NoSQL

- Ravi Kumar Gupta
- https://kravigupta.in

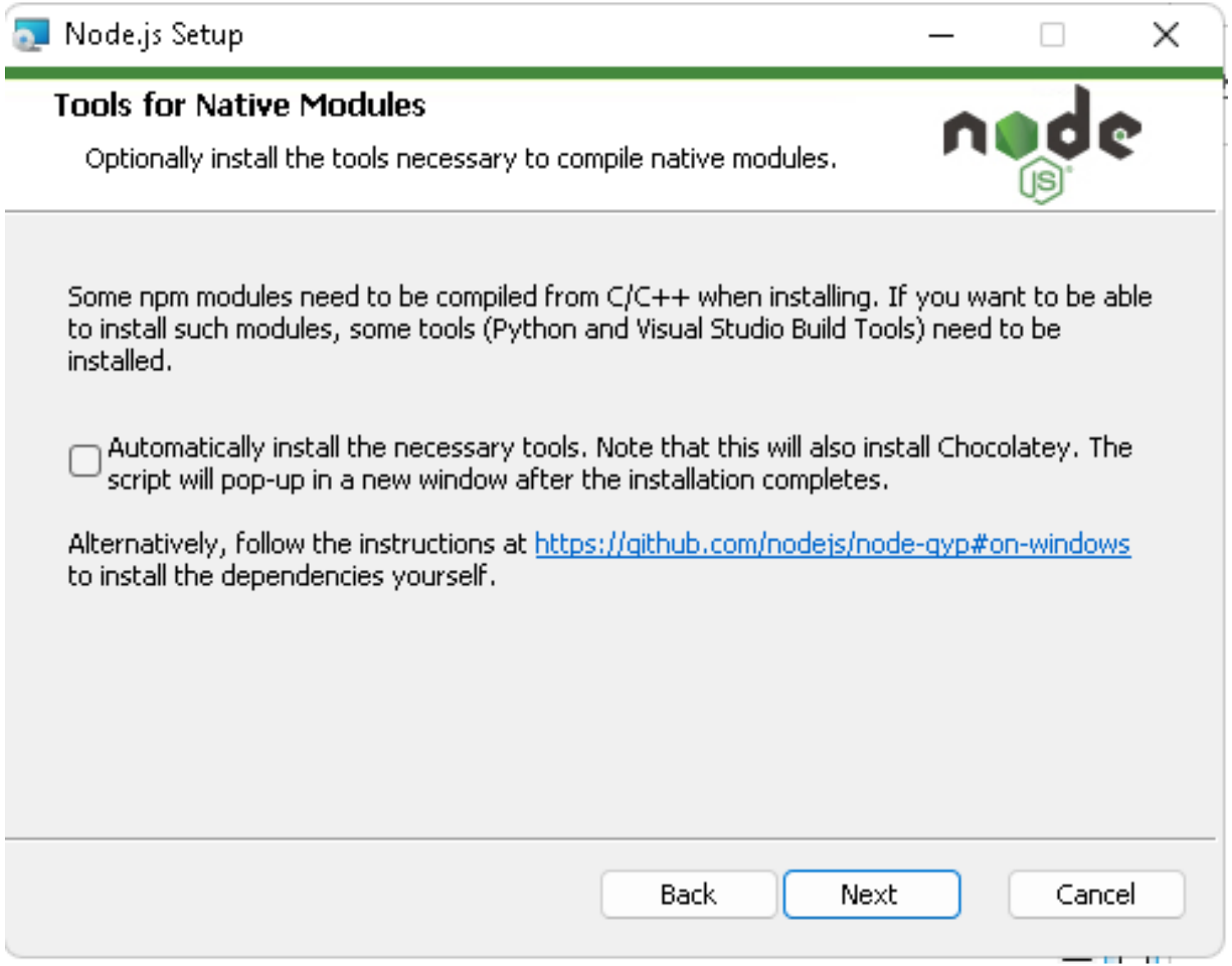
Agenda

- MongoDB Installation
- NodeJS Installation
- Generating Data into MongoDB
- Mongo Shell
- Accessing DB

Software Needed

- Mongodb
- NodeJs
- NodeJs Scripts
  - https://kravigupta.in/bda_slides/mongo-load-gen.zip

NodeJS Installation – Additional tools



NodeJS Installation

Administrator: Windows PowerShell

```
Forcing web requests to allow TLS v1.2 (Required for requests to Chocolatey.org)
Getting latest version of the Chocolatey package for download.
Not using proxy.
Getting Chocolatey from https://community.chocolatey.org/api/v2/package/chocolatey/2.2.0.
Downloading https://community.chocolatey.org/api/v2/package/chocolatey/2.2.0 to C:\Users\Dell\AppData\Local\Temp\chocola
tey\chocoInstall\chocolatey.zip
Not using proxy.
Extracting C:\Users\Dell\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip to C:\Users\Dell\AppData\Local\Temp\c
hocolatey\chocoInstall
Installing Chocolatey on the local machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
  Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
  before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
  (i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
  and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey folders if they do not already exist.

chocolatey.nupkg file not installed in lib.
 Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
WARNING: Not setting tab completion: Profile file does not exist at
'C:\Users\Dell\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
 first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
Chocolatey v2.2.0
Upgrading the following packages:
python;visualstudio2019-workload-vctools
By upgrading, you accept licenses for the packages.
python is not installed. Installing...
Progress: Downloading chocolatey-compatibility.extension 1.0.0... 100%

chocolatey-compatibility.extension v1.0.0 [Approved]
chocolatey-compatibility.extension package files upgrade completed. Performing other installation steps.
 Installed/updated chocolatey-compatibility extensions.
 The upgrade of chocolatey-compatibility.extension was successful.
  Software installed to 'C:\ProgramData\chocolatey\extensions\chocolatey-compatibility'
Progress: Downloading chocolatey-core.extension 1.4.0... 100%

chocolatey-core.extension v1.4.0 [Approved]
```

NodeJS Installation

Administrator: Windows PowerShell

```
KB3033929 package files upgrade completed. Performing other installation steps.
Skipping installation because update KB3033929 does not apply to this operating system (Microsoft Windows 11 Pro).
 The upgrade of KB3033929 was successful.
  Software install location not explicitly set, it could be in package or
  default install location of installer.
Progress: Downloading vcredist140 14.36.32532... 100%

vcredist140 v14.36.32532 [Approved]
vcredist140 package files upgrade completed. Performing other installation steps.
Downloading vcredist140-x86
  from 'https://download.visualstudio.microsoft.com/download/pr/eaab1f82-787d-4fd7-8c73-f782341a0c63/5365A927487945ECB04
0E143EA770ADBB296074ECE4021B1D14213BDE538C490/VC_redist.x86.exe'
Progress: 100% - Completed download of C:\Users\Dell\AppData\Local\Temp\chocolatey\vcredist140\14.36.32532\VC_redist.x86
.exe (13.2 MB).
Download of VC_redist.x86.exe (13.2 MB) completed.
Hashes match.
Installing vcredist140-x86...
vcredist140-x86 has been installed.
Downloading vcredist140-x64 64 bit
  from 'https://download.visualstudio.microsoft.com/download/pr/eaab1f82-787d-4fd7-8c73-f782341a0c63/917C37D816488545B70
AFFD77D6E486E4DD27E2ECE63F6BBAAF486B178B2B888/VC_redist.x64.exe'
Progress: 100% - Completed download of C:\Users\Dell\AppData\Local\Temp\chocolatey\vcredist140\14.36.32532\VC_redist.x64
.exe (24.18 MB).
Download of VC_redist.x64.exe (24.18 MB) completed.
Hashes match.
Installing vcredist140-x64...
vcredist140-x64 has been installed.
  vcredist140 may be able to be automatically uninstalled.
 The upgrade of vcredist140 was successful.
  Software installed as 'exe', install location is likely default.
Progress: Downloading vcredist2015 14.0.24215.20170201... 100%

vcredist2015 v14.0.24215.20170201 [Approved]
vcredist2015 package files upgrade completed. Performing other installation steps.
 The upgrade of vcredist2015 was successful.
  Software installed to 'C:\ProgramData\chocolatey\lib\vcredist2015'
Progress: Downloading python311 3.11.4... 100%

python311 v3.11.4 [Approved]
python311 package files upgrade completed. Performing other installation steps.
Installing 64-bit python311...
```

RDBMS vs NoSQL

Feature	RDBMS	NoSQL
Data Model	Tabular, with fixed schema	Flexible, schema-less or with dynamic schema
Scalability	Vertical (scaling up)	Horizontal (scaling out)
Consistency Model	Strong consistency (ACID)	Eventual consistency (BASE)
Transaction Support	Full ACID transactions	Limited or no ACID transactions
Query Language	SQL (Standardized)	Varies by system (e.g., MongoDB Query)
Performance	Optimized for complex queries	Optimized for read/write at scale
Relationships	Strong support for relationships (joins)	Limited or no support for complex joins
Maturity & Community	Mature, with extensive community support	Newer, with growing community support
Security Features	Generally robust	Can vary widely between systems
Use Case	Complex business applications, analytics	Big data, real-time applications, IoT

RDBMS vs NoSQL

RDBMS: Relational databases are based on a fixed schema with tables, columns, and relationships. They offer strong consistency and are suitable for applications requiring complex queries and transactions.

NoSQL: NoSQL databases are more flexible, offering various data models like key-value, document, column-family, or graph. They are designed for horizontal scalability and are often used in scenarios where data is massive or rapidly changing.

