模块五 Hive数据操作



《Hive数据仓库应用教程》

目 录 / CONTENTS

1 任务5-1 数据导入

2 任务5-2 数据导出

3 任务5-3 本章小结

4 任务5-4 本章习题

任务 5-1 数据导入

5.1.1 任务描述

本小节旨在讨论如何装载数据到创建的表中，因此，本小节假定学生已经理解了 SQL 的基础知识，因此这些语句对读者来说并不陌生，如果学对此并不熟悉，那么请到相关章节获取相关更详细的介绍。

处理数据并形成信息需要对数据进行呈现。Hive 生态系统可以接受任何可用分隔符来结构化的数据。

既然 Hive 没有行级别的数据插入、数据更新和删除操作，那么往表中装载数据的唯一途径就是使用一种“大量”的数据装载操作。或者通过其他方式仅仅将文件写入到正确的目录下。这两种方式是本小节旨在着重说明的部分。







5.1.2 任务准备知识

在本章节中，为了避免与前面几个章节的数据库中的数据发生冲突，因此，在本章节中，重新创建一个新的数据库，名称为 unit5db。并且在本章的讲解中，一直使用该数据库，此处请读者注意。

为了更好的说明本小节的内容，需要说明该数据库的创建过程。

具体的创建过程如下所述。

例题 1：创建本小节所需要的数据库 unit5db。

操作过程：

step01： 创建数据库 unit5db

创建本小节所需要的数据库，名称为 unit5db，具体操作命令如下：

```
hive> CREATE DATABASE IF NOT EXISTS unit5db;
```

step02： 查看数据库 unit5db

查看创建的数据库 unit5db 是否成功，具体操作命令如下：

```
hive> SHOW DATABASES;
```

step03： 使用数据库 unit5db

使用该数据库 unit5db，具体操作命令如下： hive> USE unit5db;

5.1.2 任务准备知识

创建数据库 unit5db、使用数据库 unit5db 的操作过程，以上步骤的操作的具体执行和操作结果如下图 5-1 所示，由操作结果可以看出，创建数据库 unit5db成功、使用数据库 unit5db 成功。

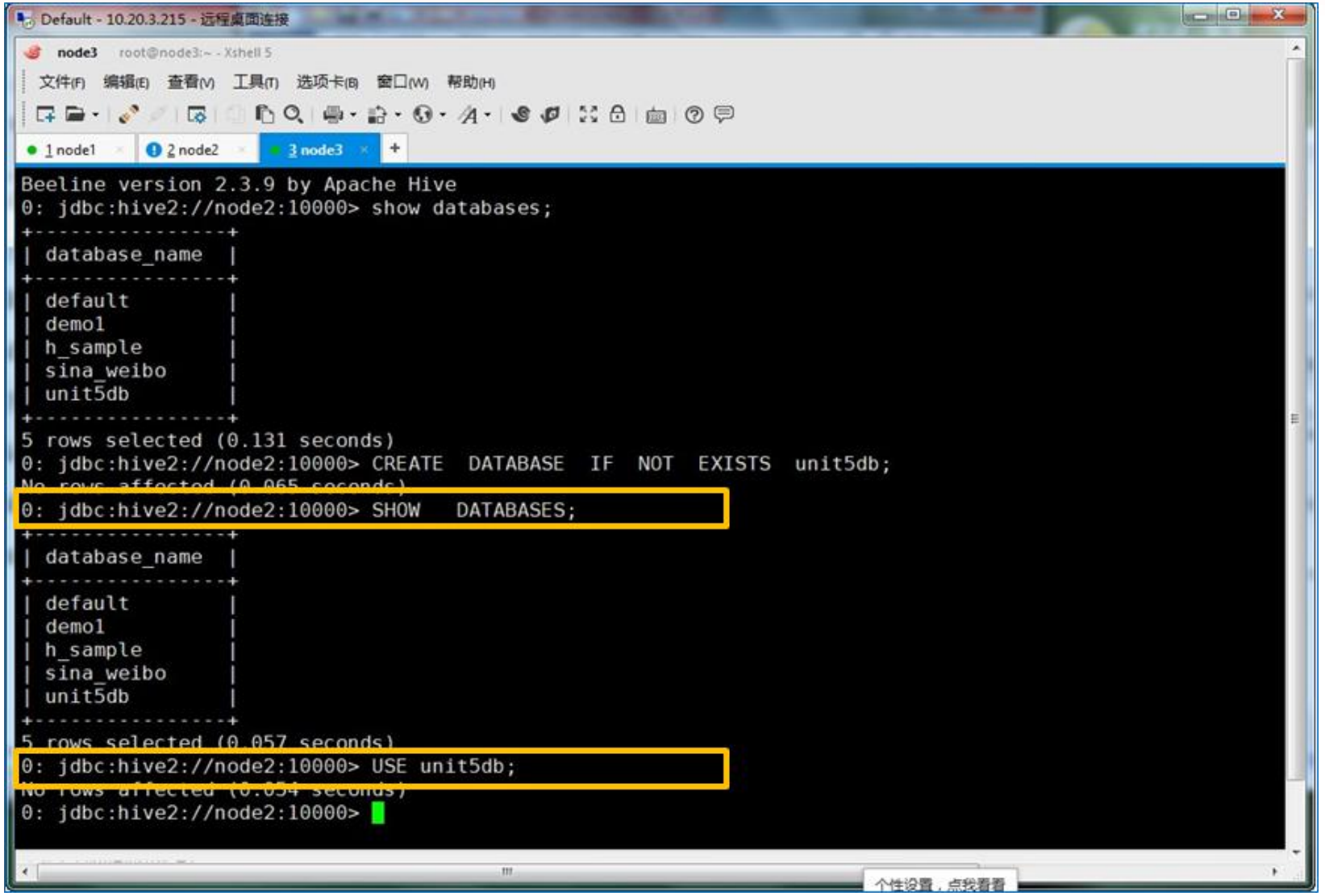


图 5-1

Hive生态系统中向表中装载数据的方式主要有

① 直接向表中加载数据、

② 通过 Load 加载数据、

③ 通过查询加载数据、

④ 查询语句中创建表并加载数据、

⑤ import 导入 Hive 表数据

等五种。

一．直接向表中加载数据

在使用 Hive 进行数据操作时，有时候需要插入少量的测试数据，那么，可以使用 INSERT INTO 语句。

语法格式（standard syntax）：

```
INSERT INTO TABLE tablename
PARTITION (partcol1 [=val1], partcol2[=val2] ...) ]
VALUES values_row [, values_row ...]
```

例题 2：向学生信息表students 中插入 2 条新记录，分别是('fred 1.28)和('barney rubble', 32, 2.32)。

操作过程：

```
--step01：创建学生信息表 students
--创建本例题所需要的学生信息表 students，具体操作命令如下：
CREATE TABLE students (name string, age int, gpa decimal(3,2)
) CLUSTERED BY (age) INTO 2 BUCKETS STORED as orc;

--step02：查看学生信息表 students
--查看创建的学生信息表 students 是否成功，具体操作命令如下：
show tables;

--step03：查看学生信息表 students 中的数据
-- 查看创建的学生信息表 students 是否成功，具体操作命令如下：
SELECT * FROM students;
```

