目 录 / CONTENTS

1 任务3-1 认识数据类型

2 任务3-2 认识运算符

3 任务3-3 认识Hive命令

4 任务3-4 存储数据

任务 3-2 认识运算符

3.2.1 任务描述

在Hive的基本操作中，经常需要查询出数据表中满足条件的数据，往往条件不是单一的，需要配合相应的运算符来进行实现。

本任务将学会Hive的不同运算符，

包括关系运算符、算术运算符、逻辑运算符、复杂运算符等。

为实现数据表的查询打下基础。







3.2.2 任务准备知识

Hive 运算符与其他高级语言中运算符类似，由变量、常量等连接表达式中的各个操作数，其作用是指明对操作数进行运算。

Hive中常见的运算符有

- 关系运算符、
- 算术运算符、
- 逻辑运算符、
- 复杂运算符。

1．关系运算符(Comparison Operators)

关系运算符又称比较运算符，用于比较两个表达式的值，其运算结果为逻辑值，可以为True、False或NULL。

（1）=运算符

=运算符用于比较表达式的两边是否相等，也可以对字符串进行比较：

```
select 3.14=3.142,2.12=2.120,'a'='a','apple'='pear';
```

```
0: jdbc:hive2://node2:10000/> select 3.14=3.142,2.12=2.120,'a'='a','apple'='pear';
+--------+-------+-------+--------+
|  _c0   |  _c1  |  _c2  |  _c3   |
+--------+-------+-------+--------+
| false  | true  | true  | false  |
+--------+-------+-------+--------+
1 row selected (0.314 seconds)
0: jdbc:hive2://node2:10000/>
```

3.2.3 任务实施

```
Terminal File Edit View Search Terminal Help
hadoop@hadoop01:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.36.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]
0: jdbc:hive2://localhost:10000> select 3.14=3.142, 2.12=2.120, 'a'='a','apple'='pear';
+--------+-------+-------+--------+
|  _c0   |  _c1  |  _c2  |  _c3   |
+--------+-------+-------+--------+
| false  | true  | true  | false  |
+--------+-------+-------+--------+
0: jdbc:hive2://localhost:10000>
```

1．关系运算符(Comparison Operators)

（2）<>运算符

与=运算符相反的是!=运算符，它用来检测表达式的两边是否相等，如果不相等返回真值，相等则返回假值。示例如下：

```
select 3<>3,5<>6,'5a'<>'5c';
```

```
0: jdbc:hive2://node2:10000/> select 3<>3,5<>6,'5a'<>'5c';
+--------+-------+-------+
|  _c0   |  _c1  |  _c2  |
+--------+-------+-------+
| false  | true  | true  |
+--------+-------+-------+
1 row selected (0.202 seconds)
0: jdbc:hive2://node2:10000/>
```

3.2.3 任务实施

```
Terminal File Edit View Search Terminal Help
hadoop@hadoop01:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.17
.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib
/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explana
tion.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFacto
ry]
0: jdbc:hive2://localhost:10000> select 3<>3, 5<>6, '5a'<>'5c';
+--------+-------+-------+
|  _c0   |  _c1  |  _c2  |
+--------+-------+-------+
| false  | true  | true  |
+--------+-------+-------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D , Ctrl+L
!history	!set	!set verbose false

1．关系运算符(Comparison Operators)

（3）>、>=、<、<=运算符

与=运算符相反的是!=运算符，它用来检测表达式的两边是否相等，如果不相等返回真值，相等则返回假值。示例如下：

select 3<3,5>6, 'a'>='c','A'<='B';

```
+--------+--------+--------+--------+
0: jdbc:hive2://localhost:10000> select 3<3,5>6, 'a'>='c','A'<='B';
+--------+--------+--------+--------+
|  _c0   |  _c1   |  _c2   |  _c3   |
+--------+--------+--------+--------+
| false  | false  | false  | true   |
+--------+--------+--------+--------+
0: jdbc:hive2://localhost:10000>
```

1．关系运算符(Comparison Operators)

（4）[not] null运算符

判断表达式的结果是否为**null**,为null则为真，否则相反，如果是**not null**则情况正好和**null**相反。举例如下：

```
select 1 from dual where 'hello' is null;
```

```
0: jdbc:hive2://node2:10000/> select 1 from dual where 'hello' is null;
+------+
| _c0  |
+------+
+------+
No rows selected (0.469 seconds)
0: jdbc:hive2://node2:10000/>
```

3.2.3 任务实施

```
Terminal File Edit View Search Terminal Help
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000> select  1 from dual where 'hello' is null;
+------+
| _c0  |
+------+
+------+
0: jdbc:hive2://localhost:10000> select 1 from dual where 'hello' is not null;
+------+
| _c0  |
+------+
| 1    |
+------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D, Ctrl+L
!history	!set	!set verbose false

1．关系运算符(Comparison Operators)

（5）[not] like运算符

like运算符是字符串模式匹配，可以包含字符或数字：%代表零或多个字符，_代表一个字符。若与其匹配，则返回true，否则返回false。举例：

```
select 1 from dual where 'hello' like 'he%';
```

```
0: jdbc:hive2://node2:10000/> select 1 from dual where 'hello' like 'he%';
+------+
| _c0  |
+------+
| 1    |
+------+
1 row selected (0.182 seconds)
0: jdbc:hive2://node2:10000/>
```

3.2.3 任务实施

```
Terminal File Edit View Search Terminal Help
0: jdbc:hive2://localhost:10000> select 1 from dual  where 'hello' like 'he%';
+------+
| _c0  |
+------+
| 1    |
+------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D, Ctrl+L
!history	!set	!set verbose false

1．关系运算符(Comparison Operators)

（6）rlike运算符

确定字符串是否匹配正则表达式，是**REGEXP_LIKE**的同义词。例如：

判断起始为h，结尾为o的字符串匹配：

