第3章 Hive的数据定义语言

学习目标：

- 掌握数据库的基本操作，能够灵活使用 HiveQL 语句对 Hive 中的数据库进行创建、查询、显示信息、切换、修改以及删除的操作。
- 了解 CREATE TABLE 句式语法，能够描述 CREATE TABLE 句式中不同子句的作用。
- 掌握数据表的基本操作，能够灵活使用 HiveQL 语句对 Hive 中的数据表进行创建、查看、修改和删除的操作。
- 掌握分区表的基本操作，能够灵活使用 HiveQL 语句对 Hive 中的分区表进行创建、查询、添加、重命名、移动和删除的操作。
- 掌握分桶表的基本操作，能够灵活使用 HiveQL 语句对 Hive 中的分桶表进行创建和查看信息的操作。
- 掌握临时表的基本操作，能够灵活使用 HiveQL 语句在 Hive 中创建临时表。
- 掌握视图的基本操作，能够灵活使用 HiveQL 语句对 Hive 中的视图进行创建、查看、修改以及删除的操作。
- 了解索引的原理，能够描述 Hive 中索引的作用与优势。
- 掌握索引的基本操作，能够灵活使用 HiveQL 语句对 Hive 数据表中的索引进行创建、查看、重建和删除的操作。

Hive 提供了用于定义数据表结构和数据库对象的语言，称为数据定义语言（简称 DDL）。接下来，本章针对 Hive 的数据定义语言（DDL）进行详细讲解。

3.1 数据库的基本操作

3.1.1 创建数据库

Hive 中创建数据库的语法格式如下。

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name
[COMMENT database_comment]
[LOCATION hdfs_path]
[WITH DBPROPERTIES (property_name=property_value, ...)];
```

上述语法的具体讲解如下。

- CREATE (DATABASE|SCHEMA)：表示创建数据库的语句，其中 DATABASE 和 SCHEMA 含义相同，可以切换使用。
- IF NOT EXISTS：可选，用于判断创建的数据库是否已经存在，若不存在则创建数据库，反之不创建数据库。
- database_name：表示创建的数据库名称。
- COMMENT database_comment：可选，表示数据库的相关描述。
- LOCATION hdfs_path：可选，用于指定数据库在 HDFS 上的存储位置，默认存储位置取决于 Hive 配置文件 hive-site.xml 中参数 hive.metastore.warehouse.dir 指定的存储位置。
- WITH DBPROPERTIES (property_name=property_value, ...)：可选，用于设置数据库属性，其中 property_name 表示属性名称，该名称可以自定义；property_value 表示属性值，该值可以自定义。

接下来，在 Hive 客户端工具 Beeline 中创建数据库 itcast，并指定数据库文件存放在 HDFS 的/hive_db/create_db/目录中，具体命令如下。

```
CREATE DATABASE IF NOT EXISTS itcast
COMMENT "This is itcast database"
LOCATION '/hive_db/create_db/'
WITH DBPROPERTIES ("creator"="itcast", "date"="2020-08-08");
```

上述命令中，添加了数据库描述和数据库属性，其中数据库描述为 This is itcast database；数据库属性为 creator 和 date，这两个属性对应的值分别是 itcast 和 2020-08-08。

3.1.2　查询数据库

Hive 中查询数据库的语法格式如下所示。

```
SHOW (DATABASES|SCHEMAS) [LIKE 'identifier_with_wildcards'];
```

上述语法的具体讲解如下。

- SHOW (DATABASES|SCHEMAS)：表示查询数据库的语句，其中 DATABASE 和 SCHEMA 含义相同，可以切换使用。
- LIKE 'identifier_with_wildcards'：可选，LIKE 子句用于模糊查询，identifier_with_wildcards 用于指定查询条件。

接下来，查询 Hive 中所有数据库，具体命令如下。

```
SHOW DATABASES;
```

如果要查询 Hive 中数据库名称的首字母是 i 的数据库，具体命令如下。

```
SHOW DATABASES LIKE 'i*';
```

上述命令在 Hive 客户端工具 Beeline 的执行效果如图 3-1 所示。

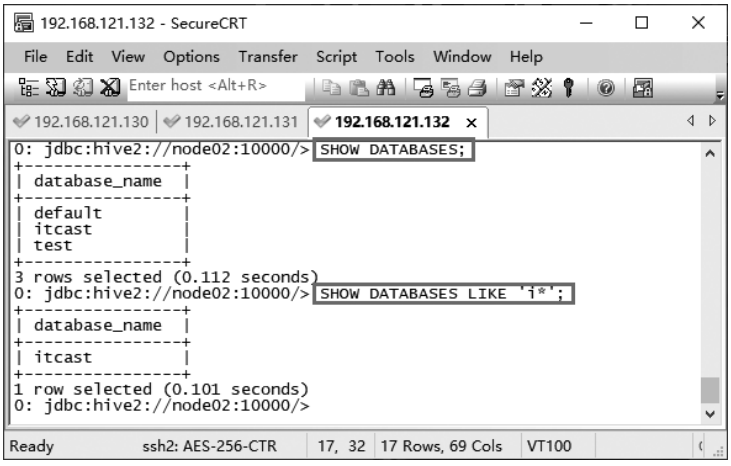

图 3-1 查询 Hive 所有数据库和指定数据库

3.1.3 查看数据库信息

Hive 中查看数据库信息的语法格式如下所示。

```
DESCRIBE|DESC (DATABASES|SCHEMAS) [EXTENDED] db_name;
```

上述语法的具体讲解如下。

- DESCRIBE|DESC (DATABASES|SCHEMAS)：表示查询数据库信息的语句，其中 DESCRIBE 和 DESC 含义相同，可以切换使用。
- EXTENDED：可选，在查询数据库的信息中显示属性。
- db_name：用于指定查询的数据库名称。

接下来，查看 Hive 中数据库 itcast 的信息，具体命令如下。

