# mysql的索引

解决的问题：

1. 什么是聚集索引？
2. 为什么建议主键使用自增id？
3. 索引的本质是什么？
4. 为什么会存在最左前缀原则
5. 一条sql到底什么时候走索引？什么时候进行全表扫描？

## b+tree是如何生成的

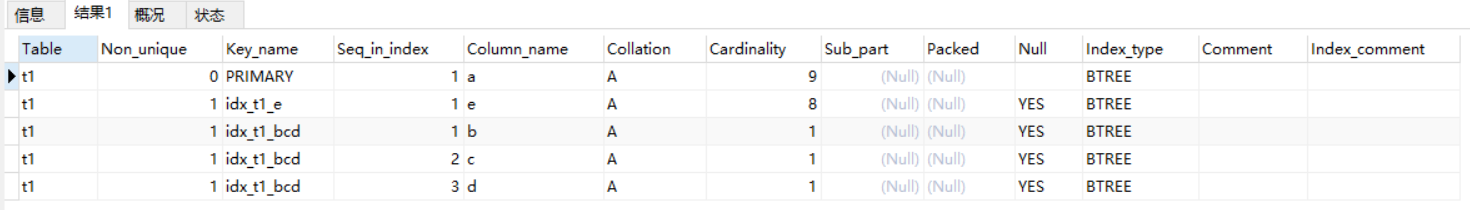
## 2 索引优化实战

## 3 面试分析

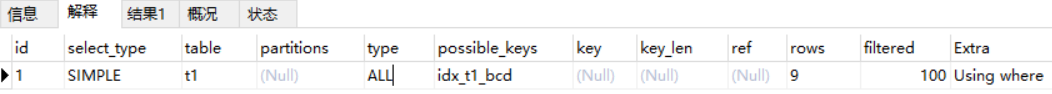
sql:

|  |
| --- |
| create table t1(  a int primary key,  b int,  c int,  d int,  e varchar(20)  )engine=InnoDB;  insert into t1 values(4,3,1,1,'d');  insert into t1 values(1,1,3,1,'a');  insert into t1 values(8,8,3,6,'c');  insert into t1 values(2,3,2,4,'c');  insert into t1 values(5,3,6,1,'f');  insert into t1 values(3,2,8,2,'g');  insert into t1 values(7,4,1,1,'r');  insert into t1 values(6,3,9,8,'y');  insert into t1 values(9,6,1,1,'w');  create index idx\_t1\_bcd on t1(b,c,d);  create index idx\_t1\_e on t1(e);  alter table t1 drop index idx\_t1\_bcd;  show index from t1; |

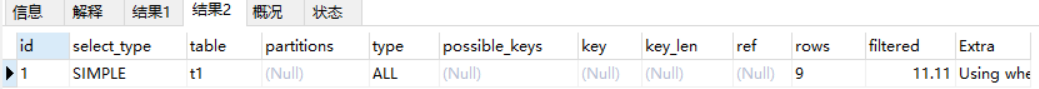




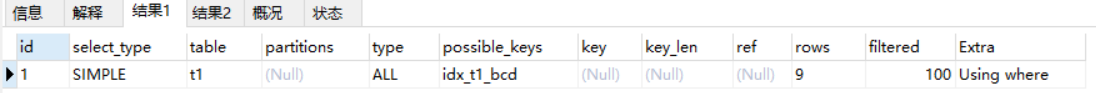
select \* from t1 where b =3;



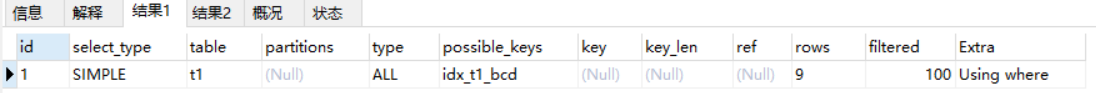
EXPLAIN select \* from t1 where c =1;

