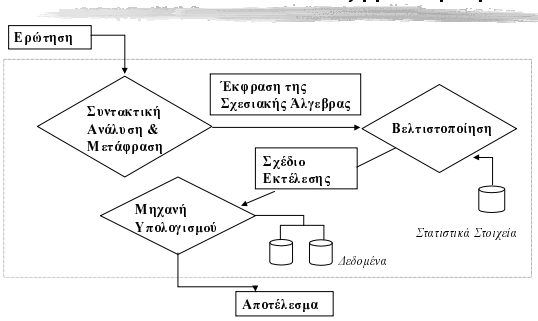
Επεξεργασία Ερωτήσεων

Εισαγωγή

1. ΜΟΝΤΕΛΑ ΔΕΔΟΜΕΝΩΝ
   - Μοντέλα
   - Γλώσσες Ερωτήσεων
   - **Επεξεργασία Ερωτήσεων**
     - **Επεξεργασία Ερωτήσεων σε Σχεσιακά ΣΔΒΔ**
     - **Επεξεργασία Ερωτήσεων σε Κατανεμημένα Σχεσιακά ΣΔΒΔ**
     - **Επεξεργασία Ερωτήσεων σε Ημιδομημένα Δεδομένα**
2. ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ
3. ΤΡΟΠΟΙ ΜΕΤΑΔΟΣΗΣ

Επεξεργασία Ερωτήσεων σε Σχεσιακά ΣΔΒΔ

Επεξεργασία Ερωτήσεων



Επεξεργασία Ερωτήσεων

Τα βασικά βήματα στην επεξεργασία μιας ερώτησης είναι

1. Συντακτική Ανάλυση & Μετάφραση
2. Βελτιστοποίηση
3. Υπολογισμός

Επεξεργασία Ερωτήσεων

1. Συντακτική Ανάλυση (Parsing) & Μετάφραση

Η ερώτηση μεταφράζεται σε μια εσωτερική μορφή αφού γίνει ο απαραίτητος συντακτικός και σημασιολογικός έλεγχος (π.χ., τα ονόματα που αναφέρονται είναι ονόματα σχέσεων που υπάρχουν)

Αντικατάσταση των όψεων από τον ορισμό τους

Εσωτερική μορφή; Έκφραση της σχεσιακής άλγεβρας

2. Βελτιστοποίηση

Μια SQL ερώτηση μπορεί να μεταφραστεί σε διαφορετικές (ισοδύναμες) εκφράσεις της σχεσιακής άλγεβρας

select balance
from account
where balance < 25000

- $\sigma_{balance < 2500}(\pi_{balance}(account))$
- $\pi_{balance}(\sigma_{balance < 2500}(account))$

2. Βελτιστοποίηση

Άρα δεν αρκεί ο προσδιορισμός της πράξης - πρέπει να προσδιορίζεται και ο αλγόριθμος που θα χρησιμοποιηθεί για την υλοποίηση της

π.χ., για την υλοποίηση της επιλογής μπορεί

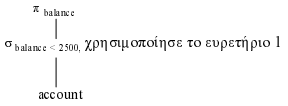
είτε να σαρώσουμε (scan) όλο το αρχείο ελέγχοντας κάθε εγγραφή

είτε αν υπάρχει π.χ., ένα B^+ ευρετήριο στο γνώρισμα balance να χρησιμοποιήσουμε το ευρετήριο

2. Βελτιστοποίηση

Κάθε πράξη της σχεσιακής άλγεβρας μπορεί να υλοποιηθεί με διαφορετικούς αλγορίθμους: βασικές (primitive) πράξεις (πράξη + αλγόριθμος)

Σχέδιο εκτέλεσης (execution plan): μια ακολουθία από βασικές πράξεις



2. Βελτιστοποίηση

- Τα διαφορετικά σχέδια εκτέλεσης έχουν και διαφορικό κόστος
- Βελτιστοποίηση: η διαδικασία επιλογής του σχεδίου εκτέλεσης που έχει το μικρότερο κόστος
- Εκτίμηση του κόστους (συνήθως χρήση στατιστικών στοιχείων)

3. Εκτέλεση

Μηχανή εκτέλεσης που εκτελεί τις βασικές πράξεις

1. Αλγόριθμους εκτέλεσης βασικών πράξεων
2. Ισοδυναμία σχεσιακών εκφράσεων - ευριστική επιλογή

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Εκτίμηση κόστους - αρχείο δεδομένων

- n_R: αριθμός πλειάδων της σχέσης R
- b_R: αριθμός blocks της σχέσης R
- s_R: μέγεθος σε bytes κάθε πλειάδας της σχέσης R
- f_R: παράγοντας ομαδοποίησης

  αν μη εκτεινόμενη, $f_R = \lfloor B / s_R \rfloor$ και $b_R = \lceil n_R / f_R \rceil$
- V(A, R): αριθμός διαφορετικών τιμών του A

  $|\pi_A(R)|$ -- αν A κλειδί:
- SC(A, R): μέσος αριθμός πλειάδων που ικανοποιεί μια συνθήκη (δεδομένου ότι υπάρχει μια τουλάχιστον που την ικανοποιεί)

  1 αν κλειδί, αν ομοιόμορφη;

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Εκτίμηση κόστους - αρχείο ευρετηρίου

- f_i: παράγοντας διακλάδωσης,

  πολυεπίπεδο f_0, B^+ δέντρο ~ βαθμό
- H_i: αριθμός επιπέδων
- LB_i: αριθμός block φύλλων

