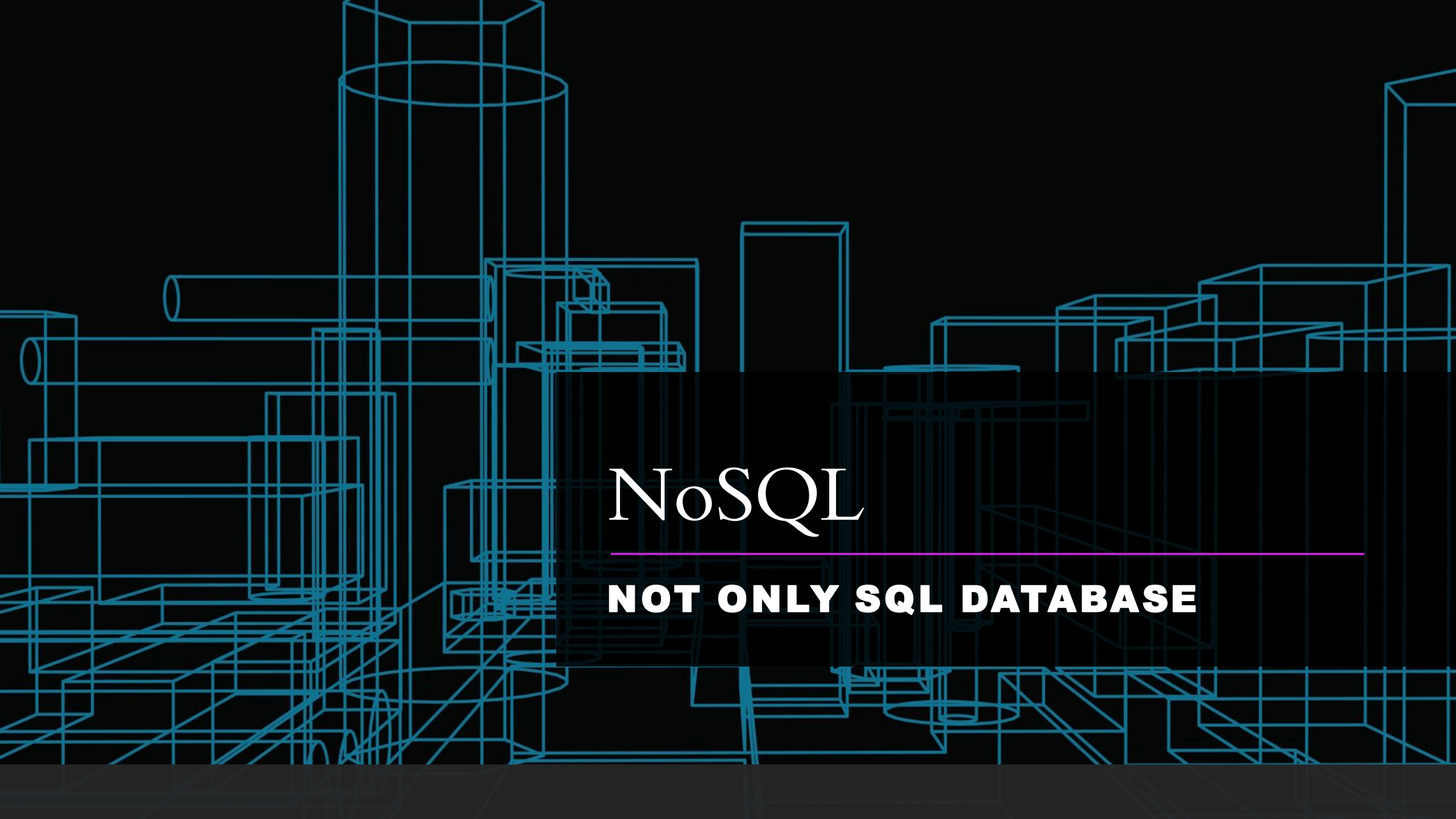
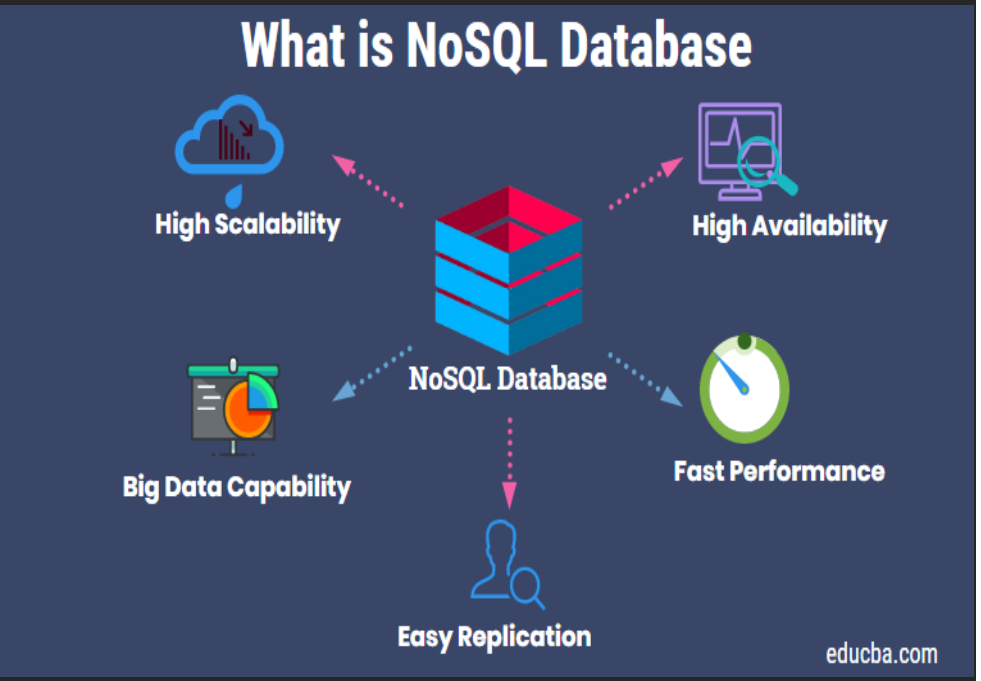


