# Basi di Dati: Strutture ed Algoritmi
# Soluzione degli Appelli del 2002

A cura di: Giacomo Del Rio

## Appello del 10.1.2002

1. Si consideri la relazione:

   Impiegati(Codice, Nome, AnnoNascita, Dipartimento)

   Per le seguenti interrogazioni si dia un piano di accesso e si stimi il costo considerando due casi: il primo con l'interrogazione nel modo i cui è scritta e il secondo riscrivendo l'interrogazione in modo da trovare un piano migliore:

   a) Esiste un indice a $B^+$–albero su AnnoNascita:

   ```
   SELECT    Dipartimento
   FROM      Impiegati
   WHERE     AnnoNascita = 1960 OR AnnoNascita = 1970 ;
   ```

   b) Esiste un indice a $B^+$–albero su AnnoNascita:

   ```
   SELECT    Dipartimento
   FROM      Impiegati
   WHERE     AnnoNascita < 1970 AND AnnoNascita > 1968 ;
   ```

   c) Esiste un indice a $B^+$–albero su AnnoNascita:

   ```
   SELECT    Dipartimento
   FROM      Impiegati
   WHERE     2*AnnoNascita = 3920;
   ```

   d) Non esistono indici:

   ```
   SELECT    Dipartimento, AVG(AnnoNascita)
   FROM      Impiegati
   GROUP BY  Dipartimento
   HAVING    Dipartimento = 20;
   ```

   e) Si consideri anche la tabella

   Dipartimenti(Numero, NomeD, Indirizzo, Telefono, Direttore)

in cui Direttore è una chiave esterna per Impiegati, con chiave primaria Codice, e possono esistere dipartimenti privi di direttori:

```
SELECT    Codice
FROM      Impiegati, Dipartimenti
WHERE     Codice = Direttore;
```

2. Si considerino le transazioni:

   $T_1 = \{r_1[A], w_1[A], r_1[B], w_1[B]\}$,

   $T_2 = \{r_2[B], r_2[A], w_2[A], w_2[B]\}$

   Si supponga che il serializzatore aggiunga le richieste di blocco esclusivo come segue:

   $T_1 = \{wl_1(A), r_1[A], w_1[A], wl_1(B), r_1[B], w_1[B], commit, wu_1(A), wu_1(B)\}$

   $T_2 = \{wl_2(B), r_2[B], wl_2(A), r_2[A], w_2[A], w_2[B], commit, wu_2(A), wu_2(B)\}$

   Supponendo che le transazioni vengano eseguite concorrentemente, si risponda alle seguenti domande (giustificando le risposte):

   a) La storia prodotta è c-serializzabile?

   b) Può verificarsi uno stallo? Supponendo che parta prima $T_1$, quale transazione riparte se si adotta il protocollo di prevenzione dello stallo *wait-die*?

   c) Può verificarsi una cascata di abort?

   Si supponga che il serializzatore aggiunga le richieste di blocco esclusivo in questo altro modo:

   $T_1 = \{wl_1(A), r_1[A], w_1[A], wl_1(B), wu_1(A), r_1[B], w_1[B], wu_1(B)\}$

   $T_2 = \{wl_2(A), wl_2(B), r_2[B], r_2[A], w_2[A], wu_2(A), w_2[B], wu_2[B]\}$

   d) La storia prodotta è c-serializzabile?

   e) Può verificarsi uno stallo? Supponendo che parta prima $T_1$, quale transazione riparte se si adotta il protocollo di prevenzione dello stallo *wound-wait*?

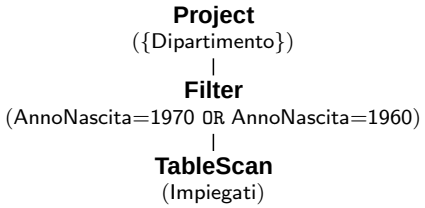
   f) Può verificarsi una cascata di abort?

3. Spiegare come viene eseguito dal gestore del buffer il comando di lettura di una pagina. Quando il gestore del buffer scrive una pagina nella memoria permanente?

**Soluzione Appello del 10.1.2002**

**Esercizio 1**

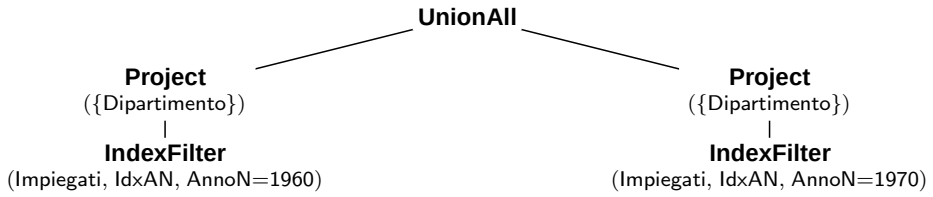
a) *Normale*:




$$C_{TS} = N_{pag}(Impiegati)$$

*Riscritta*:

```
SELECT    Dipartimento
FROM      Impiegati
WHERE     AnnoNascita = 1960
          UNION ALL
SELECT    Dipartimento
FROM      Impiegati
WHERE     AnnoNascita = 1970
```

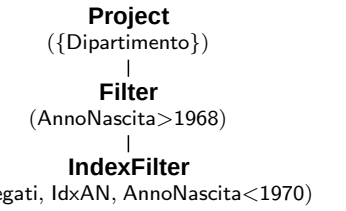



$$C_{IF} = C_I + C_D = f_s(AnnoNascita = 1970) \times N_{leaf}(IdxAN) + \Phi\left(\frac{N_{reg}(Impiegati)}{N_{key}(AnnoNascita)}, N_{pag}(Impiegati)\right)$$

$$C_{UA} = 2 \times C_{IF}$$

In questo caso è possibile sfruttare l'operatore UnionAll in quanto i due insiemi di record ritornati dagli operandi sono disgiunti.

