# Basi di dati distribuite

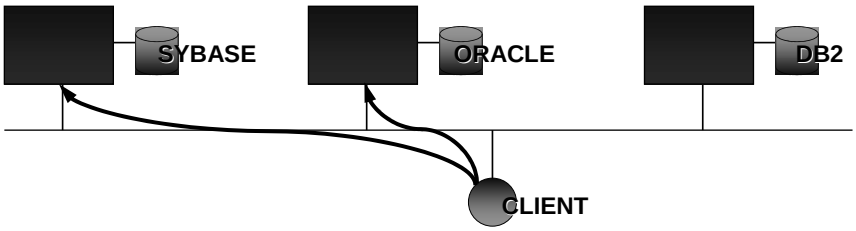
## Motivazioni della distribuzione dei dati

- **Natura intrinsecamente distribuita delle organizzazioni**
- **Evoluzione degli elaboratori**
  - **aumento della capacita' elaborativa**
  - **riduzione di prezzo**
- **Evoluzione della tecnologia dei DBMS**
- **Standard di interoperabilita'**

# 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
- Affidabilita'

## Autonomia e cooperazione

**L'esigenza di autonomia:**
- Una reazione ai ''Centri EDP''
- Portare competenze e controllo laddove vengono gestiti i dati
- Rendere la maggior parte delle applicazioni NON distribuite (!)

**L'esigenza di cooperazione:**
- Alcune applicazioni sono intrinsecamente distribuite e richiedono l'accesso a piu' basi di dati

