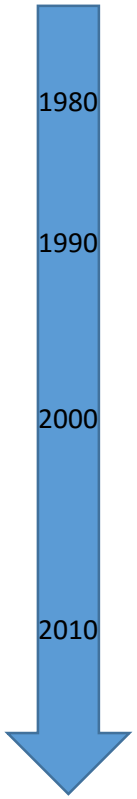
Introduction to MongoDB

Requirements of Modern Application

- Big Data
  - Lots of storage
  - Lots of access
- Ever Changing Features
  - Simpler data models
- Flexible deployments
  - Cloud-enabled

Relational Databases



Persistence

Reporting

Integration

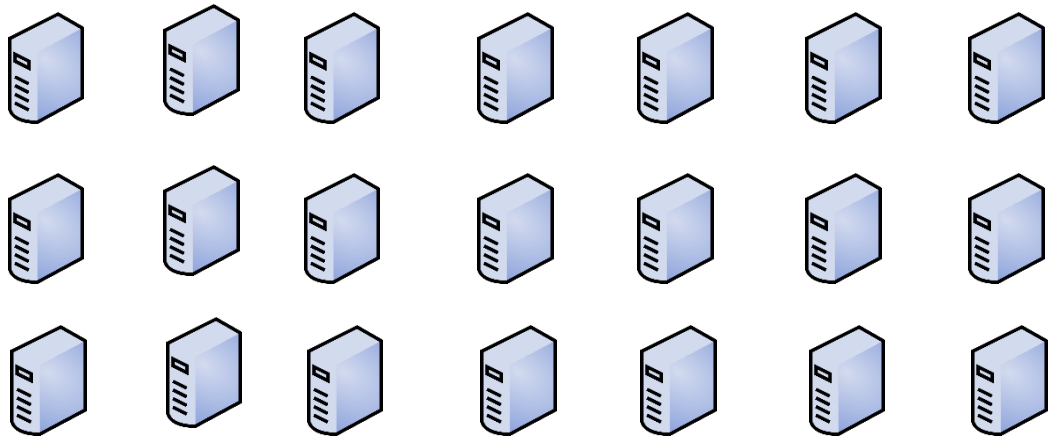
Transactions

SQL

Relational Database

- Impedance Mismatch

The age of Big Data

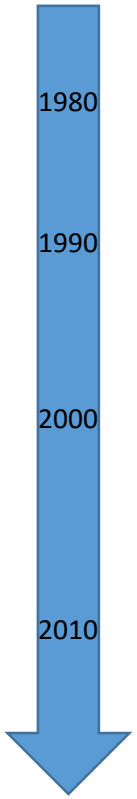


Not a Relational Database

Google  Bigtable

Amazon  Dynamo

NoSQL



Non-relational

Open-source

