

**Module Code & Module Title**

**CC6001NI - Advanced Database System Development**

**Assessment Weightage & Type**

**40% Individual Coursework**

**Year and Semester**

**2020-21 Autumn**

**Student Name: Suyogya Luitel**

**Group: C3**

**London Met ID: 19031784**

**College ID: NP01CP4A190035**

**Assignment Due Date: 09th March 2022**

**Assignment Submission Date: 09th March 2022**

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

# Introduction

This is a report of the coursework involving analysis, design, and implementation of a web-based database application in accordance with the given business case study with the help of tools such as Oracle SQL Developer Data Modeler, Oracle SQL Developer and ASP.NET. According to the presented business scenario, the database should contain information regarding college departments; students, including fee status and attendance; assignments, and their results. The scenario also elaborates that a teacher may be associated wit multiple modules and that a student may become a teacher after graduation.

# Normalization

## Figure 1: Example of Teacher allocation list

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.N. | Teacher Name | Address | Email | Module  Code | Module Name | Credit Hours |
| 1 | Saul Goodman | 595 Green Lake Road  Black Lake  9115 Lake Street Harrietsfield | Saulthegoodman@ abc.edu.np | CC12 | Data Structure and Algorithm | 30 |
| 2 | Walter White | 696 Madison St. Pierrefonds | [whitywalker@abc.e](mailto:whitywalker@abc.e) du.np | CC12 | Data Structure and Algorithm | 30 |
| 3 | Santana Lopez | 6 Valley View Street Griffintown | Santanalopez@abc. edu.np | CC49 | Engineering Thermodynamic | 60 |
| 4 | Rust Cohle | 89 Coffee Dr. Plaster Rock | [rustycohle@abc.ed](mailto:rustycohle@abc.ed) u.np | SG101 | Software engineer | 30 |
|  |  |  |  | TG405 | Data Analysis | 50 |

Table : Example of Teacher allocation list

While the given example table is very informative, it does not contain information regarding college departments that may associate with the teacher. Hence, additional fields "department id", and "department name" seem to be necessary addition.

Assumptions:

* A teacher is allowed to be in only one department at a time.
* A department can facilitate multiple teachers.

Applying these assumptions and including additional fields, the given example becomes as shown below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.N. | Teacher Name | Address | Email | Module  Code | Module Name | Credit Hours | Department id | Department name |
| 1 | Saul Goodman | 595 Green Lake Road  Black Lake  9115 Lake Street Harrietsfield | Saulthegoodman@ abc.edu.np | CC12 | Data Structure and Algorithm | 30 | 1 | Information Technology |
| 2 | Walter White | 696 Madison St. Pierrefonds | [whitywalker@abc.e](mailto:whitywalker@abc.e) du.np | CC12 | Data Structure and Algorithm | 30 | 1 | Information Technology |
| 3 | Santana Lopez | 6 Valley View Street Griffintown | Santanalopez@abc. edu.np | CC49 | Engineering Thermodynamic | 60 | 2 | Mechanical Engineering |
| 4 | Rust Cohle | 89 Coffee Dr. Plaster Rock | [rustycohle@abc.ed](mailto:rustycohle@abc.ed) u.np | SG101 | Software engineer | 30 | 3 | Computer Engineering |
|  |  |  |  | TG405 | Data Analysis | 50 |  |  |

Table : Addition of departments into given example.

Normalization of teacher allocation:

### UNF

Adding the above mention fields to the given Figure 1 fields, and identifying the repeating groups among them, we get the following UNF:

Teacher(Teacher\_ID, Teacher\_Name, {Address}, Email, {Module\_Code, Module\_Name, Credit\_Hours, Department\_ID, Department\_Name })

### 1NF

Separating the repeating groups identified in the UNF above and assigning composite keys, we get the following entities:

Teacher-1 (Teacher\_ID, Teacher\_Name, Email)

Address-1(Address\_ID, Teacher\_ID\*, Address)

Module-1(Module\_Code, Teacher\_ID\*, Module\_Name, Credit\_Hours, Department\_ID, Department\_Name)

### 2NF

After the repeating groups were separated into different entities and assigned new composite keys, they needed to be checked for partial dependencies.

For Teacher-1 :

* No composite keys were present and hence there were no partial dependencies.

