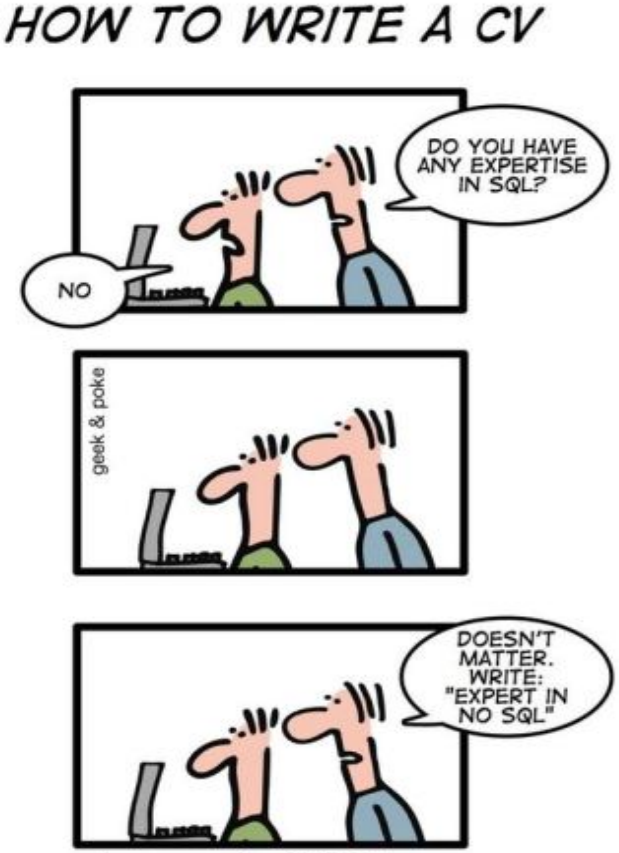
NoSQL technologies

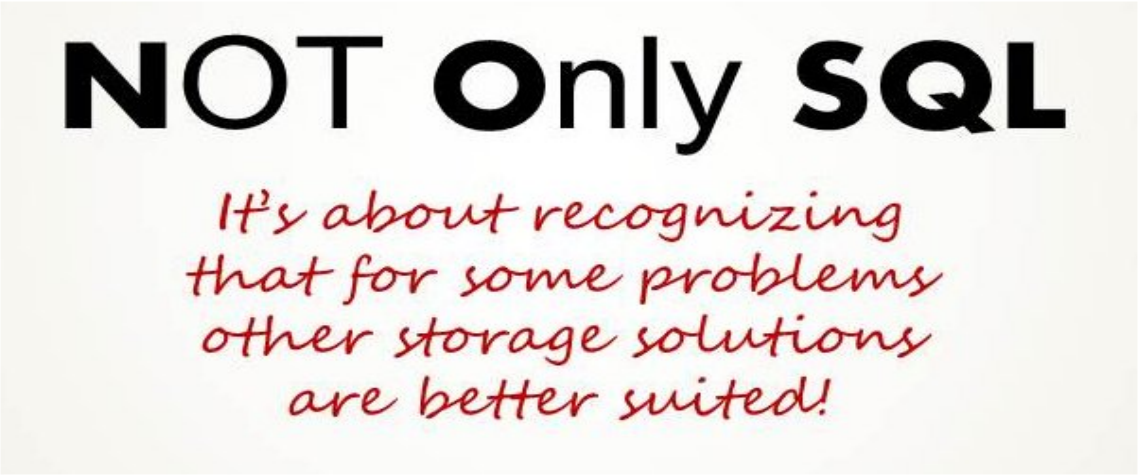
Naren Ramakrishnan
Thomas L. Phillips Professor of Engineering
Director, Discovery Analytics Center

1872 VirginiaTech

NoSQL


HOW TO WRITE A CV
DO YOU HAVE ANY EXPERTISE IN SQL?
NO
geek & poke
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"

Leverage the NoSQL boom



NOT **O**nly **SQL**

It's about recognizing that for some problems other storage solutions are better suited!

Motivations for NoSQL

- Non-traditional applications
  - – Key-value stores (e.g., Redis)
  - – Column-oriented storage (e.g., Cassandra)
  - – Graph databases (e.g., Neo4j)
  - – Document databases (e.g., MongoDB)
- Moral of the story
  - – The NoSQL landscape is a mess!

