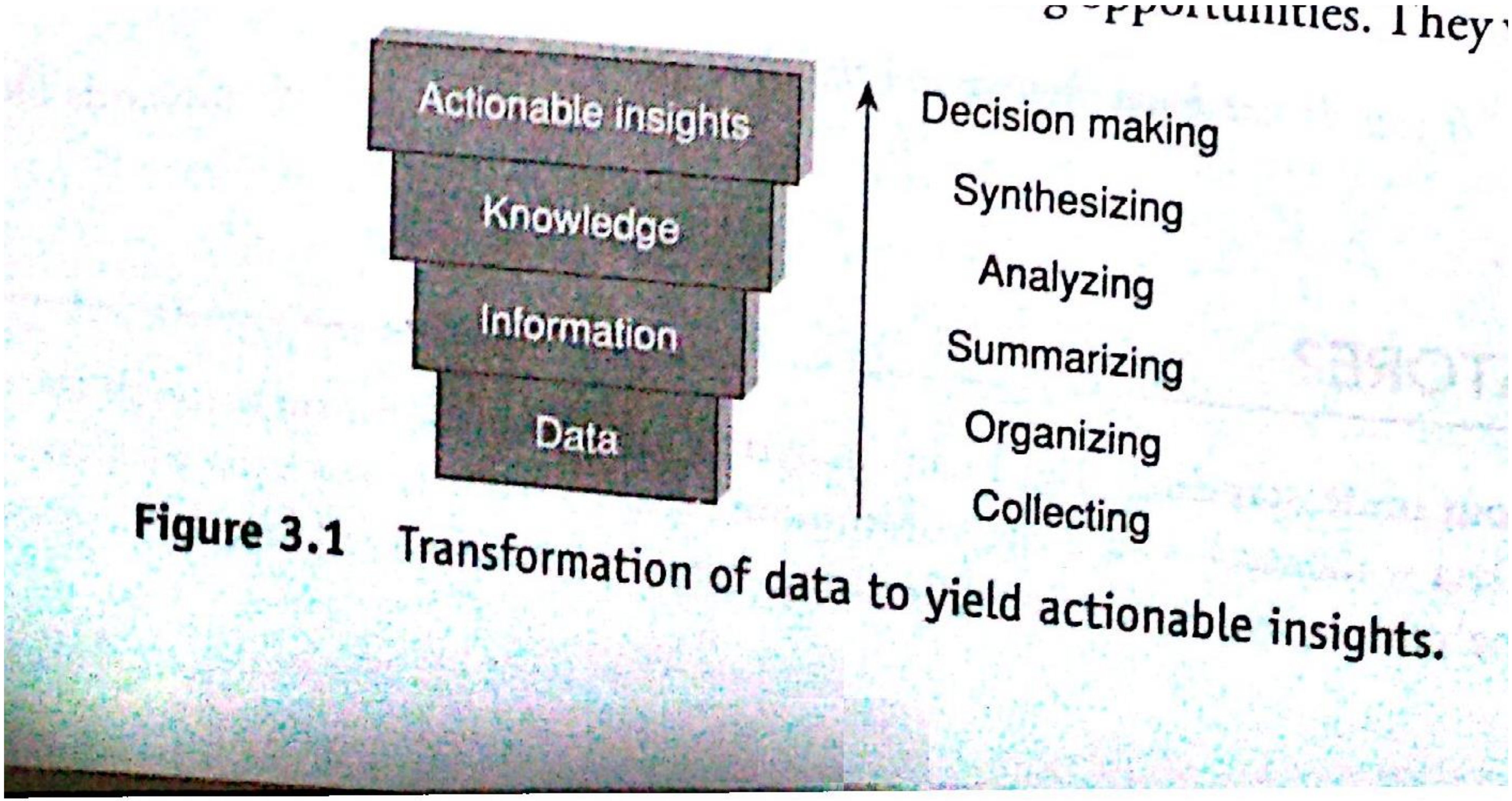
Day2
A.Baskar
ASE,CSE

Where do we Begin?



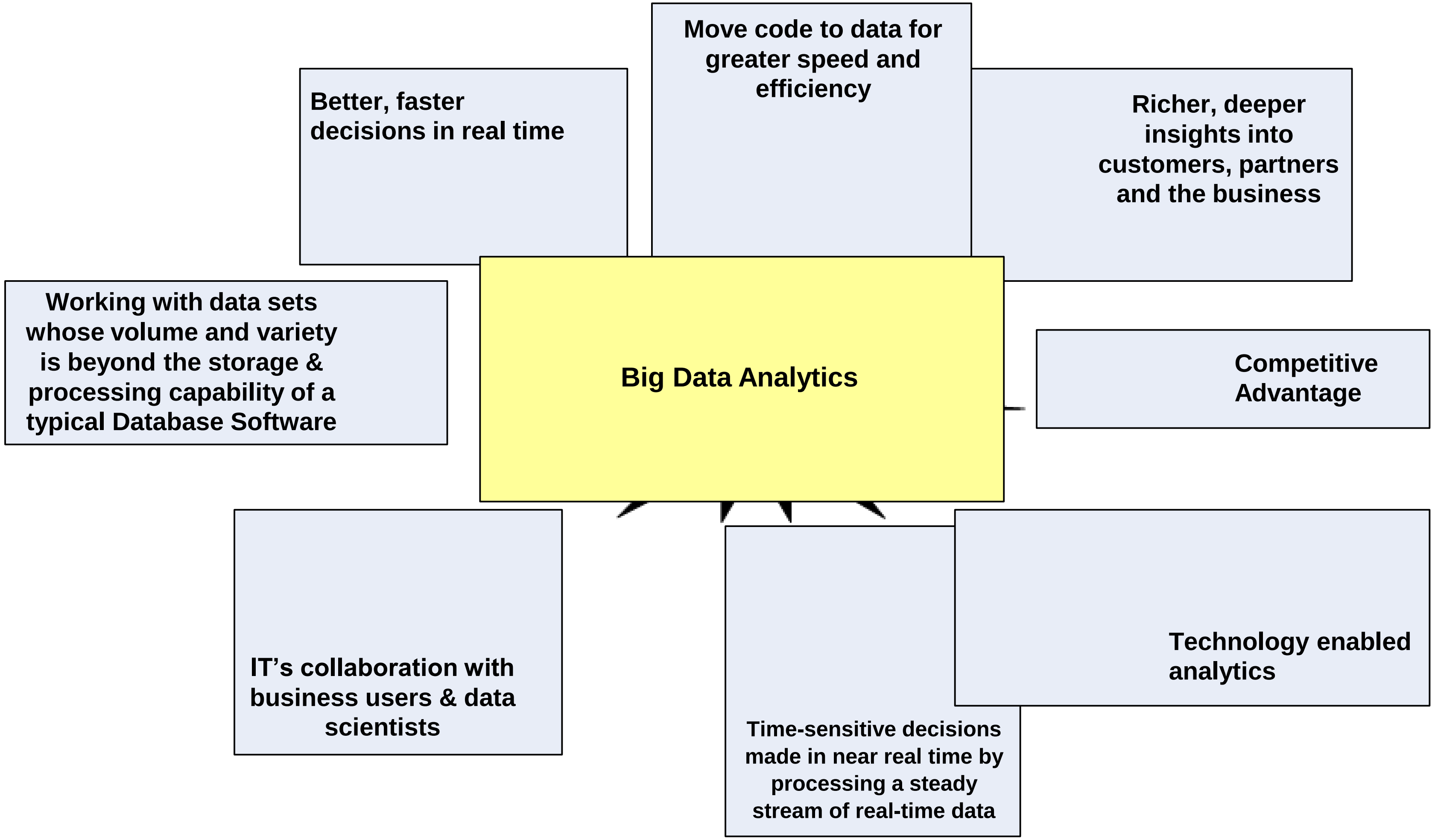
opportunities. They

Figure 3.1 Transformation of data to yield actionable insights.

What is Big Data Analytics?

- Big data analytics is the process of examine a big data to discover patterns , unearth features, finding correlation and finding other useful information for making faster and better decision.
- Technology enabled
- About gaining a meaningful, deeper and richer insights into your business to steer into the right direction, understanding customer demographics, better leveraging the services to vendors and suppliers.
- About a competitive edge over your company
- working with data set whose volume and variety exceed the storage and processing capacities of available infrastructure.
- About moving code to data.

What is Big Data Analytics?



Big Data Analytics

Better, faster decisions in real time

Move code to data for greater speed and efficiency

Richer, deeper insights into customers, partners and the business

Working with data sets whose volume and variety is beyond the storage & processing capability of a typical Database Software

Competitive Advantage

IT's collaboration with business users & data scientists

Time-sensitive decisions made in near real time by processing a steady stream of real-time data

Technology enabled analytics

What Big Data Analytics isn’t?

“One-size fit all” traditional RDBMS built on shared disk & memory

Only about Volume

Big Data Analytics isn’t ...

