BASI DI DATI 2

BASI DI DATI DISTRIBUITE E ARCHITETTURE CLIENT-SERVER

a.a 2024/2025

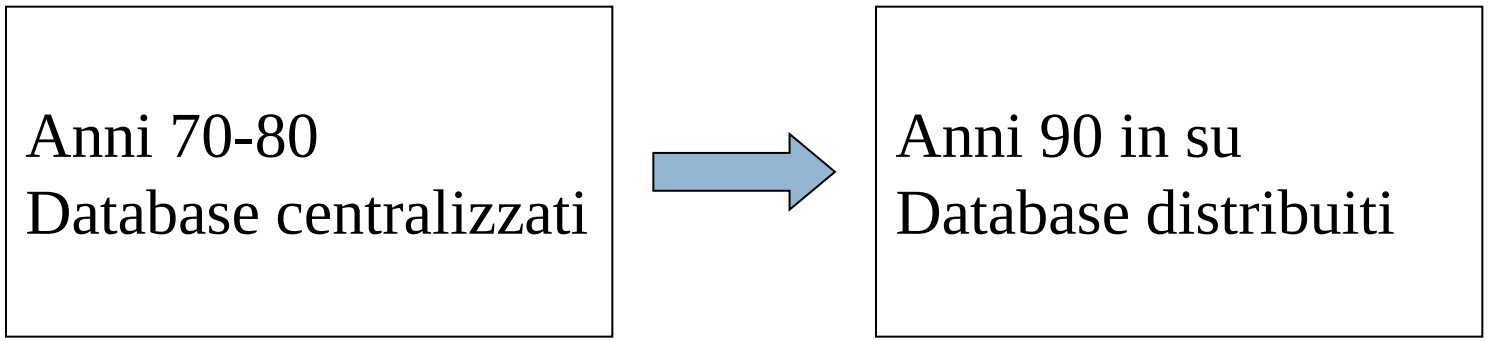
Prof.ssa G. Tortora

Obiettivi della lezione

- In questa lezione focalizzeremo la nostra attenzione su:
  - Database distribuiti (**DDB**).
  - Sistemi per la gestione di Database Distribuiti (**DDBMS**).
  - L'architettura Client-Server e i DDB.

La tecnologia per i DDB

- La tecnologia per i DDB è il risultato dell'unione di due tecnologie:
  - La tecnologia delle **basi di dati**.
  - La tecnologia delle **reti** e della **comunicazione** che ha avuto un'evoluzione enorme (tel. cellulari, Internet, …).



Introduzione ai DDB

- Utilizzare un DDB consente alle organizzazioni di:
  - effettuare un processo di decentralizzazione che viene raggiunta a livello di sistema,
  - e di integrare le informazioni a livello logico.

- *Un **database distribuito** è un singolo database logico che è sparso fisicamente attraverso computer in località differenti e connessi attraverso una rete di comunicazione dati.*

DDBMS

- Un **sistema per la gestione di basi di dati distribuite** (DDBMS) è un sistema software che gestisce un database distribuito rendendo la distribuzione trasparente all'utente.

- La rete deve consentire agli utenti di condividere i dati:
  - un utente della località **A** deve essere in grado di accedere (ed eventualmente aggiornare) i dati della località **B**.

- I computer possono andare dai micro-computer ai super-computer.

Tipi di architetture multiprocessore

- **Architettura a memoria condivisa *(shared memory)***:
  - diversi processori condividono sia la memoria primaria, sia la secondaria.

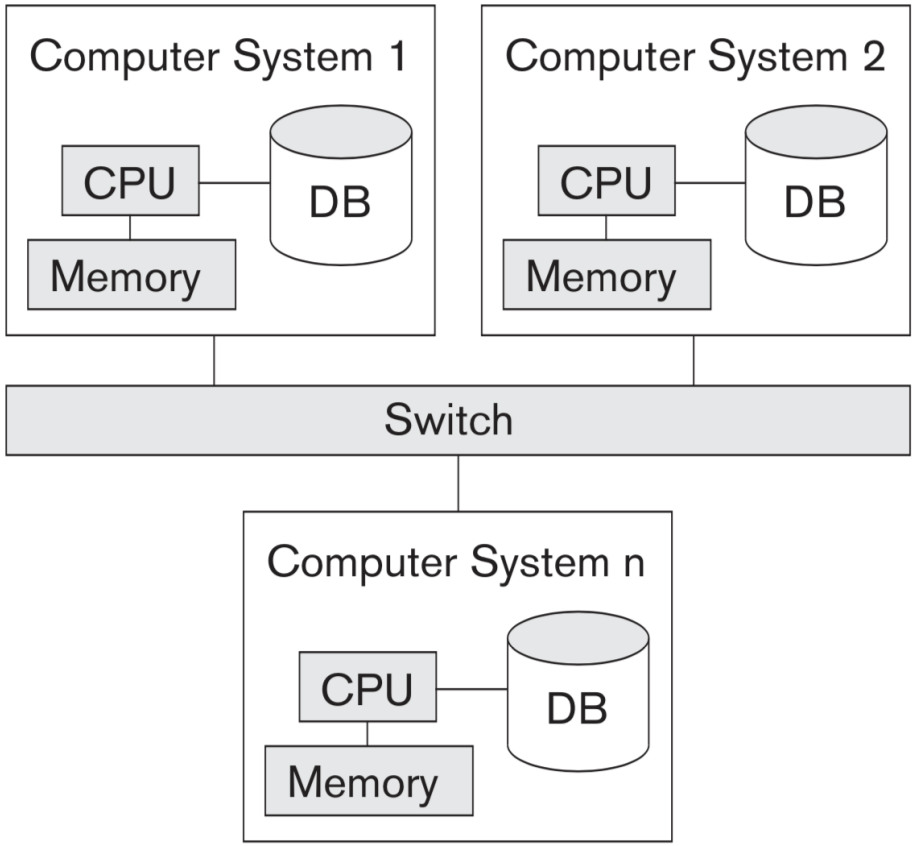
- **Architettura a disco condiviso *(shared disk)***:
  - i processori condividono la memoria secondaria, ma ognuno possiede la propria memoria primaria.

- Usando queste architetture i processori possono comunicare senza utilizzare la rete (meno overhead).
  - I DBMS che si basano su queste architetture si chiamano **Parallel DBMS**.

Architettura Shared Nothing

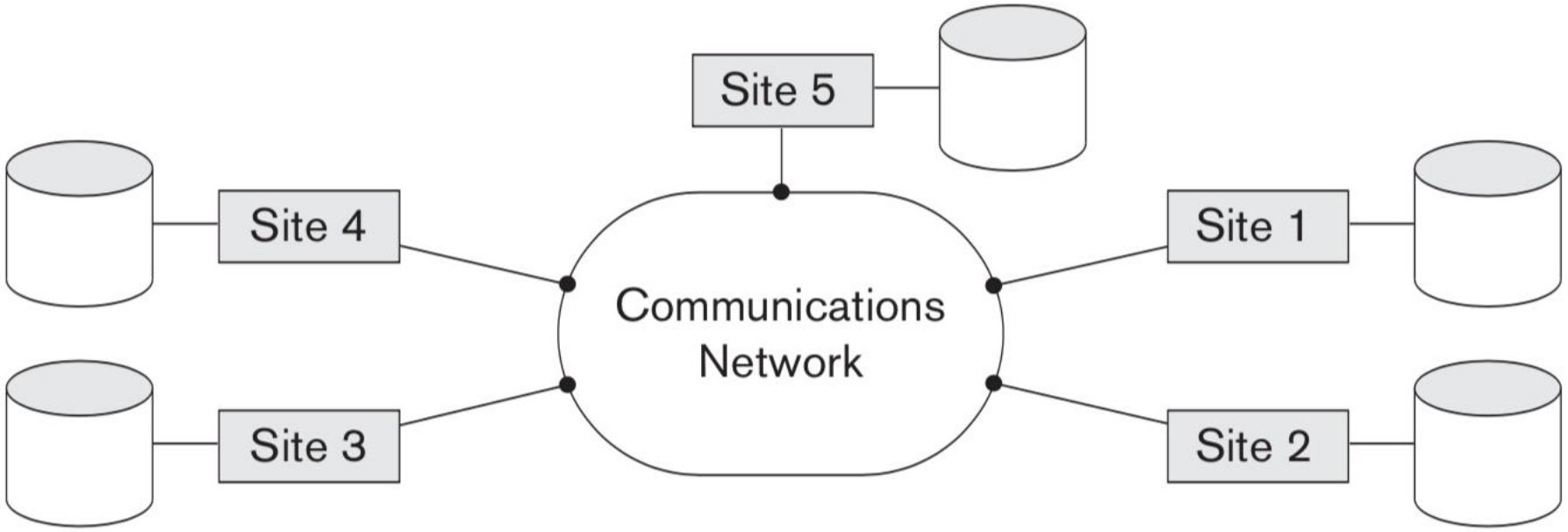
(architettura a memorie separate)

- Ogni processore possiede la sua memoria primaria e secondaria e comunicano attraverso la rete.
  - Non esiste memoria comune.
  - I nodi sono simmetrici ed omogenei.



Architettura di un DDB

- Un database distribuito è caratterizzato dall'**eterogeneità** dell'hardware e dei sistemi operativi di ogni nodo.



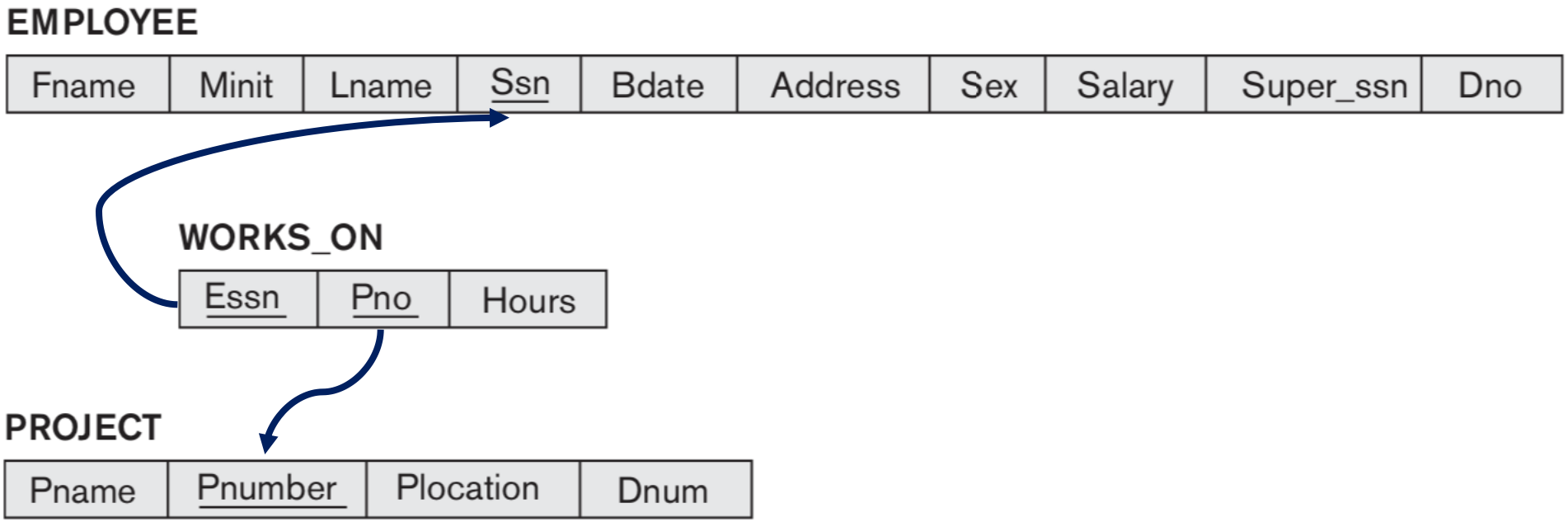
Tipi di DDB

- Un database distribuito richiede **DDBMS multipli**, funzionanti ognuno su di un sito remoto (nodo).

