# MongoDB（4.4）

- MongoDB的CURD
  - MongoDB特点：
    - 默认端口：27017
    - 密码：mongodb的用户名和密码是基于特定数据库的，而不是基于整个系统的。
    - 数据格式
      - DB：相当于一个库
      - Collection：相当于表
      - Documents: 相当于表的行
        - 数据结构：和JSON基本一样. 所有存储在集合中的数据都是BSON格式
  - 插入文档
    - 特点：
      - 1. 如果该集合不存在，则插入操作将创建该集合
      - 2. 每个文档都需要一个唯一的_id字段作为主键（未指定会自动生成）
    - 操作类型：
      - insertOne：单个文档插入集合
      - inserMany：多个文档插入集合
      - insert：将单个文档或多个文档插入到集合中
  - 更新文档
    - 特点：
      - 1. 更新操作符（例如$set）来修改字段值
      - 2. 未查询到符合条件的数据, 则不做任何操作
      - 3. $set的字段不存在, 则添加该字段
    - 操作类型：
      - updateOne()：更新符合条件的第一个文档
      - updateMany()：更新多个文档
      - update：更新单个/多个文档

        upsert：这个参数的意思是，如果不存在update的记录，是否插入objNew,true为插入，默认是false，不插入。 multi：mongodb默认是false,只更新找到的第一条记录，如果这个参数为true,就把按条件查出来多条记录全部更新
      - replaceOne()：更换文档

        要替换_id字段以外的文档的全部内容，将一个全新的文档作为第二个参数传递
  - 删除文档
    - 特点：
      - 清空文档: db.collection.deleteMany({})
    - 操作类型：
      - deleteOne：仅删除一个符合条件的文档
      - deleteMany：删除所有符合条件的文档
  - 查询文档
    - 特点：
      - 1. 等值查询：使用<field>： <value>表达式
      - 2. 查询操作符：{ <field1>: { <operator1>: <value1> }, ... }
    - 操作类型：
      - find: 返回符合条件的所有数据
      - findOne： 返回满足条件的第一条数据
    - 高级查询
      - limit：取指定数量的结果集
      - skip：跳过指定的个数 等同于offset
      - sort：排序，-1是降序，值为1是升序
      - count：符合条件的结果集总数
- MongoDB索引
  - 默认索引：_id字段
  - 顺序：1: 升序 -1：降序
  - 索引的种类：
    - 单字段索引：
    - 复合索引：
    - Hash索引：
    - 多键索引：
    - 全文索引：
- MongoDB事务
  - 事务的引入
    - 4.0版本开始支持
  - 事务的特性
    - 多表事务操作仅支持在副本集上运行, 单机不支持
    - 多表事务不能用于sharding 集群
    - 仅仅支持CRUD操作，其他操作不能再事务中出现
    - 一个会话同一时刻只能开启一个事务操作
    - 对事务的大小的限制在 16MB
    - 对事务的操作整体不允许超过60秒
  - 事务中的函数
    - 开始事务：Session.startTransaction()
    - 提交事务：Session.commitTransaction()
    - 中止事务并回滚: Session.abortTransaction()
- MongoDB副本集（高可用）

  搭建副本集是为了实现mongodb高可用
  - 数据备份
    - 同步原理
      - 备份节点异步的从主节点同步oplog

        主节点负责处理客户端的读写请求，备份节点则负责映射主节点的数据 主节点上的所有数据库状态改变的操作（oplog），都会存放在一张特定的系统表中。备份节点异步的从主节点同步oplog，然后重新执行它记录的操作，以此达到了数据同步的作用。 Oplog注意点： Oplog的大小是固定的，当集合被填满的时候，新的插入的文档会覆盖老的文档。
    - 同步方式
      - intial sync: 全量同步（单线程复制）

        从节点出现如下状况时，需要先进行全量同步： 1. oplog为空（新节点加入，无任何oplog）2. _initialSyncFlag字段设置为true initial sync开始时，会主动将_initialSyncFlag字段设置为true，正常结束后再设置为false；如果节点重启时，发现_initialSyncFlag为true，说明上次全量同步中途失败了，此时应该重新进行initial sync 3. 发送resync命令 当用户发送resync命令时，initialSyncRequested会设置为true，此时会重新开始一次initial sync
      - replication: 增量同步
  - 读写分离
    - 从节点中是不允许执行写操作的
  - 自动故障转移
    - 心跳机制(Hearbeat)
      - 2s发送一次心跳信息

