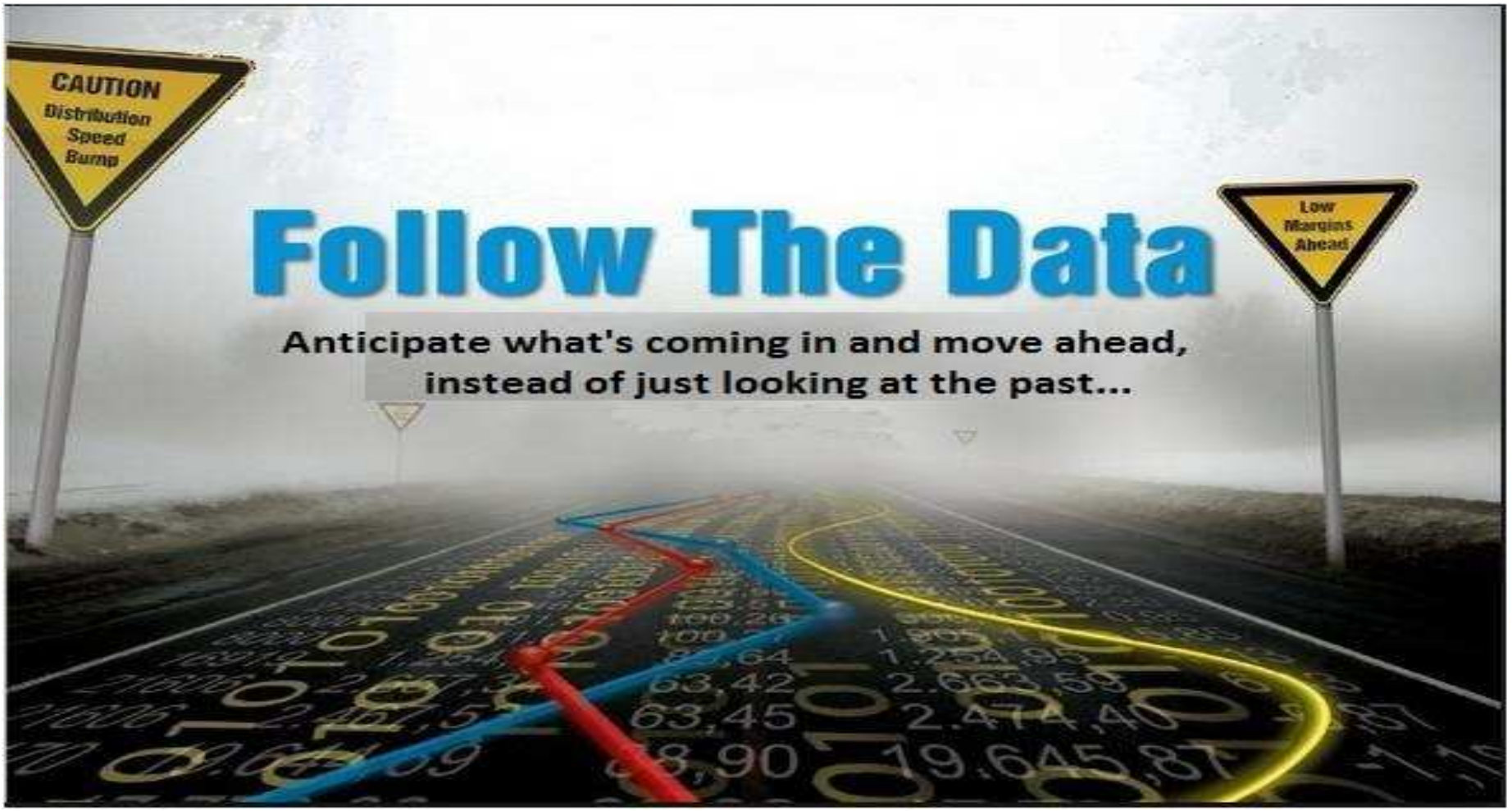


DEMYSTIFYING DATABASES: “SQL -> NoSQL -> NewSQL”

KHAJA SHAIK

PLANON

Early Y2K…

- **Early 2000's:**
  - Huge internet traffic – Google, Yahoo, Amazon etc.
  - Data storage requirements were out grown.
    a. Capacity: Ever increasing data
    b. Variety: Structured, Semi-structured (Log files) and Unstructured data (A,V,I,Text).
  - Research began to find out alternative data stores.

- **Vertical scaling/ Scale-up – Not preferabl**

  a. Add more 'power' (Servers, RAM, Faster disk).
  b. Multi-core i.e. Spread load b/w CPU & RAM.
  c. Processor bound: Scaling only to certain limit.
  d. Involves huge costs.

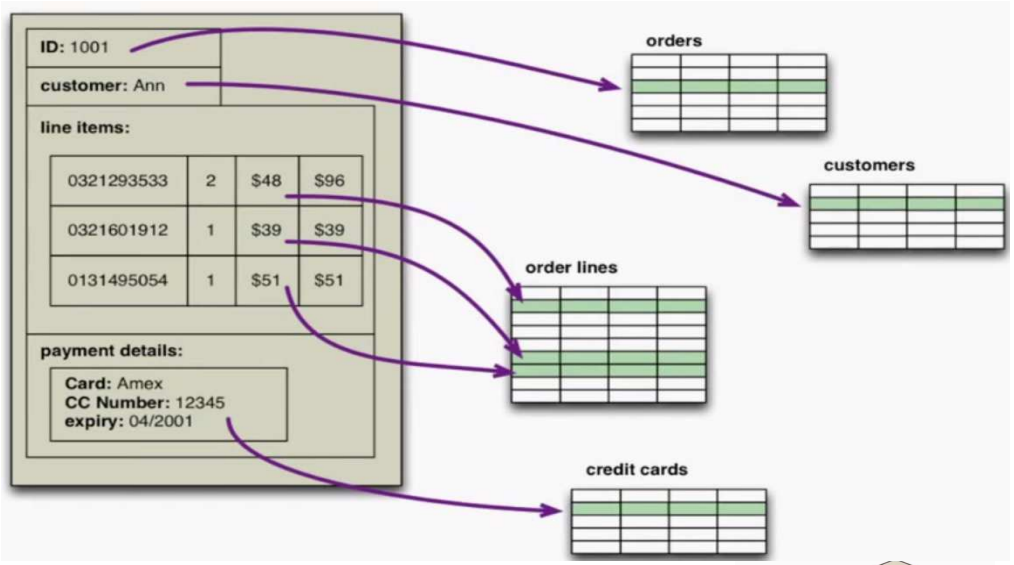
- **Horizontal scaling/ Scale-out – Preferred**.

  a. Add more machines (Cheap commodity h/w).
  b. Shrading/ Portioning across nodes.
  c. Distributed Parallel: Easier to scale dynamically.
  d. Relatively less costly.

