Problem 3:

What is database normalization and how does it help with managing information stored in a

relational database table? State if the following table is normalized, and if not, convert it to

First, Second, and Third normal forms. Also provide an explanation of each of the forms and

what changes did you make to convert the original/previous table to the current state.Table

Description automatically generated

Database normalization is a process that simplifies the relational database and reduces data redundancy. It enhances the efficiency of stored data and manipulation. Also, it helps reduce the data aberrant when updating databases and makes it easier to maintain the database. Normalization totally has several steps:

1NF(First Normal Form):

1. Each table cell only has one value
2. Delete duplicate columns and rows

First Normal Form remove repeated columns or rows into separate rows and make sure each table cell only has one single value.

2NF(Second Normal Form):

1. Delete partial dependency and make the primary key a single column.

Second Normal Form removes partial dependency, the primary key is not composed of two or more attributes, and all the attributes are fully functionally dependent on the primary key.

3NF(Third Normal Form)

1. Delete the transitive functional dependency

Third Normal Form will delete transitive functional dependency, making all the attributes only have relations with the candidate key. All the attributes except the primary key are isolated with each other and not transitively dependent on the primary key.

Table

Description automatically generated

**1NF(First Normal Form):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Employee ID | Employee Name | Projects | Department ID | Location |
| 1 | Niranjan | Website | D-1 | Urbana |
| 1 | Niranjan | SIERRA | D-1 | Urbana |
| 2 | Ethan | SSET | D-2 | Champaign |
| 2 | Ethan | LUI | D-2 | Champaign |
| 3 | Fabian | LUI | D-2 | Champaign |
| 3 | Fabian | REF | D-2 | Champaign |

Because it has multiple values in the Projects column, for the First Normal Form, we need to separate them into single rows.

**2NF(Second Normal Form):**

**Employee**

|  |  |  |  |
| --- | --- | --- | --- |
| Employee ID | Employee Name | Department ID | Location |
| 1 | Niranjan | D-1 | Urbana |
| 2 | Ethan | D-2 | Champaign |
| 3 | Fabian | D-2 | Champaign |

**Project**

|  |  |
| --- | --- |
| Employee ID | Projects |
| 1 | Website |
| 1 | SIERRA |
| 2 | SSET |
| 2 | LUI |
| 3 | LUI |
| 3 | REF |

For the 1NF table, the primary key is a compositive key (can be Employee Name+Projects), and it exists partial functionally dependent. If we know the Employee Name, we will know the Location and Department ID. Therefore, it does not match the Second Normal Form. So we divided it into two tables. The first is the Employee table, which stores the employee information, and the second is the Project table, which stores project information. And for the Project table, Employee ID is the foreign key. It helps us connect to the Employee table. If we insert a new record to the Project table and the Employee ID does not exist in the Employee table, it will cause an error, this helps us maintain the referential integrity.

**3NF(Third Normal Form):**

**Employee**

|  |  |  |  |
| --- | --- | --- | --- |
| Employee ID | Employee Name | Department ID | Location |
| 1 | Niranjan | D-1 | Urbana |
| 2 | Ethan | D-2 | Champaign |
| 3 | Fabian | D-2 | Champaign |

The functional dependency:

Employee ID <- Employee Name, Department ID, Location

Employee Name <- Location

**Employee**

|  |  |  |  |
| --- | --- | --- | --- |
| Employee ID | Employee Name | Department ID | Location ID |
| 1 | Niranjan | D-1 | L-1 |
| 2 | Ethan | D-2 | L-2 |
| 3 | Fabian | D-2 | L-2 |

**Location**

|  |  |
| --- | --- |
| Location ID | Location |
| L-1 | Urbana |
| L-2 | Champaign |

For the 2NF table, the Employee table exists transitive functional dependency, the Employee ID is the primary key of the Employee table, but if we know the Employee Name, we also know the Location, which is transitive functional dependency, and it violates the 3NF. Therefore, we divided the original Employee table into Employee table and Location table to match the 3NF requirements. For the Location table, the Location ID is the foreign key, it connects to the Employee table.

Note:

In fact, the previous result I think still does not match the 3NF, because the Department ID and Location ID are function dependent with Employee Name. Another possible result is following:

**Employee**

|  |  |
| --- | --- |
| Employee ID | Employee Name |
| 1 | Niranjan |
| 2 | Ethan |
| 3 | Fabian |

**Department**

|  |  |  |
| --- | --- | --- |
| Employee ID | Department ID | Location |
| 1 | D-1 | Urbana |
| 2 | D-2 | Champaign |
| 3 | D-2 | Champaign |

The above result matches the 3NF requirement, not existing any transitive functional dependency, and the Employee ID is the foreign key and primary key for the Department table, it connects to the Employee table.