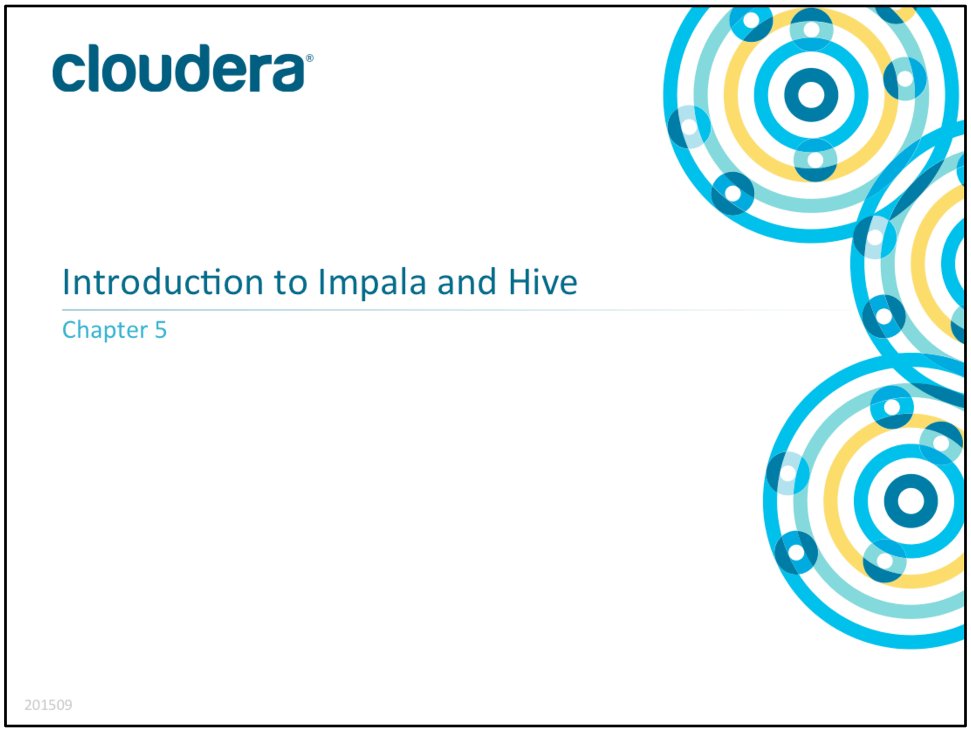
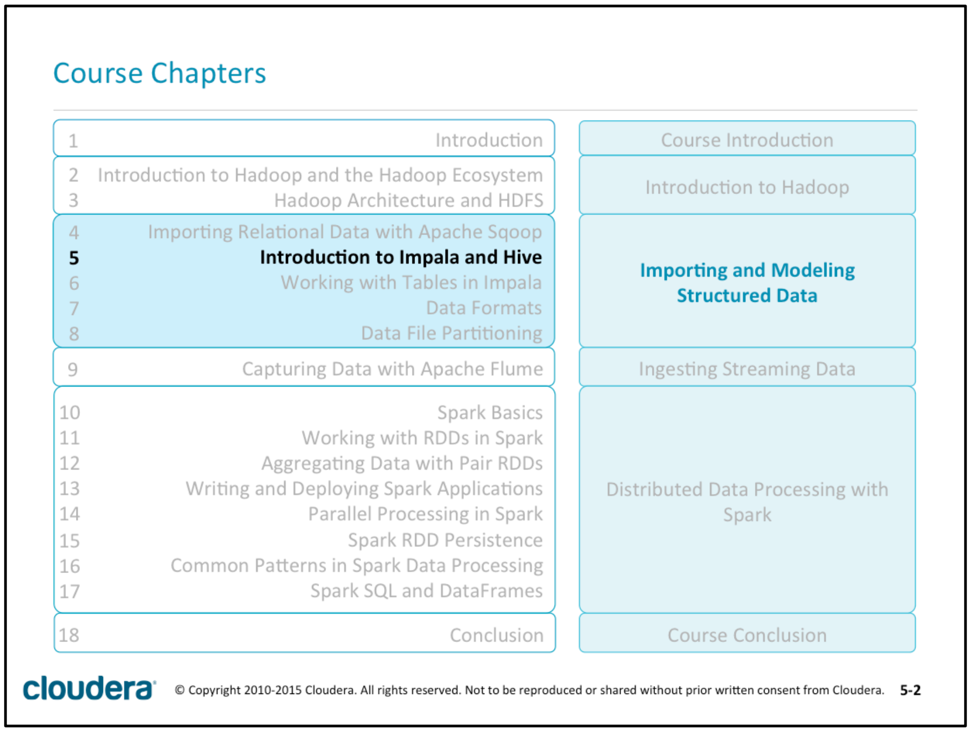


cloudera

Introduction to Impala and Hive

Chapter 5

201509



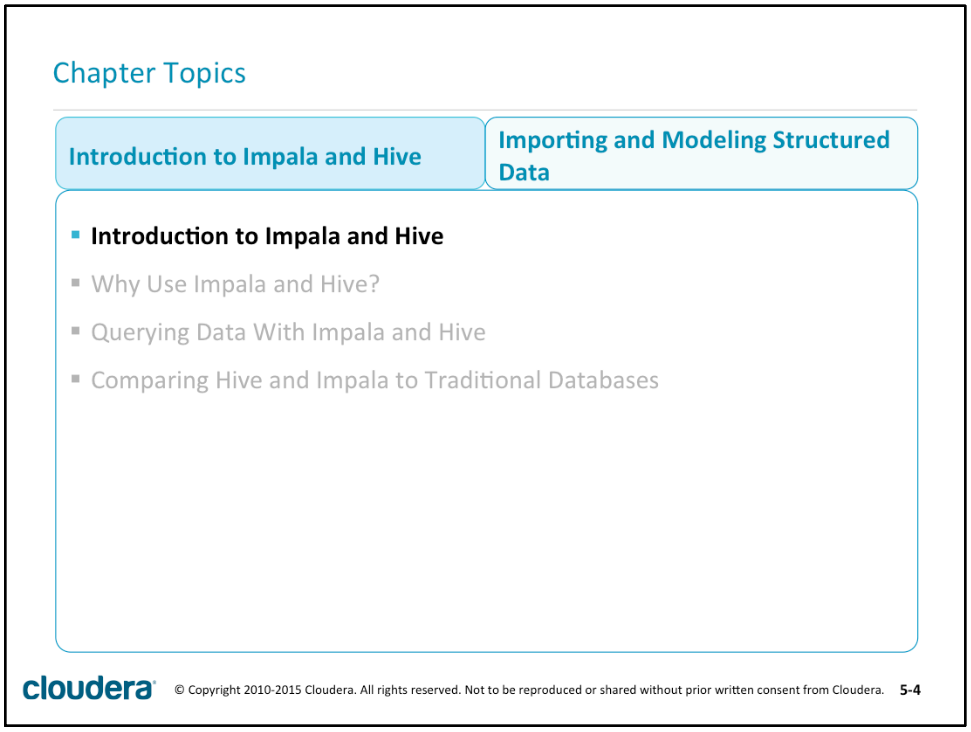
Course Chapters

1	Introduction	Course Introduction
2	Introduction to Hadoop and the Hadoop Ecosystem	Introduction to Hadoop
3	Hadoop Architecture and HDFS	
4	Importing Relational Data with Apache Sqoop	**Importing and Modeling Structured Data**
5	**Introduction to Impala and Hive**	
6	Working with Tables in Impala	
7	Data Formats	
8	Data File Partitioning	
9	Capturing Data with Apache Flume	Ingesting Streaming Data
10	Spark Basics	Distributed Data Processing with Spark
11	Working with RDDs in Spark	
12	Aggregating Data with Pair RDDs	
13	Writing and Deploying Spark Applications	
14	Parallel Processing in Spark	
15	Spark RDD Persistence	
16	Common Patterns in Spark Data Processing	
17	Spark SQL and DataFrames	
18	Conclusion	Course Conclusion

Introduction to Impala and Hive

In this chapter you will learn

- **What Hive is**
- **What Impala is**
- **How Impala and Hive Compare**
- **How to query data using Impala and Hive**
- **How Hive and Impala differ from a relational database**
- **Ways in which organizations use Hive and Impala**



Chapter Topics

Introduction to Impala and Hive | Importing and Modeling Structured Data

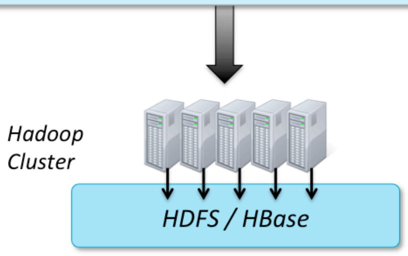
- **Introduction to Impala and Hive**
- Why Use Impala and Hive?
- Querying Data With Impala and Hive
- Comparing Hive and Impala to Traditional Databases

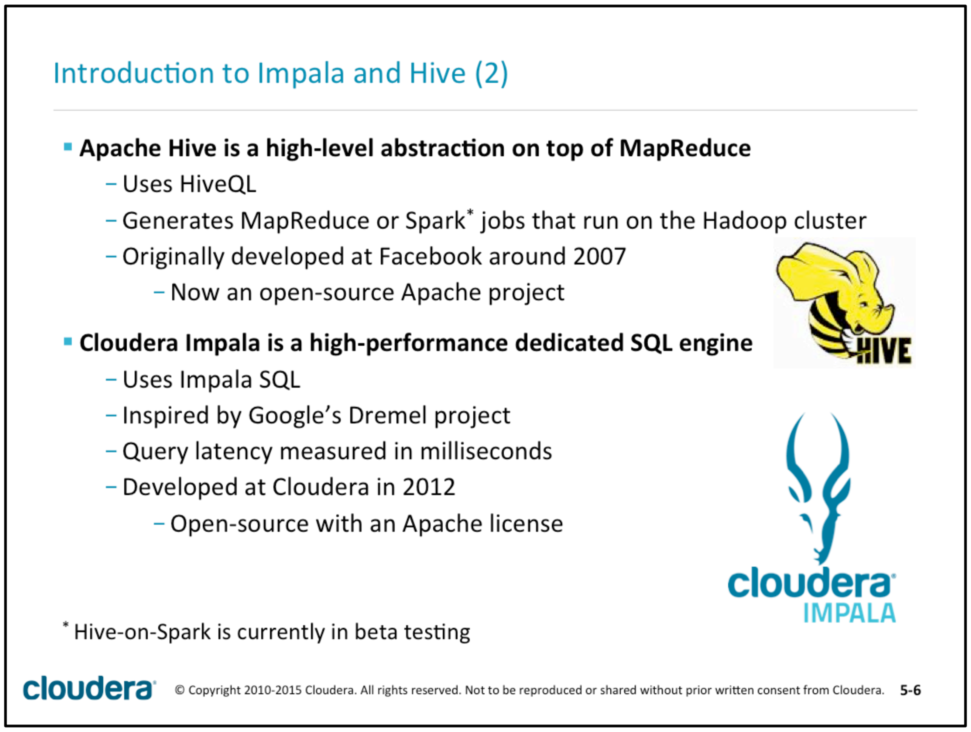
Hive was developed as an internal project at Facebook in 2007 and described to the public in this Facebook Engineering note from 2009 [`https://www.facebook.com/note.php?note_id=89508453919`].