```
DESC DATABASE EXTENDED itcast;
```

上述命令在 Hive 客户端工具 Beeline 的执行效果如图 3-2 所示。

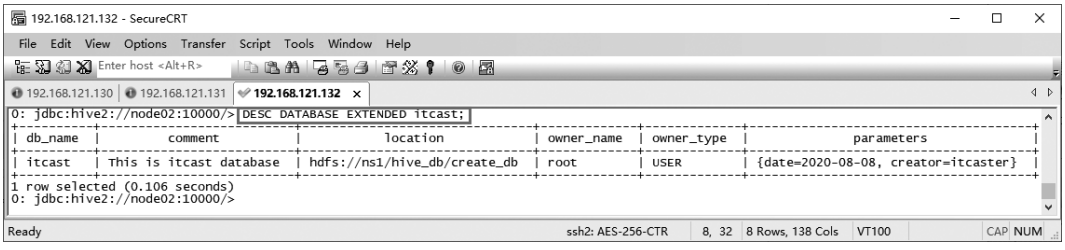

图 3-2 查看数据库 itcast 的信息

在图 3-2 中，数据库 itcast 中的信息包含 6 个字段。其中，db_name 表示数据库名称，comment 表示数据库描述，location 表示数据库在 HDFS 上的存储位置，owner_name 表示

数据库所有者名称，owner_type 表示数据库所有者类型，parameters 表示数据库属性。

3.1.4　切换数据库

使用 Hive 客户端工具 Beeline 或者 HiveCLI 操作 Hive 时，默认打开的数据库是 default。如果使用已创建的其他数据库，则需要手动切换。Hive 中切换数据库的语法格式如下所示。

```
USE db_name;
```

上述语法的具体讲解如下。

- USE：表示切换数据库的语句。
- db_name：用于指定要切换的数据库名称。

例如，将当前使用的数据库切换至数据库 itcast，具体命令如下。

```
USE itcast;
```

上述命令在 Hive 客户端工具 Beeline 执行后，使用 SELECT 语句查看当前使用的数据库，具体如图 3-3 所示。

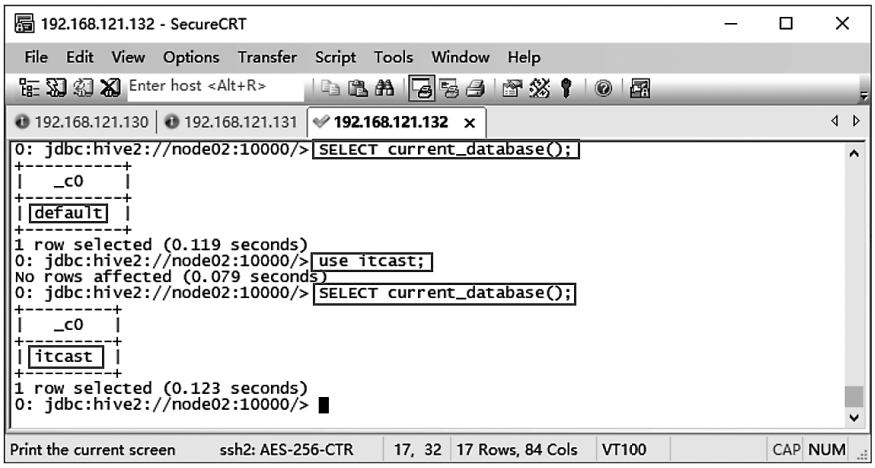

图 3-3　查看当前使用的数据库

从图 3-3 可以看出，第一次使用 SELECT 语句查看当前使用的数据库时显示的是 default，通过 USE 语句将当前使用的数据库切换到 itcast 之后，再次使用 SELECT 语句查看当前使用的数据库时显示的是 itcast，证明当前使用的数据库由 default 切换到 itcast。需要注意的是，在使用 Hive 客户端工具操作 Hive 时，一定要确认当前使用的数据库是否正确，避免将数据存储在错误的数据库中。

3.1.5　修改数据库

在 Hive 中可以修改数据库信息中的属性和所有者，修改数据库信息的语法格式如下所示。

```
/* 修改数据库属性 */
ALTER (DATABASE|SCHEMA) database_name SET DBPROPERTIES
                                        (property_name=property_value, ...);
/* 修改数据库所有者 */
ALTER (DATABASE|SCHEMA) database_name SET OWNER [USER|ROLE] user_or_role;
```

上述语法的具体讲解如下。

- ALTER（DATABASE | SCHEMA）：表示修改数据库信息的语句，其中 DATABASE 和 SCHEMA 含义相同，可以切换使用。
- database_name：指定数据库名称。
- SET DBPROPERTIES（property_name=property_value, ...）：指定修改数据库信息中的属性，在修改数据库信息的属性时，若属性已经存在则覆盖之前的属性值，反之添加该属性。
- SET OWNER [USER|ROLE] user_or_role：指定修改数据库信息中的所有者。

接下来，修改数据库 itcast 中的属性，具体命令如下。

```
ALTER DATABASE itcast SET DBPROPERTIES ("date"="2020-08-18", "locale"="beijing");
```

在 Hive 客户端工具 Beeline 中执行上述命令后，使用 DESCRIBE 查看修改后数据库 itcast 的信息，具体如图 3-4 所示。

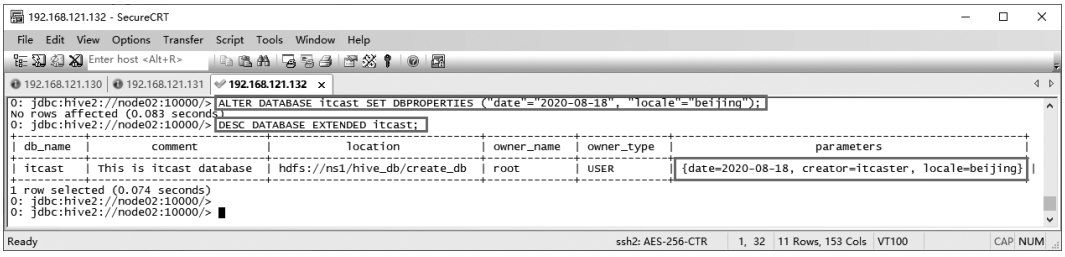

图 3-4　查看修改后数据库 itcast 的信息

通过对比图 3-2 和图 3-4 可以看出，图 3-4 展示的数据库 itcast 信息中，属性 date 的值由 2020-08-08 更改为 2020-08-18，增加了属性 locale。

3.1.6　删除数据库

Hive 中删除数据库的语法格式如下所示。

```
DROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];
```

上述语法的具体讲解如下。

- DROP（DATABASE|SCHEMA）：表示删除数据库的语句，其中 DATABASE 和 SCHEMA 含义相同，可以切换使用。
- IF EXISTS：可选，用于判断数据库是否存在。
- database_name：用于指定数据库名称。

- [RESTRICT|CASCADE]：可选，表示数据库中存在表时是否可以删除数据库。默认值为 RESTRICT，表示如果数据库中存在表则无法删除数据库。若使用 CASCADE，表示即使数据库中存在表，仍然会删除数据库并删除数据库中的表，因此需要谨慎使用 CASCADE。

接下来，删除数据库 itcast，具体命令如下。

