

Database Design and Development

Database Project

A Dental Surgery Database-Relational Approach

1. Database Design

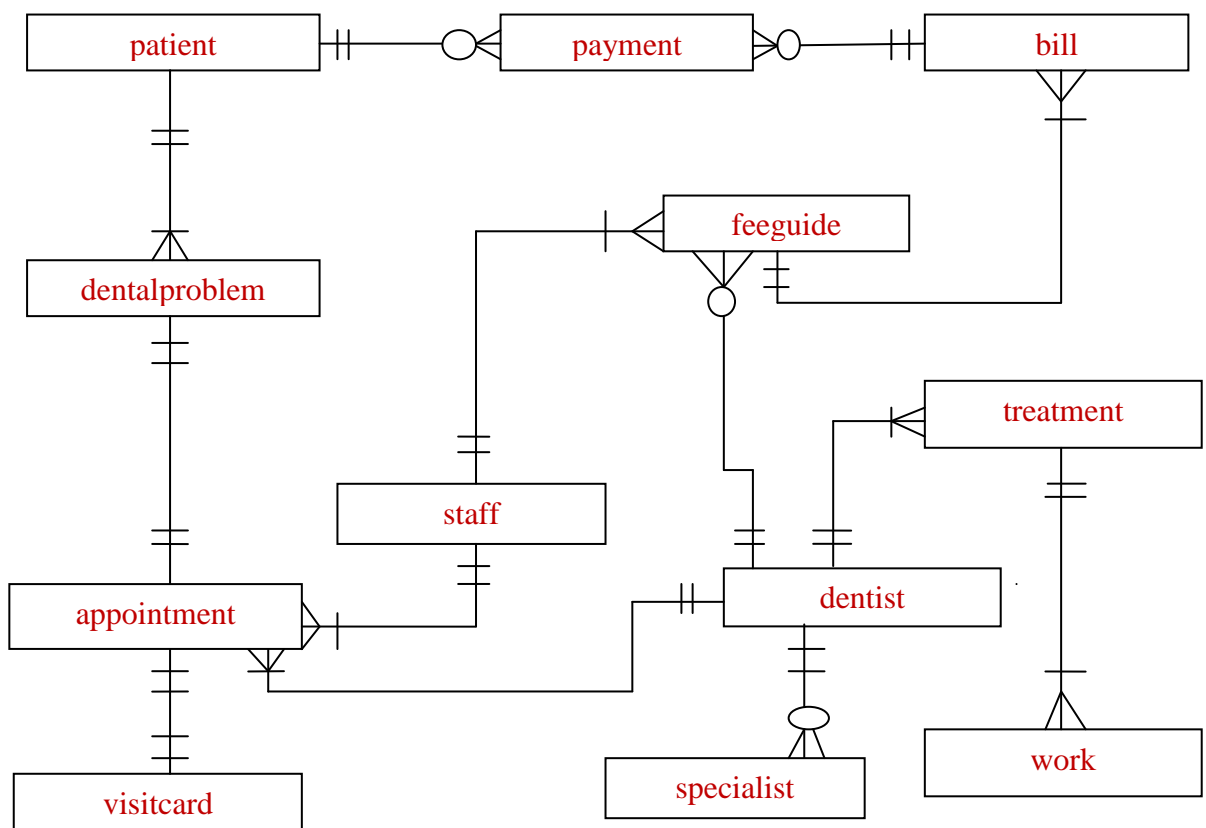
A schema is an overall representation of the database. The three levels in the design of a Relational Database are:

- 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 patients 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.

