Namdar Kabolinejad
260893536
COMP 421 Assignmnet 2 (Winter 2022)

1.
A)

Business:

Business Record Size =
Bid + ⅔ (btype) + ⅔ (bname) + ⅔ (bcity) + (bprovince) + (bstartdate) =
10 + ⅔ * 30 + ⅔ * 60 + ⅔ * 30 + ⅔ * 60 + 2 + 10 = 142 Bytes

Business Number of Pages $= \frac{Size\ of\ Table}{Size\ of\ a\ Page}$

$= \frac{\#\ Businesses * Size\ of\ each\ Business}{Size\ of\ a\ Single\ Page * Fill\ Factor}$

$= \frac{20000 * 142}{4000 * 0.75} =$ **947** Pages

Users:

Users Record Size =
⅔ (uemail) + ⅔ (uname) + (udateof birth) + (ujoindate) + (uprovince) =
⅔ * 45 + ⅔ * 45 + 10 + 10 + 2 = 82 Bytes

Users Number of Pages $= \frac{Size\ of\ Table}{Size\ of\ a\ Page}$

$= \frac{\#\ Users\ * Size\ of\ each\ User}{Size\ of\ a\ Single\ Page * Fill\ Factor}$

$= \frac{90000 * 82}{4000 * 0.75} =$ **2460** Pages

Reviews:

Reviews Record Size =
(reviewid) + (bid) + ⅔*(uemail) + (rstars) + ⅔*(rtext) + (rdate)=
8 + 10 + ⅔*45 + 1 + ⅔*600 + 10 = 459 Bytes

Review Number of Pages $= \frac{Size\ of\ Table}{Size\ of\ a\ Page}$

$= \frac{\#\ Users * \#\ Reviews\ Per\ User\ * Size\ of\ each\ Review}{Size\ of\ a\ Single\ Page * Fill\ Factor}$

$= \frac{90000 * 20 * 459}{4000 * 0.75} =$ **275400** Pages

B)

Assumptions:

- Bid is unique, have 20,000 bids
- Reviews are uniformly distributed for the Business

I.

We are indexing on bid, so we need to read at least 1 leaf page to do the point query on bid. After finding the correct data pages using the leaf page and the bid, we need to do a search on the rstars. Since the indexing is unclustered we might need to read multiple pages to get the rstars. Here's the math:

Size of each leaf pages entry = (bid) + (rid) = 10 + 10 = 20 Bytes

Data entries per leaf page =
size of each page/size of each entry =
(4000 * 0.75) / 20 = 150 data enteries / leaf page

Number of leaf pages =
# of Reviews / # entries per leaf page =
90000 * 20 / 150 = 12000 leaf pages

The average number of reviews per bid =
Total # of review / Total # of Business =
90000 * 20 / 20000 = 90 reviews / busines on average

From the calculations above, we can fit about 150 entries on each leaf page and have 90 reviews per business. So we can get the leaf page only by 1 I/O since the number of reviews is smaller than the size of the leaf page. But after getting the leaf page we need to get the data pages. Since the index is unclustered, we need to do as many I/O to retrieve the data pages that match the rstars constraint,vwe have to do an I/O for each review to make sure we check all of the rstars. We can use the distribution of rstars to figure out how many data pages we would need to read for each N.

I/O cost = 1 (leaf page) + # of reviews (data pages)

# of reviews = 90 * total distribution
= 90 * 0.15 = **13.5** for N = 1
= 90 * 0.40 = **36** for N = 2
= 90 * 0.65 = **58.5** for N = 3
= 90 * 0.90 = **81** for N = 4
= 90 * 1.00 = **90** for N = 5

II.
When we have type II indexing, we essentially have one bid and a bunch of rids for each bid. Since we are doing the indexing for the reviews and we know from above that we will have on average 90 reviews / per business:

Size of each leaf pages entry =
(bid) + (rid)*(number of reviews per business) =
10 + 10*90 = 910 Bytes

Data entries per leaf page =
size of each page/size of each entry =
(4000 * 0.75) / 910 = 3.3 data enteries / leaf page

Number of leaf pages =
# of Reviews / # entries per leaf page =
90000 * 20 / 3.3 = 545454 leaf pages

Records per data page =
size of data page / size of record =
4000 * 0.75 / 459 = 6.5 records / data page

We can store 3.3 entries per leaf page, so to retrieve a bid we only need to read 1 leaf page. The index is also clustered, so the data pages will be sorted with each page having about 6.5 Review records per data page. So apart from the 1 leaf page, we need to read in the data pages but given that they are clustered we need to read in a limited number of data pages, specifically 6.5 times less than the number of reviews for each bid, since each data page contains 6.5 review records for a given bid.

