1. 存储结构及查询
2. 线性存储

存储：链表、数组

查询：全量查询

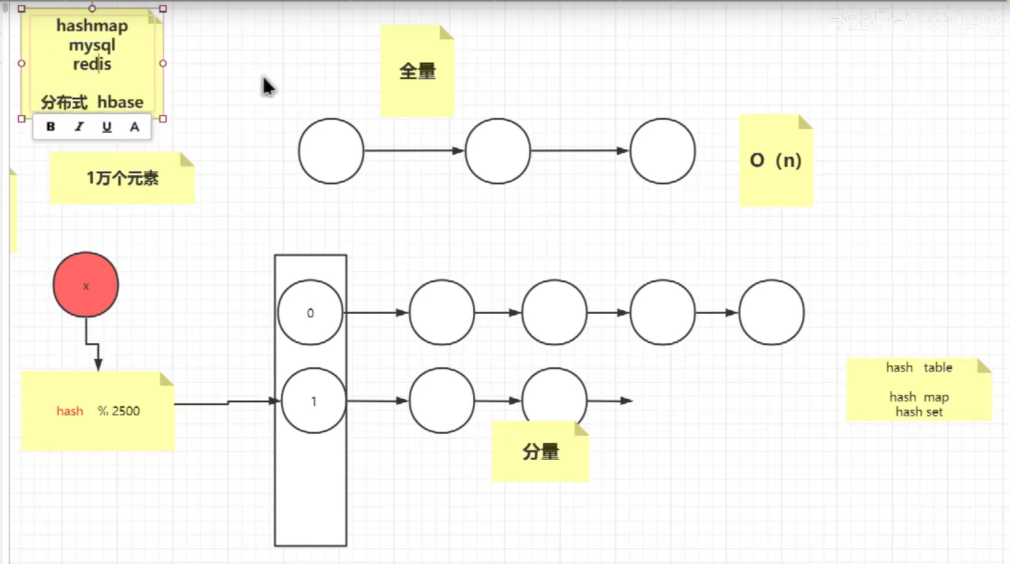
复杂度：o(n)

