## 一．大数据和大数据分析是什么？

大数据分析是企业对待数据的一种新形态，而构建数仓与技术体系则是实现大数据分析的解决方案。

面对从工业时代发展到如今的互联网到物联网，数据呈现四个新趋势：

1. 数据量巨大以及数据类型的多样化

从互联网发展开始，互联网网站到移动app、微信小程序等多种互联网服务的产生，以及越来越多的人在使用互联网服务，人们在网上每一个动作都会产生数据被互联网公司储存下来，并作为分析素材，互联网产生的数据一直呈现指数级递增；而物联网的出现，使得大量的工业物联网感知前端、智能家居、可穿戴设备、以及如现代汽车、公交车地铁刷卡、城市摄像头等等具产生数据，并通过蓝牙/NFC连接手机后间接传到数据中心的数据，或直接将数据通过互联网或专用物联网网络汇总而来的数据越来越大，人们始终围绕着存储-查询-分析决策三层递进需求使用数据，因此产生数据的源头从互联网到各种各样的物联网设备，越来越多；相对应的数据量已经从之前的一个互联网公司一年产生TB级别发展到，一家不出名的手机游戏公司，如世界争霸，每天产生50TB数据，数据量已经到达了PB为基本单位的规模。

1. “大数据”价值密度低

数据分析的定义是：基于商业目的，有目的的进行收集、整理、加工和分析，提炼有价值信息的过程。

庞大的数据量下，如果确认了一个目的，那么其中有助于实现这个目的的数据可能万不存一，即便存在，也极其分散。

因此导致数据分析的第一步是对数据进行ETL操作，即围绕目的，进行数据的抽取、清洗、转化。尽可能的将数据分析的范围缩小，以期待提升数据分析效率与成本。

1. 数据变现要求数据分析的速度要足够快

数据分析的目的，是将分析结果，围绕业务目的提供决策辅助。决策分为运营决策与系统决策两种。

运营决策：在海南智慧指挥中心的大屏上，实时显示了当前城市的人流、车流、进出港人/车/货流量、行政处理事件、反恐事件、应急处理事件，这是典型的运营性决策，通过数据反映的情况，我们可以及时的触发各种资源的调配，比如红绿灯、医疗、应急、公安等力量，这些力量我们以解决问题的出发点，统称为资源。

系统性决策：以支付宝举例子，大家每做一次交易或转账，这都是一个敏感事件的发生，支付宝作为一个服务这个事件的服务商，必须对每一个敏感事件都要做出一个风险决策，比如你的支付宝账号登录地址与设备是否异常，你的转账对象与你之间的关系是否存在风险，转账对象本身是否存在风险，你转账或交易金额是否构成风险等等，这个风险的评判就是系统发起的实时性决策，这种决策对数据的实时性要求非常之高。但说到底，系统决策仍然是围绕存储、检索、分析三个基本需求。

既然要用大数据分析，那就不要凑合，分析出的决策不仅要尽量准确，而且一定要快。尤其是对大型企业或政府用户，他们面对的运营对象不仅范围广、体量大，而且其运营对象所处的环境也是瞬息万变，一个错误决策或滞后决策带来的恶劣影响，是以巨大的金钱甚至老百姓的生命作为代价的。

所以仅仅是离线数据分析尽可能缩短时间，实时数据分析尽可能决策复杂度和精确度再提升一点后，再提升一点。这是这个快时代对大数据的另一层要求。

因此，数据存储、查询、分析工具层出不穷的随着数据市场变化而产生

承接上面所面临的数据类型众多（结构化、半结构化、非结构化）、PB起步的基本规模、低价值数据密度、，想实现数据存储，尤其是查询甚至数据分析，使得数据变现层面的需求，由于传统数据库无法实现，从而在数据库的基础上，演化相对应的作为一个完整的数据变现过程，如数据同步集成工具——以查询与分析为核心的OLAP数据仓库或数据湖——hadoop、Spark、Hive或Flink、Streaming等离线/实时数据计算技术——数据质量检测平台与数据可视化工具，一条完整的数据使用链路。

如同数据仓库泰斗Kimball所说“我们用了20年时间把数据放入数据库中，现在该是拿出来使用的时候了”

## 二．作为一个架构师怎样学习大数据

ACP大数据学习方向分为大数据工程师与数据分析师，这两个对于咱们没接触过大数据行业的人来说，不太好选择，也不太清楚两者的区别，因此我拿2021年2月份阿里集团社会招聘的岗位要求来看：

大数据工程师：

职位描述

1、参与数据中台建设，负责承担数据资产建设，通过将业务数字化，赋能业务，提升数据价值；

2、深入理解业务、产品，制定数据资产架构的规划建设，包括数据采集、资产管理、数据产品、数据质量及稳定性保障体系等；

3、深度参与到BU层面业务的重点项目和战役，通过数据技术和数据产品给业务带来增量；

4、能为团队引入创新的技术和方案，用创新的思路解决问题，能对现存或未来系统进行宏观的思考，规划形成统一的框架、平台或组件。

职位要求

1、计算机或相关专业本科及以上学历，从事数据仓库领域工作至少2年以上，熟悉数据仓库模型设计方法论，并有实际模型设计及ETL开发经验；

2、有从事分布式数据存储与计算平台应用开发经验，熟悉Hadoop生态相关技术并有相关实践经验，包括Hdfs、Mapreduce、Hive、Hbase等；

3、熟悉数据仓库领域知识和技能，包括但不局限于：元数据管理、数据开发测试工具与方法、数据质量、主数据管理；

4、了解实时计算，有使用过Storm、Spark Streaming、Flink等开源实时流引擎的优先；

5、有一定系统架构设计经验，熟悉常见的技术架构，能够将技术与业务做很好的结合；

6、对数据技术有很强的兴趣，不断自我学习，对新技术热衷热爱；

7、良好的思维逻辑性、语言表达能力；有较好英语口语能力尤佳；强烈的自我驱动力。

大数据分析师：

职位描述

1、与业务管理团队紧密合作，通过分析/挖掘数据，探索业务机会点，形成分析报告，贡献自己对业务的独特见解；

2、运营数据收集整理，监控相关核心数据，查找、分析运营过程中的异常情况；

3、通过对服务渠道、服务流程、服务资源配置等关键环节进行分析，保障服务链路的顺畅、服务品质的稳定，提升用户满意度；

4、能够从用户的角度思考问题，通过数据分析，找出产品痛点，完善用户体验；

5、能够独立推动数据项目，为客户体验及权益保障中心运营决策、服务方向、策略提供数据决策支持；

6、探索数据技术创新的机会，让数据在业务中发挥更大的价值。

职位要求

1、五年以上产品/流程/用户/决策分析经验，熟悉互联网常见产品分析，用户分析方法论，比如对用户画像、分层或者产品体验分析模型建立等有一定经验；

2、熟练使用BI报表制作工具、SQL、Excel、R语言及Python等分析工具；

3、有良好的业务嗅觉，能快速学习新业务领域并给出分析洞察 ；

4、具备较强跨团队协作和数据项目推动能力；

5、良好的沟通交流能力与分析报告撰写能力。

大数据工程师与分析师，在刚一开始都会从，都需要学习数仓构建与优化流程与原理，数据建模与数据处理的基本指令，以及具体的大数据产品集的学习与使用入门，但在长远学习后，我们去应聘是面对的岗位描述和要求看，大数据工程师会专职于根据用户业务设计与搭建整个大数据分析系统，岗位需要有一定系统架构设计经验，熟悉常见的技术架构，能够将技术与业务做很好的结合，我认为这更像是大数据架构师；分析师则是数据的使用者，通过使用数据工程师构建的技术体系，去收集与思考业务需求、数据收集与统计、数据建模、数据开发与处理这些实际数据开发工作。

作为架构师的学习角度，毫无疑问，是向ACP大数据工程师去学和考。

我嘴里的架构师是什么意思？

如果我们学习的目的是毕业生或跨行进入阿里云做销售或解决方案人员，我认为需要有独立写解决方案的能力，这种能力，模仿ACP和ACE的称呼，是全栈云架构师，需要我们同时具备云计算、企业级互联网架构（业务中台）、大数据和物联网的能力集合。我自己的学习目标也是成为全栈云架构师，或者通俗点是解决方案工程师，放在销售岗位叫专家/顾问型销售。

如果作为全栈架构师，在大数据版块的学习时，首先要思考我们要达到什么学习效果，我自己，是想达到面对一个企业或政府用户数据有汇总、分析与变现的需求时，结合户真实的业务场景和数据现状、用户未来发展规划和支持力度，在用户各种条件制约下，可以设计一套/两套解决大数据分析技术体系甚至数据中台的解决方案，并且这个解决方案中的产品组合是精简、高效、顺畅的，与用户现有的业务与需求是高度匹配的，不仅如此还能跟用户解释一下解决方案中各个产品是代替传统/开源什么产品的，它的功能和优势是什么，实际用起来大概是个什么操作流程，其中的逻辑和带来的便捷是什么。

这就够了，但这很难。

我自己在考下云计算方向的ACP和ACE认证后，曾经对ACP-大数据视频教程看了不下三遍，花了这么多时间用心去学，还是感觉没入门。我想了很久，造成这种结果的根源就是大数据与云计算没什么关系(大数据是一门数据分析学科)，它相对于我学过的云计算而言，是一个独立的新学科，我没有第一时间站在更高层次的视角对大数据分析中，整体业务流程设计与整体技术平台搭建去学习和思考过，而是跟着ACP视频一头扎进阿里云大数据产品的使用细节中，扎进去时痛苦，拔出来时迷茫，就是我之前的学习之路。

我揣测，ACP大数据教学视频的交授对象是大数据的从业者，而不是我这样的跨行业学习大数据的新人小白。如果你学过大数据ACP教程，会强烈感觉，这套视频课程对我们这些跨行学习的想从事云行业的人不友好，没有入门篇，比如大数据理论基础、数仓构建与优化流程、阿里云整个大数据产品概览这些东西。ACP教程的老师认为我们已经懂了，所以草草带了一下什么是大数据和数仓后，直接扎进了3个大数据产品的实际使用，尤其是数据开发的汪洋大海中。看到我是一脸蒙圈。

如果有人像我一样是大数据的小白，那一定要对学院派理论要沉下心去阅读和理解，大数据基本概念和基本流程尽在其中，这是我们的思维公式，有了公式，面对复杂多变的业务场景我们才能迎刃而解。

目前ACP和ACE的学习，全网缺乏学习资料，能查到的，都是一些高屋建瓴的学习指导，落不到实处。因此，我在我学习中，将阿里云大数据产品帮助文档中内容作为基础，进行了精简、关键词标注与重新排列，并将开发者论坛、云栖大会和外部资料，以及我个人见解与总结补充其中，并对所有内容中出现的陌生词汇、技术、概念做了解释，形成一份我自己的学习笔记，我想将它分享出去，方便像我一样的新人小白，初次接触云计算的人，能够顺利的学懂、学明白大数据产品。

我只是云学习路上躬身学习的一名中年学生，内容中一定有错误的东西，请大家骂的同时，能将错误指出来，我会立刻改正。

我分享这些出来只想作为大家学习路上大家互相印证、互相学习的材料，我分享的心态是做一个云学习路上的中年课代表，仅此而已。

我的学习路径通常是先对一个产品族的全览，明白它们之间的关系与要实现的整体目的后，再对单个产品进行学习，围绕六个疑问去学，分别是“它是什么”、“它是干什么的”、“哪些场景会用到它“、”这些场景中它通常与哪些产品组合形成解决方案”、“它对比同样功能的传统/开源产品优势在哪儿”、“它的使用流程是什么和功能间的使用逻辑是什么”。

我的学习规划是对这些文档最多读两遍，第一遍，是顺着文档溜达一圈，知道每个产品是什么，是做什么的，能够与哪些产品配合使用，用在什么场景；第二遍是深入的看一遍，再横向对比它与开源/传统产品的优势和差异，以及其产生背景，和支撑它实现的技术内核，并对它具体的使用步骤和使用逻辑有个框架性的认知。

下面，我会陪着同为跨行学习大数据的同学们，一路走下去，将学习步骤分为①学院派理论学习；②实战中数仓搭建与使用；③阿里云具体产品的学习，三步。

如果你愿意，请也陪我一路学下去。

## 三．数仓构建与优化的完整流程（学院派理论学习）

一个数仓的建设是从①需求调研-②架构与模型设计-③项目分配与安全-④建立性能基准-⑤数仓性能优化-⑥结果验证的闭环。

数据分析是一个学科，是学科就需要有通用的词语统一共识。因此在从①到⑥的详细分解中，会在每一步的最开始，将基本名词解释作为首要内容，方便对更详细的文档内容进行阅读。

### 数仓构建流程总览

#### 基本名词解释：（我终于理解，大学微积分老师为什么第一节课那么注重基本概念的名词解释了，这些基本概念，第一遍看，可能看不太明白，但一定要记住这些名词，建立自己的初步印象，不然后面所有的学习内容你真的看不懂，你会在茫然呆坐中口吐芳华和揪头发。有了强烈的学习欲望，这些东西你一定能看下去，还能边看边思索形成自己的初步思索框架，后面的文档中这些词汇会高频重复出现，到时候你会有更清晰的概念。）

