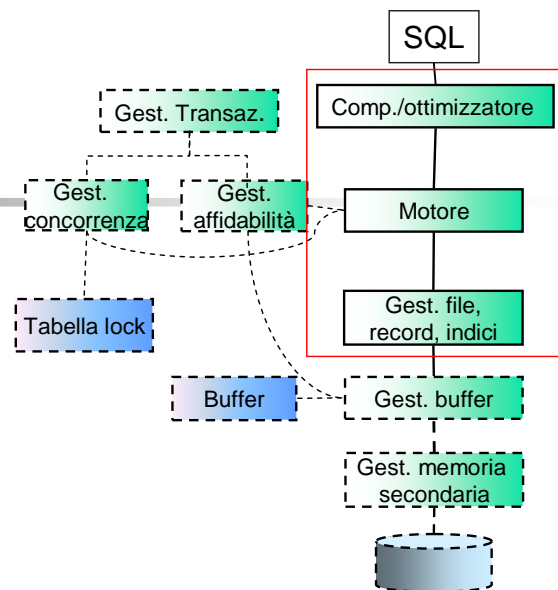
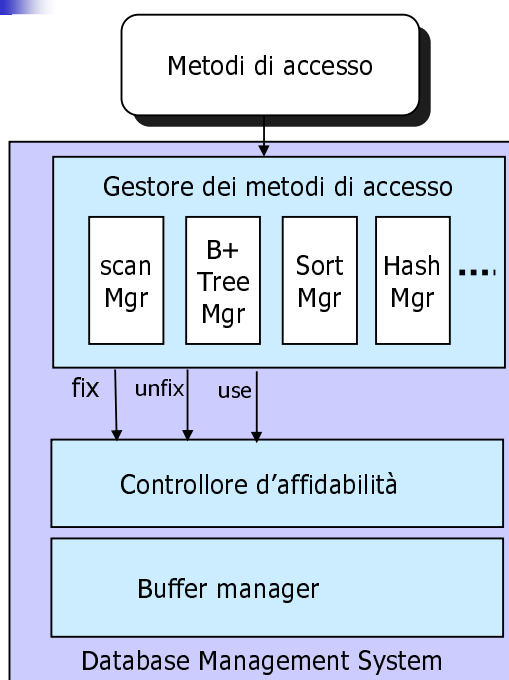


Esecuzione delle interrogazioni

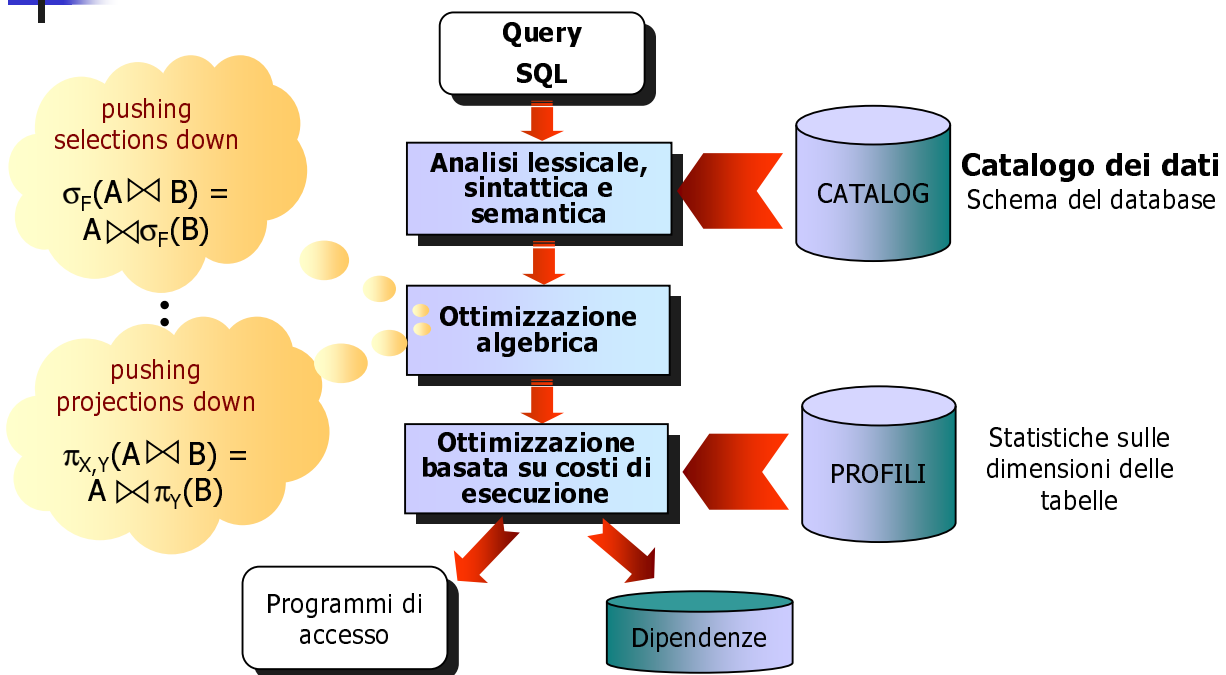


Strutture fisiche di accesso



- Riguardano l'organizzazione dei dati per rendere efficienti le operazioni di ricerca e modifica
- Si possono definire indici
- I metodi di accesso gestiscono una particolare organizzazione fisica
- Il metodo di accesso individua i blocchi fisici che devono essere caricati in memoria dal buffer manager

