UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

Module

CS4416 DATABASE SYSTEMS

Project

Educational Administration Management System (EDMS)

Team Members

Yaoting Wang - 19107668

Siming Zheng - 19108958

Yuchen Wang - 19116098

Fengyuan Zhang - 19114613

Catalog

1. About the Database

1.1 Introduction

As a modern teaching technology, the educational administration management system is increasingly valued by universities. It is an indispensable part of the university. This database system is designed about to manage the information of university people, colleges and modules. The system would standardise, systematise and program, improve the speed and accuracy of information processing, and accurately, timely and effectively query and modify the enroll info.

This well-designed database system conforms to the integrity rules. Firstly, it is more convenient for the system design and coding to developers; secondly, it is also easier for the later maintenance of the system. A well-designed database system could ensure the scalability and the transplantation.

Through this database, the administrator could use the functions such as add, delete, update and search the modules, lecturers and students etc. Also, students could log in to and query the basic information of the modules, and implement the modules selection.

1.2 Composition

(1) colleges(<u>college_id</u>, college_name, type):

Represents the college id, college name, college type.

(2) students(<u>id</u>, first_name, last_name, gender, college_id, grade):

Represents the student id, the first & last name of student, and their gender, studies in which college, and their grade.

(3) teacher(<u>lecturer_id</u>, firstname, lastname, gender, title, e-mail, college_id):

Represents lecturer id, first & last name, gender, title, email and works in which college.

(4) module(<u>module_id</u>, period, credit, optional, lecturer_id):

Represents the id, period, credit of this module, optional or not, which lecturer holds the module.

(5) enroll(enroll_date, <u>module_id, student_id</u>):

Represents the date and the id of module and student.

(6) deleted_students(<u>student_id</u>, first_name, last_name):

Represents the id and first & last name of students who has been deleted.

**PKs have been <u>underlined</u>.*

1.3 Concise Description

(1) This database could form a convenient and quick educational administration management system by including main teaching members, university components and necessary registration records. By using this database, you could add, delete, update and check college information, student information, lecturer information, course information and registration information.

(2) The database system uses foreign key restriction to ensure security and data consistency and integrity.

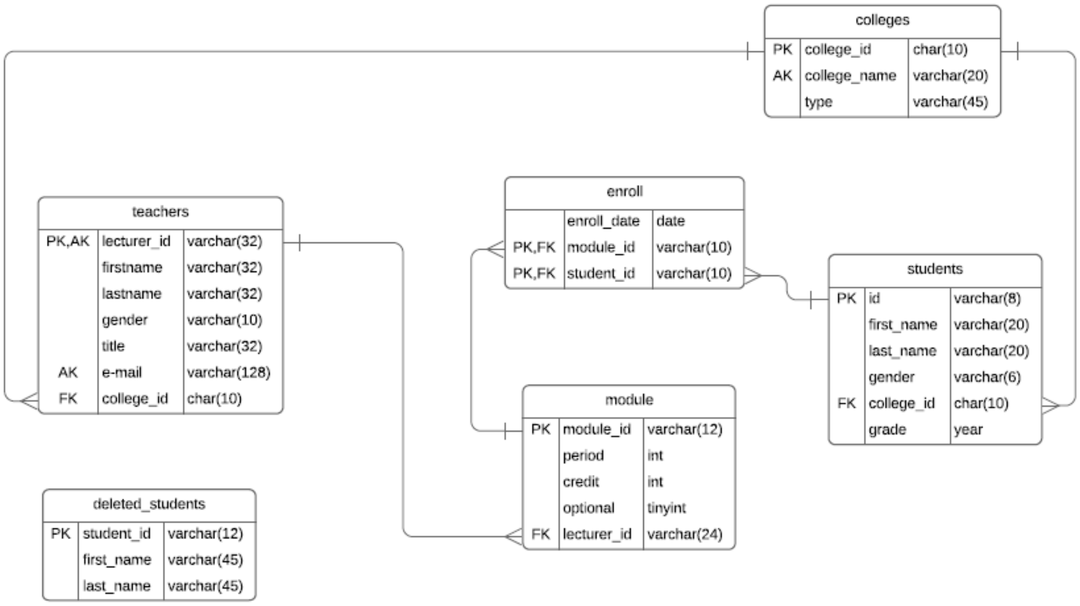
(3) This database system uses the index, which could be used to improve the speed of searching.

(4) The database has four views. Some common functions could be used directly (see 6 for details)

(5) This database has three triggers, one function and one stored procedure (see 9 for details)

(6) This database could be used by the educational administration management system, which could perform fundamental and important operations, such as searching students, lecturers, modules and registration records.

