运营分析大数据平台数据仓库架构

# 1、业务需求

基于现有包含HDFS,Spark,KuDu,Kylin,Hive,Yarn,Zookeeper,HBase组件的大数据平台上，构建数据仓库，能完成运营数据的如下功能：

1. 整合公司所有运营数据，建立统一的数据中心
2. 产生运营业务报表，用于领导作出决策
3. 为业务部门提供运营的数据支持
4. 可以作为各个运营业务的数据源，形成业务数据互相反馈的良性循环
5. 分析业务运营数据，通过数据挖掘来降低业务运营成本
6. 开发数据产品，直接或间接地为公司盈利

# 2、数据仓库定义

## 2.1数据仓库的概念

数据仓库是一个面向主题的、集成的、随时间变化的、但信息本身相对稳定的数据集合，用于对管理决策过程的支持。

## 2.2数据仓库的特点

数据仓库特点

1. 面向主题：数据仓库都是基于某个明确主题，仅需要与该主题相关的数据，其他的无关细节数据将被排除掉。
2. 集成的：从不同的数据源采集数据到同一个数据源，此过程会有一些ETL操作。
3. 随时间变化：关键数据隐式或显式的基于时间变化。
4. 信息本身相对稳定：数据装入以后一般只进行查询操作，没有传统数据库的增删改操作。

数据仓库就是整合多个数据源的历史数据进行细粒度的、多维的分析，帮助高层管理者或者业务分析人员做出商业战略决策或商业报表。

# 3大数据平台数据仓库构建

## 3.1大数据平台数据仓库模型

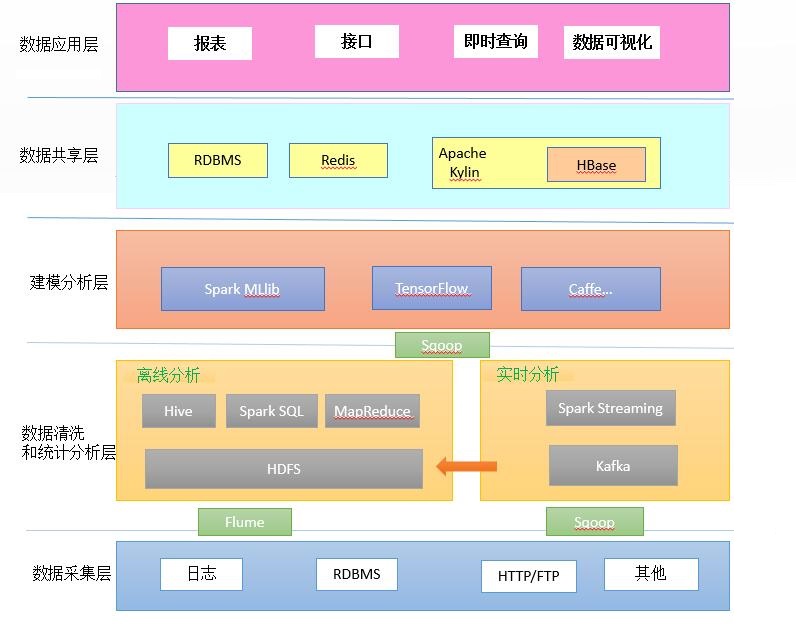
数据应用层:数据应用层承担数据应用的功能，包括对报表生成，数据接口的提供，数据即时查询和数据可视化。

数据共享层:数据共享层主要用来保存数据建模分析、数据统计分析的结果和运行数据的即时数据查询。数据保存的路径包括关系型数据库、Redis和Hbase。

建模分析层:通过不同的分析需求对象，构建数据数据主题，然后通过Spark MLlib、Tensor Flow、Caffe等机器学习或者深度学习框架对分析的数据做建模分析。

