Basi di dati distribuite

Motivazioni della distribuzione dei dati

- natura intrinsecamente **distribuita** delle organizzazioni
- evoluzione degli **elaboratori**
  - aumento della capacità elaborativa
  - riduzione di prezzo
- evoluzione della tecnologia dei **dbms**
- **standard** di interoperabilità

Tipologie di basi di dati distribuite

a **RETE :**

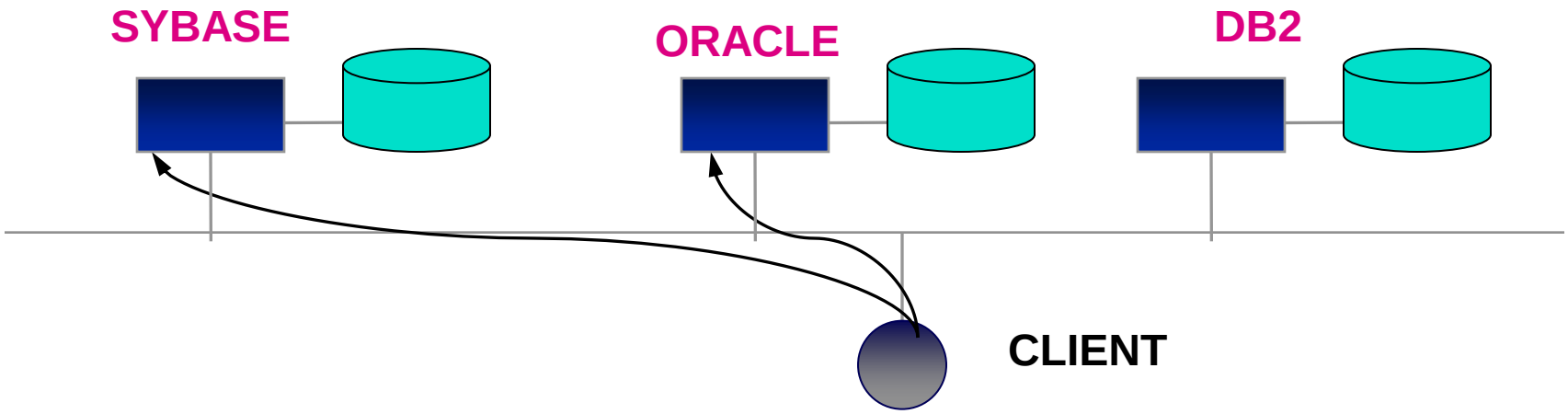
LAN (Local Area Network)

WAN (Wide Area Network)

b **DBMS :**

sistema omogeneo

sistema eterogeneo



Tipici esempi di applicazioni

	LAN	WAN
OMOGENEO	applicazioni gestionali e finanziarie	sistemi di prenotazione, applicazioni finanziarie
ETEROGENEO	applicazioni gestionali interfunzionali	sistemi di prenotazione integrati, sistemi interbancari

Problemi delle basi di dati distribuite

- **autonomia e cooperazione**
- **trasparenza**
- **efficienza**
- **affidabilità**

Frammentazione dei dati

scomposizione delle tabelle in modo da consentire la loro distribuzione

proprietà:

- completezza

- ricostruibilità

Frammentazione orizzontale

relazione

frammenti :

insiemi di tuple

completezza :

presenza

di tutte le tuple

ricostruzione :

unione

Frammentazione verticale

relazione

frammenti :

insiemi di attributi

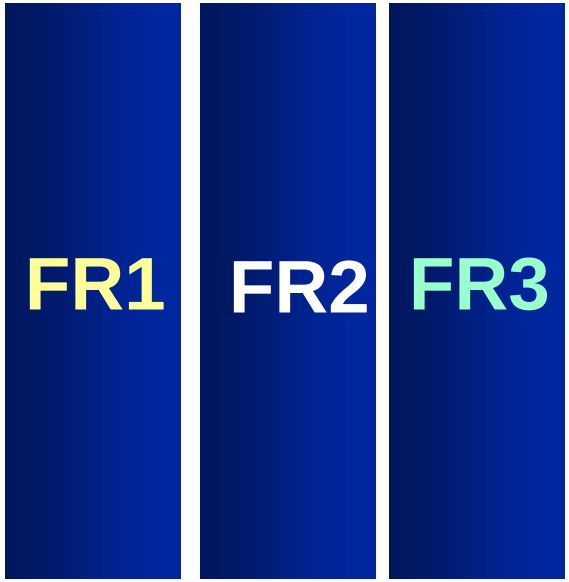
completezza :

presenza

di tutti gli attributi

ricostruzione :

