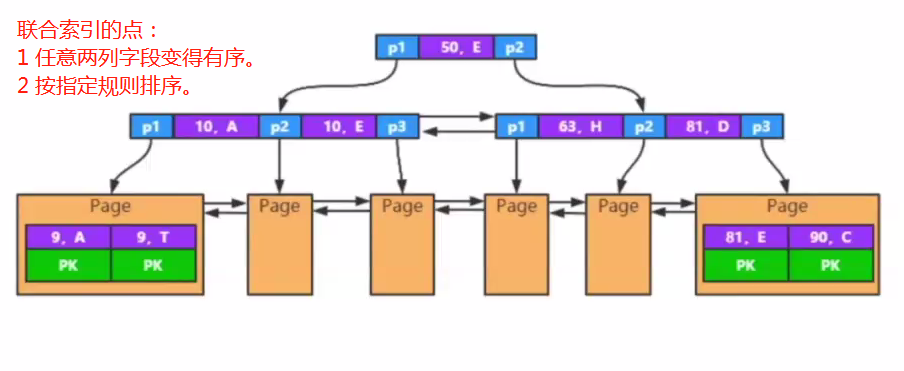
联合索引(优于多列独立索引,因为多个单列索引在多条件查询时只会生效第一个索引！所以多条件联合查询时最好建联合索引！)



Key由多个字段组成

最左匹配原则

一个联合索引只创建了一颗数

按第1列排序，第一列相同按第2列排序

如果不是按照最左开始查找，无法使用索引

不能跳过中间列  
某列使用范围查询，后面的列不能使用索引（A>xxx,b=xxx,c=xxx）

索引优化原则:

1 存储空间(数的层次太多会导致磁盘io变多)：

1.1 决定索引文件大小的是key的大小（在key字段个数已经定了的情况下）

1.2 key的大小决定了页内节点个数(因为key的个数决定了业内实际存储的数据个数)

1.3 页内节点个数(每个页的节点的多与少)又决定了树的层数（建多少层索引）

1.4 层数越多，根节点是常驻内存的，但是叶子节点未必，所以会发生越多的磁盘io.

一个bigint类型的主键，假设3层，可以存储约10亿条数据

计算公式

16KB/(8B(key)+8b(指针))==1k个key并指向下一层

根页有1k个指针往下指，一直往下指

1k X 1k X 1k =10亿个key

同理 如果32位字节主键3层可以存6400万条数据

16/(32+8)=400

400 \* 400 \* 400 = 6400w

2 主键选择

2.1 自增主键，顺序写入，写入磁盘效率高（但会暴露商机userid，orderid，goodsid）

2.2 随机主键(非主键索引)，逻辑有序，但会产生节点分裂（比如要插入的节点空间不够），要数据移动（产生磁盘io）,而且由于不是主键索引，要走最少2次才能找到数据，写入磁盘效率低

2.3 业务主键(雪花算法等) 写入性能接近自增主键，可以使用一级索引等

2.4 联合主键： 影响索引大小，不易维护，不建议使用

3 联合索引使用

3.1 按索引区分度排序

性别只有男女，两个；年龄只有0-99，100个

假设找到了性别为男的，又继续找年龄为20，因为会产生多次回表（回到性别），所以效率低.（先两个，再100个）

只有先年龄再性别，才会减少回表的次数，才会效率高。

3.2 覆盖索引

假设有pk，uid，phone，都是单个字段做索引。

假设有99%的查询都是根据uid找到phone.(select phone from user where uid =xx)

根据uid找到pk，再根据pk找到phone。

这个效率很一般了，所以会有联合索引.

比方说建uid，phone的联合索引.

只要找到uid，就直接返回phone.

3.3 索引下推

Mysql5.6的版本上推出,例如a=xxx，b>100, mysql之前应该是把a=100的条件下推给存储引擎层，但是没有把b>100下推到存储引擎层（innodb层）， 现在都把条件一起下推给存储引擎层.

4 字符串索引

设置合理长度(前32位，16位，8位)

不支持%开头的模糊查询

5 什么情况下会导致索引失效

5.1 区分度太低，多次回表，就会被mysql认为索引失效.

5.2 a=xxx or b =xxx,就算建立两张索引，也会只用一张索引，导致另外的索引不能用。

但从mysql5.1开始，引入index merge技术，对同一个表可以使用多个索引分别扫描

5.3 隐式类型转化

Where mobile = 135xxxx

Where mobile =‘135xxx‘

5.4 索引列包含计算

Where name - 20 = 30

5.5 条件超出索引范围(数据量不够)

有个age字段，数据库中的值范围为10-100.

查询的时候age>0,则不命中索引（因为所有的数据都符合>0,那干脆走全表扫描了）

Mysql库表设计实践

数据库字符集使用utf8mb4

Varchar按实际需要分配长度

文本字段建议使用varchar(用基本数据类型，哪怕换数据库也支持)

时间字段建议使用long(13位就够一个毫秒级)

Bool字段建议使用tinyint

枚举字段建议使用tinyint

交易金额建议使用long（1.8元可以表示180分）

表必须有主键，建议使用业务主键

单张表中索引数量不超过5个

单个索引字段数不超过5个(5个字段组成联合索引已经很大了)

字符串索引使用前缀索引，前缀长度不超过10个字符