BITS Pilani

Module 9 Part 2
NoSQL Databases

Harvinder S Jabbal
SSZG653 Software Architectures

NoSQL databases

- While Relational databases (SQL databases) are good for transaction management, not all applications need this feature.

- Many web applications such as marketing applications, IoT applications, need fast processing of data but do not need transaction management

- NoSQL databases came as a response to this need

https://www.dataversity.net/a-brief-history-of-non-relational-databases/#

NoSQL databases

Traditional databases also have limitations on storage

NoSQL databases are

- Scalable (Sharding)
- Fast (In Memory)
- Available (Replication)
- Handle semi-structured and unstructured data
- Rapidly adapt to changing data needs

However most lack

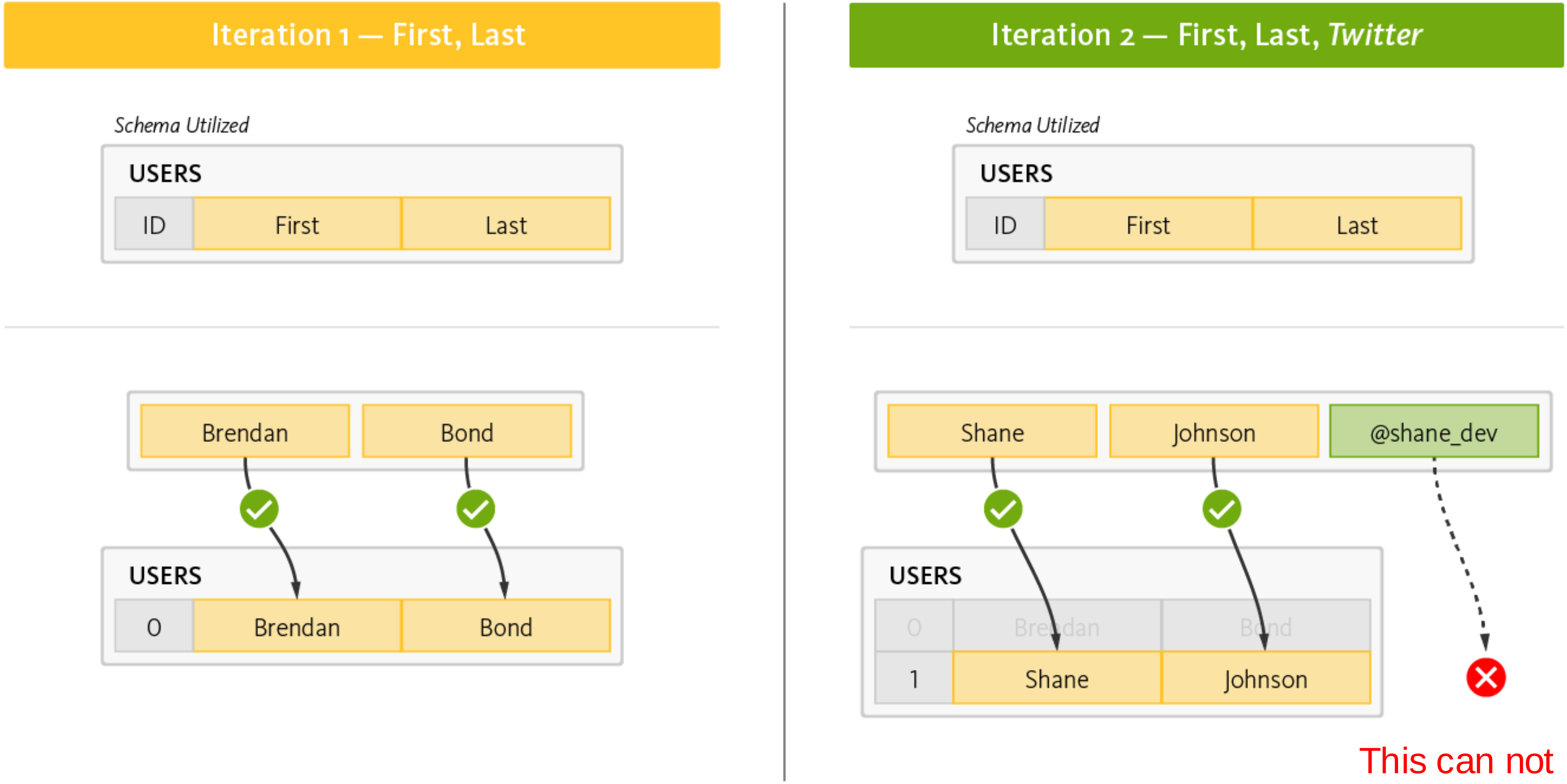
- Transaction support (ACID)- Atomicity, Consistency, Isolation, Durability.
- Join feature

Examples of usage of NoSQL DB

- **Tesco**, Europe's #1 retailer, uses NoSQL to manage millions of products, promotions supply chain, etc.
- **Ryanair**, the world's busiest airline, uses NoSQL to power its mobile app serving over 3 million users
- **Marriott** is deploying NoSQL for its reservation system that books $38 billion annually
- **Gannett** (USA Today) the #1 U.S. newspaper publisher, uses NoSQL for its proprietary content management system, Presto
- **GE** is deploying NoSQL for its Predix platform to help manage the Industrial Internet

NoSQL - Flexibility

In RDBMS, if we want to add a new column, we need to change the schema

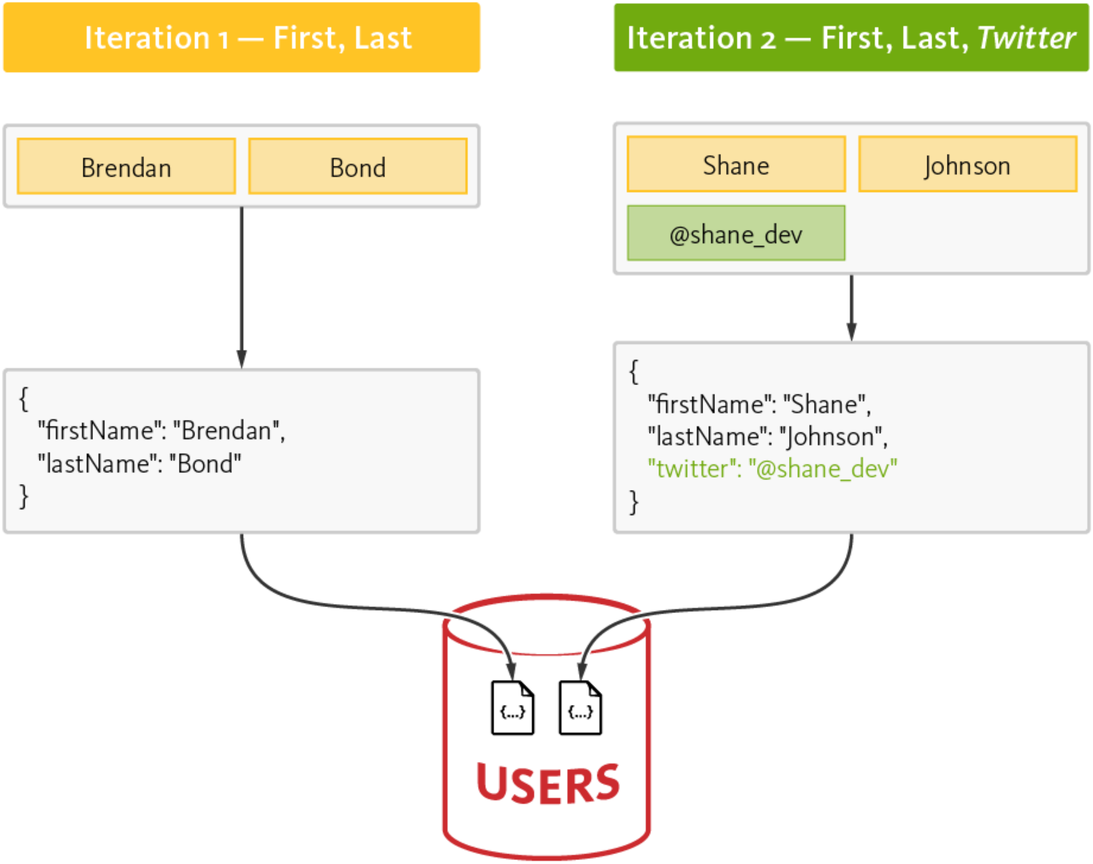


This can not be stored

Ref: https://www.couchbase.com/resources/why-nosql

NoSQL - Flexibility

In NoSQL DB, we can easily add new columns.

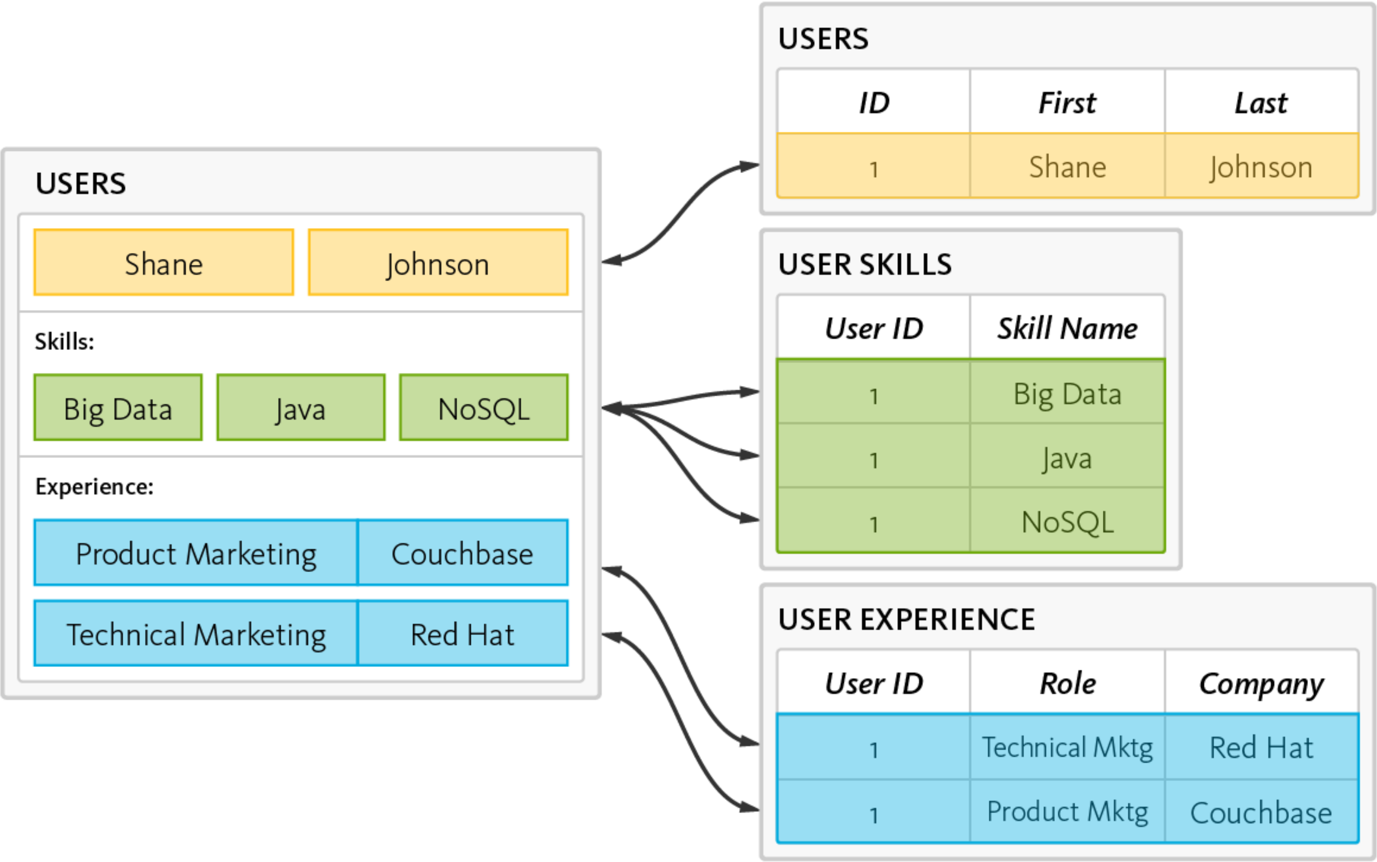

Iteration 1 — First, Last
Iteration 2 — First, Last, *Twitter*
Brendan
Bond
Shane
Johnson
@shane_dev
{...}
{...}
USERS

```
{
  "firstName": "Brendan",
  "lastName": "Bond"
}
```

```
{
  "firstName": "Shane",
  "lastName": "Johnson",
  "twitter": "@shane_dev"
}
```

