# Importing Relational Data with Sqoop

Prof. Hyuk-Yoon Kwon

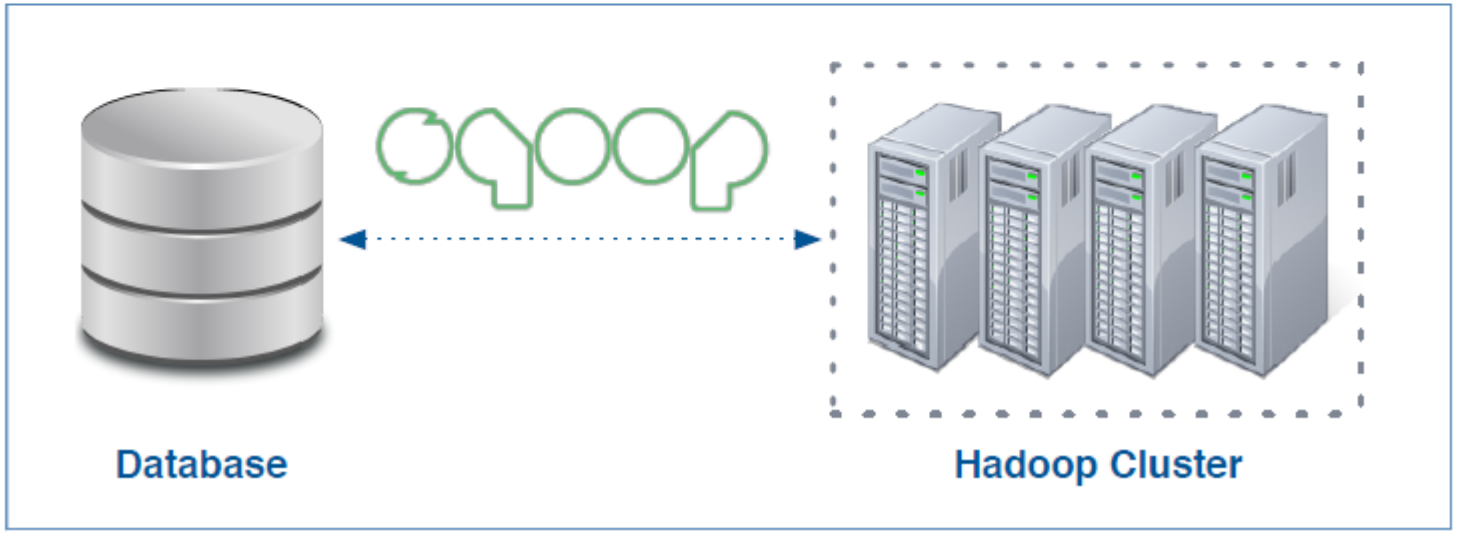
# Importing Relational Data with Apache Sqoop

In this chapter you will learn

- How to import tables from an RDBMS into your Hadoop cluster
- How to change the delimiter and file format of imported tables
- How to control which columns and rows are imported
- What techniques you can use to improve Sqoop's performance
- How the next-generation version of Sqoop compares to the original

# What is Apache Sqoop?

- **Open source Apache project originally developed by Cloudera**
  - The name is a contraction of "SQL-to-Hadoop"
- **Sqoop exchanges data between a database and HDFS**
  - Can import all tables, a single table, or a partial table into HDFS
  - Data can be imported a variety of formats
  - Sqoop can also export data from HDFS to a database

