

Gestione dei dati

Parte 9 Esercitazione di riepilogo

Maurizio Lenzerini, Riccardo Rosati

Facoltà di Ingegneria
Sapienza Università di Roma
Anno Accademico 2011/2012

<http://www.dis.uniroma1.it/~rosati/gd/>



SAPIENZA
UNIVERSITÀ DI ROMA

Esercizio 1 (su concorrenza)

Sia dato il seguente schedule S sulle transazioni t1, t2 e t3:

w3(Z) w1(Y) c3 r2(Y) w2(Z) w1(X) c1 c2

Dire se il suddetto schedule appartiene alla classe degli schedule two-phase locking con lock condivisi ed esclusivi. Nel caso in cui la risposta sia negativa, motivare la risposta stessa, e nel caso in cui sia positiva, mostrare lo schedule two-phase locking ottenuto da S inserendo opportunamente i comandi di lock (xli per lock esclusivo, sli per lock condiviso) e unlock (ui). Si chiede anche di dire, motivando la risposta, se lo schedule appartiene alla classe strict 2PL, ed alla classe ACR (Avoid Cascading Rollback).

Soluzione esercizio 1

Inseriamo i comandi di lock e unlock seguendo la regola di inserirli il più tardi possibile. Otteniamo:

xl3(Z) w3(Z) xl1(Y) w1(Y) c3 xl1(X) u1(Y) sl2(Y) r2(Y) u3(Z) xl2(Z)
w2(Z) w1(X) c1 c2 u1(X) u2(Z) u2(Y)

- Questo è uno schedule two-phase locking con lock condivisi ed esclusivi.
- Al contrario, non appartiene alla classe strict 2PL, perché u1(Y) non può che essere messo prima di c1.
- Infine lo schedule S non è ACR, perché t2 legge Y scritto da t1 quando t1 non ha ancora fatto commit.

Esercizio 1b (su concorrenza)

Dato il seguente schedule S:

$S = r_1(A) \ r_2(B) \ w_1(A) \ w_1(C) \ c_1 \ r_2(C) \ w_2(B) \ w_3(B) \ w_3(A) \ r_2(A) \ w_4(D) \ c_2$
 $w_4(B) \ c_3 \ c_4$

1. dire se S è eseguibile da uno scheduler che segue il protocollo 2PL con lock esclusivi e condivisi. In caso positivo, completare lo schedule S con le opportune istruzioni di lock e unlock. In caso negativo, motivare la risposta;
2. dire se S è uno schedule stretto, motivando la risposta;
3. dire se S è ACR (Avoid Cascading Rollback), motivando la risposta;
4. dire se S è conflict-serializzabile. In caso positivo, scrivere uno schedule seriale S' che sia conflict-equivalente a S. In caso negativo, motivare la risposta.

Soluzione esercizio 1b

Risposta 1:

$S = r_1(A) \ r_2(B) \ w_1(A) \ w_1(C) \ c_1 \ r_2(C) \ w_2(B) \ w_3(B) \ w_3(A) \ r_2(A) \ w_4(D) \ c_2$
 $w_4(B) \ c_3 \ c_4$

Si ha che:

(1) La transazione T2 deve rilasciare il lock su B prima di W3(B) (per permettere a T3 di acquisire il lock esclusivo su B).

(2) D'altra parte però, T2 deve necessariamente acquisire il lock su A dopo w3(A) (prima di tale istruzione, T2 ha il lock esclusivo su A).

Le condizioni (1) e (2) non possono essere soddisfatte nel protocollo 2PL (perché T3 effettuerebbe il lock su A dopo aver effettuato l'unlock di B). Quindi lo schedule S non è eseguibile dal protocollo 2PL.

Soluzione esercizio 1b (segue)

Risposta 2:

$S = r1(A) r2(B) w1(A) w1(C) c1 r2(C) w2(B) w3(B) w3(A) r2(A) w4(D) c2$
 $w4(B) c3 c4$

Uno schedule è stretto se ogni transazione legge da e scrive su transazioni che hanno già fatto commit.