NoSQL

NOT ONLY SQL DATABASE

What is No SQL ?

NoSQL Database is a database implementation method used for stacking, managing and fetching the data from the relational databases that are structured in any model, but not in the typical tabular formatted relationship model. These types of databases are also known as ‘Not only SQL’, ‘non-SQL’ or ‘non-relational’ databases, as it also allows and supports SQL programming languages. The NoSQL databases are progressively applied to big data applications, as well as many other web-based applications.

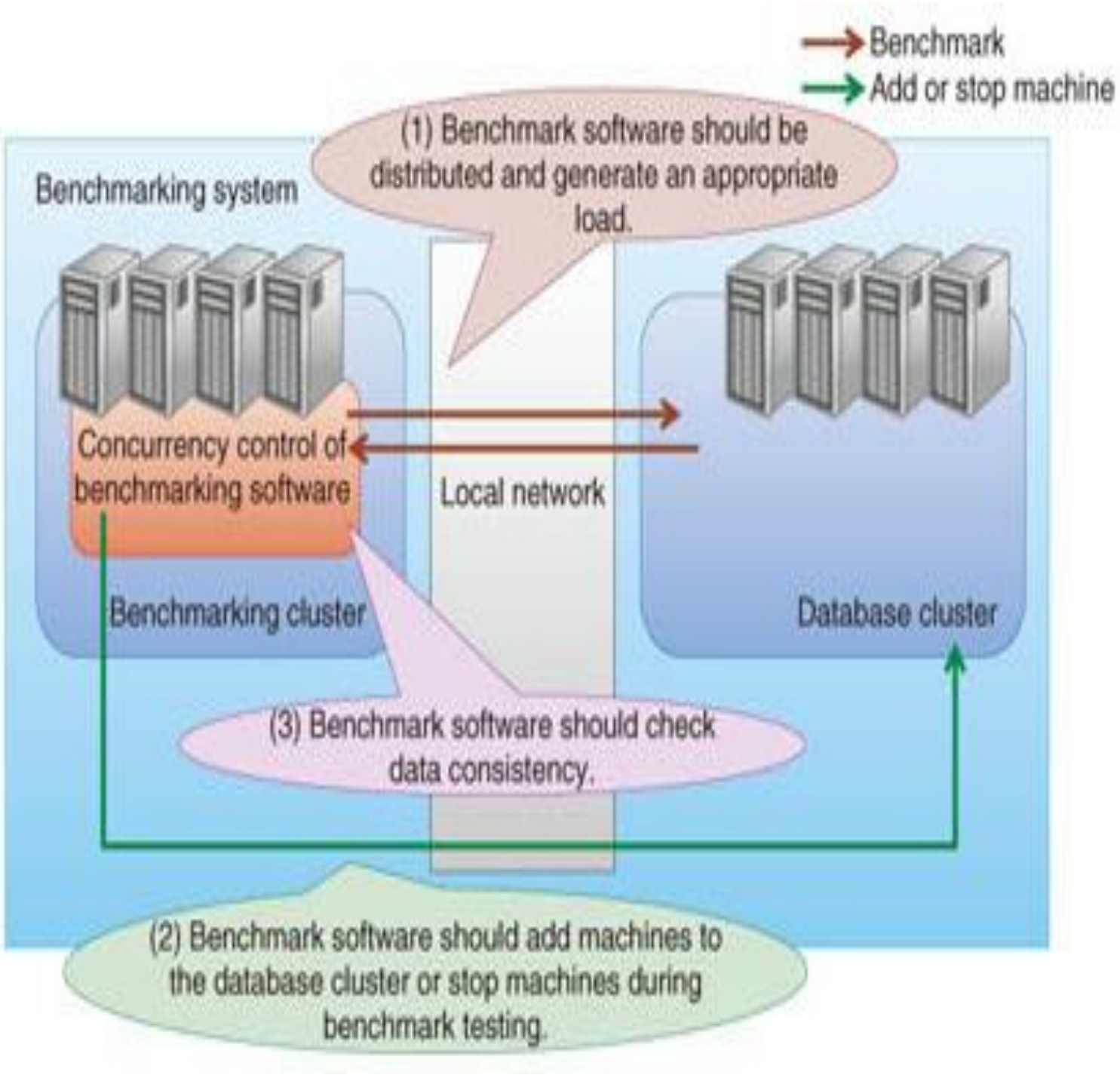


- To gain a competitive edge, businesses need to innovate and adopt agile methodologies. NoSQL is the perfect fit for such needs in the database sphere. They allow you to store data in a flexible and fluid data model, provide scaling up capabilities to your database tier, and provide 100% uptime through replication.

- To top it off, NosDB is a NoSQL Document Database written 100% in .NET. and it comes with algorithms like MapReduce baked right into the database. To assist in migration from an RDBMS, NosDB provides an official ADO.NET wrapper. So if you're a .NET developer and want to try a .NET native NoSQL database, give NosDB a try.

- With a NoSQL database offering so much, the question "Why NoSQL?" should be "Why not NoSQL?"

Characteristics of NoSQL databases



characteristics of NoSQL databases

1. Dynamic schemas

NoSQL databases permit to insert the data without the predefined schema. Real-time application changes can be made easily without the need to worry about service interruptions. This makes development faster, more reliable and less time consuming for the database administrator.

2. Auto-sharding

Horizontal scaling is done in a NoSQL database i.e. servers are added instead of increasing the capacity of a single server. It provide auto-sharding feature i.e. it automatically spread data across various numbers of servers. Application need not be aware of the composition of the server pool. A load of data and query are automatically balanced among the servers. If any server fails, then it is replaced quickly and transparently without disrupting the application.

characteristics of NoSQL databases

. Replication

The database allows automatic database replication.
It is done to maintain availability in case
of outages. Some sophisticated NoSQL databases provide aut
omated recovery and are fully self- healing. To enable
data localization and
to withstand regional failures, it can distribute the database acr
oss multiple geographic regions.
NoSQL does not require a separate application
to implement replication.

4. Integrated Caching

NoSQL databases have integrated caching capability i.e. they
keep frequently used data in system memory
and remove the need for the separate caching layer.