PLANON

Relational databases…

- Persistence.
- Concurrency for transactions: ACID.
- SQL de-facto language.
- Reporting.
- Integration.



- Impedance mismatch.

Limitations:

- Does not work well in a distributed environment.
- Not designed to handle:
  a. Semi-structured data: Server log records.
  b. Unstructured data: Audio, Video, Images, Text.
- Not suited for analysis i.e. OLAP.
- Too many disk seeks during read-operations.

Era of NOSQL (Not only SQL) !!

- **NOSQL name coined ?..**
  - ‘Twitter tag’ for a meeting on database trends.

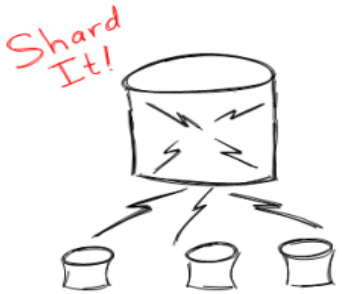
- **Characteristics of NOSQL:**
  - Schema-less, Non-relational.
  - Open-source.
  - Cluster-friendly.

- **2003-04: Google (GFS & MapReduce papers)**
  - Google File System: Data storage.
  - MapReduce: Processing data.

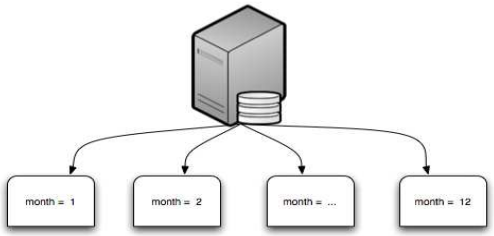
- **2006: Google (BigTable).**
  - High performance storage system built on GFS.

- **2007: Amazon (Dynamo).**
  - Key-value structured distributed storage system.

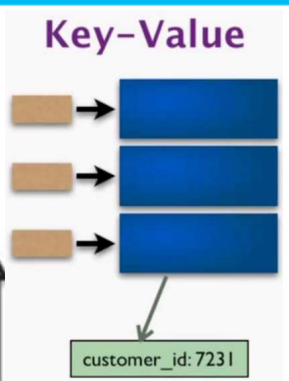
- **Data Distribution:**
  - **Sharding**:
    a. One copy of data split across different nodes.
    b. Each piece of data resides on one node.



  - **Partitioning:**
    a. Replicate data, same piece of data on multiple nodes.
    b. Performance: More nodes to handle same request.
    c. Resilience: One node goes down, replicas available.

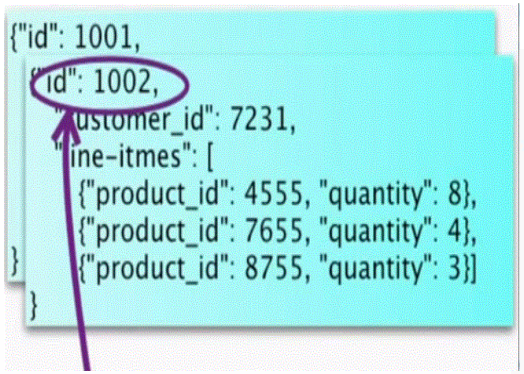


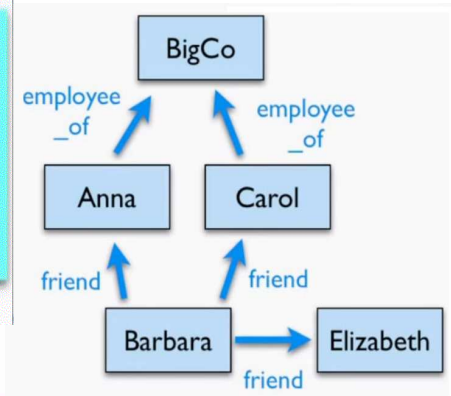
NOSQL Umbrella…



PERSON TABLE

row key	personal_data		demographic	
PersonID	Name	Address	BirthDate	Gender
1	H. Houdini	Budapest, Hungary	1926-10-31	M
2	D. Copper	New Jersey, USA	1956-09-16	M
3	Merlin	Stonehenge, England	1136-12-03	F
...	...	...	...	...
500,000,000	F. Cadillac	Nevada, USA	1964-01-07	M