数据清洗和统计分析层：对于实时数据分析采用Spark Streaming消费kafka中的数据，对数据做实时数据分析。对于离线数据分析，通常通过构建Hive

数据采集层：通过多个数据源通过flume，Sqoop大数据导入工具将日志数据，RDBMS，HTTP/FTP和其他数据源的放入kafka或者HDFS上。

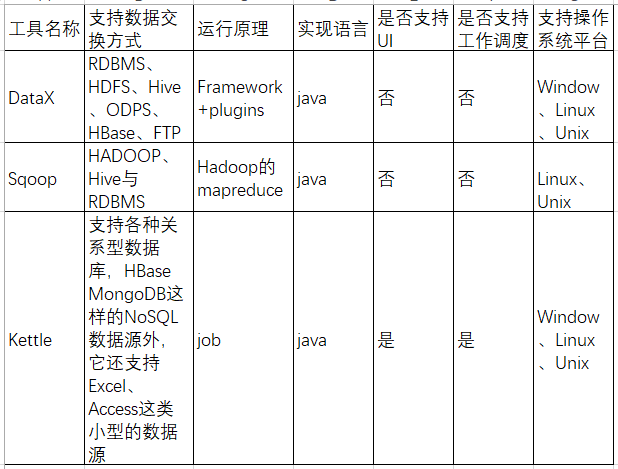


## 3.2数据仓库构建流程



## 3.3大数据数据仓库技术选型

### 3.3.1 ETL工具选型



通过上述技术选型的对比，大数据数据仓库的ETL工具为Kettle。

### 3.3.2 大数据开源OLAP引擎对比

通过查询相关资料对主流开源的大数据OLAP工具presto,druid和kylin做了对比分析。上述三个使用工具的特征如下：

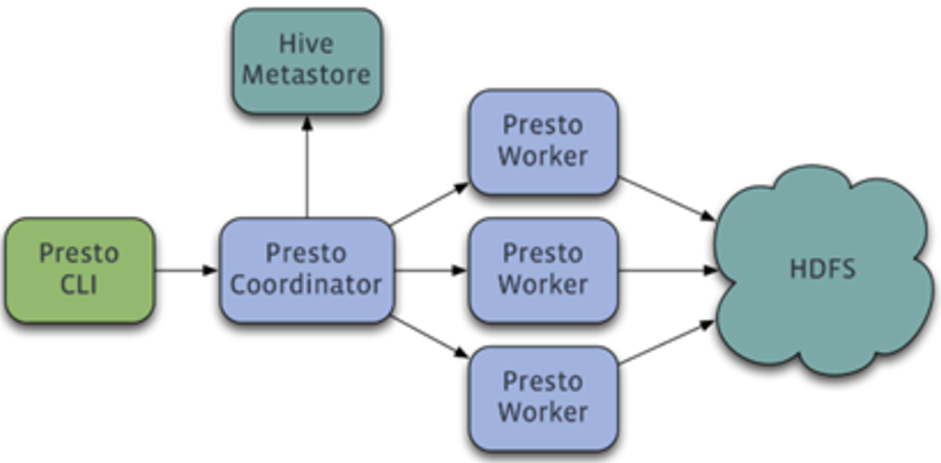
1、presto

Presto是Facebook开发的分布式大数据SQL查询引擎，专门进行快速数据分析。

特点：

* 可以将多个数据源的收据进行合并，可以跨越整个组织进行分析。
* 直接从HDFS读取数据，在使用前不需要大量的ETL操作

架构图:



