



Architetture Dati

Proprietà e caratteristiche strutturali dei sistemi DDBMS



1. Proprietà generali di un DDBMS

Vantaggi dei DDBMS: discuteremo

- Localita', i dati si trovano "vicino" alle applicazioni che li utilizzano piu' frequentemente
- Modularita', le modifiche alle applicazioni e ai dati possono essere effettuate a basso costo
- Resistenza ai guasti
- Prestazioni/Efficienza

Localita'

- La partizione dei dati corrisponde spesso ad una partizione naturale delle applicazioni e degli utenti
- Utenti locali vs utenti globali
 - Es. Un' organizzazione con sedi diverse distribuite geograficamente
 - I dati risiedono vicino a dove vengono usati piu' spesso
 - Ma sono globalmente raggiungibili

Flessibilità'

- + Distribuzione dei dati incrementale e progressiva: la configurazione si adatta alle esigenze delle applicazioni

Resistenza ai guasti

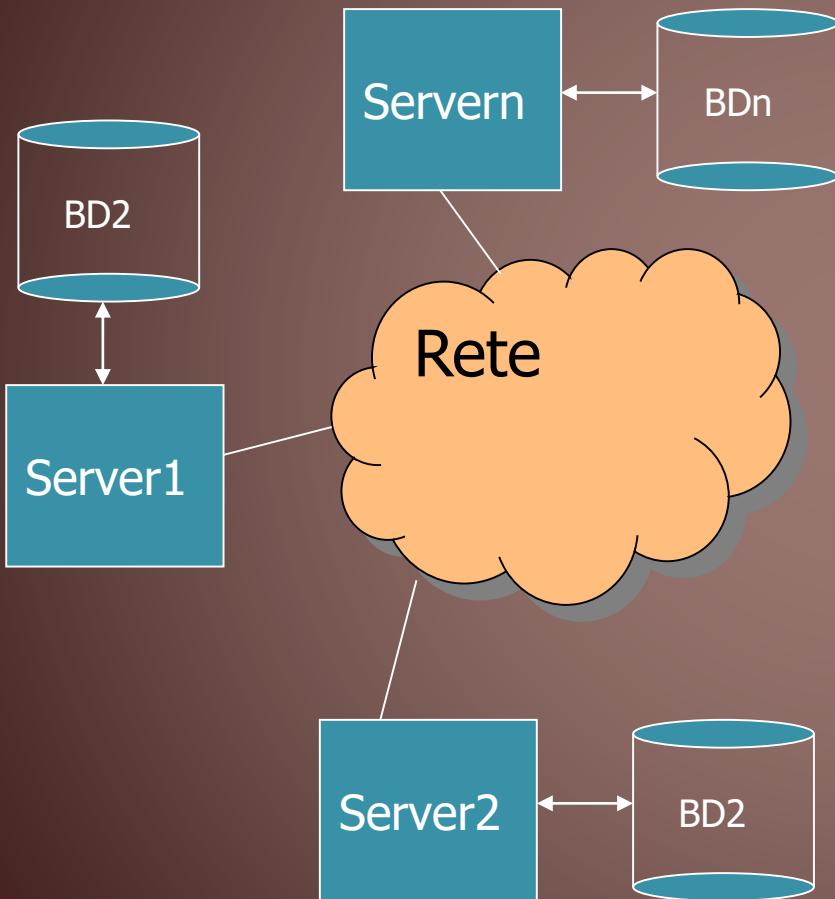
- Presenza di elementi di rete: maggiore fragilità
- + Ma: presenza di ridondanza → maggiore resistenza ai guasti di dati e applicazioni ridondate ("fail soft")

Prestazioni

- Distribuendo un DB su nodi diversi, ogni nodo gestisce un DB di dimensioni piu' ridotte
- + Piu' semplice da gestire e ottimizzare nelle applicazioni locali
- + Ogni nodo puo' essere ottimizzato indipendentemente dagli altri
- + Carico totale (transazioni /sec) distribuito sui nodi
- + Parallelismo tra transazioni locali che fanno parte di una stessa transazione distribuita
- - Necessita' di coordinamento tra i nodi
- - Presenza di traffico di rete

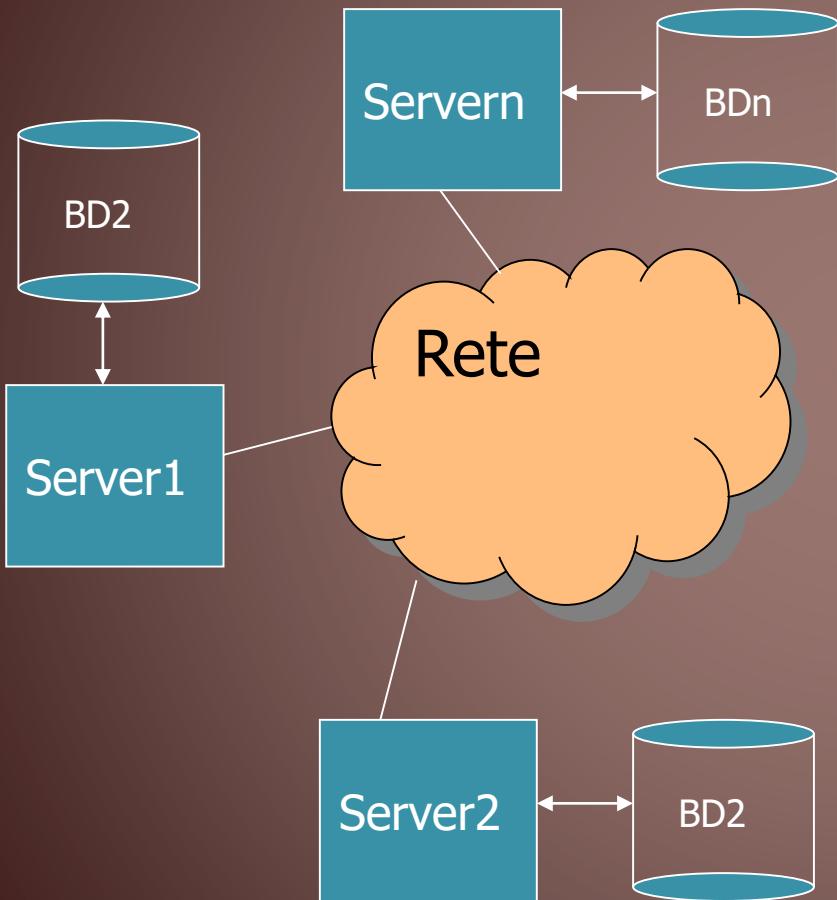
Funzionalita' specifiche dei DDBMS
rispetto ai DBMS centralizzati

