

2.4.2 考虑一个磁盘：它有 5 个双面盘片，每个盘面 2,000 个磁道，每个磁道 50 个扇区，每个扇区 512 字节。另外，假使它的平均寻道时间为 10ms。

计算每个盘面的格式化和整个磁盘的格式化的时间。如果磁盘转速为 5,400 转/分，计算磁盘的最大旋转延迟和平均旋转延迟时间。

在 256、2048 和 51,200 三个值中，那些值可能是有效的块大小？为什么？如果每个磁盘大小是 2 个扇区，试估算传输一个块的平均时间。

(1) 每个盘面的格式化容量 = 磁道数 \* 扇区数 \* 字节数 = 2000\*50\*512 B = 49M 整个磁盘的格式化容量 = 盘面数 \* 每个盘面的容量 = 10\*49M = 490M (2) 最大的旋转延迟时间 = 磁盘旋转一周所需的时间 / 1/转速 = 60/5400 = 0.011s 平均旋转等待时间 = 最大的旋转延迟时间/2 = 0.011s/2 = 0.0055s (3) 块是 DBMS 与 OS 进行读写磁盘的基本单位，必须用足够大的整数表示；其次，块大小选择要适中，太小会导致 I/O 次数增加，太大则会造成磁盘读写操作代价加大，得不偿失。因此，三个值中，只有 2048 可能是有效块大小。(4) 旋转传输 1 个块的时间 = 该两个扇区所需的时间 = (60/5400) \* (2/50) = 0.44 ms

传输一个块的时间 = 寻道时间 + 旋转延迟时间 + 传输时间 = 10ms + 5.5ms + 0.5ms = 16ms

习题 4.3 对习题 4.2 中磁盘，存储磁盘大小为 1,024 字节。假设每个页面包含 100,000 个元组。记录 100 个字节的关系文件存储在磁盘上，并假设记录不允许跨页存储。

每个块可存放多少元组？存储整个文件需要多少个块？估算存储该关系文件需要的时间。

(1) 每个块可存放多少元组 = 磁盘块大小/每个元组字节数 = 1024/100 = 10 个完整文件需要磁盘块的总数/每个块元组数 = 100,000/10 = 10,000 块 (2) 顺序扫描文件需总时间 = 文件总存储块数 \* 每块存取时间 ≈ 10,000 \* 16ms = 160,000ms = 27 分钟 (3) 根据题意，可从文件中找到一个元组的时间 = 最大的旋转延迟时间 = 0.011s 一个柱面大小 = 盘面数 [10] \* 扇区数 \* 字节数 = 10\*50\*512 B 按柱面交叉链接存储文件需要的页数 = 100,000/100/(10\*50\*512) = 40 (向上取整)。在这种情况下，顺序扫描文件需总时间 = 40\*0.011s ≈ 0.5s

习题 4.4 假设你们使用 RAID4 级方案，有 4 个数据盘和一个冗余盘。假使发生以下情况，如果数据盘中的某块值如下，试写出冗余盘的块值。

01010110, 11000000,00111011 和 11111111

01010000, 11111000, 00111111 和 00000001

习题 4.9 采用有 7 个 01101110 的 RAID6 级方案，描述从该数据盘中恢复所需采取的步骤。

(1) 首先 2#、3#、5# 号盘恢复 1# 的数据，再用 1#、3#、4# 号盘恢复 7# 的数据。(2) 先用 2#、3#、5# 号盘恢复 4# 的数据，再用 1#、2#、6# 号盘恢复 4# 的数据。

习题 5.2 考虑图 5.21 所示的阶数  $m=4$  的 B+ 树索引。注：图 5.21 显示了两个  $m=4$  的 B+ 树和多个叶节点。

标识键 A 数据项 9\* 之上的 B+ 树，并指出完成该插入需要多少个页面的重排。

给出在删除中删除数据项 8\* 之后的 B+ 树，并指出删除操作需要多少个页面的重排。

给出在删除中删除数据项 46\*，然后重新删除数据项 52\* 之后的 B+ 树。

给出在删除中，依次删除数据项 32\*、39\*、41\*、45\* 和 73\* 之后的 B+ 树。

图 5.21：原始 B+ 树结构 ( $m=4$ )

(1) 由图看出插入 9\* 不需要页分裂。直接插入即可。由于索引项数据文件本身有 9 个页，所以有 9 个叶结点经过，所以为 9 个页。

(2) 删除 8\* 后紧跟前一个索引项数据文件本身有 8 个页，删除后紧跟前一个索引项数据文件本身有 8 个页，删除后紧跟前一个索引项数据文件本身有 8 个页。

(3) 删除 46\* 后紧跟前一个索引项数据文件本身有 46 个页，删除后紧跟前一个索引项数据文件本身有 46 个页。

(4) 依次删除 32\*、39\*、41\*、45\*、73\* 后的图：

习题 5.3 考虑图 5.22 所示 B+ 树索引。内节点可容纳 4 个键值和 6 个指针；叶节点可直接存储数据记录。4 个键值 4 条记录且相邻叶节点之间用双链接在一起。假设以下情况：

