A MongoDB White Paper

MongoDB Architecture

Dhivya Rajasekhar

Introduction

MongoDB architecture brings together the best features of NOSQL databases with the best features of relational databases.

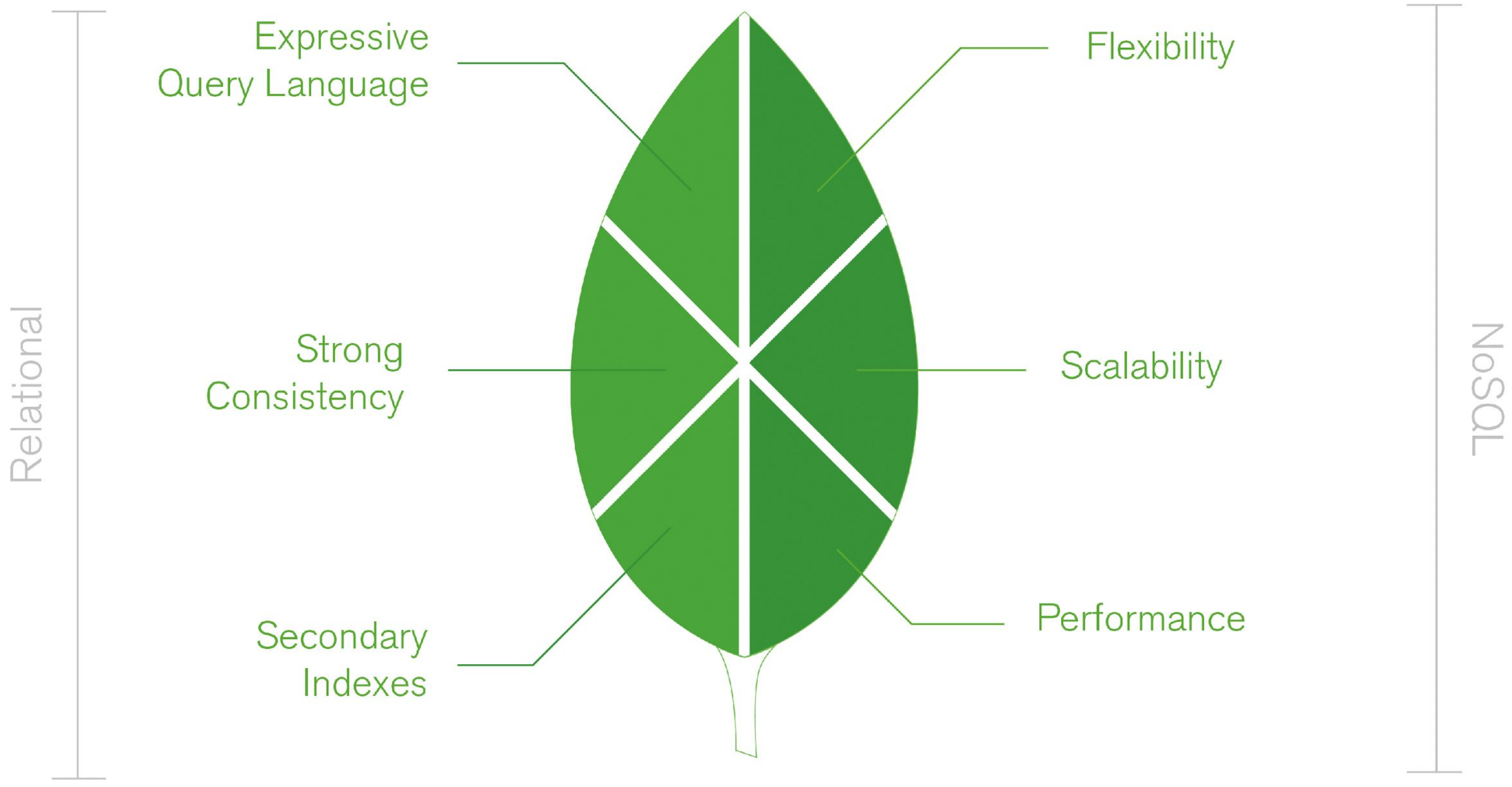
Features from relational databases:

✓ Expressive Query Language
✓ Secondary Indexes
✓ Strong Consistency

Features from NOSQL databases:

✓ Flexible Data Model
✓ Elastic Scalability
✓ High Performance

Features of MongoDB



❖ Expressive Query Language

MongoDB allows its users to use create expressive queries to access or update the data in sophisticated ways.

Various types of queries available in MongoDB are:

- ❖ Key-Value Queries
- ❖ Map-Reduce
- ❖ Text-Search Queries
- ❖ Geospatial Queries
- ❖ Range Queries
- ❖ Aggregation Queries

❖ Secondary Indexes

Indexes play a key role in MongoDB for accessing the data. It is absolutely critical to select the correct indexes to allow for efficient access of data for reading or writing.

