B Phase Report

George Valavanis
George Andreadis

Task 2:

Step 2

At first we implemented the request with the following query:

```
with std as
( select  amka
   from ("Workgroup" wg join "LabModule" lm
                  using (module_no, course_code, serial_number) join "Joins" sjw
       using("wgID", module_no, course_code, serial_number))

   where type = 'project' and grade > 8)


select name, surname
from "Student" natural join "Person" natural join std
where entry_date between '2010-9-1' and '2011-9-1';
```

Executing EXPLAIN ANALYZE with the query we had the following results:

```
Nested Loop  (cost=1320.49..13717.66 rows=5 width=34) (actual time=14.462..282.458 rows=13 loops=1)
[...] Join Filter: ((lm.module_no = wg.module_no) AND (lm.course_code = wg.course_code) AND (lm.serial_number = wg.serial_number))
[...] ->  Hash Join  (cost=1320.07..13691.74 rows=54 width=76) (actual time=5.351..122.006 rows=24422 loops=1)
[...] Hash Cond: ((sjw.module_no = lm.module_no) AND (sjw.course_code = lm.course_code) AND (sjw.serial_number = lm.serial_number))
[...] ->  Nested Loop  (cost=1268.63..12557.11 rows=137549 width=57) (actual time=5.068..95.791 rows=133187 loops=1)
[...] ->  Hash Join  (cost=1268.20..2769.41 rows=4978 width=58) (actual time=5.034..18.736 rows=4861 loops=1)
[...] Hash Cond: (("Person".amka)::text = ("Student".amka)::text)
[...] ->  Seq Scan on "Person"  (cost=0.00..1354.78 rows=55778 width=46) (actual time=0.009..3.451 rows=55778 loops=1)
[...] ->  Hash  (cost=1205.98..1205.98 rows=4978 width=12) (actual time=4.998..4.998 rows=4861 loops=1)
[...] Buckets: 8192  Batches: 1  Memory Usage: 273kB
[...] ->  Seq Scan on "Student"  (cost=0.00..1205.98 rows=4978 width=12) (actual time=0.018..4.344 rows=4861 loops=1)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 49071
[...] ->  Index Only Scan using "Joins_pkey" on "Joins" sjw  (cost=0.43..1.69 rows=28 width=35) (actual time=0.010..0.013 rows=27 loops=4861)
[...] Index Cond: (amka = ("Person".amka)::text)
[...] Heap Fetches: 0
[...] ->  Hash  (cost=45.40..45.40 rows=345 width=19) (actual time=0.273..0.273 rows=345 loops=1)
[...] Buckets: 1024  Batches: 1  Memory Usage: 26kB
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..45.40 rows=345 width=19) (actual time=0.025..0.201 rows=345 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1527
[...] ->  Index Scan using "Workgroup_pkey" on "Workgroup" wg  (cost=0.42..0.46 rows=1 width=23) (actual time=0.006..0.006 rows=0 loops=24422)
[...] Index Cond: ((course_code = sjw.course_code) AND (serial_number = sjw.serial_number) AND (module_no = sjw.module_no) AND ("wgID" = sjw."wgID"))
[...] Filter: (grade > '8'::numeric)
[...] Rows Removed by Filter: 1
```

"Planning Time: 2.366 ms"
"Execution Time: 287.506 ms"

Disabling hashjoin the following times were obtained:

```
Nested Loop  (cost=33383.40..34788.38 rows=5 width=34) (actual time=271.487..463.386 rows=13 loops=1)
[...] Join Filter: ((lm.module_no = wg.module_no) AND (lm.course_code = wg.course_code) AND (lm.serial_number = wg.serial_number))
[...] ->  Merge Join  (cost=33382.98..34762.46 rows=54 width=76) (actual time=248.556..328.283 rows=24422 loops=1)
[...] Merge Cond: ((sjw.module_no = lm.module_no) AND (sjw.course_code = lm.course_code) AND (sjw.serial_number = lm.serial_number))
[...] ->  Sort  (cost=33323.04..33666.91 rows=137549 width=57) (actual time=248.290..308.790 rows=133013 loops=1)
[...] Sort Key: sjw.module_no, sjw.course_code, sjw.serial_number
[...] Sort Method: external merge  Disk: 9120kB
[...] ->  Nested Loop  (cost=1512.60..16410.52 rows=137549 width=57) (actual time=19.398..141.794 rows=133187 loops=1)
[...] ->  Merge Join  (cost=1512.17..6622.82 rows=4978 width=58) (actual time=19.373..74.025 rows=4861 loops=1)
[...] Merge Cond: (("Person".amka)::text = ("Student".amka)::text)
[...] ->  Index Scan using "Person_pkey" on "Person"  (cost=0.41..4897.04 rows=55778 width=46) (actual time=0.010..29.486 rows=55625 loops=1)
[...] ->  Sort  (cost=1511.66..1524.11 rows=4978 width=12) (actual time=19.343..20.042 rows=4861 loops=1)
[...] Sort Key: "Student".amka
[...] Sort Method: quicksort  Memory: 420kB
[...] ->  Seq Scan on "Student"  (cost=0.00..1205.98 rows=4978 width=12) (actual time=0.017..4.405 rows=4861 loops=1)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 49071
[...] ->  Index Only Scan using "Joins_pkey" on "Joins" sjw  (cost=0.43..1.69 rows=28 width=35) (actual time=0.008..0.011 rows=27 loops=4861)
[...] Index Cond: (amka = ("Person".amka)::text)
[...] Heap Fetches: 0
[...] ->  Sort  (cost=59.94..60.81 rows=345 width=19) (actual time=0.260..0.286 rows=345 loops=1)
[...] Sort Key: lm.module_no, lm.course_code, lm.serial_number
[...] Sort Method: quicksort  Memory: 51kB
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..45.40 rows=345 width=19) (actual time=0.028..0.203 rows=345 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1527
[...] ->  Index Scan using "Workgroup_pkey" on "Workgroup" wg  (cost=0.42..0.46 rows=1 width=23) (actual time=0.005..0.005 rows=0 loops=24422)
[...] Index Cond: ((course_code = sjw.course_code) AND (serial_number = sjw.serial_number) AND (module_no = sjw.module_no) AND ("wgID" = sjw."wgID"))
[...] Filter: (grade > '8'::numeric)
[...] Rows Removed by Filter: 1
```

"Planning Time: 1.997 ms"
"Execution Time: 468.117 ms"

Where we used to run hashjoin, hashcond now we have mergejoin,mergecond where mergecond uses various classification algorithms such as external merge, quicksort since hash is no longer allowed.

