Οι πράξεις της συνένωσης

ΠΡΟΜΗΘΕΥΤΗΣ (ΠΡΜ)

Κ_Προμ	Π_Ονομα	Είδος	Πόλη
22	Ανδρέου	7	Αθήνα
31	Πέτρου	8	Πάτρα
28	Δέδες	12	Λάρισα
58	Παππάς	7	Αθήνα

ΠΡΟΙΟΝ (ΠΡ)

Κ_Πρ	Πρ_Ονομα	Χρώμα	Βάρος
Π35	Οθόνη	Γκρι	8
Π76	DVD	Μαύρο	4
Π14	Κράνος	Άσπρο	1

ΕΡΓΟ (ΕΡ)

Κ_Ερ	Προυπ	Πόλη
Ε25	245Κ	Αθήνα
Ε32	500Κ	Άργος
Ε68	1000Κ	Αθήνα

ΠΑΡΑΓΓΕΛΙΑ (ΠΑΡ)

Κ_Προμ	Κ_Πρ	Κ_Ερ	Ποσοτ
22	Π35	Ε25	200
31	Π35	Ε25	100
31	Π76	Ε32	150
28	Π14	Ε68	150
22	Π76	Ε25	450
22	Π35	Ε32	670
22	Π35	Ε68	650
58	Π14	Ε68	670
28	Π14	Ε32	540
22	Π14	Ε25	200

Η συνένωση είναι μια από τις πιο χρήσιμες πράξεις της σχεσιακής άλγεβρας και είναι ουσιαστικά η κύρια πράξη για τον συνδυασμό πληροφοριών από δυο ή περισσότερες σχέσεις

Η πράξη της συνένωσης είναι μια από τις πιο χρονοβόρες πράξεις στην επεξεργασία επερωτήσεων. Οι πράξεις συνένωσης που εμφανίζονται πιο συχνά στις επερωτήσεις είναι η συνένωση ισότητας και η φυσική συνένωση και είναι αυτές που θα μελετήσουμε σε κάποια λεπτομέρεια.
Υπάρχουν πολλοί δυνατοί τρόποι για την υλοποίηση της συνένωσης δύο αρχείων. Συνενώσεις περισσοτέρων από δύο αρχείων καλούνται πολλαπλές συνενώσεις. Το πλήθος των δυνατών τρόπων εκτέλεσης πολλαπλών συνενώσεων αυξάνει πολύ γρήγορα, καθώς το πλήθος των αρχείων αυξάνει.

Η συνένωση αποτελείται από ένα Καρτεσιανό γινόμενο ακολουθούμενο από μια επιλογή. Όμως το Καρτεσιανό γινόμενο έχει μεγάλο κόστος (πόσο;) και η τελική επιλογή μπορεί να αφορά μικρό πλήθος πλειάδων.

Συνενώσεις σε Εμπορικά Συστήματα

Sybase ASE:Υποστηρίζει εμφωλευμένους βρόγχους με ευρετήριο και συνένωση με ταξινόμηση και συγχώνευση.

Sybase ASIQ: Υποστηρίζει εμφωλευμένους βρόγχους σε σελίδες, εμφωλευμένους βρόγχους με ευρετήριο, κατακερματισμό, και συνένωση με ταξινόμηση και συγχώνευση.

Oracle: Υποστηρίζει εμφωλευμένους βρόγχους σε σελίδες, μια παραλλαγή της συνένωσης του υβριδικού κατακερματισμού, και συνένωση με ταξινόμηση και συγχώνευση.

DB2: Υποστηρίζει εμφωλευμένους βρόγχους σε σελίδες, εμφωλευμένους βρόγχους με ευρετήριο, και συνένωση υβριδικού κατακερματισμού.

SQL Server: Υποστηρίζει εμφωλευμένους βρόγχους σε σελίδες, εμφωλευμένους βρόγχους με ευρετήριο, συνένωση με ταξινόμηση και συγχώνευση, και συνένωση με κατακερματισμό

Εμφωλευμένοι βρόχοι (εξαντλητική μέθοδος)

Ο απλούστερος αλγόριθμος συνένωσης. Για κάθε εγγραφή t της σχέσης R (εξωτερικός βρόχος), ανακτάται κάθε εγγραφή s από τη σχέση S (εσωτερικός βρόγχος) και ελέγχεται αν οι δύο εγγραφές ικανοποιούν τη συνθήκη συνένωσης.

foreach tuple $r \in R$ do
  foreach tuple $s \in S$ do
    if $r_i == s_j$ then output <r,s>

Η χειρότερη περίπτωση είναι αν εξετάζουμε μια μια τις πλειάδες της R με αυτές της S.
Αν b_R και b_S είναι οι σελίδες αντίστοιχα των σχέσεων και p_R και p_S οι πλειάδες ανά σελίδα τότε το κόστος είναι:

$$b_R + p_R * b_R * b_S.$$

Που δίνει απόθανα αποτελέσματα
Αν $b_{ΠΑΡ}$=1000, $b_{ΠΡΜ}$=500 και $p_{ΠΑΡ}$=100
Σημαίνει κόστος $1000+100*1000*500=1000+(5*10^7)$
που με χρόνο προσπέλασης 10ms δίνει 140 ώρες

Δεν λαμβάνουμε υπόψη και το κόστος εγγραφής του αποτελέσματος.