Lets get more technical

- NoSQL technologies refer to DBMSs that deviate from the traditional relational DB model
  - Less attention paid to schema (sometimes called “schema less”)
  - Avoid joins
  - Emphasis on analytical workloads rather than transactional workloads
  - Highly scalable

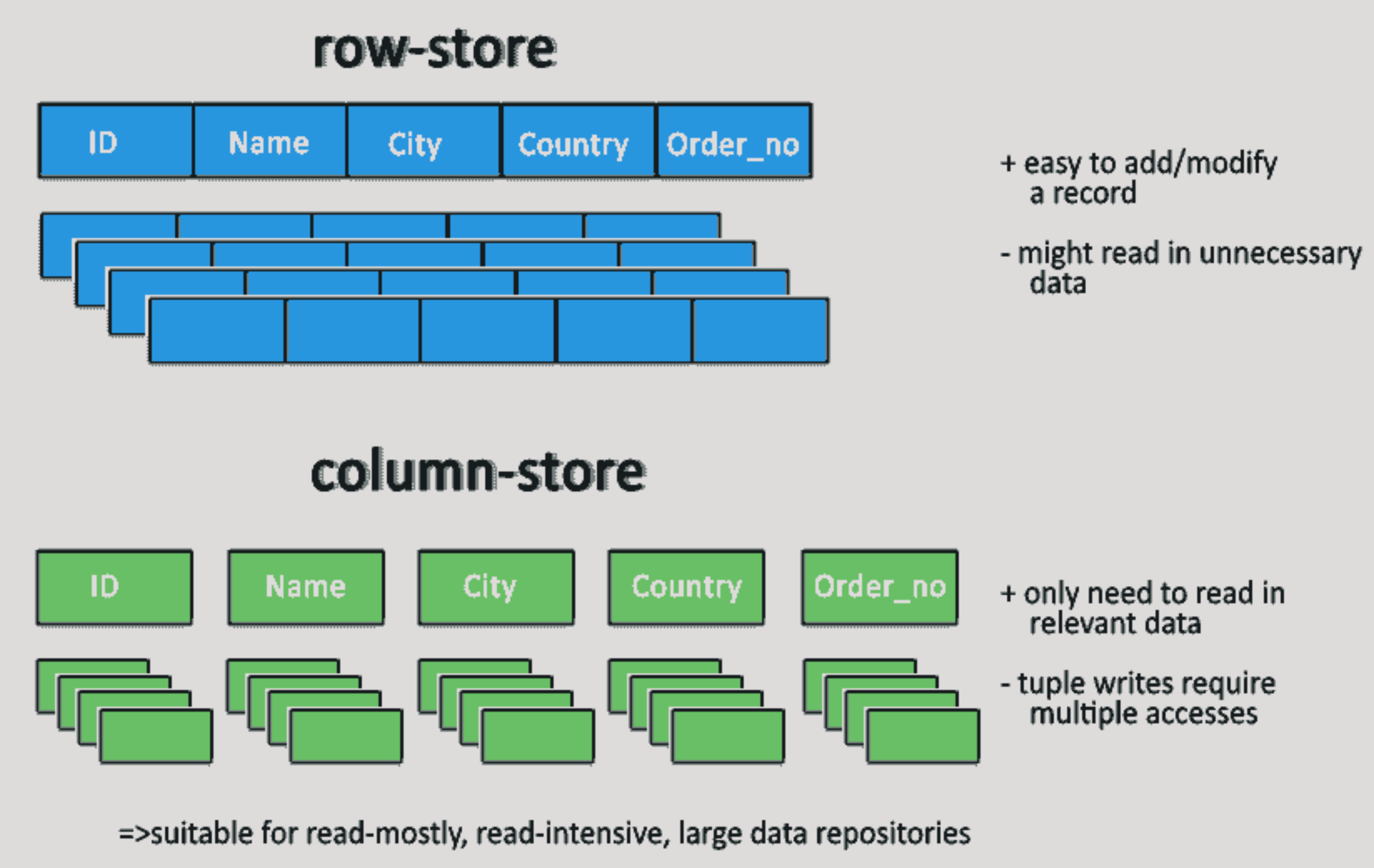
- Carlo Strozzi used NoSQL to denote his own DBMS that did not support SQL
  - But was still relational!
  - Modern NoSQL systems are not relational at all!
    - Hence NoREL would have been a better term!

- NoSQL better viewed as silly/stupid a term as “AJAX”
  - A bundle of technologies more than any coherent theme

Motivations galore

- Amount of data to be stored grows exponentially!
  - Think Facebook
    - Incoming data rate was 600TB/day two years back
    - Their DBs store O(hundreds of PBs)!
- The types of workloads are novel!
- NoSQL should be viewed in conjunction with the underlying system architecture
  - Dynamically scalable
    - Ability to add/remove servers, minimize disruption