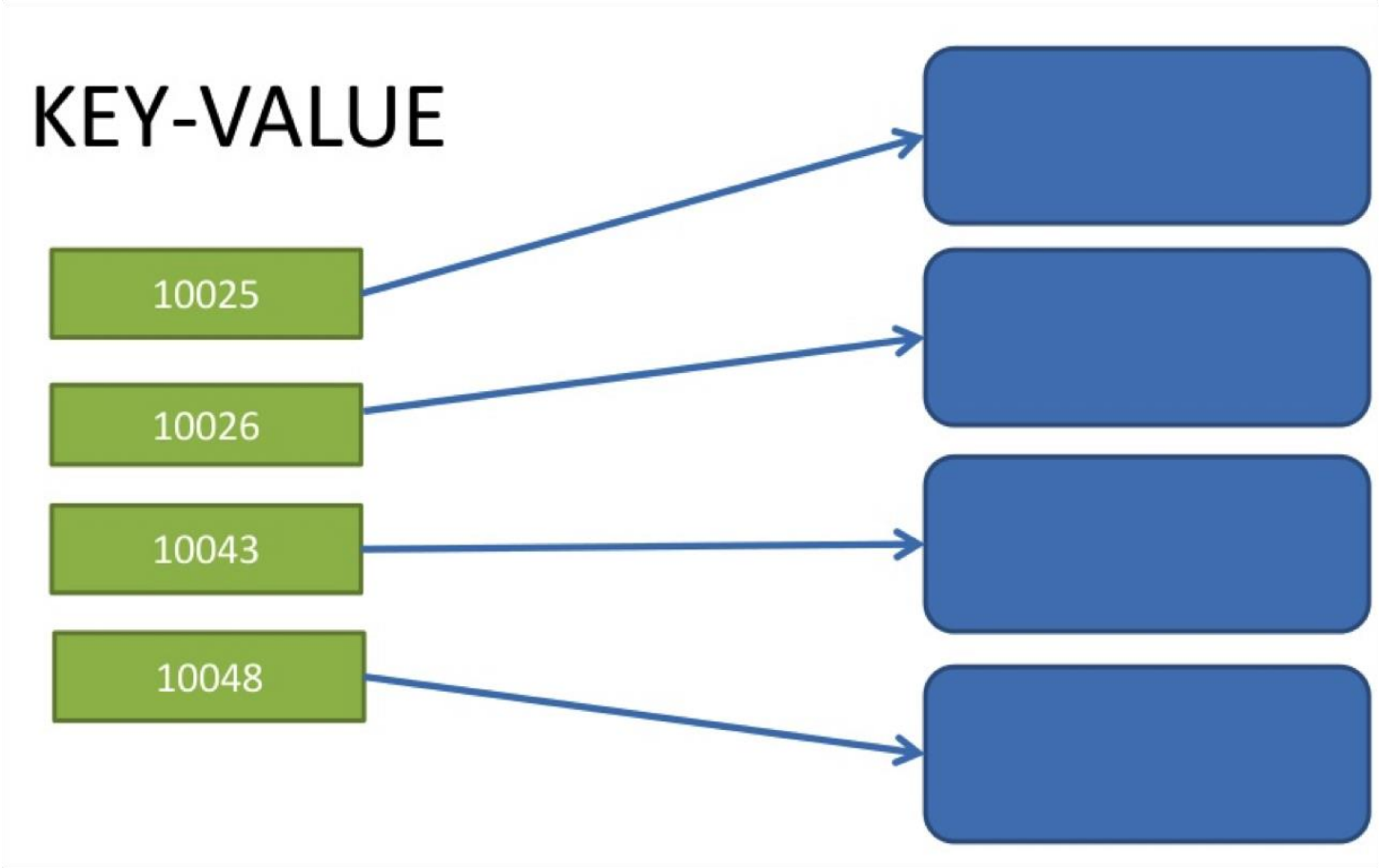
Cluster friendly

Schema-less

Data Model

- Key-value
- Document
- Graph
- Column-family

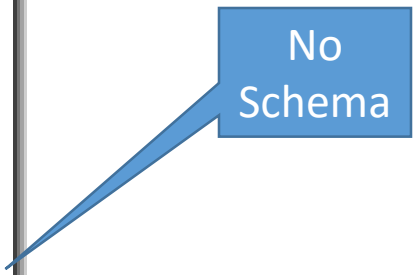
Key-value



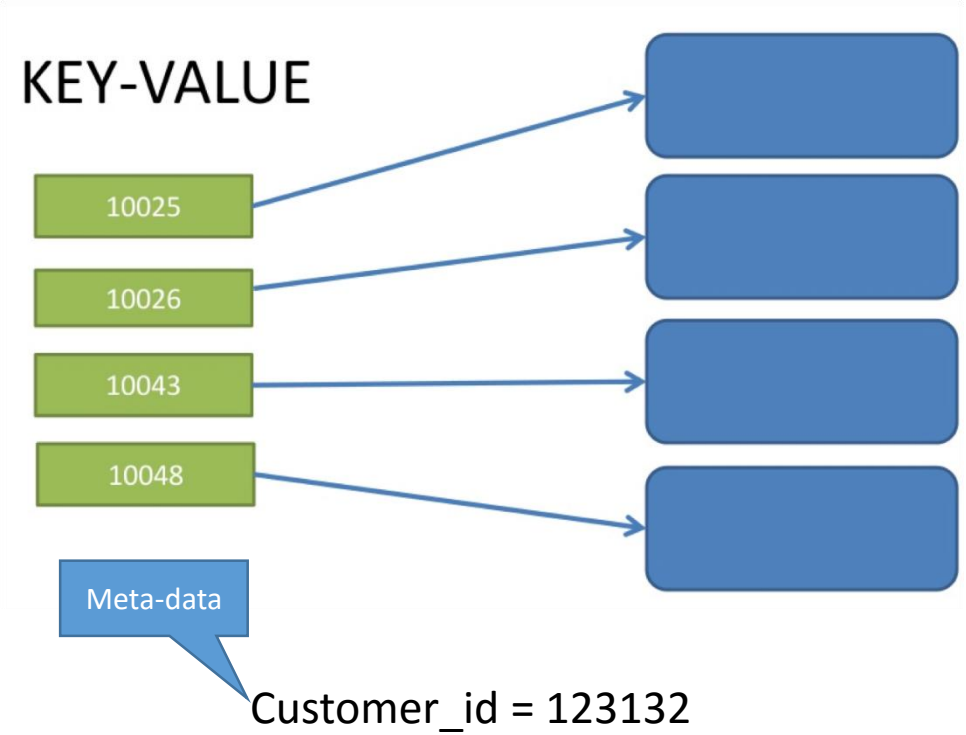
Source: http://bigdata-blog.com/wp-content/uploads/2015/05/keyvalue.png

Document


```
{
  "_id": "4c6cd19abac7061798000002",
  "CityId": 42231,
  "CountryID": 1,
  "RegionID": 833,
  "City": "Herat",
  "Latitude": 34.3330001831055,
  "Longitude": 62.2000007629395,
  "TimeZone": "+04:30",
  "DmaId": 0,
  "County": "HERA",
  "Code": ""
}
```

