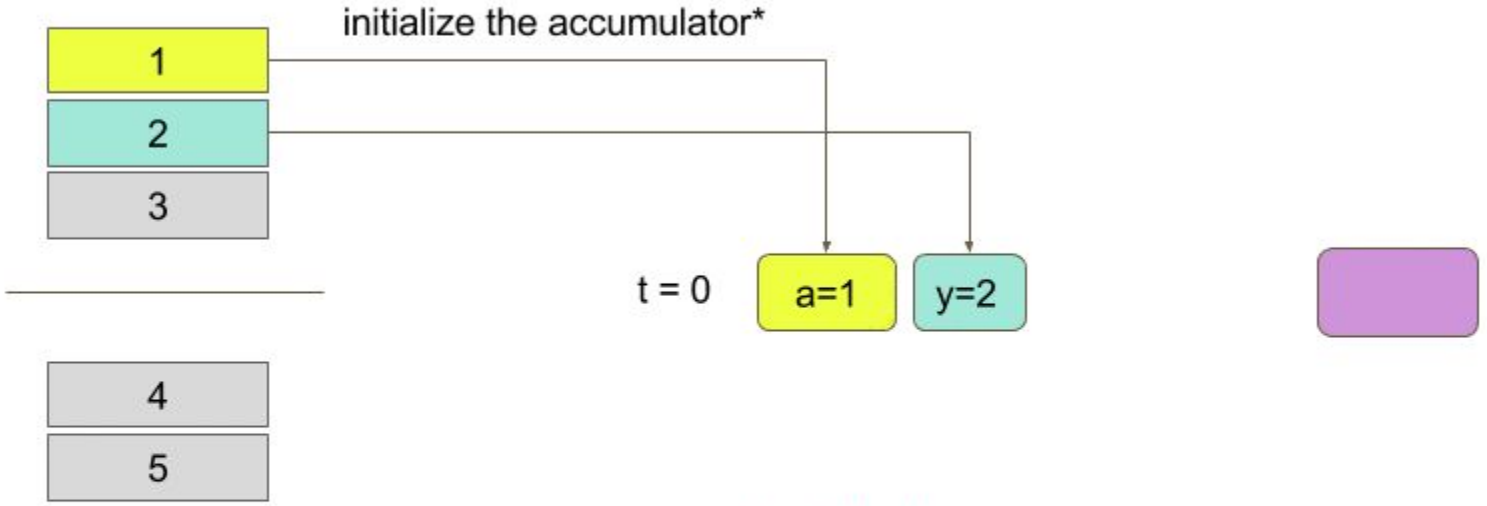
Previously - questions about RDDs

How does reduce get initialized?

What is the longevity of RDDs after an action?

I’m still aggravated by keys!

reduce (an action) - revised



reduce (***f(x,y)***)

```
reduce((a,n) -> a + n);
```

* used to contain the results of an arithmetical or logical operation.

```java
public class BasicMapThenFilter {

    public static void main(String[] args) throws Exception {

        JavaSparkContext sc = new JavaSparkContext("local", "basicmapfilter");

        JavaRDD<Integer> rdd = sc.parallelize(Arrays.asList(1, 2, 3, 4, 5, 6, 7));

        Accumulator<Integer> acc = sc.accumulator(0);

        JavaRDD<Integer> squared = rdd.map(new Function<Integer, Integer>() {
            @Override
            public Integer call(Integer x) {
                acc.add(1);
                return x * x;
            }
        });
        JavaRDD<Integer> result = squared.filter(new Function<Integer, Boolean>() {
            @Override
            public Boolean call(Integer x) {
                return x % 2 != 0;
            }
        });
        JavaRDD<Integer> equivalentResult = squared.filter(x -> x % 2 != 0);

        System.out.println("anonymous function results:  " + result.collect() + "\nlambda results:  "
                + equivalentResult.collect());

        System.out.println("Number of times accumulator is called:  " + acc.value());

    }
}
```

size of input = 7

Using accumulator to count operations

```
2016-11-14 12:47:27,333 INFO  [main] scheduler.DAGScheduler
anonymous function results:  [1, 9, 25, 49]
lambda results:  [1, 9, 25, 49]
Number of times accumulator is called:  14
2016-11-14 12:47:27,335 INFO  [Thread-3] spark.SparkContext
```

Input contains 7 numbers

Number of operations on input to calculate RDD squared: 14

Number of times RDD squared is calculated: twice

Upshot: if you don't cache the RDD before you call an action, it will be recalculated if it is used in a later action.

