GAZI UNIVERSITY

ENGINEERING FACULTY

COMPUTER ENGINEERING

CENG316 - DATABASE SYSTEMS

FINAL

Gamze Aksu

171180005

JUNE 2021

A. The ER diagram created for Midterm (Shema 1) has been implemented with Microsoft SQL Server Management Studio 18. Two different databases were created for normalization. Of these, the CourseCredit database was created from the ER diagram in midterm. The other CourseCreditSystem is the normalized version of the CourseCredit database.

To create a CourseCredit database, the STUDENT table was created first. Student_id is the primary key of this table. STUDENT table is created to store student information.

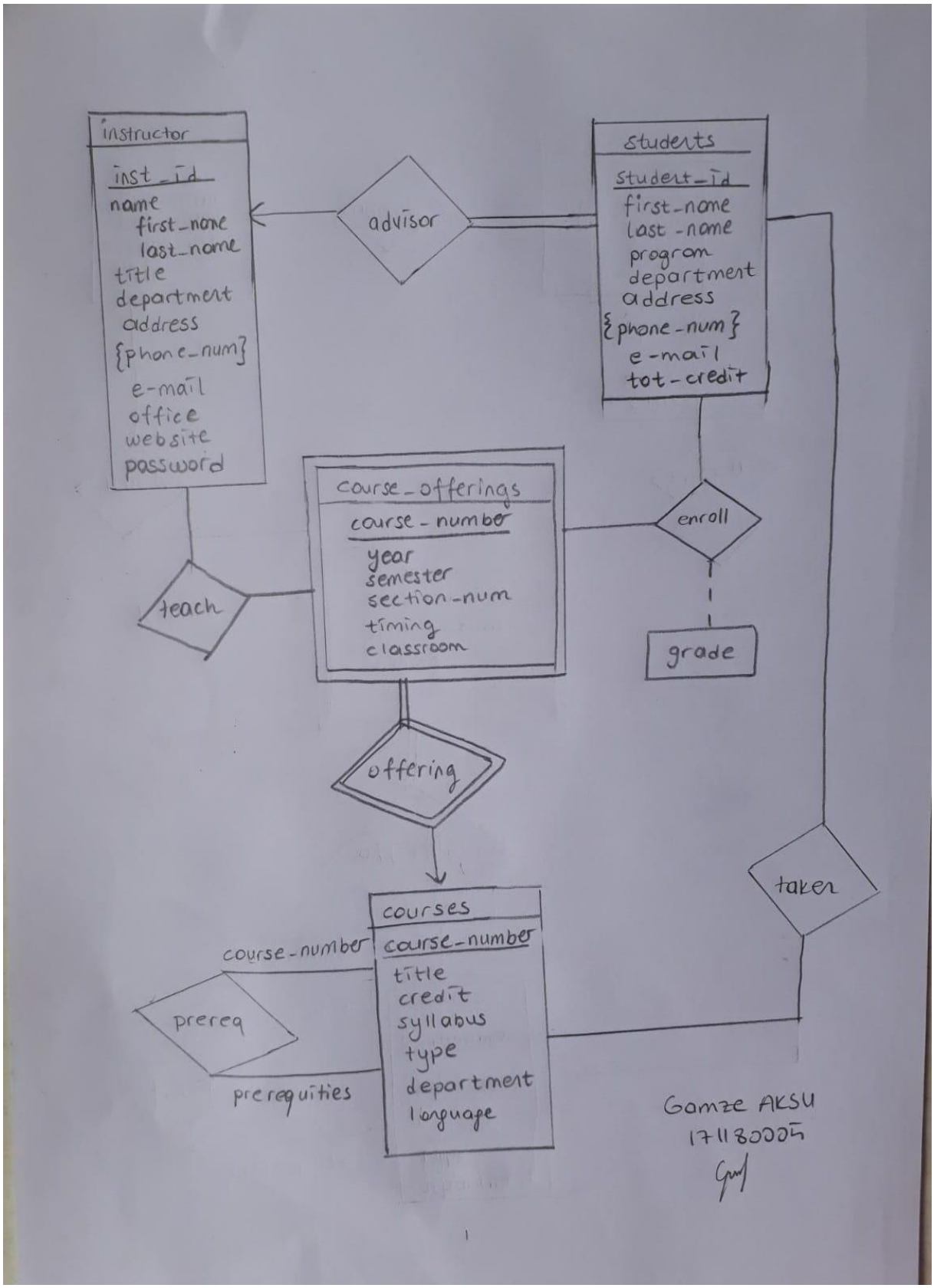
Column Name	Data Type	Allow Nulls
student_id	varchar(10)	☐
first_name	varchar(50)	☐
last_name	varchar(50)	☐
program	varchar(20)	☐
department	varchar(50)	☐
address	varchar(200)	☐
phone_number	varchar(500)	☐
mail	varchar(50)	☐
tot_credit	numeric(3, 0)	☐
		☐

Table 1: STUDENT table

Then the INSTRUCTOR table was created. The primary key of this table is inst_id. The INSTRUCTOR table was created to store instructor information.

Column Name	Data Type	Allow Nulls
inst_id	varchar(10)	☐
first_name	varchar(50)	☐
last_name	varchar(50)	☐
title	varchar(20)	☐
department	varchar(50)	☐
address	varchar(200)	☑
phone_number	varchar(500)	☑
mail	varchar(50)	☐
office	varchar(10)	☑
website	varchar(50)	☑
password	varchar(50)	☐
		☐

Table 2: INSTRUCTOR table



Shema 1: The ER diagram created for Midterm

Now a new ADVISOR table has been created for the relationship between the INSTRUCTOR table and the STUDENT table. Here, since each student can have only one advisor (1:M), the student_advisor_id created with the student_id in the STUDENT table becomes the primary key.

Column Name	Data Type	Allow Nulls
student_advisor_id	varchar(10)	☐
inst_advisor_id	varchar(10)	☐
		☐

Table 3:ADVISOR table

Here student_advisor_id is both primary key and foreign key. It comes from the primary key of the STUDENT table. Inst_advisor_id comes from inst_id, which is the primary key in the INSTRUCTOR table.

```
ALTER TABLE ADVISOR
ADD FOREIGN KEY (inst_advisor_id) REFERENCES INSTRUCTOR(inst_id);
```

```
Messages
Commands completed successfully.
```

```
ALTER TABLE ADVISOR
ADD FOREIGN KEY (student_advisor_id) REFERENCES student(student_id);
```

```
Messages
Commands completed successfully.
```

```
dbo.ADVISOR
  Columns
    student_advisor_id (PK, FK, varchar(10), not null)
    inst_advisor_id (FK, varchar(10), not null)
```

Now the COURSES table will be created. Primary key is course_number.

Column Name	Data Type	Allow Nulls
course_number	varchar(10)	☐
title	varchar(50)	☐
credit	numeric(2, 0)	☐
type	varchar(1)	☐
department	varchar(50)	☐
language	varchar(10)	☐
syllabus	text	☑
		☐

Table 4: COURSES table

The COURSE_OFFERINGS table has been created. It is also a weak entity. That's why all the attributes it contains are primary keys.

```
CREATE TABLE COURSE_OFFERINGS(
    course_number VARCHAR(10) NOT NULL,
    year_ VARCHAR(4) NOT NULL,
    semester VARCHAR(6) NOT NULL,
    section_number VARCHAR(10) NOT NULL,
    timing VARCHAR(19) NOT NULL,
    classroom VARCHAR(100) NOT NULL,
    PRIMARY KEY (course_number,year_,semester,section_number,timing,classroom)
    );
```

Messages

```
Commands completed successfully.
```

Column Name	Data Type	Allow Nulls
course_number	varchar(10)	☐
year_	varchar(4)	☐
semester	varchar(6)	☐
section_number	varchar(10)	☐
timing	varchar(19)	☐
classroom	varchar(100)	☐
		☐

Table 5: COURSE_OFFERINGS table

In this table, course_number is the foreign key for the course_number in the COURSES table.

```sql
ALTER TABLE COURSE_OFFERINGS
ADD CONSTRAINT FK_course_offerings
FOREIGN KEY (course_number) REFERENCES COURSES(course_number);
```

Messages

```
Commands completed successfully.
```

- dbo.COURSE_OFFERINGS
  - Columns
    - course_number (PK, FK, varchar(10), not null)
    - year_ (PK, varchar(4), not null)
    - semester (PK, varchar(6), not null)
    - section_number (PK, varchar(10), not null)
    - timing (PK, varchar(19), not null)
    - classroom (PK, varchar(100), not null)

