سُہیل عِمران

Dr. SOHAIL IMRAN

Intro

sohailimran2@yahoo.com

Ref: Web

intro

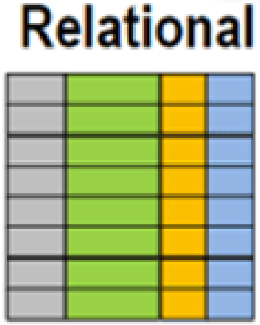
The concept of NOSQL databases gained popularity among Internet giants like Google, Yahoo, Facebook, and Amazon, which deal with huge volumes of data.
Carl Strozz introduced the NOSQL concept in 1998.

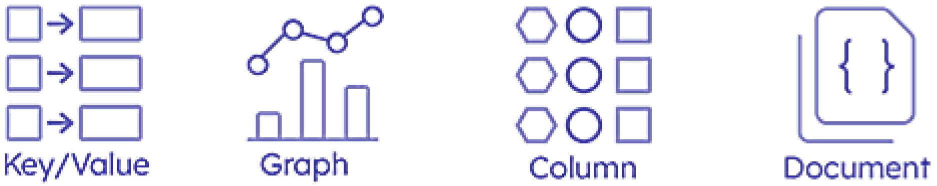
NOSQL stands for "**N**ot **O**nly **S**tructured **Q**uery **L**anguage“.

NOSQL should not be misleading:
the approach does not prohibit Structured Query Language (SQL)

It is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale.
The purpose of using a NOSQL database is for distributed data stores with humongous data storage needs. NOSQL is used for Big data and real-time web apps.

Traditional **RDBMS** uses **SQL** syntax to store and retrieve data for further insights. Instead, a NOSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.



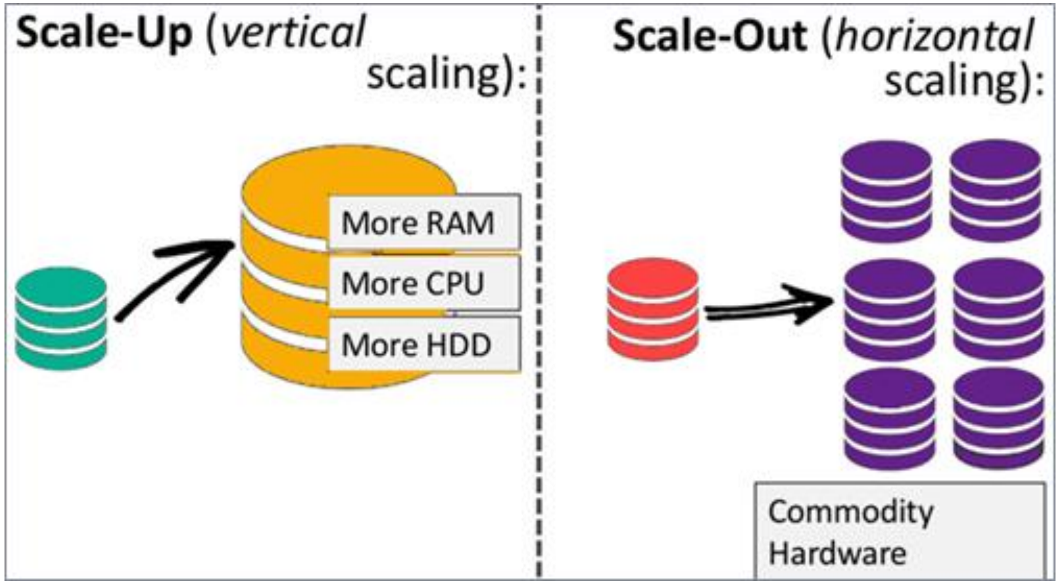


scale-up Vs scale-out

Problem with RDMS:
The system response time becomes slow when you use RDBMS for massive volumes of data.

Solution:
To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is **expensive**.
The alternative to this issue is to distribute the database load across multiple hosts whenever the load increases. This method is known as "**scaling out**."

A NOSQL database is non-relational, so it scales out better than relational databases, as they are designed with web applications in mind.



Upgrading equipment to improve performance

Adding new equipment and distributing workloads across it

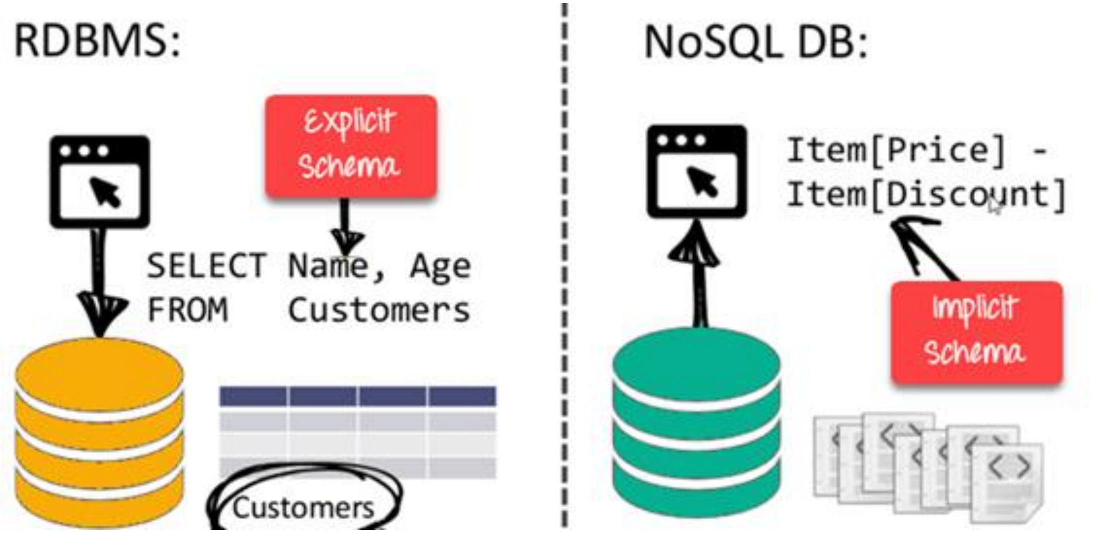
features

Non-relational

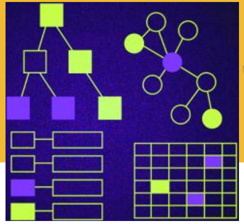
- NOSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners, referential integrity joins, ACID

Schema-free

- NOSQL databases are either schema-free or have relaxed schemas
- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain

