ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Βάσεις Δεδομένων ΙΙ

Κατανεμημένα συστήματα ΒΔ

Μανώλης Μαραγκουδάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Πανεπιστήμιο Αιγαίου-
Τμήμα Μηχανικών
Πληροφοριακών και
Επικοινωνιακών Συστημάτων

Βάσεις Δεδομένων ΙΙ

Ενότητα 4-Κατανεμημένα συστήματα ΒΔ

Μανώλης Μαραγκουδάκης

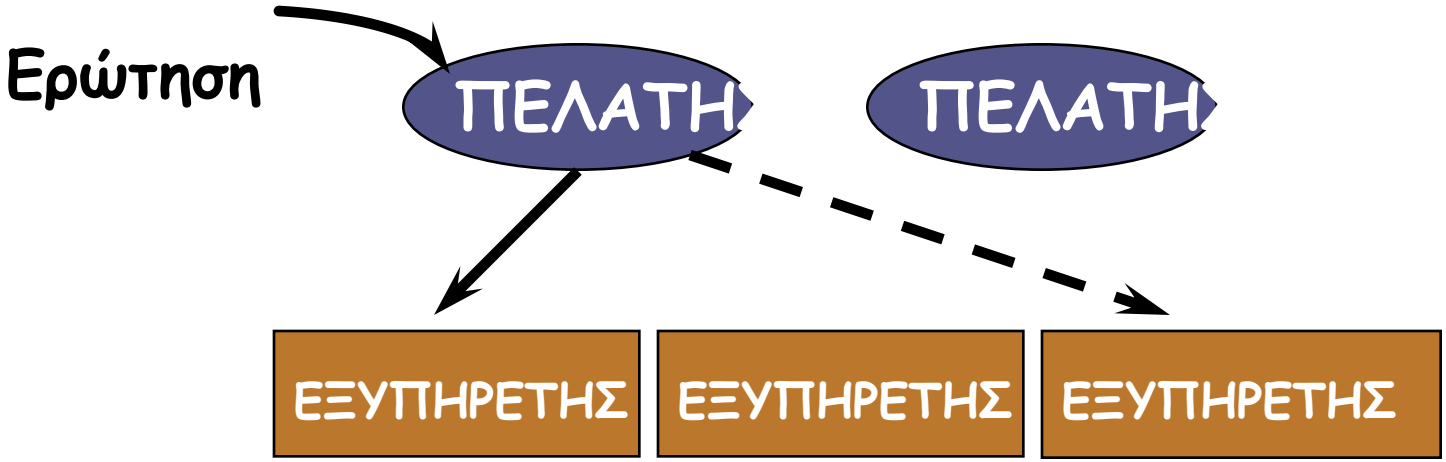
www.icsd.aegean.gr/mmarag

- Τα δεδομένα είναι αποθηκευμένα σε διαφορετικούς κόμβους και κάθε κόμβο τον διαχειρίζεται ένα ΣΔΒΔ που τρέχει ανεξάρτητα