2. Entity-relationship diagram



colleges

PK	college_id	char(10)
AK	college_name	varchar(20)
	type	varchar(45)

teachers

PK,AK	lecturer_id	varchar(32)
	firstname	varchar(32)
	lastname	varchar(32)
	gender	varchar(10)
	title	varchar(32)
AK	e-mail	varchar(128)
FK	college_id	char(10)

enroll

	enroll_date	date
PK,FK	module_id	varchar(10)
PK,FK	student_id	varchar(10)

students

PK	id	varchar(8)
	first_name	varchar(20)
	last_name	varchar(20)
	gender	varchar(6)
FK	college_id	char(10)
	grade	year

module

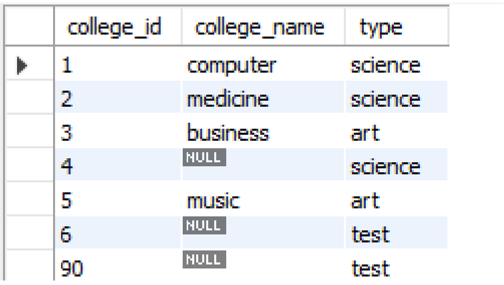
PK	module_id	varchar(12)
	period	int
	credit	int
	optional	tinyint
FK	lecturer_id	varchar(24)

deleted_students

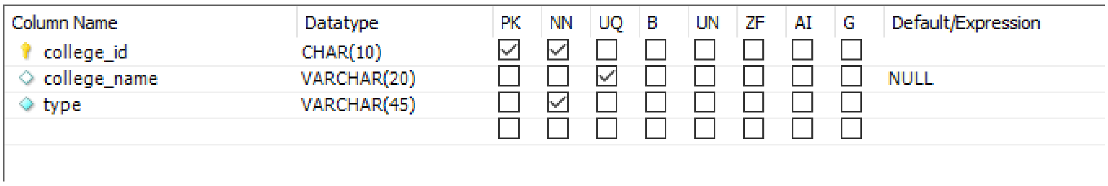
PK	student_id	varchar(12)
	first_name	varchar(45)
	last_name	varchar(45)

3. Example Data Screenshot

- colleges:

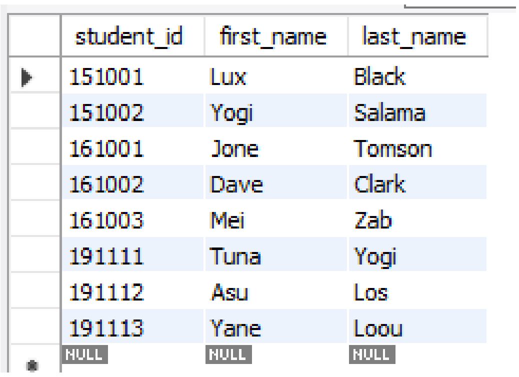


college_id	college_name	type
1	computer	science
2	medicine	science
3	business	art
4	NULL	science
5	music	art
6	NULL	test
90	NULL	test

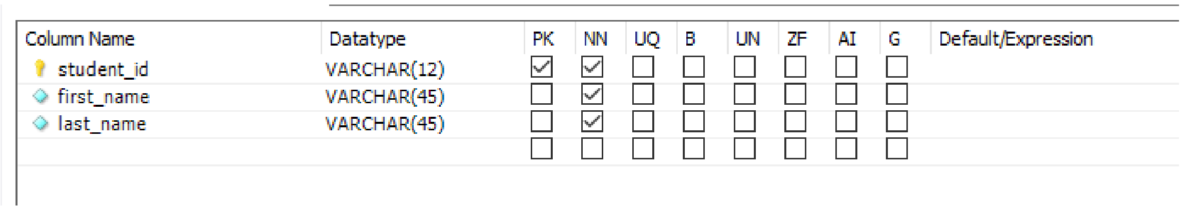


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
college_id	CHAR(10)	☑	☑	□	□	□	□	□	□	
college_name	VARCHAR(20)	□	□	☑	□	□	□	□	□	NULL
type	VARCHAR(45)	□	☑	□	□	□	□	□	□	
		□	□	□	□	□	□	□	□	

- deleted_students:

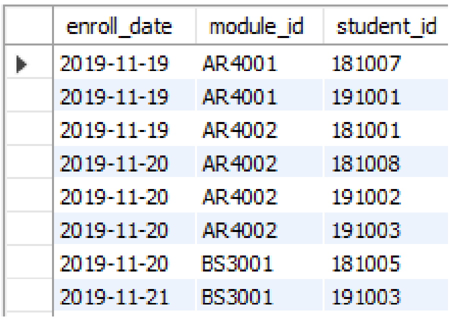


student_id	first_name	last_name
151001	Lux	Black
151002	Yogi	Salama
161001	Jone	Tomson
161002	Dave	Clark
161003	Mei	Zab
191111	Tuna	Yogi
191112	Asu	Los
191113	Yane	Loou
NULL	NULL	NULL

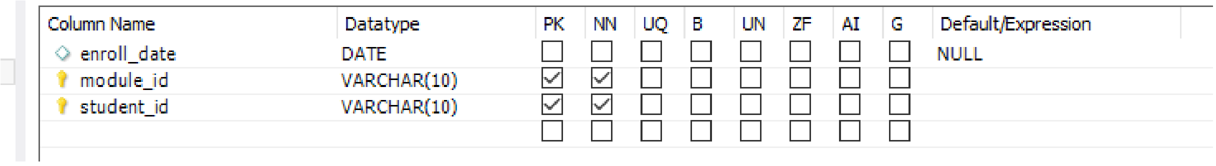


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
student_id	VARCHAR(12)	☑	☑	□	□	□	□	□	□	
first_name	VARCHAR(45)	□	☑	□	□	□	□	□	□	
last_name	VARCHAR(45)	□	☑	□	□	□	□	□	□	
		□	□	□	□	□	□	□	□	

