Αρχιτεκτονικές Συστημάτων Βάσεων Δεδομένων

1. Συγκεντρωτικά συστήματα ΒΔ
2. Συστήματα πελάτη - εξυπηρετητή (client-server)
3. Παράλληλα συστήματα ΒΔ
4. Κατανεμημένα συστήματα ΒΔ

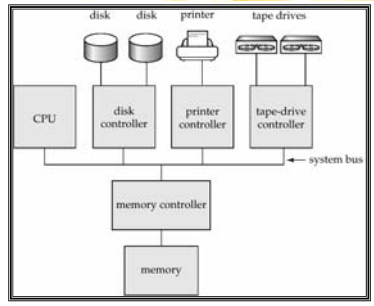
{

1. Συστήματα Ομοτίμων
2. Αποθήκες Δεδομένων

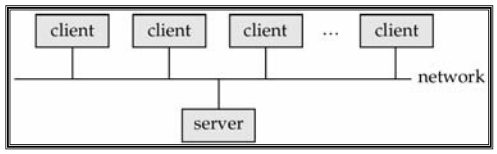
}

Εκτελούνται σε *ένα μόνο* υπολογιστικό σύστημα και δεν αλληλεπιδρούν με άλλα υπολογιστικά συστήματα

- Γενικής χρήσης υπολογιστικό σύστημα: μία ή το πολύ λίγες CPUs και ένας αριθμός device controllers που συνδέονται διαμέσου ενός κοινού διαύλου (bus), ο οποίος παρέχει πρόσβαση σε διαμοιρασμένη μνήμη.
- Σύστημα ενός χρήστη (π.χ. προσωπικός υπολογιστής ή σταθμός εργασίας): συνήθως έχει μία μόνο CPU και ένα ή δύο σκληρούς δίσκους, το λειτουργικό σύστημα μπορεί να υποστηρίξει μόνο έναν χρήστη.
- Σύστημα πολλαπλών χρηστών: περισσότεροι δίσκοι, μεγαλύτερη μνήμη, πολλαπλές CPUs, και λειτουργικό σύστημα πολλαπλών χρηστών. Εξυπηρετεί έναν μεγάλο αριθμό χρηστών οι οποίοι είναι συνδεδεμένοι στο σύστημα μέσω τερματικών. Συχνά καλείται σύστημα εξυπηρετητή (server system).



- Το σύστημα εξυπηρετητή (server) ικανοποιεί αιτήσεις που παράγονται από *m* συστήματα πελατών (clients):



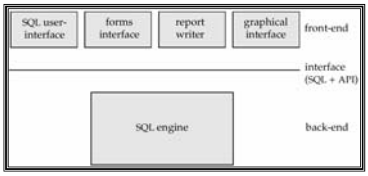
Πλεονεκτήματα

- Απλή υλοποίηση
- Αξιοποίηση «ακριβών» εξυπηρετητών
- Περιβάλλον χρήστη προσαρμοσμένο στις ανάγκες του

- Πλεονεκτήματα της αντικατάστασης των mainframes με *δίκτυα* σταθμών εργασίας ή προσωπικών υπολογιστών συνδεδεμένων σε back-end εξυπηρετητές:
  - Καλύτερη λειτουργικότητα σε σχέση με το κόστος
  - Ευελιξία στην ανάθεση πόρων (resource allocation)
  - Καλύτερα user interfaces
  - Ευκολότερη συντήρηση

Συστήματα πελάτη – εξυπηρετητή (συν.)

- Η λειτουργικότητα της βάσης δεδομένων μπορεί να χωριστεί σε:
  - Back-end: διαχειρίζεται τις δομές προσπέλασης στα δεδομένα, την αξιολόγηση και βελτιστοποίηση των επερωτήσεων, τον έλεγχο συγχρονισμού (concurrency control) και ανάκτησης (recovery).
  - Front-end: αποτελείται από εργαλεία για δημιουργία φορμών, αναφορών και γραφικών user interfaces
- Η διασύνδεση μεταξύ front-end και back-end γίνεται με την SQL ή μέσω ενός προγράμματος εφαρμογής (API).



Συστήματα πελάτη – εξυπηρετητή (συν.)

- Οι εξυπηρετητές μπορούν να κατηγοριοποιηθούν σε δύο είδη:
  - **Εξυπηρετητές δοσοληψιών (transaction servers)** οι οποίοι χρησιμοποιούνται κυρίως σε Σχεσιακά ΣΔΒΔ, και
  - **Εξυπηρετητές δεδομένων (data servers)**, οι οποίοι συνήθως χρησιμοποιούνται σε Αντικειμενοστραφή ΣΔΒΔ

Εξυπηρετητές Δοσοληψιών

- Ονομάζονται και **query servers ή SQL server** systems.
- Οι πελάτες στέλνουν αιτήματα στον εξυπηρετητή, όπου εκτελούνται οι δοσοληψίες, και τα αποτελέσματα επιστρέφονται στον πελάτη.
- Τα αιτήματα εκφράζονται σε SQL και μεταφέρονται στον εξυπηρετητή μέσω ενός μηχανισμού κλήσης απομακρυσμένης διαδικασίας (remote procedure call - RPC). Πολλές κλήσεις RPC μπορούν να σχηματίσουν μία συλλογική δοσοληψία.
- Το πρότυπο **Open Database Connectivity (ODBC)** είναι ένα API, γραμμένο σε C από την Microsoft, για διασύνδεση σε έναν εξυπηρετητή, αποστολή αιτημάτων σε SQL και λήψη αποτελεσμάτων
- Το πρότυπο **JDBC** είναι παρόμοιο με το ODBC, γραμμένο σε Java

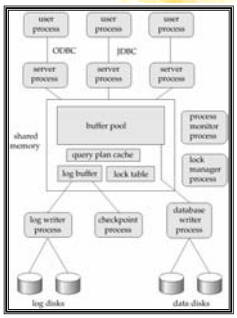
Εξυπηρετητές Δοσοληψιών (συν.)

- Ένας τυπικός εξυπηρετητής δοσοληψιών αποτελείται από πολλαπλές διεργασίες πρόσβασης δεδομένων σε διαμοιρασμένη μνήμη.
  - Οι διεργασίες λαμβάνουν τις επερωτήσεις των χρηστών, σε μορφή δοσοληψιών, τις εκτελούν και στέλνουν πίσω τα αποτελέσματα.
  - Οι διεργασίες μπορεί να είναι **πολυνηματικές (multithreaded)**, δηλαδή μία μοναδική διεργασία μπορεί να εκτελέσει αρκετές επερωτήσεις ταυτόχρονα.

    Η διαμοιρασμένη μνήμη περιλαμβάνει διαμοιρασμένα δεδομένα
    Όλες οι διεργασίες μπορούν να προσπελάσουν τη διαμοιρασμένη μνήμη
    Για να εξασφαλιστεί ότι δύο διεργασίες δεν έχουν πρόσβαση στα ίδια δεδομένα ταυτόχρονα, τα συστήματα πελάτη-εξυπηρετητή εκτελούν αμοιβαίο αποκλεισμό (mutual exclusion) π.χ. με semaphores λειτουργικού συστήματος

Θέμα: άνοιγμα cursor (δρομέα!) και η φόρτωση πλειάδων μία τη φορά έχει μεγάλο κόστος

Εξυπηρετητές Δοσοληψιών (συν.)



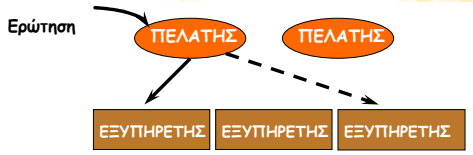
Εξυπηρετητές Δεδομένων

- *Μεταφέρονται δεδομένα στις μηχανές πελατών* όπου και εκτελείται η επεξεργασία τους, και κατόπιν μεταφέρονται τα αποτελέσματα πίσω στη μηχανή του εξυπηρετητή.
- Χρησιμοποιούνται στα τοπικά δίκτυα LAN, όπου η σύνδεση ανάμεσα στον πελάτη και στον εξυπηρετητή είναι υψηλής ταχύτητας, οι μηχανές πελάτη είναι συγκρίσιμες σε δύναμη επεξεργασίας με τη μηχανή του εξυπηρετητή και οι εργασίες που πρόκειται να εκτελεστούν είναι «βαριές» υπολογιστικά (compute intensive).
- Θέματα που υπεισέρχονται:
  - Μεταφορά σελίδας (page-shipping) ή στοιχείου (item-shipping)
  - Κλειδώματα
  - Δεδομένα στην cache

- Δεδομένα στην cache
  - Δεδομένα μπορεί να γίνουν cached στον πελάτη ακόμα και στο ενδιάμεσο των δοσοληψιών
  - Απαραίτητος είναι ο έλεγχος των δεδομένων προτού χρησιμοποιηθούν. Ο έλεγχος μπορεί να γίνει τη στιγμή που υπάρχει αίτηση κλειδώματος στα δεδομένα αυτά

- Μεταφορά σελίδας ή Μεταφορά στοιχείου
  - Μικρότερη μονάδα μεταφοράς ⇒ περισσότερα μηνύματα
  - Αξίζει να γίνει προανάκτηση (prefetching) στοιχείων σχετικών με αυτά για τα οποία γίνεται η αίτηση
  - Η μεταφορά σελίδας μπορεί να θεωρηθεί σαν μία μορφή προανάκτησης

- Κλειδώματα
  - Λόγω των καθυστερήσεων των μηνυμάτων, συχνά υπάρχουν υπερβολικές αιτήσεις και παροχές κλειδωμάτων από τον εξυπηρετητή
  - Κλειδώματα σε ένα προανακτημένο στοιχείο μπορεί ναι είναι P{called-back} από τον εξυπηρετητή και να επιστραφούν εάν το στοιχείο αυτό δεν έχει χρησιμοποιηθεί.