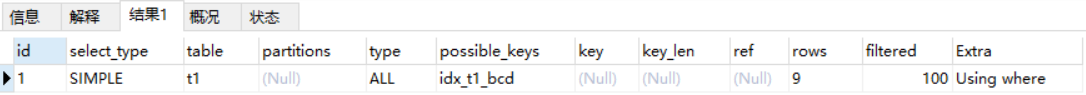


EXPLAIN select \* from t1 where b=3 and c =1;

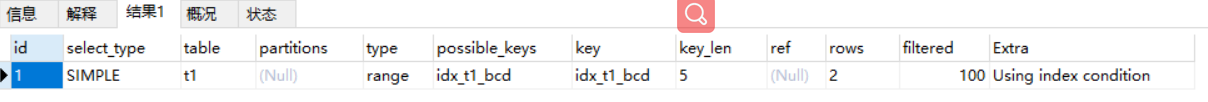


EXPLAIN select \* from t1 where c =1 and b=3;



EXPLAIN select \* from t1 where b >1; 

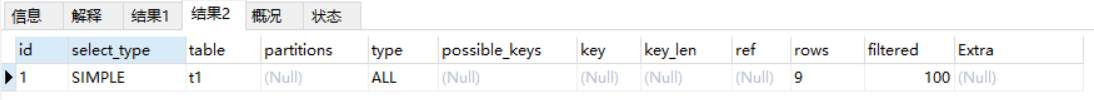
EXPLAIN select \* from t1 where b >5;



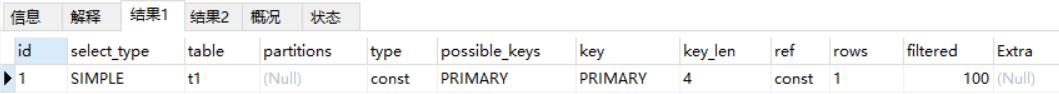
EXPLAIN select b from t1;



EXPLAIN select b,e from t1;



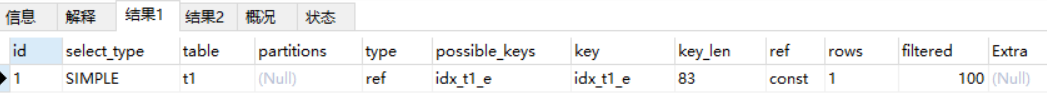
EXPLAIN select \* from t1 where a='1';



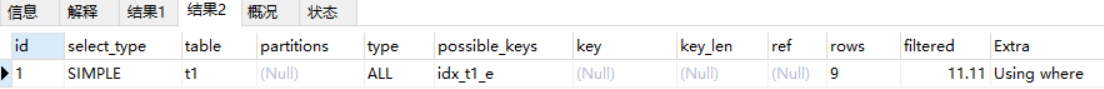
EXPLAIN select \* from t1 where a=1;



EXPLAIN select \* from t1 where e='1';



EXPLAIN select \* from t1 where e=1;



EXPLAIN select \* from t1 ORDER BY b asc, c asc, d asc;



EXPLAIN select \* from t1 ORDER BY b asc, c asc, d desc;



mysql基本概念：

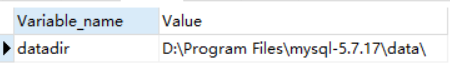
show GLOBAL STATUS like 'innodb\_page\_size'; --页 4kb