Various types of indexes available in MongoDB are:

- ❖ Unique indexes
- ❖ Compound indexes
- ❖ Array indexes
- ❖ TTL indexes
- ❖ Geospatial indexes
- ❖ Text search indexes
- ❖ Sparse indexes

❖ Strong Consistency

It is essential for any database to maintain consistency of data so that a user may read the data as soon as it is written to the database. Although this sounds like a basic feature, it is extremely complicated for a developer to build applications based on a consistent model.

In MongoDB, strong consistency is achieved with the help of the following :

- Object level atomic updates
- ACID transactions
- Distributed transactions
- Durable Writes

❖ Flexible Data Model

MongoDB allows for flexible data model and schema design. Unlike relational databases, where schema design needs to be established before insertion of records into the table, MongoDB allows for the schema design to be set during the inserting of a document into a collection.

While designing the MongoDB data models, it is essential to consider the following:

- ❖ Data retrieval patterns
- ❖ The needs of the applications, and
- ❖ The performance characteristics of the database engine.

❖ Elastic Scalability

Like most NOSQL databases, MongoDB has been built with focus on unlimited growth through elastic scalability. MongoDB meets this demand of unlimited growth by using the concept of **Sharding** (or Partitioning). Sharding is a functionality that is automatic and built into the database. Sharding allows MongoDB to distribute data across various physical partitions (referred to as shards).

Sharding may be of three types:

- ❖ Range-based Sharding
- ❖ Hash-based Sharding
- ❖ Location-aware Sharding

❖ High Performance

Combining all the characteristics viz. Expressive Query Language, Strong Consistency, Elastic Scalability, Flexible Data Model and Secondary Indexes, MongoDB allows its developers to

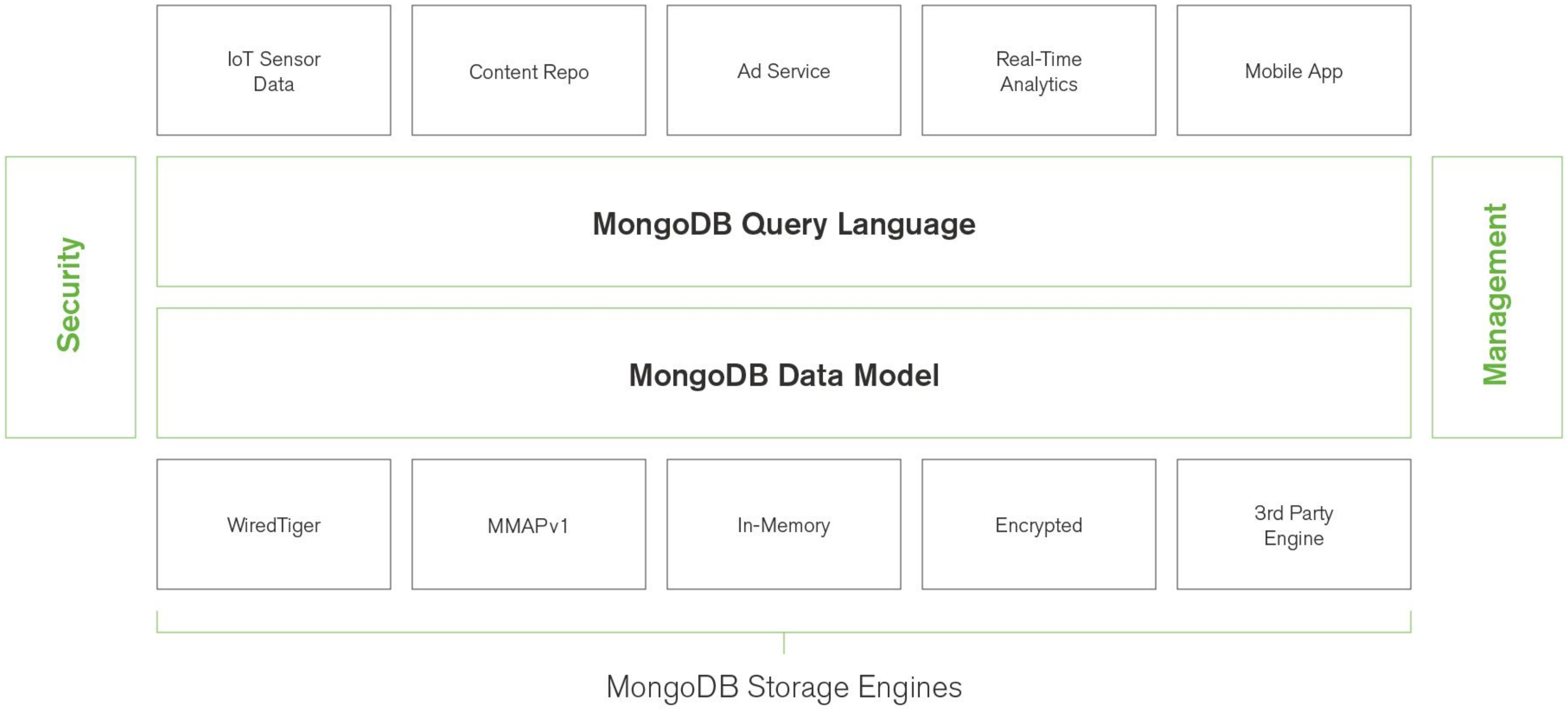
- ❖ build high performance applications
- ❖ build applications at any scale, in terms of throughput and latency.

