Note05 Hive

1. 什么是数仓

1.1. 基本概念

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

1.2. 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。

1.2.1. 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。

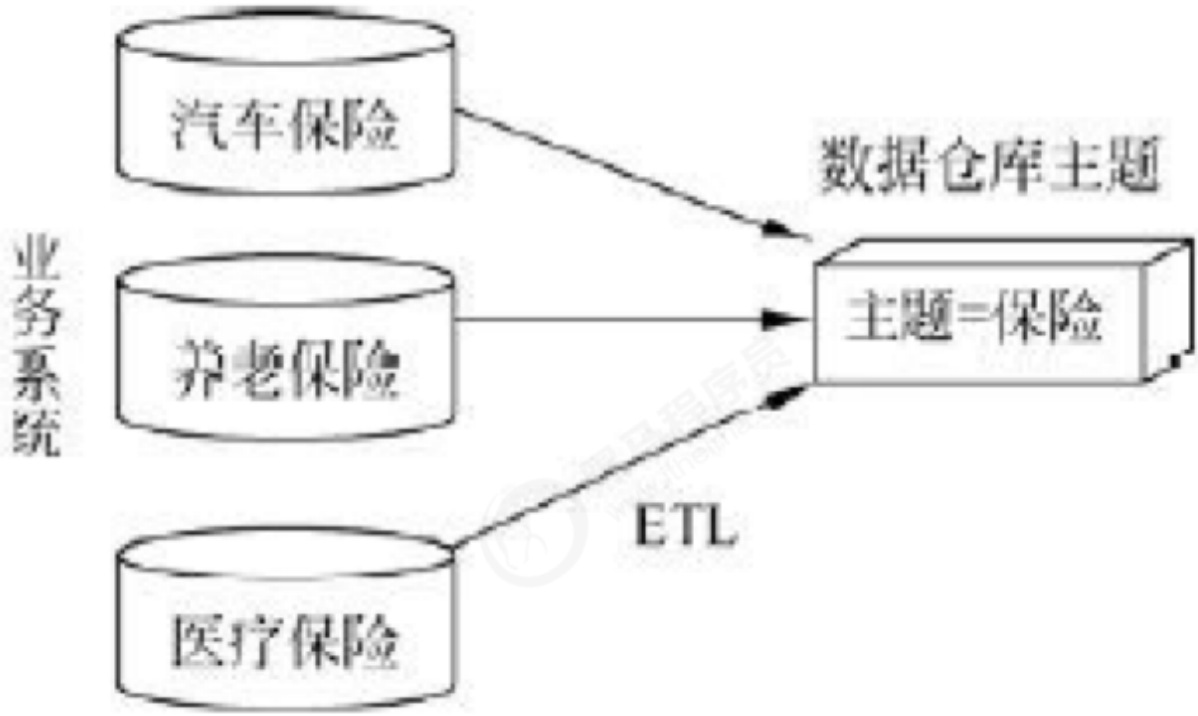
1.2.2. 集成性

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。



1.2.3. 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。

1.2.4. 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。

