# Database Design and Development Database Project

## *A Dental Surgery Database-Relational Approach*

## 1. Database Design

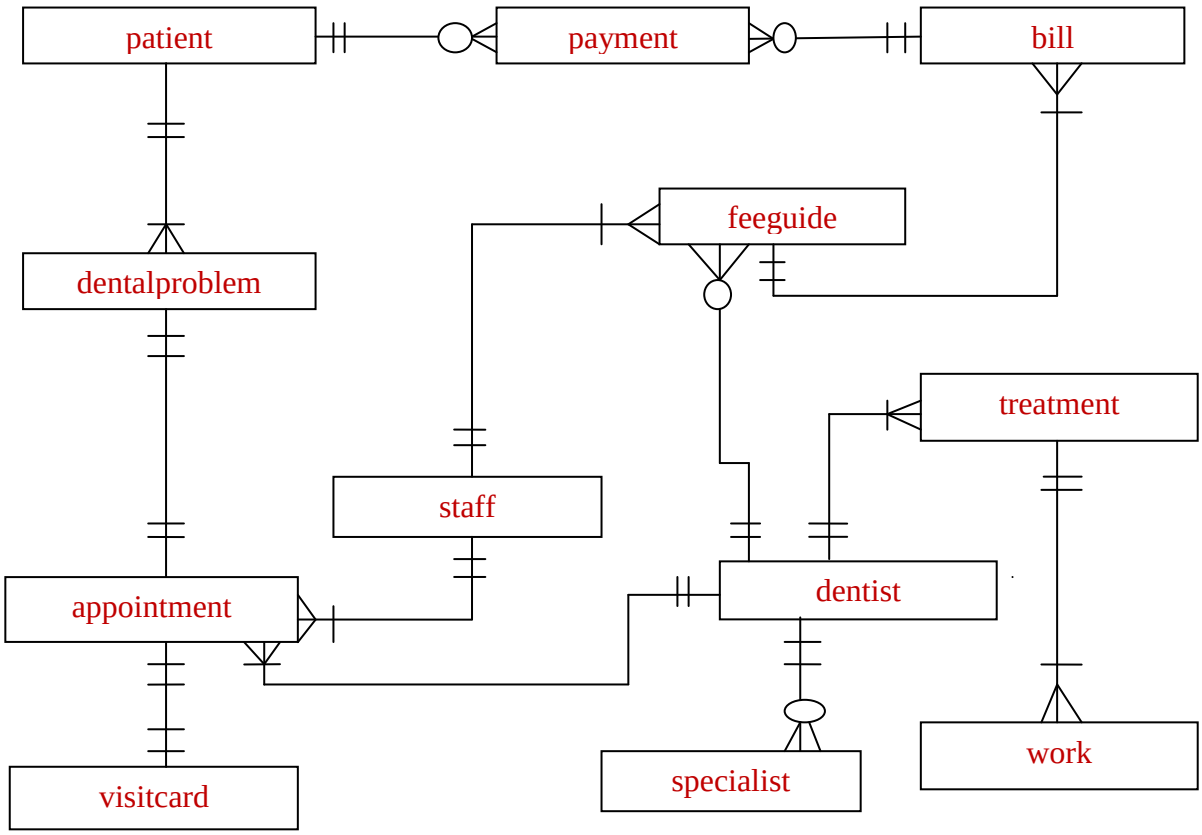
A schema is an overall representation of the database. The three levels in the design of a Relational Database are:

- Conceptual/Semantic Design- Entity Relationship Design ( ERD)
- Logical  Modelling ( Relational Model)
- Physical Design ( Database Tables)

### 1.1 Conceptual Design (ERD)

Entities and their relationships are identified at this level. It is an abstract form of logical model.

The entities, their relationships and cardinalities of the required ***dentalSurgery*** are identified and the ERD can be represented using Crow Foot notation as shown below.



