

大数据培训2

笔记本: Big data

创建时间: 2020/6/20 10:44

更新时间: 2020/6/20 10:58

URL: <https://qingjiaoclass.com/tenant/qyzpbaerkbax/course/datum/1525/12207/40107>

1.1 初识 Hive

任务目的

- 了解 Hive 和数据仓库的概念
- 理解 Hive 的优缺点
- 熟知 Hive 与 Hadoop 的关系

任务清单

- 任务1: Hive 简介
- 任务2: Hive 特点
- 任务3: Hive 与 Hadoop 的关系

任务步骤

任务1: Hive 简介

1.1 Hive 概念

Facebook 为了解决海量结构化日志数据的分析而开发了 Hive，后来开源给了 Apache 软件基金会。

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，Hive 底层是**将 SQL 语句转换为 MapReduce 任务**运行。



图1: Hive Logo

什么是数据仓库?

数据仓库, 英文名称 Data Warehouse, 简称为DW。数据仓库顾名思义, 是一个**很大的数据存储集合**, 出于企业的分析性报告和决策支持目的而创建, 对多样的业务数据进行筛选与整合。它为企业提供一定的BI (商业智能) 能力, 指导业务流程改进、监视时间、成本、质量以及控制。

数据仓库的**输入方是各种各样的数据源**, 最终的输出用于企业的数据分析、数据挖掘、数据报表等方向。

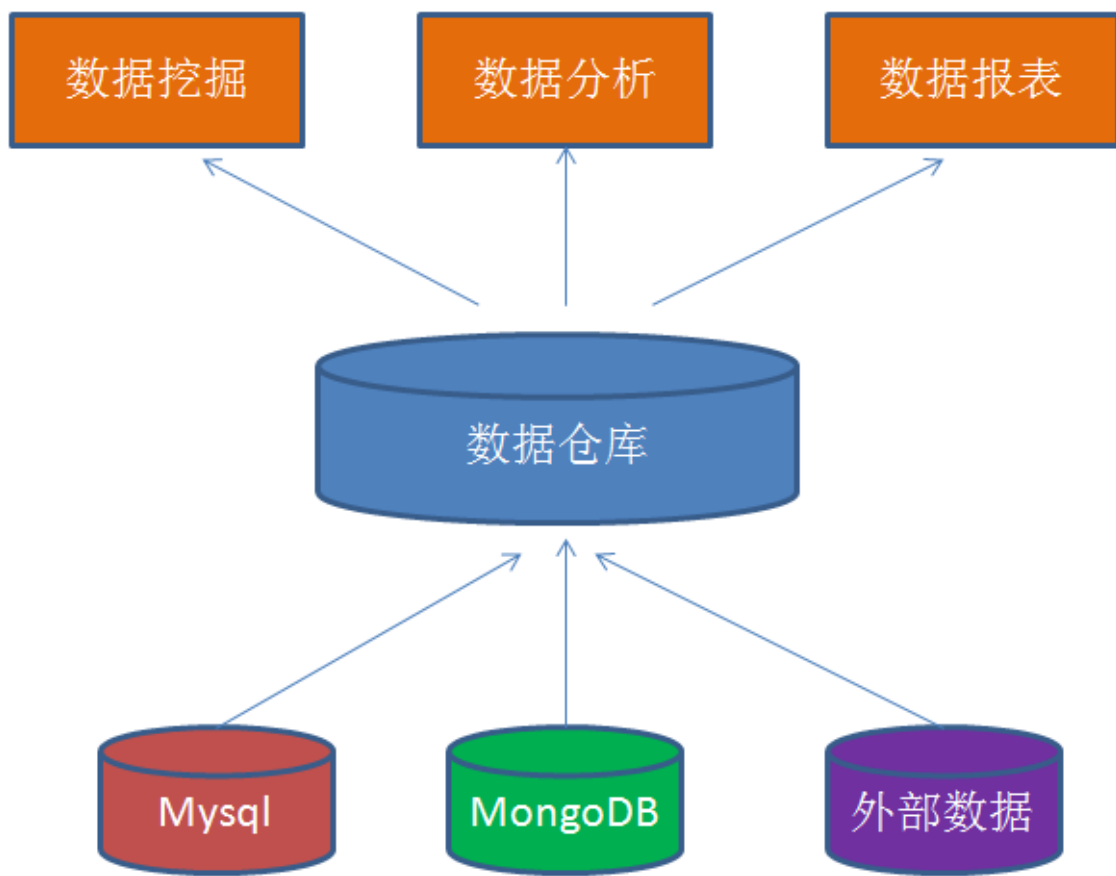


图2

既然数据源有多种多样，数据仓库是如何来集成不同的数据源呢？

不同的数据源的数据集成，所依靠的是 **ETL**。

既然数据源有多种多样，数据仓库
是如何来集成不同的数据源呢？



图3

什么是 ETL？

ETL 的英文全称是 Extract-Transform-Load 的缩写，用来描述将数据从来源迁移到目标的几个过程：

- (1) **Extract**：数据抽取，也就是把数据从数据源读出来。
- (2) **Transform**：数据转换，把原始数据转换成期望的格式和维度。如果用在数据仓库的场景下，Transform也包含数据清洗，清洗掉噪音数据。
- (3) **Load**：数据加载，把处理后的数据加载到目标处，比如数据仓库。

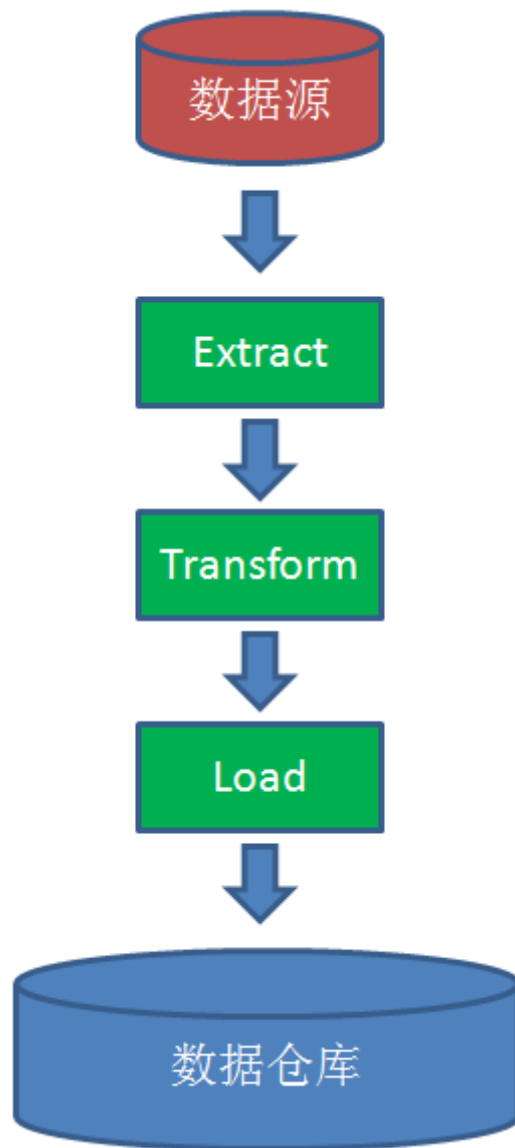


图4

主流的数据仓库有哪些？

道理大概懂了，市面上常用的数据仓库解决方案都有哪些呢？



在国内最常用，是基于一款Hadoop的开源数据仓库，



图5

Hive又是何方神圣呢？为什么 Hive 这么流行呢？

为什么Hive这么流行？它的主要优势是什么？



Hive的主要优势是
世界五百强公司大多采用的是商业
数据仓库

图6

1.2 为什么使用 Hive？

1. 直接使用 MapReduce 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

2. Hive 优势：

- **更友好的接口**：操作接口采用类 SQL 的语法，提供快速开发的能力
- **更低的学习成本**：避免了写 MapReduce，减少开发人员的学习成本
- **更好的扩展性**：可自由扩展集群规模而无需重启服务，还支持用户自定义函数