Ref: https://www.couchbase.com/resources/why-nosql

NoSQL - Simplicity



USERS

Shane | Johnson

Skills:

Big Data | Java | NoSQL

Experience:

Product Marketing	Couchbase
Technical Marketing	Red Hat

USERS

ID	*First*	*Last*
1	Shane	Johnson

USER SKILLS

User ID	*Skill Name*
1	Big Data
1	Java
1	NoSQL

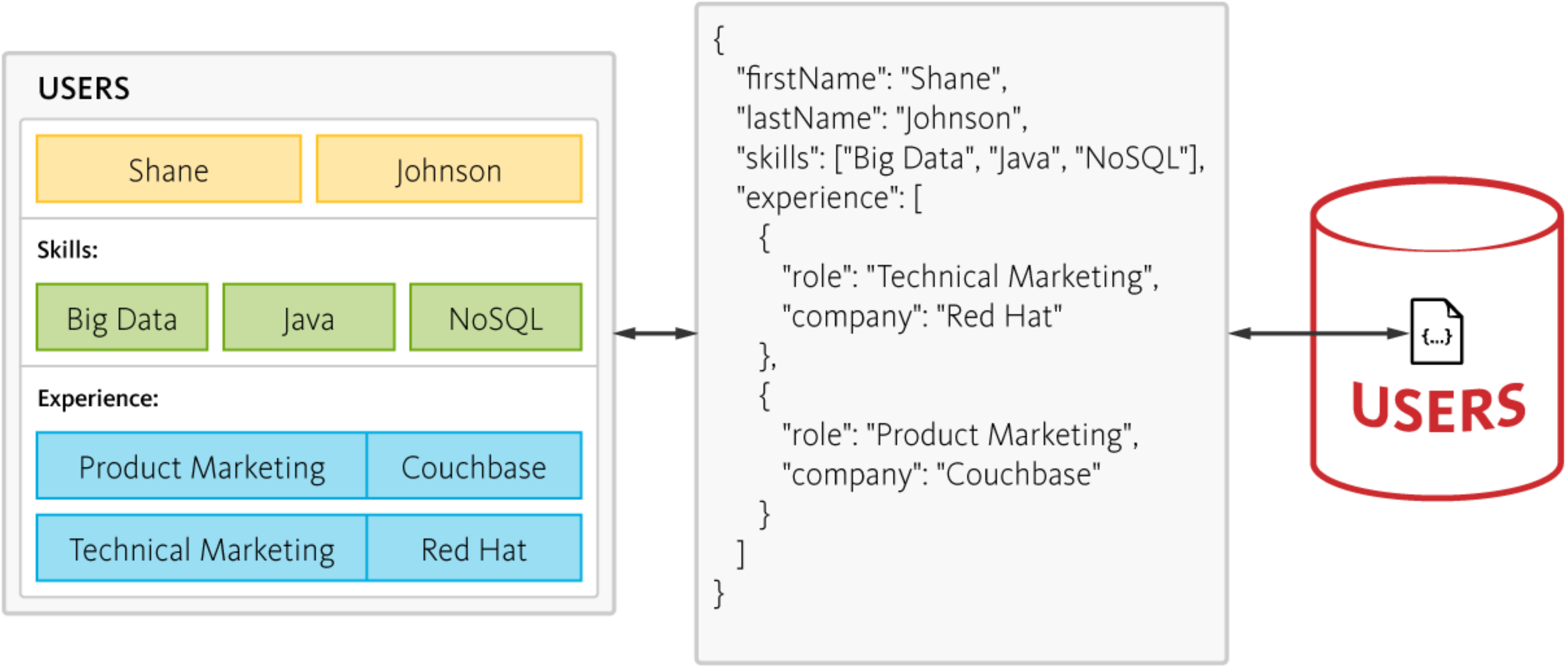
USER EXPERIENCE

User ID	*Role*	*Company*
1	Technical Mktg	Red Hat
1	Product Mktg	Couchbase

NoSQL

SQL

NoSQL - Simplicity



USERS

Shane	Johnson

Skills:

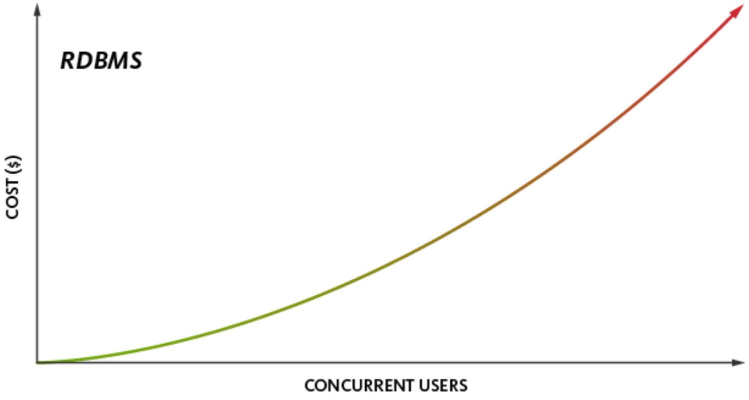
Big Data	Java	NoSQL

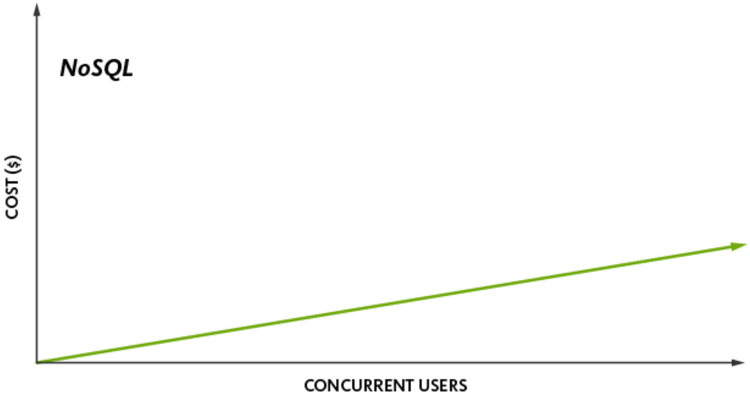
Experience:

Product Marketing	Couchbase
Technical Marketing	Red Hat

```
{
  "firstName": "Shane",
  "lastName": "Johnson",
  "skills": ["Big Data", "Java", "NoSQL"],
  "experience": [
    {
      "role": "Technical Marketing",
      "company": "Red Hat"
    },
    {
      "role": "Product Marketing",
      "company": "Couchbase"
    }
  ]
}
```

NoSQL – Cost effective





Cost: Memory, CPU, storage

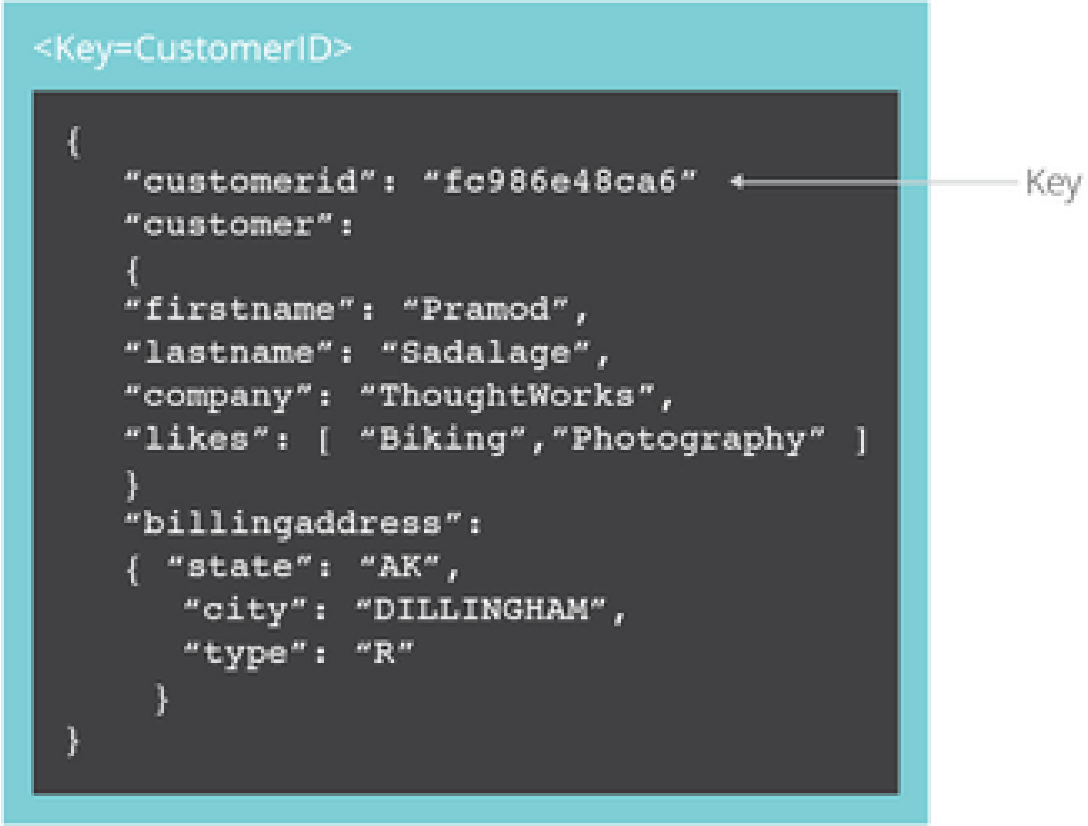
NoSQL Database

Types of NoSQL databases:

- Document
- Key Value
- Column stores
- Graph

Document NoSQL database

- The database stores and retrieves documents, which can be XML, JSON, BSON, and so on.
- Some of the popular document databases we have seen are MongoDB, CouchDB , Terrastore, OrientDB , RavenDB, and of course the well-known and often reviled Lotus Notes that uses document storage.



```
{
   "customerid": "fc986e48ca6"
   "customer":
   {
   "firstname": "Pramod",
   "lastname": "Sadalage",
   "company": "ThoughtWorks",
   "likes": [ "Biking","Photography" ]
   }
   "billingaddress":
   { "state": "AK",
     "city": "DILLINGHAM",
     "type": "R"
    }
}
```

Example of one record

RDBMS- single node for read and write

- Consistent
- Not partition tolerant.
- Compromise availability during write.