join sulla chiave



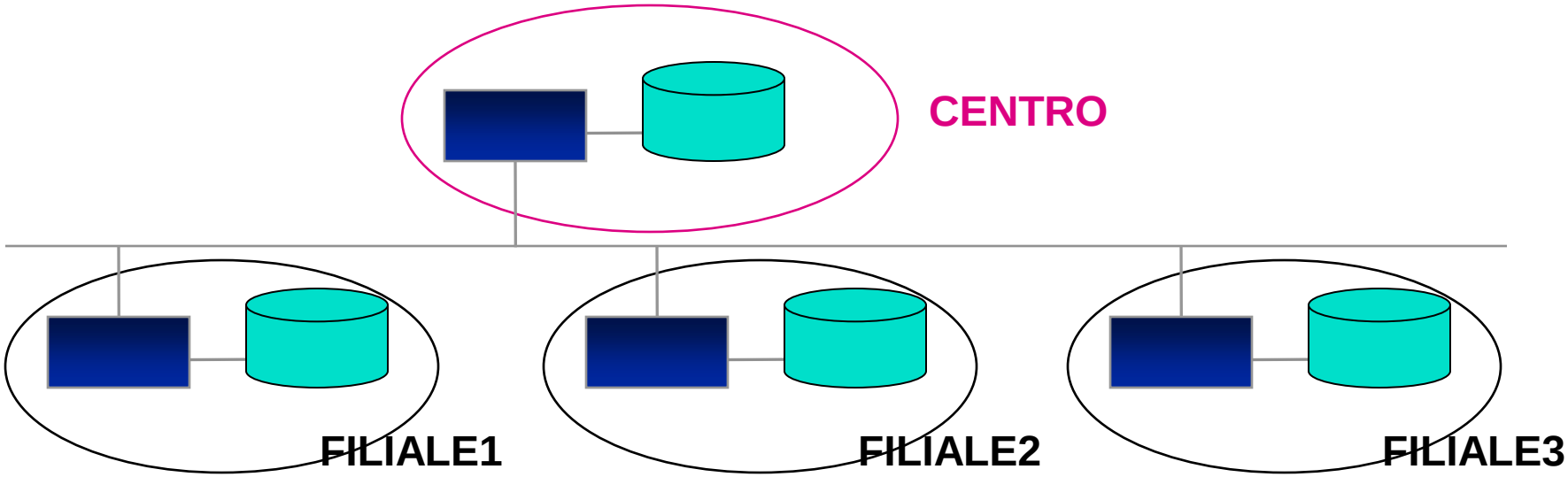
Esempio: conti correnti bancari

CONTO-CORRENTE

(NUM-CLI, NOME, FILIALE, SALDO)

TRANSAZIONE

(NUM-CLI, DATA, AMMONTARE, CAUSALE)



Frammentazione orizzontale

esempio:

CONTO1 = SELECT * FROM CONTO CORRENTE
WHERE FILIALE=1

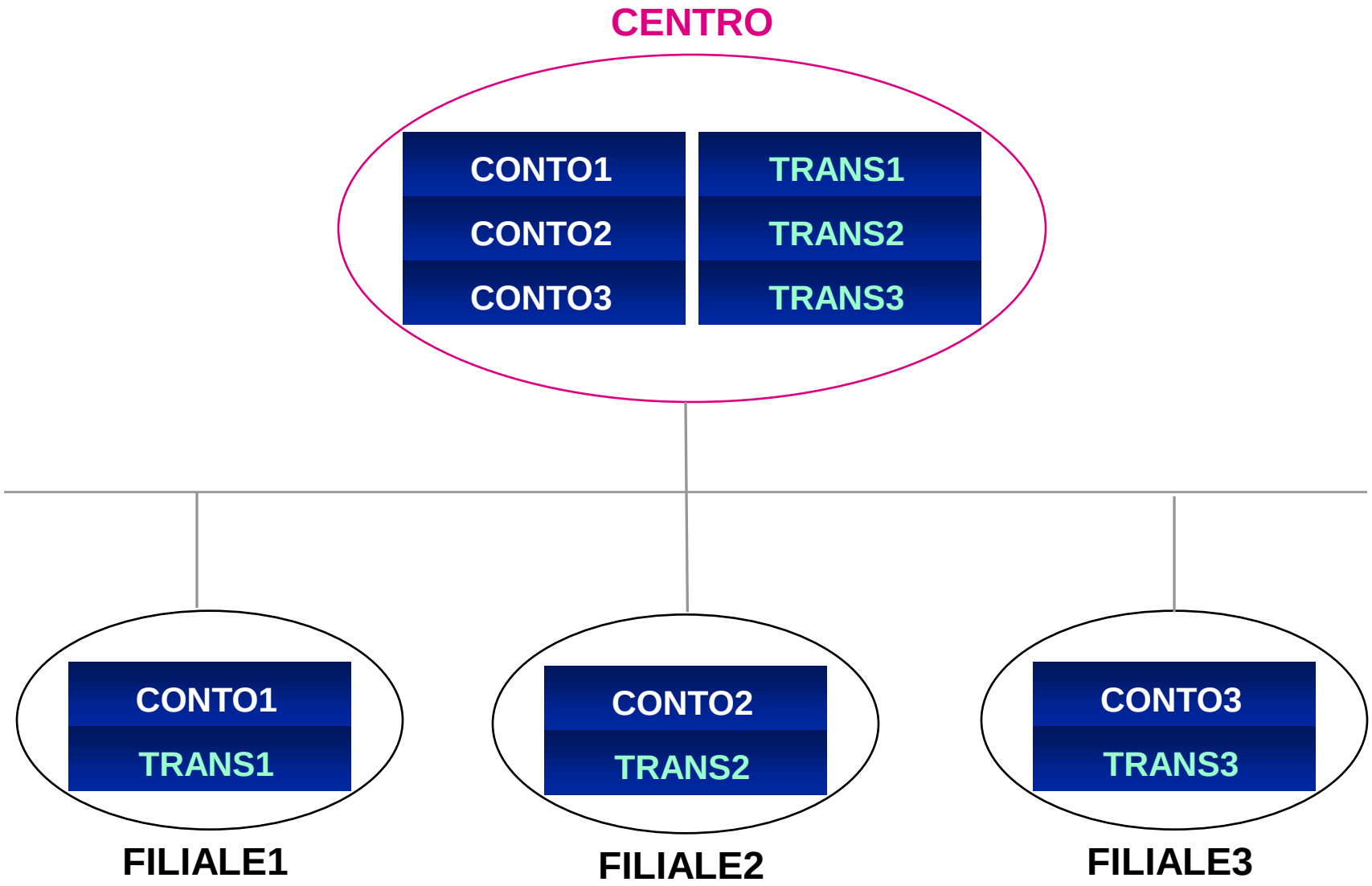
CONTO2 = …..

