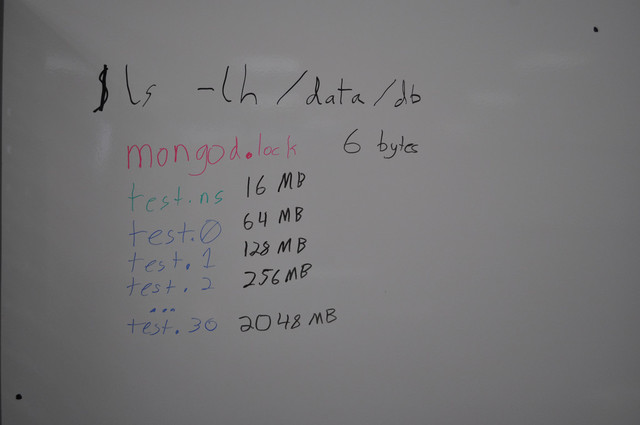
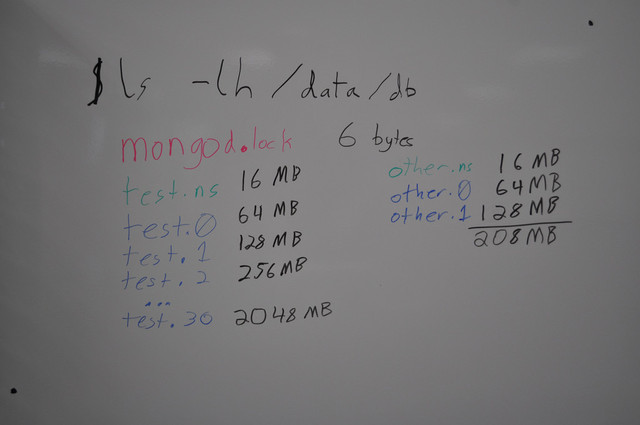
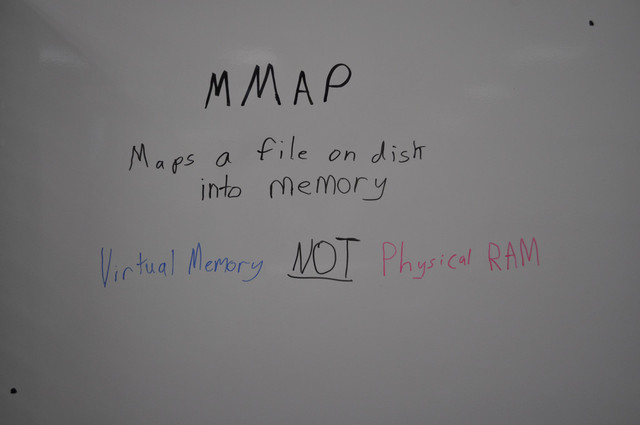
**MongoDB内部结构**