任务2：Hive 特点

2.1 优点

1. **简单易上手**：操作接口采用类 SQL 语法，提供快速开发的能力
2. **可扩展性，横向扩展**：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
 - **横向扩展**：通过分担压力的方式扩展集群的规模
 - **纵向扩展**：一台服务器cpu i7-6700k 4核心8线程，8核心16线程，内存 64G => 128G
3. **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
4. **良好的容错性**：可以保障即使有节点出现问题，SQL 语句仍可完成执行

2.2 缺点

1. Hive 的 HQL 表达能力有限

- (1) 迭代式算法无法表达，比如 PageRank
- (2) 数据挖掘方面不擅长，由于 MapReduce 数据处理流程的限制，效率更高的算法却无法实现

2. Hive 的效率比较低

- (1) Hive 自动生成的 MapReduce 作业，通常情况下不够智能化
- (2) Hive 调优比较困难，粒度较粗

3. Hive 的查询延时很严重

- (1) 因为 MapReduce Job 的启动过程消耗很长时间，所以不能用在交互查询系统中
- (2) 对于处理小数据没有优势，优势在于处理大数据

任务3：Hive 与 Hadoop 的关系

本质是：将 HQL 转化成 MapReduce 程序。

图7

- (1) Hive 处理的数据存储在 HDFS;
- (2) Hive 分析数据底层的实现是 MapReduce, 利用 MapReduce 查询数据;
- (3) 执行程序运行在 YARN 上。

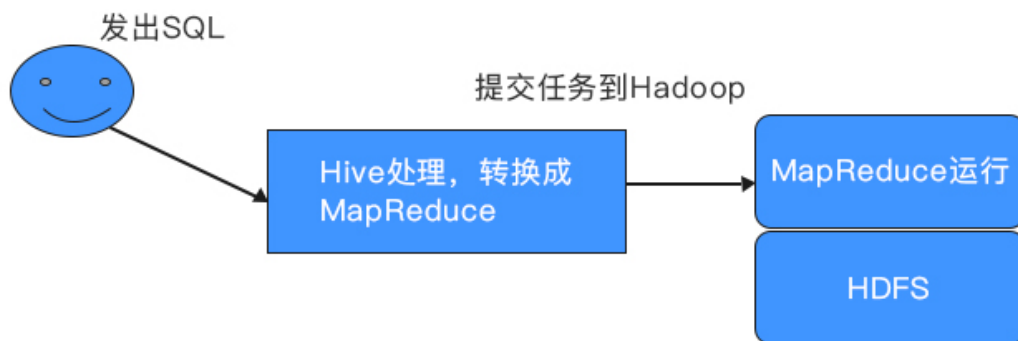


图8

1.2 Hive 架构原理

任务目的

- 熟记 Hive 内部架构的四个组成部分
- 了解 Hive 和 RDBMS 的区别

任务清单

- 任务1: Hive 架构原理

- 任务2: Hive 和 RDBMS 的对比

任务步骤

任务1: Hive 架构原理

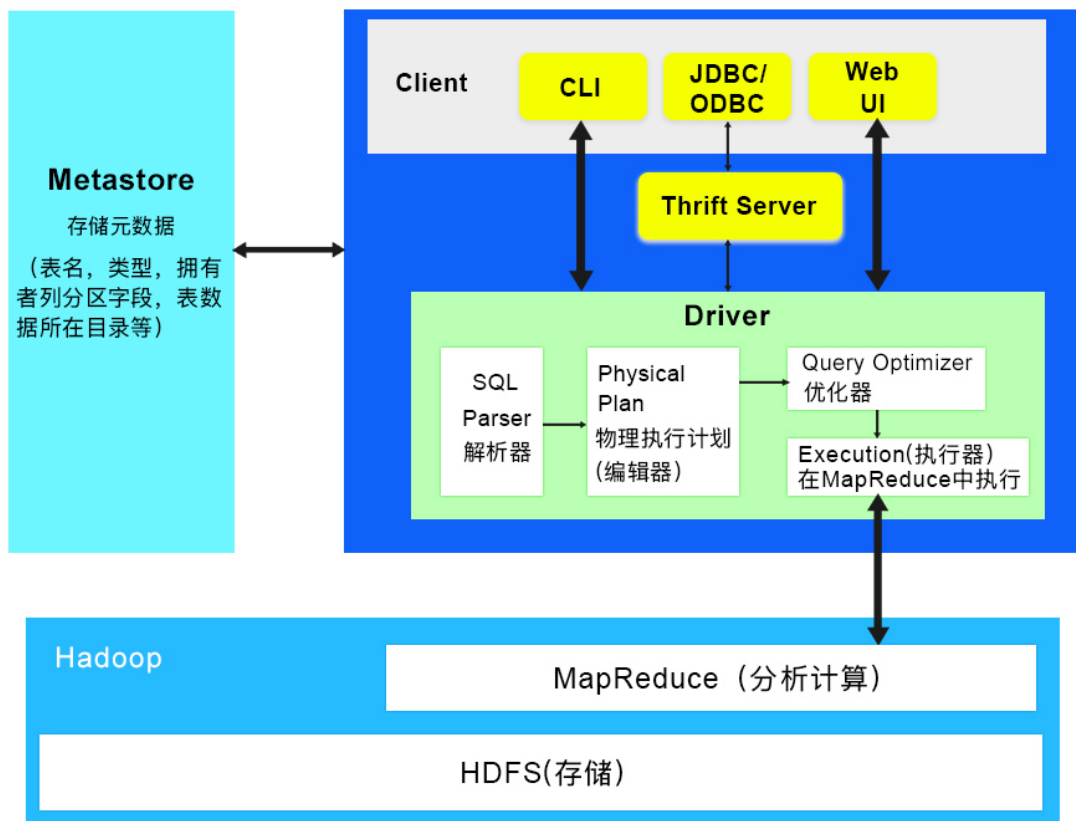


图1

从上图看出 Hive 的内部架构由四部分组成：

1. 用户接口: Client

- **CLI**: Shell 终端命令行 (Command Line Interface) , 采用交互形式使用 Hive 命令行与 Hive 进行交互, **最常用 (学习, 调试, 生产)**
- **JDBC/ODBC**: 是 Hive 的基于 JDBC 操作提供的客户端, 用户 (开发员, 运维人员) 通过这连接至 Hive server 服务
 - **跨语言服务 Thrift Server**: Thrift 是 Facebook 开发的一个软件框架, 可以用来进行可扩展且跨语言的服务的开发, Hive 集成了该服务, 能让不同的编程语言调用 Hive 的接口。
- **Web UI**: 通过浏览器访问 Hive

