A Few Big Data Tools

(Rao)

CS5540

Agenda

- Apache Hadoop
- Apache Pig
- Apache HBase
- IBM Jaql
- Apache Hive

All of the above are part of IBM BigInsights/Hadoop distribution.

- Other tools of interest
  - MongoDB, CouchDB, Tajo, Spark, Flink

Hadoop Distributed File System (HDFS)

- http://hadoop.apache.org
- **hadoop fs** […….]
- Examples

```
hadoop fs [-help [cmd]]
hadoop fs [-ls <path>]
hadoop fs [-lsr <path>]
hadoop fs [-du <path>]
hadoop fs [-cat <src>]
hadoop fs [-mv <src> <dst>]
hadoop fs [-cp <src> <dst>]
hadoop fs [-mkdir <path>]
hadoop fs [-touchz <path>]
```

Commands in HDFS

- Examples

```
hadoop fs [-rm <path>]
hadoop fs [-rmr <path>]
hadoop fs [-copyFromLocal <localsrc> ... <dst>]
hadoop fs [-moveFromLocal <localsrc> ... <dst>]
hadoop fs [-copyToLocal <src> <localdst>]
hadoop fs [-moveToLocal <src> <localdst>] // not implemented yet?
hadoop fs [-stat <path>]
hadoop fs [-tail <file>]
hadoop fs [-chmod <MODE[,MODE]... | OCTALMODE> PATH...]
hadoop fs -setrep            // change replication factor
hadoop fs –put  <src> <dst> // copy multiple files to HDFS
hadoop fs –get  <src> <dst> // copy multiple files from HDFS
```

Examples

```
biadmin@bivm:~> hadoop fs -ls
Found 1 items
drwx------   - biadmin biadmin          0 2014-07-27 22:58 .staging
biadmin@bivm:~> hadoop fs -ls /
Found 5 items
drwxrwxr-x   - hdfs  biadmin          0 2014-06-14 15:10 /biginsights
drwxr-xr-x   - hdfs  biadmin          0 2014-06-14 14:49 /hadoop
drwxr-xr-x   - hbase biadmin          0 2014-07-31 22:44 /hbase
drwxrwxrwt   - hdfs  biadmin          0 2014-06-14 15:37 /tmp
drwxrwxrwx   - hdfs  biadmin          0 2014-07-31 22:03 /user
biadmin@bivm:~> hadoop fs -mkdir /user/rao
biadmin@bivm:~> hadoop fs -copyFromLocal /mnt/hgfs/DBS/
10_2012.tar.gz  12_2012.tar.gz  3_2014.tar.gz   dataset1/       nested.json
11_2012.tar.gz  2_2013.tar.gz   4_2014.tar.gz   example.json    scripts/
1_2014.tar.gz   2_2014.tar.gz   9_2012.tar.gz   hashtags.json   test.json
biadmin@bivm:~> hadoop fs -copyFromLocal /mnt/hgfs/DBS/scripts/
extract.py   output.json
biadmin@bivm:~> hadoop fs -copyFromLocal /mnt/hgfs/DBS/scripts/output.json /user/rao/
biadmin@bivm:~> hadoop jar /opt/ibm/biginsights/IHC/hadoop-example.jar wordcount /user/r
ao/output.json /usr/rao/wordcount_output
14/07/31 22:50:34 INFO input.FileInputFormat: Total input paths to process : 1
14/07/31 22:50:34 INFO mapred.JobClient: Running job: job_201407312244_0001
14/07/31 22:50:35 INFO mapred.JobClient:  map 0% reduce 0%
14/07/31 22:51:08 INFO mapred.JobClient:  map 96% reduce 0%
14/07/31 22:51:11 INFO mapred.JobClient:  map 100% reduce 0%
```

Examples

```
14/07/31 22:52:15 INFO mapred.JobClient:     SPLIT_RAW_BYTES=110
14/07/31 22:52:15 INFO mapred.JobClient:     COMBINE_INPUT_RECORDS=2270404
14/07/31 22:52:15 INFO mapred.JobClient:     COMBINE_OUTPUT_RECORDS=781528
14/07/31 22:52:15 INFO mapred.JobClient:     REDUCE_INPUT_GROUPS=390601
14/07/31 22:52:15 INFO mapred.JobClient:     REDUCE_SHUFFLE_BYTES=10153979
14/07/31 22:52:15 INFO mapred.JobClient:     REDUCE_INPUT_RECORDS=390601
14/07/31 22:52:15 INFO mapred.JobClient:     REDUCE_OUTPUT_RECORDS=390601
14/07/31 22:52:15 INFO mapred.JobClient:     SPILLED_RECORDS=1172129
14/07/31 22:52:15 INFO mapred.JobClient:     CPU_MILLISECONDS=7890
14/07/31 22:52:15 INFO mapred.JobClient:     PHYSICAL_MEMORY_BYTES=1350909952
14/07/31 22:52:15 INFO mapred.JobClient:     VIRTUAL_MEMORY_BYTES=2488422400
14/07/31 22:52:15 INFO mapred.JobClient:     COMMITTED_HEAP_BYTES=2097152000
14/07/31 22:52:15 INFO mapred.JobClient:   File Input Format Counters
14/07/31 22:52:15 INFO mapred.JobClient:     Bytes Read=22881018
14/07/31 22:52:15 INFO mapred.JobClient:   org.apache.hadoop.mapreduce.lib.output.FileOu
tputFormat$Counter
14/07/31 22:52:15 INFO mapred.JobClient:     BYTES_WRITTEN=8591896
biadmin@bivm:~> hadoop fs -ls /user/rao/wordcount_output
ls: `/user/rao/wordcount_output': No such file or directory
biadmin@bivm:~> hadoop fs -ls /usr/rao/wordcount_output
Found 3 items
-rw-r--r--   1 biadmin biadmin          0 2014-07-31 22:52 /usr/rao/wordcount_output/_SU
CCESS
drwxr-xr-x   - biadmin biadmin          0 2014-07-31 22:50 /usr/rao/wordcount_output/_lo
gs
-rw-r--r--   1 biadmin biadmin    8591896 2014-07-31 22:52 /usr/rao/wordcount_output/par
t-r-00000
biadmin@bivm:~>
```