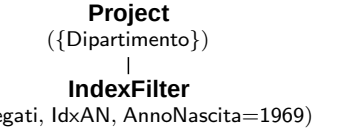
b) *Normale*:




$$C_{IF} = C_I + C_D = f_s(AnnoNascita < 1970) \times N_{leaf}(IdxAN) + f_s(AnnoNascita < 1970) \times N_{key}(IdxAN) \times \Phi\left(\frac{N_{reg}(Impiegati)}{N_{key}(AnnoNascita)}, N_{pag}(Impiegati)\right)$$

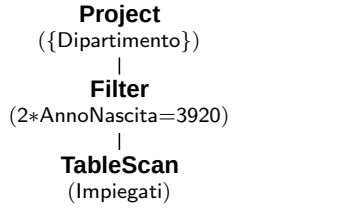
*Riscritta*:

```
SELECT    Dipartimento
FROM      Impiegati
WHERE     AnnoNascita = 1969
```




$$C_{IF} = C_I + C_D = f_s(AnnoNascita = 1969) \times N_{leaf}(IdxAN) + f_s(AnnoNascita = 1969) \times \Phi\left(\frac{N_{reg}(Impiegati)}{N_{key}(AnnoNascita)}, N_{pag}(Impiegati)\right)$$

c) *Normale*:




$$C_{TS} = N_{pag}(Impiegati)$$

*Riscritta*:

```
SELECT    Dipartimento
FROM      Impiegati
WHERE     AnnoNascita = 1960
```

**Project**
({Dipartimento})
|
**IndexFilter**
(Impiegati, IdxAN, AnnoNascita=1960)

$$C_{IF} = C_I + C_D = f_s(AnnoNascita = 1969) \times N_{leaf}(IdxAN) + \\ f_s(AnnoNascita = 1969) \times \Phi\left(\frac{N_{reg}(Impiegati)}{N_{key}(AnnoNascita)}, N_{pag}(Impiegati)\right)$$

Non tutti gli ottimizzatori sono in grado di riscrivere la condizione in modo da poter utilizzare un indice.

d) *Normale*:

**Filter**
(Dipartimento=20)
|
**GroupBy**
({Dipartimento},{AVG(AnnoNascita)})
|
**SortScan**
(Impiegati, {Dipartimento})

$$C_{SS} = 5 \times N_{pag}(Impiegati)$$

*Riscritta*:

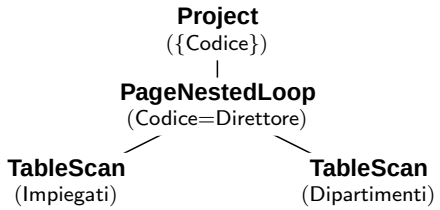
```
SELECT    Dipartimento, AVG(AnnoNascita)
FROM      Impiegati
WHERE     Dipartimento = 20
GROUP BY  Dipartimento;
```

**GroupBy**
({Dipartimento},{AVG(AnnoNascita)})
|
**Filter**
(Dipartimento=20)
|
**TableScan**
(Impiegati)

$$C_{TS} = N_{pag}(Impiegati)$$

La coppia Group By-Having dell'interrogazione originale si limitava a considerare il gruppo dei record con codice dipartimento uguale a 20; questo può essere fatto in maniera più efficiente con una restrizione. Inoltre, poichè filtrare su una condizione di uguaglianza equivale anche ad ordinare sull'attributo della condizione, non è necessario ordinare i dati sui quali opera il GroupBy.

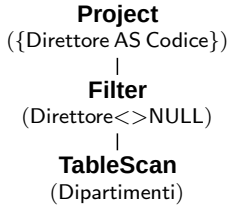
e) *Normale*:




$$C_{PNL} = N_{pag}(Impiegati) + N_{pag}(Impiegati) \times N_{pag}(Dipartimenti)$$

*Riscritta*:

```
SELECT    Direttore AS Codice
FROM      Dipartimenti
WHERE     Direttore <> NULL;
```




$$C_{TS} = N_{pag}(Dipartimenti)$$

L'interrogazione cercava i codici dei direttori di dipartimenti e questa informazione si può ottenere direttamente dai dipartimenti.

**Esercizio 2**

a) Se osserviamo come sono aggiunte le richieste di blocco nelle due transazioni scopriamo che la storia ottenuta è legale secondo il protocollo 2PL-stretto; per questo motivo, sapendo che il protocollo 2PL-stretto produce storie c-serializzabili, anche la storia ottenuta sarà c-serializzabile

b) Il protocollo 2PL-stretto è prono allo stallo, dunque è possibile che un tale evento si verifichi. Se si utilizza il protocollo di prevenzione dello stallo *wait-die*, in caso di richieste di blocco non soddisfacibili, una transazione rimane in attesa solo se il dato bloccato appartiene ad una transazione più giovane, in caso contrario la transazione viene interrotta e fatta ripartire.