Disabling mergejoin too the following were obtained:

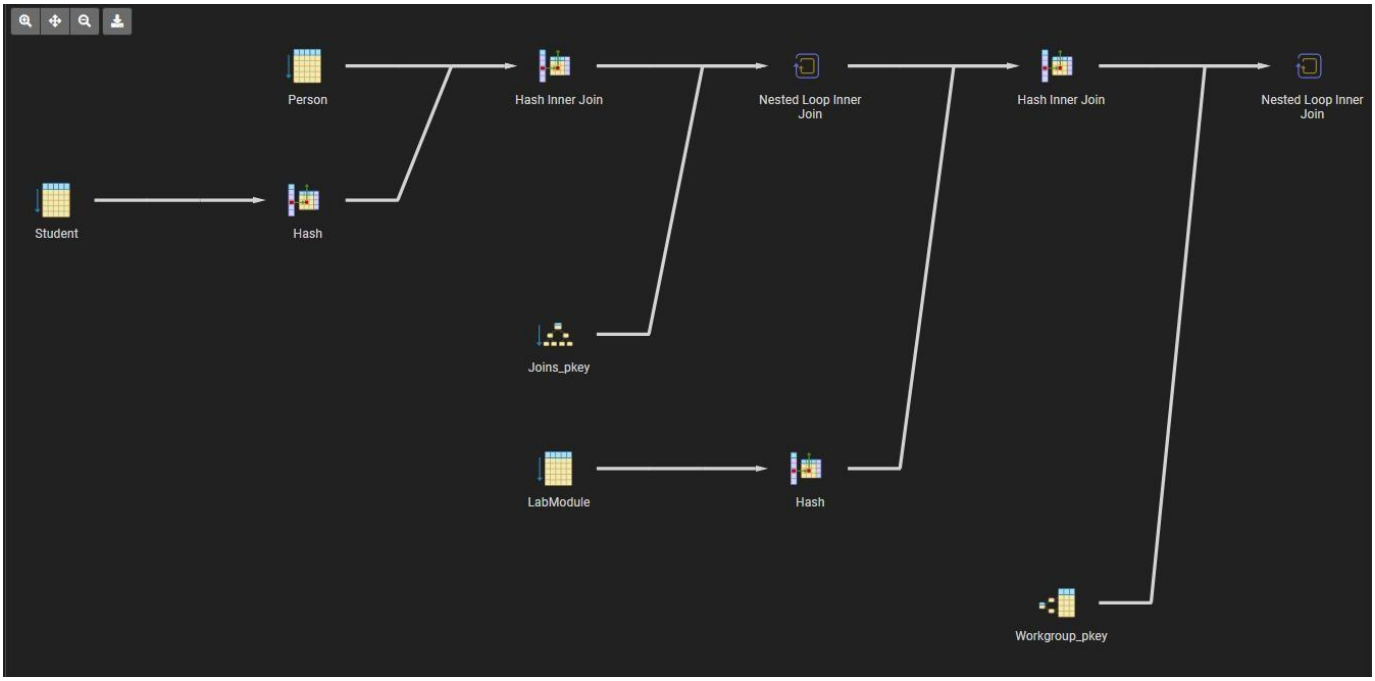
```
Nested Loop  (cost=1.54..58853.70 rows=5 width=34) (actual time=59.523..832.474 rows=13 loops=1)
[...] Join Filter: ((lm.module_no = wg.module_no) AND (lm.course_code = wg.course_code) AND (lm.serial_number = wg.serial_number))
[...] ->  Nested Loop  (cost=1.12..58827.79 rows=54 width=76) (actual time=0.076..667.057 rows=24422 loops=1)
[...] ->  Nested Loop  (cost=0.84..17206.66 rows=137549 width=57) (actual time=0.053..129.680 rows=133187 loops=1)
[...] ->  Nested Loop  (cost=0.41..7418.97 rows=4978 width=58) (actual time=0.039..44.361 rows=4861 loops=1)
[...] ->  Seq Scan on "Student"  (cost=0.00..1205.98 rows=4978 width=12) (actual time=0.016..5.148 rows=4861 loops=1)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 49071
[...] ->  Index Scan using "Person_pkey" on "Person"  (cost=0.41..1.25 rows=1 width=46) (actual time=0.008..0.008 rows=1 loops=4861)
[...] Index Cond: ((amka)::text = ("Student".amka)::text)
[...] ->  Index Only Scan using "Joins_pkey" on "Joins" sjw  (cost=0.43..1.69 rows=28 width=35) (actual time=0.010..0.014 rows=27 loops=4861)
[...] Index Cond: (amka = ("Person".amka)::text)
[...] Heap Fetches: 0
[...] ->  Index Scan using "LabModule_pkey" on "LabModule" lm  (cost=0.28..0.30 rows=1 width=19) (actual time=0.004..0.004 rows=0 loops=133187)
[...] Index Cond: ((course_code = sjw.course_code) AND (serial_number = sjw.serial_number) AND (module_no = sjw.module_no))
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1
[...] ->  Index Scan using "Workgroup_pkey" on "Workgroup" wg  (cost=0.42..0.46 rows=1 width=23) (actual time=0.006..0.006 rows=0 loops=24422)
[...] Index Cond: ((course_code = sjw.course_code) AND (serial_number = sjw.serial_number) AND (module_no = sjw.module_no) AND ("wgID" = sjw."wgID"))
[...] Filter: (grade > '8'::numeric)
[...] Rows Removed by Filter: 1
```

"Planning Time: 1.038 ms"
"Execution Time: 829.190 ms"

Now mergejoins are not allowed either so there are only nested loops which slow down the execution time quite a bit.

Next mergejoin, hashjoin were enabled again.

Changing the order of the natural join , putting std first and then Student and Person resulted in the following :



"Planning Time: 2.139 ms"
"Execution Time: 281.898 ms"

There is no difference neither in times nor in the execution plan , as previously the Person table was done first and then the Student join. The optimizer gives priority to the largest table which is Person.

Looking at the execution plan we see the following:

```
[...] -> Hash Join  (cost=1320.07..13691.74 rows=54 width=76) (actual time=5.381..120.353 rows=24422 loops=1)
[...] Hash Cond: ((sjw.module_no = lm.module_no) AND (sjw.course_code = lm.course_code) AND (sjw.serial_number = lm.serial_number))
```

There is a high cost to hash join. Therefore if we create an index in the Joins table or the LabModule table we can probably improve the execution time.

We choose the following index: (we use btree while we have equalities since we can't use hash on a combination of features)

create index joins_pkey_idx on "Joins" using btree(course_code, serial_number, module_no, "wgID");

The following times and execution plan were achieved:

```
Nested Loop  (cost=1.68..2912.10 rows=5 width=34) (actual time=2.629..23.240 rows=13 loops=1)
[...] Join Filter: (("Student".amka)::text = ("Person".amka)::text)
[...] -> Nested Loop  (cost=1.26..2909.86 rows=5 width=24) (actual time=2.614..23.117 rows=13 loops=1)
[...] -> Nested Loop  (cost=0.85..2885.71 rows=55 width=12) (actual time=0.089..21.972 rows=141 loops=1)
[...] -> Nested Loop  (cost=0.42..2767.24 rows=14 width=42) (actual time=0.079..21.692 rows=32 loops=1)
[...] -> Seq Scan on "LabModule" lm  (cost=0.00..45.40 rows=345 width=19) (actual time=0.020..0.216 rows=345 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1527
[...] -> Index Scan using "Workgroup_pkey" on "Workgroup" wg  (cost=0.42..7.88 rows=1 width=23) (actual time=0.060..0.062 rows=0 loops=345)
[...] Index Cond: ((course_code = lm.course_code) AND (serial_number = lm.serial_number) AND (module_no = lm.module_no))
[...] Filter: (grade > '8'::numeric)
[...] Rows Removed by Filter: 201
[...] -> Index Scan using joins_pkey_idx on "Joins" sjw  (cost=0.43..8.45 rows=1 width=35) (actual time=0.007..0.008 rows=4 loops=32)
[...] Index Cond: ((course_code = wg.course_code) AND (serial_number = wg.serial_number) AND (module_no = wg.module_no) AND ("wgID" = wg."wgID"))
[...] -> Index Scan using "Student_pkey" on "Student"  (cost=0.41..0.44 rows=1 width=12) (actual time=0.008..0.008 rows=0 loops=141)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 1
[...] -> Index Scan using "Person_pkey" on "Person"  (cost=0.41..0.44 rows=1 width=46) (actual time=0.009..0.009 rows=1 loops=13)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

"Planning Time: 2.257 ms"
"Execution Time: 35.504 ms"

Now the execution plan uses the index and there is a noticeable improvement in time.

We try to make an index for LabModule, again btree in course_code, serial_number, module_no.

create index lm_pkey_idx on "LabModule" using btree(course_code, serial_number, module_no)

The following were achieved:

```
Nested Loop  (cost=1.68..2912.10 rows=5 width=34) (actual time=2.710..23.335 rows=13 loops=1)
[...] Join Filter: (("Student".amka)::text = ("Person".amka)::text)
[...] ->  Nested Loop  (cost=1.26..2909.86 rows=5 width=24) (actual time=2.693..23.209 rows=13 loops=1)
[...] ->  Nested Loop  (cost=0.85..2885.71 rows=55 width=12) (actual time=0.125..22.095 rows=141 loops=1)
[...] ->  Nested Loop  (cost=0.42..2767.24 rows=14 width=42) (actual time=0.112..21.813 rows=32 loops=1)
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..45.40 rows=345 width=19) (actual time=0.048..0.245 rows=345 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1527
[...] ->  Index Scan using "Workgroup_pkey" on "Workgroup" wg  (cost=0.42..7.88 rows=1 width=23) (actual time=0.060..0.062 rows=0 loops=345)
[...] Index Cond: ((course_code = lm.course_code) AND (serial_number = lm.serial_number) AND (module_no = lm.module_no))
[...] Filter: (grade > '8'::numeric)
[...] Rows Removed by Filter: 201
[...] ->  Index Scan using joins_pkey_idx on "Joins" sjw  (cost=0.43..8.45 rows=1 width=35) (actual time=0.007..0.008 rows=4 loops=32)
[...] Index Cond: ((course_code = wg.course_code) AND (serial_number = wg.serial_number) AND (module_no = wg.module_no) AND ("wgID" = wg."wgID"))
[...] ->  Index Scan using "Student_pkey" on "Student"  (cost=0.41..0.44 rows=1 width=12) (actual time=0.008..0.008 rows=0 loops=141)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 1
[...] ->  Index Scan using "Person_pkey" on "Person"  (cost=0.41..0.44 rows=1 width=46) (actual time=0.009..0.009 rows=1 loops=13)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
Planning Time: 2.387 ms
Execution Time: 23.384 ms
```

The runtime is not improved and the index is not used in the execution plan. We therefore delete this

index since it is unnecessary.

We also notice the following::

```
[...] ->  Index Scan using "Workgroup_pkey" on "Workgroup" wg  (cost=0.42..7.88 rows=1 width=23) (actual time=0.060..0.062 rows=0 loops=345)
[...] Index Cond: ((course_code = lm.course_code) AND (serial_number = lm.serial_number) AND (module_no = lm.module_no))
[...] Filter: (grade > '8'::numeric)
```

The index scan is low cost but it has 345 loops. We may be able to avoid it entirely by creating an index for the grade which is used as a filter. Probably also reduce the costs of nested loops since the grade is in the where condition of the std table.

Therefore we create it:

create index workgroup_grade on "Workgroup" using btree(grade);

The following were achieved:

```
Nested Loop  (cost=57.71..469.10 rows=5 width=34) (actual time=0.423..1.914 rows=13 loops=1)
[...] Join Filter: (("Student".amka)::text = ("Person".amka)::text)
[...] ->  Nested Loop  (cost=57.30..466.87 rows=5 width=24) (actual time=0.413..1.807 rows=13 loops=1)
[...] ->  Nested Loop  (cost=56.88..442.71 rows=55 width=12) (actual time=0.296..0.763 rows=141 loops=1)
[...] ->  Hash Join  (cost=56.46..324.24 rows=14 width=42) (actual time=0.283..0.491 rows=32 loops=1)
[...] Hash Cond: ((wg.module_no = lm.module_no) AND (wg.course_code = lm.course_code) AND (wg.serial_number = lm.serial_number))
[...] ->  Bitmap Heap Scan on "Workgroup" wg  (cost=5.02..272.20 rows=77 width=23) (actual time=0.042..0.197 rows=209 loops=1)
[...] Recheck Cond: (grade > '8'::numeric)
[...] Heap Blocks: exact=205
[...] ->  Bitmap Index Scan on workgroup_grade  (cost=0.00..5.00 rows=77 width=0) (actual time=0.026..0.026 rows=209 loops=1)
[...] Index Cond: (grade > '8'::numeric)
[...] ->  Hash  (cost=45.40..45.40 rows=345 width=19) (actual time=0.236..0.236 rows=345 loops=1)
[...] Buckets: 1024  Batches: 1  Memory Usage: 26kB
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..45.40 rows=345 width=19) (actual time=0.009..0.175 rows=345 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1527
[...] ->  Index Scan using joins_pkey_idx on "Joins" sjw  (cost=0.43..8.45 rows=1 width=35) (actual time=0.007..0.008 rows=4 loops=32)
[...] Index Cond: ((course_code = wg.course_code) AND (serial_number = wg.serial_number) AND (module_no = wg.module_no) AND ("wgID" = wg."wgID"))
[...] ->  Index Scan using "Student_pkey" on "Student"  (cost=0.41..0.44 rows=1 width=12) (actual time=0.007..0.007 rows=0 loops=141)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 1
[...] ->  Index Scan using "Person_pkey" on "Person"  (cost=0.41..0.44 rows=1 width=46) (actual time=0.008..0.008 rows=1 loops=13)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

"Planning Time: 2.337 ms"
"Execution Time: 1.968 ms"

Looking at the execution plan, a nested loop has been replaced with a hash inner join and now instead of an index scan in the WorkGroup table the following is done:

```
[...] ->  Bitmap Heap Scan on "Workgroup" wg  (cost=5.02..272.20 rows=77 width=23) (actual time=0.042..0.197 rows=209 loops=1)
[...] Recheck Cond: (grade > '8'::numeric)
[...] Heap Blocks: exact=205
[...] ->  Bitmap Index Scan on workgroup_grade  (cost=0.00..5.00 rows=77 width=0) (actual time=0.026..0.026 rows=209 loops=1)
[...] Index Cond: (grade > '8'::numeric)
[...] ->  Hash  (cost=45.40..45.40 rows=345 width=19) (actual time=0.236..0.236 rows=345 loops=1)
[...] Buckets: 1024  Batches: 1  Memory Usage: 26kB
```

In addition, the costs of the remaining nested loops have been reduced considerably compared to before. The efficiency of this index is evident.