Examples

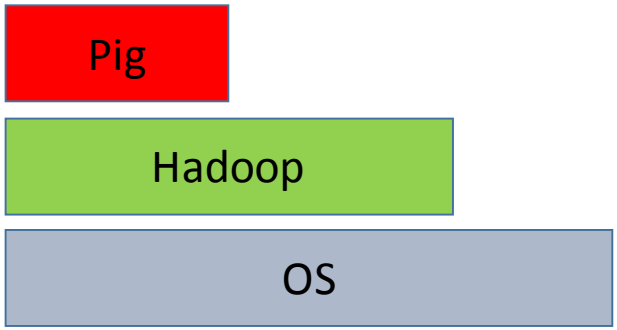
```
  aggregatewordhist: An Aggreg
togram of the words in the inp
  dbcount: An example job that
  grep: A map/reduce program t
  join: A job that effects a j
  multifilewc: A job that coun
  pentomino: A map/reduce tile
blems.
  pi: A map/reduce program tha
  randomtextwriter: A map/redu
 per node.
  randomwriter: A map/reduce p
  secondarysort: An example de
  sleep: A job that sleeps at
  sort: A map/reduce program t
  sudoku: A sudoku solver.
  teragen: Generate data for t
  terasort: Run the terasort
  teravalidate: Checking resul
  wordcount: A map/reduce prog
biadmin@bivm:~> hadoop jar /op
biadmin@bivm:~> hadoop jar /op
Usage: wordcount <in> <out>
biadmin@bivm:~>
```

```
289464568026120192}      1
289464568026132480}      1
Africa",         387
Aires", 3997
America",        827
Bataar",         4
Caledonia",      33
Canada)",        72097
Central 387
City",   2542
Date     269
Delhi", 277
Dhabi", 2280
Is.",    30
Island",         18
Jayawardenepura",        16
Kong",   175
Line     269
Lumpur",         1644
Moresby",        2
Paz",    376
Petersburg",     138
Time     79383
Verde    17
West",   269
null,    109221
{"user:time_zone":       390585
biadmin@bivm:~>
```