The ENROL table has been created. This table exists for the relationship between the STUDENT table and the COURSE_OFFERINGS table. It shows the courses that students are enrolled in. All attributes except grade are primary keys.

```
CREATE TABLE ENROL(
    student_id VARCHAR(10) NOT NULL,
    course_number VARCHAR(10) NOT NULL,
    year_ VARCHAR(4) NOT NULL,
    semester VARCHAR(6) NOT NULL,
    section_number VARCHAR(10) NOT NULL,
    timing VARCHAR(19) NOT NULL,
    classroom VARCHAR(100) NOT NULL,
    grade VARCHAR(2) NOT NULL,
    PRIMARY KEY (student_id,course_number,year_,semester,section_number,timing,classroom)
    );
```

Messages

```
Commands completed successfully.
```

	Column Name	Data Type	Allow Nulls
🔑	student_id	varchar(10)	☐
🔑	course_number	varchar(10)	☐
🔑	year_	varchar(4)	☐
🔑	semester	varchar(6)	☐
🔑	section_number	varchar(10)	☐
🔑	timing	varchar(19)	☐
🔑	classroom	varchar(100)	☐
	grade	varchar(2)	☐
▶			☐

Table 6: ENROL table

The student_id in this table came from the student_id in the STUDENT table. The student_id in this table is the foreign key.

```
ALTER TABLE ENROL
ADD FOREIGN KEY (student_id)
REFERENCES STUDENT(student_id);
```

Messages

```
Commands completed successfully.
```

The remaining attributes (course_number, year_,semester, section_number, timing, classroom) came from the COURSE_OFFERINGS table. These attributes are also foreign keys.

```sql
ALTER TABLE ENROL
ADD FOREIGN KEY (course_number,
                 year_,
                 semester,
                 section_number,
                 timing,
                 classroom)
REFERENCES COURSE_OFFERINGS(course_number,
                            year_,
                            semester,
                            section_number,
                            timing,
                            classroom);
```

Messages

```
Commands completed successfully.
```

- dbo.ENROL
  - Columns
    - student_id (PK, FK, varchar(10), not null)
    - course_number (PK, FK, varchar(10), not null)
    - year_ (PK, FK, varchar(4), not null)
    - semester (PK, FK, varchar(6), not null)
    - section_number (PK, FK, varchar(10), not null)
    - timing (PK, FK, varchar(19), not null)
    - classroom (PK, FK, varchar(100), not null)
    - grade (varchar(2), not null)

TEACH table created. It consists of the relationship between the INSTRUCTOR table and the COURSE_OFFERINGS table. It is a table where the offered courses that the Instructor teaches are stored. The inst_id, course_number, year_, semester, section_number, timing, classroom attributes are both a primary key and a foreign key.

```sql
CREATE TABLE TEACH(
    inst_id VARCHAR(10) NOT NULL ,
    course_number VARCHAR(10) NOT NULL,
    year_ VARCHAR(4) NOT NULL,
    semester VARCHAR(6) NOT NULL,
    section_number VARCHAR(10) NOT NULL,
    timing VARCHAR(19) NOT NULL,
    classroom VARCHAR(100) NOT NULL,
    PRIMARY KEY (inst_id,course_number,year_,
    semester,section_number,timing,classroom)
    );
```

Messages

```
Commands completed successfully.
```

	Column Name	Data Type	Allow Nulls
🔑	inst_id	varchar(10)	☐
🔑	course_number	varchar(10)	☐
🔑	year_	varchar(4)	☐
🔑	semester	varchar(6)	☐
🔑	section_number	varchar(10)	☐
🔑	timing	varchar(19)	☐
🔑	classroom	varchar(100)	☐
▶			☐

Table 7: TEACH table

The course_number, year_, semester, section_number, timing, classroom attributes come from the COURSE_OFFERINGS table. That's why these attributes are foreign keys.

```
ALTER TABLE TEACH
ADD FOREIGN KEY (course_number,
                 year_,
                 semester,
                 section_number,
                 timing,
                 classroom)
REFERENCES COURSE_OFFERINGS(course_number,
                            year_,
                            semester,
                            section_number,
                            timing,
                            classroom);
```

```
Messages
Commands completed successfully.
```

The inst_id in the table comes from the inst_id in the INSTRUCTOR table. That's why inst_id becomes a foreign key.

```
ALTER TABLE TEACH
ADD FOREIGN KEY (inst_id)
REFERENCES INSTRUCTOR(inst_id);
```

```
Messages
Commands completed successfully.
```

```
dbo.TEACH
  Columns
    inst_id (PK, FK, varchar(10), not null)
    course_number (PK, FK, varchar(10), not null)
    year_ (PK, FK, varchar(4), not null)
    semester (PK, FK, varchar(6), not null)
    section_number (PK, FK, varchar(10), not null)
    timing (PK, FK, varchar(19), not null)
    classroom (PK, FK, varchar(100), not null)
```

The PREREQ table has been created. Both attributes in the PREREQ table come from the course_number attribute in the COURSES table.

```
CREATE TABLE PREREQ(
course_number VARCHAR(10) NOT NULL,
prerequisites VARCHAR(10) NOT NULL,
PRIMARY KEY(course_number));
```

0 %

Messages

```
Commands completed successfully.
```

	Column Name	Data Type	Allow Nulls
🔑	course_number	varchar(10)	☐
	prerequisites	varchar(10)	☐
▶			☐

Table 8: PREREQ table

The course_number in the PREREQ table is the primary key. It is also the foreign key from the COURSES table.

```
ALTER TABLE PREREQ ADD FOREIGN KEY
(course_number) REFERENCES
COURSES(course_number);
```

0 %

Messages

```
Commands completed successfully.
```

The prerequisites attribute in the table also comes from the course_number attribute of the COURSES table. That's why it's a foreign key.

```
ALTER TABLE PREREQ ADD FOREIGN KEY
(prerequisites) REFERENCES COURSES(course_number);
```

%

Messages

```
Commands completed successfully.
```

The TAKEN table has been created.

	Column Name	Data Type	Allow Nulls
🔑	taken_id	int	☐
	student_id	varchar(10)	☐
	course_number	varchar(10)	☐
▶			☐

Table 9: TAKEN table

The student_id in the table comes from the student_id in the STUDENT table. That's why it's a foreign key.

```
ALTER TABLE TAKEN
ADD FOREIGN KEY (student_id)
REFERENCES STUDENT(student_id);
```

```
%
Messages
Commands completed successfully.
```

The course_number in the table is a foreign key from the course_number attribute in the COURSES table.

```
ALTER TABLE TAKEN
ADD FOREIGN KEY (course_number)
REFERENCES COURSES(course_number);
```

```
%
Messages
Commands completed successfully.
```

Dummy data has been added to measure performance. Only the INSTUCTOR table will be queryed. Therefore, data must be added to the tables that are related to the INSTRUCTOR table first. Therefore, 100 dummy data were added to the DEPARTMENTS table by running the code below.

```
declare @i as int=1
while @i<101
begin

insert into DEPARTMENTS
values
(
'Bilgisyar Mühendisliği '+
cast(@i as varchar(100)),
'Mühendislik Binası '+
cast(@i as varchar(100))
)
set @i=@i+1
end
go
```

150 %

Messages | Execution plan

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

150 %

Query executed successfully.