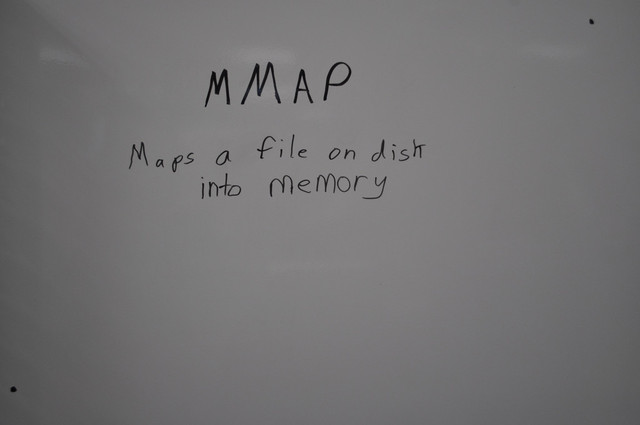
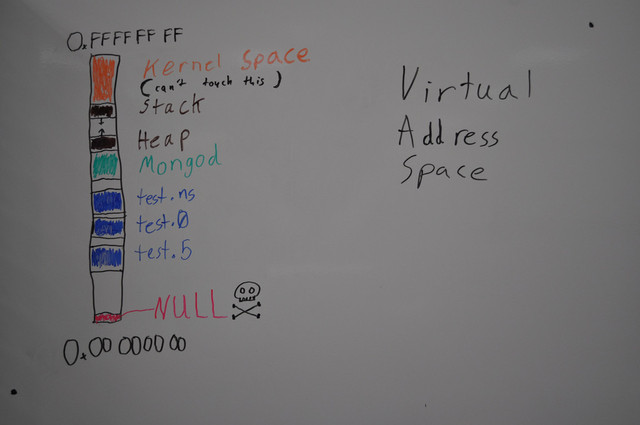
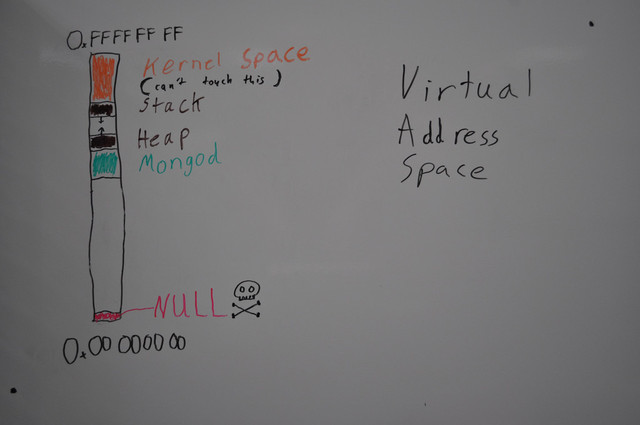
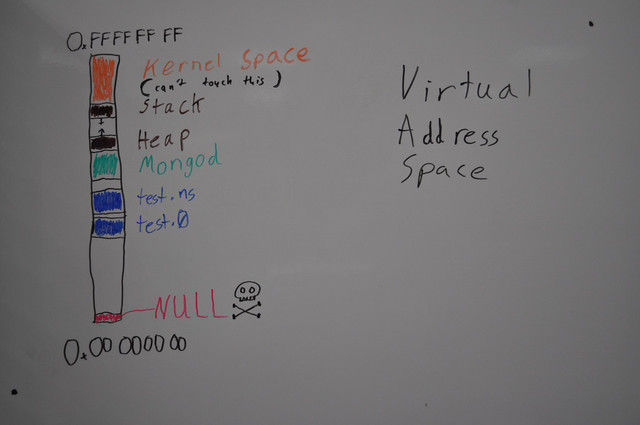
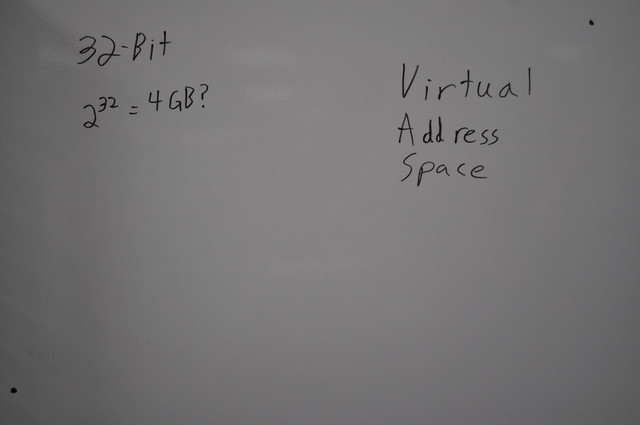
每一个数据库都有自己独立的文件。如果你开启了【directoryperdb】选项，那你每个库的文件会单独放在一个文件夹里。

