

# RDBMS 到 MongoDB 迁移指南

注意事项和最佳实践  
2015 年 11 月

# 目录

<b>简介</b>	1
<b>为成功而组织</b>	1
<b>架构设计</b>	2
从僵化的表格到灵活且动态的 BSON 文档	3
文档模型的其他优势	4
联接集合以进行数据分析	5
定义文档架构	5
通过嵌入和引用为关系建模	5
嵌入	5
引用	6
不同的设计目标	6
索引	7
索引类型	7
使用索引优化性能	8
架构演变和对架构设计的影响	9
<b>应用程序集成</b>	10
MongoDB 驱动程序和 API	10
将 SQL 映射到 MongoDB 语法	10
MongoDB Aggregation Framework	10
适用于 BI 的 MongoDB Connector	11
MongoDB 中的原子性	11
<b>维</b> 持强一致性	12
写持久性	12
实现校验和约束	13
外键	13
文档校验	13
使用索引强制执行约束	14
将数据迁移到 MongoDB	14
<b>大规模的运维敏捷性</b>	15
支持您的迁移 : MongoDB 服务	15
MongoDB 大学	16
社区资源和咨询	16
<b>总结</b>	16
<b>我们可以提供帮助</b>	16

# 简介

关系型数据库经过三十多年的时间已成为企业数据管理的基础。

但是我们如今构建和运行应用程序的方式外加新数据源的迅猛增长和用户负载的不断增长，正在超出关系型数据库所能承受的极限。这可能会抑制业务灵活性、限制扩展性并带来预算压力，从而引起越来越多的组织迁移 to MongoDB 或 NoSQL 数据库等替代产品。

如图 1 所示，来自各行业的企业已将无数的应用程序从关系型数据库管理系统 (RDBMS) 成功迁移到 MongoDB。

本指南面向希望了解如何从 RDBMS 迁移到 MongoDB 的项目团队而提供。我们提供图 2 所示的分步路线图。

本文档提供了许多链接，可帮助将用户引导至相应的联机资源。有关特定主题的最新详细信息，请参阅 联机文档。

# 为成功而组织

在考虑技术和体系结构前，要获得成功的秘诀在于应用程序的所有关键利益相关者的参与，包括业务线、开发人员、数据架构师、DBA 和系统管理员。在某些组织中，这些角色可能组合在一起。

项目团队应共同合作来定义业务和技术目标、时间线和责任，定期开会以监视进度并处理任何问题。

MongoDB 和社区中有多种服务和资源可帮助您掌握 MongoDB 技能并提高熟练度，包括基于 Web 的免费培训，以及支持和咨询。有关详细信息，请参阅本指南后面的“MongoDB 服务”部分。

# 架构设计

从关系型数据库迁移到 MongoDB 的最根本改变是为数据建模的方式。

组织	迁移自	应用程序
eHarmony	Oracle 和 Postgres	客户数据管理和分析
Shutterfly	Oracle	Web 和移动服务
Cisco	多个 RDBMS	分析、社交网络
Craigslist	MySQL	存档
Under Armour	Microsoft SQL Server	eCommerce
Foursquare	PostgreSQL	社交、移动网络平台
MTV Networks	多个 RDBMS	集中式内容管理
Buzzfeed	MySQL	实时分析
Verizon	Oracle	单一视图、员工系统
The Weather Channel	Oracle 和 MySQL	移动网络平台

图 1：案例研究

和任何数据建模实践一样，每个用例都不同，但有一些常规注意事项适用于大多数架构迁移项目。

在探索架构设计前，图 3 提供了关于将关系型数据库中的术语转换为 MongoDB 术语的有用参考。

对于数据架构师、开发人员和 DBA 来说，要进行架构设计需要改变视角：

- 从将数据平摊到严格二维行列表结构的传统关系型数据模型。
- 到具有嵌入式子文档和数组的丰富而动态的文档数据模型。

Figure 2

RDBMS	MongoDB
数据库	数据库
表	集合
行	文档
索引	索引
联接	嵌入式文档、文档引用或 \$lookup，用于将来自不同集合的数据组合起来

图 3：术语转换

从僵化的表格到灵活且动态的 BSON 文档

我们如今使用的许多数据都具有复杂的结构，可以使用

图 2：迁移路线图（JavaScript 对象表示法）文档而不是表进行更高效的建模和呈现。

MongoDB 使用名为 BSON (Binary JSON) 的二进制表示形式存储 JSON 文档。BSON 编码扩展了流行的 JSON 表示形式以包含其他数据类型，如 int、long 和浮点。

借助子文档和数组，JSON 文档还在应用程序级别上与对象的结构一致。这使开发人员可以轻松地将应用程序使用的数据映射到数据库中与其关联的文档。

相反，尝试将数据的对象表现形式映射到 RDBMS 的表格表现形式会降低部署速度。添加对象关系映射器 (ORM) 可能会增加额外的复杂度，因为这会降低改进架构和优化队列以满足新应用程序要求的灵活性。