Frammentazione orizzontale derivata

esempio:

TRANS1 = SELECT
NUM-CLI, DATA, AMMONTARE, CAUSALE
FROM TRANS, CONTO1
WHERE TRANS. NUM-CLI = CONTO1. NUM-CLI

Allocazione dei frammenti



Livelli di trasparenza

FRAMMENTAZIONE

ALLOCAZIONE

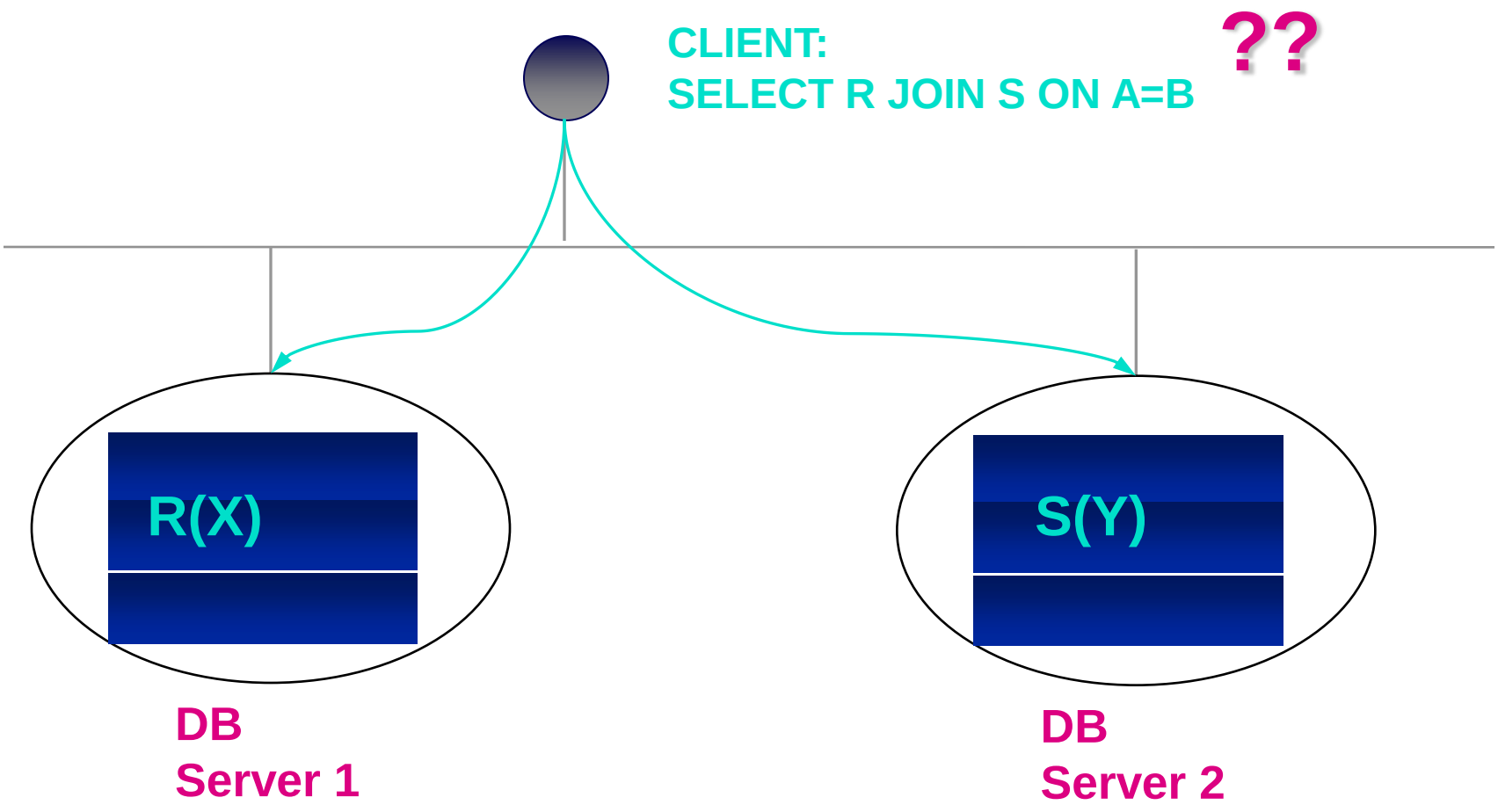
LINGUAGGIO

QUERY : estrarre il conto corrente del cliente 45

```
SELECT * FROM CONTO-CORRENTE WHERE NUM-CLI=45

SELECT * FROM CONTO1 WHERE NUM-CLI=45
UNION
SELECT * FROM CONTO2 WHERE NUM-CLI=45
UNION
SELECT * FROM CONTO3 WHERE NUM-CLI=45
```