Μια βελτίωση μπορεί να γίνει αν δεν εξετάζουμε μια-μια τις εγγραφές αλλά σελίδες.

Με τον τρόπο αυτό το κόστος θα είναι:

$\mathbf{b_R + b_R * b_S}$**.**

Οπότε στη συγκεκριμένη περίπτωση

1000+1000*500=501000

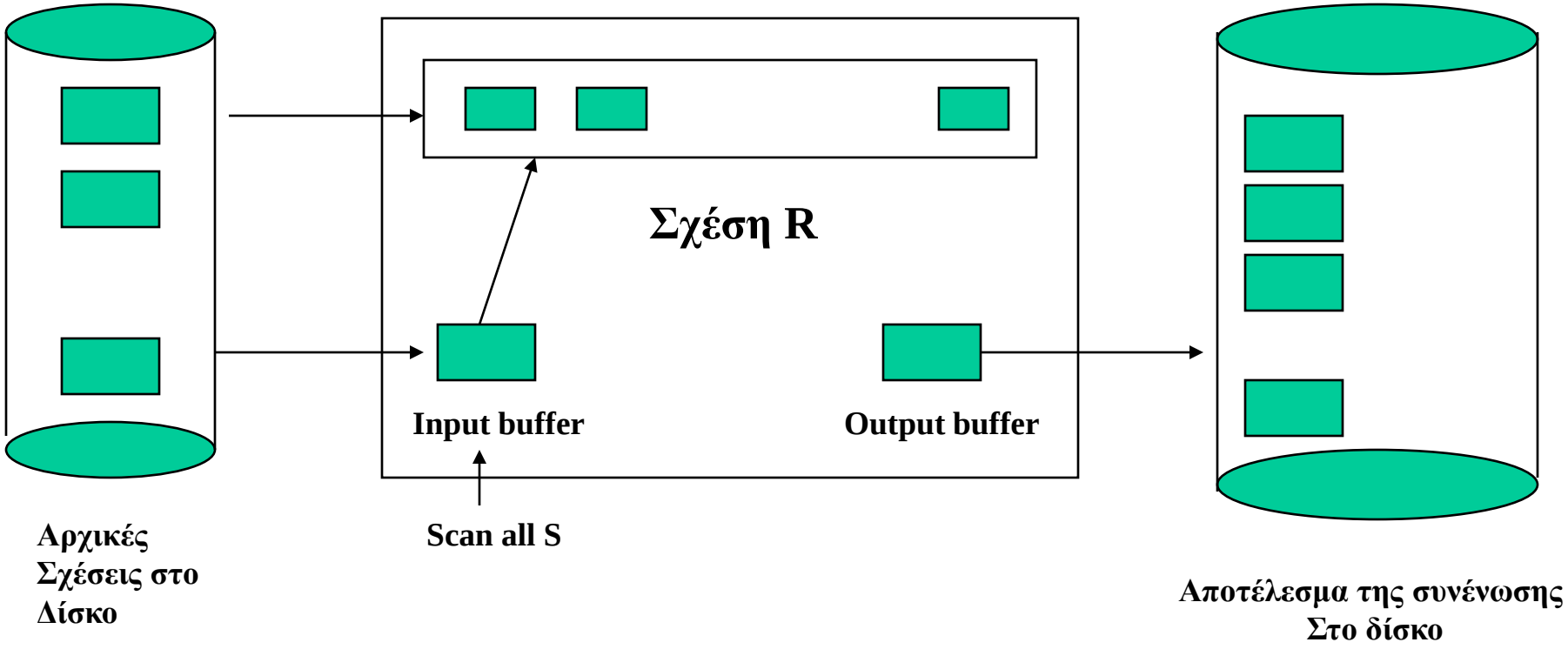
Ή 1.4 ώρες

Από τους τύπους αυτούς παρατηρούμε ότι έχει σημασία ποια σχέση θα διαλέξουμε σαν εξωτερική στο βρόγχο. Η εξωτερική θα πρέπει να είναι η μικρότερη από τις δυο.