Then, the data in this DEPARTMENT table were randomly selected and the INSTRUCTOR table in the CourseCredit database, the INSTRUCTOR table and the INSTRUCTOR_PHONES table in the CourseCreditSystem database were filled with the same dummy data.

```sql
declare @i int = 14100000
while @i<14200000  --100 000 data
begin
declare @dept_id as smallint
declare @dept_name as varchar(50)
declare @building as varchar(100)
declare @rand as int

set @rand =RAND()*100

select @dept_id=dept_id,
@dept_name=dept_name,
@building =building
from [CourseCreditSystem].[dbo].DEPARTMENTS
where dept_id=@rand
```

150 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

150 %

Query executed successfully.

```sql
insert [CourseCredit].[dbo].INSTRUCTOR
select cast(@i as varchar(10)), --inst_id
'Gamze'+cast(@i as varchar(50)),
'K'+cast(@i as varchar(50)),
'Prof'+cast(@i as varchar(20)),
@dept_name + ', '+@building,
'address '+cast(@i as varchar(200)),
'05'+cast(@i as varchar(10)),
cast(@i as varchar(10))+'_@gmail.com',
'Z-'+cast(@i as varchar(10)),
'website'+cast(@i as varchar(10))+'.com',
cast(@i as varchar(50))+'password'
```

150 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

150 %

Query executed successfully.

```sql
insert [CourseCreditSystem].[dbo].INSTRUCTOR
select cast(@i as varchar(10)), --inst_id
'Gamze'+cast(@i as varchar(50)),
'K'+cast(@i as varchar(50)),
'Prof'+cast(@i as varchar(20)),
@dept_id,
cast(@i as varchar(10))+'_@gmail.com',
'Z-'+cast(@i as varchar(10)),
'website'+cast(@i as varchar(10))+'.com',
cast(@i as varchar(50))+'password'
```

150 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

150 %

Query executed successfully.

```
insert into [CourseCreditSystem].[dbo].INSTRUCTOR_PHONES
values(cast(@i as varchar(10)), --inst_id
'05'+cast(@i as varchar(10)))

set @i= @i+1
end
go

-----------------------------
```

150 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

150 %

Query executed successfully.

```
declare @i int = 14100000
while @i<14200000  --100 000 data
begin
declare @dept_id as smallint
declare @dept_name as varchar(50)
declare @building as varchar(100)
declare @rand as int

set @rand =RAND()*100

select @dept_id=dept_id,
@dept_name=dept_name,
@building =building
from [CourseCreditSystem].[dbo].DEPARTMENTS
where dept_id=@rand

insert [CourseCredit].[dbo].INSTRUCTOR
select cast(@i as varchar(10)), --inst_id
'Gamze'+cast(@i as varchar(50)),
'K'+cast(@i as varchar(50)),
'Prof'+cast(@i as varchar(20)),
 @dept_name + ', '+@building,
'address '+cast(@i as varchar(200)),
```

```
'05'+cast(@i as varchar(10)),
cast(@i as varchar(10))+'_@gmail.com',
'Z-'+cast(@i as varchar(10)),
'website'+cast(@i as varchar(10))+'.com',
cast(@i as varchar(50))+'password'

insert [CourseCreditSystem].[dbo].INSTRUCTOR
select cast(@i as varchar(10)), --inst_id
'Gamze'+cast(@i as varchar(50)),
'K'+cast(@i as varchar(50)),
'Prof'+cast(@i as varchar(20)),
@dept_id,
cast(@i as varchar(10))+'_@gmail.com',
'Z-'+cast(@i as varchar(10)),
'website'+cast(@i as varchar(10))+'.com',
cast(@i as varchar(50))+'password'

insert into [CourseCreditSystem].[dbo].INSTRUCTOR_PHONES
values(cast(@i as varchar(10)), --inst_id
'05'+cast(@i as varchar(10)))

set @i= @i+1
end
go
```

The 5 metrics chosen to be measured are as follows [1]:

- Logical reads: Indicates the number of pages read from the data cache.
- Physical reads: Indicates the number of pages read from the disc.
- Read-ahead reads: Indicates the number of pages placed into the cache for the query.
- CPU time: Shows total CPU time.
- Elapsed time: Indicates the total time to run the query.

A query has been written to display the inst_id, first_name, last_name, department name and phone number of instructors whose department name ends with 17, that is, whose dept_id is 17 in the INSTRUCTOR table.

Cache should be cleaned to get accurate results. So the following piece of code is run.

```
DBCC FREEPROCCACHE;
DBCC DROPCLEANBUFFERS;
```

50 %

Messages

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

In order to look at the values of the performance metrics, the following piece of code is run.

```
SET STATISTICS IO ON
SET STATISTICS TIME ON
```

%

Messages

```
Commands completed successfully.
```

```
select inst_id as ID,
first_name as NAME,
last_name as SURNAME,
department as 'DEPARTMENT NAME',
phone_number as PHONE
from INSTRUCTOR
where department like '%17';
```

150 %

Results | Messages

	ID	NAME	SURNAME	DEPARTMENT NAME	PHONE
1	14100229	Gamze14100229	K14100229	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514100229
2	14100234	Gamze14100234	K14100234	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514100234
3	14100490	Gamze14100490	K14100490	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514100490
4	14100982	Gamze14100982	K14100982	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514100982
5	14101138	Gamze14101138	K14101138	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514101138
6	14101195	Gamze14101195	K14101195	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514101195
7	14101501	Gamze14101501	K14101501	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514101501
8	14101778	Gamze14101778	K14101778	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514101778
9	14101846	Gamze14101846	K14101846	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514101846
10	14101994	Gamze14101994	K14101994	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514101994
11	14102010	Gamze14102010	K14102010	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102010
12	14102125	Gamze14102125	K14102125	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102125
13	14102318	Gamze14102318	K14102318	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102318
14	14102467	Gamze14102467	K14102467	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102467
15	14102525	Gamze14102525	K14102525	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102525
16	14102592	Gamze14102592	K14102592	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102592
17	14102595	Gamze14102595	K14102595	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102595
18	14102712	Gamze14102712	K14102712	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102712
19	14102746	Gamze14102746	K14102746	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102746
20	14102753	Gamze14102753	K14102753	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102753
21	14102868	Gamze14102868	K14102868	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102868
22	14102875	Gamze14102875	K14102875	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514102875
23	14103113	Gamze14103113	K14103113	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514103113
24	14103198	Gamze14103198	K14103198	Bilgisyar Mühendisliği 17, Mühendislik Binası 17	0514103198

Query executed successfully.

```sql
select inst_id as ID,
first_name as NAME,
last_name as SURNAME,
department as 'DEPARTMENT NAME',
phone_number as PHONE
from INSTRUCTOR
where department like '%17';
```

150 %

Results | Messages

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 2 ms.

(1008 rows affected)
Table 'INSTRUCTOR'. Scan count 1, logical reads 2721, physical reads 3, read-ahead reads 2557, lob logical reads 0, lob physical

 SQL Server Execution Times:
   CPU time = 500 ms,  elapsed time = 520 ms.
```

150 %

Query executed successfully. | DESKTOP-QK3DF15 (14.0 RTM) | sa (51) | CourseCredit | 00:00:00 | 1.008 rows

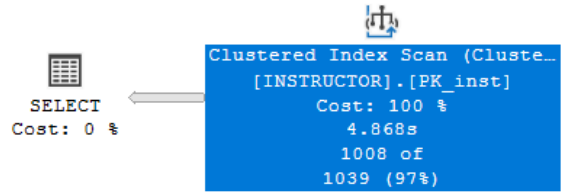
```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 2 ms.

(1008 rows affected)
Table 'INSTRUCTOR'. Scan count 1, logical reads 2721, physical reads 3, read-ahead reads 2557, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

 SQL Server Execution Times:
   CPU time = 500 ms,  elapsed time = 520 ms.
```

```sql
select inst_id as ID,
first_name as NAME,
last_name as SURNAME,
department as 'DEPARTMENT NAME',
phone_number as PHONE
from INSTRUCTOR
where department like '%17';
```

124 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
select inst_id as ID, first_name as NAME, last_name



Query executed successfully.

At the end of this query:

- Logical reads: 2721
- Physical reads: 3
- Read-ahead reads: 2557
- CPU time: 500ms
- Elapsed time: 520ms

B. CourseCreditSystem database, which is the normalized version of the database created in option A, is created in this section.

First Normal Form (1NF): Multivalued attributes, composite attributes, or combinations of them are not allowed. In this way, only tables containing atomic values comply with 1NF. [2]

Second Normal Form (2NF): For a table to fit 2NF, it must be 1NF. All columns that do not have any key attributes should be fully dependent on the primary key and tables should be split accordingly. [2]

Third Normal Form (3NF): In order for a table to fit 3NF, it must be 2NF and not have transitive partial dependency. [2]

Removed the phone_number and address attributes from the INSTRUCTOR table. Created INSTRUCTOR_PHONES table instead of the phone_number and INST_ADDRESSES table instead of the address. The DEPARTMENTS table has been created.

Column Name	Data Type	Allow Nulls
inst_id	varchar(10)	☐
first_name	varchar(50)	☐
last_name	varchar(50)	☐
title	varchar(20)	☐
department	smallint	☐
mail	varchar(50)	☑
office	varchar(10)	☑
website	varchar(50)	☑
password	varchar(50)	☐
		☐

Table 10: New INSTRUCTOR table

The INSTRUCTOR_PHONES table has been created.

Column Name	Data Type	Allow Nulls
inst_phones_id	int	☐
inst_id	varchar(10)	☐
phone	varchar(10)	☐
		☐

Table 11: INSTRUCTOR_PHONES table

The inst_id in the table is a foreign key from the inst_id in the INSTRUCTOR table.

