6.3.3. Bináris műveletek egymenetes algoritmusai

Foglalkozzunk most a bináris műveletekkel: ezek az egyesítés, a metszet, a különbség, a szorzat és az összekapcsolás. Az összekapcsolások tárgyalásának egyszerűsítése céljából csak a természetes összekapcsolással foglalkozunk. Egy egyenlőségen alapuló összekapcsolás – az attribútumok megfelelő átnevezését követően – hasonló módon valósítható meg, a théta-összekapcsolásokat pedig felfoghatjuk úgy, mint egy szorzatot vagy egyenlőségen alapuló összekapcsolást, amelyet egy olyan kiválasztás követ, amit az egyenlőséges összekapcsolásban nem lehet kifejezni.

Létezik egy kivételes művelet – a multihalmazos egyesítés –, amelyet egy nagyon egyszerű egymenetes algoritmussal elvégezhetünk. $R \cup_B S$ kiszámításához először R minden sorát kitesszük a kimenetbe, majd ezt követően lemásoljuk S minden sorát, ahogy azt a 6.13. példában tettük. A lemez I/O-műveletek száma $B(R) + B(S)$, amint annak egy R és S operandusú egymenetes algoritmus esetén lennie is kell; az $M = 1$ pedig elegendő feltétel, függetlenül attól, hogy R és S mekkorák.

Más bináris műveletek esetén az R és S operandusok közül a kisebbiket kell beolvasni az elsődleges memóriába, majd olyan alkalmas adatstruktúrát kell kialakítani, hogy a sorokat mind beilleszteni, mind kikeresni könnyű legyen, amint azt a 6.3.2. részben ismertettük. Csakúgy mint korábban, itt is elegendő egy tördelőtáblázat vagy egy kiegyensúlyozott fa használata. A struktúra felépítéséhez kevés helyre van szükség (a sorok által amúgy is elfoglalt hely mellett), amit a továbbiakban elhanyagolunk. Így az R és S relációkon egy menetben végrehajtandó bináris művelettel szemben hozzávetőleges a következő követelmény adódik:

- $\min\big((B(R), B(S)\big) \le M$.

A fenti egyenlőtlenség azt tételezi fel, hogy egy puffert használunk a nagyobb reláció blokkjainak beolvasására, míg körülbelül M darab pufferre van szükség a teljes kisebbik reláció és a hozzá tartozó memóriabeli adatstruktúra befogadására.

Az alábbiakban megadjuk a különféle műveletekhez tartozó részleteket. Minden alkalommal feltesszük, hogy a relációk közül R a nagyobb, és S-et tartjuk az elsődleges memóriában.

Halmaz egyesítés

S-et beolvassuk M–1 memóriapufferbe, majd olyan keresési struktúrát építünk fel, amelyben a keresési kulcs maga a teljes sor. Az összes beolvasott sort ezután a kimenetbe is bemásoljuk. Ezután R minden egyes blokkját egyenként beolvassuk az M-edik pufferbe. Az R-ben lévő minden t sorra megnézzük, hogy az benne van-e S-ben, és ha nem, akkor t-t a kimenetbe másoljuk.

Halmaz metszet

S-et beolvassuk M–1 pufferbe, és olyan keresési struktúrát építünk fel, amelyben a teljes sorok alkotják a keresési kulcsot. Beolvassuk R minden blokkját, majd minden t sorára megnézzük, hogy az S-ben is szerepel-e. Ha igen, akkor t-t a kimenetbe másoljuk, ha pedig nem, akkor figyelmen kívül hagyjuk.

Halmaz különbség

Mivel a különbség nem kommutatív operátor, különbséget kell tennünk $R -_S S$ és $S -_S R$ között. Továbbra is feltételezzük, hogy R a nagyobbik reláció. Mindkét esetben S-et beolvassuk M–1 pufferbe, majd olyan keresési struktúrát építünk fel, amelyben teljes sorok alkotják a keresési kulcsot.

$R -_S S$ kiszámításához beolvassuk R minden blokkját, és egyenként megvizsgáljuk az adott blokk t sorait. Ha t S-ben van, akkor figyelmen kívül hagyjuk, ha pedig nincs S-ben, akkor bemásoljuk a kimenetbe.

$S -_S R$ kiszámításához ismét beolvassuk R blokkjait, és egymás után megvizsgáljuk a t sorokat. Ha t S-ben van, akkor töröljük azt S memóriabeli másolatából, ha pedig nincs, akkor nem csinálunk semmit. Miután R minden egyes sorát megvizsgáltuk, bemásoljuk a kimenetbe S megmaradó sorait.

Multihalmaz metszet

S-et beolvassuk *M*–1 pufferbe, majd minden egyes különböző sorhoz egy *számlálót* (count) rendelünk, amely kezdetben azt mutatja, hogy a szóban forgó sor hányszor fordul elő *S*-ben. Egy *t* sor több példányát nem többször, külön-külön tároljuk, hanem csak egyetlen másolatát, és ehhez társítjuk azt a számlálót, ami megegyezik *t* előfordulásainak számával.

Az említett struktúra valamivel több helyet igényelhetne mint $B(S)$ darab blokk, ha kevés számú ismétlődés fordulna elő, bár gyakran az a helyzet, hogy *S* kellően tömör. Így továbbra is azt tesszük fel, hogy a $B(S) \leq M-1$ elegendő feltétel egy egymenetes algoritmus működéséhez, bár ez a feltétel inkább csak közelítés.

