大数据离线阶段

05

一、 课程计划

目录

二、 数据仓库

1. 数据仓库的基本概念

数据仓库，英文名称为Data Warehouse，可简写为DW或DWH。数据仓库的目的是构建面向分析的集成化数据环境，为企业提供决策支持（Decision Support）。它出于分析性报告和决策支持目的而创建。

数据仓库本身并不“生产”任何数据，同时自身也不需要“消费”任何的数据，数据来源于外部，并且开放给外部应用，这也是为什么叫“仓库”，而不叫“工厂”的原因。

2. 数据仓库的主要特征

数据仓库是面向主题的（Subject-Oriented ）、集成的（Integrated）、非易失的（Non-Volatile）和时变的（Time-Variant ）数据集合，用以支持管理决策 。

2.1. 面向主题

传统数据库中，最大的特点是面向应用进行数据的组织，各个业务系统可能是相互分离的。而数据仓库则是面向主题的。主题是一个抽象的概念，是较高层次上企业信息系统中的数据综合、归类并进行分析利用的抽象。在逻辑意义上，它是对应企业中某一宏观分析领域所涉及的分析对象。

操作型处理（传统数据）对数据的划分并不适用于决策分析。而基于主题组织的数据则不同，它们被划分为各自独立的领域，每个领域有各自的逻辑内涵但互不交叉，在抽象层次上对数据进行完整、一致和准确的描述。一些主题相关的数据通常分布在多个操作型系统中。

2.2. 集成性

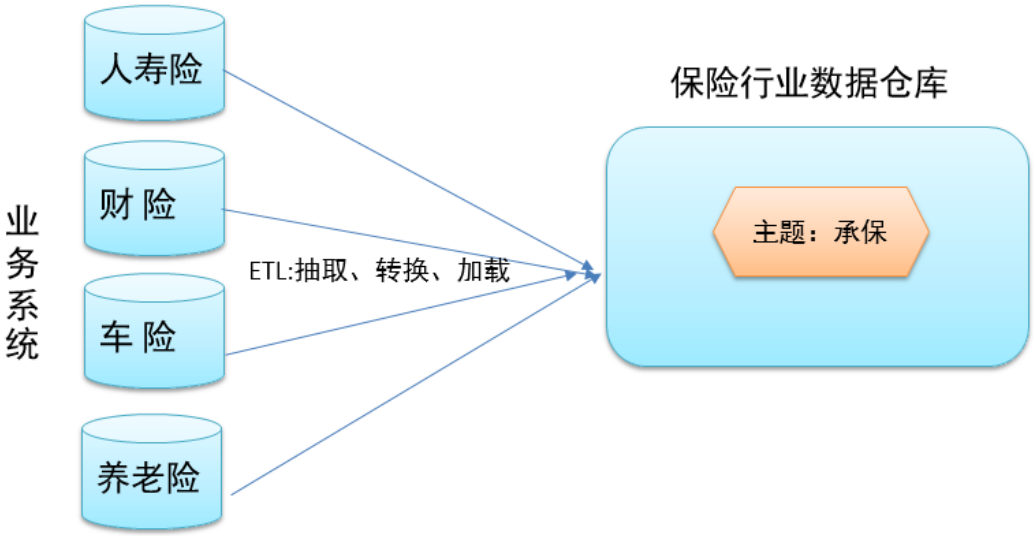
通过对分散、独立、异构的数据库数据进行抽取、清理、转换和汇总便得到了数据仓库的数据，这样保证了数据仓库内的数据关于整个企业的一致性。

数据仓库中的综合数据不能从原有的数据库系统直接得到。因此在数据进入数据仓库之前，必然要经过统一与综合，这一步是数据仓库建设中最关键、最复杂的一步，所要完成的工作有：

（1）要统一源数据中所有矛盾之处，如字段的同名异义、异名同义、单位不统一、字长不一致，等等。

（2）进行数据综合和计算。数据仓库中的数据综合工作可以在从原有数据库抽取数据时生成，但许多是在数据仓库内部生成的，即进入数据仓库以后进行综合生成的。

下图说明一个保险公司综合数据的简单处理过程，其中数据仓库中与“保险”主题有关的数据来自于多个不同的操作型系统。这些系统内部数据的命名可能不同，数据格式也可能不同。把不同来源的数据存储到数据仓库之前，需要去除这些不一致。



2.3. 非易失性（不可更新性）

操作型数据库主要服务于日常的业务操作，使得数据库需要不断地对数据实时更新，以便迅速获得当前最新数据，不至于影响正常的业务运作。在数据仓库中只要保存过去的业务数据，不需要每一笔业务都实时更新数据仓库，而是根据商业需要每隔一段时间把一批较新的数据导入数据仓库。

数据仓库的数据反映的是一段相当长的时间内历史数据的内容，是不同时点的数据库快照的集合，以及基于这些快照进行统计、综合和重组的导出数据。

数据非易失性主要是针对应用而言。数据仓库的用户对数据的操作大多是数据查询或比较复杂的挖掘，一旦数据进入数据仓库以后，一般情况下被较长时间保留。数据仓库中一般有大量的查询操作，但修改和删除操作很少。因此，数据经加工和集成进入数据仓库后是极少更新的，通常只需要定期的加载和更新。

