MongoDB for DBAs/Developers

Fábio Miguel Blasak da Fonseca

Image Copyright: https://www.percona.com/blog/2019/08/16/long-awaited-mongodb-4-2-ga-has-landed/

About me

Fábio Miguel Blasak da Fonseca

SCAN ME

Linkedin
https://www.linkedin.com/in/fabiomiguel/

GUORS

SCAN ME

Education

- MBA in Big Data and Data Science – Uniritter (WIP)
- MsC - Master in Computer Science – PUCRS
- MBA in Project Management – ESPM
- Bachelor in System Analysis - UNISINOS

Professional

GUDS

SCAN ME

- Database Administrator Consultant – Dell Technologies
- 18+ years experience in IT area
- Areas of Interest: Machine Learning, IA, Python, Git, Automation and Database
- GUORS (Oracle Users Group RS) Coordinator
- GUDS (Data Science Users Group) Coordinator

ORACLE ACE Associate · PMP · Open Certified Distinguished Technical Specialist · ITIL Foundation V3 · Lean Six Sigma Yellow Belt · Python Institute · Oracle Certified Professional · Oracle Certified Specialist · Oracle Certified Associate · Oracle Certified Associate · Oracle Certified Associate

Agenda

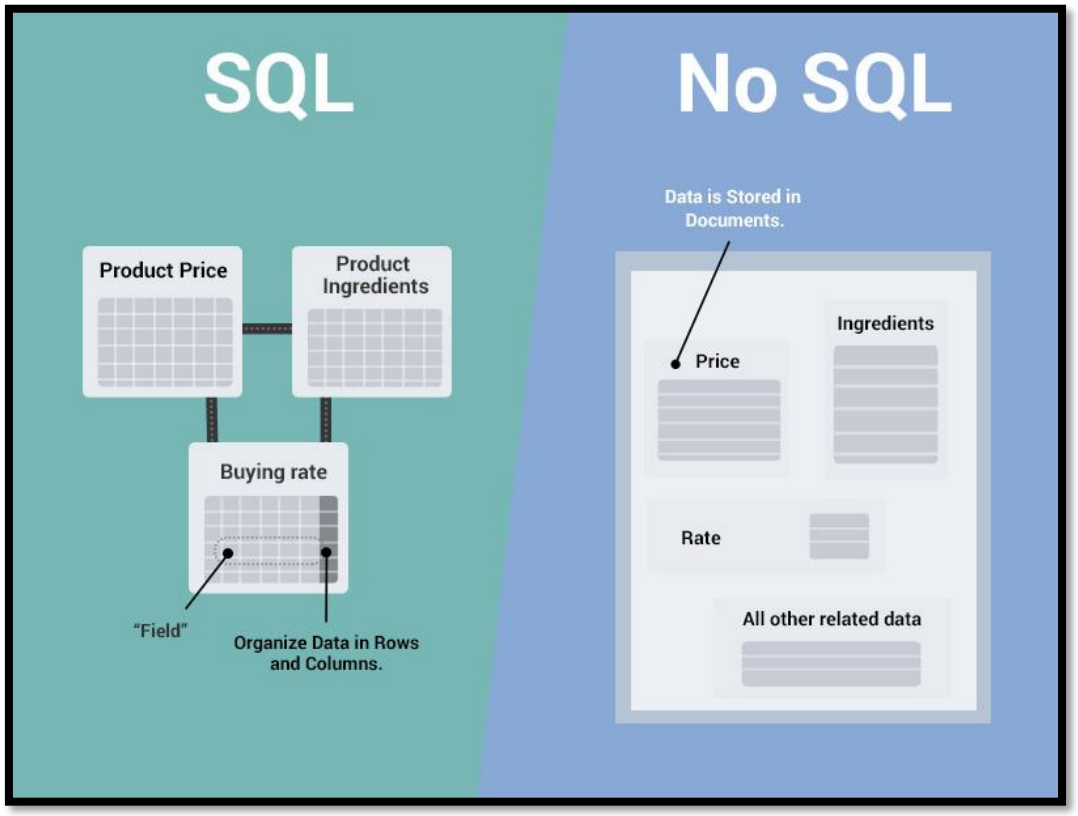
- Ice Breaker
- MongoDB Basic Concepts
- MongoDB Architecture
- MongoDB Tools
- Software Development Advices
- MongoDB Log
- Q&A

Ice Breaker

https://www.menti.com/bwh6659eum

MongoDB Basic Concepts

1. Data Model



RDBMS	MongoDB
Supports SQL Query Language	Supports JSON Query Language
Table	Collection
Row	Document or BSON Document
Column	Field
Supports Foreign Keys	N/A
Supports Triggers	N/A
Schema	Dynamic Schema (Schemaless)
Vertically Scalable	Horizontally Scalable

