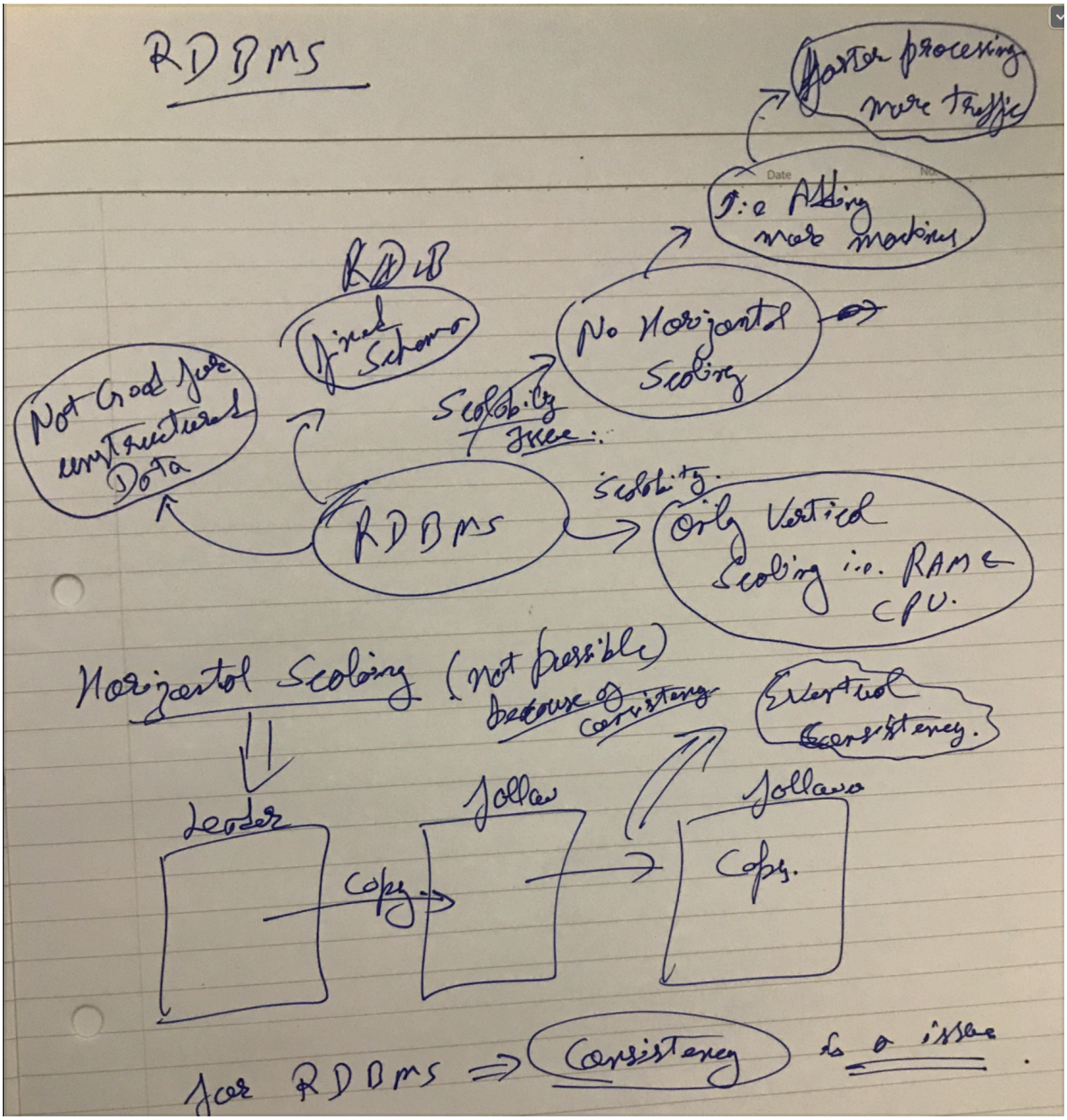
NoSQL_SQL_Mongo_Cassandra_KeyValue_DocumentDB_Columnar

Notes Composed by Akash Tyagi on 14th-April-2021

Link to refer for below Notes:
https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models

RDBMS:

> For instance, it can be difficult to scale a relational database horizontally. *Horizontal scaling*, or *scaling out*, is the practice of adding more machines to an existing stack in order to spread out the load and allow for more traffic and faster processing. This is often contrasted with *vertical scaling* which involves upgrading the hardware of an existing server, usually by adding more RAM or CPU.