In-Memory NoSQL DB

AEROSPIKE

GridGain
REAL TIME BIG DATA

STARCOUNTER

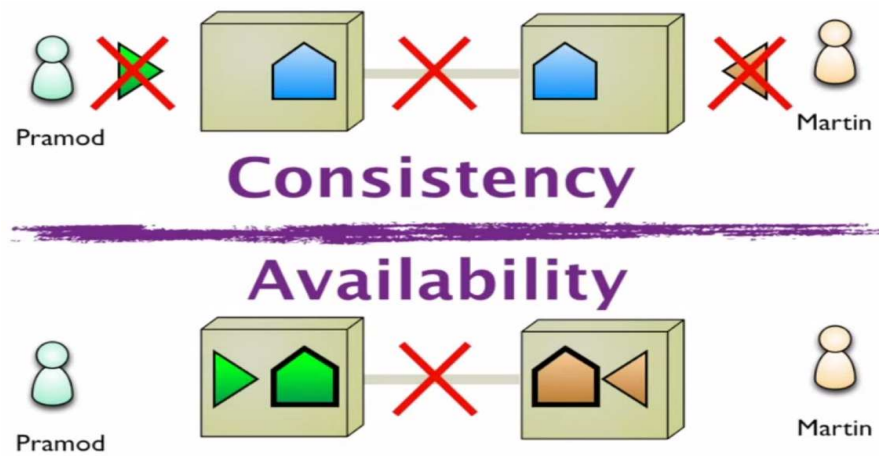
CAP Theorem…

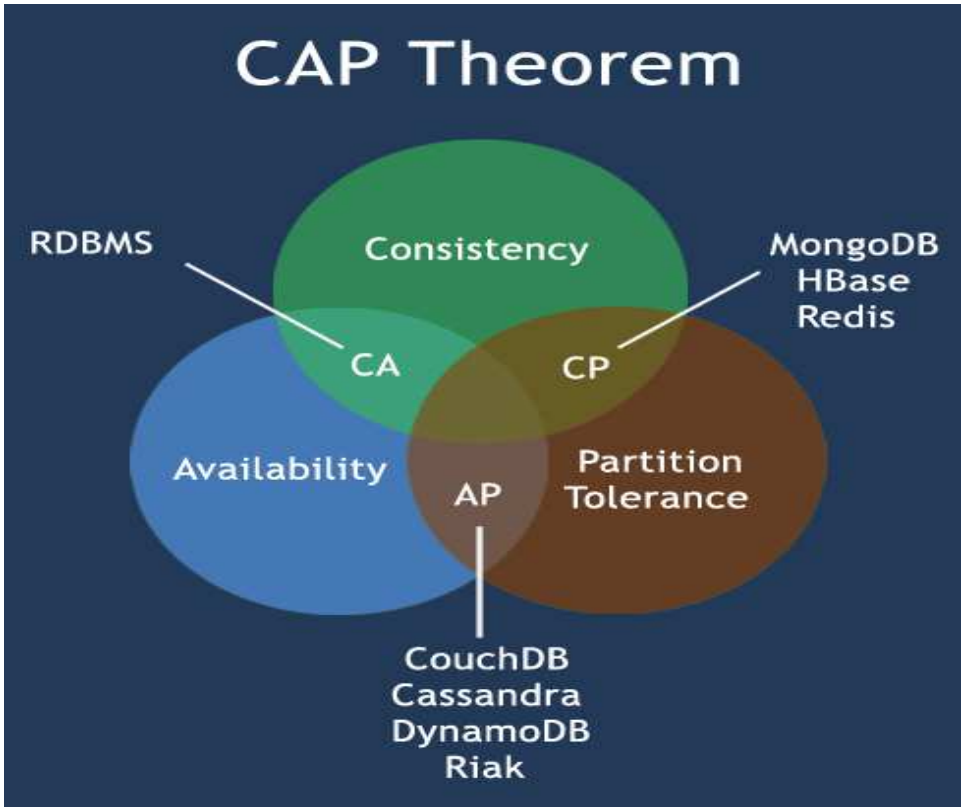
Impossible for a distributed computer system to achieve all 3 guarantees.

- **Consistency**: All nodes see same data at the same time.
- **Availability**: Node failures do not prevent survivors from continuing to operate.
- **Partition tolerance**: System continues to operate despite arbitrary message loss.

- **Online ticket-booking/ Shopping application.**

Consistency: Don't continue when network goes down.
Availability: Keep going at the risk of introducing an inconsistent behaviour.

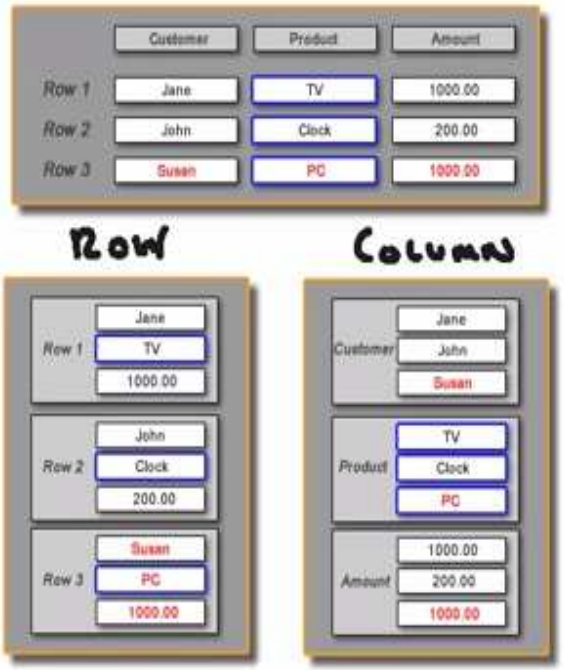




PLANON

Column-oriented databases…

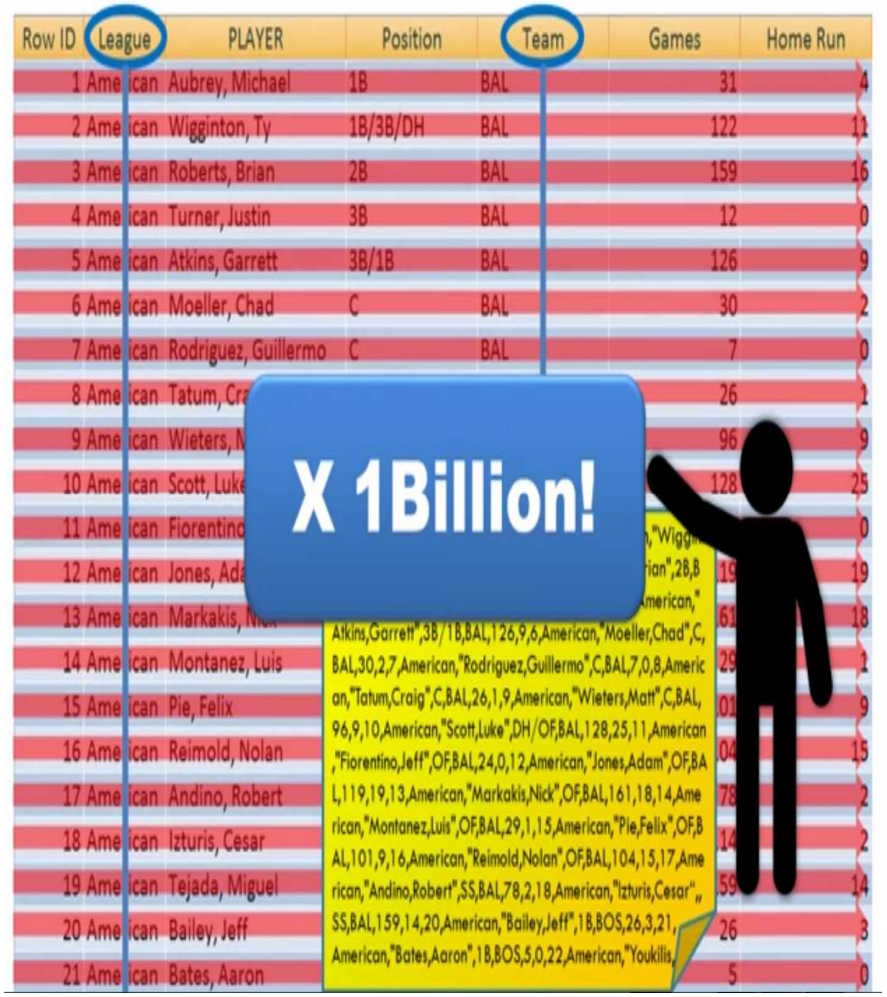
ROW-oriented database	Column-oriented database
RDMS assigns unique key to each row and stores entire row in a single block on disk.	Column values are stored together in a single block on disk.
OLTP (Efficient for single read/write operations).	OLAP (Aggregation/ Analysis over many rows or columns).
ACID (Atomicity, Consistency, Isolation, Durability).	BASE (Basically Available, Soft state, Eventually consistent).
Schema. Normalized (No duplication).	Schema-less. De-normalized.
Structured data.	Semi/Unstructured data.
Pre-build composite indexes on different sets of columns for improving efficiency.	High degree of compression (Stores multiple occurrences only once e.g. ProductID).
Thin (Few Rows & Columns).	Wide (Too many Rows & Cols).
Scale horizontally.	Scale vertically.
No support for SQL.	ANSI-92 SQL specification.
Eg: MySQL, Oracle, SQL Server.	Eg: HBase, Cassandra, BigTable.