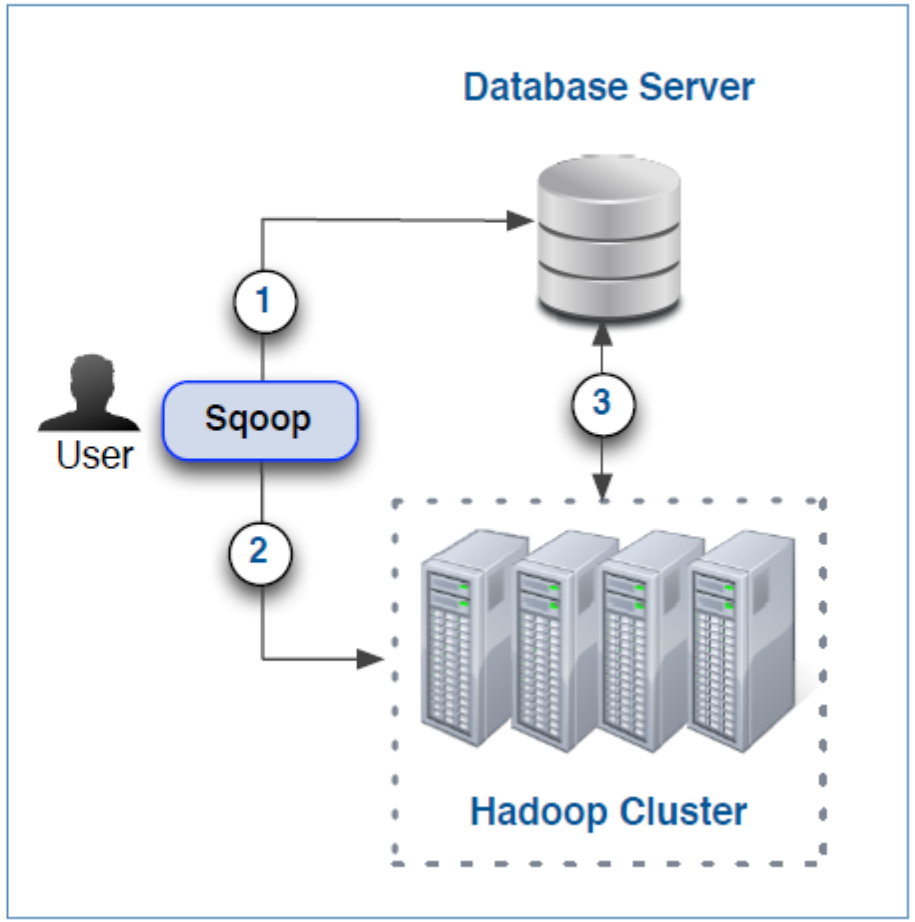


# How Does Sqoop Work?

application of MapReduce

- Sqoop is a client-side application that imports data using Hadoop MapReduce

- A basic import involves three steps orchestrated by Sqoop
  1. Examine table details
  2. Create and submit job to cluster
  3. Fetch records from table and write this data to HDFS



# Basic Syntax

- **Sqoop is a command-line utility with several subcommands, called *tools***
  - There are tools for import, export, listing database contents, and more
  - Run **`sqoop help`** to see a list of all tools
  - Run **`sqoop help tool-name`** for help on using a specific tool
- **Basic syntax of a Sqoop invocation**

```
$ sqoop tool-name [tool-options]
```

- **This command will list all tables in the `loudacre` database in MySQL**

```
$ sqoop list-tables \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser \
    --password pw
```

authentication.

# Overview of the Import Process

- **Imports are performed using Hadoop MapReduce jobs**
- **Sqoop begins by examining the table to be imported**
  - Determines the primary key, if possible
  - Runs a *boundary query* to see how many records will be imported
  - Divides result of boundary query by the number of tasks (mappers)
    - Uses this to configure tasks so that they will have equal loads
- **Sqoop also generates a Java source file for each table being imported**
  - It compiles and uses this during the import process
  - The file remains after import, but can be safely deleted

# Importing an Entire Database with Sqoop

- The `import-all-tables` tool imports an entire database
  - Stored as comma-delimited files
  - Default base location is your HDFS home directory
  - Data will be in subdirectories corresponding to name of each table

```
$ sqoop import-all-tables \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw
```