The entities identified are:

- patient - used to store patients data
- dentalproblem – stores history of patents dental problems
- appointment – stores appointment information
- visitcard – stores information of patients each appointment
- staff – stores staff data
- dentist - details of dentist
- treatment - stores information regarding treatments carried out
- work- details of works (processes) involved in a treatment
- feeguide – data about fees charged on different procedures
- specialist – stores data regarding specialists
- bill – invoice information
- payment- information of payments made by customers on bill

There exists one to one, one to many, and many to many relationships between entities. It is possible to have cardinalities (number of rows) such as one and only one, one or many, zero or one or many, and zero or one in relations. The possible relationships and cardinalities of the ***dentalSurgery*** Database are noted as follows:

- A patient can have one or many dental problems; one to (one or many)
- A patient with a dental problem can have one appointment at a time and a visit card for an appointment; one to (one and only one)
- A patient makes zero or one or many payments; one to (zero or one or many)
- The dentist can take one or many appointments; one to (one or many)
- The dentist may update zero or one or more feeguide; one to (zero or one or many)
- The dentist can do one or many treatments; one to (one or many)
- The dentist can send zero or one or many referrals to specialists; one to (zero or one or many)
- There involves one or many works in a treatment; one or many to (one and only one)
- The staff issues one or many appointments; one to (one or many)
- The staff issues one or many bills based on the feeguide; one to (one or many)
- A bill may have zero or one or many payments; one to (zero or one or many)

## 1.2 Logical Model/Relational Model

It forms the basis of the physical model. At this level the business needs and data structure are decided, and is important in database designing.

Detailed structure of the entities and their relationships are outlined at this level based on the findings from the conceptual level. Attributes (including keys) and data types (platform independent) for each entity are identified.

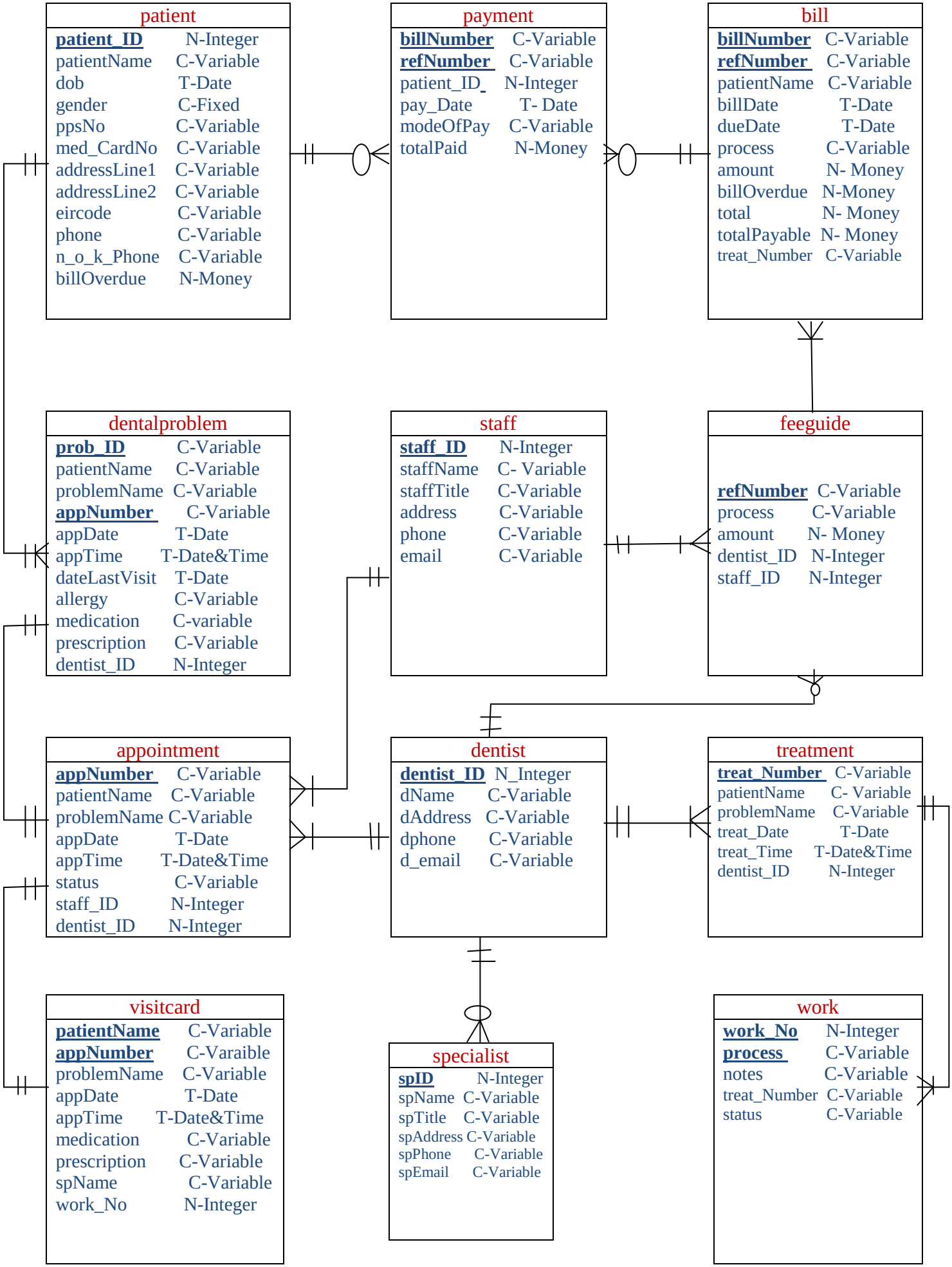
The attributes which are bold and underlined represents the **primary key** for each entity in the diagram below.

