Chapter 10 文件，外部排序与搜索

文件组织

逻辑文件：字节流或字符流

物理文件：磁盘上存储的文件

数据库文件：具有结构的记录集合或序列，各记录间是线性关系，每个记录由若干数据项组成，记录是文件存取的基本单位，数据项是文件可用的最小单位。能够唯一标识记录的数据项称为**主键**

文件的检索分为：

- 顺序存取
- 直接存取
- 关键码存取

文件的存储结构

- 顺序文件：实际顺序与进入文件的顺序一致，适合顺序存取，直接存取和关键码存取的效率很低。修改时采用批处理的操作实现
- 散列文件：散列技术组织的文件，类似散列表。不能顺序存取
- 索引文件：由索引表和数据表组成，索引表记录主键和存储位置。可分为稠密索引和稀疏索引
    - 多重表：为次关键码建立次索引，具有同一次关键码的记录构成链表
    - 倒排表：次关键码的索引中存储主键

多级索引结构

动态 m 路搜索树

1. 根结点最多有 m 棵子树，且有结构：$n, P_0, (K_1, P_1), (K_2, P_2), \ldots, (K_n, P_n)$，其中 P_i 是指向子树的指针，$0 \leqslant i \leqslant n < m$ ，K_i 是关键码，$0 \leqslant i \leqslant n < m$
2. $K_i < K_{i+1}, 1 \leqslant i < n$
3. 在子树 P_i 中所有关键码都小于 K_{i+1} ，且大于 K_i ，$1 \leqslant i < n$
4. 在子树 P_n 中所有关键码都大于 K_n ，在子树 P_0 中所有关键码都小于 K_1

5. 子树也是 m 路搜索树

B 树

平衡的 m 路搜索树，满足以下性质

- 根结点至少两个子女
- 除根结点以外所有结点（不包括失败结点）至少 $\lceil m/2 \rceil$ 个子女
- 所有失败结点位于同层

B 树高度 h 与关键码个数 N 与阶数 m 的关系：

- $h \leqslant log_{\lceil m/2 \rceil}(\frac{N+1}{2}) + 1$
- $N \geqslant 2\lceil m/2 \rceil^{h-1} - 1$

B 树的插入

每个结点的关键码范围都是 $[\lceil m/2 \rceil - 1, m-1]$，当超出范围时必须要分裂结点

设结点 p 中已经有 $m-1$ 个关键码，必须把结点分裂为两个结点：

- 结点 p：$(\lceil m/2 \rceil - 1, P_0, K_1, P_1, \ldots, K_{\lceil m/2 \rceil - 1}, P_{\lceil m/2 \rceil - 1})$
- 结点 q：$(m - \lceil m/2 \rceil, P_{\lceil m/2 \rceil}, K_{\lceil m/2 \rceil + 1}, P_{\lceil m/2 \rceil + 1}, \ldots, K_m, P_m)$

位于中间的关键码 $K_{\lceil m/2 \rceil}$ 和新结点 q 形成新的二元组 $(K_{\lceil m/2 \rceil}, q)$ 插入 p 的父结点

m 较大时访问磁盘的次数平均为 $h+1$

B 树的删除

若删除的结点不是叶结点，被删除的关键码为 $K_i, 1 \leqslant i \leqslant n$，则在删去关键码后，用子树 P_i 所指子树中最小关键码 x 代替被删除关键码，然后在 x 所在结点删除 x，这样最终就转化为在叶结点删除关键码

若被删关键码所在叶结点同时也是根结点且删去前关键码个数 $n \geqslant 2$ 则直接删去该关键码

若被删关键码所在叶结点不是根结点且删除前关键码个数 $n \geqslant \lceil m/2 \rceil$ 则直接删去该关键码

若被删关键码所在叶结点删除前关键码个数 $n = \lceil m/2 \rceil - 1$，若这时与该结点相邻的右兄弟（没有则左兄弟）结点关键码个数 $n \geqslant \lceil m/2 \rceil$，则按以下步骤调整该结点、右兄弟（左兄弟）结点及其父结点

1. 将其父结点中刚好大于（小于）该被删关键码的关键码 $K_i (1 \leqslant i \leqslant n)$ 下移到被删关键码所在结点
2. 将右兄弟（或左兄弟）结点中最小（或最大）关键码上移到父结点的 K_i 位置

3. 将右兄弟（或左兄弟）结点中的最左（或最右）子树指针平移到被删关键码所在结点中最后（或最前）子树指针位置
4. 在右兄弟（或左兄弟）结点中，将被移走的关键码和指针位置用剩余的关键码和指针填补调整，并将结点中关键码个数减1

若被删关键码所在叶结点删除前关键码个数 $n = \lceil m/2 \rceil - 1$，若这时与该结点相邻的右兄弟（没有则左兄弟）结点关键码个数 $n = \lceil m/2 \rceil - 1$，则合并这两个结点

1. 将父结点 p 中相应关键码下移到选定保留的结点中，若要合并 p 的子树 P_i, P_{i+1} 且保留 P_i，则将关键码 K_{i+1} 下降到 P_i 所指结点
2. 把 p 中子树指针 P_{i+1} 所指结点全部关键码和指针搬到 P_i 所指结点后，删去 P_{i+1} 所指结点
3. 在父结点 p 中用后面剩余关键码和指针填补关键码 K_{i+1} 和指针 P_{i+1}
4. 修改父结点 p 和所选保留结点的关键码个数

合并结点的过程中，父结点关键码减少，若父结点是根结点且关键码减少到 0，则父结点删去，合并后的结点成为根结点，否则若父结点关键码个数减到 $n = \lceil m/2 \rceil - 2$ 则又要与兄弟结点合并，重复合并，最坏情况下合并会回溯到根结点

B+ 树

B+ 树与 B 树的不同在于

- 所有关键码都存放在叶结点，非叶结点的关键码是其子树中最小/最大关键码的复写
- 叶结点包含了全部关键码及指向相应数据存放记录的指针，且叶结点本身按关键码升序连接

按最大关键码复写原则定义 m 阶 B+ 树：

1. 每个结点最多 m 棵子树
2. 根结点最少有 1 颗子树，其他结点最少有 $\lceil m/2 \rceil$ 颗子树
3. 所有叶结点在同层，按从小到大顺序存放全部关键码，叶结点顺序连接
4. 有 n 个子树的结点有 n 个关键码
5. 所有非叶结点可以看做叶结点的索引，结点中关键码 K_i 与指针 P_i 构成索引项，K_i 是子树中最大的关键码

通常 B+ 树有两个头指针，指向根结点和指向最小的叶结点，可以顺序访问或随机访问

插入：仅在叶结点进行，当插入后叶结点关键码个数 $n > m$ 时，需要将叶结点分裂为两个结点，包含关键码个数分别为 $\lceil \frac{m+1}{2} \rceil$ 和 $\lfloor \frac{m+1}{2} \rfloor$，且父结点中应当同时包含这两个结点的最大关键码和结点地址，此后问题变为在非叶结点中插入。分裂至根结点后没有父结点需要创建新的父结点作为根，树的高度就增加了

简单删除：删除后叶结点关键码个数 $n \geqslant \lceil m/2 \rceil$ ，此时上层索引项不用改变

合并删除：删除后叶结点关键码个数 $n < \lceil m/2 \rceil$ ，则看其右结点的关键码个数决定是移动一个关键码还是合并结点，此处同 B 树。删除后要修改上层索引项