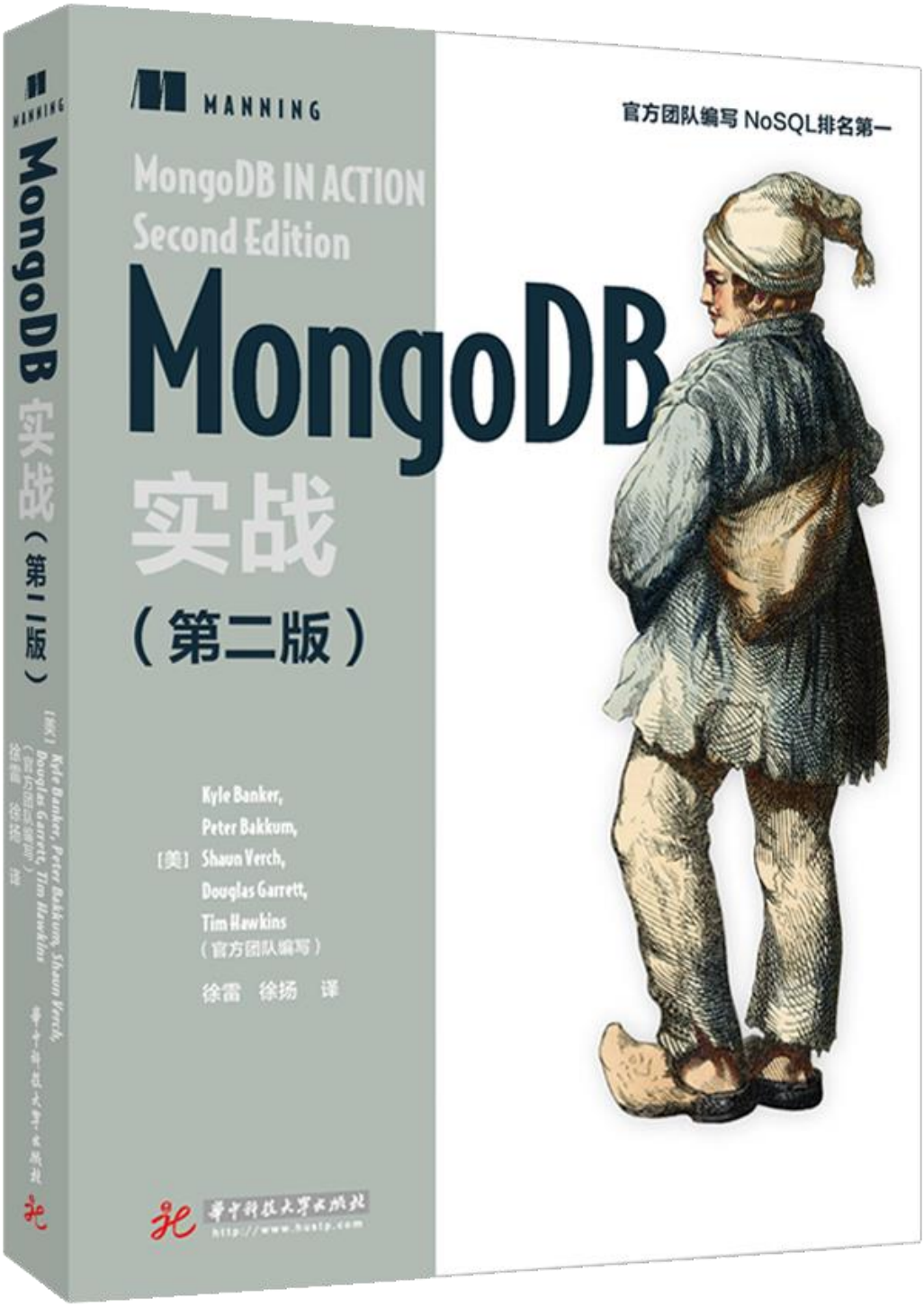
MANNING
官方团队编写 NoSQL排名第一
MongoDB IN ACTION
Second Edition
MongoDB
实战
（第二版）
Kyle Banker,
Peter Bakkum,
[美] Shaun Verch,
Douglas Garrett,
Tim Hawkins
（官方团队编写）
徐雷 徐扬 译
MongoDB 实战（第二版）
华中科技大学出版社
http://www.hustp.com

第一部分　Mongodb 实战入门

Getting started

1.**MongoDB优点**

感谢各位读者支持厚爱，本京东等多次断货。本次再版感谢热心读者的反馈建议。MongoDB是最流行的NoSQL数据库！在NoSQL中排名第一！高并发、高性能、灵活数据模型、易伸缩、易扩展、支持分布式查询。适用于互联网敏捷开发的需求。互联网架构师必备技术！MongoDB现在也成为Google、Facebook、阿里巴巴、腾讯、百度等一线互联网公司招聘的关键技能！

2.**NoSQL排名第一**

非常荣幸获得作者授权翻译MongoDB官方团队撰写的《MongoDB in Action》第2版。

非常荣幸受邀成为阿里巴巴Mongodb技术大会嘉宾讲师，并成为Mongodb中国专家组成员。MongoDB官方团队Kyle Banker和Shaun Verch等精心撰写的书籍。此书是MongoDB领域最权威的书籍，也是Manning出版社的“In Action实战”系列的经典书籍之一。全球读者五星好评。

3.MongoDB互联网和物联网公司必备技术

MongoDB的公司国外著名公司：Google、Facebook、Tweet、思科、Bosch、Adobe、SAP等，国内著名公司包括：阿里巴巴、腾讯、百度、360公司、新浪微博、京东、携程、滴滴打车、摩拜单车等名企。新版MongoDB 3.4更是对云计算和大数据分析Spark提供更强的支持，大数据工程师也必须掌握MongoDB，重要性不言而喻。

《Mongodb实战》第一部分对MongoDB做了整体介绍，并介绍了实际的开发例子。另外还介绍了JavaScript shell和Ruby驱动，Java和C#例子在读者群中下载。这两个概念会贯穿于本书的所有例子中。

本书主要是面对开发人员，但是如果你是临时使用MongoDB，本书也可以提供很多帮助。虽然我们主要关注MongoDB数据库，但是之前的编程经验会帮助我们理解例子代码。如果之前使用过关系型数据库，那就太棒了！我们会经常对比这两种数据库。

在编写本书的时候，MongoDB 3.6版本是最新的版本，但是本书大部分介绍会兼顾之前的MongoDB版本。我们也会强调哪些特性不适用于旧版本。

大部分例子我们会使用JavaScript编写，因为MongoDB JavaScript shell便于实验操作。Ruby是特殊的MongoDB语言之一，我们的例子代码会介绍如何在真实的项目中使用Ruby操作MongoDB数据库。放心，就算大家不是Ruby程序员，我们也可以使用其他语言来开发MongoDB，因为语法都是相似的。

第1章介绍MongoDB的历史、设计目标，以及典型的使用场景。大家也会看到为什么MongoDB可以在与其他NoSQL数据库的竞争中胜出，做到独一无二。

第2章介绍MongoDB shell的用法，让大家熟悉常见的命令工具。我们会介绍MongoDB基本的查询语言，并且通过创建、查询、更新和删除文档来练习实战。本章还会介绍一些高级的shell技巧和MongoDB命令。

第3章主要介绍MongoDB驱动以及BSON数据格式。

这里我们会介绍如何使用Ruby语言来操作数据库，而且会使用Ruby开发一个Demo程序来演示MongoDB的灵活性和强大的查询机制。

要充分掌握本书介绍的知识，请仔细阅读并且亲自尝试每一个例子。如果还没有安装MongoDB，就请参考本书附录A的内容，里面有详细的安装向导。[1]

2017年3月12日，阿里巴巴Mongodb技术大会新青年合影留念

[1]【译注】MongoDB 的安装可以参考官方文档 mongodb.org，如果你是其他语言比如 C#、Java 或者 Node 的开发者，也不要担心，官方文档提供了详细的介绍。大家也可以加群 203822816 索取例子代码。

关于译者：

1) Mongodb中国区专家、微软特邀讲师、阿里巴巴Mongodb大会嘉宾讲师

2) 获得吉林大学计算机科学与技术学士学位，上海交通大学硕士学位

3) 曾任：购酒网、北极绒 复旦元方、广西省政府云平台等公司技术架构顾问

4) 国外经典《MongoDB实战》第2版、《24种云计算架构模式》等译者

5) 受邀为微软中国、盛大网络、玫琳凯中国研发中心、世界500强约翰迪尔、一嗨租车、沪江网、中国东方航空、美国IGT、世界500强花旗银行、世界500强达丰集团、世界500强台达集团、美国国家仪器、上海交通大学软件学院、奥伯特石油、中国体彩集团总部、世界500强南方电网集团、阿里系恒生证券、丹麦诺和诺德集团、阿里巴巴Mongodb大会、中国电信科学技术第10所等中外名企、名校讲授架构课程。

