Project 1: Student Database Management System

S.Shreya(2018CS10377), Jatin Munjal(2018CS10343), Gautami Kandhare(2018CS10333)

Submitted on : April 4, 2021

1 Section 1

Our student management system is aimed to be a common portal for students, professors and administrators to access course, grade and statistical data. This project is loosely based on IITD's eacademics portal. It also included Virtual Learning Environment data (vle) which shows interaction with Virtual learning platforms and resources.

1.1 ER Diagram

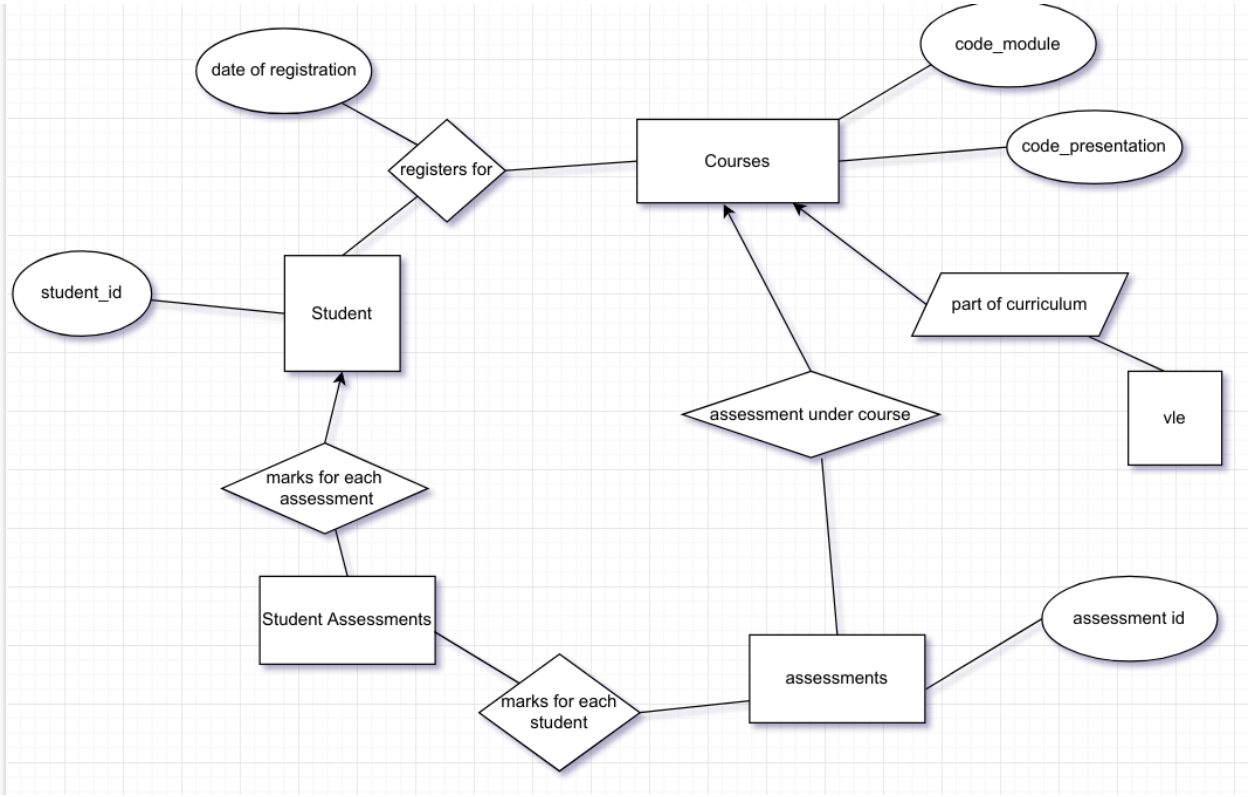

Figure 1: ER diagram.