Esecuzione query distribuite



Ottimizzazione query distribuite

- **Il Join coinvolge relazioni (o frammenti) che risiedono su siti diversi**
- **Come (e dove?) eseguire il Join?**
- **Occorre trasferire dei dati: i modelli di costo per l'ottimizzazione devono tenere conto dei costi di trasmissione:**

  C_0 = costo di avvio di una trasmissione

  C_1 = costo di trasmissione di una tupla

  $C_t(R) = C_0 + C_1 \times NT_R$
  = costo di trasmissione della relazione R

Join distribuito

- **Anche in ambiente distribuito l'esecuzione del join è l'operazione più onerosa**

- **Alcuni metodi molto usati per l'esecuzione del join distribuito si basano sull'operazione di semijoin e sulle sue proprietà algebriche**

- **Tali metodi consentono di minimizzare la quantità di dati che devono essere trasferiti (e quindi di ridurre i costi di trasmissione)**

Semi-Join: definizioni

- **Partendo dal join naturale r1 ▷◁ r2 fra due relazioni r1 e r2, istanze rispettivamente di R1(X1) e R2(X2), si possono definire due operazioni di semi-join come segue:**

$$r1 \triangleright\!< r2 = \pi_{X1}\,(r1 \triangleright\!\triangleleft r2)$$
$$r1 >\!\triangleleft r2 = \pi_{X2}\,(r1 \triangleright\!\triangleleft r2)$$

- **Il join "intero" può essere ricostruito come:**

$$r1 \triangleright\!\triangleleft r2 = (r1 \triangleright\!< r2) \triangleright\!\triangleleft r2 = (r1 >\!\triangleleft r2) \triangleright\!\triangleleft r1$$

Semijoin: esempi

Voli

Codice	Data	Comandante
AZ427	21/07/2001	Bianchi
LH427	23/07/2001	Rossi
TW056	21/07/2001	Smith

Linee

Codice	Partenza	Arrivo
AZ427	FCO	JFK
AF235	CDG	MPX
TW056	LAX	FCO

Voli ▷◁ Linee

Codice	Data	Comandante	Partenza	Arrivo
AZ427	21/07/2001	Bianchi	FCO	JFK
TW056	21/07/2001	Smith	LAX	FCO

Voli ▷< Linee

Codice	Data	Comandante
AZ427	21/07/2001	Bianchi
TW056	21/07/2001	Smith

Voli >◁ Linee

Codice	Partenza	Arrivo
AZ427	FCO	JFK
TW056	LAX	FCO

Metodo del semijoin

- **Esecuzione distribuita del Join $r \bowtie s$**
- **Sfrutta l'equivalenza algebrica con $(s \ltimes r) \bowtie r$**
- **L'algoritmo consta dei seguenti passi:**

1. **Sul sito di r: calcola $r1 := \pi_{X \cap Y}(r)$**
2. **Trasferisci r1 nel sito di s**
3. **Sul sito di s: calcola $s1 := s \bowtie r1$**
   (NB: s1 non è altro che il semijoin $s \ltimes r$)
4. **Trasferisci s1 nel sito di r**
5. **Sul sito di r: calcola $q := s1 \bowtie r$**

Convenienza del metodo

- **Con metodo "naïve"**

  (trasferimento dell'intera s sul sito di r

  dove viene eseguito il Join r ▷◁ s)

  ho costi di trasmissione:

  $Ct' = C_0 + C_1 \times NT_S$

- **Col metodo del semijoin**

  ho costi di trasmissione:

  $Ct'' = 2\,C_0 + C_1 \times (NT_{R1} + NT_{S1})$

  ➔ **Risulta + conveniente ($Ct'' < Ct'$) se:**

  $C_0 + C_1 \times (NT_{R1} + NT_{S1}) < C_1 \times NT_S$

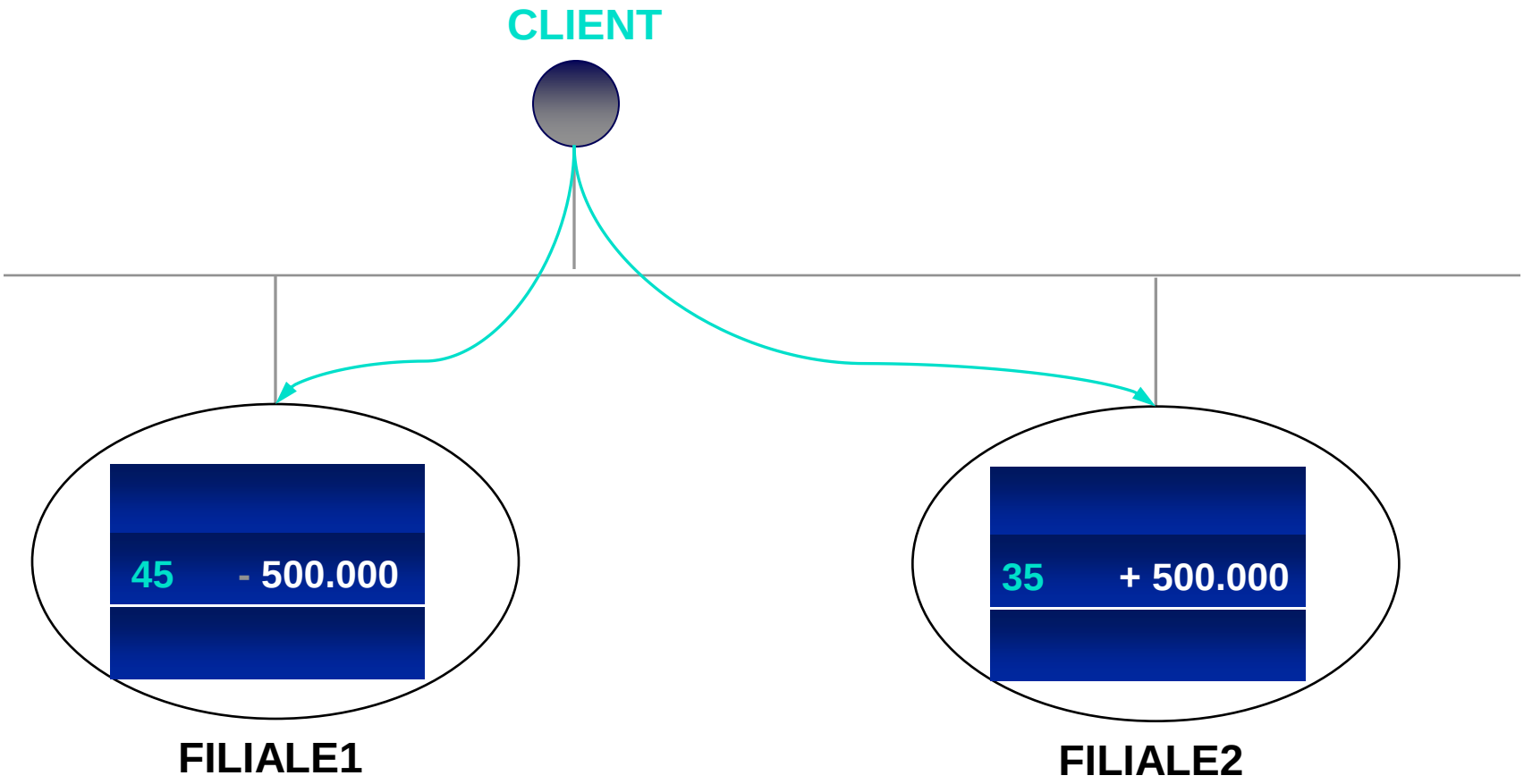
Conclusioni

- **Il metodo può essere facilmente esteso al caso di θ-join**
- **Presuppone che i costi di trasmissione siano più elevati rispetto a quelli di elaborazione locale (non si usa in sistemi basati su LAN con velocità paragonabile al transfer rate da disco a memoria centrale)**
- **Se sono molti gli attributi coinvolti dal join può essere comunque più vantaggioso trasferire l'intera relazione piuttosto che procedere al computo della proiezione**
- **Ci sono anche metodi più avanzati (es. Two-way semijoin)**

Transazioni distribuite

```
BEGIN TRANSACTION
    UPDATE CONTO1
    SET SALDO=SALDO - 500.000
    WHERE NUM-CLI=45;
    UPDATE CONTO2
    SET SALDO=SALDO + 500.000
    WHERE NUM-CLI=35;
COMMIT-WORK
```

Transazioni distribuite



Proprietà acid dell'esecuzione distribuita

- **isolamento**

  se ciascuna sottotransazione è a due fasi la transazione è globalmente serializzabile

- **persistenza**

  se ciascuna sottotransazione gestisce correttamente i log, i dati sono globalmente persistenti

