

ARGOMENTO 9

Metodi di storage & Modello di costo

Introduzione

- Le basi di dati sono normalmente memorizzate su dischi magnetici
 - Ai dati si accede mediante la struttura fisica dei file della base di dati
- Gerarchia di memorizzazione
 - Primaria
 - CPU, memoria principale, cache
 - Secondaria
 - Dischi magnetici, memoria flash, dischi a stato solido
 - Terziaria
 - Media rimovibili

Organizzazione della memoria delle basi di dati

- Dati persistenti
 - La base di dati vera e propria
- Dati temporanei
 - Esistono solo durante l'esecuzione di programmi
- Organizzazione dei file
 - Determina come le tuple (o *record*) sono fisicamente memorizzate sul disco
 - Determina come si può accedere ai *record* sul disco

Memorizzazione dei record su disco

- Record: collezione di valori tra loro collegati
 - Ogni valore corrisponde a un campo del record
- Data types:
 - Numerico
 - Stringa
 - Booleano
 - Data/ora
- Binary Large Objects (BLOBs)
 - Oggetti non strutturati

...

- Ovviamente, la lunghezza di un record può variare, anche di molto
- Esempi di ragioni per avere record di lunghezza variabile
 - Uno o più campi hanno lunghezza variabile
 - Uno o più campi sono opzionali
 - ...

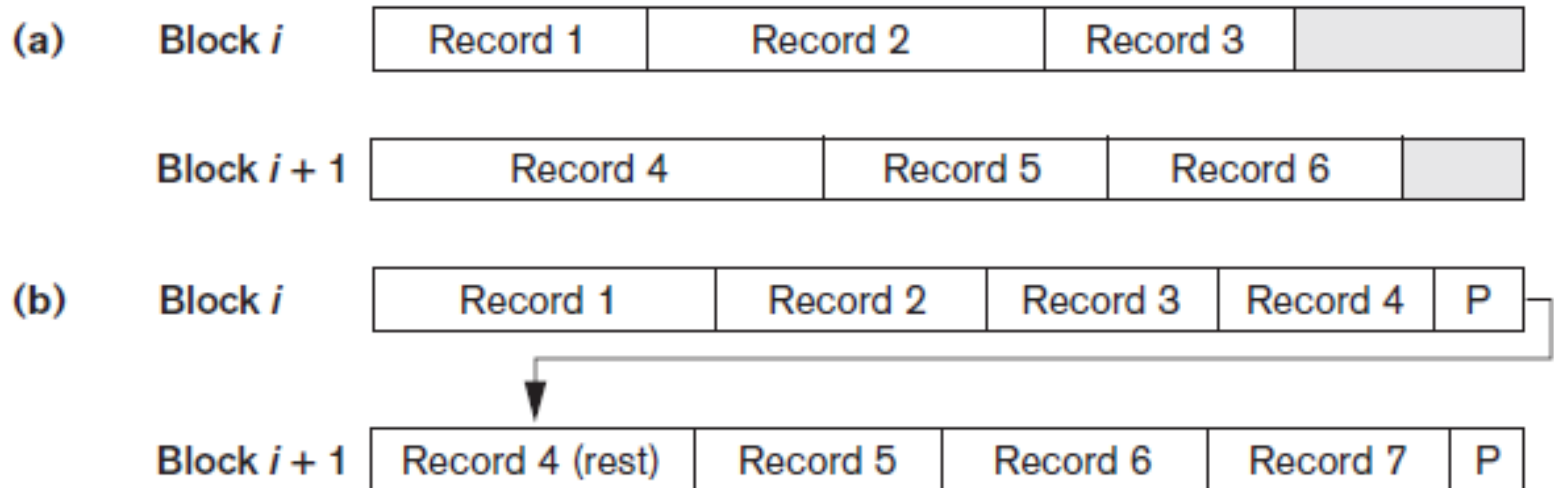
Record e blocchi del disco: allocazione

- I file contenenti record sono allocati in blocchi del disco (spesso chiamati **pagine** quando si parla di gestione di basi di dati)
- Record “esteso” (*spanned block*)
 - E’ un record più grande di un singolo blocco sul disco
 - Un puntatore alla fine del primo blocco punta al blocco contenente il resto del record
- Record non esteso (*unspanned block*)
 - Record che non può superare le dimensioni di un singolo blocco

Record e blocchi del disco (cont.)