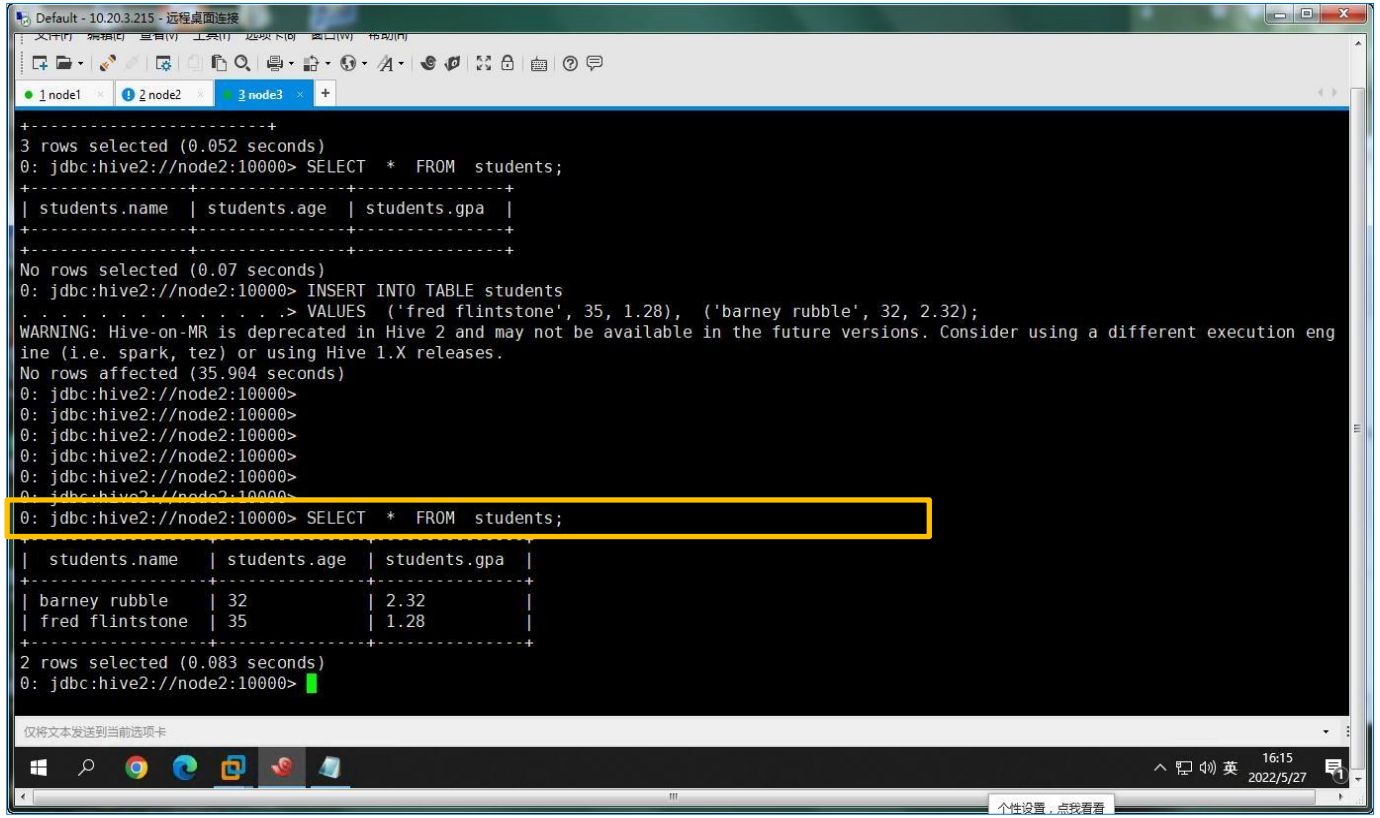
5.1.3 任务实施

```
--step04：将数据插入到学生信息表 students 中
--将数据插入到学生信息表 students 中，具体操作命令如下：
INSERT INTO TABLE students
VALUES ('fred flintstone',35,1.28), ('barney rubble',32,2.32);

--step05：再次查看表中的数据
--再次查看学生信息表 students 中的数据，具体操作命令如下：
SELECT * FROM students;
```

5.1.3 任务实施

上述步骤的操作过程和操作结果如下图 5-3 所示，由操作结果可以看出，目前，学生信息表 students 插入 2 条新数据成功。



5.1.3 任务实施

例题3：向浏览信息表pageviews中插入2条新记录，
分别是('jsmith', 'mail.com', 'sports.com')和('jdoe', 'mail.com', null),,

并更新表中的2条记录('tjohnson', 'sports.com', 'finance.com', '2026-02-07')
和
('tlee', 'finance.com', null, '2026-02-07')。

操作过程：

5.1.3 任务实施

```
--step01：创建浏览信息表pageviews
--创建本例题所需要的浏览信息表pageviews，具体操作命令如下：
CREATE TABLE pageviews (
userid string,
link string,
came_from string
) PARTITIONED BY (stat_date STRING)
CLUSTERED BY (userid) INTO 256 BUCKETS
STORED AS ORC;

--step02：查看浏览信息表pageviews
--查看创建的浏览信息表pageviews是否成功，具体操作命令如下：
SHOW TABLES;

--step03：查看浏览信息表pageviews的数据
--查看创建的浏览信息表pageviews的数据，具体操作命令如下：
SELECT * FROM pageviews;
```

5.1.3 任务实施

```
--step04：将数据插入到浏览信息表 pageviews 中
--将数据插入到浏览信息表 pageviews 中，具体操作命令如下：
INSERT INTO TABLE pageviews PARTITION (stat_date='2026-02-07')
VALUES ('jsmith', 'mail.com','sports.com'),
('jdoe','mail.com',null);

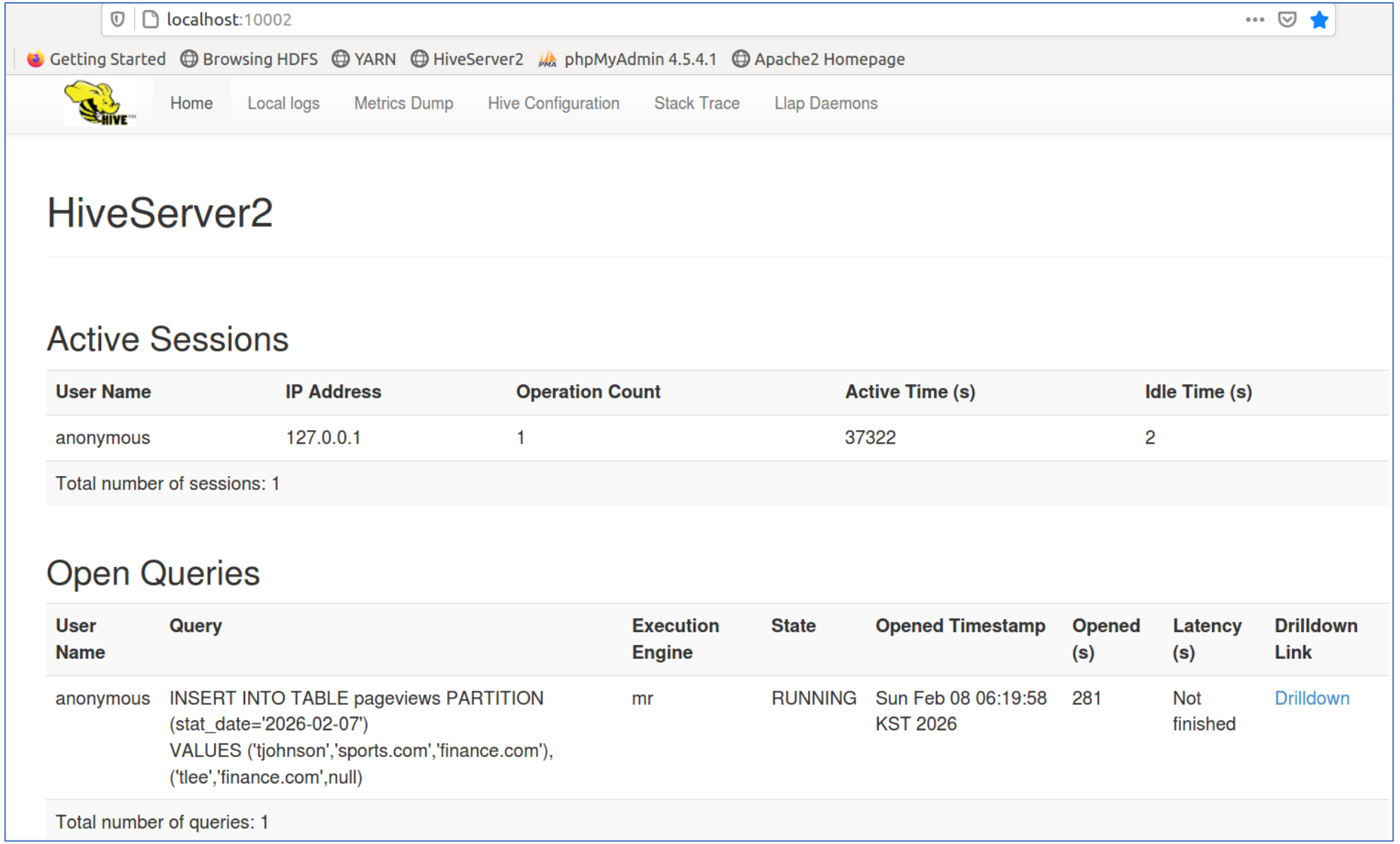

--step05：查看表中的数据
--查看浏览信息表 pageviews 中的数据，具体操作命令如下：
SELECT * FROM pageviews;
--目前，浏览信息表 pageviews 插入 2 条新数据成功，
--且 2 条数据全部插入到分区 '2026-02-07'中。
```

