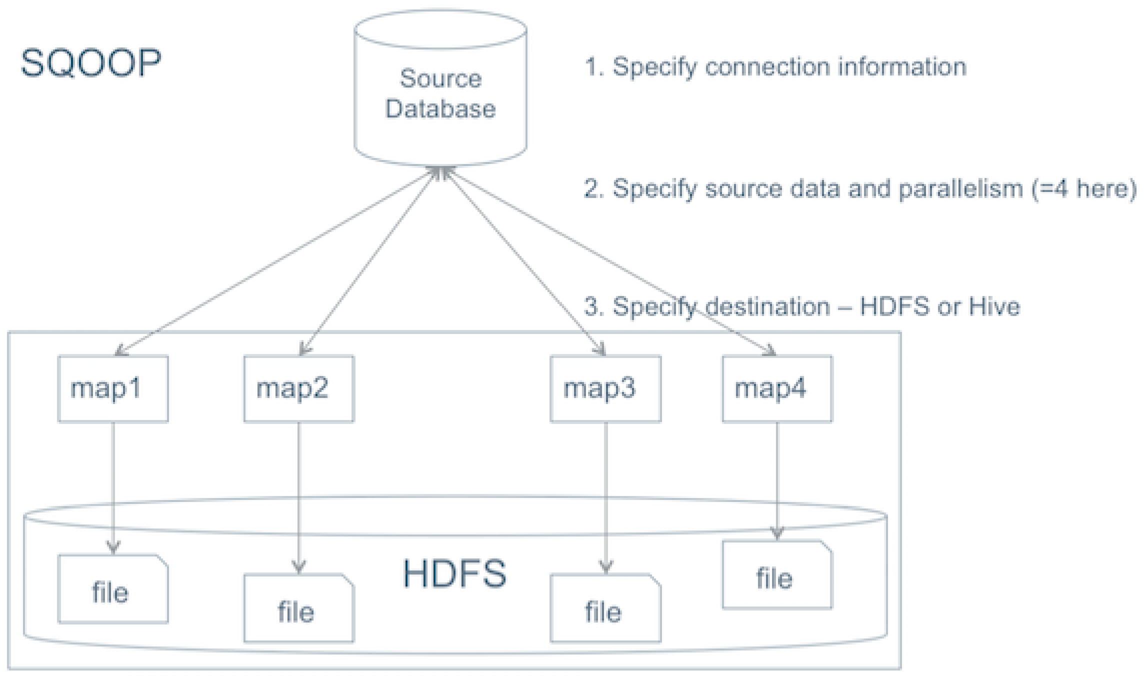
3.2. Using Sqoop to Move Data into Hive

Sqoop is a tool that enables you to bulk import and export data from a database. You can use Sqoop to import data into HDFS or directly into Hive. However, Sqoop can only import data into Hive as a text file or as a SequenceFile. To use the ORC file format, you must use a two-phase approach: first use Sqoop to move the data into HDFS, and then use Hive to convert the data into the ORC file format as described in the above Steps 3 and 4 of "Moving Data from HDFS to Hive Using an External Table."

A detailed Sqoop user guide is available on the Apache web site here.

The process for using Sqoop to move data into Hive is shown in the following diagram:

Figure 1.2. Using Sqoop to Move Data into Hive


SQOOP
Source Database
1. Specify connection information
2. Specify source data and parallelism (=4 here)
3. Specify destination – HDFS or Hive
map1
map2
map3
map4
HDFS
file
file
file
file

Moving Data into Hive Using Sqoop

1. **Specify the source connection information**.

   First, you must specify the:

   - database URI (`db.foo.com` in the following example)
   - database name (`bar`)
   - connection protocol (`jdbc:mysql:`)

   For this example, use the following command:

   ```
   sqoop import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES
   ```

   If the source database requires credentials, such as a username and password, you can enter the password on the command line or specify a file where the password is stored.

   For example:

- Enter the password on the command line:

```
sqoop import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES --username <username> -P
Enter password: (hidden)
```

- Specify a file where the password is stored:

```
sqoop import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES --username <username> --password-file ${user.home}/.password
```

More connection options are described in the Sqoop User Guide on the Apache web site.