ACID

Atomicity	Consistency	Isolation	Durability
• Transactions are all-or-nothing. • If one part fails, the entire transaction fails, and the database state is left unchanged. • Example: Bank transfer between two accounts.	• Ensures that the database remains in a consistent state before and after the transaction. • Enforces constraints, cascades, triggers, etc. • Example: Enforcing a unique constraint on an email field.	• Concurrent transactions are executed in isolation from each other. • Ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially. • Example: Isolating a report generation from an ongoing data update.	• Once committed, the results of a transaction are permanent. • Survives system failures. • Example: Writing transaction logs to a disk to recover from crashes.

CAP Theorem

- **C**onsistency
- **A**vailability
- **P**artition Tolerance
- Also known as Brewer’s theorem

CAP Theorem

- CAP theorem for NoSQL states that there are three basic requirements which exist in a special relation when designing applications for a distributed architecture.

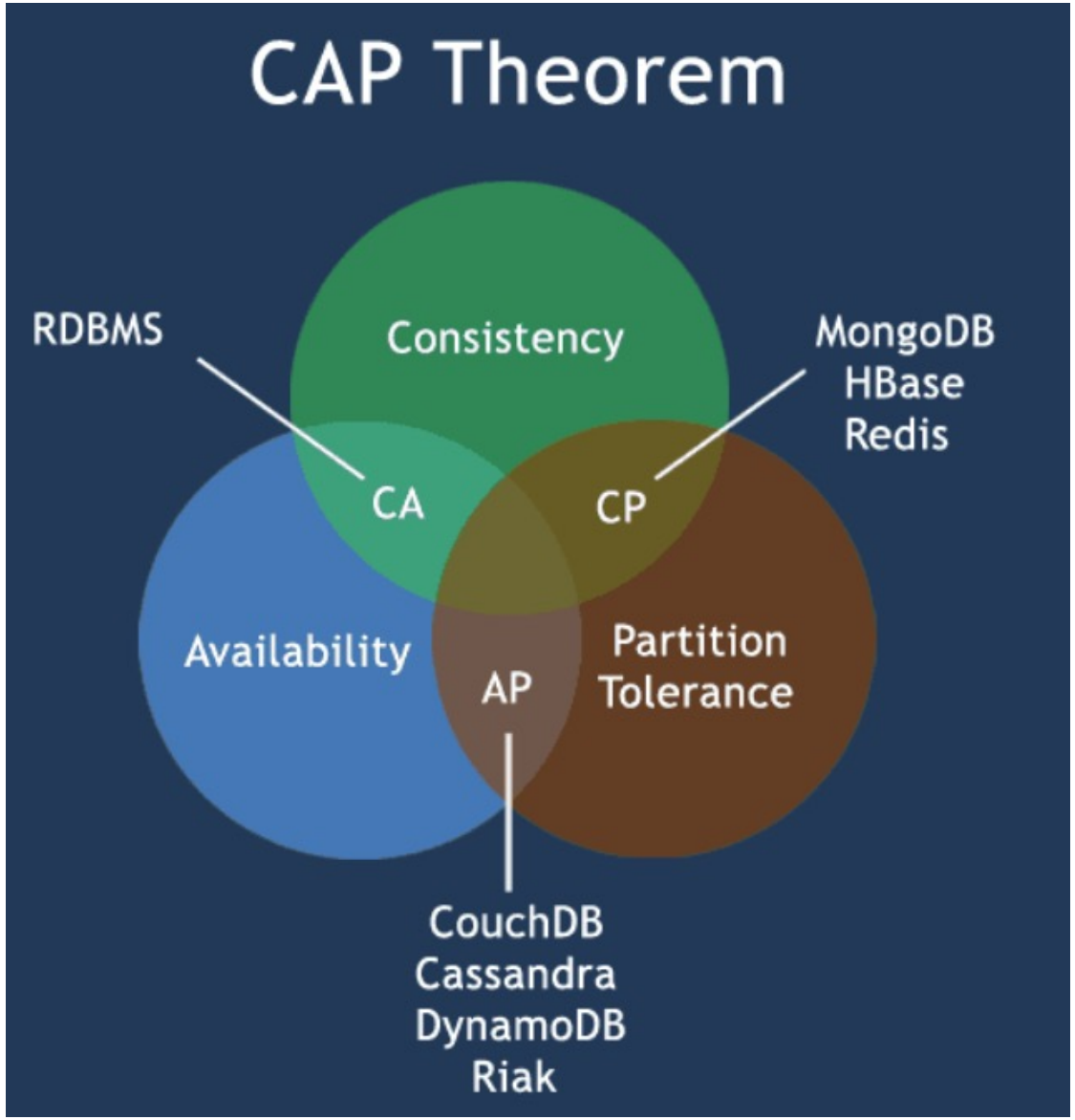
- **Consistency** - This means that the data in the database remains consistent after the execution of an operation. *Ex. after an update operation all clients see the same data.*
- **Availability** - This means that the system is always on (service guarantee availability), no downtime.
- **Partition Tolerance** - This means that the system continues to function even when the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

CAP Theorem

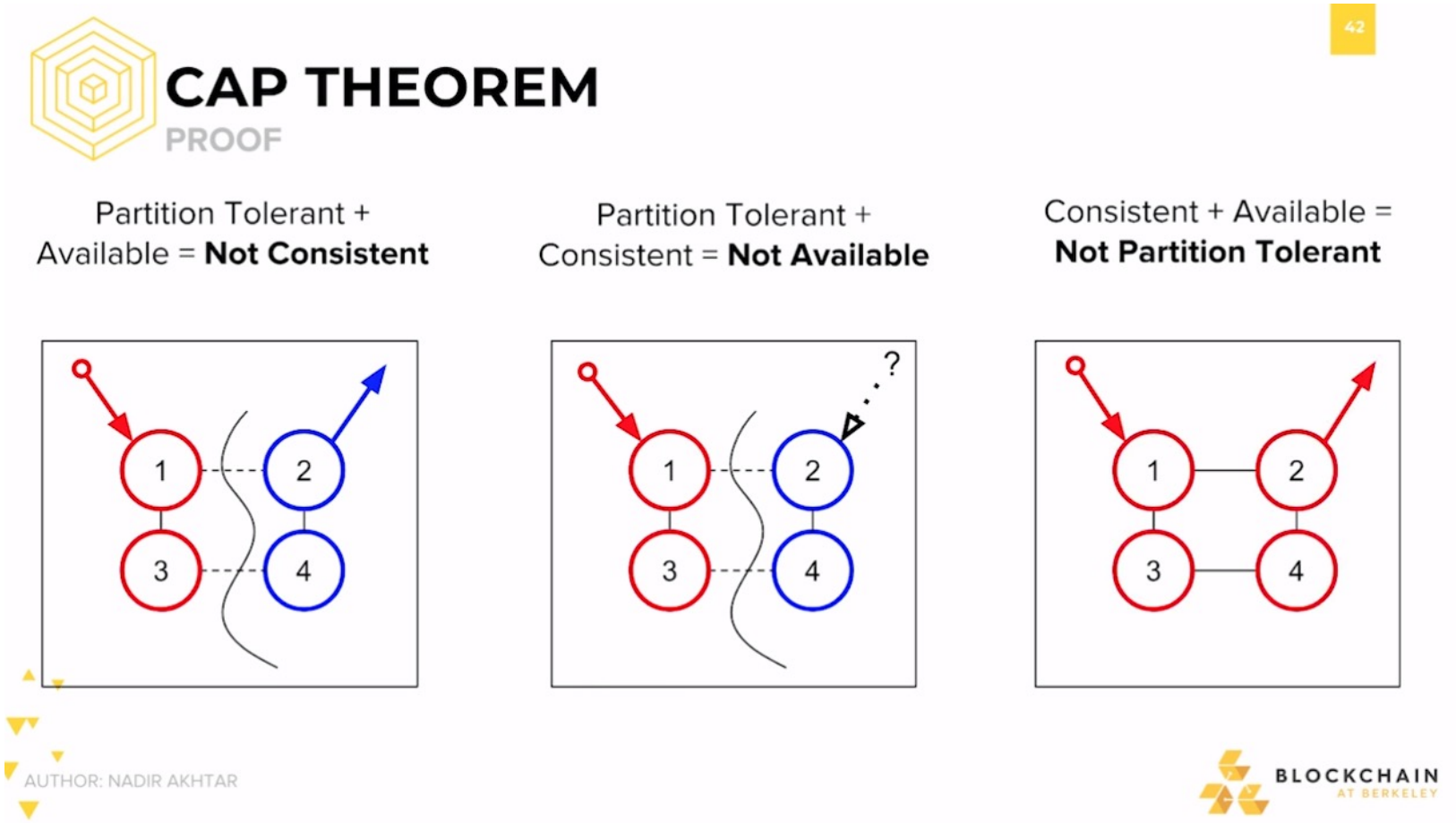
- Theoretically, it is impossible to fulfil all 3 requirements.
- CAP provides the basic requirements for a distributed system to follow 2 of the 3 requirements.
- Therefore, all the current NoSQL database follow the different combinations of the C, A, P from the CAP theorem.

- **CA -** Single site cluster, therefore all nodes are always in contact. When a partition occurs, the system blocks.
- **CP -**Some data may not be accessible, but the rest is still consistent/accurate.
- **AP -** System is still available under partitioning, but some of the data returned may be inaccurate.

