RWTH AACHEN UNIVERSITY

Implementation of Databases
Assignment #4

Due on December 13, 2016

Prof. Dr. rer. pol. Matthias Jarke

Dr. rer. nat. Christoph Quix

Submitted by:

Sanchit Alekh, Idil Esen Zulfikar

MSc. Software Systems Engineering

Problem 1

I/O Costs of Access Paths

Referring to Slide 31 of Chapter 2, please reason below formulas in detail:

1. Why is the cost for an equality selection using a sorted relation $D \times \log_2 B$?

   ## Solution
   Since the relation is sorted, the cost for searching the records that satisfy the equality condition is equal to the cost of page access while performing a Binary Search, i.e. $\log_2 B$. Hence the cost of equality selection is $D \times \log_2 B$

2. Why is the cost for a Range Selection using a clustered tree index $D \times (\log_G 0.15B + \#matchingpages)$?

   ## Solution
   Assuming that the load factor per page is 67%, the actual number of physical pages will be equal to $\frac{3}{2} \times B = 1.5B$. Furthermore, assuming that the data entry in leaf nodes is 10% of the actual record size, we conclude that there are $\frac{1}{10} \times 1.5B = 0.15B$ leaf pages. Consequently, traversing the B+ Tree Index will take $\log_G 0.15B$ I/Os. There will also be extra overhead cost of I/O for each matching page containing records satisfying the range selection. Therefore, the total cost involved is $D \times (\log_G 0.15B + \#ofmatchingpages)$

3. Why is the cost for an equality selection using an unclustered hash index $2D$?

   ## Solution
   After applying the hash function on the equality condition, we obtain the matching bucket, i.e. the page that contains the records matching the equality condition. Retrieving this page costs 1 I/O. Furthermore, retrieving the physical page where the qualifying record is present on disk costs 1 more I/O. Therefore the total cost is $2D$ I/Os.

4. Why is the cost for a delete operation using an unclustered tree index $D \times (3 + \log_G 0.15B)$?

   ## Solution
   Deletion involves two types of costs. Foremost is the I/O cost for finding and accessing the record to be deleted. This involves the access cost for the index and the relevant page. Again, assuming a 67% load factor and 10% data entry in the leaf

node, the cost of index access will be $D \times \log_G 0.15B$. The cost of retrieving the relevant page containing the record from memory costs 1D. The second type of cost is to the write back the pages after deletion. Writing back the physical page takes 1 I/O and writing back the index page, another 1 I/O. Therefore, the total cost involved is $D \times (\log_G 0.15B + 1 + 1 + 1) = D \times (3 + \log_G 0.15B)$

Problem 2

Given a relational table EMPL(eno,name,salary,marstat,dno) which is stored in an unsorted heap file with 1,000 pages (primary key is eno). Your system should be optimized for the following queries:

1. `Q1: SELECT * FROM EMPL WHERE eno = 4711`
2. `Q2: SELECT name,salary FROM EMPL WHERE salary > 40000 AND salary < 50000`
3. `Q3: SELECT dno, AVG(salary) FROM EMPL WHERE marstat = 'single' group by dno }`

How do you physically organize your database? Which indexes(clustered/unclustered) should be created to optimize the overall performance for all three queries? What are the estimated costs for your solution based on the information on Slide 31 of Chapter 2 (for the fan-out of tree index G we take 100)?

Solution

We suggest the creation of the following indexes on the *EMPL* relation to optimize the queries listed in the question:

1. An unclustered B+ Tree Index on eno
2. A clustered B+ Tree Index on the composite key $<marstat, dno>$

The I/O costs required to execute the above queries if we create the suggested indexes on the database are as follows:

1. **Q1:** It will take $\lceil \log_{100} 0.15B \rceil + 1$ number of I/Os to execute, which is equal to $\lceil 1.088 \rceil + 1 = 2 + 1 = 3$ I/Os. As *eno* is the primary key, there will be only one matching record.
2. **Q2:** As there is no index on salary, we will have to do a sequential scan. Assuming that 10% of the records match the condition, the query will take $\frac{10}{100} \times 1000 = 100$ I/Os. However, in the worst case, it can take up to 1000 I/Os.

Problem 2 continued on next page...

3. **Q3:** We have used a clustered index on the composite key $<marstat, dno>$. Therefore, the equality condition will cost $\lceil \log_{100} 0.15B \rceil + m$ I/Os, where m = # of matching pages. In this case, it will be (2+m) I/Os. The composite key will make sure that retrieved tuples are already grouped according to dno.