```
DROP DATABASE IF EXISTS itcast;
```

上述命令在 Hive 客户端工具 Beeline 执行后，查询 Hive 中所有数据库，具体如图 3-5 所示。

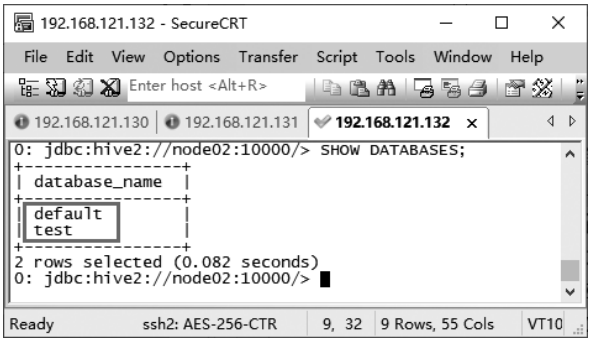

图 3-5　查询 Hive 中所有数据库

从图 3-5 可以看出，Hive 中的数据库只有 default 和 test，已经不存在数据库 itcast，证明成功删除了数据库 itcast。

3.2　数据表的基本操作

3.2.1　CREATE TABLE 句式分析

Hive 中使用 CREATE TABLE 句式创建数据表，CREATE TABLE 句式的语法格式如下。

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type [COMMENT col_comment], ... [constraint_specification])]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  [CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name [ASC|DESC], ...)]
      INTO num_buckets BUCKETS]
  [SKEWED BY (col_name, col_name, ...)]
     ON ((col_value, col_value, ...), (col_value, col_value, ...), ...)
     [STORED AS DIRECTORIES]
  [
   [ROW FORMAT row_format]
   [STORED AS file_format]
     | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]
  ]
```

```
[LOCATION hdfs_path]
[TBLPROPERTIES (property_name=property_value, ...)]
[AS select_statement];
```

上述语法的具体讲解如下。

- TEMPORARY：可选，用于指定创建的表为临时表。
- EXTERNAL：可选，用于指定创建的表为外部表，若不指定则默认创建内部表。
- IF NOT EXISTS：可选，用于判断创建的表是否存在。
- db_name：可选，用于指定创建表时存放的数据库，若不指定则默认在当前数据库创建。
- table_name：指定表名称。
- col_name：指定表中的字段名称，若创建的数据表是空表，则 col_name 为可选的。
- data_type：指定字段类型，若创建的数据表是空表，则 data_type 为可选的。
- COMMENT col_comment：可选，指定字段描述。
- constraint_specification：可选，指定字段约束，支持 Hive 3.0 及以上版本。
- COMMENT table_comment：可选，指定字段描述。
- PARTITIONED BY (col_name data_type [COMMENT col_comment],...)]：可选，用于创建分区表，指定分区名(col_name)、分区类型(data_type)和分区描述(col_comment)。
- [CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]：可选，用于创建分桶表，指定分桶的字段(col_name)、根据指定字段对桶内的数据进行升序(ASC，默认)或降序(DESC)排序以及桶的数量(num_buckets)。
- [SKEWED BY (col_name, col_name, ...)] ON ((col_value, col_value, ...), (col_value, col_value, ...), ...) [STORED AS DIRECTORIES]：可选，用于创建倾斜表解决 Hive 中数据倾斜问题，其中 SKEWED BY col_name 指定出现数据倾斜的字段，ON col_value 指定数据倾斜字段中数据倾斜的值，STORED AS DIRECTORIES 将数据倾斜字段中出现频繁的值拆分成文件夹，若不指定则拆分成文件。
- ROW FORMAT row_format：可选，用于序列化行对象。
- STORED AS file_format：可选，用于创建表时指定 Hive 表的文件存储格式。
- LOCATION hdfs_path：可选，用于指定 Hive 表在 HDFS 的存储位置。
- TBLPROPERTIES (property_name = property_value, ...)：可选，用于指定表属性。
- AS select_statement：可选，用于在创建表的同时将查询结果插入表中。

通过对上述创建数据库表 CREATE TABLE 句式的学习，读者对 Hive 中创建数据库表的方式有了初步认识，本章后续的内容会详细讲解 CREATE TABLE 句式的实际应用。

多学一招：Serde 和表属性

1. Serde

Serde 是 Serializer and Deserializer(序列化和反序列化)的简称，Hive 通过 Serde 处理

Hive 数据表中每一行数据的读取和写入，例如查询 Hive 数据表数据时，HDFS 中存放的数据表数据会通过 Serializer 序列化为字节流便于数据传输；向 Hive 数据表插入数据时，会通过 Deserializer 将数据反序列化成 Hive 数据表的每一行值，方便将数据加载到数据表中，不需要对数据进行转换。

Hive 中的 Serde 分为自定义 Serde 和内置 Serde，其中使用自定义 Serde 时需要在 CREATE TABLE 句式中指定 ROW FORMAT 子句的 row_format 值为 Serde，并根据 Serde 类型指定实现类；内置 Serde 需要在 CREATE TABLE 句式中指定 ROW FORMAT 子句的 row_format 值为 DELIMITED。Hive 中常用的自定义 Serde 和内置 Serde 如表 3-1 和表 3-2 所示。

表 3-1　常用的自定义 Serde

自定义 Serde	介　绍
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe' WITH SERDEPROPERTIES ("input.regex" ="regex") STORED AS TEXTFILE;	使用正则表达式序列化/反序列化数据表的每一行数据，其中 regex 用于指定正则表达式
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe' STORED AS TEXTFILE	使用 JSON 格式序列化/反序列化数据表的每一行数据
CREATE TABLE my_table(a string, b string, ...) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' WITH SERDEPROPERTIES ("separatorChar" = "\t", "quoteChar" = "'", "escapeChar" = "\\") STORED AS TEXTFILE;	使用 CSV 格式序列化/反序列化数据表的每一行数据，其中 separatorChar 用于指定 CSV 文件的分隔符；quoteChar 用于指定 CSV 文件的应用符；escapeChar 用于指定 CSV 文件的转义符