1.3. 数据库与数据仓库的区别

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。

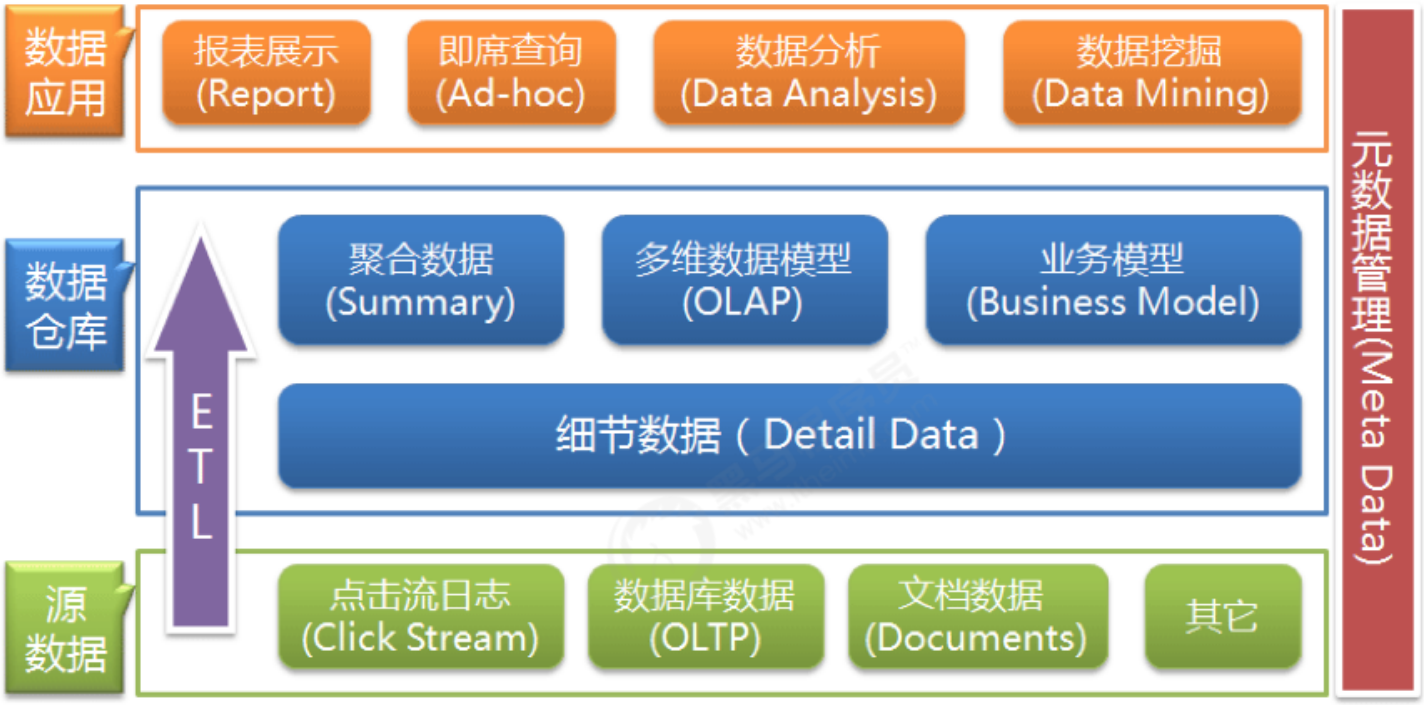
以银行业务为例。数据库是事务系统的数据平台，客户在银行做的每笔交易都会写入数据库，被记录下来，这里，可以简单地理解为用数据库记账。数据仓库是分析系统的数据平台，它从事务系统获取数据，并做汇总、加工，为决策者提供决策的依据。比如，某银行某分行一个月发生多少交易，该分行当前存款余额是多少。如果存款又多，消费交易又多，那么该地区就有必要设立ATM了。

显然，银行的交易量是巨大的，通常以百万甚至千万次来计算。事务系统是实时的，这就要求时效性，客户存一笔钱需要几十秒是无法忍受的，这就要求数据库只能存储很短一段时间的数据。而分析系统是事后的，它要提供关注时间段内所有的有效数据。这些数据是海量的，汇总计算起来也要慢一些，但是，只要能够提供有效的分析数据就达到目的了。