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

Ομογενή: Κάθε πλευρά τρέχει τον ίδιο τύπο ΣΔΒΔ

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

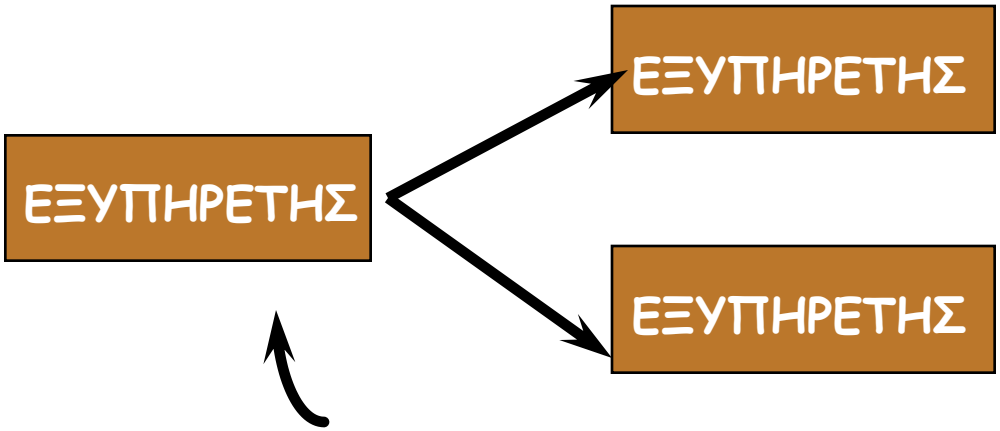


- Πελάτη-Εξυπηρέτη

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

- Συνεργαζόμενων εξυπηρετών

Μια ερώτηση εκτελείται σε πολλές πλευρές



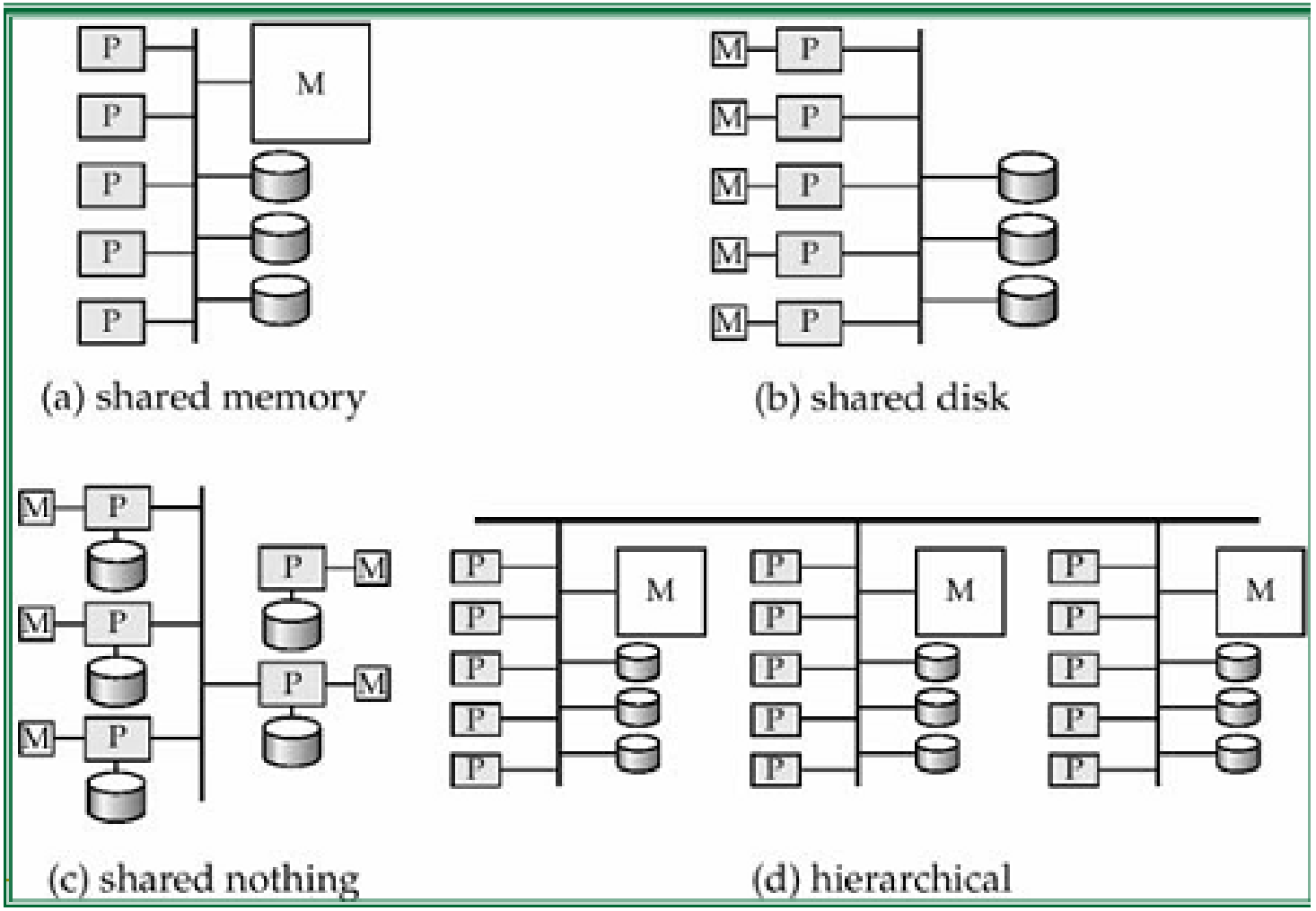
Αρχιτεκτονικές Παράλληλων ΣΒΔ

Διαμοιρασμένη μνήμη (shared memory): οι επεξεργαστές μοιράζονται κοινή μνήμη

Διαμοιρασμένο σύστημα δίσκων (shared disk): οι επεξεργαστές μοιράζονται κοινό σύστημα δίσκων

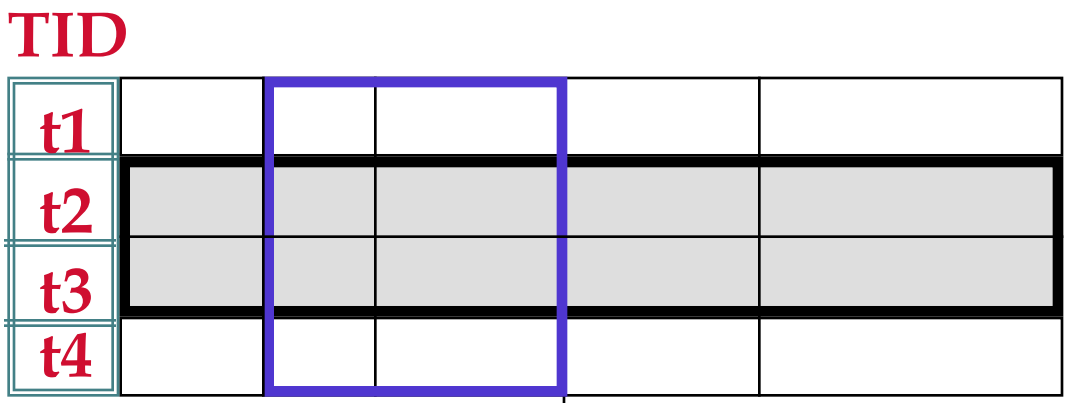
«Τίποτα» διαμοιρασμένο (shared nothing): οι επεξεργαστές ΔΕΝ μοιράζονται ούτε κοινή μνήμη ούτε κοινό δίσκο

