第 3 章 HBase的Shell操作

学习目标

- 熟悉命名空间操作，能够使用 HBase Shell 对命名空间进行创建、查看、删除等操作。
- 掌握表操作，能够使用 HBase Shell 对表进行创建、查看、删除等操作。
- 掌握数据操作，能够使用 HBase Shell 对数据进行插入、查询、删除等操作。

HBase 自身提供了 Shell 命令行工具 HBase Shell，它可以对命名空间(namespace)、表和数据进行操作。本章以操作完全分布式模式部署的 HBase 为例，演示如何使用 HBase Shell 操作 HBase。

3.1 运行 HBase Shell

HBase Shell 可以在 HBase 集群的任意节点运行，默认情况下，只需要进入 HBase 的安装目录执行“bin/hbase shell”命令即可，如果在运行 HBase Shell 的节点配置了 HBase 的系统环境变量，那么也可以直接在任意目录执行 hbase shell 命令运行 HBase Shell。

接下来以虚拟机 HBase01 为例演示如何运行 HBase Shell。在虚拟机 HBase01 执行 hbase shell 命令运行 HBase Shell，成功运行 HBase Shell 的效果如图 3-1 所示。

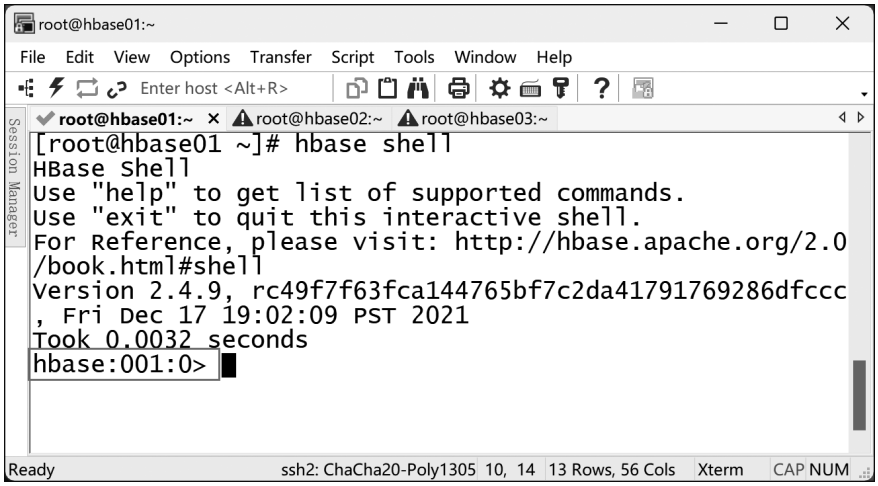

图 3-1 成功运行 HBase Shell 的效果

可以在图 3-1 的“hbase:001:0>”位置输入 HBase Shell 提供的命令来操作命名空间、

表和数据。如果要关闭 HBase Shell，在 HBase Shell 执行 exit 命令即可。

3.2　命名空间操作

命名空间的作用是将相关的表组织到一起，方便用户对表进行管理和维护，在 HBase 中每个表都必须属于一个命名空间。本节详细介绍如何通过 HBase Shell 操作 HBase 的命名空间。

3.2.1　查看命名空间

HBase Shell 提供了 list_namespace 命令用于查看命名空间，该命令可以列出所有命名空间，如果想要对查看的命名空间进行筛选，可以在查看命名空间时使用正则表达式匹配命名空间的名称。查看命名空间的语法格式如下。

```
list_namespace ['regular']
```

上述语法格式中，regular 为可选，用于指定正则表达式。为了方便讲解后续知识，接下来分步骤演示如何查看命名空间，具体步骤如下。

1. 启动 HBase

根据本书第 2 章的相关操作，在虚拟机 HBase01、HBase02 和 HBase03 启动 Hadoop、ZooKeeper 和完全分布式模式部署的 HBase。

2. 运行 HBase Shell

在虚拟机 HBase01 执行 hbase shell 命令运行 HBase Shell。

3. 查看命名空间

在 HBase Shell 执行如下命令查看命名空间。

```
>list_namespace
```

上述命令的执行效果如图 3-2 所示。

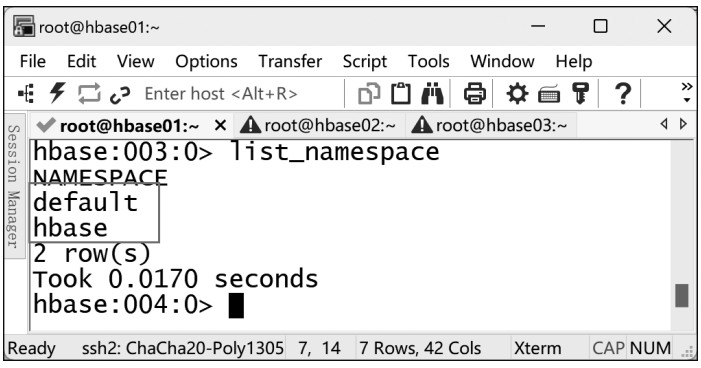

图 3-2　查看命名空间(1)

从图 3-2 可以看出，HBase 默认预留了 default 和 hbase 两个命名空间，其中 default 是 HBase 默认的命名空间，如果在创建表时没有指定命名空间，则默认将表创建到命名空间 default。hbase 存储了 HBase 的系统表，系统表是 HBase 内部使用的表，用于存储 HBase

的元数据信息。

注意：命名空间 hbase 存储了 HBase 的系统表，为了避免 HBase 的运行出现异常，建议用户不要对命名空间 hbase 进行操作。

多学一招：匹配特定命名空间

在 list_namespace 命令中，可以通过指定正则表达式来匹配特定的命名空间。例如，通过正则表达式匹配名称以字母 h 开头的命名空间，可以在 HBase Shell 执行如下命令。

```
>list_namespace 'h.*'
```

上述命令的执行效果如图 3-3 所示。

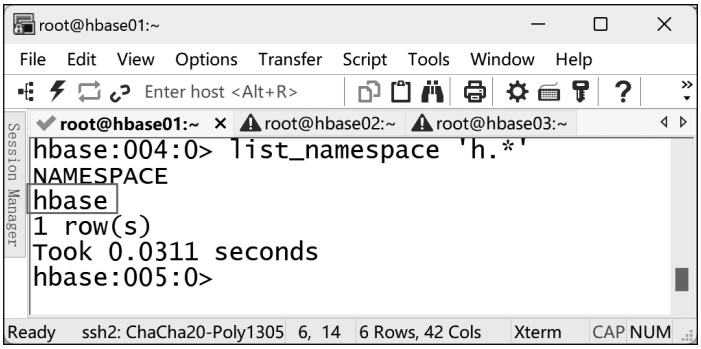

图 3-3 匹配特定命名空间

从图 3-3 可以看出，HBase 仅存在一个名称以字母 h 开头的命名空间 hbase。

3.2.2 创建命名空间

HBase Shell 提供了 create_namespace 命令用于创建命名空间，在该命令中可以通过指定属性来描述或者配置命名空间。命名空间的属性分为自定义属性和预定义属性两种类型，其中自定义属性的名称和属性值由用户定义，主要用于描述命名空间，如指定命名空间的描述信息；预定义属性的名称和属性值由 HBase 定义，主要用于配置命名空间，如限制命名空间内表的数量。

创建命名空间的语法格式如下。

```
create_namespace 'ns'[,{'PROPERTY_NAME'=>'PROPERTY_VALUE'},...]
```