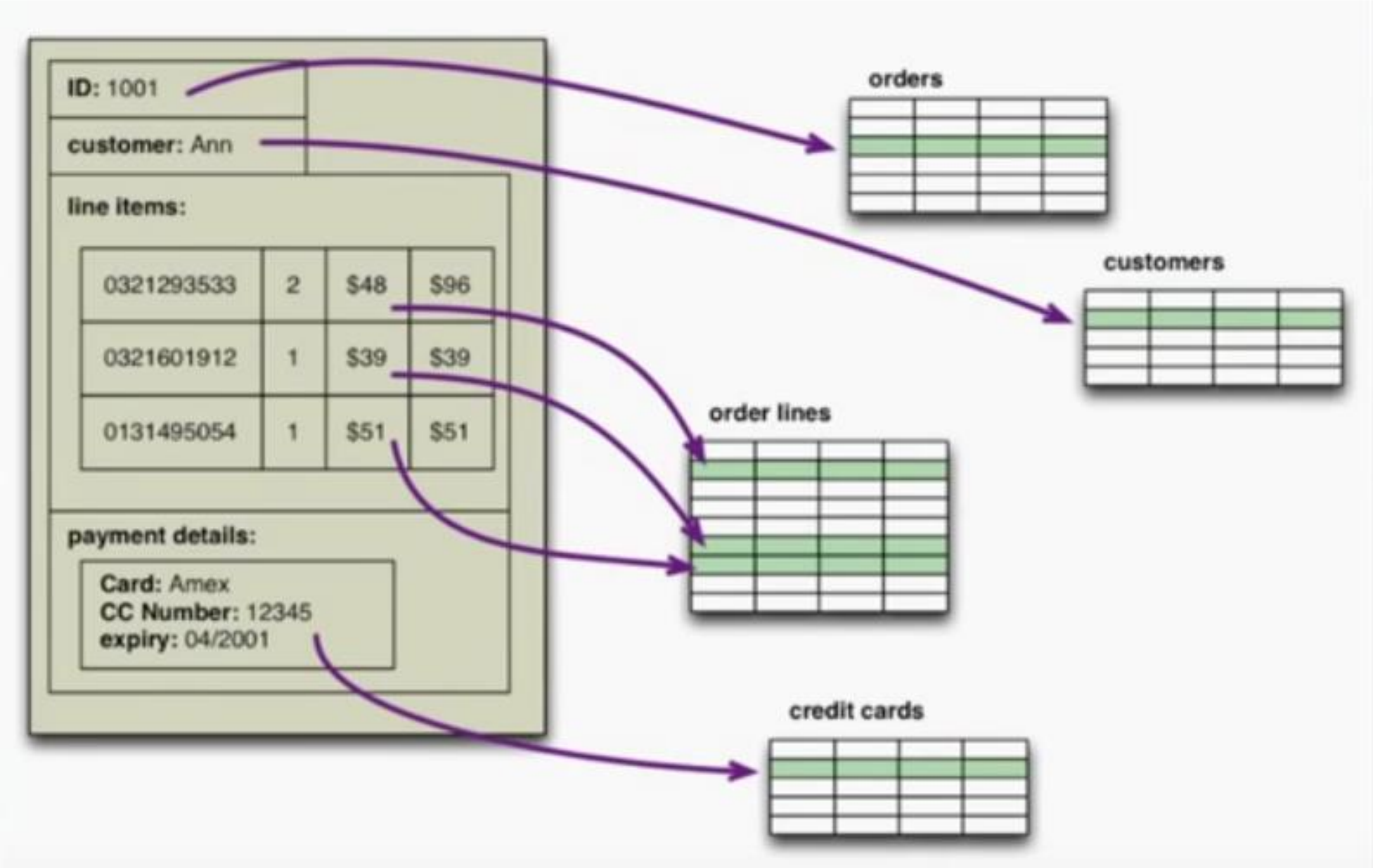
Two Machines:

- High Availability
- Partition Tolerant
- No Consistency.

One machine with backup – HDFS/RDBMS with mirror

- Consistent, Partition tolerant but not highly available.

Document DB vs Relational DB



Good for

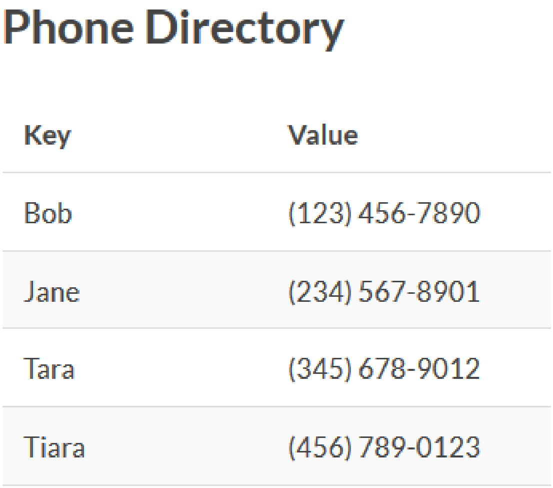
- Ecommerce platform
- Content management systems

Features of Mongo DB

- Indexing
- Ad hoc search
- Replication
- Partitioning / Sharding
  - Ex. Partition data by Product or Geography

Key-Value database

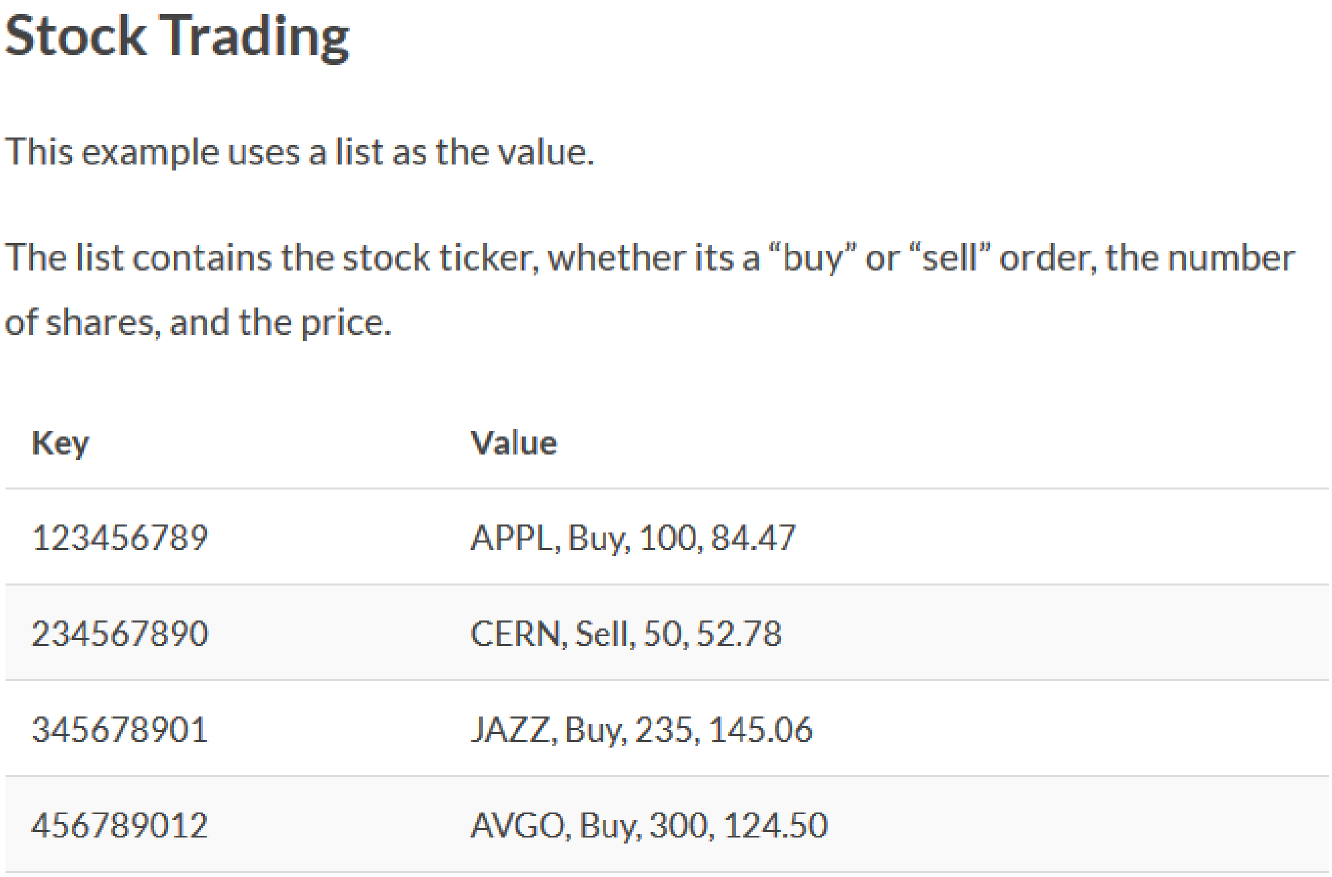
- Data model is Key value pair
- Uses hashing for fast access
- The DB does not care what is contained in the value
- Example scenarios: Phone directory, Stock trading
- Some of the popular key-value databases are Riak, Redis (often referred to as Data Structure server), Memcached and its flavors, Berkeley DB, upscaledb (especially suited for embedded use), Amazon DynamoDB (not open-source), Project Voldemort and Couchbase.



Phone Directory

Key	Value
Bob	(123) 456-7890
Jane	(234) 567-8901
Tara	(345) 678-9012
Tiara	(456) 789-0123

Example of key value database



Stock Trading

This example uses a list as the value.

The list contains the stock ticker, whether its a “buy” or “sell” order, the number of shares, and the price.

Key	Value
123456789	APPL, Buy, 100, 84.47
234567890	CERN, Sell, 50, 52.78
345678901	JAZZ, Buy, 235, 145.06
456789012	AVGO, Buy, 300, 124.50

More examples: User profiles, Blog comments

Uses of Redis

- Session cache, with persistence

Column oriented database

This is useful for data analysis scenarios

Example: Calculate the average usage of electricity in 2018 in East Bangalore region

Traditional way: Read all the billing records of 2018

Cust id, Name, Addrs, Region, Month, Year, Usage, Amt,

Record 1: 001, John Mancha, Addr 1, East, Jan, 2018, 100, 600

Record 2: 002, Vivek Kulkarni, Addr 2, East, Jan, 2018, 90, 540

Record 3: 003, Shanti Sharma, Addr 3, West, Jan, 2018, 110, 660

….

….

