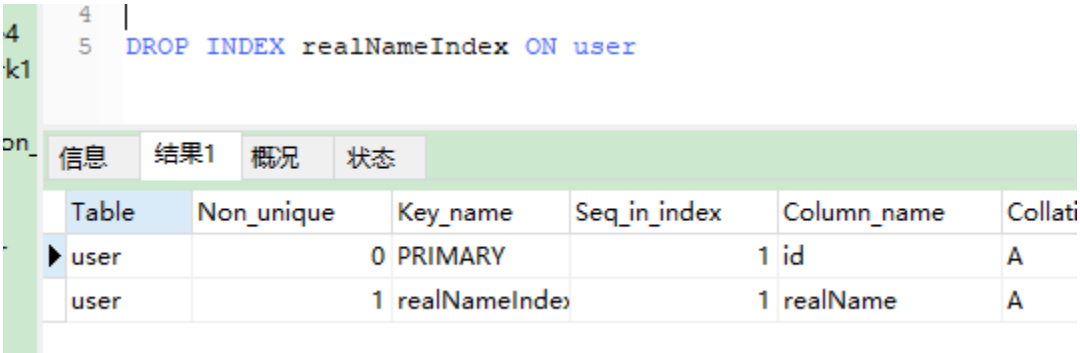
Mysql 索引学习笔记

(一) 概念

1. 概念：索引是一种可以提升查询效率但是降低修改、删除效率的数据结构。
2. 分类：
   a) Innodb 引擎下：
      - 主键索引：创建主键时自动创建的索引，因为是主键，所以不能有空值
      - 唯一索引：索引列的值必须是唯一的，但是可以有一个 null 值
      - 单值索引：一个索引只包含单个列
      - 复合索引：一个索引包含多个列

   b) MYISAM 引擎
      - 全文索引：
3. 索引的操作
   a) 展示索引：show index from ‘表名’;

   b) 删除索引：drop index 索引名 on 表名



   c) 增加索引
      - 主键索引创建主键是默认有索引
      - 唯一索引：create unique index ‘索引名’ on 表名(字段名);
      - 单值索引：create index ‘索引名’on 表名(字段名);
      - 复合索引：create index ‘索引名’on 表名(字段 1 名，字段 2 名);

```
create unique index realNameIndex on user (realName);

create index realNameIndex on user (realName);

create index realNameIndex on user (userName,realName);
```

结果1 | 概况 | 状态

le	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Carc
r	0	PRIMARY	1	id	A	
r	1	realNameIndex	1	realName	A	

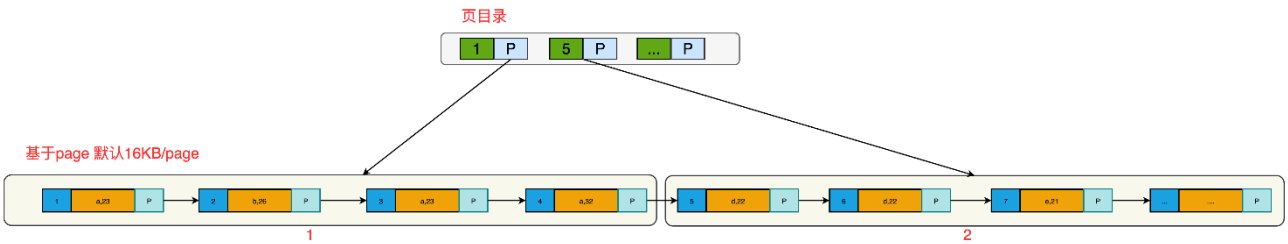
4. 经典的面试题
   a) 复合索引的性质：
      - 最左前缀原则：创建复合索引时，是按照从左到右顺序创建的，使用复合索引中的一个或是几个进行查询时，必须包含索引前面的所有数据才能使复合索引生效，否则复核索引失效。
      - 例如我们创建复核索引：
        create index realNameIndex on user (A,B,C,D,E,F,G);
        使用复核索引作为查询条件时：
        SELECT * FROM user WHERE A='' and B = '' ……..