2.4. 时变性

数据仓库包含各种粒度的历史数据。数据仓库中的数据可能与某个特定日期、星期、月份、季度或者年份有关。数据仓库的目的是通过分析企业过去一段时间业务的经营状况，挖掘其中隐藏的模式。虽然数据仓库的用户不能修改数据，但并不是说数据仓库的数据是永远不变的。分析的结果只能反映过去的情况，当业务变化后，挖掘出的模式会失去时效性。因此数据仓库的数据需要更新，以适应决策的需要。从这个角度讲，数据仓库建设是一个项目，更是一个过程 。数据仓库的数据随时间的变化表现在以下几个方面。

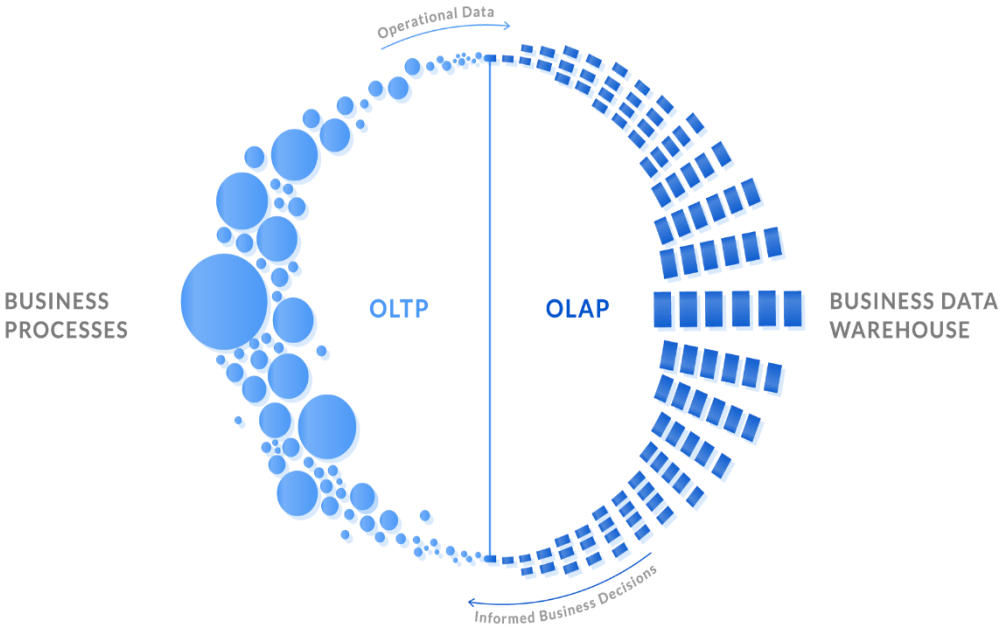
（1）数据仓库的数据时限一般要远远长于操作型数据的数据时限。

（2）操作型系统存储的是当前数据，而数据仓库中的数据是历史数据。

（3）数据仓库中的数据是按照时间顺序追加的，它们都带有时间属性。

3. 数据仓库与数据库区别

数据库与数据仓库的区别实际讲的是 OLTP 与 OLAP 的区别。



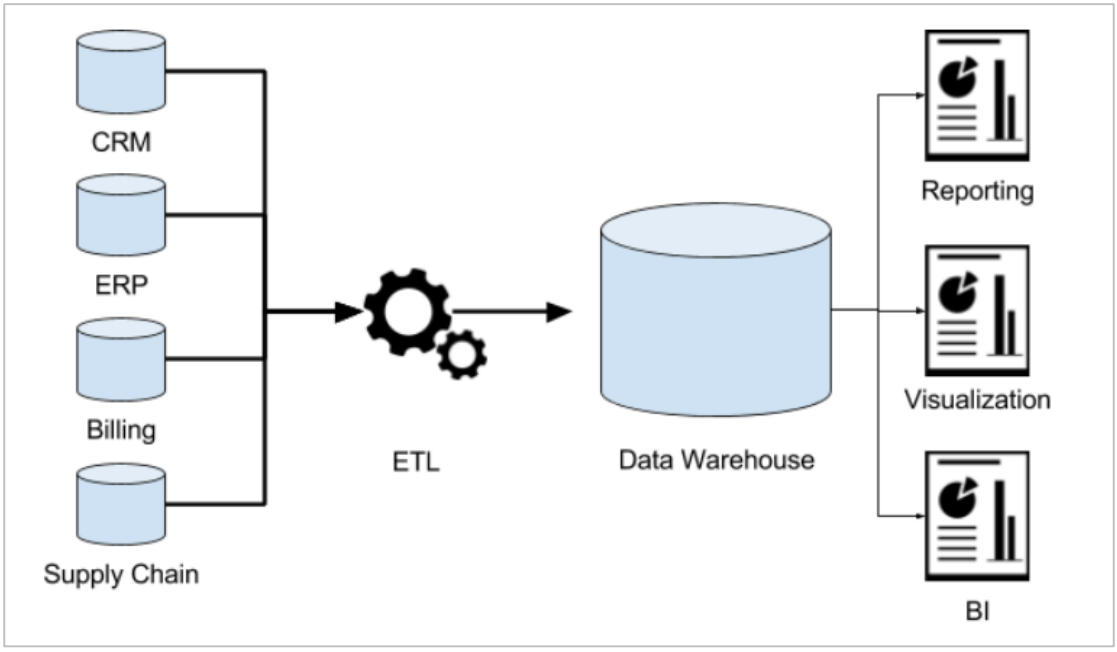
操作型处理，叫联机事务处理 OLTP（On-Line Transaction Processing,），也可以称面向交易的处理系统，它是针对具体业务在数据库联机的日常操作，通常对少数记录进行查询、修改。用户较为关心操作的响应时间、数据的安全性、完整性和并发支持的用户数等问题。传统的数据库系统作为数据管理的主要手段，主要用于操作型处理。

