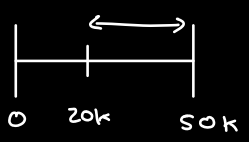
Tutorial Week 8

① Single-relation plans

NTuples (employees) = 10,000 pages × 20 tuples/page
= 200,000 tuples

a) sal > 20,000

sal ranges from 0 to 50,000



- Reduction factor (RF):

$$RF = \frac{High(sal) - value}{High(sal) - low(sal)}$$

$$= \frac{50{,}000 - 20{,}000}{50{,}000 - 0}$$

$$= 0.6$$

- 2 possible access paths:
  - full table scan
  - unclustered B+ tree index on sal

- Full table scan / sequential scan / heap scan

Cost = NPages (employees)

= 10,000 IO

careful: find the product of the reduction factors of matching predicates

- Unclustered B+ tree index on sal

Cost = (NPages (Index) + NTuples (employees)) × Π RF$_i$

= (500 + 200,000) × 0.6

= 120,300 IO

conditions that involve attributes in a prefix of the search key

cheapest access path is full-table scan

b) age = 25

$$RF = \frac{1}{NKeys(age)}$$

$$= \frac{1}{40}$$

40 distinct values of age

- 3 possible access paths:
  - full table scan
  - unclustered hash index on age
  - clustered B+ tree index on (age, sal) → age is within prefix of search key

- Full table scan

cost = 10,000 IO — Same as before

- unclustered hash index on age → RF of matching condition

$$Cost = NTuples(employees) \times \Pi RF_i \times 2.2$$

$$= 200{,}000 \times \frac{1}{40} \times 2.2$$

$$= 11{,}000 \text{ IO}$$

↳ 1.2 for the bucket check
1 to fetch page from disk

- clustered B+ tree index on (age, sal)

$$cost = (NPages(index) + NPages(employees)) \times \Pi RF_i$$

$$= (500 + 10{,}000) \times \frac{1}{40}$$

$$= 262.5$$

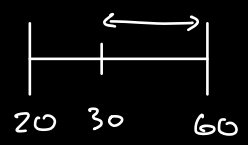
$$\simeq 263 \text{ IO}$$

RF of matching conditions

cheapest access path is clustered B+ tree index

c) age > 30

age ranges from 20 to 60



$$RF = \frac{high(age) - value}{high(age) - low(age)}$$

$$= \frac{60 - 30}{60 - 20}$$

$$= 0.75$$

- 2 possible access paths:
  - full table scan
  - clustered B^+ tree index on (age, sal) → age is within prefix of search key

Can't use unclustered hash index on age since we can't use hash index ove on range

- Full table scan

  cost = 10,000 IO Same as before

- clustered B^+ tree index on (age, sal)

  cost = (NPages(index) + NPages(employees)) × ΠRF_i

  = (500 + 10,000) × 0.75

  = 7,875 IO

  ↓ RF of matching conditions

cheapest access path is clustered B^+ tree index

d) eid = 1000

- 2 possible access paths:
  - full table scan
  - unclustered hash index on eid

- eid is a candidate key, so here, we're looking for just a single tuple
- unclustered hash index on eid is a primary index ie the search key of the index is the primary key of the employees relation
- selecting single tuple over primary key so:

unclustered hash index on eid

cost = 1.2 + 1 → data page access
(1.2 → hash lookup cost)

= 2.2 IIo

- Full table scan

cost = 10,000 IIO Same as before

cheapest access path is unclustered hash index on eid

e) $sal > 20{,}000 \wedge age > 30$

$$RF_{sal} = \frac{High(sal) - value}{High(sal) - low(sal)}$$

$$= \frac{50{,}000 - 20{,}000}{50{,}000 - 0}$$

$$= 0.6$$

as calculated before

$$RF_{age} = \frac{high(age) - value}{high(age) - low(age)}$$

$$= \frac{60 - 30}{60 - 20}$$

$$= 0.75$$

3 possible access paths:

- full table scan
- unclustered B^+ tree index on sal
- clustered B^+ tree index on (age, sal)
  - ↳ we will use product of RFs of both conditions, since attributes are in prefix of search key

filter age "on the fly" afterwards

↓

only RF_{sal} will be used since age condition doesn't match index

Can't use unclustered hash index on age since we can't use hash index ow on range

- Full table scan

  cost = 10,000 IO Same as before

- Unclustered B+ tree index on sal

  matching conditions

  Cost = (NPages(index) + NTuples(employees)) × TRF$_i$

  = (500 + 200,000) × 0.6 sal > 20,000 is the only matching condition

  = 120,300 IO Same as question 1a

- clustered B+ tree index on (age, sal)

  Cost = (NPages(index) + NPages(employees)) × TRF$_i$ 2

  = (1500 + 10,000) × 0.75 × 0.6 matching conditions

  = 4725 IO

  sal > 20,000 ∧ age > 30 ☺ a matching predicate

cheapest access path is clustered B+ tree index

② Multi-relation plans

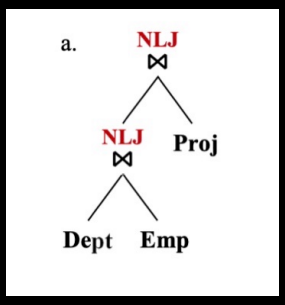
⓪ find number of pages for all relations

