
Table of Contents

Introduction	1.1
MongoDB 介绍	1.1.1
MongoDB CURD 操作	1.1.2
MongoDB 索引	1.1.3
副本集	1.1.4

MongoDB 简单介绍

简要介绍 MongoDB , 安装, 基本操作, 适用场景, 常用架构模型

MongoDB 是什么？

- MongoDB 将数据储存在灵活，类似JSON(BSON) 的文档中，这意味着字段可以随着文档的不同而变化，数据结构可以随着时间的推移而改变。
- 文档模型映射到应用程序代码中的对象，使数据易于实用
- MongoDB 是一个分布式数据库的核心，因此，高可用性，水平缩放，地理分布是内置的，易于实用
- MongoDB 是自由和开源的，出版根据GNU Affero 通用公共许可证

什么是 BSON

BSON是一种类json的一种二进制形式的存储格式，简称Binary JSON，它和JSON一样，支持内嵌的文档对象和数组对象，但是BSON有JSON没有的一些数据类型，如Date和BinData类型

MongoDB 优劣

优势：

- 高可用，支持自动故障恢复（副本集）
- 比关系型数据库更高的写入负载
- 支持大容量的存储，内置 Sharding 分片简单
- 海量数据下，性能优越
- 支持地理位置查询

缺点

- 不支持事务操作
- 占用空间大
- 不支持关联表查询
- 自由灵活的文件存储格式带来的数据错误

MongoDB 适用场景

频繁的写入操作，不可靠环境保证高可用性，数据规模大(如预估会超过5G的数据量)，数据结构变更频繁。如(爬虫，日志记录，监控记录等)

MongoDB 安装

- 下载安装 MongoDB <https://www.mongodb.com/download-center?jmp=nav#community>

MongoDB 主要程序介绍

- mongod : MongoDB 服务器

- mongo : 进入 MongoDB 命令行
- mongodump : MongoDB 数据库备份 (导出格式为 .bson)
- mongorestore: MongoDB 数据库恢复
- mongoexport : MongoDB 数据导出 (导出格式为: .json .csv)
- mongoimport : MongoDB 数据导入 (导入格式为: CSV, TSV or JSON)
- mongo

MongoDB CURD 操作

• find

- 查询一条记录: db.collection.findOne()

```
db.test.findOne({"a":"b"})
```

- 查询多条记录: db.collection.find()

```
db.test.find({})
```

```
db.collection.find(  
  <query>  
  <projection>  
)
```

查询参数说明:

- query

类型: document , 查询条件

- projection

类型: document, 指定返回的字段 1 返回 0 不返回 如: {"a":1,"b":0} , 当设定该 projection 时 查询结构的文档中将返回 1

• insert

- 插入一条记录: db.collection.insertOne()

```
db.test.insertOne({"a":"b"})
```

- 插入多条记录: db.collection.insertMany()

```
db.test.insertMany([{"a":"b"}, {"c":"d"}])
```

- 插入一条或多条记录: db.collection.insert()

```
db.collection.insert(  
  <document or array of documents>,  
  {  
    writeConcern: <document>,  
    ordered: <boolean>  
  }  
)
```

writeConcern : 写入策略

```
{w: <value>, j: <boolean> , wtimeout : <number>}
```

W Option

writeConcern 说明

- 1

应答式写入, 要求已经传播到指定的单个实例或副本集主实例 (默认值)

- 0

非应答式写入，不返回任何响应，无法知道写入结果，相对应的式更高更快的写入速度

- majority

适用于集群架构中，要求写入操作已经传递到绝大多数投票节点以及主节点进行应答

- \

要求写入操作已经传递到指定tag标记副本集中的成员后进行应答

j Option

j : 该选项要求确认写操作已经写入journal日志之后应答客户端

```
true or false
```

wtimeout

写入超时时间设定，单位为 ms，防止写操作无限制被阻塞 导致无法应答客户端

• update

- 更新一条记录: db.collection.updateOne()

```
db.collection.updateOne(
  <filter>,
  <update>,
  {
    upsert: <boolean>,
    writeConcern: <document>,
    collation: <document>,
    arrayFilters: [ <filterdocument1>, ... ]
  }
)
```

