第二版：MySQL 索引 6 道

目录

1、索引是什么?

索引是帮助 MySQL 高效获取数据的数据结构。

2、索引能干什么?

索引非常关键，尤其是当表中的数据量越来越大时，索引对于性能的影响愈发重要。 索引能够轻易将查询性能提高好几个数量级，总的来说就是可以明显的提高查询效率。

3、索引的分类?

1、从存储结构上来划分　BTree 索引（B-Tree 或 B+Tree 索引），Hash 索引，full-index 全文索引，R-Tree 索引。这里所描述的是索引存储时保存的形式，

2、从应用层次来分　普通索引，唯一索引，复合索引

3、根据中数据的物理顺序与键值的逻辑（索引）顺序关系：聚集索引，非聚集索引。

平时讲的索引类型一般是指在应用层次的划分。

就像手机分类 安卓手机，IOS 手机 与 华为手机，苹果手机，OPPO 手机一样。

普通索引：即一个索引只包含单个列，一个表可以有多个单列索引

唯一索引：索引列的值必须唯一，但允许有空值

复合索引：多列值组成一个索引，专门用于组合搜索，其效率大于索引合并

聚簇索引(聚集索引)：并不是一种单独的索引类型，而是一种数据存储方式。具体细节取决于不同的实现，InnoDB 的聚簇索引其实就是在同一个结构中保存了 B-Tree 索引(技术上来说是 B+Tree)和数据行。

非聚簇索引：不是聚簇索引，就是非聚簇索引

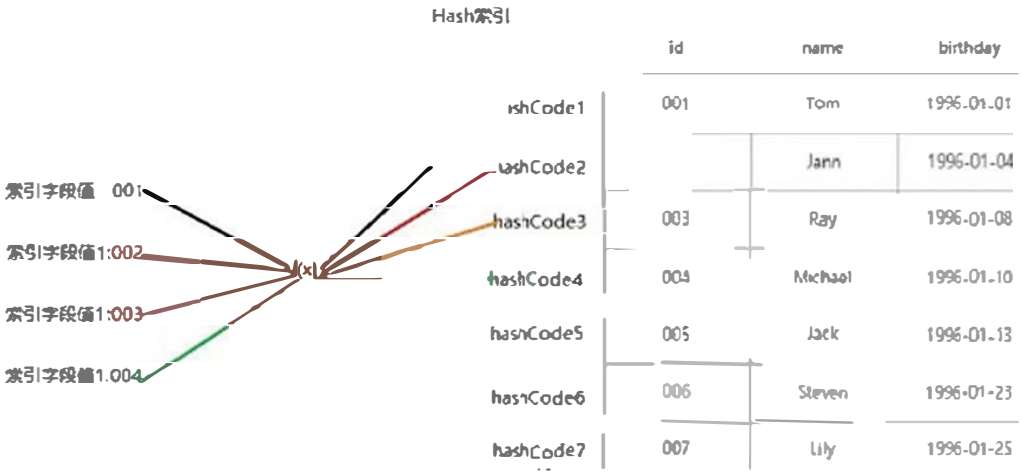
4、索引的底层实现

mysql 默认存储引擎 innodb 只显式支持 B-Tree(从技术上来说是 B+Tree)索引，对于频繁访问的表，innodb 会透明建立自适应 hash 索引，即在 B 树索引基础上建立 hash 索引，可以显著提高查找效率，对于客户端是透明的，不可控制的，隐式的。

不谈存储引擎，只讨论实现(抽象)