- Gli ambienti di database distribuiti si differenziano in base a:
  - Il grado di cooperazione dei DDBMS.
  - La presenza di un **sito master** che coordina le richieste che coinvolgono dati memorizzati in siti multipli.

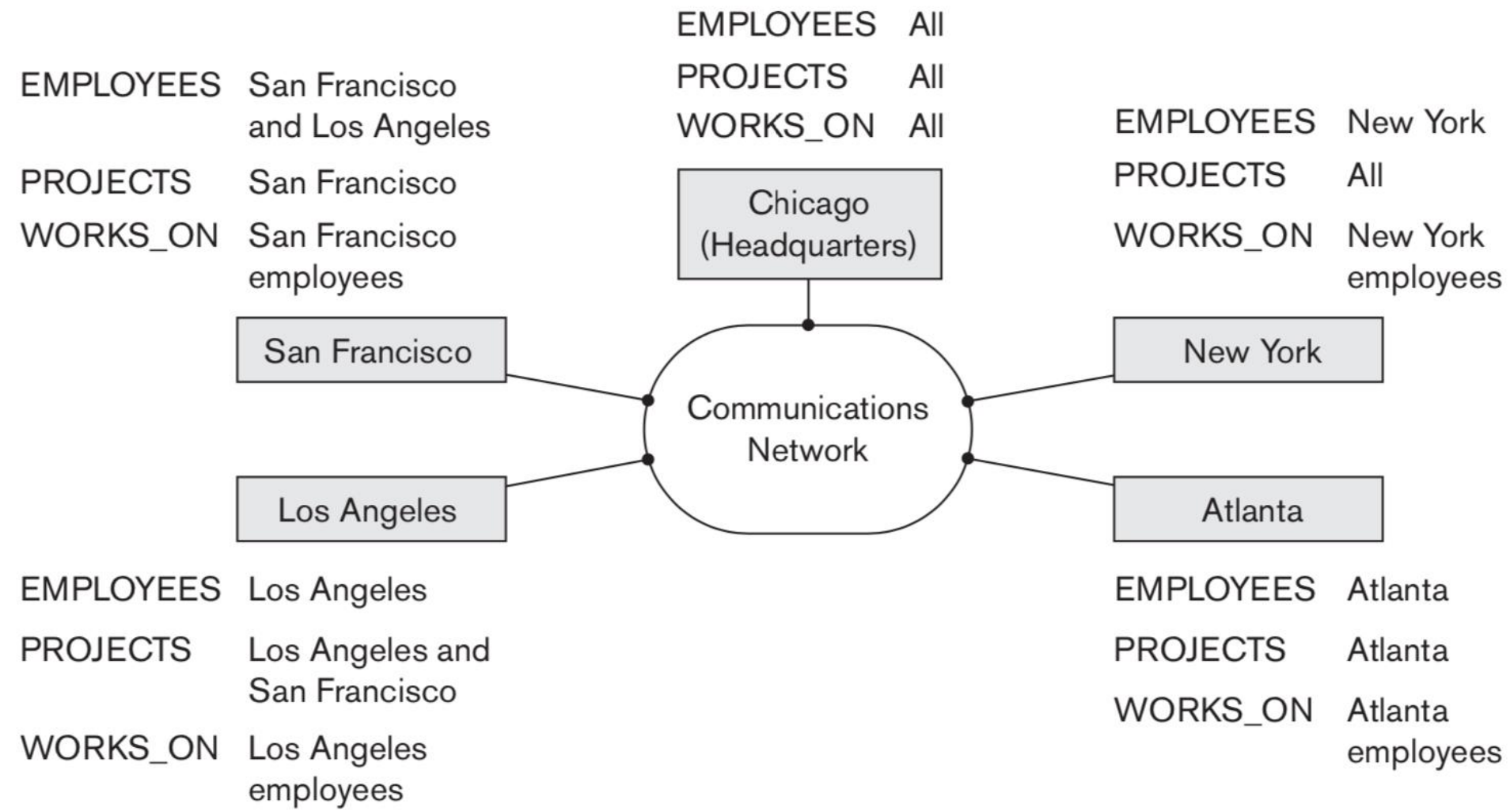
Vantaggi dei DDB

- Gestione dei dati distribuiti a diversi livelli di **trasparenza**:
  - sono nascosti i dettagli riguardo l'ubicazione di ogni data item presente nel sistema.



  - Le tabelle potrebbero essere frammentate orizzontalmente o memorizzate con replicazione.

Livelli di trasparenza



Tipi di trasparenza (2)

- Trasparenza di distribuzione o di rete:
  - Libertà dell'utente dai dettagli della rete.
- Si suddivide in:
  - **Trasparenza di locazione**: un comando utilizzato è indipendente dalla località dei dati e dalla località di emissione del comando.
  - **Trasparenza di naming**: la specifica di un nome di un oggetto implica che una volta che un nome è stato fornito, gli oggetti possono essere referenziati usando quel nome, senza dover fornire dettagli addizionali.

Tipi di trasparenza (3)

- **Trasparenza di replicazione**:
  - Copie dei dati possono essere mantenute in siti multipli per una migliore disponibilità, prestazioni ed affidabilità.
  - L'utente non percepisce l'esistenza delle copie.

- **Trasparenza di frammentazione**: Sono possibili due tipi di frammentazione:
  - La frammentazione **orizzontale** che distribuisce una relazione attraverso insiemi di tuple.
  - La frammentazione **verticale** che distribuisce una relazione in sottorelazioni formate da un sottoinsieme delle colonne della relazione originale.

- Una **query globale** dell'utente deve essere trasformata in una serie di **frammenti di query**.
- La trasparenza di frammentazione consente all'utente di non percepire l'esistenza di frammenti.

Vantaggi dei DDB: affidabilità e disponibilità

- **Affidabilità**: la probabilità che il sistema sia funzionante ad un certo istante.
- **Disponibilità**: la probabilità che il sistema è continuamente disponibile durante un intervallo di tempo.
- Quando un sito fallisce, gli altri possono continuare ad operare.
  - Solo i dati del sito fallito sono inaccessibili.
  - La replicazione dei dati aumenta ancora di più l'affidabilità e la disponibilità.

Vantaggi dei DDB: miglioramento delle prestazioni

- In un sistema distribuito è più facile espandere il sistema in termini di aggiungere nuovi dati, accrescere il numero di siti, o aggiungere nuovi processori.

Vantaggi dei DDB: espansione facilitata

- **La localizzazione dei dati**: i dati sono mantenuti nei siti più vicini a dove sono più utilizzati.
  - Ciò comporta una riduzione di accesso a reti geografiche.
  - Ogni sito ha un database di taglia più piccola e le query e le transazioni sono processate più rapidamente.
  - Le transazioni su ogni sito sono in numero minore rispetto ad un database centralizzato.

Funzionalità addizionali caratteristiche dei DDBMS

- Rispetto ai tradizionali DBMS, i DDBMS devono fornire le seguenti funzionalità:
  - Mantenere traccia dei dati, della loro distribuzione, frammentazione e replicazione nel catalogo.
  - Processare query distribuite: l'abilità di accedere a siti remoti e trasmettere query e dati tra i vari siti attraverso la rete di comunicazione.
  - Gestire le transazioni distribuite, query e transazioni devono accedere ai dati da più di un sito mantenendo l'integrità del DB.

Funzionalità addizionali caratteristiche dei DDBMS (2)

- Gestire la replicazione dei dati: decidere quando replicare un dato e mantenere la consistenza fra le copie.
- Gestire il recovery di un DDB: fallimento di uno dei siti e dei link di comunicazione.
- Sicurezza, le transazioni distribuite devono essere gestite garantendo la sicurezza dei dati e l'accesso degli utenti.
- Gestire il catalogo distribuito, il catalogo deve contenere i dati dell'intero DB ed essere globale.
  - Deve essere deciso come e dove distribuire il catalogo.

Funzionalità addizionali caratteristiche dei DDBMS: il livello fisico

- A livello hardware, si devono considerare le seguenti differenze rispetto ai DBMS:
  - Esistono diversi computer, detti siti o nodi.
  - I siti devono essere connessi attraverso una rete di comunicazione per trasmettere dati e comandi fra i siti.

- I siti possono essere connessi attraverso una rete locale e/o distribuiti geograficamente e connessi ad una rete geografica.

Il Design dei DDB

- Il design dei DDB prevede l'applicazione di tecniche relative a:
  - Frammentazione dei dati.
  - Replicazione dei dati.
  - Allocazione dei frammenti.
- Le informazioni relative saranno memorizzate nel catalogo.

Frammentazione dei dati

- Deve essere deciso quale sito deve memorizzare quale porzione del DB.
- Assumiamo che non ci sia replicazione dei dati.

- Una relazione può:
  - Essere memorizzata per intero in un sito.
  - Essere divisa in unità più piccole distribuite.

Lo schema del database ‘Company’

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATION

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPARTMENT_NAME	SEX	BDATE	RELATIONSHIP

Frammentazione orizzontale

- Un frammento orizzontale di una relazione è un sottoinsieme delle tuple della relazione.

- Le tuple vengono assegnate ad un determinato frammento orizzontale in base al valore di uno o più attributi.

- ***Es:***

  (Considerando il database “Company”) lo distribuiamo su **3** siti e definiamo tre frammenti orizzontali in base al valore del numero del dipartimento **(DNO=5) (DNO=4) (DNO=1)**.

  Ogni frammento contiene le tuple degli impiegati che lavorano per un particolare dipartimento.

La segmentazione orizzontale

- La segmentazione orizzontale divide una relazione raggruppando righe per creare sottoinsiemi di tuple, ciascuno con un significato logico.
- La segmentazione orizzontale **derivata** applica la partizione di una relazione primaria anche a relazioni secondarie, collegate alla prima con una chiave esterna.
- ***Es:***

  dal partizionamento di DEPARTMENT deriva il partizionamento di EMPLOYEE e PROJECT.

La frammentazione verticale

- Divide una relazione "*verticalmente*" per colonne.
- Un frammento verticale di una relazione mantiene solo determinati attributi di una relazione.

- ***Es:***

  EMPLOYEE può essere suddiviso in due frammenti:
  - Uno contenente le informazioni personali {NAME, BDATE, ADDRESS e SEX}.
  - L'altro contenente {SSN, SALARY, SUPERSSN, DNO}.

La frammentazione verticale (2)

- La frammentazione dell'esempio precedente non consente di ricostruire la tupla EMPLOYEE originale:
  - non ci sono attributi in comune fra i due frammenti.
- ***Soluzione:*** *aggiungere SSN agli attributi personali.*

- È necessario includere la chiave primaria o una chiave candidata in ogni frammento verticale.