Today's Agenda

IO for Hive

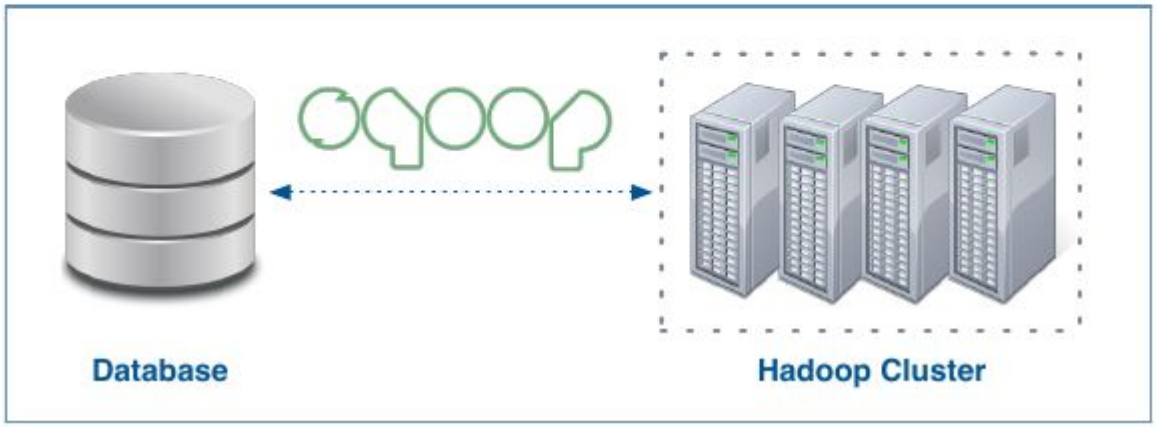
- Sqooping data into Hive
- A brief intro to Hadoop IO types
  - Sequence Files
  - Parquet
  - Avro
- Intro to using Hive and Hive SerDe