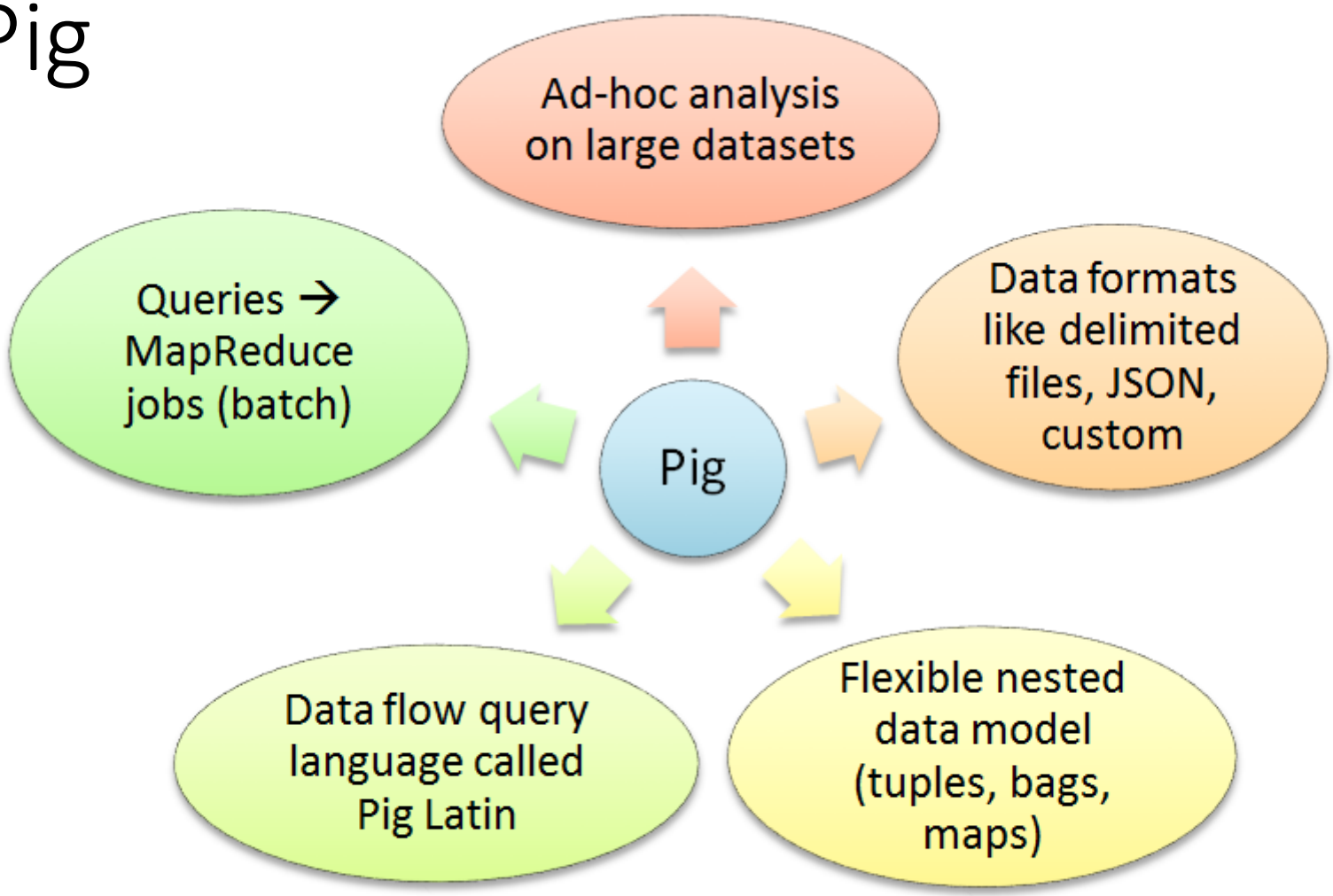
output of Word Count program

Apache Pig

- An open-source tool for data analysis on large datasets
  - http://pig.apache.org
- Developed by engineers at Yahoo!
- Pig runs on top of Hadoop
  - Pig accesses files from HDFS (and not from local file system)



Pig



{ (location, “MO”), (hashtags, (#skin, #lung, #stomach)), (lang,``en”) }

Basic Commands

- To start the Pig shell type pig in the terminal
- To load a file into a logical table
  - Type the following command in the Pig shell

```
grunt> A = load 'raopr/Subway_Entrances.csv' using PigStorage(',');
```

Creates a logical table A

$0	$1	$2	$3	$4	$5
..	..	..	..	..	..
1896	40.74531233400006	73.98895400099991	Broadway & 28th St At Sw Corner (Downtown Only)	http://www.mta.info/nyct/service/	N-R

Basic Commands

- To load a file into a logical table and assign names to the columns
  - Type the following command in the Pig shell

```
grunt> B = load 'raopr/Subway_Entrances.csv' using PigStorage(',') as (id, lat, lon, address, url, code);
```

Creates a logical table B

id	lat	lon	address	url	code
..	..	..	..	..	..
1896	40.74531233400006	73.98895400099991	Broadway & 28th St At Sw Corner (Downtown Only)	http://www.mta.info/nyct/service/	N-R

Basic Commands

- To display a table on screen, use the dump command

```
grunt> dump A;
```

```
grunt> dump B;
```

- To store a table in HDFS; output is stored in the given directory (in pieces)

```
grunt> store A into 'raopr/outA'
grunt> store A into 'raopr/outB'
```

Basic Commands

- To select a few columns in a table

```
grunt> C = foreach A generate $0, $2, $3;
grunt> dump C;
```

```
grunt> D = foreach B generate id, lon, address;
grunt> dump D;
```

- To select a subset of rows in a table

```
grunt> E = filter A by $0 == '1898' or $5 == '4-6-6 Express';
grunt> dump E;
```

```
grunt> F = filter B by id != ’1898’;
grunt> dump F;
```

Basic Commands

- To join two logical tables

```
grunt> Z = join X by id, Y by id;
grunt> dump Z;
```

- To order the results

```
grunt> U = order X by $1;
grunt> dump U;
```

Examples

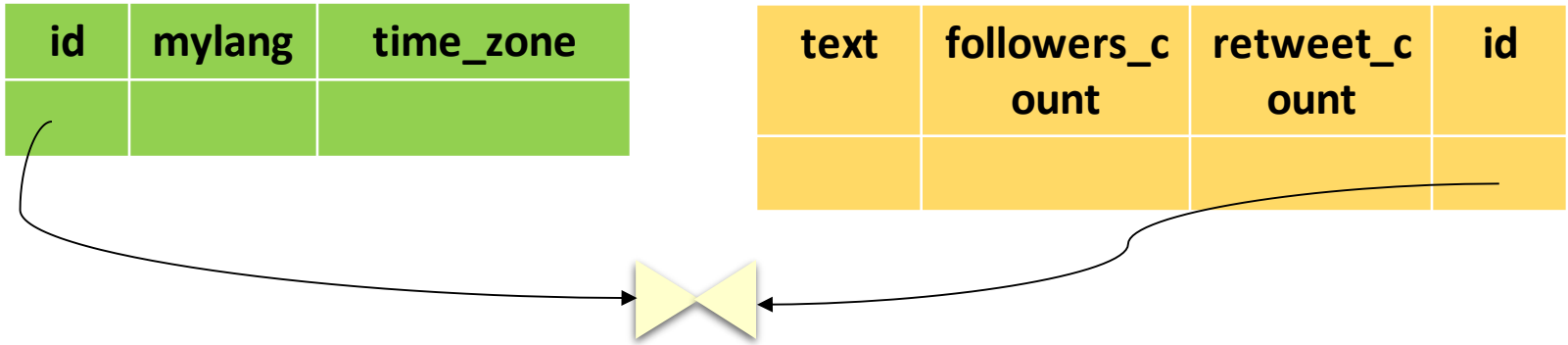
```
dbs@DBS:~/go/bin$ ./json2csv -k id,lang,user.time_zone -i ~/Tweets/dataset1/1_10.json -o ~/Tweets
/json/table1.csv
dbs@DBS:~/go/bin$ head ~/Tweets/json/table1.csv
289429398778687488,ja,""
289429398774497280,pt,""
289429398770307072,th,Pacific Time (US & Canada)
289429398761906176,nl,Athens
289429398766116864,ja,Irkutsk
289429398778691584,en,Eastern Time (US & Canada)
289429398774493184,id,Kuala Lumpur
289429398766092288,pt,""
289429398753529856,es,Quito
289429398782889984,ar,""
dbs@DBS:~/go/bin$ ./json2csv -k id,text,user.follower_count,retweet_count -i ~/Tweets/dataset1/1_
10.json -o ~/Tweets/json/table2.csv
dbs@DBS:~/go/bin$ head ~/Tweets/json/table2.csv
289429398778687488,ポカーン。 ,"",0
289429398774497280,"RT @caioabreufrases: Se não brilha mais, não insista. Lâmpada queimada não se
 arruma. Se troca por outra.","",0
289429398770307072,RT @frontagemusic: ถ้าหัวใจผมจะหยุดเต้นไป ผมจะรู้สึกอย่างไร การเคลื่อนไหวสุดท้ายจะมีความหมา
ยกับเขาแค่ไหน หรือสุดท้ายมันจะเป็นแค่ผมคนเดีย ...,"",0
289429398761906176,Hamster overleden :(,"",0
289429398766116864,なんかアイドルだらけでワロタ,"",0
```