An illustration of how NoSQL technologies evolved from workloads



row-store

+ easy to add/modify a record

- might read in unnecessary data

column-store

+ only need to read in relevant data

- tuple writes require multiple accesses

=>suitable for read-mostly, read-intensive, large data repositories

Source: w3resource.com

Types of NoSQL DBs

- Key-value stores
  - Known by various names
    - Hashes, associative arrays, dictionaries
  - Simple operations
    - Fetch/Get, Write/Put, Delete
  - Example systems
    - CouchDB, Redis
  - Advantages
    - Rapid lookups, Easy to scale
  - Disadvantages
    - Lack of flexibility

Types of NoSQL DBs

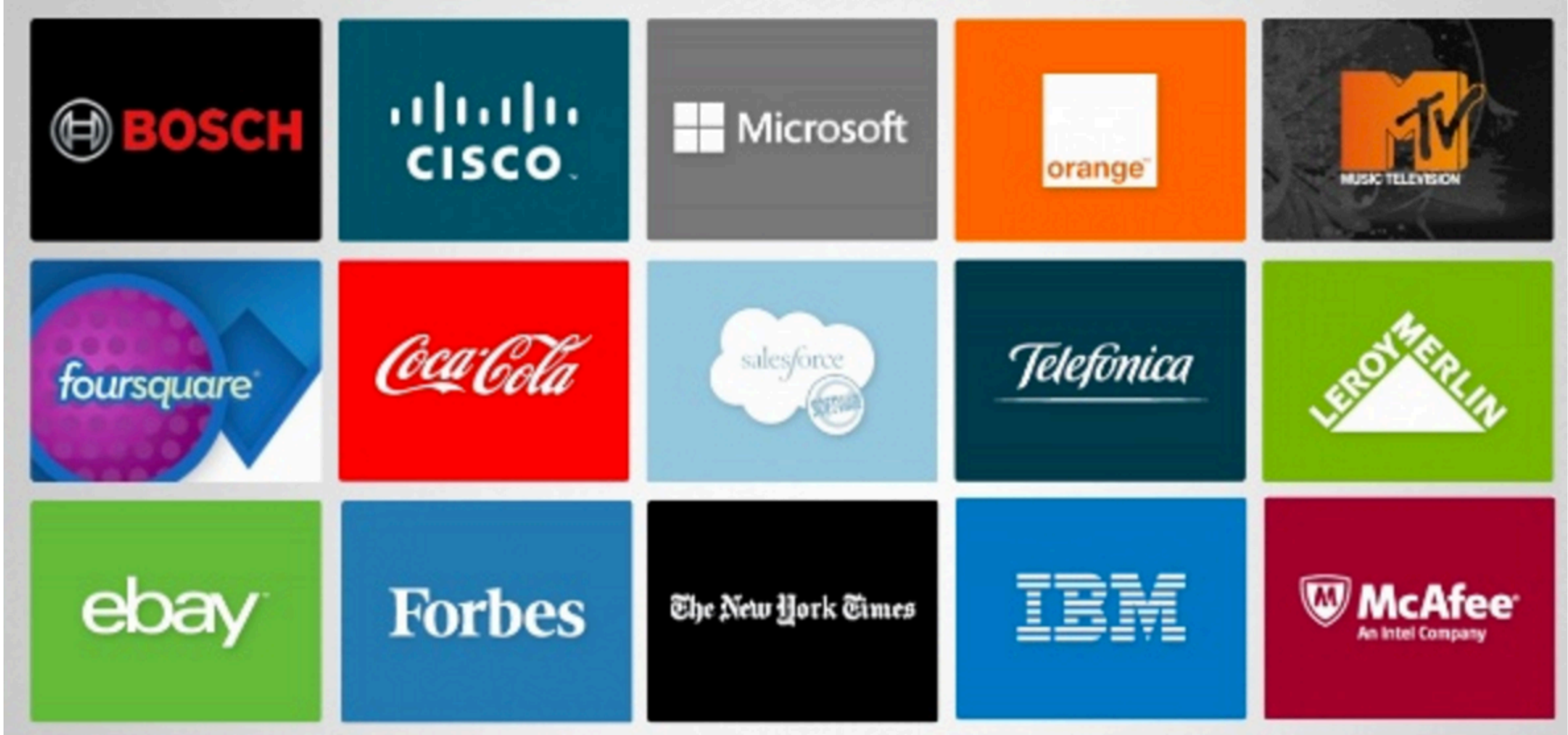
- Document stores
  - Can be viewed as a richer form of a key-value store
    - “Value” has some internal structure and might support richer query operations
  - Example systems
    - MongoDB, Cassandra
  - Advantages
    - Flexibility
  - Disadvantages
    - Can be less efficient

Types of NoSQL DBs

- Graph databases
  - Supports areas where link structure is important to be modeled
    - Nodes, edges, properties
      - E.g.,in social networking, recommendation systems, Graph search
  - Example systems
    - Neo4j, InfoGrid
  - Advantages
    - When relationship traversal becomes complex
  - Disadvantages
    - Efficiency issues, not easy to “distribute”

MongoDB

Who uses MongoDB?