2. 元数据存储: Metastore

- 元数据，通俗的讲，就是**存储在 Hive 中的数据的描述信息**。Hive 中的元数据通常包括：表名、表所属的数据库（默认是default）、表的列和分区及其属性，表的类型（是否为外部表等），表的数据所在目录等。
- MetaStore 默认存在**自带的 Derby 数据库**中。缺点就是不适合多用户操作，并且数据存储目录不固定。数据库跟着 Hive 走，极度不方便管理。
- **解决方案**：通常将其存在我们**自己创建的 MySQL 库中**（本地 或 远程）
- **Hive 和 MySQL 之间通过 MetaStore 服务交互**

3. 驱动器 Driver

驱动器 Driver 的主要组件：**解析器 (SQL Parser)**、**编译器 (Physical Plan)**，**优化器 (Query Optimizer)** 和 **执行器 (Execution)**。

(1) **解析器 (SQL Parser)**：将 SQL 字符串转换成抽象语法树 AST，这一步一般都用第三方工具库完成，比如 antlr；对 AST 进行语法分析，比如表是否存在、字段是否存在、SQL 语义是否有误。

(2) **编译器 (Physical Plan)**：将 AST 编译生成逻辑执行计划。

(3) **优化器 (Query Optimizer)**：对逻辑执行计划进行优化。

(4) **执行器 (Execution)**：把逻辑执行计划转换成可以运行的物理计划。对于 Hive 来说，就是 MapReduce/Spark。

执行流程：HiveQL 通过命令行或者客户端提交，经过解析器和编译器，运用 MetaStore 中的元数据进行类型检测和语法分析，生成一个逻辑方案，然后通过优化处理，产生一个 MapReduce 任务。

4. Hadoop

使用 HDFS 进行存储，使用 MapReduce 进行计算。

任务2：Hive 和 RDBMS 的对比

由于 Hive 采用了**类似 SQL 的查询语言 HQL(Hive Query Language)**，因此很容易将 Hive 理解为数据库。其实从结构上来看，Hive 和数据库除了拥有类似的查询语言，再无类似之处。本文将从多个方面来阐述 Hive 和数据库的差异。数据库可以用在 Online 的应用中，但是**Hive是为数据仓库而设计的**，清楚这一点，有助于从应用角度理解Hive的特性。

1. 查询语言

由于 SQL 被广泛的应用在数据仓库中，因此，专门针对 Hive 的特性设计了**类 SQL 的查询语言 HQL**。熟悉 SQL 开发的开发者可以很方便的使用 Hive

进行开发。

2. 数据存储位置

Hive 是建立在 Hadoop 之上的，所有 **Hive 的数据都是存储在 HDFS 中的**。而**数据库**则可以将数据保存在**块设备或者本地文件系统中**。

3. 数据格式

Hive 中**没有定义专门的数据格式**，**数据格式可以由用户指定**，用户定义数据格式需要指定三个属性：**列分隔符（通常为空格、“\t”、“\x001”）、行分隔符（“\n”）以及读取文件数据的方法**（Hive中默认有三个文件格式 TextFile、SequenceFile 以及 RCFile）。

由于在加载数据的过程中，不需要从用户数据格式到 Hive 定义的数据格式的转换，因此，Hive在加载的过程中不会对数据本身进行任何修改，而只是**将数据内容复制或者移动到相应的 HDFS 目录中**。

而在数据库中，不同的数据库有不同的存储引擎，**定义了自己的数据格式**。所有**数据都会按照一定的组织存储**，因此，数据库加载数据的过程会比较耗时。

4. 数据更新

由于 Hive 是针对数据仓库应用设计的，而数据仓库的内容是**读多写少**的。因此，**Hive中不建议对数据的改写**，所有的数据都是在加载的时候确定好的。

而数据库中的数据通常是**需要经常进行修改的**，因此可以使用 `INSERT INTO ... VALUES` 添加数据，使用 `UPDATE ... SET` 修改数据。

5. 索引

Hive 不建立索引：Hive 在加载数据的过程中不会对数据进行任何处理，甚至不会对数据进行扫描，因此也没有对数据中的某些 Key 建立索引。Hive 要访问数据中满足条件的特定值时，需要**暴力扫描整个数据**，因此**访问延迟较高**。由于 MapReduce 的引入，Hive 可以并行访问数据，因此即使没有索引，对于大数据量的访问，Hive 仍然可以体现出优势。

数据库中，通常会**针对一个或者几个列建立索引**，因此对于少量的特定条件的数据的访问，数据库可以有很高的效率，较低的延迟。由于数据的访问延迟较高，决定了 Hive 不适合在线数据查询。

6. 执行器

Hive 中大多数查询的执行是通过 Hadoop 提供的 **MapReduce** 来实现的。而数据库通常有自己的执行引擎。

7. 执行延迟

Hive 在查询数据的时候，由于没有索引，**需要扫描整个表，因此延迟较高**。另外一个导致 Hive 执行延迟高的因素是 **MapReduce 框架**。由于 MapReduce 本身具有较高的延迟，因此在利用 MapReduce 执行 Hive 查询时，也会有较高的延迟。

相对的，**数据库的执行延迟较低**。当然，这个也是有条件的，即数据规模较小，当数据规模大到超过数据库的处理能力的时候，Hive 的并行计算显然能体现出优势。

8. 可扩展性

由于 Hive 是建立在 Hadoop 之上的，因此 Hive 的可扩展性是和 Hadoop 的可扩展性是一致的（世界上最大的 Hadoop 集群在 Yahoo!，2009 年的规模在 4000 台节点左右）。

而数据库由于 ACID 语义的严格限制，**扩展行非常有限**。目前最先进的并行数据库 Oracle 在理论上的扩展能力也只有 100 台左右。

9. 处理数据规模

由于 Hive 建立在集群上并可以利用 MapReduce 进行并行计算，因此可以**支持很大规模的数据**；对应的，**数据库可以支持的数据规模较小**。

- Hive 和 RDBMS 的对比表：

对比项	Hive	RDBMS
查询语言	HQL	SQL
数据存储位置	HDFS	Raw Device 或者 Local FS
数据格式	用户自定义	系统决定
数据更新	从 Hive 0.14开始支持（只能在支持事务的表上执行）	支持
索引	无	有
执行器	MapReduce	Executor
执行延迟	高	低

可扩展性	高	有限
处理数据规模	大	小

总结：Hive 具有 SQL 数据库的外表，但应用场景完全不同，**Hive 适合用来做批量海量数据统计分析，也就是数据仓库。**

2.1 Hive 内嵌模式安装

任务目的

- 了解 Hive 的三种运行模式
- 知道 Hive 的官方地址
- 掌握 Hive 内嵌模式的安装

任务清单

- 任务1：Hive 三种运行模式
- 任务2：Hive 安装地址
- 任务3：Hive 内嵌模式安装

任务步骤

任务1：Hive 三种运行模式

1. 内嵌模式

将元数据保存在本地内嵌的 **Derby 数据库**中，这是使用 Hive 最简单的方式。但是这种方式缺点也比较明显，因为一个内嵌的 Derby 数据库每次只能访问一个数据文件，这也就意味着它**不支持多会话连接**。

2. 本地模式

这种模式是将元数据保存在**本地独立的数据库**中（一般是 **MySQL**），这种方式是一个多用户的模式，运行多个用户 Client 连接到一个数据库中。这种方式一般**作为公司内部同时使用 Hive**。这里有一个前提，每一个用户必须要有对 MySQL 的访问权利，即每一个客户端**使用者需要知道 MySQL 的用户名和密码**才行。