项目团队应从考虑应用程序的要求开始进行架构设计过程。为数据建模的方式应充分利用文档模型的灵活性。在架构迁移中，将关系型数据库的平面架构镜像复制到文档模型可能很容易。但是，此方法会抵消文档模型的丰富嵌入式数据结构所带来的优势。例如，在两个 RDBMS 表中属于父子关系的数据通常会收缩（嵌入）到 MongoDB 中的单个文档中。

在图 4 中，RDBMS 使用“Pers\_ID”字段将“Person”表与“Car”表联接以使应用程序可以报告每辆汽车的所有者。使用文档模型时，嵌入式子文档和数组可以通过将相关字段组合到单个数据结构中来有效地预联接数据。进行传统规范化且分布在单独的表上的行和列现在可以共同存储在单个文档中，当应用程序必须检索完整记录时便无需联接单独的表。

在 MongoDB 中为相同的数据建模使我们可以创建一个这样的架构：我们直接在“Person”文档内为每辆汽车嵌入子文档的数组。

```
{  
  first_name: "Paul",  
  surname: "Miller",  
  city: "London",  
  location: [45.123,47.232],  
  cars: [  
    { model: "Bentley",  
      year: 1973,  
      value: 100000, ...},  
    { model: "Rolls Royce",  
      year: 1965,  
      value: 330000, ...},  
  ]  
}
```

Figure 4

图 4：关系型架构，平面二维表

Figure 5

图 5：预联接数据以将 5 个 RDBMS 表收缩为 2 个 BSON 文档。

在这一简单示例中，关系型模型仅由两个表组成。（在现实中，大多数应用程序将需要数十个、数百个甚至数千个表。）此方法不会反映架构师考虑数据的方法，也不会影响开发人员编写应用程序的方式。文档模型使数据能够以更自然和直观的方式呈现。

为了进一步演示关系型数据库和文档模型之间的区别，请考虑图 5 中的博客平台示例。在此示例中，应用程序依靠 RDBMS 将五个单独的表格联接起来，以便生成博客条目。在 MongoDB 中，所有博客数据都包含在单个文档中，通过对包含博客和评论作者的用户文档的单个引用链接起来。

## 文档模型的其他优势

除了在数据库级别上使数据更自然地呈现，文档模型还提供性能和可扩展性的优势：

- 可以通过对数据库的单个调用访问完整文档，而不是必须联接多个表才能响应查询。MongoDB 文档在物理上存储为单个对象，只需从内存或磁盘读取一次即可。另一方面，RDBMS 联接需要从多个物理位置多次读取。
- 由于文档是自包含的，因此将数据库分发到多个节点（此过程称为分片）变得更加简单，并且使用户能够在商品硬件上实现大规模水平扩展性。DBA 不再需要担心因执行跨节点联接以从不同的表收集数据（如果它们在现有 RDBMS 中可行）而获得性能惩罚。

## 联接集合以进行数据分析

通常对操作型数据库采取反规范化数据建模方法最具优势，在单个操作中读取或写入整个记录的效率远超任何适度的存储需求增长程度。但是在某些示例中，规范化数据更为有利，在需要将来自多个源的数据混合以进行

分析时尤其如此，MongoDB 3.2 使用 \$lookup 阶段添加了该功能（该阶段位于 MongoDB Aggregation Framework 中）。

Aggregation Framework 是基于数据处理管道概念建模的数据聚合的管道。文档将进入多阶段管道，该管道将文档转换为聚合结果。管道由多个阶段组成；每个阶段在文档通过时对其进行转换。

尽管不提供一组像某些 RDBMS 一样丰富的联接操作，但 \$lookup 提供等值左外联接，该联接为选择分析用例提供了便利。左外部等值联接将来自“右”集合的文档匹配并嵌入到来自“左”集合的文档中。

例如，如果左集合包含购物车应用程序中的 order 文档，则 \$lookup 运算符可以匹配来自这些文档的 product\_id 引用以嵌入来自 products 集合的匹配产品详细信息。

使用 \$lookup 以及其他聚合阶段的样例可以在博客“Joins and Other Aggregation Enhancements”（联接和其他聚合增强）中找到。

## 定义文档架构

应由应用程序的数据访问模式控制架构设计，同时还需要对以下方面进行具体了解：

- 数据库操作的读取/写入率以及针对其中之一优化性能是否比另一个更重要
- 数据库执行的查询和更新的类型
- 数据的生命周期和文档的增长率

首先，项目团队应记录对应用程序数据所执行的操作，从而比较：

1. 关系型数据库当前如何实现这些操作；
2. MongoDB 可以如何实现它们。

图 6 显示了此类运用的示例。

此分析基于要针对应用程序执行的查询和操作，帮助确定用于应用程序数据和工作负载的理想文档架构。

项目团队还可以通过分析 RDBMS 保留的日志来确定现有应用程序的最常用查询。此分析可确定最常在一起受到访问、并因此可能一起存储在单个 MongoDB 文档中的数据。此过程的示例记录在 Apollo Group 从 Oracle 到

应用程序	RDBMS 操作	MongoDB 操作
创建产品记录	INSERT 到 (n) 个表 (产品说明、价格、制造商等)	insert() 到 1 个文档
显示产品记录	SELECT 并 JOIN (n) 个产品表	find() 单个文档
添加评论	INSERT 到“review”表， 产品记录为外键	insert() 到 “review”集合，引用产品文档
更多操作	..... ...	.....