5.1.3 任务实施

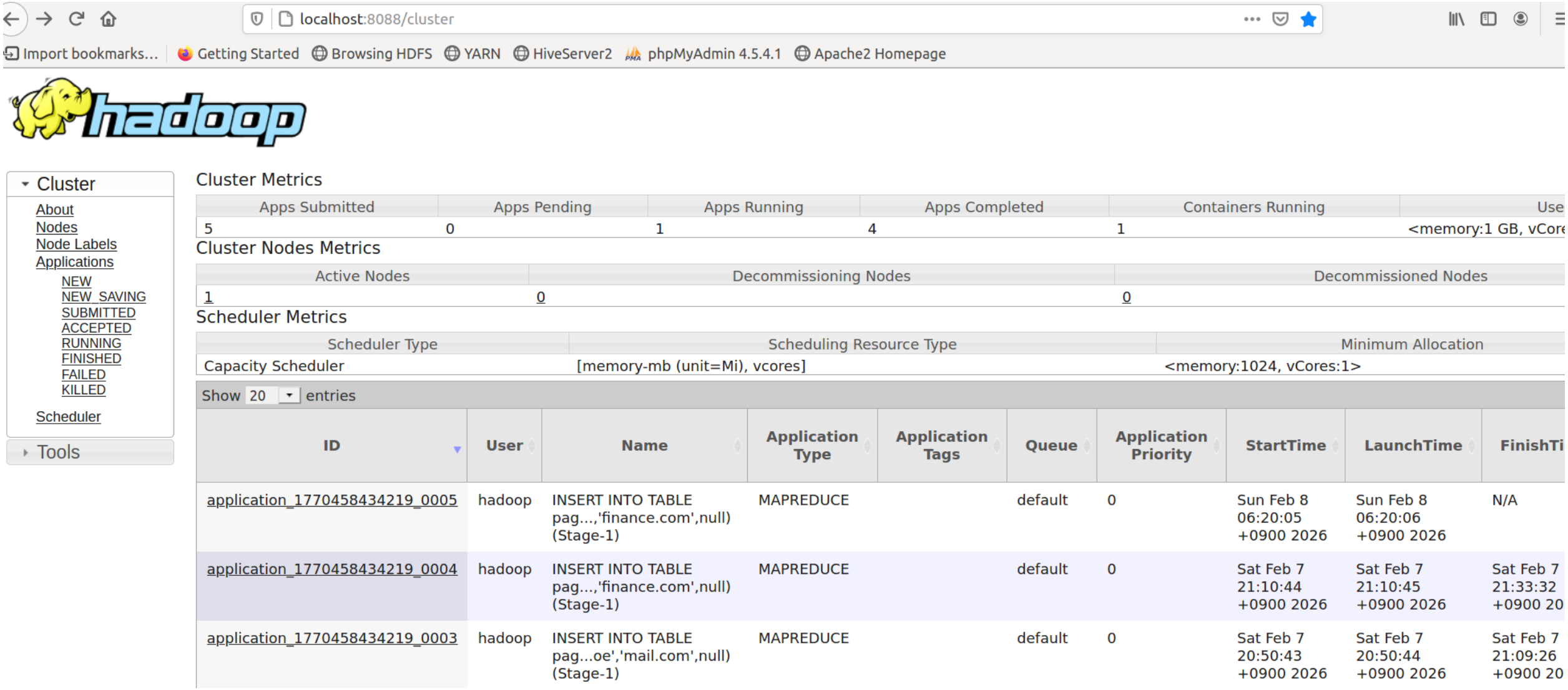
```
--step06：将更新的数据插入到浏览信息表 pageviews 中
--将需要更新的 2 条新数据插入到浏览信息表 pageviews 中，具体操作命令如下：
INSERT INTO TABLE pageviews PARTITION (stat_date='2026-02-07')
VALUES ('tjohnson','sports.com','finance.com'),
('tlee','finance.com',null);

--step07：查看表中的数据
--再次查看浏览信息表 pageviews 中的数据，具体操作命令如下：
SELECT * FROM pageviews;
--由操作结果可以看出，目前，浏览信息表 pageviews 更新插入的 2 条新数据成功，
--且 2 条数据全部插入到分区'2026-02-07'中。
```

5.1.3 任务实施


localhost:10002
Getting Started
Browsing HDFS
YARN
HiveServer2
phpMyAdmin 4.5.4.1
Apache2 Homepage
Home
Local logs
Metrics Dump
Hive Configuration
Stack Trace
Llap Daemons
HiveServer2
Active Sessions
User Name | IP Address | Operation Count | Active Time (s) | Idle Time (s)
anonymous | 127.0.0.1 | 1 | 37322 | 2
Total number of sessions: 1
Open Queries
User Name | Query | Execution Engine | State | Opened Timestamp | Opened (s) | Latency (s) | Drilldown Link
anonymous | INSERT INTO TABLE pageviews PARTITION (stat_date='2026-02-07') VALUES ('tjohnson','sports.com','finance.com'), ('tlee','finance.com',null) | mr | RUNNING | Sun Feb 08 06:19:58 KST 2026 | 281 | Not finished | Drilldown
Total number of queries: 1

5.1.3 任务实施


localhost:8088/cluster
Import bookmarks...
Getting Started
Browsing HDFS
YARN
HiveServer2
phpMyAdmin 4.5.4.1
Apache2 Homepage
hadoop
Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools
Cluster Metrics
Apps Submitted
Apps Pending
Apps Running
Apps Completed
Containers Running
Use
5
0
1
4
1
<memory:1 GB, vCore
Cluster Nodes Metrics
Active Nodes
Decommissioning Nodes
Decommissioned Nodes
1
0
0
Scheduler Metrics
Scheduler Type
Scheduling Resource Type
Minimum Allocation
Capacity Scheduler
[memory-mb (unit=Mi), vcores]
<memory:1024, vCores:1>
Show 20 entries
ID
User
Name
Application Type
Application Tags
Queue
Application Priority
StartTime
LaunchTime
FinishTi
application_1770458434219_0005
hadoop
INSERT INTO TABLE pag...,'finance.com',null) (Stage-1)
MAPREDUCE
default
0
Sun Feb 8 06:20:05 +0900 2026
Sun Feb 8 06:20:06 +0900 2026
N/A
application_1770458434219_0004
hadoop
INSERT INTO TABLE pag...,'finance.com',null) (Stage-1)
MAPREDUCE
default
0
Sat Feb 7 21:10:44 +0900 2026
Sat Feb 7 21:10:45 +0900 2026
Sat Feb 7 21:33:32 +0900 20
application_1770458434219_0003
hadoop
INSERT INTO TABLE pag...oe','mail.com',null) (Stage-1)
MAPREDUCE
default
0
Sat Feb 7 20:50:43 +0900 2026
Sat Feb 7 20:50:44 +0900 2026
Sat Feb 7 21:09:26 +0900 20

5.1.3 任务实施

Logged in as: dr.who

Application application_1770458434219_0005

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Kill Application

Application Overview	
User:	hadoop
Name:	INSERT INTO TABLE pag...,'finance.com',null) (Stage-1)
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	RUNNING: AM has registered with RM and started running.
Queue:	default
FinalStatus Reported by AM:	Application has not completed yet.
Started:	Sun Feb 08 06:20:05 +0900 2026
Launched:	Sun Feb 08 06:20:06 +0900 2026
Finished:	N/A
Elapsed:	1mins, 50sec
Tracking URL:	ApplicationMaster
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

Application Metrics	
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	317164 MB-seconds, 305 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds

Hive生态系统中向表中装载数据的方式主要有

① 直接向表中加载数据、

② 通过 Load 加载数据、

③ 通过查询加载数据、

④ 查询语句中创建表并加载数据、

⑤ import 导入 Hive 表数据

等五种。

二、通过 Load 加载数据

Hive 是一种底层封装了 Hadoop 的数据仓库处理工具，使用类 SQL 的 HiveSQL 语言实现数据查询。

Hive 在加载数据过程中不会对数据进行任何的修改，只是将数据移动到 HDFS 中 Hive 设定的目录下，因此，Hive 不支持对数据的改写和添加，所有的数据都是在加载的时候确定的。

Hive 在向表中装载数据的时候，除了直接向表中插入数据的方式之外，还可以通过 Load 加载数据。

二、通过 Load 加载数据

1. 通过本地将数据文件加载到 Hive 表中

在使用 Hive 进行操作过程中，有时候需要将本地文件装载到 Hive 表中，那么，可以使用 Load 语句加载数据。

1) 语法格式

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)] [INPUTFORMAT 'inputformat' SERDE 'serde']
(3.0 or later)
```

语法规则之filepath

- **filepath** 表示待移动数据的路径。它既可以指向文件（在这种情况下，Hive会将文件移动到表中），也可以指向目录（在这种情况下，Hive会将该目录中的所有文件移动到表中）。
- **filepath** 文件路径支持以下三种形式，并且需要与 **LOCAL** 关键字一起考虑：

1.相对路径，例如：project/data1

2.绝对路径，例如：/user/hive/project/data1

3.带有schema的完整URI，例如：hdfs://namenode:9000/user/hive/project/data1

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)] [INPUTFORMAT 'inputformat' SERDE 'serde']
(3.0 or later)
```