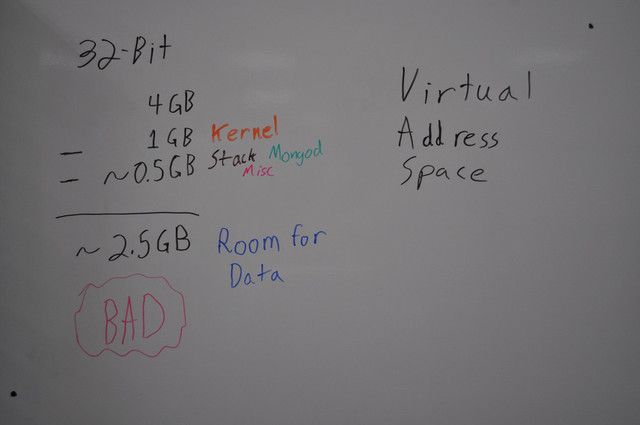
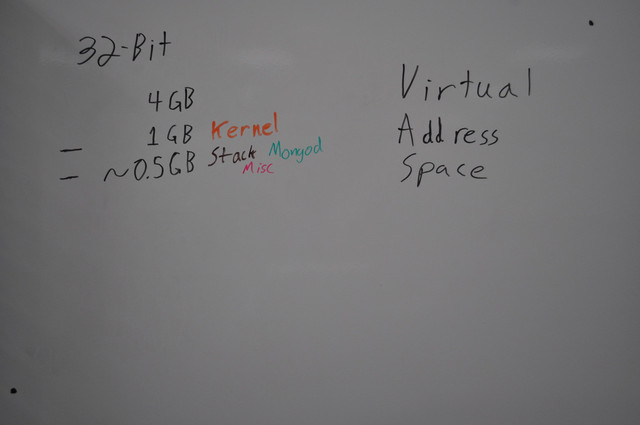
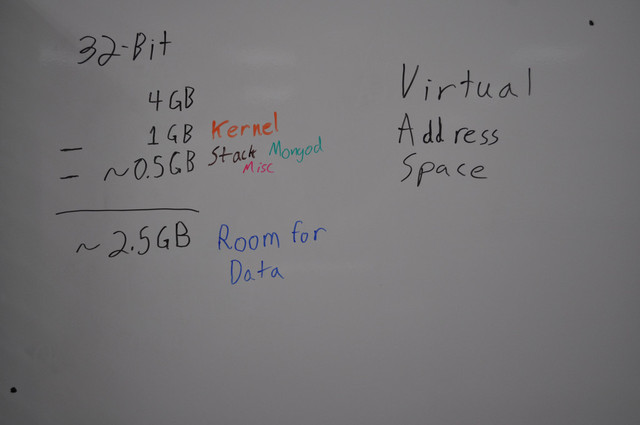
数据库文件在内部会被切分成单个的块，每个块只保存一个名字空间的数据。在MongoDB中，名字空间用于区分不同的存储类别。比如每个collection有一个独立的名字空间，每个索引也有自己的名字空间。