Introduction to Impala and Hive (1)

- **Impala and Hive are both tools that provide SQL querying of data stored in HDFS / HBase**

```
SELECT zipcode, SUM(cost) AS total
FROM customers
JOIN orders
ON (customers.cust_id = orders.cust_id)
WHERE zipcode LIKE '63%'
GROUP BY zipcode
ORDER BY total DESC;
```



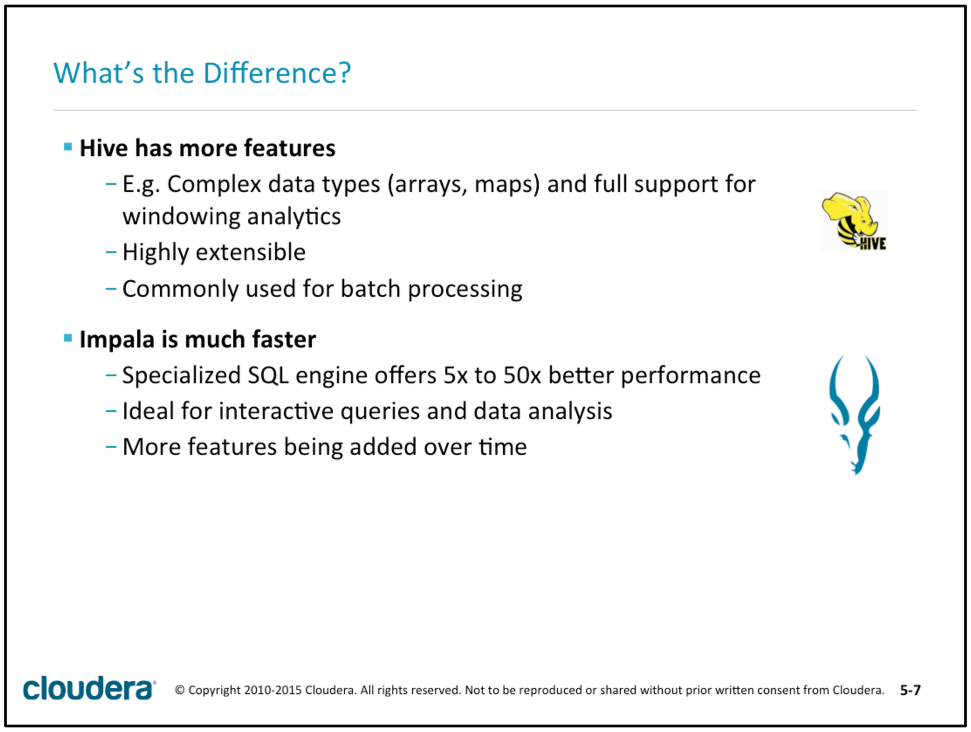


Introduction to Impala and Hive (2)

- **Apache Hive is a high-level abstraction on top of MapReduce**
  - Uses HiveQL
  - Generates MapReduce or Spark* jobs that run on the Hadoop cluster
  - Originally developed at Facebook around 2007
    - Now an open-source Apache project
- **Cloudera Impala is a high-performance dedicated SQL engine**
  - Uses Impala SQL
  - Inspired by Google's Dremel project
  - Query latency measured in milliseconds
  - Developed at Cloudera in 2012
    - Open-source with an Apache license

* Hive-on-Spark is currently in beta testing

Key point: Hive was developed FIRST, so many things are named "Hive" that actually apply to both, like Hive Metastore, covered soon.



What's the Difference?

- **Hive has more features**
  - E.g. Complex data types (arrays, maps) and full support for windowing analytics
  - Highly extensible
  - Commonly used for batch processing
- **Impala is much faster**
  - Specialized SQL engine offers 5x to 50x better performance
  - Ideal for interactive queries and data analysis
  - More features being added over time

There are lots of technical differences between the two, but the differences we care about at a high level come down to speed vs. features
Impala v. Hive benchmarks: http://blog.cloudera.com/blog/2014/01/impala-performance-dbms-class-speed/

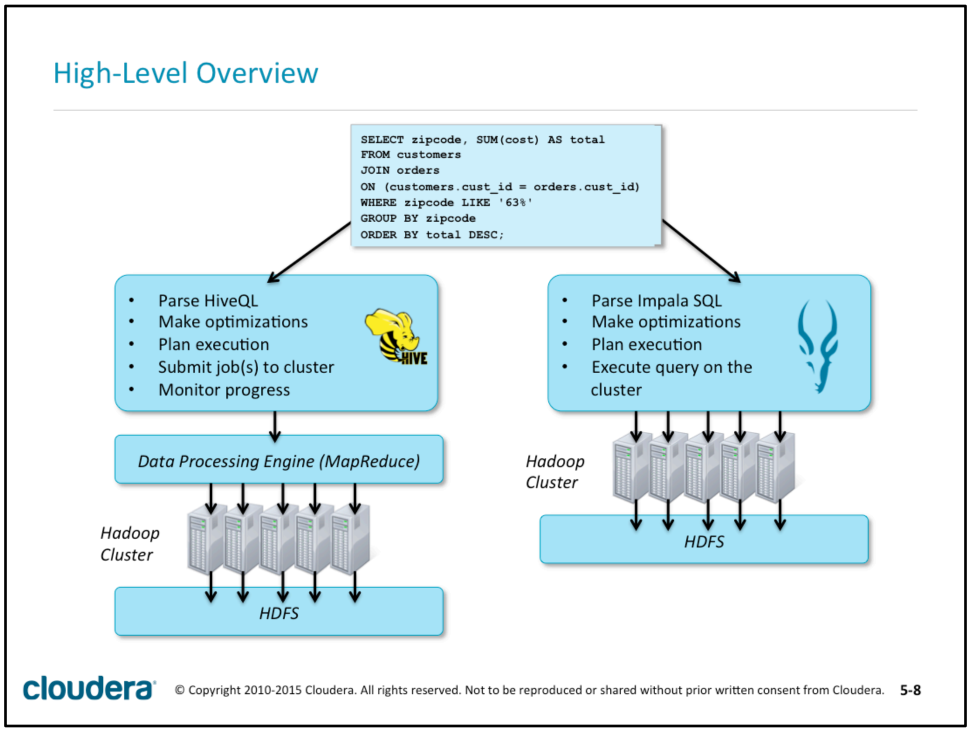
Cloudera had been working on Impala (as an internal project) for nearly two years before Apache Drill was announced. Although the two projects are inspired by Google's Dremel, they have somewhat different approaches. The main difference between Impala and Apache Drill that is that Impala is a fully-supported system used for daily production work by lots of companies (many of whom started using it in 2012 during the beta testing program), while Apache Drill is still in the design phase as an Incubator project at Apache. Although Cloudera's Impala is not managed by the Apache Software Foundation, both Impala and Drill are free/open source and available under the terms of the Apache Software License (which is the same license used by Hadoop, Hive, Pig, Sqoop, etc.)

As of this writing (December 2013), Milestone 1 of Drill is available. (Milestone 5 is listed as the production release.) The only way to run it is by checking out the source code from the revision control system and compiling the binaries yourself. If you want to follow the progress of the Drill project, refer to [`http://incubator.apache.org/drill`] and [`http://drill-user.org`].

Performance numbers: Depending on a number of factors (data format, table sizes, cluster configuration, etc.), Impala is usually at least five times faster than Hive and can be as much as 68 times faster for real-world workloads. See [`http://tiny.cloudera.com/dac15c`] for more specific information. Anecdotally, you might find that it's 100 times faster than Hive for relatively small queries in which the latency associated with MapReduce dominates the total running time.

"Highly-optimized for queries" – unlike Hive or Pig which satisfy queries by running predefined MapReduce jobs according to an execution plan, Impala has a special-purpose execution engine that was built specifically to do queries. On a more technical level, it uses code generation and can take advantage of instruction sets available in the latest CPUs, but this is beyond the scope of this course.
Cloudera does not distribute Drill with CDH nor does it support it.

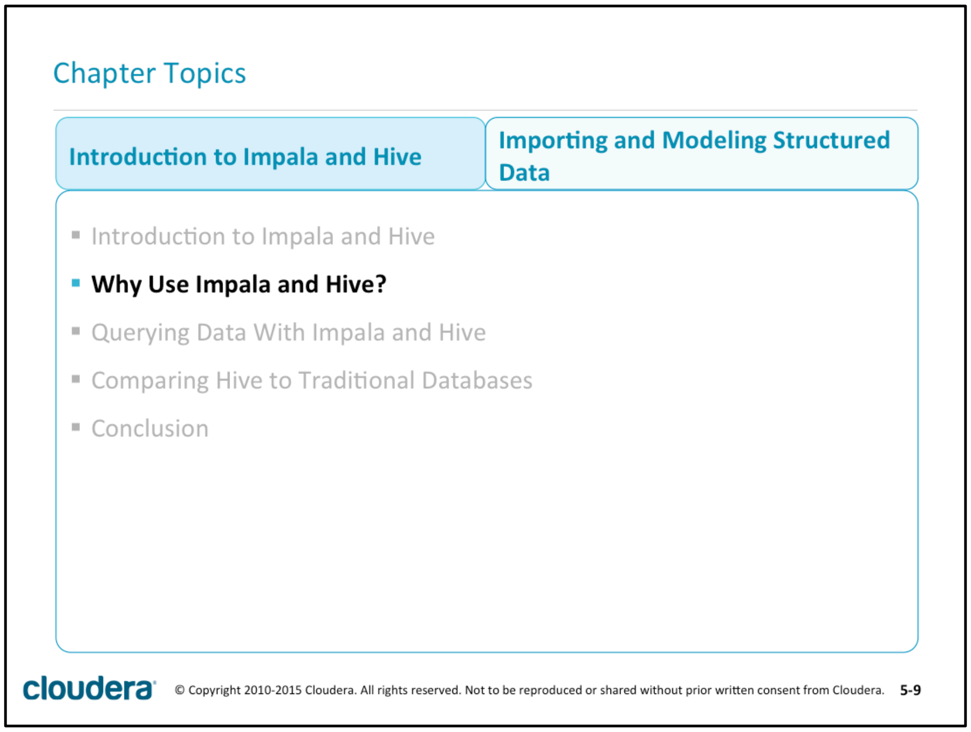
From Wikipedia: Apache Drill is an open-source software framework that supports data-intensive distributed applications for interactive analysis of large-scale datasets. Drill is the open source version of Google's Dremel system which is available as an infrastructure service called Google BigQuery.