Ιεραρχική: υβριδική αρχιτεκτονική – συνδυασμός των παραπάνω

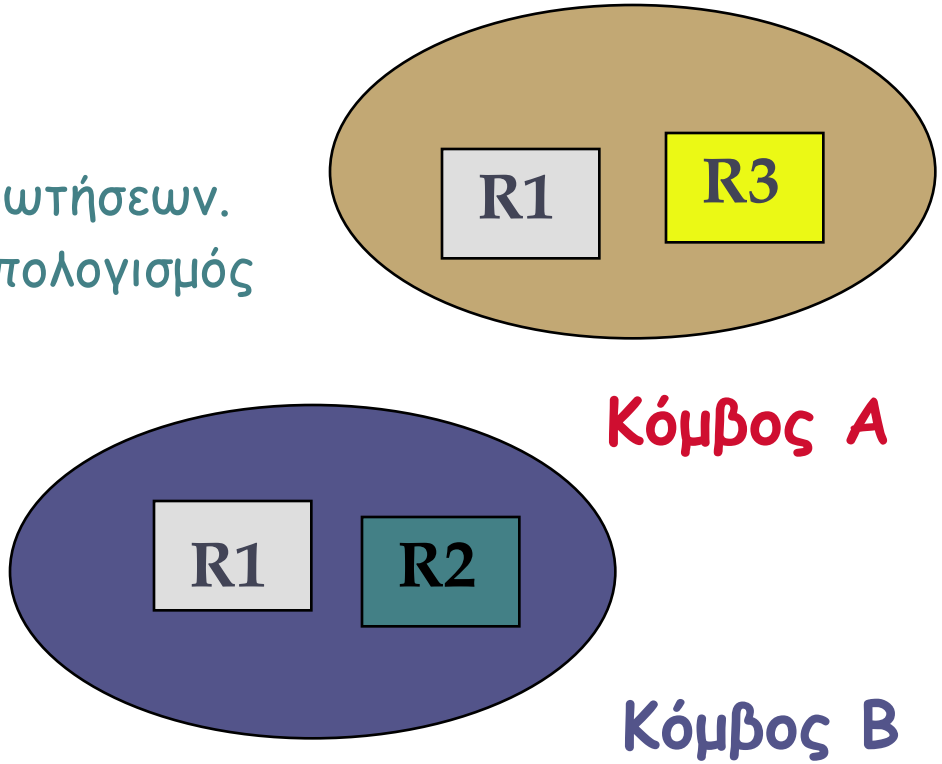

P
M
(a) shared memory
(b) shared disk
(c) shared nothing
(d) hierarchical

- 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

- Κατατεμαχισμός (fragmentation)
  - Οριζόντιος: Με βάση κάποια συνθήκη – πράξη (σ)
  - Κάθετος: με βάση την πράξη (Π)
    - Χωρίς Απώλειες στη συνένωση; tids.



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



Άσκηση….

Δίνεται η σχέση R(A, B, C, D, E) με κλειδί το A. Υπάρχουν 3 εφαρμογές που προσπελαύνουν την R:

- Η εφαρμογή 1 προσπελαύνει τα A, B, C και συμβάλει στο 40% των προσπελάσεων στην R.
- Η εφαρμογή 2 προσπελαύνει τα C, E και συμβάλει στο 20% των προσπελάσεων στην R
- Η εφαρμογή 3 προσπελαύνει τα D, E και συμβάλει στο 40% των προσπελάσεων στην R

Ποια από τις παρακάτω είναι καλύτερη κάθετη κατάτμηση της R σε δύο τμήματα; Για ποιο λόγο;

- R1(A,B,C,E), R2(A,D,E)
- R1(A,B,C,E), R2(A,D)
- R1(A,B,C), R2(D,E)
- R1(A,B,C), R2(A,D,E)

Λύση…..

Η πρώτη κατάτμηση δεν είναι επιθυμητή, καθώς αντιγράφει τη στήλη Ε που δεν είναι κλειδί.

Η δεύτερη κατάτμηση επίσης δεν είναι επιθυμητή, καθώς 40% των προσπελάσεων (εφαρμογή 3) απαιτούν τη συνένωση των R1 και R2.