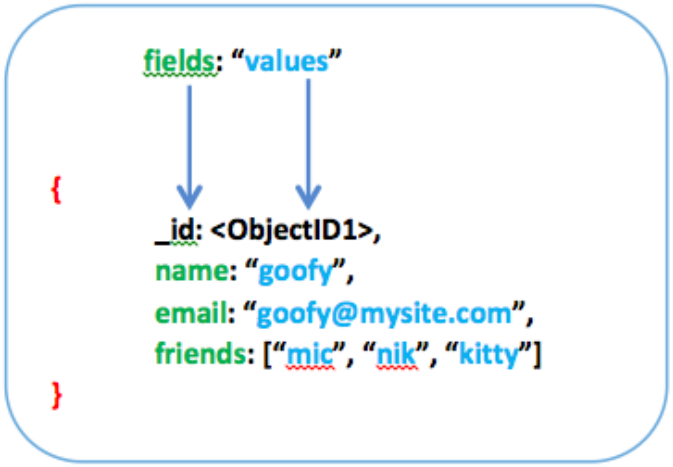
MongoDB Basic Concepts

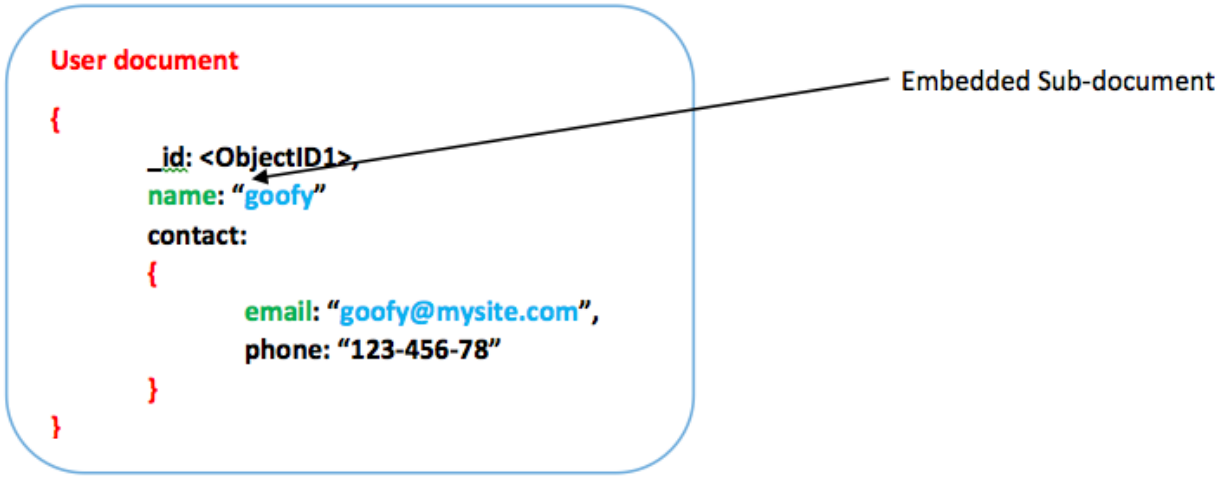
1. Data Model

MongoDB Basic Concepts

1. Data Model

Sample of a Document:





Sample of a Collection:

Documents are stored in MongoDB in JSON (JavaScript Object Notation) format.

MongoDB Basic Concepts

1. Data Model

Sample of a table in a relational database called “users”:

Name	Age
Albert Smith	10
Fabio Fonseca	20
John Smith	25
Sofia Rogers	24

If you want to run a query statement to return rows where age >=18 and ordered by age in ASC:

```
SELECT * FROM users where age >= 18 ORDER BY age ASC;
```

If you want to run same query against a collection called “users” with same two fields – “name” and “age”:

```
db.users.find({ age: { $gte: 18 } }).sort( { age: 1 } )
```

MongoDB uses **CRUD** operations to interact with database.

MongoDB Basic Concepts

2. CRUD

Data modifications are operations that involve creating, updating, or deleting data. These operations in MongoDB help modify the data of a collection.

Basic Syntax:
db.<collection>.<CRUD_Operation>({field1:value,field2:value…},{…})
db.<collection>.<CRUD_Operation>([{field1:value,field2:value…},{…},{…}],{…})

MongoDB Basic Concepts

2. CRUD

2.1 Type of Documents during a CRUD Operation:

1) Document
{ item: "journal", qty: 25, status: "A" }

2) Embedded/Nested Document
{ item: "journal", qty: 25, **size: { h: 14, w: 21, uom: "cm" }**, status: "A" }

3) Array
{ item: "journal", qty: 25, **tags: ["blank", "red"]**, dim_cm: [14, 21] }

4) Array of Embedded Documents
{ item: "journal", **instock: [{ warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 }]** }

MongoDB Basic Concepts

2. CRUD

2.2 Common CRUD Operations:

CRUD Operation	CRUD Command	RDBMS SQL/DML Statement
Create	db.products.**insertOne**({item:”card”, qty: 15});	insert into products (item,qty) values (‘card’,15);
Create	db.products.**insertMany**([{item:”card”, qty: 15}, {item:”stamps”, qty: 15}]);	insert into products (item,qty) values (‘card’,15); insert into products (item,qty) values (‘stamps’,15);
Read	db.products.**find**({item:”card”});	select * from products where item = ‘card’;
Update	db.products.**updateOne**({item:’card’}, {$set:{item:’new_card’});	update products set item = ‘new_card’ where item = ‘card’;
Update	db.products.**updateMany**({qty:15},{$set:{item:’new_item_title’});	update products set item = ‘new_item_title’ where qty = 15;
Update	db.products.**replaceOne**({item:’card’,qty:15},{item:’new item’,qty:100});	delete products where item=‘card’ and qty=15; insert into products (item,qty) values (‘new item’,100);
Delete	db.products.**deleteOne**({ "_id" : ObjectId("563237a41a4d68582c2509da") });	delete products where id = ‘563237a41a4d68582c2509da’;
Delete	db.products.**deleteMany**({qty:15});	delete products where qty = 15;

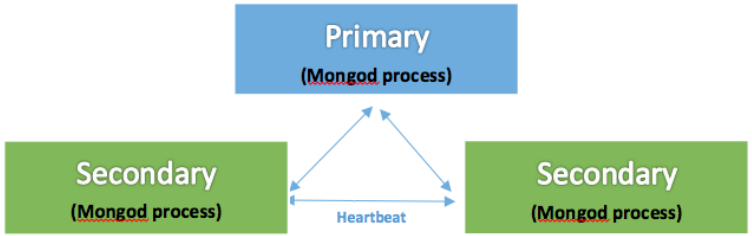
Note: If specify upsert:true in a CRUD operation (updateOne/updateMany/replaceOne), if document does not exist, it will be created.