表 3-2　常用的内置 Serde

内置 Serde	介　绍
FIELDS TERMINATED BY char [ESCAPED BY char]	FIELDS TERMINATED 指定字段分隔符；ESCAPED 指定转义符，避免数据中存在与字段分隔符一样的字符，造成混淆
COLLECTION ITEMS TERMINATED BY char	指定集合中元素的分隔符，集合包含数据类型为 MAP、ARRAY 和 STRUCT
MAP KYS TERMINATED BY char	指定 MAP 中 Key 和 Value 的分隔符
LINES TERMINATED BY char	指定行分隔符
NULL DEFINED AS char	自定义空值格式，Hive 默认为'\N'

2. 表属性

通过 CREATE TABLE 句式创建数据表时可以使用 TBLPROPERTIES 子句指定表属性，Hive 表属性分为自定义属性和预定义属性，其中使用自定义属性时，用户可以自定义属性名称(property_name)和属性值(property_value)，用于为创建的数据表指定自定义标签，例如指定创建表的作者、创建表的时间等；使用预定义属性时，需要根据 Hive 规定的属性名称和属性值使用，用于为创建的数据表指定相关配置，有关 Hive 预定义属性如表 3-3 所示。

表 3-3 Hive 预定义属性

属　性	值	描　述
comment	table_comment	表描述
hbase.table.name	table_name	集成 HBase
immutable	true 或 false	防止意外更新，若为 true，则无法通过 Insert 实现数据的更新和插入
orc.compress	ZLIB 或 SNAPPY 或 NONE	指定 ORC 压缩方式
transactional	true 或 false	指定表是否支持 ACID(更新、插入、删除)
NO_AUTO_COMPACTION	true 或 false	表事务属性，指定表是否支持自动紧缩
compactor.mapreduce.map.memory.mb	mapper_memory	表事务属性，指定紧缩 map(内存/MB)作业的属性
compactorthreshold.hive .compactor.delta.num.threshold	threshold_num	表事务属性，如果有超过 threshold_num 个增量目录，则触发轻度紧缩
compactorthreshold.hive .compactor.delta.pct.threshold	threshold_pct	表事务属性，如果增量文件的大小与基础文件的大小比率大于 threshold_pct(区间为 0～1)，则触发深度紧缩
auto.purge	true 或 false	若为 true，则删除或者覆盖的数据会不经过回收站直接被删除
EXTERNAL	true 或 false	内部表和外部表的转换

3.2.2 数据表简介

数据表是 Hive 存储数据的基本单位，Hive 数据表主要分为内部表(又叫托管表)和外部表，以内部表和外部表为基础可以创建分区表或分桶表，即内/外部分区表或内/外部分桶表。接下来，针对内部表和外部表进行详细讲解。

默认情况下，内部表和外部表的数据都存储在 Hive 配置文件中参数 hive.metastore.warehouse.dir 指定的路径。它们的区别在于删除内部表时，内部表的元数据和数据会一同删除；而删除外部表时，只删除外部表的元数据，不会删除数据。外部表相对来说更加安全，数据组织更加灵活并且方便共享源数据文件。

3.2.3 创建数据表

在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline，远程连接虚拟机 Node_02 的

HiveServer2 服务操作 Hive。在 Hive 中创建一个数据库 hive_database，并在该数据库中通过 CREATE TABLE 句式创建内部表 managed_table 和外部表 external_table。

(1) 创建内部表 managed_table 的命令如下。

```
CREATE  TABLE IF NOT EXISTS
hive_database.managed_table(
staff_id INT COMMENT "This is staffid",
staff_name STRING COMMENT "This is staffname",
salary FLOAT COMMENT "This is staff salary",
hobby ARRAY<STRING> COMMENT "This is staff hobby",
deductions MAP<STRING, FLOAT> COMMENT "This is staff deduction",
address STRUCT<street:STRING,city:STRING> COMMENT "This is staff address"
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '_'
MAP KEYS TERMINATED BY ':'
LINES TERMINATED BY '\n'
STORED AS textfile
TBLPROPERTIES("comment"="This is a managed table");
```

上述命令中，指定 ROW FORMAT DELIMITED 子句使用 Hive 内置的 Serde，自定义字段(FIELDS)分隔符为“,”；自定义集合元素(COLLECTION ITEMS)的分隔符为“_”；自定义 MAP(MAP KEYS)的键值对分隔符为“:”；自定义行(LINES)分隔符为\n。

(2) 创建外部表 external_table 的命令如下。

```
CREATE EXTERNAL TABLE IF NOT EXISTS
hive_database.external_table
(
staff_id INT COMMENT "This is staffid",
staff_name STRING COMMENT "This is staffname",
salary FLOAT COMMENT "This is staff salary",
hobby ARRAY<STRING> COMMENT "This is staff hobby",
deductions MAP<STRING, FLOAT> COMMENT "This is staff deduction",
address STRUCT<street:STRING,city:STRING> COMMENT "This is staff address"
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '_'
MAP KEYS TERMINATED BY ':'
LINES TERMINATED BY '\n'
STORED AS textfile
LOCATION '/user/hive_external/external_table/'
TBLPROPERTIES("comment"="This is a external table");
```

上述命令中，通过在 CREATE TABLE 句式中指定 EXTERNAL 子句创建外部表。创建外部表时通常配合 LOCATION 子句指定数据的存储位置，便于数据的维护与管理。

3.2.4 查看数据表

Hive 提供了查看当前数据库的所有数据表以及查看指定数据表结构信息的语句，具体语法格式如下。

```
/* 显示出当前数据库的所有数据表 */
SHOW TABLES [LIKE 'identifier_with_wildcards'];
/* 查看指定数据表结构信息 */
DESCRIBE|DESC [FORMATTED] table_name;
```

上述语法的具体讲解如下。

- SHOW TABLES：表示查看当前数据库的所有数据表。
- LIKE 'identifier_with_wildcards'：可选，LIKE 子句用于模糊查询，identifier_with_wildcards 用于指定查询条件。
- DESCRIBE|DESC：表示查询指定数据表基本结构信息，其中 DESCRIBE 和 DESC 含义相同，可以切换使用。
- FORMATTED：可选，表示查询指定数据表详细结构信息。

接下来，在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline，远程连接虚拟机 Node_02 的 HiveServer2 服务操作 Hive，讲解查看数据表语法的实际应用，具体操作步骤如下。