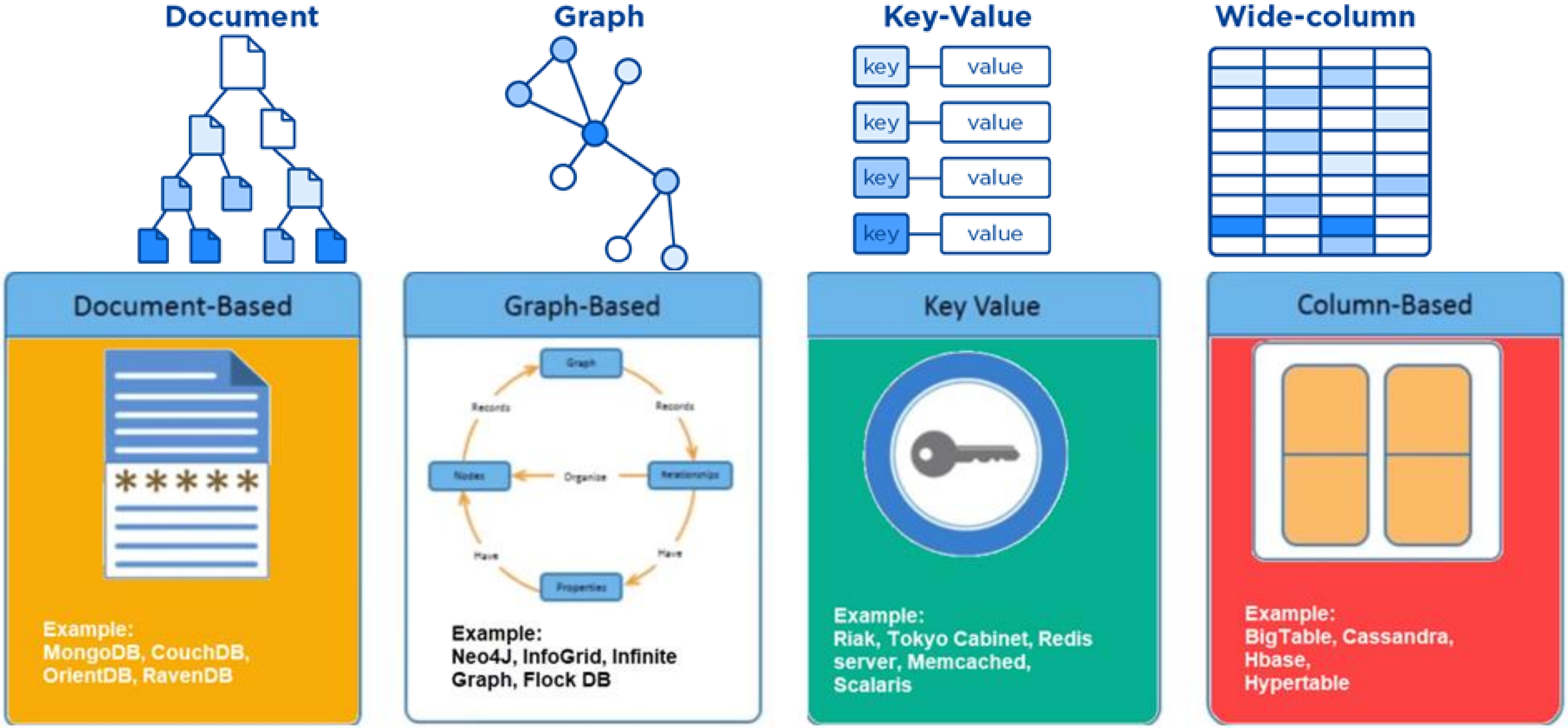


An **implicit schema** means that a document can be stored as it is without having to define a **schema** for it and without checking that it conforms to a **schema**.



types

There are mainly four categories of NOSQL databases.
Each of these categories has its unique attributes and limitations.
No specific database is better to solve all problems.
One should select a database based on one’s product needs.



Document-Based

Example:
MongoDB, CouchDB,
OrientDB, RavenDB

Graph-Based

Example:
Neo4J, InfoGrid, Infinite
Graph, Flock DB

Key Value

Example:
Riak, Tokyo Cabinet, Redis
server, Memcached,
Scalaris

Column-Based

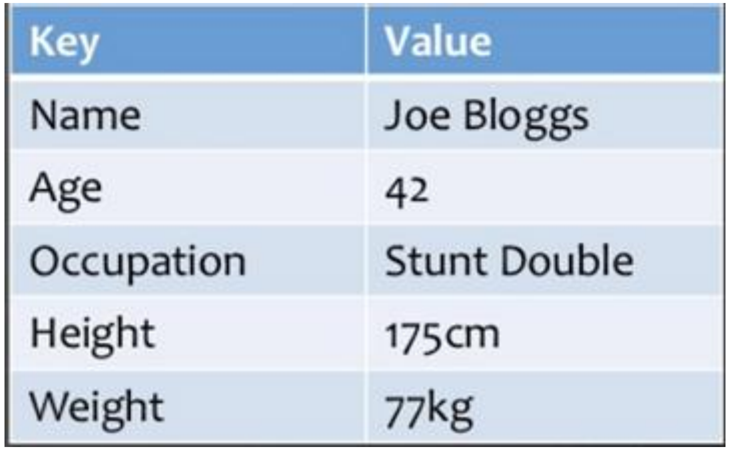
Example:
BigTable, Cassandra,
Hbase,
Hypertable

key value pair-based DB

Data is stored in key/value pairs. It is designed in such a way to handle lots of data and a heavy load.
Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.

It is one of the most basic types of NOSQL databases. This kind of NOSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.

Redis, Dynamo, Riak are some examples of key-value store Databases. They are all based on Amazon's Dynamo paper.



Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

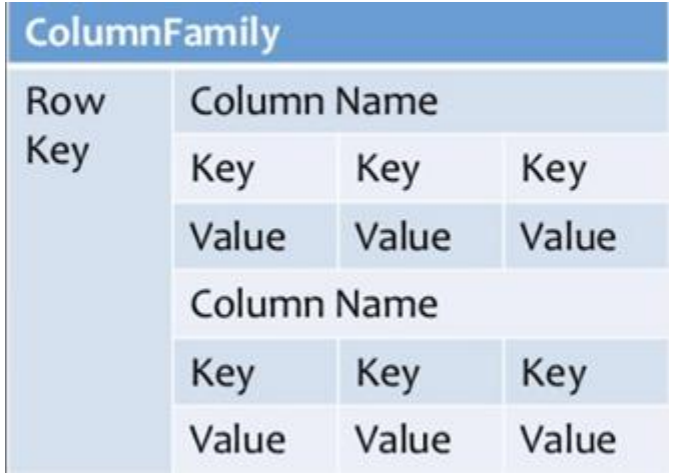
column-oriented database

Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately.
Values of single column databases are stored contiguously.

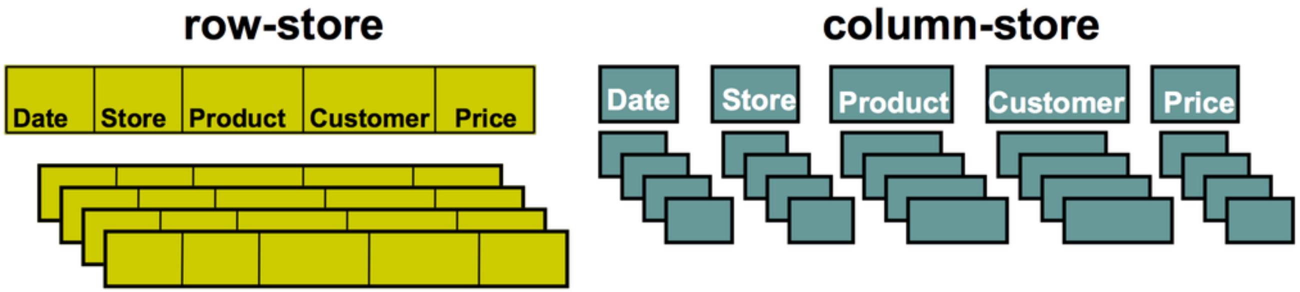
They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Column-based NOSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,

HBase, Cassandra, HBase, Hypertable are examples of column based database.



ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

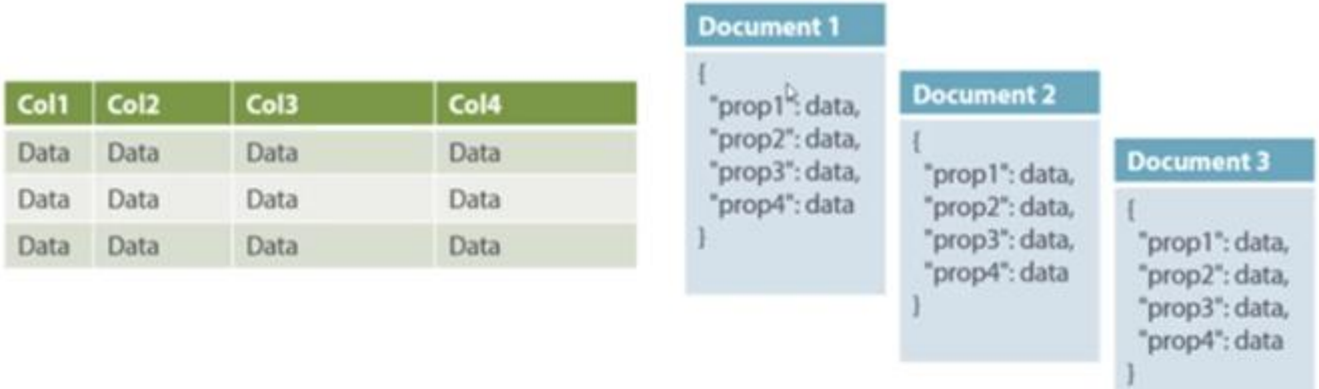


document-oriented DB store

Document-Oriented NOSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.

The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.

Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes, are popular Document originated DBMS systems.



Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

Document 2

```
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

Document 3

```
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

In this diagram, you can see we have rows and columns, and in the right, we have a document database which has a similar structure to JSON. Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have data store like JSON object. You do not require to define which make it flexible.