3. 远程模式

此模式应用于 **Hive 客户端较多**的情况。把 MySQL 数据库独立出来，**将元数据保存在远程独立的 MySQL 服务中**，避免了在每个客户端都安装 MySQL 服务从而造成冗余浪费的情况。此时**客户端无需知道 MySQL 的用户名和密码**。

任务2：Hive 安装地址

1. Hive 官网地址

<http://hive.apache.org/>

2. 文档查看地址

<https://cwiki.apache.org/confluence/display/Hive/GettingStarted>

3. 下载地址

<http://archive.apache.org/dist/hive/>

4. github 地址

<https://github.com/apache/hive>

任务3：Hive 内嵌模式安装

3.1 Hive 安装

Hive是大数据生态圈中最常用的数据仓库，也是有 Hadoop 集群的公司的必备。所以 Hive 安装和使用也是大数据开发或运维人员都必须掌握的。Hive的安装很简单，而且 Hive1.0 和 Hive2.0 版本的安装非常类似，完全可以套用。

本课程选择 hive2.3.4 作为教学版本！

1. 解压安装包

现在已经为大家下载好了 hive2.3.4 的安装包，存放在 `/root/software` 目录下，首先进入此目录下，使用如下命令进行解压即可使用：

```
tar -zxvf apache-hive-2.3.4-bin.tar.gz
```

将其**解压到当前目录下**，即 `/root/software` 中，效果图如下所示：

```
→ software ll
total 374320
drwxr-xr-x 10 root root      184 Apr 28 16:46 apache-hive-2.3.4-bin
-rw-r--r--  1 root root 232234292 Dec  7 2018 apache-hive-2.3.4-bin.tar.gz
drwxrwxr-x  8 root root      191 Dec 12 23:25 eclipse
drwxr-xr-x  1 1000 ftp       18 Apr 28 16:41 hadoop-2.7.7
drwxr-xr-x 16 root root      4096 Jul 19 2018 hadoop-2.7.7-src
-rw-r--r--  1 root root 113383841 Feb 28 14:18 hbase-1.4.10-bin.tar.gz
drwxr-xr-x  7  10 143        245 Jul  4 2019 jdk1.8.0_221
-rw-r--r--  1 root root 37676320 Feb 28 14:19 zookeeper-3.4.14.tar.gz
→ software
```

图1

2. 配置环境变量

(1) 首先打开 /etc/profile 文件：

```
vim /etc/profile
```

(2) 将以下内容添加到配置文件的底部，添加完成输入“:wq”保存退出：

```
# 配置Hive的安装目录
export HIVE_HOME=/root/software/apache-hive-2.3.4-bin
# 在原PATH的基础上加入Hive的bin目录
export PATH=$PATH:$HIVE_HOME/bin
```

注意：

- export 是把这两个变量导出为全局变量。
- 大小写必须严格区分。

(3) 让配置文件立即生效，使用如下命令：

```
source /etc/profile
```

(4) 检测 Hive 环境变量是否设置成功，使用如下命令查看 Hive 版本：

```
hive --version
```

执行此命令后，若是出现 Hive 版本信息说明配置成功：

```
→ apache-hive-2.3.4-bin hive --version
Hive 2.3.4
Git git://daijymacpro-2.local/Users/daijy/commit/hive -r 56acdd2120b9ce6790185c679223b8b5e884aaf2
Compiled by daijy on Wed Oct 31 14:20:50 PDT 2018
From source with checksum 9f2d17b212f3a05297ac7dd40b65bab0
```

图2

3.2 内嵌模式安装

1. 修改配置文件 hive-env.sh

切换到 `${HIVE_HOME}/conf` 目录下，将 `hive-env.sh.template` 文件复制一份并重命名为 `hive-env.sh`：

```
cp hive-env.sh.template hive-env.sh
```

修改完成，使用 **vi 编辑器** 进行编辑：

```
vim hive-env.sh
```

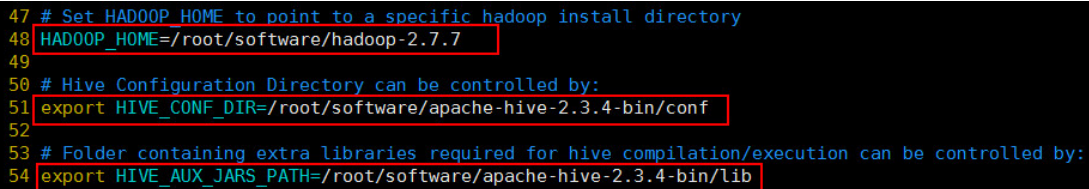
在文件中配置 **HADOOP_HOME**、**HIVE_CONF_DIR** 以及 **HIVE_AUX_JARS_PATH** 参数：

```
# 配置Hadoop安装路径
HADOOP_HOME=/root/software/hadoop-2.7.7

# 配置Hive配置文件存放路径
export HIVE_CONF_DIR=/root/software/apache-hive-2.3.4-bin/conf

# 配置Hive运行资源库路径
export HIVE_AUX_JARS_PATH=/root/software/apache-hive-2.3.4-bin/lib
```

如下图所示：



```
47 # Set HADOOP_HOME to point to a specific hadoop install directory
48 HADOOP_HOME=/root/software/hadoop-2.7.7
49
50 # Hive Configuration Directory can be controlled by:
51 export HIVE_CONF_DIR=/root/software/apache-hive-2.3.4-bin/conf
52
53 # Folder containing extra libraries required for hive compilation/execution can be controlled by:
54 export HIVE_AUX_JARS_PATH=/root/software/apache-hive-2.3.4-bin/lib
```

图3

配置完成，输入“**:wq**”保存退出。

2. 初始化元数据库

注意：当使用的 Hive 是 2.x 之前的版本时，不做初始化也是 OK 的。Hive 第一次启动时会自动进行初始化，只不过不会生成足够多的元数据库的表，其它的在使用过程中会慢慢生成。****如果使用的是 2.x 版本的 Hive，那么就必须手动初始化元数据库。****使用如下命令进行初始化，这里我们使用 **Hive 默认的 db 类型 “derby”**：

```
schematool -dbType derby -initSchema
```

若是出现“`schemaTool completed`”则初始化成功，如下图所示：

```

→ conf schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/root/software/apache-hive-2.3.4-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/root/software/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/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.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL: jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver : org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User: APP
Starting metastore schema initialization to 2.3.0
Initialization script hive-schema-2.3.0.derby.sql
Initialization script completed
schemaTool completed
→ conf

```

图4

初始化完成，会在当前目录下生成一个 **derby.log** 文件和一个 **metastore_db** 目录：

```

→ conf ll
total 296
-rw-r--r-- 1 root root 1596 Nov 1 2018 beeline-log4j2.properties.template
-rw-r--r-- 1 root root 711 Apr 28 18:18 derby.log
-rw-r--r-- 1 root root 257573 Nov 1 2018 hive-default.xml.template
-rw-r--r-- 1 root root 2448 Apr 28 18:01 hive-env.sh
-rw-r--r-- 1 root root 2365 Nov 1 2018 hive-env.sh.template
-rw-r--r-- 1 root root 2274 Nov 1 2018 hive-exec-log4j2.properties.template
-rw-r--r-- 1 root root 2925 Nov 1 2018 hive-log4j2.properties.template
-rw-r--r-- 1 root root 2060 Nov 1 2018 ivysettings.xml
-rw-r--r-- 1 root root 2719 Nov 1 2018 llap-cli-log4j2.properties.template
-rw-r--r-- 1 root root 7041 Nov 1 2018 llap-daemon-log4j2.properties.template
drwxr-xr-x 5 root root 133 Apr 28 18:18 metastore_db
-rw-r--r-- 1 root root 2662 Nov 1 2018 parquet-logging.properties
→ conf