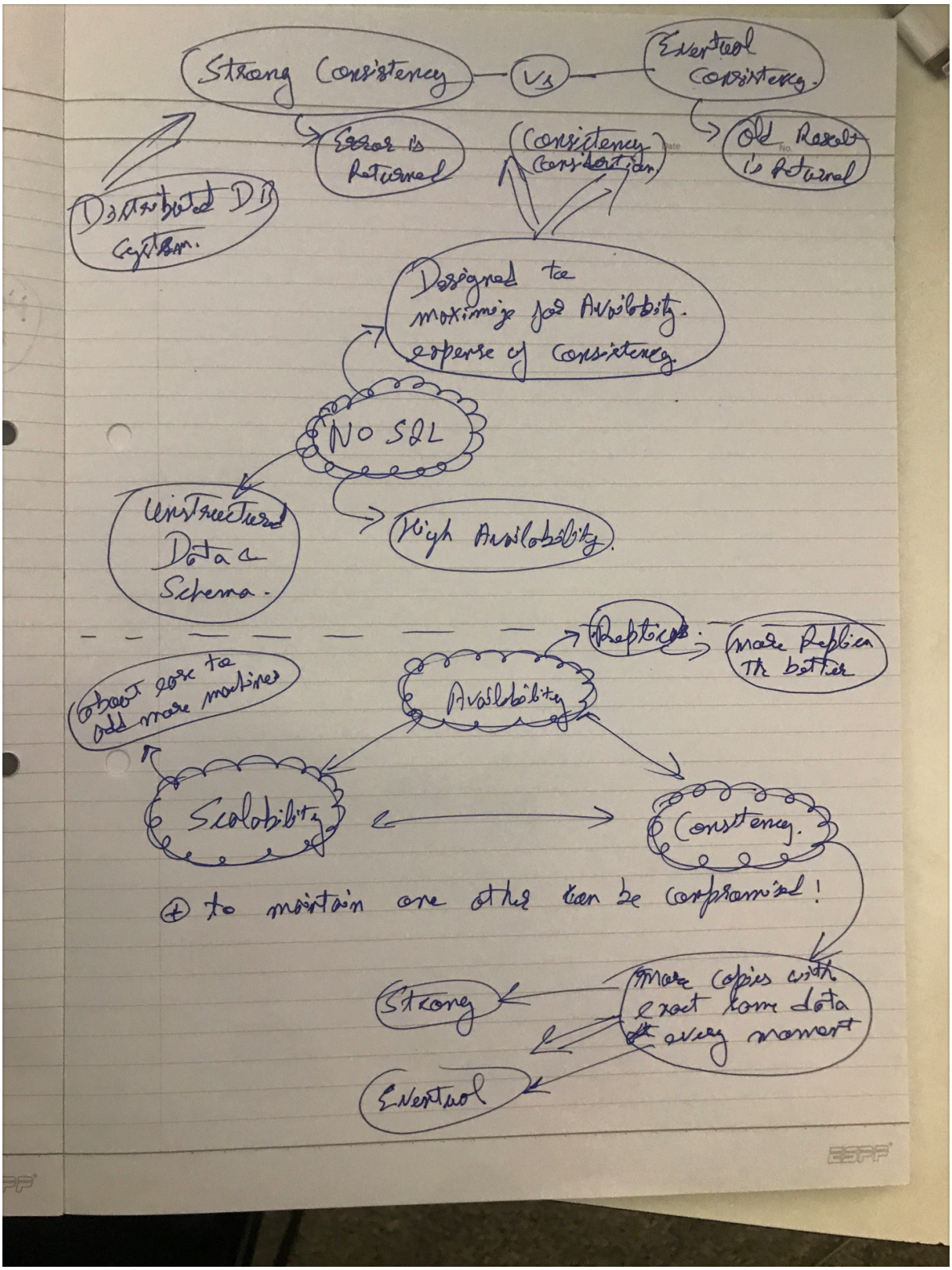
NO SQL

Operational Database Model	Example DBMSs
Key-value store	Redis, MemcacheDB
Columnar database	Cassandra, Apache HBase
Document store	MongoDB, Couchbase
Graph database	OrientDB, Neo4j

1. For one, NoSQL databases are typically designed to maximize

availability at the expense of consistency

Despite these different underlying data models, most NoSQL databases share several characteristics. For one, NoSQL databases are typically designed to maximize availability at the expense of consistency. In this sense, consistency refers to the idea that any read operation will return the most recent data written to the database. In a distributed database designed for strong consistency, any data written to one node will be immediately available on all other nodes; otherwise, an error will occur.



Key Value Store:

Disadvantage:

1. Complex queries are not possible
2. does not have much operations available, only put, update, delete

3. No Schema or structure, everything goes.

Key-value databases are often described as highly performant, efficient, and scalable. Common use cases for key-value databases are caching, message queuing, and session management.