Column oriented database

Instead if we store the data as follows:

Record 1: 001, John Mancha, Addr 1, East // Customer details
Record 2: 002, Vivek Kulkarni, Addr 2, East

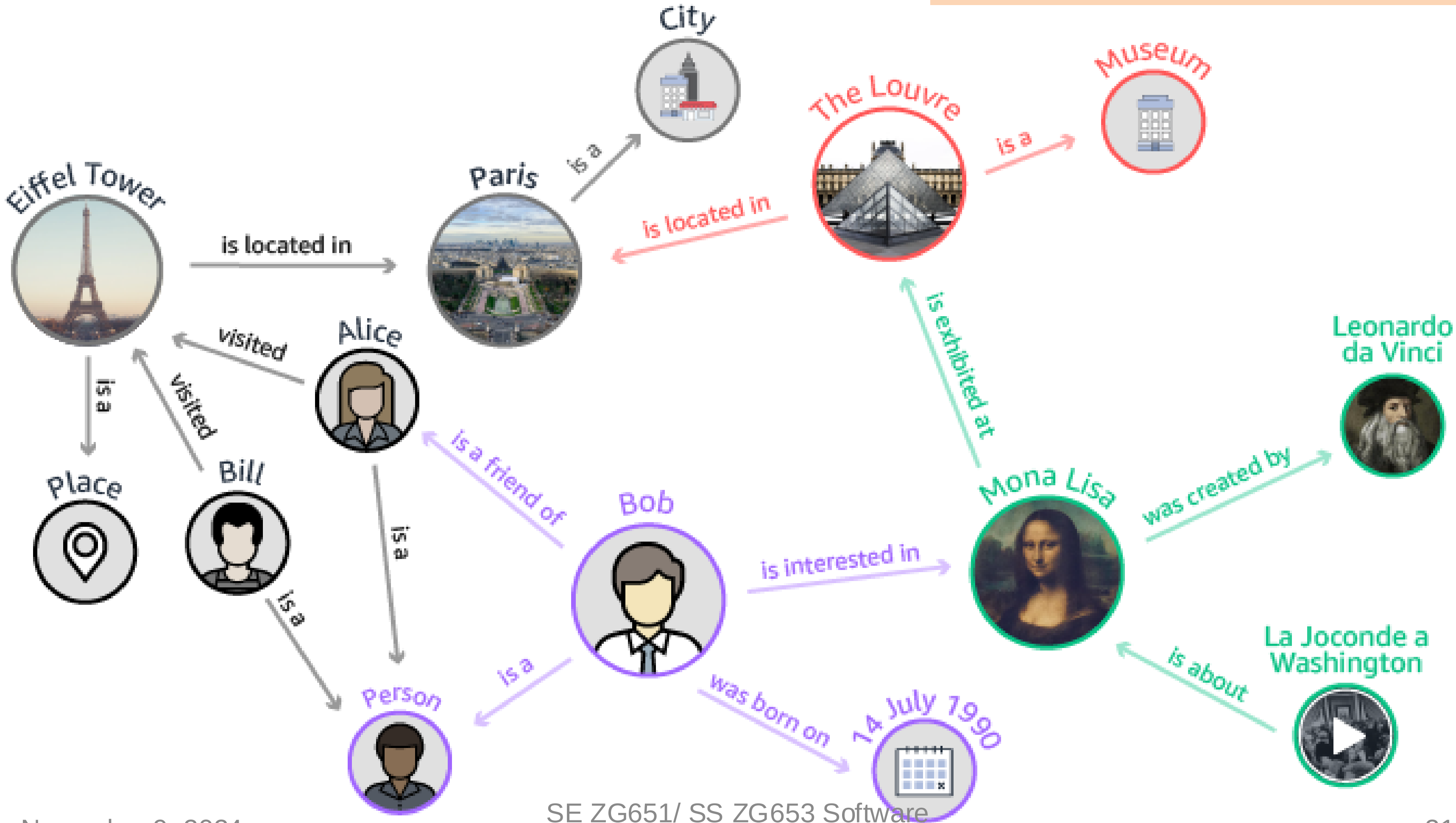
Record A: Jan, 2018, 100, 90, 110, … // Usage – in customer order
Record B: Feb, 2018, 110, 92, 115, …

Only one record 'Record A' is needed to calculate average usage in Jan 2018

Good for summarization

Graph database: Example: Knowledge graph

Entities and their relationships



Graph database

Entities and relationships have properties (attributes)

Ex.

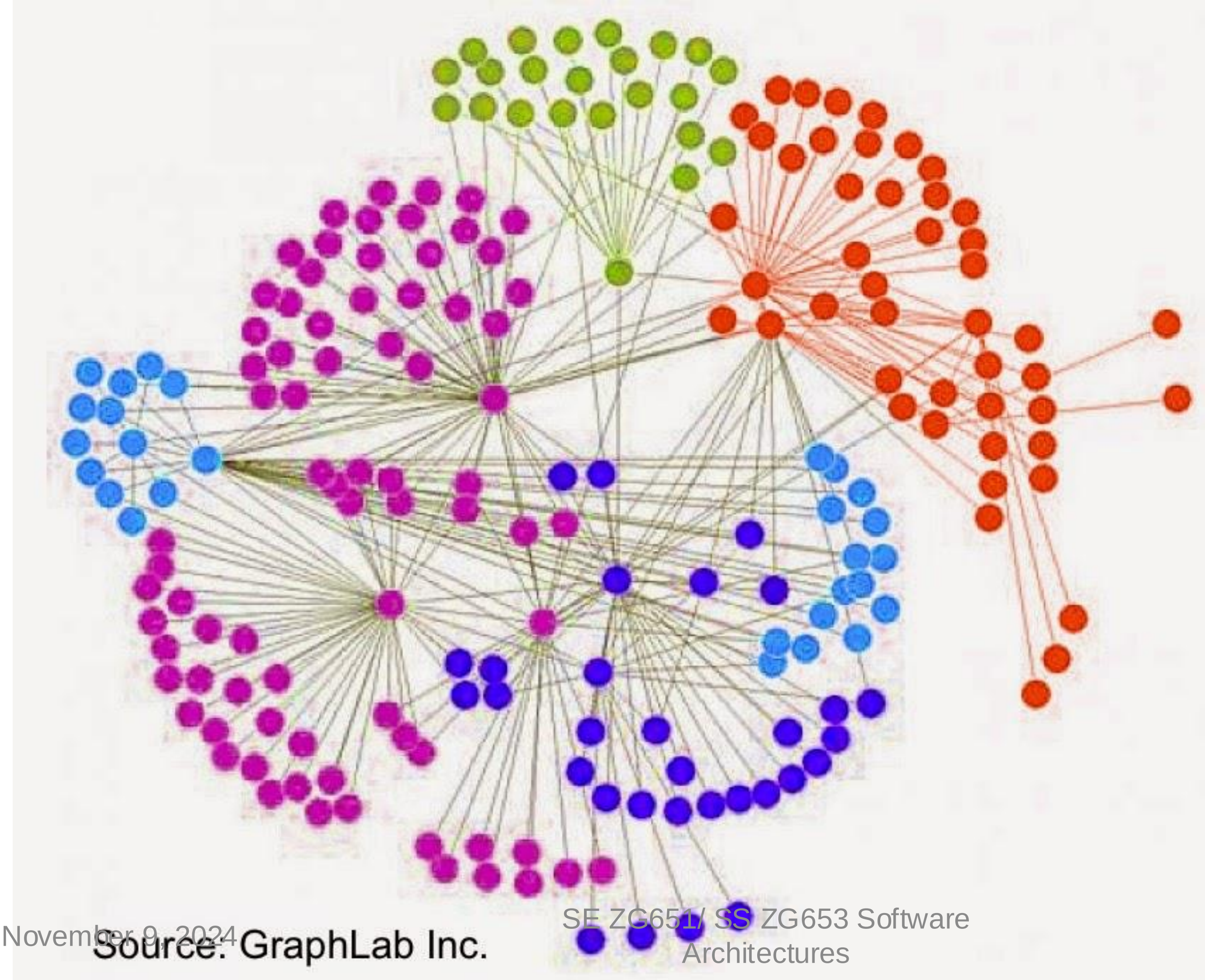
- Eifel tower properties can be height, date of construction
- Visited relationship can have properties such as date of visit

Uses

- Store a large amount of inter-related information and to search for an entity along with its relationships
- Fraud detection
- Social networks

Source: https://opensourceforu.com/2017/05/different-types-nosql-databases/

Can be use to detect hidden patterns



Source: GraphLab Inc.

Easy to understand clusters

Graph database

- Graph databases allow you to store entities and relationships between these entities.
- Entities are also known as nodes, which have properties. Think of a node as an instance of an object in the application.
- Relations are known as edges that can have properties. Edges have directional significance; nodes are organized by relationships which allow you to find interesting patterns between the nodes.
- There are many graph databases available, such as Neo4J, Infinite Graph, OrientDB, or FlockDB

In-Memory databases

- An **in-memory database** (**IMDB**; also **main memory database system** or **MMDB** or **memory resident database**) is a database management system that primarily relies on main memory
- Applications where response time is critical, such as those running telecommunications network equipment and mobile advertising networks, often use main-memory databases
- With the introduction of non-volatile random access memory technology (Flash memory), in-memory databases will be able to run at full speed and maintain data in the event of power failure
- Popular In-memory databases are SAP's HANA, IBM DB2 BLU, Oracle
- These databases support OLTP and OLAP (Online Analytical Processing)

