Laboration 2

Laborant: Isak Nilsson
Program: Högskoleingenjör Datateknik
Datum: 2024-04-25

Innehållsförteckning

sid

Inledning

Syftet med denna laboration är att få ett djupare förståelse för hur man designar och hanterar databaser, i denna laboration genom hanteringen av en databas för ett läroverk där information kring fakulteter,kurser,lärare och studenter sparas. Laborationen gick ut på att designa databasen med val av primärnycklar, kandidatnycklar och relationer mellan tabeller, skapa triggers för att hantera förändringar i databasen samt att skapa tilldelade “Queries” som hanterade filtrering av specifik data från databasen.

Relationsmodell

Student
Innehåller studenter som är inskrivna på lärosätet där denne deltar i minst en kurs som antingen tillhör “Faculty of Science” eller “Faculty of Arts”.

Attribut

spid		
	Beskrivning	Personligt id nummer för studenten
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
Name		
	Beskrivning	Studentens namn
	Datatyp	char(20)
	Verifiering	-
	NULL ok	Nej
Year		
	Beskrivning	identifieringskod för året studenten går
	Datatyp	char(1)
	Verifiering	F,S,J,E
	NULL ok	Nej
sup		
	Beskrivning	personligt id för studentens handledare
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Ja

Nycklar

Primärnyckel	spid
Kandidatnyckel	spid
Främmandenyckel	sup (Refererar → teacher.tpid)

Teacher
Innehåller lärare som håller i minst en kurs vid någon “Department”

Attribut

tpid		
	Beskrivning	Personligt id nummer för lärare
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
Forename		
	Beskrivning	Lärarens förnamn
	Datatyp	char(20)
	Verifiering	-
	NULL ok	Nej
Surname		
	Beskrivning	Lärarens efternamn
	Datatyp	char(20)
	Verifiering	-
	NULL ok	Nej
dept		
	Beskrivning	Kod som visar vilken department läraren tillhör
	Datatyp	Char(2)
	Verifiering	-
	NULL ok	Nej
Rank		
	Beskrivning	Den rank som läraren har
	Datatyp	ENUM
	Verifiering	“asis prof”, ”Professor”, “Assoc prof”
	NULL ok	Nej

Nycklar

Primärnyckel	tpid
Kandidatnyckel	tpid
Främmandenyckel	dept (Refererar → department.dcode)

Course

Innehåller de kurser som ges på lärosätet

Attribut

ccode		
	Beskrivning	Kursens unika identifieringskod
	Datatyp	char(5)
	Verifiering	-
	NULL ok	Nej
Name		
	Beskrivning	Kursens namn
	Datatyp	char(20)
	Verifiering	-
	NULL ok	Nej
Does_teach		
	Beskrivning	tpid för den lärare som lär ut i kursen
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
Responsible		
	Beskrivning	tpid för den lärare som är ansvarig för kursen
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej

Nycklar

Primärnyckel	ccode
Kandidatnyckel	ccode
Främmandenyckel	Does_teach, responsible (Refererar → teacher.tpid)

Grade

Innehåller värden för att avgöra vilket bokstavsbetyg ett viss poäng ger.

Attribut

letter		
	Beskrivning	Den bokstav ett visst betyg ger
	Datatyp	char(1)
	Verifiering	“A”, “B”, “C”, “D”, “E”, “F”
	NULL ok	Nej
MIN		
	Beskrivning	Min-värdet för varje bokstavsbetyg
	Datatyp	Integer
	Verifiering	0 <= x <= 100
	NULL ok	Nej
MAX		
	Beskrivning	Max-värdet för varje bokstavsbetyg
	Datatyp	Integer
	Verifiering	0 <= x <= 100
	NULL ok	Nej