Database	Description
Redis	An in-memory data store used as a database, cache, or message broker, Redis supports a variety of data structures, ranging from strings to bitmaps, streams, and spatial indexes.
Memcached	A general-purpose memory object caching system frequently used to speed up data-driven websites and applications by caching data and objects in memory.
Riak	A distributed key-value database with advanced local and multi-cluster replication.

Scalability and Reliability

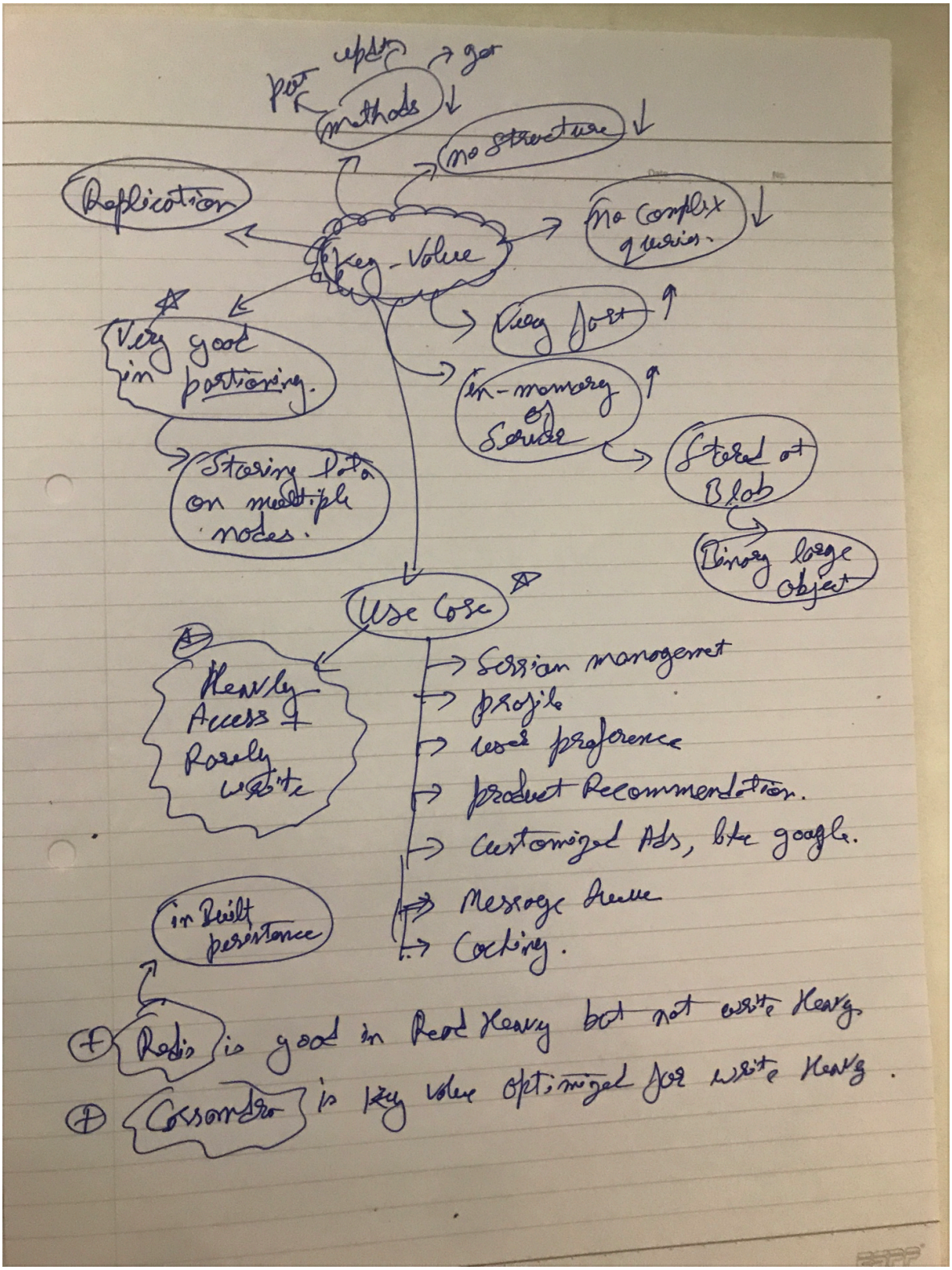
Key-value stores scale out by implementing partitioning (storing data on more than one node), replication and auto recovery. They can scale up by maintaining the database in RAM and minimize the effects of ACID guarantees (a guarantee that committed transactions persist somewhere) by avoiding locks, latches and low-overhead server calls.

