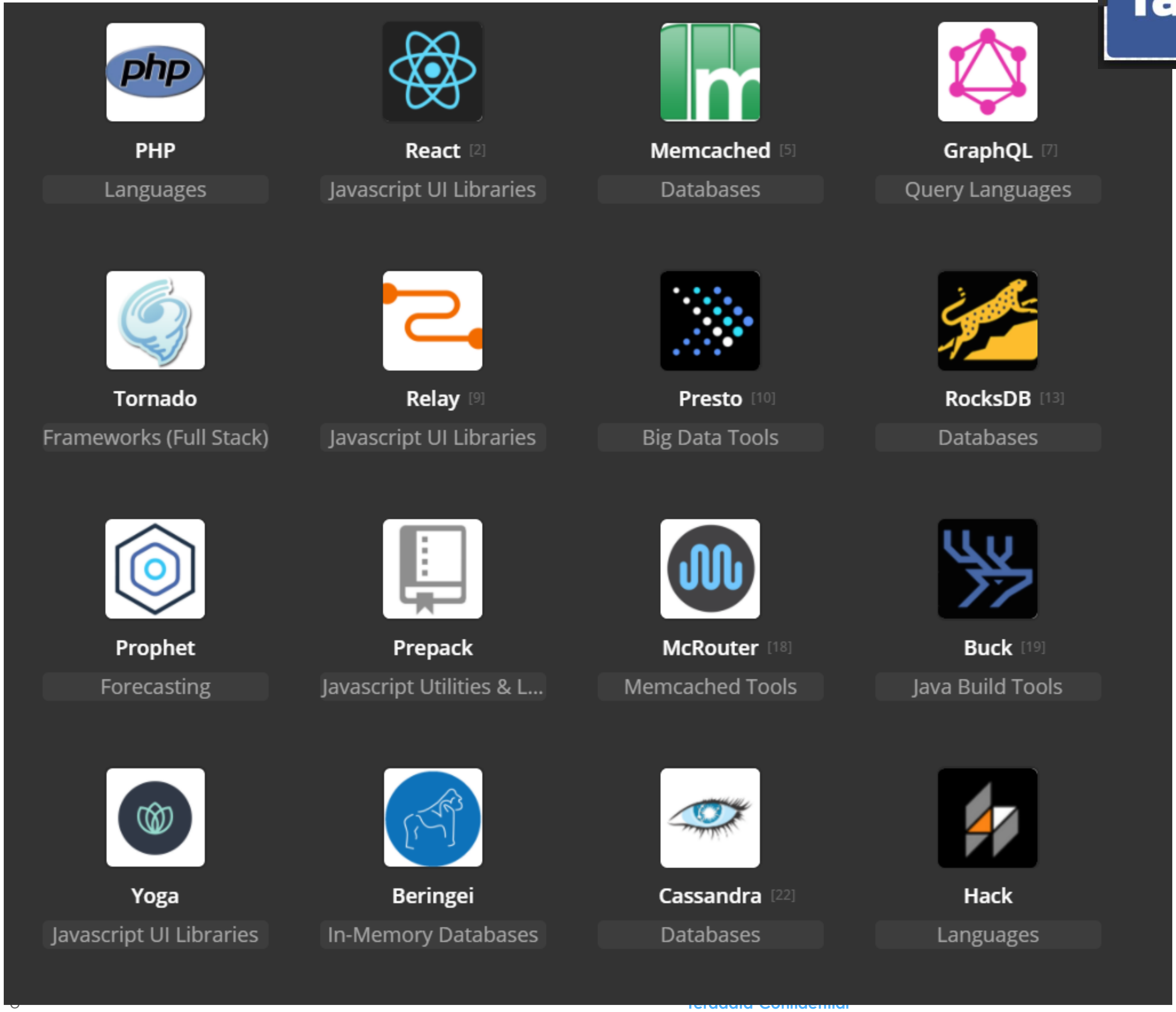
Big Data Basics

Capability Team

THE DATA STORY

- A Day In The Life: -
  - > 500TB
  - 2.7 billion likes
  - 300 Million Photos Uploaded
  - 70,000 Queries executed
  - 105 TB Data scanned by HIVE (30 minutes)

THE DATA STORY



PHP
Languages

React [2]
Javascript UI Libraries

Memcached [5]
Databases

GraphQL [7]
Query Languages

Tornado
Frameworks (Full Stack)

Relay [9]
Javascript UI Libraries

Presto [10]
Big Data Tools

RocksDB [13]
Databases

Prophet
Forecasting

Prepack
Javascript Utilities & L...

McRouter [18]
Memcached Tools

Buck [19]
Java Build Tools

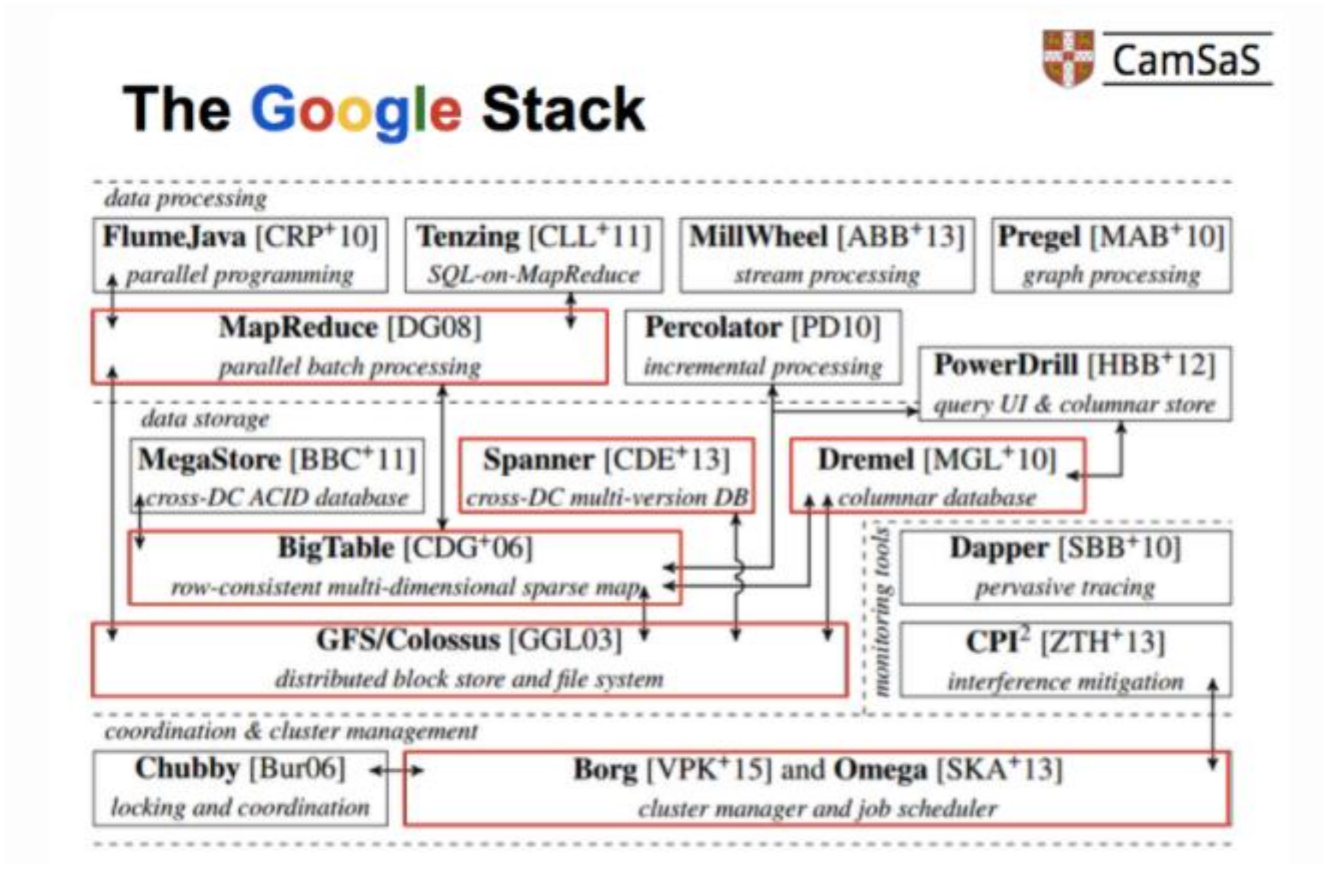
Yoga
Javascript UI Libraries

Beringei
In-Memory Databases

Cassandra [22]
Databases

Hack
Languages

GOOGLE



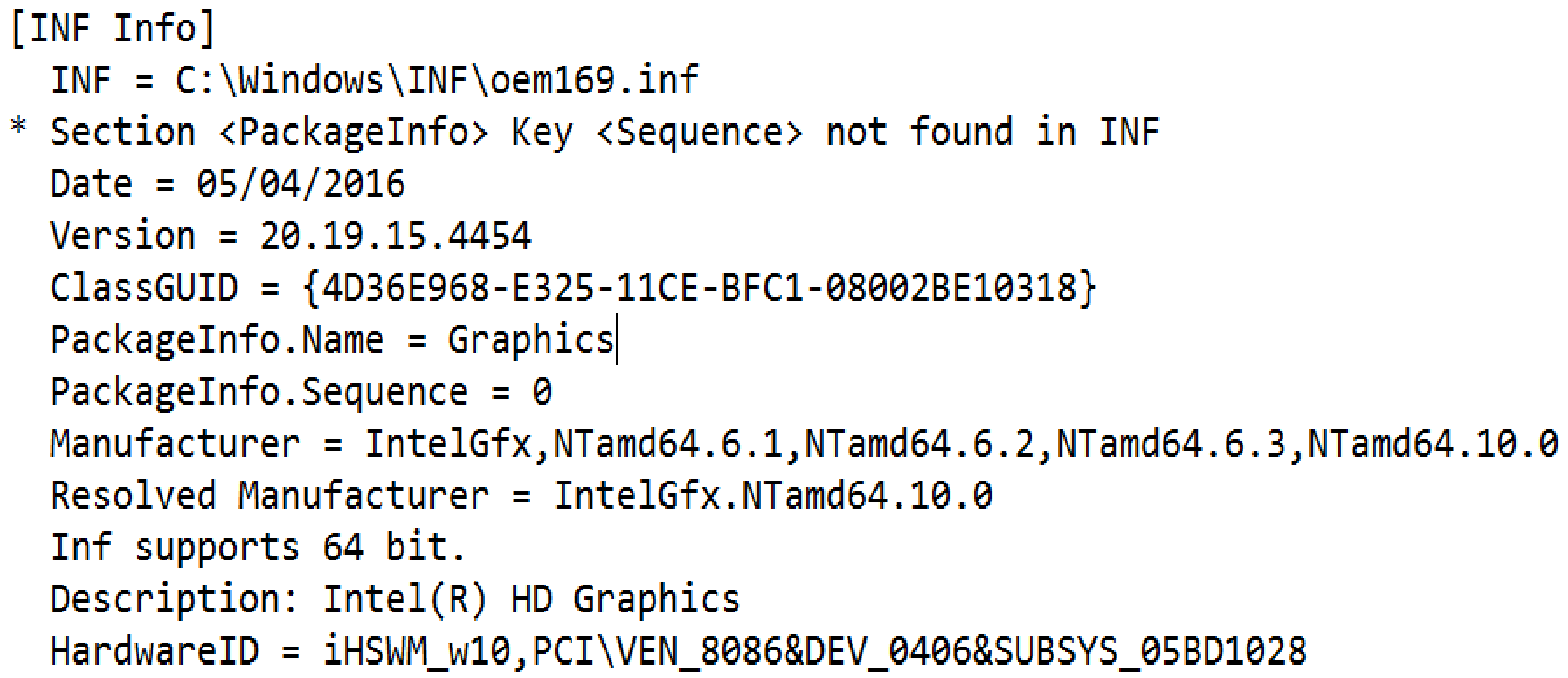
CamSaS
The Google Stack
data processing
FlumeJava [CRP+10] parallel programming
Tenzing [CLL+11] SQL-on-MapReduce
MillWheel [ABB+13] stream processing
Pregel [MAB+10] graph processing
MapReduce [DG08] parallel batch processing
Percolator [PD10] incremental processing
PowerDrill [HBB+12] query UI & columnar store
data storage
MegaStore [BBC+11] cross-DC ACID database
Spanner [CDE+13] cross-DC multi-version DB
Dremel [MGL+10] columnar database
BigTable [CDG+06] row-consistent multi-dimensional sparse map
GFS/Colossus [GGL03] distributed block store and file system
monitoring tools
Dapper [SBB+10] pervasive tracing
CPI2 [ZTH+13] interference mitigation
coordination & cluster management
Chubby [Bur06] locking and coordination
Borg [VPK+15] and Omega [SKA+13] cluster manager and job scheduler

- Core Components
  - Big Table
  - Dremmel
  - Map Reduce
  - GFS

UNSTRUCTURED / STRUCTURED DATA??

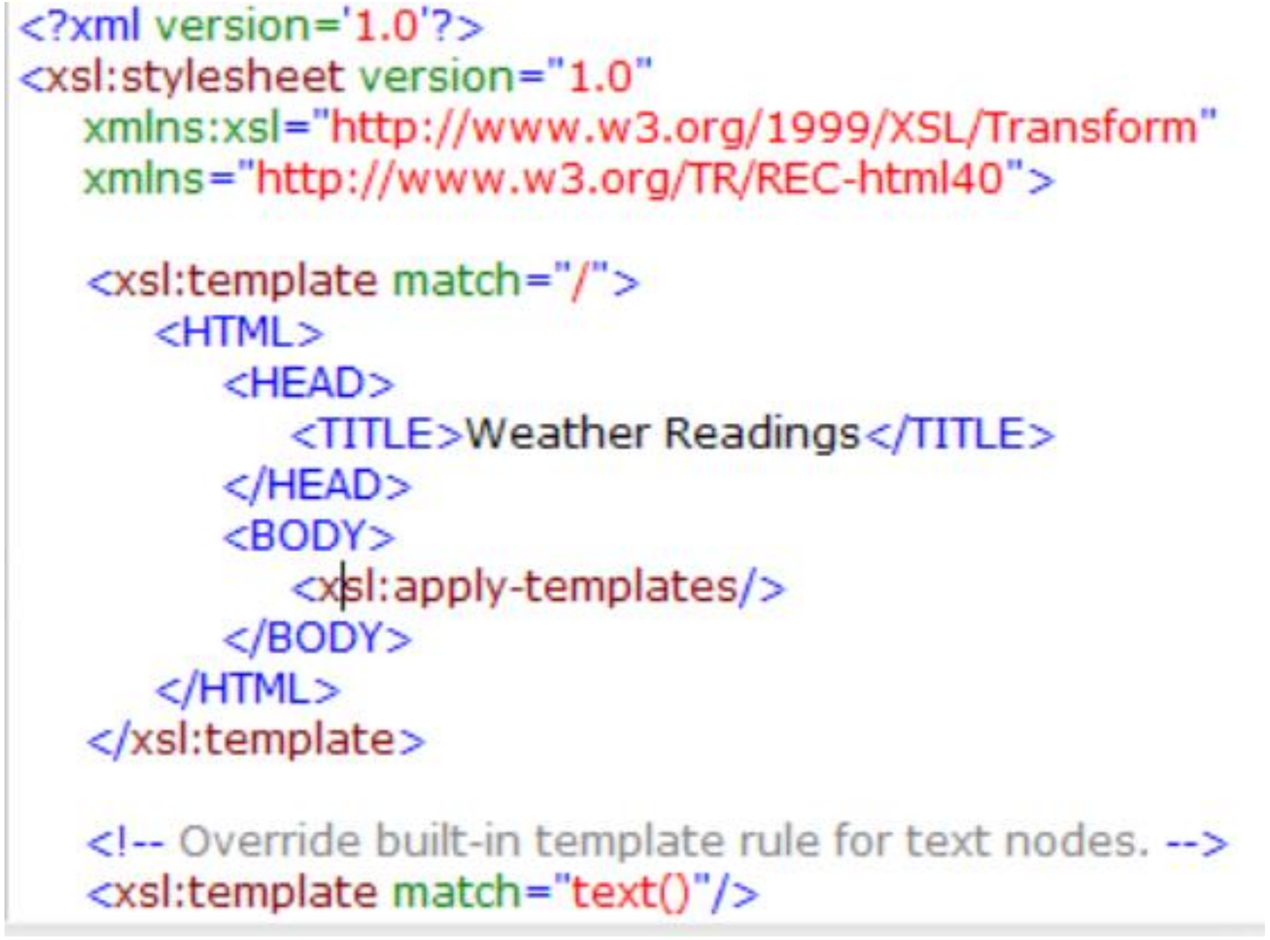
State	Year	Population	Violent crime total	Robbery	Property crime total	Burglary	Motor vehicle theft
Alabama	1960	3266740	6097	898	33823	11626	2853
Alabama	1961	3302000	5564	630	32541	11205	2535
Alabama	1962	3358000	5283	754	35829	11722	2801
Alabama	1963	3347000	6115	828	38521	12614	3033
Alabama	1964	3407000	7260	992	46290	15898	3679
Alabama	1965	3462000	6916	992	48215	16398	3702
Alabama	1966	3517000	8098	1124	53740	18551	4606
Alabama	1967	3540000	8448	1167	57079	20227	5170
Alabama	1968	3566000	8288	1462	62997	22403	6086
Alabama	1969	3531000	8842	1448	66248	23559	6045
Alabama	1970	3444165	10185	1731	75214	26739	7696
Alabama	1971	3479000	10835	2005	76084	27547	7696
Alabama	1972	3510000	10994	2407	73053	27714	6846
Alabama	1973	3539000	12390	2809	78999	31754	8039
Alabama	1974	3577000	13338	3562	93976	37841	9322
Alabama	1975	3614000	14201	4446	111296	42059	9767

UNSTRUCTURED / STRUCTURED DATA??



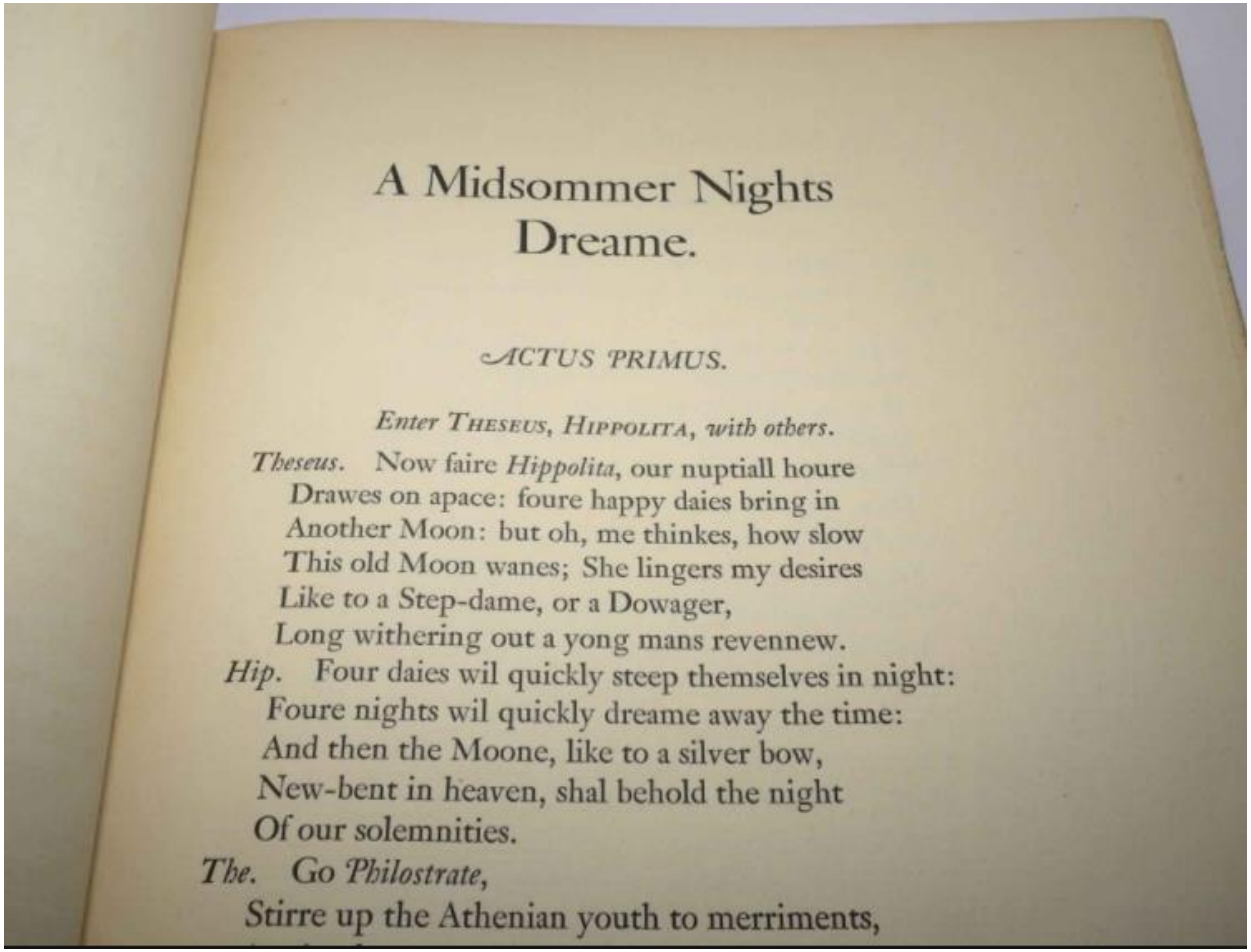
```
[INF Info]
  INF = C:\Windows\INF\oem169.inf
* Section <PackageInfo> Key <Sequence> not found in INF
  Date = 05/04/2016
  Version = 20.19.15.4454
  ClassGUID = {4D36E968-E325-11CE-BFC1-08002BE10318}
  PackageInfo.Name = Graphics
  PackageInfo.Sequence = 0
  Manufacturer = IntelGfx,NTamd64.6.1,NTamd64.6.2,NTamd64.6.3,NTamd64.10.0
  Resolved Manufacturer = IntelGfx.NTamd64.10.0
  Inf supports 64 bit.
  Description: Intel(R) HD Graphics
  HardwareID = iHSWM_w10,PCI\VEN_8086&DEV_0406&SUBSYS_05BD1028
```

UNSTRUCTURED / STRUCTURED DATA??



```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns="http://www.w3.org/TR/REC-html40">

   <xsl:template match="/">
      <HTML>
         <HEAD>
            <TITLE>Weather Readings</TITLE>
         </HEAD>
         <BODY>
            <xsl:apply-templates/>
         </BODY>
      </HTML>
   </xsl:template>

   <!-- Override built-in template rule for text nodes. -->
   <xsl:template match="text()"/>
```

UNSTRUCTURED / STRUCTURED DATA??



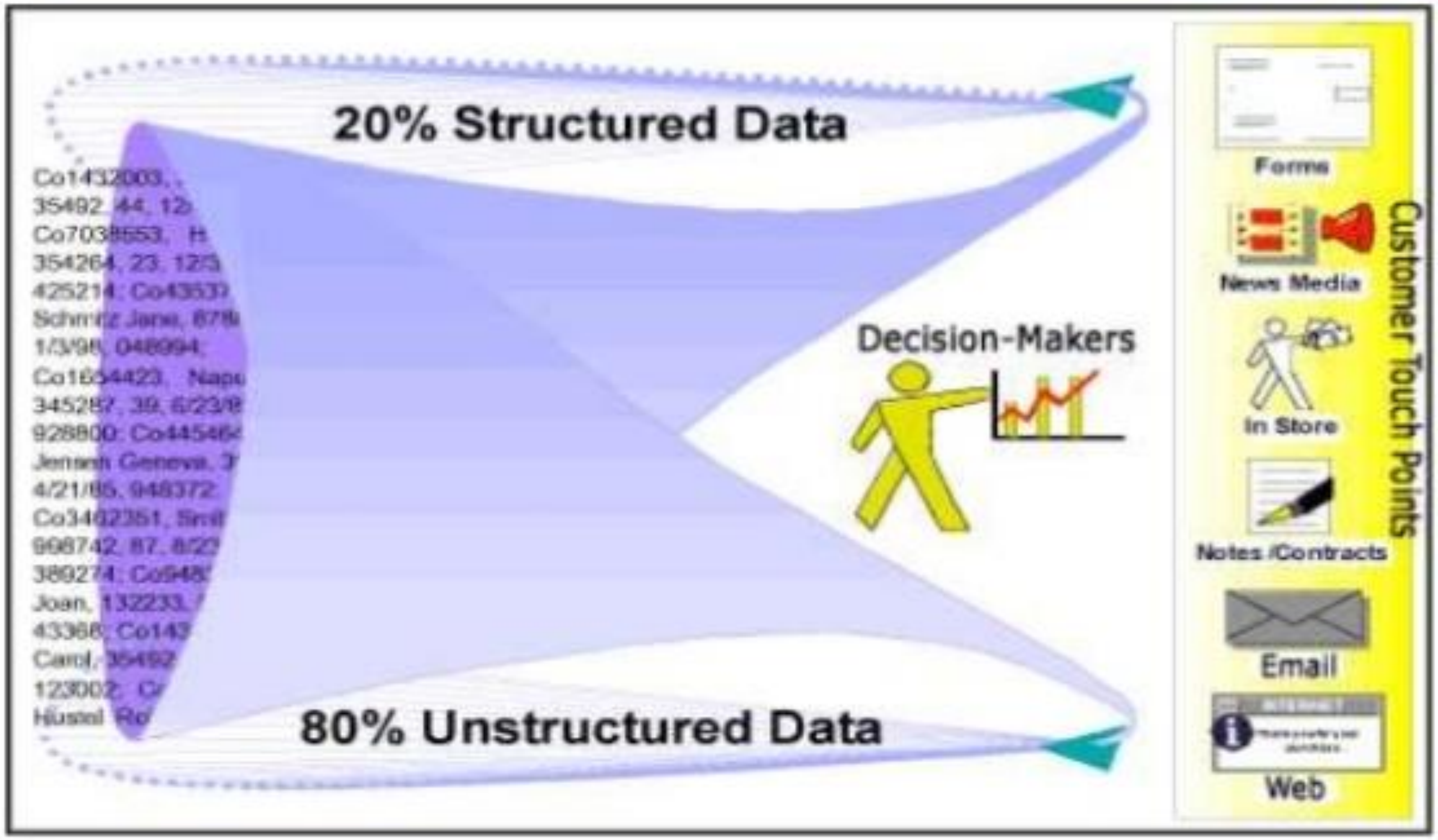
A Midsommer Nights
Dreame.

ACTUS PRIMUS.

