**Project Part 1**

***Question 1***

Normalize the following initial design to 2NF :

VIDEO (Video-ID, Title, (Client-ID, Client-Name, Rented-Date, Returned-Date))

The initial design above, VIDEO, is not in 1FN because there is a repeating group of columns which are Client-ID, Client-Name, Rented-Date and Returned-Date.

1. 1st Normal Form

These following rules must be respected for the table to be in 1FN :

* Each column of the table must be single-valued.
* Each column should have a unique name.
* A column should contain the same type of values.
* The order of the data stored does not matter.

VIDEO1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Video-ID | Title | Client-ID | Client-Name | Rented-Date | Returned-Date |
| 20201 | Abc | 01 | Achan | 08/08 | 08/16 |
| 20202 | Def | 01 | Achan | 08/08 | 08/16 |
| 20203 | Ghi | 02 | Ichan | 09/09 | 09/18 |
| 20204 | Jkl | 03 | Ochan | 10/10 | 10/20 |
| 20205 | Mno | 01 | Achan | 11/11 | 11/22 |
| 20201 | Abc | 03 | Ochan | 11/11 | 11/22 |

The primary key is a composite key of Video-ID and Client-ID because they are in a many-to-many relationship : a video can be opted by more than one client, and a client can opt for more than one video.

Now we have the following design in 1FN :

VIDEO1 (Video-ID, Title, Client-ID, Client-Name, Rented-Date, Returned-Date)

The design above, VIDEO1, is not in 2FN because there are partial dependencies.

1. 2nd Normal Form

These following rules must be respected for the table to be in 2FN :

* The table should be in 1st Normal Form.
* The table should not have any partial dependencies.

By observing the table VIDEO1, we notice two partial dependencies :

* The Title only depends on Video-ID and not necessarily on Client-ID.
* The Client-Name only depends on Client-ID and not necessarily on Video-ID.

Rented-Date and Returned-Date are functionally dependent on the whole primary key (Video-ID + Client-ID) because many clients could have rented or returned a video on a same day (date) and many videos could have been rented or could have been returned by a client on a same day (date).

It is impossible to find a specific Rented-Date record or a specific Returned-Date record with Video-ID only or with Client-ID only, which means Rented-Date and Returned-Date are not partial dependencies.

We are to divide the table VIDEO1 in 3 new tables to remove the partial dependencies :

* Video2 (Video-ID, Title)
* Client2 (Client-ID, Client-Name)
* Date2 (Video-ID, Client-ID, Rented-Date, Returned-Date)

Video2 Client2

|  |  |
| --- | --- |
| Video-ID | Title |
| 20201 | Abc |
| 20202 | Def |
| 20203 | Ghi |
| 20204 | Jkl |
| 20205 | Mno |

|  |  |
| --- | --- |
| Client-ID | Client-Name |
| 01 | Achan |
| 02 | Ichan |
| 03 | Ochan |

Date2

|  |  |  |  |
| --- | --- | --- | --- |
| Video-ID | Client-ID | Rented-Date | Returned-Date |
| 20201 | 01 | 08/08 | 08/16 |
| 20202 | 01 | 08/08 | 08/16 |
| 20203 | 02 | 09/09 | 09/18 |
| 20204 | 03 | 10/10 | 10/20 |
| 20205 | 01 | 11/11 | 11/22 |
| 20201 | 03 | 11/11 | 11/22 |

PK in Video2 = Video-ID

PK in Client2 = Client-ID

PK in Date2 = Video-ID + Client-ID (composite key)

FK1 in Date2 = Video-ID

FK2 in Date2 = Client-ID

***Question 2***

Normalize the following initial design to 2NF :

STUDENT (StudID, StudName, (CourseNo, CourseName, GRADE))

The initial design above, STUDENT, is not in 1FN because there is a repeating group of columns which are CourseNo, CourseName and GRADE.

1. 1st Normal Form

These following rules must be respected for the table to be in 1FN :

* Each column of the table must be single-valued.
* Each column should have a unique name.
* A column should contain the same type of values.
* The order of the data stored does not matter.

STUDENT1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StudID | StudName | CourseNo | CourseName | GRADE |
| 01 | Saku | 703 | C++ | A |
| 01 | Saku | 702 | Maths | B |
| 02 | Gen | 702 | Maths | B |
| 03 | Sai | 701 | Java | B |

The primary key is a composite key of StudID and CourseNo because they are in a many-to-many relationship : a student can opt for more than one course, and a course can be opted by more than one student.

Now we have the following design in 1FN :

STUDENT1 (StudID, StudName, CourseNo, CourseName, GRADE)

The design above, STUDENT1, is not in 2FN because there are partial dependencies.

1. 2nd Normal Form

These following rules must be respected for the table to be in 2FN :

* The table should be in 1st Normal Form.
* The table should not have any partial dependencies.

By observing the table STUDENT1, we notice two partial dependencies :

* The StudName only depends on StudID and not necessarily on CourseNo.
* The CourseName only depends on CourseNo and not necessarily on StudID.

GRADE is functionally dependent on the whole primary key (StudID + CourseNo) because a student might achieve a different grade depending on the course, and a course’s grade can vary depending on the student that produced the result.