Relazione tra indipendenza locale / cooperazione tra server - 1



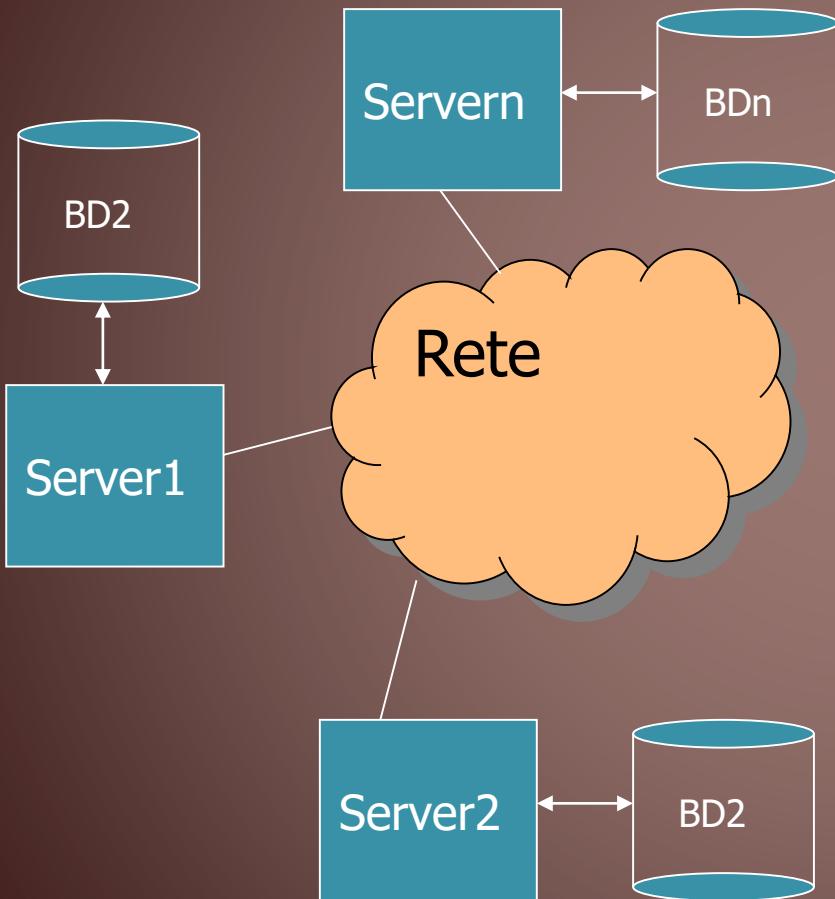
- Ogni server mantiene la capacita' di gestire applicazioni in modo indipendente
- Le interazioni con altri server e applicazioni remote rappresentano un carico supplementare sul sistema

Relazione tra indipendenza locale / cooperazione tra server - 2



- Traffico di rete in questa configurazione:
 - Per le interrogazioni
 - Queries provenienti dalle applicazioni
 - Risultati provenienti dal server
 - Per le transazioni
 - Richieste transazionali dalle applicazioni
 - Dati di controllo per il coordinamento

Relazione tra indipendenza locale / cooperazione tra server - 3



- Ottimizzazione: l'elemento critico e' la rete
- Esigenza di distribuire i dati in modo che:
 - La maggior parte delle transazioni sia locale, o eviti trasmissione di dati tra nodi

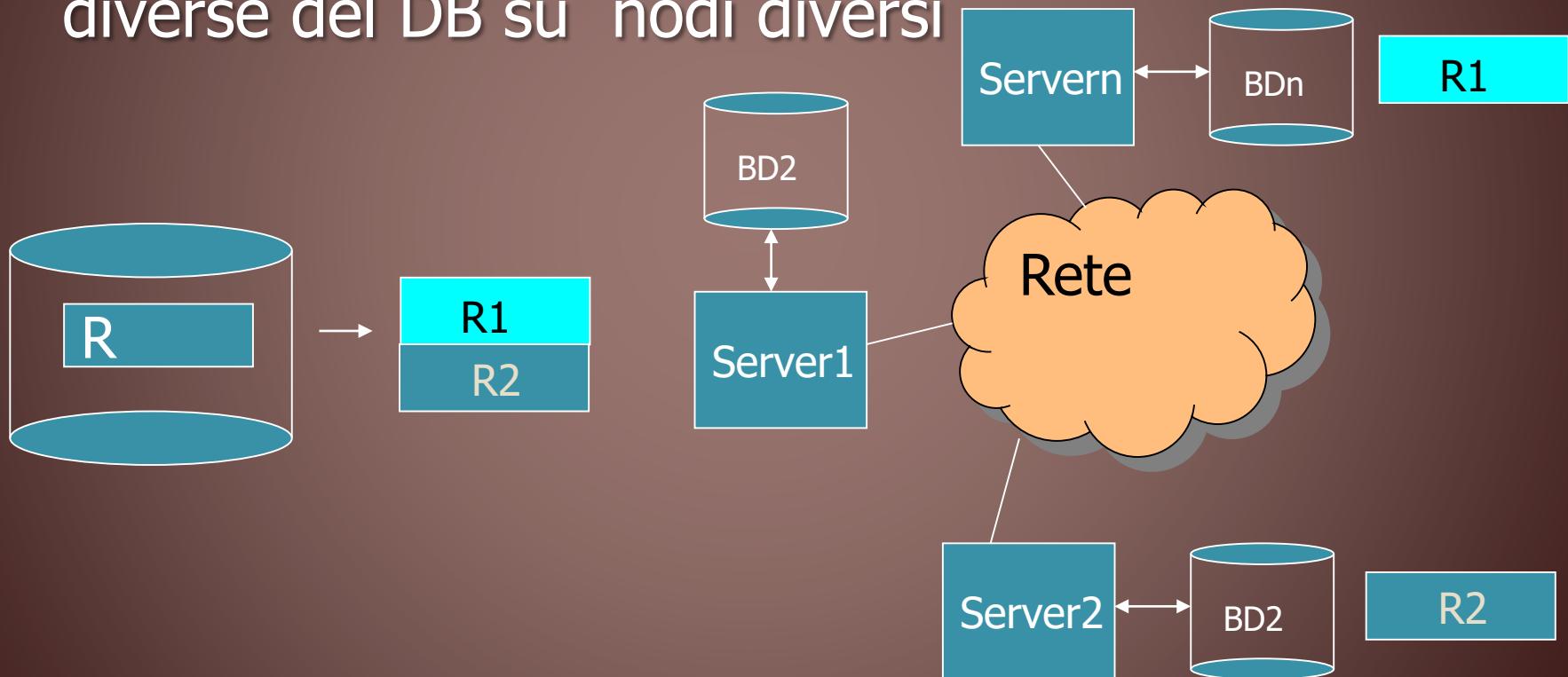
Funzionalita' specifiche dei DDBMS (rispetto ai DBMS)

- **Trasmissione** sia di queries/transazioni che di *frammenti di DB/dati di controllo* tra i nodi
- **Frammentazione/replicazione/trasparenza**: problemi legati al fatto che i dati sono allocati in modo distribuito
- **Query processor**: il query plan deve prevedere una strategia globale accanto a strategie per le query locali
- **Controllo di concorrenza**: algoritmi distribuiti
- **Strategie di recovery di singoli nodi, gestione di guasti globali**