上述语法格式中，ns 用于指定命名空间的名称。{'PROPERTY_NAME'=>'PROPERTY_VALUE'}为可选，用于指定命名空间的属性(PROPERTY_NAME)和属性值(PROPERTY_VALUE)。

接下来演示如何创建命名空间 itcast，并为命名空间指定自定义属性 describe 和属性值 This is my first namespace，在 HBase Shell 执行如下命令。

```
>create_namespace 'itcast',{'describe'=>'This is my first namespace'}
```

上述命令执行完成后，查看命名空间，如图 3-4 所示。

从图 3-4 可以看出，HBase 存在名称为 itcast 的命名空间，说明已成功创建命名空间

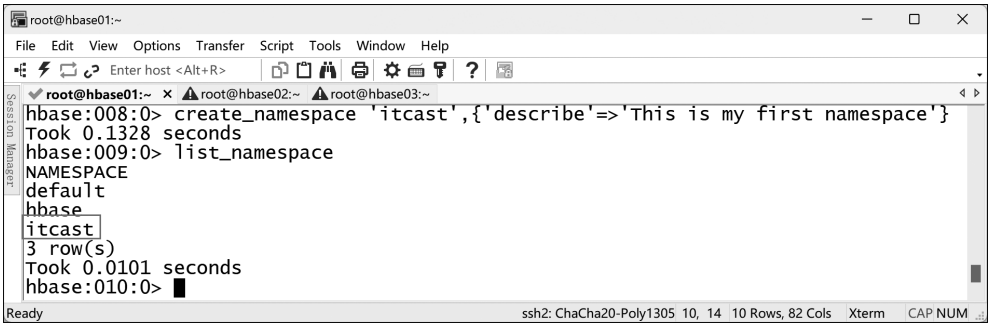

图 3-4　查看命名空间(2)

itcast。

多学一招：命名空间的预定义属性

HBase 定义了多种命名空间的预定义属性，关于命名空间常用的预定义属性如下。

- hbase.namespace.quota.maxtables：用于限制命名空间内表的数量，默认情况下不做限制，如限制命名空间内表的数量为 2，那么可以指定属性值为 2。
- hbase.namespace.quota.maxregions：用于限制命名空间占用 Region 的数量，默认情况下不做限制，如限制命名空间占用 Region 的数量为 10，那么可以指定属性值为 10。

需要说明的是，在创建命名空间时通过上述两个预定义属性限制表和 Region 的数量时，需要开启 HBase 的 Quota 功能，该功能的开始方式是在 HBase 集群所有节点的配置文件 hbase-site.xml 添加如下内容。

```
<property>
  <name>hbase.quota.enabled</name>
  <value>true</value>
</property>
```

上述内容添加完成后，还需要重新启动 HBase 使配置内容生效。

3.2.3　查看命名空间属性

HBase Shell 提供了 describe_namespace 命令用于查看命名空间属性，其语法格式如下。

```
describe_namespace 'ns'
```

接下来演示如何查看命名空间 itcast 的属性，在 HBase Shell 执行如下命令。

```
>describe_namespace 'itcast'
```

上述命令的执行效果如图 3-5 所示。

从图 3-5 可以看出，命名空间 itcast 包含 NAME 和 describe 两个属性，其中 NAME 为命名空间默认的属性，用于标识命名空间的名称；describe 为创建命名空间 itcast 时指定的自定义属性。

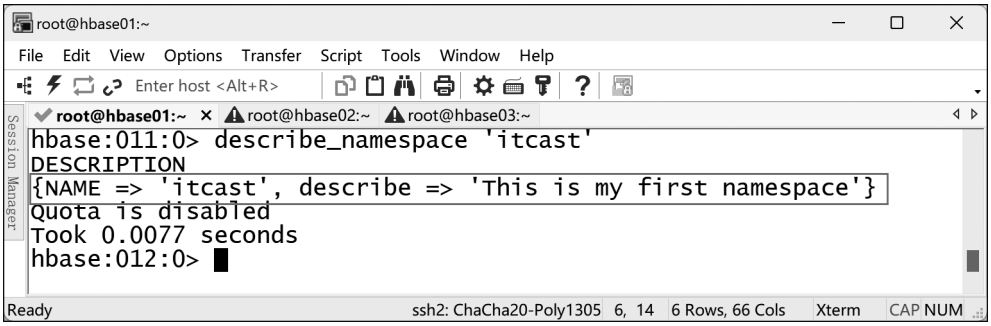

图 3-5 查看命名空间 itcast 的属性(1)

3.2.4 修改命名空间

HBase Shell 提供了 alter_namespace 命令用于添加或删除命名空间的属性,具体内容如下。

1. 添加属性

为命名空间添加属性的语法格式如下。

```
alter_namespace 'ns',{METHOD =>'set','PROPERTY_NAME' =>'PROPERTY_VALUE'}
```

上述语法格式中,METHOD => 'set'表示添加属性。需要说明的是,如果添加的属性在命名空间已存在,那么会根据指定的属性值来修改该属性的属性值。

接下来演示如何为命名空间 itcast 添加属性,具体内容如下。

(1) 为命名空间 itcast 添加自定义属性 create_user,并指定属性值为 bozai,在 HBase Shell 执行如下命令。

```
>alter_namespace 'itcast',{METHOD =>'set','create_user' =>'bozai'}
```

上述命令执行完成后,查看命名空间 itcast 属性的效果如图 3-6 所示。

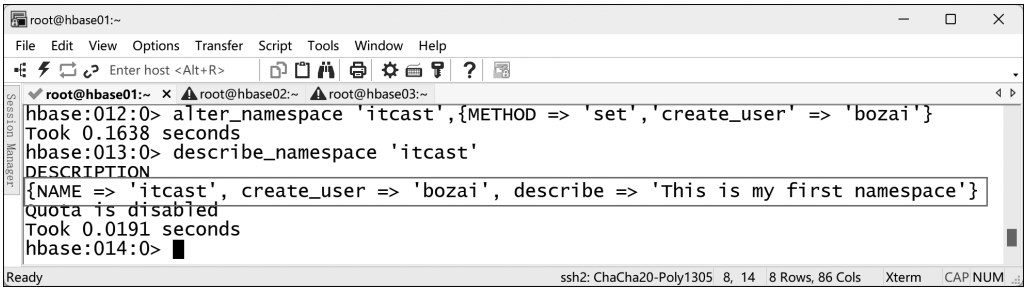

图 3-6 查看命名空间 itcast 的属性(2)

从图 3-6 可以看出,命名空间 itcast 包含属性 create_user,并且该属性的属性值为 bozai,因此说明已成功为命名空间 itcast 添加属性。

(2) 为命名空间 itcast 添加自定义属性 describe,并指定属性值为 This namespace is modified,在 HBase Shell 执行如下命令。

```
    >alter_namespace 'itcast',{METHOD =>'set',
'describe' =>'This namespace is modified'}
```