2、Druid

Druid是广告分析公司Metamarkets开发的一个用于大数据实时查询和分析的分布式实时处理系统，主要用于广告分析，互联网广告系统监控、度量和网络监控

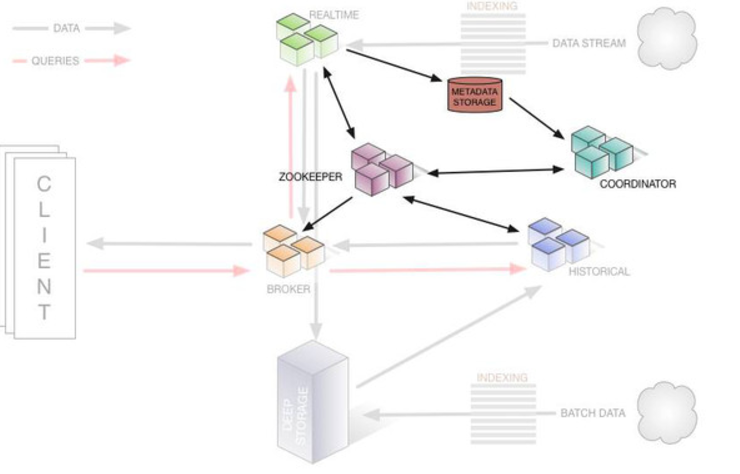
特点：

* 快速的交互式查询—Druid的低延迟数据摄取架构允许事件在它们创建后毫秒内可被查询到
* 高可用性---Druid的数据在系统更新时依然可用，规模的扩大和缩小都不会造成数据丢失
* 可拓展性---Druid已实现每天能够处理数十亿事件和TB级数据
* 为分析而设计---Druid是为OLAP工作流的探索性分析而构建，它支持各种过滤、聚合和查询。

应用场景：

* 需要实时查询分析时
* 具有大量数据时，如每天数亿事件的增量，每天数10T数据的增量
* 需要一个高可用、高容错、高性能数据库时
* 需要交互式聚合和快速探究大量数据时

Druid架构图：



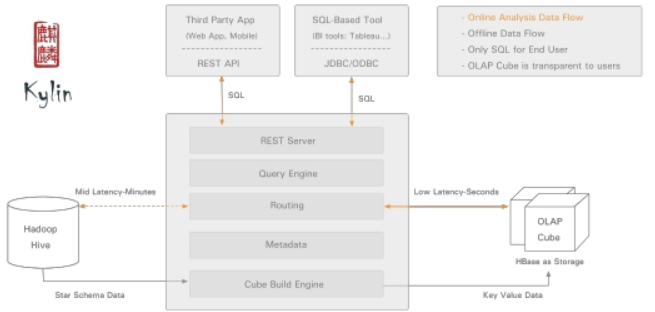
3、Apache kylin

Apache Kylin最初由eBay开发并贡献至开源社区的分布式分析引擎,提供Hadoop之上的SQL查询接口及多维分析(OLAP)能力以支持超大规模数据。

特点：

* 用户为百亿以上数据集定义数据模型并构建立方体
* 亚秒级的查询速度，同时支持高并发
* 为Hadoop提供标准SQL支持大部分查询功能
* 提供与BI工具，如Tableau的整合能力
* 项目及立方体级别的访问控制安全

Apache kylin架构图：



数据成本对比分析：

在查询成本上，Presto是最好的，因为几乎不需要做什么特殊的处理，基本上Hive能读的数据Presto也都能读，所以这个成本非常低。Druid和Kylin的成本相对比较高，因为都需要提前的预计算，尤其是Kylin如果维度数特别多，而且不做特别优化的话，数据存储比较耗资源。

Kylin相对于Presto和druid的优势：

1. 性能非常稳定。因为Kylin依赖的所有的服务，比如Hive，Hbase都是非常成熟的，Kylin本身的逻辑并不复杂，所以稳定性有一个很好的保证
2. 易用。首先是外围的服务，不管是Hive还是Hbase，只要大家用Hadoop系统的话基本都有了，不需要额外工作。在部署运维和使用成本上来讲，都是比较低的。其次，有一个公共的Web页面来做模型的配置。相比之下Druid现在还是基于配置文件来说。这里就有一个问题，配置文件一般都是平台方或者管理员来管理的，没办法把这个配置系统开放出去，这样在沟通成本和响应效率上都不够影响。
3. 活跃开发的社区和热心的核心开发团队，社区里讨论非常开放，大家可以提自己的意见和patch，修复bug以及提交新的功能等。