La frammentazione orizzontale e l'algebra relazionale

- Ogni frammento orizzontale su di una relazione **R** può essere specificato attraverso un'operazione $\sigma_{Ci}(R)$.

- Un insieme di frammenti orizzontali tale che le condizioni $C_1, \ldots, C_n$ includono tutte le tuple della relazione è detto frammentazione orizzontale completa.
- In molti casi i frammenti sono disgiunti:
  - Nessuna tupla in **R** soddisfa C_i **AND** C_J per $i \neq j$.

La frammentazione verticale e l'algebra relazionale

- Un frammento verticale può essere specificato da una operazione dell'algebra relazionale $\pi_{Li}(R)$.

- Un insieme di frammenti verticali la cui lista di proiezioni $L_1, \ldots, L_n$ include tutti gli attributi di **R** e condivide solo la chiave primaria è detta frammentazione verticale completa.

Caratteristiche della frammentazione verticale completa

- Devono essere soddisfatte le seguenti condizioni:
  - $L_1 \cup L_2 \cup ... \cup L_n$ = ATTRIBUTI (R)
  - $L_i \cap L_j$ = PK(R) per ogni i ≠ j

- Per ricostruire la relazione R dai frammenti verticali è necessario fare l'OUTER JOIN dei frammenti.
- ***Es:***

  i due frammenti L_1={NAME, ADDRESS}

  L_2={SSN, NAME, SALARY}

  non costituiscono un frammento verticale completo.

Frammentazione Mista

- È possibile combinare la frammentazione orizzontale e verticale.
- ***Es:*** combiniamo le due frammentazioni viste per la relazione EMPLOYEE.
- La relazione originaria può essere ricostruita applicando OUTER JOIN e UNION nell'ordine appropriato.

Schema di frammentazione

- Uno **schema di frammentazione** di un DB è la definizione di un insieme di frammenti che include tutte le tuple e gli attributi del DB e consente la ricostruzione dell'intero DB applicando una sequenza di operazioni di OUTER JOIN e UNION.

Schema di allocazione

- Lo **schema di allocazione** descrive l'allocazione dei frammenti ai siti del DB.
- Associa ad ogni frammento il sito in cui deve essere memorizzato.
- Un frammento memorizzato su più di un sito si dice replicato.

Replicazione ed allocazione di dati

- La replicazione consente di migliorare la disponibilità dei dati.
- Modalità di replicazione:
  - Replicare l'intero DB in ogni sito, migliora la prestazione per le query, overhead eccessivo per l'aggiornamento dei dati, controllo della concorrenza e recovery complessi.
  - Nessuna replicazione, frammenti disgiunti (eccetto per la chiave).
  - Replicazione parziale, soluzioni intermedia: alcuni frammenti possono essere replicati ed altri no.
- ***Es:*** lavoratori con PC wireless (venditori), memorizzano sul proprio PC parte del DB e la sincronizzano periodicamente con il DB server.

Schema di replicazione

- Lo **schema di replicazione** è una descrizione della replicazione dei frammenti.

- Ogni segmento deve essere assegnato ad un sito del DB.
  - Questo processo si chiama allocazione dei dati.

- La scelta dei siti e del grado di replicazione dipende da:
  - Prestazioni.
  - Obiettivi di disponibilità del sistema.
  - Tipo e frequenza delle transazioni sottomesse ad ogni sito.

Criteri per distribuire i dati

- Trovare una soluzione ottima alla distribuzione dei dati è un problema difficile:
  - Se è richiesta un'alta disponibilità e le transazioni possono essere sottomesse ad ogni sito e molte transazioni sono solo di retrieval, si può adottare una soluzione di replicazione totale.
  - Se alcune transazioni che accedono a porzioni del DB sono sottomesse solo da particolari siti, su quei siti possono essere allocati i frammenti corrispondenti.
  - Se sono richiesti molti aggiornamenti è conveniente ridurre la replicazione dei dati.

Esempi di frammentazione, allocazione e replicazione

- Supponiamo che un'organizzazione ("Company") abbia tre siti, uno per ogni dipartimento.

- Il **sito 2** e il **sito 3** sono relativi ai dipartimenti 5 e 4, rispettivamente.
  - Da ognuno di questi siti si accede di frequente ai dati associati agli impiegati e ai progetti dei dipartimenti corrispondenti.
  - Assumiamo che i siti 2 e 3 accedano soprattutto alle informazioni NAME, SSN, SALARY, SUPERSSN di EMPLOYEE.

- Il **sito 1** è usato dagli amministratori per accedere ai dati dell'intera compagnia regolarmente.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Esempio: la base di dati relazionale "COMPANY".

Esempi di frammentazione, allocazione e replicazione (2)

- L'intero db può essere memorizzato nel **sito1**.
- I frammenti del **sito 2** e **sito 3** sono determinati frammentando orizzontalmente DEPARTMENT in base alla chiave DNUMBER.
- Applichiamo la frammentazione derivata alle relazioni EMPLOYEE, PROJECT e DEPT_LOCATION basate sulla chiave esterna che referenzia il dipartimento.
- Frammentiamo verticalmente la relazione EMPLOYEE per includere solo gli attributi {NAME, SSN, SALARY, SUPERSSN, DNO}.

Esempi di frammentazione, allocazione e replicazione (3)

- Problemi con la relazione WORKS_ON (*M:N*):
  - WORKS_ON(ESSN, PNO, Hours) non ha per attributo il numero di dipartimento.

- Ogni tupla collega un impiegato ed un progetto.
- Possiamo frammentarla in base:
  - Al dipartimento per cui l'impiegato lavora.
  - Al dipartimento che controlla il progetto.

- Scegliamo l'OR delle ipotesi.

Dati al sito 2

EMPD_5

Fname	Minit	Lname	Ssn	Salary	Super_ssn	Dno
John	B	Smith	123456789	30000	333445555	5
Franklin	T	Wong	333445555	40000	888665555	5
Ramesh	K	Narayan	666884444	38000	333445555	5
Joyce	A	English	453453453	25000	333445555	5

DEP_5

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22

DEP_5_LOCS

Dnumber	Location
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON_5

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0

PROJS_5

Pname	Pnumber	Plocation	Dnum
Product X	1	Bellaire	5
Product Y	2	Sugarland	5
Product Z	3	Houston	5

Data at site 2

Dati al sito 3

EMPD_4

Fname	Minit	Lname	Ssn	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	25000	987654321	4
Jennifer	S	Wallace	987654321	43000	888665555	4
Ahmad	V	Jabbar	987987987	25000	987654321	4

DEP_4

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Administration	4	987654321	1995-01-01

DEP_4_LOCS

Dnumber	Location
4	Stafford

WORKS_ON_4

Essn	Pno	Hours
333445555	10	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0

PROJS_4

Pname	Pnumber	Plocation	Dnum
Computerization	10	Stafford	4
New_benefits	30	Stafford	4

Data at site 3

Relazione WORKS_ON

a) Frammenti di WORKS_ON per gli impiegati che lavorano nel dipartimento 5:

Employees in Department 5

G1

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0

C1 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 5))

G2

Essn	Pno	Hours
333445555	10	10.0

C2 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 4))

G3

Essn	Pno	Hours
333445555	20	10.0

C3 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 1))

C = [ESSN **IN** (**SELECT** SSN **FROM** EMPLOYEE **WHERE** DNO=5)]

Relazione WORKS_ON (2)

b) Frammenti di WORKS_ON per gli impiegati che lavorano nel dipartimento 4:

Employees in Department 4

G4

Essn	Pno	Hours

C4 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 5))

G5

Essn	Pno	Hours
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0

C5 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 4))

G6

Essn	Pno	Hours
987654321	20	15.0

C6 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 1))

C = [ESSN **IN** (**SELECT** SSN **FROM** EMPLOYEE **WHERE** DNO=4)]

Relazione WORKS_ON (3)

c) Frammenti di WORKS_ON per gli impiegati che lavorano nel dipartimento 1:

Employees in Department 1

G7

Essn	Pno	Hours

C7 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 5))

G8

Essn	Pno	Hours

C8 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 4))

G9

Essn	Pno	Hours
888665555	20	Null

C9 = C and (Pno in (SELECT Pnumber FROM PROJECT WHERE Dnum = 1))

C = [ESSN **IN** (**SELECT** SSN **FROM** EMPLOYEE **WHERE** DNO=1)]

Allocazione dei segmenti

- L'unione dei frammenti G1, G2, G3, G4 e G7 sono allocati nel **sito 2**.
- L'unione dei frammenti G2, G4, G5, G6, e G8 sono allocati nel **sito 3**.
  - I frammenti G2 e G4 sono replicati in entrambi i siti.

- Questa strategia di allocazione permette di eseguire il JOIN tra i frammenti locali EMPLOYEE o PROJECT dei siti 2 e 3 e il frammento locale WORKS_ON completamente in locale.
  - **Es:** Per il sito 2, l'unione dei frammenti G1, G4 e G7 restituisce tutte le tuple di WORKS_ON relativi ai progetti controllati dal dipartimento 5.
  - Per il sito 3 vale lo stesso per i segmenti G2, G5 e G8.

Allocazione dei segmenti (2)

(a)

G1	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0

C1=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=5))

G2	ESSN	PNO	HOURS
	333445555	10	10.0

C2=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=4))

G3	ESSN	PNO	HOURS
	333445555	20	10.0

C3=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=1))

Frammenti del sito 2

Employees in Department 5

(b)

G4	ESSN	PNO	HOURS

C4=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=5))

G5	ESSN	PNO	HOURS
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0

C5=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=4))

G6	ESSN	PNO	HOURS
	987654321	20	15.0

C6=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=1))

Employees in Department 4

(c)

G7	ESSN	PNO	HOURS

C7=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=5))