I/O cost = 1 (leaf page) + # of reviews (data pages)

# of reviews = 90 * total distribution / 6.5
= 90 * 0.15 / 6.5 = **2.07** for N = 1
= 90 * 0.40 / 6.5 = **5.53** for N = 2
= 90 * 0.65 / 6.5 = **9** for N = 3
= 90 * 0.90 / 6.5 = **12.46** for N = 4
= 90 * 1.00 / 6.5 = **13.84** for N = 5

III.
We are doing an unclustered type II on (bid, rstars). So for each pair of (bid, rstars) we would have to hold a set of rids. So the leaf pages would look like ((bid = 1, rstars = 2), [rid1, rid2, ...]), please note this is what I'm getting from slide 27 of lec 14 (only refrance to multi-attribute indexing) and an ed question I asked. We know from before that each business has 90 review on average, and we can assume that 0.20 percent of these reviews are for each of the 1-5 starts. That means that each pair of (bid, rstars) will have 90*0.20 = 18 reviews on average.

Size of each leaf pages entry =
(bid) + (rstars) + (rid)*(number of reviews per for each (bid, rstars) pair) =
10 + 1 + 10*18 = 191 Bytes

Data entries per leaf page =
size of each page/size of each entry =
(4000 * 0.75) / 191 = 15.7 data enteries / leaf page

We can get the leaf page only by 1 I/O. But after getting the leaf page we need to get the data pages. Since the index is unclustered, we have to do as we did in part I.

I/O cost = 1 (leaf page) + # of reviews (data pages)

# of reviews = 90 * total distribution
= 90 * 0.15 = **13.5** for N = 1
= 90 * 0.40 = **36** for N = 2
= 90 * 0.65 = **58.5** for N = 3
= 90 * 0.90 = **81** for N = 4
= 90 * 1.00 = **90** for N = 5

3.

A)

Approach:

First we will do the individual table selections combined with projection to only keep the data we need. Then we will continue to do the joins.

$$\rho(B1,\sigma_{B.btype='restaurants'}(B))$$

$$\rho(B2,\Pi_{B1.bid \wedge B1.bname \wedge B1.bprovince}(B1))$$

$$\rho(R1,\Pi_{R.bid \wedge R.uemail \wedge R.rstars \wedge R.rdate}(R))$$

$$\rho(T1,(R1 \bowtie B2))$$

$$\rho(T2,\Pi_{T2.bname \wedge T2.bprovince \wedge T2.uemail \wedge T2.rstars \wedge T2.rdate}(T1))$$

$$\rho(U1,\sigma_{U.udateofbirth \geq '1990-01-01' \wedge U.udateofbirth \geq '1990-12-31'}(U))$$

$$\rho(U2,\Pi_{U1.uemail}(U1))$$

$$\rho(T,(T2 \bowtie U2))$$

T holds final results

B)

I have two secnarios for optimizing this, the second scenario build on the first scenario and makes it more optimal:

1.

We initially have to read in all the records of the Business table to do the selection on btype, we can pipeline the results to do a projection on them as well. This will cost us 947 pages, and since we are pipelining for the selection it will not cost any extra I/Os. Since we have 100 btype, and we are assuming uniform distribution, we can assume that 1% of the records wee remain after the selection so 0.01 * 20,000 = 200 records will be left with attributes (bid, bname, bprovince).
Size of Business records left = (bid) + (bname) + (bprovince) = 10 + ⅔* (60) + 2 = 52 Bytes
Size of total Business records left = 52*200 = 10,400 Bytes
Number of pages needed to store the records = 10,400 / 4000*0.75 = 3.5 Pages

Since the records we get after doing a selection and projection on the Business table we can store them in about 3.5 pages in the buffer.
We will then continue to do a Block Nested Join between Business and Reviews, with Business as the other block since it's already in the buffers.
The cost of the join will be equal to the number of pages we have to read in which is **275400 pages**.