**数仓规划：**包括逻辑空间与物理空间的全局架构，以实现业务划分、资源管控、项目管理等目的。

**逻辑空间：**基于业务特征划分命名空间及其核心对象，包括业务板块、数据域和公共定义模块。

**物理空间：**基于开发协作管理需求划分物理项目，包括项目管理、数据源管理等模块。

**业务板块：**业务板块是逻辑空间的重要组成部分，是基于业务特征划分的命名空间。在同一个业务板块中可能包含多个不同的项目，业务板块与项目的关系为1:N。

**数据域：**用于存放同一业务板块内不同意义的指标，如商品域、交易域、会员域等。

**业务板块参数：**板块参数用于定义业务板块内的时间分区和业务日期

**业务过程：**是业务活动中不可拆分的事件，例如下单、支付和退款。创建业务过程，即从顶层视角，规范业务活动中事件的内容类型及唯一性。

**业务限定**：统计的业务范围，筛选出符合业务规则的记录（类似于SQL中where后的条件，不包括时间区间）

**项目：**项目是一种物理空间上的划分，便于用户在数据中台建设过程中对物理资源及开发人员进行隔离化管理。一个业务板块可以包含多个项目，每个系统成员可以加入多个不同的项目。

**维度**：维度建模由Kimball（这个名字上文中有提到过，是不是很熟悉， Ralph Kimbal大神和Bill Inmon大神分别提出Kimball和Inmon是两种主流的数据仓库方法论，在实际数据仓库建设中,业界往往会相互借鉴使用两种开发模式）提出。维度模型主张从分析决策的需求出发构建模型，为分析需求服务。维度是人们观察事物的角度，是确定事物的多方位、多角度、多层次的条件和概念，在数据开发中，维度是我们观察业务的角度，用来反映业务的一类属性。属性的集合构成维度，维度也可以称为实体对象。例如，在分析交易过程时，可以通过买家、卖家、商品和时间等维度描述交易发生的环境。

**属性（维度属性）**：维度所包含的表示维度的列称为维度属性。维度属性是查询约束条件、分组和报表标签生成的基本来源，是数据易用性的关键。

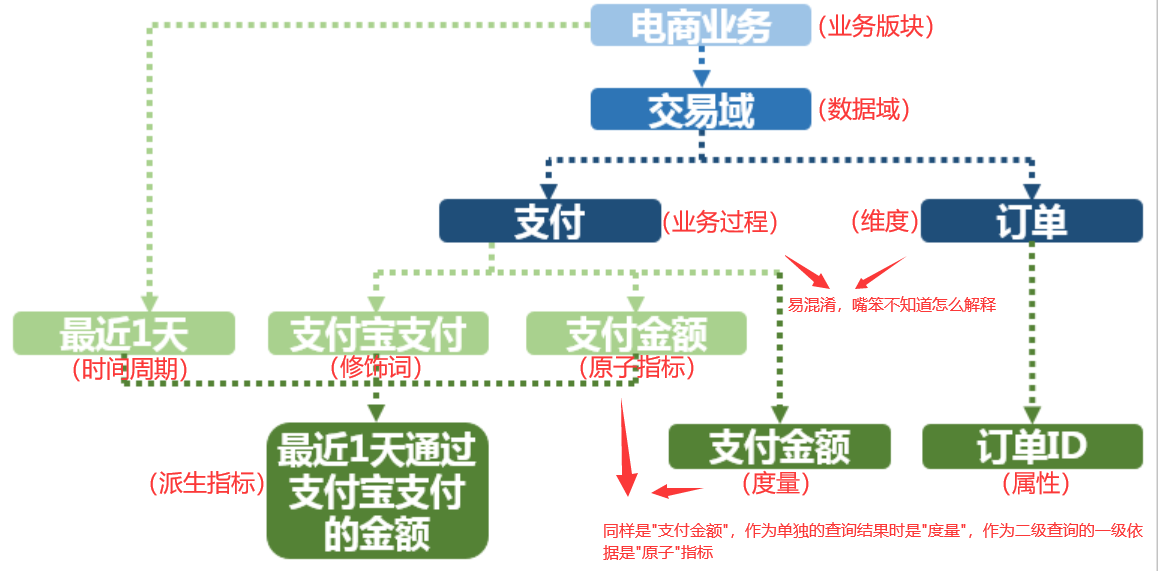
**度量**：在维度建模中，将度量称为事实，将环境描述为维度，维度是用于分析事实所需要的多样环境。度量通常为数值型数据，作为事实逻辑表的事实。

**指标**：指标分为原子指标和派生指标。原子指标是基于某一业务事件行为下的度量，是业务定义中不可再拆分的指标，是具有明确业务含义的名词，体现明确的业务统计口径和计算逻辑，例如支付金额。

1. 原子指标=业务过程+度量。
2. 派生指标=时间周期+修饰词+原子指标，派生指标可以理解为对原子指标业务统计范围的圈定。

**统计周期**：统计的时间范围，例如最近一天，最近30天等（类似于SQL中where后的时间条件）。

**统计粒度**：统计分析的对象或视角，定义数据需要汇总的程度，可理解为聚合运算时的分组条件（类似于SQL中的group by的对象）。粒度是维度的一个组合，指明的统计范围。例如，某个指标是某个卖家在某个省份的成交额，则粒度就是卖家、地区这两个维度的组合。如果需要统计全表的数据，则粒度为全表。在指定粒度时，需要充分考虑到业务和维度的关系。统计粒度常作为派生指标的修饰词而存在。



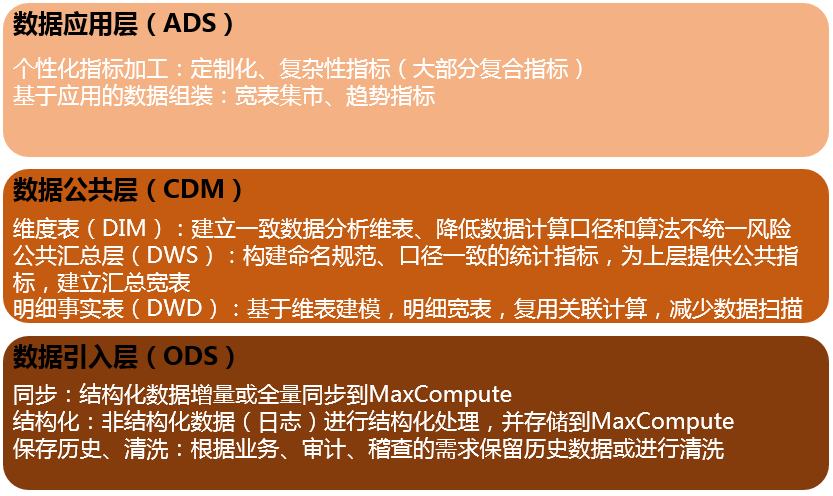
**数仓分层：**在阿里巴巴在大数据分析应用中，经过一代代数据科学家的多年摸索和一代代演进，最终阿里的数据体系中，将数据仓库分为三层，自下而上为：**数据引入层**（**ODS**，Operation Data Store）、**数据公共层**（**CDM**，Common Data Model）和**数据应用层**（**ADS**，Application Data Service）。

**数据引入层ODS**（Operation Data Store）：存放未经过处理的原始数据至数据仓库系统，结构上与源系统保持一致，是数据仓库的数据准备区。主要完成基础数据引入到MaxCompute的职责，同时记录基础数据的历史变化。

**数据公共层CDM**（Common Data Model，又称通用数据模型层），**包括DIM维度表**、**DWD和DWS**，由ODS层数据加工而成。主要完成数据加工与整合，建立一致性的维度，构建可复用的面向分析和统计的明细事实表，以及汇总公共粒度的指标。

* **公共维度层（DIM）：基于维度建模理念思想，建立整个企业的一致性维度。降低数据计算口径和算法不统一风险。公共维度层的表通常也被称为逻辑维度表，维度和维度逻辑表通常一一对应。**
* **公共汇总粒度事实层（DWS）：以分析的主题对象作为建模驱动，基于上层的应用和产品的指标需求，构建公共粒度的汇总指标事实表，以宽表化手段物理化模型。构建命名规范、口径一致的统计指标，为上层提供公共指标，建立汇总宽表、明细事实表。公共汇总粒度事实层的表通常也被称为汇总逻辑表，用于存放派生指标数据。**
* **明细粒度事实层（DWD）：以业务过程作为建模驱动，基于每个具体的业务过程特点，构建最细粒度的明细层事实表。可以结合企业的数据使用特点，将明细事实表的某些重要维度属性字段做适当冗余，即宽表化处理。明细粒度事实层的表通常也被称为逻辑事实表。**

数据应用层ADS（Application Data Service）：存放数据产品个性化的统计指标数据。根据CDM与ODS层加工生成。



### 1.业务调研

一个完整的业务调研分为：①确定需求—②分析业务过程—③划分数据维度—④定义维度与构建总线矩阵—⑤明确统计指标

#### 1.1确定需求

在构建数据仓库之前，首先需要知道用户的组织架构，以及组织中都有谁会参与和使用到数据仓库，了解到不同部门所扮演的角色，以及不同部门对数据的具体需求。

只有深入到用户中与不同角色和部门的调研和对话，才能了解到真实的业务需求，以及确定建设数仓要解决的具体问题。

这是数据仓库建设的起点，决定了最终建成的数据库是否能真的为用户带来价值；同样也是我们从“一套方案走天下”的初级架构师，成长为能为用户贴身定制个性化方案的中级架构师的必经之路。

1. 以**部门**为调研对象，了解部门现状与部门需求：

可以通过调查表和访谈等形式详细了解以下信息：

**1.用户的组织架构和分工界面。**

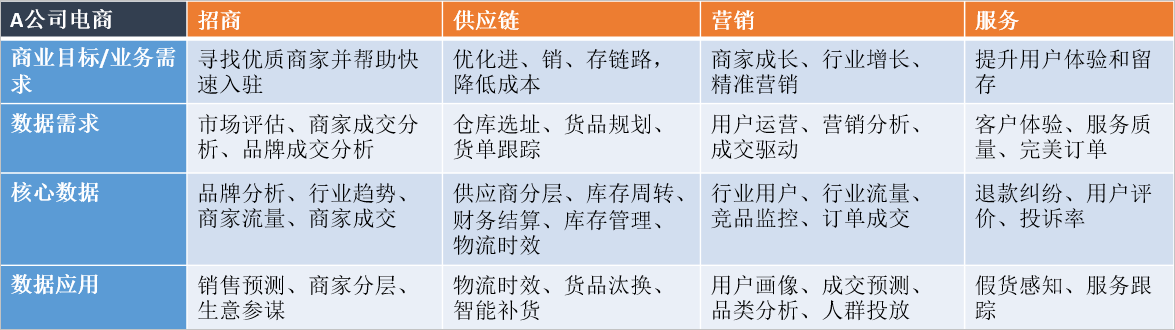
例如，用户的同一个部门可能分为数据分析、运营和维护多个角色人员，各个角色对数据仓库的需求不同，需要对不同角色分别进行调研。

**2.用户的整体业务架构，各个业务部门之间的联系和信息流动的流程。**

需要梳理出整体的业务数据框架。

**3.各个已有的业务部门的主要功能及获取的数据。**

以A公司的电商业务为例，梳理出业务数据框架（列为部门维度，行为需求维度）。A公司的电商业务板块分为招商、供应链、营销和服务四个大部门，每个部门的需求和数据应用都不同



此外，还需要进一步了解各业务部门中已有的数据功能模块。数据功能模块通常和业务板块紧耦合，对应一个或多个表，**可以作为构建数据仓库的数据源**。

下表展现的是**营销部门**的数据功能模块。



（Y代表包含该数据功能模块，N代表不包含。）

1. 以部门中关键数据使用**人**为调研对象，对需求进行进一步明确与分析：

完成各部门现状与部门需求调研后，需要**进一步收集**用户各部门数据分析师和业务运营人员这些**数据使用人的需求**，进而对需求进行深度的思考和分析。

需求分析的途径有两种：

1.根据与分析师和业务运营人员的沟通获知需求。

2.对报表系统中现有的报表进行研究分析。

在需求分析阶段，需要沉淀出业务分析或报表中的指标，以及指标的定义和粒度。粒度可以作为维度的输入。

建议思考下列问题，对后续的数据建模将有巨大的帮助：

**※业务数据是根据什么（维度、粒度）汇总的，衡量标准是什么？**例如，成交量是维度，订单数是成交量的度量。

**※明细数据层和汇总数据层应该如何设计？公共维度层该如何设计？是否有公共的指标？**

**※数据是否需要冗余或沉淀到汇总数据层中？**

（关于公共维度层DIM/公共汇总事实层DWS/明细实施层DWD的概念直接下翻看2.2 数据分层章节）

**从业务侧重点分析：**

如果我们设计数仓的用户是电商营销部门，而营销部门中数据使用人是营销数据分析师。数据需求为最近一天某个类目（例如，厨具）商品在各省的销售总额、该类目Top10销售额商品名称和各省客户购买力分布（人均消费额）等，用于营销分析。最终的业务需求是通过营销分析完成该类目的精准营销，提升销售总额。通过业务调研，我们将着力分析营销业务板块的交易订单数据功能模块。

**从数据建模分析：**

如果我们设计数仓的用户是电商营销部门，而营销部门中数据使用人是营销数据分析师。需要了解A公司电商业务中厨具类目的成交金额。当获知这个需求后，我们需要分析：根据什么（维度）汇总、汇总什么（度量）以及汇总的范围多大（粒度）。例如，类目是维度，金额是度量，范围是全表。此外，还需要思考明细数据和汇总数据应该如何设计、是否是公共层的报表及数据是否需要沉淀到汇总表中等因素。

