

目录

[P1 001.MySQL如何实现的索引机制 1](#_Toc148280851)

[P2 002.InnoDB索引与MyISAM索引实现的区别是什么 2](#_Toc148280852)

[MylSAM 2](#_Toc148280853)

[P3 003.—个表中如果没有创建索引，那么还会.. 3](#_Toc148280854)

[P4 004.说—下B+树索引实现原理〔数据结构) 3](#_Toc148280855)

[P5 005.聚簇索引与非聚簇索引b+树实现有什... 3](#_Toc148280856)

[P6 006.说—下B+树中聚簇索引的查找（匹配... 3](#_Toc148280857)

[P7 007.说—下B+树中非聚簇索引的查找（匹... 3](#_Toc148280858)

[P8 008.平衡二叉树，红黑树，B树和B+树的区... 3](#_Toc148280859)

[P9 009.—个b+树中大概能存放多少条索引记 3](#_Toc148280860)

# P1 001.MySQL如何实现的索引机制

MySQL中索引分三类:B+树索引、Hash索引、全文索引

B+树索引，存放全量数据；Hash索引快速定位，做一些辅助上的查询；两者同时存在，hash配合B+使用。

# P2 002.InnoDB索引与MyISAM索引实现的区别是什么

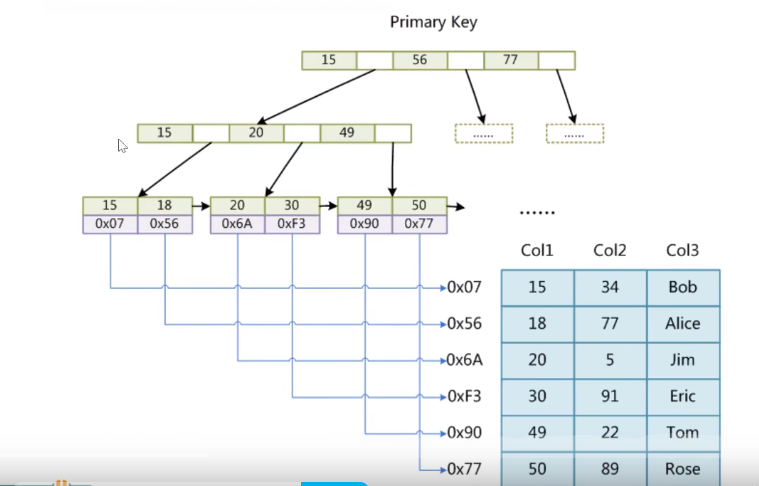
InnoDB索引存储和数据放在同一个文件里。数据会存在磁盘文件上，索引就直接在文件里追加下。

MyISAM文件和数据不是存放在一起。

B+数的叶子节点存放的数据类型，InnoDB指向的是Id,MyISAM指向的是指针，或者是物理地址。

## MylSAM

MylSAM索引文件和数据文件是分离的，使用B+树实现，主键索引和辅助索引实现一致，索引文件仅保存记录所在页的指针（物理位置)，通过这些地址来读取页，进而读取被索引的行。



## InnoDB

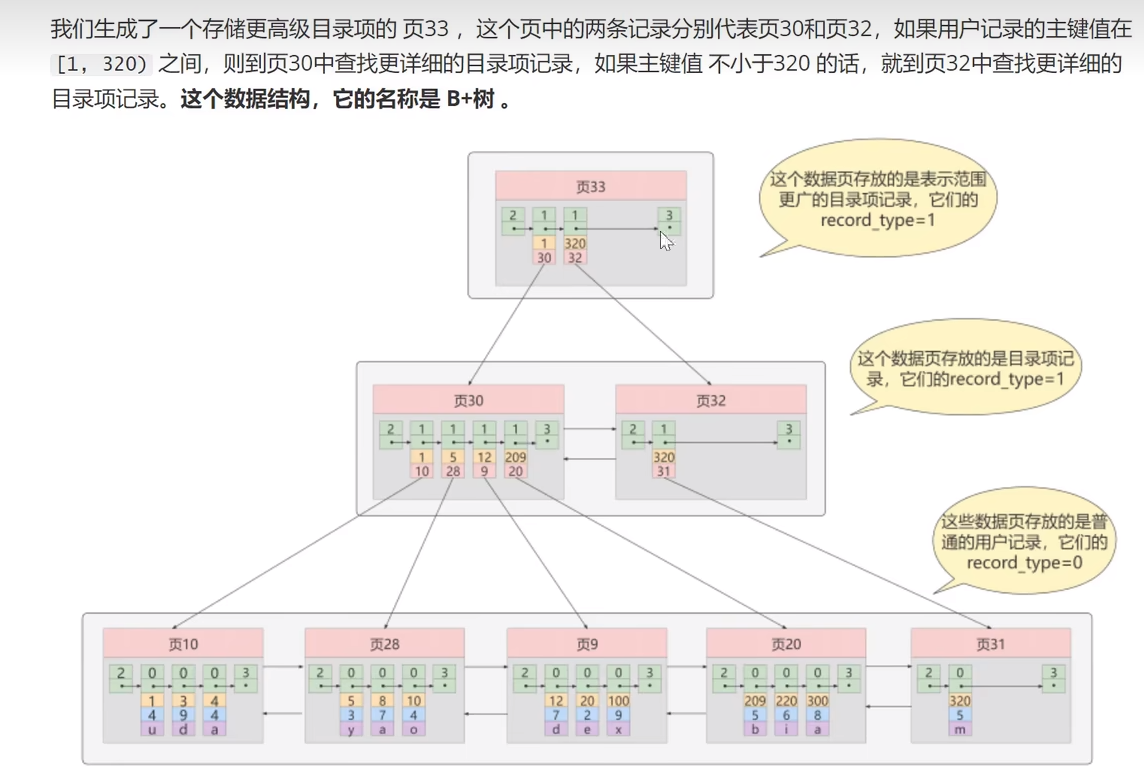
InnoDB分聚簇索引和非聚簇索引，如果是聚簇索引，则叶子节点存放的是数据本身，如果是非聚簇索引，则指向一个id，查询数据需要进行回表操作。