- enroll:

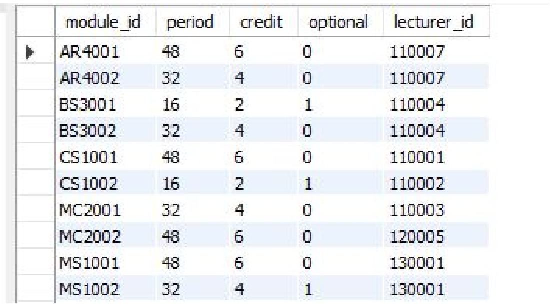


enroll_date	module_id	student_id
2019-11-19	AR4001	181007
2019-11-19	AR4001	191001
2019-11-19	AR4002	181001
2019-11-20	AR4002	181008
2019-11-20	AR4002	191002
2019-11-20	AR4002	191003
2019-11-20	BS3001	181005
2019-11-21	BS3001	191003

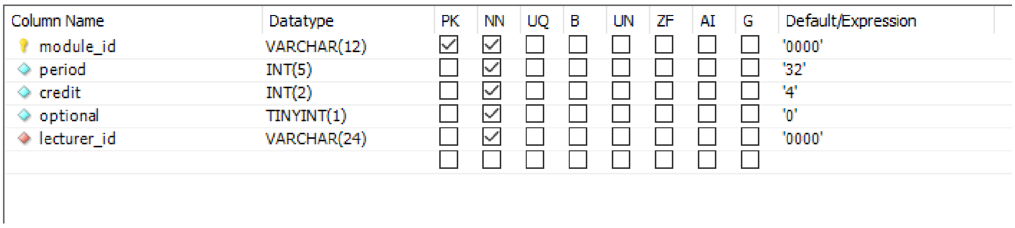


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
enroll_date	DATE	□	□	□	□	□	□	□	□	NULL
module_id	VARCHAR(10)	☑	☑	□	□	□	□	□	□	
student_id	VARCHAR(10)	☑	☑	□	□	□	□	□	□	
		□	□	□	□	□	□	□	□	

- module:

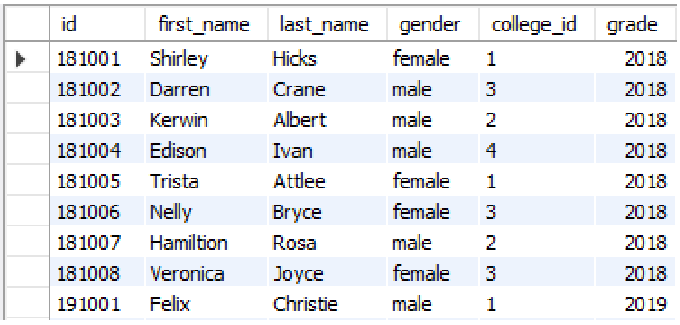


module_id	period	credit	optional	lecturer_id
AR4001	48	6	0	110007
AR4002	32	4	0	110007
BS3001	16	2	1	110004
BS3002	32	4	0	110004
CS1001	48	6	0	110001
CS1002	16	2	1	110002
MC2001	32	4	0	110003
MC2002	48	6	0	120005
MS1001	48	6	0	130001
MS1002	32	4	1	130001

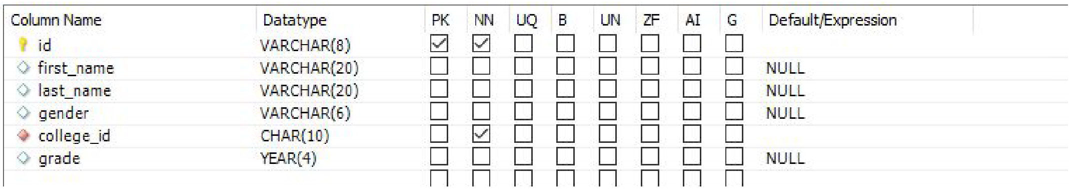


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
module_id	VARCHAR(12)	☑	☑	□	□	□	□	□	□	'0000'
period	INT(5)	□	☑	□	□	□	□	□	□	'32'
credit	INT(2)	□	☑	□	□	□	□	□	□	'4'
optional	TINYINT(1)	□	☑	□	□	□	□	□	□	'0'
lecturer_id	VARCHAR(24)	□	☑	□	□	□	□	□	□	'0000'
		□	□	□	□	□	□	□	□	

- students:

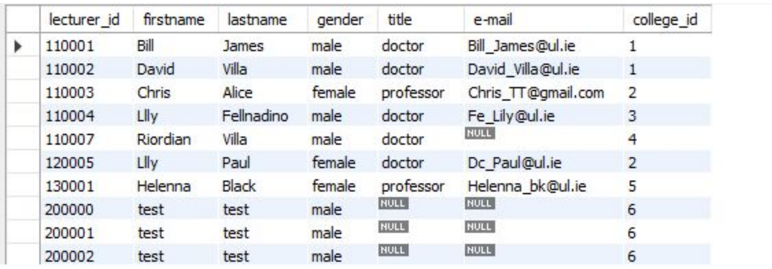


id	first_name	last_name	gender	college_id	grade
181001	Shirley	Hicks	female	1	2018
181002	Darren	Crane	male	3	2018
181003	Kerwin	Albert	male	2	2018
181004	Edison	Ivan	male	4	2018
181005	Trista	Attlee	female	1	2018
181006	Nelly	Bryce	female	3	2018
181007	Hamiltion	Rosa	male	2	2018
181008	Veronica	Joyce	female	3	2018
191001	Felix	Christie	male	1	2019

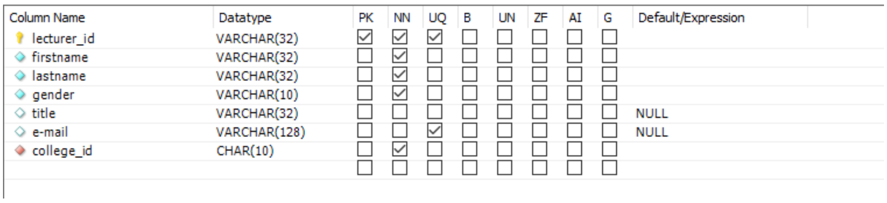


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	VARCHAR(8)	☑	☑	□	□	□	□	□	□	
first_name	VARCHAR(20)	□	□	□	□	□	□	□	□	NULL
last_name	VARCHAR(20)	□	□	□	□	□	□	□	□	NULL
gender	VARCHAR(6)	□	□	□	□	□	□	□	□	NULL
college_id	CHAR(10)	□	☑	□	□	□	□	□	□	
grade	YEAR(4)	□	□	□	□	□	□	□	□	NULL
		□	□	□	□	□	□	□	□	

- teachers:



lecturer_id	firstname	lastname	gender	title	e-mail	college_id
110001	Bill	James	male	doctor	Bill_James@ul.ie	1
110002	David	Villa	male	doctor	David_Villa@ul.ie	1
110003	Chris	Alice	female	professor	Chris_TT@gmail.com	2
110004	Llly	Fellnadino	male	doctor	Fe_Lily@ul.ie	3
110007	Riordian	Villa	male	doctor	NULL	4
120005	Llly	Paul	female	doctor	Dc_Paul@ul.ie	2
130001	Helenna	Black	female	professor	Helenna_bk@ul.ie	5
200000	test	test	male	NULL	NULL	6
200001	test	test	male	NULL	NULL	6
200002	test	test	male	NULL	NULL	6



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
lecturer_id	VARCHAR(32)	☑	☑	☑	□	□	□	□	□	
firstname	VARCHAR(32)	□	☑	□	□	□	□	□	□	
lastname	VARCHAR(32)	□	☑	□	□	□	□	□	□	
gender	VARCHAR(10)	□	☑	□	□	□	□	□	□	
title	VARCHAR(32)	□	□	□	□	□	□	□	□	NULL
e-mail	VARCHAR(128)	□	□	☑	□	□	□	□	□	NULL
college_id	CHAR(10)	□	☑	□	□	□	□	□	□	
		□	□	□	□	□	□	□	□	

4. FDs list

college
college_id -> college_name
college_id -> type

deleted_students
student_id -> first_name
student_id -> last_name

enroll
module_id, student_id -> enroll_date

module
module_id -> period
module_id -> credit
module_id -> optional
module_id -> lecturer_id

students
id -> first_name
id -> last_name
id -> gender
id -> college_id
id -> grade

teachers
lecturer_id -> firstname
lecturer_id -> lastname
lecturer_id -> gender
lecturer_id -> title
lecturer_id -> e-mail
lecturer_id -> college_id

5. Proof 3NF

5.1 college:

1NF:

- There is only one single value for each intersection of column and row.
- All values in the same column are of same type.
- college_id is the primary key which is unique and not null.

2NF:

- In the 1NF.
- Attributes college_name and type are not the keys, and they depend on the PK college_id.

3NF:

- In the 1NF & 2NF.
- No transitive dependencies existed in college_id -> college_name, college_id -> type.

```
*By FDs
   step 0: the PK is college_id, so the prime is college_id.
                college_id -> college_name,
                college_id -> type,
           the LHS attribute is key, and could not be removed and
           the FDs could not be removed as well. So, the 3NF is
           tenable.
```

5.2 deleted_students

1NF:

- There is only one single value for each intersection of column and row.
- All values in the same column are of same type.
- student_id is the primary key which is unique and not null.

2NF:

- In the 1NF.
- Attributes first_name and last_name are not the keys, and they depend on the PK student_id.