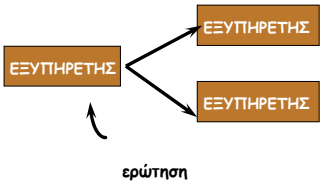


⌘ Πελάτη-Εξυπηρέτη

Ο πελάτης στέλνει (ships) την ερώτηση σε μία πλευρά. Η επεξεργασία της ερώτησης γίνεται στον εξυπηρέτη.

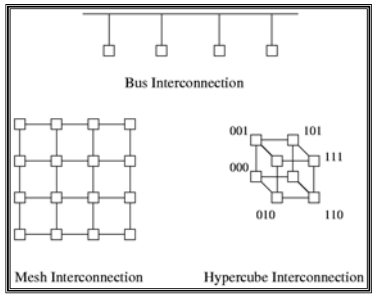
⌘ Συνεργαζόμενων εξυπηρετητών

Μια ερώτηση εκτελείται σε πολλές πλευρές

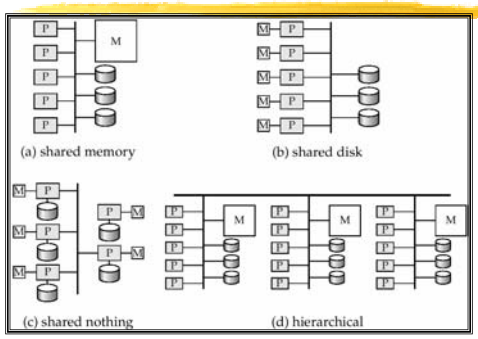


- Τα **παράλληλα συστήματα** ΒΔ αποτελούνται από ένα σύνολο πολλών επεξεργαστών και πολλών δίσκων δια-συνδεδεμένων σε ένα *γρήγορο* δίκτυο.
- Μία coarse-grain παράλληλη μηχανή αποτελείται από ένα μικρό αριθμό ισχυρών επεξεργαστών
- Μία μαζικά παράλληλη (massively parallel) ή fine grain παράλληλη μηχανή χρησιμοποιεί χιλιάδες μικρότερους επεξεργαστές

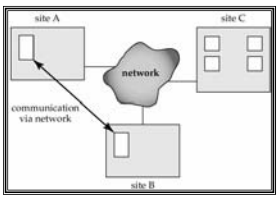
- Δύο κύρια μέτρα απόδοσης:
  - Ρυθμός εξόδου ή ολοκλήρωσης (throughput): ο αριθμός των εργασιών που μπορούν να ολοκληρωθούν σε συγκεκριμένο χρονικό διάστημα
  - Χρόνος απόκρισης (response time) : ο χρόνος που απαιτείται για να ολοκληρωθεί *μία* εργασία από τη στιγμή που θα υποβληθεί



- **Διαμοιρασμένη μνήμη (shared memory)**: οι επεξεργαστές μοιράζονται κοινή μνήμη
- **Διαμοιρασμένο σύστημα δίσκων (shared disk)**: οι επεξεργαστές μοιράζονται κοινό σύστημα δίσκων
- **«Τίποτα» διαμοιρασμένο (shared nothing)**: οι επεξεργαστές ΔΕΝ μοιράζονται ούτε κοινή μνήμη ούτε κοινό δίσκο
- **Ιεραρχική**: υβριδική αρχιτεκτονική – συνδυασμός των ανωτέρω



- Τα δεδομένα είναι απλωμένα σε πολλές μηχανές, που ονομάζονται «τόποι» (**sites**) ή «κόμβοι» (**nodes**), οι οποίες μηχανές είναι διασυνδεδεμένες μέσω *ενός δικτύου* (LAN ή WAN)
- Τα δεδομένα είναι κοινά για χρήστες που τα προσπελαύνουν από πολλές μηχανές



Ομογενή: Κάθε πλευρά τρέχει τον ίδιο τύπο ΣΔΒΔ

Ετερογενή : Κάθε πλευρά τρέχει διαφορετικά ΣΔΒΔ (διαφορετικά σχεσιακά ΣΔΒΔ ή και μη-σχεσιακά ΣΔΒΔ) – ενδιάμεσο λογισμικό

Κατανεμημένες Βάσεις Δεδομένων

Γενικές Ιδιότητες

Ανεξαρτησία των Κατανεμημένων Δεδομένων: Οι χρήστες δεν πρέπει να γνωρίζουν που βρίσκονται τα δεδομένα (επέκταση της Φυσικής και Λογικής Ανεξαρτησίας)

Ατομικότητα των Κατανεμημένων Συναλλαγών

- Ομογενείς Κατανεμημένες ΒΔ
  - Το ίδιο λογισμικό και σχήμα ΒΔ σε όλους τους κόμβους, τα δεδομένα είναι μοιρασμένα μεταξύ των κόμβων.
  - Στόχος: να φαίνεται σαν μια ενιαία ΒΔ, κρύβοντας την κατανομή της σε κόμβους
- Ετερογενείς Κατανεμημένες ΒΔ ή Σύστημα Πολλαπλών ΒΔ (multidatabases)
  - Διαφορετικό λογισμικό και σχήμα ΒΔ από κόμβο σε κόμβο
  - Στόχος: η ολοκλήρωση διαφορετικών ΒΔ που ήδη υπάρχουν
- Διάκριση μεταξύ τοπικών (*local*) και γενικών (*global*) δοσοληψιών
  - Μια τοπική δοσοληψία προσπελαύνει δεδομένα στον κόμβο από τον οποίο προήλθε.
  - Μια γενική δοσοληψία είτε προσπελαύνει δεδομένα σε κόμβο διαφορετικό από αυτόν από τον οποίο προήλθε είτε προσπελαύνει δεδομένα σε πολλούς κόμβους.

- Χαρακτηριστικά Κατανεμημένων ΒΔ
  - Διαμοίραση δεδομένων: οι χρήστες ενός τόπου μπορούν να προσπελάσουν δεδομένα που βρίσκονται σε άλλο κόμβο
  - Αυτονομία: κάθε κόμβος μπορεί να έχει έλεγχο στα «δικά του» δεδομένα.
  - Πλεονασμός (redundancy): τα ίδια δεδομένα μπορούν να επαναληφθούν σε διαφορετικούς κόμβους (διπλο-αποθήκευση) ώστε το σύστημα να λειτουργήσει ακόμη και αν κάποιος τόπος «πέσει»
  - Μειονέκτημα: η πρόσθετη πολυπλοκότητα που απαιτείται για τον συντονισμό των κόμβων (κόστος ανάπτυξης λογισμικού, αυξημένη πιθανότητα bugs, επιπλέον επεξεργασία/συντήρηση κλπ.)