分析型处理，叫联机分析处理 OLAP（On-Line Analytical Processing）一般针对某些主题的历史数据进行分析，支持管理决策。

首先要明白，数据仓库的出现，并不是要取代数据库。

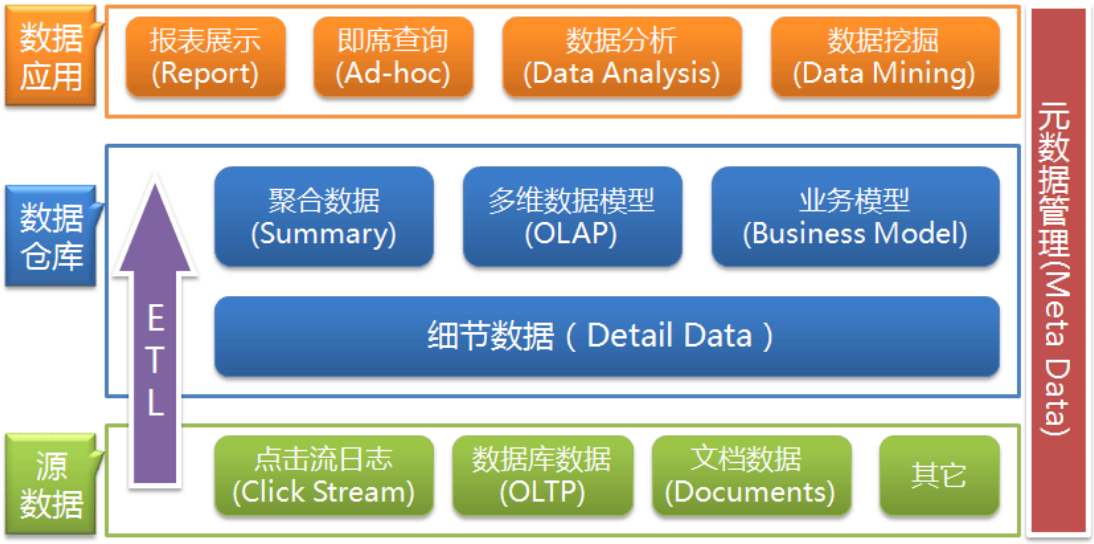
- 数据库是面向事务的设计，数据仓库是面向主题设计的。
- 数据库一般存储业务数据，数据仓库存储的一般是历史数据。
- 数据库设计是尽量避免冗余，一般针对某一业务应用进行设计，比如一张简单的 User 表，记录用户名、密码等简单数据即可，符合业务应用，但是不符合分析。数据仓库在设计是有意引入冗余，依照分析需求，分析维度、分析指标进行设计。
- 数据库是为捕获数据而设计，数据仓库是为分析数据而设计。

