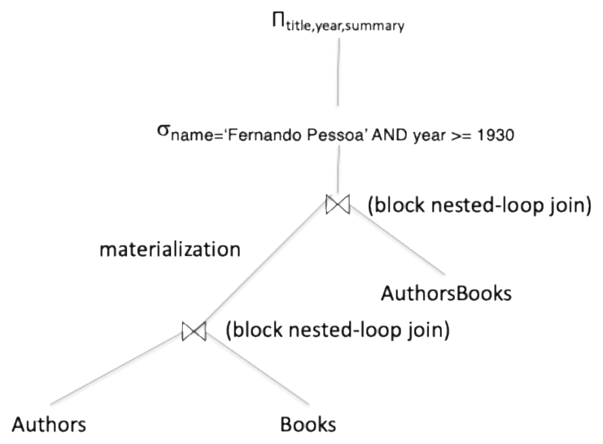
Solutions for the 1st SBD test 2023/24

It is easy to determine that the tables occupy the following number of disk blocks:

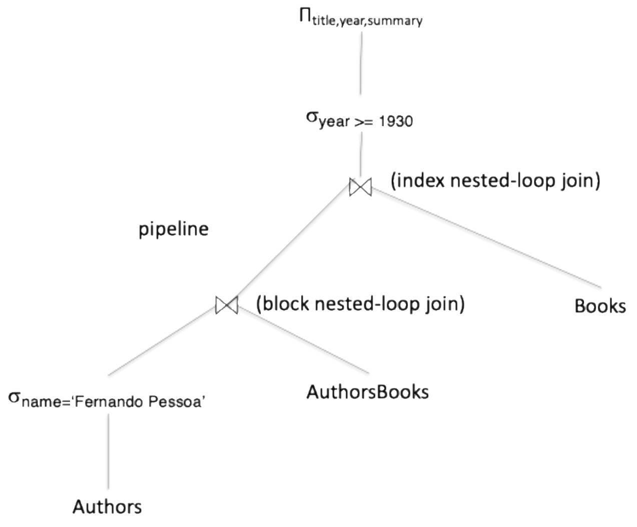
- Authors: 12,500 blocks
- AuthorsBooks: 62,500 blocks
- Books: 250,000 vlocks
- BooksGenres: 7,813 blocks
- Genres: 16 blocks
- Copies: 78,125 blocks
-

1a) The first (very bad plan) is:



Since there are no attributes in common between tables Authors and Books, the join operation is in fact a cartesian product. The result has 100,000 * 1,000,000 tuples, i.e. 100,000,000,000 tuples.

A much better plan would be:



Let's try to see why. The selection on Authors will return just one tuple. The block nested-loop join can be efficiently implemented just by using the single tuple of the author to join with authors books, just requiring one scan of the AuthorsBook (a index nested-loop join would certainly be better).

For the index nested-loop join of the final join, let us assume that N is the number of books written by Fernando Pessoa (which are just a few), then the cost is:

$b_r (t_T + t_S) + N * c$

since the left-hand expression is pipelined then the cost is simplified to

$N * (5+1) * (t_T + t_S)$ = N*6*11* t_T = 66*N* t_T

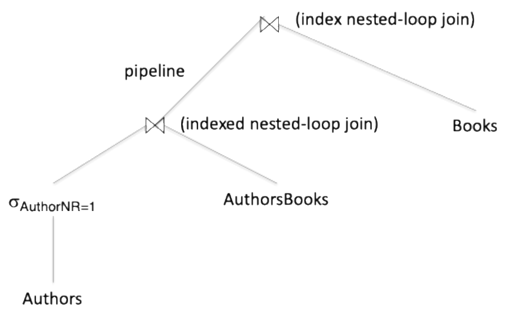
Since this is a secondary index, equality on key with height 4 (the height is 5 since 4 is not enough in the worst-case : 2*50*50*50 = 125,000 search-keys while with height 5 the number of search keys that might be stored are 2*50*50*50*50 – the factor 2 in the root level is due to the fact that the minimum number of children of the root node is 2). Note that moving down the selection in year would prevent the use of a indexed nested-loop join and thus required at list a linear scan with cost 1* t_S + 250,000*t_T = 250,010*t_T, thus for N < 250,010*t_T / 66* t_T = 3787 is preferable to use the index nested-loop join (the numbers are slightly different from the ones derived in the class because we assumed the height of the B+-tree index to be 4 instead of 5 due to use 50 also for the root node).

1b) An appropriate query that makes use of multi-clustered table file organization can be:

SELECT *
FROM Books NATURAL INNER JOIN Copies

By storing a book with its copies, this would avoid extra seeks and block transfers to get the copies of each book since they would be stored together, and index lookups would not be necessary.

