任务 2-4　Hive 体验

《Hive 数据仓库应用教程》

课堂讲授脚本（中文口语体 + 拼音标注）

李阳源教授　2026/3/9

使用说明：本脚本按幻灯片（PPT）页码编排，教授翻页时对照朗读即可。重要术语和难词附有拼音(pīnyīn)标注。代码部分用灰色底色框标出，需要在终端实际操作演示。

幻灯片 1 封面 — 任务 2-4 Hive 体验

讲授脚本

同学们好!

上节课我们已经成功地把 Hive 安装(ānzhuāng)好了，

也验证(yànzhèng)了 Hive 可以正常启动(qǐdòng)。

那么今天呢，我们先不急着学习 Hive 的详细理论(lǐlùn)知识，

我们先来 **体验(tǐyàn)** 一下 Hive。

通过几个简单的体验，让大家对 Hive 有一个**直观的感受(gǎnshòu)**。

今天一共有三个体验任务(rènwù)：

第一个，Hive 使用起来和 MySQL 差不多吗?

第二个，Hive 如何将结构化(jiégòuhuà)数据映射(yìngshè)成为表?

第三个，使用 Hive 处理小数据效率(xiàolǜ)如何?

好，我们开始吧!

请大家跟着我一起操作(cāozuò)。

幻灯片 2 2.4.1 体验 1：Hive 使用起来和 MySQL 差不多吗？（思考）

讲授脚本

好，现在请大家先思考(sīkǎo)一个问题。

你们之前学过 MySQL 对不对?

MySQL 是一个典型的**关系型数据库(guānxìxíng shùjùkù)**。

现在我们安装了 Hive，那 Hive 看起来呢，跟 MySQL 好像很像(hěn xiàng)。

比如说，Hive 的**数据模型(shùjù móxíng)**——数据库、表、字段(zìduàn)这些概念(gàiniàn)跟 MySQL 是一样的。

然后呢，Hive SQL 也是一种 **类SQL语言(lèi SQL yǔyán)**，

语法(yǔfǎ)跟标准 SQL 非常接近。

那么问题来了：既然看起来这么像，

实际使用起来到底如何呢?

是不是跟 MySQL 一模一样(yī mó yī yàng)呢?

我们现在就来亲自(qīnzì)试一试!

幻灯片 3 2.4.1 体验 1：体验过程 — 创建数据库和表

讲授脚本

好，现在我们来按照 MySQL 的思维(sīwéi)方式，在 Hive 里面做一些基本操作。

我们要做什么呢?

创建数据库、切换(qiēhuàn)数据库、创建表、插入(chārù)数据、查询(cháxún)数据。

跟 MySQL 完全一样的流程(liúchéng)。

请大家打开 beeline 客户端(kèhùduān)，跟我一起输入(shūrù)：

操作步骤 — 请跟着输入

第一步：启动 Beeline 连接 Hive

```
-- 体验1
~/hadoop.sh
1번 선택
N 선택(현재는 PyCharm이 필요없음)
hive-bee
```

这个命令会连接(liánjiē)到 HiveServer2。

等它连上以后，我们就可以开始了。

第二步：查看现有数据库

```
show databases;    -- 列出所有数据库
```

大家看，现在只有一个 **default** 数据库，

这是 Hive 默认(mòrèn)的数据库。

第三步：创建新数据库并切换

```
create database itcast;     -- 创建数据库
use itcast;                 -- 切换数据库
show tables;                -- 查看表（应该是空的）
```

好的，我们成功创建了 itcast 数据库，并且切换过去了。

现在里面还没有任何表。

第四步：创建表

```
create table t_user (id int, name string);
```

注意(zhùyì)这里！

跟 MySQL 不一样的地方来了——Hive 里的字符串(zìfúchuàn)类型不叫 **varchar**，

而是叫 **string**。这是一个小区别(qūbié)，大家记住。

第五步：插入数据（重点观察！）

```
-- 这里要注意观察所花的时间！大约 30秒
insert into table t_user values(1,'zhangsan');
```

【重要提醒】同学们，请注意看屏幕！输入这条 insert 语句以后，**不要着急(zháojí)**，要耐心等待(nàixīn děngdài)。

大家看一下时间，大概需要**30秒左右**才能执行(zhíxíng)完毕。

这跟 MySQL 完全不同！

MySQL 插入一条数据几乎是瞬间(shùnjiān)完成的，对不对？

但是 Hive 为什么这么慢(màn)呢？我们等一下再解释。

第六步：查询验证

```
select * from t_user;
```

好的，数据插入成功了！我们可以看到 id 是 1，name 是 zhangsan。

功能上没问题，但是那个速度……确实太慢了！

幻灯片 4 2.4.1 体验 1：体验过程 — 为什么插入这么慢?