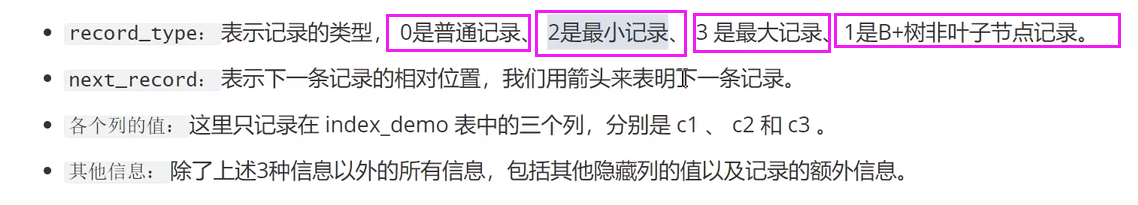
# P3 003.—个表中如果没有创建索引，那么还会去创建B+树吗？

创建表没有创建索引，有主键就创建聚簇索引，就会有B+树，没有主键，mysq会生成一个rowId,rowId不会重复，是隐式主键，也会有B+树。

# P4 004.说—下B+树索引实现原理〔数据结构)

页的大小是16K。



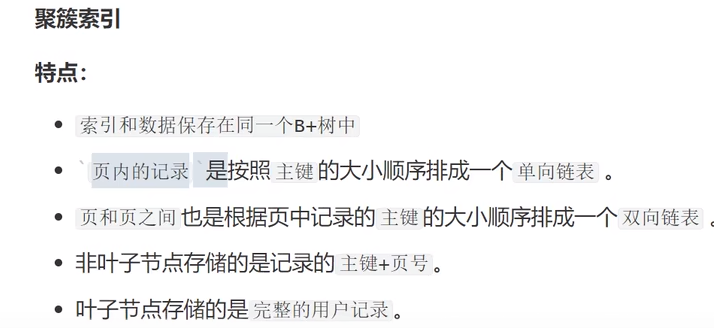


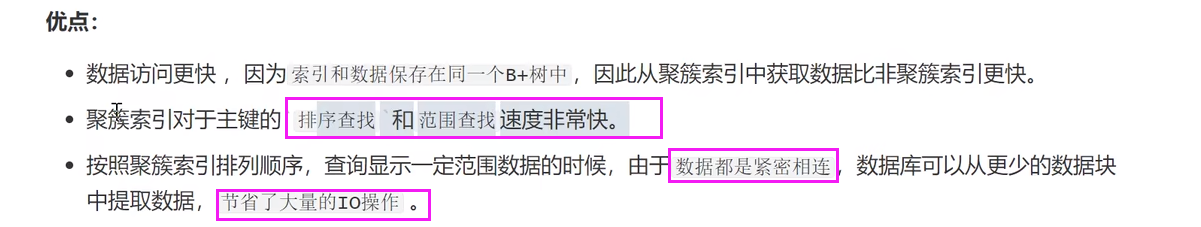