* --target-dir
⇒ specify directory itself. (no sub-dir)
importing single table

- Use the `--warehouse-dir` option to specify a different base directory

parent directory
⇒ create sub-directories
importing multi-tables

```
$ sqoop import-all-tables \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --warehouse-dir /loudacre
```

# Importing a Single Table with Sqoop

- The `import` tool imports a single table
- This example imports the `accounts` table
  - It stores the data in HDFS as comma-delimited fields

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw
```

- This variation writes tab-delimited fields instead

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --fields-terminated-by "\t"
```

changing delimeter

# Practice: Import Data

1. List the tables in the **loudacre** database
2. Use Sqoop to import the **accounts** table in the **loudacre** database and save it in HDFS under /loudacre
3. List the contents of the **accounts** directory under /loudacre
4. Use HDFS tail command to view the last part of the file for each of the MapReduce partition files
5. The first six digits in the output are the **account ID**. Take note of highest **account ID** because you will use it in the next step

# Incremental Imports (1)

modification

- **What if records have changed since last import?** how to update data?
  - Could re-import all records, but this is inefficient
- **Sqoop's `incremental lastmodified` mode imports new and modified records**
  - Based on a timestamp in a specified column
  - You must ensure timestamps are updated when records are added or changed in the database

```
$ sqoop import --table invoices \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --incremental lastmodified \
    --check-column mod_dt \
    --last-value '2015-09-30 16:00:00'
```

⇒ modified after this time, update is reflected.

# Incremental Imports (2)

- **Or use Sqoop's `incremental` append mode to import only *new* records**
  - Based on value of last record in specified column

```
$ sqoop import --table invoices \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --incremental append \
    --check-column id \
    --last-value 9478306
```

# Practice: Import Incremental Update

1. Run the add_new_accounts.py script to add the latest accounts to MySQL
2. Incrementally import and append the newly added accounts to the accounts directory. Use Sqoop to import on the last value on the **acct_num** column largest **account ID**
3. List the contents of the accounts directory to verify the Sqoop import.
4. You should see three new files. Use Hadoop's cat command to view the entire contents of these files

# Exporting Data from Hadoop to RDBMS with Sqoop

- **Sqoop's `import` tool pulls records from an RDBMS into HDFS**
- **It is sometimes necessary to *push* data in HDFS back to an RDBMS**
  - Good solution when you must do batch processing on large data sets
  - Export results to a relational database for access by other systems
- **Sqoop supports this via the `export` tool**
  - The RDBMS table must already exist prior to export

```
$ sqoop export \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --export-dir /loudacre/recommender_output \
    --update-mode allowinsert \
    --table product_recommendations
```

# Importing Partial Tables with Sqoop

- Import only specified columns from accounts table

"projection" operation

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --columns "id,first_name,last_name,state"
```

only 4 columns are imported.

- Import only matching rows from accounts table

"selection" operation

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --where "state='CA'"
```

specific rows are imported

# Using a Free-Form Query

- You can also import the results of a query, rather than a single table
- Supply a complete SQL query using the `--query` option
  - You must add the *literal* `WHERE $CONDITIONS` token
  - Use `--split-by` to identify field used to divide work among mappers
  - The `--target-dir` option is required for free-form queries

```
$ sqoop import \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --target-dir /data/loudacre/payable \
    --split-by accounts.id \
    --query 'SELECT accounts.id, first_name,
last_name, bill_amount FROM accounts JOIN invoices ON
(accounts.id = invoices.cust_id) WHERE $CONDITIONS'
```

# Using a Free-Form Query with WHERE Criteria

- **The `--where` option is ignored in a free-form query**
  - You must specify your criteria using `AND` following the `WHERE` clause

instead use!

```
$ sqoop import \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --target-dir /data/loudacre/payable \
    --split-by accounts.id \
    --query 'SELECT accounts.id, first_name,
last_name, bill_amount FROM accounts JOIN invoices ON
(accounts.id = invoices.cust_id) WHERE $CONDITIONS AND
bill_amount >= 40'
```

# Options for Database Connectivity

- **Generic (JDBC)** Java protocol
  - Compatible with nearly any database
  - Overhead imposed by JDBC can limit performance
- **Direct Mode**
  - Can improve performance through use of database-specific utilities efficient !
  - Currently supports MySQL and Postgres (use `--direct` option)
  - Not all Sqoop features are available in direct mode

# Controlling Parallelism

# mappers depends on size of data

MapReduce is parallel.

