Some corrections for the recap

- **This document is mostly for the first group since we simplified some of the problems (and haven‘t done the last task which we did with other groups)**
- **I realized (too late) that it would be better to solve the tasks how Iztok would solve them. Sorry for that =)**
- **So here are now two evaluation exercises done in a way so that the index is calculated with extra steps**
- **Last exercise added is calculating the B+ index in a way where you do not use rough estimations (do that only if an exercise specifies that – for example in 3rd exercise on 2nd exam in 21/22)**

- **And last thing: don‘t worry for the HW you can solve the evaluation task the way we did it or in the way presented here, both will be correct; but on the exam, be sure to use Iztok‘s way ☺**

Bank – Exercise 3

1 page on disk = 8K

|Client| = 100.000 records, 100 bytes, 80 records/page, 1250 pages

|Account| = 120.000 records, 80 bytes, 100 records/page, 1200 pages

|Deposit| = 1.000.000 records, 120 bytes, 66 records/page, 15151 pages

|Withdrawal| = 1.500.000 records, 120 bytes, 66 records/page, 22727 pages

What is the optimal query execution plan if the system has hash indexes and B+ trees. Demonstrate the overall procedure and the algorithms used to perform the relational algebra operations. This time assume that selection „Select [Amount>1000] (Deposit)“ matches 30% of records.

Join(Select [amount>1000] (Deposit), Account, Deposit.rid=Account.rid)

Bank – Exercise 3

-> The tree is already optimised.
-> Scan of Deposit through index -> record in TMP:
We would like to use the B+ index on the amount attribute, because it is efficient for the selector. Let‘s calculate it:
|Data entry| = 16B
Page capacity / #Data_entries = 8000B / 16 = 500 (8000B is given in the task: „**1 page on disk = 8K“**)
Size of level of the leaves = 1‘000‘000 (records Deposit) / 500 = 200
Size of the B+ index = 1,5 * (2000 + 1) = 3002 pages (from the formula – check the reuploaded tutorials on eval. Opt.)
Assume that the index is unconnected (If it is connected -> calculate 0,3 * 3002):
Selection match (30%) * (Pages for Deposit + 3002) = 0,3 * (15151 + 3002) = 5446 pages => store into TMP

-> On TMP.rid and Account.rid we want the scatter index, which is the most appropriate for the equivalence. On Account.rid we assume that we already have it before running the query. On TMP, we create it before the contact. To create the index on TMP we do the following calculation:
1,25 * (5446 + 1) = 6809 pages (from the formula – check the reuploaded tutorials on eval. Opt.)
We now have 6809 pages used for TMP with hash index on it (TMP.rid). We can now use a hash join

-> Use a hash join 3*(M+N) IF B > √N:
Number of pages in memory B must be greater than the square root of N (N = number of pages of the smaller relation)
B > √1200 ... so B must be at least 35 pages. (Since it‘s not given we can assume that it is at least as big)
1st phase (construction of partitions): 2*(M+N) = 2*(6809+1200) = 2*8009 = 16‘018 pages
2nd phase (matching): M+N = 8009 pages
-> Total cost: 5446 + 6809 + 16‘018 + 8009 = 36‘282 pages

Videocenter – Exercise 3

1 page on the disk = 8K; Buffer can store 1000 pages
|Films| = 96.000 records, 80 records/page, 1.200 pages
|Members| = 120.000 records, 100 records/page, 1.250 pages
|Rental| = 800.000 records, 80 records/page, 10.000 pages
|Employees| = 400 records, 100 records/page, 4 pages

Assume that no index exists and that all selections are made after the join. How many pages does the following query read per each possible join?

SELECT fid, title FROM Rental R, Members M
WHERE R.date > '1/1/2012‘
AND M.surname = „Novak“
AND Rental.mid = M.mid

(Including this exercise in this powerpoint, just because I change the data to make more sense when calculating)

Videocenter – Exercise 3

Projection fid (on the fly)
|
Selection surname = novak (on the fly)
|
Selection date > 1.1.2012 (on the fly)
|
Join ⋈ (we need to read from the disk, below are the calculations)
| |
Members Rental

Simple nested loops join:

M + M*prm*N = |Rental|+|Rental|*rppRental*|Members| = 10.000 + 10.000 * 800.000 * 1.250 = 10.000.000.010.000 pages

Block nested loops join:

M+M*N = |Rental| + |Rental|*|Members| = 10.000 + 10.000 * 1250 = 12.510.000 pages

Merge-sort join:

2*|Rental|* (1+logB-1(|Rental|/B)) + 2*|Members|* (1+logB-1(|Members|/B)) + |Rental| +|Members|=
2*10.000*(1+log999 (10.000/1.000)) + 2*1.250* (1+log999 (1.250/1.000)) + 10.000 + 1.250 =
20.000*(1+(log(10)/log(999))) + 2.500 * (1+(log(1,25)/log(999))) + 11.250 =
20.000*1,3 + 2.500*0,3 + 11.250 = 26.000 + 750 + 11.250 = 38.000 pages

We can see that the best option is to merge-sort join.
Selections play no role in this case, as they are done by join, i.e. on the fly. However, we do not have indexes, so we cannot perform other join.