数据库的磁盘放到一个独立的磁盘上，不受其它程序干扰。顺序写，顺序读是最快的。

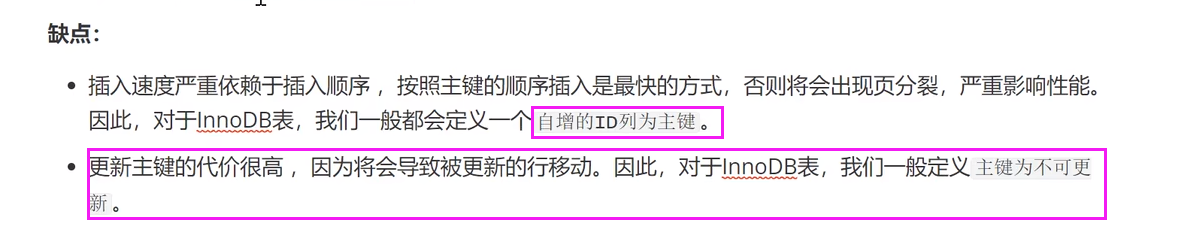
数据库的页最好是在物理磁盘上连续的。

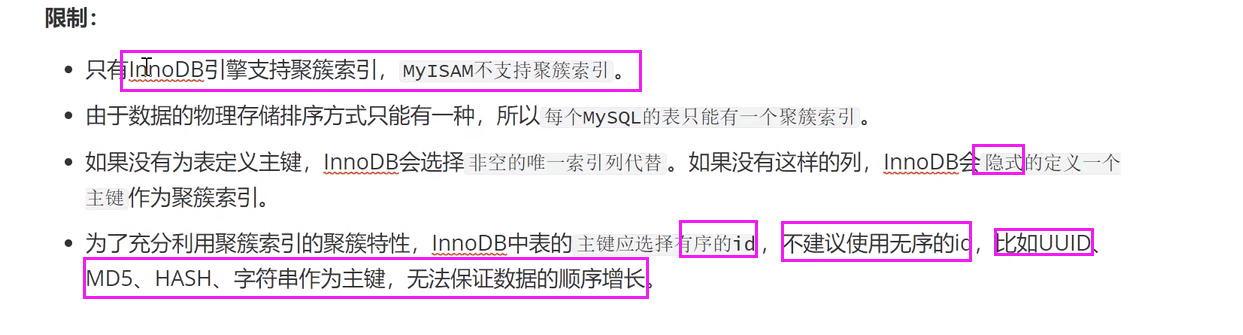
# P5 005.聚簇索引与非聚簇索引b+树实现有什区别

## 聚簇索引

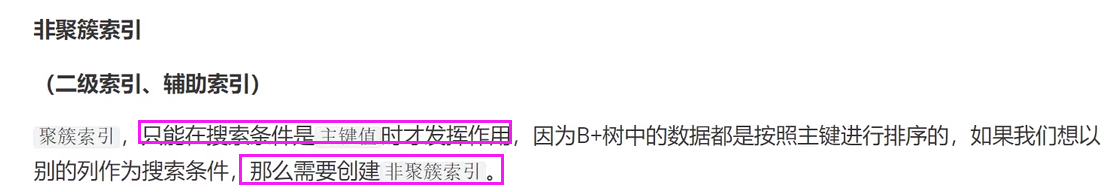


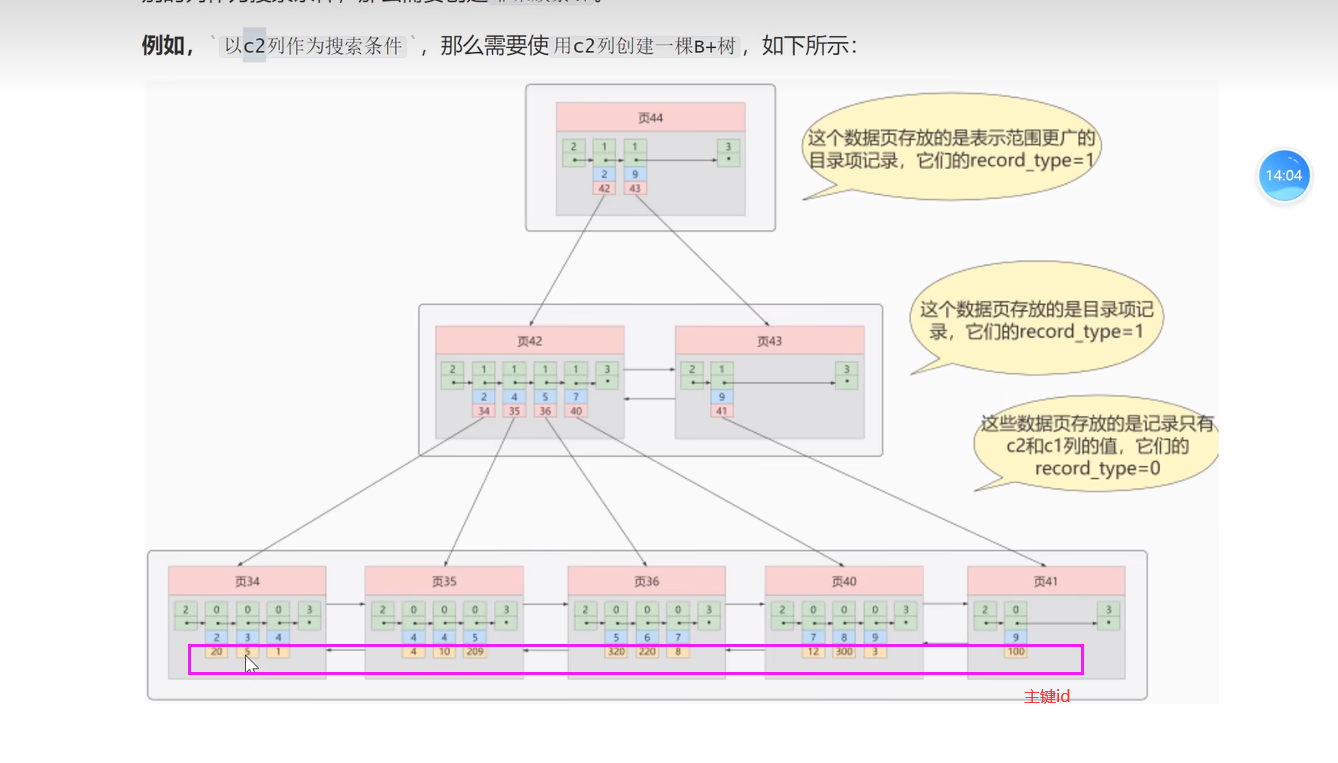