- 更新多条记录: db.collection.updateMany()

```
db.collection.updateMany(
  <filter>,
  <update>,
  {
    upsert: <boolean>,
    writeConcern: <document>,
    collation: <document>,
    arrayFilters: [ <filterdocument1>, ... ]
  }
)
```

- 替换一条记录: db.collection.replaceOne()

```
db.collection.replaceOne(
  <filter>,
  <replacement>,
  {
    upsert: <boolean>,
    writeConcern: <document>,
    collation: <document>
  }
)
```

- 更新一条或多条记录: db.collection.update()

```
db.collection.update(
  <query>,
  <update>,
  {
    upsert: <boolean>,
    multi: <boolean>,
    writeConcern: <document>,
    collation: <document>,
    arrayFilters: [ <filterdocument1>, ... ]
  }
)
```

update 参数说明

- query

类型: document, 更新的选择标准。可以使用与find () 方法相同的查询选择器。

- update

类型: document, 要应用的修改

- upsert

类型: boolean , true 文档中没有则新建并插入, 文档中存在则更新

- multi

类型: boolean, true 更新多条, false 更新一条

- writeConcern

类型: document , 写入安全策略

- arrayFilters

类型: document, 筛选器 , 条件筛选, 如:

```
[ { "x.a": { $gt: 85 } }, { "x.b": { $gt: 80 } } ]
```

• **delete**

- 删除一条记录

```
db.collection.deleteOne()
```

- 删除多条记录

```
db.collection.deleteMany()
```

- 删除一条或多条记录

```
db.collection.remove()
```

```
db.collection.remove(
  < query >
  < options >
)
```

query

查询条件, 要删除那些文档, 当用 {} 表示时, 将匹配所有文档

options

- **justOne**

 | boolean 是否仅删除一条记录, true 只删除一条文档, false 匹配文档全部删除

- **writeConcern**

 | 写入安全策略

MongoDB 索引

`_id` 为所有文档的默认索引，所有文档都含有该索引

- **创建索引**

```
db.collection.createIndex(  
    <key and index type specification>, <options>  
)
```

3。0 版本后 `ensureIndex` 被放弃，被设定为 `createIndex` 别名

```
db.collection.createIndex( {  
    < keys >,  
    < options >  
})
```

字段 `key` 对应 `value` 可取 值 1 或 -1，1 表示索引默认升序排序，-1 表示索引默认降序排序

options :

- `background`

类型：boolean, 指定返回的字段 1 返回 0 不返回 如：{"a":1,"b":0}，当设定该 projection 时 查询结构的文档中将返回 1

- `unique`

类型：boolean, true, 创建唯一索引，该选项对哈希索引不可用

- `name`

类型：string, 索引名称，如未指定,MongoDB串联索引字段的名称和排序顺序生成索引名称

- `partialFilterExpression`

类型：document, 部分索引，可指定仅索引特条件下的文本，其它则不索引,3.2 以后的版本才有该属性

- `sparse`

类型：boolean, true, 稀疏索引，索引只引用具有指定字段的文档，3.2 之后的版本应当优先使用部分索引

- `expireAfterSeconds`

类型：int , TTL 索引，指定一个整数值(以秒为单位)，控制MongoDB集合中文档的存活时间

- `storageEngine`

类型：document, 创建索引用户自主配置存储引擎

- **索引类型**

- 单个字段索引

索引建立在单一字段上 如：

```
db.collection.createIndex({ name : -1 })
```

- 复合索引

多个字段联合建立索引 如:

```
db.collection.createIndex({ userid : 1, score : -1 })
```

- Multikey

multikey 索引用来索引存储在数组中的内容，如果为保存数组值的字段编制索引，MongoDB 将为数组的每个元素创建单独的索引项 如：

```
存在文档：
{
  "name" : "xyz",
  "addr" : [
    { "zip" : "10036" },
    { "rar" : "30005" }
  ]
}
若要为 addr 数组中的 zip 字段建立索引则应：
db.collection.createIndex( { "addr.zip" : 1 } )
```

此外 MongoDB 还支持地理空间索引，文本索引，哈希索引，详情可产看[MongoDB 文档](#)

• 索引属性

- 唯一索引

唯一索引拒绝索引字段的重复值，插入更新集合时，MongoDB 将检查是否满足唯一条件，否则插入更新失败，创建方法如下，设置unique 为true 即可

```
db.collection.createIndex({ "user_name" : 1,{ unique : true } })
```

