- NoSQL databases create materialized views using a map-reduce computation
- materialised view:
    - Si salvano le query più frequenti, andando a tenere delle viste in cui dobbiamo andare ad aggiornare i dati una volta che li cambiamo, però possiamo andare ad ottenere dele risposte alle query molto efficaci e rapide, in alternativa si possono fare dei cron-job in cui ogni tot si calcolano i valori.
- I dati possono essere distribuiti tramite:
    - **Replica**: stessi dati su server diversi
    - **Partizionamento/sharding**: dati diversi su nodi diversi
- Ci sono diverse tecniche:
    - **singolo server** -> graph db
    - **multiple server sharding** -> si dividono i dati orizzontalmente: parti diverse del dato in server diversi. è importate bilanciare bene il carico tra i server
    - **master-slave** -> si ha un master su cui si scrive, le sue modifiche si propgagano agli slave e quindi le letture si possono fare sul master e sugli slave, di solito si usa un numero di slave dispari per poter bneficiare del meccanismo di voto. se un sistema fa molti inserimenti non conviene un modello del genere.
    - **Peer-to-peer** -> risolve il problema di prima facendo in modo che non vi sia un solo master ma distribuendo il problema tra tutti i nodi. tutti i nodi scrivono e leggono i dati.
- si può combinare sharding e master-slave, avendo più nodi che sono indipendenti, ciascuno con il suo master

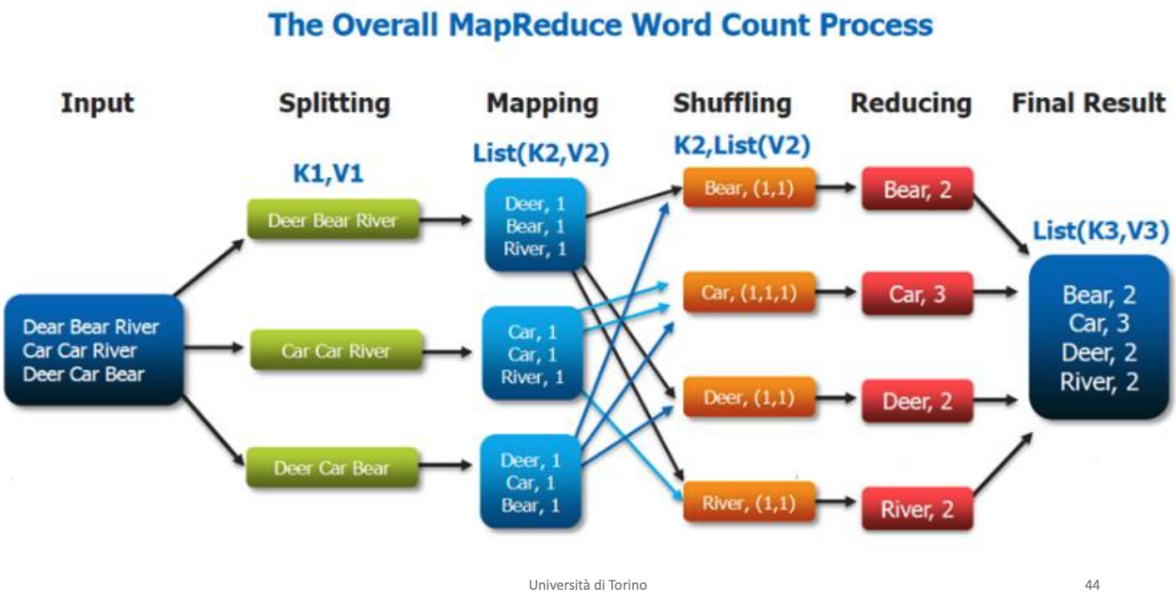
- Il **CAP THEOREM** che si applica alle transazioni in ambiente distribuito. Significa che un database distribuito può garantire **CONSISTENCY** (la distribuzione dei dati è uguale tra tutti i nodi) e **AVAILABILITY** (il dato può essere sia letto che scritto) dopo una **PARTITION** (il dato può sopravvivere anche se il cluster viene separato in più partizioni), ma non contemporaneamente. In una situazione di partizione di rete, deve essere scelto se privilegiare la Consistenza o la Disponibilità.