## Frammentazione dei dati

**Scomposizione delle tabelle in modo da consentire la loro distribuzione**

**proprieta':**

- **Completezza**
- **Ricostruibilita'**

## Frammentazione orizzontale

**FRAMMENTI:**

**insiemi di tuple**

**COMPLETEZZA:**

**presenza**

**di tutte le tuple**

**RICOSTRUZIONE:**

**unione**

# Frammentazione verticale

**FRAMMENTI:**

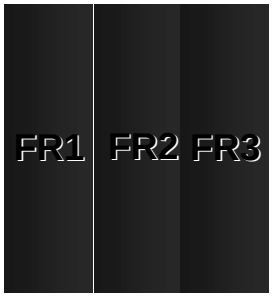
**insiemi di attributi**

**COMPLETEZZA:**

**presenza**
**di tutti gli attributi**

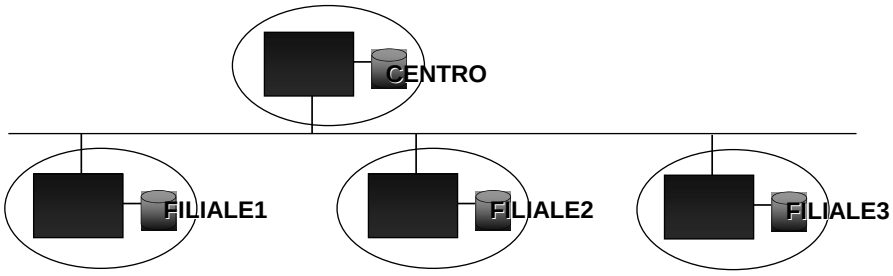
**RICOSTRUZIONE:**

**join sulla chiave**




# Esempio: conti correnti bancari

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



## Frammentazione orizzontale principale

$Ri = \sigma_{Pi} R$

**esempio:**

$CONTO1 = \sigma_{Filiale=1}$ CONTO-CORRENTE

$CONTO2 = \sigma_{Filiale=2}$ CONTO-CORRENTE

$CONTO3 = \sigma_{Filiale=3}$ CONTO-CORRENTE

## Frammentazione orizzontale derivata

$Si = S \ltimes Ri$

**esempio:**

TRANS1 = TRANSAZIONE $\ltimes$ CONTO1

TRANS2 = TRANSAZIONE $\ltimes$ CONTO2

TRANS3 = TRANSAZIONE $\ltimes$ CONTO3

# Allocazione dei frammenti

**Rete :**

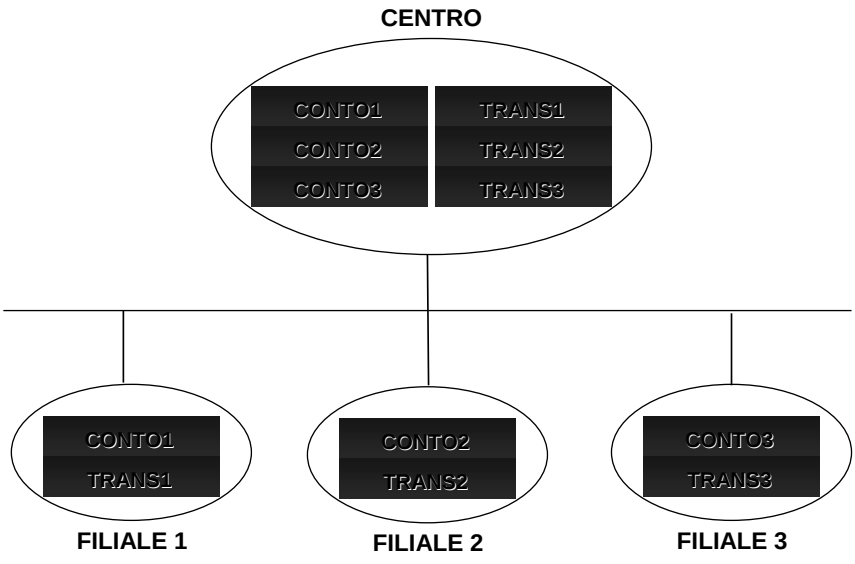
**3 siti periferici, 1 sito centrale**

**Allocazione:**

**locale**

**centrale**

# Allocazione dei frammenti



## Livelli di trasparenza

**Modalita' per esprimere interrogazioni offerte dai DBMS commerciali:**

**LIVELLI: FRAMMENTAZIONE**
**ALLOCAZIONE**
**LINGUAGGIO**

## 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

**IPOTESI :**

**Allocazione incerta, probabilmente alla filiale 1**

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

## Trasparenza di linguaggio

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

## Trasparenza di frammentazione

**QUERY :**
**estrarre i movimenti dei conti con saldo negativo**

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

# Trasparenza di linguaggio

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

## Trasparenza di frammentazione

**UPDATE :**
**sposta il conto 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
```

## Trasparenza di linguaggio

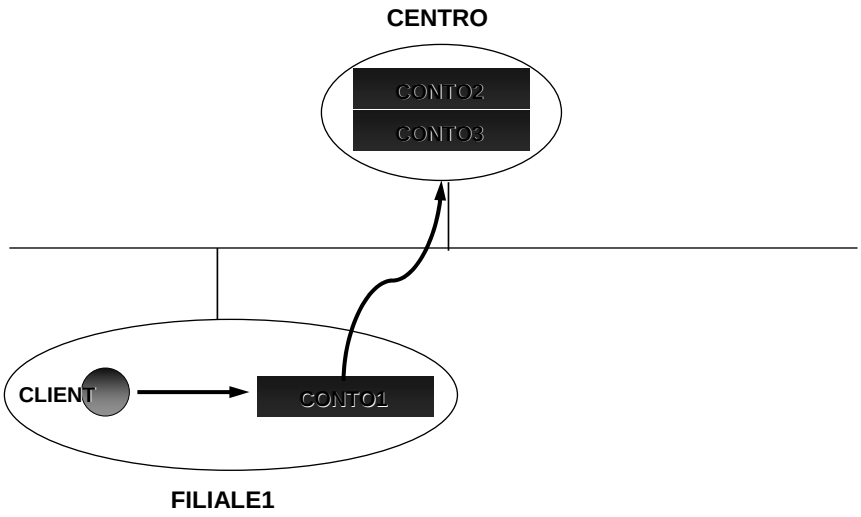
```
INSERT INTO CONTO2@2
  SELECT * FROM CONTO1 WHERE NUM-CC=45
INSERT INTO CONTO2@C
   SELECT * FROM CONTO1 WHERE NUM-CC=45

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

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

## Efficienza

- **Ottimizzazione delle query**
- **Modalita' di esecuzione**
  - **seriale**
  - **parallela**

# Esecuzione seriale




# Esecuzione parallela