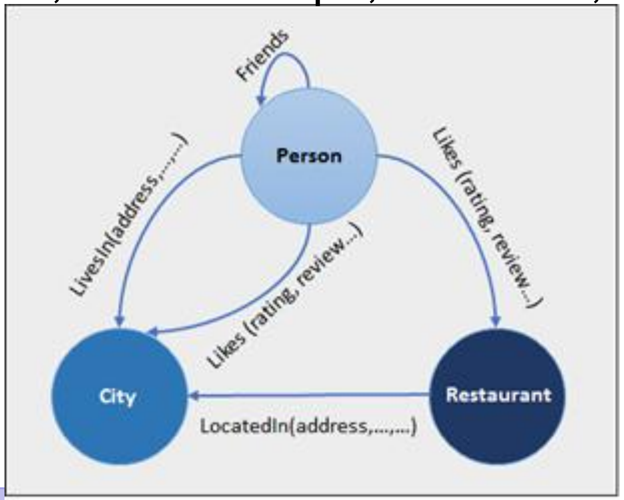
graph-based database

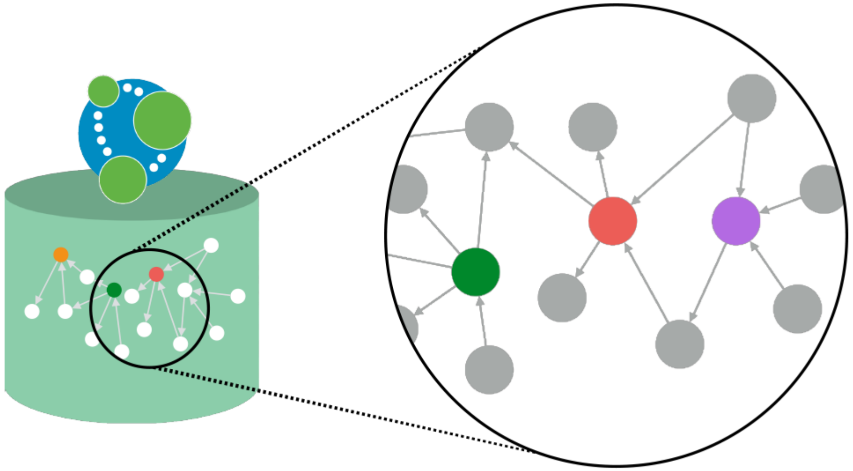
A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.

Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

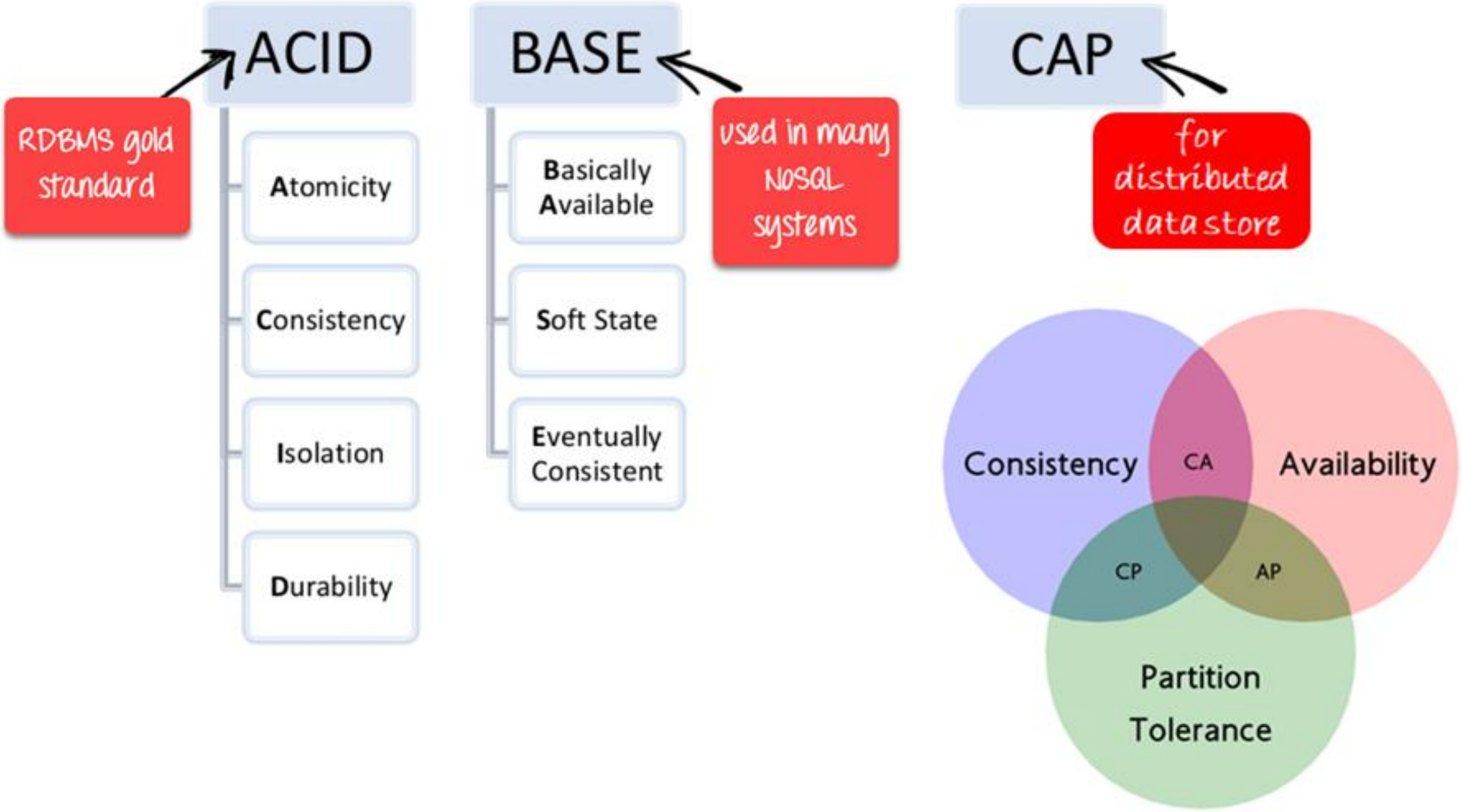
Graph base database mostly used for social networks, logistics, spatial data.

Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.


Friends
Person
LivesIn(address,.......)
Likes (rating, review...)
Likes (rating, review...)
City
Restaurant
LocatedIn(address,....,...)



A B C of data stores



ACID

RDBMS gold standard

- Atomicity
- Consistency
- Isolation
- Durability

BASE

used in many NoSQL systems

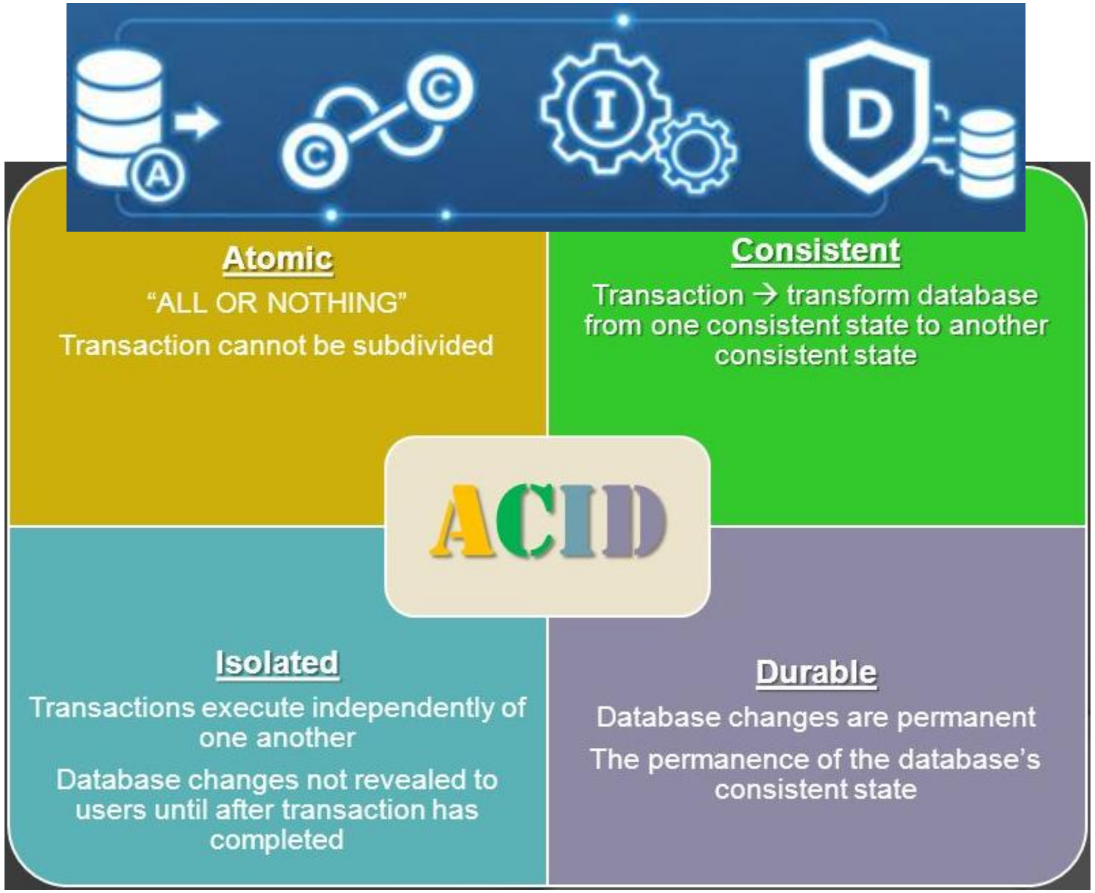
- Basically Available
- Soft State
- Eventually Consistent