图 6：分析查询以设计最佳架构

MongoDB 的迁移（开发基于云的全新学习管理平台时）中。

## 通过嵌入和引用为关系建模

确定何时嵌入文档或在不同集合中的单独文档之间创建引用是特定于应用程序的注意事项。但是，有一些常规注意事项也可指导在架构设计期间作出的决策。

### 嵌入

具有一对一或一对多关系的数据（其中“多”对象始终通过其父文档的上下文显示或查看）是嵌入单个文档中的自然候选项。数据所有权和包含的概念可以通过嵌入进行建模。使用上述产品数据示例时，产品定价（当前和历史定价）应嵌入产品文档中，因为它由该特定产品所有并包含在其中。如果删除该产品，定价则变为不相关。

架构师还应嵌入需要一起进行小幅修改的字段。（有关详细信息，请参考本指南的“应用程序集成”部分。）

并非所有 1:1 关系都应嵌入单个文档中。在以下情况下应在不同位置的文档之间进行引用：

- 文档经常被读取，但包含极少受到访问的嵌入文档。一个可能的示例为嵌入每年常规报告副本的客户记录。嵌入报告仅增加集合的常驻内存要求（工作集）

- 文档的一部分经常更新，并且大小不断增长，同时文档的其余部分相对处于静态
- 文档大小超过 MongoDB 的当前 16MB 文档限制

## 引用

引用支持数据规范化，并且可以提供比嵌入更多的灵活性。但是应用程序将发出后续查询来解析引用，从而需要额外地来回访问服务器。

引用的实现方式通常是将一个文档的 `_id` 字段<sup>1</sup> 作为引用保存在相关文档中。然后由应用程序执行第二次查询以返回引用的数据。

### 应使用引用的情况：

- 当嵌入无法提供足够的读取性能优势来超过数据重复的影响时
- 从许多不同的源引用对象时
- 要表示复杂的多对多关系
- 要为较大的、分层数据集建模。

聚合管道中的 `$lookup` 阶段可用于将引用与第二个集合中的 `_id` 匹配，以在结果集中自动嵌入引用的数据。

## 不同的设计目标

比较这两个设计选项（嵌入子文档与在文档之间引用）突出了关系型数据库和文档数据库之间的根本区别。

- RDBMS 可优化数据存储效率（因为在构想它的时代，存储是系统中最昂贵的组件）
- MongoDB 的文档模型针对应用程序访问数据的方式进行优化（因为开发人员的时间和上市速度现在比存储更昂贵）

数据建模注意事项、模式和示例（包括嵌入与引用关系）在相关 文档 中进行了更详细的讨论。

## 索引

在任何数据库中，索引都是最大的一个可调节性能因素，因此它对于架构设计是必不可少的。

MongoDB 中的索引很大程度上对应于关系型数据库中的索引。MongoDB 使用 B 树索引，并且本身支持二级索引。因此，来自 SQL 背景的人员可以立即熟悉它。

应用程序查询的类型和频率将对索引选择产生影响。和所有数据库一样，索引不是没有代价的：它会对写入和资源（磁盘和内存）使用施加开销。

## 索引类型

MongoDB 具有丰富的查询模型，可在数据访问方式方面提供灵活性。默认情况下，MongoDB 会在文档的 `_id` 主键字段上创建索引。

所有用户定义的索引都是二级索引。任何字段都可用于二级索引，包括数组内的字段。

MongoDB 的索引选项包括：

- **复合索引**。借助索引相交，MongoDB 可使用多个索引来满足查询。在运行即席查询时，此功能非常有用，因为事先通常并不了解访问模式。当基于多个断言访问数据的查询已知时，使用复合索引性能更高，因为它使用单个索引结构来保留对多个字段的引用。例如，请考虑存储客户相关数据的应用程序。此应用程序可能需要基于姓、名和居住省份来查找客户。借助对姓、名和居住省份的复合索引，查询可以一起使用所指定的这三个值高效查找人员。复合索引的其他优势是索引内的任何先导字段都可以使用，因此单个字段上可能需要较少的索引：此复合索引还可优化按姓或姓和名查找客户的查询。
- **唯一索引**。通过将索引指定为唯一，MongoDB 将拒绝对现有文档插入新文档或进行更新，因为这会导致索引字段产生重复值。默认情况下，并非所有索引都设置为唯一。如果复合索引指定为唯一，则值的组合必须唯一。
- **数组索引**。对于包含数组的字段，每个数组值都存储为单独的索引项。例如，介绍产品的文档可能包含一个存储其主要属性的字段。如果该属性字段上存在索引，则每个属性都编入索引，并且对该属性字段的查询可由此索引优化。创建数组索引不需要特殊的语法：如果字段包含数组，它将以数组索引的形式编入索引。

