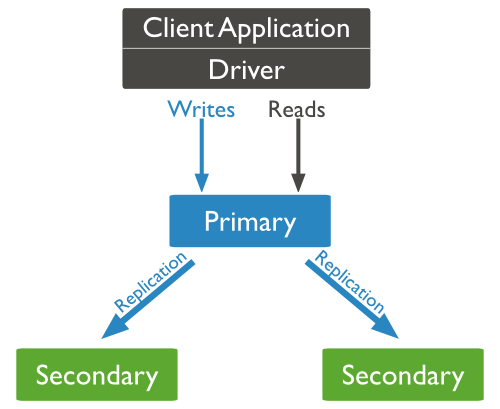
**MongoDB 复制集学习—-基础篇**

话不多说，先说说复制集的作用：复制在为数据提供了冗余同时，也提高了数据的可用性。 （摘自MongoDB 中文社区）

* **MongoDB 复制集的结构以及基本概念**



图一 MongoDB 基本结构图

正如上图所示，MongoDB 复制集的架构中，主要分为两部分：主节点（Primary）和从节点（Secondary）。

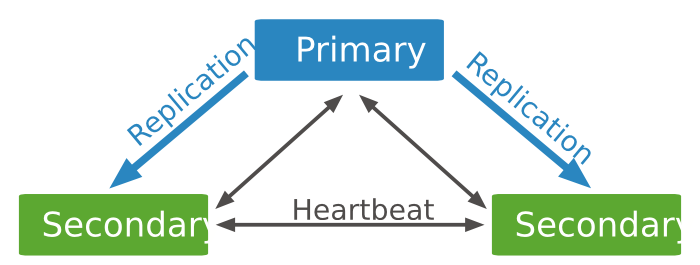
**主节点**：在一个复制集中只有并且必须有一个主节点，主节点点也是众多实例中唯一可以接收客户端写操作的节点，当然也可以进行读操作；

**从节点**：从节点会复制主节点的操作，以获取完全一致的数据集。客户端不能够直接对从节点进行写操作，但是可以进行读操作，这个需要通过复制集选项进行设置。

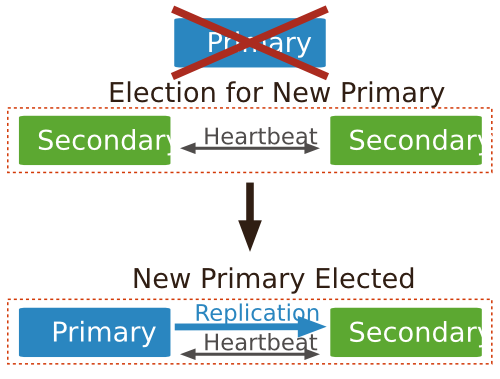
注：MongoDB 3.0 把复制集中的成员数量从原来的12个提升到了50个，但是投票节点的数量仍然保持不变，还是7个。

**投票节点**：投票节点 并不含有 复制集中的数据集副本，且也 无法 升职为主节点。投票节点的存在是为了使复制集中的节点数量为奇数，这样保证在进行投票的时候不会出现票数相同的情况。如果添加了一个节点后，总节点数为偶数，那么就需要相应的增加一个投票节点。