PLANON

Row oriented vs. Column oriented…

Select all American league teams with a particular team name like ‘BAL’.



PLANON

Indexes won't help: Storage, Lookup…

Indexes

Key	RowID
1	0001B008D23A671A
2	0001B008D23A671B
3	0001B008D23A671C
4	0001B008D23A671D
5	0001B008D23A671E

WHERE key=4

Elmer Fudd calls customer service

Phone	RowID
(207) 882-7323	0001B008D23A671D
(209) 375-6572	0001B008D23A671B
(212) 227-1810	0001B008D23A671C
(718) 938-3235	0001B008D23A671A
(978) 744-0991	0001B008D23A671E

WHERE phone='(207) 882-7323'

Indexes on high-cardinality columns make accessing a single row very fast

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F

but don't help on analytical queries scanning many rows

e.g.

What's the average age of males?

Row-oriented: Usually requires rebuilding table

Key	Fname	Lname	State	Zip	Phone	Age	Sex	Golf
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M	Y
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M	N
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M	Y
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M	Y
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F	N

Addition of column shifts every row

Column-oriented: Just create another file

Key
1
2
3
4
5

Fname
Bugs
Yosemite
Daffy
Elmer
Witch

Lname
Bunny
Sam
Duck
Fudd
Hazel

State
NY
CA
NY
ME
MA

Zip
11217
95389
10013
04578
01970

Phone
(718) 938-3235
(209) 375-6572
(212) 227-1810
(207) 882-7323
(978) 744-0991

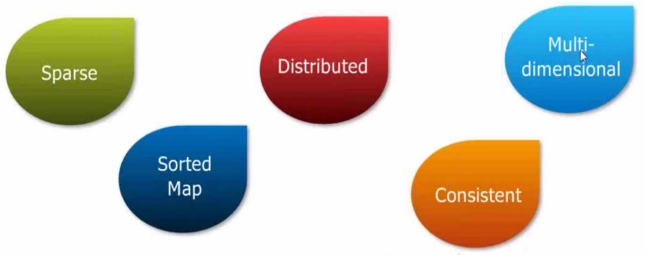
Age
34
52
35
43
57

Sex
M
M
M
M
F

Golf
Y
N
Y
Y
N

HBase: Column-oriented database…

- Developed part of Apache Hadoop project (based on Google's BigTable paper).
- Written in Java with support for Millions of columns * Billions of Rows.
- Distributed, Scalable, Column-oriented data store on top of HDFS (Hadoop) .
- Everything is stored in the form of byte-array.
- Column family: Columns part of a family stored together in one file.
- Each Record has a time-stamp.



Purpose:

- Real-time, fast-random access/updates.
- Stores structured data.
- Schema flexible – Add columns.
- Auto-sharding / partitioning of data.
- Fault tolerant - Replication.

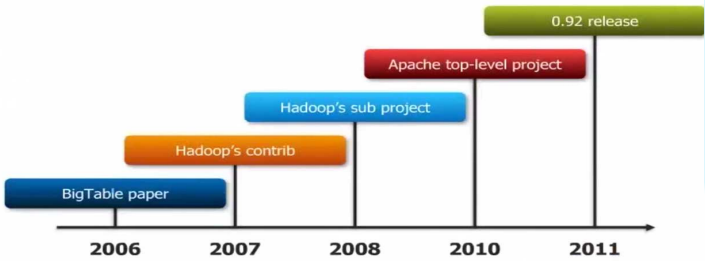
Queries:

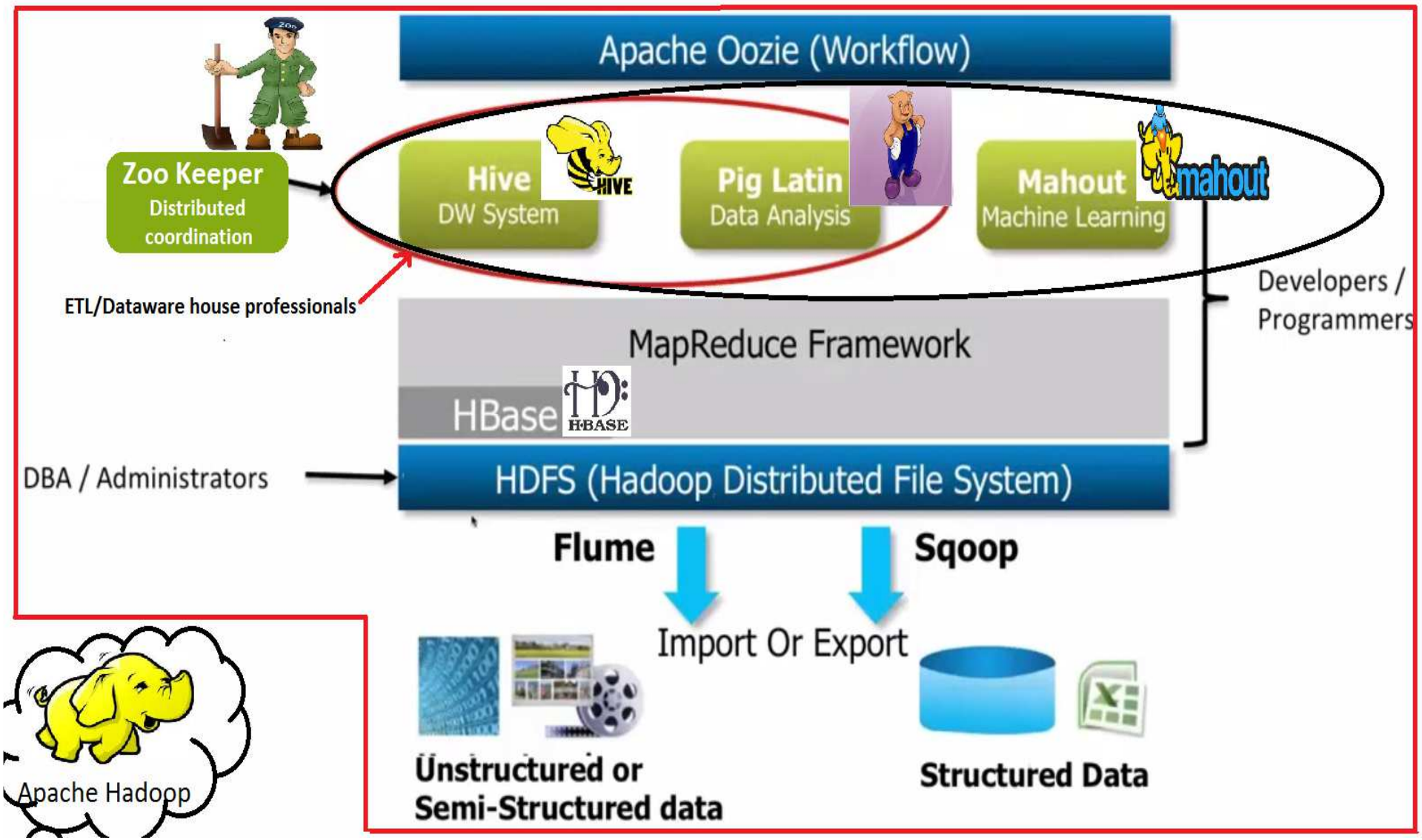
- Average sales for a product.
- Number of customer residing in LA.
- Product with amount > $1500.

Row Key	Customer		Sales	
Customer Id	Name	City	Product	Amount
101	John White	Los Angeles, CA	Chairs	$400.00
102	Jane Brown	Atlanta, GA	Lamps	$200.00
103	Bill Green	Pittsburgh, PA	Desk	$500.00
104	Jack Black	St. Louis, MO	Bed	$1600.00

Column Families

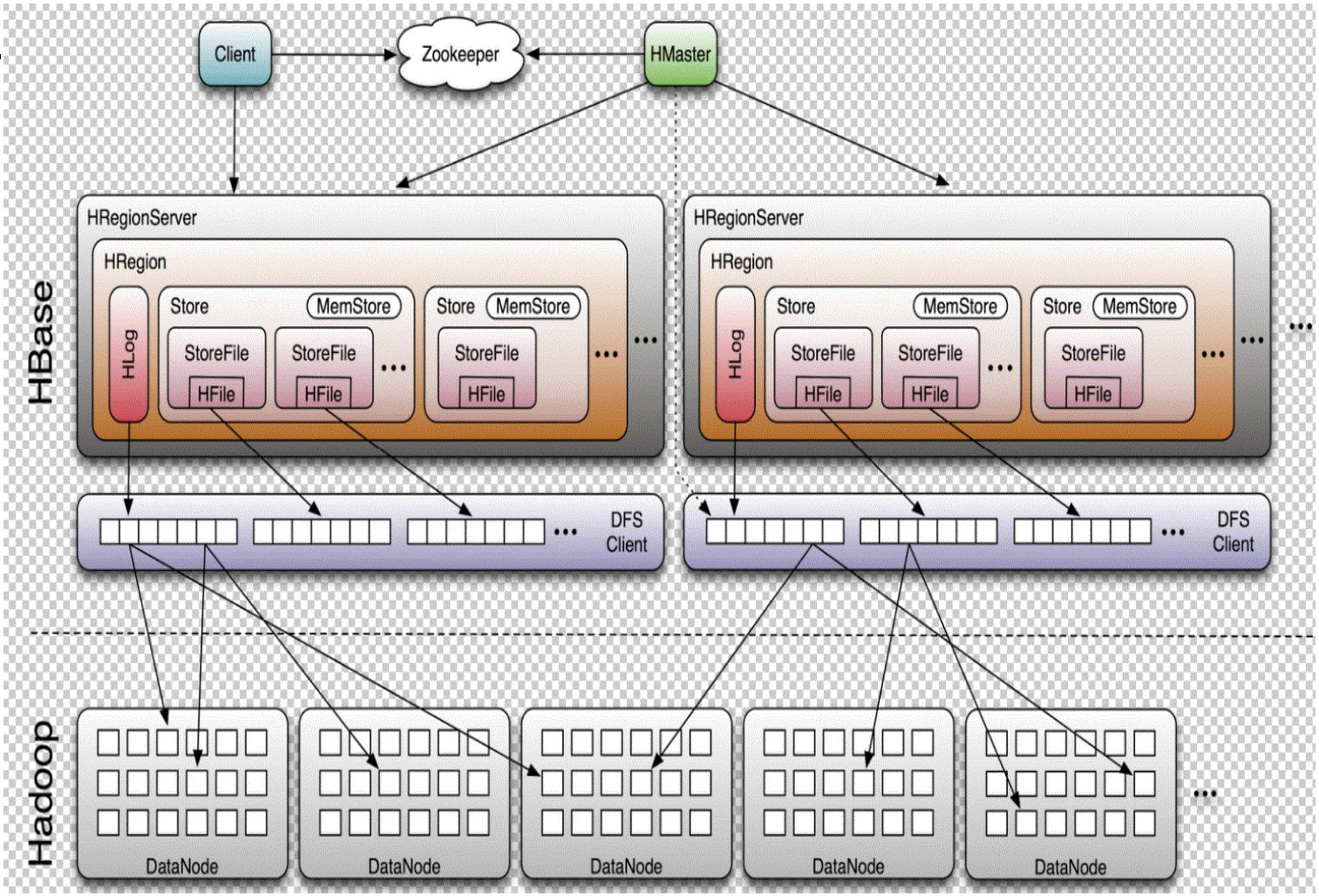
Hadoop framework…





HBase Architecture…

- Region Server.
- Block cache.
- MemStore.
- Regions.
- Meta.
- Read/Write.
- HStore.
- HFile.
- HLog
- HMaster.
- DataNode.
- DFS Client.
- Store.



PLANON

Utilities…

Interfaces:

- Shell.
- REST.
- Thrift API.
- JVM Client.
- MapReduce.
- Avro.