Εμφωλευμένοι βρόγχοι με χρήση ενδιάμεσης μνήμης

Στον προηγούμενο αλγόριθμο δεν κάναμε χρήση της δυνατότητας του buffer εισόδου. Αν ένα buffer μεγέθους B σελίδων τότε ο αλγόριθμος μπορεί να τροποποιηθεί σε:

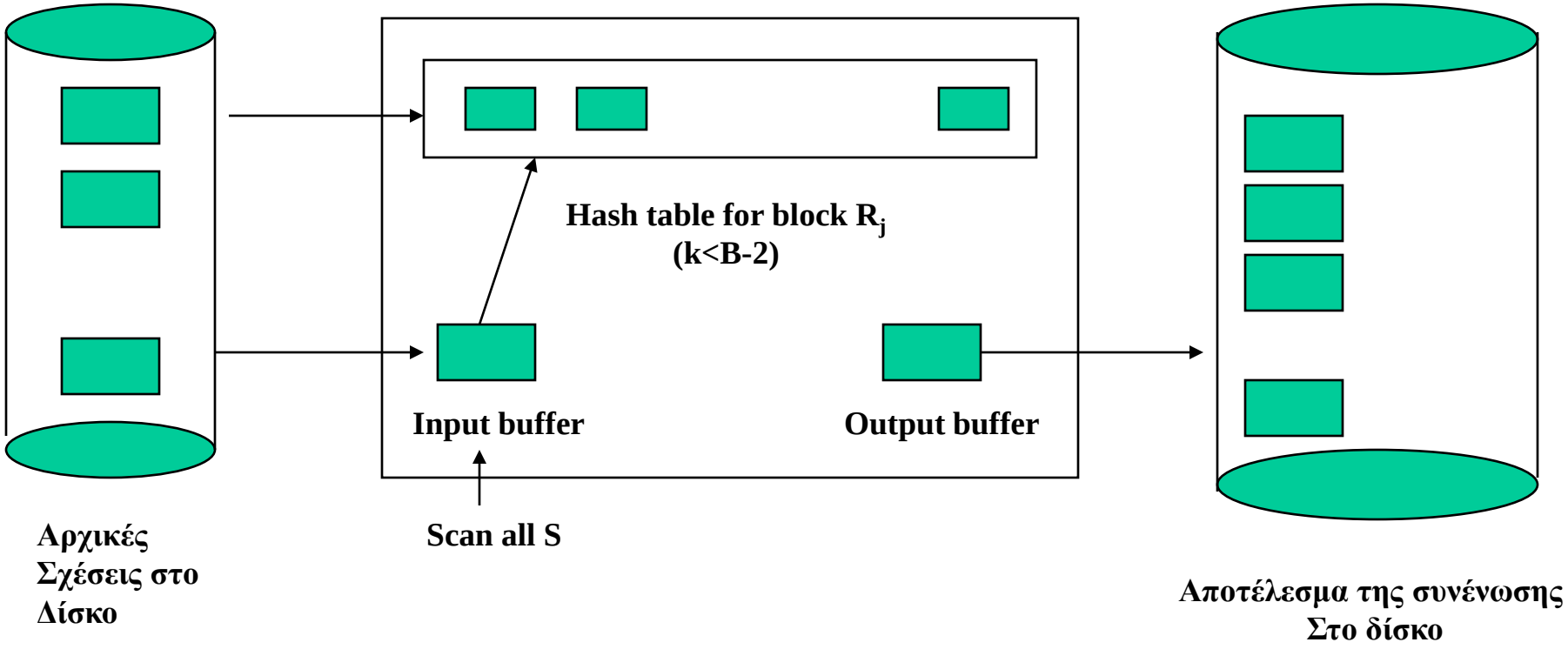
foreach block από B-2 σελίδες της R do
foreach σελίδα της S do
για όλες τις πλειάδες r∈R στο block που αντιστοιχούν σε πλειάδες s∈S της σελίδας
output <r,s>


Σχέση R
Input buffer
Output buffer
Αρχικές
Σχέσεις στο
Δίσκο
Scan all S
Αποτέλεσμα της συνένωσης
Στο δίσκο

Στην περίπτωση αυτή το κόστος θα είναι:
$b_R + \lceil b_R /(B-2) \rceil * b_S$.

Αν επομένως στην περίπτωση μας B=102 το κόστος θα είναι:
1000+10*500=6000
Ενώ αν η εξωτερική είναι η ΠΡ
500+5*1000=5500

Μπορούμε επίσης να βελτιώσουμε κάπως το CPU κόστος αν για την σχέση R κρατήσουμε ένα πίνακα κατακερματισμού στην κύρια μνήμη


Hash table for block R_j
(k<B-2)
Input buffer
Output buffer
Αρχικές
Σχέσεις στο
Δίσκο
Scan all S
Αποτέλεσμα της συνένωσης
Στο δίσκο

Συνένωση απλού βρόγχου (χρήση μιας δομής προσπέλασης για ανάκτηση των εγγραφών που ικανοποιούν τη συνθήκη)

Αν υπάρχει ευρετήριο (ή κλειδί κατακερματισμού) για ένα από τα δύο γνωρίσματα της συνένωσης -έστω το j της σχέσης S- ανακτώνται όλες οι εγγραφές r της σχέσης R, μία προς μία, και χρησιμοποιείται η δομή προσπέλασης για να ανακτηθούν άμεσα όλες οι εγγραφές της σχέσης S που ικανοποιούν τη συνθήκη $s_j=r_i$

foreach tuple r∈R do
foreach tuple s∈S where
$r_i==s_j$ then output <r,s>

Η αναζήτηση στο S με χρήση του ευρετηρίου

Για κάθε πλειάδα r∈R χρησιμοποιούμε το ευρετήριο για ανάκτηση των αντίστοιχων πλειάδων της S.
Η διαφορά της μεθόδου είναι ότι δεν κάνει αναζήτηση στο Καρτεσιανό γινόμενο των σχέσεων. Το κόστος σάρωσης της R είναι το ίδιο b_R όπως και στην προηγούμενη μέθοδο.Όμως το κόστος ανάκτησης των πλειάδων της S εξαρτάται από το είδος του ευρετηρίου και την επιλεξιμότητα.