上述命令执行完成后，查看命名空间 itcast 的属性，如图 3-7 所示。

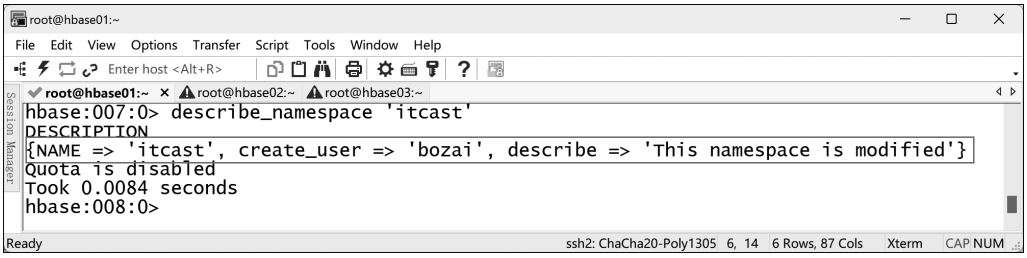

图 3-7　查看命名空间 itcast 的属性(3)

从图 3-7 可以看出，由于属性 describe 已经存在于命名空间 itcast，所以在执行上述命令时，会将该属性的属性值由 This is my first namespace 修改为 This namespace is modified。因此说明已成功为命名空间 itcast 添加属性，并且将相同属性的属性值进行了修改。

2. 删除属性

为命名空间删除属性的语法格式如下。

```
alter_namespace 'ns',{METHOD =>'unset',NAME=>'PROPERTY_NAME'}
```

上述语法格式中，ns 用于指定命名空间的名称。METHOD => 'unset'表示删除属性的固定语法。PROPERTY_NAME 用于指定删除属性的名称，该属性必须已存在。

接下来演示如何删除命名空间 itcast 的属性 describe，在 HBase Shell 执行如下命令。

```
>alter_namespace 'itcast',{METHOD =>'unset', NAME =>'describe'}
```

上述命令执行完成后，查看命名空间 itcast 的属性，如图 3-8 所示。

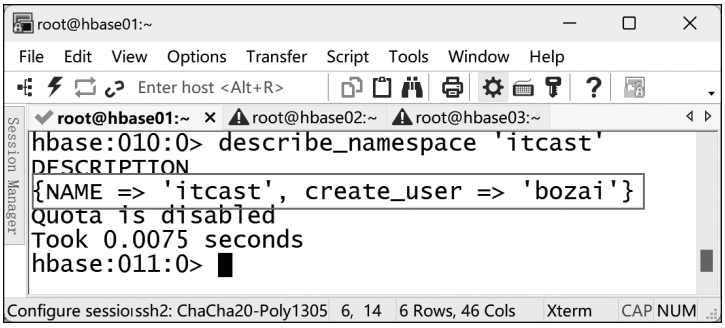

图 3-8　查看命名空间 itcast 的属性(4)

从图 3-8 可以看出，命名空间 itcast 的描述信息中已经不存在属性 describe。因此说明已成功删除命名空间 itcast 的属性。

注意：命名空间默认的属性 NAME 是不能修改的。

3.2.5　删除命名空间

HBase Shell 提供了 drop_namespace 命令用于删除命名空间，其语法格式如下。

```
drop_namespace 'ns'
```

接下来演示如何删除命名空间 itcast，在 HBase Shell 执行如下命令。

```
>drop_namespace 'itcast'
```

上述命令执行完成后，查看命名空间，如图 3-9 所示。

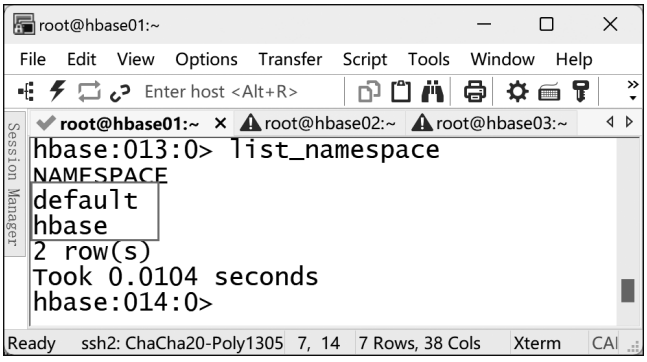

图 3-9 查看命名空间(3)

从图 3-9 可以看出，HBase 已经不存在命名空间 itcast。因此说明已成功删除命名空间 itcast。

注意：如果命名空间包含表，那么需要将命名空间的表删除之后才能删除命名空间。

3.2.6 查看命名空间的表

HBase Shell 提供了 list_namespace_tables 命令用于查看命名空间的表，其语法格式如下。

```
list_namespace_tables 'ns'
```

接下来演示如何查看命名空间 hbase 的表，在 HBase Shell 执行如下命令。

```
>list_namespace_tables 'hbase'
```

上述命令的执行效果如图 3-10 所示。

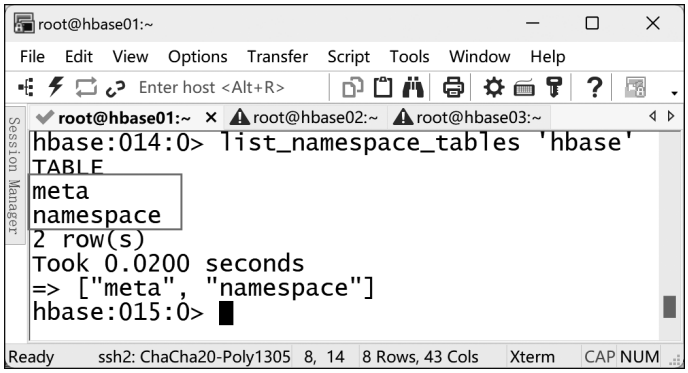

图 3-10 查看命名空间 hbase 的表

从图 3-10 可以看出，命名空间 hbase 包含表 meta 和 namespace，其中，meta 用于存储表的元数据；namespace 用于存储命名空间的元数据。

3.3　表操作

表的作用是将相同类型或者同一业务的数据组织在一起，方便用户对数据进行管理和维护。本节详细介绍如何通过 HBase Shell 操作 HBase 的表。

3.3.1　创建表

在创建表时，必须为表指定至少一个列族，并且可以通过指定属性来配置表和列族。HBase 为表和列族定义了丰富的预定义属性，供用户在创建表时根据实际需求对表和列族进行配置，关于表和列族常用的预定义属性如表 3-1 和表 3-2 所示。

表 3-1　表常用的预定义属性

预定义属性	含　义	示　例
SPLITS_FILE	用于根据用户指定的拆分文件对表进行预拆分	SPLITS_FILE => 'splits.txt'
SPLITS	用于根据用户指定的数组对表进行预拆分	SPLITS => ['10', '20', '30', '40']
SPLITALGO	用于根据用户指定的拆分算法对表进行预拆分，须配合预定义属性 NUMREGIONS 一同使用。支持的拆分算法有 HexStringSplit、DecimalStringSplit 和 UniformSplit	SPLITALGO => 'HexStringSplit'
NUMREGIONS	用于指定预拆分的数量，须配合预定义属性 SPLITALGO 一同使用	NUMREGIONS => 15
METADATA	用于根据用户指定的自定义属性和属性值来描述表	METADATA => { 'mykey1' => 'myvalue1', 'mykey2' => 'myvalue2'}
OWNER	用于指定表的所有者，表默认的所有者为创建该表的用户	OWNER => 'itcast'
SPLIT_ ENABLED	用于开启或关闭 Region 自动拆分，默认为开启。若该预定义属性的属性值为 false 表示关闭	SPLIT_ENABLED => 'false'
MERGE_ ENABLED	用于开启或关闭 Region 自动合并，默认为开启。若该预定义属性的属性值为 false 表示关闭	MERGE_ENABLED => 'false'
READONLY	用于指定表是否为只读模式，当表为只读模式时无法写入数据，默认为关闭。若该预定义属性的属性值为 true 表示开启	READONLY => 'true'