1.2 Entity-Attributes Table

Attribute	Datatype
code module	character varying(45)
code presentation	character varying(45)
id student	integer
gender	character varying(3)
region	character varying(45)
highest education	character varying(45)
imd band	character varying(16)
age band	character varying(16)
num of prev attempts	integer
studied credits	integer
disability	character varying(3)
final result	character varying(45)

Table 1: Student Info

Attribute	Datatype
code module	character varying(45)
code presentation	character varying(45)
id assessment	integer
assessment type	character varying(45)
date	integer
weight	double precision

Table 2: Assessments

Attribute	Datatype
id assessment	integer
id student	integer
date submitted	integer
is banked	small int
score	double precision

Table 3: Student Assessments

Attribute	Datatype
code module	character varying(45)
code presentation	character varying(45)
module presentation length	integer

Table 4: Courses

Attribute	Datatype
code module	character varying(45)
code presentation	character varying(45)
id student	integer
date registration	integer
date unregistration	integer

Table 5: Student Registration

Attribute	Datatype
id site	integer
code module	character varying(45)
code presentation	character varying(45)
activity type	character varying(45)
week from	integer
week to	integer

Table 6: vle

Attribute	Datatype
code module	character varying(45)
code presentation	character varying(45)
id student	integer
id site	integer
date	integer
sum	big int

Table 7: Student vle

2 Section 2

2.1 Sources

1. The dataset was obtained from `https://analyse.kmi.open.ac.uk/open_dataset#data`
2. We downloaded the "ready made" dataset and performed some cleanup to make it more suitable for query execution

2.2 Cleanup Steps

In table student VLE (with no primary key defined) we combined rows based on their course-module, course-presentation, student id, site id and date and added the number of clicks(the number of times the site was visited).
We also replaced empty strings in the dataset with NULL values
No other cleanup was done.

2.3 Statistics

TableName	Number of Tuples	Load Time	Size of Dataset
studentvle	8459336	92593 ms	486 MB (raw size = 612 MB)
studentassessment	173912	1018 ms	14 MB
studentinfo	32593	1621 ms	4760 kB
studentregistration	32593	664 ms	2712 kB
vle	6364	561 ms	656 kB
assessments	206	177 ms	56 kB
courses	22	187 ms	24 kB

Table 8: Database statistics

3 Section 3

3.1 User View

Home page : includes 3 login options to choose from - Student, Professor and Admin

1. Login pages
   (a) Student Login Page : Checks if student id exists in the studentInfo relation
   (b) Professor Login Page : Enter code-module (Course Name) and code-presentation (Course Yr and Sem - "B" for the presentation starting in February and "J" for the presentation starting in October.) - Checks if course exists in courses relation
   (c) Admin - asks for admin password (Password set as 'root') - checks password which is set in the back-end code.
2. Student Home Page
   (a) Display courses taken by student
   (b) View Gradesheet
   (c) View All assessments of Chosen course
   (d) Withdraw from course
   (e) View CGPA
   (f) Check Student Ranking in chosen course
3. Professor Home Page
   (a) See list of students in course - Order by (student id, imd-band(poverty index),age-band or total credits)
   (b) Display student lists based on Course results - Distinction, Pass, Fail and Withdrawn
   (c) Diversity lists - Display lists of students based on gender/disability with order by options (student id, imd-band(poverty index),age-band or total credits)
   (d) Student Statistics - displays comparison graphs on basis of region, gender and disability.
   (e) Display results of students for chosen assessment
   (f) Student details - Given student id, display individual student results in course
   (g) Student Grades - total assignment marks of all students
   (h) Student VLE engagement statistics - displays the average number of site visits by the students on different days.
   (i) Late submissions - view all late submissions of students for each assessment
   (j) Give new assessment cutoffs and view the list of students who will pass with the new cutoff values.
   (k) Add a new course
   (l) Add a new assessment for current course
   (m) Remove current course
4. Admin Home Page
   (a) Student Statistics - prints the number of students in each course
   (b) Number of students in each course - Diversity Statistics - number of students in each category of all courses in decreasing order within each category (gender, region,disability, final result)
   (c) Number of credits taken Statistics - list of students with the number of credits registered and number of credits completed