Sqoop

Importing and exporting data from an RDBMS

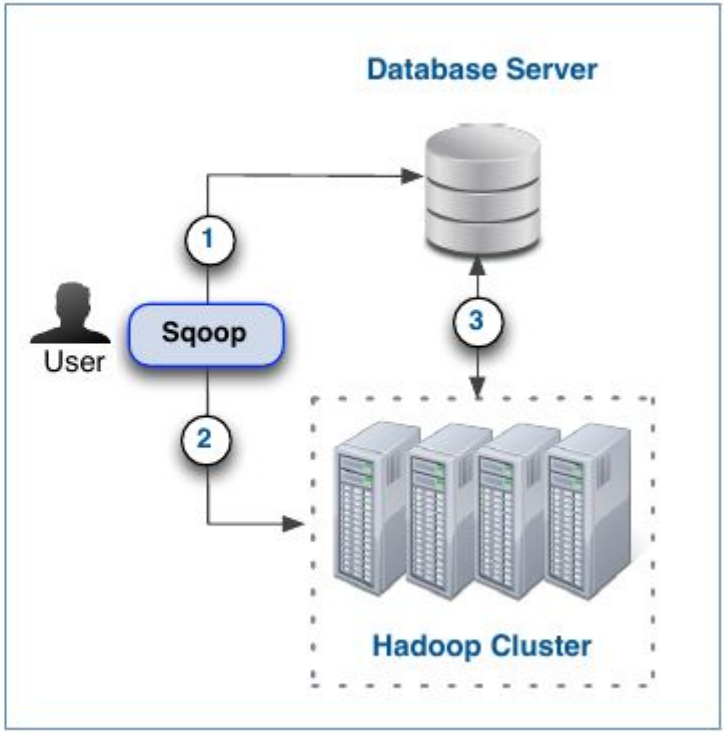
Sqoop functionality

- **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

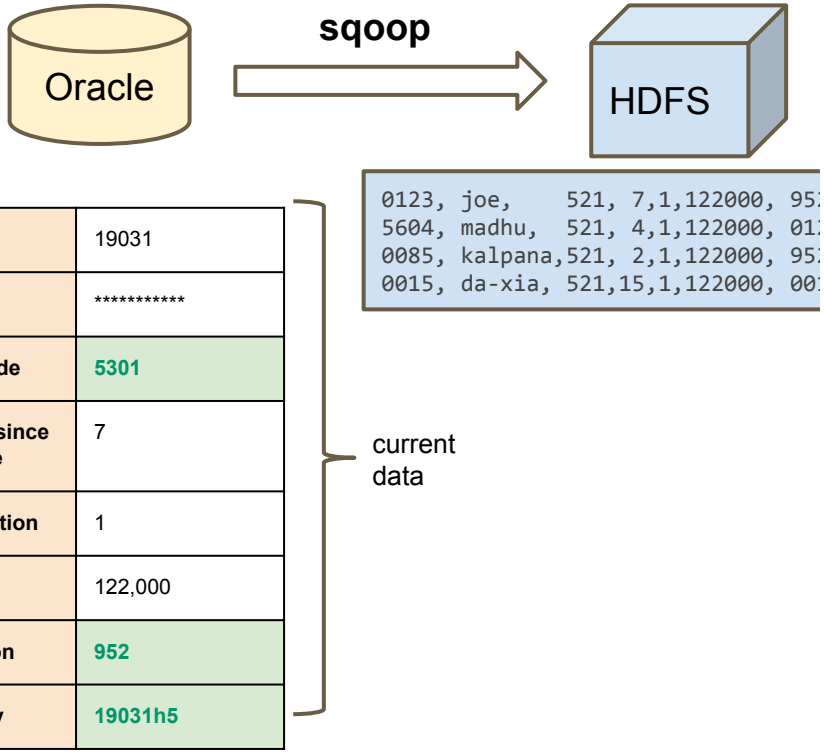


How does Sqoop Work?

- **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



Loading from Oracle with Sqoop



- **Transfers data between Hadoop and databases.**
  - Import data from RDBMS to HDFS
  - Analyze the data on Hadoop cluster
  - Export the data back into an RDBMS.
- **Automates the process**
  - The database provides table schema
  - Sqoop uses Map Reduce for import and export
    - parallel operation
    - fault tolerance

Sqoop: importing to Hadoop

- Imports to HDFS, Hive, HCatalog, HBase and Accumulo
  - Can import as text (default), Avro, SequenceFile, or Parquet
- Imports from an RDBMS
  - Just one table or all tables in a database
  - Parts of a table: Sqoop supports a WHERE clause
  - Incremental imports
- Use MapReduce to actually import the data
  - Determines number of tasks, with balanced load
  - Throttles number of Mappers to avoid DDoS scenarios
- Usually uses a JDBC interface
  - works with most JDBC-compatible databases
  - many RDBMS (Oracle, Teradata, mySQL) have specialized connectors

Importing a table

Import table employee from database called HR_DB using Oracle

```
$ sqoop import --username mcorey --password secret \
    --connect jdbc:oracle:thin:@dbhost:1521/HR_DB \
    --table employee
```

- By default, table employee imports to directory employee in your home directory in HDFS.
- For example, for user cloudera, the imported table is written to /user/cloudera/employee/(files).

Importing An Entire Database with Sqoop

```
$ sqoop import-all-tables \
    --username mcorey --password secret \
    --connect jdbc:oracle:thin://hostname/HR_DB \
    --warehouse-dir /my_hr_data
```

You must specify the parent directory with --warehouse-dir

Notice, you cannot use --target-dir in this context

Example: Selecting data to import

Import employee with titlecode of geophysicist (5301):

```
$ sqoop import --connect jdbc:oracle:thin:@dbhost:1521/HR_DB \
    --username mcorey --password secret \
    --table employee \
    --columns "employee_id, titlecode, eval, salary" \
    --where "titlecode =5301"
```

By default, mporting table is stored as a CSV file in a directory in HDFS.

Summary: selecting data to import

- Select table using --table argument.
  - Example, `--table employee`
  - Default: all columns are imported.
- Select columns using --columns
  - Example, `--columns "employee_id, titlecode, eval, salary"`
- Select rows by adding a --where clause
  - Example: `--where "title =5301"`
  - Sqoop *generates* SQL select statement.

