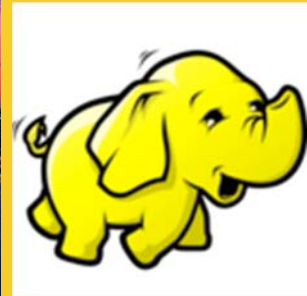




Hive

Originals of slides and source code for examples: <http://www.coreservlets.com/hadoop-tutorial/>
Also see the customized Hadoop training courses (onsite or at public venues) – <http://courses.coreservlets.com/hadoop-training.html>

Customized Java EE Training: <http://courses.coreservlets.com/>
Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live customized Hadoop training (including prep for the Cloudera certification exam), please email info@coreservlets.com

Taught by recognized Hadoop expert who spoke on Hadoop several times at JavaOne, and who uses Hadoop daily in real-world apps. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
 - JSF 2.2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 7 or 8 programming, custom mix of topics
 - Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Spring, Hibernate/JPA, GWT, Hadoop, HTML5, RESTful Web Services

Contact info@coreservlets.com for details



Agenda

- **Hive Overview and Concepts**
- **Installation**
- **Table Creation and Deletion**
- **Loading Data into Hive**
- **Partitioning**
- **Bucketing**
- **Joins**



4

Hive

- **Data Warehousing Solution built on top of Hadoop**
- **Provides SQL-like query language named HiveQL**
 - Minimal learning curve for people with SQL expertise
 - Data analysts are target audience
- **Early Hive development work started at Facebook in 2007**
- **Today Hive is an Apache project under Hadoop**
 - <http://hive.apache.org>

5

Hive Provides

- **Ability to bring structure to various data formats**
- **Simple interface for ad hoc querying, analyzing and summarizing large amounts of data**
- **Access to files on various data stores such as HDFS and HBase**

6

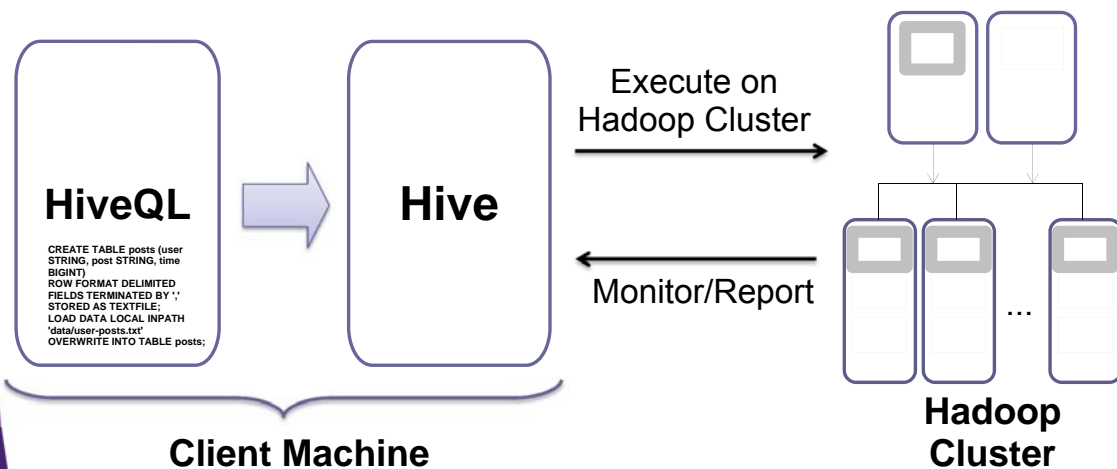
Hive

- **Hive does NOT provide low latency or real-time queries**
- **Even querying small amounts of data may take minutes**
- **Designed for scalability and ease-of-use rather than low latency responses**

7

Hive

- **Translates HiveQL statements into a set of MapReduce Jobs which are then executed on a Hadoop Cluster**