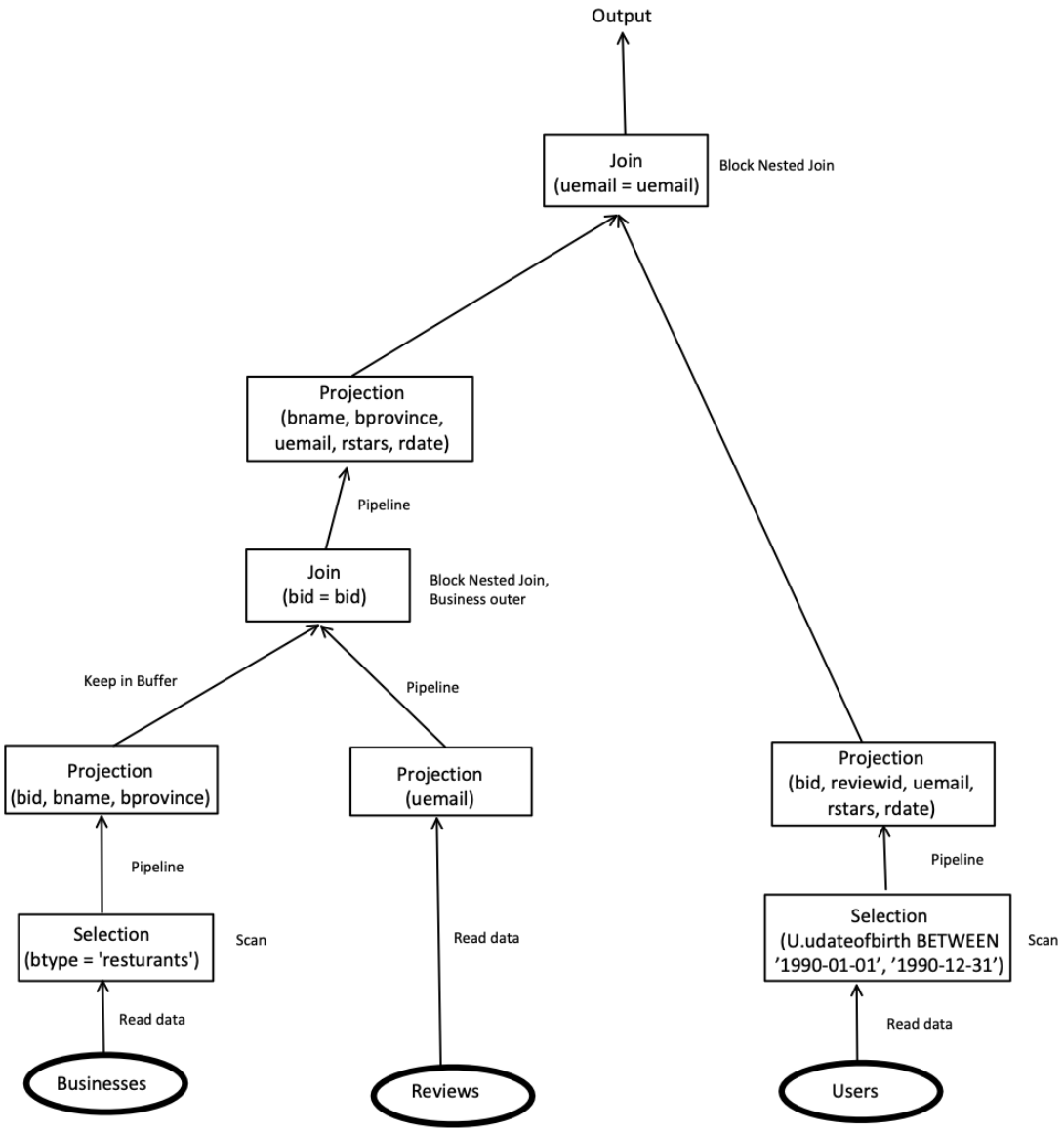
So until this point we have done 947 + 275400 = 276347 I/Os
After doing the join we can pipeline the results to do a projection with no costs. At this point we will have 1% of the review records = 0.01 * 90000 * 20 = 18000 records with attributes (bname,bprovince,uemail,rstars,rdate).
Size of joined records left = (bname) + (bprovince) + (uemail) + (rstars) + (rdate) = ⅔* (60) + 2 + ⅔*45 + 1 + 10 = 83 Bytes
Size of total Business records left = 18000 * 82 = 1,476,000 Bytes
Number of pages needed to store the records = 1,476,000 / 4000*0.75 = 277 Pages

