# NoSQL Databases Overview

---

## Introduction

Traditional relational databases (SQL databases) are ideal for transaction management but aren't suitable for all types of applications. Many web applications, IoT solutions, and systems needing high data processing speed but no transaction management turn to **NoSQL databases** as an alternative.

---

## Characteristics of NoSQL Databases

NoSQL databases offer several benefits, including:

- **Scalability** (Sharding)
- **Speed** (In-Memory)
- **High Availability** (Replication)
- **Flexibility** to handle semi-structured and unstructured data

However, NoSQL databases often lack traditional SQL features like:

- **ACID Transactions** (Atomicity, Consistency, Isolation, Durability)
- **Join operations**

---

## Real-World Examples of NoSQL Use

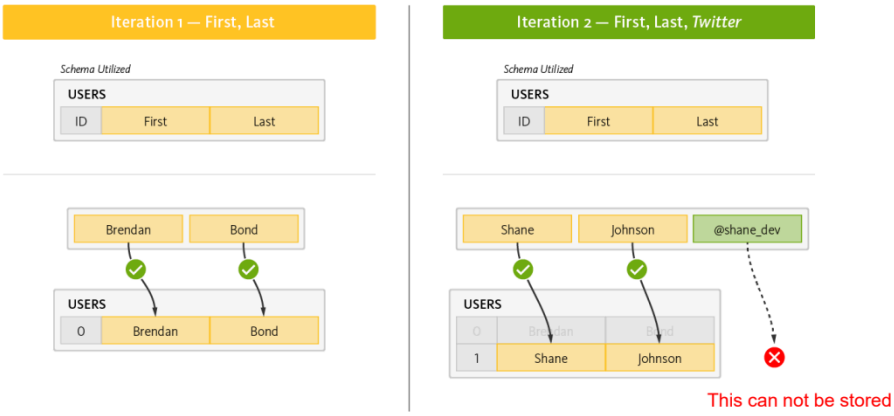
1. **Tesco**: Manages millions of products and promotions.
2. **Ryanair**: Supports its mobile app for over 3 million users.
3. **Marriott**: Uses NoSQL for a $38 billion reservation system.
4. **Gannett (USA Today)**: Uses NoSQL for a custom content management system.
5. **GE**: Employs NoSQL for the Predix platform in the Industrial Internet.

---

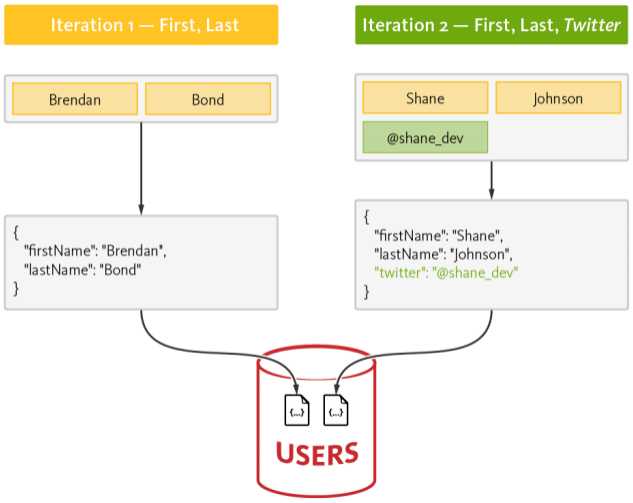
## Benefits of NoSQL Databases

*Diagram Placeholder: NoSQL - Flexibility*

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



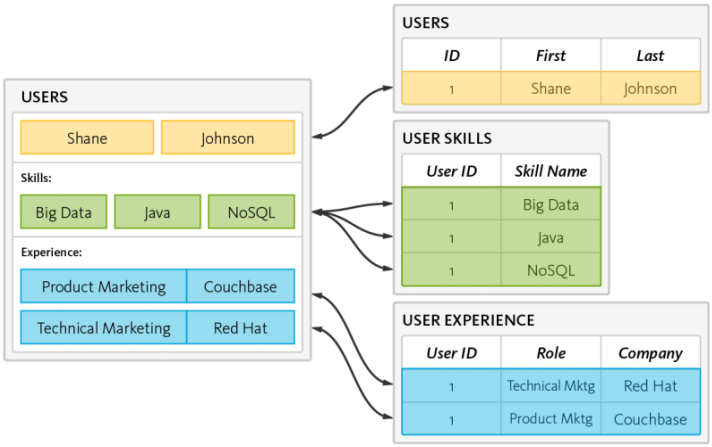

In NoSQL DB, we can easily add new columns.



