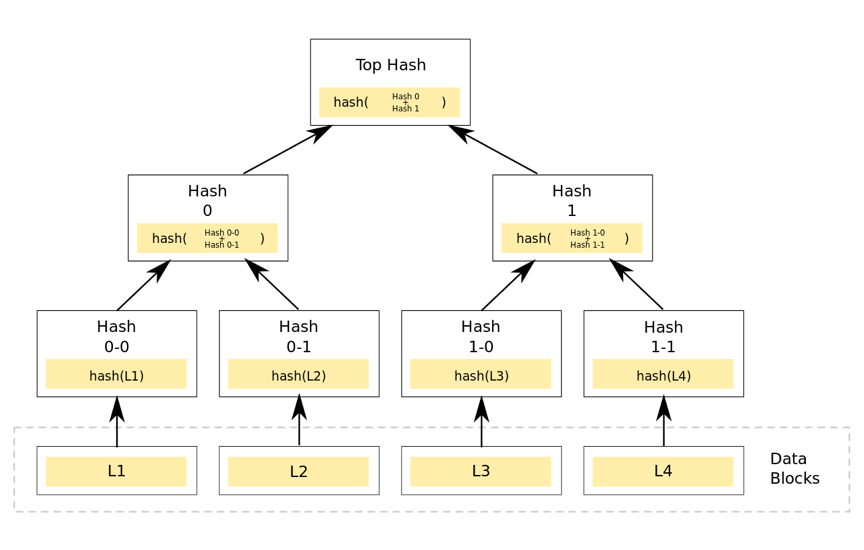
**Research Report on MPT**

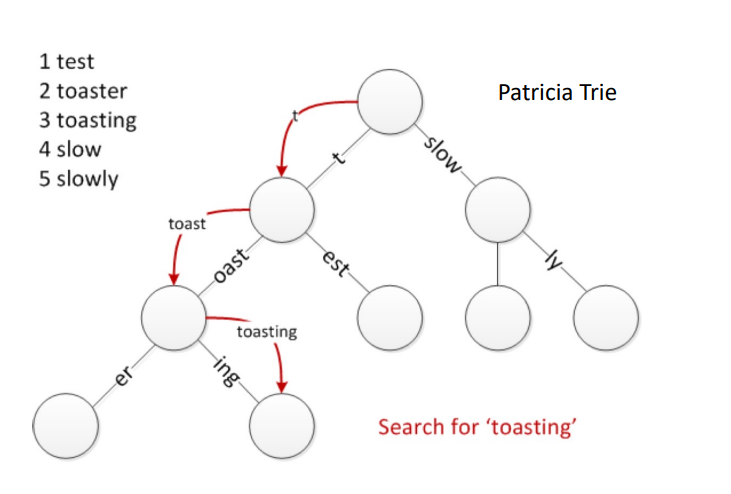
MPT全称Merkle Patricia Trie，在以太坊中用于组织和管理账户数据以及生成交易集合记录。MPT结合了Merkle tree和前缀树，形成了一种新的树形数据结构。

