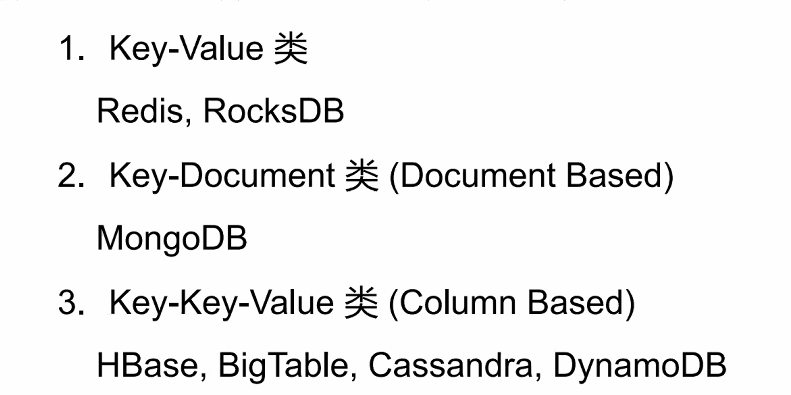
nosql：硬盘上的哈希表，常见的有mongoDB，Redis，HBase，RocksDB。

HBase访问时不需要用户名和密码，所以一般情况下都配置成某类或某些机器内部网络访问，不对外提供。



MongoDB中 的Document其实是一个json格式的哈希表。

Key-Key-Value其实就是哈希表套哈希表，要想访问数据，首先要拿到第一个Key，row key，再拿到第二个Key，column key，才能最终获取到数据。

nosql数据库存储数据的特点(不一定全部满足，满足越多越适合nosql)：

1. 数据结构简单，复杂的数据不太适合放到nosql中
2. 数据量极大
3. 读少写多

mysql不合适的地方：

1. 无法自动完成数据的分布式
2. 写效率不够高

Redis更像是一个缓存数据库。

存储NewsFeed选HBase而不是Redis，因为，Redis不支持查询过去某段时间内属于某个user的newsfeeds，Redis本身不支持范围查询。

nosql最初就是为了解决日志的大量日志，尽可能提高写的效率，而且要支持分布式。

对于读少写多的情况，读的时候稍微慢点无所谓，写要尽可能快。

nosql，key-value的数据库，相比mysql，结构简单，qps比mysql至少是10倍大小。

选择HBase：

1. 高并发
2. 最主要就是天然的分布式功能，带来了安全，可靠的提升。
3. 写优先，写多读少。

HBase是一个在硬盘上的大哈希表，外层是row key，它对应的value又是一个哈希表，其对应的key是column key。如果去对应java的数据结构，就是TreeMap<string, HashMap<string, string>>，对应C++就是map<string, unordered\_map<string, string>>。其中row，column和mysql中的行，列不太一样，它是一级维度，二级维度的意思。

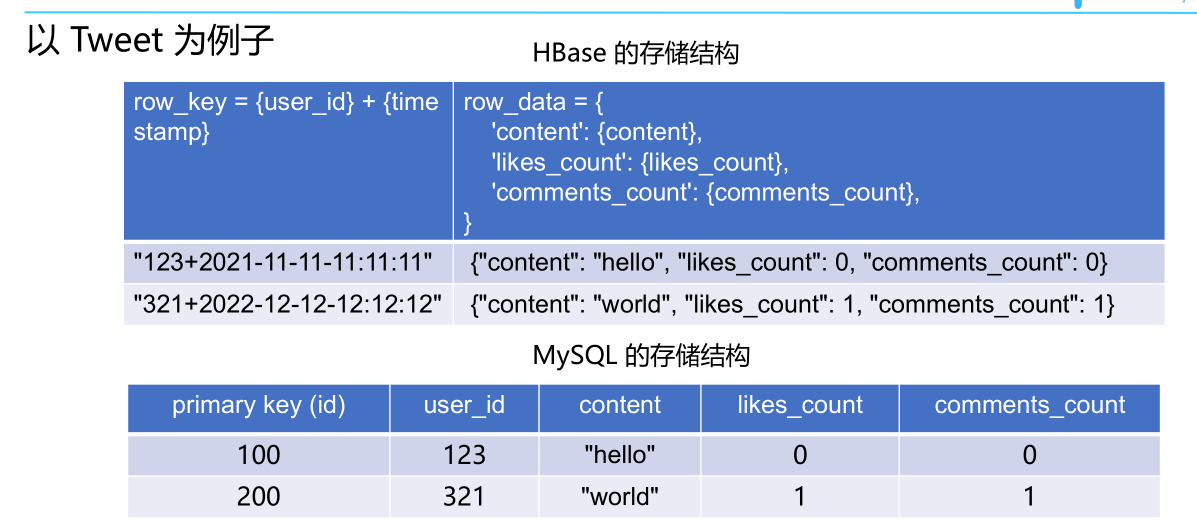
Cassandra和HBase正好反过来，它类似HashMap<string, TreeMap<string, string>>，它是局部有序的。

HBase是基于row key的全局排序，这意味着不同机器上存储的key也是有序的。column key是在每个row 可以下面，它是无序的。

对row key可以进行range query，范围查询，例如指定row key在’abc’到’def’之间的所有数据。

MySQL支持范围查询，索引的结构是B+树。之所以使用B+树，而不是平衡二叉查找树，是因为B+树是多路的，这样树的高度就比较矮，在磁盘上进行查找时，可以降低磁头寻道的次数，这样节省时间。

HBase和MySQL的存储对比：



HBase中的Row key对应的是mysql中的Primary key。

HBase是java写的，所以需要jdk。

HBase中Row Key的设计，如果没有设计好，会导致数据扎堆，大量的key都存储在一小部分机器上，导致机器之间的数据的存储分布不均匀，从而导致热点问题(Hotspot)

项目中采用user\_id:timestamp来作为row key，对user\_id进行了反转和补0的操作，这不会影响到后续的范围查询，因为friendship和newsfeed进行范围查询是user\_id等于某个值，然后timestamp在某个范围内，而我们并没有对timestamp进行其他的操作，所以它仍然可以进行范围查询。

如果直接查询某个时间范围内的数据，则row key就不能像上面那样设计，而是要把timestamp直接放到row key的最前面，这是就要另创建一个表单。

HBase中Row key的排序规则是字符串的字典排序。

加盐的row key能够一定程度的缓解热点问题。

加盐的Row key的问题：如果需要进行范围查询，需要枚举所有可能的盐作为前缀分别查询，增加了查询的时间。所以它一般用于不需要进行范围查询的情况。

HBase ORM的设计：直接调用happybase的接口会比较麻烦，包括happybase接口自身的参数在程序里直接传递比较麻烦，另外HBase的row key的设计也需要技巧，这样就需要对进行一层封装，构建一个类似Django ORM的HBase ORM，这样调用hbase时能更加方便，另外希望包装的接口更类似Django ORM，方便代码风格统一。

异步任务的重试：第三方的API都不可信，需要重试。

column\_family：可以把一部分column key进行组合，通常是把需要一起访问的column key放到一个column family中，这样hbase会把他们存在一起，提高读取速度。