Import/Export: Sqoop.

PLANON

HBase Setup…

- Download HBase from Apache.

  http://archive.apache.org/dist/hbase/hbase-0.94.0/

- Extract the folder into below location. As a pre-requisite, Hadoop must have been installed by now, else download it from Apache.

➢ **C:\cygwin\usr\local\hadoop-0.20\hbase-0.94.0**

- Launch Cygwin

➢ **cd /usr/local/hadoop**

➢ **Start DFS Node: bin/start-dfs.sh**

➢ **Start MapRed Node: bin/map-red.sh**

- Launch HBase

➢ **cd hbase-0.94.0/bin**

➢ **start-hbase.sh**

➢ **hbase shell**

PLANON

Setup HBase…

```
/usr/local/hadoop/hbase-0.94.0/bin
Your group is currently "mkgroup".  This indicates that neither
your gid nor your pgsid (primary group associated with your SID)
is in /etc/group.

The /etc/group (and possibly /etc/passwd) files should be rebuilt.
See the man pages for mkpasswd and mkgroup then, for example, run

mkpasswd -l [-d] > /etc/passwd
mkgroup  -l [-d] > /etc/group

Note that the -d switch is necessary for domain users.

kashai@PC08022 ~
$ cd /usr/local/hadoop

kashai@PC08022 /usr/local/hadoop
$ cd hbase-0.94.0/bin

kashai@PC08022 /usr/local/hadoop/hbase-0.94.0/bin
$ start-hbase.sh
localhost: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
localhost: @    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
localhost: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
localhost: IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
localhost: Someone could be eavesdropping on you right now (man-in-the-middle at
tack)!
localhost: It is also possible that a host key has just been changed.
localhost: The fingerprint for the ECDSA key sent by the remote host is
localhost: 79:a7:47:03:0c:a7:0c:37:44:49:62:98:a5:88:6d:12.
localhost: Please contact your system administrator.
localhost: Add correct host key in /home/kashai/.ssh/known_hosts to get rid of t
his message.
localhost: Offending ECDSA key in /home/kashai/.ssh/known_hosts:1
localhost: ECDSA host key for localhost has changed and you have requested stric
t checking.
localhost: Host key verification failed.
starting master, logging to /usr/local/hadoop/hbase-0.94.0/bin/../logs/hbase-PC0
```

Commands…

- **Create table (Table name, Column family name).**

  Create ‘stocks’, ‘perf’

- **Describe table.**

  Describe ‘stocks’.

```
create 'stocks', 'perf'
create 'stocks', 'perf'
0 row(s) in 1.4070 seconds
```

```
describe 'stocks'
describe 'stocks'
DESCRIPTION  ENABLED
 {NAME => 'stocks', FAMILIES => [{NAME => 'perf', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE', REPLICATION_SC
OPE => '0', VERSIONS => '3', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => '2147483647', KEEP_DELETED_CELLS => 'f
alse', BLOCKSIZE => '65536', IN_MEMORY => 'false', ENCODE_ON_DISK => 'true', BLOCKCACHE => 'true'}]} true
1 row(s) in 0.0260 seconds
```

Commands…

- **Put data into table ‘stocks’.**

  Put ‘tablename’, ‘rowkey’, ‘column family name: Column name1’, ‘value1’.
  Put ‘tablename’, ‘rowkey’, ‘column family name: Column name2’, ‘value2’.

```
/usr/local/hadoop/hbase-0.94.0/bin
put 'stocks', 'GOOGLE_2', 'perf:High', '66.3'
put 'stocks', 'GOOGLE_2', 'perf:High', '66.3'
0 row(s) in 0.0060 seconds

put 'stocks', 'GOOGLE_2', 'perf:Low', '62.7'
put 'stocks', 'GOOGLE_2', 'perf:Low', '62.7'
0 row(s) in 0.0050 seconds

put 'stocks', 'YAHOO_1', 'perf:Symbol', 'YAHOO'
put 'stocks', 'YAHOO_1', 'perf:Symbol', 'YAHOO'
0 row(s) in 0.0070 seconds

put 'stocks', 'YAHOO_1', 'perf:Date', '1/21/2015'
put 'stocks', 'YAHOO_1', 'perf:Date', '1/21/2015'
0 row(s) in 0.0080 seconds

put 'stocks', 'YAHOO_1', 'perf:Close', '551.01'
put 'stocks', 'YAHOO_1', 'perf:Close', '551.01'
0 row(s) in 0.0050 seconds

put 'stocks', 'YAHOO_1', 'perf:Volume', '4345677'
put 'stocks', 'YAHOO_1', 'perf:Volume', '4345677'
0 row(s) in 0.0040 seconds

put 'stocks', 'YAHOO_1', 'perf:Open', '547.89'
put 'stocks', 'YAHOO_1', 'perf:Open', '547.89'
0 row(s) in 0.0050 seconds

put 'stocks', 'YAHOO_1', 'perf:High', '556.88'
put 'stocks', 'YAHOO_1', 'perf:High', '556.88'
0 row(s) in 0.0050 seconds

put 'stocks', 'YAHOO_1', 'perf:Low', '550.05'
put 'stocks', 'YAHOO_1', 'perf:Low', '550.05'
0 row(s) in 0.0040 seconds
```

Commands…

- Retrieve data from table ‘stocks’.

Get ‘table name’, ‘row key’, ‘column family name’.

```
get 'stocks', 'GOOGLE_1', 'perf'
get 'stocks', 'GOOGLE_1', 'perf'
COLUMN  CELL
 perf:Close timestamp=1421899912891, value=89.54
 perf:Date timestamp=1421899903099, value=1/21/2015
 perf:High timestamp=1421899940222, value=67.8
 perf:Low timestamp=1421899956527, value=64.5
 perf:Open timestamp=1421899932237, value=61.5
 perf:Symbol timestamp=1421899870138, value=GOOGLE
 perf:Volume timestamp=1421899922910, value=67890123
7 row(s) in 0.0250 seconds

get 'stocks', 'YAHOO_2', 'perf'
get 'stocks', 'YAHOO_2', 'perf'
COLUMN  CELL
 perf:Close timestamp=1421900393284, value=505.06
 perf:Date timestamp=1421900386600, value=1/22/2015
 perf:High timestamp=1421900414779, value=560.02.3
 perf:Low timestamp=1421900422554, value=502.99
 perf:Open timestamp=1421900408127, value=556.67
 perf:Symbol timestamp=1421900376788, value=YAHOO
 perf:Volume timestamp=1421900400164, value=4567890
7 row(s) in 0.0070 seconds
```