**最终，确定需求这一步的分析产出通常是记录原子与派生指标的文档**。

#### 1.2分析业务过程（初步形成DIM/DWD表想法，以及决定数据粒度）

业务过程可以概括为一个个不可拆分的行为事件。用户的业务系统中，通过埋点或日常积累，通常已经获取了充足的业务数据。**为理清数据之间的逻辑关系和流向**，首先需要理解用户的业务过程，了解过程中涉及到的数据系统。

可以采用过程分析法，将整个业务过程涉及的每个环节一一列清楚，包括技术、数据、系统环境等。在分析企业的工作职责范围（部门）后，我们也可以借助工具通过逆向工程抽取业务系统的真实模型。可以参考业务规划设计文档以及业务运行（开发、设计、变更等）相关文档，全面分析数据仓库涉及的源系统及业务管理系统：

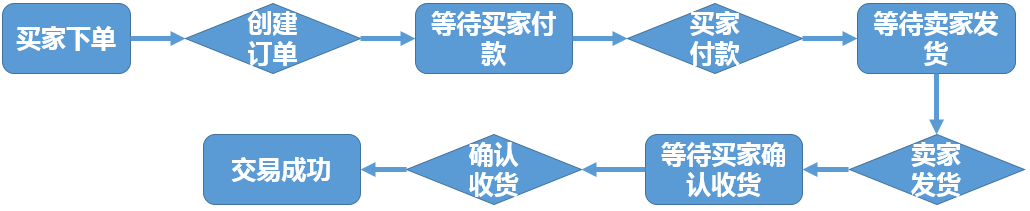
1. **每个业务会生成哪些种类数据，不同种类数据存在于什么数据库中**。
2. 对每个业务过程进行分解，了解过程中的**每一个环节会产生的具体数据内容是什么**。
3. 数据在**什么情况下会更新**，更新的逻辑是什么。

业务过程可以是单个业务事件，例如交易的支付、退款等；也可以是某个事件的状态，例如当前的账户余额等；还可以是一系列相关业务事件组成的业务流程。具体取决于分析的是某些事件过去发生情况、当前状态还是事件流转效率。

分析业务过程的目的是**决定数据粒度**，在业务过程事件分析中，需要预判所有分析需要细分的程度和范围，从而决定选择的粒度。

识别维表、选择好粒度之后，我们需要**基于此粒度设计维表**，包括维度属性等，用于分析时进行分组和筛选。**最后，需要确定衡量的指标**。

上一步模拟的场景中，经过业务过程调研，我们了解到A用户电商营销业务的交易订单功能模块的业务流程如下。



这是一个非常典型的电商交易业务流程图。在该业务流程图中，有创建订单、买家付款、卖家发货、确认收货四个核心业务步骤。由于确认收货代表交易成功，因此对营销部门的数据分析师作为用户，我们重点分析确认收货（交易成功）步骤即可。

在明确用户的业务过程之后，可以根据需要对进行分析决策的业务划分数据域。

#### 1.3划分数据域（在一个业务下按具体流程水平分表）

数据仓库是面向主题（数据综合、归类并进行分析利用）的应用。数据仓库模型设计**除横向的分层外**，通常也需要根据业务情况**纵向划分数据域**。数据域是联系较为紧密的数据主题的集合，是业务对象高度概括的概念，目的是便于管理和应用数据。

通常，咱们需要阅读各源系统的设计文档、数据字典和数据模型，研究逆向导出的物理数据模型。进而，可以进行跨源的主题域合并，跨源梳理出整个企业的数据域。

**数据域是指面向业务分析，将业务过程或者维度进行抽象的集合**。为保障整个体系的生命力，数据域需要抽象提炼，并长期维护更新。在划分数据域时，既能涵盖当前所有的业务需求，又能让新业务在进入时可以被包含进已有的数据域或扩展新的数据域。数据域的划分工作可以在业务调研之后进行，需要分析各个业务模块中有哪些业务活动。

**数据域可以按照用户企业的部门划分，也可以按照业务过程或者业务板块中的功能模块进行划分**。例如A公司电商营销业务板块可以划分为如下数据域，数据域中每一部分都是实际业务过程经过归纳抽象之后得出的。



#### 1.4定义维度与构建数据矩阵（定义表中的行与列）

明确每个数据域下有哪些业务过程后，需要开始定义维度，并基于维度构建总线矩阵。

##### 定义维度

在划分数据域、构建总线矩阵时，需要**结合对业务过程的分析定义维度**。以本教程中A电商公司的营销业务板块为例，在**交易数据域中**，我们**重点考察确认收货（交易成功）的业务过程**。

在**确认收货的业务过程**中，主要有**商品和收货地点**（本教程中，假设收货和购买是同一个地点）**两个维度**所依赖的业务角度。

从商品维度我们可以定义出以下维度的属性：

* 商品ID（主键）
* 商品名称
* 商品交易价格
* 商品新旧程度： 1 全新 2 闲置 3 二手
* 商品类目ID
* 商品类目名称
* 品类ID
* 品类名称
* 买家ID
* 商品状态： 0 正常 1 删除 2 下架 3 从未上架
* 商品所在城市
* 商品所在省份

从地域维度，我们可以定义出以下维度的属性：

* 城市code
* 城市名称
* 省份code
* 省份名称

作为维度建模的核心，在企业级数据仓库中**必须保证维度的唯一性**。以A公司的商品维度为例，有且只允许有一种维度定义。例如，省份code这个维度，对于任何业务过程所传达的信息都是一致的。

##### 构建总线矩阵

明确每个数据域下有哪些业务过程后，即可构建总线矩阵。我们需要**明确业务过程与哪些维度相关**，**并定义每个数据域下的业务过程和维度**。如下所示是A公司电商板块交易功能的总线矩阵，我们定义了购买省份、购买城市、类目名称、类目ID、品牌名称、品牌ID、商品名称、商品ID、成交金额等维度。



#### 1.5明确统计指标

需求调研输出的文档中，含有原子指标与派生指标，此时我们需要在设计汇总层表模型前完成指标的设计。

##### 指标定义注意事项

原子指标是明确的统计口径、计算逻辑： 原子指标=业务过程+度量。派生指标是常见的统计指标：派生指标=时间周期+修饰词+原子指标。原子指标的创建需要在业务过程定义后方才可创建。派生指标的创建一般需要在了解具体报表需求之后展开创建，在新建派生指标前必须新建好原子指标。 注意事项如下：

* 原子指标、修饰类型及修饰词，直接归属在业务过程下，其中修饰词继承修饰类型的数据域。
* 派生指标可以选择多个修饰词，由具体的派生指标语义决定。例如，支付金额为原子指标，则客单价（支付金额除以买家数）为派生指标。
* 派生指标唯一归属一个原子指标，继承原子指标的数据域，与修饰词的数据域无关。

##### 根据业务需求确定指标（定义查询与分析的数据在数仓中所处位置）

按模拟场景中，用户是电商营销部门的营销数据分析师。数据需求为最近一天厨具类目的商品在各省的销售总额、该类目Top10销售额商品名称、各省用户购买力分布（人均消费额）等，用于营销分析。

根据之前的分析，我们确认业务过程为：确认收货（交易成功），而度量为商品的销售金额。因此根据业务需求，我们可以定义出**原子指标：商品成功交易金额**。

派生指标为时间周期+修饰词+原子指标：

* 最近一天/全省/厨具类目各商品/销售总额
* 最近一天/全省厨具类目/人均/消费额（消费总额除以人数）

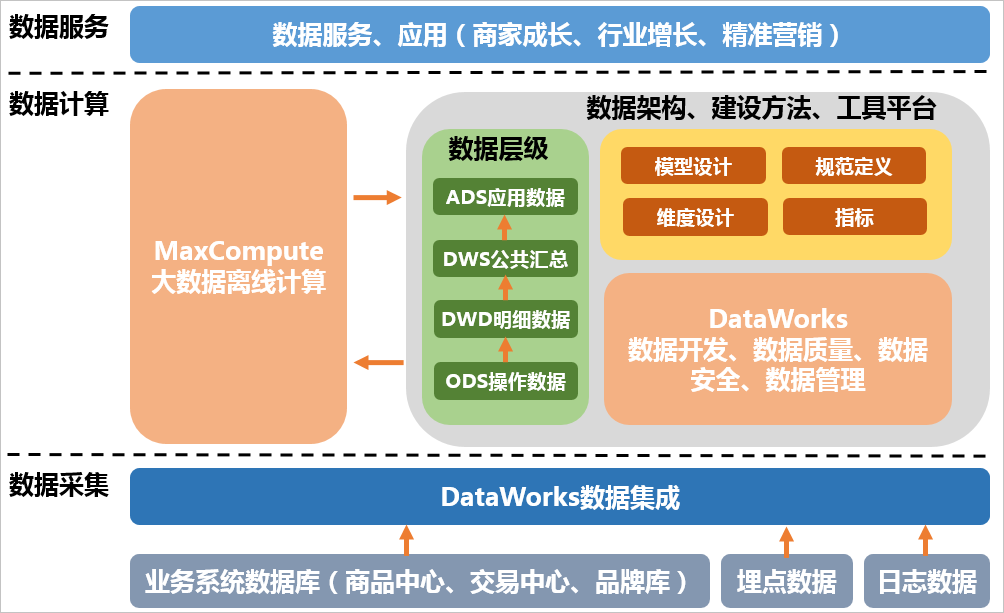
最近一天全省厨具类目各商品销售总额进行降序排序后取前10名的名称，即可得到该类目Top10销售额商品名称。

### 2.架构与模型设计

#### 2.1 技术架构选型

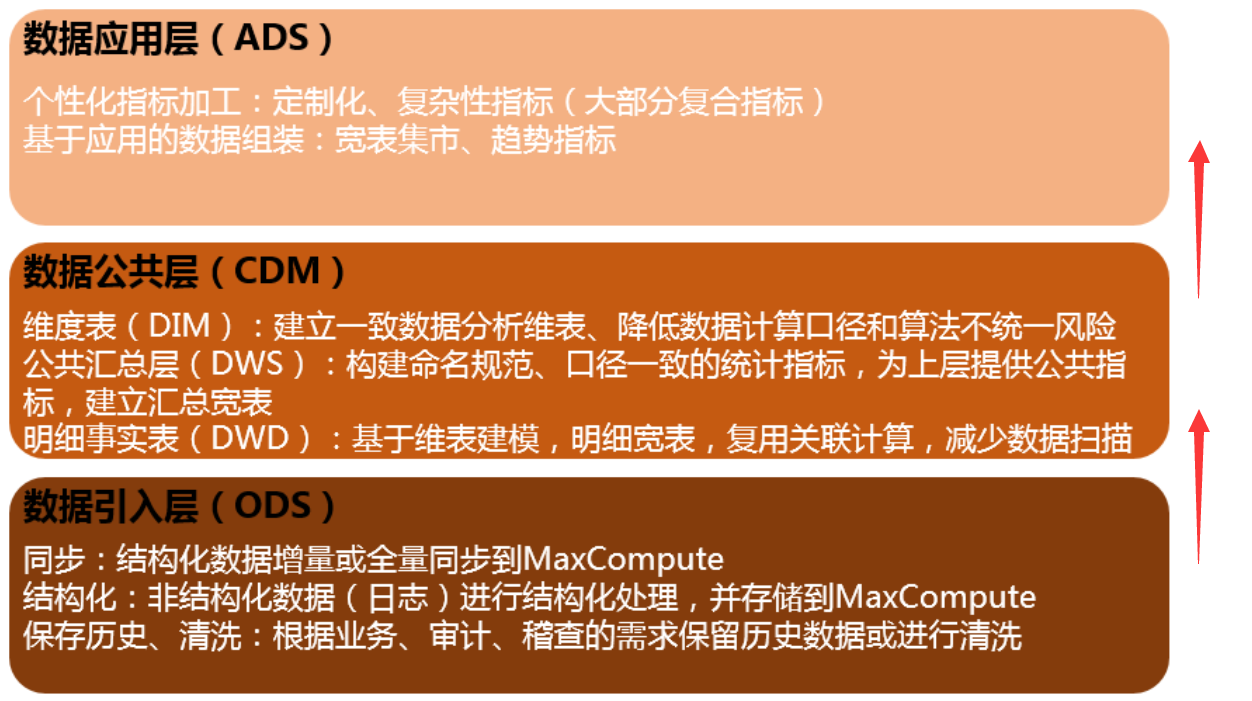
在数据模型设计之前，我们需要首先完成技术架构的选型。本教程中使用阿里云大数据产品MaxCompute配合DataWorks，完成整体的数据建模和研发流程。

完整的技术架构图如下图所示。其中，DataWorks的数据集成负责完成数据的采集和基本的ETL。MaxCompute作为整个大数据开发过程中的离线计算引擎。DataWorks则作为MaxCompute的数据仓库解决方案PaaS平台，包括数据集成、数据开发、数据地图、数据质量、数据安全、数据管理等在内的一站式开发管理界面。



#### 2.2 数据分层

在阿里巴巴的数据体系中，建议将数据仓库分为三层，自下而上为：数据引入层（ODS，Operation Data Store）、数据公共层（CDM，Common Data Model）和数据应用层（ADS，Application Data Service）。



* **数据引入层ODS**（Operation Data Store）：存放**未经过处理**的原始数据至数据仓库系统，结构上与源系统保持一致，是数据仓库的数据准备区。主要**完成基础数据引入到MaxCompute的职责，同时记录基础数据的历史变化**。
* **数据公共层CDM**（Common Data Model，又称通用数据模型层），包括DIM维度表、DWD和DWS。主要**对ODS数据引入层完成数据加工与整合**，建立**一致性的维度**，构建**可复用**的**面向分析和统计**的**明细事实表**，以及**汇总公共粒度的指标**。
  + **公共维度层（DIM）**：基于维度建模理念思想，**建立**整个企业的**一致性维度。**降低数据计算口径和算法不统一风险。