Enter THESEUS, HIPPOLITA, *with others.*

Theseus. Now faire *Hippolita*, our nuptiall houre
Drawes on apace: foure happy daies bring in
Another Moon: but oh, me thinkes, how slow
This old Moon wanes; She lingers my desires
Like to a Step-dame, or a Dowager,
Long withering out a yong mans revennew.
Hip. Four daies wil quickly steep themselves in night:
Foure nights wil quickly dreame away the time:
And then the Moone, like to a silver bow,
New-bent in heaven, shal behold the night
Of our solemnities.
The. Go *Philostrate*,
Stirre up the Athenian youth to merriments,

BIRTH OF UNSTRUCTURED



BUSINESS DRIVERS

- **Customer 360 Degree View**
- **Retail**
  - Cross selling & Recommendation Engines
- **Teleccomunications**
  - Growing Data / Preventive Analytics
- **Finance**
  - Fraud Analytics
- **Life Sciences**
  - More targeted treatments
- **Media**
  - Personalized content

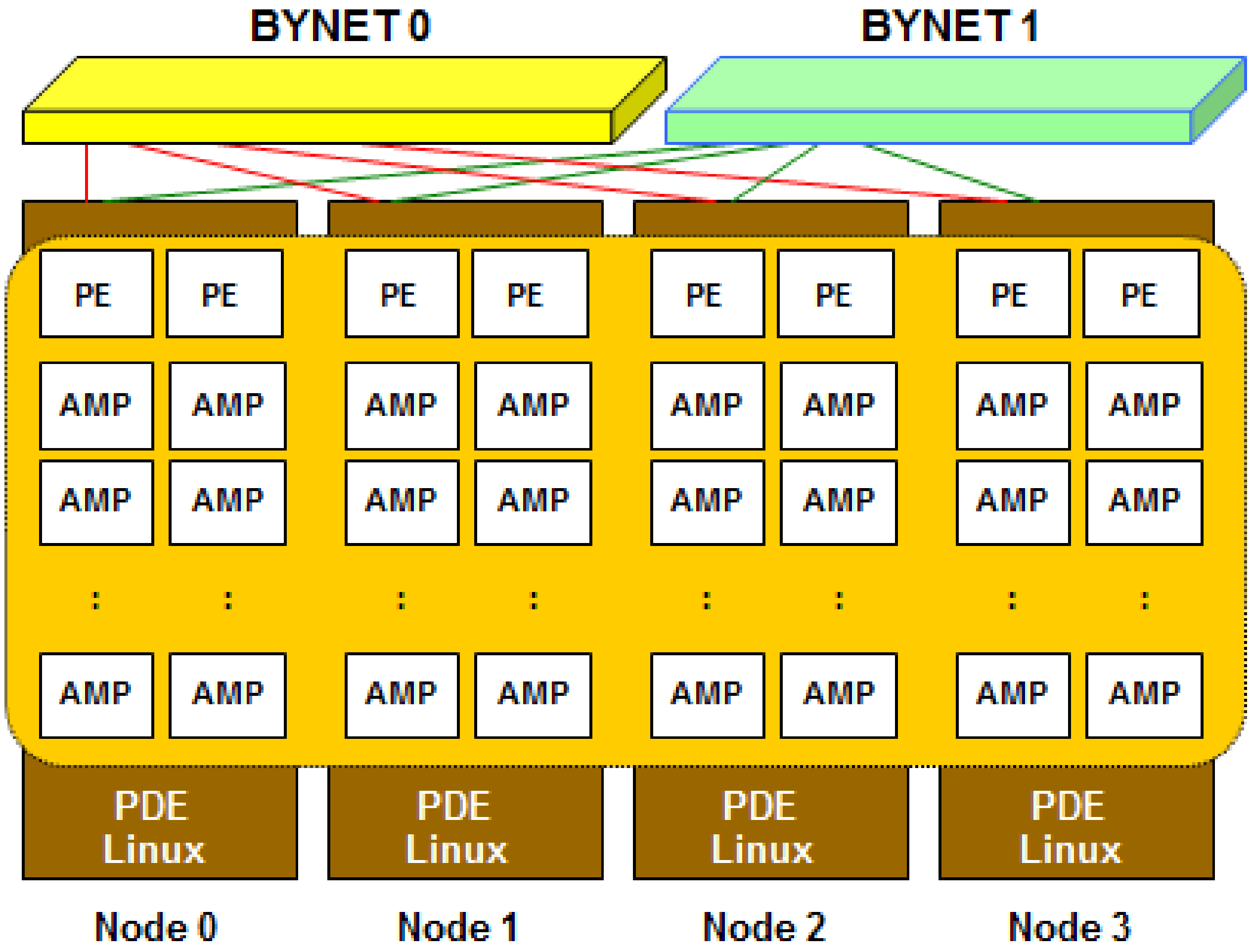
CHALLENGES WITH STORING LARGE DATASETS

- Disk Seek every access (IO Overhead)
- Hard Drive Capacity
- Spiraling Cost
- System Scalability

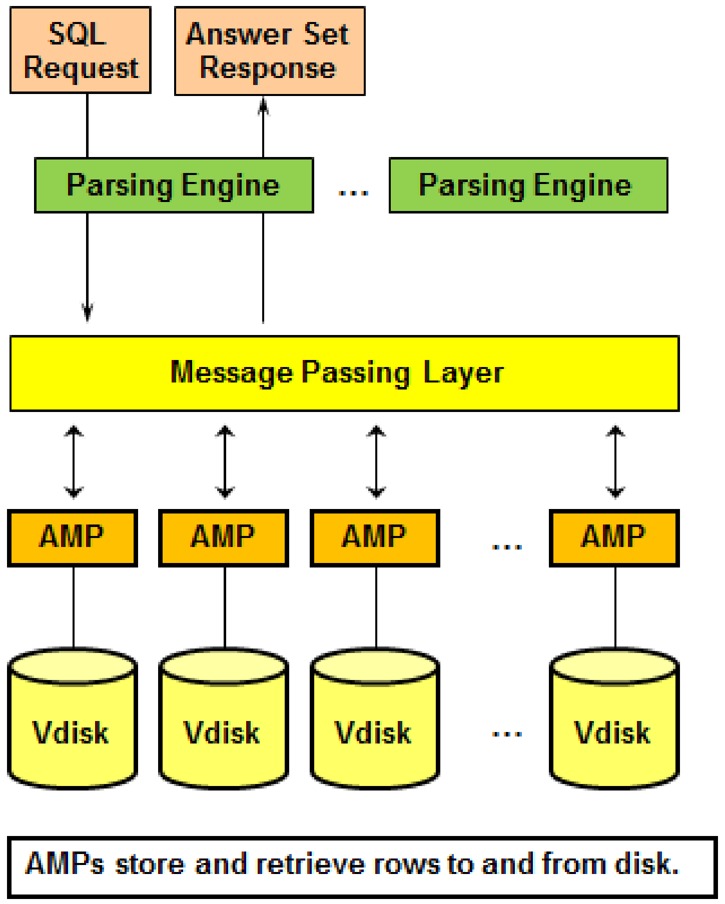
CONCERNS SURROUNDING PARALLELISM

- Splits & Aggregation
- Synchronization
- Deadlocks
- Transparency
- Failover & Redundancy

TERADATA



TERADATA NODE



- Data Split Up
- Compute Moves to Data
- Processed in parallel
- Aggregation possible
- Failover
- Fault Tolerance

MPP VENDORS

TERADATA

Raising Intelligence

IBM

NETEZZA

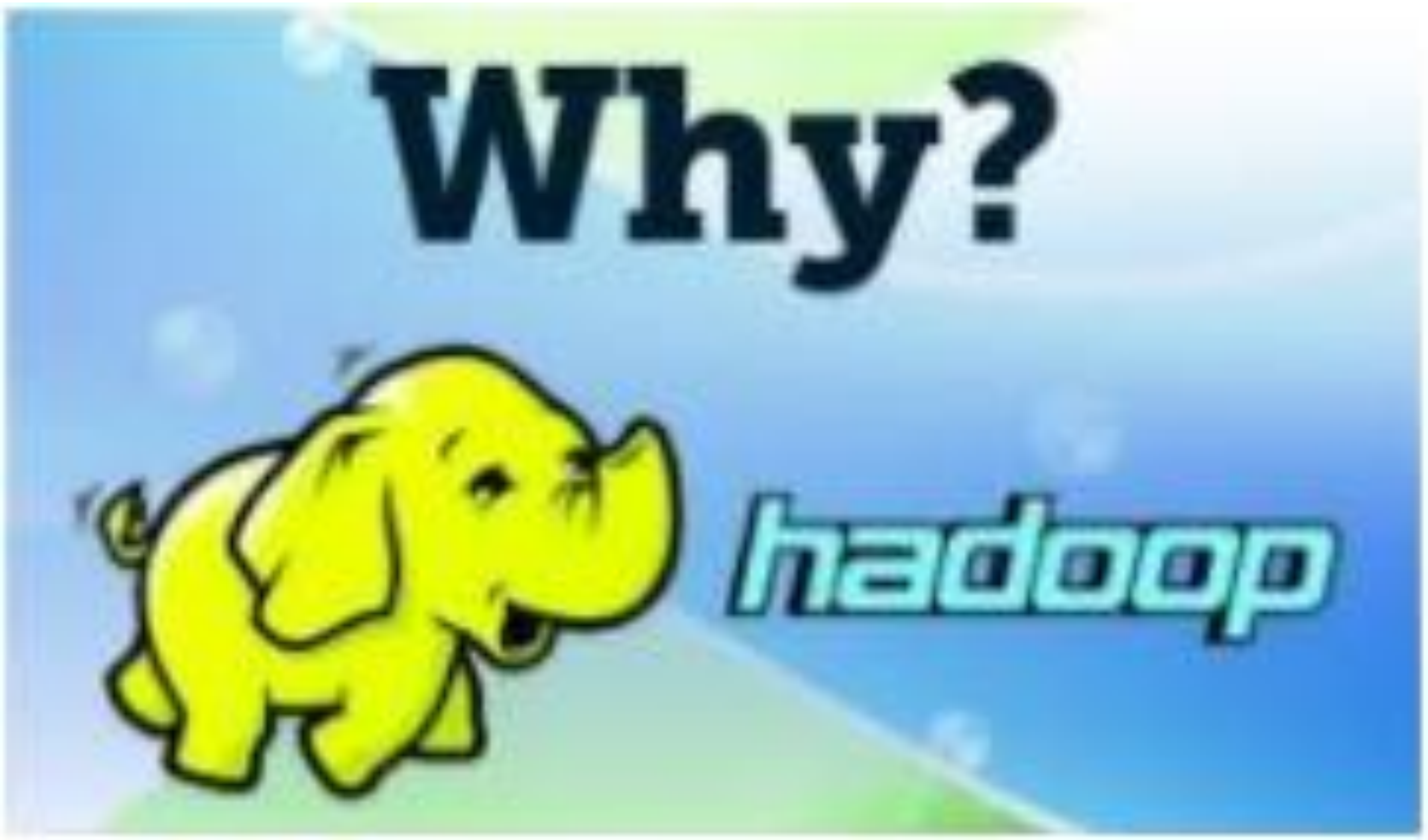
Greenplum

hp

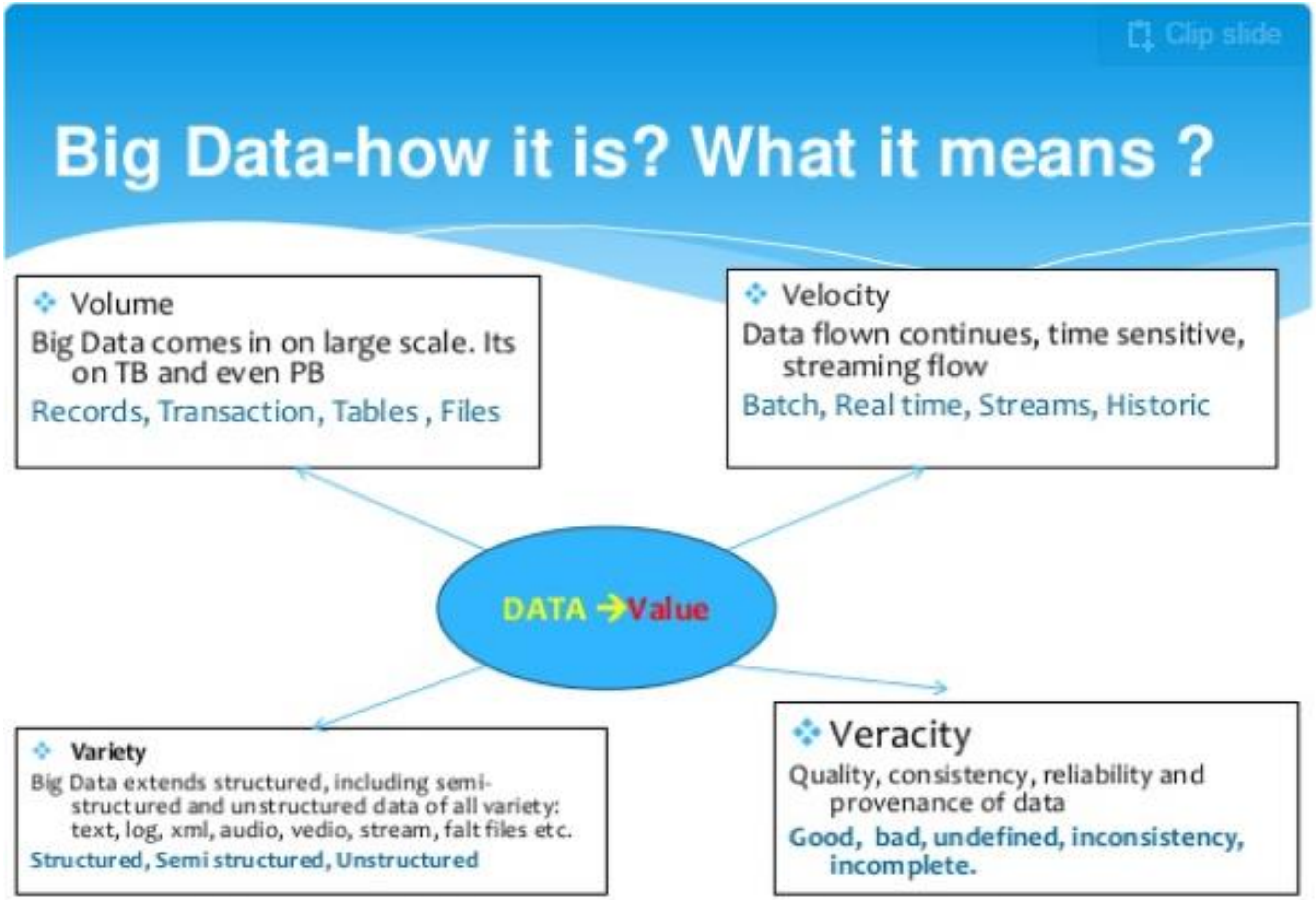
VERTICA

ORACLE

EXADATA


Why?
hadoop

V – V – V – V



Clip slide

Big Data-how it is? What it means ?

Volume
Big Data comes in on large scale. Its on TB and even PB
Records, Transaction, Tables , Files

Velocity
Data flown continues, time sensitive, streaming flow
Batch, Real time, Streams, Historic

DATA → Value

Variety
Big Data extends structured, including semi-structured and unstructured data of all variety: text, log, xml, audio, vedio, stream, falt files etc.
Structured, Semi structured, Unstructured

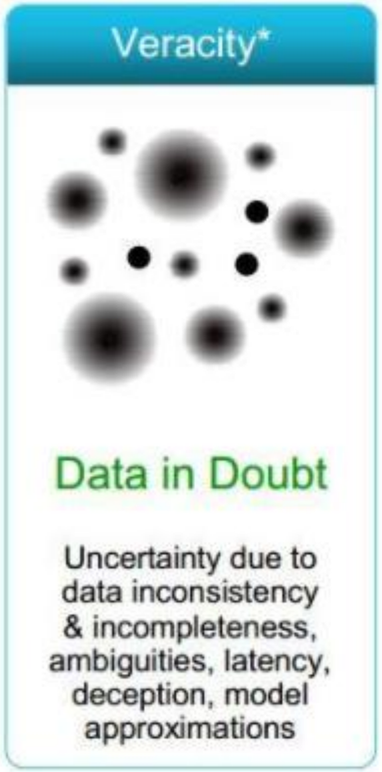
Veracity
Quality, consistency, reliability and provenance of data
Good, bad, undefined, inconsistency, incomplete.

DATA VOLUME

Terabyte
Gigabyte
Megabyte
Kilobyte
Kilobyte
Byte
Byte
Byte

DATA VELOCITY

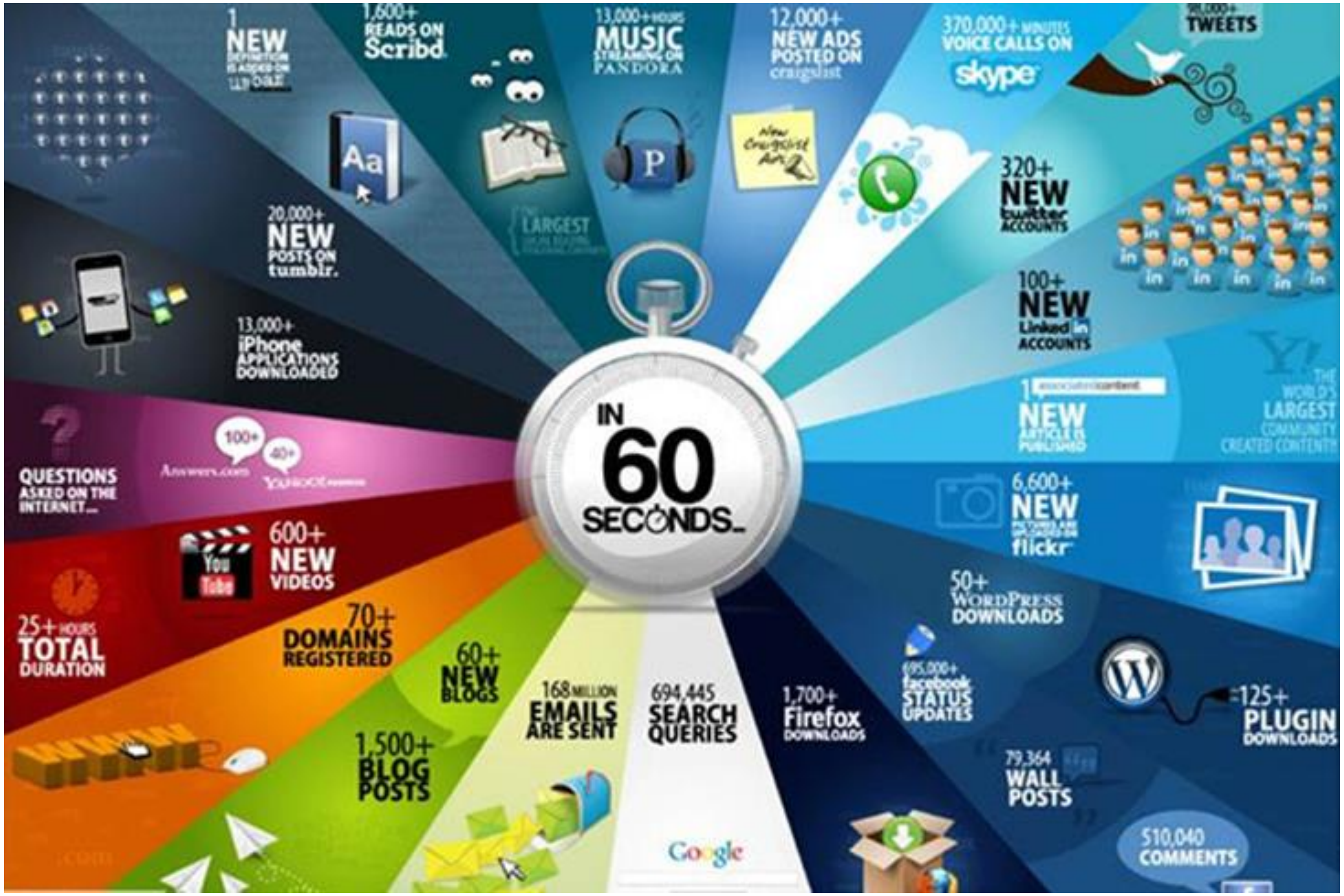
DATA VERACITY


Veracity*

Data in Doubt

Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

DATA VARIETY


IN 60 SECONDS
1 NEW DEFINITION IS ADDED ON URBAN DICTIONARY
1,600+ READS ON Scribd
13,000+ HOURS MUSIC STREAMING ON PANDORA
12,000+ NEW ADS POSTED ON craigslist
370,000+ MINUTES VOICE CALLS ON skype
98,000+ TWEETS
320+ NEW twitter ACCOUNTS
100+ NEW LinkedIn ACCOUNTS
1 NEW ARTICLE IS PUBLISHED
THE WORLD'S LARGEST COMMUNITY CREATED CONTENT
6,600+ NEW PICTURES ARE UPLOADED ON flickr
50+ WORDPRESS DOWNLOADS
125+ PLUGIN DOWNLOADS
695,000+ facebook STATUS UPDATES
79,364 WALL POSTS
510,040 COMMENTS
1,700+ Firefox DOWNLOADS
694,445 SEARCH QUERIES
Google
168 MILLION EMAILS ARE SENT
60+ NEW BLOGS
1,500+ BLOG POSTS
70+ DOMAINS REGISTERED
600+ NEW VIDEOS
YouTube
25+ HOURS TOTAL DURATION
QUESTIONS ASKED ON THE INTERNET...
Answers.com
100+
40+
YAHOO! ANSWERS
13,000+ iPhone APPLICATIONS DOWNLOADED
20,000+ NEW POSTS ON tumblr.
THE LARGEST
New Craigslist Art

BIG DATA

- Large Datasets Hard to manage with RDBMS
  - IoT
  - Social Media Data

- TCO
  - Parallel computation on 1000's of Machines
  - Affordability

- Legacy ETL needs batch window

HADOOP ARRIVES

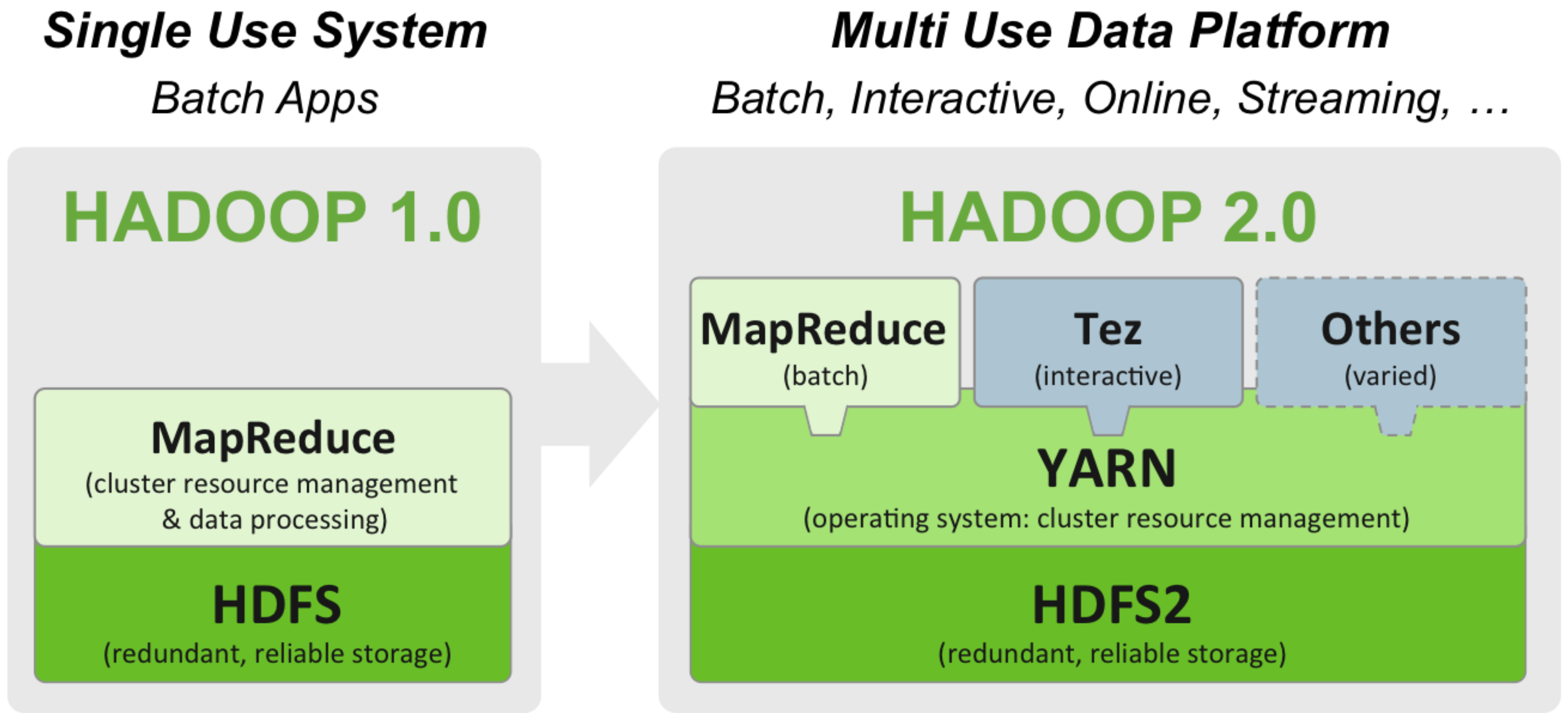
A Framework that allows **distributed processing** of large datasets across clusters of **large commodity servers** using a **simple programming model**

Open Source

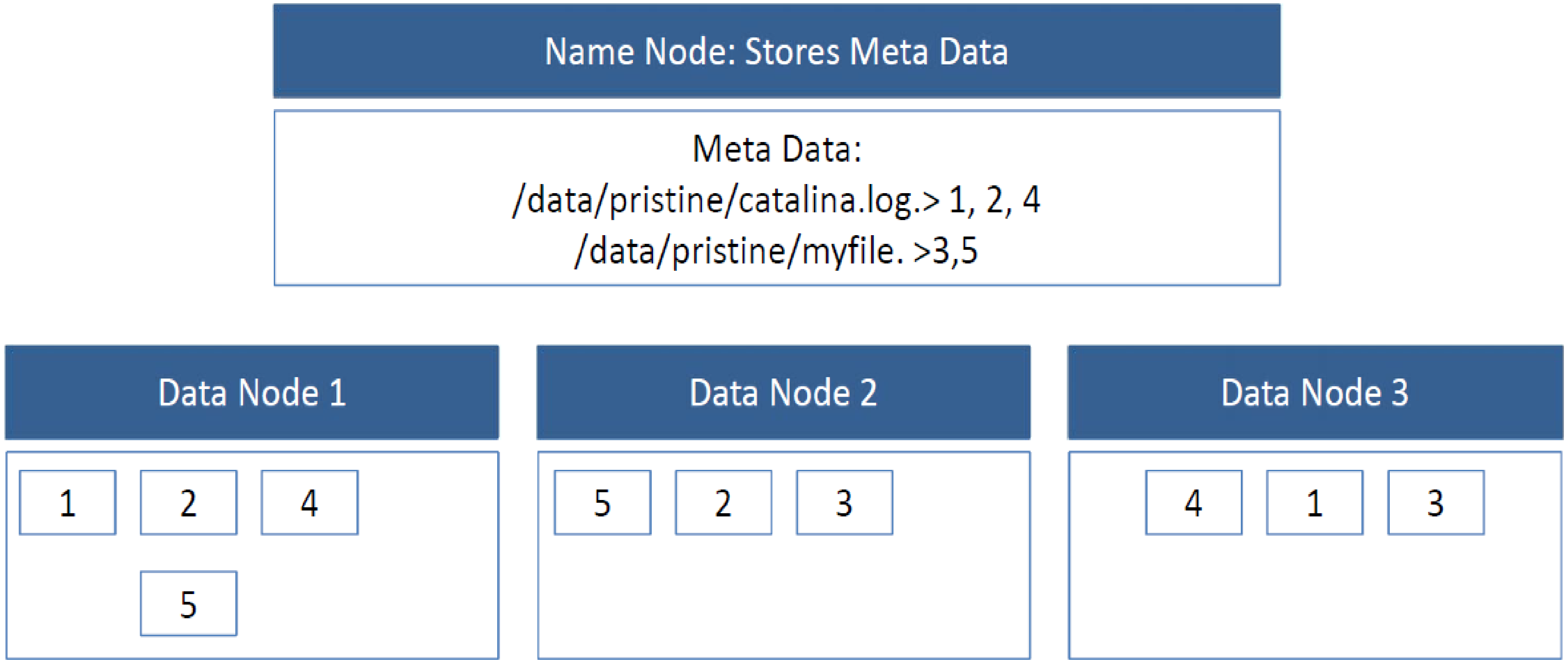
Structured & Unstructured

Self Healing

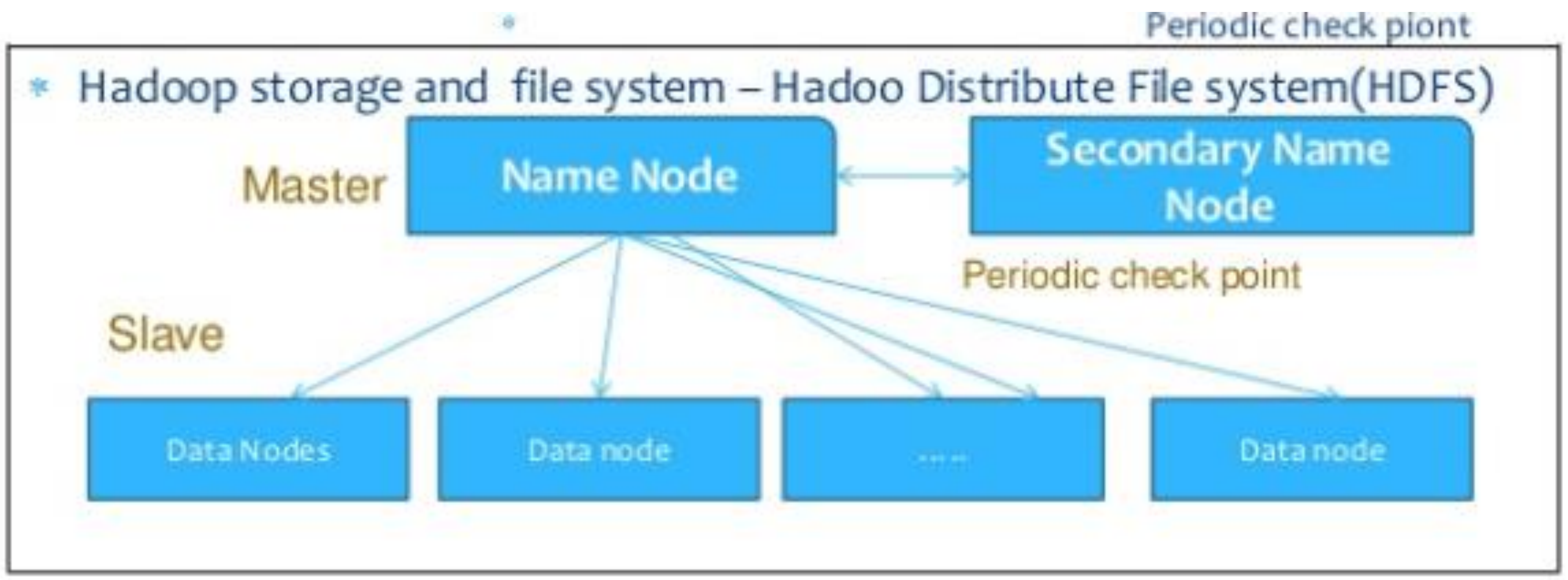
WHAT IS HADOOP?


