HBase, ZooKeeper, and Sqoop

Learning Objectives

- Introduce HBase
- Introduce ZooKeeper
- Introduce Sqoop

What Is HBase

- HBase is a distributed column-oriented database built on top of HDFS.
- HBase is used for real-time read/write random access to very large datasets.
- HBase is not relational and does not support SQL.
- Hbase can do what an RDBMS cannot: host very large, sparsely populated tables on clusters.

Column Family

- Row columns are grouped into column families and all column family members have a common prefix.
- The columns temperature:air and temperature:dew_point are both members of the temperature column family.
- Whereas station:identifier belongs to the station family.
- The column family prefix must be composed of printable characters. The qualifying tail and the column family qualifier can be made of any arbitrary bytes.

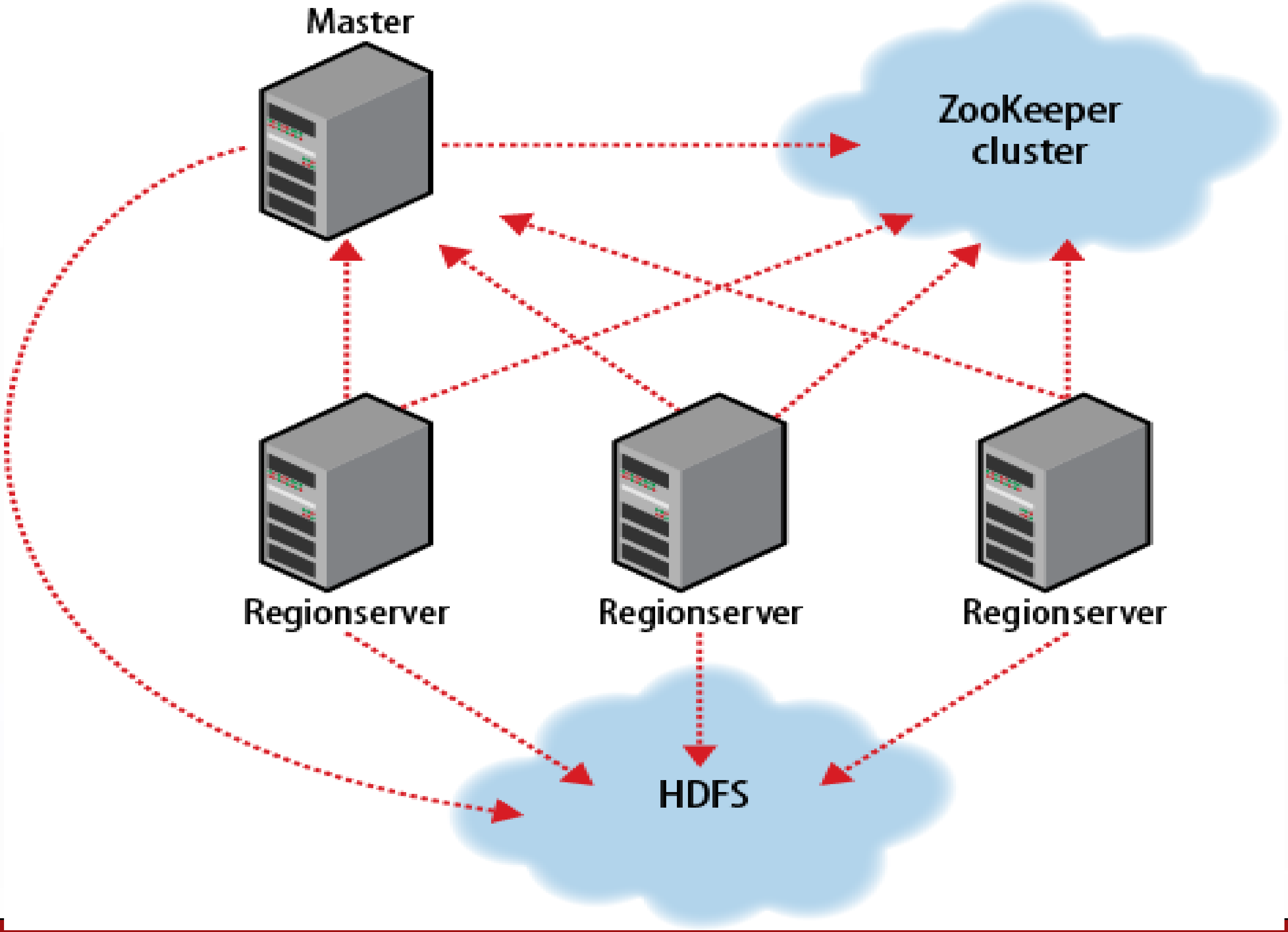
Regions

- Tables are automatically partitioned horizontally by HBase into regions.
- Each region comprises a subset of a table’s rows.
- A region is denoted by the table it belongs to, its first row, inclusive, and last row, exclusive.
- Initially, a table comprises a single region, but as the size of the region grows, after it crosses a configurable size threshold, it splits at a row boundary into two new regions of approximately equal size.
- Until this first split happens, all loading will be against the single server hosting the original region.

Hbase Configuration

- HBase includes an HBase *master* node and a cluster of one or more *regionserver* slaves.
- The HBase master is responsible for bootstrapping a virgin install, for assigning regions to registered regionservers, and for recovering regionserver failures.
- The regionservers carry zero or more regions and field client read/write requests.
- The regionservers also manage region splits, informing the HBase master about the new daughter regions.