Για κάθε πλειάδα της R το κόστος αναζήτησης της S είναι:

1. **Αν το ευρετήριο της S είναι B+ δένδρο το κόστος εύρεσης του κόμβου φύλου είναι μεταξύ 2 και 4. Αν είναι ευρετήριο κατακερματισμού 1 με 2.**
2. **Μόλις βρεθεί ο κατάλληλος τερματικός κόμβος το κόστος ανάκτησης εξαρτάται από το αν το ευρετήριο είναι συστάδα. Αν είναι το κόστος είναι μια ακόμη προσπέλαση. Αν όχι το κόστος θα είναι τόσες προσπελάσεις όσες και οι αντίστοιχες πλειάδες της S (υποθέτοντας ότι βρίσκονται σε διαφορετική σελίδα).**

Παράδειγμα
Έστω ΠΑΡ*ΠΡΜ

Ας υποθέσουμε ότι η ΠΡΜ έχει ευρετήριο κατακερματισμού στο Κ_Προμ που είναι και κλειδί με προσπέλαση 1.2 κατά μέσο όρο.

Το ΠΑΡ έχει 1000 σελίδες των 100 πλειάδων η σελίδα. Επομένως το συνολικό κόστος είναι:

1000+100000(1+1.2)=222000

Συνένωση ταξινόμησης-συγχώνευσης

Αν οι εγγραφές των σχέσεων R και S είναι φυσικά ταξινομημένες σύμφωνα με τις τιμές των γνωρισμάτων συνένωσης Α και Β αντίστοιχα, μπορούμε να υλοποιήσουμε τη συνένωση κατά τον πιο αποδοτικό τρόπο. Και τα δύο αρχεία σαρώνονται σύμφωνα με τη διάταξη των γνωρισμάτων συνένωσης και επιλέγεται ο συνδυασμός των εγγραφών με τις ίσες τιμές στα Α και Β. Αν τα αρχεία δεν είναι ταξινομημένα, μπορεί να ταξινομηθούν πρώτα χρησιμοποιώντας μια εξωτερική ταξινόμηση. Με τη μέθοδο αυτή οι εγγραφές κάθε αρχείου σαρώνονται μόνο μιά φορά, προκειμένου να ταιριάξουν με αυτές του άλλου αρχείου -εκτός αν και τα Α και Β δεν είναι γνωρίσματα-κλειδιά, στην οποία περίπτωση η μέθοδος πρέπει να τροποποιηθεί ελαφρά.

Μια παραλλαγή του αλγόριθμου ταξινόμησης-συγχώνευσης μπορεί να χρησιμοποιηθεί όταν υπάρχουν δευτερεύοντα ευρετήρια και στα δύο γνωρίσματα συνένωσης. Τα ευρετήρια παρέχουν τη δυνατότητα προσπέλασης (σάρωσης) των εγγραφών με διάταξη των γνωρισμάτων συνένωσης, αλλά οι ίδιες οι εγγραφές μπορεί να είναι φυσικά διεσπαρμένες σε όλα τα μπλοκ του αρχείου και έτσι αυτή η μέθοδος μπορεί να μην είναι αποτελεσματική καθώς κάθε προσπέλαση εγγραφής μπορεί να απαιτεί προσπέλαση ενός μπλοκ δίσκου.

Η βασική ιδέα του αλγόριθμου είναι να ταξινομηθούν οι δυο σχέσεις στο γνώρισμα συνένωσης και στη συνέχεια να γίνει έλεγχος για τις πλειάδες που αντιστοιχούν.
Για την ταξινόμηση μπορεί να χρησιμοποιηθεί μια μέθοδος εξωτερικής ταξινόμησης.

ΠΡΜ

Κ_Προμ	Π_Ονομα	Είδος	Πόλη
22	Ανδρέου	7	Αθήνα
31	Πέτρου	8	Πάτρα
28	Δέδες	12	Λάρισα
58	Παππάς	7	Αθήνα

ΠΑΡ

Κ_Προμ	Κ_Πρ	Κ_Ερ	Ποσοτ
22	Π35	Ε25	200
31	Π35	Ε25	100
31	Π76	Ε32	150
28	Π14	Ε68	150
22	Π76	Ε25	450
22	Π35	Ε32	670
22	Π35	Ε68	650
58	Π14	Ε68	670
28	Π14	Ε32	540
22	Π14	Ε25	200