Use Cases and Implementations

Key-value stores handle size well and are good at processing a constant stream of read/write operations with low latency making them perfect for:

- Session management at high scale
- User preference and profile stores
- Product recommendations; latest items viewed on a retailer website drive future customer product recommendations
- Ad servicing; customer shopping habits result in customized ads, coupons, etc. for each customer in real-time
- Can effectively work as a cache for heavily accessed but rarely updated data

Key-value stores differ in their implementation where some support ordering of keys like Berkeley DB, FoundationDI and MemcacheDB, some maintain data in memory (RAM) like Redis, some, like Aerospike, are built natively to suppor both RAM and solid state drives (SSDs). Others, like Couchbase Server, store data in RAM but also support rotating


put
update
get
methods
no structure
Replication
Key - Value
No complex queries.
Very good in partitioning.
Very fast
In-memory of Server
Stored at Blob
Binary large object
Storing Data on multiple nodes.
Use Case
Heavily Access & Rarely write
Session management
profile
user preference
product Recommendation.
Customized Ads, like google.
Message queue
Caching.
in Built persistance
Redis is good in Read Heavy but not write Heavy.
Cassandra is key value optimized for write Heavy.

Columnar DB:

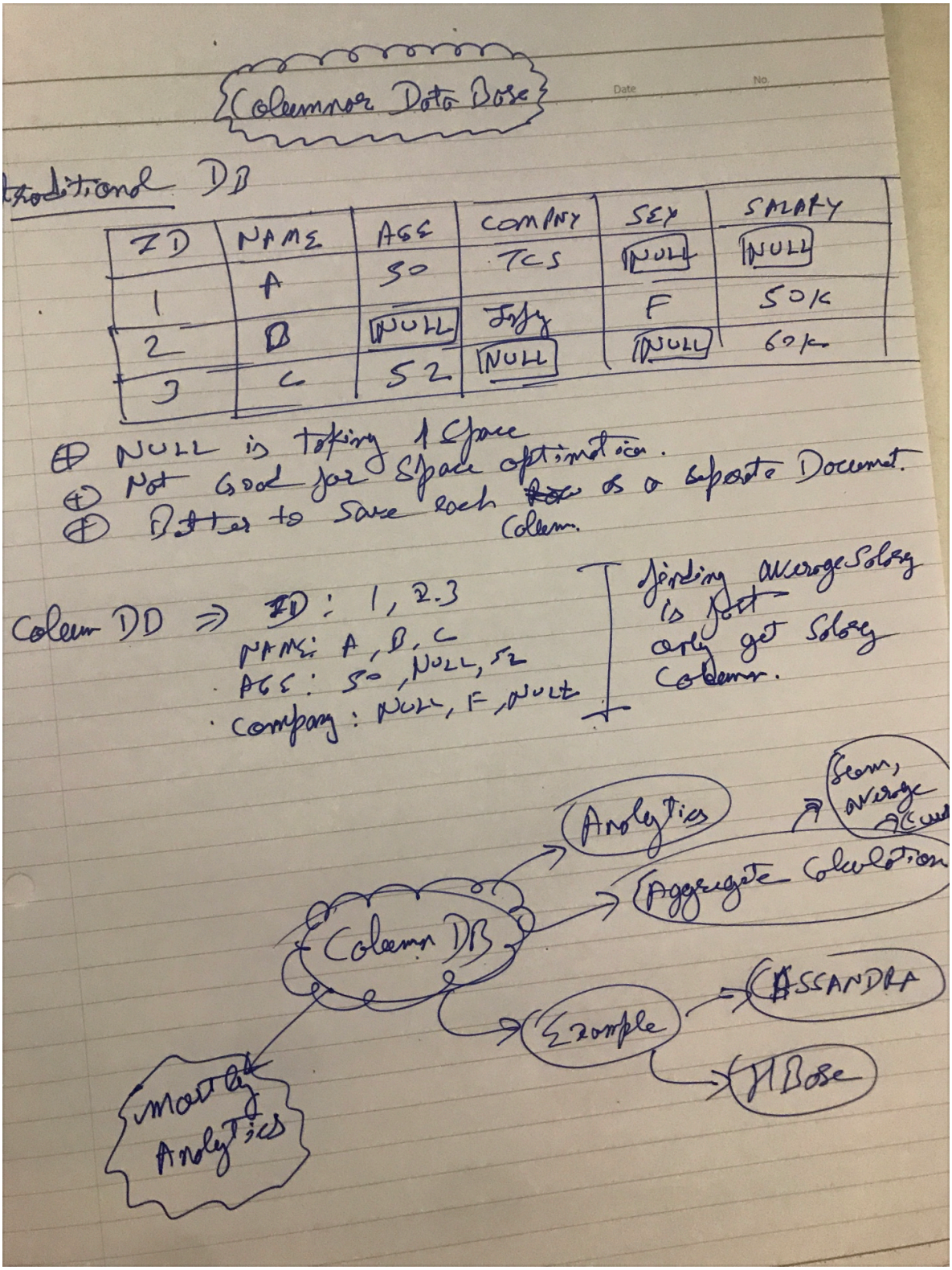
Columnar Databases

Columnar databases, sometimes called *column-oriented databases*, are database systems that store data in columns. This may seem similar to traditional relational databases, but rather than grouping columns together into tables, each column is stored in a separate file or region in the system's storage.

Database	Description
Apache Cassandra	A column store designed to maximize scalability, availability, and performance.
Apache HBase	A distributed database that supports structured storage for large amounts of data and is designed to work with the Hadoop software library.
ClickHouse	A fault tolerant DBMS that supports real time generation of analytical data and SQL queries.

Disadvantage of Columnar DB:

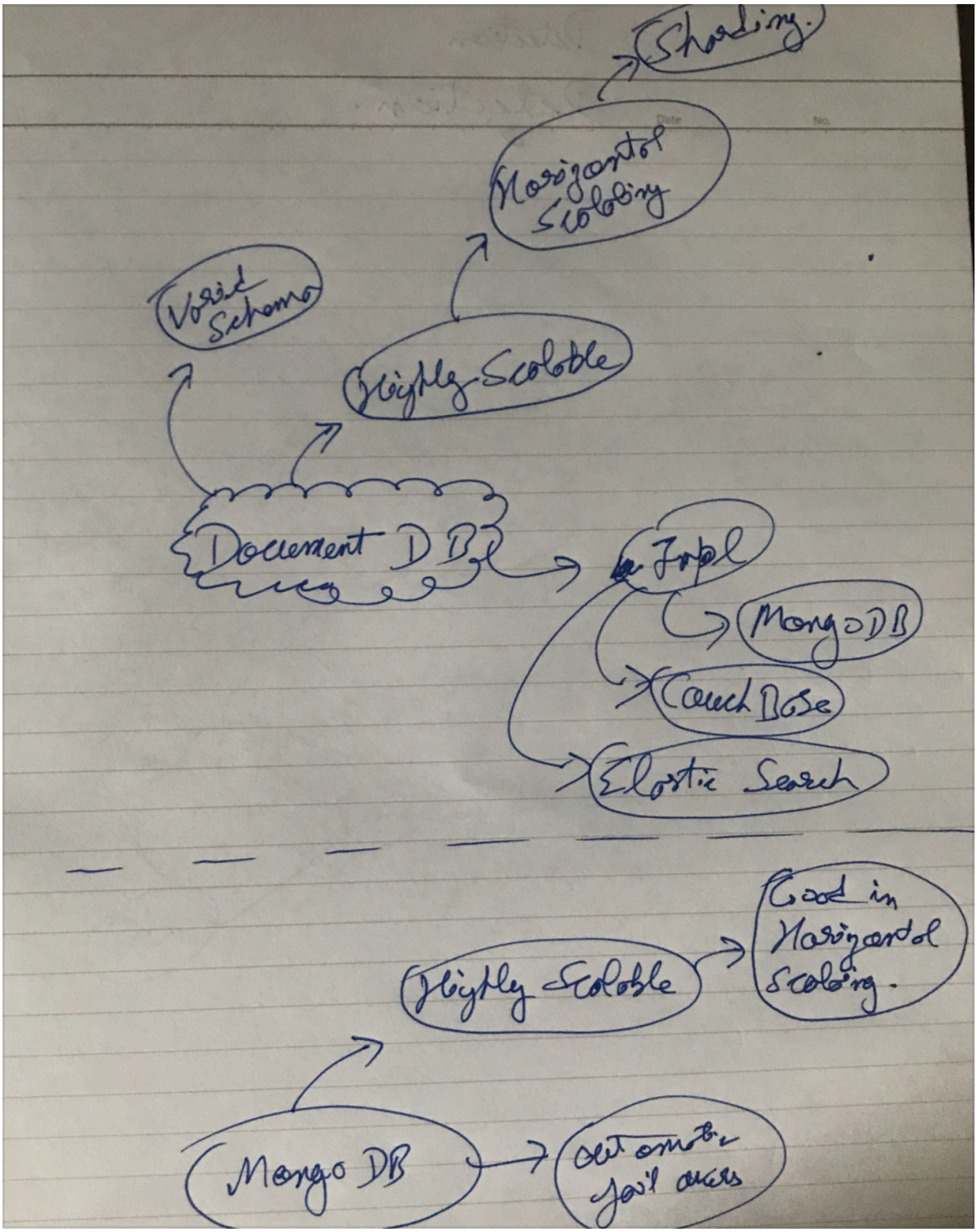
Unfortunately there is no free lunch, which means that while columnar databases are great for analytics, the way in which they write data makes it very difficult for them to be strongly consistent as writes of all the columns require multiple write events on disk. Relational databases don't suffer from this problem as row data is written contiguously to disk.