CAP

for distributed data store

A C I D properties

A **transaction** is a single logical unit of work that accesses and possibly modifies the contents of a database. Transactions access data using read and write operations. In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.



Atomic

"ALL OR NOTHING"

Transaction cannot be subdivided

Consistent

Transaction → transform database from one consistent state to another consistent state

Isolated

Transactions execute independently of one another

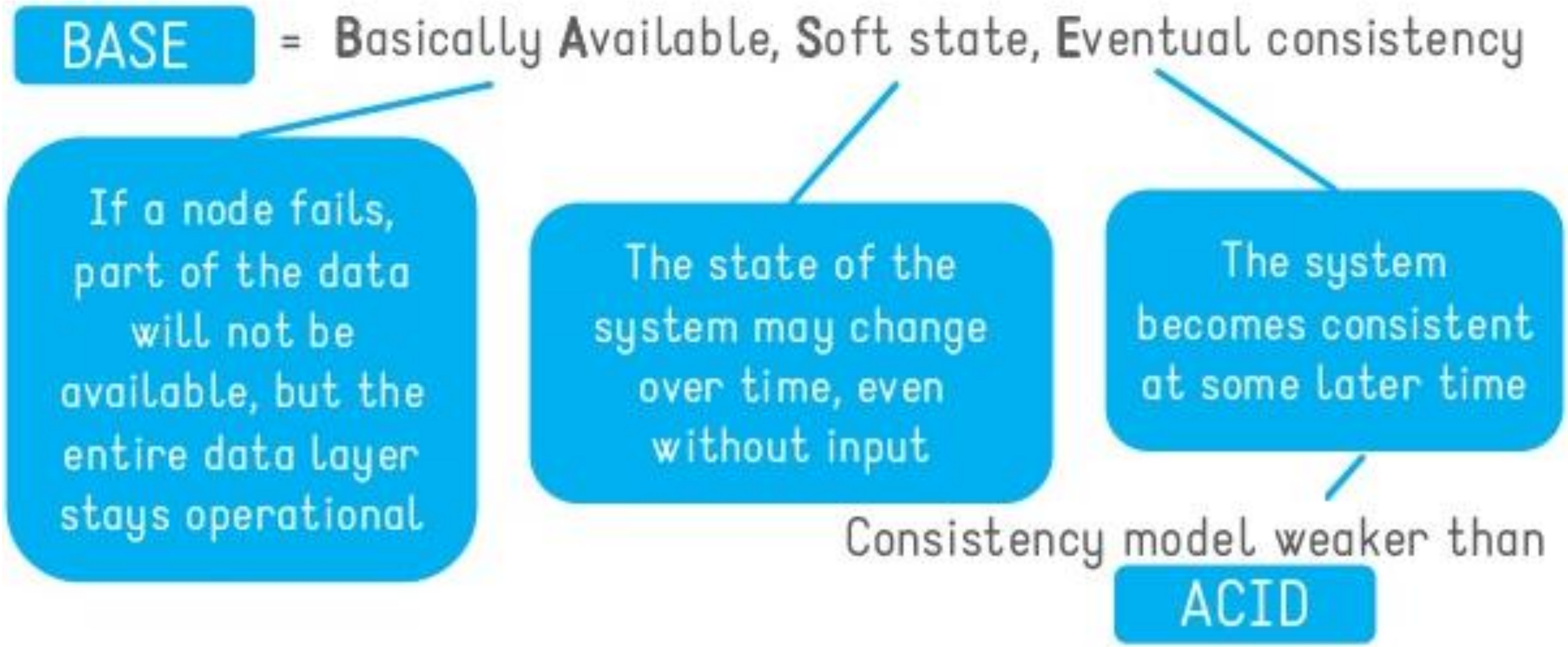
Database changes not revealed to users until after transaction has completed

Durable

Database changes are permanent

The permanence of the database's consistent state

BASE



BASE = Basically Available, Soft state, Eventual consistency

If a node fails, part of the data will not be available, but the entire data layer stays operational

The state of the system may change over time, even without input

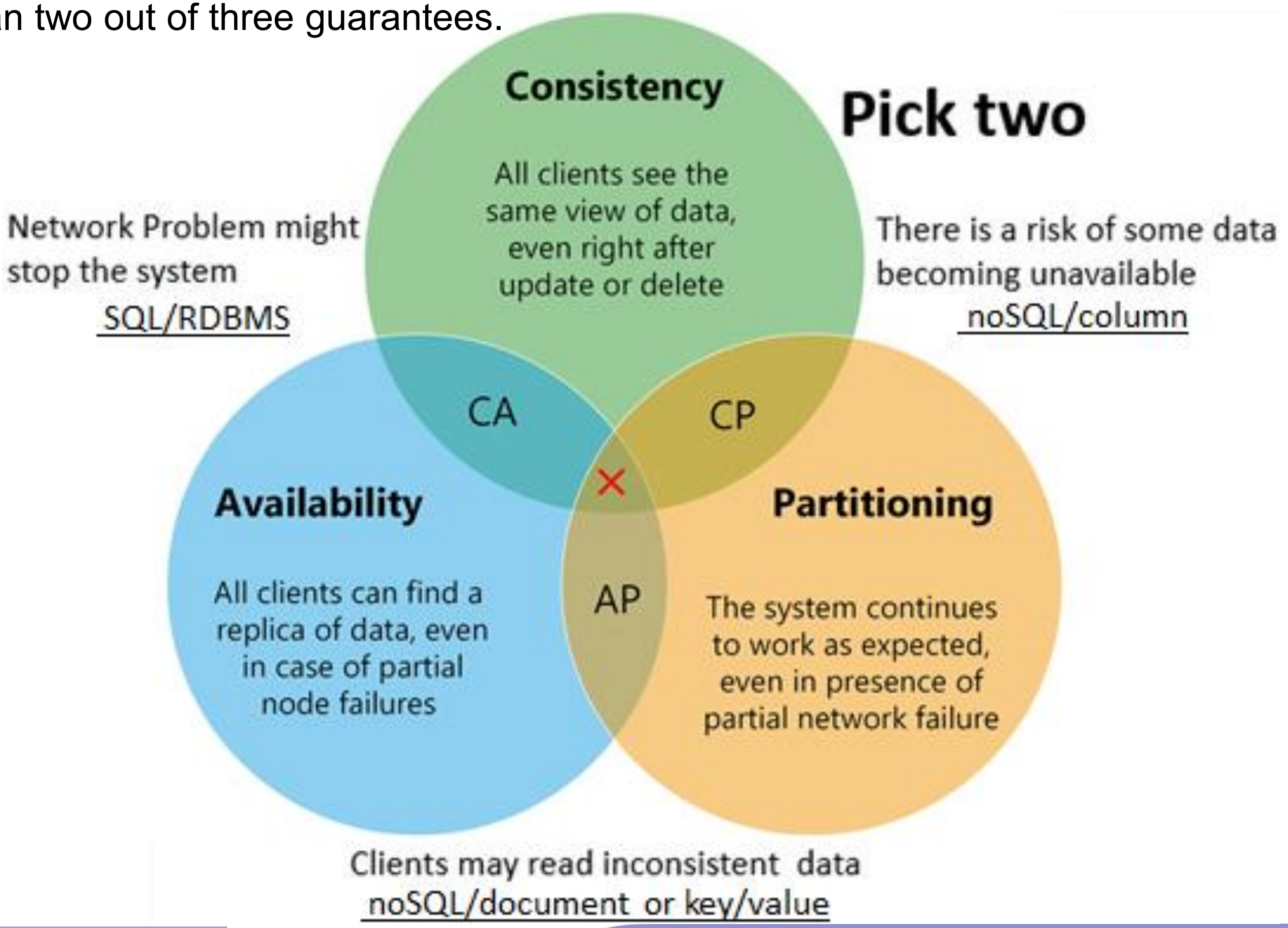
The system becomes consistent at some later time

Consistency model weaker than ACID

Thus, changes made to any data item on one machine has to be propagated to other replicas.
Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time.
These copies may be mutually, but in due course of time, they become consistent.
Hence, the name eventual consistency.