Scaling comparison across DBs

	MongoDB	Relational	Key-value
Scale-out Commodity Hardware	Yes	No	Yes
Automatic Sharding	Yes	No	Yes
Shard by Hash	Yes	Manual	Yes
Shard by Range	Yes	Manual	No
Shard by Location	Yes	Manual	No
Automatic Data Rebalancing	Yes	Manual	Limited

Pluggable Storage Architecture

Pluggable Storage Architecture



Pluggable Storage Architecture

For application flexibility and to address the diverse needs of organizations, MongoDB has a pluggable storage architecture. With the use of its pluggable storage engines, MongoDB can extend its own capabilities and configure itself for optimized use of specific hardware architectures.

Using this approach, users can leverage this single database and its abundant features across various applications where each application is being powered by a different MongoDB pluggable storage engine.

Four storage engines are supported within MongoDB version 3.2. They are:

- ❖ The WiredTiger storage engine
- ❖ The Encrypted storage engine
- ❖ The In-memory storage engine
- ❖ The MMAPv1 storage engine

WiredTiger Storage Engine

✓For read and write operations, it uses document-level concurrency control .

✓Only intent-locks are used by WiredTiger at global, database and collection level

✓When a conflict is detected by the storage engine while performing two separate operations, one operation is made to undergo the write-conflict to ensure that MongoDB is transparently retrying the operation.

✓WiredTiger also provides the functionality of point-in-time snapshot of any transaction to check and maintain data consistency.

✓WiredTiger also creates checkpoints (recovery points) to help in data recovery and also maintain consistency.

Encrypted Storage Engine

- ✓ As the name suggests, the Encrypted storage engine is used for protecting data.
- ✓ This is achieved by using encrypted keys that have been secured by the KMIP – Key Management Interoperability Protocol.
- ✓ This storage engine provides encryption “at rest”. This feature is essential for organizations that belong to the financial industry, healthcare industry and so on where data is highly sensitive.
- ✓ To allow for only authorized access to all protected data, MongoDB data is encrypted at all levels including operating system level as well as at the storage level.

Encrypted Storage Engine (Cntd.)

✓ Transparent disk-level encryption is provided by the ability to integrate with 3rd party libraries such as :

- ➢ Linux Unified Key Setup (LUKS)
- ➢ IBM Guardium Data Encryption
- ➢ Vormetric Data Security Platform
- ➢ Bitlocker Drive Encryption

In-memory Storage Engine

✓ An in-memory storage engine is that which allows for the storage of table data in memory to allow for faster querying and data retrieval.

✓ MongoDB's usage of in-memory storage engine allows for faster read and write operations.

✓ The in-memory storage engine improves the query throughput by drastically reducing the response time for any operation.

✓ The delay that is caused due to data retrieval or data storage by frequently accessing the storage disk is avoided by use of this storage engine.

MMAPv1 Storage Engine

✓ The MMAPv1 storage engine is MongoDB’s original storage engine.

✓ Its key functionality is to manage how data gets stored at the disk level.

✓ This storage engine uses memory-mapped files for storing data.

✓ Files are assigned to virtual memory blocks.

✓ These virtual memory blocks have direct byte-for-byte correlation.

✓As and when the data is accessed, they are mapped to memory.

✓ Data that is not accessed remains un-mapped.

✓ Once the files are mapped, interaction with MongoDB takes place as it would if the files were in memory.

Reference

MongoDB Strong Consistency Features
https://quabase.sei.cmu.edu/mediawiki/index.php/MongoDB_Consistency_Features

Architecture Concepts
http://docs.mongodb.org

In-memory Storage Engine
http://siliconangle.com/blog/2015/11/03/mongodb-3-2-gets-an-in-memory-storage-engine/

MMAPv1 Storage Engine
http://learnmongodbthehardway.com/schema/chapter3/