We will not be able to hold the results after the join the buffers and will need to do a write for **277 pages** as temporaray table T.

We will also have to read in all the pagess for the Users do a secltion and pipeline to a projection, this will cost us **2460 page** reads. From all the Users we read in 1825 records will be left after the selction with only the uemail attribute.
Size of User records left = (uemail) = ⅔* (45) = 30 Bytes
Number of pages needed to store the records = 1825 * 30 / 4000*0.75 = 18 Pages

We can store all these 18 pages in the buffer and continye to do the second Block Nested Join with the table T. This join will cost us **277 pages** in.

Total cost =
947 (Selection on B) + 275400 (Join B with R as T) + 277 (Write T to disk) + 2460 (Selection on U) + 277 (Join U and T) = **279361 I/Os in total**

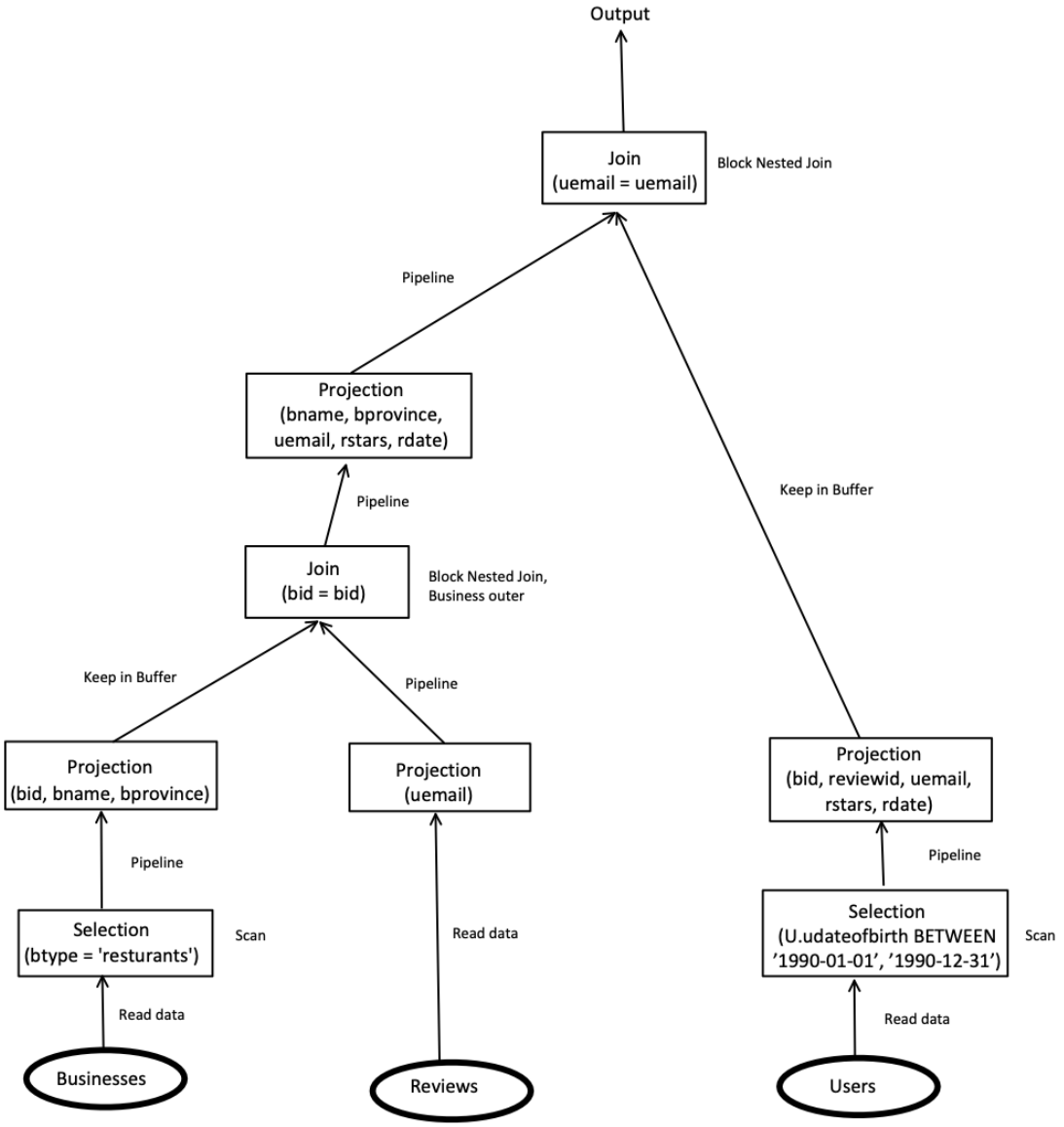


2.

In the second senrario (which I'm not sure would be possible in the real world), we would:

1. Selection + pipeline projection on Business = **947** I/Os → store 3 pages of results in buffer
2. Selection + pipeline projection on Users = **2460** I/Os → store 18 pages of results in buffer
3. Nested Block Join Business and Reviews, with Business as the outer Block (already in buffer) = **275400** I/Os → pipeline the results to do a Nested Block Join with Users as the outer block (already in buffer) and out put the results

