CS 186 Fall 2024	Introduction to Database Systems Alvin Cheung	DIS 6

1 Assorted Joins

- Companies: (company_id, industry, ipo_date)
- NYSE: (company_id, date, trade, quantity)

We have 20 pages of memory, and we want to join two tables Companies and NYSE on C.company_id = N.company_id. Attribute company_id is the primary key for Companies. For every tuple in Companies, assume there are 4 matching tuples in NYSE.

NYSE contains [N] = 100 pages, NYSE holds p_N = 100 tuples per page.

Companies contains [C] = 50 pages, C holds p_C = 50 tuples per page.

There are alternative 3 unclustered B+ tree indexes of height 1 on C.company_id and N.company_id. Throughout the problem, do not assume any caching of index nodes.

(a) How many disk I/Os are needed to perform a simple nested loop join?

[C] + p_c*[C]*[N] = 50 + 50*50*100 = 250050

(b) How many disk I/Os are needed to perform a block nested loop join?

With N as the outer relation: [N] + ceil([N]/B-2)*[C] = 100 + ceil(100/18)* 50 = 400 I/Os
With C as the outer relation: [C] + ceil([C]/B-2)*[N] = 50 + ceil(50/18)* 100 = 350 I/Os
Most of the time, putting the smaller relation on the outside leads to a better I/O cost for BNLJ, but this is not always true. Therefore, we try both options, computing the I/O cost with N on the outside and with C on the outside and taking the smaller of the two.
I/O cost for BNLJ: 350 I/Os

(c) How many disk I/Os are needed to perform an index nested loop join?

[C] + [C] * p_c * (cost to find matching NYSE tuples) = 50 + 50 * 50 * (2 + 4) = 15,050
You should check which relation on the outside results in a lower I/O cost. See 1(b) for more description.

(d) For this part only, assume the index on NYSE.company_id is clustered. What is the cost of an index nested loop join using companies as the outer relation?

[C] + [C]*p_c * (cost to find matching NYSE tuples) = 50 + 50 * 50 * (2 + ceil(4/100)) = 7,550 I/Os
You should check which relation on the outside results in a lower I/O cost. See 1(b) for more description.

(e) In the average case, how many disk I/Os are needed to perform a sort merge join without optimization? If we can perform the sort merge join optimization, how many disk I/Os are needed with optimization?

Without optimization: 750 I/Os
With optimization: 450 I/Os

See discussion slides for step-by-step visualization of both the unoptimized and optimized solution.

2 Grace Hash Join

We have 2 tables – Catalog and Transactions.

Catalog has a total of 100 pages and 20 tuples per page. Transactions has a total of 50 pages and 50 tuples per page. Assume the hash functions uniformly distribute the data for both tables.

(a) If we had 10 buffer pages, how many partitioning phases would we require for grace hash join? Consider which table we should build the hash table in the probing phase on.

T is smaller, so we need its partitions to be at most B − 2 = 8 pages. After 1 partitioning pass, we have partitions of size 6, which is ≤ 8 so we only need 1 partitioning pass.

(b) What is the I/O cost for the grace hash join then?

We need 1 partitioning pass.
Partitioning phase:
ceil([C]/(B - 1)) = 12 pages per partition for C, 12(9) pages in total after partitioning
ceil([T]/(B - 1)) = 6 pages per partition for T, 6(9) pages in total after partitioning
Partitioning IOs: 100 I/Os to read from Catalog + 12(9) to write for Catalog + 50 I/Os to read from Transactions + 6(9) to write for Transactions = 312 I/Os
Probing phase: 12(9) + 6(9) = 162 I/Os to read from Catalog and Transactions
Total: 312 + 162 = 474 I/Os

(c) For the above question, if we only had 8 buffer pages, how many partitioning phases would there be?

T is smaller, so we need its partitions to be at most B - 2 = 6 pages. After 1 partitioning pass, we have partitions of size 8, which is too big to fit in B-2 buffer pages. We need a second partitioning pass. 8 / 7 = 1.1 → 2 pages, which is small enough to fit in B-2 buffer pages. Therefore, we need 2 passes in total.

(d) What will be the I/O cost?

Partitioning phase:
ceil([C]/(B - 1)) = 15 pages per partition for C
ceil([T]/(B - 1)) = 8 pages per partition for T
ceil([C]/(B - 1)) = 3 pages per partition for second pass for C
ceil([T]/(B - 1)) = 2 pages per partition for second pass for T
Partitioning IOs: [100 + 50] (1st read) + [15(7) + 8(7)] (1st write) + [15(7) + 8(7)] (2nd read) + [3(49) + 2(49)] (2nd write) = 717 I/Os
Build and Probe Phase: 3(49) + 2(49) = 245 IOs
Total: 717 + 245 = 962 I/Os

3 Relational Algebra

Consider the schema:

- `Songs(SONG_ID, song_name, album_id, weeks_in_top_40)`
- `Artists(ARTIST_ID, artist_name, first_yr_active)`
- `Albums(ALBUM_ID, album_name, artist_id, yr_released, genre)`

Write relational algebra expressions for the following queries:

(a) Find the names of the artists who have albums with a genre of either 'pop' or 'rock'.

$\pi_{artists.artist\_name} (\sigma_{albums.genre = \text{'pop'} \vee albums.genre = \text{'rock'}} (Artists \bowtie Albums))$

(b) Find the names of the artists who have albums of genre 'pop' and 'rock'.

Solution 1:

$$\pi_{artists.artist\_name}((\sigma_{albums.genre = 'pop'} Albums) \bowtie Artists) \cap$$
$$\pi_{artists.artist\_name}((\sigma_{albums.genre = 'rock'} Albums) \bowtie Artists)$$

Solution 2:

$$\pi_{artist\_name}(\; Artists \bowtie (\; \pi_{artist\_id}(\sigma_{genre = 'pop'}(Albums)) \cap \pi_{artist\_id}(\sigma_{genre = 'rock'}(Albums)) \;) \;)$$

(c) Find the id of the artists who have albums of genre 'pop' or have spent over 10 weeks in the top 40.

$$\pi_{artist\_id}(\sigma_{genre = 'pop'}(Albums)) \cup$$
$$\pi_{artist\_id}(\sigma_{weeks\_in\_top\_40 > 10}(Albums \bowtie Songs))$$

(d) Find the names of the artists who do not have any albums.

$$\pi_{artists.artist\_name} (Artists \bowtie ((\pi_{artists.artist\_id} Artists) - (\pi_{albums.artist\_id} Albums)))$$