```
ALTER TABLE INSTRUCTOR_PHONES
ADD FOREIGN KEY (inst_id)
REFERENCES INSTRUCTOR(inst_id);
```

Messages

```
Commands completed successfully.
```

In order to create the INST_ADDRESSES table, first the tables showing the districts and neighbourhoods in Antalya must be created.

The DISTRICT table has been created. ID is primary key.

```
CREATE TABLE DISTRICT (
 ID tinyint NOT NULL PRIMARY KEY IDENTITY(1,1),
 district_name VARCHAR(50) NOT NULL
);
```

Messages

```
Commands completed successfully.
```

The NEIGHBOURHOODS table has been created. ID is primary key.

```
CREATE TABLE NEIGBOURHOODS
(
    ID  SMALLINT IDENTITY (1,1)
        PRIMARY KEY NOT NULL,
    district_id     TINYINT     NOT NULL,
    neigbourhood_name VARCHAR(90) NOT NULL
);
```

```
4 %
Messages
Commands completed successfully.
```

The district_id is the foreign key. It comes from the ID attribute in the NEIGBOURHOODS table.

```
ALTER TABLE NEIGBOURHOODS
ADD FOREIGN KEY (district_id)
REFERENCES DISTRICT(ID);
```

```
4 %
Messages
Commands completed successfully.
```

The INST_ADDRESSES table has been created. The address_id attribute is an auto-increment primary key. The inst_id attribute is a foreign key from the INSTRUCTOR table. The district_id attribute is a foreign key from the DISTRICT table. The neigbourhood_id attribute is a foreign key from the NEIGBOURHOODS table.

```
create table INST_ADDRESSES(
    address_id int identity(1,1) primary key,
    inst_id varchar(10),
    district_id tinyint,
    neigbourhood_id smallint
);
```

```
%
Messages
Commands completed successfully.
```

```sql
alter table INST_ADDRESSES
add foreign key (district_id)
references DISTRICT(ID);
```

Messages

Commands completed successfully.

```sql
alter table INST_ADDRESSES
add foreign key (inst_id)
references INSTRUCTOR(inst_id);
```

Messages

Commands completed successfully.

```sql
alter table INST_ADDRESSES
add foreign key (neigbourhood_id)
references NEIGBOURHOODS(ID);
```

Messages

Commands completed successfully.

The STUDENT table has been updated. The STU_ADDRESSES table was created instead of the address attribute in this table. Likewise, STUDENT_PHONES table was created instead of phone_number attribute.

Column Name	Data Type	Allow Nulls
student_id	varchar(10)	☐
first_name	varchar(50)	☐
last_name	varchar(50)	☐
program	varchar(20)	☐
department	smallint	☐
mail	varchar(50)	☐
tot_credit	numeric(3, 0)	☐
		☐

Table 12: New STUDENT table

The STUDENT_PHONES table has been created.

Column Name	Data Type	Allow Nulls
student_phones_id	int	☐
student_id	varchar(10)	☐
phone	varchar(10)	☐
		☐

Table 13: STUDENT_PHONES table

The student_id attribute in the STUDENT_PHONES table comes from the student_id in the STUDENT table. That's why it's a foreign key.

```
ALTER TABLE STUDENT_PHONES
ADD FOREIGN KEY (student_id)
REFERENCES STUDENT(student_id);
```

0 %

Messages

```
Commands completed successfully.
```

STU_ADDRESSES table has been created. The address_id attribute is an auto-increment primary key. The student attribute is a foreign key from the STUDENT table. The district_id attribute is a foreign key from the DISTRICT table. The neigbourhood_id attribute is a foreign key from the NEIGBOURHOODS table.

```
create table STU_ADDRESSES(
    address_id int identity(1,1) primary key,
    inst_id varchar(10),
    district_id tinyint,
    neigbourhood_id smallint
);
```

Messages

Commands completed successfully.

```
alter table STU_ADDRESSES
add foreign key (inst_id)
references INSTRUCTOR(inst_id);
```

Messages

Commands completed successfully.

```
alter table STU_ADDRESSES
add foreign key (district_id)
references DISTRICT(ID);
```

Messages

Commands completed successfully.

```
alter table STU_ADDRESSES
add foreign key (neigbourhood_id)
references NEIGBOURHOODS(ID);
```

Messages

Commands completed successfully.

The DEPARTMENTS table has been created. Previously, department attribute stored all department information in the INSTRUCTOR table, the STUDENT table, and the COURSES table. Now only the dept_id in the DEPARTMENT table is stored.

Column Name	Data Type	Allow Nulls
dept_id	smallint	☐
dept_name	varchar(100)	☐
building	varchar(100)	☐
		☐

Table 14: DEPARTMENTS table

```
ALTER TABLE INSTRUCTOR
ADD FOREIGN KEY (department)
REFERENCES DEPARTMENTS(dept_id);
```

```
Messages
Commands completed successfully.
```

```
ALTER TABLE STUDENT
ADD FOREIGN KEY (department)
REFERENCES DEPARTMENTS(dept_id);
```

```
Messages
Commands completed successfully.
```

```
ALTER TABLE COURSES
ADD FOREIGN KEY (department)
REFERENCES DEPARTMENTS(dept_id);
```

```
Messages
Commands completed successfully.
```

COURSE_OFFERINGS table has been updated.

	Column Name	Data Type	Allow Nulls
🔑	course_number	varchar(10)	☐
🔑	year_	varchar(4)	☐
🔑	semester	varchar(6)	☐
🔑	section_number	varchar(10)	☐
	classroom_address_id	int	☐
	timing_id	int	☐
▶			☐

Table 15: New COURSE_OFFERINGS table

```sql
CREATE TABLE COURSE_OFFERINGS(
    course_number VARCHAR(10) NOT NULL,
    year_ VARCHAR(4) NOT NULL,
    semester VARCHAR(6) NOT NULL,
    section_number VARCHAR(10) NOT NULL,
    PRIMARY KEY (course_number,year_,semester,section_number)
    );
```

%

Messages

```
Commands completed successfully.
```

The CLASSROOMS table has been created.

	Column Name	Data Type	Allow Nulls
🔑	classroom_id	int	☐
	building	varchar(50)	☐
	classroom	varchar(50)	☐
	capacity	varchar(3)	☐
▶			☐

Table 16: The CLASSROOMS table

The TIMING table has been created.

Column Name	Data Type	Allow Nulls
timing_id	int	☐
day	varchar(10)	☐
start_time	time(2)	☐
end_time	time(2)	☐
		☐

Table 17: TIMING table

To create a relationship between the COURSE_OFFERING table and the CLASSROOMS table, the classroom_address_id becomes a foreign key. Likewise, timing_id becomes a foreign key for a relationship between the COURSE_OFFERING table and the TIMING table.

```
ALTER TABLE COURSE_OFFERINGS
ADD FOREIGN KEY (classroom_address_id)
REFERENCES CLASSROOMS(classroom_id);
```

```
Messages
Commands completed successfully.
```

```
ALTER TABLE COURSE_OFFERINGS
ADD FOREIGN KEY (timing_id)
REFERENCES TIMING(timing_id);
```

```
Messages
Commands completed successfully.
```

The ENROL table has been updated. The student_id, course_number, year_, semester and section_number attributes are both a primary key and a foreign key.

```
CREATE TABLE ENROL(
    student_id VARCHAR(10) NOT NULL,
    course_number VARCHAR(10) NOT NULL,
    year_ VARCHAR(4) NOT NULL,
    semester VARCHAR(6) NOT NULL,
    section_number VARCHAR(10) NOT NULL,
    grade VARCHAR(2) NOT NULL,
    PRIMARY KEY (student_id,course_number,year_,semester,section_number)
    );
```

Messages

```
Commands completed successfully.
```

Column Name	Data Type	Allow Nulls
student_id	varchar(10)	☐
course_number	varchar(10)	☐
year_	varchar(4)	☐
semester	varchar(6)	☐
section_number	varchar(10)	☐
grade	varchar(2)	☐
		☐

Table 18: New ENROL table

The course_number, year_, semester and section_number attributes come from the COURSE_OFFERINGS table, while the student_id comes from the STUDENT table.