(d) Return course result percentage (highest to lowest) - returns percentage of students who have passed, failed, received distinction or withdrawn in all courses

(e) Displays pass percentage of all courses

(f) Returns number of students with distinction based on new distinction cutoff entered

(g) Enroll in course

(h) Check course log file

3.2 System View

- As mentioned in cleanup we have used views to create a logically correct table with a key
- We have used triggers in the following 2 ways :-
  - We used trigger course-log-t to create a course log table that will be updated with every insert, delete and update of any course with the date on which operation was performed.
  - We have used trigger prevent-redundancy to prevent a student from registering for the same course again.

Queries Run include the following

1. (a) Student Login Page :Checks if student id exists in the studentInfo relation

   (b) Professor Login Page : Enter code-module (Course Name) and code-presentation - Checks if course exists in courses relation

   (c) Admin - asks for admin password (Password set as 'root') - checks password which is set in the back-end code.

2. Queries for student home page

```
global TABLE, TITLE
TITLE = ['id_student','total_marks']
TABLE = crs.fetchall()
```

Figure 2: 2a

(a)

```
with t as (
SELECT code_module, code_presentation,SUM(SA.score * A.weight)/100 as
Total_assignment_marks
FROM studentAssessment as SA, assessments as A
WHERE id_student = argv[1]
AND SA.id_assessment = A.id_assessment AND (A.assessment_type = 'CMA' OR A.assessment_type
= 'TMA')
GROUP BY code_module, code_presentation
ORDER BY SUM(SA.score * A.weight)/100 DESC
)
select t.code_module, t.code_presentation,t.Total_assignment_marks as total_marks,case
WHEN t.Total_assignment_marks  > 90 THEN 'A'
            WHEN t.Total_assignment_marks> 80 THEN 'B'
            WHEN t.Total_assignment_marks  > 70 THEN 'C'
            WHEN t.Total_assignment_marks  > 60 THEN 'D'
            WHEN t.Total_assignment_marks  > 40 THEN 'E'
            ELSE 'F'
      END
from t
```

Figure 3: 2b

(b)

```
select a.assessment_type, a.id_assessment, sa.score, sa.date_submitted, a.date as deadline
from assessments as a, studentAssessment as sa
where sa.id_student = argv[1]
and a.code_module = 'argv[2]'
and a.code_presentation = 'argv[3]'
and sa.id_assessment = a.id_assessment;
```

Figure 4: 2c

(c)

```
DELETE FROM studentInfo
WHERE id_student = argv[1] AND code_module = 'argv[2]' AND code_presentation = 'argv[3]'
AND final_result = NULL;

--this triggers this --
UPDATE studentRegistration
SET date_unregistration = argv[4]
WHERE id_student = argv[1] AND code_module = 'argv[2]' AND code_presentation = 'argv[3]'
```

Figure 5: 2d

(d)

```
with t as (
SELECT code_module, code_presentation,SUM(SA.score * A.weight)/100 as Total_assignment_marks
FROM studentAssessment as SA, assessments as A
WHERE id_student = argv[1] AND SA.id_assessment = A.id_assessment AND (A.assessment_type = 'CMA' OR A.
assessment_type = 'TMA')
GROUP BY code_module, code_presentation
ORDER BY SUM(SA.score * A.weight)/100 DESC
) ,
t2 as (
select t.code_module, t.code_presentation, t.Total_assignment_marks, case WHEN t.Total_assignment_marks > 90 THEN
'A'
            WHEN t.Total_assignment_marks  > 80 THEN 'B'
            WHEN t.Total_assignment_marks  > 70 THEN 'C'
            WHEN t.Total_assignment_marks  > 60 THEN 'D'
            WHEN t.Total_assignment_marks > 40 THEN 'E'
            ELSE 'F'
      END
from t
),
t3 as (
select t2.*, case when t2.case = 'A' then 10 when t2.case= 'B' then 9 when t2.case= 'C' then 8 when t2.case= 'D'
then 7 when t2.case= 'E' then 6 ELSE 0 end grade_point
from t2)
select round(avg(t3.grade_point),2) as gpa from t3 ;
```