To make the execution even faster we will cluster the Workgroup using the index above. Therefore elements with a degree greater than 8 will be stored close to each other.

cluster "Workgroup" using workgroup_grade

The following were achieved:

```
Nested Loop  (cost=57.71..469.10 rows=5 width=34) (actual time=0.389..1.849 rows=13 loops=1)
[...] Join Filter: (("Student".amka)::text = ("Person".amka)::text)
[...] ->  Nested Loop  (cost=57.30..466.87 rows=5 width=24) (actual time=0.378..1.712 rows=13 loops=1)
[...] ->  Nested Loop  (cost=56.88..442.71 rows=55 width=12) (actual time=0.284..0.603 rows=141 loops=1)
[...] ->  Hash Join  (cost=56.46..324.24 rows=14 width=42) (actual time=0.271..0.329 rows=32 loops=1)
[...] Hash Cond: ((wg.module_no = lm.module_no) AND (wg.course_code = lm.course_code) AND (wg.serial_number = lm.serial_number))
[...] ->  Bitmap Heap Scan on "Workgroup" wg  (cost=5.02..272.20 rows=77 width=23) (actual time=0.026..0.040 rows=209 loops=1)
[...] Recheck Cond: (grade > '8'::numeric)
[...] Heap Blocks: exact=2
[...] ->  Bitmap Index Scan on workgroup_grade  (cost=0.00..5.00 rows=77 width=0) (actual time=0.020..0.021 rows=209 loops=1)
[...] Index Cond: (grade > '8'::numeric)
[...] ->  Hash  (cost=45.40..45.40 rows=345 width=19) (actual time=0.240..0.240 rows=345 loops=1)
[...] Buckets: 1024  Batches: 1  Memory Usage: 26kB
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..45.40 rows=345 width=19) (actual time=0.009..0.177 rows=345 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1527
[...] ->  Index Scan using joins_pkey_idx on "Joins" sjw  (cost=0.43..8.45 rows=1 width=35) (actual time=0.007..0.008 rows=4 loops=32)
[...] Index Cond: ((course_code = wg.course_code) AND (serial_number = wg.serial_number) AND (module_no = wg.module_no) AND ("wgID" = wg."wgID"))
[...] ->  Index Scan using "Student_pkey" on "Student"  (cost=0.41..0.44 rows=1 width=12) (actual time=0.008..0.008 rows=0 loops=141)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 1
[...] ->  Index Scan using "Person_pkey" on "Person"  (cost=0.41..0.44 rows=1 width=46) (actual time=0.009..0.009 rows=1 loops=13)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

"Planning Time: 2.340 ms"
"Execution Time: 1.883 ms"

The resulting times are slightly better.

Looking at the graphical execution plan provided by explain analyse we see that the indexing and clustering in the Workgroup table is done before the costly join. Observing the shape of our tree we see that we have ended up with a left deep tree in which we have moved all selections and all projections to the leaves of the tree to do the filtering. As a result, the joins have a smaller number of tuples to manage, hence less cost in terms of lines and loops.

This image shows the form of the left deep tree that was created, also the image shows the two indexes mentioned below that do not affect the form of the tree.

Studying the execution plan we observe the following :

```
19  [...] -> Index Scan using "Student_pkey" on "Student"  (cost=0.41..0.44 rows=1 width=12) (actual time=0.008..0.008 rows=0 loops=141)
20  [...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

Low cost with many loops. We can probably improve it with an index in Student's amka. So we create the following index:

create index std_amka_idx on "Student" using hash(amka)

The following were achieved:

```
Nested Loop  (cost=57.30..446.29 rows=5 width=34) (actual time=0.411..1.189 rows=13 loops=1)
[...] Join Filter: (("Student".amka)::text = ("Person".amka)::text)
[...] -> Nested Loop  (cost=56.88..444.05 rows=5 width=24) (actual time=0.395..1.069 rows=13 loops=1)
[...] -> Nested Loop  (cost=56.88..442.71 rows=55 width=12) (actual time=0.333..0.671 rows=141 loops=1)
[...] -> Hash Join  (cost=56.46..324.24 rows=14 width=42) (actual time=0.315..0.373 rows=32 loops=1)
[...] Hash Cond: ((wg.module_no = lm.module_no) AND (wg.course_code = lm.course_code) AND (wg.serial_number = lm.serial_number))
[...] -> Bitmap Heap Scan on "Workgroup" wg  (cost=5.02..272.20 rows=77 width=23) (actual time=0.040..0.053 rows=209 loops=1)
[...] Recheck Cond: (grade > '8'::numeric)
[...] Heap Blocks: exact=2
[...] -> Bitmap Index Scan on workgroup_grade  (cost=0.00..5.00 rows=77 width=0) (actual time=0.033..0.034 rows=209 loops=1)
[...] Index Cond: (grade > '8'::numeric)
[...] -> Hash  (cost=45.40..45.40 rows=345 width=19) (actual time=0.265..0.266 rows=345 loops=1)
[...] Buckets: 1024  Batches: 1  Memory Usage: 26kB
[...] -> Seq Scan on "LabModule" lm  (cost=0.00..45.40 rows=345 width=19) (actual time=0.014..0.194 rows=345 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1527
[...] -> Index Scan using joins_pkey_idx on "Joins" sjw  (cost=0.43..8.45 rows=1 width=35) (actual time=0.008..0.009 rows=4 loops=32)
[...] Index Cond: ((course_code = wg.course_code) AND (serial_number = wg.serial_number) AND (module_no = wg.module_no) AND ("wgID" = wg."wgID"))
[...] -> Index Scan using std_amka_idx on "Student"  (cost=0.00..0.02 rows=1 width=12) (actual time=0.003..0.003 rows=0 loops=141)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 1
[...] -> Index Scan using "Person_pkey" on "Person"  (cost=0.41..0.44 rows=1 width=46) (actual time=0.009..0.009 rows=1 loops=13)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

"Planning Time: 2.314 ms"
"Execution Time: 1.109 ms"

Specifically:

```
[...] -> Index Scan using std_amka_idx on "Student"  (cost=0.00..0.02 rows=1 width=12) (actual time=0.003..0.003 rows=0 loops=141)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

The loops were not reduced, but the cost was almost zero as shown above.