```
select 1 from dual where 'hello' rlike '^h.*o$';
```

```
0: jdbc:hive2://node2:10000/> select 1 from dual where 'hello' rlike '^h.*o$';
+------+
| _c0  |
+------+
| 1    |
+------+
1 row selected (0.223 seconds)
0: jdbc:hive2://node2:10000/>
```

符号	含义	说明
^	**字符串开始**	表示字符串的起始位置
h	**字符 'h'**	第一个字符必须是 'h'
.*	**任意字符 0个以上**	. = 任意一个字符，* = 重复0次或多次
o	**字符 'o'**	最后一个字符必须是 'o'
$	**字符串结束**	表示字符串的结束位置

整体含义:

- 字符串以 **'h' 开头**
- 中间有 **任意字符0个或多个**
- 以 **'o' 结尾**

3.2.3 任务实施

```
Terminal File Edit View Search Terminal Help
0: jdbc:hive2://localhost:10000> select 1 from dual where 'hello' rlike '^h.*o$';
+------+
| _c0  |
+------+
| 1    |
+------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

2．算术运算符(Arithmetic Operators)

算术运算符是Hive中最基本的运算符，用于各类数值运算，返回数值类型。

运算符	支持数据类型	示例	示例描述
+	所有数字数据类型	A+B	A和B相加
-	所有数字数据类型	A-B	A减去B
*	所有数字数据类型	A*B	A和B相乘
/	所有数字数据类型	A/B	A除以B，返回商
%	所有数字数据类型	A%B	A对B取余，返回余数

3．逻辑运算符(Logical Operators)

逻辑运算符用于对某个条件进行测试，运算结果为TRUE或FALSE。

逻辑运算符	支持数据类型	示例	示例描述
AND	Boolean数据类型	A	逻辑与，表达式A与表达式B必须都为true，表达式结果才为true
OR	Boolean数据类型	A	逻辑或，表达式A或表达式B至少一个为true，表达式结果才为true
NOT	Boolean数据类型	NOT	逻辑非

3．逻辑运算符(Logical Operators)

（1）and运算符

and运算符用于测试两个或更多的值（或表达式）的有效性，如果所有表达式的值为真，并且不是NULL，结果为真值，否则为假值。例如:

```
select 1=1 and 8>10,'a'='a' and 'b'<'d';
```

```
0: jdbc:hive2://node2:10000/> select 1=1 and 8>10,'a'='a' and 'b'<'d';
+--------+-------+
|  _c0   |  _c1  |
+--------+-------+
| false  | true  |
+--------+-------+
1 row selected (0.481 seconds)
0: jdbc:hive2://node2:10000/>
```

3.2.3 任务实施

```
Terminal File Edit View Search Terminal Help
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000> select 1=1 and 8>10, 'a'='a' and 'b' <'d';
+--------+-------+
|  _c0   |  _c1  |
+--------+-------+
| false  | true  |
+--------+-------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

3．逻辑运算符(Logical Operators)

（2）or运算符

如果表达式的值有一个为真并且不是NULL值，结果为真值，若表达式的值全为假则结果为假值。例如：

```
select 1=1 or 8>10,'a'='b' or 'b'>'d';
```

```
0: jdbc:hive2://node2:10000/> select 1=1 or 8>10,'a'='b' or 'b'>'d';
+-------+--------+
|  _c0  |  _c1   |
+-------+--------+
| true  | false  |
+-------+--------+
1 row selected (0.084 seconds)
0: jdbc:hive2://node2:10000/>
```

3.2.3 任务实施

```
0: jdbc:hive2://localhost:10000> select 1=1 or 8>10, 'a'='a' or 'b' <'d';
+-------+-------+
|  _c0  |  _c1  |
+-------+-------+
| true  | true  |
+-------+-------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

3．逻辑运算符(Logical Operators)

（3）not运算符

逻辑运算符中最简单的是not运算符，它对跟在其后面的逻辑测试判断取反，把真值变为假值，假值变为真值。例如：

```
select not 8>9,not 1=1,not 10>6;
```

```
0: jdbc:hive2://node2:10000/> select not 8>9,not 1=1,not 10>6;
+-------+--------+--------+
|  _c0  |  _c1   |  _c2   |
+-------+--------+--------+
| true  | false  | false  |
+-------+--------+--------+
1 row selected (0.091 seconds)
0: jdbc:hive2://node2:10000/>
```