This would result in a total of **278807 I/Os** and we would avoid the need to write the results of the first join to the disc and read it back again.



4.

We first do a selection for the Review table and pipeline the results to do a projections. For this we have to read in all the Review pages so **275400 I/Os**. After doing the selection we will be left with with about 1/(12*4) records with attributes (bid, uemail, rstars), since we have the Review data for 4 years (2018-2021) and rdate >= 2021-12-01 would result in about a month of reviews from 4 years of reviews, so the reduction factor is 1/48.

Size of records left = (bid) + (uemail) + (rstars) = 10 + ⅔*45 + 1 = 41 Bytes
Number of records left after selection = (90000 * 20)/48 = 37,500 Records
Size of total Review records left = 37,500 * 41 = 1,537,500 Bytes
Number of pages needed to store the records = 1,537,500 / 4000*0.75 = 288 Pages

We can't keep all these pages in the memeory can will need to do a write to disc, this will cost us **288 I/Os**. After this we need to do a join between Reviews and Users based on

uemail. We have 90,000 Users, each with about 20 Reviews, so we son't be able to store the Users in the Buffer either and the result from the join will be massive. Since we have 37,500 review records after the selection and each Users has about 20 reviews, when we do a join on uemail, we coul expect somthing around 37,500*20 = 750,000 records with (bid, uemail, rstars, uprovince), we also won't be able to store these in the buffers.

