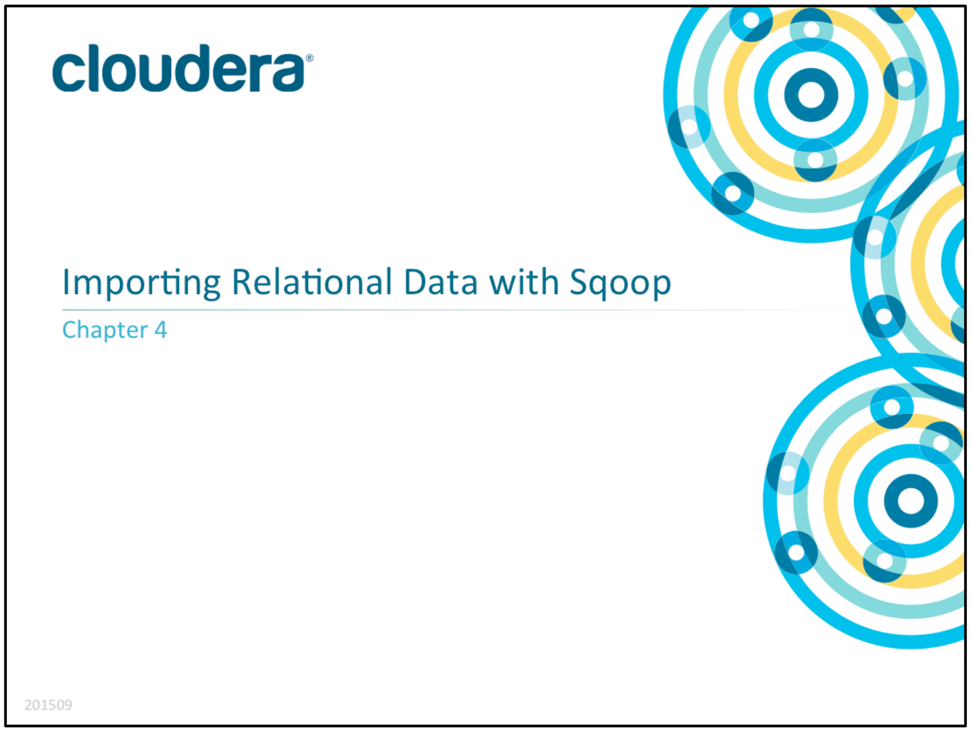


Importing Relational Data with Sqoop

Chapter 4

Course Chapters

Importing Relational Data with Apache Sqoop

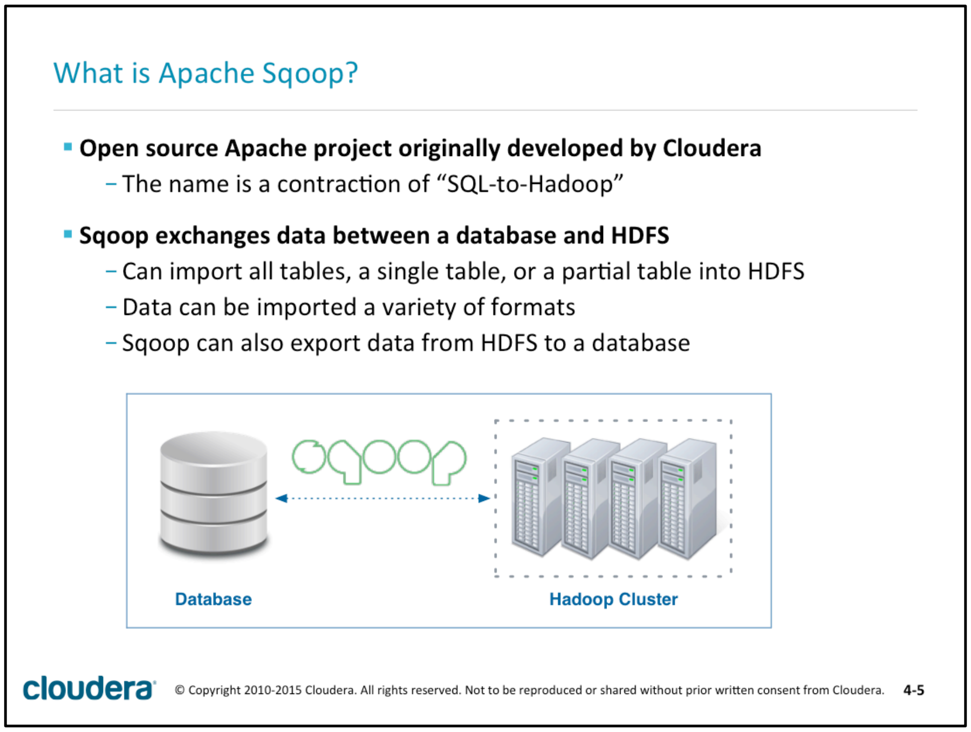
In this chapter you will learn

- **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**

Chapter Topics

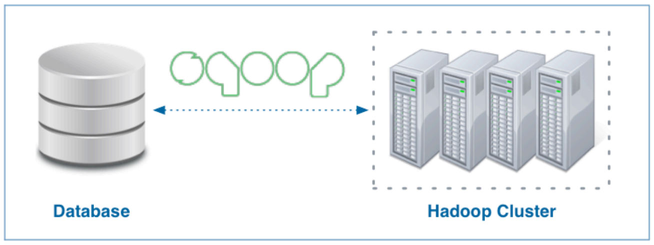
Importing Relational Data with Apache Sqoop | **Importing and Modeling Structured Data**

- **Sqoop Overview**
- Basic Imports and Exports
- Limiting Results
- Improving Sqoop's Performance
- Sqoop 2
- Conclusion
- Homework: Import Data from MySQL Using Sqoop