3.2.3 任务实施

```
0: jdbc:hive2://localhost:10000> select not 8>9, not 1=1, not 10>6;
+-------+--------+--------+
|  _c0  |  _c1   |  _c2   |
+-------+--------+--------+
| true  | false  | false  |
+-------+--------+--------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

4．复杂运算符(Complex Type Logical Operators)

复杂运算符提供一个表达式来引用复杂类型的元素

复杂运算符	支持数据类型	描述
A[n]	ARRAY	返回数字A的第n个元素的值，第一个元素的索引为0
M[key]	MAP	返回M指定key的value值
S.x	STRUCT	返回S的x字段值

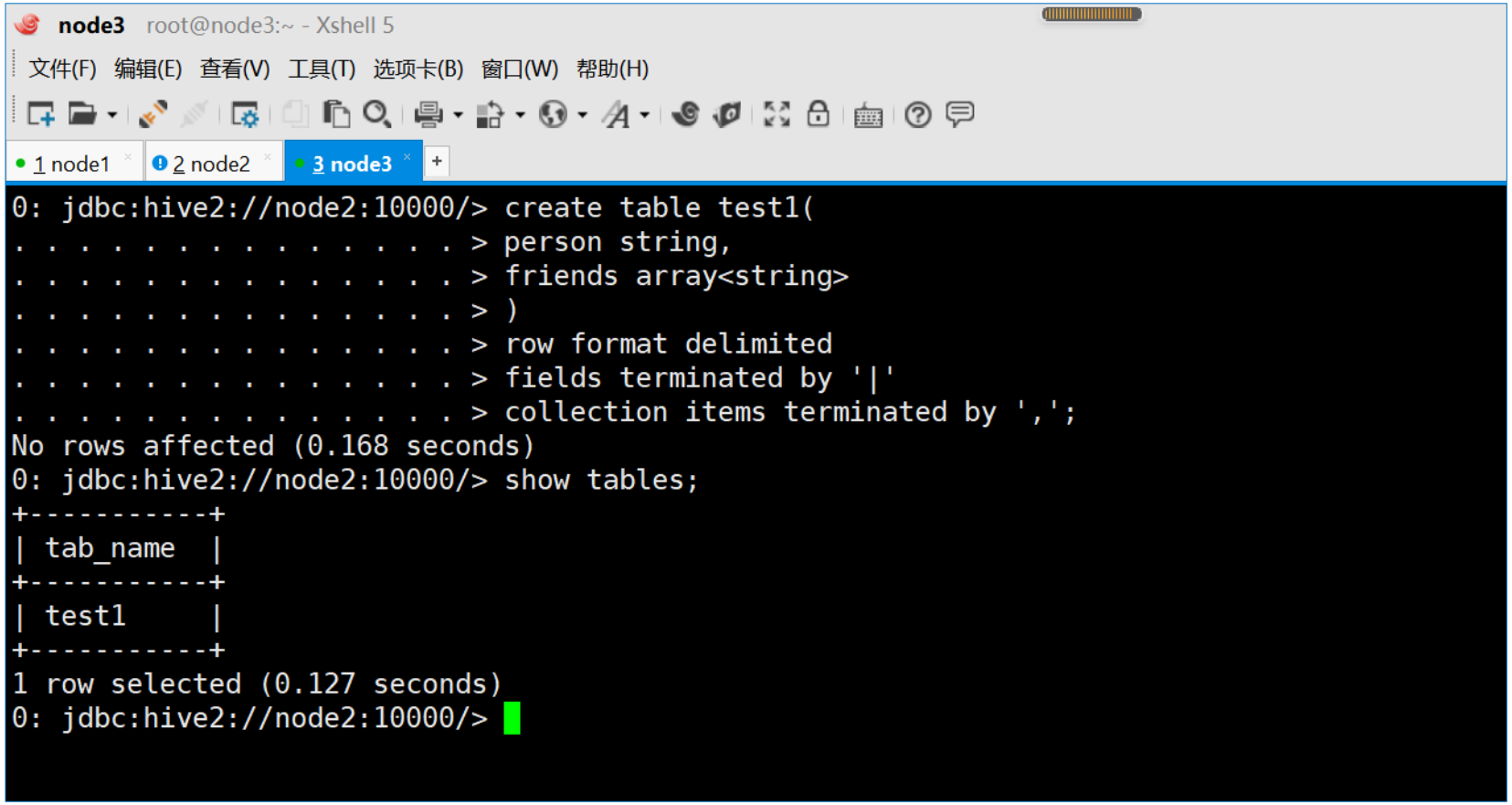
4．复杂运算符(Complex Type Logical Operators)

（1）ARRAY运算符

创建名为test1的表，只有person和friends两个字段，person为字符串类型，friends是array数组类型，通过文本文件导入数据时，person和friends之间的分隔符是竖线，friends内部的多元素之间的分隔符是逗号。

4．复杂运算符(Complex Type Logical Operators)

（1）ARRAY运算符



3.2.3 任务实施

```
0: jdbc:hive2://localhost:10000> create table  test1(
0: jdbc:hive2://localhost:10000> person string,
0: jdbc:hive2://localhost:10000> friends array<string>
0: jdbc:hive2://localhost:10000> )
0: jdbc:hive2://localhost:10000> row format delimited
0: jdbc:hive2://localhost:10000> fields terminated by '|'
0: jdbc:hive2://localhost:10000> collection items terminated by ',';
0: jdbc:hive2://localhost:10000> show tables;
+------------+
|  tab_name  |
+------------+
| dual       |
| employees  |
| test1      |
+------------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

4．复杂运算符(Complex Type Logical Operators)

（1）ARRAY运算符

创建文本文件test1.txt，内容如下：可见只有两条记录，第一条person字段值为Lucy, friends字段里面有三个元素，用逗号分隔

```
[root@node2 data]# cat test1.txt
Lucy|Lucy_friend_0,Lucy_friend_1,Lucy_friend_2
Tom|Tom_friend_0,Tom_friend_1,Tom_friend_2,Tom_friend_3,Tom_friend_4
Sophie|Sophie_friend_0,Sophie_friend_1
[root@node2 data]#
```

3.2.3 任务实施(实践)

步骤1. 数据文件准备

test1.txt

```
Lucy|Lucy_friend_0,Lucy_friend_1,Lucy_friend_2
Tom|Tom_friend_0,Tom_friend_1,Tom_friend_2,Tom_friend_3,Tom_friend_4
Sophie|Sophie_friend_0,Sophie_friend_1
```

重要提示：数组元素之间不能有空格，否则 Hive 无法正确识别。

步骤2. 上传文件到 HDFS# 创建 HDFS 目录

```
hdfs dfs -mkdir -p /user/hive/warehouse/test_data/

# 上传文件
hdfs dfs -put test1.txt /user/hive/warehouse/test_data/test1.txt

# 确认上传
hdfs dfs -ls /user/hive/warehouse/test_data/
hdfs dfs -cat /user/hive/warehouse/test_data/test1.txt
```

3.2.3 任务实施(实践)

步骤3[hive]. 加载数据到 Hive 表

```
-- 从 HDFS 路径加载数据
LOAD DATA INPATH '/user/hive/warehouse/test_data/test1.txt'
INTO TABLE test1;
```