- 部分索引

部分索引仅索引满足条件的文档。部分索引具有较低的存储需求，降低了索引创建和维护的性能成本，如：

```
db.restaurants.createIndex(
  { cuisine: 1, name: 1 },
  { partialFilterExpression: { rating: { $gt: 5 } } }
)
```

该语句创建了一个复合索引的部分索引，仅当文档中的rating 值大于等于5时，才联合索引 cuisine , name 字段

- 稀疏索引

稀疏索引仅包含具有索引字段的文档的条目，3.2 之后版本优先使用部分索引，创建时将options 参数sparse 设定为true 即可，如：

```
db.addresses.createIndex(
  { "xmpp_id": 1 }, { sparse: true }
)
```

- TTL索引

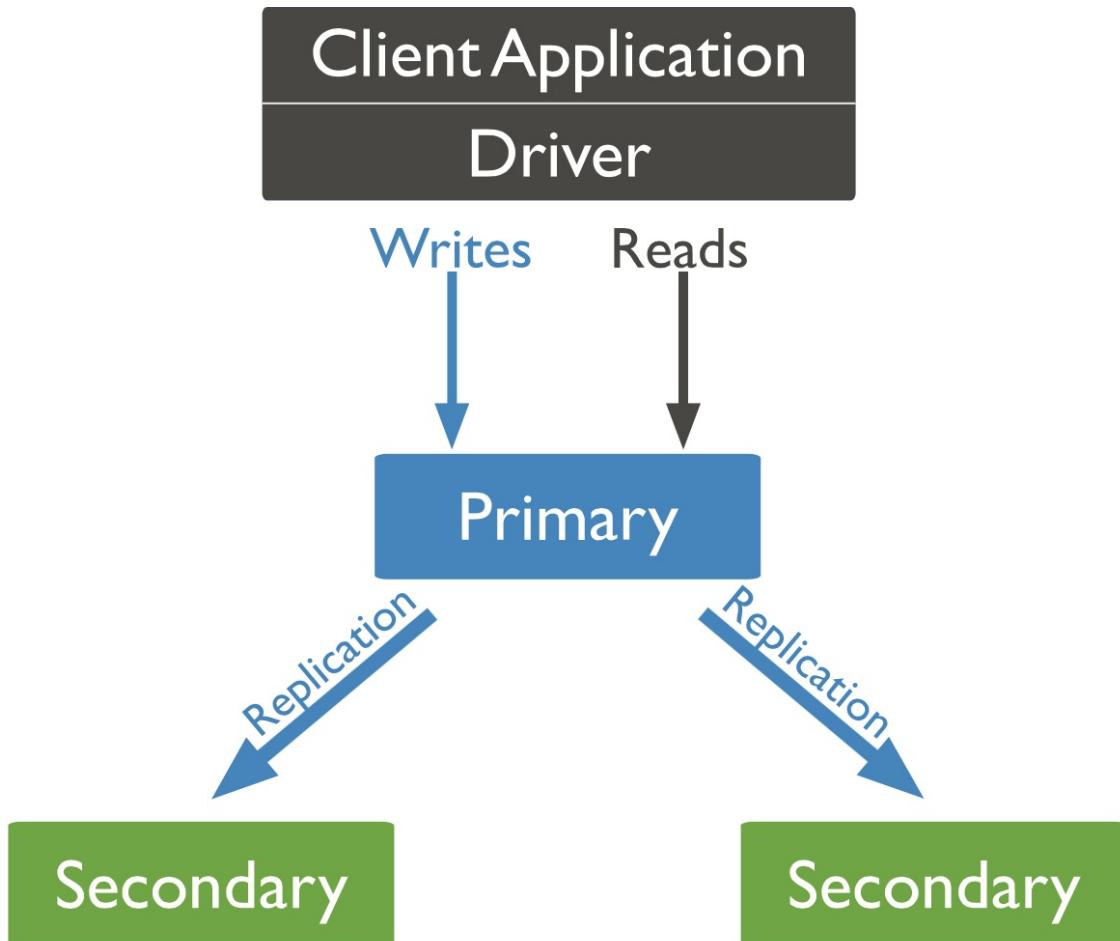
TTL索引可控制文档在集合中的存活时间，当索引时间存活时间超过TTL设定时间时，MongoDB将自动删除过期的索引与相对应的文档