C A P

CAP theorem is also called brewer's theorem. It states that is impossible for a distributed data store to offer more than two out of three guarantees.



Pick two

Consistency

All clients see the same view of data, even right after update or delete

Availability

All clients can find a replica of data, even in case of partial node failures

Partitioning

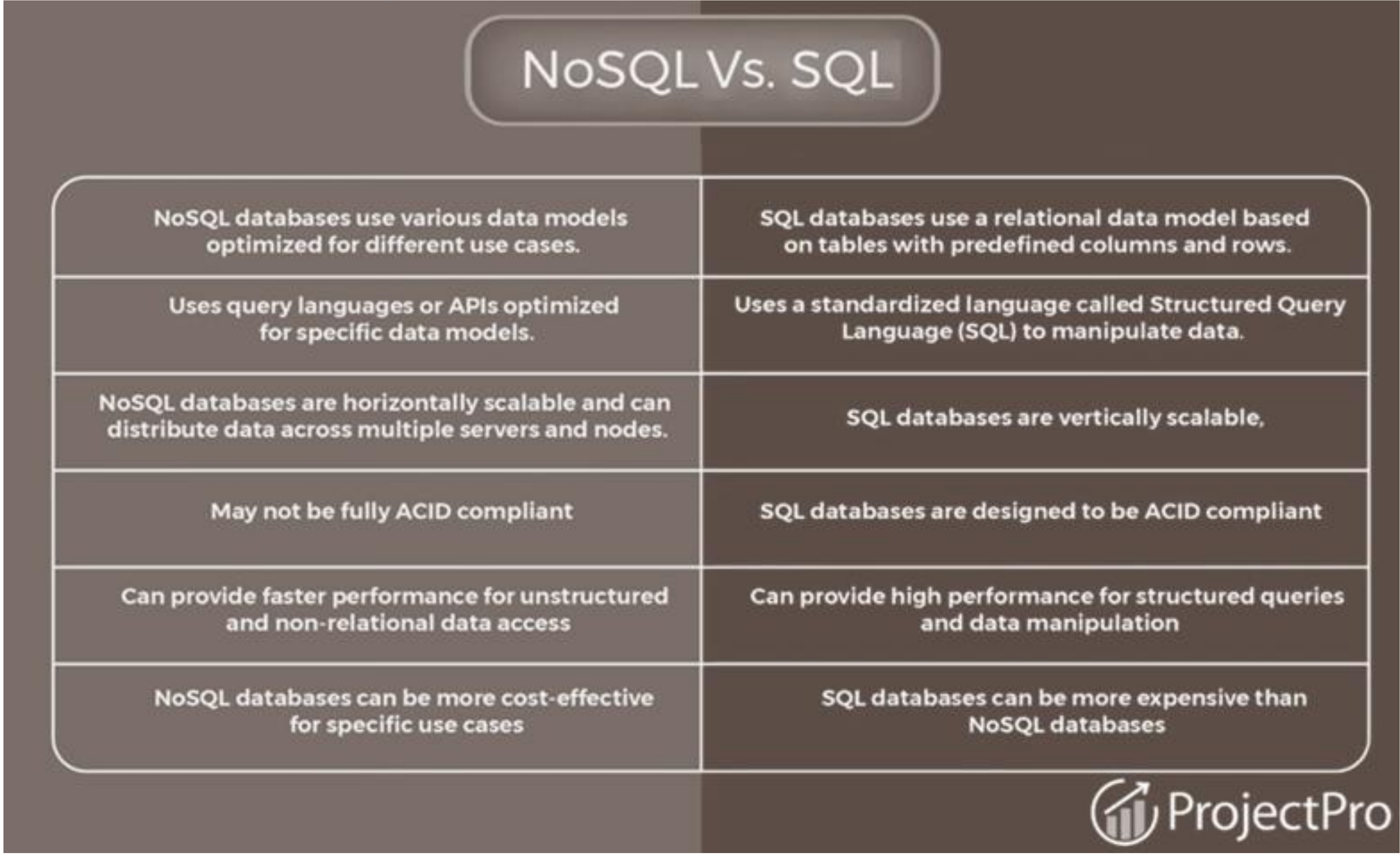
The system continues to work as expected, even in presence of partial network failure

Network Problem might stop the system
SQL/RDBMS

There is a risk of some data becoming unavailable
noSQL/column

Clients may read inconsistent data
noSQL/document or key/value

pros & cons

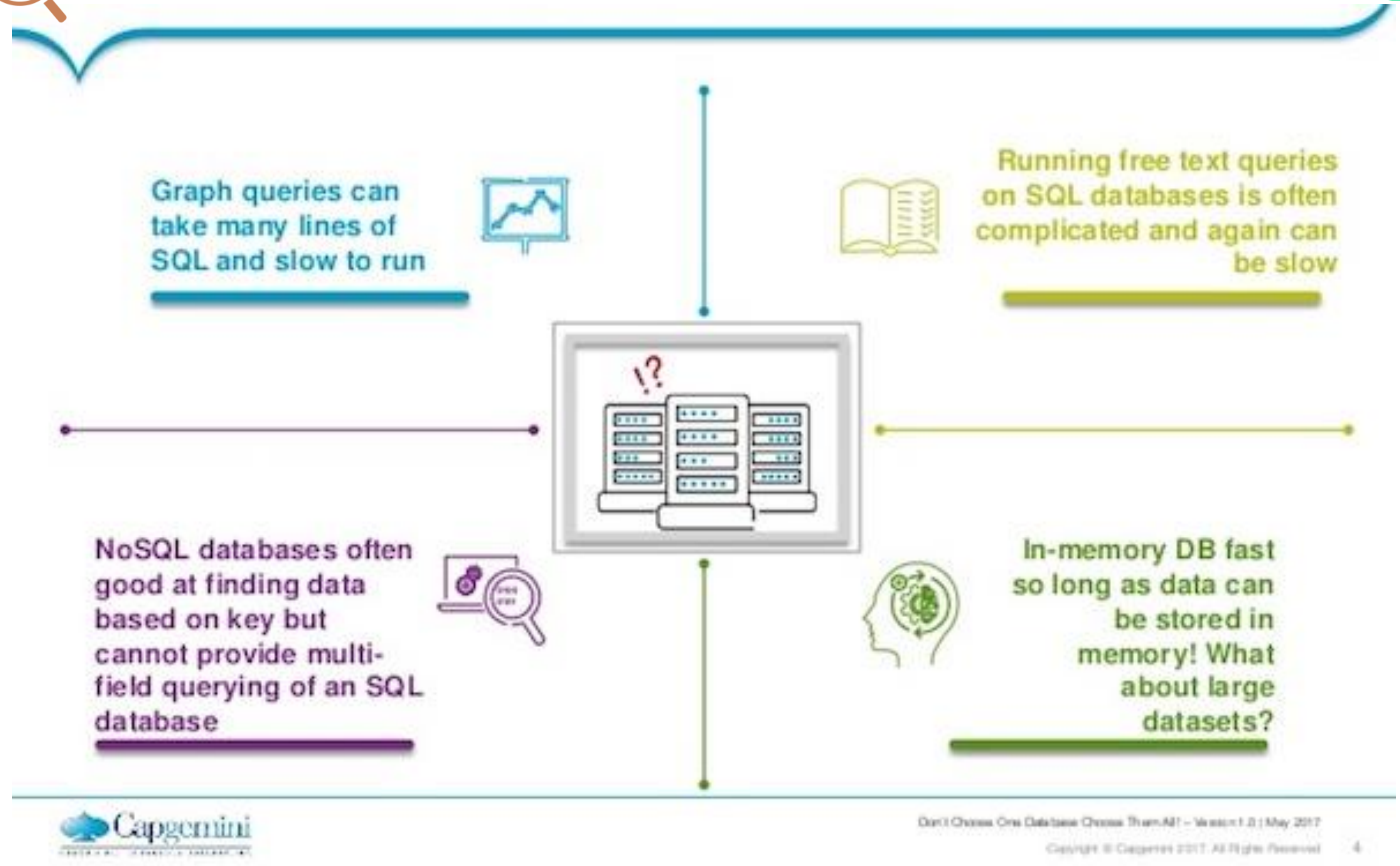


NoSQL Vs. SQL

NoSQL databases use various data models optimized for different use cases.	SQL databases use a relational data model based on tables with predefined columns and rows.
Uses query languages or APIs optimized for specific data models.	Uses a standardized language called Structured Query Language (SQL) to manipulate data.
NoSQL databases are horizontally scalable and can distribute data across multiple servers and nodes.	SQL databases are vertically scalable.
May not be fully ACID compliant	SQL databases are designed to be ACID compliant
Can provide faster performance for unstructured and non-relational data access	Can provide high performance for structured queries and data manipulation
NoSQL databases can be more cost-effective for specific use cases	SQL databases can be more expensive than NoSQL databases