Hbase Cluster Members



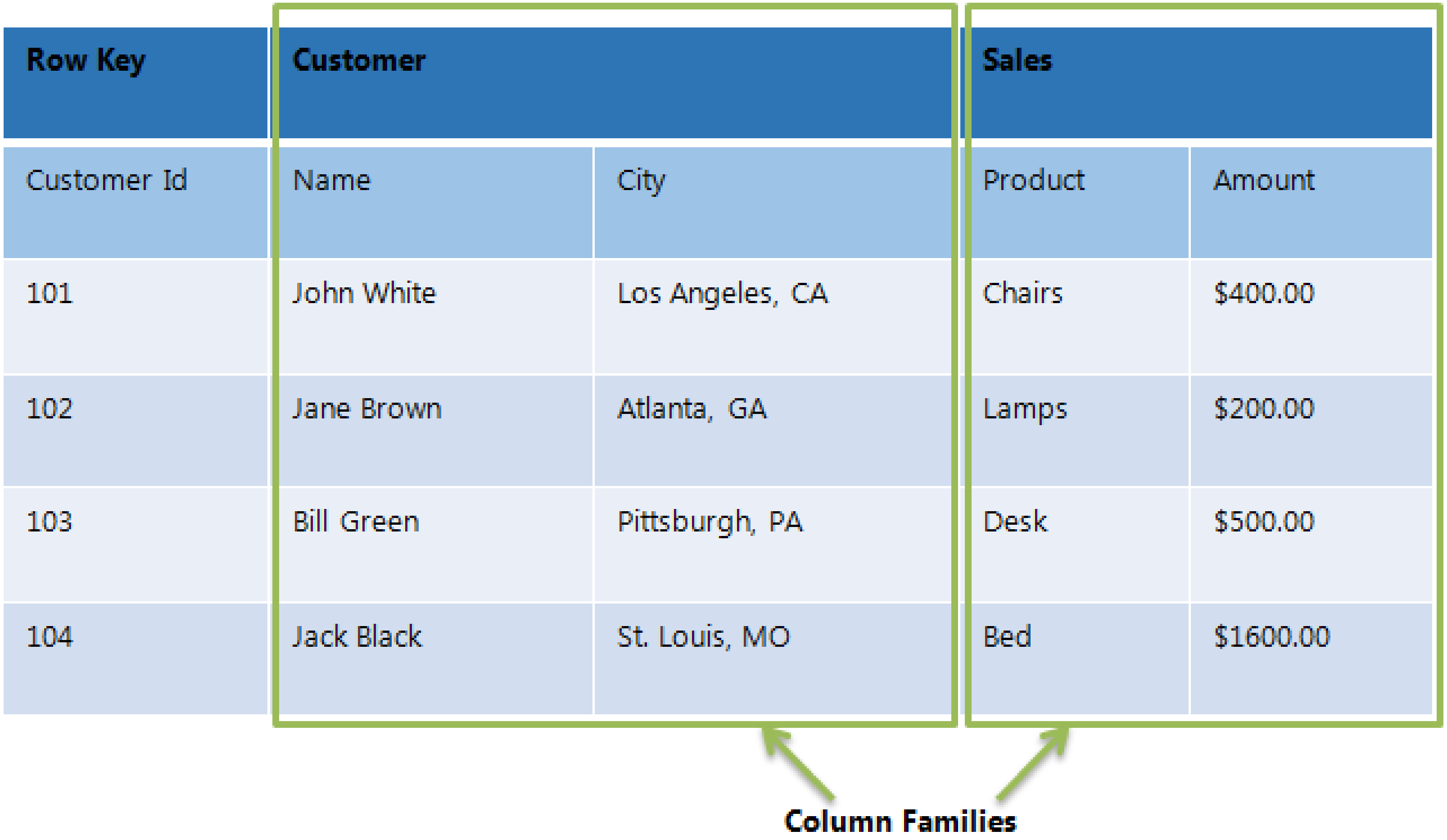
Hbase Needs ZooKeeper

- HBase depends on ZooKeeper (Chapter 14), and by default it manages a ZooKeeper instance as the authority on cluster state.
- HBase hosts vitals such as the location of the root catalog table and the address of the current cluster master.
- Assignment of regions is mediated via ZooKeeper in case participating servers crash mid-assignment.
- Hosting the assignment transaction state in ZooKeeper makes it so recovery can pick up on the assignment where the crashed server left off.

Hadoop – The Definitive Guide

Chapters 13, 14 & 15: HBase, ZooKeeper, and Sqoop

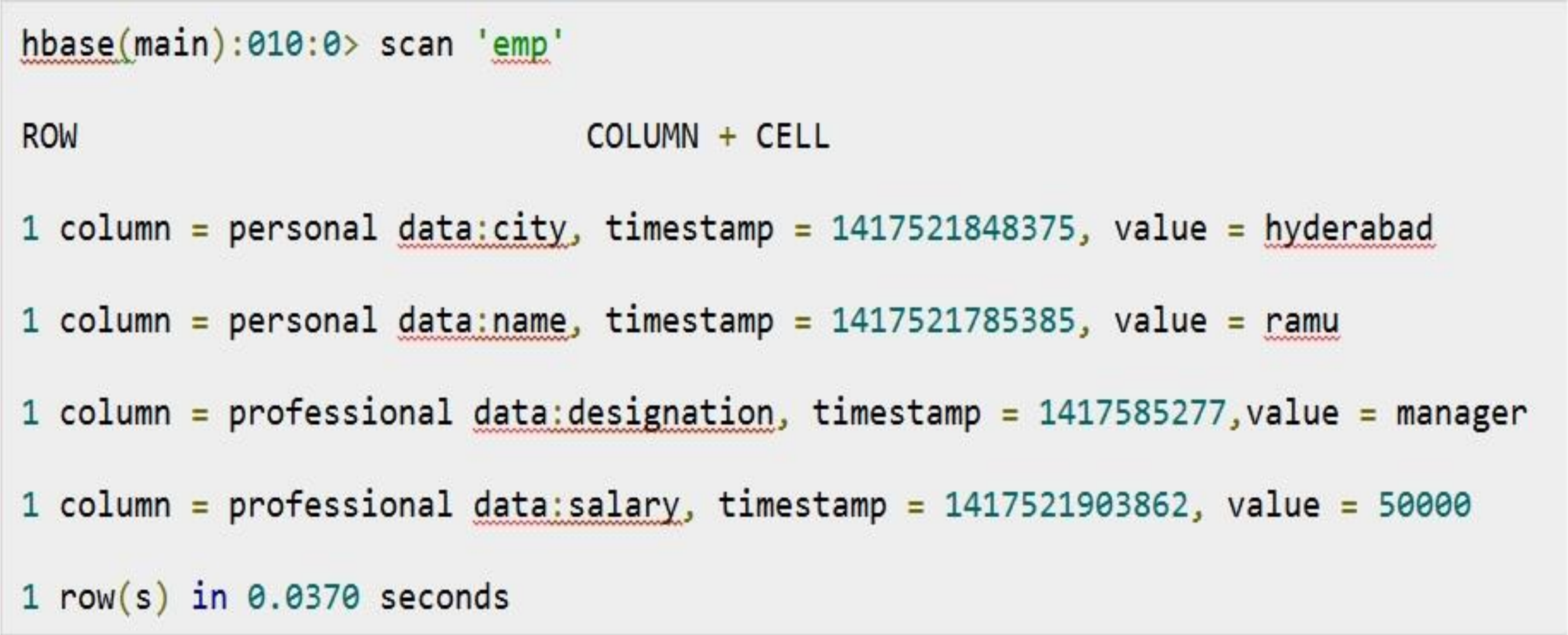
Hbase Example – 1



Row Key	Customer		Sales	
Customer Id	Name	City	Product	Amount
101	John White	Los Angeles, CA	Chairs	$400.00
102	Jane Brown	Atlanta, GA	Lamps	$200.00
103	Bill Green	Pittsburgh, PA	Desk	$500.00
104	Jack Black	St. Louis, MO	Bed	$1600.00