```sql
ALTER TABLE ENROL
ADD FOREIGN KEY (course_number,
                 year_,
                 semester,
                 section_number)
REFERENCES COURSE_OFFERINGS(course_number,
                            year_,
                            semester,
                            section_number);
```

0 %

Messages

```
Commands completed successfully.
```

```sql
ALTER TABLE ENROL
ADD FOREIGN KEY (student_id)
REFERENCES STUDENT(student_id);
```

%

Messages

```
Commands completed successfully.
```

- dbo.ENROL
  - Columns
    - student_id (PK, FK, varchar(10), not null)
    - course_number (PK, FK, varchar(10), not null)
    - year_ (PK, FK, varchar(4), not null)
    - semester (PK, FK, varchar(6), not null)
    - section_number (PK, FK, varchar(10), not null)
    - grade (varchar(2), not null)

The TEACH table has been updated. The inst_id, course_number, year_, semester and section_number attributes are both a primary key and a foreign key.

```
CREATE TABLE TEACH(
    inst_id VARCHAR(10) NOT NULL ,
    course_number VARCHAR(10) NOT NULL,
    year_ VARCHAR(4) NOT NULL,
    semester VARCHAR(6) NOT NULL,
    section_number VARCHAR(10) NOT NULL
    PRIMARY KEY (inst_id,course_number,year_,
    semester,section_number)
    );
```

Messages

```
Commands completed successfully.
```

Column Name	Data Type	Allow Nulls
inst_id	varchar(10)	☐
course_number	varchar(10)	☐
year_	varchar(4)	☐
semester	varchar(6)	☐
section_number	varchar(10)	☐
		☐

Table 19: New TEACH table

The course_number, year_, semester and section_number attributes come from the COURSE_OFFERINGS table, while the inst_id comes from the INSTRUCTOR table.

```sql
ALTER TABLE TEACH
ADD FOREIGN KEY (course_number,
                 year_,
                 semester,
                 section_number)
REFERENCES COURSE_OFFERINGS(course_number,
                            year_,
                            semester,
                            section_number);
```

%

Messages

```
Commands completed successfully.
```

```sql
ALTER TABLE TEACH
ADD FOREIGN KEY (inst_id)
REFERENCES INSTRUCTOR(inst_id);
```

%

Messages

```
Commands completed successfully.
```

- dbo.TEACH
  - Columns
    - inst_id (PK, FK, varchar(10), not null)
    - course_number (PK, FK, varchar(10), not null)
    - year_ (PK, FK, varchar(4), not null)
    - semester (PK, FK, varchar(6), not null)
    - section_number (PK, FK, varchar(10), not null)

Cache should be cleaned to get accurate results. So the following piece of code is run.

```
DBCC FREEPROCCACHE;
DBCC DROPCLEANBUFFERS;
```

50 %

Messages

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

In order to look at the values of the performance metrics, the following piece of code is run.

```
SET STATISTICS IO ON
SET STATISTICS TIME ON
```

%

Messages

```
Commands completed successfully.
```

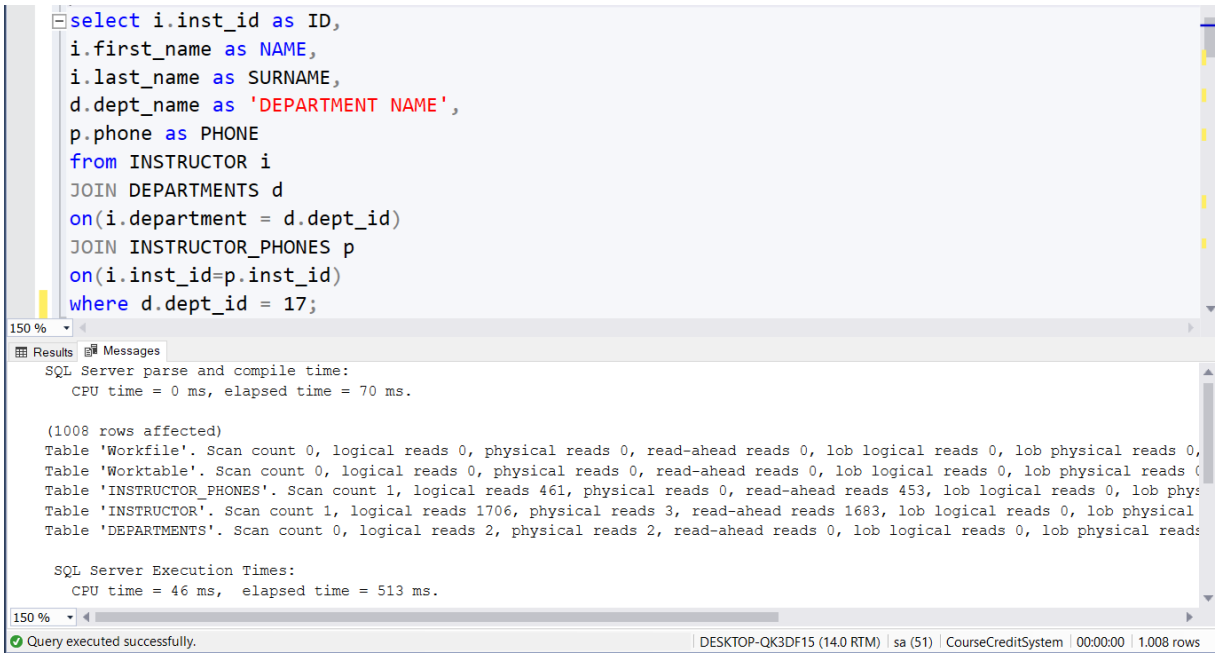
```
select i.inst_id as ID,
i.first_name as NAME,
i.last_name as SURNAME,
d.dept_name as 'DEPARTMENT NAME',
p.phone as PHONE
from INSTRUCTOR i
JOIN DEPARTMENTS d
on(i.department = d.dept_id)
JOIN INSTRUCTOR_PHONES p
on(i.inst_id=p.inst_id)
where d.dept_id = 17;
```

150 %

Results | Messages

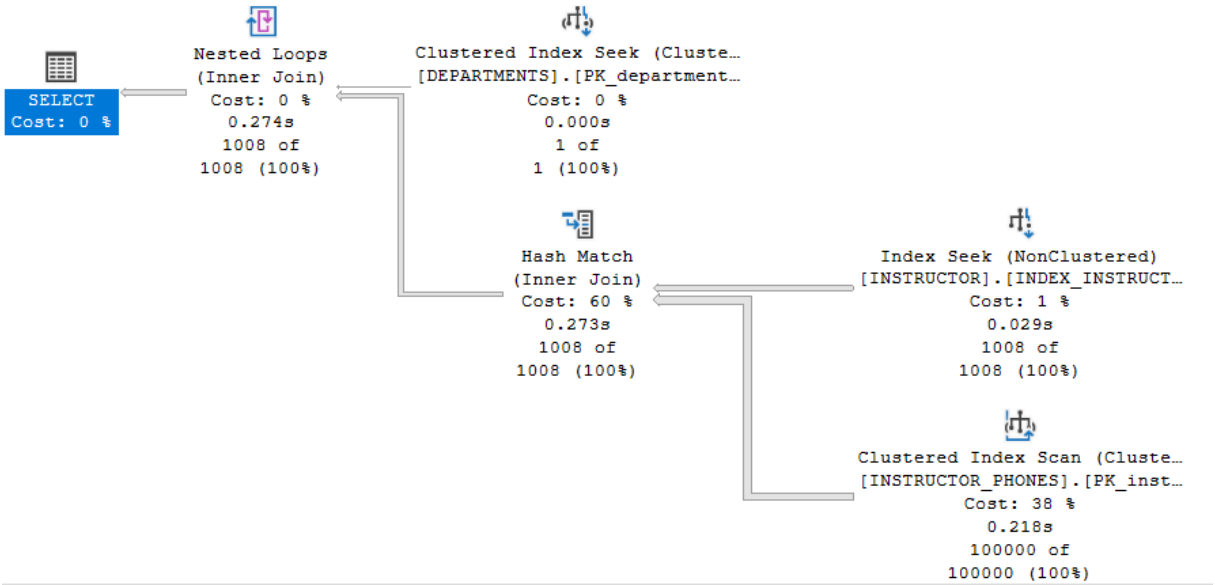
	ID	NAME	SURNAME	DEPARTMENT NAME	PHONE
1	14100229	Gamze14100229	K14100229	Bilgisyar Mühendisliği 17	0514100229
2	14100234	Gamze14100234	K14100234	Bilgisyar Mühendisliği 17	0514100234
3	14100490	Gamze14100490	K14100490	Bilgisyar Mühendisliği 17	0514100490
4	14100982	Gamze14100982	K14100982	Bilgisyar Mühendisliği 17	0514100982
5	14101138	Gamze14101138	K14101138	Bilgisyar Mühendisliği 17	0514101138
6	14101195	Gamze14101195	K14101195	Bilgisyar Mühendisliği 17	0514101195
7	14101501	Gamze14101501	K14101501	Bilgisyar Mühendisliği 17	0514101501
8	14101778	Gamze14101778	K14101778	Bilgisyar Mühendisliği 17	0514101778
9	14101846	Gamze14101846	K14101846	Bilgisyar Mühendisliği 17	0514101846
10	14101994	Gamze14101994	K14101994	Bilgisyar Mühendisliği 17	0514101994
11	14102010	Gamze14102010	K14102010	Bilgisyar Mühendisliği 17	0514102010
12	14102125	Gamze14102125	K14102125	Bilgisyar Mühendisliği 17	0514102125
13	14102318	Gamze14102318	K14102318	Bilgisyar Mühendisliği 17	0514102318
14	14102467	Gamze14102467	K14102467	Bilgisyar Mühendisliği 17	0514102467
15	14102525	Gamze14102525	K14102525	Bilgisyar Mühendisliği 17	0514102525
16	14102592	Gamze14102592	K14102592	Bilgisyar Mühendisliği 17	0514102592
17	14102595	Gamze14102595	K14102595	Bilgisyar Mühendisliği 17	0514102595

Query executed successfully.