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”。

1.4. 数仓的分层架构

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。

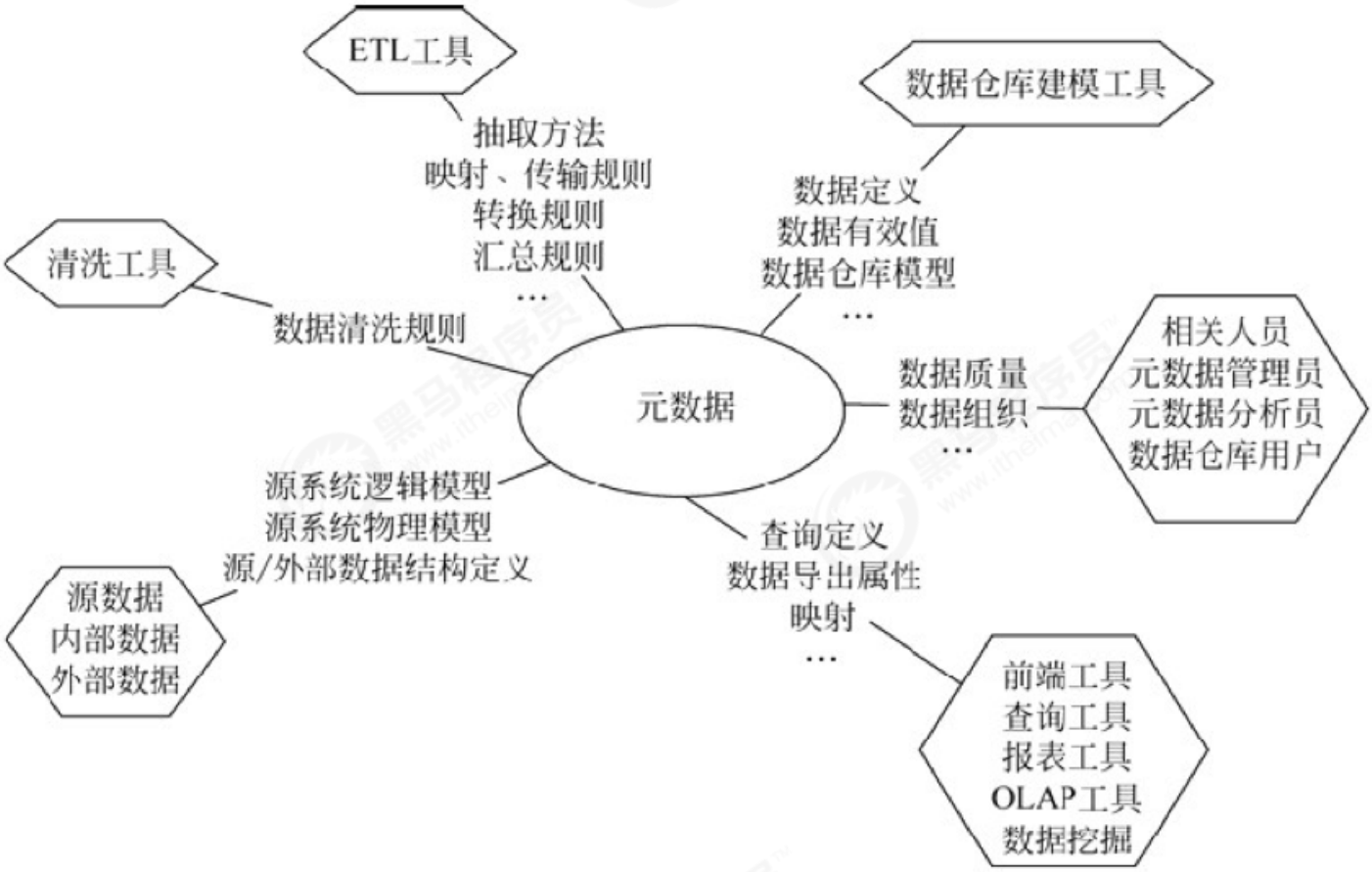
1.5. 数仓的元数据管理

元数据（Meta Date），主要记录数据仓库中模型的定义、各层级间的映射关系、监控数据仓库的数据状态及ETL的任务运行状态。一般会通过元数据资料库（Metadata Repository）来统一地存储和管理

