TESTO ESERCIZI

A. Si consideri il seguente schema relazionale

Clienti(c_id, nome, cognome, email)
Ordini(o_id, cliente[Clienti], data, totale)

1) Scrivere un espressione algebrica per restituire tutti gli id degli ordini eseguiti da un cliente con cognome Rossi e in data 15/05/2023
2) Disegnare il query tree (albero dell'interrogazione) dell'espressione ottenuta nel punto 1
3) Ottimizzare il query tree, se necessario, ottenuto nel punto 2
4) Proporre l'algoritmo per il join ipotizzando la seguente configurazione:
    a. La tabella clienti occupa 1000 blocchi
    b. La tabella Ordini occupa 100 blocchi
    c. Presenza di un indice sul c_id in Clienti

B. Si consideri il seguente schema relazionale

Clienti(c_id, nome, cognome, email)
Ordini(o_id, cliente[Clienti], data, totale)

Con la seguente configurazione:

La tabella Clienti ha 1.500 record di lunghezza 110 byte
La tabella Ordini ha 100.000 record di lunghezza 20 byte
Il blocco ha dimensione 4000 byte
Un puntatore al blocco occupa 7 byte
Il campo email è unique e occupa 20 byte
Il file dati di Ordini è ordinato fisicamente su o_id
il file Clienti non è ordinato ma esiste un indice secondario sul campo email

Indicare il costo delle seguenti query

B.1 - Select * from Ordini WHERE o_id BETWEEN 2000 and 3000, dove il predicato BETWEEN 2000 and 3000 ha una selettività del 0.01

B.2 - Select * from Ordini WHERE data='15/05/2023', dove il predicato data='15/05/2023' ha una selettività del 0.001

B.3 - Select * from Clienti WHERE email='barbara.carminati@uninsubria.it'

SOLUZIONI

A. Si consideri il seguente schema relazionale

Clienti(c_id, nome, cognome, email)
Ordini(o_id, cliente^{Clienti}, data, totale)

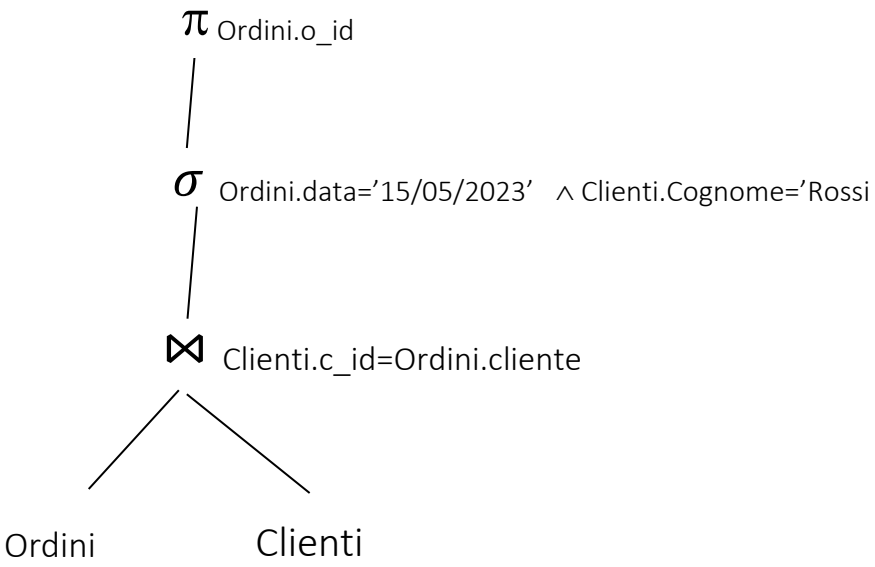
1) Scrivere un espressione algebrica per restituire tutti gli id degli ordini eseguiti da un cliente con cognome Rossi e in data 15/05/2023
2) Disegnare il query tree (albero dell'interrogazione) dell'espressione ottenuta nel punto 1
3) Ottimizzare il query tree, se necessario, ottenuto nel punto 2
4) Proporre l'algoritmo per il join ipotizzando la seguente configurazione:
    a. La tabella clienti occupa 1000 blocchi
    b. La tabella Ordini occupa 100 blocchi
    c. Presenza di un indice sul c_id in Clienti