步骤43[hive]. 验证数据加载

```
0: jdbc:hive2://localhost:10000> select * from test1;
+---------------+-----------------------------------------------------+
| test1.person  |                   test1.friends                     |
+---------------+-----------------------------------------------------+
| Lucy          | ["Lucy_friend_0","Lucy_friend_1","Lucy_friend_2"]   |
| Tom           | ["Tom_friend_0"," Tom_friend_1","Tom_friend_3","Tom_friend_4"
] |
| Sophie        | ["Sophie_friend_0"," Sophie_friend_1"]              |
+---------------+-----------------------------------------------------+
0: jdbc:hive2://localhost:10000>
```

4．复杂运算符(Complex Type Logical Operators)

（1）ARRAY运算符

加载test1.txt数据到test表后，查询friends中某个元素的SQL语句，例如：查询每个人的第一个朋友

```
select person, friends[0] from test1;
```

```
0: jdbc:hive2://node2:10000/> select person, friends[0] from test1;
+---------+-----------------+
| person  |       _c1       |
+---------+-----------------+
| Lucy    | Lucy_friend_0   |
| Tom     | Tom_friend_0    |
| Sophie  | Sophie_friend_0 |
+---------+-----------------+
3 rows selected (0.183 seconds)
0: jdbc:hive2://node2:10000/>
```

3.2.3 任务实施

```
0: jdbc:hive2://localhost:10000> select person, friends[0] from test1;
+---------+-------------------+
| person  |        _c1        |
+---------+-------------------+
| Lucy    | Lucy_friend_0     |
| Tom     | Tom_friend_0      |
| Sophie  | Sophie_friend_0   |
+---------+-------------------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

4．复杂运算符(Complex Type Logical Operators)

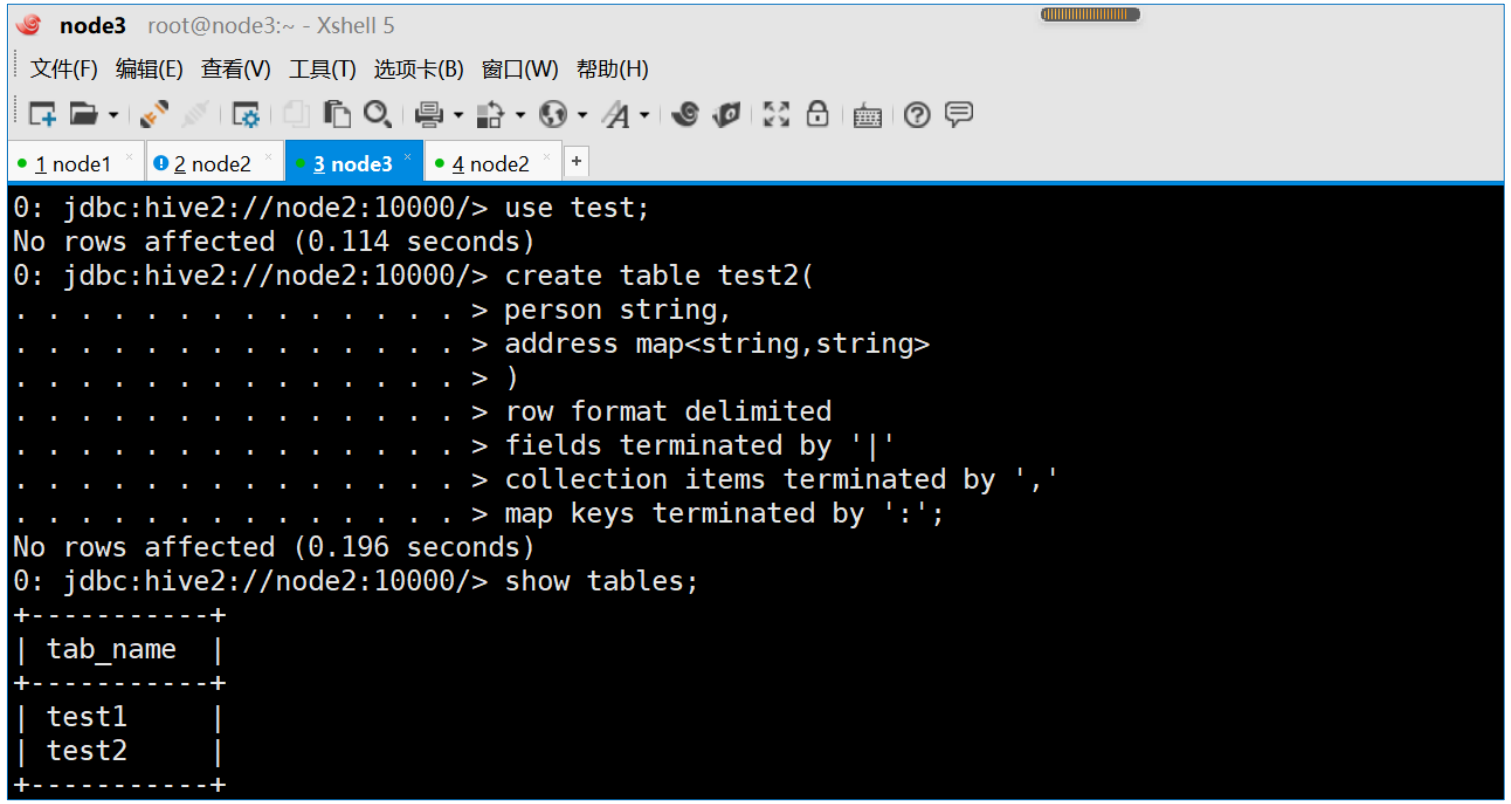
（2）MAP运算符

创建名为test2的表，只有person和address两个字段，person为字符串类型，address为MAP类型，通过文本导入数据时，对分隔符的定义如下：

- Person和address之间的分隔符是竖线;
- address内部多个键值对，它们的分隔符是逗号;
- 每个键值对的键和值的分隔符是冒号。

4．复杂运算符(Complex Type Logical Operators)

（2）MAP运算符



3.2.3 任务实施