Figure 6: 2e

(e)

```
with t as(
    SELECT row_number() over (ORDER BY (SUM(SA.score * A.weight)/100) DESC) as rank, SA.id_student,  SUM(SA.score *
    A.weight)/100 as Total_assignment_marks

FROM studentAssessment as SA, assessments as A
WHERE A.code_module ='argv[2]'  AND A.code_presentation = 'argv[3]'
AND SA.id_assessment = A.id_assessment AND (A.assessment_type = 'CMA' OR A.assessment_type = 'TMA')

GROUP BY SA.id_student
ORDER BY SUM(SA.score * A.weight)/100 DESC, SA.id_student
)
select rank
from t
where id_student = argv[1];
```

Figure 7: 2f

(f)

3. Queries for Professor home page

```
SELECT id_student,gender,region,highest_education,  imd_band,   age_band,   num_of_prev_attempts,   studied_credits,
    disability, final_result
FROM studentInfo as si
WHERE code_module = 'argv[1]' AND code_presentation = 'argv[2]'
ORDER BY argv[3] ASC;
```

Figure 8: 3a

(a)

```
SELECT id_student,gender,region,highest_education,  imd_band,   age_band,   num_of_prev_attempts,   studied_credits,
    disability, final_result
FROM studentInfo
WHERE code_module = 'argv[1]' AND code_presentation = 'argv[2]'
AND final_result = 'argv[3]'
ORDER BY id_student ASC;
```

Figure 9: 3b

(b)

```
SELECT id_student,gender,region,highest_education,  imd_band,   age_band,   num_of_prev_attempts,   studied_credits,
    disability, final_result
FROM studentInfo
WHERE code_module = 'argv[1]'
AND code_presentation = 'argv[2]'
AND argv[3] = 'argv[4]'
ORDER BY argv[5];
```

Figure 10: 3c

(c)

```
SELECT argv[3], COUNT(*)
FROM studentInfo
WHERE code_module = 'argv[1]' AND code_presentation = 'argv[2]'
GROUP BY argv[3]
ORDER BY count DESC;
```

Figure 11: 3d

(d)

```sql
SELECT SA.id_student,SA.is_banked, SA.date_submitted,(SA.date_submitted - A.date) as days_late, SA.score
FROM studentAssessment as SA, assessments as A, studentInfo as SI
WHERE SI.code_module = 'argv[1]' AND SI.code_presentation = 'argv[2]'
AND SA.id_assessment = 'argv[3]' AND SA.id_student = SI.id_student
AND SA.id_assessment = A.id_assessment AND SI.code_module = A.code_module AND SI.code_presentation = A.
code_presentation
AND SA.date_submitted > A.date;
```

Figure 12: 3e

(e)

```sql
SELECT * FROM StudentInfo
WHERE code_module = 'argv[1]'
AND code_presentation = 'argv[2]'
AND id_student = argv[3];

SELECT SA.*,A.date as last_date,A.weight as weightage, A.assessment_type
FROM studentAssessment as SA, assessments as A
WHERE A.code_module = 'argv[1]'
AND A.code_presentation = 'argv[2]'
AND SA.id_student = argv[3]
AND SA.id_assessment = A.id_assessment
ORDER BY A.date ASC;
```