https://github.com/jehiah/json2csv

```
dbs@DBS:~/Tweets/scripts$ ./extract.py ../dataset1/1_10.json ../json/output.json id text retweet_count user.followe
rs_count
```

https://github.com/raopr/twitter-project.git

Example: Join

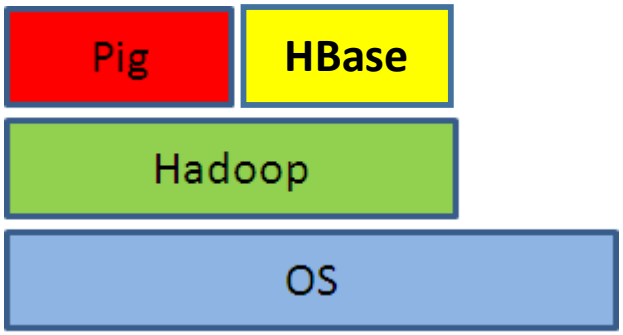



```
grunt> A = load '/user/rao/table1.csv' using PigStorage(',') as (id:chararray, mylang:chararray, time_zone:chararray);
grunt> B = load '/user/rao/table3.json' using JsonLoader('text:chararray,followers_count:int,retweet_count:int,id:chararray');
grunt> C = filter A by mylang == 'en';
grunt> D = filter B by followers_count > 5000;
grunt> E = join C by id, D by id;
grunt> store E into '/user/rao/output' using JsonStorage();
INFO  [JobControl] org.apache.hadoop.mapreduce.lib.input.FileInputFormat     - Total input paths to process : 1
INFO  [JobControl] org.apache.hadoop.mapreduce.lib.input.FileInputFormat     - Total input paths to process : 1
```

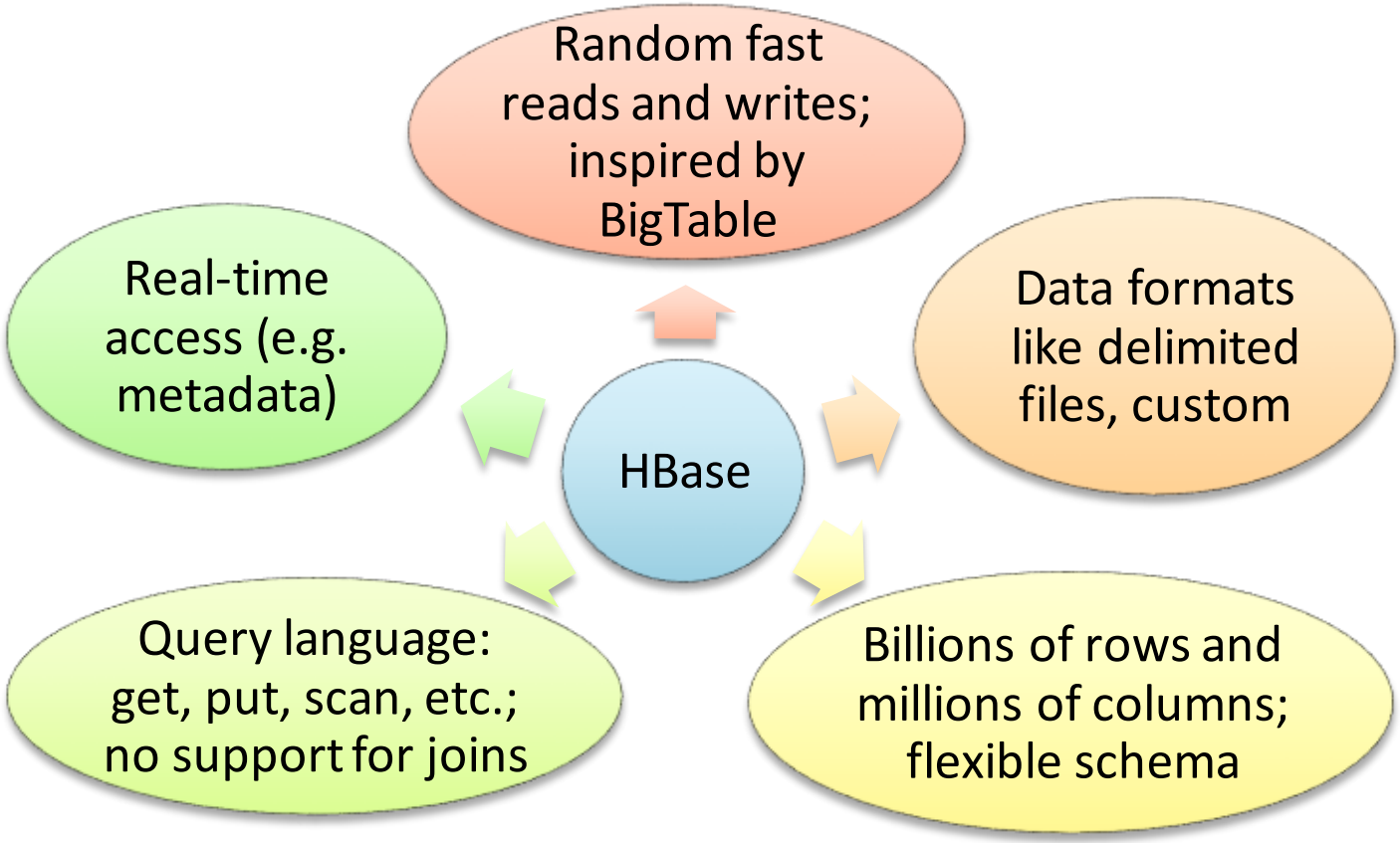
Join

Apache HBase

- HBase is an open-source project and is based on Google's BigTable
  - http://hbase.apache.org
- Used in Big Data applications where there is a need to manage very large tables with billions of rows and millions of columns
  - Cluster of machines
  - Random reads/writes