Source: http://new.mongovue.com/wp-content/uploads/2010/11/image2.png



Document

Key

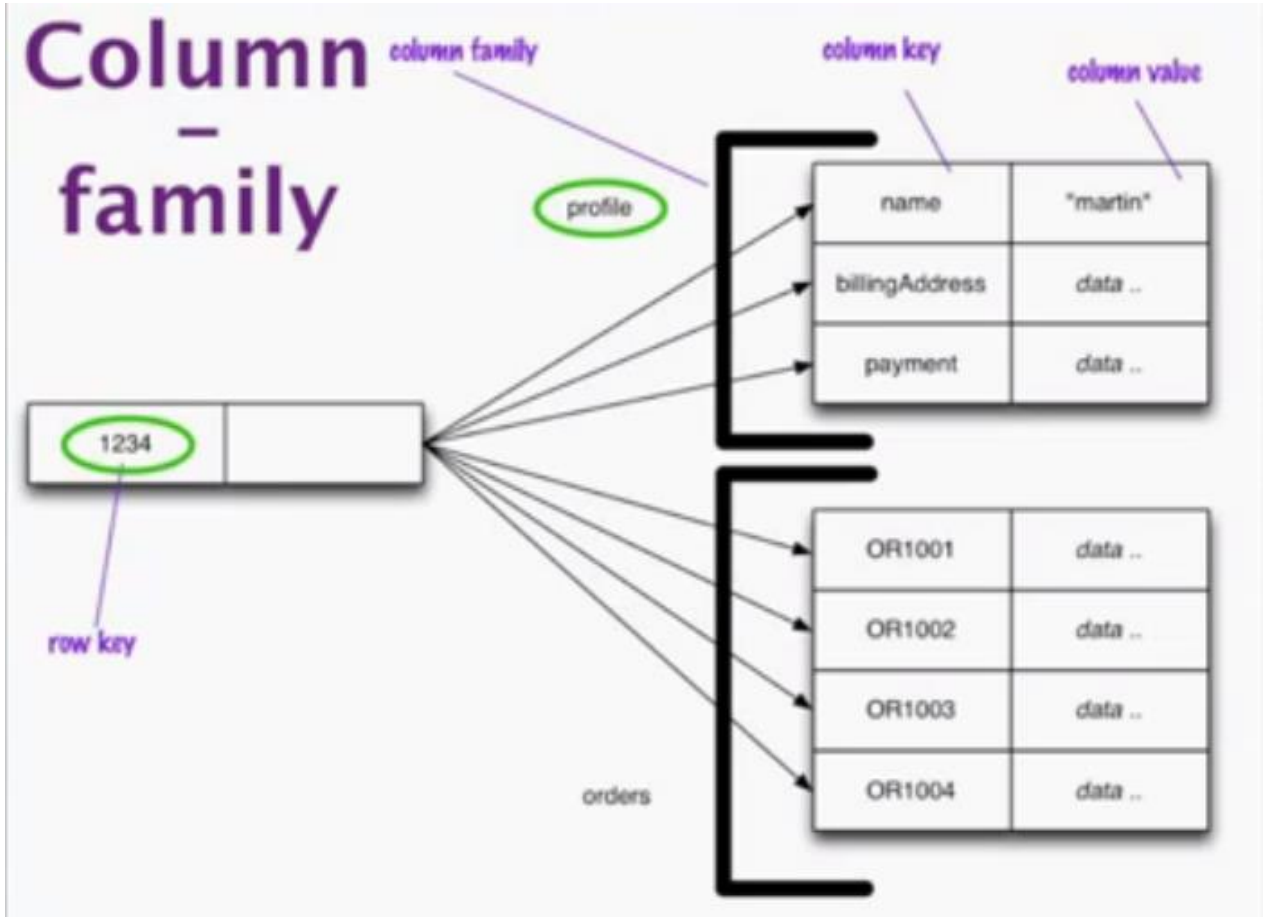

```
{
  "_id": "4c6cd19abac7061798000002",
  "CityId": 42231,
  "CountryID": 1,
  "RegionID": 833,
  "City": "Herat",
  "Latitude": 34.3330001831055,
  "Longitude": 62.2000007629395,
  "TimeZone": "+04:30",
  "DmaId": 0,
  "County": "HERA",
  "Code": ""
}
```


Source: http://new.mongovue.com/wp-content/uploads/2010/11/image2.png

Aggregate

- Aggregate-oriented databse
  - Value
  - Document
- Aggregate is the single unit that we access

