Practice Problem for the Bonus Part of Take Home Assignment 2

The table titled *Student* given below stores a set of attributes found in a University database. The following conditions apply:

- Each student in the *Student* table is in a program and can take any number of courses during the program.
- A course may be taken as part of any program and is always taught by a particular instructor.

You are required to show the table(s) in first, second, and third normal forms storing all the data currently stored in the following unnormalized *Student* table. In addition, show the relationships among the table(s) you create in each normal form.

Student

StudentID	FirstName	LastName	ProgramCode	ProgramLengthInMonths	CourseCode	CourseTitle	Instructor
000001	John	Doe	MSc	24	CPSC653	Computational Geometry	Jerry Miller
					CPSC689	Modelling For Computer Graphics	Dave Jones
					CPSC607	Biological Computation	John Stiles
000002	Richard	Miles	PhD	48	SENG697	Agent-Based Software Engineering	Ben Stuart
					CPSC607	Biological Computation	John Stiles
000003	Mary	Lange	MSc	24	CPSC653	Computational Geometry	Jerry Miller
000004	Jane	Roe	BSc	48	CPSC457	Principles of Operating Systems	Greg Brown
					CPSC653	Computational Geometry	Jerry Miller

This practice problem has been designed with concepts from the following two sources:

[1] P. Cherry, Central Queensland University: *Normalisation Example 2* (15 November, 2008).
http://webfuse.cqu.edu.au/Courses/aut2001/95169/Extra_Examples/Normalisation_Example_2/

[2] Microsoft Help and Support: *Description of the database normalization basics* (15 November, 2008).
http://support.microsoft.com/kb/283878/en-us

First Normal Form (1NF): Eliminate Repeating Groups

To get the table(s) in 1NF, we need to carry out the following steps:

- Eliminate repeating groups in the individual tables given.
- Create a separate table for each set of related attributes.
- Identify each set of related attributes with a primary key.

According to the given conditions in the problem specification and the data contained in the *Student* table, the attribute *StudentID* can be chosen as the primary key for the *Student* table. The attributes which repeat for each value of *StudentID* are *CourseCode*, *CourseTitle*, and *Instructor*. Removing these attributes of each student from the *Student* table following the steps listed above, we get the following two tables in 1NF.

Student1NF

StudentID	FirstName	LastName	ProgramCode	ProgramLengthInMonths
000001	John	Doe	MSc	24
000002	Richard	Miles	PhD	48
000003	Mary	Lange	MSc	24
000004	Jane	Roe	BSc	48

Registration1NF

StudentID	CourseCode	CourseTitle	Instructor
000001	CPSC653	Computational Geometry	Jerry Miller
000001	CPSC689	Modelling For Computer Graphics	Dave Jones
000001	CPSC607	Biological Computation	John Stiles
000002	SENG697	Agent-Based Software Engineering	Ben Stuart
000002	CPSC607	Biological Computation	John Stiles
000003	CPSC653	Computational Geometry	Jerry Miller
000004	CPSC457	Principles of Operating Systems	Greg Brown
000004	CPSC653	Computational Geometry	Jerry Miller

The primary key for the *Student1NF* table is *StudentID* because it uniquely identifies each record in that table. However, the table *Registration1NF* requires a composite primary key (*StudentID*, *CourseCode*) to uniquely identify each record contained in it. Now, there are no more repeating groups in the tables *Student1NF* and *Registration1NF*. So they are in first normal form. The relationship between these two tables in 1NF is shown in Fig. 1.

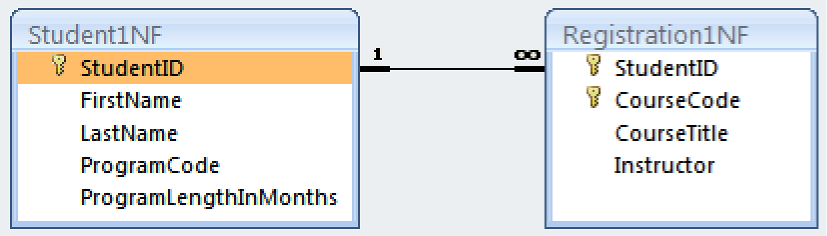

Fig. 1 Relationships among the tables in 1NF

Second Normal Form (2NF): Eliminate Redundant Data

To get the table(s) in 2NF, we need to eliminate partial dependencies from each table in 1NF so that the attributes depend only on the whole primary key. In order to do this, we need to carry out the following steps on each table in 1NF:

- Create a separate table for each set of related attributes that partially depends on the primary key.
- Relate these tables with foreign keys.

The table *Student1NF* does not have a composite primary key. Therefore, it does not contain any partial dependencies and is already in 2NF. The table *Registration1NF* has the functional dependencies shown in the following table.

Dependency	**Description**
(StudentID, CourseCode) → CourseTitle, Instructor	The primary key determines all attributes.
CourseCode → CourseTitle	Each course has a title.
CourseCode → Instructor	Each course is taught by a particular instructor.

Therefore, the attributes *CourseTitle* and *Instructor* are partially dependent on a part of the primary key, which is *CourseCode*. After the elimination of these partial dependencies following the steps listed above, we get the following three tables in 2NF.