数据仓库，是在数据库已经大量存在的情况下，为了进一步挖掘数据资源、为了决策需要而产生的，它决不是所谓的“大型数据库”。



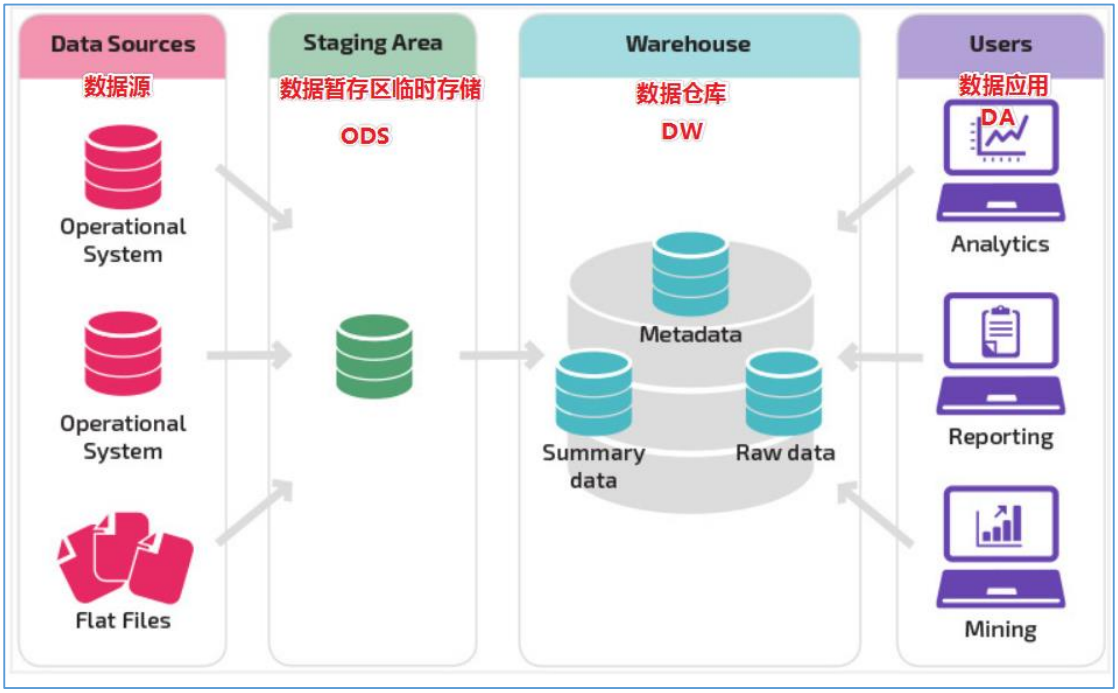
4. 数据仓库分层架构

按照数据流入流出的过程，数据仓库架构可分为三层——**源数据、数据仓库、数据应用。**



数据仓库的数据来源于不同的源数据，并提供多样的数据应用，数据自下而上流入数据仓库后向上层开放应用，而数据仓库只是中间集成化数据管理的一个平台。

- 源数据层（ODS）：此层数据无任何更改，直接沿用外围系统数据结构和数据，不对外开放；为临时存储层，是接口数据的临时存储区域，为后一步的数据处理做准备。
- 数据仓库层（DW）：也称为细节层，DW 层的数据应该是一致的、准确的、干净的数据，即对源系统数据进行了清洗（去除了杂质）后的数据。
- 数据应用层（DA 或 APP）：前端应用直接读取的数据源；根据报表、专题分析需求而计算生成的数据。



数据仓库从各数据源获取数据及在数据仓库内的数据转换和流动都可以认为是 ETL（抽取 Extra，转化 Transfer，装载 Load）的过程，ETL 是数据仓库的流水线，也可以认为是数据仓库的血液，它维系着数据仓库中数据的新陈代谢，而数据仓库日常的管理和维护工作的大部分精力就是保持 ETL 的正常和稳定。

为什么要对数据仓库分层？

用空间换时间，通过大量的预处理来提升应用系统的用户体验（效率），因此数据仓库会存在大量冗余的数据；不分层的话，如果源业务系统的业务规则发生变化将会影响整个数据清洗过程，工作量巨大。

通过数据分层管理可以简化数据清洗的过程，因为把原来一步的工作分到了多个步骤去完成，相当于把一个复杂的工作拆成了多个简单的工作，把一个大的黑盒变成了一个白盒，每一层的处理逻辑都相对简单和容易理解，这样我们比较容易保证每一个步骤的正确性，当数据发生错误的时候，往往我们只需要局部调整某个步骤即可。

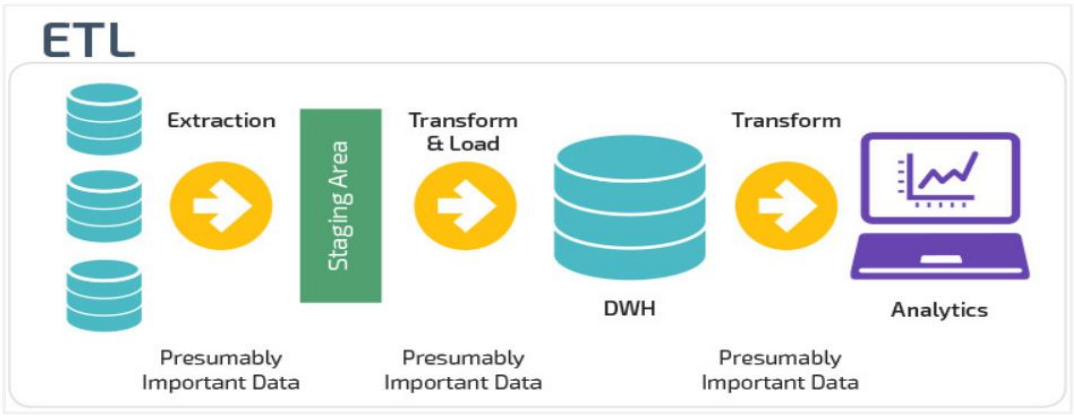
5. ETL、ELT

数据仓库从各数据源获取数据及在数据仓库内的数据转换和流动都可以认为是 ETL（抽取 Extract，转化 Transform，装载 Load）的过程。

但是在实际操作中将数据加载到仓库却产生了两种不同做法：ETL 和 ELT。

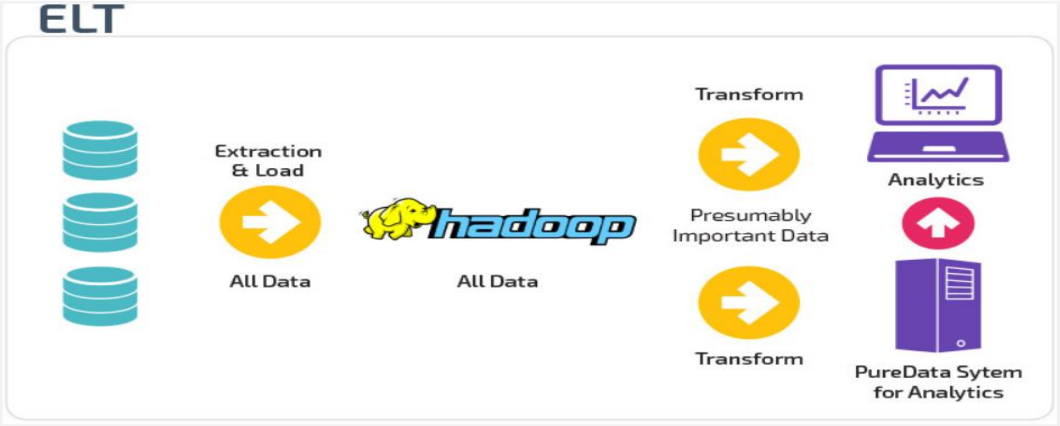
5.1. ETL

首先从数据源池中提取数据，这些数据源通常是事务性数据库。数据保存在临时暂存数据库中（ODS）。然后执行转换操作，将数据结构化并转换为适合目标数据仓库系统的形式。然后将结构化数据加载到仓库中，以备分析。