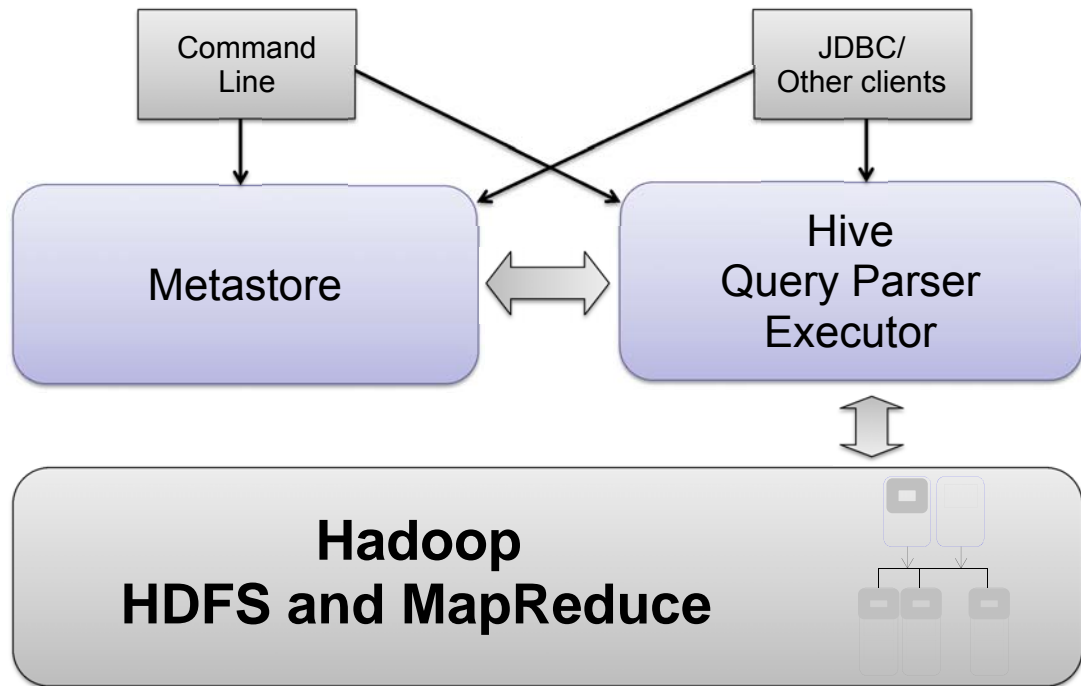
8

Hive Metastore

- **To support features like schema(s) and data partitioning Hive keeps its metadata in a Relational Database**
 - Packaged with Derby, a lightweight embedded SQL DB
 - Default Derby based is good for evaluation and testing
 - Schema is not shared between users as each user has their own instance of embedded Derby
 - Stored in metastore_db directory which resides in the directory that hive was started from
 - Can easily switch another SQL installation such as MySQL

9

Hive Architecture



10

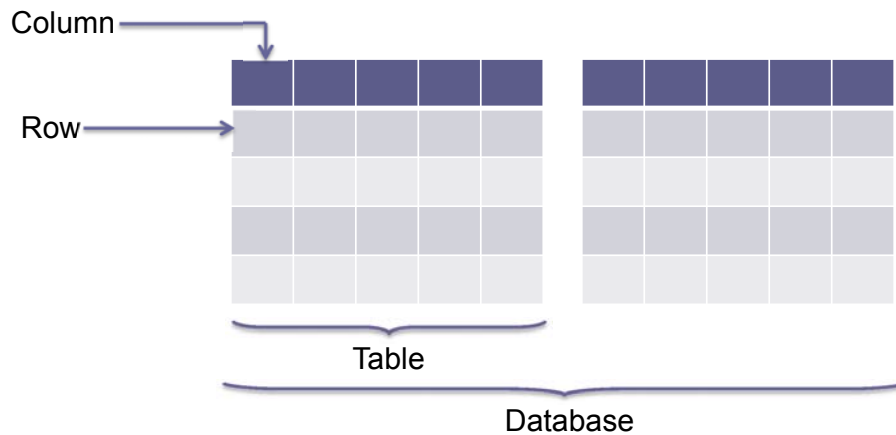
Hive Interface Options

- **Command Line Interface (CLI)**
 - Will use exclusively in these slides
- **Hive Web Interface**
 - <https://cwiki.apache.org/confluence/display/Hive/HiveWebInterface>
- **Java Database Connectivity (JDBC)**
 - <https://cwiki.apache.org/confluence/display/Hive/HiveClient>

11

Hive Concepts

- **Re-used from Relational Databases**
 - **Database:** Set of Tables, used for name conflicts resolution
 - **Table:** Set of Rows that have the same schema (same columns)
 - **Row:** A single record; a set of columns
 - **Column:** provides value and type for a single value



12

Installation Prerequisites

- **Java 6**
 - Just Like Hadoop
- **Hadoop 0.20.x+**
 - No surprise here

13

Hive Installation

- **Set \$HADOOP_HOME environment variable**
 - Was done as a part of HDFS installation
- **Set \$HIVE_HOME and add hive to the PATH**

```
export HIVE_HOME=$CDH_HOME/hive-0.8.1-cdh4.0.0
export PATH=$PATH:$HIVE_HOME/bin
```