Αριθμό blocks που μεταφέρονται

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Επιλογή

E1 Σειριακή αναζήτηση

E2 Δυαδική αναζήτηση

E3 Χρήση πρωτεύοντος ευρετηρίου/ κατακερματισμού

E4 Χρήση δευτερεύοντος ευρετηρίου/ κατακερματισμού

Μονοπάτι προσπέλασης

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Επιλογή - συνθήκη ισότητας

$\sigma_{A = \alpha}(R)$

E1 Σειριακή αναζήτηση

b_R

$b_R / 2$ αν το A υποψήφιο κλειδί

Μπορεί να χρησιμοποιηθεί σε οποιοδήποτε αρχείο

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Επιλογή - συνθήκη ισότητας

E2 Δυαδική αναζήτηση

Μπορεί να χρησιμοποιηθεί μόνο αν το αρχείο είναι διατεταγμένο με βάση το γνώρισμα της επιλογής

$\lceil \log(b_R) \rceil$ ← Εύρεση της πρώτης

\+ $\lceil SC(A, r) / f_R \rceil - 1$ ← Εύρεση των υπόλοιπων

Αν το A υποψήφιο κλειδί;

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Επιλογή - συνθήκη ισότητας

E3 Χρήση πρωτεύοντος ευρετηρίου

Μπορεί να χρησιμοποιηθεί μόνο αν υπάρχει πρωτεύον ευρετήριο στο A

$HT_i + 1$ ← Εύρεση και μεταφορά της πρώτης

Αν το A δεν είναι υποψήφιο κλειδί -- ευρετήριο συστάδων

$HT_i + \lceil SC(A, R) / f_R \rceil$

Επιλογή - συνθήκη ισότητας

Ε4 Χρήση δευτερεύοντος ευρετηρίου

Μπορεί να χρησιμοποιηθεί μόνο αν υπάρχει δευτερεύον ευρετήριο στο Α

Αν το Α είναι υποψήφιο κλειδί

$HT_i + 1$

Αν το Α δεν είναι υποψήφιο κλειδί

$HT_i + SC(A, R)$ Στη χειρότερη περίπτωση κάθε εγγραφή που ικανοποιεί τη συνθήκη σε διαφορετικό block

Επιλογή - συνθήκη με σύγκριση

$\sigma_{A \le u}(R)$ ή $\sigma_{A \ge u}(R)$

Έστω ότι c πλειάδες ικανοποιούν τη συνθήκη

Γενικά $c = n_r / 2$

Έστω min, max, ομοιόμορφη κατανομή και $\sigma_{A \le u}(R)$

$$c = \begin{cases} 0 \text{ αν } u < \min \\ n_R \text{ αν } u \ge \max \\ n_R * [(u - \min) / (\max - \min)] \end{cases}$$

Δ1 Σειριακή αναζήτηση; Δ2 Δυαδική αναζήτηση;

Επιλογή - συνθήκη με σύγκριση

Ε5 Χρήση πρωτεύοντος ευρετηρίου

$A \ge u$

1. Χρήση ευρετηρίου για την εύρεση της πρώτης εγγραφής $A \ge u$
2. Σάρωση όλου του αρχείου ξεκινώντας από αυτήν την εγγραφή

$HT_i + \lceil c / f_R \rceil$

$A \le u$

Δε χρειάζεται ευρετήριο, γιατί;

Επιλογή - συνθήκη με σύγκριση

Ε6 Χρήση δευτερεύοντος ευρετηρίου

Σάρωση των φύλλων του δέντρου

$A \le u$ από την αρχή έως το u

$A \ge u$ από το u έως το τέλος

Αν $c = n_R / 2$, τότε (αν κάθε εγγραφή σε διαφορετικό block)

$HT_i + LB_i / 2 + n_R / 2$

Επιλογή - συνθήκη σύζευξης

$\sigma_{\Theta 1 \text{ AND } \Theta 2 \ldots \text{ AND } \Theta n}(R)$

Επιλεκτικότητα μιας συνθήκης:

$$\frac{\text{το πλήθος των εγγραφών (πλειάδων) που την ικανοποιούν}}{\text{το πλήθος των εγγραφών (πλειάδων) του αρχείου (σχέσης)}}$$

Επιλογή - συνθήκη σύζευξης

• Έστω $s_i = | \sigma_{\Theta i}(R) |$ -- επιλεκτικότητα: s_i / n_R

Αν Θi συνθήκη ισότητας σε ένα γνώρισμα υποψήφιο κλειδί $s_i = 1 / n_R$

Αν Θi συνθήκη ισότητας σε ένα γνώρισμα, ομοιόμορφη κατανομή, k διακριτές τιμές, $s_i = k / n_R$

• Αν οι συνθήκες είναι ανεξάρτητες, το μέγεθος του αποτελέσματος:

$$\frac{n_R * s_1 * s_2 * \ldots * s_n}{n_R^{\,n}}$$

Επιλογή - συνθήκη σύζευξης

Ε7 Συζευκτική επιλογή με χρήση ενός απλού ευρετηρίου

Υπάρχει διαδρομή προσπέλασης για ένα από τα γνωρίσματα που εμφανίζονται σε οποιαδήποτε απλή συνθήκη

Επιλογή του γνωρίσματος στην απλή συνθήκη με τη *μικρότερη* επιλεκτικότητα

Χρήση μιας από τις προηγούμενες μεθόδους για την ανάκτηση των εγγραφών που ικανοποιούν αυτήν την συνθήκη και έλεγχος για κάθε επιλεγμένη εγγραφή αν ικανοποιεί και τις υπόλοιπες συνθήκες

Επιλογή - συνθήκη σύζευξης

Ε8 Συζευκτική επιλογή με χρήση σύνθετου ευρετηρίου

Αν υπάρχει ευρετήριο στο συνδυασμό δύο ή περισσοτέρων γνωρισμάτων που εμφανίζονται σε οποιαδήποτε απλές συνθήκες

Επιλογή - συνθήκη σύζευξης

Ε9 Συζευκτική επιλογή με τομή δεικτών

Αν υπάρχουν ευρετήρια σε περισσότερα από ένα από τα γνωρίσματα

Επιλογή - συνθήκη διάζευξης

$$\sigma_{\Theta 1 \text{ OR } \Theta 2 \ldots \text{ OR } \Theta n}(R)$$

Αν κάποια από τις συνθήκες δεν έχει διαδρομή προσπέλασης -> σάρωση όλου του αρχείου