High-Level Overview

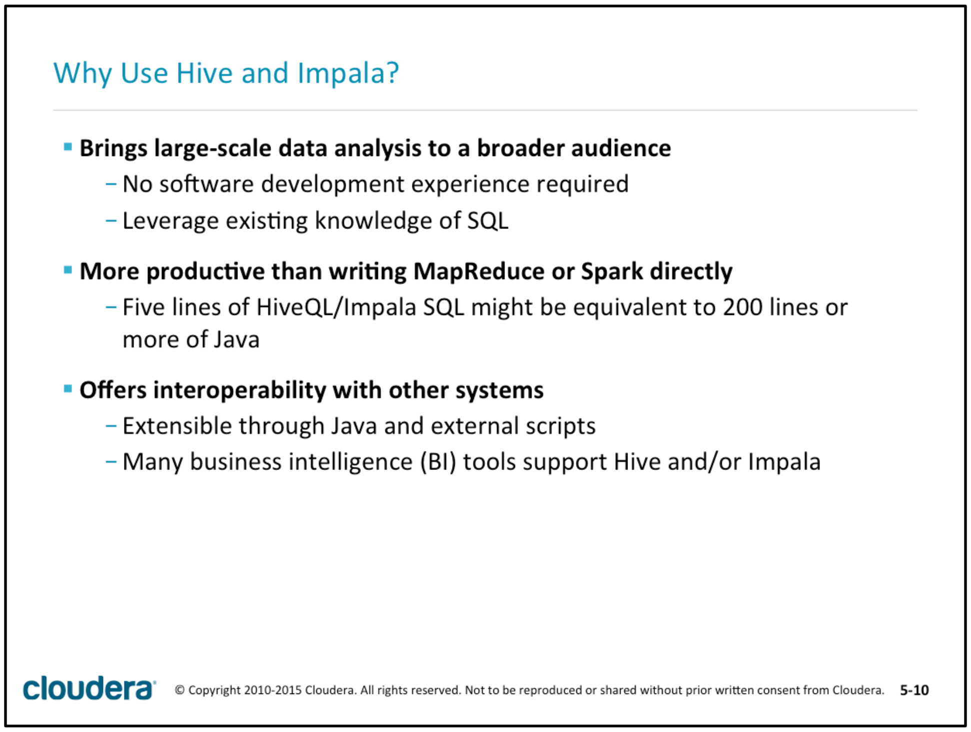
```
SELECT zipcode, SUM(cost) AS total
FROM customers
JOIN orders
ON (customers.cust_id = orders.cust_id)
WHERE zipcode LIKE '63%'
GROUP BY zipcode
ORDER BY total DESC;
```

Your HiveQL statements are interpreted by Hive. Hive then produces one or more MapReduce jobs, and then submits them for execution on the Hadoop cluster. A more detailed illustration of the Hive architecture (more appropriate for a technical audience) is shown in PH1e p. 7 or TDG p. 420). There isn't a lot of detailed documentation on Hive's architecture, nor is an in-depth discussion of such details relevant for our target audience. But if you'd like to know more as an instructor, you might read the slides from this 2011 presentation [`http://www.slideshare.net/recruitcojp/internal-hive#btnNext`] or this one from 2009 [`http://www.slideshare.net/nzhang/hive-anatomy`]. The Hive wiki has a developer guide [`https://cwiki.apache.org/Hive/developerguide.html`], but this page hasn't been updated since 2011 and may be outdated.



Chapter Topics

Introduction to Impala and Hive | **Importing and Modeling Structured Data**

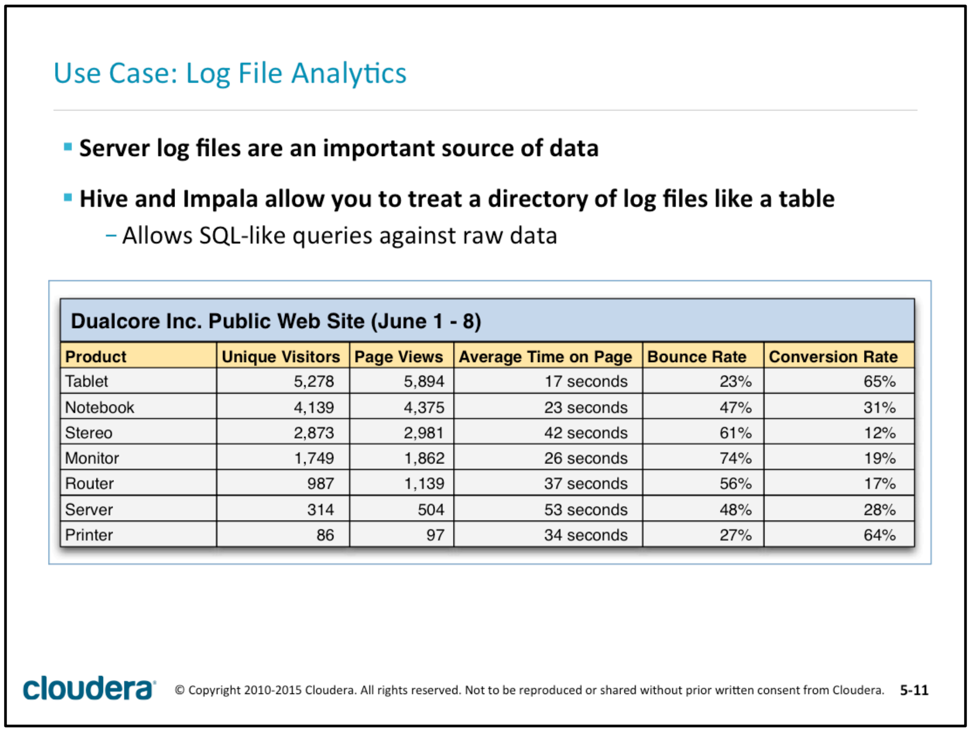


Why Use Hive and Impala?

- **Brings large-scale data analysis to a broader audience**
  - No software development experience required
  - Leverage existing knowledge of SQL
- **More productive than writing MapReduce or Spark directly**
  - Five lines of HiveQL/Impala SQL might be equivalent to 200 lines or more of Java
- **Offers interoperability with other systems**
  - Extensible through Java and external scripts
  - Many business intelligence (BI) tools support Hive and/or Impala

"Five lines of HiveQL might be equivalent to 100 lines of Java" –it might require writing 100 lines of code using the Java MapReduce API in order to accomplish the same result as a five line HiveQL query, particularly if it involves joining multiple datasets. Writing Java MapReduce not only requires far more code, it requires far more time to develop and debug. Even software developers who have extensive experience with Java and MapReduce benefit from using a higher-level abstraction like Hive (or Pig, Impala, etc.) for the typical case. Writing Java MapReduce code should generally be reserved for certain tasks, such as graph analysis (i.e. finding connections on a social network or transportation route planning), that cannot be easily done in Hive.

"Many business intelligence (BI) tools support it" – Datameer, MicroStrategy, Pentaho, and Qlikview are just a few examples of companies that produce BI tools capable of reading data from Hive (technical details vary for each product, but this is often done via JDBC or ODBC connectivity such that Hive looks like any other database to these programs). We'll see later in the course that Impala is often better suited to BI applications because of its speed. One disadvantage of Hive is that the code is ultimately executed as a MapReduce job. While this is extremely scalable, the MapReduce infrastructure comes with a certain amount of overhead. As with Pig or writing MapReduce directly, even a trivial job operating on a tiny amount of data might take 10-15 seconds to complete.



Use Case: Log File Analytics

- Server log files are an important source of data
- Hive and Impala allow you to treat a directory of log files like a table
  - Allows SQL-like queries against raw data

Dualcore Inc. Public Web Site (June 1 - 8)

Product	Unique Visitors	Page Views	Average Time on Page	Bounce Rate	Conversion Rate
Tablet	5,278	5,894	17 seconds	23%	65%
Notebook	4,139	4,375	23 seconds	47%	31%
Stereo	2,873	2,981	42 seconds	61%	12%
Monitor	1,749	1,862	26 seconds	74%	19%
Router	987	1,139	37 seconds	56%	17%
Server	314	504	53 seconds	48%	28%
Printer	86	97	34 seconds	27%	64%

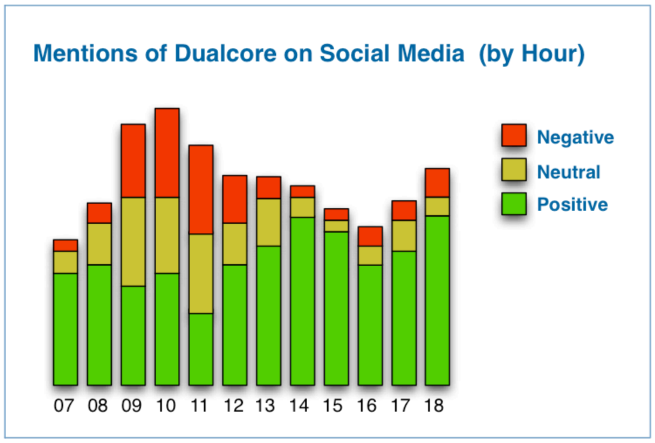
cloudera 5-11

Hive makes it easy to query the raw logs. You can parse Web server log files using the regular expression support built into Hive (RegexSerDe), which means that you can effectively treat a directory full of raw log files as a table that you can execute queries against using HiveQL (e.g. `SELECT COUNT(*) FROM logs WHERE date = '10/May/2013' AND url = '/product/foo' GROUP BY ip_address`). The diagram here is simply an example to illustrate what type of insight you could extract from Web server log files by issuing the right queries in Hive (i.e. Hive does not have support for creating reports like this).

Jonathan Seidman goes into some detail about how Orbitz uses Hive to process Web logs starting at slide 27 of this presentation [`http://tiny.cloudera.com/dac09b`]. However, you need not limit analysis to Web logs – you could just as easily analyze firewall, proxy server, or application logs. Mozilla wrote an article a few years ago [`http://blog.mozilla.org/data/2010/08/15/collecting-and-analyzing-log-data-via-flume-and-hive/`] about some of the ways they use Hive for log analysis.

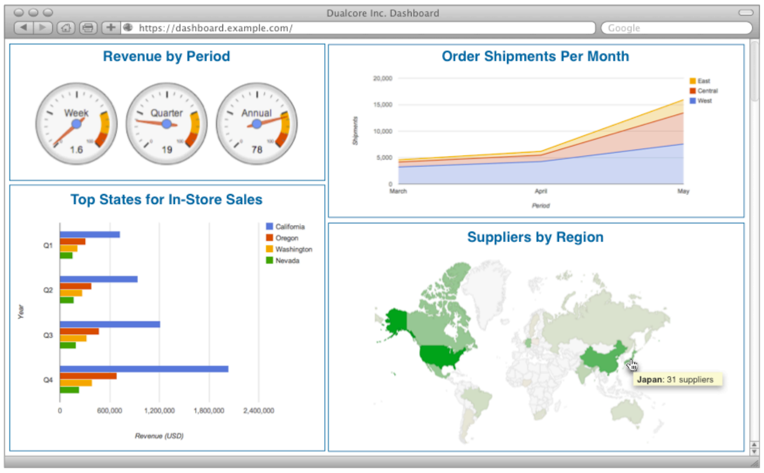
Use Case: Sentiment Analysis

- **Many organizations use Hive or Impala to analyze social media coverage**



Use Case: Business Intelligence

- **Many leading business intelligence tools support Hive and Impala**



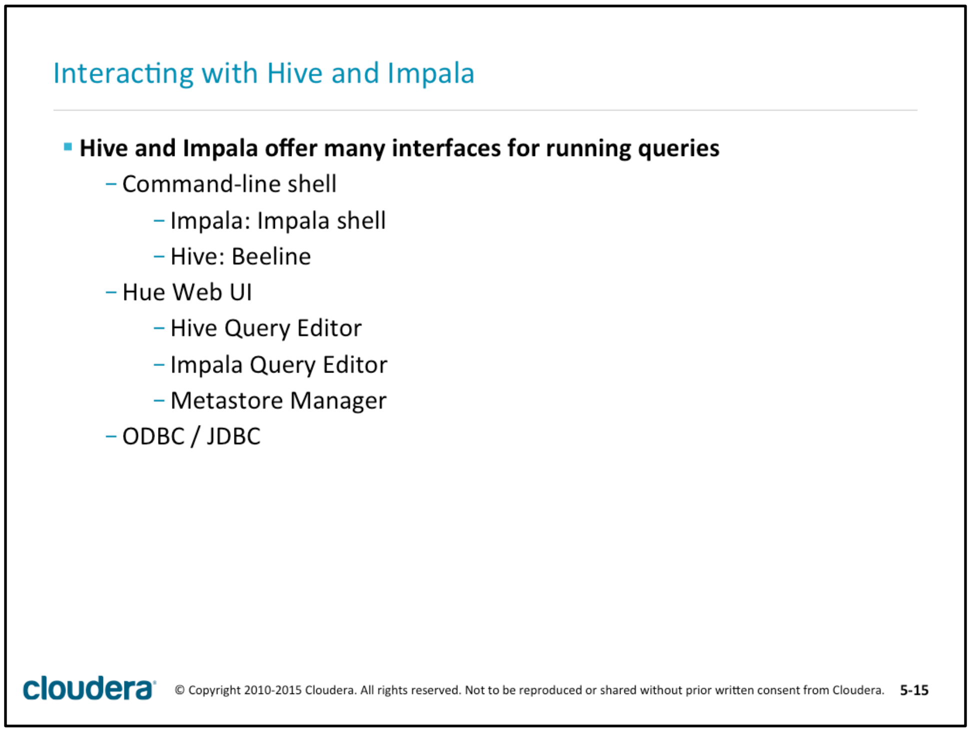
Although many BI tools support Hive, many also support Impala (which is much faster and better suited to interactive/ad hoc analysis, as we'll cover later in the course). As the videos mentioned in the previous slide's notes imply, the ability to query and visualize data almost instantly using a BI tool with Impala is a far superior experience than having to wait several minutes for Hive to complete a query.