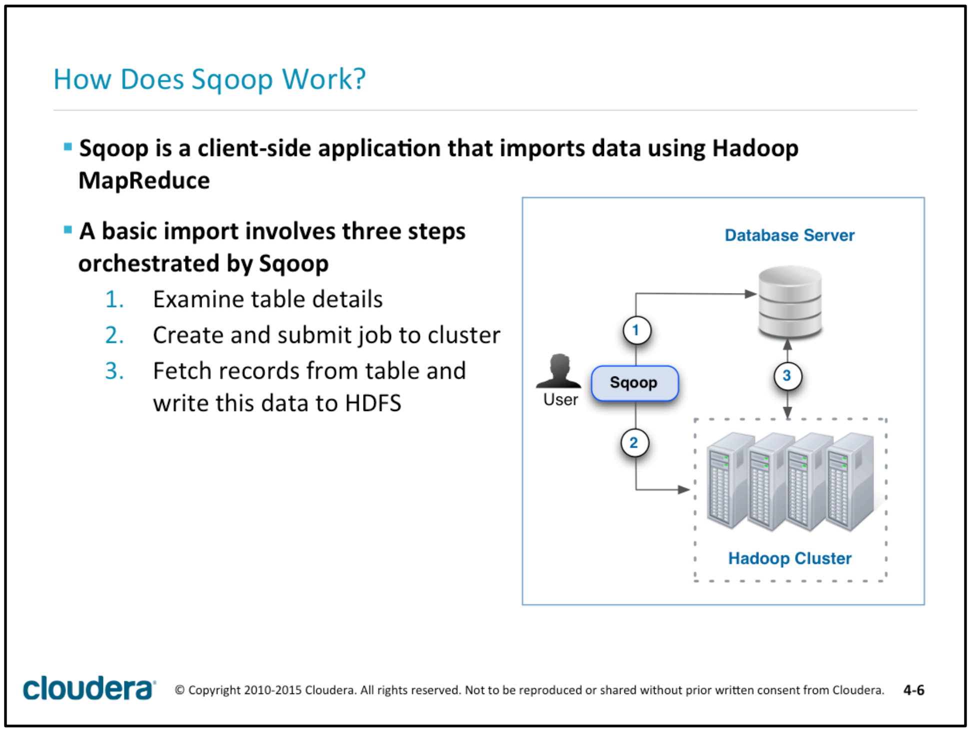


What is Apache Sqoop?

- **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

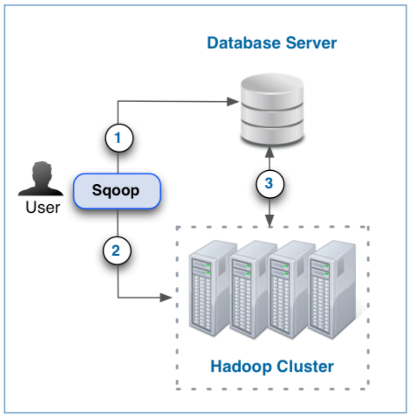


Note that the database is typically an RDBMS or EDW, but need not be. It's theoretically possible to import data from NoSQL databases (or potentially other databases). A good example of this is Couchbase [`http://www.couchbase.com/develop/connectors/hadoop`]. This will become relevant when we discuss Sqoop 2 later, since one of the limitations of Sqoop is that non-relational databases don't necessarily conform well to JDBC semantics.



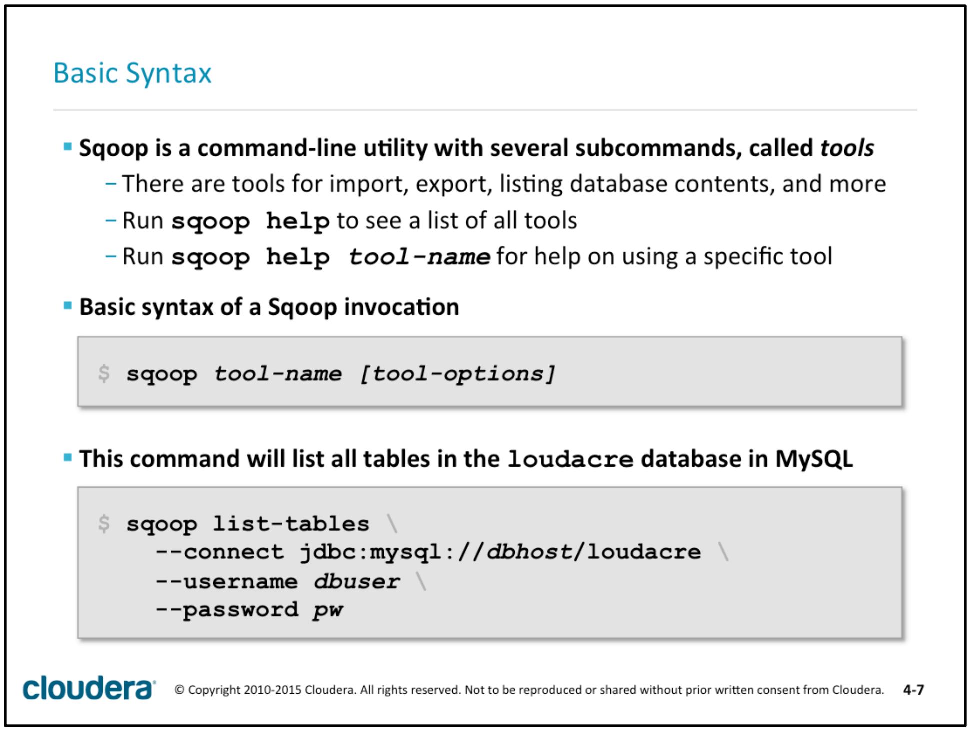
How Does Sqoop Work?