(a) Unspanned blocks

(b) Spanned blocks



Operazioni su file

- Retrieval
 - Nessun cambiamento ai dati su file
- Update
 - I file sono modificati mediante inserimento, cancellazione o modifica
- Tipicamente i record sono selezionati mediante condizioni di selezione

Operazioni su file

- Esempi di operazioni che accedono ai record su file:
 - Open (R)
 - Find (R)
 - Read (R)
 - FindNext (R)
 - Delete (U)
 - Insert (U)
 - Close (R)
 - Scan (R)

Tuple, tabelle, pagine

Prendiamo la seguente relazione R (tabella):

- La prima tupla occupa 19 byte (4+5+6+4)
- Per semplicità, nel seguito assumeremo che ogni tupla occupi lo stesso spazio (es. quello della tupla più lunga)
- Indichiamo con t_R la dimensione di una tupla t appartenente a una relazione R

id	name	city	age
1	Marco	Trento	33
2	Andrea	Cosenza	24
3	Giuseppe	Milano	27
4	Federica	Trento	20
5	Rossana	Cosenza	18
6	Laura	Trento	25

Tuple, tabelle, pagine (cont.)

- Il DBMS organizza logicamente il disco, raggruppando più settori in una pagina
 - Ad esempio, in MySQL la dimensione predefinita di una pagina è di 16KB (se un settore è di 512 byte, sono 32 settori per pagina)
- Per ogni operazione da eseguire (lettura, scrittura), il DBMS carica sempre pagine intere (e mai solo parti di pagina)

id	name	city	age
1	Marco	Trento	33
2	Andrea	Cosenza	24
3	Giuseppe	Milano	27
4	Federica	Trento	20
5	Rossana	Cosenza	18
6	Laura	Trento	25

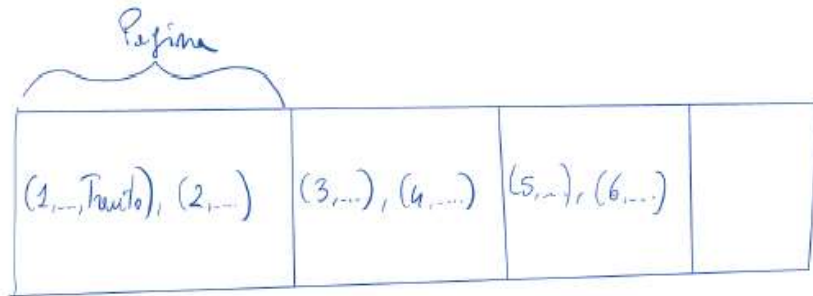
Costo delle operazioni

- Introduciamo ora un modello di costo basato sul numero di pagine che il DBMS deve leggere / scrivere per eseguire una certa operazione
- Essendo le operazioni di lettura / scrittura in memoria centrale MOLTO più veloci, il modello ignorerà il costo delle operazioni in memoria centrale per concentrarsi solo sulle operazioni in memoria secondaria

Metodi di memorizzazione di una relazione R

- *Heap (o pile) file*
 - Le tuple sono semplicemente scritte una dopo l'altra nell'ordine di inserimento
- *Sorted file* (file di tuple ordinate)
 - Le tuple sono ordinate in base al valore di un attributo (o insieme di attributi)
- *Indexed file*
 - Viene creato un indice separato rispetto al file che contiene le tuple vere e proprie della relazione

Heap file



- L'inserimento di nuove tuple in un heap file è molto facile e veloce:
 - Le tuple vengono memorizzate in pagine in base all'ordine di inserimento
 - Quando una pagina è piena, si passa alla successiva eventualmente lasciando spazio vuoto in fondo alla pagina precedente (così si evita di leggere due pagine per leggere una singola tupla a cavallo tra le due pagine)

Heap file: costo delle principali operazioni

SCAN

- Immaginiamo di voler leggere tutte le tuple di una relazione R con una query del tipo:

SELECT * FROM R ;

- Se P_R è il numero di pagine che contengono le tuple di R , allora:

$$\mathbf{Costo}_{\text{Scan}} = P_R$$

[ovvero il costo di leggere tutte le pagine che contengono le tuple di R]

Heap file - Costo delle principali operazioni

EQUALITY SEARCH

- Immaginiamo di voler trovare tutte le tuple di una relazione R che corrispondono a un filtro:

```
SELECT * FROM R WHERE city = 'Trento';
```

dove $city$ è la *chiave di ricerca*.