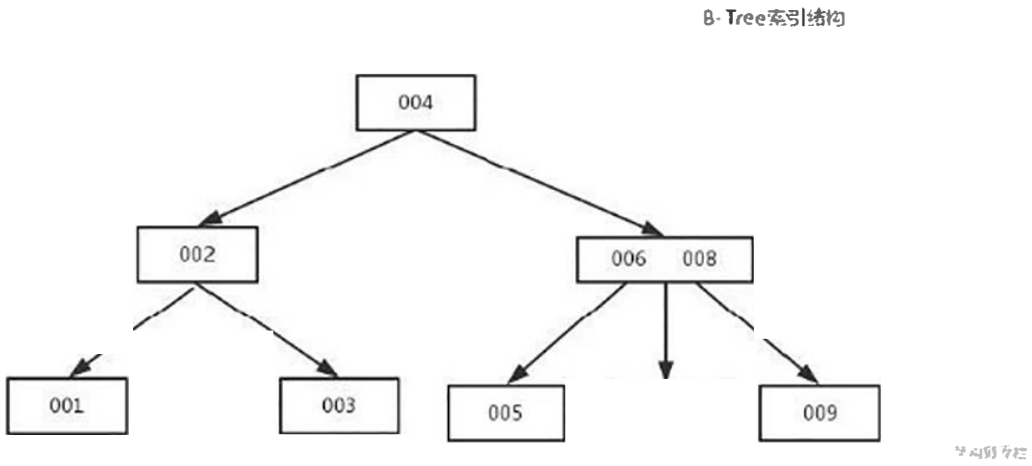
Hash 索引

基于哈希表实现，只有精确匹配索引所有列的查询才有效，对于每一行数据，存储引擎都会对所有的索引列计算一个哈希码（hash code），并且 Hash 索引将所有的哈希码存储在索引中，同时在索引表中保存指向每个数据行的指针。



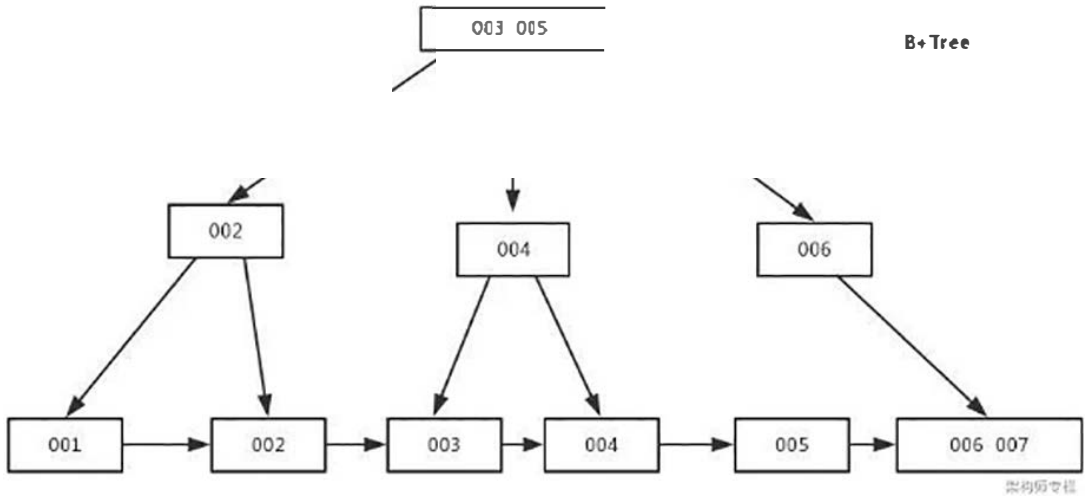
B-Tree 索引（MySQL 使用 B+Tree）

B-Tree 能加快数据的访问速度，因为存储引擎不再需要进行全表扫描来获取数据，数据分布在各个节点之中。



B+Tree 索引

是 B-Tree 的改进版本，同时也是数据库索引索引所采用的存储结构。数据都在叶子节点上，并且增加了顺序访问指针，每个叶子节点都指向相邻的叶子节点的地址。相比 B-Tree 来说，进行范围查找时只需要查找两个节点，进行遍历即可。而 B-Tree 需要获取所有节点，相比之下 B+Tree 效率更高。