- **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



The diagram illustrates how Sqoop imports a table from a database into HDFS:

1. The user runs the Sqoop import command (e.g., on his or her workstation)
2. Sqoop connects to the database and gets metadata needed for the import job. This includes the table definition as well as determining the number of rows that the import will produce.
3. Sqoop submits a Map-only job (consisting of four map tasks by default, but configurable) that runs on the Hadoop cluster. Each task pulls a subset of the table from the database server.
4. The Map-only tasks write their data as output in 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**

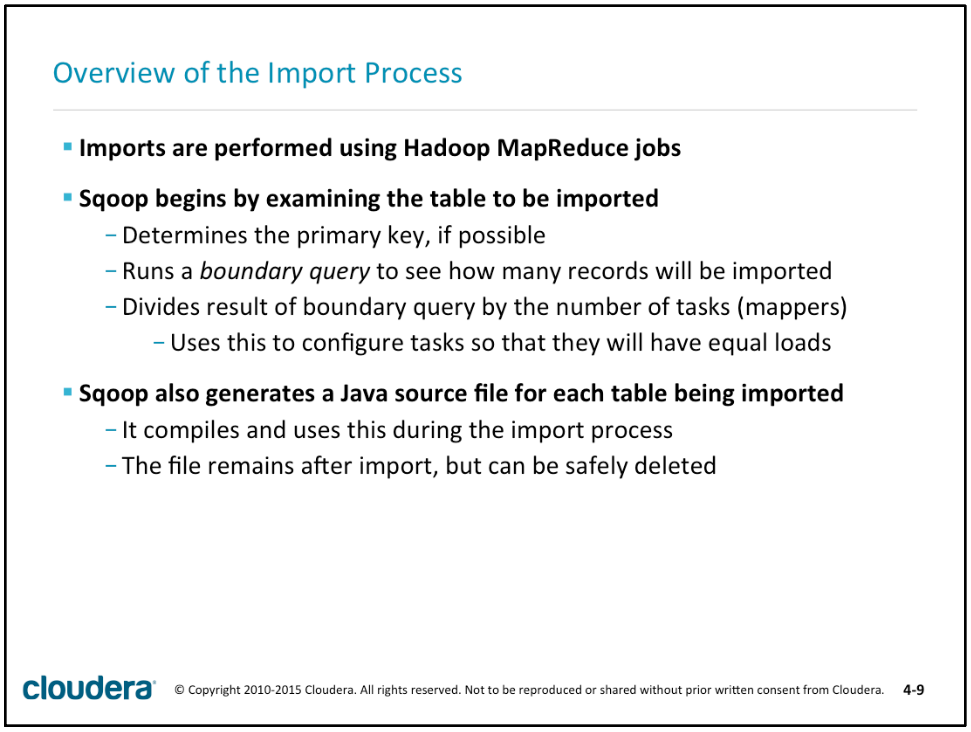
```
$ sqoop list-tables \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser \
    --password pw
```

Although we use the hostname "localhost" for the connection string, in real life the RDBMS you import data from will almost certainly be running on some machine that is not part of your cluster (i.e. it will be some server managed by your DBA).

Students will use the sqoop eval tool in the homework for this chapter.
From the Sqoop Users Guide:
https://sqoop.apache.org/docs/1.4.6/
SqoopUserGuide.html#_literal_sqoop_eval_literal

The eval tool allows users to quickly run simple SQL queries against a database; results are printed to the console. This allows users to preview their import queries to
ensure they import the data they expect. The eval tool is provided for evaluation purpose only.
You can use it to verify database connection from within the Sqoop or to test simple queries.
It's not suppose to be used in production workflows.

Chapter Topics



Overview of the Import Process

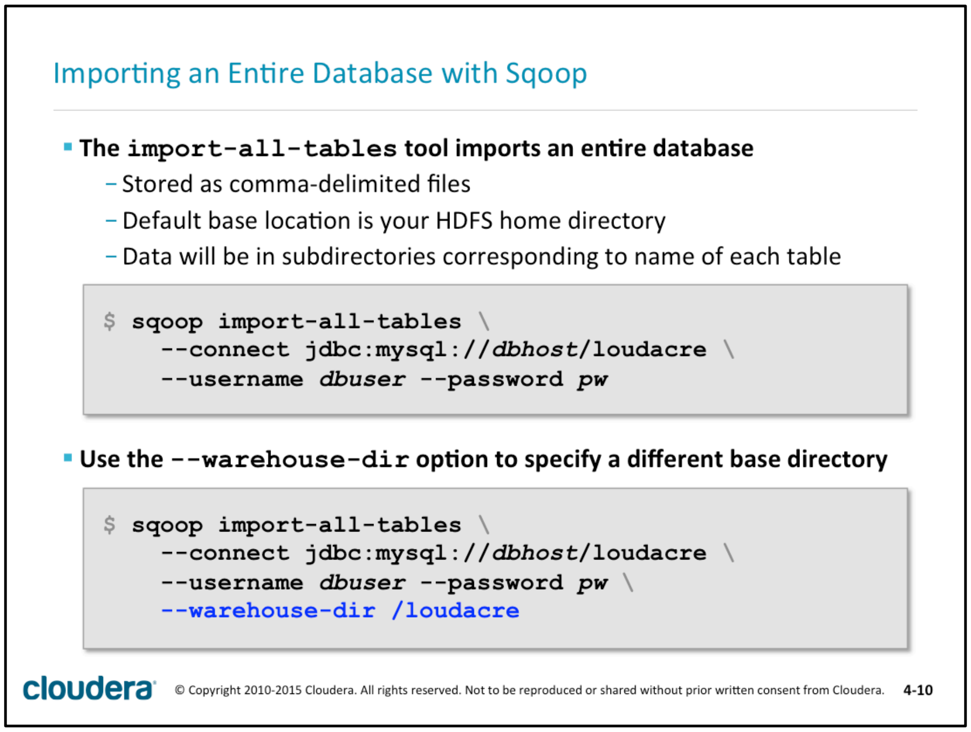
- **Imports are performed using Hadoop MapReduce jobs**
- **Sqoop begins by examining the table to be imported**
  - Determines the primary key, if possible
  - Runs a *boundary query* to see how many records will be imported
  - Divides result of boundary query by the number of tasks (mappers)
    - 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

cloudera

Sqoop determines the primary key by examining the table to be imported. If it cannot determine this (or if you have a non-numeric primary key), then you must specify a column to use with `--split-by` (this should have evenly distributed values, or the tasks will be skewed). The boundary query just selects the min and max values on that column, but you can also specify your own boundary query if desired using the –boundary-query flag.