Figure 13: 3f

(f)

```sql
with t as (
SELECT SA.id_student,  SUM(SA.score * A.weight)/100 as Total_assignment_marks
FROM studentAssessment as SA, assessments as A
WHERE A.code_module = 'argv[1]' AND A.code_presentation = 'argv[2]'
AND SA.id_assessment = A.id_assessment AND (A.assessment_type = 'CMA' OR A.assessment_type = 'TMA')

GROUP BY SA.id_student
ORDER BY SUM(SA.score * A.weight)/100 DESC, SA.id_student
),
t1 as (
SELECT SA.id_student, avg(SA.score) as Total_assignment_marks
FROM studentAssessment as SA, assessments as A
WHERE A.code_module = 'argv[1]' AND A.code_presentation = 'argv[2]'
AND SA.id_assessment = A.id_assessment AND A.assessment_type = 'Exam'

GROUP BY SA.id_student
ORDER BY avg(SA.score) DESC, SA.id_student
)
select t.id_student, (t.Total_assignment_marks+t1.Total_assignment_marks)/2 as total_marks
from t, t1
where t.id_student = t1.id_student;
```

Figure 14: 3g

(g)

```sql
SELECT id_assessment,date,assessment_type
FROM assessments
WHERE code_module = 'argv[1]' AND code_presentation = 'argv[2]';

SELECT SV.code_module,SV.code_presentation,SV.id_site,SV.date, ROUND(cast(SUM(SV.sum) as decimal)/COUNT(DISTINCT SV.
id_student),2) as avg_sum_clicks
FROM studentVle as SV
WHERE SV.code_module = argv[1] AND SV.code_presentation = argv[2] AND SV.id_site = argv[3]
GROUP BY SV.code_module,SV.code_presentation,SV.id_site,SV.date;
```

Figure 15: 3h

(h)

```
select a.id_assessment,sa.id_student,sa.date_submitted,(sa.date_submitted-a.date) as late_by,sa.score
from studentAssessment as sa, assessments as a
where sa.id_assessment = a.id_assessment and sa.date_submitted > a.date and code_module = 'argv[1]' and
code_presentation = 'argv[2]'
order by  a.id_assessment ASC,(sa.date_submitted-a.date) DESC;
```

Figure 16: 3i

(i)

```
SELECT id_student
FROM studentAssessment as SA, assessments as A
WHERE A.code_module = 'argv[1]' AND A.code_presentation = 'argv[2]'
AND SA.id_assessment = A.id_assessment
AND ((score >= argv[3] AND assessment_type = 'TMA') OR (score >= argv[4] AND assessment_type = 'CMA') OR (score >=
argv[5] AND assessment_type = 'Exam'))
GROUP BY SA.id_student
HAVING COUNT(*) = (SELECT COUNT(*) FROM assessments WHERE  code_module = 'argv[1]' AND code_presentation = 'argv[2]
')
ORDER BY SA.id_student;
```

Figure 17: 3j

(j)

```
INSERT INTO courses VALUES ('argv[1]', 'argv[2]', argv[3]);
```

Figure 18: 3k

(k)

```
INSERT INTO assessments VALUES ('argv[1]','argv[2]',argv[3],'argv[4]',argv[5],argv[6]);
```

Figure 19: 3l

(l)

```
DELETE FROM courses WHERE code_module = 'argv[1]' AND code_presentation = 'argv[2]';
```

Figure 20: 3m

(m)

4. Queries for admin home page

```
select code_module,code_presentation,count(id_student)
from studentInfo

group by code_module, code_presentation
order by count(id_student) desc
```

Figure 21: 4a

(a)

```
SELECT code_module, code_presentation, argv[1], COUNT(*)
FROM studentInfo
GROUP BY code_module, code_presentation, argv[1]
ORDER BY argv[1] ASC, count DESC,code_module, code_presentation;
```

Figure 22: 4b

(b)

```
with t as (
select id_student, count(*)*3 as credits_registered
from studentRegistration
where date_unregistration is null
group by id_student
)
select t.*, si.studied_credits
from t, studentInfo as si
where t.id_student = si.id_student;
```