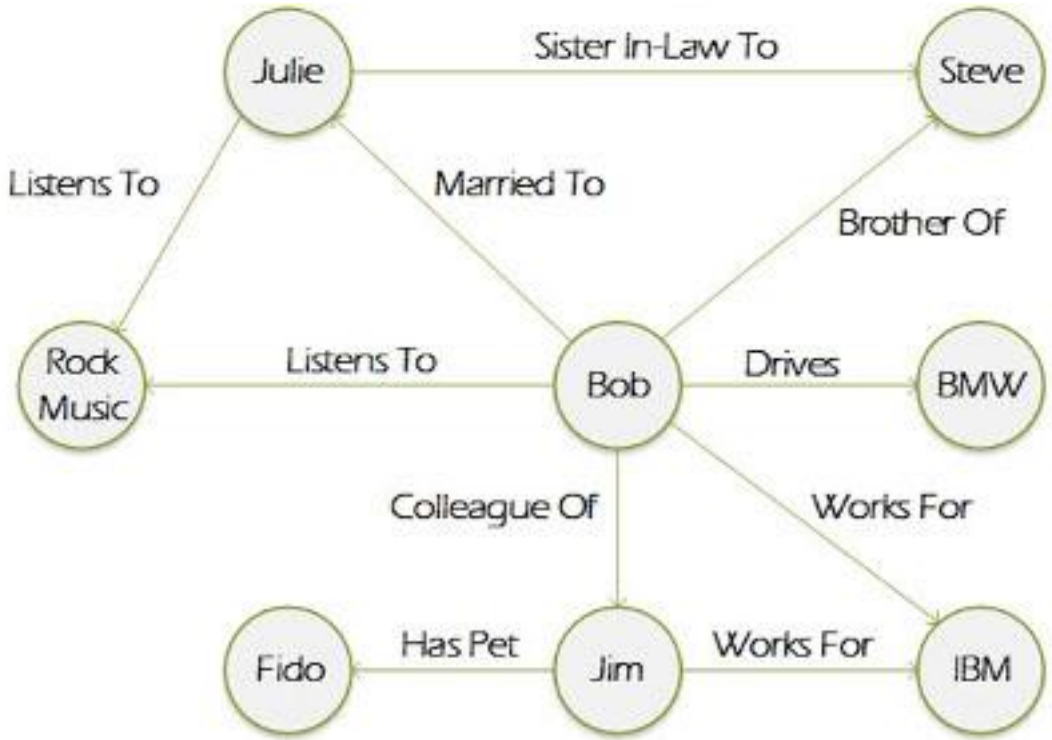
Column-family



Source: https://csetech.files.wordpress.com/2013/06/colfamily.png

Aggregate model on Clusters

Graph