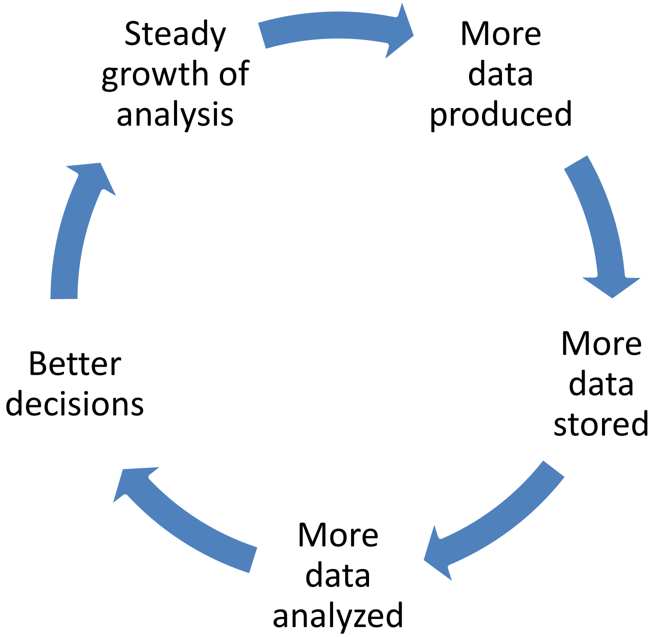
Just about Technology

Only used by huge online companies like Google or Amazon

Meant to replace RDBMS

Meant to replace Data Warehouse

Why this sudden hype around Big data analytics?



Steady growth of analysis

More data produced

More data stored

More data analyzed

Better decisions

Classification of analytics

- Two thoughts:

1.Classify analytics into basic, operationalized, advanced and monetized.

2.Classify analytics into analytics1.0,2.0 and3.0

- **Basic analytics:**
- Slicing and dicing data to help with basic business insights
- Based on reporting historical data ,visualization etc.

- **Operational Analytics**:
- Enterprise business

Classification of analytics

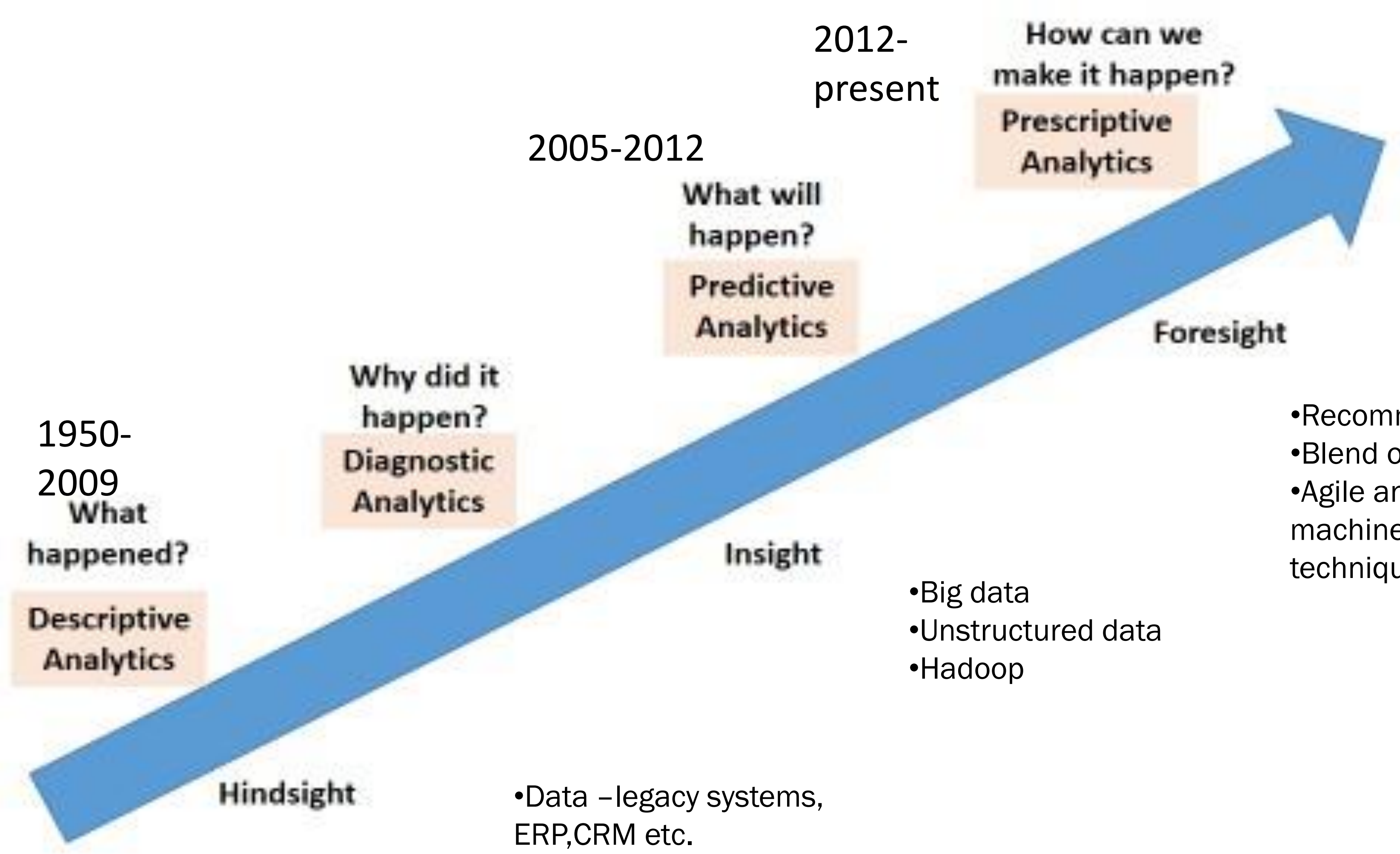
- Advanced analytics
- Forecasting for the future/Prediction

- Monetized Analytics:
- derive direct business revenue.

Analytics 1.0, 2.0 and 3.0

Analytics 1.0	Analytics 2.0	Analytics 3.0
Mid 1950 to 2009	2005-2012	2012-present
Descriptive statistics	Descriptive statistics + Predictive statistics	Descriptive+ Predictive+ Prescriptive
What happened? Why did it happen?	What will happen? Why will it happen?	What will happen? When it will happen? Why will it happen? What should be the action to taken?
Data from legacy system	Big data	A blend of big data and legacy systems
Small and structed data	Data is mainly unstructured	Big data + traditional
Data was internally stored	Externally stored	Both
Relational DBs	Hadoop cluster	Agile analytical methods, In memory analytics

Analytics 1.0, 2.0 and 3.0



1950-2009

2005-2012

2012-present

- Data –legacy systems, ERP,CRM etc.
- small and structured data .
- Data stored in DWH .
- Relational DBS

- Big data
- Unstructured data
- Hadoop

- Recommendations
- Blend of big data
- Agile analytical, machine learning techniques etc.

3.6 GREATEST CHALLENGES THAT PREVENT BUSINESSES FROM CAPITALIZING ON BIG DATA

1. Obtaining executive sponsorships for investments in big data and its related activities (such as training, etc.).
2. Getting the business units to share information across organizational silos.
3. Finding the right skills (business analysts and data scientists) that can manage large amounts of structured, semi-structured, and unstructured data and create insights from it.
4. Determining the approach to scale rapidly and elastically. In other words, the need to address the storage and processing of large volume, velocity, and variety of big data.
5. Deciding whether to use structured or unstructured, internal or external data to make business decisions.
6. Choosing the optimal way to report findings and analysis of big data (visual presentation and analytics) for the presentations to make the most sense.
7. Determining what to do with the insights created from big data.

Top challenges facing Big data

1. Scale: Horizontal and vertical scalability
2. Security: NOSQL dB
3. Schema: Dynamic schemas
4. Continuous availability: 24*7
5. Consistency: Opt for consistency
6. Partition tolerant: tolerating both hardware and software failures
7. Data quality: Data accuracy , completeness

Why big data analytics is important?

- Reactive- BI: Better Decision making
- Reactive-Big data analytics: For huge data
- Proactive- Analytics: Prediction, text mining and statistical analysis
- Proactive-Big data analytics: High performance analytics

Kind of technologies looking toward to meet big data challenges

- cheap and abundant storage
- Faster processors
- Affordable open sources, distributed platforms
- Parallel processing, clustering, Large grid environments
- Cloud computing and other flexible resource allocation arrangements.

DATA SCIENCE

Introduction to data science

- Data science is the science of extracting knowledge from data.