G8	ESSN	PNO	HOURS

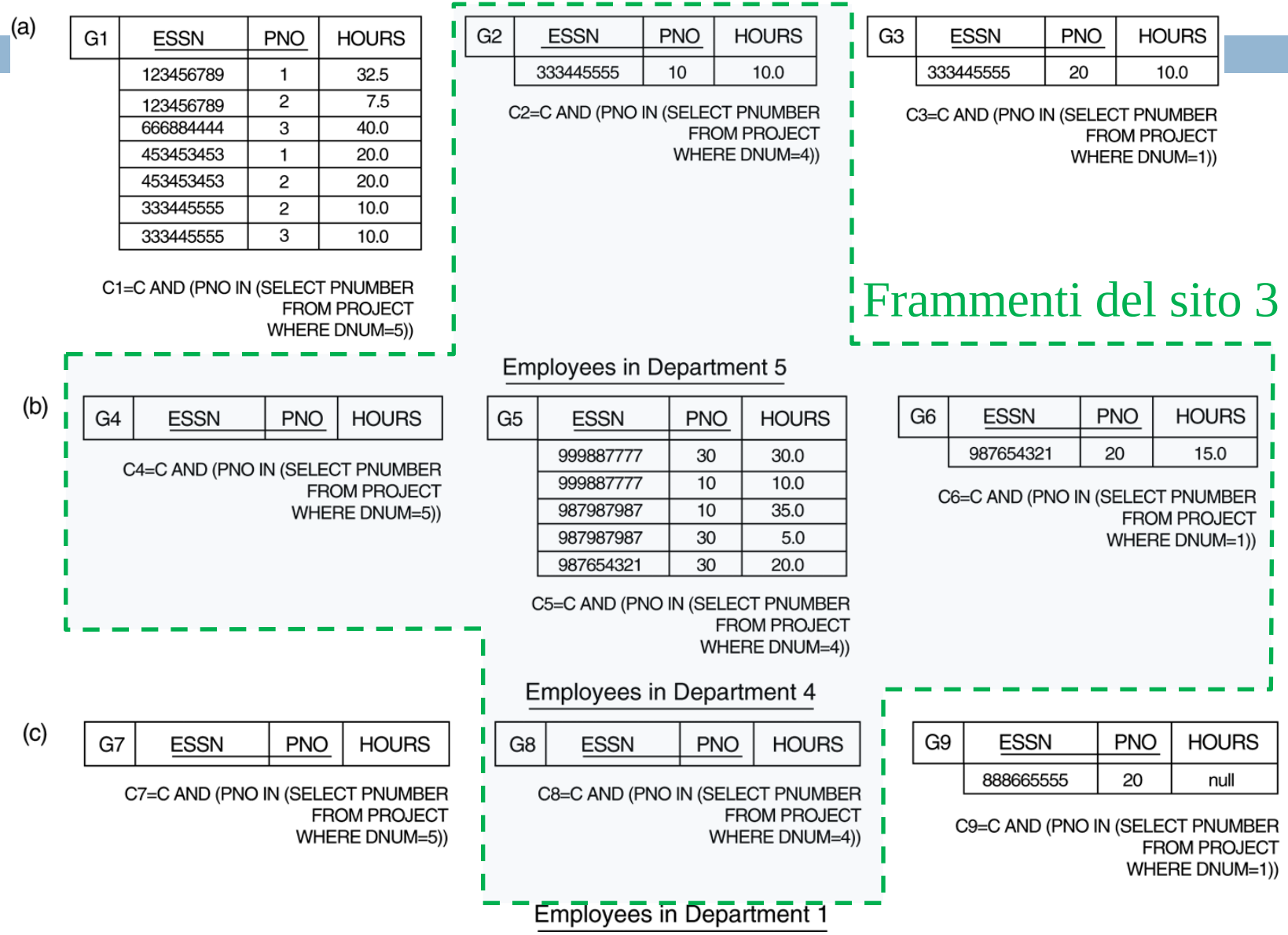
C8=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=4))

G9	ESSN	PNO	HOURS
	888665555	20	null

C9=C AND (PNO IN (SELECT PNUMBER FROM PROJECT WHERE DNUM=1))

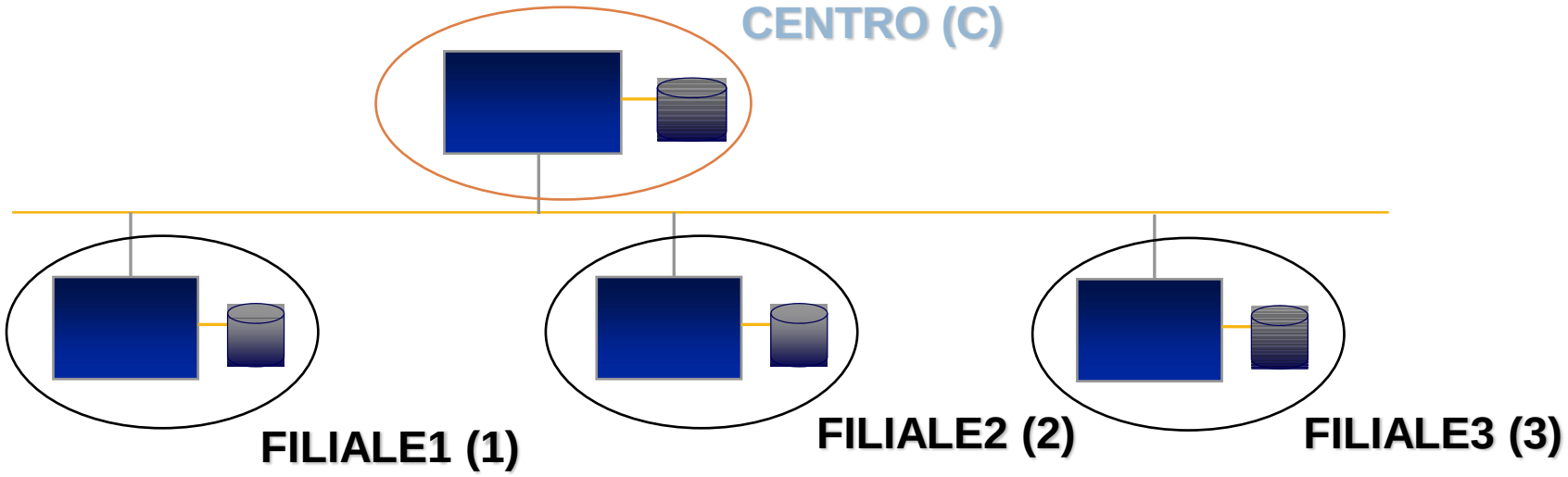
Employees in Department 1

Allocazione dei segmenti (3)



Esempio: Conti correnti bancari

- **CONTO-CORRENTE** (NUM-CC, NOME, FILIALE, SALDO)
- **TRANSAZIONE** (NUM-CC, DATA, PROGR, AMMONTARE, CAUSALE)



Frammentazione orizzontale principale

- $R_i = \sigma_{Pi}(R)$

- ***Es:***
  - CONTO1 = $\sigma_{Filiale=1}$ (CONTO-CORRENTE)
  - CONTO2 = $\sigma_{Filiale=2}$ (CONTO-CORRENTE)
  - CONTO3 = $\sigma_{Filiale=3}$ (CONTO-CORRENTE)

Frammentazione orizzontale derivata

- $S_i = S \rtimes R_i$

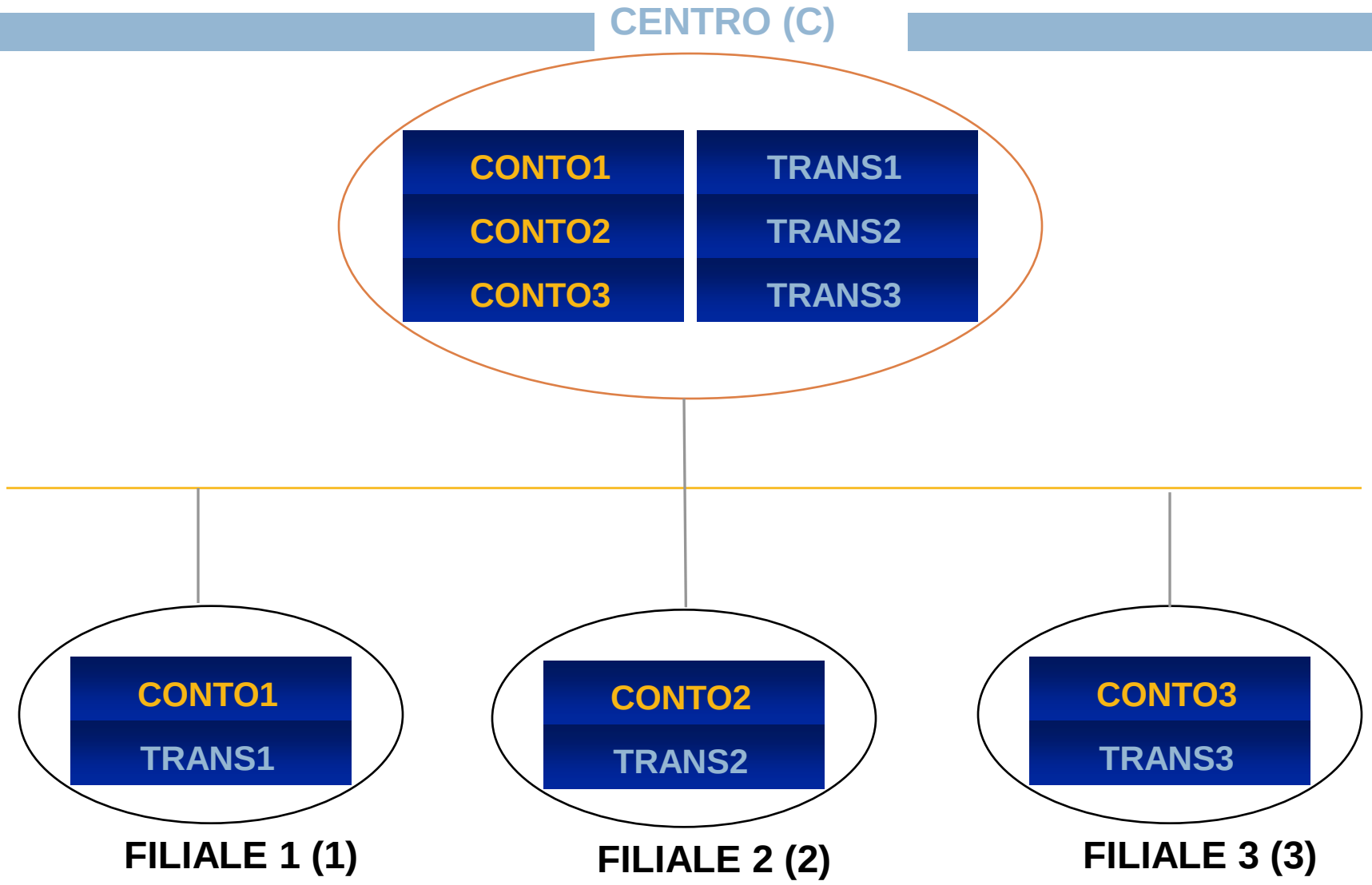
- ***Es:***
  - **TRANS1 = TRANSAZIONE ⋊ CONTO1**
  - **TRANS2 = TRANSAZIONE ⋊ CONTO2**
  - **TRANS3 = TRANSAZIONE ⋊ CONTO3**

Allocazione dei frammenti

- **Rete:**
  - 3 siti periferici
  - 1 sito centrale

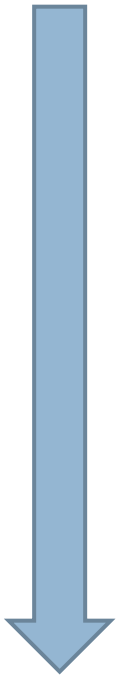
- **Allocazione:**
  - locale
  - centrale

Allocazione dei frammenti (2)



Livelli di trasparenza

- Modalità per esprimere interrogazioni offerte dai DBMS commerciali (LIVELLI):
  - **FRAMMENTAZIONE**
    - Il programmatore non si deve preoccupare se o no il DB è distribuito o frammentato.
  - **ALLOCAZIONE**
    - Il programmatore dovrebbe conoscere la struttura dei frammenti, ma non deve indicare la loro allocazione.
  - **LINGUAGGIO**
    - Il programmatore deve indicare nella query sia la struttura dei frammenti che la loro allocazione.



Trasparenza di frammentazione

- **QUERY:**
  - Estrarre il saldo del conto corrente 45.

SELECT SALDO

FROM CONTO-CORRENTE

WHERE NUM-CC=45

Trasparenza di allocazione

- **IPOTESI:**
  - Conto corrente 45 presso la Filiale 1 (locale).