MongoDB Architecture

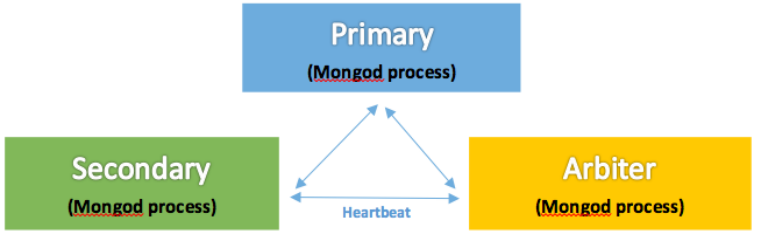
1. Common Architectures

1.1 Standalone

1.2 PSS – Primary Secondary Secondary



1.3 PSA – Primary Secondary Arbiter



Components:

- Primary: Read Write node on Standalone/Cluster
- Secondary: Read Only node on Cluster
- Arbiter: Auxiliary service to support election process during a failover

Replication:

Replication is a practice to keeping identical copies of data on multiple servers to keep applications running and data safe.

Clustering:

A cluster of MongoDB servers implement Master-Slave replication for automated failover.

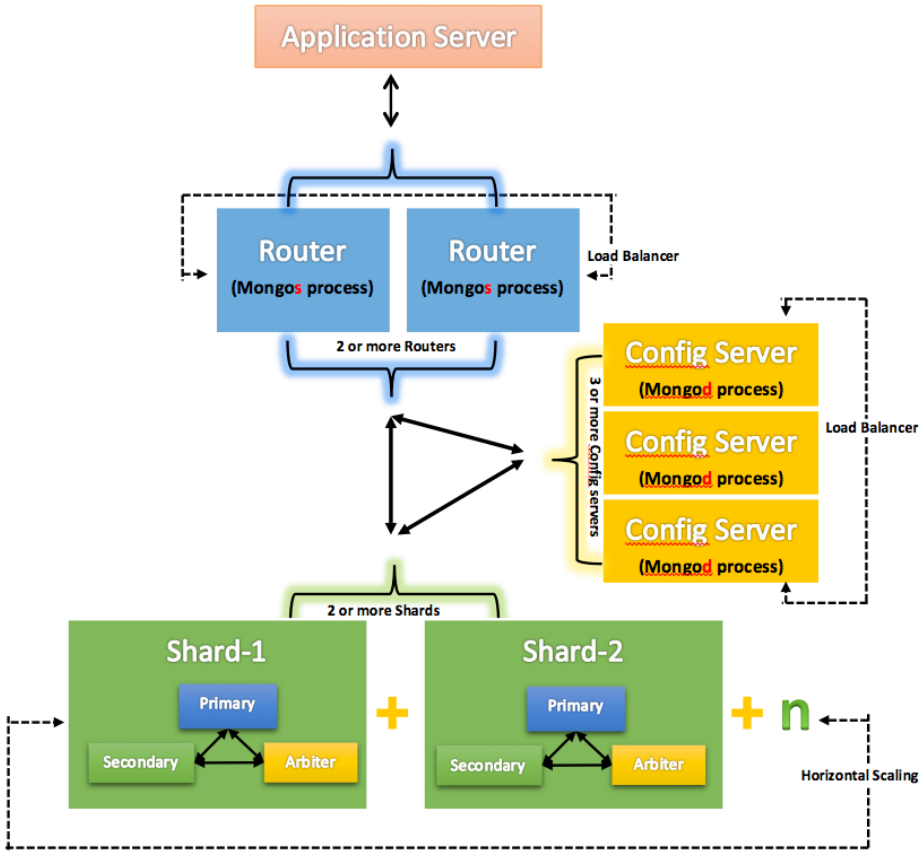
Arbiter:

It is a mongod instance in a ReplicaSet, but it does not have any data.

MongoDB Architecture

1. Common Architectures

1.3 Sharded Cluster



Sharding:

Break up a collection in subsets of data to store them across multiple shards.

Shared Cluster – Router (mongos):

Acts as a gatekeeper for all requests coming from the client.

Configuration Server:

It has metadata and makes the decision of which data should be extracted from which shard.

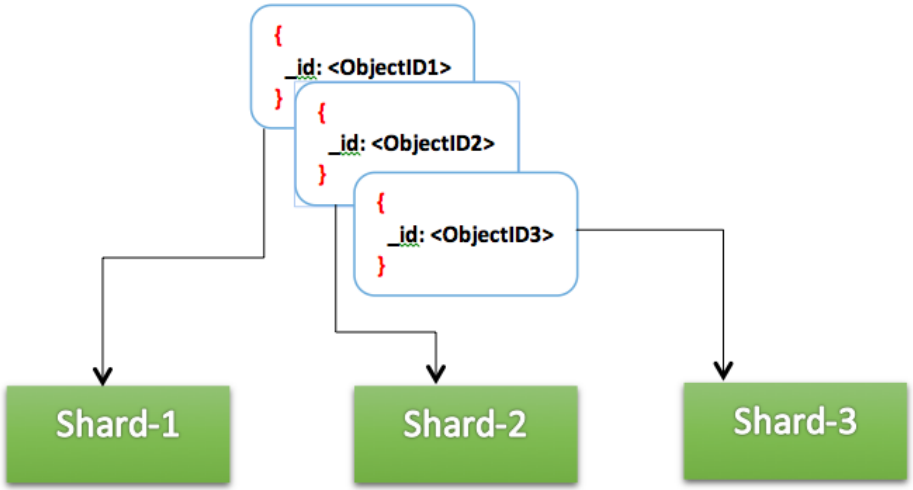
Shard in Sharded Cluster:

Each shard is a replicaSet cluster to hold a portion of data.

MongoDB Architecture

1. Common Architectures

1.3 Sharded Cluster



Shard:

Similar concept of Table Partitioning in Oracle Database, you can split data based on column across different partitions. In MongoDB, it will be split across multiple shards.