```

图5

3. Hive 连接

在此目录下使用 Hive 的三种连接方式之一：**CLI 启动 Hive**。由于已经在环境变量中配置了 **HIVE_HOME**，所以这里直接在命令行执行如下命令即可：

```

hive
或者
hive --service cli

```

效果如下图所示：

```

→ conf hive
which: no hbase in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root/software/jdk1.8.0_221/bin:/root/software/hadoop-2.7.7/bin:/root/software/hadoop-2.7.7/sbin:/root/software/apache-hive-2.3.4-bin/bin)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/root/software/apache-hive-2.3.4-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/root/software/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/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.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/root/software/apache-hive-2.3.4-bin/lib/hive-common-2.3.4.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive>

```

图6

这种存储方式的弊端是在同一个目录下同时只能有一个 Hive 客户端能使用数据库，否则会提示 Hive 元数据有问题（这是一个很常见的错误）：

```
→ conf hive
which: no hbase in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root/software/jdk1.8.0_221/bin:/root/software/hadoop-2.7.7/bin:/root/software/hadoop-2.7.7/sbin:/root/software/apache-hive-2.3.4-bin/bin)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/root/software/apache-hive-2.3.4-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/root/software/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/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.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/root/software/apache-hive-2.3.4-bin/lib/hive-common-2.3.4.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive> show databases;
FAILED: SemanticException org.apache.hadoop.hive.ql.metadata.HiveException: java.lang.RuntimeException: Unable to instantiate org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient
hive>
```

图7

另外，要是想要在其它目录下操作 Hive，必须要在其它目录下重新初始化元数据库。

2.2 Hive 本地模式安装

任务目的

- 掌握使用 RPM 离线安装 MySQL 组件的方法
- 掌握初始化和登录 MySQL 的命令
- 掌握 Hive 的安装部署
- 掌握 Hive 元数据配置到 MySQL 的步骤

任务清单

- 任务1：安装 MySQL
- 任务2：Hive 安装部署
- 任务3：Hive 元数据配置到 MySQL

任务步骤

任务1：安装 MySQL

1.1 解压安装包

现在已经为大家下载好了 MySQL 5.7.25 的安装包，存放在 `/root/software` 目录下，首先进入此目录下，使用如下命令进行解压即可使用：

```
tar -xvf mysql-5.7.25-1.el7.x86_64.rpm-bundle.tar
```

将其解压到当前目录下，即 /root/software 中，效果图如下所示：

```
→ software tar -xvf mysql-5.7.25-1.el7.x86_64.rpm-bundle.tar
mysql-community-libs-5.7.25-1.el7.x86_64.rpm
mysql-community-libs-compat-5.7.25-1.el7.x86_64.rpm
mysql-community-embedded-devel-5.7.25-1.el7.x86_64.rpm
mysql-community-embedded-5.7.25-1.el7.x86_64.rpm
mysql-community-client-5.7.25-1.el7.x86_64.rpm
mysql-community-server-5.7.25-1.el7.x86_64.rpm
mysql-community-embedded-compat-5.7.25-1.el7.x86_64.rpm
mysql-community-test-5.7.25-1.el7.x86_64.rpm
mysql-community-devel-5.7.25-1.el7.x86_64.rpm
mysql-community-common-5.7.25-1.el7.x86_64.rpm
→ software █
```

图1

1.2 安装 MySQL 组件

在大多数情况下，我们需要安装 mysql-community-server、mysql-community-client、mysql-community-libs、mysql-community-common 和 mysql-community-libs-compat 包，才能实现功能标准的 MySQL 安装。要是不想安装功能标准的 MySQL，也可以只安装 mysql-community-server 和 mysql-community-client 即可。

使用 `rpm -ivh` 命令依次安装以下组件：

(1) 首先安装 `mysql-community-common`（服务器和客户端库的公共文件），使用命令：

```
rpm -ivh mysql-community-common-5.7.25-1.el7.x86_64.rpm
```

命令解析：

- rpm：RPM软件包的管理工具。
- -ivh：--install--verbose--hash，安装rpm包且显示安装进度。

若是出现“100%”，则表示安装成功：

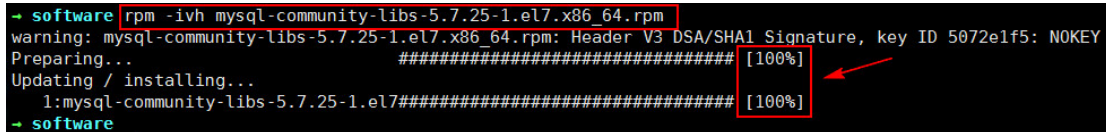
```
→ software rpm -ivh mysql-community-common-5.7.25-1.el7.x86_64.rpm
warning: mysql-community-common-5.7.25-1.el7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Preparing... ##### [100%]
Updating / installing...
 1:mysql-community-common-5.7.25-1.e##### [100%]
→ software █
```

图2

(2) 其次安装 `mysql-community-libs` (MySQL数据库客户端应用程序的共享库) , 使用命令:

```
rpm -ivh mysql-community-libs-5.7.25-1.el7.x86_64.rpm
```

若是出现“100%”, 则表示安装成功:



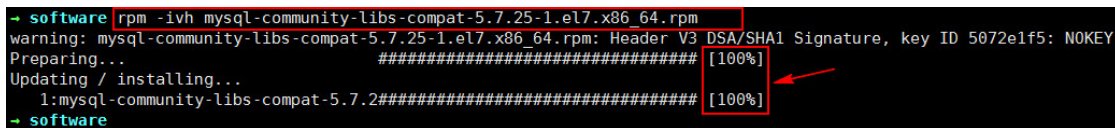
```
→ software rpm -ivh mysql-community-libs-5.7.25-1.el7.x86_64.rpm
warning: mysql-community-libs-5.7.25-1.el7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Preparing...
Updating / installing...
 1:mysql-community-libs-5.7.25-1.el7##### [100%]
→ software
```

图3

(3) 之后安装 `mysql-community-libs-compat` (MySQL 之前版本的共享兼容库) , 使用命令:

```
rpm -ivh mysql-community-libs-compat-5.7.25-1.el7.x86_64.rpm
```

若是出现“100%”, 则表示安装成功:



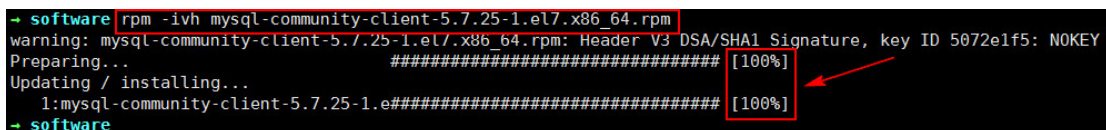
```
→ software rpm -ivh mysql-community-libs-compat-5.7.25-1.el7.x86_64.rpm
warning: mysql-community-libs-compat-5.7.25-1.el7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Preparing...
Updating / installing...
 1:mysql-community-libs-compat-5.7.2##### [100%]
→ software
```

图4

(4) 之后安装 `mysql-community-client` (MySQL客户端应用程序和工具) , 使用命令:

```
rpm -ivh mysql-community-client-5.7.25-1.el7.x86_64.rpm
```