Η τρίτη κατάτμηση επίσης δεν είναι επιθυμητή, η R δεν μπορεί να ανασυντεθεί από τις R1 και R2.

Η τέταρτη κατάτμηση είναι η καλύτερη, καθώς μόνο 20% των προσπελάσεων (εφαρμογή 2) απαιτούν τη συνένωση των R1 και R2.

Κατάλογος Συστήματος

- Διατήρηση πληροφορίας για την κατανομή των δεδομένων στους κόμβους
- Όνομα για κάθε αντίγραφο σε κάθε τμήμα. Διατήρηση τοπικής αυτονομίας
  - **<local-name, birth-site>**
- Κατάλογος σε κάθε κόμβο: Περιγράφει κάθε αντικείμενο (τεμάχιο, αντίγραφο) στον κόμβο + 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.

- Εκτός του I/O κόστους (**D**), το κόστος μετάδοσης δεδομένων (**T**) στο δίκτυο
- Πιθανό κέρδος από το ότι 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 κατατεμαχισμός: 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

Οριζόντιος τεμαχισμός

$$\sigma_{S.rating > \kappa1 \text{ and } S.rating < \kappa2}(S) =$$

$$\sigma_{S.rating > \kappa1 \text{ and } S.rating < \kappa2}(S1 \cup S2) =$$

$$\sigma_{S.rating > \kappa1 \text{ and } S.rating < \kappa2}(S1) \cup$$

$$\sigma_{S.rating > \kappa1 \text{ and } S.rating < \kappa2}(S2)$$

Εκτέλεση στη Shanghai

$$\sigma_{S.rating > \kappa1 \text{ and } S.rating < \kappa2}(\sigma_{S.rating < 5}(S)) \cup$$

$$\sigma_{S.rating > \kappa1 \text{ and } S.rating < \kappa2}(\sigma_{S.rating >= 5}(S))$$

Εκτέλεση στο Tokyo

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

- Κάθετος κατατεμαχισμός : *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

$\sigma_{S.rating > \kappa1 \text{ and } S.rating < \kappa2}(S) =$ Κάθετος κατατεμαχισμός

