DATABASE Decision Matrix :Data-Models Their Capabilities, Applications and limitations

Data Base and Data Model	Capabilities	Applications	Limitations
Key-Value MemcacheDB, Redis, DynamoDB, Riak	- The simplest model where each object is retrieved with a unique key, with values having no inherent model - Utilize in-memory storage to provide fast access to optional persistence - Other data models built on top of this model to provide more complex objects	- Applications requiring fast access to a large number of objects, such as caches or queues - Applications that require fast-changing data environments like mobile, gaming, online ads	- Cannot update subset of a value - Does not provide querying - As number of objects becomes large, generating unique keys could become complex
Document-oriented MongoDB,CouchDB,Apache Solr, Elastic Search	- Extension of the key-value model, where value is a structured document - Documents can be highly complex, hierarchical data structures without requiring pre-defined “schema” - Supports queries on structured documents - Search platforms are also document-oriented	- Applications that need to manage a large variety of objects that differ in structure - Large product catalogs in e-commerce, customer profiles, content management applications	- No standard query syntax - Query performance not linearly scalable - Join queries across collections not efficient
Column-Oriented Cassandra,BigTable,HBase,Apache Accumulo	- Extension of the key-value model, where the value is a set of columns (column-family) - A column can have multiple time-stamped versions - Columns can be generated at run-time and not all rows need to have all columns	- Storing a large number of time-stamped data like event logs, sensor data - Analytics that involve querying entire columns of data such as trends or time-series analytics	- No join queries or sub-queries - Limited support for aggregation - Ordering is done per partition, specified at table creation time
Graph-oriented Neo4J, Apache Giraph	- Models graphs consisting of nodes and edges with properties (meta-data) describing them - Implement very fast graph traversal operations - Also support indexing of metadata to enable graph traversal combined with search queries	- Applications that deal with objects with a large number of inter-relations - Applications like social networking friends-networks, hierarchical role-based permissions, complex decision trees, maps, network topologies	- Difficult to scale for large data sets for generic graphs - Giraph uses the Bulk Synchronous Parallel model to overcome some of the scalability limitations
Relational(FYI) MySql,PostgreSQL,MariaDB,Oracle, SQL Server,CockroachDB	- Conventional RDBMS structure consisting of fixed schema with ACID properties - Provides well-documented and widely supported SQL syntax - Capable of complex queries including subqueries and joins	- Transactional data applications like ERP, CRM, Banking, etc. - Applications where data volume is limited and schema are by and large fixed	- Lacks horizontal scalability and hence limited in handling “big data” - Not efficient at handling complex multi-level nested data - Cannot handle “unstructured” data where the structure is not known at design time

Comparison Of NoSQL Database: Cassandra Vs MongoDB Vs CouchBase

Features	Cassandra	MongoDB	CouchBase
Features	**Cassandra**	**MongoDB**	**CouchBase**
CAP Theorem	**AP**(Cassandra with tunable consistency)	**CP**	Naturally **CP** type system and it can be set up as an **AP** system with multiple clusters(XDCR).
Features	FLEXIBILITY SCALABILITY PERFORMANCE DISTRIBUTION SIMPLICITY TRANSACTION	FLEXIBILITY SCALABILITY PERFORMANCE EXPRESSIVE INDEXATION COMMUNITY	FLEXIBILITY SCALABILITY PERFORMANCE MOBILITY REAL-TIME ADMINISTRATION