Source: http://itknowledgeexchange.techtarget.com/overheard/files/2014/01/Graph-database-sketch.jpg

NoSQL and Consistency

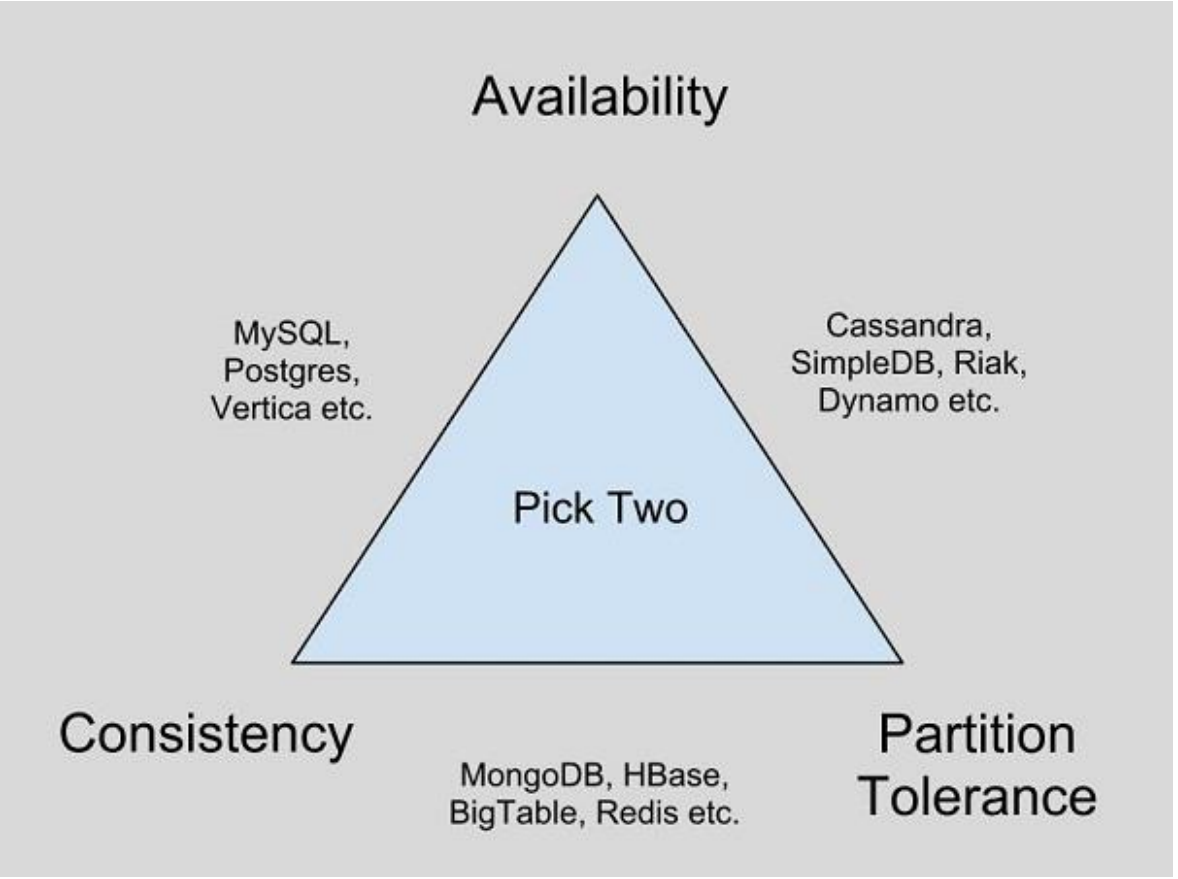
- ACID?

