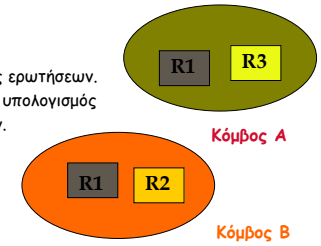
Κατανεμημένες Βάσεις Δεδομένων

Αντίγραφα -- Ομοιοτυπία (Replication)

- Διαθεσιμότητα
- Γρηγορότερος υπολογισμός ερωτήσεων.
- Σύγχρονο και Ασύγχρονος υπολογισμός
  - ενημέρωση αντιγράφων.



- Διατήρηση πληροφορίας για την κατανομή των δεδομένων στους κόμβους
- Όνομα για κάθε αντίγραφο σε κάθε κόμβο. 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.

Lets assume the existence of a data item x with copies x_1, x_2, …, x_n

x: logical data item

x_i's: physical data items

A replication control protocol is responsible for mapping each read/write on a logical data item (R(x)/W(x)) to a set of read/writes on a (possibly) proper subset of the physical data item copies of x

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

- **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.
- ⇒ 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 αντιγράφψν 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., 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

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.

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

One Copy Serializability

Correctness

A DBMS for a replicated database should behave like a DBMS managing a one-copy (i.e., nonreplicated) database insofar as users can tell

One-copy serializable (1SR)

the schedule of transactions on a replicated database be *equivalent* to a serial execution of those transactions on a one-copy database

ROWA

Read One/Write All (ROWA)

A replication control protocol that maps each read to only *one* copy of the item and each write to a set of writes on *all* physical data item copies.

Even if one of the copies is unavailable an update transaction cannot terminate

Write-All-Available

Write-all-availabe

A replication control protocol that maps each read to only *one* copy of the item and each write to a set of writes on *all available* physical data item copies.

Quorum-Based Voting

Read quorum V_r and a write quorum V_w to read or write a data item

If a given data item has a total of V votes, the quorums have to obey the following rules:

1. $V_r + V_w > V$
2. $V_w > V/2$

Rule 1 ensures that a data item is not read or written by two transactions concurrently (R/W)

Rule 2 ensures that two write operations from two transactions cannot occur concurrently on the same data item (W/W)

Quorum-Based Voting

In the case of network partitioning,

determine which transactions are going to terminate based on the votes they can acquire

the rules ensure that two transactions that are initiated in two different partitions and access the same data item cannot terminate at the same time

Distributing Writes

Immediate writes

Deffered writes: the DBMS access only one copy of the data item, it delays the distribution of writes to other sites until the transaction has terminated and is ready to commit.

It maintains an intention list of deferred updates

After the transaction terminates, it send the appropriate portion of the intention list to each site that contains replicated copies

Optimizations – aborts cost less – may delay commitment – delays the detection of copies

Primary copy: use the same copy of a data item

Eager vs Lazy Replication

Eager replication: keeps all replicas synchronized by updating all replicas in a single transaction

Lazy replication: asynchronously propagate replica updates to other nodes after replicating transaction commits

Κατανεμημένη Ανάκαμψη

- 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

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).

Όταν μια δοσοληψία πρέπει να επικυρωθεί:

Φάση 1

1. Ο συντονιστής στέλνει ένα μήνυμα **prepare** 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.

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 περιέχουν τα ids όλων των κόμβων που συμμετέχουν

- 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

- 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

- 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 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 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

⌘ 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

2PC

Recovery::
New kinds of failures (communication links and remote sites)

Transaction atomicity:: either all of its operation are carried out or none

In a distributed environment, all of the servers involved in a transaction must agree on the final outcome of the transaction (i.e., a transaction must either commit or abort at all servers)

▪ COMMIT PROTOCOL: Enable the servers to reach a joint decision as to whether a transaction can be committed or aborted

Develop an **atomic commit protocol**

▪ A cooperative procedure used by a set of servers involved in a distributed transaction

▪ Enable the servers to reach a joint decision as to whether a transaction can be committed or aborted

▪ Why is needed?