```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 70 ms.

(1008 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0,
lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0,
lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'INSTRUCTOR_PHONES'. Scan count 1, logical reads 461, physical reads 0, read-ahead
reads 453, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'INSTRUCTOR'. Scan count 1, logical reads 1706, physical reads 3, read-ahead reads
1683, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'DEPARTMENTS'. Scan count 0, logical reads 2, physical reads 2, read-ahead reads
0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

 SQL Server Execution Times:
   CPU time = 46 ms,  elapsed time = 513 ms.
```

```
Query 1: Query cost (relative to the batch): 100%
select i.inst_id as ID, i.first_name as NAME, i.last_name as SURNAME, d.dept_name as '
Missing Index (Impact 97.7127): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sys
```



Query executed successfully.

At the end of this query:

	INSTRUCTOR_PHONES	INSTRUCTOR	DEPARTMENTS
Logical reads:	**461**	**1706**	**2**
Physical reads:	**0**	**3**	**2**
Read-ahead reads:	**453**	**1683**	**0**
CPU time:	**46 ms**		
Elapsed time:	**513 ms**		

C.

Clustered Index: Clustered index structure physically sorts the data. It provides fast access to data. Only one clustered index can be defined as it is a physical order. This is called a primary key. All tables created within the scope of the final assignment have a primary key. [3]

Non-Clustered Index: Data is ordered logically, not physically. The location of the data is kept. No direct access to data is provided. Therefore, more than one non-clustered index can be recognized for a table. [3]

```
SET STATISTICS IO OFF
SET STATISTICS TIME OFF
```

```
Messages
 SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

  SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 0 ms.
```

Since each table has a primary key, the clustered index already exists. A non-clustered index is defined for the department attribute in the INSTRUCTOR table.

```
CREATE NONCLUSTERED INDEX INDEX_DEPARTMENT
ON INSTRUCTOR(department)
INCLUDE (first_name,last_name,
title,mail,office,website,password);
```

```
Messages
  Commands completed successfully.
```

A non-clustered index is defined for the inst_id attribute in the INSTRUCTOR_PHONES table.

```
CREATE NONCLUSTERED INDEX INDEX_PHONE
ON INSTRUCTOR_PHONES(inst_id)
INCLUDE (phone);
```

```
Messages
  Commands completed successfully.
```

Cache cleaning:

```
DBCC FREEPROCCACHE;
DBCC DROPCLEANBUFFERS;
```

50 %

Messages

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