Συνένωση

$$R \bowtie_{R.A \text{ op } S.B} S$$

Σ1 Εμφωλευμένος (εσωτερικός - εξωτερικός) βρόγχος

Σ2 Χρήση μιας δομής προσπέλασης

Σ3 Ταξινόμηση-Συγχώνευση

Σ4 Συνένωση με κατακερματισμό

Συνένωση

Σ1 Εμφωλευμένος (εσωτερικός - εξωτερικός) βρόγχος

```
Για κάθε εγγραφή t της R
        Για κάθε εγγραφή s της S
                Αν t[A] op s[B] πρόσθεσε το t s στο αποτέλεσμα
```

Αγνοώντας την εγγραφή των blocks του αποτελέσματος

$$n_R * b_S + b_R$$

Συνένωση

```
Για κάθε block B_r της R
    Για κάθε block B_s της S
        Για κάθε εγγραφή t του B_r
            Για κάθε εγγραφή s του B_s
                Αν t[A] op s[B] πρόσθεσε το t s στο αποτέλεσμα
```

Αγνοώντας την εγγραφή των blocks του αποτελέσματος $b_R * b_S + b_R$

Συμφέρει η τοποθέτηση της *μικρότερης* σχέσης στο εξωτερικό βρόγχο

Συνένωση

Αν υπάρχουν $n_B > 2$ blocks στη μνήμη που μπορεί να χρησιμοποιηθούν για τον υπολογισμό της συνένωσης συμφέρει να διαβάζουμε τα blocks της σχέσης του εξωτερικού βρόγχου ανά $n_B - 1$

```
Για κάθε n_B - 1 block B_r της R
    Για κάθε block B_s της S
        Για κάθε εγγραφή t του B_r
            Για κάθε εγγραφή s του B_s
                Αν t[A] op s[B] πρόσθεσε το t s στο αποτέλεσμα
```

$\lceil (b_R / (n_B - 1)) \rceil * b_S + b_R$

Συνένωση

Σ2 Χρήση μιας δομής προσπέλασης

Η σχέση για την οποία υπάρχει ευρετήριο τοποθετείται στο εσωτερικό βρόγχο. Έστω ότι υπάρχει ευρετήριο για το γνώρισμα Β της σχέσης S

```
Για κάθε block B_r της R
    Για κάθε εγγραφή t του B_r
        Χρησιμοποιείσαι το ευρετήριο στο Β για να βρεις τις εγγραφές s
        της S τέτοιες ώστε t[A] op s[B]
```

$n_R * c + b_R$ όπου c το κόστος μιας επιλογής στο S

Συνένωση

Επιλεκτικότητα συνένωσης μιας σχέσης:

$$\frac{\text{το πλήθος των εγγραφών (πλειάδων) που επιλέγονται}}{\text{το πλήθος των εγγραφών (πλειάδων) του αρχείου (σχέσης)}}$$

• Σε ορισμένες περιπτώσεις μπορεί να δημιουργηθεί ένα ευρετήριο ειδικά για τη συνένωση

Συνένωση

Σ3 Ταξινόμηση - Συγχώνευση

```
Ταξινόμησε τις πλειάδες της R στο γνώρισμα A
Ταξινόμησε τις πλειάδες της S στο γνώρισμα B
i := 1;   j := 1;
while (i ≤ n_R and j ≤ n_S)
        if (R_i[A] < S_j[B])
                i := i + 1;
        if (R_i[A] > S_j[B])
                j := j + 1;
```


Συνένωση

```
else   (* R_i[A] = S_j[B] *)
    πρόσθεσε το R_i . S_j στο αποτέλεσμα
    k := j + 1;  (* γράψε και τις άλλες πλειάδες της S που ταιριάζουν, αν υπάρχουν *)
        while ((k ≤ n_S) and (R_i[A] = S_k[B]))
            πρόσθεσε το R_i . S_k στο αποτέλεσμα
            k := k + 1;
    m := i + 1;  (* γράψε και τις άλλες πλειάδες της R που ταιριάζουν, αν υπάρχουν *)
        while ((m ≤ n_R)  and (R_m[A] = S_j[B]))
            πρόσθεσε το R_m . S_j στο αποτέλεσμα
            k := k + 1;
    i := m; j := k;
```

Συνένωση

Αν αγνοήσουμε τη ταξινόμηση για τη σάρωση:

$b_R + b_S$

Ταξινόμηση: $b_R * \log(b_R) + b_S * \log(b_S)$

Συνένωση

Σ4 Συνένωση με κατακερματισμό

- χωρίζουμε με βάση μια συνάρτηση κατακερματισμού h τις πλειάδες της S και της R σε κάδους -- στον ίδιο κάδο αν $h(t_R[A]) = h(t_S[B])$
- δηλαδή οι πλειάδες με $t_R[A] = t_S[B]$ πέφτουν στον ίδιο κάδο άρα αρκεί να ελέγξουμε μεταξύ τους τις πλειάδες που πέφτουν στον ίδιο κάδο

Συνένωση

Κατακερμάτισε τις εγγραφές της R χρησιμοποιώντας την $h(t_R[A])$

Για κάθε εγγραφή t_S της S

k := $h(t_S[B])$

σύγκρινε το $t_S[B]$ με $t_{Ri}[A]$ για όλες τις εγγραφές t_{Ri} της R στον κάδο k

Χρησιμοποιούμε την μικρότερη σχέση για το πρώτο πέρασμα. Αν όλοι οι κάδοι που προκύπτουν χωράνε στη μνήμη, κόστος $b_R + b_S$

Συνένωση

Έστω ότι δε χωρούν όλοι οι κάδοι.