讲授脚本

好，刚才大家都亲眼看到了——插入一条数据花了大约30秒。

那为什么会这么慢呢？我来给大家解释(jiěshì)一下。

原因一：Hive 的设计目的(shèjì mùdì)不一样。Hive 是为**批量数据分析(pīliàng shùjù fēnxī)**设计的，不是为**实时事务处理(shíshí shìwù chǔlǐ)**设计的。

MySQL 是 OLTP 系统，Hive 是 OLAP 系统，它们的定位(dìngwèi)完全不同。

原因二：每一条 insert 语句，Hive 底层(dǐcéng)都要启动一个完整的 **MapReduce 作业(zuòyè)** 或者 Tez 作业。启动、调度(diàodù)、执行、完成——整个过程开销(kāixiāo)非常大。

原因三：每条插入都会在 HDFS 上生成(shēngchéng)一个小文件，这对 HDFS 的 **NameNode** 会造成很大压力(yālì)。

所以同学们要理解(lǐjiě)——Hive **不适合(bù shìhé)**一条一条地插入数据，它天生就不是干这个事情的!

幻灯片 5 2.4.1 体验 1：结论

讲授脚本

好，我们来总结(zǒngjié)一下体验1的结论(jiélùn)：

第一，Hive SQL 的语法(yǔfǎ)跟标准 SQL 确实很类似(lèisì)，学习成本(chéngběn)不高。如果你会写 SQL，那基本上就会写 Hive SQL。

第二，但是呢，Hive 的底层(dǐcéng)执行方式不一样——它是通过 **MapReduce** 来执行的，所以速度很慢。

第三，如果面对大数据集，这样一条一条插入是**完全不现实的(wánquán bù xiànshí de)**，成本太高了。

第四，那 Hive 到底怎么高效地**加载**(jiāzǎi)数据呢?

答案是——**把结构化文件直接映射(yìngshè)成表**！这才是 Hive 的正确用法(yòngfǎ)。

好，这就引出了我们第二个体验！

幻灯片 6 2.4.2 体验 2：Hive 如何将结构化数据映射成为表？（思考）

讲授脚本

好，我们已经知道了——用 insert + values 的方式插入数据效率(xiàolǜ)很低。

那现在我们回到 Hive 的**本质(běnzhì)**：

Hive 的核心(héxīn)能力是什么？

就是**把结构化(jiégòuhuà)的数据文件映射成为一张表**，

然后用 SQL 来查询分析(fēnxī)。

那么问题来了：假如(jiǎrú)我现在有一份结构化的数据文件，

我要怎么才能让 Hive 成功识别(shíbié)它呢？

需要注意哪些问题？

大家想一想：

1）文件要放在什么**路径(lùjìng)**？

2）建表时**字段类型(zìduàn lèixíng)**要怎么定义(dìngyì)？

3）**字段顺序(shùnxù)**重要吗？

4）文件中字段之间的**分隔符(fēngéfú)**怎么处理？

带着这些问题，我们来动手实验(shíyàn)！

幻灯片 7 2.4.2 体验 2：体验过程 — 准备数据文件

讲授脚本

首先(shǒuxiān)，我们要在 HDFS 上准备(zhǔnbèi)一个数据文件。

我们要在 HDFS 的 **/user/hive/warehouse** 目录(mùlù)下创建一个文件叫 **user.txt**。

这个文件里面的内容(nèiróng)是这样的——每行(háng)一条记录，用逗号(dòuhào)分隔：

数据文件内容 user.txt

```
1,zhangsan,18
2,lisi,24
3,allen,22
4,james,28
```

大家看，这是一个非常简单的结构化文件：第一列是 id，第二列是 name，

第三列是 age，用逗号分隔。

现在请大家打开浏览器(liúlǎnqì)，访问(fǎngwèn) HDFS 的 Web 界面(jièmiàn)：

```
use itcast;

create table t_user2 (id int, name string, age int);

show tables; (2개의 테이블 확인, t_user, t_user2)

http://localhost:9870/explorer.html#
```

