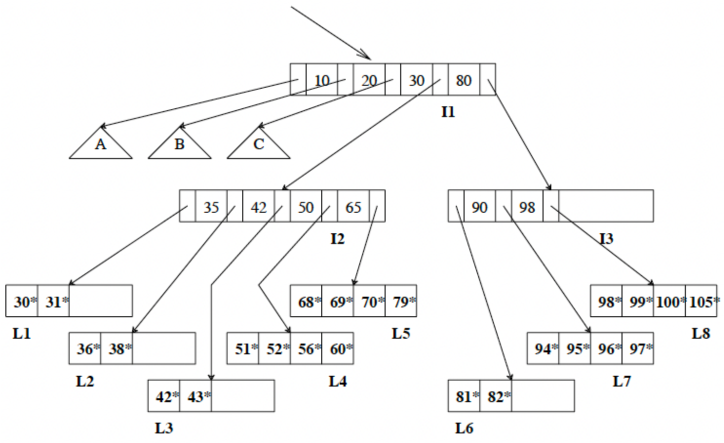
CSDS 433 Database Systems
Assignment 3

1. [B+ tree] (20) Consider the following B+ tree (partly shown). Leaf nodes are linked (the links are not shown in the figure). The nodes are annotated with a label (e.g., "L1" for leaf node 1). Answer the following questions.



(1) Give the labels of all the nodes that need to be visited to answer the query: "find all records with search key at least 51".
(2) Answer the same question, but for the query: "find the records with search key in [68, 100)".
(3) Subtree C is not shown. What can we infer about the contents and shape of this subtree?

(1) I1, I2, L4, L5, L6, L7, L8
(2) I1, I2, L5, L6, L7, L8
(3) Subtree C has height one and contains keys ≥ 20 and < 30, the intermediate node has 2 - 4 key values and 3 - 5 pointers.

2. [Equivalence of RA expressions] (20)
Consider the following equivalence transformation rules for RA expressions. For each rule, either briefly explain why the rule holds (and clarify the condition/assumption when it holds), or if the rule does not hold, provide a counter example.

a. $\sigma_{\theta \wedge \theta'}(E) = \sigma_{\theta}(\sigma_{\theta'}(E))$
b. $\sigma_{\theta}(E \bowtie_{\theta'} E') = E \bowtie_{\theta \wedge \theta'} E'$
c. E – E' = E' – E
d. $\sigma_{\theta}(E - E') = \sigma_{\theta}(E) - E'$

E: Relational-algebra expressions.
θ: Predicates.

a. ✓
Conjunctive selection operations can be deconstructed into a sequence of individual selections.
Assuming E have attributes in θ and θ'.

b. ✓
Selections can be combined with Cartesian products and theta joins.
Assuming both E and E' have attributes in θ and θ'.

c. ×
Counter Example: E = $\sigma_{eid = 11}$(Employee), E' = $\sigma_{eid > 10}$(Employee).

d. ✓
The selection operation distributes over the union, intersection, and set-difference operations.
Assuming E and E' are union compatible (have the same number of attributes, and each attribute in E associate with an attribute in E' with the same name and type).

3. [**Selection**] (30) Consider a schema with an Employee relation:
Employee (<u>eid</u>: integer, ssn: string, ename: string, age: integer, salary: integer)

CSDS 433 Database Systems

Assignment 3

Assume each tuple of Employee is 50 bytes long, a block can hold 80 Employee tuples, and there are in total 500 blocks of such tuples in an Employee table. The table stores information of employees with eid from 1 to 100,000. For each of the following selection query, estimate the number of blocks (worst case) to be retrieved. eid is the primary key, and ssn is a candidate key.

a). There is no index available;
query: $\sigma_{\text{Employee.salary = 2000}}$**(Employee)**

b). There is a B+ tree index on search key Employee.eid with range 1 to 100,000, with height 4;
query: $\sigma_{\text{Employee.eid=50,000}}$**(Employee)**

c). There is a B+ tree primary index on search key Employee.eid with range 1 to 100,000, with height 4;
query: $\sigma_{\text{Employee.eid<50,000}}$**(Employee)**

d). There is a secondary B+ tree index on search key Employee.ssn with range "000001" to "100000", with height 5;
query: $\sigma_{\text{Employee.ssn="13432"}}$**(Employee)**

e). There is a secondary B+ tree index on search key Employee.salary with range 1 to 100,000, with height 4;
query: $\sigma_{\text{Employee.salary<45000}}$**(Employee)**

a) 500 blocks (scan whole table)
b) 5 blocks (h + 1, from root to leaf)
c) 254 blocks (500 × (50000 ÷ 100000) + h, assuming eid follows uniform distribution)
d) 6 blocks (h + 1, from root to leaf)
e) 504 blocks (If done with index) or 500 blocks (If done by file scan)

4. [**Join**] (30) Consider a join $R \bowtie_{R.A=S.B} S$. We ignore the cost of output the result and measure the cost with the number of I/O – the number of data blocks that need to be transferred. Given the information about relations to be joined below: (1) Relation S contains 20K tuples and has 10 tuples per block; (2) Relation R contains 100K tuples and has 10 tuples per block. (3) Attribute B of S is the primary key of S. In total 52 pages (blocks) are available in memory.

ts = 20k, bs = 20k ÷ 10 = 2k, tr = 100k, br = 100k ÷ 10 = 10k

a. (10) Assume neither relation has any index. Describe a block nested join algorithm to evaluate the query. Give the cost of joining R and S with a block nested loops join.

Memory: give 50 pages to outer relation, 1 to inner relation, and 1 for output buffer.
Algorithm:
Since bs < br, so we set relation S as the outer relation to make less cost:

```
Read in 50 pages of S:
    Read in a page of R:
        Print tuples matching R.A = S.B
```

Cost:

1) Read S into memory once → bs,
2) Read R once for every 50 blocks of S → bs ÷ 50 × br.

So, the total cost is bs + bs ÷ 50 × br = 2k + 2K / 50 × 10k = 402K.

b. (20) Assume R contains 1000 blocks, 10,000 tuples; S contains 10,000 blocks, 50,000 tuples. There are clustering B+ tree index on R.A and S.B. Both indexes contain two levels with the root note in the first level and all leaf nodes in the second level. The indexes on R.A has 25 leaf nodes; the index on S.B contain 250 leaf nodes. Describe *two* algorithms that can correctly compute $\sigma_{R.A}$ **R** $\bowtie_{R.A=S.B}$ **S** and give the corresponding cost measured by the number of block transfers. (*There are multiple solutions; describing any two correct algorithms with cost analysis suffice*).

CSDS 433 Database Systems

Assignment 3

1. Indexed Nested-Loop join:

Algorithm:

Since br < bs, so we set relation R as the outer relation to make less cost:

```
For each tuple tr in R:
    Look up tuples in S by index:
        Print tuples matching R.A = S.B
```

Cost:

1) Read R into memory once → br,
2) For each tr, perform index loop up on s → tr × (h + 1),
3) Fetching all matching ts for each tr → tr.

So, the total cost is br + tr × (h + 1) + tr = br + 3tr = 1k + 3 × 10k = 31k.

2. Hash join:

Algorithm:

Partitioning:

1) Partition both relations using hash function h.

Probing:

2) Read in a partition of R, hash it using h2, then an in-memory hash index will be built on it using R.A,
3) Scan the corresponding partition of S and search for matches.

Cost:

1) Partitioning: scan R and S and write them out → 2(br + bs)
2) Probing: scan each partition once (assuming no partition overflows) of R and S → br + bs

So, the total cost is 3(br + bs) = 3 × (1k + 10k) = 33k.