NOSQL DATABASES

NoSQL

- NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications. NoSQL databases are widely recognized for their ease of development, functionality, and performance at scale.
- NoSQL databases use a variety of data models for accessing and managing data. These types of databases are optimized specifically for applications that require large data volume, low latency, and flexible data models, which are achieved by relaxing some of the data consistency restrictions of other databases.
- Example:
  - In a relational database, a book record is often dissembled (or “normalized”) and stored in separate tables, and relationships are defined by primary and foreign key constraints.
  - In a NoSQL database, a book record is usually stored as a JSON document. For each book, the item, ISBN, Book Title, Edition Number, Author Name, and AuthorID are stored as attributes in a single document.

Why NoSQL

- As storage costs rapidly decreased, the amount of data applications needed to store and query increased. This data came in all shapes and sizes—structured, semistructured, and polymorphic—and defining the schema in advance became nearly impossible. NoSQL databases allow developers to store huge amounts of unstructured data, giving them a lot of flexibility.
- Cloud computing also rose in popularity, and developers began using public clouds to host their applications and data. They wanted the ability to distribute data across multiple servers and regions to make their applications resilient, to scale-out instead of scale-up, and to intelligent geo-place their data. Some NoSQL databases like MongoDB provided these capabilities.

NoSQL Characteristics

- **Flexibility**: NoSQL databases generally provide flexible schemas that enable faster and more iterative development. The flexible data model makes NoSQL databases ideal for semi-structured and unstructured data.
- **Scalability**: NoSQL databases are generally designed to scale out by using distributed clusters of hardware instead of scaling up by adding expensive and robust servers.
- **High-performance**: NoSQL database are optimized for specific data models and access patterns that enable higher performance than trying to accomplish similar functionality with relational databases.
- **Highly functional**: NoSQL databases provide highly functional APIs and data types that are purpose built for each of their respective data models.