ΠΡΜ

Κ_Προμ	Π_Ονομα	Είδος	Πόλη
22	Ανδρέου	7	Αθήνα
28	Δέδες	12	Λάρισα
31	Πέτρου	8	Πάτρα
58	Παππάς	7	Αθήνα

ΠΑΡ

Κ_Προμ	Κ_Πρ	Κ_Ερ	Ποσοτ
22	Π35	Ε25	200
22	Π14	Ε25	200
22	Π35	Ε68	650
22	Π76	Ε25	450
22	Π35	Ε32	670
28	Π14	Ε68	150
28	Π14	Ε32	540
31	Π35	Ε25	100
31	Π76	Ε32	150
58	Π14	Ε68	670

procedure join(R,S, ‘$R_i = S_j$ ‘)
Tr = η πρώτη πλειάδα της R;
Ts= η πρώτη πλειάδα της S;
Gs η πρώτη πλειάδα της S
while Tr≠ eof and Gs ≠ eof do{
while Tr_i< Gs_j do
Tr = η επόμενη πλειάδα της Tr από την R;
while Tr_i> Gs_j do
Gs = η επόμενη πλειάδα της Gs από την S;
Ts=Gs;
while Tr_i == Gs_j do{
Ts=Gs;
while Ts_j== Tr_i do{
output <Tr, Ts>
Ts= η επόμενη πλειάδα της Ts από την S;}
Tr= η επόμενη πλειάδα της Ts από την R;
}
Gs=Ts
}

Το κόστος ταξινόμησης της R είναι $O(b_R \log b_R)$
Το κόστος ταξινόμησης της S είναι $O(b_S \log b_S)$
Το κόστος της συγχώνευσης είναι b_R+b_S αν δεν χρειασθεί να επαναφερθούν κάποια τμήματα της S.

Η μέθοδος αυτή είναι ιδιαίτερα καλή αν τουλάχιστον μια από τις σχέσεις είναι ταξινομημένη στο γνώρισμα της συνένωσης ή αν έχει ευρετήριο συστάδα στο γνώρισμα συνένωσης.