元数据，其主要目的是使数据仓库的设计、部署、操作和管理能达成协同和一致。

元数据是数据仓库管理系统的重要组成部分，元数据管理是企业级数据仓库中的关键组件，贯穿数据仓库构建的整个过程，直接影响着数据仓库的构建、使用和维护。

- 构建数据仓库的主要步骤之一是ETL。这时元数据将发挥重要的作用，它定义了源数据系统到数据仓库的映射、数据转换的规则、数据仓库的逻辑结构、数据更新的规则、数据导入历史记录以及装载周期等相关内容。数据抽取和转换的专家以及数据仓库管理员正是通过元数据高效地构建数据仓库。
- 用户在使用数据仓库时，通过元数据访问数据，明确数据项的含义以及定制报表。
- 数据仓库的规模及其复杂性离不开正确的元数据管理，包括增加或移除外部数据源，改变数据清洗方法，控制出错的查询以及安排备份等。



元数据可分为技术元数据和业务元数据。技术元数据为开发和管理数据仓库的IT 人员使用，它描述了与数据仓库开发、管理和维护相关的数据，包括数据源信息、数据转换描述、数据仓库模型、数据清洗与更新规则、数据映射和访问权限等。而业务元数据为管理层和业务分析人员服务，从业务角度描述数据，包括商务术语、数据仓库中有什么数据、数据的位置和数据的可用性等，帮助业务人员更好地理解数据仓库中哪些数据是可用的以及如何使用。

由上可见，元数据不仅定义了数据仓库中数据的模式、来源、抽取和转换规则等，而且是整个数据仓库系统运行的基础，元数据把数据仓库系统中各个松散的组件联系起来，组成了一个有机的整体。

2. Hive 的基本概念

2.1. Hive 简介

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。

其本质是将SQL转换为MapReduce的任务进行运算，底层由HDFS来提供数据的存储，说白了hive可以理解为一个将SQL转换为MapReduce的任务的工具，甚至更进一步可以说hive就是一个MapReduce的客户端

