BigTable

> Chang F, Dean J, Ghemawat S, et al. Bigtable:A Distributed Storage System for Structured Data[J]. Acm Transactions on Computer Systems, 2008, 26(2):1-26.

BigTable是谷歌开发的分布式存储系统，可以用来管理半结构化数据。具有较强的扩展性，可以存储PB级的数据。谷歌网页的索引、Google Earth以及Google金融的数据都使用到了Big Table技术。

从功能上看，BigTable很像一个数据库，实现了很多数据库的策略，例如增删查改等。但BigTable提供了另一种组织数据的思路：

数据通过行和列名进行索引，这些名字可以是任意的字符串；

所有数据的类型都是无解释的字符串；

支持从内存或者硬盘中读取数据；

Row Key	Time Stamp	Column Contents	Column Anchor		Column "mime"
			cnnsi.com	my.look.ca	
"com.cnn.www"	T9		CNN		
	T8			CNN.COM	
	T6	"<html>.. "			Text/html
	T5	"<html>.. "			
	t3	"<html>.. "			

BigTable的具体数据组织方式如上图。

BigTable采用行键（row key）、列键（column key）和时间戳（timestamp）对图进行索引：

- 一个表中的行键，是任意的字符串。对每个行键下所包含的数据的读或写都是一个原子操作，（类似于行级别锁），以维护BigTable内数据的一致性。由于BigTable的存储是分布式的，行键还是数据划分的依据。
- 列键，也可以称为列簇，是一组相关属性的集合，通常都属于同一个数据类型。列键采用下面的语法命名：family:qualifier。例如上表中的cnnsi.com，my.look.ca都属于同一个列簇，代表他们都引用了"www.cnn.com"这个网页。在BigTable中，访问控制以及磁盘和内存审计是在列家族层面上进行的。
- 时间戳用来区分相同数据的不同版本。BitTable时间戳是64位整数。BigTable对时间戳进行分配，时间戳代表了真实时间，以微秒来计算。客户应用也可以直接分配时间戳。需要避免冲突的应用必须生成唯一的时间戳。一个单元格的不同版本是根据时间戳降序的顺序进行存储的，这样，最新的版本可以被最先读取。

MongoDB

MongoDB是一个基于文档的数据库系统，有着灵活的存储模式。这种模式在MongoDB中被称为集合（Collections）。每一个集合都是一个数据库，类似于RDBMS中的表的概念；每个集合都可以有多个文档(Documents)，类似于RDBMS的一行。然而不同于RDBMS的是集合的模式并非预先定义好或者说是一种松散的模式。

MongoDB使用二进制json（Binary JSON，BSON）存储数据。BSON格式的数据(文档)存储有尺寸限制，最大为16M。因此对于较大的数据，MongoDB使用GridFS来辅助管理。GridFS使用两个集合（collection）存储文件：files集合用于存储文件的元数据；chunks集合用于存储文件内容的二进制数据。

BSON格式

BSON这种格式是专门为MongoDB而开发的，类似json的一种二进制格式。这种格式不一定比json存储的文件小，其优点是解释快。另外BSON增加了一些数据类型，如Date.

一个具体的BSON示例如下，前面的四个字节（"\x16\x00\x00\x00"） 即文档的长度，这里是小端表示，即文档的长度是22个字节。第5个字节（\x02）代表元素的类型，即“world”的类型。接下来的字节分表表示元素的名字、元素类型的长度以及元素的内容。最后一个字节（\x00）表示文档的结尾。