Similarly, we could reduce the runtime a bit more if we made a corresponding index in the Person table since we have the following:

```
[...] -> Index Scan using "Person_pkey" on "Person"  (cost=0.41..0.44 rows=1 width=46) (actual time=0.009..0.009 rows=1 loops=13)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

Therefore we create the following index:

create index person_amka_idx on "Person" using hash(amka)

The following were achieved:

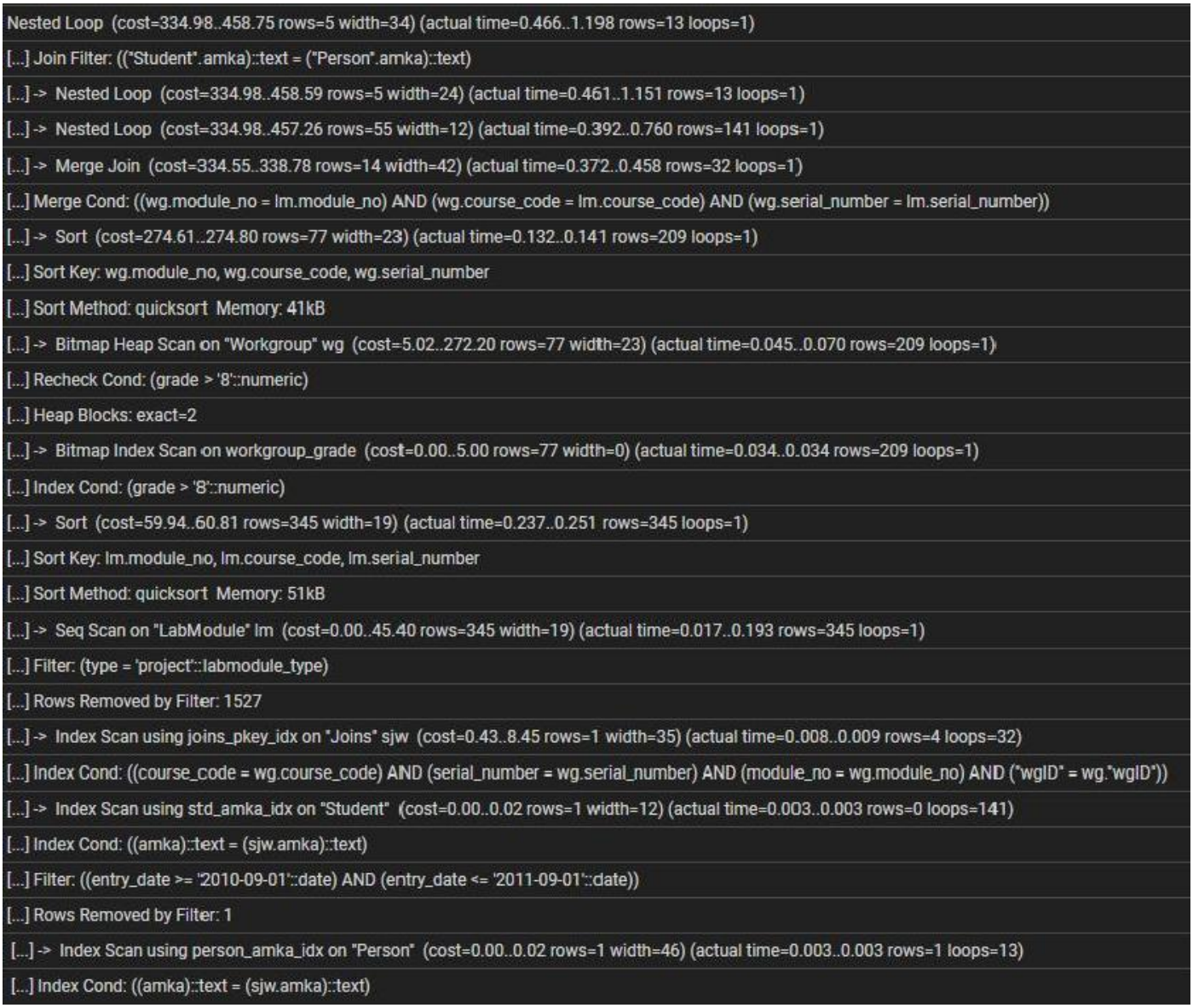
```
Nested Loop  (cost=56.88..444.21 rows=5 width=34) (actual time=0.338..0.986 rows=13 loops=1)
[...] Join Filter: (("Student".amka)::text = ("Person".amka)::text)
[...] ->  Nested Loop  (cost=56.88..444.05 rows=5 width=24) (actual time=0.335..0.946 rows=13 loops=1)
[...] ->  Nested Loop  (cost=56.88..442.71 rows=55 width=12) (actual time=0.284..0.605 rows=141 loops=1)
[...] ->  Hash Join  (cost=56.46..324.24 rows=14 width=42) (actual time=0.272..0.328 rows=32 loops=1)
[...] Hash Cond: ((wg.module_no = lm.module_no) AND (wg.course_code = lm.course_code) AND (wg.serial_number = lm.serial_number))
[...] ->  Bitmap Heap Scan on "Workgroup" wg  (cost=5.02..272.20 rows=77 width=23) (actual time=0.028..0.041 rows=209 loops=1)
[...] Recheck Cond: (grade > '8'::numeric)
[...] Heap Blocks: exact=2
[...] ->  Bitmap Index Scan on workgroup_grade  (cost=0.00..5.00 rows=77 width=0) (actual time=0.022..0.022 rows=209 loops=1)
[...] Index Cond: (grade > '8'::numeric)
[...] ->  Hash  (cost=45.40..45.40 rows=345 width=19) (actual time=0.238..0.239 rows=345 loops=1)
[...] Buckets: 1024  Batches: 1  Memory Usage: 26kB
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..45.40 rows=345 width=19) (actual time=0.011..0.176 rows=345 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1527
[...] ->  Index Scan using joins_pkey_idx on "Joins" sjw  (cost=0.43..8.45 rows=1 width=35) (actual time=0.007..0.008 rows=4 loops=32)
[...] Index Cond: ((course_code = wg.course_code) AND (serial_number = wg.serial_number) AND (module_no = wg.module_no) AND ("wgID" = wg."wgID"))
[...] ->  Index Scan using std_amka_idx on "Student"  (cost=0.00..0.02 rows=1 width=12) (actual time=0.002..0.002 rows=0 loops=141)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 1
[...] ->  Index Scan using person_amka_idx on "Person"  (cost=0.00..0.02 rows=1 width=46) (actual time=0.002..0.002 rows=1 loops=13)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