Figure 23: 4c

(c)

```
with t1 as (
    select code_module,code_presentation,count(id_student)
    from studentInfo
    where final_result = 'argv[1]'
    group by code_module, code_presentation
),
t2 as (
    select code_module,code_presentation,count(id_student)
    from studentInfo
    group by code_module, code_presentation
)
select t1.code_module, t1.code_presentation, round ( cast(t1.count as numeric) / cast(t2.count as numeric), 2) as
rate
from t1,t2
where t1.code_module=t2.code_module
and t1.code_presentation=t2.code_presentation
order by rate desc;
```

Figure 24: 4d

(d)

```
with t as (
select code_module,code_presentation,count(id_student) as num
from studentInfo

group by code_module, code_presentation
order by count(id_student) desc  ),
t2 as (

select code_module,code_presentation, count(id_student) as num
from studentInfo
where final_result = 'Pass'
group by code_module, code_presentation
order by count(id_student) desc
)

 select t2.code_module, t2.code_presentation, (t2.num * 100) /t.num  as percent
from t, t2
where t.code_module = t2.code_module and t.code_presentation = t2.code_presentation
```

Figure 25: 4e

(e)

```
with t as (
SELECT SA.id_student, A.code_module, A.code_presentation, (SUM(SA.score * A.weight))/100 as Total_assignment_marks
FROM studentAssessment as SA, assessments as A
WHERE SA.id_assessment = A.id_assessment AND (A.assessment_type = 'CMA' OR A.assessment_type = 'TMA')
AND SA.score IS NOT NULL
GROUP BY sa.id_student, A.code_module, A.code_presentation
ORDER BY SUM(SA.score * A.weight)/100 DESC )

select code_module,code_presentation,count(*)
from t
where Total_assignment_marks  > 75
group by code_module, code_presentation
order by count desc
```

Figure 26: 4f

(f)

```
INSERT INTO studentInfo VALUES ('argv[1]','argv[2]',argv[3],'argv[4]','argv[5]','argv[6]','argv[7]','argv[8]',argv
[9],argv[10],'argv[11]',NULL);
INSERT INTO studentRegistration VALUES ('argv[1]','argv[2]',argv[3],argv[12],NULL);
```

Figure 27: 4g

(g)

```
select *
from course_log;
```

Figure 28: 4h

(h)

3.3 Tested Query Examples

Query number	Parameter values	Rows in output	Sample run time
2(a)	584077	5	0.00503
2(b)	584077	4	0.054243
2(c)	584077, CCC, 2014B	3	0.041875
2(d)	12345, XXX, 2015J, 20	0	0.079914
2(e)	11391	1	0.035985
2(f)	11391, AAA, 2013J	1	0.144634
3(a)	id_student	383	0.10742
3(b)	AAA, 2013J, Pass	258	0.037219
3(c)	AAA, 2013J, gender, F, id_student	149	0.040614
3(d)	AAA, 2013J, region	13	0.107508
3(e)	AAA, 2013J, 1752	66	0.211304
3(f)	AAA, 2013J, 11391	5	0.004175
3(g)	CCC, 2014B	747	0.049603
3(h)	AAA, 2013J, 546614	6, 279	7.104088
3(i)	AAA , 2013J	386	0.038531
3(j)	DDD, 2013B, 30, 30, 30	287	0.077897
3(k)	XXX, 2015J, 100	0	0.04322
3(l)	XXX, 2015J, 99999, TMA, 10, 10	0	0.2471
3(m)	XXX, 2015J	0	0.126864
4(a)	-	22	0.335266
4(b)	region	282	0.042465
4(c)	-	24132	0.102791
4(d)	fail	22	0.071282
4(e)	-	22	0.127746
4(f)	75	19	0.213346
4(g)	XXX, 2015J, 12345	0	0.298292
4(h)	-	26	0.001234

Table 9: Tested Queries