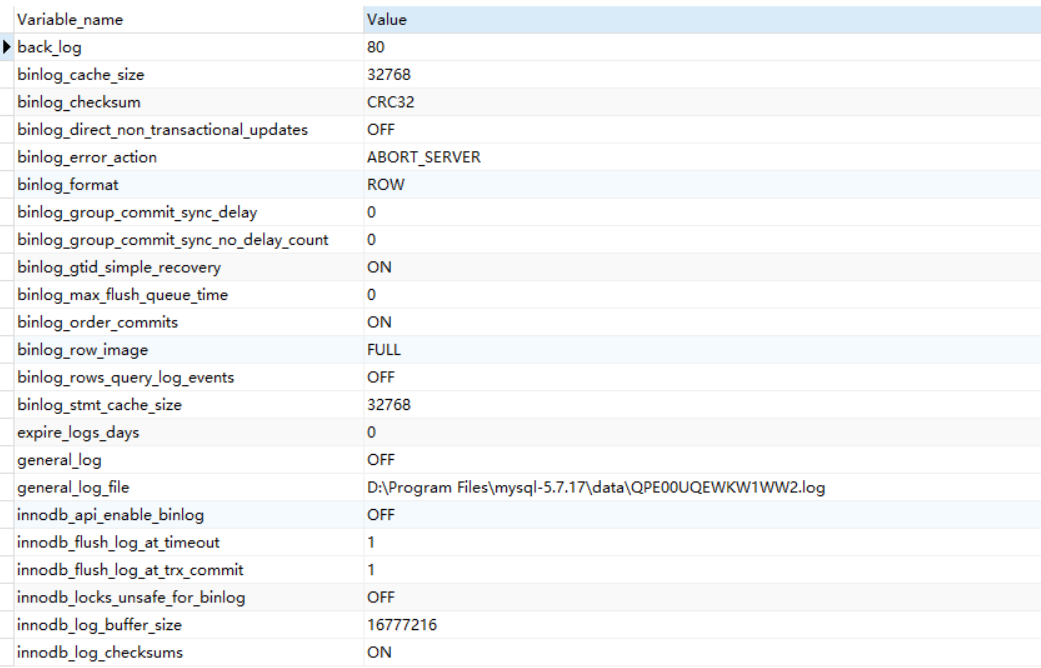
select 16384/1024 ;