This screenshot is a mockup created in a graphics editor but depicts a fairly standard dashboard and most analysts should be familiar with the basic concept. While this is a graphical representation of business intelligence (mainly for visual effect), Impala can also be used for reporting, just as you might use Hive or other enterprise data warehouse systems.

Chapter Topics

Introduction to Impala and Hive | **Importing and Modeling Structured Data**

- Introduction to Impala and Hive
- Why Use Impala and Hive?
- **Querying Data With Hive and Impala**
- Comparing Hive to Traditional Databases
- Conclusion



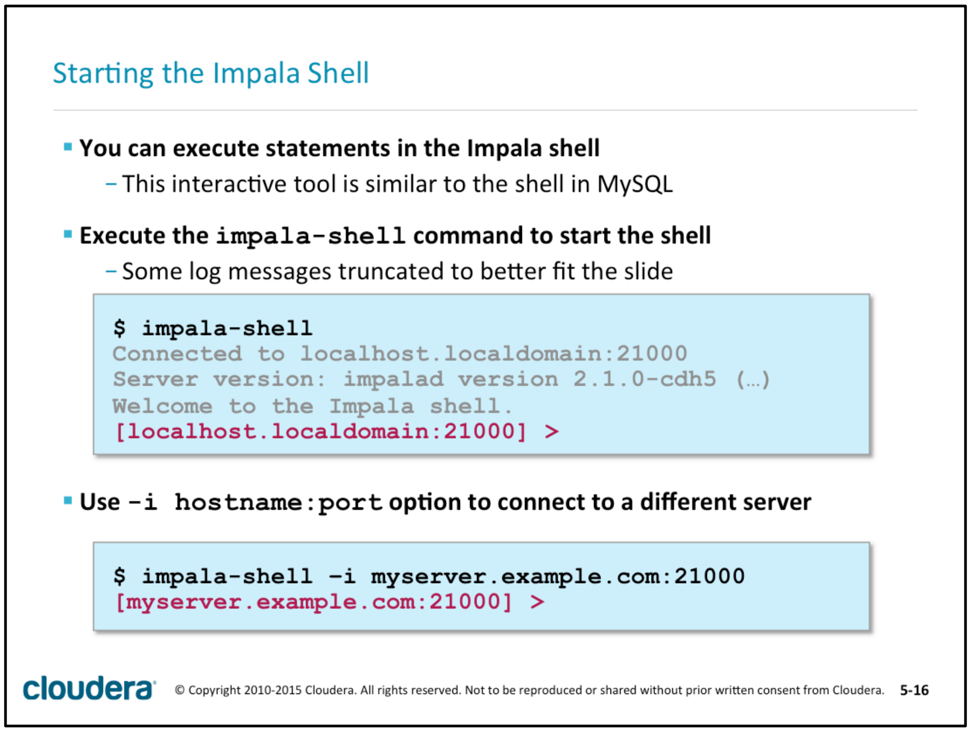
Interacting with Hive and Impala

- **Hive and Impala offer many interfaces for running queries**
  - Command-line shell
    - Impala: Impala shell
    - Hive: Beeline
  - Hue Web UI
    - Hive Query Editor
    - Impala Query Editor
    - Metastore Manager
  - ODBC / JDBC

cloudera

We'll cover the Impala shell later in this chapter and will also show two screenshots illustrating Hue's Impala integration.

The mention of ODBC / JDBC offers a segue into an upcoming slide, as BI applications typically use this for connectivity to Impala. Most leading BI tool vendors are Cloudera partners and many of them have already had their products certified to work with Impala, including MicroStrategy [`http://www.cloudera.com/content/cloudera/en/solutions/partner/Microstrategy.html`] and Tableau [`http://www.tableausoftware.com/about/blog/2013/4/tableau-cloudera-impala-bring-hadoop-data-life`]. The preceding links show short demonstration videos of these tools. Note that the connection dialog shown in the Tableau demo says "Impala (beta)" simply because that video was created prior to the 1.0 release of Impala.



Starting the Impala Shell

- **You can execute statements in the Impala shell**
  - This interactive tool is similar to the shell in MySQL
- **Execute the `impala-shell` command to start the shell**
  - Some log messages truncated to better fit the slide

```
$ impala-shell
Connected to localhost.localdomain:21000
Server version: impalad version 2.1.0-cdh5 (…)
Welcome to the Impala shell.
[localhost.localdomain:21000] >
```

- **Use `-i hostname:port` option to connect to a different server**

```
$ impala-shell -i myserver.example.com:21000
[myserver.example.com:21000] >
```

Color cue: Blue boxes = SQL (either HiveQL or Impala QL – differences are minor, it will be stated explicitly if the code refers to only one or the other). Grey = data or Linux shell. red text is what impala/hive returns from a query

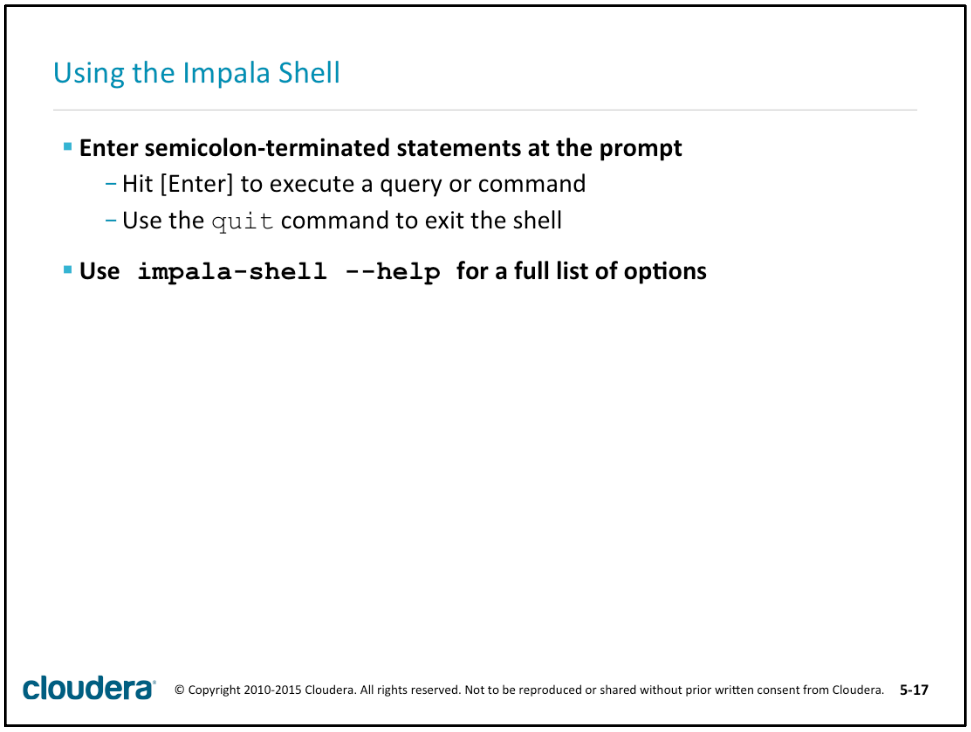
Starting the shell actually shows slightly more output than this, but it was truncated to fit on the slide.

The prompt identifies the current server. We'll abbreviate this as just > in subsequent slides.

Use `impala-shell -i hostname:port` to access another server. The port number is the TCP at which the server listens, typically 21000. If you don't specify this, the shell will connect to the default server (localhost). You can also type `CONNECT hostname` from within the Impala shell to connect to a different host.

Tip: The Impala shell has some support for autocompletion, similar to what's available in the bash shell. As with bash, you can type the first few letters of command and then hit the TAB key to either complete the command (if you've typed enough of it that no other commands match) or to see which keywords match what you've typed. Also similar to bash is the ability to go to the front of the current line by hitting Ctrl+A or to the end of the current line by hitting Ctrl+E. This is handy when you realize you made a typo at the beginning of the query, since you can hit Ctrl+A, fix the typo, and then hit Ctrl+E to get back to the end (a much better alternative to hitting the left and right arrow keys numerous times to move around). Finally, you can hit the up and down arrow keys to navigate through the list of previously-typed statements.

Although these features are convenient, they are also fairly buggy when compared to the same features in bash. Even simple combinations like trying to use the up arrow to recall a command and then using the Ctrl+A or left arrow to edit it often confuses the line editor built into Hive's shell. Impala's shell is much better in this regard.



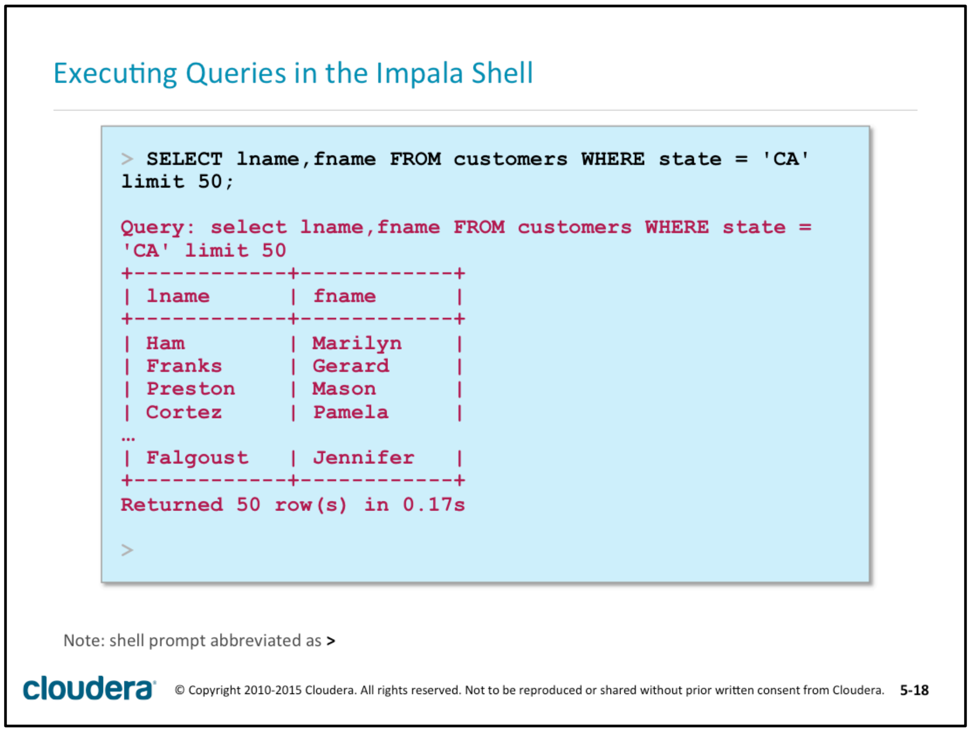
Using the Impala Shell

- **Enter semicolon-terminated statements at the prompt**
  - Hit [Enter] to execute a query or command
  - Use the `quit` command to exit the shell
- **Use `impala-shell --help` for a full list of options**

Impala's shell is similar to Hive's shell (or Grunt, Pig's shell). However, one difference you'll find obvious after using Impala's shell for a few minutes is that line editing works very well (unlike Hive's shell, which gets confused when you try to edit a previous command that spans multiple lines).