若是出现“100%”, 则表示安装成功:



```
→ software rpm -ivh mysql-community-client-5.7.25-1.el7.x86_64.rpm
warning: mysql-community-client-5.7.25-1.el7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Preparing...
Updating / installing...
 1:mysql-community-client-5.7.25-1.e##### [100%]
→ software
```

图5

(5) 最后安装 `mysql-community-server` (数据库服务器和相关工具) , 使用命令:

```
rpm -ivh mysql-community-server-5.7.25-1.el7.x86_64.rpm
```

若是出现“100%”, 则表示安装成功:


```
→ software rpm -ivh mysql-community-server-5.7.25-1.el7.x86_64.rpm
warning: mysql-community-server-5.7.25-1.el7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Preparing...##### [100%]
Updating / installing...
1:mysql-community-server-5.7.25-1.e##### [100%]
```

图6

1.3 登录 MySQL

1. 初始化 MySQL 的数据库

安装好 MySQL 后，我们需要初始化数据库，初始化和启动数据库时最好不要使用root用户，而是使用mysql用户启动。

```
/usr/sbin/mysqld --initialize-insecure --user=mysql
```

对于MySQL 5.7.6以后的5.7系列版本，MySQL也可以使用 `mysqld --initialize` 初始化数据库，该命令会在 `/var/log/mysqld.log` 文件中生成一个登录 MySQL 的随机密码，而 `mysqld --initialize-insecure` 命令不会生成随机密码，而是设置 MySQL 的密码为空。

2. 启动 MySQL 服务

使用如下命令开启 MySQL 服务，让其在后台运行：

```
/usr/sbin/mysqld --user=mysql &
```

说明：一定要加“&”，才能脚本放到后台运行。

执行后，系统会打印出命令执行的PID，如下图所示：

```
→ ~ /usr/sbin/mysqld --user=mysql &
[1] 1360 ← 进程号
→ ~
```


图7

3. 登录 MySQL

使用root用户无密码登录 MySQL：

```
mysql -uroot
```

若是出现下图效果，说明登录成功：

```
→ ~ mysql -uroot  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 2  
Server version: 5.7.25 MySQL Community Server (GPL)  
  
Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> 
```

登录成功

图8

4. 重置 MySQL 密码

在 5.7 版本后, 我们可以使用 `alter user...identified by` 命令把root用户的密码修改为“123456”, 具体命令如下所示:

```
mysql> alter user 'root'@'localhost' identified by '123456';
```

效果如下图所示:


```
mysql> alter user 'root'@'localhost' identified by '123456';  
Query OK, 0 rows affected (0.00 sec)
```

图9

修改完成, 使用 `exit` 或者 `quit` 命令退出 MySQL, 重新登录验证密码是否修改成功, 具体命令如下所示:

```
mysql -uroot -p123456
```

效果如下图所示:

```
mysql> quit 退出  
Bye  
→ ~ mysql -uroot -p123456 使用新密码重新登录  
mysql: [Warning] Using a password on the command line interface can be insecure.  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 3  
Server version: 5.7.25 MySQL Community Server (GPL)  
  
Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> 
```

登录成功

图10

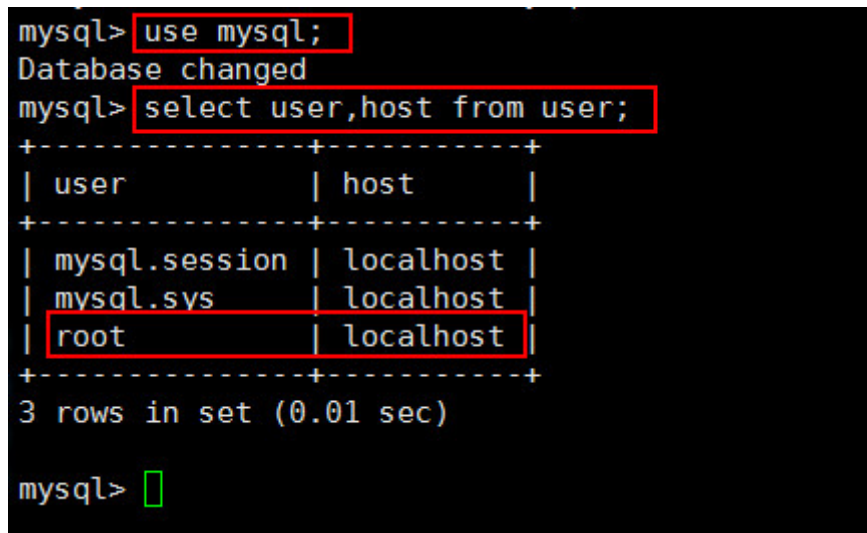
1.4 增加远程登录权限

当我们的帐号不允许从远程登录，只能在 `localhost` 连接时。这个时候只要在 MySQL 服务器上，更改mysql数据库里的 `user` 表里的 `host` 项，从 `localhost` 改成 `%` 即可实现用户远程登录。

(1) 首先我们来查看 mysql 数据库下的 user表信息：

```
mysql> use mysql; # 切换到mysql数据库
mysql> select user,host from user; # 查询用户信息
```

效果如下图所示：



```
mysql> use mysql;
Database changed
mysql> select user,host from user;
+-----+-----+
| user           | host       |
+-----+-----+
| mysql.session  | localhost  |
| mysql.sys      | localhost  |
| root           | localhost  |
+-----+-----+
3 rows in set (0.01 sec)

mysql>
```

图11

可以看到在user表中已创建的root用户。host字段表示登录的主机，其值可以用IP地址，也可用主机名。

(2) 实现远程连接（授权法）

将 `host` 字段的值改为`%`就表示在任何客户端机器上能以root用户登录到MySQL 服务器，建议在开发时设为`%`。

```
# 设置远程登录权限
mysql> update user set host='%' where host='localhost';
# 刷新配置信息
mysql> flush privileges;
```

效果图如下所示：


```
mysql> update user set host='%' where host='localhost';
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

修改远程登录权限

```
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

刷新配置信息

```
mysql> select user,host from user;
```

查询用户信息

user	host
mysql.session	%
mysql.sys	%
root	%

3 rows in set (0.00 sec)

```
mysql>
```

图12

任务2: Hive 安装部署

Hive是大数据生态圈中最常用的数据仓库，也是有 Hadoop 集群的公司的必备。所以 Hive 安装和使用也是大数据开发或运维人员都必须掌握的。Hive的安装很简单，而且 Hive1.0 和 Hive2.0 版本的安装非常类似，完全可以套用。本门课程选择 hive2.3.4 作为教学版本！

2.1 解压安装包

现在已经为大家下载好了 hive2.3.4 的安装包，存放在 `/root/software` 目录下，首先进入此目录下，使用如下命令进行解压即可使用：

```
tar -zxvf apache-hive-2.3.4-bin.tar.gz
```

将其解压到当前目录下，即 `/root/software` 中，效果图如下所示：

```
→ software ll
total 374320
drwxr-xr-x 10 root root      184 Apr 28 16:46 apache-hive-2.3.4-bin
-rw-r--r--  1 root root 232234292 Dec  7 2018 apache-hive-2.3.4-bin.tar.gz
drwxrwxr-x  8 root root      191 Dec 12 23:25 eclipse
drwxr-xr-x  1 1000 ftp       18 Apr 28 16:41 hadoop-2.7.7
drwxr-xr-x 16 root root     4096 Jul 19 2018 hadoop-2.7.7-src
-rw-r--r--  1 root root 113383841 Feb 28 14:18 hbase-1.4.10-bin.tar.gz
drwxr-xr-x  7  10  143       245 Jul  4 2019 jdk1.8.0_221
-rw-r--r--  1 root root  37676320 Feb 28 14:19 zookeeper-3.4.14.tar.gz
→ software
```