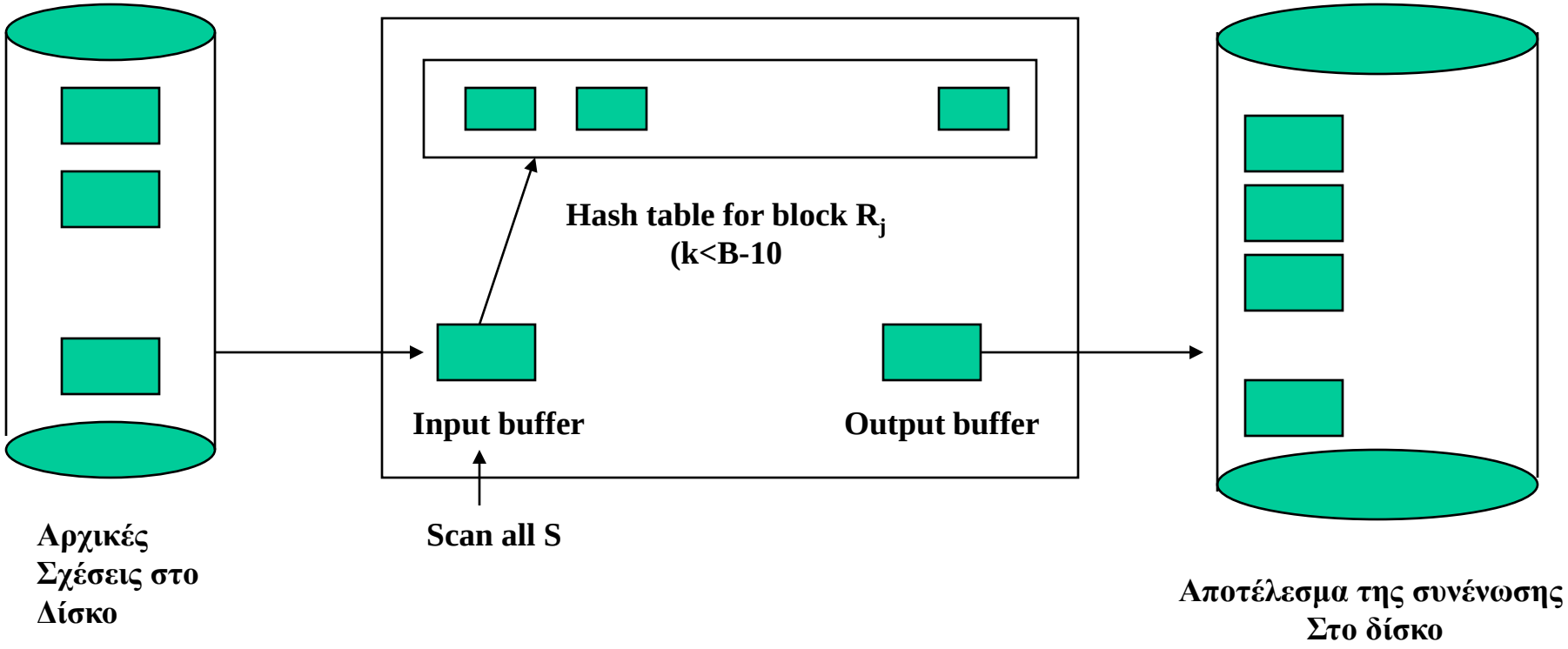
Συνένωση με κατακερματισμό

Κατά την μέθοδο αυτή δημιουργείται ένα ευρετήριο κατακερματισμού για τις εγγραφές των αρχείων R και S, χρησιμοποιώντας τα γνωρίσματα συνένωσης A και B ως κλειδιά κατακερματισμού. Και για τα δύο αρχεία χρησιμοποιείται το ίδιο αρχείο κατακερματισμού και η ίδια συνάρτηση κατακερματισμού.

Πρώτη φάση (φάση διαμέρισης) :Πρώτα διατρέχεται το αρχείο με τις λιγότερες εγγραφές (έστω το R) μία μόνο φορά, προκειμένου να απεικονισθούν οι εγγραφές του στους κάδους του αρχείου κατακερματισμού (οι εγγραφές του R μοιράζονται στους κάδους κατακερματισμού).

Δεύτερη φάση (που ονομάζεται φάση της αναζήτησης) το άλλο αρχείο (S) διατρέχεται μία μόνο φορά με απεικόνιση κάθε εγγραφής του στον κατάλληλο κάδο, όπου η εγγραφή συνδυάζεται με όλες τις εγγραφές του αρχείου R που ταιριάζουν με αυτή. Αυτή η απλουστευμένη περιγραφή της συνένωσης με κατακερματισμό υποθέτει ότι το μικρότερο από τα δύο αρχεία *χωράει τελείως σε κάδους της μνήμης* μετά την πρώτη φάση.

Η ιδέα είναι να γίνει κατακερματισμός και των δυο σχέσεων στο γνώρισμα της συνένωσης χρησιμοποιώντας την ίδια συνάρτηση κατακερματισμού h. Αν ο κατακερματισμός χωρίσει σε k διαμερίσεις εξασφαλίζουμε ότι οι πλειάδες της R στην i διαμέριση έχουν αντίστοιχες πλειάδες στην S μόνο από την ίδια διαμέριση. Αυτό σημαίνει ότι μπορούμε να διαβάσουμε μια πλήρη διαμέριση της μικρότερης σχέσης R και απλά να σαρώσουμε μόνο την αντίστοιχη διαμέριση της S. Αυτές οι πλειάδες των R και S δεν θα χρειασθεί να εξετασθούν ξανά. Επομένως μετά την διαμέριση των R και S μπορεί να γίνει η συγχώνευση μόνο με μια ανάγνωση των R και S, με την προϋπόθεση ότι υπάρχει διαθέσιμη μνήμη για κάθε διαμέριση της R.


Hash table for block R_j
(k<B-10
Input buffer
Output buffer
Scan all S
Αρχικές
Σχέσεις στο
Δίσκο
Αποτέλεσμα της συνένωσης
Στο δίσκο

Πρακτικά για κάθε διαμέριση χτίζεται ένας πίνακας κατακερματισμού για κάθε διαμέριση της R, χρησιμοποιώντας μια συνάρτηση κατακερματισμού h2 που είναι διαφορετική από την h, ώστε να μειωθεί το CPU κόστος. Χρειαζόμαστε αρκετή μνήμη για να μπορεί να καταχωρηθεί αυτός ο πίνακας κατακερματισμού, που θα είναι λίγο μεγαλύτερη από την διαμέριση της R.

//Διαμέριση της R
foreach tuple r∈R
read r και πρόσθεσε την στην σελίδα εξόδου h(r_i)
//Διαμέριση της S
foreach tuple s∈S
read r και πρόσθεσε την στην σελίδα εξόδου h(s_i)
//Φάση συνένωσης
for l=1,…,k do {
//Δημιουργία του πίνακα κατακερματισμού για την R_i, με χρήση της h_2
foreach tuple r ∈R_l do
read r και εισαγωγή στον πίνακα κατακερματισμού με χρήση της $h_2(r_l)$;
//Σάρωση της Sl για αντίστοιχες πλειάδες στην R_l
foreach tuple s∈ στη διαμέριση S_l do {
read s και απεικόνισε με χρήση της h_2;
Για τις πλειάδες r της R που αντιστοιχούν output <r,s> };
Καθαρισμός του πίνακα κατακερματισμού;
}

Κόστος της συνένωσης με κατακερματισμό

Στη φάση της διαμέρισης σαρώνουμε την R και την S μια φορά και γράφουμε μια φορά. Επομένως το κόστος αυτής της φάσης είναι $2(b_R+b_S)$.

Στη δεύτερη φάση σαρώνουμε κάθε διαμέριση μια φορά, υποθέτοντας ότι κάθε διαμέριση χωράει στη μνήμη, το κόστος είναι b_R+b_S. Επομένως το συνολικό κόστος είναι $3(b_R+b_S)$.