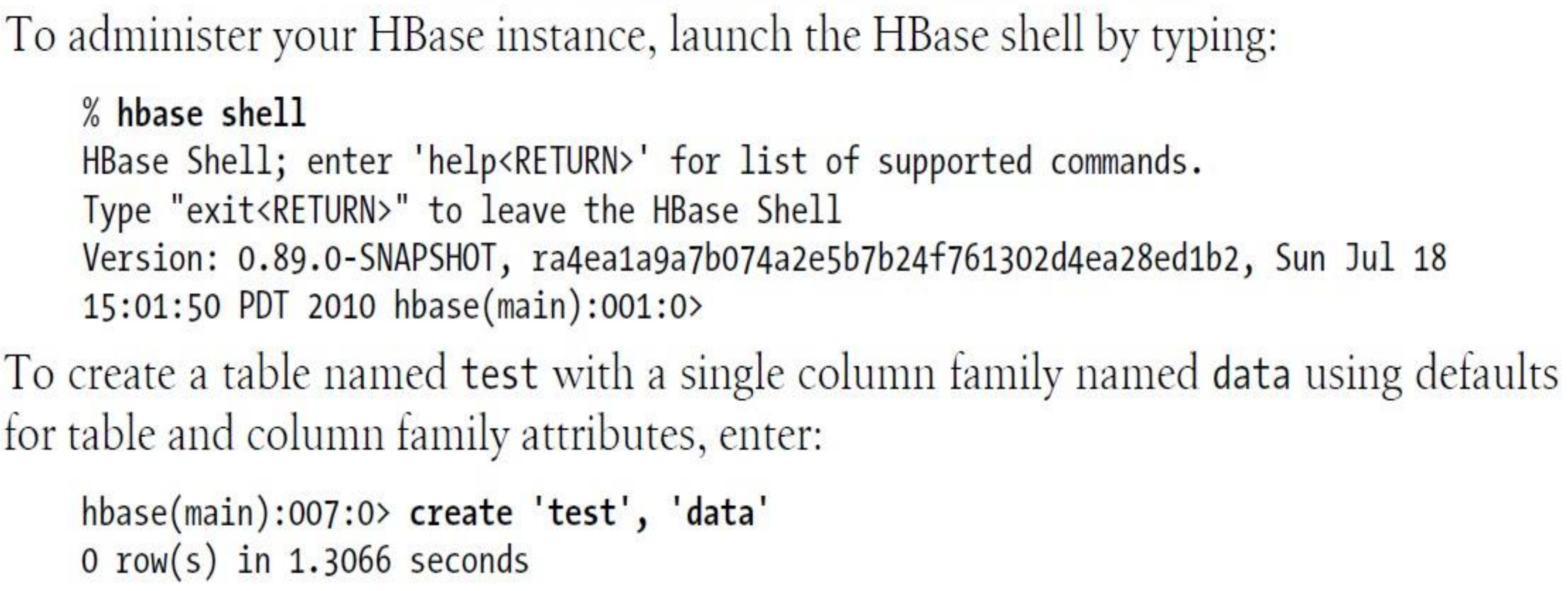
Column Families

Hbase Example – 2



```
hbase(main):010:0> scan 'emp'

ROW                              COLUMN + CELL

1 column = personal data:city, timestamp = 1417521848375, value = hyderabad

1 column = personal data:name, timestamp = 1417521785385, value = ramu

1 column = professional data:designation, timestamp = 1417585277,value = manager

1 column = professional data:salary, timestamp = 1417521903862, value = 50000

1 row(s) in 0.0370 seconds
```

Launch Hbase



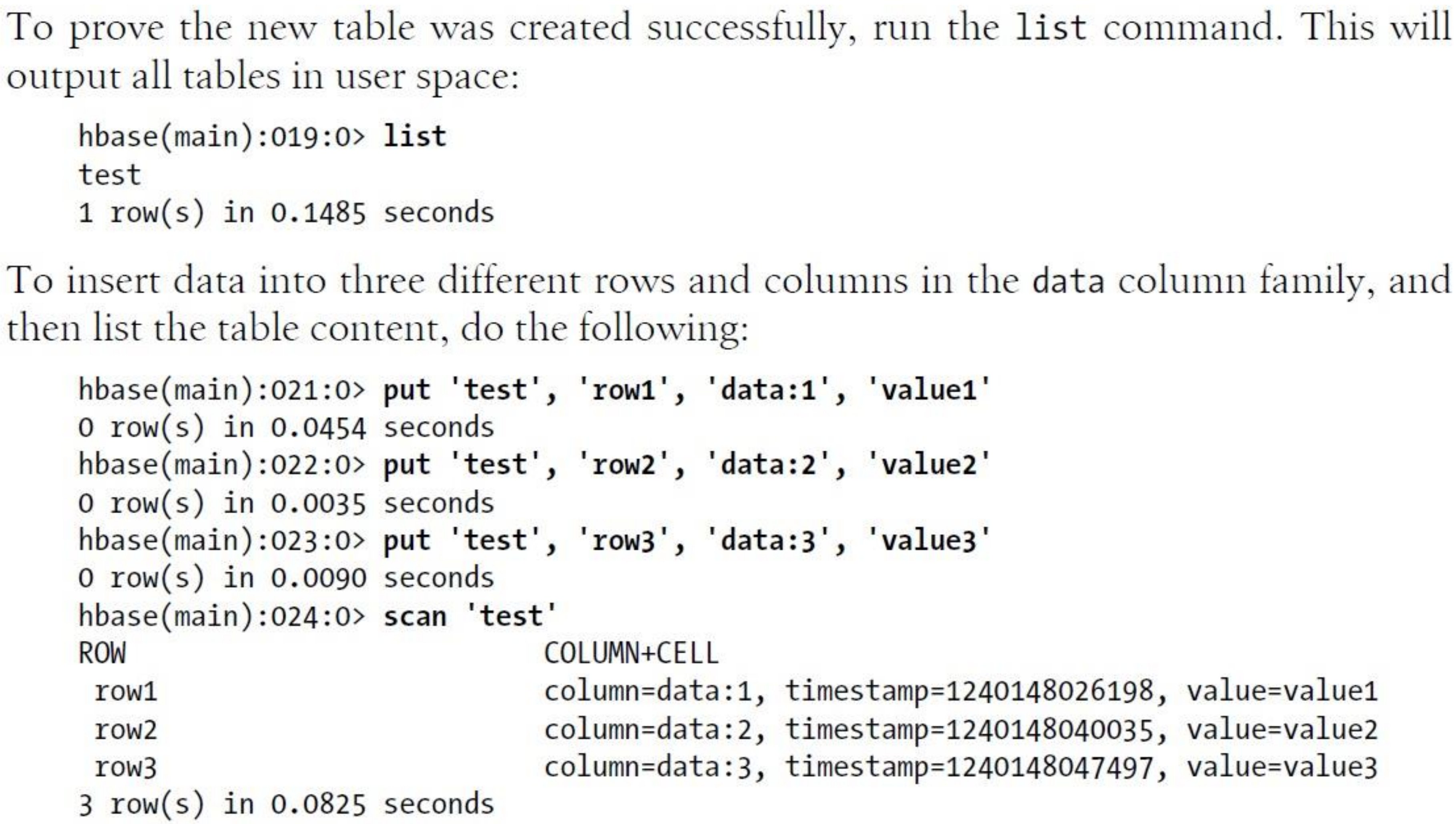
To administer your HBase instance, launch the HBase shell by typing:

```
% hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version: 0.89.0-SNAPSHOT, ra4ea1a9a7b074a2e5b7b24f761302d4ea28ed1b2, Sun Jul 18
15:01:50 PDT 2010 hbase(main):001:0>
```