Trigo, Cesar. "An Introduction to MongoDB." *An Introduction to MongoDB*. SlideShare, 29 June 2014. Web. 24 July 2016. http://www.slideshare.net/CesarTrigo/an-introduction-to-mongodb-36429852

JSON Format Summary

- JSON stands for JavaScript Object Notation.

- Each object is surrounded by curly braces. The entire JSON document is itself surrounded by braces.

- Each object may have multiple name/value pairs separated by a comma. Name/value pairs may represent a single value or multiple values in an array
  - Can be recursive
  - A colon separates the name from the associated value(s).

- Arrays are surrounded by brackets, and each element/item in the array is separated by a comma.

MongoDB (contd.)

- Data stored as whole documents

- Data stored in JSON format.
  - Text-based key-value storage format.
  - The basic structures of JSON are:
    1. A set of name/value pairs
    2. An ordered list of values

- Maps well to an object oriented programming model.

- Default unique *“_id” attribute in each mongo document.*

<u>Example Document:</u>

```
{
"address": {
            "building": "1007",
            "coord": [ -73.856077, 40.848447 ],
            "street": "Morris Park Ave",
            "zipcode": "10462"
},
"borough": "Bronx",
"cuisine": "Bakery",
"grades": [
            { "date": { "$date": 1393804800000 }, "grade":
"A", "score": 2 },
            { "date": { "$date": 1378857600000 }, "grade":
"A", "score": 6 },
],
"name": "Morris Park Bake Shop",
"restaurant_id": "30075445" }
```

MongoDB (contd.)

- Information in MongoDB is stored as *documents*
- A collection consists of a set of one or more documents
- A mongo database consists of one or more *collections*
- *_id* is a special key present in all documents.

Install MongoDB Locally

- Download appropriate distribution from http://www.mongodb.org/downloads

- Unzip the download to location of choice (let’s assume this is your desktop represented on the command line by ~/Desktop/)

Install MongoDB Locally (contd.)

- Open the terminal app and type the following commands:

```
$ sudo mkdir –p /data/db
```

- Open two additional tabs of the terminal app.
  - In tab1 we will start the mongo server locally:

```
$ sudo ~/Desktop/mongodb-xxxx/bin/mongod
```

  - In tab2 we will start an interactive shell where we will interact with mongo server that we just started.

```
$ sudo ~/Desktop/mongodb-xxxx/bin/mongo
```

Simple MongoDB commands

- help
- db.restaurants.insert(…)
  - Insert a document into the “restaurants” collection
- db.restaurants.find()
  - Find all documents in the restaurants collection
- db.restaurants.find({"cuisine" : "Italian”})
- db.restaurants.drop()
  - Removes the collection

MongoDB Update

- *Embedding a comment*

```
c = {
        “author” : “naren”,
        “date”: new Date(),
        ”text”: “great one!“
    }
```

```
    > db.restaurant.update( { _id : post._id } ,
              {$push : { comments: c }} )
```

MongoDB Update

- *Embedding a comment*

```
c = {
        “author” : “naren”,
        “date”: new Date(),
        ”text”: “great one!“
  }
```

```
    > db.restaurant.update( { _id : post._id } ,
              {$push : { comments: c }} )
```

Identify document using unique `_id` field.

MongoDB Update

- *Embedding a comment*

```
c = {
        “author” : “naren”,
        “date”: new Date(),
        ”text”: “great one!“
   }
```

```
  > db.restaurant.update( { _id : post._id } ,
            {$push : { comments: c }} )
```

Augment document and add a “comments” array field with the first comment stored in the `c` variable.

Before the update operation, the document didn’t have a “comments” key so the $push command creates a “comments” key in the document and adds an array with one comment as the value.