在这个界面上，找到路径 **user → hive → warehouse→itcast.db→t_user2**，然后上传

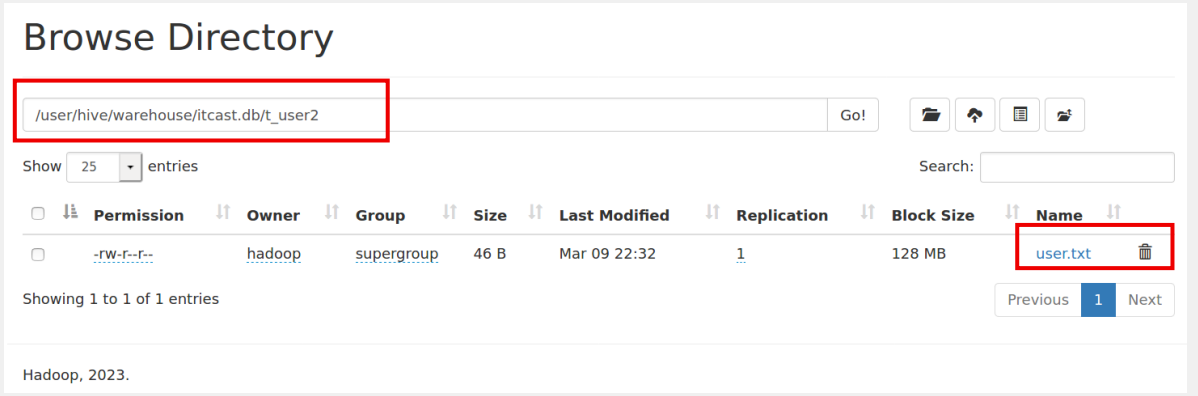
(shàngchuán)这个 user.txt 文件。→ 브라우저에 있는 upload는 동작 안됨.

也可以用命令行的方式上传：

```
# 先在本地创建 user.txt 文件
vim /tmp/user.txt
# 输入以下内容（逗号分隔）：
# 1,zhangsan,18
# 2,lisi,24
# 3,allen,22
# 4,james,28


# 然后上传到 HDFS
hdfs dfs -put /tmp/user.txt /user/hive/warehouse/itcast.db/t_user2
```

```
(base) hadoop@hadoop01:~$ hdfs dfs -put user.txt /user/hive/warehouse/itcast.db/t_user2/
```



幻灯片 8 2.4.2 体验 2：验证 — 第一次尝试（失败）

讲授脚本

好，文件已经放好了。现在我们来查询(cháxún)一下之前体验1中创建的那个 t_user 表——注意，那个表只有 id 和 name 两个字段(zìduàn)，没有 age。

我们在 Beeline（hive-bee） 中执行：

```
select * from t_user;
```

```
0: jdbc:hive2://localhost:10000> select * from t_user2;
+--------------+----------------+---------------+
| t_user2.id   | t_user2.name   | t_user2.age   |
+--------------+----------------+---------------+
| NULL         | NULL           | NULL          |
| NULL         | NULL           | NULL          |
| NULL         | NULL           | NULL          |
| NULL         | NULL           | NULL          |
+--------------+----------------+---------------+
4 rows selected (0.455 seconds)
```

大家看结果——**全是 NULL！**数据并没有映射(yìngshè)成功。

为什么呢？我们来分析一下。文件放在了 /user/hive/warehouse/ 下面，但是 t_user 这张表对应的 HDFS 路径应该是 **/user/hive/warehouse/itcast.db/t_user/**。

猜想(cāixiǎng)1：难道(nándào)数据文件必须放在**表对应的 HDFS 路径**下才行？

而且还有另一个问题——我们建表的时候**没有指定(zhǐdìng)分隔符**。

Hive 默认的分隔符是 \001（就是 Ctrl+A），不是逗号。

所以即使文件放对了位置，不指定分隔符也会出问题。

那我们来做第二次尝试(chángshì)！

幻灯片 9 2.4.2 体验 2：验证 — 重建表并指定分隔符（成功）

讲授脚本

这次我们来正确地做！我们重新(chóngxīn)建一张表，这次要做两件事：

1）字段(zìduàn)要包含 id、name、age 三个，跟文件内容一致(yīzhì)。

2）**指定分隔符为逗号。**

操作步骤 — 请跟着输入

第一步：创建新表，指定分隔符

```
create table t_user3 (id int, name string, age int)
row format delimited
fields terminated by ',';
```