SELECT SALDO

FROM CONTO1

WHERE NUM-CC=45

Trasparenza di allocazione (2)

- **IPOTESI:**
  - Allocazione incerta: probabilmente allocato presso la Filiale 1.

```
SELECT SALDO FROM CONTO1
                WHERE NUM-CC=45
IF (NOT FOUND) THEN
   ( SELECT SALDO FROM CONTO2
                WHERE NUM-CC=45
UNION
   SELECT SALDO FROM CONTO3
                WHERE NUM-CC=45 )
```

Trasparenza di linguaggio

```
SELECT SALDO FROM CONTO1@1
            WHERE NUM-CC=45
IF (NOT FOUND) THEN
   ( SELECT SALDO FROM CONTO2@C
               WHERE NUM-CC=45
   UNION
   SELECT SALDO FROM CONTO3@C
               WHERE NUM-CC=45 )
```

Trasparenza di frammentazione

- **QUERY:**
  - Estrarre i movimenti dei conti con saldo negativo.

```
SELECT NUM-CC, PROGR, AMMONTARE
FROM CONTO-CORRENTE AS C
        JOIN TRANSAZIONE AS T
        ON C.NUM-CC=T.NUM-CC
  WHERE SALDO < 0
```

Trasparenza di allocazione (join distribuito)

```
SELECT  NUM-CC, PROGR, AMMONTARE
        FROM CONTO1 JOIN TRANS1 ON ……
        WHERE SALDO < 0
UNION
SELECT  NUM-CC, PROGR, AMMONTARE
        FROM CONTO2 JOIN TRANS2 ON ……
        WHERE SALDO < 0
UNION
SELECT  NUM-CC, PROGR, AMMONTARE
        FROM CONTO3 JOIN TRANS3 ON ……
        WHERE SALDO < 0
```

Trasparenza di linguaggio

```
SELECT  NUM-CC, PROGR, AMMONTARE
        FROM CONTO1@1 JOIN TRANS1@1 ON …...
        WHERE SALDO < 0
UNION
SELECT NUM-CC, PROGR, AMMONTARE
        FROM CONTO2@C JOIN TRANS2@C ON ……
        WHERE SALDO < 0
UNION
SELECT  NUM-CC, PROGR, AMMONTARE
        FROM CONTO3@C JOIN TRANS3@C ON ……
        WHERE SALDO < 0
```

Trasparenza di frammentazione

- **UPDATE:**
  - Sposta il cliente 45 dalla Filiale 1 alla Filiale 2.

```
UPDATE CONTO-CORRENTE
   SET FILIALE = 2
   WHERE NUM-CC=45 AND FILIALE=1
```

Trasparenza di allocazione (e replicazione)

```
INSERT INTO CONTO2
  SELECT * FROM CONTO1 WHERE NUM-CC=45

INSERT INTO TRANS2
  SELECT * FROM TRANS1 WHERE NUM-CC=45

DELETE FROM CONTO1 WHERE NUM-CC=45
DELETE FROM TRANS1 WHERE NUM-CC=45
```

Bisogna settare anche la FILIALE a 2

Trasparenza di linguaggio

```
INSERT INTO CONTO2@2
  SELECT * FROM CONTO1@1 WHERE NUM-CC=45
INSERT INTO CONTO2@C
  SELECT * FROM CONTO1@C WHERE NUM-CC=45
INSERT INTO TRANS2@2
  SELECT * FROM TRANS1@1 WHERE NUM-CC=45
INSERT INTO TRANS2@C
  SELECT * FROM TRANS1@C WHERE NUM-CC=45
```

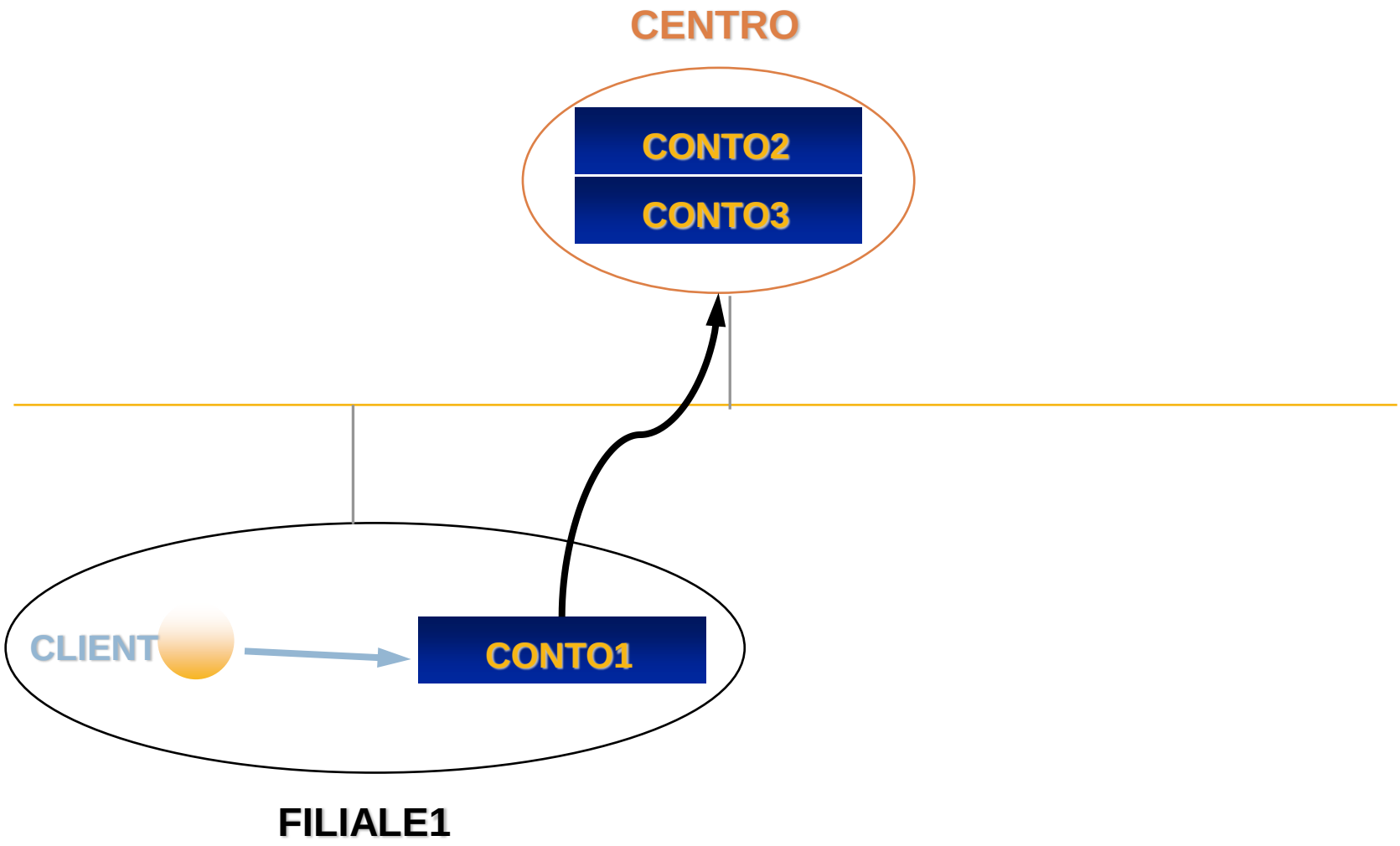
- (in modo analogo: 2 coppie di **DELETE**)

Bisogna settare anche la FILIALE a 2

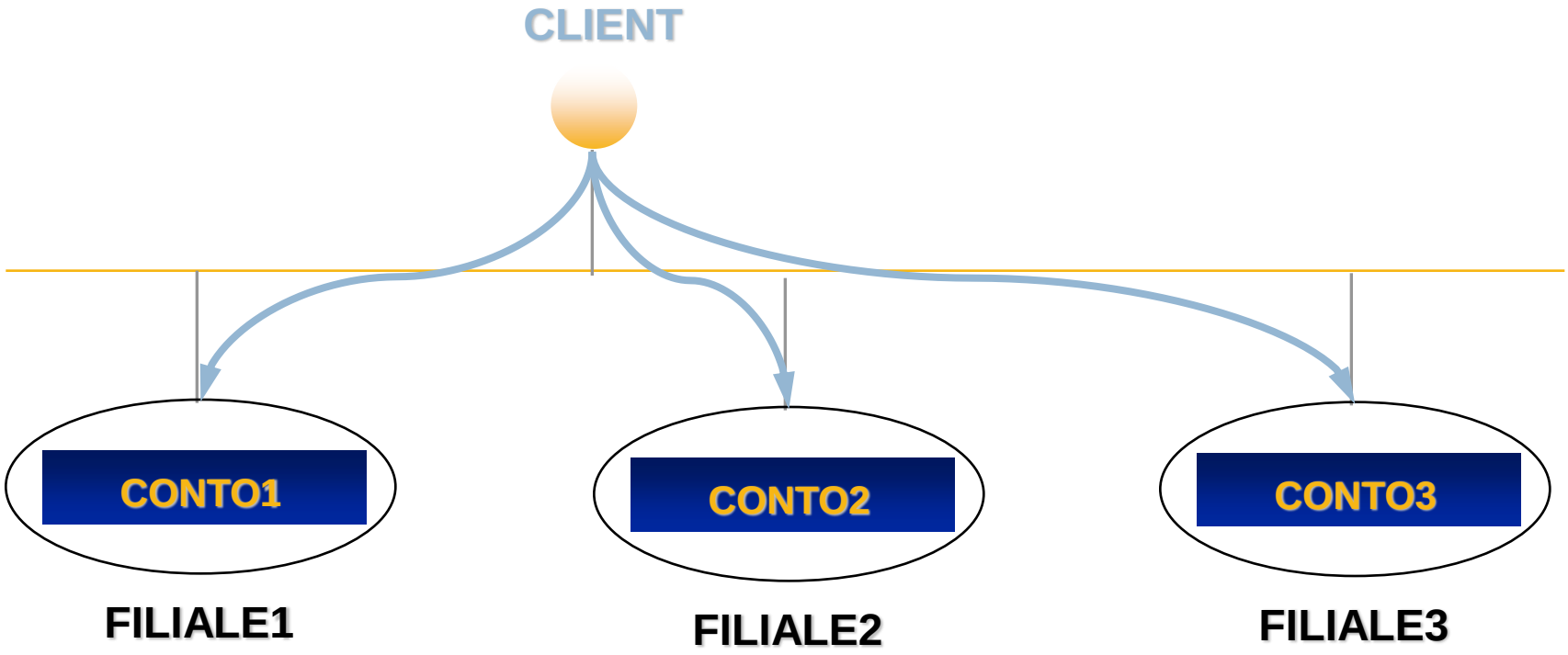
Efficienza

- Ottimizzazione delle query.
- Modalità di esecuzione:
  - seriale
  - parallela