Types of trade-offs (default)	Databases	Comments
Consistent and available (CA)	Redis, PostgreSQL, Neo4J	They do not distribute the data and they do not have the partitions
Consistent and partition tolerant (CP)	MongoDB, HBase	The hardware faults are managed disabling a partition and allowing the active nodes to answer instead of the missing node
Available and partition tolerant (AP)	CouchDB	It does not guarantee the consistency of data between the two servers

- La consistenza è garantita tramite due tipi di approccio:
  - **Pessimistico**: si usano i lock
  - **Ottimistico**: di guarda il valore prima di fare l'update (approccio usato da git)
- Nella realtà si rilassa il problema della consistenza e si fa in modo che sia tollerato un po' di incosistenza, si possono trovare tramite i **vector stamps** (Supponiamo di avere due nodi, A e B, in un database distribuito con un vettore timestamp vettoriale di lunghezza 2: [0, 0]. Quando un dato viene modificato su A, il vettore timestamp diventa [1, 0] per riflettere la modifica su A. Successivamente, quando la stessa modifica viene replicata su B, il vettore timestamp diventa [1, 1] per indicare l'ordine delle modifiche tra i due nodi).
- L' **eventual consistency** nei database significa che la coerenza dei dati non è garantita istantaneamente dopo un aggiornamento (è il compromesso tra **consistency** e **availability**). Ad esempio, in un

database distribuito, se un dato viene modificato su un nodo, potrebbe passare del tempo prima che la modifica venga riflessa su tutti gli altri nodi. Durante questo periodo, i clienti che accedono ai diversi nodi potrebbero vedere i dati in uno stato non coerente. Solo dopo che la propagazione delle modifiche è completata, i dati diventano coerenti su tutti i nodi. Questo ritardo nella coerenza dei dati rappresenta l'eventual consistency.

- Map Reduce, il framework sviluppato da Google nel 2004, si articola in due fasi fondamentali che consentono di eseguire computazioni parallele su una vasta quantità di dati distribuiti su un cluster di nodi:
  - **Map** : Durante la fase di map, ogni nodo del cluster elabora i dati in ingresso e genera coppie chiave/valore. Ad esempio, consideriamo un'operazione di conteggio delle parole su un insieme di documenti. In questa fase, il framework divide ogni documento in parole e associa a ciascuna parola il valore "1".
  - **Reduce** : Nella fase di reduce, i risultati intermedi prodotti dalla fase di map vengono consolidati e combinati per ottenere l'output finale. Continuando con l'esempio precedente, i valori associati alle stesse chiavi (parole) vengono sommati per ottenere il conteggio totale di ciascuna parola nel set di documenti.
    Usa l'approccio funzionale in cui i risultati vengono poi mandati nuovamente all'application server centrale, evita di mandare troppi dati sulla rete, andando a fare la computazione sul nodo (è simile al **federated learning**, che ha sviluppato sempre google per l'AI però).



Nella map-reduce le operazioni di map possono essere inserite dentro una pipeline in cui l'output di una è l'input della successiva.

L'utilizzo di più tecnologie per lo storage è detta "**Polyglot Persistance**"