Hive 只是针对导入该表中的数据按照自己的方式进行处理，对于错误的数据不进行核对处理，对于超出的数据直接丢弃。

Load 操作只是单纯的复制/移动操作，将数据文件移动到 Hive 表中对应的位置。Hive 本身对导入的数据不具备差错的功能，只是按照表字段进行匹配，格式匹配不了的默认用 Null 替代，Hive 属于读时模式。

例题 4：将本地文件 order-info-0208.txt 加载到 Hive 的 order_info_load 表中。

(位置：/usr/datas/order-info-0208.txt)

操作过程：

step01：创建本地文件 order-info-0208.txt

在本地创建本地文件order-info-0208.txt，

在本地/usr 下创建一个文件夹datas，用来存放本章节所有的本地文件，具体操作命令如下：

```
cd /usr
sudo mkdir datas
sudo chown hadoop:hadoop -R datas

touch order-info-0208.txt
vim order-info-0208.txt
1,待支付,2026-02-08 12:20:11,2026-02-08 12:20:11
2,待支付,2026-02-08 14:20:11,2026 02-08 14:20:11
3,待支付,2026-02-08 16:20:11,2026-02-08 16:20:11
```

5.1.3 任务实施

```
--step02：查看数据库 unit5db 的位置
--查看数据库 unit5db 在 Hive 中的位置，具体操作命令如下：
hive-bee
SHOW DATAFILES;
USE unit5db;
DESC DATABASE unit5db;
```

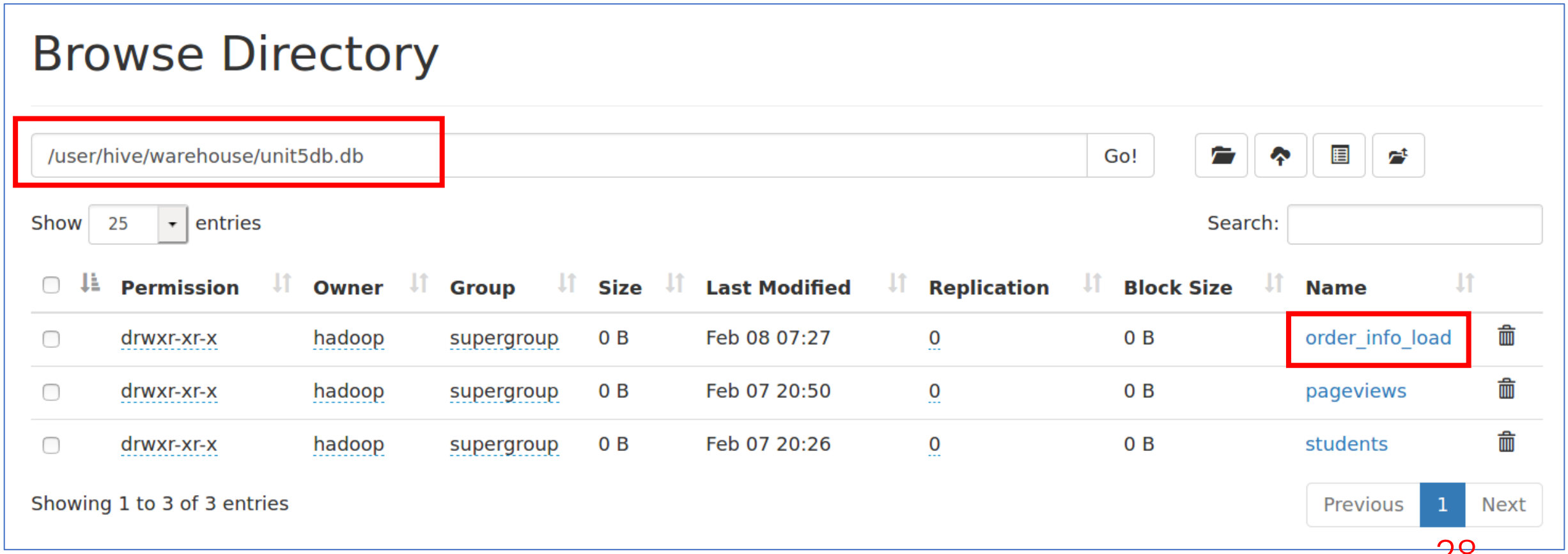
```
0: jdbc:hive2://localhost:10000> DESC DATABASE unit5db;
+----------+----------+-----------------------------------------------------+-------------+-------------+-------------+
| db_name  | comment  |                      location                       | owner_name  | owner_type  | parameters  |
+----------+----------+-----------------------------------------------------+-------------+-------------+-------------+
| unit5db  |          | hdfs://hadoop01:9000/user/hive/warehouse/unit5db.db | hadoop      | USER        |             |
+----------+----------+-----------------------------------------------------+-------------+-------------+-------------+
1 row selected (0.156 seconds)
```

5.1.3 任务实施

```
--step03：创建数据库中的表order_info_load
--在数据库unit5db中，创建表order_info_load，具体操作命令如下：
CREATE TABLE order_info_load(
order_id STRING COMMENT '订单id',
order_status STRING COMMENT '订单状态',
create_time STRING COMMENT '创建时间',
update_time STRING COMMENT '最后更新时间'
)
partitioned BY (stat_date string)
row format DELIMITED
FIELDS TERMINATED by ',';
```

5.1.3 任务实施

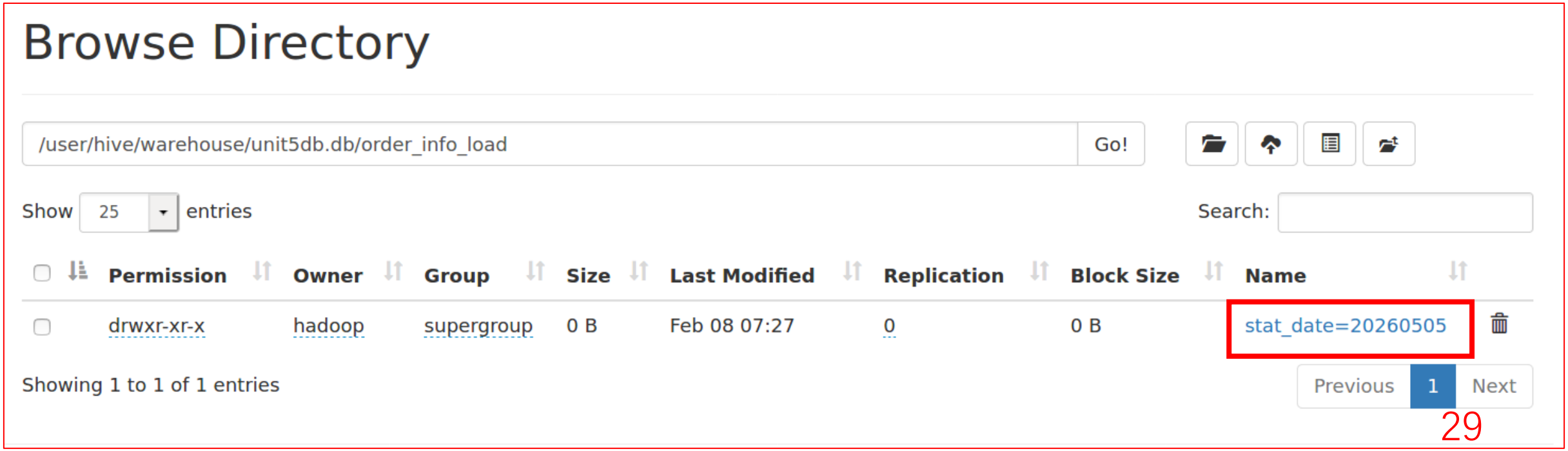
```
--step04：查看数据库中的表
--在数据库unit5db中，查看表order_info_load是否创建成功，具体操作命令如下：
SHOW tables;
```



5.1.3 任务实施

```
--step05：将本地文件 order-info-0208.txt 上传到服务器 Hive 中

--在数据库 unit5db 中， 加载服务器本地 order-info-0208.txt
--文本数据到order_info_load 订单表的 20260208 分区中，具体操作命令如下：
LOAD DATA LOCAL inpath '/usr/datas/order-info-0208.txt'
OVERWRITE INTO TABLE order_info_load
PARTITION(stat_date='20260505');
```