（1）切换到数据库 hive_database，具体命令如下。

```
USE hive_database;
```

（2）查看数据库 hive_database 的所有数据表，具体命令如下。

```
SHOW TABLES;
```

上述命令在 Hive 客户端工具 Beeline 中的执行效果如图 3-6 所示。

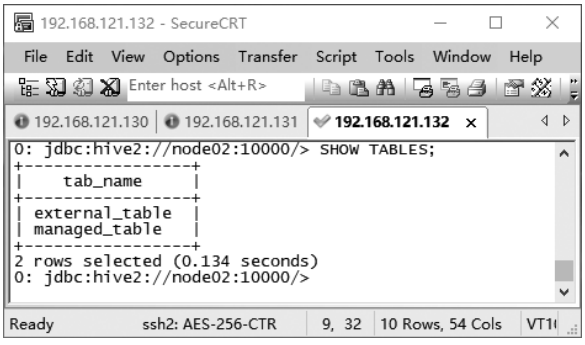

图 3-6 查看数据库 hive_database 的所有数据表

从图 3-6 可以看出，数据库 hive_database 中包含两个表，其中表 external_table 是 3.2.3 小节创建的外部表；表 managed_table 是 3.2.3 小节创建的内部表。

（3）通过模糊查询查看数据库 hive_database 中数据表名称首字母为 m 的数据表，具体

命令如下。

```
SHOW TABLES LIKE 'm* ';
```

上述命令在 Hive 客户端工具 Beeline 中的执行效果如图 3-7 所示。

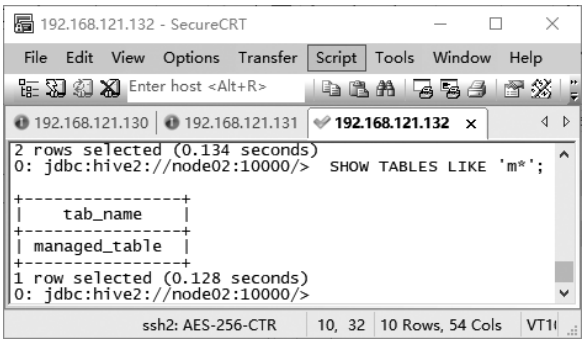

图 3-7　查看数据表名称首字母为 m 的数据表

从图 3-7 可以看出，数据库 hive_database 中数据表名称首字母为 m 的数据表只有 managed_table。

(4) 查看数据库 hive_database 中内部表 managed_table 表结构的基本信息，具体命令如下。

```
DESC managed_table;
```

上述命令在 Hive 客户端工具 Beeline 中的执行效果如图 3-8 所示。

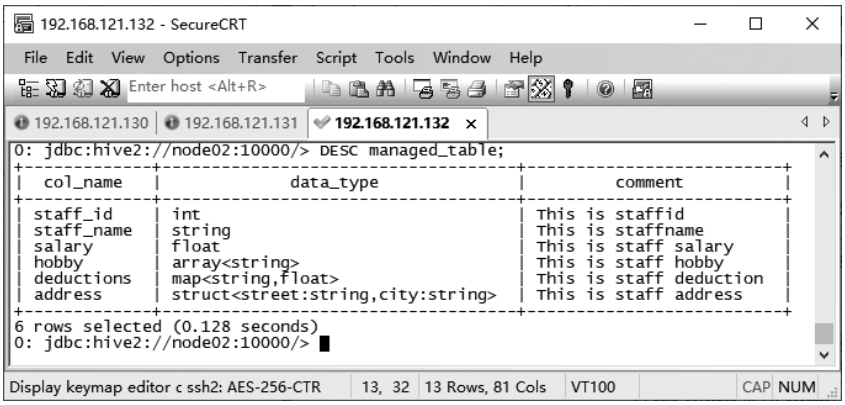

图 3-8　查看内部表 managed_table 表结构的基本信息

从图 3-8 可以看出，内部表 managed_table 表结构的基本信息包含字段名称(col_name)、字段数据类型(data_type)和字段描述(comment)。

(5) 查看数据库 hive_database 中外部表 external_table 表结构的详细信息，具体命令如下。

```
DESC FORMATTED external_table;
```

上述命令在 Hive 客户端工具 Beeline 中的执行效果如图 3-9 所示。

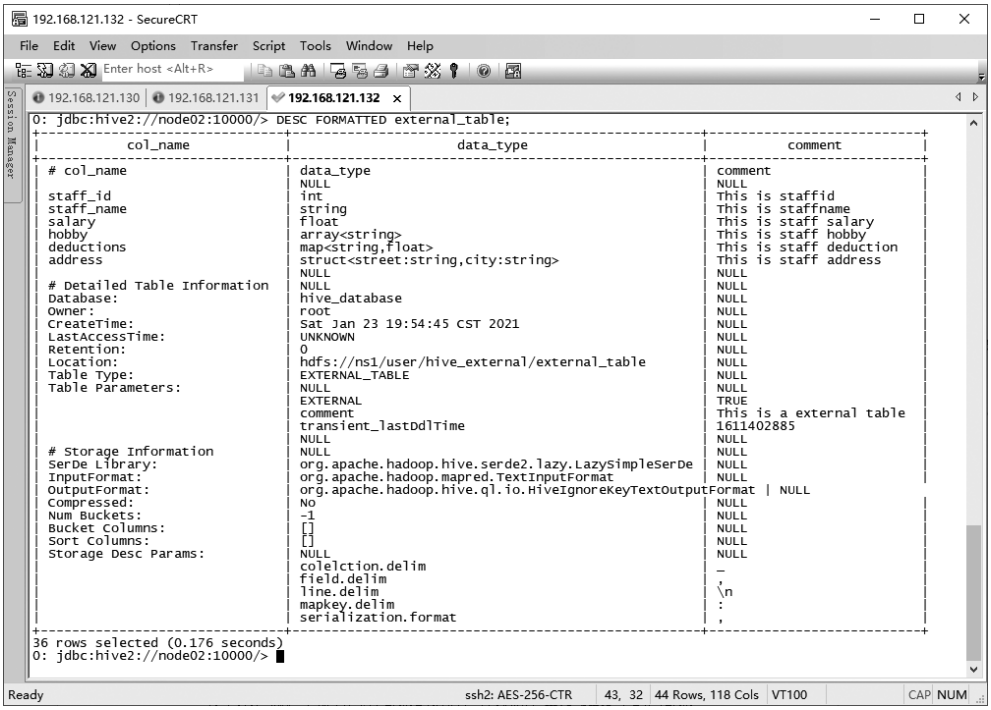

图 3-9 查看外部表 external_table 表结构的详细信息

从图 3-9 可以看出，外部表 external_table 表结构的详细信息中不仅包含了 col_name（字段）的相关信息，而且还包含了 Detailed Table Information（表详细信息）和 storage Information（存储信息）。