表 3-2　列族常用的预定义属性

预定义属性	含　义	示　例
BLOOMFILTER	用于指定列族中布隆过滤器的工作模式，默认属性值为 ROW(行模式)，可选属性值为 NONE(关闭)和 ROWCOL(行列模式)	BLOOMFILTER => 'NONE'
IN_MEMORY	用于将列族存储的数据缓存到内存，提高数据的读取效率，默认属性值为 false，表示关闭	IN_MEMORY => 'true'

续表

预定义属性	含　义	示　例
VERSIONS	用于指定列族存储数据的最大版本数，若数据的版本数超出最大版本数，则 Region 在合并过程中会根据版本的新旧程度删除相对较旧版本的数据，以确保数据的版本数量不超出最大版本数。默认属性值为 1，即列族只存储最新版本的数据	VERSIONS => 2
KEEP_DELETED_CELLS	用于指定是否保存列族中已删除的数据，默认属性值为 false，表示不保存列族中已删除的数据，此时 HBase 会定期自动清除已删除的数据。若属性值为 true，那么会根据预定义属性 TTL 指定的时间来清除已删除的数据	KEEP_DELETED_CELLS => 'true'
DATA_BLOCK_ENCODING	用于指定列族中数据的编码方式，默认属性值为 NONE，即不对列族中的数据进行编码处理。可选属性值包括 PREFIX、DIFF、FAST_DIFF 和 ROW_INDEX_V1	DATA_BLOCK_ENCODING => 'FAST_DIFF'
COMPRESSION	用于指定列族中数据的压缩格式，默认属性值为 NONE，即不对列族中的数据进行压缩。可选属性值包括 LZO、GZ、SNAPPY 和 LZ4	COMPRESSION=> 'SNAPPY'
TTL	用于指定列族中数据的生存时间，时间单位为 SECONDS（秒），当数据自写入后超过指定的生存时间时，会被 HBase 自动清除。默认属性值为 FOREVER，即列族中的数据永久存在	TTL => '10 SECONDS'
MIN_VERSIONS	用于指定列族中数据的最小版本数，主要用于约束属性 TTL，如果指定了列族中数据的生存时间，那么当数据超过生存时间时，在 HBase 自动清除数据之前会先判断当前数据在列族中的版本数是否小于或等于最小版本数，如果是就放弃清除数据。默认属性值为 0	MIN_VERSIONS=>1
BLOCKCACHE	用于设置列族是否开启 BlockCache，默认属性值为 true，即开启 BlockCache	BLOCKCACHE => 'false'
BLOCKSIZE	用于指定 HFile 中每个数据块的大小，默认属性值为 65536(64KB)	BLOCKSIZE => 131072
REPLICATION_SCOPE	用于定义数据的复制范围，复制类似于 HDFS 的副本机制，主要是为了提高数据的可靠性。该参数的可选属性值为 0、1 和 2，其中 0 为默认属性值，表示不进行数据复制；1 表示数据只会在同一数据中心进行复制；2 表示数据可以在不同数据中心进行复制	REPLICATION _ SCOPE => 1

在表 3-1 中关于预拆分、Region 自动拆分和 Region 合并的内容会在本书的第 6 章进行深入讲解，这里读者仅需了解即可。

在表 3-2 中，布隆过滤器（Bloom filter）是一种基于概率的数据结构，在 HBase 中每个列族都可以有一个布隆过滤器，布隆过滤器会为列族的数据创建索引，当客户端执行读操作时，布隆过滤器会先通过索引查询数据是否存在，如果存在则继续查询对应的数据，否则直接返回不存在。需要说明的是，如果预定义属性的属性值为数字类型，那么可以不使用单引号进行修饰。

HBase Shell 提供了 create 命令用于创建表，根据创建表时是否指定列族的属性，可以将创建表的方式分为两种，具体内容如下。

1. 创建表时指定列族的属性

在创建表时，可以通过指定列族的属性，将预定义属性的默认属性值根据实际需求进行修改，其语法格式如下。

```
create '[namespace:]table_name',
{NAME =>'columnfamily',CF_PROPERTY_NAME=>'CF_PROPERTY_VALUE',...},
{NAME =>'columnfamily',CF_PROPERTY_NAME=>'CF_PROPERTY_VALUE',...},...
[,TABLE_PROPERTY_NAME =>'TABLE_PROPERTY_VALUE',
TABLE_PROPERTY_NAME =>'TABLE_PROPERTY_VALUE',...]
```

上述语法格式中，namespace 为可选，用于指定表所属的命名空间。如果没有指定命名空间，那么将使用默认的命名空间 default。table_name 用于指定表的名称，columnfamily 用于指定列族的名称。

CF_PROPERTY_NAME 和 CF_PROPERTY_VALUE 用于根据 HBase 为列族定义的预定义属性来指定列族的属性及其属性值，可以同时指定列族的多个属性。

TABLE_PROPERTY_NAME 和 TABLE_PROPERTY_VALUE 为可选，用于根据 HBase 为表定义的预定义属性指定表的属性及其属性值，可以同时指定表的多个属性。

接下来演示如何在创建表时指定列族的属性，具体需求如下。

- 在命名空间 school 创建表 teacher_info，指定表的列族为 grade1 和 grade2。
- 通过表的预定义属性 METADATA，为表指定自定义属性 comment 和属性值 Class of 2022 Teacher List。
- 通过表的预定义属性 MERGE_ENABLED 关闭 Region 自动合并。
- 通过列族的预定义属性 VERSIONS，将列族 grade1 存储数据的最大版本数调整为 2。
- 通过列族的预定义属性 IN_MEMORY，将列族 grade1 存储的数据缓存到内存。
- 通过列族的预定义属性 TTL，将列族 grade2 中数据的生存时间调整为 10 秒。
- 通过列族的预定义属性 MIN_VERSIONS，将列族 grade2 存储数据的最小版本数调整为 1。

根据上述需求，在 HBase Shell 执行下列命令。

```
    # 创建命名空间 school
    >create_namespace 'school'
    # 在命名空间 school 创建表 teacher_info
    >create 'school:teacher_info',{NAME =>'grade1',VERSIONS =>2,
IN_MEMORY =>'true'},{NAME =>'grade2',TTL =>'10 SECONDS',
MIN_VERSIONS =>1},MERGE_ENABLED =>'false',
METADATA =>{'comment' =>'Class of 2022 Teacher List'}
```

上述命令执行完成后，查看命名空间 school 的表，如图 3-11 所示。

从图 3-11 可以看出，命名空间 school 包含表 teacher_info。因此说明已成功在命名空间 school 创建表 teacher_info。

2. 创建表时不指定列族的属性

出于便捷性考虑，可以在创建表时不指定列族的属性，使列族直接应用预定义属性默认

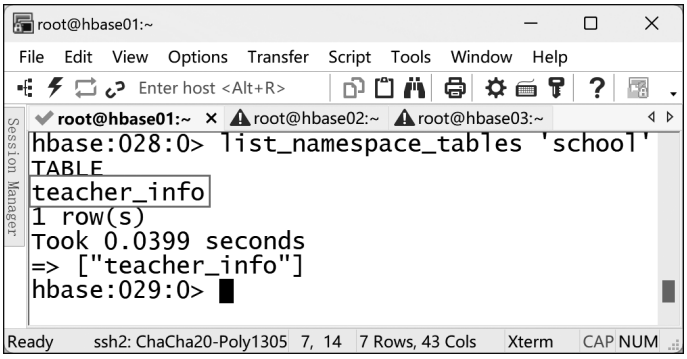

图 3-11 查看命名空间 school 的表

的属性值，其语法格式如下。

```
create '[namespace:]table_name','columnfamily','columnfamily',...
[,'TABLE_PROPERTY_NAME' =>'TABLE_PROPERTY_VALUE',...]
```