Beta versions of Impala didn't require you to terminate commands with a semicolon, but Impala 1.0 and later versions require this just like Hive shell (or Grunt).

You can also use `exit` to terminate the shell (as in Hive's shell), but `quit` is the preferred command (and the one shown in the documentation).



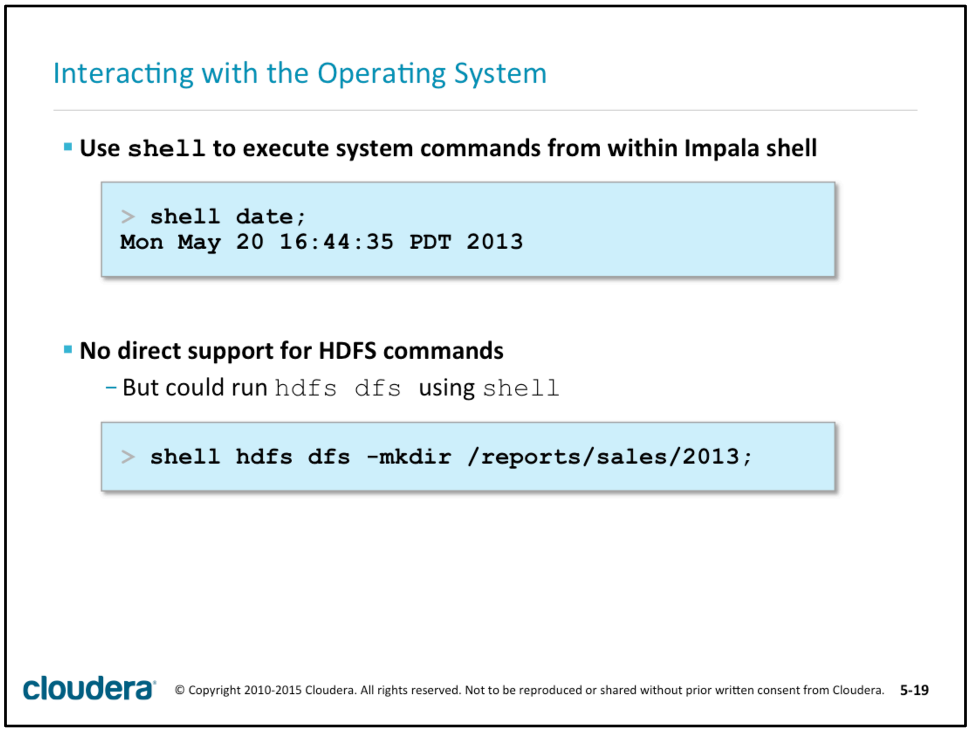
Executing Queries in the Impala Shell

```
> SELECT lname,fname FROM customers WHERE state = 'CA'
limit 50;

Query: select lname,fname FROM customers WHERE state =
'CA' limit 50
+------------+------------+
| lname      | fname      |
+------------+------------+
| Ham        | Marilyn    |
| Franks     | Gerard     |
| Preston    | Mason      |
| Cortez     | Pamela     |
…
| Falgoust   | Jennifer   |
+------------+------------+
Returned 50 row(s) in 0.17s

>
```

Note: shell prompt abbreviated as **>**

Color cue: Blue boxes = SQL. Red text is what impala/hive returns from a query

Tip: Impala shell has some support for autocompletion, similar to what's available in the bash shell. As with bash, you can type the first few letters of command and then hit the TAB key to either complete the command (if you've typed enough of it that no other commands match) or to see which keywords match what you've typed. Also similar to bash is the ability to go to the front of the current line by hitting Ctrl+A or to the end of the current line by hitting Ctrl+E. This is handy when you realize you made a typo at the beginning of the query, since you can hit Ctrl+A, fix the typo, and then hit Ctrl+E to get back to the end (a much better alternative to hitting the left and right arrow keys numerous times to move around). Finally, you can hit the up and down arrow keys to navigate through the list of previously-typed statements.



Interacting with the Operating System

- **Use `shell` to execute system commands from within Impala shell**

```
> shell date;
Mon May 20 16:44:35 PDT 2013
```

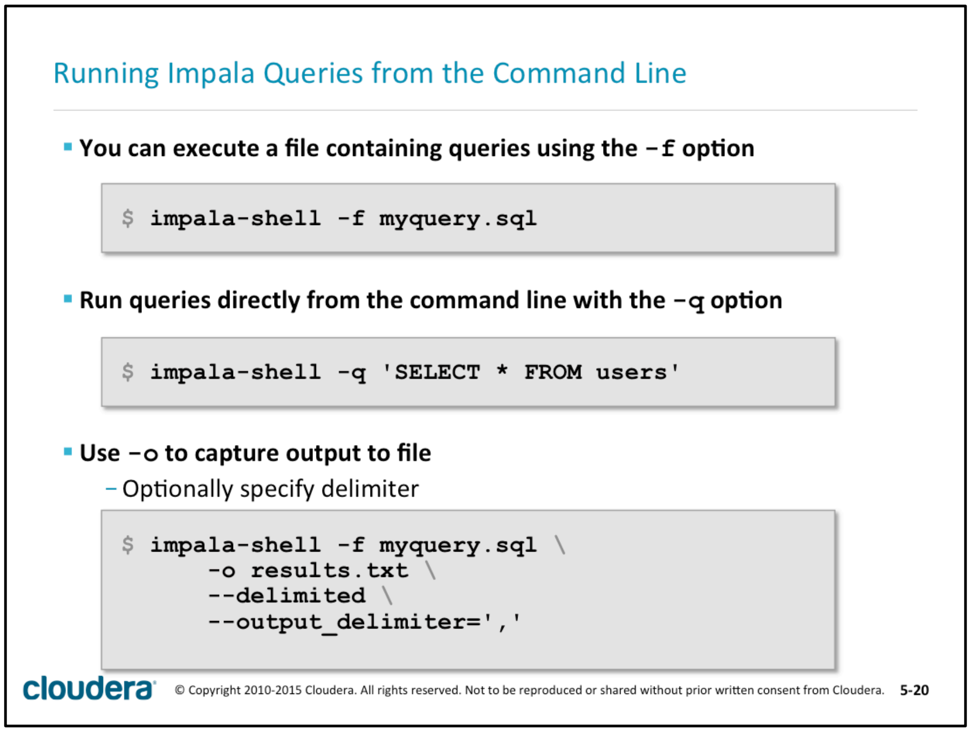
- **No direct support for HDFS commands**
  - But could run `hdfs dfs` using `shell`

```
> shell hdfs dfs -mkdir /reports/sales/2013;
```

Color cue: Blue boxes = SQL.

There is no support for running HDFS commands from within the Impala shell except to run them as shell commands (e.g. `shell hadoop fs -ls /dualcore`).

Wildcards and pipes both appear to work in commands run using `shell` (though this seems to be an undocumented feature).



Running Impala Queries from the Command Line

- **You can execute a file containing queries using the `-f` option**

```
$ impala-shell -f myquery.sql
```

- **Run queries directly from the command line with the `-q` option**

```
$ impala-shell -q 'SELECT * FROM users'
```

- **Use `-o` to capture output to file**
  - Optionally specify delimiter

```
$ impala-shell -f myquery.sql \
      -o results.txt \
      --delimited \
      --output_delimiter=','
```

Color cue: Grey = data or Linux shell.

Using the −o option is definitely recommended if you want to capture output to a file, because simply using shell redirection would include the pretty printing shown in the previous slide. Using the −o option will produce comma-delimited output by default, but you can specify an alternate delimiter as shown here (a tab, in this example). The output is still pretty-printed in the console even when using the the −o option, but the results contained in the file are delimited.

$ impala-shell –f myquery.sql –o myresults_just_o.txt

The file myresults_just_o.txt will contain pretty printing.

$ impala-shell -f myquery.sql -o myresults_with_delimiter.txt --output_delimiter='\t'

The file myresults_with_delimiter.txt also includes pretty printing!

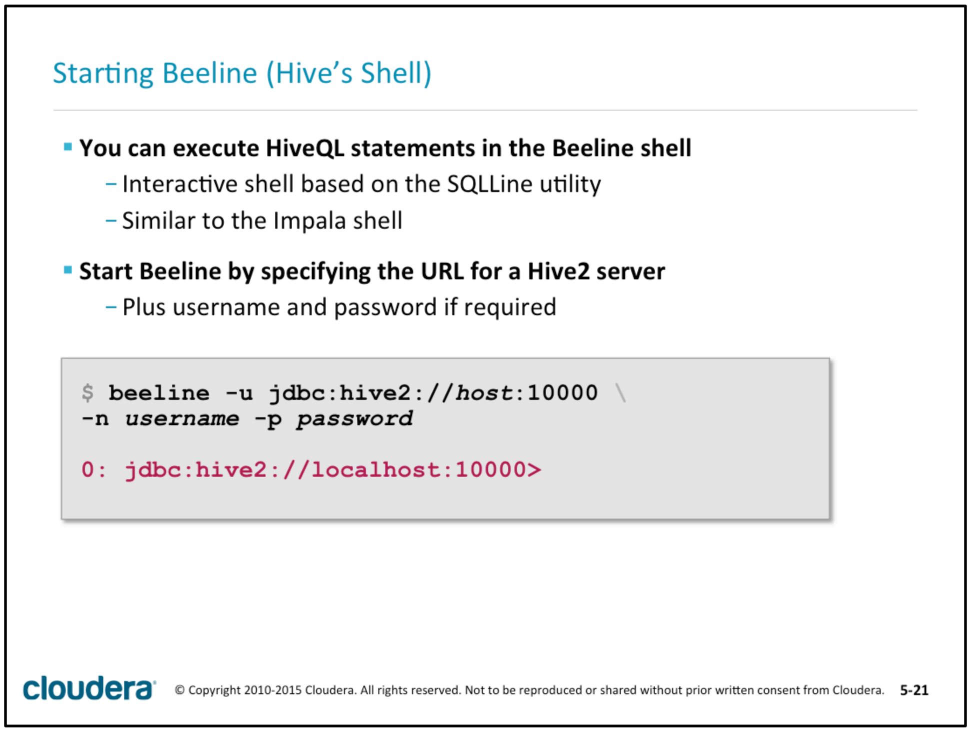
$ impala-shell -f myquery.sql -o myresults_with_delimited.txt --delimited

The file myresults_with_delimited.txt does not include pretty printing and uses a tab as the delimiter (not a comma!!!)