- **correttezza**

  se ciascuna sottotransazione preserva l'integrità locale,

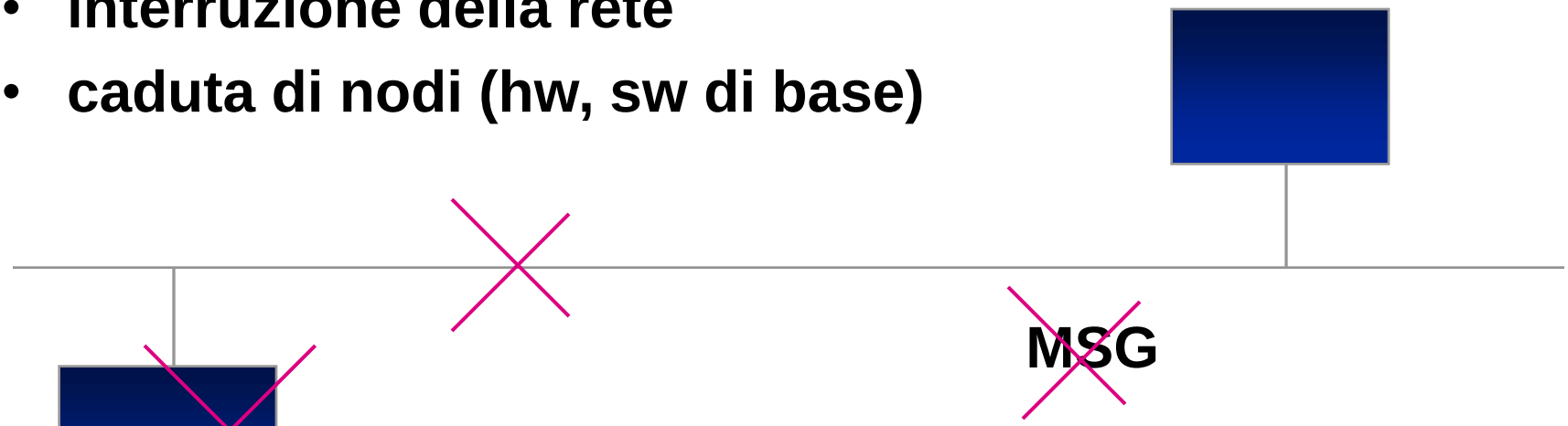
  i dati sono globalmente consistenti

- **atomicità**

  è il principale problema delle transazioni distribuite

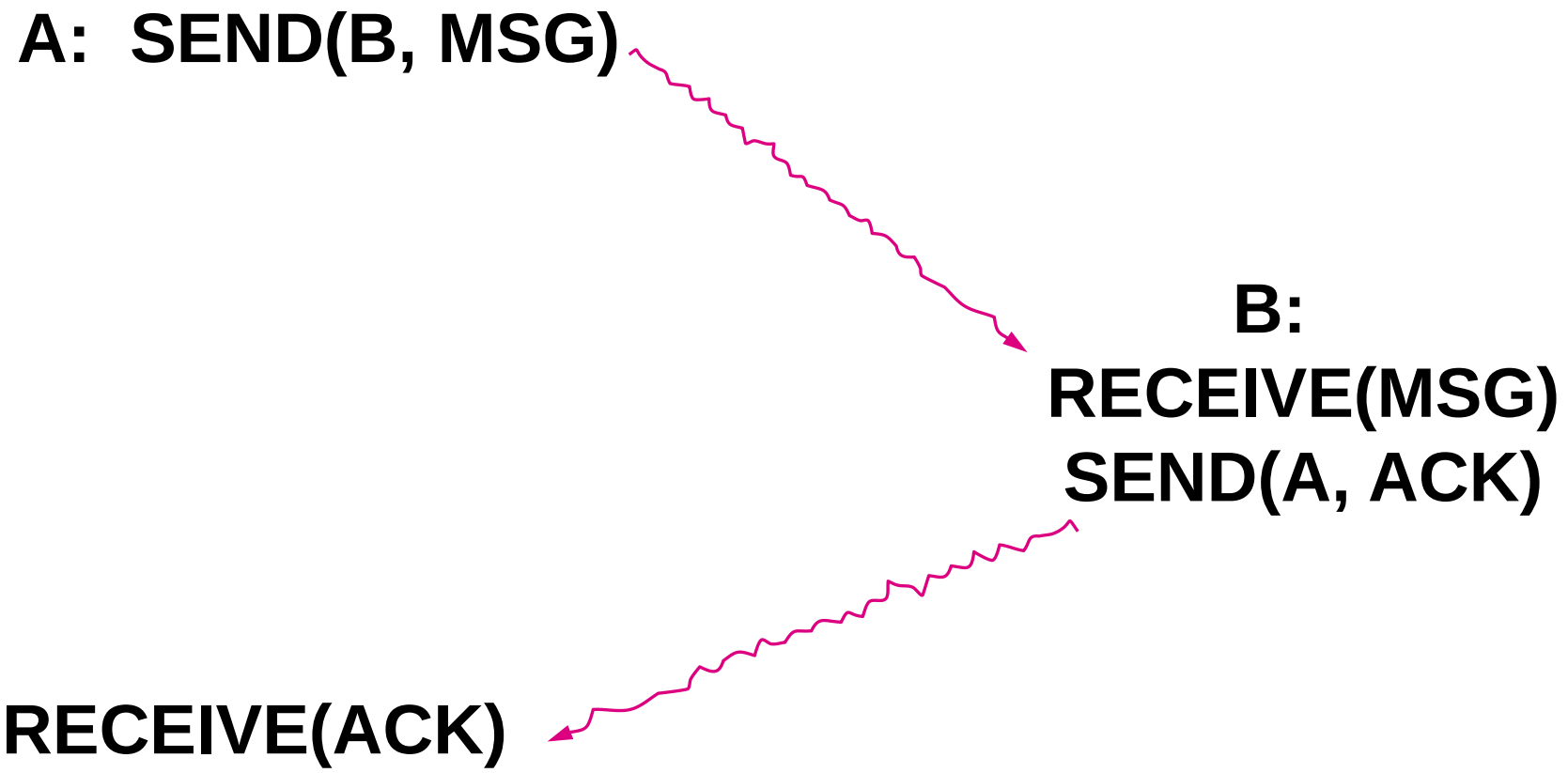
Guasti in un sistema distribuito

- **perdita di messaggi**
- **interruzione della rete**
- **caduta di nodi (hw, sw di base)**



- **errore nel sw applicativo**
- **violazione di vincoli**
- **altre cause di impossibilità di condurre a termine la sottotransazione**

Protocolli distribuiti e perdita di messaggi



SCAMBIO DI MESSAGGI TRA NODI

Il messaggio MSG parte da A :

- **se A non riceve un avviso (ACK) entro un tempo prefissato T**
- **A non è in grado di distinguere se:**
  - **MSG non è mai arrivato a B.**
  - **MSG è arrivato a B ma ACK non arriva ad A.**
- **È possibile dimostrare che nessun protocollo di lunghezza finita può eliminare questa incertezza.**
- **Ulteriori incertezze per A:**
  - **il nodo B e/o la linea sono guasti**
  - **il nodo B e/o la linea sono sovraccarichi**

Commit a due fasi (2-phase)

Anche se il recovery system di ogni nodo garantisce l'atomicità di ogni sottotransazione, il problema è che per garantire l'atomicità della TRANSAZIONE DISTRIBUITA, tutte le sottotransazioni devono terminare allo stesso modo (Abort o Commit GLOBALI).