```
0: jdbc:hive2://localhost:10000> CREATE TABLE test2 (
0: jdbc:hive2://localhost:10000>     person STRING,                      -- 字符串类型
0: jdbc:hive2://localhost:10000>     address MAP<STRING, STRING>         -- MAP类，键值都是字符串
0: jdbc:hive2://localhost:10000> )
0: jdbc:hive2://localhost:10000> ROW FORMAT DELIMITED
0: jdbc:hive2://localhost:10000> FIELDS TERMINATED BY '|'               -- Person和address之间的分隔符：竖线
0: jdbc:hive2://localhost:10000> COLLECTION ITEMS TERMINATED BY ','     -- address内部多个键值对的分隔符：逗号
0: jdbc:hive2://localhost:10000> MAP KEYS TERMINATED BY ':'             -- 每个键键和值的分隔符：冒号
0: jdbc:hive2://localhost:10000> STORED AS TEXTFILE;                    -- 文本文件格式
0: jdbc:hive2://localhost:10000> show tables;
+------------+
|  tab_name  |
+------------+
| dual       |
| employees  |
| test1      |
| test2      |
+------------+
0: jdbc:hive2://localhost:10000>
```

4．复杂运算符(Complex Type Logical Operators)

（2）MAP运算符

创建文本文件test2.txt，用三种分隔符来分隔字段、MAP中的多个元素、每个元素的键和值。

```
[root@node2 data]# cat test2.txt
Lucy|province:guangdong,city:shenzhen
Tom|province:anhui,city:hefei
Sophie|province:jiangsu,city:nanjing
[root@node2 data]#
```

3.2.3 任务实施(实践)

步骤1. 数据文件准备

test2.txt

```
# 创建test2.txt文件
cat > test2.txt <<EOF
张三|城市:北京,街道:朝阳区,邮编:100000
李四|城市:上海,电话:13800138000,邮箱:lisi@example.com
王五|年龄:25,部门:技术部,工资:8000
赵六|公司:阿里巴巴,职位:工程师,工号:A1001
EOF
```

重要提示：数组元素之间不能有空格，否则 Hive 无法正确识别。

步骤2. 上传文件到 HDFS# 创建 HDFS 目录

```
# 上传文件
hdfs dfs -put test2.txt /user/hive/warehouse/test_data/test2.txt

# 确认上传
hdfs dfs -ls /user/hive/warehouse/test_data/
hdfs dfs -cat /user/hive/warehouse/test_data/test2.txt
```

3.2.3 任务实施(实践)

步骤3[hive]. 加载数据到 Hive 表

```
-- 从 HDFS 路径加载数据
LOAD DATA INPATH '/user/hive/warehouse/test_data/test2.txt'
INTO TABLE test2;
```

步骤43[hive]. 验证数据加载

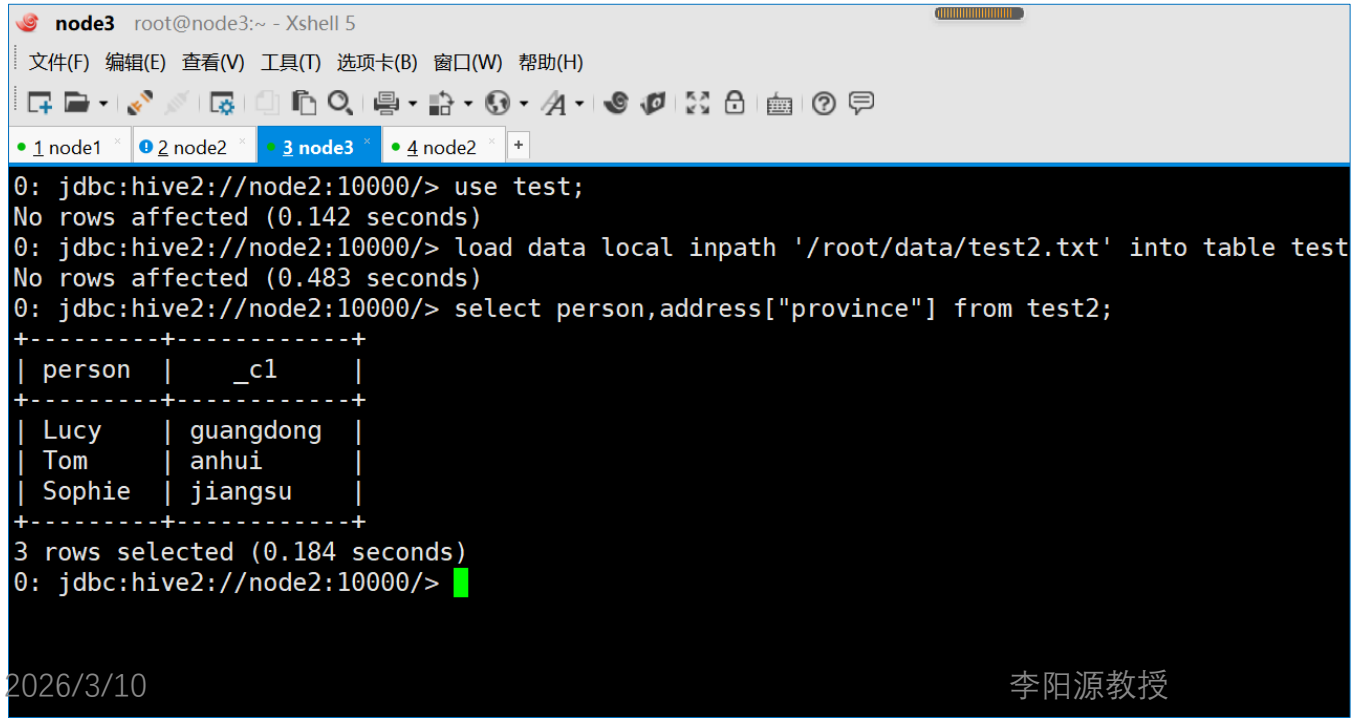
```
0: jdbc:hive2://localhost:10000> show tables;
+------------+
|  tab_name  |
+------------+
| dual       |
| employees  |
| test1      |
| test2      |
+------------+
0: jdbc:hive2://localhost:10000> select * from  test2;
+---------------+-----------------------------------------------------+
| test2.person  |                   test2.address                     |
+---------------+-----------------------------------------------------+
| 张三          | {"城市":"北京","街道":"朝阳区","邮编":"100000"}              |
| 李四          | {"城市":"上海","电话":"13800138000","邮箱":"lisi@example.com"} |
| 王五          | {"年龄":"25","部门":"技术部","工资":"8000"}                  |
| 赵六          | {"公司":"阿里巴巴","职业":"工程师","工号":"A1001"}             |
+---------------+-----------------------------------------------------+
0: jdbc:hive2://localhost:10000>
```

4．复杂运算符(Complex Type Logical Operators)

（2）MAP运算符

加载test2.txt数据到test2表中后，查看MAP中某个key，如下：