"Planning Time: 2.283 ms"
"Execution Time: 1.039 ms"

```
[...] ->  Index Scan using person_amka_idx on "Person"  (cost=0.00..0.02 rows=1 width=46) (actual time=0.002..0.002 rows=1 loops=13)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

We see that the costs are almost zero but because we had few iterations there was no huge difference in execution time.

Then disabling hashjoin resulted in the following times:

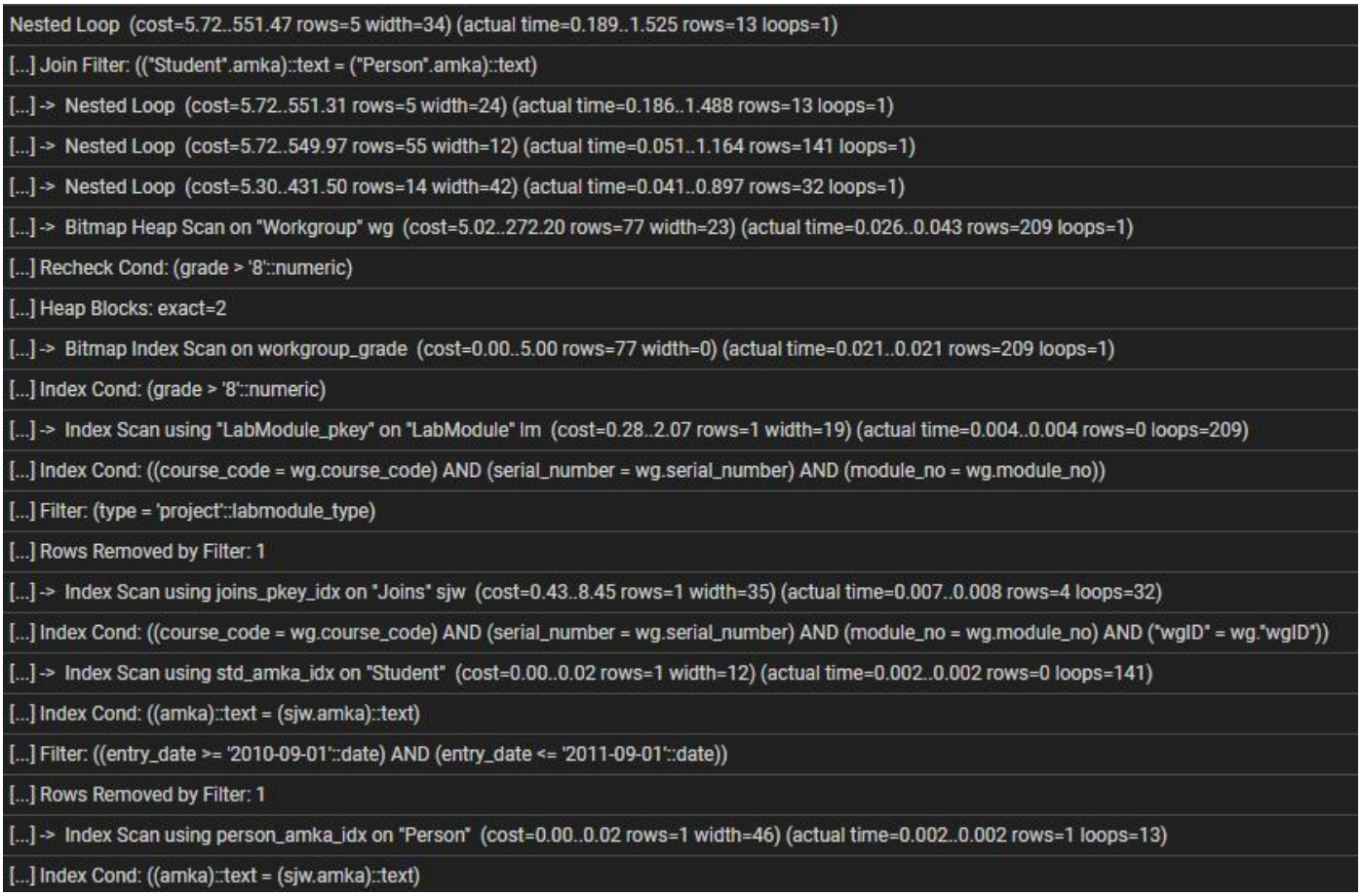


"Planning Time: 2.376 ms"
"Execution Time: 1.282 ms"

The execution and planning times are slightly increased.

And we notice that in the detailed execution plan the hashjoin between Workgroup and LabModule is converted to a merge join using a different method to sort the data instead of the hash.

Then disabling mergejoin too resulted in the following :

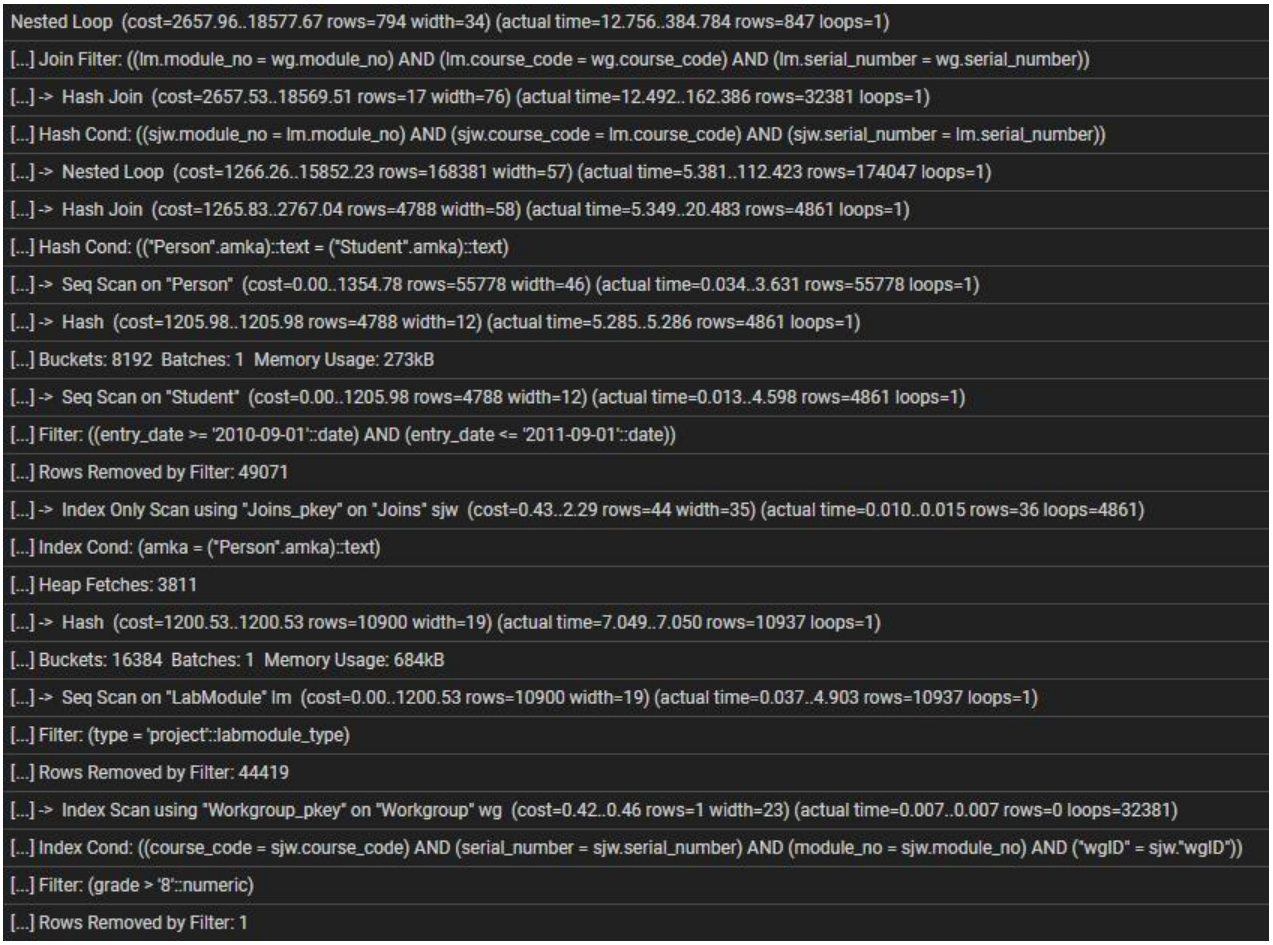


"Planning Time: 1.074 ms"
"Execution Time: 1.598 ms"

In this case we see the replacement of the mergejoin and sort methods with nested loop and index scan.

Step 3

After entering hundreds of data for step 3 , and deleting the indexes we made, we obtained the following times:



"Planning Time: 3.658 ms"
"Execution Time: 393.146 ms"

It is obvious that compared to before the execution time is much longer (almost 5 times longer). Observing the new execution plan is similar to the previous one , only that the costs are higher (e.g. in nested loops) and the iterations in the index scan.

Disabling hashjoin resulted in the following :