È necessario un protocollo di recovery coordinato:

protagonisti

- **un coordinatore**
- **molti partecipanti**

come un contratto

- **fase uno: dichiarazione di intenti**
- **fase due: conclusione del contratto**

Record nel log del coordinatore

a. **PREPARE, identità dei partecipanti**
b. **GLOBAL COMM IT/ABORT, decisione**
c. **COMPLETE, termine del protocollo**

Record nel log del partecipante

a. **READY , dichiarazione positiva data**
b. **LOCAL COMMM IT/ABORT , decisione ricevuta**

CRITERI GENERALI nel 2-ph-com

1. **Nessuna sottotransazione può eseguire un commit locale in modo autonomo**

2. **Se una transazione abortisce ne viene data notizia (da parte del coordinatore) a tutti i nodi coinvolti che provocano l'abort di tutte le sottotransazioni.**

3. **Se nessuna sottotransazione abortisce localmente, un processo (nodo) COORDINATORE manda a tutte le sottotransazioni un ordine di "prepare to commit" registrando anticipatamente questo fatto sul suo LOG.**

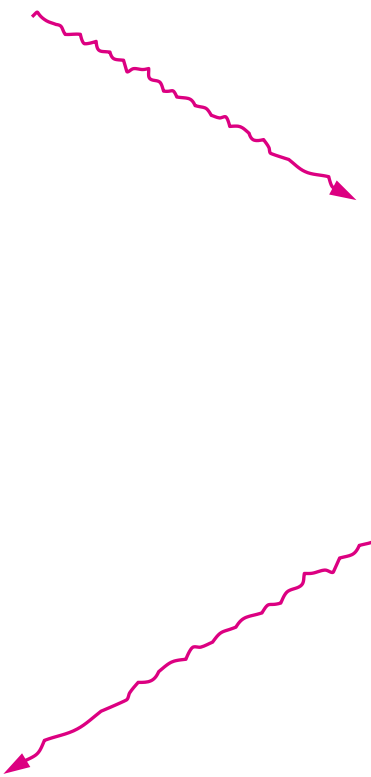
CRITERI GENERALI nel 2-ph-com

4 **Ogni transazione, se riceve l'ordine di prepare, entra in una fase di "prepare-to-commit" ed invia un ack (ready) al coordinatore, se non può avverte (not ready) il coordinatore che causa il 2. Lo stato di ready/not-ready è registrato nel LOG locale (sempre prima di ogni altra operazione)**

5 **Se il coordinatore riceve entro un tempo T (time-out) le risposte positive, invia a tutte le sottotransazioni l'ordine di commit registrando anticipatamente nel LOG che la transazione globale è terminata correttemente. Se invece non riceve le risposte positive in tempo utile manda un ordine di ABORT a tutte le sottotransazioni.**

Fase 1

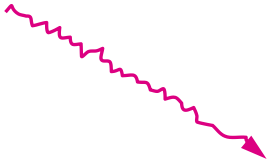
C: WRITE-LOG(PREPARE)
SET TIME-OUT
for all i: SEND (Pi, PREPARE)



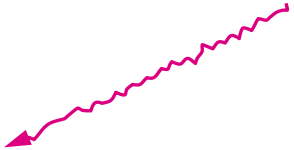
Pi: RECEIVE(C, PREPARE)
IF OK THEN MSG=READY
ELSE MSG=NO
WRITE-LOG(MSG)
SEND (C, MSG)

Fase 2

C: for all i: RECEIVE(Pi, MSGi)
IF TIME-OUT OR (one of MSGi=NO)
THEN DECISION=ABORT
ELSE DECISION=COMMIT
WRITE-LOG(DECISION)
for all i: SEND(Pi, DECISION)