```
select person,address["province"] from test2;
```



3.2.3 任务实施

```
0: jdbc:hive2://localhost:10000> select person, address["城市"] from test2;
+---------+-------+
| person  |  _c1  |
+---------+-------+
| 张三      | 北京    |
| 李四      | 上海    |
| 王五      | NULL  |
| 赵六      | NULL  |
+---------+-------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

4．复杂运算符(Complex Type Logical Operators)

（3）STRUCT运算符

创建名为test3的表，只有person和info两个字段，

person为字符串类型，

info为struct类型，

里面有age和city两个属性。

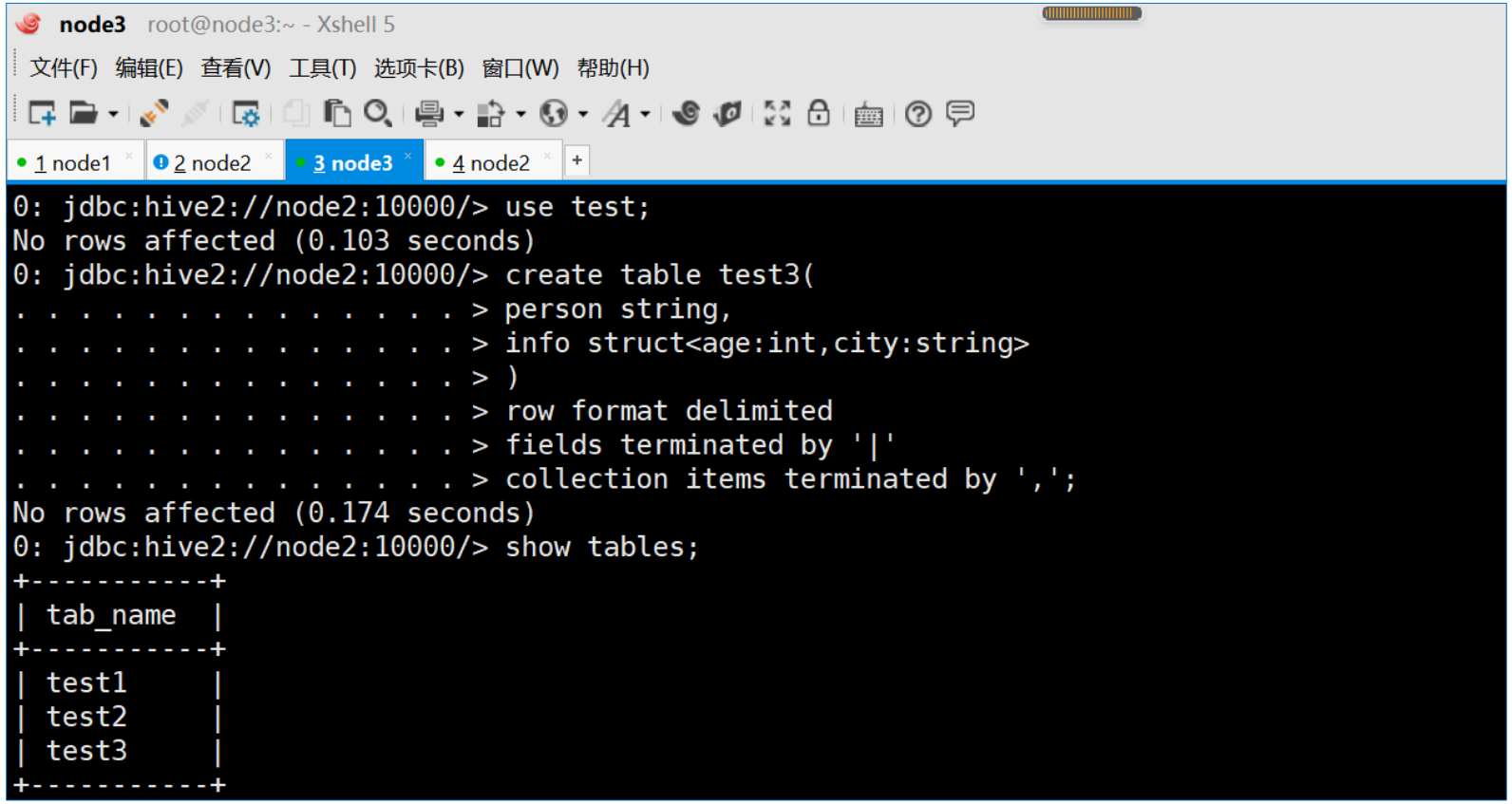
通过文本导入数据时，对分隔符的定义如下：

person和info之间的分隔符为竖线；

info内部元素之间的分隔符是逗号。

4．复杂运算符(Complex Type Logical Operators)