Document DB:

Document-oriented databases, or *document stores*, are NoSQL databases that store data in the form of documents. Document stores are a type of key-value store: each document has a unique identifier — its key — and the document itself serves as the value.

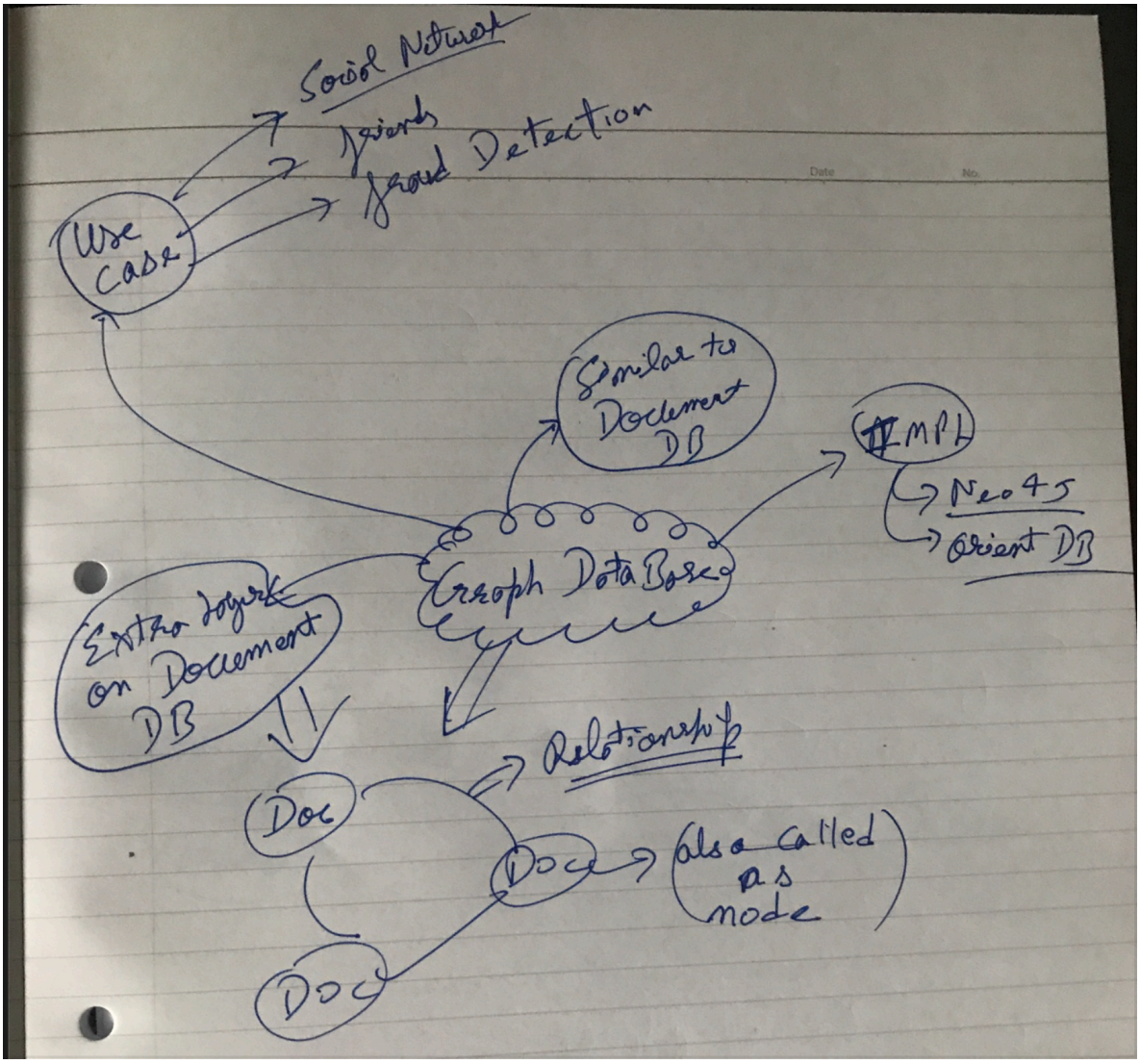
Document-oriented databases have seen an enormous growth in popularity in recent years. Thanks to their flexible schema, they've found regular use in e-commerce, blogging, and analytics platforms, as well as content management systems. Document stores are considered highly scalable, with sharding being a common horizontal scaling strategy. They are also excellent for keeping large amounts of unrelated, complex information that varies in structure.