- **Hive will store its tables on HDFS and those locations needs to be bootstrapped**

```
$ hdfs dfs -mkdir /tmp
$ hdfs dfs -mkdir /user/hive/warehouse
$ hdfs dfs -chmod g+w /tmp
$ hdfs dfs -chmod g+w /user/hive/warehouse
```

14

Hive Installation

- **Similar to other Hadoop's projects Hive's configuration is in \$HIVE_HOME/conf/hive-site.xml**

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>

  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:10040</value>
  </property>

</configuration>
```

Specify the location of ResourceManager so Hive knows where to execute MapReduce Jobs; by default Hive utilizes LocalJobRunner

15

Run Hive

- **HDFS and YARN need to be up and running**

```
$ hive  
Hive history file=/tmp/hadoop/hive_job_log_hadoop_201207312052_1402761030.txt  
hive>
```

Hive's Interactive Command Line Interface (CLI)



16

Simple Example

- 1. Create a Table**
- 2. Load Data into a Table**
- 3. Query Data**
- 4. Drop the Table**

17

1: Create a Table

- Let's create a table to store data from `$PLAY_AREA/data/user-posts.txt`

```
$ cd $PLAY_AREA
$ hive
Hive history file=/tmp/hadoop/hive_job_log_hadoop_201208022144_2014345460.txt
hive> !cat data/user-posts.txt;
user1,Funny Story,1343182026191
user2,Cool Deal,1343182133839
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394
hive>
```

Launch Hive Command Line Interface (CLI)

Location of the session's log file

Can execute local commands within CLI, place a command in between ! and ;

Values are separate by ',' and each row represents a record; first value is user name, second is post content and third is timestamp

18

1: Create a Table

```
hive> CREATE TABLE posts (user STRING, post STRING, time BIGINT)
> ROW FORMAT DELIMITED
> FIELDS TERMINATED BY ','
> STORED AS TEXTFILE;
OK
Time taken: 10.606 seconds
hive> show tables;
OK
posts
Time taken: 0.221 seconds
hive> describe posts;
OK
user    string
post    string
time    bigint
Time taken: 0.212 seconds
```

1st line: creates a table with 3 columns
2nd and 3rd line: how the underlying file should be parsed
4th line: how to store data

Statements must end with a semicolon and can span multiple rows

Display all of the tables

Result is displayed between "OK" and "Time taken..."

Display schema for posts table

19

2: Load Data Into a Table

```
hive> LOAD DATA LOCAL INPATH 'data/user-posts.txt'  
> OVERWRITE INTO TABLE posts;  
Copying data from file:/home/hadoop/Training/play_area/data/user-posts.txt  
Copying file: file:/home/hadoop/Training/play_area/data/user-posts.txt  
Loading data to table default.posts  
Deleted /user/hive/warehouse/posts  
OK  
Time taken: 5.818 seconds  
hive>
```

Existing records the table *posts* are deleted; data in *user-posts.txt* is loaded into Hive's *posts* table

```
$ hdfs dfs -cat /user/hive/warehouse/posts/user-posts.txt  
user1,Funny Story,1343182026191  
user2,Cool Deal,1343182133839  
user4,Interesting Post,1343182154633  
user5,Yet Another Blog,13431839394
```

Under the covers Hive stores its tables in */user/hive/warehouse* (unless configured differently)

20

3: Query Data

```
hive> select count (1) from posts; ← Count number of records in posts table  
Total MapReduce jobs = 1 ← Transformed HiveQL into 1 MapReduce Job  
Launching Job 1 out of 1  
...  
Starting Job = job_1343957512459_0004, Tracking URL =  
http://localhost:8088/proxy/application_1343957512459_0004/  
Kill Command = hadoop job -Dmapred.job.tracker=localhost:10040 -kill  
job_1343957512459_0004  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  
2012-08-02 22:37:24,962 Stage-1 map = 0%, reduce = 0%  
2012-08-02 22:37:30,497 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.87 sec  
2012-08-02 22:37:31,577 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 0.87 sec  
2012-08-02 22:37:32,664 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 2.64 sec  
MapReduce Total cumulative CPU time: 2 seconds 640 msec  
Ended Job = job_1343957512459_0004  
MapReduce Jobs Launched:  
Job 0: Map: 1 Reduce: 1 Accumulative CPU: 2.64 sec HDFS Read: 0 HDFS Write: 0  
SUCCESS  
Total MapReduce CPU Time Spent: 2 seconds 640 msec  
OK  
4 ← Result is 4 records  
Time taken: 14.204 seconds
```

21

3: Query Data