6) 2017年3月12日，受邀作为嘉宾讲师在阿里巴巴Mongodb技术大会，发表Mongodb分布式高可用架构主题演讲。

7) 专注于Linux和Win平台分布式高并发高可用HA架构技术领域

8) 喜欢格言座右铭：Stay Hungry,Stay Foolish(谦卑若愚，好学若饥)。

（Mongodb中国学习交流群 二维码）

第 1 章

全新Web数据库

A database for the modern Web

本章内容

- MongoDB历史、设计目标以及关键特性
- 简要介绍shell和驱动
- 使用场景和限制
- MongoDB 最新更新

如果你最近几年在开发Web应用，则可能一直使用关系型数据库作为主要的存储方式。如果你熟悉SQL，可能非常喜欢标准化[1]的数据模型、事务的必要性，以及持久化引擎提供的保证。简单来说就是关系型数据库成熟并且大名鼎鼎。当开发者开始寻找其他替代数据库时，关于新技术的有效性和实用性的疑问不断产生：这些新的数据库要取代传统关系型数据库吗？谁在生产环境中使用它，为什么？迁移到非关系型数据库的动机是什么？其实所有问题的答案归结到一起就只有一个：为什么开发者喜欢MongoDB？

MongoDB是为快速开发互联网Web应用而设计的数据库系统。其数据模型和持久化策略就是为了构建高读/写吞吐量和高自动灾备伸缩性的系统。无论系统需要单个还是多个节点，MongoDB都可以提供高性能。如果你经历过关系型数据库的伸缩困境，那么使用MongoDB就可避免这种困境。但是并非每个人都需要伸缩性操作。如果你需要的就是单台数据库服务器，那么为什么还要使用MongoDB呢？

或许开发者使用MongoDB的最大理由并非是其伸缩策略特性，而是其直观的数据模型。MongoDB在文档里存储数据而不是在行里。什么是文档？下面就是一个例子：

```
{
```

[1]当我们提到标准化的数据模型时，通常是指数据库减少冗余的设计范式。例如，在 SQL 数据库中我们可以把数据分割为不同的部分，比如 Users 和 Orders 表，然后减少用户信息存储的冗余数据，我们称作 3NF。

```
  _id: 10,
  username: 'peter',
  email: 'pbbakkum@gmail.com'
}
```

这是个非常简单的文档；它存储了用户的一些简单信息字段（听起来很酷）。那这个数据模型的优势是什么？设想这种情况：需要为每个用户存储多个email。在关系型数据库里，需要创建单独的email表，然后通过外键关联起来；而MongoDB提供了非常简单的方法来解决这种问题：

```
{
  _id: 10,
  username: 'peter',
  email: [
    'pbbakkum@gmail.com',
    'pbb7c@virginia.edu'
  ]
}
```

如上所示，我们只是创建了一个email数组就解决了问题。作为开发者，我们会发现，这一优点非常有用，在数据经常变化的时候我们只需要存储一个结构化文档即可，无须担心数据模型的变化。

MongoDB的文档格式基于JSON，一种流行的数据存储格式。JSON是的JavaScript Object Notation的简称。正如我们看到的，JSON数据结构由键值对组成，也可以内置嵌套。这一点与其他语言中的哈希映射以及字典类型相似。

基于文档的数据模型可以表示丰富的、多层次的数据结构。它经常用来处理无须多表关联的工作。

例如，假设要为电商网站的数据库建模，若使用标准的范式设计数据库，信息可能分割到几十个表中存储，要获得完整的数据库里存储的商品信息，则可能需要关联SQL查询。

相比之下，使用文档模型时，绝大部分单个商品信息都可以存储在单个文档里。打开MongoDB JavaScript shell就可以轻易获取商品的类JSON数据信息。我们也可以查询或者操作它。MongoDB的查询功能是专门用于处理结构化文档操作的，所以用户从关系型数据库到非关系型数据库查询体验基本在同一层次。此外，大部分开发者使用的是面向对象编辑语言，他们希望找到更好的数据库用于映射对象。使用MongoDB，语言中定义的对象可以原样持久化保存，减少了对象映射的复杂性。如果你开发过关系型数据库，这些经验也有助于转换现有技能到新的数据库中。

如果你不熟悉表格数据和对象数据之间的区别，则可能会有很多疑问。不过请放心，学习完

第1章你就会对MongoDB的特性以及设计目标有个清晰的认识。我们会先介绍一下MongoDB的发展历史，以及其主要特性。然后会介绍其他NoSQL[1]解决方案，以及MongoDB是如何解决问题的。最后我们会介绍MongoDB最佳的使用场景，以及它的局限性。

MongoDB在多个方面备受批评，其中有些是公正的，但有些是歪曲的。我们的观点是，它就是开发人员武器库中的一件兵器，和其他数据库一样，你应该知道它的优点和缺点。某些工作需要多表关联和更多的内存管理，而有些工作更适合使用基于文档的数据模型。缺少数据架构定义意味着MongoDB更灵活，更适合快速开发模式。我们的目标是告诉大家自己决定MongoDB是否适合自己，以及如何高效地使用它。

1.1 为互联网而生

Built for the Internet

MongoDB的历史不长，但是它诞生于一个雄心勃勃的项目。在2007年，纽约一个叫10gen的创业团队开始工作在一个平台即服务(PaaS)上，由一个应用服务器和一个数据库组成，托管Web应用，根据需要伸缩。与Google App Engine类似，10gen平台的设计目标就是自动处理伸缩和管理硬件与软件基础架构，解放开发者，使得他们可以专注于应用开发。10gen最终发现，开发者并不喜欢放弃对于技术栈的控制，但是用户却喜欢上了10gen的数据库技术。这就导致了10gen团队专注于开发这个数据库产品，最后就形成了MongoDB项目。

10gen公司的名字也已经修改为MongoDB, Inc。该公司继续支持这个开源项目的开发工作。代码完全公开下载，并且可以免费修改、使用，只要遵守代码开源协议即可。而且鼓励社区提交Bug和补丁。另外，MongoDB核心开发者要么是公司的联合创始人，要么是公司的员工，项目的路线图专注于满足用户社区的需求以及创建兼具关系型数据库和分布式键值存储最佳特性的数据库。因此MongoDB公司的业务模型与其他著名的开源公司不同：支持开源产品的开发并且提供给终端用户订阅服务。

从MongoDB历史中我们了解的最重要的事情就是MongoDB的设计目标就是极简、灵活、作为Web应用栈的一部分。这些使用情况驱动了MongoDB开发过程，并且可以帮助解释它的特性。

[1] 2009 年出现的 NoSQL 一词主要用于描述当时日益流行的非关系型数据库，它们的共同点是使用了 SQL 之外的查询语言。

1.2 MongoDB 关键特性

MongoDB's key features

数据库很大程度上是由其数据模型定义的。本节里，我们会看到文档数据模型，然后会看到MongoDB对此数据模型的高效处理特性。本节里也会介绍其他操作，专注于MongoDB的主从复制以及水平扩展策略。

1.2.1 文档数据模型

MongoDB的数据模型是面向文档的。如果对于这里的文档一词不熟悉，则可以通过下面的例子来理解。

JSON文档中除了数值类型以外，其他都需要一对引号。

列表1.1展示了JavaScript版本的JSON文档。这里的引号不是必须的。

列表1.1　表示新闻网入口的文档

```
{
  _id: ObjectID('4bd9e8e17cefd644108961bb'),
  title: 'Adventures in Databases',
  url: 'http://example.com/databases.txt',
  author: 'msmith',
  vote_count: 20,
  tags: ['databases', 'mongodb', 'indexing'],
  image: {
    url: 'http://example.com/db.jpg',
    caption: 'A database.',
    type: 'jpg',
    size: 75381,
    data: 'Binary'
  },
  comments: [
    {
      user: 'bjones',
      text: 'Interesting article.'
    },
    {
      user: 'sverch',
      text: 'Color me skeptical!'
    }
  ]
}
```

id 字段
主键

❶ 作为字符串数组存储的标签

❷ 指向另外一个文档的特性

❸ 作为对象数组存储的评论

这个例子代码展示了新闻网站上一篇文章的JSON文档格式（想想Reddit或者Twitter，国内读者若无法访问，则可以类比网易或者今日头条）。正如大家看到的，该文档包含一系列名称

和值的集合。这些值可以是简单的数据类型，比如字符串、数字和日期等。当然，这些值也可以是数组，甚至是JSON文档❷。后面构造的文档表示了各种不同的数据结构。我们可以看到例子文档包含tags属性❶，它把文章的标签存储到数组里。但是最有意思的是comments属性❸，它是一个由评论组成的数组。

从内部来讲，MongoDB以二进制JSON格式存储文档数据，或者叫做BSON。BSON有相似的数据结构，但是专门为文档存储设计。当查询MongoDB并返回结果时，这些数据就会转换为易于阅读的数据格式。MongoDB shell使用JavaScript获取JSON格式的文档数据，这也是我们绝大多数例子使用的格式。我们会在后面的章节里深入讨论BSON数据格式。

关系型数据库包含表，MongoDB 拥有集合。换句话说，MySQL在表的行里保存数据，而MongoDB在集合的文档里保存数据，你可以把集合当做一组文档数据。集合是MongoDB中非常重要的概念。集合中的数据存储在磁盘上，而且大部分查询需要指定查询的目标集合。

