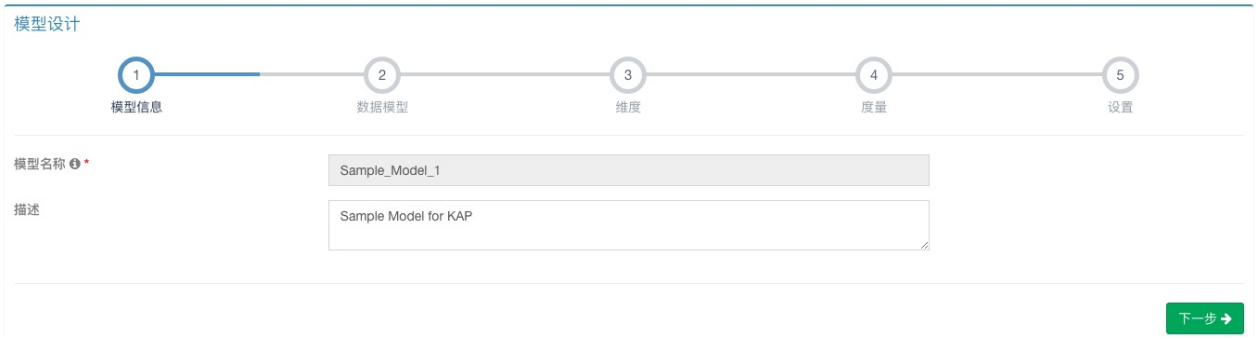
在数据源就绪的基础之上，我们开始创建数据模型。以KAP自带的数据集为例，该数据集的数据模型包含1个事实表和2个维表，表间通过外键进行关联。实际上，并不是表上所有的字段都有被分析的需要，因此我们可以有目的地仅选择所需字段添加到数据模型中；然后，根据分析师用户的具体分析场景，把这些字段设置为维度或度量。

创建步骤

打开KAP的Web UI，在左上角项目列表中选择刚刚创建的KAP_Sample_1项目，然后进入“模型”页面，并创建一个模型。



第一步，在基本信息页，输入模型名称为Sample_Model_1，然后单击下一步。



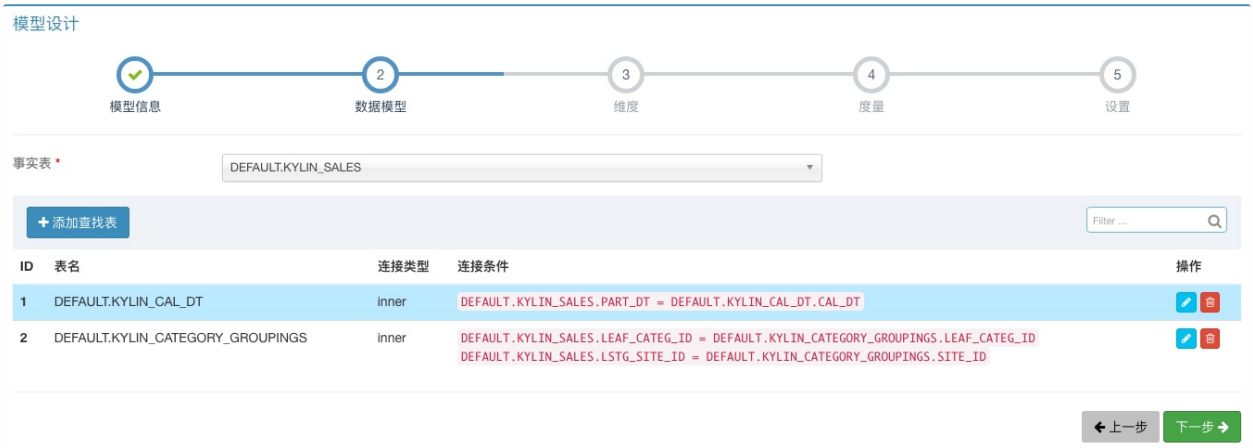
第二步，为模型选择事实表（Fact Table）和查找表（Lookup Table）。根据星型模型结构，选择KYLIN_SALES为事实表，然后添加KYLIN_CAL_DT和KYLIN_CATEGORY_GROUPINGS作为查找表，并设置好连接条件：

KYLIN_CAL_DT

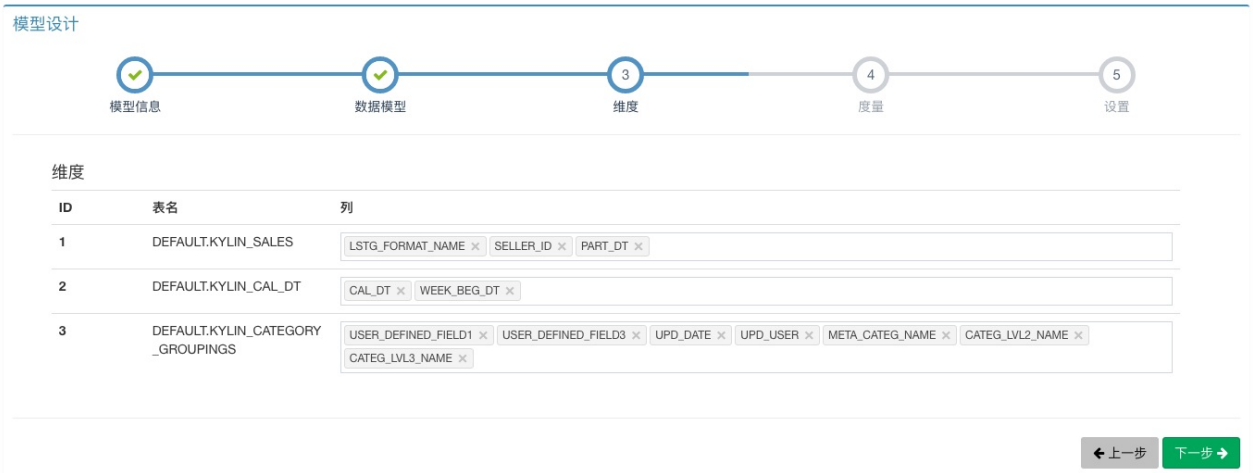
- 连接类型：Inner
- 连接条件：DEFAULT.KYLIN_SALES.PART_DT = DEFAULT.KYLIN_CAL_DT.CAL_DT

KYLIN_CATEGORY_GROUPINGS

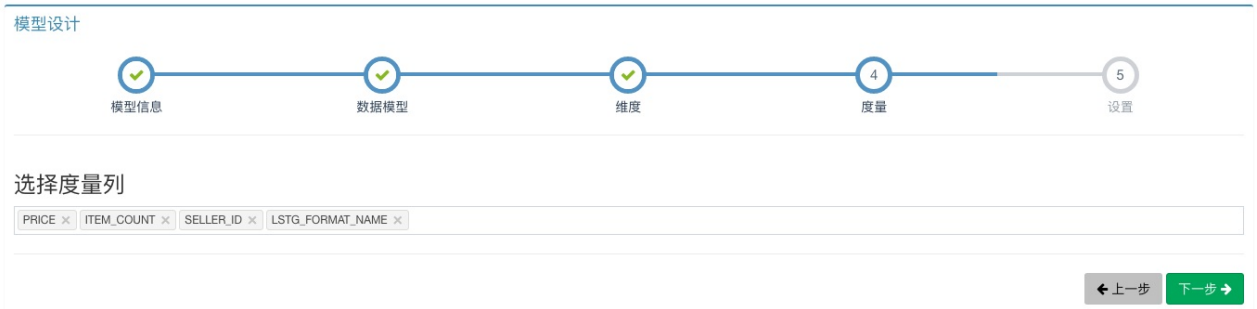
- 连接类型：Inner
- 连接条件:KYLIN_SALES.LEAF_CATEG_ID=KYLIN_CATEGORY_GROUPINGS.LEAF_CATEG_ID AND KYLIN_SALES.LSTG_SITE_ID=KYLIN_CATEGORY_GROUPINGS.SITE_ID 下图是设置好之后的界面：



第三步，从上一步添加的事实表和查找表中选择需要作为维度的字段。一般的，时间经常用来作为过滤条件，所以我们首先添加时间字段。此外，我们再添加商品分类、卖家ID等字段为维度，具体情况如下图所示：



第四步，根据业务需要，从事实表上选择衡量指标的字段作为度量。例如，PRICE字段用来衡量销售额，ITEM_COUNT字段用来衡量商品销量，SELLER_ID用来衡量卖家的销售能力。最终结果如下图所示：



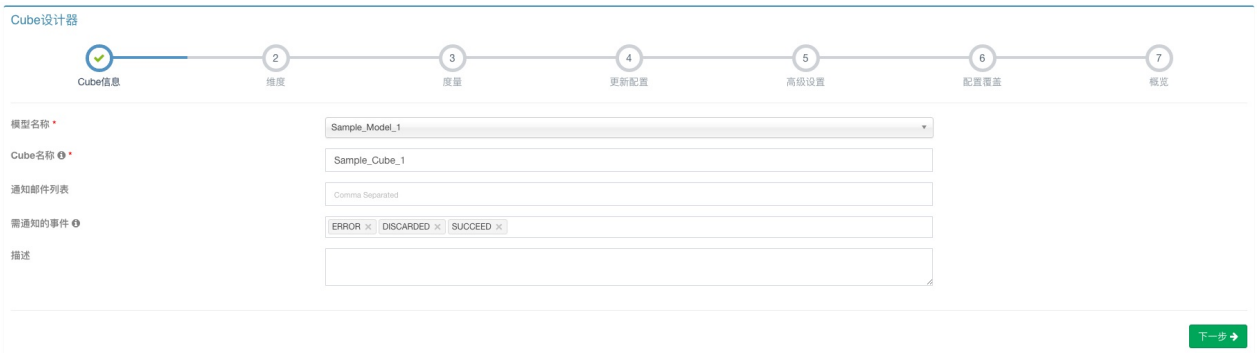
第五步，设置根据时间分段。一般来说，销售数据都是与日俱增的，每天都会有新数据通过ETL到达Hive中，需要选择增量构建方式构建Cube，所以需要选择用于分段的时间字段DEFAULT.KYLIN_SALES.PART_DT。根据样例数据可以看到，这一列时间的格式是yyyy-MM-dd，所以选择对应的日期格式。此外，我们既不需要设置单独的分区时间列，也不需要添加固定的过滤条件。设置效果如下图所示。

最终，单击“保存”按钮，到此数据模型就创建成功了。

创建Cube

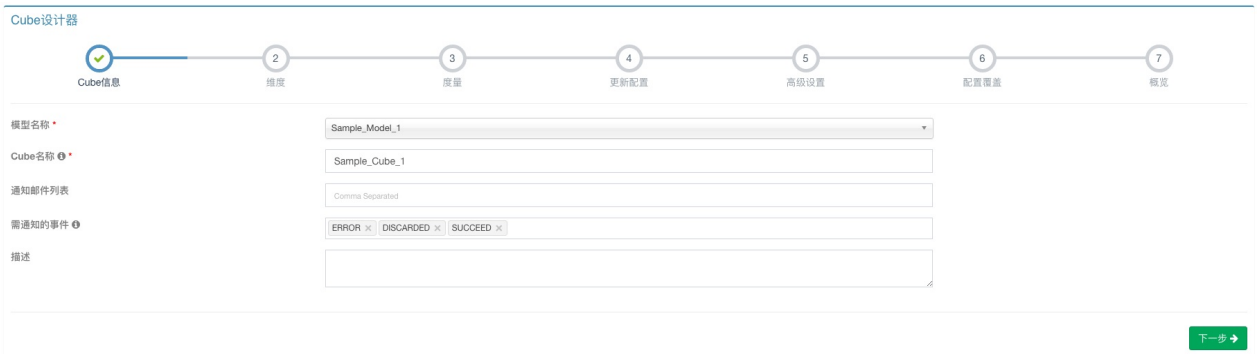
在创建好数据模型的基础上，我们还需要根据查询需求定义度量的预计算类型、维度的组合等，这个过程就是Cube设计的过程。本文将以KAP自带的样例数据为例，介绍Cube的创建过程。 打开KAP的Web UI，首先选择KAP_Sample_1项目，跳转到模型页面，然后按照下图所示创建一个Cube。

第一步，在“模型名称”中选择Sample_Model_1，输入新建Cube的名称Sample_Cube_1，其余字段保持默认，然后单击“下一步”。

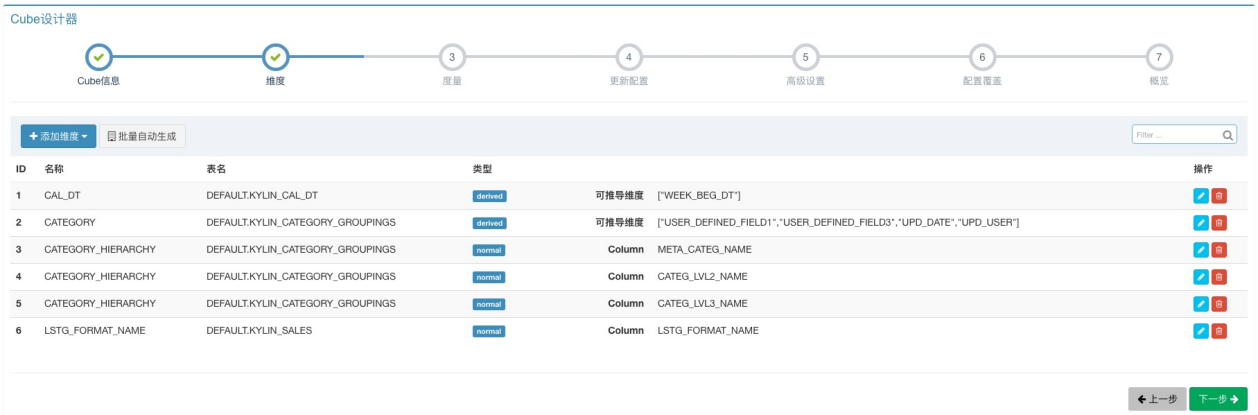


第二步，从数据模型的维度中选择一些列作为Cube的维度。这里的设置会影响到生成的Cuboid数量，进而影响Cube的数据量大小。 在KYLIN_CATEGORY_GROUPINGS表里，和商品分类相关的三个字段（META_CATEG_NAME、CATEG_LVL2_NAME、CATEG_LVL3_NAME）都可能出现在过滤条件中，我们先把他们添加为普通类型维度。因为从查询表上添加普通维度不能通过自动生成器（Auto Generator）生成，因此采用手动添加方式，过程如下：

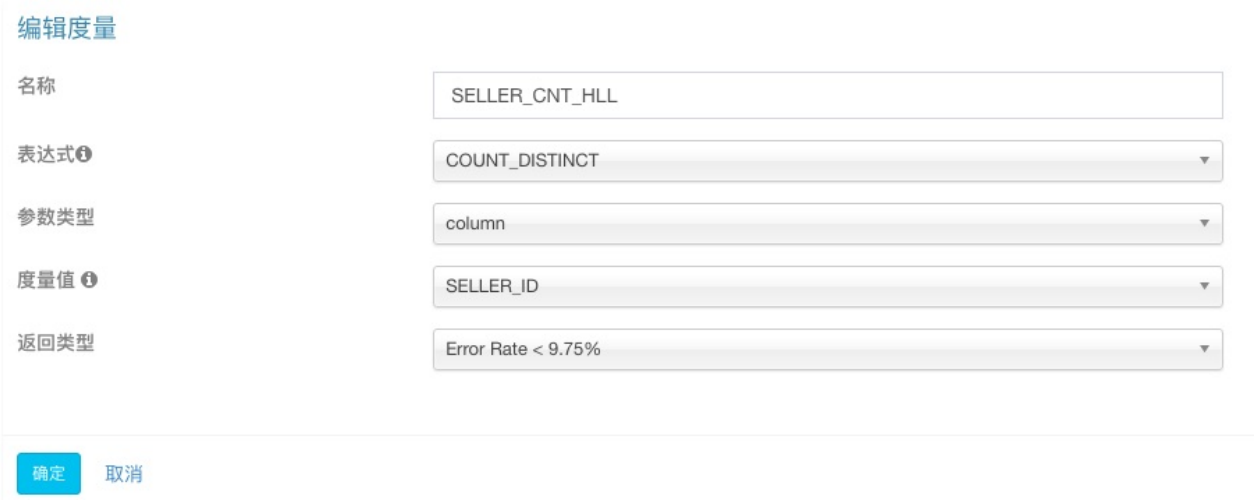
1. 单击“添加维度”按钮，然后选择“普通维度”。
2. 针对每一个维度字段，首先在Name输入框中输入维度名称，在Table Name中选择KYLIN_CATEGORY_GROUPINGS表，然后在Column Name中选择相应的列名。 此外，在查询中还经常把时间作为过滤或聚合的条件，如按周过滤、按周聚合等。这里我们以按周为例，需要用到KYLIN_CAL_DT中的WEEK_BEG_DT字段，但是该字段实际上可以由PART_DT字段决定，即根据每一个PART_DT值可以对应出一个WEEK_BEG_DT字段，因此，我们添加WEEK_BEG_DT字段为可推倒维度。 同样的，KYLIN_CATEGORY_GROUPINGS表中还有一些可作为可推到维度的字段，如USER_DEFINED_FIELD1、USER_DEFINED_FIELD3、UPD_DATE、UPD_USER等。 在事实表上，表征交易类型的LSTG_FORMAT_NAME字段也会用于过滤或聚合条件，因此，我们再添加LSTG_FORMAT_NAME字段作为普通维度。 最终，维度的设置结果如下图所示：



第三步，根据数据分析中的聚合需求，我们需要为Cube定义度量的聚合形式。默认的，系统会自动创建好一个COUNT()聚合，用于考量交易订单的数量。在这个案例中，我们还需要通过PRICE的不同聚合形式考量销售额，如总销售额为SUM(PRICE)、最高订单金额为MAX(PRICE)、最低订单金额为MIN(PRICE)。因此，我们手动创建三个度量，分别选择聚合表达式为SUM、MIN、MAX，并选择PRICE列作为目标列。



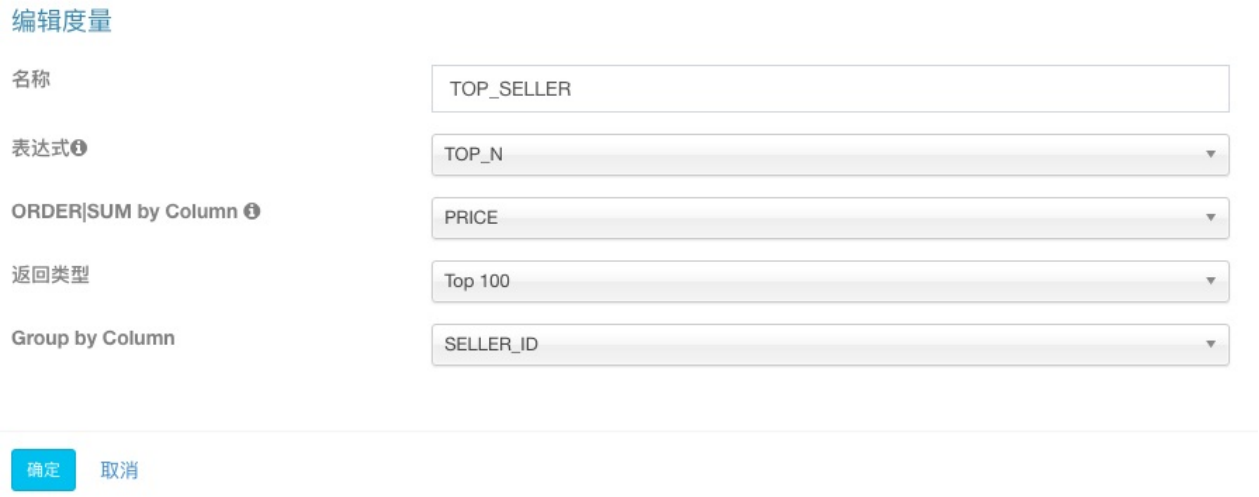
其次，我们还需要通过COUNT(DISTINCT SELLER_ID)考量卖家个数。根据前面章节的介绍，KAP默认使用HyperLogLog算法进行COUNT_DISTINCT的计算，该算法是个近似算法，在创建度量时需要选择一个近似度，本案例对精确性要求不高，为了提升查询性能，我们选择精度较低的“Error Rate < 9.75%”。同样的，我们再创建一个COUNT(DISTINCT LSTG_FORMAT_NAME)的度量考量不同条件下的交易类型。



在销售业务分析的场景中，往往需要挑选出销售业绩最好的商家，这时候就需要TOP-N的度量了。在这个例子中，我们会选出SUM(PRICE)最高的一些SELLER_ID，实际上就是执行如下的SQL语句：

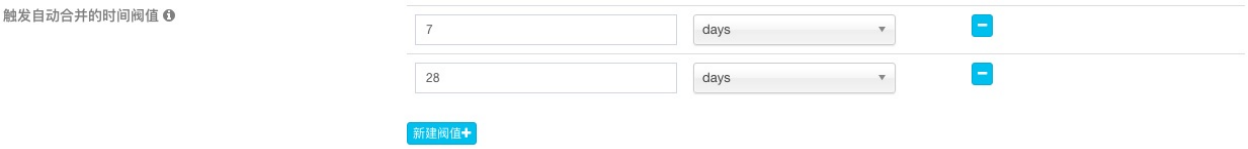
```
SELECT SELLER_ID, SUM(PRICE) FROM KYLIN_SALES
GROUP BY SELLER_ID
ORDER BY SUM(PRICE)
```

因此，我们创建一个TOP-N的度量，选择PRICE字段作为SUM/OPDERBY字段，选择SELLER_ID字段作为GROUP BY字段，并选择TOPN(100)作为度量的精度。



最终添加的度量如下图所示：

第四步，我们对Cube的构建和维护进行配置。一般的，一个销售统计的SQL查询往往会按月、周进行过滤和聚合，所以我们可以设置Cube自动按周、月进行自动合并，即每7天进行一次合并，每4周（28天）进行一次合并，设置“触发自动合并的时间阀值”如下所示：



因为存在对于历史订单的查询需求，我们在此不对Cube做自动清理，所以需要设置“保留时间阀值”为0。

在创建数据模型的时候我们提到，我们希望采用增量构建方式对Cube进行构建，并选择了PART_DT字段作为分区时间列。在创建Cube时，我们需要指定Cube构建的起始时间，在这个例子中，根据样例数据中的时间条件，我们选择2012-01-01 00:00:00作为分区起始时间。 第五步，通过对Cube进行高级设置优化Cube的存储大小和查询速度，主要包括聚合组和Rowkey。在第六章我们提到，添加聚合组可以利用字段间的层级关系和包含关系有效地降低Cuboid的数量。在这个案例当中，与商品分类相关的三个字段（META_CATEG_NAME、CATEG_LVL2_NAME、CATEG_LVL3_NAME）实际上具有层级关系，如一级类别（META_CATEG_NAME）包含多个二级类别（CATEG_LVL2_NAME），二级类别又包含多个三级类别（CATEG_LVL3_NAME），所以，我们可以为它们创建层级结构的组合（Hierarchy Dimensions）。最终，聚合组的设计如下图所示：

聚合组

点击 聚合组 查看更多相关聚合组信息

1	包含的维度	CATEG_LVL2_NAME × CATEG_LVL3_NAME × LEAF_CATEG_ID × LSTG_FORMAT_NAME × LSTG_SITE_ID × META_CATEG_NAME × PART_DT ×
	必需维度	选择列
	层级维度	META_CATEG_NAME × CATEG_LVL2_NAME × CATEG_LVL3_NAME × 新的层级+
	组合维度	新的组合+

新建聚合组+

由于参与Cuboid生成的维度都会作为Rowkey，因此我们需要把这些列添加为Rowkey当中。在这个案例中，总共需要添加7个Rowkey。在每个Rowkey上，还需要为列值设置编码方法。在这个案例中，我们除了把LSTG_FORMAT_NAME设置为fixed_length类型（长度为12）外，将其余的Rowkey都设置为dict编码。Rowkey的顺序对于查询性能来说至关重要，如第六章所讲，一般把最经常出现在过滤条件中的列放置在Rowkey的前面，在这个案例中，我们首先把PART_DT放在Rowkey的第一位。接下来，按照层级把商品分类的字段跟随其后。最终，Rowkey的设置如下图所示：

Rowkey

ID	列	编码Encoding	长度	Shard By
1	PART_DT	date	0	false
2	LEAF_CATEG_ID	dict	0	false
3	META_CATEG_NAME	dict	0	false
4	CATEG_LVL2_NAME	dict	0	false
5	CATEG_LVL3_NAME	dict	0	false
6	LSTG_FORMAT_NAME	fixed_length	12	false
7	LSTG_SITE_ID	dict	0	false

新加Rowkey列+

第六步，设置Cube的配置覆盖。在这里添加的配置项可以在Cube级别覆盖从kylin.properties配置文件读取出来的全局配置。在这个案例中，我们可以直接采用默认配置，在此不做任何修改。

第七步，对Cube的信息进行概览。请读者仔细确认这些基本信息，包括数据模型名称、事实表以及维度和度量个数。确认无误后单击“保存”按钮，并在弹出的确认提示框中选择“Yes”。

最终，Cube的创建就完成了。我们可以刷新Model页面，在Cube列表中就可以看到新创建的Cube了。因为新创建的Cube没有被构建过，是不能被查询的，所以状态仍然是“禁用”。

数据源 模型 Cubes

Cubes in: +Cube

名称	模型	状态	Cube大小	源数据条目	最后构建时间	所有者	创建时间	操作	Admins	Streaming
Sample_Cube_1	Sample_Model_1	DISABLED	0.00 KB	0		ADMIN	2016-07-27 00:54:16 GMT-8	Action	Action	false

总数: 1
存储: 0.00 KB

构建Cube

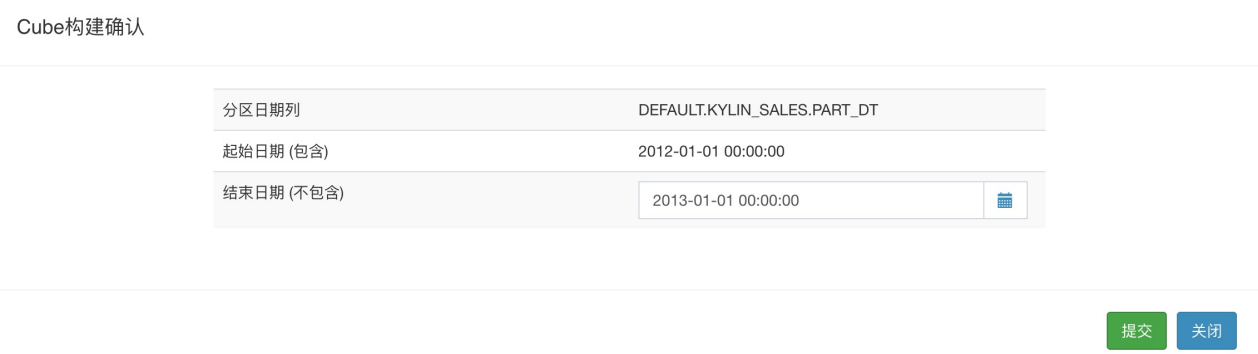
在创建好Cube之后，只有对Cube进行构建，才能利用它执行SQL查询。本文以KAP样例数据为例，介绍Cube构建的过程。

初次构建

首先打开KAP的Web UI，并选择Kylin_Sample_1项目，然后跳转到模型页面，找到Cube列表。 第一步，在Cube列表中找到Kylin_Sample_Cube_1。单击右侧的Action按钮，在弹出的菜单中选择“构建”。

第二步，在弹出的Cube构建确认对话框中确认Cube的分段时间列（Partition Date Column）是DEFAULT.KYLIN_SALES.PART_DT，以及起始时间是2012-01-01 00:00:00。在KAP中，一次构建会为Cube产生一个新的Segment，每次的SQL查询都会访问一个或多个符合条件的Segment；我们需要尽可能地让一个Segment更好地适用于查询条件，因此我们可以按年构建，即每个年份构建一个Segment。在这个例子中，我们输入结束日期（End Date）为2013-01-01 00:00:00。设置完成后单击Submit按钮。

> 注意：增量构建是具体按年构建还是按月构建应该根据实际的业务需求、ETL时效及数据量大小而定。如果一次构建的数据量过大，可能导致构建时间过长，或出现内存溢出等异常。在当前的样例数据中，数据量较小，按年构建是可以顺利完成的。

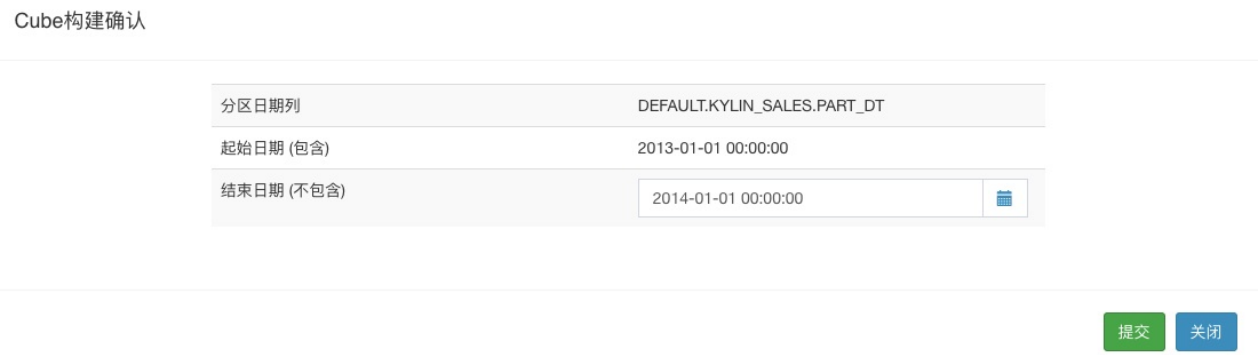


当任务成功提交之后，切换到Monitor页面，这里会列出所有的任务列表。我们找到列表最上面的一个任务（名称是：Kylin_Sample_Cube_1 - 20120101000000_20130101000000），这就是我们刚刚提交的任务。在这一行双击或单击右侧的箭头图标，页面右侧会显示当前任务的详细信息。 待构建任务完成，可以

在Monitor页面看到该任务状态已被置为完成（Finished）。这时候，第一个Segment就构建完成了。前往Cube列表中查看，会发现该Cube的状态已被置为“就绪（Ready）”了。