大家注意看这两行新语法：**row format delimited** 表示行格式(háng géshì)是分隔的；

fields terminated by ',' 表示字段之间用逗号来分隔。这是 Hive 特有(tèyǒu)的语法！

第二步：将数据文件放到表对应的 HDFS 路径下

```
hdfs dfs -put user.txt /user/hive/warehouse/itcast.db/t_user3/
```

注意！这次我们把文件放到了 **t_user1 表对应的目录**下面。

第三步：查看 HDFS 目录结构

```
hdfs dfs -ls -R /user/hive/warehouse
```

第四步：查询数据

```
select * from t_user3;
```

大家看结果——**成功了！**

```
+-------------+-------------+-------------+
| t_user3.id  | t_user3.name | t_user3.age  |
+-------------+-------------+-------------+
| 1           | zhangsan     | 18           |
| 2           | lisi         | 24           |
| 3           | allen        | 22           |
| 4           | james        | 28           |
+-------------+-------------+-------------+
```

```
0: jdbc:hive2://localhost:10000> select * from t_user3;
+-------------+---------------+--------------+
| t_user3.id  | t_user3.name  | t_user3.age  |
+-------------+---------------+--------------+
| 1           | zhangsan      | 18           |
| 2           | lisi          | 24           |
| 3           | allen         | 22           |
| 4           | james         | 28           |
+-------------+---------------+--------------+
```

这次数据完全正确！id 是整数(zhěngshù)，name 是字符串，age 也是整数。映射(yìngshè)成功！

而且大家注意——这次**没有使用 insert 语句**，也**没有经过 MapReduce**，数据直接就加载(jiāzǎi)进来了，速度非常快！

这才是 Hive 加载数据的正确方式。

幻灯片 10 2.4.2 体验 2：验证 — 字段类型不匹配会怎样?

讲授脚本

好，接下来我们再做一个实验(shíyàn)——如果我故意(gùyì)把字段类型搞错，会发生(fāshēng)什么?

操作步骤 — 请跟着输入

创建一张字段类型故意搞错的表：

```
-- 注意：name 本来是字符串，这里故意写成 int
-- age 本来是整数，这里故意写成 string
create table t_user4 (id int, name int, age string)
row format delimited
fields terminated by ',';
```

然后把同样的数据文件也放进去：

```
hdfs dfs -put user.txt /user/hive/warehouse/itcast.db/t_user4/
```

现在查询：

```
select * from t_user2;
```

大家看结果：

```
+-------------+--------------+--------------+
| t_user4.id  | t_user4.name | t_user4.age  |
+-------------+--------------+--------------+
| 1           | NULL         | 18           |
| 2           | NULL         | 24           |
| 3           | NULL         | 22           |
| 4           | NULL         | 28           |
+-------------+--------------+--------------+
```

发现了吗？**name 列全部显示 NULL**，但是 age 列却正常(zhèngcháng)显示了！

为什么呢？因为 name 列的数据本来是字符串（zhangsan、lisi……），你让 Hive 把它转换(zhuǎnhuàn)成 int？这当然转不了！所以显示 NULL。

但是 age 列呢？本来是数字 18、24……你让 Hive 把数字转成 string？这个可以的！数字

转字符串是没问题的。

所以结论是：**Hive 有自动类型转换(zìdòng lèixíng zhuǎnhuàn)功能**，但是**不保证(bù bǎozhèng)一定成功**。转换失败就显示 NULL。

幻灯片 11-12 2.4.2 体验 2：体验过程 — HDFS 目录结构

讲授脚本

现在我们打开浏览器，看一下 HDFS 的目录结构(mùlù jiégòu)。

```
http://localhost:9870/explorer.html#/
```

请大家按照这个路径导航(dǎoháng)：

/ → user → hive → warehouse → itcast.db → t_user →

大家可以看到，Hive 在 HDFS 上的存储(cúnchú)结构是这样的：

根目录下有 **/user/hive/warehouse/** 这是 Hive 的默认(mòrèn)仓库(cāngkù)路径。

每个数据库对应一个 **.db 后缀(hòuzhuì)的目录**，比如 itcast.db。

每张表对应数据库目录下的一个**子目录(zǐ mùlù)**，比如 t_user。

表目录下面就是实际(shíjì)的数据文件。

这就是 Hive 的**"读时模式"(dú shí móshì)**——Schema on Read。数据先存进去，读取(dúqǔ)的时候才去解析(jiěxī)。跟 MySQL 的 Schema on Write 完全不同！