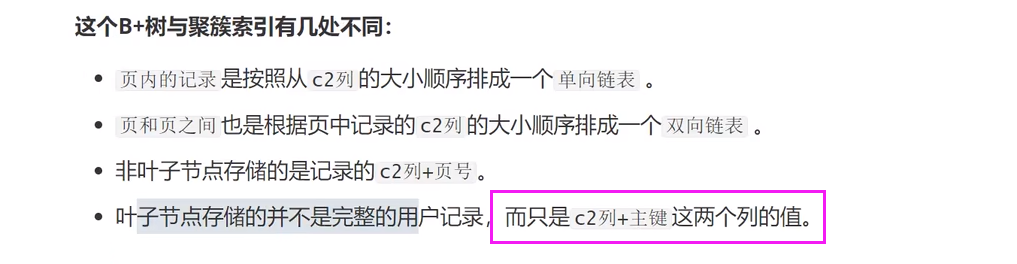




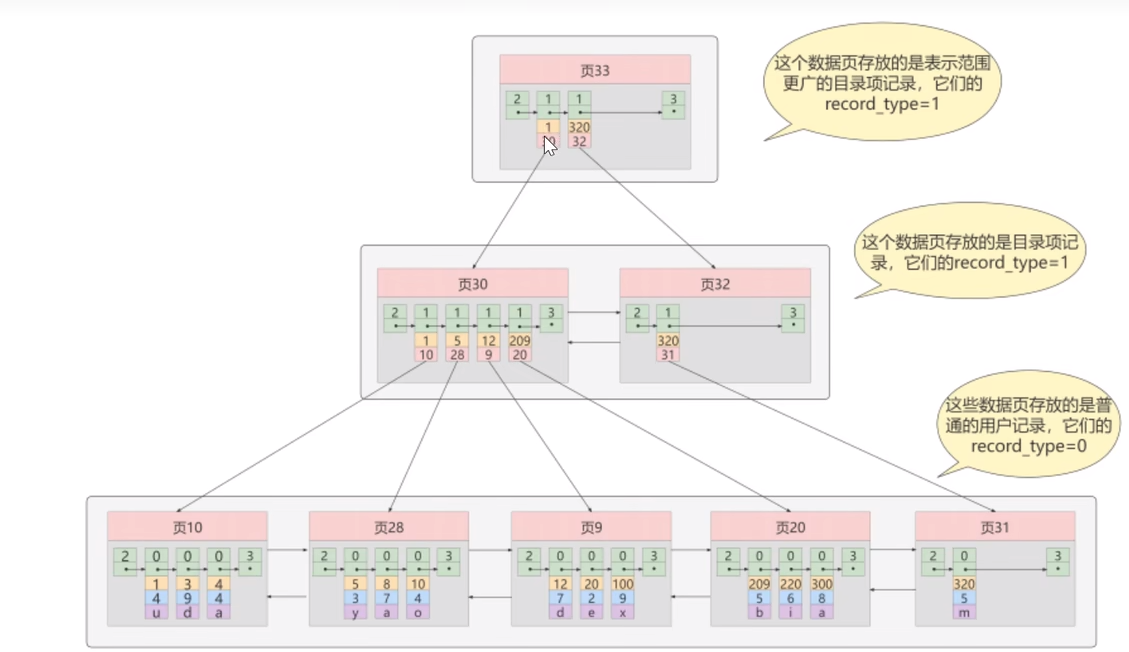
## 非聚簇索引





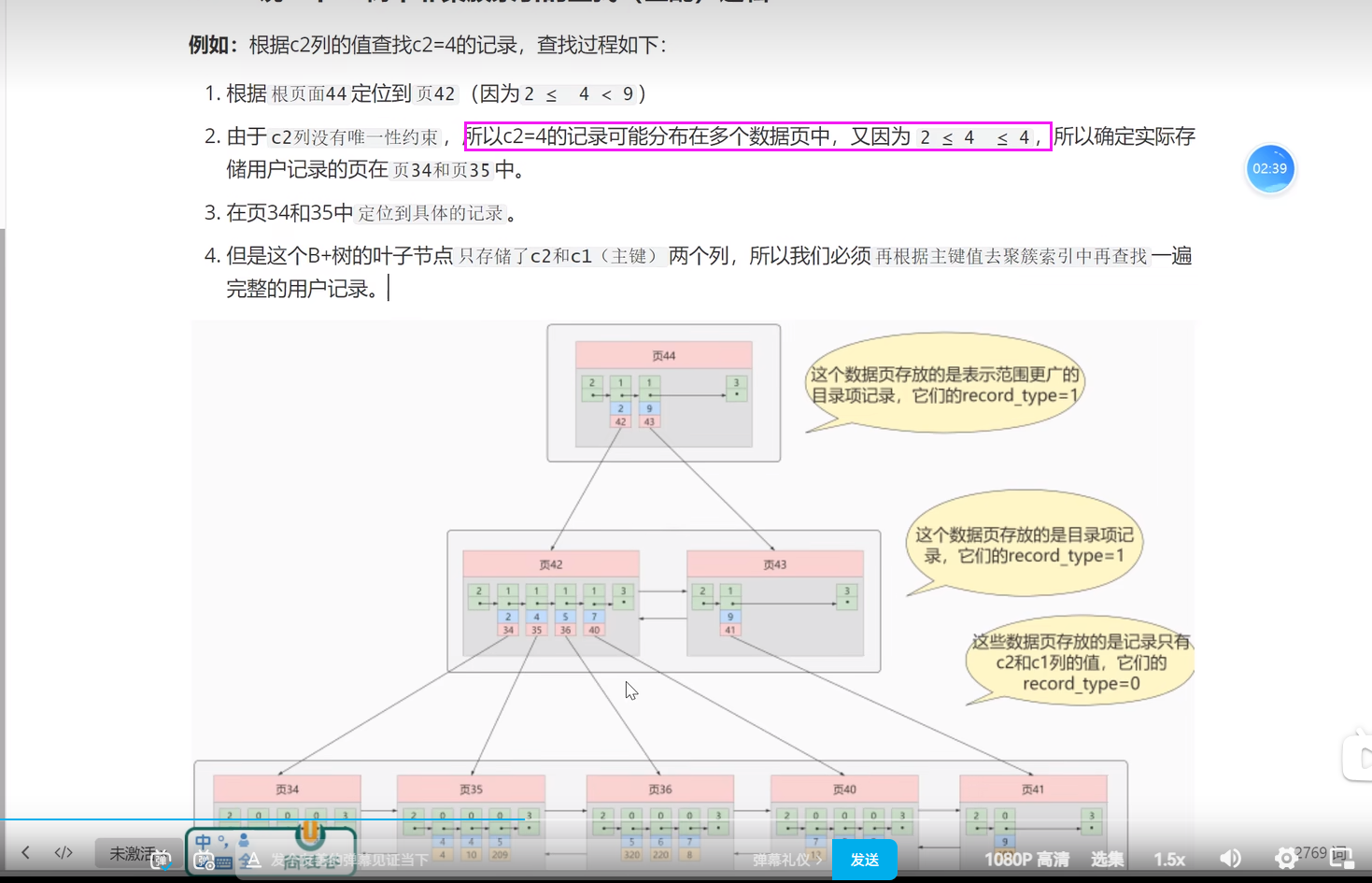


# P6 006.说—下B+树中聚簇索引的查找（匹配逻辑



# P7 007.说—下B+树中非聚簇索引的查找（匹配逻辑）

非聚簇索引，如果是字符串怎么比较大小，会转成ask码来比较大小。