ProjectPro

Challenges of using Wrong Database



Graph queries can take many lines of SQL and slow to run

Running free text queries on SQL databases is often complicated and again can be slow

NoSQL databases often good at finding data based on key but cannot provide multi-field querying of an SQL database

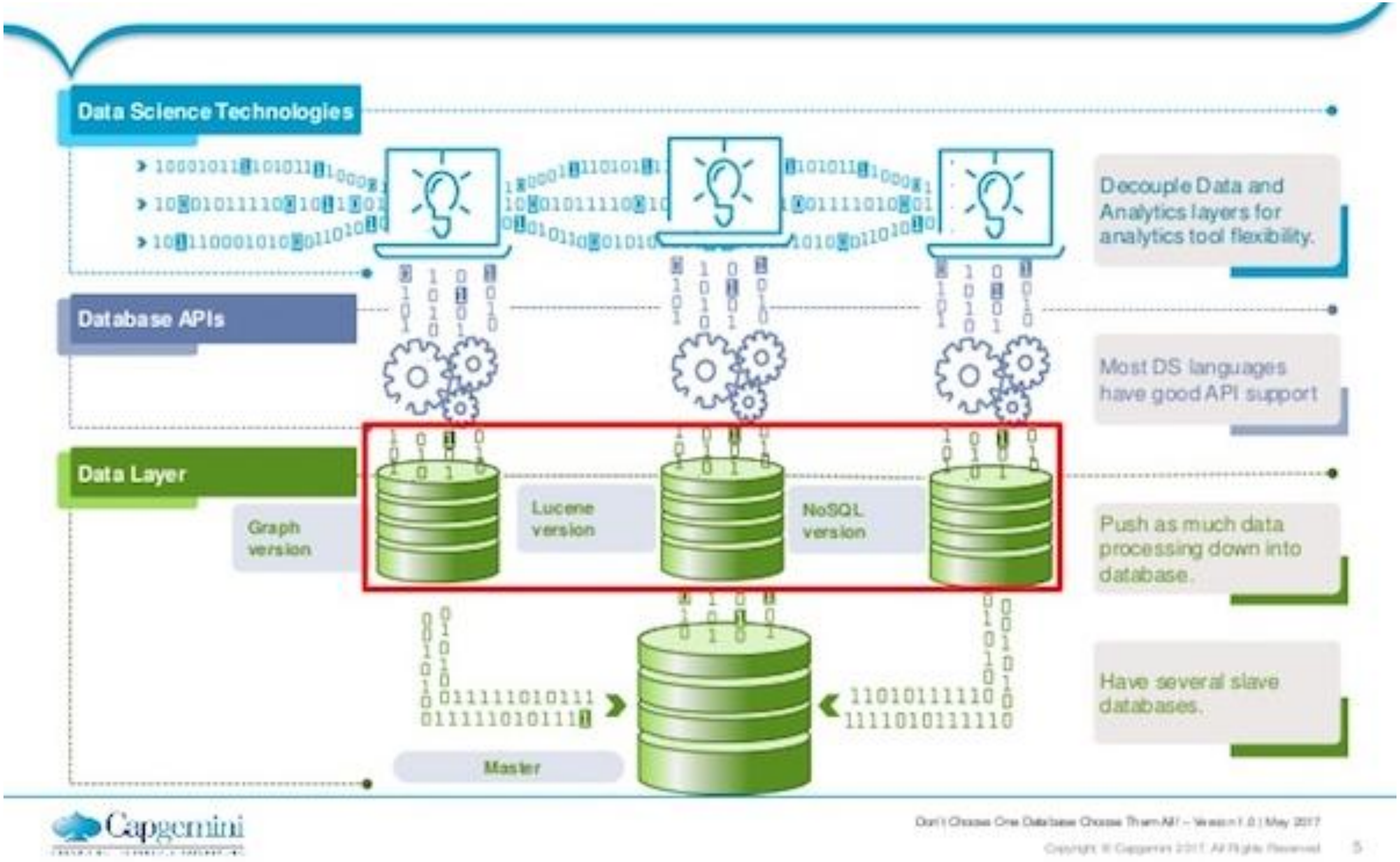
In-memory DB fast so long as data can be stored in memory! What about large datasets?

Don't Choose One Database Choose Them All! – Version 1.0 | May 2017

Copyright © Capgemini 2017. All Rights Reserved

4

Why Not just use Multiple Databases?



Decouple Data and Analytics layers for analytics tool flexibility.

Most DS languages have good API support

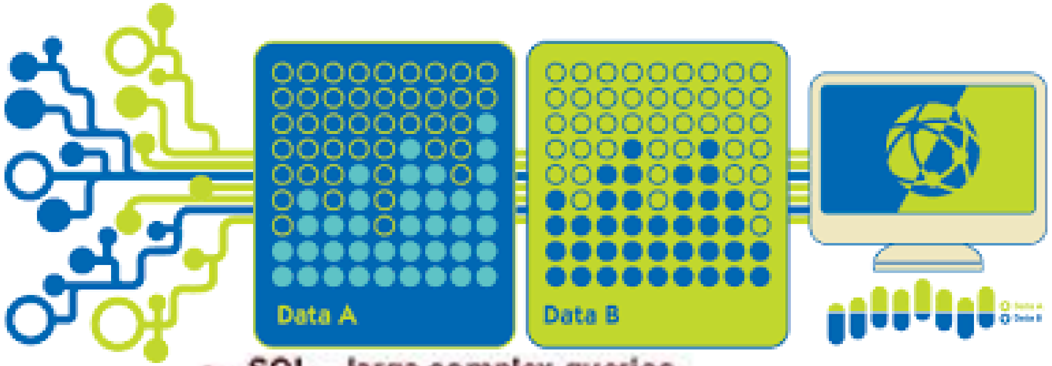
Push as much data processing down into database.