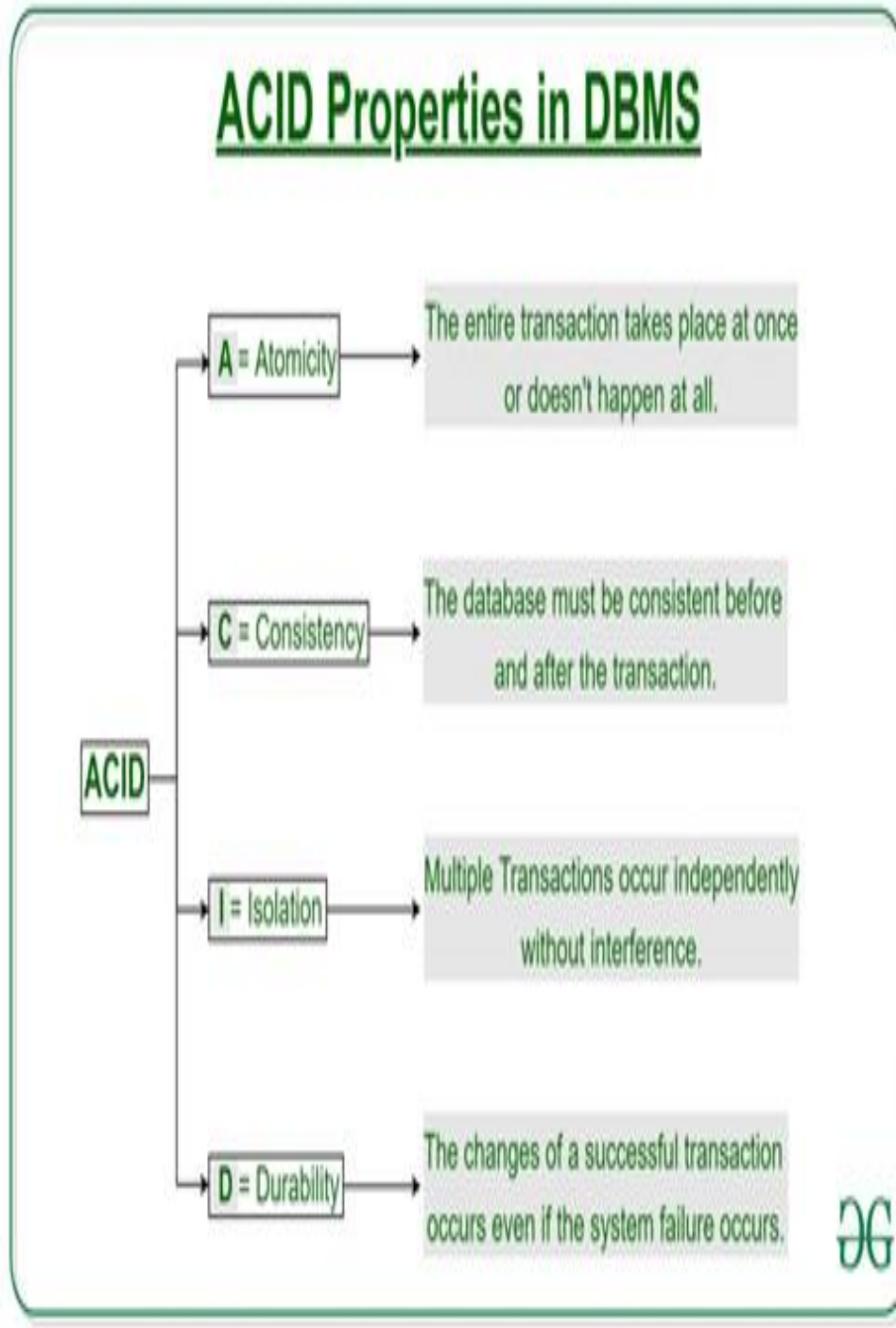
Popular Course in this category

ACID THEOREM

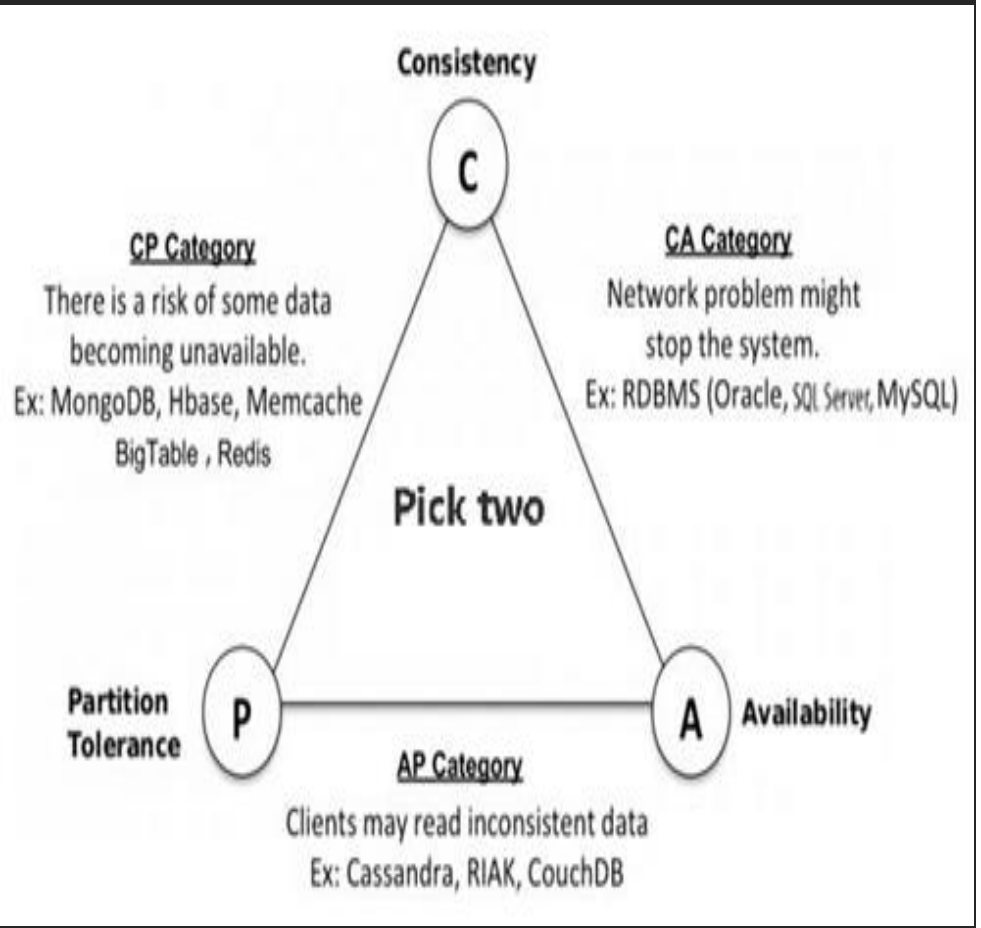
In computer science, ACID (atomicity, consistency, isolation, durability) is a set of properties of database transactions intended to guarantee data validity despite errors, power failures, and other mishaps. In the context of databases, a sequence of database operations that satisfies the ACID properties (which