

Högskolan i Gävle

Projekt - Databasteknik

StudioX

Emre Özata

Emre10909@gmail.com

Linus Renström

Linus.renstrom@icloud.com

Brandon Adams Nkata

brandondalima@gmail.com

2024-10-30

Innehållsförteckning

1	Projektbeskrivning	1
2	Genomförande	2
2.1	Use-case diagram	3
2.1.1	Användningsfall 1	4
2.1.2	Användningsfall 2	4
2.1.3	Användningsfall 3	4
2.1.4	Användningsfall 4	4
2.1.5	Användningsfall 5	4
2.1.6	Användningsfall 6	4
2.1.7	Användningsfall 7	4
2.1.8	Användningsfall 8	4
2.1.9	Användningsfall 9	4
2.1.10	Användningsfall 10	5
2.1.11	Användningsfall 11	5
2.1.12	Användningsfall 12	5
2.1.13	Användningsfall 13	5
2.1.14	Användningsfall 14	5
2.1.15	Användningsfall 15	5
2.2	ER-Diagram och relationsmodell	7
2.2.1	ER-Diagram.....	7
2.2.2	Relationsmodell.....	8
2.3	Motivering och beskrivning till den fysiska databasen	13
2.3.1	Hur en användare hamnar i systemet	13
2.3.2	Hur en användare blir student, lärare alternativt examinator.....	14
2.3.3	Lärares betydelse för systemet	14
2.3.4	Examinators betydelse för systemet	16
2.3.5	Studentens betydelse för systemet	17
2.3.6	SQL-satser för att skapa tabeller och kopplingar mellan nycklar	18
2.3.7	Valet på namn av främmande nycklar	22
2.4	Termkatalog.....	23
2.5	Beskrivning av gränssnittet och kodexempel på koppling mellan databas och gränssnitt.	26
3	Reflektion över system och insatser inom grupp	30
4	Bilagor	31

1 Projektbeskrivning

Till detta projekt har det genomförts ett projekt där det har utvecklats en prototyp för ett administrativt system avsett för högskolor och universitet. Vår inriktning kring detta projekt var att skapa en databas som kan hantera både lärare samt studenter. Där studenter kan ansöka till kurser, program och kan se sina betyg. Läraren har utformats till att kunna sätta betyg, boka rum och administrera ett program.

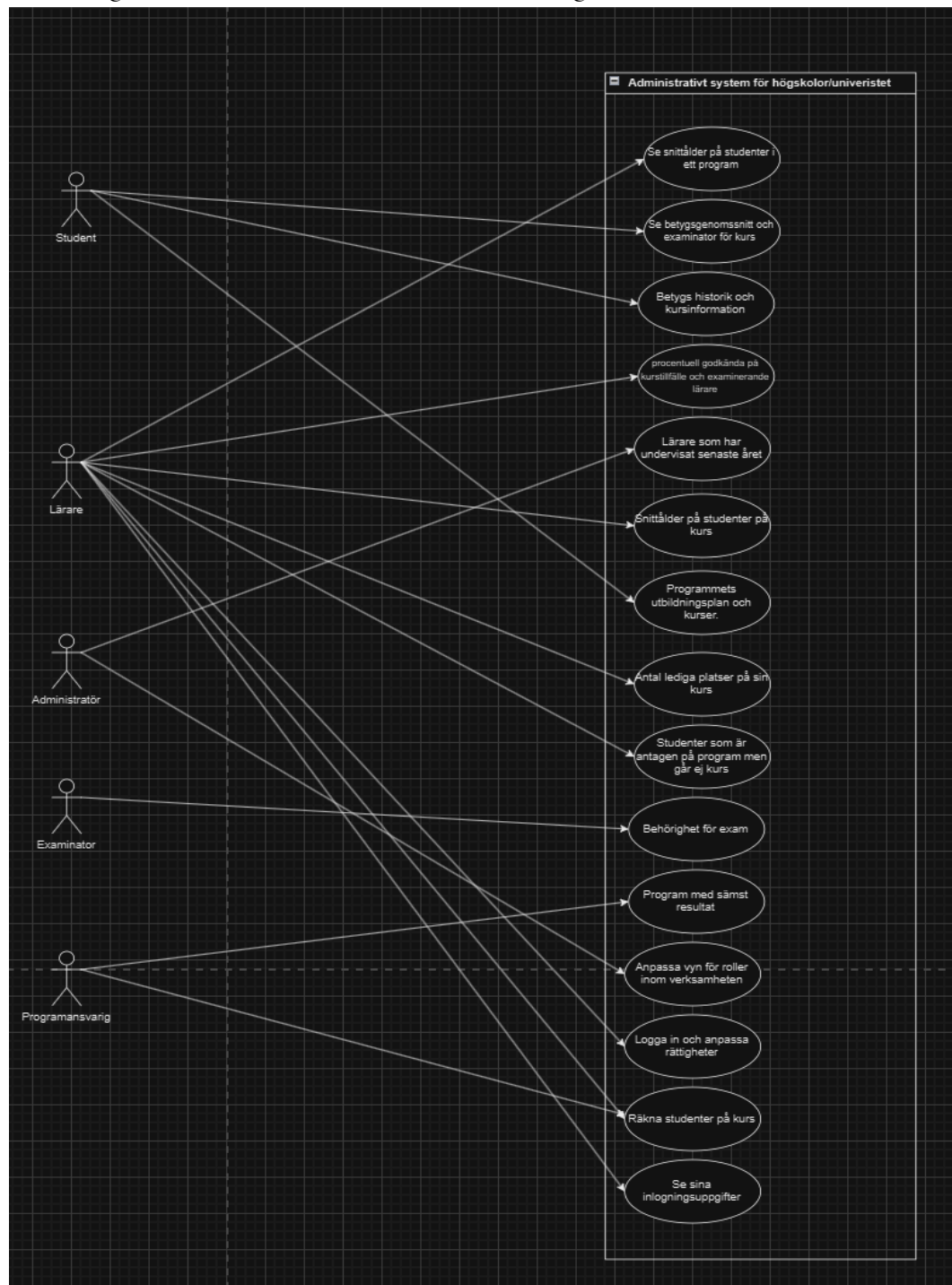
2 Genomförande

I denna del presenteras dem användningsfallen som databasen skulle klara av. En kort beskrivning av användningsfallen ges nedan och vilka specifika aktörer som skulle kunna ha nytta av användningsfallen med hänsyn till fokusområdet som systemet är användbar. Det har också beskrivits kring det skapade ER-Diagram, samt relationsmodell.

Se även infogad bilaga som innehåller dem detaljerade SQL-satser som beskrivs i denna del.

2.1 Use-case diagram

Till detta projekt har det skapats ett användningsfallsdiagram (Se figur 1). Varje användningsfall kommer att beskrivas nedan inom egna rubriker.



Figur 1 Visar det skapade användningsfallsdiagrammet.

2.1.1 Användningsfall 1

En lärare eller programansvarig skall kunna räkna antal studenter på en specifik kurs.

2.1.2 Användningsfall 2

En lärare skall kunna logga in på samma gränssnitt som studenter och systemet ska kunna kontrollera att det är en lärare och därav anpassa åtkomsträttigheter.