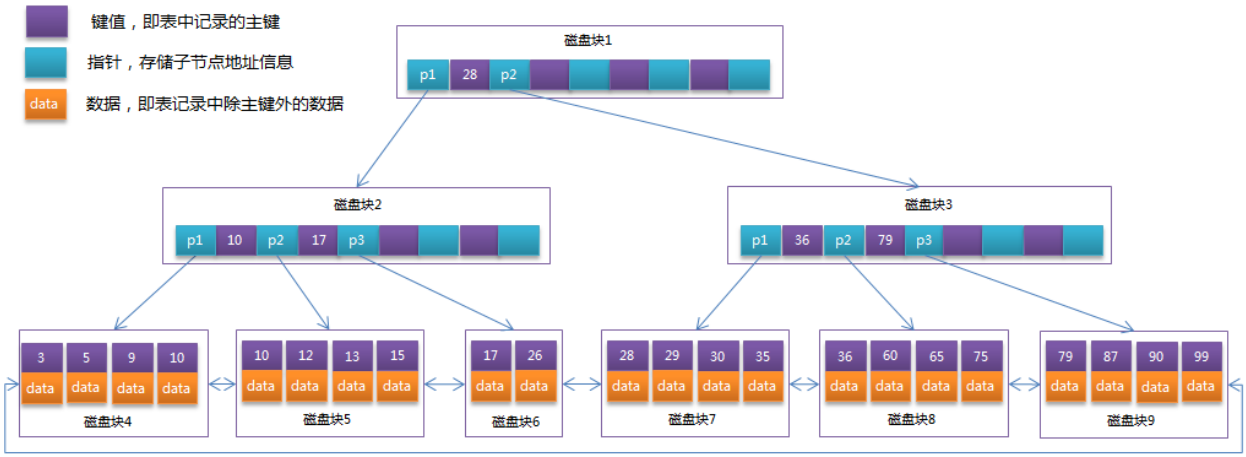
条件	整理后	有效/失效
A	A	有效
CAB	ABC	有效
EBAC	ABCE	缺少 D 失效
FEDCB	BCDEF	缺少 A 失效

(二) B+Tree(B+树)

1. 问：为什么索引的查询速度快呀？
   - 索引用的是 B+树进行的数据快速查询。
   - 当数据存入 MYSQL 中时，Mysql 会自动的根据索引值进行排序，形成链表一样的数据结构

   - 这样的数据结构每 16KB 会作为一页，并且每一页开头的索引值会上传值上一层，且生成指针指向该页，且该层同样为 16KB 一页。该层的每一页开头的索引值会同样上传至上一层，生成指针指向该层。





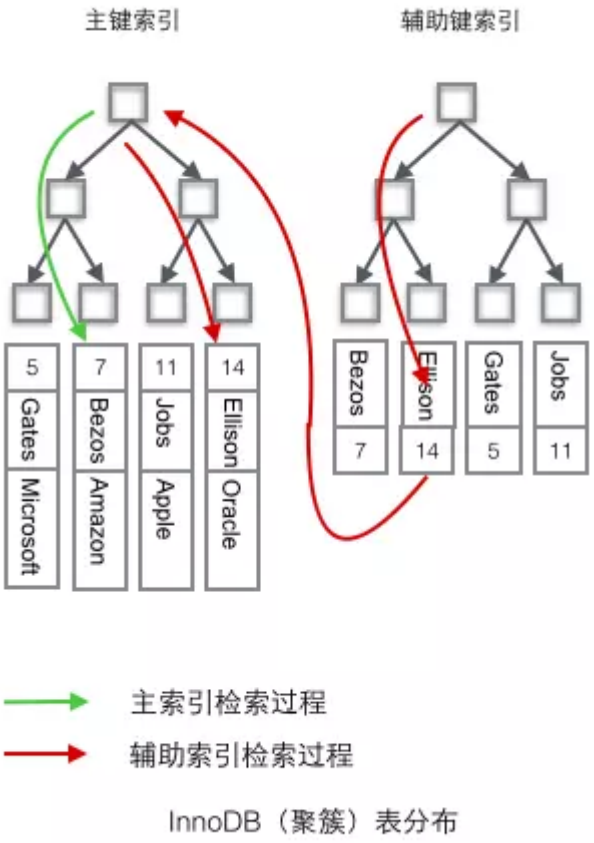
- 当我们进行数据查询时，会根据索引值，从最上层找到区间，一直定位到最下面具体页上的数据。