```
hive> select * from posts where user="user2";
```

```
...
```

```
...
```

```
OK
```

```
user2 Cool Deal 1343182133839
```

```
Time taken: 12.184 seconds
```

Select records for "user2"

Select records whose
timestamp is less or equals
to the provided value

```
hive> select * from posts where time<=1343182133839 limit 2;
```

```
...
```

```
...
```

```
OK
```

```
user1 Funny Story 1343182026191
```

```
user2 Cool Deal 1343182133839
```

```
Time taken: 12.003 seconds
```

```
hive>
```

Usually there are too
many results to display,
then one could utilize
limit command to
bound the display

22

4: Drop the Table

```
hive> DROP TABLE posts; ← Remove the table; use with caution
```

```
OK
```

```
Time taken: 2.182 seconds
```

```
hive> exit;
```

```
$ hdfs dfs -ls /user/hive/warehouse/
```

```
$
```

If hive was managing underlying file then it
will be removed

23

Loading Data

- **Several options to start using data in HIVE**

- Load data from HDFS location

```
hive> LOAD DATA INPATH '/training/hive/user-posts.txt'  
> OVERWRITE INTO TABLE posts;
```

- File is copied from the provided location to /user/hive/warehouse/ (or configured location)

- Load data from a local file system

```
hive> LOAD DATA LOCAL INPATH 'data/user-posts.txt'  
> OVERWRITE INTO TABLE posts;
```

- File is copied from the provided location to /user/hive/warehouse/ (or configured location)

- Utilize an existing location on HDFS

- Just point to an existing location when creating a table

24

Re-Use Existing HDFS Location

```
hive> CREATE EXTERNAL TABLE posts  
> (user STRING, post STRING, time BIGINT)  
> ROW FORMAT DELIMITED  
> FIELDS TERMINATED BY ','  
> STORED AS TEXTFILE  
> LOCATION '/training/hive/';
```

OK

Time taken: 0.077 seconds

hive>

Hive will load all the files under /training/hive directory in posts table

25

Schema Violations

- What would happen if we try to insert data that does not comply with the pre-defined schema?

```
hive> !cat data/user-posts-inconsistentFormat.txt;
user1,Funny Story,1343182026191
user2,Cool Deal,2012-01-05
user4,Interesting Post,1343182154633
user5,Yet Another Blog,13431839394
```

```
hive> describe posts;
OK
user  string
post  string
time  bigint
Time taken: 0.289 seconds
```

Third Column 'post' is of type bigint;
will not be able to convert
'2012-01-05' value

26

Schema Violations

```
hive> LOAD DATA LOCAL INPATH
> 'data/user-posts-inconsistentFormat.txt'
> OVERWRITE INTO TABLE posts;
```

```
OK
Time taken: 0.612 seconds
```

```
hive> select * from posts;
OK
```

```
user1 Funny Story 1343182026191
user2 Cool Deal NULL
user4 Interesting Post 1343182154633
user5 Yet Another Blog 13431839394
Time taken: 0.136 seconds
hive>
```

null is set for any value that
violates pre-defined schema

27

Partitions

- **To increase performance Hive has the capability to partition data**
 - The values of partitioned column divide a table into segments
 - Entire partitions can be ignored at query time
 - Similar to relational databases' indexes but not as granular
- **Partitions have to be properly created by users**
 - When inserting data must specify a partition
- **At query time, whenever appropriate, Hive will automatically filter out partitions**

28

Creating Partitioned Table

```
hive> CREATE TABLE posts (user STRING, post STRING, time BIGINT)
> PARTITIONED BY(country STRING)
> ROW FORMAT DELIMITED
> FIELDS TERMINATED BY ','
> STORED AS TEXTFILE;
```

Partition table based on the value of a country.