Επομένως στο παράδειγμα μας θα είναι:

3*(500+1500)=4500

που για πρόσβαση 10ms κοστίζει λιγότερο από ένα λεπτό.

Απαιτήσεις σε μνήμη

Για να αυξήσουμε τις πιθανότητες να χωράει κάθε διαμέριση στη μνήμη πρέπει να ελαχιστοποιήσουμε το μέγεθος κάθε διαμέρισης μεγιστοποιώντας το πλήθος των διαμερίσεων. Στη φάση της διαμέρισης, για διαμέριση της R (και της S) σε k διαμερίσεις χρειαζόμαστε k block εξόδου και ένα εισόδου. Επομένως με B σελίδες το μέγιστο πλήθος διαμερίσεων είναι k=B-1. Υποθέτοντας ότι οι διαμερίσεις έχουν ίδιο μέγεθος, τότε το μέγεθος κάθε διαμέρισης της R είναι b_R /(B-1). Το πλήθος των σελίδων του πίνακα κατακερματισμού (στη μνήμη) κατά την φάση της συνένωσης είναι f*b_R /(B-1) (όπου f ο παράγοντας αύξησης λόγω κατακερματισμού).

Κατά την φάση της συγχώνευσης θέλουμε μια σελίδα για την σάρωση των διαμερίσεων της S και μια για την έξοδο. Επομένως για να χωράει στη μνήμη η διαμέριση B>(f*b_R /(B-1))+2. Επομένως για B>√f* √ b_R έχουμε καλή απόδοση του αλγόριθμου.

Αν επομένως $B>\sqrt{f}*\sqrt{b_R}$ αναμένουμε ότι κάθε διαμέριση θα έχει μέγεθος $b_R/(B-1)$. Αν δεν έχουμε τόση μνήμη η αν δεν γίνει ομοιόμορφη κατανομή θα έχουμε κάποια υπερχείλιση. Η απόδοση στην περίπτωση της υπερχείλισης μπορεί να βελτιωθεί με επανάληψη του κατακερματισμού (της R).

Υβριδικός Αλγόριθμος Συνένωσης με Κατακερματισμό

Αν διαθέτουμε περισσότερη μνήμη μπορεί να χρησιμοποιήσουμε μια παραλλαγή του αλγόριθμου κατακερματισμού. Έστω ότι $B>f*(b_R/k)$ για κάποιο ακέραιο k. Δηλαδή αν διαμερίσουμε την R σε k διαμερίσεις μεγέθους b_R/k μπορούμε να έχουμε ένα πίνακα κατακερματισμού στη μνήμη. Για την διαμέριση της R (και της S) σε k σελίδες εξόδου και μια σελίδα εισόδου, δηλαδή k+1 σελίδες. Επομένως έχουμε $B-(k+1)$ επιπλέον σελίδες κατά την φάση της διαμέρισης.

Έστω ότι $B-(k+1)>f*(b_R/k)$. Τότε έχουμε επιπλέον μνήμη κατά την φάση της διαμέρισης για να χωρέσει και μια διαμέριση της R. Η ιδέα είναι να κρατήσουμε στη μνήμη την πρώτη διαμέριση της R. Επίσης κατά της διαμέριση της S η πρώτη διαμέριση δεν γράφεται αλλά συγκρίνεται άμεσα με την πρώτη της R. Το κέρδος είναι ότι δεν θα ξαναδιαβαστούν οι πρώτες διαμερίσεις των R και S.

Για παράδειγμα αν b_R=500 σελίδες και b_S=1000 σελίδες και B=300. Μπορούμε να κρατήσουμε στη μνήμη την πρώτη διαμέριση της R όταν χωρίζεται σε δυο διαμερίσεις. Κατά τη φάση της διαμέρισης της R σαρώνουμε την R και γράφουμε μια διαμέριση δηλαδή το κόστος είναι 500+250. Σαρώνουμε την S γράφουμε μια διαμέριση. Το κόστος είναι 1000+500 (αν υποθέσουμε ότι οι διαμερίσεις έχουν ίδιο μέγεθος. Στην συνένωση σαρώνουμε την δεύτερη διαμέριση των R και S με κόστος 250+500. Συνολικό κόστος 750+1500+750=3500.

Σύγκριση Συνένωσης με κατακερματισμό με συνένωση συγχώνευσης ταξινόμησης

Αν $B>\sqrt{b_R}$ όπου b_R το μέγεθος της μικρότερης σχέσης, και υποθέσουμε ομοιόμορφη κατανομή στις διαμερίσεις το κόστος του κατακερματισμού είναι $3(b_R+b_S)$.

Αν $B>\sqrt{b_S}$ όπου b_S το μέγεθος της μεγαλύτερης σχέσης, τότε το πλήθος των ταξινομημένων σειρών κάθε σχέσης είναι μικρότερο του $\sqrt{b_S}$ και αν υποθέσουμε ότι οι διαθέσιμες σελίδες για την συγχώνευσης είναι περισσότερες από $2\sqrt{b_S}$ δηλαδή περισσότερες από το συνολικό πλήθος των ταξινομημένων σειρών, μπορούμε να διαθέσουμε μια σελίδα για κάθε σειρά της R και της S. Τότε μπορεί να γίνει ταυτόχρονα η συγχώνευση των σειρών της R και της S. Βέβαια χρειαζόμαστε μνήμη $B>2\sqrt{b_S}$.