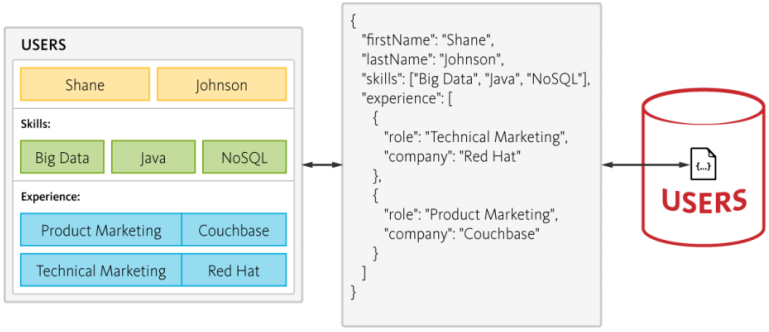

**Flexibility**

In relational databases, modifying the schema (e.g., adding a new column) requires changing the entire structure. In contrast, NoSQL databases easily accommodate new fields without modifying the overall schema.

*Diagram Placeholder: NoSQL - Simplicity*



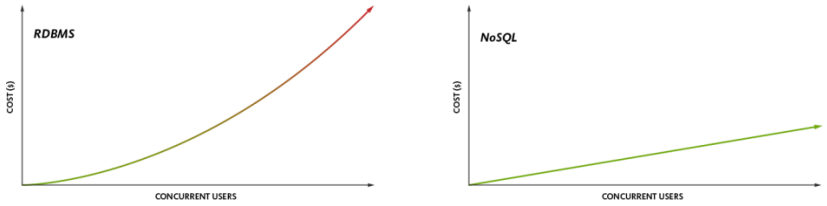



**Simplicity**

NoSQL offers straightforward structures, making it simpler to store and retrieve data without complex query operations.

*Diagram Placeholder: NoSQL - Cost-Effectiveness*




Cost: Memory, CPU, storage

**Cost-Effectiveness**

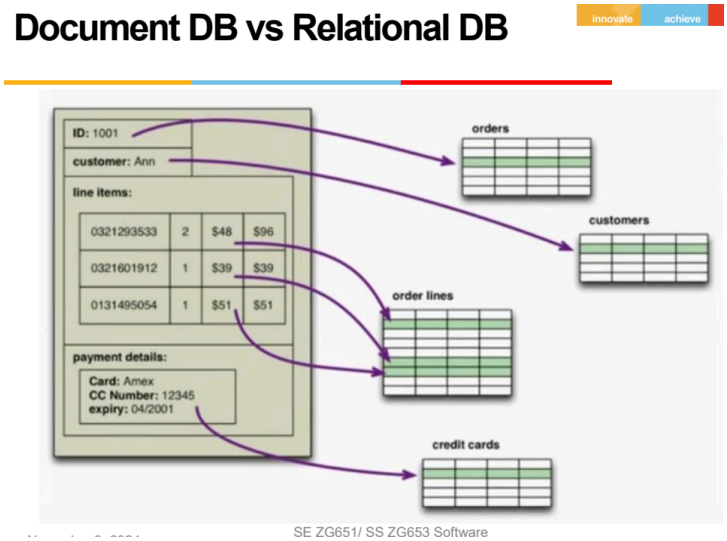
NoSQL solutions typically have lower hardware and operational costs, especially for high-volume data applications.