2.1.3 Användningsfall 3

Med detta användningsfall skall en lärare kunna se snittåldern på studenterna som går ett programtillfälle.

2.1.4 Användningsfall 4

En inloggad student kan se betygsgenomsnitt för ett kurstillfälle och dessutom se den examinerande lärare.

2.1.5 Användningsfall 5

På detta användningsfall är det tänkt att en student ska kunna kolla på sin betygshistorik och kursinformation på de bedömda kurserna.

2.1.6 Användningsfall 6

På detta användningsfall är det tänkt att en lärare skall kunna se procentuell andelen för godkända studenter på ett kurstillfälle och dessutom ser namn på den examinerande lärare.

2.1.7 Användningsfall 7

Med detta användningsfall skall det vara möjligt att för en aktör möjligtvis en administrator att hämta detaljer för alla lärare som examinerat minst ett kurstillfälle under ett år.

2.1.8 Användningsfall 8

Syftet för detta användningsfall är att en lärare skall kunna se snittåldern på studenterna som deltar på ett kurstillfälle.

2.1.9 Användningsfall 9

Detta användningsfall handlar om att en student som går ett program kan se de kurser som tillhör programmets utbildningsplan och vilka kurser de har anmält sig till. Dessa detaljer är bland annat programnamnet, programmets-startår osv.

2.1.10 Användningsfall 10

Med detta användningsfall är det tänkt att en lärare skall kunna se antal lediga platser på sin egen kurs förutsatt att vi vet vilket lärare ID som läraren har.

2.1.11 Användningsfall 11

Tanken med användningsfallet är att en lärare skall kunna lista studenter som är med på ett program men ännu inte deltar på någon kurs.

2.1.12 Användningsfall 12

Med detta användningsfall skall en examinator kunna se vilka studenter som är behörig samt ej behörig för att begära ut ett exam förutsatt att kravet är att studenten har 180 HP klara.

2.1.13 Användningsfall 13

Syftet för detta användningsfall är att en aktör möjligtvis en rektor, programansvarig skall kunna kolla vilket program som har sämst resultat per student. Programmen som jämförs är förutsatt att ha samma start år.

2.1.14 Användningsfall 14

Med detta användningsfall skall en administratör kunna tilldela en student att jobba som lärarassistent på en kurs. Den ska kunna lista detaljer på kursen, namn och betyg på studenter som går det kurstillfället.

2.1.15 Användningsfall 15

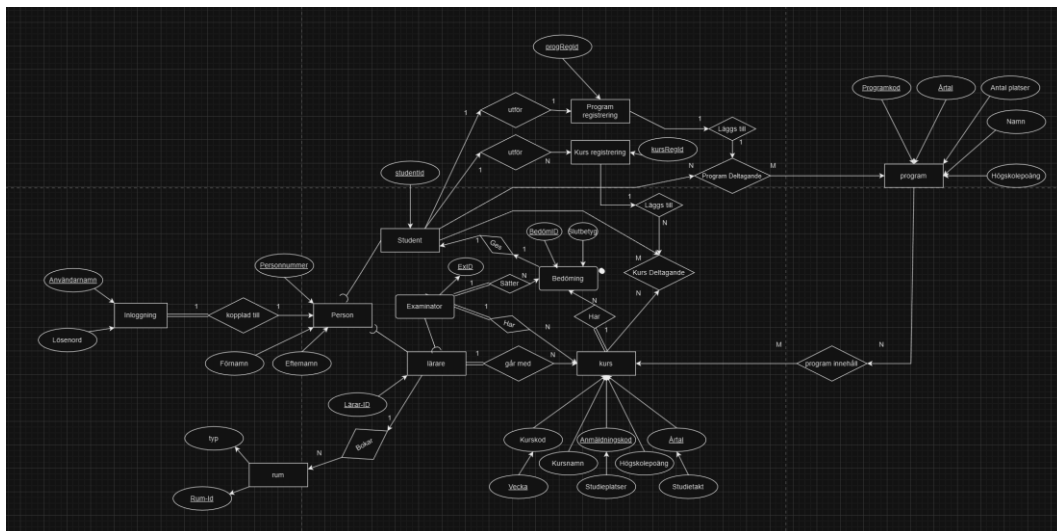
En inloggad lärare kollar sina inloggningsuppgifter. Där läraren får sitt användarnamn och lösenord.

2.2 ER-Diagram och relationsmodell

Till detta avsnitt kommer det att beskrivas kring det skapade ER-Diagrammet samt relationsmodell.

2.2.1 ER-Diagram

Detta ER-Diagram visar en databasstruktur för ett system som hanterar högskoleprogram, kurser, studenter och examinatorer (Se figur 2).



Figur 2 Visar en bild över ER-Diagrammet.



- **Person**: Symboliserar en individ med attribut som förnamn, efternamn och ID. En person kan ha roller likt "Student" eller "Lärare".
- **Student och Lärare**: Båda entiteter ärver från "Person". Studenter kan registrera sig på kurser samt program. Medan en lärare kan boka rum och anmäla sig att vara lärare på en kurs.
- **Kurs**: Innehåller information likt anmälningsskod, årtal och vecka.
- **Program**: Innehåller information kring programkod och årtal.

ER-Diagrammet visar också relationstyper likt 1 till N, M till N samt 1 till 1.

2.2.2 Relationsmodell










1. Examiners: Denna tabell lagrar information om examinatorer, denna tabell kommer att skapa ett unikt "examiner_id" och ett "teacher_id" som refererar till en lärare (Se tabell 1). Denna tabell är skapad för att koppla lärare till deras roll som examinator.

Tabell 1 Examiners

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
examiner_id	integer	NOT NULL	nextval('examiners_examiner_id_seq'::regclass)	
teacher_id	integer			



2. Course_participation: Denna tabell är sambandstabell och använder sig av flera nycklar främmande nycklar (Se tabell 2). Tabellen lagrar studenter som blivit antagen på kurser. Den innehåller exempelvis kolumnerna "student_id", "week" och "course_reg_id". Denna tabell håller koll på vilka kurser en student är antagen till.

Tabell 2 course_participation

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
student_id	integer	NOT NULL		 
start_year_course	integer	NOT NULL		 
week	integer	NOT NULL		 
registration_code	character varying(20)	NOT NULL		 
course_reg_id	integer			






3. Course_registrations: Denna tabell lagrar registrering av studenter på kurser. Den innehåller ett unikt "course_reg_id" och "student_id" (Se tabell 3). Tabellen håller koll på vilka kurser en student registrerar sig på.

Tabell 3 course_registrations

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
course_reg_id	integer	NOT NULL	nextval('registrations_reg_id_seq'::regclass)	
student_id	integer			











Courses: Tabellen är utformad för att lagra kursregistreingar, lärare, examinerator, takten på kursen, högskolepoäng och antal studenter som får plats på kursen (Se tabell 4). Detta medför till att databasen kan hantera olika kurser där det finns möjlighet att tilldela en lärare eller examinerator till en kurs.

Tabell 4 courses

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
registration_code	character varying	NOT NULL		
week	integer	NOT NULL		
start_year_course	integer	NOT NULL		
teacher_id	integer			
examiner_id	integer			
course_code	character varying			
name	character varying			
pace	numeric(3,2)			
credits	numeric(3,1)			
student_limit	integer			







Curriculum: Tabellens uppgift är att beskriva olika kursplaner med registreingskod, vecka, startår och programmets kod (Se tabell 5). Det används för att kunna skapa en struktur för en utbildningsplan som är kopplade till specifika program.