$\sigma_{S.rating > \kappa1 \text{ and } S.rating < \kappa2}(S1 * S2) =$

```
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, κόστος για την μεταφορά του αποτελέσματος

Ημι-Συνένωση vs. Natural Join

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Production

Dept

DeptName	Manager
Sales	Harriet
Production	Charles

Employee $\ltimes$ *Dept*

Name	EmpId	DeptName
Sally	2241	Sales
Harriet	2202	Production

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Manager
Sales	Harriet
Production	Charles

Employee $\bowtie$ *Dept*

Name	EmpId	DeptName	Manager
Sally	2241	Sales	Harriet
Harriet	2202	Sales	Harriet

Ημι-Συνένωση

- Αν υποθέσουμε ότι έχουμε μια σύνδεση μεταξύ δύο σχέσεων, τότε έχουμε διάφορους τρόπους να γράψουμε τη σύνδεση αυτή, ισοδύναμα, χρησιμοποιώντας semi-join.
- Το ποιον από αυτούς θα διαλέξουμε, εξαρτάται από τη φύση της συγκεκριμένης βάσης δεδομένων.

$$R \triangleright\!\triangleleft_{A=A} S \Leftrightarrow (R \triangleright\!<_{A=A} S) \triangleright\!\triangleleft_{A=A} S$$

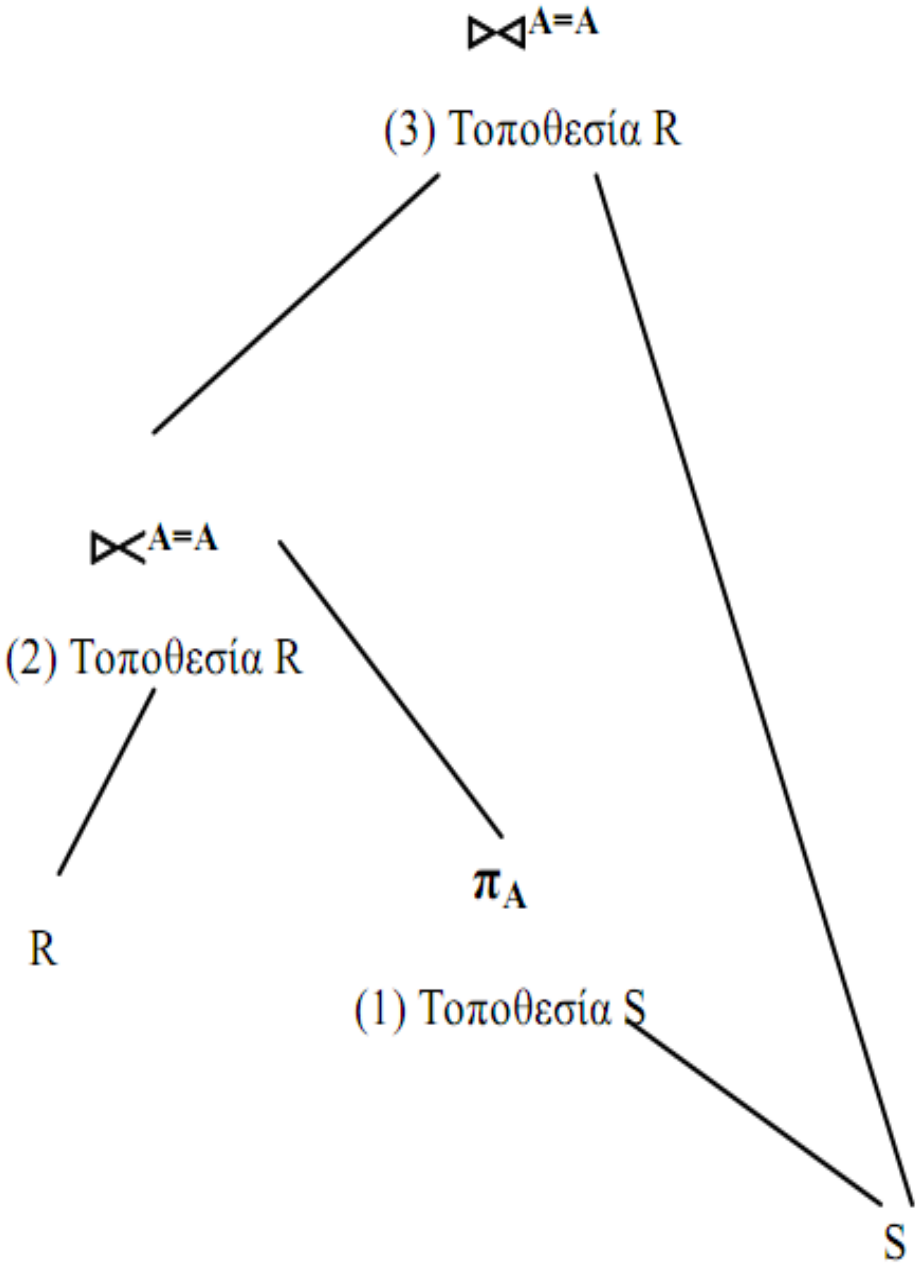
$$\Leftrightarrow (R \triangleright\!<_{A=A} \pi_A(S)) \triangleright\!\triangleleft_{A=A} S$$

$$\Leftrightarrow R \triangleright\!\triangleleft_{A=A} (S \triangleright\!<_{A=A} R)$$

$$\Leftrightarrow (R \triangleright\!<_{A=A} S) \triangleright\!\triangleleft_{A=A} (S \triangleright\!<_{A=A} R)$$

Ημι-Συνένωση

- Συνήθως επιλέγεται η δεύτερη παραλλαγή, η οποία χρησιμοποιεί την πολύ γρήγορη ***Προβολή*** και γιατί εμφανίζει τη σχέση S και στις 2 μερές



Παράδειγμα: Επεξεργασία Ερωτήσεων: Κατανεμημένη Συνένωση

D κόστος εγγραφής/ανάγνωσης σελίδας

T κόστος μεταφοράς σελίδας

- Μεταφορά όταν χρειάζεται, Εμφωλευμένος βρόγχος με τη σχέση Sailors στον εξωτερικό βρόγχο :
  - **Κόστος:** 500 D + 500 * 1000 (D+T)

- Μεταφορά όταν χρειάζεται, Εμφωλευμένος βρόγχος με τη σχέση Sailors στον εξωτερικό βρόγχο (συνέχεια) :
  - Αν η ερώτηση δεν είχε υποβληθεί στο Λονδίνο πρέπει να προσθέσουμε και το κόστος μεταφοράς του αποτελέσματος στον κόμβο που αρχικά υποβλήθηκε η ερώτηση.

Επεξεργασία Ερωτήσεων: Κατανεμημένη Συνένωση

- Μεταφορά όλης της σχέσης σε έναν κόμβο: μεταφορά της Reserves στο Λονδίνο
  - **Κόστος**: 1000 T + 4500 D (με ταξινόμηση/συγχώνευση; κόστος = 3*(500+1000))
  - 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 Reserves στο Λονδίνο -- με την επιβάρυνση του υπολογισμού και της μεταφοράς της προβολής της Sailors και του υπολογισμού της συνένωσης της προβολής με τη Reserves

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

- Ημι-συνένωση

Βήμα 1: Στο Λονδίνο, προβολή (project) Sailors στις στήλες του join

- Μεταφορά του αποτελέσματος στο Παρίσι

Βήμα 2: Στο Παρίσι, συνένωση της προβολής Sailors με τη Reserves.

Το αποτέλεσμα ονομάζεται **Ελάττωση (reduction)** της Reserves σε σχέση με τη Sailors.

- Μεταφορά της ελάττωσης της σχέσης Reserves στο Λονδίνο

Βήμα 3: Στο Λονδίνο, συνένωση Sailors με την ελάττωση της Reserves.

Παράδειγμα