the server's decision is affected by cc, server and network failures

2PC protocol

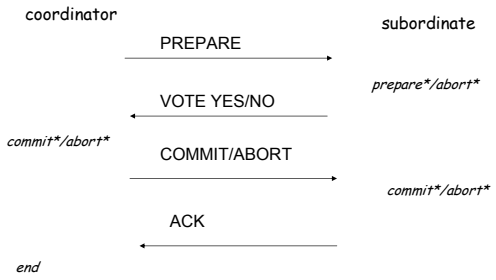
During normal execution

- Each site maintains a log, actions of a subtransaction are logged at the site it is executed
- In addition, a commit protocol

Coordinator (transaction manager at the site where the transaction originated)

Subordinates (transaction managers at the sites where its subtransactions execute)

2PC protocol



2PC protocol

4 types of messages:: Prepare, vote y/n, commit/abort, ack

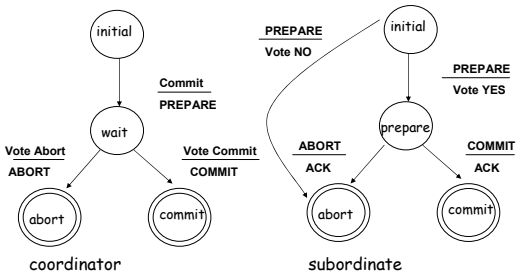
4 types of log records:: Prepare*, commit*, abort*, end

Subordinates force-write log records – Why?

Why are ACKs required

The log record describing a message is forced to stable storage before the message is sent

2PC protocol



2PC protocol

Notes

1. 2PC permits a participant (subordinate or coordinator) to unilaterally abort a transaction until it registers an affirmative vote
2. Once a participant votes, it cannot change its vote
3. Once in Wait state, it can either abort or commit
4. The global termination decision is taken by the coordinator
5. The participants enter certain states where they have to wait for messages from one another

2PC protocol

Blocking

There are various stages at which a server cannot progress its part of the protocol until it receives another message

If a server has voted Yes and is waiting for the decision of the coordinator

Timeouts may avoid the long waiting

2PC protocol

A transaction is officially committed at the time the coordinator's commit log record reaches stable storage

Subsequent failures cannot affect the outcome of the transaction

Log records contain:
- the type of the record
- the transaction id
- the identity of the coordinator

A coordinator's commit or abort also contains:
- the identities of the subordinates

2PC protocol

3(N – 1) messages (excluding the ACK messages)

2PC and Failures

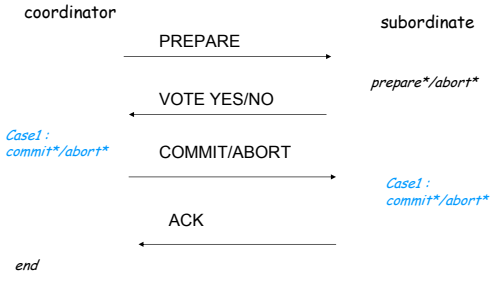
When a site comes back up after a crash

invokes a recovery process;

Reads the log and process all transactions that were executing the commit protocol

Coordinator or subordinate (how can this be determined?)

2PC and Failures



2PC and Failures

1. a **commit*** or **abort*** log record for transaction T

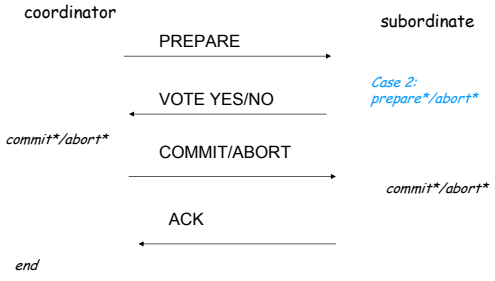
Respectively, Undo or Redo T

If the site is the coordinator:

periodically (why?) resend a commit or abort message to each subordinate (how does it knows them?) until receiving an ACK

after receiving ACKs from all, write an end log record for T

2PC and Failures



2PC and Failures

2. a **prepare*** log record for transaction T (but no commit or abort log records)

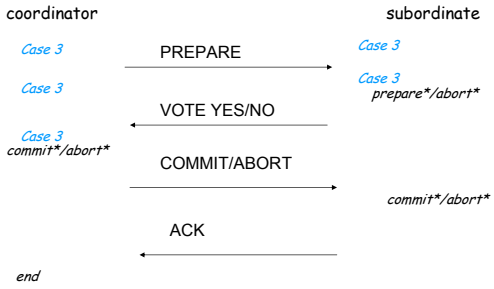
This site is a subordinate

Determine the coordinator from the prepare record

Repeatedly (why?) contact the coordinator site to determine the status of T (blocking)