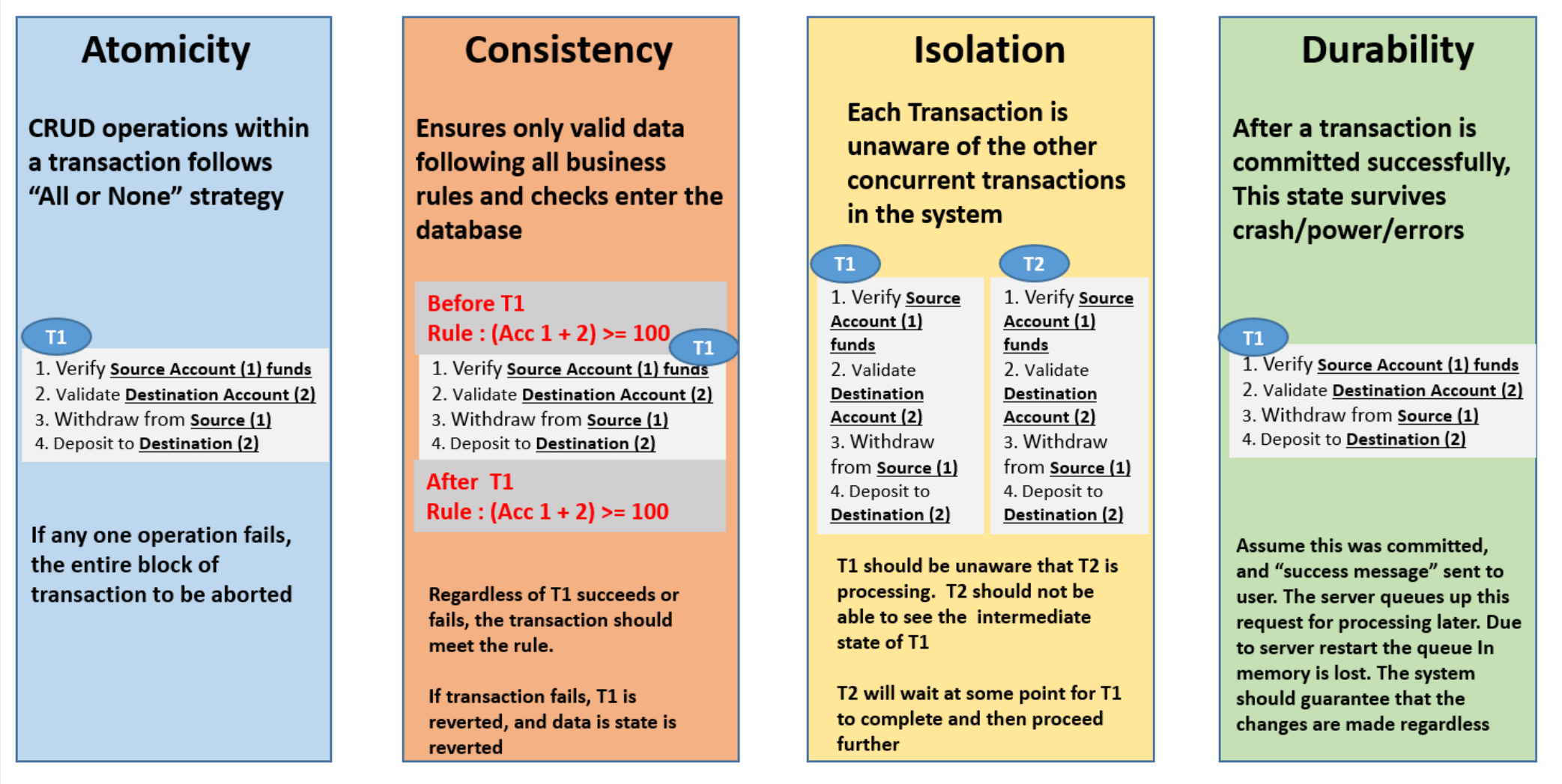
NoSQL Databases Types

- **Document databases:** Store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects.
- **Key-value databases:** A simpler type of database where each item contains keys and values. A value can typically only be retrieved by referencing its value.
- **Wide-column stores:** store data in tables, rows, and dynamic columns. Wide-column stores provide a lot of flexibility over relational databases because each row is not required to have the same columns.
- Graph databases store data in nodes and edges. Nodes typically store information about people, places, and things while edges store information about the relationships between the nodes.

ACID Theorem

- The idea of transactions, their semantics and guarantees, evolved with data management itself. As computers became more powerful, they were tasked with managing more data. Eventually, multiple users shared data on a machine. This led to problems where data could be changed or overwritten while other users were in the middle of a calculation. To solve this problem, they came with the idea of ACID rules. So, what does ACID stand for?
  - ***Atomic:*** *A transaction is fully treated or not at all.*
  - ***Consistent:*** *The content of a database must be coherent at the start and the end of a transaction.*
  - ***Isolated:*** *The modifications of a transaction are visible/modifiable only when it has been modified.*
  - ***Durable:*** *One transaction is committed, it will persist and will be undone to accomodate conflicts with other operations.*