NoSQL and Consistency

- Graph
- Aggregation-oriented
  - Aggregate is the boundary
- Version stamps

NoSQL and Consistency

- Logical
- Replication



Source: http://wp1.cdn.brainsins.com/en/wp-content/uploads/2013/10/cap-theorem.png

Trade-offs

- Consistency vs. Response-time

When and Why to use a NoSQL database

- Large scale data
  - Mobile Cloud Applications
- Easier Development
  - Natural Aggregate
- Web Services/Applications enabling access to data

Introduction

- A NO-SQL database
- Conceptualized to be a distributed database
- Easy to scale out
- Allows schema-free storage of data as collections
- Current version 3.2

source: MongoDB: The Definitive Guide

Installation

- Windows install
  - Download the Windows installer
  - Create a C:\data\db directory
  - bin\mongod.exe
    - --dbpath PATH to specify any other directory as the base directory
  - bin\mongo.exe
  - Can also be installed as a service

source: MongoDB: The Definitive Guide

Documents & Collections

- A *document* is the basic unit of data for MongoDB, roughly equivalent to a row in a relational database management system

- A *collection* can be thought of as the schema-free equivalent of a table.

source: MongoDB: The Definitive Guide

Documents

- Documents: an ordered set of keys with associated values
  - Naturally fits with data structures like map, hash etc.
  - E.g. {“greeting” : “Hello”, “foo”: 3}

- Key/value pairs in documents are ordered
  - {“greeting” : “Hello”, “foo”: 3} & {“foo”: 3, “greeting” : “Hello”}

  - Values could be several different data types including embedded documents

source: MongoDB: The Definitive Guide

Keys in Documents

- The keys in a document are strings. Any UTF-8 character is allowed in a key, with a few notable exceptions:
  - Keys must not contain the character \0 (the null character). This character is used to signify the end of a key.
  - The . and $ characters have some special properties and should be used only in certain circumstances, as described in later chapters. In general, they should be considered reserved.
  - Keys starting with _ should be considered reserved;
- MongoDB is type-sensitive and case-sensitive. For example, these documents are distinct:
  - {"foo" : 3} & {"foo" : "3"}
  - {"foo" : 3} & {"Foo" : 3}
- No duplicate keys
  - {"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}

source: MongoDB: The Definitive Guide

Example


```
{
  "_id" : ObjectId("54c955492b7c8eb21818bd09"),
  "address" : {
     "street" : "2 Avenue",
     "zipcode" : "10075",
     "building" : "1480",
     "coord" : [ -73.9557413, 40.7720266 ],
  },
  "borough" : "Manhattan",
  "cuisine" : "Italian",
   "name" : "Vella",
  "restaurant_id" : "41704620"
}
```


source: https://docs.mongodb.org/getting-started/shell/introduction/

Collections

- A group of documents like tables are a group of rows
- Collections are schema-free: documents within a single collection can have any number of different “shapes.”
  - {"greeting" : "Hello, world!"}
  - {"foo" : 5}

source: MongoDB: The Definitive Guide

Collections

- Why do we need separate collections at all?
- Keeping different kinds of documents in the same collection can be a nightmare for developers and admins.
  - Each query should return document of a certain kind
- It is much faster to get a list of collections than to extract a list of the types in a collection.
- Grouping documents of the same kind together in the same collection allows for data locality
- Putting only documents of a single type into the same collection, we can index our collections more efficiently.