接下来演示如何在创建表时不指定列族的属性。这里在命名空间 default 创建表 user_info，指定表的列族为 person 和 address，在 HBase Shell 执行如下命令。

```
>create 'user_info','person','address'
```

上述命令执行完成后，查看命名空间 default 的表，如图 3-12 所示。

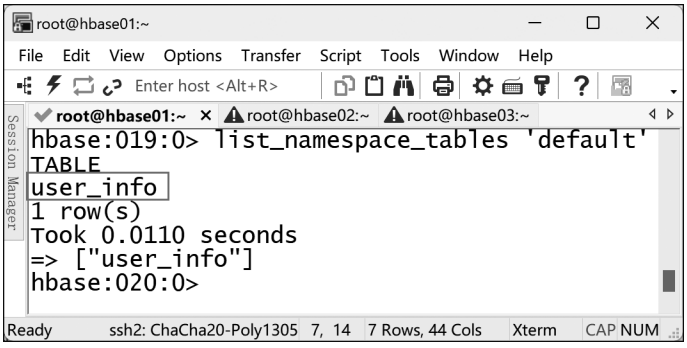

图 3-12 查看命名空间 default 的表

从图 3-12 可以看出，命名空间 default 包含表 user_info。因此说明已成功在命名空间 default 创建表 user_info。

多学一招：克隆表

除了通过 HBase Shell 提供的 create 命令创建表之外，HBase Shell 还提供了 clone_table_schema 命令，可以基于已存在的表克隆一个新表，从而实现创建表的目的。新表与被克隆的表具有相同的元数据，但不包含被克隆表的数据。关于克隆表的语法格式如下。

```
clone_table_schema '[namespace:]table_name','[namespace:]new_table_name'
[,'split_key']
```

上述语法格式中，table_name 用于指定被克隆的表名；new_table_name 用于指定新表的表名；split_key 为可选，用于指定新表是否包含被克隆表的 Region 拆分信息，默认值为

true，表示包含。

例如，通过克隆命名空间 school 的表 teacher_info，在命名空间 default 创建表 student_info，具体命令如下。

```
>clone_table_schema 'school:teacher_info','student_info'
```

3.3.2　查看表信息

HBase Shell 提供了 desc 命令用于查看表信息，表信息的内容包含表的状态信息，以及表和列族的属性信息。关于查看表信息的语法格式如下。

```
desc '[namespace:]table_name'
```

接下来演示如何查看命名空间 school 中表 teacher_info 的信息，在 HBase Shell 执行如下命令。

```
>desc 'school:teacher_info'
```

上述命令的执行效果如图 3-13 所示。

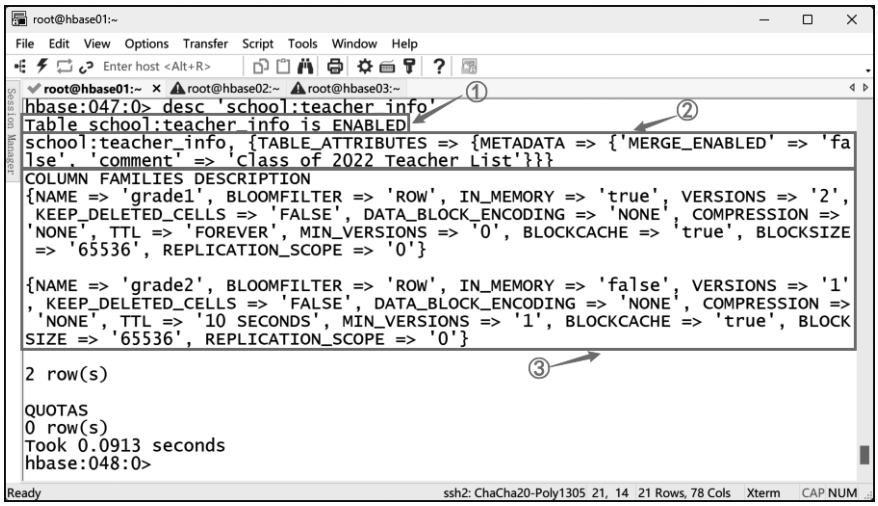

图 3-13　查看命名空间 school 中表 teacher_info 的信息(1)

在图 3-13 中，①标注的部分为表的状态信息，其中状态信息中的 ENABLED 表示表处于启用状态，若状态信息中出现 DISABLE 表示表处于停用状态，有关停用和启用表的相关内容会在 3.3.4 节进行讲解。②标注的部分为表的属性信息。③标注的部分为列族的属性信息，其中属性 NAME 用于标注列族名称，其他属性为 HBase 为列族定义的预定义属性。

3.3.3　查看表

HBase Shell 提供了 list 命令用于查看表，该命令可以列出用户创建的所有表，也可以使用正则表达式对用户创建的所有表进行筛选。关于查看表的语法格式如下。

```
list '[[namespace:]regular]'
```

上述语法格式中，[[namespace:]regular]为可选，用于通过正则表达式对指定命名空

间中用户创建的所有表进行筛选，其中 namespace 为可选，用于指定命名空间，如果不指定命名空间，那么正则表达式会对用户创建的所有表进行筛选；regular 用于指定正则表达式。

需要说明的是，在查看表的结果中，除了属于命名空间 default 的表之外，其他命名空间中表的展现形式为“命名空间：表”，例如，命名空间 school 中表 teacher_info 的展现形式为 school:teacher_info。

此外，使用正则表达式对用户创建的所有表进行筛选时，只有命名空间 default 的表是基于表名与正则表达式进行匹配，而其他命名空间的表则基于“命名空间：表”的形式与正则表达式进行匹配。

接下来分别演示如何查看用户创建的所有表，以及如何使用正则表达式对用户创建的表进行筛选，具体内容如下。

（1）查看用户创建的所有表，在 HBase Shell 执行如下命令。

```
>list
```

上述命令的执行效果如图 3-14 所示。

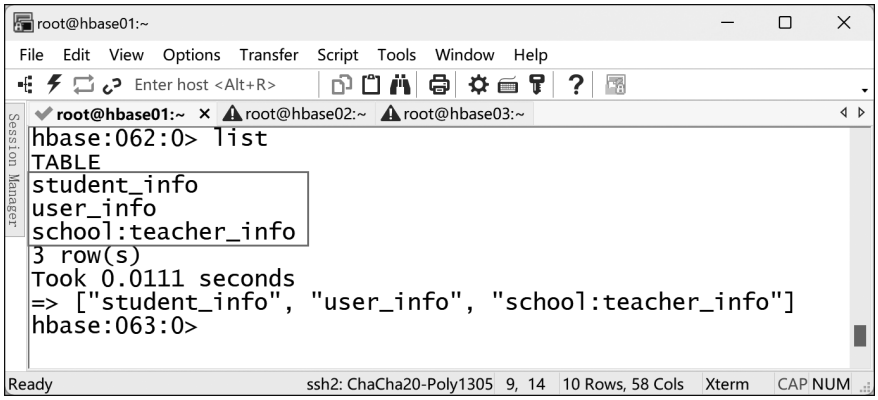

图 3-14 查看表（1）

从图 3-14 可以看出，用户共创建了 3 个表，其中表 student_info 和 user_info 属于命名空间 default，表 teacher_info 属于命名空间 school。