公共维度层的表通常也被称为逻辑维度表，维度和维度逻辑表通常一一对应。

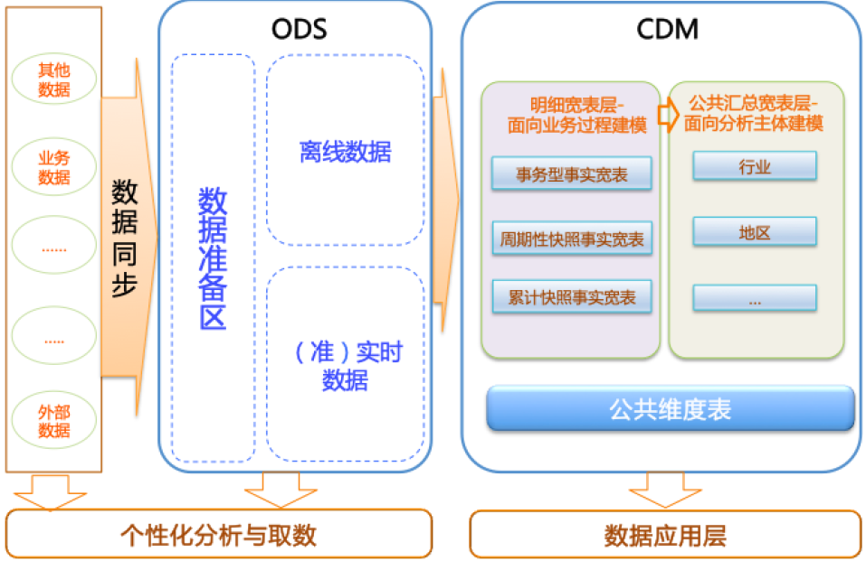
* + **公共汇总事实层（DWS**）：以**分析的主题对象**作为建模驱动，基于上层的应用和产品的指标需求，构建**公共粒度**的汇总指标事实表，以宽表化手段物理化模型。构建命名规范、口径一致的统计指标，为上层提供公共指标，建立汇总宽表、明细事实表。

公共汇总粒度事实层的表通常也被称为汇总逻辑表，用于存放派生指标数据。

* + **明细事实层（DWD）**：以**业务过程**作为建模驱动，基于每个具体的业务过程特点，构建**最细粒度**的明细层事实表。可以结合企业的数据使用特点，将明细事实表的某些重要维度属性字段做适当冗余，即宽表化处理。明细粒度事实层的表通常也被称为逻辑事实表。

**以二维结构关系型数据库为例，公共维度层类似于建立列（维度）；明细粒度实施层类似于建立行（对应列中的维度，描述列下的业务流程包含的所有事实，用度量数据作为描述事实的依据）；公共汇总粒度实施层类似于以一个数据域为对象进行跨表jion查询后的查询结果存储）。为方便记忆，数据高度看 DIM＞DWD＞DWS，I＞D＞S（这纯粹是为了强制记忆编的胡言乱语，不可较真）**

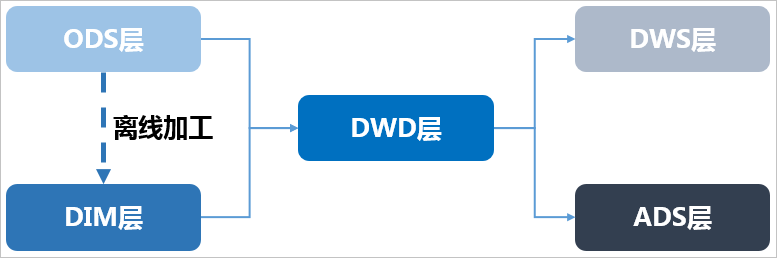
* 数据应用层ADS（Application Data Service）：**存放数据产品个性化的统计指标数据**。根据CDM与ODS层加工生成。



在A电商公司的模拟场景中，从交易数据系统的数据经过DataWorks数据集成，同步到数据仓库的ODS层。经过数据开发形成事实宽表后，再以商品、地域等为维度进行公共汇总。

整体的数据流向如下图所示。其中，ODS层到DIM层的ETL（萃取（萃取真好听）（Extract）、转置（Transform）及加载（Load））处理是在MaxCompute中进行的，处理完成后会同步到所有存储系统。ODS层和DWD层会放在数据中间件中，供下游订阅使用。而DWS层和ADS层的数据通常会落地到在线存储系统中，下游通过接口调用的形式使用。

（**为方便记忆，在CDM内，数据高度看 DIM＞DWD＞DWS，I＞D＞S；从整体数据分层的先后次序看，ODS＞CDM＞ADS，O＞C＞A，这纯粹是为了强制记忆编的胡言乱语，不可较真，记都记不住，谈日后什么理解**）



#### 2.3 数据模型

##### 数据接入层ODS设计

ODS层存放从业务系统获取的最原始的数据，是其他上层数据的源数据。业务数据系统中的数据通常为非常细节的数据，经过长时间累积，且访问频率很高，是面向应用的数据。

在模拟场景A电商公司的电商营销部门中，在ODS层主要包括的数据有：交易系统订单详情、用户信息详情、商品详情等。这些数据未经处理，是最原始的数据。逻辑上，这些数据都是以二维表的形式存储。虽然严格的说ODS层不属于数仓建模的范畴，但是合理的规划ODS层并做好数据同步也非常重要。针对模拟场景，使用了6张ODS表：

* 记录用于拍卖的商品信息：s\_auction。
* 记录用于正常售卖的商品信息：s\_sale。
* 记录用户详细信息：s\_users\_extra。
* 记录新增的商品成交订单信息：s\_biz\_order\_delta。
* 记录新增的物流订单信息：s\_logistics\_order\_delta。
* 记录新增的支付订单信息：s\_pay\_order\_delta。

##### 公共维度汇总层DIM设计

公共维度汇总层（DIM）基于维度建模理念，建立整个企业的一致性维度。

公共维度汇总层（DIM）主要由维度表（维表）构成。维度是逻辑概念，是衡量和观察业务的角度。维表是根据维度及其属性将数据平台上构建的物理化的表，采用宽表设计的原则。因此，公共维度汇总层（DIM）首先需要定义维度。

###### 定义维度

在划分数据域、构建总线矩阵时，需要结合对业务过程的分析定义维度。模拟场景中以A电商公司的营销业务板块为例，在交易数据域中，我们重点考察确认收货（交易成功）的业务过程。

确认收货的业务过程中，主要有商品和收货地点（本教程中，假设收货和购买是同一个地点）两个维度所依赖的业务角度。从商品角度可以定义出以下维度：

* 商品ID
* 商品名称
* 商品价格
* 商品新旧程度

0表示全新，1表示闲置，2表示二手。

* 商品类目ID
* 商品类目名称
* 品类ID
* 品类名称
* 买家ID
* 商品状态

0表示正常，1表示用户删除，2表示下架，3表示从未上架。

* 商品所在城市
* 商品所在省份

从地域角度，可以定义出以下维度：

* 买家ID
* 城市code
* 城市名称
* 省份code
* 省份名称

作为维度建模的核心，在企业级数据仓库中必须保证维度的唯一性。以A公司的商品维度为例，有且只允许有一种维度定义。例如，省份code这个维度，对于任何业务过程所传达的信息都是一致的。

###### 设计维表

完成维度定义后，可以对维度进行补充，进而生成维表。维表的设计需要注意：

* 建议维表单表信息不超过1000万条。
* 维表与其他表进行Join时，建议使用Map Join。
* 避免过于频繁的更新维表的数据。

在设计维表时，我们需要从下列方面进行考虑：

* 维表中数据的稳定性。

例如，A公司电商会员通常不会出现消亡，但会员数据可能在任何时候更新，此时要考虑创建单个分区存储全量数据。如果存在不会更新的记录，可能需要分别创建历史表与日常表。日常表用于存放当前有效的记录，保持表的数据量不会膨胀；历史表根据消亡时间插入对应分区，使用单个分区存放分区对应时间的消亡记录。

* 维表是否需要垂直拆分。

如果一个维表存在**大量属性不被使用**，或由于**承载过多属性字段导致查询变慢**，则需要考虑对字段进行拆分，**创建多个维表**。

* 维表是否需要水平拆分。

如果记录之间有明显的界限，可以考虑拆成多个表或设计成多级分区。

* 核心维表的产出时间。通常有严格的要求。

设计维表的主要步骤如下：

1. 初步定义维度。

保证维度的一致性。

1. 确定主维表（中心事实表，模拟场景中采用星型模型）。

此处的主维表通常是数据引入层（ODS）表，直接与业务系统同步。例如，s\_auction是与前台商品中心系统同步的商品表，此表即是主维表。

1. 确定相关维表。

数据仓库是业务源系统的数据整合，不同业务系统或者同一业务系统中的表之间存在关联性。根据对业务的梳理，确定哪些表和主维表存在关联关系，并选择其中的某些表用于生成维度属性。以商品维度为例，根据对业务逻辑的梳理，可以得到商品与类目、卖家和店铺等维度存在关联关系。

1. 确定维度属性。

主要包括两个阶段。第一个阶段是从主维表中选择维度属性或生成新的维度属性；第二个阶段是从相关维表中选择维度属性或生成新的维度属性。以商品维度为例，从主维表（s\_auction）、类目、卖家和店铺等相关维表中选择维度属性或生成新的维度属性。维度属性的设计需要注意：

* + 尽可能生成丰富的维度属性。
  + 尽可能多地给出富有意义的文字性描述。
  + 区分数值型属性和事实。
  + 尽量沉淀出通用的维度属性。

##### 明细粒度事实层DWD设计

###### 明细粒度事实层DWD拆解

明细粒度事实层以业务过程驱动建模，基于每个具体的业务过程特点，构建最细粒度的明细层事实表。可以结合用户的数据使用特点，将明细事实表的某些重要维度属性字段做适当冗余，**即宽表化处理**。

明细粒度事实层（DWD）的事实表通过获取描述业务过程的度量来描述业务过程，包括引用的维度和与业务过程有关的度量。**度量通常为数值型数据**，作为事实逻辑表的依据。事实逻辑表的描述信息是事实属性，事实属性中的外键字段通过对应维度进行关联。

**事实表中一条记录所表达的业务细节程度被称为粒度**。通常粒度可以通过两种方式来表述：一种是维度属性组合所表示的细节程度，一种是所表示的具体业务含义。（在选择维度和事实之前必须先声明粒度）

作为度量业务过程的事实，通常为整型或浮点型的十进制数值，有可加性、半可加性和不可加性三种类型：

* 可加性事实是指可以按照与事实表关联的任意维度进行汇总。
* 半可加性事实只能按照特定维度汇总，不能对所有维度汇总。例如库存可以按照地点和商品进行汇总，而按时间维度把一年中每个月的库存累加则毫无意义。
* 完全不可加性，例如比率型事实。对于不可加性的事实，可分解为可加的组件来实现聚集。

事实表相对维表通常更加细长，行增加速度也更快。维度属性可以存储到事实表中（将DIM维度表中的维度属性存储到DWD明细粒度事实表中），这种存储到事实表中的维度列称为**维度退化**，**可加快查询速度**。与其他存储在维表中的维度一样，维度退化可以用来进行事实表的过滤查询、实现聚合操作等。

明细粒度事实层（DWD）通常分为三种：事务事实表、周期快照事实表和累积快照事实表。

* 事务事实表用来描述业务过程，跟踪空间或时间上**某点的度量事件**，保存的是最原子的数据，也称为原子事实表。
* 周期快照事实表以具有规律性的、可预见的**时间间隔记录事实**。
* 累积快照事实表用来表述过程开始和结束之间的关键步骤事件，覆盖过程的整个生命周期，通常具有多个日期字段来记录关键时间点。当累积快照事实表随着生命周期不断变化时，记录也会随着过程的变化而被修改。

###### 明细粒度事实表设计原则

明细粒度事实表设计原则如下所示：

* 通常，一个明细粒度事实表仅和一个维度关联。
* 尽可能包含所有与业务过程相关的事实 。
* 只选择与业务过程相关的事实。
* 分解不可加性事实为可加的组件。
* 在选择维度和事实之前必须先声明粒度。
* 在同一个事实表中不能有多种不同粒度的事实。
* 事实的单位要保持一致。
* 谨慎处理Null值。
* 使用退化维度提高事实表的易用性。

明细粒度事实表整体设计流程如下图所示。



##### 公共汇总事实表DWS设计

公共汇总粒度事实层以分析的主题对象作为建模驱动，基于上层的应用和产品的指标需求构建公共粒度的汇总指标事实表。公共汇总层的一个表通常会对应一个派生指标。

聚集是指针对原始明细粒度的数据进行汇总。DWS公共汇总层是**面向分析对象的主题聚集建模**。在模拟场景A电商中，最终的分析目标为：最近一天某个类目（例如：厨具）商品在各省的销售总额、该类目Top10销售额商品名称、各省用户购买力分布。因此，我们可以以最终交易成功的商品、类目、买家等角度对最近一天的数据进行汇总。

* 聚集是不跨越事实的。聚集是针对原始星形模型进行的汇总。为获取和查询与原始模型一致的结果，聚集的维度和度量必须与原始模型保持一致，因此聚集是不跨越事实的。
* 聚集会带来查询性能的提升，但聚集也会增加ETL维护的难度。当子类目对应的一级类目发生变更时，先前存在的、已经被汇总到聚集表中的数据需要被重新调整。