- Se P_R è il numero di pagine che contengono le tuple di R , allora anche in questo caso:

$$\mathbf{Costo}_{EqSearch} = P_R$$

[ovvero il costo di leggere tutte le pagine che contengono le tuple di R]

Heap file - Costo delle principali operazioni

INSERT

- Immaginiamo di voler inserire una o più nuove tuple in R.

$$\text{Costo}_{\text{Insert}} = 2$$

ovvero il costo di leggere l'ultima pagina del file (o eventualmente crearne una nuova se non ci fosse spazio per la/le nuove tuple) e poi scrivere la pagina modificata sul disco

NB: per trovare l'ultima pagina basta trovare il numero totale di pagine dividendo la dimensione S del file (nota) per la dimensione P delle singole pagine (fissa), ovvero $P_R = S/P$, e poi leggere la pagina $(P_R - 1)$

Heap file: costo delle principali operazioni

DELETE

- Immaginiamo di voler cancellare delle tuple di una relazione R con una query del tipo:

```
DELETE FROM R WHERE city='Trento';
```
- Se P_R è il numero di pagine che contengono le tuple di R e $|R_{A=c}|$ è il numero di tuple in cui l'attributo A (es. `city`) ha valore c (es. `Trento`) allora:

$$P_R + |R_{A=c}| = P_R + \lceil f \cdot |R| \rceil$$

Costo_{Delete} =

Heap file: costo delle principali operazioni

DELETE

- Cosa facciamo se non conosciamo a priori il valore di $|R_{A=c}|$ (per un qualsiasi attributo A di R e un qualsiasi valore c di A)?
- Il valore può essere approssimato come segue:
 - Sia $|R|$ il numero di tuple in R e $|R.A|$ il numero di valori diversi dell'attributo A in R
 - Possiamo assumere che *in media*

$$|R_{A=c}| = \left\lceil \frac{|R|}{|R.A|} \right\rceil$$

- Quindi il costo della DELETE sarà approssimato con la formula $P_R + |R_{A=c}|$

DELETE: esempio

Prendiamo la tabella iniziale:

- $|R.city| = 3$ (perché l'attributo *city* ha 3 valore distinti

- Quindi $\frac{|R|}{|R.city|} = \frac{6}{3} = 2$

id	name	city	age
1	Marco	Trento	33
2	Andrea	Cosenza	24
3	Giuseppe	Milano	27
4	Federica	Trento	20
5	Rossana	Cosenza	18
6	Laura	Trento	25

[Ovviamente si assume una distribuzione uniforme di valori, che non sempre è garantita!].

Heap file: il *selectivity factor*

- Il rapporto tra $|R|$ e $|R.A|$ può essere usato per calcolare il cosiddetto *selectivity factor* (f) per l'attributo A , ovvero $\frac{1}{|R.A|}$
- Quindi il costo per il DELETE può essere riscritto come segue:

id	name	city	age
1	Marco	Trento	33
2	Andrea	Cosenza	24
3	Giuseppe	Milano	27
4	Federica	Trento	20
5	Rossana	Cosenza	18
6	Laura	Trento	25

$$\text{Costo}_{\text{Delete}} = P_R + |R_{A=c}| = P_R + [f \cdot |R|]$$

Sorted file

$(5, \dots, \text{'Lorenza'})$, $(2, \dots, \text{'Lorenza'})$	$(3, \dots, \text{'Milano'})$, $(1, \dots, \text{'Trento'})$	$(6, \dots, \text{'Trento'})$, $(4, \dots, \text{'Trento'})$
---	---	--