HBase


Random fast reads and writes; inspired by BigTable
Real-time access (e.g. metadata)
Data formats like delimited files, custom
HBase
Query language: get, put, scan, etc.; no support for joins
Billions of rows and millions of columns; flexible schema

Data Model

rowkey	timestamp	ColumnFamily **meta**	ColumnFamily **content**
www.bbc.co.uk	t2	meta:creator="johndoe"	
www.bbc.co.uk	t3	meta:location="london"	
www.cnn.com	t9		content:info="royals"
www.cnn.com	t7		content:info="hello world"

Rows are sorted by rowkey

We refer to attributes using the ColumnFamily name as the prefix
- E.g., meta:creator, content:info

Another View

rowkey	TS	columnfamily_1		columnfamily_2	
		attr1	attr2	attr3	Attr4
123456790		NULL	….	….	….
123456791		….	NULL	NULL	….
123456792		….	….	….	….
123456793		NULL	NULL	….	….
123456794		….	….	….	NULL

Sparse table

Basic Commands

- To start HBase, type **hbase shell**
- To create a table with name **webtable** and two column families **meta** and **content**

```
hbase(main):001:0> create 'webtable', 'meta', 'content'
```

- To delete a table (first disable it)

```
hbase(main):005:0> disable 'webtable'
hbase(main):006:0> drop 'webtable'
```

Basic Commands

- Insert a row with rowkey www.bbc.co.uk and value abc for attribute creator in ColumnFamily meta
- Insert using the same rowkey and value 123 for attribute location in meta
  - The system picks a timestamp if not specified

```
hbase(main):009:0> put 'webtable', 'www.bbc.co.uk', 'meta:creator', 'abc'
0 row(s) in 0.1410 seconds

hbase(main):010:0> put 'webtable', 'www.bbc.co.uk', 'meta:location', '123'
0 row(s) in 0.0080 seconds
```

Basic Commands

- Print a table

```
hbase(main):011:0> scan 'webtable'
ROW                    COLUMN+CELL
 www.bbc.co.uk         column=meta:creator, timestamp=1372698985649, value=abc
 www.bbc.co.uk         column=meta:location, timestamp=1372698995764,
value=123
1 row(s) in 0.0270 seconds
```

Assignment

Create the below table using HBase

people

rowkey	TS	courses		hobby	
		CS	ECE	soccer	baseball
John		cs490			
Mary					yes
Bob			ece210		
Alice				yes	
Mike					yes
Jill				no	

Basic Commands

- To get a row or cell contents

```
hbase(main):012:0> get 'webtable', 'www.bbc.co.uk'

hbase(main):013:0> get 'webtable', 'www.bbc.co.uk', 'meta:location'
```

- To delete a row or cell contents

```
hbase(main):014:0> deleteall 'webtable', 'www.bbc.co.uk'

hbase(main):015:0> delete 'webtable', 'www.bbc.co.uk', 'meta:location'
```

- If a file contains HBase commands, you can run it

```
hbase shell < filename
```

Examples

```
biadmin@bivm:/mnt/hgfs/DBS/json> tail table1.csv
289464568021917696,sv,Central Time (US & Canada)
289464568013533184,en,London
289464567988379648,en,""
289464567988379648,en,""
289464568026132480,en,Berlin
289464567992578048,en,""
289464568005156864,en,Central Time (US & Canada)
289464568005160960,en,Central Time (US & Canada)
289464568026107904,en,""
289464568021913600,und,Mexico City
biadmin@bivm:/mnt/hgfs/DBS/json>
```

Input CSV file

Bulk loading

```
biadmin@bivm:~> /opt/ibm/biginsights/hbase/bin/hbase org.apache.hadoop.hbase.mapreduce.ImportT
sv '-Dimporttsv.separator=,' -Dimporttsv.columns=HBASE_ROW_KEY,tweet:lang,user:time_zone twitt
er /user/rao/table1.csv
```

Examples

```
hbase(main):006:0> get 'twitter', '289432712283033600'
COLUMN                                    CELL
 tweet:lang                               timestamp=1406945902170, value=en
 user:time_zone                           timestamp=1406945902170, value=""
2 row(s) in 0.0260 seconds

hbase(main):007:0> get 'twitter', '289432712283033600', 'tweet:lang'
COLUMN                                    CELL
 tweet:lang                               timestamp=1406945902170, value=en
1 row(s) in 0.0110 seconds
```

Retrieve a particular rowkey and column attribute

Examples

Insert a new column attribute for a rowkey

```
hbase(main):014:0> put 'twitter', '289432712283033600', 'tweet:text', 'Hello World'
```