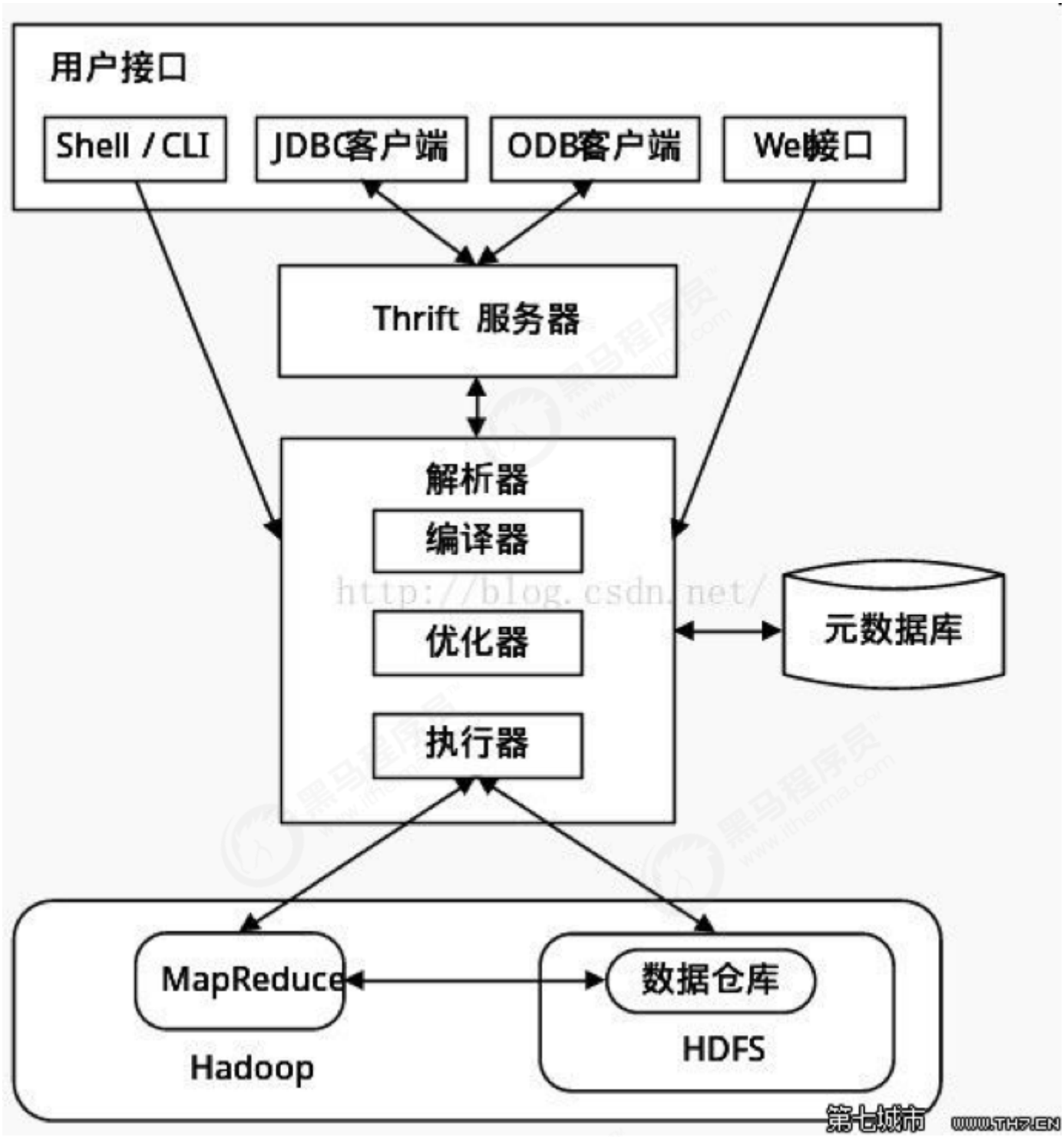
为什么使用 Hive

- 直接使用hadoop所面临的问题
  - 人员学习成本太高
  - 项目周期要求太短
  - MapReduce实现复杂查询逻辑开发难度太大
- 为什么要使用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。

Hive 的特点

- 可扩展
  Hive可以自由的扩展集群的规模，一般情况下不需要重启服务。
- 延展性
  Hive支持用户自定义函数，用户可以根据自己的需求来实现自己的函数。
- 容错
  良好的容错性，节点出现问题SQL仍可完成执行。