Caratteristiche strutturali dei DDBMS frammentazione, replicazione, trasparenza

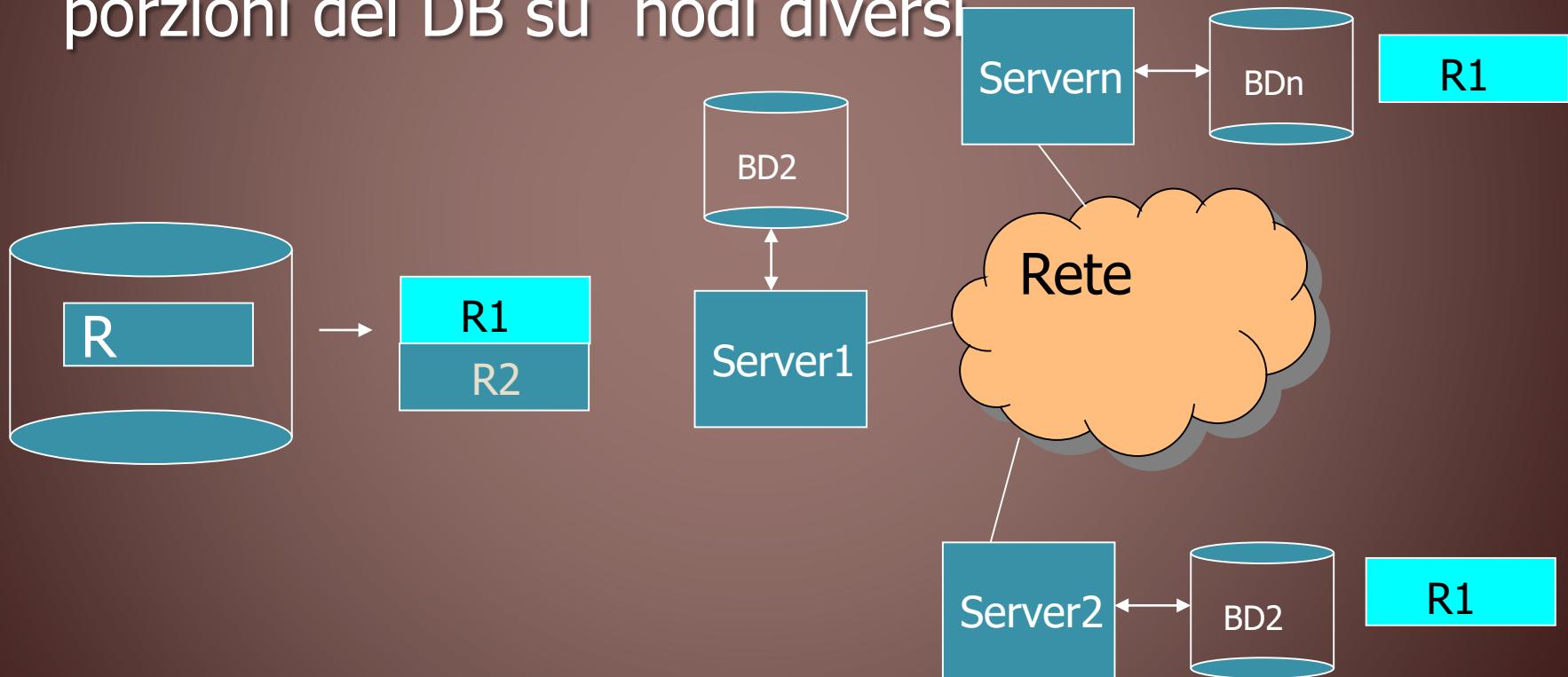
Caratteristiche strutturali dei sistemi DDBMS

- **Frammentazione:** Possibilita' di allocare porzioni diverse del DB su nodi diversi



Caratteristiche strutturali dei sistemi DDBMS

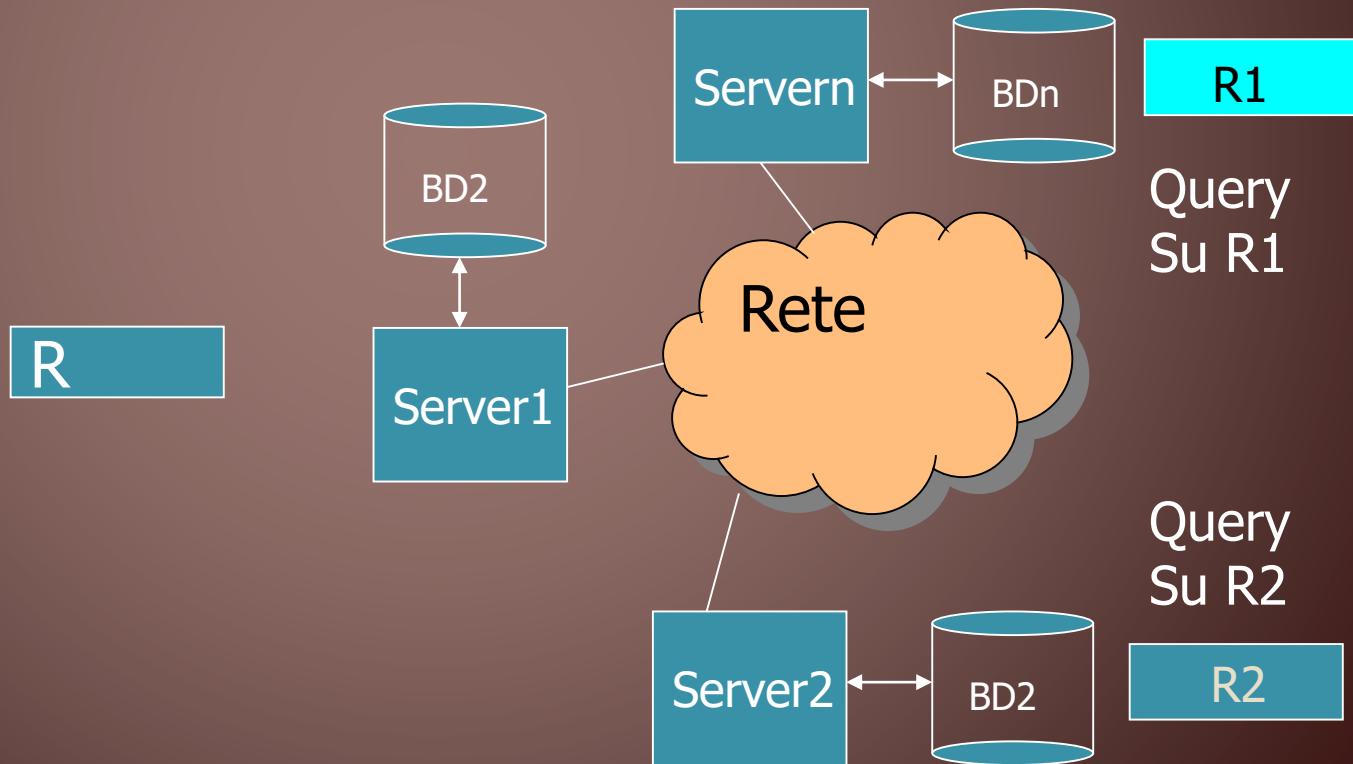
- **Replicazione:** Possibilita' di allocare stesse porzioni del DB su nodi diversi