增量构建

在第一个Segment构建完成之后，我们开始构建第二个Segment。首先在Model页面的Cube列表中找到该Cube，单击右侧的Actions按钮，然后选择“Build”，打开Cube构建确认对话框。 在这个对话框中，首先确认起始时间（Start Date）是2013-01-01 00:00:00，因为这是上次构建的结束日期，为保障所构建数据的连续性，Apache Kylin自动为新一次构建的起始时间更新为上次构建的结束日期。同样的，在结束时间（End Date）里输入2014-01-01 00:00:00，然后单击Submit按钮，开始构建下一年的Segment。



待构建完成，我们可以在Cube的详情页中查看，发现Cube的两个Segment都已就绪。

SQL查询

当Cube构建任务完成，系统一般会自动把Cube的状态切换为就绪（Ready）。我们以KAP样例数据为例介绍SQL查询方式。 首先切换到Insight页面，在Web UI上选择本案例所用的Kylin_Sample_1项目。然后根据Cube上维度和度量的设计，在查询输入框中输入SQL语句，然后单击Submit按钮。下面给出一下SQL查询的例子和相应的结果介绍。

单表行数统计

```
SELECT COUNT(*) FROM KYLIN_SALES
```

这条SQL用于查询KYLIN_SALES表中总行数，读者可以同时在Hive中执行同样的查询进行性能对比。在笔者的对比中，Hive查询耗时29.385秒，KAP查询耗时0.18秒。

多表连接

```
SELECT
KYLIN_SALES.PART_DT
,KYLIN_SALES.LEAF_CATEG_ID
,KYLIN_SALES.LSTG_SITE_ID
,KYLIN_CATEGORY_GROUPINGS.META_CATEG_NAME
,KYLIN_CATEGORY_GROUPINGS.CATEG_LVL2_NAME
,KYLIN_CATEGORY_GROUPINGS.CATEG_LVL3_NAME
,KYLIN_SALES.LSTG_FORMAT_NAME
,SUM(KYLIN_SALES.PRICE)
,COUNT(DISTINCT KYLIN_SALES.SELLER_ID)
FROM KYLIN_SALES as KYLIN_SALES
INNER JOIN KYLIN_CAL_DT as KYLIN_CAL_DT
ON KYLIN_SALES.PART_DT = KYLIN_CAL_DT.CAL_DT
INNER JOIN KYLIN_CATEGORY_GROUPINGS as KYLIN_CATEGORY_GROUPINGS
ON KYLIN_SALES.LEAF_CATEG_ID = KYLIN_CATEGORY_GROUPINGS.LEAF_CATEG_ID AND KYLIN_SALES.L
STG_SITE_ID = KYLIN_CATEGORY_GROUPINGS.SITE_ID
GROUP BY
KYLIN_SALES.PART_DT
,KYLIN_SALES.LEAF_CATEG_ID
,KYLIN_SALES.LSTG_SITE_ID
,KYLIN_CATEGORY_GROUPINGS.META_CATEG_NAME
,KYLIN_CATEGORY_GROUPINGS.CATEG_LVL2_NAME
,KYLIN_CATEGORY_GROUPINGS.CATEG_LVL3_NAME
,KYLIN_SALES.LSTG_FORMAT_NAME
```

这条SQL将事实表KYLIN_SALES和两张维表根据外键进行了内部连接。在笔者的对比试验中，Hive查询耗时34.361秒，KAP查询耗时0.33秒。

维度列COUNT_DISTINCT

```
SELECT
COUNT(DISTINCT KYLIN_SALES.PART_DT)
FROM KYLIN_SALES as KYLIN_SALES
INNER JOIN KYLIN_CAL_DT as KYLIN_CAL_DT
ON KYLIN_SALES.PART_DT = KYLIN_CAL_DT.CAL_DT
INNER JOIN KYLIN_CATEGORY_GROUPINGS as KYLIN_CATEGORY_GROUPINGS
ON KYLIN_SALES.LEAF_CATEG_ID = KYLIN_CATEGORY_GROUPINGS.LEAF_CATEG_ID AND KYLIN_SALES.L
STG_SITE_ID = KYLIN_CATEGORY_GROUPINGS.SITE_ID
```

这条SQL对PART_DT字段进行了COUNT_DISTINCT查询，但是该字段并没有被添加为COUNT_DISTINCT的度量。这个功能是从1.5.2版本开始支持的，即对于未定义的维度列，可以执行COUNT_DISTINCT的查询。在笔者的对比试验中，Hive查询耗时44.911秒，KAP查询耗时0.12秒。

全表查询

```
SELECT * FROM KYLIN_SALES
```

默认的，KAP并不对原始数据的明细进行保存，因此并不支持形如 `SELECT *` 的SQL查询。但是，用户经常希望通过执行 `SELECT *` 获取部分样例数据；因此KAP对这种SQL会返回不精确的查询结果。如果用户希望KAP支持原始数据的保存和查询，可以在Cube中定义RAW类型的度量。

查询页概览

KAP的分析页面即为查询页面，点击后左边列出来所有可以查询的表，这些表是Cube 构建好以后才会显示出来，右边给出了输入框，在这输入SQL, 下方显示结果。下面是不同标识对应的解释。

- FACT - 事实表
- LOOKUP - 维度表
- D - 维度列
- M - 度量列
- D&M - 该列既是维度又是度量
- PK - 主键
- FK - 外键

SQL查询

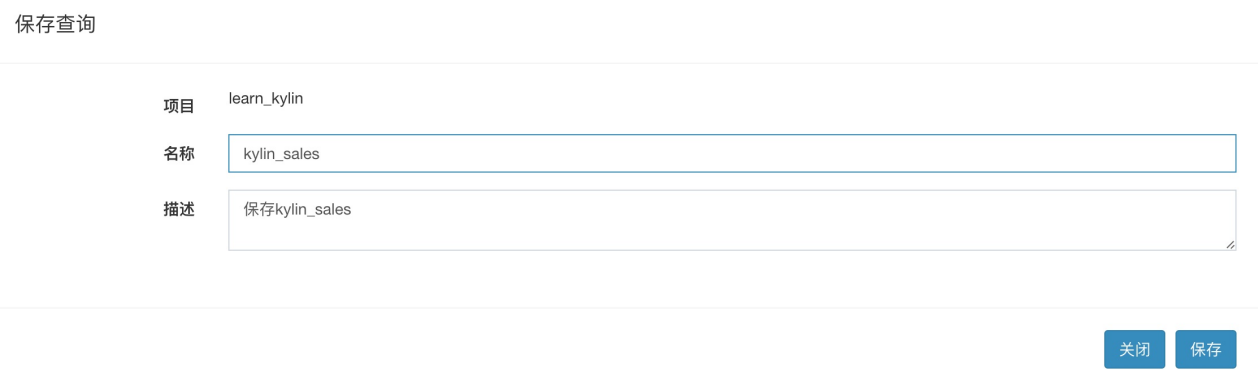
- 提供一个输入框输入SQL, 点击提交即可查询结果。在右下角有一个limit字段，如果SQL中没有limit字段，这里默认会拼接上limit 50000，如果SQL中有limit 那么以SQL中为准。假如用户想去掉limit限制，那么SQL中不要加limit，右下角的limit输入框也改为0。

- 如果SQL去掉了所有的limit限制，后台最多也只会返回一百万条数据，如果超过了一百万条，前台会显示一个 "显示所有记录" 字样的按钮，点击后即可获取所有结果集。当查询运行成功后，它将呈现一个成功指标与被访问的cube名字。同时它将会呈现这个查询在后台引擎运行了多久（不包括从KAP服务器到浏览器的网络通信）

> 支持的浏览器
>
> Windows: Google Chrome, FireFox
>
> Mac: Google Chrome, FireFox, Safari
>
> 查询限制
>
> 1. 仅支持SELECT查询
> 2. 为了避免从服务器到客户端产生巨大的网络流量，扫描范围阀值被设置为1,000,000。
> 3. SQL在cube中无法找到的数据将不会重定向到Hive

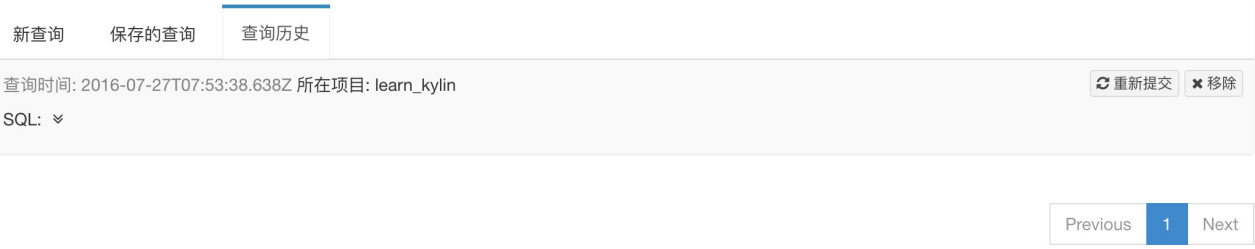
保存查询

与用户账号关联，你将能够从不同的浏览器甚至机器上获取已保存的查询。在结果区域点击保存图标，将会弹出名称和描述来保存当前查询。



查询历史

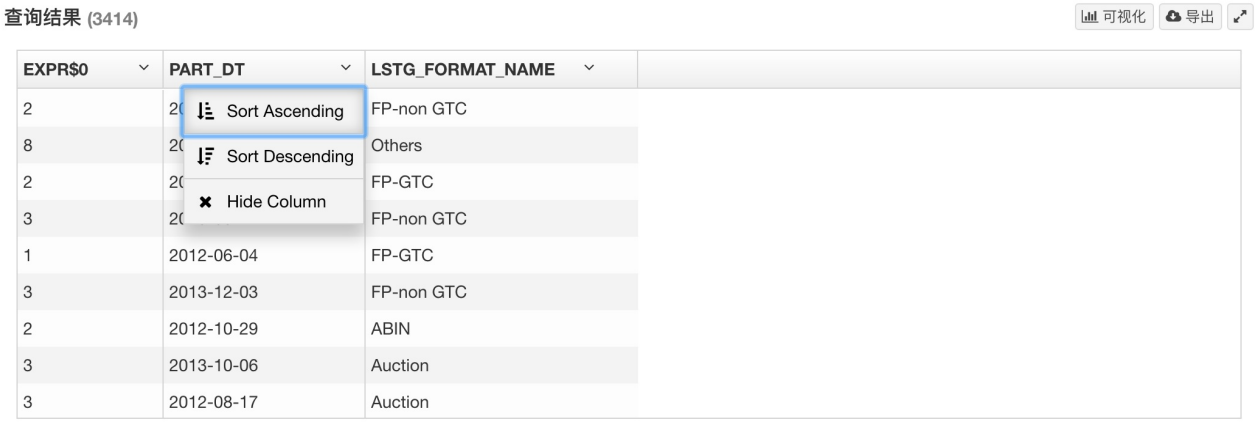
仅保存当前用户在当前浏览器中的查询历史，这将需要启用cookie，并且如果你清理浏览器缓存将会丢失数据。点击“Query History”标签，你可以直接重新提交其中的任何一条并再次运行。



数据展现的方式

表格

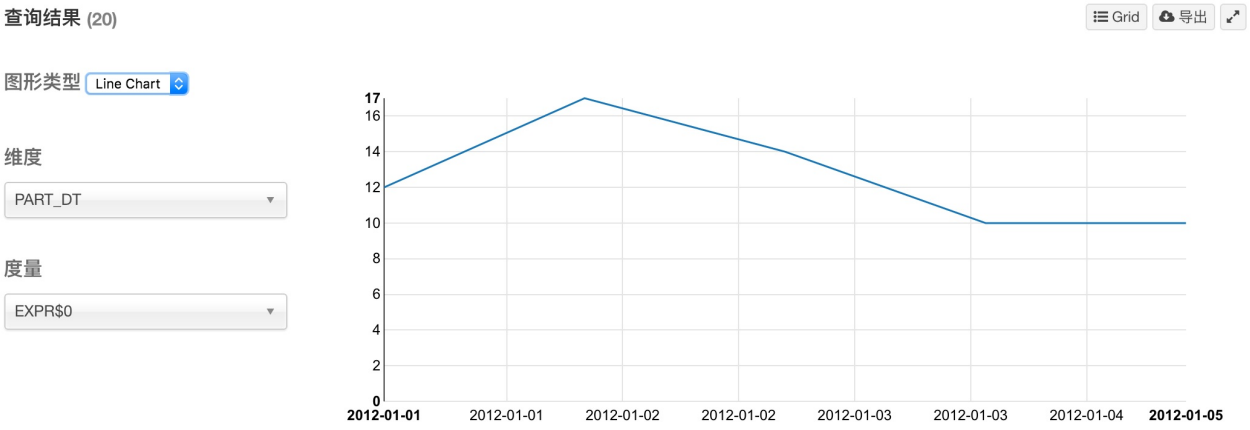
默认情况下，KAP会以表格形式展示数据，可以对数据进行升序和降序排列，也可以对字段进行隐藏。也可以点击'导出'按钮到处csv文件。



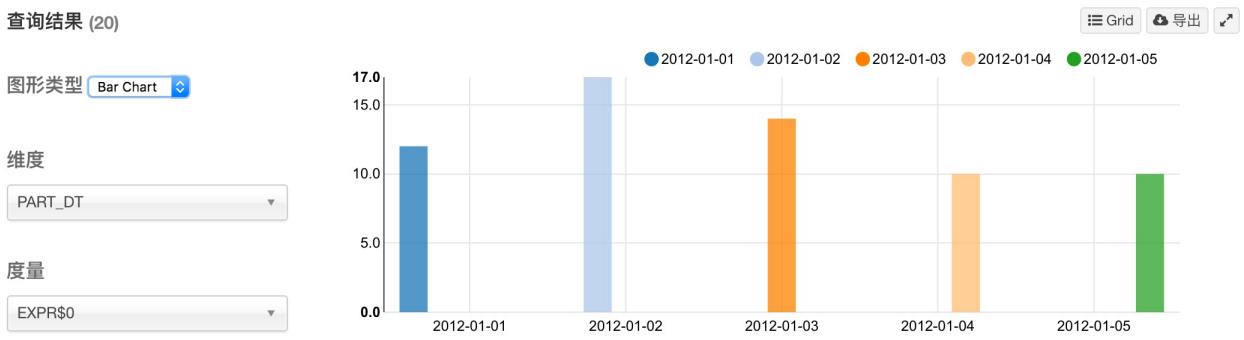
点击'可视化'按钮，可以对数据进行可视化展示，KAP默认支持三种图形展示。分别是线形图，柱状图，饼图。

图表展示

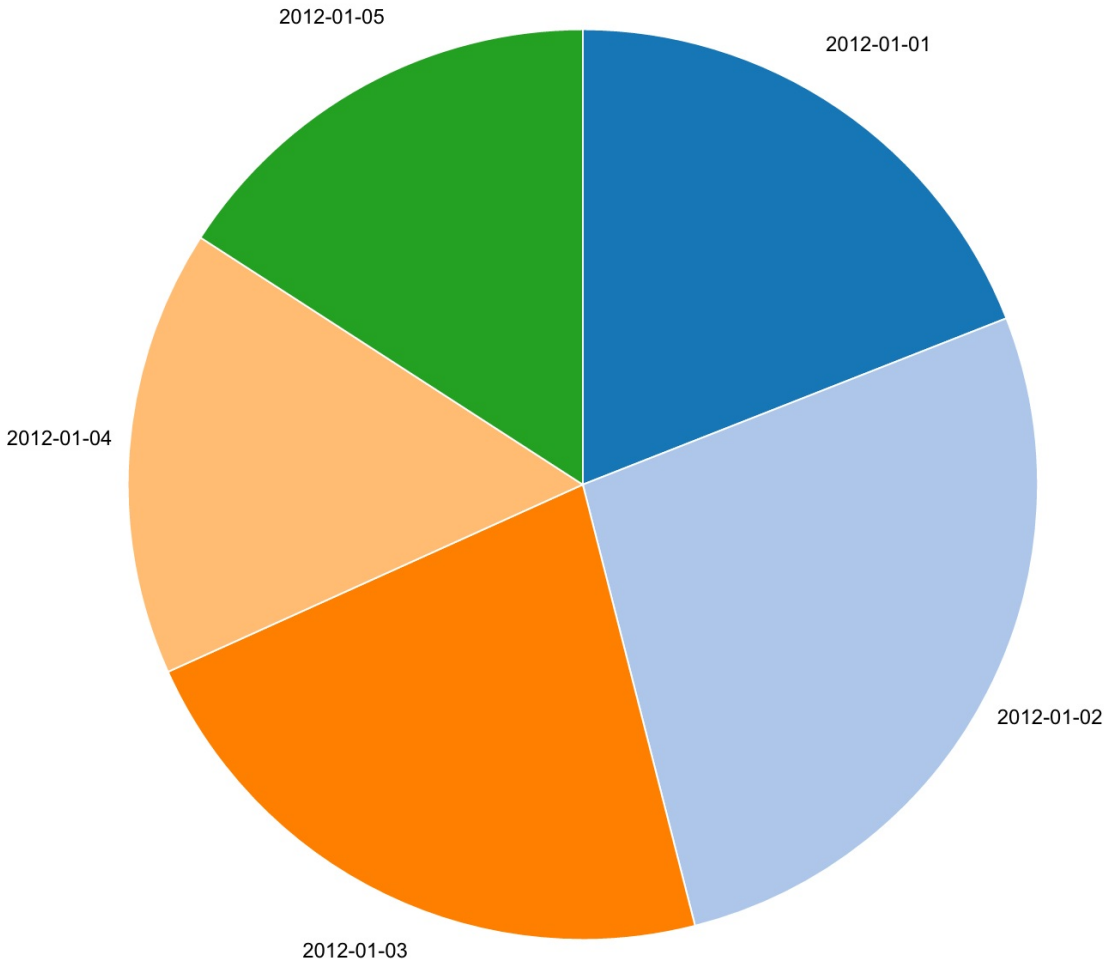
- 线形图展示



- 柱状图展示


查询结果 (20)
Grid
导出
图形类型
Bar Chart
维度
PART_DT
度量
EXPR$0
2012-01-01
2012-01-02
2012-01-03
2012-01-04
2012-01-05
17.0
15.0
10.0
5.0
0.0

- 饼图展示


2012-01-05
2012-01-01
2012-01-04
2012-01-02
2012-01-03

访问与安全认证

访问

KAP API的访问前缀为 ‘/kylin/api‘, 不管对哪个模块的API访问都需要加上该前缀，比如访问所有的cube的API为 “/kylin/api/cubes”，对应完整的路径为 http://host:port/kylin/api/cubes

认证

KAP所有的API都是基于basic authentication认证机制，Basic Authentication是非常简单的一种访问控制机制，将帐号密码基于Base64编码后作为请求头添加到HTTP 请求头中, 后端会读取请求头中的帐号密码信息进行认证。以KAP默认的账户密码ADMIN/KYLIN为例，对应帐号密码编码后结果为“Basic QURNSU46S1lMSU4=“，那么HTTP对应的头信息为“Authorization: Basic QURNSU46S1lMSU4=”。

认证要点

- 在HTTP头添加 `Authorization` 信息
- 或者可以通过 `POST http://localhost:7070/kylin/api/user/authentication` 进行认证，一旦认证通过，接下来对API请求基于cookies在HTTP头中免去 `Authorization` 信息

```
POST http://localhost:7070/kylin/api/user/authentication
Authorization:Basic xxxxJD124xxxGFxxxSDF
Content-Type: application/json;charset=UTF-8
```

这里以javascript和curl为例介绍在访问API时如何添加认证信息。

Query API示例

```
$.ajaxSetup({
      headers: { 'Authorization': "Basic eWFu**********X***ZA==", 'Content-Type': 'appl
ication/json;charset=utf-8' } // use your own authorization code here
    });
    var request = $.ajax({
       url: "http://hostname/kylin/api/query",
       type: "POST",
       data: '{"sql":"select count(*) from SUMMARY;","offset":0,"limit":50000,"acceptPa
rtial":true,"project":"test"}',
       dataType: "json"
    });
    request.done(function( msg ) {
       alert(msg);
    });
    request.fail(function( jqXHR, textStatus ) {
       alert( "Request failed: " + textStatus );
  });
```

Javascript如何生成authorization信息(下载"jquery.base64.js" https://github.com/yckart/jquery.base64.js).

```
var authorizationCode = $.base64('encode', 'NT_USERNAME' + ":" + 'NT_PASSWORD');

$.ajaxSetup({
   headers: {
    'Authorization': "Basic " + authorizationCode,
    'Content-Type': 'application/json;charset=utf-8'
   }
});
```

Curl 示例

```
curl -c /path/to/cookiefile.txt -X POST -H "Authorization: Basic XXXXXXXXX" -H 'Content
-Type: application/json' http://<host>:<port>/kylin/api/user/authentication
```

访问成功后JSESSIONID会存在cookie文件中， 在接下来的访问中添加上cookie信息:

```
curl -b /path/to/cookiefile.txt -X PUT -H 'Content-Type: application/json' -d '{"startT
ime":'1423526400000', "endTime":'1423526400', "buildType":"BUILD"}' http://<host>:<port
>/kylin/api/cubes/your_cube/rebuild
```

查询 REST API

> 提示
>
> 使用API前请确保已阅读前面的 访问及安全验证 章节，知道如何在API中添加认证信息

访问KAP构建的数据集，主要两个API，一个是直接查询Cube数据，一个是列出所有可以查询的表。

- 查询
  - 查询Cube数据
  - 列出可查询的表

查询Cube数据

`请求方式 POST`

`访问路径 http://host:port/kylin/api/query`

请求主体

- sql - `必选` `string` 查询的sql.
- offset - `可选` `int` 查询默认从第一行返回结果，可以设置改参数设置返回数据从哪一行开始往后返回
- limit - `可选` `int` • 加上limit参数后会从offset开始返回对应的行数，不足limt以实际行数为准
- acceptPartial - `可选` `bool` 默认是“true”，如果为true，那么当前实际最多会返回一百万行数据，如果要返回超过一百万行的结果集，那该参数需要设置为“false”
- project - `可选` `string` 默认为 ‘DEFAULT’，在实际使用时，如果对应查询的项目不是“DEFAULT”，需要设置为自己的项目

请求示例

```
{
   "sql":"select * from TEST_KYLIN_FACT",
   "offset":0,
   "limit":50000,
   "acceptPartial":false,
   "project":"DEFAULT"
}
```

响应信息

- columnMetas - 每个列的元数据信息.
- results - 返回的结果集.
- cube - 这个查询对应使用的CUBE.
- affectedRowCount - 这个查询关系到的总行树.
- isException - 这个查询返回是否是异常.
- ExceptionMessage - 返回异常对应的内容.
- Duration - 查询消耗时间

- Partial - 这个查询结果是否为部分返回，这个取决于请求参数中的 acceptPartial 为true或者false.

响应示例


```
{
   "columnMetas":[
      {
         "isNullable":1,
         "displaySize":0,
         "label":"CAL_DT",
         "name":"CAL_DT",
         "schemaName":null,
         "catelogName":null,
         "tableName":null,
         "precision":0,
         "scale":0,
         "columnType":91,
         "columnTypeName":"DATE",
         "readOnly":true,
         "writable":false,
         "caseSensitive":true,
         "searchable":false,
         "currency":false,
         "signed":true,
         "autoIncrement":false,
         "definitelyWritable":false
      },
      {
         "isNullable":1,
         "displaySize":10,
         "label":"LEAF_CATEG_ID",
         "name":"LEAF_CATEG_ID",
         "schemaName":null,
         "catelogName":null,
         "tableName":null,
         "precision":10,
         "scale":0,
         "columnType":4,
         "columnTypeName":"INTEGER",
         "readOnly":true,
         "writable":false,
         "caseSensitive":true,
         "searchable":false,
         "currency":false,
         "signed":true,
         "autoIncrement":false,
         "definitelyWritable":false
      }
   ],
   "results":[
      [
         "2013-08-07",
```

```
            "32996",
            "15",
            "15",
            "Auction",
            "10000000",
            "49.048952730908745",
            "49.048952730908745",
            "49.048952730908745",
            "1"
        ],
        [
            "2013-08-07",
            "43398",
            "0",
            "14",
            "ABIN",
            "10000633",
            "85.78317064220418",
            "85.78317064220418",
            "85.78317064220418",
            "1"
        ]
    ],
    "cube":"test_kylin_cube_with_slr_desc",
    "affectedRowCount":0,
    "isException":false,
    "exceptionMessage":null,
    "duration":3451,
    "partial":false
}
```

Curl 访问示例

```
curl -X POST -H "Authorization: Basic XXXXXXXXX" -H "Content-Type: application/json" -d
 '{ "sql":"select count(*) from TEST_KYLIN_FACT", "project":"learn_kylin" }' http://YOU
R_HOST:7070/kylin/api/query
```

列出可查询的表

`请求方式 GET`

`访问路径 http://host:port/kylin/api/tables_and_columns`

请求参数

- project - `必选` `string` 说明对应要列出哪个Project下的表

响应示例

```
[
    {
```

```
        "columns":[
           {
              "table_NAME":"TEST_CAL_DT",
              "table_SCHEM":"EDW",
              "column_NAME":"CAL_DT",
              "data_TYPE":91,
              "nullable":1,
              "column_SIZE":-1,
              "buffer_LENGTH":-1,
              "decimal_DIGITS":0,
              "num_PREC_RADIX":10,
              "column_DEF":null,
              "sql_DATA_TYPE":-1,
              "sql_DATETIME_SUB":-1,
              "char_OCTET_LENGTH":-1,
              "ordinal_POSITION":1,
              "is_NULLABLE":"YES",
              "scope_CATLOG":null,
              "scope_SCHEMA":null,
              "scope_TABLE":null,
              "source_DATA_TYPE":-1,
              "iS_AUTOINCREMENT":null,
              "table_CAT":"defaultCatalog",
              "remarks":null,
              "type_NAME":"DATE"
           },
           {
              "table_NAME":"TEST_CAL_DT",
              "table_SCHEM":"EDW",
              "column_NAME":"WEEK_BEG_DT",
              "data_TYPE":91,
              "nullable":1,
              "column_SIZE":-1,
              "buffer_LENGTH":-1,
              "decimal_DIGITS":0,
              "num_PREC_RADIX":10,
              "column_DEF":null,
              "sql_DATA_TYPE":-1,
              "sql_DATETIME_SUB":-1,
              "char_OCTET_LENGTH":-1,
              "ordinal_POSITION":2,
              "is_NULLABLE":"YES",
              "scope_CATLOG":null,
              "scope_SCHEMA":null,
              "scope_TABLE":null,
              "source_DATA_TYPE":-1,
              "iS_AUTOINCREMENT":null,
              "table_CAT":"defaultCatalog",
              "remarks":null,
              "type_NAME":"DATE"
           }
        ],
        "table_NAME":"TEST_CAL_DT",
```

```
        "table_SCHEM":"EDW",
        "ref_GENERATION":null,
        "self_REFERENCING_COL_NAME":null,
        "type_SCHEM":null,
        "table_TYPE":"TABLE",
        "table_CAT":"defaultCatalog",
        "remarks":null,
        "type_CAT":null,
        "type_NAME":null
    }
]
```

模型 REST API

> 提示
>
> 使用API前请确保已阅读前面的 访问及安全验证 章节，知道如何在API中添加认证信息

- 返回多个模型
- 返回指定模型
- 克隆模型
- 删除模型

返回多个模型

`请求方式 GET`

`访问路径 http://host:port/kylin/api/models`

请求主体

- offset - `必选` `int` 返回数据起始下标
- limit - `必选` `int` 分页返回对应每页返回多少
- modelName - `可选` `string` 返回名称等于该关键字的Model
- projectName - `可选` `string` 指定返回该项目下

响应示例

```
[
  {

    "uuid" : "ff527b94-f860-44c3-8452-93b17774c647",
    "name" : "test_kylin_inner_join_model_desc",
    "lookups" : [ {
      "table" : "EDW.TEST_CAL_DT",
      "join" : {
        "type" : "inner",
        "primary_key" : [ "CAL_DT" ],
        "foreign_key" : [ "CAL_DT" ]
      }
    }, {
      "table" : "DEFAULT.TEST_CATEGORY_GROUPINGS",
      "join" : {
        "type" : "inner",
        "primary_key" : [ "LEAF_CATEG_ID", "SITE_ID" ],
        "foreign_key" : [ "LEAF_CATEG_ID", "LSTG_SITE_ID" ]
      }
    }, {
      "table" : "EDW.TEST_SITES",
      "join" : {
        "type" : "inner",
        "primary_key" : [ "SITE_ID" ],
```

```
        "foreign_key" : [ "LSTG_SITE_ID" ]
      }
    }, {
      "table" : "EDW.TEST_SELLER_TYPE_DIM",
      "join" : {
        "type" : "inner",
        "primary_key" : [ "SELLER_TYPE_CD" ],
        "foreign_key" : [ "SLR_SEGMENT_CD" ]
      }
    } ],
    "dimensions": [
      {
        "table": "default.test_kylin_fact",
        "columns": [
          "lstg_format_name",
          "LSTG_SITE_ID",
          "SLR_SEGMENT_CD",
          "TRANS_ID",
          "CAL_DT",
          "LEAF_CATEG_ID",
          "SELLER_ID"
        ]
      },
      {
        "table": "default.test_category_groupings",
        "columns": [
          "leaf_categ_id",
          "site_id",
          "USER_DEFINED_FIELD1",
          "USER_DEFINED_FIELD3",
          "UPD_DATE",
          "UPD_USER",
          "meta_categ_name",
          "categ_lvl2_name",
          "categ_lvl3_name"
        ]
      },
      {
        "table": "edw.test_sites",
        "columns": [
          "site_id",
          "site_name",
          "cre_user"
        ]
      },
      {
        "table": "edw.test_seller_type_dim",
        "columns": [
          "seller_type_cd",
          "seller_type_desc"
        ]
      },
      {
```

```
        "table": "edw.test_cal_dt",
        "columns": [
          "cal_dt",
          "week_beg_dt"
        ]
      }
    ],
    "metrics": [
    "PRICE",
    "ITEM_COUNT",
    "SELLER_ID"
    ],
    "last_modified" : 1422435345352,
    "fact_table" : "DEFAULT.TEST_KYLIN_FACT",
    "filter_condition" : null,
    "partition_desc" : {
      "partition_date_column" : "DEFAULT.TEST_KYLIN_FACT.cal_dt",
      "partition_date_start" : 0,
      "partition_type" : "APPEND"
    }
  }
  ]
```

返回指定模型

`请求方式 GET`

`访问路径 http://host:port/kylin/api/model/{modelName}`

路径变量

- modelName - `必选` `string` 要获取的model 名称.