Graph Database:

Graph databases can be thought of as a subcategory of the document store model, in that they store data in documents and don't insist that data adhere to a predefined schema. The difference, though, is that graph databases add an extra layer to the document model by highlighting the relationships between individual documents.

Certain operations are much simpler to perform using graph databases because of how they link and group related pieces of information. These databases are commonly used in cases where it's important to be able to gain insights from the relationships between data points or in applications where the information available to end users is determined by their connections to others, as in a social network. They've found regular use in fraud detection, recommendation engines, and identity and access management applications.



Cassandra vs HBase. Vs MongoDB

Link: https://logz.io/blog/nosql-database-comparison/
Link: https://www.dnsstuff.com/nosql-database-comparison

	Free Trial?	Bottom Line
Cassandra	Free and Open-Source	Written in Java, Cassandra uses a masterless "ring" architecture, uses CQL rather than SQL, and is known for high durability and scalability.
MongoDB	Free Limited Version	As a popular document store, MongoDB uses JSON-like documents, offers high availability, and can be a cloud or on-premises tool.
Apache HBase	Free and Open-Source	HBase provides support for external APIs, includes automatic recovery for failover support, and uses a master-slave architecture.

Name	Cassandra	MongoDB	HBase
Architecture	Wide Column Store	Document Store	Wide Column Store
Server OS	FreeBSD, Linux, OS X, Windows	Linux, OS X, Solaris, Windows	Linux, Unix, Windows
Distributed System Consistency	Eventual and Immediate Consistency	Eventual and Immediate Consistency	Immediate Consistency
Owner and Developer	Apache Software Foundation	MongoDB, Inc.	Apache Software Foundation
Replication	Masterless Ring	Master-Slave Replication	Master-Slave Replication
Programming Language (Base Code)	Java	C++	Java
Supported Programming Languages	C#, C++, Clojure, Erlang, Go, Haskell, Java, Node.js, Perl, PHP	C, C#, C++, Erlang, Haskell, Java, JavaScript, Perl, PHP	C, C#, C++, Groovy, Java, PHP, Python, Scala

Languages	Node.js, Perl, PHP, Python, Ruby, Scala	JavaScript, Perl, PHP, Python, Ruby, Scala	Scala
Editions	Community with Option of Third-Party Support	Community (Free) and Enterprise	Community
Popular Use Cases	Sensor Data, Messaging Systems, E-commerce Websites, Always-On Applications, Fraud Detection for Banks	Operational Intelligence, Product Data Management, Content Management Systems, IoT, Real-Time Analytics	Online Log Analytics, Hadoop, Write Heavy Applications, MapReduce
DBaaS	InstaClustr Cassandra as a Service, DataStax Database as a Service	MongoDB Atlas,mLab MongoDB, ScaleGrid MongoDB Hosting	None
Key Customers	eBay, McDonald's, Walmart, Comcast, Facebook, Instagram, GitHub, CERN, Netflix,	Adobe, eBay, Google, Cisco, SAP, Facebook, Royal Bank of Scotland, Trend Micro	23andMe, Salesforce, Netflix, Bloomberg, Sophos, Adobe, Xiaomi, Yahoo

Cassandra:

Link: https://logz.io/blog/nosql-database-comparison/

Although Cassandra stores data in columns and rows like a traditional RDBMS, it provides agility in the sense that it allows rows to have different columns, and even allows a change in the format of the columns. Apart from this, its query language, Cassandra Query Language (CQL), closely resembles the traditional SQL syntax, and thus, can be easier for SQL users to understand. This gives it some leverage in any comparison of Cassandra vs. HBase.

Some of Cassandra's most common use cases include messaging systems (for its superior read and write performance), real-time sensor data, and e-commerce websites.