此外，进行DWS层设计时还需遵循以下原则：

* 数据公用性：需考虑汇总的聚集是否可以提供给第三方使用。我们可以判断，基于某个维度的聚集是否经常用于数据分析中。如果答案是肯定的，就有必要把明细数据经过汇总沉淀到聚集表中。
* 不跨数据域。数据域是在较高层次上对数据进行分类聚集的抽象。数据域通常以业务过程进行分类，例如交易统一划到交易域下， 商品的新增、修改放到商品域下。
* 区分统计周期。在表的命名上要能说明数据的统计周期，例如\_1d表示最近1天，td表示截至当天，nd表示最近N天。

#### 2.4层次调用规范

ADS应用层优先调用数据仓库公共层数据。如果已经存在CDM层数据，不允许ADS应用层跨过CDM中间层从ODS层重复加工数据。

CDM中间层应该积极了解应用层数据的建设需求，将统一维度后的公用数据沉淀到公共层，为其他数据层次提供数据服务。同时，ADS应用层也需积极配合CDM中间层进行持续的数据公共建设的改造。避免出现过度的ODS层引用、不合理的数据复制和子集合冗余。总体遵循的层次调用原则如下：

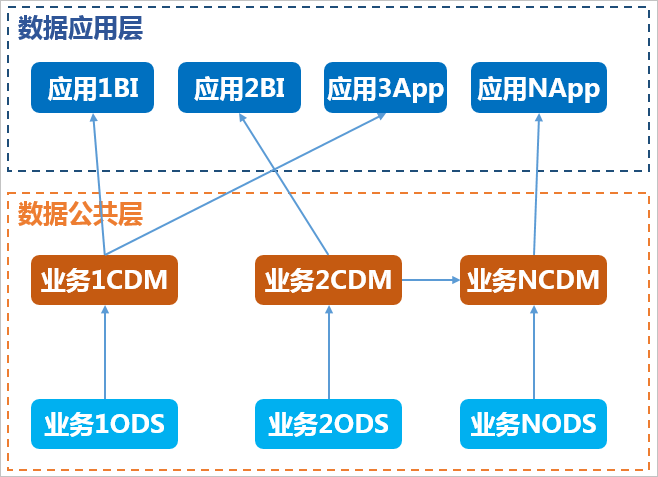
* ODS层数据不能直接被应用层任务引用。如果中间层没有沉淀的ODS层数据，则通过CDM层的视图访问。CDM层视图必须使用调度程序进行封装，保持视图的可维护性与可管理性。
* CDM层任务的深度不宜过大（建议不超过10层）。
* 一个计算刷新任务只允许一个输出表，特殊情况除外。
* 如果多个任务刷新输出一个表（不同任务插入不同的分区），DataWorks上需要建立一个虚拟任务，依赖多个任务的刷新和输出。通常，下游应该依赖此虚拟任务。
* 公共汇总事实层（DWS）优先调用明细事实层层（DWD），可累加指标计算。公共汇总事实层（DWS）尽量优先调用已经产出的粗粒度汇总层，避免大量汇总层数据直接从海量的明细数据层中计算得出。
* 在DWD明细事实层中，累计快照事实表优先调用事务型事实表，保持数据的一致性产出。
* 有针对性地建设CDM公共汇总层，避免应用层过度引用和依赖CDM层明细数据。

### 3.MaxCompute项目分配与安全（数据分层落地在MaxConpute产品中）

#### 项目分配

在为企业级大数据平台创建项目时，建议对ODS层、DWD及DWS层的数据按照**业务板块**的粒度建立项目，对于ADS层的数据，按照**应用**的粒度建立项目。

在模拟场景A电商公司中，用阿里云Maxcompute项目建立图：



* 对于ODS层项目，建议以ods为后缀，例如asaleods。
* 对于CDM层项目，建议以cdm为后缀，例如asalecdm。
* ADS应用层数据分为两类：
  + 数据报表、数据分析等以bi作为后缀，例如asalebi。
  + 数据产品应用以app作为后缀，例如asaleapp。

考虑到模拟场景中的用户仅局限于电商营销部门，该部门数据分析师的核心需求仅聚焦于电商业务板块中交易成功的业务流程，可以为ODS、CDM和ADS层分别仅建立一个项目。

#### 项目模式选择

MaxCpmpute的标准模式是指一个DataWorks项目对应两个MaxCompute项目，可设置开发和生产双环境，提升代码开发规范，并能够对表权限进行严格控制，禁止随意操作生产环境的表，保证生产表的数据安全。

在DataWorks建立项目时，建议使用标准模式以保证生产环境项目安全。完成项目创建后，会得到一个生产环境项目和以\_dev结尾的开发环境项目。例如**asaleods**和**asaleods\_dev。**

#### 项目权限配置

需要重点考虑为项目中的不同成员角色赋予不同的权限，例如生产任务如何保障不可随意变更、哪些成员可以进行代码编辑调试、哪些成员可以进行发布生产任务等。同时要为在数据开发过程中的资源使用赋权，并做好数据安全隔离。

### 4.数仓性能优化前，建立MaxCompute性能基准

MaxCompute性能表现优劣，主要取决表设计是否符合规范。为方便衡量MaxCompute表的性能表现，建议在优化性能之前首先建立性能基准。

在优化表前后测试系统性能时，需要记录每张表的数据同步时间、占用存储大小以及查询性能的详细信息。如果使用的是包年包月方式购买的MaxCompute项目资源，还需要记录购买数。

### 5.数仓性能优化

针对数仓的性能优化，主要是针对表和数据分布的优化。

#### 表设计优化参考（太过细碎，无法全部记忆，只做参考）

##### 产生大量小文件的操作（重点）

MaxCompute表的小文件会影响存储和计算性能。在进行表设计时，应考虑避开产生大量小文件的操作。会产生大量小文件的操作如下：

* 使用MaxCompute Tunnel SDK上传数据时，每1次Commit会产生1个文件。这时每个文件过小（例如几KB），并且频繁上传（例如每5秒上传一次），则一小时就会产生720个小文件，一天就会产生17280个小文件。
* 使用MaxCompute Tunnel SDK上传数据时，如果创建了Session却没有上传数据，而是直接Commit，则会产生大量空目录（在服务侧等同于小文件）。
* 使用MaxCompute客户端执行Tunnel命令上传时，将**本地大文件切分过小会导致上传后产生大量小文件**。
* 通过DataHub执行数据归档，DataHub的每个Shard写入MaxCompute时存在条件限制，即数据总量达到64 MB就Commit 1次，或每隔5分钟Commit 1次，形成1个文件。当开启的Shard数过多（例如20个Shard）时，每个Shard数据在5分钟内都远达不到64 MB（例如几百KB），就会产生大量小文件。相应地，一天会产生2412×20=5760个小文件。
* 通过DataWorks等数据开发工具将数据增量插入（INSERT INTO）MaxCompute表（或表分区）时，每次进行数据增量插入都会产生1个文件。若每次插入10条，则每天累计插入10000条记录，即会产生1000个小文件。
* 使用阿里云DTS将数据从RDS等数据库同步到MaxCompute时，会创建全量表和增量表。在增量表进行数据插入的过程中，会因为每次数据插入条数较少而造成增量表中的小文件问题。例如，每隔5分钟执行1次同步，每次同步的数据量为10条，一天内的增量为10000条，则会产生1000个小文件。此种场景下，需要在数据同步完成后合并全量极限表和增量数据表。
* 源数据采集客户端太多时，如果源数据通过Tunnel直接进入到一个分区，则每个源数据采集客户端提交一次数据，都会在同一分区下产生一个独立的文件，从而导致大量小文件的出现。
* 当SLS触发FunctionCompute持续高频地向MaxCompute中传入文件时，小文件流式数据会进入MaxCompute。

##### 根据数据划分项目空间

项目空间（Project）是MaxCompute最上层的对象。按项目空间进行资源的分配、隔离和管理，实现了多租户的管理能力。如果多个应用需要**共享数据**，则推荐**使用同一个项目空间**；反之，如果多个应用所需的数据是**无关**的，则推荐使用**不同的项目空间**。项目空间的表和分区可以通过Package授权的方式进行交换。

##### 维度表的设计

描述属性的表通常被设计为维度表。维度表可与任意表组中的任意表进行关联，且创建时无需配置分区信息，但是对单表个数有所限制。通常要求维度表的单表量不超过1000万个。

##### 采集源表的设计

**数据采集方式**包括**流式数据写入（实时）**、**批量数据写入**和**周期调度条式数据插入**。数据量较大时，需确保同一个业务单元的数据使用分区表设计；数据量较小时，需优化采集频率。

* 流式数据写入
  + 对于流式写入的数据，采集的通道通常较多，相关采集通道应该进行有效区分。在单个数据通道写入量较大的情况下，应该按照时间进行分区设计。
  + 在采集通道数据量较小的情况下，适合采取非分区表设计，将终端类型和采集时间设计成标准列字段。
  + 采用DataHub进行数据写入时，应该合理规划Shard数量，避免出现由于Shard过多导致采集通道流量较小、通道数量较多的问题。
* 批量数据写入

使用批量数据写入方式时，应重点关注写入周期。

* 周期调度条式数据插入

应避免使用周期调度条式数据插入的方法。若无法避免使用此方法，则需建立分区表，在新分区进行插入操作，减小对于原来分区的影响。

##### 日志表的设计

日志的本质是个流水表，**不涉及记录的更新**。日志表设计需注意以下几点：

* 考虑是否需要对**日志**进行**去重处理**。
* 考虑是否需要**扩展维度属性**。
  + 通过考虑业务使用的频次以及关联是否会造成产出的延迟，来确定是否需要关联维度表、扩展维度属性字段。
  + 需要谨慎选择是否对维度表进行扩展。
* 考虑**区分终端类型**。
  + 日志表中的数据量很庞大，在业务分析使用时，通常会按PC端、APP端来统计分析。由于PC端、APP端采用不同的体系采集数据，所以通常需要按照终端设计多个明细DWD表。
  + 如果终端较多但数据量不大，例如，一个终端的数据量小于1 TB但采集次数较多，则可以不对终端进行分区，设置终端信息为普通列。

##### 表创建设计示例

* 场景

天气情况信息采集。

* 基本信息
  + 数据信息包括地名、关于此地的属性信息（例如，面积和基本人口数量等）和天气信息。
  + 属性的数据变化较小，但天气信息数采用多个终端采集，且数据量较大。
  + 天气信息变化较大，但在终端数量稳定的情况下流量基本稳定。
* 表设计指南
  + 建议将数据信息划分为基本属性表和天气日志表，分别用于存储变化小和变化大的数据。
  + 因为天气信息的数据量巨大，在对天气日志表按照地域进行分区后，再次按照时间（例如，天）进行二级分区。此种分区方式可避免发生因某一个地点或某一个时间的天气变化而造成其他无关数据变化。
  + 建议采集终端上使用DataHub进行数据汇聚，然后依据稳定的流量值选择合适的Shard通道数量，以批量数据传输的方式写入到天气日志表中，而非INSERT INTO。

#### 表的其他优化技巧

* + 中间表的利用：适用于数据量非常大，下游任务很多的表。
  + 拆表：适用于个别字段产出极慢的情况，可以将字段拆分为单独的表。
  + 合表：随着数仓的发展，针对业务重叠或重复的表，可以进行任务和数据合并。
  + 拉链表：合理利用拉链表能减少的存储消耗。
  + 利用MaxCompute表的特殊功能。

#### 数据分布优化参考

Hash Clustering表的优势在于可以实现Bucket Pruning优化、Aggregation优化以及存储优化。在创建表时，使用clustered by指定Hash Key后，MaxCompute将对指定列进行Hash运算，按照Hash值分散到各个Bucket里。Hash Key值的选择原则为选择重复键值少的列。

### 6.优化效果验证

完成数仓的优化后，需要对结果进行评估验证，确认优化的有效性。

如果在优化过程中改变了表结构，需要删除原有的表，并根据优化策略新建表和分区。

在重新创建表并导入数据后，需要重新测试数仓性能。可以通过下列表格记录相关数据，并与性能基准进行比对，性能基准详情请参见[建立性能基准](https://help.aliyun.com/document_detail/120840.html)。

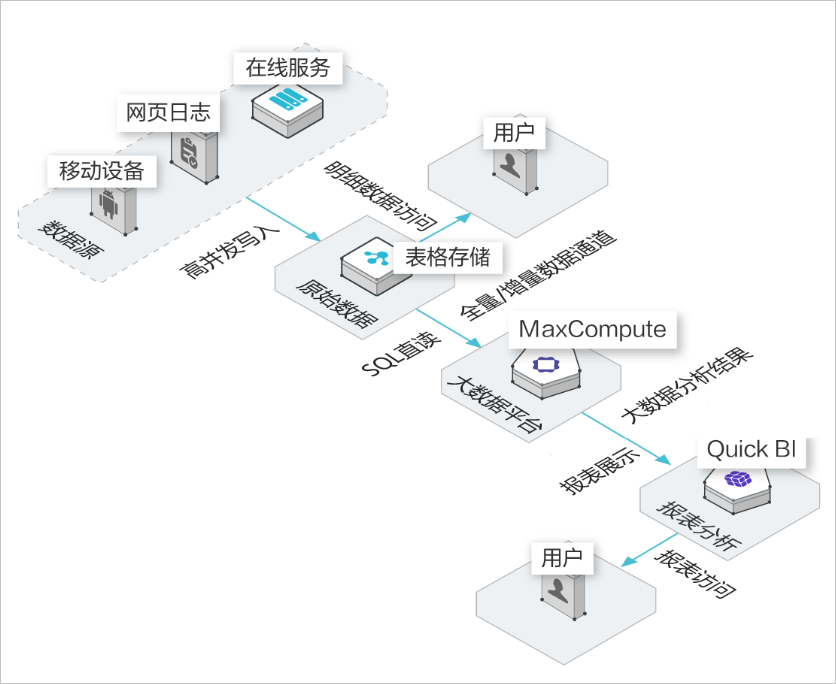


## 四．搭建互联网在线运营分析平台（理论投入实践）

本文的示例基于真实的网站日志数据集，数据来源于某网站上的HTTP访问日志数据。基于这份网站日志，用户可以实现两个分析需求：

* 统计并展现网站的PV（浏览次数）和UV（独立访客），并能够按照用户的终端类型（例如，Android、iPad、iPhone和PC等）分别统计。
* 统计并展现网站的流量来源地域。