我们来花点时间来比较MongoDB集合与标准的关系型数据库表示相同数据的差别。图1.1所示为可能的关系结构。因为表本质上是平滑的，表示文档中的文章一对多关系需要多个表。我们可以创建一个posts表存储文章的核心信息。然后可以再创建三个其他的表，每个表包含一个外键字段post_id，关联最初的文章。符合范式设计的数据集保证数据集只在一个地方存储一次。

但是严格的范式设计是需要成本的。值得注意的是，有时候需要一些程序集。要显示刚才关联的文章post，需要执行post和comments表的链接查询。是否需要严格范式设计则最终取决于要建模的数据类型。第4章会深入讨论这个问题。面向文档的数据模型天生就适合表示集中形式的数据，允许我们处理整个数据，从评论到标签，都可以包含在单个数据库对象中。

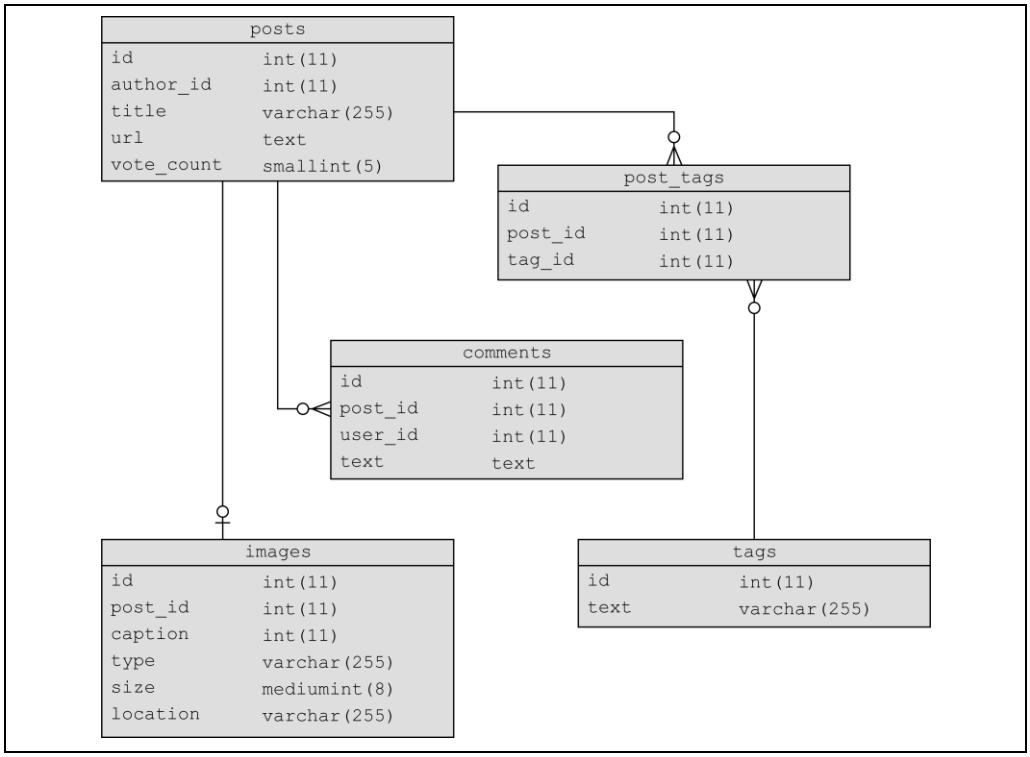

图 1.1　社交新闻网站的实体关联数据模型

十字架线条代表一对一关系，所以一个posts表记录对应一个images记录。三叉线表示一对多关系，所以comments表可以多个评论关联posts表的一条记录。

你可能已经注意到了，除了支持丰富的数据结构外，文档不需要遵守严格的数据定义schema。我们在表里存储行，每个表有严格的schema，指定每个列的数据类型。如果某行需要扩展字段，就必须修改表结构。MongoDB把文档归集到集合中，集合不需要定义任何schema。理论上，每个集合中的文档都可以拥有不同的数据结构；实际上，集合中的文档都是相对一致的。例如，每个posts文章集合中的文档都有标题、标签和评论字段等。

无 schema 模型的优点

不强制定义schema带来了一些好处。首先，应用程序的代码强制数据结构而不是数据库。在频繁修改数据定义的时候这可以加速应用程序开发。

其次，无schema模型允许用户使用真正的变量属性来表示数据。例如，假设你在构建一个电商的商品目录表。由于没有办法知道一个商品包含什么属性，因此应用程序需要处理这些变

化。传统的处理方法是在固定schema数据库里使用实体属性值模式[1]，如图1.2所示。

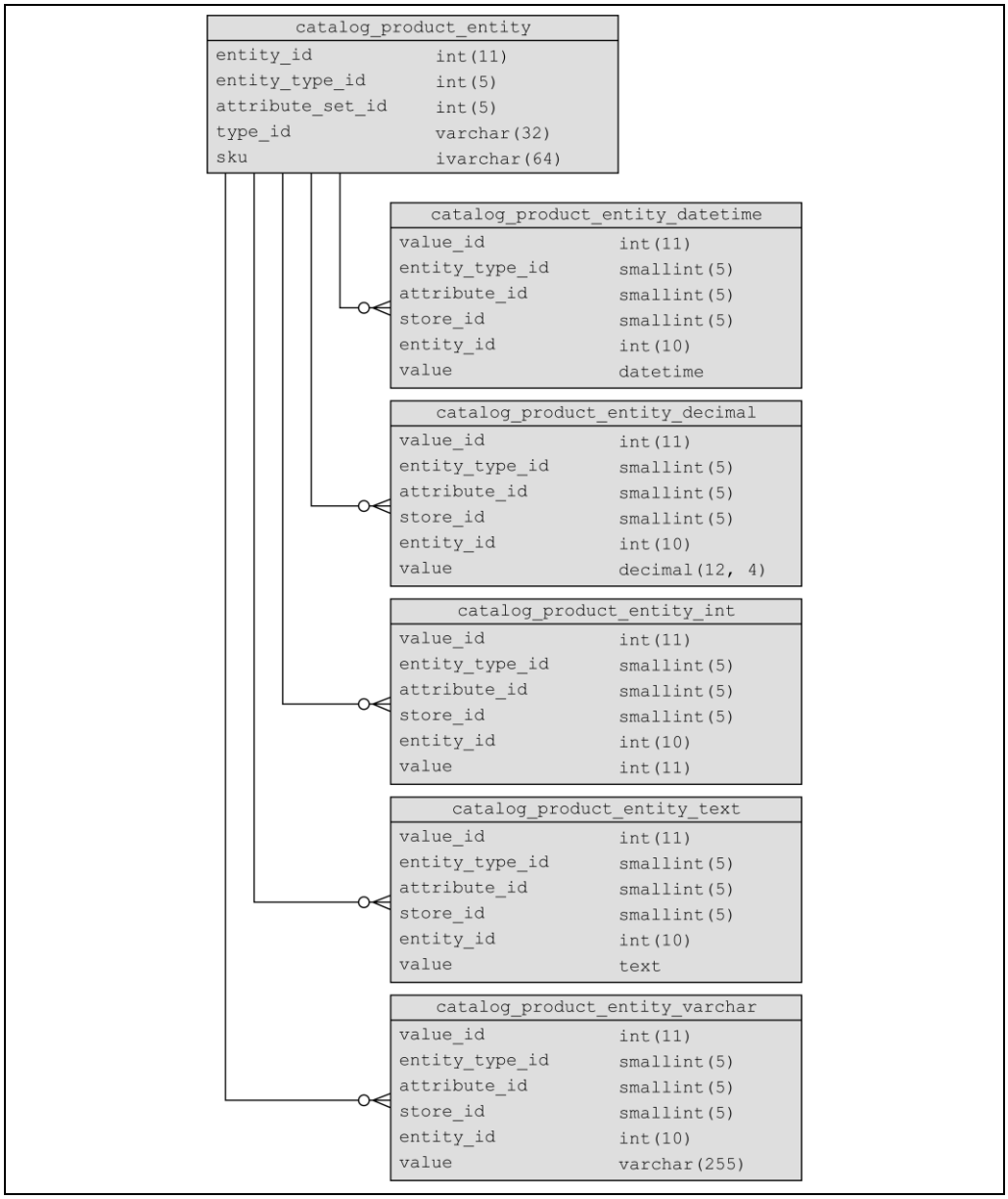

图 1.2　电商应用的部分 schema，这些商品表使用了动态属性创建模式

你所看到的只是电商网站的部分数据模型。注意：这些基本都是相似的，除了一个属性值是

[1]更多信息请参考 http://en.wikipedia.org/wiki/Entity-attribute-value_model.