Esecuzione seriale



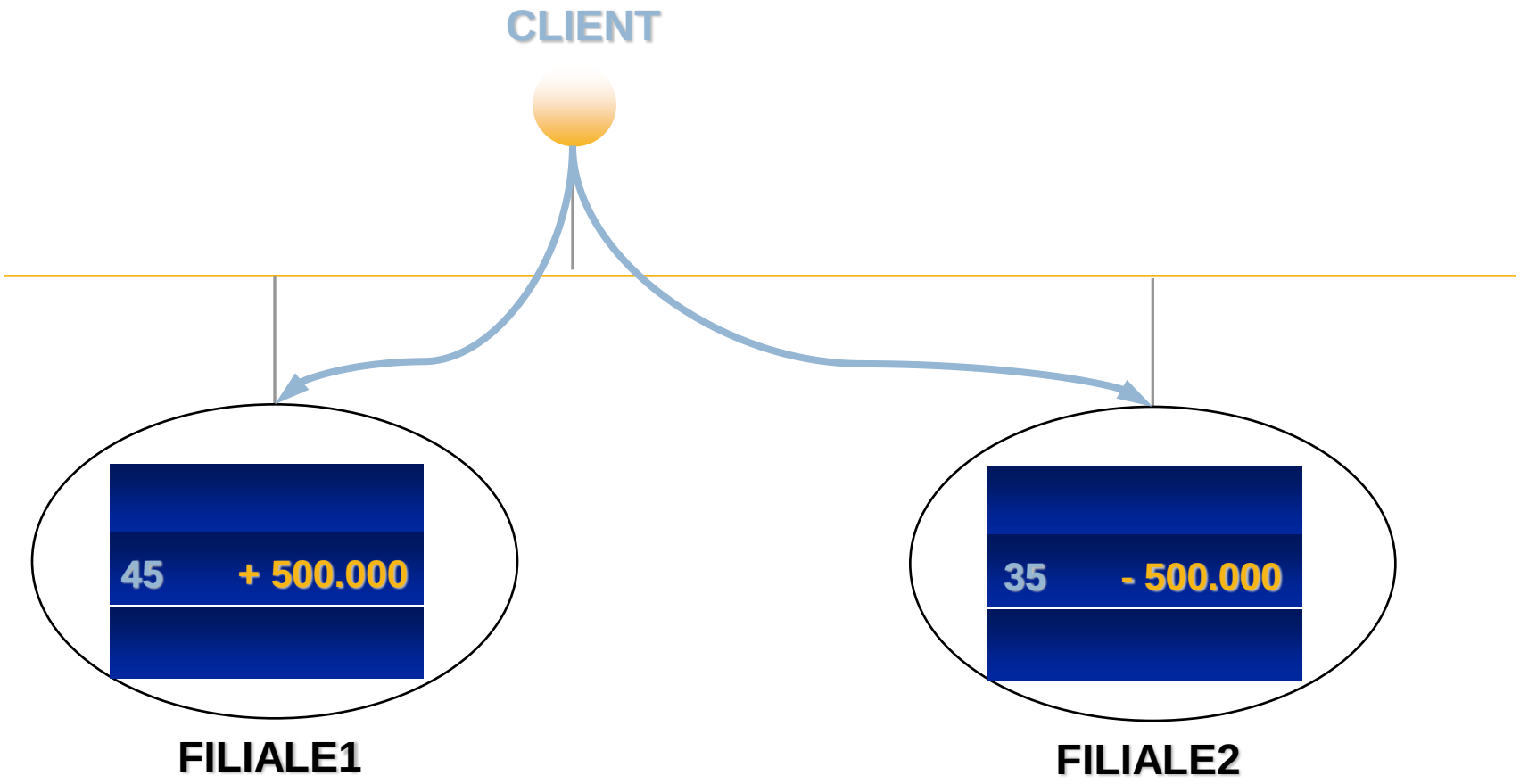
Esecuzione parallela



Transazioni distribuite

```
BEGIN TRANSACTION
    UPDATE CONTO1@1
    SET SALDO = SALDO + 500.000
    WHERE NUM-CC=45;
    UPDATE CONTO2@2
    SET SALDO = SALDO - 500.000
    WHERE NUM-CC=35;
COMMIT-WORK
END TRANSACTION
```

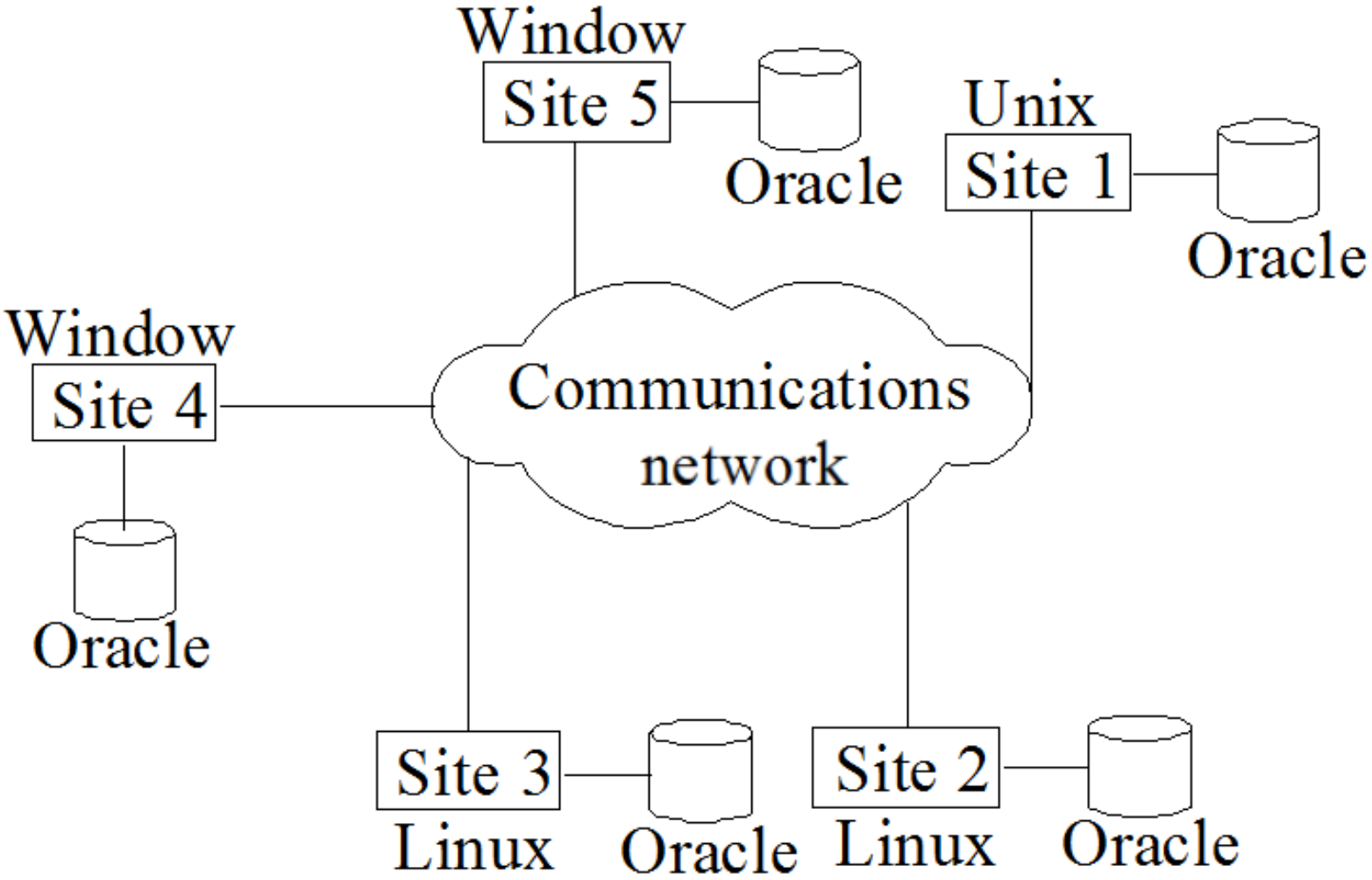
Transazioni distribuite (2)



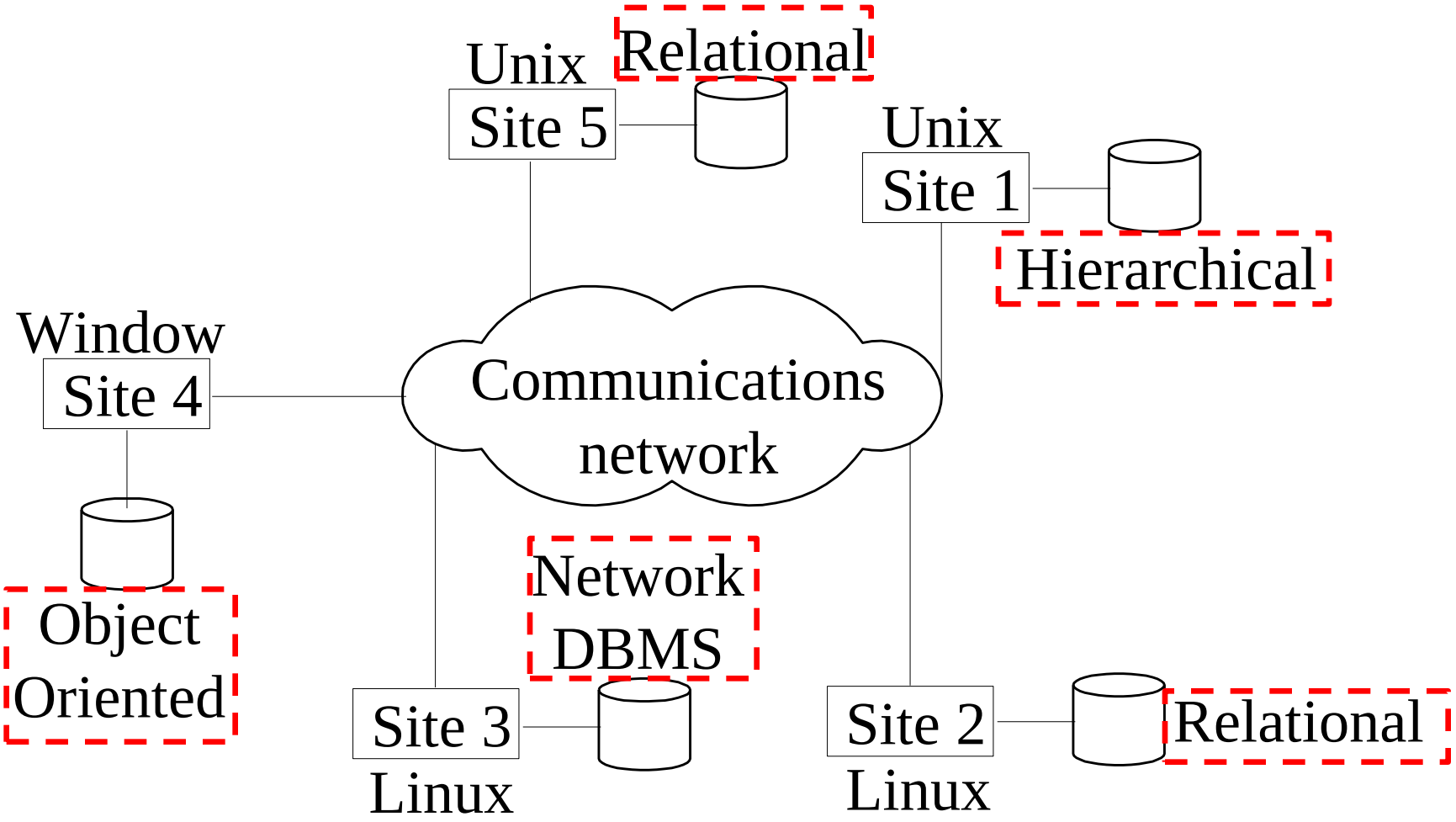
Tipi di DDB

- Si differenziano per diversi fattori:
  - Grado di omogeneità del software dei DDBMS
    - **DDB omogenei**: tutti i server e tutti gli utenti usano lo stesso software.
    - **DDB eterogenei**, altrimenti.
  - Grado di autonomia locale:
    - Se il sito locale non può funzionare come un DBMS stand-alone non ha nessuna autonomia locale, all'utente il sistema "*appare*" come un DBMS centralizzato.
    - Se le transazioni locali è permesso l'accesso diretto a un server possono accedervi, ha qualche grado di autonomia locale.

