COMP421 Assignment 2

# Exercise 1: Data and Leaf Pages

1. Info:

Samples has 10,000 tuples

Data pages are filled to ~75%

Size of tuple in Samples: INT + VARCHAR(20) + DATE + CHAR(4) + VARCHAR(100) + pointer + = 4 + 10 + 10 + 4 + 50 = 78 bytes

Date page size: 4000 bytes

Calculations:

Number of tuples per page = (data page size) \* (fill) / (size of tuple)

= 4000\*0.75/78

= 38.4 ≈ 38 tuples per page

Number of Pages = (total number of tuples) / (tuples per page)

= 10,000 / 38

= 263.15 ≈ 260

**The relation Samples requires 260 data pages.**

1. Info:

Indirect index alternative II on region of Sequences

Number of data entries = distinct values of region = 500

Records per value of region: 50,000/500 = 100 records per val of Region

Size of data entry: sizeof(region) + 100 + sizeof(rids) = 3 + 100 \* 10 = 1003 bytes

Calculations:

Data entries per page = (data page size) \* (fill) / (data entry size)

= 4000 \* 0.75 / 1003

= 2.99 ≈ 3 data entries per page

Number of leaf pages = (number of data entries) / ( data entries per page)

= 500 / 3

= 166.67 ≈ 167

**The index has 167 leaf pages.**

# Exercise 2: Query Execution

1. SELECT seq

FROM sequences

WHERE size > 700 AND region = ’POL’

1. Indicate the access costs (in number of pages retrieved leading to I/O) and how you derive the calculation (by describing the calculations and the steps that are executed to perform the access) for this query if…
   1. There is a non-clustered indirect index, type II on size only and you use it for the query

Info:

Data entries per page = (data page size) \* (fill) / (data entry size)

= 4000 \* 0.75 / (4+50(10))

= 05.95 ≈ 6 data entries per page

Number of leaf pages = (# data entries) / ( data entries per page)

= 1000 / 6

≈ 167

Size: [100, 1100], uniformly distributed => 40% are > 700

Region: 100 records per value of region => 100 with value ‘POL’

Index is on size => cost is the same as a query for size > 700

Calculations:

67 = 40% of all leaf pages

20,000 = 40% of all data pages

Cost: 67 + 20,000 = 20,067

**Access cost is 20,067**

* 1. There is a non-clustered indirect index, type II on region only and you use it for the query

Info:

Number of leaf pages: 167 (from exercise 1)

Index is on region => cost is the same as a query for region = ‘POL’

Region: 100 records per value of region => 100 with value ‘POL’

=> 0.2%

Calculations:

1 = 0.2% of all leaf pages

100 data pages

Cost: 1 + 100 = 101

**Access cost is 21**

* 1. There is a non-clustered indirect index on size and a non-clustered indirect index on region, both type II and you use both.

Calculations:

# rids where size > 700: access 67 leaf pages (from part i)

# rids where region = ’POL’: access 1 leaf page (from part ii)

50,000\*0.4 (size)\* (1/500 (regions) ) = 40 tuples

Cost: 67 + 1 + 40 = 108

**Access cost is 108**

1. If size > 200 then the reduction factor is 90% and the query should use a non-clustered indirect index, type II on region only. If size > 1000 then reduction factor is 10% then a non-clustered indirect index on size and a non-clustered indirect index on region, both type II should be used for the query.
2. A clustered indirect index either size or region would increase performance because only 1 leaf page would need to be accessed.

2) SELECT region, count(\*)

FROM sequences

GROUP BY region

HAVING count(\*) > 120

1. Execution strategy:

Scan all tuples (10,000 pages) of Sequences into main memory and project region column (3\*50,000/4000 = 38 frames, fits in main memory), pipeline to scan and sort, this fits in main memory, pipeline to group by region, this fits in main memory. Scan the relation by group and select where count(\*) > 120. The result takes up (4+3)\*500/4000 = 0.88 data pages. Write this out as the result.

Number of I/O:

Number of data pages read = 10,000

Number of data pages written ≈ 1

I/O = 10,001

1. Execution strategy with index on region:

Scan in all leaf pages . (167 I/O total, in blocks of <50)

Count the number of rids per region (contained within the same leaf page), if there are greater than 120, retrieve this result.

Number of I/O:

167 Leaf pages read in

(4+3)\*500/4000 = 0.88 = 1 data page written out

I/O = 168

1. If the projection is count(DISTINCT sampleid) instead, I would create a clustered index on region and sampleid to speed up this query. Thus the indexes are grouped by region and the number of distinct sample ids can be determined with no page access.