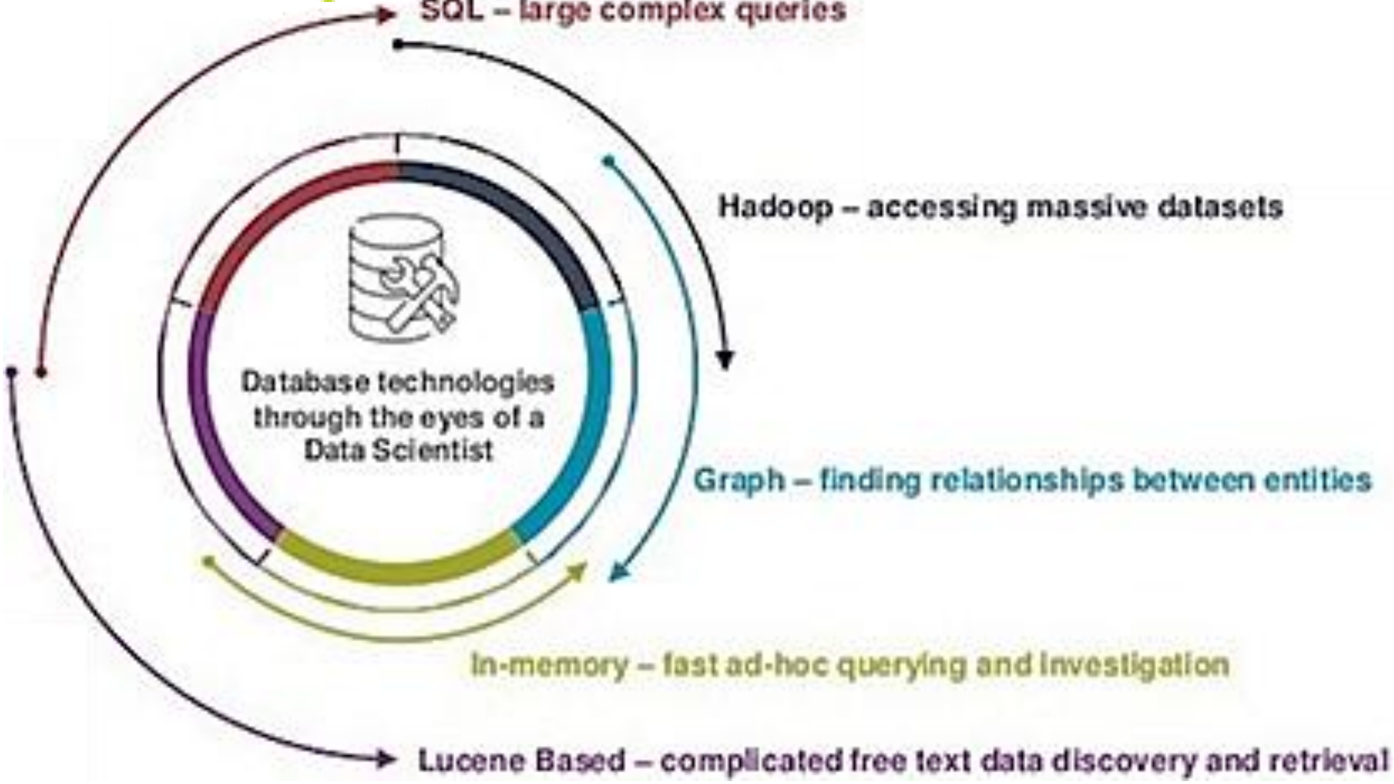
Have several slave databases.

Don't Choose One Database Choose Them All! – Version 1.0 | May 2017

Copyright © Capgemini 2017. All Rights Reserved

Big Database Solutions





SQL – large complex queries

Hadoop – accessing massive datasets

Graph – finding relationships between entities

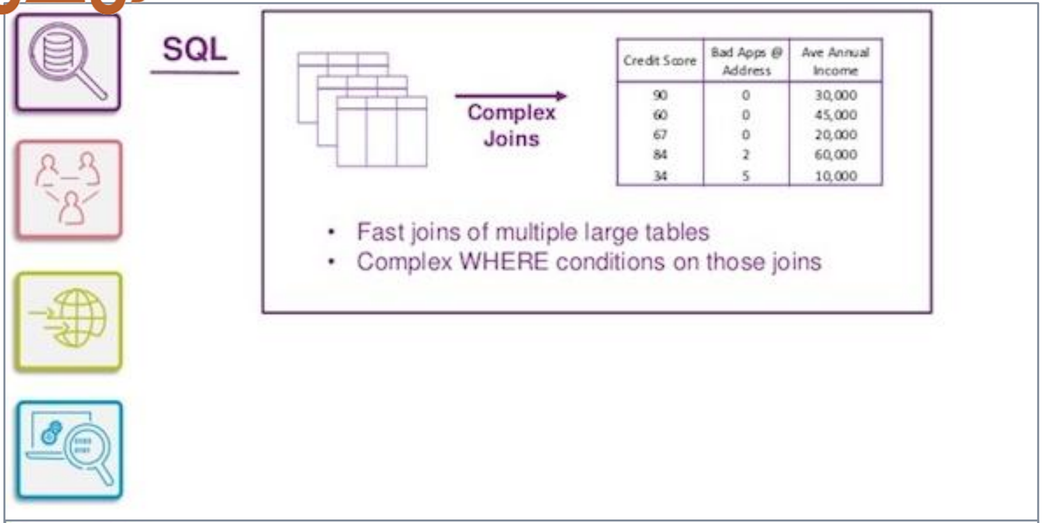
In-memory – fast ad-hoc querying and investigation

Lucene Based – complicated free text data discovery and retrieval

Database technologies through the eyes of a Data Scientist

Case Study: Car Insurance Fraud (CIF)

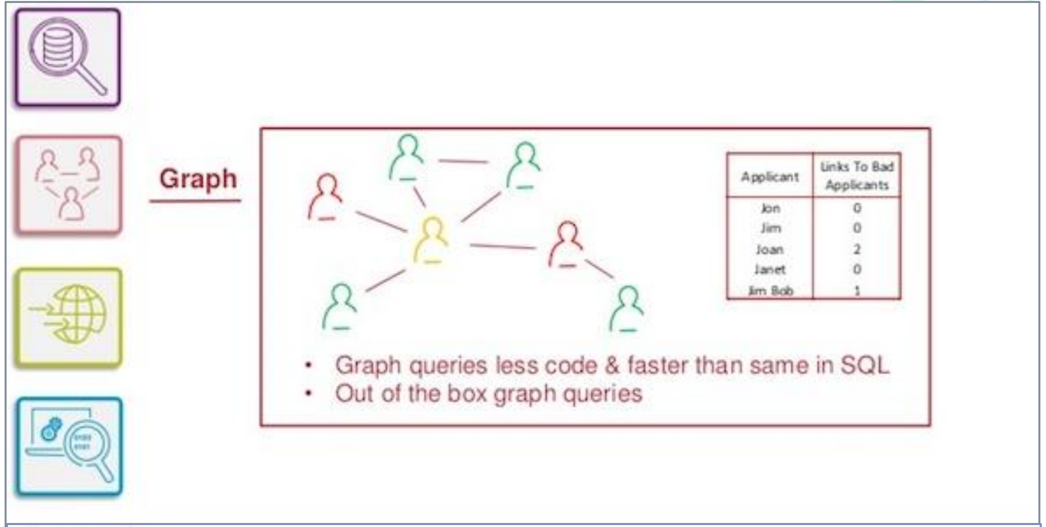
neo4j



SQL

Credit Score	Bad Apps @ Address	Ave Annual Income
90	0	30,000
60	0	45,000
67	0	20,000
84	2	60,000
34	5	10,000

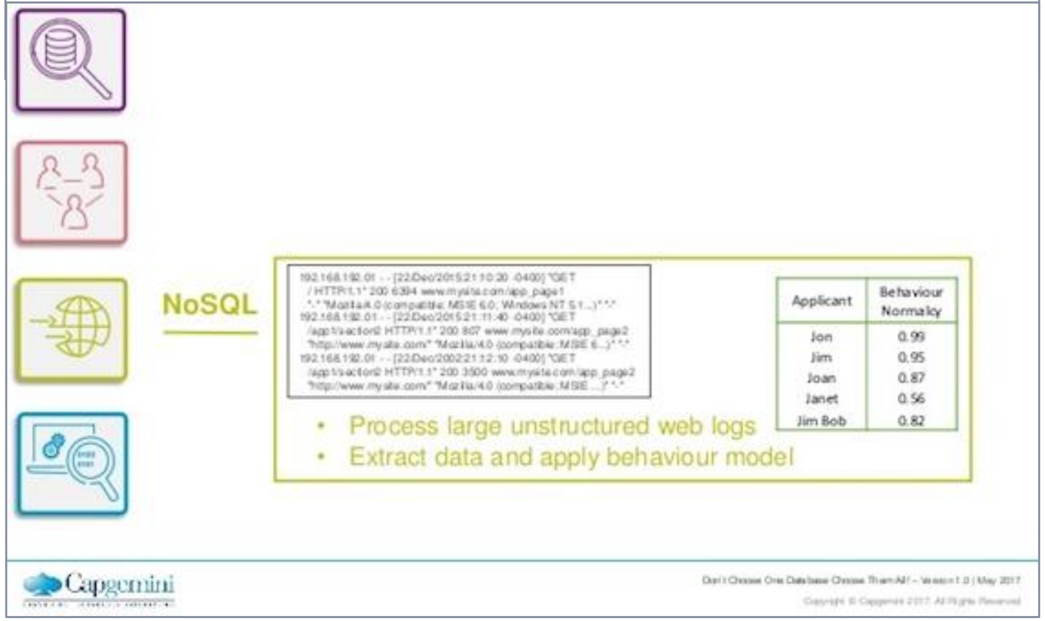
- Fast joins of multiple large tables
- Complex WHERE conditions on those joins