通过数据类型datatype变换的以外。这个数据结构允许管理员来定义额外的产品类型以及属性，但是导致的问题相当复杂。思考一下，如果使用MySQL shell来检查或者更新某个产品模型的数据，则商品SQL关联查询就会相当复杂。而文档建模就不需要关联，而且可以动态添加新属性。虽然不是所有的模型都这么复杂，但是使用MongoDB开发应用就不需要担心未来可能的数据字段的变化。

1.2.2 ad hoc 查询

常说的系统支持主动查询模式（ad hoc queries）是指不需要事先定义系统接收何种查询。关系型数据库有这个属性，它们忠实地执行格式正确的包含各种条件的SQL查询。如果使用过关系型数据库，就知道ad hoc查询很容易。但是不是所有的数据库都支持动态查询，例如，键值存储的查询只支持一个领域的查询：键key。与其他系统类似，键-值存储数据库牺牲丰富的查询功能来换取更简单的伸缩模型。MongoDB的设计目标之一是保留大部分关系型数据库的功能。

要看下MongoDB查询语言如何工作，可先来看个简单的例子：包含文章和评论。假设我们要查询所有包含标签politics并且有10个以上投票的文章。若使用SQL语句，代码如下：

```
SELECT * FROM posts
  INNER JOIN posts_tags ON posts.id = posts_tags.post_id
  INNER JOIN tags ON posts_tags.tag_id == tags.id
  WHERE tags.text = 'politics' AND posts.vote_count > 10;
```

等价的MongoDB查询，使用了document文档作为匹配器，则代码如下。特别的`$gt`键表示大于条件。

```
db.posts.find({'tags': 'politics', 'vote_count': {'$gt': 10}});
```

注意：两个查询假设了不同的数据模型。SQL查询依赖于严格的范式模型，posts和tags存储在不同的表里，而MongoDB假设tags保存在每个post文档对象里。但是两个查询演示了任意组合属性的功能，这也是ad hoc查询的功能。

1.2.3 索引

ad hoc查询的一个关键元素就是查找在创建数据库时还不知道的值。随着数据库中添加的文档数据越来越多，查询值的成本变得越来越高。这有时无异于大海捞针。因此，需要一种高效的方式来搜索数据。

理解数据库索引最好的方法就是类比：许多书籍都有索引，包含关键字和页码。假设数据库索引也是提供相似服务的数据结构。假设你有一本秘笈食谱，而你想找出所有和梨子有关的烹饪方法（假设你有许多梨子，又不希望它们坏掉）。耗时的方法就是翻遍图书的所有页面查找每个方法，逐页查找梨子的做法。而大部分人会选择查看书籍的索引，找出所有包含梨子关键字的食谱列表。

MongoDB中的索引就是使用了B-树（平衡树）数据结构。B-树索引也大量使用于许多关系型数据库中，对于不同的查询做了优化，包括范围扫描和条件子句查询。但是新的引擎已经支持日志结构合并-树(LSM)，可以在MongoDB 3.2版本中使用。

大部分数据库会给每个文档对象一个主键（primary key），一个唯一的数据标识。每个主键会自动索引，这样就可以使用唯一的键来高效地访问每个数据，MongoDB也不例外。但是不是所有数据库都允许我们来为单个的行或者文档建立索引。这些叫做辅助索引（secondary indexes）。许多NoSQL数据库，比如HBase，被当做keyvalue数据库，这是因为它们不允许辅助索引（secondary indexes）。通过允许多个辅助索引，MongoDB可以允许用户优化不同的查询。这是MongoDB重要的功能特性。

使用MongoDB，每个集合我们可以创建64个索引。这些索引也可以在其他关系型数据库中找到；升序、降序、复合键、哈希、文本以及地理空间索引[1]。因为MongoDB和绝大多数关系型数据库RDBMSs使用了相同的索引数据结构，所以管理这些系统的建议都是类似的。你会在下一章里阅读索引的内容，而且必须明白索引对于高效操作数据库至关重要，第8章里会深入介绍这个主题。

1.2.4 复制

MongoDB提供了数据库复制特性，叫做可复制集合（replica set）。可复制集合在多个机器上分布式存储数据，在服务器或者网络出错时，实现数据冗余存储和自动灾备。此外，复制还用于伸缩数据库读操作。如果你在开发读取密集型应用，比如常见的一些网站项目，就可以通过分散读取压力到可复制集群中的服务器来实现。

可复制集合由多台服务器组成集群。通常，每个服务器有独立的物理机。我们调用这些节点，任意时候，一个节点作为主节点，则其他的作为次节点。与其他数据库中的主从复制类似，可复制集合的主节点可以同时接受读/写操作，但是从服务器只能进行读操作。

[1]地理位置索引允许高效率查询经纬度点信息；会在本书后面进行讨论。

真正让可复制集独一无二的是它可以支持自动化灾备：如果主节点失败，则集群中会选择一个从节点，并自动提升为主节点。当之前的主节点回归时，它会继续作为从节点。整个过程如图1.3所述。

复制是MongoDB最有用的特性，我们会在后面的章节里深入讲解。

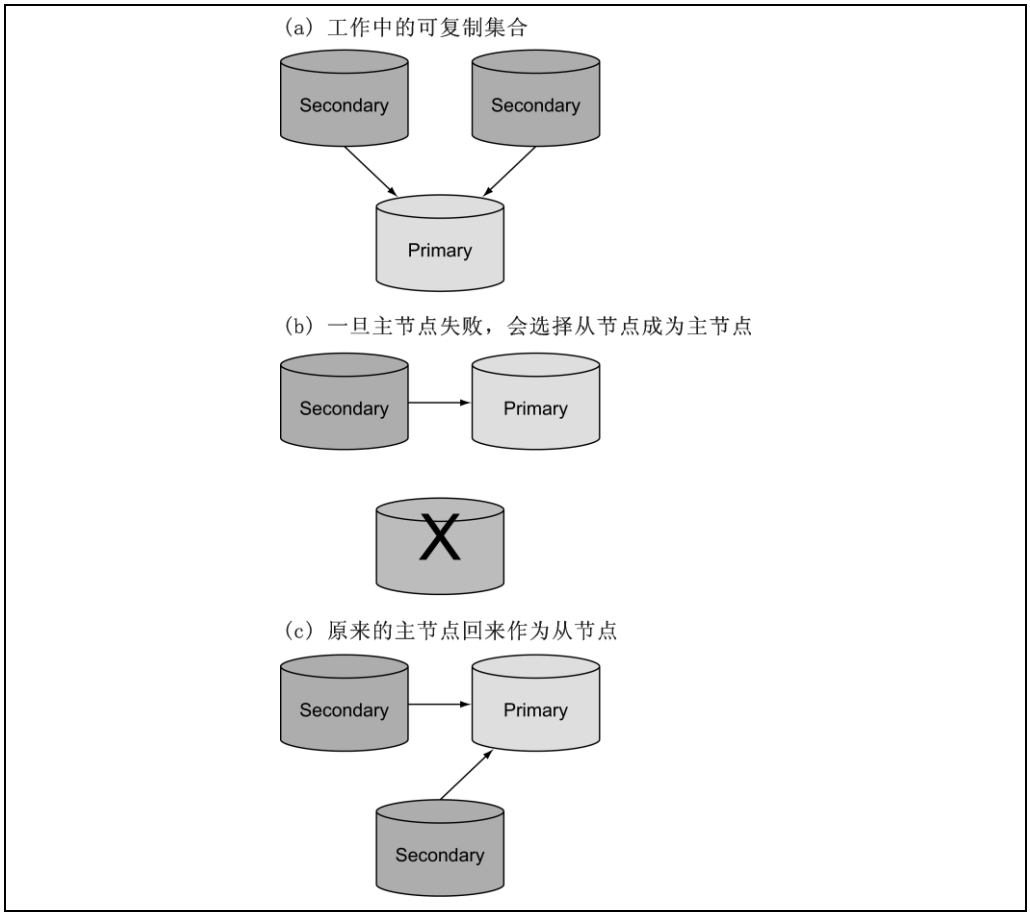

图 1.3　可复制集合的自动化灾备

1.2.5　加速与持久化

要理解MongoDB的持久化方法，首先要思考一些新概念。在数据库系统领域，写入速度和持久性之间存在矛盾的关系。

写入速度（write speed）可以理解为数据库在给定的时间内插入、更新和删除的容量。持久

性（durability）指的是这些写操作被永久保存的保证级别。

例如，假如写入100条每条50 KB的记录到数据库中，然后立即切断电源。当重新开机的时候，这些数据还能恢复吗？答案是：取决于数据库系统的配置以及托管硬件。绝大部分数据库默认启动了很好的持久性，所以如果发生这种事情也是安全的。对于一些应用，比如存储日志线，即使可能导致数据丢失，快速写入也很有意义。写入磁盘的速度远远慢于写入内存RAM的速度。特定的数据库，比如Memcached，专门写入RAM，所以它的速度非常快，但是数据容易丢失。换句话说，很少有数据库只写磁盘，因为这种操作性能太低，无法接受。因此，数据库设计者经常需要在速度和持久性之间妥协，做出最佳平衡。

事 务 日 志

可以在MySQL的InnoDB里看到写入速度和持久性之间的妥协。InnoDB是一个事务性存储引擎，它可以确保持久性。它通过两个地方确保实现这一目标：一是在事务日志，二是在内存缓存池里。事务日志会立即同步到磁盘，而缓存池会通过后台线程同步到磁盘上。双写的原因是随机I/O比顺序I/O慢得多。因为写入主数据文件是随机I/O，而把这些修改写入RAM就快得多，而后再允许同步到磁盘上。