克隆模型

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/models/{modelName}/clone`

路径变量

- modelName - `必选` `string` 被克隆模型名称.

请求主体

- modelName - `必选` `string` 新模型名称.
- project - `必选` `string` 新项目名称

响应示例

```
  {
    message:null,
    modelDescData:null,
    modelName:"kylin_sales_model_clone",
    project:"learn_kylin",
    successful:true,
    uuid:"c9d164f9-3e1a-425f-b41e-591223e6e91a"
}
```

删除数据模型

`请求方式 DELETE`

`访问路径 http://host:port/kylin/api/model/{modelName}`

路径变量

- modelName - `必选` `string` 数据模型名称.

Cube REST API

> 提示
>
> 使用API前请确保已阅读前面的 访问及安全验证 章节，知道如何在API中添加认证信息

- 返回多个Cube
- 返回指定cube
- 返回Cube描述信息(维度, 度量，等)
- 返回数据模型 (事实表及维度表等信息)
- 构建 cube
- 克隆 cube
- 禁用 cube
- 清理 cube
- 启用 cube

返回多个Cube

`请求方式 GET`

`访问路径 http://host:port/kylin/api/cubes`

请求主体

- offset - `必选` `int` 返回数据起始下标
- limit - `必选` `int` 分页返回对应每页返回多少
- cubeName - `可选` `string` 返回名称等于该关键字的Cube
- modelName - `可选` `string` 返回对应模型名称等于该关键字的Cube
- projectName - `可选` `string` 指定返回该项目下Cube

响应示例

```
[
   {
      "uuid":"1eaca32a-a33e-4b69-83dd-0bb8b1f8c53b",
      "last_modified":1407831634847,
      "name":"test_kylin_cube_with_slr_empty",
      "owner":null,
      "version":null,
      "descriptor":"test_kylin_cube_with_slr_desc",
      "cost":50,
      "status":"DISABLED",
      "segments":[
      ],
      "create_time":null,
      "source_records_count":0,
      "source_records_size":0,
      "size_kb":0
   }
]
```

返回指定cube

`请求方式 GET`

`访问路径 http://host:port/kylin/api/cubes/{cubeName}`

路径变量

- cubeName - `必选` `string` 要获取的Cube 名称.

返回Cube描述信息(维度, 度量，等)

`请求方式 GET`

`访问路径 http://host:port/kylin/api/cube_desc/{cubeName}/desc`

路径变量

- cubeName - `必选` `string` Cube 名称.

响应示例

```
    {
  "uuid" : "a24ca905-1fc6-4f67-985c-38fa5aeafd92",

  "name" : "test_kylin_cube_with_slr_desc",
  "description" : null,
  "dimensions" : [ {
    "name" : "CAL_DT",
    "table" : "EDW.TEST_CAL_DT",
    "column" : "{FK}",
    "derived" : [ "WEEK_BEG_DT" ]
```

```
    }, {
      "name" : "CATEGORY",
      "table" : "DEFAULT.TEST_CATEGORY_GROUPINGS",
      "column" : "{FK}",
      "derived" : [ "USER_DEFINED_FIELD1", "USER_DEFINED_FIELD3", "UPD_DATE", "UPD_USER"
  ]
    }, {
      "name" : "CATEGORY_HIERARCHY",
      "table" : "DEFAULT.TEST_CATEGORY_GROUPINGS",
      "column" : "META_CATEG_NAME",
      "derived" : null
    }, {
      "name" : "CATEGORY_HIERARCHY",
      "table" : "DEFAULT.TEST_CATEGORY_GROUPINGS",
      "column" : "CATEG_LVL2_NAME",
      "derived" : null
    }, {
      "name" : "CATEGORY_HIERARCHY",
      "table" : "DEFAULT.TEST_CATEGORY_GROUPINGS",
      "column" : "CATEG_LVL3_NAME",
      "derived" : null
    }, {
      "name" : "LSTG_FORMAT_NAME",
      "table" : "DEFAULT.TEST_KYLIN_FACT",
      "column" : "LSTG_FORMAT_NAME",
      "derived" : null
    }, {
      "name" : "SITE_ID",
      "table" : "EDW.TEST_SITES",
      "column" : "{FK}",
      "derived" : [ "SITE_NAME", "CRE_USER" ]
    }, {
      "name" : "SELLER_TYPE_CD",
      "table" : "EDW.TEST_SELLER_TYPE_DIM",
      "column" : "{FK}",
      "derived" : [ "SELLER_TYPE_DESC" ]
    }, {
      "name" : "SELLER_ID",
      "table" : "DEFAULT.TEST_KYLIN_FACT",
      "column" : "SELLER_ID",
      "derived" : null
    } ],
    "measures" : [ {
      "name" : "GMV_SUM",
      "function" : {
        "expression" : "SUM",
        "parameter" : {
          "type" : "column",
          "value" : "PRICE",
          "next_parameter" : null
        },
        "returntype" : "decimal(19,4)"
      },
```

```
      "dependent_measure_ref" : null
    }, {
      "name" : "GMV_MIN",
      "function" : {
        "expression" : "MIN",
        "parameter" : {
          "type" : "column",
          "value" : "PRICE",
          "next_parameter" : null
        },
        "returntype" : "decimal(19,4)"
      },
      "dependent_measure_ref" : null
    }, {
      "name" : "GMV_MAX",
      "function" : {
        "expression" : "MAX",
        "parameter" : {
          "type" : "column",
          "value" : "PRICE",
          "next_parameter" : null
        },
        "returntype" : "decimal(19,4)"
      },
      "dependent_measure_ref" : null
    }, {
      "name" : "TRANS_CNT",
      "function" : {
        "expression" : "COUNT",
        "parameter" : {
          "type" : "constant",
          "value" : "1",
          "next_parameter" : null
        },
        "returntype" : "bigint"
      },
      "dependent_measure_ref" : null
    }, {
      "name" : "ITEM_COUNT_SUM",
      "function" : {
        "expression" : "SUM",
        "parameter" : {
          "type" : "column",
          "value" : "ITEM_COUNT",
          "next_parameter" : null
        },
        "returntype" : "bigint"
      },
      "dependent_measure_ref" : null
    } ],
    "rowkey" : {
      "rowkey_columns" : [ {
        "column" : "seller_id",
```

```
      "encoding" : "int:4",
      "isShardBy" : true
    }, {
      "column" : "cal_dt",
      "encoding" : "dict"
    }, {
      "column" : "leaf_categ_id",
      "encoding" : "fixed_length:18"
    }, {
      "column" : "meta_categ_name",
      "encoding" : "dict"
    }, {
      "column" : "categ_lvl2_name",
      "encoding" : "dict"
    }, {
      "column" : "categ_lvl3_name",
      "encoding" : "dict"
    }, {
      "column" : "lstg_format_name",
      "encoding" : "fixed_length:12"
    }, {
      "column" : "lstg_site_id",
      "encoding" : "dict"
    }, {
      "column" : "slr_segment_cd",
      "encoding" : "dict"
    } ]
  },
  "signature" : null,
  "last_modified" : 1448959801271,
  "model_name" : "test_kylin_inner_join_model_desc",
  "null_string" : null,
  "hbase_mapping" : {
    "column_family" : [ {
      "name" : "f1",
      "columns" : [ {
        "qualifier" : "m",
        "measure_refs" : [ "gmv_sum", "gmv_min", "gmv_max", "trans_cnt", "item_count_su
m" ]
      } ]
    } ]
  },
  "aggregation_groups" : [ {
    "includes" : [ "cal_dt", "categ_lvl2_name", "categ_lvl3_name", "leaf_categ_id", "ls
tg_format_name", "lstg_site_id", "meta_categ_name", "seller_id", "slr_segment_cd" ],
    "select_rule" : {
      "hierarchy_dims" : [ [ "META_CATEG_NAME", "CATEG_LVL2_NAME", "CATEG_LVL3_NAME" ]
],
      "mandatory_dims" : ["seller_id"],
      "joint_dims" : [ [ "lstg_format_name", "lstg_site_id", "slr_segment_cd" ] ]
    }
  } ],
  "notify_list" : null,
```

```
  "status_need_notify" : [ ],
  "auto_merge_time_ranges" : null,
  "retention_range" : 0,
  "engine_type" : 2,
  "storage_type" : 2,
  "partition_date_start" : 0
}
```

返回数据模型 (事实表及维度表等信息)

`请求方式 GET`

`访问路径 http://host:port/kylin/api/model/{modelName}`

路径变量

- modelName - `必选` `string` 数据模型名称.

响应示例

```
{

  "uuid" : "ff527b94-f860-44c3-8452-93b17774c647",
  "name" : "test_kylin_inner_join_model_desc",
  "lookups" : [ {
    "table" : "EDW.TEST_CAL_DT",
    "join" : {
      "type" : "inner",
      "primary_key" : [ "CAL_DT" ],
      "foreign_key" : [ "CAL_DT" ]
    }
  }, {
    "table" : "DEFAULT.TEST_CATEGORY_GROUPINGS",
    "join" : {
      "type" : "inner",
      "primary_key" : [ "LEAF_CATEG_ID", "SITE_ID" ],
      "foreign_key" : [ "LEAF_CATEG_ID", "LSTG_SITE_ID" ]
    }
  }, {
    "table" : "EDW.TEST_SITES",
    "join" : {
      "type" : "inner",
      "primary_key" : [ "SITE_ID" ],
      "foreign_key" : [ "LSTG_SITE_ID" ]
    }
  }, {
    "table" : "EDW.TEST_SELLER_TYPE_DIM",
    "join" : {
      "type" : "inner",
      "primary_key" : [ "SELLER_TYPE_CD" ],
      "foreign_key" : [ "SLR_SEGMENT_CD" ]
    }
```

```
  } ],
  "dimensions": [
    {
      "table": "default.test_kylin_fact",
      "columns": [
        "lstg_format_name",
        "LSTG_SITE_ID",
        "SLR_SEGMENT_CD",
        "TRANS_ID",
        "CAL_DT",
        "LEAF_CATEG_ID",
        "SELLER_ID"
      ]
    },
    {
      "table": "default.test_category_groupings",
      "columns": [
        "leaf_categ_id",
        "site_id",
        "USER_DEFINED_FIELD1",
        "USER_DEFINED_FIELD3",
        "UPD_DATE",
        "UPD_USER",
        "meta_categ_name",
        "categ_lvl2_name",
        "categ_lvl3_name"
      ]
    },
    {
      "table": "edw.test_sites",
      "columns": [
        "site_id",
        "site_name",
        "cre_user"
      ]
    },
    {
      "table": "edw.test_seller_type_dim",
      "columns": [
        "seller_type_cd",
        "seller_type_desc"
      ]
    },
    {
      "table": "edw.test_cal_dt",
      "columns": [
        "cal_dt",
        "week_beg_dt"
      ]
    }
  ],
  "metrics": [
  "PRICE",
```

```
    "ITEM_COUNT",
    "SELLER_ID"
    ],
    "last_modified" : 1422435345352,
    "fact_table" : "DEFAULT.TEST_KYLIN_FACT",
    "filter_condition" : null,
    "partition_desc" : {
      "partition_date_column" : "DEFAULT.TEST_KYLIN_FACT.cal_dt",
      "partition_date_start" : 0,
      "partition_type" : "APPEND"
    }
  }
```

构建 cube

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/cubes/{cubeName}/rebuild`

路径变量

- cubeName - `必选` `string` Cube 名称.

请求主体

- startTime - `必选` `long` 要计算的数据对应起始时间对应的timestamp，应为GMT0格式的 timestamp , e.g. 1388563200000 for 2014-1-1
- endTime - `必选` `long` 要计算的数据对应起始时间对应的timestamp，应为GMT0格式的 timestamp
- buildType - `必选` `string` 支持的计算类型: 'BUILD', 'MERGE', 'REFRESH'

响应示例

```
{
   "uuid":"c143e0e4-ac5f-434d-acf3-46b0d15e3dc6",
   "last_modified":1407908916705,
   "name":"test_kylin_cube_with_slr_empty - 19700101000000_20140731160000 - BUILD - PDT
 2014-08-12 22:48:36",
   "type":"BUILD",
   "duration":0,
   "related_cube":"test_kylin_cube_with_slr_empty",
   "related_segment":"19700101000000_20140731160000",
   "exec_start_time":0,
   "exec_end_time":0,
   "mr_waiting":0,
   "steps":[
      {
         "interruptCmd":null,
         "name":"Create Intermediate Flat Hive Table",
         "sequence_id":0,
         "exec_cmd":"hive -e \"DROP TABLE IF EXISTS kylin_intermediate_test_kylin_cube_
with_slr_desc_19700101000000_20140731160000_c143e0e4_ac5f_434d_acf3_46b0d15e3dc6;\nCREA
```

```
TE EXTERNAL TABLE IF NOT EXISTS kylin_intermediate_test_kylin_cube_with_slr_desc_197001
01000000_20140731160000_c143e0e4_ac5f_434d_acf3_46b0d15e3dc6\n(\nCAL_DT date\n,LEAF_CAT
EG_ID int\n,LSTG_SITE_ID int\n,META_CATEG_NAME string\n,CATEG_LVL2_NAME string\n,CATEG_
LVL3_NAME string\n,LSTG_FORMAT_NAME string\n,SLR_SEGMENT_CD smallint\n,SELLER_ID bigint
\n,PRICE decimal\n)\nROW FORMAT DELIMITED FIELDS TERMINATED BY '\\177'\nSTORED AS SEQUE
NCEFILE\nLOCATION '/tmp/kylin-c143e0e4-ac5f-434d-acf3-46b0d15e3dc6/kylin_intermediate_t
est_kylin_cube_with_slr_desc_19700101000000_20140731160000_c143e0e4_ac5f_434d_acf3_46b0
d15e3dc6';\nSET mapreduce.job.split.metainfo.maxsize=-1;\nSET mapred.compress.map.outpu
t=true;\nSET mapred.map.output.compression.codec=com.hadoop.compression.lzo.LzoCodec;\n
SET mapred.output.compress=true;\nSET mapred.output.compression.codec=com.hadoop.compre
ssion.lzo.LzoCodec;\nSET mapred.output.compression.type=BLOCK;\nSET mapreduce.job.max.s
plit.locations=2000;\nSET hive.exec.compress.output=true;\nSET hive.auto.convert.join.n
oconditionaltask = true;\nSET hive.auto.convert.join.noconditionaltask.size = 300000000
;\nINSERT OVERWRITE TABLE kylin_intermediate_test_kylin_cube_with_slr_desc_197001010000
00_20140731160000_c143e0e4_ac5f_434d_acf3_46b0d15e3dc6\nSELECT\nTEST_KYLIN_FACT.CAL_DT\
n,TEST_KYLIN_FACT.LEAF_CATEG_ID\n,TEST_KYLIN_FACT.LSTG_SITE_ID\n,TEST_CATEGORY_GROUPING
S.META_CATEG_NAME\n,TEST_CATEGORY_GROUPINGS.CATEG_LVL2_NAME\n,TEST_CATEGORY_GROUPINGS.C
ATEG_LVL3_NAME\n,TEST_KYLIN_FACT.LSTG_FORMAT_NAME\n,TEST_KYLIN_FACT.SLR_SEGMENT_CD\n,TE
ST_KYLIN_FACT.SELLER_ID\n,TEST_KYLIN_FACT.PRICE\nFROM TEST_KYLIN_FACT\nINNER JOIN TEST_
CAL_DT\nON TEST_KYLIN_FACT.CAL_DT = TEST_CAL_DT.CAL_DT\nINNER JOIN TEST_CATEGORY_GROUPI
NGS\nON TEST_KYLIN_FACT.LEAF_CATEG_ID = TEST_CATEGORY_GROUPINGS.LEAF_CATEG_ID AND TEST_
KYLIN_FACT.LSTG_SITE_ID = TEST_CATEGORY_GROUPINGS.SITE_ID\nINNER JOIN TEST_SITES\nON TE
ST_KYLIN_FACT.LSTG_SITE_ID = TEST_SITES.SITE_ID\nINNER JOIN TEST_SELLER_TYPE_DIM\nON TE
ST_KYLIN_FACT.SLR_SEGMENT_CD = TEST_SELLER_TYPE_DIM.SELLER_TYPE_CD\nWHERE (test_kylin_f
act.cal_dt < '2014-07-31 16:00:00')\n;\n\"",
         "interrupt_cmd":null,
         "exec_start_time":0,
         "exec_end_time":0,
         "exec_wait_time":0,
         "step_status":"PENDING",
         "cmd_type":"SHELL_CMD_HADOOP",
         "info":null,
         "run_async":false
      },
      {
         "interruptCmd":null,
         "name":"Extract Fact Table Distinct Columns",
         "sequence_id":1,
         "exec_cmd":" -conf C:/kylin/Kylin/server/src/main/resources/hadoop_job_conf_me
dium.xml -cubename test_kylin_cube_with_slr_empty -input /tmp/kylin-c143e0e4-ac5f-434d-
acf3-46b0d15e3dc6/kylin_intermediate_test_kylin_cube_with_slr_desc_19700101000000_20140
731160000_c143e0e4_ac5f_434d_acf3_46b0d15e3dc6 -output /tmp/kylin-c143e0e4-ac5f-434d-ac
f3-46b0d15e3dc6/test_kylin_cube_with_slr_empty/fact_distinct_columns -jobname Kylin_Fac
t_Distinct_Columns_test_kylin_cube_with_slr_empty_Step_1",
         "interrupt_cmd":null,
         "exec_start_time":0,
         "exec_end_time":0,
         "exec_wait_time":0,
         "step_status":"PENDING",
         "cmd_type":"JAVA_CMD_HADOOP_FACTDISTINCT",
         "info":null,
         "run_async":true
      },
```

```
        {
           "interruptCmd":null,
           "name":"Load HFile to HBase Table",
           "sequence_id":12,
           "exec_cmd":" -input /tmp/kylin-c143e0e4-ac5f-434d-acf3-46b0d15e3dc6/test_kylin
_cube_with_slr_empty/hfile/ -htablename KYLIN-CUBE-TEST_KYLIN_CUBE_WITH_SLR_EMPTY-19700
101000000_20140731160000_11BB4326-5975-4358-804C-70D53642E03A -cubename test_kylin_cube
_with_slr_empty",
           "interrupt_cmd":null,
           "exec_start_time":0,
           "exec_end_time":0,
           "exec_wait_time":0,
           "step_status":"PENDING",
           "cmd_type":"JAVA_CMD_HADOOP_NO_MR_BULKLOAD",
           "info":null,
           "run_async":false
        }
    ],
    "job_status":"PENDING",
    "progress":0.0
}
```

克隆Cube

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/cubes/{cubeName}/clone`

路径变量

- cubeName - `必选` `string` 被克隆Cube名称.

请求主体

- cubeName - `必选` `string` 新Cube名称.
- project - `必选` `string` 新项目名称

响应示例

(同 "启用 Cube")

启用 Cube

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/cubes/{cubeName}/enable`

路径变量

- cubeName - `必选` `string` Cube 名称.

响应示例

```
{
   "uuid":"1eaca32a-a33e-4b69-83dd-0bb8b1f8c53b",
   "last_modified":1407909046305,
   "name":"test_kylin_cube_with_slr_ready",
   "owner":null,
   "version":null,
   "descriptor":"test_kylin_cube_with_slr_desc",
   "cost":50,
   "status":"ACTIVE",
   "segments":[
      {
         "name":"19700101000000_20140531160000",
         "storage_location_identifier":"KYLIN-CUBE-TEST_KYLIN_CUBE_WITH_SLR_READY-19700
101000000_20140531160000_BF043D2D-9A4A-45E9-AA59-5A17D3F34A50",
         "date_range_start":0,
         "date_range_end":1401552000000,
         "status":"READY",
         "size_kb":4758,
         "source_records":6000,
         "source_records_size":620356,
         "last_build_time":1407832663227,
         "last_build_job_id":"2c7a2b63-b052-4a51-8b09-0c24b5792cda",
         "binary_signature":null,
         "dictionaries":{
            "TEST_CATEGORY_GROUPINGS/CATEG_LVL2_NAME":"/dict/TEST_CATEGORY_GROUPINGS/CA
TEG_LVL2_NAME/16d8185c-ee6b-4f8c-a919-756d9809f937.dict",
            "TEST_KYLIN_FACT/LSTG_SITE_ID":"/dict/TEST_SITES/SITE_ID/0bec6bb3-1b0d-469c
-8289-b8c4ca5d5001.dict",
            "TEST_KYLIN_FACT/SLR_SEGMENT_CD":"/dict/TEST_SELLER_TYPE_DIM/SELLER_TYPE_CD
/0c5d77ec-316b-47e0-ba9a-0616be890ad6.dict",
            "TEST_KYLIN_FACT/CAL_DT":"/dict/PREDEFINED/date(yyyy-mm-dd)/64ac4f82-f2af-4
76e-85b9-f0805001014e.dict",
            "TEST_CATEGORY_GROUPINGS/CATEG_LVL3_NAME":"/dict/TEST_CATEGORY_GROUPINGS/CA
TEG_LVL3_NAME/270fbfb0-281c-4602-8413-2970a7439c47.dict",
            "TEST_KYLIN_FACT/LEAF_CATEG_ID":"/dict/TEST_CATEGORY_GROUPINGS/LEAF_CATEG_I
D/2602386c-debb-4968-8d2f-b52b8215e385.dict",
            "TEST_CATEGORY_GROUPINGS/META_CATEG_NAME":"/dict/TEST_CATEGORY_GROUPINGS/ME
TA_CATEG_NAME/0410d2c4-4686-40bc-ba14-170042a2de94.dict"
         },
         "snapshots":{
            "TEST_CAL_DT":"/table_snapshot/TEST_CAL_DT.csv/8f7cfc8a-020d-4019-b419-3c6d
eb0ffaa0.snapshot",
            "TEST_SELLER_TYPE_DIM":"/table_snapshot/TEST_SELLER_TYPE_DIM.csv/c60fd05e-a
c94-4016-9255-96521b273b81.snapshot",
            "TEST_CATEGORY_GROUPINGS":"/table_snapshot/TEST_CATEGORY_GROUPINGS.csv/363f
4a59-b725-4459-826d-3188bde6a971.snapshot",
            "TEST_SITES":"/table_snapshot/TEST_SITES.csv/78e0aecc-3ec6-4406-b86e-bac4b1
0ea63b.snapshot"
         }
      }
   ],
   "create_time":null,
```

```
    "source_records_count":6000,
    "source_records_size":0,
    "size_kb":4758
}
```

禁用 Cube

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/cubes/{cubeName}/disable`

路径变量

- cubeName - `必选` `string` Cube 名称.

响应示例

(同 "启用 Cube")

清理 Cube

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/cubes/{cubeName}/purge`

路径变量

- cubeName - `必选` `string` Cube 名称.

响应示例

(同 "启用 Cube")

Job REST API

> 提示
>
> 使用API前请确保已阅读前面的 访问及安全验证 章节，知道如何在API中添加认证信息

- 恢复 job
- 抛弃 job
- 返回 job 信息
- 返回 job 每步输出

恢复 job

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/jobs/{jobId}/resume`

路径变量

- jobId - `必选` `string` Job id.

响应示例

```
{
   "uuid":"c143e0e4-ac5f-434d-acf3-46b0d15e3dc6",
   "last_modified":1407908916705,
   "name":"test_kylin_cube_with_slr_empty - 19700101000000_20140731160000 - BUILD - PDT
 2014-08-12 22:48:36",
   "type":"BUILD",
   "duration":0,
   "related_cube":"test_kylin_cube_with_slr_empty",
   "related_segment":"19700101000000_20140731160000",
   "exec_start_time":0,
   "exec_end_time":0,
   "mr_waiting":0,
   "steps":[
      {
         "interruptCmd":null,
         "name":"Create Intermediate Flat Hive Table",
         "sequence_id":0,
         "exec_cmd":"hive -e \"DROP TABLE IF EXISTS kylin_intermediate_test_kylin_cube_
with_slr_desc_19700101000000_20140731160000_c143e0e4_ac5f_434d_acf3_46b0d15e3dc6;\nCREA
TE EXTERNAL TABLE IF NOT EXISTS kylin_intermediate_test_kylin_cube_with_slr_desc_197001
01000000_20140731160000_c143e0e4_ac5f_434d_acf3_46b0d15e3dc6\n(\nCAL_DT date\n,LEAF_CAT
EG_ID int\n,LSTG_SITE_ID int\n,META_CATEG_NAME string\n,CATEG_LVL2_NAME string\n,CATEG_
LVL3_NAME string\n,LSTG_FORMAT_NAME string\n,SLR_SEGMENT_CD smallint\n,SELLER_ID bigint
\n,PRICE decimal\n)\nROW FORMAT DELIMITED FIELDS TERMINATED BY '\\177'\nSTORED AS SEQUE
NCEFILE\nLOCATION '/tmp/kylin-c143e0e4-ac5f-434d-acf3-46b0d15e3dc6/kylin_intermediate_t
est_kylin_cube_with_slr_desc_19700101000000_20140731160000_c143e0e4_ac5f_434d_acf3_46b0
d15e3dc6';\nSET mapreduce.job.split.metainfo.maxsize=-1;\nSET mapred.compress.map.outpu
t=true;\nSET mapred.map.output.compression.codec=com.hadoop.compression.lzo.LzoCodec;\n
```

```
SET mapred.output.compress=true;\nSET mapred.output.compression.codec=com.hadoop.compre
ssion.lzo.LzoCodec;\nSET mapred.output.compression.type=BLOCK;\nSET mapreduce.job.max.s
plit.locations=2000;\nSET hive.exec.compress.output=true;\nSET hive.auto.convert.join.n
oconditionaltask = true;\nSET hive.auto.convert.join.noconditionaltask.size = 300000000
;\nINSERT OVERWRITE TABLE kylin_intermediate_test_kylin_cube_with_slr_desc_197001010000
00_20140731160000_c143e0e4_ac5f_434d_acf3_46b0d15e3dc6\nSELECT\nTEST_KYLIN_FACT.CAL_DT\
n,TEST_KYLIN_FACT.LEAF_CATEG_ID\n,TEST_KYLIN_FACT.LSTG_SITE_ID\n,TEST_CATEGORY_GROUPING
S.META_CATEG_NAME\n,TEST_CATEGORY_GROUPINGS.CATEG_LVL2_NAME\n,TEST_CATEGORY_GROUPINGS.C
ATEG_LVL3_NAME\n,TEST_KYLIN_FACT.LSTG_FORMAT_NAME\n,TEST_KYLIN_FACT.SLR_SEGMENT_CD\n,TE
ST_KYLIN_FACT.SELLER_ID\n,TEST_KYLIN_FACT.PRICE\nFROM TEST_KYLIN_FACT\nINNER JOIN TEST_
CAL_DT\nON TEST_KYLIN_FACT.CAL_DT = TEST_CAL_DT.CAL_DT\nINNER JOIN TEST_CATEGORY_GROUPI
NGS\nON TEST_KYLIN_FACT.LEAF_CATEG_ID = TEST_CATEGORY_GROUPINGS.LEAF_CATEG_ID AND TEST_
KYLIN_FACT.LSTG_SITE_ID = TEST_CATEGORY_GROUPINGS.SITE_ID\nINNER JOIN TEST_SITES\nON TE
ST_KYLIN_FACT.LSTG_SITE_ID = TEST_SITES.SITE_ID\nINNER JOIN TEST_SELLER_TYPE_DIM\nON TE
ST_KYLIN_FACT.SLR_SEGMENT_CD = TEST_SELLER_TYPE_DIM.SELLER_TYPE_CD\nWHERE (test_kylin_f
act.cal_dt < '2014-07-31 16:00:00')\n;\n\"",
         "interrupt_cmd":null,
         "exec_start_time":0,
         "exec_end_time":0,
         "exec_wait_time":0,
         "step_status":"PENDING",
         "cmd_type":"SHELL_CMD_HADOOP",
         "info":null,
         "run_async":false
      },
      {
         "interruptCmd":null,
         "name":"Extract Fact Table Distinct Columns",
         "sequence_id":1,
         "exec_cmd":" -conf C:/kylin/Kylin/server/src/main/resources/hadoop_job_conf_me
dium.xml -cubename test_kylin_cube_with_slr_empty -input /tmp/kylin-c143e0e4-ac5f-434d-
acf3-46b0d15e3dc6/kylin_intermediate_test_kylin_cube_with_slr_desc_19700101000000_20140
731160000_c143e0e4_ac5f_434d_acf3_46b0d15e3dc6 -output /tmp/kylin-c143e0e4-ac5f-434d-ac
f3-46b0d15e3dc6/test_kylin_cube_with_slr_empty/fact_distinct_columns -jobname Kylin_Fac
t_Distinct_Columns_test_kylin_cube_with_slr_empty_Step_1",
         "interrupt_cmd":null,
         "exec_start_time":0,
         "exec_end_time":0,
         "exec_wait_time":0,
         "step_status":"PENDING",
         "cmd_type":"JAVA_CMD_HADOOP_FACTDISTINCT",
         "info":null,
         "run_async":true
      },
      {
         "interruptCmd":null,
         "name":"Load HFile to HBase Table",
         "sequence_id":12,
         "exec_cmd":" -input /tmp/kylin-c143e0e4-ac5f-434d-acf3-46b0d15e3dc6/test_kylin
_cube_with_slr_empty/hfile/ -htablename KYLIN-CUBE-TEST_KYLIN_CUBE_WITH_SLR_EMPTY-19700
101000000_20140731160000_11BB4326-5975-4358-804C-70D53642E03A -cubename test_kylin_cube
_with_slr_empty",
         "interrupt_cmd":null,
```

```
            "exec_start_time":0,
            "exec_end_time":0,
            "exec_wait_time":0,
            "step_status":"PENDING",
            "cmd_type":"JAVA_CMD_HADOOP_NO_MR_BULKLOAD",
            "info":null,
            "run_async":false
        }
    ],
    "job_status":"PENDING",
    "progress":0.0
}
```

抛弃 job

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/jobs/{jobId}/cancel`

路径变量

- jobId - `必选` `string` Job id.