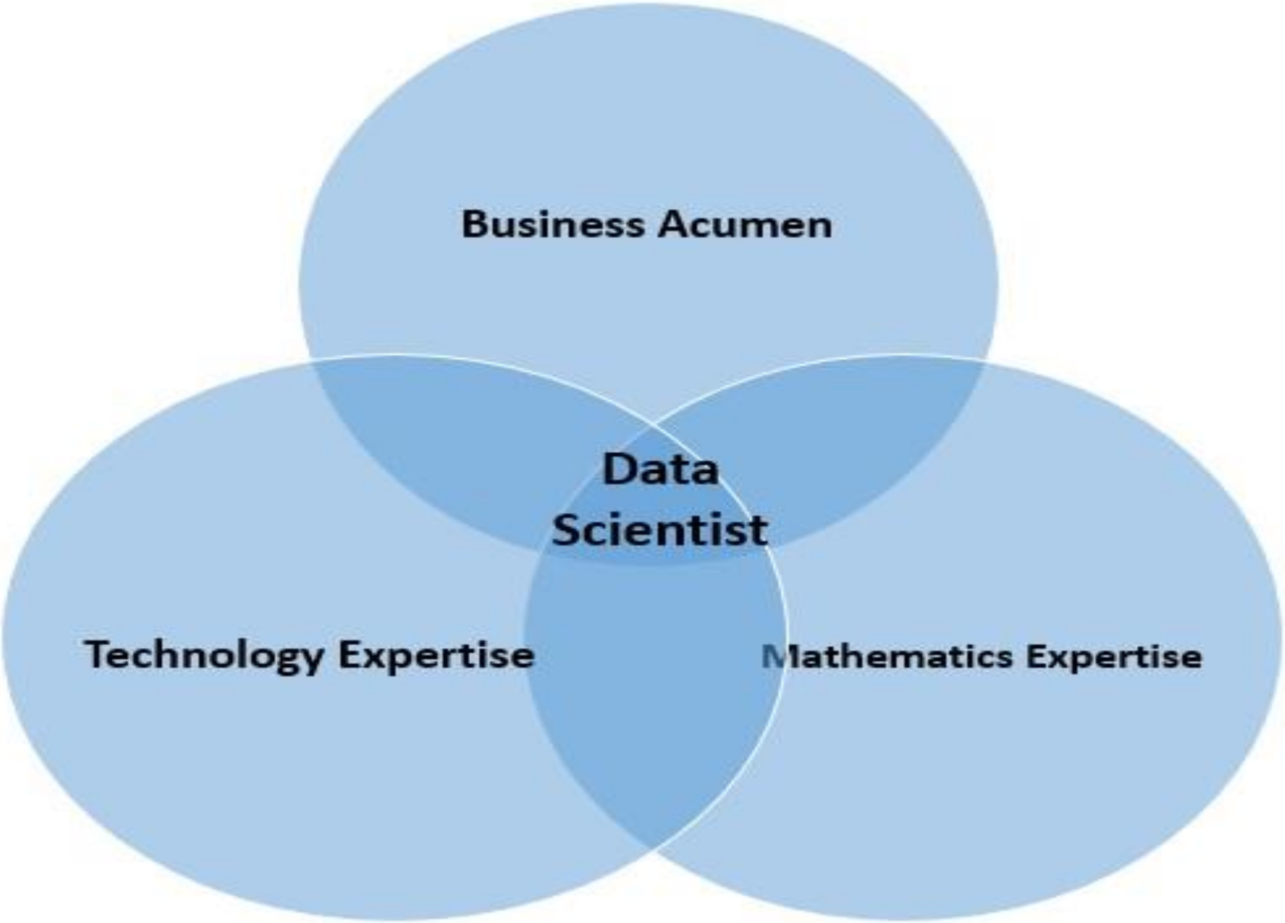
- It is a science of drawing out hidden patterns amongst data using statistical and mathematical techniques.

- It includes fields like maths, statistics, information technology, including machine learning, data engineering, probability models and PR etc.

- Data science is a multi disciplinary field.

- Examples:
  - Weather prediction
  - Financial frauds
  - Social media analytics

Data Scientist



Business Acumen skills

1. Understanding of domain
2. Business strategy
3. Problem solving communication
4. Presentation
5. Inquisitiveness

Technology Expertise

1. Good dB knowledge
2. Good NoSQL DB Knowledge
3. Programming languages
4. Open source tools
5. Data warehousing
6. Data mining
7. Visualization

Mathematical Expertise

1. Mathematics
2. Statistics
3. Artificial intelligence
4. Algorithms
5. Machine learning
6. Pattern recognition
7. NLP

Data Science Process is

1. Collecting raw data

2. Processing data

3. Integrating data

4. Engaging in explorative data analysis using models and algorithms

5. Preparing presentations

6. Communicating the findings to stakeholders

7. Making faster and better decisions

Responsibilities of Data scientists

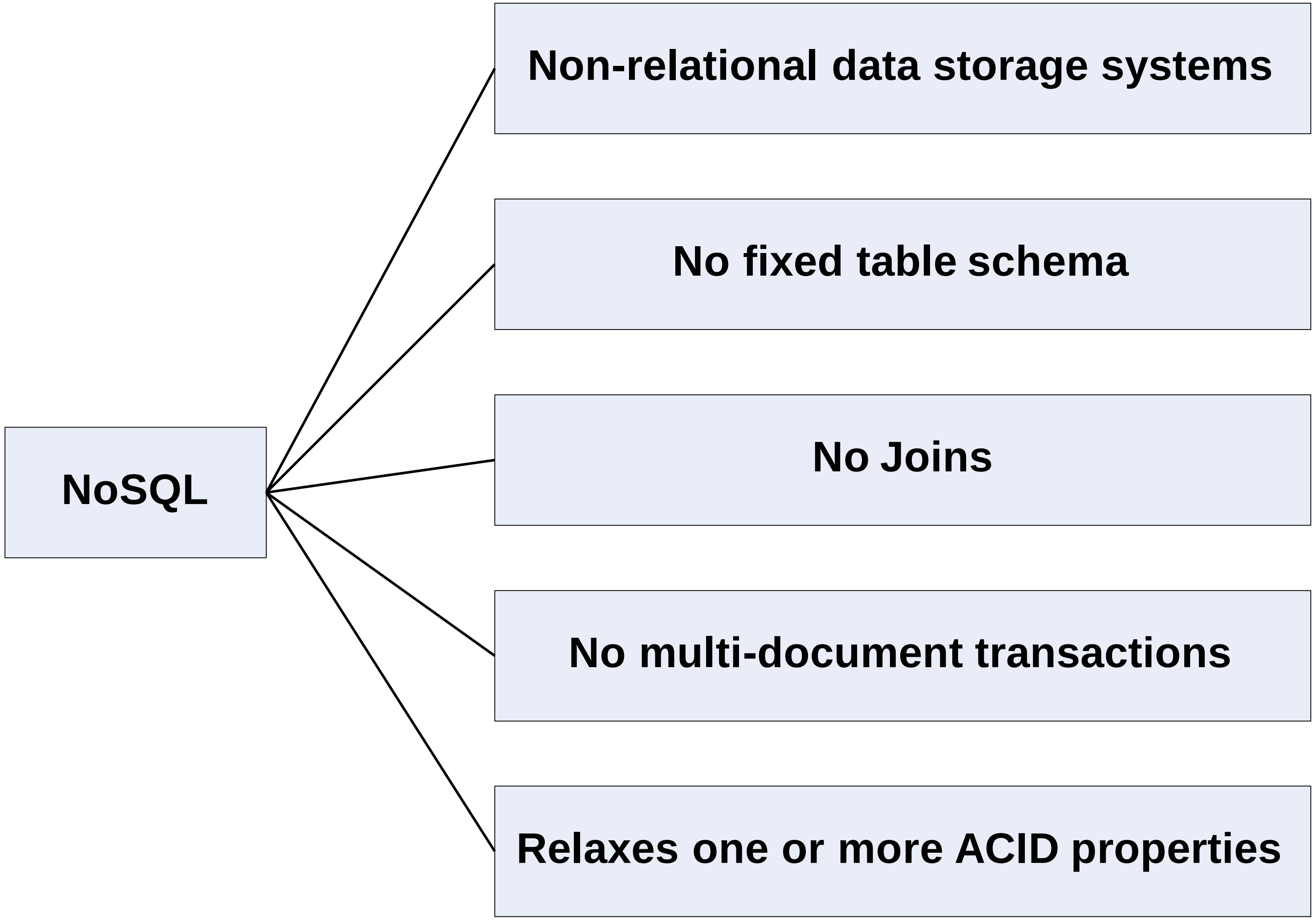
1.Data management: prepare and integrate large varied databases

2.Analytical Techniques: Models and analyses to comprehend, interpret relationships, Patterns and trends. Communicating /presents findings/results

3. Business analysts: Applies Business /Domain Knowledge to provide context

What is NoSQL ?

- NoSQL stands for Not only SQL.
- These are non- relational, opensource and distributed DB.
- They are huge popular because of their ability,
  - Scale out data
  - Handling variety of data.



NoSQL

Non-relational data storage systems

No fixed table schema

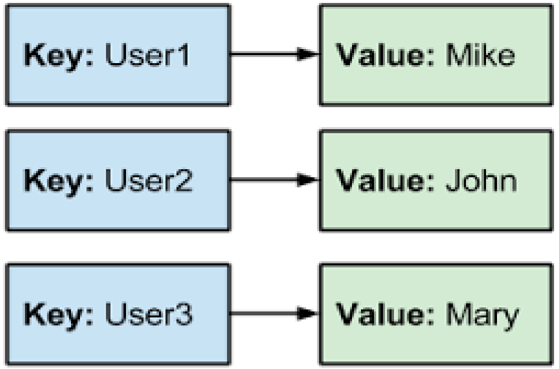
No Joins

No multi-document transactions

Relaxes one or more ACID properties

Types of NoSQL

- Key- value data store:
  - It maintains big hash tables for keys and values.