NPages (Emp) = NTuples (Emp) / NTuples/page
= 20,000 / 20
= 1,000 Pages

NPages (Dept) = NTuples (Dept) / NTuples/page
= 5,000 / 40
= 125 Pages

NPages (Proj) = NTuples (Proj) / NTuples/page
= 10,000 / 10
= 100 Pages

a) this left-deep plan is joining Dept with Emp using Nested Loop Join and then joining the result with Proj using Nested Loop Join



cost analysis :

① find cost of child join (join b/w first 2 relations)

Cost (D JOIN E) : NPages(R) + NPages(R) × NPages(S)
PNLJ — let D be the outer relation (R)

= NPages(D) + NPages(D) × NPages(E)
= 125 + 125 × 1000
= 125,125 IO

② find result size of child join in PAGES!!!

result size (D JOIN E) = $\prod$ NTuples(R_i) × $\prod$ RF$_i$ → in each relation

= NTuples(D) × NTuples(E) × $\frac{1}{\text{NKeys(J)}}$ → # distinct keys / values of the joining attribute

= 5,000 × 20,000 × $\frac{1}{500}$ → # distinct did since that's what you're joining E and D on

= 200,000 tuples

NPages (D JOIN E) = 200,000 tuples / 100 tuples/page
= 2,000 pages

③ find cost of parent join

Cost (JOIN P) = ~~NPages (D JOIN E)~~ + NPages (D JOIN E) × NPages (P)
PNLJ — outer relation is now the result of child join

↳ apply pipelining when performing parent join, so we can discard cost of 1st read of left input
↳ pipelining is the direct "streaming" in memory of the output of one operation (in this case, a join) as the input of another operation (eg another join) without writing the output to disk

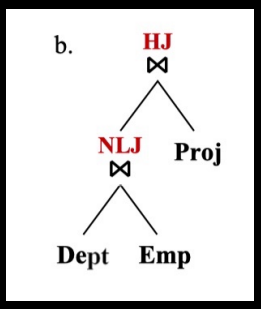
= 2,000 × 100
= 200,000 IO

④ add cost of child join and parent join

Total cost = 125,125 + 200,000

= 325,125 IO

b) this left-deep plan is joining Dept with Emp using Nested Loop Join and then joining the result with Proj using Hash Join



cost analysis:

↗ shorthand for "JOIN"

① cost (D × E) = 125,125 IIO same as before
PNLJ

② result size (D×E) in pages = 2,000 pages same as before

③ Cost (× P) = 3 × (NPages (D×E) + NPages (P)) − NPages (D×E)
HJ
↳ discard cost of first read of left input due to pipe-lining

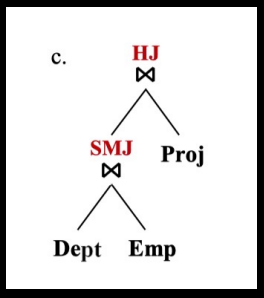
= 3 × (2,000 + 100) − 2,000

= 4,300 IIO

④ Total cost = 125,125 + 4,300

= 129,425 IIO

c) this left-deep plan is joining Dept with Emp using sort-merge join and joining results with Proj using Hash Join



① cost (D ⋈ E) = NPages (D) + NPages (E)
SMJ + 2 × NPages (D) × num_passes (D)
+ 2 × NPages (E) × num_passes (E)

= 125 + 1000 + 2 × 125 × 2 + 2 × 1000 × 2

= 5,625 IIO

② result size (D ⋈ E) in pages = 2,000 pages Same as before

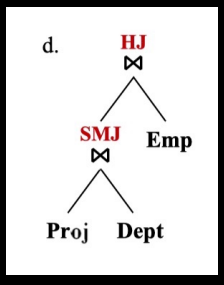
③ cost (⋈ P) = 4,300 IIO Same as before
HJ

④ Total cost = 5,625 + 4,300

= 9,925 IIO

Note: if data file is already sorted, cost of sorting can be ignored in SMJ

d) this left-deep plan is joining Proj with Dept using sort-merge join and joining results with Emp using Hash Join



① cost (P ⋈ D) = NPages(P) + NPages(D)
SMJ + 2 × NPages(P) × num_passes(P)
+ 2 × NPages(D) × num_passes(D)

= 100 + 125 + 2 × 100 × 2 + 2 × 125 × 2

= 1,125 I/O

② result size (P ⋈ D) = π NTuples(R_i) × π RF$_i$

= NTuples(P) × NTuples(D) × $\frac{1}{\text{NKeys(J)}}$ → # unique values of joining attribute

= 1,000 × 5,000 × $\frac{1}{1,000}$ → # distinct projid since that's what you're joining P and D on

= 5,000 Tuples

NPages(P ⋈ D) = 5,000 tuples / 50 tuples/page

= 100 Pages

③ cost(⋈ E) = 3 × (NPages(P ⋈ D) + NPages(E)) − NPages(P ⋈ D)
HJ

= 3 × (100 + 1,000) − 100

= 3,200 I/O

↳ discard cost of first read of left input due to pipelining

④ Total cost = 1,125 + 3,200

= 4,325 I/O