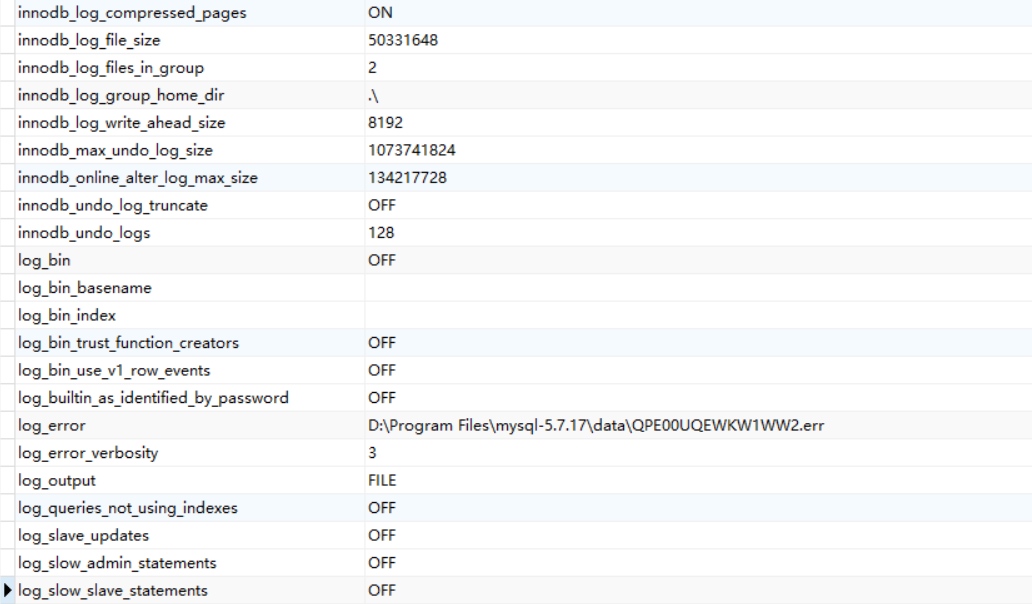


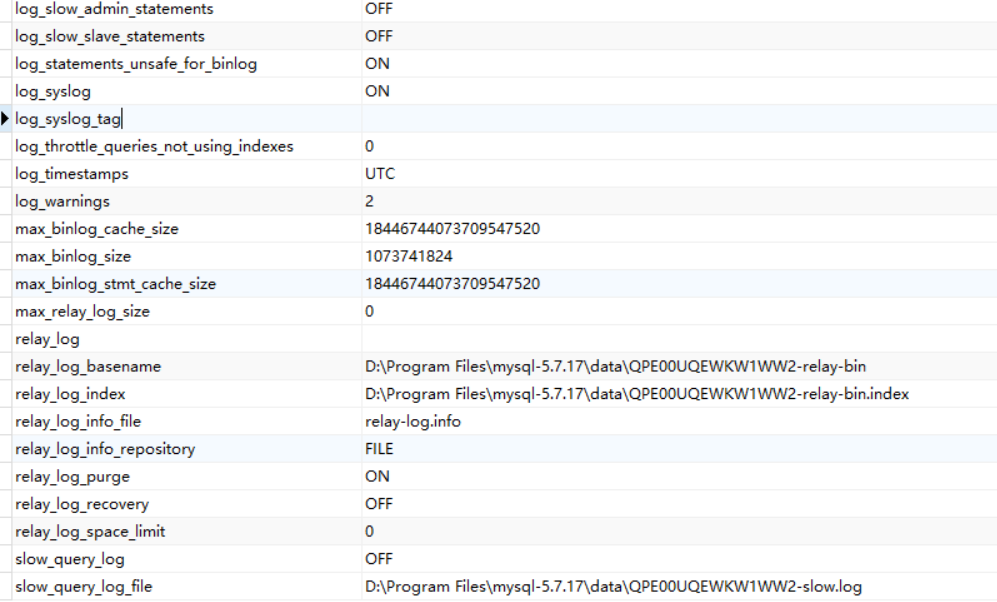
show GLOBAL VARIABLES like '%datadir%';

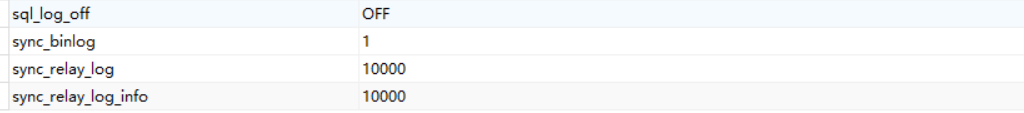


show GLOBAL VARIABLES like '%log%';









show GLOBAL VARIABLES like '%innodb\_file\_per\_table%';



show GLOBAL VARIABLES like '%tmp\_table\_size%';

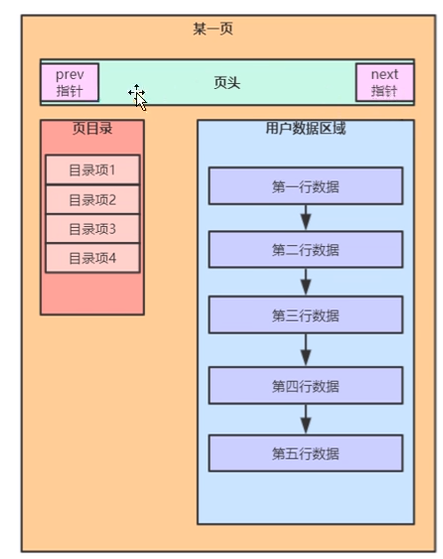


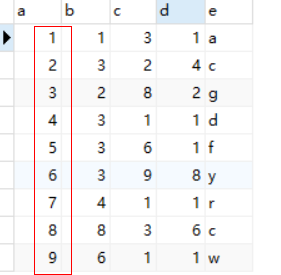
show GLOBAL VARIABLES like '%innodb\_stats\_method%';