Assuming evenly distributed values for the primary key, if you had one millions records in your table and four mappers, then each should have about 250,000 records to import.

From the Sqoop user guide, "A by-product of the import process is a generated Java class which can encapsulate one row of the imported table. This class is used during the import process by Sqoop itself. The Java source code for this class is also provided to you, for use in subsequent MapReduce processing of the data."



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

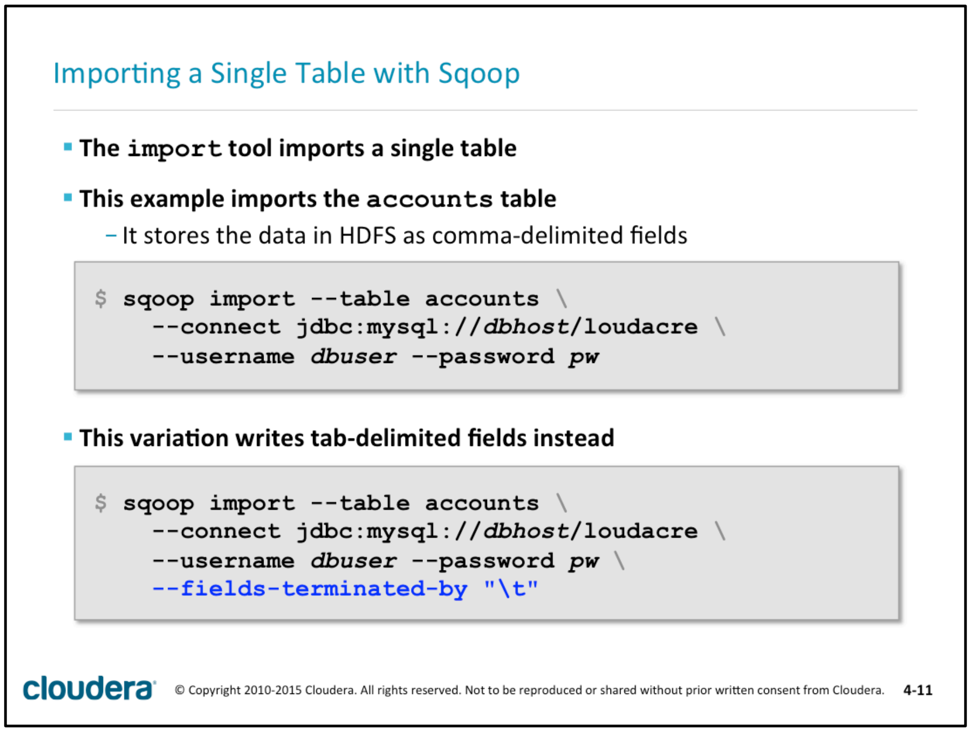
```
$ sqoop import-all-tables \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw
```

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

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

This example creates tab-delimited because we explicitly specify that. There will be a subdirectory of `/mydata` in HDFS for each table in the target database.

Note that we are showing the `--warehouse-dir` option, but there is also a `--target-dir` option which is sometimes used when importing a single table. The difference is that `--warehouse-dir` specifies the *parent* directory for the import, while `--target-dir` specifies the directory itself. That is, if you import a table '`foo`' using a `--warehouse-dir` value of '`/data`' then Sqoop will create '`/data/foo`' and import the data there. If you had used `--target-dir` with a value of '/data' then no subdirectory is created (i.e. your data will be directly inside the '`/data`' directory).



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"
```

NOTE: the tab sequence (\t) is quoted using double quotes because this enables it to work correctly whether you specify the import on the command line or in Oozie, while a single quote only works correctly when invoked from the command line.

Incremental Imports (1)

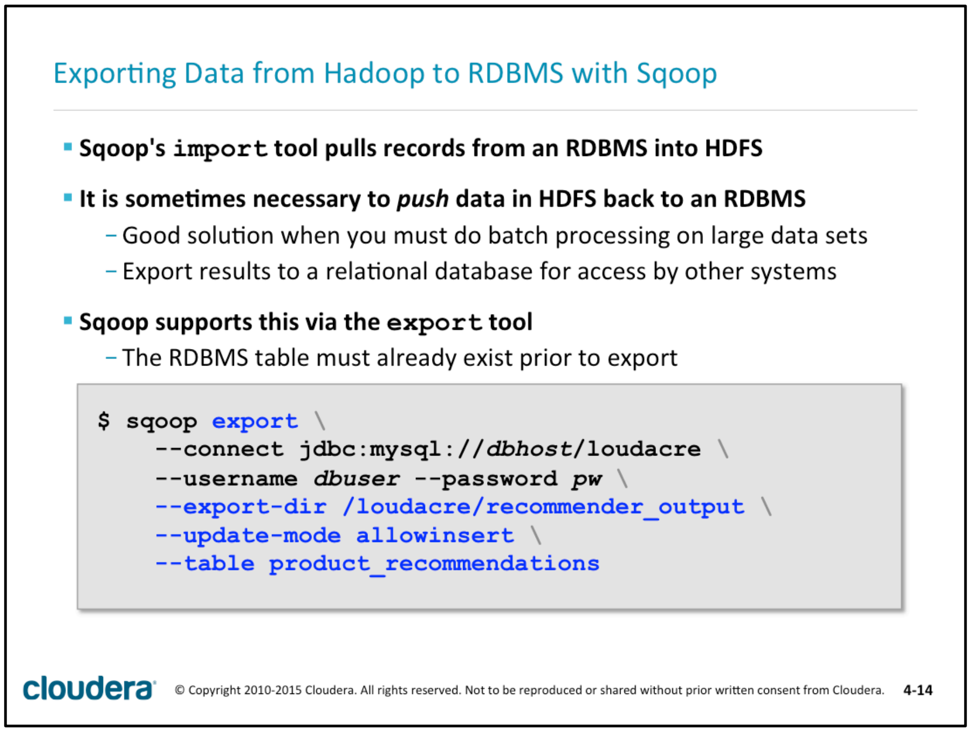
- **What if records have changed since last import?**
  - Could re-import all records, but this is inefficient
- **Sqoop's `incremental lastmodified` mode imports new and modified records**
  - Based on a timestamp in a specified column
  - You must ensure timestamps are updated when records are added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

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
```