```
Nested Loop  (cost=42574.24..44350.19 rows=794 width=34) (actual time=292.138..549.137 rows=847 loops=1)
[...] Join Filter: ((lm.module_no = wg.module_no) AND (lm.course_code = wg.course_code) AND (lm.serial_number = wg.serial_number))
[...] ->  Merge Join  (cost=42573.81..44342.02 rows=17 width=76) (actual time=266.721..366.783 rows=32381 loops=1)
[...] Merge Cond: ((sjw.module_no = lm.module_no) AND (sjw.course_code = lm.course_code) AND (sjw.serial_number = lm.serial_number))
[...] ->  Sort  (cost=40640.26..41061.21 rows=168381 width=57) (actual time=259.983..333.078 rows=174039 loops=1)
[...] Sort Key: sjw.module_no, sjw.course_code, sjw.serial_number
[...] Sort Method: external merge  Disk: 11896kB
[...] ->  Nested Loop  (cost=1499.49..19692.14 rows=168381 width=57) (actual time=6.338..135.143 rows=174047 loops=1)
[...] ->  Merge Join  (cost=1499.07..6606.95 rows=4788 width=58) (actual time=6.311..56.912 rows=4861 loops=1)
[...] Merge Cond: (("Person".amka)::text = ("Student".amka)::text)
[...] ->  Index Scan using "Person_pkey" on "Person"  (cost=0.41..4897.03 rows=55778 width=46) (actual time=0.009..26.655 rows=55625 loops=1)
[...] ->  Sort  (cost=1498.65..1510.62 rows=4788 width=12) (actual time=6.285..6.628 rows=4861 loops=1)
[...] Sort Key: "Student".amka
[...] Sort Method: quicksort  Memory: 420kB
[...] ->  Seq Scan on "Student"  (cost=0.00..1205.98 rows=4788 width=12) (actual time=0.028..4.610 rows=4861 loops=1)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 49071
[...] ->  Index Only Scan using "Joins_pkey" on "Joins" sjw  (cost=0.43..2.29 rows=44 width=35) (actual time=0.008..0.013 rows=36 loops=4861)
[...] Index Cond: (amka = ("Person".amka)::text)
[...] Heap Fetches: 3811
[...] ->  Sort  (cost=1931.48..1958.73 rows=10900 width=19) (actual time=6.729..7.245 rows=10937 loops=1)
[...] Sort Key: lm.module_no, lm.course_code, lm.serial_number
[...] Sort Method: quicksort  Memory: 1239kB
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..1200.53 rows=10900 width=19) (actual time=0.028..5.032 rows=10937 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 44419
[...] ->  Index Scan using "Workgroup_pkey" on "Workgroup" wg  (cost=0.42..0.46 rows=1 width=23) (actual time=0.005..0.005 rows=0 loops=32381)
[...] Index Cond: ((course_code = sjw.course_code) AND (serial_number = sjw.serial_number) AND (module_no = sjw.module_no) AND ("wgID" = sjw."wgID"))
[...] Filter: (grade > '8'::numeric)
[...] Rows Removed by Filter: 1
```

"Planning Time: 3.653 ms"
"Execution Time: 575.957 ms"

The same changes apply as explained for step 2.

Disabling mergejoin too resulted in the following :

```
Nested Loop  (cost=1.68..94850.53 rows=794 width=34) (actual time=21.310..1272.773 rows=847 loops=1)
[...] Join Filter: ((lm.module_no = wg.module_no) AND (lm.course_code = wg.course_code) AND (lm.serial_number = wg.serial_number))
[...] ->  Nested Loop  (cost=1.26..94842.36 rows=17 width=76) (actual time=0.084..1048.811 rows=32381 loops=1)
[...] ->  Nested Loop  (cost=0.84..20421.98 rows=168381 width=57) (actual time=0.056..133.592 rows=174047 loops=1)
[...] ->  Nested Loop  (cost=0.41..7336.79 rows=4788 width=58) (actual time=0.042..40.155 rows=4861 loops=1)
[...] ->  Seq Scan on "Student"  (cost=0.00..1205.98 rows=4788 width=12) (actual time=0.019..5.458 rows=4861 loops=1)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 49071
[...] ->  Index Scan using "Person_pkey" on "Person"  (cost=0.41..1.28 rows=1 width=46) (actual time=0.007..0.007 rows=1 loops=4861)
[...] Index Cond: ((amka)::text = ("Student".amka)::text)
[...] ->  Index Only Scan using "Joins_pkey" on "Joins" sjw  (cost=0.43..2.29 rows=44 width=35) (actual time=0.008..0.015 rows=36 loops=4861)
[...] Index Cond: (amka = ("Person".amka)::text)
[...] Heap Fetches: 3811
[...] ->  Index Scan using "LabModule_pkey" on "LabModule" lm  (cost=0.41..0.44 rows=1 width=19) (actual time=0.005..0.005 rows=0 loops=174047)
[...] Index Cond: ((course_code = sjw.course_code) AND (serial_number = sjw.serial_number) AND (module_no = sjw.module_no))
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 1
[...] ->  Index Scan using "Workgroup_pkey" on "Workgroup" wg  (cost=0.42..0.46 rows=1 width=23) (actual time=0.007..0.007 rows=0 loops=32381)
[...] Index Cond: ((course_code = sjw.course_code) AND (serial_number = sjw.serial_number) AND (module_no = sjw.module_no) AND ("wgID" = sjw."wgID"))
[...] Filter: (grade > '8'::numeric)
[...] Rows Removed by Filter: 1
```

"Planning Time: 1.504 ms"
"Execution Time: 1318.337 ms"

The same changes apply as explained for step 2.

We choose the same indexes as before, since these were found to be the most effective.
While importing them , we observed similar changes as discussed in step 2.
After importing the indexes and clustering the Workgroup, the following execution times and plan were obtained:

```
Nested Loop  (cost=1615.45..16884.52 rows=794 width=34) (actual time=8.219..44.399 rows=847 loops=1)
[...] Join Filter: (("Student".amka)::text = ("Person".amka)::text)
[...] ->  Nested Loop  (cost=1615.45..16858.93 rows=794 width=24) (actual time=8.212..42.720 rows=847 loops=1)
[...] ->  Nested Loop  (cost=1615.45..16645.36 rows=8943 width=12) (actual time=8.142..29.962 rows=8717 loops=1)
[...] ->  Hash Join  (cost=1615.02..5477.12 rows=2322 width=42) (actual time=8.108..12.402 rows=2498 loops=1)
[...] Hash Cond: ((wg.module_no = lm.module_no) AND (wg.course_code = lm.course_code) AND (wg.serial_number = lm.serial_number))
[...] ->  Bitmap Heap Scan on "Workgroup" wg  (cost=223.75..3993.05 rows=11784 width=23) (actual time=0.830..1.735 rows=12553 loops=1)
[...] Recheck Cond: (grade > '8'::numeric)
[...] Heap Blocks: exact=93
[...] ->  Bitmap Index Scan on workgroup_grade  (cost=0.00..220.80 rows=11784 width=0) (actual time=0.820..0.820 rows=12553 loops=1)
[...] Index Cond: (grade > '8'::numeric)
[...] ->  Hash  (cost=1200.53..1200.53 rows=10900 width=19) (actual time=7.231..7.232 rows=10937 loops=1)
[...] Buckets: 16384  Batches: 1  Memory Usage: 684kB
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..1200.53 rows=10900 width=19) (actual time=0.013..4.964 rows=10937 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 44419
[...] ->  Index Scan using joins_pkey_idx on "Joins" sjw  (cost=0.43..4.80 rows=1 width=35) (actual time=0.006..0.006 rows=3 loops=2498)
[...] Index Cond: ((course_code = wg.course_code) AND (serial_number = wg.serial_number) AND (module_no = wg.module_no) AND ("wgID" = wg."wgID"))
[...] ->  Index Scan using std_amka_idx on "Student"  (cost=0.00..0.02 rows=1 width=12) (actual time=0.001..0.001 rows=0 loops=8717)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 1
[...] ->  Index Scan using person_amka_idx on "Person"  (cost=0.00..0.02 rows=1 width=46) (actual time=0.001..0.001 rows=1 loops=847)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

"Planning Time: 3.867 ms"
"Execution Time: 47.462 ms"

It is obvious that the runtime has been reduced quite a bit, as expected.