整体使用流程



技术平台在此场景下的使用流程为：

技术平台搭建—测试数据准备—使用MaxCompute与DataWorks进行数据建模与开发（MaxCompute中新建数据表**→→**设计工作流**→→**节点配置**→→**任务提交测试）—使用QBI进行数据可视化展示**→→**质量保障管理

### 1.技术平台搭建

在阿里云分别开通，用于数据接入与存储的表格存储产品Tablestore、大数据计算引擎MaxCompute、基于计算引擎形成的大数据分析解决方案PAAS平台产品DataWorks、以及最终的数据可视化展示的报表生成工具QuickBI。

MaxComoute（后面用MX作为简称代替）分为简单模式（只有一个MX项目空间，为开发测试环境）和标准环境（在MX中有两个项目空间，分别为开发测试环境与生产环境），在这个案例中MX用简单模式作为演示。

在各项产品创建后，需要在如MX和DataWorks（后面用DW作为简称代替），创建项目空间，在项目空间中管理项目成员（添加授权账号，并对添加的账号赋予如开发、运维、管理员等相关角色授权），以及在TableStore（后面用表格存储作为简称代替）中配置MX与DW的访问权限，构建产品之间的连接，方面后面直接应用。

### 2.测试数据准备

在数据准备阶段，需要通过数据Demo包生成模拟真实环境的数据，以便后续数据开发使用。

下载Demo包，并配置Demo包环境后，启动Demo会会自动帮助我们在表格存储中分别创建用于测试技术平台是否可以正常运作的数据样本——原始记录表与分析结果表。

### 3.数据建模与开发

#### MaxCompute中新建数据表

点击DW中创建好的项目空间后，在项目空间的详细页面中，选择“数据开发”（PS: 项目空间Project是MX的最上层目录，项目空间下的详细页面中有数据集成、数据开发、数据地图、运维中心等详细功能洞天）后进入数据开发界面（PS:DW的数据开发界面是托拉拽如同搭积木的可视化操作形式，就像画拓扑图一样，将各种产品的图标在空白画布上排列后，用线头相互关联表示图标间依赖，组成一个有向无环图工作流程，再对拓扑图中的各个产品点开进行详细的配置，这个拓扑图就变成一个真实的系统开始工作，这很像阿里云的资源编排服务ROS和中间件的应用实时监控服务ARMS，云产品的使用流程，从之前多产品组成一个系统后进行复杂代码操作实现一个个操作，演化为将多个产品与复杂的代码操作封装为一个傻瓜式的拖拉拽视图系统，只需要做简单配置与简单代码操作即可实现复杂功能），在数据开发界面中的业务流程分项点击右键新建业务流程，本教程中，业务流程名为BiliBili，分别创建外部表（用来调用表格存储中的数据样本），以及创建MX内部中的ODS层表、DWD层表、ADS层表（因本场景只是做网站日志分析因此不创建公共维度表DIM层表与公共汇总事实表DWS层表）

#### 设计工作流

对于本模拟场景中这种较为简单的单数据流场景（网站的日志分析），我们可以选择每个数据表（数仓层次）对应一个工作流。通过设计工作流，可以明确在整体数据开发过程中各任务节点的排布。

在DW选中项目空间-数据开发界面中点击业务流程，形成一张空白画板，在画板上①拖入一个虚拟节点图表，命名为start；②向画布中拖入3个ODPS SQL节点图标，分别命名为ods\_user\_trace\_log、dw\_user\_trace\_log、rpt\_user\_trace\_log（分别为之前在MX中创建好的ODS层表、DWD层表、ADS层表），并形成上下顺序排列的依赖关系。

#### 节点配置

完成工作流设计后，我们需要对每个数据开发节点进行配置，填写SQL语句。

##### 注册自定义函数

在日志分析场景中，**日志类型数据的清洗转换过程中把IP地址转换为归属地必须有IP地址库**，因此需要下载用于IP地转换的自定义函数Java包getaddr.jar以及地址库ip.dat。

* 下载自定义函数资源包：下载用于IP地转换的自定义函数Java包getaddr.jar以及地址库ip.dat（阿里云MX文档工具页面提供下载地址）。
* 通过DW向MX上传资源包：在DW数据开发-业务流程下级分目录-咱们创建的BiliBili业务流程下的**MaxCompute**，选择**新建 > 资源**。需要分别新建**File**和**JAR**类型的资源。**File**类型上传地址库ip.dat，**JAR**类型对应Java包getaddr.jar。
* 通过DW向MX注册函数：在业务流程下右键单击**MaxCompute**，选择**新建 > 函数**，将函数命名为**getregion—**在**注册函数**页面，依次填写类名为**odps.test.GetAddr**，资源列表为**getaddr.jar,ip.dat**，命令格式为**getregion(ip string)**，保存后提交函数注册

##### 配置工作流中的节点

* ① 配置虚拟节点start：双击start节点，进入节点配置页面**→→**单击右侧的调度配置，在调度依赖区域下单击使用工作空间根节点完成配置**→→**在时间属性区域选择重跑属性为运行成功或失败后皆可重跑**→→**提交；
* ② 配置ODPS SQL节点ods\_user\_trace\_log（ODS层表）：双击ods\_user\_trace\_log节点，进入节点配置界面，编写SQL代码，形成处理逻辑**→→**完成代码编写后，单击右侧的调度配置，选择自动解析为否**→→**手动删除错误的依赖关系**→→**按照业务流程顺序搜索正确的上游节点，例如此处为start，并单击添加**→→**在时间属性区域选择重跑属性为运行成功或失败后皆可重跑—提交；
* ③ 配置ODPS SQL节点dw\_user\_trace\_log（DWD层表）：可以使用与ods\_user\_trace\_log节点一样的方法配置dw\_user\_trace\_log节点，仅SQL代码不一样；
* ④ 配置ODPS SQL节点rpt\_user\_trace\_log（AWS层表）：同DWD层表一样设置，仅SQL代码不一样；
* ⑤ 结果验证：双击业务流程BiliBili（咱们之前创建的），打开画布面板，点击运行，查看有向无环图中各节点是否均运行成功，如有节点运行失败，则右键点击该失败节点查看日志进行分析，查看SQL命令或上下依赖关系是否错误。

#### 任务提交与跑数

打开业务流程BiliBili右上角运维中心，在页面的左侧导航栏点击周期任务运维＞周期任务，双击节点列表中的虚拟节点start**→→**在右侧流程图上，右键单击虚拟节点start，选择**补数据 > 当前节点及下游节点→→**在**补数据**页面，选中所有需要补数据的节点，选择业务日期为过去一周，单击**确定→→**在左侧导航栏，单击**补数据实例**，查看补数据实例的运行情况，并通过单击**刷新**查看实时状态**→→**补数据实例运行完成后，验证结果（在左侧导航栏，单击业务流程BiliBili**> MaxCompute**，右键单击**数据开发**，选择**新建 > ODPS SQL**，新建名为query的SQL节点**→→**输入如下SQL语句，查询2019年6月11日到2019年6月17日之间表rpt\_user\_trace\_log中的数据，确认数据是否成功写入rpt\_user\_trace\_log表。）

### 4.数据可视化展示

在上一步任务测试与跑数中，数据表rpt\_user\_trace\_log加工完成后，表中包含了country、province、city、device\_brand、use\_time、pv等字段信息。我们可以通过仪表板展示用户的核心指标、周期变化、用户地区分布和记录。

在QBI控制台的个人空间，操作步骤分为获取数据源**→→**数据建模**→→**数据分析（仪表板/电子表格）

#### 获取数据源

单击左侧导航栏上的**数据源**，进入**数据源**页面**→→**单击右上角的**新建数据源**。选择**云数据库 > MaxCompute→→**在**添加MaxCompute数据源**页面，配置数据源连接参数**→→**完成填写后，单击**连接测试**，待显示**数据源连通性正常**后单击**添加**即可**→→**单击**连接测试**，进行数据源连通性测试**→→**单击**添加**，完成数据源添加，成功添加完成后，页面自动跳转到**数据源**管理页面，并在页面右侧展示出数据源所包含的所有数据表。



#### 数据建模

##### 创建数据集

在添加好数据源后，**数据源**管理页面右侧展示出数据源所包含的所有数据表，找到并点击rpt\_user\_trace\_log表，**创建好确认名称的BiliBili数据集**

##### 对数据集进行编辑

单击刚刚创建的BiliBili数据集，对数据集进行编辑（常见的数据集加工包括**维度、度量的切换**、**修改维度的类型**、**增加计算字段**、**创建层次结构**、**修改字段的数据类型**、**更改度量聚合方式**、**制作关联模型**）

以转换字段的维度类型为例：

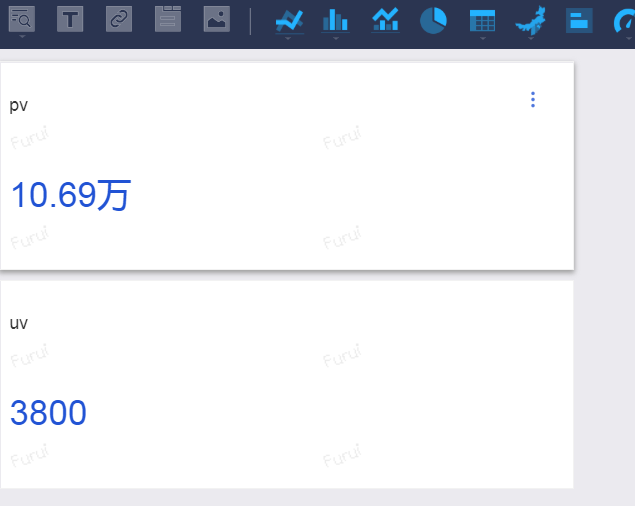
* 右键单击BiliBili数据集中的维度下的dt字段，选择维度类型切换 > 日期（源数据格式） > yyyyMMdd
* 右键单击BiliBili数据集中的维度下的province字段，选择维度类型切换 > 地理信息 > 省/直辖市
* 右键单击BiliBili数据集中的维度下的city字段，选择维度类型切换 > 地理信息 > 市。转换成功后，在左侧维度栏中会看到字段前多一个地理位置图标
* 新建层次结构，将city字段移到province层次结构的树下
* 完成上述操作后，单击保存，返回数据集列表

#### 数据分析

在数据分析界面，可以选择仪表盘或电子表格两种分析结果，这里以仪表盘为例，作为带咱们领略一下阿里云产品业务逻辑的操作路径。

**1.** 单击BiliBili数据集后的新建仪表板图标，选择**常规模式**，进入仪表板编辑页，从仪表板空间中向空白区拖入2个指标看板，调整布局成一排。

* 指标看板一：选择数据来源为数据集BiliBili，选择度量为pv。由于数据表rpt\_user\_trace\_log为分区表，因此必须在过滤器处选择筛选的日期，本例中筛选为2019~2019年，完成设置后单击更新。
* 指标看板二：选择数据来源为来自数据集BiliBili，选择度量为uv，其他操作同上。完成设置后单击更新样式处设置指标看板显示的名称，显示效果如下。