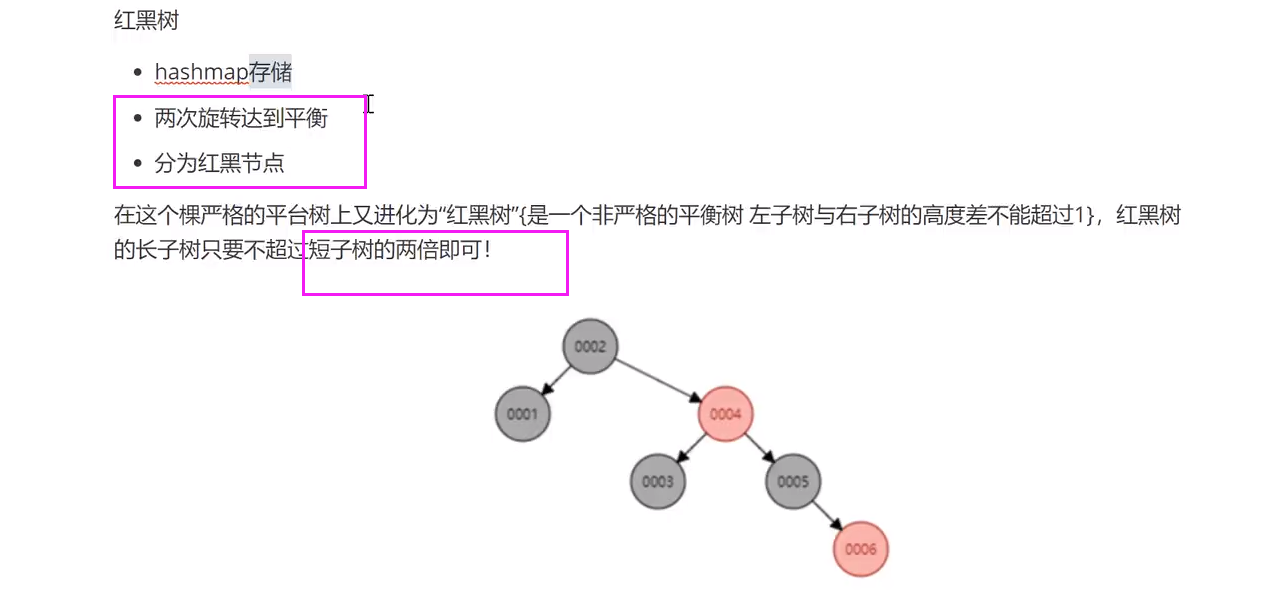
# P8 008.平衡二叉树，红黑树，B树和B+树的区别，都有哪些应用场景

## 平衡二叉树

自身节点只能存一份数据。



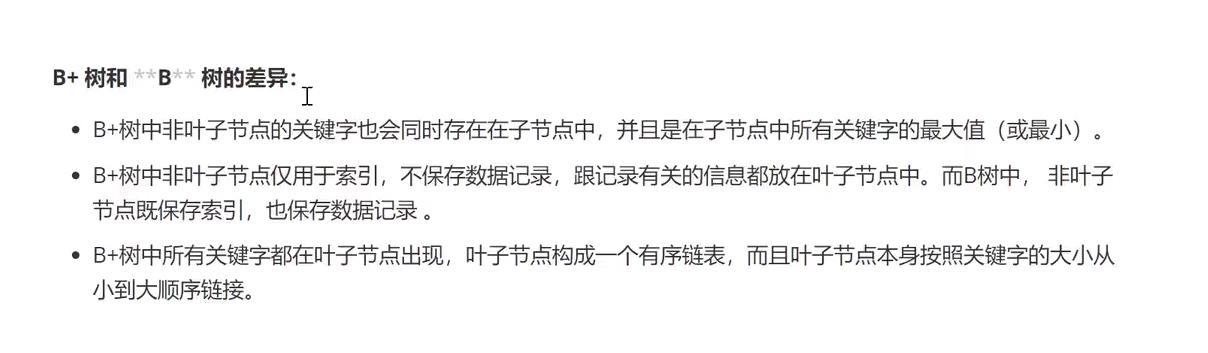
## 红黑树



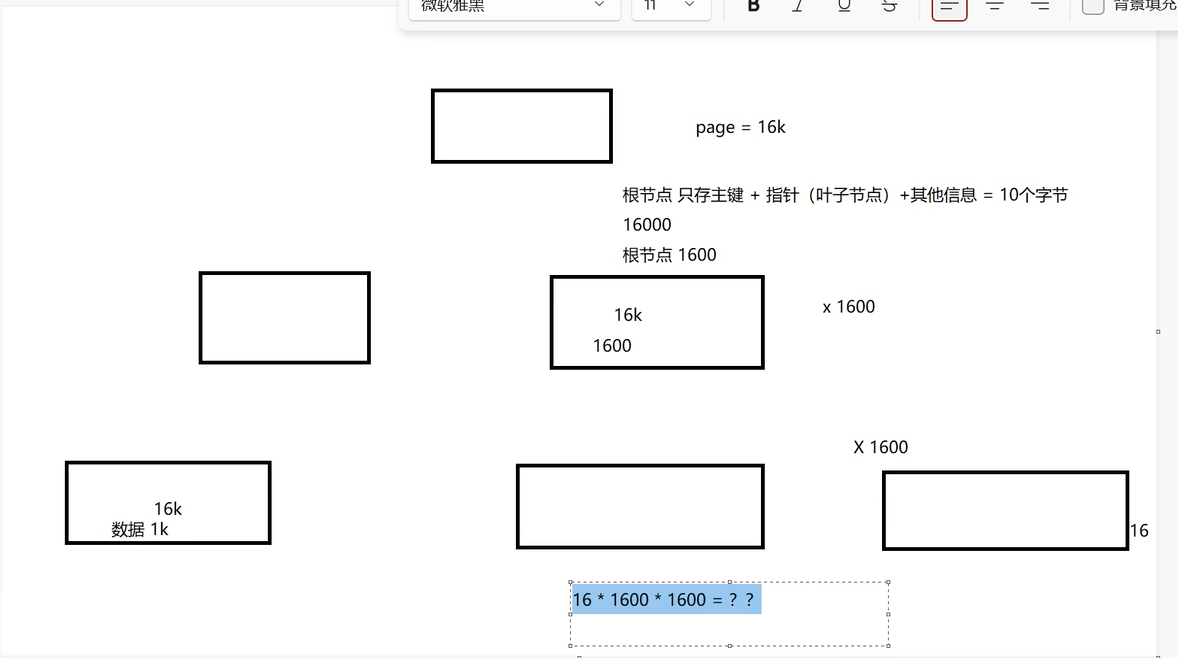
### 红黑树的应用场景？

可用于存储有序数据、增删频发的场景。