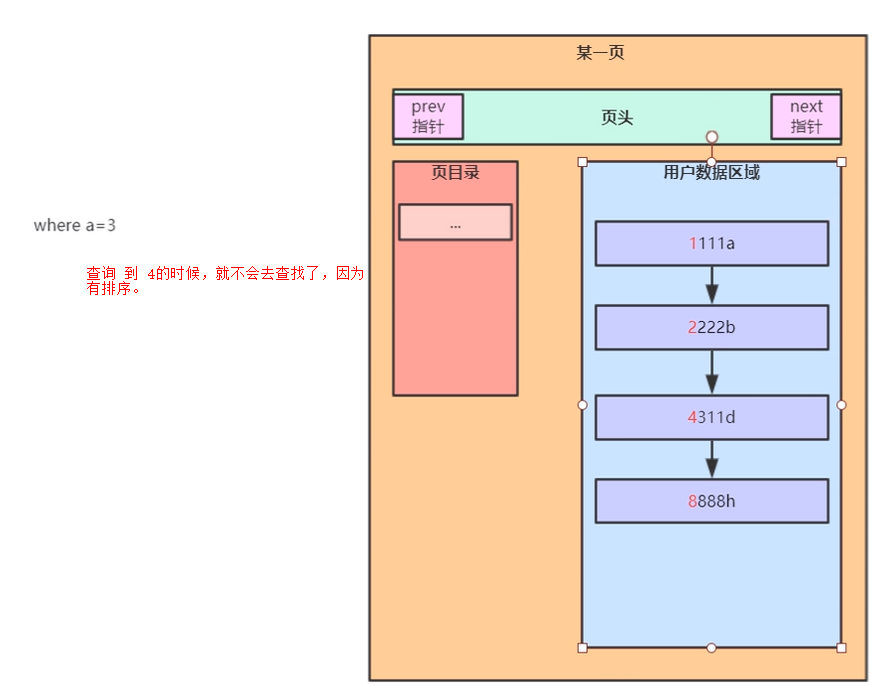


去磁盘取出第一条数据 放到内存 交给cpu去比较，如果不符合，继续去磁盘取第二条数据（两次磁盘io了），每次从磁盘取出一条数据就不好了，引出一个 页 的概念，页（16kb），这里的8条数据，比如这里一行数据占了10多个字节，去取第一条数据的时候顺便把以下的数据行业放到内存中，下次去从内存中找数据。

