경희대학교 시험용지

2010년도 제 2 학기

과목명	파일처리	교수명	이경구 교수님	성적	검인

학부 학과	컴퓨터공학과	학년	1	학번	20103102444	성명	황현우

1. merge join은 만약 자료가 정렬되어 있지 않은 자료라면 알고리즘을 이용하여 정렬한다. 예를 들어 R_1 R_2가 정렬되어 있지 않다면 정렬한 후 R_1을 읽고 난 후 R_2를 읽는다.

2(a) $3(B(R)+B(S)) = 3(1,000+1,000) = 6,000$

(b) $\min(B(R), B(S)) \leq M^2$ $(M = 201-1 = 200)$

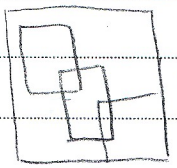
$= 1000 \leq 40000$

1000 block ÷ 200 block = 5 bucket ∴ 5 bucket

(c) $3(B(R)+B(S)) \div$ block $= 6000 \div 200 = 30$

∴ 30 bucket

3. R-tree는 사각형들을 이루어진 트리이며 트리안에 포인터를 저장이 가능하다. 저장이 안되는 것도 가능하다.



2 dimension을 가지고 있다.

B트리와 비슷하지만 다차원 공간을 갖는다.

4. Step1. Estimation = E⋈F⋈G⋈H

$= T(E) \times T(F) \times T(G) \times T(H)$

$= 1,000 \times 2,000 \times 3,000 \times 4,000$

$= 24,000,000,000,000$

Step2. attribute a = e / MAX(V(E,a), V(F,a), V(G,a))

$= 24,000,000,000,000 / 500 = 48,000,000,000$

b = e / MAX(V(E,b), V(F,b), V(H,G))

$= 24,000,000,000,000 / 400 = 6,000,000,000$

c = e / MAX(V(E,c), V(G,c), V(H,c))

$= 24,000,000,000,000 / 300 = 80,000,000,000$

d = e / MAX(V(F,d), V(G,d), V(H,d))

$= 24,000,000,000,000 / 800 = 3,000,000,000$

∴ e = a+b+c+d+e = 137,000,000,000

(Estimation)

(후면계속)

※반드시 검정색펜으로 쓰시오

(2)

경희대학교 시험용지

Query Processing (Professor Young-Koo Lee)

년도 제 학기

과목명		교수명	

성적	검인

학부 학과		학년		학번	2010311044	성명	Thai Thuy Han Uyen

※반드시 검정색 펜으로 쓰시오

(4) * Attribute a: we join E and F ; F and G

max [V(E,a), V(F,a)] = 500

max [V(E,a), V(G,a)] = 500

* Attribute b: join E and F ; F and H

max [V(E,b), V(F,b)] = 200

max [V(F,b), V(H,b)] = 400

* Attribute c: join E and G ; G and H

max [V(E,c), V(G,c)] = 300

max [V(G,c), V(H,c)] = 300

* Attribute d: join E and G, G and H

max [V(F,d), V(G,d)] = 100

max [V(G,d), V(H,d)] = 800

$$\text{Result} = \frac{1000 \times 2000 \times 3000 \times 4000}{500 \times 500 \times 200 \times 400 \times 300 \times 300 \times 100 \times 800} = \frac{1}{6000000} = \frac{1}{6.10^6}$$

(3) R-tree:

* R tree is used for answering question "where am I ?"

* A interior node is a region that has location (x_1, y_1) (x_2, y_2)

where (x_1, y_1) is a corner which is left, low

(x_2, y_2) is a corner which is right, above

Example: [(0,0)(40,50)]