(三) 聚簇索引和非聚簇索引

a) 什么是聚簇索引，什么是非聚簇索引

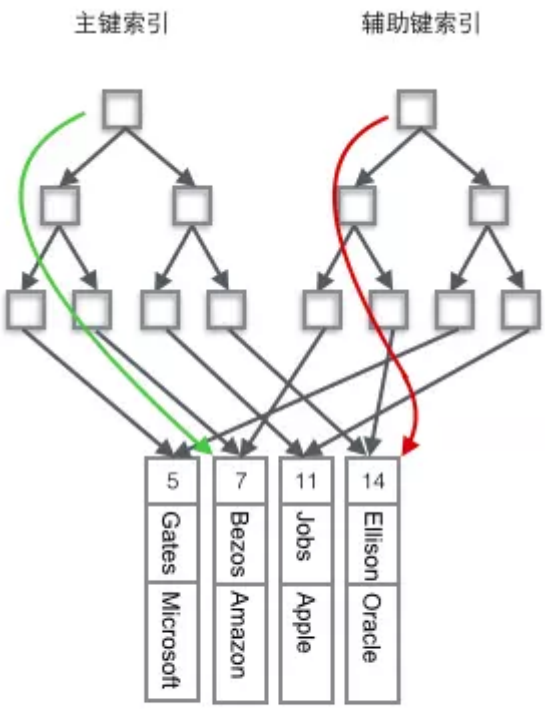
- 在 B+树中，页子节点（即最底层的链表的节点）存储的是索引值和相关数据，就叫聚簇索引
- 在 B+数中，页子节点存储的是索引值和主键（或物理地址）就叫非聚簇索引。
- InnoDB 中非聚簇索引需要依赖聚簇索引实现查找，我们需要根据非聚簇索引找到主键值，然后再去聚簇索引 B+树中找到该行的所有数据
- InnoDB 中一个表只能有一个聚簇索引，剩下的都是非聚簇索引。非聚簇索引之所以用主键而不用物理地址，是因为增删改会导致物理地址的重新排序。聚簇索引默认是组件的索引。
- MYISAM 中不存在聚簇索引，全部为非聚簇索引，且存储的是物理地址。

b) 下面是聚簇索引和非聚簇索引的例子。



InnoDB（聚簇）表分布

- InnoDB 使用的是聚簇索引，将主键组织到一棵 B+树中，而行数据就储存在叶子节点上，若使用"where id = 14"这样的条件查找主键，则按照 B+树的检索算法即可查找到对应的叶节点，之后获得行数据。
- 若对 Name 列进行条件搜索，则需要两个步骤：第一步在辅助索引 B+树中检索 Name，到达其叶子节点获取对应的主键。第二步使用主键在主索引 B+树种再执行一次 B+树检索操作，最终到达叶子节点即可获取整行数据。（重点在于通过其他键需要建立辅助索引）



MyISAM（非聚簇）表分布

c) 一些常见的面试题。

[1] 聚簇索引的好处

- 可以把每次查询的页放在缓存下，下次再次查询页中数据时，速度更快
- 增删改造作时，依赖聚簇索引的非聚簇索引存放的是主键，而不是物理地址。这样只需要主要维护聚簇索引就可以了，而纯非聚簇索引，则需要维护很多 B+树

[2] 聚簇索引注意事项

- 主键尽量不要用 UUID，因为 UUID 是随机无序的，在存储数据时，B+树底层的链表需要重新排列，应尽量使用自增的主键

(四) 索引失效的条件。

a) 如果表中 id name user password 为索引，且 user 和 password 为复合索引

b) 失效条件 1，使用 like 进行索引查询时，%放在模糊字段的前面

- 如：name like ‘%小红’ 索引失效
- 如：name like ‘小红%’ 索引正常使用

c) 失效条件 2：违反了复合索引的最左前缀原则

- 如：where password = ‘’ 索引失效
- 如：where user = ‘’ and password = ‘’ 索引正常使用

d) Or 查询时 左右两边必须是索引

- 如：where name = ‘’ or money = ‘’ 索引失效
- 如：where id = ‘’ or name = ‘’ 索引正常