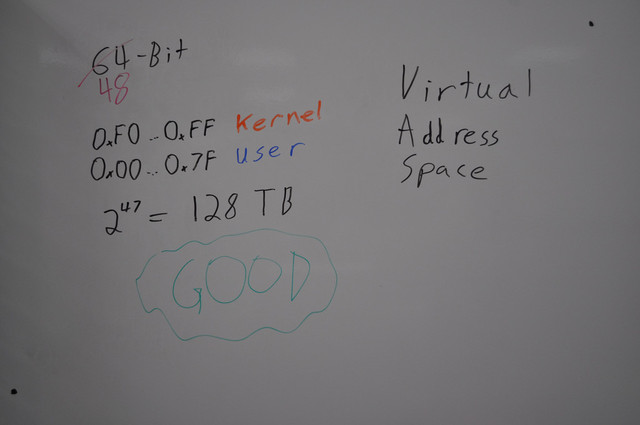
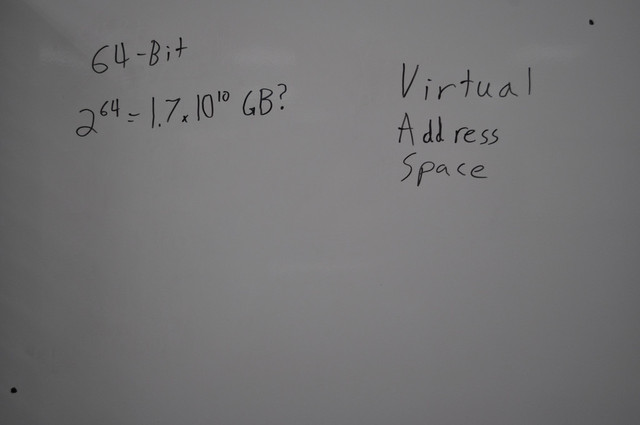
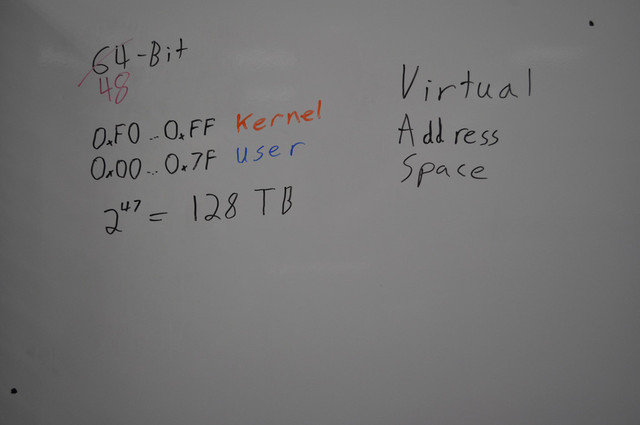
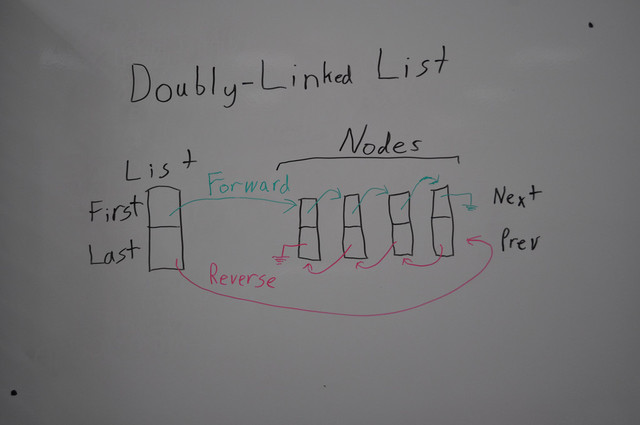
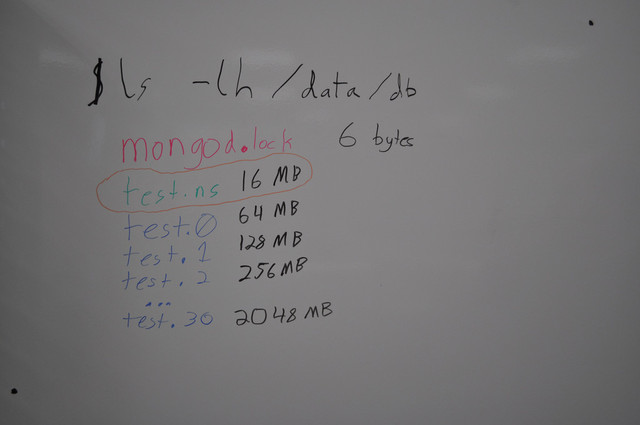
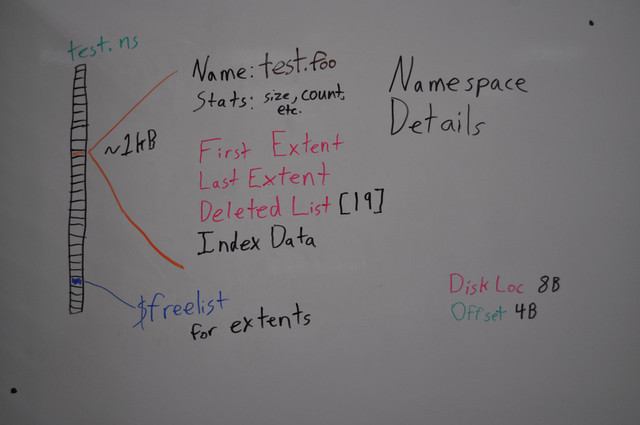