$ impala-shell -f myquery.sql -o myresults_with_delimiter_comma.txt --delimited --output_delimiter=','

The file myresults_with_delimiter_comma.txt does not include pretty printing and uses a comma as the delimiter.

If you omit the -o option, the output can still be modified with --output_delimiter and --delimited. The results on the screen will not include pretty printing and, as expected, there will be no output file created.



Starting Beeline (Hive's Shell)

- **You can execute HiveQL statements in the Beeline shell**
  - Interactive shell based on the SQLLine utility
  - Similar to the Impala shell
- **Start Beeline by specifying the URL for a Hive2 server**
  - Plus username and password if required

```
$ beeline -u jdbc:hive2://host:10000 \
-n username -p password

0: jdbc:hive2://localhost:10000>
```

cloudera

Beeline is based on http://sqlline.sourceforge.net

http://blog.cloudera.com/blog/2014/02/migrating-from-hive-cli-to-beeline-a-primer/

The prompt shows the number of the connection (because you can have multiple connections – in this example there's just one, shown as "0") and the URL of the server. In the future, the prompt will be shown simply as > to save space.

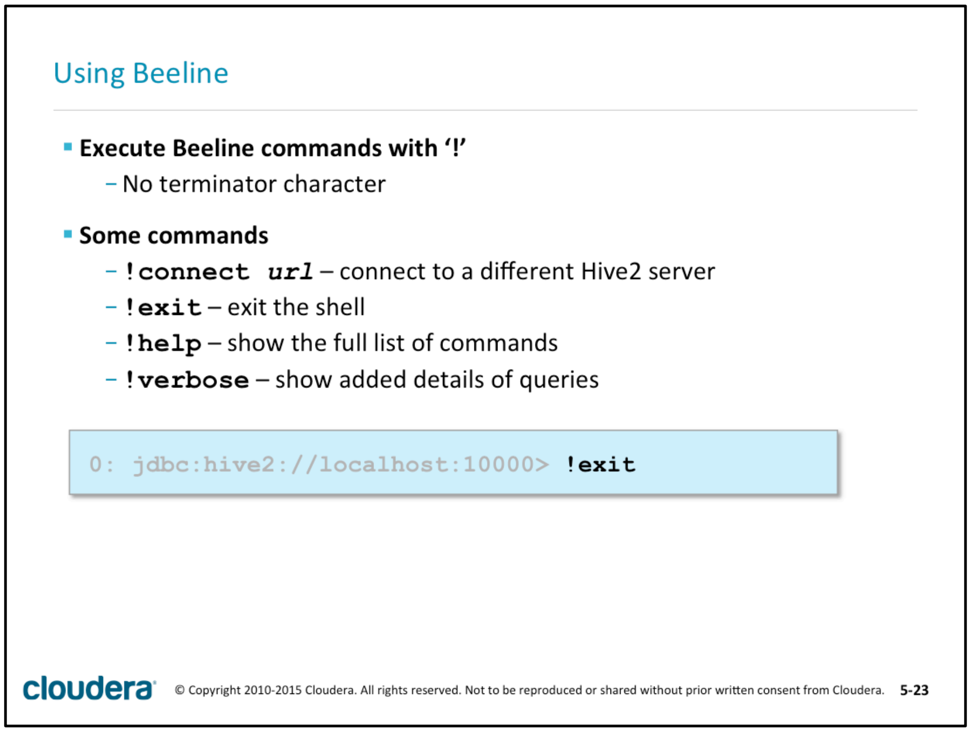
The need to terminate statements with a semicolon is familiar to those who've used a database shell (or Grunt), and as in either you must hit the Enter key after typing a statement in order to execute it (i.e. simply adding a semicolon at the end isn't enough since statements may span lines). This example shows a session in which we start the Hive shell from the UNIX command line, run a query, see the results displayed as tab-separated columns on the terminal's standard output, then quit Hive and return to the UNIX shell. In reality, this would also usually display several log messages (depending on configuration), but I have omitted them here for brevity. The `-S` option, discussed on the next slide, can be used to suppress these messages. You can also use the "`SOURCE path`" command from within Hive shell to execute HiveQL statements in the file referenced by 'path'.

Hive's configuration is stored in XML files, typically in `/etc/hive/conf`, but you can specify an alternate configuration directory via the `HIVE_CONF_DIR` environment variable. Those settings apply to everyone who uses Hive on that machine (i.e. not much of a concern if you run Hive on your laptop, since you're likely the only user, but be careful if changing settings on server used by others). This is more an area for system administrators, so we don't cover configuration in depth in this class. See TDG3e pp. 417-419 or PH1e pp. 24-34 for details on configuration.

Executing Queries in Beeline

- **SQL commands are terminated with semi-colon (;)**
- **Similar to Impala shell**
  - Results formatting is slightly different

```
1: url> SELECT lname,fname FROM customers
. . . > WHERE state = 'CA' LIMIT 50;

+-------------+-------------+
|    lname    |    fname    |
+-------------+-------------+
| Ham         | Marilyn     |
| Franks      | Gerard      |
| Preston     | Mason       |
...
| Falgoust    | Jennifer    |
+-------------+-------------+
50 rows selected (15.829 seconds)

1: url>
```



Using Beeline

- **Execute Beeline commands with '!'**
  - No terminator character
- **Some commands**
  - **`!connect`** ***`url`*** – connect to a different Hive2 server
  - **`!exit`** – exit the shell
  - **`!help`** – show the full list of commands
  - **`!verbose`** – show added details of queries

```
0: jdbc:hive2://localhost:10000> !exit
```

The !run command can be used from the beeline shell to run a (local) script.

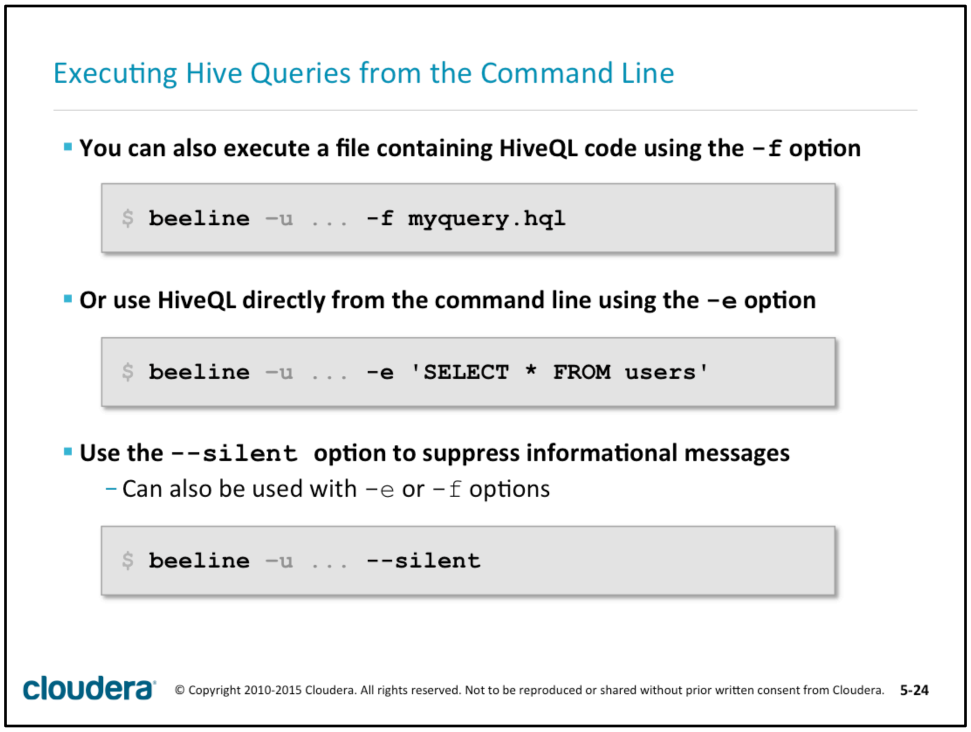
```
0: jdbc:hive2://localhost:10000> !run myquery.hql
```

SHOW TABLES; and !tables produce markedly different results.

```
0: jdbc:hive2://localhost:10000> SHOW TABLES;
+----------------+--+
|   tab_name     |
+----------------+--+
| customers      |
| order_details  |
| orders         |
| products       |
+----------------+--+
4 rows selected (0.026 seconds)
0: jdbc:hive2://localhost:10000> !tables
+------------+--------------+----------------+-------------+----------+--+
| TABLE_CAT  | TABLE_SCHEM  |  TABLE_NAME    | TABLE_TYPE  | REMARKS  |
+------------+--------------+----------------+-------------+----------+--+
|            | accounting   | accounts       | TABLE       | NULL     |
|            | default      | customers      | TABLE       | NULL     |
|            | default      | order_details  | TABLE       | NULL     |
|            | default      | orders         | TABLE       | NULL     |
|            | default      | products       | TABLE       | NULL     |
+------------+--------------+----------------+-------------+----------+--+
```

The !rehash command can assist with auto-completion. This command will retrieve the table column and names to the shell.

```
0: jdbc:hive2://localhost:10000> !rehash
Building list of tables and columns for tab-completion (set fastconnect to true to skip)...
22/22 (100%) Done
Done
```


Executing Hive Queries from the Command Line

- You can also execute a file containing HiveQL code using the -f option

```
$ beeline -u ... -f myquery.hql
```

- Or use HiveQL directly from the command line using the -e option

```
$ beeline -u ... -e 'SELECT * FROM users'
```

- Use the --silent option to suppress informational messages
  - Can also be used with -e or -f options

```
$ beeline -u ... --silent
```

The file containing HiveQL is simply a text file and it's often referred to as a "script". It's customary to use the .hql (HiveQL) file extension, but .q (query) is also common. We will use the former in this class. If you're only executing a single statement with the -e option, it's not necessary to terminate it with a semicolon.

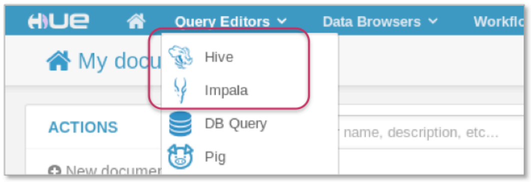
Hive frequently displays informational messages in normal usage; for example, when starting Hive, you might see information about the configuration files it has loaded or where it will store logs for the current session (depending on how Log4J is configured), and you'll see MapReduce status messages while your query runs.

Using --silent=true will enable silent mode, which suppresses all non-essential output.