Tabell 5 curriculum

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
registration_code	character varying	NOT NULL		 
week	integer	NOT NULL		 
start_year_course	integer	NOT NULL		 
start_year_programs	integer	NOT NULL		 
program_code	character varying	NOT NULL		 



Grades: Denna tabell lagras information kring en students betyg, där varje resultat får ett unikt ID, kurs kod, student-ID och betyget som lagras som en siffra (Se tabell 6). Denna tabell kommer att anropas av examinatorn av programmet och underlättar för studenten om den vill se sina betyg eller vilken examinator som tilldelat betyget.

Tabell 6 grades

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
grade_id	integer	NOT NULL	nextval('grades_grade_id_seq'::regclass)	
examiner_id	integer			
registration_code	character varying			
week	integer			
year	integer			
student_id	integer			
grade	integer			


Login: Denna tabell lagrar information kring användarnamn, lösenord och personnummer som en främmande nyckel (Se tabell 7). Denna tabell skapades för att varje användare skall kunna logga in med unika inloggningsuppgifter.

Tabell 7 login

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
username	character varying(30)	NOT NULL		
password	character varying(30)			
ssn	bigint			





Persons: Denna tabell innehåller information likt personnummer och namn (Se tabell 8). Denna tabell används som en grund till att lagra information kring varje användare av systemet, där varje person kommer att krävas att ha ett unikt personnummer.

Tabell 8 persons

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
ssn	bigint	NOT NULL		
first_name	character varying(30)			
last_name	character varying(30)			



Program_participation: Denna tabell är en sambandstabell och har samma utformning som "Course_participation" där denna tabell innehåller flera främmande nycklar från andra tabeller (Se tabell 9). Denna tabell har skapats för att hantera en student som deltar på ett unikt program med unika årgångar.

Tabell 9 program_participation

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
student_id	integer	NOT NULL		
program_code	character varying	NOT NULL		
start_year_programs	integer	NOT NULL		
program_reg_id	integer			



Program_registrations: Lagrar en koppling från studenten och deras registreringar på ett program. Tabellen innehåller kolumnerna student-ID och ett unikt programmeringsregistrerings-ID (Se tabell 10). Denna tabell kommer att underlätta administrationen kring att godkänna eller neka en students ansökan till ett program.

Tabell 10 program_registrations

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
student_id	integer			
program_reg_id	integer	NOT NULL	nextval('program_registrations_program_reg_id_seq'::regclass)	



Programs: Tabellen innehåller kolumnerna programkod, år, namn på programmet, antal platser på programmet och högskolepoäng (Se tabell 11). Denna tabell används för att visa de program som erbjuds inom vårt skapade system.

Tabell 11 programs

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
program_code	character varying	NOT NULL		
start_year_programs	integer	NOT NULL		
name	character varying			
student_limit	integer			
credits	integer			



Rooms: Denna tabell har kolumnerna rum-ID, lärar-ID och vilken typ av rum det är (Se tabell 12). Denna tabell är skapad eftersom en lärare skall ha möjlighet till att boka ett rum för deras lektioner. Läraren får i denna tabell reda på vilken typ av rum det är innan det bokas.

Tabell 12 rooms

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
room_id	integer	NOT NULL	nextval('rooms_room_id_seq'::regclass)	
teacher_id	integer			
type	character varying			



Students: Tabellen lagrar information kring de studenter som är registrerade på högskolan eller universitetet och används för att kolla om en student verkligen tillhör denna verksamhet. Denna tabell lagrar personnummer samt ett unikt ID för varje student (Se tabell 13).

Tabell 13 students

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
student_id	integer	NOT NULL	nextval('students_student_id_seq'::regclass)	
ssn	bigint			

Teachers: Denna tabell lagrar information rörande lärare, denna tabell används i vårt system för att kolla om läraren tillhör verksamheten och att den har en lärarroll. Denna tabell innehåller ett unikt lärar-ID och ett personnummer för att koppla läraren till en unik person (Se tabell 14).

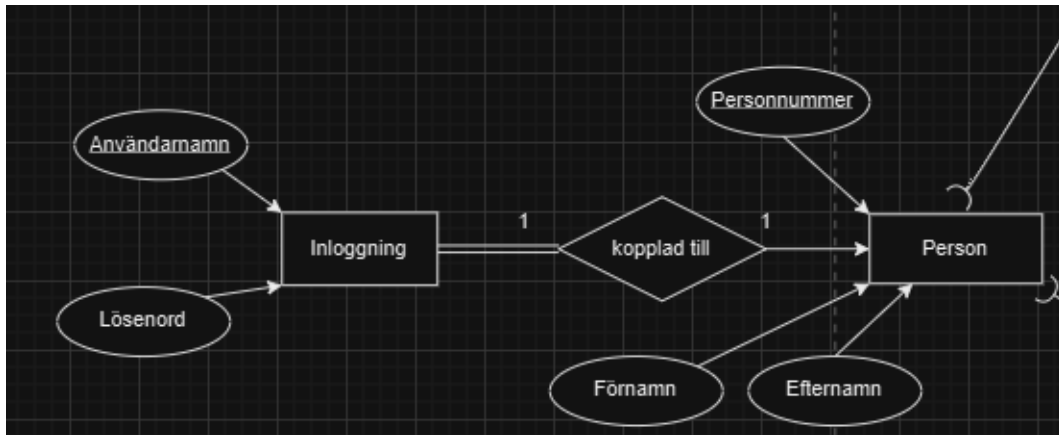
Tabell 14 teachers

Column	Typ	Inte Ingenting	Standardvärde	Begränsningar
teacher_id	integer	NOT NULL	nextval('teachers_teacher_id_seq'::regclass)	
ssn	bigint			

2.3 Motivering och beskrivning till den fysiska databasen

2.3.1 Hur en användare hamnar i systemet

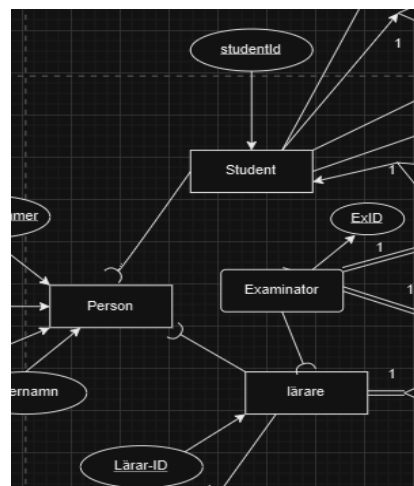
I systemet är tanken att en användare registreras till själva systemet genom att hamna i person-entiteten där personnummer, förnamn och efternamn sparas. Därefter kan ett inlogg skapas med hjälp av att personnummer blir en främmande nyckel i inloggning-entiteten (se figur 3). Med denna struktur så kan exempeldata användas senare med hjälp av *inserts* till övriga entiteter som ska representera användartyper (lärare, student och examinator).



Figur 3 - bilden visar hur en användare kommer med i systemet.