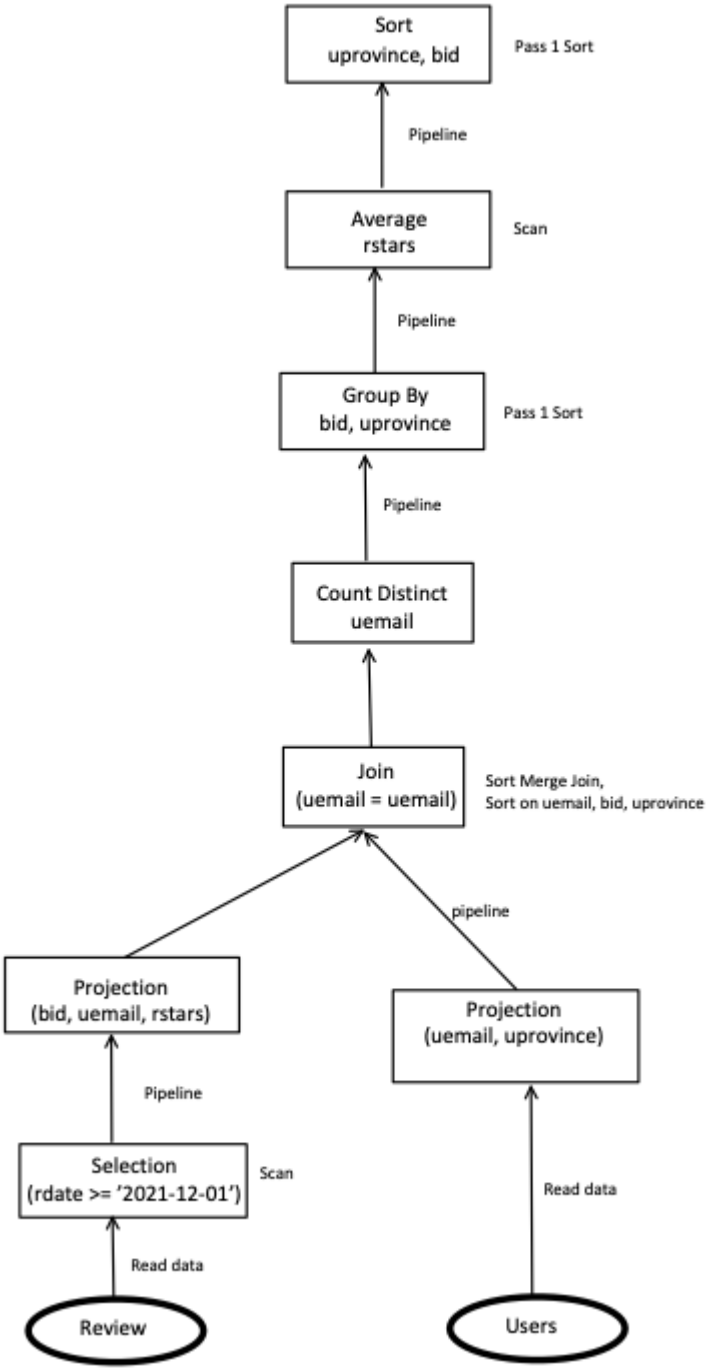
Size of records left = (bid) + (uemail) + (rstars) + (uprovince)= 10 + ⅔*45 + 1 + 2 = 43 Bytes
Number of records left after selection = 37,500*20 = 750,000 Records
Size of total Review records left = 750,000 * 43 = 32,250,000 Bytes
Number of pages needed to store the records = 32,250,000 / 4000*0.75 = 6047 Pages

I have decided to use a Sort Merge Join to join the Users and the Reviews, this is because after doing a join we could use the fact the the uemails are sorted to count them. I have also decided to sort the values based on bid and uprovince as it will speed up our work later on. So we will first do a sort and merge the results.

The cost of the merge sort join depends on the height of the tree which we have not been given, but on we would have to do about 5 passes to complete it. So the sorting our data would cost approximatly 10 * 6047 = **60470 I/Os**

After the merge sort we will still have to write to disc since we have too many records too keep resulting in **6047 I/Os**. After that we will have to sort the records based on uid, uprovince inorder to do the grouping, in the mean while we would count the number of distinct uemails (ther are sorted at this point) this will cost us **60470 I/Os**, as explained above. After doing the grouping we will do a scan and average for rstars we finally pipeline the results to do a final sort on uprovince,bid and output the results.

I know the I/Os are too high, but this is the best I could come up with, it's been a very busy few days. Tree is on the next page.


Sort
uprovince, bid
Pass 1 Sort
Pipeline
Average
rstars
Scan
Pipeline
Group By
bid, uprovince
Pass 1 Sort
Pipeline
Count Distinct
uemail
Join
(uemail = uemail)
Sort Merge Join,
Sort on uemail, bid, uprovince
pipeline
Projection
(bid, uemail, rstars)
Projection
(uemail, uprovince)
Pipeline
Selection
(rdate >= '2021-12-01')
Scan
Read data
Read data
Review
Users