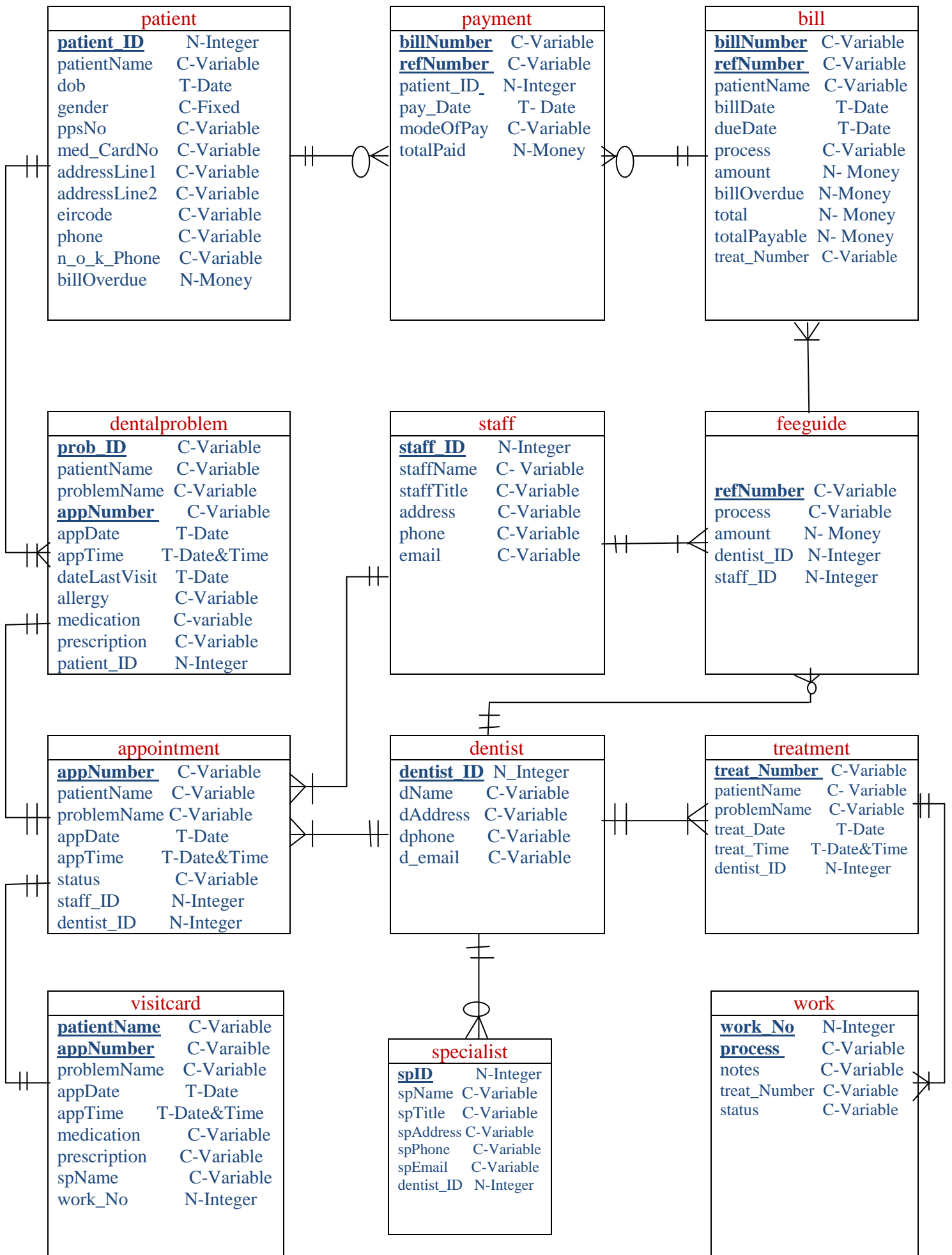
The attributes which are bold and underlined represents the **primary key** for each entity in the diagram below.

