Lecture 21 Multikey Files

Last Day: Query Processing with Inverted Lists
- Query Language
- Parse Trees
- Algorithms

Today: Threaded Files
- Answering queries
- Parse trees
- Algorithms

(2) Threaded files —— which we shall see next

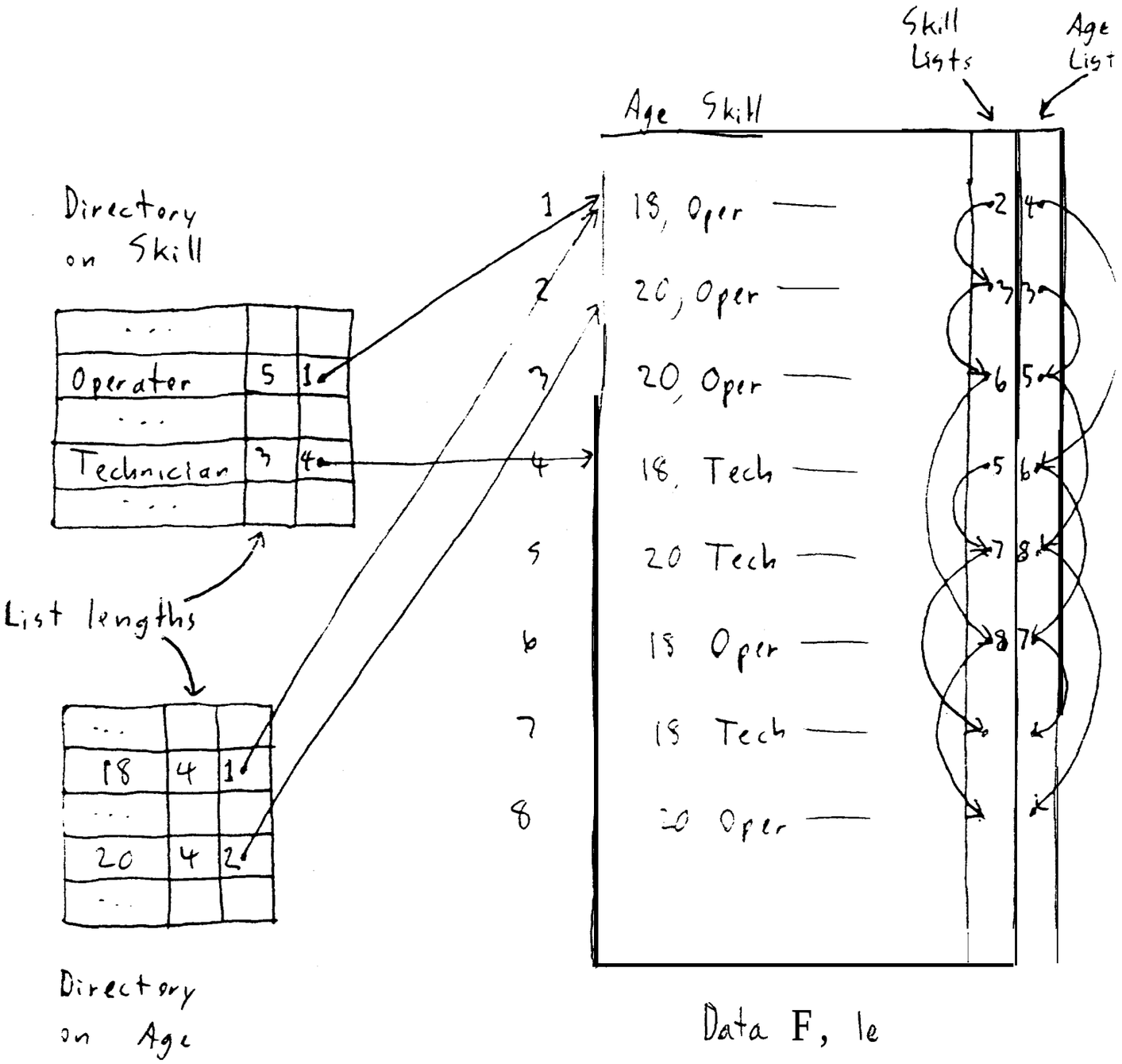
Threaded Files

- Unlike inverted lists, the secondary indices for a threaded file do not have accession lists.