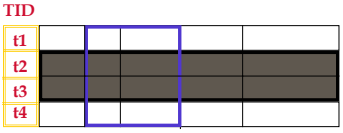
1. **Αποθήκευση Δεδομένων (κατατεμαχισμός, αντίγραφα (ομοιότυπα)**
2. Επεξεργασία Ερωτήσεων
3. Επεξεργασία Δοσοληψιών

Σε ένα κατανεμημένο ΣΔΒΔ πολλοί κόμβοι – πως μοιράζουμε τα δεδομένα στους κόμβους

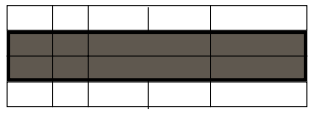
Έστω μια σχέση *r*. Πως θα μπορούσε να αποθηκευτεί σε μια κατανεμημένη βάση δεδομένων;

- Τεμαχισμός (τμηματοποίηση) (fragmentation): «κόβουμε» τη σχέση σε τμήματα και τα τοποθετούμε σε διαφορετικούς κόμβους
- Αντίγραφα (ομοιότυπα)
- Τεμαχισμός και αντίγραφα
- Τοποθέτηση (allocation)

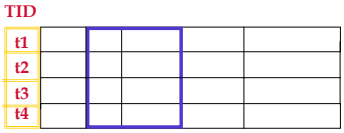
- Τεμαχισμός (fragmentation)
  - Οριζόντιος (horizontal)
  - Κάθετος(vertical)
  - Υβριδικός



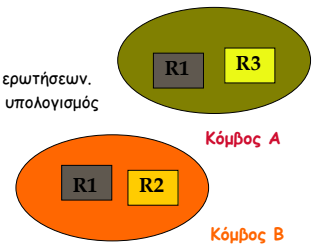
- Οριζόντιος Τεμαχισμός (horizontal fragmentation)
  - Με βάση κάποια συνθήκη -- πράξη (σ)
  - Συνήθως ξένα
  - Πως παίρνουμε πάλι την αρχική σχέση;



- Κάθετος Τεμαχισμός (vertical fragmentation)
  - με βάση ποια πράξη (προβολή π)
  - πως παίρνουμε την αρχική σχέση
  - χωρίς απώλειες στη συνένωση; tids.



- Αντίγραφα -- Ομοιοτυπία (Replication)
  - Διαθεσιμότητα
  - Γρηγορότερος υπολογισμός ερωτήσεων.
  - Σύγχρονος και Ασύγχρονος υπολογισμός
    - ενημέρωση αντιγράφων.



- Διατήρηση πληροφορίας για την κατανομή των δεδομένων στους κόμβους

Θέματα

Μοναδικά ονόματα

Αρχιτεκτονικές/Δομή Καταλόγου

- Όνομα για κάθε αντίγραφο σε κάθε κόμβο. Διατήρηση τοπικής αυτονομίας
  - **<local-name, birth-site>**
- Συγκεντρωτικές κατάλογος (και αντίγραφά του)
- Κατάλογος σε κάθε κόμβο: (τοπικός κατάλογος) Περιγράφει κάθε αντικείμενο (τεμάχιο, αντίγραφο) που είναι αποθηκευμένα στον κόμβο **+** (κατάλογος καταγωγής) Κρατά πληροφορία για τα αντίγραφα των σχέσεων που δημιουργήθηκαν στον κόμβο.
  - Εύρεση σχέσης, αναζήτηση στον κατάλογο στον κόμβο που δημιουργήθηκε.
  - Ο κόμβος που δημιουργήθηκε η σχέση δεν αλλάζει ακόμα και αν η σχέση μετακινηθεί.

1. Αποθήκευση Δεδομένων (τεμαχισμός, αντίγραφα -ομοιότυπα)
2. **Επεξεργασία Ερωτήσεων**
3. Επεξεργασία Δοσοληψιών

⌘ Εκτός του I/O κόστους (D), το κόστος μετάδοσης δεδομένων (T) στο δίκτυο

⌘ Πιθανό κέρδος από το ό,τι διάφοροι κόμβοι μπορεί να επεξεργάζονται μια ερώτηση παράλληλα (ταυτόχρονα) : διαφορά συνολικού χρόνου υπολογισμού και χρόνου απόκρισης

Παράδειγμα

Sailors(sid, sname, rating, age)

Reserves(sid, bid, day, rname)

Reserves: 40 bytes ανά πλειάδα - μία σελίδα 100 πλειάδες – 1.000 σελίδες – 100.000 πλειάδες

Sailors: 50 bytes ανά πλειάδα - μία σελίδα 80 πλειάδες – 500 σελίδες – 40.000 πλειάδες


```
SELECT AVG(S.age)
FROM Sailors S
WHERE S.rating > 3
   AND S.rating < 7
```

⌘ Οριζόντιος τεμαχισμός: Οι πλειάδες με rating < 5 στη Shanghai, >= 5 στοTokyo.

- Πρέπει να υπολογίσουμε SUM(age), COUNT(age) και στους δύο κόμβους.
- Αν το WHERE περιείχε μόνο S.rating > 6, θα αρκούσε ο υπολογισμός σε έναν μόνο κόμβο.


```
SELECT ...
FROM Sailors S
WHERE S.rating > κ1
   AND S.rating < κ2
```

⌘ Οριζόντιος τεμαχισμός:

⌘ Οι πλειάδες S1 με rating < 5 στη Shanghai, οι πλειάδες S2 με rating >= 5 στοTokyo.

Μετασχηματισμοί της ερώτησης

$\sigma_{S.rating > κ1 \text{ and } S.rating < κ2}(S) =$ Οριζόντιος τεμαχισμός

$\sigma_{S.rating > κ1 \text{ and } S.rating < κ2}(S1 \cup S2) =$

$\sigma_{S.rating > κ1 \text{ and } S.rating < κ2}(S1) \cup$

$\sigma_{S.rating > κ1 \text{ and } S.rating < κ2}(S2)$ Εκτέλεση στη Shanghai

$\sigma_{S.rating > κ1 \text{ and } S.rating < κ2}(\sigma_{S.rating < 5}(S)) \cup$

$\sigma_{S.rating > κ1 \text{ and } S.rating < κ2}(\sigma_{S.rating >= 5}(S))$

Εκτέλεση στο Tokyo

Δυνατές απλοποιήσεις;


```
SELECT AVG(S.age)
FROM Sailors S
WHERE S.rating > 3
   AND S.rating < 7
```

⌘ Κάθετος τεμαχισμός : *sid* και *rating* στη Shanghai, *sname* and *age* στοTokyo, *tid* και στους δύο κόμβους.

- Πρέπει πρώτα να *ξανά-σχηματιστεί η σχέση με συνένωση* στο *tid*, και μετά να υπολογιστεί η ερώτηση


```
SELECT ...
FROM Sailors S
WHERE S.rating > κ1
   AND S.rating < κ2
```

⌘ Κάθετος κατατεμαχισμός:

⌘ *sid* και *rating* στη Shanghai, *sname* and *age* στοTokyo, *tid* και στους δύο κόμβους.

Μετασχηματισμοί της ερώτησης

$\sigma_{S.rating > κ1 \text{ and } S.rating < κ2}(S) =$ Κάθετος κατατεμαχισμός

$\sigma_{S.rating > κ1 \text{ and } S.rating < κ2}(S1 * S2) =$

```
SELECT AVG(S.age)
FROM Sailors S
WHERE S.rating > 3
    AND S.rating < 7
```

⌘ Αντίγραφα: αντίγραφα της σχέσης Sailors και στους δύο κόμβους.
- Επιλογή του κόμβου με βάση το τοπικό κόστος και το κόστος για την μεταφορά του αποτελέσματος

Γενική παρατήρηση: Συνυπολογισμός τους κόστους για τη μεταφορά του αποτελέσματος στον κόμβο στον οποίον υποβλήθηκε η ερώτηση

D κόστος εγγραφής/ανάγνωσης σελίδας

T κόστος μεταφοράς σελίδας

⌘ Μεταφορά όταν χρειάζεται, Εμφωλευμένος βρόγχος με τη σχέση S στον εξωτερικό βρόγχο :
- **Κόστος:** 500 D + 500 * 1000 (D+T)
- Αν χρειάζεται μεταφορά αποτελέσματος: Κόστος Μεταφοράς
  Μέγεθος αποτελέσματος
  100.000 πλειάδες μεγέθους 40+50 = 90 bytes → 2.273 σελίδες

⌘ Μεταφορά όταν χρειάζεται, Εμφωλευμένος βρόγχος με τη σχέση S στον εξωτερικό βρόγχο :
- Αν η ερώτηση δεν είχε υποβληθεί στο Λονδίνο πρέπει να προσθέσουμε και το κόστος μεταφοράς του αποτελέσματος στον κόμβο που αρχικά υποβλήθηκε η ερώτηση
- Ο υπολογισμός μπορεί επίσης να γίνει στο Παρίσι

⌘ Μεταφορά όλης της σχέσης σε έναν κόμβο: μεταφορά της R στο Λονδίνο
- **Κόστος:** 1000 T + 4500 D (με ταξινόμηση/συγχώνευση: κόστος = 3*(500+1000)D)
- Αν το μέγεθος του αποτελέσματος είναι πολύ μεγάλο, μπορεί να συμφέρει να μεταφέρουμε και τις δύο σχέσεις στον κόμβο υποβολής της ερώτησης και να υπολογίσουμε τη συνένωση εκεί

• Ημι-συνένωση

• Ιδέα: Αποφυγή μεταφοράς *όλης* της σχέσης R στο Λονδίνο – αλλά μεταφορά μόνο των πλειάδων που συνενώνονται με πλειάδες της S

•Πως; Πρέπει να καθορίσουμε ποιες είναι αυτές οι πλειάδες

• Ημι-συνένωση

Βήμα 1: Στο Λονδίνο, προβολή (project) S στις στήλες του join
• Μεταφορά του αποτελέσματος στο Παρίσι
Βήμα 2: Στο Παρίσι, συνένωση της προβολής S με τη R.
Το αποτέλεσμα ονομάζεται **Ελάττωση (reduction)** της R σε σχέση με τη S.
• Μεταφορά της ελάττωσης της σχέσης R στο Λονδίνο
Βήμα 3: Στο Λονδίνο, συνένωση S με την ελάττωση της R.