Vediamo un esempio con le transazioni $T_1$ e $T_2$:

| | Comandi | | Blocchi Concessi | |
|---|---|---|---|---|
| | $T_1$ | $T_2$ | $T_1$ | $T_2$ |
| $t_0$ | $wl(A)$ | | $W_A$ | |
| $t_1$ | $r[A]$ | | $W_A$ | |
| $t_2$ | \| | $wl(B)$ | $W_A$ | $W_B$ |
| $t_3$ | \| | $r[B]$ | $W_A$ | $W_B$ |
| $t_4$ | $w[A]$ | \| | $W_A$ | $W_B$ |
| $t_5$ | \| | $wl(A)$ | $W_A$ | $W_B$ |
| $t_6$ | \| | $abort$ | $W_A$ | |
| $t_7$ | $wl(B)$ | | $W_A$ $W_B$ | |
| $t_9$ | $r[B]$ | | $W_A$ $W_B$ | |
| $t_{10}$ | $w[B]$ | | $W_A$ $W_B$ | |
| $t_{11}$ | $commit$ | | $W_A$ $W_B$ | |
| $t_{12}$ | $wu(A)$ | | $W_B$ | |
| $t_{13}$ | $wu(B)$ | | | |
| $t_{14}$ | | $wl(B)$ | | $W_B$ |
| $t_{15}$ | | ... | | ... |

La prima transazione a partire è $T_1$ dunque si avrà che, in caso di richieste di blocco non soddisfacibili, $T_1$, la più vecchia, potrà aspettare, mentre $T_2$ dovrà essere interrotta. Al tempo $t_5$ abbiamo la prima richiesta di blocco non soddisfacibile, infatti il blocco in scrittura su $A$ richiesto da $T_2$ è posseduto da $T_1$: il protocollo *wait-die* entra in funzione e fa abortire $T_2$. A questo punto $T_1$ può tranquillamente procedere con la sua esecuzione e portare a termine le operazioni. È interessante notare che, se non ci fosse stato l'abort di $T_2$, al tempo $t_7$ si sarebbe creata una condizione di stallo.

Nel nostro caso la transazione $T_2$ viene fatta ripartire dopo il commit di $T_1$, ma questo non è un comportamento obbligatorio: $T_2$ sarebbe potuta ripartire in ogni momento dopo il suo abort. Tuttavia, se lo avesse fatto in un tempo anteriore a $t_{12}$ (cioè prima che $T_1$ rilasciasse il blocco su $A$), ancora una volta sarebbe stata costretta ad abortire, in quanto la sua marca temporale, identica a quella della prima partenza, sarebbe stata più grande di quella di $T_1$ (cioè sarebbe stata ancora la transazione più giovane delle due). Questo fatto ci garantisce che sia sempre la transazione ad aver svolto il minor carico di lavoro a doversi interrompere.

c) No, perchè sappiamo che il protocollo 2PL-stretto è ripristinabile e senza cascata di abort (*cascadeless*).

d) Questa volta siamo di fronte a storie legali per il protocollo 2PL classico, che sappiamo produrre storie c-serializzabili.

e) Come per la versione stretta è ancora possibile avere situazioni di stallo. Se si utilizza il protocollo di prevenzione dello stallo *wound-wait*, in caso di una richiesta di blocco non soddisfacibile, una transazione rimane in attesa solo se il dato bloccato appartiene ad una transazione più vecchia, in caso contrario la transazione più giovane viene interrotta e fatta ripartire.

Vediamo un esempio con le transazioni $T_1$ e $T_2$: *(l'asterisco indica transazioni in attesa)*

| | Comandi | | Blocchi Concessi | |
|---|---|---|---|---|
| | $T_1$ | $T_2$ | $T_1$ | $T_2$ |
| $t_0$ | $wl(A)$ | | $W_A$ | |
| $t_1$ | $r[A]$ | | $W_A$ | |
| $t_2$ | $w[A]$ | | $W_A$ | |
| $t_3$ | \| | $wl(A)$ | $W_A$ | |
| $t_4$ | $wl(B)$ | $*$ | $W_A$ $W_B$ | |
| $t_5$ | $wu(A)$ | $*$ | $W_B$ | $W_A$ |
| $t_6$ | \| | $wl(B)$ | $W_B$ | $W_A$ |
| $t_7$ | $r[B]$ | $*$ | $W_B$ | $W_A$ |
| $t_9$ | $w[B]$ | $*$ | $W_B$ | $W_A$ |
| $t_{10}$ | $wu(B)$ | $*$ | | $W_A$ $W_B$ |
| $t_{11}$ | | $r[B]$ | | $W_A$ $W_B$ |
| $t_{12}$ | | $r[A]$ | | $W_A$ $W_B$ |
| $t_{13}$ | | $w[A]$ | | $W_A$ $W_B$ |
| $t_{14}$ | | $wu(A)$ | | $W_B$ |
| $t_{15}$ | | $w[B]$ | | $W_B$ |
| $t_{16}$ | | $wu(B)$ | | |

La prima transazione a partire è $T_1$ dunque si avrà che, in caso di richieste di blocco non soddisfacibili, $T_2$, la più giovane, potrà aspettare, mentre, se il fatto dovesse capitare a $T_1$ sarebbe $T_2$ ad essere interrotta. Al tempo $t_3$ abbiamo la prima richiesta di blocco non soddisfacibile, infatti il blocco in scrittura su $A$ richiesto da $T_2$ è posseduto da $T_1$: il protocollo *wait-die* entra in funzione e mette in attesa $T_2$. $T_1$ da parte sua va avanti con la sua esecuzione e al tempo $t_5$ rilascia il blocco su $A$ che viene passato a $T_2$. Di nuovo $T_2$ fa una richiesta non soddisfacibile e viene posta nuovamente in attesa fino al tempo $t_{11}$. Come per il protocollo precedente, la priorità di esecuzione viene data alla transazione più vecchia, che si suppone abbia svolto una maggiore quantità di lavoro.

f) Si, il protocollo 2PL ammette cascate di abort.