# Exercise 3: Joins

Indirect index on sampleid on Samples

1. The number of output tuples of an equality join between Samples and Sequences:

Join attribute is sampleid, a primary key in Samples and a foreign key in Sequence. Each Sequences tuple matches with exactly one Samples tuple. **Thus number of output tuples is |Sequences| = 50,000**

1. Calculate the estimated I/O (a bit more than 50 buffer pages available) if…
   1. Index nested loop join between Samples and Sequences

Index must be on inner relation thus Samples is the inner relation.

Cost of finding matching tuples in inner relation:

Index is clustered:

1 leaf page + 1 data page = 2

Index is non-clustered:

1leaf page + (50,000 sequences/10,000 samples) = 6

Cost = outer pages + |outer relation| \* (cost of finding matching tuples in inner relation)

= 10,000 + 50,000 \* 2 = 110,000

or

= 10,000 + 50,000 \* 6 = 310,000

**Estimated I/O is 110,000 or 310,000**

* 1. Block nested loop join between Samples and Sequences

For each block of pages bpSq of Sequences, get each page pSm of Samples Sm.

One block of pages bpSq and one page of Sm must fit in main memory.

Main memory has 50 pages.

Samples has 260 data pages (from question 1)

Cost = Sequence pages + sequence pages / |bpSq| \* sample pages

= 10,000 + 10,000 / 50\* 260

= 62,000

**Estimated I/O is 62,000**

* 1. Sort merge join between Samples and Sequences

Cost of sorting Samples:

N = number of pages = 260

B = buffer pages ≈ 50

# runs = N/B =260 / 50 = 6

# passes = 1 + log base B-1 (N/B) = 1.46 ≈ 2

Cost = 2N \* (# passes)

= 2 \* 260 \* 2

= 1040

Cost of sorting Sequences:

N = number of pages = 10,000

B = buffer pages ≈ 50

# runs = N/B = 10,000 / 50 = 200

# passes = 1 + log base B-1 (N/B) = 2.36 ≈ 3

Cost = 2N \* (# passes)

= 2 \* 10,000 \* 3

= 60,000

Total Cost:

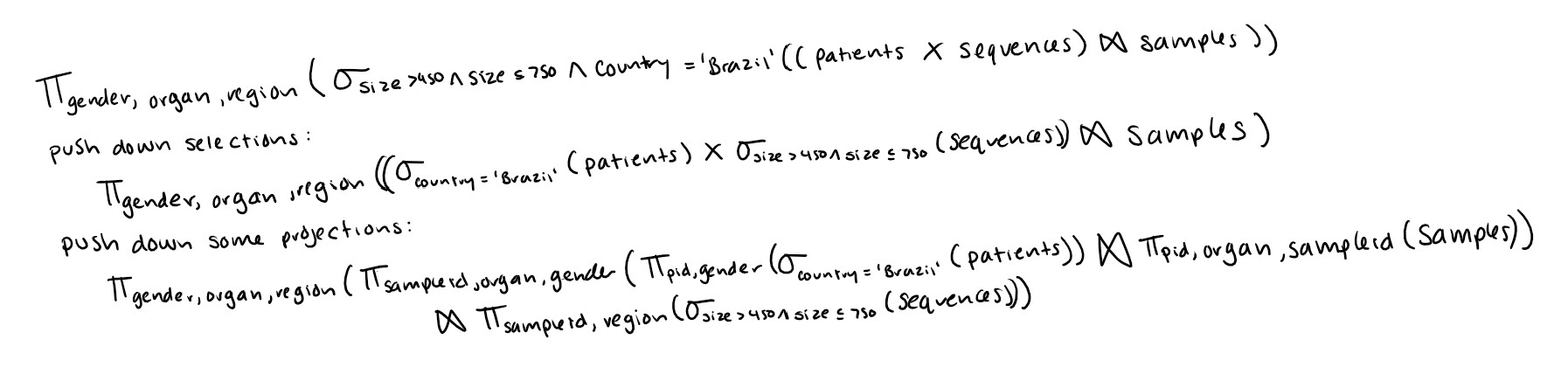
Samples pages + Sequences pages + cost of sorting