由于某些类型的写磁盘会确保持久性，而且重要的是顺序写磁盘的，因此很快；这也是事物日志提供的特性。

对于意外关机的情况，InnoDB可以替换事物日志，并且更新主数据文件。这在确保高级别持久性的同时提供了一个可以接受的性能级别。

对于MongoDB，用户通过选择写入语义来维持速度与持久性之间的平衡，决定是否启用日志。从MongoDB 2.0以后就默认启用了日志功能。在2012年11月份发布的驱动里，MongoDB确保写入操作在返回用户之前已经写进了RAM，但此特性可以配置。我们也可以配置MongoDB为fire-and-forget，发送给服务一个写命令而不需要等待确认结果。也可以配置MongoDB来确保已经写入到各个可复制集群的节点中，再返回确认结果。对于高容量、低价值的数据（比如点击流和日志），写入后就不用管的模式比较理想。对于重要的数据，安全模式的设置是必须的。

要知道在MongoDB 2.0之前的版本里，默认是使用不安全的写入后不用管的模式fire-and-forget，因为10gen开始开发MongoDB时，只专注于数据层，相信应用层会处理这些错误。但是随着MongoDB使用越来越流行，不仅仅在Web层，对于不想丢失数据的应用来说

这样太不安全了。

从MongoDB 2.0开始，日志功能默认是启用的。启用日志功能后，默认100毫秒就会写一次日志文件。如果服务器意外关机，日志会通过重启服务器来确保MongoDB数据文件恢复为一致状态。这是运行MongoDB最安全的方式。

对于写入压力，可以通过关闭日志功能以提高性能。坏处是意外关机之后可能导致数据文件冲突。因此，关闭日志功能，就应该使用主从复制模式，推荐一个从服务器，即使一台机器关机，还有一台机器保证数据的完整性。

设计MongoDB的目标就是让大家可以保持速度与持久性平衡，但是对于重要的数据，我们强烈推荐安全设置。复制和持久化的主题范围很大，我们会在第11章里详细介绍。

1.2.6 伸缩

伸缩数据库的最简单方式就升级服务器硬件。如果你的应用是运行在单个节点上，则通常可行的方案就是通过组合添加更快的磁盘、更多内存以及更强的CPU来解除数据库性能瓶颈。提升单节点参数的做法通常也称垂直扩展（vertical scaling或scaling up）。垂直扩展非常简单、可靠，但是达到某个点后成本很高，但是最终我们会达到一个无法低成本垂直扩展的临界点。

然后可以考虑水平扩展（horizontally或scaling out），如图1.4所示。水平扩展指的是在多台机器上分布式存储数据库，而不是提升单个节点的配置。水平扩展架构可以运行在许多台很小的、廉价的机器上，通常可以减少硬件的成本。而且，跨机器分布式存储数据可以降低宕机带来的丢失数据的后果。服务器无法避免关机。如果你已经做过垂直伸缩，经历了宕机，那就应该处理一下系统最依赖的服务器失败问题了。相比水平扩展架构的失败，作为一个整体它降低了单个节点宕机带来的风险概率。

设计MongoDB的目标就是利用其水平伸缩。它通过基于范围的分区机制来实现水平扩展，称为分片机制，它可以自动化管理每个分布式节点存储的数据。另外，还有基于哈希和基于tag的分片机制，这也是另外一种形式的基于范围的分片机制。

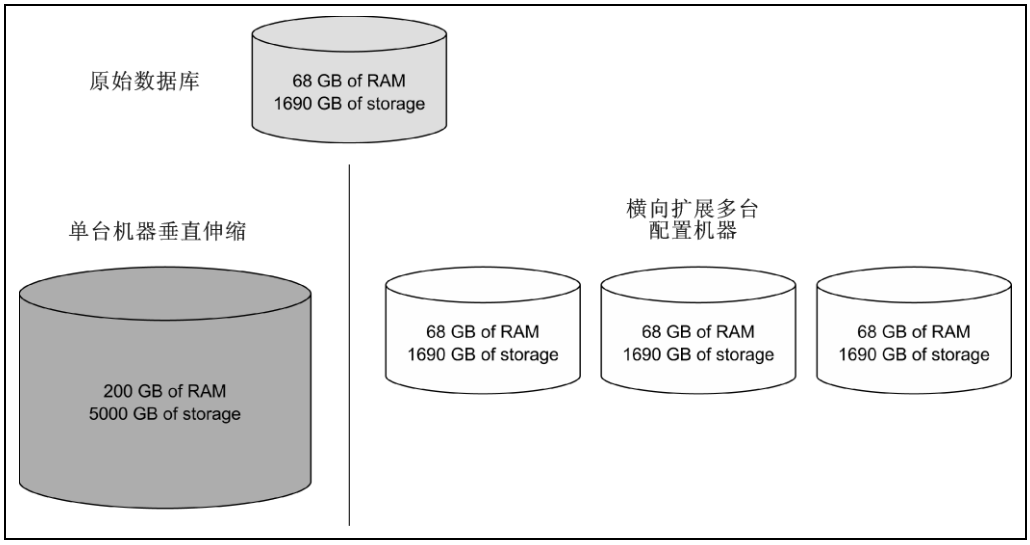

图 1.4　水平与垂直扩展

分片系统处理额外的分片节点，而且它还会处理自动化灾备。每个独立的节点是一个可复制集合，至少由2台机器组成，确保节点失败的时候可以自动恢复。所有这些都意味着没有应用节点必须处理这些逻辑；我们的应用与MongoDB集群通信就好像与单个节点通信一样。第12章会深入讲解分片集群的知识。

我们已经了解了MongoDB最重要的功能特性；第2章里我们会开始学习如何实际开发MongoDB。现在我们近距离来看看这个数据库。下一节，我们会在它的环境里学习MongoDB，与核心服务器一起发布的工具，以及一些存取数据的方法。

1.3　核心服务和工具

MongoDB's core server and tools

MongoDB使用C++编写，由MongoDB，Inc公司负责开发。这个项目支持主流的操作系统，包括Mac OS X、Windows、Solaris 以及最流行的Linux版本。这些平台的预编译版本可以在http://mongodb.org下载。MongoDB是开源的，基于GNU-Affero General Public License (AGPL)开源协议。源代码可以在GitHub下载，社区也可以贡献力量。但是项目由MongoDB, Inc核心服务器团队主导，而且他们开发了绝大多数的代码[1]。

[1]【译者注】翻译此书时最新的版本是 3.2.6。我们也使用了最新的 3.4 进行集群和安全的实战配置。

> **关于 GNU-AGPL 开源协议**
>
> GNU-AGPL开源协议存在一些争议。实际上，这个许可协议意味着源代码可以下载，鼓励社区参与。但是GNU-AGPL要求，任何对于源码的修改必须公开发布造福社区。这可能对于那些想修改MongoDB源码，但是不想公开的公司是个困扰。对于这些想要保护自己公司核心服务器代码的公司，MongoDB，Inc提供了特别的商业许可证。

MongoDB 1.0于2009年11月发布。主要的发布大约3个月一次，偶数版本作为稳定版，奇数版本作为开发版。编写此书时最新的版本是v3.0[1]。

下面介绍的是与MongoDB一起发布的开发应用需要的工具和各种语言的驱动组件。

1.3.1 核心服务器

核心服务器通过名为`mongod`的可执行文件运行(`mongodb.exe`在Windows系统中运行)。`Mongod`服务器进程使用自定义的二进制协议从网络上接受命令。所有`mongod`进程的数据库在类Unix系统中默认存储在/data/db路径下，在Windows中存储在c:\data\db下。本书中的某些例子更适用于Linux系统。绝大部分MongoDB生产服务器都运行在Linux上，因为它们更加可靠、应用更广泛和更具优越性。

Mongod可以在几种模式下运行，比如独立模式，或者可复制集群模式。在生产环境中，我们推荐使用可复制群模式MongoDB。通常我们会看到，可复制集群由两台服务器加上一个mongod作为裁判组成。最后，还有一个独立的mongos路由服务器，它用来在分片集群中转发不同的请求到后台服务器。不用担心这些选项，我们会在第11章和第12章中详细介绍这些知识。

配置mongod进程相对简单，它可以接受命令行参数，也可以接受配置文件文本控制。一些常见的配置可以通过修改mongod侦听的端口和存储数据的目录来实现。要查看这些配置参数，可以运行mongod 帮助文件。

[1]你应该使用最新的稳定版本，例如 v3.6（2017 年 9 月发布）。在附录 A 里有完整的安装指南。

1.3.2 JavaScript shell

MongoDB命令行工具是一个基于JavaScript[1]的数据库操作和管理工具。`Mongo`加载命令`shell`后连接特定的`mongod`进程，或者默认本地运行。MongoDB shell和MySQL shell类似，最大的不同是它基于JavaScript和SQL脚本。例如，可以选择一个数据库，然后插入一个简单的文档对象到`users`集合中：

```
> use my_database
> db.users.insert({name: "Kyle"})
```

