https://mp.weixin.qq.com/s/QGf9m18b2uQJRHUYesOwGw

**了解 MongoDB 看这一篇就够了**

## 基本模型

| **SQL概念** | **MongoDB概念** |
| --- | --- |
| database | database |
| table | collection |
| row | document |
| column | field |

**说明**

* database 数据库，与SQL的数据库(database)概念相同，一个数据库包含多个集合(表)
* collection 集合，相当于SQL中的表(table)，一个集合可以存放多个文档(行)。不同之处就在于集合的结构(schema)是动态的，不需要预先声明一个严格的表结构。更重要的是，默认情况下 MongoDB 并不会对写入的数据做任何schema的校验。
* document 文档，相当于SQL中的行(row)，一个文档由多个字段(列)组成，并采用bson(json)格式表示。
* field 字段，相当于SQL中的列(column)，相比普通column的差别在于field的类型可以更加灵活，比如支持嵌套的文档、数组。此外，MongoDB中字段的类型是固定的、区分大小写、并且文档中的字段也是有序的。

另外，SQL 还有一些其他的概念，对应关系如下：

| **SQL概念** | **MongoDB概念** |
| --- | --- |
| primary key | \_id |
| foreign key | reference |
| view | view |
| index | index |
| join | $lookup |
| transaction | trasaction |
| group by | aggregation |

**说明**

* id 主键，MongoDB 默认使用一个id 字段来保证文档的唯一性。
* reference 引用，勉强可以对应于 外键(foreign key) 的概念，之所以是勉强是因为 reference 并没有实现任何外键的约束，而只是由客户端(driver)自动进行关联查询、转换的一个特殊类型。
* view 视图，MongoDB 3.4 开始支持视图，和 SQL 的视图没有什么差异，视图是基于表/集合之上进行动态查询的一层对象，可以是虚拟的，也可以是物理的(物化视图)。
* index 索引，与SQL 的索引相同。
* $lookup，这是一个聚合操作符，可以用于实现类似 SQL-join 连接的功能
* transaction 事务，从 MongoDB 4.0 版本开始，提供了对于事务的支持
* aggregation 聚合，MongoDB 提供了强大的聚合计算框架，group by 是其中的一类聚合操作。

### BSON 数据类型

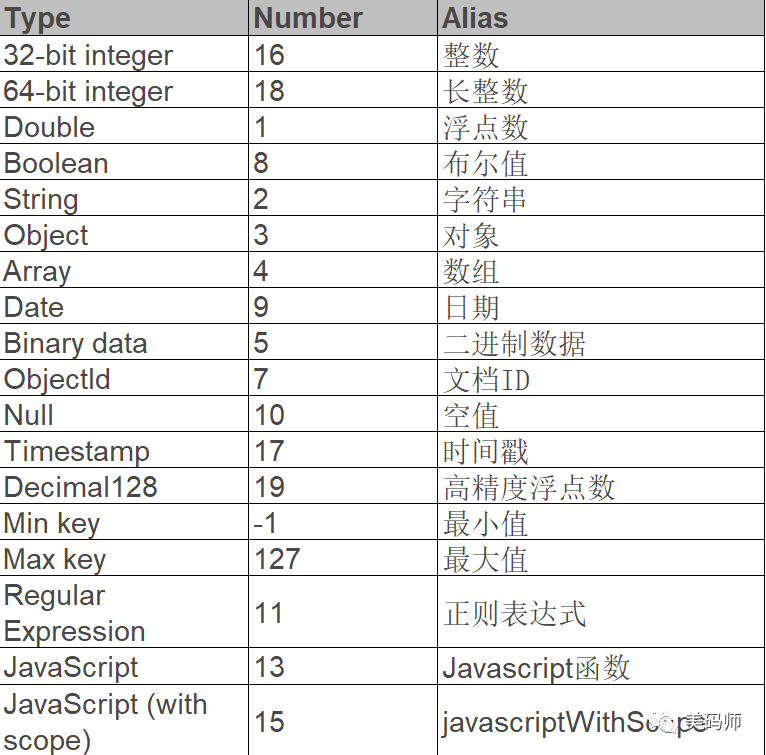
MongoDB 文档可以使用 Javascript 对象表示，从格式上讲，是基于 JSON 的。

一个典型的文档如下：



曾经，JSON 的出现及流行让 Web 2.0 的数据传输变得非常简单，所以使用 JSON 语法是非常容易让开发者接受的。但是 JSON 也有自己的短板，**比如无法支持像日期这样的特定数据类型**，因此 MongoDB 实际上使用的是一种扩展式的JSON，叫 BSON(Binary JSON)。