3.2.5 修改数据表

修改数据表结构信息使用 ALTER TABLE 语句实现，该语句只是修改数据表的元数据信息，数据表中的数据不会随之发生变化。下面介绍几种修改数据表结构信息的方法。

1. 重命名数据表

重命名数据表是指修改数据表的名称，语法格式如下。

```
ALTER TABLE table_name RENAME TO new_table_name;
```

上述语法的具体讲解如下。

- ALTER TABLE：表示修改数据表结构信息的语句。
- RENAME TO：用于数据表的重命名操作。
- table_name：指定需要重命名的数据表名称。
- new_table_name：指定重命名后的数据表名称。

接下来，在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline，远程连接虚拟机 Node_02

的 HiveServer2 服务操作 Hive，将数据库 hive_database 的内部表 managed_table 重命名为 managed_table_new，具体命令如下。

```
ALTER TABLE managed_table RENAME TO managed_table_new;
```

上述命令执行完成后，在 Hive 客户端工具 Beeline 中执行“SHOW TABLES;”命令，查看数据库 hive_database 的所有数据表，如图 3-10 所示。

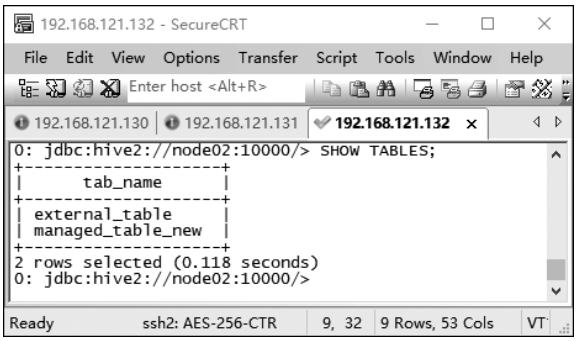

图 3-10　查看数据库 hive_database 的所有数据表

从图 3-10 可以看出，数据库 hive_database 中原有的内部表 managed_table 已经重命名为 managed_table_new。

2. 修改数据表的属性

修改数据表属性的语法格式如下。

```
ALTER TABLE table_name SET TBLPROPERTIES(property_name = property_value, property_name =
property_value, ... );
```

上述语法的具体讲解如下。

- ALTER TABLE：表示修改数据表结构信息的语句。
- SET TBLPROPERTIES：表示修改数据表属性的操作。
- table_name：指定需要修改属性的数据表名称。
- property_name = property_value：指定修改的属性(property_name)和属性值(property_value)。

接下来，在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline，远程连接虚拟机 Node_02 的 HiveServer2 服务操作 Hive，修改内部表 managed_table_new 的属性 comment，具体命令如下。

```
ALTER TABLE managed_table_new SET TBLPROPERTIES("comment"="This is a new managed table");
```

上述命令执行完成后，在 Hive 客户端工具 Beeline 中执行“DESC FORMATTED managed_table_new;”命令，查看数据库 hive_database 中内部表 managed_table_new 表结构的详细信息，如图 3-11 所示。

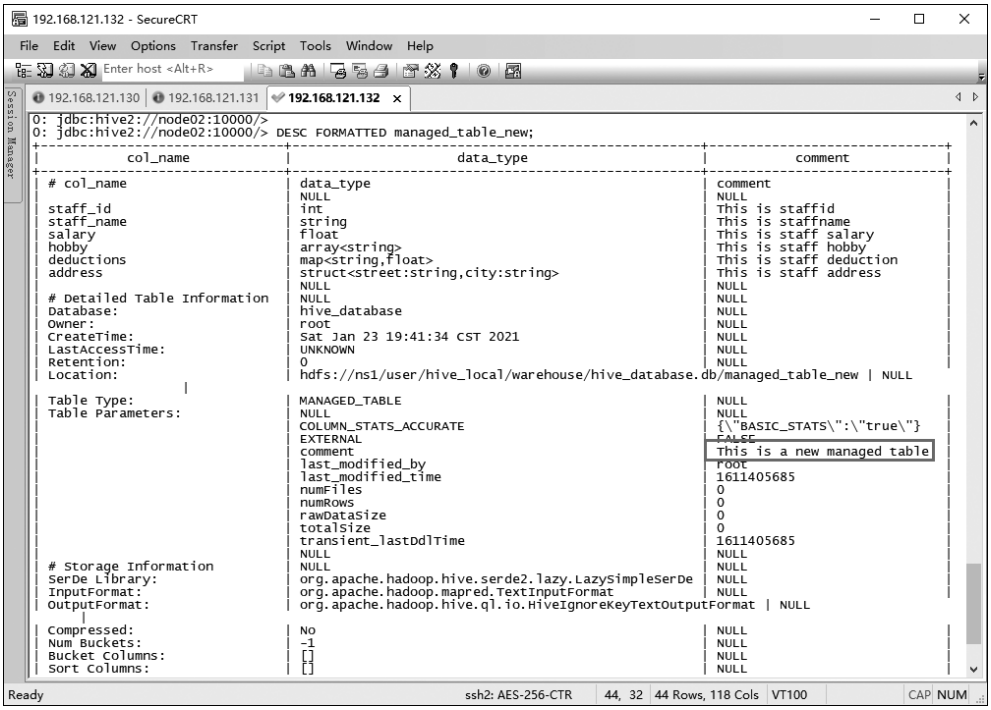

图 3-11 查看内部表 managed_table_new 表结构的详细信息

从图 3-11 可以看出，内部表 managed_table_new 属性 comment 的属性值由原始的 This is a managed table 修改为 This is a new managed table。

3. 修改数据表列

修改数据表列是指修改数据表中列的名称、描述、数据类型或者列的位置，语法格式如下。

```
ALTER TABLE table_name CHANGE [COLUMN] col_old_name col_new_name column_type [COMMENT col_comment] [FIRST|AFTER column_name];
```

