**ARINDA HILLARY**

**DBMS**

**ASSIGNMENT 3: Normalization**

**3/2/2024**

**1.**[5]**Consider a relation with following attributes:**

|  |  |
| --- | --- |
| **EmpNo** | **: Employee Number unique across all employees** |
| **EmpName** | **: Employee Name** |
| **EmpEmail** | **: Employee Email unique across all employees** |
| **ProjNo** | **: Project Number unique across all projects** |
| **ProjName** | **: Project Name** |
| **EmpGrade** | **: Employee Grade** |
| **HrlyRate** | **: Hourly rate of compensation.**Employees of the same grade receive the same hourly compensation. |
| **HrsWorked** | **: Hours a particular employee worked on a particular project.** |

* **EmpGrade is like “s/w developer”, “QA tester”, etc. One employee will have only one grade.**
* **An employee can work on many different projects and a project can have many different employees working on it.**

1. **Create this table and some sample data**(either on paper or in SQL Server)**. There must be at least 6 rows. There must be 2 to 4 Employees and 2 to 4 projects. Your submission must show all the rows from this table.**

A table with text on it

Description automatically generated

A table with text and images

Description automatically generated with medium confidence

1. **Find all Functional Dependencies.**

|  |
| --- |
| **Functional dependencies** |
| EmpNo->EmpName, EmpEmail,EmpGrade,HrlyRate |
| EmpEmail->EmpNo, EmpName, EmpGrade, HrlyRate |
| projNo->projName |
| EmpGrade->HrlyRate |
| (EmpNo, projNo)->HrsWorked |
| (EmpEmail, projNo)->HrsWorked |

1. **Find all Candidate Keys.**

|  |
| --- |
| (EmpNo, projNo)->HrsWorked |
| (EmpEmail, projNo)->HrsWorked |

1. **Find a Primary Key.**

(EmpNo, projNo)

1. **Find all partial dependencies.**

|  |
| --- |
| EmpNo -> EmpName, EmpEmail, EmpGrade,HrlyRate |
| EmpEmail->EmpNo, EmpName, EmpGrade, HrlyRate |
| projNo->projName |

1. **Normalize to 2NF.**

**Employee (**EmpNo, EmpName, EmpEmail,EmpGrade,HrlyRate)

**Project** (projNo, projName)

**EmployeeProjectHours (**EmpNo, projNo, HrsWorked)

1. **Show new tables**(based on the sample data you created in (a) above)**.**

A screenshot of a graph

Description automatically generated

1. **Normalize to 3NF.**

**Employee (**EmpNo, EmpName, EmpEmail,EmpGrade)

**Project** (projNo, projName)

**EmployeeProjectHours (**EmpNo, projNo, HrsWorked)

**Grade (**EmpGrade, HrlyRate)

1. **Show new tables**(based on the sample data you created in (a) above)**.**

A screenshot of a computer

Description automatically generated

1. **With the help of an example from this data, explain why a relation must be in 3NF.**

A relation should be in 3NF to:

* Reduce data redundancy and inconsistency: Normalization eliminates duplicate data and ensures that all data is stored in one place.
* Facilitate easier database maintenance: Updates, deletions, and insertions are simpler and less error-prone when data redundancy is minimized.

For example, if we need to update the hourly rate for a 'S/W Dev', in 3NF, we update it in a single place (the Grade table). Without 3NF, we would need to update multiple rows in the original table, increasing the risk of errors and inconsistencies.

**2.** [5]**Solve exercise 14.14 (a, b, c) on page 428 (476) from the course textbook (6th edition).**

 ﻿ Examine the Patient Medication Form for the Wellmeadows Hospital case study (see Appendix B) shown in

Figure 14.18.

1. Identify the functional dependencies represented by the attributes shown in the form in Figure 14.18. State any assumptions that you make about the data and the attributes shown in this form.

* Patient Number -> Full Name, Ward Number, Bed Number, Ward Name
* Drug Number -> Name, Description, Dosage, Method of Admin
* Patient Number, Drug Number, Start Date -> Units per Day, Finish Date

**Key Assumptions**

* Patient Information is Static per Admission:

A patient's number is unique and is associated with a static set of attributes for each admission (i.e., Full Name, Ward Number, Bed Number, Ward Name). This assumes that these details do not change during the patient's stay in the hospital.

* Drug Information is Consistent:

A drug's number is unique and each number corresponds to one set of drug attributes (i.e., Name, Description, Dosage, Method of Admin). This assumes the information about the drug doesn't vary and is consistent across all instances of its administration.

* Units per Day Can Vary for the Same Drug:

For the same patient and the same drug, the Units per Day can vary. This is evidenced by the different entries for the drug Morphine for the same patient PID0034, where the units per day change from 50 to 10.

* The Composite Key for Medication Records:

The combination of Patient Number, Drug Number, and Start Date is needed to uniquely identify a specific medication record. This assumes that a patient can start and finish the same drug multiple times with potentially different dosages (Units per Day), and that these instances do not overlap.

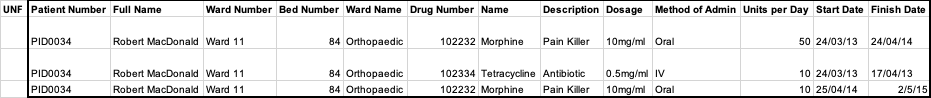
* Dates Indicate Specific Medication Periods:

The Start Date and Finish Date refer to the specific period during which a drug was administered at the specified Units per Day. The same drug can have different administration periods for the same patient with possibly different Units per Day.