Key: User1 → **Value:** Mike

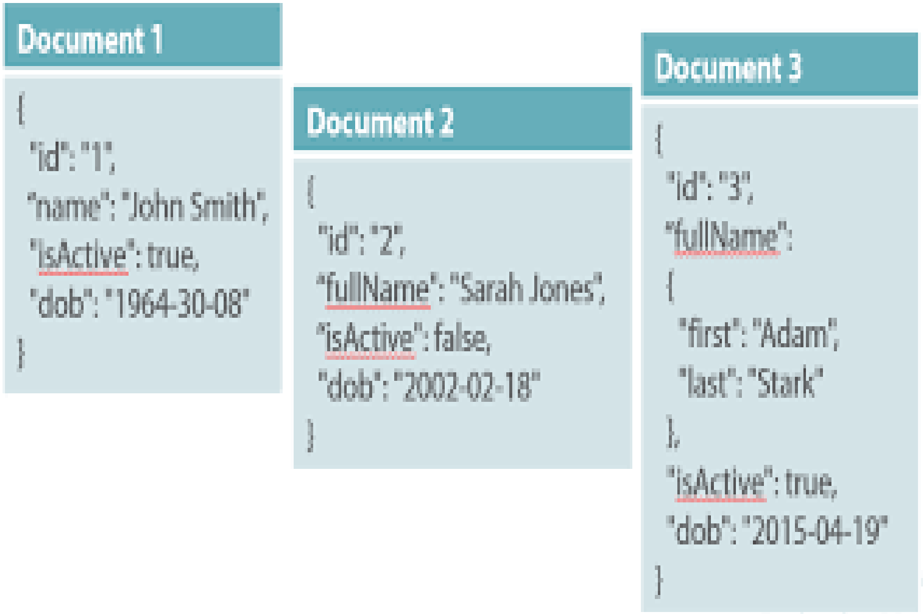
Key: User2 → **Value:** John

Key: User3 → **Value:** Mary

Example tools:
Riak
Membase
Redis

Types of NoSQL

- Document store:
    - It maintains data in a collection of constituted documents



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

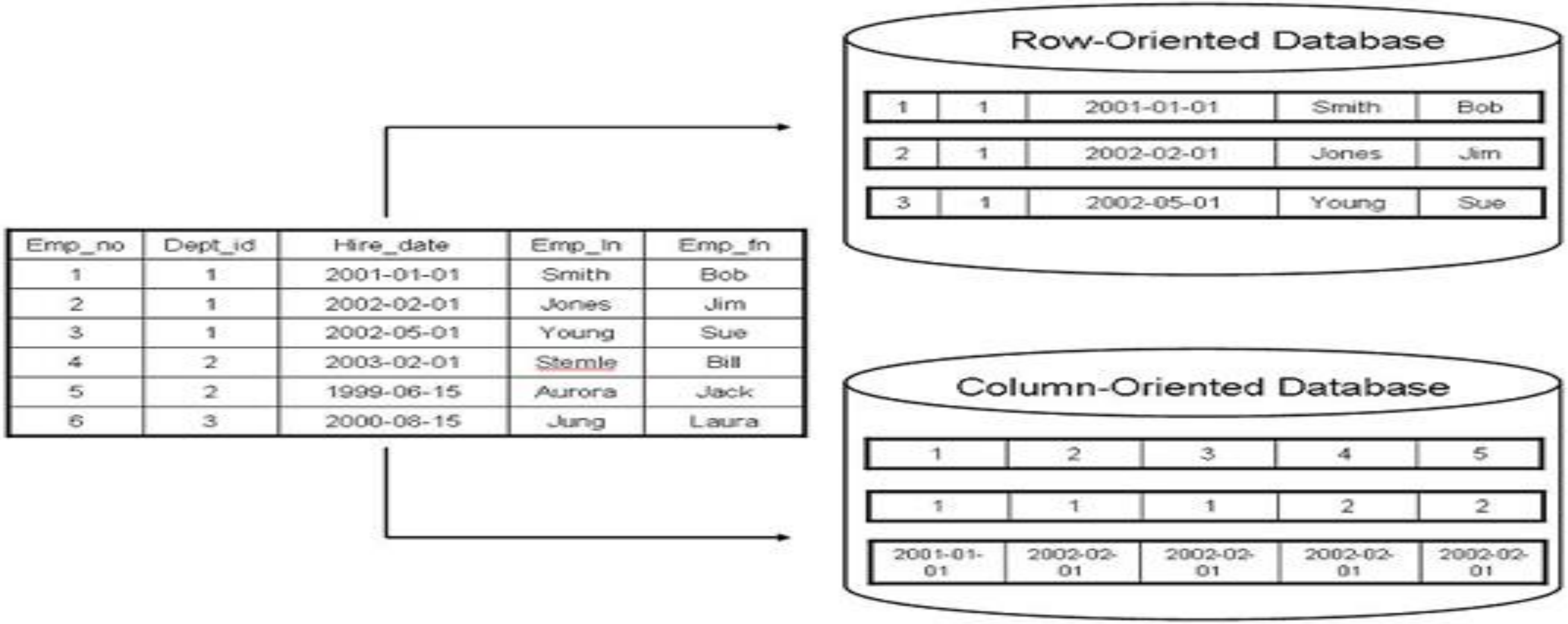
```
{
 "id": "3",
 "fullName":
 {
  "first": "Adam",
  "last": "Stark"
 },
 "isActive": true,
 "dob": "2015-04-19"
}
```

Example tools;
MongoDB
CouchDB

Types of NoSQL

Column oriented: Each storage block has from only one column

Example tools: Cassandra, HBase



Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura

Row-Oriented Database

1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue

Column-Oriented Database

1	2	3	4	5
1	1	1	2	2
2001-01-01	2002-02-01	2002-02-01	2002-02-01	2002-02-01

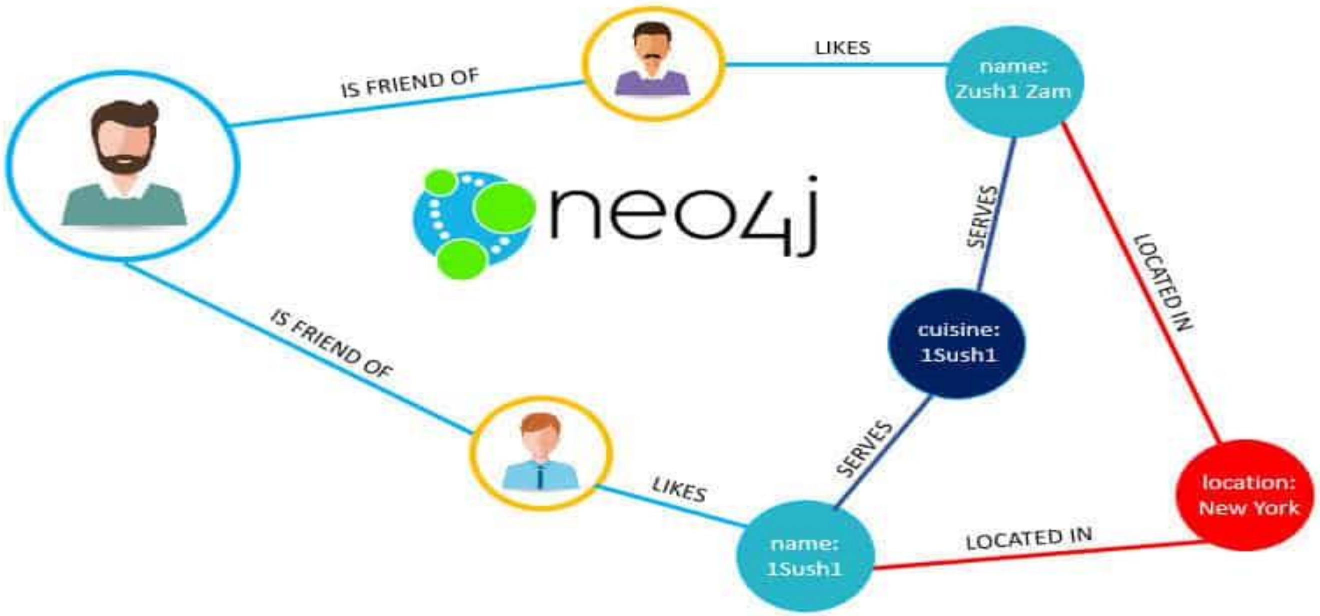
Types of NoSQL

- Graph Data base:
  - The data are stored in nodes.
  - Example tools:

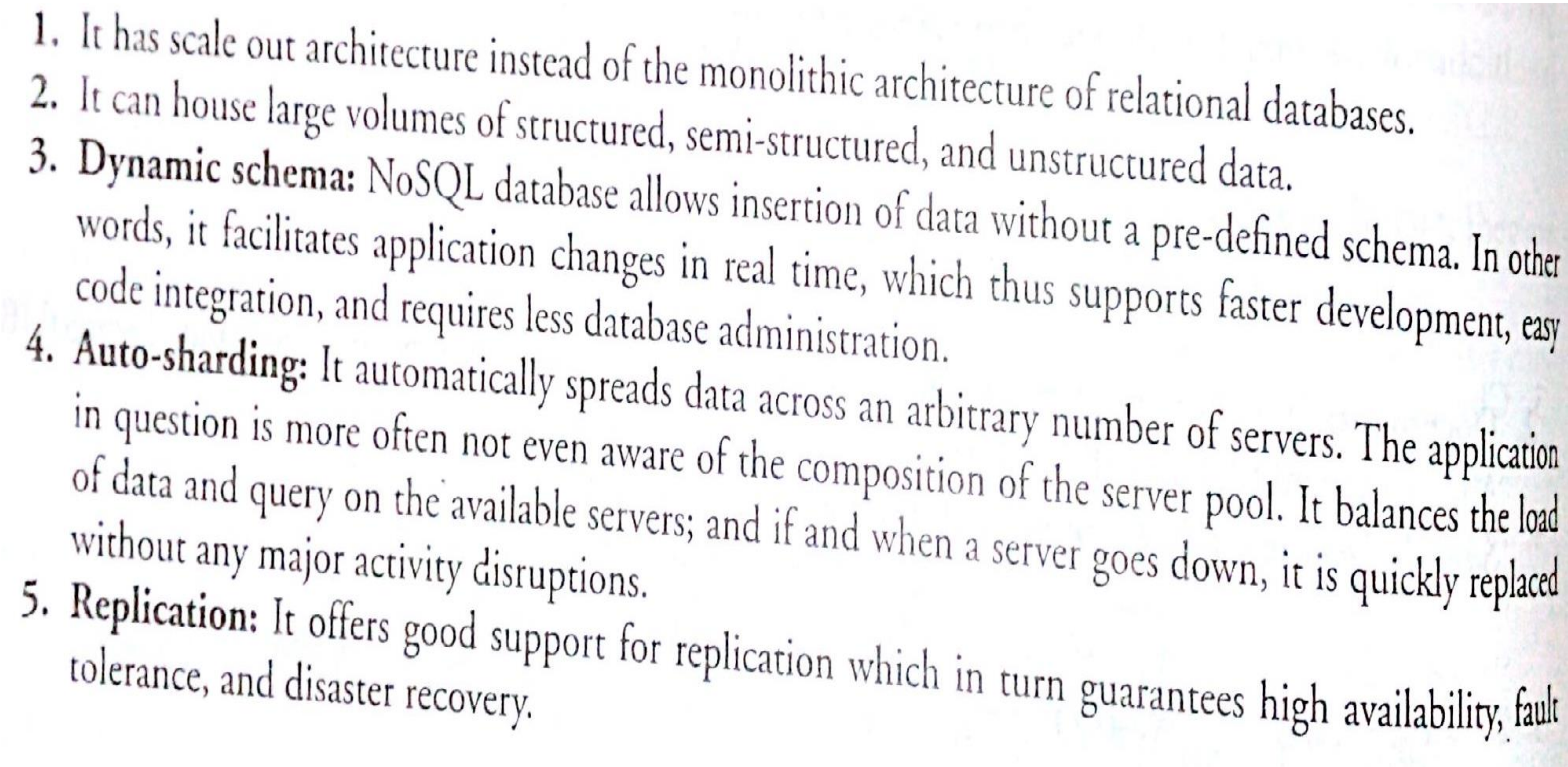
  Neo4

  Allegro Graph

  Infinite graph

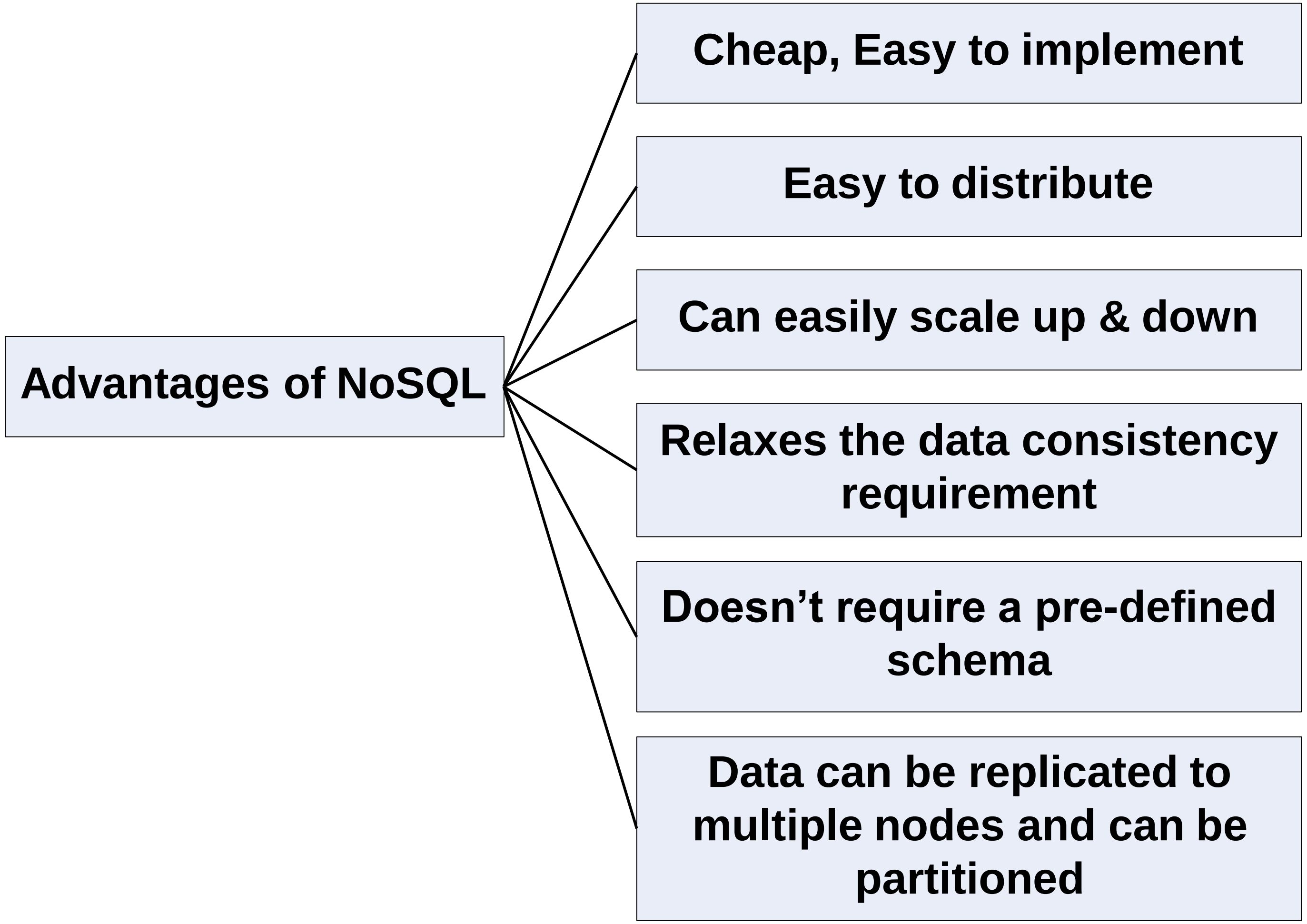


Why No SQL



1. It has scale out architecture instead of the monolithic architecture of relational databases.
2. It can house large volumes of structured, semi-structured, and unstructured data.
3. **Dynamic schema:** NoSQL database allows insertion of data without a pre-defined schema. In other words, it facilitates application changes in real time, which thus supports faster development, easy code integration, and requires less database administration.
4. **Auto-sharding:** It automatically spreads data across an arbitrary number of servers. The application in question is more often not even aware of the composition of the server pool. It balances the load of data and query on the available servers; and if and when a server goes down, it is quickly replaced without any major activity disruptions.
5. **Replication:** It offers good support for replication which in turn guarantees high availability, fault tolerance, and disaster recovery.