Si ha che $w3(B)$ NON rispetta la condizione, in quanto T3 scrive B su T2 che non ha ancora fatto commit.

Pertanto S non è uno schedule stretto.

Soluzione esercizio 1b (segue)

Risposta 3:

$S = r1(A) r2(B) w1(A) w1(C) c1 r2(C) w2(B) w3(B) w3(A) r2(A) w4(D) c2$
 $w4(B) c3 c4$

Uno schedule è ACR se ogni transazione legge da transazioni che hanno già fatto commit.

Si ha che:

- $r2(C)$ rispetta la condizione, in quanto T2 legge C da T1 che ha già fatto commit;
- $r2(A)$ NON rispetta la condizione, in quanto T2 legge A da T3 che non ha ancora fatto commit.

Pertanto S non è uno schedule ACR.

Soluzione esercizio 1b (segue)

Risposta 4:

$S = r1(A) r2(B) w1(A) w1(C) c1 r2(C) w2(B) w3(B) w3(A) r2(A) w4(D) c2$
 $w4(B) c3 c4$

Uno schedule è conflict-serializzabile se il grafo delle precedenze dello schedule è aciclico.

Nel grafo delle precedenze di S c'è l'arco da $T2$ a $T3$ a causa della presenza di $w3(B)$ dopo $w2(B)$, e c'è l'arco da $T3$ a $T2$ a causa della presenza di $r2(A)$ dopo $w3(A)$. Di conseguenza, il grafo delle precedenze è ciclico e pertanto S non è uno schedule conflict-serializzabile.

Esercizio 2 (organizzazione fisica)

Si consideri la relazione

SCRITTURA(codiceaut, codicelibro, titololibro, ...),

che memorizza informazioni sui libri scritti dagli autori. Si assuma che gli autori siano 65.000, che mediamente ogni autore abbia scritto 10 libri, che sulla relazione SCRITTURA sia stato costruito un indice ad albero B+-tree non clusterizzato con search key <codiceaut>, con fan-out pari a 10, e tale che in ogni foglia ci sia un numero di data entries pari a circa 650. Si assuma che il fan-out e il numero di data entries sopra riportati rappresentino il 67% dei corrispondenti valori teorici massimi.

Sulla base di queste assunzioni e del fatto che non si ritiene trascurabile il tempo medio di elaborazione su un record, fornire una valutazione del costo medio dell'operazione che trova tutte le informazioni sui libri scritti da un autore con un certo codiceaut. Infine, dire quanti record della relazione SCRITTURA entrano in una pagina.

Soluzione esercizio 2

Sia D il tempo di lettura di una pagina su disco, e C il tempo medio di elaborazione di un record. Sappiamo che una foglia contiene 650 data entry, che la chiave di ricerca è $\langle \text{codiceaut} \rangle$, e che gli autori sono 65.000. Siccome l'indice non è clusterizzato, nelle foglie dobbiamo prevedere, per ogni valore di codiceaut , tanti data entry quanti sono i libri scritti da quell'autore. Pertanto, si hanno in tutto 650.000 data entry, e quindi 1000 foglie. Poiché il fan-out è 10, il costo dell'operazione è:

$$D \log_{10} 1000 + C \log_2 650 + 10 D$$

(assumiamo che la radice del B+-tree sia nel buffer)

Infine, sappiamo che

- un data entry di una foglia dell'indice ha una dimensione pari circa ad un decimo di un data record
- una pagina con data entry è occupata circa al 67%

Abbiamo allora che il numero di record contenuti in una pagina del disco è pari a $R=650/(10*0.67)=650/6.7$.