Έστω M καταχωρητές, N κάδοι, και η συνάρτηση h(k) = k mod N

(* Κατακερμάτισε τις εγγραφές της R χρησιμοποιώντας την $h(t_R[A])$ - στη μνήμη όλοι οι κάδοι του πρώτου κάδου και ένα block για τους υπόλοιπους *)

Για κάθε t_R της R

k := $h(t_R[A])$

if (k > 1 και το block του k-οστού κάδου είναι γεμάτο)

μετέφερε το block στη δίσκο

Συνένωση

(* στη μνήμη όλα τα blocks του 1ου κάδου - γράψε τα blocks των άλλων κάδων στο δίσκο *)

Για κάθε εγγραφή t_S της S

k := $h(t_S[B])$

αν k = 1 σύγκρινε το $t_S[B]$ με όλες τις εγγραφές t_{Ri} στον κάδο 1

αν k > 1, γράψε την εγγραφή στο δίσκο

Συνένωση

(* μετά το πρώτο βήμα - έχει γίνει συνένωση όλων των εγγραφών του πρώτου κάδου και στο δίσκο υπάρχουν N - 1 κάδοι της R και N - 1 κάδοι της S *)

i := 2; j := 1;

while (i ≤ N)

φέρε στη μνήμη όλα τα blocks του i-οστού κάδου

while (j ≤ # blocks του i-οστού κάδου του αρχείου S)

φέρε στη μνήμη το j-οστό block B_j του i-οστού κάδου του αρχείου S

Για κάθε εγγραφή t_S του B_j

σύγκρινε το $t_S[B]$ με όλες τις εγγραφές t_{Ri} στον κάδο i

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Πράξεις Συνόλων

Ταξινόμησε τις πλειάδες της R σε ένα γνώρισμα A

Ταξινόμησε τις πλειάδες της S στο ίδιο γνώρισμα

i := 1; j := 1;

while (i ≤ n_R and j ≤ n_S)

if (R_i[A] > S_j[B])

Τομή	**Ένωση**	**Διαφορά**
τίποτα	γράψε το S_j στο αποτέλεσμα	τίποτα

j := j + 1

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Πράξεις Συνόλων

else if (R_i[A] < S_j[B])

Τομή	**Ένωση**	**Διαφορά**
τίποτα	γράψε το R_i στο αποτέλεσμα	γράψε το R_i στο αποτέλεσμα

j := j + 1

else (* R_i[A] = S_j[B] *)

Τομή	**Ένωση**	**Διαφορά**

γράψε το R_i στο αποτέλεσμα
i: = i + 1;
j := j + 1;
	i: = i + 1;	i: = i + 1;
		j := j + 1;

Αλγόριθμοι Εκτέλεσης Βασικών Πράξεων

Πράξεις Συνόλων

Ένωση	**Διαφορά**

while (i ≤ n_R)
γράψε το R_i στο αποτέλεσμα
i: = i + 1;
while (j ≤ n_S)
γράψε το S_j στο αποτέλεσμα
j: = j + 1;
	while (i ≤ n_R)
	γράψε το R_i στο αποτέλεσμα
	i: = i + 1;

Βελτιστοποίηση Ερωτήσεων

select $A_1, A_2, \ldots, A_n$

from $R_1, R_2, \ldots, R_m$

where P

$\pi_{A1, A2, \ldots, An} (\sigma_P (R_1 \times R_2 \times \ldots \times R_m))$

Βελτιστοποίηση Ερωτήσεων

Δέντρο ερώτησης

Φύλλα: σχέσεις εισόδου

Εσωτερικοί κόμβοι: πράξεις της σχεσιακής άλγεβρας

Εκτέλεση δέντρου ερώτησης

Βελτιστοποίηση Ερωτήσεων

1. Διάσπαση των πράξεων επιλογής με συζευκτικές συνθήκες σε ακολουθίες πράξεων επιλογής

2. Μετατοπίζουμε την πράξη επιλογής όσο πιο κάτω επιτρέπεται από τα γνωρίσματα που περιλαμβάνονται στη συνθήκη

3. Επαναδιευθέτηση των φύλλων ώστε να εκτελούνται πρώτα οι σχέσεις που έχουν τις πιο περιοριστικές πράξεις επιλογής

Βελτιστοποίηση Ερωτήσεων

4. Συνδυασμός μιας πράξης καρτεσιανού γινομένου με μια πράξη επιλογής που ακολουθεί

5. Διάσπαση και μετακίνηση των λιστών προβολής όσο πιο κάτω γίνεται στο δέντρο

6. Εντοπισμός υποδέντρων με ομάδες πράξεων που μπορεί να εκτελεστούν με κοινό αλγόριθμο

Βελτιστοποίηση Ερωτήσεων

Επιπρόσθετες διαφάνειες από το βιβλίο:

Database Management Systems, 2nd ed.
Raghu Ramakrishnan and Johahnes Gehrke,
McGraw Hill, 1999

Βελτιστοποίηση Ερωτήσεων σε Κατανεμημένα Σχεσιακά ΣΔΒΔ

Περιεχόμενα

- Εισαγωγή
- Διάσπαση της Ερώτησης
- Data Localization
- Ολική Βελτιστοποίηση Ερωτήσεων
- Τοπική Βελτιστοποίηση Ερωτήσεων

Resources

- Principles of Distributed Database Systems, M. Tamer Ozsu, Patrick Valduriez, Second Edition, 1999.