比如:Java集合中的TreeSet和TreeMap；JDK中HashMap，当链表长度大于8时会转化成红黑树；C++STL中的set、map，以及Linux虚拟内存的管理。

## B+树和B树



# P9 009.—个b+树中大概能存放多少条索引记



# P10 010.使用B+树存储的索引crud执行效率如何?

三层节点的B+树，除非数据量特别大编程四层，才会发生自旋。

新增，删除容易引起页的变化，所以麻烦点。更新和读取非常方便。



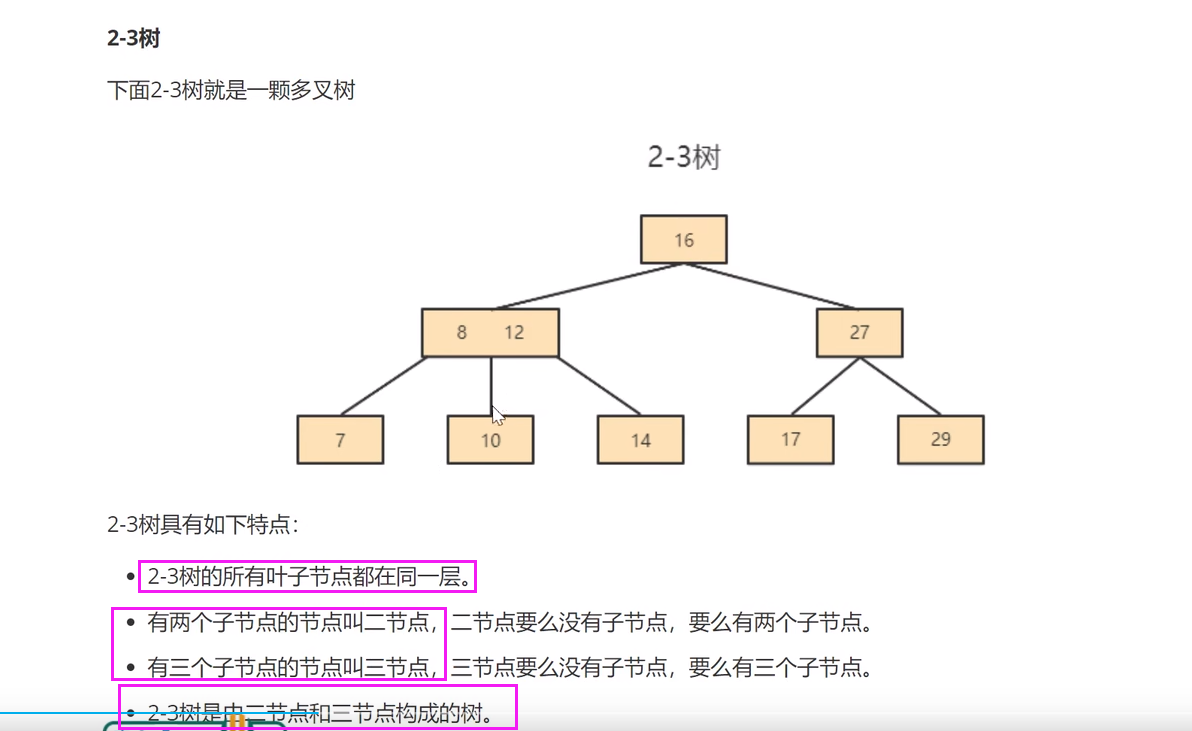
# P11 011.什么是自适应哈希索引?