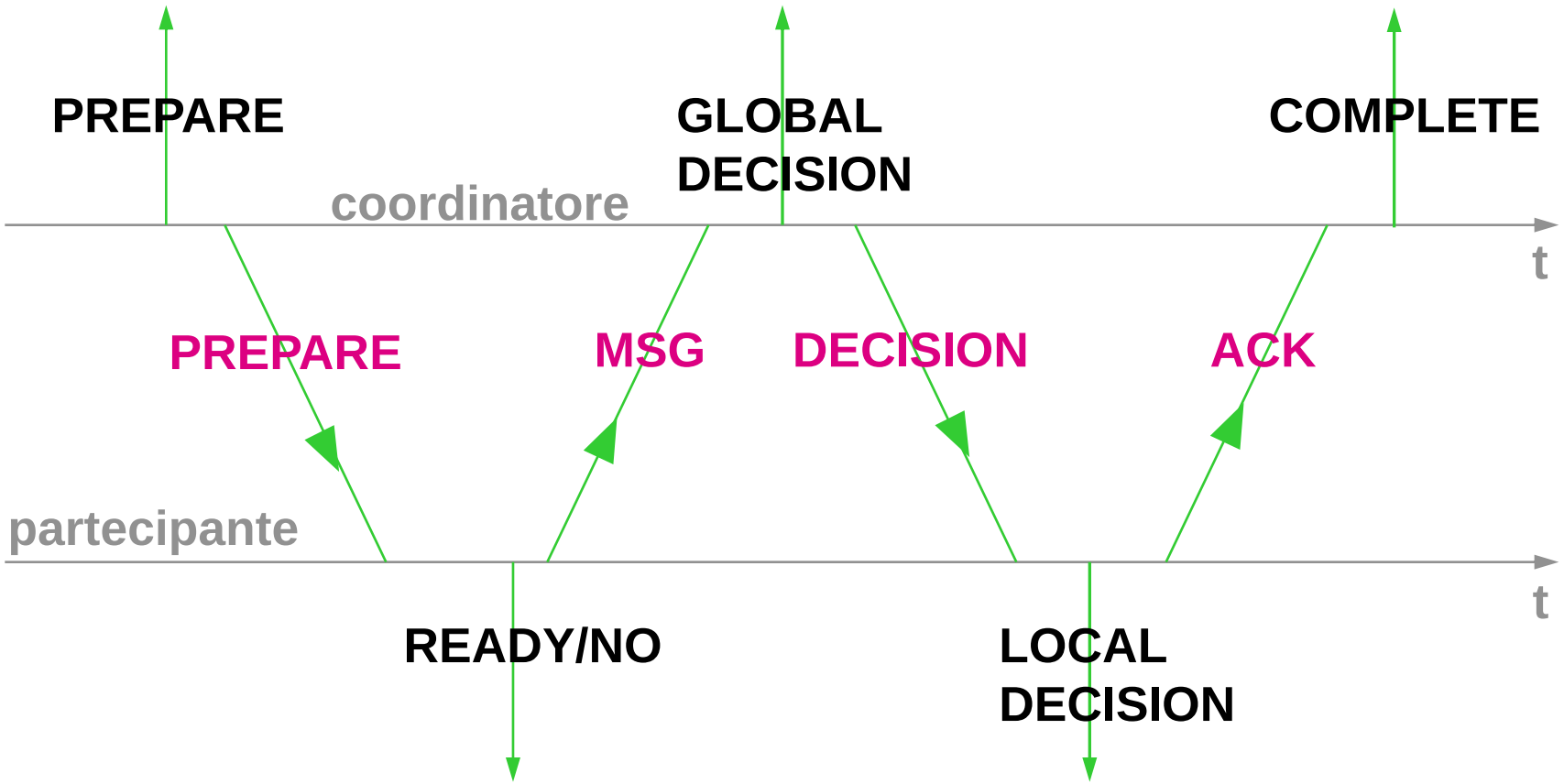


Pi: RECEIVE(C, DECISION)
WRITE-LOG(DECISION)
SEND (C, ACK)



C: for all i: RECEIVE(Pi, ACK)
WRITE-LOG(COMPLETE)

Diagramma del commit a due fasi



Comportamento in conseguenza ai guasti

Il protocollo si comporta in modo WAP (write ahead protocol) nei confronti delle operazioni sul LOG.

1) GUASTO NEL NODO DI UNA SOTTOTRANSAZIONE

Quando il sistema fa il recovery cerca nel log le sottotransazioni coordinate da nodi esterni e per ogni sottotransazione:

- **Se nel file LOG locale non c'è l'ordine di prepare la transazione abortisce e ne viene data notizia al coordinatore**
- **Se nel file LOG locale c'è l'ordine di prepare e la transazione chiede al coordinatore se deve eseguire il commit oppure no.**

Comportamento in conseguenza ai guasti

2) GUASTO NEL NODO COORDINATORE

Quando il sistema riparte il coordinatore guarda nel LOG e cerca le transazioni distribuite da lui coordinate ed il processo 2-phase-commit viene ripreso da dove era stato interrotto ripetendo l'ultima operazione effettuata perché non è possibile sapere se tutti i messaggi sono partiti.

Il protocollo 2-PHASE-COMMIT è "bloccante" nel senso che, se il coordinatore si guasta, i partecipanti devono attendere il suo recovery bloccando risorse utilizzabili altrimenti.

Comportamento in conseguenza ai guasti

3) PERDITA DI MESSAGGI E PARTIZIONAMENTO RETE

- **La perdita di un PREPARE o del successivo READY non sono distinguibili. C'e` time-out sulla prima fase e scatta il *global abort*.**
- **La perdita di un DECISION o del successivo ACK non sono distinguibili. C'e` time-out sulla seconda fase che pertanto viene ripetuta.**
- **Un partizionamento della rete non da problemi aggiuntivi:**
  - **se causa perdite di messaggi allora si torna ai casi precedenti**
  - **se riguarda parti della rete non coinvolte nella transazione, puo` essere ininfluente ai fini della transazione stessa**

MONITORAGGIO DELLA RETE

- **Per far partire le transazioni distribuite ogni nodo deve sapere se gli altri nodi sono in stato UP o DOWN.**
- **Si mantengono a questo scopo delle TABELLE DI STATO che contengono la situazione degli altri nodi.**
  - **Periodicamente ogni nodo invia agli altri un messaggio di "sono-UP", se il messaggio non arriva il nodo è classificato DOWN.**
  - **Un nodo può autoclassificarsi DOWN.**
  - **Un nodo può considerarsi UP ma gli altri lo considerano DOWN a causa di errori di comunicazione o interruzioni.**