- http://www.cs.ualberta.ca/~database/ddbook.html
- http://www.cs.ualberta.ca/~database/ddbook/notes/Query/ppframe.htm

Partitioning in Distributed Databases

- Global Relations
- Horizontal Partitioning
- Vertical Partitioning
- Hybrid Partitioning

Παραδείγματα Διαμερισμού

Global relations:

EMP(ENO,ENAME,TITLE)

ASG(ENO,PNO,RESP,DUR)

PRJ(PNO,PNAME,BUDGET)

SAL(TITLE,SALARY)

Horizontal Partitioning:

EMP_1:$\sigma_{ENO<=E3}$(EMP)

EMP_2:$\sigma_{E3<ENO<=E6}$(EMP)

EMP_1:$\sigma_{ENO>E6}$(EMP)

ASG_1:$\sigma_{ENO<=E3}$(ASG)

ASG_2:$\sigma_{ENO>E3}$(ASG)

Vertical Partitioning:

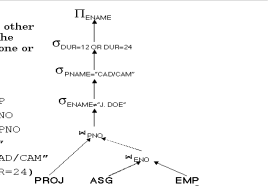
PRJ_1:$\pi_{PNO,PNAME}$(PRJ)

PRJ_2:$\pi_{PNO,BUDGET}$(PRJ)

Objectives of distributed Q.O.

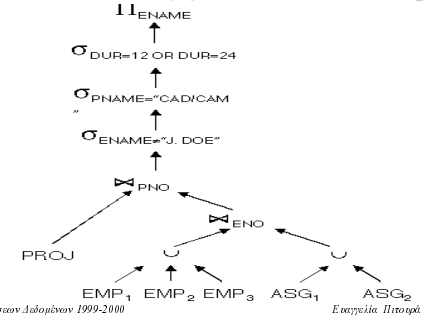
Find the names of employees other than J. Doe who worked on the CAD/CAM project for either one or two years.

```
SELECT   ENAME
FROM     PROJ, ASG, EMP
WHERE    ASG.ENO=EMP.ENO
AND      ASG.PNO=PROJ.PNO
AND      ENAME<>"J. Doe"
AND      PROJ.PNAME="CAD/CAM"
AND      (DUR=12 OR DUR=24)
```



Distributed DBMS © 1998 M. Tamer Özsu & Patrick Valduriez Page 7-9. 26

Objectives of distributed Q.O.



Objectives of distributed Q.O.

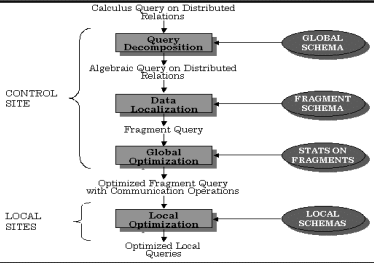
- types of optimization (deterministic, randomized)
- optimization timing (dynamic, static)
- statistics
- decision sites
- exploitation of network topology
- exploitation of replicated fragments
- use of semijoins

Query Optimization Objectives

- Minimize a cost function
  - I/O cost + CPU cost + communication cost
- These might have different weights in different distributed environments
- Wide area networks
  - ➠ communication cost will dominate
    - ◆ low bandwidth
    - ◆ low speed
    - ◆ high protocol overhead
  - ➠ most algorithms ignore all other cost components
- Local area networks
  - ➠ communication cost not that dominant
  - ➠ total cost function should be considered
- Can also maximize throughput

Distributed DBMS © 1998 M. Tamer Özsu & Patrick Valduriez Page 7-9. 7

Distributed Query Processing Methodology



Distributed DBMS © 1998 M. Tamer Özsu & Patrick Valduriez Page 7-9. 15

Step 1 – Query Decomposition

Input : Calculus query on global relations

- Normalization
  - manipulate query quantifiers and qualification
- Analysis
  - detect and reject "incorrect" queries
  - possible for only a subset of relational calculus
- Simplification
  - eliminate redundant predicates
- Restructuring
  - calculus query ⇒ algebraic query
  - more than one translation is possible
  - use transformation rules

Normalization

- Lexical and syntactic analysis
  - check validity (similar to compilers)
  - check for attributes and relations
  - type checking on the qualification
- Put into normal form
  - Conjunctive normal form
    $(p_{11} \vee p_{12} \vee \ldots \vee p_{1n}) \wedge \ldots \wedge (p_{m1} \vee p_{m2} \vee \ldots \vee p_{mn})$
  - Disjunctive normal form
    $(p_{11} \wedge p_{12} \wedge \ldots \wedge p_{1n}) \vee \ldots \vee (p_{m1} \wedge p_{m2} \wedge \ldots \wedge p_{mn})$
  - OR's mapped into union
  - AND's mapped into join or selection

Analysis

If the query graph is not connected, the query is wrong.

```
SELECT   ENAME,RESP
FROM     EMP, ASG, PROJ
WHERE    EMP.ENO = ASG.ENO
AND      PNAME = "CAD/CAM"
AND      DUR ≥ 36
AND      TITLE = "Programmer"
```

Simplification – Example

```
SELECT   TITLE
FROM     EMP
WHERE    EMP.ENAME = "J. Doe"
OR       (NOT(EMP.TITLE = "Programmer")
AND      (EMP.TITLE = "Programmer"
OR       EMP.TITLE = "Elect. Eng.")
AND      NOT(EMP.TITLE = "Elect. Eng."))
```

⇓

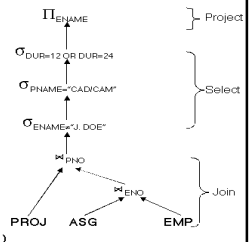
```
SELECT   TITLE
FROM     EMP
WHERE    EMP.ENAME = "J. Doe"
```


Restructuring

- Convert relational calculus to relational algebra
- Make use of query trees
- Example

  Find the names of employees other than J. Doe who worked on the CAD/CAM project for either 1 or 2 years.

```
SELECT   ENAME
FROM     EMP, ASG, PROJ
WHERE    EMP.ENO = ASG.ENO
AND      ASG.PNO = PROJ.PNO
AND      ENAME ≠ "J. Doe"
AND      PNAME = "CAD/CAM"
AND      (DUR = 12 OR DUR = 24)
```

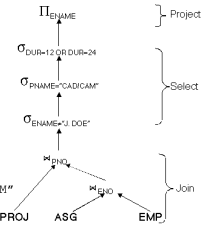


Example

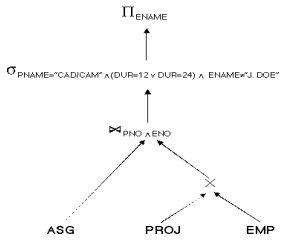
Recall the previous example:

Find the names of employees other than J. Doe who worked on the CAD/CAM project for either one or two years.