Replication:

Duplicates data in multiple secondary servers (similar as we have in Oracle DataGuard) to be used in a critical issue on environment.

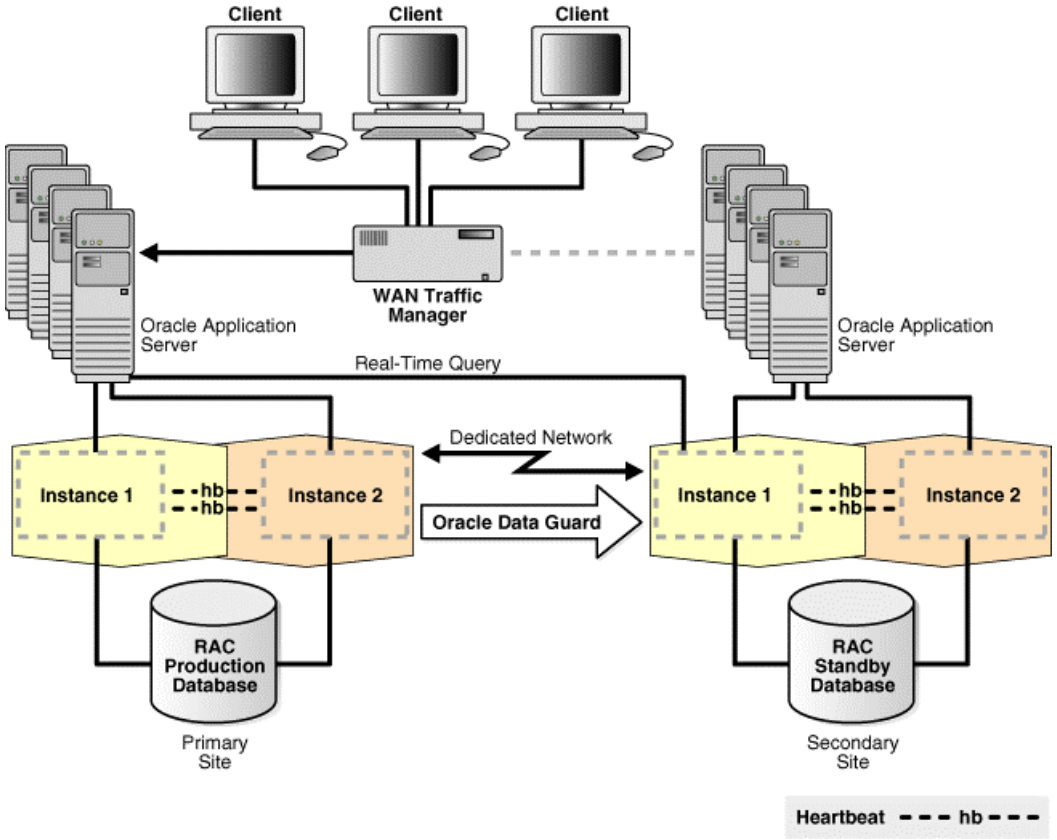
Arbiter:

Similar concept of Arbiter in Oracle of FSFO – Fast-Start-FailOver where there is a service to monitor cluster nodes and participates in election process to define a new primary in case of issues.