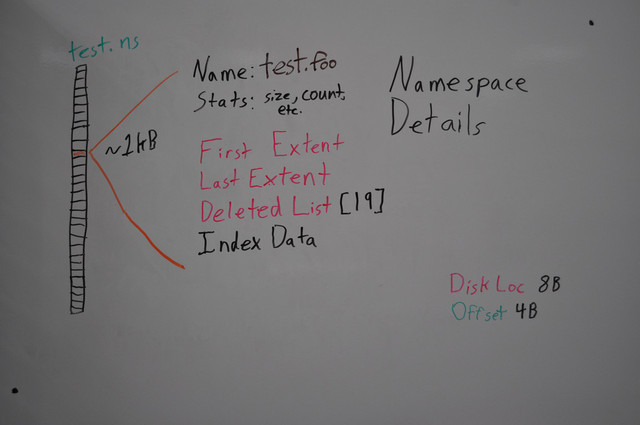
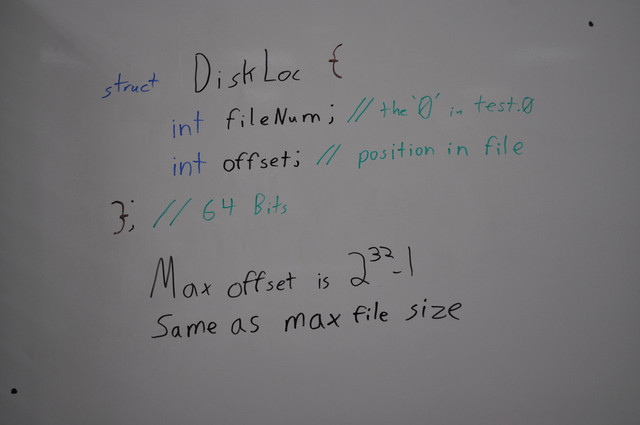
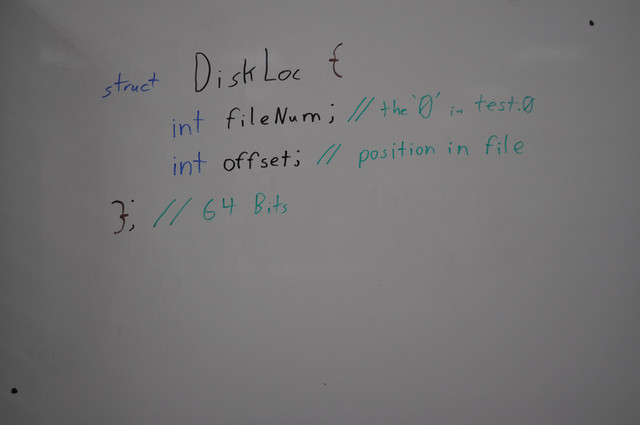
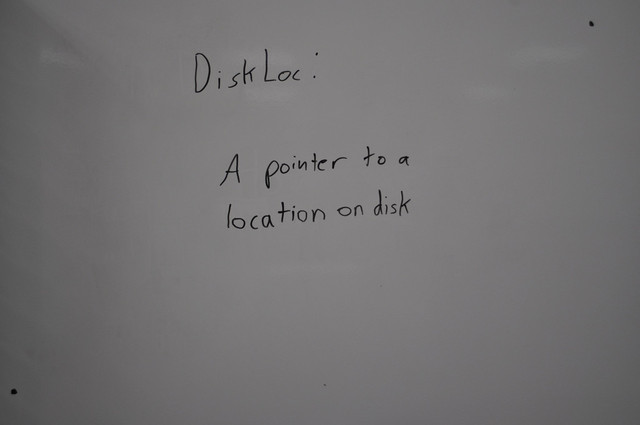
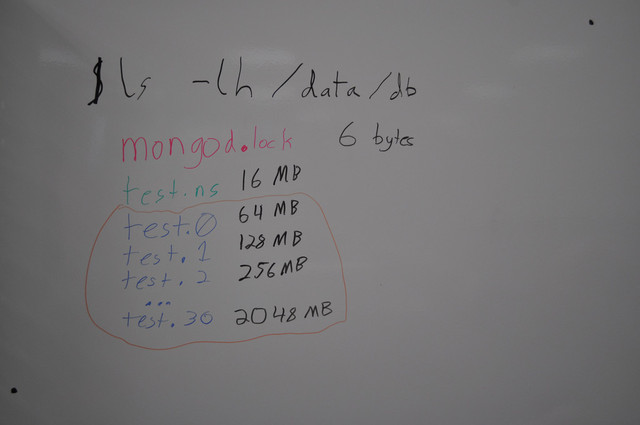
在一个块中，会保存多条记录，每条记录是BSON格式的，记录与记录之间通过**双向链表**进行连接。

