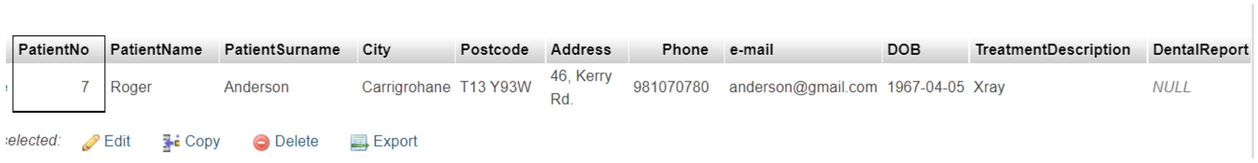
Contents

Rule 1: The Information Rule

The information in the relation database must be presented in the form of values stored in tables. Values stored in cells must be atomic. The order of rows in a relational table should not affect the meaning of the values.

SELECT * FROM patientchart WHERE PatientNo = 7



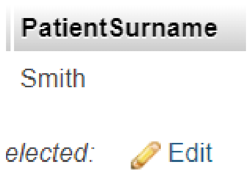
PatientNo	PatientName	PatientSurname	City	Postcode	Address	Phone	e-mail	DOB	TreatmentDescription	DentalReport
7	Roger	Anderson	Carrigrohane	T13 Y93W	46, Kerry Rd.	981070780	anderson@gmail.com	1967-04-05	Xray	NULL

The sample above is from the PatientChart table. The record of every patient is stored in the table PatientChart and it contains 15 different patients, and each of them is uniquely identified with a PatientNo, which is the primary key.

Rule 2: The Guaranteed Access Rule

Each individual data element (value) is guaranteed to be logically accessible by a combination of table name, primary key (row value), and attribute name (column value).

SELECT PatientSurname FROM patientchart where PatientNo = 3



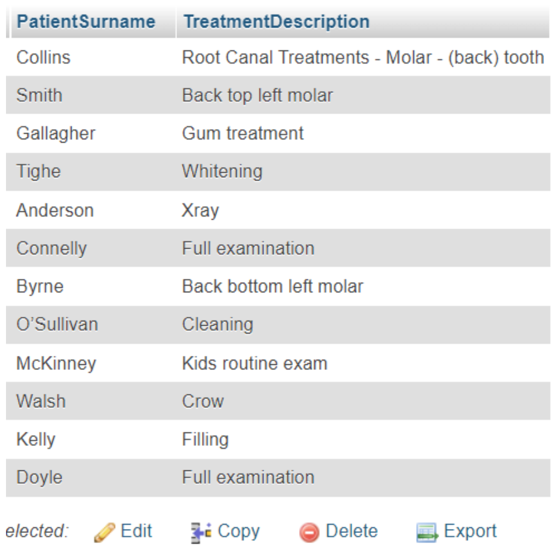
PatientSurname
Smith

The patient’s surname (single value) can be accessed from the table (patientchart) using primary key (PatientNo) and specifying the column name (PatientSurname).

Rule 3: Systematic Treatment of Null Values

Unknown NULL values other than any known value must be supported for all data types in all operations. For example, for numeric data, unknown values should not be treated as zeros, and for character data, as empty strings.

SELECT PatientSurname, TreatmentDescription FROM patientchart WHERE DentalReport IS NULL



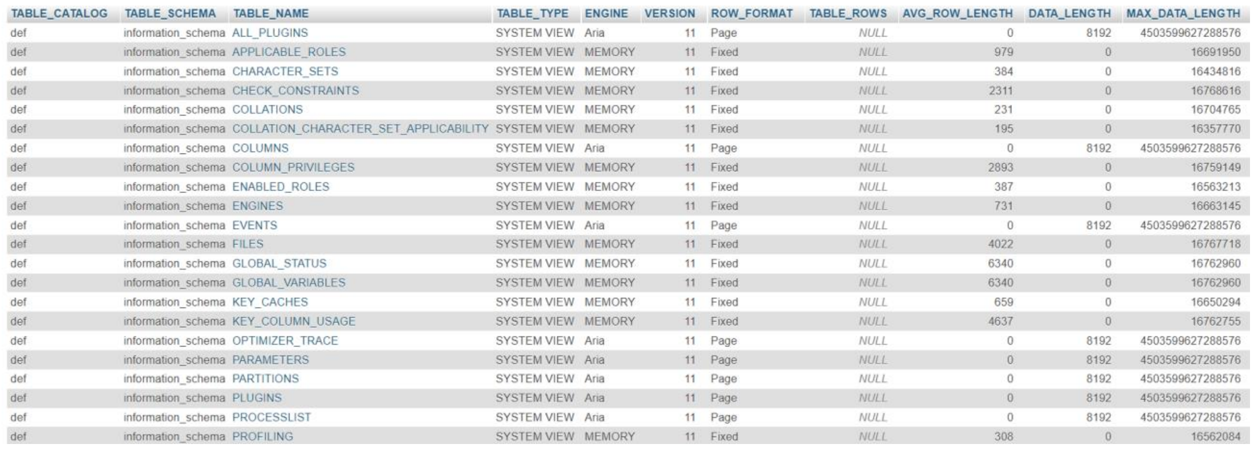
PatientSurname	TreatmentDescription
Collins	Root Canal Treatments - Molar - (back) tooth
Smith	Back top left molar
Gallagher	Gum treatment
Tighe	Whitening
Anderson	Xray
Connelly	Full examination
Byrne	Back bottom left molar
O'Sullivan	Cleaning
McKinney	Kids routine exam
Walsh	Crow
Kelly	Filling
Doyle	Full examination

Some patients have data in dental report and some of them do not, so their DentalReport is NULL. We were able to use a NULL placeholder irrespective of data type used. Records were retrieved without issue.