Advantages of NoSQL



Advantages of NoSQL

Cheap, Easy to implement

Easy to distribute

Can easily scale up & down

Relaxes the data consistency requirement

Doesn’t require a pre-defined schema

Data can be replicated to multiple nodes and can be partitioned

What we miss with NoSQL

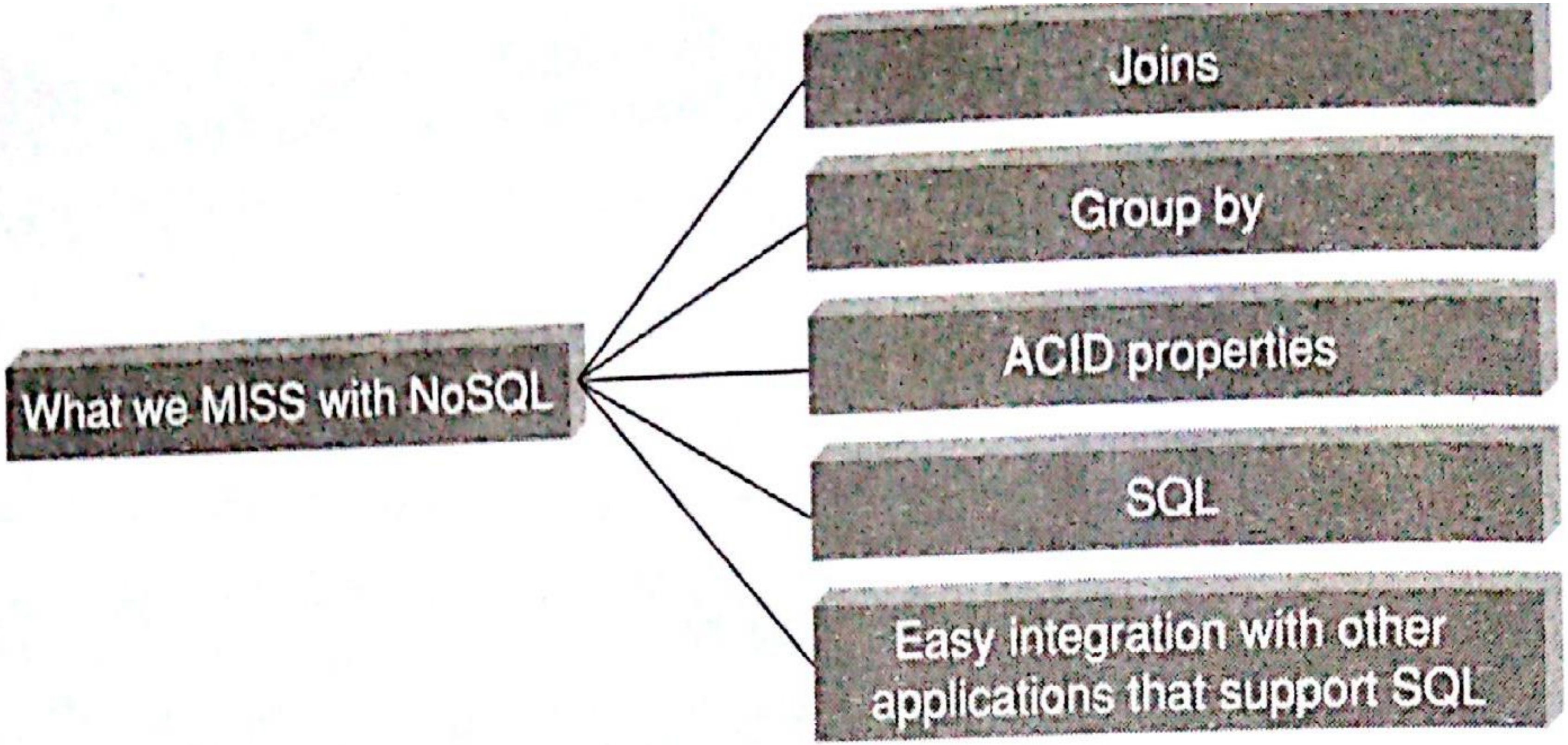
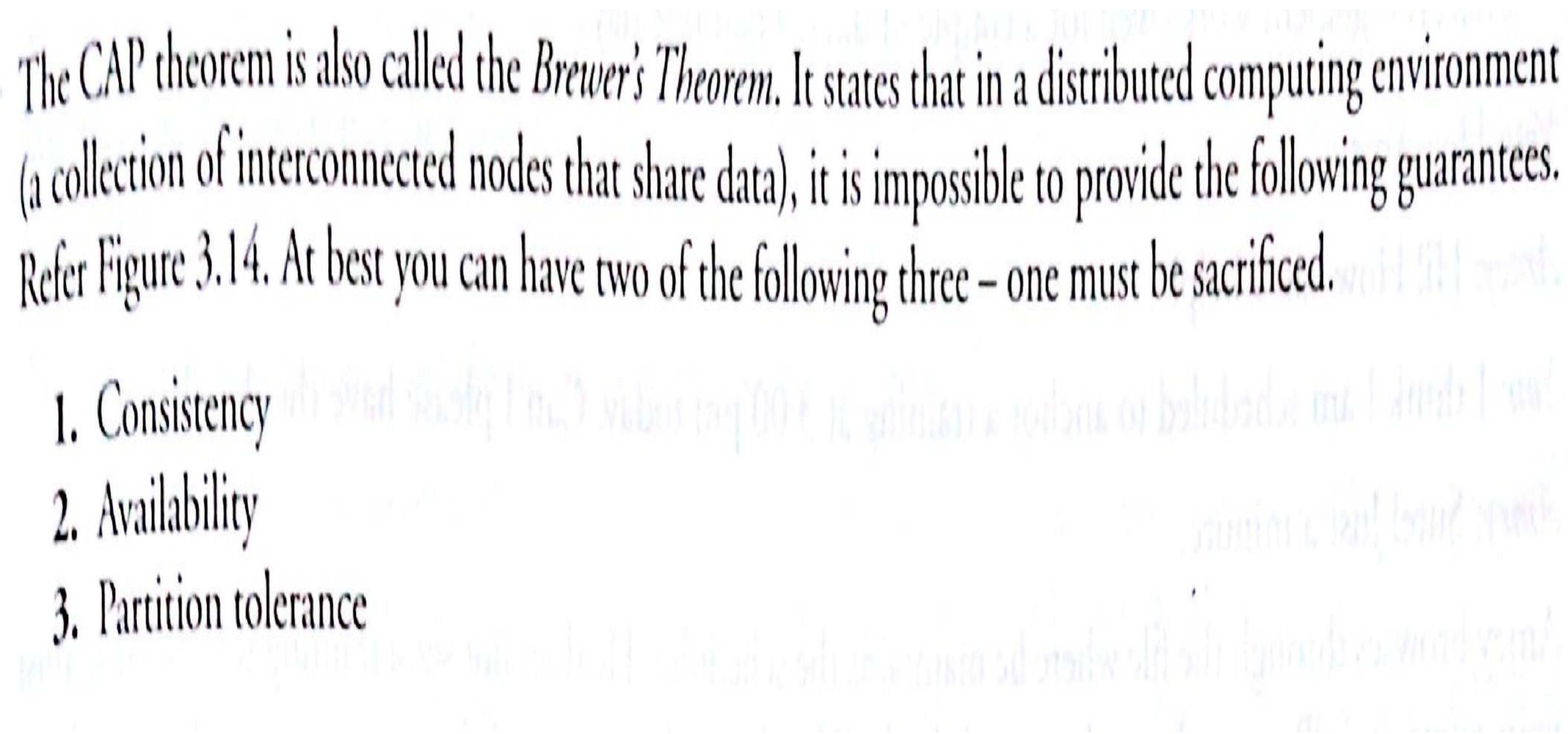


Figure 4.5 What we miss with NoSQL?

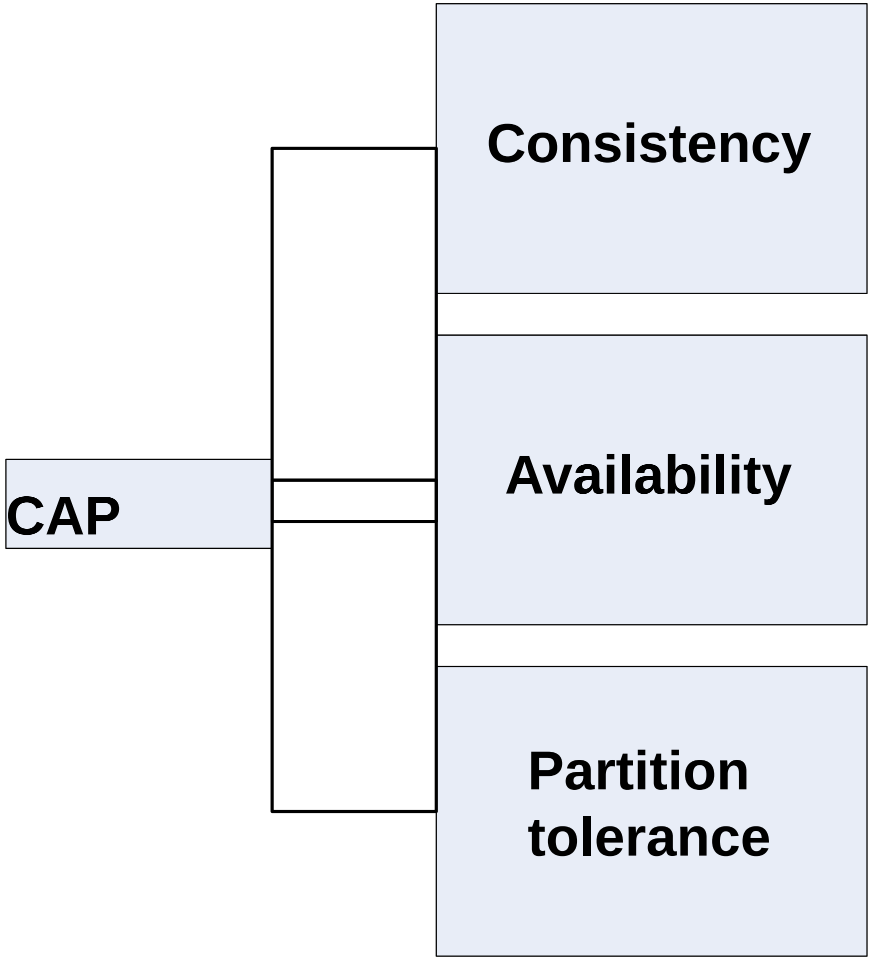
CAP Theorem



The CAP theorem is also called the *Brewer's Theorem*. It states that in a distributed computing environment (a collection of interconnected nodes that share data), it is impossible to provide the following guarantees. Refer Figure 3.14. At best you can have two of the following three – one must be sacrificed.