To create a table named **test** with a single column family named **data** using defaults for table and column family attributes, enter:

```
hbase(main):007:0> create 'test', 'data'
0 row(s) in 1.3066 seconds
```

Insert and List Data



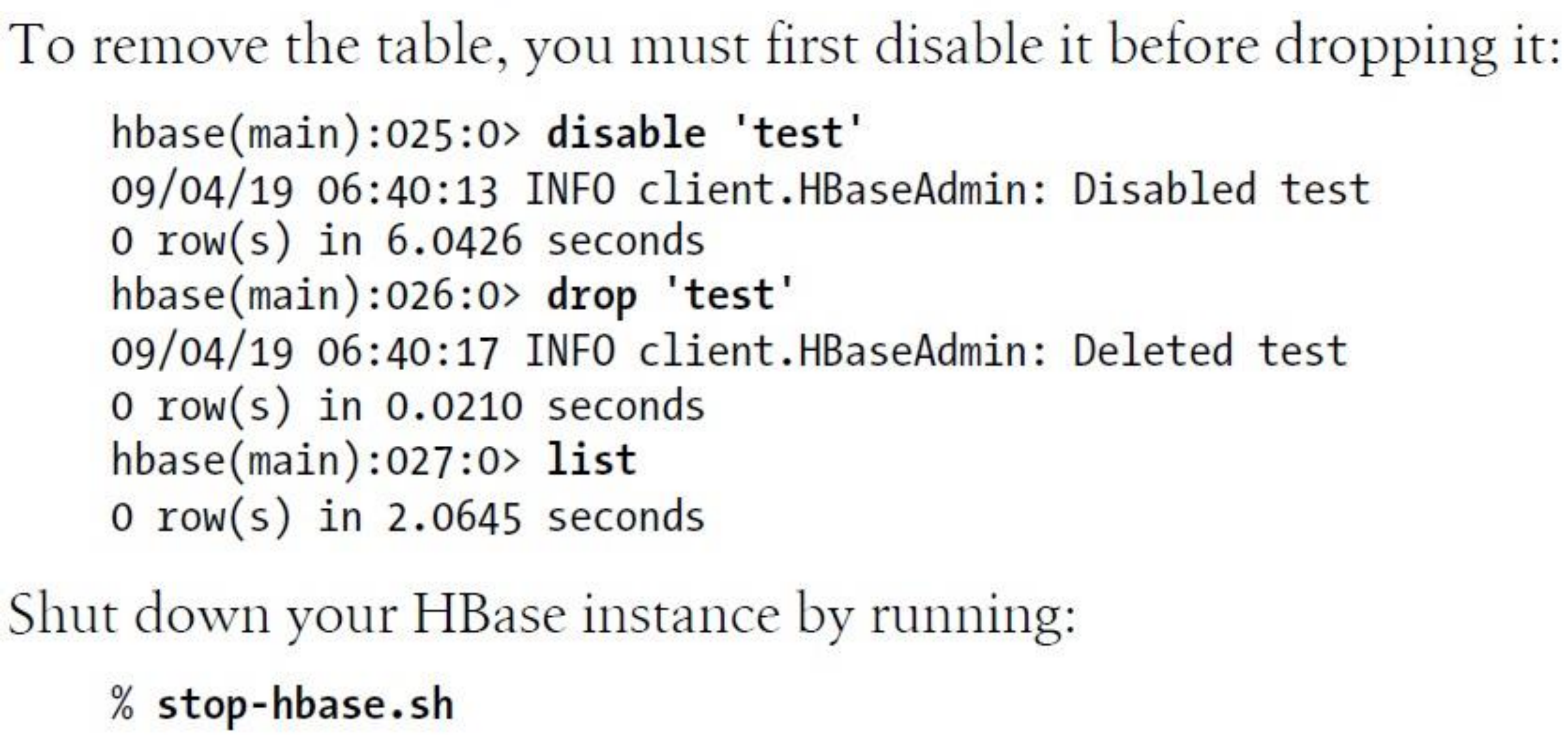
To prove the new table was created successfully, run the `list` command. This will output all tables in user space:

```
hbase(main):019:0> list
test
1 row(s) in 0.1485 seconds
```

To insert data into three different rows and columns in the `data` column family, and then list the table content, do the following:

```
hbase(main):021:0> put 'test', 'row1', 'data:1', 'value1'
0 row(s) in 0.0454 seconds
hbase(main):022:0> put 'test', 'row2', 'data:2', 'value2'
0 row(s) in 0.0035 seconds
hbase(main):023:0> put 'test', 'row3', 'data:3', 'value3'
0 row(s) in 0.0090 seconds
hbase(main):024:0> scan 'test'
ROW                          COLUMN+CELL
 row1                        column=data:1, timestamp=1240148026198, value=value1
 row2                        column=data:2, timestamp=1240148040035, value=value2
 row3                        column=data:3, timestamp=1240148047497, value=value3
3 row(s) in 0.0825 seconds
```

More Hbase Commands



To remove the table, you must first disable it before dropping it:

```
hbase(main):025:0> disable 'test'
09/04/19 06:40:13 INFO client.HBaseAdmin: Disabled test
0 row(s) in 6.0426 seconds
hbase(main):026:0> drop 'test'
09/04/19 06:40:17 INFO client.HBaseAdmin: Deleted test
0 row(s) in 0.0210 seconds
hbase(main):027:0> list
0 row(s) in 2.0645 seconds
```