DDB omogenei



DDB eterogenei



Tipi di DDB (2)

- Sistemi di Database **federati**: ogni server è un DBMS centralizzato e autonomo con:
  - I suoi propri utenti locali;
  - Transazioni locali;
  - DBA;
  - e, quindi, un alto grado di autonomia locale.

- È presente una vista globale o schema della federazione di database.

Tipi di DDB (3)

- Un sistema multidatabase (DB federati o FDB) non ha uno schema globale e interattivamente ne viene costruito uno (vista globale) in base alle necessità dell'applicazione.

- In un FDB eterogeneo, i DB coinvolti possono essere relazionali, reticolari, gerarchici, …
  - È necessario introdurre dei **traduttori di query**.

Sistemi per la gestione dei DB Federati

- Tipi differenti di eterogeneità:
  - Differenze nei modelli di dati.
  - Differenze nei vincoli:
    - le modalità per la specifica dei vincoli varia da sistema a sistema.
  - Differenze nei linguaggi di query:
    - anche con lo stesso modello di dati i linguaggi e le loro versioni possono variare.

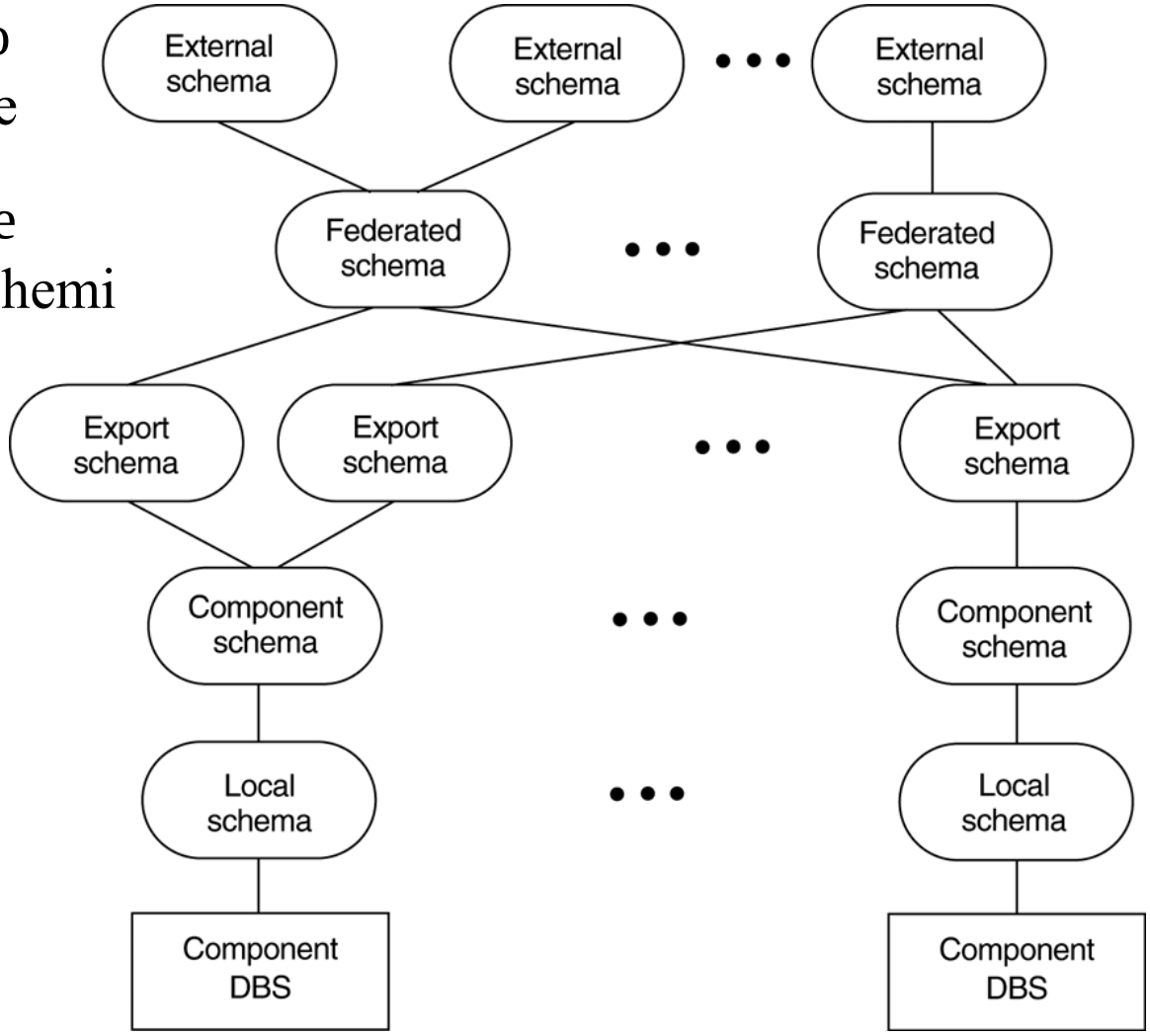
Sistemi per la gestione dei DB Federati (2)

- Eterogeneità semantica:
  - Differenze di significato, interpretazione, uso degli stessi dati o di dati correlati.

- *Costituisce la principale difficoltà nella progettazione degli schemi globali di basi di dati eterogenee.*

Lo schema a cinque livelli (FDBS)

- Lo schema per un gruppo di utenti o un'applicazione
- Schema globale risultante dall'integrazione di più schemi
- Sottoinsieme dello schema componente
- Modello dati comune
- Schema concettuale



Controllo della concorrenza e Recovery

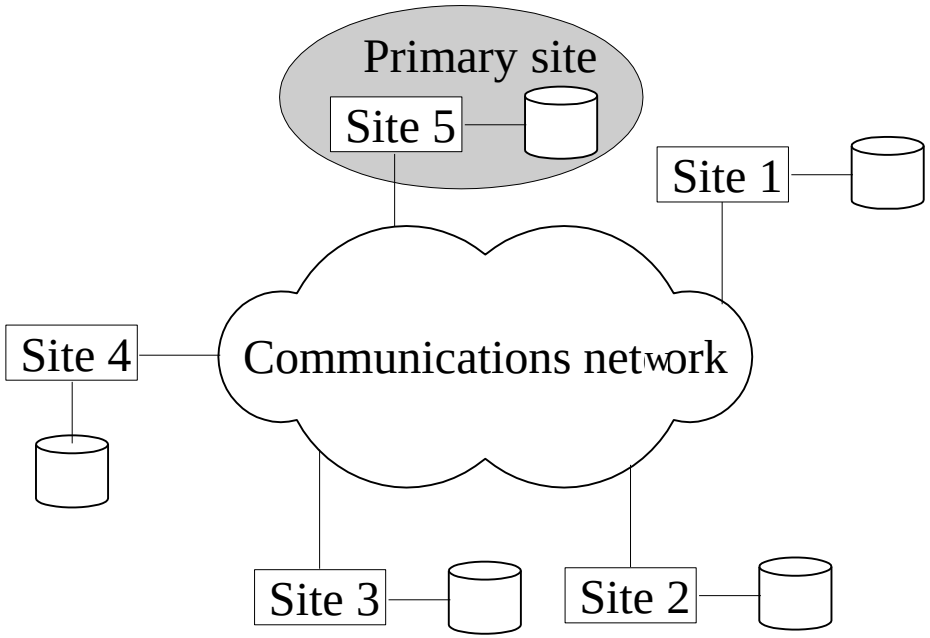
- I DB distribuiti incontrano un numero di controlli di concorrenza e problemi di recovery che non sono presenti nei DB centralizzati.
- Alcuni sono elencati di seguito:
  - Trattamento di **copie multiple** di dati:
    - Il controllo della consistenza deve mantenere una **consistenza globale**. Allo stesso modo il meccanismo di recovery deve recuperare tutte le copie e conservare la consistenza dopo il recovery.
  - Fallimenti di singoli siti:
    - La disponibilità del DB non deve essere influenzata dai guasti di uno o due siti e lo schema di recovery li deve recuperare prima che siano resi disponibili.

Controllo della concorrenza e Recovery (2)

- Guasto dei collegamenti di comunicazione:
  - Tale guasto può portare ad una partizione della rete che può influenzare la disponibilità del DB anche se tutti i siti sono in esecuzione.
- Commit distribuito:
  - Una transazione può essere frammentata ed essere eseguita su un numero di siti. Questo richiede un approccio basato sul ***commit a due fasi*** per il commit della transazione.
- Deadlock distribuito:
  - Poiché le transazioni sono processate su siti multipli, due o più siti possono essere coinvolti in un ***deadlock***.
  - Di conseguenza devono essere considerate le tecniche per il trattamento dei deadlock.

Controllo della concorrenza e Recovery (3)

- Controllo della concorrenza distribuito basato su una copia designata dei dati:
  - Si designa una particolare copia di ogni dato (**copia designata**).
  - Tutte le richieste di ***lock*** ed ***unlock*** vengono inviate solo al sito che la contiene.
- Tecnica del sito primario:
  - Un singolo sito è designato come **sito primario** il quale fa da coordinatore per la gestione delle transazioni.



Tecnica del sito primario

- Gestione delle transazioni:
  - Il controllo della concorrenza ed il commit sono gestiti da questo sito. La tecnica del lock a due fasi (2PL) è usata per bloccare e rilasciare i data item.
  - Se tutte le transazioni in tutti i siti seguono la politica delle due fasi allora viene garantita la serializzabilità.
- Vantaggi:
  - I data item sono bloccati (locked) solamente in un sito ma possono essere usati da qualsiasi altro sito.
- Svantaggi:
  - Tutta la gestione delle transazioni passa per il sito primario che potrebbe essere **sovraccaricato**. Nel caso in cui il sito primario fallisce, l'intero sistema è inaccessibile.
- Per assistere il recovery, un sito di backup viene designato come copia di backup dei sito primario.
  - Nel caso di fallimento del sito primario, il sito di backup funziona da sito principale.

Tecnica della copia primaria

- Tecnica della copia primaria:
  - In questo approccio, invece di un sito, una partizione dei data item è designata come copia primaria.
  - Per bloccare un data item soltanto sulla copia primaria di quel data item viene eseguito il lock.
- Vantaggi:
  - Le copie primarie sono distribuite su vari siti, e un singolo sito **non viene sovraccaricato** da un numero elevato di richieste di lock ed unlock.
- Svantaggi:
  - L'identificazione della copia primaria è complesso. Una directory distribuita deve essere gestita possibilmente in tutti i siti.