**Esercizio 3**

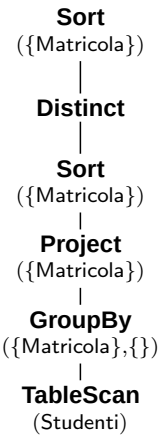
Vedi soluzione esercizio numero 3 del 10.7.2001.

## Appello del 14.2.2002

1. Si consideri la relazione Studenti(Matricola, Nome, AnnoNascita), ordinata sulla chiave primaria Matricola, e l'interrogazione:

```
SELECT      DISTINCT Matricola, COUNT(*)
FROM        Studenti
WHERE       AnnoNascita = 1974
GROUP BY    Matricola
ORDER BY    Matricola;
```

   Si dica se il seguente piano d'accesso produce il risultato cercato. Se non va bene, lo si modifichi aggiungendo le parti mancanti:




   Si semplifichi il piano di accesso eliminando gli operatori inutili e si dia una stima del suo costo. Come cambia il piano di accesso e il suo costo se la relazione non è ordinata su Matricola?

2. Siano $S_1$ e $S_2$ due storie sullo stesso insieme di transazioni. Si supponga che $S_1$ sia c-serializzabile e che i grafi di serializzabilità delle due storie $GS(S_1)$ e $GS(S_2)$ siano uguali. Si considerino le seguenti affermazioni su $S_2$:

   (a) $S_2$ è seriale

   (b) $S_2$ è c-equivalente a $S_1$

   (c) $S_2$ è c-serializzabile.

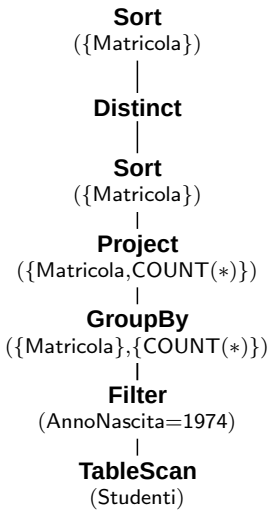
   Giustificare quali delle seguenti conclusioni sono vere:
   (a) sono vere 1, 2 e 3; (b) solo 2 e 3 sono vere; (c) solo 2; (d) solo 3; (d) nessuna.

3. Come viene garantita l’atomicità e la persistenza delle transazioni dal gestore dell’affidabilità ?

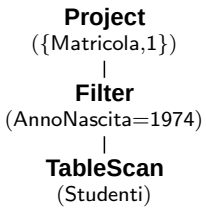
**Soluzione Appello del 14.2.2002**

**Esercizio 1**

Completiamo il piano di accesso aggiungendo le parti mancanti, che sono il filtro su AnnoNascita e la funzione di aggregazione COUNT(*):




Adesso eliminiamo gli operatori superflui:




Partendo dal basso in ordine abbiamo eliminato:

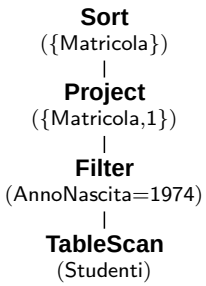
1. GroupBy: l'attributo di raggruppamento è chiave, e quindi non serve raggruppare.
2. Project: l'operatore GroupBy restituisce record aventi solamente i campi passati come argomento, non c'è bisogno di proiettare ancora,
3. Sort: la tabella è memorizzata ordinata sulla chiave Matricola, non è necessario ordinare ancora,

4. Distinct: tra gli attributi presenti nei record che arrivano all'operatore Distinct è compresa una chiave, dunque non possono esistere record uguali,

5. Sort: non serviva prima, non serve adesso

Con queste semplificazioni il costo si riduce a $N_{pag}(Studenti)$.

Se la relazione non fosse ordinata su Matricola, si dovrebbe procedere ad un ordinamento finale:




$$
\begin{aligned}
C_{Sort} &= C_O + 6 \times N_{pag}(T) \\
&= N_{pag}(Studenti) + 6\times \\
&\quad \left( \frac{L_k(Matricola) \times f_s(AnnoNascita = 1974) \times N_{reg}(Studenti)}{D_{pag}} \right)
\end{aligned}
$$

**Esercizio 2**

Valutiamo le affermazioni:

1. *$S_2$ è seriale*
   (Falsa) Non ci sono né prove né indizi che $S_2$ sia seriale.

2. *$S_2$ è c-equivalente a $S_1$*
   (Vera) Se $S_1$ e $S_2$ hanno lo stesso grafo di serializzazione vuol dire che hanno nello stesso ordine le operazioni in conflitto, dunque sono c-equivalenti (Nitto).

3. *$S_2$ è c-serializzabile*
   (Vera) Il grafo di serializzazione di una storia c-serializzabile non ha cicli, $S_2$ ha lo stesso grafo di serializzabilità di $S_1$, $S_1$ è c-serializzabile dunque il grafo di $S_2$ è privo di cicli, $S_2$ è c-serializzabile.

   Oppure: $S_2$ è c-equivalente a $S_1$, la c-equivalenza è transitiva ed S1 è c-equivalente ad una storia seriale (Nitto).