Παράδειγμα

Sailors(sid, sname, rating, age)
Reserves(sid, bid, day, rname)

Reserves: 40 bytes ανά πλειάδα - μία σελίδα 100 πλειάδες – 1.000 σελίδες – 100.000 πλειάδες
Sailors: 50 bytes ανά πλειάδα - μία σελίδα 80 πλειάδες – 500 σελίδες – 40.000 πλειάδες

Πεδίο sid 10 bytes

Βήμα 1: 500 D (σάρωση S) + 100 D (προσωρινό αποτέλεσμα) (διπλότιμα;)
100 T (για τη μεταφορά)
Βήμα 2: 3 (100 + 1.000) = 3.300 D (για τον υπολογισμό της ελάττωσης)
20% συμμετοχή τότε 200 D (μεταφορά)
Βήμα 3: 3 (200+500) D = 2.100 D

• Ημι-συνένωση

• Ιδέα: Αποφυγή μεταφοράς *όλης* της σχέσης R στο Λονδίνο -- με την επιβάρυνση του υπολογισμού και της μεταφοράς της προβολής της S και του υπολογισμού της συνένωσης της προβολής με τη R – κέρδος; Πότε;

• Ιδιαίτερα χρήσιμο όταν υπάρχει μια συνθήκη επιλογής στη σχέση S, και η απάντηση πρέπει να δοθεί στο Λονδίνο

Σύζευξη Bloom

(Βήμα 1) Αντί της προβολής, στέλνουμε ένα δυαδικό διάνυσμα μεγέθους k!

Κατακερματίζουμε κάθε πλειάδα σε έναν αριθμό 0 έως k-1

(Βήμα 2) Ο υπολογισμός της ελάττωσης χρησιμοποιεί το διάνυσμα

⌘Παραλληλισμός στην εκτέλεση της συνένωσης

R1 Join R2 Join R3 … Join Rn

⌘ Κατασκευή ολικού πλάνου – αντικατάσταση σχέσεων με τα τεμάχια τους -> προτεινόμενα τοπικά πλάνα

⌘ Υπολογισμός όλων των πλάνων - επιλογή αυτού με το μικρότερο κόστος
- Διαφορά 1: Κόστος επικοινωνίας
- Διαφορά 2: Τοπική αυτονομία κάθε κόμβου (δηλαδή, κάθε κόμβος αποφασίζει τοπικά τον καλύτερο τρόπο να υπολογίσει κάθε τοπική υπο-ερώτηση
- Διαφορά 3: Νέοι τρόποι υπολογισμού κατανεμημένης συνένωσης

1. Αποθήκευση Δεδομένων (διαμέριση, αντίγραφα (ομοιότυπα)
2. Επεξεργασία Ερωτήσεων
3. **Επεξεργασία Δοσοληψιών**

1. Διαχείριση αντιγράφων
2. Κατανεμημένη Διαχείριση Δοσοληψιών
3. Κατανεμημένη Ανάρρωση από Αποτυχίας

Πολλαπλά αντίγραφα δεδομένων

- **Σύγχρονη Ενημέρωση Αντιγράφων**: Όλα τα αντίγραφα μιας τροποποιημένης σχέσης (τεμάχια) πρέπει να τροποποιηθούν *πριν* την επικύρωση της δοσοληψίας
  - Η κατανομή των δεδομένων είναι αδιαφανής (transparent) στους χρήστες
- **Ασύγχρονη Ενημέρωση Αντιγράφων** : Τα αντίγραφα μιας σχέσης ενημερώνονται *περιοδικά*, διαφορετικά αντίγραφα μπορεί στα ενδιάμεσα διαστήματα να μην είναι ενημερωμένα (out of synch)
  - Οι χρήστες γνωρίζουν την κατανομή των δεδομένων
  - Πολλά εμπορικά συστήματα ακολουθούν αυτήν την προσέγγιση

- **Μέθοδος της Πλειοψηφίας:** Μια δοσοληψία πρέπει να γράψει την πλειοψηφία των αντιγράφων για να τροποποιήσει ένα αντικείμενο -- πρέπει να διαβάσει αρκετά αντίγραφα έτσι ώστε να δει τουλάχιστον ένα καινούριο (πιο πρόσφατο) αντίγραφο
  - Για παράδειγμα, 10 αντίγραφα, 7 να γραφούν για τροποποίηση, 4 αντίγραφα για ανάγνωση
  - Κάθε αντίγραφο έχει έναν αριθμό έκδοσης
  - Πρόβλημα γιατί συνήθως οι αναγνώσεις είναι πιο συχνές από τις εγγραφές

- **Read-any Write-all:** Σχετικά με τη μέθοδο της πλειοψηφίας, οι εγγραφές είναι αργές και οι αναγνώσεις πιο γρήγορες
  - Η πιο συνηθισμένη προσέγγιση στη σύγχρονη ενημέρωση αντιγράφων
- Η επιλογή της τεχνικής καθορίζει ποια κλειδιά πρέπει να ζητηθούν

- Πριν επικυρωθεί μια δοσοληψία (που περιλαμβάνει τροποποιήσεις) πρέπει να αποκτήσει κλειδιά σε όλα τα τροποποιημένα αντίγραφα
  - Στέλνει αιτήσεις για κλειδιά σε απομακρυσμένους κόμβους και ενώ περιμένει για απάντηση κρατά τα άλλα κλειδιά
  - Αν ένας κόμβος ή μια σύνδεση πέσει (αποτύχει), η δοσοληψία δε μπορεί να επικυρωθεί μέχρι να επιστρέψουν σε λειτουργία
  - Ακόμα και αν δεν υπάρξει αποτυχία, η επικύρωση πρέπει να ακολουθήσει ένα ακριβό πρωτόκολλο επικύρωσης (commit protocol) με πολλά μηνύματα.
- $\Rightarrow$ Ασύγχρονη ενημέρωση των αντιγράφων

- Επιτρέπει την επικύρωση (commit) μιας δοσοληψίας πριν την ενημέρωση όλων των αντιγράφων (και οι αναγνώσεις μπορεί να διαβάζουν μόνο ένα αντίγραφο) να επικυρωθεί.
  - Οι χρήστες γνωρίζουν ποιο αντίγραφο διαβάζουν και ότι τα αντίγραφα μπορεί να μην είναι ενημερωμένα για κάποια μικρά χρονικά διαστήματα
- Δύο προσεγγίσεις: Πρωτεύον Κόμβος (primary site) και Peer-to-Peer : Βάσει του αριθμού των αντιγράφων που μπορούν να ενημερωθούν (master copies)

Πρωτεύουσα Αντιγραφή

- **Τροποποίηση ακριβώς ενός** αντιγράφου μιας σχέσης (αντικειμένου) χαρακτηρίζεται ως **πρωτεύον ή master** αντίγραφο. Αντίγραφα σε άλλους κόμβους δε μπορούν να τροποποιηθούν άμεσα.
  - Κοινοποιείται ποιο είναι το πρωτεύον αντίγραφο.
  - Οι άλλοι κόμβοι εγγράφονται σε (τεμάχια) της σχέσης, είναι **δευτερεύοντα** αντίγραφα.

- Βασικό Θέμα: Πως οι τροποποιήσεις στο πρωτεύον αντίγραφο μεταδίδονται στα δευτερεύοντα αντίγραφα

Γίνεται σε δύο βήματα
- Εντοπισμός των αλλαγών
- Εφαρμογή των αλλαγών

Εντοπισμός των Αλλαγών

- **Log-Based Εντοπισμός**: Το ημερολόγιο του συστήματος (log) που κρατείται για ανάρρωση χρησιμοποιείται για τη δημιουργία ενός πίνακα που καλείται Change Data Table (CDT).
  - Αν αυτό γίνεται όταν το log tail γράφεται στο δίσκο τότε με κάποιο τρόπο να σβηστούν οι τροποποιήσεις που οφείλονται σε δοσοληψίες που απορρίπτονται αργότερα.
- **Διαδικαστικός Εντοπισμός**: Μια διαδικασία καλείται αυτόματα (συνήθως παίρνει απλώς ένα στιγμιότυπο).
- Ο Log-Based εντοπισμός είναι καλύτερος (φθηνότερος, πιο γρήγορος) αλλά βασίζεται σε proprietary λεπτομέρειες του log.