HBase:

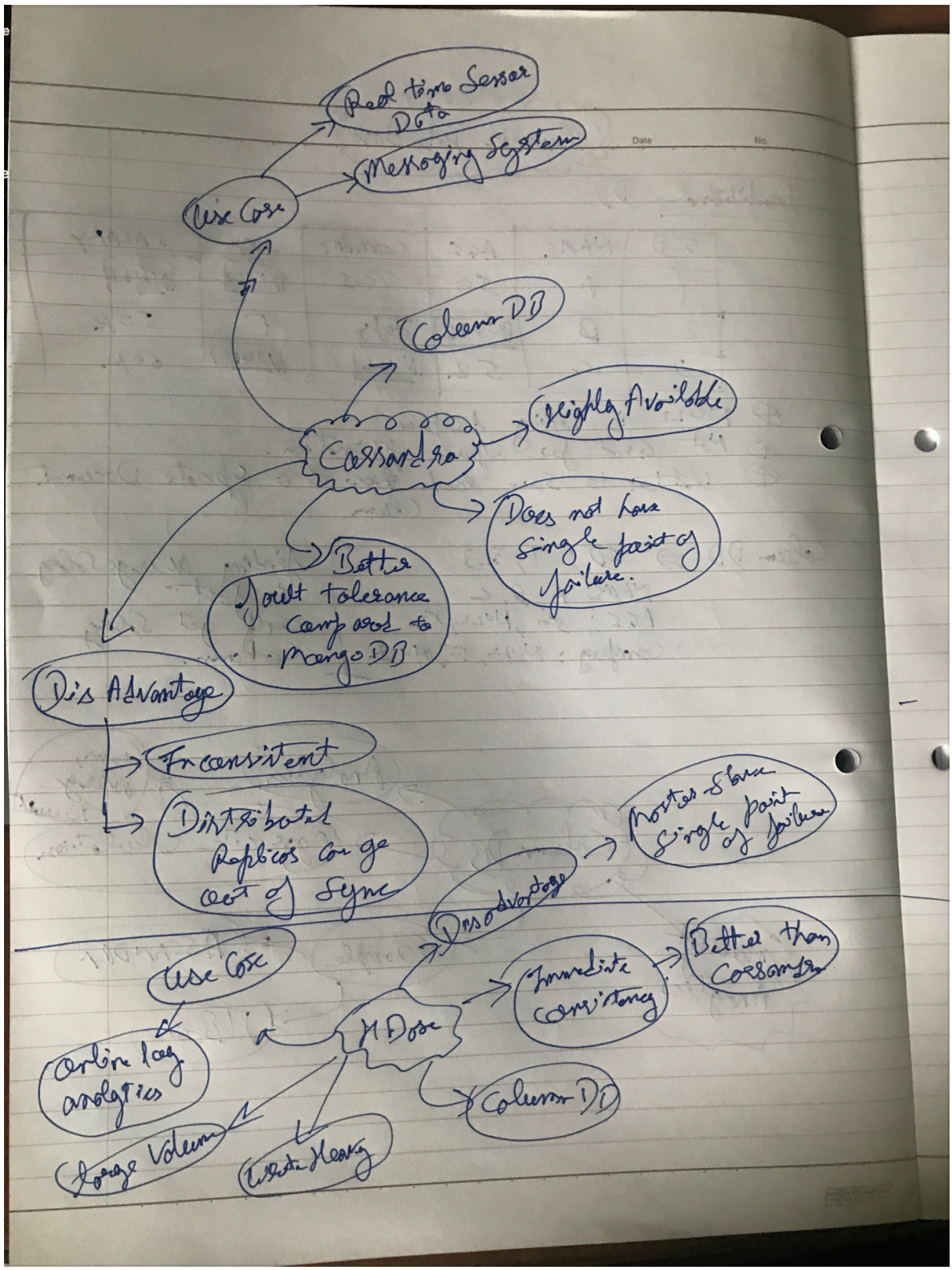
Link: https://logz.io/blog/nosql-database-comparison/

Several concepts from Bigtable, like Bloom filters and block caches, can also be used for query optimization. HBase also has a leg up in any HBase vs. Cassandra comparison when it comes to consistency, as the reads and writes adhere to immediate consistency, compared to the eventual consistency in Cassandra. Its close integration with Hadoop projects and MapReduce

HBase Disadvantages

Although HBase shares several similarities with Cassandra, one major difference in its architecture is the use of a master-slave architecture. This also proves to be a single point of failure, as failing from one HMaster to another can take time, which can also be a performance bottleneck. If you are looking for an always-available system, then Cassandra might be a better choice.

Diagrams:


Real time Sensor Data
Messaging System
Use Case
Column DB
Highly Available
Cassandra
Does not have single point of failure.
Better fault tolerance compared to MongoDB
Dis Advantage
Inconsistent
Distributed Replicas can get out of Sync
Disadvantage
Master Slave single point of failure
Use Case
Online log analytics
Large Volume
Write Heavy
H Base
Immediate consistency
Better than Cassandra
Column DB