SOLUZIONE A

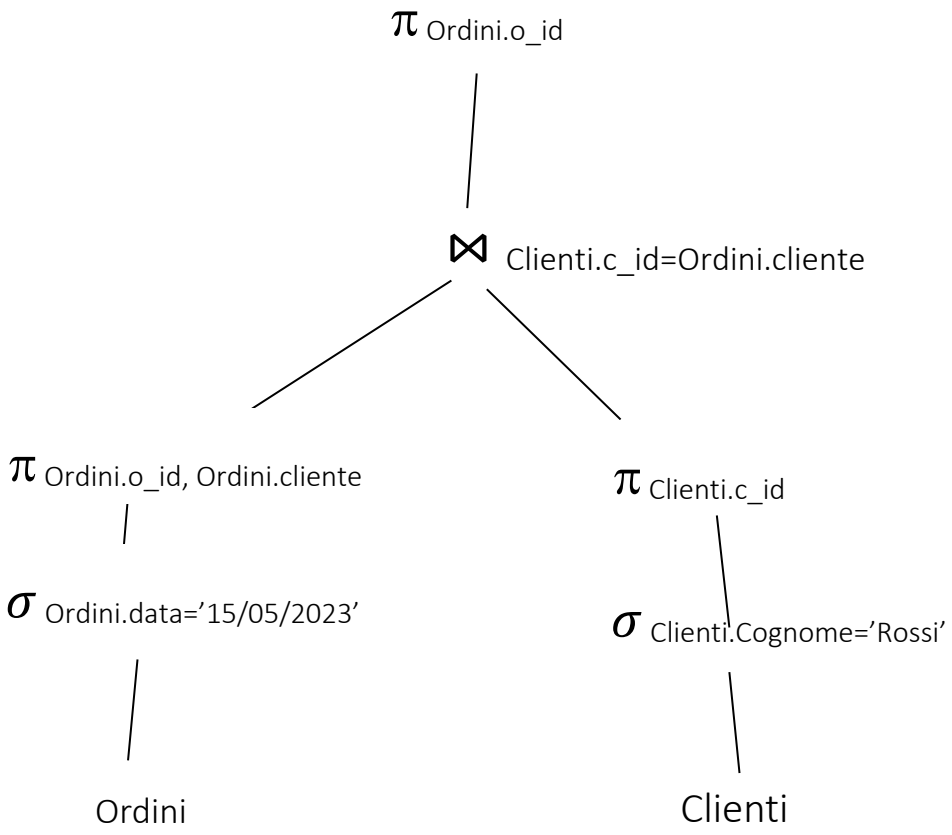
A.1 Una possibile espressione algebrica è la seguente:

$\pi_{\text{Ordini.o\_id}} (\sigma_{\text{Ordini.data='15/05/2023'} \wedge \text{Clienti.Cognome='Rossi'}} (\text{Clienti} \bowtie_{\text{Clienti.c\_id=Ordini.cliente}} \text{Ordini}))$

A.2 la query tree dell'espressione al punto 1 è:



A.3 una possibile ottimizzazione del query tree al punto 2 è la seguente



Si scompone l'operatore σ in due operatori con condizione separate su Ordini e Cliente.
Sfruttando la proprietà commutativa del σ, si sposta ogni operatore σ quanto più possibile verso il basso del query tree.
Sfruttando la proprietà commutativa del Π, si sposta l'operatore Π quanto più possibile verso il basso dell'albero di interrogazione, mantenendo solo gli attributi necessari per eseguire le operazioni successive.