Ottimizzazione delle interrogazioni



Compilazione delle query

- I programmi di accesso sono in formato "oggetto"
- **compile-and-store**
 - l'interrogazione viene compilata una sola volta
 - il codice oggetto viene memorizzato insieme alle dipendenze dalle versioni di tabelle e indici
 - il codice oggetto viene invalidato se la struttura della base di dati cambia significativamente per l'interrogazione (es. aggiunta di un indice)
- **compile-and-go**
 - l'interrogazione viene compilata ed eseguita, ma non è memorizzata

Analisi lessicale, sintattica e semantica

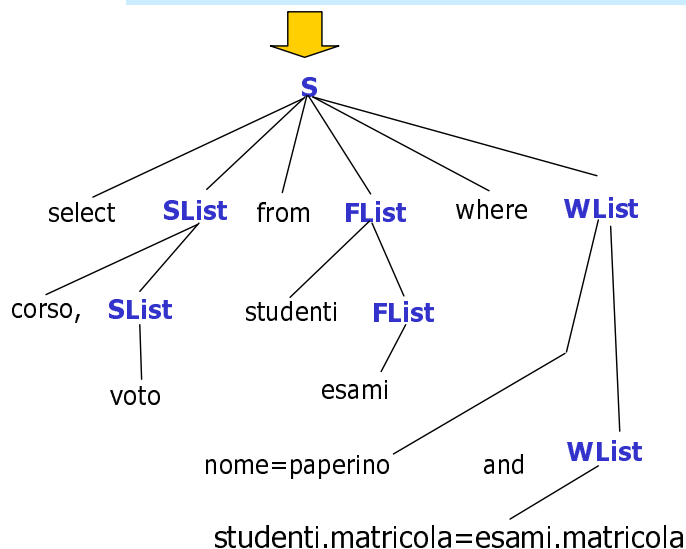
Analisi lessicale e sintattica

- un parser legge la query in SQL e produce il rispettivo albero di analisi
- il parser è costruito a partire dalla grammatica dell'SQL con le tecniche usate anche per gli altri linguaggi

Analisi semantica:

- si controlla l'esistenza delle tabelle e attributi indicati nell'interrogazione
- si associa ad ogni nome il relativo oggetto
- si controlla che i tipi dei dati coinvolti nelle operazioni sia corretto

```
SELECT corso, voto
FROM studenti, esami
WHERE nome='paperino'
AND studenti.matricola=esami.matricola
```



Traduzione dell'interrogazione in algebra relazionale

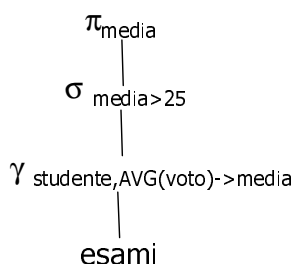
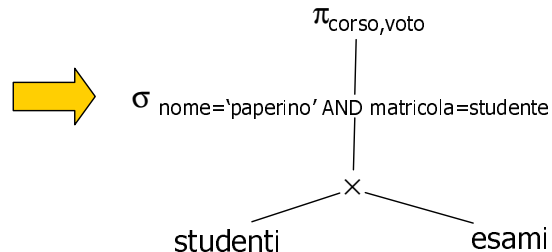
Traduzione in algebra relazionale

- l'albero sintattico viene tradotto in un altro albero che rappresenta l'interrogazione in algebra relazionale estesa
 - le foglie contengono tabelle
 - i nodi interni operatori relazionali
- operatori dell'algebra
 - unione, intersezione, differenza
 - selezione (ad es. $\sigma_{\text{nome}=\text{'paperino'}}(\text{STUDENTI})$)
 - proiezione (ad es. $\pi_{\text{matricola}}(\text{STUDENTI})$)
 - prodotto e join (\times, \bowtie)
 - aggregazione (ad es. $\gamma_{\text{matricola}, \text{AVG}(\text{voto}) \rightarrow \text{media}}(\text{ESAMI})$)
 - ordinamento (ad es. $\tau_{\text{nome}}(\text{STUDENTI})$)

Traduzione dell'interrogazione in algebra relazionale II

- nel caso di una interrogazione select-from-where la trasformazione è semplice
- se l'interrogazione contiene operatori di raggruppamento o ordinamento, la traduzione richiede l'uso dell'algebra estesa

```
SELECT corso, voto
FROM studenti, esami
WHERE nome='paperino'
AND matricola=studente
```

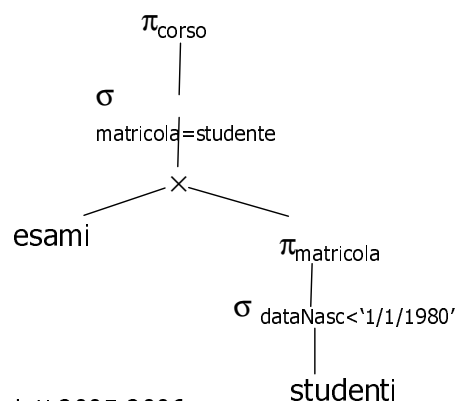
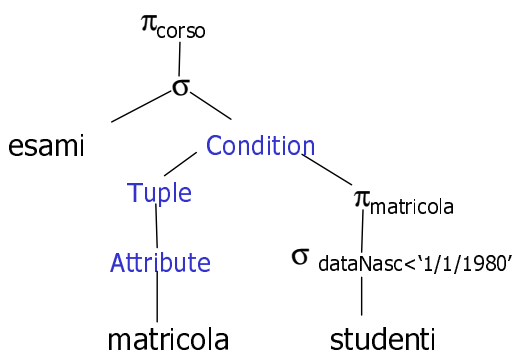


```
SELECT AVG(voto) AS media
FROM esami
GROUP BY studente
HAVING media > 25
```