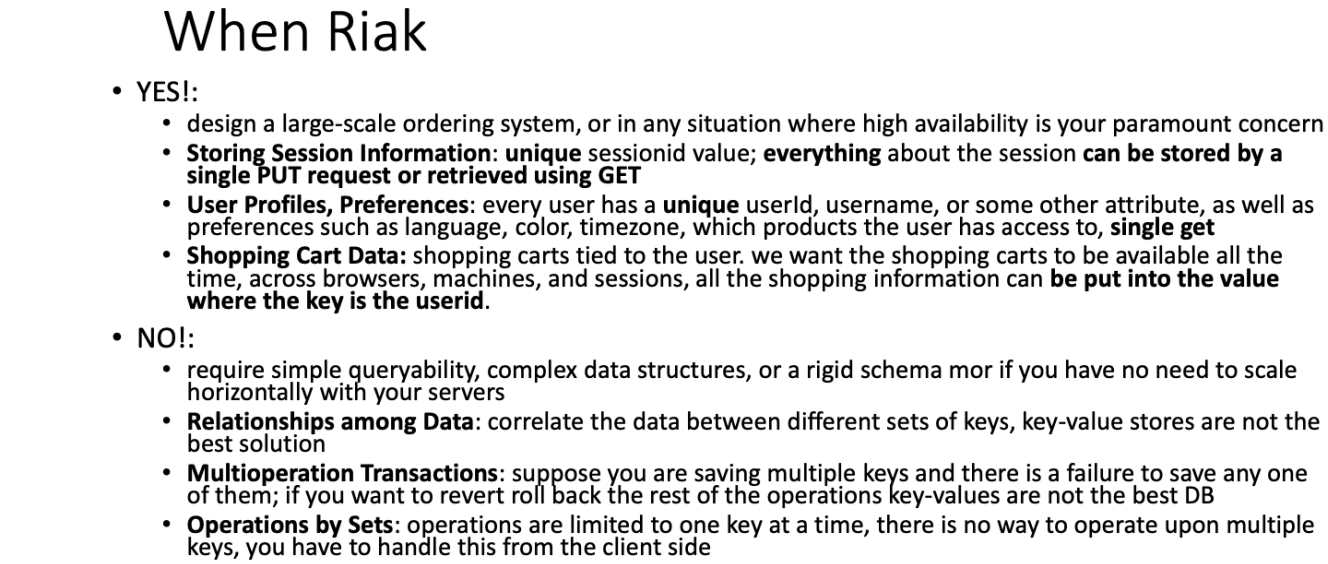
Database type	Needs, goal, requirement	Examples
Relational	Query flexibility (SQL); easy retrieval of data with join and views (set theory); Data have a predefined type	Oracle, PostgreSQL, Mysql
Key - value	Map each key with a value in a similar way as with a programming map or with a hash table with a possible iteration on the keys; caching in main memory of objects for dynamic web applications	Memcached (memcachedb, membase) Voldemort, Redis, Riak, Amazon Dynamo
Columnar	Store big data volumes on different servers (nodes); Easy in horizontal scalability (addition of servers and columns); Data sparsity; the relationships between columns have minor importance; Guarantee of consistency	Hbase, Cassandra, Hypertable, Google BigTable
Document-based	A document consists in a field ID associated to values of various type such as hash; can contain nested structures; There exist some restrictions due to the document representation	MongoDB, CouchDB
Graph-based	Data are highly interconnected in a graph, in which both nodes and edges might have a property (key-value); ease in browsing the graph	Neo4J

45

Key-Value

- **RIAK** : usa chiamate REST per comunicare, per gestire lo storage applica una funzione di hashing alla chiave e usa i *vnode* che sono memorizzati su *n* server su cluster diversi. Riak si basa sul teorema CAP rispettando la **consistenza** e la **disponibilità** , **senza garantire la**

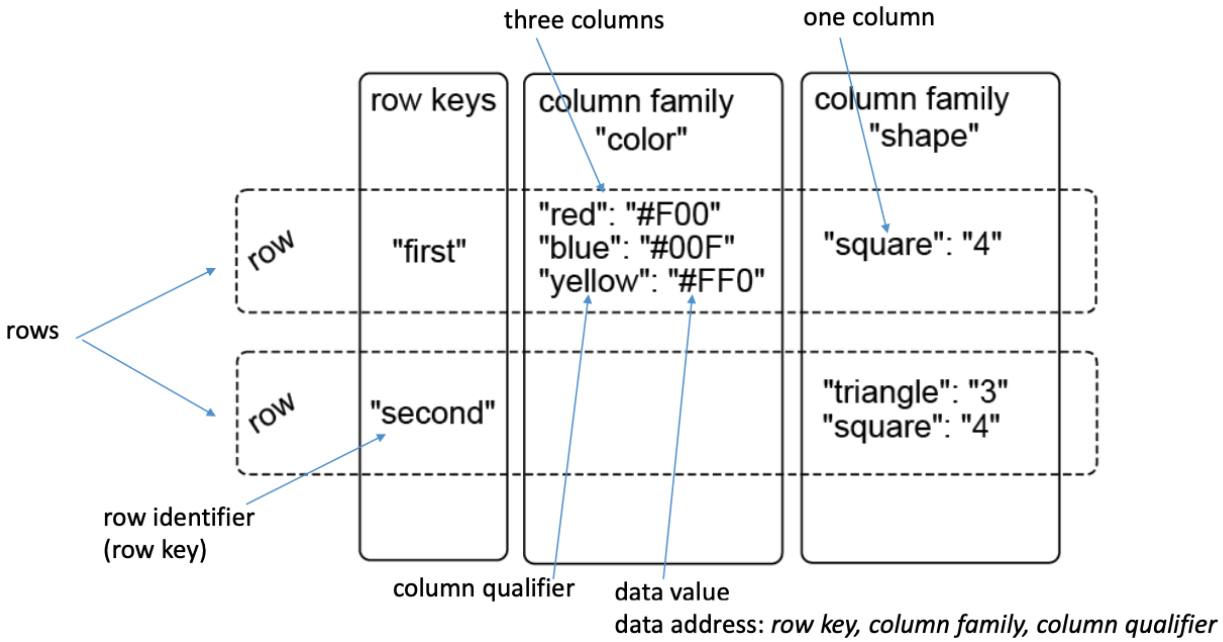
partizione. Utilizza un approccio AP fornendo disponibilità elevata e tolleranza ai guasti in caso di partizioni di rete. È adatto a problemi come sistemi di ordinazione su larga scala (simili ad Amazon), o in qualsiasi situazione in cui sia necessaria alta disponibilità sul web.