CAP Theorem


CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
AP
Partition
Tolerance
CouchDB
Cassandra
DynamoDB
Riak

CAP Theorem


42
CAP THEOREM
PROOF
Partition Tolerant + Available = Not Consistent
Partition Tolerant + Consistent = Not Available
Consistent + Available = Not Partition Tolerant
1
2
3
4
?
AUTHOR: NADIR AKHTAR
BLOCKCHAIN
AT BERKELEY

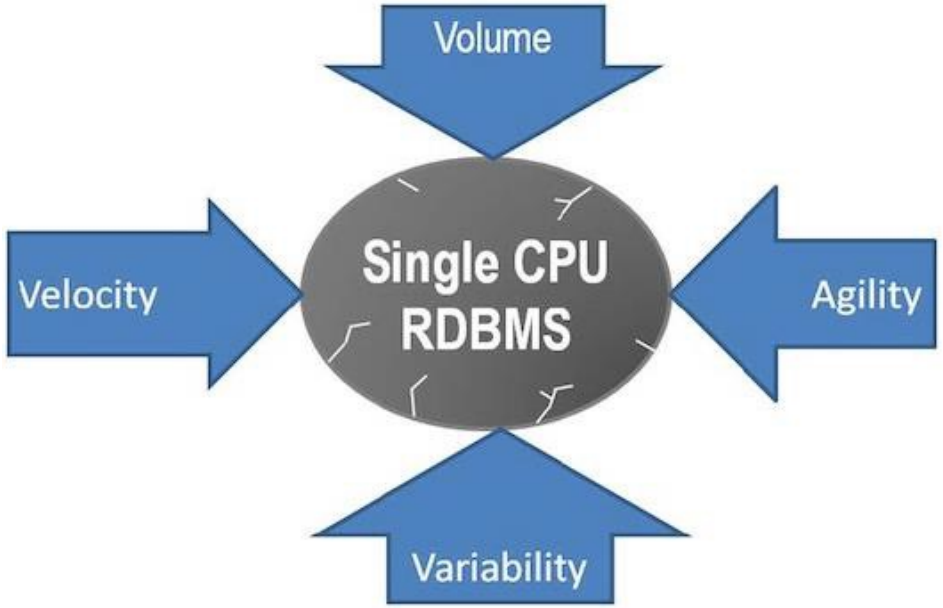
BASE Properties of NoSQL

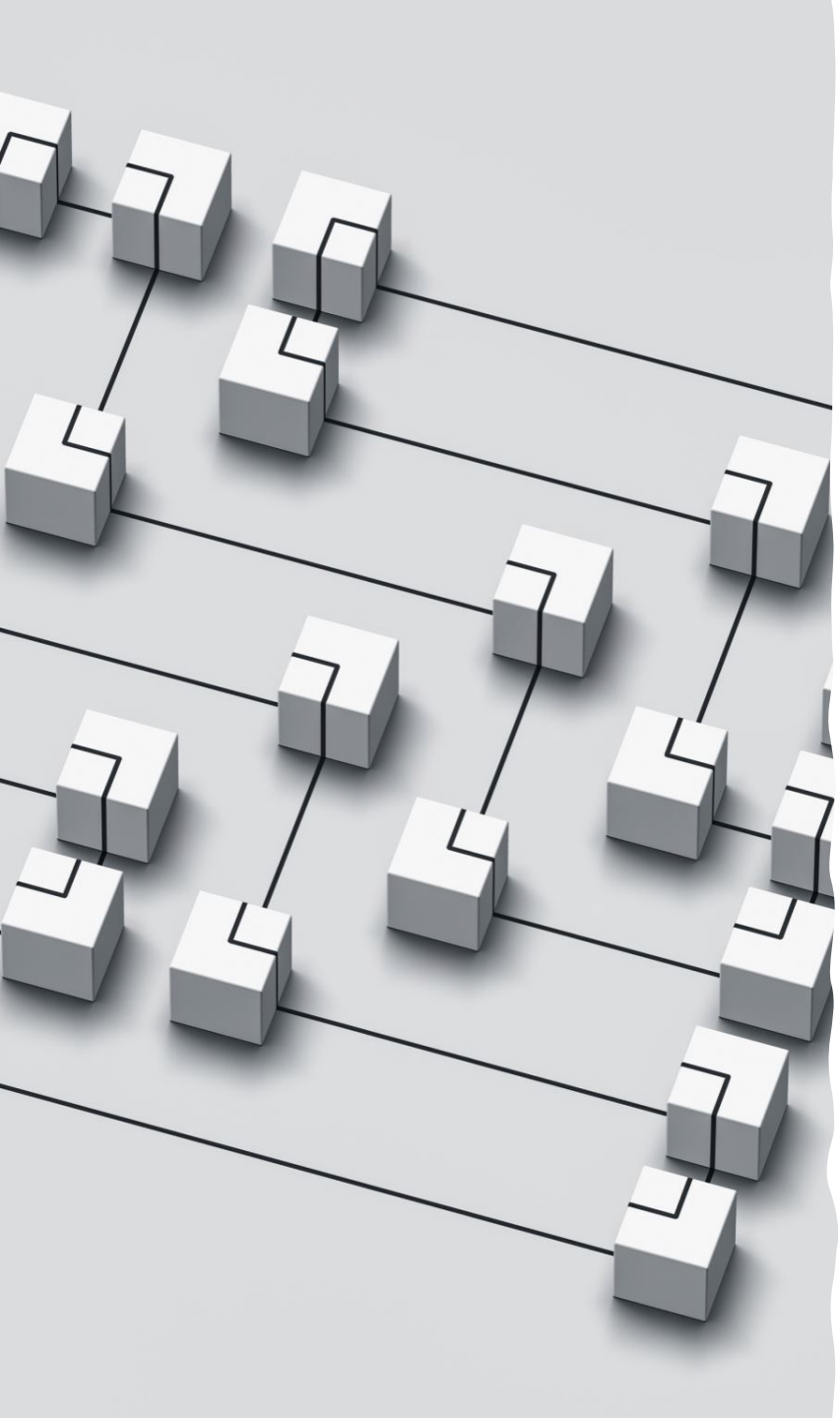
- **Basically Available**
  - System responds to all requests but without guarantee of being up-to-date.
  - Example: Amazon DynamoDB with eventual consistency.
- **Soft State**
  - The state of the system may change over time, even without input.
  - Reflects the flexible nature of the system.
  - Example: Temporary inconsistencies in a distributed cache like Memcached.
- **Eventually Consistent**
  - Given enough time, the system will become consistent.
  - Given that system does not receive input during that time
  - Example: Eventual consistency in Apache Cassandra.

NoSQL Business Drivers

- Demands of following play a key role in emergence of NoSQL solutions

1. Volume
2. Velocity
3. Variability
4. Agility





Desirable Features of NoSQL

- 24x7 Data availability
- Location transparency | Location independence
- Schema-less data model
- Modern day transaction analysis
- Architecture that suits big data
- Analytics and business intelligence