第一个命令用于选择你要使用的数据库，MySQL用户很熟悉。第二个命令是Javascript表达式，用于插入一个简单的文档。要查看插入的结果，可以使用如下简单的查询：

```
> db.users.find()
{ _id: ObjectId("4ba667b0a90578631c9caea0"), name: "Kyle" }
```

`Find`方法返回插入的文档数据，带有一个对象`ID`。所有的文档都需要一个`_id`字段作为主键。如果可以确保唯一，也可以自己设置`_id`的值。如果忽略这个`_id`值，MongoDB会自动生成一个唯一的ID插入数据库。

除了允许插入和查询数据，shell还允许我们运行管理员命令。某些例子还保护查看当前数据库操作，检查复制到从节点的状态，以及配置分片集群的某个集合。正如你看到的，MongoDB shell是个非常强大的工具，值得我们学习。

所有这些知识点，都会通过使用某个语言结合MongoDB开发一个应用来完成。为此，我们必须了解一些MongoDB的语言驱动程序。

1.3.3 数据库驱动

也许数据库驱动的概念让人想起底层设备驱动的黑客噩梦，其实不要怕，因为MongoDB使用起来非常简单。驱动是应用程序用来与MongoDB服务器通信的代码。所有的驱动都保护查询功能、搜索结果、写入数据和运行数据库命令。针对MongoDB驱动开发提供的API都尽量保证所有的语言使用统一的接口。例如，所有的驱动都实现了相似的方法来保存数据到集合中，但是文档的表示类型都是基于自己语言最原始的类型定义。

[1]如果要学习 Javascript，可以阅读 http://eloquentjavascript.net 网站。Javascript 的语法与 C 语言的很像，C++、Java 和 C#都属于 C 语言语系。如果你熟悉其中一种语言，应该很容易理解 Javascript 的例子代码。

在Ruby语言中使用Ruby哈希；在Python中，字典是合适的数据结构类型；在Java中，缺少类似的语言基元类型，通常要使用Map对象表示，或者其他类似的类型。有些开发者喜欢使用ORM框架来管理这些数据的表示工作，但是实际上，MongoDB驱动已经非常强大，这些工作都是多余的[1]。

语 言 驱 动

在撰写本书时，MongoDB公司官方支持C、C++、C#、Erlang、Java、Node.js、JavaScript、Perl、PHP、Python、Scala、Ruby语言的驱动——这个列表还在增加。如果需要其他语言开发新的驱动，就很有可能已经有社区开发了，虽然并非MongoDB公司官方管理的驱动项目，但是应该也不错。

如果没有你的语言对应的社区支持的驱动，则也可以找到构建新驱动的官方文档规范http://mongodb.org。所有官方的驱动都大量使用在生产环境中，而且遵守Apache许可证，为驱动开发者提供了许多好的参考例子。

从第3章开始，我们将会深入介绍驱动如何工作，以及如何使用它们编写程序。

1.3.4 命令行工具

MongoDB包含了几个命令行工具。

- mongodump 和 mongorestore——备份和恢复数据库的工具。mongodump把数据库数据保存为原生的BSON格式，因此最适用于备份。这个工具的一大优点是适合热备份，而且非常容易使用mongorestore命令恢复。
- mongoexport 和 mongoimport——导入或者导出JSON、CSV、TSV[2]格式的数据。这在大家需要多种格式的数据时非常有用。Mongoimport还可以用来导入大数据集合，只是经常需要在导入之前调整数据模型以便于发挥MongoDB的最大优势。此时最简单的导入数据的方式就是使用自定义脚本。
- mongosniff——一个用于查看发送给数据库命令的嗅探工具。通常会把BSON转换为人

[1]【译者注】在 Java 和 C#等语言中，MongoDB 不同的客户端驱动也提供了自定义封装的文档类型。

[2] CSV 表示逗号分隔的值(comma-separated value)，意味着使用逗号把数据分割为几块。这是表示表格数据的流行方式，因为列名和行值可以列举在可读的文件中。TSV 表示制表符分割的值（tab-separated values），用制表符 Tab 来分割数据。中国 MongoDB 学习交流群 511943641

类可读的shell语句。

- mongostat——与iostat类似，这个工具用来轮训MongoDB，提供有帮助的状态信息，包括每秒的操作数(增、删、改、查等)，分配虚拟内存的数量，以及服务器的连接数量。
- mongotop——与top类似，这个工具用来轮训MongoDB，并且显示它在每个集合里花费的读取和写入数据的时间总数。
- mongoperf——帮助我们了解MongoDB实例磁盘操作的情况。
- mongooplog——展示MongoDB操作日志里的信息。
- Bsondump——把BSON文件转换为人类可读的格式，包括JSON。

我们会在第2章里详细介绍这些知识点。

1.4 为什么是 MongoDB?

Why MongoDB?

我们已经了解了为什么MongoDB是项目不错的选择的一些原因。这里，我们要更加详细地澄清一下，首先，要考虑一下MongoDB的设计目标。根据MongoDB之父的解释，它被用来设计组合键值对存储和关系数据库的最佳特性。因为简单，所以键值对存储非常快，而且容易伸缩。关系型数据库难以伸缩，至少在水平方向是这样的，但是关系型数据库拥有更加丰富的数据模型和查询语言。MongoDB在两者之间做了妥协，具备了二者的某些有用的功能。它最终的目标就是易于伸缩，存储丰富的数据结构，并且提供复杂的查询语言。

关于使用场景：MongoDB是做Web应用、分析应用的首要数据库。此外，它还比较容易存储无schema数据，也就是弱数据结构的数据。MongoDB也适用于存储无法事先知道数据结构的数据。

这些说法看起来比较虚幻。为了验证这些说法，我们来对比MongoDB和当前使用的不同数据库。接下来你会看到一些MongoDB的专门使用场景，还包括一些生产环境使用的例子。然后，我们会讨论一些使用MongoDB进行实际开发的重要考虑。

1.4.1 MongoDB 与其他数据库对比

数据库的数量非常多，而且对比所有的数据库是不现实的。幸运的是，绝大部分数据库属于

某个类别。在表1.1以及接下来的章节里，我们描述了简单而且详细的键值对存储、关系型数据库以及文档数据库，把它们与MongoDB做了对比。

表 1.1　数据库家族

	例子	数据模型	伸缩性模型	使用场景
简单的键值存储	Memcached	键值，值是二进制对象	变化的 Memcached 可以跨节点伸缩，把所有可用的 RAM 变为一个存储库	缓存、Web 网站等

续表

	例子	数据模型	伸缩性模型	使用场景
复查键值存储	HBase, Cassandra, Riak KV,Redis, CouchDB	变化的 Cassandra 使用的键值结构是列；HBase 和 Redis 存储二进制对象，CouchDB 存储 JSON 文档	最终一致性、多节点、分布式高可用和容易灾备	高吞吐量（活动源、消息队列）、缓存、Web 网站等
关系型数据库	Oracle 数据库、IBM DB2、SQL Server、MySQL、 PostgreSQL	表	垂直伸缩。限制支持集群和手动分区	需要事务的系统(银行和金融)或SQL、规范化数据模型

简单的键值存储

简单的键值存储功能如其名字所示：索引值是基于提供的key键。常见的使用场景就是缓存。例如，假设要缓存App渲染的HTML页面。此时的key可能就是页面的URL，值就是HTML页面本身的数据。注意，对于键值对存储而言，数据值是字节数组。没有强制的schema数据定义，这一点和关系型数据库不同，也没有数据类型的概念。这自然也限制了键值存储的操作：可以插入新的值，然后使用key来查询或者删除值。如此简单的系统自然也就非常快速和容易伸缩。

最有名的键值存储就是Memcached，它只在内存里存储数据，是以牺牲持久性来换取速度。它也是分布式的。Memcached节点运行在多个服务器上，也作为单个存储库，去除了跨机器节点保持高速缓存状态的复杂性。

与MongoDB相比，像Memcached这样简单的键值存储允许更快的读/写速度。与MongoDB不同，这些系统不能作为主要的存储数据库。简单的键值存储最好作为辅助手段，或者用作关系型数据库的缓存层，或者作为简单持久性的临时服务，比如工作队列。

复杂的键值存储

可以通过完善简单的键值存储模型来处理复杂的读/写模式或者提供更丰富的数据类型。此时，需要使用复杂的键值存储。其中一个例子就是亚马逊的Dynamo，在一篇非常流行的论文里有介绍，标题是“Dynamo: Amazon’s Highly Available Key-Value Store”（亚马逊高可用键值存储）。Dynamo的设计目标是在网络失败、数据中心故障时可以正常提供强壮的数据库功能，

这需要数据可以一直读/写，本质上需要数据自动化复制到各个节点上。如果一个节点失败，系统的用户——此时使用亚马逊购物车——不会经历任何的服务中断。Dynamo提供了一种系统向多个节点写入相同数据时解决不可避免冲突的方法，而且Dynamo易于伸缩。因为它是无主的——所有的节点都一样——这样比较容易理解系统是一个整体，而且可以方便地加入新节点。虽然Dynamo是个亚马逊的数据库系统，但是它的设计思想启发和影响了许多NoSQL数据库的设计，包括Cassandra、HBase、Riak KV。