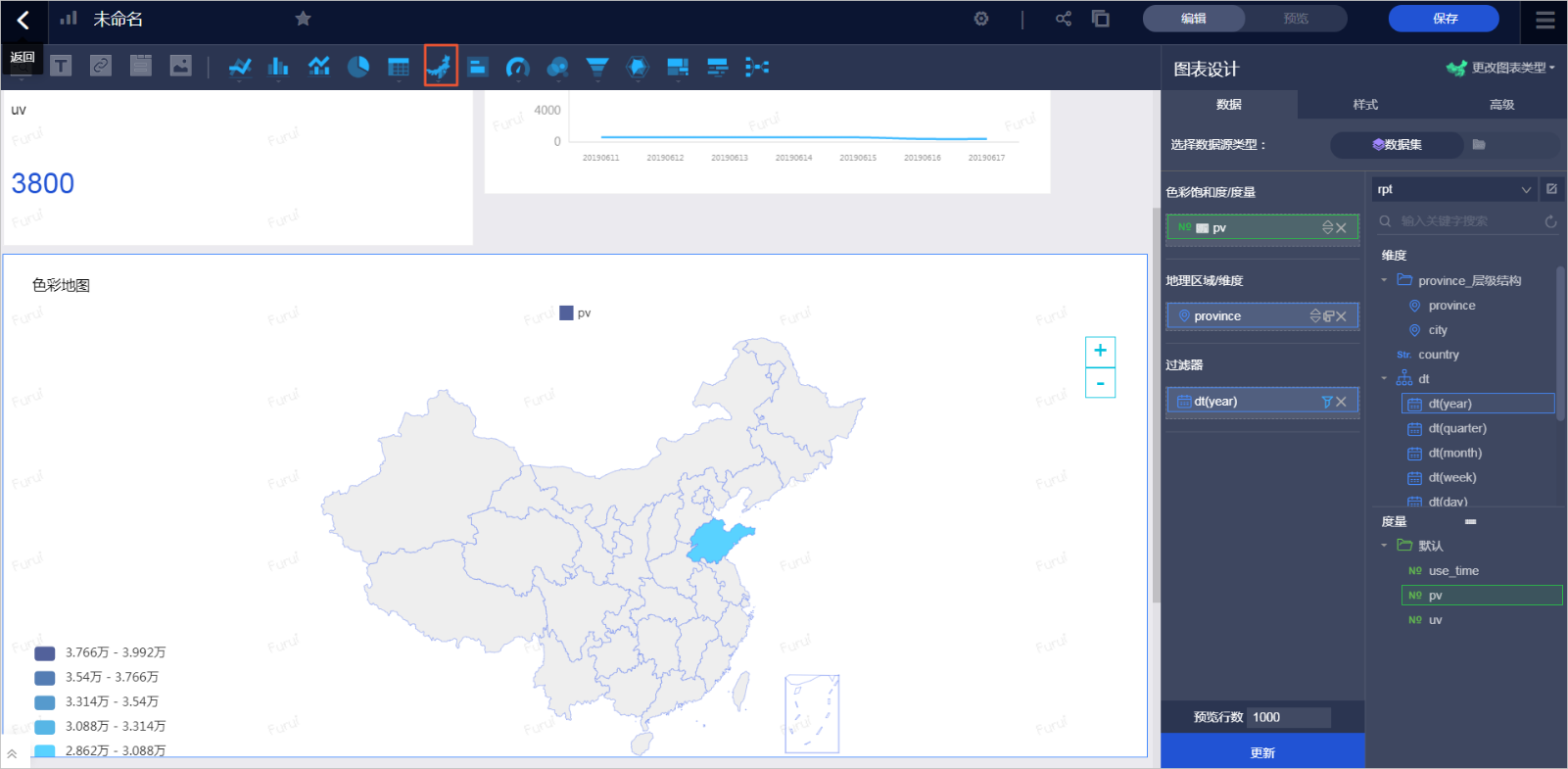
**2.** 制作趋势图：将图表区域内的**线图**拖拽到左侧画布。

参数配置如下，完成之后单击**更新**：

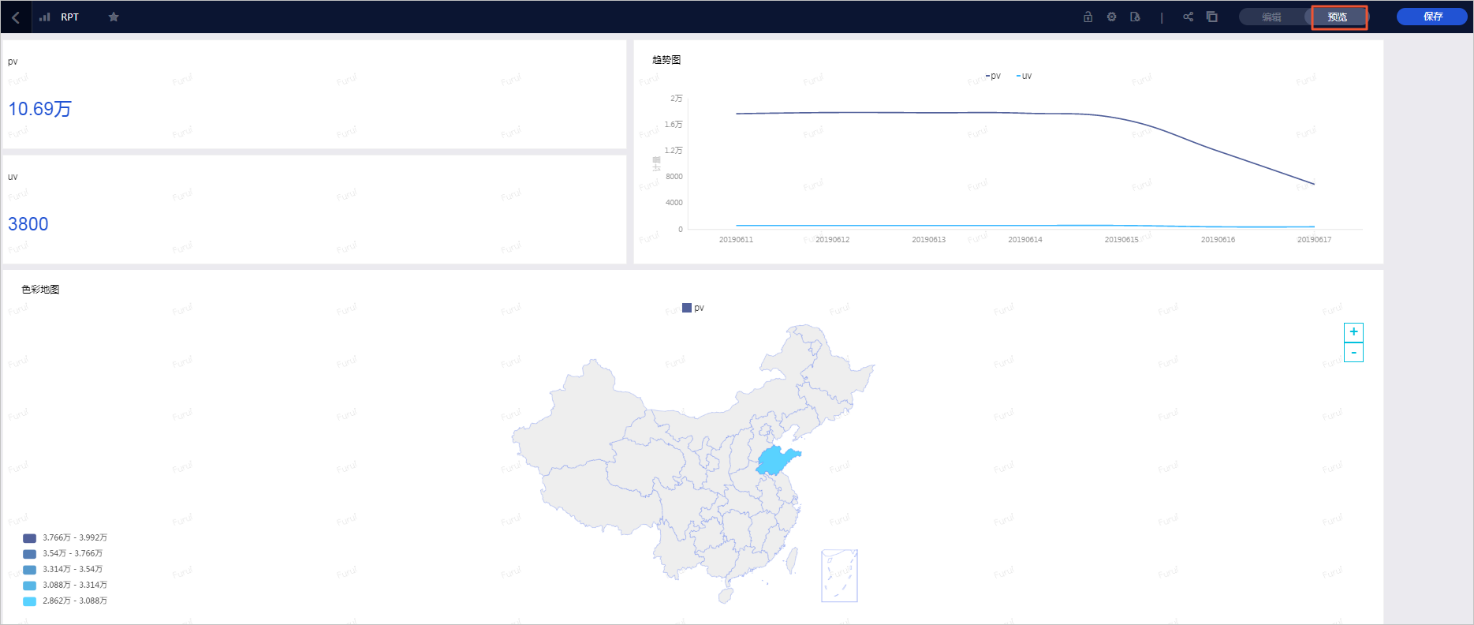
* **值轴/度量**：pv、uv
* **类别轴/维度**：dt（day）
* **过滤器**：dt（year）



**3.** 制作色彩地图：单击图表区域内的色彩地图，并选择数据源来源为数据集rpt\_user\_trace\_log，选择**地理区域/维度**为province（地区）、**色彩饱和度/度量**为pv，选择完成后单击**更新**。



**4.** 完成配置后，单击**保存**及**预览**，即可看到展示效果



### 5.质量保障管理

在整体数据链路的处理过程中，为保证最终产出数据的质量，我们需要对数据仓库ODS、CDM和ADS层的数据分别进行监控。ods\_user\_trace\_log、dw\_user\_trace\_log、rpt\_user\_trace\_log分别代表数据仓库的ODS、CDM和ADS层。

数据质量可以从**完整性**、**准确性**、**一致性**和**及时性**共四个角度进行评估：

* 完整性

完整性是指数据的记录和信息是否完整、不缺失。数据的缺失包括数据记录的缺失（表行数异常）和记录中某字段信息的缺失（字段出现空值）。在本教程中，我们需要重点关注数据的生产环节（MaxCompute外部表引用的表格存储数据）和加工环节（数据仓库CDM及ADS层）中表行数是否大于0、表行数波动是否正常以及字段是否出现空值或重复的情况。

* 准确性

准确性是指数据记录中信息和数据是否准确、不存在错误或异常。例如，在本教程中，如果UV、PV数值小于0，则明显是错误数据。

* 一致性

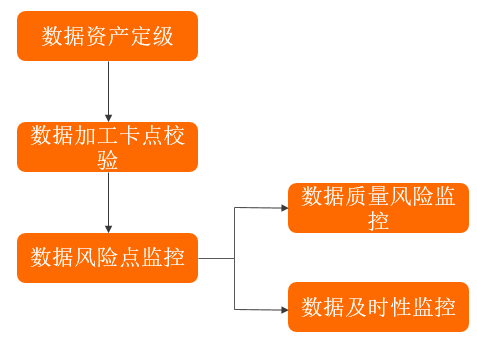
对于不同的业务流程和节点，同一份数据必须保持一致性。例如表province字段中如果有**浙江**、**ZJ**两种表述，在**group by province**时会出现两条记录。

* 及时性

及时性主要体现在最终ADS层的数据可以及时产出。为保证及时性，需要确保整条数据加工链路上的每个环节都可以及时产出数据。本教程将利用DataWorks智能监控功能保证数据加工每个环节的及时性。

为了实现以上四个角度的评估，阿里云给出的数据质量标准管理流程：

**业务数据资产定级→→加工卡点效验→→风险点监控和及时性监控**



##### 数据资产定级

数据的资产等级，可以根据数据质量不满足完整性、准确性、一致性、及时性对业务的影响程度进行划分。

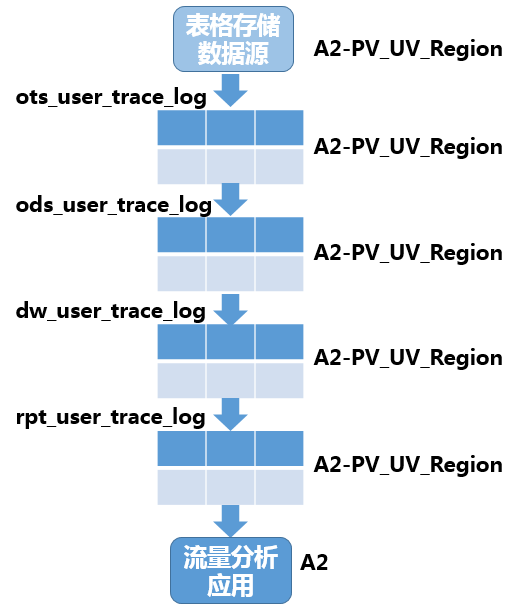
数据等级定义如下：

* **毁灭性质**：数据一旦出错，将会引起重大资产损失，面临重大收益损失等。标记为**A1**。
* **全局性质**：数据直接或间接用于企业级业务、效果评估和重要决策等。标记为**A2**。
* **局部性质**：数据直接或间接用于某些业务线的运营、报告等，如果出现问题会给业务线造成一定的影响或造成工作效率降低。标记为**A3**。
* **一般性质**：数据主要用于日常数据分析，出现问题带来的影响极小。标记为**A4**。
* **未知性质**：无法明确数据的应用场景。标记为**Ax**。

资产等级标记包含毁灭性质为A1、全局性质为A2、局部性质为A3、一般性质为A4、未知性质为Ax。**重要程度与灾难程度排序为：A1>A2>A3>A4>Ax**。

在数据流转链路上，我们需要整理消费各个表的应用业务。通过给这些应用业务划分数据资产等级，结合数据的上下游依赖关系，将**整个链路打**上某一类资产等级的**标签**。在本教程中，互联网在线运营分析平台只存在一个应用，统计并展现网站的PV和UV，并能够按照用户的终端类型和地域进行统计，命名为PV\_UV\_Region。假设该应用会直接影响整个企业的重要业务决策，可以定级应用为A2，从而整个数据链路上的表的数据等级，都可以标记为A2-PV\_UV\_Region。

按照数据资产定级，给外部表ots、数据接入层ODS、明细事实层DWD、数据应用层ADS分别打标签：



##### 离线数据加工卡点效验

离线数据分析系统的数据在生成过程中涉及到：代码提交卡点**→→**任务发布上线卡点**→→**任务变更卡点，三个会影响到数据分析业务是否能够成功运行的关键卡点，需要对这三个环节所提交的代码或操作重点进行正确性效验（DataWorks在这三个卡点上已经提供了自动工具，但工具毕竟只能检查明显错误，最终数据分析系统运行的性能和性价比表现，是由表设计与数据分布的优化程度直接影响的，这里会把数据分析师拉开层级），降低系统运行失败风险。

###### 代码提交卡点效验

代码提交卡点校验主要包括**在提交代码时**，手动或自动进行SQL扫描，检查我们的SQL逻辑。校验规则分类如下：

* 代码规范类规则。

例如，表命名规范、生命周期设置及表注释等。

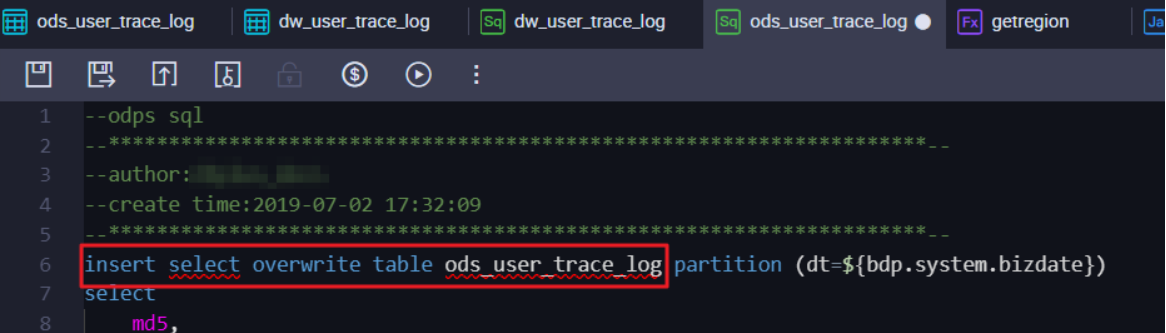
* 代码质量类规则。

例如，分母为0提醒、NULL值参与计算影响结果提醒及插入字段顺序错误等。

* 代码性能类规则。

例如，分区裁剪失效、扫描大表提醒及重复计算检测等。

在使用DataWorks数据开发功能时提供语法自动检测功能，如果所编写的代码中有语法错误，会出现如下红色波浪线提示。



###### 任务发布上线卡点效验

为保障线上数据的准确性，每次变更都需要经过测试再发布到线上生产环境，且生产环境测试通过后才算发布成功。

发布上线前的测试包括代码审查和回归测试。对于资产等级较高的应用，必须在完成回归测试之后，才允许任务发布，本教程中应用为A2等级，属于高资产级别应用。

回归测试需要我们能充分模拟真实环境进行测试：

* 对于标准模式项目，可使用SQL语句将数据从生产环境复制到开发环境，运行业务流程，观察是否存在报错。
* 对于简单模式项目，可以直接运行业务流程，观察是否存在报错。