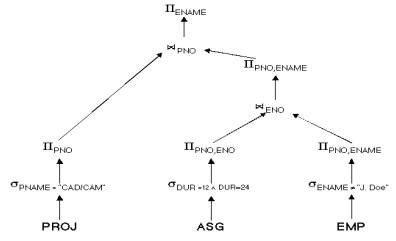
```
SELECT   ENAME
FROM     PROJ, ASG, EMP
WHERE    ASG.ENO=EMP.ENO
AND      ASG.PNO=PROJ.PNO
AND      ENAME≠"J. Doe"
AND      PROJ.PNAME="CAD/CAM"
AND      (DUR=12 OR DUR=24)
```



Equivalent Query



Restructuring



Step 2 – Data Localization

Input: Algebraic query on distributed relations

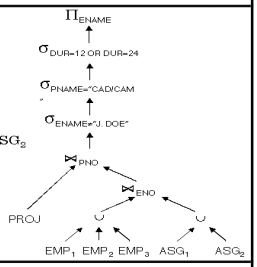
- Determine which fragments are involved
- Localization program
  - substitute for each global query its materialization program
  - optimize

Example

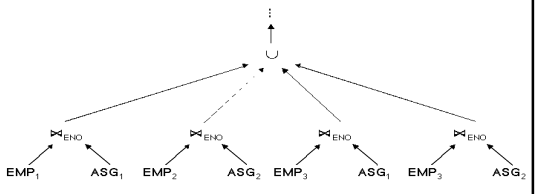
Assume

- EMP is fragmented into EMP_1, EMP_2, EMP_3 as follows:
  - $EMP_1=\sigma_{ENO\leq "E3"}(EMP)$
  - $EMP_2= \sigma_{"E3"<ENO\leq "E6"}(EMP)$
  - $EMP_3=\sigma_{ENO>"E6"}(EMP)$
- ASG fragmented into ASG_1 and ASG_2 as follows:
  - $ASG_1=\sigma_{ENO\leq "E3"}(ASG)$
  - $ASG_2=\sigma_{ENO>"E3"}(ASG)$

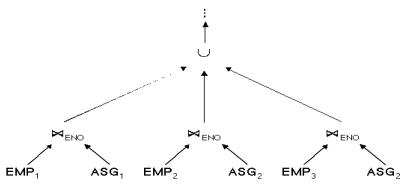
Replace EMP by $(EMP_1 \cup EMP_2 \cup EMP_3)$ and ASG by $(ASG_1 \cup ASG_2)$ in any query



Provides Parallellism



Eliminates Unnecessary Work



Reduction for HF

- Combine the rules already specified:
  - Remove empty relations generated by contradicting selections on horizontal fragments;
  - Remove useless relations generated by projections on vertical fragments;
  - Distribute joins over unions in order to isolate and remove useless joins.

Reduction for HF

Example

Consider the following hybrid fragmentation:

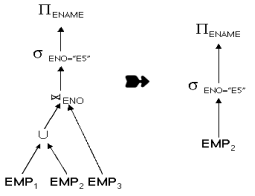
$EMP_1=\sigma_{ENO\leq "E4"}(\Pi_{ENO,ENAME}(EMP))$

$EMP_2=\sigma_{ENO>"E4"}(\Pi_{ENO,ENAME}(EMP))$

$EMP_3=\Pi_{ENO,TITLE}(EMP)$

and the query

```
SELECT   ENAME
FROM     EMP
WHERE    ENO="E5"
```



Step 3 – Global Query Optimization

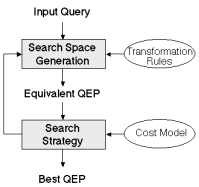
Input: Fragment query

- Find the *best* (not necessarily optimal) global schedule
  - Minimize a cost function
  - Distributed join processing
    - Bushy vs. linear trees
    - Which relation to ship where?
    - Ship-whole vs ship-as-needed
  - Decide on the use of semijoins
    - Semijoin saves on communication at the expense of more local processing.
  - Join methods
    - nested loop vs ordered joins (merge join or hash join)

Cost-Based Optimization

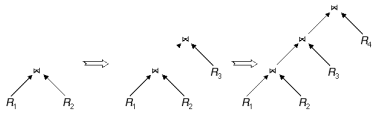
- Solution space
  - The set of equivalent algebra expressions (query trees).
- Cost function (in terms of time)
  - I/O cost + CPU cost + communication cost
  - These might have different weights in different distributed environments (LAN vs WAN).
  - Can also maximize throughput
- Search algorithm
  - How do we move inside the solution space?
  - Exhaustive search, heuristic algorithms (iterative improvement, simulated annealing, genetic,…)

Query Optimization Process

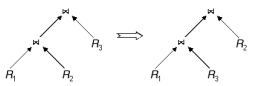


Search Strategies

- Deterministic

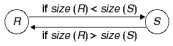


- Randomized



Join Ordering

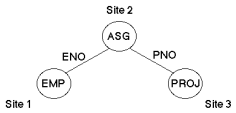
- Consider two relations only



- Multiple relations more difficult because too many alternatives.
  - Compute the cost of all alternatives and select the best one.
    - Necessary to compute the size of intermediate relations which is difficult.
  - Use heuristics

Join Ordering – Example

Consider
PROJ $\bowtie_{PNO}$ASG $\bowtie_{ENO}$EMP



Join Ordering – Example

Execution alternatives:

1. EMP → Site 2
   Site 2 computes EMP'=EMP⋈ASG
   EMP' → Site 3
   Site 3 computes EMP'⋈PROJ

2. ASG → Site 1
   Site 1 computes EMP'=EMP⋈ASG
   EMP' → Site 3
   Site 3 computes EMP'⋈PROJ

3. ASG → Site 3
   Site 3 computes ASG'=ASG⋈PROJ
   ASG' → Site 1
   Site 1 computes ASG' ⋈ EMP

4. PROJ → Site 2
   Site 2 computes PROJ'=PROJ⋈ASG
   PROJ' → Site 1
   Site 1 computes PROJ' ⋈ EMP

5. EMP → Site 2
   PROJ → Site 2
   Site 2 computes EMP' ⋈PROJ⋈ ASG

Semijoin of R with S is the subset of tuples of R that participate in the join of R with S

Semijoin Algorithms

- Consider the join of two relations:
  - $R[A]$ (located at site 1)
  - $S[A]$ (located at site 2)
- Alternatives:
  1. Do the join $R \bowtie_A S$
  2. Perform one of the semijoin equivalents

$$R \bowtie_A S \Leftrightarrow (R \ltimes_A S) \bowtie_A S$$
$$\Leftrightarrow R \bowtie_A (S \ltimes_A R)$$
$$\Leftrightarrow (R \ltimes_A S) \bowtie_A (S \ltimes_A R)$$

Semijoin Algorithms

- Perform the join
  - send R to Site 2
  - Site 2 computes $R \bowtie_A S$
- Consider semijoin $(R \ltimes_A S) \bowtie_A S$
  - $S' \leftarrow \prod_A(S)$
  - $S' \rightarrow$ Site 1
  - Site 1 computes $R' = R \ltimes_A S'$
  - $R' \rightarrow$ Site 2
  - Site 2 computes $R' \bowtie_A S$

Semijoin is better if

$$size(\prod_A(S)) + size(R \ltimes_A S)) < size(R)$$

R* Algorithm

- Cost function includes local processing as well as transmission
- Considers only joins
- Exhaustive search
- Compilation
- Published papers provide solutions to handling horizontal and vertical fragmentations but the implemented prototype does not

Input: Query QT
Output: miminum cost strategy
begin
  for each fragment in QT, decide best access plan;
  for all the possible *states*
    build *strategy* and compute cost;
  obtain best strategy;
  perform local optimization at each site;
end.

state: = orders of joins. No. of states = $n!$

strategy = see next

R* Algorithm

Performing joins
- Ship whole
  - larger data transfer
  - smaller number of messages
  - better if relations are small
- Fetch as needed *semijoin internal with external*
  - number of messages = O(cardinality of external relation)
  - data transfer per message is minimal
  - better if relations are large and the selectivity is good

R* Algorithm – Vertical Partitioning & Joins

1. Move outer relation tuples to the site of the inner relation
   (a) Retrieve outer tuples
   (b) Send them to the inner relation site
   (c) Join them as they arrive

   Total Cost = cost(retrieving qualified outer tuples)
   + no. of outer tuples fetched * cost(retrieving qualified inner tuples)
   + msg. cost * (no. outer tuples fetched * avg. outer tuple size) / msg. size

R* Algorithm – Vertical Partitioning & Joins

2. Move inner relation to the site of outer relation
   cannot join as they arrive; they need to be stored

   Total Cost = cost(retrieving qualified outer tuples)
   + no. of outer tuples fetched * cost(retrieving matching inner tuples from temporary storage)
   + cost(retrieving qualified inner tuples)
   + cost(storing all qualified inner tuples in temporary storage)
   + msg. cost * (no. of inner tuples fetched * avg. inner tuple size) / msg. size

R* Algorithm – Vertical Partitioning & Joins

3. Move both inner and outer relations to another site

   Total cost = cost(retrieving qualified outer tuples)
   + cost(retrieving qualified inner tuples)
   + cost(storing inner tuples in storage)
   + msg. cost * (no. of outer tuples fetched * avg. outer tuple size) / msg. size
   + msg. cost * (no. of inner tuples fetched * avg. inner tuple size) / msg. size
   + no. of outer tuples fetched * cost(retrieving inner tuples from temporary storage)

R* Algorithm – Vertical Partitioning & Joins

4. Fetch inner tuples as needed
   (a) Retrieve qualified tuples at outer relation site
   (b) Send request containing join column value(s) for outer tuples to inner relation site
   (c) Retrieve matching inner tuples at inner relation site
   (d) Send the matching inner tuples to outer relation site
   (e) Join as they arrive

   Total Cost = cost(retrieving qualified outer tuples)
   + msg. cost * (no. of outer tuples fetched)
   + no. of outer tuples fetched * (no. of inner tuples fetched * avg. inner tuple size * msg. cost / msg. size)
   + no. of outer tuples fetched * cost(retrieving matching inner tuples for one outer value)

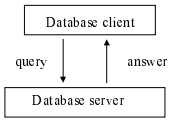
Step 4 – Local Optimization

Input: Best global execution schedule

- Select the best access path
- Use the centralized optimization techniques

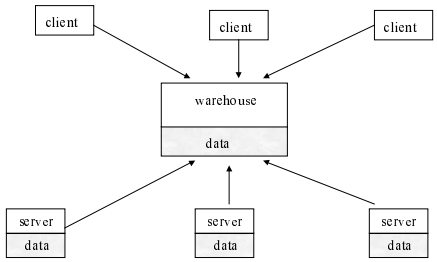
Επεξεργασία Ερωτήσεων για Ημι-δομημένα Δεδομένα