        复制集成员间默认每2s会发送一次心跳信息，如果10s未收到某个节点的心跳，则认为该节点已宕机不可以访问;
    - 选举原理
      - "大多数"和"一票否决"

        如果Primary连接中断超过10s，其他节点会自动选举出一个Primary节点，负责响应客户端的请求，实现数据的自动故障转移。 选举Primary成员时，使用"大多数"和"一票否决"原则。在Replica Set中，每个成员只能要求自己被选举为Primary节点。当一个Secondary成员无法与Primary成员连通时，该成员就会发起选举，请求其他成员将自己选举为Primary成员， 只有得到"大多数"成员的支持，该成员才能被选举为Primary成员；只要有一个成员否决，选举就会取消。
    - 相关节点
      - 主节点： 负责客户端的读写请求
      - 备份节点： 负责映射主节点的数据
      - 仲裁节点（可选）： 不存储数据

        客户端不需要连接此节点， 能决定哪一个备节点在主节点挂掉之后提升为主节点 副本集应该确保具有奇数个投票成员，如果您拥有偶数个投票成员，请部署仲裁节点，以便该集合具有奇数个投票成员。 仲裁节点不存储数据的副本并且需要更少的资源
      - 投票节点: 最多有7个成员可以投票

        不是每一个成员都有投票选举的权利，在一个Replica Set中，最多有7个成员用于投票选举的权利， Primary成员是由这7个成员选举出来的。有投票权利的成员，其属性："votes" 是1；若为0，表示该成员没有投票权利。
- MongoDB分片集群
  - 分片集群的作用
    - 保证可扩展性
  - 何时使用分片集群
    - 存储容量需求超出单机磁盘容量
    - 活跃的数据集超出单机内存容量

      导致很多请求都要从磁盘读取数据，影响性能
  - 分片集群组件
    - shard：每个分片是整体数据的一部分子集且可以部署为副本集
    - mongos：充当查询路由器，将读写请求指引到合适的分片上
    - config servers：存储了分片群集的配置信息，包括chunk 信息

      从 MongoDB 3.4 开始，必须将配置服务器部署为副本集
  - 数据分片策略
    - 基于Hashed的分片
      - 分片原理
        - 计算分片键字段值的哈希值，用以确定该文档存于哪个chunk
      - 优点
        - 数据整体分布比较均匀，对于等值查询效率很高
      - 缺点
        - 相邻的数据通常不在同一个数据块上， 范围查询效率低
    - 基于Ranged的分片
      - 分片原理
        - 将相似的值放到一个chunk中
      - 优点
        - 范围查询效率高
      - 缺点
        - 数据分布不均匀
  - 数据块管理
    - 块拆分：当块超过指定大小时，就会进行块拆分

      无论采用何种分片策略，数据最终都被存储到对应范围的数据块 (chunk) 上 每个块默认的大小都是64M。由于数据源源不断的加入，当块超过指定大小时，就会进行块拆分 config servers 更新关于块的元数据信息。
    - 块迁移：当某个分片服务器上的数据块过多时候,就会发生块迁移

      当某个分片服务器上的数据块过多时候， 就会发生块迁移，将块从一个分片迁移到另外一个分片。 块迁移是由在后台运行的平衡器 (balancer) 所负责的， 它在后台进行持续监控， 如果最大和最小分片之间的块数量差异超过迁移阈值，平衡器则开始在群集中迁移块以确保数据的均匀分布。
    - 数据查询：块的迁移不会对查询造成任何影响

      块的迁移不会对查询造成任何影响。 查询需要先经过mongos ， mongos会从config servers 上获取块的位置信息和数据范围，然后按照这些信息进行匹配后再路由到正确的分片上。
- MongoDB聚合
  - 作用：用于复杂查询

    Find命令只适合用于单表查询 聚合查询用于复杂查询、多集合连接查询以及数据分析和计算(诸如统计平均值,求和等)
  - 操作对比

    MongoDB聚合操作 MySql操作/函数 $project select $match where $group group by $sort order by $limit limit $sum sum() $lookup join