It is impossible to find a specific record of a student’s grade because we do not know which course we are to take in account with StudID only. If we were to find a specific record of a grade achieved in a course, we would not be able to know which student we are to take in account with CourseNo only. Therefore, GRADE is not a partial dependency.

We are to divide the table STUDENT1 in 3 new tables to remove the partial dependencies :

* Student2 (StudID, StudName)
* Course2 (CourseNo, CourseName)
* Grade2 (StudID, CourseNo, GRADE)

Student2 Course2

|  |  |
| --- | --- |
| StudID | StudName |
| 01 | Saku |
| 02 | Gen |
| 03 | Sai |

|  |  |
| --- | --- |
| CourseNo | CourseName |
| 701 | Java |
| 702 | Maths |
| 703 | C++ |

Grade2

|  |  |  |
| --- | --- | --- |
| StudID | CourseNo | GRADE |
| 01 | 703 | A |
| 01 | 702 | B |
| 02 | 702 | B |
| 03 | 701 | B |

PK in Student2 = StudID

PK in Course2 = CourseNo

PK in Grade2 = StudID + CourseNo (composite key)

FK1 in Grade2 = StudID

FK2 in Grade2 = CourseNo

***Question 3***

Normalize the following initial design to 2NF :

COURSE (CourseNo, CourseName, (StudID, StudName, GRADE))

The initial design above, COURSE, is not in 1FN because there is a repeating group of columns which are StudID, StudName and GRADE.

1. 1st Normal Form

These following rules must be respected for the table to be in 1FN :

* Each column of the table must be single-valued.
* Each column should have a unique name.
* A column should contain the same type of values.
* The order of the data stored does not matter.

COURSE1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CourseNo | CourseName | StudID | StudName | GRADE |
| 401 | C# | 01 | Li | A |
| 402 | Python | 02 | Liu | C |
| 402 | Python | 03 | Lai | A |
| 403 | P.E. | 03 | Lai | B |

The primary key is a composite key of CourseNo and StudID because they are in a many-to-many relationship : a course can be opted by more than one student, and a student can opt for more than one course.

Now we have the following design in 1FN :

COURSE1 (CourseNo, CourseName, StudID, StudName, GRADE)

The design above, COURSE1, is not in 2FN because there are partial dependencies.

1. 2nd Normal Form

These following rules must be respected for the table to be in 2FN :

* The table should be in 1st Normal Form.
* The table should not have any partial dependencies.

By observing the table COURSE1, we notice two partial dependencies :

* The CourseName only depends on CourseNo and not necessarily on StudID.
* The StudName only depends on StudID and not necessarily on CourseNo.

GRADE is functionally dependent on the whole primary key (StudID + CourseNo) because a student might achieve a different grade depending on the course, and a course’s grade can vary depending on the student that produced the result.

It is impossible to find a specific record of a student’s grade because we do not know which course we are to take in account with StudID only. If we were to find a specific record of a grade achieved in a course, we would not be able to know which student we are to take in account with CourseNo only. Therefore, GRADE is not a partial dependency.

We are to divide the table COURSE1 in 3 new tables to remove the partial dependencies :

* Course2 (CourseNo, CourseName)
* Student2 (StudID, StudName)
* Grade2 (CourseNo, StudID, GRADE)

Course2 Student2

|  |  |
| --- | --- |
| CourseNo | CourseName |
| 401 | C# |
| 402 | Python |
| 403 | P.E. |

|  |  |
| --- | --- |
| StudID | StudName |
| 01 | Li |
| 02 | Liu |
| 03 | Lai |

Grade2

|  |  |  |
| --- | --- | --- |
| CourseNo | StudID | GRADE |
| 401 | 01 | A |
| 402 | 02 | C |
| 402 | 03 | A |
| 403 | 03 | B |

PK in Course2 = CourseNo

PK in Student2 = StudID

PK in Grade2 = CourseNo + StudID (composite key)

FK1 in Grade2 = CourseNo

FK2 in Grade2 = StudID

***Question 4***

What is your conclusion from the result of question 2 and question 3?

We are able to identify the same primary key, the same foreign keys, and produce the same final design and the same entity relationship diagram (ERD) between question 2 and question 3. We have started with the following initial designs :

* STUDENT (StudID, StudName, (CourseNo, CourseName, GRADE))
* COURSE (CourseNo, CourseName, (StudID, StudName, GRADE))

We notice that despite the name between both tables (entities) is different, both contain the same set of attributes each. We also notice that each initial design has a different primary key identified. When we start normalizing the initial designs, we notice that both StudID and CourseNo makes together a composite primary key in question 2 and question 3 which ultimately leads us to a same or similar final design in both cases.

(An extra thought…)

For question 1 with the following initial design :

VIDEO (Video-ID, Title, (Client-ID, Client-Name, Rented-Date, Returned-Date))

I believe that we would arrive at a same or similar final design with the following initial design :

CLIENT (Client-ID, Client-Name, (Video-ID, Title, Rented-Date, Returned-Date))

Because both Video-ID and Client-ID makes together a composite primary key in both cases and we kept the same set of attributes for both initial designs.