source: MongoDB: The Definitive Guide

Collections - Naming

- The empty string ("") is not a valid collection name.
- Collection names may not contain the character \0 (the null character.
- You should not create any collections that start with *system.* - a prefix reserved for system collections.
  - For example, the *system.users* collection contains the database's users, and the *system.namespaces* collection contains information about all of the database's collections.
- User-created collections should not contain the reserved character $ in the name.

source: MongoDB: The Definitive Guide

Databases

- MongoDB groups collections into databases
- A single instance of MongoDB can host several databases – each of which is completely independent
- Each database is stored in separate files on disk

source: MongoDB: The Definitive Guide

Databases - Naming

- The empty string ("") is not a valid database name.
- A database name cannot contain any of these characters: ' ' (a single space), ., $, /, \, or \0 (the null character).
- Database names should be all lowercase.
- Database names are limited to a maximum of 64 bytes.

source: MongoDB: The Definitive Guide

Reserved Databases

- *admin*
  - This is the “root” database, in terms of authentication. If a user is added to the *admin* database, the user automatically inherits permissions for all databases. There are also certain server-wide commands that can be run only from the *admin* database, such as listing all of the databases or shutting down the server.
- *local*
  - This database will never be replicated and can be used to store any collections that should be local to a single server
- *config*
  - When Mongo is being used in a sharded setup, the *config* database is used internally to store information about the shards.

source: MongoDB: The Definitive Guide

Mongo Shell

- The shell can connect to daemon running on any other machine
  - bin/mongo www.server.com:port
  - By default starts with “test” databse
  - db variable points to the current database
    - Can use connect to connect with multiple databases
  - bin/mongo localhost:27017/admin – now connects to admin database
  - bin/mongo –nodb - now does not connect with any database

source: MongoDB: The Definitive Guide

Binary JSON (BSON)

- Binary JSON (BSON): representation of documents that is shared by all drivers, tools, and processes in the MongoDB ecosystem.
- BSON is a lightweight binary format capable of representing any MongoDB document as a string of bytes.
- The database understands BSON, and BSON is the format in which documents are saved to disk.

source: MongoDB: The Definitive Guide

BSON

- **Lightweight**
  - Overhead is minimum
  - Good to use over the network.
- **Traversable**
  - BSON is designed to be traversed easily.
- **Efficient**
  - Encoding data to BSON and decoding from BSON can be performed very quickly in most languages due to the use of C data types.

source: MongoDB: The Definitive Guide

Wire Protocol

- Clients communicate with the MongoDB server using a lightweight TCP/IP Wire protocol; default port is 27017
  - A regular TCP/IP Socket
- Two types of messages
  - Client request
  - Database responses

source: MongoDB: The Definitive Guide

Wire protocol

Standard Message Header

```
struct MsgHeader {
    int32   messageLength; // total message size, including this
    int32   requestID;     // identifier for this message
    int32   responseTo;    // requestID from the original request
                           //   (used in reponses from db)
    int32   opCode;        // request type - see table below
}
```

source: MongoDB: The Definitive Guide

Wire Protocol -opcodes

Opcode Name	opCode value	Comment
OP_REPLY	1	Reply to a client request. responseTo is set
OP_MSG	1000	generic msg command followed by a string
OP_UPDATE	2001	update document
OP_INSERT	2002	insert new document
RESERVED	2003	formerly used for OP_GET_BY_OID
OP_QUERY	2004	query a collection
OP_GET_MORE	2005	Get more data from a query. See Cursors
OP_DELETE	2006	Delete documents
OP_KILL_CURSORS	2007	Tell database client is done with a cursor

source: MongoDB: The Definitive Guide

References

- https://www.youtube.com/watch?v=qI_g07C_Q5I