Recovery dal fallimento di un coordinatore

- In entrambi gli approcci un **sito coordinatore** o un **sito copia** possono essere indisponibili. Questo richiede la selezione di un nuovo coordinatore.
  - Approccio del sito primario senza sito di backup:
    - Abortire e far ripartire tutte le transazioni attive in tutti i siti. Si elegge un nuovo coordinatore che inizia il processing delle transazioni.
  - Sito primario con copia di backup:
    - Sospende tutte le transazioni attive, designa il sito di backup come sito primario ed identifica un nuovo sito di backup. Il nuovo sito primario riceve il compito di gestire tutte le transazioni per riprendere il processo.
- Il sito primario e quello di backup falliscono:
  - Si usa un processo di elezione per selezionare un nuovo sito coordinatore.

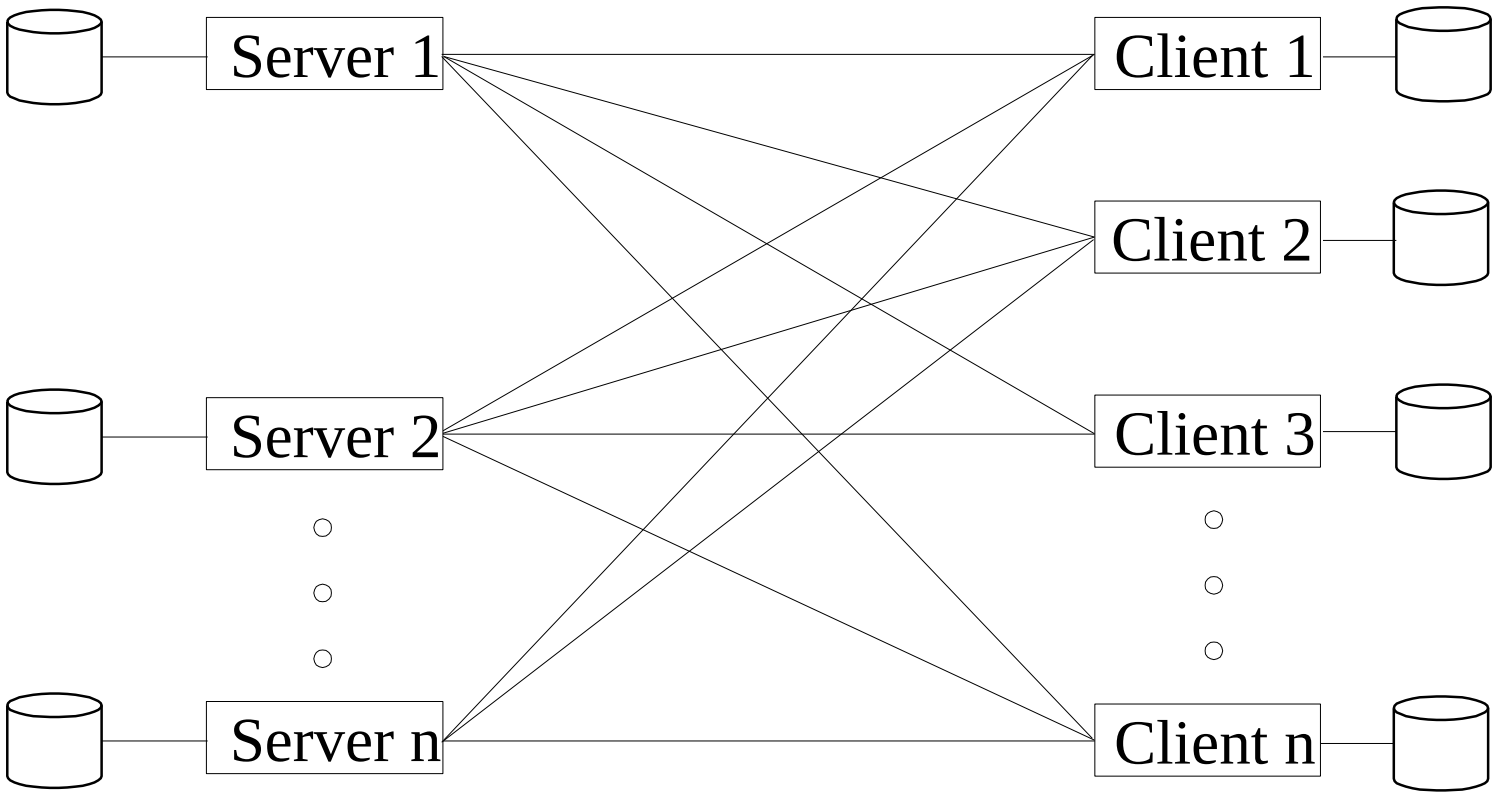
Controllo della concorrenza basata sul voting

- Controllo della concorrenza basata sul **voting**: *Non esiste la copia primaria del coordinatore.*
  - Spedire una richiesta di lock ai siti che hanno i data item.
  - Se la maggioranza dei siti concedono il lock allora la transazione richiedente ottiene il data item.
  - Le informazioni di lock (concesse o negate) sono spedite a tutti questi siti.
  - Per evitare un tempo di attesa inaccettabile, viene definito un periodo di **timeout**. Se la transazione richiedente non riceve alcuna informazione di voto allora la transazione viene abortita.

Architettura Client-Server

- Discutiamo l'architettura Client-Server in generale e poi l'applichiamo ai DBMS.
  - In un ambiente con un grande numero di PC, stampanti, etc …
  - Si definiscono dei **server specializzati** con funzionalità specifiche.
  - **Es:**
    - il file server mantiene i file delle macchine client.
    - Il printer server gestisce la stampa su diverse stampanti,
    - l'e-mail server gestisce la posta elettronica.

Architettura Client-Server (2)

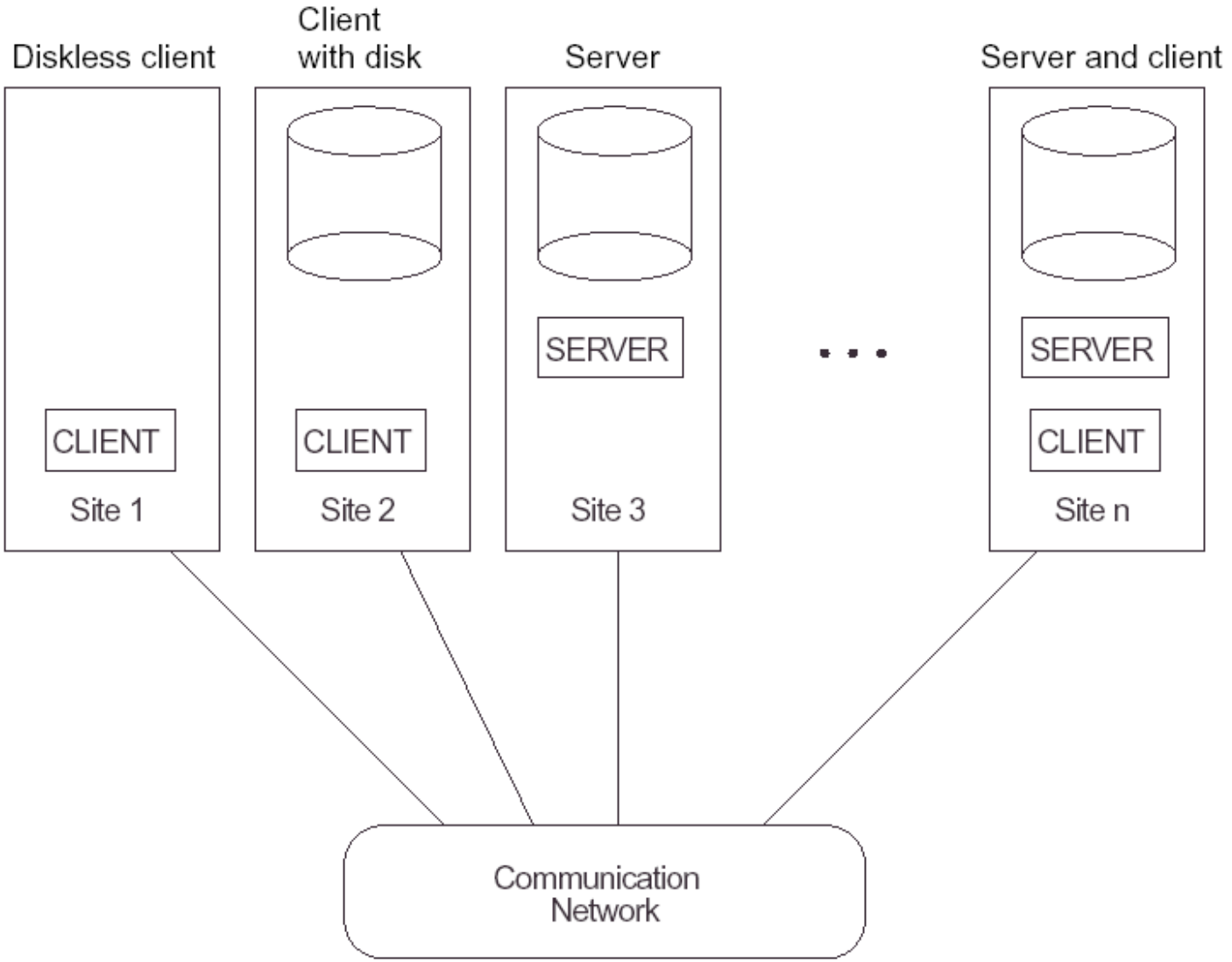


Architettura Client-Server (2)

- Le risorse fornite da server specializzati possono essere messe a disposizione di diversi **client**.

- Una macchina client fornisce all'utente l'appropriata interfaccia per utilizzare questi server impiegando la potenza di calcolo locale per eseguire applicazioni locali.

Architettura Client-Server (3)



Architettura Clent-Server per DBMS

- Differenti approcci sono stati proposti su come suddividere le funzionalità tra client e server.
- Una possibilità è di includere le funzionalità in un DBMS centralizzato a livello server.

- Un **SQL server** è fornito al client (ogni client):
  - Formula la query.
  - Fornisce l'interfaccia utente.
  - Fornisce funzioni di interfaccia dei linguaggi di programmazione.

Architettura Clent-Server per DBMS (2)

- SQL è uno standard:
  - I server SQL di diversi produttori possono accettare query SQL.

- I client possono collegarsi al dizionario dei dati che include la distribuzione dei dati fra i vari server SQL.

Interazione fra Client e Server

- Elaborazione di una query:
  1. Il client parsa una query e la decompone in un certo numero di query ai siti indipendenti.
  2. Ogni server processa la query locale e manda il risultato al sito client.
  3. Il client combina il risultato delle sottoquery per produrre il risultato della query sottomessa in origine.

Interazione fra Client e Server (2)

- In questo approccio, il server SQL è detto anche **transaction server** o *back-end machine*.
- Il client è detto **application processor** o *front-end machine*.

- In un tipico DDBMS, si è soliti dividere i moduli software su tre livelli:
  - **Software server**, responsabile per la gestione dei dati locali di un sito.
  - **Software client**, gestisce l'interfaccia utente, accede al catalogo del DB e processa tutte le richieste che richiedono l'accesso a più di un sito.
  - **Software di comunicazione**, fornisce le primitive di comunicazione che sono usate dal client per trasmettere comandi e dati tra i vari siti (server).