今天给大家介绍一篇论文ForkBase: An Efficient Storage Engine for Blockchainand Forkable Applications
VLSD2018上的文章。作者团队来自新加坡国立。他们在区块链系统的评测方面做了很多工作。关注区块链和数据库结合的研究点。

forkbase 是一个为区块链以及 可分叉应用设计的 高效存储引擎

当前存储系统种类很多。像传统的关系型数据库，键值对数据库，存储文档的数据库。开发者根据他们的需求来选择比较适合的数据库。

随着一些新型应用的产生，对数据库提出了新的要求。区块链应用要求 存证，防篡改。
协同工作的数据集管理（分叉 合并操作，去重）Datahub

能否原生地支持这些新特性呢？

第一个Motivation 是要做到版本控制

在协作型（可能是结构化的）数据集管理中

如果是分散的文件或者目录，为了更好地管理，并且做到跨版本的去重，就需要git这样的版本控制。为了支持结构化数据，和内置的一些查询函数，就需要Datahub这样的数据管理系统。

更多需要的去重的场景包括

隐式去重
数据模型 中不同的版本对应不同的记录，如Wikipedia

全局去重
用户独立上传 相同的或者数据集的子集。如云存储

第二个motivation是去重
对于非结构化的数据来说，文件存储领域已经有深入的研究。经典方式是采用
基于块内容的去重。

对于结构化数据来说，页去重不是很有效（如B+树）。由于索引的性质，数据更新顺序不同的时候，索引也不同。右图，本来是两棵一样的数，需要插入5和9两个数字。先插入5和先插入9最后得到的索引结构是不同的。

forkbase 提出了一种新的索引 Structurally-Invariant Reusable Indexes (SIRI)
结构不变 可以重用的索引
可以比较高效地做页级别的去重

第三个motivation是普遍适用的分叉

这里的分叉语义有两种，主动和被动。
passive 被动，当发生不一致时，分叉。分支S1检测到w1和w2的改变，会分叉成 S2 和 S3。如amazon Dynamo DB

active 主动，当用户需要分叉时，主动进行分叉。如Git

forkbase 原生地支持这两种分叉语义
简化应用逻辑，对开发者来说更友好减少部署成本，同时保持较好的性能

forkbase支持区块链应用。当前区块链的存储一般用简单的键值对存储来实现，如level

密码哈希计算验证后形成区块链，链上的数据需要序列化后才能存储到数据库里，但是对于这些序列化的数据无法直接进行分析查询。

forkbase是第一个为区块链设计的原生存储。
提供防篡改的数据类型，来建模更复杂的数据结构
内置支持分叉语义和查询函数
将数据相关的逻辑从上层解耦合出来
使得区块链一旦产生就是可供分析的。

核心设计包括：默克尔有向无环图 进行版本控制和
防篡改存证；用SIRI索引来进行去除；使用协同工作流来实现 不同的分叉语义。

forkbase 支持了Git database的查询功能和区块链的 完整性验证。

工作栈
应用层可以支持 区块链 Git 等
语义视图层 基于分支的权限控制、数据完整性、合并操作的一致性。

在数据获取api方面，forkbase 提供 put get merge函数。

分支表示（实现版本控制） 块存储（实现去重、不可篡改）

具体来讲一下 结构不变　可以重用的索引。需要满足三个特性。

首先是结构不变，对于任何索引，如果页相同，则索引结构相同

第二递归相同，如果索引2等于索引1加一条记录，则大部分现存的页都是可以使用的。
更新后的索引 可以重用大部分的 页

普遍可重用，如果页p属于索引1，存在页p属于索引2，且索引2与索引1不同
任何的页都是可以重用的，并不单单属于某个实例。

一个符合上述条件的树实例

POS-Tree根据模式来进行分裂的树
1.基于内容进行节点分裂，所以结构是不变的。当一个预先定义的模式发生时，进行节点分裂

2.B+数的变种，是概率平衡的

3.默克尔树的变种，防篡改，可以去重

4.使用块类型的数据，一些数据结构

索引节点的模式用hash，数据节点用滚动哈希。

这页列举了通用分叉语义
Fork on Demand　和 Fork on　Conflict
显式分叉
接着分别给出两个示例，分别是 数据集管理和区块链如何使用forkbase命令来实现数据存储。

基于forkbase的区块链数据模型
使用的区块链平台是Fabric。RocksDB是键值对数据库。需要定制结构，如相连的区块，状态默克尔树，状态的改变值。实现难度较大。
forkbase实现了内置的类型如• UBlob • UMap。代码比较容易维护。

最后来看一下实验结果。部署在区块链上之后的效果。原本需要扫描整个区块历史记录，使用forkbase内置的数据类型就可以很快查找。对于状态的查询，对于区块的查询。延迟都比较低。

对比使用forkbase，rocksdb，ForkbaseKV后，在线交易的吞吐量。
ForkbaseKV是指把forkbase当作一个单纯地键值对数据库使用。可以看到交易吞吐量基本不受影响。

这张图比较了数据存储的延迟。rocksDB 后面这个数字是 桶 bucket
的数量,数量大的时候延迟会比较低。
tire指的是字典树。可见forkbase的延迟是比较稳定的。

第二组实验是用forkbase部署一个wiki应用，对照组是redis。可以看到POS-Tree的去重，节省了大量的内存空间。这张图是连续读操作的tps。可以看到读取的版本增加后，forkbase的性能优于redis。使用forkbase的数据结构，块数据可以缓存在客户端，在读取更早版本的块时，可以在缓存的基础上获取一个改变值来得到。

最后一组实验是在协作数据集管理应用上进行的。or ph eus
DB是VLDB2017上提出的一个数据库系统。
使用内置的数据类型建模关系型表。
POStree支持相同子树的快速剪枝。
无需在查询之前重组整个数据集。
比较在两个不同数据集差异比例增加时，查询时间的变化。
当差异比例较低时，forkbase可以快速地使用POStree来定位差别；比例过高时，forkbase需要遍历更多的树节点，导致延迟增长。

聚合查询
forkbase同时使用了行(row)和列。差了10倍。
这是因为forkbase无需在查询之前重组整个数据集。

两种分叉语义 可以支持不同的协作工作流程