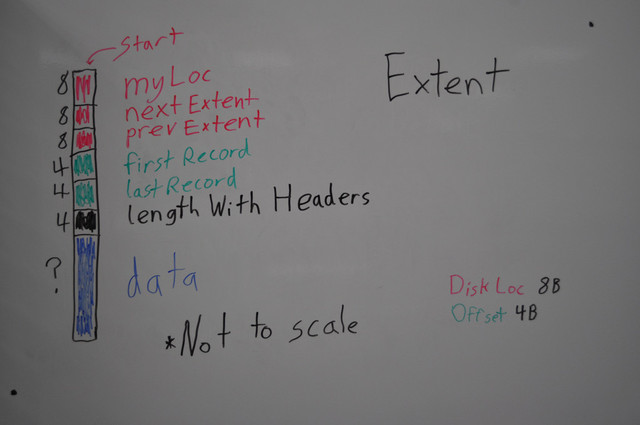
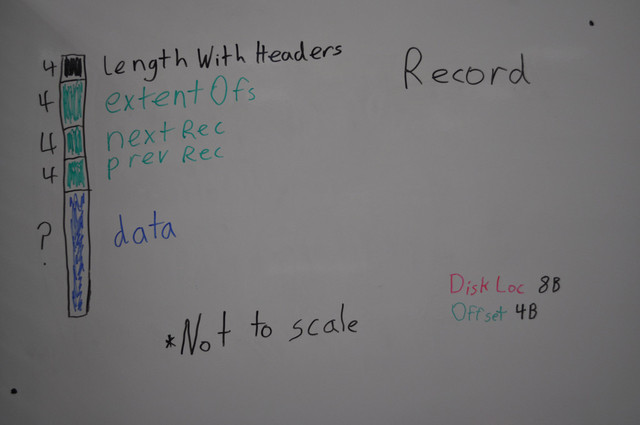
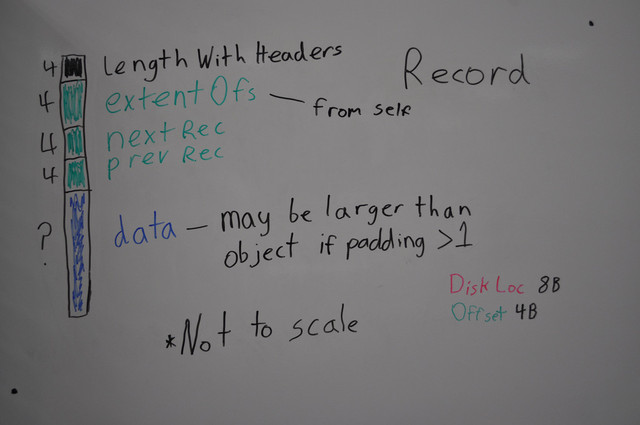
索引数据也存在数据文件中，不过索引是被组织成B Tree结构，而不是双向链表。