Single Use System
Batch Apps
Multi Use Data Platform
Batch, Interactive, Online, Streaming, …
HADOOP 1.0
MapReduce
(cluster resource management & data processing)
HDFS
(redundant, reliable storage)
HADOOP 2.0
MapReduce
(batch)
Tez
(interactive)
Others
(varied)
YARN
(operating system: cluster resource management)
HDFS2
(redundant, reliable storage)

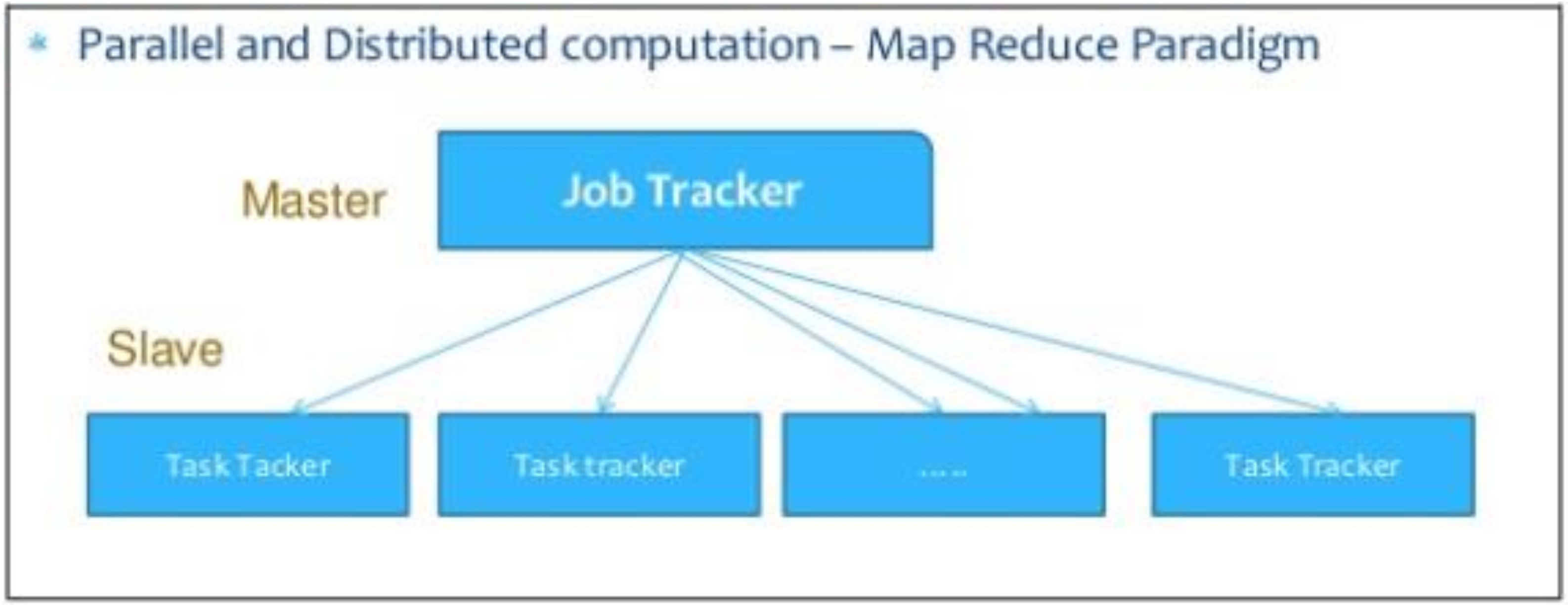
HADOOP ARCHITECTURE


Name Node: Stores Meta Data
Meta Data:
/data/pristine/catalina.log.> 1, 2, 4
/data/pristine/myfile. >3,5
Data Node 1
1
2
4
5
Data Node 2
5
2
3
Data Node 3
4
1
3

HDFS ARCHITECTURE

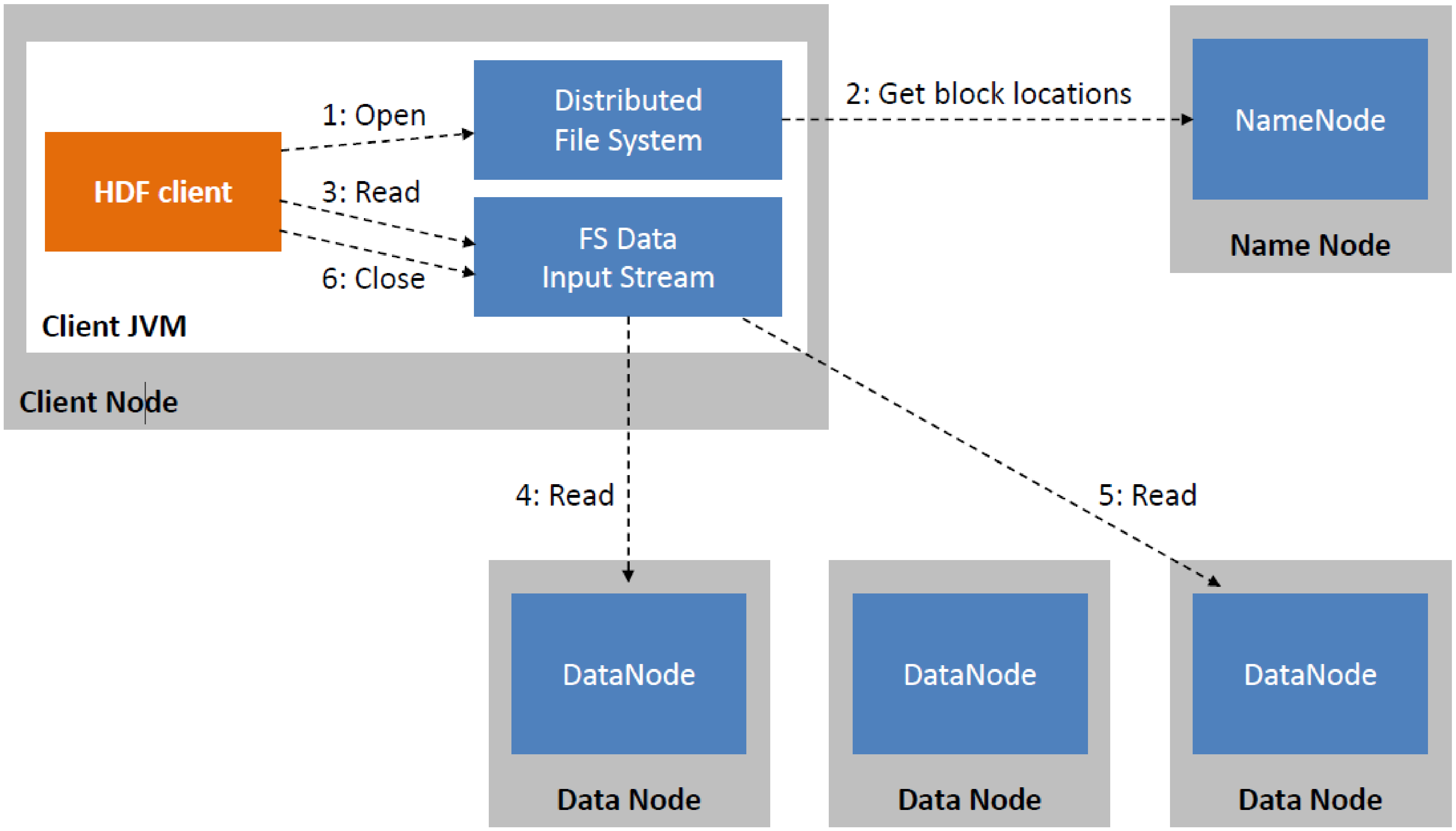


MAP-REDUCE PARADIGM



Then what's the Difference???

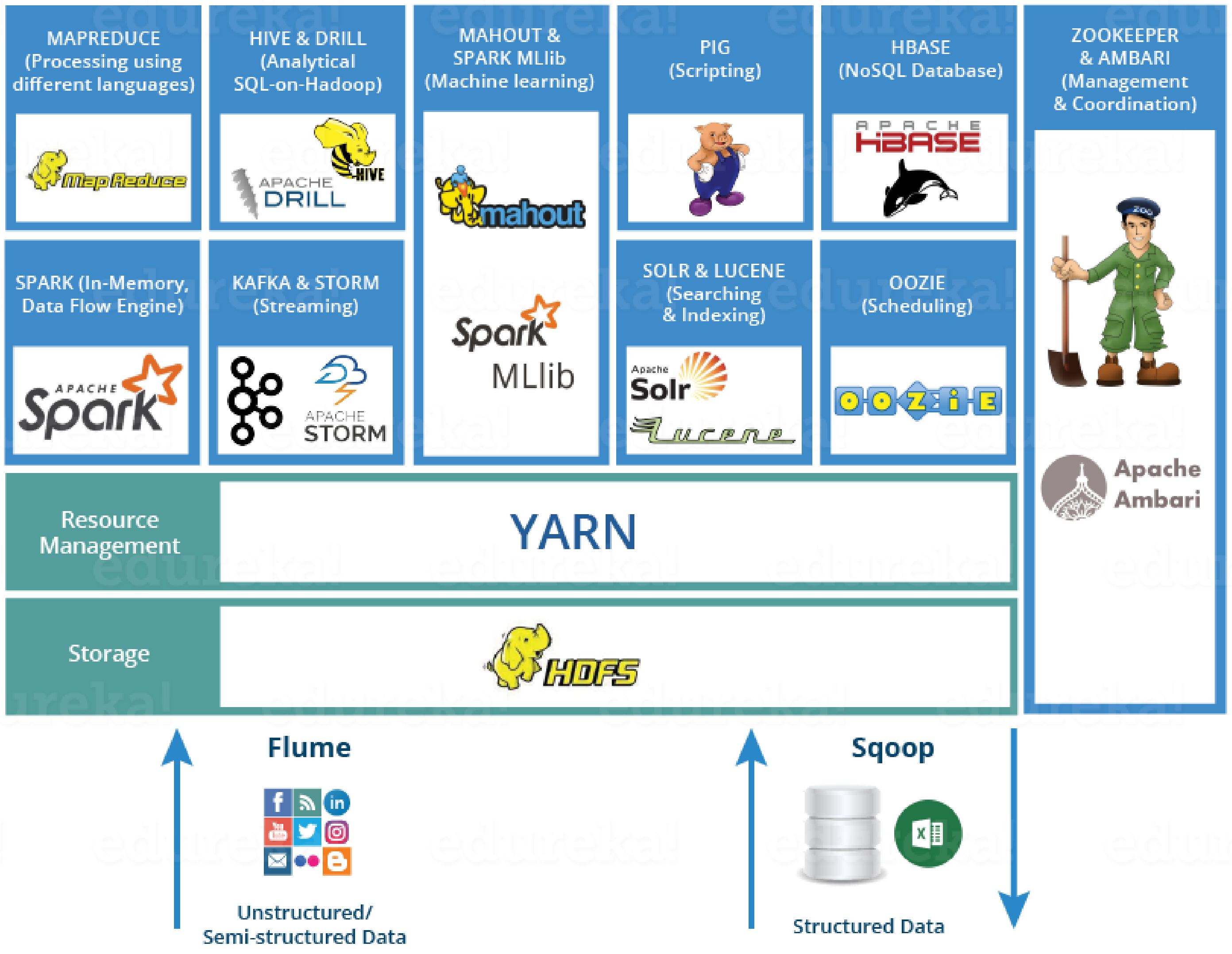
MAP-REDUCE ARCHITECTURE



HADOOP’S APPEAL

- Open Source
- Community Based
- Allow competing projects (survival of the fittest)
- Loose federation of projects
- Insures against vendor lock-in

HADOOP ECOSYSTEM

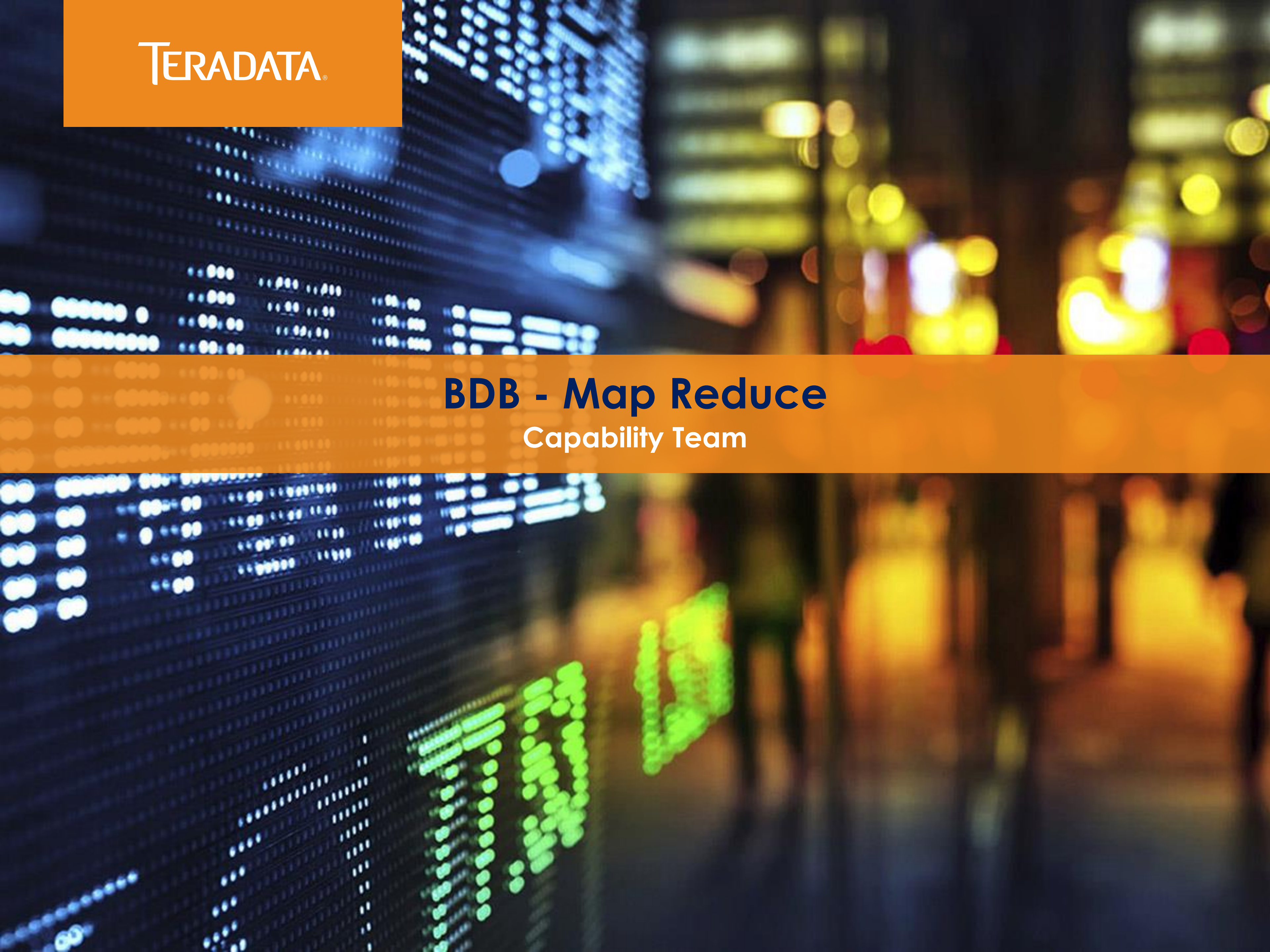


MAPREDUCE (Processing using different languages)
HIVE & DRILL (Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib (Machine learning)
PIG (Scripting)
HBASE (NoSQL Database)
ZOOKEEPER & AMBARI (Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM (Streaming)
SOLR & LUCENE (Searching & Indexing)
OOZIE (Scheduling)
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

HADOOP vs RDBMS

Facts	MPP	HADOOP
Data Size	Terabytes	Petabytes
Access	Interactive & Batch	Batch
Updates	Multiple Read / Write	Write Once, Read Many
Structure	Static Schema	Dynamic Schema
Integrity	High	Low
Scaling	Linear (TD)	Linear

TERADATA.



BDB - Map Reduce

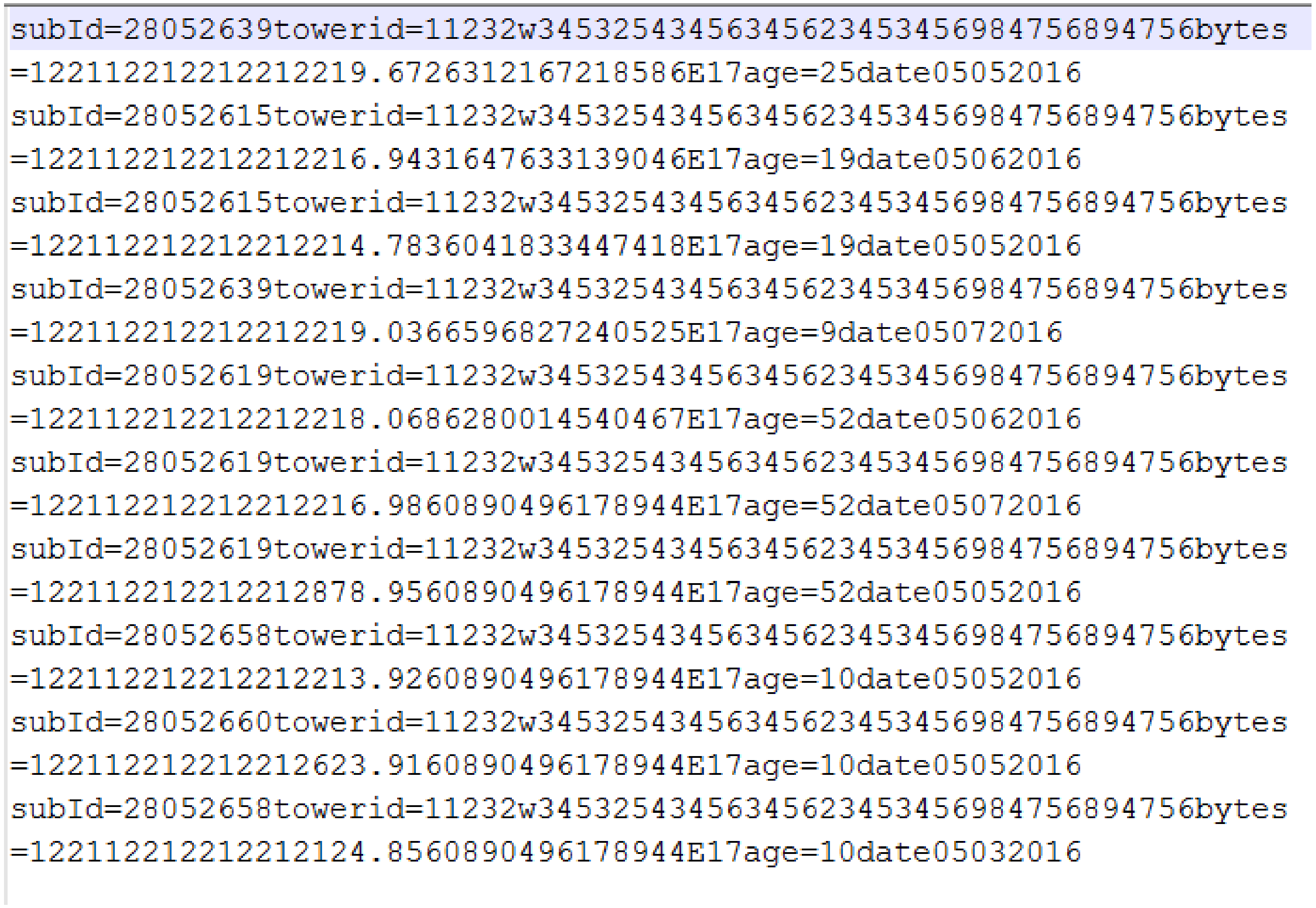
Capability Team

WHAT IS MAP REDUCE

- Split input files(e.g., by HDFS blocks – 64mb)
- Operate on key / value pairs
- Mappers filter and transform input data
- Reducers operate on mapper output
- YARN :- Yet Another Resource Negotiator , allows other applications on top of it.