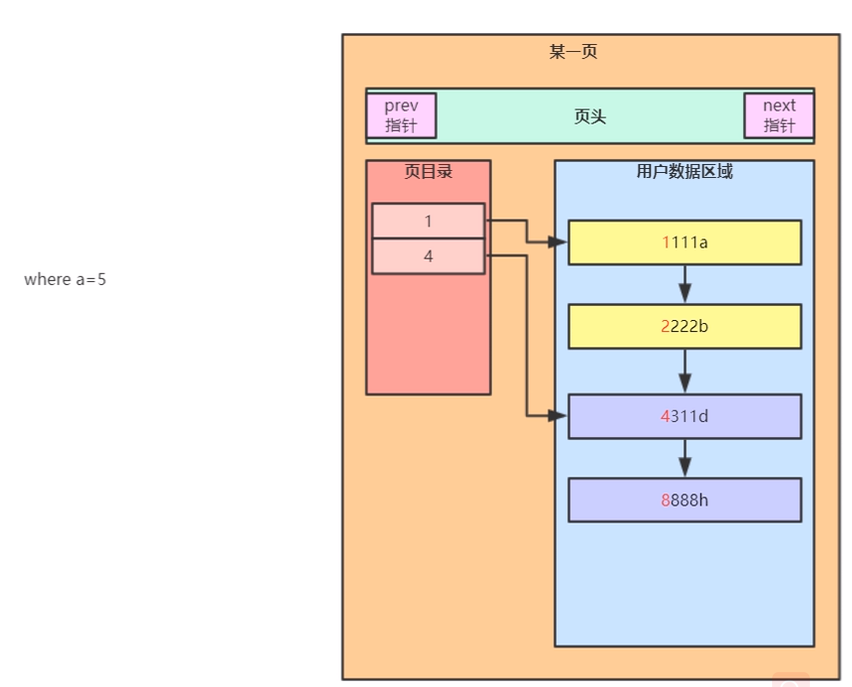
页结构



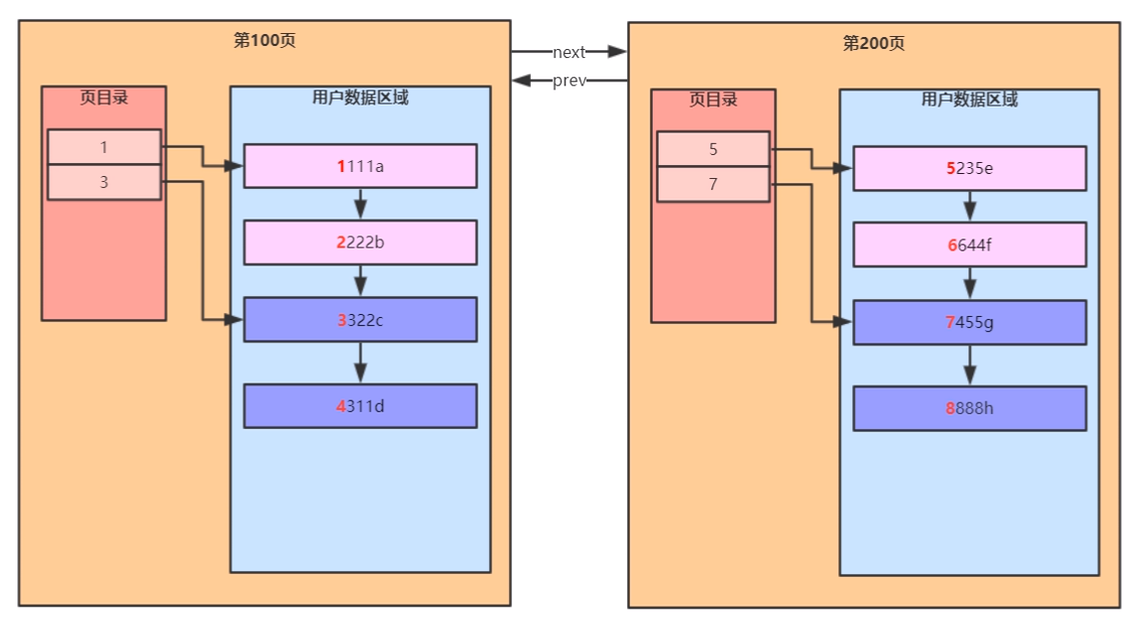
在插入的时候按照 主键 升序排序，即使主键有乱序，mysql内部会自动给排序了。