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
```

While Sqoop is most often used to import data into Hadoop, it can also export data to a relational database. This is a good solution when you need to process data using one of CDH's batch processing tools (MapReduce, Hive, Pig, Crunch, etc.), which are much too slow for interactive use. You can do this processing as, for example, a nightly job, and then push the results (typically much smaller than the input data set, it certainly should not be terabytes of data!) back to an RDBMS where it can be accessed by other systems that don't interact with Hadoop. This is a fairly minimal example of the export command – it assumes that the data is already in the default (comma-delimited) format and that the columns in the files being imported are in the same order as in which they appear in the table (the table must exist before the export). It is possible to export data in other formats (such as Avro) back to the relational database. The `--update-mode` parameter shown here allows "upsert" mode, in which records in the target table will be updated if they already exist or inserted if they do not. The other allowed argument for this option is `updateonly`, which only updates existing records (i.e. it will not add any new records).

Chapter Topics

Importing Relational Data with Apache Sqoop	Importing and Modeling Structured Data

- Sqoop Overview
- Basic Imports and Exports
- **Limiting Results**
- Improving Sqoop's Performance
- Sqoop 2
- Conclusion
- Homework: Import Data from MySQL Using Sqoop

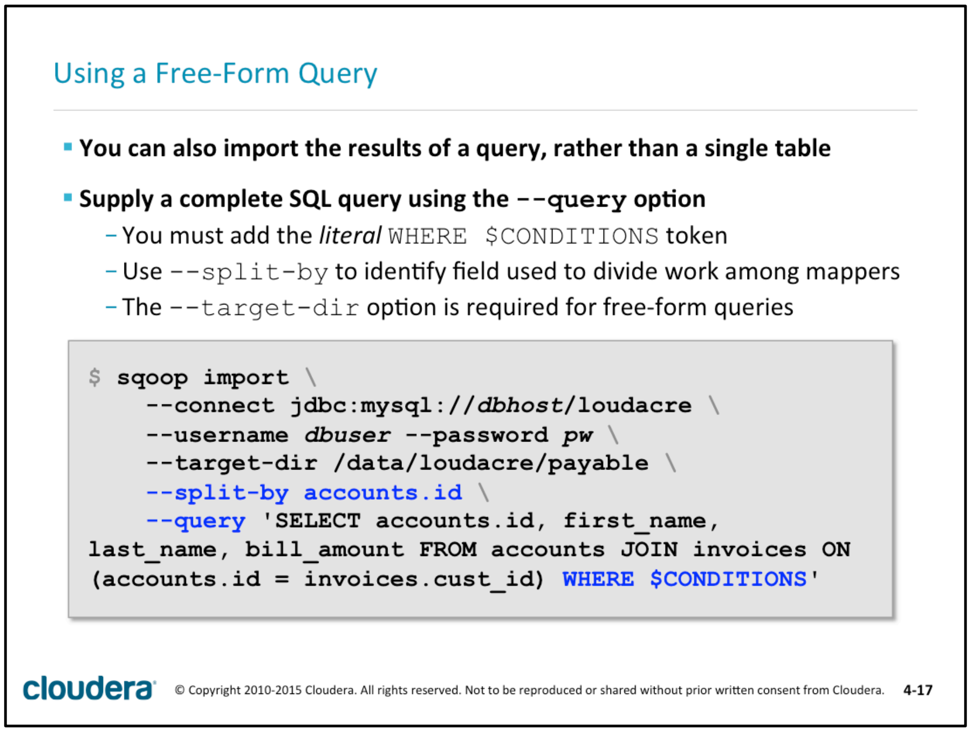
Importing Partial Tables with Sqoop

- **Import only specified columns from `accounts` table**

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

- **Import only matching rows from `accounts` table**

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



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'
```

WHERE $CONDITIONS needs to be included explicitly - typed as shown. This is where Sqoop itself will insert its range conditions for each tasks, e.g. task1: rows 1-1000, task2: rows 1001-2000, etc.
If using --query, don't combine with - -columns or - -where, they will be disregarded. $CONDITIONS doesn't include any of your own query conditions. If you want to specifying your own query conditions, see next slide…

The --target-dir option is required for free-form queries" – this argument is normally optional because it defaults to importing the data to a subdirectory of the user's home directory that is based on the table name, but in this case, there are multiple tables so this is ambiguous; therefore, the target directory must be explicitly specified.

More info: [
https://sqoop.apache.org/docs/1.4.2/
SqoopUserGuide.html#_free_form_query_imports].

Regarding the "literal" WHERE $CONDITIONS, note that we are using single quotes for the query argument. This is needed to prevent the UNIX shell from interpreting $CONDITIONS as a variable - we want Sqoop, rather the the shell, to interpret this value.