---

## Types of NoSQL Databases

1. **Document Databases**
   - **Structure**: Stores data in document formats (e.g., JSON, BSON).
   - **Examples**: MongoDB, CouchDB.
   - **Use Cases**: Content management, e-commerce.
2. *Diagram Placeholder: Example of a Document Record*


Example of one record



3. **Key-Value Databases**
    - **Structure**: Simple key-value pairs; efficient for fast retrieval.
    - **Examples**: Redis, Amazon DynamoDB.
    - **Use Cases**: Caching, session management.
4. *Diagram Placeholder: Example of a Key-Value Database*

### 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 |

5. **Column-Oriented Databases**
    - **Structure**: Data stored in columns, suitable for analytical queries.
    - **Example**: Calculate average electricity usage in a specific region.
    - **Use Cases**: Data analysis, summarization.
6. *Diagram Placeholder: Column-Oriented Database Example*

## Column oriented database

innovate achieve

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

Instead if we store the data as follows:

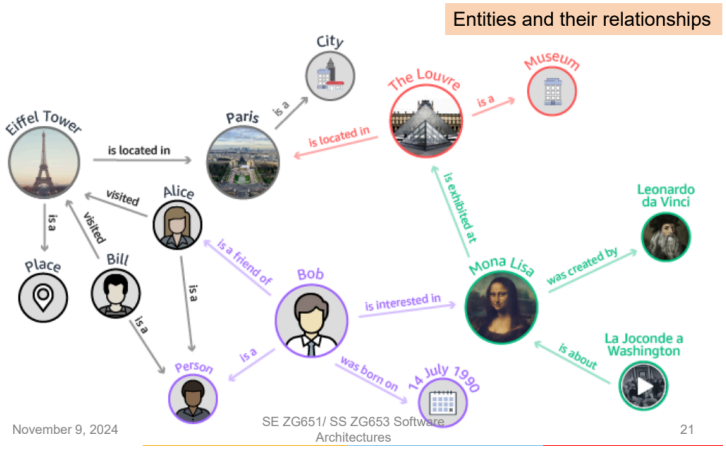
Record 1: 001, John Mancha, Addr 1, East // Customer details
Record 2: 002, Vivek Kulkarni, Addr 2, East

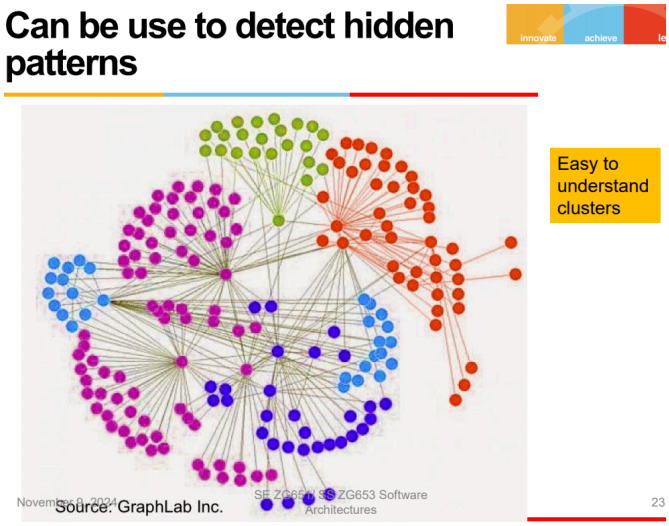
Record A: Jan, 2018, 100, 90, 110, … // Usage – in customer order
Record B: Feb, 2018, 110, 92, 115, …

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

Good for summarization

7. **Graph Databases**
   - **Structure**: Stores data as nodes (entities) and edges (relationships).
   - **Examples**: Neo4J, OrientDB.
   - **Use Cases**: Fraud detection, social networks.
8. *Diagram Placeholder: Graph Database with Entities and Relationships*