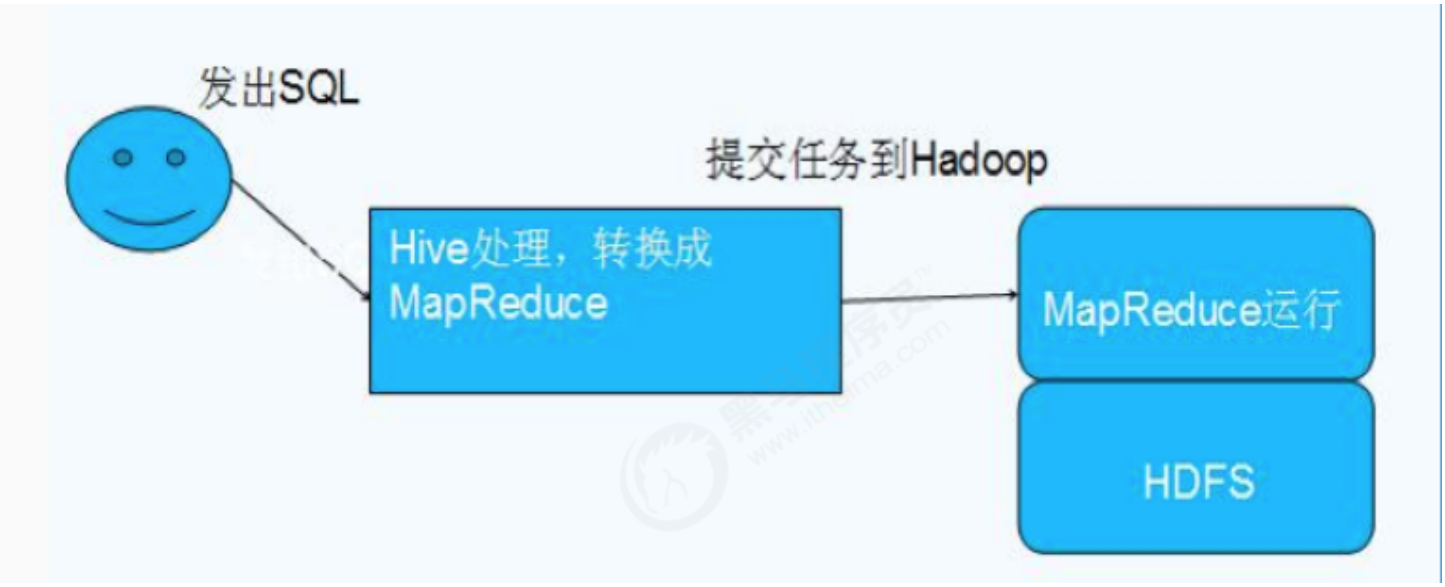
2.2. Hive 架构



- **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



2.4. Hive与传统数据库对比

hive用于海量数据的离线数据分析

	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

总结：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

2.5. Hive 的安装

这里我们选用hive的版本是3.1.0这个release版本，可以兼容我们对应的hadoop3.x的版本
下载地址为：
http://archive.apache.org/dist/hive/hive-3.1.0/apache-hive-3.1.0-bin.tar.gz

下载之后，将我们的安装包上传到第三台机器的/export/softwares目录下面去

第一步：上传并解压安装包

将我们的hive的安装包上传到第三台服务器的/export/softwares路径下，然后进行解压

```
cd /export/softwares/
tar -zxvf apache-hive-3.1.0-bin.tar.gz -C ../servers/
```

第二步：centos6.9安装mysql

第一步：在线安装mysql相关的软件包

```
yum install mysql mysql-server mysql-devel
```

第二步：启动mysql的服务

```
/etc/init.d/mysqld start
```

第三步：通过mysql安装自带脚本进行设置

```
/usr/bin/mysql_secure_installation
```

第四步：进入mysql的客户端然后进行授权

```
grant all privileges on *.* to 'root'@'%' identified by '123456' with grant option;

flush privileges;
```

第三步：修改hive的配置文件

修改hive-env.sh

```
cd /export/servers/apache-hive-3.1.0-bin/conf
cp hive-env.sh.template hive-env.sh


HADOOP_HOME=/export/servers/hadoop-3.1.1
export HIVE_CONF_DIR=/export/servers/apache-hive-3.1.0-bin/conf
```

修改hive-site.xml

```
cd /export/servers/apache-hive-3.1.0-bin/conf
vim hive-site.xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
      <name>javax.jdo.option.ConnectionUserName</name>
      <value>root</value>
  </property>
  <property>
      <name>javax.jdo.option.ConnectionPassword</name>
      <value>123456</value>
  </property>
  <property>
      <name>javax.jdo.option.ConnectionURL</name>
      <value>jdbc:mysql://node03:3306/hive?createDatabaseIfNotExist=true&amp;useSSL=fal
  </property>
  <property>
      <name>javax.jdo.option.ConnectionDriverName</name>
      <value>com.mysql.jdbc.Driver</value>
  </property>
  <property>
      <name>hive.metastore.schema.verification</name>
      <value>false</value>
  </property>
  <property>
    <name>datanucleus.schema.autoCreateAll</name>
    <value>true</value>
 </property>
 <property>
               <name>hive.server2.thrift.bind.host</name>
               <value>node03.hadoop.com</value>
   </property>
<!--
  <property>
    <name>hive.metastore.uris</name>
    <value>thrift://node03:9083</value>
    <description>JDBC connect string for a JDBC metastore</description>
  </property>
  <property>
    <name>hive.metastore.local</name>
    <value>false</value>
    <description>this is local store</description>
  </property>
-->
</configuration>
```

第四步：添加mysql的连接驱动包到hive的lib目录下

hive使用mysql作为元数据存储，必然需要连接mysql数据库，所以我们添加一个mysql的连接驱动包到hive的安装目录下，然后就可以准备启动hive了

将我们准备好的mysql-connector-java-5.1.38.jar 这个jar包直接上传到 `/export/servers/apache-hive-3.1.0-bin/lib` 这个目录下即可

