Tecnologie delle Basi di Dati M

Appello del 24/6/2011

Esercizio 1 (2 punti)

Data la relazione con schema:

`Docenti(matricola, nome, residenza, luogonascita, datanascita)`

si effettui una stima del numero di pagine necessarie per memorizzare la relazione e del numero di livelli (e di nodi) di un B^+-tree costruito sull'attributo (`residenza`). Si supponga di avere pagine di dimensione 4 KB, di cui 96 B riservati per il page header, e si considerino i seguenti valori:

- Numero di tuple = 15K
- Numero di chiavi (`residenza`) = 1.5K
- Dimensione `matricola` = 4 byte
- Dimensione `nome` = 32 byte
- Dimensione `residenza` = 22 byte
- Dimensione `luogonascita` = 18 byte
- Dimensione `datanascita` = 4 byte
- Dimensione RID = 5 byte
- Dimensione PID = 4 byte
- Percentuale di riempimento foglie = 90%

Esercizio 2 (5 punti)

Data la relazione con schema:

`Personale(matricola, nome, data, luogo, stipendio, responsabile)`

si ottimizzi l'esecuzione della seguente interrogazione SQL:

```
SELECT P.matricola, R.matricola
FROM Personale P, Personale R
WHERE P.responsabile=R.matricola
  AND P.stipendio>R.stipendio
  AND P.stipendio<38000
  AND R.luogo in ('Milano', 'Bologna')
```

tenendo conto che dai cataloghi della base di dati risulta:

- Numero di tuple `Personale` = 50K
- Numero di pagine `Personale` = 1.5K
- Numero di responsabili = 100
- Indice unclustered (TID ordinate) su `luogo`: numero foglie = 50, numero chiavi = 100
- Indice clustered su `stipendio`: numero foglie = 2500, valore minimo = 20000, valore massimo = 200000
- Indice unclustered su `matricola`: numero foglie = 2000

Si disegni infine l'albero corrispondente al piano di accesso di costo minimo e stimi il numero di risultati dell'interrogazione (suggerimento: si noti che sono presenti due diversi predicati su `P.stipendio`; per i predicati di cui non è noto il fattore di selettività si usi il valore di default).

Suggerimento: per la formula di Cardenas si utilizzino i seguenti valori, validi per P = 1500:

R	Φ(R, P)
0	0
50	49.19198
100	96.77072
150	142.7891
200	187.2984
250	230.348
300	271.9858
350	312.2581
400	351.2097
450	388.8838

R	Φ(R, P)
500	425.3225
550	460.5662
600	494.654
650	527.624
700	559.5127
750	590.3557
800	620.1872
850	649.0403
900	676.9472
950	703.939

R	Φ(R, P)
1000	730.0455
1050	755.2959
1100	779.7182
1150	803.3396
1200	826.1863
1250	848.2838
1300	869.6566
1350	890.3285
1400	910.3225
1450	929.6608

Esercizio 3 (5 punti)

Si illustri il funzionamento di un DBMS basato sul protocollo ARIES nella fase di restart seguita ad un malfunzionamento di tipo "system failure".

Esercizio 4 (3 punti)

Si illustri quale sia l'effetto di incrementare/decrementare la dimensione della pagina dati su una struttura ad indice multi-dimensionale paginata quale R-tree, discutendone in particolare l'impatto sui costi di inserimento e ricerca.

Soluzione Esercizio 1

Dimensionamento relazione:
Dimensione di ogni tupla = 4 + 32 + 22 + 18 + 4 = **80B**
Numero di tuple per pagina = (4096 – 96)/80 = 4000/80 = **50**
Numero di pagine della relazione = NT/50 = 15000/50 = **300**

Dimensionamento indice (`residenza`):
Numero di chiavi = 1.5K, mediamente ci sono 15K/1.5K = 10 tuple per ogni valore di chiave.
Dimensione di ogni record (foglia) = 22 + 10 x 5 = **72B**
Dimensione "reale" foglia = (4096 – 96) x 0.90 = **3600B**
Numero di record per foglia = 3600/72 = **50**
Numero di foglie = 1500/50 = **30**
Dimensione di ogni record (nodo interno) = 22 + 4 = **26B**
Numero nodi livello 1 = 30 x 26/4000 = **1**
Il B^+-tree corrispondente si compone quindi di 2 livelli per un totale di 1 nodo interno (radice) e 30 foglie.

Soluzione Esercizio 2

Selettività dei predicati:
Predicato `P.stipendio>R.stipendio` = selettività di default = 1/2 = **0.5**
Predicato `P.stipendio<38000` = (38000 – 20000)/(200000 – 20000) = **0.1**
Predicato su `luogo` = 1/100 = **0.01** per ogni valore di `luogo`
Predicato di join = 1/50K (chiave esterna)

Accesso a P:
Costo scan sequenziale = **1500**
Costo indice su `stipendio`: NL x 0.1 + NP x 0.1 = 2500 x 0.1 + 1500 x 0.1 = 250 + 150 = **400**
Numero tuple residue = NT x 0.1 = **5000**

Accesso a R:
Costo scan sequenziale = **1500**
Costo indice su `luogo`: 2 x (NL x 0.01 + Φ(NT x 0.01, NP)) = 2 x (50 x 0.01 + Φ(50K x 0.01, 1.5K)) = 2 x (1 + Φ(500, 1500)) = 2 x (1 + 426) = **854**
Costo indice su `stipendio`: NL x 0.5 + NP x 0.5 = 2500 x 0.5 + 1500 x 0.5 = 1250 + 750 = **2000**
Costo indice su `matricola`: 1 + 1 = **2**
Numero tuple residue = 2 x NT x 0.01 = **1000**