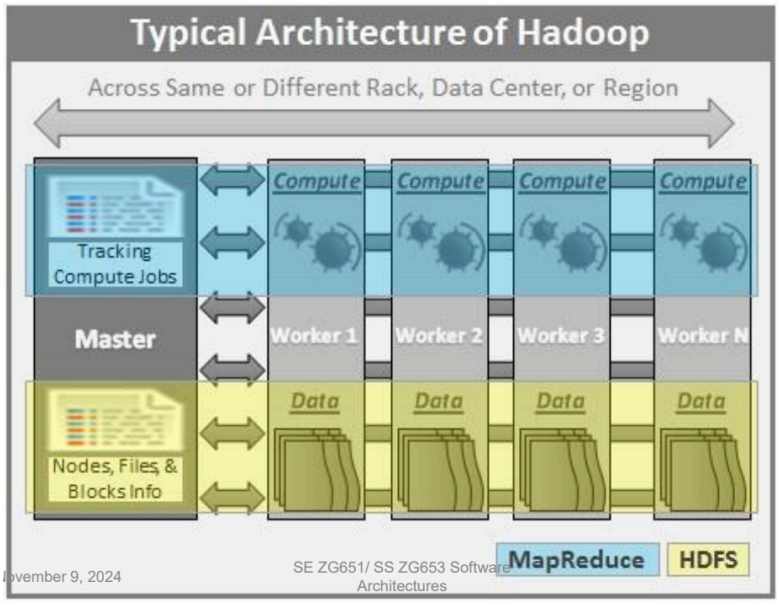

---

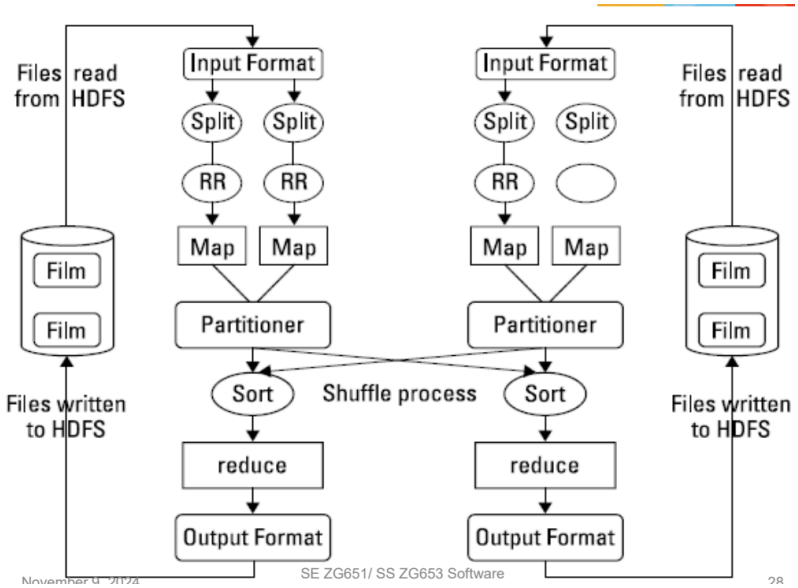
## In-Memory Databases (IMDB)

An **in-memory database** relies on main memory for data storage, offering extremely fast response times. It's useful for applications requiring high-speed data retrieval, such as telecommunications and mobile advertising networks.

**Examples**:

- SAP HANA
- IBM DB2 BLU
- Oracle





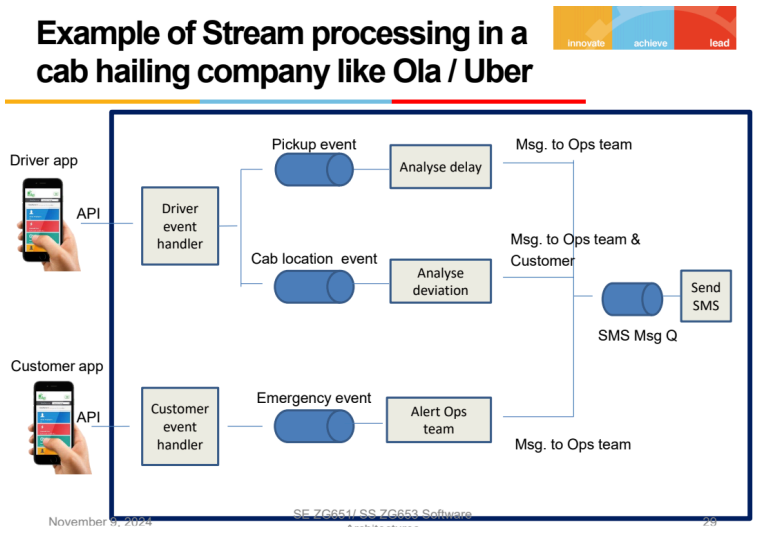


## Stream Processing in NoSQL

NoSQL databases, coupled with real-time data processing tools, support applications needing rapid event handling.