Traduzione dell'interrogazione in algebra relazionale III

- nel caso di una interrogazione che contenga una sottointerrogazione
 - prima, si trasforma parzialmente l'interrogazione in un albero con un nodo speciale che rappresenta la sottointerrogazione
 - poi, si tenta di riscrivere la sottointerrogazione in una espressione equivalente

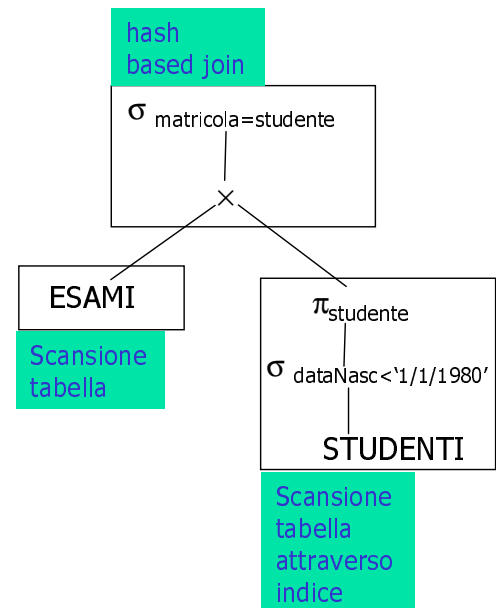
```
SELECT corso FROM esami
WHERE studente IN (SELECT matricola FROM studenti WHERE dataNasc < '1/1/1980')
```



Query plan

Query plan

- è una **racpresentazione interna** della query
- query plan **logico**
 - è un albero rappresentante un'espressione relazionale
 - suggerisce le operazioni logiche da eseguire e l'ordine in cui devono essere eseguite
- query plan **fisico**
 - è un albero indicante le operazioni fisiche necessarie ad implementare l'interrogazione
 - ogni nodo implementa un insieme di operazioni descritte dal query plan logico



Query plan II

La trasformazione da query plan logico a query plan fisico

- **ottimizzare il query plan logico** (ottimizzazione algebrica)
 - si cercano, fra quelle algebricamente equivalenti, le espressioni piu' efficienti
- **implementare gli operatori dell'algebra relazione**
 - trasformare gli operatori in operazioni da richiedere al gestore dei metodi d'accesso
- **ottimizzare il query plan fisico** (ottimizzazione basata sui costi di esecuzione previsti)
 - scegliere fra le possibili implementazioni quella piu' efficiente sulla base dello stato attuale del database

Ottimizzazione algebrica

Il primo passo dell'ottimizzazione

- si riscrive l'espressione relazionale in nuove espressioni
 - equivalenti
 - piu' efficienti
- in questa fase **non si usa informazione** sullo stato attuale del database
- la riscrittura si basa su formule di equivalenza

Idee guida

- minimizzare i numero di record nelle relazioni intermedie
- minimizzare la dimensione delle relazioni intermedie

Ottimizzazione algebrica II

Pushing selections

- consiste nel spingere il piu' possibile **verso le foglie** (anticipare) le selezioni
- minimizza il numero di record trattati
- alcune equivalenze:
 - $\sigma_C(R-S) = \sigma_C(R) - \sigma_C(S)$ (puo' convenire piu' di $\sigma_C(R-S) = \sigma_C(R) - S$)
 - $\sigma_C(R \bowtie_D S) = \sigma_C(R) \bowtie_D \sigma_C(S)$
 - $\sigma_C(R \bowtie_D S) = \sigma_C(R) \bowtie_D S$ (se C non contiene attributi di S)
 -

$\pi_{\text{corso,voto}}(\sigma_{\text{nome='paperino'}}(\text{ESAMI} \bowtie_{\text{matricola=studente}} \text{STUDENTI}))$



$\pi_{\text{corso,voto}}(\text{ESAMI} \bowtie_{\text{matricola=studente}} (\sigma_{\text{nome='paperino'}}(\text{STUDENTI})))$

Ottimizzazione algebrica III

Pushing projections

- consiste nel spingere il piu' possibile verso le foglie (anticipare) le proiezioni
- piu' in generale le proiezioni possono essere aggiunte ovunque, purché' tolgano solo attributi che non verranno piu' usati
- alcune equivalenze
 - $\pi_L(\sigma_C(R)) = \pi_L(\sigma_C(\pi_M(R)))$
dove M sono gli attributi di R che compaiono in C o in L
 - $\pi_L(R \bowtie_D S) = \sigma_C(\pi_M(R)) \bowtie_D \sigma_C(\pi_N(S))$
dove M (N) sono gli attributi di R (S) che compaiono in D o in L

$\pi_{\text{corso,voto}}(\sigma_{\text{nome='paperino'}}(\text{ESAMI} \bowtie_{\text{matricola=studente}} \text{STUDENTI}))$



$\pi_{\text{corso,voto}}(\text{ESAMI} \bowtie_{\text{matricola=studente}} (\sigma_{\text{nome='paperino'}}(\pi_{\text{nome,matricola}}(\text{STUDENTI}))))$

Scarselli Franco

Sistemi per basi di dati 2005-2006

73

Realizzazione delle operazioni fisiche

Le operazioni implementate in RDBMS tipici sono

- scansioni sequenziali
- join
- ordinamenti
- accessi tramite indice

Un'altra classificazione

- metodi one-pass che richiedono una sola lettura dei dati
 - di solito funzionano se uno degli operandi sta tutto in memoria
- metodi two-pass che richiedono due letture dei dati
 - ad esempio, il two phase multiway merge-sort
- metodi che richiedono piu' letture dei dati

Scarselli Franco

Sistemi per basi di dati 2005-2006

74



Scansioni (scan)