Εφαρμογή των Αλλαγών

- Η διαδικασία της εφαρμογής των αλλαγών στο δευτερεύοντα κόμβο δέχεται περιοδικά από τον πρωτεύοντα κόμβο ένα στιγμιότυπο ή τις αλλαγές και τροποποιεί το αντίγραφο.
  - Η περίοδος είτε ορίζεται από το χρήστη ή την εφαρμογή είτε βασίζεται στο χρόνο.
- Log-Based εντοπισμός και συνεχής εφαρμογή των αλλαγών ελαχιστοποιεί την καθυστέρηση στη μετάδοση των αλλαγών.
- Διαδικαστικός εντοπισμός και εφαρμογή που καθορίζεται από τις εφαρμογές είναι ο πιο ευέλικτος τρόπος για τη διαχείριση των αλλαγών.

Ομοτίμων

- Περισσότερα από ένα master αντίγραφα ενός αντικειμένου (αντίγραφα που μπορεί να τροποποιηθούν)
- Τροποποιήσεις κάποιου master αντιγράφου πρέπει κάπως να μεταδοθούν στα άλλα αντίγραφα
- Όταν δυο master αντίγραφα τροποποιούνται με συγκρουόμενο τρόπο, πρέπει να διευθετηθούν οι συγκρούσεις (π.χ., κόμβος 1: η ηλικία του Joe άλλαξε σε 35; Κόμβος 2: σε 36)
- Προτιμότερη όταν δεν συμβαίνουν συγκρούσεις:
  - Π.χ., κάθε master κόμβος κατέχει ένα ξένο τεμάχιο
  - Π.χ., τροποποίηση δικαιωμάτων που κατέχονται από ένα master τη φορά

Πως γίνεται η διαχείριση των κλειδιών για δεδομένα στους διαφορετικούς κόμβους;

- **Κεντρικά (Centralized):** Ένας κόμβος είναι υπεύθυνος για τη διαχείριση όλων των κλειδιών.
  - Αποτυχία του κόμβου;
- **Πρωτεύον αντίγραφο:** Η διαχείριση των κλειδιών για ένα αντικείμενο γίνεται στον κόμβο όπου βρίσκεται το πρωτεύον αντίγραφο του αντικειμένου
  - Η ανάγνωση απαιτεί προσπέλαση και στον κόμβο που διαχειρίζεται τα κλειδιά και του κόμβου όπου είναι αποθηκευμένο το αντικείμενο.
- **Πλήρως κατανεμημένη:** Η διαχείριση του κλειδιού για ένα αντίγραφο γίνεται στον κόμβο που βρίσκεται το αντίγραφο
  - Η εγγραφή απαιτεί κλειδιά σε όλους τους κόμβους

⌘ Κάθε κόμβος διατηρεί έναν τοπικό γράφο αναμονής (waits-for graph)

⌘ Ένα ολικό αδιέξοδο μπορεί να δημιουργηθεί ακόμα και αν οι τοπικοί κόμβοι δεν περιέχουν κύκλους:

- Τρεις λύσεις:
  - Κεντρικοποιημένη (- *περιοδική* - αποστολή όλων των τοπικών γράφων σε έναν κόμβο)
  - Ιεραρχική (οργάνωση κόμβων σε μια ιεραρχία και αποστολή γράφων στον γονέα στην ιεραρχία)
    - Υπόθεση ότι συχνά αδιέξοδα ανάμεσα σε συσχετιζόμενους κόμβους
    - Περιοδική αποστολή
  - Timeout (απόρριψη μιας δοσοληψίας αν περιμένει για πάρα πολύ).
    - Αυτονομία

  Αδιέξοδα φαντάσματα (λόγω της καθυστέρησης κατασκευής του γράφου)

  Αντίστοιχοι deadlock prevention

⌘ Δυο νέα θέματα:

- Δυο νέα ειδών αποτυχιών, π.χ., συνδέσεις και απομακρυσμένοι κόμβοι.
- Αν τμήματα μιας δοσοληψίας εκτελούνται σε διαφορετικούς κόμβους όλες ή καμία πρέπει να επικυρωθούν ⇒ πρωτόκολλο επικύρωσης.

⌘ Διατηρείται ένα log σε κάθε κόμβο, όπως και στα κεντρικά ΣΔΒΔ στο οποίο καταγράφονται και οι πράξεις του πρωτοκόλλου

Πρωτόκολλο Επικύρωσης Δύο Φάσεων (Two Phase Commit Protocol 2PC)

⌘ Ο κόμβος από τον οποίο ξεκίνησε μια δοσοληψία είναι ο συντονιστής (coordinate) -- οι υπόλοιποι κόμβοι που συμμετέχουν στην εκτέλεση της δοσοληψίας ορίζονται ως συμμετέχοντες ή εξαρτημένοι (subordinates).

Όταν μια δοσοληψία πρέπει να επικυρωθεί:

Φάση 1

1. Ο συντονιστής στέλνει ένα μήνυμα **prepare** (προετοιμασίας) σε όλους τους συμμετέχοντες κόμβους.
2. Κάθε συμμετέχον κόμβος force-writes μια εγγραφή **abort** ή **prepare** στο log και μετά στέλνει ένα μήνυμα **no** ή **yes** στον συντονιστή.

   (δηλαδή στα κατανεμημένα συστήματα υπάρχουν ειδικές εγγραφές στο log σχετικές με το πρωτόκολλο επικύρωσης)

Φάση 2

- Αν ο συντονιστής δεχθεί μηνύματα **yes** από όλους, force-writes μια εγγραφή **commit** στο log και μετά στέλνει ένα μήνυμα **commit** σε όλους τους συμμετέχοντες κόμβους. Αλλιώς, force-writes μια εγγραφή **abort** στο log και μετά στέλνει ένα μήνυμα **abort**.
- Κάθε συμμετέχον κόμβος βάσει του μηνύματος που δέχεται, force-writes μια εγγραφή **abort/commit** στο log, και μετά στέλνει ένα μήνυμα **ack** στον συντονιστή.
- Ο συντονιστής αφού λάβει όλα τα acks γράφει μια εγγραφή **end** στο log.

- Δυο γύροι επικοινωνίας: πρώτα ψηφοφορία (voting) και μετά τερματισμός. Και οι δυο ξεκινούν από τον συντονιστή.
- Οποιοσδήποτε κόμβος μπορεί να αποφασίσει να απορρίψει μια δοσοληψία.
- Κάθε μήνυμα αντιστοιχεί σε μια απόφαση του αποστολέα: για την αντιμετώπιση των αποτυχιών αυτή η απόφαση καταγράφεται στο τοπικό log (πριν σταλεί)
- Όλες οι εγγραφές στο log που αφορούν το πρωτόκολλο επικύρωσης περιέχουν το id της δοσοληψίας Xactid και το id του κόμβου του συντονιστή Coordinatorid. Οι εγγραφές abort/commit στο συντονιστή περιέχουν τα ids όλων των κόμβων που συμμετέχουν
- Για βελτίωση της απόδοσης, ο συντονιστής μπορεί να μη στείλει μηνύματα abort στους κόμβους που ψήφισαν όχι.