Teacher\_ID → Teacher\_Name, Email

Teacher-2 (Teacher\_ID, Teacher\_Name, Email)

For Address-1:

* Address\_ID key determines Address
* Address\_ID, Teacher\_ID\* composite key determines nothing
* Teacher\_ID\* foreign key determines nothing

Address\_ID → (Address)

Address\_ID, Teacher\_ID\* → ()

Teacher\_ID → ()

The partial dependencies are separated into new entities as:

Address-2 (Address\_ID→ Address)

Address-Teacher-2 (Address\_ID, Teacher\_ID\*)

For Module-1:

* Module\_Code key determines Module\_Name, Credit\_Hours, Department\_ID, Department\_Name
* Module\_Code, Teacher\_ID\* composite key determines nothing

Module\_Code → (Module\_Name, Credit\_Hours, Department\_ID, Department\_Name)

Module\_Code, Teacher\_ID\* → ()

Module -2 (Module\_Code, Module\_Name, Credit\_Hours, Department\_ID, Department\_Name)

Module-Teacher -2 (Module\_Code, Teacher\_ID\*)

Hence, the results of 2NF are the following entities:

Teacher-2 (Teacher\_ID, Teacher\_Name, Email)

Address-2 (Address\_ID→ Address)

Address-Teacher-2 (Address\_ID, Teacher\_ID\*)

Module -2 (Module\_Code, Module\_Name, Credit\_Hours, Department\_ID, Department\_Name)

Module-Teacher -2 (Module\_Code, Teacher\_ID\*)

### 3NF

After checking and removing any partial dependencies, transitive dependencies are required to be checked and removed.

For Teacher-2 :

* Teacher\_ID determines Teacher\_Name and Email
* Teacher\_Name determines nothing
* Email determines nothing

There are no transitive dependencies.

Teacher-3 (Teacher\_ID, Teacher\_Name, Email)

For Address-2:

* Address\_ID determines Address
* Address determines nothing

There are no transitive dependencies.

Address-3 (Address\_ID, Address)

For Address-Teacher 2:

There are no transitive dependencies, the entity contains only a composite primary key.

Address-Teacher-3 (Address\_ID, Teacher\_ID\*)

For Module-Teacher 2:

There are no transitive dependencies, the entity contains only a composite primary key.

Module-Teacher-3 (Module\_ID, Teacher\_ID\*)

For Module-2:

* Module\_Code determines Module\_Name, Credit\_Hours, Department\_ID
* Module\_Name determines nothing
* Credit\_Hours determines nothing
* Department\_ID determines Department\_Name

Module\_Code → (Module\_Name, Credit\_Hours, Department\_ID)

Department\_ID → (Department\_Name)

The transitive dependencies are separated into entities as follows:

Module-3 → (Module\_Code, Module\_Name, Credit\_Hours, Department\_ID\*)

Department-3 → (Department\_ID, Department\_Name)

Hence, the results of 3NF are the following entities :

Teacher-3 (Teacher\_ID, Teacher\_Name, Email)

Address-3 (Address\_ID, Address)

Address-Teacher-3 (Address\_ID, Teacher\_ID\*)

Module-3 (Module\_Code, Module\_Name, Credit\_Hours, Department\_ID\*)

Module-Teacher-3 (Module\_ID, Teacher\_ID\*)

Department-3 (Department\_ID, Department\_Name)

### Final entities from figure 1

After normalizing figure 1 up to third normal form, the following entities are obtained:

Teacher (Teacher\_ID, Teacher\_Name, Email)

Address (Address\_ID, Address)

Address-Teacher (Address\_ID, Teacher\_ID\*)

Module (Module\_Code, Module\_Name, Credit\_Hours, Department\_ID\*)

Module-Teacher (Module\_ID, Teacher\_ID\*)

Department (Department\_ID, Department\_Name)

## Fig 2: Example of Assignment and Examination Results

Student ID: 149893

Student Name: Mr. William Ishee

Student Address: 2508 Shinn Street New York

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module  Code | Module Name | Assignment Type | Grade | Status |
| CC12 | Data Structure and Algorithm | Coursework | A | Pass |
| CC49 | Engineering Thermodynamic | Coursework | B | Pass |
| CC49 | Engineering Thermodynamic | Written Exam | F | Fail |
| SG101 | Software engineer | Individual  Assignment | B+ | Pass |
| SG101 | Software engineer | Group Assignment | B | Pass |
| SG101 | Software engineer | Unseen Examination | A | Pass |

While the given example is very informative, it does not contain information regarding semester, semester fees, and semester attendance. Additionally, student attendance as well as student fee payment information also seems missing. Hence, additional fields "semester", "semester fees", "semester attendance", "student attendance", and "Fee status" seem to be a necessary addition.

Assumptions:

* Attendance of students is recorded for each semester.
* Each semester has a certain maximum attendance record.
* Semester fees of the students are common regardless of the modules they study. This implies that students A and B studying module sets C and D have the same semester fees if they both are in semester E.