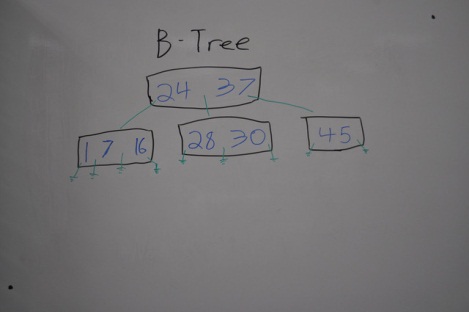
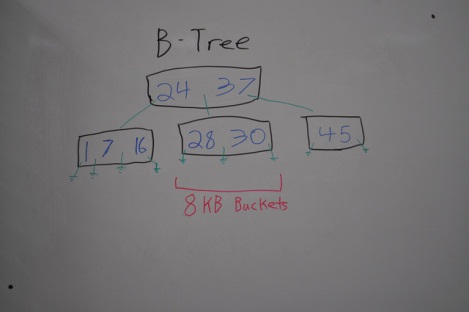
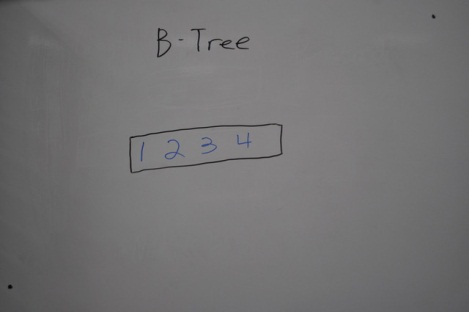
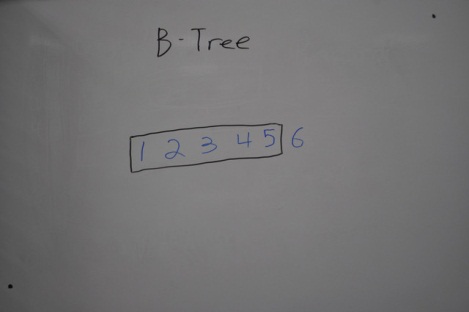
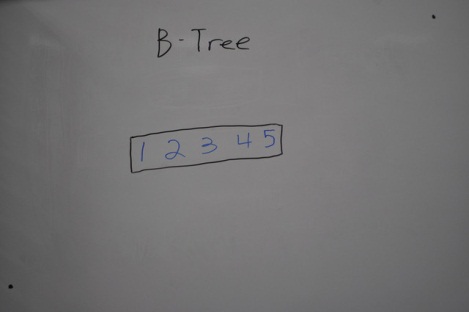
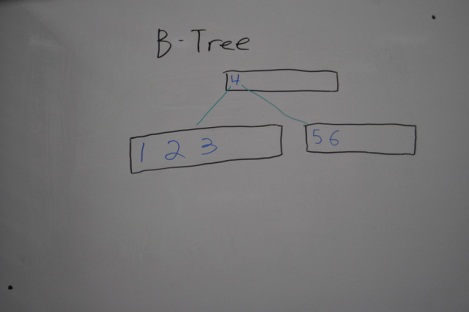
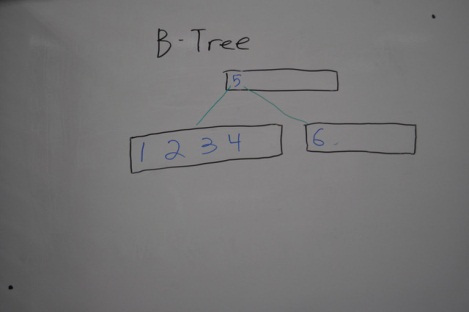
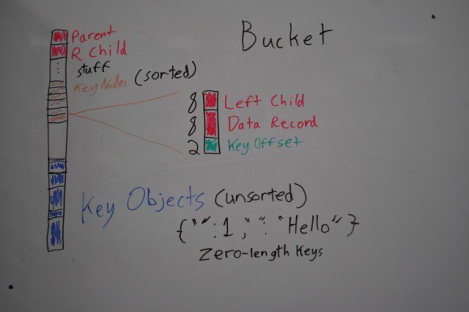
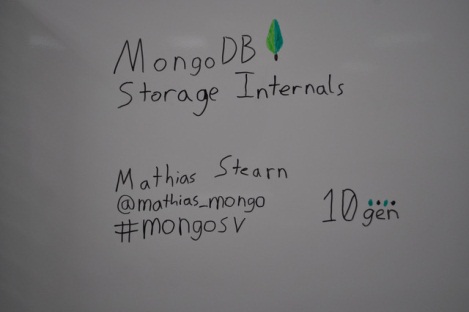
对每个数据库，有一个命名空间文件，用于保存每个名字空间对应的元数据。我们通过查询这些元数据来找到对应的名字空间的存储块位置。  
  