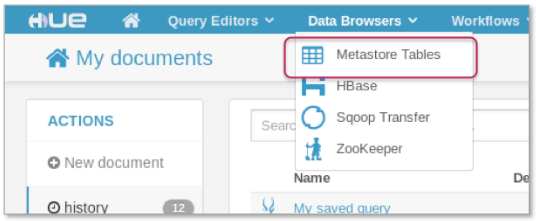
Using Hue with Hive and Impala

You can use Hue to…

Query data with Hive or Impala



View and manage the Metastore

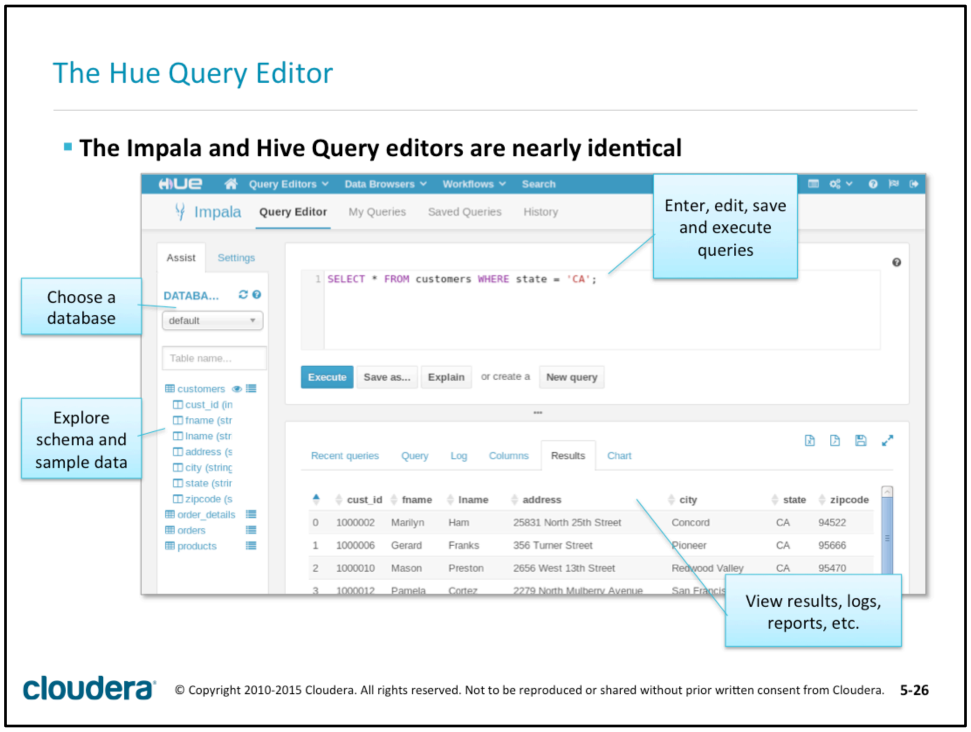


Use the Query Editors menu to access either the Hue or Impala query editor.

Note that the Hive Query Editor used to be called “Beeswax”. That’s still technically the name of the tool but it isn’t referenced as such in the UI, so stick to the term Hive Query Editor.

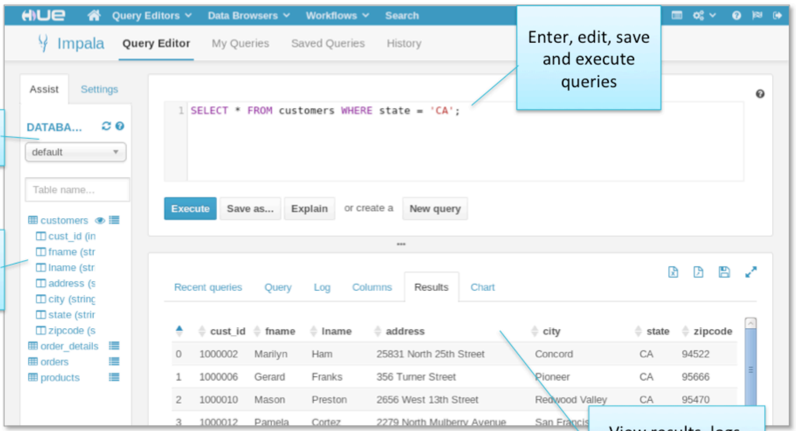
The Uis of the two editors are nearly identical so we just cover the Impala UI.

Data Browsers > Metastore Tables opens the Metastore Manager, which is covered in the next chapter.



The Hue Query Editor

- **The Impala and Hive Query editors are nearly identical**



```
SELECT * FROM customers WHERE state = 'CA';
```

	cust_id	fname	lname	address	city	state	zipcode
0	1000002	Marilyn	Ham	25831 North 25th Street	Concord	CA	94522
1	1000006	Gerard	Franks	356 Turner Street	Pioneer	CA	95666
2	1000010	Mason	Preston	2656 West 13th Street	Redwood Valley	CA	95470
3	1000012	Pamela	Cortez	2279 North Mulberry Avenue	San Francis		

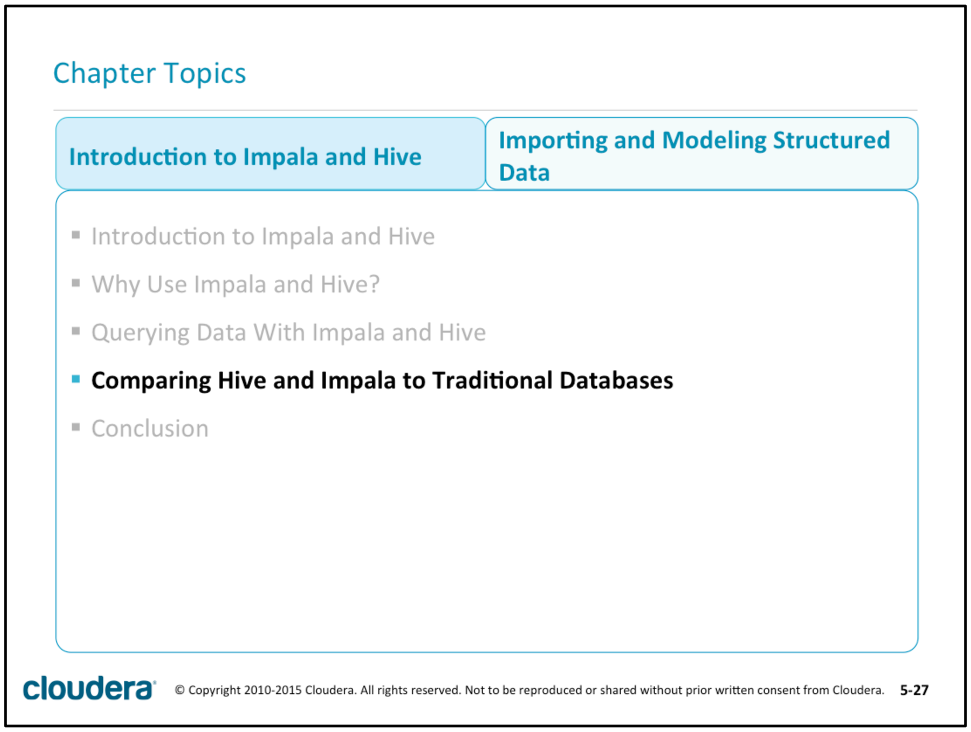
When you on the Impala query editor, you'll see the query editor (shown here). As with Hive's `-e` command-line option, it's not strictly required to terminate your query with a semicolon in Hue's query editor before executing it (assuming you have only a single statement to execute), but it certainly doesn't hurt to do so.

When you click the blue 'Execute' the editor will execute the query and display the results in the Results tab as shown.

You can also save the query itself with the 'Save As…' button at the bottom, allowing you to run it again in a later session. This prompts you to specify a name and description for the query, and is a very handy feature for when you need to do repetitive queries, such as for monthly or quarterly reports.

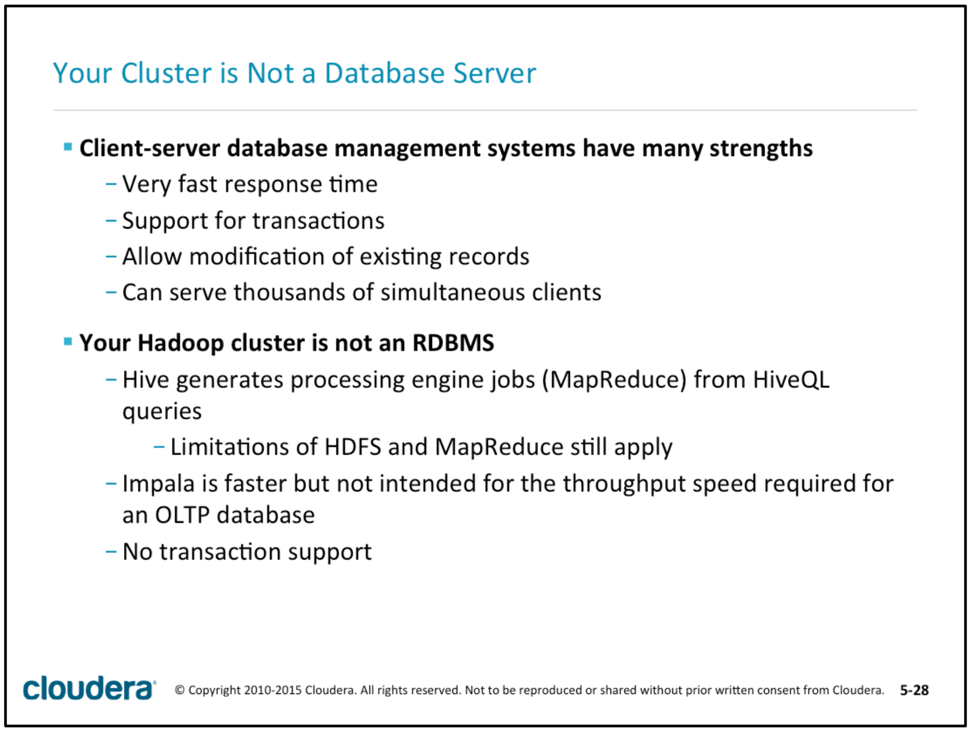
While the query runs, the logs tab will show you a constantly updated view of the logs. This continues until the query is complete, at which the results are displayed.

This screenshot shows the results of the query They are displayed one page at a time (each page contains a maximum of 100 records) and you can navigate through each page using the links in the lower right. You can click a column heading to sort the results by that heading. You can also click the links on the left side to save these results in CSV or XLS (Microsoft Excel) format, or to save the query results to a new table or HDFS directory. Note that the 'Save' link here is for saving the query results, which is NOT the same as saving the query itself (as discussed in the notes for the first screenshot).



Chapter Topics

Introduction to Impala and Hive | **Importing and Modeling Structured Data**



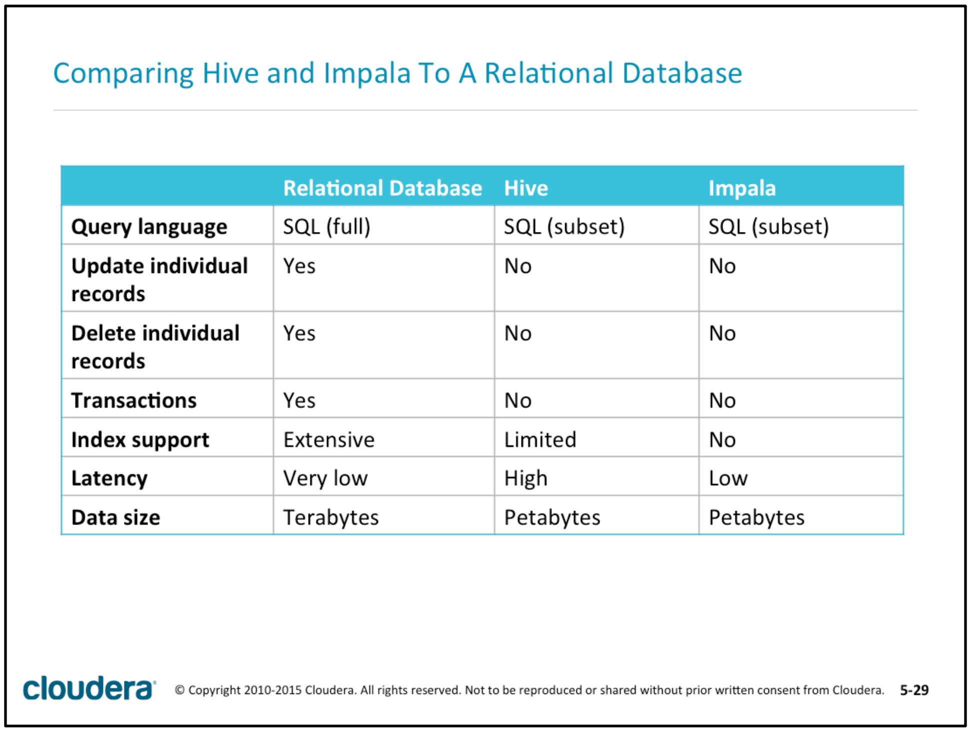
Your Cluster is Not a Database Server

- **Client-server database management systems have many strengths**
  - Very fast response time
  - Support for transactions
  - Allow modification of existing records
  - Can serve thousands of simultaneous clients
- **Your Hadoop cluster is not an RDBMS**
  - Hive generates processing engine jobs (MapReduce) from HiveQL queries
    - Limitations of HDFS and MapReduce still apply
  - Impala is faster but not intended for the throughput speed required for an OLTP database
  - No transaction support

cloudera © Copyright 2010-2015 Cloudera. All rights reserved. Not to be reproduced or shared without prior written consent from Cloudera. 5-28