Caratteristiche strutturali dei sistemi DDBMS

- **Trasparenza:** Possibilità per la applicazione di accedere ai dati senza sapere dove sono allocati

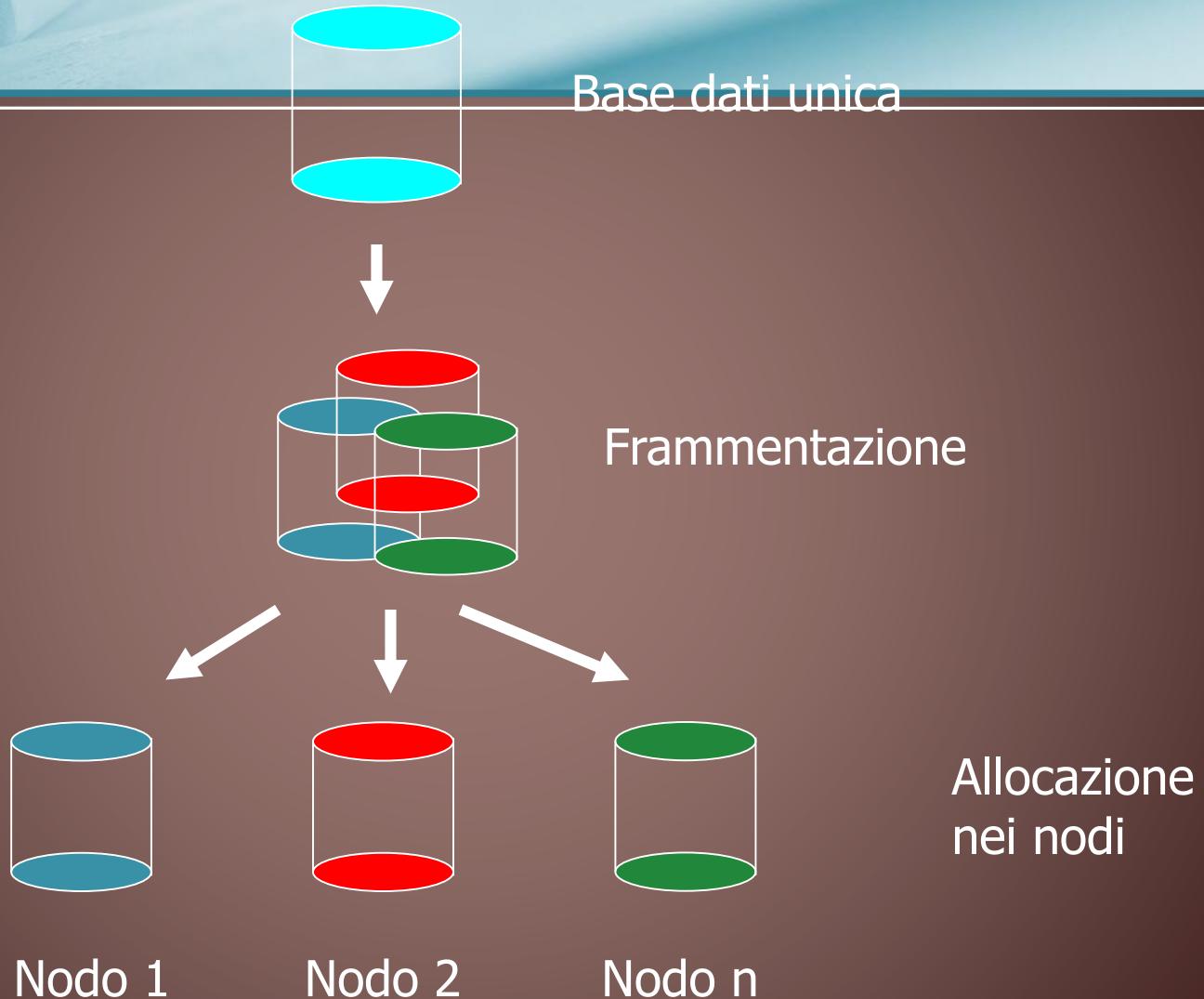
Query
Espressa
Su R





Frammentazione e replicazione dei dati

Il processo di frammentazione

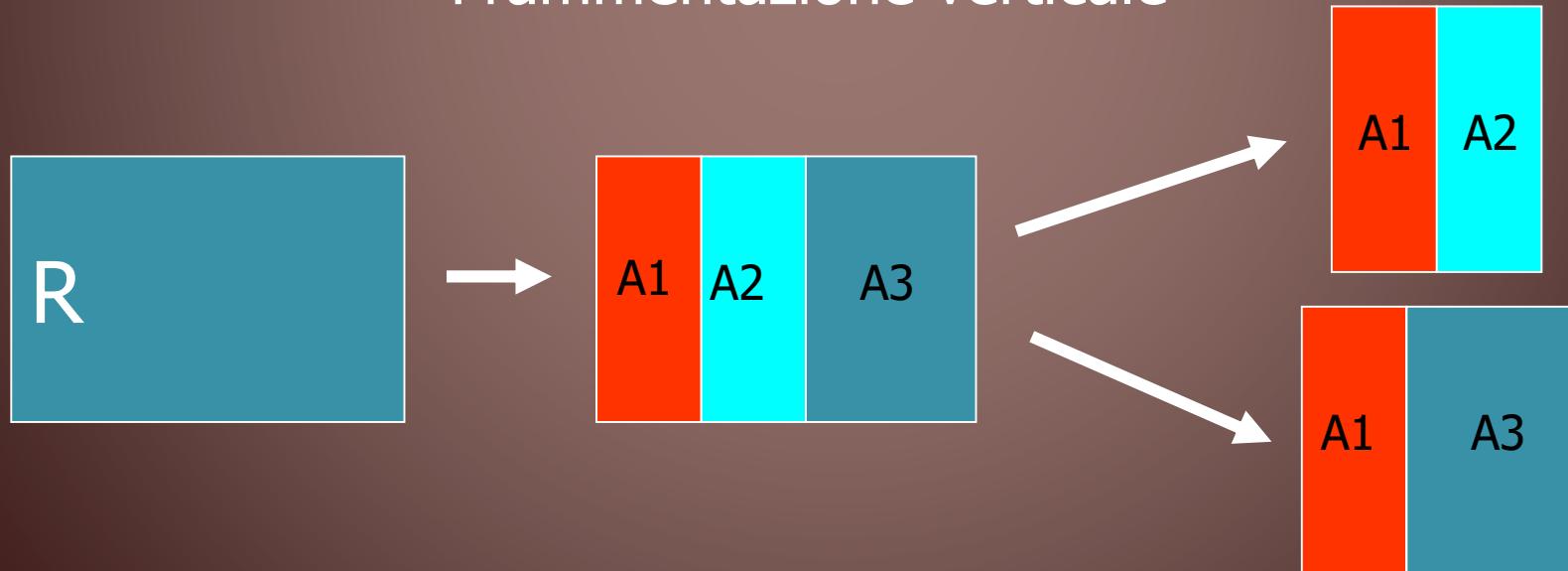


A ben vedere, due tipi di frammentazione

Frammentazione orizzontale



Frammentazione verticale



Esempio: Relazione non frammentata

EMPLOYEE	EmpNum	Name	DeptName	Salary	Tax
	1	Robert	Production	3.7	1.2
	2	Greg	Administration	3.5	1.1
	3	Anne	Production	5.3	2.1
	4	Charles	Marketing	3.5	1.1
	5	Alfred	Administration	3.7	1.2
	6	Paolo	Planning	8.3	3.5
	7	George	Marketing	4.2	1.4