5.2. ELT

使用 ELT，数据在从数据源中提取后立即加载。没有专门的临时数据库（ODS），这意味着数据会立即加载到单一的集中存储库中。数据在数据仓库系统中进行转换，以便与商业智能工具（BI 工具）一起使用。大数据时代数仓这个特点很明显。



三、 Apache Hive

1. Hive 简介

1.1. 什么是 Hive

Hive 是基于 Hadoop 的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供类 SQL 查询功能。

本质是将 SQL 转换为 MapReduce 程序。

主要用途：用来做离线数据分析，比直接用 MapReduce 开发效率更高。

1.2. 为什么使用 Hive

直接使用 Hadoop MapReduce 处理数据所面临的问题：

人员学习成本太高

MapReduce 实现复杂查询逻辑开发难度太大

使用 Hive ：

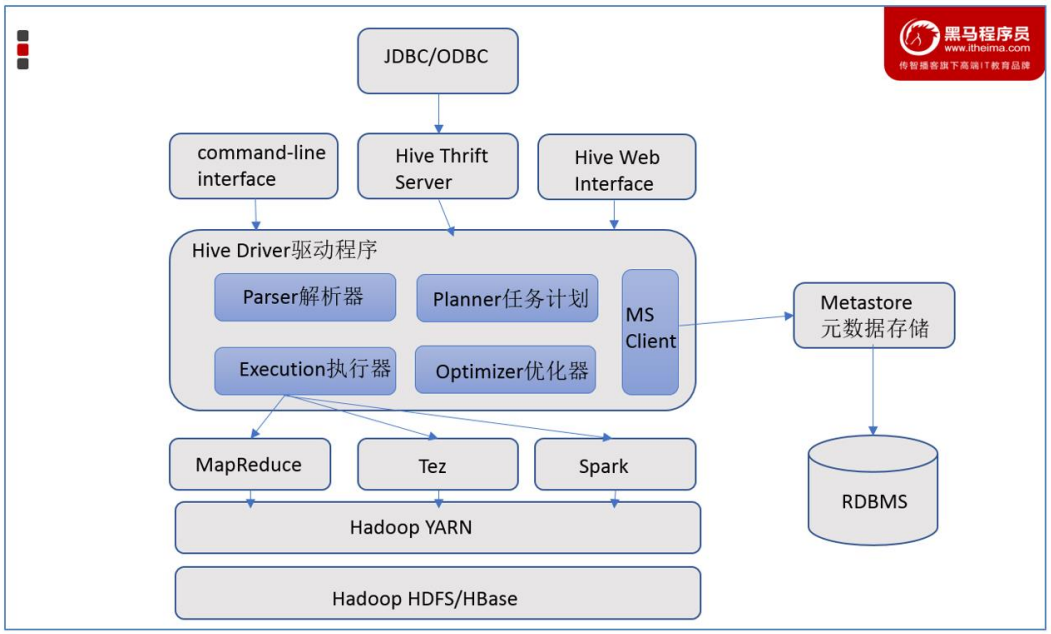
操作接口采用类 SQL 语法，提供快速开发的能力

避免了去写 MapReduce，减少开发人员的学习成本

功能扩展很方便

2. Hive 架构

2.1. Hive 架构图



2.2. Hive 组件

用户接口：包括 CLI、JDBC/ODBC、WebGUI。其中，CLI(command line interface)为 shell 命令行；JDBC/ODBC 是 Hive 的 JAVA 实现，与传统数据库 JDBC 类似；WebGUI 是通过浏览器访问 Hive。

元数据存储：通常是存储在关系数据库如 mysql/derby 中。Hive 将元数据存储在数据库中。Hive 中的元数据包括表的名字，表的列和分区及其属性，表的属性（是否为外部表等），表的数据所在目录等。

解释器、编译器、优化器、执行器:完成 HQL 查询语句从词法分析、语法分析、编译、优化以及查询计划的生成。生成的查询计划存储在 HDFS 中，并在随后有 MapReduce 调用执行。

2.3. Hive 与 Hadoop 的关系

Hive 利用 HDFS 存储数据，利用 MapReduce 查询分析数据。

3． Hive 与传统数据库对比

hive 用于海量数据的离线数据分析。

hive 具有 sql 数据库的外表，但应用场景完全不同，hive 只适合用来做批量数据统计分析。

更直观的对比请看下面这幅图：

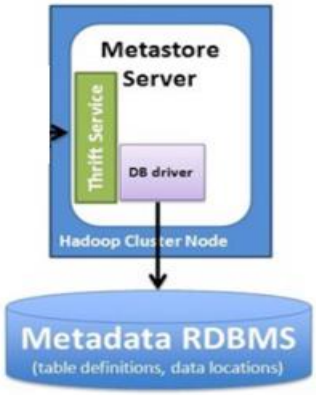
	Hive	RDBMS
查询语言	HQL	SQL
数据存储	HDFS	Raw Device or Local FS
执行	MapReduce	Excutor
执行延迟	高	低
处理数据规模	大	小
索引	0.8版本后加入位图索引	有复杂的索引