- **REDIS** : Un'alternativa a RIAK in cui si può definire anche una expiry policy per non saturare il disco, ha la repica master-slave, è in RAM, quindi è velocissima, però è costosa.

Colonnari

- **H-BASE** : É stato costruito su Hadoop - una piattaforma di calcolo solida e scalabile che fornisce un file system distribuito e capacità di map-reduce. Viene usato da grandi company come Facebook e Ebay. *HBASE è una mappa di mappe*. Viene usato per i big data, in particolare per analizzare grandi moli di dati e farvici sopra analytics. Necessita di almeno 5 nodi quanod viene distribuito.



- HBase includes a robust scale-out architecture and built-in versioning and compression capabilities.

✓It keeps the version history automatically

✓It is meant to scale out

- It does not scale down
- it doesn't offer any sorting or indexing capabilities aside from the row keys.
- The concept of datatypes is missing.

Documentali

- **MONGODB**: Usa una struttura JSON like. Le join non sono efficienti perchè, per natura è un sistema che nasce distribuito. MOnogDB usa una struttura a B-Tree per gestire gli indici. Prevede anche lui l'operazione di Map-reduce. Per quanto riguarda le repliche, mongo ne crea delle copie su server diversi, in modo da resistere nel caso il sistema dovesse andare giù. Il network si bas sul file system distribuito **GridFS**. UNa replica è un insieme di processi che garantiscono **ridondanza e high availability**. Il partizionamento si fa tramite chiave sui documenti. MongoDB utilizza la chiave di shard associata alla

raccolta per suddividere i dati in chunk.

- MongoDB is a document-based database.
- ✓It provides a rich query language and a semi-structured schema
- ✓It distributes objects by sharding and maintains consistency by means of a primary copy and many secondary copies of the objects
- ✓It provides indices

- Joins are not performing well in a distributed environment
- Schemas might be too much denormalised and too much flexible allowing the user to be «lost» in her own data
- The cluster nodes might be a bit difficult to manage

- **COUCHDB** :

  - It has a data model based on JSON and queries based on REST
  - ✓It is highly scalable and might go as well on big cluster of nodes and on installation on smart phones
  - ✓It has views realised with MapReduce
  - ✓It is robust and stable

  - It does not have the same flexibility of MongoDB in queries
  - it does not have the same capability to build replicas and distribute data (with sharding) on nodes

Graph Databases

- **NEO4J** : memorizzano i dati come grafi, usano *groovy* per le procedure di map reduce, usa map-reduce e permette di avere delle transazioni che rispettano la politica ACID. Il linguaggio di query è *gramlin*s che

permette di traversare il grafico per ottenere le informazioni.

- Neo4J is (one of the few) graph databases
✓Graph databases are perfect for unstructured data
✓It is typeless and schemaless, and it puts no constraints on how data is related
✓It is fast (graph traversals are constant time)
✓supports large graphs (more than 34.4 billion of nodes and relationships)
✓provides several tools for fast lookups with Lucene
✓provides easy-to-use (although cryptic) language like Gremlin and the REST interface
- Edges cannot direct a vertex back on itself
- It cannot currently shard subgraphs
- It is not completely an open source project (in production, you need the license)