3NF:

- In the 1NF & 2NF.
- No transitive dependencies existed in student_id -> first_name, student_id -> last_name.

```
*By FDs
   step 0: the PK is student_id, so the prime is student_id.
                student_id -> first_name,
                student_id -> last_name,
           the LHS attribute is key, and could not be removed and
           the FDs could not be removed as well. So, the 3NF is
           tenable.
```

5.3 enroll

1NF:

- There is only one single value for each intersection of column and row.
- All values in the same column are of same type.
- The combination of (module_id, student_id) is the primary key which is unique and not null.

2NF:

- In the 1NF.
- Attributes enroll_date are not the keys, and they depend on the PK (module_id, student_id).

3NF:

- In the 1NF & 2NF.
- No transitive dependencies existed as there is only the FD module_id, student_id -> enroll_date.

***by FDs**

```
step 0: the PK is student_id & module _id so the prime is
        module_id, student_id.
             module_id, student_id -> enroll_date
        the LHS attributes are superkey, and could not be
        removed and the FDs could not be removed as well. So,
        the 3NF is tenable.
```

5.4 module

1NF:

- There is only one single value for each intersection of column and row.
- All values in the same column are of same type.
- module_id is the primary key which is unique and not null.

2NF:

- In the 1NF.
- Attributes period, credit, optional and lecturer_id are not the keys, and they depend on the PK module_id.

3NF:

- In the 1NF & 2NF.
- No transitive dependencies existed in module_id -> period, module_id -> credit, module_id -> optional, module_id -> lecturer_id.

***by FDs**

```
step 0: the PK is module _id so the prime is module_id;
             module_id -> period,
             module_id -> credit,
             module_id -> optional,
                module_id -> lecturer_id,
          the LHS attribute is key, and could not be removed and
          the FDs could not be removed as well. So, the 3NF is
          tenable.
```

5.5 students

1NF:

- There is only one single value for each intersection of column and row.
- All values in the same column are of same type.
- id is the primary key which is unique and not null.

2NF:

- In the 1NF.
- Attributes first_name, last_name, gender, college_id and grade are not the keys, and they depend on the PK id.

3NF:

- In the 1NF & 2NF.
- No transitive dependencies existed in id -> first_name, id -> last_name, id -> gender, id -> college_id, id -> grade.

***By FDs**

```
   step 0: the PK is id, so the prime is id.
                id -> first_name,
                id -> last_name,
                id -> gender,
                id -> college_id,
                id -> grade,
          the LHS attribute is key, and could not be removed and
          the FDs could not be removed as well. So, the 3NF is
          tenable.
```

5.6 teachers

1NF:

- There is only one single value for each intersection of column and row.
- All values in the same column are of same type.
- lecturer_id is the primary key which is unique and not null.

2NF:

- In the 1NF.
- Attributes first_name, last_name, gender, title, email and college_id are not the keys, and they depend on the PK id.

3NF:

- In the 1NF & 2NF.
- No transitive dependencies existed in lecturer_id -> first_name, lecturer_id -> last_name, lecturer_id -> gender, lecturer_id -> title, lecturer_id -> email, lecturer_id -> college_id.

```
*By FDs
   step 0: the PK is id, so the prime is id.
                lecturer_id -> first_name,
                lecturer_id -> last_name,
                lecturer_id -> gender,
                lecturer_id -> title,
                lecturer_id -> email,
                lecturer_id -> college_id,
           the LHS attribute is key, and could not be removed and
           the FDs could not be removed as well. So, the 3NF is
           tenable.
```

6. About the Views (usefulness)

6.1 stu_at_most_reg:

Function: It is used to display which students are enrolled in the most number enrolments days and as well as the modules they choose.

Justification: Universities could use EDMS with this query to analyse which modules are most favoured by students.

6.2 lec_teach_two:

Functions: It is used to query the names of all teachers who teach a course that is selected by more than 2 students.

Justification: Lecturers need to teach a corresponding number of lessons each school year. At the end of the module selection, the manager could use EDMS to check whether the total lesson time in the lecturer’s plan meets the corresponding requirements, and the manager needs to arrange corresponding teaching tasks for lecturers who do not have enough lesson time in the plan.

6.3 stu_credit_big6:

Function: Find out the full name and total course credits of students whose total credits of modules are greater than or equal to 6.

Justification: Students need to take some modules every school year and get enough credits. University could use EDMS with this query to check which students will lack credits, and then help them arrange the rest of the modules.

6.4 stu_ time_less72:

Function: Find out students with less than 72 module hours.

Justification: In order to ensure effective education, students must have more than 72 hours of class time per semester. Through EDMS with this query, university could find out which students have less than 72 hours of class time, and then help them to arrange other courses or re-plan the module.

7. Analysis of Views

The combination of view and index could improve the query speed.