```
SET STATISTICS IO ON
SET STATISTICS TIME ON
```

%

Messages

```
Commands completed successfully.
```

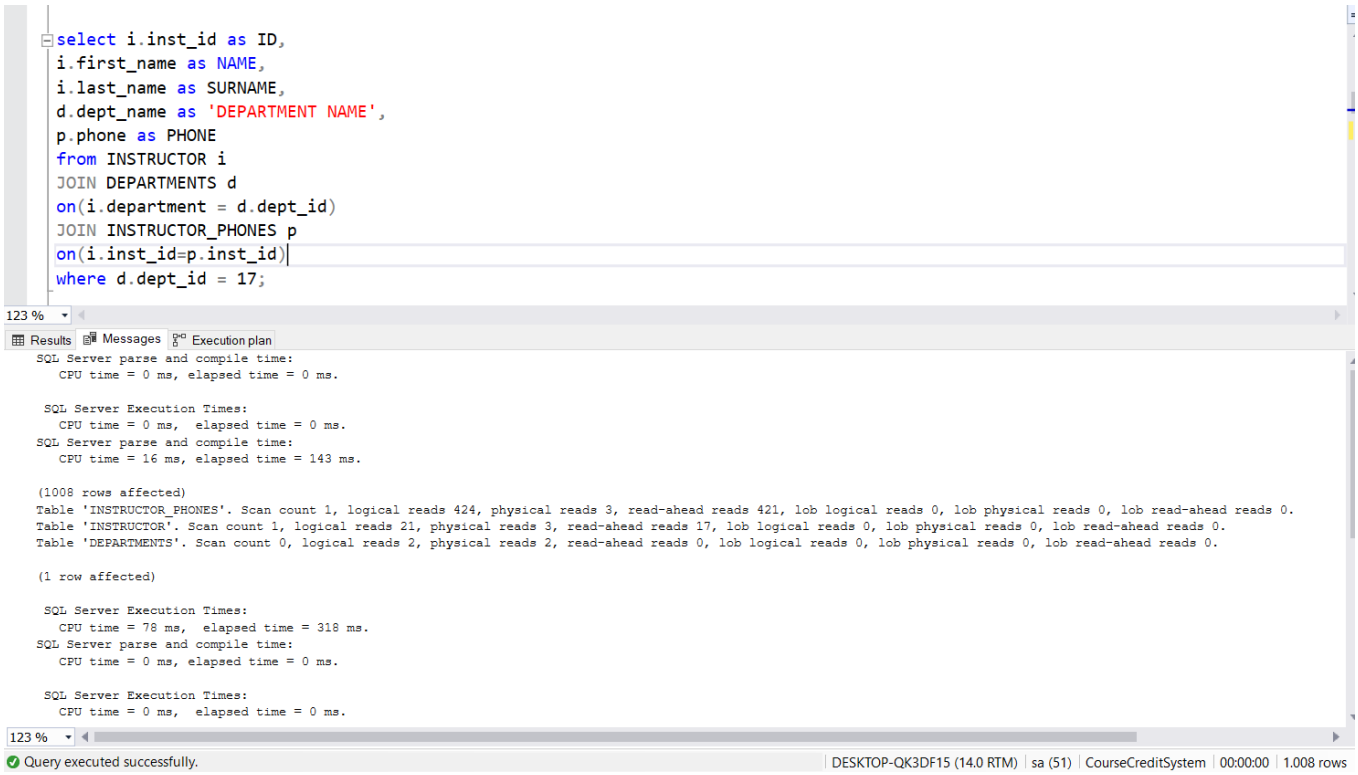
```
select i.inst_id as ID,
i.first_name as NAME,
i.last_name as SURNAME,
d.dept_name as 'DEPARTMENT NAME',
p.phone as PHONE
from INSTRUCTOR i
JOIN DEPARTMENTS d
on(i.department = d.dept_id)
JOIN INSTRUCTOR_PHONES p
on(i.inst_id=p.inst_id)
where d.dept_id = 17;
```

150 %

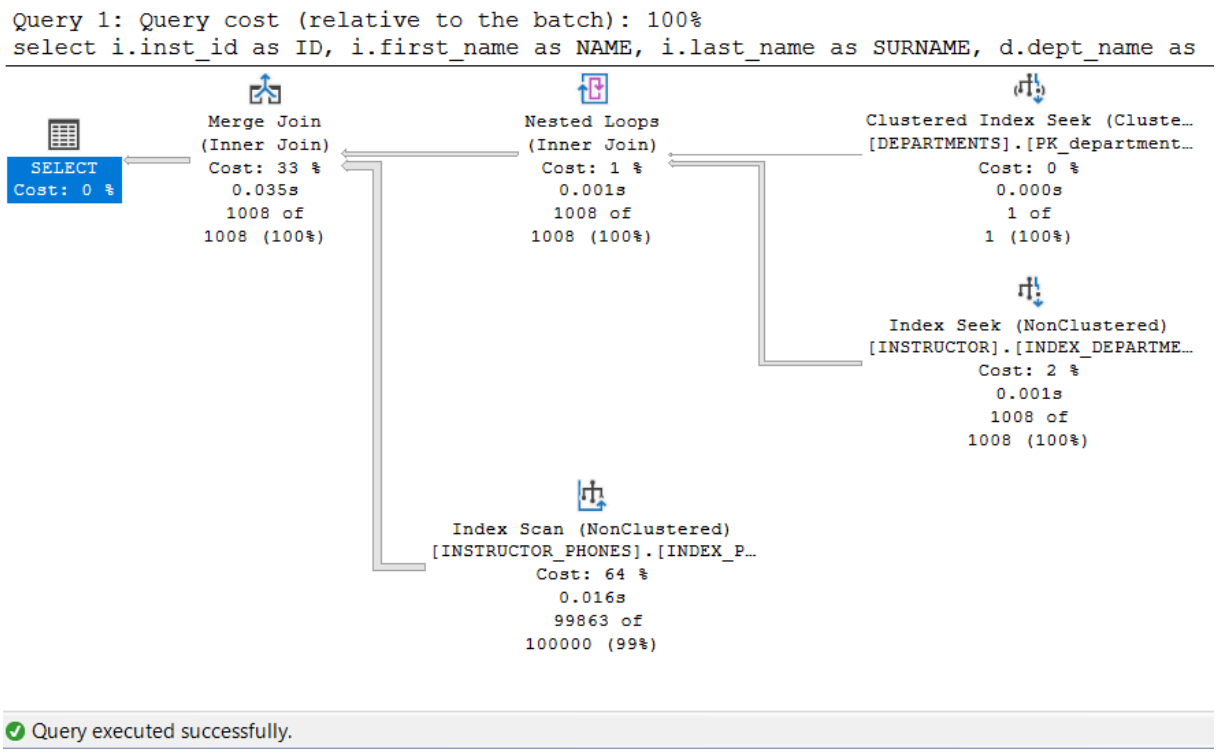
Results | Messages

	ID	NAME	SURNAME	DEPARTMENT NAME	PHONE
1	14100229	Gamze14100229	K14100229	Bilgisyar Mühendisliği 17	0514100229
2	14100234	Gamze14100234	K14100234	Bilgisyar Mühendisliği 17	0514100234
3	14100490	Gamze14100490	K14100490	Bilgisyar Mühendisliği 17	0514100490
4	14100982	Gamze14100982	K14100982	Bilgisyar Mühendisliği 17	0514100982
5	14101138	Gamze14101138	K14101138	Bilgisyar Mühendisliği 17	0514101138
6	14101195	Gamze14101195	K14101195	Bilgisyar Mühendisliği 17	0514101195
7	14101501	Gamze14101501	K14101501	Bilgisyar Mühendisliği 17	0514101501
8	14101778	Gamze14101778	K14101778	Bilgisyar Mühendisliği 17	0514101778
9	14101846	Gamze14101846	K14101846	Bilgisyar Mühendisliği 17	0514101846
10	14101994	Gamze14101994	K14101994	Bilgisyar Mühendisliği 17	0514101994
11	14102010	Gamze14102010	K14102010	Bilgisyar Mühendisliği 17	0514102010
12	14102125	Gamze14102125	K14102125	Bilgisyar Mühendisliği 17	0514102125
13	14102318	Gamze14102318	K14102318	Bilgisyar Mühendisliği 17	0514102318
14	14102467	Gamze14102467	K14102467	Bilgisyar Mühendisliği 17	0514102467
15	14102525	Gamze14102525	K14102525	Bilgisyar Mühendisliği 17	0514102525
16	14102592	Gamze14102592	K14102592	Bilgisyar Mühendisliği 17	0514102592
17	14102595	Gamze14102595	K14102595	Bilgisyar Mühendisliği 17	0514102595
18	14102712	Gamze14102712	K14102712	Bilgisyar Mühendisliği 17	0514102712
19	14102746	Gamze14102746	K14102746	Bilgisyar Mühendisliği 17	0514102746

Query executed successfully.




```
SQL Server parse and compile time:
   CPU time = 16 ms, elapsed time = 143 ms.

(1008 rows affected)
Table 'INSTRUCTOR_PHONES'. Scan count 1, logical reads 424, physical reads 3, read-
ahead reads 421, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'INSTRUCTOR'. Scan count 1, logical reads 21, physical reads 3, read-ahead reads
17, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'DEPARTMENTS'. Scan count 0, logical reads 2, physical reads 2, read-ahead reads
0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)

 SQL Server Execution Times:
   CPU time = 78 ms,  elapsed time = 318 ms.
```



According to these values, a significant decrease is achieved in logical reads from 1706 to 21 and read-ahead reads from 1683 to 17 in the INSTRUCTOR table.

At the end of this query:

	INSTRUCTOR_PHONES	INSTRUCTOR	DEPARTMENTS
Logical reads:	**424**	**21**	**2**
Physical reads:	**3**	**3**	**2**
Read-ahead reads:	**421**	**17**	**0**
CPU time:	**78 ms**		
Elapsed time:	**318 ms**		

D.

Query optimization is applied to increase performance and to access data quickly [4]. Clustered and non-clustered indexes are also an optimization, as well as making the code more optimized. For this, creating temporary tables for tables using join can be used. Performance metrics for temporary tables decrease when temporary tables are applied.

```
SET STATISTICS IO OFF
SET STATISTICS TIME OFF
```

```
%
Messages
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
```

Instead of DEPARTMENTS table #temp_department and INSTRUCTOR_PHONES table #temp_inst_phone table is created and these tables are filled with necessary data.

```
create table #temp_inst_phone
(inst_id varchar(10),
phone varchar(10));

insert into #temp_inst_phone
select inst_id,phone
from INSTRUCTOR_PHONES;

create table #temp_department
(dept_id smallint,
dept_name varchar(100));

insert into #temp_department
select dept_id,dept_name
from DEPARTMENTS;
```

```
150 %
Messages

    (100000 rows affected)

    (100 rows affected)
150 %
Query executed successfully.
```

```sql
SET STATISTICS IO ON
SET STATISTICS TIME ON
```

%

Messages

```
Commands completed successfully.
```

```sql
select i.inst_id as ID,
i.first_name as NAME,
i.last_name as SURNAME,
d.dept_name as 'DEPARTMENT NAME',
p.phone as PHONE
from INSTRUCTOR i
JOIN #temp_department d
on(i.department = d.dept_id)
JOIN #temp_inst_phone p
on(i.inst_id=p.inst_id)
where d.dept_id = 17;
```

150 %

Results | Messages

	ID	NAME	SURNAME	DEPARTMENT NAME	PHONE
1	14100229	Gamze14100229	K14100229	Bilgisyar Mühendisliği 17	0514100229
2	14100234	Gamze14100234	K14100234	Bilgisyar Mühendisliği 17	0514100234
3	14100490	Gamze14100490	K14100490	Bilgisyar Mühendisliği 17	0514100490
4	14100982	Gamze14100982	K14100982	Bilgisyar Mühendisliği 17	0514100982
5	14101138	Gamze14101138	K14101138	Bilgisyar Mühendisliği 17	0514101138
6	14101195	Gamze14101195	K14101195	Bilgisyar Mühendisliği 17	0514101195
7	14101501	Gamze14101501	K14101501	Bilgisyar Mühendisliği 17	0514101501
8	14101778	Gamze14101778	K14101778	Bilgisyar Mühendisliği 17	0514101778
9	14101846	Gamze14101846	K14101846	Bilgisyar Mühendisliği 17	0514101846
10	14101994	Gamze14101994	K14101994	Bilgisyar Mühendisliği 17	0514101994
11	14102010	Gamze14102010	K14102010	Bilgisyar Mühendisliği 17	0514102010
12	14102125	Gamze14102125	K14102125	Bilgisyar Mühendisliği 17	0514102125
13	14102318	Gamze14102318	K14102318	Bilgisyar Mühendisliği 17	0514102318
14	14102467	Gamze14102467	K14102467	Bilgisyar Mühendisliği 17	0514102467
15	14102525	Gamze14102525	K14102525	Bilgisyar Mühendisliği 17	0514102525
16	14102592	Gamze14102592	K14102592	Bilgisyar Mühendisliği 17	0514102592
17	14102595	Gamze14102595	K14102595	Bilgisyar Mühendisliği 17	0514102595
18	14102712	Gamze14102712	K14102712	Bilgisyar Mühendisliği 17	0514102712
19	14102746	Gamze14102746	K14102746	Bilgisyar Mühendisliği 17	0514102746
20	14102753	Gamze14102753	K14102753	Bilgisyar Mühendisliği 17	0514102753

Query executed successfully.