Merkle tree的基本结构如下：

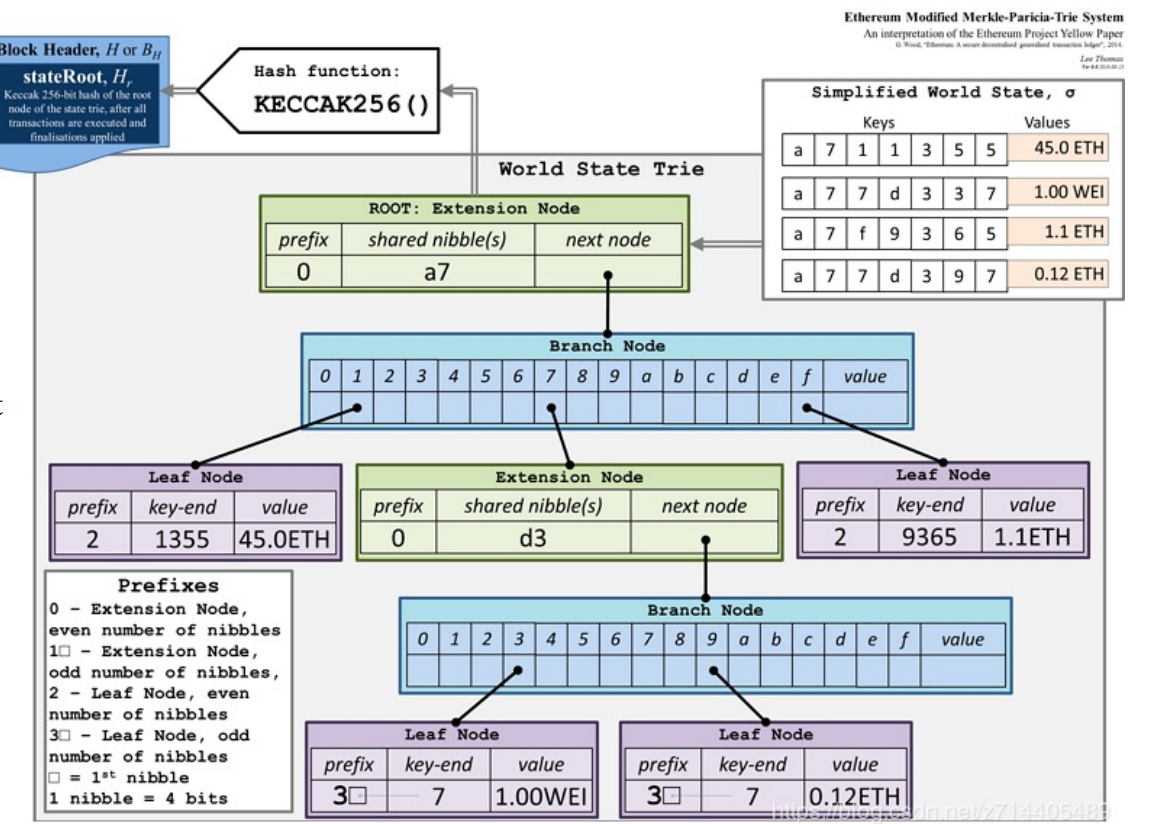


Merkle tree应用于比特币网络中，自底向上构建，底层的四个单元数据值依次hash并合并，形成了叶子结点和各自回溯的根结点。同时可以知道，如果两棵树的根哈希值相同，那么这两棵树的结构和每个结点存储的值也相同。Merkle tree的优势在于方便更新，即党某一结点的内容发生变化时，仅需将被修改的结点重新计算哈希值，即可得到一个新的根hash代表整棵树的状态。而它的缺点就在于存储空间的开销巨大。

前缀树的基本结构如图：



上图展现的是一个经过路径压缩的前缀图，其构造特点较为明显，用于保存关联数组，以键为每个结点的内容，最终呈现为字符串。前缀树的优点在于查询效率高于哈希表，且不会出现哈希冲突的问题；但其劣势为直接查找的效率过于低下，可能造成空间浪费。

MPT基于上述两种树形结构，设计了四类结点，以下图为例：

四类结点分别是空结点、扩展结点、分支结点和叶子结点。

扩展结点和叶子结点类似，有两个关键字段，分别是key和value，其中key解决了前缀树中可能存在的严重的存储空间浪费的情况，利用路径压缩，提高了查找效率，减少了存储浪费。因此其基本结构如下：