Nycklar

Primärnyckel	letter
Kandidatnyckel	letter
Främmandenyckel	-

Faculty

Innehåller information om varje fakultet som finns på lärosätet

Attribut

fcode		
	Beskrivning	Unik kod för varje fakultet
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
name		
	Beskrivning	Namnet på fakulteten
	Datatyp	char(20)
	Verifiering	-
	NULL ok	Nej

Nycklar

Primärnyckel	fcode
Kandidatnyckel	fcode
Främmandenyckel	-

Department

Innehåller information om alla olika avdelningar på lärosätet

Attribut

dcode		
	Beskrivning	Unik kod för varje avdelning
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
name		
	Beskrivning	Namn på avdelningen
	Datatyp	char(20)
	Verifiering	-
	NULL ok	Nej
fcode		
	Beskrivning	Unik kod för den fakultet som avdelningen tillhör
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej

Nycklar

Primärnyckel	dcode
Kandidatnyckel	dcode
Främmandenyckel	fcode (Referar → faculty.fcode)

Can_teach

Innehåller information om vilka lärare som får lära ut vilka kurser

Attribut

tpid		
	Beskrivning	Unik kod för varje lärare
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
ccode		
	Beskrivning	Unik kod för varje kurs
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej

Nycklar

Primärnyckel	tpid + ccode
Kandidatnyckel	tpid + ccode
Främmandenyckel	tpid, ccode (Refferer ar →teacher.tpid & course.ccode)

Committee

Innehåller information om de olika kommitteer som finns på lärosätet

Attribut

xcode		
	Beskrivning	Unik kod för varje kommitte
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
name		
	Beskrivning	Namn på kommitte
	Datatyp	char(20)
	Verifiering	-
	NULL ok	Nej

Nycklar

Primärnyckel	xcode
Kandidatnyckel	xcode
Främmandenyckel	-

stud_course

Innehåller information för varje kurs en viss student har tagit

Attribut

dcode		
	Beskrivning	Personligt id nummer för studenten
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
ccode		
	Beskrivning	Unik kod för varje kurs
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
grade		
	Beskrivning	Det betyg studenten fått på kursen
	Datatyp	Integer
	Verifiering	0 <= x <= 100
	NULL ok	Nej

Nycklar

Primärnyckel	spid + ccode
Kandidatnyckel	spid + ccode
Främmandenyckel	spid,ccode (Reffererar →student.spid & course.ccode)

teach_comm

Innehåller information om vilka lärare som tillhör en viss kommitte

Attribut

xcode		
	Beskrivning	Unik kod för varje kommitte
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej
tpid		
	Beskrivning	Unik kod för varje lärare
	Datatyp	Integer
	Verifiering	>0
	NULL ok	Nej

Nycklar

Primärnyckel	xcode + tpid
Kandidatnyckel	xcode + tpid
Främmandenyckel	xcode,tpid (Refferar → committe.xcode & teacher.tpid)

Slutsatts

Tidsåtgången för hela laborationen har legat på ca åtta timmar. Genom att utföra laborationen har jag lärt mig grunderna i SQL och databashantering genom praktiskt arbete. Jag har fått en djupare förståelse för hur man skapar, manipulerar och frågar en databas om information. Detta har gett mig en grund för att fortsätta utforska och arbeta med SQL i framtida projekt.

QUERIES

1. How many courses does each department gives? Include departments (if any) that does not give any courses

```
SELECT department.name, COUNT(course.ccode)
FROM department
LEFT JOIN course ON department.dcode = SUBSTR(course.ccode,1,2)
GROUP BY department.dcode
```

2. List the teachers that does not teach this semester

```
SELECT  teacher.surname,teacher.forename
FROM teacher LEFT JOIN course
ON course.does_teach = teacher.tpid
WHERE course.does_teach IS NULL
```

3. List the members in the committee "Course Planning" (name, department)

```
SELECT teacher.surname, teacher.forename, department.name
FROM teacher
JOIN teach_comm ON teacher.tpid = teach_comm.tpid
JOIN committee ON teach_comm.xcode = committee.xcode
JOIN department ON teacher.dept = department.dcode
WHERE committee.name = "Course Planning"
```