Big Data Architecture Considerations

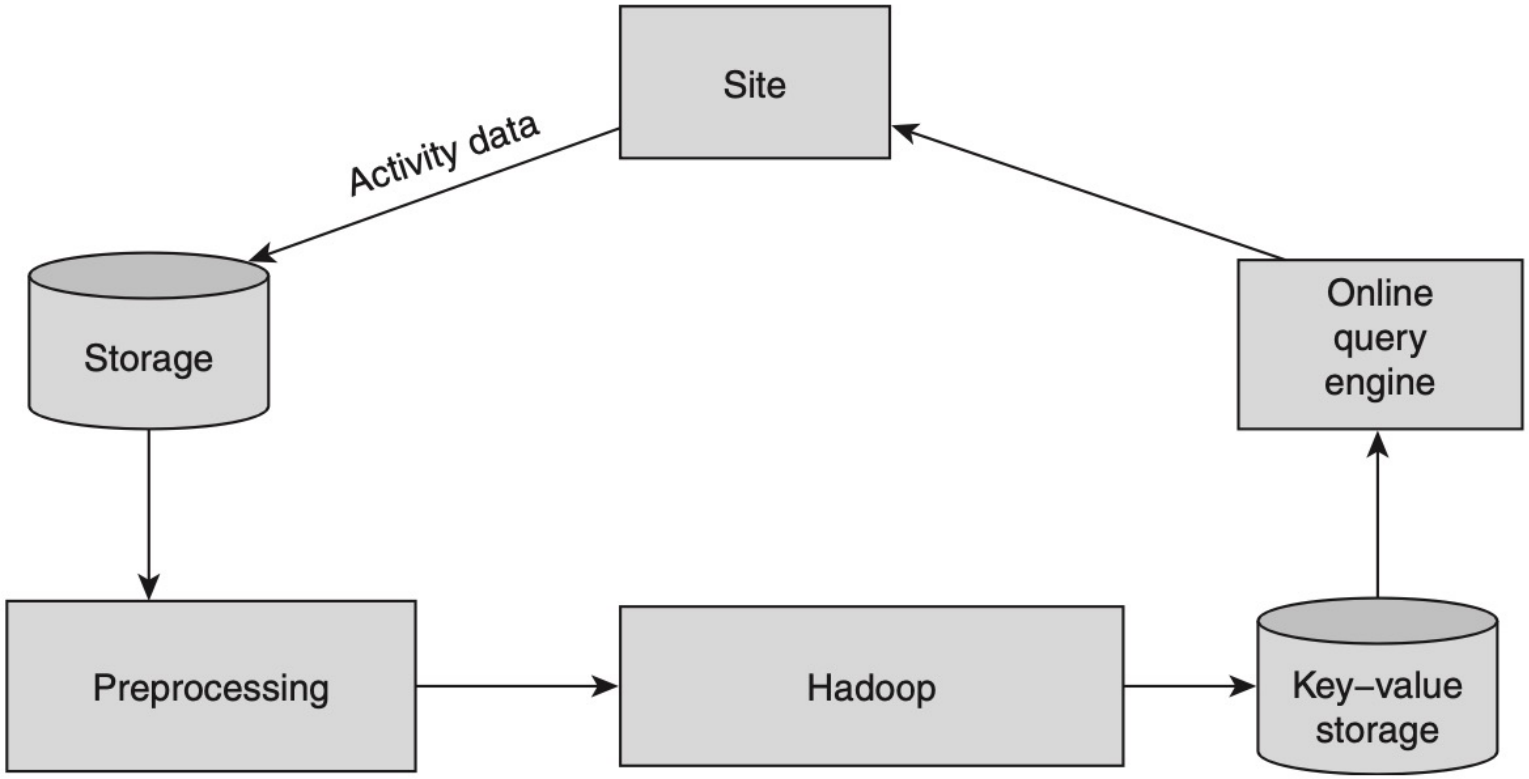
SCALE OF DATA SOURCES

SPEED IS ESSENTIAL

CHANGE IN STORAGE MODELS

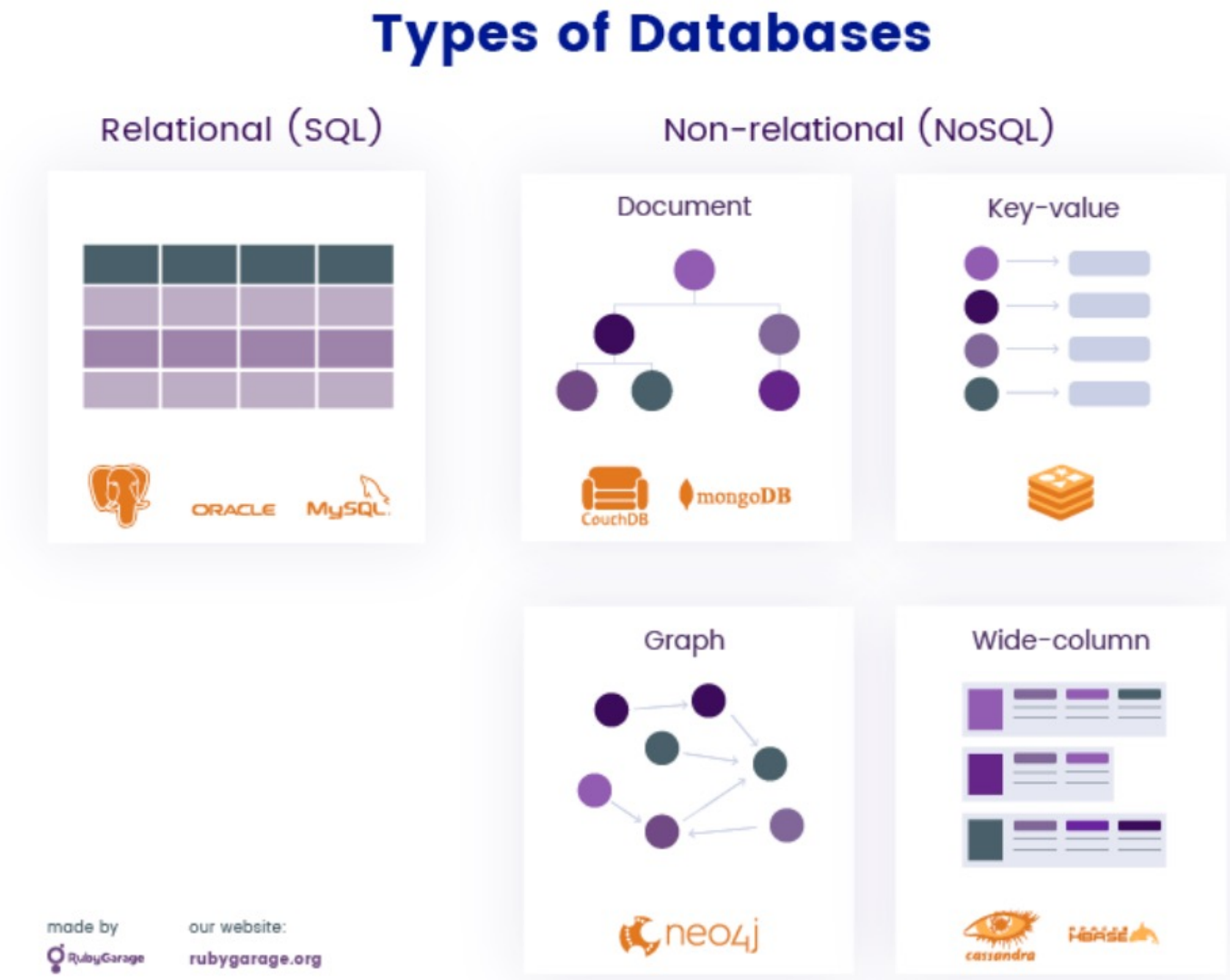
MULTIPLE COMPUTE MODELS

A sample architecture



Types of NoSQL Data Stores

- Key-value store
- Column Store
- Document Store
- Graph Store



NoSQL Classification

- **Data Model**
  1. Key-Value Store
  2. Column Family Store
  3. Document Store
  4. Graph Store

- **Properties (Consistency/Availability/ Partition Tolerance-Trade-Off)**

1. CAP (but not all three at once!)
2. AP tolerant
3. CP tolerant
4. CA tolerant

Key Value Store

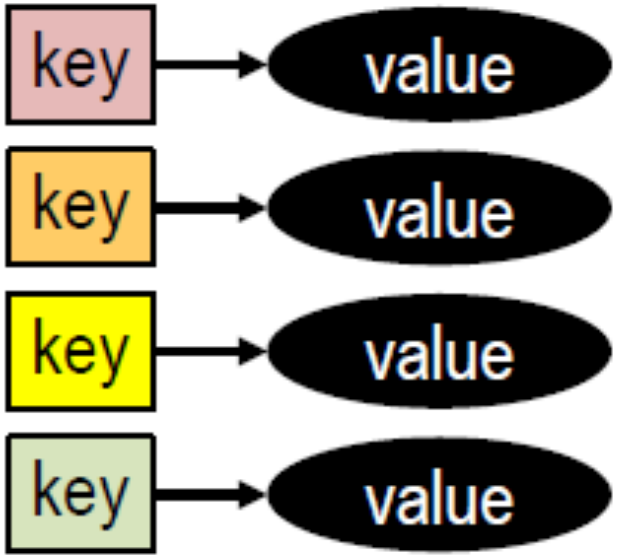
Key-Value Store

- Designed to handle huge amounts of data.
- Based on Amazon’s Dynamo paper.
- Key value stores allow developer to store schema-less data.
- In the key-value storage, database stores data as hash table where each key is unique and the value can be string, JSON, BLOB (Binary Large OBject) etc.

e.g. a key-value pair might consist of a key like “College_Name" that is associated with a value like “DBIT".

- Can be used as collections, dictionaries, associative arrays
- Follow the 'Availability' and 'Partition' aspects of CAP theorem.
- Work well for shopping cart contents, or individual values like color schemes, a landing page URI, or a default account number

Key Value Store

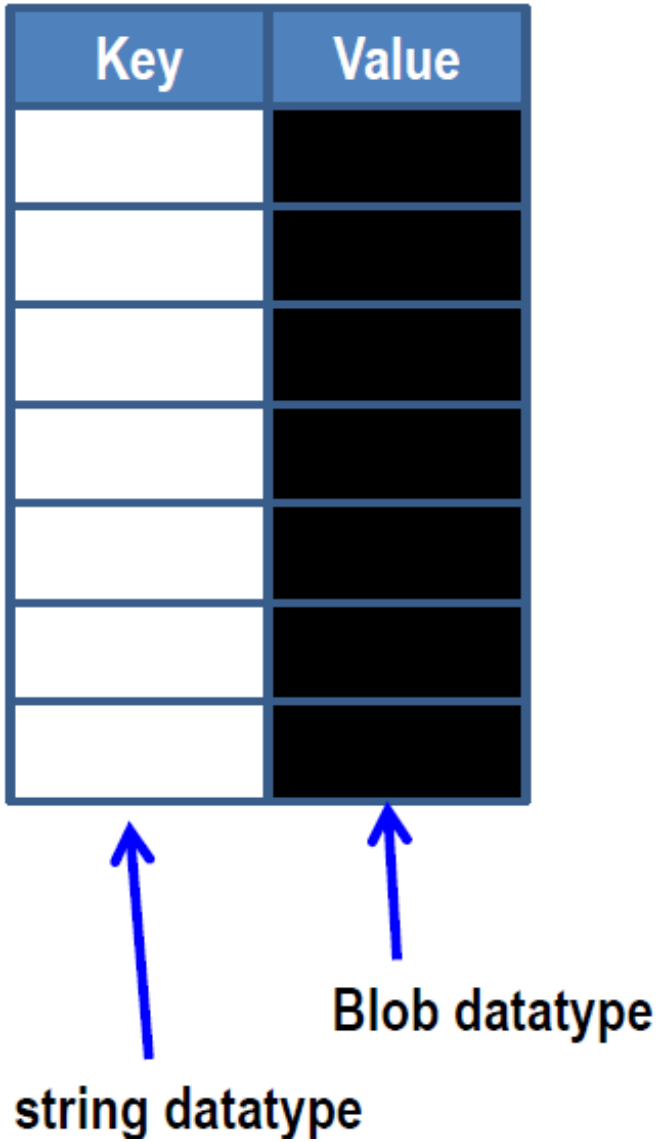


Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Key Value Store

- A table with two columns and a simple interface
  - Add a key-value
  - For this key, give me the value
  - Delete a key

- Blazingly fast and easy to scale (no joins)



Key Value Store

The "key" is just the word "gouge"

The "value" is all the definitions and images

gouge |gouj|

noun

1 a chisel with a concave blade, used in carpentry, sculpture, and surgery.