```sql
select i.inst_id as ID,
i.first_name as NAME,
i.last_name as SURNAME,
d.dept_name as 'DEPARTMENT NAME',
p.phone as PHONE
from INSTRUCTOR i
JOIN #temp_department d
on(i.department = d.dept_id)
JOIN #temp_inst_phone p
on(i.inst_id=p.inst_id)
where d.dept_id = 17;
```

150 %

Results | Messages

```
SQL Server parse and compile time:
   CPU time = 125 ms, elapsed time = 320 ms.

(1008 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0,
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads (
Table '#temp_inst_phone_____________________________________________________________________________________________000000
Table 'INSTRUCTOR'. Scan count 1, logical reads 21, physical reads 3, read-ahead reads 17, lob logical reads 0, lob physical reac
Table '#temp_department_____________________________________________________________________________________________000000

 SQL Server Execution Times:
   CPU time = 31 ms,  elapsed time = 127 ms.
```

150 %

Query executed successfully. | DESKTOP-QK3DF15 (14.0 RTM) | sa (51) | CourseCreditSystem | 00:00:00 | 1.008 rows

```sql
select i.inst_id as ID,
i.first_name as NAME,
i.last_name as SURNAME,
d.dept_name as 'DEPARTMENT NAME',
p.phone as PHONE
from INSTRUCTOR i
JOIN #temp_department d
on(i.department = d.dept_id)
JOIN #temp_inst_phone p
on(i.inst_id=p.inst_id)
where d.dept_id = 17;
```

150 %

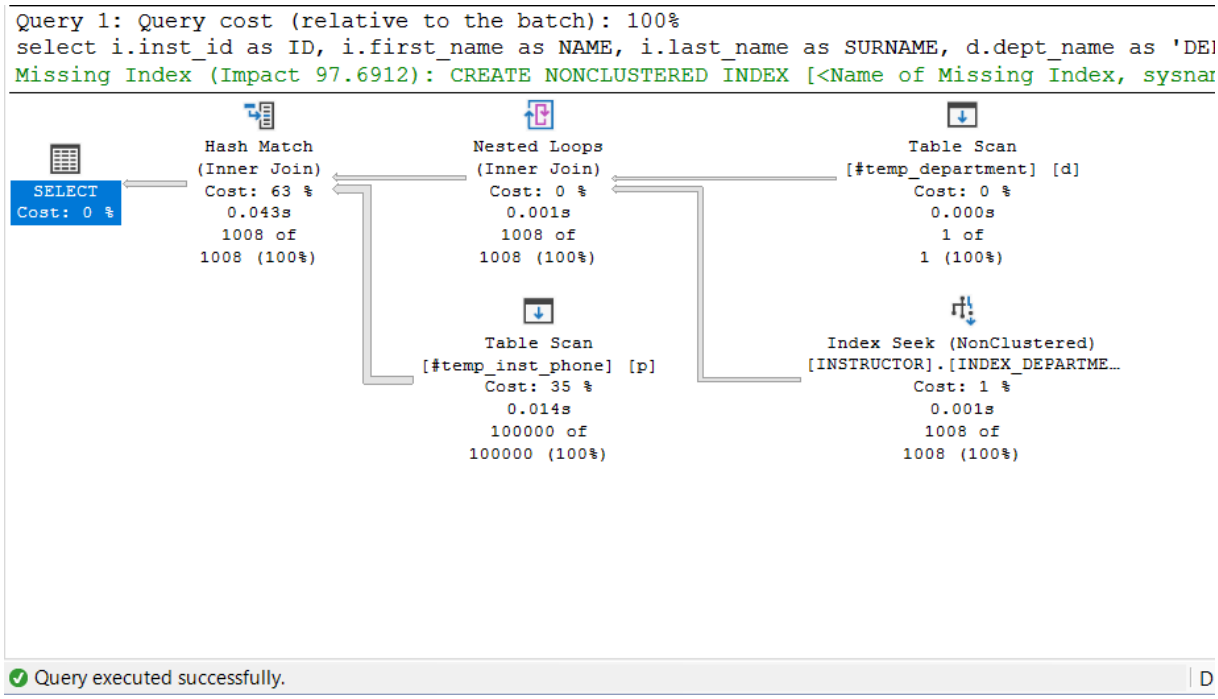
Results | Messages

```
; 0, lob read-ahead reads 0.
is 0, lob read-ahead reads 0.
)00000002F'. Scan count 1, logical reads 409, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lol
:eads 0, lob read-ahead reads 0.
)000000030'. Scan count 1, logical reads 1, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob :
```

150 %

Query executed successfully. | DESKTOP-QK3DF15 (14.0 RTM) | sa (51) | CourseCreditSystem | 00:00:00 | 1.008 rows

```
SQL Server parse and compile time:
   CPU time = 125 ms, elapsed time = 320 ms.

(1008 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0,
lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0,
lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table
'#temp_inst_phone_____________________________________________________________________
_______________________________00000000002F'. Scan count 1, logical reads 409, physical
reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead
reads 0.
Table 'INSTRUCTOR'. Scan count 1, logical reads 21, physical reads 3, read-ahead reads
17, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table
'#temp_department_____________________________________________________________________
_______________________________000000000030'. Scan count 1, logical reads 1, physical
reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead
reads 0.

 SQL Server Execution Times:
   CPU time = 31 ms,  elapsed time = 127 ms.
```



At the end of this query:

	TEMP_PHONE	INSTRUCTOR	TEMP_DEPARTMENT
Logical reads:	**409**	**21**	**1**
Physical reads:	**0**	**3**	**0**
Read-ahead reads:	**0**	**17**	**0**
CPU time:	**31 ms**		
Elapsed time:	**127 ms**		

Conclusion

		Logical reads	Physical reads	Read-ahead reads	CPU time	Elapsed time
A	INSTRUCTOR	**2721**	**3**	**2557**	**500ms**	**520ms**
B	INSTRUCTOR_PHONES	**461**	**0**	**453**	**46ms**	**513ms**
	INSTRUCTOR	**1706**	**3**	**1683**		
	DEPARTMENTS	**2**	**2**	**0**		
C	INSTRUCTOR_PHONES	**424**	**3**	**421**	**78ms**	**318ms**
	INSTRUCTOR	**21**	**3**	**17**		
	DEPARTMENTS	**2**	**2**	**0**		
D	TEMP_PHONE	**409**	**0**	**0**	**31ms**	**127ms**
	INSTRUCTOR	**21**	**3**	**17**		
	TEMP_DEPARTMENT	**1**	**0**	**0**		

- When normalization is applied to the database created in option A, a decrease is observed in all metrics except physical reads.
- There are decreases in all metrics except physical reads for the INSTRUCTOR table when indexing is applied.
- When Query Optimization is applied, the data for the INSTRUCTOR table remains the same, while there are decreases in metrics for the temporary tables.
- The physical reads value for the INSTRUCTOR table is always the same.
- While there are fluctuations in CPU time, there is always a decrease in Elapsed time.

REFERNCES

1. https://docs.microsoft.com/en-us/sql/t-sql/statements/set-statistics-io-transact-sql?view=sql-server-ver15
2. https://www.yazilimkodlama.com/sql-server-2/normalizasyon-kurallari/
3. https://medium.com/trendyol-tech/sql-server-index-yap%C4%B1s%C4%B1-aba652828fe2
4. https://www.mshowto.org/sql-sorgu-optimizasyonu-nasil-yapilir-1.html#close