（2）通过正则表达式“.*:t.*”对用户创建的所有表进行筛选，匹配表名以字母 t 开头的表，在 HBase Shell 执行如下命令。

```
>list '.*:t.*'
```

上述命令的执行效果如图 3-15 所示。

从图 3-15 可以看出，用户创建了一个表名以字母 t 开头的表 teacher_info，该表属于命名空间 school。

（3）通过正则表达式“u.*”对命名空间 default 中用户创建的所有表进行筛选，匹配表名以字母 u 开头的表，在 HBase Shell 执行如下命令。

```
>list 'default:u.*'
```

上述命令的执行效果如图 3-16 所示。

从图 3-16 可以看出，在命名空间 default 中用户创建了一个表名以字母 u 开头的表

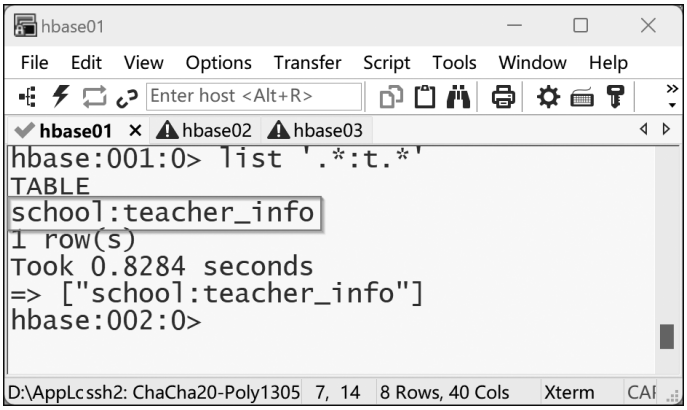

图 3-15　查看表(2)

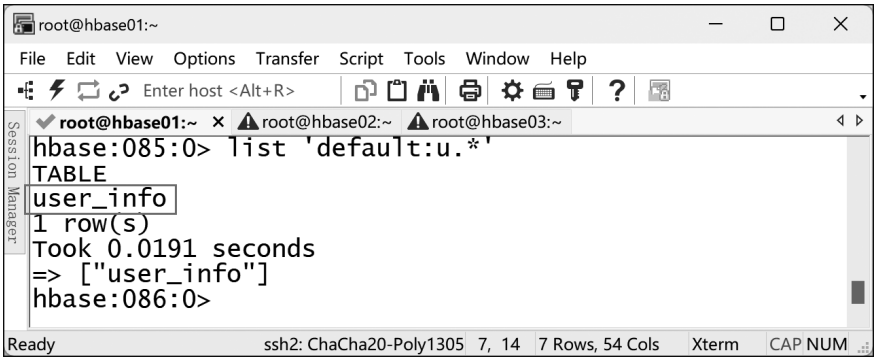

图 3-16　查看表(3)

user_info。

注意：查看表的范围仅限于用户创建的表，并不会涉及 HBase 的系统表，如属于命名空间 hbase 的表。

3.3.4　停用和启用表

HBase 中表的状态分为停用和启用，当表处于停用状态时，用户无法访问和操作表中的数据。默认情况下，用户创建的表为启用状态。HBase Shell 提供了相应的命令来停用和启用表，具体介绍如下。

1. 停用表

停用表的主要目的是在进行某些特定操作时，保护表中的数据不受并发访问的干扰，从而确保数据的一致性和完整性。例如，执行诸如修改表、删除表、数据迁移等操作。

HBase Shell 提供了 disable 命令用于停用表，其语法格式如下。

```
disable '[namespace:]table_name'
```

接下来演示如何停用命名空间 school 的表 teacher_info，在 HBase Shell 执行如下命令。

```
>disable 'school:teacher_info'
```

上述命令执行完成后，查看命名空间 school 中表 teacher_info 的信息，如图 3-17 所示。

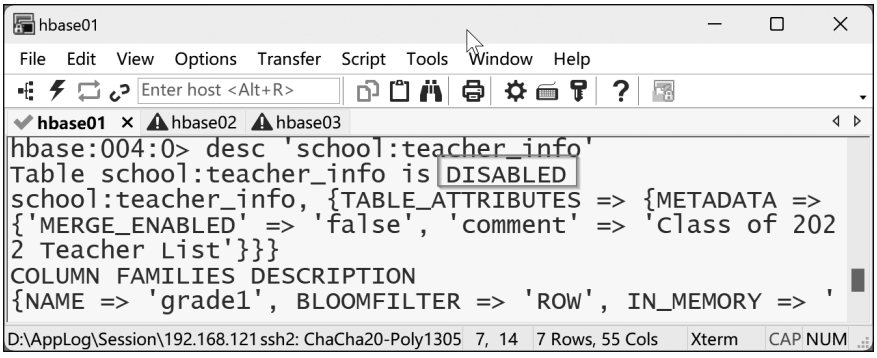

图 3-17 查看命名空间 school 中表 teacher_info 的信息（2）

从图 3-17 可以看出，命名空间 school 中表 teacher_info 的状态信息中出现 DISABLED 的内容，因此说明成功停用表。

2. 启用表

启用表的主要目的是在完成表的维护或修改操作后，允许用户重新访问和操作表中的数据。HBase Shell 提供了 enable 命令用于启用表，其语法格式如下。

```
enable '[namespace:]table_name'
```

接下来演示如何启用命名空间 school 的表 teacher_info，在 HBase Shell 执行如下命令。

```
>enable 'school:teacher_info'
```

上述命令执行完成后，查看命名空间 school 中表 teacher_info 的信息，如图 3-18 所示。

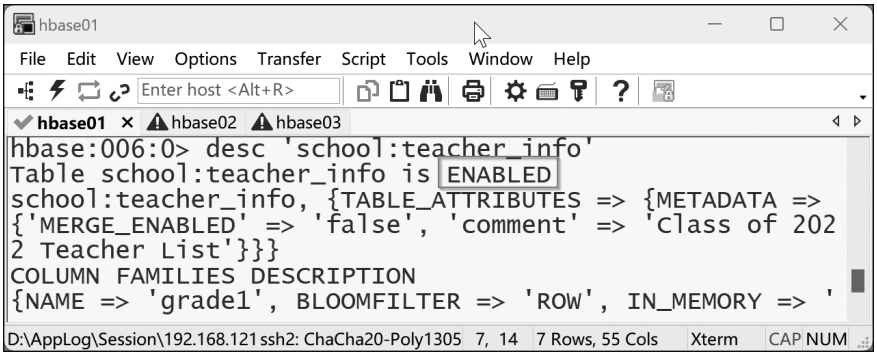

图 3-18 查看命名空间 school 中表 teacher_info 的信息（3）

从图 3-18 可以看出，命名空间 school 中表 teacher_info 的状态信息中出现 ENABLED 的内容，因此说明成功启用表。

注意：系统表是无法停用的。

多学一招：停用或启用多个表

当需要停用或启用多个表时，逐个执行 disable 或 enable 命令可能会非常耗时。为了

提高效率，HBase Shell 提供了 disable_all 和 enable_all 命令，用于一次性停用或启用多个表。这些命令可以根据指定的正则表达式来匹配表名，并对符合条件的表执行停用或启用操作。停用或启用多个表的语法格式如下。

```
# 停用多个表
disable_all '[namespace:]regular'
# 启用多个表
enable_all '[namespace:]regular'
```