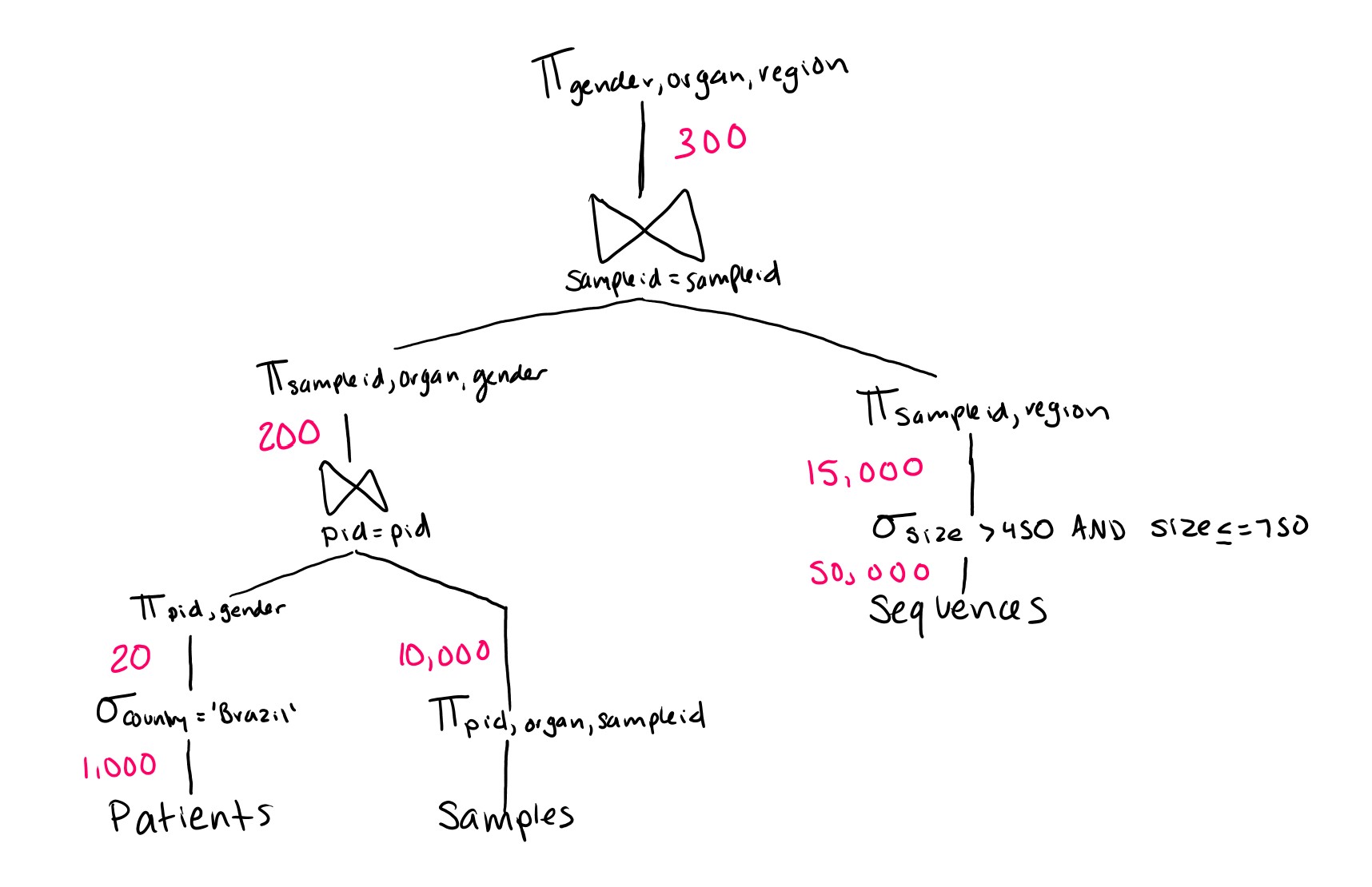
= 260 + 10,000 + 60,000 + 1040

= 71,300

**Estimated I/O is 71,300**

# Exercise 4: Optimization





Assumptions:

20 patients per country

10 samples per patient

5 sequences per sample

30% of sequences between 450 and 750

15,000/50=300

1. Finding an optimal execution plan:

Pass 1: *Determine the best plans for each single relation*

Patients: (non-clustered index on pid)

Index is not useful to find patients based on country, so we need

to scan the entire relation => 1,000 tuples on **15 pages**

Result is 20 tuples stored on <1 page

Do projection on the fly with join later.