- **By default, Sqoop typically imports data using four parallel tasks (called mappers)** (default) ⇒ split by 4 mappers
  - Increasing the number of tasks might improve import speed
  - Caution: Each task adds load to your database server
- **You can *influence* the number of tasks using the `-m` option**
  - Sqoop views this only as a hint and might not honor it

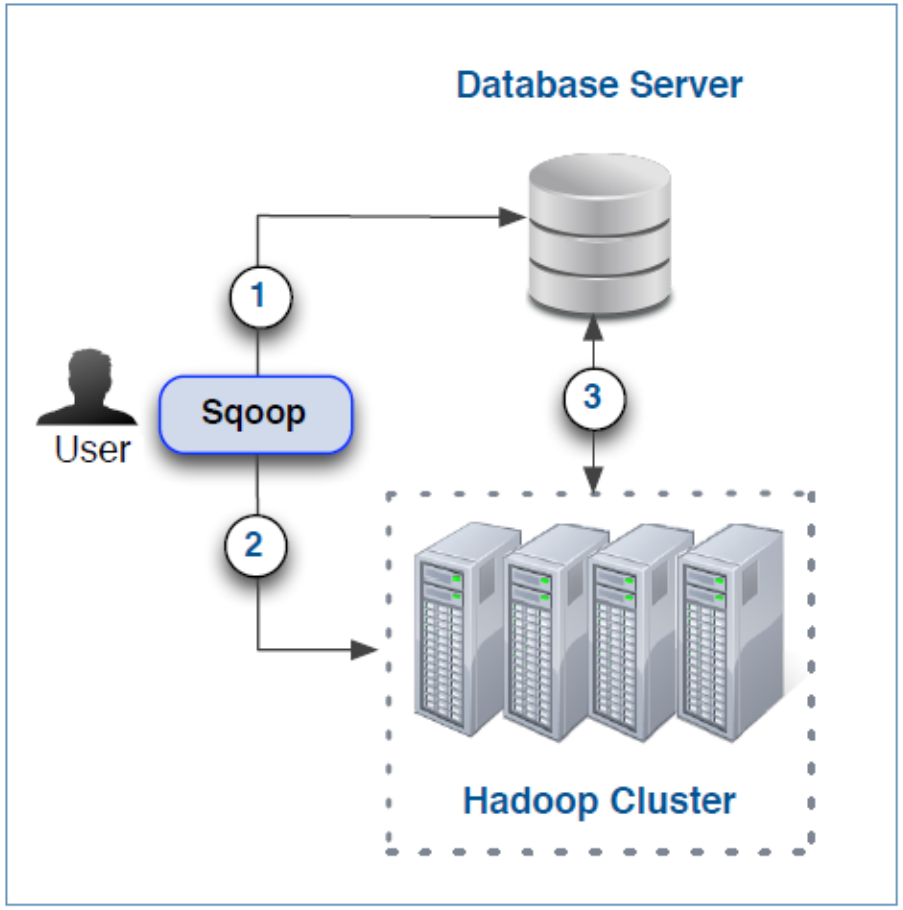
```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    -m 8
```

- **Sqoop assumes all tables have an evenly-distributed numeric primary key**
  - Sqoop uses this column to divide work among the tasks
  - You can use a different column with the `--split-by` option ⇒ using other column instead of primary key