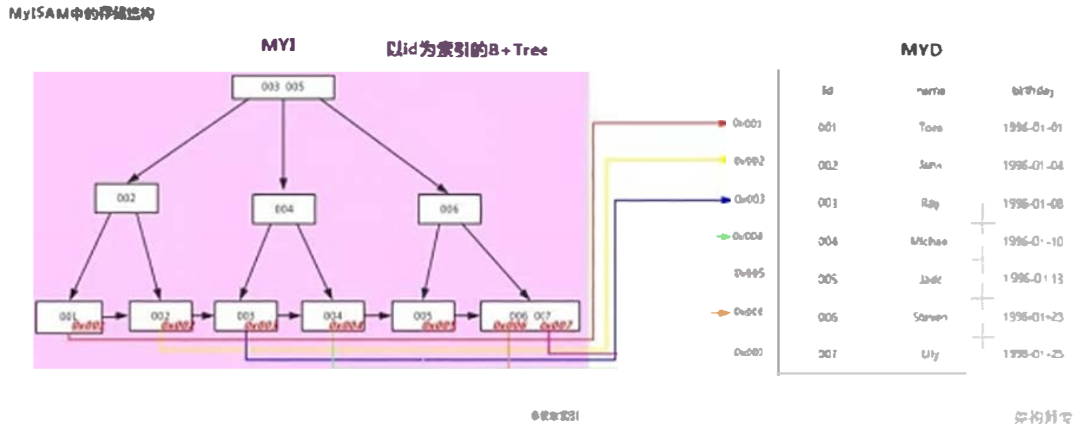


结合存储引擎来讨论（一般默认使用 B+Tree）

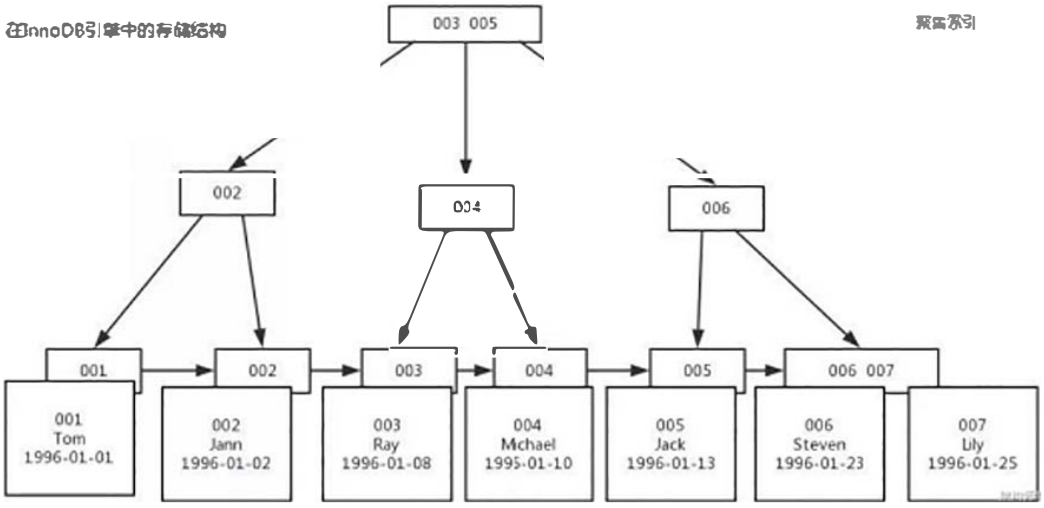
案例：假设有一张学生表，id 为主键

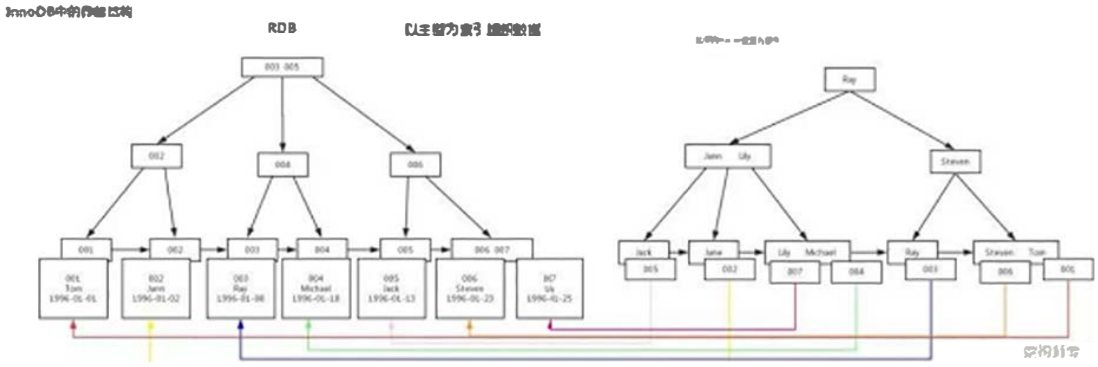
id	name	birthday
1	Tom	1996-01-01
2	Jann	1996-01-04
3	Ray	1996-01-08
4	Michael	1996-01-10
5	Jack	1996-01-13
6	Steven	1996-01-23
7	Lily	1996-01-25

在 MyISAM 引擎中的实现（二级索引也是这样实现的）



在 InnoDB 中的实现





5、为什么索引结构默认使用 B+Tree，而不是 Hash，二叉树，红黑树?

B+tree　因为 B 树不管叶子节点还是非叶子节点，都会保存数据，这样导致在非叶子节点中能保存的指针数量变少（有些资料也称为扇出），指针少的情况下要保存大量数据，只能增加树的高度，导致 IO 操作变多，查询性能变低；

Hash　虽然可以快速定位，但是没有顺序，IO 复杂度高。

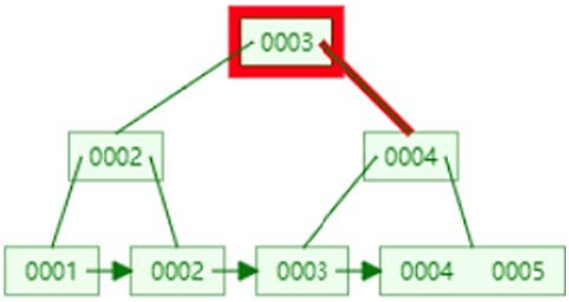
二叉树　树的高度不均匀，不能自平衡，查找效率跟数据有关（树的高度），并且 IO 代价高。

红黑树：树的高度随着数据量增加而增加，IO 代价高。

