2-3树，一种多路查找树

前驱节点，指非叶子节点的左子节点的最右叶子子树

后继节点，指非叶子节点的右子节点的最左叶子子树

2节点包含一个元素两个孩子节点或者没有孩子节点，所有叶子节点都在同一个层次 ，节点数等于孩子节点指针数，元素数=孩子节点数-1；

B数，高度平衡的多路查找树，我们把节点最大的指针数目称之为B树的阶通常记为m

根节点不为终端节点（非叶子节点），至少有两棵子树；

除了根节点外的所有非叶子节点，至少有m/2棵子树（向上取整）；

非叶子节点结构：开头n记录节点中有多少个关键字，k1到kn主键增大

b树的插入；

元素小于m-1直接插入，元素等于m-1就分裂，分裂找中间关键字，n/2向上取整，以中间关键字向上插入到指针位置，左右关键字成为该节点的左右孩子节点，

b树的删除：

终端节点：自己够用自己，自己不够借兄弟，兄弟不够兄父合并；

非终端节点：转换为终端节点，找做孩子最右叶子节点或右孩子最左节点，节点元素替换后删叶子

b+树：

与b树不同点，每个关键字对应一个子树

每个节点关键字个数范围是[m/2]<=n<=m；

非叶子节点只保存关键字，叶子节点保存完整的数据记录

b+树有一个指针指向关键字最小的叶子节点，所有叶子节点链接成为一个单链表

MYSQL:

计算机的局部性原理：取一个数据的时候会从磁盘多取相邻部分的数据进入内存，方便下一次读取；

InnoDB页格式：页时InnoDB管理存储数据的基本单位，一页默认大小16kb；

File Header文件头部

Page Header页面头部

User Records用户记录

Free Space空闲空间（和用户记录为一体）

Page Directory

InnoDB行格式：

变长字段列表

用于记录varchar或其他变长字段的长度，它会和表中列表数据的顺序排列相同

NULL标志位

000100，记录行中每个字段的值是否为null，为null的值就直接返回null，不为null去行中对应字段查找其值；

//注意，如果在建表时，规定每个字段均不为null，和字段均非变长，前两个字段可以去掉

记录头信息

next\_record,表示下一条记录的相对位置

record\_type

//具体信息可以去网页首相夹内寻找

列表数据1,2,3.......

row\_id

默认id，当建表语句没有指定id时，InnoDB存储引擎会默认设置一个自增id；

transaction\_idl是6字节的事务id

roll\_pointertl是7字节回滚指针

索引：

聚集，自动排序，数据和索引是合在一起的

堆表，按插入顺序排序

InoDB的事务：

MVCC多版本并发控制

读未提交：就一条数据，大家都可以读和改

读已提交：

引擎会根据该条记录的事务版本链进行查找，一个事务可以去遍历版本链，找到最新一条版本号和自身版本号相同的链表

视图ReadView会在查询时生成，里面属性有m\_ids[]保存当前有哪些事务是活跃的还未提交，遍历版本链中的时候会判断版本id是否存在于活跃事务id中

|>如果存在就继续遍历，不存在就取出该版本的数据。版本操作提交后，对应的版本链节点会被被移到前面。回滚就是把与该事务版本号相同的版本链节点删除

可重复读：

该隔离级别和读已提交原理相似，只不过第一次查找获取的m\_ids[]也就是活跃版本id ，即使已经提交单在第二次获取时也不会删除，而是直接使用之前获取过得m\_ids[]；

InoDB的写机制：

锁会在事务结束之后或者回滚的时候释放掉。

读锁：

其他人不能更新这条数据，但是可以读这条数据也可以加读锁，一条数据可以被多个事务加读锁。

select \* from table lock in share mode;

写锁：

其他人不能更新数据也不能对数据加锁，其他事务不能在加锁。

select \* from table for update;

//注意，单纯的select语句没有加锁的代码快，查询语句不受锁影响。

行锁：

单个行上的锁：锁定一行数据

间隙锁：锁定一个间隙

间隙锁（传统）：锁定行和间隙

读已提交：

a事务写锁查询时，会锁行。b事务的写锁查询可以查询到a未查询的数据，也可以插入；

可重复读：

a事务写锁查询时，会锁间隙。b事务不能插入到a已锁定的行的间隙中。