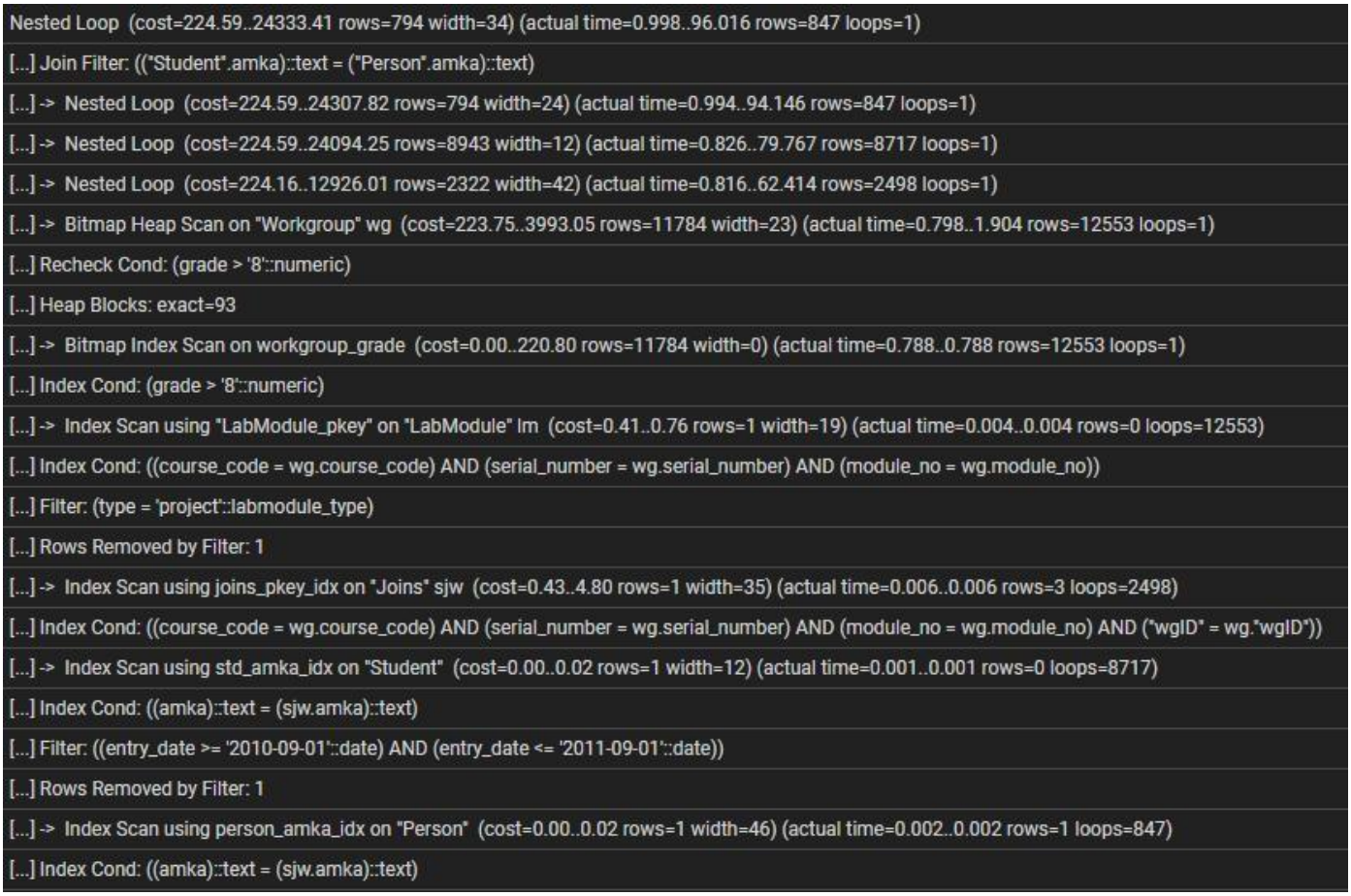
Disabling hashjoin resulted in the following :

```
Nested Loop  (cost=6721.98..18351.35 rows=794 width=34) (actual time=13.010..50.751 rows=847 loops=1)
[...] Join Filter: (("Student".amka)::text = ("Person".amka)::text)
[...] ->  Nested Loop  (cost=6721.98..18325.77 rows=794 width=24) (actual time=13.005..48.847 rows=847 loops=1)
[...] ->  Nested Loop  (cost=6721.98..18112.20 rows=8943 width=12) (actual time=12.947..34.935 rows=8717 loops=1)
[...] ->  Merge Join  (cost=6721.56..6943.96 rows=2322 width=42) (actual time=12.920..17.035 rows=2498 loops=1)
[...] Merge Cond: ((wg.module_no = lm.module_no) AND (wg.course_code = lm.course_code) AND (wg.serial_number = lm.serial_number))
[...] ->  Sort  (cost=4789.91..4819.37 rows=11784 width=23) (actual time=6.487..7.220 rows=12552 loops=1)
[...] Sort Key: wg.module_no, wg.course_code, wg.serial_number
[...] Sort Method: quicksort  Memory: 1365kB
[...] ->  Bitmap Heap Scan on "Workgroup" wg  (cost=223.75..3993.05 rows=11784 width=23) (actual time=0.793..2.445 rows=12553 loops=1)
[...] Recheck Cond: (grade > '8'::numeric)
[...] Heap Blocks: exact=93
[...] ->  Bitmap Index Scan on workgroup_grade  (cost=0.00..220.80 rows=11784 width=0) (actual time=0.782..0.783 rows=12553 loops=1)
[...] Index Cond: (grade > '8'::numeric)
[...] ->  Sort  (cost=1931.48..1958.73 rows=10900 width=19) (actual time=6.425..6.853 rows=10937 loops=1)
[...] Sort Key: lm.module_no, lm.course_code, lm.serial_number
[...] Sort Method: quicksort  Memory: 1239kB
[...] ->  Seq Scan on "LabModule" lm  (cost=0.00..1200.53 rows=10900 width=19) (actual time=0.034..4.870 rows=10937 loops=1)
[...] Filter: (type = 'project'::labmodule_type)
[...] Rows Removed by Filter: 44419
[...] ->  Index Scan using joins_pkey_idx on "Joins" sjw  (cost=0.43..4.80 rows=1 width=35) (actual time=0.006..0.007 rows=3 loops=2498)
[...] Index Cond: ((course_code = wg.course_code) AND (serial_number = wg.serial_number) AND (module_no = wg.module_no) AND ("wgID" = wg."wgID"))
[...] ->  Index Scan using std_amka_idx on "Student"  (cost=0.00..0.02 rows=1 width=12) (actual time=0.001..0.001 rows=0 loops=8717)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
[...] Filter: ((entry_date >= '2010-09-01'::date) AND (entry_date <= '2011-09-01'::date))
[...] Rows Removed by Filter: 1
[...] ->  Index Scan using person_amka_idx on "Person"  (cost=0.00..0.02 rows=1 width=46) (actual time=0.002..0.002 rows=1 loops=847)
[...] Index Cond: ((amka)::text = (sjw.amka)::text)
```

"Planning Time: 3.771 ms"
"Execution Time: 51.132 ms"

Small difference in time compared to before, since the hash was not used as much (a hash join between Workgroup and LabModule).

Disabling mergejoin resulted in the following :



"Planning Time: 1.618 ms"
"Execution Time: 96.324 ms"

In this case we observe the replacement of the mergejoin and sort methods with nested loop and index scan, as in step 2. Hence the long delay.