注意事项：

条件参数为空对象表示全部匹配

1. 数据库分类：
   1. 关系型数据库（RDBMS）如MySQL、Oracle、DB2、SQL Server... 关系数据库中都是表
   2. 非关系型数据库（No【Not Only】 SQL【结构化操作语言】）：键值对数据库。如：MongoDB、Redis... MongoDB为文档数据库
2. MongoDB是为快速开发互联网Web应用而设计的数据库系统。MongoDB中存储的是各种各样的BSON（类似JSON）
3. 基本使用：
   1. 安装MongoDB： 安装软件、配置环境变量，在c盘根目录创建一个文件夹 data，在data中创建一个文件夹db（作为默认数据库）
   2. 启动MongoDB：打开cmd命令行窗口，输入 mongod启动mongoDB服务器
      1. 32位系统在第一次启动服务器时，需要输入mongod --storageEngine=mmapv1
      2. 通过指定路径启动服务器：mongod --dbpath 数据库路径 --port 端口号
   3. 连接MongoDB：打开一个cmd窗口，输入mongo连接mongoDB，出现 >
4. 基本概念：
   1. 数据库（database）：数据库的服务器用来保存数据，通过mongod命令启动；客户端用来操作服务器，对数据进行增删改查的操作，通过mongo
   2. - 命令启动；
   3. 集合（collection）：在数据库内；Mongoose会自动将集合名转换为复数
   4. 文档（document）：对象或数组形式，在集合内。文档的属性值也可以是一个文档，称内部的文档为内嵌文档
      1. 文档之间的关系：
         1. 一对一：如夫妻。通过内嵌文档的形式体现
         2. 一对多/多对一：学生/老师。可以通过内嵌文档的形式体现，或者插入一个字段，值为对应文档 \_id 的值
         3. 多对多：分类对商品。插入一个字段，值为数组，内包含对应文档 \_id 的值
   5. 数据库和集合无需手动创建
5. MongoDB Manager的控制台命令：
   1. 基本指令：
      1. show dbs / databases 显示当前的所有数据库 use 数据库名 进入到指定的数据库中
      2. db 变量，代表当前所处的数据库 show collections 显示数据库中所有的集合
      3. db.集合名.drop（） 删除集合 db.dropDatabase（） 删除数据库
   2. 数据库的CRUD（增删改查）操作：
      1. 插入文档：db.集合名.insert（doc / [doc, doc...]） 若需同时传入多个，可以传入一个文档数组
         1. 在向集合中插入文档时，若没有给文档指定“\_id”属性，则数据库会自动为文档添加“\_id”属性作为文档的唯一标识符
         2. db.集合名.insertOne（xx）/ db.集合名.insertMany（xx） 功能同insert，仅作语义区别（v3.2+）
      2. 查找文档：db.集合名.find（条件参数对象[，投影]） 查询集合中所有符合条件的文档，不传参则显示所有文档，返回数组
         1. 条件参数对象：如： ｛字段名：值，...｝ 当字段名=值或字段数组中包含值时匹配
            1. 查询操作符：｛字段名：｛$gt：值1，$lt：值2｝｝ 代替条件参数中的值，匹配符合条件的文档

$eq 等于 $gt大于 $gte大于等于 $lt小于 $lte小于等于 $ne不等于

$in：[xx,xx] 能与数组中元素匹配的

$or：[{字段名1：{$lt：1000}}，｛字段名1：｛$gt：2500｝｝] 逻辑或

* + - 1. 投影：｛字段名：1/0｝ 结果中显示/不显示指定字段
      2. db.集合名.findOne（） 查询集合中符合条件的第一个文档，返回文档对象
      3. find返回的数组可以使用count（）和length（）方法返回数组长度
      4. 如果要查询内嵌文档，可通过 . 的形式匹配，此时字段名必须加“”，如｛“字段名.内嵌文档”：“值”｝
      5. 方法：
         1. find（）.limit（整数n） 只查询前n条数据
         2. find（）.skip（整数n） 跳过前n条数据
         3. find（）.sort（排序规则对象） 指定结果的排序规则，