MongoDB Search and Sort

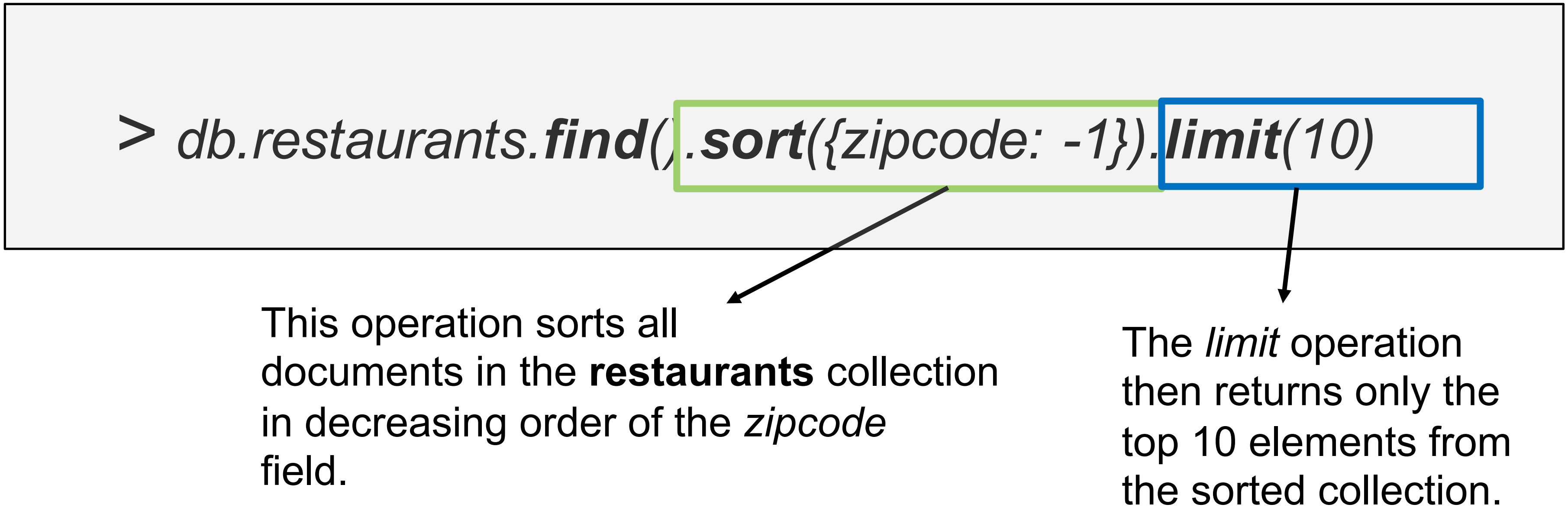
- *Return the last 10 blog entries in the collection* ***restaurants****.*

```
> db.restaurants.find().sort({zipcode: -1}).limit(10)
```

This operation sorts all documents in the **restaurants** collection in decreasing order of the *zipcode* field.

MongoDB Search and Sort

- *Return the last 10 blog entries in the collection* ***restaurants****.*



```
> db.restaurants.find().sort({zipcode: -1}).limit(10)
```

This operation sorts all documents in the **restaurants** collection in decreasing order of the *zipcode* field.

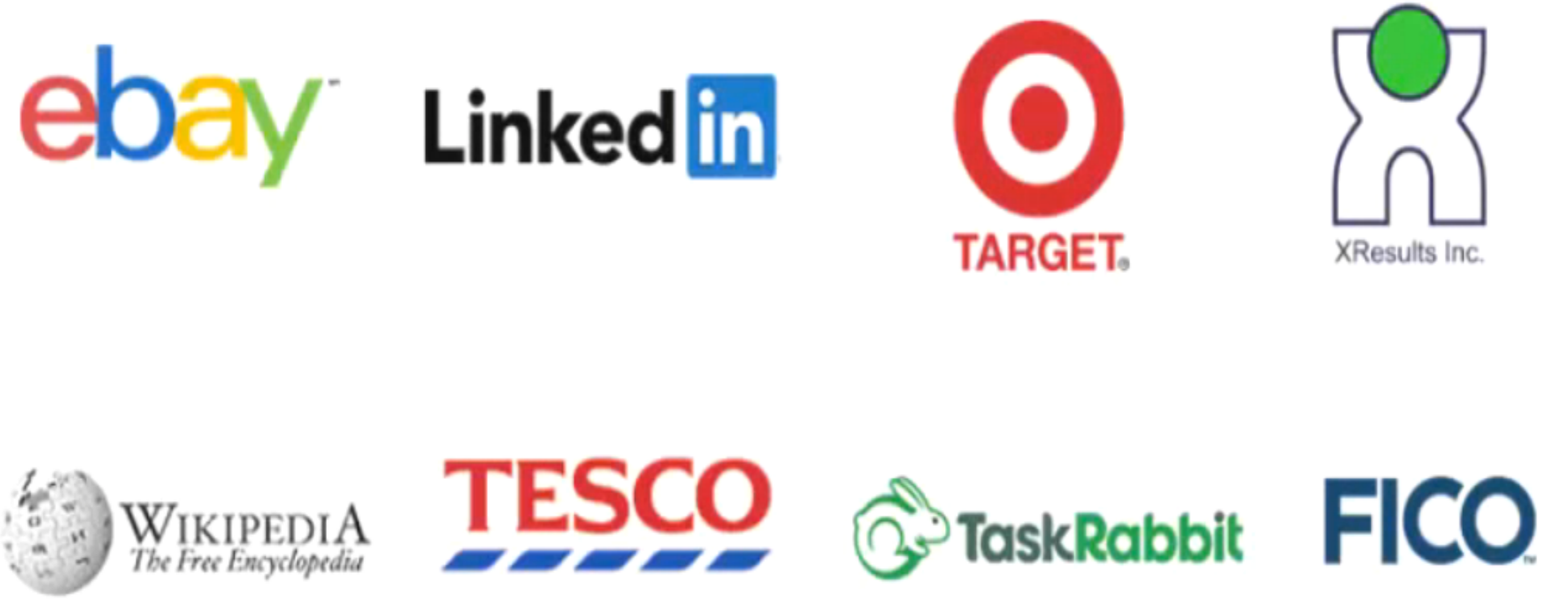
The *limit* operation then returns only the top 10 elements from the sorted collection.