The **foreign keys** are:

| Entity | Foreign Key |
|---|---|
| **dentalproblem** | **dentist_ID** |
| **appointment** | **staff_ID,dentist_ID** |
| **visitcard** | **work_No** |
| **treatment** | **dentist_ID** |
| **work** | **treat_Number** |
| **payment** | **patient_ID** |
| **feeguide** | **staff_ID,dentist_ID** |

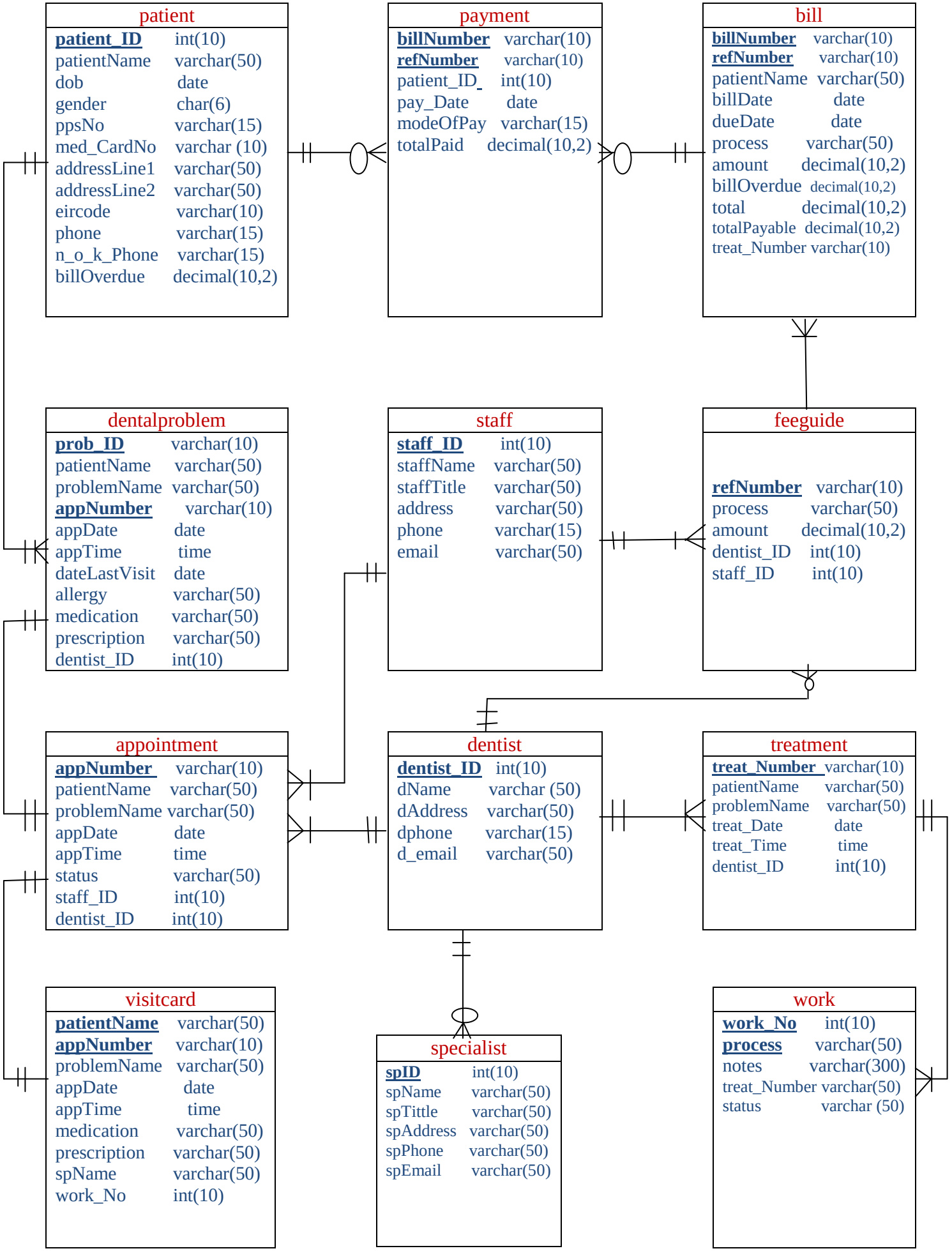
As been stated before this model is platform independent, and can be implemented in any database of choice (ex: ORACLE, SQL SERVER etc).