5.1.3 任务实施

```
--step06：验证上传结果
验证 Load 加载 HDFS 数据结果集，具体操作命令如下：
SELECT * FROM order_info_load;
SELECT * FROM order_info_load WHERE stat_date = '20260505';
```

```
0: jdbc:hive2://localhost:10000> SELECT * FROM order_info_load;
+---------------------------+-----------------------------+----------------------------+----------------------------+---------------------------+
| order_info_load.order_id  | order_info_load.order_status | order_info_load.create_time | order_info_load.update_time | order_info_load.stat_date |
+---------------------------+-----------------------------+----------------------------+----------------------------+---------------------------+
| 1                         | 待支付                       | 2026-02-08 12:20:11        | 2026-02-08 12:20:11        | 20260505                  |
| 2                         | 待支付                       | 2026-02-08 14:20:11        | 2026 02-08 14:20:11        | 20260505                  |
| 3                         | 待支付                       | 2026-02-08 16:20:11        | 2026-02-08 16:20:11        | 20260505                  |
+---------------------------+-----------------------------+----------------------------+----------------------------+---------------------------+
3 rows selected (3.313 seconds)
0: jdbc:hive2://localhost:10000> SELECT * FROM order_info_load WHERE stat_date = '20260505';
+---------------------------+-----------------------------+----------------------------+----------------------------+---------------------------+
| order_info_load.order_id  | order_info_load.order_status | order_info_load.create_time | order_info_load.update_time | order_info_load.stat_date |
+---------------------------+-----------------------------+----------------------------+----------------------------+---------------------------+
| 1                         | 待支付                       | 2026-02-08 12:20:11        | 2026-02-08 12:20:11        | 20260505                  |
| 2                         | 待支付                       | 2026-02-08 14:20:11        | 2026 02-08 14:20:11        | 20260505                  |
| 3                         | 待支付                       | 2026-02-08 16:20:11        | 2026-02-08 16:20:11        | 20260505                  |
+---------------------------+-----------------------------+----------------------------+----------------------------+---------------------------+
```

由执行结果可以看出，上传本地文件 order-info-0208.txt 到服务器 Hive 中的表 order_info_load，实际操作是向表中复制数据。

例题 5: 将本地文件 order-info-0209.txt 上传到服务器 Hive 中

step01：创建本地文件 /usr/datas/order-info-0209.txt

```
cd /usr/dat,as
touch order-info-0209.txt
1,待支付,2026-02-08 12:20:11,2026-02-08 12:20:11
2,已支付,2026-02-09 14:20:11,2026-02-09 14:21:11
3,已支付,2026-02-09 16:20:11,2026-02-09 16:21:11
3,待支付,2026-02-09 12:20:11,2026-02-09 12:20:11
3,待支付,2026-02-09 14:20:11,2026-02-09 14:20:11


Ctrl+C 复制

vim order-info-0209.txt
--建i打印，然后 Ctrl+Shiht+C
```

step02：查看数据库 unit5db 的位置

```
--step02：查看数据库 unit5db 的位置
--查看数据库 unit5db 在 Hive 中的位置，具体操作命令如下：
DESC DATABASE unit5db;

--有结果可知，hdfs://hadoop01:9000/user/hive/warehouse/unit5db.db,
```

```
0: jdbc:hive2://localhost:10000> DESC DATABASE unit5db;
+----------+----------+-----------------------------------------------------+-------------+-------------+-------------+
| db_name  | comment  |                      location                       | owner_name  | owner_type  | parameters  |
+----------+----------+-----------------------------------------------------+-------------+-------------+-------------+
| unit5db  |          | hdfs://hadoop01:9000/user/hive/warehouse/unit5db.db | hadoop      | USER        |             |
+----------+----------+-----------------------------------------------------+-------------+-------------+-------------+
```

5.1.3 任务实施

step03：查看数据库中的表

```
--step03：查看数据库中的表
--在数据库 unit5db 中，查看表 order_info_load 是否存在，具体操作命令如下：
SHOW TABLES;
SELECT * FROM order_info_load;
```

```
0: jdbc:hive2://localhost:10000> SHOW TABLES;
+------------------+
|     tab_name     |
+------------------+
| order_info_load  |
| pageviews        |
| students         |
+------------------+
3 rows selected (0.137 seconds)
0: jdbc:hive2://localhost:10000> SELECT * FROM order_info_load;
+---------------------------+--------------------------------+-----------------------------+-----------------------------+---------------------------+
| order_info_load.order_id  | order_info_load.order_status   | order_info_load.create_time | order_info_load.update_time | order_info_load.stat_date |
+---------------------------+--------------------------------+-----------------------------+-----------------------------+---------------------------+
| 1                         | 待支付                         | 2026-02-08 12:20:11         | 2026-02-08 12:20:11         | 20260505                  |
| 2                         | 待支付                         | 2026-02-08 14:20:11         | 2026 02-08 14:20:11         | 20260505                  |
| 3                         | 待支付                         | 2026-02-08 16:20:11         | 2026-02-08 16:20:11         | 20260505                  |
+---------------------------+--------------------------------+-----------------------------+-----------------------------+---------------------------+
```

step04：将本地文件 order-info-0209.txt 上传到服务器 Hive 中

在数据库 unit5db 中， 加载服务器本地 order-info-0209.txt 文本数据到 order_info_load 订单表的 20260506 分区中

```
--step04：将本地文件 order-info-0209.txt 上传到服务器 Hive 中
--在数据库 unit5db 中,
--加载服务器本地 order-info-0209.txt 文本数据到order_info_load 订单表的 20220608 分区中,
--具体操作命令如下：
LOAD DATA LOCAL inpath '/usr/datas/order-info-0209.txt'
OVERWRITE INTO TABLE order_info_load
PARTITION (stat_date='20260506');
```

5.1.3 任务实施

step05：验证上传结果

```
--step05：验证上传结果
--验证 Load 加载 HDFS 数据结果集，具体操作命令如下：
SELECT * FROM order_info_load WHERE stat_date='20260506';
```

```
0: jdbc:hive2://localhost:10000> SELECT * FROM order_info_load;
+-----------------------------+-------------------------------+-------------------------------+-------------------------------+-----------------------------+
| order_info_load.order_id    | order_info_load.order_status  | order_info_load.create_time   | order_info_load.update_time   | order_info_load.stat_date   |
+-----------------------------+-------------------------------+-------------------------------+-------------------------------+-----------------------------+
| 1                           | 待支付                        | 2026-02-08 12:20:11           | 2026-02-08 12:20:11           | 20260505                    |
| 2                           | 待支付                        | 2026-02-08 14:20:11           | 2026 02-08 14:20:11           | 20260505                    |
| 3                           | 待支付                        | 2026-02-08 16:20:11           | 2026-02-08 16:20:11           | 20260505                    |
+-----------------------------+-------------------------------+-------------------------------+-------------------------------+-----------------------------+
3 rows selected (0.348 seconds)
0: jdbc:hive2://localhost:10000> LOAD DATA LOCAL inpath '/usr/datas/order-info-0209.txt'
. . . . . .> OVERWRITE INTO TABLE order_info_load
. . . . . .> PARTITION (stat_date='20260506');
No rows affected (2.447 seconds)
0: jdbc:hive2://localhost:10000> SELECT * FROM order_info_load WHERE stat_date='20260506';
+-----------------------------+-------------------------------+-------------------------------+-------------------------------+-----------------------------+
| order_info_load.order_id    | order_info_load.order_status  | order_info_load.create_time   | order_info_load.update_time   | order_info_load.stat_date   |
+-----------------------------+-------------------------------+-------------------------------+-------------------------------+-----------------------------+
| 1                           | 待支付                        | 2021-02-08 12:20:11           | 2021-02-08 12:20:11           | 20260506                    |
| 2                           | 已支付                        | 2021-02-09 14:20:11           | 2021-02-09 14:21:11           | 20260506                    |
| 3                           | 已支付                        | 2021-02-09 16:20:11           | 2021-02-09 16:21:11           | 20260506                    |
| 3                           | 待支付                        | 2021-02-09 12:20:11           | 2021-02-09 12:20:11           | 20260506                    |
| 3                           | 待支付                        | 2021-02-09 14:20:11           | 2021-02-09 14:20:11           | 20260506                    |
+-----------------------------+-------------------------------+-------------------------------+-------------------------------+-----------------------------+
```

需要注意的是：

和我们熟悉的关系型数据库不一样，

- Hive 现在还不支持在INSERT 语句里面直接给出一组记录的文字形式，也就是说，Hive 并不支持 INSERT INTO….VALUES 形式的语句。
- Hive 在将本地数据加载到 Hive 表中时不做任何转换，在 Hive 3.0 版本前，加载操作只是单纯的复制操作，将数据文件移动到 Hive 对应表的位置；在 Hive3.0 版本之后，则是在内部将加载重写为 INSERT AS SELECT。

- 通过 HDFS 将数据文件加载到 Hive 表中
  - 从本地文件系统中将数据导入到 Hive 表的过程中，其实是先将数据临时复制到 HDFS 的一个目录下（典型的情况是复制到上传用户的 HDFS tmp 目录下，比如/tmp），然后再将数据从这个临时目录下移动（注意，这里说的是移动，不是复制！）到对应的 Hive 表的数据目录里面。
  - 既然如此，那么 Hive 肯定支持将数据直接从 HDFS 上的一个目录移动到相应 Hive 表的数据目录下。