	Input	Output
Map	<k1, v1>	list(<k2, v2>)
Reduce	<k2, list(v2)>	list(<k3, v3>)

UNSTRUCTURED DATA



```
subId=28052639towerid=11232w34532543456345623453456984756894756bytes
=12211221221221219.6726312167218586E17age=25date05052016
subId=28052615towerid=11232w34532543456345623453456984756894756bytes
=12211221221221216.9431647633139046E17age=19date05062016
subId=28052615towerid=11232w34532543456345623453456984756894756bytes
=12211221221221214.7836041833447418E17age=19date05052016
subId=28052639towerid=11232w34532543456345623453456984756894756bytes
=12211221221221219.0366596827240525E17age=9date05072016
subId=28052619towerid=11232w34532543456345623453456984756894756bytes
=12211221221221218.0686280014540467E17age=52date05062016
subId=28052619towerid=11232w34532543456345623453456984756894756bytes
=12211221221221216.9860890496178944E17age=52date05072016
subId=28052619towerid=11232w34532543456345623453456984756894756bytes
=12211221221212878.9560890496178944E17age=52date05052016
subId=28052658towerid=11232w34532543456345623453456984756894756bytes
=12211221221221213.9260890496178944E17age=10date05052016
subId=28052660towerid=11232w34532543456345623453456984756894756bytes
=12211221221212623.9160890496178944E17age=10date05052016
subId=28052658towerid=11232w34532543456345623453456984756894756bytes
=12211221221212124.8560890496178944E17age=10date05032016
```

MAP PHASE

Key	Value
28052639	1221122122122219.6726312167218586E17
28052615	1221122122122216.9431647633139046E17
28052615	1221122122122214.7836041833447418E17
28052639	1221122122122219.0366596827240525E17
28052619	1221122122122218.0686280014540467E17
28052619	1221122122122216.9860890496178944E17
28052619	1221122122122878.9560890496178944E17
28052658	1221122122122213.9260890496178944E17
28052660	1221122122122623.9160890496178944E17
28052658	1221122122122124.8560890496178944E17

MAP – SHUFFLE & SORT PHASE

Key	Value
28052639	(1221122122212219.6726312167218586E17, 1221122122212219.0366596827240525E17)
28052615	(1221122122212216.9431647633139046E17, 1221122122212214.7836041833447418E17)
28052619	(1221122122212218.0686280014540467E17, 1221122122212216.9860890496178944E17, 1221122122212878.9560890496178944E17)
28052658	(1221122122212213.9260890496178944E17, 1221122122212124.8560890496178944E17)
28052660	1221122122212623.9160890496178944E17

REDUCE PHASE - OUTPUT

Key	Value
28052639	SUM(1221122122122219.6726312167218586E17, 1221122122122219.0366596827240525E17) **2442244244244438.7092908994459**
28052615	SUM(1221122122122216.9431647633139046E17, 1221122122122214.7836041833447418E17) **=2442244244244431.72676894665864**
28052619	SUM(1221122122122218.0686280014540467E17, 1221122122122216.9860890496178944E17, 1221122122122878.9560890496178944E17) =**3663366366637314.01080610068982**
28052658	SUM(1221122122122213.9260890496178944E17, 1221122122122124.8560890496178944E17) =**2442244244244338.78217809923578**
28052660	SUM(1221122122122623.9160890496178944E17) =**1221122122122623.9160890496178944E17**

MAP REDUCE STEPS

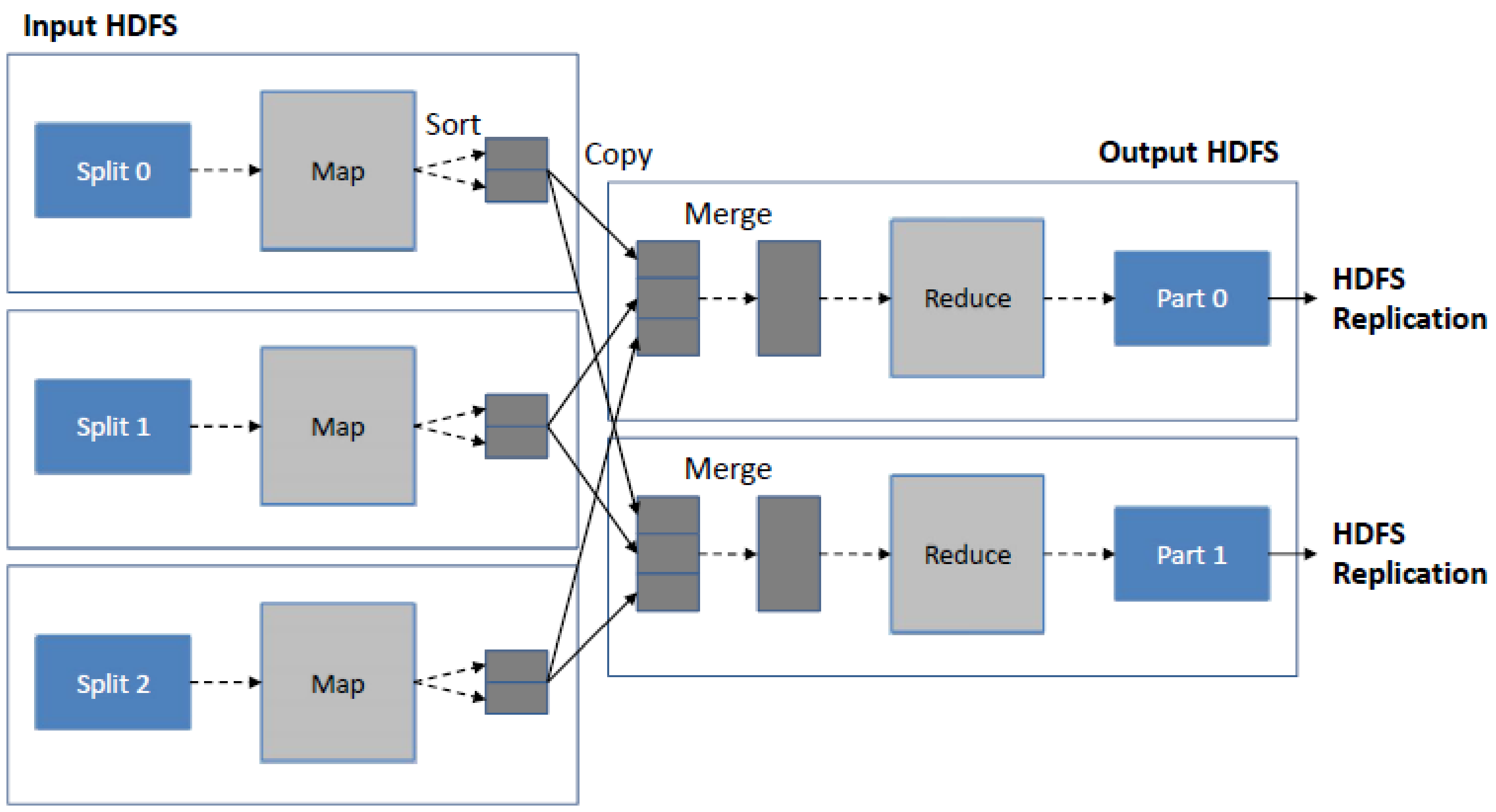
- Split input data into independent chunks
- MAP Phase (input/output key value pair)
- Shuffling and sorting
- Reduce Phase(input/output key value pair)
- Compute and storage nodes are same. i.e., MR and HDFS run on same nodes.
- Schedule tasks on nodes where data is already present

MAPPERS vs REDUCERS

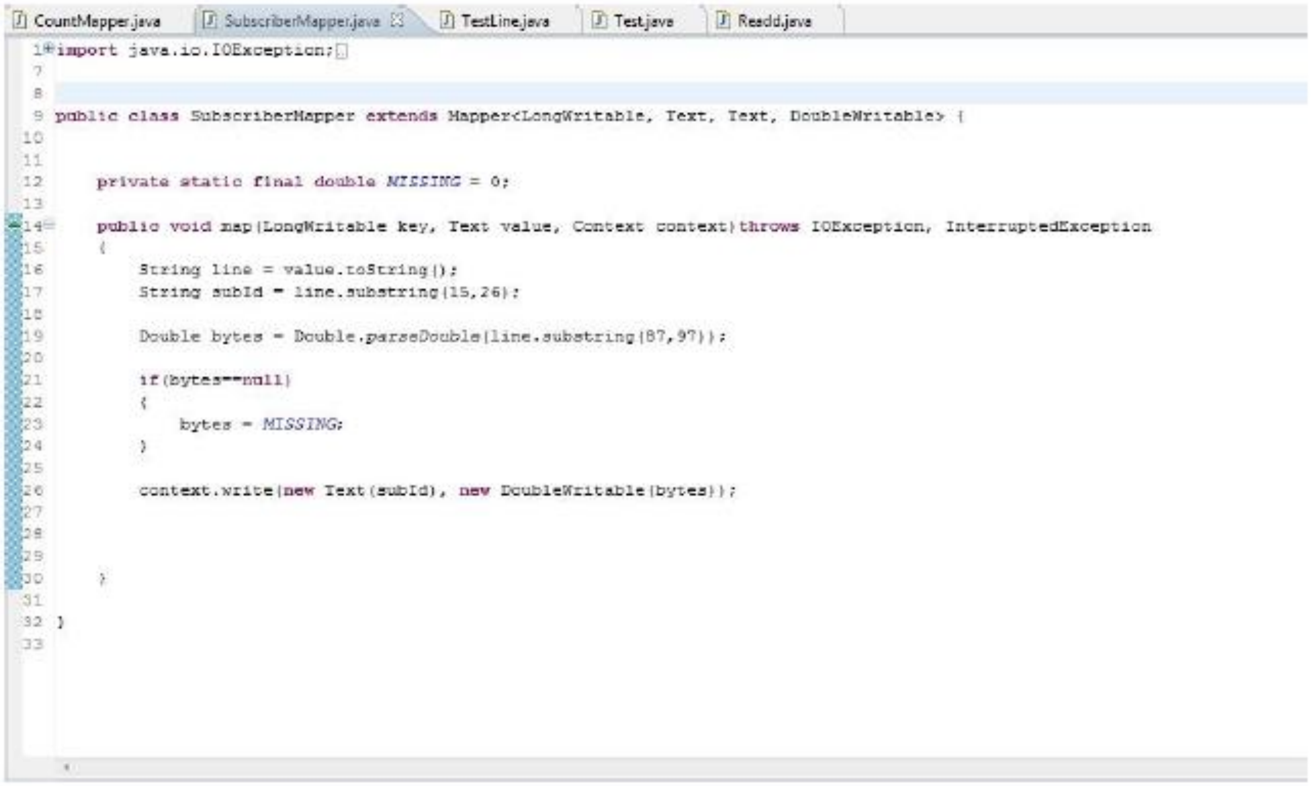
- Number of mappers required for a job: -
  - size of data to be processed. i.e.,
  - total number of blocks of input files

- Number of reduces required for a job: -
  - is approximately 0.95 or 0.75* (<numofnodes>*mapred.tasktracker.reduce.tasks.maximum)
  - Increasing reducers increases overhead, but the load balancing improves

MAP REDUCE – THE BIG PICTURE

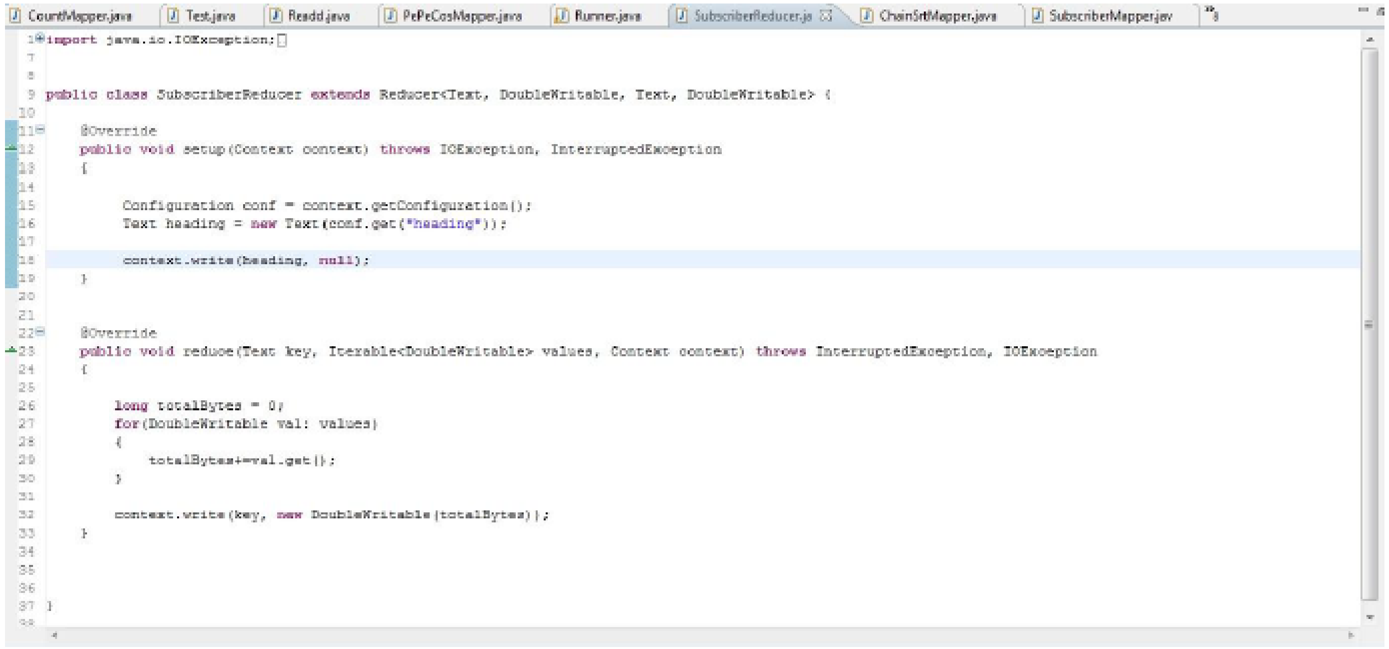


MAP REDUCE – The MAPPER



```
import java.io.IOException;


public class SubscriberMapper extends Mapper<LongWritable, Text, Text, DoubleWritable> {


    private static final double MISSING = 0;

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException
    {
        String line = value.toString();
        String subId = line.substring(15,26);

        Double bytes = Double.parseDouble(line.substring(87,97));

        if(bytes==null)
        {
            bytes = MISSING;
        }

        context.write(new Text(subId), new DoubleWritable(bytes));



    }

}
```

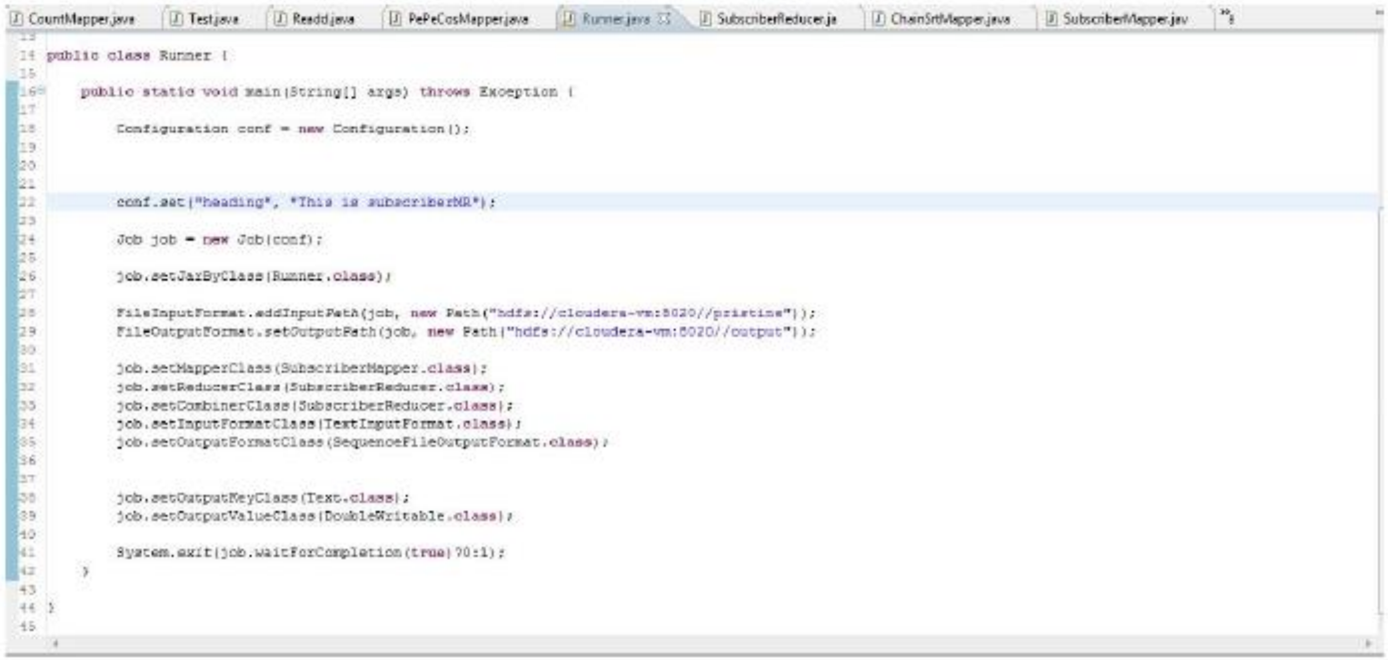
MAP REDUCE – The REDUCER



CountMapper.java | Test.java | Readd.java | PePeCosMapper.java | Runner.java | SubscriberReducer.ja | ChainSrtMapper.java | SubscriberMapper.jav

```java
import java.io.IOException;


public class SubscriberReducer extends Reducer<Text, DoubleWritable, Text, DoubleWritable> {

    @Override
    public void setup(Context context) throws IOException, InterruptedException
    {

        Configuration conf = context.getConfiguration();
        Text heading = new Text(conf.get("heading"));

        context.write(heading, null);
    }


    @Override
    public void reduce(Text key, Iterable<DoubleWritable> values, Context context) throws InterruptedException, IOException
    {

        long totalBytes = 0;
        for(DoubleWritable val: values)
        {
            totalBytes+=val.get();
        }

        context.write(key, new DoubleWritable(totalBytes));
    }



}
```

MAP REDUCE – MAIN CLASS



```java
public class Runner {

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

        Configuration conf = new Configuration();



        conf.set("heading", "This is subscriberMR");

        Job job = new Job(conf);

        job.setJarByClass(Runner.class);

        FileInputFormat.addInputPath(job, new Path("hdfs://cloudera-vm:8020//pristine"));
        FileOutputFormat.setOutputPath(job, new Path("hdfs://cloudera-vm:8020//output"));

        job.setMapperClass(SubscriberMapper.class);
        job.setReducerClass(SubscriberReducer.class);
        job.setCombinerClass(SubscriberReducer.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(SequenceFileOutputFormat.class);


        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);

        System.exit(job.waitForCompletion(true)?0:1);
    }

}
```

MAP REDUCE – what next?

- Lots of disk I/O
- Not good for Interactive
- Programming Language Familiarity

TERADATA

BDB - HIVE

Capability Team

WHAT IS HIVE

- Data warehouse system for Hadoop
- Developed by FACEBOOK
- Run SQL-like queries that get compiled and run as Map Reduce jobs.
- Displays the result back to the user
- Data in hadoop even though generally unstructured has some vague structure associated

TERADATA.

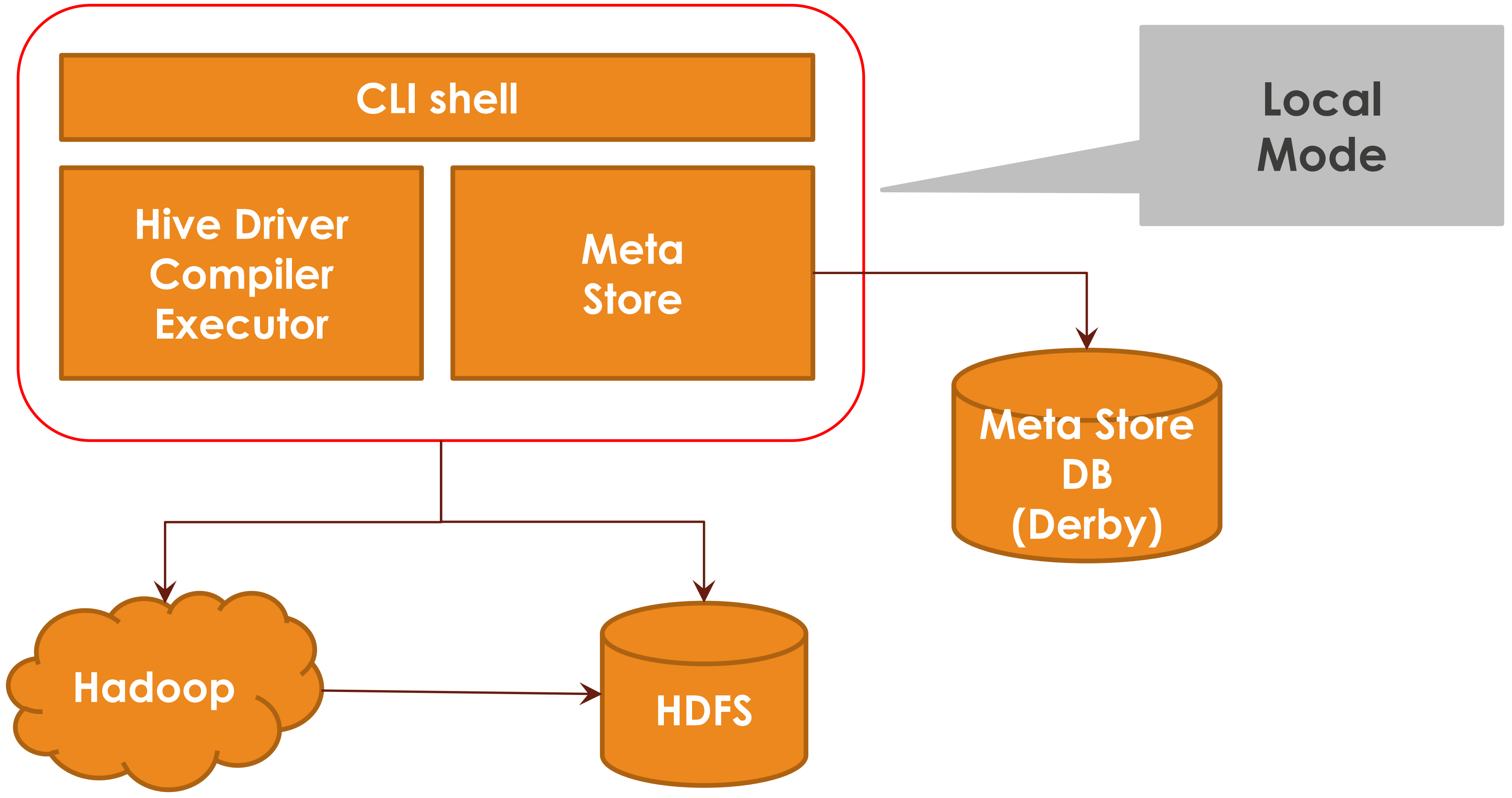
HIVE FEATURES

- Create table, create view, create index –DDL
- Select, where clause, group by, order by and joins – DML
- Pluggable Input output format
- Pluggable:
  - User Defined functions – UDFs
  - User Defined Aggregate Functions – UDAF
  - User Defined Table Functions – UDTF
- Pluggable Serializable-Deserializable libraries (Serde's)

WHAT HIVE IS NOT?

- It is not RDBMS
- OLTP workloads- low latency
- Correlated sub queries
- Even with small amount of data time to return the response can’t be compared to RDBMs

HIVE ARCHITECTURE



HIVE PRIMITIVE DATA-TYPES

CATEGORY	DATATYPE
Integers	TINYINT, INT, SMALLINT, BIGINT
Boolean	BOOLEAN
Floating Point	FLOAT, DOUBLE
Fixed Point	DECIMAL
String	STRING, VARCHAR, CHAR
Date & Time	TIMESTAMP, DATE
Binary	BINARY

HIVE COMPLEX DATA-TYPES

CATEGORY	DATATYPE
STRUCTS	(a:int, b:int)
MAP	(key,value)
ARRAYS	(1,2,3,4 – same data type)

HIVE-QL

- INSERT OVERWRITE … (***)
- filter rows with WHERE clause.
- SELECT clause.
- EQUIJOINS
- aggregations on multiple "group by" columns for data in a table.
- Store query results in another table.
- store query results in a hadoop dfs directory.
- Manage tables and partitions (create, drop and alter).

HIVE FILE FORMATS

- Text File (default)
- Sequence File
- RC File
- ORC
- AVRO
- PARQUEET

EXTERNAL TABLES vs INTERNAL TABLES

- Points to directories in HDFS
- Can Create tables and Partitions
- Partition columns become annotations

HIVE-TRANSACTIONS

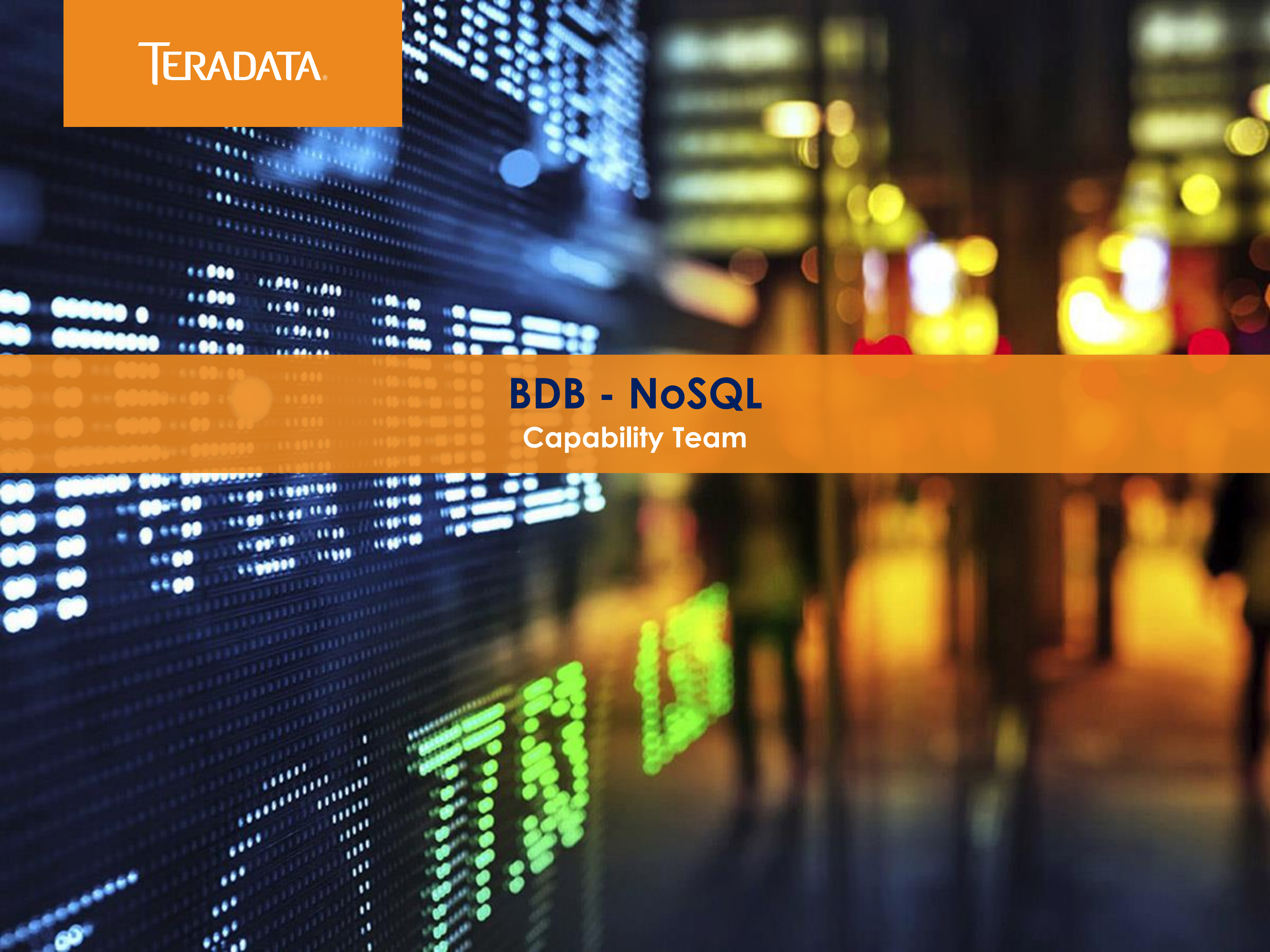
- Auto-commit (BEGIN, COMMIT & ROLLBACK)
- From version 0.14
- Only on ORC files
- No External Tables
- Tables should be Bucketed
- Hive Transaction manager Must be set
- LOAD DATA not supported
- Base Files Directory & Delta Files Directory

HIVE – TABLE CREATION

- CREATE TABLE t (colname DATATYPE,..)
    - ROW FORMAT DELIMITED
    - FIELDS TERMINATED BY char
    - STORED AS {TEXTFILE/SEQUENCEFILE}

HIVE LIMITATIONS

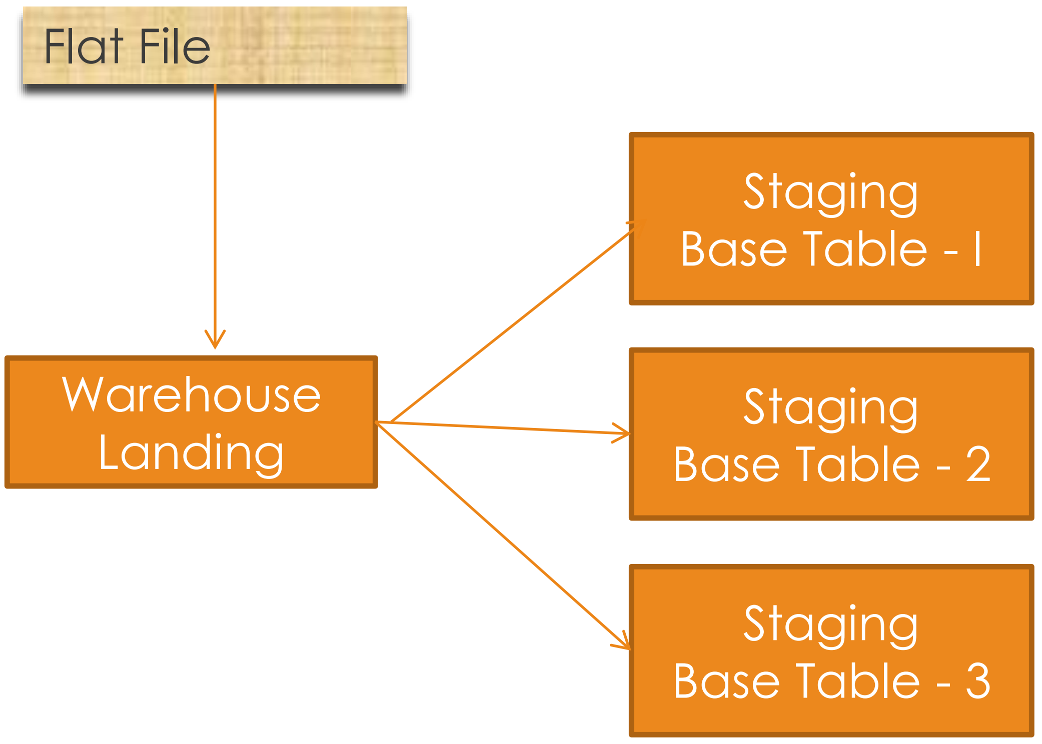
- Slower Response Time
- Cannot be used sequence of steps for applications like ETL
- Insert only, no Update *(newer versions possible – Transaction table only)*