* These regions can be overlapped

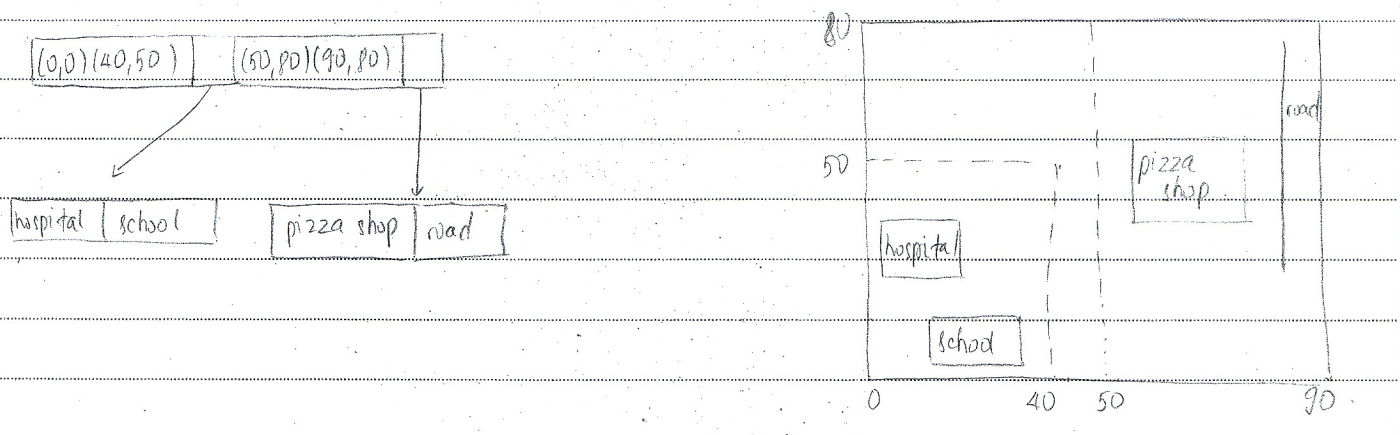
* A leaf node is object in a region

Example: [(0,0)(40,50)]

hospital | school

(후면계속)

Example for R - tree:



① Merge - join algorithm:

* Assume R_1, R_2 are sorted:

+ We read some blocks of R_1, R_2 into memory

→ Compare and join them.

Ex:

	i = 1	2	3	4	5
R_1	1	2	2	3	4
R_2	1	1	1	8	9
	j = 1	2	3	4	5

Pseudo - code:

```
if (a[i] == a[j])
    join them; j++; //continue compare a[i] and a[j+1]
else
    if(a[i] < a[j])
        i++;
    else // (a[i] > a[j])
        j++;
```

I/O's: $B(R_1) + B(R_2)$

* Assume R_1, R_2 are not sorted:

I/O's for sorting is $4(B(R_1) + B(R_2))$

$\Rightarrow$ I/O's $= 5[B(R_1) + B(R_2)]$

* Improve this algorithm:

We save I/O's by combining second - step in sort with join

$\Rightarrow$ I/O's $= 3[B(R_1) + B(R_2)]$

② Hash - join algorithm

a/ the minimum amount of memory needed to perform:

$$M = \sqrt{B(R) + B(S)} = \sqrt{1600 + 1000} = \sqrt{2000} = 45 \text{ blocks}$$

경희대학교시험용지

Query Processing (Professor Young-koo Lee)

년도 제 학기

과목명		교수명	

학부 학과		학년		학번	2010311044	성명	Thai Thuy Han Uyen

성적	검인

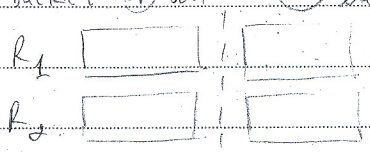
※반드시 검정색펜으로 쓰시오

② b/ Assume R and S are joined on attribute a (where a is integer)

We create two buckets:

① $x \bmod 2 \rightarrow$ odd number

② $x \bmod 2 \rightarrow$ even number



Because $N = 2021$ blocks

$\Rightarrow$ Each bucket = 100 blocks

And we have the minimum number of buckets is two buckets

c/ the maximum number of buckets is 100 buckets

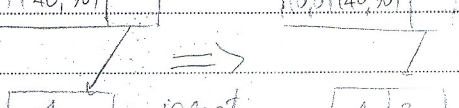
And each bucket = 2 blocks

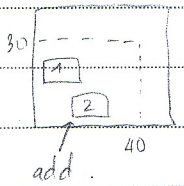
Add to problem 3 ③ Insert into R-tree

* If a object belongs completely to a region

→ It is okay, we add to leaf node this object

Ex: [0,0](40,30) ⟹ [0,0](40,30)





* not belong to completely a region

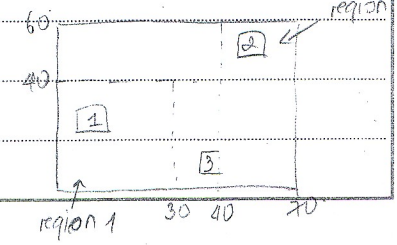
We can — add a new region (case 1)

extend a old region (case 2)

In case (2), assume we have many old regions for extending

Then we choose the region which save more square unit

Ex object 3 is inserted into region ② because it save more square unit than region ① after extending



(후면계속)

경희대학교시험용지

2010 년도 제 2 학기

과목명	Query Processing	교수명	Young-Koo Lee

학부 학과	Computer Engineering	학년	3	학번	2009315302	성명	Guillermo Crocker

성 적	검 인

※반드시 검정색펜으로 쓰시오

1) Merge Join algorithm

The merge join algorithm at first make the selection of 2 tables, then arranges the content of this 2 tables in another one or one projection, one by one.