Κλειδί	Μέγεθος (bytes)
S#	4
P#	4
QTY	10
SNAME	96
PNAME	196

- Έστω οι σχέσεις
  - S(S#,SNAME)
  - SP(S#,P#,QTY)
  - P(P#,PNAME)

  που βρίσκονται αποθηκευμένες στους κόμβους T1, T2 και T3 αντίστοιχα.
- Ας υποθέσουμε ότι γνωρίζουμε τα παρακάτω στοιχεία για τις σχέσεις αυτές:
  - S: έχει 6 εγγραφές με κλειδιά s1, s2, s3, s4, s5, s6.
  - SP: έχει 8 εγγραφές με κλειδιά (s1,p1), (s1,p2), (s1,p3), (s2,p1), (s2,p2), (s2,p3), (s3,p1), (s3,p3).
  - P: έχει 6 εγγραφές με κλειδιά p1, p2, p3, p4, p5, p6.

Παράδειγμα

- Έστω, τώρα, ότι μια τοποθεσία Τ4 ενεργοποιεί την ερώτηση $S \bowtie SP \bowtie P$

- Έστω ότι το κόστος μεταφοράς = μέγεθος των μεταφερόμενων δεδομένων (Μ).

Το κόστος χωρίς λειτουργίες semi-join.

Κλειδί	Μέγεθος (bytes)
S#	4
P#	4
QTY	10
SNAME	96
PNAME	196

- Η σχέση S (S#, SNAME) έχει 6 εγγραφές και καθεμία από αυτές έχει μέγεθος 4+96=100 bytes. Άρα, το μέγεθος της S είναι 6x100 bytes=600 bytes.
- Η σχέση SP (S#, P#, QTY) έχει 8 εγγραφές και καθεμία από αυτές έχει μέγεθος 4+4+10=18 bytes. Άρα, το μέγεθος της SP είναι 8x18 bytes=144 bytes.
- Η σχέση P (P#, PNAME) έχει 6 εγγραφές και καθεμία από αυτές έχει μέγεθος 4+196=200 bytes. Άρα, το μέγεθος της P είναι 6x200 bytes=1200 bytes.

Παράδειγμα

- Το αποτέλεσμα της ερώτησης S JOIN SP JOIN P, είναι μία σχέση R, η οποία αποτελείται από 8 εγγραφές και τα πεδία κάθε εγγραφής είναι τα παρακάτω:
  - R(S#, SNAME, P#, PNAME, QTY).
  - Το μέγεθος κάθε εγγραφής είναι 4+96+4+196+10=310 bytes.
- Συνεπώς, το μέγεθος της σχέσης R θα είναι 8*310 bytes=2480 bytes.

Κλειδί	Μέγεθος (bytes)
S#	4
P#	4
QTY	10
SNAME	96
PNAME	196

Παράδειγμα

- Αν η ερώτηση εκτελεστεί σε οποιονδήποτε άλλο κόμβο εκτός από τον κόμβο Τ4, αυτό σημαίνει ότι θα πρέπει να μεταφέρουμε το αποτέλεσμα στον κόμβο Τ4 όπου έχει ενεργοποιηθεί η ερώτηση.

- Το κόστος μεταφοράς δεδομένων της λύσης R, είναι ίσο με το μέγεθος των μεταφερόμενων δεδομένων, δηλ. είναι C(R)= 2480 bytes.

Κόστος με χρήση S ⋉ SP

- Εύρεση κόστους αν υποθέσουμε ότι κάνουμε
  - S SemiJOIN SP
- Η ερώτηση θα εκτελεστεί ως εξής:
  - Προβάλλουμε το S# από την SP στον κόμβο T2 και θα το στείλουμε στον T1.
  - Εκτελουμε το semi-join το οποίο θα δημιουργήσει το αποτέλεσμα S'.
  - Στέλνουμε στον κόμβο T4, τα S', SP, P και εκτελουμε την ένωση.
- Το SP πρέπει να το στείλουμε πάλι στον T4, άσχετα από το αν έχει ήδη συμμετάσχει στο semi-join στον κόμβο T1.

Παράδειγμα

- Αναλυτικά έχουμε:
- ☐ Πρώτα θα γίνει στον κόμβο Τ2, προβολή των τιμών S# της σχέσης SP.
  - Δηλαδή θα γίνει η πράξη: F = ΠS#(SP). Η σχέση F έχει 3 εγγραφές:
    - s1, s2, s3. Αυτές οι τιμές μεταφέρονται στον κόμβο Τ1, όπου βρίσκεται η σχέση S.
  - Δηλαδή, μεταφέρονται 3*4=12 bytes.
- Στον κόμβο Τ1, γίνεται το join της σχέσης S και των τιμών S# που ήρθανε από τον κόμβο Τ2, δηλαδή γίνεται η πράξη:
  - S'= F JOIN S.
    - Η σχέση S' έχει μέγεθος 3*(4+96)=300 bytes.
- Στη συνέχεια, μεταφέρεται η σχέση S' από τον κόμβο Τ1 στον κόμβο Τ4, δηλαδή μεταφέρονται 300 bytes.