Hive is NOT an acceptable substitute for an online transaction processing (OLTP) [`http://en.wikipedia.org/wiki/Online_transaction_processing`] database, although it may be a good substitute for (or complement to) an online analytical processing (OLAP) [`http://en.wikipedia.org/wiki/Online_analytical_processing`] database, as is typically used for analytical queries and reporting on large amounts of data. In fact, this is precisely why Facebook originally created Hive (see the original paper *Hive: A Warehousing Solution Over a MapReduce Framework* [`http://www.vldb.org/pvldb/2/vldb09-938.pdf`]). Perhaps the most striking difference between Hive and an RDBMS is that Hive does not enforce referential integrity. Additionally, you cannot reliably query data in Hive while loading data.

"Limitations of HDFS and MapReduce still apply" – the overhead for submitting and executing a MapReduce job (as Hive does) means that even a simple query and a small amount of data will probably take at least ten seconds to return results. Furthermore, the "write once, read many" design constraint of HDFS means that Hive does not allow updates or deletes of individual records. One exception to the MapReduce overhead is when you want to read data from a table without any processing at all (e.g. `SELECT * from mytable`); in such a case, Hive may return data directly from HDFS without running a MapReduce job at all (and thus, without the overhead of scheduling and executing a MapReduce job, the latency will be much lower). These and other limitations are summarized on the next slide.



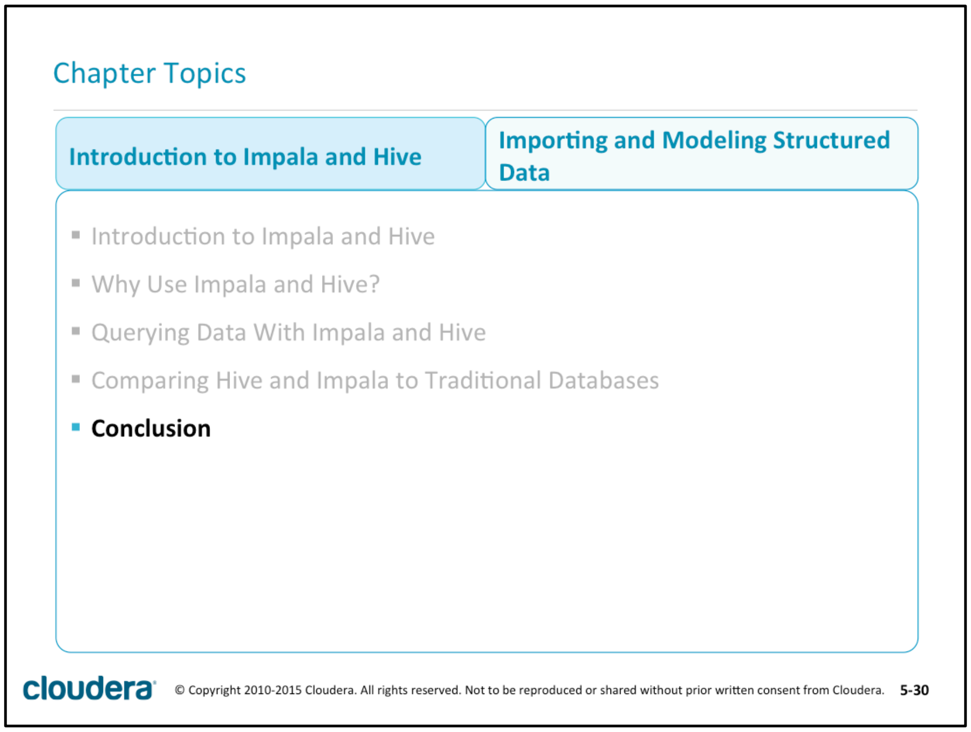
Comparing Hive and Impala To A Relational Database

	Relational Database	Hive	Impala
Query language	SQL (full)	SQL (subset)	SQL (subset)
Update individual records	Yes	No	No
Delete individual records	Yes	No	No
Transactions	Yes	No	No
Index support	Extensive	Limited	No
Latency	Very low	High	Low
Data size	Terabytes	Petabytes	Petabytes

cloudera

5-29

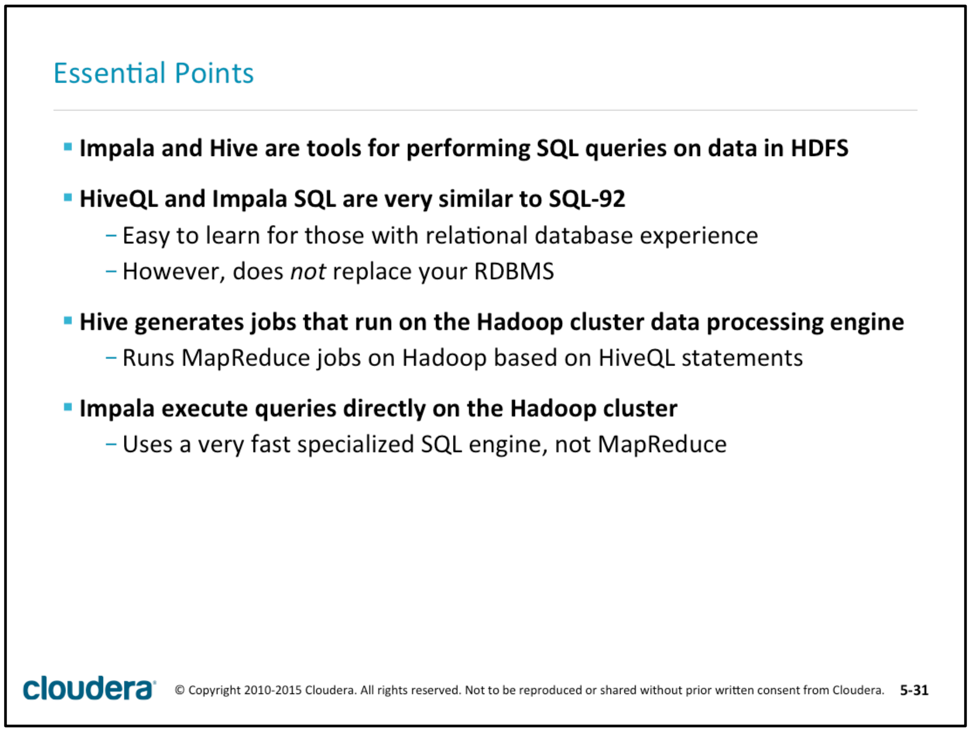
HiveQL is a subset of the SQL-92 standard (SQL=Structured Query Language), plus Hive-specific extensions. The syntax of HiveQL is similar but not identical to the SQL you might have used with Oracle, MySQL, Microsoft SQL Server, IBM DB2, and so on. Because Hive operates on data stored in HDFS, the limitations of HDFS affect Hive. Since HDFS does not support modifications to existing files (as explained during the Hadoop Fundamentals chapter), Hive does not support the `UPDATE` or `DELETE` statements from SQL. As of release 0.8.0 in December 2011, Hive now supports `INSERT INTO` [`https://issues.apache.org/jira/browse/HIVE-306`], so it's possible to add new data to an existing table (previously Hive supported only `INSERT OVERWRITE`). In practice, this may be less of a concern than it first seems because Hive is typically used for large-scale analytical queries and reporting (i.e. data warehousing), where the need to update or delete specific records (or the real need for transactions) is relatively uncommon. Support for indexes was finally added to Hive in release 0.7.0 [see `HIVE-417`], but as PH1e p. 117 points out, this support is primitive. Latency refers to the minimum duration between when issuing a query and receiving its results. RDBMS are highly optimized for small amounts of data and can return results for a simple query in a fraction of a second, while Hive uses MapReduce for analysis and has the latency that entails (i.e. even a trivial query on a small amount of data will likely take 10 seconds or more; more typical queries generally take minutes or possibly even hours). However, Hive and Hadoop can handle far more data (at far less cost) than an RDBMS so trading features and performance for scalability (and savings) can be quite worthwhile.



Chapter Topics

Introduction to Impala and Hive | **Importing and Modeling Structured Data**

- Introduction to Impala and Hive
- Why Use Impala and Hive?
- Querying Data With Impala and Hive
- Comparing Hive and Impala to Traditional Databases
- **Conclusion**



Essential Points

- **Impala and Hive are tools for performing SQL queries on data in HDFS**
- **HiveQL and Impala SQL are very similar to SQL-92**
  - Easy to learn for those with relational database experience
  - However, does *not* replace your RDBMS
- **Hive generates jobs that run on the Hadoop cluster data processing engine**
  - Runs MapReduce jobs on Hadoop based on HiveQL statements
- **Impala execute queries directly on the Hadoop cluster**
  - Uses a very fast specialized SQL engine, not MapReduce

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 are some reasons you wouldn't replace a typical relational database with Hive or Impala? (Answers might include: latency, lack of support for transactions, lack of support for update/delete of specific records)
- **Q2**: What are two benefits that Hive/Impala and Hadoop have over typical data warehousing systems? (Answer: low cost and high scalability)
- **Q3**: What are two interfaces you can use for running queries in Hive or Impala? (Answers: Impala shell, Beeline shell, Hue, or an external application that uses ODBC/JDBC such as a BI tool)
- **Q4**: Would Hive or Impala be a better choice for a nightly ETL process on large amounts of data? (Answer: Hive, since it is better suited for large, long-running batch processes)
- **Q5**: Would Hive or Impala be a better choice to power a dashboard that analysts use to explore sales trends? (Answer: Impala, because we've implied these queries are interactive. Hive is much too slow to use as a backend for a Web site)

The Wiki on Hive's Web site is probably the most complete source of information about Hive on the Web, though like most wikis, it's often outdated or invalid. The

Bibliography

The following offer more information on topics discussed in this chapter

- ***Cloudera Impala*** **(free O'Reilly book)**
  - `http://tiny.cloudera.com/impalabook`
- ***Programming Hive*** **(O'Reilly book)**
  - `http://tiny.cloudera.com/programminghive`
- **Data Analysis with Hadoop and Hive at Orbitz**
  - `http://tiny.cloudera.com/dac09b`
- **Sentiment Analysis Using Apache Hive**
  - `http://tiny.cloudera.com/dac09c`
- ***Wired*** **Article on Impala**
  - `http://tiny.cloudera.com/wiredimpala`