**Example: Cab Hailing Service (e.g., Uber or Ola)**

*Diagram Placeholder: Stream Processing Architecture in Cab Hailing*





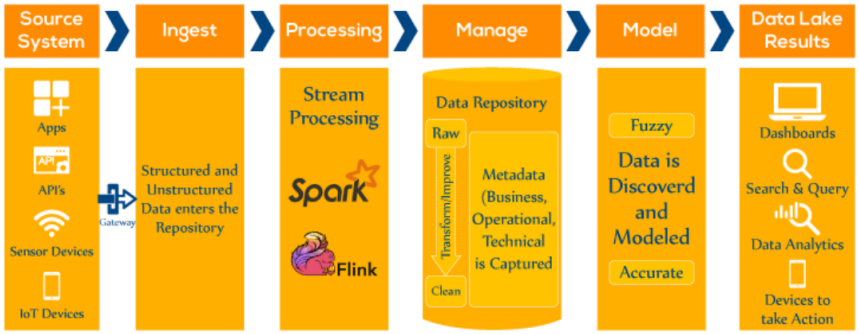
- **Driver Events**: Process location, emergency events.
- **Customer Notifications**: Send alerts based on driver's location or delays.
- **Operations Alerts**: Notify teams of issues or emergencies.

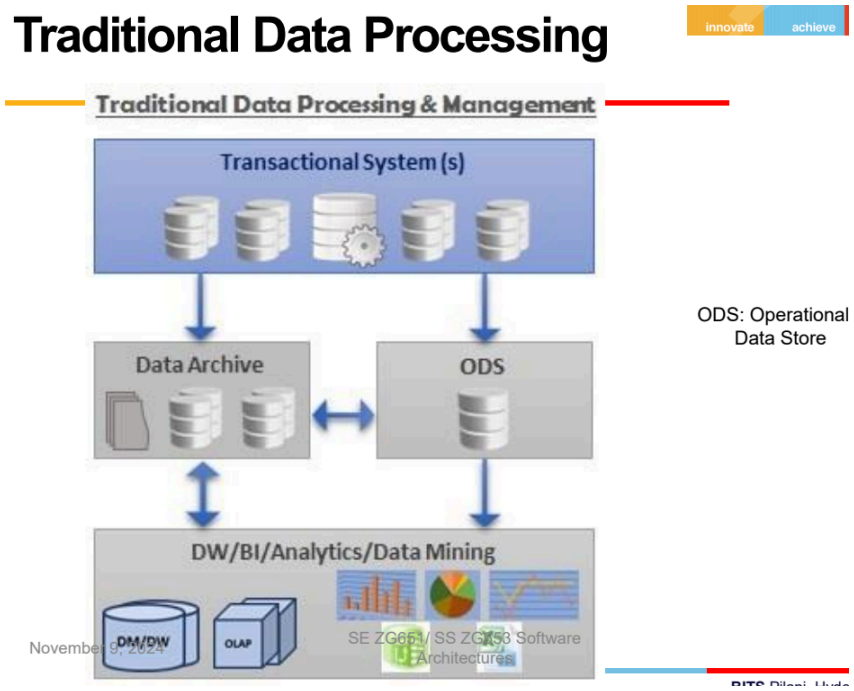
## Use Cases for Stream Processing

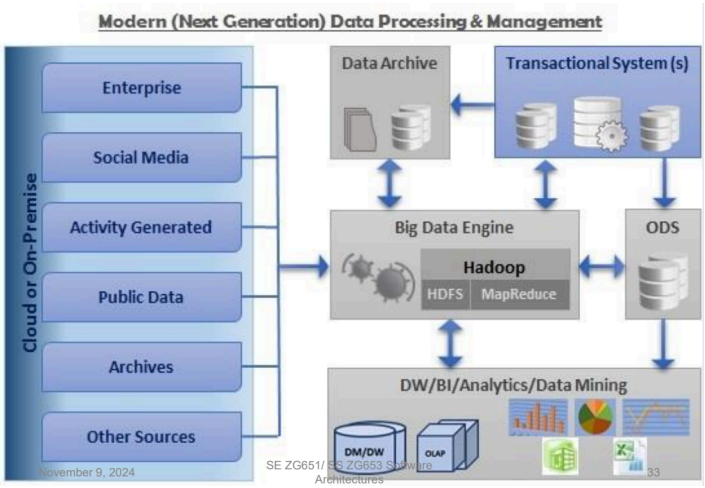
Stream processing in NoSQL supports various real-time applications:

- **Algorithmic Trading**: High-speed stock trading.
- **Smart Patient Care**: Continuous monitoring of patient health.
- **Supply Chain Optimization**: Real-time updates on logistics.
- **Fraud Detection**: Instantaneous detection of unusual transactions.
- **Traffic Monitoring**: Real-time traffic and geo-fencing alerts.