幻灯片 13 2.4.2 体验 2：结论

讲授脚本

好，我们来总结体验2的结论：

第一，要让 Hive 成功映射结构化文件，建表时的**字段顺序和字段类型**必须和文件内容保持一致(bǎochí yīzhì)。

第二，如果类型不一致，Hive 会尝试(chángshì)类型转换，但不一定成功。失败了就显示 NULL。

第三，数据文件似乎(sìhū)需要放在表对应的 HDFS 目录下。其他路径(lùjìng)行不行？这个我们以后再深入(shēnrù)探讨(tàntǎo)。

第四，建表时需要根据文件内容指定分隔符(fēngéfú)。如果不指定呢？默认是 \001，那逗号分隔的文件就读不了了。

这些问题都是 Hive 的重要知识点(zhīshì diǎn)，我们后续(hòuxù)课程会一个一个深入学习。

幻灯片 14 2.4.3 体验 3：使用 Hive 小数据分析如何？（思考）

讲授脚本

好，我们进入最后一个体验。这个体验的问题是：

我们知道 Hive 是基于 HDFS 的，理论上可以存储(cúnchú)海量(hǎiliàng)数据，天生适合大数据分析。

但是，如果 Hive 里的数据很少呢？只有几条记录(jìlù)？用 Hive 来分析这种**小数据(xiǎo shùjù)**，效率(xiàolǜ)如何？

我们来试试看！

幻灯片 15 2.4.3 体验 3：体验过程 — 小数据聚合查询

讲授脚本

好，我们用之前创建的 t_user1 表来做一个简单的分析。先看一下表里有什么数据：

操作步骤 — 请跟着输入

第一步：查看表数据

```
show tables;
select * from t_user1;
```

好的，表里一共就4条数据。现在我们来做一个**聚合(jùhé)查询**——查一下年龄大于(dàyú)20岁的有几个人？

第二步：执行聚合查询（注意观察时间！）

```
select count(id) from t_user1 where age > 20;
```

【重要提醒】同学们，请再次注意看时间！这个查询对 MySQL 来说几乎是瞬间(shùnjiān)完成的——4条数据，一个 count 加一个 where，太简单了对不对？

但是在 Hive 里面……大家看，又启动了 MapReduce！

最终结果是 **3**（lisi 24岁、allen 22岁、james 28岁，三个人大于20岁）。结果没问题。

但是！花了多长时间？**大约47秒！**47秒啊！MySQL 可能只需要几毫秒(háomiǎo)。

这就说明了——对小数据来说，Hive 的执行效率(zhíxíng xiàolǜ)是非常低的。因为 MapReduce 的启动和调度本身就需要很长时间，这个固定开销(gùdìng kāixiāo)是省不掉的。

幻灯片 16 2.4.3 体验 3：结论

讲授脚本

好，我们来总结体验3的结论：

第一，Hive 底层(dǐcéng)确实是通过 **MapReduce 执行引擎(zhíxíng yǐnqíng)**来处理数据的。不管你的数据量多少，它都要走 MapReduce 的流程。

第二，执行一个 MapReduce 程序需要不短的时间——启动 JVM、分配(fēnpèi)资源(zīyuán)、调度(diàodù)任务、写中间结果、最终输出……这些都需要时间。

第三，如果是**小数据集**，使用 Hive 来分析完全是**得不偿失(dé bù cháng shī)**的，延迟(yánchí)非常高。这种场景(chǎngjǐng)应该用 MySQL 或者其他 OLTP 数据库。

第四，但是如果是**大数据集**——几个 TB、几十个 TB 的数据——使用 Hive 就非常合适(héshì)了！因为底层 MapReduce 可以**分布式(fēnbùshì)并行(bìngxíng)计算**，速度反而(fǎn'ér)很快！

总结回顾

好的同学们，今天我们完成了三个 Hive 体验：

体验1：Hive SQL 语法跟 MySQL 很像，但底层执行机制(jīzhì)完全不同，insert 很慢。

体验2：Hive 的正确使用方式是把结构化文件映射成表，

需要注意路径、字段类型、分隔符。

体验3：Hive 适合大数据分析，不适合小数据处理。

从下节课开始，我们就要正式(zhèngshì)深入学习 Hive 的各个知识点了。

今天的课就到这里，下课(xiàkè)！