Applying these assumptions and including additional fields, the given example becomes as shown below:

St*udent ID: 149893*

*Student Name: Mr. William Ishee*

*Student Address: 2508 Shinn Street New York*

*Student Attendance: 69*

*Fee Status: Paid*

*Semester No. : 1*

*Semester Fees : 650*

*Semester attendance: 75*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module  Code | Module Name | Assignment Type | Grade | Status |
| CC12 | Data Structure and Algorithm | Coursework | A | Pass |
| CC49 | Engineering Thermodynamic | Coursework | B | Pass |
| CC49 | Engineering Thermodynamic | Written Exam | F | Fail |
| SG101 | Software engineer | Individual  Assignment | B+ | Pass |
| SG101 | Software engineer | Group Assignment | B | Pass |
| SG101 | Software engineer | Unseen Examination | A | Pass |

### UNF

Adding the above mention fields to the given Figure 2 fields, and identifying the repeating groups among them, we get the following UNF:

Student (Student\_ID, Sudent\_Name, Student\_Address, Semester\_No, Semester\_Fees Semester\_Attendance, Student\_Attendance, Fee\_Status, {Module\_Code, Module\_Name {Assignment\_Type, Grade, Status}})

### 1NF

Separating the repeating groups identified in the UNF above and assigning composite keys, we get the following entities:

Student-1(Student\_ID, Student\_Name, Student\_Address, Semester\_No, Semester\_Fees Semester\_Attendance, Student\_Attendance, Fee\_Status)

Module-1 (Module\_Code, Student\_ID\*, Module\_Name)

Assignment-1(Assignment\_ID, Student\_ID\*, Module\_Code\*, Assignment\_Type, Grade, Status)

### 2NF

After the repeating groups were separated into different entities and assigned new composite keys, they needed to be checked for partial dependencies.

For Student-1 :

* No composite keys were present and hence there were no partial dependencies.

Student\_ID → (Student\_Name, Student\_Address, Semester\_No, Semester\_Fees Semester\_Attendance, Student\_Attendance, Fee\_Status)

Student-2 (Student\_ID, Student\_Name, Student\_Address, Semester\_No, Semester\_Fees Semester\_Attendance, Student\_Attendance, Fee\_Status)

For Module-1:

* Module\_Code key determines Module\_Name
* Module\_Code, Student\_ID\* composite key determines nothing
* Student \_ID\* foreign key determines nothing

Student\_ID, Module\_Code → ()

Student\_ID → ()

Module\_Code → (Module\_Name)

Module-Student-2 (Student\_ID\*, Module\_Code\*)

Module-2(Module\_Code, Module\_Name)

For Assignment-1:

* Assignment\_ID key determines Assigment\_Type
* Assignment\_ID, Student\_ID\*, Module\_Code\* composite key determines Grade, Status
* Student \_ID\* foreign key determines nothing
* Module \_Code\* foreign key determines nothing

Assignment\_ID → (Assignment\_Type)

Assignment\_ID, Student\_ID, Module\_Code → (Grade, Status)

Student\_ID → ()

Module\_Code → ()

The partial dependencies are separated into new entities as:

Assignment-2 (Assignment\_ID, Assignment\_Type)

Student-Assignment-2(Assignment\_ID\*, Student\_ID\*, Module\_Code\*, Grade, Status)

Hence, the results of 2NF are the following entities:

Student-2 (Student\_ID, Student\_Name, Student\_Address, Semester\_No, Semester\_Fees Semester\_Attendance, Student\_Attendance, Fee\_Status)

Module-Student-2 (Student\_ID\*, Module\_Code\*)

Module-2(Module\_Code, Module\_Name)

Assignment-2 (Assignment\_ID, Assignment\_Type)

Student-Assignment-2(Assignment\_ID\*, Student\_ID\*, Module\_Code\*, Grade, Status)

### 3NF

After checking and removing any partial dependencies, transitive dependencies are required to be checked and removed.

For Student-2 :

* Student\_ID determines Student\_Name, Student\_Address, Fee\_Status, Student\_Attendance
* Student\_Name determines nothing
* Student\_Address determines nothing
* Fee\_Status determines nothing
* Student\_Attendance determines nothing
* Semester\_No determines Semester\_Fees, Semester\_Attendance
* Semester\_Fees determines nothing
* Semester\_Attendance determines nothing

Student\_ID → (Student\_Name, Student\_Address, Fee\_Status, Student\_Attendance)

Semester\_No → (Semester\_Fees, Semester\_Attendance)

The transitive dependencies are separated into entities as follows:

Student-3 (Student\_ID, Student\_Name, Student\_Address, Fee\_Status, Student\_Attendance, Semester\_ID\*)

Semester-3 (Semester\_No, Semester\_Fees, Semester\_Attendance)

For Module-Student-2:

There are no transitive dependencies, the entity contains only a composite primary key.