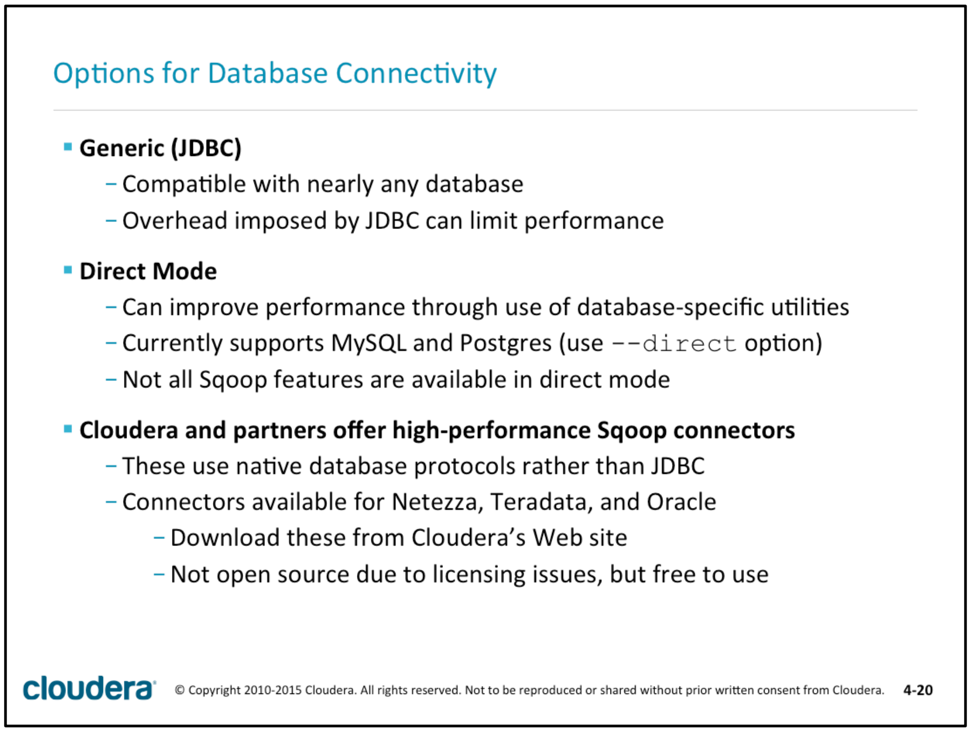
Using a Free-Form Query with WHERE Criteria

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

```
$ 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'
```

Chapter Topics

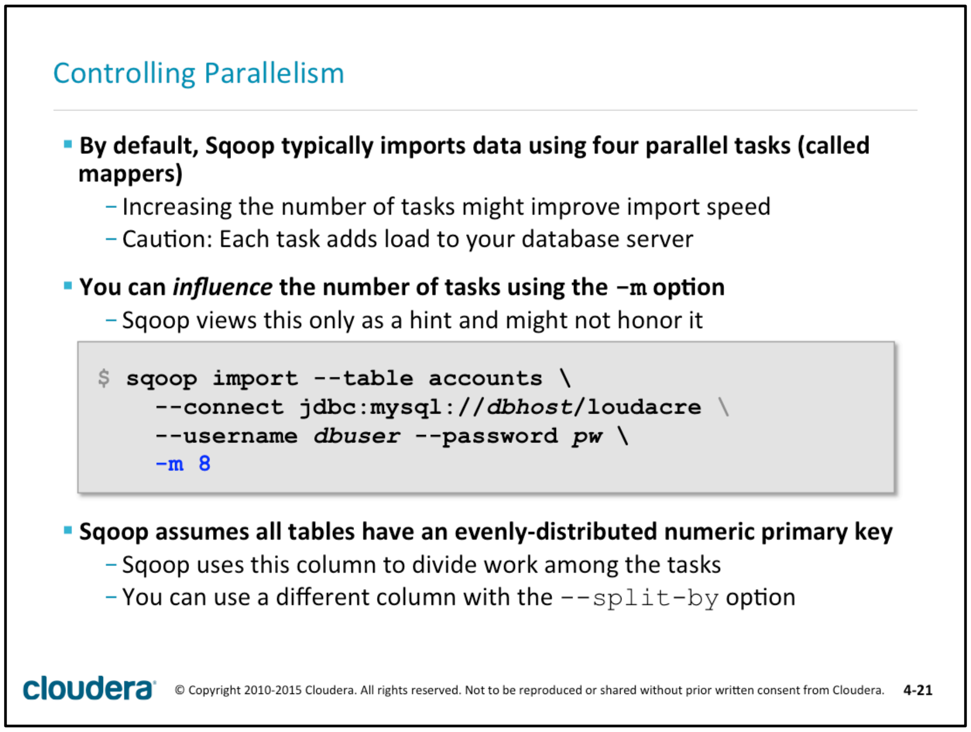
Importing Relational Data with Apache Sqoop	Importing and Modeling Structured Data



Options for Database Connectivity

- **Generic (JDBC)**
  - Compatible with nearly any database
  - Overhead imposed by JDBC can limit performance
- **Direct Mode**
  - Can improve performance through use of database-specific utilities
  - Currently supports MySQL and Postgres (use `--direct` option)
  - Not all Sqoop features are available in direct mode
- **Cloudera and partners offer high-performance Sqoop connectors**
  - These use native database protocols rather than JDBC
  - Connectors available for Netezza, Teradata, and Oracle
    - Download these from Cloudera's Web site
    - Not open source due to licensing issues, but free to use

Nearly every database server in common production use supports JDBC, so Sqoop is a nearly universal tool. Using JDBC can be slow, however, since it's a generic approach. Direct mode uses database-specific utilities (like `mysqldump`) to do faster imports, but not all databases are supported. Even where a database is supported with direct mode, not all Sqoop features are supported. For example, you cannot import a table in Avro or SequenceFile format when using direct mode, because mysqldump does not support those formats.



Controlling Parallelism

- **By default, Sqoop typically imports data using four parallel tasks (called 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

If specifying a different column using `--split-by` does not suffice, you can also use Sqoop's `--boundary-query` option to specify a different query (that returns two numeric columns) to help divide up the keyspace among mappers.

We haven't and won't explain mappers, because it isn't relevant, but you can mention that the "-m" option is named for the Map phase of the MapReduce paradigm.

By default, Sqoop will use four mappers and will split work between them by taking the minimum and maximum values of the primary key column and dividing the range equally among the mappers. The split-by parameter lets you specify a different column for splitting the table between mappers. As we note below, one reason to specify a parameter for split-by is to avoid data skew. Note that each mapper will have its own connection to the database and each will retrieve its portion of the table by specifying its portion limits in a "where" clause. It is important to choose a split column that has an index or is a partition key to avoid each mapper having to scan the entire table. If no such key exists, specifying only one mapper is the preferred solution.