At logical level the relationship between the entities of ***dentalSurgery*** Database are shown in the following diagram.


patient
patient_ID N-Integer
patientName C-Variable
dob T-Date
gender C-Fixed
ppsNo C-Variable
med_CardNo C-Variable
addressLine1 C-Variable
addressLine2 C-Variable
eircode C-Variable
phone C-Variable
n_o_k_Phone C-Variable
billOverdue N-Money
payment
billNumber C-Variable
refNumber C-Variable
patient_ID N-Integer
pay_Date T- Date
modeOfPay C-Variable
totalPaid N-Money
bill
billNumber C-Variable
refNumber C-Variable
patientName C-Variable
billDate T-Date
dueDate T-Date
process C-Variable
amount N- Money
billOverdue N-Money
total N- Money
totalPayable N- Money
treat_Number C-Variable
dentalproblem
prob_ID C-Variable
patientName C-Variable
problemName C-Variable
appNumber C-Variable
appDate T-Date
appTime T-Date&Time
dateLastVisit T-Date
allergy C-Variable
medication C-variable
prescription C-Variable
dentist_ID N-Integer
staff
staff_ID N-Integer
staffName C- Variable
staffTitle C-Variable
address C-Variable
phone C-Variable
email C-Variable
feeguide
refNumber C-Variable
process C-Variable
amount N- Money
dentist_ID N-Integer
staff_ID N-Integer
appointment
appNumber C-Variable
patientName C-Variable
problemName C-Variable
appDate T-Date
appTime T-Date&Time
status C-Variable
staff_ID N-Integer
dentist_ID N-Integer
dentist
dentist_ID N_Integer
dName C-Variable
dAddress C-Variable
dphone C-Variable
d_email C-Variable
treatment
treat_Number C-Variable
patientName C- Variable
problemName C-Variable
treat_Date T-Date
treat_Time T-Date&Time
dentist_ID N-Integer
visitcard
patientName C-Variable
appNumber C-Varaible
problemName C-Variable
appDate T-Date
appTime T-Date&Time
medication C-Variable
prescription C-Variable
spName C-Variable
work_No N-Integer
specialist
spID N-Integer
spName C-Variable
spTitle C-Variable
spAddress C-Variable
spPhone C-Variable
spEmail C-Variable
work
work_No N-Integer
process C-Variable
notes C-Variable
treat_Number C-Variable
status C-Variable