(b) Describe and illustrate the process of normalizing the attributes shown in Figure 14.18 to produce a set of well-designed 3NF relations.

The process involves removing redundancy and ensuring that each table contains data pertaining to only a single concept. First lets begin with the unnormalized format as shown below

* **Unnormalized Data (UNF)**



The unnormalized table includes the following columns, with some data repeating:

Patient Number, Full Name, Ward Number, Bed Number, Ward Name, Drug Number, Name (of the drug), Description, Dosage, Method of Admin, Units per Day, Start Date, Finish Date

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

The table is already in 1NF because each attribute contains only atomic values, and there are no repeating groups.

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

To achieve 2NF, we must remove partial dependencies on the composite key. Since Full Name, Ward Number, Bed Number, and Ward Name depend only on Patient Number, and Name, Description, Dosage, and Method of Admin depend only on Drug Number, we'll separate these into different tables.

A table with numbers and a number on it

Description automatically generated

These are the tables in 2NF

* **Patient** (Patient Number ,Full Name, Ward Number, Bed Number, Ward Name)
* **Drug** (Drug Number, Name, Description, Dosage, Method of Admin)
* **Medication** (Patient Number, Drug Number, Start Date, Units per Day, Finish Date)
* **Third Normal Form(3NF)**

To advance to the Third Normal Form (3NF), we need to remove any transitive dependencies within these tables. Transitive dependency occurs when one non-key attribute depends on another non-key attribute.

The Patient table in 2NF contains Full Name, Ward Number, Bed Number, Ward Name which all depend on Patient Number. However, since Ward Number is likely to determine Bed Number and Ward Name, there is a transitive dependency in this table.

To convert the Patient table to 3NF:

Remove the transitive dependency by separating the Ward Number, Bed Number, Ward Name from the Patient table into a new Ward Assignment table.

The resulting 3NF tables would be:

A screenshot of a medical report

Description automatically generated

These are the tables in 3NF

* **Patient** (Patient Number ,Full Name, Ward Number, Bed Number)
* **Drug** (Drug Number, Name, Description, Dosage, Method of Admin)
* **Medication** (Patient Number, Drug Number, Start Date, Units per Day, Finish Date)
* **Ward (**Ward Number, Ward Name)

(c) Identify the primary, alternate, and foreign keys in your 3NF relations

A medical form with a number of medication

Description automatically generated with medium confidence

**3.** [Optional-5 bonus pts] **Solve exercise 14.15 (a, b, c) on page 429 from the course textbook (6th edition).**

**﻿**﻿ The table shown in Figure 14.19 lists sample dentist/patient appointment data. A patient is given an appointment at a specific time and date with a dentist located at a particular surgery. On each day of patient appointments, a dentist is allocated to a specific surgery for that day.

(a) The table shown in Figure 14.19 is susceptible to update anomalies. Provide examples of insertion, deletion, and update anomalies.

(b) Identify the functional dependencies represented by the attributes shown in the table of Figure 14.19. State any assumptions you make about the data and the attributes shown in this table.

(c) Describe and illustrate the process of normalizing the table shown in Figure 14.19 to 3NF relations. Identify the primary, alternate, and foreign keys in your 3NF relations.

A table with names and numbers

Description automatically generated

**a). Update Anomalies:**

The update anomalies are described as follows: -

**Insertion Anomaly:**

* If a new dentist needs to be added, but they haven't yet had any appointments, it's impossible to insert their record without violating the table's structure, as there would be no associated patient or appointment details.

**Deletion Anomaly:**

* If a patient cancels all their appointments, their record might be deleted, losing information about them.

**Update Anomaly:**

* If a dentist's staff number changes, it would need to be updated in every row where the dentist appears, potentially causing inconsistencies if the update is not done consistently.

**b). Functional Dependencies :**

Assumptions:

* StaffNo uniquely identifies a dentist.
* patNo uniquely identifies a patient.
* SugeryNo uniquely identifies a surgery.

The Functional Dependencies are:

* StaffNo -> dentistName (determines the dentist's name)
* patNo -> patName (determines the patient's name)
* (StaffNo, appointment date time) -> SugeryNo (combines dentist + appointment time to determine assigned surgery)

**c). Normalization to 3NF:**

To eliminate redundancy and ensure data integrity, we need to decompose the table into three separate relations:

* **Dentist:**
  + StaffNo (Primary Key, uniquely identifies a dentist)
  + DentistName
* **Patient:**
  + patNo (Primary Key, unique patient identifier)
  + patName
* **Appointment:**
  + StaffNo (Foreign Key, references Dentist)
  + patNo (Foreign Key, references Patient)
  + appointmentDateTime
  + SugeryNo (Foreign Key, references Sugery)

**Details about the tables above:**

* The original table combined information about dentists, patients, and appointments into a single entity.
* By separating the data into three tables:
  + **Dentist:** Captures dentist details, independent of specific appointments.
  + **Patient:** Stores patient information, independent of appointments.
  + **Appointment:** Links dentists, patients, and appointment details, referencing the other two tables through foreign keys.

**Keys:**

* **Dentist:**
  + Primary Key: StaffNo (uniquely identifies a dentist)
* **Patient:**
  + Primary Key: patNo (uniquely identifies a patient)
* **Appointment:**
  + Primary Key: (composite key) (StaffNo, appointmentDateTime) (combination of dentist and appointment time uniquely identifies an appointment)
  + Foreign Key: StaffNo (references Dentist.StaffNo)
  + Foreign Key: patNo (references Patient.patNo)