四、 Hive 安装部署

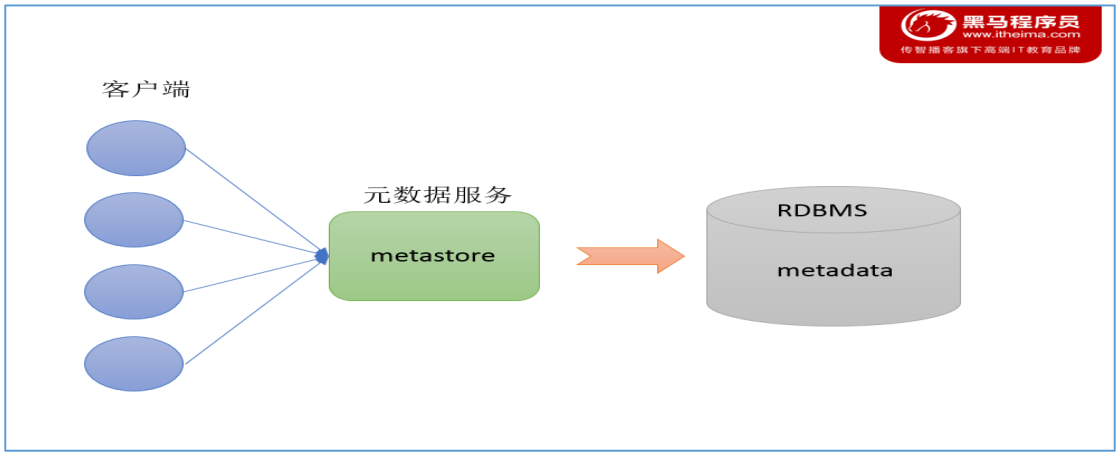
Hive 安装前需要安装好 JDK 和 Hadoop。配置好环境变量。如果需要使用 mysql 来存储元数据，则需要 mysql 也安装好。

1. metadata 、metastore

Metadata 即元数据。元数据包含用 Hive 创建的 database、table、表的字段等元信息。元数据存储在关系型数据库中。如 hive 内置的 Derby、第三方如 MySQL 等。



Metastore 即元数据服务，作用是：客户端连接 metastore 服务，metastore 再去连接 MySQL 数据库来存取元数据。有了 metastore 服务，就可以有多个客户端同时连接，而且这些客户端不需要知道 MySQL 数据库的用户名和密码，只需要连接 metastore 服务即可。



2. metastore 三种配置方式

metastore 服务配置有 3 种模式：内嵌模式、本地模式、远程模式。区分 3 种配置方式的关键是弄清楚两个问题：

- Metastore 服务是否需要单独配置、单独启动？
- Metadata 是存储在内置的 derby 中，还是第三方 RDBMS，比如 Mysql。

	内嵌模式	本地模式	远程模式
Metastore单独配置、启动	否	否	是
Metadata存储介质	Derby	Mysql	Mysql

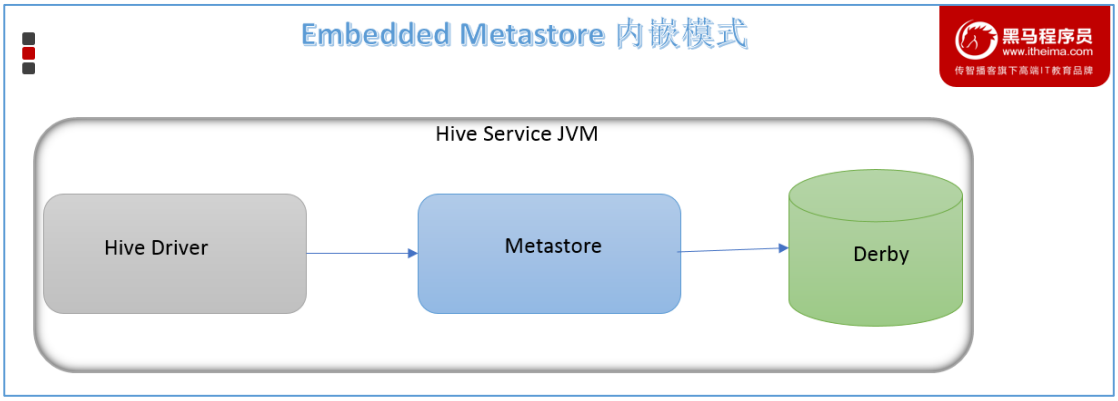
本系列课程中使用企业推荐模式--远程模式部署。

2.1. 内嵌模式

内嵌模式使用的是内嵌的 Derby 数据库来存储元数据，也不需要额外起 Metastore 服务。数据库和 Metastore 服务都嵌入在主 Hive Server 进程中。这个是默认的，配置简单，但是一次只能一个客户端连接，适用于用来实验，不适用于生产环境。

