mongoDB. 中文社区

IT大咖说 知识分享平台

MongoDB应用从设计到实现

{ Name: "Yaoxing Zhang"
Title: "Senior Consulting Engineer"

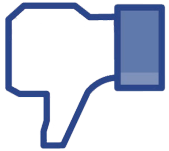
MongoDB的哲学

初识MongoDB

- MongoDB是什么？
- SQL与NoSQL的区别在哪？
- MongoDB能干什么？
- 应该怎么干？
- 什么事情不能干？

别人怎么说

- 正方：美得不像话
  - ✓ 读写速度快
  - ✓ 模式灵活
  - ✓ 高可用
  - ✓ 地理位置索引
  - ✓ 水平扩展
  - ~~✓ 完美取代RDBMS~~



- 反方：感觉不咋地
  - ✓ 不支持事务
  - ~~✓ 硬件数量要求多~~
  - ~~✓ 吃内存~~
  - ~~✓ CPU消耗大~~
  - ~~✓ 函数不够丰富~~
  - ~~✓ 丢失数据~~

关系与非关系模型

mongoDB 中文社区

IT大咖说 知识分享平台

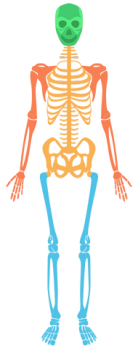
mongoDB

关系模型

非关系模型

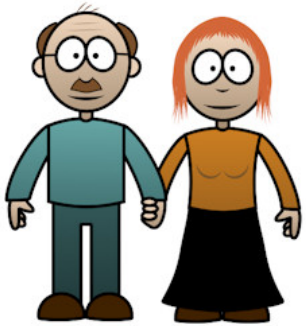
理论与现实

理论:



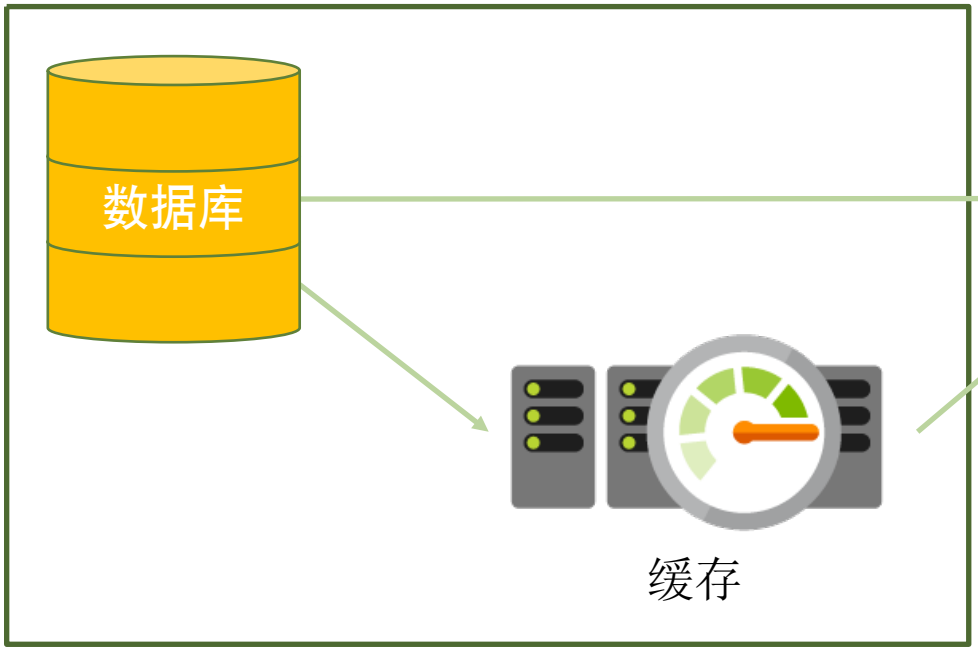
继承/实现

现实:



继承/实现

MongoDB的哲学

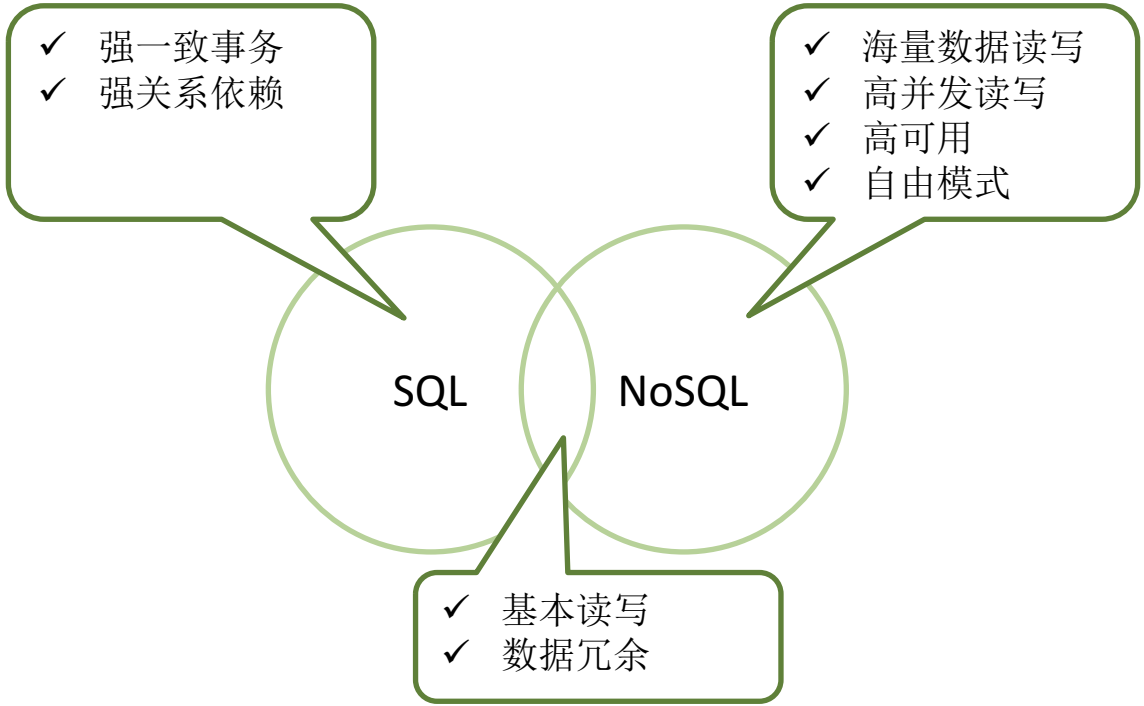


- ✓ 反范式
- ✓ 水平扩展
- ✓ 高可用
- ✓ 泛目的

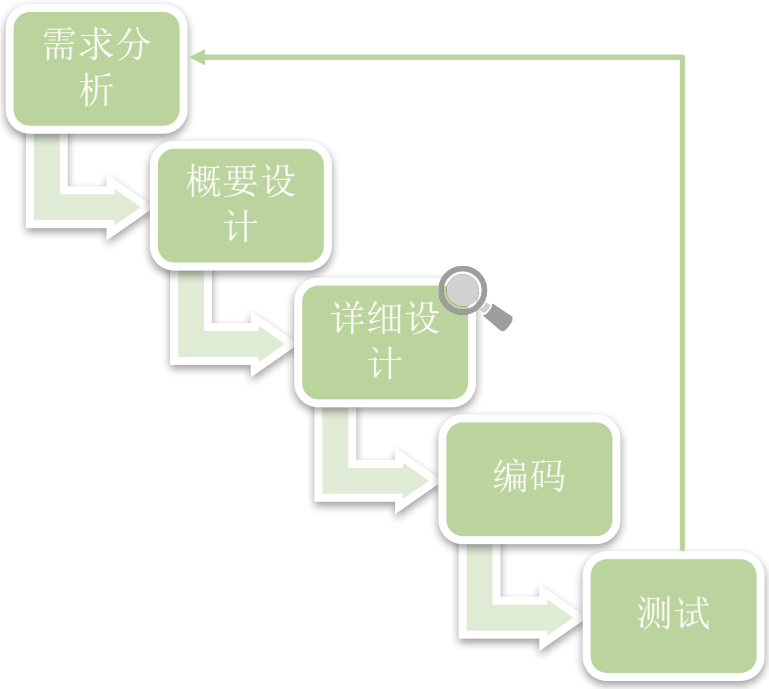
使用MongoDB解决问题

问题特点

- ✓ 强一致事务
- ✓ 海量数据读写
- ✓ 高并发读写
- ✓ 高可用
- ✓ 强关系依赖
- ✓ 基本读写
- ✓ 自由模式
- ✓ ……



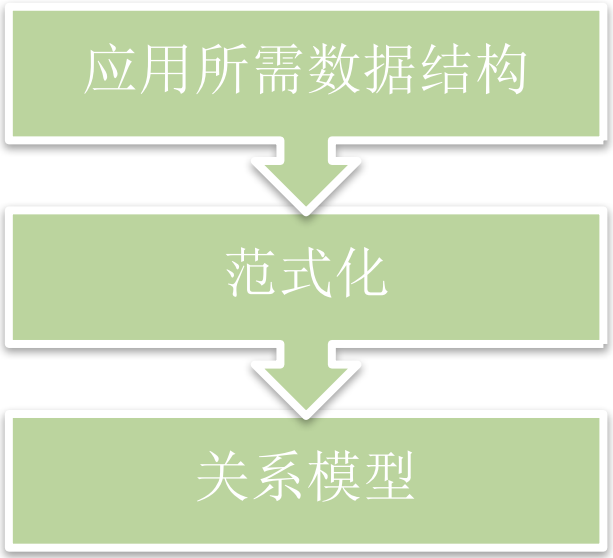
软件生命周期



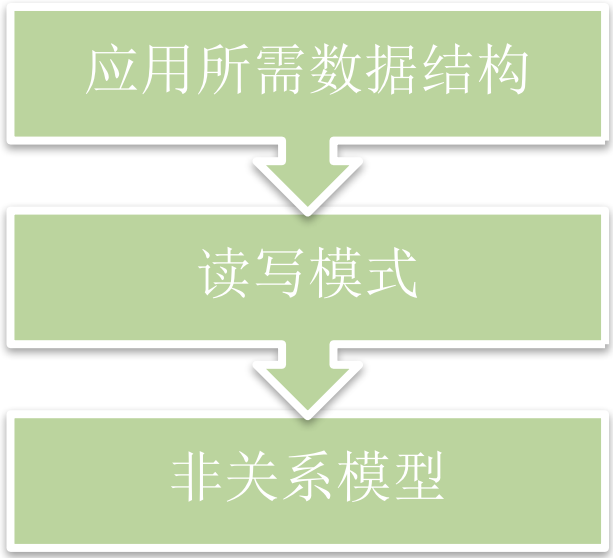
- ✓ .....
- ✓ 数据结构设计+数据读写模式
- ✓ 数据模型设计
- ✓ 详细设计说明书
- ✓ .....

数据模型设计

关系模型



非关系模型



要点

- 转变思路：根据读写模式而不是范式设计数据模型
- 积累经验：了解常见的数据结构对应的模式
- 理解思想：数据是应用的一部分，随应用一起迭代

实际应用

需求

- 存储应用程序日志
- 日志存储应尽可能不影响应用运行，所以存储速度要快
- 日志量可能会非常大，但日志通常不必保存很久
- 根据一些常见的条件查询日志，如：
  - ✓ 时间范围
  - ✓ 日志来源服务器
  - ✓ 异常类型

源代码：https://github.com/zhangyaoxing/MongoDbAppender

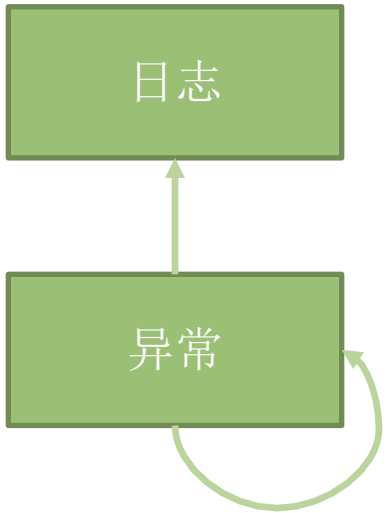
读写模式


```
> db.log.find({timestamp: {$gt: ISODate(“”), $lt: ISODate(“”)}});
> db.log.find({source: “”, timestamp: {…}});
> db.log.find({“exception.name”: “NullReferenceException”});
> db.log.insert({
    // everything of the log
  });

> db.log.createIndex({timestamp: 1});
> db.log.createIndex({source: 1, timestamp: 1});
> db.log.createIndex({“exception.name”: 1});
> …
```

数据模型

范式化



反范式化

```
{
      _id: ObjectId("…"),
      level: "Fatal",
      message: "Shit just happened!",
      source: "EC2-10.0.0.0",
      timestamp: ISODate("…"),
      exception: {
            name: "NullReferenceException",