Discussion to get a better performance the arrange should be done in block, ~~an~~ indexing skema should be necesary and maybe some concurrency should be done.

2) R, S → 10,000 tuples → 1,000 blocks 10 tuples per relation

M → 201 blocks ~~#~~ of memory

a) $M = \sqrt{\max B(R), B(S)}$ simple sort based join algorithm

$M = \sqrt{1000} = 31.62 \approx 32$ blocks

b) Minimum number of buckets, how large?

Memory is available to allocate 20 blocks of ~~each~~ both of the relations, each relation has 100 blocks to join ⇒ each relation can ~~load~~ send 10 block to memory per I/O ∴ the minimum needed is 10 buckets of 10 pointers ~~to~~ block.

c) Maximum number of buckets, how large?

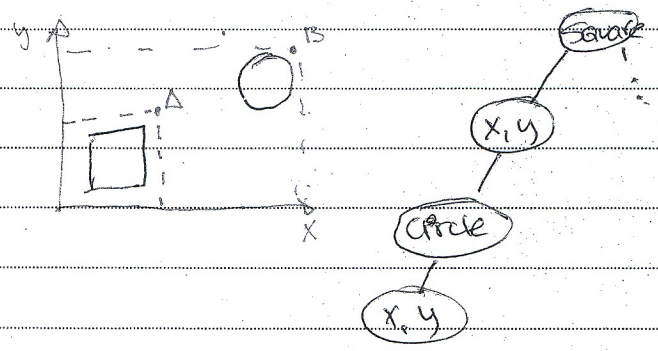
Each relation has 100 blocks to join, ~~the~~ we can send to memory just 1 block per relation, because we want to maximize the buckets ⇒ 2 blocks will be processed per I/O ∴ the maximum number of buckets is 100 with 1 pointer, ~~to~~ to block.

3) B - tree

Is a tree used to describe subdivided regions, it has the disadvantage of redundancy and its algorithm is recursive. ↳ or interpolated points

(후면계속)

3. The representation could be:



4) Multiple $T(w) = \frac{T(R) \times T(S)}{\max(V(R,a), V(S,a))}$

EMF = $\frac{1000 \times 2000^{a}}{500} + \frac{1000 \times 2000^{b}}{200} = \frac{1000 \times 2000}{100}$

4000 + 10,000 = 14,000 tuples

EMG = $\frac{1000 \times 3000^{a}}{500} + \frac{1000 \times 3000^{c}}{300}$ = 4000 + 10,000 = 14,000 tupl

EMH = $\frac{1000 \times 4000^{b}}{400} + \frac{1000 \times 4000^{c}}{200}$ = 10,000 + 20,000 = 30,000 tuples

FMG = $\frac{2000 \times 3000^{a}}{500} + \frac{2000 \times 3000^{d}}{100}$ = 12,000 + 60,000 = 72,000 tu

FMH = $\frac{2000 \times 4000^{b}}{400} + \frac{2000 \times 4000^{d}}{800}$ = 20,000 + 10,000 = 30,000 tuple

GMH = $\frac{3000 \times 4000^{c}}{300} + \frac{3000 \times 4000^{d}}{800}$ = 40,000 + 15,000 = 55,000 tuple

(후면 참)

경희대학교 시험용지

년도 제 학기

과목명	QUERY PROCESSING	교수명	PROF. YOUNG-KOO LEE.	성적	검인

학부 학과	COMPUTER ENGINEERING	학년	2nd.	학번	2010311076	성명	FATIMA IRAM.

※반드시 검정색펜으로 쓰시오

Q4-

$$E \bowtie F \bowtie G \bowtie H = \frac{T(E) \cdot T(F) \cdot T(G) \cdot T(H)}{\max(V(E,a), V(F,a)) \cdot \max(V(F,d), V(G,d)) \cdot \max(V(G,c), V(H,c))}$$

Consider that Join attribute between E & F is a and between F & G is d and between G and H is c.

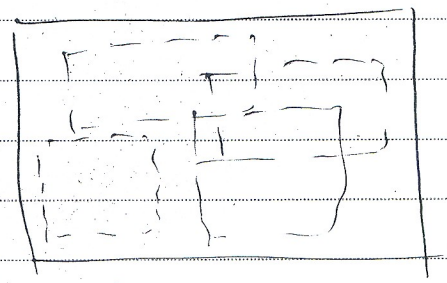
$$= \frac{1000 \times 2000 \times 3000 \times 4000}{500 \times 100 \times 300} = \frac{24,000\,000}{15} = 1600000 \text{ Tuples.}$$

So Joing these Relation with the mentioned Joining attributes have, 1600000 Tuples as a result of Join.

Q3-

R-Tree.