图13

2.2 配置环境变量

(1) 首先打开 /etc/profile 文件：

```
vim /etc/profile
```

(2) 将以下内容添加到配置文件的底部，添加完成输入“:wq”保存退出：

```
# 配置Hive的安装目录
export HIVE_HOME=/root/software/apache-hive-2.3.4-bin
# 在原PATH的基础上加入Hive的bin目录
export PATH=$PATH:$HIVE_HOME/bin
```

注意：

- export 是把这两个变量导出为全局变量。
- 大小写必须严格区分。

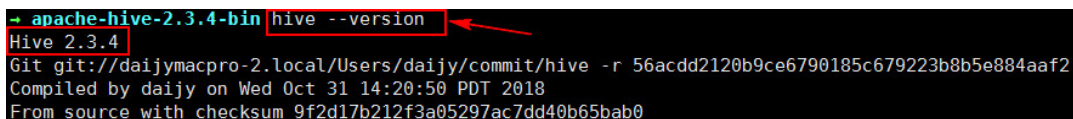
(3) 让配置文件立即生效，使用如下命令：

```
source /etc/profile
```

(4) 检测 Hive 环境变量是否设置成功，使用如下命令查看 Hive 版本：

```
hive --version
```

执行此命令后，若是出现 Hive 版本信息说明配置成功：



```
→ apache-hive-2.3.4-bin hive --version
Hive 2.3.4
Git git://daijymacpro-2.local/Users/daijy/commit/hive -r 56acdd2120b9ce6790185c679223b8b5e884aaf2
Compiled by daijy on Wed Oct 31 14:20:50 PDT 2018
From source with checksum 9f2d17b212f3a05297ac7dd40b65bab0
```

图14

2.3 修改配置文件 hive-env.sh

切换到 \${HIVE_HOME}/conf 目录下，将 hive-env.sh.template 文件复制一份并重命名为 hive-env.sh：

```
cp hive-env.sh.template hive-env.sh
```

修改完成，使用 vi 编辑器进行编辑：

```
vim hive-env.sh
```

在文件中配置 HADOOP_HOME、HIVE_CONF_DIR 以及 HIVE_AUX_JARS_PATH 参数：

```
# 配置Hadoop安装路径
HADOOP_HOME=/root/software/hadoop-2.7.7

# 配置Hive配置文件存放路径
export HIVE_CONF_DIR=/root/software/apache-hive-2.3.4-bin/conf

# 配置Hive运行资源库路径
export HIVE_AUX_JARS_PATH=/root/software/apache-hive-2.3.4-bin/lib
```

如下图所示：

```
47 # Set HADOOP_HOME to point to a specific hadoop install directory
48 HADOOP_HOME=/root/software/hadoop-2.7.7
49
50 # Hive Configuration Directory can be controlled by:
51 export HIVE_CONF_DIR=/root/software/apache-hive-2.3.4-bin/conf
52
53 # Folder containing extra libraries required for hive compilation/execution can be controlled by:
54 export HIVE_AUX_JARS_PATH=/root/software/apache-hive-2.3.4-bin/lib
```

图15

配置完成，输入“:wq”保存退出。

任务3：Hive元数据配置到MySQL

3.1 驱动拷贝

将/root/software目录下的 MySQL 驱动包 mysql-connector-java-5.1.47-bin.jar 拷贝到 \${HIVE_HOME}/lib 目录下。

```
cd /root/software/
cp mysql-connector-java-5.1.47-bin.jar apache-hive-2.3.4-bin/lib/
```

进入 \${HIVE_HOME}/lib 目录下进行验证：

```
-rw-r--r-- 1 root root 111908 May 3 2018 metrics-core-3.1.0.jar
-rw-r--r-- 1 root root 15823 May 3 2018 metrics-json-3.1.0.jar
-rw-r--r-- 1 root root 35907 May 3 2018 metrics-jvm-3.1.0.jar
-rw-r--r-- 1 root root 1007505 Apr 30 17:48 mysql-connector-java-5.1.47-bin.jar
-rw-r--r-- 1 root root 7954 Oct 25 2018 mysql-metadata-storage-0.9.2.jar
```

拷贝成功

图16

3.2 配置 Metastore到MySQL

(1) 在\${HIVE_HOME}/conf目录下创建一个名为hive-site.xml的文件，并使用vi编辑器进行编辑：

```
touch hive-site.xml
vim hive-site.xml
```

(2) 根据官方文档配置参数

(<https://cwiki.apache.org/confluence/display/Hive/AdminManual+Metastore+Administration>)，拷贝数据到 hive-site.xml 文件中：

```
<configuration>
<!--连接元数据库的链接信息 -->
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://localhost:3306/hivedb?createDatabaseIfNotExist=true&useSSL=false
<description>JDBC connect string for a JDBC metastore</description>
</property>
<property>
<!--连接数据库驱动 -->
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<!--连接数据库用户名称 -->
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>root</value>
<description>username to use against metastore database</description>
</property>
<!--连接数据库用户密码 -->
<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>123456</value>
<description>password to use against metastore database</description>
</property>
</configuration>
```

在 `javax.jdo.option.ConnectionURL` 参数的 `value` 值中添加 `&useSSL=false`，主要是为了避免之后初始化产生警告信息。添加 `&useUnicode=true&characterEncoding=UTF-8` 用来指定字符的编码、解码格式。另外，在 XML 中，包括 HTML 中，要显示 `&` 这个字符，必须对其转义转义，使用 `&` 进行转义。

3.3 初始化元数据库

注意：当使用的 Hive 是 2.x 之前的版本时，不做初始化也是 OK 的。Hive 第一次启动时会自动进行初始化，只不过不会生成足够多的元数据库的表，其它的在使用过程中会慢慢生成。**如果使用的是 2.x 版本的 Hive，那么就必须手动初始化元数据库。**使用 `schematool -dbType <db type> -initSchema` 命令进行初始化：

```
schematool -dbType mysql -initSchema
```

若是出现“schemaTool completed”则初始化成功，如下图所示：

```
→ conf schematool -dbType mysql -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/root/software/apache-hive-2.3.4-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/
SLF4J: Found binding in [jar:file:/root/software/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL: jdbc:mysql://localhost:3306/hivedb?createDatabaseIfNotExist=true&useSSL=false
Metastore Connection Driver : com.mysql.jdbc.Driver
Metastore connection User: root
Starting metastore schema initialization to 2.3.0
Initialization script hive-schema-2.3.0.mysql.sql
Initialization script completed
schemaTool completed
```

初始化成功

图17

3.4 Hive 连接

在任意目录下使用 Hive 的三种连接方式之一：**CLI 启动 Hive**。由于已经在环境变量中配置了 HIVE_HOME，所以这里直接在命令行执行如下命令即可：

```
hive
或者
hive --service cli
```

效果如下图所示：

```
→ ~ hive --service cli
which: no hbase in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root/
apache-hive-2.3.4-bin/bin)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/root/software/apache-hive-2.3.4-bin/lib/log4j-slf4j-
SLF4J: Found binding in [jar:file:/root/software/hadoop-2.7.7/share/hadoop/common/lib/s
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/root/software/apache-hive-2.3.4-bi
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Con
hive> █ 连接成功
```