Relazione frammentata orizzontalmente

EMPLOYEE1	EmpNum	Name	DeptName	Salary	Tax
	1	Robert	Production	3.7	1.2
	2	Greg	Administration	3.5	1.1
	3	Anne	Production	5.3	2.1

EMPLOYEE2	EmpNum	Name	DeptName	Salary	Tax
	4	Charles	Marketing	3.5	1.1
	5	Alfred	Administration	3.7	1.2
	6	Paolo	Planning	8.3	3.5
	7	George	Marketing	4.2	1.4

Relazione frammentata verticalmente

EMPLOYEE1	EmpNum	Name
	1	Robert
	2	Greg
	3	Anne
	4	Charles
	5	Alfred
	6	Paolo
	7	George

EMPLOYEE2	EmpNum	DeptName	Salary	Tax
	1	Production	3.7	1.2
	2	Administration	3.5	1.1
	3	Production	5.3	2.1
	4	Marketing	3.5	1.1
	5	Administration	3.7	1.2
	6	Planning	8.3	3.5
	7	Marketing	4.2	1.4

Quali regole di correttezza devono rispettare le frammentazioni?

- **Completezza** – Ogni record della relazione R di partenza deve poter essere ritrovato in almeno uno dei frammenti
- **Ricostruibilità** – La relazione R di partenza deve poter essere ricostruita senza perdita di informazione a partire dai frammenti
- **Disgiunzione** – Ogni record della Relazione R deve essere rappresentato in uno solo dei frammenti, o in alternativa
- **Replicazione**

Proprieta' dei tipi di frammentazione

- Data una relazione R, si definiscono due tipi di frammenti di R:
- **Frammentazione orizzontale:**
 - Partizione di R in n relazioni (frammenti) $\{R_1, \dots, R_n\}$ tali che:
 - $\text{Schema}(R_i) = \text{schema}(R)$ per tutti gli i;
 - Ogni R_i contiene un sottoinsieme dei record di R
 - Normalmente definito da una selezione, i.e. $R_i = \sigma_{C_i}(R)$
 - **Per la completezza:** $R_1 \cup R_2 \cup \dots \cup R_n = R$
 - **La ricostruibilita' e' sempre garantita** dalla unione
- **Frammentazione verticale:**
 - Partizione di R in n relazioni (frammenti) $\{R_1, \dots, R_n\}$ tali che:
 - $\text{Schema}(R) = L = (A_1, \dots, A_m)$, $\text{Schema}(R_i) = L_i = (A_{i1}, \dots, A_{ik})$
 - **Per la completezza:** $L_1 \cup L_2 \cup \dots \cup L_n = L$
 - **Per garantire la ricostruibilita':**
 - $L_i \cap L_j \supseteq \text{chiave primaria } (R)$ per ogni $i \neq j$

Esempio di frammentazione - 1

- $R = \text{EMPLOYEE} (\text{Empnum}, \text{Name}, \text{Deptnum}, \text{Salary}, \text{Taxes})$
- F1: Frammentazione orizzontale:
 - $\text{EMPLOYEE1} = \sigma_{\text{Empnum} \leq 3} \text{EMPLOYEE}$
 - $\text{EMPLOYEE2} = \sigma_{\text{Empnum} > 3} \text{EMPLOYEE}$
- La ricostruzione richiede l'unione:
 - $\text{EMPLOYEE} = \text{EMPLOYEE1} \cup \text{EMPLOYEE2}$

Esempio di frammentazione - 2

- R = EMPLOYEE (Empnum, Name, Deptnum, Salary, Taxes)
- F2: Frammentazione verticale:
 - $\text{EMPLOYEE1} = \Pi_{\text{EmpNum}, \text{Name}} \text{EMPLOYEE}$
 - $\text{EMPLOYEE2} = \Pi_{\text{EmpNum}, \text{DeptNum}, \text{Salary}, \text{Tax}} \text{EMPLOYEE}$
- La ricostruzione richiede un equi-join sulla chiave in comune (natural join):
 - $\text{EMPLOYEE} = \text{EMPLOYEE1} \text{ join } \text{EMPLOYEE2}$
 - Un esempio di frammentazione verticale che non garantisce la ricostruibilità?

Le due frammentazioni sono corrette perche'

- **Completezza**

F1, F2: Per costruzione delle due tabelle

Ricostruibilita'

F1: ovvio

F2: Empnum, Name \cap Empnum, Deptnum, Salary,
Tax \supseteq Empnum

Disgiunzione

F1, F2: Per costruzione delle due tabelle

Frammentazione verticale

- Dato F2 e supponiamo la seguente Query
 - $\text{EMPLOYEE1} = \Pi_{\text{EmpNum}, \text{Name}} \text{EMPLOYEE}$
 - $\text{EMPLOYEE2} = \Pi_{\text{EmpNum}, \text{DeptNum}, \text{Salary}, \text{Tax}} \text{EMPLOYEE}$
- UPDATE EmpNum SET EmpNum = 30
WHERE EmpNul = 6749 and Tax=20;
- Su quale dei due frammenti si applica?
- Cosa succede alla ricostruibilità
- È necessario propagare le modifiche su tutti i frammenti.