## 3.4大数据数据仓库构建步骤

数据仓库系统的原始需求不明确，且不断变化与增加，开发者最初不能确切了解到用户的明确而详细的需求，用户所能提供的无非是需求的大的方向以及部分需求， 更不能较准确地预见到以后的需求。因此，采用原型法来进行数据仓库的开发是比较合适的，因为原型法的思想是从构建系统的简单的基本框架着手，不断丰富与完 善整个系统。但是，数据仓库的设计开发又不同于一般意义上的原型法，数据仓库的设计是数据驱动的。这是因为数据仓库是在现存数据库系统基础上进行开发，它 着眼于有效地抽取、综合、集成和挖掘已有数据库的数据资源，服务于企业高层领导管理决策分析的需要。但需要说明的是，数据仓库系统开发是一个经过不断循 环、反馈而使系统不断增长与完善的过程，这也是原型法区别于系统生命周期法的主要特点。因此，在数据仓库的开发的整个过程中，自始至终要求决策人员和开发 者的共同参与和密切协作，要求保持灵活的头脑，不做或尽量少做无效工作或重复工作。

数据仓库的设计大体上可以分为以下几个步骤：

* 概念模型设计；
* 技术准备工作；
* 逻辑模型设计；
* 物理模型设计；
* 数据仓库生成；
* 数据仓库运行与维护。

下面我们六个主要设计步骤为主线，介绍在各个设计步骤中设计的基本内容。

### 3.4.1概念模型设计

进行概念模型设计所要完成的工作是：

1、界定系统边界

2、确定主要的主题域及其内容

概念模型设计的成果是，在原有的数据库的基础上建立了一个较为稳固的概念模型。因为数据仓库是对原有数据库系统中的数据进行集成和重组而形成的数据集合， 所以数据仓库的概念模型设计，首先要对原有数据库系统加以分析理解，看在原有的数据库系统中“有什么”、“怎样组织的”和“如何分布的”等，然后再来考虑 应当如何建立数据仓库系统的概念模型。一方面，通过原有的数据库的设计文档以及在数据字典中的数据库关系模式，可以对企业现有的数据库中的内容有一个完整 而清晰的认识；另一方面，数据仓库的概念模型是面向企业全局建立的，它为集成来自各个面向应用的数据库的数据提供了统一的概念视图。

概念模型的设计是在较高的抽象层次上的设计，因此建立概念模型时不用考虑具体技术条件的限制。

1、界定系统的边界

数据仓库是面向决策分析的数据库，我们无法在数据仓库设计的最初就得到详细而明确的需求，但是一些基本的方向性的需求还是摆在了设计人员的面前：

* 要做的决策类型有哪些?
* 决策者感兴趣的是什么问题?
* 这些问题需要什么样的信息?
* 要得到这些信息需要包含原有数据库系统的哪些部分的数据?

这样，我们可以划定一个当前的大致的系统边界，集中精力进行最需要的部分的开发。因而，从某种意义上讲，界定系统边界的工作也可以看作是数据仓库系统设计的需求分析，因为它将决策者的数据分析的需求用系统边界的定义形式反映出来。

2. 确定主要的主题域

在这一步中，要确定系统所包含的主题域，然后对每个主题域的内容进行较明确的描述，描述的内容包括：

* 主题域的公共码键；
* 主题域之间的联系；
* 充分代表主题的属性组。

### 3.4.2技术准备工作

这一阶段的工作包括：技术评估，技术环境准备。

这一阶段的成果是：技术评估报告、软硬件配置方案、系统(软、硬件)总体设计方案。管理数据仓库的技术要求与管理操作型环境中的数据与处理的技术要求区别 很大，两者所考虑的方面也不同。我们之所以在一般情况下总是将分析型数据与操作型数据分离开来，将分析型数据单独集中存放，也就是用数据仓库来存放，技术 要求上的差异是一个重要原因。

