MySQL 学习笔记 ----- 深入浅出索引（上）

作者：wky181
原文链接：https://ld246.com/article/1579095249266
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

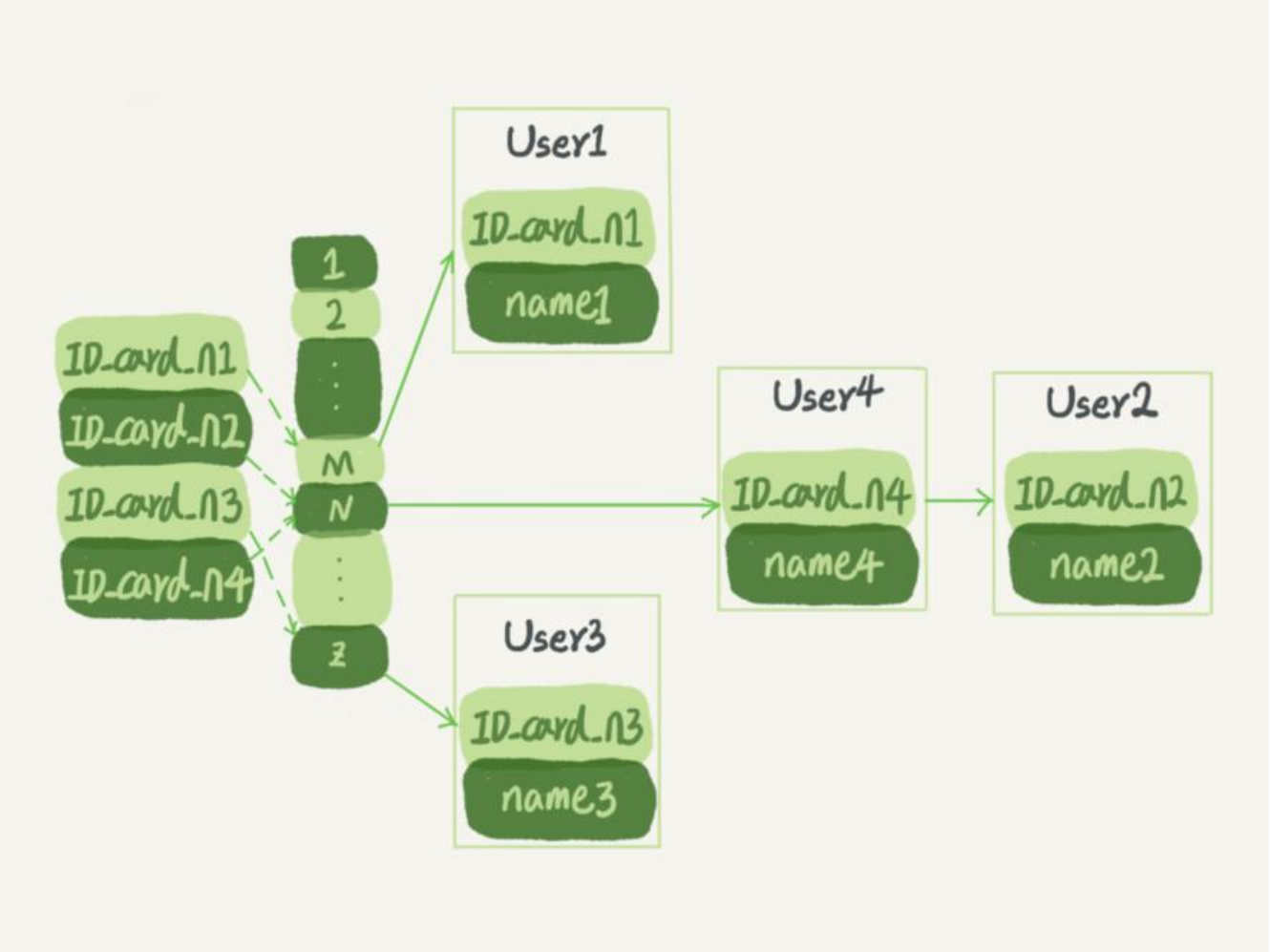
索引

可以把索引理解为书的目录，比如书有1000页，想要找你想要看的部分，如果不用目录，那么查找起就很费劲，用了目录之后，就能很方便的找到。所以索引就是为了方便查找数据，对于数据库的表而，索引其实就是它的“目录”。