2 an indentation or groove made by gouging.

gouge 1

verb [trans.]

1 make (a groove, hole, or indentation) with or as if with a gouge : *the channel had been* ***gouged out*** *by the ebbing water.*

• make a rough hole or indentation in (a surface), esp. so as to mar or disfigure it : *he had wielded the blade inexpertly, gouging the grass in several places.*

• (**gouge something out**) cut or force something out roughly or brutally : *one of his eyes had been gouged out.*

2 informal overcharge; swindle : *the airline ends up gouging the very passengers it is supposed to assist.*

Key Value Store

- **Schema less format**
  - Key is auto generated ,value can be string JSON, BLOB
  - Key could be web page, file path, image name, SQL Query
  - Key –value uses hash table, with unique key and pointer to data .
  - Bucket is logical group of key, different bucket can have identical key
  - Real key is a hash (bucket + key)
  - Cache mechanism improves the performance

❖ Client can read/write values Getkey(fetch key),

❖ Putkey (associate value with key),

❖ Multigetkey(fetch list associated with numerous key),

❖ Deletekey(remove key)

- **<u>Disadvantage :</u>** Consistency impossible, as volume increase difficult to maintain unique key

Key Value Store

Rules:

- **Distinct key:** All key are distinct
- **No query on values:** no query can be performed on values of table

- In general, key-value stores have no query language.
- They provide a way to store, retrieve and update data using simple *get, put* and *delete* commands;
- The path to retrieve data is a direct request to the object in memory or on disk.

Example - Redis

- **<u>Open source in-memory key-value store with optional durability</u>**
- Focus on high-speed reads and writes of common data structures to RAM
- Allows simple lists, sets and hashes to be stored within the value and manipulated
- In-memory database (IMDB, also main memory database)
  - Database management system that primarily relies on main memory for computer data storage.
  - Faster than disk-optimized databases because disk access is slower than memory access

Example - Redis

- Main memory databases store data on volatile memory devices.
- These devices lose all stored information when the device loses power or is reset.
  - Lacks support for the "durability" portion of the ACID
  - Supports atomicity, consistency and isolation of ACID

Amazon DynamoDB

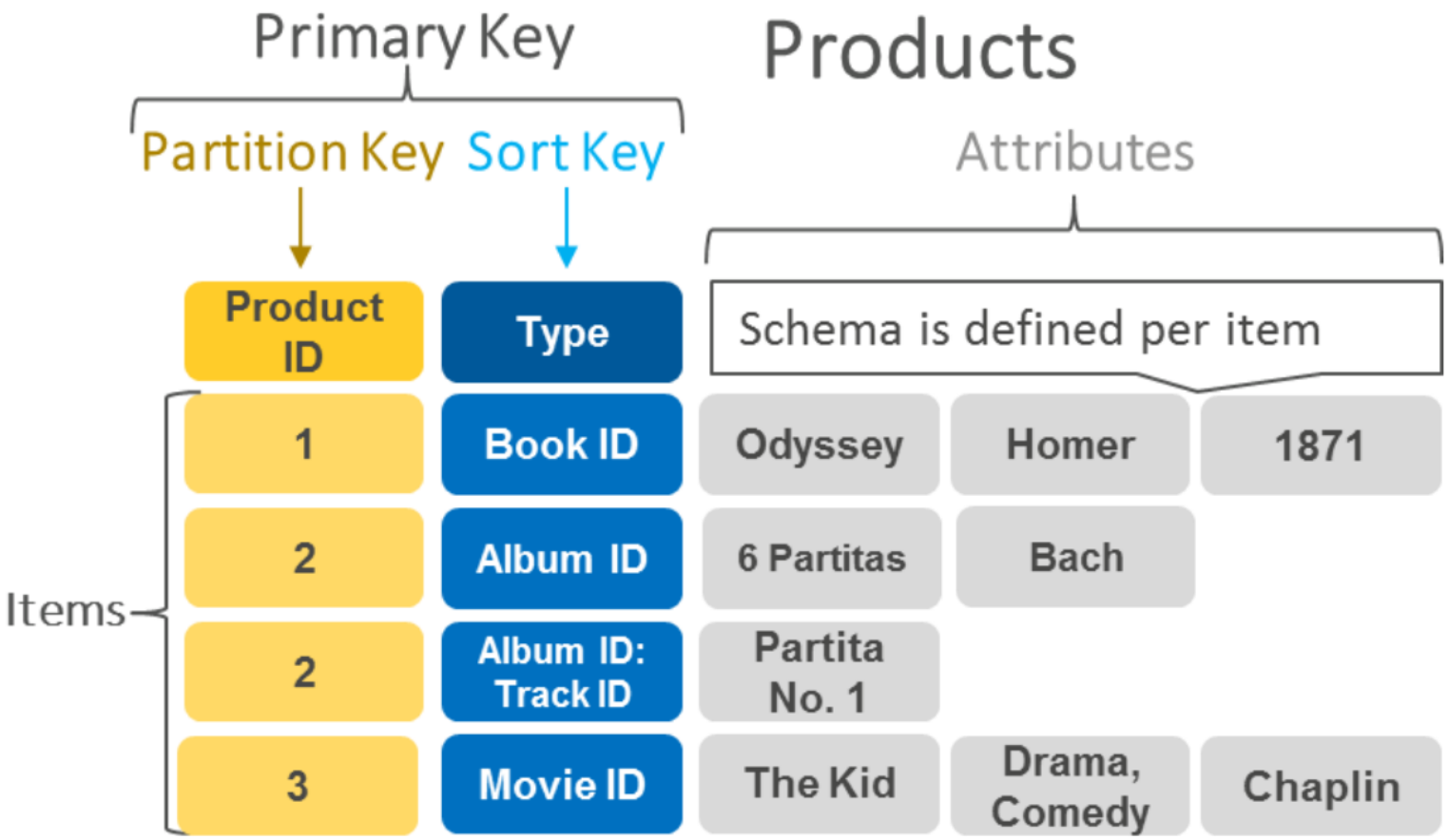
- Based around scalable key-value store
- Fastest growing product in Amazon's history
- SSD **only** database service
- Stored in 3 geographical regions
- Focus on throughput not storage and predictable read and write times
- Strong integration with S3 and Elastic MapReduce

Amazon DynamoDB

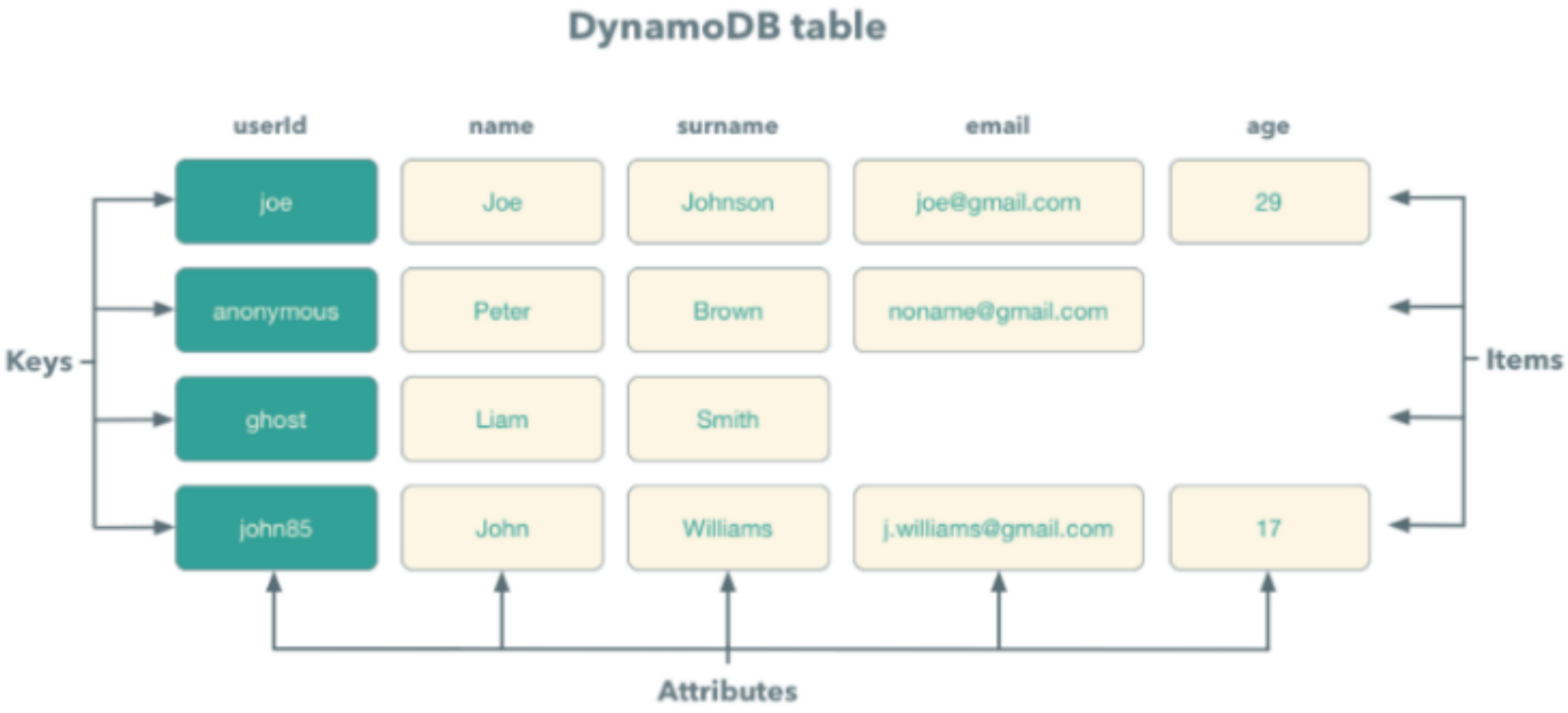
Amazon DynamoDB

- An Item is composed of a primary or composite key and a flexible number of attributes.
- No explicit limitation on the number of attributes associated with an individual item
  - But the aggregate size of an item, including all the attribute names and attribute values, cannot exceed 400 KB.
- A table is a collection of data items, just as a table in a relational database is a collection of rows.
- Each table can have an infinite number of data items.
- Fully managed, multi-region, multi-active, durable database
- DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second.