这里这时候会出现 链表的查询结构，如果查询的数字 比较大的时候，这个链表就太长了，影响查询效率，要改进。

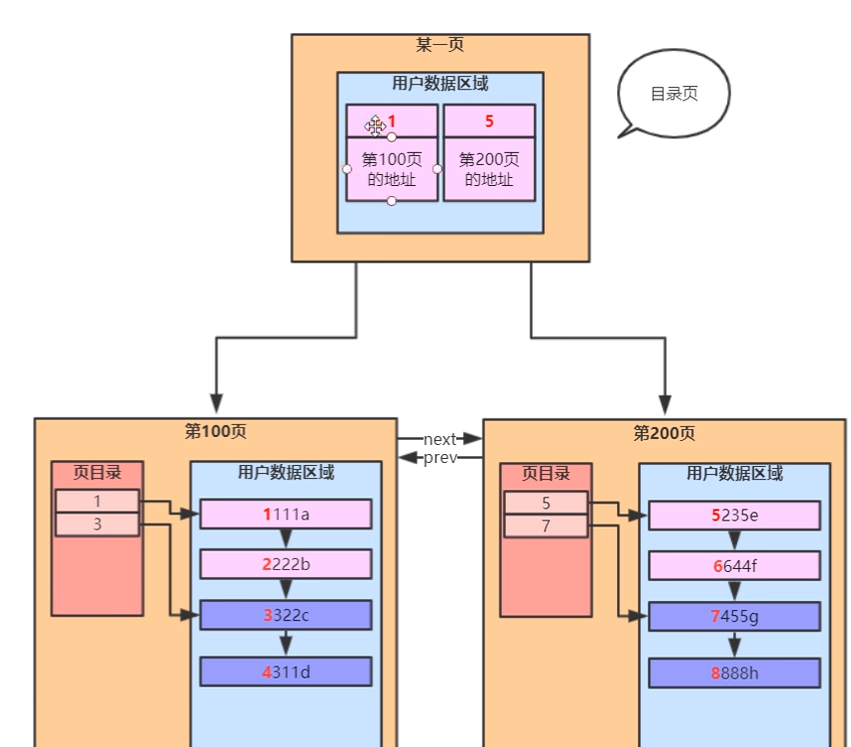


两页（如果一行记录占得字节比较大，一页只能放4条记录）



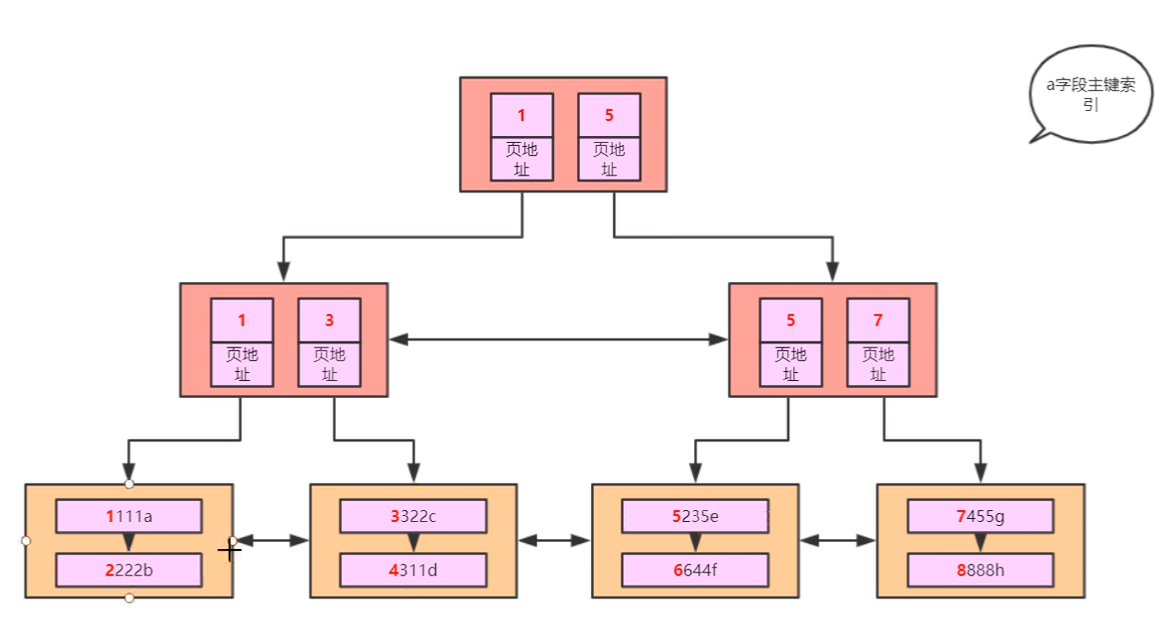
利用页目录解决了 数据行 形成链表过长的问题，但是页数比较多的情况呢？

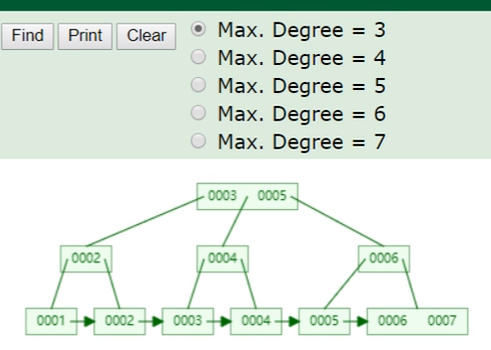
目录页 数据页 页目录（目录项）



目录 就是 索引

如果是两条记录占用一页的话：





叶子节点都有一个冗余，冗余索引的键值。

全表扫描：从叶子节点开始找

走索引：从目录页 开始查找

如果没有主键 如何排序呢？就会把唯一索引（去找）当一个主键来排序。唯一索引也没有的话，自己生成一个隐藏的id（rowid），是一定要有一个主键的。

create index idx\_t1\_bcd on t1(b,c,d);

