Introduction to database systems
2021/2022
Homework 2

SÜLEYMAN GÖLBOL [Erasmus Student]

Task 1 (50 %): Linear Hash Index

Given is the following linear hash index. Bucket is split every time when a new overflow page is added.
Show the state of the index after adding each key, 18, 20 and 27.

h_1	h_0	$Level = 1, N = 4$				
000	00	8*	16*			
001	01	5*	13*	21*		Next = 1
010	10	2*	10*	34*	70*	
011	11	7*	15*	59*	91*	
100	00	12*	20*	28*		

Adding 18 (next=1)

	h_1	h_0	$Level = 1, N = 4$				Overflow Pages			
0	000	00	8*	16*						
1	001	01	5*	13*	21*					
2	010	10	2*	10*	34*	70*	18*			
3	011	11	7*	15*	59*	91*				
4	100	00	12*	20*	28*					

RULES:
Split condition is when a new overflow page.
Before moving to other level, all 4 pages must be split.

formula for insertion: key mod $(2^{level} \cdot N)$ ↗ initial number of buckets at a certain level
formula for redistribution: key mod $(2^{level+1} \cdot N)$

(level=1) so 18 mod $(2^1.4) = 2$
18* will be added to row number 2
split condition is when a new overflow page
redistibute row=next which is 1

So redistribute;
5 mod $(2^1.4) = 5$
13 mod $(2^1.4) = 5$
21 mod $(2^1.4) = 5$
(18 was added so next++ => next = 2)

The new table

DECIMAL	h_1	h_0	$Level = 1, N = 4$, Next= 2				Overflow Pages			
0	000	00	8*	16*						
1	001	01								
2	010	10	2*	10*	34*	70*	18*			
3	011	11	7*	15*	59*	91*				
4	100	00	12*	20*	28*					
5	101	01	5*	13*	21*					

Adding 20

20 mod $(2^1*4) = 4$ but 20 is already added so we don't have to do anything.

Adding 27

27 mod $(2^1*4) = 3$
Redistribute row=next which is 2
2 mod $(2^1.4) = 2$
10 mod $(2^1.4) = 2$
34 mod $(2^1.4) = 2$
70 mod $(2^1.4) = 6$
18 mod $(2^1.4) = 2$

next++ => so next=3

DECIMAL	h_1	h_0	$Level = 1, N = 4$, Next= 2				Overflow Pages			
0	000	00	8*	16*						
1	001	01								
2	010	10	2*	10*	34*	18*				
3	011	11	7*	15*	59*	91*	27*			
4	100	00	12*	20*	28*					
5	101	01	5*	13*	21*					
6	110	10	70*							

# of index entries per page => should be rounded DOWN
# of within cotain index level => should be rounded UP
(15,6 = 16 Pages)

Task 2 (50 %): Query evaluation and optimisation

The most important tables in the library information system are the following.

int 8 Varchar100 varchar100 varchar100 int 12
Books (bid, author, title, publisher, year);
int 8 varchar 50 varchar 50 varchar 50 int42
Members (mid, name, surname, address, telephone);
int8 int 8 int8 int8 varchar8
Rental (rid, mid, bid, eid, date);
int 8 varchar 50 varchar 50 varchar 50 int 42
Employee (eid, name, surname, address, telephone)

The following information is given.

Multiple-byte units					
Decimal			Binary		
Value	Metric		Value	IEC	Legacy
1000	kB	kilobyte	1024	KiB kibibyte	KB kilobyte

my assumption for 1 KB is 1024 bytes (like prominently used by the Microsoft Windows operating system)

1 page on the disk = 8KB
|Books| = 1.000.000 records, 320 bytes, 25 records/page, 40000 pages
|Members| = 10.000 records, 200 bytes, 40 records/page, 250 pages
|Rental| = 300.000 records, 40 bytes, 200 records/page, 1500 pages
|Employee| = 100 records, 200 bytes, 40 records/page, 3 pages

We can have 10000 pages in the buffer.

The following indexes are set and available on the database:
- unclustered B+ index on Rental.date attribute
- clustered B+ index on Books.year attribute and
- hash indexes on all relation keys.

Assume that all tables are ordered by the values of their keys.

Our SUPB has the following join algorithms available:
- Sort-merge Join
- Hash Join

PART 1 (20%). What is the size of B+ index on attribute Books.year? DO NOT use estimations. Write down all assumptions.

clustered B+ index is available.