A.4 Si sfrutta l'indice su c_id della tabella Clienti, e si propone un algoritmo di nested loop con indici per il join Ordini ⋈ Clienti

B. Si consideri il seguente schema relazionale

Clienti(c_id, nome, cognome, email)
Ordini(o_id, cliente^{Clienti}, data, totale)

Con la seguente configurazione:

- La tabella Clienti ha 1.500 record di lunghezza 110 byte
- La tabella Ordini ha 100.000 record di lunghezza 20 byte
- Il blocco ha dimensione 4000 byte
- Un puntatore al blocco occupa 7 byte
- Il campo email è unique e occupa 20 byte
- Il file dati di Ordini è ordinato fisicamente su o_id
- il file Clienti non è ordinato ma esiste un indice secondario sul campo email

Indicare il costo delle seguenti query

B.1 - Select * from Ordini WHERE o_id BETWEEN 2000 and 3000, dove il predicato BETWEEN 2000 and 3000 ha una selettività del 0.01

B.2 - Select * from Ordini WHERE data='15/05/2023', dove il predicato data='15/05/2023' ha una selettività del 0.001

B.3 - Select * from Clienti WHERE email='barbara.carminati@uninsubria.it'

SOLUZIONE

B.1. Il file è ordinato sull'attributo del predicato. E' possibile fare una ricerca binaria per trovare il primo record (o_id=2000), e leggere poi i blocchi successivi per recuperare tutti i record che soddisfano la condizione.
La stima del costo è data due fattori:

- (1)Costo per la ricerca binaria su file ordinato (i.e., per cercare o_id=2000)
- (2)Costo lettura blocchi di dati contenti i record con o_id=> 2000 e o_id <=3000

(1) Costo della ricerca binaria:
Per stimare il costo della ricerca binario, devo prima calcolare quanti blocchi occupa il file dati di Ordini.
La tabella ordini ha un fattore di blocco Bfr=$\lceil 4000/20 \rceil$=200
100.000 record di Ordini sono memorizzati in $\lceil 100.000/200 \rceil = 500$ blocchi
Una ricerca binaria su file ordinato di 500 blocchi costa 9 operazioni I/O

(2) Costo della lettura dei record nell'intervallo (2000, 3000)
Il predicato ha una selettività del 0.01. Questo comporta che la cardinalità dell'insieme che soddisfa la condizione è $Card_{\sigma}$=100.000 *0.01= 1.000
Con Bfr=200, 1000 record sono contenuti in 5 blocchi.
Costo della lettura è 5 operazioni I/0

La stima totale del costo della query è 9+5-1= 13 operazioni di I/O

NB. Si è tolto una lettura (-1) in quanto la ricerca binaria porta già al primo blocco contente i primi record che soddisfano il predicato.

B.2 Il file dati di Ordini è ordinato fisicamente su o_id, ma il predicato è sull'attributo data, quindi non è possibile fare una ricerca binaria. E' necessario fare una ricerca lineare.
Dato che ci sono più record che soddisfano il predicato (selettività=0.001), non è possibile fermarsi al primo record che soddisfa il predicato, ma è richiesto leggere tutti i blocchi del file Ordini per ricercare tutti i record.

Stima del costo della query 500 operazioni di I/O

B.3 Il file dati di Clienti non è ordinato fisicamente ma esiste un indice secondario su email.

La stima del costo è data due fattori:

- (1) Ricerca binaria sull'indice secondario per trovare la voce con chiave email='barbara.carminati@uninsubria.it'
- (2) Lettura del blocco riferito dal puntatore della voce trovata nell'indice.

(1) Ricerca sull'indice secondario sul campo email
Per calcolare il costo della ricerca sull'indice secondario, dobbiamo prima calcolare il numero di blocchi dell'indice.
Il file non è ordinato sul email, quindi l'indice è denso. E' necessaria una voce per ogni record, quindi 1500 voci.
Ogni voce richiede 27 byte, 20 per il campo email e 7 per il puntatore al blocco.
In un blocco da 4000 byte si possono memorizzare $\lceil 4000/27 \rceil$=149 voci
L'indice secondario occupa, quindi, $\lceil 1500/149 \rceil$=11 blocchi
La ricerca sull'indice secondario è una ricerca binaria su 11 blocchi: richiede 4 operazioni I/O

Il costo della query è quindi dato dal costo della ricerca su indice secondario (4) + un'operazione di lettura del blocco riferito dal puntato trovato nell'indice.

Costo totale 5 operazioni I/O