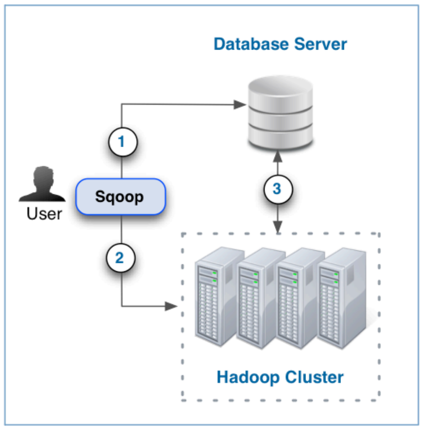
Chapter Topics

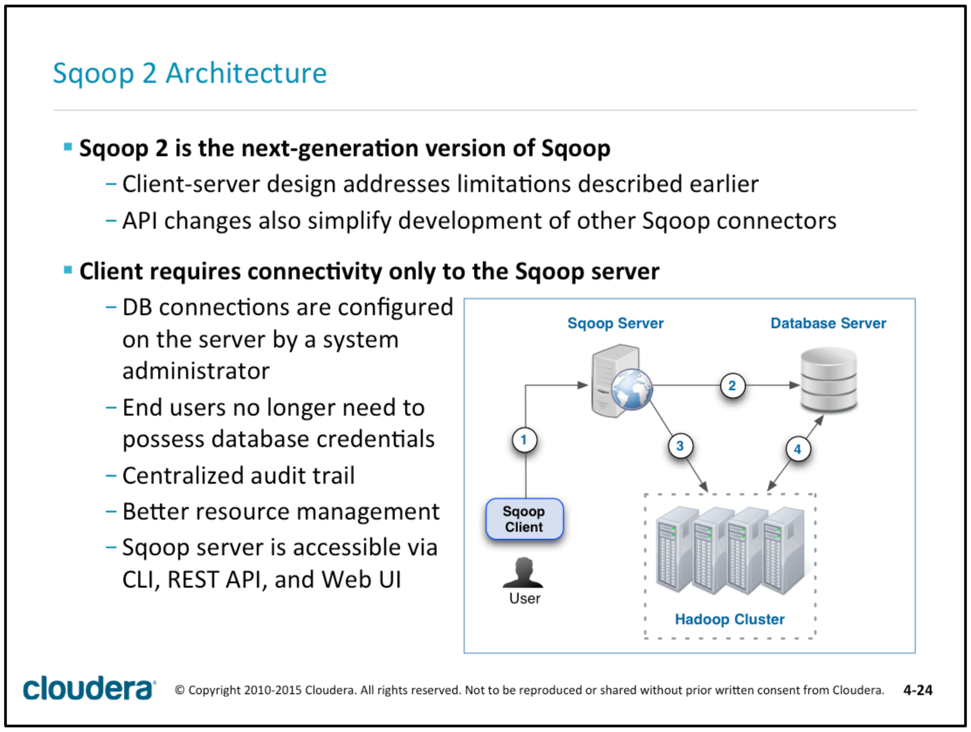
Importing Relational Data with Apache Sqoop	Importing and Modeling Structured Data

- Sqoop Overview
- Basic Imports and Exports
- Limiting Results
- Improving Sqoop's Performance
- **Sqoop 2**
- Conclusion
- Homework: Import Data from MySQL Using Sqoop

Limitations of Sqoop

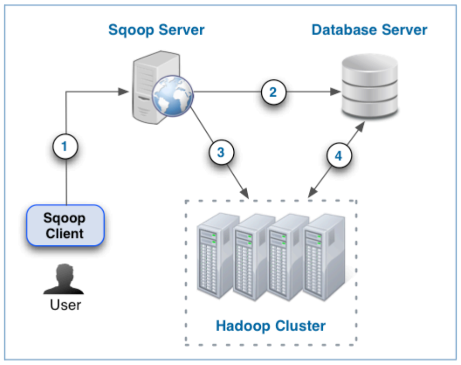
- **Sqoop is stable and has been used successfully in production for years**
- **However, its client-side architecture does impose some limitations**
  - Requires connectivity to RDBMS from the client (client must have JDBC drivers installed)
  - Requires connectivity to cluster from the client
  - Requires 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





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

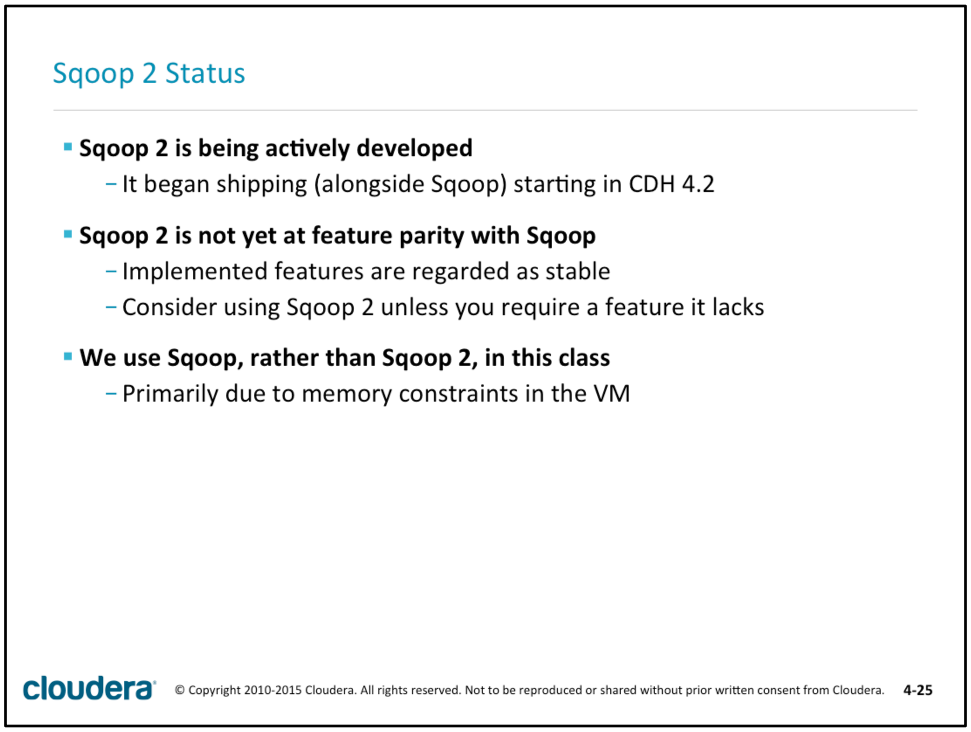


For the sake of simplicity, I have left out details of the metastore even though this fairly important in Sqoop 2, since we're not using Sqoop 2 in class. Whereas Sqoop has the *option* of saving jobs, everything is a saved job in Sqoop 2. See here [`http://sqoop.apache.org/docs/1.99.3/Sqoop5MinutesDemo.html`] for a quick tutorial on using Sqoop 2.