- Si accede in sequenza alle tuple
 - open - si apre la scansione
 - next - si avanza il puntatore alla tupla successiva
 - read/modify/delete - legge/modifica/cancella la tupla corrente
 - insert - inserisce una nuova tupla nella posizione corrente
 - close - chiude la scansione
- Durante lo scan si possono applicare delle operazioni
 - proiezione, selezione su un predicato semplice, ordinamento



Accesso tramite indici (index scan)

- Gli indici sono realizzati con B-tree (B+ tree)
- Favoriscono l'accesso associativo per interrogazioni che comprendono predicati tipo $A_i = v$ o $v_1 \leq A_i \leq v_2$ (predicati valutabili con l'indice)
- $C_1 \wedge C_2$ dove entrambe le condizioni sono valutabili con indici, si sceglie la più selettiva fra le due per l'accesso tramite indice. Il secondo predicato viene valutato sulle pagine caricate nel buffer. Se gli indici sono densi si possono valutare entrambe le condizioni sugli indici e poi unire i risultati.
- $C_1 \vee C_2$ se una delle due non è valutabile, occorre fare una scansione completa
- $C_1 \vee C_2$ se sono entrambe valutabili si possono usare gli indici ma non è detto che sia conveniente (se c'è molta sovrapposizione)

Accesso tramite indici II

Definiamo

- $B(R)$ – numero di blocchi che costituiscono la relazione R
- $T(R)$ – numero di record che costituiscono la relazione R
- $V(R,A)$ – numero di valori diversi per l'attributo R di A
- M dimensione in blocchi della memoria disponibile

Costo di realizzazione della selezione $\sigma_{A=v}(R)$

- **clustered index su R.A (tipico per indici primari)**
se i record che hanno valore uguale per A sono memorizzati il più vicino possibile nell'indice
 - circa $B(R)/V(R,A)$
- **nonclustered index su R.A (tipico indici secondari densi)**
se i record che hanno valore uguale sono memorizzati in blocchi diversi
 - circa $T(R)/V(R,A)$
- **senza indice**
 - circa $B(R)$

Scarselli Franco

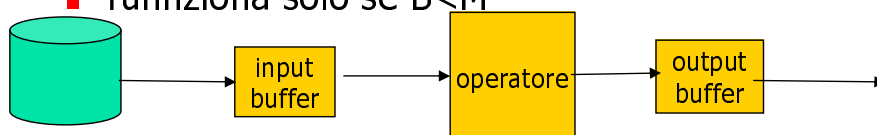
Sistemi per basi di dati 2005-2006

77

Metodi one-pass: esempi

Selezione, proiezione e altri operatori unari

- **funzionamento**
 - si leggono i dati dalla memoria secondaria nel buffer
 - si applica l'operatore
 - si scrive il risultato nel buffer di output
- **costo**
 - se i dati sono **clustered** B (numero dei blocchi) accessi alla memoria secondaria
 - se i dati sono **nonclustered** T (numero di record) accessi alla memoria secondaria
 - funziona solo se $B < M$



Scarselli Franco

Sistemi per basi di dati 2005-2006

78

Metodi one-pass: esempi II

Join

■ funzionamento

- si carica la prima relazione R dalla memoria secondaria alla memoria primaria (ad. es. tabella hash)
- si costruisce una struttura di accesso ad R
- si legge la seconda relazione S un blocco alla volta
- per ogni record di S si controlla se esiste uno o più record in R che soddisfano il join e si scrive il record risultato in un buffer di uscita

■ costo

- se i dati sono **clustered** $B(R)+B(S)$
- funziona solo se $B(R) < M$

Metodi two pass

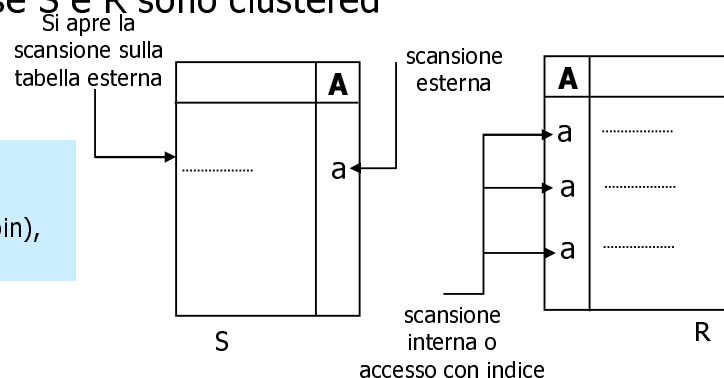
Cosa succede se le relazioni non entrano in memoria ?

Nested-loop Join

- si sceglie la relazione con un **numero minore di record** (S)
- si scandisce S e per ogni record di S si lancia una scansione su R

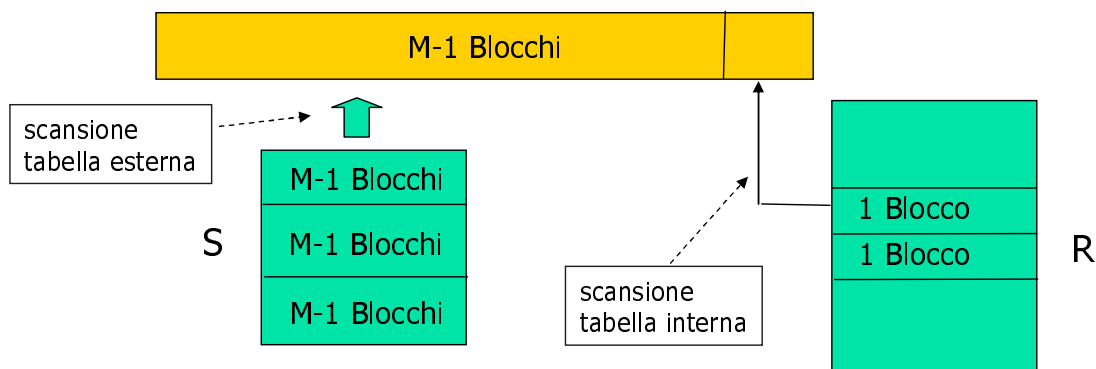
Servono $B(S)+T(S)*B(R)$ letture se S e R sono clustered

```
FOR EACH tupla s in S DO
  FOR EACH tupla r in R DO
    IF (r and s soddisfano la condizione di join),
      THEN costruisci la tupla di unione
```