Student2NF

StudentID	FirstName	LastName	ProgramCode	ProgramLengthInMonths
000001	John	Doe	MSc	24
000002	Richard	Miles	PhD	48
000003	Mary	Lange	MSc	24
000004	Jane	Roe	BSc	48

Registration2NF

StudentID	CourseCode
000001	CPSC653
000001	CPSC689
000001	CPSC607
000002	SENG697
000002	CPSC607
000003	CPSC653
000004	CPSC457
000004	CPSC653

Course2NF

CourseCode	CourseTitle	Instructor
CPSC457	Principles of Operating Systems	Greg Brown
CPSC607	Biological Computation	John Stiles
CPSC653	Computational Geometry	Jerry Miller
CPSC689	Modelling For Computer Graphics	Dave Jones
SENG697	Agent-Based Software Engineering	Ben Stuart

The relationships among these three tables in 2NF are shown in Fig. 2.

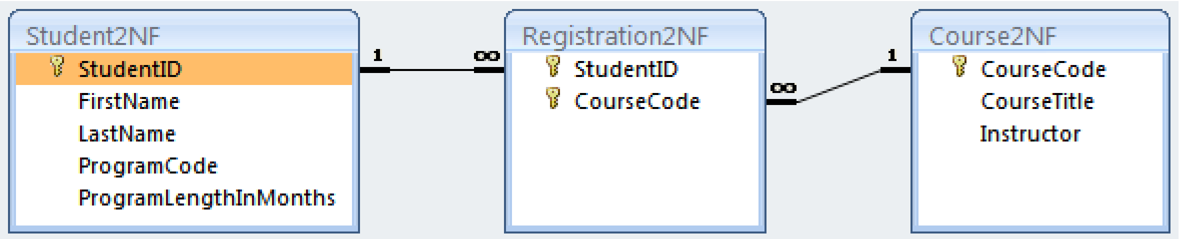

Fig. 2 Relationships among the tables in 2NF

Third Normal Form (3NF): Eliminate Transitive Dependencies

To get the table(s) in 3NF, we need to eliminate transitive dependencies from each table in 2NF so that the attributes depend on nothing but the primary key. In order to do this, we need to carry out the following steps on each table in 2NF:

- Create a separate table for each set of related attributes that transitively depends on the primary key.
- Relate these tables with foreign keys.

The table *Registration2NF* contains no non-key attributes. So it does not contain any transitive dependency and is already in 3NF. The table *Course2NF* contains the functional dependencies shown in the following table.

Dependency	Description
CourseCode → CourseTitle, Instructor	The primary key determines all attributes.

Therefore, there are no transitive dependencies in the table *Course2NF* and it is already in 3NF. The table *Student1NF* has the functional dependencies shown in the following table.

Dependency	Description
StudentID → FirstName, LastName, ProgramCode, ProgramLengthInMonths	The primary key determines all attributes.
ProgramCode → ProgramLengthInMonths	The program determines the program length.

Therefore, the following transitive dependency exists: *StudentID* → *ProgramCode* → *ProgramLengthInMonths*. After the elimination of these transitive dependencies following the steps listed above, we get the following four tables in 3NF.

Student3NF

StudentID	FirstName	LastName	ProgramCode
000001	John	Doe	MSc
000002	Richard	Miles	PhD
000003	Mary	Lange	MEng
000004	Jane	Roe	BSc

Program3NF

ProgramCode	ProgramLengthInMonths
BSc	48
MSc	24
PhD	48

Registration3NF

StudentID	CourseCode
000001	CPSC653
000001	CPSC689
000001	CPSC607
000002	SENG697
000002	CPSC607
000003	CPSC653
000004	CPSC457
000004	CPSC653

Course3NF

CourseCode	CourseTitle	Instructor
CPSC457	Principles of Operating Systems	Greg Brown
CPSC607	Biological Computation	John Stiles
CPSC653	Computational Geometry	Jerry Miller
CPSC689	Modelling For Computer Graphics	Dave Jones
SENG697	Agent-Based Software Engineering	Ben Stuart

The relationships among these four tables in 3NF are shown in Fig. 3.

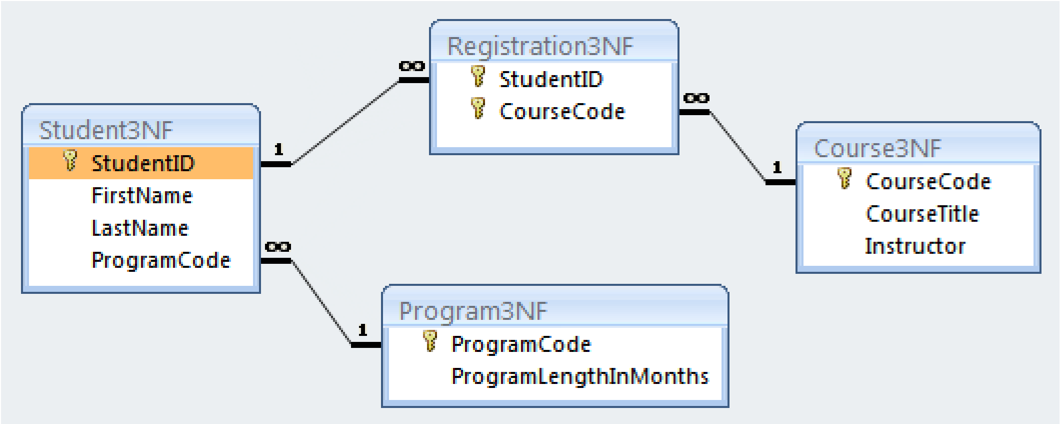

Fig. 3 Relationships among the tables in 3NF