通过了解谁开发了这些复杂的键值数据库，以及如何在实际项目中使用，我们可以知道这些系统如何发挥作用。我们来看下Cassandra，它实现了Dynamo的许多伸缩属性，而且受到Google的BigTable启发，提供了面向列的数据模型。Cassandra是个开源数据库，由Facebook构建，其目标是支持站内搜索功能。这个系统水平伸缩，支持索引超过50TB的数据，允许基于用户关键字搜索，属于基于用户ID索引，每条记录都由一组搜索关键字数组和用户ID数组组成，就是为了基于用户进行搜索[1]。

复杂键值存储由主流的互联网公司开发，比如Amazon、Google、Facebook，用来管理分布式系统中的超大量数据。换句话说，复杂键值存储管理着一个要求大存储与高可用的相对独立的域。因为它们采用了无主架构，系统容易使用额外的节点进行伸缩。它们选择了最终的一致性，意味着读不一定必须反映最新的写。但是为了换取弱一致性，牺牲的是写入时可能出现的节点失败。

对比MongoDB，它提供了更强的一致性、更丰富的数据模型以及辅助索引。后两者特性简单容易；键值存储可以在值里存储任意结构的数据，但是数据库不能查询这些值，除非他们被索引过。也可以使用主键进行查询，或者扫描索引的键，但是如果不使用索引，数据库对于这种查询毫无用处。

关系型数据库

我们已经介绍了很多关系型数据库知识，为了简洁起见，我们只需要讨论RDBMS（关系型数据库管理系统）与MongoDB的异同。流行的关系型数据库包括MySQL、PostgreSQL、Microsoft SQL Server、Oracle Database、IBM DB2等，有些开源，有些不开源。MongoDB和关系型数据库都可以表示丰富的数据模型。而关系型数据库使用固定格式的schema表，MongoDB属于无schema文档。绝大部分关系型数据库支持辅助索引和聚合查询。

从用户角度来看，关系型数据库最大的特性就是支持SQL查询语言。SQL是处理数据强大的

[1]参见 《Cassandra: A Decentralized Structured Storage System》，载于 http://mng.bz/5321。

工具，但是并非能完美解决所有的工作问题。某些情况下，相比MongoDB，它处理数据时更易于表示和更简单。此外，SQL在各个数据库之间的移植性很强，即使各个数据库支持有点差别。一种思考方式就是，SQL对于数据科学家和分析师来说更容易编写查询语句。MongoDB的查询语言更偏向于开发者，它们在程序里嵌入自己的查询代码。两个模型各有自己的优点和缺点，有时候还取决于个人喜好。

许多关系型数据库支持数据分析（或者数据仓库），而不仅仅是作为数据库。通常数据可以大量导入数据库，然后通过分析来支持商务智能决策问题。这个领域被HP Vertica 或Teradata Database大公司垄断了，它们都可以提供水平伸缩的SQL数据库。

现在通过在Hadoop存储的数据上运行SQL查询来分析数据的情况越来越多。Apache Hive就是一个广泛使用的工具，可以把SQL查询语句转换为Map-Reduce的工作，这提供了一种伸缩性的方式来查询大的数据集。这些查询使用了关系型模型，但是只针对慢速的分析查询，而不是应用程序内置的查询。

文档数据库

很少有数据库标示自己是文档数据库。在编写本书时，对比MongoDB最近的开源数据库是Apache的CouchDB。CouchDB的文档模型与MongoDB的有些相似，虽然数据使用了原始的JSON格式，而MongoDB使用了BSON格式。与MongoDB类似，CouchDB支持辅助索引；不同点在于，CouchDB的索引通过编写mapreduce函数代码实现，相比MySQL 和 MongoDB，其进程不仅仅使用的是声明式语法。在伸缩性方面也不一样，CouchDB不支持跨机器分区；相反，CouchDB节点只是其他节点的完整的复制。

1.4.2 使用场景和部署

坦率地说，你不会只根据数据库特性来选择数据库。我们需要知道真实行业里的成功使用案例。我们来看下MongoDB广泛定义的使用案例，以及一些生产环境下使用的例子[1]。

Web 应用

MongoDB非常适合做Web应用的主存储数据库。即使是简单的Web应用，也需要使用许多数据模型，比如管理用户、会话、App专有数据、上传以及权限等。这也可以与关系型数据库提供的表格方法很好地兼容，它也可以从MongoDB的集合和文档模型里获取好处。因为文档模

[1]最新的 MongoDB 企业名单，参见 http://mng.bz/z2CH。Google,facebook，BAT，360,，新浪微博、摩拜单车等

型可以表示更丰富的数据结构，需要的集合数量肯定比关系型数据库表的数量要少很多，因为关系型数据库的表需要规范化设计范式。此外，动态查询和索引让我们可以轻易实现绝大多数SQL支持的查询功能。最后，随着Web应用的增长，MongoDB提供了更加清晰的伸缩路径。

MongoDB可以很好地解决高吞吐量的Web网站需求，例如The Business Insider (TBI)案例。它们从2008年1月就开始用MongoDB作为主要的存储库。TBI是一个新闻网站，它每天的吞吐量超过100万独立的页面浏览量。有意思的是，除处理网站主要的内容（文章、评论、用户等）以外，MongoDB还要处理和存储实时的分析数据。这些分析数据被TBI用来生成动态心跳地图，以显示不同新闻的点击量。

敏捷开发

无论你怎么看待敏捷开发，都无法否认快速构建应用系统的热情。许多开发团队，包括Shutterfly和纽约时报，都部分选择了MongoDB，因为它们可以比关系型数据库更快速地开发应用。另外一个显著的原因就是MongoDB没有固定的数据架构定义schema，所以大量节约了花费在提交、沟通和应用schema修改上的时间。

此外，花费在将数据关系表示推进到面向对象数据模型中和优化ORM框架生成的SQL语句工作上的时间大大减少了。因此，MongoDB通常适用于实现较短的开发周期的项目，以及敏捷、中型规模的团队。

分析和日志

我们之前说过MongoDB也适用于分析和日志，而且使用MongoDB分析的应用还在增长。通常情况下，一个完善的公司会开始考虑使用MongoDB来做特殊的App分析工作。这些公司包括GitHub、Disqus、Justin.tv、Gilt Groupe等。

MongoDB的相关性分析得益于它的速度和两个特性：针对性原子更新和盖子集合。原子更新允许客户端高效地增加计数器，而且把值推进数组里。盖子集合对于日志非常有用，因为它只存储最近的文档数据。存储日志数据在数据库里与之对应的是文件系统，提供了更简单的组织和更强的查询功能。现在，用户可以使用MongoDB查询来获取日志信息，而不是`grep`或者自定义搜索工具。

缓存

许多Web应用使用缓存层来帮助快速返回内容数据。允许支持更多对象结构的数据模型（可

以把任意文档存储到MongoDB而不需要担心数据结构），结合更快的查询速度，经常允许MongoDB可以作为支持更多查询功能的缓存使用，或者直接与缓存层集成到一起。以The Business Insider网站为例，可以抛弃Memcached，直接从MongoDB处理页面请求。

可变的 Schema

你可以从https://dev.twitter.com/rest/tools/console获取一些JSON数据的例子，只要知道如何使用它即可。在获取数据后，保存为sample. Json文件，可以使用如下方式把它导入MongoDB数据库里：

```
$ cat sample.json | mongoimport -c tweets
2015-08-28T11:48:27.584+0300     connected to: localhost
2015-08-28T11:48:27.660+0300     imported 1 document
```

也可以下载一些Twitter流的例子数据，然后直接导入MongoDB集合里。因为流生成了JSON文档，在发给数据库之前不需要修改数据。Mongoimport工具可以直接把数据翻译给BSON。这意味着每个推文微博都会按照自己的格式存储，以一个独立文档存储在集合中。索引和查询内容时，不需要提前声明数据的结构。

你的应用需要调用JSON API时，有这样一个可以自动转换JSON的系统是非常棒的。在存储数据之前不用知道数据结构，而且MongoDB缺少schema约束，因此可以简化我们的数据模型。

1.5 提示和限制

Tips and limitations

对于所有这些好的功能特性，值得记住的是系统权衡和限制。在大家使用MongoDB开发项目之前，我们要先强调一些限制。这些知识都是关于MongoDB如何管理数据，以及如何使用memorymapped文件在磁盘和内存之间移动数据的。

首先，MongoDB通常用于64位系统。32位系统只能寻址4GB内存。这意味着只要你的数据集，包括元数据和存储库达到4GB，MongoDB就无法存储额外的数据。绝大部分生产环境需要的内存比这大，所以64位系统是必须的[1]。