Costi di join:
`P` esterna: costo = costo indice su `stipendio` + 5000 x costo indice `matricola` = 400 + 5000 x 2 = **10400**
`R` esterna: costo = costo indice `luogo` + 1000 x costo indice su `stipendio` = 854 + 1000 x 400 = **400854**
Il numero di risultati dell'interrogazione è 50K x 0.02 x 0.1 x 0.5 = **50**

Tecnologie delle Basi di Dati M

Appello del 24/6/2011

Exercise 1 (2 points)

Given the relation with schema:

`Teachers(code, name, address, birthplace, birthdate)`

estimate the number of disk pages needed to store the relation and the number of levels (and of nodes) of a B^+-tree built on the attribute (`address`). Suppose that the disk page size is 4 KB, of which 96 B are reserved for the page header, e consider the following values:

- Number of tuples = 15K
- Number if values (`address`) = 1.5K
- Size of `code`= 4 byte
- Size of `name` = 32 byte
- Size of address = 22 byte
- Size of birthplace = 18 byte
- Size of birthdate = 4 byte
- RID size = 5 byte
- PID size = 4 byte
- Fill ratio of leaf nodes = 90%

Exercise 2 (5 points)

Given the relation with schema:

`Employee(code, name, date, place, salary, director)`

optimize the processing of the following SQL query:

```
SELECT P.code, R. code
FROM Employee P, Employee R
WHERE P. director=R.code
  AND P.salary>R.salary
  AND P.salary<38000
  AND R.place in ('Milano', 'Bologna')
```

considering that database catalogs contain the following values:

- Number of tuples in `Employee` = 50K
- Number of pages of `Employee` = 1.5K
- Number of directors = 100
- Unclustered index (ordered TIDs) on `place`: number of leaves = 50, number of keys = 100
- Clustered index on `salary`: number of leaves = 2500, minimum value = 20000, maximum value = 200000
- Unclustered index on `code`: number of leaves = 2000

Finally, draw the tree corresponding to the minimum cost access plan and estimate the number of result tuples (advice: note that two different predicates involve `P.salary`; use default values for those predicates where the selectivity factor is unknown).

Advice: for the Cardenas formula, use the following values, valid for P = 1500:

R	Φ(R, P)
0	0
50	49.19198
100	96.77072
150	142.7891
200	187.2984
250	230.348
300	271.9858
350	312.2581
400	351.2097
450	388.8838

R	Φ(R, P)
500	425.3225
550	460.5662
600	494.654
650	527.624
700	559.5127
750	590.3557
800	620.1872
850	649.0403
900	676.9472
950	703.939

R	Φ(R, P)
1000	730.0455
1050	755.2959
1100	779.7182
1150	803.3396
1200	826.1863
1250	848.2838
1300	869.6566
1350	890.3285
1400	910.3225
1450	929.6608

Exercise 3 (5 points)

Show how a DBMS based on the ARIES protocol would manage the restart phase following a "system failure".

Exercise 4 (3 points)

Discuss the effects of increasing/decreasing the size of a node page on a multi-dimensional paged index structure like R-tree, in particular showing the impact on inserting/searching costs.

Answer to Exercise 1

Relation size:
Tuple size = 4 + 32 + 22 + 18 + 4 = **80B**
Number of tuples per page = (4096 – 96)/80 = 4000/80 = **50**
Number of pages for the relation = NT/50 = 15000/50 = **300**

Index size (`birthplace`):
Number of keys = 1.5K, on average there are 15K/1.5K = 10 tuples for each key.
Record size (leaf node) = 22 + 10 x 5 = **72B**
"Real" size leaf node = (4096 – 96) x 0.90 = **3600B**
Number of records per leaf node = 3600/72 = **50**
Number of leaf nodes = 1500/50 = **30**
Record size (internal node) = 22 + 4 = **26B**
Number of level 1 nodes = 30 x 26/4000 = **1**
The B^+-tree has 2 levels, with 1 internal node and 30 leaf nodes.

Answer to Exercise 2

Selectivity of predicates:
Predicate `P.salary>R.salary` = default selectivity = 1/2 = **0.5**
Predicate `P.salary<38000` = (38000 – 20000)/(200000 – 20000) = **0.1**
Predicate on `place` = 1/100 = **0.01** for each value of `place`
Join predicate = 1/50K (external key)

Access to `P`:
Cost of sequential scan = **1500**
Cost for index on `place`: NL x 0.1 + NP x 0.1 = 2500 x 0.1 + 1500 x 0.1 = 250 + 150 = **400**
Number of residual tuples = NT x 0.1 = **5000**

Access to `R`:
Cost of sequential scan = **1500**
Cost for index on `place`: 2 x (NL x 0.01 + Φ(NT x 0.01, NP)) = 2 x (50 x 0.01 + Φ(50K x 0.01, 1.5K)) = 2 x (1 + Φ(500, 1500)) = 2 x (1 + 426) = **854**
Cost for index on `salary`: NL x 0.5 + NP x 0.5 = 2500 x 0.5 + 1500 x 0.5 = 1250 + 750 = **2000**
Cost for index on `code`: 1 + 1 = **2**
Number of residual tuples = 2 x NT x 0.01 = **1000**

Join costs:
`P` external: cost = cost for index on `salary` + 5000 x cost for index on `code` = 400 + 5000 x 2 = **10400**
`R` external: cost = cost for index on `place` + 1000 x cost for index on `salary` = 854 + 1000 x 400 = **400854**
The number of query results is 50K x 0.02 x 0.1 x 0.5 = **50**