响应示例

(同 "恢复 job")

返回 job 信息

`GET /jobs/{jobId}`

路径变量

- jobId - `必选` `string` Job id.

响应示例

(同 "恢复 Job")

返回 Job 每步输出

`GET /{jobId}/steps/{stepId}/output`

路径变量

- jobId - `必选` `string` Job id.
- stepId - `必选` `string` 步骤 id; 步骤id由Job id和步骤序列id组成; 例如, jobId 是 "fb479e54-837f-49a2-b457-651fc50be110", 第三步 步骤 id 是 "fb479e54-837f-49a2-b457-651fc50be110-3",

响应示例

```
{
   "cmd_output":"log string"
}
```

数据源 REST API

> 提示
>
> 使用API前请确保已阅读前面的 访问及安全验证 章节，知道如何在API中添加认证信息

- 返回 Hive 表信息
- 返回 Hive 表扩展信息
- 返回多个 Hive 表
- 加载 Hive 表

Get Hive Table

`请求方式 GET`

`访问路径 http://host:port/kylin/api/tables/{tableName}`

请求参数

- tableName - `必选` `string` 表名.

响应示例

```
{
    uuid: "69cc92c0-fc42-4bb9-893f-bd1141c91dbe",
    name: "SAMPLE_07",
    columns: [{
        id: "1",
        name: "CODE",
        datatype: "string"
    }, {
        id: "2",
        name: "DESCRIPTION",
        datatype: "string"
    }, {
        id: "3",
        name: "TOTAL_EMP",
        datatype: "int"
    }, {
        id: "4",
        name: "SALARY",
        datatype: "int"
    }],
    database: "DEFAULT",
    last_modified: 1419330476755
}
```

返回 Hive 表扩展信息

`请求方式 GET`

`访问路径 http://host:port/kylin/api/tables/{tableName}/exd-map`

请求参数

- tableName - `可选` `string` 表名.

响应示例

```
{
    "minFileSize": "46055",
    "totalNumberFiles": "1",
    "location": "hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/sample_07",
    "lastAccessTime": "1418374103365",
    "lastUpdateTime": "1398176493340",
    "columns": "struct columns { string code, string description, i32 total_emp, i32 sa
lary}",
    "partitionColumns": "",
    "EXD_STATUS": "true",
    "maxFileSize": "46055",
    "inputformat": "org.apache.hadoop.mapred.TextInputFormat",
    "partitioned": "false",
    "tableName": "sample_07",
    "owner": "hue",
    "totalFileSize": "46055",
    "outputformat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat"
}
```

返回多个 Hive 表

`请求方式 GET`

`访问路径 http://host:port/kylin/api/tables`

请求参数

- project- `必选` `string` 返回该项目下所有表.
- ext- `可选` `boolean` 要返回扩展信息设置该参数为true.

响应示例

```
[
 {
    uuid: "53856c96-fe4d-459e-a9dc-c339b1bc3310",
    name: "SAMPLE_08",
    columns: [{
        id: "1",
        name: "CODE",
        datatype: "string"
    }, {
        id: "2",
        name: "DESCRIPTION",
        datatype: "string"
    }, {
        id: "3",
        name: "TOTAL_EMP",
        datatype: "int"
    }, {
        id: "4",
        name: "SALARY",
        datatype: "int"
    }],
    database: "DEFAULT",
    cardinality: {},
    last_modified: 0,
    exd: {
        minFileSize: "46069",
        totalNumberFiles: "1",
        location: "hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/sample_08",
        lastAccessTime: "1398176495945",
        lastUpdateTime: "1398176495981",
        columns: "struct columns { string code, string description, i32 total_emp, i32
salary}",
        partitionColumns: "",
        EXD_STATUS: "true",
        maxFileSize: "46069",
        inputformat: "org.apache.hadoop.mapred.TextInputFormat",
        partitioned: "false",
        tableName: "sample_08",
        owner: "hue",
        totalFileSize: "46069",
        outputformat: "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat"
    }
  }
]
```

加载 Hive 表

`请求方式 POST`

`访问路径 http://host:port/kylin/api/tables/{tables}/{project}`

请求参数

- tables - `必选` `string` 你想要加载的hive表, 用逗号分隔
- project - `必选` `String` 指定hive表将要加载到哪个项目

响应示例

```
{
    "result.loaded": ["DEFAULT.SAMPLE_07"],
    "result.unloaded": ["sapmle_08"]
}
```

缓存 REST API

> 提示
>
> 使用API前请确保已阅读前面的 访问及安全验证 章节，知道如何在API中添加认证信息

清理缓存

`请求方式 PUT`

`访问路径 http://host:port/kylin/api/cache/{type}/{name}/{action}`

路径变量

- type - `必选` `string` 'METADATA' 或者 'CUBE'
- name - `必选` `string` 缓存键值, 比如 Cube 名称.
- action - `必选` `string` 'create', 'update' or 'drop'

JDBC 驱动程序

KAP提供了JDBC驱动程序，支持JDBC接口的应用可以通过该驱动程序访问KAP进行数据查询。

用户可以在KAP安装目录的lib子目录下，找到名称为kylin-jdbc-kap-*.jar的jar包，这就是KAP的JDBC驱动。

连接

KAP JDBC驱动程序遵循了JDBC标准接口，用户可通过URL制定JDBC所连接的KAP服务，这个URL格式为：

```
jdbc:kylin://<hostname>:<port>/<project_name>
```

- 如果KAP服务启用了SSL，则端口号应该使用KAP服务的HTTPS端口
- 如果端口未指定，则JDBC驱动将使用HTTP和HTTPS的默认端口
- project_name必须制定，用户必须保证该项目在KAP服务中存在

此外，用户需要为连接制定登陆的用户名、密码，以及SSL是否开启的标志，这些属性是：

- user: 登陆KAP服务的用户名
- password: 登陆KAP服务的密码
- ssl: true/false. 默认为false，如果是true，所有对KAP的访问都将基于HTTPS

以下给出一个建立Connection的例子：

```
Driver driver = (Driver) Class.forName("org.apache.kylin.jdbc.Driver").newInstance();
Properties info = new Properties();
info.put("user", "ADMIN");
info.put("password", "KYLIN");
Connection conn = driver.connect("jdbc:kylin://localhost:7070/kylin_project_name", info
);
```

基于Statement的查询

这里给出一个直接基于Statement进行查询的例子：

```
Driver driver = (Driver) Class.forName("org.apache.kylin.jdbc.Driver").newInstance();
Properties info = new Properties();
info.put("user", "ADMIN");
info.put("password", "KYLIN");
Connection conn = driver.connect("jdbc:kylin://localhost:7070/kylin_project_name", info
);
Statement state = conn.createStatement();
ResultSet resultSet = state.executeQuery("select * from test_table");
while (resultSet.next()) {
    System.out.println(resultSet.getString(1));
    System.out.println(resultSet.getString(2));
    System.out.println(resultSet.getString(3));
}
```

基于PreparedStatement的查询

这里给出一个基于PreparedStatement进行查询的例子：

```
Driver driver = (Driver) Class.forName("org.apache.kylin.jdbc.Driver").newInstance();
Properties info = new Properties();
info.put("user", "ADMIN");
info.put("password", "KYLIN");
Connection conn = driver.connect("jdbc:kylin://localhost:7070/kylin_project_name", info
);
PreparedStatement state = conn.prepareStatement("select * from test_table where id=?");
state.setInt(1, 10);
ResultSet resultSet = state.executeQuery();
while (resultSet.next()) {
    System.out.println(resultSet.getString(1));
    System.out.println(resultSet.getString(2));
    System.out.println(resultSet.getString(3));
}
```

其中，对PreparedStatement支持对以下类型的赋值操作：

- setString
- setInt
- setShort
- setLong
- setFloat
- setDouble
- setBoolean
- setByte
- setDate
- setTime
- setTimestamp

ODBC 驱动程序

KAP提供了Windows系统下的ODBC驱动程序，支持ODBC接口的应用可以通过该驱动程序访问KAP进行数据查询。

测试操作系统：Windows 7, Windows Server 2008 R2 测试应用：Tableau 8.x, Table 9.x, Power Desktop, Excel

在本文中，我们以Windows 7和Microsoft Office Excel为例，介绍ODBC驱动的安装和使用步骤。

前提条件

1. 安装Microsoft Visual C++ 2012（Redistributable）
    - 32位：下载：32bit version
    - 64位：下载：64bit version
2. ODBC驱动程序会调用KAP的Rest服务器，请提前确保KAP服务已正常运行。

安装

1. 如果你已经安装过KAP ODBC驱动，首先卸载已存在的Apache Kylin或KAP ODBC驱动。
2. 从下载链接下载附件驱动安装程序，并运行。
    - 32位Microsoft Office Excel：请安装KylinODBCDriver (x86).exe
    - 64位Microsoft Office Excel：请安装KylinODBCDriver (x64).exe

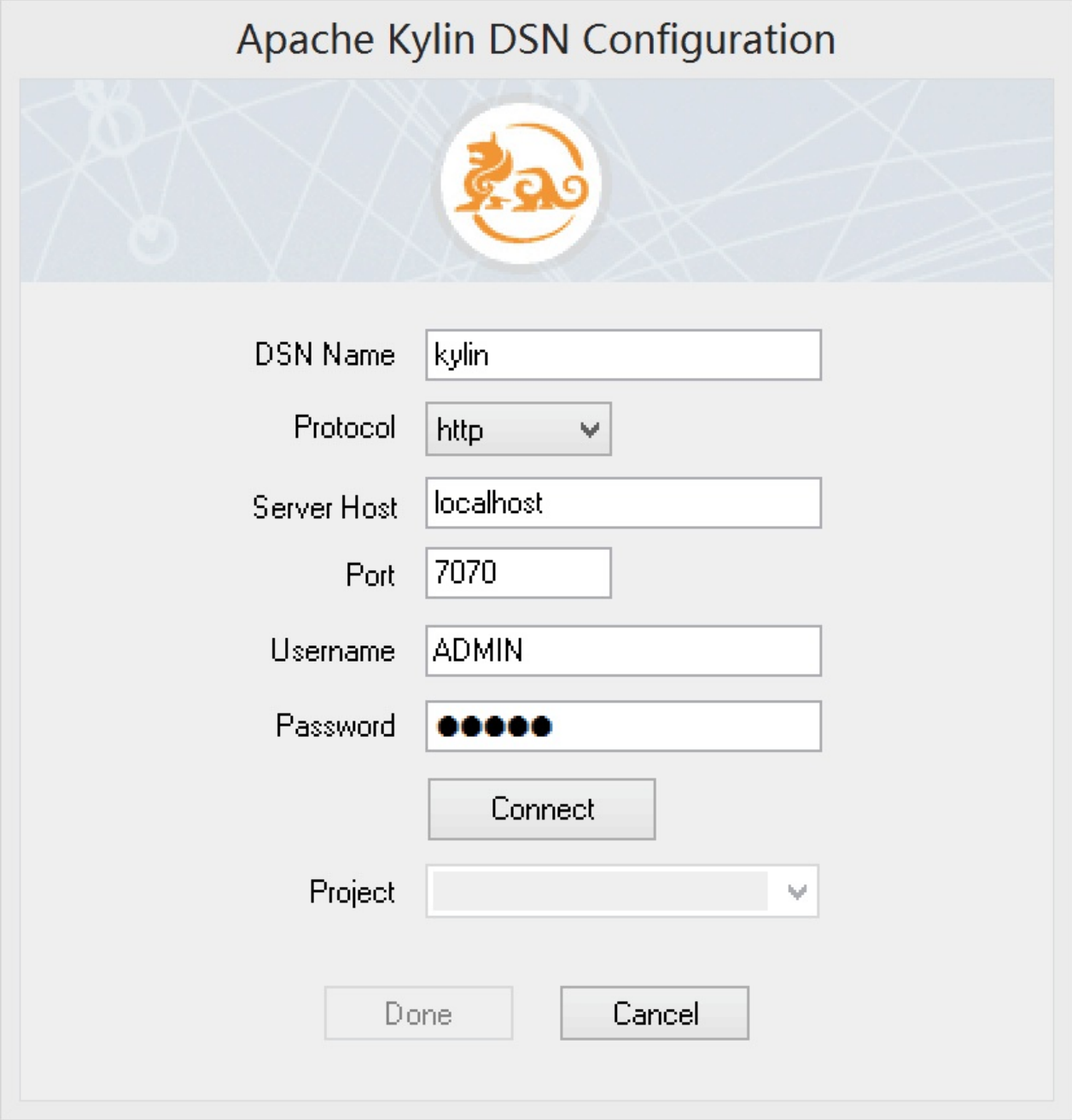
配置DSN

1.打开ODBC数据源管理器

- 32位ODBC驱动：请在“开始>运行”中打开C:\Windows\SysWOW64\odbcad32.exe
- 64位ODBC驱动：请直接在“控制面板>管理工具”中找到“ODBC Data Source Administrator”并打开

2.切换到系统DSN选项卡，单击“Add”，在弹出的驱动程序选择框中选择“KylinODBCDriver”并单击Finish按钮

3.在弹出的Kylin DSN Configuration Dialog中输入KAP服务器信息，如图所示



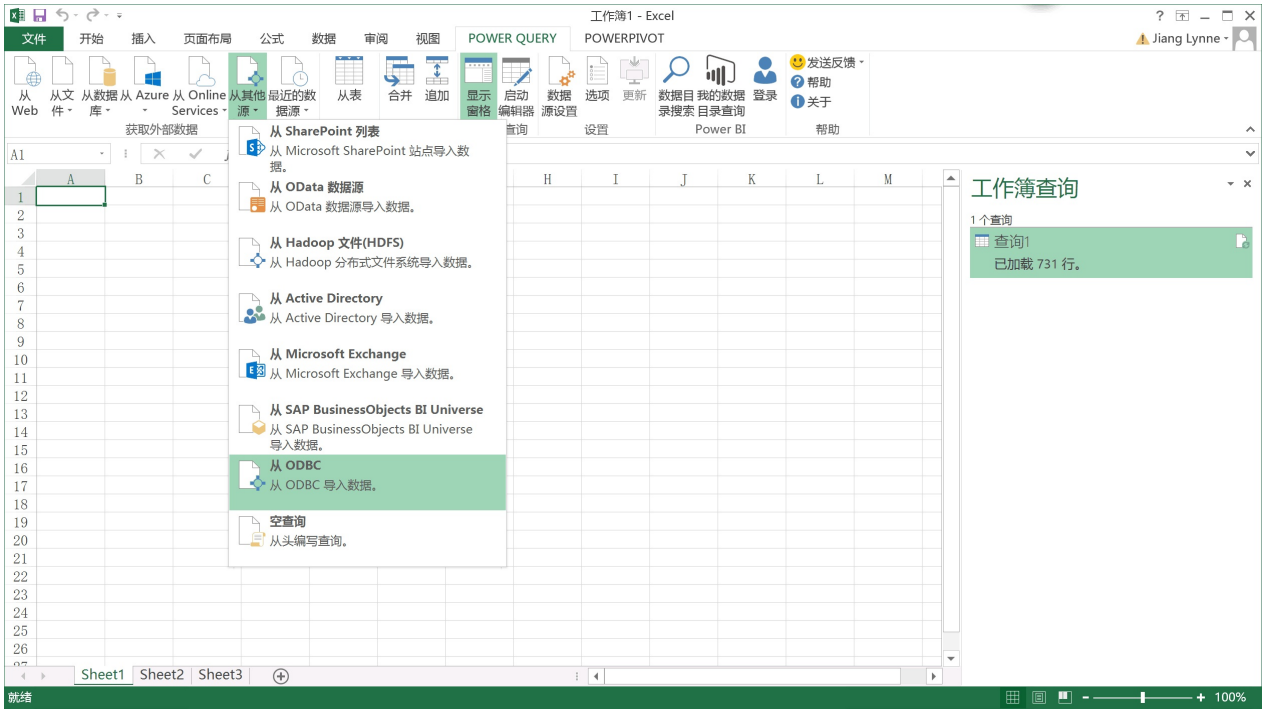
其中，各项参数介绍如下：

- DSN Name: DSN数据源名称
- Server Host: KAP服务器地址
- Port: KAP服务器端口号
- Username: KAP服务登录用户名
- Password: KAP服务登录密码
- Project: 查询所使用的KAP项目名称；输入完上面空白后单击Connect按钮，才可激活Project选择框

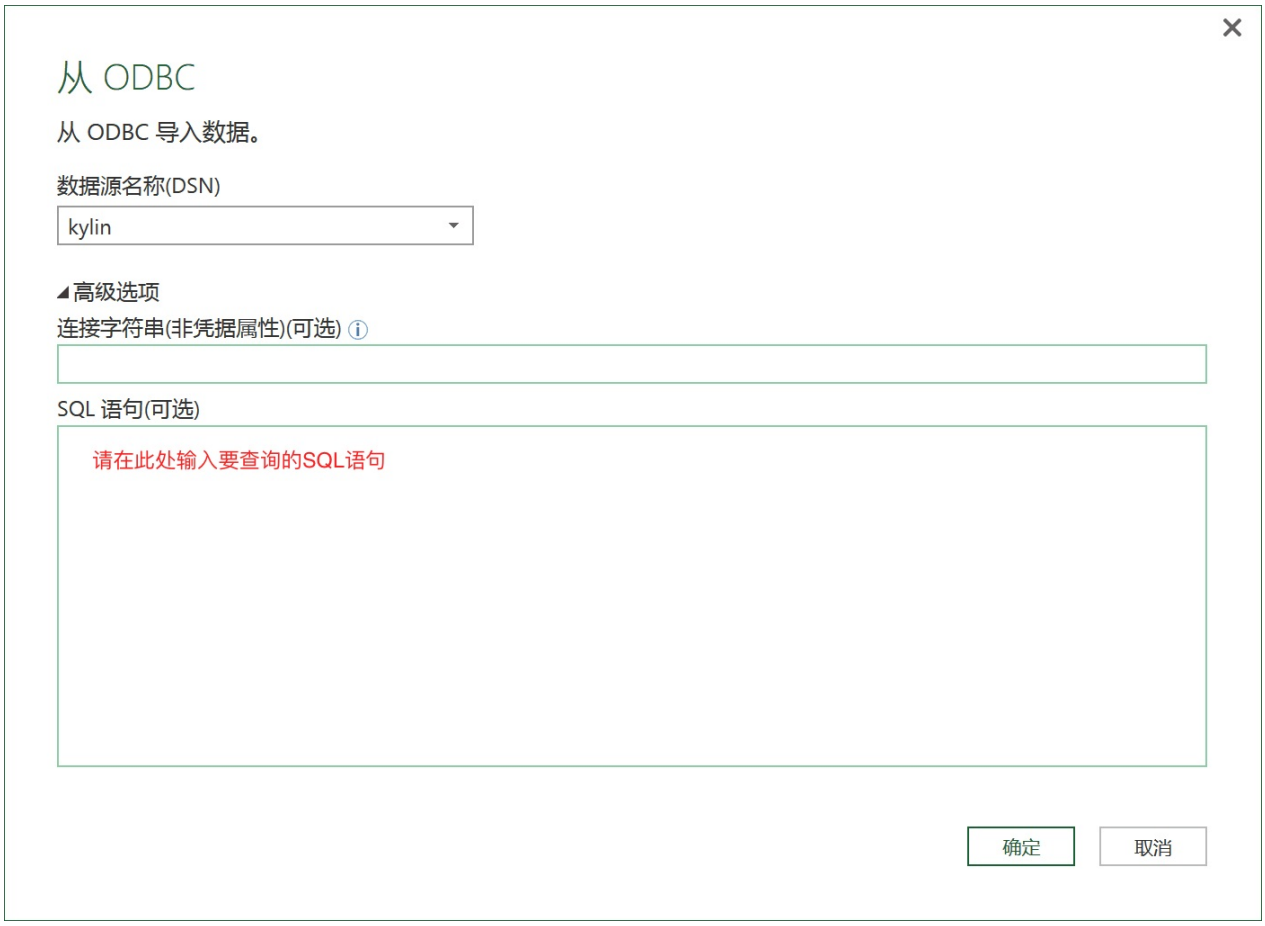
4.单击Done按钮

配置Excel

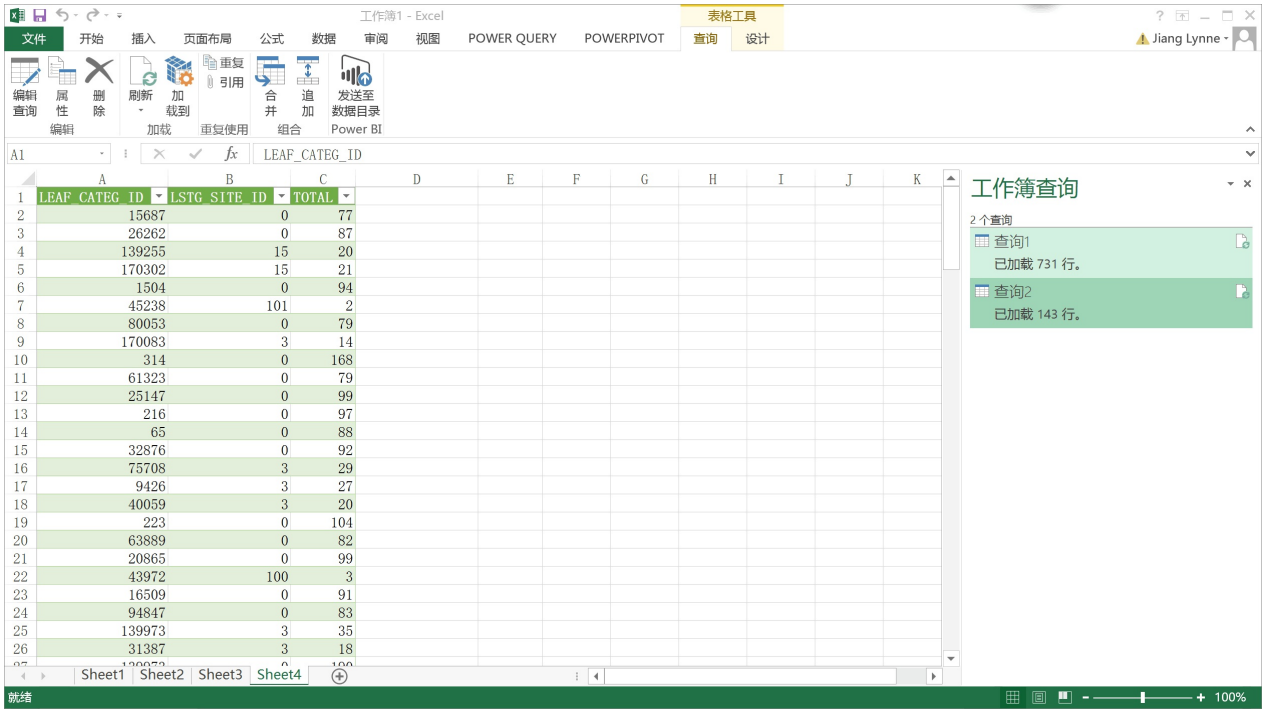
1.从Microsoft网站下载并安装Power Query 2.打开Excel，切换到Power Query的FastTab，单击From Other Sources并展开菜单，选择From ODBC项



3.在弹出的“从ODBC”输入框中，首先选择前面创建的DSN作为数据源（如kylin），然后在下方输入希望执行的SQL语句



4.这时，SQL的查询结果就显示到Excel表格中了。



5.如果KAP中有数据更新，用户可以直接单击Excel上的Refresh按钮对表格中的数据进行刷新。

特别提醒

如果用户希望使用ODBC驱动连接其他客户端应用，配置方式和该例类似。如果用户希望使用Tableau，可以访问Apache Kylin官网查看相关文档。

Tableau 8.x 集成

> KAP ODBC驱动程序与Tableau存在一些限制，请在尝试前仔细阅读本说明书。
>
> - 仅支持“managed”分析路径，KAP引擎将对意外的维度或度量报错
> - 请始终优先选择事实表，然后使用正确的连接条件添加查找表（cube中已定义的连接类型）
> - 请勿尝试在多个事实表或多个查找表之间进行连接；
> - 你可以尝试使用类似Tableau过滤器中seller id这样的高基数维度，但引擎现在将只返回有限个Tableau过滤器中的seller id。
>
> 如需更多详细信息或有任何问题，请联系KAP团队：`support@kyligence.io`

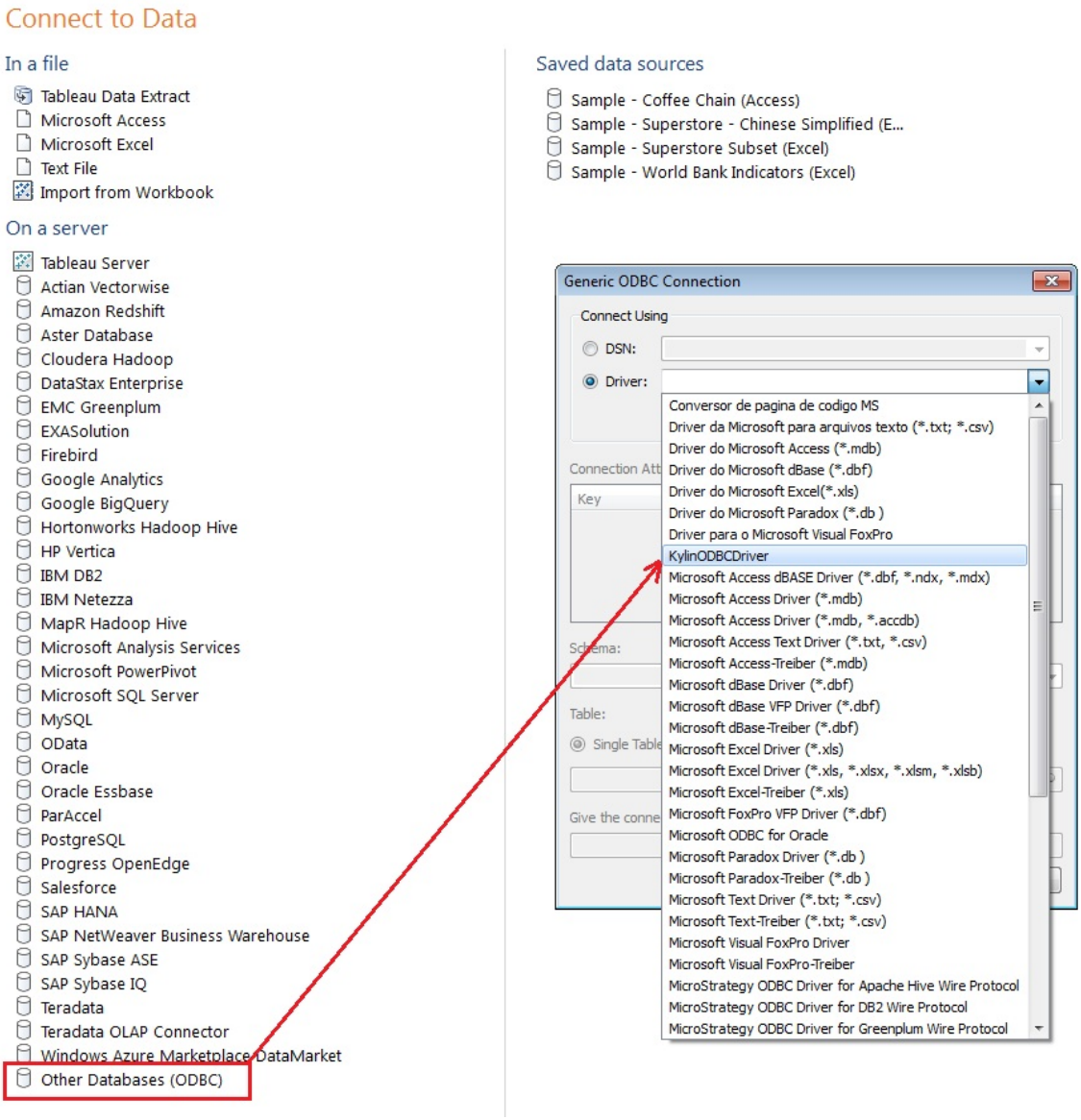
步骤1. KAP ODBC驱动程序

参考页面KAP ODBC 驱动程序教程。

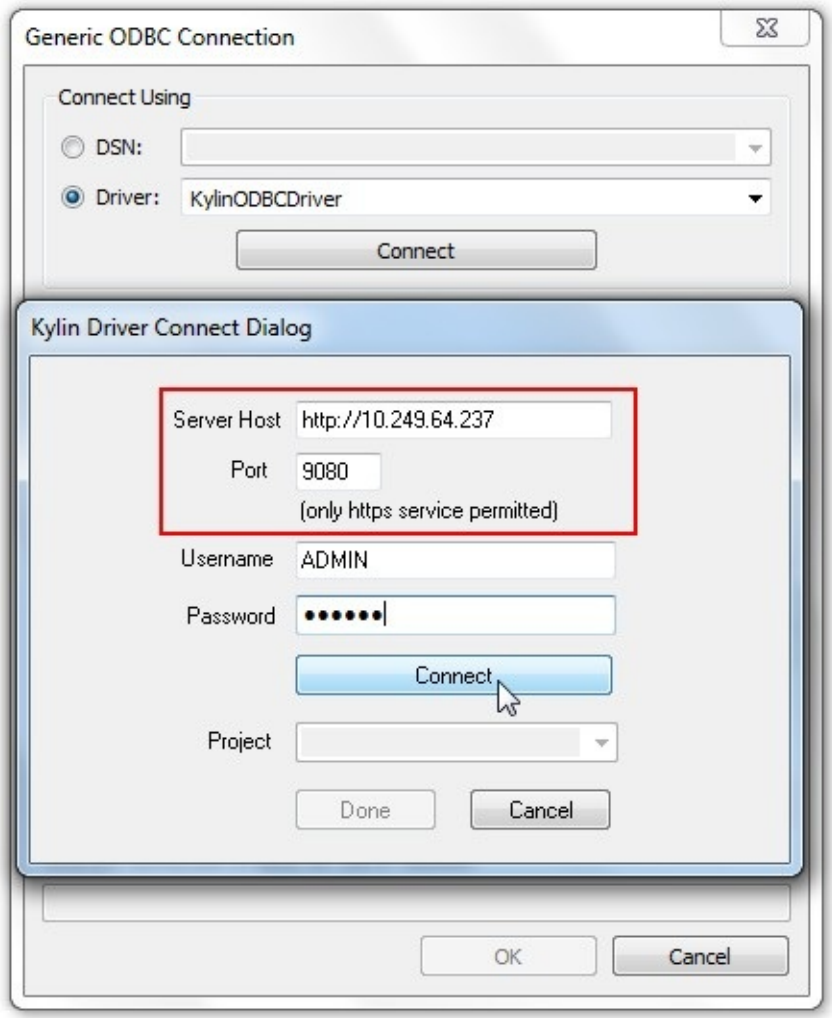
步骤2. 连接到KAP服务器

> 我们建议使用Connect Using Driver而不是Using DSN。

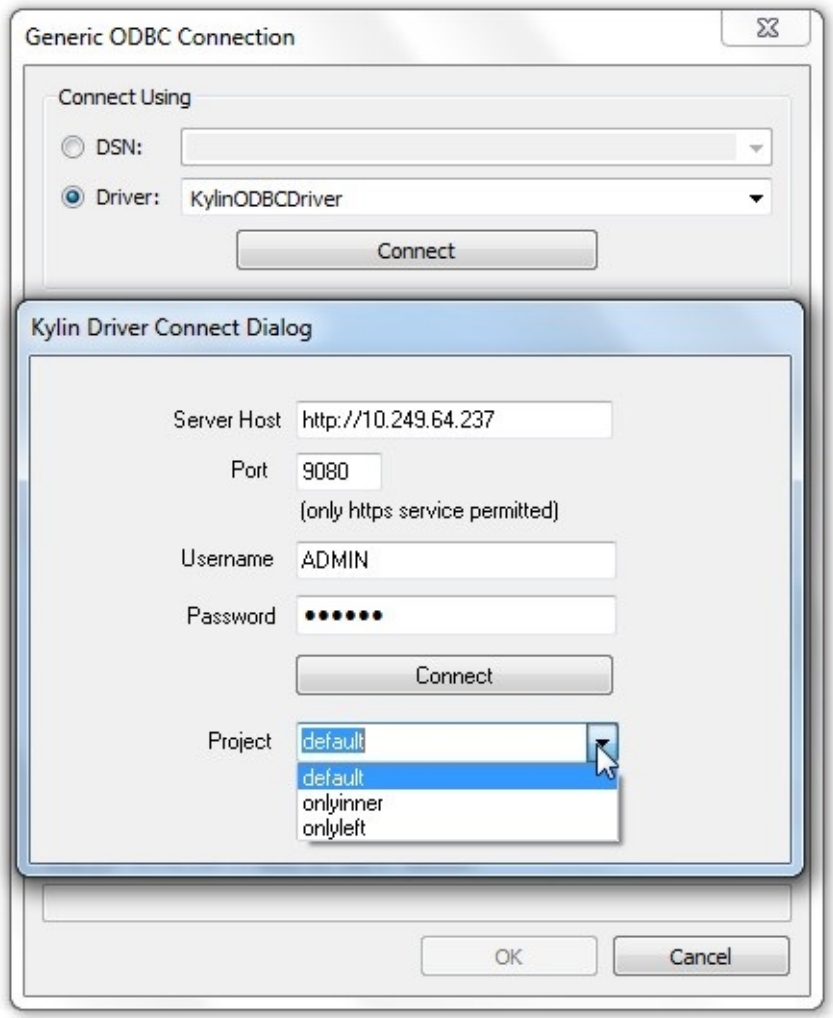
Connect Using Driver: 选择左侧面板中的“Other Database(ODBC)”和弹出窗口的“KylinODBCDriver”。