上述语法的具体讲解如下。

- ALTER TABLE：表示修改数据表结构信息的语句。
- CHANGE [COLUMN]：用于修改数据表列的操作，其中 COLUMN 为可选。
- table_name：指定需要修改列的数据表名称。
- col_old_name：指定需要修改的列名。
- col_new_name：指定列名被修改后的新列名。
- column_type：指定列修改后的字段类型。需要注意的是，指定列修改后的字段类型与原始字段类型之间需要符合 Hive 的强制类型转换规则。
- COMMENT col_comment：可选，修改列的描述。
- FIRST|AFTER column_name：可选，指定列修改后的位置。FIRST 表示在第一列，AFTER 表示在 column_name 之后，其中 column_name 表示已经存在的列。需要注意的是，如果表中每个列的数据类型不一致，则无法使用该功能。

接下来，在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline，远程连接虚拟机 Node_02 的 HiveServer2 服务操作 Hive，讲解修改数据表列语法的实际应用，具体操作步骤如下。

（1）为了便于演示修改列位置及列类型的操作，这里在数据库 hive_database 中创建数据表 alter_managed_table，该表中所有列的数据类型一致，具体命令如下。

```
CREATE TABLE IF NOT EXISTS alter_managed_table(
id STRING,
sex STRING,
name STRING);
```

（2）修改数据库 hive_database 中数据表 alter_managed_table 的列 sex，首先重命名列 sex 为 gender，然后移动列 gender 的位置到列 name 的后边，最后修改列 gender 的描述为“This is gender”，具体命令如下。

```
ALTER TABLE alter_managed_table CHANGE sex gender STRING COMMENT "This is gender" AFTER name;
```

（3）修改数据库 hive_database 中内部表 alter_managed_table 的列 gender，将列的数据类型由 STRING 修改为 VARCHAR(30)，具体命令如下。

```
ALTER TABLE alter_managed_table CHANGE gender gender VARCHAR(30);
```

（4）在 Hive 客户端工具 Beeline 中执行“DESC alter_managed_table;”命令，查看数据库 hive_database 中数据表 alter_managed_table 表结构的基本信息，如图 3-12 所示。

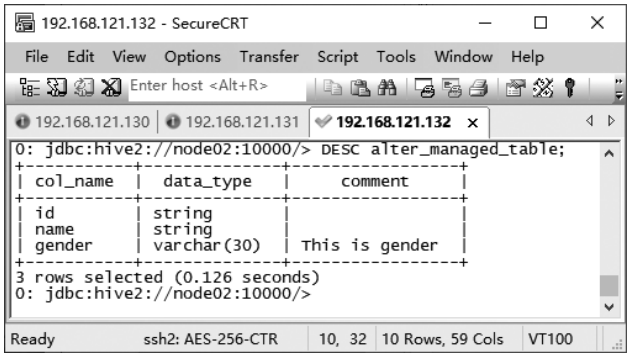

图 3-12　查看数据表 alter_managed_table 表结构的基本信息（1）

从图 3-12 可以看出，数据表 alter_managed_table 的列 sex 重命名为 gender，数据类型由 string 修改为 varchar(30)，位置由列 name 之前移动到了列 name 之后，由无描述修改描述为 This is gender。

4. 添加数据表列

添加数据表列会在数据表的尾部添加指定列，语法格式如下。

```
ALTER TABLE table_name ADD COLUMNS (col_name data_type [COMMENT col_comment], ...)
```

上述语法的具体讲解如下。

- ALTER TABLE：表示修改数据表结构信息的语句。
- ADD COLUMNS：用于向数据表中添加列。
- table_name：指定需要添加列的数据表名称。
- col_name：指定需要添加的列名。
- data_type：指定需要添加列的数据类型。
- COMMENT col_comment：指定需要添加列的描述。

接下来，在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline，远程连接虚拟机 Node_02 的 HiveServer2 服务操作 Hive，在数据表 alter_managed_table 中添加列，具体命令如下。

```
ALTER TABLE alter_managed_table ADD COLUMNS (age INT COMMENT "This is age",phone STRING COMMENT "This is phone");
```

上述命令中，在数据表 alter_managed_table 中添加列 age，指定列 age 的数据类型为 INT 并指定列描述为“This is age”；在数据表 alter_managed_table 中添加列 phone，指定列 phone 的数据类型为 STRING 并指定列描述为“This is phone”。

上述命令执行完成后，在 Hive 客户端工具 Beeline 中执行“DESC alter_managed_table;”命令，查看数据库 hive_database 中数据表 alter_managed_table 表结构的基本信息，如图 3-13 所示。

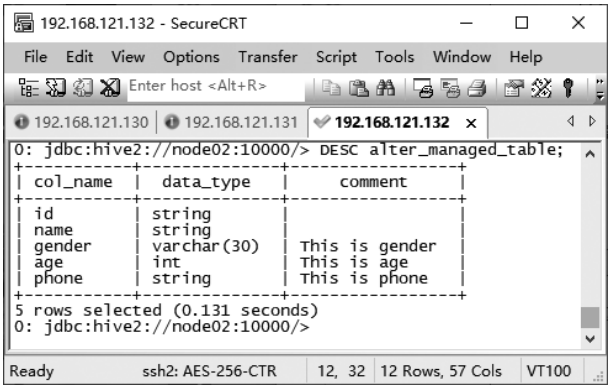

图 3-13　查看数据表 alter_managed_table 表结构的基本信息(2)

从图 3-13 可以看出，内部表 alter_managed_table 由原始的 3 列增加为 5 列，新增的 2 列分别是列 age 和列 phone。

5. 替换数据表列

替换数据表列是指替换当前数据表中的所有列，语法格式如下。

```
ALTER TABLE table_name REPLACE COLUMNS (col_name data_type [COMMENT col_comment], ...)
```

上述语法的具体讲解如下。

- ALTER TABLE：表示修改数据表结构信息的语句。

- REPLACE COLUMNS：用于替换数据表中已存在的所有列。
- table_name：指定需要替换列的数据表名称。
- col_name：指定替换列的列名。
- data_type：指定替换列的数据类型。
- COMMENT col_comment：指定替换列的描述。

接下来，在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline，远程连接虚拟机 Node_02 的 HiveServer2 服务操作 Hive，替换数据表 alter_managed_table 的列，具体命令如下。

```
ALTER TABLE alter_managed_table REPLACE COLUMNS (username STRING COMMENT "This is username",
password STRING COMMENT "This is password");
```

上述命令中，将数据表 alter_managed_table 中的所有列替换为列 username 和列 password。

上述命令执行完成后，在 Hive 客户端工具 Beeline 中执行“DESC alter_managed_table;”命令，查看数据库 hive_database 中数据表 alter_managed_table 表结构的基本信息，如图 3-14 所示。