上述语法格式中，namespace 和 regular 分别用于指定命名空间和正则表达式，如果不指定命名空间，那么会根据指定的正则表达式匹配用户创建的所有表。

这里以 disable_all 命令为例，演示如何使用正则表达式“.*:.*_info$”来匹配表名，停用用户创建的所有表中表名以字符串_info 结尾的表，在 HBase Shell 执行如下命令。

```
>disable_all '.*:.*_info$'
```

上述命令执行完成的效果如图 3-19 所示。

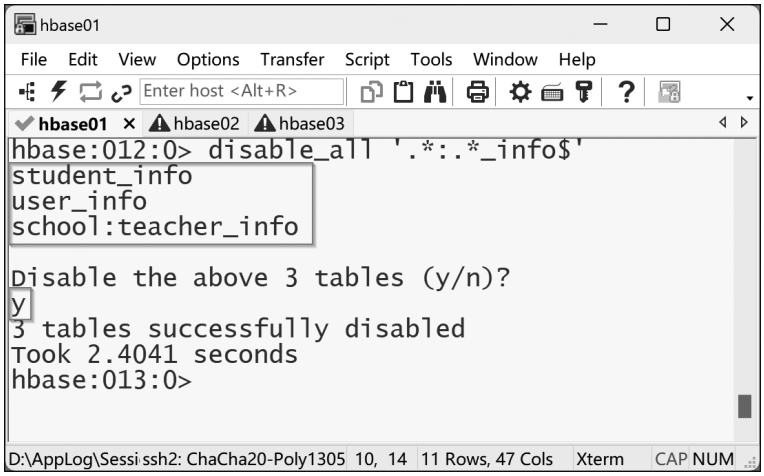

图 3-19　停用多个表

从图 3-19 可以看出，当执行 disable_all 命令时，会列出匹配到的所有表，然后询问是否对这些表执行停用操作，如果确认无误的话可以输入字母 y，此时便会停用匹配到的所有表。

使用 enable_all 命令启用多个表的方式与 disable_all 命令相同，这里不再赘述。

3.3.5　判断表

HBase Shell 提供了 exists 命令、is_enabled 命令和 is_disabled 命令用于判断表，具体内容如下。

1. exists 命令

exists 命令用于判断表是否存在，当该命令返回结果为 true 时，表示表存在；若返回结果为 false，则表示表不存在，其语法格式如下。

```
exists '[namespace:]table_name'
```

接下来演示如何判断命名空间 school 是否存在表 student_info，在 HBase Shell 执行如下命令。

```
>exists 'school:student_info'
```

上述命令的执行效果如图 3-20 所示。

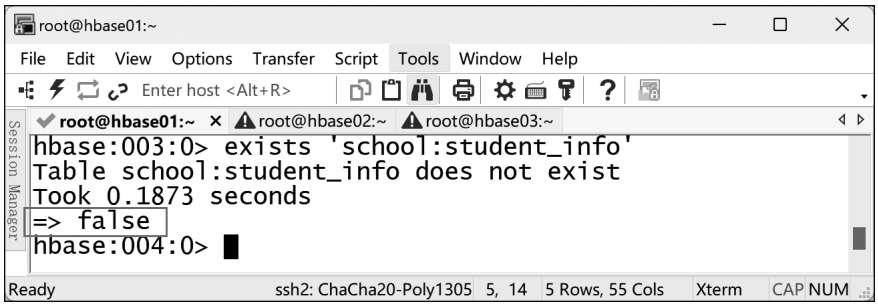

图 3-20 判断表是否存在

从图 3-20 可以看出，exist 命令的返回结果为 false，因此说明命名空间 school 不存在表 student_info。

2. is_enabled 命令

is_enabled 命令用于判断表是否处于启用状态，当该命令返回结果为 true 时，表示表处于启用状态；若返回结果为 false，则表示表处于停用状态，其语法格式如下。

```
is_enabled '[namespace:]table_name'
```

接下来演示如何判断命名空间 default 的表 user_info 是否处于启用状态，在 HBase Shell 执行如下命令。

```
>is_enabled 'user_info'
```

上述命令的执行效果如图 3-21 所示。

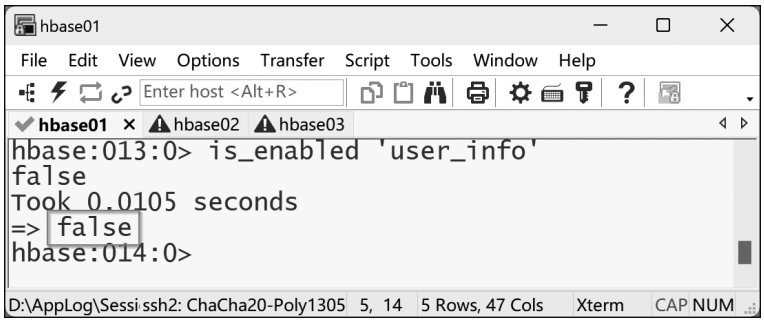

图 3-21 判断表是否处于启用状态

从图 3-21 可以看出，is_enabled 命令的返回结果为 false，因此说明命名空间 default 的表 user_info 处于停用状态。

3. is_disabled 命令

is_disabled 命令用于判断表是否处于停用状态，当该命令返回结果为 true 时，表示表处于停用状态；若返回结果为 false，则表示表处于启用状态，其语法格式如下。

```
is_disabled '[namespace:]table_name'
```

接下来演示如何判断命名空间 school 的表 teacher_info 是否处于停用状态，在 HBase Shell 执行如下命令。

```
>is_disabled 'school:teacher_info'
```

上述命令的执行效果如图 3-22 所示。

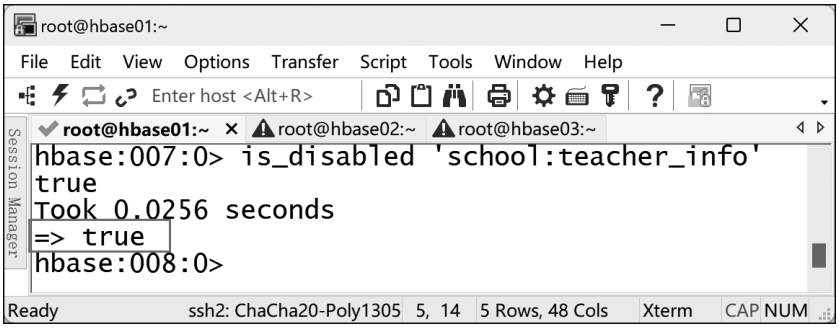

图 3-22　判断表是否处于停用状态

从图 3-22 可以看出，is_disabled 命令的返回结果为 true，因此说明命名空间 school 的表 teacher_info 处于停用状态。

3.3.6　修改表

HBase Shell 提供了 alter 命令用于修改表，包括修改表属性和列族，具体内容如下。

1. 修改表属性

修改表属性主要涉及对表的属性进行添加或删除的操作，具体内容如下。

1）添加属性

添加属性表示根据表的预定义属性为表添加属性及其属性值，如果添加的属性在表中已存在，那么可以根据指定的属性值来修改该属性的属性值。关于添加属性的语法格式如下。

```
alter '[namespace:]table_name',TABLE_PROPERTY_NAME =>
'TABLE_PROPERTY_VALUE'[,TABLE_PROPERTY_NAME =>'TABLE_PROPERTY_VALUE',...]
```