Amazon DynamoDB



Amazon DynamoDB

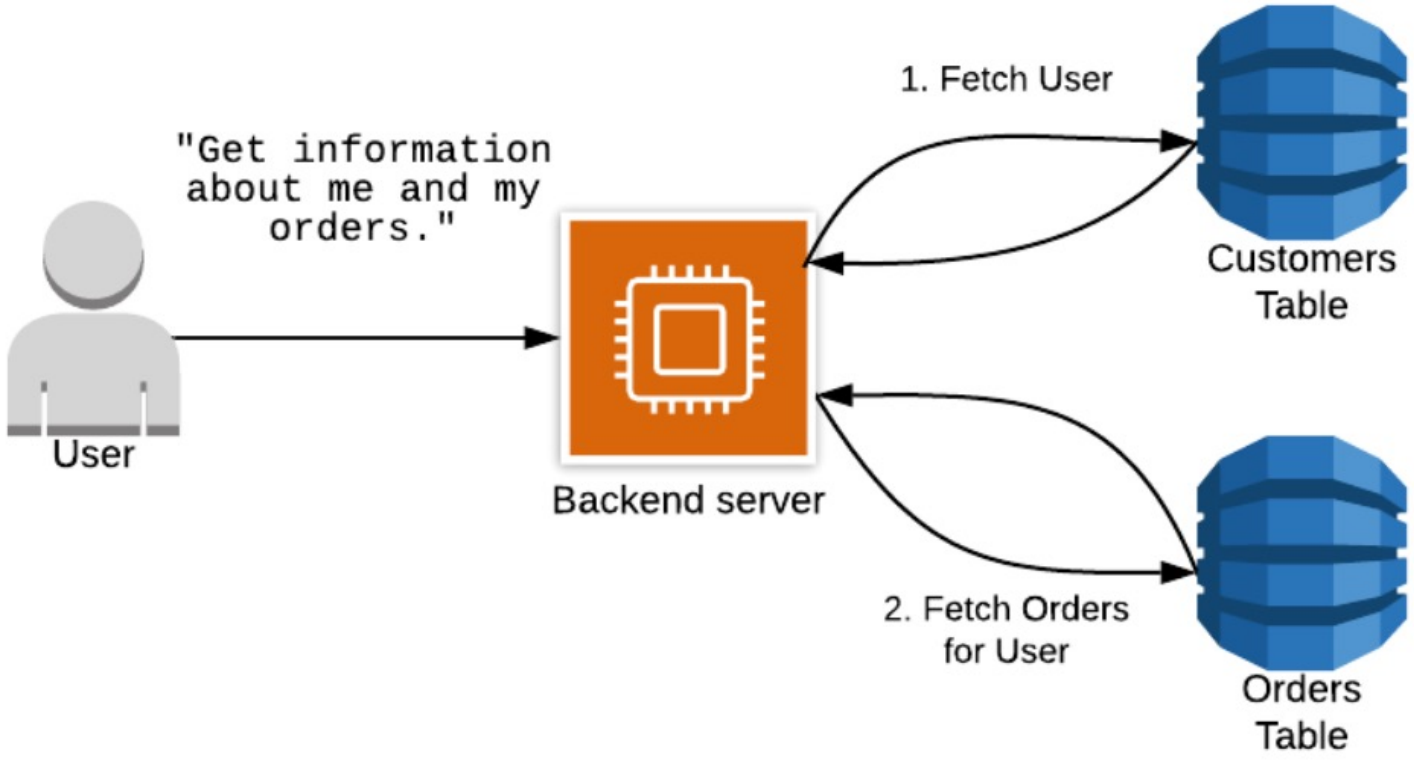


DynamoDB Keys and Attributes

RDBMS way

Customers		
CustomerId	CustomerName	CustomerBirthdate
741	Alex DeBrie	05/26/1988
742	Albert Einstein	03/14/1879

Orders		
OrderId	CustomerId	OrderDate
11578	741	12/20/2019
11579	910	12/21/2019



Single Table Design

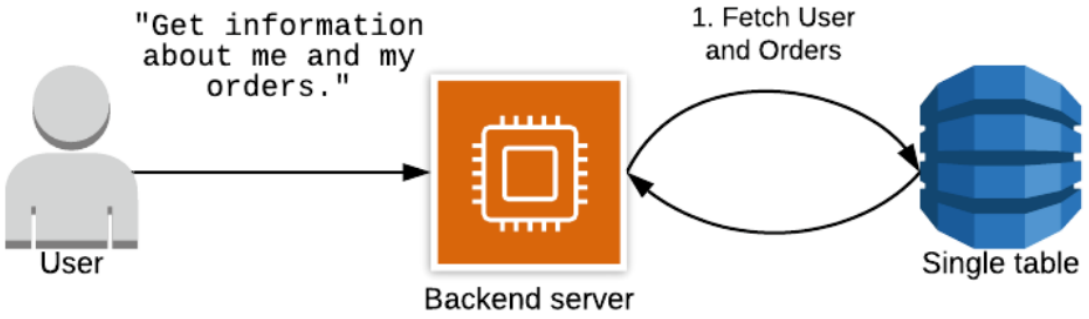
Primary Key		Attributes		
Actor (PARTITION)	**Movie (SORT)**			
Tom Hanks	Cast Away	**Role**	**Year**	**Genre**
		Chuck Noland	2000	Drama
	Toy Story	**Role**	**Year**	**Genre**
		Woody	1995	Children's
Tim Allen	Toy Story	**Role**	**Year**	**Genre**
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	**Role**	**Year**	**Genre**
		Nina Sayers	2010	Drama

Single Table Design

Primary Key		Attributes				
PK	SK					
USER#alexdebrie	#PROFILE#alexdebrie	**Username**	**FullName**	**Email**	**CreatedAt**	**Addresses**
		alexdebrie	Alex DeBrie	alexdebrie1@gmail.com	03/23/2018	{"Home":{"StreetAddress":"1111 1st St","State":"Nebr
	ORDER#5e7272b7	**Username**	**OrderId**	**Status**	**CreatedAt**	**Address**
		alexdebrie	5e7272b7	PLACED	04/21/2019	{"StreetAddress":"1111 1st St","State":"Nebraska","Co
	ORDER#42ef295e	**Username**	**OrderId**	**Status**	**CreatedAt**	**Address**
		alexdebrie	42ef295e	PLACED	04/25/2019	{"StreetAddress":"1111 1st St","State":"Nebraska","Co
	ORDER#2e7abecc	**Username**	**OrderId**	**Status**	**CreatedAt**	**Address**
		alexdebrie	2e7abecc	SHIPPED	12/25/2018	{"StreetAddress":"1111 1st St","State":"Nebraska","Co
USER#nedstark	#PROFILE#nedstark	**Username**	**FullName**	**Email**	**CreatedAt**	**Addresses**
		nedstark	Eddard Stark	lord@winterfell.com	02/27/2016	{"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir
	ORDER#2eae1dee	**Username**	**OrderId**	**Status**	**CreatedAt**	**Address**
		nedstark	2eae1dee	SHIPPED	01/15/2019	{"StreetAddress":"Suite 200, Red Keep","City":"King's L
	ORDER#f4f80a91	**Username**	**OrderId**	**Status**	**CreatedAt**	**Address**
		nedstark	f4f80a91	PLACED	05/12/2019	{"StreetAddress":"Suite 200, Red Keep","City":"King's L

DynamoDB Way

- Reasons for using single table
  - To retrieve multiple, heterogeneous items
  - In a single requests



Limitations of Key Store DB

- Good for only OLTP
- Bad for OLAP
- Only key can be queried
  - All values are obtained together
- General SQL type queries can not be performed
- Work around the Availability and Partition
- Lacks in Consistency

Key features of Key-Value Store [in DynamoDB]

- Scalable
- Flexible
- Distributed horizontally
- Efficient Indexing
- Strong consistency, Atomic counters
- Secure
- Resource consumption monitoring
- MapReduce Integration – with Amazon Elastic MapReduce

Column Family Store / Wide Column Store

Column Family Store / Wide Column Store

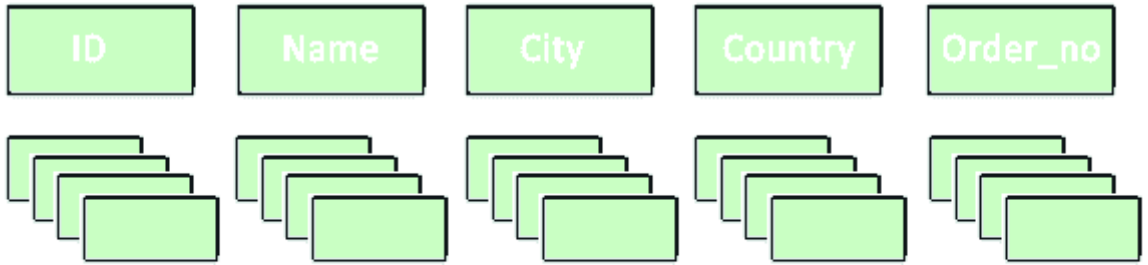
- Instead of storing data in rows,
- Stores data tables as section of column of data
- A relational database is optimized for storing rows of data, typically for transactional applications
- Important factor in analytic query performance
- Drastically reduces the overall disk I/O requirements and reduces the amount of data you need to load from disk.