PLANON

Question 1…

What is the oldest volume for a given stock?

scan 'stocks',
{COLUMNS=>['perf:Symbol','perf:Date','perf:Volume'],LIMIT=>1,STARTROW=>'
GOOGLE_1'}

```
get 'stocks', 'YAHOO_2', ['perf:Date', 'perf:Close', 'perf:Volume', 'perf:Open', 'perf:High', 'perf:Low']
get 'stocks', 'YAHOO_2', ['perf:Date', 'perf:Close', 'perf:Volume', 'perf:Open', 'perf:High', 'perf:Low']
COLUMN  CELL
 perf:Close timestamp=1421900393284, value=505.06
 perf:Date timestamp=1421900386600, value=1/22/2015
 perf:High timestamp=1421900414779, value=560.02.3
 perf:Low timestamp=1421900422554, value=502.99
 perf:Open timestamp=1421900408127, value=556.67
 perf:Volume timestamp=1421900400164, value=4567890
6 row(s) in 0.0080 seconds

get 'stocks', 'GOOGLE_1', ['perf:Symbol', 'perf:High', 'perf:Low']
get 'stocks', 'GOOGLE_1', ['perf:Symbol', 'perf:High', 'perf:Low']
COLUMN  CELL
 perf:High timestamp=1421899940222, value=67.8
 perf:Low timestamp=1421899956527, value=64.5
 perf:Symbol timestamp=1421899870138, value=GOOGLE
3 row(s) in 0.0050 seconds
```

Question 2…

What is the stock performance in the past two days for Google?

scan 'stocks', {LIMIT=>2,STARTROW=>'GOOGLE_1'}

```
scan 'stocks', {LIMIT=>2,STARTROW=>'GOOGLE_2'}
scan 'stocks', {LIMIT=>2,STARTROW=>'GOOGLE_2'}
ROW   COLUMN+CELL
 GOOGLE_2 column=perf:Close, timestamp=1421900082978, value=88.76
 GOOGLE_2 column=perf:Date, timestamp=1421900074798, value=1/22/2015
 GOOGLE_2 column=perf:High, timestamp=1421900107384, value=66.3
 GOOGLE_2 column=perf:Low, timestamp=1421900114859, value=62.7
 GOOGLE_2 column=perf:Open, timestamp=1421900099150, value=60.4
 GOOGLE_2 column=perf:Symbol, timestamp=1421900066224, value=GOOGLE
 GOOGLE_2 column=perf:Volume, timestamp=1421900090411, value=66890771
 YAHOO_1 column=perf:Close, timestamp=1421900298734, value=551.01
 YAHOO_1 column=perf:Date, timestamp=1421900291221, value=1/21/2015
 YAHOO_1 column=perf:High, timestamp=1421900320911, value=556.88
 YAHOO_1 column=perf:Low, timestamp=1421900328158, value=550.05
 YAHOO_1 column=perf:Open, timestamp=1421900313302, value=547.89
 YAHOO_1 column=perf:Symbol, timestamp=1421900282846, value=YAHOO
 YAHOO_1 column=perf:Volume, timestamp=1421900306058, value=4345677
2 row(s) in 0.0360 seconds

scan 'stocks', {LIMIT=>2,STARTROW=>'GOOGLE_1'}
scan 'stocks', {LIMIT=>2,STARTROW=>'GOOGLE_1'}
ROW   COLUMN+CELL
 GOOGLE_1 column=perf:Close, timestamp=1421899912891, value=89.54
 GOOGLE_1 column=perf:Date, timestamp=1421899903099, value=1/21/2015
 GOOGLE_1 column=perf:High, timestamp=1421899940222, value=67.8
 GOOGLE_1 column=perf:Low, timestamp=1421899956527, value=64.5
 GOOGLE_1 column=perf:Open, timestamp=1421899932237, value=61.5
 GOOGLE_1 column=perf:Symbol, timestamp=1421899870138, value=GOOGLE
 GOOGLE_1 column=perf:Volume, timestamp=1421899922910, value=67890123
 GOOGLE_2 column=perf:Close, timestamp=1421900082978, value=88.76
 GOOGLE_2 column=perf:Date, timestamp=1421900074798, value=1/22/2015
 GOOGLE_2 column=perf:High, timestamp=1421900107384, value=66.3
 GOOGLE_2 column=perf:Low, timestamp=1421900114859, value=62.7
 GOOGLE_2 column=perf:Open, timestamp=1421900099150, value=60.4
 GOOGLE_2 column=perf:Symbol, timestamp=1421900066224, value=GOOGLE
 GOOGLE_2 column=perf:Volume, timestamp=1421900090411, value=66890771
2 row(s) in 0.0320 seconds
```

Question 3…

What is the latest high/low by stock?

scan 'stocks',
{COLUMNS=>['perf:Symbol','perf:Date','perf:High','perf:Low'],LIMIT=>1,START ROW=>'GOOGLE_2'}

```
scan 'stocks', {COLUMNS=>['perf:Symbol','perf:Date','perf:Volume'],LIMIT=>1,STARTROW=>'GOOGLE_1'}
scan 'stocks', {COLUMNS=>['perf:Symbol','perf:Date','perf:Volume'],LIMIT=>1,STARTROW=>'GOOGLE_1'}
ROW  COLUMN+CELL
 GOOGLE_1 column=perf:Date, timestamp=1421899903099, value=1/21/2015
 GOOGLE_1 column=perf:Symbol, timestamp=1421899870138, value=GOOGLE
 GOOGLE_1 column=perf:Volume, timestamp=1421899922910, value=67890123
1 row(s) in 0.0330 seconds
```


PLANON

Question 4…

What is the performance of stock on '1/22/2015'.

scan 'stocks', {COLUMNS=>['perf'],FILTER=>"(SingleColumnValueFilter('perf','Date',=,'regexstring:1/22/2015',false,false))"}