Nested-loop join

- un miglioramento si ottiene di usare un indice per R
 - costo con R e S clustered: circa $B(S) + T(S) * (B(R)/V(R,A))$
- oppure caricando nei buffer M-1 record di S e confrontandoli tutti con i record di R recuperati dalla seconda scansione
 - costo con R e S clustered: circa $(B(S)/M-1) * (M-1 + B(R))$



Scarselli Franco

Sistemi per basi di dati 2005-2006

81

Merge scan join

Funzionamento

- si ordinano le due relazioni R e S
- si fa il "merge" delle due relazioni scandendole contemporaneamente

Osservazioni

- il costo di questo metodo è $5 (B(R) + B(S))$
 - $4 (B(R) + B(S))$ è il costo dell'ordinamento, $(B(R) + B(S))$ il costo del merge
- un miglioramento si ottiene fondendo la seconda fase dell'ordinamento con il "merge" $3(B(R) + B(S))$
- se sono definiti degli indici, l'ordinamento non è più necessario
- nested-loop join è più efficiente solo se una delle due relazioni è particolarmente piccola
- L'algoritmo richiede $M^2 > (\max(B(R), B(S)))$

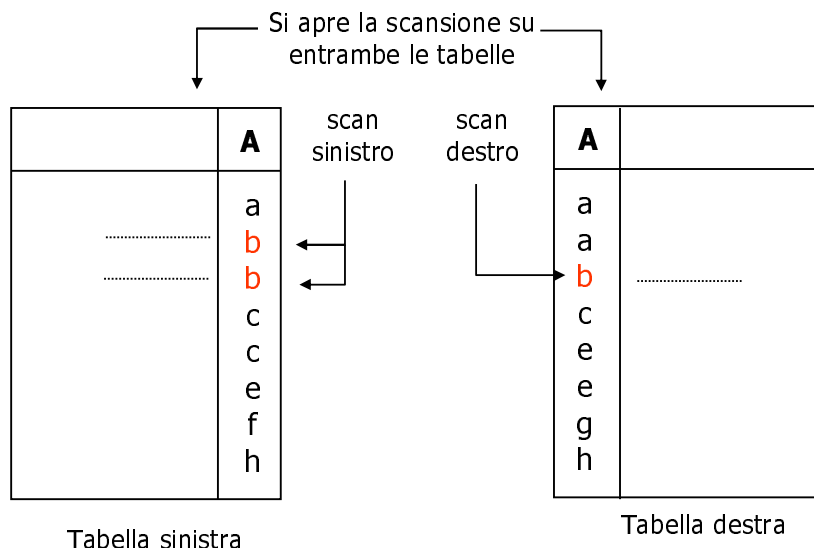
Scarselli Franco

Sistemi per basi di dati 2005-2006

82

Merge scan join II

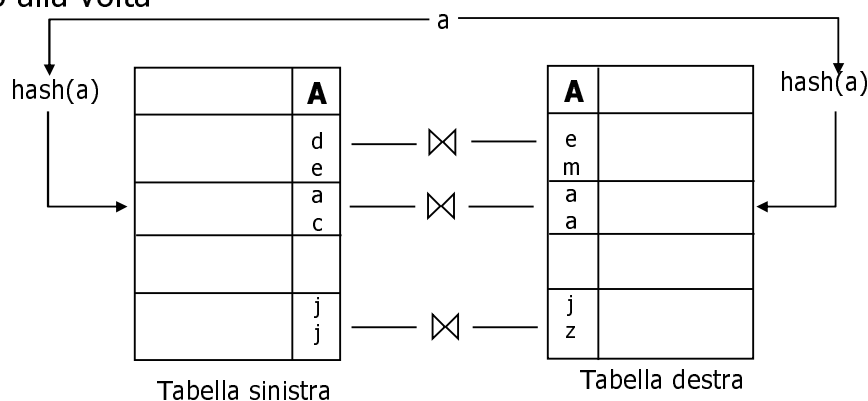
- Le tabelle devono essere ordinate in base agli attributi di join



Hash based join

Funzionamento

- le relazioni R e S sono **riorganizzate attraverso una funzione hash** che indica in quale bucket (fra M-1) ogni record deve essere memorizzato
 - in questo modo record con lo **stesso campo** finiscono nello **stesso bucket**
- si carica un bucket di S e il corrispondente di R e si uniscono i record
 - la memoria deve contenere uno dei bucket, l'altro può essere caricato un blocco alla volta