1. 技术评估

进行技术评估，就是确定数据仓库的各项性能指标。一般情况下，需要在这一步里确定的性能指标包括：

* 管理大数据量数据的能力；
* 进行灵活数据存取的能力；
* 根据数据模型重组数据的能力；
* 透明的数据发送和接收能力；
* 周期性成批装载数据的能力；
* 可设定完成时间的作业管理能力。

2. 技术环境准备

一旦数据仓库的体系化结构的模型大体建好后，下一步的工作就是确定我们应该怎样来装配这个体系化结构模型，主要是确定对软硬件配置的要求；我们主要考虑相关的问题：

* 预期在数据仓库上分析处理的数据量有多大?
* 如何减少或减轻竞争性存取程序的冲突?
* 数据仓库的数据量有多大?
* 进出数据仓库的数据通信量有多大?等等。

根据这些考虑，我们就可以确定各项软硬件的配备要求，并且在这一步工作结束时各项技术准备工作应已就绪，可以装载数据了。这些配备有：

* 直接存取设备(DASD)；
* 网络；
* 管理直接存取设备(DASD)的操作系统；
* 进出数据仓库的界面(主要是数据查询和分析工具)；
* 管理数据仓库的软件，目前即选用数据库管理系统及有关的选件，购买的DBMS产品不能满足管理数据仓库需要的，还应考虑自己或软件集成商开发有关模块等等。

### 3.4.3逻辑模型设计

在这一步里进行的工作主要有：

* 分析主题域，确定当前要装载的主题；
* 确定粒度层次划分；
* 确定数据分割策略；
* 关系模式定义；
* 记录系统定义

逻辑模型设计的成果是，对每个当前要装载的主题的逻辑实现进行定义，并将相关内容记录在数据仓库的元数据中，包括：

* 适当的粒度划分；
* 合理的数据分割策略；
* 适当的表划分；
* 定义合适的数据来源等。

1. 分析主题域

在概念模型设计中，我们确定了几个基本的主题域，但是，数据仓库的设计方法是一个逐步求精的过程，在进行设计时，一般是一次一个主题或一次若干个主题地逐 步完成的。所以，我们必须对概念模型设计步骤中确定的几个基本主题域进行分析，并选择首先要实施的主题域。选择第一个主题域所要考虑的是它要足够大，以便 使得该主题域能建设成为一个可应用的系统；它还要足够小，以便于开发和较快地实施。如果所选择的主题域很大并且很复杂，我们甚至可以针对它的一个有意义的 子集来进行开发。在每一次的反馈过程中，都要进行主题域的分析。

2. 粒度层次划分

数据仓库逻辑设计中要解决的一个重要问题是决定数据仓库的粒度划分层次，粒度层次划分适当与否直接影响到数据仓库中的数据量和所适合的查询类型。确定数据 仓库的粒度划分，可以使用在粒度划分一节中介绍的方法，通过估算数据行数和所需的DASD数，来确定是采用单一粒度还是多重粒度，以及粒度划分的层次。

3. 确定数据分割策略

在这一步里，要选择适当的数据分割的标准，一般要考虑以下几方面因素：数据量(而非记录行数)、数据分析处理的实际情况、简单易行以及粒度划分策略等。数 据量的大小是决定是否进行数据分割和如何分割的主要因素；数据分析处理的要求是选择数据分割标准的一个主要依据，因为数据分割是跟数据分析处理的对象紧密 联系的；我们还要考虑到所选择的数据分割标准应是自然的、易于实施的：同时也要考虑数据分割的标准与粒度划分层次是适应的。

4. 关系模式定义

数据仓库的每个主题都是由多个表来实现的，这些表之间依靠主题的公共码键联系在一起，形成一个完整的主题。在概念模型设计时，我们就确定了数据仓库的基本 主题，并对每个主题的公共码键、基本内容等做了描述在这一步里，我们将要对选定的当前实施的主题进行模式划分，形成多个表，并确定各个表的关系模式。

