Tecnologie delle Basi di Dati M

Appello del 21/9/2012

Esercizio 1 (4 punti)

Si considerino le seguenti transazioni:

```
T1: read(T1, PA), write(T1, PA), commit(T1)
T2: read(T2, PB), write(T2, PA), commit(T2)
```

ed il seguente schedule:

```
read(T1, PA), read(T2, PB), write(T2, PA), commit(T2),
write(T1, PA), commit(T1)
```

si risponda alle seguenti domande, giustificando le risposte:

1. Lo schedule è serializzabile?
2. Lo schedule potrebbe essere prodotto da un protocollo Strict-2PL?
3. Quale schedule verrebbe generato da un protocollo Strict-2PL che ricevesse i comandi nell'ordine indicato in precedenza (si supponga che ogni comando di lettura/scrittura sia preceduto dalla corrispondente richiesta di lock)?
4. A quale schedule seriale è equivalente lo schedule generato al passo precedente?

Esercizio 2 (3 punti)

Data la relazione con schema:

`Impiegati(matricola, nome, residenza, mansione)`

si consideri la seguente interrogazione SQL:

```
SELECT I.nome
FROM Impiegati I
WHERE I.mansione = dirigente
  AND I.residenza = 'Bologna'
```

tenendo conto che entrambi i predicati hanno un fattore di selettività del 10% e che la relazione Impiegati contiene 2000 tuple ed occupa 100 pagine. Si indichi il costo del miglior piano di accesso nei seguenti casi:

1. Presenza di un solo indice clustered su `mansione` (10 foglie).
2. Presenza di un solo indice clustered su `residenza` (15 foglie).
3. Presenza di un solo indice unclustered (TID ordinate) su `mansione` (10 foglie).
4. Presenza di un solo indice unclustered (TID ordinate) su `residenza` (15 foglie).
5. Presenza di un solo indice clustered su <`mansione`, `residenza`> (30 foglie).
6. Presenza di un solo indice clustered su <`mansione`, `nome`> (30 foglie).

Suggerimento: per la formula di Cardenas si utilizzino i seguenti valori, validi per P = 100:

R	Φ(R, 100)
10	9.56
20	18.21
30	26.03
40	33.10
50	39.50
60	45.28
70	50.52
80	55.25
90	59.53
100	63.40

R	Φ(R, 100)
110	66.90
120	70.06
130	72.92
140	75.51
150	77.85
160	79.97
170	81.89
180	83.62
190	85.19
200	86.60

R	Φ(R, 100)
210	87.88
220	89.04
230	90.09
240	91.04
250	91.89
260	92.67
270	93.37
280	94.00
290	94.58
300	95.10

Esercizio 3 (5 punti)

Si illustrino le modalità di gestione dell'overflow in strutture hash statiche.

Esercizio 4 (3 punti)

Si consideri un sistema che usi l'algoritmo ARIES per il crash recovery. Si indichino quali sono le due condizioni per cui è possibile evitare il REDO di un record di log di update facente riferimento ad una pagina P (suggerimento: si ricordino le due strutture accessorie utilizzate assieme al log durante il recovery). Si illustri anche perché convenga testare una delle condizioni prima dell'altra.

Soluzione Esercizio 1

1. No: lo schedule potrebbe essere equivalente solo allo schedule seriale (T2, T1), ma è possibile che il valore di PA scritto da T1 dipenda dal valore letto in precedenza.
2. No, in quanto T2 non è in grado di ottenere un lock esclusivo su PA fino al termine di T1.
3. `read(T1, PA), read(T2, PB), write(T1, PA), commit(T1), write(T2, PA), commit(T2)`
4. Lo schedule precedente è equivalente allo schedule seriale (T1, T2).

Soluzione Esercizio 2

Costo scan sequenziale = **100**
Costo indice clustered su `mansione`: NL x 0.1 + NP x 0.1 = 10 x 0.1 + 100 x 0.1 = 1 + 10 = **11**
Costo indice clustered su `residenza`: NL x 0.1 + NP x 0.1 = 15 x 0.1 + 100 x 0.1 = 2 + 10 = **12**
Costo indice unclustered su `mansione`: NL x 0.1 + Φ(NT/10, NP) = 10 x 0.1 + Φ(200, 100) = 1 + 87 = **88**
Costo indice unclustered su `residenza`: NL x 0.1 + Φ(NT/10, NP) = 15 x 0.1 + Φ(200, 100) = 2 + 87 = **89**
Costo indice clustered su <`mansione`, `residenza`>: NL x 0.01 + NP x 0.01 = 30 x 0.01 + 100 x 0.01 = 1 + 1 = **2**
Costo indice clustered su <`mansione`, `nome`>: NL x 0.1 + NP x 0.01 = 30 x 0.1 + 100 x 0.1 = 3 + 10 = **13**