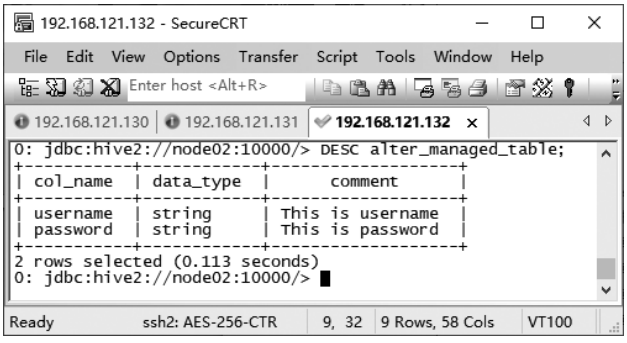

图 3-14　查看数据表 alter_managed_table 表结构的基本信息(3)

从图 3-14 可以看出，数据表 alter_managed_table 中只包含两列，即列 username 和列 password。

多学一招：Hive 中强制数据类型转换规则

在 Hive 中执行修改列的数据类型操作时只能按照强制数据类型转换规则进行修改，Hive 的强制数据类型转换规则如图 3-15 所示。

	boolean	tinyint	smallint	int	bigint	float	double	decimal	string	varchar	timestamp	date	binary
boolean	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
tinyint	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
smallint	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
int	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
bigint	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
float	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
double	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
decimal	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
string	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
varchar	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
timestamp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE
date	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE
binary	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

图 3-15　Hive 的强制数据类型转换规则

从图 3-15 可以看出，string 类型数据可以转换为 double、decimal、string 和 varchar 这 4 种数据类型。

3.2.6 删除数据表

删除数据表的语法格式如下。

```
DROP TABLE [IF EXISTS] table_name [PURGE];
```

上述语法的具体讲解如下。

- DROP TABLE：表示删除数据表的语句。
- IF EXISTS：可选，用于判断要删除的数据表是否存在。
- table_name：表示要删除的数据表名称。
- PURGE：可选，当删除内部表时，若使用 PURGE，则内部表的数据不会放入回收站，后续无法通过回收站恢复内部表的数据，反之，内部表的数据会放入回收站，这里指的恢复数据不包含元数据，元数据删除后无法恢复。

接下来，在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline，远程连接虚拟机 Node_02 的 HiveServer2 服务操作 Hive，删除数据库 hive_database 的数据表 alter_managed_table，具体命令如下。

```
DROP TABLE IF EXISTS alter_managed_table PURGE;
```

上述命令中，通过 PURGE 子句指定数据表 alter_managed_table 删除后的数据不放入回收站。

上述命令执行完成后，在 Hive 客户端工具 Beeline 中执行“SHOW TABLES;”命令，查看数据库 hive_database 中的所有数据表，如图 3-16 所示。

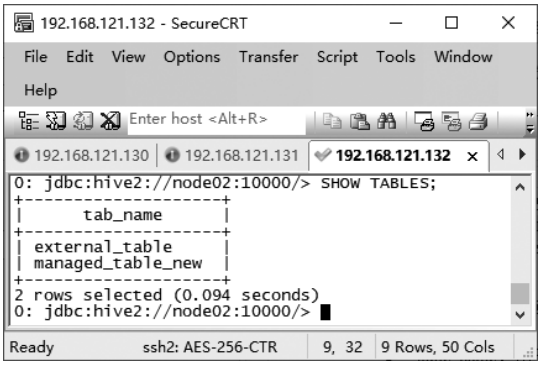

图 3-16 查看数据库 hive_database 中的所有数据表

从图 3-16 可以看出，数据库 hive_database 中只有数据表 external_table 和 managed_table_new，说明数据表 alter_managed_table 成功从数据库 hive_database 中删除了。

多学一招：回收站

Hive 中的回收站是通过 HDFS 的 Trash 功能实现，Trash 功能可以将 HDFS 中删除的

文件放入回收站目录(默认回收站目录/user/root/.Trash/Current,其中回收站目录中的root 会根据当前操作 HDFS 的用户名而变化),防止用户意外删除文件,出现无法找回的情况。Hive 内部表的数据存放在 HDFS 中,并且删除内部表时数据也会一同被删除,所以为了防止用户意外删除 Hive 内部表造成数据丢失的情况,可以在删除内部表的语句中指定 PURGE,将删除的内部表数据放入回收站目录,后续复制回收站目录中删除的内部表数据即可。

HDFS 默认情况下并没有开启 Trash 功能,需要在 Hadoop 的配置文件 core-site.xml 的＜configuration/＞标签中添加如下配置内容。

```
<property>
    <name>fs.trash.interval</name>
    <value>1440</value>
</property>
<property>
    <name>fs.trash.checkpoint.interval</name>
    <value>60</value>
</property>
```

上述配置内容中,参数 fs.trash.interval 表示回收站目录中文件保存的时间,该参数的默认值为 0(分钟),也就是不保存,这里指定参数值为 1440,也就是被删除的文件会在回收站目录保存一天;参数 fs.trash.checkpoint.interval 表示 NameNode 检查回收站目录间隔的时长,这里指定参数值为 60,也就是 NameNode 每间隔一小时检查一次回收站目录,永久删除回收站目录中存放时长超过一天的文件。

在 3 台虚拟机 Node_01、Node_02 和 Node_03 的 Hadoop 配置文件 core-site.xml 中分别添加上述内容,添加完成后需要重新启动 Hadoop 集群使配置内容生效。

3.3　分区表

随着系统运行时间的增加,表的数据量会越来越大,而 Hive 查询数据通常是使用全表扫描,这会导致对大量不必要数据的扫描,从而降低查询效率。为了解决这一问题,Hive 引进了分区技术,分区主要是将表的整体数据根据业务需求,划分成多个子目录进行存储,每个子目录对应一个分区。通过扫描分区表中指定分区的数据,避免 Hive 全表扫描,从而提升 Hive 查询数据的效率。本节针对 Hive 的分区表进行详细讲解。

3.3.1　创建分区表

由于分区表是基于内/外部表创建,所以分区表的创建方式和创建数据表的方式类似。

接下来,在虚拟机 Node_03 中使用 Hive 客户端工具 Beeline,远程连接虚拟机 Node_02 的 HiveServer2 服务操作 Hive,在数据库 hive_database 中创建分区表 partitioned_table,具体命令如下。

```
CREATE TABLE IF NOT EXISTS
hive_database.partitioned_table(
```