Once the coordinator responds (with either a commit or abort), write a corresponding log record, redo or undo T

2PC protocol



2PC and Failures

3. No prepare*, commit*, or abort* log record for transaction T

No way to determine whether the site is the coordinator or a subordinate for T

Unilaterally decide to abort and undo T

If this site is the coordinator, may have sent a prepare to commit message, other sites might have voted, should respond with abort

2PC protocol

Blocking

If a server has voted YES and is waiting for the decision of the coordinator

T is blocked

Active subordinates communicate with each other, check whether at least one contains an abort* or commit* log record

Else, must wait for the coordinator (who also has a vote)

2PC and Failures

What a site should do if a site that it is communicating with fails?

Coordinator notices subordinate failure:

- If subordinate has not sent vote

  coordinator aborts the transaction

- If subordinate has not sent ACK

  coordinator hands transaction over to recovery process

2PC and Failures

Subordinate notices coordinator failure:

- If subordinate has not sent vote (not prepared)

  coordinator aborts the transaction

- If subordinate is in prepare state

  coordinator hands transaction over to recovery process to find out status

2PC Optimizations

Reduce

- the number of messages that are transmitted between the coordinator and the subordinates
- the number logs are written and their size

2PC with Presumed Abort

Presume Abort

Observation 1: The ACK messages are used to determine when a coordinator can forget about a transaction

Observation 2: If the coordinator fails after sending out PREPARE and before writing a commit* or abort* log record, when it comes up, it can unilaterally abort T (presume abort)

2PC with Presumed Abort

1. When a **coordinator** aborts a transaction T, it can undo T and remove it from the transaction table immediately
2. If a **subordinate** receives an ABORT, no need to send an ACK
3. The **coordinator** no need to record the names of the subordinates in the abort* log record
4. No need to force write an abort* log record; just append it to the log tail

2PC with Presumed Abort

Read-only transactions

Observation 3: If a subtransaction does no updates, it has no changes to either redo or undo; its commit or abort status is irrelevant.

2PC with Presumed Abort

1. If a subtransaction does no updates, the subordinate responds to a PREPARE message with a READER message – Writes no log records
2. When the coordinator receives a READER message, it treats it as a YES message, but it sends any more messages to the subordinate
3. If all subtransactions send a READER message, no need for the second phase of the commit protocol

2PC with Presumed Commit

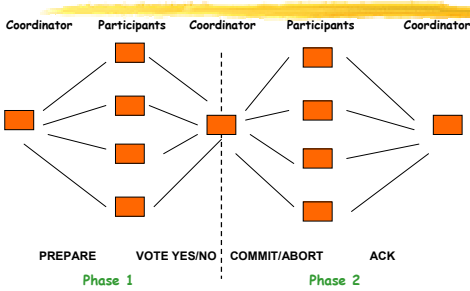
Observation: Transactions usually commit

If no information about the transaction exists, it should be considered committed

Cheaper to

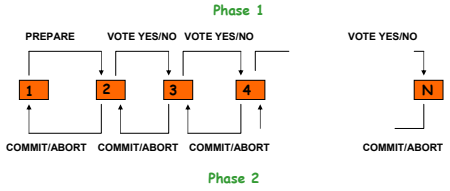
Require ACKs for aborts and eliminate ACKs for commit

Centralized 2PC



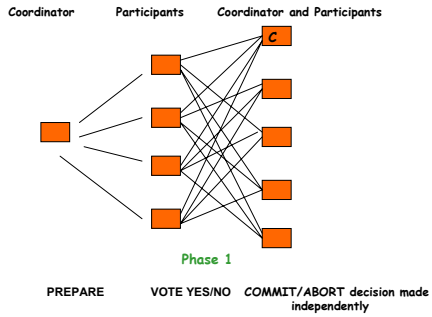
Centralized 2PC Communication Structure

Linear 2PC



Linear 2PC Communication Structure

Distributed 2PC



Communication Failures

If a communication line fails, in addition to losing the message(s) in transit, it might divide the network into two pr more disjoint groups.

This is called **network partitioning**

If the network is partitioned, the sites in each site continue to operate

Simple partitioning if the network is divided into only two components; otherwise it is called **multiple partitioning**

Περίληψη

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