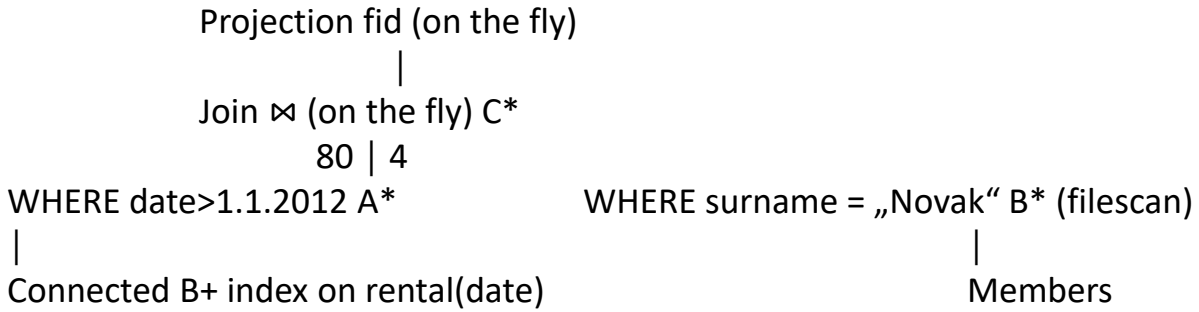
Videocenter – Exercise 4

1 page on the disk = 8K; Buffer can store 1000 pages
|Films| = 96.000 records, 80 records/page, 1.200 pages
|Members| = 120.000 records, 100 records/page, 1.250 pages
|Rental| = 800.000 records, 80 records/page, 10.000 pages
|Employees| = 400 records, 100 records/page, 4 pages

Analyse the optimal execution plan of the query from the previous task. You can use any join. Consider the following:

- we have connected B+ index on the attribute date of the relation Rental
- we have an unconnected hash index on the attribute mid of the relation Members
- 40% of all the rents happens after year 2011
- all the rents are randomly distributed among the members

Videocenter – Exercise 4



A* |Data entry| = 16B
Page capacity / #Data_entries = 8.000B / 16 = 500 (8.000B is given in the task: „**1 page on disk = 8K“**)
Size of level of the leaves = 800.000 (records Rental) / 500 = 1.600
Size of the B+ index = 1,5 * (1.600 + 1) = 2402 pages (from the formula – check the reuploaded tutorials on eval. Opt.)
Since it‘s connected -> calculate 0,4 * 2402 = 961 pages (0,4 comes from selection match of 40%) => Store 961 to T1

B* (We don't have an index on the surname attribute so that we can read all 1250 pages. Let's say 1% of the records correspond to our selection. 1% of 120.000 records is 1.200 records... 1.200/100rpp = 12 pages, which we store in T2)
C* (Since we have a hash index on one side of the contact, we can use an index nested loops join)
N + M*prn*1,2 = |T2|+|T1|* rppT2*1,2 = 12 + 961*100*1,2 = 115.332 pages
Total cost = 961 + 1250 + 12 + 115.332 = 117.555 pages

From the previous exercise that the fastest contact was with sort-merge, so let's try that again:
2*|T1|* (1+logB-1(|T1|/B)) + 2*|T2|* (1+logB-1(|T2|/B)) + |T1| +|T2|= 2.919 pages (logs round to 0)
Total cost = 961 + 1.250 + 12 + 2.919 = 5142 pages

B+ index

The following schema is given:

Cd(cid, author, title, abbreviation, year)
Member (lid, name, surname, address, email, phone)
Loan (iid, cid, lid, zid, date, loan_term, comentary)
Employees (zid, name, surname, address, email, phone)

Additional informations:

1 page on the disk = 8K
|Cd| = 300.000 records, 400 bytes, 20 records/page, 15.000 pages
|Member| = 10.000 records, 200 bytes, 40 records/ page, 250 pages
|Loan| = 300.000 records, 100 bytes, 80 records/ page, 3750 pages
|Employees| = 100 records, 200 bytes, 40 records/ page, 3 pages

Calculate the size of the B + index on the attribute Cd.title. Don‘t use rough estimations. Write down all the assumptions.

The following schema is given:

Cd(cid, author, title, abbreviation, year)
Member (lid, name, surname, address, email, phone)
Loan (iid, cid, lid, zid, date, loan_term, comentary)
Employees (zid, name, surname, address, email, phone)

Additional informations:

1 page on the disk = 8K
|Cd| = 300.000 records, 400 bytes, 20 records/page, 15.000 pages
|Member| = 10.000 records, 200 bytes, 40 records/ page, 250 pages
|Loan| = 300.000 records, 100 bytes, 80 records/ page, 3750 pages
|Employees| = 100 records, 200 bytes, 40 records/ page, 3 pages

Calculate the size of the B + index on the attribute Cd.title. Don‘t use rough estimations. Write down all the assumptions.

|page|= 8K # **size of 1 page on the disk 8000 B (from the task)**
|rid| = 16B # **size of the pointer (record identifier)**
|Cd.title| = 80B # **we determine size for title (since we have 5 atributes in Cd => 400B/5 = 80B)**
|Data entry| = |Index entry| = |Cd.title| + |rid| = 80B + 16B = 96B

Calculate the number of data entries per page : 8000/96 = 84
Data entries per page = 84 = DEP

|Cd| = 300K records ==> 300K data entries in the leaves of the index
1st level of the index (Leaves) = 300K / DEP = 300k/84 = 3572 pages = 1LVL
2nd level of the index (Nodes) = 1LVL / DEP = 3572/84 = 43 pages = 2LVL
3rd level of the index (Root) = 2LVL/DEP = 43/84 = 0,…. => 1 page

Total size of the index = 3572 + 43 + 1 = 3616 pages