1. 分治+算法（hash）

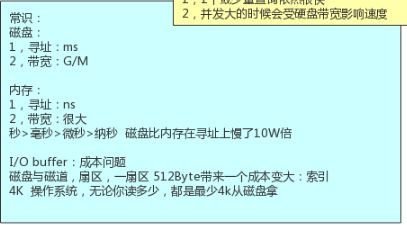
存储：hashTable（数组+链表）、hashMap（数组+红黑树）

查询：分量查询

复杂度：o(n)，因为最坏情况下存储结构变成链表



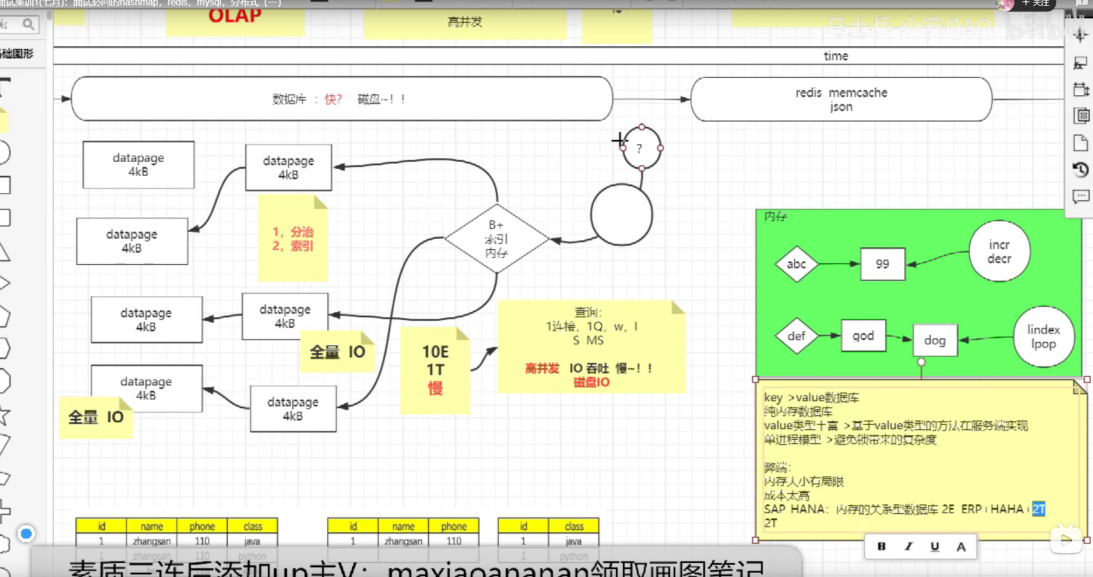
1. 常识

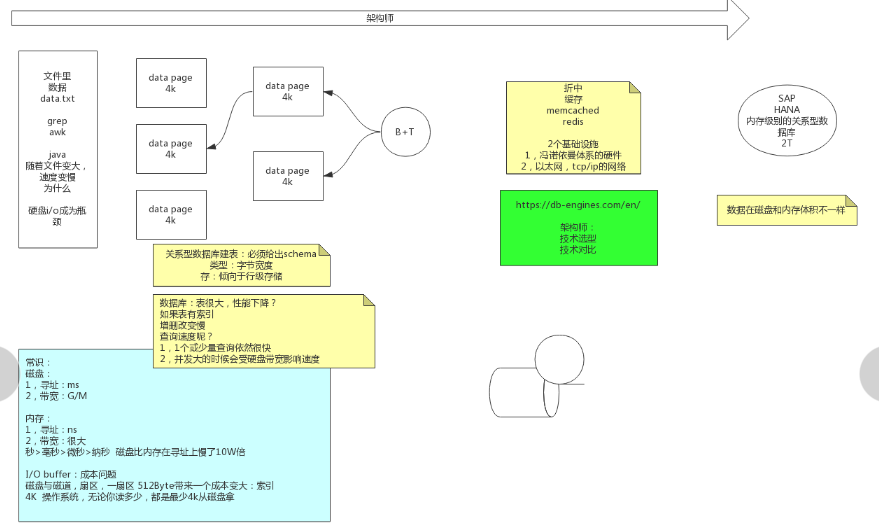


读取单位：真正读取磁盘，不是按照扇区大小512字节为单位（因为索引成本大），而是按照4k为单位。

无论你读取多少，最小都是4k。

硬盘慢：寻址慢、带宽小





1. 存储演变之文件存储
2. 存储格式

线性存储（以换行符作为分隔，每条数据占一行）。

1. 查询

全量查询（grep、awk、java程序、c++程序）。

1. 问题：随着文件变大，速度变慢？

全量I/O，磁盘I/O成为瓶颈

1. 存储演变之磁盘关系型数据库
2. 存储格式

* 数据

磁盘中。分治存储，以datapage为单位，大小为4K，符合I/O读取。

* 一级索引（B+树叶子）

磁盘中。通过索引定位到数据的datapage。

* 二级索引（B+树的树干）

内存中，因为索引的datapage可能也会很大，受限于磁盘I/O，所以就有了二级索引，二级索引主要存储的是区间，并存储在内存中。定位到索引的datapage，因为在内存速度肯定比磁盘快。

1. 查询

查询数据->内存B+树（ns级）->索引的datapage（ms级）->数据的datapage（ms级）

1. 问题：

* 为什么数据库比文件快？

磁盘中存储数据、索引，内存中存储一些索引区间，充分利用了磁盘存储空间，内存速度快，又有B+树加快遍历查找速度。

最终是为了减少磁盘IO、磁盘寻址。

减少IO：减少IO量、减少IO次数

* 没命中索引：退化为全量IO。
* 数据库表很大，性能一定下降吗？

增删改：如果有索引，由于需要维护索引，使得增删改变慢。

查询：

1. 一个或者少个查询，且命中索引，依然很快
2. 并发大，可能获取很多个datapage，因此受到硬盘带宽影响，使得速度变慢
3. 存储演变之NoSql缓存数据库-Redis、Memcache

折中方案：磁盘数据库慢，内存数据库贵，把热数据存储到内存中

1. 存储格式

Key-Value（Value有类型，且有本地方法）

热数据：存储在内存

s

1. 查询

Key-Value

1. 问题

* 为什么Redis是NoSql，有Sql版本吗？

首先，关系型数据库的数据存储要么是展开扁平化（虽然简单，都是可能突破硬件瓶颈，存储空间不足），要么是根据范式分割

1. 存储演变之内存关系型数据库



问题：数据在内存和磁盘的大小一样吗？

不一样。数据在内存的大小比在磁盘小，比如：1w对象有个String属性值都为”abc”,则在内存中只有一份String类型的”abc”，而如果序列化到磁盘中，就会有1W份”abc”。