4. List the teachers that are responsible for more than two courses

```
SELECT teacher.surname, teacher.forename,COUNT(*) AS NumberOfClasses
FROM teacher
JOIN course ON course.responsible = teacher.tpid
GROUP BY teacher.forename, teacher.surname
HAVING COUNT(*) > 2
```

5. List the teachers that can teach the course "Linear Algebra"

```
SELECT teacher.surname, teacher.forename
FROM teacher
JOIN can_teach ON can_teach.tpid = teacher.tpid
JOIN course ON can_teach.ccode = course.ccode
WHERE course.name = "Linear Algebra"
```

6. List, for each student, the number of course the student takes.

```
SELECT students.sname, COUNT(stud_course.CCODE)
FROM students
JOIN stud_course ON students.spid = stud_course.SPID
GROUP BY students.spid
```

7. List for each course in Computer Science that has more than four students: the name of the course, the highest grade, the lowest grade, and the average grade on the course.

```
SELECT course.name,
MAX(stud_course.grade) AS Highest,
MIN(stud_course.grade) AS Lowest,
AVG(stud_course.grade) AS Average
FROM course
JOIN department ON department.dcode = SUBSTR(course.ccode, 1, 2)
JOIN stud_course ON course.ccode = stud_course.CCODE
WHERE department.name = "Computer Science"
GROUP BY course.ccode
HAVING COUNT(*) > 4
```

8. List name and (numeric) grade for the students on the course "CS Introduction" that have received the grade 'C'.

```
SELECT students.sname, stud_course.grade
FROM stud_course
INNER JOIN course ON course.ccode = stud_course.ccode
INNER JOIN students ON students.spid = stud_course.spid
INNER JOIN grade ON grade.min <= stud_course.grade AND grade.max >=
stud_course.grade
WHERE course.name = 'CS INTRODUCTION'
AND grade.letter = 'C'
```

9. List for each course in Mathematics: the name of the course, the number of students and the teacher that does teach the course

```
SELECT course.name, teacher.forename, teacher.surname, COUNT(stud_course.SPID)
FROM course
JOIN department ON department.dcode = SUBSTRING(course.ccode,1,2)
JOIN stud_course ON stud_course.CCODE = course.ccode
JOIN teacher ON teacher.tpid = course.does_teach
WHERE department.name = "Mathematics"
GROUP By course.ccode
```

10.List for each student that has failed a course: the student's name, the name of the course and the grade

```
SELECT students.sname, course.name, stud_course.grade, grade.letter
FROM students
JOIN stud_course ON students.spid = stud_course.SPID
JOIN course ON stud_course.CCODE = course.ccode
JOIN grade ON grade.min <= stud_course.grade AND grade.max >= stud_course.grade
WHERE grade.letter = "F"
```

TRIGGERS

1. A trigger for UPDATE of the field grade: if the new value > 100 an error message should be printed

```
BEGIN
   IF NEW.grade > 100 THEN
         SIGNAL SQLSTATE '45000'
         SET MESSAGE_TEXT = "VALUE ABOVE MAX (100)";
   END IF;
END
```

2. A trigger for INSERT of a new course: if does_teach and responsible are not employed at the same department an error message should be printed

```
BEGIN
   DECLARE X INT;
        SELECT COUNT(*) INTO X
         FROM course
         INNER JOIN teacher tdt ON tdt.tpid = course.does_teach
         INNER JOIN teacher trs ON trs.tpid = course.responsible
         WHERE tdt.dept != trs.dept;

        IF (X > 0) THEN
         SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Department not matching for selected teacher';
        END IF;
END
```

3. A trigger for DELETE of a student: the deleted student should be added to the table Alumni_students (which you create in advance)

```
BEGIN
         INSERT INTO alumni_students VALUE(OLD.spid, OLD.sname,OLD.year,OLD.sup);
END
```