图18

可以使用如下命令退出 Hive 客户端：

```
hive> exit;
或者
hive> quit;
```

2.3 Hive 常见属性配置

任务目的

- 了解 Hive 数据仓库位置的相关配置信息
- 掌握在客户端显示当前数据库以及表头信息的配置步骤
- 了解 Hive 运行日志相关的配置信息

任务清单

- 任务1: Hive 数据仓库位置配置
- 任务2: 查询后信息显示配置
- 任务3: Hive 运行日志信息配置

任务步骤

任务1: Hive 数据仓库位置配置

default 数据仓库的最原始位置是在 HDFS 上的: `/user/hive/warehouse` 路径下。

在仓库目录下, 没有对默认的数据库 default 创建文件夹。如果某张表属于 default 数据库, 直接在数据仓库 `/user/hive/warehouse` 目录下创建一个文件夹。

要想修改 default 数据仓库原始位置, 方式是将 `hive-default.xml.template` 如下配置信息拷贝到 `hive-site.xml` 文件中, 并将 value 值修改为自己想要的 HDFS 路径。

```
<!--Hive默认数据库的位置 -->
<property>
<name>hive.metastore.warehouse.dir</name>
<value>/user/hive/warehouse</value>
<description>location of default database for the warehouse</description>
</property>
```

任务2: 查询后信息显示配置

Hive 中默认是**不显示当前使用的数据库和查询表的头信息**的, 我们可以在 `hive-default.xml.template` 配置文件中查询涉及这两项内容的默认配置:

```
<!-- 客户端显示当前数据库名称信息 -->
<property>
<name>hive.cli.print.current.db</name>
<value>false</value>
<description>Whether to include the current database in the Hive prompt.</description>
```



```

</property>
<!-- 客户端显示当前查询表的头信息 -->
<property>
<name>hive.cli.print.header</name>
<value>false</value>
<description>Whether to print the names of the columns in query output.</description>
</property>

```

为了让 Hive 使用起来更便捷，我们可以在 hive-site.xml 文件中添加如下配置信息，这样就可以实现**显示当前数据库**，以及**查询表的头信息**配置了。

```

<!-- 客户端显示当前数据库名称信息，默认为false -->
<property>
<name>hive.cli.print.current.db</name>
<value>true</value>
<description>Whether to include the current database in the Hive prompt.</description>
</property>
<!-- 客户端显示当前查询表的头信息，默认为false -->
<property>
<name>hive.cli.print.header</name>
<value>true</value>
<description>Whether to print the names of the columns in query output.</description>
</property>

```

在任意目录下使用 Hive 的三种连接方式之一：**CLI 启动 Hive**。对比配置前后差异。

```

hive
或者
hive --service cli

```

配置前，如下图所示：

```

hive> select * from student;
OK
1001    shiny    23
1002    cendy    22
1003    angel    23
1009    ella    21
1012    eva    24
Time taken: 3.796 seconds, Fetched: 5 row(s)
hive>

```

图1：配置前

配置后，如下图所示：

```
hive (default)> select * from student;
OK
student.id      student.name    student.age
1001    shiny      23
1002    cendy      22
1003    angel      23
1009    ella       21
1012    eva        24
Time taken: 12.807 seconds, Fetched: 5 row(s)
```

图2: 配置后

任务3: Hive 运行日志信息配置

在很多程序中, 我们都可以通过输出日志的形式来得到程序运行情况, 通过这些输出日志来调试程序, Hive也不例外。在Hive中, 使用的是 Log4j 来输出日志, 默认情况下, CLI 是不能将日志信息输出到控制台的。在 Hive0.13.0 之前的版本, 默认的日志级别是 WARN, 在 Hive0.13.0 开始, 默认的日志级别是 INFO。默认的日志存放在 /tmp/\${user.name} 文件夹的 hive.log 文件中, 全路径就是 /tmp/\${user.name}/hive.log。

在 \${HIVE_HOME}/conf/hive-log4j2.properties.template 文件中记录了 Hive日志的存储情况, 默认的存储情况如下所示:

```
property.hive.log.dir = ${sys:java.io.tmpdir}/${sys:user.name} # 默认的存储位置
property.hive.log.file = hive.log # 默认的文件名
```

我们可以通过修改以上两个参数的值来重新设置 Hive log 日志的存放地址。

首先, 将 \${HIVE_HOME}/conf 目录下的 hive-log4j2.properties.template 文件复制一份并重命名为 hive-log4j2.properties, 具体命令如下所示:

```
cp hive-log4j2.properties.template hive-log4j2.properties
```

效果图如下所示:

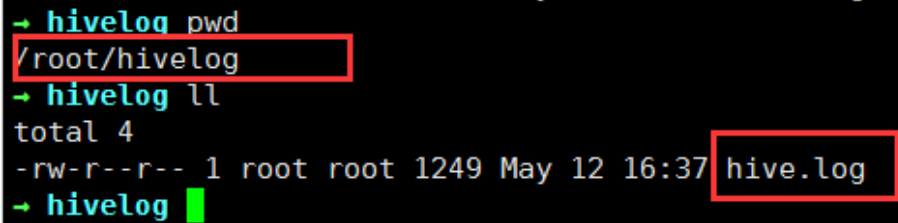
```
→ conf cp hive-log4j2.properties.template hive-log4j2.properties
→ conf ll
total 300
-rw-r--r-- 1 root root 1596 Nov 1 2018 beeline-log4j2.properties.template
-rw-r--r-- 1 root root 257573 May 12 14:57 hive-default.xml.template
-rw-r--r-- 1 root root 2328 May 11 17:12 hive-env.sh
-rw-r--r-- 1 root root 2365 Nov 1 2018 hive-env.sh.template
-rw-r--r-- 1 root root 2274 Nov 1 2018 hive-exec-log4j2.properties.template
-rw-r--r-- 1 root root 2925 May 12 16:30 hive-log4j2.properties
-rw-r--r-- 1 root root 2925 May 12 15:30 hive-log4j2.properties.template
-rw-r--r-- 1 root root 1299 May 12 15:05 hive-site.xml
-rw-r--r-- 1 root root 2060 Nov 1 2018 ivysettings.xml
-rw-r--r-- 1 root root 2719 Nov 1 2018 llap-cli-log4j2.properties.template
-rw-r--r-- 1 root root 7041 Nov 1 2018 llap-daemon-log4j2.properties.template
-rw-r--r-- 1 root root 2662 Nov 1 2018 parquet-logging.properties
```


图3

之后使用 vi 编辑器进行编辑，将 Hive 日志配置到 /root/hivelog/hive.log 文件中。

```
property.hive.log.dir = /root/hivelog  
property.hive.log.file = hive.log
```

重新启动 Hive，验证新的日志文件是否自动创建：



A terminal window showing the execution of commands to verify the Hive log configuration. The first command is `hivelog pwd`, which returns `/root/hivelog`. The second command is `hivelog ll`, which lists the contents of the directory, showing a file named `hive.log` with permissions `-rw-r--r--`, size `1249`, and timestamp `May 12 16:37`. The third command is `hivelog`, which shows a green cursor.

```
→ hivelog pwd  
/root/hivelog  
→ hivelog ll  
total 4  
-rw-r--r-- 1 root root 1249 May 12 16:37 hive.log  
→ hivelog
```

图4

从上图可以看出，我们成功将 Hive 日志存放路径修改为 /root/hivelog/hive.log

。