2.3.2 Hur en användare blir student, lärare alternativt examinator

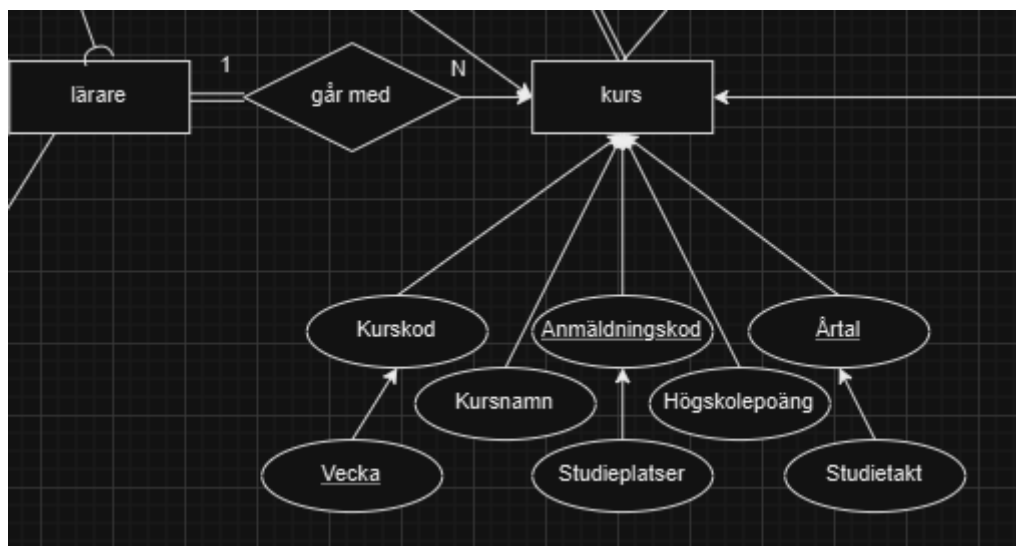
De olika användartyperna för programmet är student, lärare och alternativt examinator. Både lärare och elever ärver utav person-entiteten vilket innebär att person-numret blir en främmande nyckel i båda entiteterna. Som primärnyckel får både student och lärare en varsin primärnyckel där de särskiljs med student-id respektive lärare-id. För att bli en examinator måste du vara en lärare och endast en del lärare är examinatore. Därför användes arv för att få med lärare-id som främmande nyckel och en examinator får ett examinator-id som primärnyckel. Detta ger oss 3 olika entiteter som kan ha olika kopplingar till uppkommande övriga entiteter (se figur 4), vilket skapar varierande användningsfall för olika användartyper.



Figur 4 - bilden visar entiteterna student, lärare och examinator

2.3.3 Lärarens betydelse för systemet

Som en lärare ska du kunna undervisa tilldelade kurser och kunna få information om kurser den tillhör. Därför ska lärare ha en koppling till en kurs-entitet (se figur 5).

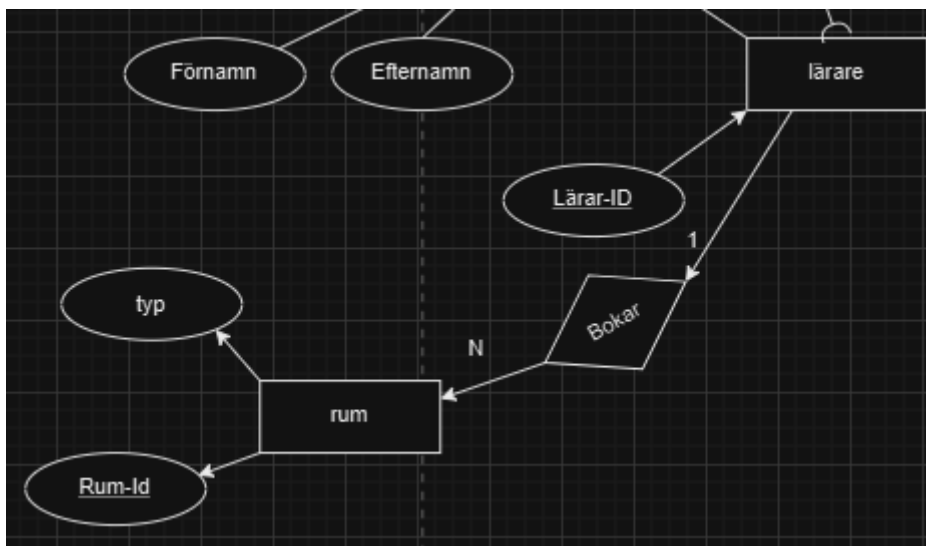


Figur 5 - bilden visar lärarens koppling till entiteten kurs.

För att kunna särskilja samma kurs som utförs olika år och terminer så används en sammansatt primärnyckel med attributen anmältningskod, vecka och årtal. Utöver primärnyckeln så har entiteten övriga attribut som ska efterlikna en verklig högskolekurs som poäng, studietakt, namn och antal platser. I kurs-entiteten blir även lärarens primärnyckel

en främmande nyckel. Detta ger oss överblick i kurs-entiteten om vilken lärare som undervisar kursen under en specifik tidsperiod, eftersom en lärare inte undervisar samma kurs varje år och termin.

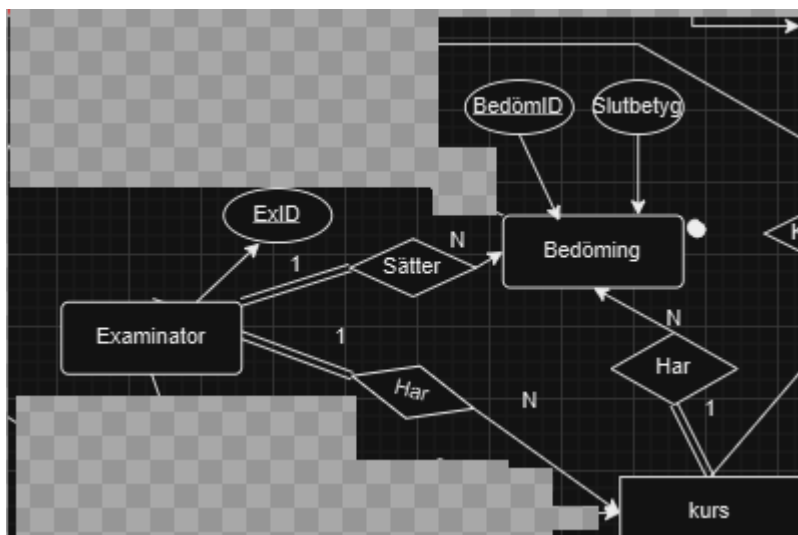
Utöver detta har lärare en koppling till rum. Läraren ska kunna boka ett eller flera rum, vilket innebär att lärar-id hamnar i rum-entiteten. Övriga attribut för rum är vilken typ av rum (datasal, föreläsningssal) och ett rum-id som primär nyckel. Om ett rum är bokad så står lärar-id i kolumnen, annars är värdet null (se figur 6).



Figur 5 - bilden visar lärarens koppling till rum-entiteten.