Module-Student-3 (Student\_ID\*, Module\_Code\*)

For Module-2:

* Module\_Code determines Module\_Name
* Module\_Name determines nothing

There are no transitive dependencies.

Module-3(Module\_Code, Module\_Name)

For Assignment-2:

* Assignment\_ID determines Assigment\_Type
* Assignment\_Type determines nothing
* Module\_Name determines nothing

There are no transitive dependencies.

Assignment-3 (Assignment\_ID, Assignment\_Type)

For Student-Assignment-2:

* Assignment\_ID, Student\_ID, Module\_Code composite key determines Grade
* Grade determines Status
* Status determines nothing

Assignment\_ID\*, Student\_ID\*, Module\_Code\* → Grade

Grade → Status

The transitive dependencies are separated as:

Student-Assignment-3 (Assignment\_ID\*, Student\_ID\*, Module\_Code\*, Grade\_ID\*)

Grade-3 (Grade\_ID, Grade, Status)

Hence, the results of 3NF are the following entities :

Student-3 (Student\_ID, Student\_Name, Student\_Address, Fee\_Status, Student\_Attendance, Semester\_ID\*)

Semester-3 (Semester\_No, Semester\_Fees, Semester\_Attendance)

Module-Student-3 (Student\_ID\*, Module\_Code\*)

Module-3(Module\_Code, Module\_Name)

Assignment-3 (Assignment\_ID, Assignment\_Type)

Student-Assignment-3 (Assignment\_ID\*, Student\_ID\*, Module\_Code\*, Grade\_ID\*)

Grade-3 (Grade\_ID, Grade, Status)

### Final entities from figure 2

After normalizing figure 2 up to third normal form, the following entities are obtained:

Student (Student\_ID, Student\_Name, Student\_Address, Fee\_Status, Student\_Attendance, Semester\_ID\*)

Semester (Semester\_No, Semester\_Fees, Semester\_Attendance)

Module-Student (Student\_ID\*, Module\_Code\*)

Module(Module\_Code, Module\_Name)

Assignment (Assignment\_ID, Assignment\_Type)

Student-Assignment (Assignment\_ID\*, Student\_ID\*, Module\_Code\*, Grade\_ID\*)

Grade (Grade\_ID, Grade, Status)

## Integration

Assumptions:

* A teacher can have multiple addresses but a student can only have one address.
* Grade is evaluated in a single alphabet optionally followed by a sign.

After normalizing figure 1 up to third normal form, the following entities are obtained:

Teacher (Teacher\_ID, Teacher\_Name, Email)

Address (Address\_ID, Address)

Address-Teacher (Address\_ID, Teacher\_ID\*)

Module (Module\_Code, Module\_Name, Credit\_Hours, Department\_ID\*)

Module-Teacher (Module\_ID, Teacher\_ID\*)

Department (Department\_ID, Department\_Name)

After normalizing figure 2 up to third normal form, the following entities are obtained:

Student (Student\_ID, Student\_Name, Student\_Address, Fee\_Status, Student\_Attendance, Semester\_ID\*)

Semester (Semester\_No, Semester\_Fees, Semester\_Attendance)

Module-Student (Student\_ID\*, Module\_Code\*)

Module(Module\_Code, Module\_Name)

Assignment (Assignment\_ID, Assignment\_Type)

Student-Assignment (Assignment\_ID\*, Student\_ID\*, Module\_Code\*, Grade\_ID\*)

Grade (Grade\_ID, Grade, Status)

Combining them into a single database, the following entities are obtained:

Teacher (Teacher\_ID, Teacher\_Name, Email)

Address (Address\_ID, Address)

Address-Teacher (Address\_ID, Teacher\_ID\*)

Module (Module\_Code, Module\_Name, Credit\_Hours, Department\_ID\*)

Module-Teacher (Module\_ID, Teacher\_ID\*)

Department (Department\_ID, Department\_Name)

Student (Student\_ID, Student\_Name, Student\_Address, Fee\_Status, Student\_Attendance, Semester\_ID\*)

Semester (Semester\_No, Semester\_Fees, Semester\_Attendance)

Module-Student (Student\_ID\*, Module\_Code\*)

Assignment (Assignment\_ID, Assignment\_Type)

Student-Assignment (Assignment\_ID\*, Student\_ID\*, Module\_Code\*, Grade\_ID\*)

Grade (Grade\_ID, Grade, Status)