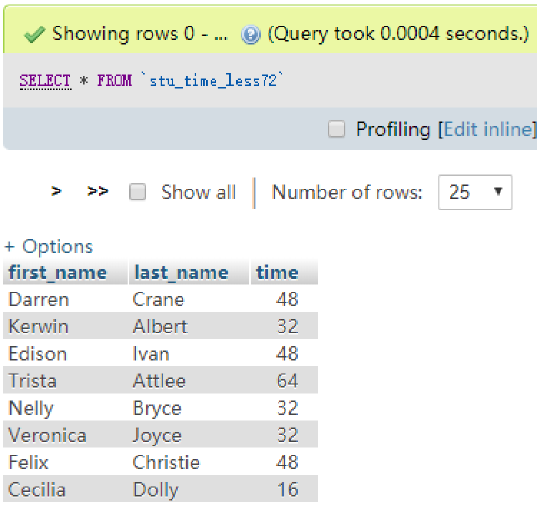
The view could simplify the query operation, but the view itself is just mainly used for convenience and security, and it would not greatly improve the query speed if only the view is used without indexes.

Thus, to have a quicker query, the indexes are needed. Once indexes are added to database tables, their query speed could be improved because the attribute column which has been indexed could be retrieved quicker and easier.

*Comparison of whether the view added the index:

1. stu_time_less72

Indexed



✓ Showing rows 0 - ... (Query took 0.0004 seconds.)

```
SELECT * FROM `stu_time_less72`
```

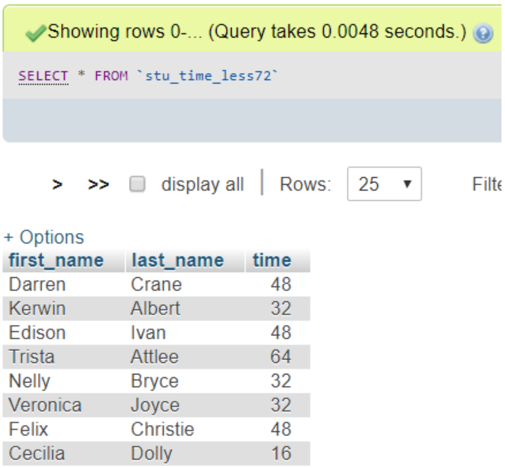
□ Profiling [Edit inline]

> >> □ Show all | Number of rows: 25 ▾

+ Options

first_name	last_name	time
Darren	Crane	48
Kerwin	Albert	32
Edison	Ivan	48
Trista	Attlee	64
Nelly	Bryce	32
Veronica	Joyce	32
Felix	Christie	48
Cecilia	Dolly	16

Unindexed



✓Showing rows 0-... (Query takes 0.0048 seconds.)

```
SELECT * FROM `stu_time_less72`
```

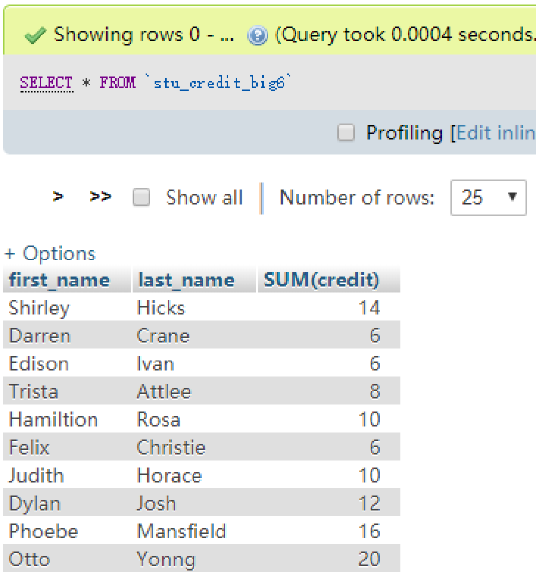
> >> □ display all | Rows: 25 ▾ Filte

+ Options

first_name	last_name	time
Darren	Crane	48
Kerwin	Albert	32
Edison	Ivan	48
Trista	Attlee	64
Nelly	Bryce	32
Veronica	Joyce	32
Felix	Christie	48
Cecilia	Dolly	16

2.stu_credit_big6

Indexed



✓ Showing rows 0 - ... (Query took 0.0004 seconds.

```
SELECT * FROM `stu_credit_big6`
```

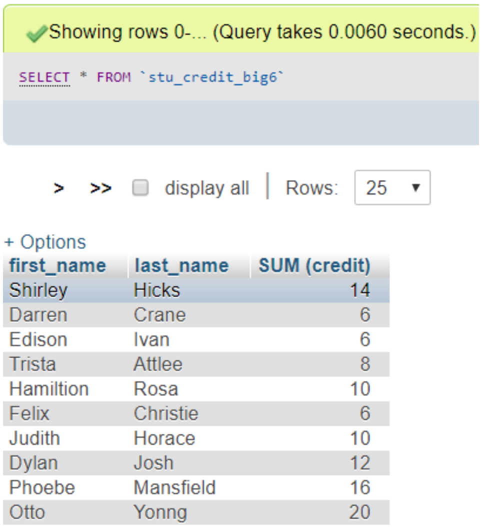
□ Profiling [Edit inlin

> >> □ Show all | Number of rows: 25 ▾

+ Options

first_name	last_name	SUM(credit)
Shirley	Hicks	14
Darren	Crane	6
Edison	Ivan	6
Trista	Attlee	8
Hamiltion	Rosa	10
Felix	Christie	6
Judith	Horace	10
Dylan	Josh	12
Phoebe	Mansfield	16
Otto	Yonng	20