自动生成，无法干预。

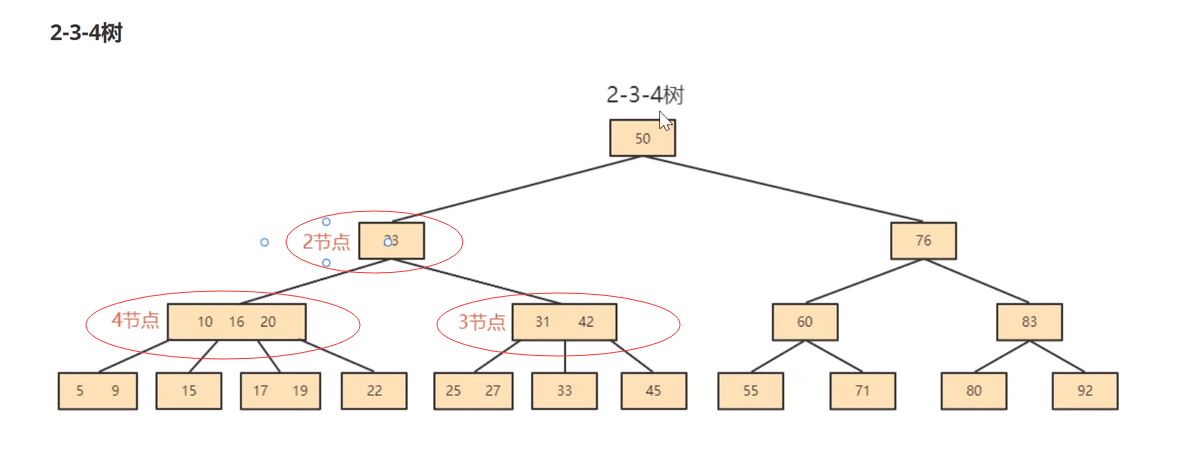


# P12 012.什么是2-3树2-3-4树?

## 2-3树

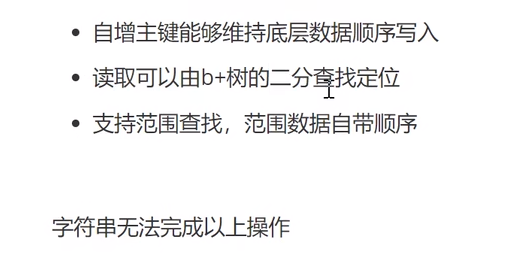


## 2-3-3树

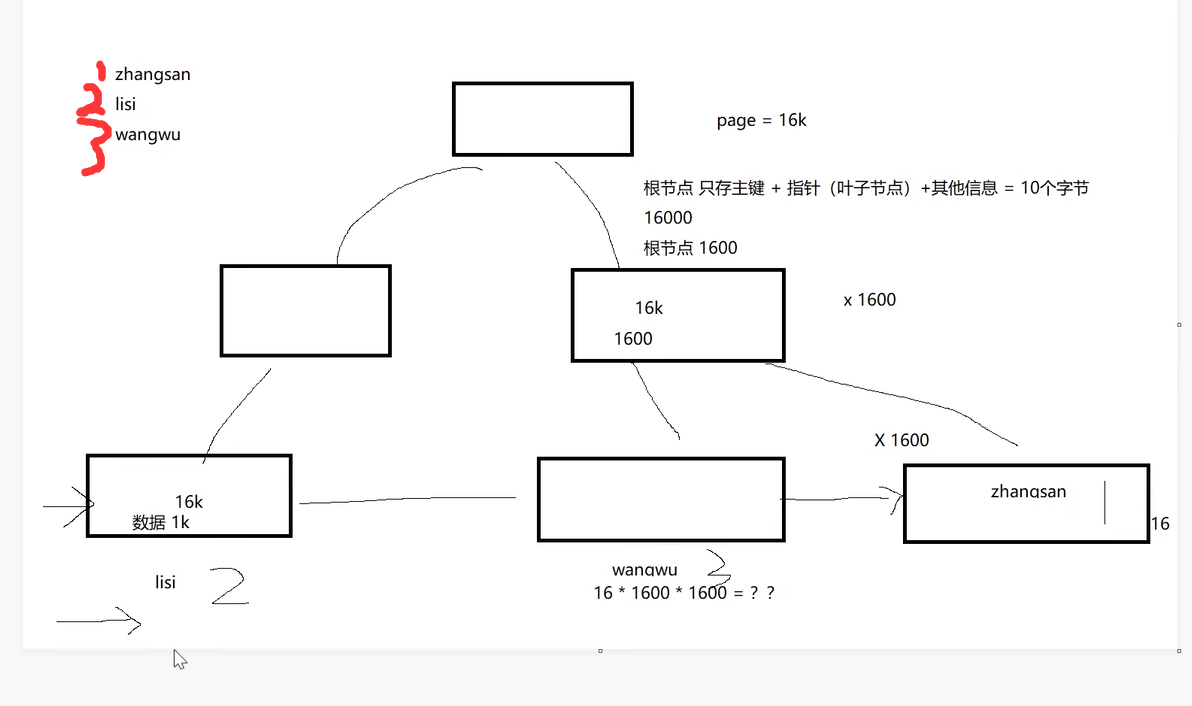


# P13 013.说一下自增主键和字符串类型主键的区别和影响

字符串要转ask码来匹配

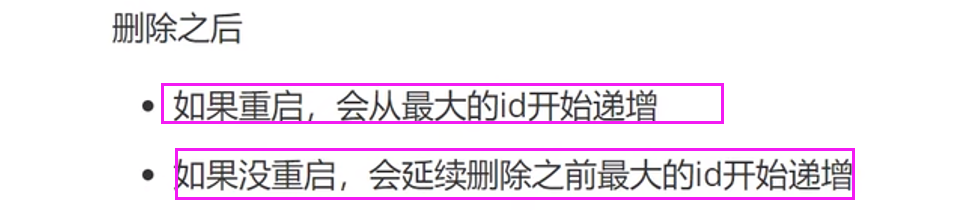


## 字符串类型主键



# P14 014.使用int自增主键后最大id是10，删除id 10和9，再添加一条记录，最后添加的id是几。删除后重启mysql然后添加一条记录最后id是几

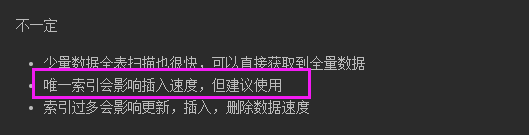
重启mysql后，会重置id的序号



# P15 015.索引的优缺点是什么?



# P16 016.使用索引一定能提升效率吗?



# P17 017.如果是大段文本内容，如何创建（优化)索

第一种方式是分表存储，然后创建索引（大文本需要指定索引长度，开头必须精确，后面可以模糊匹配）

第二是使用es为大文本创建索引（推荐）

# P18 018.什么是聚簇索引?

聚簇索引数据和索引存放在一起组成一个b+树

# P19 019.一个表中可以有多个（非)聚簇索引吗?

聚簇索引只能有一个

非聚簇索引可以有多个

# P20 020.聚簇索引与非聚集索引的特点是什么?

前面讲过

# P21 021.CRUD时聚簇索引与非聚簇索引的区别是什么

聚簇索引插入新值比采用非聚簇索引插入新值的速度要慢很多，因为插入要保证主键不能重复

聚簇索引范围，排序查找效率高，因为是有序的

非聚簇索引访问需要两次索引查找，第一次找到主键值，第二次根据主键值找到行数据

# P22 022.非聚簇索引为什么不存数据地址值而存储主键

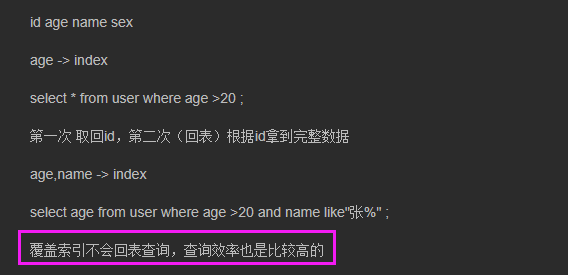
因为聚簇索引中有时会引发分页操作、重排操作数据有可能会移动

# P23 023.什么是回表操作?

查完id后查所有列

# P24 024.什么是覆盖索引?

查询的列在索引中存在



# P25 025.非聚集索引一定回表查询吗

不一定，只要b+树中包含的字段（创建索引的字段），覆盖（包含）想要select 的字段，那么就不会回表查询了。

# P26 026.为什么要回表查询?直接存储数据不可以吗?

为了控制非聚簇索引的大小