Acknowledgement

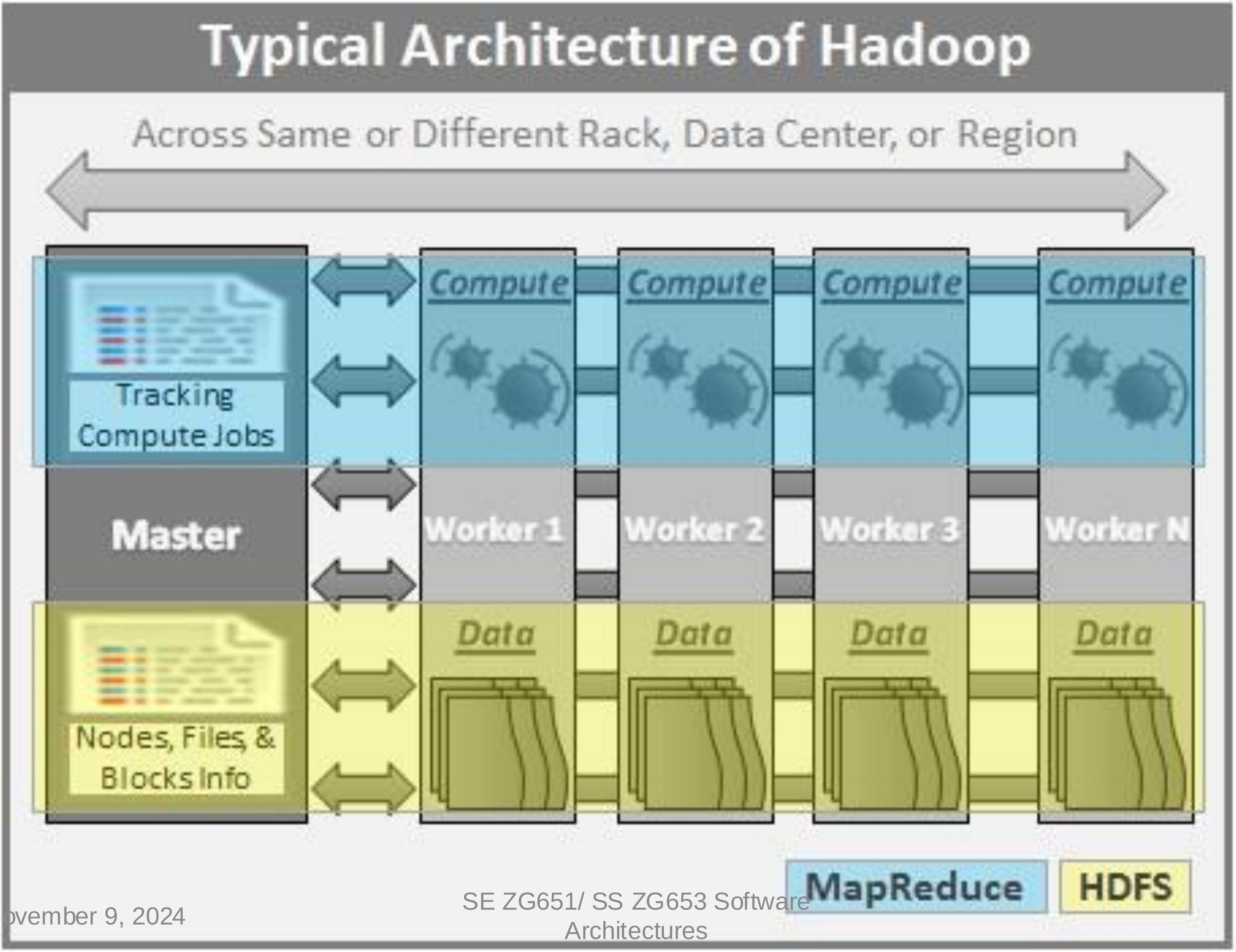
Sources

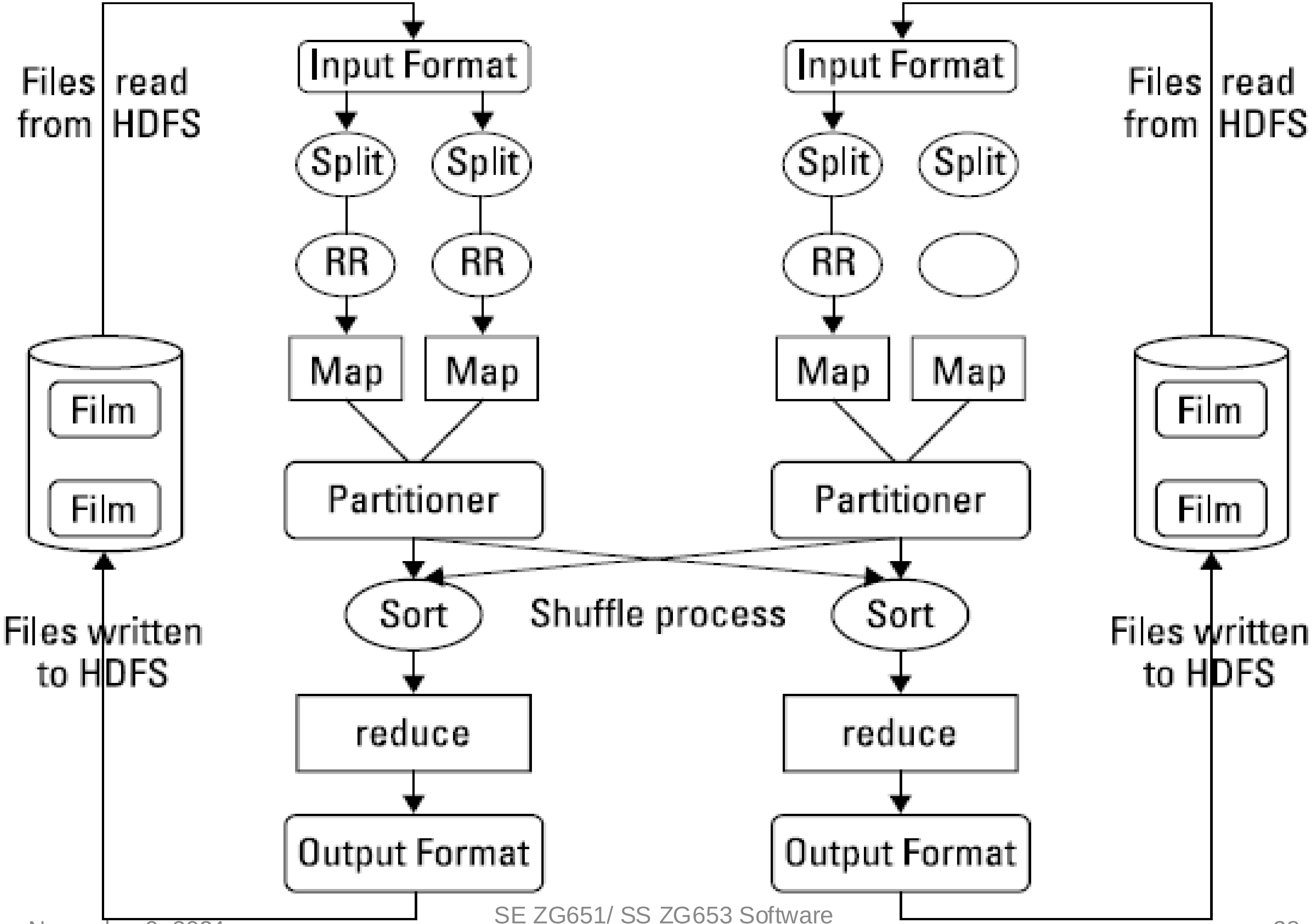
Hadoop

https://www.mssqltips.com/sqlserverauthor/77/dattatrey-sindol/

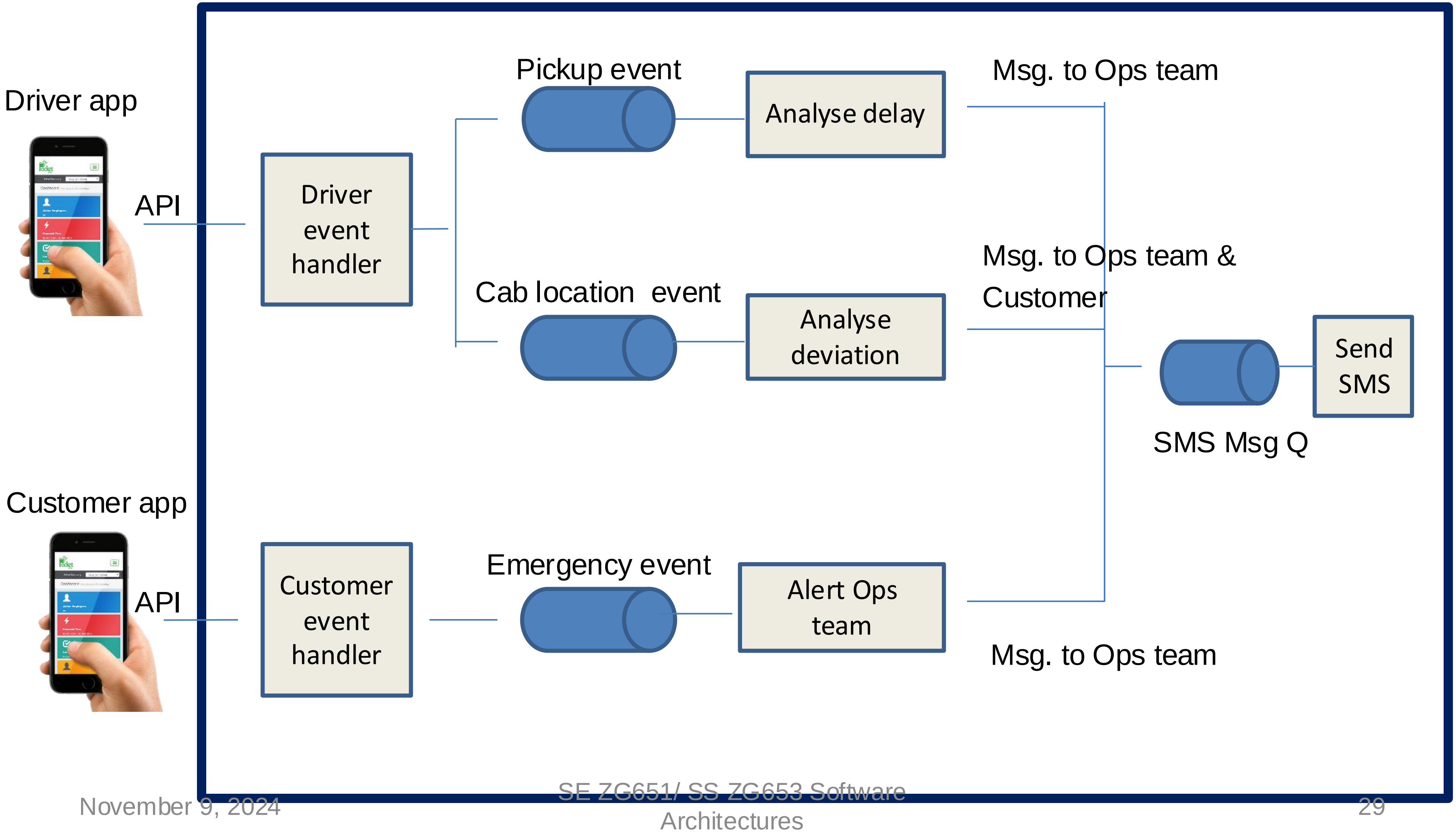
NoSQL Database

https://www.thoughtworks.com/insights/blog/nosql-databases-overview


Typical Architecture of Hadoop
Across Same or Different Rack, Data Center, or Region
Tracking Compute Jobs
Master
Nodes, Files & Blocks Info
Compute
Compute
Compute
Compute
Worker 1
Worker 2
Worker 3
Worker N
Data
Data
Data
Data
MapReduce
HDFS


Files read from HDFS
Input Format
Split
Split
RR
RR
Map
Map
Partitioner
Film
Film
Input Format
Split
Split
RR
Map
Map
Partitioner
Files read from HDFS
Film
Film
Sort
Shuffle process
Sort
Files written to HDFS
Files written to HDFS
reduce
reduce
Output Format
Output Format