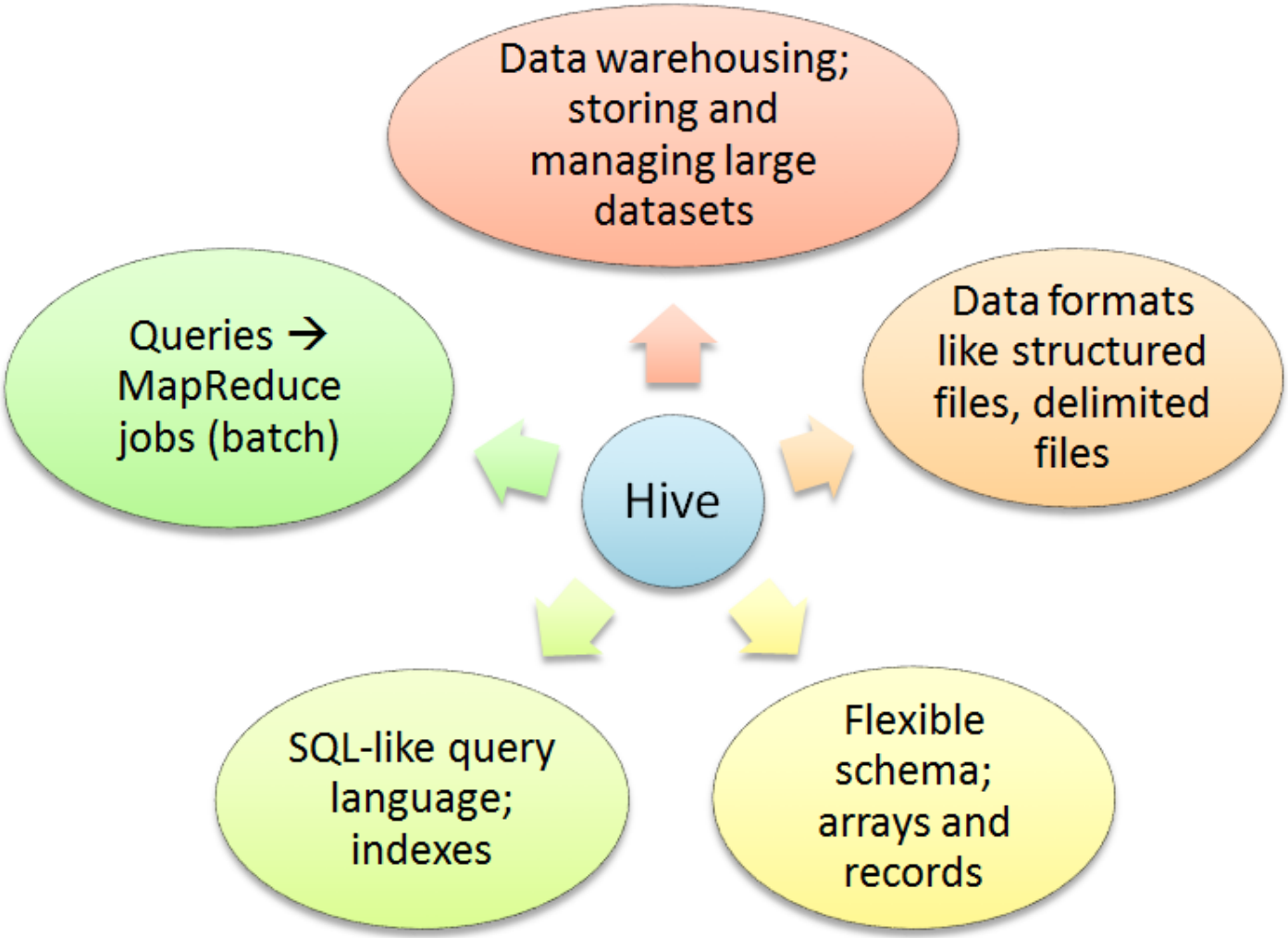
```
hbase(main):013:0> get 'twitter', '289432712283033600'
COLUMN                                CELL
 tweet:lang                           timestamp=1406946761615, value=jp
 tweet:text                           timestamp=1406946782120, value=Hello World
 user:time_zone                       timestamp=1406945902170, value=""
3 row(s) in 0.0050 seconds
```

Examples

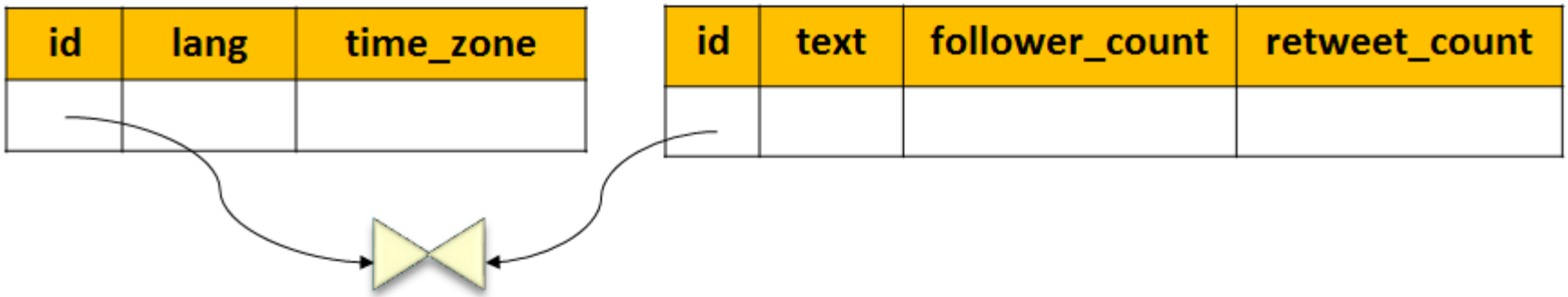
Selection

```
hbase(main):033:0> scan 'twitter', { FILTER => "SingleColumnValueFilter('tweet','lang',=, 'binary:hi')" }
ROW                                    COLUMN+CELL
 289429830800400384                    column=tweet:lang, timestamp=1406945902170, value=hi
 289429830800400384                    column=user:time_zone, timestamp=1406945902170, value=Chennai
 289431936332611584                    column=tweet:lang, timestamp=1406945902170, value=hi
 289431936332611584                    column=user:time_zone, timestamp=1406945902170, value=Kolkata
 289432125055315968                    column=tweet:lang, timestamp=1406945902170, value=hi
 289432125055315968                    column=user:time_zone, timestamp=1406945902170, value=New Delhi
3 row(s) in 0.1360 seconds
```