B+ tree

Except root, minimum %50 occupancy
d <= m <= 2d
Order of tree

1 page on disk => 8.1024 bytes

Books.year = 12 bytes (my assumption for Books.year)
page index identifier size = 4 bytes (my assumption for page identifier)
+
Size of 1 page index entry
12 + 4 = 16 bytes

Books.year = 12 bytes (my assumption for Books.year)
data record identifier size = 8 bytes (my assumption for record identifier)
+
Size of 1 data entry
12 + 8 = 20 bytes

Number of index entries/page = $\frac{8.1024}{16} = 512$

Number of data record entries/page = $\frac{8.1024}{20} \cong 410$

Leaf (Level 2) $\frac{1000000 \text{ record}}{410 \text{ record/page}} \cong 2440$ pages (block)

Level 1 $\frac{2440}{512} = 4,76 \cong 5$ pages

Root (Level 0) $\frac{5}{512} = 0,0097 = 1$ page

Total
2446 pages

$2446 \times 8 \text{ kB} = \frac{19568 \text{ KB}}{1024}$
$= 19,10$ MB

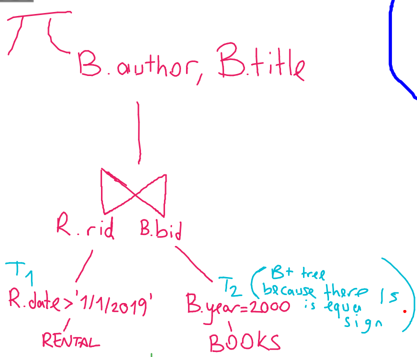
PART 2 (30%). Translate the following SQL statement into a relational algebra tree (query execution plan) and find a plan that reads the least pages:

```
SELECT B.author, B.title
FROM Rental R, Books
B
WHERE R.bid=B.bid AND B.year='2000' AND R.date > '1/1/2019'
```

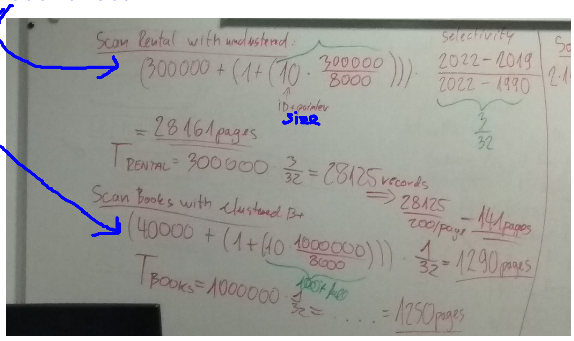
Rental.date ranges from 1.1.1990 to 31.12.2021. Books.year ranges from 1990 to 2021. For the sake of simplicity, assume that we are already in year 2022.

Costs of access path should be calculated with known formula.

Write down all assumptions.



cost of scan



Scanning Rental

1/1/2019 to 31/12/2021 is 3 years.
Totally there are 32 years.

$$300000 \cdot \frac{3}{32} = \frac{28125 \text{ records}}{200 \ \frac{\text{records}}{\text{page}}} = 140{,}6 \sim 141 \text{ page}$$

scan with unclustered b+ index and store to Trental

Scanning Books

From 1990 to 2021 there are 32 years.

$$\frac{1 \text{ million records}}{32} = \frac{31250 \text{ records}}{25 \ \frac{\text{records}}{\text{page}}} = 1250 \text{ pages}$$

scan with clustered b+ index and store to Tbooks

Sort Merge Join:

formula: $2M\left(1+\log_{B-1}\left(\frac{M}{B}\right)\right)+2N\left(1+\log_{B-1}\left(\frac{N}{B}\right)\right)+M+N$ $B=10k$, $B-1=999$

For T_1 Sorting ⟹

$$\underbrace{2 \cdot 141 \cdot \left(1+\frac{\log \frac{141}{10000}}{\log(10000-1)}\right)}_{110} + \underbrace{2 \cdot 1250\left(1+\frac{\log\frac{1250}{10000}}{\log(10000-1)}\right)}_{1750} + \underbrace{141+1250}_{1391}$$

$= 3232$ pages

Better option

Hash Join

$$3(\underset{\text{Table1}}{M}+\underset{\text{Table2}}{N})$$

$3(141+1250) = 3 \cdot 1391$

$= 4173$ pages