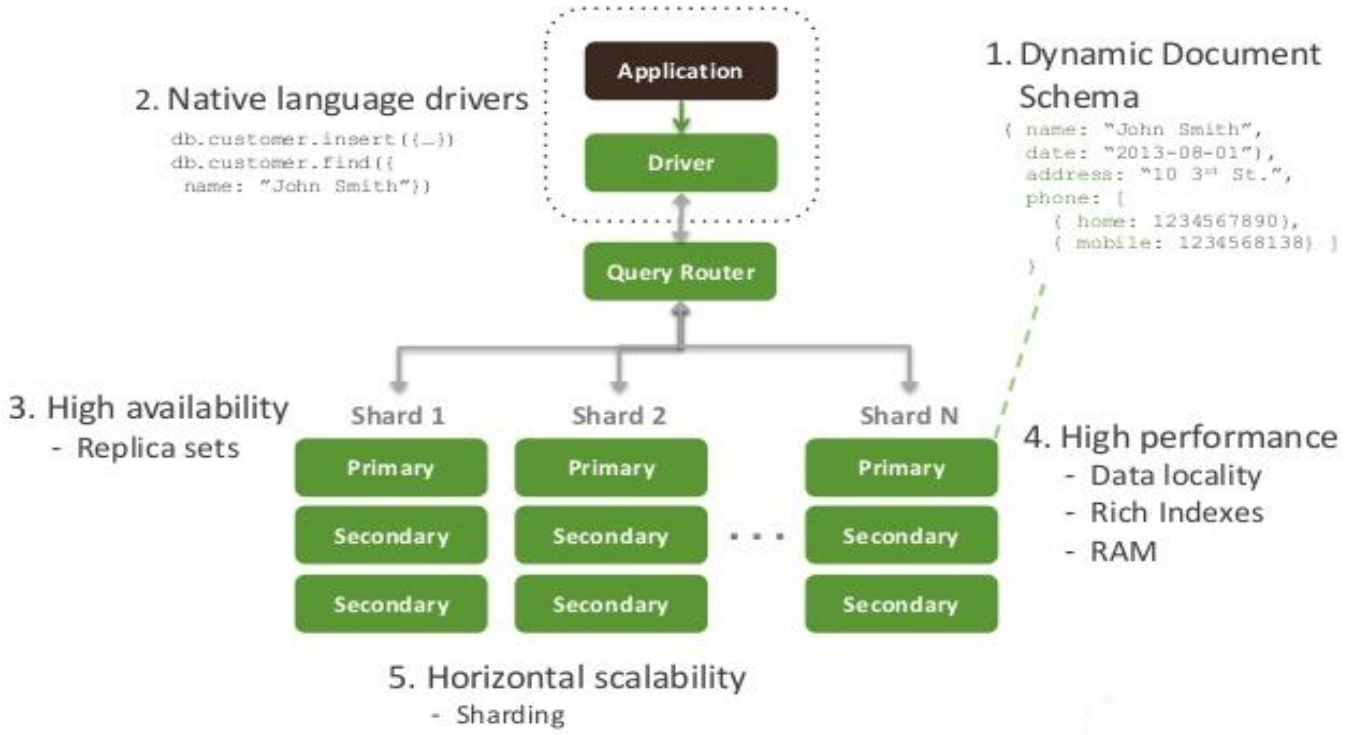
MongoDB Architecture

2. MongoDB Architecture x Oracle Architecture

Sample of Oracle Database Implementation



Sample of MongoDB Implementation



MongoDB Architecture

3. Components Comparison – Oracle x MongoDB

Oracle	MongoDB	Description
SGA	No SGA	MongoDB relies completely on OS Page Cache and Flushing Mechanisms
Buffer Pool	OS Level Page Cache	Controls by OS
Shared Pool	Query Cache	Internalquerycachesize parameter in MongoDB
Spfile/pfile	/etc/mongod.conf	Contains data directory location etc.
Database Server	mongod	Mongod is database instance
Database Client	mongo	Like SQL*Plus Shell
Database Listener	mongos	In a sharded cluster the mongos instance manage all connections
Dictionary	Config Servers – mongod	MongoD instances roles can be Database, configserver, sharded database
Redolog	Journal file	.j_0 files created on journal folder

MongoDB Architecture

3. Components Comparison – Oracle x MongoDB

Oracle	MongoDB
Alertlog	/var/log/mongodb/mongod.log
Shutdown	use admin; db.shutdownServer()
Startup	service mongod start
expdp	mongodump --out <directory>
impdp	mongorestore <directory>
Standby (dataguard)	ReplicaSet
Table Partitioning	Sharding – Split data across multiple nodes
Cluster Services	Arbiter
Storage Engine	mmapv1 (3.0), wiredTiger (3.2), inmemory

MongoDB Tools

1. Cloud

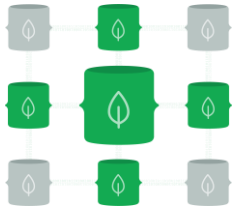
MongoDB Tool Name	Oracle Tool Name	Description
MongoDB Atlas	Oracle Cloud	DBAAS (Database As A Service)

mongoDB® Atlas

ORACLE CLOUD

Always on and durable
Geographically distributed database instances to ensure no single point of failure.

Secure from the start
Databases deployed in Virtual Private Cloud (VPC) to ensure network isolating, IP whitelisting, always-on authentication, encryption at rest and encryption in transit, role-based access management, etc.

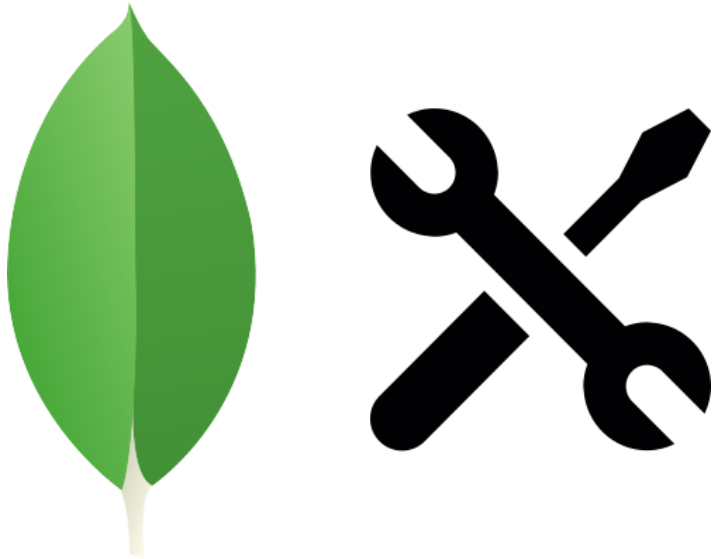


Fully Automated and Elastic
Automated provisioning and setup, patches and minor version upgrades are applied automatically, scalability functionalities, etc.

MongoDB Tools

2. On-Premises

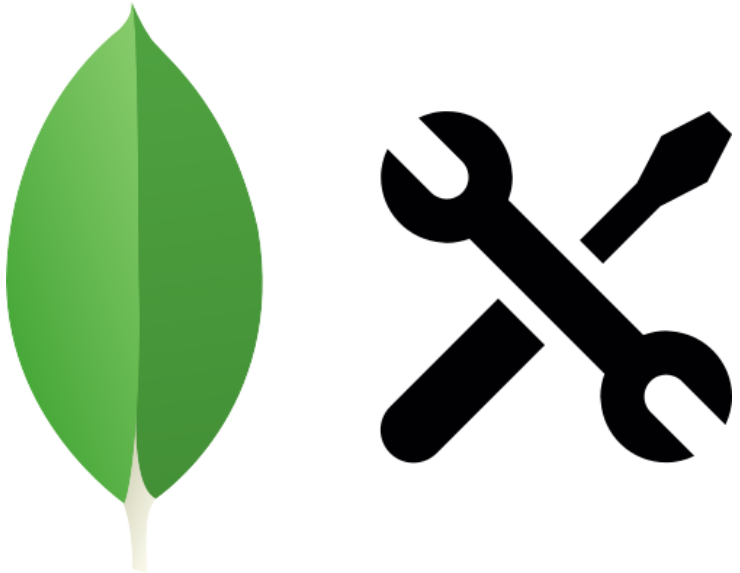
MongoDB Tool Name	Oracle Tool Name	Description
MongoDB Enterprise/Community Server	Oracle Standard/Enterprise/Express/Personal DB	Database
MongoDB OpsManager	Oracle Enterprise Manager	Monitoring Tool
MongoDB Ent./Comm. Kubernetes Operator	Oracle Container Engine for Kubernetes (OCI)	Kubernetes Container
MongoDB Charts	Oracle Analytics / Oracle Data Visualization Desktop	Analyze data and generate charts



