Codd's Rules

Rule 1. The information Rule.

All the information in the database is represented one way i.e. values in column positions within rows of tables. For example, Helen can update the database by inserting a new patient :
insert into patient (id, name, surname, address, email, phone) values ('109', 'John', 'Martin', 'Shady Grove, Castletownbere, Co. Cork', 'jMartin@google.com', '09766434163');

Rule 2. The Guaranteed Access Rule.

Every entry i.e. information that was entered into the data base can be found using a table name, primary key of the row and the column name. For example for Helen to look up the contact details of a particular specialist she can use :

select * from specialist where id = '13';

Rule 3. The Systematic Treatment of Null Values.

This database adheres to rule three of Codd's rules by treating missing data or unknown data as null as opposed to zero. An example of this is :

```
select bill.id  , appointment.id, appointment.patientId as appointmentId,
payment.id as paymentId, bill.date as billDate, bill.appointmentId, bill.amount
as billAmount , payment.amount as paymentAmount, payment.date,
(select sum(payment.amount) from payment where payment.billId = bill.id)
as totalPayments from bill right join payment on bill.id = payment.billId
right join appointment on bill.appointmentId = appointment.id
right join patient on patient.id = appointment.patientId
```

-- When queried, some information is shown as null such as payment amount and payment date, because the appointment is in the future and thus no bill or payment has occurred for this appointment/ patient yet.

Rule 4. Dynamic online catalogbased on the relational model.

In compliance with rule four, dml commands can be used to modify rows in catalog tables and to create or destroy catalog elements. For example Helen can delete or update an appointment :

delete from appointments where id = '11015' and patientId = 103;

Rule 5. The comprehensive data sub language rule.

This database adheres to rule five with the use of sql. The data can be accessed, manipulated and have maintained consistency/ integrity through the use of sql. Queries also make sure transactions are either fully completed or not done. For example :

Helen can use sql in order to update the database e.g.
Alter table dentists drop column specialisation;

Helen can create a view to summarise patient and appointment data e.g : CREATE VIEW a ppointmentDiary AS SELECT appointment.id, appointment.time, appointment.status, appointment.duration, appointment.reason,patient.email, patient.phone, patient.firstName, patient.surname, appointment.date from patient left JOIN appointment on patient.id = appointment.patientId ;

Helen can update the database with new specialist details e.g.

insert into specialisation (dentistId, specialisation) values ('20', 'general practice');

Rule 6. The view updating rule.

This database adheres to this rule by having updatable views. Helen can therefore update the relevant information in the appointment diary.

update appointmentDiary set date = '2020-06-14' where id = 10412;

Rule 7. High level insert, update and delete.

Helen can update the database by only using one command for instance to update the patients table by inserting multiple new patients at once i.e. a family or multiple appointments at once. e.g. ;

```
INSERT INTO appointment (`patientId`, `date`, `time`, `reason`, `duration`, `status`, `id`,
`dentistId`) VALUES
(101, '2020-06-13', '13:00:00', 'checkup', '00:15:00', 'scheduled', 10113, 11),
(102, '2020-06-12', '10:30:00', 'filling', '00:20:00', 'scheduled', 10212, 11),
(103, '2020-05-12', '12:45:00', 'oral cleaning', '00:30:00', 'late cancellation', 10312, 11),
(104, '2020-06-12', '14:30:00', 'filling', '00:20:00', 'scheduled', 10412, 11),
(105, '2020-06-13', '09:45:00', 'cleaning', '00:30:00', 'cancelled', 10513, 11);
```

Rule 8. Physical data independence.

Sql is used to manipulate / define the data at a logical level. For instance if a file name is changed in the memory, this will not affect any tables or Helen's view of the table.

Rule 9. Logical Data Independence.

If Helen needs to alter a table , a select statement will still run as written.

i.e. if she adds a row or column to the appointment table the select * from appointment; will still return the contents of the table.

Rule 10. Integrity Independence.

This database complies to this rule by having primary keys with non null values. This is done in order to maintain data integrity as data integrity in a relational database is based on keys. All forgeign keys in this database reference individual primary keys.

For instance, if Helen needs to compare the payment amounts against their respective bills she can use joins e.g.

```
select bill.id as billId , concat(patient.name,' ', patient.surname) as patient,
appointment.date as appointmentDate, patient.email as patientEmail,
payment.receiptNumber as paymentReceipt, bill.date as billDate, appointment.reason as
reason, bill.amount
as billAmount , payment.amount as paymentAmount, payment.date as paymentDate,
```

```
(select sum(payment.amount) from payment where payment.billId = bill.id)
as totalPayments
from bill
right join payment on bill.id = payment.billId
right join appointment on bill.appointmentId = appointment.id
right join patient on patient.id = appointment.patientId
where bill.id is not null;
```

Rule 11. Distribution Independence.

The users must not be able to see that the data is distributed over various locations.
This database uses sql and therefore has no syntax that locates where the data is physically stored.

Rule 12. The Non Subversion rule.

This database requires all users to use a high level language i.e. sql for all database structural modifications. Only Helen and Dr. Mulcahy have access rights and can use low level languages if need be.