6、为什么官方建议使用自增长主键作为索引?

结合 B+Tree 的特点，自增主键是连续的，在插入过程中尽量减少页分裂，即使要进行页分裂，也只会分裂很少一部分。并且能减少数据的移动，每次插入都是插入到最后。总之就是减少分裂和移动的频率。

插入连续的数据：



插入非连续的数据

7、简单总结下

1、MySQL 使用 B+Tree 作为索引数据结构。 2、B+Tree 在新增数据时，会根据索引指定列的值对旧的 B+Tree 做调整。 4、从物理存储结构上说，B-Tree 和 B+Tree 都以页(4K)来划分节点的大小,但是由于 B+Tree 中中间节点不存储数据，因此 B+Tree 能够在同样大小的节点中，存储更多的 key，提高查找效率。 5、影响 MySQL 查找性能的主要还是磁盘 IO 次数，大部分是磁头移动到指定磁道的时间花费。 6、MyISAM 存储引擎下索引和数据存储是分离的，InnoDB 索引和数据存储在一起。 7、InnoDB 存储引擎下索引的实现，(辅助索引)全部是依赖于主索引建立的(辅助索引中叶子结点存储的并不是数据的地址，还是主索引的值，因

此，所有依赖于辅助索引的都是先根据辅助索引查到主索引，再根据主索引查数据的地址)。 8、由于 InnoDB 索引的特性，因此如果主索引不是自增的(id 作主键)，那么每次插入新的数据，都很可能对 B+Tree 的主索引进行重整，影响性能。因此，尽量以自增 id 作为 InnoDB 的主索引。