Example of Stream processing in a cab hailing company like Ola / Uber



Analytics

Data Visualization

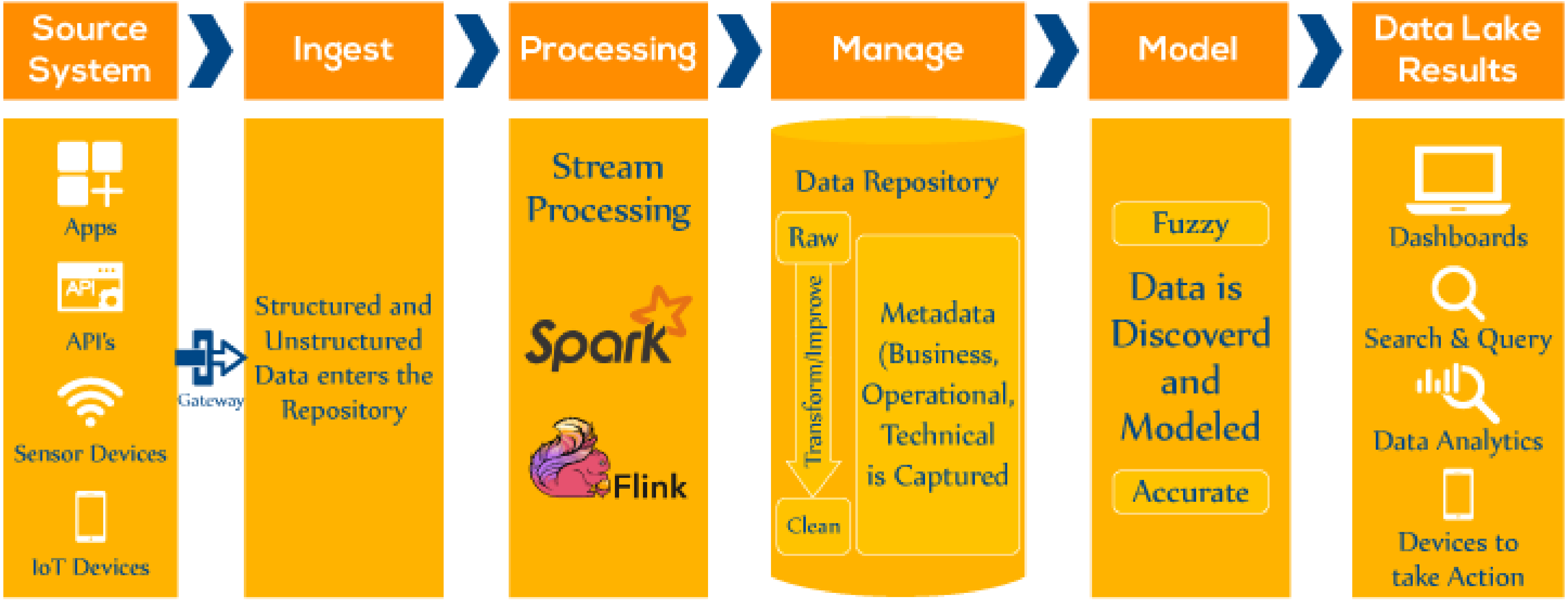
Multi-dimensional data

Data mining

Examples…

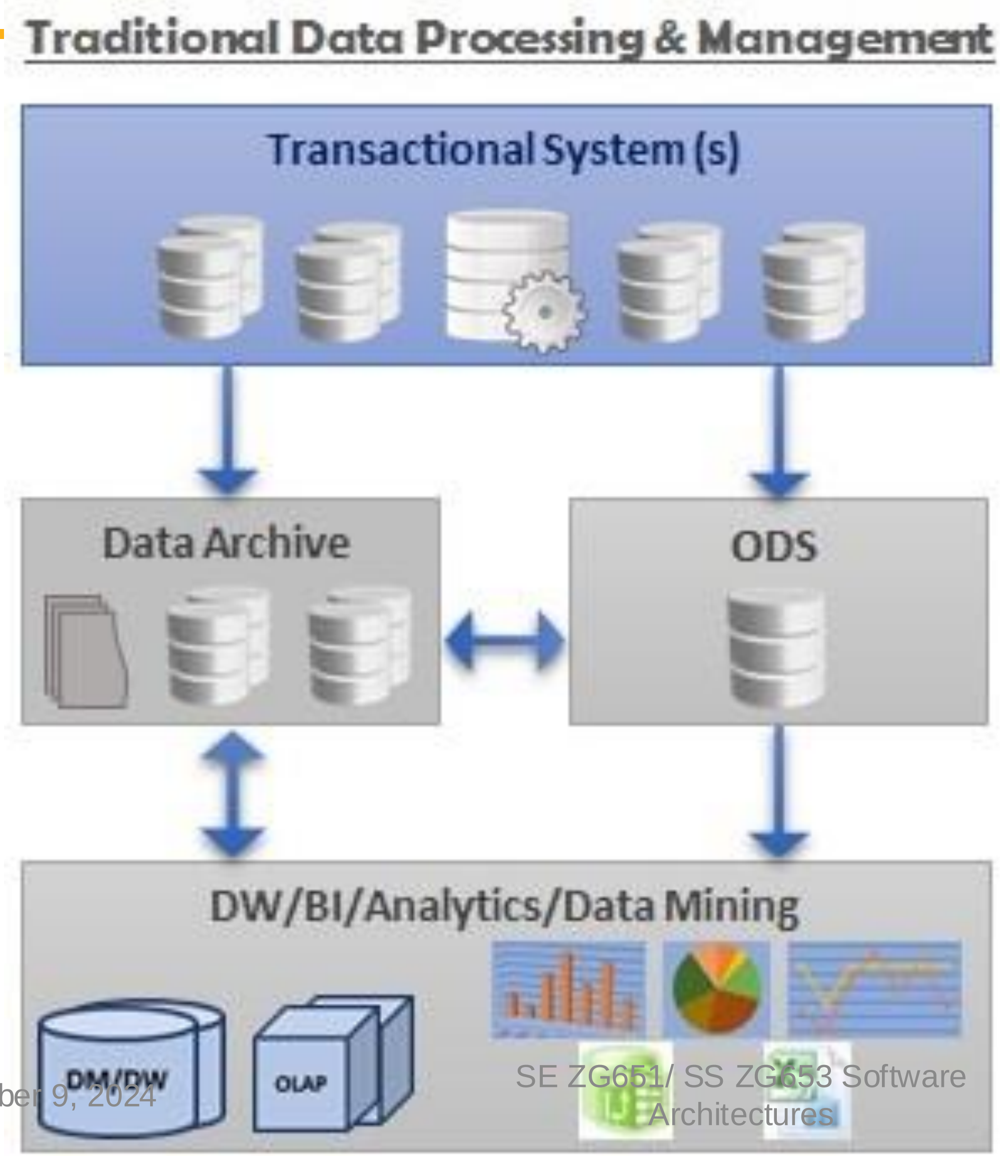
Tools ….

Real-Time Streaming and Data Analytics For IoT



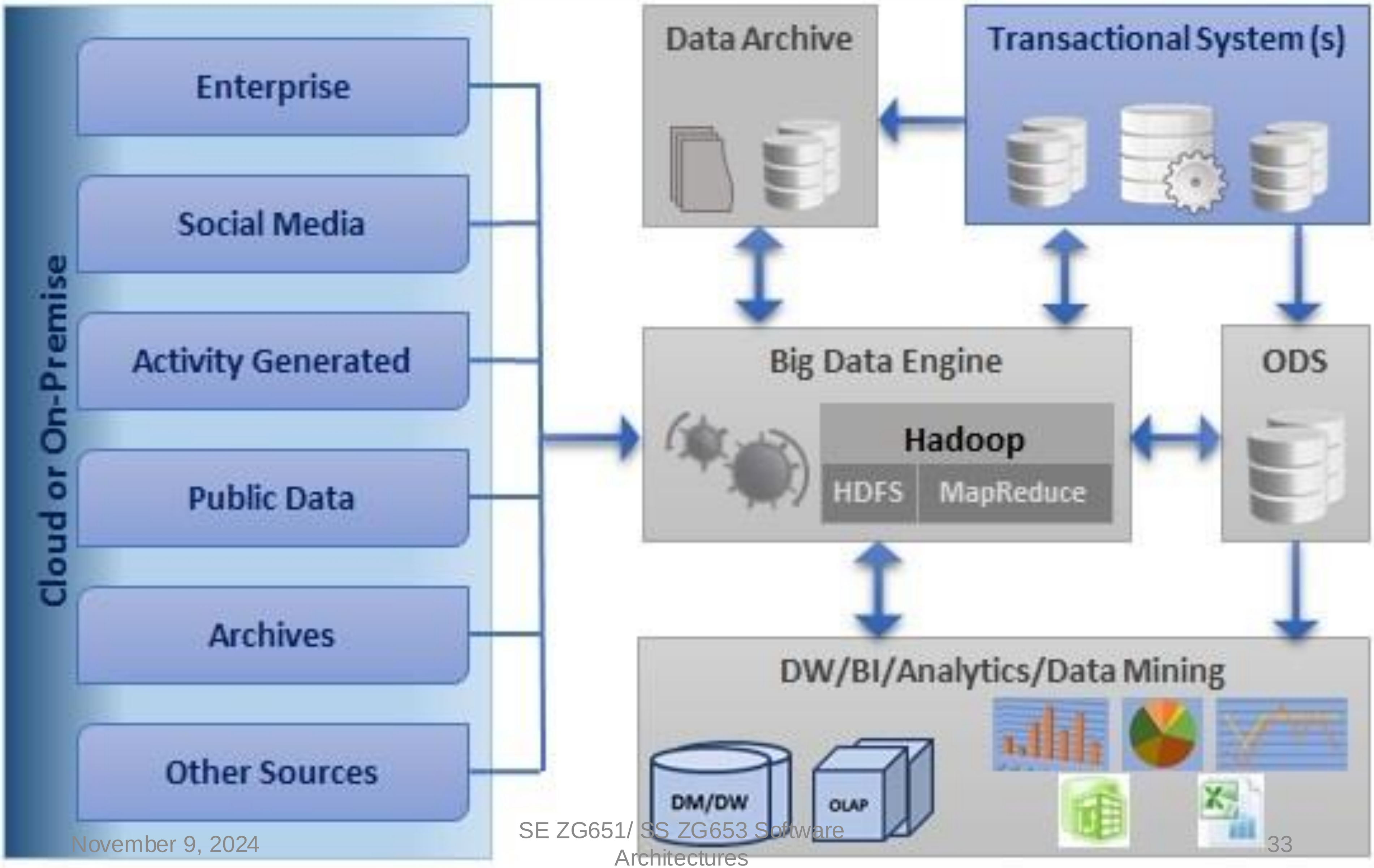
Source: https://www.xenonstack.com/blog/big-data-engineering/real-time-streaming-analytics-tools/

Traditional Data Processing


Traditional Data Processing & Management
Transactional System (s)
Data Archive
ODS
DW/BI/Analytics/Data Mining
DM/DW
OLAP

ODS: Operational Data Store

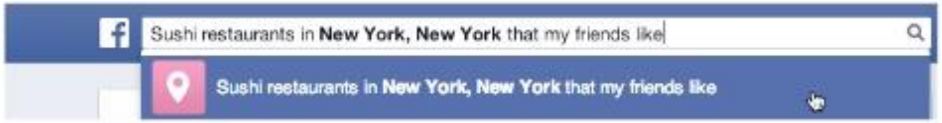
Modern (Next Generation) Data Processing & Management



Use cases of Stream Processing

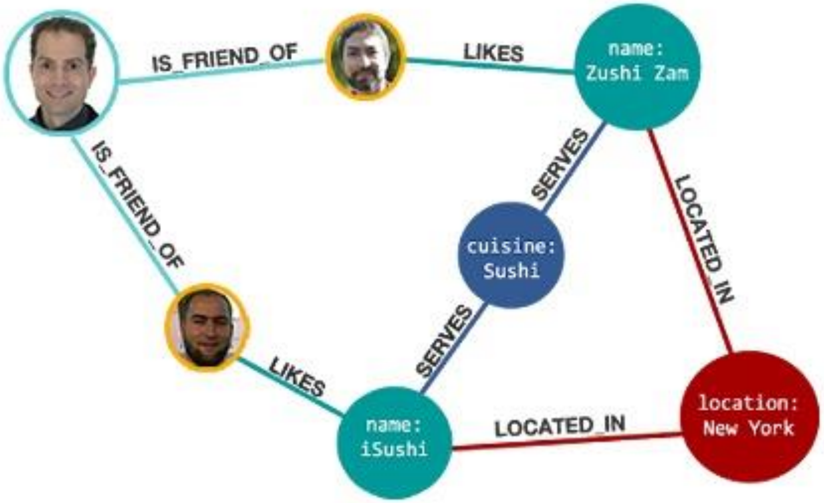
Following are some of the use cases.