  - Η αξία της κίνησης αυτής είναι ότι με το S SEMIJOIN SP, αντί να μεταφερθεί από τον κόμβο Τ1 στον κόμβο Τ4 ολόκληρη η σχέση S, μεταφέρονται μόνο εκείνες οι εγγραφές που πραγματικά χρειάζονται και θα χρησιμοποιηθούν.

Παράδειγμα

- Κατόπιν, μεταφέρεται η σχέση SP από τον κόμβο Τ2 στον κόμβο Τ4 (144 bytes) και επίσης μεταφέρεται η σχέση P από τον κόμβο Τ3 στον κόμβο Τ4 (1200 bytes).
- Τέλος, στον κόμβο Τ4, πραγματοποιείται η ερώτηση
  - R = S' JOIN SP JOIN P.
- Το κόστος της ερώτησης αν χρησιμοποιηθεί το
  - S SEMIJOIN SP, είναι:
    - 12+300+144+1200=1656 bytes.

Εύρεση του βέλτιστου πλάνου, με τη χρήση semi-joins

- Παρατηρούμε ότι προβάλλοντας τα πεδία S# και P# της σχέσης SP παίρνουμε 3 τιμές τόσο για το κλειδί S#, όσο και για το P#.
- Από την άλλη μεριά, προβάλλοντας το πεδίο S# της S και το πεδίο P# της P παίρνουμε 6 τιμές και στις δύο περιπτώσεις.
- Δεδομένου ότι τα πεδία S# και P# αποτελούν τα πεδία σύνδεσης για τη σύνδεση των σχέσεων S, SP και P, και ότι είναι ίδιου μεγέθους,
  - συμφέρει να προβάλλουμε τα πεδία S# και P#, της σχέσης SP.

Το καλύτερο πλάνο είναι το:

- Προβάλλουμε το πεδίο S# της SP στον κόμβο Τ2 και μεταφέρουμε τη σχέση $F=\pi_{S\#}(SP)$,
  - Δηλαδή 12 bytes, στον κόμβο Τ1.
- Προβάλλουμε το πεδίο P# της SP στον κόμβο Τ2 και μεταφέρουμε τη σχέση $G=\pi_{P\#}(SP)$,
  - δηλαδή 12 bytes, στον κόμβο Τ3.
- Κάνουμε τη σύνδεση F'=S $JOIN_{S\#=S\#}$ F στον κόμβο Τ1 και προκύπτει μια σχέση με μέγεθος :
  - 3*100=300 bytes.
- Επίσης, κάνουμε τη σύνδεση G'= P $JOIN_{P\#=P\#}$ G στον κόμβο Τ3 και προκύπτει μια σχέση με μέγεθος :
  - 3*200=600 bytes.

Το καλύτερο πλάνο είναι το:

- Μεταφέρουμε τη σχέση SP από τον κόμβο T2 στον κόμβο T4,
  - δηλαδή μεταφέρουμε 144 bytes.
- Μεταφέρουμε την F', δηλαδή 300 bytes, από τον κόμβο T2 και κάνουμε τη σύνδεση F"=F' $JOIN_{S\#=S\#}$ SP, στον κόμβο T4.
- Επίσης, μεταφέρουμε την G', δηλαδή 600 bytes, από τον κόμβο T3 και κάνουμε τη σύνδεση F"$JOIN_{P\#=P\#}$ G', για να πάρουμε την τελική σχέση R στον κόμβο T4.
- ΣΥΝΟΛΙΚΑ:
  - 12 + 12 + 144 + 300 + 600 = 1068 bytes

Ενημέρωση Κατανεμημένων Δεδομένων

- Διαχείριση αντιγράφων
- Κατανεμημένη Διαχείριση Δοσοληψιών
- Κατανεμημένη Ανάνηψη από Αποτυχία

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 προϊόντα ακολουθούν αυτήν την προσέγγιση

Σύγχρονη Ενημέρωση Αντιγράφων

- 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.

Ασύγχρονη Ενημέρωση Αντιγράφων

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

Peer-to-Peer

- 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., Site 1: Αλλάζει η τιμη του X σε 35; Site 2: αλλάζει την τιμή του X σε 36)
- Προτιμότερη όταν δεν συμβαίνουν συγκρούσεις:
  - Π.χ., Κάθε master κόμβος κατέχει ένα ξένο τεμάχιο

Πρωτεύουσα Αντιγραφή

- **Ακριβώς ένα** αντίγραφο 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

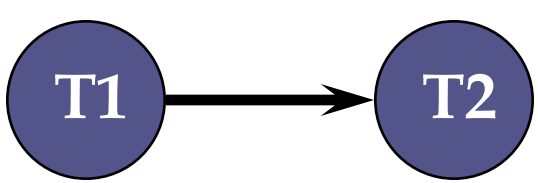
- **Log-Based Εντοπισμός**: Το ημερολόγιο του συστήματος (log) που κρατείται για ανάνηψη χρησιμοποιείται για τη δημιουργία ενός πίνακα που καλείται Change Data Table (CDT).