BDB - NoSQL

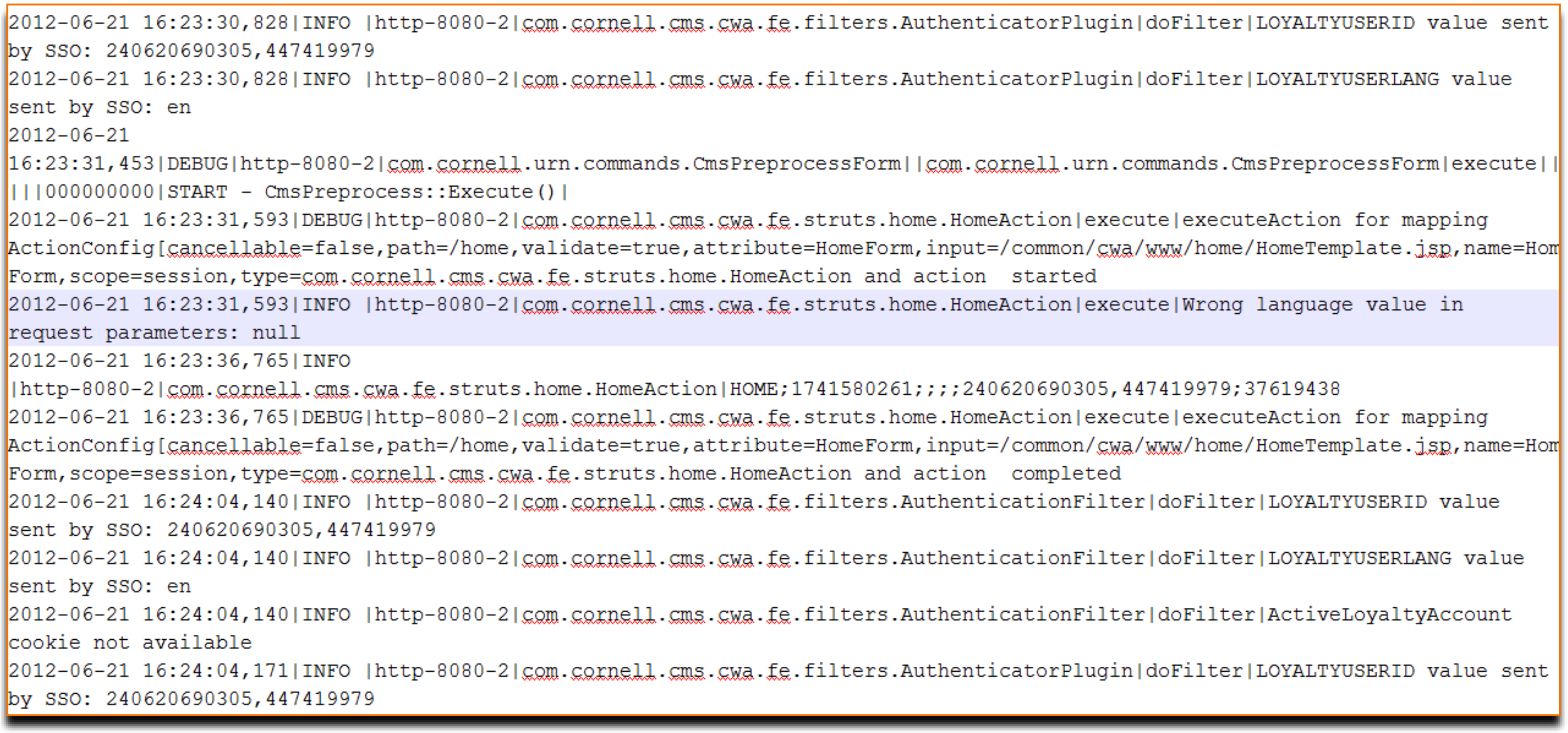
Capability Team

RDBMS Challenges



Analytics User

LOYALTY FLAT FILE

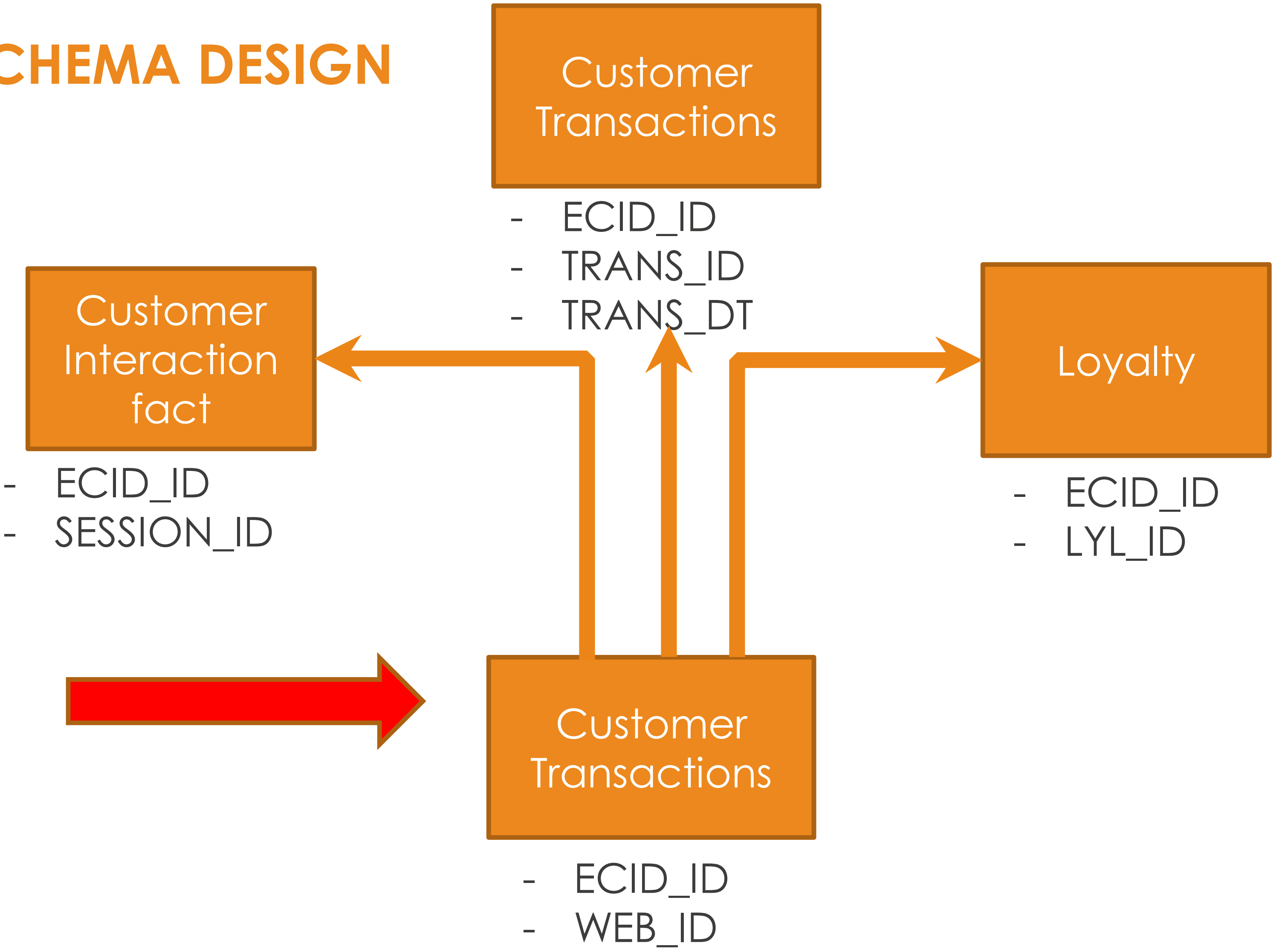


```
2012-06-21 16:23:30,828|INFO |http-8080-2|com.cornell.cms.cwa.fe.filters.AuthenticatorPlugin|doFilter|LOYALTYUSERID value sent
by SSO: 240620690305,447419979
2012-06-21 16:23:30,828|INFO |http-8080-2|com.cornell.cms.cwa.fe.filters.AuthenticatorPlugin|doFilter|LOYALTYUSERLANG value
sent by SSO: en
2012-06-21
16:23:31,453|DEBUG|http-8080-2|com.cornell.urn.commands.CmsPreprocessForm||com.cornell.urn.commands.CmsPreprocessForm|execute||
|||000000000|START - CmsPreprocess::Execute()|
2012-06-21 16:23:31,593|DEBUG|http-8080-2|com.cornell.cms.cwa.fe.struts.home.HomeAction|execute|executeAction for mapping
ActionConfig[cancellable=false,path=/home,validate=true,attribute=HomeForm,input=/common/cwa/www/home/HomeTemplate.jsp,name=Hom
Form,scope=session,type=com.cornell.cms.cwa.fe.struts.home.HomeAction and action  started
2012-06-21 16:23:31,593|INFO |http-8080-2|com.cornell.cms.cwa.fe.struts.home.HomeAction|execute|Wrong language value in
request parameters: null
2012-06-21 16:23:36,765|INFO
|http-8080-2|com.cornell.cms.cwa.fe.struts.home.HomeAction|HOME;1741580261;;;;240620690305,447419979;37619438
2012-06-21 16:23:36,765|DEBUG|http-8080-2|com.cornell.cms.cwa.fe.struts.home.HomeAction|execute|executeAction for mapping
ActionConfig[cancellable=false,path=/home,validate=true,attribute=HomeForm,input=/common/cwa/www/home/HomeTemplate.jsp,name=Hom
Form,scope=session,type=com.cornell.cms.cwa.fe.struts.home.HomeAction and action  completed
2012-06-21 16:24:04,140|INFO |http-8080-2|com.cornell.cms.cwa.fe.filters.AuthenticationFilter|doFilter|LOYALTYUSERID value
sent by SSO: 240620690305,447419979
2012-06-21 16:24:04,140|INFO |http-8080-2|com.cornell.cms.cwa.fe.filters.AuthenticationFilter|doFilter|LOYALTYUSERLANG value
sent by SSO: en
2012-06-21 16:24:04,140|INFO |http-8080-2|com.cornell.cms.cwa.fe.filters.AuthenticationFilter|doFilter|ActiveLoyaltyAccount
cookie not available
2012-06-21 16:24:04,171|INFO |http-8080-2|com.cornell.cms.cwa.fe.filters.AuthenticatorPlugin|doFilter|LOYALTYUSERID value sent
by SSO: 240620690305,447419979
```

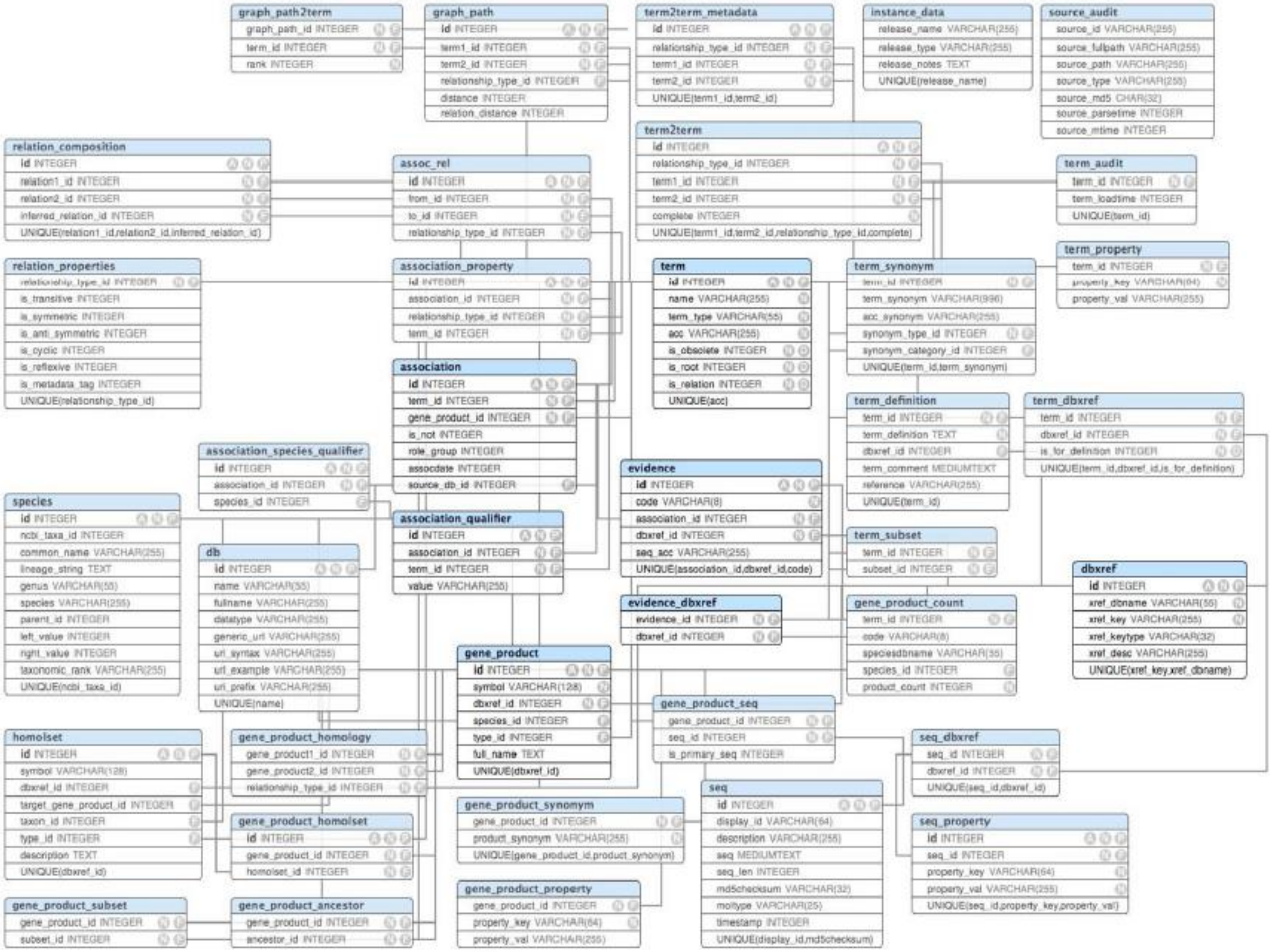
LOYALTY TABLE

ACTION	Session_ID	Status	Membership Change	Loyalty Code	Web ID	EC_ID
LOGIN_ATTEMPT	2104020178	SUCCESS	FALSE			
HOME	2104020178					37619438
LOYALTY_MARK	2104020178			RW44005	54332, 43800, 66429, 99123, 37455	37619438
PURCHASE		FAILURE		RW44005	54332, 43800, 66429, 99123, 37455	37619438
PURCHASE		SUCCESS			54332, 43800, 66429, 99123, 37455	37619438
REWARDS_CATALOGUE _OVERVIEW	2104020178			RW44005	12367, 23876, 11675, 64555, 65766, 54332, 43800	37619438

SCHEMA DESIGN



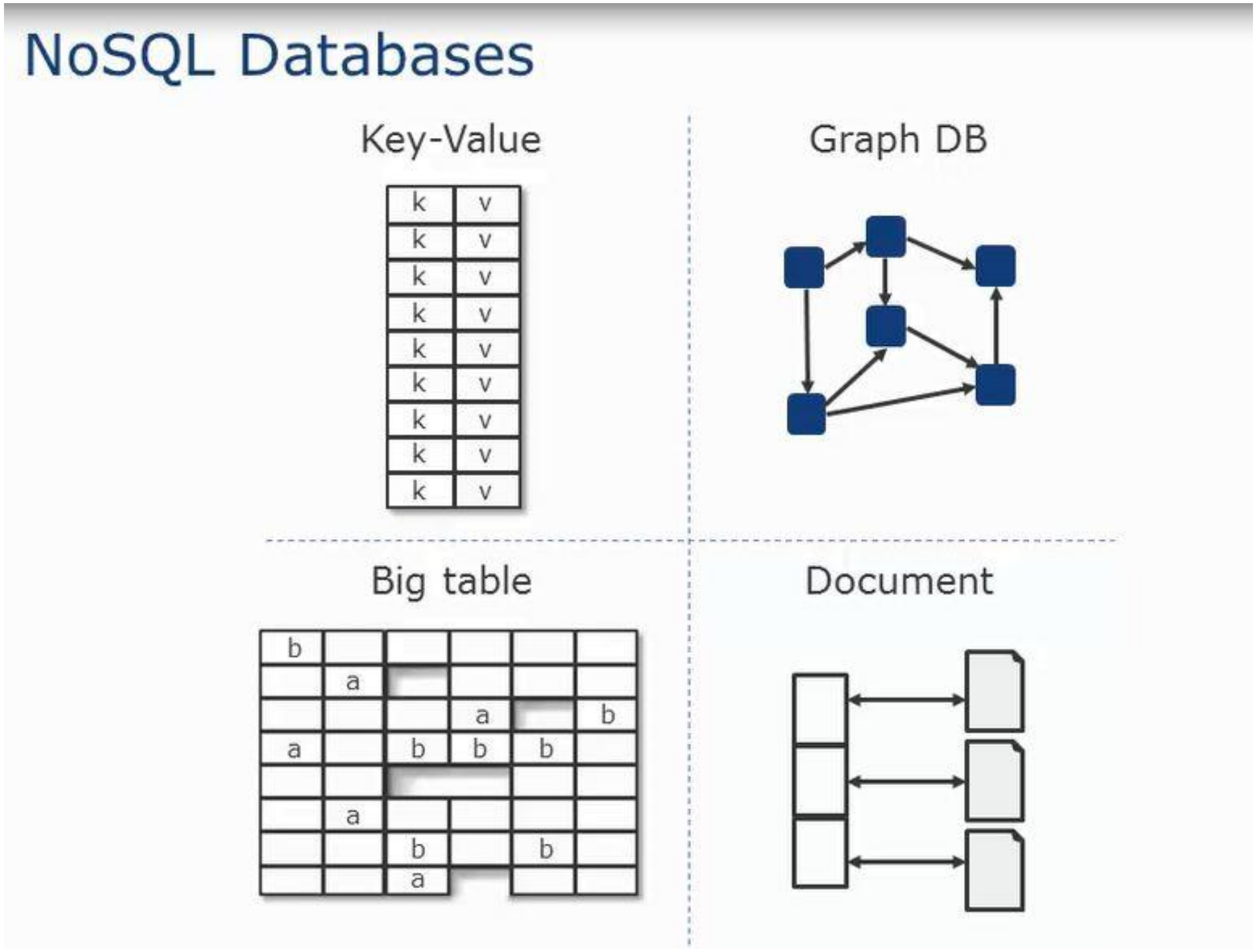
BIG PICTURE


graph_path2term
graph_path_id INTEGER
term_id INTEGER
rank INTEGER
graph_path
id INTEGER
term1_id INTEGER
term2_id INTEGER
relationship_type_id INTEGER
distance INTEGER
relation_distance INTEGER
term2term_metadata
id INTEGER
relationship_type_id INTEGER
term1_id INTEGER
term2_id INTEGER
UNIQUE(term1_id,term2_id)
instance_data
release_name VARCHAR(255)
release_type VARCHAR(255)
release_notes TEXT
UNIQUE(release_name)
source_audit
source_id VARCHAR(255)
source_fullpath VARCHAR(255)
source_path VARCHAR(255)
source_type VARCHAR(255)
source_md5 CHAR(32)
source_parsetime INTEGER
source_mtime INTEGER
relation_composition
id INTEGER
relation1_id INTEGER
relation2_id INTEGER
inferred_relation_id INTEGER
UNIQUE(relation1_id,relation2_id,inferred_relation_id)
assoc_rel
id INTEGER
from_id INTEGER
to_id INTEGER
relationship_type_id INTEGER
term2term
id INTEGER
relationship_type_id INTEGER
term1_id INTEGER
term2_id INTEGER
complete INTEGER
UNIQUE(term1_id,term2_id,relationship_type_id,complete)
term_audit
term_id INTEGER
term_loadtime INTEGER
UNIQUE(term_id)
term_property
term_id INTEGER
property_key VARCHAR(64)
property_val VARCHAR(255)
relation_properties
relationship_type_id INTEGER
is_transitive INTEGER
is_symmetric INTEGER
is_anti_symmetric INTEGER
is_cyclic INTEGER
is_reflexive INTEGER
is_metadata_tag INTEGER
UNIQUE(relationship_type_id)
association_property
id INTEGER
association_id INTEGER
relationship_type_id INTEGER
term_id INTEGER
term
id INTEGER
name VARCHAR(255)
term_type VARCHAR(55)
acc VARCHAR(255)
is_obsolete INTEGER
is_root INTEGER
is_relation INTEGER
UNIQUE(acc)
term_synonym
term_id INTEGER
term_synonym VARCHAR(996)
acc_synonym VARCHAR(255)
synonym_type_id INTEGER
synonym_category_id INTEGER
UNIQUE(term_id,term_synonym)
association
id INTEGER
term_id INTEGER
gene_product_id INTEGER
is_not INTEGER
role_group INTEGER
assocdate INTEGER
source_db_id INTEGER
term_definition
term_id INTEGER
term_definition TEXT
dbxref_id INTEGER
term_comment MEDIUMTEXT
reference VARCHAR(255)
UNIQUE(term_id)
term_dbxref
term_id INTEGER
dbxref_id INTEGER
is_for_definition INTEGER
UNIQUE(term_id,dbxref_id,is_for_definition)
association_species_qualifier
id INTEGER
association_id INTEGER
species_id INTEGER
evidence
id INTEGER
code VARCHAR(8)
association_id INTEGER
dbxref_id INTEGER
seq_acc VARCHAR(255)
UNIQUE(association_id,dbxref_id,code)
species
id INTEGER
ncbi_taxa_id INTEGER
common_name VARCHAR(255)
lineage_string TEXT
genus VARCHAR(55)
species VARCHAR(255)
parent_id INTEGER
left_value INTEGER
right_value INTEGER
taxonomic_rank VARCHAR(255)
UNIQUE(ncbi_taxa_id)
association_qualifier
id INTEGER
association_id INTEGER
term_id INTEGER
value VARCHAR(255)
term_subset
term_id INTEGER
subset_id INTEGER
db
id INTEGER
name VARCHAR(55)
fullname VARCHAR(255)
datatype VARCHAR(255)
generic_url VARCHAR(255)
url_syntax VARCHAR(255)
url_example VARCHAR(255)
uri_prefix VARCHAR(255)
UNIQUE(name)
dbxref
id INTEGER
xref_dbname VARCHAR(55)
xref_key VARCHAR(255)
xref_keytype VARCHAR(32)
xref_desc VARCHAR(255)
UNIQUE(xref_key,xref_dbname)
evidence_dbxref
evidence_id INTEGER
dbxref_id INTEGER
gene_product_count
term_id INTEGER
code VARCHAR(8)
speciesdbname VARCHAR(55)
species_id INTEGER
product_count INTEGER
gene_product
id INTEGER
symbol VARCHAR(128)
dbxref_id INTEGER
species_id INTEGER
type_id INTEGER
full_name TEXT
UNIQUE(dbxref_id)
gene_product_seq
gene_product_id INTEGER
seq_id INTEGER
is_primary_seq INTEGER
homolset
id INTEGER
symbol VARCHAR(128)
dbxref_id INTEGER
target_gene_product_id INTEGER
taxon_id INTEGER
type_id INTEGER
description TEXT
UNIQUE(dbxref_id)
gene_product_homology
gene_product1_id INTEGER
gene_product2_id INTEGER
relationship_type_id INTEGER
seq_dbxref
seq_id INTEGER
dbxref_id INTEGER
UNIQUE(seq_id,dbxref_id)
seq
id INTEGER
display_id VARCHAR(64)
description VARCHAR(255)
seq MEDIUMTEXT
seq_len INTEGER
md5checksum VARCHAR(32)
moltype VARCHAR(25)
timestamp INTEGER
UNIQUE(display_id,md5checksum)
gene_product_synonym
gene_product_id INTEGER
product_synonym VARCHAR(255)
UNIQUE(gene_product_id,product_synonym)
gene_product_homolset
id INTEGER
gene_product_id INTEGER
homolset_id INTEGER
seq_property
id INTEGER
seq_id INTEGER
property_key VARCHAR(64)
property_val VARCHAR(255)
UNIQUE(seq_id,property_key,property_val)
gene_product_property
gene_product_id INTEGER
property_key VARCHAR(64)
property_val VARCHAR(255)
gene_product_subset
gene_product_id INTEGER
subset_id INTEGER
gene_product_ancestor
gene_product_id INTEGER
ancestor_id INTEGER

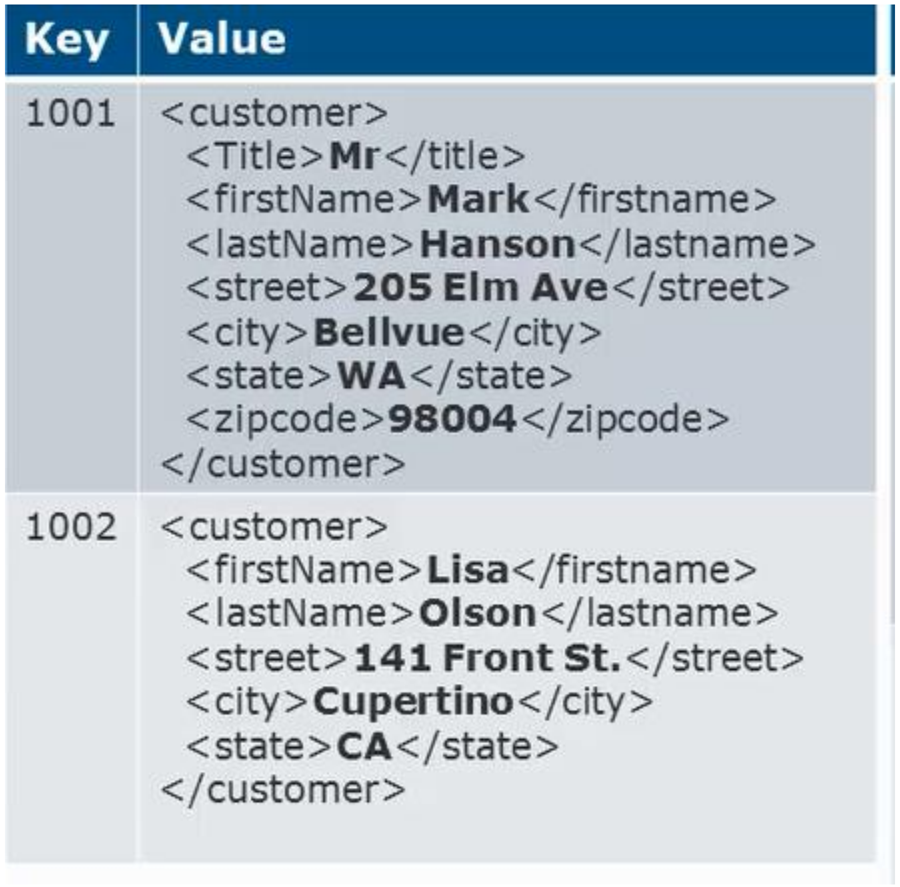
RDBMS Challenges

- Structured Data – Static Schema
- Slow for – Iterative Development
- Non-Application Specific - Normalized