1. Consistency
2. Availability
3. Partition tolerance

Brewer’s CAP



CAP

Consistency

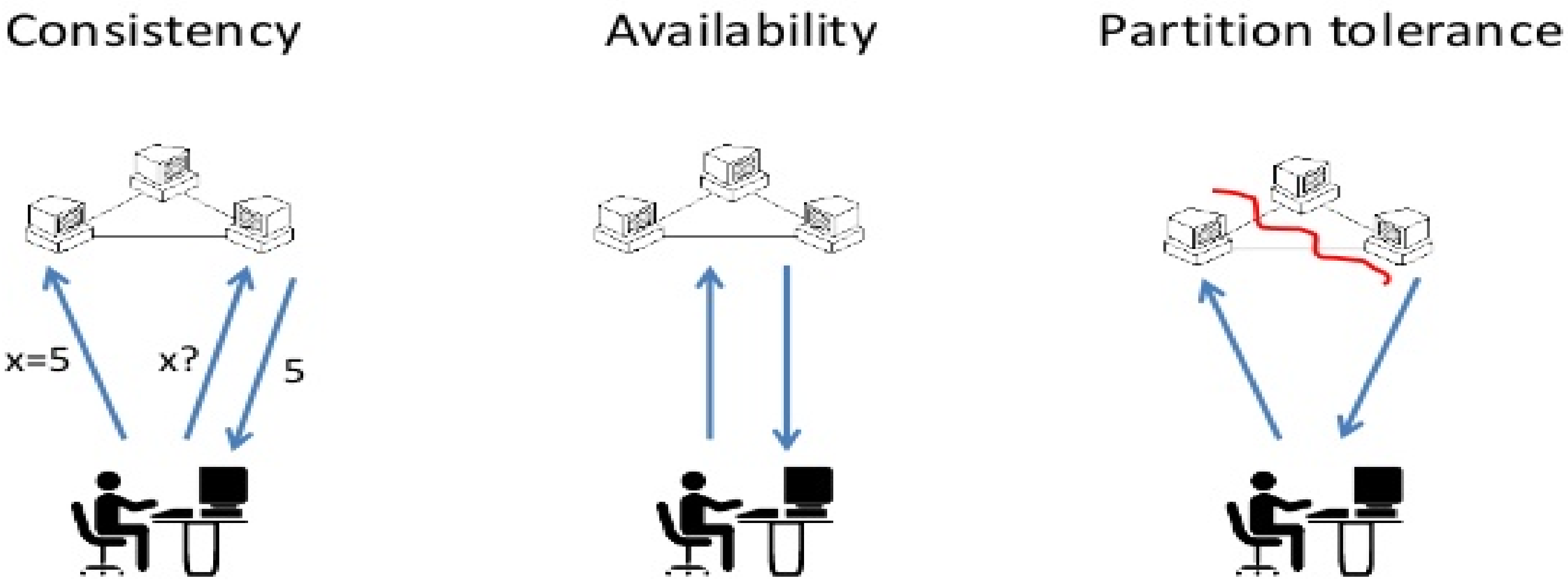
Availability

Partition tolerance

CAP Theorem

- Consistency:
  - All nodes should see the same data at the same time. Every Read fetches last write.
- Availability:
  - Node failures do not prevent survivors from continuing to operate. Read and write always succeed.
- Partition-tolerance:
  - The system continues to operate despite network partitions
- A distributed system can satisfy any two of these guarantees at the same time **but not all three.**

CAP Theorem



Consistency

Availability

Partition tolerance

Why this is important?

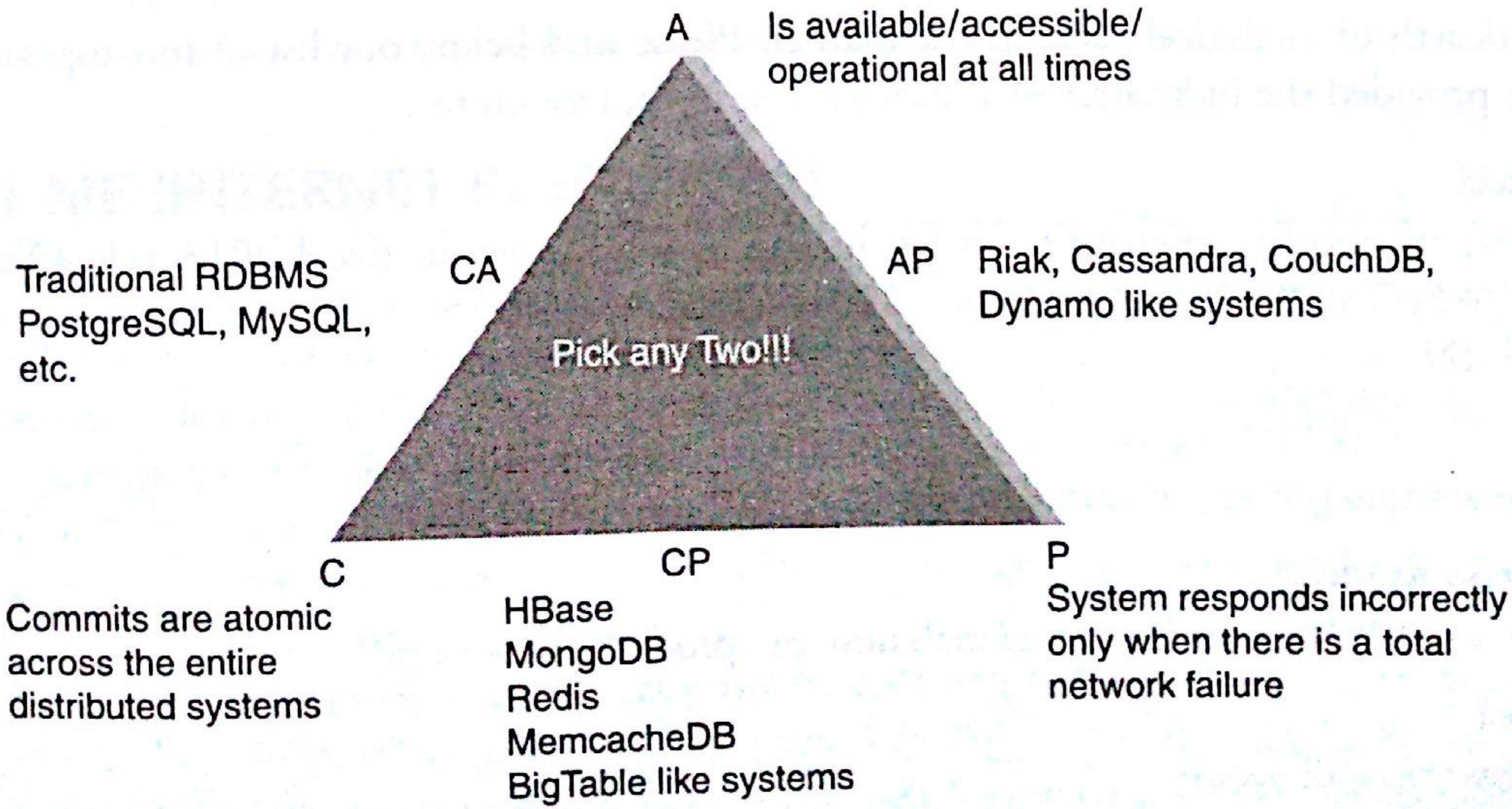
- The future of databases is **distributed** (Big Data Trend, etc.)

- CAP theorem describes the **trade-offs** involved in distributed systems

- A proper understanding of CAP theorem is essential to **making decisions** about the future of distributed database **design**

- Misunderstanding can lead to **erroneous or inappropriate** design choices

Example of data bases that follows CAP theorem



Is available/accessible/ operational at all times

Traditional RDBMS
PostgreSQL, MySQL,
etc.

Riak, Cassandra, CouchDB,
Dynamo like systems

Commits are atomic across the entire distributed systems

HBase
MongoDB
Redis
MemcacheDB
BigTable like systems

System responds incorrectly only when there is a total network failure

Figure 3.15 Databases and CAP.

When to consider consistency over availability and Vice -versa

1.Choose **availability over consistency** when your business requirements allow some flexibility around when data in the system synchronizes.

2.Choose **consistency over availability** when your business requirements demand atomic reads and writes.

Types of Consistency

Strong Consistency

After the update completes, **any subsequent access** will return the **same** updated value.

Weak Consistency

It is **not guaranteed** that subsequent accesses will return the updated value.

Eventual Consistency

Specific form of weak consistency

It is guaranteed that if **no new updates** are made to object, **eventually** all accesses will return the last updated value (e.g., *propagate updates to replicas in a lazy fashion*)

Eventual Consistency Variations

<u>Causal consistency</u>

Processes that have causal relationship will see consistent data

<u>Read-your-write consistency</u>

A process always accesses the data item after it’s update operation and never sees an older value

<u>Session consistency</u>

As long as session exists, system guarantees read-your-write consistency

Guarantees do not overlap sessions

Eventual Consistency Variations

<u>Monotonic read consistency</u>

If a process has seen a particular value of data item, any subsequent processes will never return any previous values

<u>Monotonic write consistency</u>

The system guarantees to serialize the writes by the *same* process

In practice
A number of these properties can be combined
Monotonic reads and read-your-writes are most desirable

Eventual Consistency- A Facebook Example

Bob finds an interesting story and shares with Alice by posting on her Facebook wall
Bob asks Alice to check it out
Alice logs in her account, checks her Facebook wall but finds:

- **Nothing is there!**

Wall
facebook
?

Eventual Consistency- A Facebook Example

Bob tells Alice to wait a bit and check out later
Alice waits for a minute or so and checks back:

- **She finds the story Bob shared with her!**

→

Wall

facebook

→

Eventual Consistency- A Facebook Example

Reason: it is possible because Facebook uses an **eventual consistent model**

Why Facebook chooses eventual consistent model over the strong consistent one?

- Facebook has more than 1 billion active users
- It is non-trivial to efficiently and reliably store the huge amount of data generated at any given time
- Eventual consistent model offers the option to **reduce the load and improve availability**

BASE

- Basically Available Soft state Eventual consistency

- Where it is used? – Distributed Computing

- Why?- To achieve high availability

- How it is achieved?-No new updates made to the data for a stipulated period of time , eventually all accesses to this data will return the updated value.

- What is replica convergence?- system that has achieved eventual consistency is said to have converged.