Hash based join

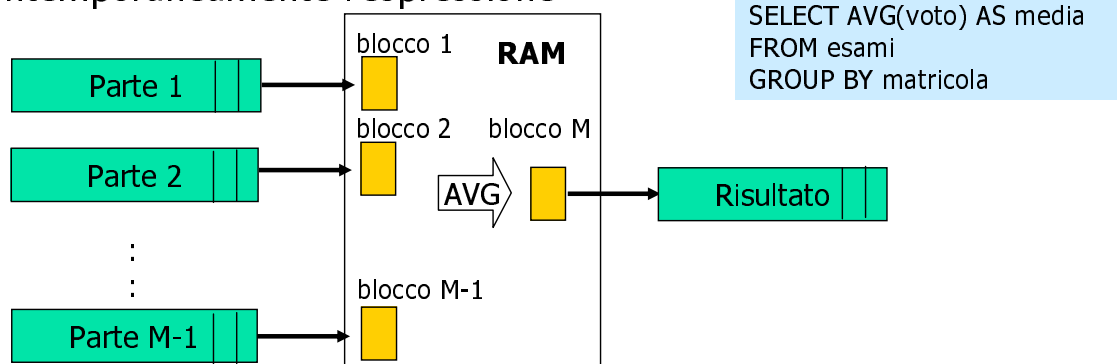
Osservazioni

- il costo di questo metodo è $3 (B(R) + B(S))$
 - $2 (B(R) + B(S))$ è il costo della prima fase
 - $(B(R) + B(S))$ il costo della seconda fase
- si richiede che la memoria contenga i bucket di S
 - implica $M^2 > \min(B(R), B(S))$
- l'hash based join è leggermente più veloce del merge scan join se non sono presenti indici
- l'hash based join può trattare relazioni leggermente più grandi
- il merge scan join ha il vantaggio di lasciare ordinati i dati

Altri esempi di algoritmi two pass

Raggruppamento basato su ordinamento

- si suddivide la relazione R in M-1 sottoparti R_1, \dots, R_{M-1}
- si ordinano R_1, \dots, R_{M-1} in base alla chiave del group by
- si caricano R_1, \dots, R_{M-1} in M-1 blocchi
- si scelgono i record con la stessa chiave valutando contemporaneamente l'espressione





Altri esempi di algoritmi two pass II

Raggruppamento basato su Hashing

- si suddivide la relazione R in M-1 buckets usando la funzione hash sulle chiavi di raggruppamento
- ogni bucket viene caricato in memoria, si scelgono i record con la stessa chiave valutando contemporaneamente l'espressione
- occorrono $3B(R)$ accessi alla memoria secondaria
- deve essere verificato $M^2 > B(R)$



Algoritmi che richiedono piu' di tre passi

Algoritmi Multipass

- servono ad elaborare relazioni troppo grandi per essere elaborate in due passi
- esistono estensioni degli algoritmi basati sull'ordinamento e di quello basati su hashing

Basati su ordinamento

- i dati vengono **ricorsivamente** divisi in M-1 parti R_1, \dots, R_n uguali ($n > M$) fino quando la memoria non li contiene
- le parti R_1, \dots, R_n sono riunite insieme valutando contemporaneamente la funzione voluta

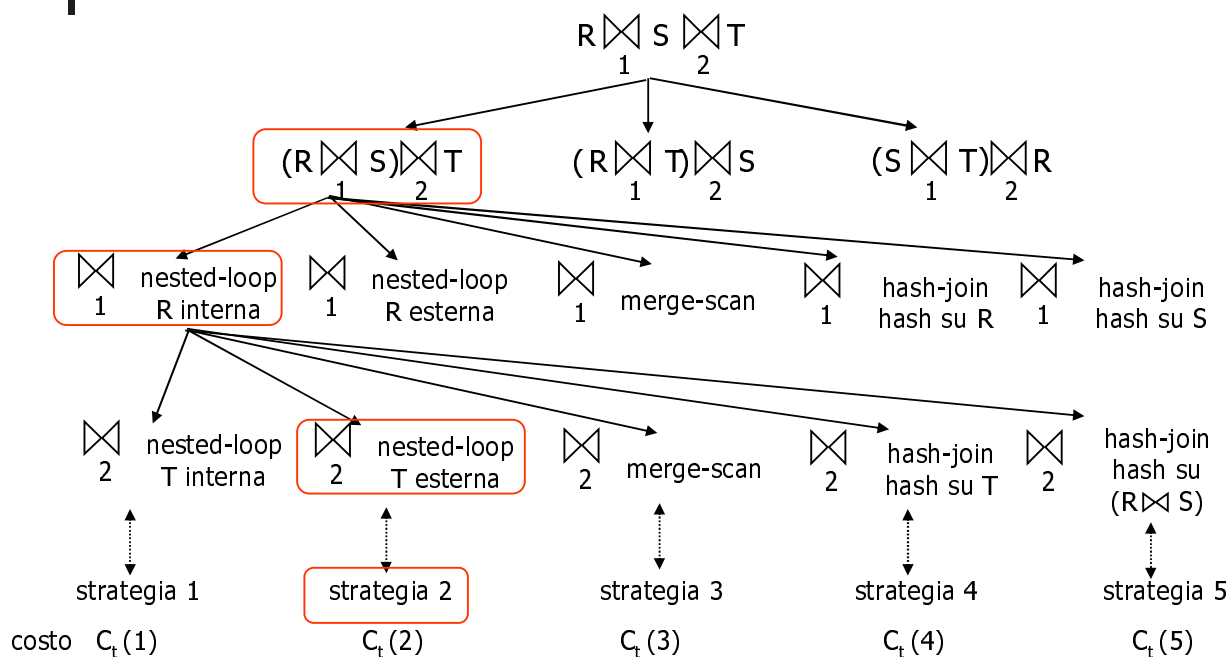
Basati su hashing

- i dati vengono **ricorsivamente** divisi in M-1 bucket R_1, \dots, R_n ($n > M$) con una funzione di hash fino quando la memoria non li contiene
- la funzione voluta viene calcolato separatamente su tutti i bucket R_1, \dots, R_n o coppie di bucket