使用虚拟内存映射的第二个结果是数据内存会根据需要自动分配。这使得在共享环境中运行数据库变得复杂。通常对于数据库服务器，MongoDB最好运行在专门的服务器上。

[1] 16EB 的内存，几乎可以满足所有的需求和目标。中国 MongoDB 学习交流群 511943641

或许关于MongoDB使用内存映射文件最重要的知识就是它在底层如何处理超过内存大小的数据集。当查询如此大的数据集时，它通常需要通过访问磁盘来获取额外的数据。结果是许多用户报告卓越的MongoDB性能，直到处理的数据超出了内存，而且查询开始变慢。这个问题不仅仅存在于MongoDB中，它也是一个常见的陷阱，值得去留意。

一个关联的问题是MongoDB用来存储集合和文档的数据结构，从数据大小的角度来看并非是最有效的。例如，MongoDB在每个文档里存储key，这意味着对于每个包含“username”字段的文档，都必须使用8个字节来存储该字段的名称。

对于SQL开发者，使用MongoDB常见的痛苦就是，它的查询语言与SQL差别很大，而且有时候确实是事实。MongoDB比绝大多数数据库更针对开发人员——不是分析员。它的哲学是查询一次编写，然后嵌入应用中。正如你将要看到的，MongoDB查询通常由JSON对象组成而不是SQL文本。这使得查询更容易创建和解析，这是个重要的考虑，但是很难为ad-hoc查询去改变。如果你是个分析员，每天要编写查询语句，你会愿意选择使用SQL语句。

最后值得一提的是，虽然MongoDB是最简单的数据库之一，可以作为单个节点运行，但是运行大规模集群还是有维护的成本的。大部分分布式数据库都有类似的问题，而MongoDB更是如此，因为它的集群需要三个配置节点来单独处理分片集群的复制问题。

在一些数据库，例如HBase中，数据存储到每个片中，每个分片数据可以在集群的任意集群上复制。MongoDB不会在所有分片节点上复制数据，而是在每个可复制集群里复制数据。分片和可复制集群是单独的概念，这样有特殊的优势，但是也意味着在配置MongoDB集群时需要单独配置和管理。

我们来快速看下MongoDB里的其他改变。

1.6 MongoDB 历史

当第一版《MongoDB in Action》出版时，MongoDB 1.8.x是最稳定的版本，2.0.0版本刚刚开始启动。对于本书的第二版，3.4.x是最稳定的版本[1]。

官方每次重大修改的版本列表如下所示。你最好使用最新的稳定版本，如果是这样，你可以忽略本列表。否则，这个列表可以帮助你决定自己的版本和本书内容的不同。这绝不是一个详尽的列表，因为篇幅限制，我们只列举了每个发布的4～5个项目。

[1] MongoDB 实际上从 2.6 直接跳到 3.0，忽略了 2.8。参考 http://www.mongodb.com/blog/post/announcing-mongodb-30，获取更多关于 3.4 的信息。

版本 1.8.X

(官方不再支持)

- 分片——分片集群由实验状态修改为产品环境准备状态。
- 可复制集——可复制集状态为产品环境准备。
- 可复制对弃用——可复制集对不再被MongoDB公司支持。
- GEO搜索——引入二维GEO索引（坐标系、2D索引）。

版本 2.0.X

(官方不再支持)

- 默认弃用日志——新版本默认弃用日志功能，日志是阻止数据冲突的重要功能。
- 查询——此版本增加了`$and`查询操作符来完善`$or`操作。
- 稀疏索引——之前的MongoDB保护每个文档的索引节点，即使文档部包括索引跟踪的字段。稀疏索引只添加包含相关字段的文档节点。这个功能显著降低了索引的大小。某些情况下还可以改善索引的性能，因为小索引可以更有效地使用内存。
- 可复制集优先级——这个版本允许指定可复制集中服务器的优先级，以便于选择新的主服务器。
- 集合级别的压缩和修复——之前的版本只能执行在单个数据库上压缩和修复；这次已扩展到单个的集合中。

版本 2.2.X

(官方不再支持)

- 聚合框架——这个改变使得数据分析和转换更加简单、高效。从某些方面而言，这个工具代替了map/reduce的部分工作；它是基于管道构建，而不是map/reduce模型（难以理解掌握）。
- TTL集合——引入了带有生命周期的集合，允许我们创建与Memcached类似的缓存模型。
- DB级别锁——此版本添加了数据库级别的锁来代替全局锁,它通过允许多个操作同时在不同的数据库发生来改善写并发。
- 标签识别分片——此版本允许节点可以使用ID来标识数据存储的物理位置。这样的应用可

以控制数据存储在集群中的位置，因此提升效率（只读节点部署在同一个数据中心），减少协作管理的问题（只能在某个国家的服务器上存储该国家需要的数据）。

版本 2.4.X

(最老的稳定版本)

- 企业版——MongoDB的第一个订阅者版本，包括额外的验证模块，可以使用Kerberos验证系统来管理登录数据。免费版包含企业版其他的所有功能。
- 聚合框架性能——改进聚合框架的性能来支持实时分析。第6章会详细介绍聚合框架。
- 文本搜索——企业级的搜索方案作为MongoDB的实验特性集成进来。第9章会介绍这个新的搜索功能。
- 增强GEO地理位置索引——此版本包括支持多边形交叉查询和GeoJSON，以及球星模型的改进，支持椭球模型。
- V8 JavaScript引擎——MongoDB以及从Spider Monkey JavaScript引擎切换到Google V8引擎，这改进了多线程操作，并且提升了基于JavaScript的MongoDB map/reduce系统性能。

版本 2.6.X

(稳定发布)

- `$text`查询——此版本添加了`$text`操作符来支持正常查询中的文本搜索。
- 聚合改进——此版本中聚合有很大的改进。可以在游标上流处理数据，也可以输出数据到集合中。除了其他特性和性能改进，还有许多新增的操作符和管道阶段。
- 为写入改进wire协议——现在大量写入将会受到更细粒度的应答。批量写入中幸亏有了每次写入的成功或者失败状态，使得写入错误可以通过网络返回给客户端。
- 新更新操作符——已经为更新操作符添加了`$mul`，它可以乘以要更新的值。
- Sharding改进——为了更好地处理特定的情况，已经改进了分片集群特性。连续块可以合并，而且重复数据留下来等到数据块迁移完成后自动清理干净。
- 安全改进——此版本支持集合级别的访问控制，还有用户角色定义。另外还改进了SSL和X509证书支持。
- 查询系统改进——查询系统的许多部分都被重构过了。这改进了性能和查询的可预测性。

- 企业模块——MongoDB企业模块改进并扩展了已有的功能，还有审计支持。

版本 3.4.6

(最新的稳定发布版本，3.6版本9月发布)

- MMAPv1存储引擎选择支持集合级别的锁。
- 可复制集选择可以有50个成员。
- 支持WiredTiger存储引擎；WiredTiger只有在MongoDB 3.0以后的64位版本可用。
- WiredTiger 3.0存储引擎提供了文档级别的锁和压缩功能。
- 可拔插存储引擎API允许第三方开发MongoDB存储引擎。
- 改进了解释功能。
- SCRAM-SHA-1验证机制。
- `ensureIndex()`函数被`createIndex()`取代，不应该再使用。

1.7 其他资源

Additional resources

本书的目标是作为教程和权威参考，所以许多语言都是为了介绍主题，并且详细地介绍这些概念。如果需要纯参考，最好的资源是MongoDB的用户手册http://docs.mongodb.org/manual。这是一个深入的数据库指南，在需要复习这些概念的时候非常有用，而且我们强烈推荐官方文档。

如果你遇到特别的MongoDB问题，可能其他人已经知道了。简单的搜索可能都会返回结果，像博客或者网站Stack Overflow (http://stackoverflow.com) ——全球最大的面向技术问答的网站。在遇到问题的时候这些都是很大的帮助，但是在用于自己的MongoDB之前要详细检查下答案。

你也可以从MongoDB IRC聊天组和用户论坛获取帮助。MongoDB公司也提供了咨询服务来帮助企业使用MongoDB数据库。许多城市有自己的MongoDB用户组，可以通过http://meetup.com查询。还有一些方式比如接触熟悉MongoDB的人来学习如何使用数据库。最后，你也可以在曼宁出版社论坛直接联系我们，《MongoDB in Action》专门的论坛为http://manning-sandbox.com/forum.jspa?forumID=677。在这里可以咨询本书中可能没有介

绍的难题，也可以指出漏洞和勘误表。发表问题吧，请不要犹豫！

1.8 总结

Summary

我们已经介绍了许多内容。总结一下，MongoDB是一个开源的、面向文档的数据库管理系统，为全新的互联网应用的数据模型和伸缩性而设计，具有动态查询和辅助索引、快速原子更新以及复杂聚合，支持自动化灾备的复制，还有水平伸缩的分片集群等特性。

虽然知识点很多，但是如果你仔细阅读，你可能现在已经急不可耐要为这些功能写代码了。介绍数据库新特性只是其中一个任务，还要在实际中使用它。幸运的是，这就是你在下两章里要学习的知识。首先，大家要熟悉MongoDB JavaScript shell，它可以很方便地与数据库服务引擎交互。其次，在第2章，会开始试验驱动，并尝试构建简单的基于MongoDB的Ruby程序。