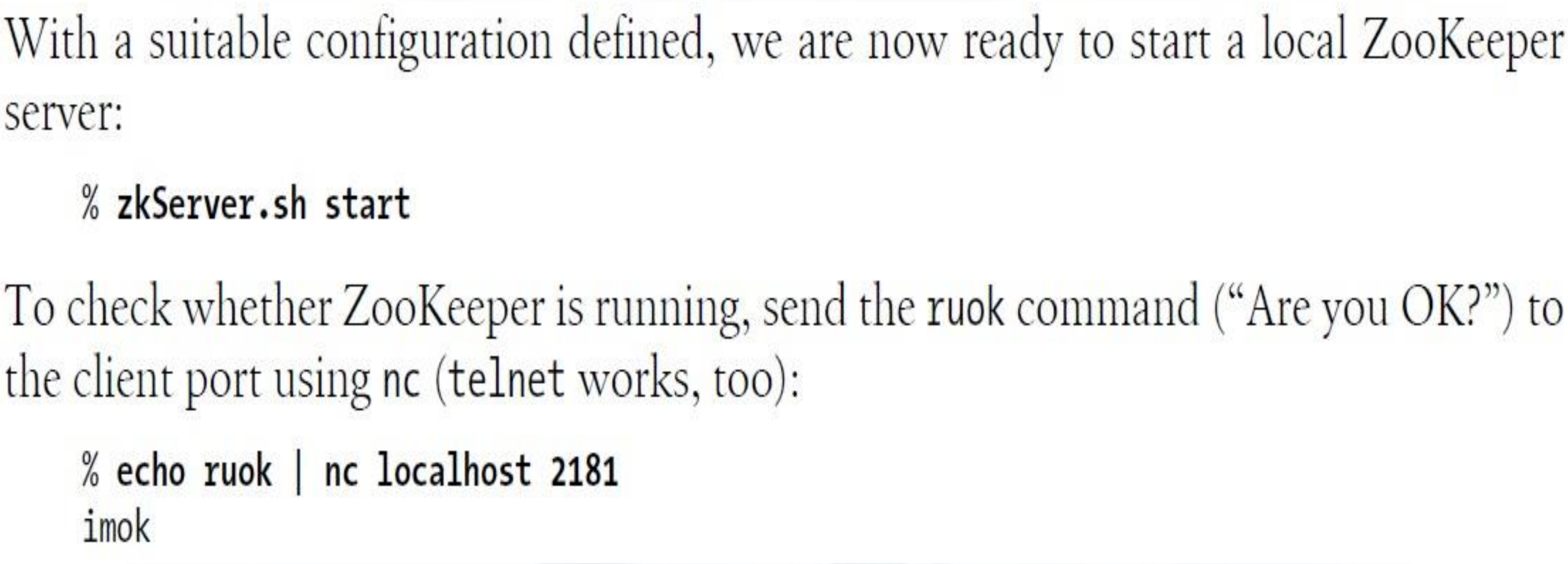
Shut down your HBase instance by running:

```
% stop-hbase.sh
```

What Is ZooKeeper

- ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.
- ZooKeeper provides users a set of tools to build distributed applications that can safely handle partial failures.
- ZooKeeper has a rich set of building blocks that can be used to build a large class of coordination data structures and protocols. Examples include: distributed queues, distributed locks, and leader election among a group of peers.

Start ZooKeeper



With a suitable configuration defined, we are now ready to start a local ZooKeeper server:

```
% zkServer.sh start
```

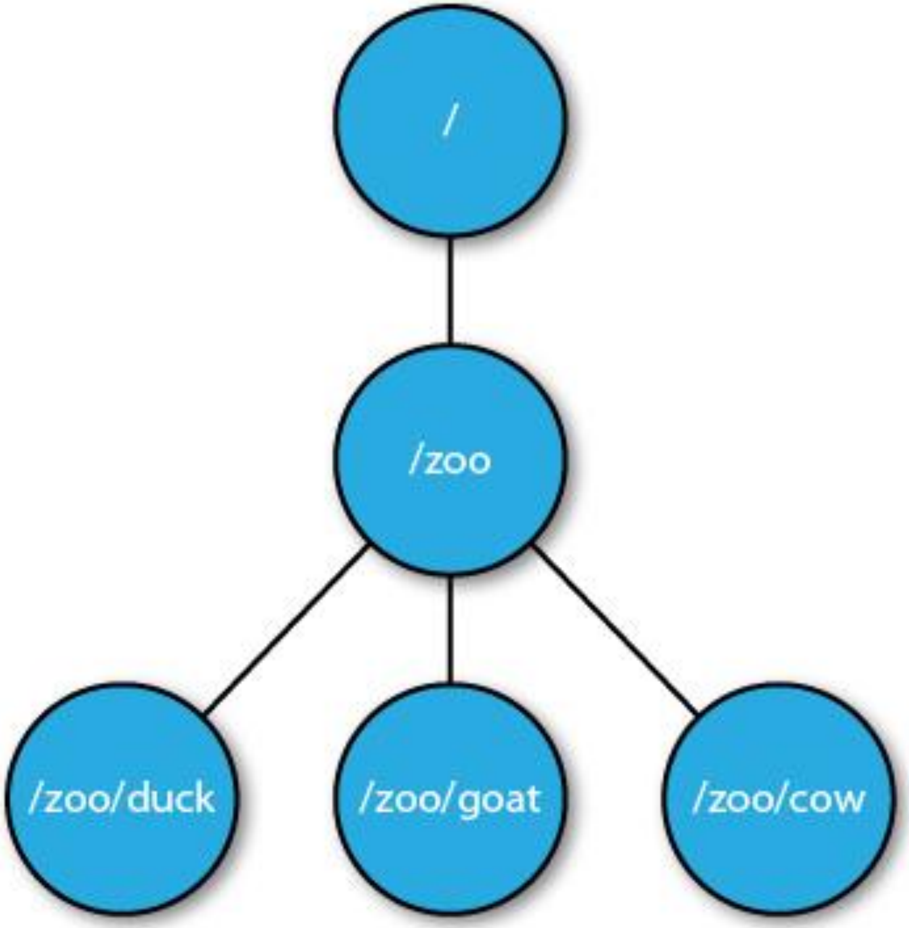
To check whether ZooKeeper is running, send the ruok command (“Are you OK?”) to the client port using nc (telnet works, too):

```
% echo ruok | nc localhost 2181
imok
```

Znodes – 1

- One way of understanding ZooKeeper is to think of it as providing a high-availability filesystem.
- It doesn’t have files and directories, but a unified concept of a node, called a znode.
- Znode acts both as a container of data (like a file) and a container of other znodes (like a directory).