###### 任务变更

在进行更新操作前，需要通知下游变更原因、变更逻辑、变更时间等信息。下游对此次变更没有异议后，再按照约定时间执行发布变更，这样可以将变更对下游的影响降到最低。

例如，在本教程中，如果表格存储数据源的表结构发生了变更，需要通知ots\_user\_trace\_log、ods\_user\_trace\_log、dw\_user\_trace\_log及rpt\_user\_trace\_log表的责任人，及时更新表结构。

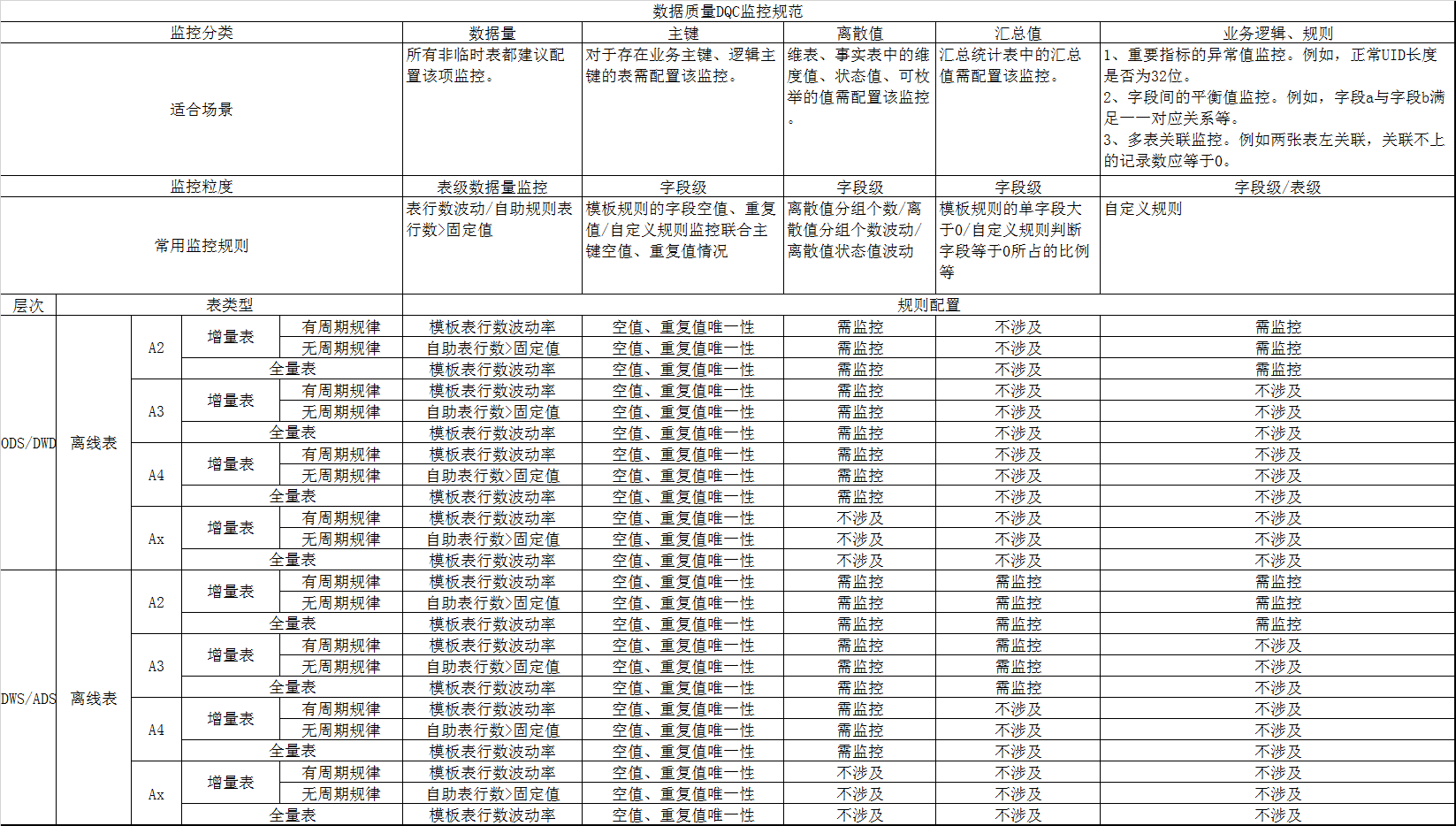
##### 数据质量风险监控

DataWorks中自有的数据质量（DQC）功能，完成数仓各层次的数据的准确性、一致性和完整性质量监控。

数据质量监控和数据资产等级对应，可以根据以下因素细化我们的监控配置：

* 监控分类：数据量、主键、离散值、汇总值、业务规则和逻辑规则。
* 监控粒度：字段级别、表级别。
* 监控层次：ODS、CDM、ADS三层数据，其中ODS和DWD层主要偏重数据的完整性和一致性。DWS和ADS层数据量较小、逻辑复杂，偏重数据的准确性。

还有一张根据各数据层表，与各层表对应的资产等级，做的数据质量DQC监控规范，非常密密麻麻，不做细学。



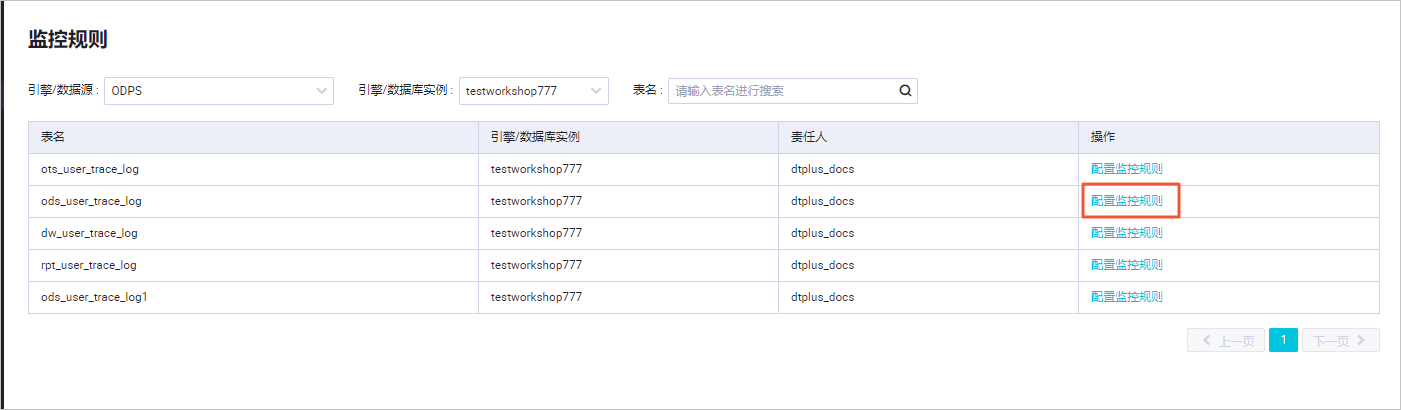
###### 数据质量监控使用演示（透过实践了解数据质量监控的功能，不要拘泥于操作具体步骤）

ODS/CDM/ADS三层的监控配置方法相同，只是每层的监控规则有所不同，因此只以ODS数据接入层数据质量监控为演示

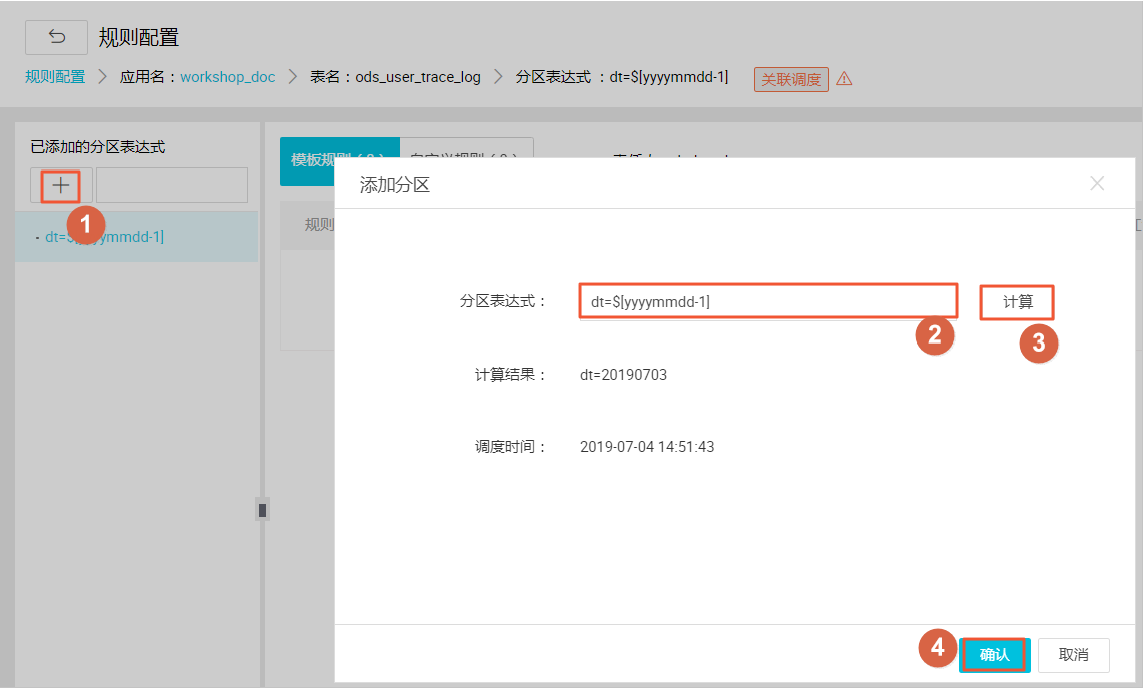
开篇：

ODS层表中的数据来源于OSS上的日志文件，作为源头表，您需要尽早判断此表分区中是否有数据。如果这张表中没有数据，则后续任务运行无意义，需要阻止后续任务运行。

* 1. 在Dataworks中点击选中的项目空间，进入数据开发页面，在左上角，进入数据质量页面。
  2. 在数据质量页面的左侧导航栏点击监控规则，在监控规则页面中会列出数据源中的所有外部表和内部层表，在ODS层表ods\_user\_trace\_log，单击其后的**配置监控规则**。



* 1. 在监控规则页面里，点击添加分区
     + 单击**+**，选择**分区表达式**为**dt=$[yyyymmdd-1]**，对应表ods\_user\_trace\_log的分区格式为**${bdp.system.bizdate}**（即获取到前一天的日期）。
     + 单击**计算**，验证计算结果是否正确。
     + 单击**确认**，完成分区的添加



* 1. 创建规则后，**添加规则确保ODS层表分区内存在数据**；添加规则，**监控重复数据**；添加规则，**监控空值数据**
  2. 批量保存规则后，点击试跑，进行数据质量的效验。
  3. 关联调度：在规则配置完毕，且试跑成功的情况下，您需要将表和其产出任务进行关联，这样每次表的产出任务运行完毕后，都会触发数据质量规则的校验，以保证数据的准确性。
  4. 配置任务订阅：关联调度后，每次调度任务运行完毕，都会触发数据质量的校验。数据质量支持设置规则订阅，可以针对重要的表及其规则设置订阅，设置订阅后会根据数据质量的校验结果进行告警，从而实现对校验结果的跟踪。

单击订阅管理，设置接收人以及订阅方式。目前支持邮件通知、邮件和短信通知、钉钉群机器人和钉钉群机器人@ALL四种方式。

##### 数据及时性监控

基于MaxCompute的离线任务对数据产出有严格的时间要求，在确保数据准确性的前提下，还需要让数据能够及时提供服务。标准版及以上版本DataWorks提供智能监控功能，可以用智能监控的规则管理功能监控数据的及时性。

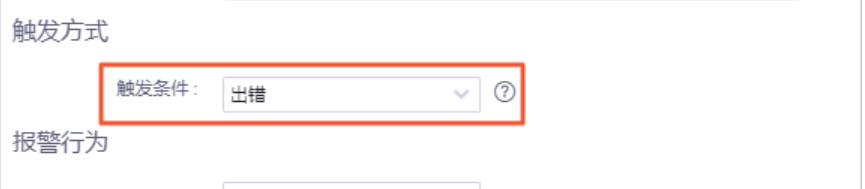
在监控数据产出的及时性前，首先需要确定调度任务的优先级。数据资产等级越高的任务节点，优先级越高，我们可以给予更加严格的数据及时性监控和告警规则。

###### 数据及时性监控使用演示（透过实践了解数据智能监控的功能，不要拘泥于操作具体步骤）

* 1. 在DataStudio页面单击**运维中心（工作流）**/在DataWorks控制台选中一个项目空间，该项目下会有数据集成、数据开发、数据地图、运维中心等功能集，**点击运维中心；**
  2. 在**运维中心**页面，单击左侧导航栏上的**智能监控 > 规则管理；**
  3. 单击右上角的**新建自定义规则**，输入参数后单击**确定**即可。在本例中，监控整个业务流程每次运行时间不可超过30分钟。如果运行时间超过30分钟，则上报1次告警。连续上报3次告警，系统自动以邮件及短信的方式来上报。



* 1. 对于重要的任务节点，您还可以单独设置任务节点规则，并定义其他**触发条件**。



* 1. 数据及时性优化：通常，影响数据按时产出的主要原因和优化方式如下表所示。