Samples: (a non-clustered indirect index on pid)

Do projection on the fly with join later

Sequences: (clustered indirect index on sampleid)

Index is not useful so we need to scan the relation => **10,000 page**s

Result is 15,000 tuples on 3,000 pages

Do projection on the fly with join later

Pass 2: *Determine best 2-relation plans*

Join Samples and Patients:

Merge-sort join-

Cost of sorting Samples:

N = number of pages = 1

B = buffer pages ≈ 50

# runs = N/B =1 / 50 = 1

# passes = 1 + log base B-1 (N/B) = 1

Cost = 2N \* (# passes)

= 2

Cost of sorting Patients:

N = number of pages = 60

B = buffer pages ≈ 50

# runs = N/B = 60 / 50 = 1

# passes = 1 + log base B-1 (N/B) = 2

Cost = 2N \* (# passes)

= 240

Total cost = samples pages + patients pages + sort

= 60 + 1 + 2 + 240

= 303

Samples as outer relation:

Block nested loop join-

For each block of pages of bpS of Samples, get

each page pP of Patients.

One block of pages bpS and one page pP must fit in

Main memory at a time.

=> #pages per bpS = 50 - 1 = 49

Samples has 10,000 tuples on 60 pages

Patients has 1 data page

Cost = samples pages + sample pages /

|bpS|\*patients pages

= 60+ (60 / 50 \* 1)

≈ 61.2

Index nested loop join-

Patients is inner relation, index on pid

Cost of finding matching tuples in inner relation:

1 leaf page + (10,000 samples / 20

patients) = 501

Cost = outer pages + |outer relation| \* (cost

matching)

= 60+ 10,000 \* 501

≈ 5,010,060

Patients as outer relation:

**Block nested loop join-**

For each block of pages bpP of Patients, get

each page pS of Samples.

One block of pages bpP and one page pS must fit in

main memory at a time

=> #pages per bpP = 49

Patients has 20 tuples on 1 data page

|bpP| = 1 / 49 ≈ 1 block of pages

Samples has 60 data pages

Cost = patients pages + patients pages /

|bpP|\*samples pages

= 1 + (1 / 50\*60)

≈ 3

Index nested loop join-

Samples is inner relation, index on pid

Cost of finding matching tuples in inner relation:

1 leaf page + (20 patients / 10,000

samples) ≈ 2

Cost = outer pages + |outer relation| \* (cost

matching)

= 1 + 20 \* 2

= 41

Pass 3: Join Samples/Patients and Sequences:

Merge-sort join-

Cost of sorting Samples/Patients on sampleid:

N = number of pages = 1 (calculated same method)

B = buffer pages ≈ 50

runs = N/B = 1/50 ≈ 1

# passes = 1 + log base B-1 (N/B) ≈ 1

Cost = 2N \* (# passes)

= 2

Cost of sorting Sequences on sampleid:

N = number of pages = 35 (calculated same method)

B = buffer pages ≈ 50

runs = N/B = 35/50 ≈ 1

# passes = 1 + log base B-1 (N/B) ≈ 1

Cost = 2N \* (# passes)

= 70

Total Cost:

Sequences pages + samples/patients pages + cost of sorting

Sequences

= 35 + 1 + 4 + 70

= 108

Samples/Patients as outer relation:

**Block nested loop join-**

For each block of pages of bpSP of

Samples/Patients, get each page pS of

Sequences.

One block of pages bpSP and one page pS must fit in Main memory at a time.

=> #pages per bpSP = 50 - 1 = 49

Samples/Patients on 1 page

Sequences has 35 pages

Cost = samples/patients pages + sample/patients pages /

|bpSP|\*sequences pages

= 1+ (1 / 50 \* 35)

≈ 1.7

Index nested loop join-

Sequences is inner relation, index on sampleid

Cost of finding matching tuples in inner relation:

1 leaf page + (200 sequences / 15000

samples/patients) ≈ 2

Cost = outer pages + |outer relation| \* (cost matching)

= 1 + 200 \* 2

= 401

Sequences as outer relation:

Block nested loop join-

For each block of pages bpS of Sequences, get each

page pSP of patients/samples

One block of pages bpS and one page pSP must fit in main memory at a time

=> #pages per bpS = 50 - 1 = 49

Sequences on 35 pages

Samples/Patients has 1

Cost = sequences pages + sequences pages / |

|bpS|\*samples/patients pages

= 35 + 35 / 50 \* 1

≈ 36

Index nested loop join-

Samples/Patients is inner relation

Cost of finding matching tuples in inner relation:

1 leaf page + (15000 / 200) = 76

Cost = outer pages + |outer relation| \* (cost matching)

= 35 + 15,000 \* 76

= 1140035

Flow between operators:

Next page