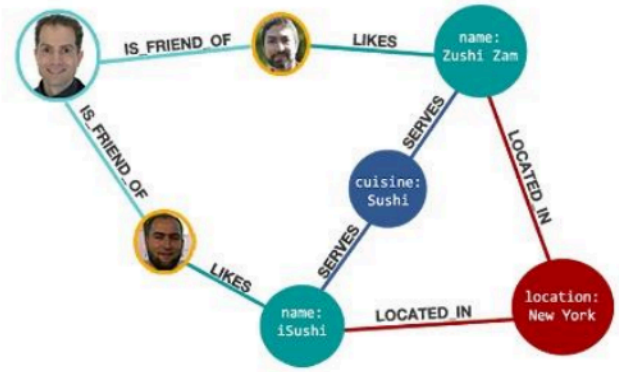
*Diagram Placeholder: Real-Time Streaming Use Cases*



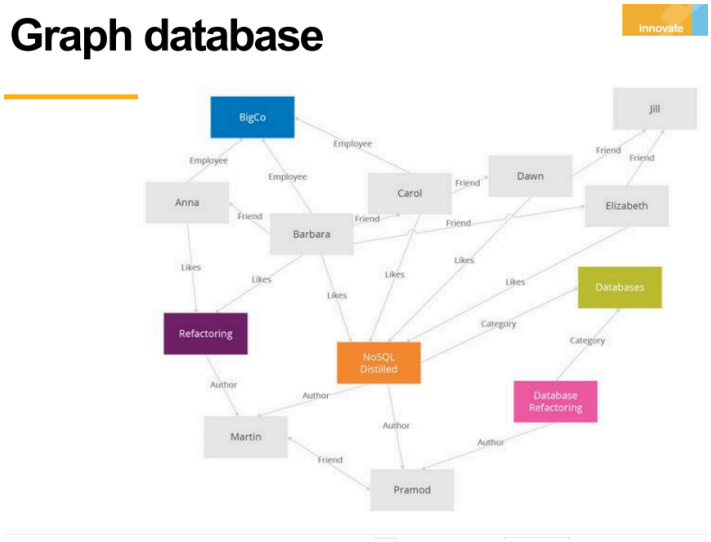



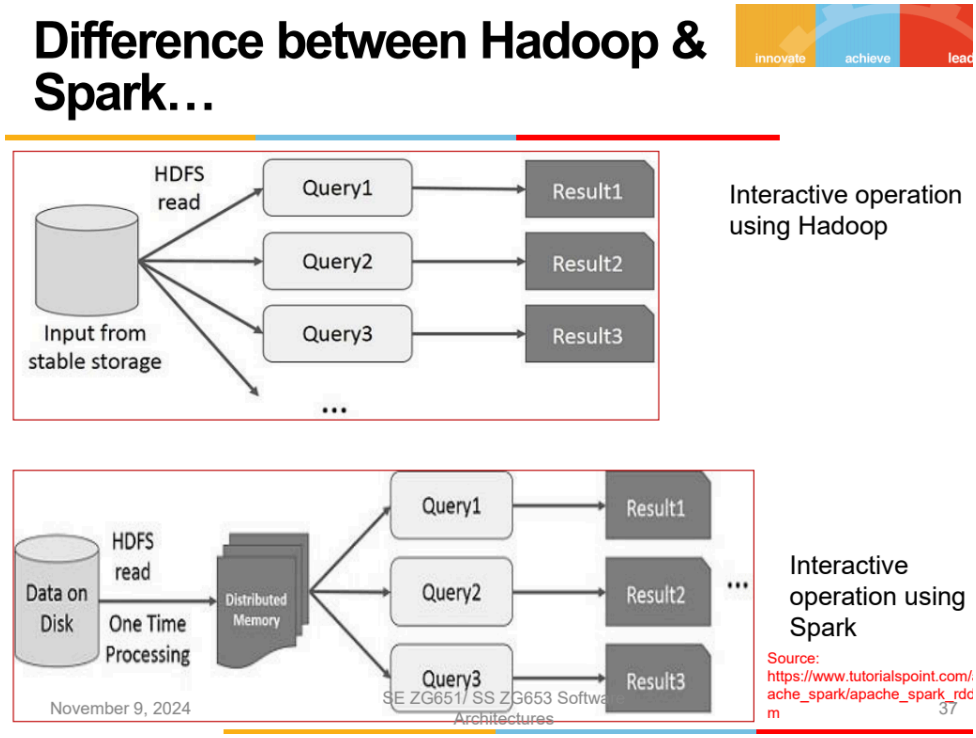







# Graph database



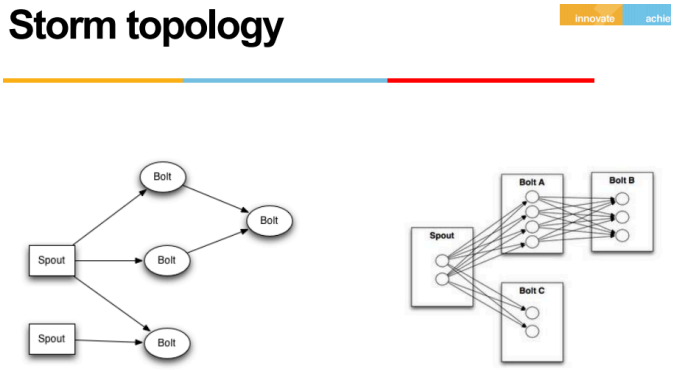


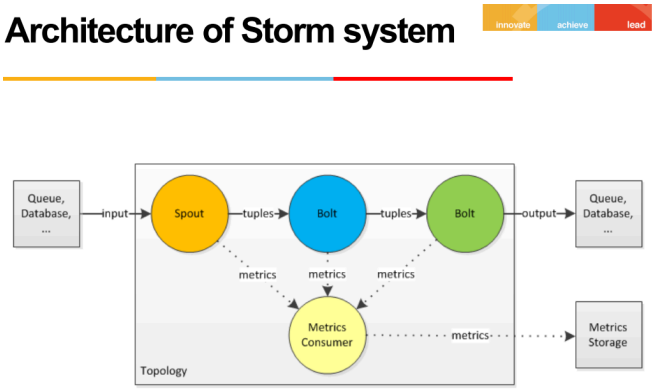

---

## Real-Time Analytics with Tools like Storm and Kafka

For applications requiring real-time processing, tools like Apache Storm and Kafka are commonly used.

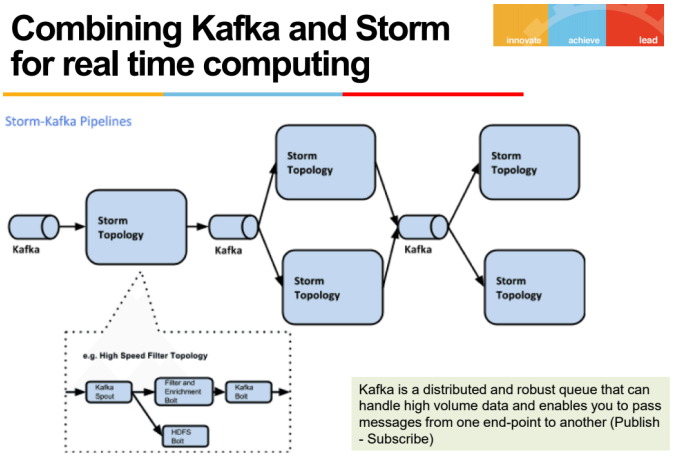
- **Storm**: Processes unbounded data streams in real time. It's scalable, fault-tolerant, and widely used by companies like Twitter for analytics.
  *Diagram Placeholder: Architecture of Storm System*






- **Kafka**: A robust, distributed messaging system. Kafka works as a publish-subscribe model to relay high-volume data between systems.

*Diagram Placeholder: Combining Kafka and Storm for Real-Time Computing*




**Example Workflow**

1. **Event Source** → Kafka → Storm → NoSQL → Analytics