Ottimizzazione dei costi

- Le dimensioni per l'ottimizzazione sono molte
 - tipologia dell'operazione di accesso ai dati (es. scan/indice)
 - ordine delle operazioni (es. ordine dei join)
 - modalità di realizzazione di una operazione (es. modalità di join)
 - livello in cui eseguire gli ordinamenti
- Si costruisce un albero delle alternative
 - ogni nodo corrisponde ad una scelta di un'opzione
 - ogni foglia rappresenta una strategia di esecuzione (descritta dal cammino dalla radice alla foglia)
 - Si dovrebbe scegliere il percorso radice-foglia con il costo minore

Esempio





Profili delle relazioni

- Informazioni quantitative relative alle caratteristiche delle tabelle
 - cardinalità della tabella T - $CARD(T)$
 - dimensione in byte delle tuple di T - $SIZE(T)$
 - dimensione in byte di ciascun attributo - $SIZE(A_j, T)$
 - numero di valori distinti di ciascun attributo - $VAL(A_j, T)$
 - i valori minimo e massimo di ciascun attributo - $MIN(A_j, T)$ $MAX(A_j, T)$
- I profili sono costruiti periodicamente
- I profili permettono di fare delle stime sulle dimensioni dei risultati intermedi delle interrogazioni



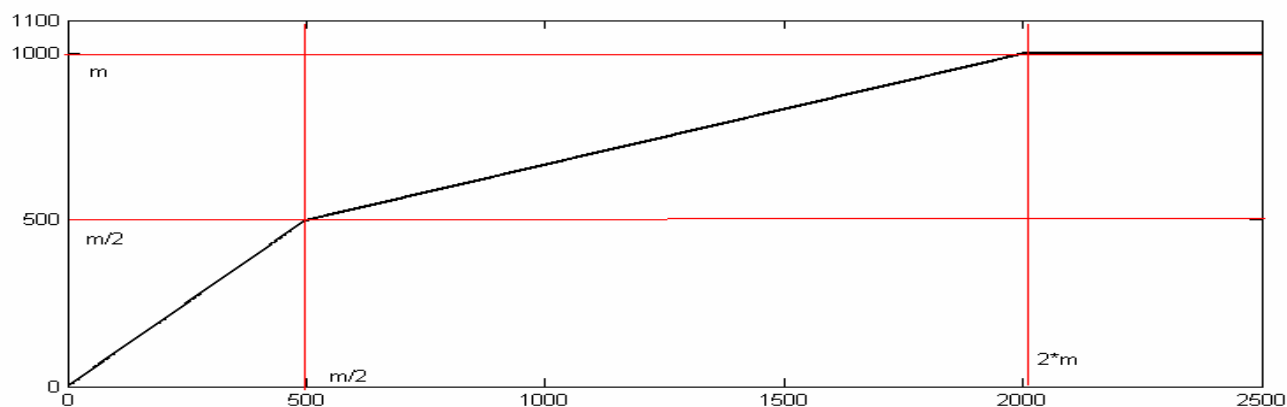
Profili per le selezioni

- Se $T' = \sigma_{A_i=v}(T)$
 - $CARD(T') = CARD(T)/VAL(A_i, T)$ (migliore stima in media)
 - $SIZE(T) = SIZE(T')$
 - $VAL(A_i, T') = 1$
 - $VAL(A_j, T') = col(CARD(T), VAL(A_j, T), CARD(T'))$ $j \neq i$
 - $MAX(A_i, T') = MIN(A_i, T') = v$
 - $MAX(A_j, T')$ $MIN(A_j, T')$ $j \neq i$ mantengono i valori precedenti (ipotesi)
- $\sigma_{C_1 \wedge C_2}(T)$ è equivalente a $\sigma_{C_1} \sigma_{C_2}(T)$
- $\sigma_{C_1 \vee C_2}(T)$ è più difficile da stimare $CARD(T')$ - la somma dei risultati è una sovrastima

Implementazione di col

$\text{col}(n, m, k)$

- Può essere implementata come
 - $\text{col}(n, m, k) = k$, se $k \leq m/2$
 - $\text{col}(n, m, k) = m$, $k \geq 2*m$
 - $\text{col}(n, m, k) = (k+m)/3$ se $m/2 < k < 2*m$