- Μηνύματα Ack χρησιμοποιούνται για να ειδοποιήσουν τον συντονιστή ότι μπορεί να «ξεχάσει» τη δοσοληψία, μέχρι να λάβει αυτό το μήνυμα πρέπει να κρατά τη δοσοληψία στον πίνακα δοσοληψιών
- Αν ο συντονιστής αποτύχει αφού στείλει μηνύματα prepare αλλά πριν γράψει τις εγγραφές commit/abort, τότε κάνει τη δοσοληψία abort
- Αν η δοσοληψία δεν περιλαμβάνει ενημερώσεις, τότε δεν έχει σημασία αν γίνει commit ή abort

Επαναφορά από Αποτυχία

- Αν για μια δοσοληψία T έχουμε μια εγγραφή commit ή abort αλλά δεν έχουμε μια εγγραφή end, πρέπει η T να γίνει ή redo ή undo
  - Αν αυτός ο κόμβος είναι ο συντονιστής για την T, θα πρέπει να συνεχίσει να στέλνει μηνύματα **commit/abort** μέχρι να λάβει **acks** από όλους τους συμμετέχοντες κόμβους

Επαναφορά από Αποτυχία

- Αν για μια δοσοληψία T έχουμε μια εγγραφή prepare αλλά δεν έχουμε μια εγγραφή commit/abort, αυτός ο κόμβος πρέπει να είναι συμμετέχον κόμβος
  - Επαναληπτικά επικοινωνεί με το συντονιστή για να βρει την κατάσταση της T,
  - γράφει εγγραφές **commit/abort** στο log
  - redo/undo T
  - γράφει εγγραφές **end** στο log

Επαναφορά από Αποτυχία

- Αν για μια δοσοληψία T δεν έχουμε ούτε μια εγγραφή prepare, abort ή undo T.
  - Αυτός ο κόμβος μπορεί να είναι ο συντονιστής. Σε αυτήν την περίπτωση μπορεί οι συμμετέχοντες να στείλουν μηνύματα.
    Τότε διαπιστώνει ο κόμβος ότι ήταν ο συντονιστής και απαντά αρνητικά

- Αν ο συντονιστής για την T αποτύχει, οι συμμετέχοντες κόμβοι που έχουν ψηφίσει yes δεν μπορούν να αποφασίσουν αν θα πρέπει να επικυρώσουν ή να ακυρώσουν την T μέχρι να αναρρώσει ο συντονιστής.
  - T είναι αποκλεισμένη - blocked.
  - Ακόμα και αν όλοι οι συμμετέχοντες κόμβοι γνωρίζουν ο ένας τον άλλο (επιπλέον overhead στα μηνύματα prepare) δεν μπορούν να αποφασίσουν εκτός αν ένας από αυτούς έχει ψηφίσει no

- Αν ένας κόμβος δεν αποκρίνεται σε έναν κόμβο x κατά τη διάρκεια του πρωτοκόλλου επικύρωσης επειδή είτε ο κόμβος είτε η σύνδεση έχει αποτύχει:
  - Αν ο κόμβος x είναι ο συντονιστής, πρέπει να απορρίψει την T.
  - Αν ο κόμβος x είναι συμμετέχον, και δεν έχει ακόμα ψηφίσει yes, πρέπει να απορρίψει την T.
  - Αν ο κόμβος x είναι συμμετέχον, και έχει ψηφίσει yes, πρέπει να περιμένει μέχρι να αποκριθεί ο συντονιστής

Βελτιώσεις

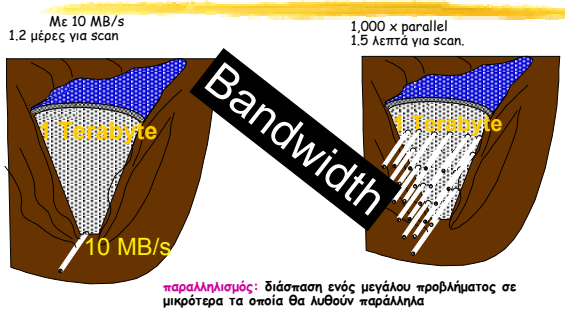
1. Όταν ο συντονιστής aborts T, μπορεί να την απομακρύνει από τον πίνακα συναλλαγών
2. Όταν ένας συμμετέχον αποφασίσει abort, δε χρειάζεται να περιμένει ack

Και μια σειρά αντίστοιχες βελτιστοποιήσεις

- Τα κατανεμημένα ΣΔΒΔ προσφέρουν αυτονομία σε κάθε κόμβο και κατανεμημένη διαχείριση. Χρειάζονται νέες τεχνικές αποθήκευσης, διαχείρισης του καταλόγου, ελέγχου συνδρομικότητας και ανάκαμψης.

Παράλληλες Βάσεις Δεδομένων

1. Συγκεντρωτικά συστήματα ΒΔ
2. Συστήματα πελάτη - εξυπηρετητή (client-server)
3. Παράλληλα συστήματα ΒΔ
4. Κατανεμημένα συστήματα ΒΔ

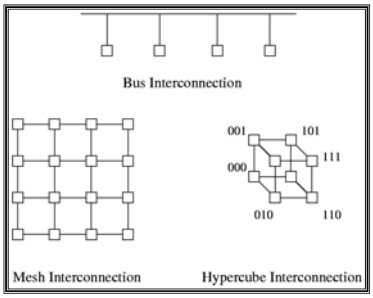


παραλληλισμός: διάσπαση ενός μεγάλου προβλήματος σε μικρότερα τα οποία θα λυθούν παράλληλα

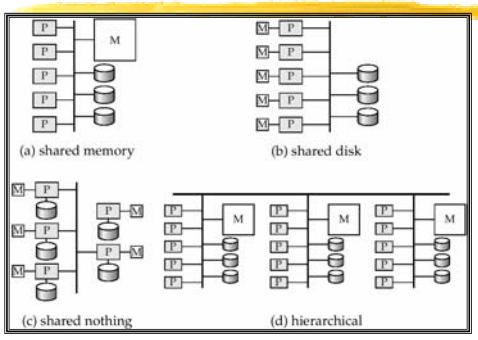
- Τα παράλληλα συστήματα ΒΔ αποτελούνται από ένα σύνολο πολλών επεξεργαστών και πολλών δίσκων διασυνδεδεμένων σε ένα *γρήγορο δίκτυο*.
- Μία coarse-grain παράλληλη μηχανή αποτελείται από ένα μικρό αριθμό ισχυρών επεξεργαστών (χοντρόκοκκοι)
- Μία μαζικά παράλληλη (massively parallel) ή fine grain παράλληλη μηχανή χρησιμοποιεί χιλιάδες μικρότερους επεξεργαστές (λεπτόκοκκοι)

- Βελτίωση της απόδοσης μέσου της υλοποίησης παράλληλων λειτουργιών
- Κατανομή δεδομένων με στόχο τη βελτίωση της απόδοσης

Σε κατανεμημένο σύστημα:
Τα δεδομένα είναι σε φυσικό επίπεδο αποθηκευμένα σε διαφορετικούς κόμβους – αυτονομία κόμβων



- **Διαμοιρασμένη μνήμη (shared memory)**: οι επεξεργαστές μοιράζονται κοινή μνήμη
- **Διαμοιρασμένο σύστημα δίσκων (shared disk)**: οι επεξεργαστές μοιράζονται κοινό σύστημα δίσκων
- **«Τίποτα» διαμοιρασμένο (shared nothing)**: οι επεξεργαστές ΔΕΝ μοιράζονται ούτε κοινή μνήμη ούτε κοινό δίσκο
- **Ιεραρχική** : υβριδική αρχιτεκτονική – συνδυασμός των ανωτέρω



Διαμοιρασμένη μνήμη (shared memory)

- Ιδιαίτερα αποδοτική επικοινωνία μεταξύ των επεξεργαστών. Όλοι προσπελαύνουν τα δεδομένα στην (κοινή) μνήμη χωρίς να χρειάζεται να τα μετακινήσουν από εκεί.
- *Πρόβλημα:* δεν κλιμακώνεται πέρα από 32 ή 64 επεξεργαστές καθώς προκαλείται κυκλοφοριακή συμφόρηση (bottleneck) στον δίαυλο μνήμης. Συνήθως χρησιμοποιείται για χαμηλό βαθμό παραλληλισμού (4-8).