In conclusione la risposta corretta è la (b).

**Esercizio 3**

Vedi soluzione n°3 dell'appello del 1.6.2001.

## Appello del 27.5.2002

1. Si consideri lo schema:

   Fornitori(Fid, FNome, Città)
   Parti(Pid, PNome, Colore, Peso)
   Ordini(Fid*, Pid*, PGid*, Qta)

   e l'interrogazione:

   ```
   SELECT     DISTINCT FNome, sum(Qta)
   FROM       Ordini O, Fornitori F
   WHERE      O.Fid = F.Fid AND O.Pid < 'P5' AND F.Città = 'Pisa'
   GROUP BY   FNome
   ORDER BY   FNome
   ```

   Si fornisca un piano di accesso usando indici a vostra scelta e si dia una stima del suo costo.

2. Si definiscano le nozioni di grafo di serializzabilità e di storia c-serializzabile. Si considerino le storie:

   $S_1 = \{r_1[A], w_1[B], r_2[B], w_2[C], r_3[C], w_3[A]\}$

   $S_2 = \{r_1[A], r_2[A], w_1[B], w_2[B], r_1[B], r_2[B], w_2[C], w_1[D]\}$

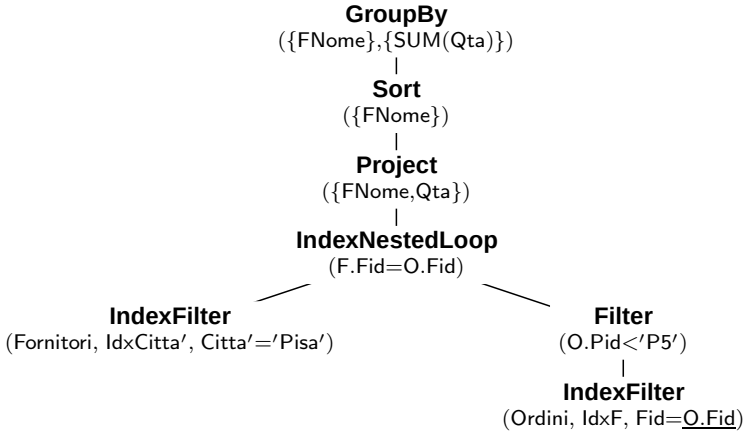
   Si dica se sono c-serializzabili. Se lo sono, si dica quali sono le storie seriali equivalenti.

3. Spiegare come viene eseguito dal gestore del buffer il comando di lettura di una pagina. Quando il gestore del buffer scrive una pagina nella memoria permanente?

**Soluzione Appello del 27.5.2002 (Igor Nitto)**

**Esercizio 1**

Supponiamo l'esistenza di indici a $B^+$-albero sugli attributi Fornitori.Città e Ordini.Fid. Piano d'accesso:




Stima del costo: valutiamo prima costo e cardinalità dell'IndexNestedLoop:

$$C_{INL} = C_A(F) + E_{reg}(F) \times C_A(O)$$
$$E_{reg}(INL) = f_s(F.Fid = O.Fid) \times E_{reg}(F) \times Ereg(O)$$

con

$$f_s(F.Fid = O.Fid) = 1/max\{Nkey(F.Fid), Nkey(O.Fid)\}$$
$$C_A(F) = C_I + C_D$$
$$C_I = (1/N_{key}(F.Citta')) \times N_{leaf}(idxCitta')$$
$$C_D = \Phi(N_{reg}(F)/N_{key}(F.Citta'), N_{pag}(F))$$
$$E_{reg}(F) = N_{reg}(F)/N_{key}(F.Citta')$$
$$C_A(O) = C_I + C_D$$
$$C_I = (1/N_{key}(O.F)) \times N_{leaf}(idxF)$$
$$C_D = \Phi(N_{reg}(O)/N_{key}(O.F), N_{pag}(O))$$
$$Ereg(O) = f_s(O.Pid <' P5') \times N_{reg}(O)$$

A partire dal costo dell'IndexNestedLoop forniamo la stima del costo di tutto il piano d'accesso, che sarà uguale al costo di visitare i record del risultato dalla proiezione della giunzione sui due

attributi FNome e Qta, di ordinarli su FNome e applicare il GroupBy (che ha costo nullo):

$$C = C_{INL} + 6 \times N_{pag}(T)$$

dove T è la relazione che risulta dalla proiezione della giunzione su FNome e Qta:

$$N_{reg}(T) = (E_{reg}(INL) \times (L(FNome) + L(Qta)))/D_{pag}$$
$$N_{pag}(T) = N_{reg}(T)/D_{pag}$$

**Esercizio 2**

1. Il grafo di serializzabilità di una storia è il grafo orientato nel quale i nodi sono le transazioni non abortite ed esiste un arco da $T_i$ a $T_j$ sse $T_i$ contiene un'operazione che precede e va in conflitto con una di $T_j$.

   Una storia è c-serializzabile sse è c-equivalente ad una storia seriale. Due storie su uno stesso insieme di operazioni sono c-equivalenti sse hanno nello stesso ordine le operazioni in conflitto di transazioni terminate regolarmente.

2. $S_1$ è già una storia seriale. $S_2$ ha il grafo di serializzazione in figura:


   di conseguenza la storia non è c-serializzabile.

**Esercizio 3**

Vedere soluzione appello 10.1.2002.

