Forrás: Molina, Ullman, Widom, Adatbázisrendszerek megvalósítása. Bp. Panem, 2001.

A végrehajtás költségei

R sorainak száma: T(R)
R blokkjainak száma: B(R)
A memóriablokkok száma: M

Egymenetes algoritmusok: az adatokat csak egyszer kell lemezről beolvasni.

Kétmenetes algoritmusok: az adatokat az első alkalommal lemezről kell beolvasni, aztán következik valamilyen típusú feldolgozásuk, majd az összes - vagy majdnem az összes - adatot lemezre kell írni, és ekkor következik a második menetben a második olvasás a további feldolgozáshoz.

Többmenetes algoritmusok: tulajdonképpen a kétmenetes algoritmusok természetes, rekurzív általánosításai.

Soronkénti, unáris műveletek: σ, π
A blokkokat egyenként olvashatjuk be, egyetlen memóriapuffert használva, majd megadhatjuk a kimenetet.

Unáris, teljes relációs műveletek: γ, δ
Az ilyen egy argumentumos műveleteknél az összes sort (vagy legalábbis a sorok nagy részét) egyszerre kell a memóriában látnunk

Bináris, teljes relációs műveletek: az összes többi művelet

Egymenetes algoritmusok

$\sigma(R)$ és $\pi(R)$
R blokkjait egyenként olvassuk be a bemeneti pufferbe, a műveleteket minden soron elvégezzük, majd a kiválasztott vagy a vetített sorokat kivisszük a kimeneti pufferbe.
Feltétel: M >= 1
Költség: B(R)

$\delta(R)$
Egy memória pufferbe folyamatosan beolvassuk *R* blokkjait, míg a fennmaradó *M-1* puffert használhatjuk arra, hogy tartalmazzák a már előfordult minden egyes sor egy másolatát.
Feltétel: $\delta(R) <= M$
Költség: B(R)

$\gamma(R)$
Egy csoport *bejegyzése* a csoportosító attribútumok értékeiből és az egyes összesítések értékeiből képzett kumulált értékekből áll.
Feltétel: (!?)
A csoportok bejegyzése lehet *R* sorainál rövidebb vagy akár hosszabb is, a csoportok száma pedig bármi lehet, ami *R* sorainak számánál kisebb vagy azzal egyenlő. A legtöbb esetben azonban a csoport bejegyzések nem hosszabbak *R* sorainál, és a csoportok száma sokkal kisebb a sorokénál.
Költség: B(R)

Összekapcsolás
Olvassuk be a kisebbik relációt (S), majd a másikat blokkonként olvasva elvégezzük a megfelelő sorok összekapcsolását.
Feltétel: B(S) <= M-1 (vagy másképp: Min(B(R), B(S)) <= M
Költség: B(S) + B(R)

Beágyazott ciklusú összekapcsolás

Tegyük fel, hogy $B(S) \leq B(R)$, de $B(S) > M$.
Beolvassuk S-et M-1 blokkonként, és az éppen beolvasott részt összekapcsoljuk R-rel, a fennmaradó 1 blokkot használva.
A külső ciklus iterációinak száma $B(S)/(M-1)$. Minden iteráció során *S*-nek *M-1* blokkját és *R*-nek $B(R)$ számú blokkját olvassuk be.
Költség tehát: $B(S)/(M-1)$ * (M-1 + B(R)) ~> B(S)B(R)/M (feltéve, hogy B>>M)
Feltétel: Nincs, vagyis ez memóriaméréttől függetlenül mindig elvégezhető

Rendezésen alapuló kétmenetes algoritmusok

1. Beolvasunk M blokkot, majd rendezzük a memóriában.
2. A rendezett részlistákat kiírjuk lemezre.
3. Egy második beolvasási menettel összefésüljük az M darab rendezett részlistát és elvégezzük a kívánt műveletet.

(A műveletet akkor tudjuk elvégezni a második menetben, ha ehhez az azonos kulcsértékkel rendelkező soroknak kell egyszerre a memóriában lenniük. Így például egy equijoin elvégezhető, de egy „T1.o1<T2.o2” típusú összekapcsolás már nem.
Feltétel: B(R) <= M^2 (bináris műveletekre B(R) + B(S) <= M^2)
Költség: 3*B(R) (bináris műveletekre 3*(B(R)+B(S))

Ha az első menet után több mint M darab rendezett részlista lesz (vagyis B(R) > M^2), akkor ezeket nem tudjuk egy második menetben rendezni. Ilyenkor további menetekre van szükség. Minden újabb menet M-ed részére tudja csökkenteni a már rendezett részlisták számát, vagyis ilyen módon rekurzívan kapjuk a többmenetes algoritmusokat.

Tördelésen (hasításon) alapuló kétmenetes algoritmusok

1. A sorokat M (pontosabban M-1) kosárba tördeljük.
2. A kosarakat (R_i illetve S_i) kiírjuk lemezre.
3. A megfelelő kosarakra (illetve kosár-párokra) alkalmazzuk a megfelelő egymenetes algoritmust.

(Ebben az esetben is csak akkor tudjuk a második menetben elvégezni a műveletet, ha ehhez elég az azonos kosárbeli sorokat beolvasni.)
Feltétel: B(R) <= M^2 (bináris műveletekre Min(B(R), B(S)) <= M^2)
Költség: 3*B(R) (bináris műveletekre 3*(B(R)+B(S))

A rekurzivitás most is hasonló a rendezés esetéhez, mivel minden további menet M-ed részére tudja csökkenteni a kosarak méretét.

Rendezés/tördelés összehasonlítása
A tördeléses algoritmusok alkalmazhatósága csak a kisebbik reláció méretétől függ nem pedig a kettő összegétől, mivel a második menetben elég a kisebbik relációt (illetve a kisebbik kosarat) beolvasni, a rendezés esetén viszont mindkét reláció összes azonos értékkel rendelkező sorát egyszerre kell a memóriában tartanunk!!!
A rendezéses algoritmusok esetén viszont az eredményt rendezett sorrendben kapjuk meg.