Znodes – 2



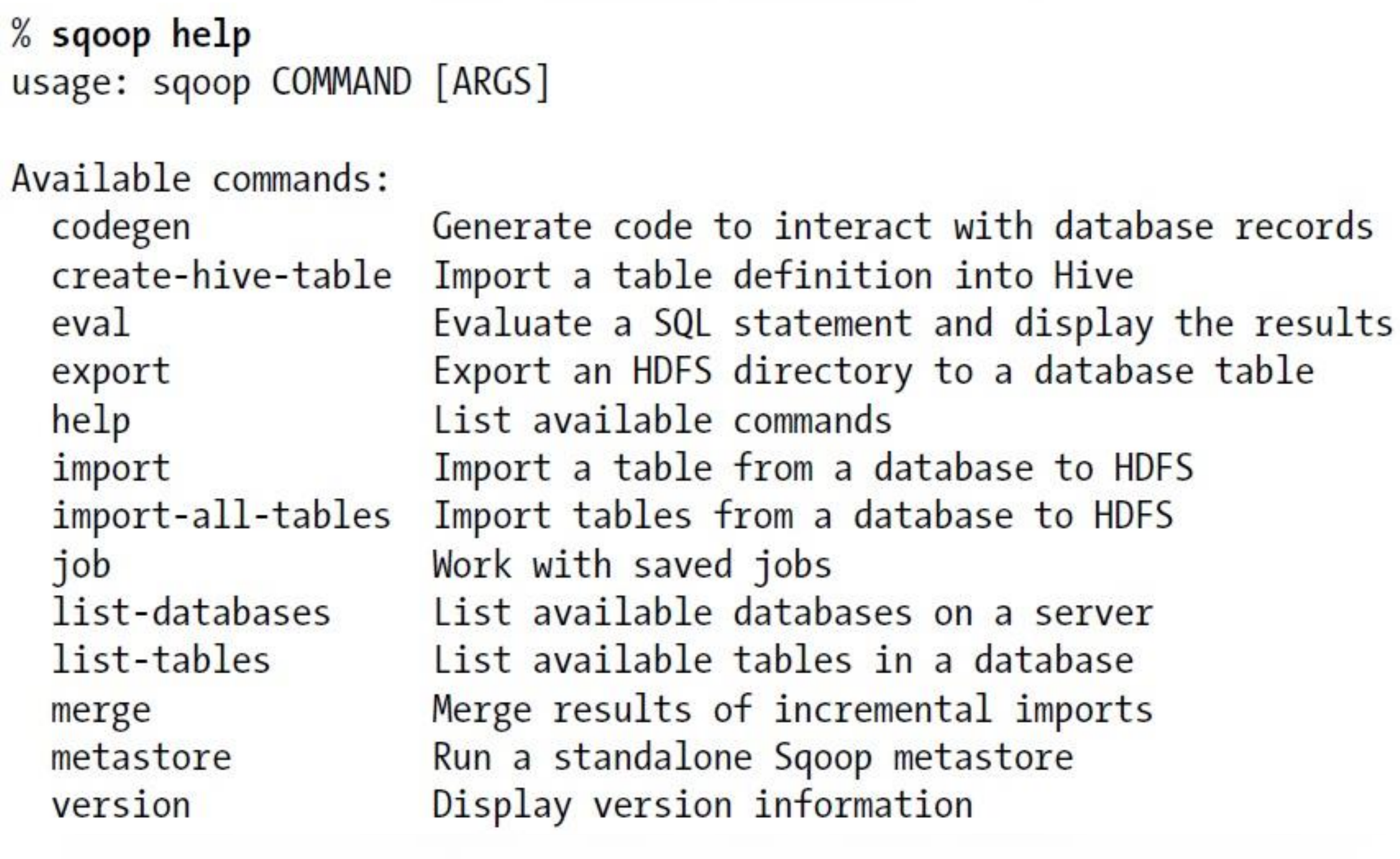
ZooKeeper Application Examples

- Create a ZooKeeper group – p. 493
- Join a ZooKeeper group – p. 495
- List members in a group – p. 496
- Delete a ZooKeeper group – p. 498

What Is Sqoop

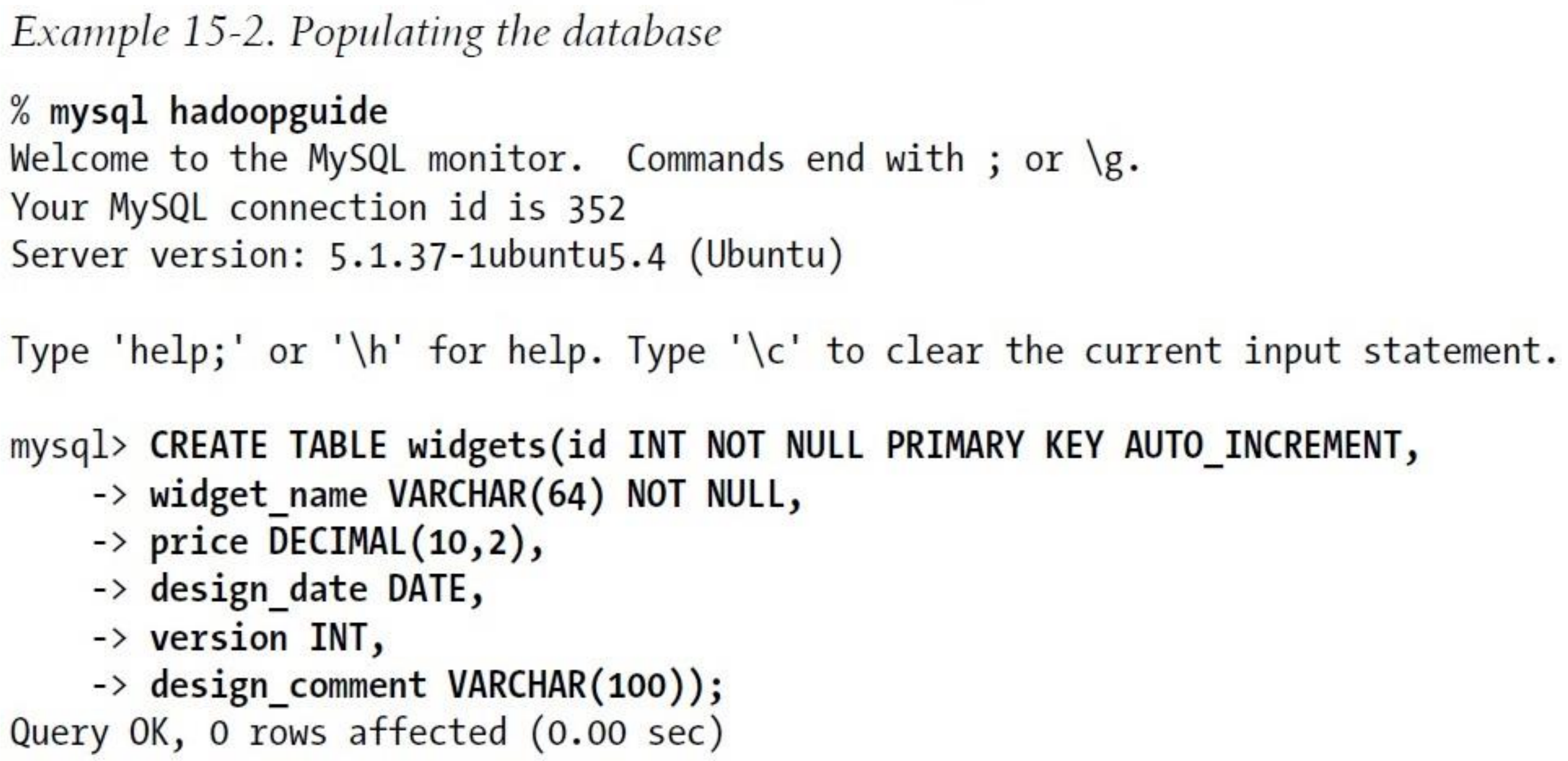
- Sqoop is an open source tool that allows users to extract data from a structured data store into Hadoop for further processing.
- It’s even possible to use Sqoop to move data from a database into HBase.
- When the final results of an analytic pipeline are available, Sqoop can export these results back to the data store for consumption by other clients.