排序规则对象：｛字段名：1/-1，。。。｝ 指定字段名升序/降序

* + - * 1. MongoDB会自动调整skip、limit、sort的位置，连用时自动先执行skip、再执行limit，再执行sort
    1. 修改文档：db.集合名.update（条件参数，新对象[，配置选项]） 默认将新对象替换查询到的第一个对象
       1. 修改操作符：{$set：｛字段名：值，...｝...} 将以上作为新对象传入，只修改指定字段
          1. $unset删除字段，值随意取 $push添加元素到数组 $addToSet添加元素到数组，若数组中已存在，则不添加
          2. $inc字段自增指定值 $pull从指定字段（数组）中删除匹配的元素
       2. 配置选项：一个对象，如｛multi：布尔｝ 配置是否修改多个
       3. updateMany（）/ updateOne（） 修改多个/修改一个
       4. replaceOne（条件参数，新对象） 替换文档，没有修改操作符
    2. 删除文档：db.集合名.remove（条件参数[，justone]） 默认删除符合条件的所有文档，若第二个参数传true则只删一个
       1. db.集合名.deleteOne（）/ db.集合名.deteleMany（）
  1. 基本语法：
     1. 变量创建：var arr = []
     2. 流程控制：for（var i = 1；i<2000；i++）{}

1. Mongoose：是一个Node模块，可以通过Node操作MongoDB，是一个对象文档模型（ODM）库，即将文档映射为Node中的对象
   1. 基本：
      1. let mongoose = require（“mongoose”）
      2. mongoose.connect（‘mongodb：//数据库的ip地址：端口号/数据库名’） 若端口号是27017，可不写端口
      3. mongoose.connction.once（“open/close”，fn） 绑定连接成功/断开时的回调函数
      4. mongoose.disconnect（） 断开连接，一般不调用，MongoDB一般只连接一次不断开
   2. 对象：
      1. Schema（模式对象）：定义约束了数据库中的文档结构
         1. let Schema = mongoose.Schema
         2. let stuSchema = new Schema（｛字段名：值的类型，。。。｝，配置对象）
            1. 类型：String、Number、Boolean、｛type：String，default：“female”｝ 传对象可以设置默认值
            2. 配置对象：｛versionKey:false｝取消\_\_v字段
      2. Model：相当于数据库中的集合
         1. let stuModel = mongoose.model（”映射的集合名”，模式对象）
         2. 方法：实例为stuModel
            1. .create（doc【s】[，fn（err，doc）｛｝]） 插入一个（对象）或多个文档对象（对象数组）
            2. .find（条件参数[，投影][，选项][，fn（err，docs）｛｝]） 需要通过回调函数获取结果，docs为对象数组

投影：设置可以用mongodb的形式，也可以传字符串，需要的字段写字段名，不要的字段名前加 - ，多个字段用空格分隔

选项：｛skip：跳过数，limit：限制数｝

* + - * 1. .findById（id，[，投影][，选项][，fn]）/ .findOne（[条件参数][，投影][，选项][，fn]） 同上
        2. .update（条件参数，doc[，选项][，fn（err）｛｝]） 修改
        3. .updateOne（）/ .updateMany（）/ .replaceOne（） 参数同上
        4. .remove（条件参数[，fn]） / .deleteOne（...）/ .deleteMany（...） 删除
    1. Document：文档对象，是Model的实例，例：let stu = new **stuModel**（｛“name”：“chen”，。。。｝）
       1. **所有字段都是文档对象的属性，字段值为属性值**
       2. 方法：实例为stu
          1. .save（[选项][，fn（err）｛｝]） 保存
          2. .update（新对象[，选项][，fn（err）｛｝]） 修改自身，可用修改操作符
          3. .remove（ [fn]） 删除自身
          4. .get（字段名）/ .set（字段名，值） 获取/修改值
          5. .toObject（） 转换为JS对象