Graph

Applicant	Links To Bad Applicants
Jon	0
Jim	0
Joan	2
Janet	0
Jim Bob	1

- Graph queries less code & faster than same in SQL
- Out of the box graph queries



NoSQL

```
192.168.192.01 - - [22/Dec/2015:21:10:20 -0400] "GET
 / HTTP/1.1" 200 6394 www.mysite.com/app_page1
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1...)" "-"
192.168.192.01 - - [22/Dec/2015:21:11:40 -0400] "GET
 /app1/section2 HTTP/1.1" 200 807 www.mysite.com/app_page2
 "http://www.mysite.com/" "Mozilla/4.0 (compatible; MSIE 6...)" "-"
192.168.192.01 - - [22/Dec/2002:21:12:10 -0400] "GET
 /app1/section2 HTTP/1.1" 200 3500 www.mysite.com/app_page2
 "http://www.mysite.com/" "Mozilla/4.0 (compatible; MSIE ...)" "-"
```

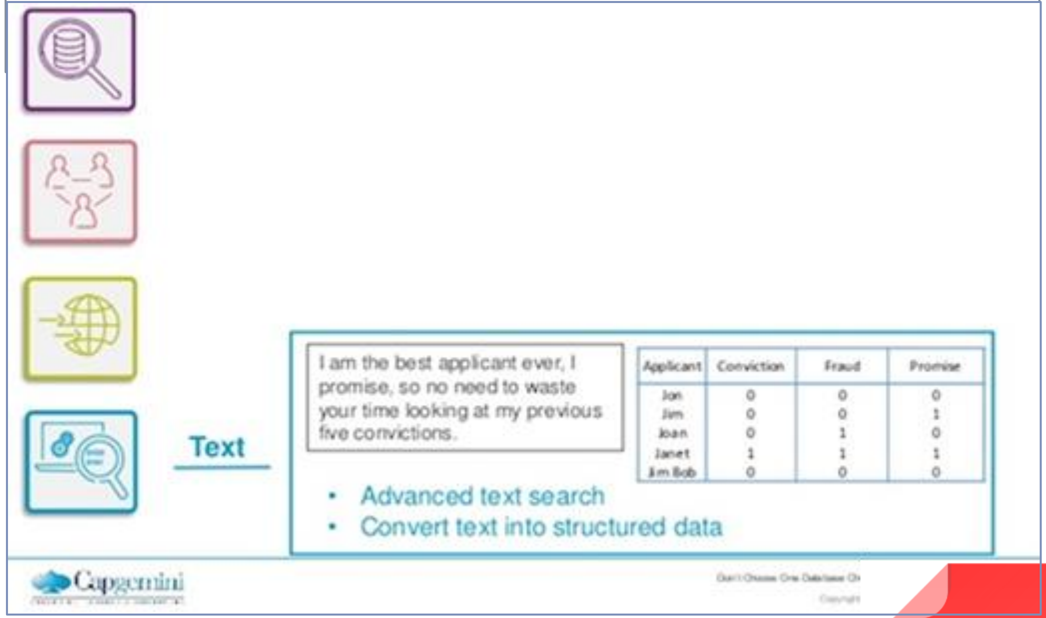
Applicant	Behaviour Normalcy
Jon	0.99
Jim	0.95
Joan	0.87
Janet	0.56
Jim Bob	0.82

- Process large unstructured web logs
- Extract data and apply behaviour model

Capgemini

Don't Choose One Database Choose Them All! – Version 1.0 | May 2017

Copyright © Capgemini 2017. All Rights Reserved



Text

I am the best applicant ever, I promise, so no need to waste your time looking at my previous five convictions.

Applicant	Conviction	Fraud	Promise
Jon	0	0	0
Jim	0	0	1
Joan	0	1	0
Janet	1	1	1
Jim Bob	0	0	0

- Advanced text search
- Convert text into structured data

Capgemini

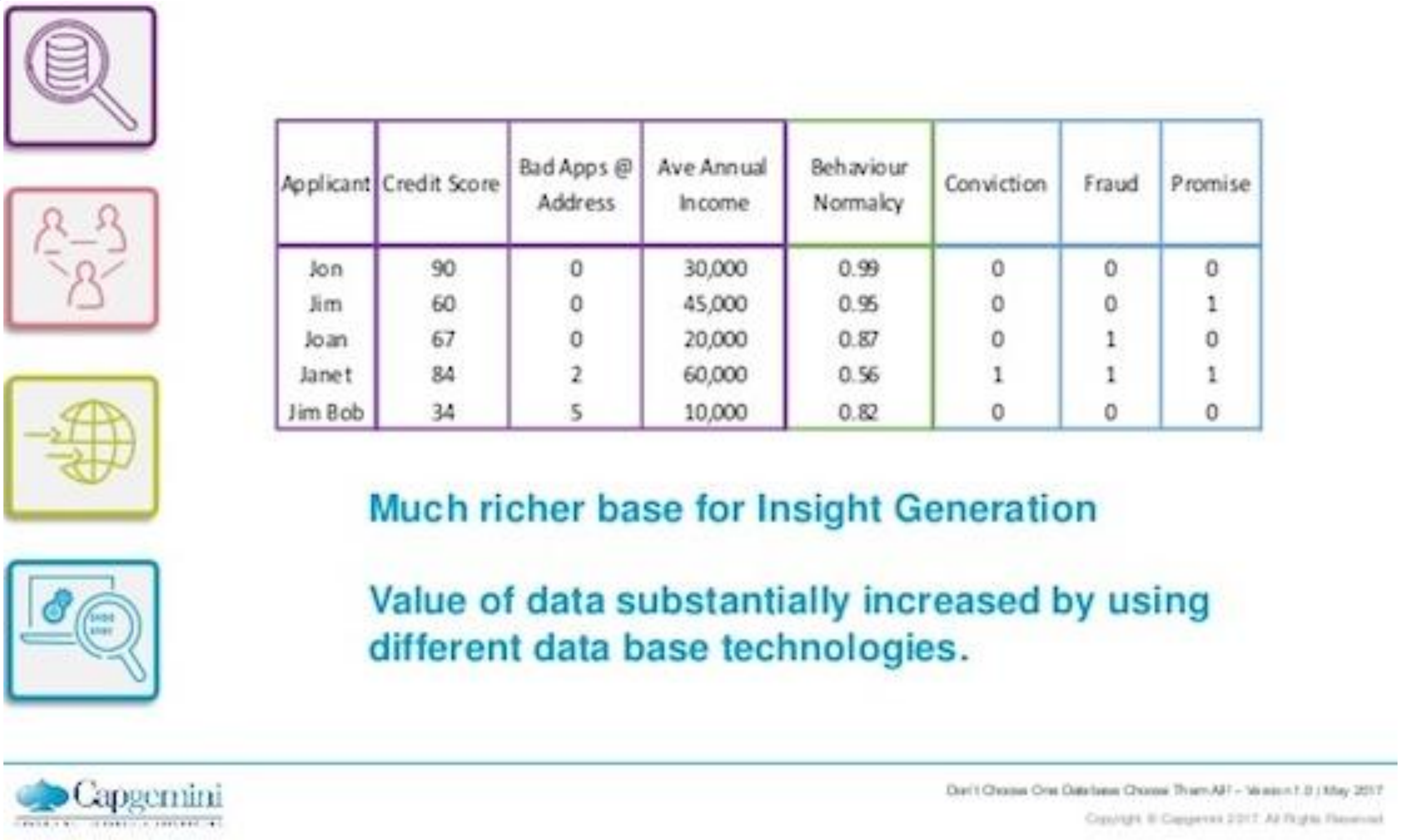
Don't Choose One Database Ch

Copyright

NoSQL

Case Study: CIF - bringing it Together

neo4j



Applicant	Credit Score	Bad Apps @ Address	Ave Annual Income	Behaviour Normalcy	Conviction	Fraud	Promise
Jon	90	0	30,000	0.99	0	0	0
Jim	60	0	45,000	0.95	0	0	1
Joan	67	0	20,000	0.87	0	1	0
Janet	84	2	60,000	0.56	1	1	1
Jim Bob	34	5	10,000	0.82	0	0	0

Much richer base for Insight Generation

Value of data substantially increased by using different data base technologies.

Capgemini

Don't Choose One Database Choose Them All! – Version 1.0 | May 2017

NoSQL