* **最基本的复制集架构**

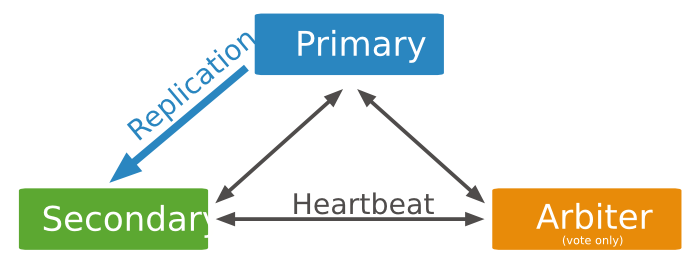
 **一个主节点，两个从节点**

   最基本的复制集架构是有3个节点的形式。这样在主节点不可用以后，从节点会进行投票选出一个节点成为主节点，继续工作。如下图所示：



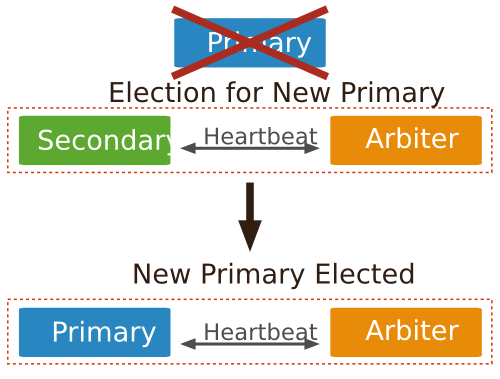
**重新投票选出主节点**

  三个节点的复制集架构，还有另外一种形式：一个主节点，一个从节点，一个投票节点。如下图所示：



**一个主节点，一个从节点，一个投票节点**

  在这种架构中，当主节点不可用时，只有从节点可以升为主节点，而投票节点是不可以成为主节点的。投票节点仅仅在选举中进行投票。如下图所示：

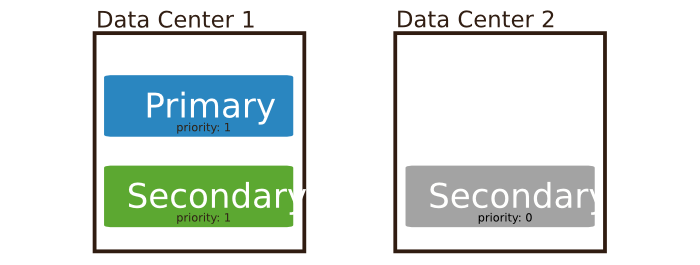


**从节点无法升职为主节点**

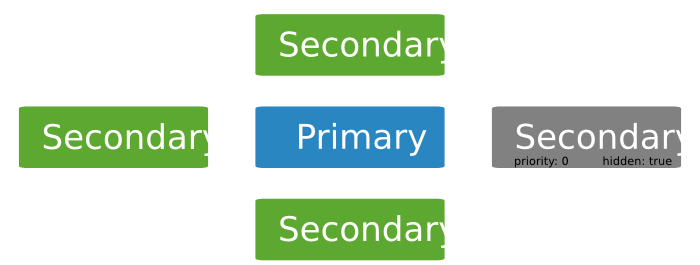
* **其他概念**

        从节点还有集中特殊的设置情况，不同的设置有不同的需求：

**优先级为0**：设置 priority:0 ，那么该结点将不能成为主节点，但是其数据仍是与主节点保持一致的,而且应用程序也可以进行读操作。这样可以在某些特殊的情况下，保证其他特定节点优先成为主节点。

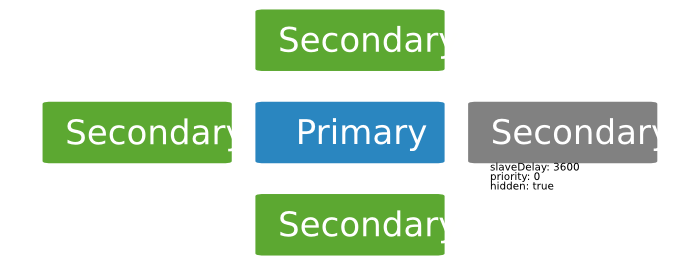


**隐藏节点**：隐藏节点与主节点的数据集一致，但是对于应用程序来说是不可见的。隐藏节点可以很好的与 复制集 中的其他节点隔离，并应对特殊的需求，比如进行报表或者数据备份。隐藏节点也应该是一个 不能升职为主节点 的 优先级为0的节点。



**延时节点**：延时节点也将从 复制集 中主节点复制数据，然而延时节点中的数据集将会比复制集中主节点的数据延后。举个例子，现在是09：52，如果延时节点延后了1小时，那么延时节点的数据集中将不会有08：52之后的操作**。**

由于延时节点的数据集是延时的，因此它可以帮助我们在人为误操作或是其他意外情况下恢复数据。举个例子，当应用升级失败，或是误操作删除了表和数据库时，我们可以通过延时节点进行数据恢复。



**oplog**：全拼 oprations log，它保存有数据库的所有的操作的记录。在复制集中，主节点产生 oplog，然后从节点复制主节点的 oplog 进行相应的操作，这样达到保持数据集一致的要求。因此从节点的数据与主节点的数据相比是有延迟的。