指出回答查询“取数据项值大于 38 的所有记录”(4) 时，需要读取的叶节点。

给出插入 100\* 后的 B+ 树。

给出从原树中删除 81\* 后的 B+ 树。

指出在删除时删除数据项 81\* 后 A、B、C，你能测出这些子树的内容和形状吗？

图 5.22：原始 B+ 树结构

(1) 查大于 38\* 的所有

记录，再读取的节点有 11,12,13,2,3,4,5,15,16,17,18 (2) 插入 100\* 后，原 LSN 节点需要分裂，完成操作后的布局图：

(3) 删除 81\* 后，L6,L7 两个节点合并。操作完成后的布局图如下：

(4) 插入插入 [65,79] 的搜索值时，都会先查 L5 节点，而 L2 也是满的，向上分裂到根结点，根结点满了，就会造成高度增加。

(5) 关于 L2 中 A、B、C，我们可推出以下儿件事：它们都是满的，因为 L2 满了，那么它的指针子树 (A)、L2、L3 为满节点 树高都是 1；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

子树 A 包含的数据肯定少于 10，子树 B 所包含的数据肯定在 10~20 之间，子树 C 所包含的数据在 20~30 之间；

扫描的代价更低，可能会超过 2500+100000，达到 1500+100000+100 次 I/Os (假设有 10 个元组大小 100 字节)。如果索引项是聚集的且记录在叶中直接存储数据，则使用索引项检索并直接读取叶中数据。(4) 利用 ename 上的聚集 B+ 树索引，可能代价为 2500，因为 ename 为主键，扫描 B+ 树索引表的 ( $\langle \text{ename}, \text{title} \rangle$ ) 页不需要进行排序扫描重复，因此，产生查询结果的总估计代价是 12500。代价远远低于简单排序扫描表的 (15000+30000) 次 I/Os。但聚集 B+ 树索引所有目标元组的代价可能达到 1,500+100=1600。

习题 4.4 考虑预算  $R \times B \times A$ ，已知：

关系 R 有 10,000 个元组，每页可存 10 个元组，其数据文件为简单链表文件；

关系 S 有 2,000 个元组，每页可存 10 个元组，B+ 索引的键，其数据文件为简单链表文件；

有 52 个可用缓存页。假设以下问题：

分析并估算以下查询的总代价，并指出扫描表和块存储循环扫描算法何时代价，实现相应算法需要的小缓存页数量分别是多少？

若采用排序-归并连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

若采用线性连接算法，则其代价和需要的小缓存页数量分别是多少？

基于页簇存储，代价 = 10 + 10 \* 100 = 1010 I/Os。结果关系元组数为 200，每个元组大小 100+500=800 字节，共有 200 个页。(1) 若只有 3 个缓存页，采用排序扫描算法结果关系。第一个阶段直接投影到结果关系，产生只有 450 字节元组 2 个元组/页，因此，扫描表 P3 需要 100 页 (33 个元组，每个元组大小为 1 页)。这些子表需要扫描 log<sub>3</sub> 36 = 4 个扫描阶段，因此，排序扫描投影的总代价 = 200+2\*6\*100=1400，加上连接的 1010 次 I/Os，整体代价为 2410 次 I/Os (2) 若只有 3 个缓存页，排序扫描算法结果关系。第一个阶段直接投影到结果关系，产生只有 450 字节元组 2 个元组/页，因此，扫描表 P3 需要 100 页，写出为 100 页 (分 10 个元组，每个元组大小为 10 页)，这些子表需要扫描 log<sub>10</sub> 10=4 个扫描阶段，因此，排序扫描投影的总代价 = 200+2\*4\*100=1000，加上连接的 1010 次 I/Os，整体代价为 2010 次 I/Os (3) “先连接后投影，且连接时通过流水线传输投影。”这意味着不计排序扫描算法直接投影的代价，整体的代价只有 1010 次 I/Os。

习题 7.5 考虑如下关系模式：Emp (dept: integer, sal: integer, age: real, did: integer); Dept (dept: real, status: c); integer, report: varchar); 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

SELECT E.d\_id,D.p\_id FROM Emp E,Dept D,Proj P WHERE E.d\_id=D.d\_id=D.p\_id AND B.budget>D.b\_id; 已知: Emp 元组数 20,000，每个元组 20 字节; Dept 元组数 5,000，每个元组 40 字节。每个 did 值平均有 10% 的元组; Proj 元组数 1,000，其平均行长度为 2,000 字节。另给定文件系统支持的元组大小为 4000 字节，有 10 个可用缓存页；另外，如果认为元组表，还可只考虑一些其他外部表。

若假定 budgets 值均分布在 0~100,000 范围内，试给出如下查询的总代价：

&lt;