（3）STRUCT运算符



3.2.3 任务实施

```
0: jdbc:hive2://localhost:10000> CREATE TABLE test3 (
0: jdbc:hive2://localhost:10000>     person STRING,                          -- 字符串类型
0: jdbc:hive2://localhost:10000>     info STRUCT<age:INT, city:STRING>       -- STRUCT类型，包含age和city两个属性
0: jdbc:hive2://localhost:10000> )
0: jdbc:hive2://localhost:10000> ROW FORMAT DELIMITED
0: jdbc:hive2://localhost:10000> FIELDS TERMINATED BY '|'                    -- person和info之间的分隔符：竖线
0: jdbc:hive2://localhost:10000> COLLECTION ITEMS TERMINATED BY ','          -- info内部元素之间的分隔符：逗号
0: jdbc:hive2://localhost:10000> STORED AS TEXTFILE;                         -- 文本文件格式
0: jdbc:hive2://localhost:10000> show tables;
+------------+
|  tab_name  |
+------------+
| dual       |
| employees  |
| test1      |
| test2      |
| test3      |
+------------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

4．复杂运算符(Complex Type Logical Operators)

（3）STRUCT运算符

创建文本文件test3.txt，数据如下：

```
[root@node2 data]# cat test3.txt
Lucy|12,shenzhen
Tom|13,hefei
Sophie|11,nanjing
[root@node2 data]#
```

3.2.3 任务实施(实践)

步骤1. 数据文件准备

test3.txt

```
# 创建测试数据文件
cat > test3.txt <<EOF
张三|30,北京
李四|25,上海
王五|35,广州
赵六|28,深圳
钱七|40,杭州
EOF
```

重要提示：数组元素之间不能有空格，否则 Hive 无法正确识别。

步骤2. 上传文件到 HDFS# 创建 HDFS 目录

```
# 上传文件
hdfs dfs -put test3.txt /user/hive/warehouse/test_data/test3.txt

# 确认上传
hdfs dfs -ls /user/hive/warehouse/test_data/
hdfs dfs -cat /user/hive/warehouse/test_data/test3.txt
```

3.2.3 任务实施(实践)

步骤3[hive]. 加载数据到 Hive 表

```
-- 从 HDFS 路径加载数据
LOAD DATA INPATH '/user/hive/warehouse/test_data/test3.txt'
INTO TABLE test3;
```

步骤43[hive]. 验证数据加载

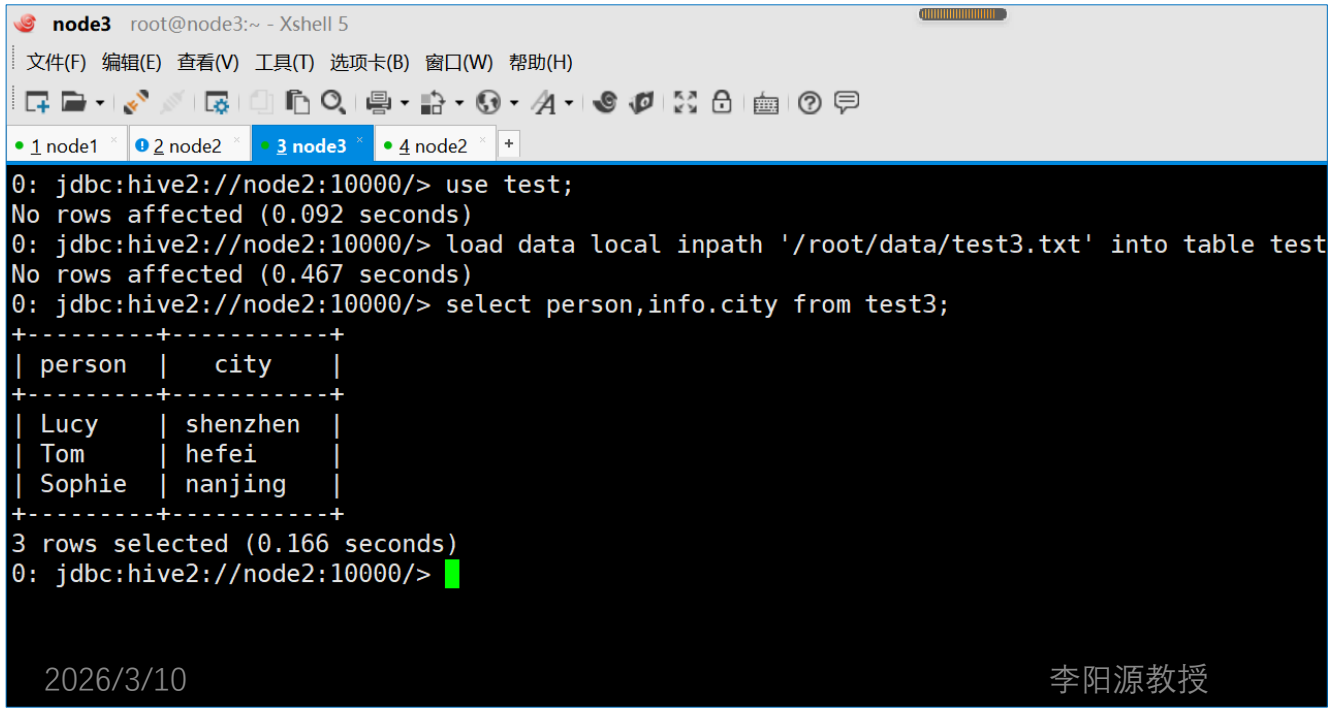
```
0: jdbc:hive2://localhost:10000> load data inpath'/user/hive/warehouse/test_data/test3.txt'
0: jdbc:hive2://localhost:10000> into table test3;
0: jdbc:hive2://localhost:10000> show tables;
+------------+
|  tab_name  |
+------------+
| dual       |
| employees  |
| test1      |
| test2      |
| test3      |
+------------+
0: jdbc:hive2://localhost:10000> select  * from test3;
+---------------+-------------------------+
| test3.person  |       test3.info        |
+---------------+-------------------------+
| 张三            | {"age":30,"city":"北京"}  |
| 李四            | {"age":25,"city":"上海"}  |
| 王五            | {"age":35,"city":"广州"}  |
| 赵六            | {"age":28,"city":"深圳"}  |
| 钱七            | {"age":40,"city":"杭州"}  |
+---------------+-------------------------+
0: jdbc:hive2://localhost:10000>
```

4．复杂运算符(Complex Type Logical Operators)

（3）STRUCT运算符

加载数据后，查看指定字段，比如查看所在的城市city，如下所示：