例题 6：将 HDFS 上的文件 order-info-0323.txt 加载到 Hive 的 order_info_load 表中

操作过程：

```
--step01：上传本地文件 order-info-0209.txt 到 HDFS 中
--在本地已经创建本地文件 order-info-0209.txt,
--在本地/usr 下创建了一个文件夹 datas，用来存放本章节所有的本地文件,
--那么本地文件的具体路径/usr/datas/- order-info-0209.txt。
其中，本地文件 order-info-0209.txt 中共有 5 条数据，分别为：

1,待支付,2026-02-08 12:20:11,2026-02-08 12:20:11
2,已支付,2026-02-09 14:20:11,2026-02-09 14:21:11
3,已支付,2026-02-09 16:20:11,2026-02-09 16:21:11
3,待支付,2026-02-09 12:20:11,2026-02-09 12:20:11
3,待支付,2026-02-09 14:20:11,2026-02-09 14:20:11
```

step02：查看数据库 unit5db 的位置

```
--step02：查看数据库 unit5db 的位置
--查看数据库 unit5db 在 Hive 中的位置，具体操作命令如下：
DESC DATABASE unit5db;
```

```
+----------------------------------------------------+
|                      location                      |
+----------------------------------------------------+
| hdfs://hadoop01:9000/user/hive/warehouse/unit5db.db |
+----------------------------------------------------+
```

5.1.3 任务实施

step03：查看 HDFS 文件系统的目录结构

```
--step03：查看 HDFS 文件系统的目录结构
--查看 HDFS 文件系统的目录结构，以方便确定待上传数据在 HDFS 中的存放路径。
--具体操作命令如下：
[Main terminal에서]
hhdfs dfs -ls /
--其中，HDFS 文件系统的目录结构，如图 所示。共有 4 个文件目录，
--其中一个文件目录为/tmp，此目录为本章节待上传数据文件的 HDFS 的存储目录。
```

```
hadoop@hadoop01:/usr/datas$ hdfs dfs -ls /
Found 4 items
drwxrwxrwx   - hadoop supergroup          0 2025-10-20 11:19 /spark-logs
drwxr-xr-x   - hadoop supergroup          0 2026-02-05 14:12 /stu
drwxrwxrwx   - hadoop supergroup          0 2026-02-04 10:20 /tmp
drwxr-xr-x   - hadoop supergroup          0 2026-01-31 20:43 /user
```

step04：上传本地文件并查看 HDFS 文件系统的目录结构

```
--step04：上传本地文件并查看 HDFS 文件系统的目录结构
--上传本地文件 order-info-0209.txt 到 HDFS 中的/tmp 目录中，
--并再次查看 HDFS 文件系统中的/tmp 目录结构，
--以确定本地文件 order-info-0209.txt 上传到 HDFS 文件系统中的/tmp 目录成功。
--具体操作命令如下：
[Main terminal에서]
hdfs dfs -put /usr/datas/order-info-0209.txt /tmp/
--然后，再次查看 HDFS 文件系统中的/tmp 目录结构，
--以确定本地文件order-info-0209.txt 上传到HDFS 文件系统中的/tmp 目录成功。
--具体操作命令如下：
hdfs dfs -ls /tmp/*.txt
```

```
hadoop@hadoop01:/usr/datas$ hdfs dfs -ls /tmp/*.txt
-rw-r--r--   1 hadoop supergroup        260 2026-02-08 11:46 /tmp/order-info-0209.txt
```

step05：上传 HDFS 的文件到 Hive 服务器中

```
--step05：上传 HDFS 的文件到 Hive 服务器中
--将 HDFS 文件系统的目录/tmp/order-info-0209.txt 下的 order-info-0209.txt 文件
--加载order_info_load 订单表的 20260506 分区中。
--具体操作命令如下：
LOAD DATA inpath 'hdfs://hadoop01:9000/tmp/order-info-0209.txt'
OVERWRITE INTO TABLE order_info_load
PARTITION (stat_date='20260506');

--然后，再次查看 order_info_load 订单表的 20260208分区中的数据,
SELECT * FROM order_info_load WHERE stat_date='20260506';

[Main terminal에서]
/usr/datas$ hdfs dfs -ls /tmp/*.txt
```

5.1.3 任务实施

```
0: jdbc:hive2://localhost:10000> LOAD DATA inpath 'hdfs://localhost:9000/tmp/order-info-0209.txt'
. . . . . .> OVERWRITE INTO TABLE order_info_load
. . . . . .> PARTITION (stat_date='20260506');
Error: Error while compiling statement: FAILED: SemanticException Line 1:17 Invalid path ''hdfs://localhost:9000/tmp/order-info-0209.txt'' (state=42000,code=
40000)
0: jdbc:hive2://localhost:10000> LOAD DATA inpath 'hdfs://hadoop01:9000/tmp/order-info-0209.txt'
. . . . . .> OVERWRITE INTO TABLE order_info_load
. . . . . .> PARTITION (stat_date='20260506');
No rows affected (1.153 seconds)
```

```
0: jdbc:hive2://localhost:10000> SELECT * FROM order_info_load WHERE stat_date='20260506';
+---------------------------+-------------------------------+-------------------------------+-------------------------------+---------------------------+
| order_info_load.order_id  | order_info_load.order_status  | order_info_load.create_time   | order_info_load.update_time   | order_info_load.stat_date |
+---------------------------+-------------------------------+-------------------------------+-------------------------------+---------------------------+
| 1                         | 待支付                        | 2026-02-08 12:20:11           | 2026-02-08 12:20:11           | 20260506                  |
| 2                         | 已支付                        | 2026-02-09 14:20:11           | 2026-02-09 14:21:11           | 20260506                  |
| 3                         | 已支付                        | 2026-02-09 16:20:11           | 2026-02-09 16:21:11           | 20260506                  |
| 3                         | 待支付                        | 2026-02-09 12:20:11           | 2026-02-09 12:20:11           | 20260506                  |
| 3                         | 待支付                        | 2026-02-09 14:20:11           | 2026-02-09 14:20:11           | 20260506                  |
+---------------------------+-------------------------------+-------------------------------+-------------------------------+---------------------------+
```

```
hadoop@hadoop01:/usr/datas$ hdfs dfs -ls /tmp/*.txt
-rw-r--r--   1 hadoop supergroup        260 2026-02-08 11:46 /tmp/order-info-0209.txt
hadoop@hadoop01:/usr/datas$ hdfs dfs -ls /tmp/*.txt
ls: `/tmp/*.txt': No such file or directory
```

5.1.3 任务实施

例题 7：将 HDFS 上的文件 Person001.csv 加载到 Hive 的 person 表中。

step01：上传本地文件Person001.csv到HDFS中

```
--在本地已经创建本地文件Person001.csv,
--在本地/usr下已经创建了一个文件夹datas,
--用来存放本章节所有的本地文件,
--那么本地文件的具体路径/usr/datas/Person001.csv,
--其中，本地文件Person001.csv中共有80条数据,
--此处，仅显示前面5条记录，分别为：

2   SMITH   AARON
3   SMITH   ABDUL
4   SMITH   ABE
5   SMITH   ABEL
6   SMITH   ABRAHAM
```

step02：查看 HDFS 文件系统的目录结构

查看 HDFS 文件系统的目录结构，以方便确定待上传数据在 HDFS 中的存放路径。其中，HDFS 文件系统的目录结构。共有 4个文件目录，其中一个文件目录为/tmp，此目录为本章节待上传数据文件的 HDFS 的存储目录。

```
--step02：上传本地文件Person001.csv到 /usr/datas/中
[Main terminal에서]
cd /usr/datas
cp ~/Downloads/Person001.csv

hdfs dfs -ls /
```

step03：查看 HDFS 文件系统的目录结构

查看 HDFS 文件系统的目录结构，以方便确定待上传数据在 HDFS 中的存放路径。其中，HDFS 文件系统的目录结构。共有 4个文件目录，其中一个文件目录为/tmp，此目录为本章节待上传数据文件的 HDFS 的存储目录。

```
--step03：上传本地文件并查看 HDFS 文件系统的目录结构
--上传本地文件 Person001.csv 到 HDFS 中的/tmp 目录中，
--并再次查看 HDFS文件系统中的/tmp 目录结构，
--以确定本地文件 Person001.csv 上传到 HDFS 文件系统中的/tmp 目录成功。
hdfs dfs -put /usr/datas/Person001.csv /tmp

--然后，再次查看 HDFS 文件系统中的/tmp 目录结构，
--以确定本地文件Person001.csv 上传到 HDFS 文件系统中的/tmp 目录成功。
hdfs dfs -ls /tmp/Person001.*
--其中，HDFS 文件系统的目录结构。
--共有 3 个文件目录，其中一个文件目录为/tmp/Person001.csv，
--本地文件 Person001.csv 上传到 HDFS指定的目录/tmp 成功。
```