2. **Specify the data and the parallelism for import**:

   a. Specify the data simply.

   Sqoop provides flexibility to specify exactly the data you want to import from the source system:

   - Import an **entire table**:

   ```
   sqoop import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES
   ```

   - Import a **subset of the columns** from a table:

   ```
   sqoop import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES --columns "employee_id,first_name,last_name,job_title"
   ```

   - Import only the **latest records** by specifying them with a WHERE clause and then that they be appended to an existing table:

   ```
   import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES  --where "start_date > '2010-01-01'"

   import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES --where "id > 100000" --target-dir /incremental_dataset --appe
   ```

   You can also use a free-form SQL statement.

   b. Specify parallelism.

   There are three options for specifying *write parallelism* (number of map tasks):

   - Explicitly set the number of mappers using `--num-mappers`. Sqoop evenly splits the primary key range of the source table:

```
sqoop import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES --num-mappers 8
```

In this scenario, the source table must have a primary key.

- Provide an alternate split key using `--split-by`. This evenly splits the data using the alternate split key instead of a primary key:

```
sqoop import --connect jdbc:mysql://db.foo.com/bar --table EMPLOYEES --split-by dept_id
```

This method is useful if primary keys are not evenly distributed.

- When there is not split key or primary key, the data import must be sequential. Specify a single mapper by using `--num-mappers 1` or `--autoreset-to-one-mapper`.

c. Specify the data using a query.

Instead of specifying a particular table or columns, you can specify the date with a query. You can use one of the following options:

- Explicitly specify a *split-by column* using `--split-by` and put `$ CONDITIONS` that Sqoop replaces with range conditions based on the split-by key. This method requires a target directory:

```
sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id)
WHERE $CONDITIONS' --split-by a.id --target-dir /user/foo/joinresults
```

- Use sequential import if you cannot specify a split-by column:

```
sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id)
WHERE $CONDITIONS' -m 1 --target-dir /user/foo/joinresults
```

To try a sample query without importing data, use the `eval` option to print the results to the command prompt:

```
sqoop eval --connect jdbc:mysql://db.foo.com/bar --query "SELECT * FROM employees LIMIT 10"
```

3. **Specify the destination for the data: HDFS or Hive**.

Here is an example of specifying the HDFS target directory:

```
sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id)
WHERE $CONDITIONS' --split-by a.id --target-dir /user/foo/joinresults
```

If you can add text data into your Hive table, you can specify that the data be directly added to Hive. Using `--hive-import` is the primary method to add text data directly to Hive:

```
sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES --hive-import
```

This method creates a metastore schema after storing the text data in HDFS.

If you have already moved data into HDFS and want to add a schema, use the `create-hive-table` Sqoop command:

```
sqoop create-hive-table (generic-args) (create-hive-table-args)
```

Additional options for importing data into Hive with Sqoop:

Table 1.6. Sqoop Command Options for Importing Data into Hive

Sqoop Command Option	Description
`--hive-home <directory>`	Overrides `$HIVE_HOME`.
`--hive-import`	Imports tables into Hive using Hive's default delimiters if none are explicitly set.
`--hive-overwrite`	Overwrites existing data in the Hive table.
`--create-hive-table`	Creates a hive table during the operation. If this option is set and the Hive table already exists, the job will fail. Set to `false` by default.
`--hive-table <table_name>`	Specifies the table name to use when importing data into Hive.

Sqoop Command Option	Description
`--hive-drop-import-delims`	Drops the delimiters `\n`, `\r`, and `\01` from string fields when importing data into Hive.
`--hive-delims-replacement`	Replaces the delimiters `\n`, `\r`, and `\01` from strings fields with a user-defined string when importing data into Hive.
`--hive-partition-key`	Specifies the name of the Hive field on which a sharded database is partitioned.
`--hive-partition-value <value>`	A string value that specifies the partition key for data imported into Hive.
`--map-column-hive <map>`	Overrides the default mapping from SQL type to Hive type for configured columns.