Apache Hive


Data warehousing; storing and managing large datasets
Queries → MapReduce jobs (batch)
Data formats like structured files, delimited files
Hive
SQL-like query language; indexes
Flexible schema; arrays and records

Examples



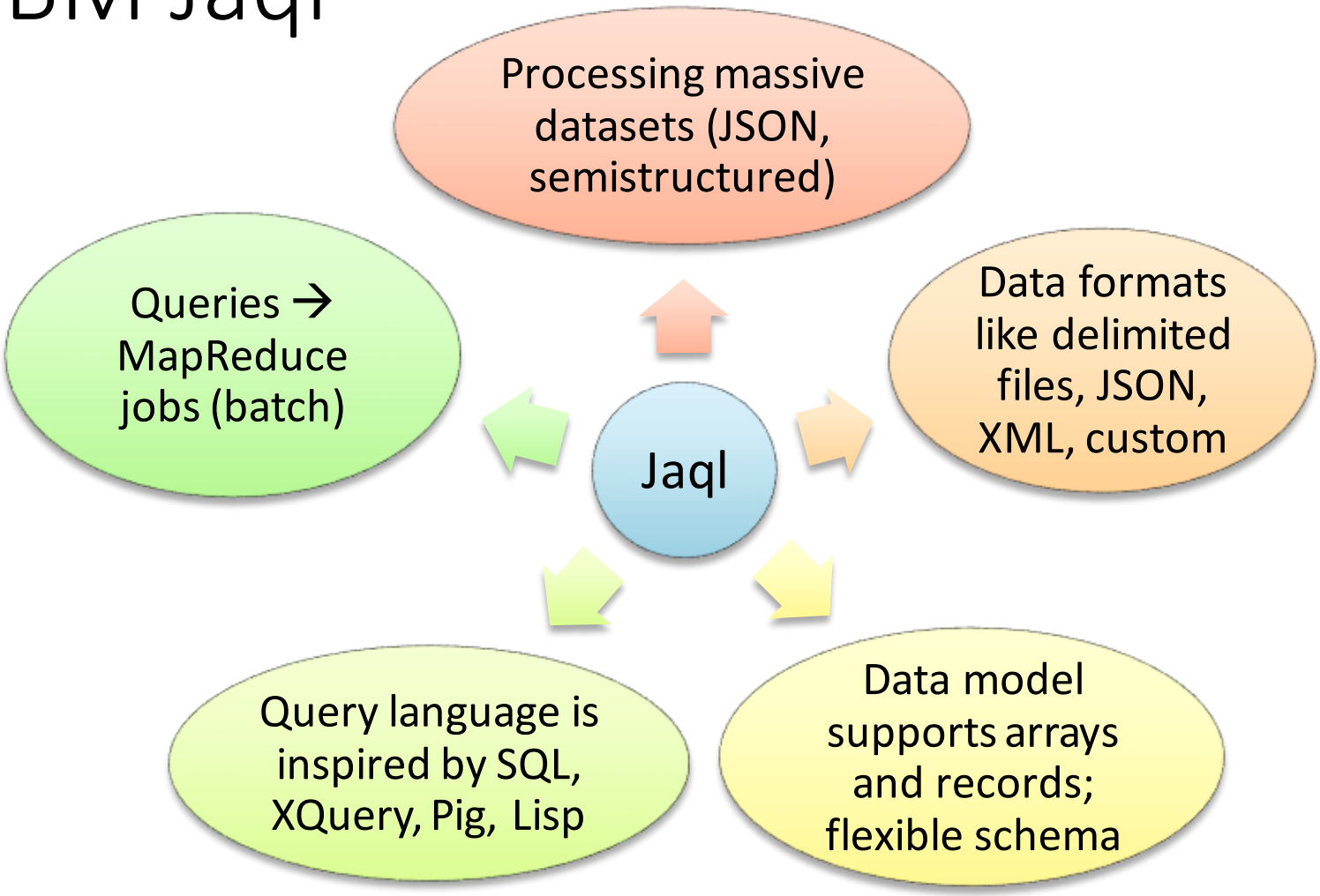

```
hive> create table A (id BIGINT, lang STRING, time_zone STRING) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

hive> create table B (id BIGINT, text STRING, follower_count INT, retweet_count INT) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',';

hive> load data local inpath '/home/biadmin/Desktop/table1.csv' overwrite into table A;

hive> load data local inpath '/home/biadmin/Desktop/table3.csv' overwrite into table B;

hive> select A.id, B.text, B.follower_count FROM A JOIN B ON (A.id = B.id);
```

Join

IBM Jaql

- A query language for JSON
- JSON records (name/value pairs), arrays
- Can do selections, joins, group by, etc.