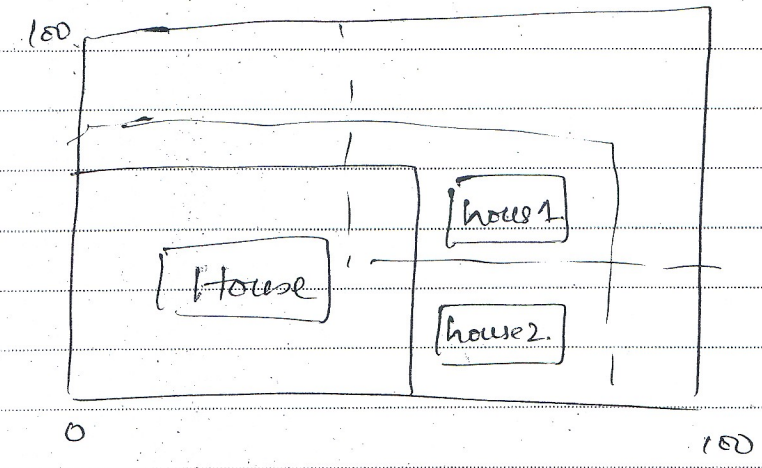
R-Trees are data structure which is used to store Two dimentional or more dimentional information. It is like a B-Tree with different Dimensions. The keys are like subregions in the R-Tree and nodes are like data regions



R-Tree node

(후면계속)

Alone
~~Below~~ is the node of the R-Tree, which has 4 leave nodes as denoted by the dashed rectangle with the data node. R-Trees are usually helpful in find "where am I" queries.
To find any point P, the search start from the root node and dig down to region that contains the value P. ~~For~~ There are three possibilities that either P do not lie, P lie in one data region. Or P lie in more than one region. So if do not lie then stop the search and if lie in One region then go for that region and if lie in more than One region then successively check that whether what is the exact location of the required P value in the available regions.
Usually some regions have overlaping regions with other regions' so in this case successively checking is necessary. Although while designing the R-Tree it should be tried that overlapping should be minimum.
For insertion of record in the R-Tree, first check the availability of the space in the region in which new record should inserted. It is highly recmended not to split the Tree too frequently while try to adjust the new records in the unallocated space of older data regions.



Example of R-Tree.

경희대학교시험용지

년도 제 학기

과목명	QUERY PROCESSING	교수명	Prof. YOUNG-KOO LEE.	성적	검인
학부 학과	COMPUTER ENGINEERING	학년	2nd.	학번	2010311076	성명	FATIMA IRAM.		

※반드시 검정색펜으로 쓰시오

Q1-

Merge-Join Algorithm :-

The cases when Table are not small enough to fit in memory then for those join, usually merge join algorithm used to join large relations so in this case more disk I/o are required as compare to joining the Tables which are already in memory.

Consider Two relation R & S and Memory Blocks available are M.

(1) Sort ~~Merge~~ Relation R (using Two phase merge Sort Algorithm) on y
(2) Sort ~~Merge~~ Relation S. Similarly on y.
(3) Now These Both sorted Relations are in memory.
(4) Take the smallest value of y from the sorted list
(5) If y exist in Both relations then Take them in memory
(6) Otherwise discard the values of y which are not in both Tables.
(7) Bring the Tuples from both Tables in memory for a particular value of y and Join the Tuples and give them output.

Problems :-

5 DISK I/o are required to perform the above. Join operation. For Sorting Both relations 4 DISK I/o are required as 2 Sort and then Sort and for the Joining One DISK I/o is more required.

(후면계속)

Improvement Techniques :-

If we consider that there are not such huge Tuples of Joining that may exceed the memory size.
So create M list of sorting of for both relations R and S.
So M are the available block and R are the no of list sorted which contain the relations. By applying this approach we can save up to two Disk I/O. As the second merge sort step will shift into join step as compare to the previous Algo.

So Take first Block of bell ~~memory~~ sorted list into memory get the min value of y. Join the Tuples on the basis of those values and & output the joined Tuples.
The constraint is that no of sorted list is M not more than that of relations satisfy this condition than No of Disk I/o are reduced by 2 and algorithm become more effecient.

Q2

(a) Min amount of memory needed = $\sqrt{B(S)} = \sqrt{1000} = 32$ ~~Blocks~~

(b) As the tuples are stored contiguously on disk so min no of buckets can be One which contain the info of the initial block of Disk which contain the Table and for the rest of the Table we know it & same contigiously. and bucket size may contain one record.

(c)

If we do not consider that Tuples are stored contiguously then we can use each Bucket for set of Block so ~~201 Block 201 Buckets~~ and Bucket size depends on the no of block it refers. max 201 Blocks can be referred through Buckets.