Importing with a SQL query

- Specify a SQL statement with the --query argument.
- Every query ends with WHERE $CONDITIONS
  - Sqoop replaces $CONDITIONS with a split on the input
  - You tell Sqoop which column to use for splitting with --split-by.
- Specify a destination directory with --target-dir.

```
$ sqoop import \
... do stuff to connect ... \
    --query 'SELECT employee.*, salary.* \
    FROM employee JOIN salary on (employee.id == salary.id) \
    WHERE $CONDITIONS' \
    --split-by employee.id
    --target-dir /user/corey/employee_salaries
```

Incremental Imports using append

- **What if new records are added to the database?**
  - Could re-import all records, but this is inefficient
- **Sqoop's incremental append mode imports only *new* records**
  - Based on value of last record in specified column

```
$ sqoop import --table employee \
    --connect jdbc:mysql://hostname/HR_DB \
    --username mcorey --password secret \
    --table employee \
    --incremental append \
    --check-column emp_id \
    --last-value 17138
```

Exporting data from Hadoop to RDBMS

- **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:oracle.thin:@db:2020/HR_DB \
  --username mcorey --password *********
  --export-dir /user/cloudera/new_hires \
  --update-mode allowinsert \
  --table employees
```

Sqoop user documentation

https://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html

-- great doc, thorough and well-written

Take 5

Yep, time for a break

Additional reference slides for Sqoop

Sqoop commands covered today

Connecting

`--connect <jdbc-uri>`	Specify JDBC connect string
`--password <password>`	Set authentication password
`--username <username>`	Set authentication username

Import specification

`--table <table name>`	Set the name of the table to import
`--columns <column names>`	Specify columns to be imported from a table
`--query <SQL query>`	Import the results of a SQL query
`--where <where clause>`	Import using a where clause to refine results

How data is formatted after import

`--as-sequencefile`	Imports data as sequence file(s)
`--as-textfile`	Imports data as plain text (default)
`--as-parquetfile`	Imports data as parquet file
`--as-avrodatafile`	Imports data as avro data

Export specification

`--export-dir`	HDFS source path for the export
`--update-mode`	Specify how updates are performed for new rows with new keys

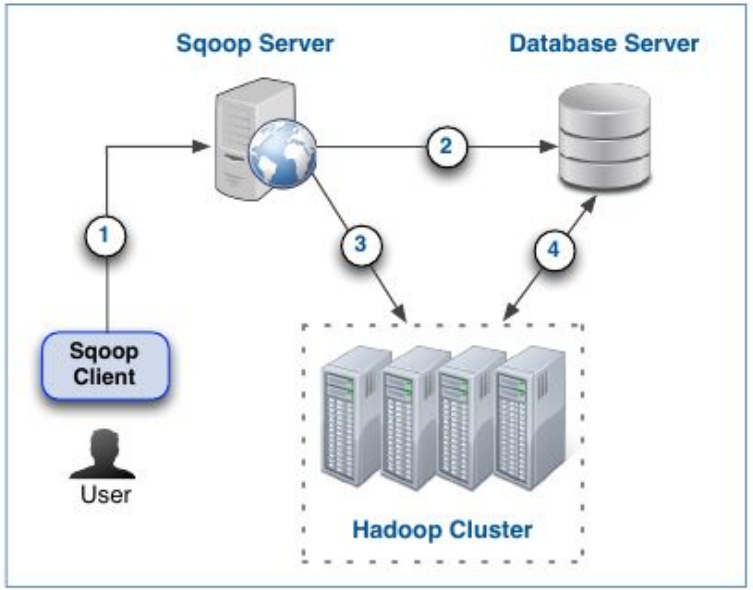
Options for Database Connectivity

- **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

http://blog.cloudera.com/blog/2013/09/understanding-connectors-and-drivers-in-the-world-of-sqoop/

Sqoop 2 - not feature complete

- **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



Incremental Imports using last modified

- **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 employee \
    --connect jdbc:mysql://hostname/HR_DB \
    --username mcorey --password secret \
    --incremental lastmodified \
    --check-column mod_dt \
    --last-value '2016-04-27 16:00:00'
```

More examples

Importing as a parquet file

```
$ sqoop import --connect jdbc:oracle.thin:@db:2020/HR_DB \
  --table employee \
  --as-parquetfile
```

Specifying the delimiters when importing text:

```
$ sqoop import --connect jdbc:oracle.thin:@db:2020/HR_DB \
  --table employee \
  --fields-terminated-by '\t' \
  --lines-terminated-by '\n' \
  --optionally-enclosed-by '\"'
```