The **foreign keys** are:

Entity	Foreign Key
dentalproblem	appNumber, patient_ID
appointment	staff_ID, dentist_ID
visitcard	appNumber
specialist	dentist_ID
treatment	dentist_ID
work	treat_Number
bill	refNumber
payment	patient_ID, billNumber
feeguide	staff_ID, dentist_ID

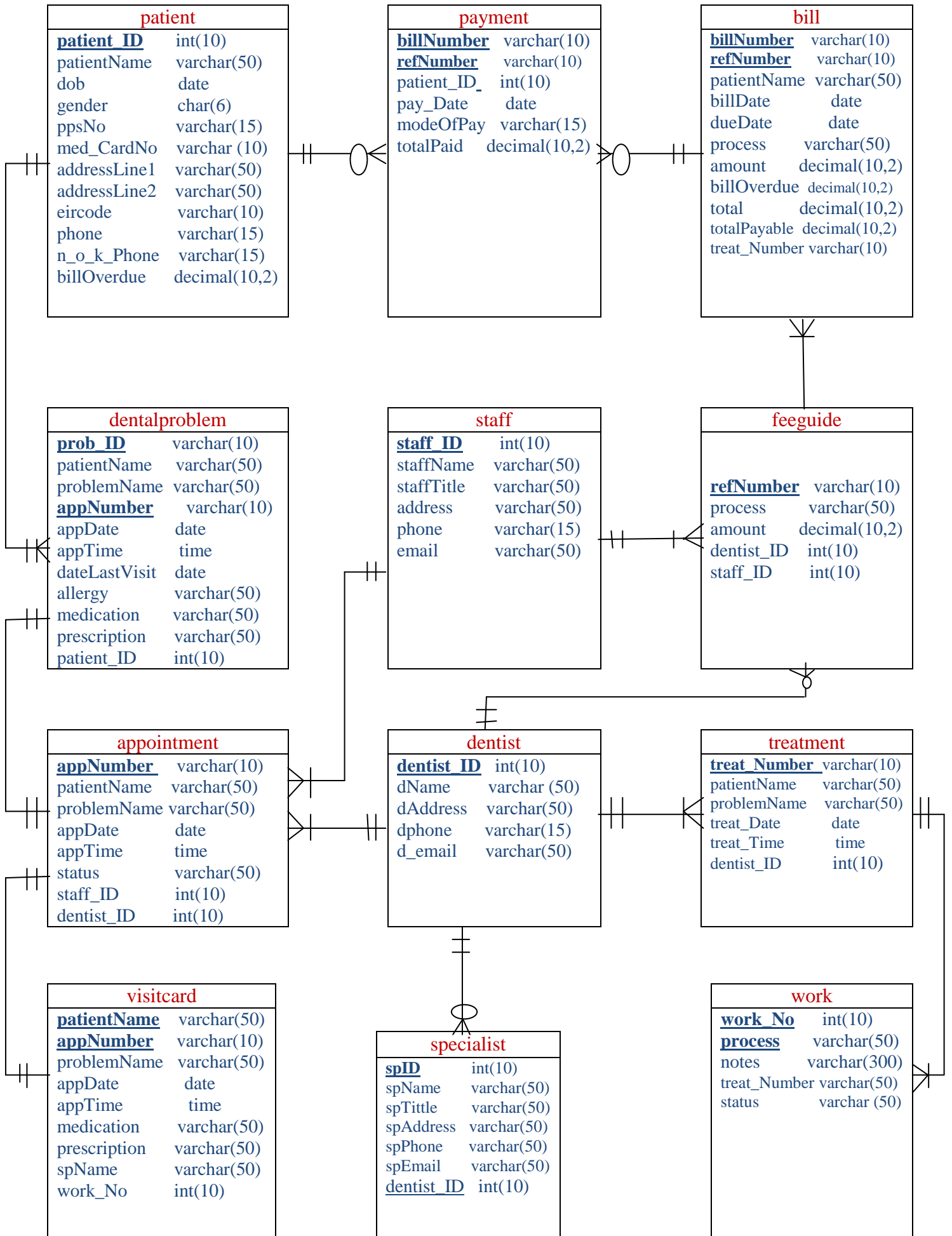
As been stated before this model is platform independent, and can be implemented in any database of choice (ex: ORACLE, SQL SERVER etc).

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



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.



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