DATABASE MANAGEMENT FOR A DENTAL CLINIC

Database Design and Development – H.Dip in Software Development 2022

Edivagner Ribeiro - G00411275@gmit.ie

Project Description

The main objective of this project is the design and development of a database for the management of a small dental clinic designed to manage patients, schedules, treatments, billing, and payments. The project was based on the narrative of the dental clinic routine.

This database contains a total of eleven tables, nine routines and six views. The entire sequence of SQL scripts to implement the table, routines and queries is in the **SQL_Script_DBprojectG00411275.sql** file that accompanies this document. In this file you can find the appropriate comments for each function and comments for the relationship between the keys.

Project Requirements and Solutions

Patient Management - we considered the patient's medical record separate from the address record. Two or more patients can share the same address (e.g., families). We have three procedures for admitting new patients:

```
CALL NEW_PATIENT('JOAO', 'PEREIRA', '1987-07-16', '01 239 5971',
'084 017 2983', 'T81 B4L28X');
```

This procedure enters a new patient and checks if the address already exists using Eircode and assigns a foreign key. If there is no record of the address, the patientdetail table will have a new key with the other fields in NULL value. Then we must call the next procedure and complete the new patient entry with the address details.

```
CALL NEW_PATIENT_DETAIL('StreetAddress', 'City YY', ' abc 123');
```

We can enter the complete data of the new patient with a single procedure:

```
-- CALL NEW_PATIENT_ALL( First Name ,  Surname , DOB, PhoneHome,
CellPhone, StreetAddress, City, EirCode);
CALL NEW_PATIENT_ALL('joao', 'Yyyy3', NULL, NULL, NULL, NULL,
NULL, 'DEF1-456');
```

Appointments - A patient can request an appointment. The entry of a new appointment depends on the availability of time slots. We assume that appointments are 30 minutes apart -- starting at 9:30 am to 12:00 and from 14:30 to 17:00 The CHECK constraint is used to limit the value range that can be placed in a column TimeVisit.

```
-- To consult the day's agenda, use the following command. 2022-04-26
CALL CHECK_DIARY(20220426);
SELECT * FROM diary;
-- The following procedure checks the appointments within 7 days of
-- the input date, so we can check a week's schedule with a single
command line.
CALL WEEKS_APPOINTMENTS(20220426);
SELECT * FROM WeekDiary;
-- CALL SET_NEW_APPOINTMENT(Patient_ID, Day, Month, Hour, Minute);
CALL SET_NEW_APPOINTMENT(115, 05, 05, 10, 30);
```

An appointment schedule can be deleted from the table, if it does not have any treatment registered, as this violates the integrity of the data. The procedure to delete is as follows (the procedure needing only the appointment ID):

```
-- late cancellations are charged a €10 late cancellation fee.
-- We will assume that appointments canceled on the same day will be
charged.
-- Only appointment with value NULL on status can be deleted
-- this prevents a patient from being billed twice, or a patient
-- who has already had the treatment performed from being charged
-- for a cancellation of a treatment already performed.
CALL DELETE_APPOINTMENTS(19);
```

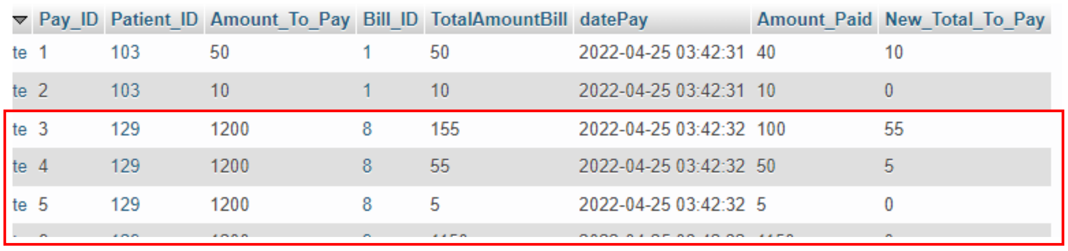
As for the treatment record, it needs the specialty ID (0 if it is in-clinic treatment), appointment ID, three treatment code entries and the last field for comments. If the record is for only one follow-up per external specialty, the first field is the specialty that the patient needs, and the treatment fields must have a value of ZERO.

```
-- ********* RECORD TREATMENT FOR TWO NEW PATIENTS
-- CALL TREATMENT_RECORD(Speciality_ID, Appointment_ID, Treat_ID,
Treat_ID, Treat_ID, 'RECORD ');
CALL TREATMENT_RECORD(0, 23, 12009, 12013, 0, 'RECORD TEST...');
CALL TREATMENT_RECORD(6, 24, 0, 0, 0, 'RECORD on specialist ...');
```

When we register a treatment, or a cancellation within 24 hours, both situations generate an invoice. To execute a payment, we use the following command.

```
-- ********* MAKE PAYMENT
-- CALL MAKE_PAYMENT(Bill_ID, Patient_ID, Amount to pay);
CALL MAKE_PAYMENT(8, 129, 100);
```

This payment registration process seeks to fulfil the narrative: " Patients often arrange to make several small payments for a large bill." You can see this in the payment table.



▼	Pay_ID	Patient_ID	Amount_To_Pay	Bill_ID	TotalAmountBill	datePay	Amount_Paid	New_Total_To_Pay
te	1	103	50	1	50	2022-04-25 03:42:31	40	10
te	2	103	10	1	10	2022-04-25 03:42:31	10	0
te	3	129	1200	8	155	2022-04-25 03:42:32	100	55
te	4	129	1200	8	55	2022-04-25 03:42:32	50	5
te	5	129	1200	8	5	2022-04-25 03:42:32	5	0

Conclusion

In this project, we meet the requirements of the proposed case through the implementation of procedures. However, due to the robustness of the database structure and the support of the SQL language, the INSERT, UPDATE, DELETE commands can be used for the same purpose. However, manual data entry makes the error possible.

The approach of procedures and functions guarantee the routine handling of data and facilitate the work of the user.