IBM Jaql



[{“location”: “MO”, “hashtags”: [“#skin”, “#lung”, “#stomach”], “lang”: “en”}, {…}, …]

Examples

```
jaql> b = {user: 'praveen', 'date': '10/12/13'};

jaql> b.user;
"praveen"

jaql> b.date;
"10/12/13"

jaql>
jaql> c = {user: 'john', name: {first: 'Johnathan', last: 'Doe'}, age: 35};

jaql> c.name.first;
"Johnathan"
```

Record

Record with nesting

Like a path expression

Examples

```
jaql> c = {user: 'john', name: {first: 'Johnathan', last: 'Doe'}, age: 35};

jaql> c.name.first;
"Johnathan"

jaql> d = [1,2,3,4,5,6];

jaql> d -> filter 2 <= $ < 4;
[
  2,
  3
]
```

Array

Filter each array element ($) using a predicate

Indicates flow of data

```
jaql> e = "Jaql syntax is ugly!";
```

You may agree!

Examples

```
jaql> employees = [{first: 'john', last: 'doe'}, {first: 'mary', last: 'smith'}, {first: 'alice', last: 'ki
ng'}];

jaql> employees -> filter $.last != 'doe' and $.first != 'mary';
[
  {
    "first": "alice",
    "last": "king"
  }
]
```

Filter each record ($) using a predicate

```
jaql> employees -> filter each person (person.last != 'doe' and person.first != 'mary');
[
  {
    "first": "alice",
    "last": "king"
  }
]
```

Another syntax in place of $

Examples

```
jaql> employees;
[
  {
    "first": "john",
    "last": "doe"
  },
  {
    "first": "mary",
    "last": "smith"
  },
  {
    "first": "alice",
    "last": "king"
  }
]

jaql> employees -> transform $.first;
[
  "john",
  "mary",
  "alice"
]
```

Extract data

Examples

Bulk loading

Schema

```
jaql> tweets = read(del(location="/user/rao/table1.csv", schema=schema{id:long, language:string, time_zone:
string}));

jaql> tweets[1];
{
  "id": 289429398774497280,
  "language": "pt",
  "time_zone": ""
}

jaql> tweets[10];
{
  "id": 289429398757707776,
  "language": "en",
  "time_zone": "Mexico City"
}
```

Examples

```
jaql>
jaql> comment = "For jaqlGet to succeed, input should be an array of JSON records [{}, {}, ...]";

jaql> mytweets = jaqlGet("file:///mnt/hgfs/DBS/dataset1/tweets_small.json");

jaql> mytweets -> transform {$.text, $.user.followers_count};
```

Bulk loading

Output

```
  {
    "text": "@JordanElliott94 hahaha, we were texting one night and you were drunk and you sent me that cau
se I didn\'t believe you hahaha",
    "followers_count": 540
  },
  {
    "text": "@WildClothes_ Merci beaucoup du renseignement ♥",
    "followers_count": 838
  },
  {
    "text": "Anyway lemme go back to this NEVER point I made. The amount of times I\'ve said I would NEVER
do this or that and I have done it..",
    "followers_count": 604
  },
  {
    "text": "As letras do Natiruts são iradasss",
    "followers_count": 143
  },
  {
    "text": "Why must doin right feel so wrong.",
    "followers_count": 159
  }
]
```

Examples

```
jaql> mytweets[0];
{
  "contributors": null,
  "coordinates": null,
  "created_at": "Thu Jan 10 17:52:00 +0000 2013",
  "entities": {
    "hashtags": [],
    "urls": [],
    "user_mentions": []
  },
  "favorited": false,
  "geo": null,
  "id": 289429398778687488,
  "id_str": "289429398778687488",
  "in_reply_to_screen_name": null,
  "in_reply_to_status_id": null,
  "in_reply_to_status_id_str": null,
  "in_reply_to_user_id": null,
  "in_reply_to_user_id_str": null,
  "lang": "ja",
  "place": null,
  "retweet_count": 0,
  "retweeted": false,
  "source": "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for iPhone</a>",
  "text": "ポカーン。",
  "truncated": false,
  "user": {
```

Examples

Let’s split a tweet into two pieces

```
jaql> tweet_info = mytweets -> transform {$.id, $.lang, $.text};

jaql> user_info = mytweets -> transform {$.id, $.user.name, $.user.followers_count, $.user.time_zone};

jaql> tweet_info[0];
{
  "id": 289429398778687488,
  "lang": "ja",
  "text": "ポカーン。"
}

jaql> user_info[0]
;
{
  "id": 289429398778687488,
  "name": "まさひろ。",
  "followers_count": 113,
  "time_zone": null
}
```

Let’s do a join on user_info and tweet_info

Examples

Join condition

Projection

```
jaql> join tweet_info, user_info where tweet_info.id == user_info.id into {tweet_info.id, tweet_info.lang,
tweet_info.text, user_info.name, user_info.followers_count, user_info.time_zone};
```

Output


```
    "lang": "en",
    "text": "@featheraee only with onew gave them http://t.co/7OrwEq6a for their radio show! ^^",
    "name": "ㅋ ",
    "followers_count": 878,
    "time_zone": "Pacific Time (US & Canada)"
  },
  {
    "id": 289429415530754048,
    "lang": "vi",
    "text": ":p RT @confessormissy: (ᴙº´º)ᴙ ode\"Nixonsamaju: @confessormissy http://t.co/trvAwM6Y\"",
    "name": "TobeChukwu",
    "followers_count": 506,
    "time_zone": null
  },
  {
    "id": 289429419720851457,
    "lang": "en",
    "text": "@Victorriaaaaa ash but you will once you\'re out I guess? We\'d need some other people too th
gh! :/ x",
    "name": "Jemma Martins",
    "followers_count": 184,
    "time_zone": null
  }
```

Summary

Tool	When to use?	Workload
Hive	Ad-hoc querying/analysis using SQL-like syntax	Batch
Pig	Ad-hoc querying/analysis when data is inherently nested	Batch
HBase	Fast reads and writes (e.g., metadata)	Interactive
Jaql	Ad-hoc querying/analysis on semistructured/JSON documents	Batch