- Instead, each data record has a number of extra pointer fields, one extra field for each secondary index.

- Data records with the same value for a given secondary key are linked together into a list (a "thread") using one of the extra fields.

Example of a Threaded File



- This data record is the head of a list of all other data records with value v for field F

- The linked lists running through the data file are called "threads" (one thread for each secondary key field F and each value, v).

Query Processing with Threaded Files

Example 1: Retrieve all operators

- Find operator entry in skill directory.
- Follow the pointer to record 1 in the data file
- Print record 1, & follow its skill ptr to record 2.
- Print record 2, & follow its skill ptr to record 3.
- Print record 3, & follow its skill ptr to record 6.
- Print record 6, & follow its skill ptr to record 8.
- Print record 8, & terminate, since its skill ptr is empty.

Records 1, 2, 3, 6, 8 have been retrieved & printed.

Example 2

Retrieve all 18-year olds.

- Find the entry for 18 in the Age directory
- Follow its pointer to record 1 in the data file.
- Follow the Age pointers to retrieve records 1, 4, 6, 7 (the age thread for 18-year olds
- Print each record as it is retrieved.

Example 3

Retrieve all 18-year old operators.

- Find the entry for 18 in the age directory.
- Note that the length of its thread is 4.

- Find the entry for operater in the skill directory
- Note that the length of its thread is 5

- Follow the Age thread, since it is shorter
- ⇒ Retrieve records 1, 4, 6, 7.

- Print these records with Skill = Operator.
- ⇒ Print records 1, 6.

Example 4

Retrieve all 18-year old technicians.

- Find the entry for 18 in the age direct
- Note that the length of its thread is 4.

- Find the entry for technician in Skill direc
- Note that the length of its thread is 3

- Follow the technician thread since it is shorter.
- i.e., Retrieve records 4, 5, 7.

- Print those records with Age = 18.
- i.e., Print records 4, 7.

Example 5

Retrieve all records for 18-year olds

OR technicians

- Find the entry for 18 in the age dir
- Follow its thread, & retrieve each record

- Find the entry for technician in the skill dir
- Follow its thread, & retrieve each record

- Print each record in the union of the retrieved record sets

- Note: Number of data-file accesses = 4+3
  ie, length of thread for 18 plus
  length of thread for technician

Query Processing for Threaded Files

In general, the approach is similar to that used for inverted lists.

- First, construct a parse tree for the query.
- Assign to each node, X, of the parse tree a label, SIZE (X).
- SIZE (X) represents the total length of all threads needed to answer the subquery at node X.
- i.e., SIZE (X) represents the number of data file accesses.
- SIZE (X) = ∞ represents the entire file

Main Questions

Given a query and a file with indices on some fields

(1) do the indices help to answer the query (or do we have to scan the entire file)?

(2) If the indices help, how should we use them?