1. 一种在 MongoDB 文档内用作主键的必需唯一字段，由驱动器自动生成或由用户指定。

- **TTL 索引**。在某些情况下，数据应自动过期。Time to Live (TTL) 索引允许用户指定数据库在自动删除数据前等待的时间段。TLL 索引的常见用例是为用户操作（如点击流）保留历史记录（例如，最近 100 天）滚动窗口的应用程序。
- **地理空间索引**。MongoDB 提供地理空间索引，以优化与二维空间内的位置相关的查询，如地球的投影系统。这些索引可使 MongoDB 优化对包含以下多边形或点的文档进行的查询：离给定的点或直线最近；在圆形、矩形或多边形内；或与圆形、矩形或多边形相交。
- **稀疏索引**。稀疏索引仅包含含有指定字段的文档条目。由于 MongoDB 允许数据模型因文档而异，因此通常某些字段仅存在于所有文档的子集中。当所有文档中都不存在字段时，稀疏索引允许更小、更高效的索引。
- **部分索引**。MongoDB 3.2 引入了部分索引，此类索引可视为稀疏索引的更灵活版本，其中 DBA 可以指定要接受检查以确定某个文档是否应包含在特定索引中的表达式。例如，对于“订单”集合，可能只有活动的订单需要针对状态和交货公司的索引，因此可以使索引成为对 {orderState: "active"} 的条件索引，从而减少对内存、存储和写入性能的影响，同时仍然优化对活动订单的搜索。
- **哈希索引**。哈希索引计算字段值的哈希，并为哈希值编制索引。此索引的主要用途是支持基于哈希的分片，它是一种跨分片的简单且均匀的文档分布方式。
- **文本搜索索引**。MongoDB 提供专业化的索引，用于为词干提取、词汇切分和停止词使用特定于语言的高级语言规则的文本搜索。使用文本搜索索引的查询将以相关性顺序返回文档。每个集合最多可以有一个文本索引，但可以包含多个字段。

MongoDB 的存储引擎全部支持所有索引类型，并且可以创建对 JSON 文档任何部分的索引（包括子文档和数组元素），从而使它们比 RDBMS 所提供的索引功能更强大。

## 使用索引优化性能

MongoDB 的查询优化器通过偶尔运行备用查询计划和选择具有最佳响应时间的计划来以经验为主地选择索引。可使用 cursor.hint() 方法覆盖查询优化器。

和关系型数据库一样，DBA 查看查询计划并确保常见查询由明确定义的索引提供服务的方法是使用 explain() 函数，该函数会报告以下信息：

- 返回的文档数
- 使用了哪个索引（如果有）
- 查询是否为覆盖查询，这意味着无需读取文档即可返回结果
- 是否执行了常驻内存排序，这表示索引是有益的
- 扫描的索引项数
- 读取的文档数
- 查询进行解析所需的时间（以毫秒为单位报告）
- 曾受到访问然后拒绝的备用查询计划

MongoDB 提供多种日志记录和监视工具来确保已为集合编制合适的索引并已优化查询。这些工具可以并且应该在开发和生产中使用。

MongoDB 数据库探查器通常在加载测试和调试期间使用，用于记录所有数据库操作或仅记录持续时间超过可配置阈值（默认值为 100 毫秒）的事件。分析数据存储在有上限的集合中，可以在此集合中轻松搜索相关事件，查询此集合通常比分析日志文件更轻松。

作为 MongoDB Ops Manager 和 Cloud Manager 平台的一部分提供的新 Visual Query Profiler 为运维团队和 DBA 提供分析特定查询或查询系列的快捷方法。Visual Query Profiler（如图 7 所示）显示查询和写入延迟如何随着时间的推移而变化，这样可以简单地借助常见访问模式和特征来确定较缓慢的查询并确定任意延迟峰值。

Visual Query Profile 将分析它所收集的数据来提供新索引的建议，用户可以创建这些新索引来提升查询性能。经确定后，这些新索引需要在生产系统中进行全面部署，并且 Ops/Cloud Manager 将自动执行该过程（执行滚动索引生成），从而避免对应用程序产生任何影响。

尽管可能没有必要在项目的开端为数据库分片，但假设将来需要扩展（例如，由于数据增长或应用程序受欢迎）始终是良好的做法。在架构设计阶段期间定义索引键还有助于确定在为应用程序透明扩展实现 MongoDB 自动分片时可使用的键。

Figure 7

### 图 7：MongoDB Ops Manager 中的可视查询分析

## 架构演变和对架构设计的影响

MongoDB 的动态架构提供了相对于关系型数据库的主要优势。

可以在未先定义结构（即文档字段及其数据类型）的情况下创建集合。给定集合中的文档无需全部具有同一组字段。只需添加新字段或删除现有字段即可更改文档的结构。

请考虑客户记录的示例：

- 有些客户有多个办公地点和业务线，而有些没有。
- 每个客户内部的联系人数量可能不同
- 存储在这些联系人上的信息可能不同。例如，有些可能有值得监视的社交媒体源，而有些没有。
- 每个客户都可能从其供应商处购买或订阅不同的服务，并且各自都有自己的联系人组。