# Limitations of Sqoop

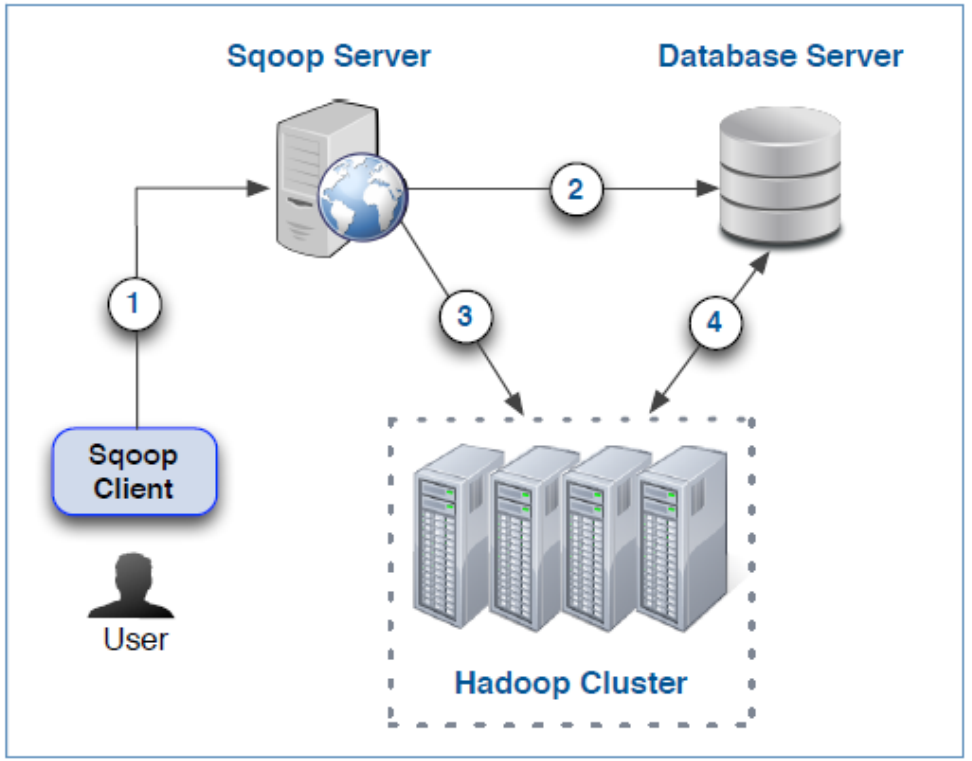
- **Sqoop is stable and has been used successfully in production for years**
- **However, its client-side architecture does impose some limitations**
  - Requires connectivity to RDBMS from the client (client must have JDBC drivers installed)
  - Requires connectivity to cluster from the client
  - Requires user to specify RDBMS username and password
  - Difficult to integrate a CLI within external applications
- **Also tightly coupled to JDBC semantics**
  - A problem for NoSQL databases

Connection with both hadoop + RDBMS required.



# Sqoop 2 Architecture

- **Sqoop 2 is the next-generation version of Sqoop**
  - Client-server design addresses limitations described earlier
  - API changes also simplify development of other Sqoop connectors
- **Client requires connectivity only to the Sqoop server**
  - DB connections are configured on the server by a system administrator
  - End users no longer need to possess database credentials
  - Centralized audit trail
  - Better resource management
  - Sqoop server is accessible via CLI, REST API, and Web UI



# Essential Points

- **Sqoop exchanges data between a database and the Hadoop cluster**
  - Provides subcommands (*tools*) for importing, exporting, and more
- **Tables are imported using MapReduce jobs**
  - These are written as comma-delimited text by default
  - You can specify alternate delimiters or file formats
  - Uncompressed by default, but you can specify a codec to use
- **Sqoop provides many options to control imports**
  - You can select only certain columns or limit rows
  - Supports using joins in free-form queries
- **Sqoop 2 is the next-generation version of Sqoop**
  - Client-server design improves administration and resource management