## Appello del 18.6.2002

1. Si consideri lo schema R(K, A), S(KE*, B), e l'interrogazione:

```
SELECT    *
FROM      R, S
WHERE     K = KE;
```

   Si dia il piano di accesso per eseguire l'interrogazione con il metodo MergeJoin senza uso di indici, oppure usando gli indici su K e KE, e si dia una stima del costo dei due piani. Spiegare in quali dei due casi seguenti non conviene usare gli indici:

   (a) $R$ e $S$ hanno un record per pagina,

   (b) $R$ e $S$ hanno molti record per pagina.

2. In figura sono mostrate le operazioni di una transazione con la notazione:

   - START inizio transazione
   - END fine transazione
   - READ(X, t) copia il valore dell'elemento X della BD nella variabile t
   - WRITE(X, t) copia il valore della variabile t nell'elemento Xdella BD che si trova nel buffer
   - FLUSH(X) trasferisce X dal buffer al disco
   - X Buf, valore di X nel buffer
   - X disco, valore di X nel disco

   Si completi la seguente tabella con le operazioni di FLUSH(X) e i valori delle colonne supponendo che la transazione sia gestita con l'algoritmo *Disfare-NonRifare*:

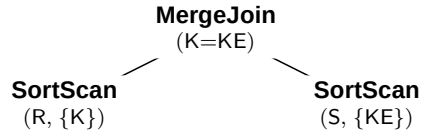
| Azione | t | A Buf | B Buf | A disco | B disco | Log |
|---|---|---|---|---|---|---|
| START | – | – | – | 8 | 8 | (Begin T) |
| READ(A,t) | 8 | 8 | – | 8 | 8 | |
| t = t*2 | 16 | 8 | – | 8 | 8 | |
| WRITE(A,t) | | | | | | |
| READ(B,t) | | | | | | |
| t = t*2 | | | | | | |
| WRITE(B,t) | | | | | | |
| END | | | | | | |

3. Si definiscano le nozioni di grafo di serializzabilità e di grafo di attesa. Si dica come vengono usati nella realizzazione di un DBMS.

**Soluzione Appello del 18.6.2002 (Igor Nitto)**

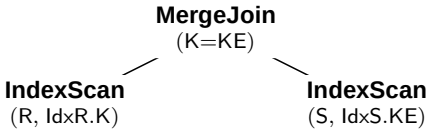
**Esercizio 1**

Piano di accesso senza uso di indici:




$$C_{SM} = 5 \times (N_{pag}(R) + N_{pag}(S))$$

Piano d'accesso con uso di indici:




$$C_{SM} = N_{leaf}(IdxR.K) + N_{reg}(R) + N_{leaf}(IdxS.KE) + N_{reg}(S)$$

Supponiamo che il numero delle foglie di un indice sia minore del numero delle pagine dei dati. Se le relazioni hanno un solo record per pagina, il piano con gli indici ha costo sempre inferiore. Se le relazioni hanno molti record per pagina, il piano con gli indici ha in generale costo superiore. In particolare, se le due relazioni hanno la stessa cardinalità e record piccoli di uguale lunghezza, il piano con gli indici ha costo inferiore se la capacità della pagina dei dati è inferiore a 4.

**Esercizio 3**

Il grafo di serializzabilità associato ad una storia è il grafo orientato nel quale i nodi sono le transazioni terminate regolarmente ed esiste un arco da $T_i$ a $T_j$ sse $T_i$ contiene un'operazione che precede e va in conflitto con una di $T_j$. Il grafo di attesa è il grafo nel quale i nodi sono le transazioni correntemente attive ed esiste un nodo tra $T_i$ e $T_j$ sse $T_i$ è in attesa di un dato bloccato da $T_j$.

Il grafo di serializzabilità è legato ad una teoria generale che consente di dimostrare la correttezza di un serializzatore. Risultato principale della teoria è che una storia è c-serializzabile se e solo se il suo grafo di serializzabilità è aciclico.

Il grafo di attesa è legato invece al controllo degli stalli: per stabilire se un insieme di transazioni attive sono in stallo è sufficiente che nel grafo di attesa vi sia un ciclo.

**Esercizio 2**

| Azione | t | A Buf | B Buf | A disco | B disco | Log |
|---|---|---|---|---|---|---|
| START | – | – | – | 8 | 8 | (begin T) |
| READ(A,t) | 8 | 8 | – | 8 | 8 | |
| t = t*2 | 16 | 8 | – | 8 | 8 | |
| WRITE(A,t) | 16 | 16 | – | 16 | 8 | (Write, A, 8) |
| READ(B,t) | 8 | 16 | 8 | 16 | 8 | |
| t = t*2 | 16 | 16 | 8 | 16 | 8 | |
| WRITE(B,t) | 16 | 16 | 16 | 16 | 16 | (Write, B, 8) |
| FLUSH(A) | 16 | 16 | 16 | 16 | 16 | |
| FLUSH(B) | 16 | 16 | 16 | 16 | 16 | |
| END | | | | | 16 | |

## Appello del 15.7.2002

1. Si consideri la relazione Impiegati(Codice, Nome, AnnoNascita). Per le seguenti interrogazioni si dia un piano di accesso (supponendo di eliminare i duplicati e di eseguire il GroupBy con l'algoritmo di ordinamento) e si giustifichi in quali casi non occorre l'operatore fisico Sort. Si stimi il costo del piano di accesso che ritenete più costoso.

```
SELECT     DISTINCT AnnoNascita
FROM       Impiegati
WHERE      AnnoNascita > 1950;

SELECT     DISTINCT AnnoNascita
FROM       Impiegati
WHERE      AnnoNascita = 1950;

SELECT     AnnoNascita
FROM       Impiegati
WHERE      AnnoNascita = 1950
ORDER BY   AnnoNascita;

SELECT     COUNT(*)
FROM       Impiegati
WHERE      AnnoNascita = 1950
GROUP BY   AnnoNascita;
```

2. Si consideri la storia:

$$S = \{r_1[A], w_1[B], r_2[A], w_2[B], r_3[A], w_3[B]\}$$

   Quante storie si possono definire con le stesse operazioni e c-equivalenti a $S$?