Διαμοιρασμένο σύστημα δίσκων (shared disk)

- Ο δίαυλος μνήμης δεν είναι πια πρόβλημα αλλά τώρα η κυκλοφοριακή συμφόρηση συμβαίνει στη σύνδεση με το σύστημα δίσκων
- Ανοχή σφαλμάτων (**fault-tolerance**). Εάν αποτύχει ένας επεξεργαστής, οι υπόλοιποι αναλαμβάνουν τις εργασίες του αφού η ΒΔ είναι αποθηκευμένη σε δίσκους προσπελάσιμους από όλους
- Κλιμάκωση σε κάπως μεγαλύτερο αριθμό επεξεργαστών
- Τα πρώτα εμπορικά συστήματα: IBM Sysplex και DEC clusters (τώρα Compaq) που έτρεχαν Rdb (τώρα Oracle Rdb)

«Τίποτα» διαμοιρασμένο (shared nothing)

- Ένας «κόμβος» αποτελείται από έναν επεξεργαστή, μια μνήμη και ένα σύστημα δίσκων. Οι κόμβοι είναι συνδεδεμένοι μεταξύ τους. Κάθε κόμβος παίζει το ρόλο του εξυπηρετητή για τα δεδομένα που είναι αποθηκευμένα στο σύστημα δίσκων του.
- Η κλιμάκωση μπορεί να γίνει σε χιλιάδες επεξεργαστές
- Πρόβλημα: κόστος επικοινωνίας μεταξύ επεξεργαστών και η προσπέλαση δεδομένων σε μη τοπικούς δίσκους (απαιτείται λογισμικό αποστολής δεδομένων από το έναν κόμβο στον άλλο)
- Εμπορικά συστήματα: Teradata, Tandem, Oracle-n CUBE

Ιεραρχική αρχιτεκτονική

Το ανώτερο επίπεδο μπορεί να ακολουθεί την αρχιτεκτονική «τίποτα» διαμοιρασμένου ενώ εσωτερικά κάθε κόμβος μπορεί να είναι διαμοιρασμένης μνήμης ή διαμοιρασμένου συστήματος δίσκων (σε πολλά επίπεδα)

Shared Nothing
- Teradata: 400 nodes
- Tandem: 110 nodes
- IBM / SP2 / DB2: 128 nodes
- Informix/SP2 48 nodes
- ATT & Sybase ? nodes

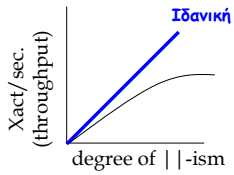
Shared Disk
- Oracle 170 nodes
- DEC Rdb 24 nodes

Shared Memory
- Informix 9 nodes
- RedBrick ? nodes

- Δύο κύρια μέτρα απόδοσης:
  - Ρυθμός εξόδου ή ολοκλήρωσης (throughput): ο αριθμός των εργασιών που μπορούν να ολοκληρωθούν σε συγκεκριμένο χρονικό διάστημα
  - Χρόνος απόκρισης (response time) : ο χρόνος που απαιτείται για να ολοκληρωθεί μία εργασία από τη στιγμή που θα υποβληθεί

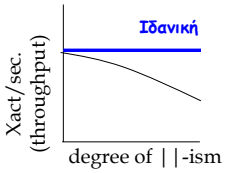
Επιτάχυνση (Speed-Up)

Περισσότεροι πόροι σημαίνει αναλογικά λιγότερος χρόνος για ένα *δοσμένη* ποσότητα δεδομένων



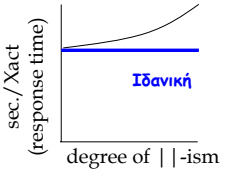
Κλιμάκωση (Scale-Up)

Αν οι πόροι αυξάνονται σε αναλογία με το μέγεθος δεδομένων, ο χρόνος είναι σταθερός

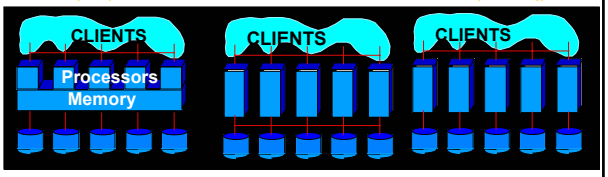


Κλιμάκωση (Scale-Up)

Αν οι πόροι αυξάνονται σε αναλογία με το μέγεθος δεδομένων, ο χρόνος είναι σταθερός



Διαμοιραζόμενης Μνήμης Shared Memory (SMP) | Διαμοιραζόμενου Δίσκου Shared Disk | Τίποτα διαμοιρασμένο Shared Nothing (δικτυακές)

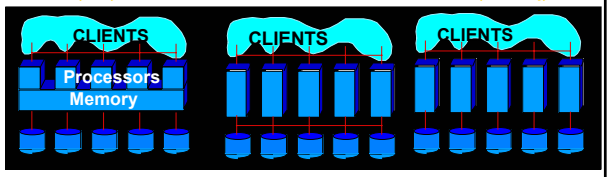


Εύκολο στον προγραμματισμό
Ακριβό στην κατασκευή
Δύσκολη κλιμάκωση

Sequent, SGI, Sun

Μια μέση επιβάρυνση 1% για κάθε επιπρόσθετη CPU σημαίνει ότι η μέγιστη επιτάχυνση είναι 37, πρόσθεση νέων CPU οδηγεί σε επιβράδυνση, 1000 CPU μόνο 4% πιο γρήγορες από ένα σύστημα με 1 CPU

Διαμοιραζόμενης Μνήμης Shared Memory (SMP) | Διαμοιραζόμενου Δίσκου Shared Disk | Τίποτα διαμοιρασμένο Shared Nothing (δικτυακές)



Εύκολο στον προγραμματισμό
Ακριβό στην κατασκευή
Δύσκολη κλιμάκωση

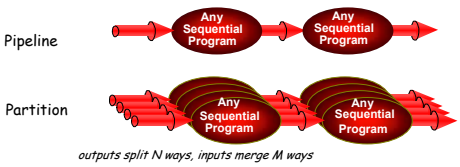
Sequent, SGI, Sun

VMScluster, Sysplex

Δύσκολο στον προγραμματισμό
Οικονομικό στην κατασκευή
Εύκολη κλιμάκωση

Tandem, Teradata, SP2

- Παραλληλισμός μέσω σωληνωτής εκτέλεσης (pipeline parallelism): πολλές μηχανές που η καθεμία εκτελεί ένα βήμα μιας διαδικασίας πολλών βημάτων – (μπλοκάρει περιμένοντας τα αποτελέσματα της προηγούμενης)
- Παραλληλισμός με διάσπαση (partition parallelism) ή διαμοιραζόμεων δεδομένων: πολλές μηχανές που εκτελούν την *ίδια* λειτουργία σε διαφορετικά τμήματα των δεδομένων
- Και οι δύο τύποι παραλληλισμού είναι δυνατοί στα ΣΔΒΔ

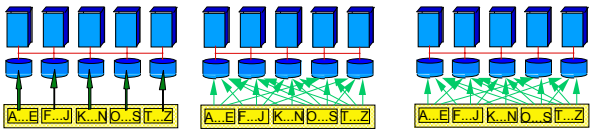


outputs split N ways, inputs merge M ways

- ΣΔΒΔ επιτυχημένα μοντέλα παραλληλισμού:
  - Teradata, Tandem vs. Thinking Machines, KSR.
  - Κάθε σημαντικό εμπορικό ΣΔΒΔ έχει κάποιο || εξυπηρέτη
- Λόγοι της επιτυχίας:
  - Bulk-processing (= παραλληλισμός διαμερισμού).
  - Φυσική διασωλήνωση.
  - Φτηνό υλικό (hardware)
  - Διαφάνεια παραλληλισμού για το χρήστη/προγραμματιστή εφαρμογών

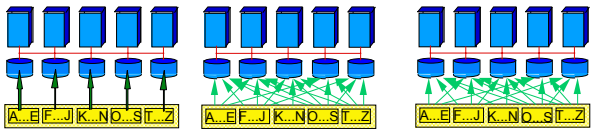
Διαμερισμός μιας σχέσης

Διαστήματα Τιμών | Κατακερματισμός | Round Robin



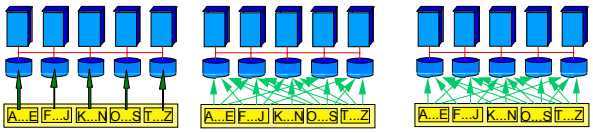
Ταξινόμηση, επιλογή n περιοχών έτσι ώστε κάθε περιοχή να περιέχει περίπου τον ίδιο αριθμό πλειάδων, πλειάδες στο διάστημα i στον επεξεργαστή i

Διαστήματα Τιμών | **Κατακερματισμός** | Round Robin



Μια συνάρτηση κατακερματισμού εφαρμόζεται σε επιλεγμένα γνωρίσματα κάθε πλειάδας και μας δίνει τον αριθμό του επεξεργαστή

Διαστήματα Τιμών | Κατακερματισμός | **Round Robin**

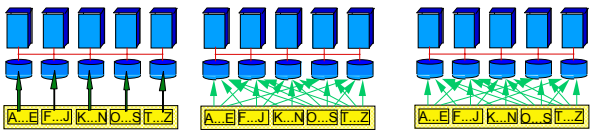


Η i-οστή πλειάδα πάει στον i mod n επεξεργαστή

- Round robin καλό για προσπέλαση όλης της σχέσης
- Αν μόνο ένα υποσύνολο της σχέσης, διαμερισμός διαστήματος τιμών και κατακερματισμού (αν ισότητα) καλύτερα – αν βέβαια η επιλογή αφορά τα ίδια γνωρίσματα
- Επιλογή περιοχής, τότε διαμερισμός διαστήματος τιμών καλύτερη
- Το data skew – ανομοιόμορφη κατανομή δεδομένων - πρόβλημα