type shortNode struct {

Key []byte

Val node

flags nodeFlag

}

分支结点用于表示树中超过一个子结点的非叶子结点，其中的孩子列表最后一个内容是value，其余作为分支索引。其基本结构如下：

type fullNode struct {

children [17]node

flags nodeFlag

}

type nodeFlag struct {

hash hashNode

gen uint16

dirty bool

}

同时，在以太坊中，key也存在三种不同的编码方式，分别是Raw编码、

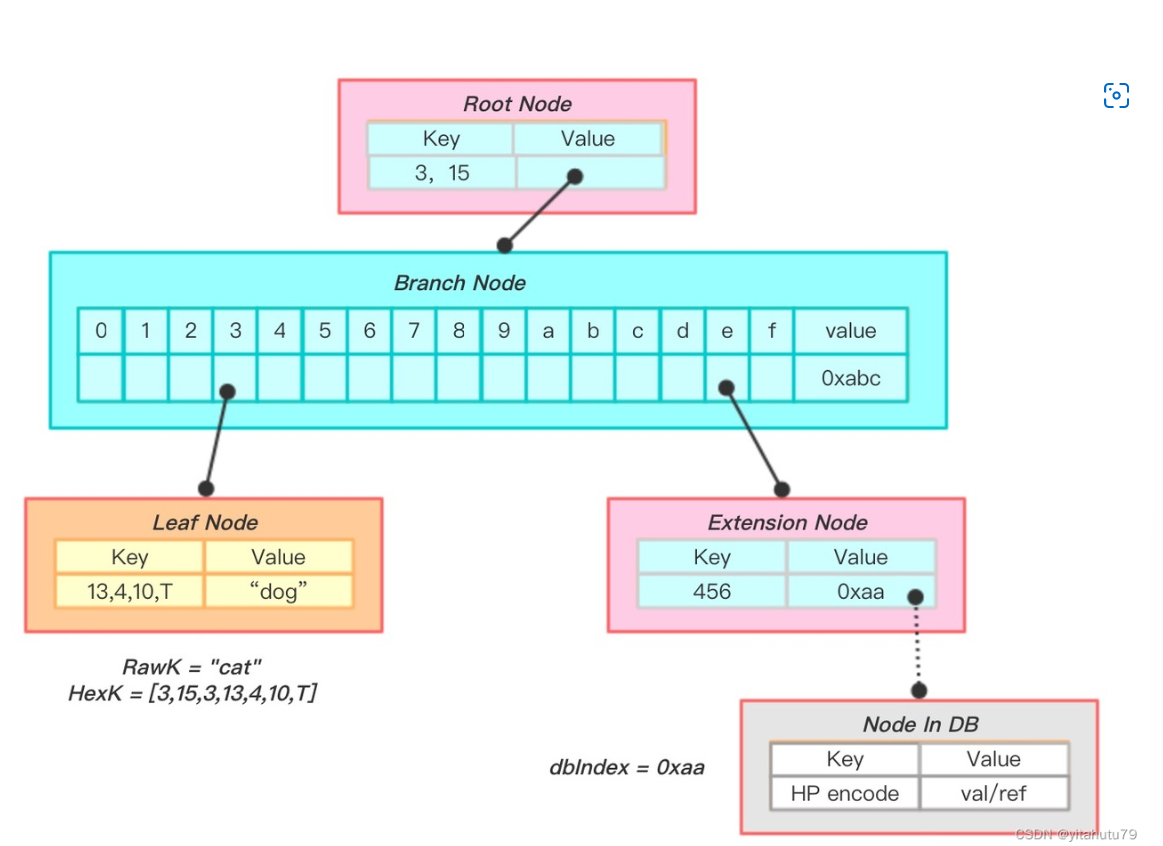
Hex编码、Hex-Prefix编码，且彼此之间存在转换关系，以raw编码为例，原生的key编码，是MPT对外提供接口中使用的编码方式，当数据项被插入到树中时，Raw编码被转换成Hex编码。

