2ID70 MILESTONE 1 (Group 21)

t.tatsumi(0980989), Jonas Martin Niederle(1256157), Thomas Jorden Quadt(1246747)

March 12, 2019

1 A:

step 1: *Loading data phase.* Adding primary/composite key to tables.
DegreeId as primary Integer key for Degrees table.
StudentId as primary Integer key for Students table.
StudentRegistrationId as primary Integer key for StudentRegistrationsToDegrees table.
TeacherId as primary Integer key for Teachers table.
CourseId as primary Integer key for Courses table.
CourseOfferId and TeacherId as composite key for TeacherAssignmentToCourses table.
CourseOfferId as primary Integer key for CourseOffers table.
CourseOfferId and StudentreRegistrationId as composite key for StudentAssistants table.
StudentRegistrationId and CourseOfferId as composite key for CourseRegistrations table.

step 2: *Create DegreeProgess table.*
Table with row value:
StudentRegistrationsToDegrees.StudentId
CourseRegistrations.StudentRegistrationId
GPA = SUM(CAST(Grade AS DECIMAL) * ECTS)/SUM(ECTS)
AcqECTS = SUM(ECTS)
TotalEcts

step 3: Create hash index *SREGsid_idx* on StudentRegistrationsToDegrees.StudentID

step 4: Create index *DP_sidSrID_index* on DegreeProgress.StudentRegistrationID and DegreeProgress.StudentID

step 5: Create index *DP_Grade_index* on DegreeProgress.GPA

step 6: Create index *CREG_coidSrid* on CourseRegistrations.CourseOfferId and CourseRegistrations.StudentRegistrationsId

2 B:(computation speed is average of three runs rounded up to the sensible value)

step 1: 4174MB to 6181MB. Performance differences measured based on the queriesAfterLoading(1):100.000ms to 15.000ms queriesAfterLoading(2):1750.000ms to 100.000ms qesriesAfterLoading(3):11000.000ms to 7000.000ms.

step 2: 6181MB to 6584MB. Original query7 from question2 was taking 250000ms to compute without utilization of this table. This step is responsible for reduction in around 219000ms on question2, query7. DegreeProgress table is made in order to speed up query2, query3 and query7. These queries requires computation of GPA for example which make the entire query slow. By crating this table we only have to do computation of the certain variable once which takes 180000ms to create.

step 3: 6584MB to 6839MB. 10000.000ms to 8000.000ms on question2, query1 with arbitrary studentid and degreeid. Hash index is used here since studentid on studentregistrationstodegrees is used to check equality and hash index is suitable for this use case. This hash index takes 13000ms to create. When there is no printing involved speed difference is 10x.

step 4: 6839MB to 6964MB. 23000.000ms(no change) on question2, query2 with arbitrary GPA. 34000.000ms(no change) on question2, query3. 31000.000ms(no change) on question2, query7. This index takes 7000ms to create.

step 5: 6946MB to 7116MB. 22500.000ms on question2, query2 with arbitrary GPA. 34000.000ms(no change) on question2, query3. 31000.000ms(no change) on question2, query7. This index takes 16000ms to create.

step 6: 7116MB to 8829MB. 8000ms to 7800ms on question2, query1. 2250ms(same) on question2, query2. 55000ms(same) on question2, query5. 22370ms(same) on question2, query6. 146000ms(same) on question2, query8. This index takes 120000ms to create.