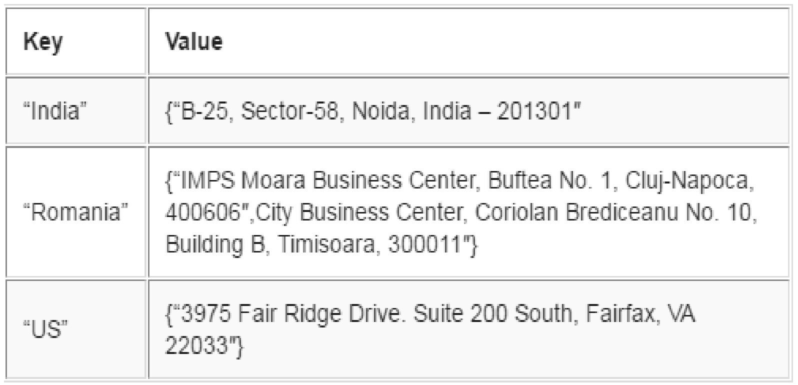
NoSQL DATABASES



KEY VALUE STORES



Key	Value
1001	<customer> <Title>**Mr**</title> <firstName>**Mark**</firstname> <lastName>**Hanson**</lastname> <street>**205 Elm Ave**</street> <city>**Bellvue**</city> <state>**WA**</state> <zipcode>**98004**</zipcode> </customer>
1002	<customer> <firstName>**Lisa**</firstname> <lastName>**Olson**</lastname> <street>**141 Front St.**</street> <city>**Cupertino**</city> <state>**CA**</state> </customer>



Key	Value
"India"	{"B-25, Sector-58, Noida, India – 201301"
"Romania"	{"IMPS Moara Business Center, Buftea No. 1, Cluj-Napoca, 400606",City Business Center, Coriolan Brediceanu No. 10, Building B, Timisoara, 300011"}
"US"	{"3975 Fair Ridge Drive. Suite 200 South, Fairfax, VA 22033"}

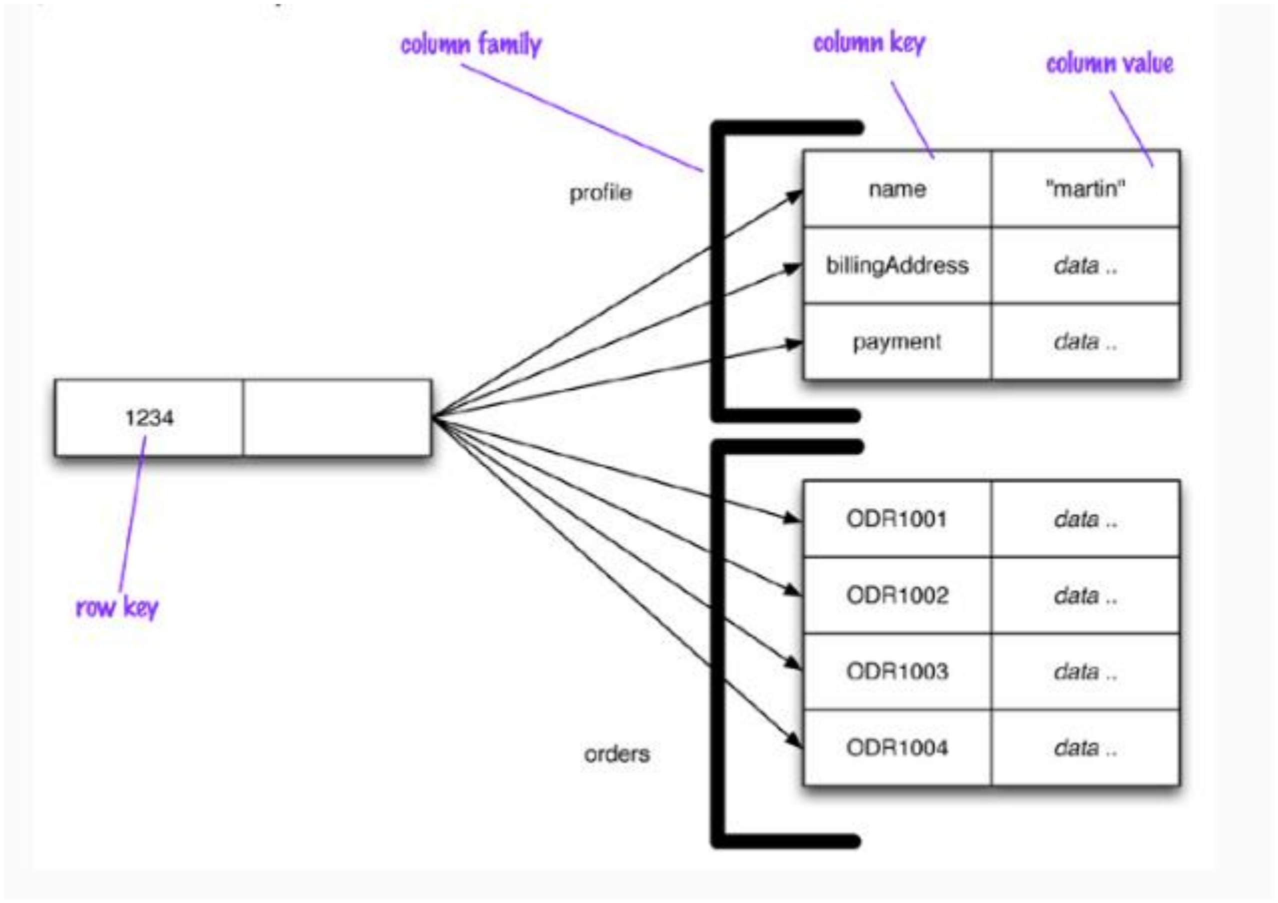
```
void Put(string key, byte[] data);
byte[] Get(string key);
void Remove(string key);
```

KEY VALUE STORES - APPLICATIONS

- Ecommerce sites
  - User sessions
  - Shopping carts

- Ex:- Amazon Dynamo, Redis, Basho Riak, Aerospike

COLUMN FAMILIES



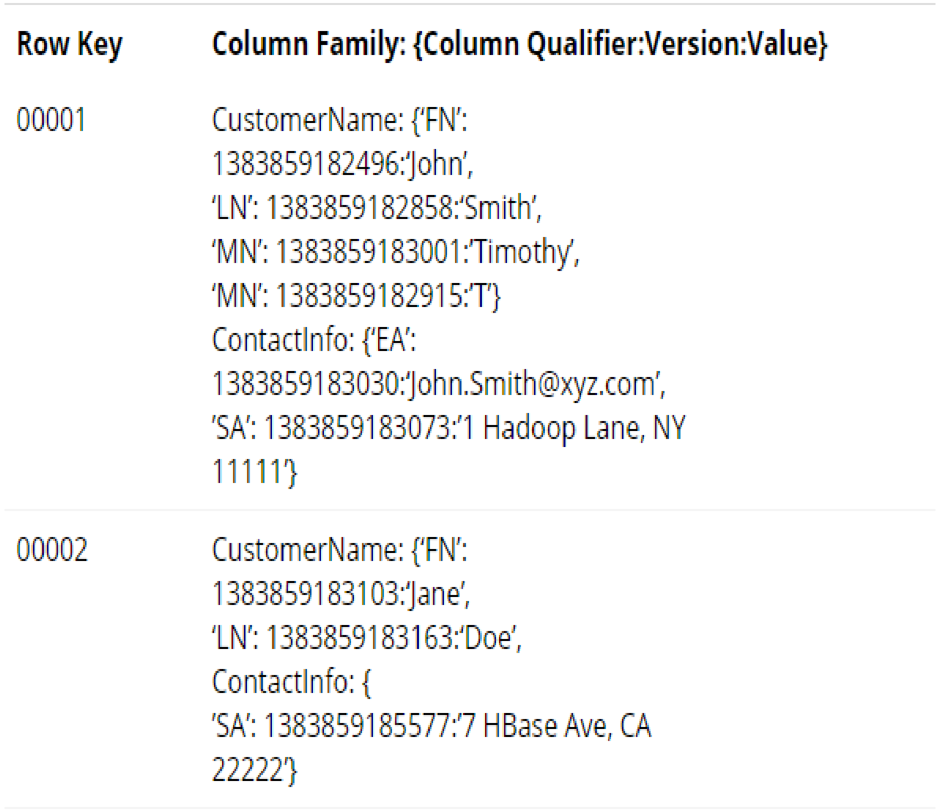
COLUMN FAMILIES - APPLICATIONS

- **Terms Used: -**
  - **Keyspaces, Column Families, Column Key &**
  - **Column Values**

- **Column families, wide and skinny**

- **Ex: - Cassandra, Amazon Dynamo DB, HBase**

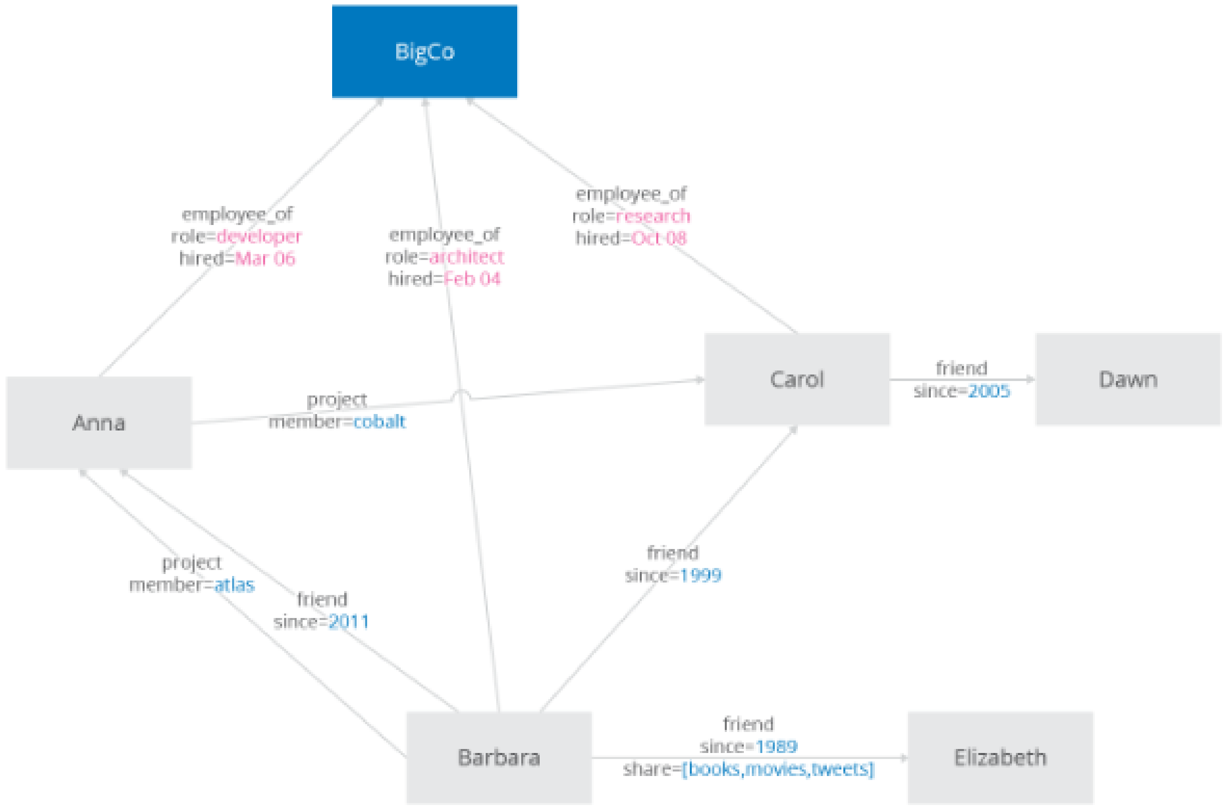
HBASE – COLUMN FAMILY



Row Key	Column Family: {Column Qualifier:Version:Value}
00001	CustomerName: {'FN': 1383859182496:'John', 'LN': 1383859182858:'Smith', 'MN': 1383859183001:'Timothy', 'MN': 1383859182915:'T'} ContactInfo: {'EA': 1383859183030:'John.Smith@xyz.com', 'SA': 1383859183073:'1 Hadoop Lane, NY 11111'}
00002	CustomerName: {'FN': 1383859183103:'Jane', 'LN': 1383859183163:'Doe', ContactInfo: { 'SA': 1383859185577:'7 HBase Ave, CA 22222'}

- Built on top of Google's Big Table
- Fast Random read or write access
- Can scale horizontally

GRAPH DATABASES



BigCo
employee_of role=developer hired=Mar 06
employee_of role=architect hired=Feb 04
employee_of role=research hired=Oct 08
Anna
project member=cobalt
Carol
friend since=2005
Dawn
project member=atlas
friend since=2011
friend since=1999
Barbara
friend since=1989 share=[books,movies,tweets]
Elizabeth

GRAPH DATABASE – Terminology & Application

- Entities & Relationships
  - Entities => Nodes
  - Relationships => Edges
  - Edges have Directions

- Applications: -
  - Social Networks
  - Fraud Detection
  - Network Configuration

- Ex:- NeoJ, Infinite Graph

DOCUMENT DATABASES

Document 1

```
{
  "id": "1",
  “name": "John Smith",
  "isActive": true,
  "dob": "1964-30-08"
}
```

Document 2

```
{
  "id": "2",
  “fullName": "Sarah Jones",
  “isActive": false,
  "dob": "2002-02-18"
}
```

Document 3

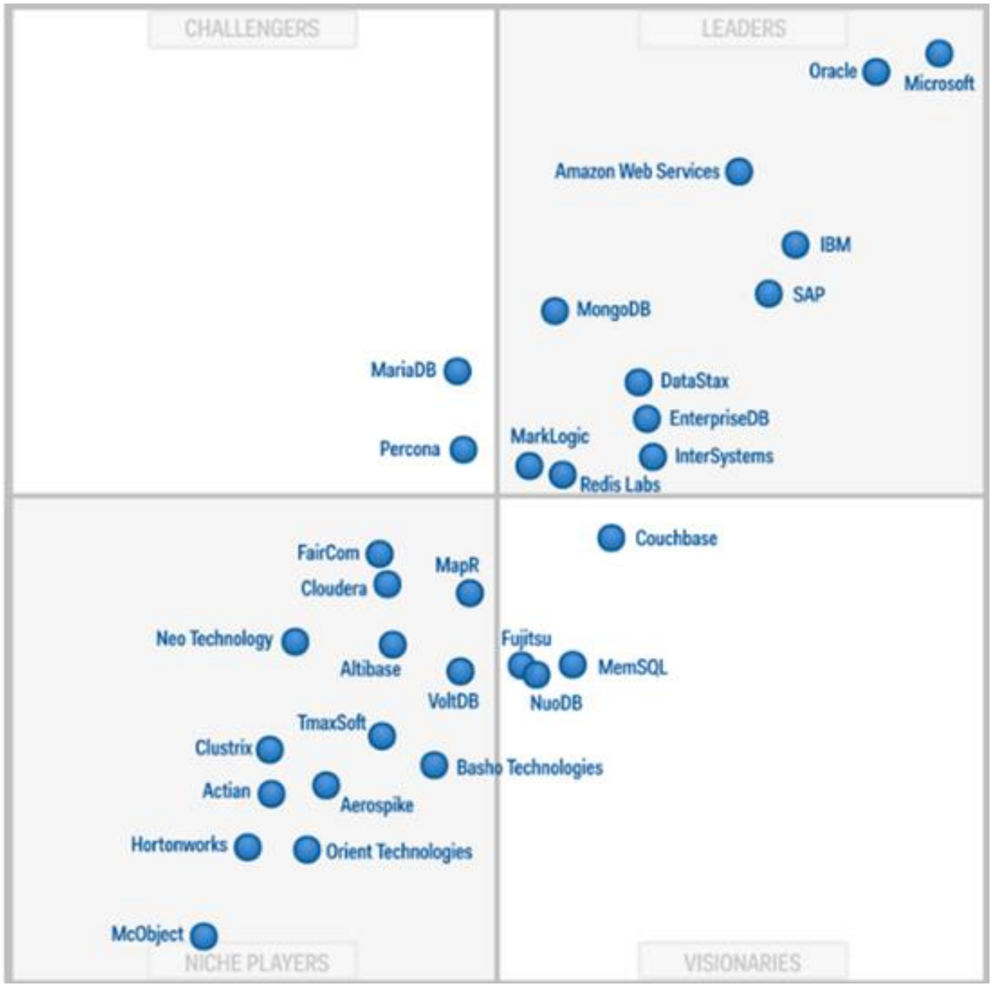
```
{
  "id": "3",
  “fullName":
  {
    "first": "Adam",
    "last": "Stark"
  },
  "isActive": true,
  "dob": "2015-04-19"
}
```

DOCUMENT DATABASES

- Based on JSON
- Schema-less

- MongoDB, CouchDB, OrientDB

RISE OF NOSQL



HBASE - INTRODUCTION

Column Family

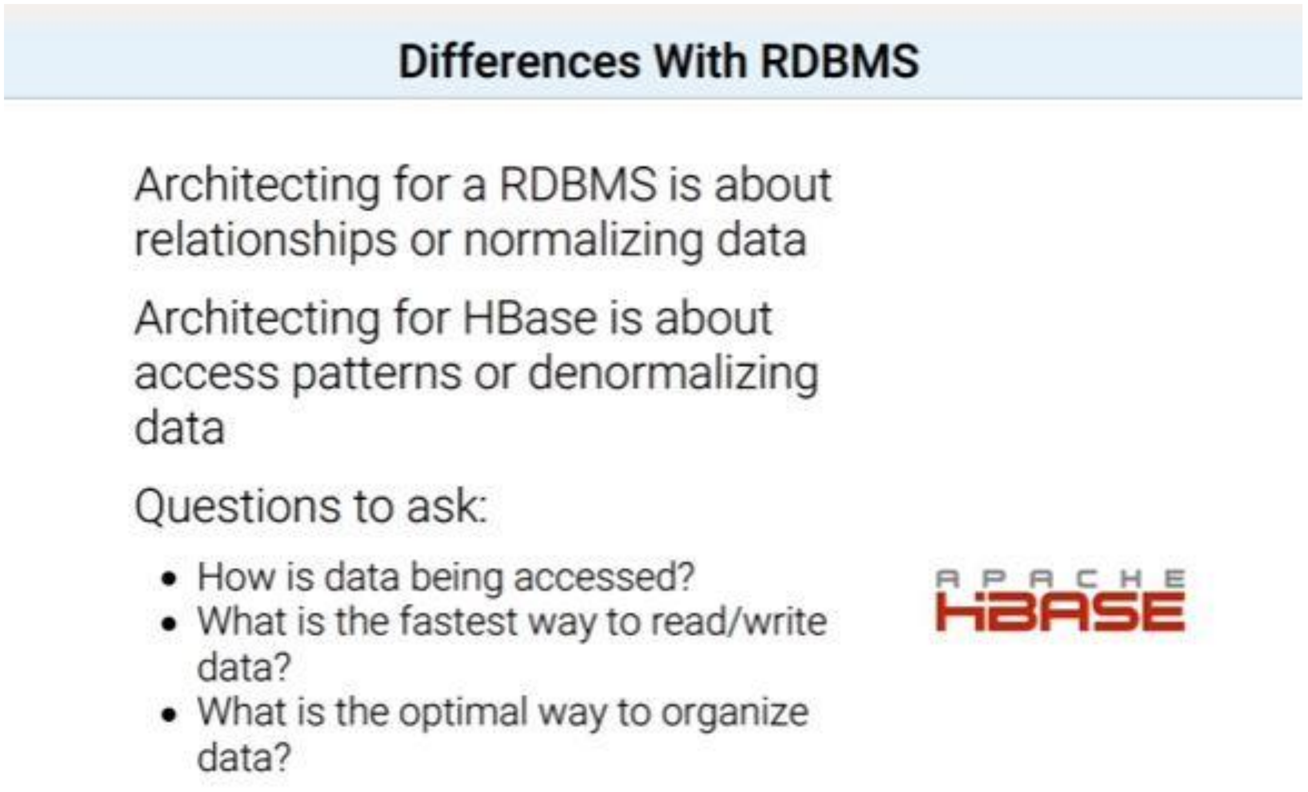
Google's Big Table

HBASE CLIENTS

Here is a very limited list of well known names

- Facebook
- Adobe
- Twitter
- Yahoo!
- Netflix
- Meetup
- Stumbleupon
- You????

HBASE - Uses



Differences With RDBMS

Architecting for a RDBMS is about relationships or normalizing data

Architecting for HBase is about access patterns or denormalizing data

Questions to ask:

- How is data being accessed?
- What is the fastest way to read/write data?
- What is the optimal way to organize data?

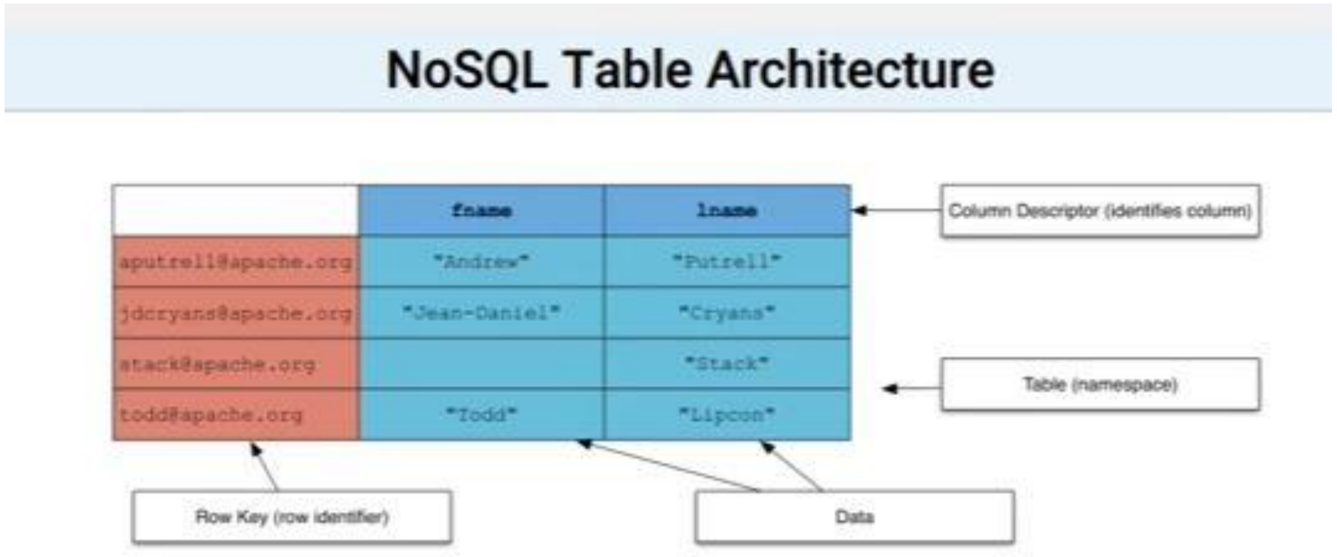
APACHE HBASE

HBASE - USEFULNESS

- RANDOM SEARCHES
- RANGE ACCESS BY KEY
- GOOD FOR VARIABLE SCHEMA

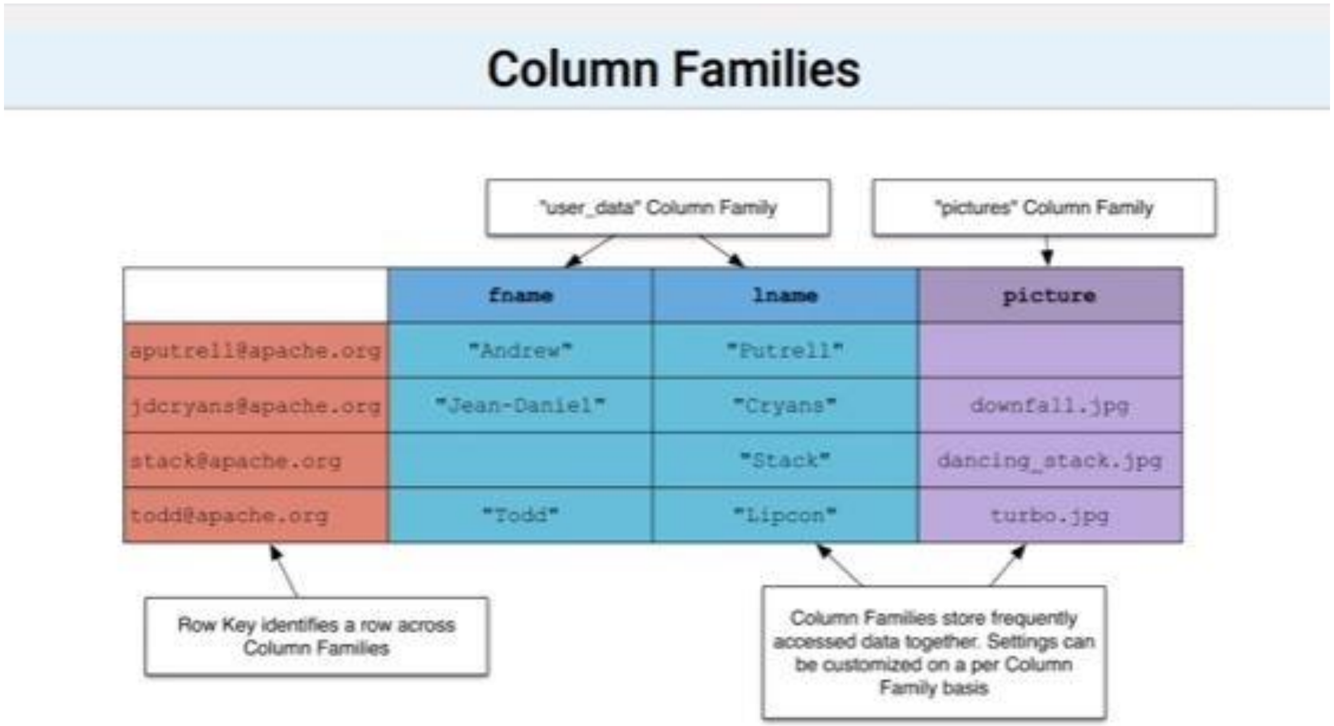
TERADATA.