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

Column Family Store / Wide Column Store

- **<u>Column Stores</u>**
- Not like relational database
- Multi-dimensional map
- Not all entries are relevant each time (Column families)

- **<u>Example</u>**
- Google BigTable
- Cassandra
- Hbase

RowId	EmpId	Lastname	Firstname	Salary
001	10	Smith	Joe	60000
002	12	Jones	Mary	80000
003	11	Johnson	Cathy	94000
004	22	Jones	Bob	55000

Row-oriented systems

```
001:10,Smith,Joe,60000;
002:12,Jones,Mary,80000;
003:11,Johnson,Cathy,94000;
004:22,Jones,Bob,55000;
```

Column-oriented systems

```
10:001,12:002,11:003,22:004;
Smith:001,Jones:002,Johnson:003,Jones:004;
Joe:001,Mary:002,Cathy:003,Bob:004;
60000:001,80000:002,94000:003,55000:004;
```

Apache Cassandra

- Keyspaces
- Tables inside
- Primary key as – Partition key
- Ordering Columns

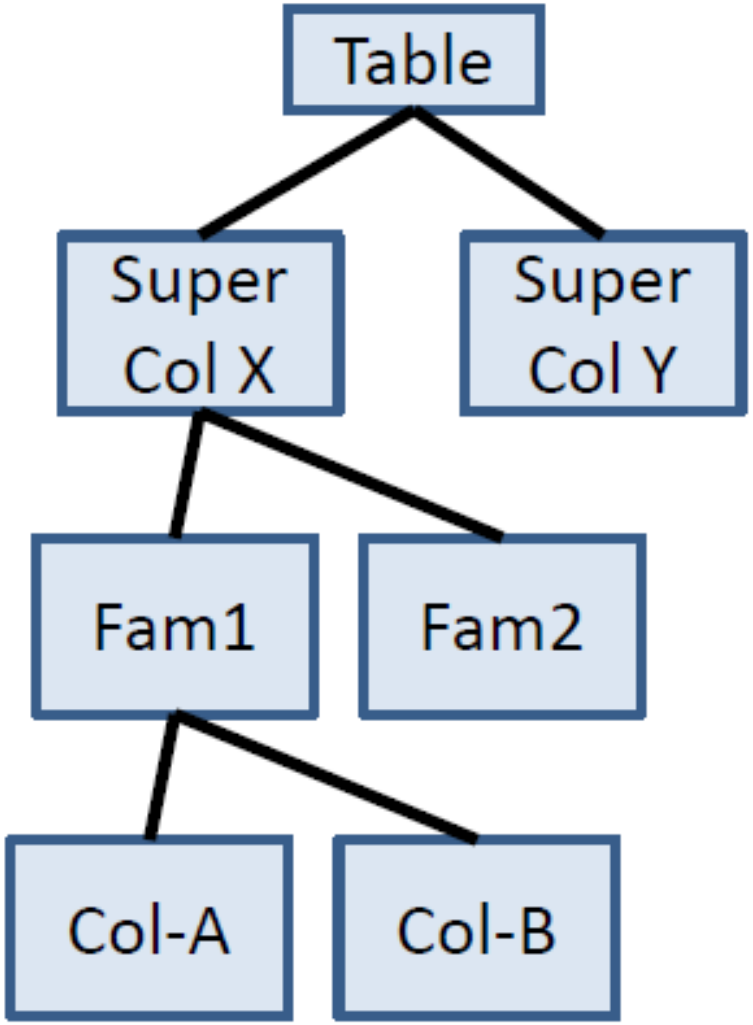
```
CREATE TABLE IF NOT EXISTS employees (
  department_id UUID,
  joiningDate TIMESTAMP,
  id UUID,
  name TEXT,
  position TEXT,
  salary DECIMAL,
  PRIMARY KEY (department_id, joiningDate, id)
);
```

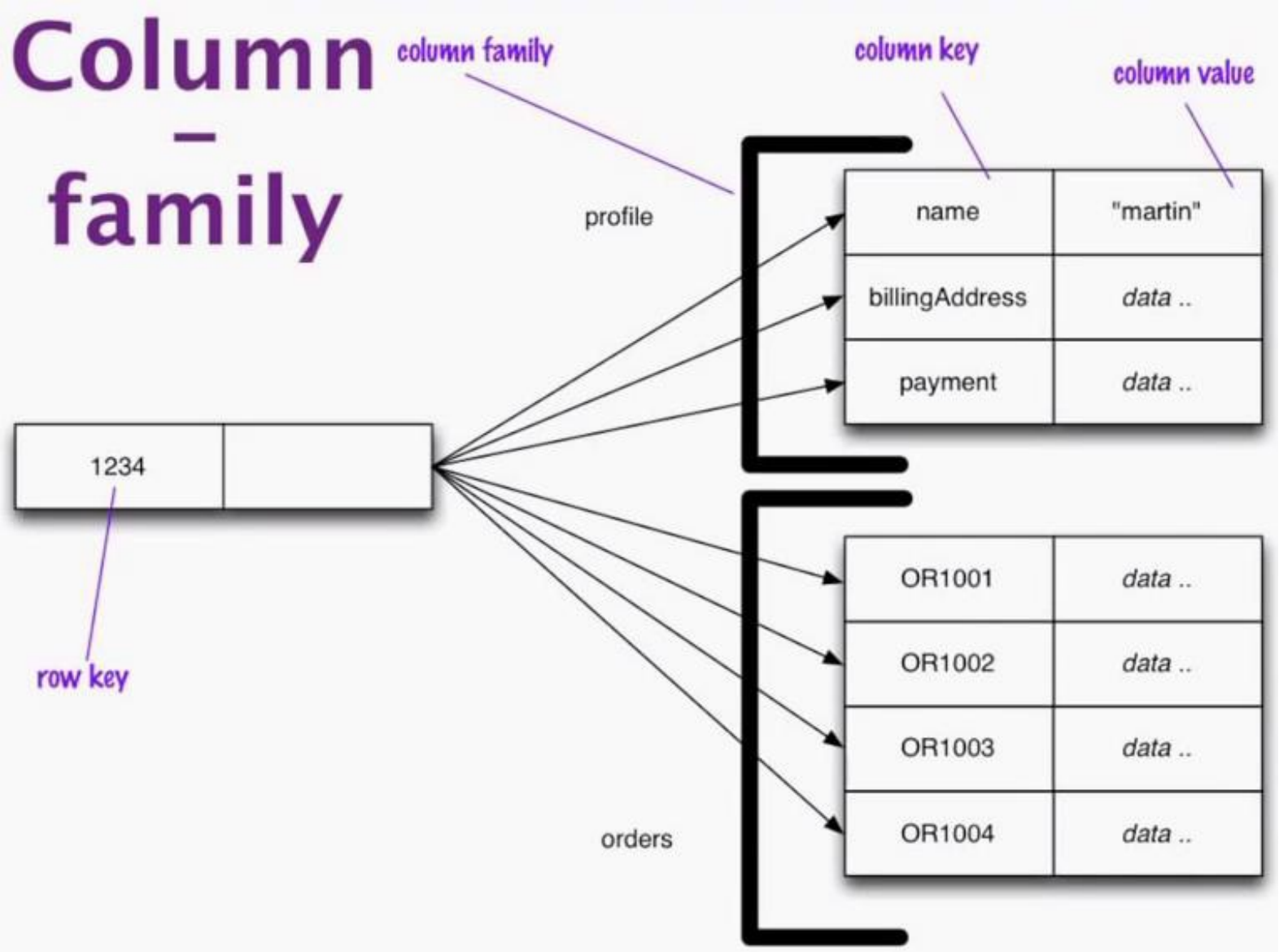
Column Store concepts

- Preserve the table-structure familiar to RDBMS systems
- Not optimized for "joins"
- One row could have millions of columns but the data can be very "sparse"
- Ideal for high-variability data sets
- Colum families allow to query all columns that have a specific property or properties
- Allow new columns to be inserted without doing an "alter table"
- Trigger new columns on inserts

Column Families

- Group columns into "Column families"
- Group column families into "Super-Columns"
- Be able to query all columns with a family or super family
- Similar data grouped together to improve speed




Column
–
family
column family
column key
column value
profile
name
"martin"
billingAddress
data ..
payment
data ..
1234
row key
OR1001
data ..
OR1002
data ..
OR1003
data ..
OR1004
data ..
orders

Disadvantages of Column Family stores

- Updates
  - Updating column family requires accessing different blocks in the disk; whereas for row based approach in single block itself all columns can be updated
  - So Updates slower
- If Multiple attributes queried, then slower