# P27 027.如果把一个InnoDB表的主键删掉，是不是就没有主键，就没办法进行回表查询了。

不是，InnoDB会生成rowid辅助回表查询

# P28 028.什么是联合索引，组合索引，复合索引?

就是一个东西，名字不已一样而已。

全职匹配

最左前缀

# P29 029.复合索引创建时字段顺序不一样使用效果一样吗？.

不一样，先按照第一个字段排序。

# P30 030.什么是唯一索引?

# P31 031.唯一索引是否影响性能?

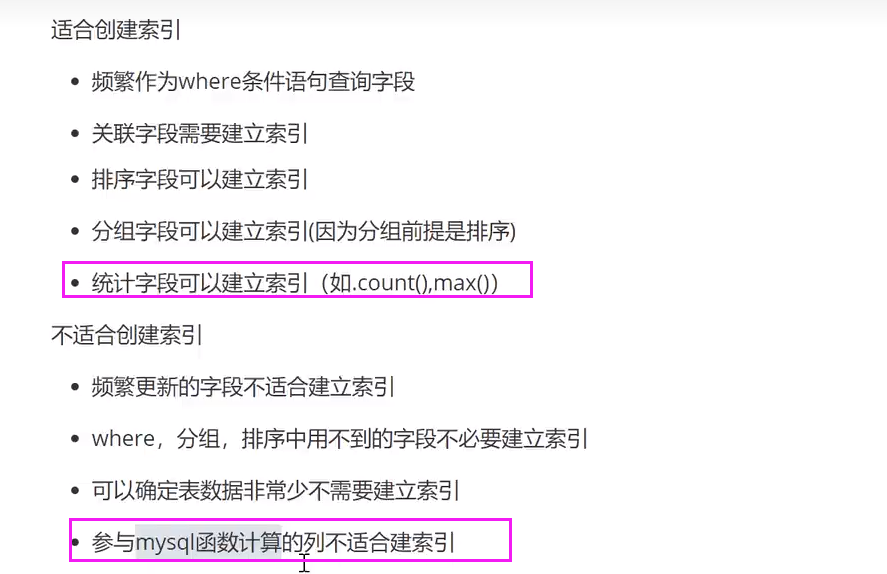
是。

# P32 032.什么时候使用唯一索引?

业务需求唯一字段的时候，一般不考虑性能问题

【强制】业务上具有唯一特性的字段，即使是多个字段的组合，也必须建成唯一索引。 说明：不要以为唯一索引影响了 insert 速度，这个速度损耗可以忽略，但提高查找速度是明 显的；另外，即使在应用层做了非常完善的校验控制，只要没有唯一索引，根据墨菲定律，必 然有脏数据产生。

# P33 033.什么时候适合创建索引，什么时候不适合创建索引



# P34 034.什么是索引下推?

5.6之前的版本是没有索引下推这个优化的

\*\*Using index condition：\*\*叫作 Index Condition Pushdown Optimization （索引下推优化）

如果没有索引下推（ICP），那么MySQL在存储引擎层找到满足content1 > 'z'条件的第一条二级索引记录。主键值进行回表，返回完整的记录给server层，server层再判断其他的搜索条件是否成立。如果成立则保留该记录，否则跳过该记录，然后向存储引擎层要下一条记录。

如果使用了索引下推（ICP），那么MySQL在存储引擎层找到满足content1 > 'z'条件的第一条二级索引记录。不着急执行回表，而是在这条记录上先判断一下所有关于idx\_content1索引中包含的条件是否成立，也就是content1 > 'z' AND content1 LIKE '%a'是否成立。如果这些条件不成立，则直接跳过该二级索引记录，去找下一条二级索引记录；如果这些条件成立，则执行回表操作，返回完整的记录给server层。

总结：

未开启索引下推：

根据筛选条件在索引树中筛选第一个条件

获得结果集后回表操作

进行其他条件筛选

再次回表查询

开启索引下推：在条件查询时，当前索引树如果满足全部筛选条件，可以在当前树中完成全部筛选过滤，得到比较小的结果集再进行回表操作

# P35 035.有哪些情况会导致索引失效?