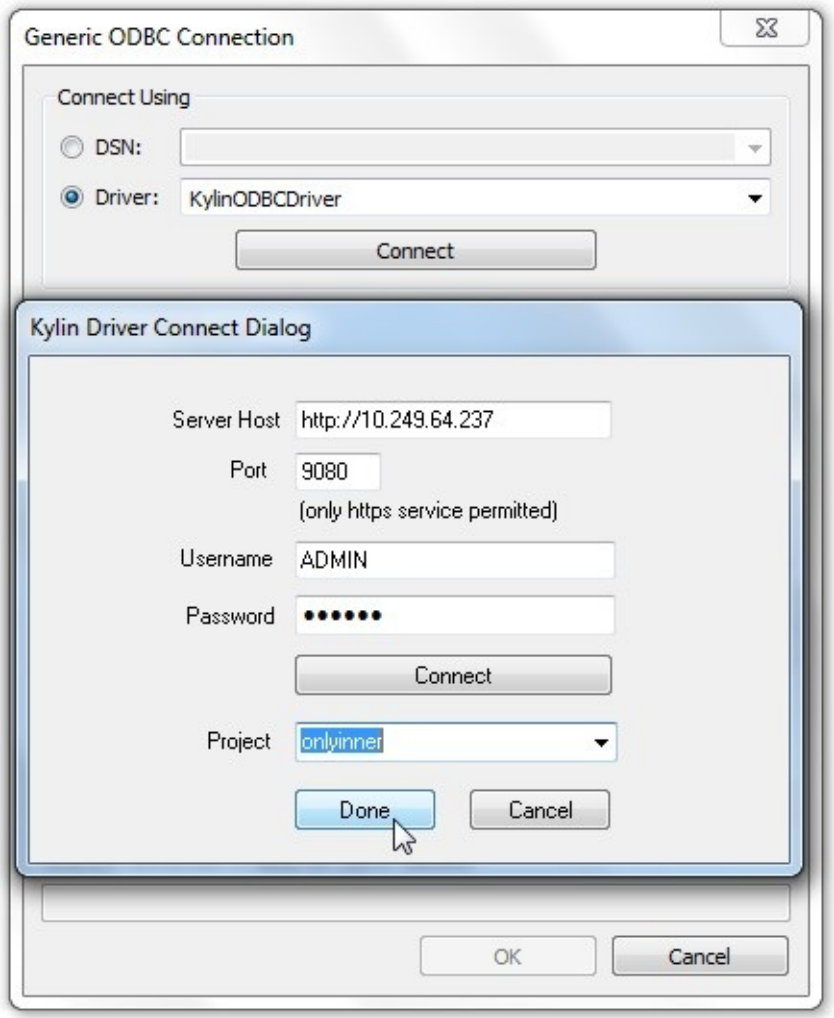
输入你的服务器位置和证书：服务器主机，端口，用户名和密码。



点击“Connect”获取你有权限访问的项目列表。有关权限的详细信息请参考Kylin Cube Permission Grant Tutorial。然后在下拉列表中选择你想要连接的项目。


Generic ODBC Connection
Connect Using
DSN:
Driver:
KylinODBCDriver
Connect
Kylin Driver Connect Dialog
Server Host
http://10.249.64.237
Port
9080
(only https service permitted)
Username
ADMIN
Password
Connect
Project
default
default
onlyinner
onlyleft
OK
Cancel

点击“Done”完成连接。

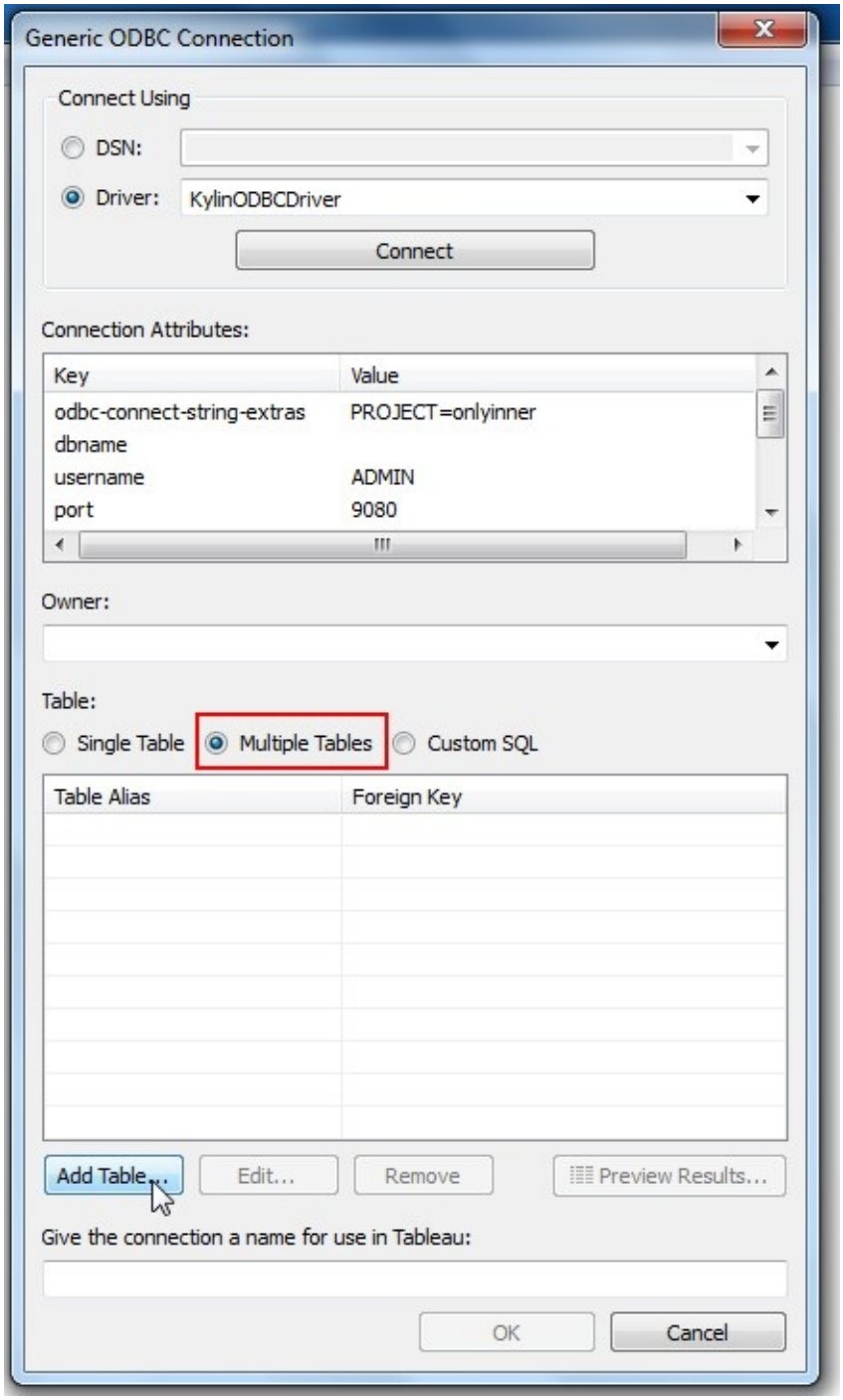


步骤3. 使用单表或多表

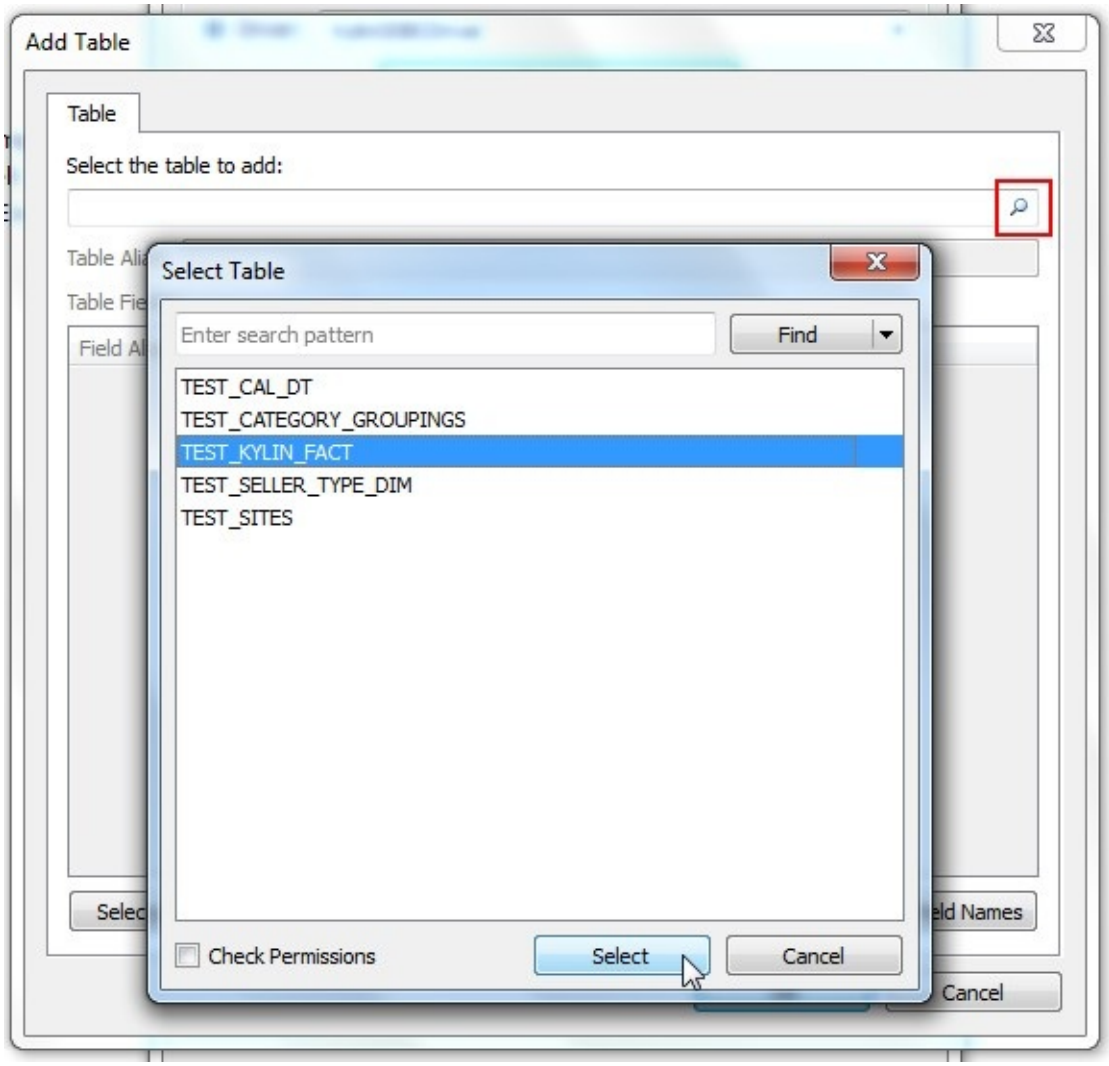
> 限制
>
> - 必须首先选择事实表
> - 请勿仅支持从查找表选择
> - 连接条件必须与cube定义匹配

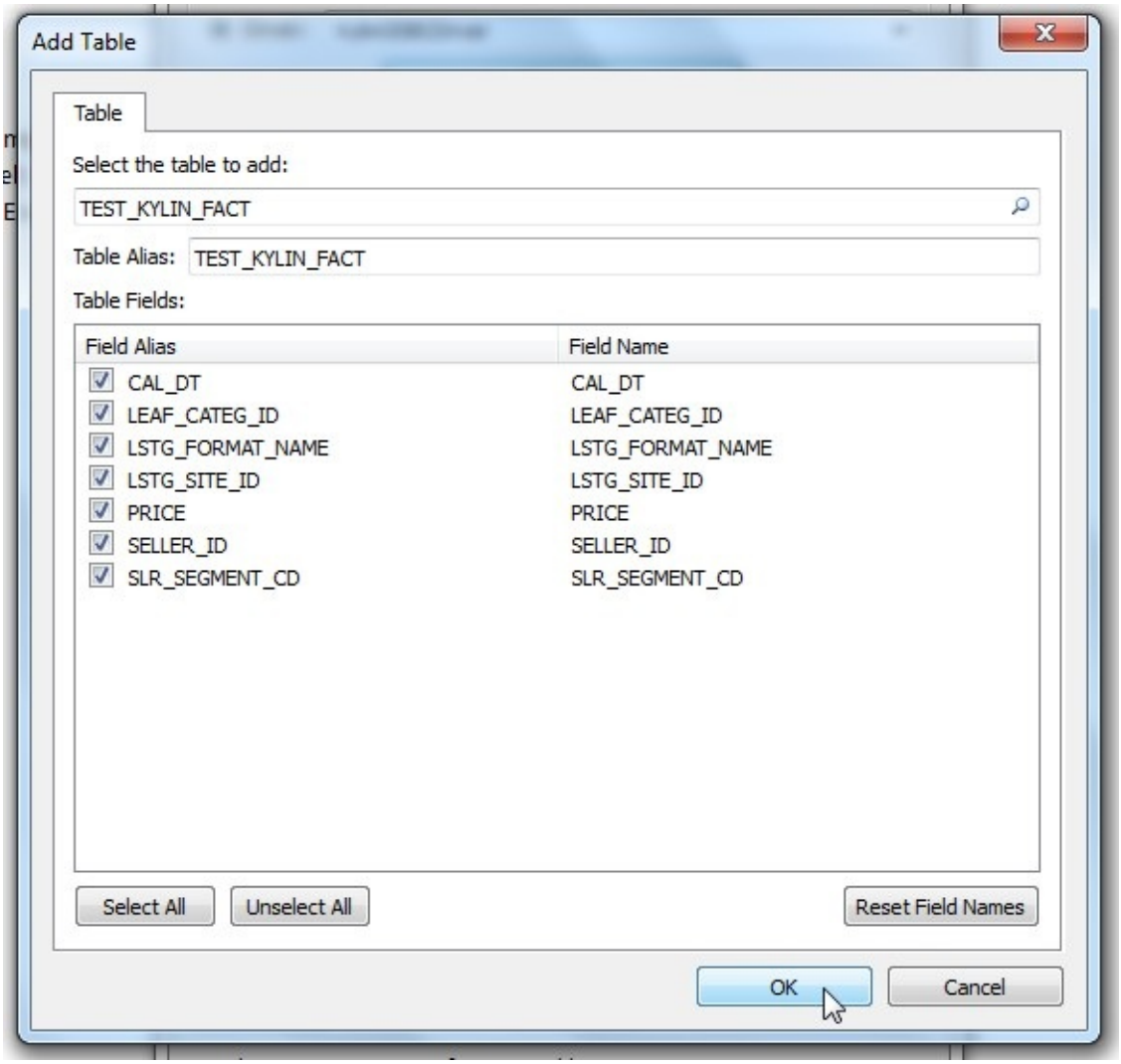
选择事实表

选择 `Multiple Tables` 。



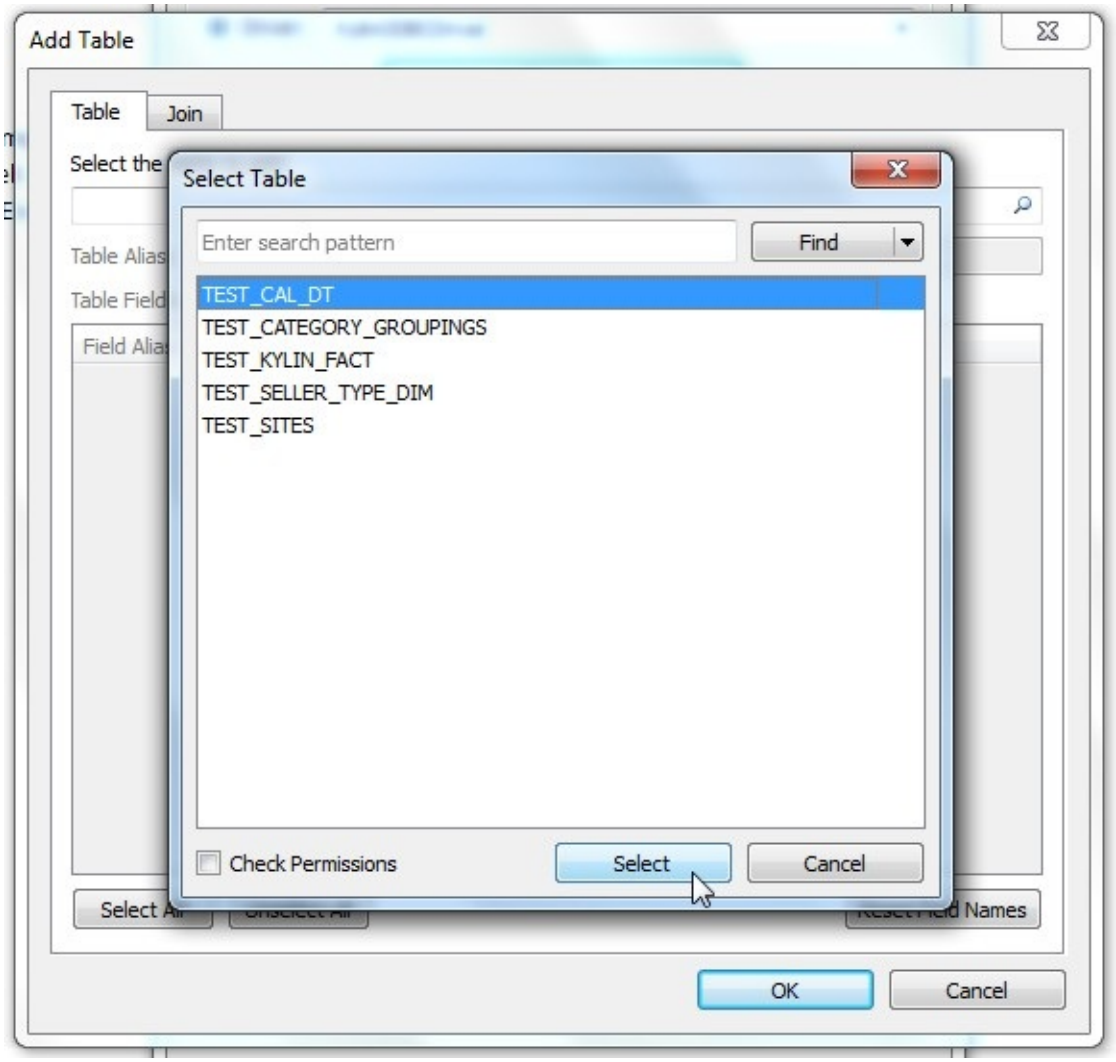
然后点击 `Add Table...` 添加一张事实表。


Add Table
Table
Select the table to add:
Select Table
Enter search pattern
Find
TEST_CAL_DT
TEST_CATEGORY_GROUPINGS
TEST_KYLIN_FACT
TEST_SELLER_TYPE_DIM
TEST_SITES
Check Permissions
Select
Cancel
Cancel

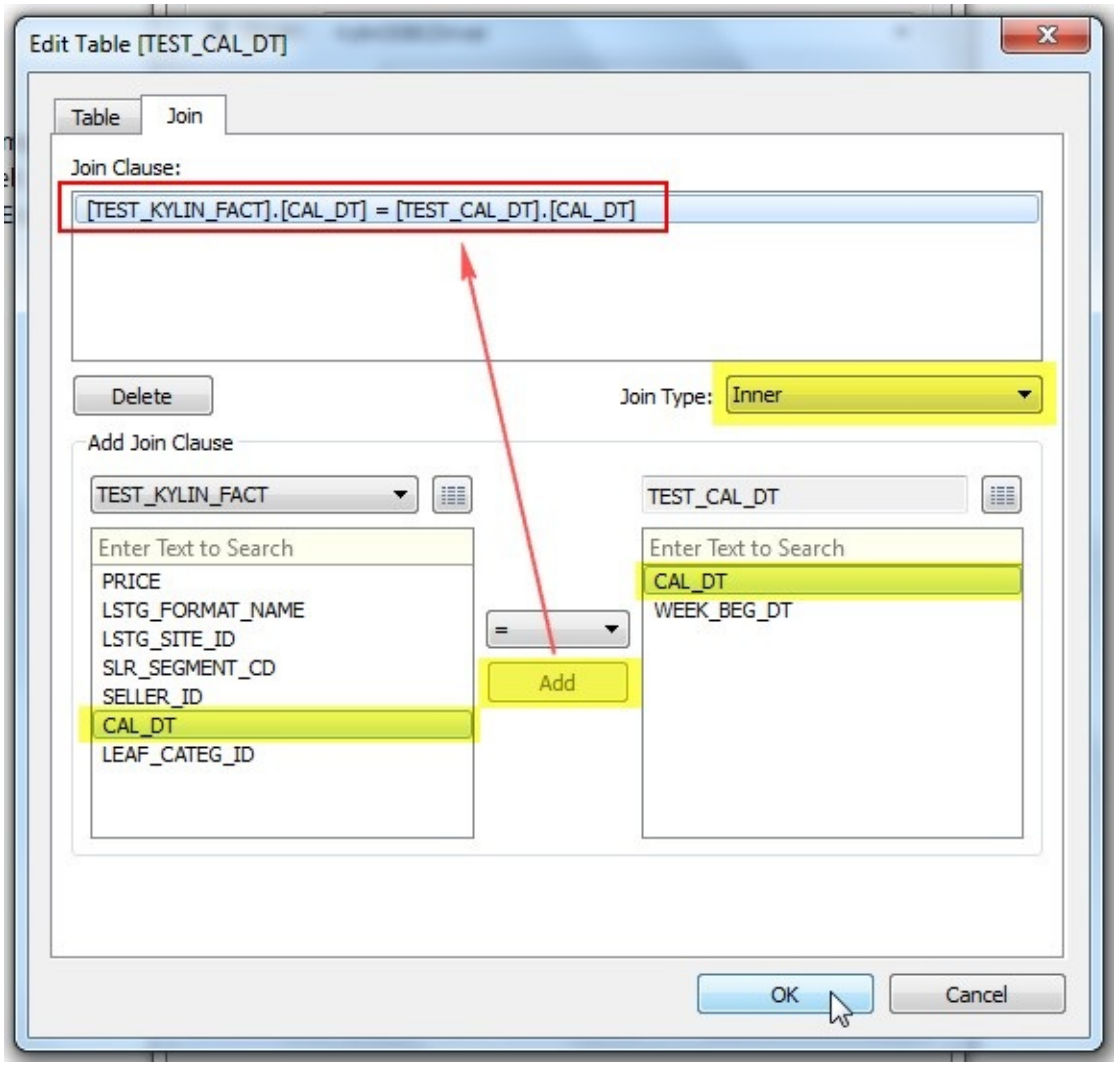


选择查找表

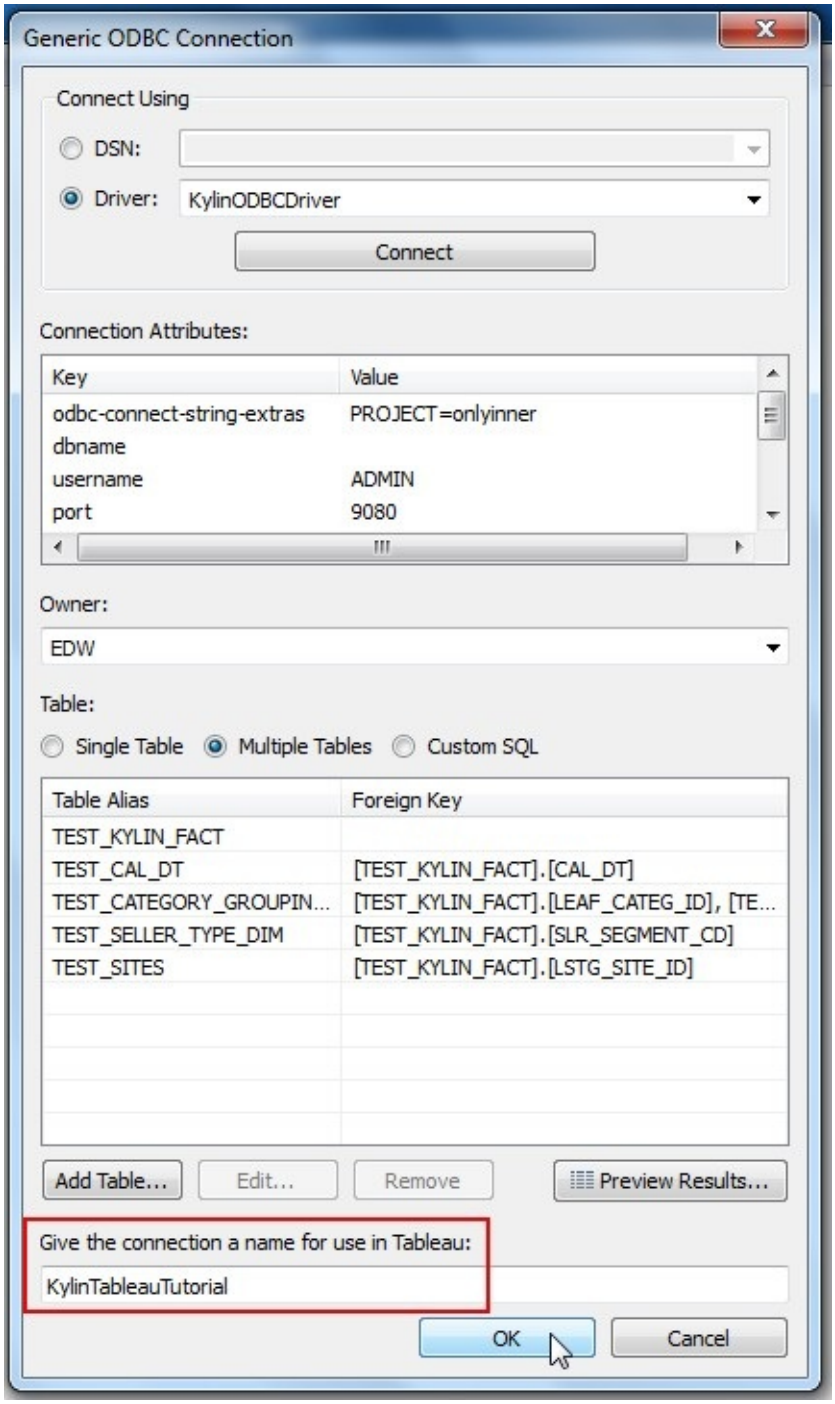
点击 `Add Table...` 添加一张查找表。


Add Table
Table
Join
Select the
Table Alias
Table Field
Field Alias
Select Table
Enter search pattern
Find
TEST_CAL_DT
TEST_CATEGORY_GROUPINGS
TEST_KYLIN_FACT
TEST_SELLER_TYPE_DIM
TEST_SITES
Check Permissions
Select
Cancel
Select All
Unselect All
Reset Field Names
OK
Cancel