索引的常见模型

索引有三种常见的模型，分别是哈希表、有序数组、搜索树。

哈希表是一种以键-值（key-value）存储数据的结构，我们只要输入待查找的值即key，就可以找到对应的值即Value。当出现哈希冲突时，一般采用的就是拉链法。注意这样的索引结构是无序的。

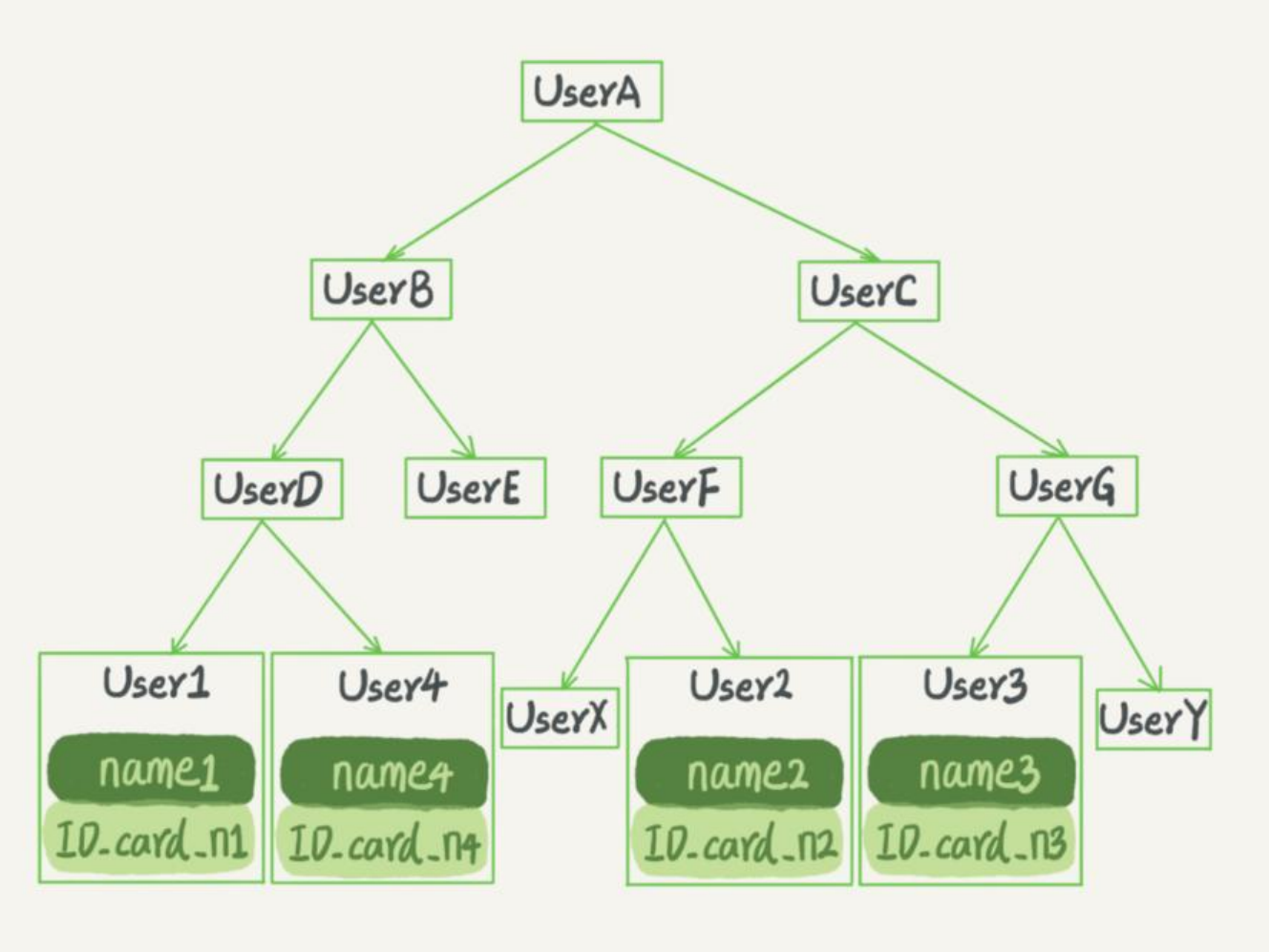


通过哈希函数计算要查询身份证号的哈希值，查到之后，按序查找相应链表中的值。对于单个查询这很方便，如果是顺序区间查询，就需要全部扫喵一遍了，因为哈希表是无序的。

对于有序数组，，**有序数组索引只适用于静态存储引擎**，表里的信息是不会改动的，可以才用二分法。

搜索树

如果根据用二叉搜索树，来建立索引。