Αρχιτεκτονικές

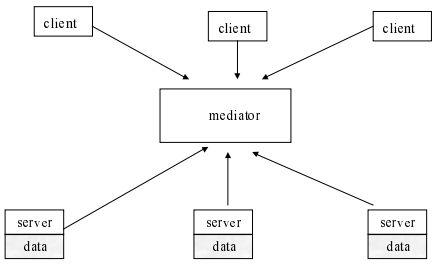


Συνήθως από τον ίδιο vendor

Αρχιτεκτονικές



Αρχιτεκτονικές



Επεξεργασία Ερωτήσεων

Τα ίδια βασικά στάδια

1. Μετάφραση - ένα σχέδιο εκτέλεσης
2. Βελτιστοποίηση
3. Μηχανή Εκτέλεσης

Κατανομή

Έλλειψη σχήματος

Αποθήκευση

1. Κείμενο
2. Σχεσιακή Βάση Δεδομένων
3. Αντικειμενο-στραφή Βάση Δεδομένων
4. Αυτό-οργάνωση - Υβριδική αποθήκευση

Αποθήκευση

Text File

```
<family id = "o1">
  <person id = "p1"> <name Joan </name>
      <age> 36 </age>
      <profession> database administrator </profession>
      <hobby> gardening </hobby>
      <cellular> 555-6234 </cellular>
  </person>
<person id = "p2"> <name John </name>
      <age> 38 </age>
      <profession> systems administrator </profession>
      <beeper> 555-3322 </beeper>
  </person>
```


Αποθήκευση

Σχεσιακή Βάση Δεδομένων

Δύο σχέσεις ref(src, label, dst) Val(oid, value)

```
<family id = 'o1'>
  <person id = 'p1'> <name Joan </name>
      <age> 36 </age>
      <profession> database administrator
</profession>
      <hobby> gardening </hobby>
      <cellular> 555-6234 </cellular>
  </person>
<person id = 'p2'> <name John </name>
      <age> 38 </age>
      <profession> systems administrator </profession>
      <beeper> 555-3322 </beeper>
  </person>
```

src	label	dst
'root'	'family'	'o1'
'o1'	'person'	'p1'
'o1'	'person'	'p2'
'p1'	'name'	'p11'
'p1'	'age'	'p12'
'p1'	'profession'	'p13'
'p1'	'hobby'	'p14'
'p1'	'cellular'	'p15'
'p2'	'name'	'p21'
'p2'	'age'	'p22'
'p2'	'profession'	'p23'
'p2'	'beeper'	'p24'

oid	value
'p11'	'Joan'
p11'	36
'p13'	'database admin'
'p14'	'gardening'
'p15'	'555-6234'
'p21'	'John'
'p22'	38
'o23'	'systems admin'
p24'	'555-3322'

Αποθήκευση

Σχεσιακή Βάση Δεδομένων

```
<family id = 'o1'>
  <person id = 'p1'> <name Joan </name>
      <age> 36 </age>
      <profession> database administrator </profession>
      <hobby> gardening </hobby>
      <cellular> 555-6234 </cellular>
  </person>
<person id = 'p2'> <name John </name>
      <age> 38 </age>
      <profession> systems administrator </profession>
      <beeper> 555-3322 </beeper>
  </person>
```

```
select X
from family.person.hobby X
```


Αποθήκευση

Σχεσιακή Βάση Δεδομένων

src	label	dst
'root'	'family'	'o1'
'o1'	'person'	'p1'
'o1'	'person'	'p2'
'p1'	'name'	'p11'
'p1'	'age'	'p12'
'p1'	'profession'	'p13'
'p1'	'hobby'	'p14'
'p1'	'cellular'	'p15'
'p2'	'name'	'p21'
'p2'	'age'	'p22'
'p2'	'profession'	'p23'
'p2'	'beeper'	'p24'

Refs

oid	value
'p11'	'Joan'
36	
'p13'	'database admin'
'p14'	'gardening'
'555-6234'	
'p21'	'John'
38	
'p23'	'systems admin'
p24'	'555-3322'

Value

```
select X
from family.person.hobby X

select v.value
from Refs r1, Refs r2, Refs r3, Value v
where  r1.src ="root" and
       r1.label = "family and
       r1.dst = r2.src and
       r2.label= person and
       r2.dst =r3.src and
       r3.label = hobby and
       r3.dst = v.oid
```

4-way join! index?

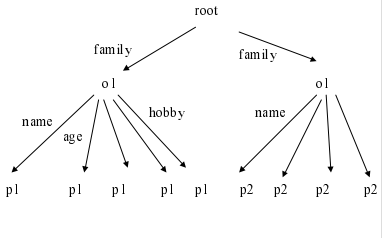
Αποθήκευση

Σχεσιακή Βάση Δεδομένων

src	label	dst
'root'	'family'	'o1'
'o1'	'person'	'p1'
'o1'	'person'	'p2'
'p1'	'name'	'p11'
'p1'	'age'	'p12'
'p1'	'profession'	'p13'
'p1'	'hobby'	'p14'
'p1'	'cellular'	'p15'
'p2'	'name'	'p21'
'p2'	'age'	'p22'
'p2'	'profession'	'p23'
'p2'	'beeper'	'p24'

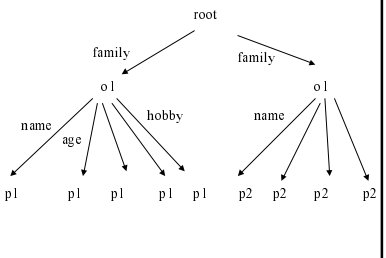
Refs

Breadth-first search!



Σχεσιακή Βάση Δεδομένων

src	label	dst
'root'	'family'	'o1'
'o1'	'person'	'p1'
'p1'	'name'	'p11'
'p1'	'name'	'p11'
'p1'	'age'	'p12'
'p1'	'profession'	'p13'
'p1'	'hobby'	'p14'
'p1'	'cellular'	'p15'
'o1'	'person'	'p2'
'p2'	'name'	'p21'
'p2'	'age'	'p22'
'p2'	'profession'	'p23'
'p2'	'beeper'	'p24'

Refs

Depth-first search!