- Algorithmic Trading, Stock Market Surveillance,
- Smart Patient Care
- Monitoring a production line
- Supply chain optimizations
- Intrusion, Surveillance and Fraud Detection (e.g. Uber)
- Most Smart Device Applications : Smart Car, Smart Home ..
- Smart Grid—(e.g. load prediction and outlier plug detection see Smart grids, 4 Billion events, throughout in range of 100Ks)
- Traffic Monitoring, Geo fencing, Vehicle and Wildlife tracking—e.g. TFL London Transport Management System
- Sport analytics—Augment Sports with realtime analytics (e.g. this is a work we did with a real football game (e.g. Overlaying realtime analytics on Football Broadcasts)
- Context-aware promotions and advertising
- Computer system and network monitoring
- Predictive Maintenance, (e.g. Machine Learning Techniques for Predictive Maintenance)
- Geospatial data processing

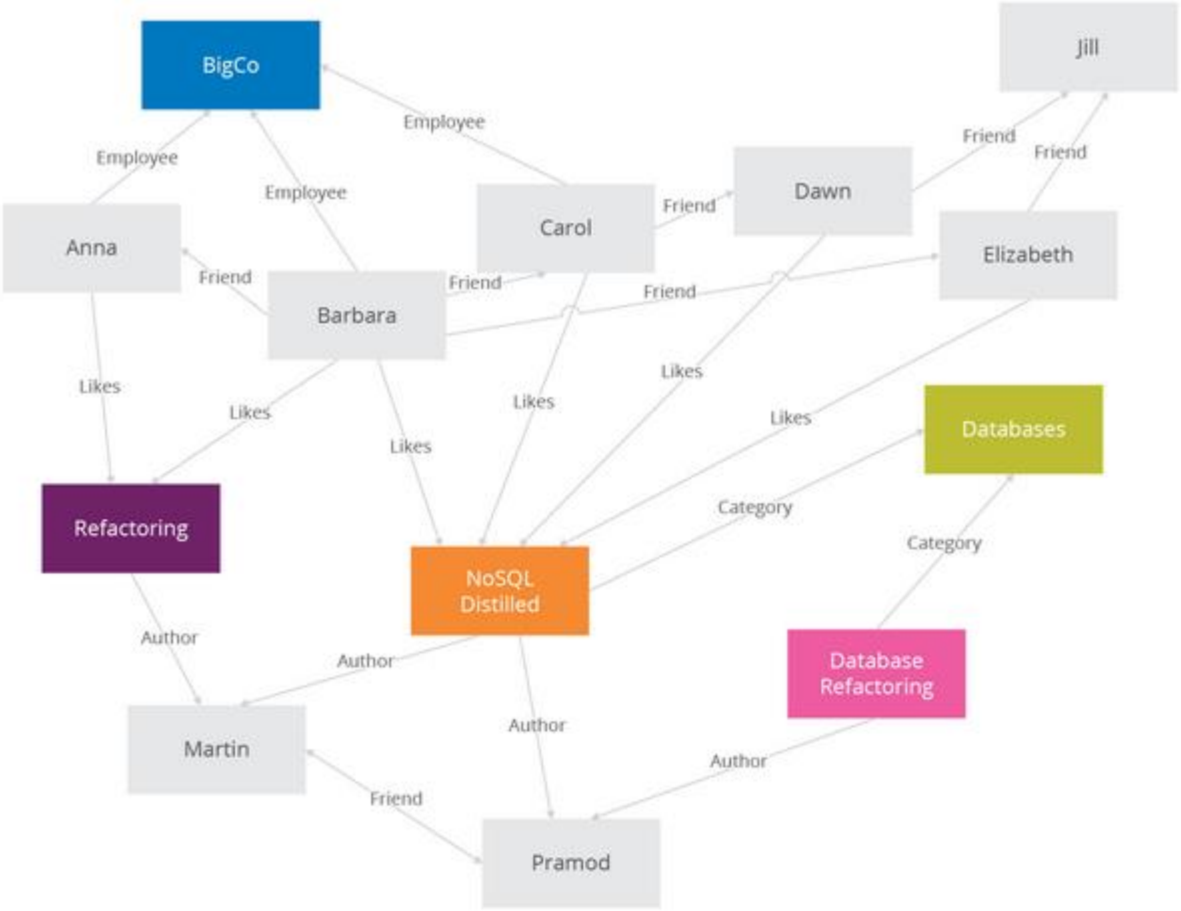


Sushi restaurants in **New York, New York** that my friends like

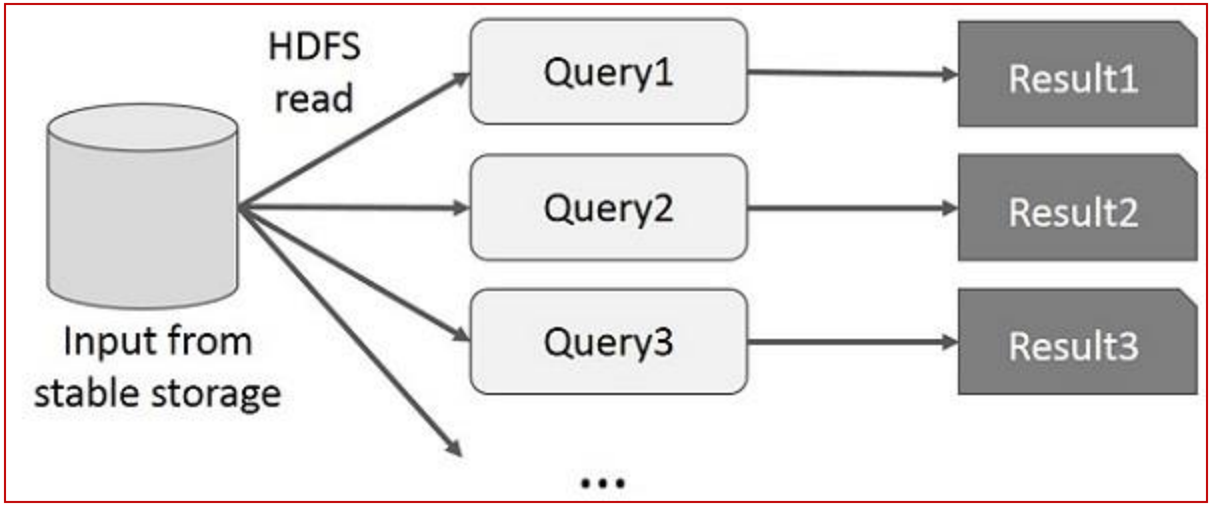
Sushi restaurants in **New York, New York** that my friends like



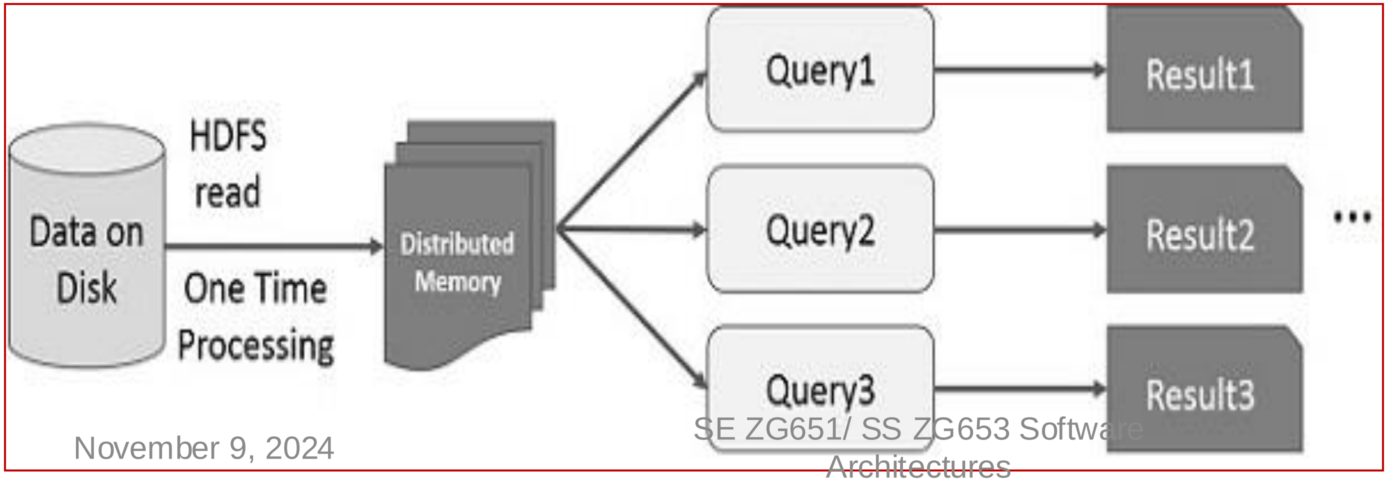
Graph database



Difference between Hadoop & Spark…



Interactive operation using Hadoop



Interactive operation using Spark

Source: https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm

Real-time analytics

Detecting bank fraud requires real-time analytics as events happen

Such situations demand processing of each event as they happen rather processing a batch of data on disk

This led to tools such as Spark Streams and Storm which support in-memory processing, than disk based processing

Storm for real time computing

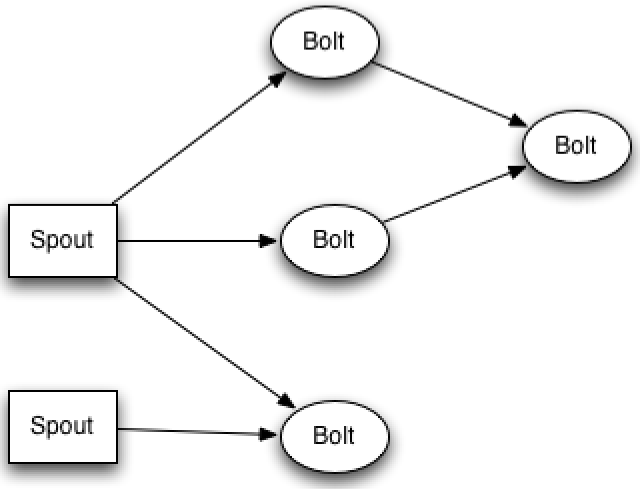
Apache Storm is a free and open source **distributed real time computation system**. Storm makes it easy to reliably process unbounded streams of data, doing for realtime processing what Hadoop did for batch processing. Storm is simple, can be used with any programming language, and is a lot of fun to use!

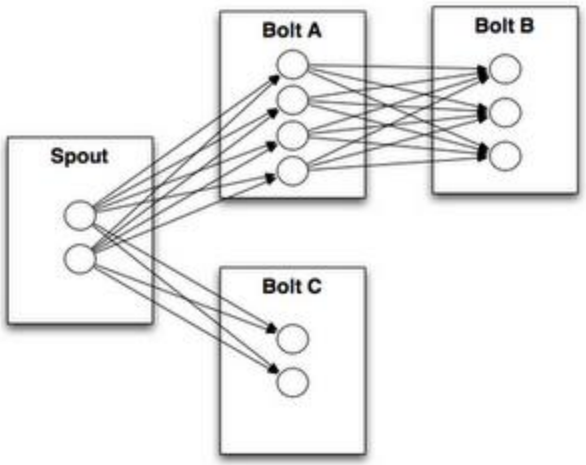
Storm has many **use cases**: **real time analytics, online machine learning, continuous computation**, distributed RPC, ETL, and more. Storm is fast: a benchmark clocked it at over a million tuples processed per second per node. It is scalable, fault-tolerant, guarantees your data will be processed, and is easy to set up and operate.

Storm **integrates with the queueing and database technologies** you already use. A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed. Read more in the tutorial.