仔细建立连接条款。



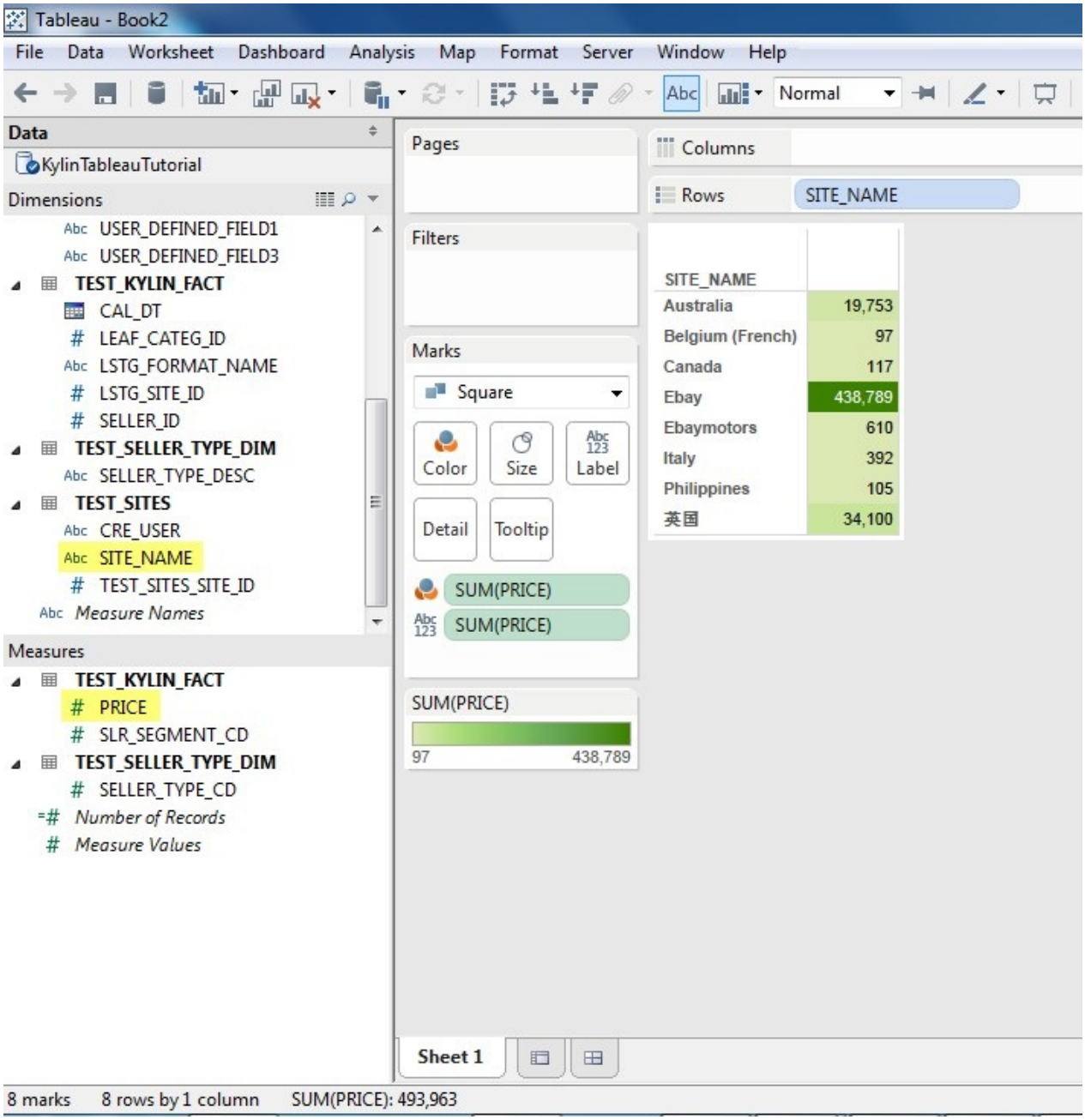
继续通过点击 `Add Table...` 添加表直到所有的查找表都被正确添加。命名此连接以在Tableau中使用。



使用**Connect Live**

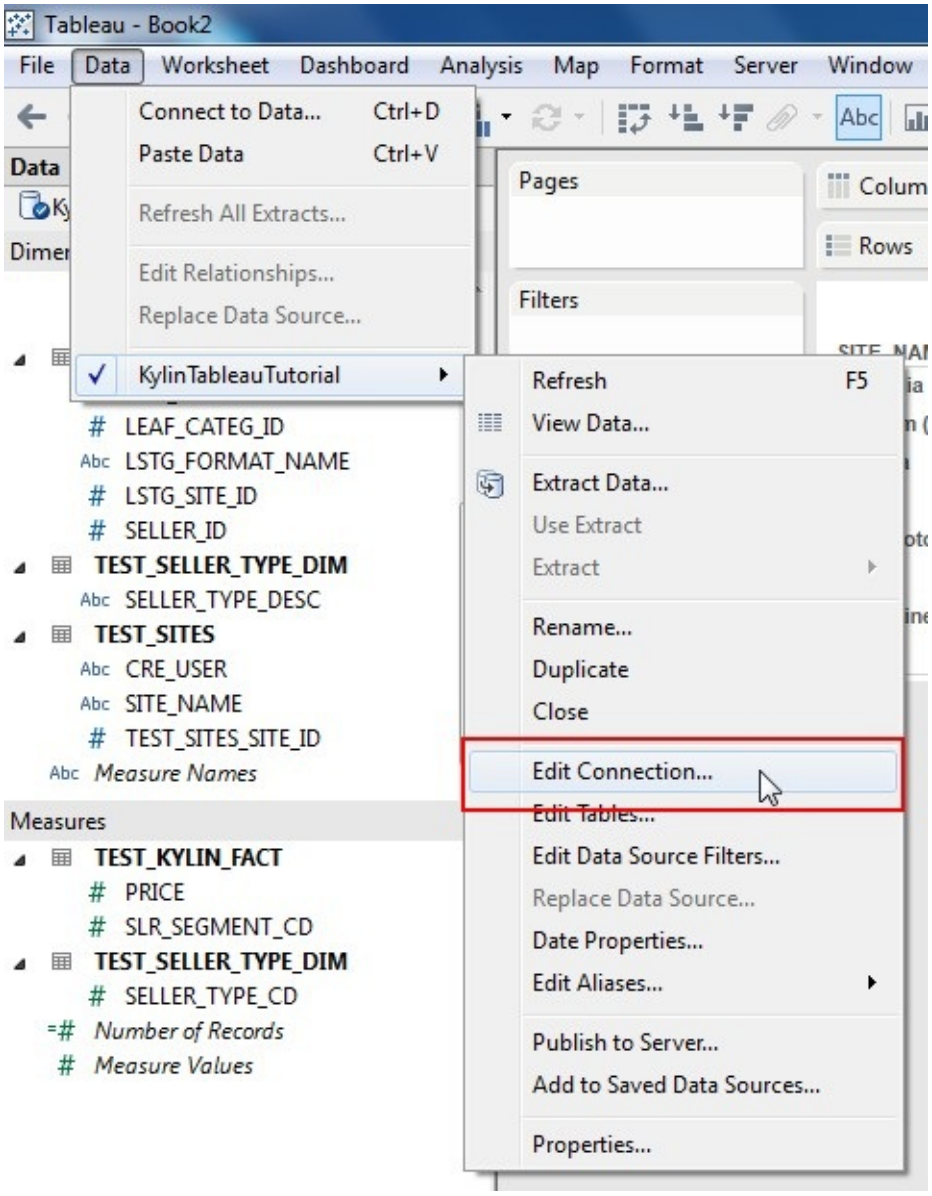
`Data Connection` 共有三种类型。选择 `Connect Live` 选项。

然后你就能够尽情使用Tableau进行分析。



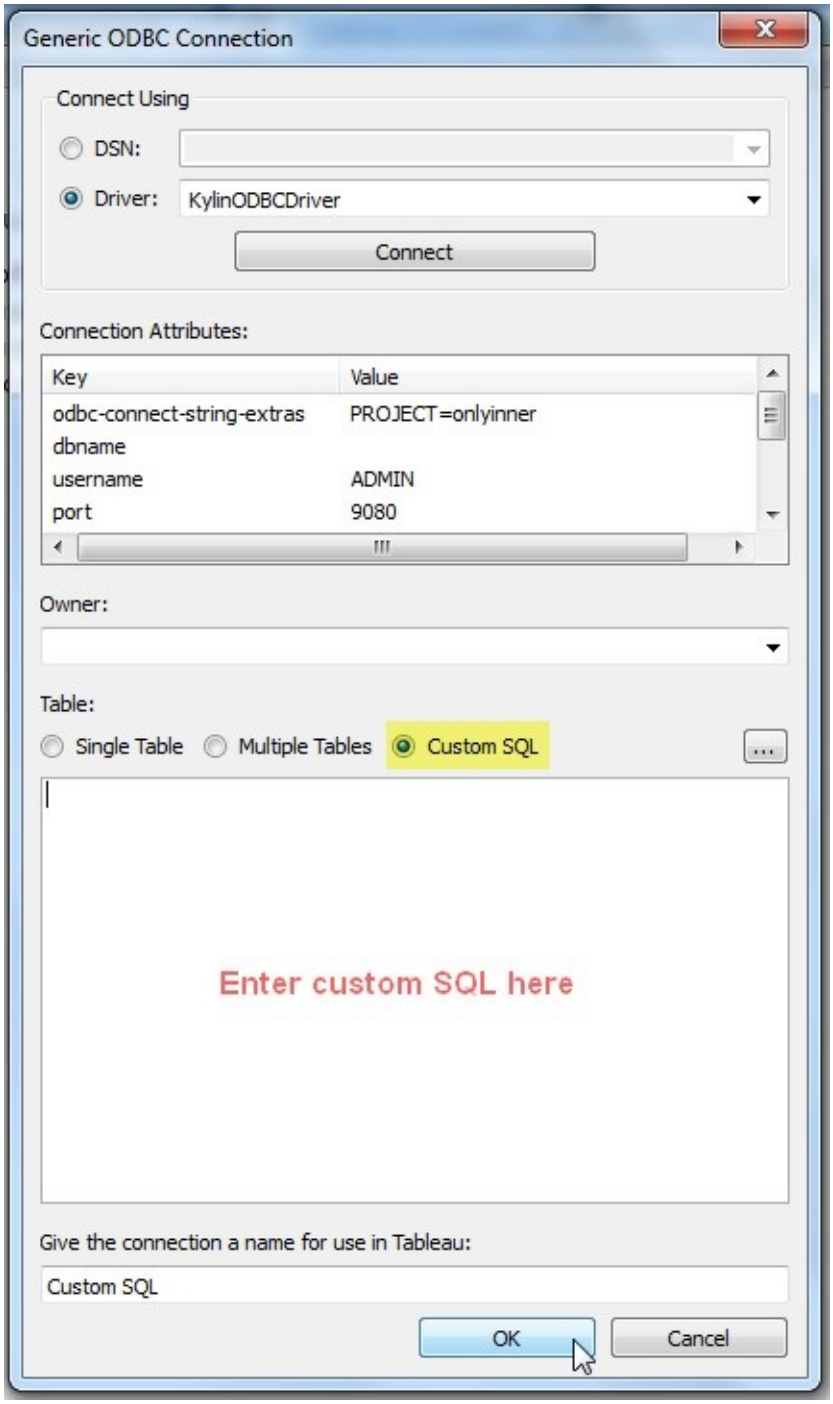
添加额外查找表

点击顶部菜单栏的 `Data` ，选择 `Edit Tables...` 更新查找表信息。



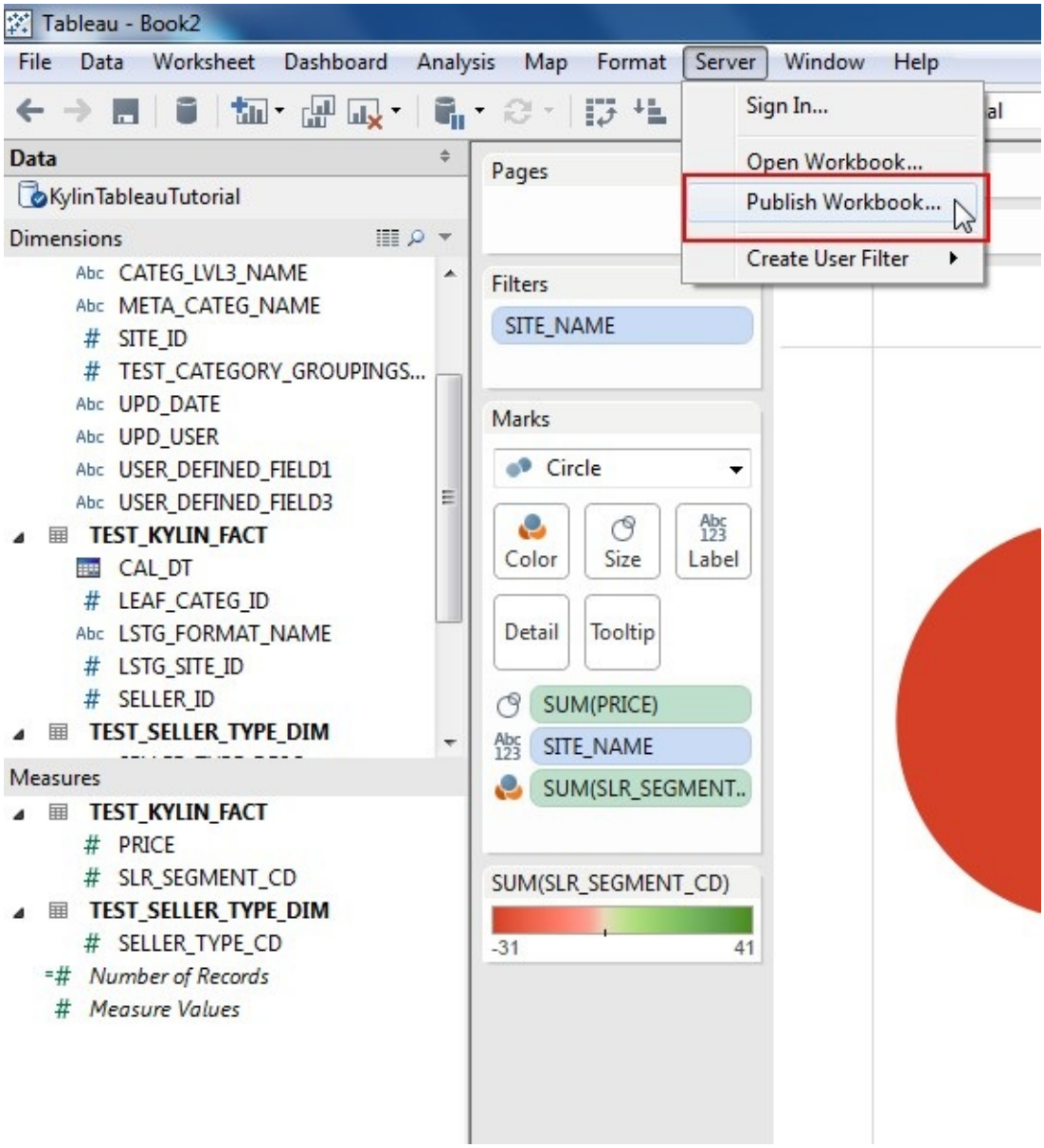
步骤4. 使用自定义SQL

使用自定义SQL类似于使用单表/多表，但你需要在 `Custom SQL` 标签复制你的SQL后采取同上指令。

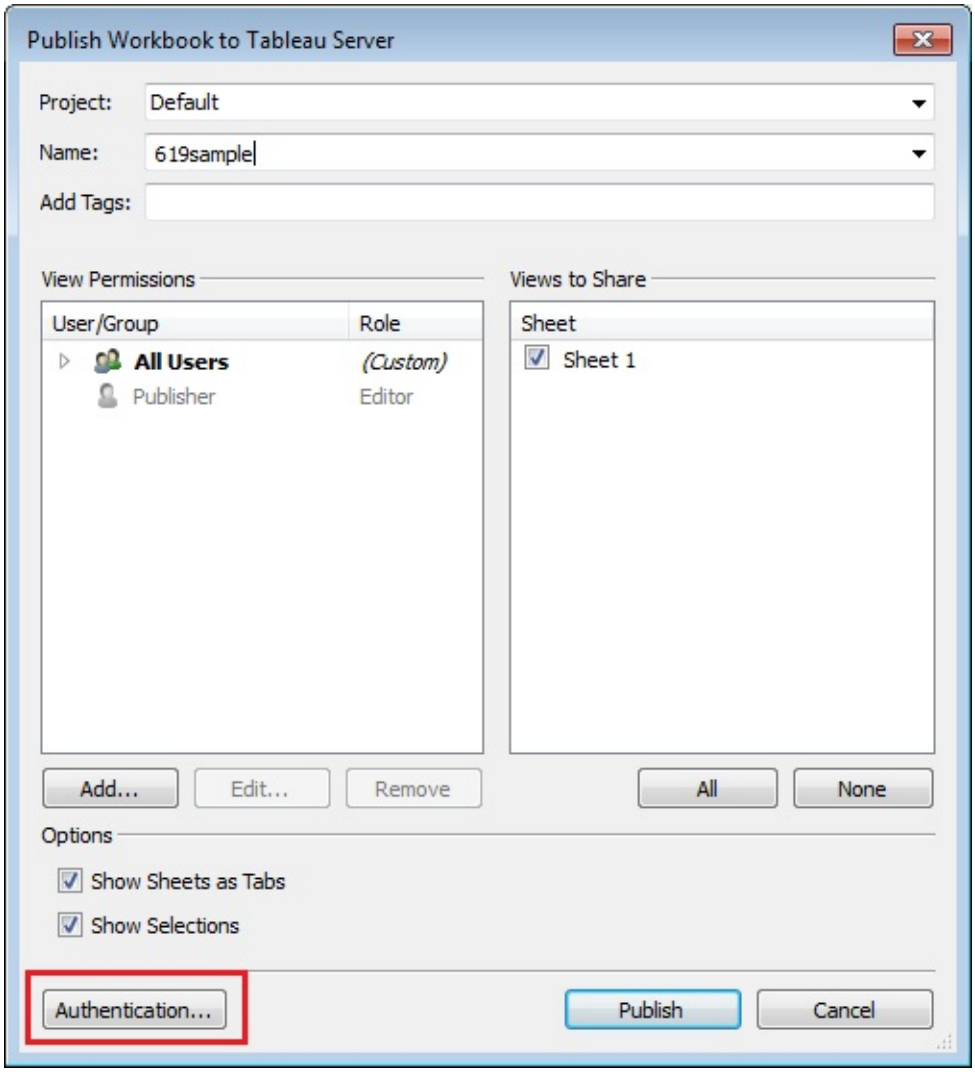


步骤5. 发布到Tableau服务器

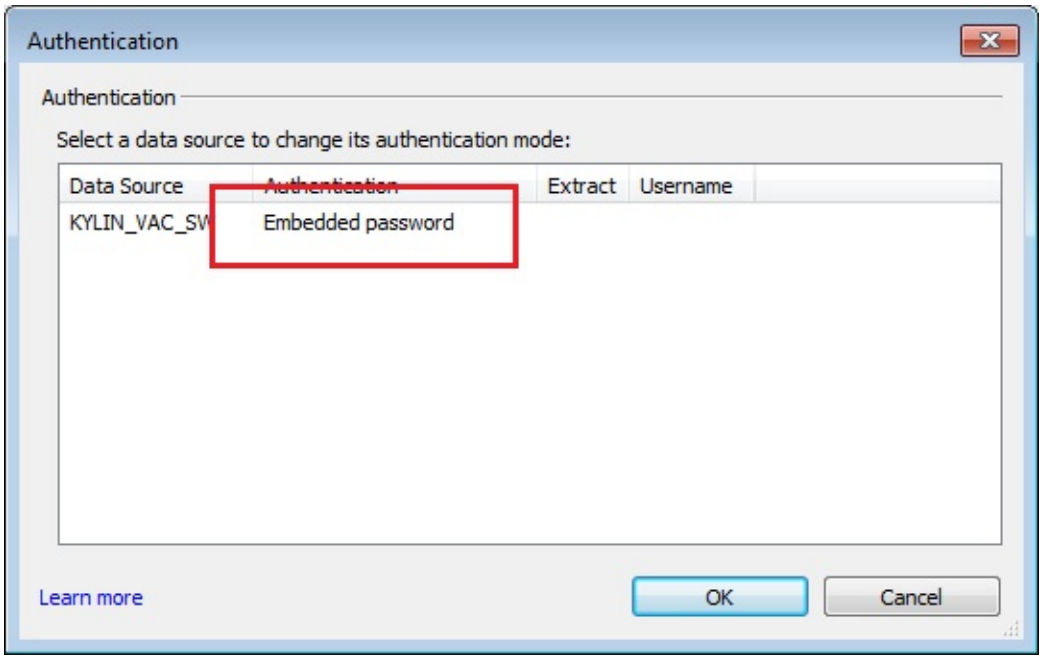
如果你已经完成使用Tableau制作一个仪表板，你可以将它发布到Tableau服务器上。 点击顶部菜单栏的 `Server` ，选择 `Publish Workbook...` 。



然后登陆你的Tableau服务器并准备发布。



如果你正在使用Connect Using Driver而不是DSN连接，你还将需要嵌入你的密码。点击左下方的 `Authentication` 按钮并选择 `Embedded Password` 。点击 `Publish` 然后你将看到结果。



小贴士

- 在Tableau中隐藏表名
  - Tableau将会根据源表名分组显示列，但用户可能希望根据其他不同的安排组织列。使用Tableau中的"Group by Folder"并创建文件夹来对不同的列分组。

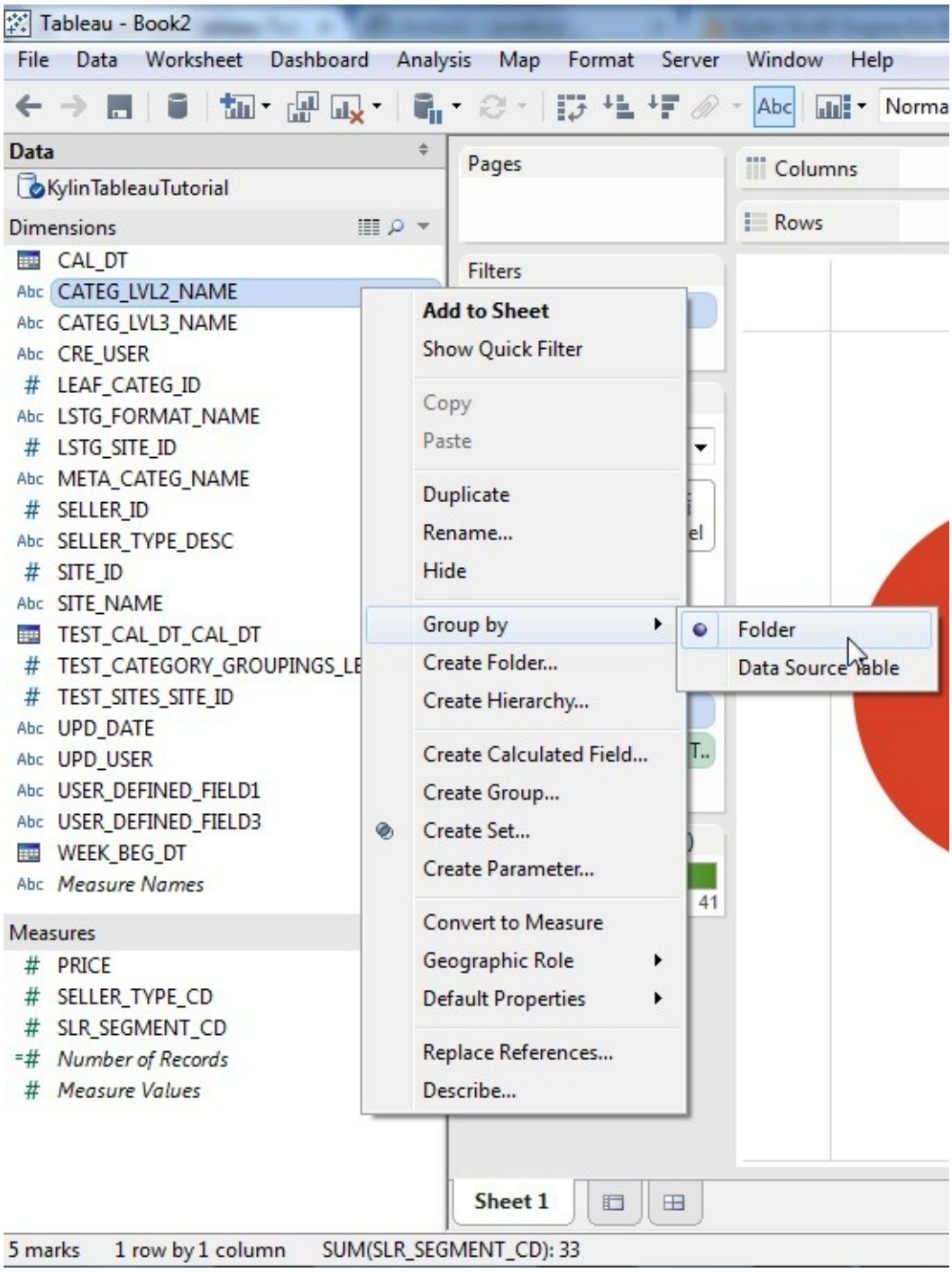


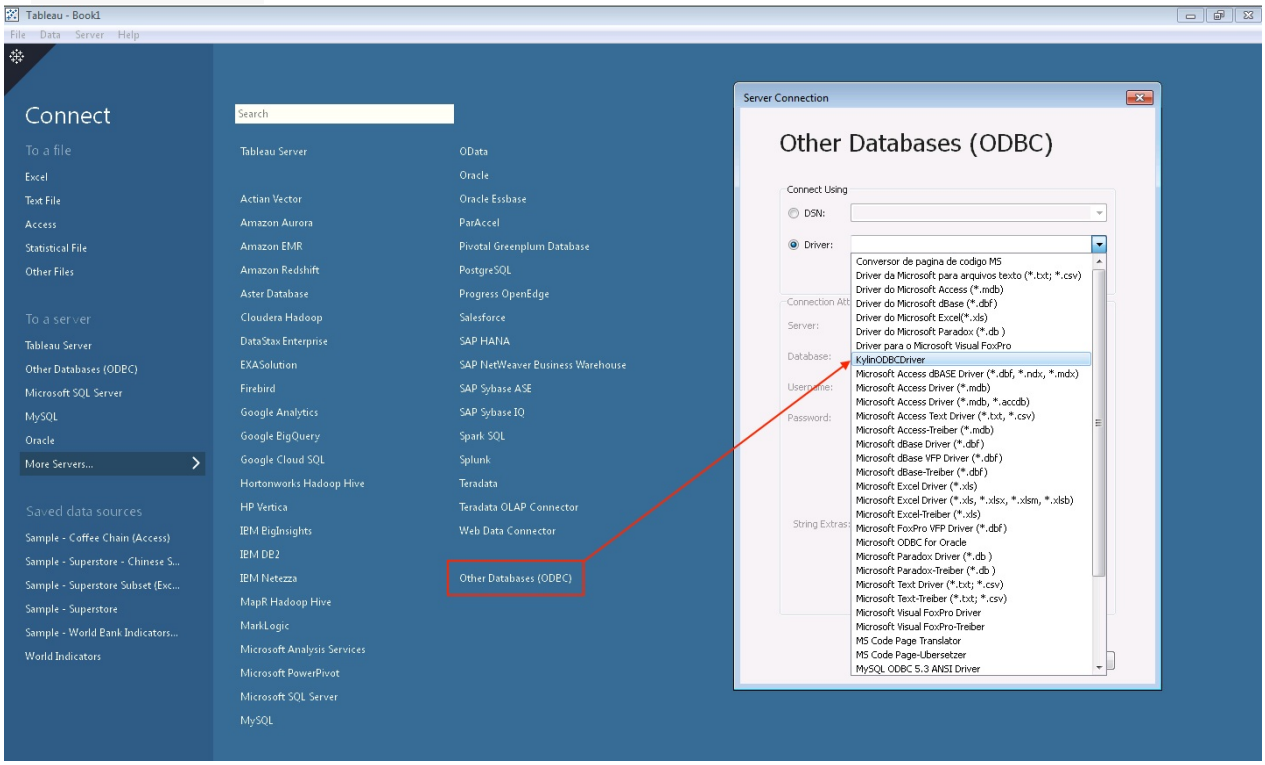
Tableau 9.x 集成

Install ODBC Driver

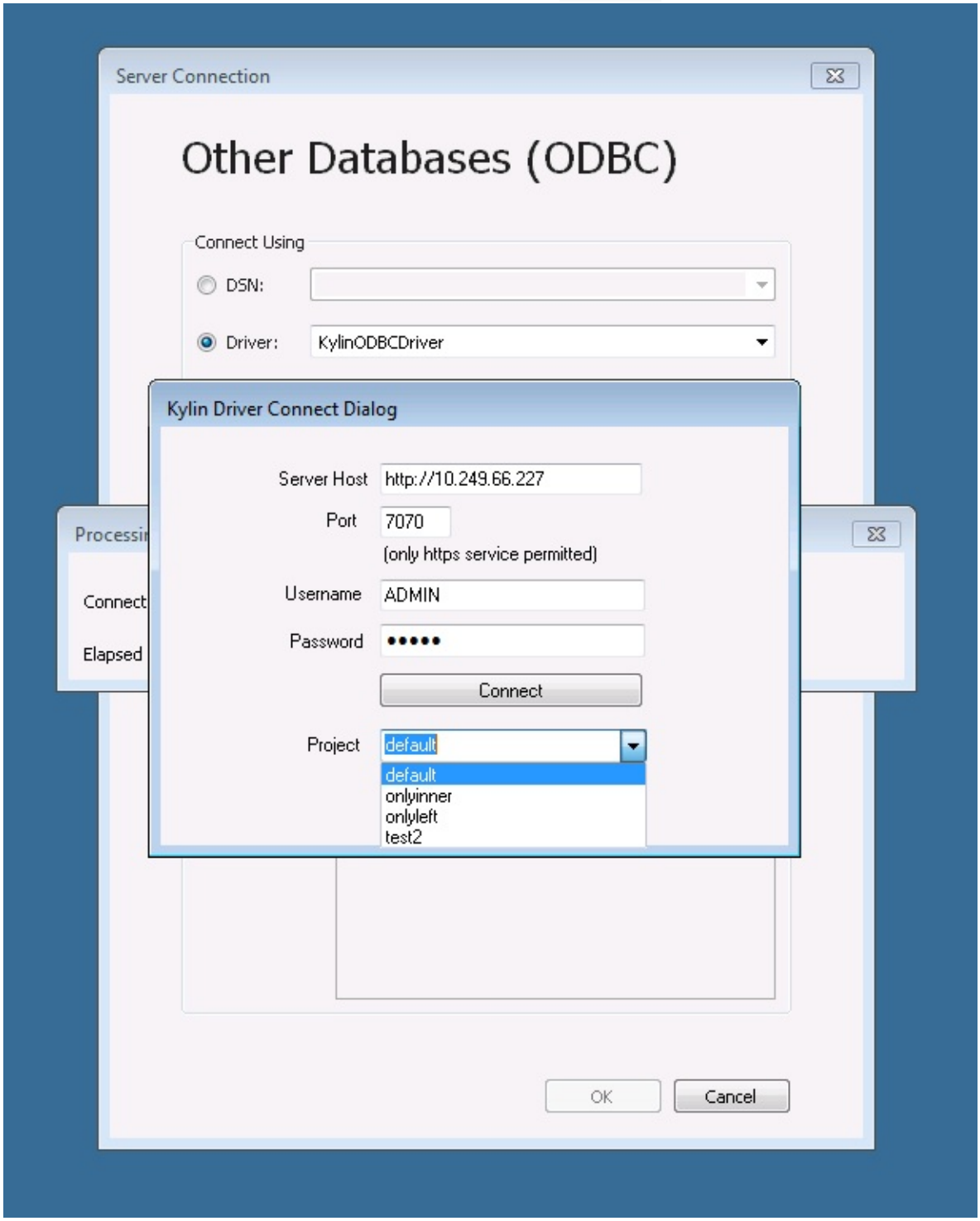
参考页面KAP ODBC 驱动程序教程KAP ODBC Driver **v1.2**. 如果你安装有早前版本，请卸载后再安装。

