Rule 1: **The information rule**

All informational in a relational database is represented explicitly at the logical level in exactly one way, by values in tables.

```
SELECT * FROM patient;
```

- This query will return four rows in the patient table with the associated values contained in its 7 columns.

Rule 2: **The guaranteed access rule**

Each and every datum (atomic value) in a relational database is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.

```
SELECT specialistArea FROM specialist WHERE specialistID_Number = 'MUR101';
```

- This will return the datum 'Periodontal Treatment' (where *MUR101* is the primary key used to identify this atomic value from the *specialistArea* column in the table, *specialist*)

Rule 3: **The systematic treatment of null values**

Null values (distinct from the empty character string or a string of blank characters or any other number) are supported in the fully relational DBMS for representing missing information in a systematic way, independent of data type.

```
SELECT specialistID_Number FROM treatment WHERE treatmentID_Number = 115;
```

- Because treatmentID_Number 115 - 'Tooth Whitening' - can be performed in-house and doesn't require specialist treatment, the datum returned in the *specialistID_Number* column is null, rather than a blank field, even though the data type in this column is varchar. Null values can be returned regardless of the data type of the column.

Rule 4: **Dynamic online catalogue based on relational model**

The database description is represented at the logical level in the same way as ordinary data, so that authorised users can apply the same relational language to its interrogation as they apply to regular data.

```
SELECT TABLE_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA =
'assignmentdatabase';
```

- This will return the name of the tables and the amount of rows in each table in the 'assignmentdatabase' database using the same relational language that would be used to return data from a table.

```
SELECT COLUMN_NAME, DATA_TYPE FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name = 'patient' AND
table_schema = 'assignmentdatabase';
```

- This will return the column name and the data type for the patient table

Rule 5: **The comprehensive data sub language rule**

A relational system may support several languages and various models of terminal use. However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings, and that is comprehensive in supporting all of the following items: data definition; view definition; data manipulation; integrity constraints; transaction boundaries.

```
ALTER TABLE patient ADD Email varchar(255);
```

- This will add an *Email* column to the *patient* table and is an example of data definition

```
CREATE VIEW outpatient AS SELECT treatmentID_Number, treatmentDescription, specialistID_Number FROM treatment
WHERE specialistTreatmentRequired = 'Yes';
```

- This will create a view called *outpatient* displaying information from the 3 columns selected above and is an example of view definition

```
SELECT * FROM payment WHERE billTotal > 100;
```

- This will display all rows where *billTotal* is greater than 100 euros and is an example of data manipulation.

```
ALTER TABLE payment ADD FOREIGN KEY bill_ID_Number REFERENCES bill(bill_ID_Number);
```

- This will add a foreign key constraint to the *payment* table that references the primary key of the *bill* table and is an example of integrity constraints;

Rule 6: **The view updating rule**

All views that are theoretically updateable are also updateable by the system.

```
UPDATE outpatient SET treatmentDescription = 'Periodontal Treatments' WHERE treatmentID_Number = 225;
```

- This will update the *outpatient* view, changing *treatmentDescription* to 'Periodontal Treatments' from 'Periodontal Treatment' for treatmentID_Number 225, while also changing *treatmentDescription* in the *treatment* table, upon which the view has been based.

Rule 7: **High-level insert, update and delete**

The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

```
UPDATE patient SET addressLine2 = 'Newtown, Co.  Waterford' WHERE addressLine2 = 'Newtown';
```

- This single command will update 4 rows in the *patient* table

Rule 8: **Physical data independence**

Applications, programmes and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.

- When working on this assignment I was able to access the database from my computer at work and also from my laptop at home as long as I exported the database sql file to a location accessible from both locations.

Rule 9: **Logical data independence**

Applications, programmes and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

```
ALTER TABLE patient ADD Email varchar(255);
```

- Adding this extra column Email to the *patient* table should not impact any other table or adversely affect the database or how it operates.

Rule 10: **Integrity independence**

The declaration of integrity constraints must be part of the language used to define the DB structure, not enforced by application programmes. This makes integrity checking a function of the DB, independent of applications.

```
ALTER TABLE treatment ADD FOREIGN KEY (specialistID_Number) REFERENCES specialist (specialistID_Number);
```

- Here we have referential integrity because the foreign key *specialistID_Number* in the *treatment* table references a valid primary key, *specialistID_Number*, in the specialist table.

Rule 11**: Distribution independence**

A relational DBMS has distribution independence.

Rule 12: **The non –subversion rule**

If a relational system has a low-level (single-record-at-a-time) language, that low-level language cannot be used to subvert or bypass the integrity or constraints expressed in the higher-level relational language (multiple-records-at-a-time).