解压 hive 安装包 bin/hive 启动即可使用

缺点：不同路径启动 hive，每一个 hive 拥有一套自己的元数据，无法共享。



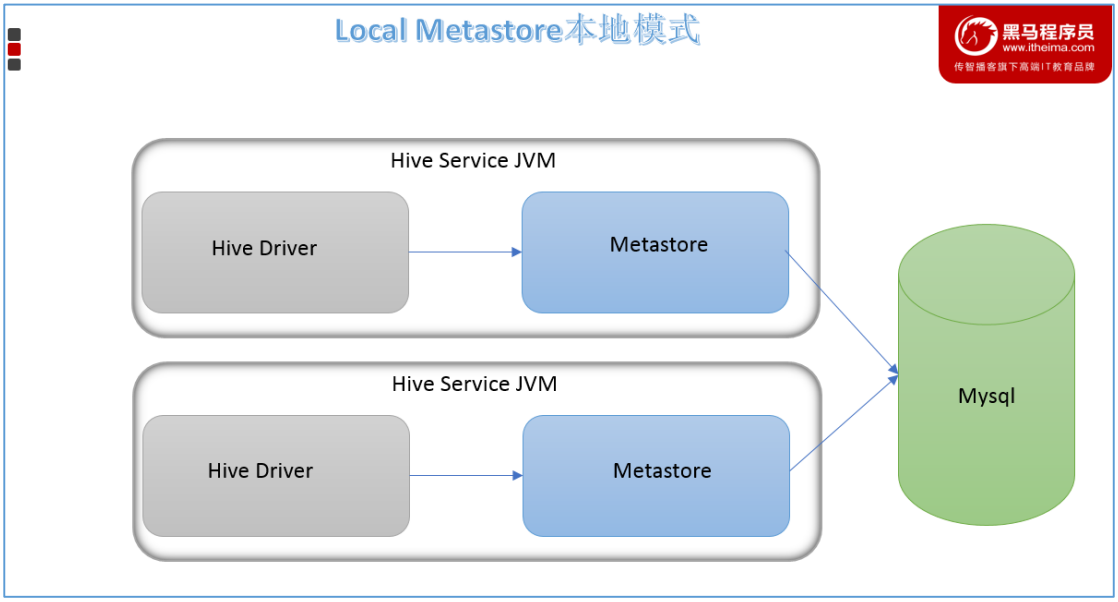
2.2. 本地模式

本地模式采用外部数据库来存储元数据，目前支持的数据库有：MySQL、Postgres、Oracle、MS SQL Server.在这里我们使用 MySQL。

本地模式不需要单独起 metastore 服务，用的是跟 hive 在同一个进程里的 metastore 服务。也就是说当你启动一个 hive 服务，里面默认会帮我们启动一个 metastore 服务。

hive 根据 hive.metastore.uris 参数值来判断，如果为空，则为本地模式。

缺点是：每启动一次 hive 服务，都内置启动了一个 metastore。



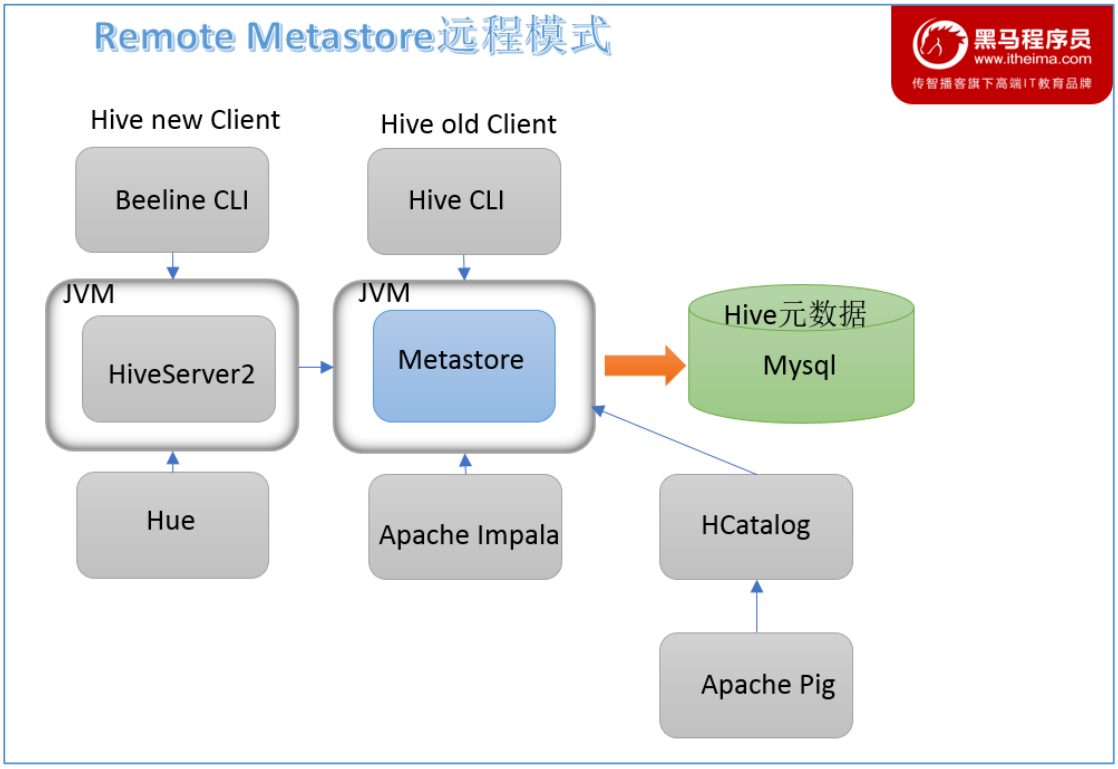
本地模式下 hive 的配置主需要指定 mysql 的相关信息即可。

2.3. 远程模式

远程模式下，需要单独起 metastore 服务，然后每个客户端都在配置文件里配置连接到该 metastore 服务。远程模式的 metastore 服务和 hive 运行在不同的进程里。

在生产环境中，建议用远程模式来配置 Hive Metastore。

在这种情况下，其他依赖 hive 的软件都可以通过 Metastore 访问 hive。



远程模式下，需要配置 hive.metastore.uris 参数来指定 metastore 服务运行的机器 ip 和端口，并且需要单独手动启动 metastore 服务。