step04：创建数据库中的person表

```
--step04：创建数据库中的person表
--在数据库unit5db中，创建person表，具体操作命令如下：
CREATE TABLE person (
perid int,
lastname string,
firstname string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

--然后，在数据库 unit5db 中，查看 person 表是否创建成功。
SHOW tables;
```

step04：创建数据库中的person表

```
0: jdbc:hive2://localhost:10000> show tables;
+------------------+
|     tab_name     |
+------------------+
| order_info_load  |
| pageviews        |
| students         |
+------------------+
3 rows selected (0.25 seconds)
0: jdbc:hive2://localhost:10000> CREATE TABLE person (
. . . . . . . . . . . . . . . .> perid int,
. . . . . . . . . . . . . . . .> lastname string,
. . . . . . . . . . . . . . . .> firstname string
. . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . .> ROW FORMAT DELIMITED
. . . . . . . . . . . . . . . .> FIELDS TERMINATED BY ',';
No rows affected (3.943 seconds)
0: jdbc:hive2://localhost:10000> show tables;
+------------------+
|     tab_name     |
+------------------+
| order_info_load  |
| pageviews        |
| person           |
| students         |
+------------------+
4 rows selected (0.254 seconds)
```

step05：上传 HDFS 的文件到 Hive 服务器中

将HDFS 文件系统的目录/tmp/Person001.csv 下的 Person001.csv 文件加载到person 表中。

```
--step05：上传 HDFS 的文件到 Hive 服务器中
--将 HDFS 文件系统的目录/tmp/Person001.csv 下的 Person001.csv 文件加载到person 表中。
LOAD DATA inpath 'hdfs://hadoop01:9000/tmp/Person001.csv'
OVERWRITE INTO TABLE person;

SELECT * FROM person;
```

三、通过查询加载数据

Hive 也可以使用 INSERT 子句将查询结果插入到表中。

在利用这种方式进行加载数据时，可以一次性加载数据到一张表中，

同时，也可以一次性加载数据到多个表中。

语法格式：
标准格式：

```
INSERT  OVERWRITE  TABLE  tablename1
[PARTITION (partcol1=val1, partcol2=val2 ...)
[IF NOT EXISTS]]   select_statement1   FROM    from_statement;

--多个表插入格式：
FROM employees
INSERT OVERWRITE TABLE emp_partitioned
    PARTITION (dept='HR')
    SELECT name, salary WHERE dept = 'HR'
INSERT OVERWRITE TABLE emp_partitioned
    PARTITION (dept='IT')
    SELECT name, salary WHERE dept = 'IT'
INSERT INTO TABLE emp_backup
    SELECT *;
```

5.1.3 任务实施

例题 8：使用 OVERWRITE 方式将 person 表查询相关数据后加载到 Hive 的personhub 表中。

操作过程：

step01： 创建数据库中的 personhub 表

```
CREATE TABLE personhub (persid int);
```

然后，在数据库 unit5db 中，查看 personhub 表是否创建成功。

```
SHOW TABLES;
```

```
0: jdbc:hive2://localhost:10000> show tables;
+-----------------+
|    tab_name     |
+-----------------+
| order_info_load |
| pageviews       |
| person          |
| personhub       |
| students        |
+-----------------+
```

step02： 利用 overwrite 方式将 person 表的数据加载到 personhub 表中 在数据库unit5db 中

```
--step02： 利用 overwrite 方式将 person 表的数据加载到 personhub 表中 在数据库unit5db 中,
--利用 overwrite 方式将person 表的数据加载到personhub表中, 覆盖 personhub 表中原有数据
INSERT OVERWRITE TABLE personhub
SELECT DISTINCT perid FROM person; --이것도 시간 많이 걸림(78.912초)
```

Last Max 25 Closed Queries

User Name	Query	Execution Engine	State	Opened (s)	Closed Timestamp	Latency (s)	Drilldown Link
anonymous	INSERT OVERWRITE TABLE personhub SELECT DISTINCT perid FROM person	mr	FINISHED	78	Sun Feb 08 17:26:13 KST 2026	78	Drilldown

step03： 验证 personhub 表中的数据

```
--step03： 验证 personhub 表中的数据
--再次查看 personhub 表中的数据

SELECT * from personhub;
```

```
0: jdbc:hive2://localhost:10000> SELECT * from personhub;
+--------------------+
| personhub.persid   |
+--------------------+
| 2                  |
| 3                  |
| 4                  |
```



```
| 79                 |
| 80                 |
| 81                 |
+--------------------+
80 rows selected (0.287 seconds)
```

5.1.3 任务实施

例题 9：使用 INTO 方式将 person 表查询相关数据后加载到 Hive 的 personhub 表中。

操作过程：查看数据库中的personhub表的数据。其中，该表中共有 80 条数据。

```
--step02：  利用 into 方式将 person 表的数据加载到 personhub 表中在数据库 unit5db 中,
--利用 into 方式将 person 表的数据加载到 personhub 表中，追加新数据到 personhub 表中原有数据后,
INSERT INTO TABLE personhub
SELECT DISTINCT perid FROM person; --이것도 시간 많이 걸림(65.947초)


--step03：  验证 personhub 表中的数据再次查看 personhub 表中的数据,
--由结果可知，整个表中共有 160 条最新的数据，加载文件成功。
SELECT * FROM personhub;
```

5.1.3 任务实施

例题 10：将 wyp 表中的数据依据查询要求查询后分别加载到 test 表、test3 表中。

操作过程：

```
--step01：   创建数据库中的 wyp 表
--在数据库 unit5db 中，创建 wyp 表
CREATE TABLE wyp (id int, name string, age int, tel string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ' '
STORED AS TEXTFILE;

--然后，在数据库unit5db中，查看wyp表是否创建成功。
--创建成功后，利用前面所学知识向wyp表中加载wyp.txt和add.txt文件中的数据，
-- 最后，查看wyp表中的数据，共有7条数据
--step02：创建wyp.txt使用如下：
[Main terminal에서]
cd /usr/datas
vim wyp.txt

5 wyp1 23 232323232323
6 wyp2 34 32454567678825
7 wyp3 36 23243243545656
8 wyp4 26 24324354667572
1 wyp 25 13188888888
2 test 30 13388888888
3 zs 34 899314121
```

5.1.3 任务实施

```
--step03：上传本地文件并查看 HDFS 文件系统的目录结构
--上传本地文件 wyp.txt 到 HDFS 中的/tmp 目录中，
--并再次查看 HDFS文件系统中的/tmp 目录结构，
--以确定本地文件 wyp.txt 上传到 HDFS 文件系统中的/tmp 目录成功。
hdfs dfs -put /usr/datas/wyp.txt /tmp

hdfs dfs -ls /tmp/wyp.*

--step04：上传 HDFS 的文件到 Hive 服务器中
--将 HDFS 文件系统的目录/tmp/wyp.txt 下的 wyp.txt 文件加载到wyp 表中。
LOAD DATA inpath 'hdfs://hadoop01:9000/tmp/wyp.txt'
OVERWRITE INTO TABLE wyp;

SELECT * FROM wyp;
```

```
0: jdbc:hive2://localhost:10000> select * from wyp;
+---------+-----------+----------+-------------------+
| wyp.id  | wyp.name  | wyp.age  |      wyp.tel      |
+---------+-----------+----------+-------------------+
| 5       | wyp1      | 23       | 232323232323      |
| 6       | wyp2      | 34       | 32454567678825    |
| 7       | wyp3      | 36       | 232432435456 56   |
| 8       | wyp4      | 26       | 243243546675 72   |
| 1       | wyp       | 25       | 13188888888       |
| 2       | test      | 30       | 13388888888       |
| 3       | zs        | 34       | 899314121         |
+---------+-----------+----------+-------------------+
7 rows selected (0.366 seconds)
```

```
--step04 创建数据库中的test表
--在数据库unit5db中，创建test表
CREATE TABLE test(id int, name string, tel string)
PARTITIONED BY (age int)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ' '
STORED AS TEXTFILE;
--然后，在数据库unit5db中，查看test表是否创建成功。
--创建成功后，查看test表中的数据，表中无数据。
SELECT * FROM test;

--step05： 创建数据库中的 test3 表
--在数据库 unit5db 中，创建 test3 表，具体操作命令如下：
CREATE TABLE test3(id int, name string);
--然后，在数据库unit5db中，查看test3表是否创建成功。创建成功后，
--查看test3表中的数据，表中无数据。
SELECT * FROM test3;
```