在僵化的二维关系型数据库架构中为现实世界的差异建模可能既复杂又令人费解。在 MongoDB 中，支持文档之间的差异是 BSON 文档的一项基本、无缝的功能。

MongoDB 灵活且动态的架构意味着架构开发和持续的演变很容易。例如，使用关系型数据库开展新开发项目的开发人员和 DBA 在编写任何代码前都必须先指定数据库架构。这最少需要数天；通常需要数周或数月。

MongoDB 使开发人员可以通过迭代和敏捷的方法发展架构。开发人员可以开始编写代码，并在创建对象后保留它们。并且当他们添加更多功能时，MongoDB 将继续存储更新的对象，而无需执行成本高昂的 ALTER TABLE 操作或从头开始重新设计架构。

这些优点还延伸到在生产中保留应用程序。在使用关系型数据库时，应用程序升级可能需要 DBA 在数据库中添加或修改字段。这些更改需要在开发、DBA 和运维团队之间进行规划以同步应用程序和数据库升级，需要他们一致决定在何时安排必要的 ALTER TABLE 操作。

由于 MongoDB 允许架构动态演变，所以此类操作只需升级应用程序即可，通常无需为 MongoDB 执行任何操作。发展应用程序很简单，并且项目团队可以提高敏捷性并缩短上市时间。

在 DBA 或开发人员决定对文档结构强制执行某些约束时，可以添加文档校验规则，本指南将在稍后对此提供进一步的详细信息。

## 应用程序集成

架构设计完成后，项目可以前进到使用 MongoDB 驱动程序和工具将应用程序与数据库集成。

DBA 还可以配置 MongoDB 以满足应用程序对数据一致性和持久性的要求。下面介绍其中每一个领域。

### MongoDB 驱动程序和 API

轻松使用和开发人员效率是 MongoDB 的两个核心设计目标。

基于 SQL 的 RDBMS 和 MongoDB 之间的一个基本区别是，MongoDB 接口作为特定编程语言的 API 内的方法实现，与完全独立的基于文本的语言（如 SQL）相反。这一区别外加 MongoDB 的 BSON 文档模型和面向对象编程中使用的数据结构共同让应用程序集成变得更加简单。

MongoDB 具有面向大部分流行语言的惯用驱动程序，包括 MongoDB 开发和支持的十一一种驱动程序（例如 Java、Python、.NET 和 PHP）以及超过三十种社区支持的驱动程序。

MongoDB 的惯用驱动程序最大程度减少了新开发人员的入门时间，并简化了应用程序开发过程。例如，Java 开发人员可以在 Java 中以原生方式针对 MongoDB 简化代码，对于 Ruby 开发人员、PHP 开发人员等同样如此。创建这些驱动程序的开发团队是其给定语言领域的专家，并且了解程序员在这些语言内的工作偏好。

### 将 SQL 映射到 MongoDB 语法

对于熟悉 SQL 的开发人员，了解核心 SQL 语句（如 CREATE, ALTER, INSERT, SELECT, UPDATE 和 DELETE）如何映射到 MongoDB API 很有用。文档包括

**比较图** 并附带示例，以帮助过渡到 MongoDB 查询语言结构和语义。此外，MongoDB 还提供一系列广泛的 **高级查询运算符**。

## MongoDB Aggregation Framework

在任何数据库内，聚合数据都是一项重要的功能，并且是 RDBMS 的强项。

许多 NoSQL 数据库都没有聚合功能。因此，迁移到 NoSQL 数据库在传统上迫使开发人员开发解决方法，例如：

1. 在其应用程序代码内生成聚合，因而增加复杂性并影响性能。
2. 将数据导出到 Hadoop 以针对数据运行 MapReduce 作业。这也将急剧增加复杂性、跨多个数据存储复制数据，并且不支持实时分析。
3. 在 NoSQL 数据库本身内编写本机 MapReduce 操作（如果可用）。

MongoDB 本身在数据库内提供 Aggregation Framework，这会向 GROUP BY 和相关 SQL 语句提供相似的功能。

在使用 Aggregation Framework 时，集合中的文档会通过聚合管道，文档在各阶段中进行处理。表达式基于在输入文档上执行的计算产生输出文档。在 \$group 阶段中使用的累加器表达式在文档通过管道的过程中保留状态（例如，总数、最大值、最小值、平均值、标准偏差和相关数据）。

在 [SQL 到 Aggregation 映射图](#) 中显示的一些示例演示了如何在 MongoDB 的 Aggregation Framework 中处理 SQL 中的查询。为了实现更复杂的分析，MongoDB 还为分片和未分片集合中的 MapReduce 操作提供原生支持。