Connect to KAP Server

在Tableau 9.1创建新的数据连接，单击左侧面板中的 `Other Database(ODBC)` ，并在弹出窗口中选择 `KylinODBCDriver`

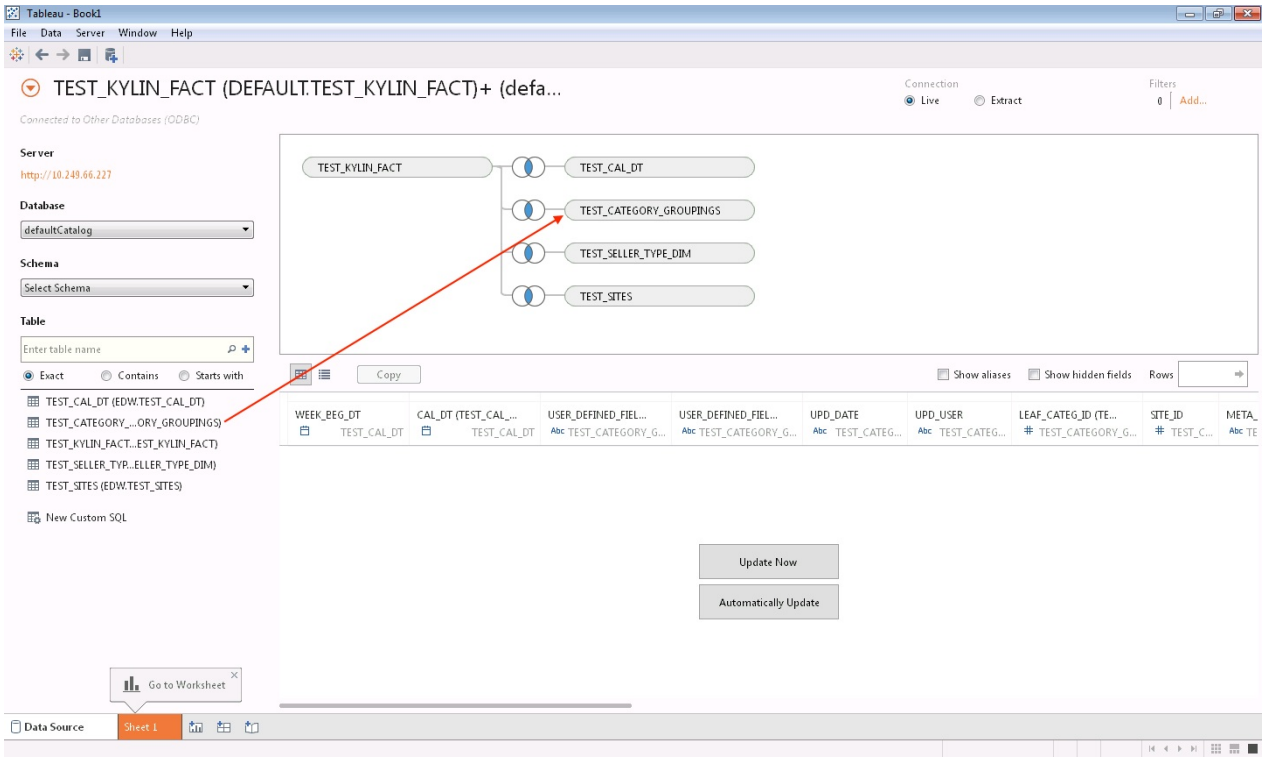


输入你的服务器地址、端口、项目、用户名和密码，点击 `Connect` 可获取有权限访问的所有项目列表。



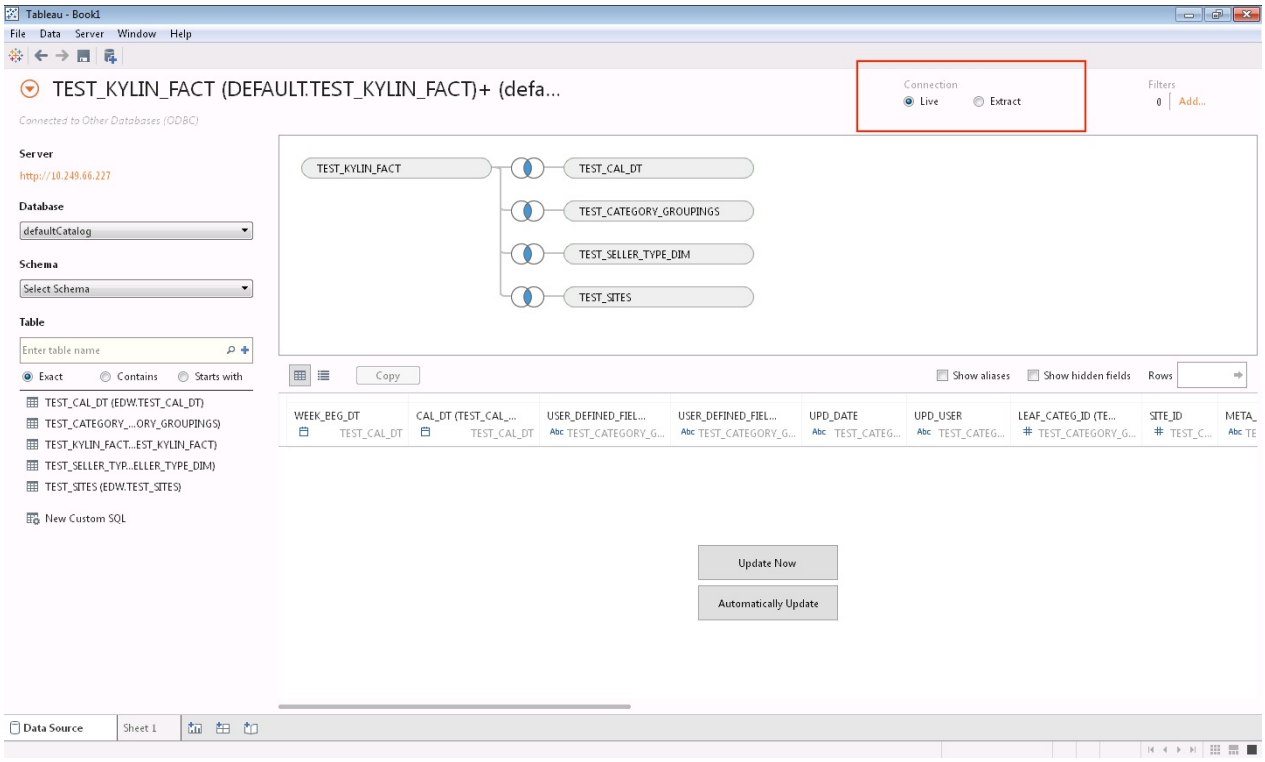
映射数据模型

在左侧的列表中，选择数据库 `defaultCatalog` 并单击"搜索"按钮，将列出所有可查询的表。用鼠标把表拖拽到右侧区域，就可以添加表作为数据源，并创建好表与表的连接关系



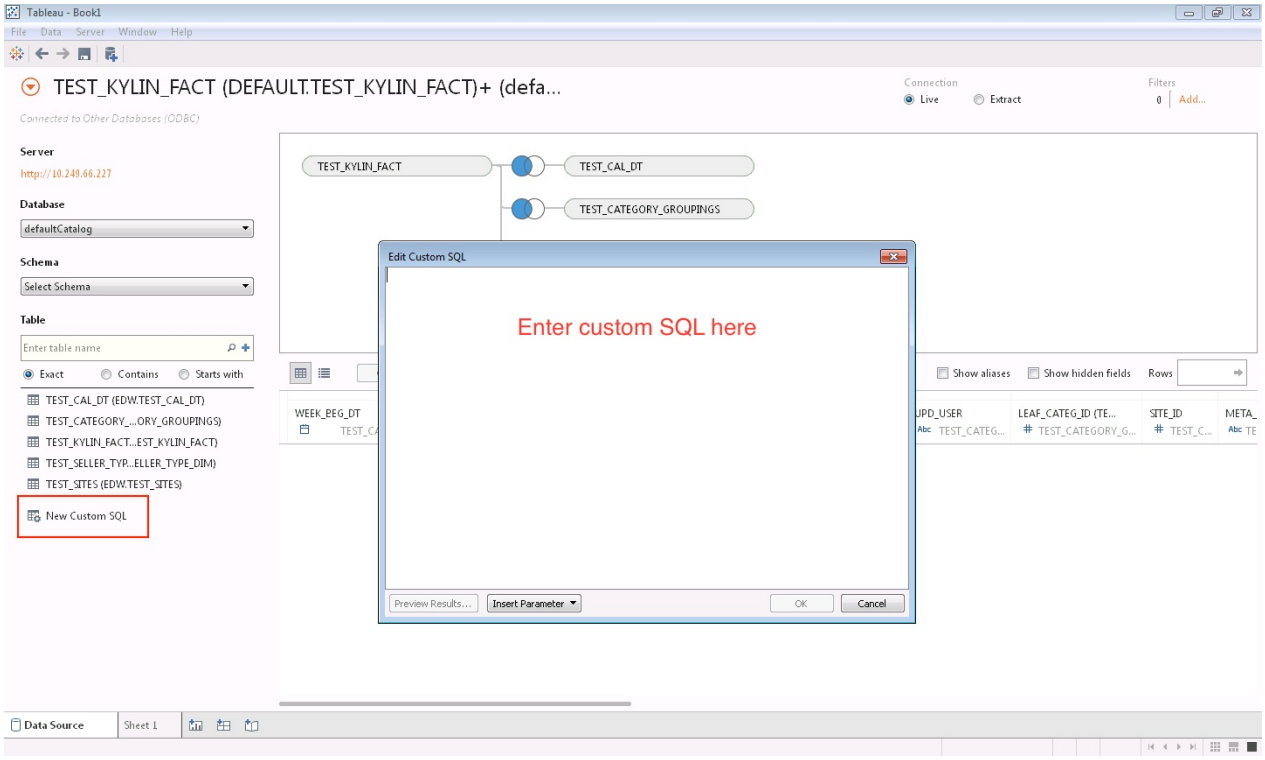
Connect Live

Tableau 9.1中有两种数据源连接类型，选择‘在线‘选项以确保使用'Connect Live'模式



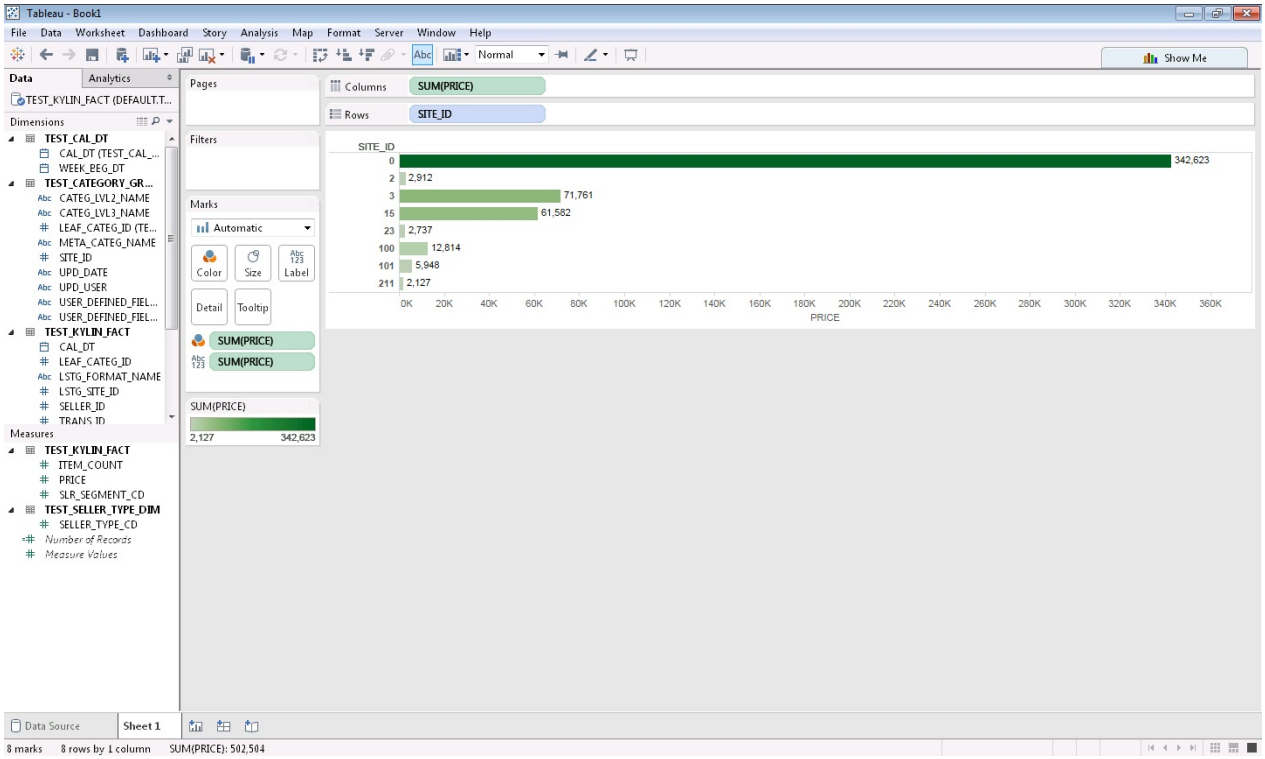
自定义SQL

如果需要使用自定义SQL，可以单击左侧‘New Custom SQL‘并在弹窗中输入SQL语句，就可添加为数据源.



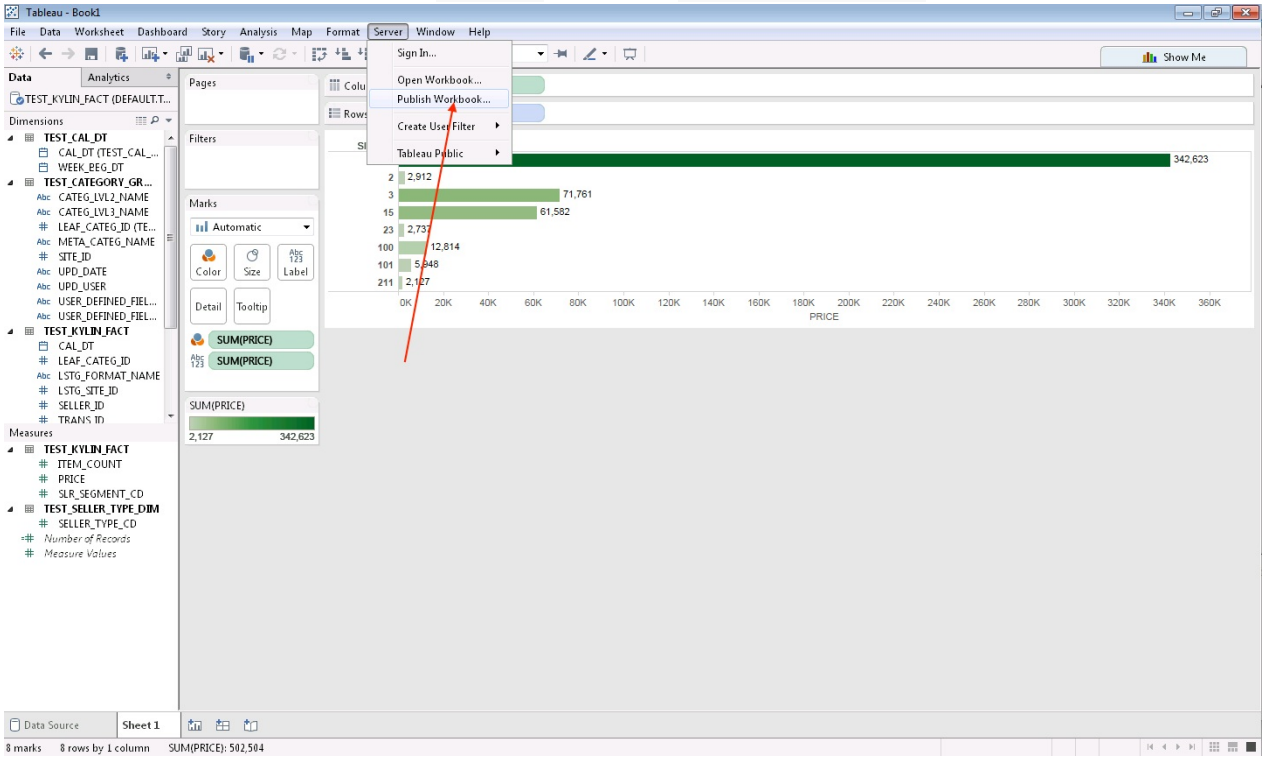
可视化

现在你可以进一步使用Tableau进行可视化分析：



发布到Tableau服务器

如果希望发布到Tableau服务器，点击 `Server` 菜单并选择 `Publish Workbook`

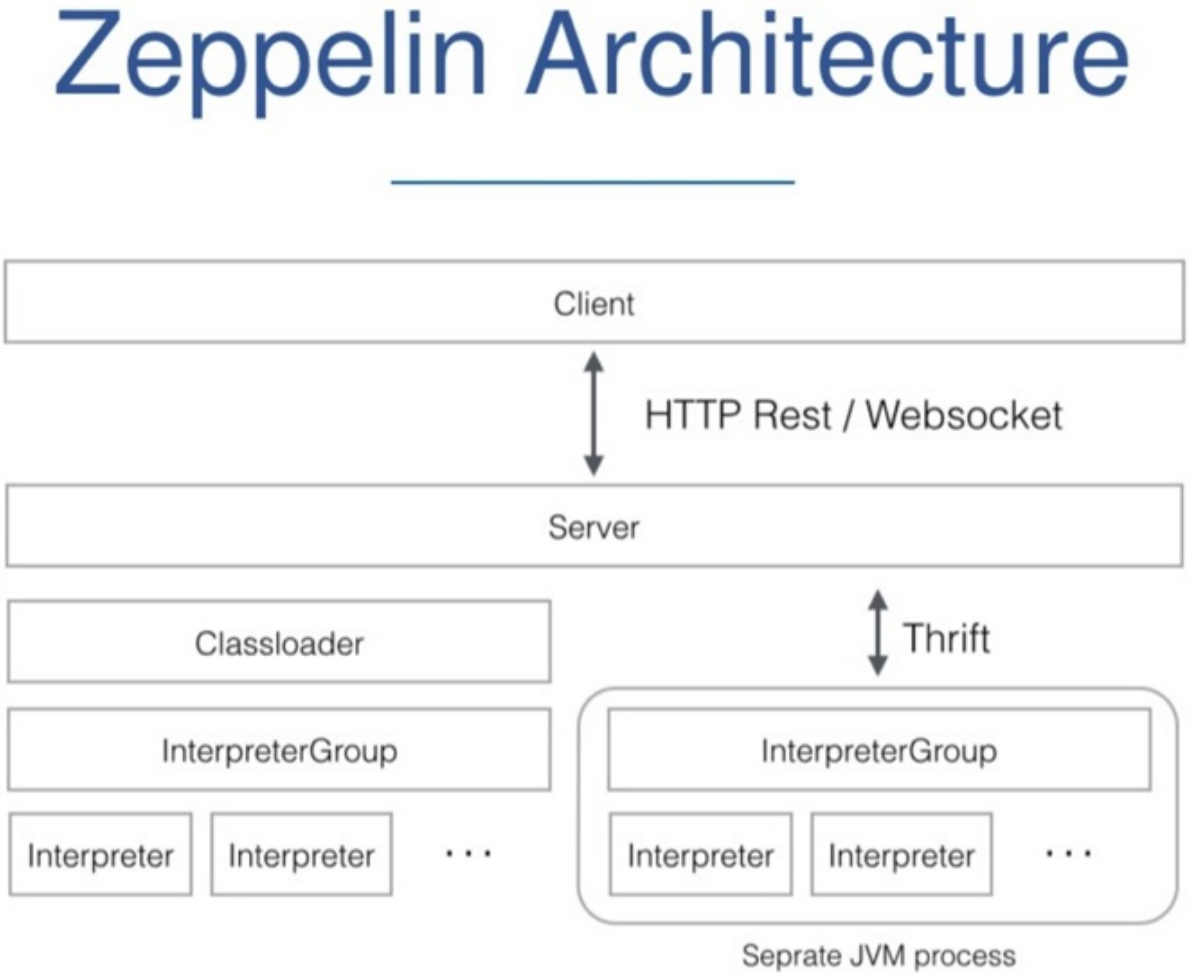


Apache Zepplin 集成

Apache Zeppelin 是一个开源的数据分析平台，为Apache顶级项目。后端以插件形式支持多种数据处理引擎，如Spark, Flink,Lens等，同时提供了notebook式的UI进行可视化相关的操作。KAP对应开发了自己的Zeppelin模块，现已经合并到Zeppelin主分支中，对应在Zeppelin 0.5.6及后续版本中都可以对接使用KAP,通过Zeppelin访问KAP的数据。

Zeppelin架构简介

如下图所示，Zeppelin客户端通过HTTP Rest和Websocket两种方式与服务端交互，在server端Zeppelin支持可插拔的Interpreter(解释器)，以Kyin为例，只需要开发Kylin的Interpreter集成进Zeppelin便可以基于Zeppelin客户端与Kylin服务端进行通信，访问Kylin相关数据。



KylinInterpreter工作原理

KylinInterpreter是构建在AP的Rest API之上的，也是一种典型的使用KAP API的场景，KylinInterpreter读取Zeppelin前端针对KAP的配置的 URL，user，password，查询对应的project，limit，以及offset，ispartial，结合前面所讲的查询Rest API，你可能已经明白，这里主要就是拼接查询请求的参数，用户再在前端页面输入SQL，结合配置的参数，就可以通过HTTP POST方式向KAP发送请求，获取数据。 下面是KylinInterpreter的部分代码，结合注释可以明白KylinInterpreter是如何访问KAP API的。

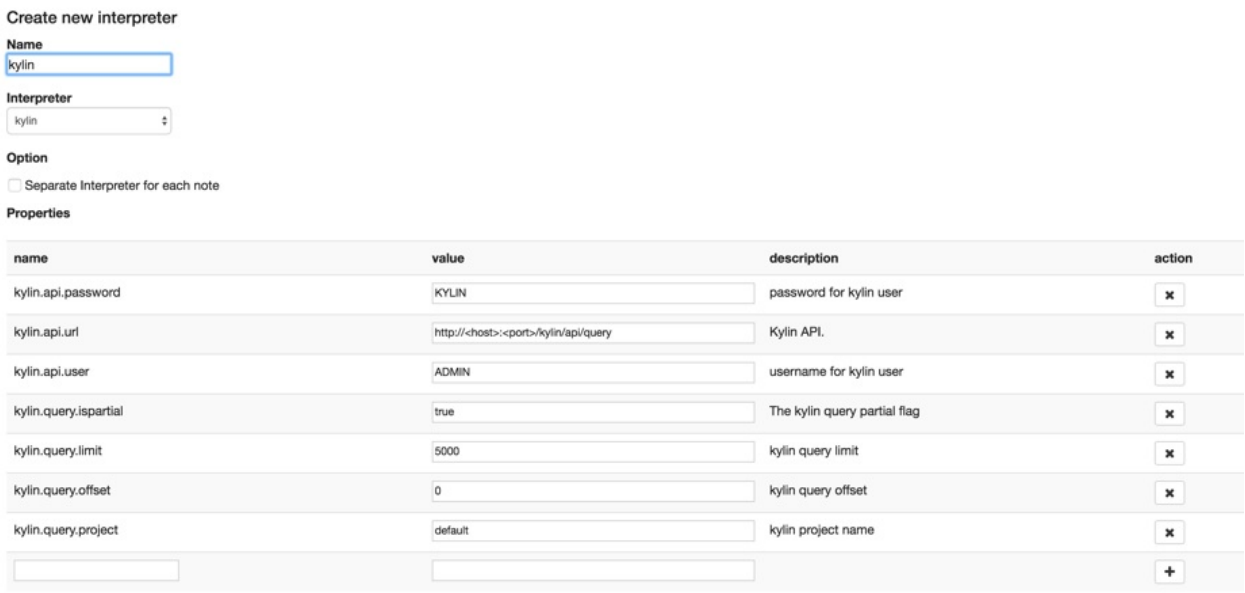
```
  public HttpResponse prepareRequest(String sql) throws IOException {
    String KYLIN_PROJECT = getProperty(KYLIN_QUERY_PROJECT);
……
……
//从配置项中读取帐号密码，并基于Base64进行编码
    byte[] encodeBytes = Base64.encodeBase64(new String(getProperty(KYLIN_USERNAME)
        + ":" + getProperty(KYLIN_PASSWORD)).getBytes("UTF-8"));
    //设置请求参数
    String postContent = new String("{\"project\":" + "\"" + KYLIN_PROJECT + "\""
        + "," + "\"sql\":" + "\"" + sql + "\""
        + "," + "\"acceptPartial\":" + "\"" + getProperty(KYLIN_QUERY_ACCEPT_PARTIAL) +
 "\""
        + "," + "\"offset\":" + "\"" + getProperty(KYLIN_QUERY_OFFSET) + "\""
        + "," + "\"limit\":" + "\"" + getProperty(KYLIN_QUERY_LIMIT) + "\"" + "}");
    postContent = postContent.replaceAll("[\u0000-\u001f]", " ");
    StringEntity entity = new StringEntity(postContent, "UTF-8");
    entity.setContentType("application/json; charset=UTF-8");
    HttpPost postRequest = new HttpPost(getProperty(KYLIN_QUERY_API_URL));
postRequest.setEntity(entity);
//设置请求头信息，加上Basic Authentication信息
    postRequest.addHeader("Authorization", "Basic " + new String(encodeBytes));
    postRequest.addHeader("Accept-Encoding", "UTF-8");

    HttpClient httpClient = HttpClientBuilder.create().build();
    return httpClient.execute(postRequest);
  }
```

Zeppelin的前端有自己的schema，所以KylinInterpreter需要把KAP返回的数据进行适当的转换让Zeppelin前端理解。所以KylinInterpreter主要做的事情就是拼接参数完成向KAP server端段HTTP 请求，然后对返回结果格式化。

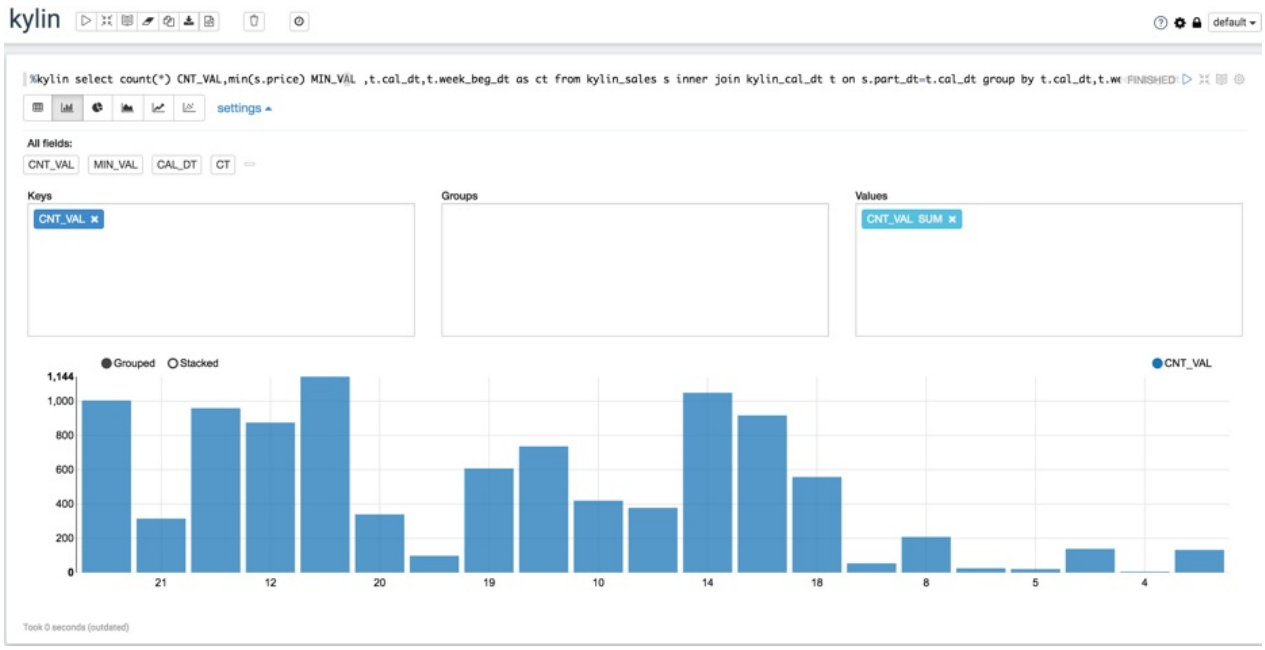
如何使用Zeppelin访问KAP

首先读者需要到Zeppelin官网下载0.5.6或之后的版本的二进制包，按照官网提示配置启动之后打开Zeppelin前端页面（官网有非常详细的介绍，这里就不赘述）。