Replicazione

Replicazione

- Aspetti positivi
- La replicazione migliora le prestazioni
 - Consente la coesistenza di applicazioni con requisiti operazionali diversi sugli stessi dati
 - Aumenta la localita' dei dati utilizzati da ogni applicazione
- Ma aspetti negativi
- Introduce complicazioni architetturali:
 - Gestione della transazioni e updates di copie multiple, che devono essere tutte aggiornate
- Richiede un nuovo passo di progettazione:
 - Quali frammenti replicare
 - Quante copie mantenere
 - Dove allocare le copie
 - Politiche di gestione delle copie



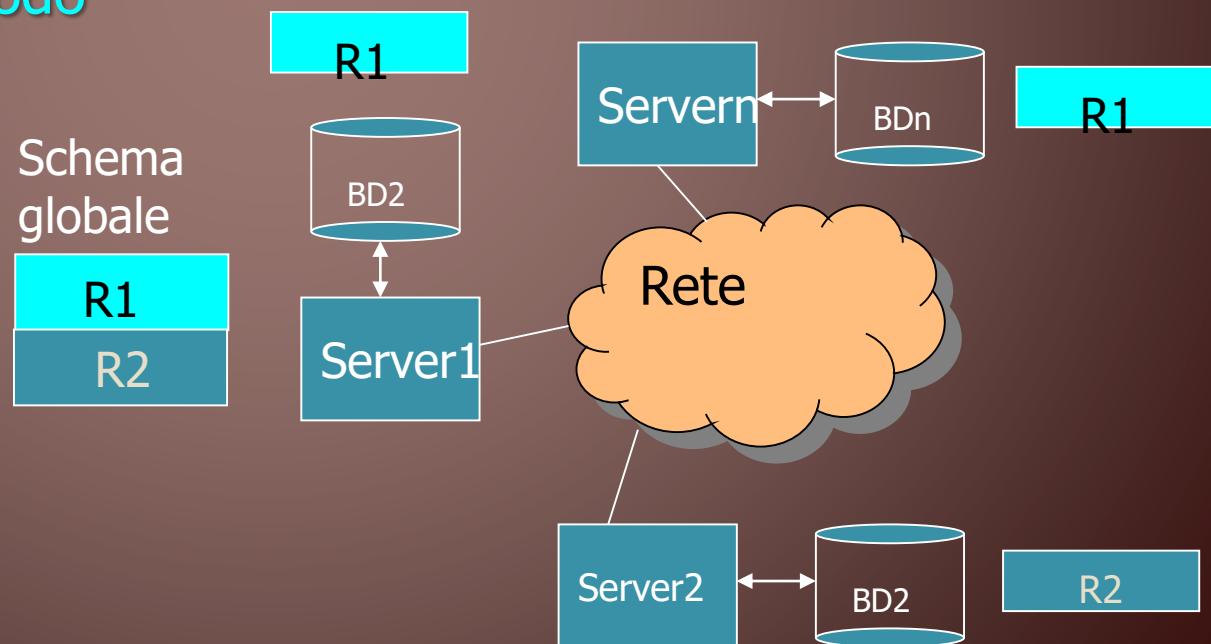
Allocazione dei frammenti: lo schema di allocazione o catalogo

Schemi di allocazione dei frammenti

- Ogni frammento e' allocato in generale su un nodo diverso (e su file diversi)
- Quindi, lo schema globale esiste solo in modo virtuale
 - Non materializzato su un unico nodo
- Lo **schema di allocazione descrive il mapping:**
 - **Frammento → nodo**

Catalogo

Frammento	Nodo
1	1
1	n
2	2





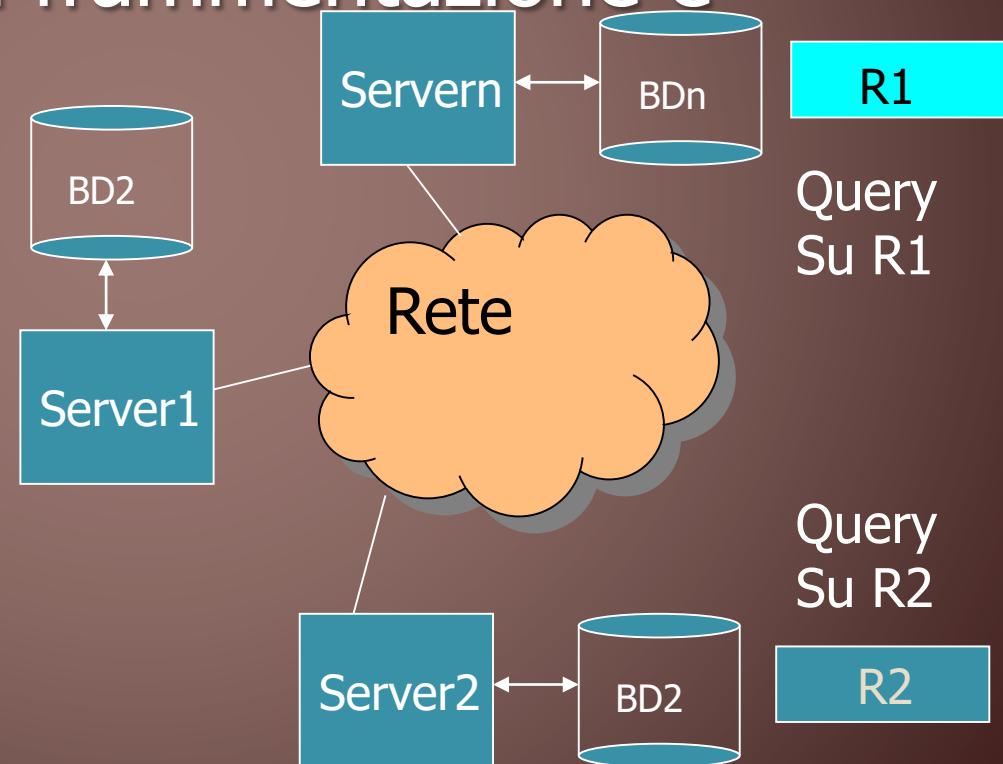
Trasparenza

Trasparenza nei DDBMS

- Separazione della semantica di alto livello dalle modalita' di frammentazione e allocazione

Query
Espressa
Su R

R



Perché è importante?