5.1.3 任务实施

```
--step06：按照查询要求加载 wyp 表中的数据分别到 test 表和 test3 表中 在数据库unit5db 中，
--按照查询要求加载wyp 表中的数据分别到test 表和test3表中，具体操作命令如下：
SET hive.exec.dynamic.partition.mode=nonstrict;

--在此处，test 表里面用 age 作为了分区字段，
--为了方便将 wyp 表中的查询结果插入到 test 表中，
--因此，需要设置 hive.exec.dynamic.partition.mode 的值为 nonstrict。
FROM wyp
INSERT INTO table test
partition(age)
SELECT id, name, tel, age
INSERT into table test3
SELECT id, name where age>25;     --시간 걸림 (65.468초)
```

5.1.3 任务实施

```
--step07：  分别验证 test 表和 test3 表中的数据
--再次查看 test 表和 test3 表中的数据，如图 5-40 和图 5-41 所示。
--由结果可知，test 表和 test3 表分别加载数据成功。
--具体操作命令如下：
SELECT * FROM test;
SELECT * FROM test3;
```

```
0: jdbc:hive2://localhost:10000> SELECT * FROM test;
+----------+------------+-----------------+-----------+
| test.id  | test.name  |    test.tel     | test.age  |
+----------+------------+-----------------+-----------+
| 5        | wyp1       | 232323232323    | 23        |
| 1        | wyp        | 13188888888     | 25        |
| 8        | wyp4       | 24324354667572  | 26        |
| 2        | test       | 13388888888     | 30        |
| 6        | wyp2       | 32454567678825  | 34        |
| 3        | zs         | 899314121       | 34        |
| 7        | wyp3       | 23243243545656  | 36        |
+----------+------------+-----------------+-----------+
```

```
0: jdbc:hive2://localhost:10000> SELECT * FROM test3;
+-----------+-------------+
| test3.id  | test3.name  |
+-----------+-------------+
| 6         | wyp2        |
| 7         | wyp3        |
| 8         | wyp4        |
| 2         | test        |
| 3         | zs          |
+-----------+-------------+
5 rows selected (0.269 seconds)
```

四、查询语句中创建表并加载数据

除了上述三种加载数据的方式之外，可以使用 CREATE TABLE AS SELECT（CTAS）命令，

该查询语句中创建表并加载数据，是将查询的结果保存到一张表中，

然后创建表最后导入数据，这种形式一般称为 AS Select 语句。

目的在于利用其结果集和查询输出模式来创建一个内部表。

这种方式读者可以在一个语句中完成创建表并将查询结果加载到这个表的操作。

语法格式：

```
CREATE table [IF NOT EXISTS] tablename
AS SELECT select_statement1 FROM from_statement;
```

5.1.3 任务实施

例题 11：将 wyp 表中的数据依据查询要求查询后加载到 test4 表中。

操作过程：

```
--step01：   查看数据库中的 wyp 表的数据 其中，由图 5-37 所示，该表中共有 7 条数据。
SELECT * from wyp;


--step02：   创建数据库中的表 test4
--在数据库 unit5db 中，创建数据库中的表 test4，
--并在创建的同时，将表 wyp中的数据加载到表 test4 中：
CREATE TABLE test4 AS SELECT id, tel
FROM wyp;                          --시간걸림 (19.4초)


--step03：   验证 test4 表中的数据
--查看 test4 表中的数据。
--由结果可知，整个表中共有 7 条最新的数据，加载数据成功。
SELECT * FROM test4;
```

5.1.3 任务实施

例题 12： 将 HDFS 上的 order_info_load 表中的数据加载到 Hive order_info_as_select_load 表中。

操作过程：

step01： 查看数据库中的 order_info_load 表的数据

其中的数据，由图 5-43 所示，该表中共有 13 条数据。具体操作命令如下：

```
hive> SELECT * FROM order_info_load;
```

step02： 创建数据库中的表 order_info_as_select_load

在数据库 unit5db 中，创建数据库中的表 order_info_as_select_load，并在创建的同时，将表 order_info_load 中的数据加载到表 order_info_as_select_load 中，具体操作命令如下：

```
hive> CREATE TABLE order_info_as_select_load as select * FROM order_info_load;
```

5.1.3 任务实施

例题 12： 将 HDFS 上的 order_info_load 表中的数据加载到 Hive order_info_as_select_load 表中。

操作过程：

```
--例题 12 ： 将 HDFS 上的 order_info_load 表中的数据加载到 Hive order_info_as_select_load 表中。
--step01： 查看数据库中的 order_info_load 表的数据
SELECT * FROM order_info_load;

--step02： 创建数据库中的表 order_info_as_select_load
--在数据库 unit5db 中，创建数据库中的表 order_info_as_select_load，
--并在创建的同时，将表 order_info_load 中的数据加载到表 order_info_as_select_load 中
CREATE TABLE order_info_as_select_load
AS SELECT * FROM order_info_load;

--step03： 验证 order_info_as_select_load 表中的数据
SELECT * FROM order_info_as_select_load;
```

五、import 导入 Hive 表数据（内部表操作）

这种加载数据的方式一般先用 Export 将数据导出后，再将数据使用 Import导入，利用 Export 和 Import 主要用于两个 Hadoop 平台集群之间 Hive 表的迁移。

语法格式：

```
import TABLE  tablename  FROM  'filepath'
```

5.1.3 任务实施

例题 13：将 HDFS 上的已有文件加载到 Hive 的 order_info_import_load 表中。

操作过程：

```
--step01：  查看 HDFS 文件系统中/tmp 的目录结构。导出数据之前，
--查看 HDFS 文件系统中/tmp 的目录结构，
--其中没有/tmp/order_info_load_out4 目录。
--具体操作命令如下：[root@node2 ~]# hadoop fs  -ls /tmp/
hdfs dfs -ls /tmp/

--step02：  将文件导出到 HDFS 相应目录中。
--首先，将文件导出到相应的目录中，
--本题目中所使用目录为 HDFS 上的文件系统/tmp/order_info_load_out4 目录。
Export table order_info_load to '/tmp/order_info_load_out4';
--此处，在导入之前，首先将需要导入的资源准备充分，
--此处为准备资源的部分。
--只是此次的资源，是通过导出得来的，而不是本来就准备好的，
--这样做的目的是为了读者可以更好的理解 Hive 生态系统中导入和导出的实际应用。

--step03：  再次查看 HDFS 文件系统中/tmp 的目录结构
hdfs dfs -ls /tmp/
--由结果可知，其中/tmp/order_info_load_out4 已经存在。
```

5.1.3 任务实施

```
--step04：  创建表 order_info_import_load
--在数据库 unit5db 中，创建所需表 order_info_import_load，该表为目标表。
CREATE TABLE order_info_import_load LIKE order_info_load;
--由结果可知，其中 order_info_import_load 表中无数据。
SELECT * FROM order_info_import_load;

--step05：  验证是否导入成功。
--其中，  将导出操作所得数据 /tmp/order_info_load_out4
--导 入 到 表order_info_import_load 中
IMPORT TABLE order_info_import_load FROM '/tmp/order_info_load_out4';
--再次查看表 order_info_import_load 中的数据
--共有 13 条数据，导入操作成功
SELECT * FROM order_info_import_load;
```

5.1.3 任务实施

```
0: jdbc:hive2://localhost:10000> SELECT * FROM order_info_import_load;
+-----------------------------------+-------------------------------------+-----------------------------------+-----------------------------------+-----------------------------------+
| order_info_import_load.order_id   | order_info_import_load.order_status | order_info_import_load.create_time | order_info_import_load.update_time | order_info_import_load.stat_date  |
+-----------------------------------+-------------------------------------+-----------------------------------+-----------------------------------+-----------------------------------+
| 1                                 | 待支付                              | 2026-02-08 12:20:11               | 2026-02-08 12:20:11               | 20260505                          |
| 2                                 | 待支付                              | 2026-02-08 14:20:11               | 2026 02-08 14:20:11               | 20260505                          |
| 3                                 | 待支付                              | 2026-02-08 16:20:11               | 2026-02-08 16:20:11               | 20260505                          |
| 1                                 | 待支付                              | 2026-02-08 12:20:11               | 2026-02-08 12:20:11               | 20260506                          |
| 2                                 | 已支付                              | 2026-02-09 14:20:11               | 2026-02-09 14:21:11               | 20260506                          |
| 3                                 | 已支付                              | 2026-02-09 16:20:11               | 2026-02-09 16:21:11               | 20260506                          |
| 3                                 | 待支付                              | 2026-02-09 12:20:11               | 2026-02-09 12:20:11               | 20260506                          |
| 3                                 | 待支付                              | 2026-02-09 14:20:11               | 2026-02-09 14:20:11               | 20260506                          |
+-----------------------------------+-------------------------------------+-----------------------------------+-----------------------------------+-----------------------------------+
```

5.1.3 小结论

Hive生态系统中向表中装载数据的方式主要有

① 直接向表中加载数据、

② 通过 Load 加载数据、

③ 通过查询加载数据、

④ 查询语句中创建表并加载数据、

⑤ import 导入 Hive 表数据

等五种。