HBASE – NoSQL


NoSQL Table Architecture

	fname	lname
aputrell@apache.org	"Andrew"	"Putrell"
jdcryans@apache.org	"Jean-Daniel"	"Cryans"
stack@apache.org		"Stack"
todd@apache.org	"Todd"	"Lipcon"

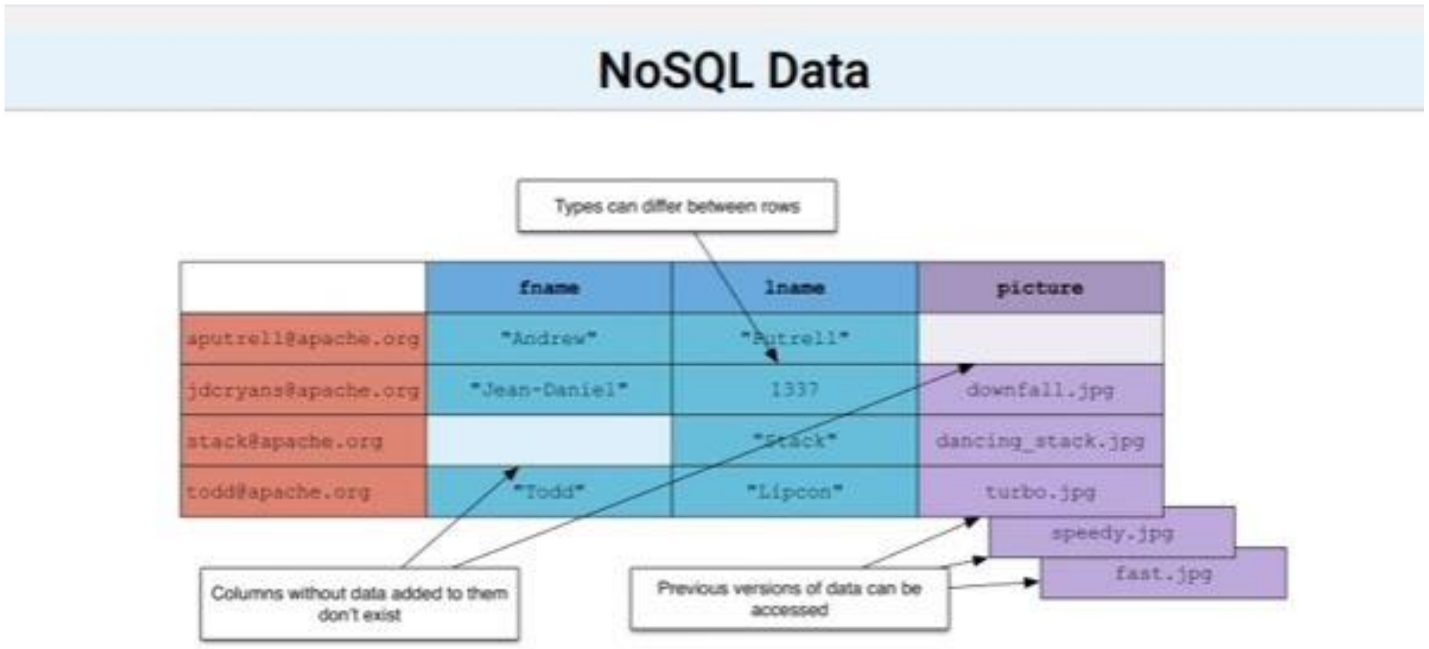
HBASE – NoSQL


Column Families

	fname	lname	picture
aputrell@apache.org	"Andrew"	"Putrell"	
jdcryans@apache.org	"Jean-Daniel"	"Cryans"	downfall.jpg
stack@apache.org		"Stack"	dancing_stack.jpg
todd@apache.org	"Todd"	"Lipcon"	turbo.jpg

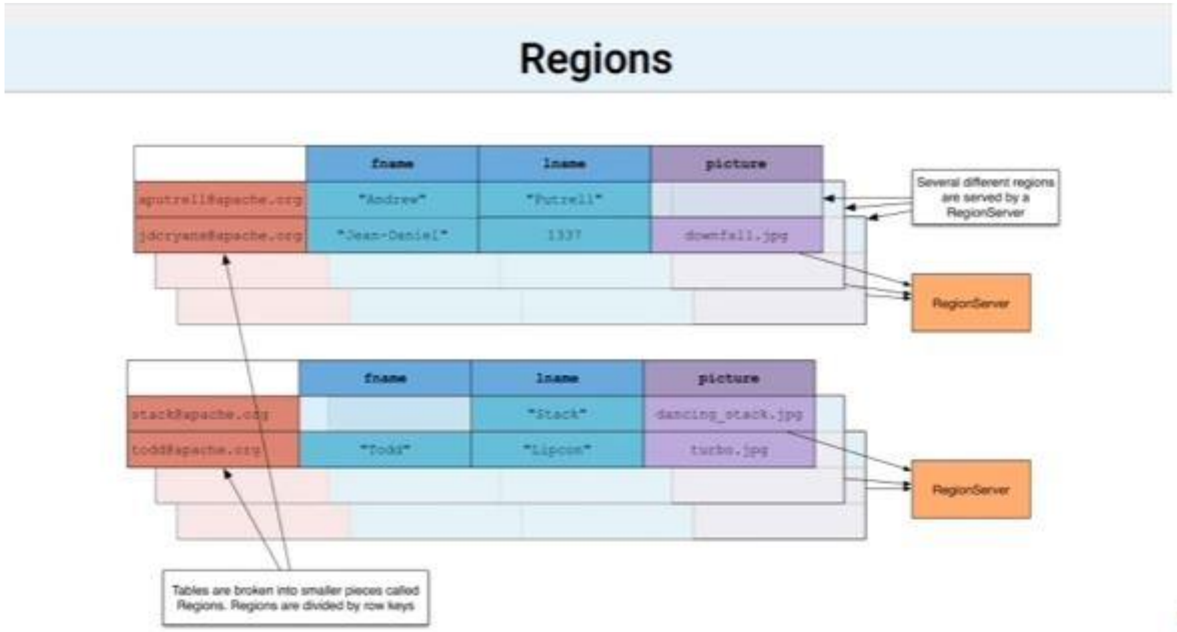
"user_data" Column Family
"pictures" Column Family
Row Key identifies a row across Column Families
Column Families store frequently accessed data together. Settings can be customized on a per Column Family basis

HBASE – NoSQL


NoSQL Data
Types can differ between rows
Columns without data added to them don't exist
Previous versions of data can be accessed
speedy.jpg
fast.jpg

	fname	lname	picture
aputrell@apache.org	"Andrew"	"Putrell"	
jdcryans@apache.org	"Jean-Daniel"	1337	downfall.jpg
stack@apache.org		"Stack"	dancing_stack.jpg
todd@apache.org	"Todd"	"Lipcon"	turbo.jpg

HBASE – Regions



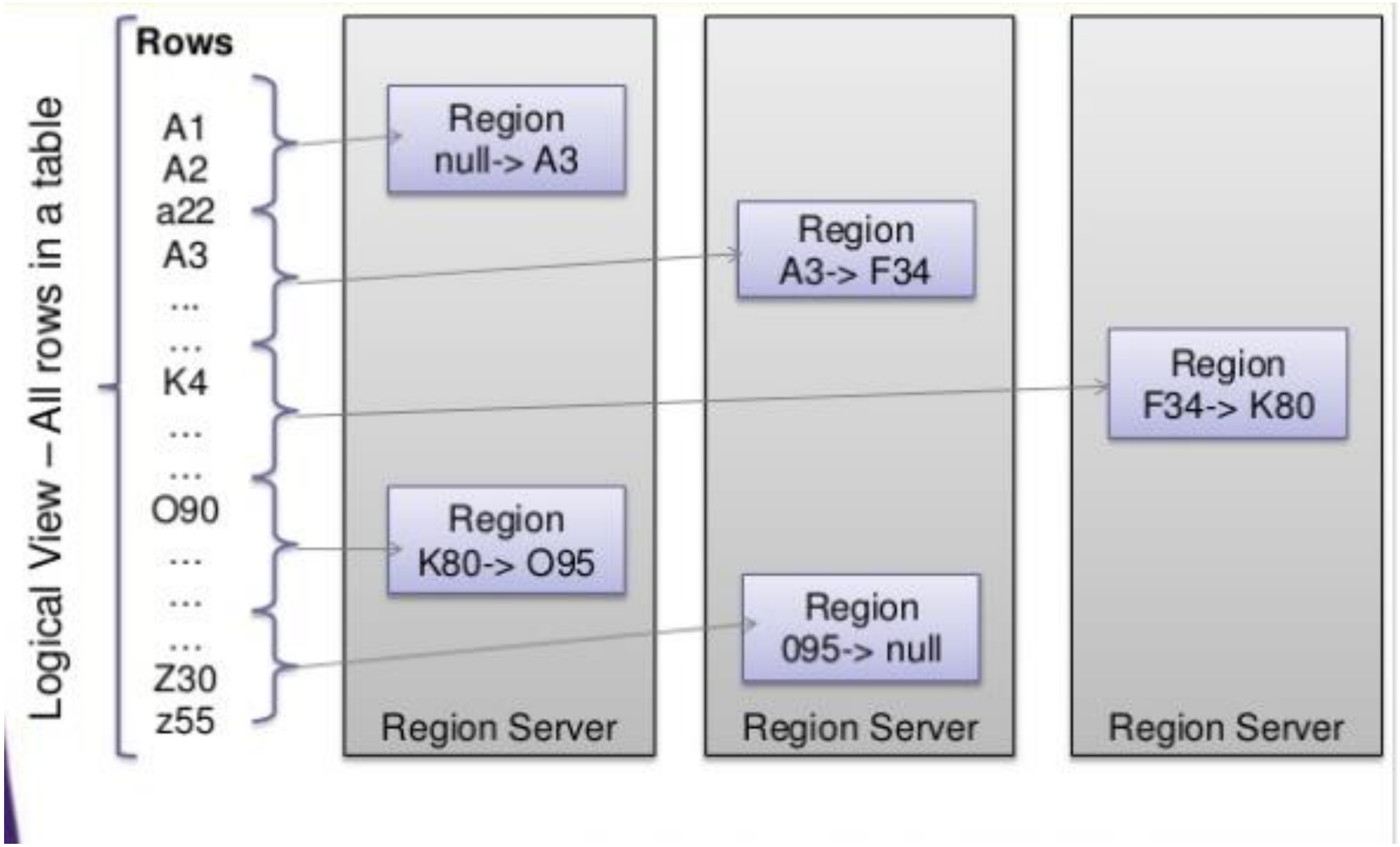
HBASE – IN SHORT

- Table is a collection of rows.
- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.

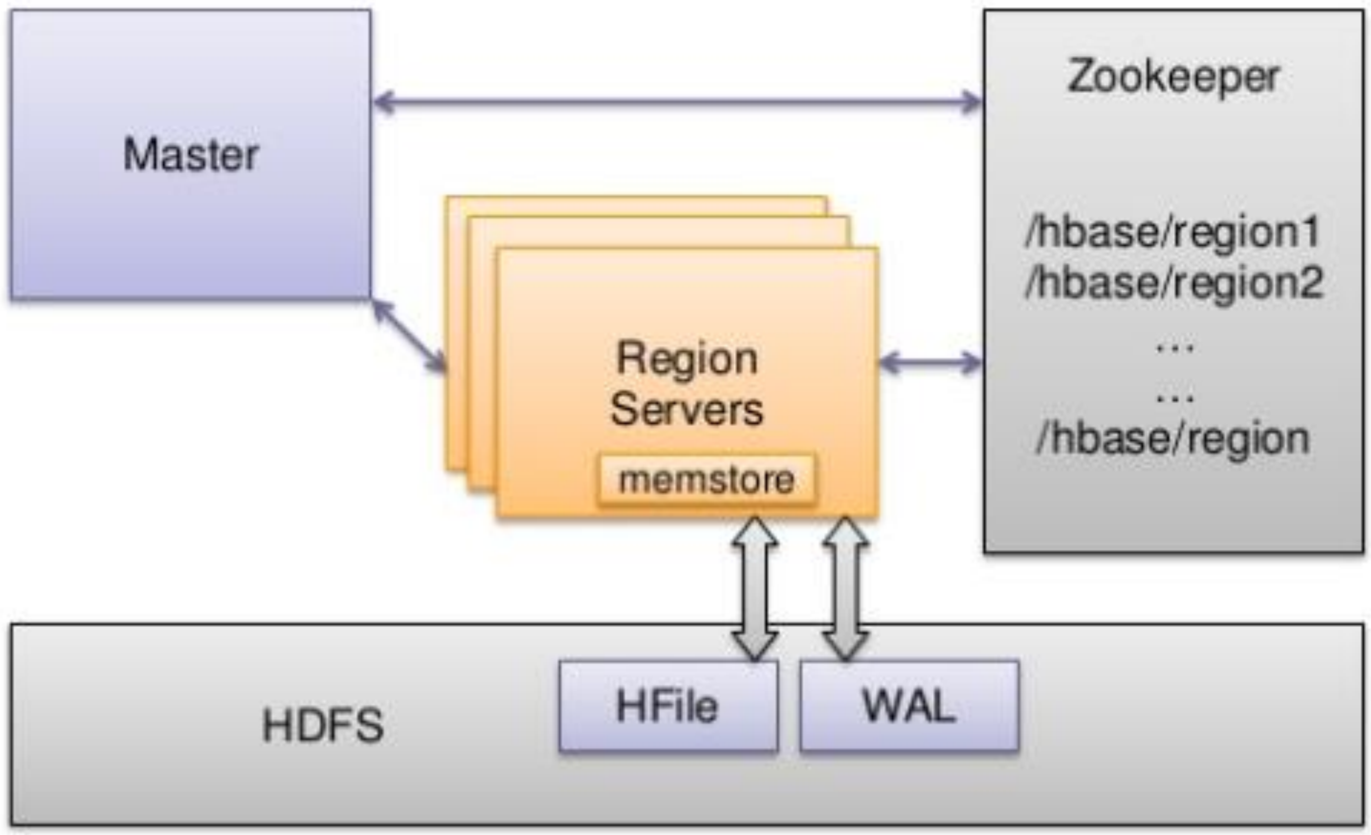
O H &

- Each cell value of the table has a timestamp

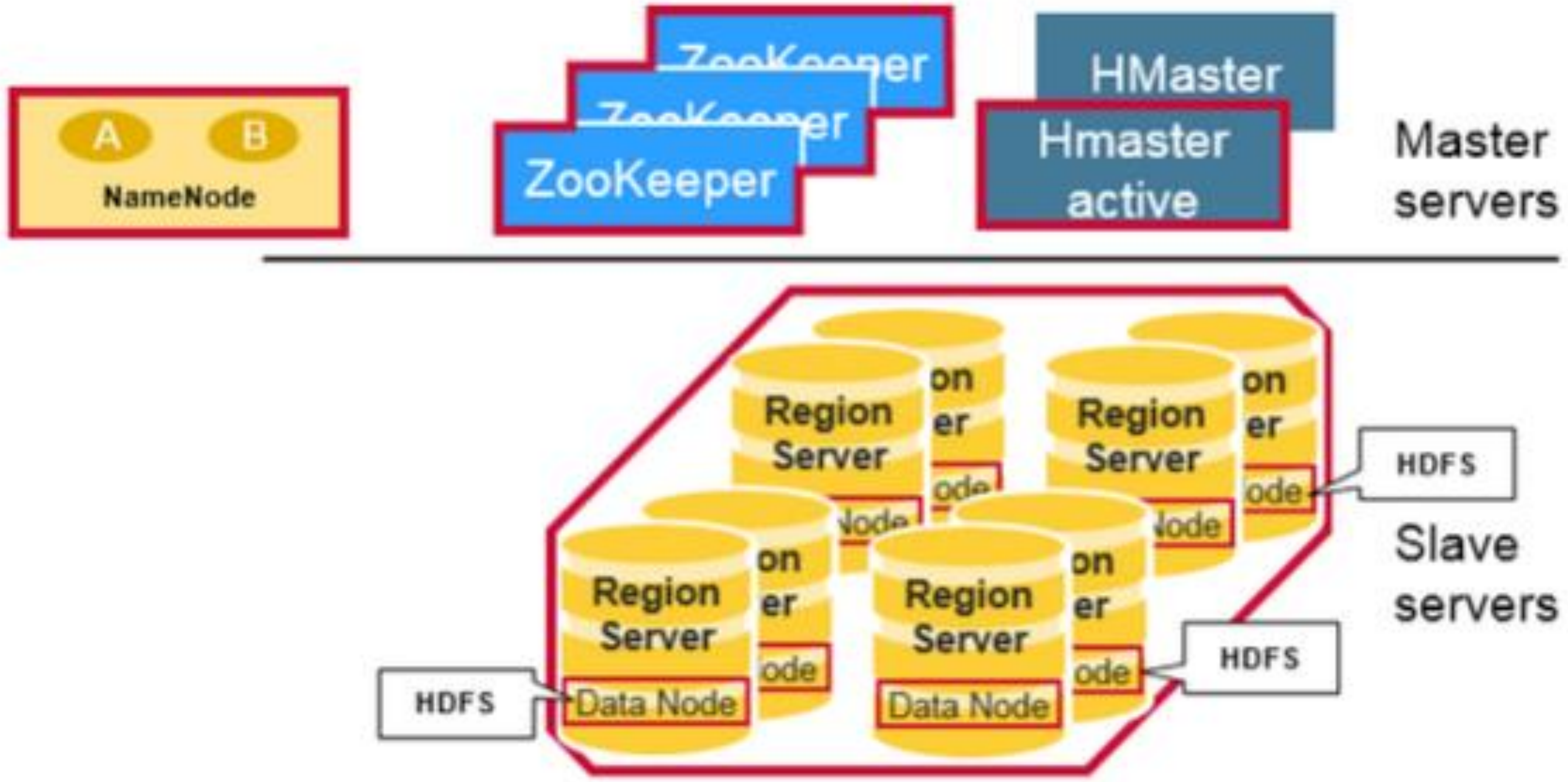
HBASE – REGION SERVER

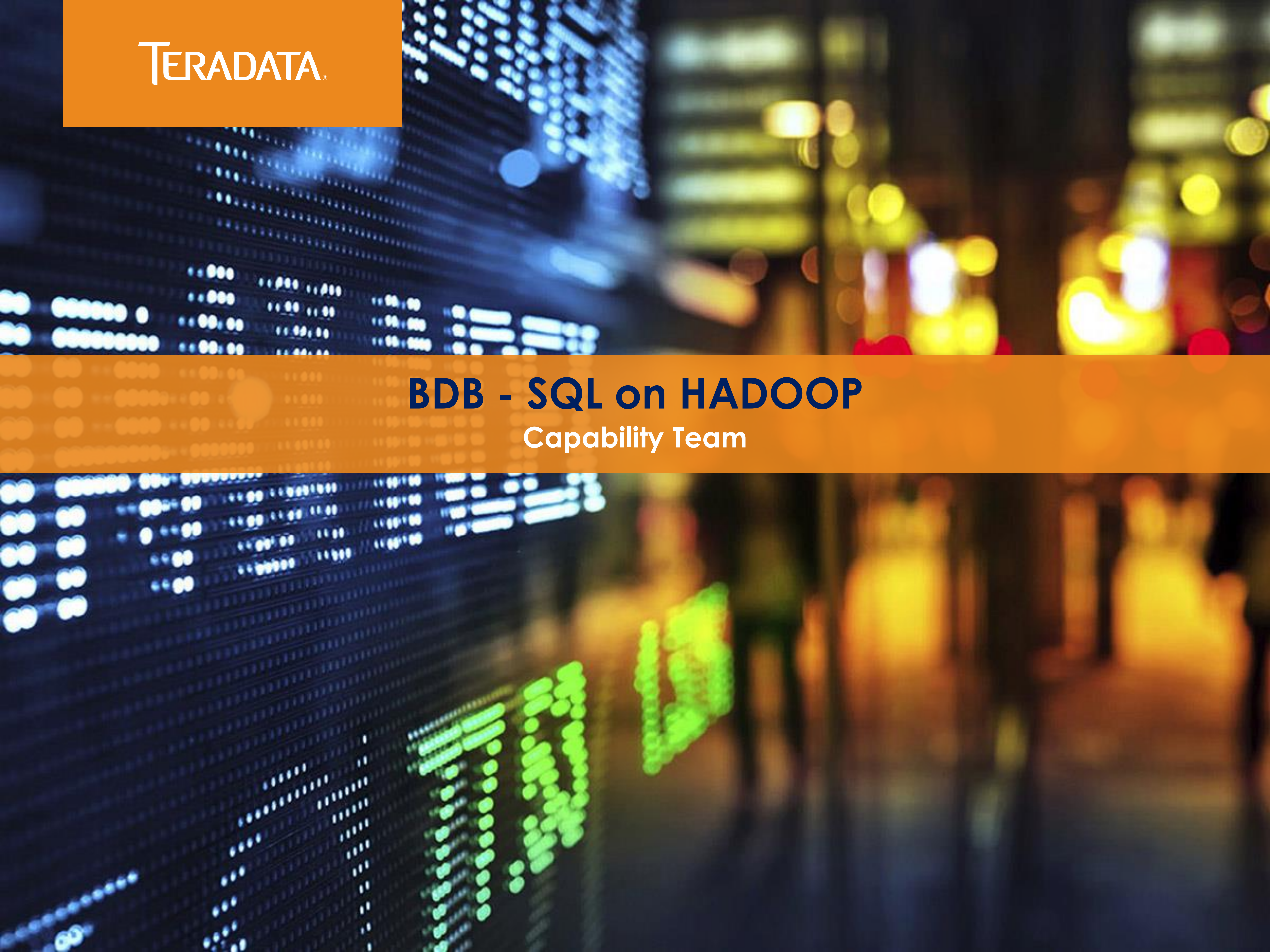


HBASE – COMPONENTS



HBASE – HIGH LEVEL





BDB - SQL on HADOOP

Capability Team

10 WAYS or so ..

Apache Hive

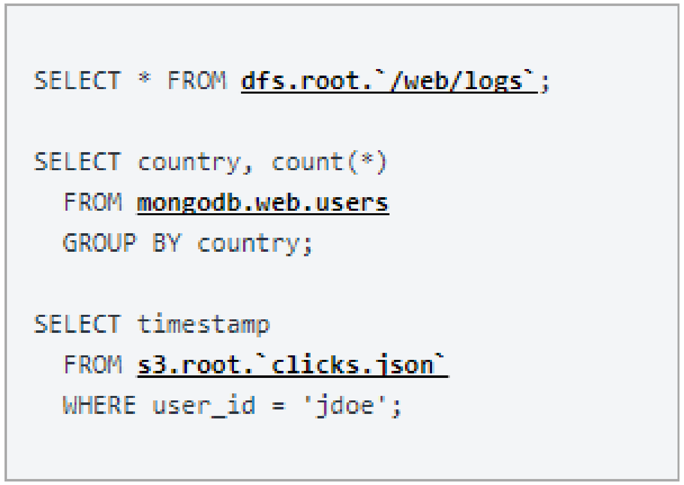
- Imposes structure on a variety of data formats
- Access to files HDFS
- Procedural Language with HPL-SQL

10 WAYS or so ..

STINGER INITIATIVE

- HIVE ++: -
  - More SQL support
  - Analytical features like OVER in WHERE clause
  - Introduced TEZ (runtime framework)
  - Directed Acyclic Graph (D.A.G)

10 WAYS or so ..



```
SELECT * FROM dfs.root.`/web/logs`;

SELECT country, count(*)
  FROM mongodb.web.users
  GROUP BY country;

SELECT timestamp
  FROM s3.root.`clicks.json`
  WHERE user_id = 'jdoe';
```

- Inspired b y Google's Drummel
- Supported by MapR
- Defines Schema on the Go supports ANSI SQL 2003
- Query multiple data stores: -
  - NoSQL (MongoDB)
  - HDFS (Hive)
- Good for BI

10 WAYS or so ..

- Real Time In Memory parallelized SQL-on-Hadoop engine
- Advanced analytics from: -
  - Stream Processing
  - Machine Learning
- Subset of SQL functionality
- Code in Python, Java or Scala

10 WAYS or so ..

cloudera
IMPALA

- Supports High concurrency workloads
- Compatible with BI Tools for Analytics
- Supported by Cloudera
- ANSI SQL Support

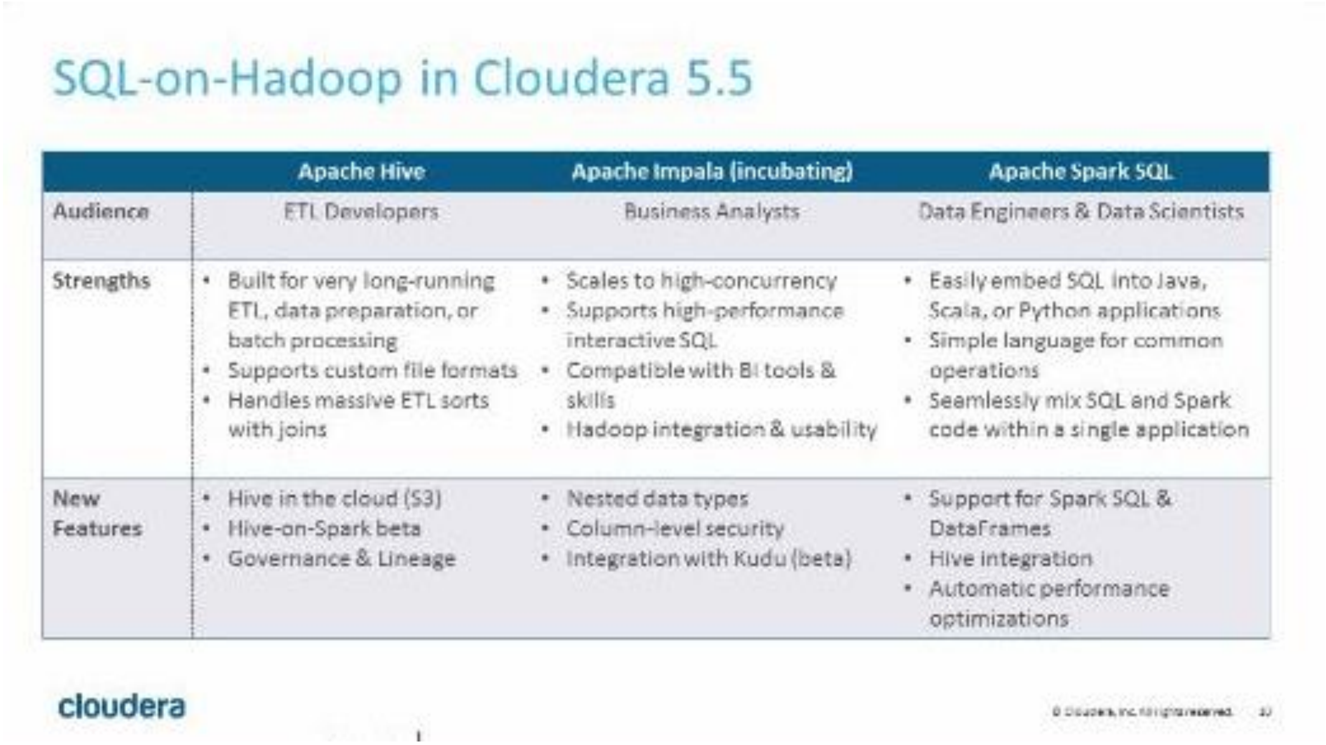
10 WAYS or so ..

presto

- In Memory SQL Engine
- Developed by Facebook
- Interactive Queries
- Optimized for Star Schema Joins

TERADATA.

HOW THEY STACK UP



SQL-on-Hadoop in Cloudera 5.5

	Apache Hive	Apache Impala (incubating)	Apache Spark SQL
Audience	ETL Developers	Business Analysts	Data Engineers & Data Scientists
Strengths	• Built for very long-running ETL, data preparation, or batch processing • Supports custom file formats • Handles massive ETL sorts with joins	• Scales to high-concurrency • Supports high-performance interactive SQL • Compatible with BI tools & skills • Hadoop integration & usability	• Easily embed SQL into Java, Scala, or Python applications • Simple language for common operations • Seamlessly mix SQL and Spark code within a single application
New Features	• Hive in the cloud (S3) • Hive-on-Spark beta • Governance & Lineage	• Nested data types • Column-level security • Integration with Kudu (beta)	• Support for Spark SQL & DataFrames • Hive integration • Automatic performance optimizations

cloudera

TERADATA.