can be perceived as a single logical operation on the data) is called a transaction. For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction.

In 1983, Andreas Reuter and Theo Härder coined the acronym ACID, building on earlier work by Jim Gray who named atomicity, consistency, and durability, but not isolation, when characterizing the transaction concept. These four properties are the major guarantees of the transaction paradigm, which has influenced many aspects of development in database systems.

According to Gray and Reuter, the IBM Information Management System supported ACID transactions as early as 1973 (although the acronym was created later).



CAP theorem



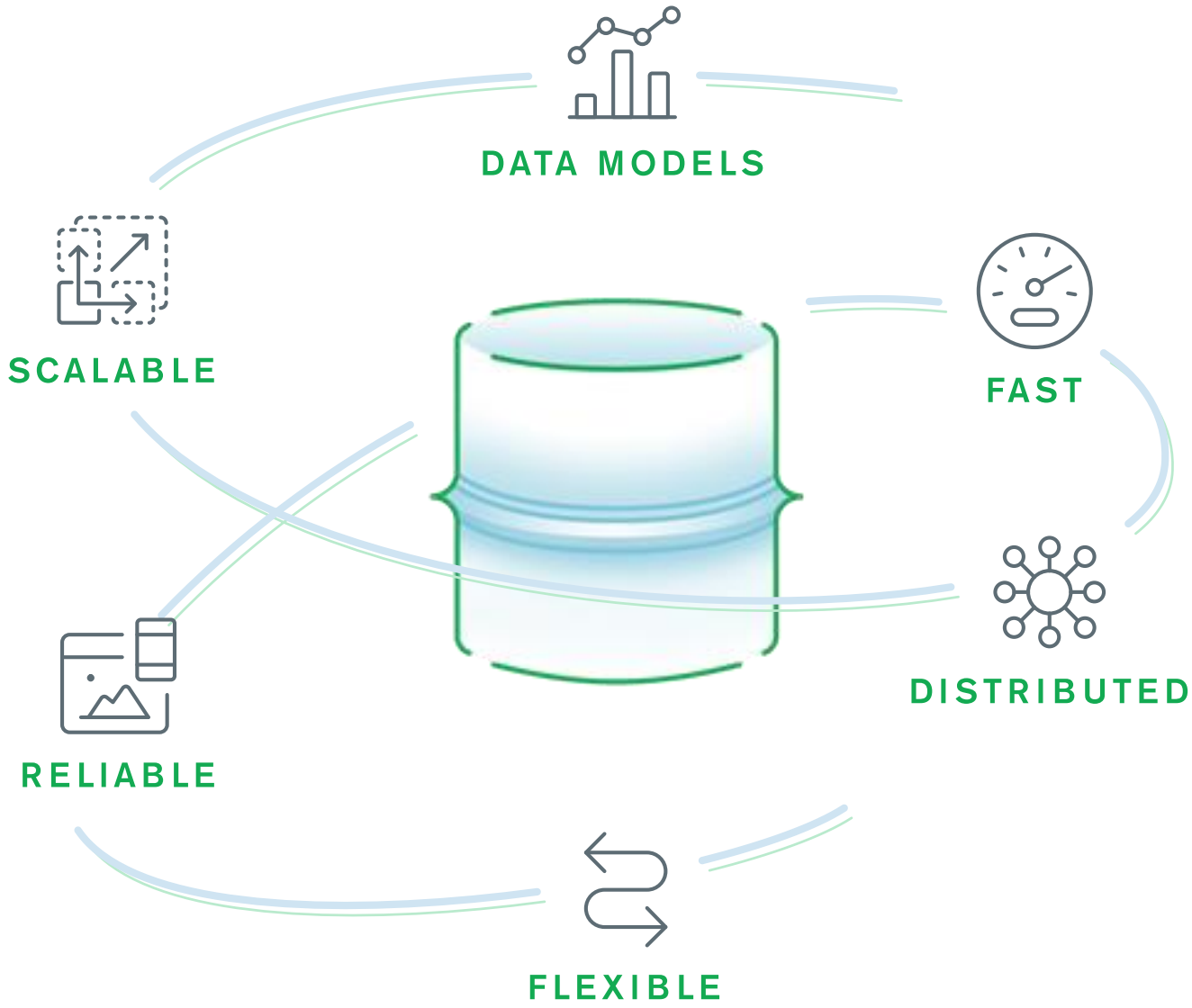
- Have you ever seen an advertisement for a landscaper, house painter, or some other tradesperson that starts with the headline, “Cheap, Fast, and Good: Pick Two”?