二叉搜索树的特点是：每个节点的左儿子小于父节点，父节点又小于右儿子。这样以来，查找的时间杂度O(log(N))。为了维持这样的复杂度，需要保持这棵树是平衡二叉树，为了做这个保证，更新的间复杂度也是O(log(N))。

对于百万行的表来说，就是一个有100万行的节点二叉树，那么行高就是２０，从磁盘随机读一个数块需要10 ms左右的寻址时间。也就是说，对于一个100万行的表，如果使用二叉树来存储，单独访一个行可能需要20个10 ms的时间，访问速度太慢了。

所以我们不应该用二叉树，可以用N叉树。这样让查询过程访问尽量少的数据块，“N”取决于数据的大小。

InnoDB 的索引模型

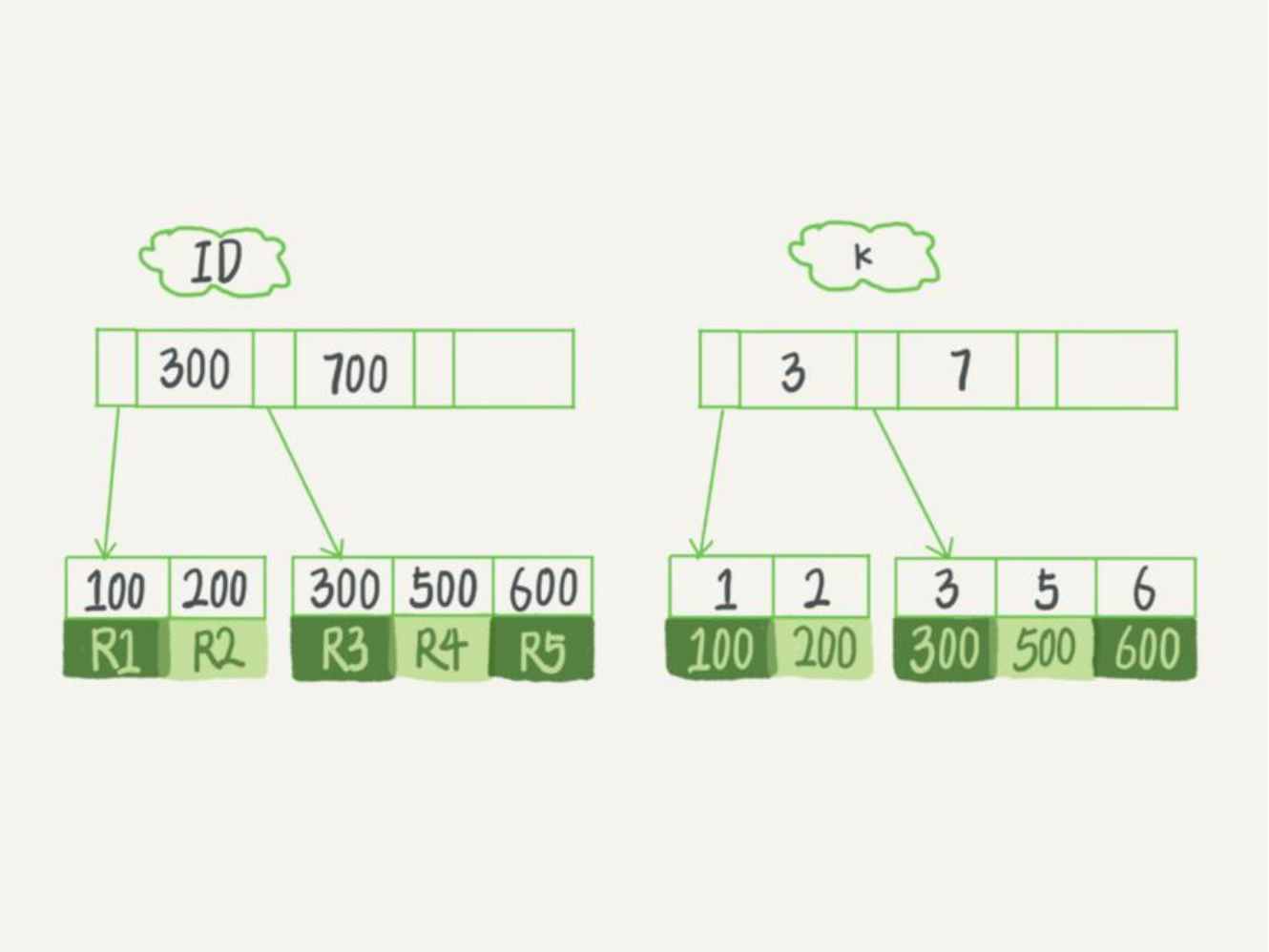
InnoDB使用了B+树索引模型，所以数据都是存储在B+树中的,每一个索引在InnoDB里面对应一棵B树。