Esercizio 3 (XML)

Data la seguente DTD:

```
<!ELEMENT root (employee*) >
<!ELEMENT employee (ssn, name, previousJob+, projects) >
<!ATTLIST employee age CDATA #required >
<!ELEMENT ssn (#PCDATA) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT previousJob (#PCDATA) >
<!ELEMENT projects (project*) >
<!ELEMENT project (#PCDATA) >
```

1) scrivere lo schema relazionale ottenuto applicando l'algoritmo schema-driven di XML shredding;

(segue...)

Esercizio 3 (XML) (segue)

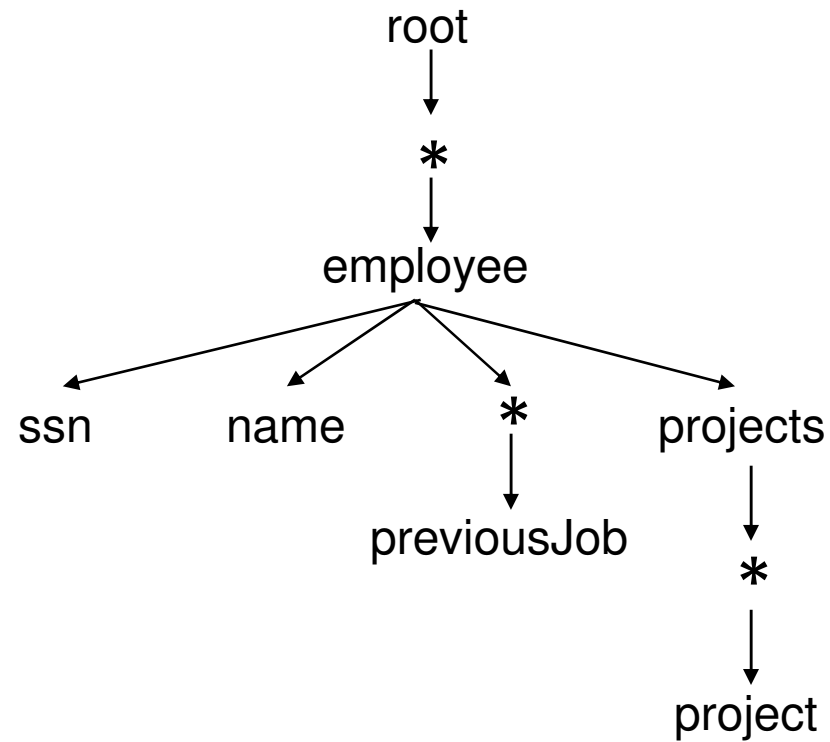
2) dato il seguente documento XML:

```
<root>
  <employee age="31">
    <ssn>1234 </ssn>
    <name> Rossi Mario</name>
    <previousJob> Soc. ABC </previousJob>
    <previousJob> Soc. XYZ </previousJob>
    <projects>
      <project> PR01 </project>
      <project> PR02 </project>
    </projects>
  </employee>
</root>
```

scrivere l'istanza della base di dati ottenuta traducendo il precedente documento nello schema relazionale ottenuto al punto 1.

Soluzione esercizio 3

1) Il grafo DTD è il seguente:



Soluzione esercizio 3 (segue)

Schema relazionale generato dall'algoritmo schema-driven:

root(rootID)

employee(empID, rootID, ssn:string, name:string, age:string, projectsID)

previousJob(previousJobID, empID, previousJob:string)

project(projectID, projectsID, project:string)

Le dipendenze funzionali sono le seguenti:

employee.rootID \subseteq root.rootID

previousJob.empID \subseteq employee.empID

project.projectsID \subseteq employee.projectsID

N.B.: l'elemento projects è inlined

Soluzione esercizio 3 (segue)

2) l'istanza di base di dati generata a partire dal documento XML è la seguente:

root(r1)

employee(e1, r1, "1234", "Rossi Mario", "31", ps1)

previousJobs(p1, e1, "Soc. ABC")

previousJobs(p2, e1, "Soc. XYZ")

project(pr1, ps1, "PR01")

project(pr2, ps1, "PR02")