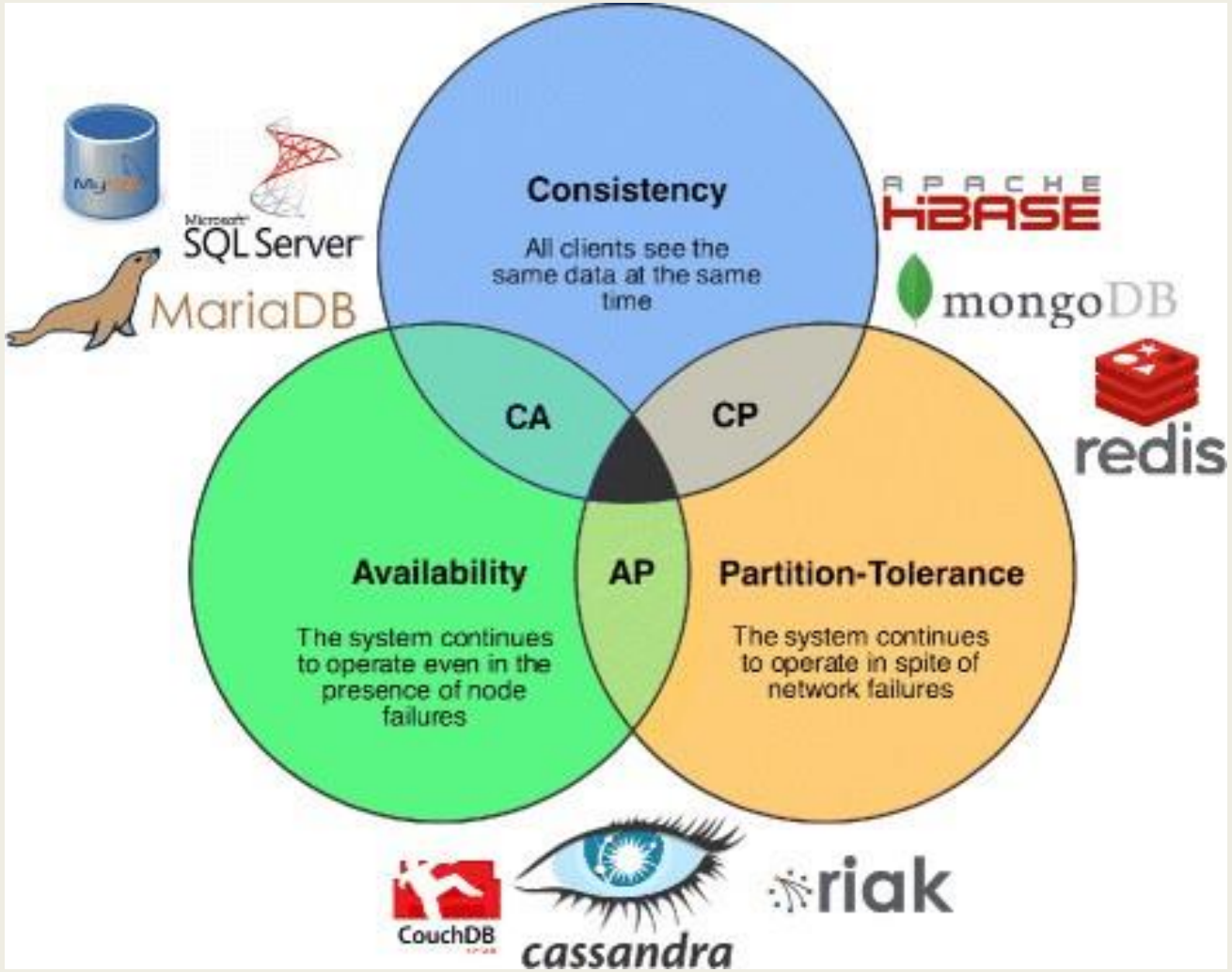
ACID Theorem



CAP Theorem

- ACID transactions offer garantees that absolve the end user of much of the headache of concurrent access to mutable database state. To solve this problem, we tend to use CAP theorem, which stands for:
  - *Consistent: All replicas of the same data will be the same value across a distributed system.*
  - *Available: All live nodes in a distributed system can pocess operations and respond to queries.*
  - *Partition Tolerant: The system is designed to operate in the face of unplanned network connectivity loss between repicas.*

CAP Theorem


Consistency
All clients see the same data at the same time
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
CA
CP
AP
SQL Server
MariaDB
APACHE HBASE
mongoDB
redis
CouchDB
cassandra
riak

NoSQL Advantages

- **Handle Large Volumes of Data at High Speed with a Scale-Out Architecture**
- **Store Unstructured, Semi-Structured, or Structured Data**
- **Enable Easy Updates to Schema and Fields**
- **Developer-Friendly**
- **Take Full Advantage of the Cloud to Deliver Zero Downtime**: provides a clear path to scaling to accommodate huge data sets and high volumes of traffic. Delivering a database using a cluster of computers also allows the database to expand and contract capacity automatically.