A következőkben beolvassuk *R* minden egyes blokkját, és annak minden *t* sorára megnézzük, hogy a sor előfordul-e *S*-ben. Ha nem, akkor figyelmen kívül hagyjuk, *t* nem jelenhet meg a metszetben. Ha azonban *t* megjelenik *S*-ben, és a hozzá tartozó számláló még mindig pozitív, akkor *t*-t a kimenetbe tesszük, és a számlálót eggyel csökkentjük. Ha *t* megjelenik *S*-ben, de számlálója elérte a nullát, akkor nem tesszük a kimenetbe, hiszen ez azt jelenti, hogy *t*-nek már annyi példányát írtuk át a kimenetbe, ahányszor az *S*-ben megtalálható volt.

Multihalmaz különbség

$S -_B R$ kiszámításához *S* sorait beolvassuk a memóriába, majd megszámoljuk minden egyes különböző sor előfordulási gyakoriságát, ahogy azt a multihalmaz metszetnél is tettük. Amikor *R*-et beolvassuk, minden *t* sornál megnézzük, hogy az előfordul-e *S*-ben, és ha igen, akkor csökkentjük a hozzá tartozó számlálót. Ha ez megtörtént, akkor átmásoljuk a kimenetbe az összes olyan memóriabeli sort, amelynek a számlálója pozitív, és a sort annyiszor másoljuk, amennyi az említett számláló.

$R -_B S$ kiszámításához *S* sorait megint beolvassuk a memóriába, majd megszámoljuk a különböző sorok előfordulásának gyakoriságát. Egy *c* számlálójú *t* sorra gondolhatunk úgy, mintha *c* darab okunk lenne arra, hogy *t*-t ne másoljuk a kimenetbe *R* sorainak beolvasásakor. Más szavakkal, amikor *R* egy *t* sorát beolvassuk, megnézzük, hogy az *S*-ben szerepel-e. Ha nem, akkor *t*-t átmásoljuk a kimenetbe. Ha viszont *t* szerepel *S*-ben, akkor megnézzük a hozzá tartozó aktuális *c* számlálót. Ha $c = 0$, akkor *t*-t a kimenetbe másoljuk. Ha viszont $c > 0$, akkor *t*-t nem másoljuk a kimenetbe, hanem *c*-t csökkentjük eggyel.

Szorzat

S-et beolvassuk az elsődleges memória *M*–1 pufferébe. Itt nincs szükség semmilyen speciális adatstruktúrára. Ezt követően beolvassuk *R* minden egyes blokkját, majd *R* minden sorára *t*-t összefűzzük *S* minden egyes sorával a memóriában. Minden összefűzött sor úgy kerül a kimenetre, ahogyan előállt.

Vegyük észre, hogy ez az algoritmus minden *R*-beli sorra jelentős processzoridőt igényelhet, mert minden sort *M*–1 darab sorokkal teli blokkhoz kell párosítani. Ugyanakkor a kimenet mérete is nagy, és azt várhatjuk, hogy az eredmény lemezre írásához vagy a kimenet más jellegű feldolgozásához szükséges idő meghaladja majd a kimenet létrehozásához szükséges processzoridőt.

Természetes összekapcsolás

Itt és a többi összekapcsolási algoritmusnál éljünk azzal a konvencióval, hogy $R(X, Y)$-t kapcsoljuk össze $S(Y, Z)$-vel, ahol *Y* jelöli *R* és *S* összes közös attribútumát, *X* jelöli *R* összes olyan attribútumát, amelyek nincsenek benne *S* sémájában, végül pedig *Z* jelöli *S* összes olyan attribútumát, amelyek nincsenek benne *R* sémájában. Továbbra is feltesszük, hogy *S* a kisebbik reláció. A természetes összekapcsolás kiszámítását a következőképpen végezzük el:

1. Olvassuk be *S* összes sorát, és alkossunk belőlük egy olyan memóriabeli keresési struktúrát, amelyben *Y* attribútumai alkotják a keresési kulcsot. Szokás szerint egy tördelőtáblázat vagy egy kiegyensúlyozott fa jó példája az ilyen struktúráknak. E célra használjunk *M*–1 memóriablokkot.
2. Olvassuk be *R* minden egyes blokkját a fennmaradó egyetlen memóriapufferbe. *R* minden egyes *t* sorára keressük meg a keresési struktúrával *S* azon sorait, amelyek megegyeznek *t*-vel *Y* összes attribútumán. *S* minden egyes megegyező sorára alkossunk egy sort a *t*-vel való összekapcsolással, majd tegyük az eredményként kapott sort a kimenetbe.

Ugyanúgy mint az összes többi bináris, egymenetes algoritmusnál, ennél is $B(R) + B(S)$ lemez I/O-műveletre van szükség az operandusok beolvasásához. Az algoritmus mindaddig működik, amíg $B(S) \leq M-1$, illetve ha közelítőleg $B(S) \leq M$. A többi tanulmányozott algoritmushoz hasonlóan a memóriabeli keresési struktúra által igénybe vett pluszhelyet itt sem vettük figyelembe, mivel ez csak csekély többletet jelent.

A természetes összekapcsolástól különböző összekapcsolásokra itt nem fogunk kitérni. Emlékeztetünk rá, hogy az egyenlőségen alapuló összekapcsolást lényegében ugyanúgy kell elvégezni, mint a természetes összekapcsolást, de figyelembe kell vennünk, hogy a két reláció „azonos" attribútumai esetleg más névvel szerepelnek. Egy egyenlőségen alapuló összekapcsolástól (equijoin) különböző théta-összekapcsolás helyettesíthető egy olyan egyenlőségen alapuló összekapcsolással vagy szorzattal, amit egy kiválasztás követ.