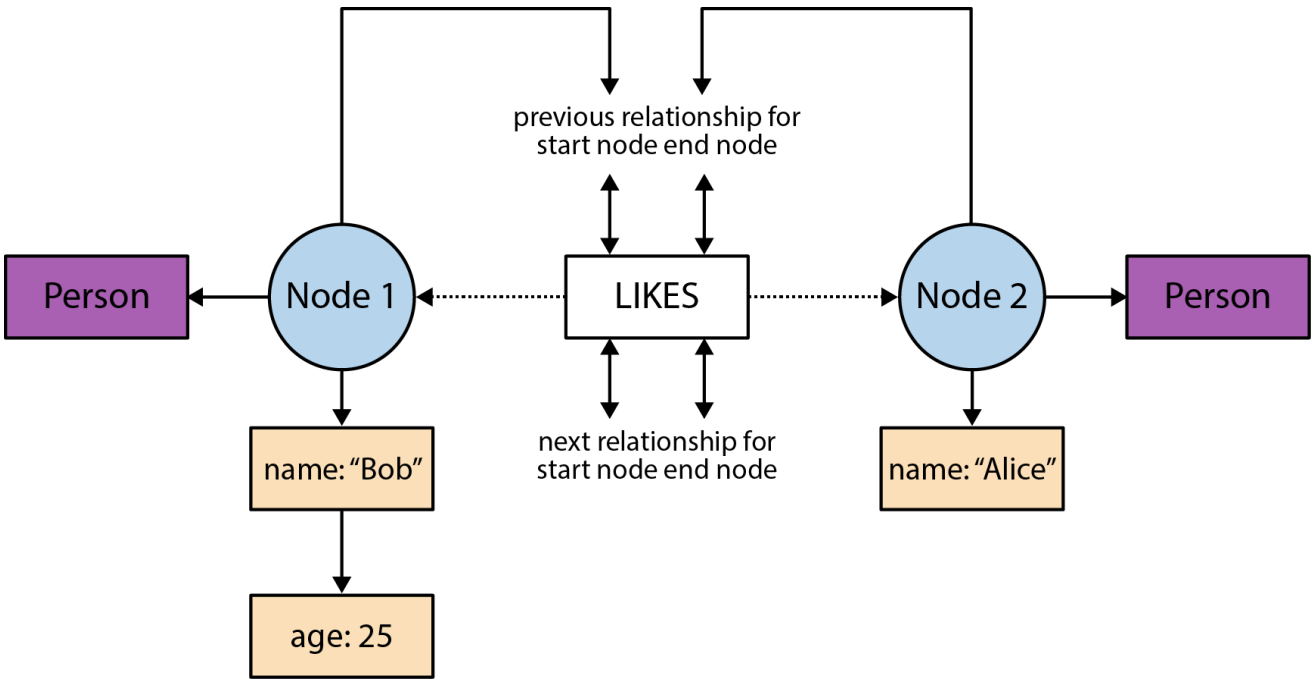
```
{"hello": "world"}          →   \x16\x00\x00\x00
                                \x02
                                hello\x00
                                \x06\x00\x00\x00world\x00
                                \x00
```