MongoDB 副本集

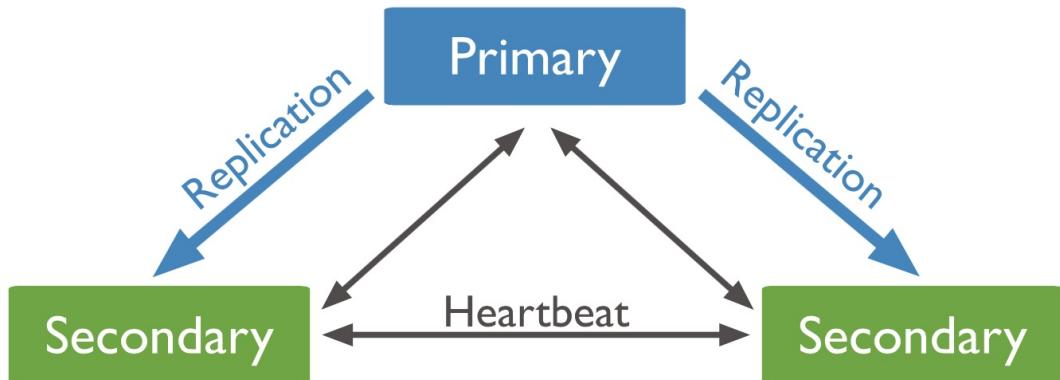
MongoDB 中的副本集是一组维护同一数据集的 mongod 进程。副本集提供冗余和高可用性，是所有生产部署的基础

- **MongoDB 中的复制**

副本集是一组维护相同数据集的 mongod 实例。副本集包含多个数据轴承节点和可选的一个仲裁节点。在数据轴承节点中，只有一个成员被视为主节点，而其他节点则被视为副本节点



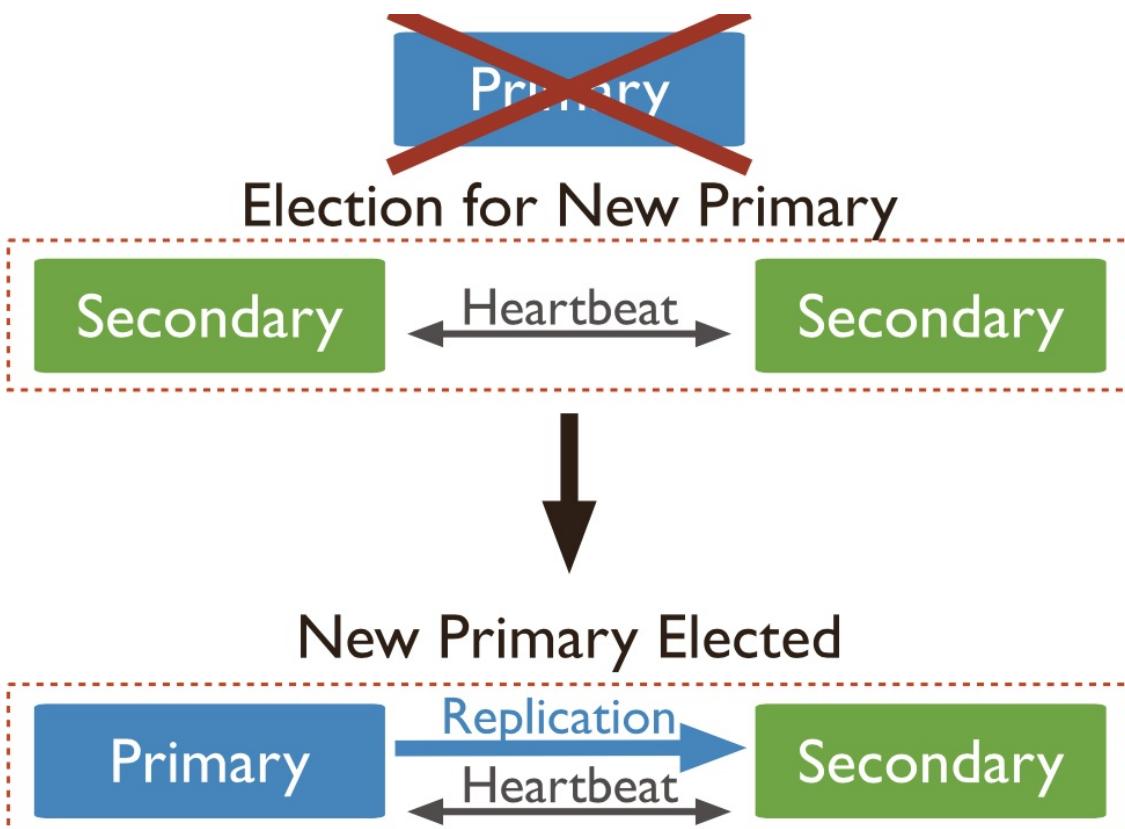
主节点接收所有的写入操作，副本集只能有一个主功能。主要记录对它的数据集的所有变动在它的操作日志，即 oplog，副本负责复制主 oplog，并将这些操作应用到其它数据集，以便副本的数据集反映主数据集，副本集实例相互保持心跳，当检验到某个mongod 实例心跳停止时，自动摘除该mongod 实例，如主节点被摘除，则进入仲裁，选举，自动切换主节点