MongoDB Tools

3. Tools

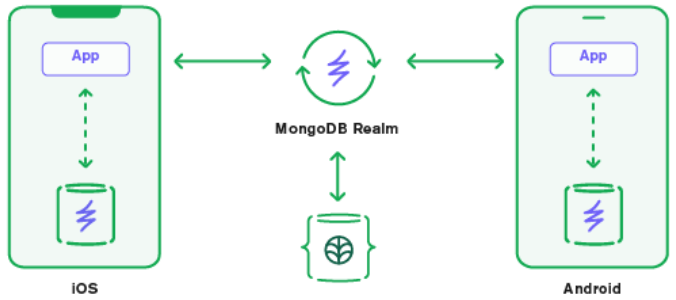
MongoDB Tool Name	Oracle Tool Name	Description
MongoDB Shell	SQLPlus	Command line
MongoDB Compass	SQL Developer or Quest Toad	Graphical IDE
MongoDB Database Tools	Oracle Client	Administrator tools
MongoDB Connector BI	Oracle Instant Client (Libraries and SDKs)	Auxiliary driver to work with BI Tools
MongoDB CLI for Cloud	Oracle Cloud Infrastructure CLI (OCI-CLI)	Cloud



MongoDB Tools

4. Mobile

MongoDB Tool Name	Oracle Tool Name	Description
MongoDB Realm	Oracle Database Mobile Server	Mobile



Sync Data Seamlessly
Build apps that reliably sync between mobile devices and your backend – no matter what platform you're working with.

Ship Key Features
Sync makes simple to keep data up-to-date across multiple users/devices for features/apps that update in real-time.

Build 5-Star Experiences
Realm Database/Ream Sync are optimized for performance, save battery and space.

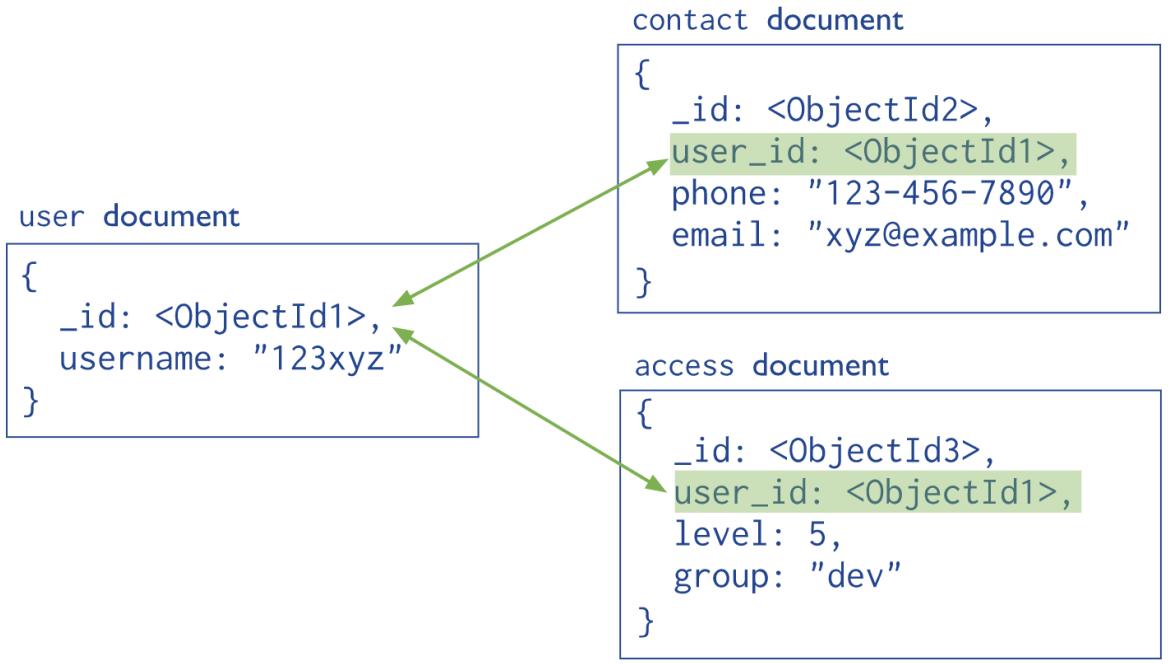
Software Development Advices

1. Embedded Data Models

- Good to use in a one-to-one and one-to-many relationships.
- Advantage:
  - Fewer queries and updates.
  - Better performance for read operations.
- Disadvantage:
  - Large documents can cause performance issues for writes.