Esercizio 3b

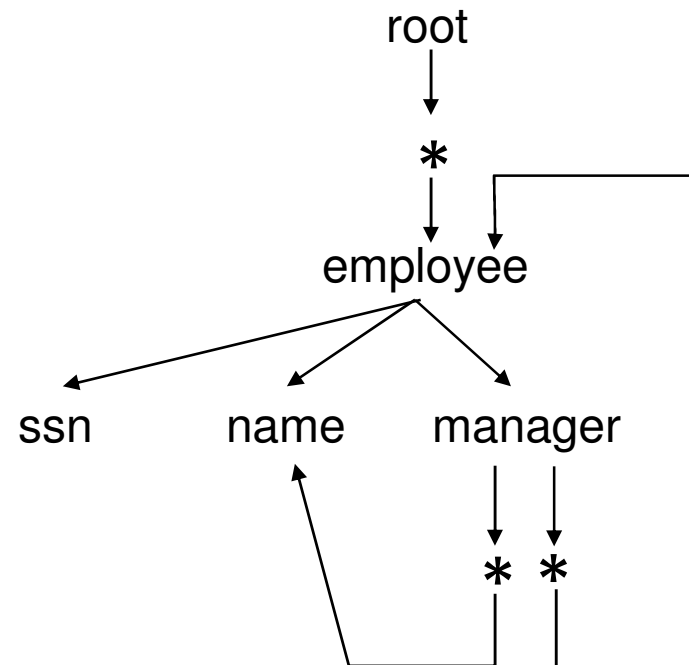
Data la seguente DTD:

```
<!ELEMENT root (employee*) >  
<!ELEMENT employee (ssn, name, manager) >  
<!ATTLIST employee age CDATA #required >  
<!ELEMENT ssn (#PCDATA) >  
<!ELEMENT name (#PCDATA) >  
<!ELEMENT manager (employee*, name*) >
```

scrivere lo schema relazionale ottenuto applicando l'algoritmo schema-driven di XML shredding.

Soluzione esercizio 3b (segue)

1) Il grafo DTD è il seguente:



Soluzione esercizio 3b (segue)

Lo schema relazionale generato dall'algoritmo schema-driven a partire dal grafo DTD è il seguente:

```
root(rootID)
employee(empID, parentID, code, ssn:string, age:string, managerID)
name(nameID, parentID, code, name:string)
```

Le dipendenze funzionali (oltre alle chiavi sottolineate) sono le seguenti:

```
if employee.code =0 then employee.parentID  $\subseteq$  root.rootID
if employee. code =1 then employee.parentID  $\subseteq$  employee.managerID
if name.code = 0 then name.parentID  $\subseteq$  employee.empID
if name.code = 1 then name.parentID  $\subseteq$  employee.managerID
```

N.B.: l'elemento manager è inlined

Esercizio 4 (valutazione query)

Si consideri una base di dati contenente la relazione *Studente* con attributi *Matricola*, *Cognome*, *Nome*, *DataNascita*, *CodiceComune*, e la relazione *Comune* con attributi *CodiceComune*, *NomeComune*, *NumeroAbitanti*, *Provincia*, *CAP*. Si assuma che la relazione *Studente* contenga 100000 record e che ogni record di tale relazione abbia dimensione $N/40$, dove N è la dimensione di una pagina di memoria. Inoltre si assuma che la relazione *Comune* contenga 1000 record e che ogni record di tale relazione abbia dimensione $N/50$.

Si consideri ora la query:

```
select S.Cognome, S.Nome, C.NumeroAbitanti
from Studente S, Comune C
where S.CodiceComune = C.CodiceComune
```

Calcolare il costo della valutazione della query nelle seguenti ipotesi:

1. il sistema esegue l'algoritmo naive nested loop;
2. il sistema esegue l'algoritmo nested loop;
3. il sistema esegue l'algoritmo block nested loop, assumendo 12 pagine disponibili del buffer.