Unindexed



✓Showing rows 0-... (Query takes 0.0060 seconds.)

```
SELECT * FROM `stu_credit_big6`
```

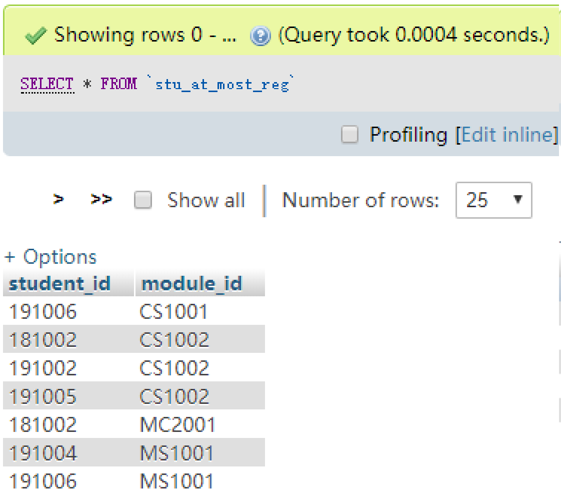
> >> □ display all | Rows: 25 ▾

+ Options

first_name	last_name	SUM (credit)
Shirley	Hicks	14
Darren	Crane	6
Edison	Ivan	6
Trista	Attlee	8
Hamiltion	Rosa	10
Felix	Christie	6
Judith	Horace	10
Dylan	Josh	12
Phoebe	Mansfield	16
Otto	Yonng	20

3.stu_at_most_reg

Indexed   Unindexed


✓ Showing rows 0 - ... (Query took 0.0004 seconds.)

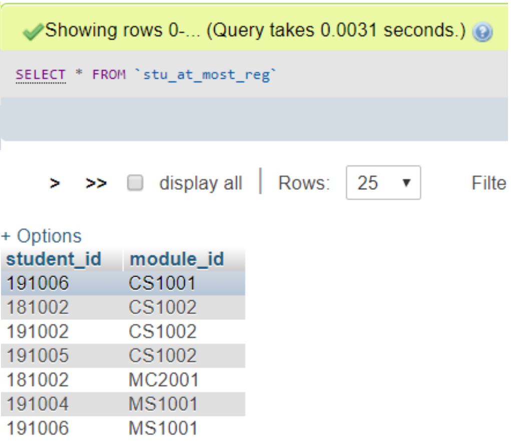
```
SELECT * FROM `stu_at_most_reg`
```

□ Profiling [Edit inline]

> >> □ Show all | Number of rows: 25

+ Options

student_id	module_id
191006	CS1001
181002	CS1002
191002	CS1002
191005	CS1002
181002	MC2001
191004	MS1001
191006	MS1001


✓Showing rows 0-... (Query takes 0.0031 seconds.)

```
SELECT * FROM `stu_at_most_reg`
```

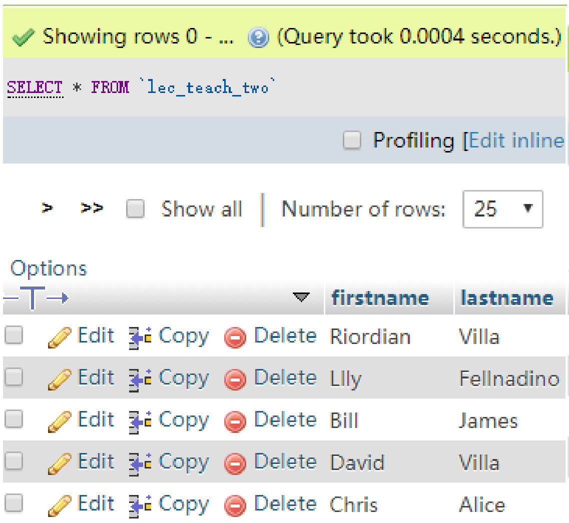
> >> □ display all | Rows: 25 Filte

+ Options

student_id	module_id
191006	CS1001
181002	CS1002
191002	CS1002
191005	CS1002
181002	MC2001
191004	MS1001
191006	MS1001

4.lec_teach_two

Indexed   Unindexed


✓ Showing rows 0 - ... (Query took 0.0004 seconds.)