至此，hive的安装部署已经完成，接下来我们来看下hive的三种交互方式

第五步：配置hive的环境变量

node03服务器执行以下命令配置hive的环境变量

```
sudo vim /etc/profile
```

```
export HIVE_HOME=/export/servers/apache-hive-3.1.0-bin
export PATH=:$HIVE_HOME/bin:$PATH
```

2.7. Hive 的三种交互方式

第一种交互方式 bin/hive

```
cd /export/servers/apache-hive-3.1.0-bin/
bin/hive
```

创建一个数据库

```
create database if not exists mytest;
```

第二种交互方式 HiveServer2

hive官方推荐使用hiveserver2的这种交互方式，需要我们启动hiveserver2这个服务端，然后通过客户端去进行连接
启动服务端（前台启动命令如下）

```
cd /export/servers/apache-hive-3.1.0-bin/
bin/hive --service hiveserver2
```

重新开一个窗口启动我们的客户单进行连接

```
cd /export/servers/apache-hive-3.1.0-bin
bin/beeline
!connect jdbc:hive2://node03.hadoop.com:10000
```

进行连接，用户名为hadoop 密码为123456出现以下错误

```
java.lang.RuntimeException: org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.sec
```

解决方法：关闭hive的服务端，在hadoop的配置文件core-site.xml当中添加以下两行配置，然后重启hdfs以及yarn集群

```
<property>
  <name>hadoop.proxyuser.hadoop.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hadoop.groups</name>
  <value>root</value>
</property>
```

重新进行启动hive的服务端，然后继续使用客户端进行连接即可

启动服务端

```
cd /export/servers/apache-hive-3.1.0-bin/
bin/hive --service hiveserver2
```

开一个新的xshell会话窗口，客户端进行连接

```
cd /export/servers/apache-hive-3.1.0-bin
bin/beeline
!connect jdbc:hive2://node03.hadoop.com:10000
```

第三种交互方式：使用sql语句或者sql脚本进行交互

不进入hive的客户端直接执行hive的hql语句

```
cd /export/servers/apache-hive-3.1.0-bin
bin/hive -e "create database if not exists mytest;"
```

或者我们可以将我们的hql语句写成一个sql脚本然后执行

```
cd /export/servers
vim  hive.sql


create database if not exists mytest;
use mytest;
create table stu(id int,name string);
```

通过hive -f 来执行我们的sql脚本

```
hive -f /export/servers/hive.sql
```

3. Hive 的基本操作

3.1. 创建数据库

```
create database if not exists myhive;
use  myhive;
```

```
0: jdbc:hive2://localhost:10000> create database if not exists myhive;
INFO  : Compiling command(queryId=root_20180610213737_710ef86c-e472-41dc-b1db-11005ad3025d): create database if not exists my
hive
INFO  : Semantic Analysis Completed
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=root_20180610213737_710ef86c-e472-41dc-b1db-11005ad3025d); Time taken: 0.003 seco
nds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=root_20180610213737_710ef86c-e472-41dc-b1db-11005ad3025d): create database if not exists my
hive
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=root_20180610213737_710ef86c-e472-41dc-b1db-11005ad3025d); Time taken: 0.067 seco
nds
INFO  : OK
No rows affected (0.088 seconds)
0: jdbc:hive2://localhost:10000> use myhive;
INFO  : Compiling command(queryId=root_20180610213737_51ca42bd-35d6-4561-8fd5-d4ef7929d8d9): use myhive
INFO  : Semantic Analysis Completed
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=root_20180610213737_51ca42bd-35d6-4561-8fd5-d4ef7929d8d9); Time taken: 0.005 seco
nds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=root_20180610213737_51ca42bd-35d6-4561-8fd5-d4ef7929d8d9): use myhive
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=root_20180610213737_51ca42bd-35d6-4561-8fd5-d4ef7929d8d9); Time taken: 0.026 seco
nds
INFO  : OK
No rows affected (0.043 seconds)
```