## 商业智能集成 - 适用于 BI 的 MongoDB Connector

在日益增长的自助分析要求、基于实时操作数据的更快速发现和预测以及集成多结构和流数据集需求的推动下，BI 和分析平台是最快速增长的软件市场之一。

为了应对这些要求，存储在 MongoDB 中的现代应用程序数据集首次可以通过基于 SQL 的行业标准 BI 和分析

平台轻松地进行探索。借助 BI Connector，分析师、数据科学家和商业用户现在可以使用在数百万企业内部署的相同 BI 工具，来无缝实现 MongoDB 中托管的半结构化和非结构化数据以及其 SQL 数据库中传统数据的可视化效果。

基于 SQL 的 BI 工具（如 Tableau）期望连接到具有呈现表格数据的固定架构的数据源。在与 MongoDB 的动态架构和丰富多维的文档合作时，这将面临一个挑战。为了使 BI 工具将 MongoDB 作为数据源查询，BI Connector 将执行以下操作：

- 为 BI 工具提供要可视化的 MongoDB 集合架构。用户可以查看架构输出以确保数据类型、子文档和数组正确呈现
- 将 BI 工具发出的 SQL 语句转换为等效的 MongoDB 查询，然后将后者发送给 MongoDB 以进行处理
- 将返回的结果转换为 BI 工具预计的表格形式，然后该工具可以基于用户要求来直观呈现数据

此外，一些商业智能 (BI) 供应商还开发了将 MongoDB 与其套件（不使用 SQL）以及传统关系型数据库集成的连接器。此集成提供报告、即席分析和仪表板，支持跨多个数据源进行可视化和分析。集成适用于众多供应商的工具，包括 Actuate、Alteryx、Informatica、JasperSoft、Logi Analytics、MicroStrategy、Pentaho、QlikTech、SAP Lumira 和 Talend。

## MongoDB 中的原子性

关系型数据库通常具有发达的数据完整性功能，包括 ACID 事务和约束强制。用户当然不希望在转移到新类型的数据数据库时牺牲数据完整性。借助 MongoDB，用户可以保留关系型数据库的许多功能，即使这些功能的技术实现可能有所不同。

MongoDB 写入操作是文档级别的 ACID，其中包括在原子级别上更新嵌入数组和子文档的功能。通过将相关字段嵌入单个文档内，用户可以获得与传统 RDBMS 相同的完整性保证，但后者必须同步成本高昂的 ACID 操作并在单独的表格上维持引用完整性。

## 图 8：通过 MongoDB 中生成的强大可视化效果来发现新的见解

MongoDB 中的文档级原子性可在文档更新时确保完全隔离；任何错误都可能导致操作回退，并且客户端会收到文档的一致视图。

尽管单文档原子操作具有强大的功能，但仍然可能出现需要多文档事务的情况。解决此问题有多种方法，包括使用 `findAndModify` 命令，该命令允许文档在原子级别上更新，并在同一往返访问中返回。`findAndModify` 是功能强大的基元，用户可以以它为基础生成更复杂的服务协议。例如，用户经常生成原子软状态锁、作业队列、计数器和状态机，有助于协调更复杂的行为。另一个备选项需要实现一个两阶段提交来实现类事务的语言。相关 [文档](#) 介绍了如何在 MongoDB 中执行此操作及其用法的重要注意事项。

## 维持强一致性

默认情况下，MongoDB 将所有读取操作指向主服务器，从而确保强一致性。此外，默认情况下，从 MongoDB 复制集内的二级服务器进行任何读取最终都将一致，非常类似于关系型数据库中的主/从复制。

管理员可以将二级复制集成员配置为使用 MongoDB 的 [读取首选项](#) 处理读取流量，这可以控制客户端的读取操作路由到复制集成员的方式。

## 写持久性

MongoDB 使用写关注来控制写保证的级别以提高数据持久性。可配置选项从简单的“发后不理”操作扩展到等待来自全局分布的多个复制集成员进行确认。

对于宽松的写关注，应用程序可以向 MongoDB 发送写操作，然后继续处理其他请求而无需等待数据库的响应，从而提供最佳性能。此选项对日志记录等应用程序非常有用，在这些应用程序中，用户通常会分析数据的趋势，而不是离散的事件。

对于较强的写关注，写操作一直等待 MongoDB 确认操作为止。这是 MongoDB 的默认配置。MongoDB 提供多个级别的写关注来应对特定的应用程序要求。例如：

- 应用程序等待主服务器的确认（默认）。

- 或者写操作还将复制到一个二级服务器。
- 或者写操作还将复制到大部分二级服务器。
- 或者写操作还将复制到所有二级服务器，即使它们部署在不同的数据中心内也是如此。（用户应仔细评估网络延迟在此方案中的影响）。