- The CAP theorem applies a similar type of logic to distributed systems—namely, that a distributed system can deliver only two of three desired characteristics: consistency, availability, and partition tolerance (the ‘C,’ ‘A’ and ‘P’ in CAP).

- A distributed system is a network that stores data on more than one node (physical or virtual machines) at the same time. Because all cloud applications are distributed systems, it’s essential to understand the CAP theorem when designing a cloud app so that you can choose a data management system that delivers the characteristics your application needs most.

- The CAP theorem is also called Brewer’s Theorem, because it was first advanced by Professor Eric A. Brewer during a talk he gave on distributed computing in 2000. Two years later, MIT professors Seth Gilbert and Nancy Lynch published a proof of “Brewer’s Conjecture.”

The benefits of NoSQL database



Data Models

NoSQL databases often leverage data models more tailored to specific use cases, making them better at supporting those workloads than relational databases. For example, key-value databases support simple queries very efficiently while graph databases are the best for queries that involve identifying complex relationships between separate pieces of data.

Performance

NoSQL databases can often perform better than SQL/relational databases for your use case. For example, if you're using a document database and are storing all the information about an object in the same document (so that it matches the objects in your code), the database only needs to go to one place for those queries. In a SQL database, the same query would likely involve joining multiple tables and records, which can dramatically impact performance while also slowing down how quickly developers write code.

Scalability

SQL/relational databases were originally designed to scale up and although there are ways to get them to scale out, those solutions are often bolt-ons, complicated, expensive to manage, and hard to evolve. Some core SQL functionality also only really works well when everything is on one server. In contrast, NoSQL databases are designed from the ground up to scale-out horizontally, making it much easier to maintain performance as your workload grows beyond the limits of a single server.

Data Distribution

Because NoSQL databases are designed from the ground up as distributed systems, they can more easily support a variety of business requirements. For example, suppose the business needs a globally distributed application that provides excellent performance to users all around the world. NoSQL databases can allow you to deploy a single distributed cluster to support that application and ensure low latency access to data from anywhere. This approach also makes it much easier to comply with data sovereignty mandates required by modern privacy regulations.

Reliability

NoSQL databases ensure high availability and uptime with native replication and built-in failover for self-healing, resilient database clusters. Similar failover systems can be set up for SQL databases but since the functionality is not native to the underlying database, this often means more resources to deploy and maintain a separate clustering layer that then takes longer to identify and recover from underlying systems failures.

Flexibility

NoSQL databases are better at allowing users to test new ideas and update data structures. For example, MongoDB, the leading document database, stores data in flexible, JSON-like documents, meaning fields can vary from document to document and the data structures can be easily changed over time, as application requirements evolve. This is a better fit for modern microservices architectures where developers are continuously integrating and deploying new application functionality.