```
OK
Time taken: 0.116 seconds
```

```
hive> describe posts;
```

```
OK
user      string
post      string
time      bigint
country   string
Time taken: 0.111 seconds
```

There is no difference in schema between "partition" columns and "data" columns

```
hive> show partitions posts;
```

```
OK
Time taken: 0.102 seconds
hive>
```

29

Load Data Into Partitioned Table

```
hive> LOAD DATA LOCAL INPATH 'data/user-posts-US.txt'  
> OVERWRITE INTO TABLE posts;  
FAILED: Error in semantic analysis: Need to specify partition  
columns because the destination table is partitioned
```

Since the posts table was defined to be partitioned
any insert statement must specify the partition

```
hive> LOAD DATA LOCAL INPATH 'data/user-posts-US.txt'  
> OVERWRITE INTO TABLE posts PARTITION(country='US');  
OK  
Time taken: 0.225 seconds  
  
hive> LOAD DATA LOCAL INPATH 'data/user-posts-AUSTRALIA.txt'  
> OVERWRITE INTO TABLE posts PARTITION(country='AUSTRALIA');  
OK  
Time taken: 0.236 seconds  
hive>
```

Each file is loaded into separate partition;
data is separated by country

30

Partitioned Table

- Partitions are physically stored under separate directories

```
hive> show partitions posts;  
OK  
country=AUSTRALIA  
country=US  
Time taken: 0.095 seconds  
hive> exit;
```

There is a directory for
each partition value

```
$ hdfs dfs -ls -R /user/hive/warehouse/posts  
/user/hive/warehouse/posts/country=AUSTRALIA  
/user/hive/warehouse/posts/country=AUSTRALIA/user-posts-AUSTRALIA.txt  
/user/hive/warehouse/posts/country=US  
/user/hive/warehouse/posts/country=US/user-posts-US.txt
```

31

Querying Partitioned Table

- There is no difference in syntax
- When partitioned column is specified in the where clause entire directories/partitions could be ignored

Only "COUNTRY=US" partition will be queried,
"COUNTRY=AUSTRALIA" partition will be ignored

```
hive> select * from posts where country='US' limit 10;
OK
user1 Funny Story 1343182026191      US
user2 Cool Deal 1343182133839      US
user2 Great Interesting Note 13431821339485      US
user4 Interesting Post 1343182154633      US
user1 Humor is good 1343182039586      US
user2 Hi I am user #2 1343182133839      US
Time taken: 0.197 seconds
```

32

Bucketing

- Mechanism to query and examine random samples of data
- Break data into a set of buckets based on a hash function of a "bucket column"
 - Capability to execute queries on a sub-set of random data
- Doesn't automatically enforce bucketing
 - User is required to specify the number of buckets by setting # of reducer

```
hive> mapred.reduce.tasks = 256;
OR
hive> hive.enforce.bucketing = true;
```

Either manually set the # of reducers to be the number of buckets or you can use 'hive.enforce.bucketing' which will set it on your behalf

33

Create and Use Table with Buckets

```
hive> CREATE TABLE post_count (user STRING, count INT)
  > CLUSTERED BY (user) INTO 5 BUCKETS;
OK
Time taken: 0.076 seconds

hive> set hive.enforce.bucketing = true;
hive> insert overwrite table post_count
  > select user, count(post) from posts group by user;
Total MapReduce jobs = 2
Launching Job 1 out of 2
...
Launching Job 2 out of 2
...
OK
Time taken: 42.304 seconds
hive> exit;
$ hdfs dfs -ls -R /user/hive/warehouse/post_count/
/user/hive/warehouse/post_count/000000_0
/user/hive/warehouse/post_count/000001_0
/user/hive/warehouse/post_count/000002_0
/user/hive/warehouse/post_count/000003_0
/user/hive/warehouse/post_count/000004_0
```

Declare table with 5 buckets for user column

of reducer will get set 5

Insert data into post_count bucketed table; number of posts are counted up for each user

A file per bucket is created; now only a sub-set of buckets can be sampled

34

Random Sample of Bucketed Table

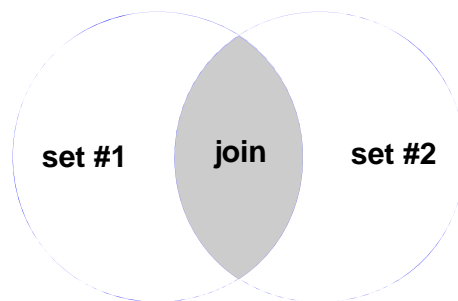
```
hive> select * from post_count TABLESAMPLE(BUCKET 1 OUT OF 2);
OK
user5 1
user1 2
Time taken: 11.758 seconds
hive>
```

Sample approximately 1 for every 2 buckets

35

Joins

- **Joins in Hive are trivial**
- **Supports outer joins**
 - left, right and full joins
- **Can join multiple tables**
- **Default Join is Inner Join**
 - Rows are joined where the keys match
 - Rows that do not have matches are not included in the result



36

Simple Inner Join

- **Let's say we have 2 tables: posts and likes**