如果你开启了jorunaling日志，那么还会有一些文件存储着你所有的操作记录。  
  
下面图片摘自10gen工程师Mathias Stearn在MongoSV2011大会上的发言稿，手绘的数据文件结构。  
  
1.每个数据库有相应的数据文件和命名空间文件  
  
2.数据文件从16MB开始，新的数据文件比上一个文件大一倍，最大为2GB  
3.文件使用MMAP进行内存映射，会将所有数据文件映射到内存中，但是只是虚拟内存，只有访问到这块数据时才会交换到物理内存。  


4.MongoDB的数据文件映射到内存表中的位置  
5.使用32位机器的话，内存地址最大可以标识4GB内存  
  
6.但是在32位机器上，4GB内存会有1GB被内核战用，大约0.5GB会用于mongod进程的stack空间，只剩下大约2.5GB可用于映射数据文件。



7.在64位机器上则最多可以表示128TB的空间（48位寻址空间）  
  
8.每个数据文件会被分成一个一个的数据块，块与块之间用双向链表连接  
  
  
9.在名字空间文件中，保存的是一个hash table，保存了每个名字空间的存储信息元数据，包括其大小，块数，第一块位置，最后一块位置，被删除的块的链表以及索引信息  
  
10.这些位置通过DiskLoc数据结构进行存储，存储了数据文件编号和块在文件中的位置  


11.对每一个块来说，其头部包含了一些块的元数据，比如自己的位置，上一个和下一个块的位置以及块中第一条和最后一条记录的位置指针。剩下的部分用于存储具体的数据，具体数据之间也是通过双向链接来进行连接。  


12.下面是B Tree的存储结构和工作原理  


**数据文件**

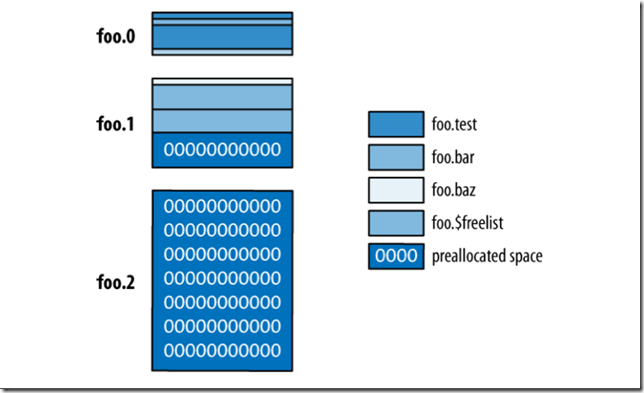
在MongoDB的数据文件夹中（默认路径是/data/db）由构成数据库的所有文件。每一个数据库都包含一个.ns文件和一些数据文件，其中数 据文件会随着数据量的增加而变多。所以如果有一个数据库名字叫做foo，那么构成foo这个数据库的文件就会由 foo.ns，foo.0，foo.1，foo.2等等组成。

数据文件每新增一次，大小都会是上一个数据文件的2倍，每个数据文件最大2G。这样的设计有利于防止数据量较小的数据库浪费过多的空间，同时又能保证数据量较大的数据库有相应的空间使用。

MongoDB会使用预分配方式来保证写入性能的稳定（这种方式可以使用–noprealloc关闭）。预分配在后台进行，并且每个预分配的文件都 用0进行填充。这会让MongoDB始终保持额外的空间和空余的数据文件，从而避免了数据增长过快而带来的分配磁盘空间引起的阻塞。

## 名字空间和盘区

每一个数据库都由多个名字空间组成，每一个名字空间存储了相应类型的数据。数据库中的每一个Collection都有各自对应的名字空间，索引文件同样也有名字空间。所有名字空间的元数据都存储在.ns文件中。

名字空间中的数据在磁盘中分为多个区间，这个叫做盘区。在下图中，foo这个数据库包含3个数据文件，第三个数据文件属于空的预分配文件。头两个数据文件被分为了相应的盘区对应不同的名字空间。

上图显示了名字空间和盘区的相关特点。每一个名字空间可以包含多个不同的盘区，这些盘区并不是连续的。与数据文件的增长相同，每一个名字空间对应的盘区大小 的也是随着分配的次数不断增长的。这样做的目的是为了平衡名字空间浪费的空间与保持某一个名字空间中数据的连续性。上图中还有一个需要注意的名字空 间：$freelist，这个名字空间用于记录不再使用的盘区（被删除的Collection或索引）。每当名字空间需要分配新的盘区的时候，都会先查 看$freelist是否有大小合适的盘区可以使用。

## 内存映射存储引擎

MongoDB目前支持的存储引擎为内存映射引擎。当MongoDB启动的时候，会将所有的数据文件映射到内存中，然后操作系统会托管所有的磁盘操作。这种存储引擎有以下几种特点：

\* MongoDB中关于内存管理的代码非常精简，毕竟相关的工作已经有操作系统进行托管。  
\* MongoDB服务器使用的虚拟内存将非常巨大，并将超过整个数据文件的大小。不用担心，操作系统会去处理这一切。  
\* MongoDB无法控制数据写入磁盘的顺序，这样将导致MongoDB无法实现writeahead日志的特性。所以，如果MongoDB希望提供一种durability的特性（这一特性可以参考我写的关于Cassandra文章：[http://www.cnblogs.com/gpcuster/tag/Cassandra/](http://www.cnblogs.com/gpcuster/tag/Cassandra/" \o "http://www.cnblogs.com/gpcuster/tag/Cassandra/)），需要实现另外一种存储引擎。  
\* 32位系统的MongoDB服务器每一个Mongod实例只能使用2G的数据文件。这是由于地址指针只能支持32位。