Figure 9

## 图 9：按操作配置持久性

写关注还可用于保证更改在得到确认前保留到磁盘。

写关注在驱动程序中进行配置，并且高度细化：可按操作、按集合或针对整个数据库进行配置。用户可以在相关 [文档](#) 中了解有关写关注的详细信息。

MongoDB 使用面向磁盘日志的预写日志记录来保证写操作持久性并提供故障恢复能力。

在将更改应用到数据库前（无论它是写操作还是索引修改），MongoDB 会将更改操作写入日志。如果服务器发生故障或 MongoDB 在可以将更改从日志写入数据库前遇到错误，可以重新应用记入日志的操作，从而在服务器恢复时维持一致的状态。

## 实现校验和约束

### 外键

如前所述，MongoDB 的文档模型通常可以通过将数据嵌入在原子级别上更新的单个 BSON 文档内来消除联接的需要。这一相同数据模型还可减少对外键完整性约束的需求。

### 文档校验

动态架构带来了出色的敏捷性，但可通过实现控制来保持数据质量也很重要，在数据库支持多个应用程序或集成到流入上游和下游系统中的较大数据管理平台时尤其如此。MongoDB 在数据库内提供文档校验，而不是将这些控制的强制执行重新降级到应用程序代码层面。用户可以对文档结构、数据类型、数据范围和是否存在必填

字段执行检查。因此，DBA 可以应用数据管理标准，同时开发人员保留灵活文档模型的优势。

对于任何集合，自定义文档的哪些部分已校验 和哪些部分未校验 具有很大的灵活性，这不同于 RDBMS，后者必须定义和强制执行一切操作。对于任何键而言，检查以下事项都会较为妥当：

- 它是否存在
- 如果它存在，值是否属于正确的类型
- 值是否采用特定的格式（可以使用正则表达式检查字符串的内容是否与特定模式匹配：例如，它是否是格式正确的电子邮件地址）
- 值是否落在给定范围内

例如，假设需要在 contacts 集合上强制执行以下检查：

- 出生年份不晚于 1994 年
- 文档包含电话号码和/或电子邮件地址
- 当存在时，电话号码和电子邮件是字符串

可通过定义此文档校验规则来实现此目标：

```
db.runCommand({  
    collMod: "contacts",  
    validator: {  
        $and: [  
            {year_of_birth: {$lte: 1994}},  
            {$or: [  
                {phone: { $type: "string" }},  
                {email: { $type: "string" }}  
            ]}  
        ]  
    }  
})
```

将校验检查添加到集合对任何熟悉 MongoDB 的开发人员或 DBA 来说都非常直观，因为文档校验使用标准 MongoDB 查询语言。

## 使用索引强制执行约束

如架构设计部分中所述，MongoDB 本身支持唯一索引，这可以检测到任何尝试将重复值加载到集合中的插入操作并引发错误。有一个 教程 可供参考，该教程介绍如何创建唯一索引并从现有集合中消除重复项。

Figure 10

## 图 10：数据迁移的多种选择

# 将数据迁移到 MongoDB

项目团队可以通过多种选择将数据从现有关系型数据库导入 MongoDB。选择的工具应取决于项目的阶段和现有环境。

许多用户都创建自己的脚本，用于将源数据转换为分层 JSON 结构，该结构可使用 mongoimport 工具导入到 MongoDB 中。

在将数据从关系型数据库迁移到 MongoDB 时，Extract Transform Load (ETL) 工具也是常用的工具。包括 Informatica、Pentaho 和 Talend 在内的一些 ETL 供应商都已开发 MongoDB 连接器，用于支持从源数据库提取数据、将数据转换为目标 MongoDB 架构、将数据分阶段加载到文档集合中的工作流。

许多迁移都涉及到与新的 MongoDB 数据库并行运行现有 RDBMS，从而逐渐转移生产数据：

- 当从 RDBMS 检索记录时，应用程序会采用所需的文档架构将它们写回到 MongoDB。
- 可以使用一致性检查器（例如使用 MD5 校验和）来校验迁移的数据。
- 所有新创建或更新的数据都仅写入 MongoDB。

Shutterfly 使用了此类渐进式方法将 60 亿个图像的元数据和 20TB 的数据从 Oracle 迁移到 MongoDB。

当新的应用程序功能通过 MongoDB 实现时或者当传统 RDBMS 不利于多个应用程序的运行时，可以使用渐进式迁移。仅迁移要现代化的应用程序可使团队将项目划分为更容易管理且敏捷的部署冲刺。

渐进式迁移消除了中断服务提供的需要，同时还提供故障回复，以应对必须恢复到传统数据库的情况。

许多组织都从他们的源系统创建源，将每日更新从现有 RDBMS 转储到 MongoDB 以运行并行操作，或执行应用程序开发和加载测试。在使用此方法时，考虑如何处理对源系统中数据的删除很重要。我们的解决方案是在

MongoDB 中创建“A”和“B”目标数据库，然后使它们交替接收每日源。在此方案中，数据库 A 接收一个每日源，然后应用程序将下一天的源切换到数据库 B。与此同时删除现有数据库 A，以便当下一次源发送到数据库 A 时，可创建一个源数据库的全新实例，从而确保同步删除源数据。

若要详细了解每种方法以及包括为 ETL 使用 Apache Hadoop 在内的其他方法，请观看 [Migration Patterns 网络讲座（迁移模式）](#)。