- Conflict resolution: solved by

1. Read repair
2. Write repair
3. Asynchronous repair

ACID vs BASE

ACID	BASE
•Strong consistency for transactions highest priority	•Availability and scaling highest priorities
•Availability less important	•Weak consistency
•Pessimistic	•Optimistic
•Rigorous analysis	•Best effort
•Complex mechanisms	•Simple and fast

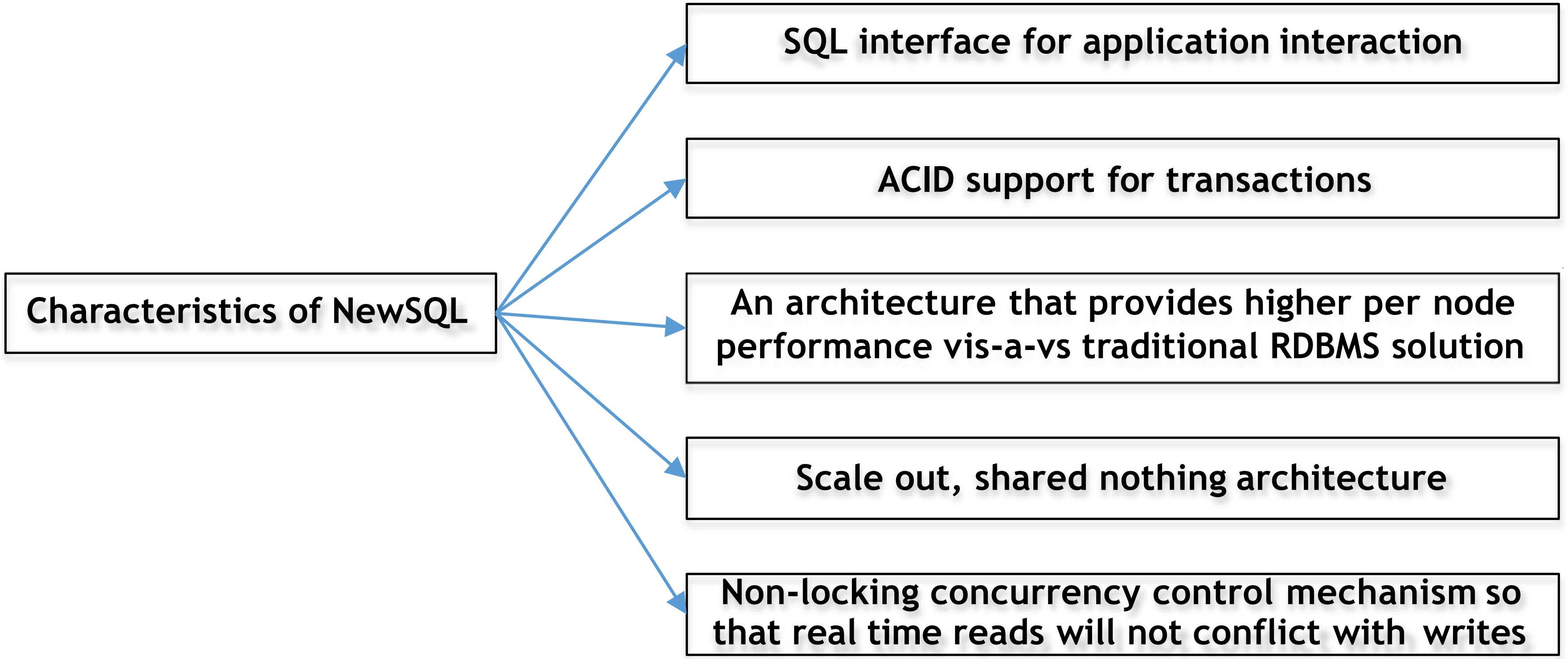
NoSQL Vendors

Company	Product	Most widely used by
Amazon	DynamoDB	LinkedIn, Mozilla
Facebook	Cassandra	Netflix, Twitter, eBay
Google	BigTable	Adobe Photoshop

SQL Vs. NoSQL

SQL	NoSQL
Relational database	Non-relational, distributed database
Relational model	Model-less approach
Pre-defined schema	Dynamic schema for unstructured data
Table based databases	Document-based or graph-based or wide column store or key-value pairs databases
Vertically scalable (by increasing system resources)	Horizontally scalable (by creating a cluster of commodity machines)
Uses SQL	Uses UnQL (Unstructured Query Language)
Not preferred for large datasets	Largely preferred for large datasets
Not a best fit for hierarchical data	Best fit for hierarchical storage as it follows the key-value pair of storing data similar to JSON (Java Script Object Notation)
Emphasis on ACID properties	Follows Brewer's CAP theorem
Excellent support from vendors	Relies heavily on community support
Supports complex querying and data keeping needs	Does not have good support for complex querying
Can be configured for strong consistency	Few support strong consistency (e.g., MongoDB), few others can be configured for eventual consistency (e.g., Cassandra)
Examples: Oracle, DB2, MySQL, MS SQL, PostgreSQL, etc.	MongoDB, HBase, Cassandra, Redis, Neo4j, CouchDB, Couchbase, Riak, etc.

NewSQL



Characteristics of NewSQL

SQL interface for application interaction

ACID support for transactions

An architecture that provides higher per node performance vis-a-vs traditional RDBMS solution

Scale out, shared nothing architecture

Non-locking concurrency control mechanism so that real time reads will not conflict with writes

SQL Vs. NoSQL Vs. NewSQL

	SQL	NoSQL	NewSQL
Adherence to ACID properties	Yes	No	Yes
OLTP/OLAP	Yes	No	Yes
Schema rigidity Adherence to data model	Yes Adherence to relational model	No	Maybe
Data Format Flexibility	No	Yes	Maybe
Scalability	Scale up Vertical Scaling	Scale out Horizontal Scaling	Scale out
Distributed Computing	Yes	Yes	Yes
Community Support	Huge	Growing	Slowly growing

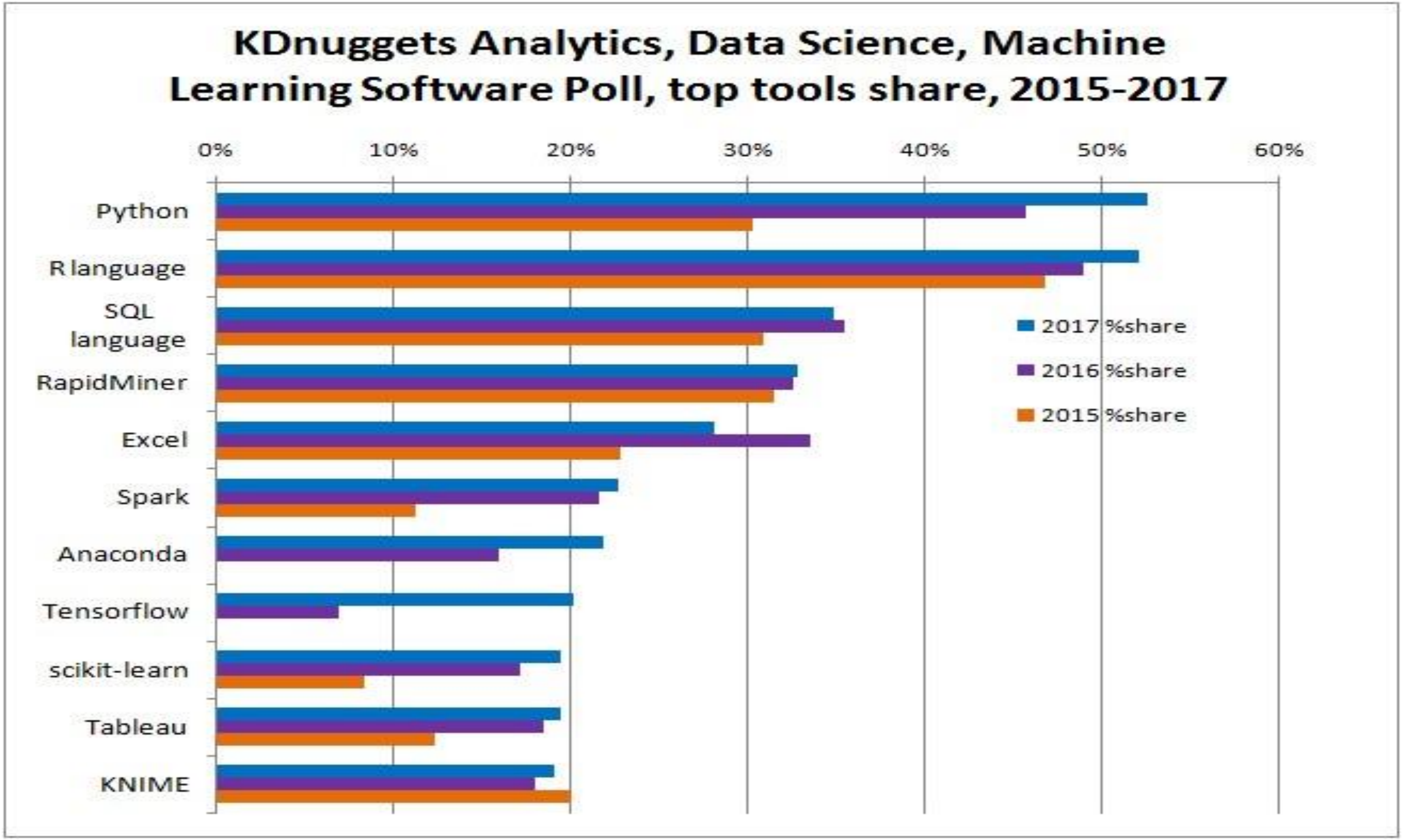
Few Top Analytical Tools

Few Top Analytical Tools

- MS Excel
https://support.office.microsoft.com/en-in/article/Whats-new-in-Excel-2013- 1cbc42cd-bfaf-43d7-9031-5688ef1392fd?CorrelationId=1a2171cc-191f-47de-8a55- 08a5f2e9c739&ui=en-US&rs=en-IN&ad=IN

- SAS
http://www.sas.com/en_us/home.html

- IBM SPSS Modeler
http://www-01.ibm.com/software/analytics/spss/products/modeler/



KDnuggets Analytics, Data Science, Machine Learning Software Poll, top tools share, 2015-2017