```
hive> select * from posts limit 10;
```

OK

```
user1 Funny Story 1343182026191
user2 Cool Deal 1343182133839
user4 Interesting Post 1343182154633
user5 Yet Another Blog 1343183939434
```

Time taken: 0.108 seconds

```
hive> select * from likes limit 10;
```

OK

```
user1 12 1343182026191
user2 7 1343182139394
user3 0 1343182154633
user4 50 1343182147364
```

Time taken: 0.103 seconds

```
hive> CREATE TABLE posts_likes (user STRING, post STRING, likes_count INT);
```

OK

Time taken: 0.06 seconds

We want to join these 2 data-sets and produce a single table that contains user, post and count of likes

37

Simple Inner Join

```
hive> INSERT OVERWRITE TABLE posts_likes
> SELECT p.user, p.post, l.count
> FROM posts p JOIN likes l ON (p.user = l.user);
OK
Time taken: 17.901 seconds
```

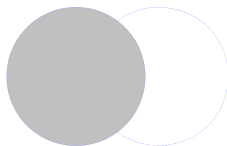
Two tables are joined based on user column; 3 columns are selected and stored in posts_likes table

```
hive> select * from posts_likes limit 10;
OK
user1 Funny Story      12
user2 Cool Deal        7
user4 Interesting Post  50
Time taken: 0.082 seconds
hive>
```

38

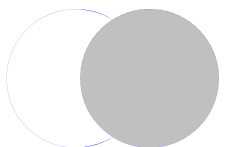
Outer Join

- Rows which will not join with the 'other' table are still included in the result



Left Outer

- Row from the first table are included whether they have a match or not. Columns from the unmatched (second) table are set to null.



Right Outer

- The opposite of Left Outer Join: Rows from the second table are included no matter what. Columns from the unmatched (first) table are set to null.



Full Outer

- Rows from both sides are included. For unmatched rows the columns from the 'other' table are set to null.

39

Outer Join Examples

```
SELECT p.*, l.*  
FROM posts p LEFT OUTER JOIN likes l ON (p.user = l.user)  
limit 10;
```

```
SELECT p.*, l.*  
FROM posts p RIGHT OUTER JOIN likes l ON (p.user = l.user)  
limit 10;
```

```
SELECT p.*, l.*  
FROM posts p FULL OUTER JOIN likes l ON (p.user = l.user)  
limit 10;
```

40

Resources

- <http://hive.apache.org/>
- **Hive Wiki**
 - <https://cwiki.apache.org/confluence/display/Hive/Home>

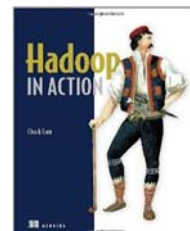


Hive

Edward Capriolo (Author), Dean Wampler (Author), Jason Rutherglen (Author)
O'Reilly Media; 1 edition (October 3, 2012)

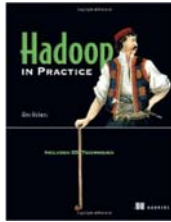
Chapter About Hive Hadoop in Action

Chuck Lam (Author)
Manning Publications; 1st Edition (December, 2010)



41

Resources



Chapter about Hive
Hadoop in Practice
Alex Holmes (Author)
Manning Publications; (October 10, 2012)



Wrap-Up

Customized Java EE Training: <http://courses.coreservlets.com/>

Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Summary

- **We learned about**
 - Hive Concepts
 - Hive Installation
 - Table Creation and Deletion
 - Loading Data into Hive
 - Partitioning
 - Bucketing
 - Joins

44

© 2012 coreservlets.com and [Dima May](#)



Questions?

More info:

<http://www.coreservlets.com/hadoop-tutorial/> – Hadoop programming tutorial
<http://courses.coreservlets.com/hadoop-training.html> – Customized Hadoop training courses, at public venues or onsite at *your* organization
<http://courses.coreservlets.com/Course-Materials/java.html> – General Java programming tutorial
<http://www.coreservlets.com/java-8-tutorial/> – Java 8 tutorial
<http://www.coreservlets.com/JSF-Tutorial/jsf2/> – JSF 2.2 tutorial
<http://www.coreservlets.com/JSF-Tutorial/primefaces/> – PrimeFaces tutorial
<http://coreservlets.com/> – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training

Customized Java EE Training: <http://courses.coreservlets.com/>

Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.