Problem 3

Part 1
Given is the following extensional database:

- Child(X,Y) : X is child of Y
- Female(X) : X is a female person

Define the following relations of the intensional database by specifying appropriate Datalog rules (you may define additional rules for your convenience):

(a) Cousin (X,Y): X is a cousin of Y

(b) Nephew (X,Y): X is a nephew of Y

(c) Uncle (X,Y): X is an uncle of Y

(d) GreatUncle (X,Y) : X is a great uncle of Y

Solution

The following additional rules are defined for convenience :

- $Parent(X, Y) : -Child(Y, X)$
- $Male(X) : - \ \neg Female(X)$
- $Sibling(X, Y) : -Parent(Z, X), Parent(Z, Y)$
- $Grandparent(X, Y) : -Parent(X, Z), Parent(Z, Y)$

Based on the rules defined above, and the rules given in the question, we define the following relations as:

(a) **Cousin (X,Y): X is a cousin of Y**

$Cousin(X, Y) : -Child(B, X), Child(A, Y), Sibling(X, Y)$
Note: This rule holds true for **First Cousins** only. We define the Second Cousin relation as:
$SecondCousin(X, Y) : -Grandparent(A, X), Grandparent(B, Y), Sibling(A, B)$

(b) **Nephew (X,Y): X is a nephew of Y**

$Nephew(X, Y) : -Sibling(Z, Y), Parent(Z, X), Male(X)$

(c) **Uncle (X,Y): X is an uncle of Y**

$Uncle(X, Y) : -Parent(Z, Y), Sibling(X, Z), Male(X)$

(d) **GreatUncle (X,Y) : X is a great uncle of Y**

$GreatUncle(X, Y) : -Parent(Z, Y), Uncle(X, Z)$

Part 2
Consider the following Datalog program

$$F = r(3, 4), r(5, 2), a(5), a(2)$$

R:
q(X) :- p(X), r(Y, X), b(Y)
b(X) :- r(X,Y), a(X)
p(X) :- a(X), NOTb(X)

(a) Define the Herbrand base for the rules R and facts F.

Solution

The Herbrand Base is given as follows:
r(2,2) r(2,3) r(2,4) r(2,5)
r(3,2) r(3,3) r(3,4) r(3,5)
r(4,2) r(4,3) r(4,4) r(4,5)
r(5,2) r(5,3) r(5,4) r(5,5)
a(2) a(3) a(4) a(5)
b(2) b(3) b(4) b(5)
p(2) p(3) p(4) p(5)

(b) Is the program stratified? Draw the stratification graph.

Solution

Yes, the program is stratified. The stratification graph is given in the Figure 1.

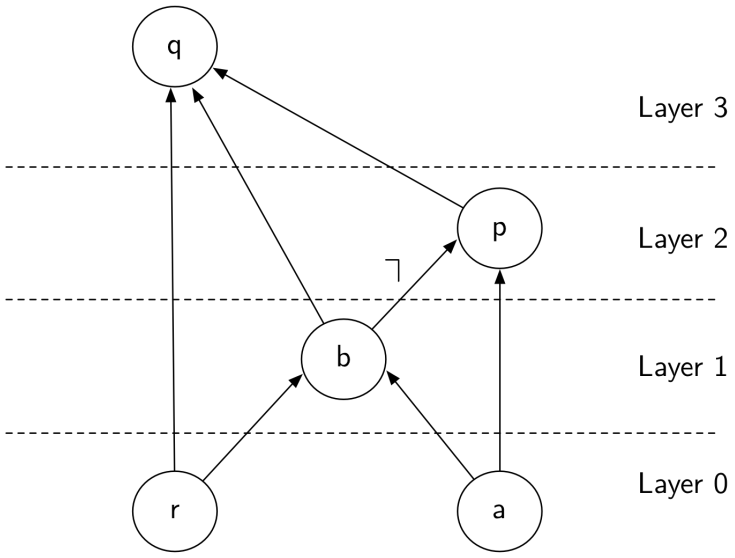

Figure 1: Stratification Graph for the Program

(c) Compute the least fixpoint for the stratified program (stratum by stratum) and the facts F.

Solution

- **Layer 0:** F (F is the set of facts)
- **Layer 1:** $F \cup \{b(5)\}$
- **Layer 2:** $F \cup \{b(5), p(2)\}$
- **Layer 3:** $F \cup \{b(5), p(2), q(2)\}$

After Layer 3, no new facts can be derived. Therefore, the least fixpoint has been reached.