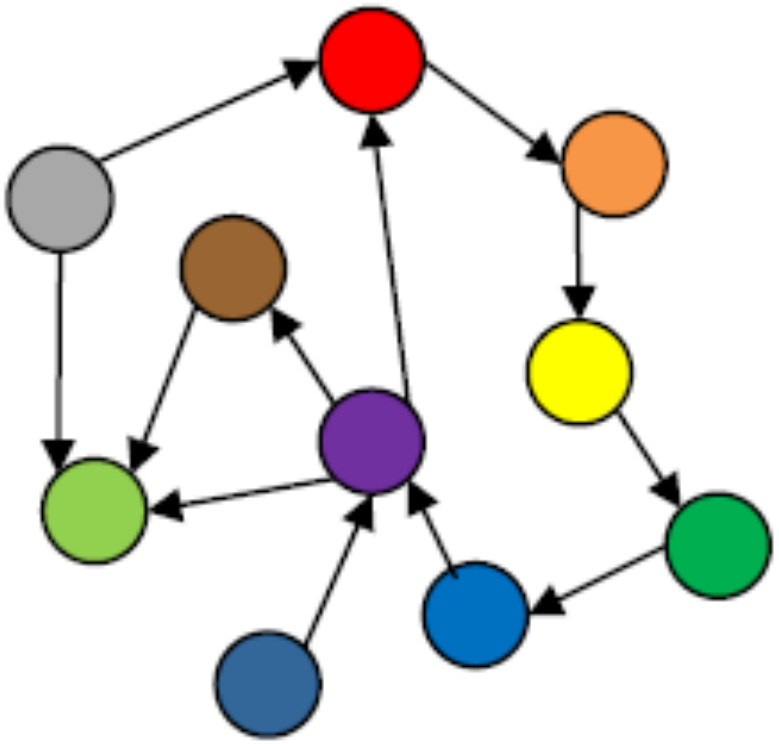
- Columnar: **Good for OLAP, Bad for OLTP**

Document Store

- Mongodb

Graph Store

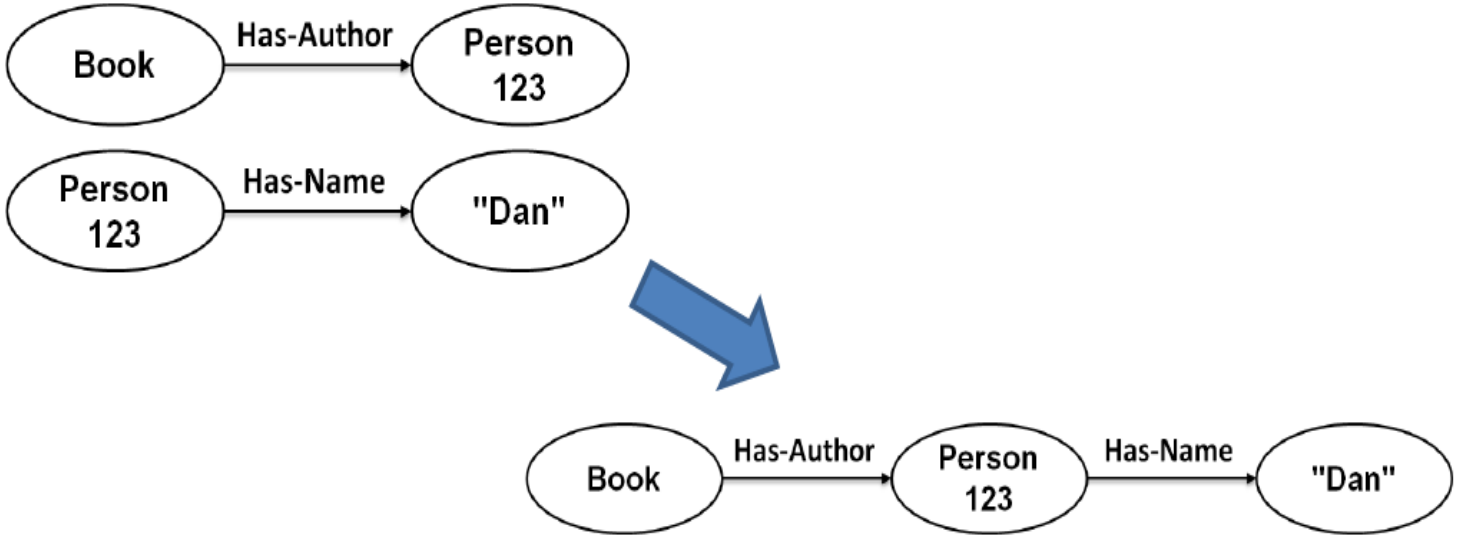
- Ideal when relationships between data is key:
  - e.g. Social Networks
- Used when the relationship and relationships types between items are critical
- Used for Social networking queries: "friends of my friends"



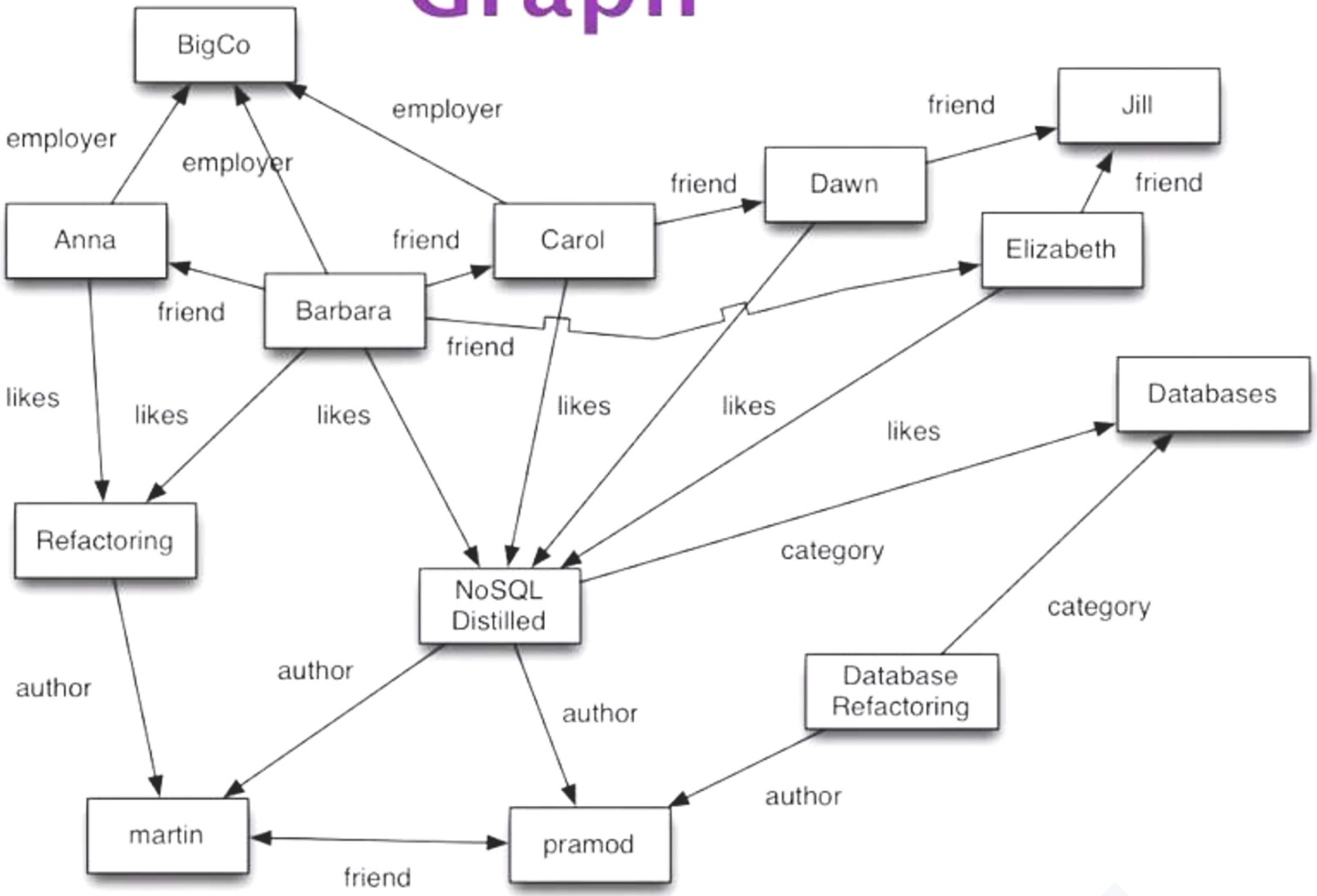
- **Pros**: fast network search, works with public linked data sets
- **Cons**: Poor scalability when graphs don't fit into RAM, specialized query languages

Graph Store

- Nodes are Joined to create graphs



Graph


BigCo
employer
employer
employer
Jill
friend
Dawn
friend
friend
Anna
friend
Carol
Elizabeth
friend
Barbara
friend
likes
likes
likes
likes
likes
likes
Databases
Refactoring
NoSQL
Distilled
category
category
author
author
author
Database
Refactoring
author
martin
pramod
friend

Graph Store – Neo4j

- Neo4j has CQL, Cypher query language much like SQL.
- Supports Neo4j Data Browser is the UI to execute CQL Commands.
- It supports
  - Indexes by using Apache Lucence.
  - UNIQUE constraints.
  - ACID properties of RDBMS.
- It uses Native Graph Processing Engine to store graphs.
- It can export query data to JSON and XLS format.
- It provides REST API to Java, Scala, etc.
- Disk-based (not just RAM)

NoSQL Database