            stack: "…",
            innerException: {
                  name: "…",
                  stack: "…",
                  innerException: {…}
            }
      }
}
```

其他方案？

```
// log实体
{
      _id: ObjectId(“…”),
      level: “Fatal”,
      message: “Shit just happened!”,
      source: “EC2-10.0.0.0”,
      timestamp: ISODate(“…”)
}
```

```
// exception实体
{
      _id: ObjectId(“…”),
      name: “NullReferenceException”,
      stack: “…”,
      parentException: ObjectId(“…”)
}
```

优势	劣势
可以根据每个innerException来查询	写入更多次，更慢
	处理逻辑更复杂

其他方案？

```
// log实体
{
      _id: ObjectId(“…”),
      timespan: ISODate(“”), // 精确到分钟
      log: [{
            level
            messa
            sourc
            times
      }, {
            level
            messa
again!”,
            source: “EC2-10.0.0.0”,
            timestamp: ISODate(“…”)
      }]
}
```

优势	劣势
数据量更少	单个文档可能过大

没有最好的方案
只有最合理的方案！

模式演化

新需求：在日志中记录用户自定义数据

```
{
     _id: ObjectId("…"),
     level: "Fatal",
     message: "Shit just happened!",
     source: "EC2-10.0.0.0",
     timestamp: ISODate("…"),
     userData: [
          {key: 'var1', value: 100},
          {key: 'var2', value: 200}
     ],
     exception: {
          name: "NullReferenceException",
          stack: "…",
          innerException: {…}
     }
}
```

Q & A