3. Hive metastore 远程模式安装部署

课程中采用远程模式部署 hive 的 metastore 服务。在 node1 机器上安装。

注意：以下两件事在启动 hive 之前必须确保正常完成。

1、选择某台机器提前安装 mysql，确保具有远程访问的权限。

2、启动 hadoop 集群 确保集群正常健康

3.1. Hadoop 中添加用户代理配置

```
#修改 hadoop 配置文件 etc/hadoop/core-site.xml,加入如下配置项
<property>
    <name>hadoop.proxyuser.root.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.root.groups</name>
    <value>*</value>
</property>
```

3.2. 上传安装包 并解压

```
tar zxvf apache-hive-3.1.2-bin.tar.gz
mv apache-hive-3.1.2-bin/ hive

# 解决 Hive 与 Hadoop 之间 guava 版本差异
cd /export/server/apache-hive-3.1.2-bin/
rm -rf lib/guava-19.0.jar
cp /export/server/hadoop-3.3.0/share/hadoop/common/lib/guava-27.0-jre.jar \
./lib/
```

3.3. 修改配置文件 hive-env.sh

```
cd /export/server/apache-hive-3.1.2-bin/conf
mv hive-env.sh.template hive-env.sh

vim hive-env.sh
export HADOOP_HOME=/export/server/hadoop-3.3.0
export HIVE_CONF_DIR=/export/server/apache-hive-3.1.2-bin/conf
export HIVE_AUX_JARS_PATH=/export/server/apache-hive-3.1.2-bin/lib
```

3.4. 添加配置文件 hive-site.xml

```
<configuration>
<!-- 存储元数据 mysql 相关配置 -->
<property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://node1:3306/hive3?createDatabaseIfNotExist=true&amp;useSSL=fals
e</value>
</property>

<property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
</property>

<property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
</property>

<property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>hadoop</value>
</property>

<!-- H2S 运行绑定 host -->
<property>
        <name>hive.server2.thrift.bind.host</name>
```

```
    <value>node1</value>
</property>

<!-- 远程模式部署 metastore metastore 地址 -->
<property>
    <name>hive.metastore.uris</name>
    <value>thrift://node1:9083</value>
</property>

<!-- 关闭元数据存储授权 -->
<property>
    <name>hive.metastore.event.db.notification.api.auth</name>
    <value>false</value>
</property>
</configuration>
```

3.5. 上传 MySQL 驱动

```
#上传 mysql jdbc 驱动到 hive 安装包 lib 下

mysql-connector-java-5.1.32.jar
```

3.6. 初始化元数据

```
cd /export/server/apache-hive-3.1.2-bin/

bin/schematool -initSchema -dbType mysql -verbos
#初始化成功会在 mysql 中创建 74 张表
```

3.7. 创建 hive 存储目录

```
hadoop fs -mkdir /tmp
hadoop fs -mkdir -p /user/hive/warehouse
hadoop fs -chmod g+w /tmp
hadoop fs -chmod g+w /user/hive/warehouse
```

4. metastore 的启动方式

```
#前台启动  关闭 ctrl+c
/export/server/apache-hive-3.1.2-bin/bin/hive --service metastore

#前台启动开启 debug 日志
/export/server/apache-hive-3.1.2-bin/bin/hive --service metastore --hiveconf hive.root.logger=DEBUG,console

#后台启动 进程挂起  关闭使用 jps+ kill -9
nohup /export/server/apache-hive-3.1.2-bin/bin/hive --service metastore &
```

5. Hive Client、Beeline Client

5.1. 第一代客户端 Hive Client

在 hive 安装包的 bin 目录下，有 hive 提供的第一代客户端 bin/hive。使用该客户端可以访问 hive 的 metastore 服务。从而达到操作 hive 的目的。

如果需要在其他机器上通过该客户端访问 hive metastore 服务，只需要在该机器的 hive-site.xml 配置中添加 metastore 服务地址即可。

scp 安装包到另一个机器上，比如 node3：

```
scp -r /export/server/apache-hive-3.1.2-bin/ node3:/export/server/
```

vim hive-site.xml 内容如下：

```
<configuration>
<property>
    <name>hive.metastore.uris</name>
    <value>thrift://node1:9083</value>
</property>
</configuration>
```

使用下面的命令启动 hive 的客户端：

/export/server/apache-hive-3.1.2-bin/bin/hive

```
[root@node-1 ~]# /export/servers/hive/bin/hive
2019-07-23 22:24:45,136 WARN  [main] mapreduce.TableMapReduceUtil: The hbase-prefix-
eeCodec is not present.  Continuing without it.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/export/servers/hbase-1.2.1/lib/slf4j-log4j12-1.7.
inder.class]
SLF4J: Found binding in [jar:file:/export/servers/hadoop-2.6.0-cdh5.14.0/share/hadoo
r!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/export/servers/hive/lib/hive-co
j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>
```

可以发现官方提示：第一代客户端已经不推荐使用了。

5.2. 第二代客户端 Hive Beeline Client

hive 经过发展，推出了第二代客户端 beeline，但是 beeline 客户端不是直接访问 metastore 服务的，而是需要单独启动 hiveserver2 服务。

在 hive 运行的服务器上，首先启动 metastore 服务，然后启动 hiveserver2 服务。

```
nohup /export/server/apache-hive-3.1.2-bin/bin/hive --service metastore &
nohup /export/server/apache-hive-3.1.2-bin/bin/hive --service hiveserver2 &
```

在 node3 上使用 beeline 客户端进行连接访问。

```
/export/server/apache-hive-3.1.2-bin/bin/beeline
beeline> ! connect jdbc:hive2://node1:10000
Enter username for jdbc:hive2://node1:10000: root
Enter password for jdbc:hive2://node1:10000: *******
```