2.3.4 Examinatorns betydelse för systemet

En examinator i systemet ska kunna sätta betyg på en elev i en kurs. Detta registreras i en egen bedömningsentitet där examinator-id blir en främmande nyckel. Utöver de så får bedömning-entiteten kursens sammansatta primärnyckel vilket skapar en överblick över vilken kurs som blivit examinerad av en examinator. I bedömnings-entiteten används bedömnings-id som primärnyckel och vilket betyg som anges (se figur 7). Detta gör att endast en särskild roll (examinator) kan sätta betyg och kan enkelt hitta vilken kurs som bedömningen utfördes i.



Figur 6 - bilden visar entiteterna examinator, bedömning, kurs och hur de har en relation till varandra.

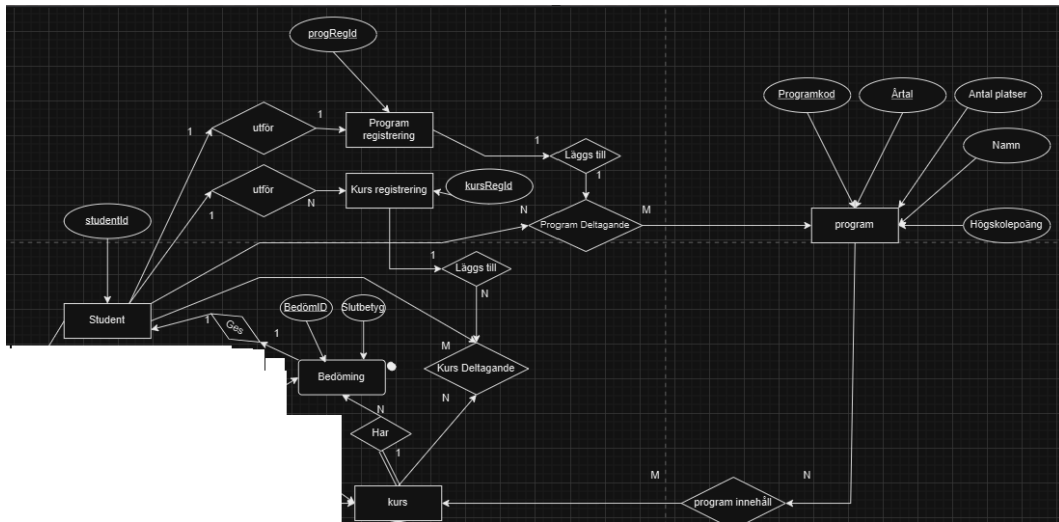
2.3.5 Studentens betydelse för systemet

Den användartyp som uppvisar flest kopplingar inom systemet är studenten (se figur 8). Syftet med en student är att möjliggöra registreringar till program och kurser, vilka i sin tur kopplas till en deltagande entitet. När en student registrerar sig för en kurs tilldelas den ett registrerings-ID, där studentens ID blir en främmande nyckel i kursregistreringsentiteten. Detta beror på att en student kan ansöka till flera kurser. I detta sammanhang skapas ett registrerings-ID som i sin tur kopplas till kursdeltagande, vilket indikerar att studenten har blivit antagen till kursen. Denna relation representeras av en separat tabell där primärnyckeln utgörs av en sammansatt nyckel bestående av främmande nycklar. Dessa främmande nycklar innefattar primärnycklarna från kursentiteten (år, vecka, registreringskod) samt primärnyckeln från kursregistreringen (registrerings-ID). Denna struktur ger en tydlig översikt över vilka studenter som deltar i vilka kurser.

En liknande princip tillämpas när en student ansöker om ett program, där den huvudsakliga skillnaden ligger i att en student endast kan registreras för ett enda program. Inom programentiteten utgör programkoden och året den sammansatta primärnyckeln, eftersom ett program bär samma kod varje år. Förutom primärnyckeln inkluderar entiteten ytterligare attribut som antal platser, namn och antal poäng, vilket efterliknar ett verkligt utbildningsprogram.

I ett program ska det finnas kurser. Eftersom kurser kan tillhöra flera olika program, och program kan innehålla flera kurser, skapas en egen tabell som fungerar som en kursplan. Den sammansatta primärnyckeln i kursplantabellen består därför av primärnycklarna från både program- och kursentiteterna. Detta skapar en effektiv struktur för att identifiera vilka kurser som ingår i ett specifikt program.

Slutligen ska studenten ha en relation till bedömningstabellen, där bedömningsentiteten får studentens primära nyckel (student-ID) som främmande nyckel. Denna struktur underlättar identifieringen av vilken student som har erhållit vilket betyg, vilken kurs, samt vilken examinator som har genomfört bedömningen.



Figur 7 - figuren visar studentens kopplingar till olika entiteter, samt kopplingen mellan kurser och program.

2.3.6 SQL-satser för att skapa tabeller och kopplingar mellan nycklar

För att kunna koppla tabeller (det vill säga att koppla primärnycklar som främmande nycklar) så var vissa tabeller tvungna att skapas och sedan kopplas. Men skapandet av tabellerna och kopplingar mellan nycklar gav följande:

```
create table login(
username VARCHAR(30),
password VARCHAR(30),
ssn INTEGER,
PRIMARY KEY (username)
)
```

```
create table persons(
ssn INTEGER,
first_name VARCHAR (30),
last_name VARCHAR (30),
PRIMARY KEY (ssn)
)
```

```
Alter table login
```

```
ADD FOREIGN KEY (ssn) REFERENCES persons(ssn)
```

```
create table students(  
  student_id SERIAL,  
  ssn INTEGER,  
  PRIMARY KEY (student_id),  
  FOREIGN key (ssn) REFERENCES persons(ssn)  
)
```

```
create table registrations(  
  reg_id SERIAL,  
  student_id INTEGER,  
  PRIMARY KEY (reg_id),  
  FOREIGN KEY (student_id) REFERENCES students(student_id)  
)
```

```
create table teachers(  
  teacher_id SERIAL,  
  admin_id INTEGER,  
  PRIMARY KEY (teacher_id)  
)
```

```
create table admins(  
  admin_id SERIAL,  
  reg_id INTEGER,  
  ssn INTEGER,  
  registered_student_id INTEGER,  
  PRIMARY KEY (admin_id),  
  FOREIGN KEY (reg_id) REFERENCES registrations (reg_id),  
  FOREIGN KEY (ssn) REFERENCES persons(ssn)  
)
```

```
ALTER TABLE teachers  
add foreign key (admin_id) REFERENCES admins(admin_id)
```

```
create table registrated_students(  
  registered_student_id SERIAL,
```

```
grade_id INTEGER,  
primary key (registered_student_id)  
)
```

```
alter table admins  
add foreign key (registered_student_id)  
references registrated_students(registered_student_id)
```

```
create table examiners(  
examiner_id SERIAL,  
teacher_id INTEGER,  
PRIMARY KEY (examiner_id),  
foreign key (teacher_id) REFERENCES teachers(teacher_id)  
)
```

```
create table grades(  
grade_id SERIAL,  
grade VARCHAR(2),  
examiner_id INTEGER,  
registration_code VARCHAR,  
week INTEGER,  
year INTEGER,  
primary key (grade_id),  
foreign key (examiner_id) REFERENCES examiners(examiner_id)  
)
```

```
create table courses(  
registration_code VARCHAR,  
week INTEGER,  
year INTEGER,  
teacher_id INTEGER,  
examiner_id INTEGER,  
registered_student_id INTEGER,  
course_code VARCHAR,  
name VARCHAR,  
pace DECIMAL(3,2),  
credits DECIMAL(3,1),
```

```

student_limit INTEGER,
primary key (registration_code, week, year),
foreign key (teacher_id) references teachers(teacher_id),
foreign key (examiner_id) references examiners(examiner_id),
foreign key (registered_student_id) references registered_students(registered_student_id)
)