SQL on HADOOP - Comparisons

BEST OF HIVE	BEST OF PRESTO
Large data aggregations	Interactive queries (where you want to wait for the answer)
Large Fact-to-Fact joins	Quickly exploring the data (e.g. what types of records are found in the table)
Large distincts (aka de-duplication jobs)	Joins with a large Fact table and many smaller Dimension tables
Batch jobs that can be scheduled	

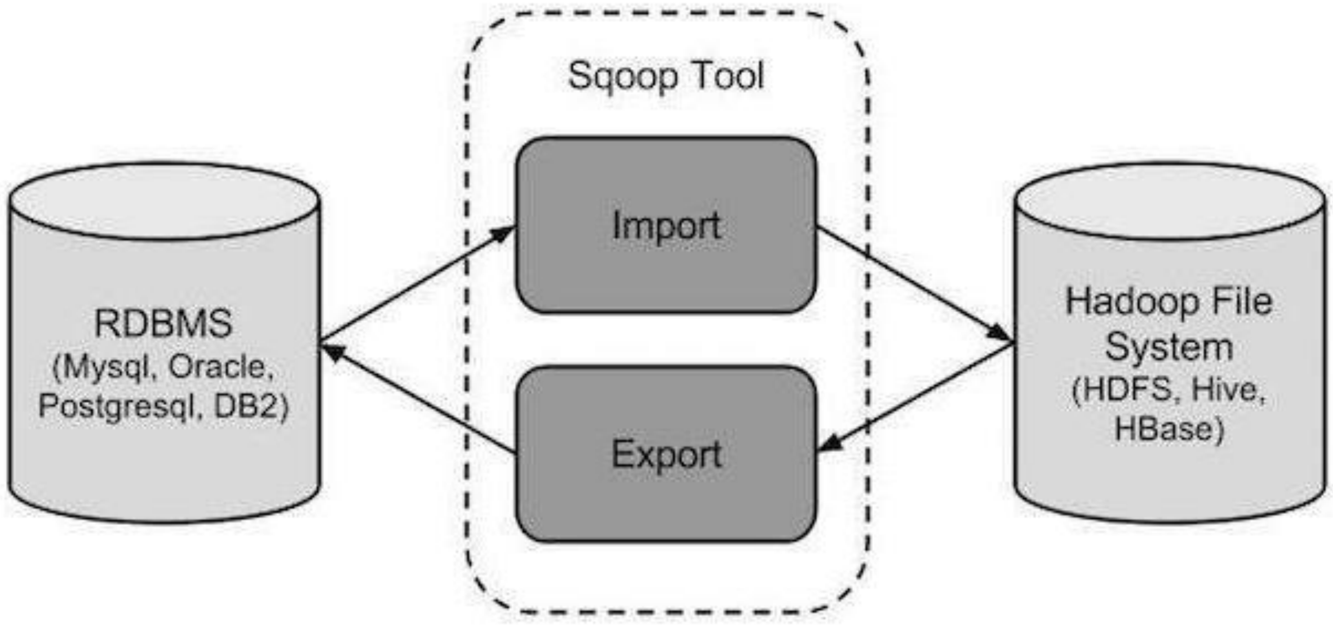
	HIVE	PRESTO
Optimized for	Throughput	Interactivity
SQL Standardized fidelity	HiveQL (subset of common data warehousing SQL)	Designed to comply with ANSI SQL
Window functions	Yes	Yes
Large JOINs	Very good for large Fact-to-Fact joins	Optimized for star schema joins (1 large Fact table and many smaller dimension tables)

BDB – SQOOP

Capability Team

- Transfer Data between RDBMS's to Hadoop
- Bulk Transfer
- 2 way transfer

SCOOP ARCHITECTURE



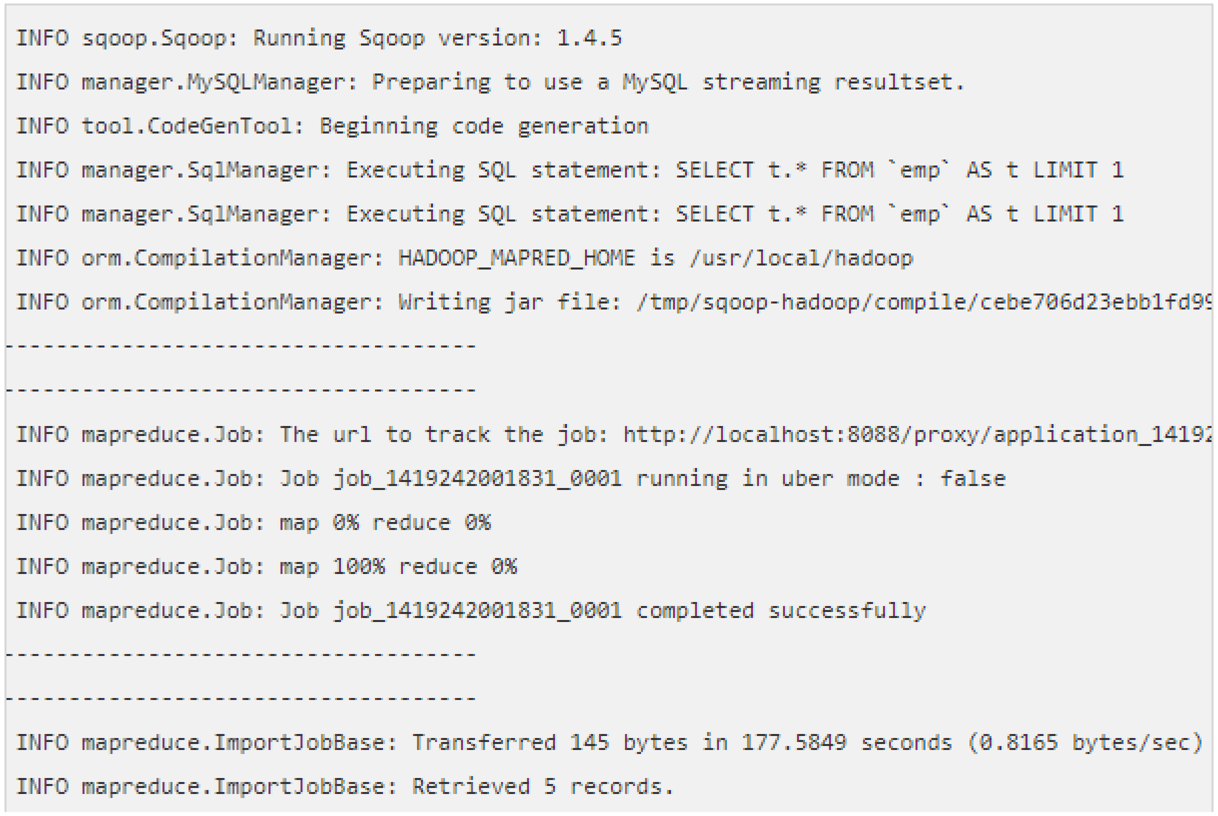
SCOOP IMPORT

```
$ sqoop import (generic-args) (import-args)
$ sqoop-import (generic-args) (import-args)
```

id	name	deg	salary	dept
1201	gopal	manager	50,000	TP
1202	manisha	Proof reader	50,000	TP
1203	khalil	php dev	30,000	AC
1204	prasanth	php dev	30,000	AC
1204	kranthi	admin	20,000	TP

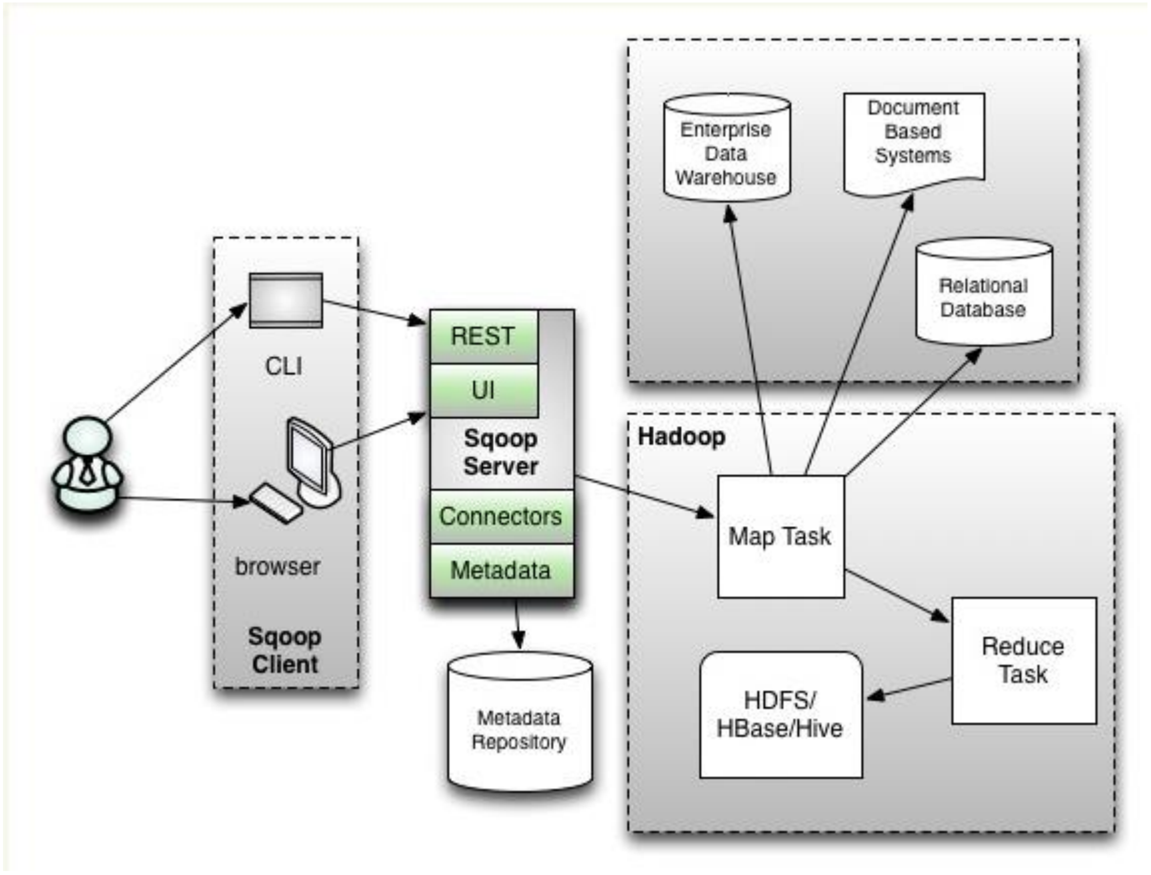
```
$ sqoop import \
--connect jdbc:<db-name>://localhost/userdb \
--username root \
--table emp
--m 1
```

SCOOP – IMPORT RESULT



```
INFO sqoop.Sqoop: Running Sqoop version: 1.4.5
INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
INFO tool.CodeGenTool: Beginning code generation
INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `emp` AS t LIMIT 1
INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `emp` AS t LIMIT 1
INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/hadoop
INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-hadoop/compile/cebe706d23ebb1fd9
------------------------------------
------------------------------------
INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1419
INFO mapreduce.Job: Job job_1419242001831_0001 running in uber mode : false
INFO mapreduce.Job: map 0% reduce 0%
INFO mapreduce.Job: map 100% reduce 0%
INFO mapreduce.Job: Job job_1419242001831_0001 completed successfully
------------------------------------
------------------------------------
INFO mapreduce.ImportJobBase: Transferred 145 bytes in 177.5849 seconds (0.8165 bytes/sec)
INFO mapreduce.ImportJobBase: Retrieved 5 records.
```

SCOOP - II



TERADATA.

SCOOP IMPORT ARCHITECTURE

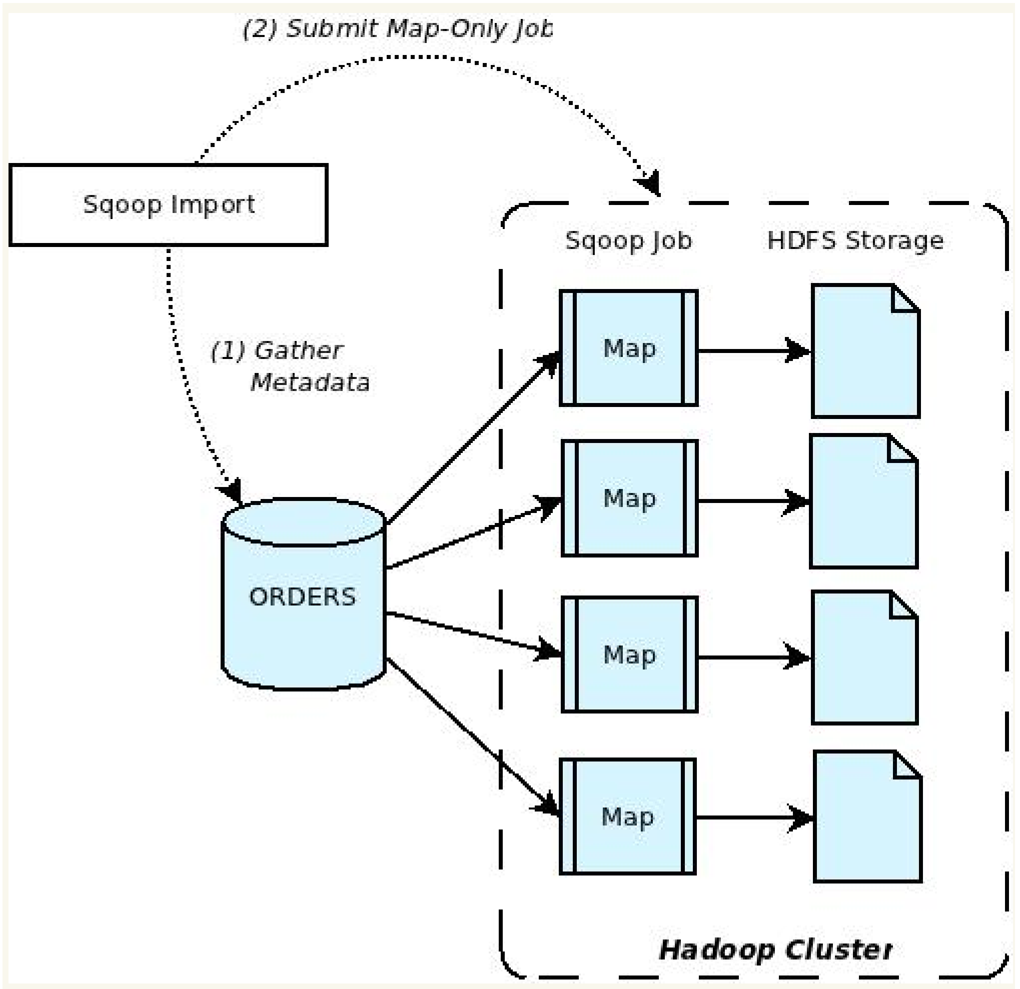
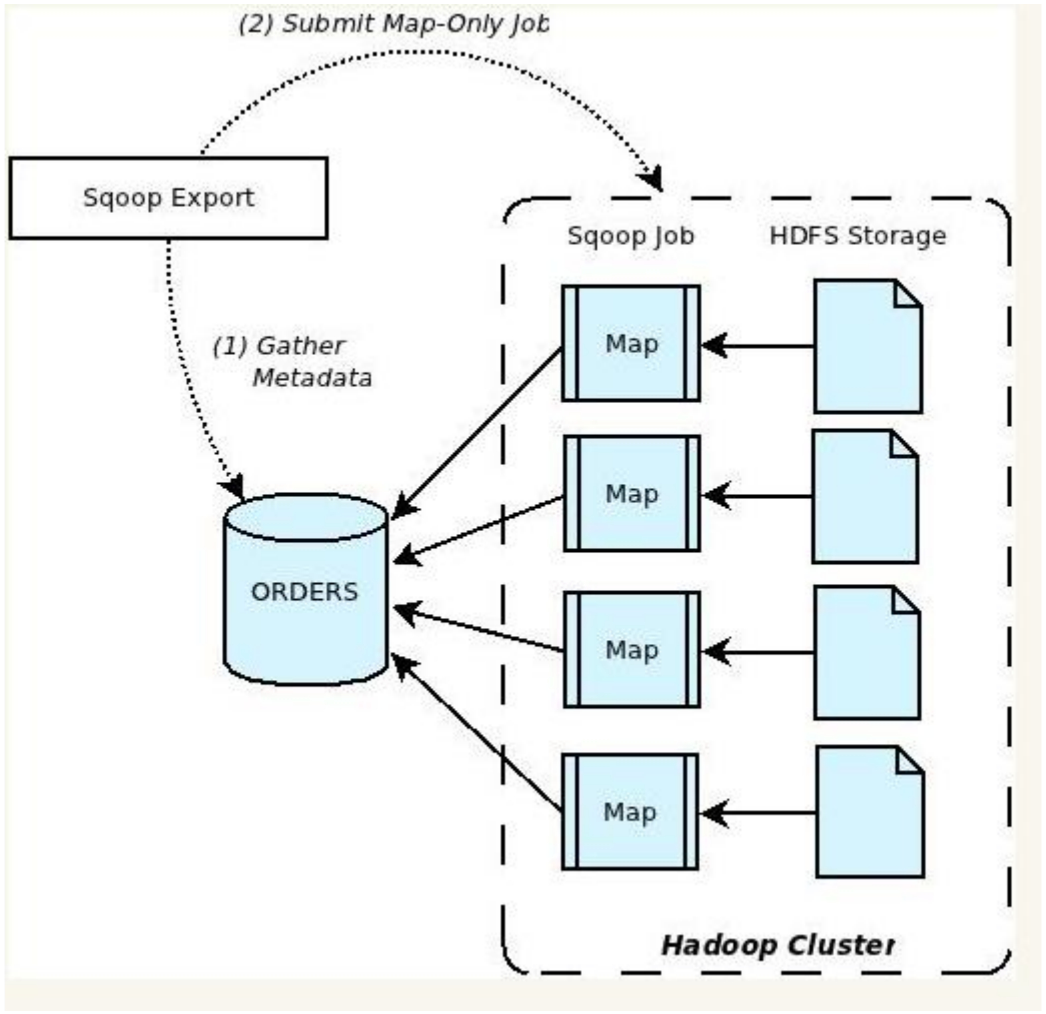


Figure 1: Sqoop Import Overview

SCOOP EXPORT ARCHITECTURE



```
sqoop export –
-connect <dbname>:
//localhost/acmedb \
 --table ORDERS
--username test
--password **** \
--export-dir /user/
<name>/ORDERS
```

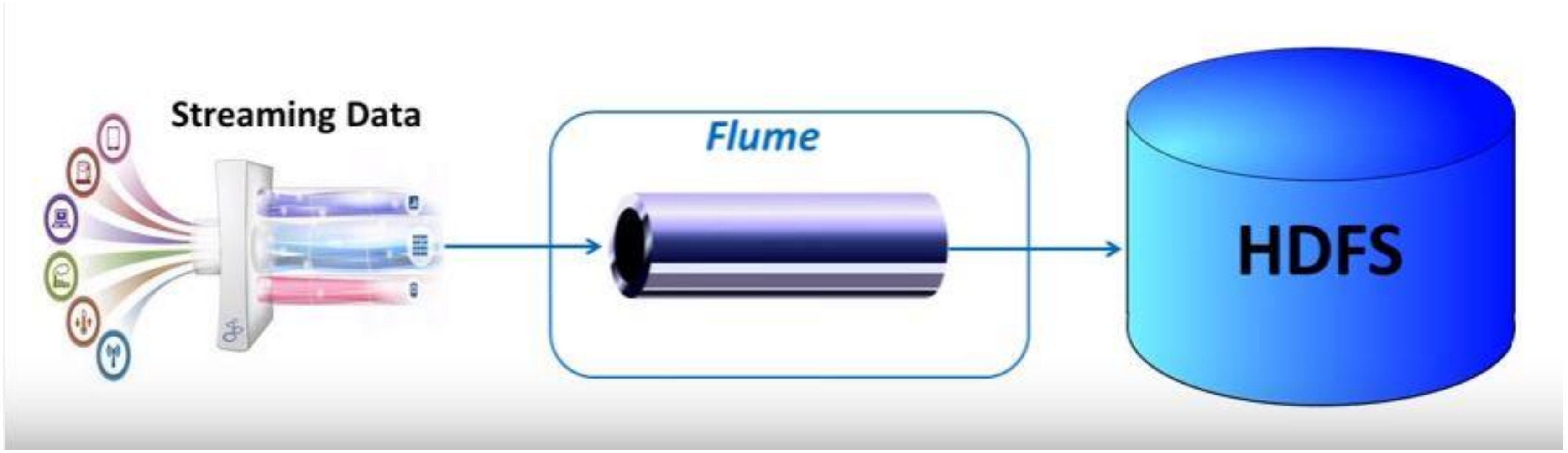
BDB - Streaming

Capability Team

STREAMING – WHAT & WHY?

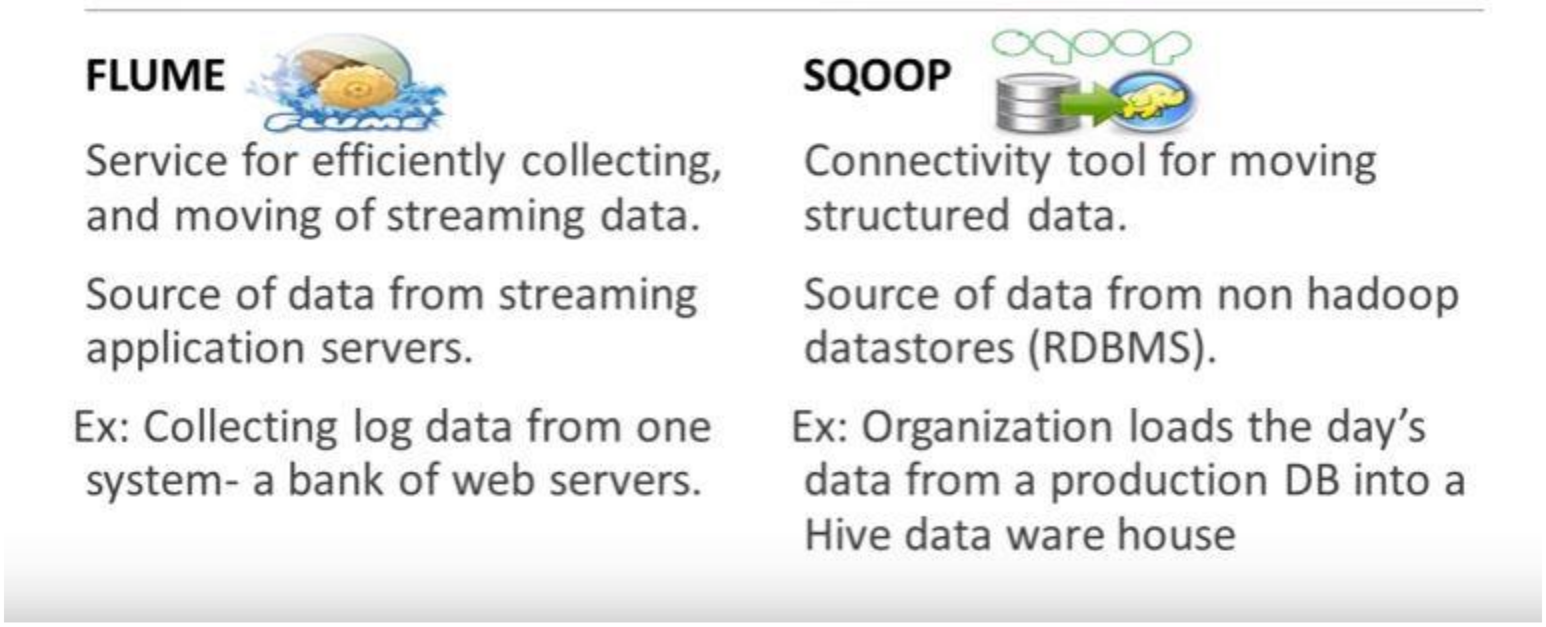
- HIGH VALUE
- SPEED OF ARRIVAL
- TIME TO ACTION
- DATA PIPELINES

STREAMING – WAYS?



TERADATA.

FLUME vs SCOOP



FLUME

Service for efficiently collecting, and moving of streaming data.

Source of data from streaming application servers.

Ex: Collecting log data from one system- a bank of web servers.

SQOOP

Connectivity tool for moving structured data.

Source of data from non hadoop datastores (RDBMS).

Ex: Organization loads the day's data from a production DB into a Hive data ware house

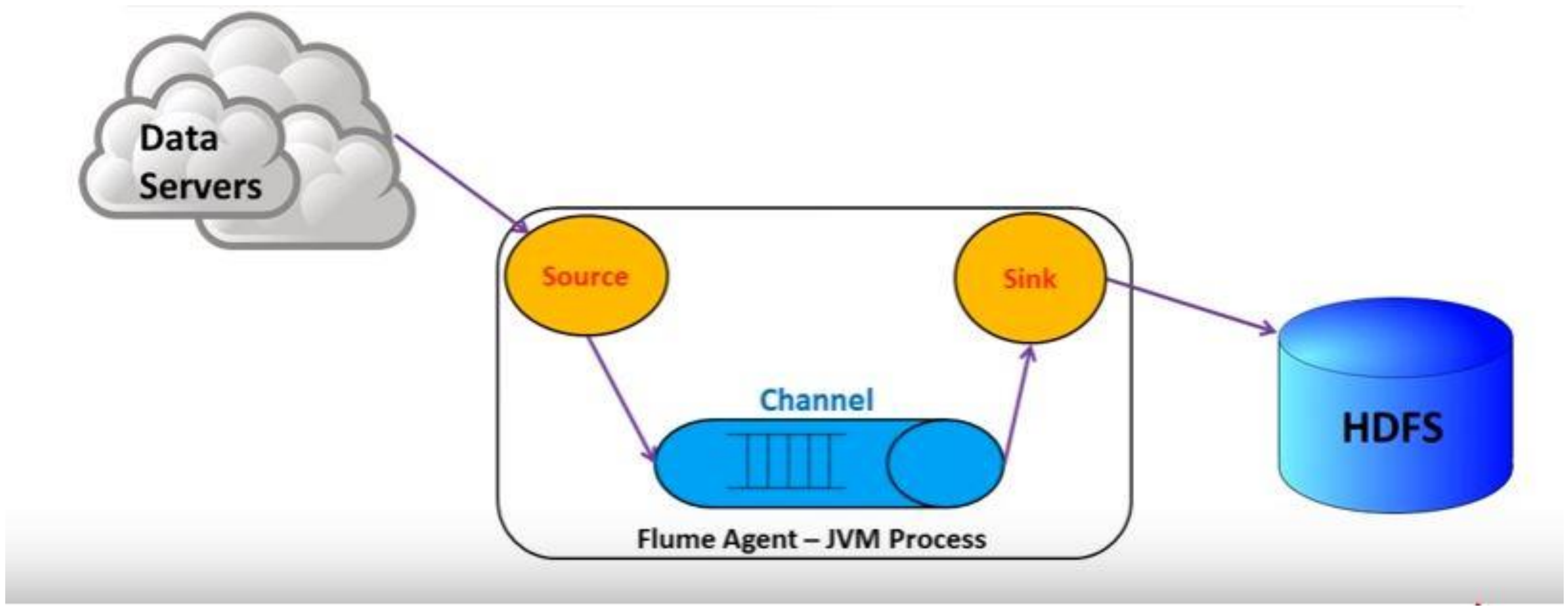
FLUME GOALS

- RELIABILITY
- SCALABILITY
- EXTENSIBILITY
- MANAGEABILITY

WHAT IS FLUME?

Apache Flume is a distributed, reliable, and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralized data store.

FLUME ARCHITECTURE



Q & A