The steps for a basic import in Sqoop 2 are illustrated here:

1. Sqoop 2 client connects to Sqoop 2 server
2. Sqoop 2 server connects to database to get details about table to be imported
3. Sqoop 2 server submits MapReduce job to server (it seems that Sqoop 2 uses MR and not just Map-only jobs)
4. MR job pulls records from the database and writes them to HDFS

"API changes also simplify development of Sqoop connectors" means that unlike in Sqoop, connectors in Sqoop 2 are responsible only for transporting data. They "Better resource management" includes things such as being able to limit the number of database connections (i.e. to prevent the RDBMS from going down if a sets the number of map tasks too high).



Sqoop 2 Status

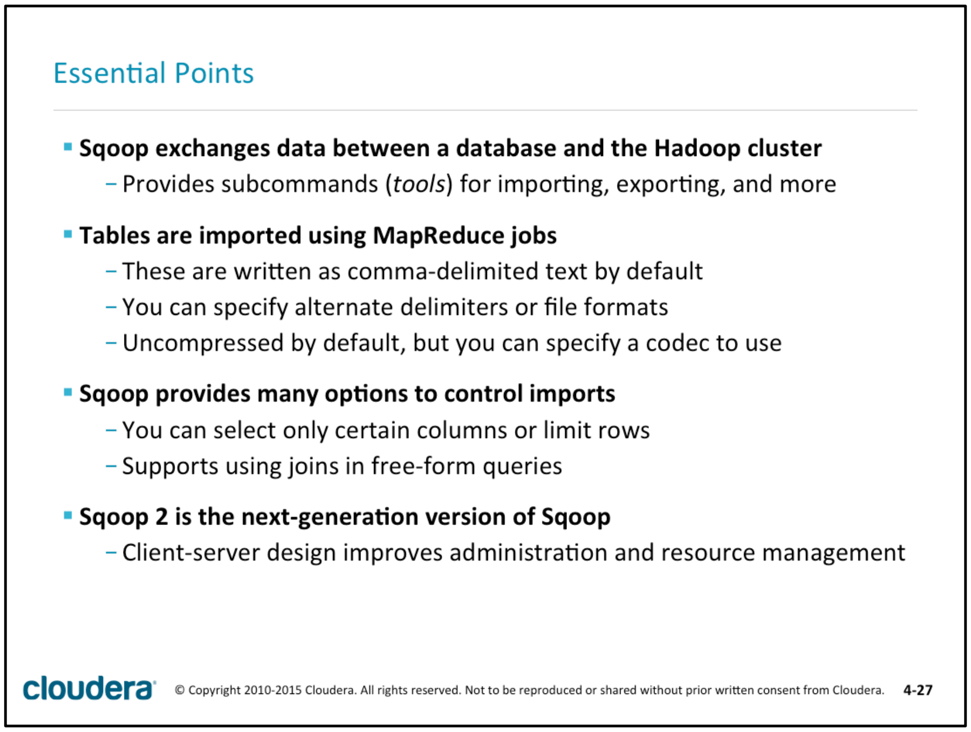
- **Sqoop 2 is being actively developed**
  - It began shipping (alongside Sqoop) starting in CDH 4.2
- **Sqoop 2 is not yet at feature parity with Sqoop**
  - Implemented features are regarded as stable
  - Consider using Sqoop 2 unless you require a feature it lacks
- **We use Sqoop, rather than Sqoop 2, in this class**
  - Primarily due to memory constraints in the VM

cloudera

http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_ig_sqoop_vs_sqoop2.html
The current versions, as of this writing (August 2015), are numbered like 1.99.x to indicate that Sqoop 2 has not officially been released. (CDH 5.4 includes Sqoop 1.99.5) Eventually, though, Sqoop will be phased out, so it's good for students to have a basic understanding of how it works. We do not use it in this class mainly because we're constrained by memory in the VM and adding another server process would make the problem worse. The advice from our product team as of December 2013 is that you should use Sqoop 2 unless you need a feature that it does not yet implement (see the CDH docs for a table listing what those are in a given version of CDH). In other words, Sqoop 2 does not yet have feature parity with Sqoop, but what is there works well.

Chapter Topics

Importing Relational Data with Apache Sqoop	Importing and Modeling Structured Data



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

REVIEW QUESTIONS (optional, but will help the instructor gauge whether the class is absorbing the material, and to provide some elasticity if you're running ahead of schedule):

- **Q1**: What does Sqoop do to try and balance the load of a database import among mappers? (Answer: it runs a boundary query to determine how many records will be imported, and then divides this among the different mappers for the job)
- **Q2**: What are three file formats that Sqoop can write data as when importing a table? (Answer: delimited, sequence file, and Avro)
- **Q3**: Can you import the results of a query using Sqoop rather than just a single table? (Answer: Yes, by using a free-form query)

Bibliography

The following offer more information on topics discussed in this chapter

- **Sqoop User Guide**
  - `http://tiny.cloudera.com/sqoopuserguide`
- ***Apache Sqoop Cookbook*** **(published by O'Reilly)**
  - `http://tiny.cloudera.com/sqoopcookbook`
- **A New Generation of Data Transfer Tools for Hadoop: Sqoop 2**
  - `http://tiny.cloudera.com/adcc05c`

Chapter Topics

Homework: Import Data from MySQL Using Sqoop

- **In this homework , you will**
  - Use Sqoop to import web page and customer account data from an RDBMS to HDFS
  - Perform incremental imports of new and updated account data
- **Please refer to the Homework description**