  Μια δυνατή λύση είναι η δειγματοληψία

Διαστήματα Τιμών | Κατακερματισμός | Round Robin



Καλός για συνενώσεις ισότητας, Ερωτήσεις περιοχής Ομαδοποίηση

Καλός για συνενώσεις ισότητας

Καλός τρόπος για καταμερισμό φορτίου

Αρχιτεκτονικές διαμοιραζόμενου δίσκου και μνήμης λιγότερο ευαίσθητες στο διαμερισμό
Πιο σημαντικός για τις αρχιτεκτονικές «Τίποτα Διαμοιραζόμενο»

- Διάσπαση, split
- Συνένωση, merge
- δίκτυο ροής δεδομένων (data flow network)

Διαφορετικοί Τύποι Παραλληλισμού σε ΣΔΒΔ

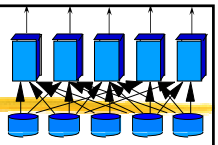
- Παράλληλη εκτέλεση μιας πράξης (Intra-operator parallelism)
  - όλες οι μηχανές εργάζονται στην εκτέλεση μιας συγκεκριμένης πράξης (scan, sort, join)
- Παράλληλη εκτέλεση πολλών πράξεων (Inter-operator parallelism)
  - κάθε πράξη μπορεί να εκτελείται ταυτόχρονα σε διαφορετικούς κόμβους (και χρήση σωληνωτής εκτέλεσης)
- Παράλληλη εκτέλεση ερωτήσεων (Inter-query parallelism)
  - διαφορετικές ερωτήσεις εκτελούνται σε διαφορετικούς κόμβους

Θα επικεντρωθούμε στο πως μπορούμε να εκτελέσουμε παράλληλα μια πράξη (intra-operator ||-ism)

Παράλληλη Σάρωση (Scan)

- Σάρωση παράλληλα, μετά συγχώνευση
- Μια επιλογή μπορεί να μην απαιτεί την προσπέλαση όλων των κόμβων στην περίπτωση του διαμερισμού διαστήματος ή κατακερματισμού.
- Μπορούμε να διατηρούμε ευρετήρια σε κάθε κόμβο (για κάθε τμήμα δεδομένων).
- Ερώτηση: Ποιο είναι το κατάλληλο ευρετήριο για κάθε είδος διαμερισμού δεδομένων;

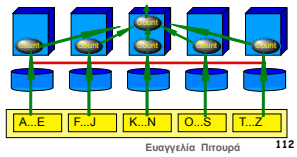
Παράλληλη Ταξινόμηση



- Σημερινά απόδοση:
  - 8.5 Gb/minute, shared-nothing; Datamation benchmark in 2.41 secs (UCB students! http://now.cs.berkeley.edu/NowSort/)
- Ιδέα: ταξινόμηση τοπικά και μετά συγχώνευση
- Καλύτερη ιδέα
  - Παράλληλη σάρωση και ταυτόχρονος διαμερισμός σε διαστήματα τιμών
  - Ταξινόμηση των τιμών τοπικά (σε κάθε κόμβο)
  - Ως αποτέλεσμα τα δεδομένα είναι ταξινομημένα και διαμοιρασμένα σε διαστήματα τιμών
  - Πρόβλημα: skew! (ανομοιόμορφη κατανομή)
  - Λύση: "δειγματοληψία" των δεδομένων στην αρχή ώστε να καθοριστούν τα σημεία διάσπασης (splitting vector – διάνυσμα διαχωρισμού)

Παράλληλες Συναθροιστικές

- Για κάθε συναθροιστική συνάρτηση χρειάζεται μια διάσπαση (decomposition):
  - count(S) = **S** count(s(i)), το ίδιο για το sum()
  - avg(S) = (**S** sum(s(i))) / **S** count(s(i))
  - κοκ ...
- Για ομαδοποίηση (group by):
  - Sub-aggregate groups close to the source.
  - Pass each sub-aggregate to its group's site.
    - Chosen via a hash fn.

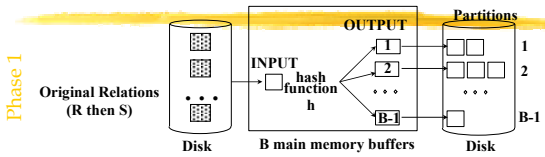


Jim Gray & Gordon Bell: VLDB 95 Parallel Database Systems Survey

Παράλληλες Συνενώσεις

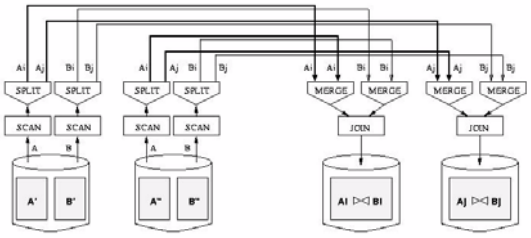
- Εμφωλευμένος Βρόγχος (Nested loop):
  - Κάθε εξωτερική πλειάδα πρέπει να συγκριθεί με κάθε εσωτερική πλειάδα με την οποία ίσως συνενώνεται
  - Εύκολο όταν έχουμε διαμοιρασμό διαστημάτων τιμών στο γνώρισμα που θα γίνει η συνένωση, σύσκολο αλλιώς
- Συνένωση αφού γίνει ταξινόμηση Sort-Merge (or plain Merge-Join):
  - Η ταξινόμηση δίνει διαχωρισμό διαστημάτων.
    - Πρόβλημα αντιμετώπισης δύο μη ομοιόμορφων κατανομών
  - Merging partitioned tables is local.

Παράλληλη συνένωση με κατακερματισμό



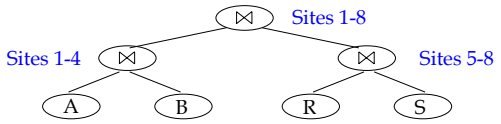
- Στην πρώτη φάση, τα τμήματα κατανέμονται σε διαφορετικούς κόμβους:
  - Μια καλή συνάρτηση κατακερματισμού ώστε ομοιόμορφη κατανομή της δουλειάς
- Συνένωση σε κάθε κόμβο

Δίκτυο Ροής για || Συνένωση



Παραλληλισμός Πλάνου Εκτέλεσης Ερώτησης

- Σύνθετες ερωτήσεις: Inter-Operator παραλληλισμός
  - Σωλήνωση μεταξύ τελεστών:
    - note that sort and phase 1 of hash-join block the pipeline!!
  - Bushy (θαμνώδη δέντρα)



Observations

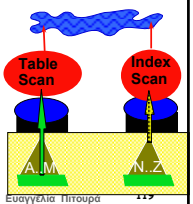
- It is relatively easy to build a fast parallel query executor
  - S.M.O.P.
- It is hard to write a robust and world-class parallel query optimizer.
  - There are many tricks.
  - One quickly hits the complexity barrier.
  - Still open research!

Παράλληλη βελτιστοποίηση ερωτήσεων

- Κοινή Προσέγγιση: 2 φάσεις
  - Επιλογή του καλύτερου σειριακού πλάνου
  - Επέλεξε το βαθμό παραλληλισμού.
- "Ανάθεσε" τελεστές (πράξεις) σε υπολογιστές
  - Δηλαδή, πρόσθεσε στο δέντρο εκτέλεσης πληροφορία για το που θα εκτελεστεί μια ερώτηση

What's Wrong With That?

- Το καλύτερο σειριακό πλάνο != Καλύτερο || πλάνο! Γιατί;
- Trivial counter-example:
  - Table partitioned with local secondary index at two nodes
  - Range query: all of node 1 and 1% of node 2.
  - Node 1 should do a scan of its partition.
  - Node 2 should use secondary index.
- SELECT *
  FROM telephone_book
  WHERE name < "NoGood";



Περίληψη

- ||-ism natural to query processing:
  - Both pipeline and partition ||-ism!
- Shared-Nothing vs. Shared-Mem
  - Shared-disk too, but less standard
  - Shared-mem easy, costly. Doesn't scaleup.
  - Shared-nothing cheap, scales well, harder to implement.
- Intra-op, Inter-op, & Inter-query ||-ism all possible.

Περίληψη

- Data layout choices important!
- Most DB operations can be done partition-||
  - Sort.
  - Sort-merge join, hash-join.
- Complex plans.
  - Allow for pipeline-||ism, but sorts, hashes block the pipeline.
  - Partition ||-ism acheived via bushy trees.

Περίληψη

- Hardest part of the equation: optimization.
  - 2-phase optimization simplest, but can be ineffective.
  - More complex schemes still at the research stage.
- We haven't said anything about Xacts, logging.
  - Easy in shared-memory architecture.
  - Takes some care in shared-nothing.