Mongodb

☯What is MongoDB

- "MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling."
- In simple words, you can say that - Mongo DB is a document-oriented database. It is an open-source product, developed and supported by a company named 10gen.
- MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.
- "MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen
- MongoDB was designed to work with commodity servers. Now it is used by the company of all sizes, across all industry.
- MongoDB is an open-source document-oriented NoSQL database which is written in C++.
- MongoDB is schema-less database system and hence it's very easy to add new fields in it. It is a distributed system hence data recovery is instant and more reliable.

✴Features of MongoDB

These are some important features of MongoDB:

1. **Support ad hoc queries** In MongoDB, you can search by field, range query and it also supports regular expression searches.

2. **Indexing**: You can index any field in a document.

3. **Replication**: MongoDB supports Master Slave replication. A master can perform Reads and Writes and a Slave copies data from the master and can only be used for reads or back up (not writes)

4. **Duplication of data**: MongoDB can run over multiple servers. The data is duplicated to keep the system up and also keep its running condition in case of hardware failure.

5. **Load balancing**: It has an automatic load balancing configuration because of data placed in shards.

6. Supports map reduce and aggregation tools.

7. Uses JavaScript instead of Procedures.

8. It is a schema-less database written in C++.

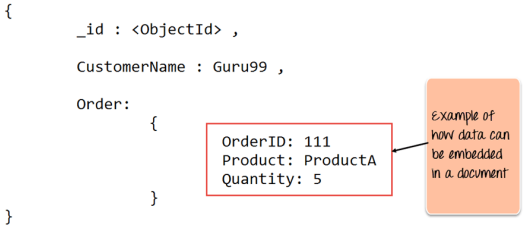
9. Provides high performance.

10. Stores files of any size easily without complicating your stack.

✴MongoDB Example:

The below example shows how a document can be modeled in MongoDB.

- The _id field is added by MongoDB to uniquely identify the document in the collection.
- What you can note is that the Order Data (OrderID, Product, and Quantity) which in RDBMS will normally be stored in a separate table, while in MongoDB it is actually stored as an embedded document in the collection itself. This is one of the key differences in how data is modeled in MongoDB.



✳REST API

- An API, or application programming interface, is a set of rules that define how applications or devices can connect to and communicate with each other.
- A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.
- REST stands for representational state transfer and was developed by Roy Fielding in 2000.
- REST is a set of architectural constraints, not a protocol or a standard.
- API developers can implement REST in a variety of ways.
- When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint.
- This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text.
- JSON is the most generally popular file format to use because, despite its name, it's language-agnostic, as well as readable by both humans and machines.

✳REST Design Principles

Principles of Rest API

1. Client-Server decoupling

In a REST API design, client and server programs must be independent. The client software should only know the URI of the requested resource; it should have no additional interaction with the server application.

2. Uniform Interface

All API queries for the same resource should look the same regardless of where they come from. The REST API should ensure that similar data, such as a user's name or email address, is assigned to just one uniform resource identifier (URI).

3. Statelessness

REST APIs are stateless, meaning each request must contain all the information needed to process it.

4. Layered System architecture

REST API requests and responses are routed through many tiers. REST APIs must be designed so neither the client nor the server can tell whether they communicate with the final application or an intermediary.

5. Cacheable

Wherever feasible, resources should be cacheable on the client or server side. Server responses must additionally indicate if caching is authorized for the offered assistance. The objective is to boost client-side speed while enhancing server-side scalability.

6. Code on Demand

REST APIs typically provide static resources, but in rare cases, responses may include executable code (such as Java applets). In these cases, perform the code when necessary.

☢Rules of REST API:

1.REST is based on the resource or noun instead of action or verb based. It means that a URI of a REST API should always end with a noun.

Example: /api/users

2. HTTP verbs are used to identify the action. Some of the HTTP verbs are – GET, PUT, POST, DELETE, UPDATE, PATCH. HTTP verbs: Some of the common HTTP methods/verbs are described below:

- GET : Retrieves one or more resources identified by the request URI and it can cache the information receive.
- POST : Create a resource from the submission of a request and response is not cacheable in this case. This method is unsafe if no security is applied to the endpoint as it would allow anyone to create a random resource by submission.
- PUT : Update an existing resource on the server specified by the request URI.
- DELETE : Delete an existing resource on the server specified by the request URI. It always return an appropriate HTTP status for every request.

3. A web application should be organized into resources like users and then uses HTTP verbs like - GET, PUT, POST, and DELETE to modify those resources. And as a developer it should be clear that what needs to be done just by looking at the endpoint and HTTP method used.

4. Always use plurals in URL to keep an API URI consistent throughout the application.

5. Send a proper HTTP code to indicate a success or error status,

URI	HTTP Verb	Description
api/users	GET	Get all users
api/users/new	GET	Show form for adding new user
api/users	POST	Add a user
api/users/1	PUT	Update a user with id = 1
api/users/1/edit	GET	Show edit form for user with id = 1
api/users/1	DELETE	Delete a user with id = 1
api/users/1	GET	Get a user with id = 1

Rest API Examples

Now that you are clear about REST API's meaning, let us know what REST API examples are.

Twitter: Twitter API permits third-party applications to access and write data from Twitter. Write and post tweets, share tweets, and read profiles using it. This API is handy for obtaining and analyzing massive quantities of tweets regarding certain subjects.

Instagram: The Instagram Basic Display API provides access to user profiles, images, and videos. You may use this API and others to create applications that pull user data and integrate it into your product. Instagram also provides a Graph API for professional Instagram accounts to manage online activity.

Characteristics of Web Services:

Web services have the following characteristics:

- XML-based
- Coarse-grained
- Loosely coupled
- Capability to be synchronous and asynchronous
- Supports RPC

XML-based

A web service uses XML at information representation and record transportation layer. Using XML, there is no need of networking, operating system, or platform binding. Web offering based application is highly interoperable application at their middle level.

Coarse-grained

In the coarse-grained operation, a few objects hold a lot of related data. It provides broader functionality in comparison to fine-grained service. It wraps one or more fine-grained services together into a coarse-grained service. It is fine to have more coarse-grained service operations.

Loosely Coupled

A web service supports loosely coupled connections between systems. It communicates by passing XML message to each other via a web API. Web API adds a layer of abstraction to the environment that makes the connection adaptable and flexible.

Capability to be synchronous and asynchronous

Synchronous Web services are invoked over existing Web protocols by a client who waits for a response. Synchronous Web services are served by RPC-oriented messaging.

Asynchronous Web services are invoked over existing Web protocols by a client who does not wait for a response. The document-oriented messaging often used for asynchronous Web services. Asynchronous Web Service is a crucial factor in enabling loosely coupled system.

Supports RPC

A web service supports RPC through offering services of its personal, equivalent to those of a traditional aspect.

A web service is a web resource. We can access a web service using platform-independent and language-neutral web protocols, such as HTTP. HTTP ensures easy integration of heterogeneous environment.

A web service is typically registered. It can be located through a web service registry. A registry enables service consumers to find service that matches their needs. The service consumers may be human or other application.