```

```

alter table grades
add foreign key (registration_code, week, year)
references courses(registration_code, week, year)

```

```

create table curriculum(
curriculum_id SERIAL,
year_of_validity INTEGER,
PRIMARY KEY (curriculum_id)
)

```

```

create table rooms(
room_id SERIAL,
teacher_id INTEGER,
type VARCHAR,
PRIMARY KEY (room_id),
foreign key (teacher_id) REFERENCES teachers(teacher_id)
)

```

```

create table programs(
program_code VARCHAR,
year INTEGER,
registered_student_id INTEGER,
curriculum_id INTEGER,
name VARCHAR,
student_limit INTEGER,
credits INTEGER,
PRIMARY KEY (program_code, year),
foreign key (registered_student_id) references registered_students(registered_student_id),
foreign key (curriculum_id) references curriculum(curriculum_id)
)

```

)

```
create table program_content(  
  curriculum_id INTEGER,  
  registration_code VARCHAR,  
  week INTEGER,  
  year INTEGER,  
  PRIMARY KEY (curriculum_id, registration_code, week, year),  
  FOREIGN KEY (registration_code, week, year)  
  REFERENCES courses(registration_code, week, year),  
  FOREIGN KEY (curriculum_id)  
  REFERENCES curriculum(curriculum_id)  
)
```

```
alter table registered_students  
add foreign key (grade_id) references grades(grade_id)
```

2.3.7 Valet på namn av främmande nycklar

I vissa databaser så byts namnet på främmande nyckeln så den inte heter likadant som primärnyckeln från den givna entiteten. Men eftersom vi behöll samma namn på främmande nycklar och primärnycklar medförde att vi enkelt kunde använda natural joins till våra SQL-satser som används för användningsfall.

2.4 Termkatalog

Här bifogas det figurer från den skapade termakatalogen (Se tabell 15-28).

Tabell 1185: Termakatalog Students

STUDENTS							
Attribut	Beskrivning	Datatyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
student_id	Unikt ID för student	Integer			Students	Ja	
ssn	Personnummer	Bigint			Students		Persons

Tabell 1176: Termakatalog Courses

Courses							
Attribut	Beskrivning	Datatyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
registration_code	Unik kod för kurs	Character varying			Courses	Ja	
week	Vecka	Integer			Courses	Ja	
start_year_course	Startår kurs	Integer			Courses	Ja	
teacher_id	Lärar-ID	Integer			Courses		Teachers
examiner_id	Ansvarig examiner	Integer			Courses		Examiners
credits	Kurspoäng	Numeric	3	1	Courses		
course_code	Kurskod	Character varying			Courses		
name	Kursens namn	Character varying			Courses		
pace	Hastighet på kurs	Numeric	3	2	Courses		
student_limit	Antal plats på kurs	Integer			Courses		

Tabell 16: Termakatalog Examiners

Examiners							
Attribut	Beskrivning	Datatyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
examiner_id	Examinator ID	Integer			Examiners	Ja	
teacher_id	Lärar-ID	Integer			Examiners		Teachers

Tabell 1158: Termakatalog Teachers

Attribut	beskrivning	datatyp	längd	decimal	tabell	primär nyckel	främmande nyckel
teacher_id	lärarens id	Integer			teachers	ja	
ssn	personnummer	BIGINT			teachers		persons

Tabell 19: Termakatalog Grades

Grades							
Attribut	Beskrivning	Datatyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
grade_id	Unikt id för betyg	Integer			Grades	Ja	
examiner_id	Examinator ID	Integer			Grades		Examiners
registration_code	Reg-kod från kurs	Character varying			Grades		Courses
week	Vecka	Integer			Grades		Courses
year	år	Integer			Grades		Courses
student_id	Student-ID	Integer			Grades		Students
grade	Betyg - 1 till 5	Integer			Grades		

Tabell 2023: Termakatalog Course_participation

Course_participation							
Attribut	Beskrivning	Datatyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
student_id	Student-ID	Integer			Course_participation	Ja	Students
start_year_course	Start år för kurs	Integer			Course_participation	Ja	Course
week	Vecka för kursstart	Integer			Course_participation	Ja	Course
registration_code	Registrerings-ID	Character varying	20		Course_participation	Ja	Course
course_reg_id	Registreringskod	Integer			Course_participation		Course_registrations

Tabell 2122: Termakatalog Rooms

Attribut	beskrivning	datatyp	längd	decimal	tabell	primär nyckel	främmande nyckel
room_id	rummets id	Integer			rooms	ja	
teacher_id	lärarens id	Integer			rooms		teachers
type	vilken typ av sal (datasal, tentasal)	VARCHAR			rooms		

Tabell 2221: Termakatalog Curriculum

Curriculum							
Attribut	Beskrivning	Datatyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
registration_code	Registrerings-ID	Character varying			Curriculum	Ja	Course
week	Vecka	Integer			Curriculum	Ja	Course
start_year_course	Start år för kurs	Integer			Curriculum	Ja	Course
start_year_program	Start år för program	Integer			Curriculum	Ja	Programs
program_code	Program kod	Character varying			Curriculum	Ja	Programs

Tabell 2320: Termakatalog Course_registrations

Course_registrations							
Attribut	Beskrivning	Datatyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
course_reg_id	Kurs-ID	Integer			Course_registrations	Ja	
student_id	Student-ID	Integer			Course_registrations		Students

Tabell 2419: Termakatalog program_participation

program_participation							
Attribut	Beskrivning	Datatyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
student_id	Student-ID	Integer			Program_participation	Ja	Students
program_code	Program kod	Character varying			Program_participation	Ja	Programs
start_year_programs	Start år program	Integer			Program_participation	Ja	Programs
program_reg_id	Program reg-kod	Integer			Program_participation		Programs

Tabell 2275: Termakatalog Login

Login							
Attribut	Beskrivning	Datotyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
username	Användarnamn	Character varying	30		Login	Ja	
password	Lösenord	Character varying	30		Login		
ssn	Personnummer	bigint			Login		persons

Tabell 2266: Termakatalog Persons

persons							
Attribut	Beskrivning	Datotyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
ssn	Personnummer	bigint			Persons	Ja	
first_name	Förnamn	Character varying	30		Persons		
last_name	Efternamn	Character varying	30		Persons		

Tabell 2257: Termakatalog program_registrations

program_registrations							
Attribut	Beskrivning	Datotyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
student_id	Student-ID	Integer			program_registrations		Students
program_reg_id	Program reg-kod	Integer			program_registrations	Ja	

Tabell 2248: Termakatalog programs

programs							
Attribut	Beskrivning	Datotyp	Längd	Decimal	Tabell	Primär-nyckel	Främmande nyckel
program_code	Program-kod	Character varying			Programs	Ja	
start_year_programs	Start år program	Integer			Programs	Ja	
name	Program namn	Character varying			Programs		
student_limit	Max antal studenter	Integer			Programs		
credits	Högskolepoäng	Integer			Programs		

2.5 Beskrivning av gränssnittet och kodexempel på koppling mellan databas och gränssnitt.

I vårt gränssnitt har vi två olika vyer beroende på vad du loggar in som. En som är lärare och en som är student. I lärarvyn kan du se kurser du undervisar och sätta betyg på en specifik elev i en specifik kurs. I studentvyn så kan du registrera till en specifik kurs, hoppa av en specifik kurs, se sina registrerade kurser och kan se alla kurser som finns tillgängliga.