## 大规模的运维敏捷性

到目前为止讨论的注意事项都属于数据架构师、开发人员和 DBA 的领域。但是，无论数据模型多么精巧或索引多么高效，如果数据库无法在大规模上可靠运行或进行高效管理，一切都毫无意义。

迁移规划的最后一组注意事项应侧重于运维问题。

我们提供了 [MongoDB 运维最佳实践指南](#)，它是用于详细了解此关键领域的权威参考。

此运维指南讨论的内容如下：

- 使用 MongoDB Ops Manager 或 MongoDB Cloud Manager 进行管理、监视和备份（这是在您自己的数据中心或公有云内运行 MongoDB 的最佳方式），并结合使用 mongotop、mongostat 和 mongodump。
- 使用 MongoDB 复制集实现高可用性，从而提供从故障自愈恢复的能力并支持零宕机时间的计划维护。
- 使用跨商品服务器集群的 MongoDB 自动分片（分区）实现扩展性，保持应用程序透明度。
- 具有内存、磁盘和 CPU 最佳配置的硬件选择。
- 安全性，包括 LDAP、Kerberos 和 x.509 身份验证、字段级别访问控制、用户定义角色、审核、未送达和静止数据的加密以及用于保护数据库的深度防护策略。

## 支持您的迁移：MongoDB 服务

MongoDB 和社区提供各种资源和服务，通过帮助用户掌握 MongoDB 技能并提高熟练度来支持迁移。MongoDB

服务包括培训、支持、论坛和咨询。若要详细了解从开发到生产整个过程的支持，请参考下面的“我们可以提供帮助”部分。

## MongoDB 大学

课程适用于开发人员和 DBA：

- 基于 Web 的免费课程，授课超过 7 周，由讲座、作业和论坛提供支持，可与指导者和其他学生互动。已经有超过 350000 名学生报名参加此类课程。
- 公开培训活动，在 MongoDB 提供的场所中举办。
- 私人培训，针对组织的特定要求进行定制，在组织地点授课。

[了解详细信息](#)。

## 社区资源和咨询

除了培训，项目团队还可以利用多种其他资源和服务：

- 技术资源，面向社区，通过 [Google Groups](#)、[StackOverflow](#) 和 [IRC 上的论坛提供](#)
- 咨询包，包括运行状况检查、定制咨询以及与专门的技术客户经理进行交流。有关详细信息，请参阅下面的“我们销售的产品”部分。我们提供了 [Database Modernization 服务](#)，用于评估 MongoDB 是否适合您的应用程序并包含一份详细报告，该报告将总结应用程序要求、提供适合度评估结果以及有关迁移到 MongoDB 的建议（如适用）。

## 总结

遵循本指南中概述的最佳实践可帮助项目团队减少数据迁移的时间和风险，同时可使他们充分利用 MongoDB 和文档模型的优势。这样做，便可以开始实现更加敏捷、可扩展且经济高效的基础设施，在应用程序上进行前所未有的创新。

# 我们可以提供帮助

我们是 MongoDB 专家。超过 2000 家组织依靠我们的商业产品，包括创业公司和三分之一以上的《财富》100 强企业。我们提供的软件和服务让您的生活变得更加简单：

MongoDB Enterprise Advanced 是在数据中心内运行 MongoDB 的最佳方式。它是一款通过精心设计集成高级软件、支持、认证和其他服务的产品，专为您的业务开展方式而设计。

MongoDB Cloud Manager 是在云中运行 MongoDB 的最简方法。它使 MongoDB 成为您最不用担心且最喜欢管理的系统。

MongoDB Professional 帮助您管理部署，使其保持稳定运行。其中包含来自 MongoDB 工程师的支持，以及对 MongoDB Cloud Manager 的访问权限。

部署支持 帮助您快速起步并开始运行。提供针对项目早期阶段的全套软件和服务。

MongoDB 咨询 包可使您更快投入生产、帮助您在生产中调整性能、帮助您扩展并使您有时间专注于生产下一版本。

MongoDB 培训 帮助您成为 MongoDB 专家，范围从设计到大规模关键任务型系统运维。无论您是开发人员、DBA 还是架构师，我们都可以使您更加精通 MongoDB。

联系我们 以了解详细信息，或者访问 [mongodb.com](http://mongodb.com)。

# 资源

有关详细信息，请访问 [mongodb.com](http://mongodb.com) 或通过 [sales@mongodb.com](mailto:sales@mongodb.com) 联系我们。

案例研究 ([mongodb.com/customers](http://mongodb.com/customers))

演讲 ([mongodb.com/presentations](http://mongodb.com/presentations))

免费在线培训 ([university.mongodb.com](http://university.mongodb.com))

网络讲座与活动 ([mongodb.com/events](http://mongodb.com/events))

文档 ([docs.mongodb.org](http://docs.mongodb.org))

MongoDB Enterprise 下载 ([mongodb.com/download](http://mongodb.com/download))