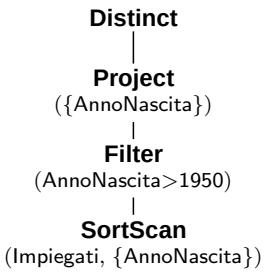
3. Descrivere il protocollo 2PL stretto e la tecnica di gestione dei blocchi.

**Soluzione Appello del 15.7.2002 (Igor Nitto)**

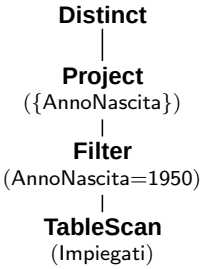
**Esercizio 1**

Piani d'accesso:

a )

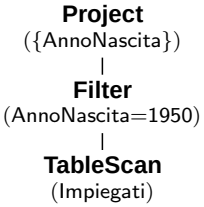



b )

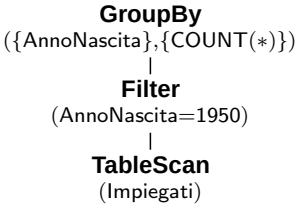



Si noti che non è necessario ordinare i dati su AnnoNascita perché una volta applicato il filtro tutti record avranno lo stesso valore di AnnoNascita.

c )



d )

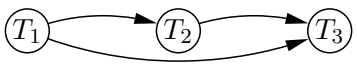



Si stimerà il costo del primo piano d'accesso:

$$C_{a)} = 5 \times N_{pag}(Impiegati)$$

**Esercizio 2 (Del Rio)**

Costruiamo, per iniziare, il grafo di serializzazione di $S$:




Osservando la storia, notiamo come siano solamente le operazioni di scrittura a determinare gli archi del grafo, cioè siamo di fronte solamenti a conflitti di tipo WW. Poichè ogni storia c-equivalente ad $S$ deve avere nello stesso ordine di $S$ le operazioni in conflitto, da un semplice ordinamento topologico del grafo otteniamo che $w_1[B]$ deve precedere $w_2[B]$ che a sua volta deve precedere $w_3[B]$:

$$\boldsymbol{w_1[B]} \quad \boldsymbol{w_2[B]} \quad \boldsymbol{w_3[B]}$$

Dobbiamo adesso piazzare le operazioni di lettura, con l'unica accortezza di rispettare l'ordine relativo delle operazioni di ciascuna transazione. Per quanto riguarda $r_1[A]$ non abbiamo molte scelte, deve necessariamente precedere $w_1[B]$:

$$r_1[A] \quad \boldsymbol{w_1[B]} \quad \boldsymbol{w_2[B]} \quad \boldsymbol{w_3[B]}$$

Per piazzare $r_2[A]$ abbiamo 3 possibilità:

$$\begin{array}{ccccc} r_2[A] & r_1[A] & \boldsymbol{w_1[B]} & \boldsymbol{w_2[B]} & \boldsymbol{w_3[B]} \\ r_1[A] & r_2[A] & \boldsymbol{w_1[B]} & \boldsymbol{w_2[B]} & \boldsymbol{w_3[B]} \\ r_1[A] & \boldsymbol{w_1[B]} & r_2[A] & \boldsymbol{w_2[B]} & \boldsymbol{w_3[B]} \end{array}$$

Per finire, per ciascuna delle 3 possibilità precedenti, $r_3[A]$ può essere piazzato in 5 modi differenti. Otteniamo quindi che in totale esistono $3 \times 5 = 15$ storie c-equivalenti a $S$ ($S$ compresa).

**Esercizio 3**

Il protocollo 2PL-stretto è una variante del 2PL nella quale si impone che i blocchi ottenuti da una transazione vengano tutti quanti rilasciati al momento del commit e che nessun blocco possa essere rilasciato prima.

Alla base della tecnica di gestione dei blocchi c'è una tabella hash che mantiene le associazioni tra dati e descrittori di blocco sui dati. Ogni descrittore di blocco contiene le seguenti informazioni: identificatore del dato bloccato D, il modo del gruppo delle richieste accolte M che riassume i modi di blocco attivi sul dato, la lista delle richieste accolte, la coda delle richieste in attesa. Quando una transazione richiede un blocco su D, allora si distinguono due casi:

1. Se il modo della richiesta è incompatibile con il modo del gruppo delle richieste accolte allora la richiesta viene messa nella coda delle richieste in attesa.

2. Se il modo della richiesta è compatibile con il modo del gruppo delle richieste accolte e la coda delle richieste in attesa è vuota allora la richiesta viene accolta e inserita nella lista delle richieste accolte, se invece la coda delle richieste in attesa non è vuota allora la nuova richiesta va in attesa e viene inserita nella coda delle richieste in attesa.