Sqoop Commands



```
% sqoop help
usage: sqoop COMMAND [ARGS]

Available commands:
  codegen            Generate code to interact with database records
  create-hive-table  Import a table definition into Hive
  eval               Evaluate a SQL statement and display the results
  export             Export an HDFS directory to a database table
  help               List available commands
  import             Import a table from a database to HDFS
  import-all-tables  Import tables from a database to HDFS
  job                Work with saved jobs
  list-databases     List available databases on a server
  list-tables        List available tables in a database
  merge              Merge results of incremental imports
  metastore          Run a standalone Sqoop metastore
  version            Display version information
```

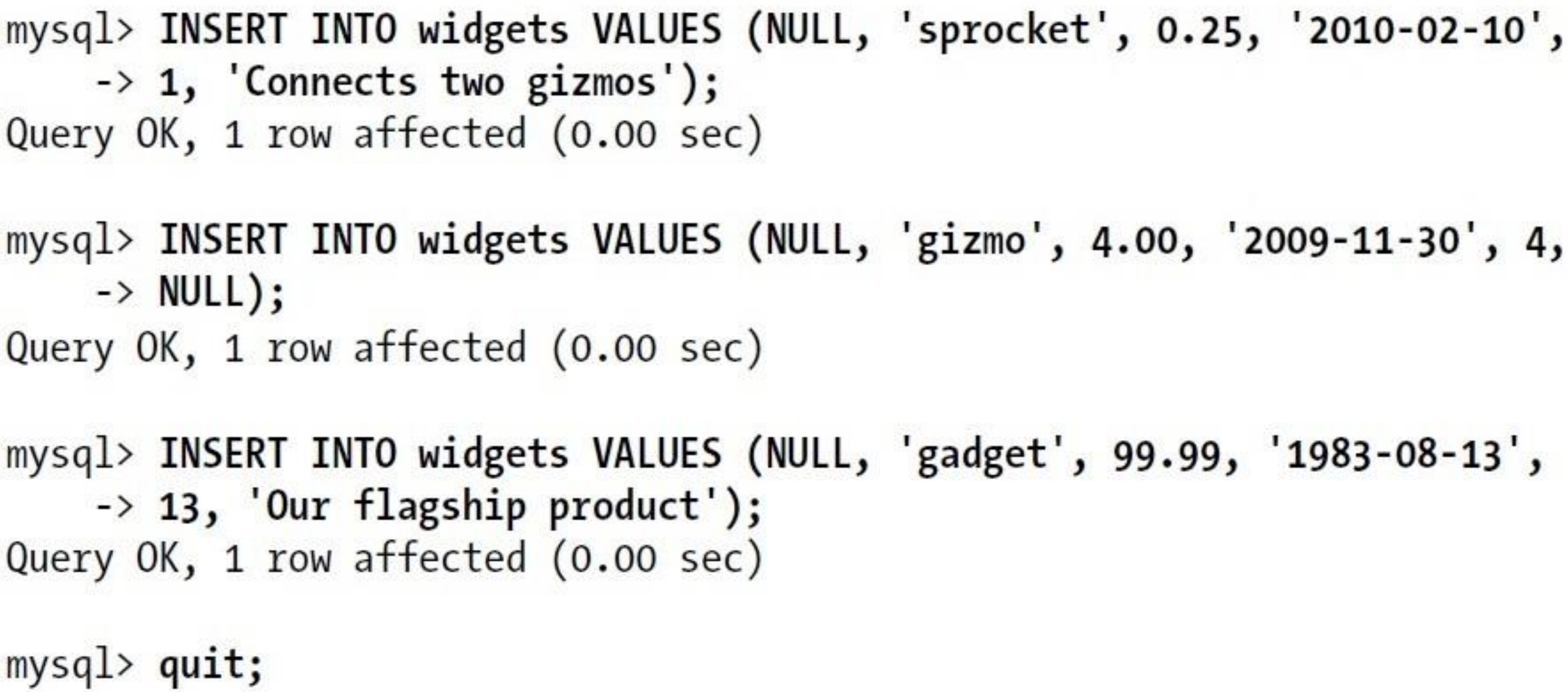
A Sqoop Example – 1



Example 15-2. Populating the database

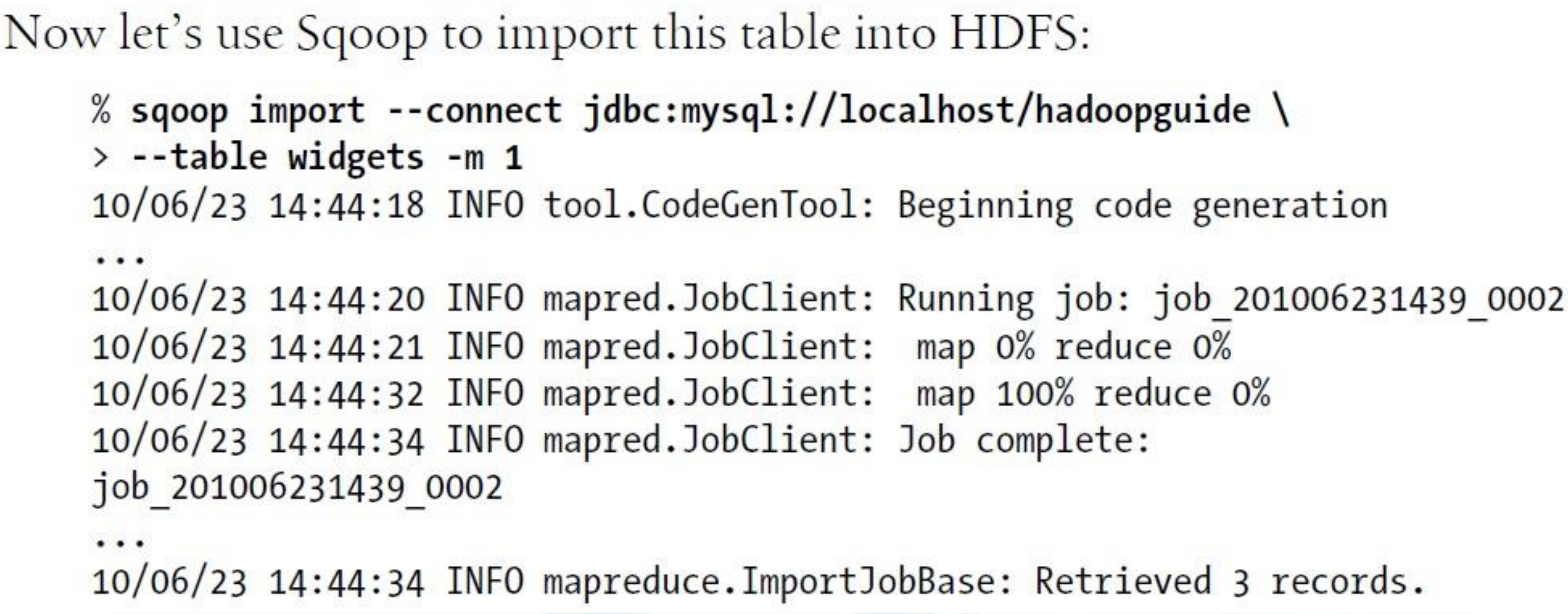
```
% mysql hadoopguide
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 352
Server version: 5.1.37-1ubuntu5.4 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE TABLE widgets(id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> widget_name VARCHAR(64) NOT NULL,
    -> price DECIMAL(10,2),
    -> design_date DATE,
    -> version INT,
    -> design_comment VARCHAR(100));