拥有上述结构和特点的MPT就可以实现任意长度的字典项（key-value对）的数据向存储，但由于没有限制，在维护较大数据量时会导致深度太大，从而降低搜索效率。因此以太坊中对数据项的key再计算一次hash，即最终作为参数传入的数对是(SHA3(key), value)，将传入MPT接口的key固定位32B的长度，避免路径过长，但相应的增加了存储负担，即需要额外存储hash前后的关系。

完成上述的组成结构之后，可以对MPT进行get、insert、delete、update、commit的操作。

下面简略介绍几个操作。

对于get操作，可以通过将需要查找的key的raw编码转换成hex编码，再从根结点开始搜索。下面引用一个例子：

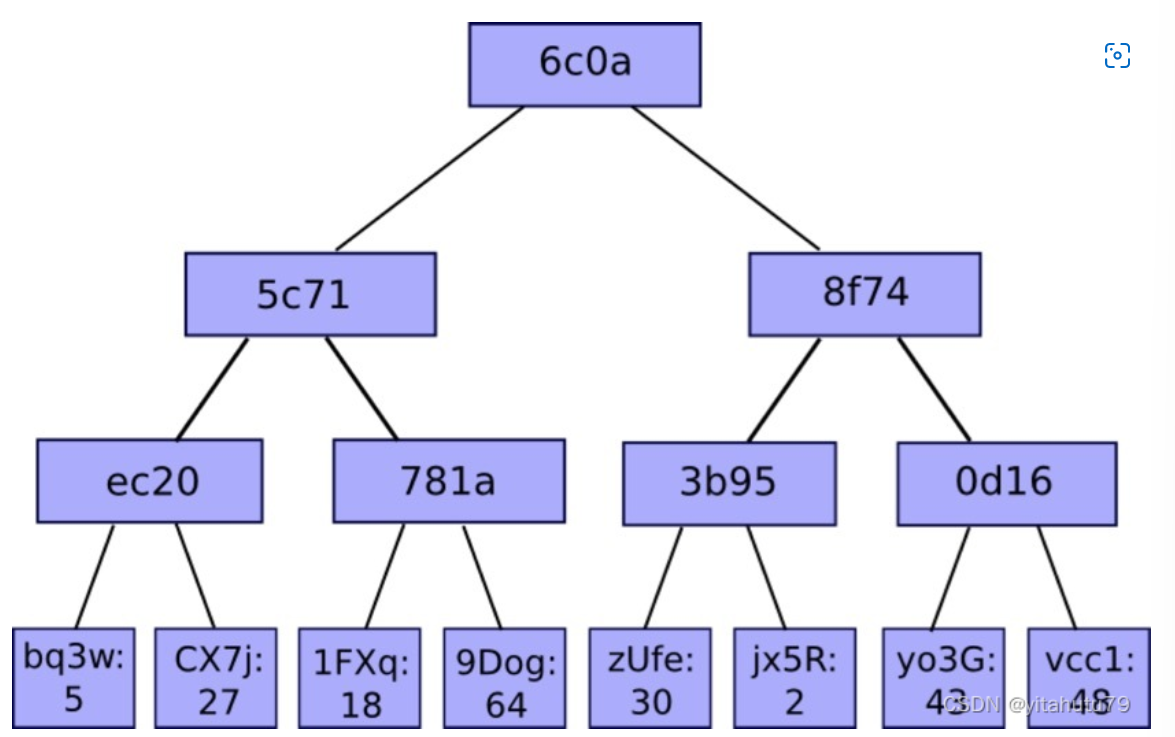


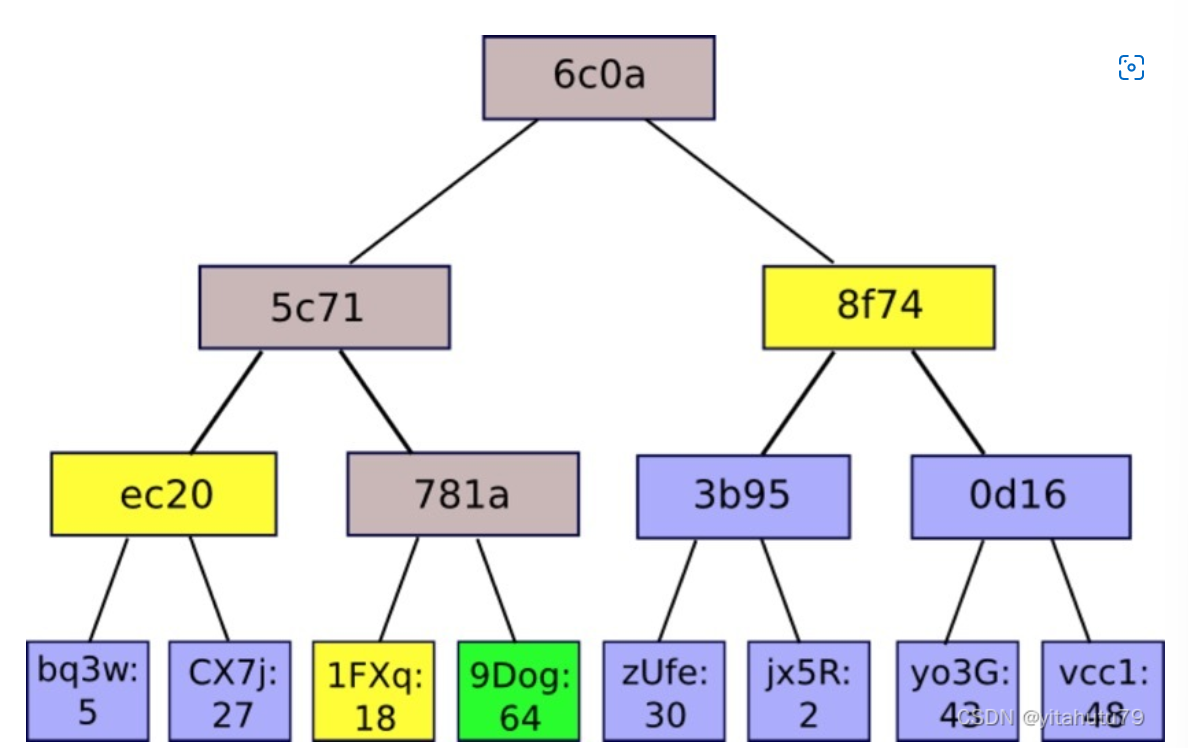
查找key为cat的结点，首先将关键字cat转换成hex编码[3,15,3,13,4,10,T]，结尾添加的终止符标志需要查找一个真实的数据项内容。因当前结点为根结点，同时是拓展结点，key值为3,15，由此递归对子节点进行查找，更新查找路径[3,13,4,10,T]。当前结点变为分支节点，按照路径第一个字节的内容，选择对4进行递归查找，更新查找路径[13,4,10,T]。接着当前结点变为叶子结点，而且与剩余查找路径内容相同，即找到目标节点，返回值value为dog。

Insert的操作类似，也是基于查找过程完成的，只是需要对查找到的结点进行类型的判断，再相应操作；delete同理，借助查找过程完成，判断节点类型后再进行操作，这里都不再赘述。Update的操作就是上述两个操作的结合，在value非空时调用insert，value为空时调用delete。

在结构与功能以外，MPT提供了一个重要功能，即Merkle证明，即一个结点向全结点发起一次证明请求，询问全结点是否存在一个指定结点，而全结点会返回一个路径，由该结点进行计算，验证其存在性。

下面简述merkle证明的过程，同样参考一个例子说明。





以某个结点想要验证9Dog:64是否存在于merkle tree中为例，在向全结点发送该请求后，会收到一个1FXq:18，ec20，8f74的返回值，得到路径后，即可用9Dog:64依次与1FXq:18、 ec20 8f74作hash运算，结果与根hash值对比是否相等即可。