För att kunna ansluta till databasen från eclipse så behövs en jdbc nedladdat. Efter detta skapas en connection med hjälp av singleton mönstret. Här är konstruktorn private och man kan endast få en instans med hjälp av den statiska metoden getInstance() (se figur 9).

```
public class DataBaseConnection {
    private String url;
    private String username;
    private String password;

    private Connection connection;
    private static DataBaseConnection dbInstance = null;

    private DataBaseConnection() {
        try {
            this.url = "jdbc:postgresql://node143915-postgresql.jls-sto2.elastx.net:11070/db_emre_group";
            this.username = "emre_group";
            this.password = "emre_group_pwd";
            this.connection = DriverManager.getConnection(url, username, password);
        } catch (Exception e) {
            System.out.println("problem step 1");
            e.printStackTrace();
        }
    }

    public static DataBaseConnection getInstance() {
        try {
            if (dbInstance == null) {
                dbInstance = new DataBaseConnection();
            }
        } catch (Exception e) {
            System.out.println("problem step 3");
        }
        return dbInstance;
    }
}
```

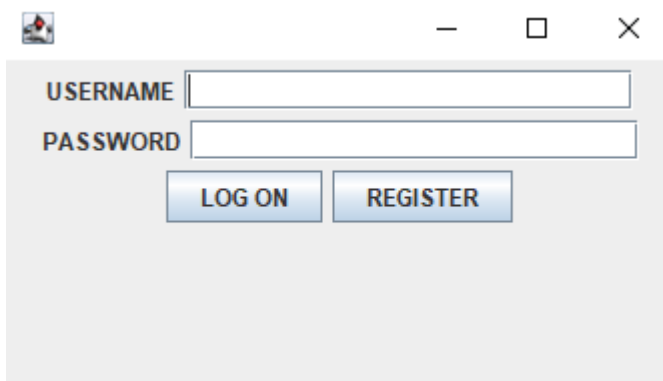
Figur 8 - bilden visar hur databasen kopplas till eclipse, med hjälp av singleton mönster.

Utöver instansen måste även en Connection metod finnas så att övriga data-access-object klasser kan få en koppling till databasen (se figur 10).

```
public Connection getConnection() {  
  
    try {  
  
    } catch (Exception e) {  
        System.out.println("Problem step 4");  
    }  
  
    return this.connection;  
}
```

Figur 9 - bilden visar metoden getConnection som ger koppling till övriga Dao-klasser

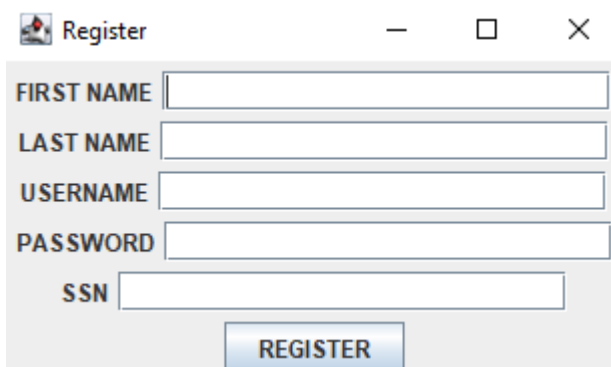
I gränssnittet skapades 4 olika vyer. Registrering, login, student och lärare. I login-vyn visas följande (se figur 11).

A screenshot of a Java Swing window titled 'Login'. It contains two text input fields labeled 'USERNAME' and 'PASSWORD'. Below the fields are two buttons: 'LOG ON' and 'REGISTER'.

Figur 10 - bilden visar login gränssnittet

Login-vyn är kopplad till vår login-tabell med hjälp av select-satser som kollar om användarnamn och lösenord matchar.

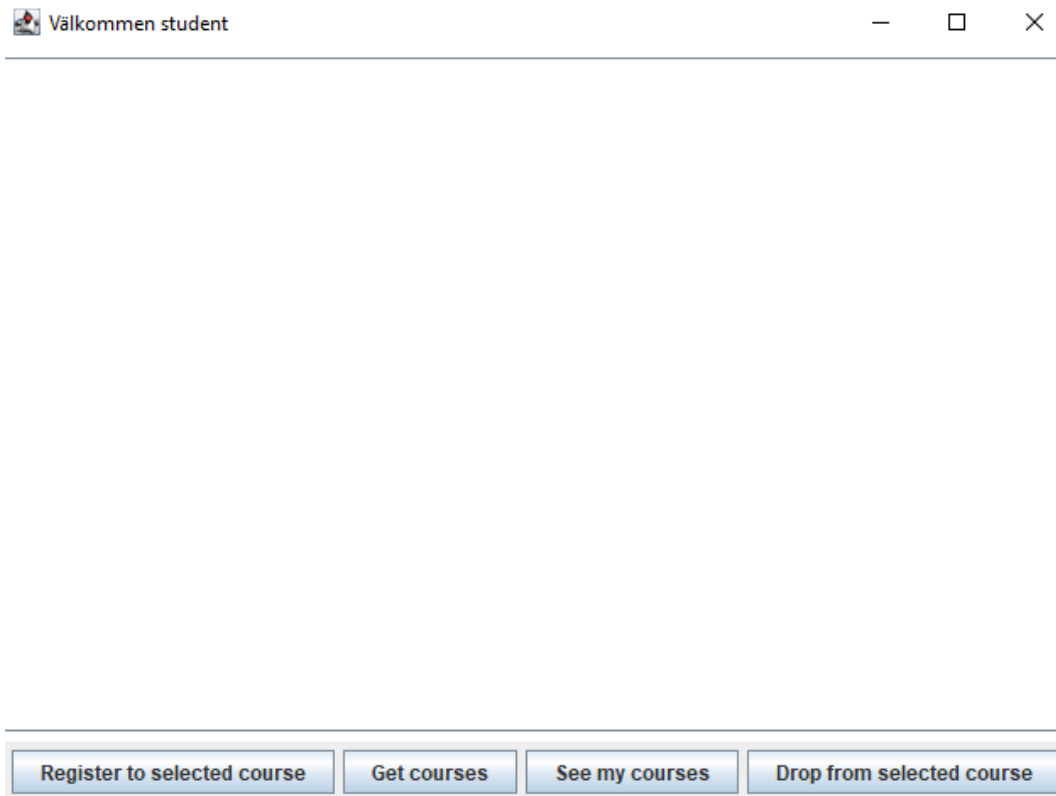
I registreringsvyn visas följande (se figur 12).

A screenshot of a Java Swing window titled 'Register'. It contains five text input fields labeled 'FIRST NAME', 'LAST NAME', 'USERNAME', 'PASSWORD', and 'SSN'. Below the fields is a single button labeled 'REGISTER'.