1c) Yes, the query can benefit from using the indexes. For instance a plan could be:



The selection returns a single tuple, and one can use the composite index on the table authors to retrieve the author 1 books' identifiers (in average they are 40), and then finally) we can use the primary key index to return the data of the books (40 books).

The first index nested-loop join would cost $(h_i + n + l) * (t_T + t_S) = (5 + 40 + 1) * (t_T + t_S)$ – we need at most an extra 1 block for the leaf storing the 40 answers. The cost for retrieving the 40 books is simply $40*(h_i +1) * (t_T + t_S) = 40 * (5 +1) * (t_T + t_S)$, where books B+-tree has also height 5. So the total is $246*(t_T + t_S) = 246 * 11 * t_T = 2706 * t_T$

This would require a very small number of disk seeks and accesses. Note that even with best case situation of block-nested loop joins or hash joins one would have to transfer all the blocks of authors, authorbooks and books which are more than 300,000 blocks.

1d) There are two genres resulting from the selection condition. However, the corresponding genreID cannot be used to lookup the BooksGenres composite index on attributes (bookID,genreID) since genreID is not the first attribute in the index.

Without using an index, since the smaller relation fits into memory, one could use an ordinary block nested-loop join with the cost of $2 * t_S +(16 + 7813)* t_T$ which reducing all to transfer times is $20 * t_S +(16 + 7813)* t_T = 7849* t_T$

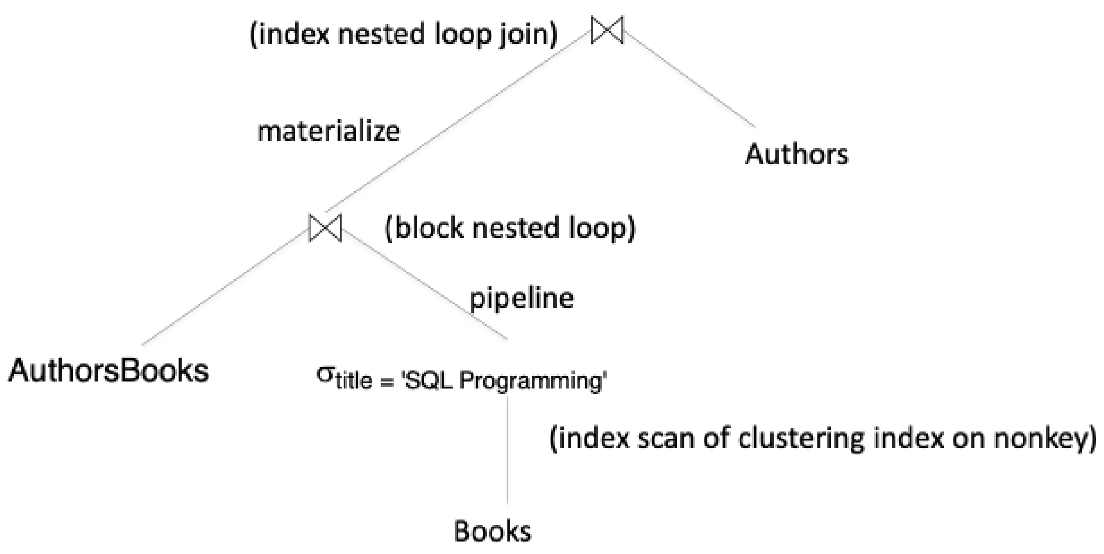
The use of a bitmap index in BooksGenres on attribute GenreID would improve a lot the cost.

So, we would need 1000 bitmaps, one for each genre, with 2,000,000 bits each to index the BooksGenres table. Thus, each bitmap would have the size

2,000,000 / 8 / 8192 blocks = 30,51 block, rounding up, 32 blocks. Thus, one would need 32000 blocks for the index. To do the counting we just need to read the genres table, $1 * t_S + 16* t_T$, and $2*(1 * t_S + 32* t_T)$ for reading the bitmaps, which is much less than the previous approaches.

Note that this approach would need an extra indirection if one would like to obtain the books (by joining with the books table)

1e)



1-Selection	$4*(t_T + t_S) + t_S + 3* t_T$ (cluster index equality on key) In fact, as discussed, the 10 answers could need 4 blocks instead of 3 (1 tuple in 1 block, 4 tuples in another, 4 tuples in another, and 1 tuple in other) resulting in the expression: $4*(t_T + t_S) + t_S + 4* t_T$
2-BNL	Since the results of the selection fit into memory, one can use as the inner relation the 10 tuples resulting from the selection occupying 3 blocks. Thus, the cost is just: $1*t_S + 62{,}500* t_T$
3-Materialization	40 tuples are returned occupying 11 blocks (2048+128)*40/8192, thus requiring $1*t_S + 11* t_T$
4 - INL	$11*(t_T + t_S) + 40* (3+1)*(t_T + t_S)$