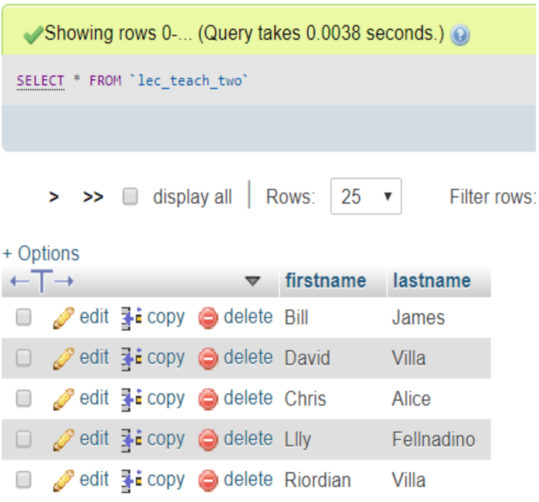
```
SELECT * FROM `lec_teach_two`
```

□ Profiling [Edit inline

> >> □ Show all | Number of rows: 25

Options

←T→				firstname	lastname
□	Edit	Copy	Delete	Riordian	Villa
□	Edit	Copy	Delete	Llly	Fellnadino
□	Edit	Copy	Delete	Bill	James
□	Edit	Copy	Delete	David	Villa
□	Edit	Copy	Delete	Chris	Alice


✓Showing rows 0-... (Query takes 0.0038 seconds.)

```
SELECT * FROM `lec_teach_two`
```

> >> □ display all | Rows: 25 Filter rows:

+ Options

←T→				firstname	lastname
□	edit	copy	delete	Bill	James
□	edit	copy	delete	David	Villa
□	edit	copy	delete	Chris	Alice
□	edit	copy	delete	Llly	Fellnadino
□	edit	copy	delete	Riordian	Villa

8. Analysis of Indexes

college: PK

college uses college_id as the index because id is the unique identifier and primary key, it could be quickly to retrieval.

deleted_students: PK

deleted_students uses student_id as the index because id is the unique identifier and primary key, it could be quickly to retrieval.

enroll: PK & fk_studentid_idx
The primary keys are module_id and student_id, and they could also be indexes as this table has a lot of data. After indexing the student IDs and module IDs, it will be faster to find which courses a student has chosen or which students are selected for a specific course.

Enroll has two foreign keys fk_moduleid which need to connect module_id in module and fk_studentid which need to connect id in students, but enroll usually queries with student id, so the table should use fk_studentid_idx to index the id in students. Thus, the queries between enroll & students could be quicker.

module: PK & fk_lecturerid_idx
The primary key is module_id, and it could also be an index as module ID is unique. Then we could quickly find information about a certain course.

Module has a foreign key fk_lecturerid which needs to connect lecturer_id in teachers, so the table should use index fk_lecturerid_idx to index the lecturer_id in teachers. Thus, the queries between module & teachers could be quicker.

students: PK & fk_collegeid_idx
The primary key is id, and it could also be an index as each student ID is unique. It is quick to find information about a student by ID.

Students has a foreign key fd_college_id which needs to connect college_id in colleges, so the table should use index fk_collegeid_idx to index college_id in colleges. Thus, the queries between students & colleges could be quicker.

teachers: PK & fk_collegeid_idx
The primary key is lecturer_id, and it could also be an index as each lecturer ID corresponds to a unique teacher, it is quick to find out information about a teacher.

Teachers has a foreign key fk_collegid which needs to connect college_id in colleges, so the table should use index fk_collegeid_idx to index college_id in colleges. Thus, the queries between teachers & colleges could be quicker

9. About Triggers, Procedure & Function

9.1 **Function:** f_add_to_deleted(f_id,f_first,f_last)

Function: could add students which have been deleted from students table into deleted_students table automatically.

Justification: When students graduate or are dropped out of school, the university will use EDMS to save a form to record which students are no longer in school.

9.2 Procedure: p_delete_relative_enroll(p_student_id)

Function: could delete relative enrolment records of students who have been deleted from students table.

Justification: When the student has been deleted from the student form of the university, the university should use EDMS to delete the relevant information in the module enrolment form, so as to ensure the timeliness and validity of the enroll information.

9.3 Trigger1: t_check_remain

Function: could call the function f_add_to_deleted(f_id,f_first,f_last) and the procedure p_delete_relative_enroll(p_student_id) once some students are deleted from students table.

Justification: this trigger occurs when deleting students' information. There are two operations for deleting students (record delete students, delete information in enrolment table). To reduce the complexity of operation, EDMS hides the details of the operation, and the administrator only needs to take the delete operation.

9.4 Trigger2: t_new_lecturer

Function: could add the new teachers who showed in the module table into teachers table.

Justification: when a new lecturer appears in the module info, EDMS could automatically add it to the teacher table which could hide the operation details and reduce the operation complexity.

9.5 Trigger3: t_re_enroll_stu

Function: could add those students who are deleted from deleted_students table into students table (which means a re-enrol for those deleted students).

Justification: when the students who are off of the university and enter the university again – which means re-enrol, EDMS will automatically transfer the important information of the students from the deleted_students table to the students table, simplifying the operation and making it more utilise.

10. Task assignment

Yaoting Wang: DB Design, View, Trigger, FDs & proof, Report, Scenario.
Siming Zheng: DB Design, View, Trigger, ERD design, Report, Data test.
Yucheng Wang: DB Design, View, Procedure, Introduction, Screenshot.
Fengyuan Zhang: DB Design, View, Function, Views & Index Analysis.