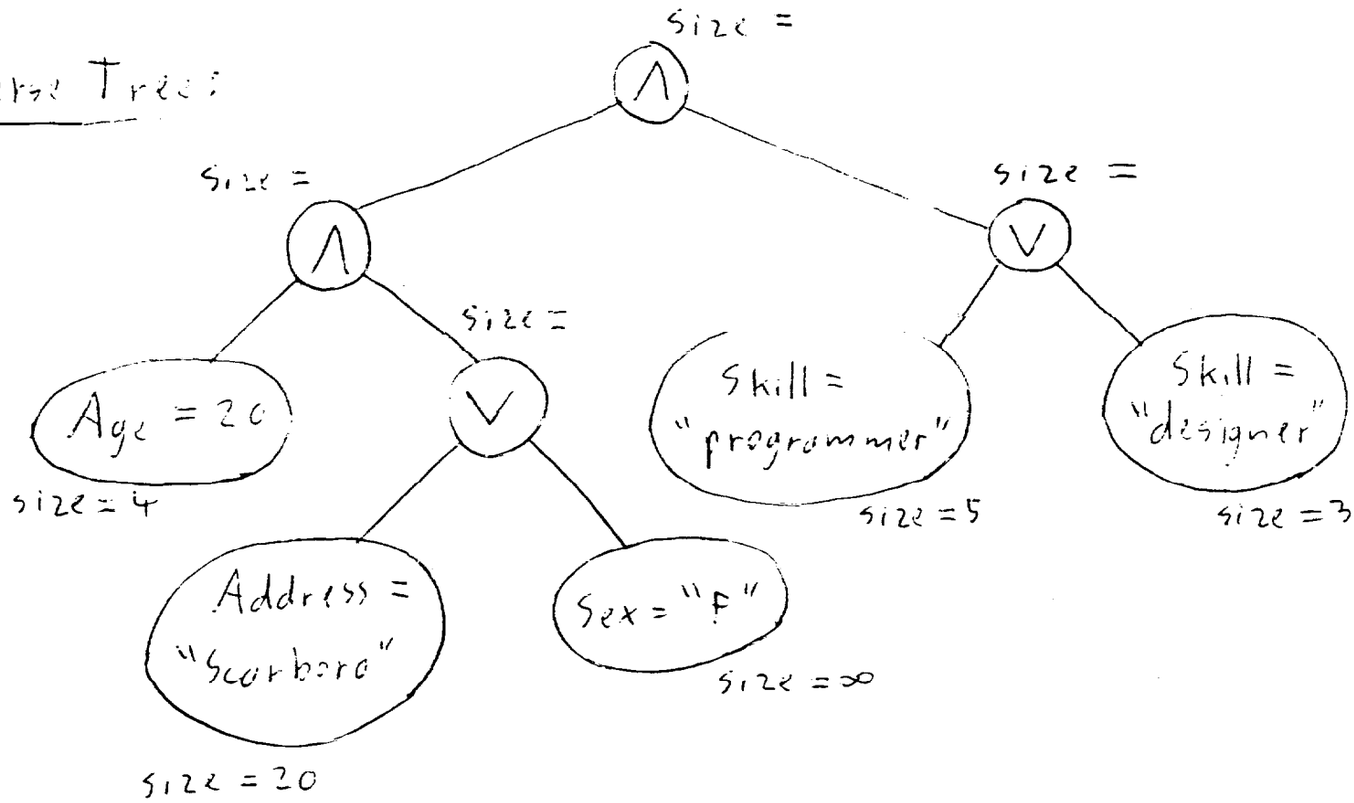
The indices help iff SIZE(root) $\neq \infty$

Example : Computing SIZE(X)

Query :

(Age = 20 AND (Address = "Scarboro" OR sex = "f"
AND
(Skill = "programmer" OR Skill = "designer")

Parse Tree :



- Given indexes on Age, Address & Skill, do they help? Yes, in this case.

To evaluate this query,

- Follow the thread for Age = 20.
- Keep those records with Address = "Scarboro" or Sex = "F".
- Of these records, keep those with Skill = "Programmer" or Skill = "designer".

Algorithm to Compute SIZE(X)

(1) If node X is a leaf of the form "F = v," then

- If F is indexed, then SIZE(X) is the length of the thread for value v + direct for F
- If F is not indexed, then SIZE(X) = ∞

(2) If X has the form (Y AND Z), then

$$SIZE(X) = \min[SIZE(Y), SIZE(Z)]$$

(3) If X has the form (Y OR Z), then

$$SIZE(X) = SIZE(Y) + SIZE(Z)$$

Here, $\infty + c = \infty$ and $\min(\infty, c) = c$.

Answering a Query

- Let XR be the set of data records (not points) satisfying the subquery at node X.
  (Eventually we want to compute XR for X = root.)

- If SIZE(X) = ∞, then the indices do not help and we must scan the entire file and check each record to see if it satisfies the query at X

- If SIZE(X) $\neq \infty$, then we can use the indices to answer the query more quickly

Algorithm to compute XR (when SIZE(X) ≠ ∞)

(1) If X is a leaf, then XR is computed directly by following a thread from an index entry

(2) If X has the form (Y and Z)

and SIZE(Y) ≤ SIZE(Z)

then compute YR

XR := ∅

For each record r ∈ YR,

if r satisfies the query at Z,

then XR := XR ∪ {r}

} i.e. follow the shortest thread.

(3) If X has the form (Y or Z)
then compute YR, ZR
XR := YR ∪ ZR

} if so, follow both threads