可以将额外的 mongod 实例添加到副本集作为仲裁者，仲裁人不维护数据集，仲裁者的目的是通过响应其他副本集成员的心跳和选举请求来维护副本集中的仲裁。因为它们不存储数据集，所以仲裁员可以提供比具有数据集的完全功能的副本集成员更便宜的资源成本，从而为复制集仲裁功能带来一个很好的方法。如果您的副本集有偶数的成员，则添加仲裁人以在初选的选举中获得多数选票。

- 故障自动切换

当主不与设置的其他成员通信超过配置的electionTimeoutMillis期间（默认为10秒）时，合格的次要要求选举提名自己为新主。该群集试图完成新的初选和恢复正常操作的选举



如果默认的 replica configuration settings，则在群集选择新主项之前的中间时间通常不应超过12秒。这包括将主标记为不可用并调用并完成选举所需的时间。通过修改 settings.electionTimeoutMillis 复制配置选项来调整这个时间段。诸如网络延迟等因素可能会延长复制副本集选举所需的时间，这反过来会影响群集在没有主计算机的情况下运行的时间量。这些因素取决于特定群集体系结构

将 electionTimeoutMillis 复制配置选项从默认的 10000 (10 秒) 中降低会导致对主故障的快速检测。但是，群集可能会更频繁地调用选举，原因是临时网络延迟，即使主要是健康的。这会导致 w: 1 写操作增加回滚。

• 副本集成员分类

- 主成员

主是副本集中唯一接收写操作的成员，主接收所有写操作，然后次复制 oplog 应用与它们的数据集，副本集的所有成员都可以接收读取操作，但是默认情况下，应用程序将读取操作定向到主，主再分发读取请求到其它成员

- 副本成员

- 优先级为 0 的副本成员

它们可以正常辅助，维护数据集的副本，接受读取请求，并在选举中投票，但是 priority 0 (即该成员) 不能成为初选成员，不参与选举，即它们永远不会成为主

- 隐藏成员

隐藏成员必须是优先级为 0 的副本成员，同时，它们维护数据集但不接受读取请求，隐藏成员不会接收除了以外其它来的任何通信

- 延迟成员

延迟成员必须是隐藏成员，它们的数据集反应了副本集中的数据集的早期或延迟状态，例如某延迟成员被设定延迟半小时，则该成员的数据集是主数据集半小时以前的数据集。所以它是数九的滚动备份，或运行的历史快照。

- 仲裁成员

仲裁成员仅在选举时投票，不参与数据集的维护

• 副本集体系结构部署

- 部署策略

- 确定成员数量

根据mongodb设定的策略，确定成员分配

- 投票成员的最大数目

副本集最多可以有 50 个 members，但只有 7 个投票成员。如果副本集已有 7 个投票成员，则其他成员必须为非投票成员

- 部署奇数个投票成员

确保副本集的投票成员数目为奇数，否则将需要部署一个仲裁者

- 为专用使用隐藏或延迟成员

添加隐藏或延迟成员以支持专用功能，如备份或报告。

- 标记副本集成员

为副本集成员打 tag，副本集允许使用副本集标记将对特定成员的读取操作设置为目标

- 使用日志记录防止电源断电

MongoDB 默认启用日志记录，日志记录可防止出现服务中断时的数据丢失