```
select person,info.city from test3;
```



3.2.3 任务实施

```
Terminal File Edit View Search Terminal Help
0: jdbc:hive2://localhost:10000> select person, info.city from test3;
+---------+-------+
| person  | city  |
+---------+-------+
| 张三      | 北京    |
| 李四      | 上海    |
| 王五      | 广州    |
| 赵六      | 深圳    |
| 钱七      | 杭州    |
+---------+-------+
0: jdbc:hive2://localhost:10000>
```

!quit	!exit	Ctrl+D
!history	!set	!set verbose false

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题1. 关于Hive关系运算符，以下说法正确的是？

A) = 运算符只能用于数字比较
B) <> 运算符表示"大于"
C) IS NULL 用于判断表达式是否为NULL
D) LIKE 运算符不能使用%通配符

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题2. 以下Hive查询的结果是什么？

```
SELECT 3.14 = 3.140, 'apple' = 'pear';
```

A) true, true
B) false, false
C) true, false
D) false, true

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题3. 关于RLIKE运算符，以下描述错误的是？

A) 是REGEXP_LIKE的同义词
B) 用于正则表达式匹配
C) ^h.*o$ 表示以h开头、以o结尾的字符串
D) 不能使用.匹配任意字符

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题4. 执行以下查询的结果是？

```
SELECT 5 <> 6, '5a' <> '5c';
```

A) false, false
B) true, false
C) false, true
D) true, true

Hive运算符知识测验（4选1）

问题5. 关于算术运算符，以下说法正确的是？

A) % 用于求平方根
B) / 只返回整数部分
C) * 优先级高于 +
D) Hive不支持浮点数运算

Hive运算符知识测验（4选1）

问题6. 逻辑运算符AND、OR、NOT中，优先级最高的是？

A) AND
B) OR
C) NOT
D) 三者优先级相同

Hive运算符知识测验（4选1）

问题7. 以下查询的结果是什么？

```
SELECT NOT 8 > 9, NOT 1 = 1;
```

A) true, true
B) false, false
C) true, false
D) false, true

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题8. 对于ARRAY类型字段friends，如何访问第一个元素？

A) friends(0)
B) friends[1]
C) friends[0]
D) friends.first

Hive运算符知识测验（4选1）

问题9. 创建包含MAP类型的表时，键和值之间的分隔符使用哪个选项定义？

A) FIELDS TERMINATED BY
B) COLLECTION ITEMS TERMINATED BY
C) MAP KEYS TERMINATED BY
D) LINES TERMINATED BY

Hive运算符知识测验（4选1）

问题10. 对于STRUCT类型字段info，如何访问其中的city属性？

A) info["city"]
B) info.city
C) info->city
D) info(city)

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题11. 根据以下表定义，正确的数据格式是?

```
CREATE TABLE test2 (
    person STRING,
    address MAP<STRING, STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':';
```

A) 张三,城市:北京,街道:朝阳区
B) 张三|城市:北京,街道:朝阳区
C) 张三|城市:北京|街道:朝阳区
D) 张三:城市,北京,街道,朝阳区

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题12. 关于复杂运算符的描述，以下错误的是?

A) ARRAY[n] 返回数组的第n个元素
B) MAP[key] 返回指定key对应的value
C) STRUCT.x 返回STRUCT的x字段值
D) 所有复杂类型的索引都从1开始

3.1.3 任务实施

Hive运算符知识测验（2选1）

问题13. LIKE 'he%' 可以匹配 'hello' 和 'he'。

A) 正确
B) 错误

3.1.3 任务实施

Hive运算符知识测验（2选1）

问题14. 在Hive中，NULL = NULL 的结果是true。

A) 正确
B) 错误

3.1.3 任务实施

Hive运算符知识测验（2选1）

问题15. 逻辑表达式 1=1 OR 8>10 AND 'a'='b' 的结果是true。

A) 正确
B) 错误

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题16. 根据以下表定义，正确的数据格式是？

```
CREATE TABLE employee (
    name STRING,
    skills ARRAY<STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
COLLECTION ITEMS TERMINATED BY ',';
```

A) 张三 Python,Java,SQL
B) 张三,Python,Java,SQL
C) 张三|Python,Java,SQL
D) 张三 Python Java SQL

3.1.3 任务实施

Hive运算符知识测验（4选1）

问题17. 根据以下表定义，正确的数据格式是？

```
CREATE TABLE student_score (
    student_id INT,
    scores MAP<STRING, FLOAT>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '#'
MAP KEYS TERMINATED BY ':';
```

A) 1001,数学:95.5#语文:88.0#英语:92.5
B) 1001#数学:95.5#语文:88.0#英语:92.5
C) 1001,数学#95.5,语文#88.0,英语#92.5
D) 1001:数学,95.5:语文,88.0:英语,92.5