Soluzione esercizio 4

1) Costo dell'algoritmo naive nested loop: $M + (p_R \times M \times N)$

con M = numero pagine outer relation

N = numero pagine inner relation

p_R = numero record outer relation

scegliamo Comune come outer relation (perché è la relazione più piccola), pertanto

$$M = 1000 / 50 = 20$$

$$N = 100000 / 40 = 2500$$

$$p_R = 1000$$

il costo è quindi $20 + (1000 \times 20 \times 2500) = 20 + 5 \times 10^7$

Soluzione esercizio 4

2) Costo dell'algoritmo nested loop: $M + (M \times N)$

con M = numero pagine outer relation

N = numero pagine inner relation

scegliamo ancora Comune come outer relation, pertanto

$$M = 1000 / 50 = 20$$

$$N = 100000 / 40 = 2500$$

il costo è quindi $20 + (20 \times 2500) = 50020$

Soluzione esercizio 4

3) Costo dell'algoritmo block nested loop: $M + (N \times M / (G - 2))$

con M = numero pagine outer relation

N = numero pagine inner relation

G = numero pagine buffer disponibili

scegliamo ancora Comune come outer relation, pertanto

$$M = 1000 / 50 = 20$$

$$N = 100000 / 40 = 2500$$

$$G = 12$$

il costo è quindi $20 + (20 \times 2500 / 10) = 5020$

Esercizio 4b

Sia data la seguente query:

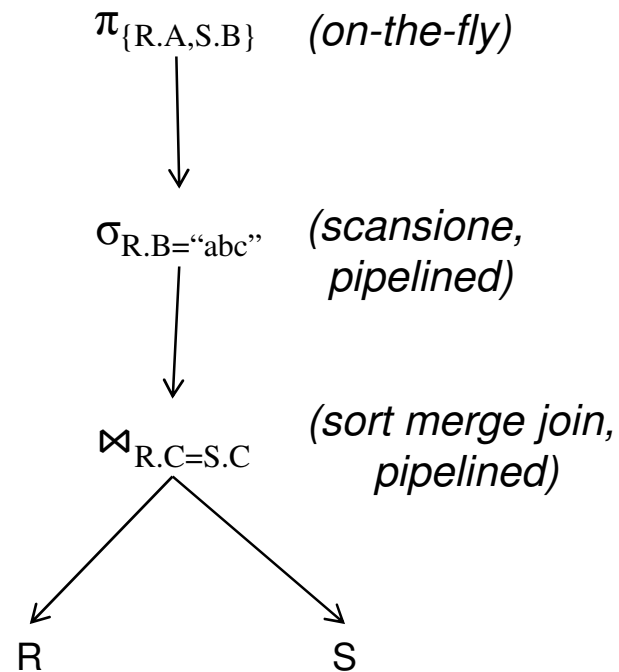
```
SELECT R.A, S.B  
FROM R, S  
WHERE R.B="abc" AND R.C = S.C
```

e si assuma che sulla relazione R sia dichiarato un indice hash con chiave di ricerca B, che sulla relazione S sia dichiarato un indice hash con chiave di ricerca C, che il DBMS possa eseguire i join sia mediante l'algoritmo sort merge join che mediante l'algoritmo block nested loop.

- 1) Scrivere due query plan per tale query.
- 2) Si assuma che R sia contenuto in 200 pagine, S sia contenuta in 100 pagine, che i record di R con B="abc" siano contenuti in 1 pagina e che il DBMS abbia a disposizione 10 pagine per eseguire i join. Valutare il costo dell'esecuzione dei due query plan (espresso come numero di trasferimenti di pagine da memoria di massa).