Συνολικό κόστος $3(b_R+b_S)$.

- **Αν οι διαμερίσεις στον κατακερματισμό δεν έχουν ομοιόμορφη κατανομή , η συνένωση με κατακερματισμό μπορεί να κοστίσει παραπάνω.**

Αν η διαθέσιμε μνήμη είναι μεταξύ $\sqrt{b_R}$ και $\sqrt{b_S}$ τότε η συγχώνευση με κατακερματισμό κοστίζει λιγότερο (αφού υπάρχει μνήμη μόνο για τις διαμερίσεις της μικρότερης σχέσης)

Επιπλέον παράγοντες που πρέπει να ληφθούν υπόψη είναι αν κάποια σχέση είναι ταξινομημένη σε κάποιο γνώρισμα συνένωσης.

Γενικότερες Συνθήκες Συνένωσης περιλαμβάνουν ισότητα σε περισσότερα από ένα γνωρίσματα και συνθήκες άλλες εκτός από ισότητα.

- **Για ισότητα σε περισσότερα από ένα γνωρίσματα στον αλγόριθμο εμφωλευμένων βρόγχων με ευρετήριο μπορούμε να χτίσουμε ένα ευρετήριο στο συνδυασμό των γνωρισμάτων της R και να την χρησιμοποιήσουμε σαν εσωτερική σχέση ή μπορούμε να χρησιμοποιήσουμε υπάρχον ευρετήριο σ' αυτό τον συνδυασμό γνωρισμάτων.**
- **Για ισότητα σε περισσότερα από ένα γνωρίσματα στον αλγόριθμο ταξινόμησης συγχώνευσης μπορούμε να ταξινομήσουμε στον συνδυασμό των γνωρισμάτων. Επίσης στην συνένωση με κατακερματισμό, ο κατακερματισμός γίνεται στο συνδυασμό των γνωρισμάτων.**

Σε περίπτωση ανισότητας R.A<S.B

- **Χρειαζόμαστε ένα B+ δενδρικό ευρετήριο για τον αλγόριθμο εμφωλευμένων βρόγχων με ευρετήριο.**
- **Οι αλγόριθμοι ταξινόμησης συγχώνευσης και κατακερματισμού συγχώνευσης δεν μπορούν να εφαρμοσθούν.**
- **Οι άλλοι αλγόριθμοι μπορούν να εφαρμοσθούν**

Συνολοθεωρητικές Πράξεις

Από άποψη υλοποίησης το Καρτεσιανό γινόμενο και η τομή μπορούν να θεωρηθούν ειδικές περιπτώσεις της συνένωσης. Για την τομή συνένωση με ισότητα για όλα τα γνωρίσματα και το Καρτεσιανό γινόμενο συνένωση χωρίς συνθήκη συνένωσης.

Το βασικό πρόβλημα για για την πράξη της ένωσης είναι η απαλοιφή των διπλότυπων.Μια παραλλαγή της μεθόδου απαλοιφής των διπλότυπων μπορεί να χρησιμοποιηθεί και για την διαφορά.

Ταξινόμηση για Ένωση και Διαφορά

1. **Ταξινόμηση της R και της S στο συνδυασμό όλων των γνωρισμάτων**
2. **Σάρωση των ταξινομημένων R και S και συγχώνευση με απαλοιφή των διπλότυπων. (Μπορεί να γίνει και με δημιουργία ταξινομημένων σειρών και συγχώνευση τους)**

Παρόμοια γίνεται και η διαφορά μόνο που κατά την φάση της συγχώνευσης γράφουμε τις πλειάδες της R αφού ελεγχθεί ότι δεν ανήκουν στην S

Κατακερματισμό για Ένωση και Διαφορά

1. **Διαμέριση των R και S με χρήση μιας συνάρτησης κατακερματισμού h.**
2. **Για κάθε διαμέριση l:**

- **Δημιουργία ενός πίνακα κατακερματισμού (χρησιμοποιώντας μια συνάρτηση κατακερματισμού h2≠h) για την S_l.**
- **Σάρωση της R_l. Για κάθε πλειάδα απεικόνιση στον πίνακα κατακερματισμού της S_l. Αν η πλειάδα υπάρχει στον πίνακα απορρίπτεται διαφορετικά προστίθεται.**
- **Εγγραφή του πίνακα και ετοιμασία για την επόμενη διαμέριση**

Με τον ίδιο τρόπο γίνεται και η διαφορά (R-S). Η διαφορά είναι στην επεξεργασία της διαμέρισης. Μετά τον πίνακα κατακερματισμού της S_l,σαρώνουμε την R_l. Αν μια πλειάδα δεν είναι στον πίνακα γράφεται στο αποτέλεσμα.