Scarselli Franco

Sistemi per basi di dati 2005-2006

93

Profili per le proiezioni

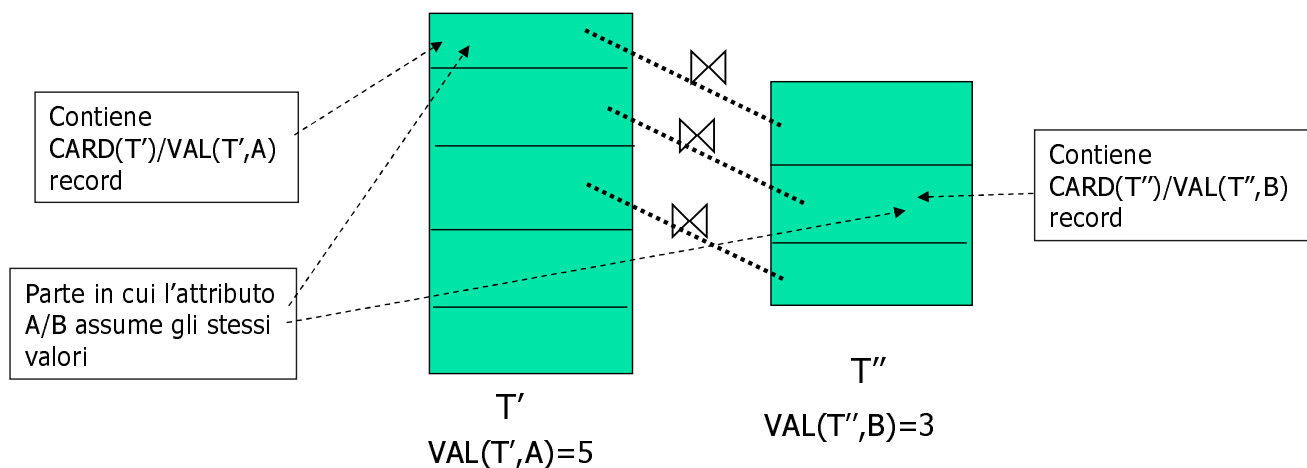
- Se $T' = \pi_L(T)$ $L = \{A_1, A_2, \dots, A_n\}$
 - $\text{CARD}(T') = \text{MIN}(\text{CARD}(T), \prod_{i=1,n} \text{VAL}(A_i, T))$
 - $\text{SIZE}(T') = \sum_{i=1,n} \text{SIZE}(A_i, T)$
 - $\text{VAL}(A_i, T')$ $\text{MAX}(A_i, T')$ $\text{MAX}(A_i, T')$ mantengono i valori precedenti

Profili per il join

- $T^J = T' \bowtie_{A=B} T''$ - si ipotizza $VAL(A, T') = VAL(B, T'')$
 - $CARD(T^J) = CARD(T') * CARD(T'') / VAL(A, T')$
 - $SIZE(T^J) = SIZE(T') + SIZE(T'')$
 - $VAL(A_i, T^J)$ $MAX(A_i, T^J)$ $MAX(A_i, T^J)$ mantengono i valori precedenti nelle rispettive tabelle
- Tutte le formule assumono una distribuzione uniforme dei valori e l'assenza di correlazione fra le varie condizioni presenti in una interrogazione
- Questi dati approssimati sono comunque sufficienti a ottenere l'ottimizzazione

Profili per il join: altri casi

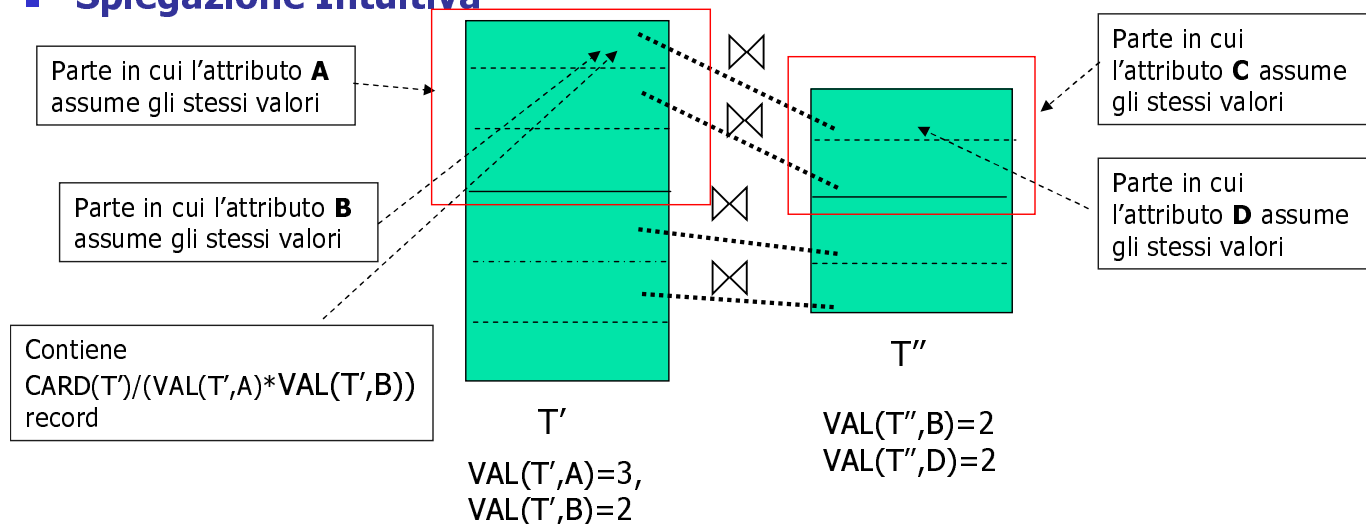
- $T^J = T' \bowtie_{A=B} T''$ - se $VAL(A, T')$ è diverso da $VAL(B, T'')$
 - $CARD(T^J) = CARD(T') * CARD(T'') / \max(VAL(A, T'), VAL(B, T''))$
- **Spiegazione Intuitiva**



Profili per il join: altri casi II

- $T^J = T' \bowtie_{A=B, C=D} T''$
 - $CARD(T^J) = CARD(T') * CARD(T'') / \max(VAL(A, T'), VAL(B, T'')) * \max(VAL(C, T'), VAL(D, T''))$

Spiegazione Intuitiva



Progettazione fisica

- Definizione dei parametri fisici nel DBMS
- Scelta degli indici su ogni relazione
 - permettono di rendere più efficienti selezione e join
 - occorre scegliere gli attributi su cui crearli
 - creare un indice può essere costoso in spazio e tempo in modifica/inserimento
 - spesso le chiavi sono coinvolte in selezioni e join per cui conviene creare indici sui campi chiave (in alcuni DBMS sono creati automaticamente)
 - un indice può velocizzare anche le operazioni di ordinamento
 - su alcuni DBMS si può ottenere il piano degli accessi per verificare se gli indici sono utilizzati