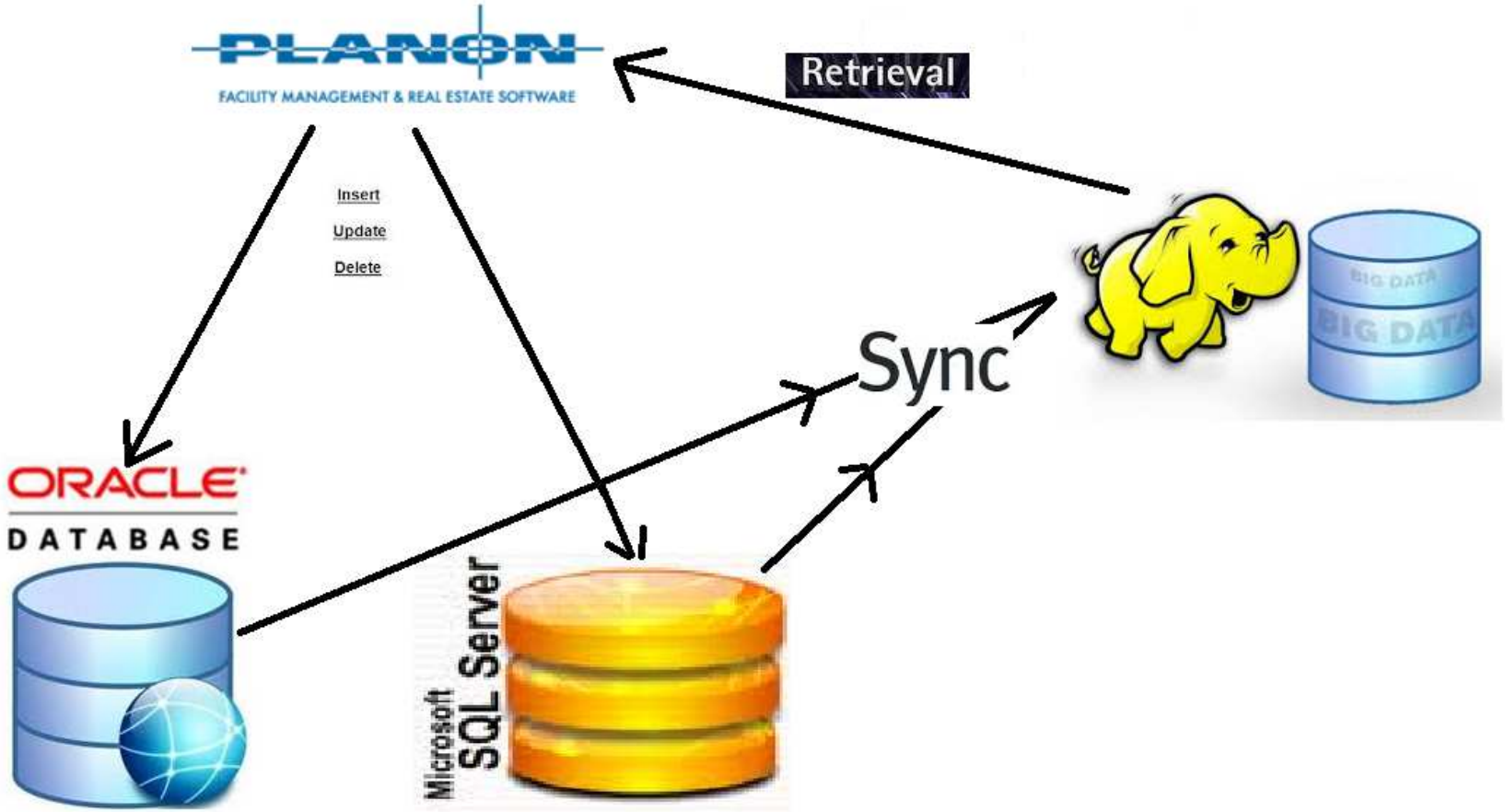
```
scan 'stocks', {COLUMNS=>['perf'],FILTER=>"(SingleColumnValueFilter('perf','Date',=,'regexstring:1/22/2015',false,fals
e))"}

scan 'stocks', {COLUMNS=>['perf'],FILTER=>"(SingleColumnValueFilter('perf','Date',=,'regexstring:1/22/2015',false,fals
e))"}
ROW  COLUMN+CELL
 GOOGLE_2 column=perf:Close, timestamp=1421900082978, value=88.76
 GOOGLE_2 column=perf:Date, timestamp=1421900074798, value=1/22/2015
 GOOGLE_2 column=perf:High, timestamp=1421900107384, value=66.3
 GOOGLE_2 column=perf:Low, timestamp=1421900114859, value=62.7
 GOOGLE_2 column=perf:Open, timestamp=1421900099150, value=60.4
 GOOGLE_2 column=perf:Symbol, timestamp=1421900066224, value=GOOGLE
 GOOGLE_2 column=perf:Volume, timestamp=1421900090411, value=66890771
 YAHOO_2 column=perf:Close, timestamp=1421900393284, value=505.06
 YAHOO_2 column=perf:Date, timestamp=1421900386600, value=1/22/2015
 YAHOO_2 column=perf:High, timestamp=1421900414779, value=560.02.3
 YAHOO_2 column=perf:Low, timestamp=1421900422554, value=502.99
 YAHOO_2 column=perf:Open, timestamp=1421900408127, value=556.67
 YAHOO_2 column=perf:Symbol, timestamp=1421900376788, value=YAHOO
 YAHOO_2 column=perf:Volume, timestamp=1421900400164, value=4567890
2 row(s) in 0.0510 seconds
```


PLANON

What's in it for Planon…

Analytics/ Reporting:

a. Sqoop/ Map Reduce program to export data from MSSQL/ Oracle database.
b. Use HBase to process the data on HDFS.
c. Import the processed results back into the database/ data ware house from Hadoop for analytics/ reporting.



New SQL…

Modern RDBMSs provide:

- Same scalable performance of NoSQL systems for OLTP read-write workloads.
- Maintain ACID guarantees of a traditional database system.

References…

PDFs (Research papers):

- Google File system (Research paper): http://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf
- Google BigTable (Research paper): http://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf
- Amazon (Dynamo research paper): http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf
- HBase IN ACTION: http://dl.e-book-free.com/2013/07/hbase_in_action.pdf

Wiki Links:

- Column-oriented DBMS: http://en.wikipedia.org/wiki/Column-oriented_DBMS
- NOSQL, NewSQL, RDBMS: http://www.dbms2.com/2014/03/28/nosql-vs-newsql-vs-traditional-rdbms/
- In-memory databases: http://www.opensourceforu.com/2012/01/importance-of-in-memory-databases/

Videos:

- NOSQL Databases: http://nosql-database.org/
- HBase Schema Design: https://www.youtube.com/watch?v=_HLoH_PgrLk
- Column oriented databases: https://www.youtube.com/watch?v=mRvkikVuojU
- Row vs. Column vs. NOSQL: https://www.youtube.com/watch?v=ja7qkg8-GYw