Figur 11 - bilden visar registreringsvyn

Registreringsvyn är kopplad till både login och person tabellen. Här skapas ett användarnamn, lösenord och en person med hjälp av insert into satser till login och person tabellen.

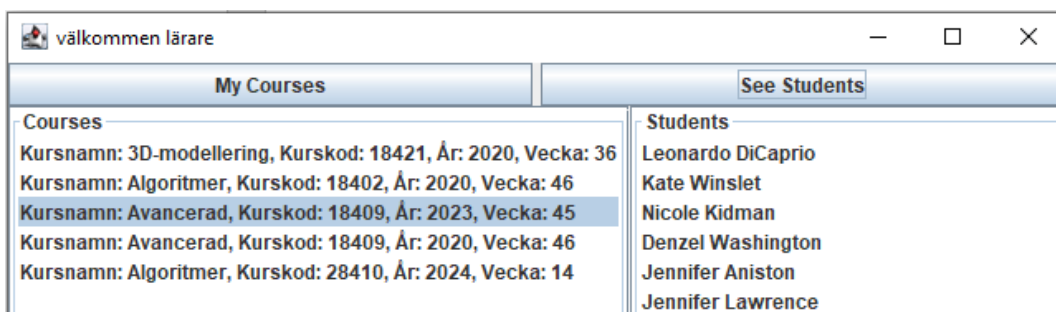
I studentvyn visas följande (se figur 13).



Figur 12 - bilden visar studentvyn

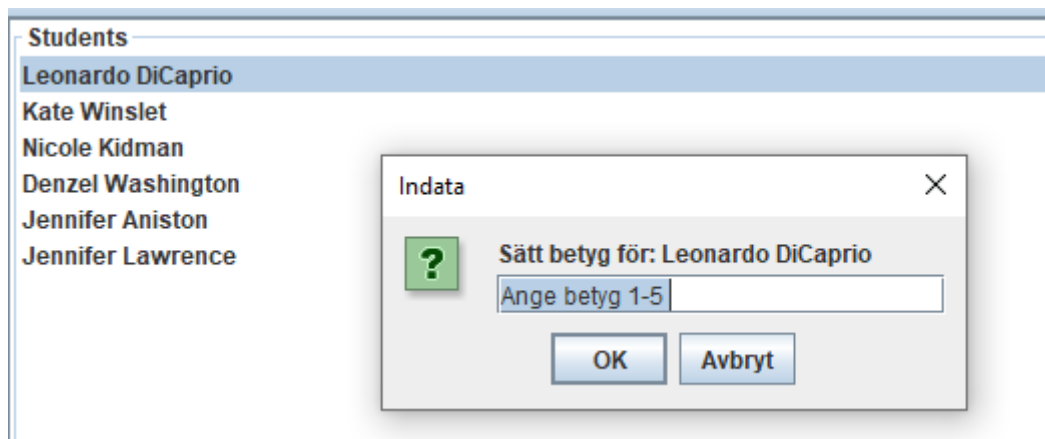
I studentvyn har vi stark koppling till kurser tabellen. Med hjälp av select satser och joins kan vi få upp namn på kurser och med hjälp av inserts kan vi registrera oss till kurs.

I lärarvyn visas följande (se figur 14).

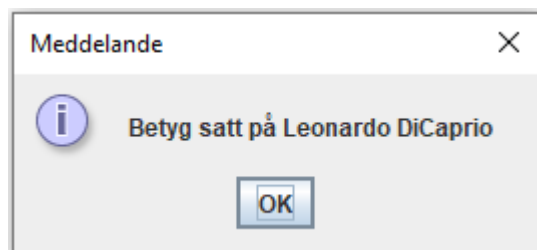


Figur 14 - bilden visar lärarvyn

Med hjälp av select satser och joins kan kurser och namnet på studenter som är deltagande på kursen visas. Om en student klickas på sker följande (se figur 15).



Figur 13 - bilden visar vad som händer om man trycker på en elev som tillhör en viss kurs. Här kan ett betyg sättas på den in klickade studenten genom att ange ett betyg, då sker följande (se figur 16).



Figur 14 Meddelande att betyget är satt.

Ett exempel på hur koden ser ut när man sätter betyg ges av följande figur (se figur 17 och 18).

```
String setGradesql = "INSERT INTO grades (examiner_id, registration_code, week, year, student_id, grade) VALUES (?, ?, ?, ?, ?, ?)";
```

Figur 17 - bilden visar sql-satsen som används i koden

```

preparedStatement = myConnection.prepareStatement(setGradesql);
preparedStatement.setInt(1, grade.getExaminerId());
preparedStatement.setString(2, grade.getRegistrationCode());
preparedStatement.setInt(3, grade.getWeek());
preparedStatement.setInt(4, grade.getYear());
preparedStatement.setInt(5, grade.getStudentId());
preparedStatement.setInt(6, grade.getGrade());

preparedStatement.executeUpdate();
System.out.println("la till examiner: " + grade.getExaminerId()
+ " kod " + grade.getRegistrationCode() + " vecka " + grade.getWeek() +
" år " + grade.getYear() + " studentId: " + grade.getStudentId() +
" betyg: " + grade.getGrade());

// myConnection.commit();
preparedStatement.close();

} catch (SQLException e) {
    e.printStackTrace();
}
}

```

Figur 18 - Bilden visar dao-klassen som har metoden setGrade. Här används vår sql-sats som deklarerats i en String som "setGradesql".

Utöver detta så används dao-klasser i kontroller klasser och våra GUI-klasser känner igen kontroller klasserna.

3 Reflektion över system och insatser inom grupp

Vi anser att vårt databassystem täcker den stora helheten inom verksamheten. Genom att påbörja skapandet ett ER-Diagram så fick vi i gruppen en tanke kring vad som behövdes för att täcka grunden för verksamheten. Detta gav oss en bra start inom projektet, med det skapade ER-Diagrammet kunde vi enkelt kunna skapa vår relationsmodell samt användningsfallsdiagrammet. Efter vi har utfört flertal sql kommandon för att simulera vårt system så anser vi att vår lösning på systemet är bra och att det täcker de mest kritiska punkterna som behövs inom en högskola/universitets verksamhet.

Vi har haft det väldigt enkelt med att arbeta tillsammans eftersom vi har använt oss av Git Lab samt använt oss av en tjänst för att skapa vår databas i molnet.

En förbättringspunkt som vi i efterhand är att när vi skapade våra tabeller kunde vi ha använt oss av domäner för att utföra kontroller. Dessa kontroller skulle exempelvis kunna kontrollera dubblett av studenter på kurs eller att kontrollera en students ålder så att en student inte är exempelvis 120 år gammal.

Vi har haft en bra gruppdynamik inom grupparbetet, där vi alla har deltagit vid skapandet av systemet. Vi alla har bidragit med vår egen kunskap inom databasteknik och tillsammans har vi utvecklat ett välfungerande system. Vi alla har varit delaktiga inom arbetet och vi anser att alla har bidragit lika mycket vid skapandet av systemet.

4 Bilagor



DatabasteknikProje
ktSatser(användning

Dubbelklicka på ikonerna ovan för att öppna PDF-filen med satserna som används för de tidigare nämnda användningsfallen.