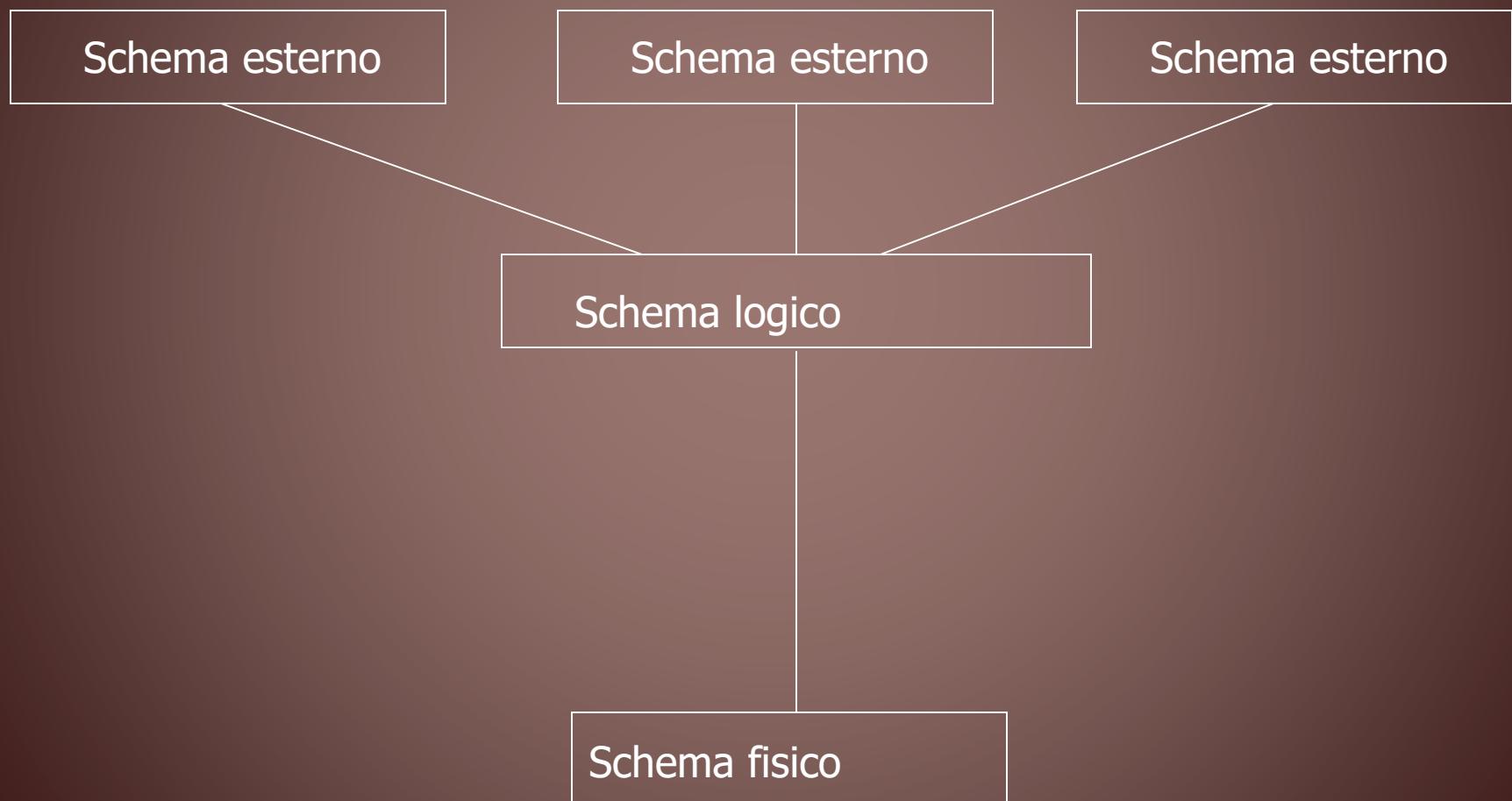
- Separazione dei concetti
 - Logica applicativa
 - Logica dei dati
- Ma....
 - Serve uno strato software che gestisce la traduzione dallo schema unico ai sottoschemi
 - Aumento della complessità del sistema
 - Perdita di performance
 - ...
- Tuttavia
 - Se il mapping è nativamente supportato dal DDBMS i problemi si riducono (non si eliminano)

La trasparenza nei DBMS e' di due tipi

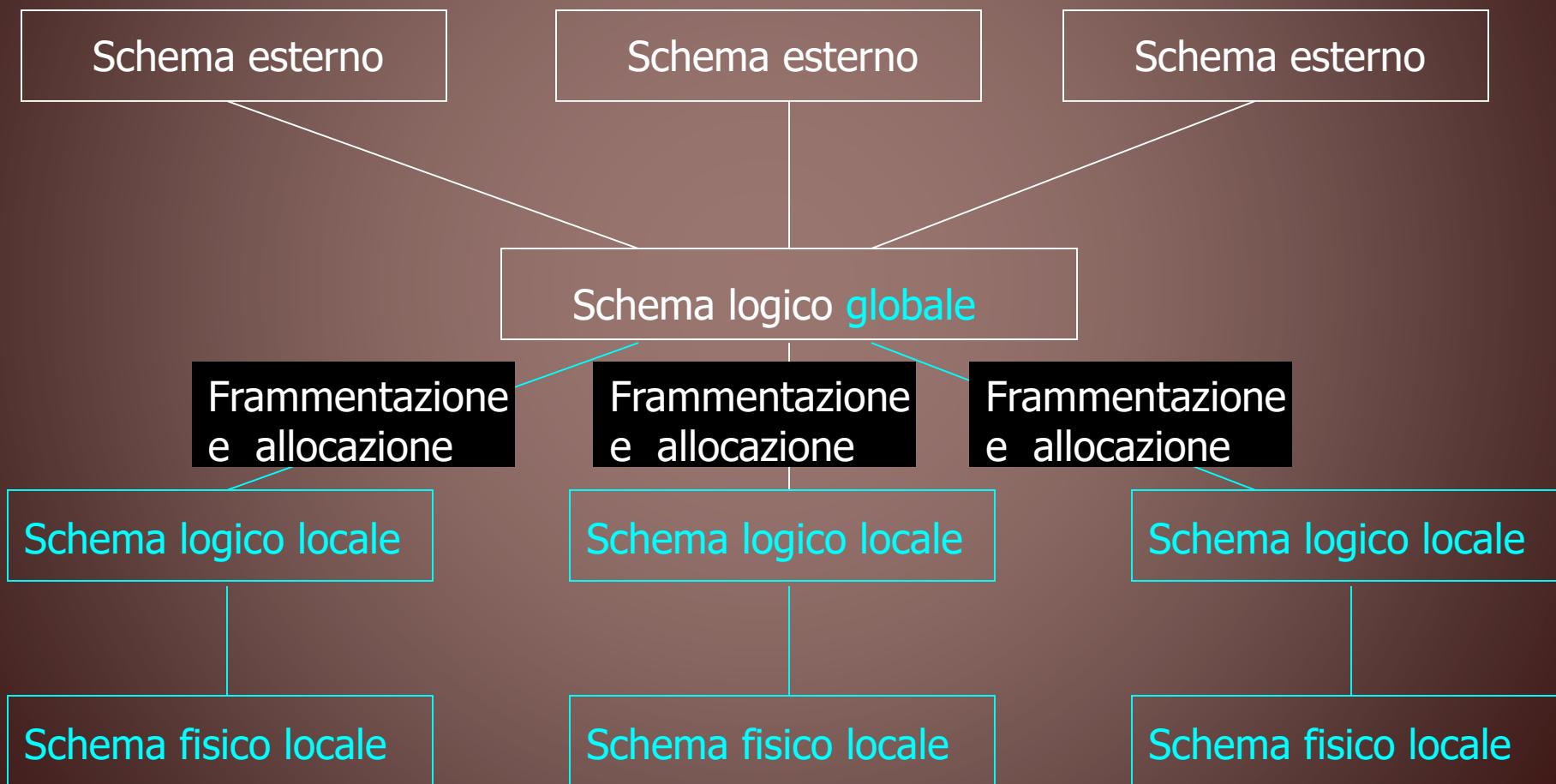
Le applicazioni (transazioni, interrogazioni) non devono essere modificate a seguito di cambiamenti nella definizione e organizzazione dei dati

- 1. Trasparenza (o indipendenza) logica: indipendenza dell'applicazione da modifiche dello schema *logico*
 - Un'applicazione che utilizza un frammento dello schema non subisce modifiche quando altri frammenti vengono modificati
- 2. Trasparenza (o indipendenza) fisica: indipendenza dell'applicazione da modifiche dello schema *fisico*
- Garantite dai tre livelli ANSI – SPARC →