ElasticSearch

What is ElasticSearch?

- In a nutshell:
  - ElasticSearch (ES) is an easy way to index and search a lot of data as it leverages the search power of Apache Lucene and combines it with an easy to use RESTful API.
  - Commonly used in applications involving log analysis:
    - Count number of user visits.
    - Detect anomalies in applications using error logs generated during the execution of the program.
    - Also can be used to measure the response time of a web-application, which is very important to any company which engages it’s consumer base with a website.

Who uses ElasticSearch?



Anthony, Ethan. "ElasticSearch Introduction." *YouTube*. YouTube, 19 June 2015. Web. 24 July 2016. https://www.youtube.com/watch?v=mo1OGUigcJA

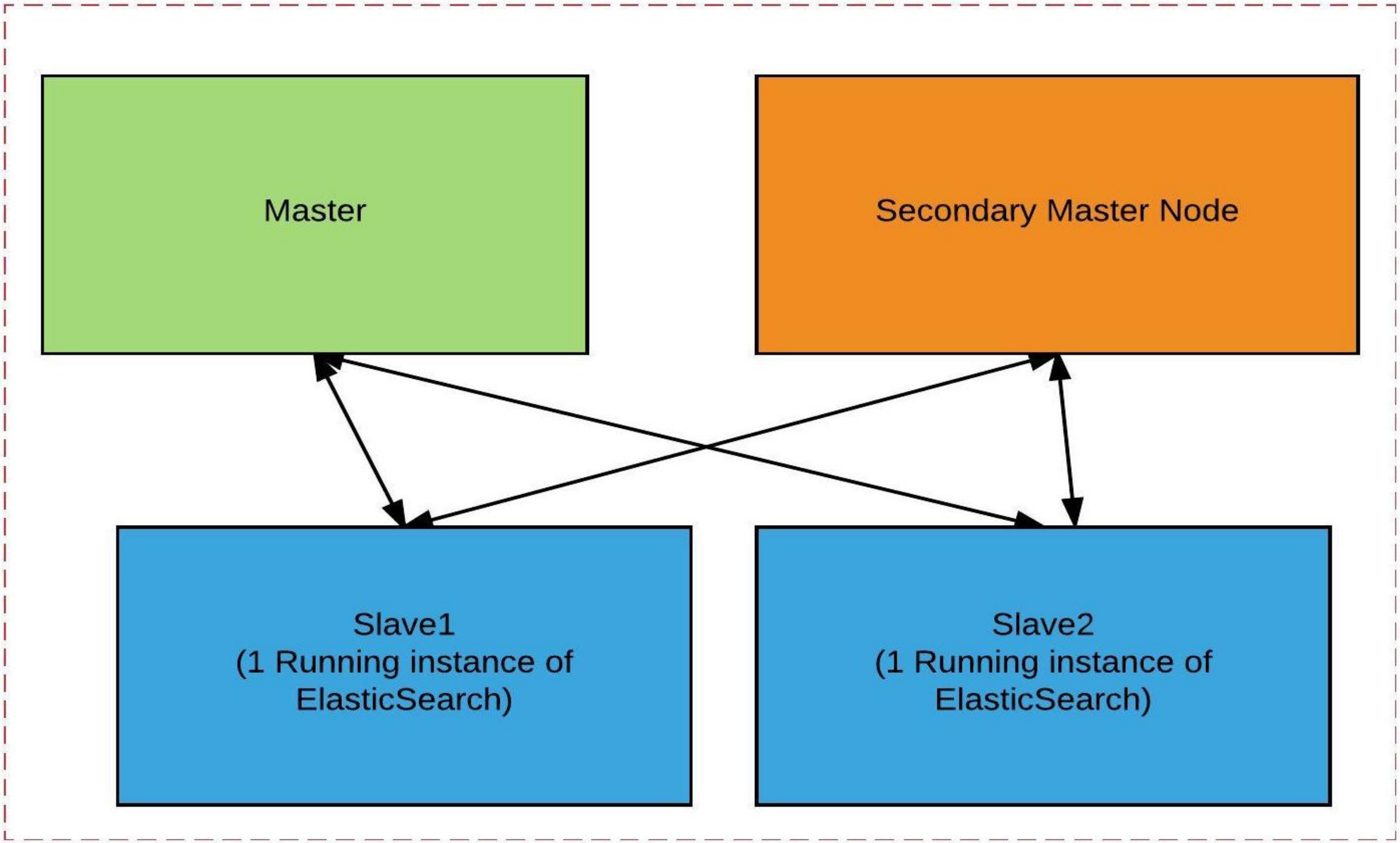
Features of ElasticSearch

- Document (JSON) oriented data storage and search engine
- Built on top of Apache Lucene Search Engine Library.
  - Lucene is a very popular state-of-the-art search engine library written in Java.
- RESTful API Centric
  - Basically: Documents can be queried over the internet with URLs
  - Language Agnostic Queries (language you use doesn’t matter as long as you have the correct URL)
- Horizontally Scalable
  - Basically: It’s easy to add more infrastructure (machines) to store more data, as your data gets bigger.
- Highly Available
  - If one of the machines in the cluster goes down, your data is not lost, it can be retrieved from other machines

Basic Concepts

- Cluster:
  - A cluster consists of one or more nodes (machines) which share the same cluster name.
  - Each cluster has a single master node which is chosen automatically by the cluster and which can be replaced if the current master node fails.
- Node :
  - A node is a running instance of elasticsearch which belongs to a cluster. Multiple nodes can be started on a single server for testing purposes, but usually you should have one node per server.

Basic Concepts



Cluster with three nodes & secondary Master Node.

Basic ES Terminology

- Index :
  - An index is like a ‘database’ in a relational database. It has a mapping which defines multiple *types*.
  - An index is a logical namespace which maps to one or more primary shards and can have zero or more replica shards.
- Type :
  - A type is like a ‘table’ in a relational database. Each type has a list of fields that can be specified for documents of that type.
  - The mapping defines how each field in the document is analyzed.

Data Visualization using Kibana

- Elastic (the company that created ES) also has a tool called Kibana.

- Kibana can be used on top of ES to create data visualizations.