Software Development Advices

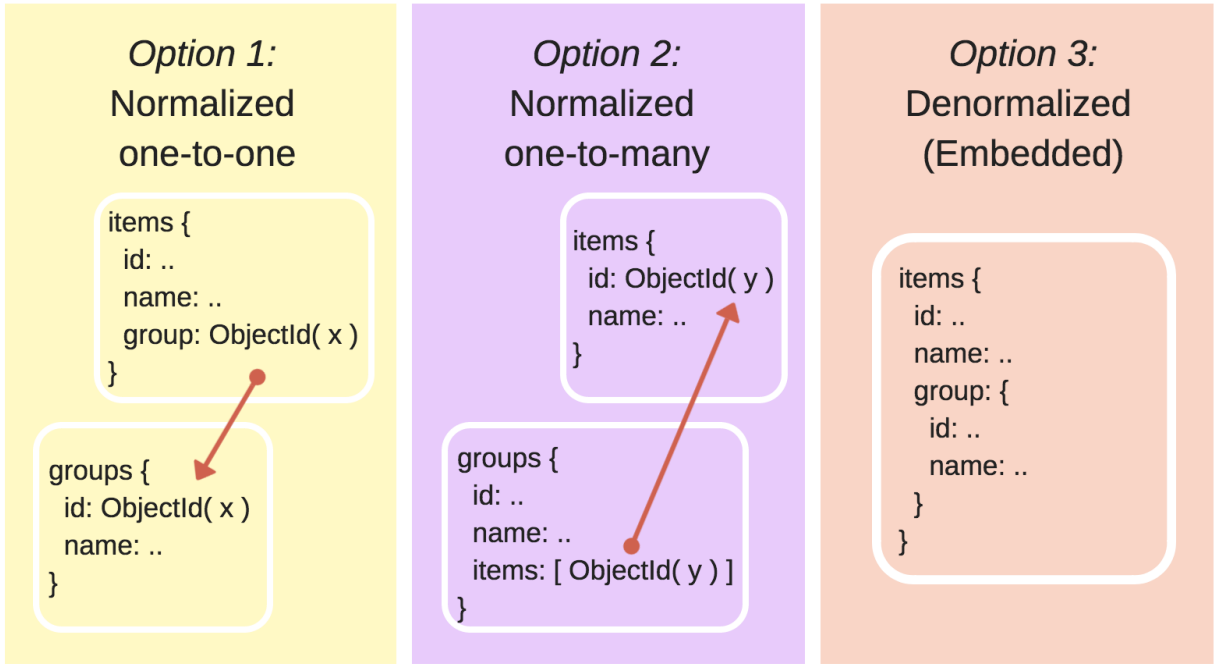
2. Normalized Data Models



- Does not support join.
- Manual reference to _id, Collection Name and sometimes Database name.
- Recommended for common cases.
- Normalized data models are good for using when embedding would result in duplication of data.
- Good for complex many-to-many relations or large hierarchal data sets.

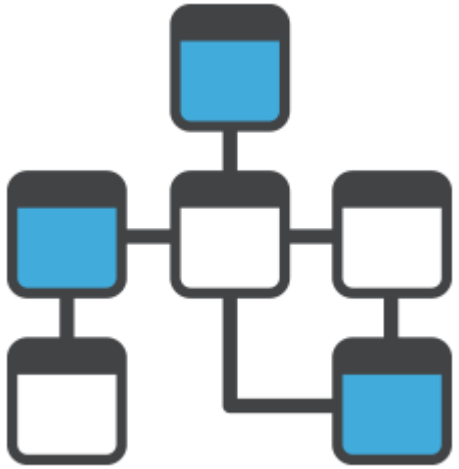
Software Development Advices

3. Restrictions / Concerns



- Interesting to map the operations that we will perform with new data in order to identify better option for data modeling.
- Points to consider:
  - Expected read / write load
  - Expected size of the data
  - Determine which kind of operation has more frequency: read or write
  - Evaluate data load through a POC
- Update Operation:
  - Option 2: It will perform two DB calls to update
  - Option 1 and 3: Only one DB call

4. MongoDB University – Data Modeling Training



M320: Data Modeling

Learn everything you need to know about data modeling for MongoDB.

Link: https://university.mongodb.com/courses/M320/about

SCAN ME

Daniel Coupal

Yulia Genkina

Noberto Leite

5. Validation Rules

You can define business rules to force a specific collection structure based on governance guidelines in your company.

Benefit: Standardize your collection and prevent missing fields or wrong types to be added/modified.

Sample:

Key	Value	Required	Format
name	String	Yes	Not Null
surname	String	Yes	Not Null
email	String	Yes	Valid E-mail Syntax
year_of_birth	Int	No	1900-2019
gender	Char	No	M or F

Software Development Advices

5. Validation Rules

```
db.createCollection( "people" , {
   validator: { $jsonSchema: {
      bsonType: "object",
      required: [ "name", "surname", "email" ],
      properties: {
         name: {
            bsonType: "string",
            description: "required and must be a string" },
         surname: {
            bsonType: "string",
            description: "required and must be a string" },
         email: {
            bsonType: "string",
            pattern: "^.+\@.+$",
            description: "required and must be a valid email address" },
         year_of_birth: {
            bsonType: "int",
            minimum: 1900,
            maximum: 2018,
            description: "the value must be in the range 1900-2018" },
         gender: {
            enum: [ "M", "F" ],
            description: "can be only M or F" }
      }
   }
})
```

Mandatory Fields

Fields Format

Software Development Advices

5. Validation Rules

```
db.runCommand( { collMod: "people",
   validator: { $jsonSchema: {
      bsonType: "object",
      required: ["name", "surname", "email" ],
properties: {
         name: {
            bsonType: "string",
            description: "required and must be a string"
},
         surname: {
            bsonType: "string",
            description: "required and must be a string"
},
         email: {
            bsonType: "string",
            pattern: "^.+\@.+$",
            description: "required and must be a valid
email address" },
         year_of_birth: {
            bsonType: "int",
            minimum: 1900,
            maximum: 2018,
            description: "the value must be in the range
1900-2018" },
         gender: {
            enum: [ "M", "F" ],
            description: "can be only M or F" }
      }
} },
   validationLevel: "moderate",
   validationAction: "warn"
} )
```

Property	Value	Description
validationLevel	off	No validation for Inserts/Updates
	strict	**Default**. Apply to all Inserts/Updates
	moderate	Apply on Inserts/Updates on existing valid documents. Do not apply rules on updates on existing invalid documents.
validationAction	error	**Default**. Documents must pass on validation before writes occurs.
	warn	Just write error message on log, but writes proceed even with validation errors.

5. Validation Rules

Is there some way to review differences among documents in a specific collection ?

Yes. There is a way.

Script: Schema Validator
Link: https://github.com/variety/variety/blob/master/variety.js

What does it do ?
Analyze all documents of a specific collection and generate a report to show column differences.