- Per abbassare il costo di alcune delle operazioni viste sopra, possiamo immaginare di ordinare le tuple rispetto a qualche attributo (o insieme di attributi)
- In questo caso, le tuple vengono memorizzate in pagine in base al valore dell'attributo (o degli attributi)

Sorted file: costo delle principali operazioni

SCAN

- Immaginiamo di voler leggere tutte le tuple di una relazione R in un *sorted file* con una query del tipo:

SELECT * FROM R ;

- Il costo è identico a quello della stessa operazione su *heap file*:

$$\text{Costo}_{\text{Scan}} = P_R$$

[ovvero il costo di leggere tutte le pagine che contengono le tuple di R]

Sorted file - Costo delle principali operazioni

EQUALITY SEARCH

- Costo della query:

```
SELECT * FROM R WHERE city = 'Trento';
```

- Possiamo usare la ricerca binaria:

- Leggiamo la pagina in mezzo al file

- Se il valore 'Trento' segue alfabeticamente il valore di *city* in quella pagina, escludiamo la pagina letta e tutte quelle che la precedono
 - Se il valore 'Trento' precede alfabeticamente il valore di *city* in quella pagina, escludiamo la pagina letta e tutte quelle che la seguono

- Ripetiamo l'operazione sulle pagine rimanenti fino a trovare la/e pagina/e che contengono il valore 'Trento'

Sorted file - Costo delle principali operazioni

EQUALITY SEARCH

- Se P_R è il numero di pagine che contengono le tuple di R , allora:

$$\text{CostoEqSearch} = \lceil \log_2(P_R) \rceil + \left\lceil \frac{|R_{\text{city}='Trento'}|}{\left\lfloor \frac{P}{t_R} \right\rfloor} \right\rceil$$

- La prima parte è il numero di pagine che devo leggere per trovare la pagina dove iniziano le tuple con 'Trento' come valore di *city*
- La seconda parte calcola il numero di pagine che devo aprire per trovare tutte le tuple che hanno 'Trento' come valore per *city*.

Sorted file - Costo delle principali operazioni

RICERCA PER INTERVALLO


- Qual è il costo di una ricerca per intervallo, per esempio della query:

```
SELECT * FROM R WHERE age >= 18 AND <=30;
```
- Anche in questo caso, possiamo usare la ricerca binaria e poi leggere tutte le pagine che contengono valori compresi tra 18 e 30 per l'attributo *age*
- Quindi il costo non cambia, una volta che calcoliamo quante tuple contengono i valori cercati e quindi quante pagine possiamo ipotizzare di dover leggere.

Sorted file: costo delle principali operazioni

DELETE

- Per cancellare tuple, il ragionamento è molto simile alla ricerca:
 - Si esegue la ricerca binaria per trovare le tuple da cancellare (in tempo $\log_2(P_R)$)
 - Tuttavia va aggiunto il costo della scrittura delle pagine modificate per la cancellazione delle tuple:


$$\text{CostoDelete} = \lceil \log_2(P_R) \rceil + 2 \cdot \left\lceil \frac{|R_{\text{city}='Trento'}|}{\left\lfloor \frac{P}{t_R} \right\rfloor} \right\rceil$$

- Nel calcolo trascuriamo il costo del riempimento dello spazio liberato dalle tuple eliminate

Sorted file - Costo delle principali operazioni

INSERT

- Immaginiamo di voler inserire una o più nuove tuple in R in un *sorted file* (es. una nuova persona che vive a Verona)
- Dobbiamo trovare il punto in cui inserirle nel corretto ordine mediante ricerca binaria e poi riscrivere la pagina modificata per l'aggiunta della nuova tupla:

$$\text{Costo}_{\text{Insert}} = \lceil \log_2(P_R) \rceil + 1$$

Sorted file - Costo delle principali operazioni INSERT

- Tuttavia, dobbiamo anche considerare il caso in cui nella pagina non ci sia spazio per aggiungere la nuova tupla
- In questo caso, l'operazione di INSERT potrebbe richiedere di leggere P_R pagine e scriverne altrettante (caso peggiore)
- Anche senza questo problema, è comunque chiaro che l'INSERT in un *sorted file* è più costoso della stessa operazione in un *heap file*