假设，我们有一个主键列为ID的表，表中有字段k，并且在k上有索引。

```
mysql> create table T(
id int primary key,
k int not null,
name varchar(16),
index (k))engine=InnoDB;
```

表中R1~R5的(ID,k)值分别为(100,1)、(200,2)、(300,3)、(500,5)和(600,6)，两棵树的示例示意图如

。



B+树所有的关键字都出现在**叶子节点**的链表（稠密索引）中，且链表中的关键字是有序的。非叶子点只起索引作用（稀疏索引）。

由图可以看出，根据**叶子**节点的内容，可分为主键索引和普通索引，主键索引的叶子节点的数据域存的是该主键所在行的所有数据（聚簇索引），而普通索引子节点的数据域存放的主键（二级索引）。

如果语句是 select * from T where ID=500，即主键查询方式，则只需要搜索 ID 这棵 B+树。

如果语句是 select * from T where k=5，即普通索引查询方式，则需要先搜索 k 索引树，ID 的值为 00，再到 ID 索引树搜索一次。这个过程称**回表**。

也就是说，基于非主键索引的查询需要多扫描一棵索引树。因此，**在应用中应该尽量使用主键查询**

索引维护

B+树为了维护索引有序性，在插入新值的时候需要做必要的维护。如果新插入的ID值为400，就相对烦了，需要逻辑上挪动后面的数据，空出位置。如果数据页满了，按照B+Tree算法，新增加一个数页，叫做页分裂，会导致性能下降。空间利用率降低大概50%。

如果使用自增主键，每次插入操作都是在后面追加，不会导致叶子节点频繁移动，而触发页分裂。而且**然**，主键长度越小，普通索引的叶子节点就越小，普通索引占用的空间也就越小。**

重建索引

为什么要重建索引?

索引可能因为删除，或者页分裂等原因，导致数据页有空洞，重建索引的过程会创建一个新的索引，数据按顺序插入，这样页面的利用率最高，也就是索引更紧凑、更省空间。

对于上面例子中的InnoDB表T，如果你要重建普通索引 k，你的两个SQL语句可以这么写：

```
alter table T drop index k;
alter table T add index(k);
```

这样做就是把索引这棵B+树删掉，重建k索引，节省空间

对于重建主键索引，不能用下述sql语句做

```
alter table T drop primary key;
alter table T add primary key(id);
```

因为不论是删除主键还是创建主键，都会将整个表重建。

声明：此篇文章的主体内容参考自极客时间的MySQL实战45讲，非本人原创，只是在学习的过程中得有必要将其写入个人博客之中，且最终目的只是为了方便自己或有需要的人进行查阅。**此外，若需载本文仍需本人同意。**

上一篇：MySQL学习笔记-----事务隔离