上述语法格式中，TABLE_PROPERTY_NAME 和 TABLE_PROPERTY_VALUE 用于根据表的预定义属性来添加表的属性及其属性值。

接下来演示如何为命名空间 school 的表 teacher_info 添加属性，具体需求如下。

- 添加预定义属性 READONLY，并指定属性值为 true。
- 添加预定义属性 MERGE_ENABLED，并指定属性值为 true。

根据上述需求，在 HBase Shell 执行如下命令。

```
>alter 'school:teacher_info',MERGE_ENABLED =>'true',READONLY =>'true'
```

上述命令执行完成后，查看命名空间 school 中表 teacher_info 的信息，如图 3-23 所示。

通过对比图 3-23 和图 3-13，可以发现命名空间 school 中表 teacher_info 的属性信息发

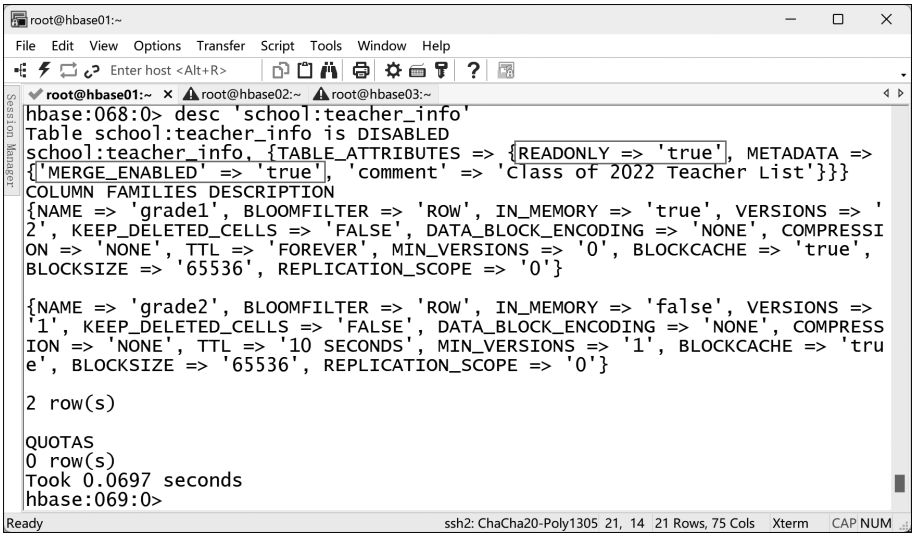

图 3-23 查看命名空间 school 中表 teacher_info 的信息(4)

生了变化。首先,预定义属性 READONLY 被添加到了表属性中;其次,由于预定义属性 MERGE_ENABLED 已经存在,所以它的属性值被修改为 true。

2) 删除属性

删除属性的语法格式如下。

```
    alter '[namespace:]table_name',METHOD =>'table_att_unset'
,NAME =>'TABLE_PROPERTY_NAME'
```

上述语法格式中,METHOD => 'table_att_unset'表示删除属性,TABLE_PROPERTY_NAME 用于指定属性名。

接下来演示如何删除命名空间 school 中表 teacher_info 的自定义属性 comment,在 HBase Shell 执行如下命令。

```
    >alter 'school:teacher_info',METHOD =>'table_att_unset',
NAME =>'comment'
```

上述命令执行完成后,查看命名空间 school 中表 teacher_info 的信息,如图 3-24 所示。

从图 3-24 可以看出,命名空间 school 中表 teacher_info 的属性信息中已经不存在自定义属性 comment。

2. 修改列族

修改列族主要涉及修改表中列族的属性,以及对表中的列族执行添加或删除操作,具体内容如下。

1) 修改列族属性

修改列族属性是指根据列族的预定义属性对列族的属性进行调整,其语法格式如下。

```
alter '[namespace:]table_name',
{NAME =>'columnfamily',CF_PROPERTY_NAME=>'CF_PROPERTY_VALUE'
[,CF_PROPERTY_NAME=>'CF_PROPERTY_VALUE',...]}
```

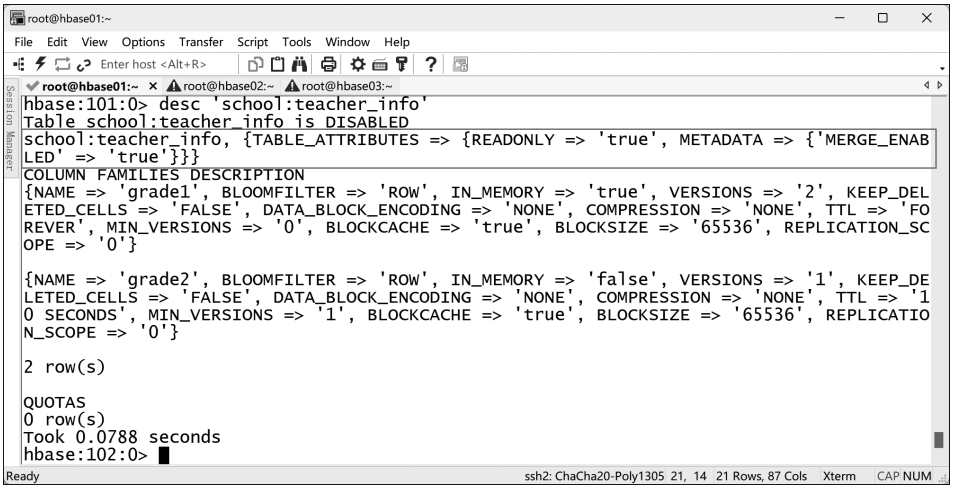

图 3-24　查看命名空间 school 中表 teacher_info 的信息(5)

```
[,{NAME =>'columnfamily',CF_PROPERTY_NAME=>'CF_PROPERTY_VALUE'
[,CF_PROPERTY_NAME=>'CF_PROPERTY_VALUE',...]},
...]
```

上述语法格式可以同时对多个列族的多个属性进行调整,其中 columnfamily 用于指定列族的名称。CF_PROPERTY_NAME 和 CF_PROPERTY_VALUE 用于指定预定义属性及其对应的属性值。

接下来演示如何对命名空间 school 的表 teacher_info 中列族 grade1 的属性进行修改,具体需求如下。

- 通过列族的预定义属性 VERSIONS,将列族存储数据的最大版本数调整为 3。
- 通过列族的预定义属性 BLOCKSIZE,将列族的 HFile 中每个数据块的大小调整为 131072。

根据上述需求,在 HBase Shell 执行如下命令。

```
    >alter 'school:teacher_info',{NAME =>'grade1',VERSIONS =>3,
BLOCKSIZE =>131072}
```

上述命令执行完成后,查看命名空间 school 中表 teacher_info 的信息,如图 3-25 所示。

通过对比图 3-25 和图 3-24,可以观察到,列族 grade1 的属性信息发生了变化,其中预定义属性 VERSIONS 的属性值变更为 3;预定义属性 BLOCKSIZE 的属性值变更为 131072。

2) 添加列族

添加列族是指在表中添加一个或多个新的列族。在添加列族时,可以根据列族的预定义属性对其进行相应的配置。

```
alter '[namespace:]table_name',
{NAME =>'columnfamily'[,'CF_PROPERTY_NAME'=>'CF_PROPERTY_VALUE',...]}
[,{NAME =>'columnfamily'[,'CF_PROPERTY_NAME'=>'CF_PROPERTY_VALUE',...]}
,...]
```