一、内部表：

内部表与关系型数据库中的 Table 在概念上类似。每一个 Table 在概念上类似。每一个 Table 在 Hive 中有一个相应的目录存储数据。所有的 Table 数据（不包括 External Table）都保存在这个目录中。删除表时，元数据与数据都会被删除。

1、新建一个测试文件：vim /usr/local/filetest/city.txt

```
beijings,j,beijing
tianjings,tj,tj
hebeish,j,shijiaz
shangxish,j,taiyuan
neimenggu,n,huhehaote
lianlins,l,shengyang
jilingsheng,ji,changchu
heilongj,h,haerbing
shanghaishi,hu,shanghai
jiangsusheng,su,nanjing
```

2、

```
CREATE TABLE city(
province string,
code string ,
capital string
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

在 hive 命令行中中执行此数据脚本

3、对需要进行的数据进行加载。

```
LOAD DATA LOCAL INPATH '/usr/local/filetest/city.txt' INTO TABLE city
```

4、查询所有城市

```
select * from city;
```

5、执行统计信息：

```
SELECT COUNT(*) FROM city;
```

```
hive> SELECT COUNT(*) FROM city;
WARNING: 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.
Query ID = root_20170414132149_45f93111-2f59-4cbf-ad9b-968e9142688a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
set mapreduce.job.reduces=<number>
```

```
Job running in-process (local Hadoop)
2017-04-14 13:21:50,772 Stage-1 map = 100%, reduce = 100%
Ended Job = job_local1234318840_0005
MapReduce Jobs Launched:
Stage-Stage-1: HDFS Read: 2898 HDFS Write: 418 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
10
Time taken: 1.292 seconds, Fetched: 1 row(s)
```

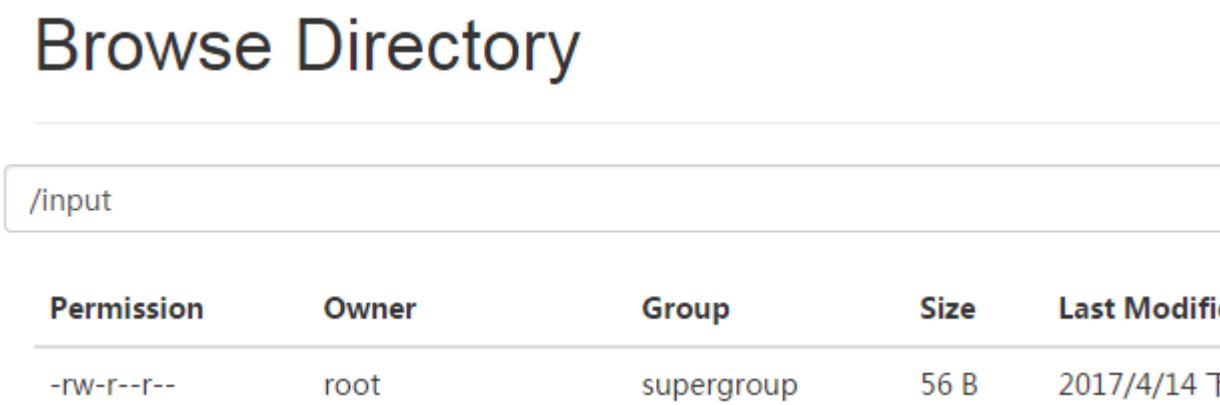
二、外部表

外部表指向已经在 HDFS 存在的数据，可以创建 Partition。它和内部表在元数据的组织上是相同的，而实际数据存储存在的较大的差异。

建立一个文件:vim /usr/local/filetest/person.txt

```
1,a,19
2,b,18
3,c,17
4,d,19
5,e,20
6,f,22
7,g,56
8,h,29
```

将文件放到 hdfs 上:hadoop fs -put /usr/local/filetest/person.txt /input



建立一个外部表：

```
CREATE EXTERNAL TABLE person(
pid int ,
name string ,
age int
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/input';
```

查询:SELECT * FROM person ;

```
hive> SELECT * FROM person ;
```

```
OK
1 a 19
2 b 18
3 c 17
4 d 19
5 e 20
6 f 22
7 g 56
8 h 29
```

三、分区表：

```
CREATE TABLE stu(
sid int ,
name string ,
score double
) PARTITIONED BY (tdate string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

此字段作为标记，一般用时间作为标记。

对要分析的文件加载:

```
LOAD DATA LOCAL INPATH '/usr/local/stu1.txt' INTO TABLE stu
PARTITION(tdate=20191010) ;
LOAD DATA LOCAL INPATH '/usr/local/stu2.txt' INTO TABLE stu
PARTITION(tdate=20191111) ;
```

查看全部数据：`SELECT * FROM stu;`
指定时间查看：`SELECT * FROM stu3 WHERE tdate=20191111 ;`

四、桶表

用算法（hash）将将不同数据保存在桶空间内。

```
CREATE TABLE stu2(
sid int ,
name string ,
score double
) CLUSTERED BY (sid) INTO 2 BUCKETS ;
```

根据 sid%2 原则分桶，0 一个桶。非 0 一个桶。

加入数据：

INSERT INTO stu2(sid,name,score) VALUES (1,'nihao',99) ;
INSERT INTO stu2(sid,name,score) VALUES (2,'hello',67) ;
查询全部数据:
SELECT * FROM stu8;
查询第一桶数据:SELECT * FROM stu8 TABLESAMPLE(bucket 1 OUT OF 2 ON sid);
查询第二桶数据:SELECT * FROM stu8 TABLESAMPLE(bucket 2 OUT OF 2 ON sid);