### 3.4.4物理模型设计

这一步所做的工作是确定数据的存储结构，确定索引策略，确定数据存放位置，确定存储分配。

确定数据仓库实现的物理模型，要求设计人员必须做到以下几方面：

* 要全面了解所选用的数据库管理系统，特别是存储结构和存取方法。
* 了解数据环境、数据的使用频度、使用方式、数据规模以及响应时间要求等，这些是对时间和空间效率进行平衡和优化的重要依据。
* 了解外部存储设备的特性，如分块原则，块大小的规定，设备的I／O特性等。

1. 确定数据的存储结构

一个数据库管理系统往往都提供多种存储结构供设计人员选用，不同的存储结构有不同的实现方式，各有各的适用范围和优缺点，设计人员在选择合适的存储结构时应该权衡三个方面的主要因素：存取时间、存储空间利用率和维护代价。

2. 确定索引策略

数据仓库的数据量很大，因而需要对数据的存取路径进行仔细的设计和选择。由于数据仓库的数据都是不常更新的，因而可以设计多种多样的索引结构来提高数据存取效率。

在数据仓库中，设计人员可以考虑对各个数据存储建立专用的、复杂的索引，以获得最高的存取效率，因为在数据仓库中的数据是不常更新的，也就是说每个数据存储是稳定的，因而虽然建立专用的、复杂的索引有一定的代价，但一旦建立就几乎不需维护索引的代价。

3. 确定数据存放位置

我们说过，同一个主题的数据并不要求存放在相同的介质上。在物理设计时，我们常常要按数据的重要程度、使用频率以及对响应时间的要求进行分类，并将不同类 的数据分别存储在不同的存储设备中。重要程度高、经常存取并对响应时间要求高的数据就存放在高速存储设备上，如硬盘；存取频率低或对存取响应时间要求低的 数据则可以放在低速存储设备上，如磁盘或磁带。

数据存放位置的确定还要考虑到其它一些方法，如：决定是否进行合并表；是否对一些经常性的应用建立数据序列；对常用的、不常修改的表或属性是否冗余存储。如果采用了这些技术，就要记入元数据。

4. 确定存储分配

许多数据库管理系统提供了一些存储分配的参数供设计者进行物理优化处理，如：块的尺寸、缓冲区的大小和个数等等，它们都要在物理设计时确定。这同创建数据库系统时的考虑是一样的。

### 3.4.5数据仓库的生成

在这一步里所要做的工作是接口编程，数据装入。

这一步工作的成果是，数据已经装入到数据仓库中，可以在其上建立数据仓库的应用，即DSS应用。

1、设计接口

将操作型环境下的数据装载进入数据仓库环境，需要在两个不同环境的记录系统之间建立一个接口。乍一看，建立和设计这个接口，似乎只要编制一个抽取程序就可 以了，事实上，在这一阶段的工作中，的确对数据进行了抽取，但抽取并不是全部的工作，这一接口还应具有以下的功能：

* 从面向应用和操作的环境生成完整的数据；
* 数据的基于时间的转换；
* 数据的凝聚；
* 对现有记录系统的有效扫描，以便以后进行追加。
* 当然，考虑这些因素的同时，还要考虑到物理设计的一些因素和技术条件限制，根据这些内容，严格地制定规格说明，然后根据规格说明，进行接口编程。从操作型 环境到数据仓库环境的数据接口编程的过程和一般的编程过程并无区别，它也包括伪码开发、编码、编译、检错、测试等步骤。

在接口编程中，要注意：

* 保持高效性，这也是一般的编程所要求的；
* 要保存完整的文档记录；
* 要灵活，易于改动；
* 要能完整、准确地完成从操作型环境到数据仓库环境的数据抽取、转换与集成。