6. Query Explain

If you need to review query explain, there is a method called explain().
There are the following options:

1) queryPlanner Mode: MongoDB runs the query optimizer to choose the winning plan for the operation under evaluation. cursor.explain() returns the queryPlanner information for the evaluated method.

2) executionStats Mode: MongoDB runs the query optimizer to choose the winning plan, executes the winning plan to completion, and returns statistics describing the execution of the winning plan.

3) allPlansExecution Mode: MongoDB runs the query optimizer to choose the winning plan and executes the winning plan to completion. In "allPlansExecution" mode, MongoDB returns statistics describing the execution of the winning plan as well as statistics for the other candidate plans captured during plan selection.

Syntax:
db.**<collection>**.find({}).explain(**'<MODE>'**);

MongoDB Log

1. Overview about mongod.log

The detail level of these log messages are defined by verbosity.
Possible Log Verbosity Levels:

- 0: Default MongoDB verbosity level. Includes information messages.
- 1-5: Increases the verbosity level to include Debug messages.

How can we get current settings of versobity level?
Command: db.getLogComponents();

How can we get log from Mongo Shell?
Command: db.adminCommand({“getLog”: “global”});

How can we change versobity level?
Command: db.setLogLevel(0);

How can we change versobity level for a specific component?
Command: db.setLogLevel(2, “storage.journal");

MongoDB Log

1. Overview about mongod.log

Log Message Structure:

"2019-03-15T06:36:54.479-0300 I COMMAND [conn1] command admin.$cmd appName: \"MongoDB Shell\" command: setParameter { setParameter: 1.0, logComponentVerbosity: { command: { verbosity: 1.0 } }, $db: \"admin\" } numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r",

It is related to timestamp when the event occurs.

MongoDB Log

1. Overview about mongod.log

Log Message Structure:

"2019-03-15T06:36:54.479-0300 I COMMAND [conn1] command admin.$cmd appName: \"MongoDB Shell\" command: setParameter { setParameter: 1.0, logComponentVerbosity: { command: { verbosity: 1.0 } }, $db: \"admin\" } numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r",

It is related to Severity Level. Follows the list of possible values:
- F - Fatal
- E - Error
- W - Warning
- I - Informational (Verbosity Level 0)
- D – Debug (Verbosity Level 1-5)

MongoDB Log

1. Overview about mongod.log

Log Message Structure:

"2019-03-15T06:36:54.479-0300 I COMMAND [conn1] command admin.$cmd appName: \"MongoDB Shell\" command: setParameter { setParameter: 1.0, logComponentVerbosity: { command: { verbosity: 1.0 } }, $db: \"admin\" } numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r",

It is related to log component: ACCESS, COMMAND, CONTROL, FTDC, GEO, INDEX, NETWORK, QUERY, RELP_HB, ROLLBACK, SHARDING, STORAGE, RECOVERY, JOURNAL.

MongoDB Log

1. Overview about mongod.log

Log Message Structure:

"2019-03-15T06:36:54.479-0300 I COMMAND [conn1] command admin.$cmd appName: \"MongoDB Shell\" command: setParameter { setParameter: 1.0, logComponentVerbosity: { command: { verbosity: 1.0 } }, $db: \"admin\" } numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r",

It is related to connection (session ID) from user/process that executed this statement.

MongoDB Log

1. Overview about mongod.log

Log Message Structure:

"2019-03-15T06:36:54.479-0300 I COMMAND [conn1] command admin.$cmd appName: \"MongoDB Shell\" command: setParameter { setParameter: 1.0, logComponentVerbosity: { command: { verbosity: 1.0 } }, $db: \"admin\" } numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r",

It shows us that command was executed under “admin” database and “$cmd” means that it was a database command.

Note: Full list of possible values for this piece of information can be checked on mongodb documentation.

MongoDB Log

1. Overview about mongod.log

Log Message Structure:

"2019-03-15T06:36:54.479-0300 I COMMAND [conn1] command admin.$cmd appName: \"MongoDB Shell\" command: setParameter { setParameter: 1.0, logComponentVerbosity: { command: { verbosity: 1.0 } }, $db: \"admin\" } numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r",

It is related to which client raises this statement. In this case, it has been executed from MongoDB shell client.

MongoDB Log

1. Overview about mongod.log

Log Message Structure:

"2019-03-15T06:36:54.479-0300 I COMMAND [conn1] command admin.$cmd appName: \"MongoDB Shell\" command: setParameter { setParameter: 1.0, logComponentVerbosity: { command: { verbosity: 1.0 } }, $db: \"admin\" } numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r",

It is related to command that was executed. As we can see on this sample:
- Parameter change;
- It has been changed verbosity level to 1 on “command” component;
- Affected database: admin.

Executed command on MongoDB Shell: db.setLogLevel(1,”command”);

MongoDB Log

1. Overview about mongod.log

Log Message Structure:

"2019-03-15T06:36:54.479-0300 I COMMAND [conn1] command admin.$cmd appName: \"MongoDB Shell\" command: setParameter { setParameter: 1.0, logComponentVerbosity: { command: { verbosity: 1.0 } }, $db: \"admin\" } numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r",

It is related to metadata about executed operation. Last piece of this metadata is related to how much time it took to be executed:

```
numYields:0 reslen:644 locks:{} protocol:op_msg 0ms\r
```

In this case, it has been executed on 0 milliseconds.

Q&A

Reference

- MongoDB University: https://university.mongodb.com/
- MongoDB Download Center: https://www.mongodb.com/try/download/community
- Schema Validator: https://github.com/variety/variety/blob/master/variety.js
- Robo3T: https://robomongo.org/
- NoSQLBooster: https://nosqlbooster.com/
- Dockerhub - Mongo: https://hub.docker.com/_/mongo
- Docker Labs: https://labs.play-with-docker.com/