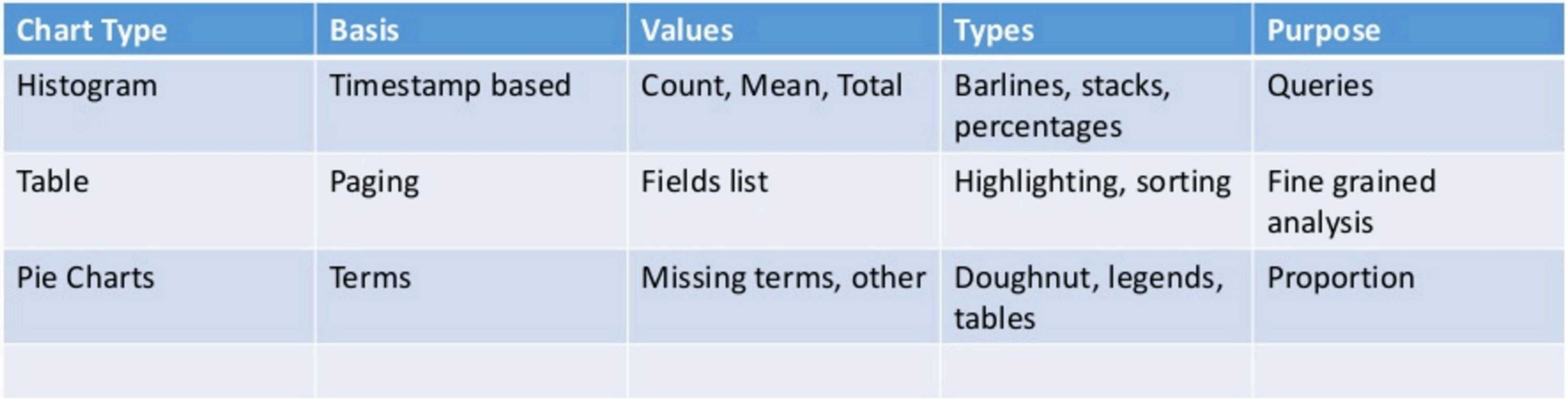
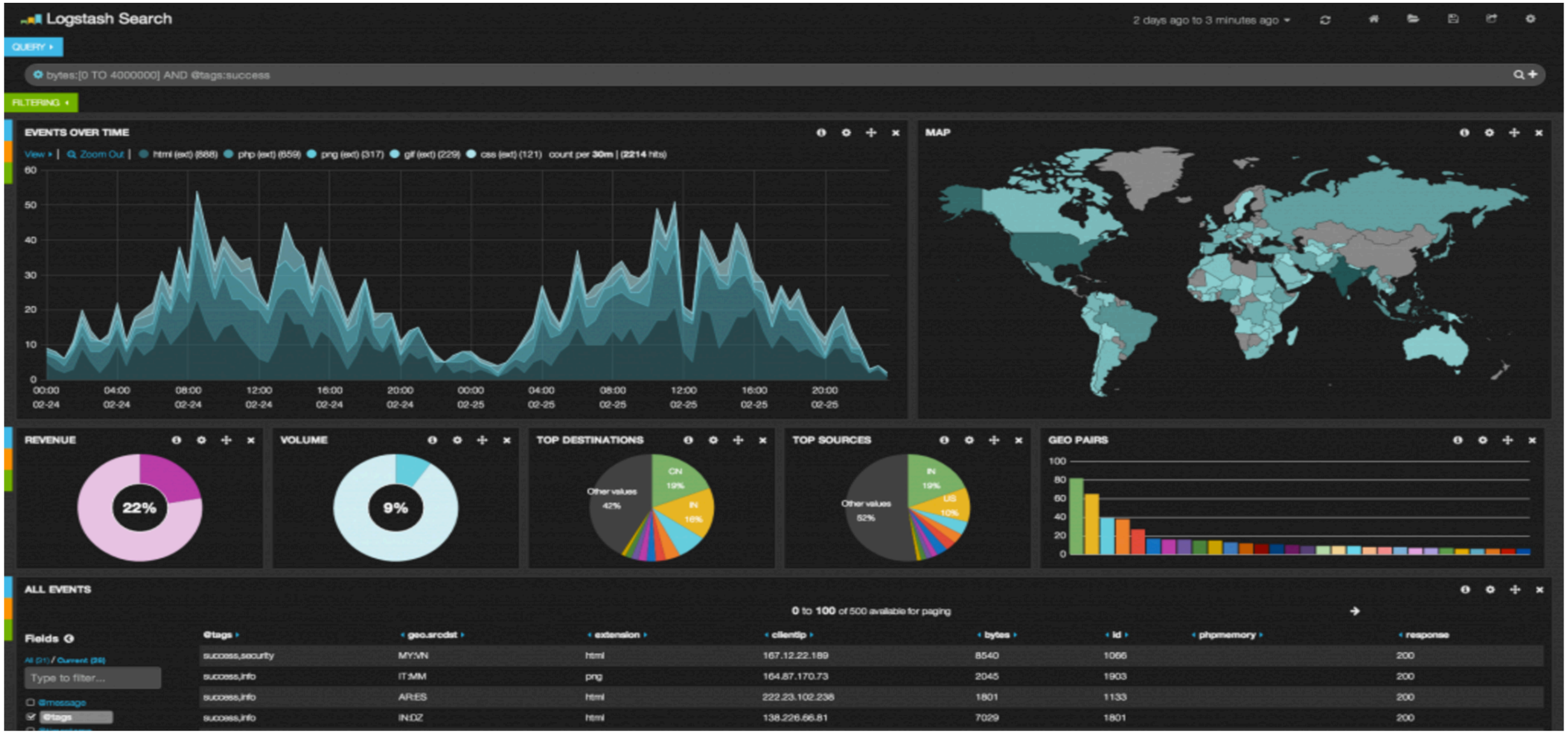


Chart Type	Basis	Values	Types	Purpose
Histogram	Timestamp based	Count, Mean, Total	Barlines, stacks, percentages	Queries
Table	Paging	Fields list	Highlighting, sorting	Fine grained analysis
Pie Charts	Terms	Missing terms, other	Doughnut, legends, tables	Proportion

Stirrup, Jen. "Introduction To Kibana." *Introduction To Kibana*. SlideShare, 17 Mar. 2015. Web. 24 July 2016. http://www.slideshare.net/jenstirrup/introduction-to-kibana-45933070

Example Kibana Dashboard



bytes:[0 TO 4000000] AND @tags:success

0 to 100 of 500 available for paging

@tags	geo.srcdst	extension	clientip	bytes	id	phpmemory	response
success,security	MY:VN	html	167.12.22.189	8540	1066		200
success,info	IT:MM	png	164.87.170.73	2045	1903		200
success,info	AR:ES	html	222.23.102.238	1801	1133		200
success,info	IN:DZ	html	138.226.66.81	7029	1801		200

Can be used to visualize :

- Time series data
- Geographical maps
- Histograms and pie charts.

Khan, Rashid. "Kibana 3.0.0 GA Is Now Available!" *Elastic Blog*. Elastic, 18 Mar. 2014. Web. 24 July 2016. https://www.elastic.co/blog/kibana-3-0-0-ga-now-available