Type	Column Based	Document Based	Document +Key / Value Based
	This model looks at first glance at a table in RDBMS except that with a column-oriented NoSQL DB, the number of columns is dynamic. Indeed, in a relational table, the number of columns is fixed from the creation of the table schema and this number remains the same for all the records in this table.	This model is based on the key-value paradigm. The value, in this case, is a JSON or XML document. The advantage is to be able to retrieve, via a single key, a hierarchically structured information set. The same operation in the relational world would involve several joins.	**Key-Value:** This model can be assimilated to a distributed hashmap. The data are therefore simply represented by a key/value pair. The value can be a simple string, a serialized object ... This lack of structure or typing has an important impact on the query. **Document**: This model is based on the key /value paradigm. The value, in this case, is a JSON or XML document. The advantage is to be able to retrieve, via a single key, a hierarchically structured information set. The same operation in the relational world would involve several joins.
Description	**FLEXIBILITY:** semi-structured and unstructured, which runs through the demands of modern applications today. Also dynamically satisfies changes in your data structures when your data needs grow. **SCALABILITY:** Allows you to easily add capacity online to accommodate more customers and more data when you need. Adding nodes and clusters on the fly without the need to restart a Cassandra server **PERFORMANCE:** With an ultra-short response to CRUD operations and a linear load-to-time curve that doubles Nodes to ensure a response that meets your customers' expectations. **DISTRIBUTION:** Gives you the maximum flexibility to distribute data wherever you need by reproducing data across multiple data centers, such as in the public and private clouds that are becoming extremely common deployment environments. **SIMPLICITY:** With all the incorporation of several nodes into a cluster or cluster master, there is no complex configuration to implement, so the management of the administrative roles becomes greatly simplified. **TRANSACTION:** Atomicity, isolation, and durability according to the ACID model through logs ensure data durability in the event of hardware failures, as well as transaction isolation, atomicity, with consistency And high consistency. www.datastax.com	**FLEXIBILITY:** The MongoDB document data model makes it easy for you to store data from any structure and allows you to dynamically modify your NoSQL database schemas **SCALABILITY:** Increase proportionally or horizontally, from a single server to thousands of nodes. Deploy in the cloud and across multiple data centers. **PERFORMANCE:** High-performance systems executed on a scale. Performs millions of operations per second. It reduces the workload for data read and writes operations. **EXPRESSIVE:** The MongoDB data query language provides superior-level operators for on-site updates. Drivers for just about any computer programming languages, such as Java, PHP, NodeJS, **INDEXATION:** Fast access, fine-grained data, including fully consistent indexes on any field, As well as geospatial, as a search text and TTL type indexes. **COMMUNITY:** MongoDB has the advantage of having a large community of developers and DBAs experiencing strong growth, the fastest one in the world of NoSQL, and split behind their software to ensure its success, providing software services to make life easier for Developers. www.mongodb.org	**FLEXIBILITY:** Configurable cross-data for replication, including active/active, for large-scale implementation in public or private clouds **SCALABILITY:** Add or remove nodes in an on-demand cluster by sharing the architecture with no point of failure. **PERFORMANCE:** An integrated cache system, in this case, Memcached, which provides low latency access to data with or without the lock to ensure I / O operations at the various nodes and clusters. **MOBILITY:** An embedded database for mobile and IoT application requests for offline access and automatic synchronization. **REAL-TIME:** Integration with Hadoop, Elasticsearch, and an API for continuous data to storm for real-time analytics. **ADMINISTRATION:** An integrated administration console and scripting API with a large group control to manage large deployments. www.couchbase.com
When To Use	• Keeping unstructured, non-volatile information: If a large collection of attributes and values needs to be kept for long periods of time, column-based data stores come in extremely handy. • Scaling: Column-based data stores are highly scalable by nature. They can handle an awful amount of information.	• Nested information: Document-based data stores allow you to work with deeply nested, complex data structures. • JavaScript friendly: One of the most critical functionalities of document-based data stores is the way they interface with applications: Using JS-friendly JSON.	• Nested information: Document-based data stores allow you to work with deeply nested, complex data structures. • JavaScript friendly: One of the most critical functionalities of document-based data stores is the way they interface with applications: Using JS-friendly JSON.
IoT system Analysis	Cassandra can be a better choice, with an interesting MongoDB alternative to the large quantity management level and data stream at the same time. Apache Cassandra performs better at reading, Insert, Updating and Delete levels, ie CRUD operations necessarily used in mass in a modern IoT architecture.	MongoDB does not handle the task of administering the cache and leaves this approach to a low-level operating system for the managed.	Couchbase, its behavior, and its performance are proportional to the hardware aspects and the amount of memory allocated for the Buckets instances to handle the data in a normal situation exercised by the other two NoSQL engines.

Use Comcast tool : https://friday.po-g1.cf.comcast.net/dbRegistration