## A. Optimization process

1. Loading the data into the database and tested it without any optimization. Only primary keys, as specified in the milestone1.pdf, were added.
2. Changed queries after initial set-up. Idea is to create as few joins as possible. This was found with the EXPLAIN ANALYZE-function. The idea was in the back of our minds during the whole project and set-up of the database.
3. Experimentation with different indices:
The indices that we tried and that improved the queries, but we still found better ones that are described in B:
- hash index on student registration ID; Q7 decreased 70 seconds in running time locally
- hash index on course offer ID. improved Q5 by 3 seconds locally
- hash index on student ID decreased Q0 by 5 seconds and Q7 decreased with 4 seconds,

4. Created materialized views for slowest part of queries which were used in a with-statement. This could not be optimized in altering the queries.
5. Altered the loading of the tables to need less joins in queries. So, columns were added to existing tables in the database.

## B. Chosen optimizations

- Materialized views:
    - pointsperS; has the studentID, student registration ID, the sum of acquired ECTS and the GPA corresponding to the student registration ID.
    - Have_not_taken_yet; contains the student registration IDs and the sum of their acquired ECTS (0) of the students that are enrolled but have not successfully taken a course yet.
    - noFails; has student ID, student registration ID and the GPA corresponding to the student registration ID of students that have not gotten a grade below 5 while completing their degree.
    - HighestGrade; has the highest grade for each course offer. This view has only the courses offered in quartile 1 of 2018.
- Non-clustered Index on student registration ID, course offer ID and grade in table course registrations, referred to as idx_courseregistrations
- Non-clustered index on grade for table students referred to as idx_grade
- Altering the loading of course registrations by adding columns student ID, degree ID and course ID.
- Altering the loading of course offers by adding column courseName and ECTS.

| | | Without | With | |
|---|---|---|---|---|
| What? | Name | Performance/Cost | Extra Space | Performance/Usage |
| Loading data | Courseregistrations<br>CourseOffers | 5-10s per join locally | 636 MB<br>5 MB | Used in 2Qs, 1 view<br>Used in 2Qs, 1 view |
| Materialized views | Pointspers<br>have_not_taken_yet<br>NoFails<br>Highest_grade | ~ 2m locally<br>~ 38s locally<br>~ 270s locally<br>~ 25s locally | 288 MB<br>76 MB<br>296 KB<br>192 KB | Used in 3Qs, 4 views<br>Used in 2Qs, 1 view<br>Used in Q1<br>Used in Q5 |
| Indices | idx_courseRegistrations<br>idx_gender | Q0 on server ~ 302s<br>Q2 locally ~ 1:10 | 2407 MB<br>86MB | Q0 on server ~ 0s<br>Q2 locally ~ 1:05 |

*Table 1: Performance and space of chosen optimizations for the database (experiments locally)*