  - Αν αυτό γίνεται όταν το log tail γράφεται στο δίσκο τότε με κάποιο τρόπο να σβηστούν οι τροποποιήσεις που οφείλονται σε δοσοληψίες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

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

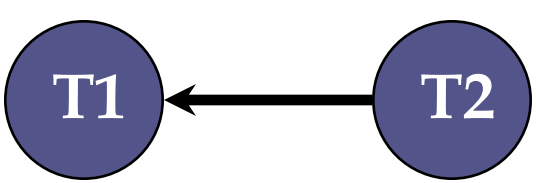
- **Κεντρικά (Centralized):** Ένας κόμβος είναι υπεύθυνος για τη διαχείριση όλων των κλειδιών.
  - Αποτυχία του κόμβου;
- **Πρωτεύον αντίγραφο:** Η διαχείριση των κλειδιών για ένα αντικείμενο γίνεται στον κόμβο όπου βρίσκεται το πρωτεύον αντίγραφο του αντικειμένου
  - Η ανάγνωση απαιτεί προσπέλαση και 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

Κατανεμημένη Ανίχνευση Αδιεξόδων

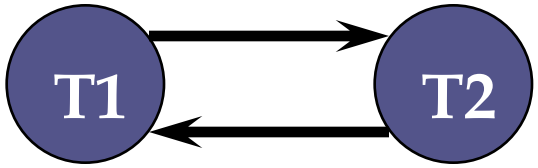
- Κάθε κόμβος διατηρεί έναν τοπικό waits-for γράφο.
- Ένα ολικό αδιέξοδο μπορεί να δημιουργηθεί ακόμα και αν οι τοπικοί κόμβοι δεν περιέχουν κύκλους:



SITE A



SITE B



GLOBAL

Κατανεμημένη Ανίχνευση Αδιεξόδων

- Τρεις λύσεις:
  - Κεντρική (αποστολή όλων των τοπικών γράφων σε έναν κόμβο)
  - Hierarchical (οργάνωση κόμβων σε μια ιεραρχία και αποστολή γράφων στον γονέα στην ιεραρχία
  - Timeout (απόρριψη μιας δοσοληψίας αν περιμένει για πάρα πολύ).

- Δυο νέα θέματα:
  - Δυο νέα ειδών αποτυχιών, π.χ., συνδέσεις και απομακρυσμένοι κόμβοι.
  - Αν τμήματα μιας δοσοληψίας εκτελούνται σε διαφορετικούς κόμβους όλες ή μια πρέπει να επικυρωθούν $\Rightarrow$ 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 (subordinates).

Πρωτόκολλο Επικύρωσης Δύο Φάσεων

- Όταν μια δοσοληψία πρέπει να επικυρωθεί:
  - Ο συντονιστής στέλνει ένα μήνυμα **prepare** σε όλους τους συμμετέχοντες κόμβους.
  - Κάθε συμμετέχον κόμβος force-writes μια εγγραφή **abort** ή **prepare** στο log και μετά στέλνει ένα μήνυμα **no** ή **yes** στον συντονιστή.
  - 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.

Πρωτόκολλο Επικύρωσης Δύο Φάσεων

- 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 περιέχουν τα ids όλων των κόμβων που συμμετέχουν

Πρωτόκολλο Επικύρωσης Δύο Φάσεων

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

Πρωτόκολλο Επικύρωσης Δύο Φάσεων

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

Πρωτόκολλο Επικύρωσης Δύο Φάσεων

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

- Αν ο συντονιστής για την T αποτύχει, οι συμμετέχοντες κόμβοι που έχουν ψηφίσει δεν μπορούν να αποφασίσουν αν θα πρέπει να επικυρώσουν ή να ακυρώσουν την T μέχρι να αναρρώσει ο συντονιστής.
  - T is 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 ακόμα ψηφίσει yes, πρέπει να περιμένει μέχρι να αποκριθεί ο συντονιστής

Πρωτόκολλο Επικύρωσης Δύο Φάσεων

- Μηνύματα Ack χρησιμοποιούνται για να ειδοποιήσουν τον συντονιστή ότι μπορεί να «ξεχάσει» τη δοσοληψία , μέχρι να λάβει αυτό το μήνυμα πρέπει να κρατά την 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

Περίληψη

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