Quando una transazione rilascia un blocco allora si aggiornano la lista delle richieste accolte e il modo del gruppo, poi si visita la coda delle richieste accolte a partire dalla prima richiesta entrata spostando le richieste che possono essere accolte nella lista delle richieste accolte, la visita procede fino a che non si trova la prima richiesta che non può essere accolta.

## Appello dell'11.9.2002

1. Si definiscano le nozioni di grafo di serializzabilità e di grafo di attesa. Si dica come vengono usati nella realizzazione di un DBMS.

2. Si consideri la base di dati

   Impiegati(CodiceI, Nome, Stipendio, Età, Dipartimento*)
   Dipartimenti(CodiceD, Budget, Università, Direttore*)

   Stipendio ha valori fra 10M e 100M, Età ha valori fra 20 e 80, Budget ha valori fra 10 e 100M, ogni dipartimento ha in media 30 impiegati e ci sono 30 Università. Si supponga una distribuzione uniforme dei valori.

   Per ognuna delle seguenti interrogazioni, quali degli indici proposti definereste per agevolare l'esecuzione dell'interrogazione? Dare il piano di accesso e una stima del suo costo. Se il DBMS non considera la possibilità di usare solo indici per rispondere all'interrogazione (ovvero, proceda sempre con accessi agli indici e poi ai dati, pur avendo tutte le informazioni necessarie per la risposta nell'indice), come cambierebbe la vostra risposta?

   a) Trovare Nome, Età e Stipendio di tutti gli impiegati
      a1) nessun indice
      a2) indice a $B^+$–albero su (Nome, Età, Stipendio)
      a3) indice a $B^+$–albero su (Nome, Età)
   b) Trovare i codici dei dipartimenti dell'Università di Pisa con un budget inferiore a 30
      b1) nessun indice
      b2) indice di ordinamento a $B^+$–albero su (Università, Budget)
      b3) indice a $B^+$–albero su (Università, Budget)
      b4) indice di ordinamento a $B^+$–albero su Budget
   c) Trovare i nomi degli impiegati che dirigono dipartimenti ed hanno uno stipendio maggiore di 80M
      c1) nessun indice
      c2) indice a $B^+$–albero su CodiceD
      c3) indice a $B^+$–albero su CodiceI
      c4) indice di ordinamento a $B^+$–albero su Stipendio

3. Dire quali delle seguenti affermazioni è vera o falsa e giustificate la risposta:

   (a) Il costo di una ricerca per intervallo della chiave K con l'uso di un indice su K è sempre inferiore alla scansione del file.

(b) Con il metodo rifare, tutte le modifiche di una T devono essere riportate nella BD prima che il record di commit sia scritto nel giornale.

(c) Tutte le storie prodotte dal 2PL-stretto sono seriali.

**Soluzione Appello dell'11.9.2002 (Igor Nitto)**

**Esercizio 1**

Vedi esercizio 3 dell'appello del 18.6.2002.

**Esercizio 2**

1. a2) indice denso a $B^+$–albero su (Nome, Età, Stipendio)

**IndexOnlyFilter**
(Impiegati, idxNES, {Nome, Eta′, Stipendio}, True)

$$C = N_{leaf}(idxNES)$$

Se il DBMS non prevede l'IndexOnlyFilter allora la risposta è

a1) nessun indice:

**Project**
({Nome, Eta′, Stipendio})
|
**TableScan**
(Impiegati)

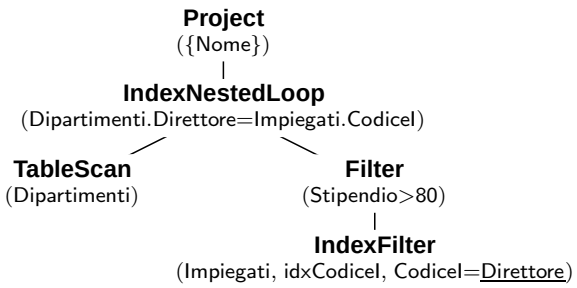
$$C = N_{pag}(Impiegati)$$

2. b2) indice su Università, Budget:

**Project**
({CodiceD})
|
**IndexFilter**
(Dipartimenti, idxUB, Universita′='PI′ AND Budget<30)

$$C = f_s(Universita' =' PI' \wedge Budget < 30)(N_{leaf}(idxUB) + N_{pag}(Dipartimenti))$$
$$C = 1/30 \times 20/90 \times (N_{leaf}(idxUB) + N_{pag}(Dipartimenti))$$

3. c3) indice su CodiceI:




$$C = C_{O_E} + E_{reg}(O_E) \times C_{O_I} = N_{pag}(Dipartimenti) + N_{reg}(Dipartimenti) \times 2$$

**Esercizio 3**

1. Falso. Il costo di una ricerca per intervallo non è sempre inferiore al costo della scansione del file. Ad esempio se l'indice non è di ordinamento e K è chiave primaria il costo della ricerca con indice si può stimare come $f_s(C) \times N_{leaf} + f_s(C) \times N_{reg}(R)$ che non è assolutamente detto essere più piccolo di $N_{pag}(R)$.

2. Falso. Con il metodo rifare non è necessario che le transazioni terminate abbiano già apportato modifiche alla BD, cioè non vale la regola del commit anticipato. Se si verifica un fallimento di sistema, alla ripartenza le operazioni delle transazioni terminate saranno ripetute con la procedura rifare.

3. Falso. Il serializzatore garantisce solamente che le storie prodotte siano c-serializzabili ma non seriali, infatti operazioni di transazioni differenti non in conflitto possono benissimo intercalarsi.