BSON 所支持的数据类型包括：

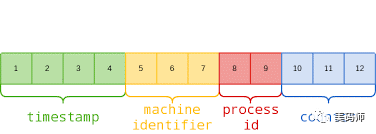


### 分布式ID

在单机时代，大多数应用可以使用数据库自增式ID 来作为主键。 传统的 RDBMS 也都支持这种方式，比如 mysql 可以通过声明 auto\_increment来实现自增的主键。 但一旦数据实现了分布式存储，这种方式就不再适用了，原因就在于无法保证多个节点上的主键不出现重复。

为了实现分布式数据ID的唯一性保证，应用开发者提出了自己的方案，而大多数方案中都会将ID分段生成，如著名的 snowflake 算法中就同时使用了时间戳、机器号、进程号以及随机数来保证唯一性。

MongoDB 采用 ObjectId 来表示主键的类型，数据库中每个文档都拥有一个\_id 字段表示主键。\_id 的生成规则如下：



其中包括：

* 4-byte Unix 时间戳
* 3-byte 机器 ID
* 2-byte 进程 ID
* 3-byte 计数器(初始化随机)

值得一提的是 id 的生成实质上是由客户端(Driver)生成的，这样可以获得更好的随机性，同时降低服务端的负载。当然服务端也会检测写入的文档是否包含id 字段，如果没有就自动生成。

## 三、操作语法

除了文档模型本身，对于数据的操作命令也是基于JSON/BSON 格式的语法。

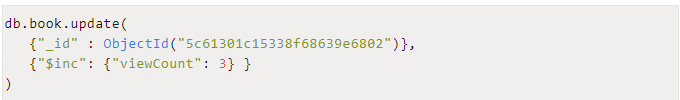
比如插入文档的操作：



执行文档查找：



更新文档的命令：



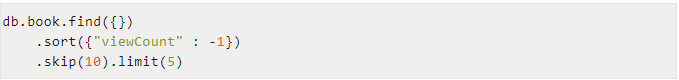
删除文档的命令：



在传统的SQL语法中，可以限定返回的字段，MongoDB可以使用Projection来表示：



实现简单的分页查询：

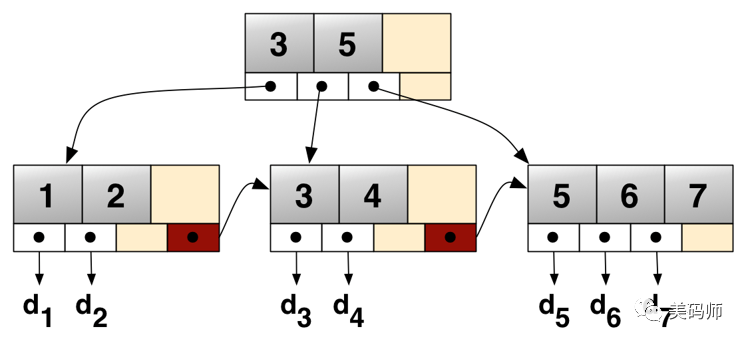


这种基于BSON/JSON 的语法格式并不复杂，它的表达能力或许要比SQL更加强大。与 MongoDB 做法类似的还有 ElasticSearch，后者是搜索数据库的佼佼者。

## 四、索引

无疑，索引是一个数据库的关键能力，MongoDB 支持非常丰富的索引类型。利用这些索引，可以实现快速的数据查找，而索引的类型和特性则是针对不同的应用场景设计的。

索引的技术实现依赖于底层的存储引擎，在当前的版本中 MongoDB 使用 wiredTiger 作为默认的引擎。在索引的实现上使用了 B+树的结构，这与其他的传统数据库并没有什么不同。所以这是个好消息，**大部分基于SQL数据库的一些索引调优技巧在 MongoDB 上仍然是可行的。**



使用 ensureIndexes 可以为集合声明一个普通的索引：

1. db.book.ensureIndex({author: 1})

author后面的数字 1 代表升序，如果是降序则是 -1

实现复合式(compound)的索引，如下：

1. db.book.ensureIndex({type: 1, published: 1})

只有对于复合式索引时，索引键的顺序才变得有意义

如果索引的字段是数组类型，该索引就自动成为数组(multikey)索引：

1. db.book.ensureIndex({tags: 1})

MongoDB 可以在复合索引上包含数组的字段，但最多只能包含一个

### 索引特性

在声明索引时，还可以通过一些参数化选项来为索引赋予一定的特性，包括：

* unique=true，表示一个唯一性索引
* expireAfterSeconds=3600，表示这是一个TTL索引，并且数据将在1小时后老化
* sparse=true，表示稀疏的索引，仅索引非空(non-null)字段的文档
* partialFilterExpression: { rating: { $gt: 5 }，条件式索引，即满足计算条件的文档才进行索引