MongoDB 内部有预分配空间的机制，每个预分配的文件都用0 进行填充，由于有了这个机制,MongoDB 始终保持额外的空间和空余的数据文件。

Neo4j

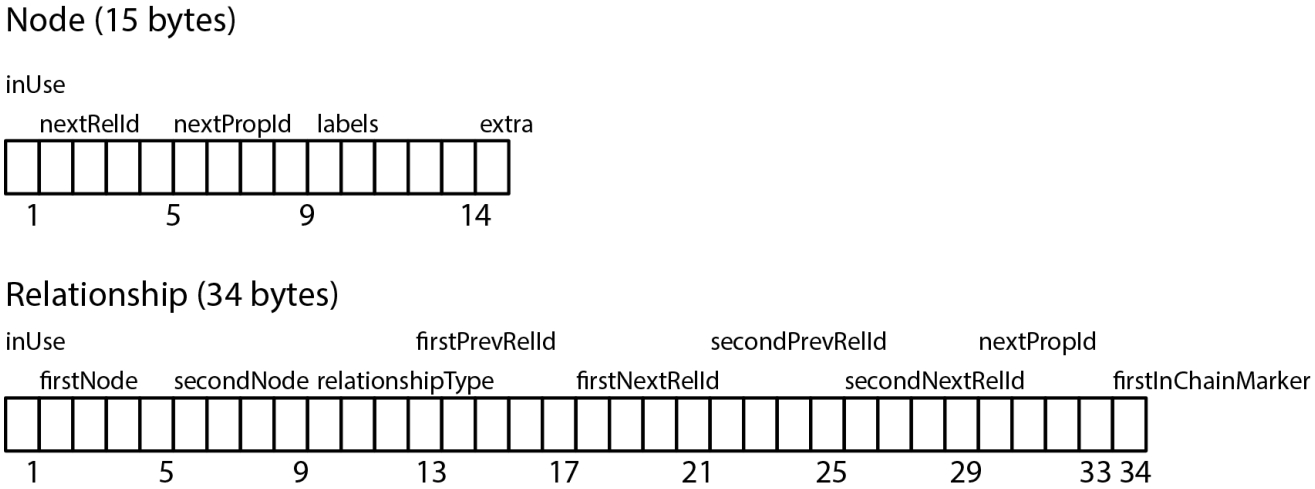
> Graph Databases

Neo4j是一个高性能的NOSQL图形数据库，它将结构化数据存储在网络上而不是表中。下图描绘了Neo4j的存储模型。图中的顶点（Node)包含自身所拥有的属性结构（Property）和关系结构（Relationship ），这些结构都以链表的形式存储，Node分别保存着这些链表的头部指针。属性链表是一条单链表，为了读取节点的属性，我们从指向属性链表的头部指针出发依次向后读取；而关系链表是一条双向循环链表，同样遍历关系链表也能找到顶点所有的关系。



在Neo4j中,主要有4类节点，属性，关系等文件是以数组作为核心存储结构，每条记录都是定长的；同时对节点，属性，关系等类型的每个数据项都会分配一个唯一的ID，在存储时以该ID 为数组的下标。这样，在访问时通过其ID作为下标，实现快速定位。所以在图遍历等操作时，可以实现 free-index。这些类的具体结构可以参看下图。inUse是标志位，标志该条记录是否正在使用（是否可以进行回收）。nextRelID和nextPropId对应着前面提到的链

表结构。Label是节点的标签信息；Extra是标志位。例如用来识别密集连接顶点（densely connected nodes）。同样的，在关系结构中，同样有inUse的标志位。跟随其后的是和关系有关联的两个定点的ID。其次还保存着关系类型的指针、两个端点在关系链表中的上一个/下一个关系的指针，以及当前关系是否位于关系链（relation chain）头部的一个标志位。



行列存储

在传统的关系型数据库中，数据是在逻辑上组织成表的形式。如表4-1所示，表的每一行代表一个元组，按照关系型数据库的这种天然属性把数据按行来组织在一起。行存储模式下同一个元组的不同属性在存储上是连续的。如果要存储一个元组的某个属性值，首先读取元组的所有数据，然后进行投影操作，从整个元组中分离出需要的属性值。

	o_orderkey	o_custkey	o_orderstatus	o_totalprice
Tuple1	...			
Tuple2				
Tuple3				
Tuple4				

表4-1 行存储数据排列

列存储模式下数据按照属性列来进行组织，如表4-2所示。表的不同元组的同种属性在存储上是连续的，不同的列独立存储。在Greenplum中，每一个属性列在操作系统中都是以单个文件的形式存放，根据实际需要读取对应的属性列。Sybase在2004年左右推出基于列存储的Sybase IQ数据库系统。目前的列式数据库包括列式数据库的代表包括：Sybase IQ，infobright、infiniDB、GBase 8a，ParAccel, Sand/DNA Analytics和 Vertica等。

	Tuple1	Tuple2	Tuple3	Tuple4
o_orderkey	...			
o_custkey				
o_orderstatus				
o_totalprice				

表4-2 列存储数据排列