## 1.3 Physical Model (MYSQL)

It is the visualisation of the actual database schema based on a specific platform (MYSQL in this case). Details such as table name, column name, and data types (column) are finalised at this stage. The physical model of the ***dentalSurgery*** Database (MYSQL) is shown in the following page.


patient
patient_ID int(10)
patientName varchar(50)
dob date
gender char(6)
ppsNo varchar(15)
med_CardNo varchar (10)
addressLine1 varchar(50)
addressLine2 varchar(50)
eircode varchar(10)
phone varchar(15)
n_o_k_Phone varchar(15)
billOverdue decimal(10,2)
payment
billNumber varchar(10)
refNumber varchar(10)
patient_ID int(10)
pay_Date date
modeOfPay varchar(15)
totalPaid decimal(10,2)
bill
billNumber varchar(10)
refNumber varchar(10)
patientName varchar(50)
billDate date
dueDate date
process varchar(50)
amount decimal(10,2)
billOverdue decimal(10,2)
total decimal(10,2)
totalPayable decimal(10,2)
treat_Number varchar(10)
dentalproblem
prob_ID varchar(10)
patientName varchar(50)
problemName varchar(50)
appNumber varchar(10)
appDate date
appTime time
dateLastVisit date
allergy varchar(50)
medication varchar(50)
prescription varchar(50)
dentist_ID int(10)
staff
staff_ID int(10)
staffName varchar(50)
staffTitle varchar(50)
address varchar(50)
phone varchar(15)
email varchar(50)
feeguide
refNumber varchar(10)
process varchar(50)
amount decimal(10,2)
dentist_ID int(10)
staff_ID int(10)
appointment
appNumber varchar(10)
patientName varchar(50)
problemName varchar(50)
appDate date
appTime time
status varchar(50)
staff_ID int(10)
dentist_ID int(10)
dentist
dentist_ID int(10)
dName varchar (50)
dAddress varchar(50)
dphone varchar(15)
d_email varchar(50)
treatment
treat_Number varchar(10)
patientName varchar(50)
problemName varchar(50)
treat_Date date
treat_Time time
dentist_ID int(10)
visitcard
patientName varchar(50)
appNumber varchar(10)
problemName varchar(50)
appDate date
appTime time
medication varchar(50)
prescription varchar(50)
spName varchar(50)
work_No int(10)
specialist
spID int(10)
spName varchar(50)
spTittle varchar(50)
spAddress varchar(50)
spPhone varchar(50)
spEmail varchar(50)
work
work_No int(10)
process varchar(50)
notes varchar(300)
treat_Number varchar(50)
status varchar (50)

## 1.4 Constraints

The application of **NOT NULL** constraints for required ***dentalSurgery*** Database is as follows:

- For the table **patient** all columns except ***med_CardNo***, and ***addressLine2*** are NOT NULL.
- For the table **dentalproblem** all columns except ***dateLastVisit***, ***allergy***, ***medication***, and ***prescription*** are NOT NULL.
- All columns of **appointment** are NOT NULL.
- For **visitcard** all columns except ***medication***, ***prescription, spName*** and ***work_No*** are NOT NULL.
- treat_Number in bill can be NULL, (as per the given information late cancellation is charged 10).
- For the remaining tables all columns set to be NOT NULL.

It is not necessary that all patients have medical card. The **DEFAULT** constraint is used to set the values of column ***med_CardNO*** to NULL. The same case is applied to columns mentioned in above paragraph where NOT NULL constraint is not used.

All **PRIMARY KEYS** must have UNIQUE and NOT NULL values (should satisfy **entity integrity)**, and there must be only one PRIMARY KEY per table, but there may be more than one column in a primary key. A PRIMARY KEY constraint automatically has a UNIQUE constraint.

**Referential integrity** applies to **FOREIGN KEYS.** The values of a foreign key should match a **CANDIDATE KEY** value of some row in the parent relation, otherwise it must be NULL.

# References


- https://www.codeproject.com/Articles/878359/Data-Modelling-using-ERD-with-Crow-Foot-Notation
- https://beginnersbook.com/2015/04/constraints-in-dbms/
- https://www.w3schools.com/sql/sql_constraints.asp
- https://www.w3schools.com/sql/sql_foreignkey.asp