- 配置Interpreter 打开Zeppelin配置页面，点击‘Interpreter’页面，创建针对KAP某个项目的Interpreter配置，如下图。



- 查询 打开Notebook创建一个新的note，在该note中输入SQL，注意针对KAP的查询需要在SQL前面加上 ‘%kylin’，Zeppelin后端需要知道对应用哪个Interpreter去处理查询。效果如下图，可以拖拽维度和度量灵活获取自己想要的结果。



- Zeppelin发布功能

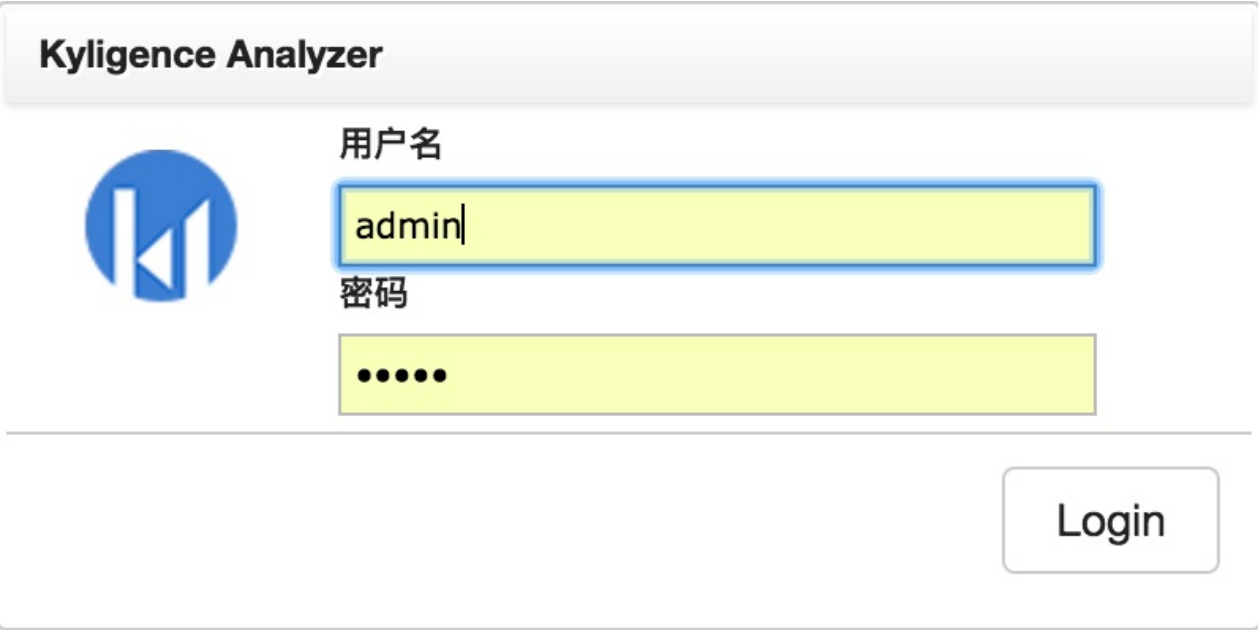
Zeppelin中的任何一个查询，你都可以创建一个链接，并且将该链接分享给其他人。 感兴趣的读者可以到Zeppelin官网 http://zeppelin.apache.org/ 了解更多特性。

> Kyligence公司基于Saiku进行二次开发，提供定制版对KAP友好的访问页面

安装

通过Kyligence公司获得定制版的saiku安装包 KyAnalyzer-{version}.zip,由于Saiku是基于Mondrian转换SQL，因此我们需要获得对应KAP版本的Mondrian包。对应Mondrian包可在 github 上获得。 解压saiku安装包，把mondrian-kylin对应的jar包拷到 saiku-server/tomcat/webapps/saiku/WEB-INF/lib 目录下。

通过saiku-server 目录下的 start-saiku.sh启动saiku，默认端口为8080, 初始默认帐号密码为admin/admin。可通过 http://host:8080 访问页面。



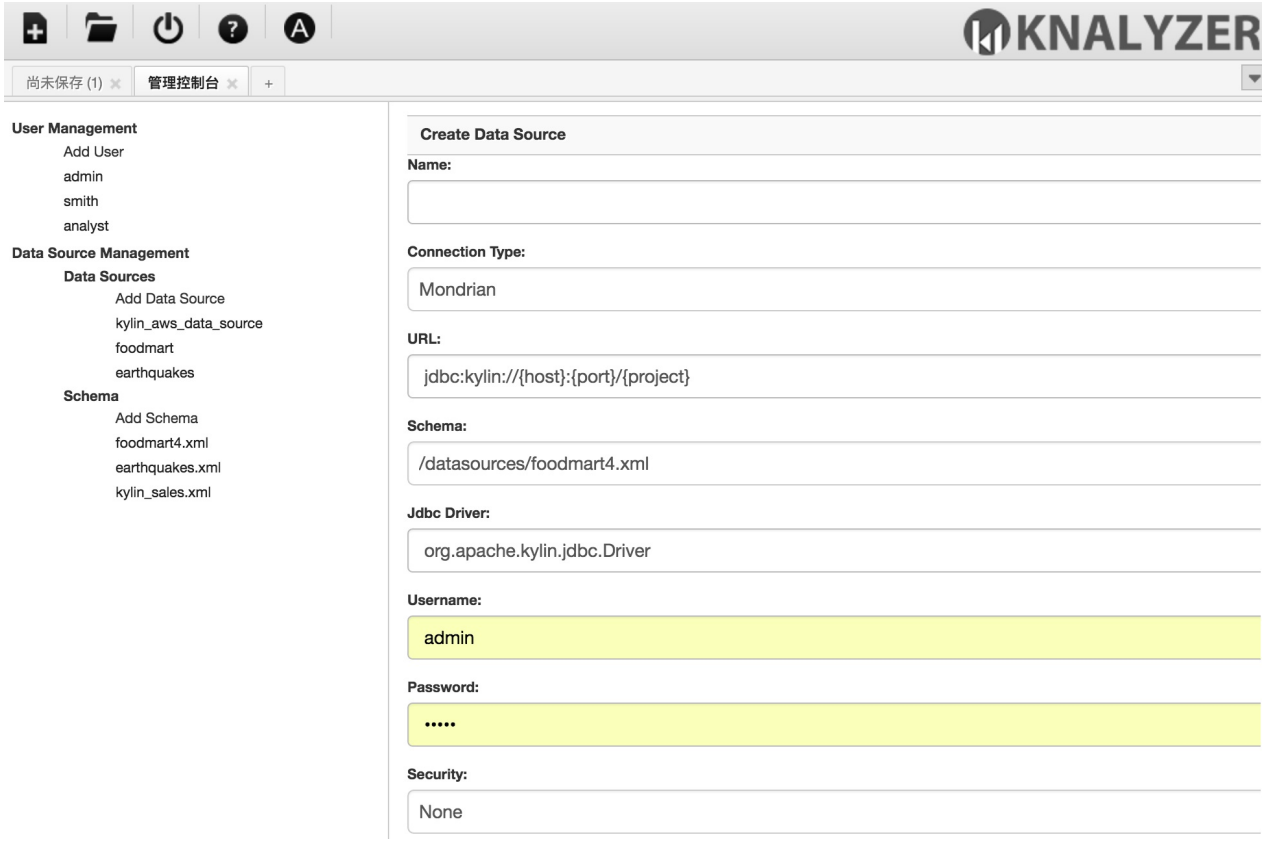
Saiku中的Schema

KAP中的每一Cube都对应saiku中的一个schema文件，因此我们需要创建一个schema文件。 mondrian的schema有各种语法。这里提供官方的文档链接供参考。 请参见下一节了解如何创建Schema文件

创建schema及数据源

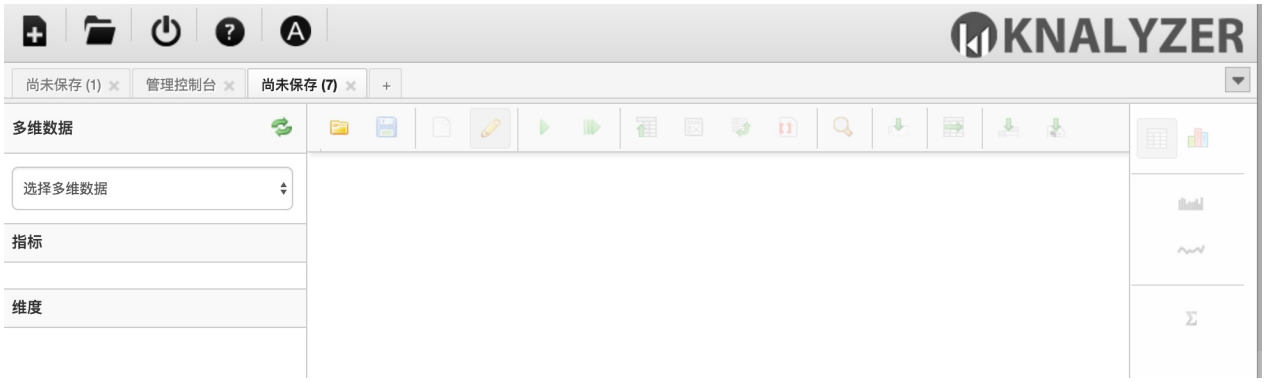
点击导航栏第一排中间含'A'的图标，进入创建数据源页面。点击添加schema (Add Schema), 在该页面上传本地编辑好的schema文件并命名。 点击添加数据源(Add Data Source)按钮，进入创建页面，为saiku创建数据源, saiku访问KAP数据时需要对应的URL，帐号密码等信息。。

- Name — 自定义名称
- Connection Type — 选默认'Mondrian'
- Url - jdbc访问Url
- Schema - 为KAP中Cube建的schema文件
- Jdbc Driver - 驱动类，保留默认值
- Username - 访问KAP用户名
- Password - 访问KAP密码



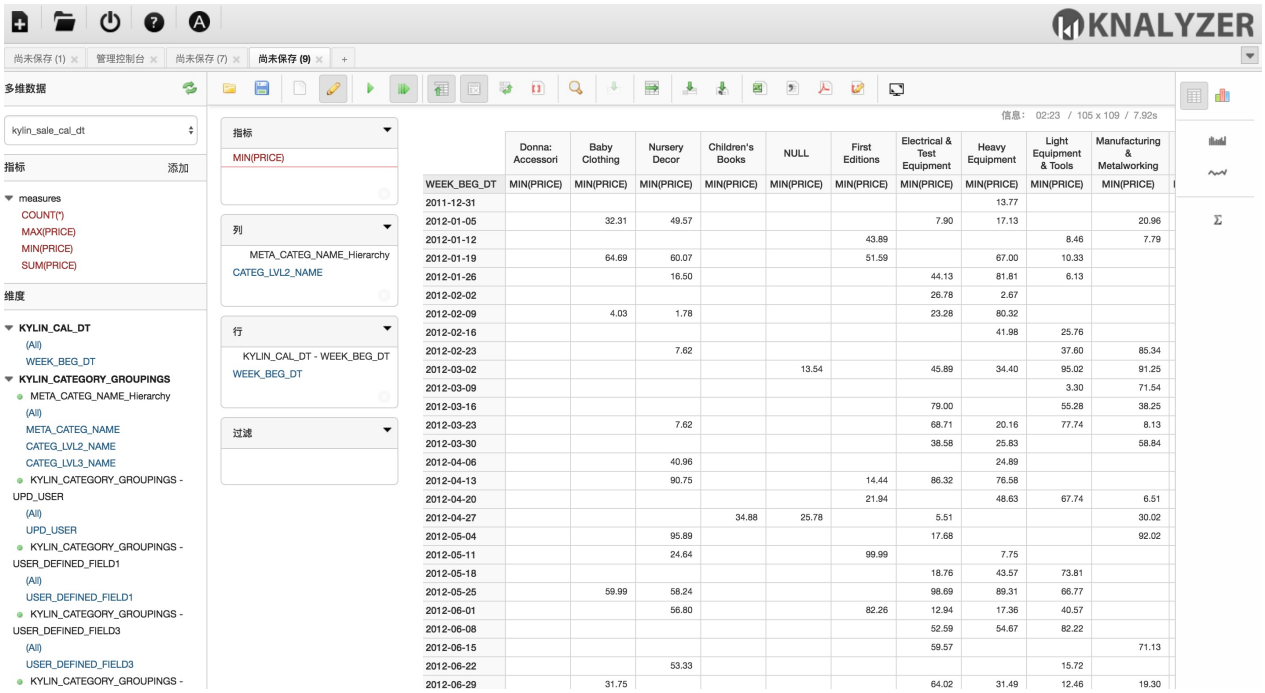
查询页

点击导航栏的新建查询按钮，点击刷新按钮获取最新的数据，在'选择多维数据' 下拉框中选中要查询的Cube，



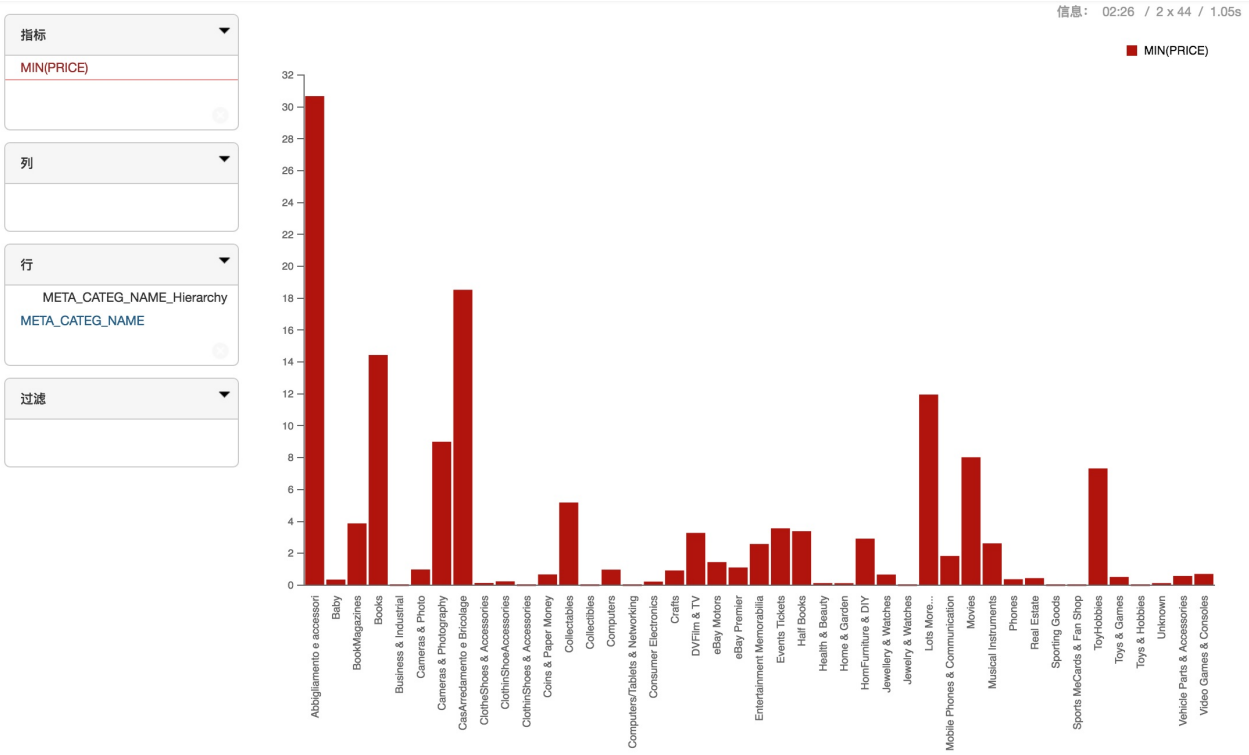
左侧列出维度和度量， 选择自己要查询的数据自由查询。saiku支持多种展现形式，表格，格式图表都可以。

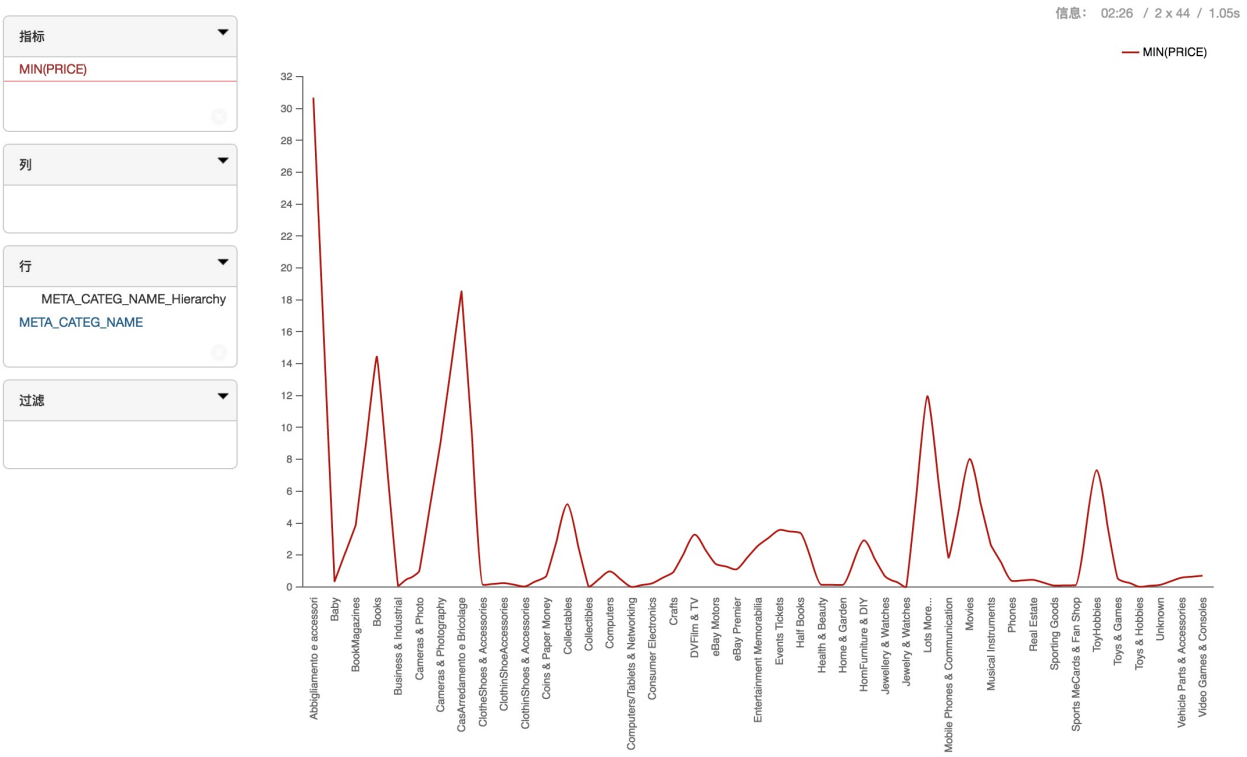
表格



	Donna: Accessori	Baby Clothing	Nursery Decor	Children's Books	NULL	First Editions	Electrical & Test Equipment	Heavy Equipment	Light Equipment & Tools	Manufacturing & Metalworking
WEEK_BEG_DT	MIN(PRICE)	MIN(PRICE)	MIN(PRICE)	MIN(PRICE)	MIN(PRICE)	MIN(PRICE)	MIN(PRICE)	MIN(PRICE)	MIN(PRICE)	MIN(PRICE)
2011-12-31								13.77		
2012-01-05		32.31	49.57				7.90	17.13		20.96
2012-01-12						43.89			8.46	7.79
2012-01-19		64.69	60.07			51.59		67.00	10.33	
2012-01-26			16.50				44.13	81.81	6.13	
2012-02-02							26.78	2.67		
2012-02-09		4.03	1.78				23.28	80.32		
2012-02-16								41.98	25.76	
2012-02-23			7.62						37.60	85.34
2012-03-02					13.54		45.89	34.40	95.02	91.25
2012-03-09									3.30	71.54
2012-03-16							79.00		55.28	38.25
2012-03-23			7.62				68.71	20.16	77.74	8.13
2012-03-30							38.58	25.83		58.84
2012-04-06			40.96					24.89		
2012-04-13			90.75			14.44	86.32	76.58		
2012-04-20						21.94		48.63	67.74	6.51
2012-04-27				34.88	25.78		5.51			30.02
2012-05-04			95.89				17.68			92.02
2012-05-11			24.64			99.99		7.75		
2012-05-18							18.76	43.57	73.81	
2012-05-25		59.99	58.24				98.69	89.31	66.77	
2012-06-01			56.80			82.26	12.94	17.36	40.57	
2012-06-08							52.59	54.67	82.22	
2012-06-15							59.57			71.13
2012-06-22			53.33						15.72	
2012-06-29		31.75					64.02	31.49	12.46	19.30

图形化展示




指标
MIN(PRICE)
列
行
META_CATEG_NAME_Hierarchy
META_CATEG_NAME
过滤
信息： 02:26 / 2 x 44 / 1.05s
MIN(PRICE)
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
0
Abbigliamento e accessori
Baby
BookMagazines
Books
Business & Industrial
Cameras & Photo
Cameras & Photography
CasArredamento e Bricolage
ClotheShoes & Accessories
ClothinShoeAccessories
ClothinShoes & Accessories
Coins & Paper Money
Collectables
Collectibles
Computers
Computers/Tablets & Networking
Consumer Electronics
Crafts
DVFilm & TV
eBay Motors
eBay Premier
Entertainment Memorabilia
Events Tickets
Half Books
Health & Beauty
Home & Garden
HomFurniture & DIY
Jewellery & Watches
Jewelry & Watches
Lots More...
Mobile Phones & Communication
Movies
Musical Instruments
Phones
Real Estate
Sporting Goods
Sports MeCards & Fan Shop
ToyHobbies
Toys & Games
Toys & Hobbies
Unknown
Vehicle Parts & Accessories
Video Games & Consoles

> 下面以KAP发布的样例Cube讲解schema的结构。其中包含了对联合主键，层级(Hierarchy)维度等的支持，供参考。

首先文件格式为xml格式。分为两大块， PhysicalSchema 和 Cube。 PhysicalSchema 定义了表信息， 在里面指明有哪些表即可。

```
<Table name="KYLIN_SALES">
</Table>
```

表定义好后，我们需要定义表中的维度，度量，表与表之间的连接。这些都在标签下定义。

维度

- Dimension 在下定义一个标签， 下定义多个，每一个标签对应一个表,注意Dimension标签有一个key属性，该属性指明了该表的主键，在关联查询时会作为join的条件。如果是联合主键的情况，则不在这里定义，在度量标签中定义。 key属性对应关联一个维度，所以 标签下需要定义一个属性与之关联。
- 联合主键 如果表与表之间是通过多个字段关联，可以在某个Attribute中定义一个key属性，里面指明该表与事实表join的字段。将Dimension的key定义为Attribute的name。如KYLIN_CATEGORY_GROUPINGS 指明key为UPD_USER与对应名为UPD_USER的Attribute关联，该Attribute指明了两个字段，LEAF_CATEG_ID和SITE_ID，那么涉及该维度的查询时，会将事实表的联合主键和这两个字段关联。事实表的主键在下面的度量中会说明。
- Hierarchy 关于Hierarchy可以参考标签Attribute对应的META_CATEG_NAME、CATEG_LVL2_NAME、CATEG_LVL3_NAME的设置，注意Hierarchy结构的列每一个子元素定义Attribute时都会同时定义它的父元素，最终Hierarchies标签会统一列出这几个Attribute。

度量

- MeasureGroup KAP中度量默认都在事实表上，所以这里需要通过table属性关联对应的表。 对应的Measure标签，说明对应的列(可选)和类型(min,max,sum,avg,count等），以及对应的数值类型。
- DimensionLinks 在该标签下说明每一个Dimension与事实表是如何关联的，如果维度本身在事实表上，使用FactLink标签，否则通过ForeignKeyLink，对应foreignKeyColumn指向事实表的关联字段。如果是联合主键，在ForeignKeyLink下定义ForeignKey

下面是一个完整的示例，对应的是KAP提供的样例Cube 'kylin_sales_cube'

```
 <?xml version="1.0" encoding="UTF-8"?>
<Schema name="kylin_sale_hierarchy_name" metamodelVersion="4.0">
   <PhysicalSchema>
      <!-- FACT TABLE -->
      <Table name="KYLIN_SALES">
      </Table>

      <!-- KYLIN_CAL_DT -->
      <Table name="KYLIN_CAL_DT">
      </Table>
      <!-- KYLIN_CATEGORY_GROUPINGS -->
      <Table name="KYLIN_CATEGORY_GROUPINGS">
      </Table>
   </PhysicalSchema>
```

```
   <Cube name="kylin_sale_hierarchy" defaultMeasure="MIN(PRICE)">
      <Dimensions>
         <Dimension name="KYLIN_SALES" table="KYLIN_SALES" key="PART_DT">
            <Attributes>
               <Attribute name="LSTG_FORMAT_NAME" keyColumn="LSTG_FORMAT_NAME"  />
               <Attribute name="PART_DT" keyColumn="PART_DT"  />
               <Attribute name="SELLER_ID" keyColumn="SELLER_ID"  />
               <Attribute name="LEAF_CATEG_ID" keyColumn="LEAF_CATEG_ID"  />
               <Attribute name="LSTG_SITE_ID" keyColumn="LSTG_SITE_ID"  />

            </Attributes>
         </Dimension>
         <Dimension name="KYLIN_CAL_DT" table="KYLIN_CAL_DT"  key="CAL_DT">
            <Attributes>
               <Attribute name="WEEK_BEG_DT" keyColumn="WEEK_BEG_DT"  />
               <Attribute name="CAL_DT" keyColumn="CAL_DT"/>
            </Attributes>
         </Dimension>

         <!--  -->
        <Dimension name='KYLIN_CATEGORY_GROUPINGS' table='KYLIN_CATEGORY_GROUPINGS' key
="UPD_USER">
               <Attributes>
                   <Attribute name='UPD_USER' >
                       <key>
                           <Column name="LEAF_CATEG_ID"/>
                           <Column name="SITE_ID"/>
                       </key>
                       <Name>
                           <Column name='UPD_USER'/>
                       </Name>
                   </Attribute>

                    <Attribute name='USER_DEFINED_FIELD1' >
                         <key>
                             <Column name='USER_DEFINED_FIELD1'/>
                         </key>
                     </Attribute>

                   <Attribute name='USER_DEFINED_FIELD3' >
                         <key>
                             <Column name='USER_DEFINED_FIELD3'/>
                         </key>
                     </Attribute>

                   <Attribute name='UPD_DATE' >
                         <key>
                             <Column name='UPD_DATE'/>
                         </key>
                     </Attribute>

                   <Attribute name='META_CATEG_NAME' hasHierarchy='false'>
```

```
                        <key>
                           <Column name='META_CATEG_NAME'/>
                        </key>
                    </Attribute>

                    <Attribute name='CATEG_LVL2_NAME' hasHierarchy='false'>
                        <key>
                            <Column name='META_CATEG_NAME'/>
                            <Column name='CATEG_LVL2_NAME'/>
                        </key>
                        <Name>
                             <Column name='CATEG_LVL2_NAME'/>
                        </Name>
                    </Attribute>

                    <Attribute name='CATEG_LVL3_NAME' hasHierarchy='false'>
                       <key>
                            <Column name='META_CATEG_NAME'/>
                            <Column name='CATEG_LVL2_NAME'/>
                            <Column name='CATEG_LVL3_NAME'/>
                        </key>
                        <Name>
                            <Column name='CATEG_LVL3_NAME'/>
                        </Name>
                    </Attribute>
                </Attributes>

                <Hierarchies>
                   <Hierarchy name='META_CATEG_NAME_Hierarchy' hasAll='true'>
                     <Level attribute="META_CATEG_NAME"/>
                     <Level attribute="CATEG_LVL2_NAME"/>
                     <Level attribute="CATEG_LVL3_NAME"/>
                   </Hierarchy>
                </Hierarchies>
        </Dimension>

      </Dimensions>
      <MeasureGroups>
         <MeasureGroup name="measures" table="KYLIN_SALES">
            <Measures>
               <Measure name="COUNT(*)" aggregator="count" formatString="#,###" />
               <Measure name="MAX(PRICE)" column="PRICE" aggregator="max" formatString=
"#,###.00" />
               <Measure name="MIN(PRICE)" column="PRICE" aggregator="min" formatString=
"#,###.00" />
               <Measure name="SUM(PRICE)" column="PRICE" aggregator="sum" formatString=
"#,###.00" />
            </Measures>

            <DimensionLinks>
               <FactLink dimension="KYLIN_SALES" />

               <ForeignKeyLink foreignKeyColumn="PART_DT" dimension="KYLIN_CAL_DT" />
```

```
                <ForeignKeyLink dimension='KYLIN_CATEGORY_GROUPINGS' attribute='UPD_USER
'>
                      <ForeignKey>
                          <Column name="LEAF_CATEG_ID"/>
                          <Column name="LSTG_SITE_ID"/>
                      </ForeignKey>
                </ForeignKeyLink>

            </DimensionLinks>

         </MeasureGroup>
      </MeasureGroups>
   </Cube>
</Schema>
```

常见问题

联系我们

欢迎您通过任何方式与我们联系：

商务咨询：info@kyligence.io

加入我们：talent@kyligence.io

产品问题反馈：support@kyligence.io

微信公众号：kyligence

地址：上海市浦东新区亮秀路112号Y1座405室