2、数据装入

在这一步里所进行的就是运行接口程序，将数据装入到数据仓库中。主要的工作是：

* 确定数据装入的次序；
* 清除无效或错误数据；
* 数据“老化”；
* 数据粒度管理；
* 数据刷新等。

最初只使用一部分数据来生成第一个主题域，使得设计人员能够轻易且迅速地对已做工作进行调整，而且能够尽早地提交到下一步骤，即数据仓库的使用和维护。这 样既可以在经济上最快地得到回报，又能够通过最终用户的使用、尽早发现一些问题并提出新的需求，然后反馈给设计人员，设计人员继续对系统改进、扩展。

### 3.4.6数据仓库的使用和维护

在这一步中所要做的工作有建立DSS应用，即使用数据仓库理解需求，调整和完善系统，维护数据仓库。

建立企业的体系化环境，不仅包括建立起操作型和分析型的数据环境，还应包括在这一数据环境中建立起企业的各种应用。数据仓库装入数据之后，下一步工作是： 一方面，使用数据仓库中的数据服务于决策分析的目的，也就是在数据仓库中建立起DSS应用；另一方面，根据用户使用情况和反馈来的新的需求，开发人员进一 步完善系统，并管理数据仓库的一些日常活动，如刷新数据仓库的当前详细数据、将过时的数据转化成历史数据、清除不再使用的数据、调整粒度级别等。我们把这 一步骤称为数据仓库的使用与维护。

1. 建立DSS应用

使用数据仓库，即开发DSS应用，与在操作型环境中的应用开发有着本质区别，开发DSS应用不同于联机事务处理应用开发的显著特点在于：

* DSS应用开发是从数据出发的；
* DSS应用的需求不能在开发初期明确了解；
* DSS应用开发是一个不断循环的过程，是启发式的开发。

DSS应用主要可分为两类：例行分析处理和启发式分析处理。例行分析处理是指那些重复进行的分析处理，它通常是属于部门级的应用，如部门统计分析，报表分 析等等；而个人级的分析应用经常是随机性很大的，企业经营者受到某种信息启发而进行的一些即席的分析处理，所以我们称之为启发式的分析处理。

DSS应用开发的大致步骤如下：

步骤l——确定所需的数据。为满足DSS应用的要求，我们必须从数据仓库中确定一个可能用到的数据范围。这是一个试探的过程。

步骤2——编程抽取数据。根据上面得到的数据范围，编写一个抽取程序来获得这些数据。为适应分析需求多变的特点，要求所编写的抽取程序应该通用，易于修改。

步骤3——合并数据。如果有多个数据抽取源，要将抽取来的数据进行合并、提炼，使数据符合分析处理的要求。

步骤4——分析数据。在上步准备好的数据基础上进行分析处理，并看所得的结果是否满足了原始的要求，如果不能满足，则返回步骤1，开始新的一次循环，否则就准备最终分析结果报告。

步骤5——回答问题。生成最终分析结果报告。—般情况下，最终的分析结果报告是在许多次的循环后得到的，因为一次分析处理很少是在一次循环后就完成的。

步骤6——例行化、一次分析处理的最后、我们要决定是否将在上面已经建立的分析处理例行化。如果建立的分析处理是重复进行的部门级的DSS应用，那么最好 是将它例行化，这样在进行下一次同样的分析处理时，不必再重复上述六步的循环过程。而且，不断地积累这种例行处理，形成一个集合，我们就可以通过组合这些 已有的处理来生成新的一个较大的复杂处理，或完成一个复杂处理的一部分。

2. 理解需求，改善和完善系统，维护数据仓库

数据仓库的开发是逐步完善的原型法的开发方法，它要求：要尽快地让系统运行起来，尽早产生效益；要在系统运行或使用中，不断地理解需求，改善系统；不断地考虑新的需求，完善系统。

维护数据仓库的工作主要是管理日常数据装入的工作，包括刷新数据仓库的当前详细数据，将过时的数据转化成历史数据．清除不再使用的数据，管理元数据，等等；另外，如何利用接口定期从操作型环境向数据仓库追加数据，确定数据仓库的数据刷新频率等等。

CMDB

实体关系

Kylin

OpenTS DB

Superset

查看原始日志格式