说明：hive的表存放位置模式是由hive-site.xml当中的一个属性指定的

```
<name>hive.metastore.warehouse.dir</name>
<value>/user/hive/warehouse</value>
```

3.2. 创建数据库并指定位置

```
create database myhive2 location '/myhive2';
```

3.3. 修改数据库

可以使用alter database 命令来修改数据库的一些属性。但是数据库的元数据信息是不可更改的，包括数据库的名称以及数据库所在的位置

```
alter  database  myhive2  set  dbproperties('createtime'='20180611');
```

3.4. 查看数据库详细信息

查看数据库基本信息

```
desc  database  myhive2;
```

查看数据库更多详细信息

```
desc database extended  myhive2;
```

3.5. 删除数据库

删除一个空数据库，如果数据库下面有数据表，那么就会报错

```
drop  database  myhive2;
```

强制删除数据库，包含数据库下面的表一起删除

```
drop  database  myhive  cascade;   # 不要执行了
```

3.6. 创建数据库表的语法

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
   [(col_name data_type [COMMENT col_comment], ...)]
   [COMMENT table_comment]
   [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
   [CLUSTERED BY (col_name, col_name, ...)
   [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]
   [ROW FORMAT row_format]
   [STORED AS file_format]
   [LOCATION hdfs_path]
```

说明：

1. CREATE TABLE 创建一个指定名字的表。如果相同名字的表已经存在，则抛出异常；用户可以用 IF NOT EXISTS 选项来忽略这个异常。
   . EXTERNAL关键字可以让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION），Hive 创建内部表时，会将数据移动到数据仓库指向的路径；若创建外部表，仅记录数据所在的路径，不对数据的位置做任何改变。在删除表的时候，内部表的元数据和数据会被一起删除，而外部表只删除元数据，不删除数据。
   . LIKE 允许用户复制现有的表结构，但是不复制数据。
   . ROW FORMAT DELIMITED [FIELDS TERMINATED BY char] [COLLECTION ITEMS TERMINATED BY char] [MAP KEYS TERMINATED BY char] [LINES TERMINATED BY char] | SERDE serde_name [WITH SERDEPROPERTIES (property_name=property_value, property_name=property_value, ...)]
   用户在建表的时候可以自定义 SerDe 或者使用自带的 SerDe。如果没有指定 ROW FORMAT 或者 ROW FORMAT DELIMITED，将会使用自带的 SerDe。在建表的时候，用户还需要为表指定列，

用户在指定表的列的同时也会指定自定义的 SerDe，Hive通过 SerDe 确定表的具体的列的数据。
. STORED AS
SEQUENCEFILE|TEXTFILE|RCFILE
如果文件数据是纯文本，可以使用 STORED AS TEXTFILE。如果数据需要压缩，使用 STORED AS SEQUENCEFILE。

2. CLUSTERED BY
对于每一个表（table）或者分区， Hive可以进一步组织成桶，也就是说桶是更为细粒度的数据范围划分。Hive也是 针对某一列进行桶的组织。Hive采用对列值哈希，然后除以桶的个数求余的方式决定该条记录存放在哪个桶当中。
把表（或者分区）组织成桶（Bucket）有两个理由：
    1. 获得更高的查询处理效率。桶为表加上了额外的结构，Hive 在处理有些查询时能利用这个结构。具体而言，连接两个在（包含连接列的）相同列上划分了桶的表，可以使用 Map 端连接（Map-side join）高效的实现。比如JOIN操作。对于JOIN操作两个表有一个相同的列，如果对这两个表都进行了桶操作。那么将保存相同列值的桶进行JOIN操作就可以，可以大大较少JOIN的数据量。
    2. 使取样（sampling）更高效。在处理大规模数据集时，在开发和修改查询的阶段，如果能在数据集的一小部分数据上试运行查询，会带来很多方便。

4. Hive Shell 参数

5. Hive 函数