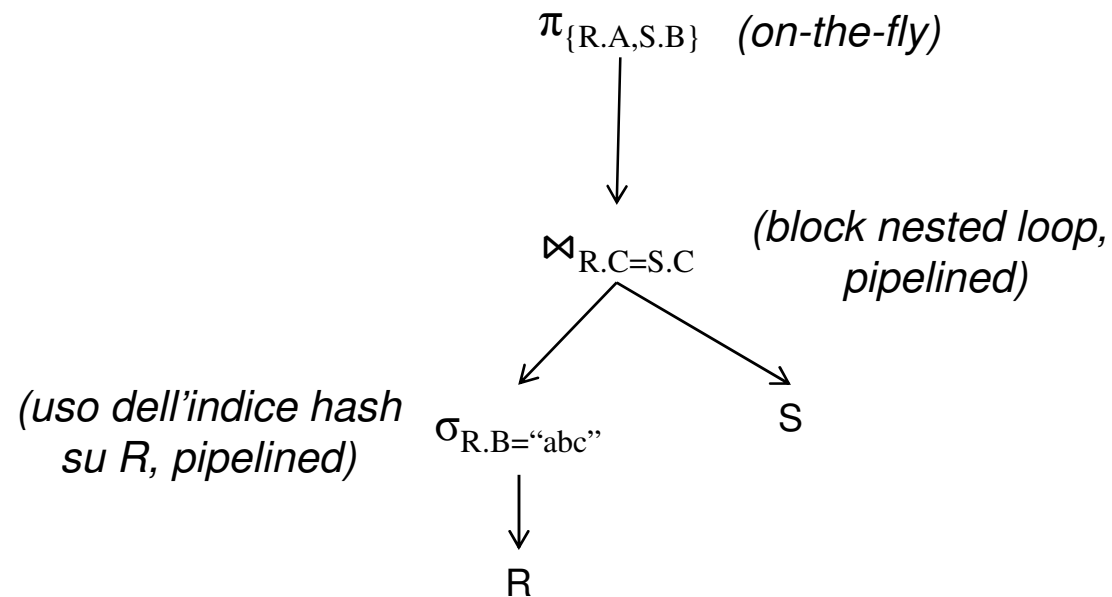
Soluzione esercizio 4b

Primo query plan QP1:



Soluzione esercizio 4b

Secondo query plan QP2:



Soluzione esercizio 4b

Costo dell'esecuzione di QP1:

- 1) Sort merge join di R e S = $100 \log_{10} 100 + 200 \log_{10} 200 + 100 + 200$
= $200 + 600 + 100 + 200 = 1100$ pagine trasferite da memoria di massa
- 2) Selezione sul risultato di 1) = nessun trasferimento di pagina da memoria di massa (pipelined)
- 3) Priorizzazione sul risultato di 2) = nessun trasferimento di pagina da memoria di massa (pipelined)

Numero totale di trasferimenti di pagine da memoria di massa = 1100

Soluzione esercizio 4b

Costo dell'esecuzione di QP2:

- 1) Selezione su R tramite indice hash = 1 pagina trasferita da memoria di massa
- 2) Block nested loop tra il risultato di 1) e S = 100 = 100 pagine trasferite da memoria di massa
- 3) Priorizzazione sul risultato di 2) = nessun trasferimento di pagina da memoria di massa (pipelined)

Numero totale di trasferimenti di pagine da memoria di massa = $1+100 = 101$

Pertanto QP2 risulta più efficiente di QP1.

Esercizio proposto

Sia data la seguente query:

```
SELECT R.A, S.B  
FROM R, S  
WHERE R.B="abc" AND R.C = S.C
```

e si assuma che sulla relazione R sia dichiarato un indice hash con chiave di ricerca B, che sulla relazione S sia dichiarato un indice hash con chiave di ricerca C, che il DBMS possa eseguire i join sia mediante l'algoritmo sort merge join che mediante l'algoritmo block nested loop. Si assuma inoltre che R sia contenuto in 200 pagine, S sia contenuta in 100 pagine, che i record di R con B="abc" siano contenuti in 1 pagina e che il DBMS abbia a disposizione 10 pagine per eseguire i join.

Scrivere il **migliore** query plan per tale query e calcolare il costo dell'esecuzione del query plan (espresso come numero di trasferimenti di pagine da memoria di massa).