Architettura dati di riferimento per un DBMS



Architettura dati di riferimento per un DDBMS





Livelli di trasparenza nei DDBMS

Livelli di trasparenza nei DDBMS

- Trasparenza di Frammentazione
- Trasparenza di Replicazione
(Allocazione)
- Trasparenza di Linguaggio

Schema esempio

SUPPLIER(SNum, Name, City)

Frammenti logici

- Due frammenti orizzontali:
 - SUPPLIER1 = $\sigma_{City='London'}$ SUPPLIER
 - SUPPLIER2 = $\sigma_{City='Manchester'}$ SUPPLIER
- Schema di allocazione su tre nodi:
 - SUPPLIER1@company.London.uk
 - SUPPLIER2@company.Manchester1.uk
 - SUPPLIER2@company.Manchester2.uk
- Interrogazione

Due repliche dello stesso frammento

Select Name from Supplier where Snum = snum

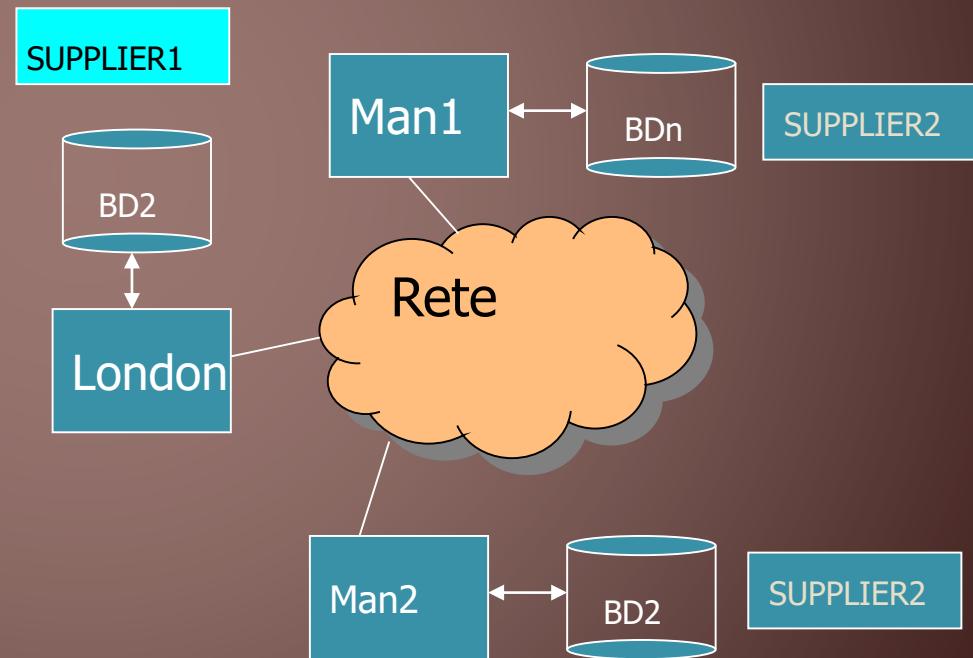
Trasparenza di Frammentazione

- L'applicazione ignora del tutto l'esistenza di frammenti
- E' lo scenario migliore dal punto di vista della programmazione applicativa
- L'applicazione e' scritta in SQL standard

Trasparenza di frammentazione

- Esempio di query “Trova il supplier con SNum = snum”

```
procedure
Query1(:snum,:name)
select Name into :name
  from Supplier
 where SNum = :snum;
end procedure
```



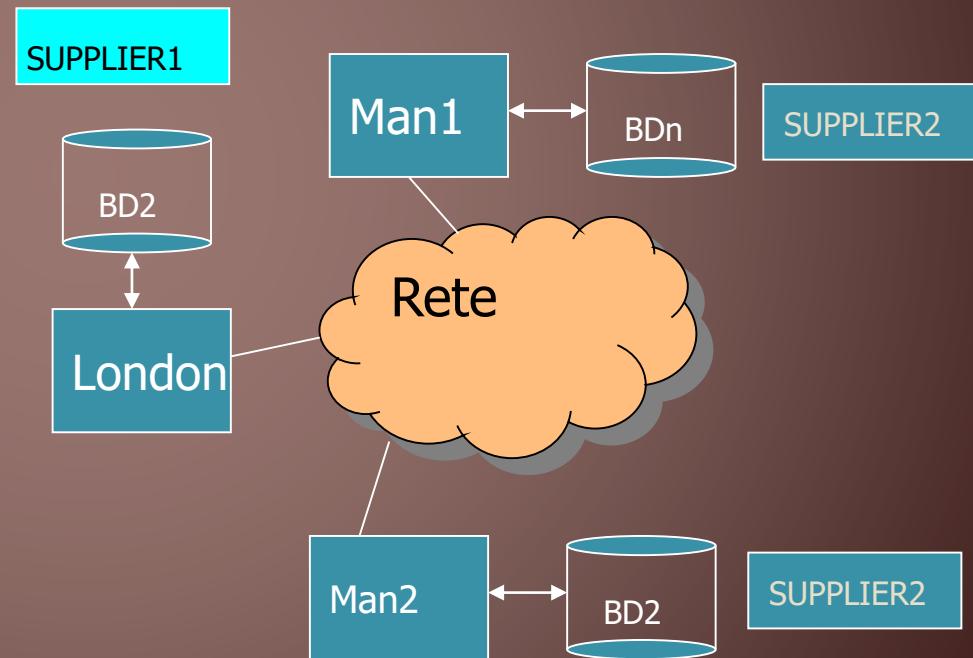
Trasparenza di Frammentazione

- E' a carico del sistema la traduzione della query globale in query locali
 - Frammenti: relazione → sotto-relazioni
 - Global query → fragment queries
- Problema: una query definita su un'intera relazione va ora scomposta in piu' query, una per ogni sotto-relazione
 - Query Rewriting

Trasparenza di Allocazione

- L'applicazione e' consapevole dei frammenti, ma ne ignora l'allocazione sui nodi

```
procedure
Query2(:snum,:name);
select Name into :name
  from SUPPLIER1
  where SNum = :snum;
if isEmpty(:name) then
  select Name into :name
    from SUPPLIER2
    where SNum = :snum;
end procedure;
```



Trasparenza di linguaggio

- L'applicazione deve specificare sia i frammenti che il loro nodo
- E' il livello minimo di trasparenza tra i tre descritti
- Un nodo puo' offrire interfacce che non sono standard SQL
- Tuttavia, l'applicazione e' scritta in SQL standard, a prescindere da eventuali altri linguaggi locali al nodo

Trasparenza di linguaggio

Queries espresse a livelli più alti di trasparenza vengono tradotte a questo livello dall' ottimizzatore di query distribuite

```
procedure Query3(:snum,:name);
select Name into :name
from
SUPPLIER1@company.London.uk
where SNum = :snum;
if :empty then
select Name into :name from
SUPPLIER2@company.Manchester
1.uk
where SNum = :snum;
end procedure;
```

Indirizza esplicitamente una delle due repliche