Source: http://storm.apache.org/

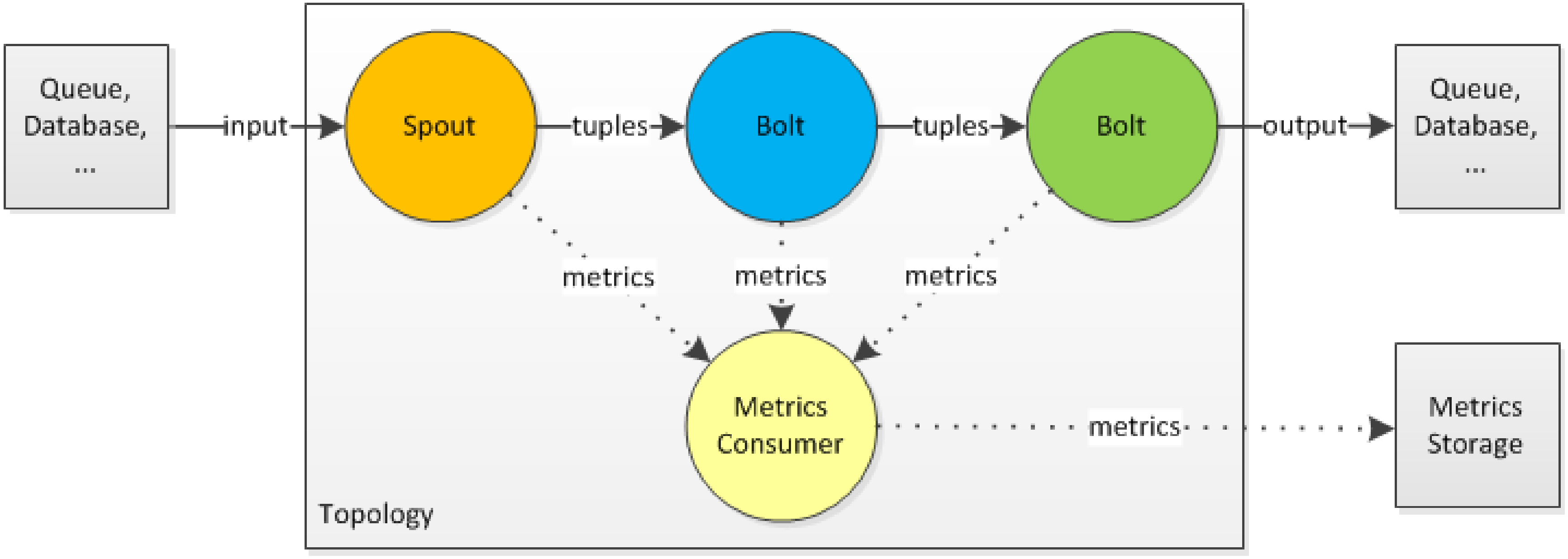
Storm topology



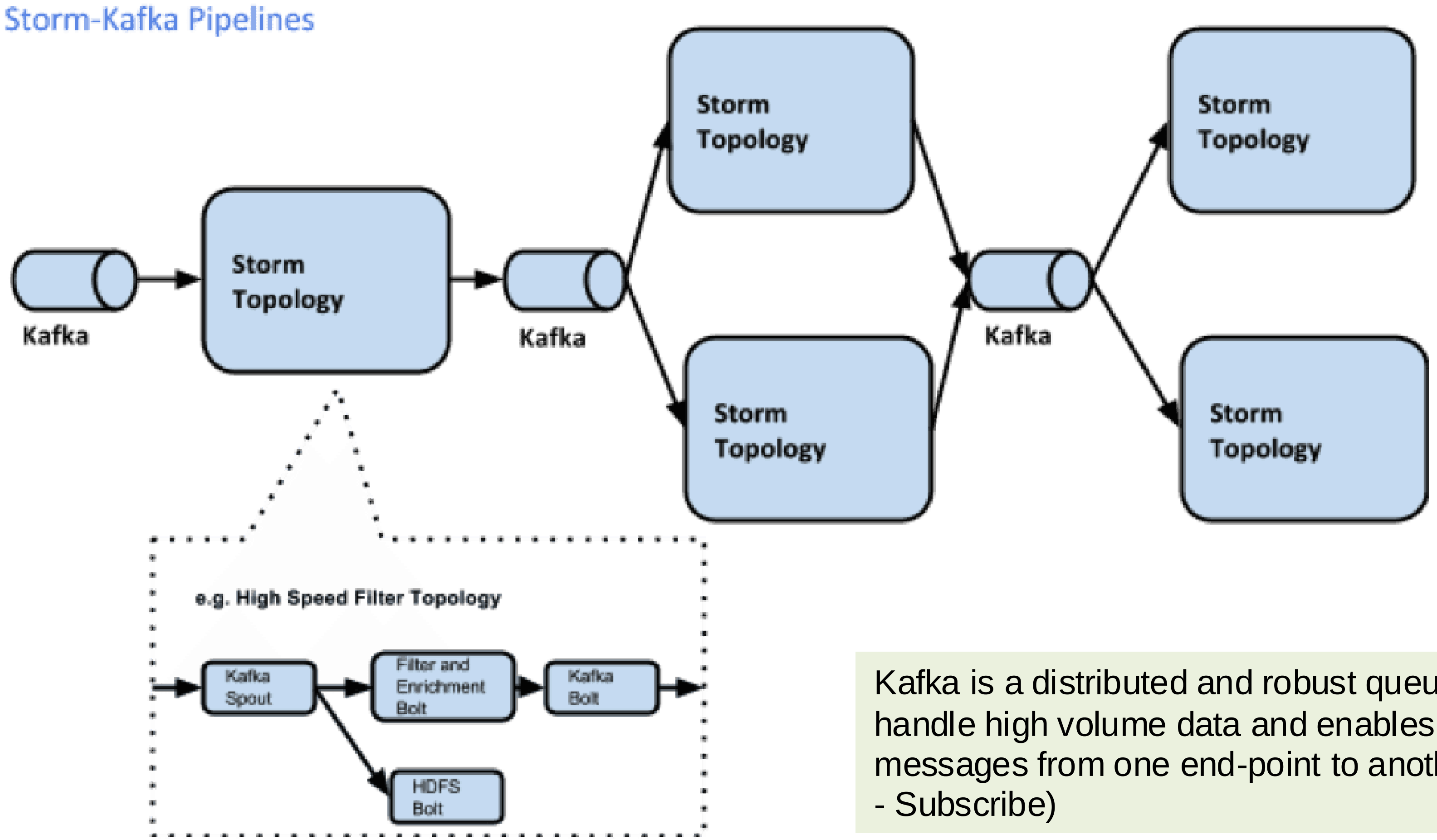


Twitter uses Storm for real-time analytics, personalization, search, revenue optimization
Groupon uses Storm for Real-time data integration systems
Yahoo! Uses Storm for processing user events, content feeds, and application logs.

Architecture of Storm system



Combining Kafka and Storm for real time computing



Kafka is a distributed and robust queue that can handle high volume data and enables you to pass messages from one end-point to another (Publish - Subscribe)

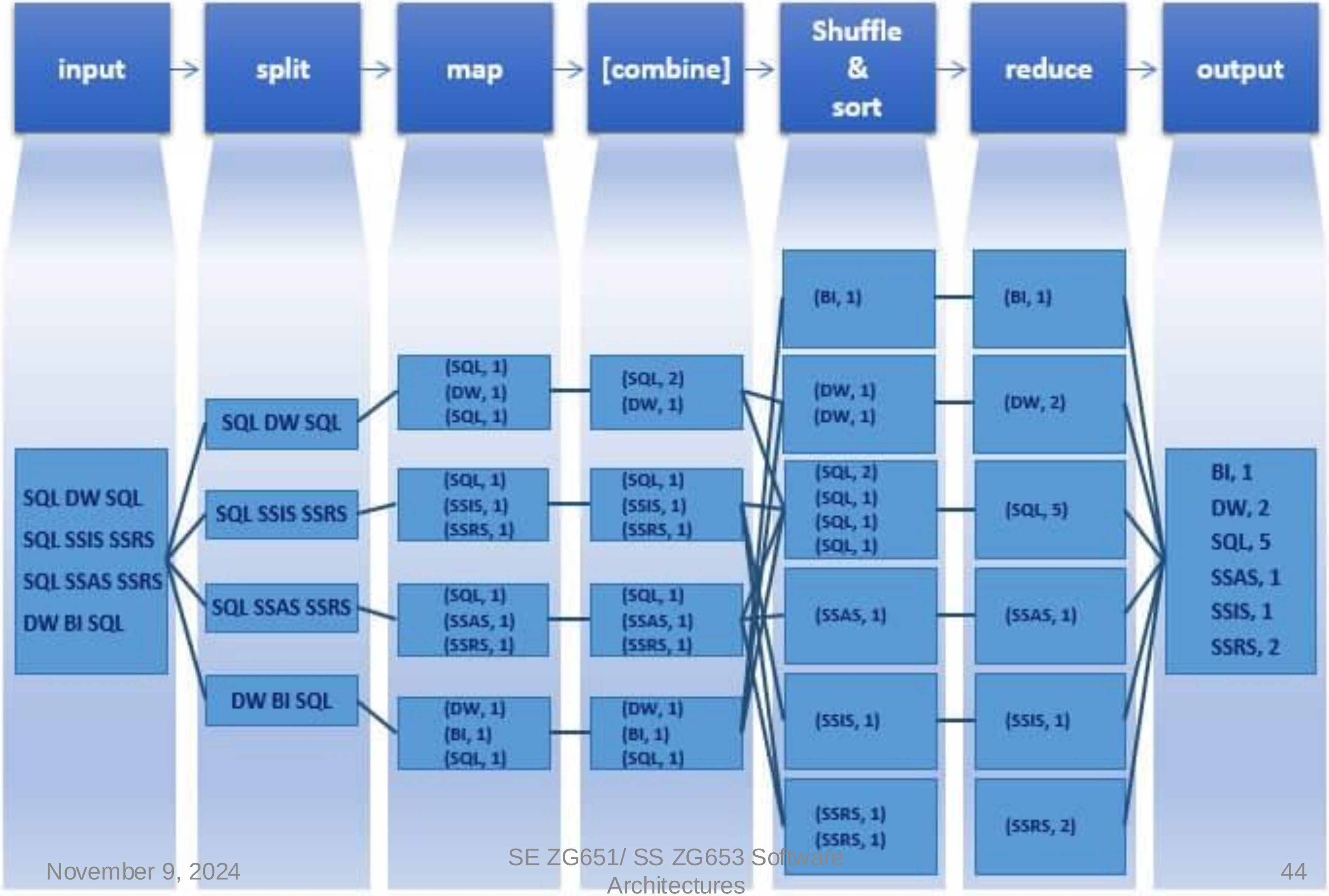
Source: https://hortonworks.com/blog/storm-kafka-together-real-time-data-refinery/

Storm & Kafka

The common flow of these tools (as I know it) goes as follows:

real-time-system --> Kafka --> Storm --> NoSql --> BI(optional)

MapReduce – Word Count Example Flow


input
split
map
[combine]
Shuffle & sort
reduce
output
SQL DW SQL
SQL SSIS SSRS
SQL SSAS SSRS
DW BI SQL
SQL DW SQL
SQL SSIS SSRS
SQL SSAS SSRS
DW BI SQL
(SQL, 1) (DW, 1) (SQL, 1)
(SQL, 1) (SSIS, 1) (SSRS, 1)
(SQL, 1) (SSAS, 1) (SSRS, 1)
(DW, 1) (BI, 1) (SQL, 1)
(SQL, 2) (DW, 1)
(SQL, 1) (SSIS, 1) (SSRS, 1)
(SQL, 1) (SSAS, 1) (SSRS, 1)
(DW, 1) (BI, 1) (SQL, 1)
(BI, 1)
(DW, 1) (DW, 1)
(SQL, 2) (SQL, 1) (SQL, 1) (SQL, 1)
(SSAS, 1)
(SSIS, 1)
(SSRS, 1) (SSRS, 1)
(BI, 1)
(DW, 2)
(SQL, 5)
(SSAS, 1)
(SSIS, 1)
(SSRS, 2)
BI, 1
DW, 2
SQL, 5
SSAS, 1
SSIS, 1
SSRS, 2