Query OK, 0 rows affected (0.00 sec)
```

A Sqoop Example – 2



```
mysql> INSERT INTO widgets VALUES (NULL, 'sprocket', 0.25, '2010-02-10',
    -> 1, 'Connects two gizmos');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO widgets VALUES (NULL, 'gizmo', 4.00, '2009-11-30', 4,
    -> NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO widgets VALUES (NULL, 'gadget', 99.99, '1983-08-13',
    -> 13, 'Our flagship product');
Query OK, 1 row affected (0.00 sec)

mysql> quit;
```

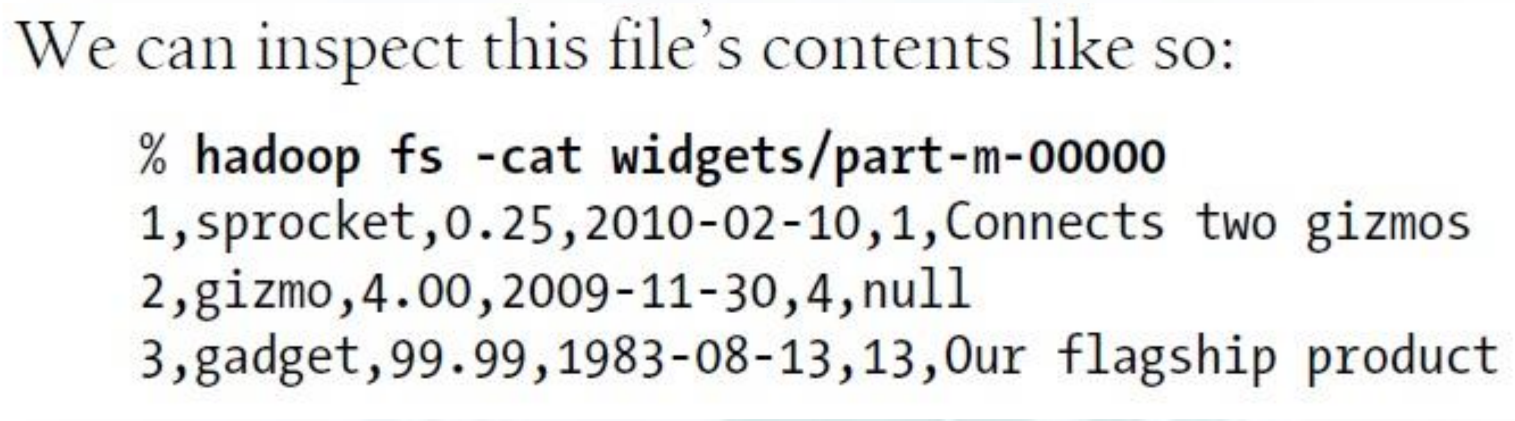
A Sqoop Example – 3



Now let’s use Sqoop to import this table into HDFS:

```
% sqoop import --connect jdbc:mysql://localhost/hadoopguide \
> --table widgets -m 1
10/06/23 14:44:18 INFO tool.CodeGenTool: Beginning code generation
...
10/06/23 14:44:20 INFO mapred.JobClient: Running job: job_201006231439_0002
10/06/23 14:44:21 INFO mapred.JobClient:  map 0% reduce 0%
10/06/23 14:44:32 INFO mapred.JobClient:  map 100% reduce 0%
10/06/23 14:44:34 INFO mapred.JobClient: Job complete:
job_201006231439_0002
...
10/06/23 14:44:34 INFO mapreduce.ImportJobBase: Retrieved 3 records.
```

A Sqoop Example – 4



We can inspect this file's contents like so:

```
% hadoop fs -cat widgets/part-m-00000
1,sprocket,0.25,2010-02-10,1,Connects two gizmos
2,gizmo,4.00,2009-11-30,4,null
3,gadget,99.99,1983-08-13,13,Our flagship product
```