Rule 4: Dynamic Online Catalog based on the relational model

INFORMATION_SCHEMA provides access to database metadata and provides information about its structure - the name of the database, tables, column data types, etc. It looks like a database with several read-only tables. In fact, the tables are views, and the base itself is virtual. It is not stored as files on the server and is generated in memory during SQL startup.

```
SELECT * FROM information_schema.tables
```



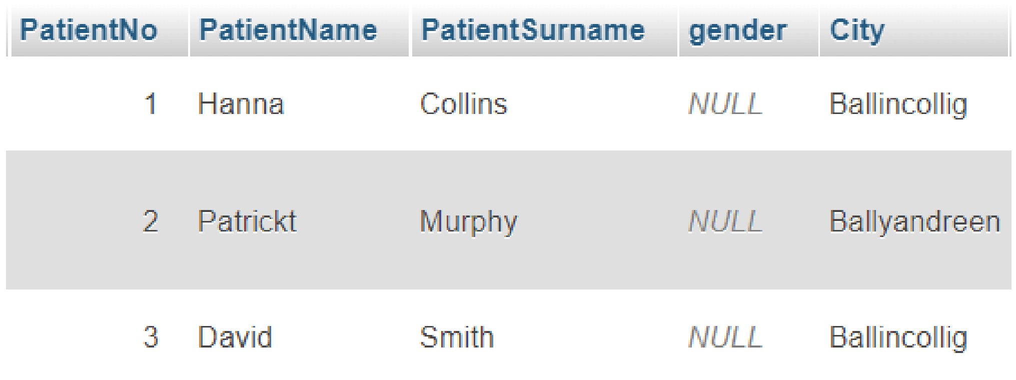
TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_ROWS	AVG_ROW_LENGTH	DATA_LENGTH	MAX_DATA_LENGTH
def	information_schema	ALL_PLUGINS	SYSTEM VIEW	Aria	11	Page	NULL	0	8192	4503599627288576
def	information_schema	APPLICABLE_ROLES	SYSTEM VIEW	MEMORY	11	Fixed	NULL	979	0	16691950
def	information_schema	CHARACTER_SETS	SYSTEM VIEW	MEMORY	11	Fixed	NULL	384	0	16434816
def	information_schema	CHECK_CONSTRAINTS	SYSTEM VIEW	MEMORY	11	Fixed	NULL	2311	0	16768616
def	information_schema	COLLATIONS	SYSTEM VIEW	MEMORY	11	Fixed	NULL	231	0	16704765
def	information_schema	COLLATION_CHARACTER_SET_APPLICABILITY	SYSTEM VIEW	MEMORY	11	Fixed	NULL	195	0	16357770
def	information_schema	COLUMNS	SYSTEM VIEW	Aria	11	Page	NULL	0	8192	4503599627288576
def	information_schema	COLUMN_PRIVILEGES	SYSTEM VIEW	MEMORY	11	Fixed	NULL	2893	0	16759149
def	information_schema	ENABLED_ROLES	SYSTEM VIEW	MEMORY	11	Fixed	NULL	387	0	16563213
def	information_schema	ENGINES	SYSTEM VIEW	MEMORY	11	Fixed	NULL	731	0	16663145
def	information_schema	EVENTS	SYSTEM VIEW	Aria	11	Page	NULL	0	8192	4503599627288576
def	information_schema	FILES	SYSTEM VIEW	MEMORY	11	Fixed	NULL	4022	0	16767718
def	information_schema	GLOBAL_STATUS	SYSTEM VIEW	MEMORY	11	Fixed	NULL	6340	0	16762960
def	information_schema	GLOBAL_VARIABLES	SYSTEM VIEW	MEMORY	11	Fixed	NULL	6340	0	16762960
def	information_schema	KEY_CACHES	SYSTEM VIEW	MEMORY	11	Fixed	NULL	659	0	16650294
def	information_schema	KEY_COLUMN_USAGE	SYSTEM VIEW	MEMORY	11	Fixed	NULL	4637	0	16762755
def	information_schema	OPTIMIZER_TRACE	SYSTEM VIEW	Aria	11	Page	NULL	0	8192	4503599627288576
def	information_schema	PARAMETERS	SYSTEM VIEW	Aria	11	Page	NULL	0	8192	4503599627288576
def	information_schema	PARTITIONS	SYSTEM VIEW	Aria	11	Page	NULL	0	8192	4503599627288576
def	information_schema	PLUGINS	SYSTEM VIEW	Aria	11	Page	NULL	0	8192	4503599627288576
def	information_schema	PROCESSLIST	SYSTEM VIEW	Aria	11	Page	NULL	0	8192	4503599627288576
def	information_schema	PROFILING	SYSTEM VIEW	MEMORY	11	Fixed	NULL	308	0	16562084

Rule 5: The Comprehensive Data Sub Language Rule

A relational database management system must support at least one relational language that
(a) has a linear syntax,
(b) can be used both interactively and in application programs,
(c) supports data definition (create, insert, update), view definition, data manipulation (interactive and programmatic), integrity constraints (primary key, foreign key, null values), access control, and transaction control operations (begin, commit, and rollback).

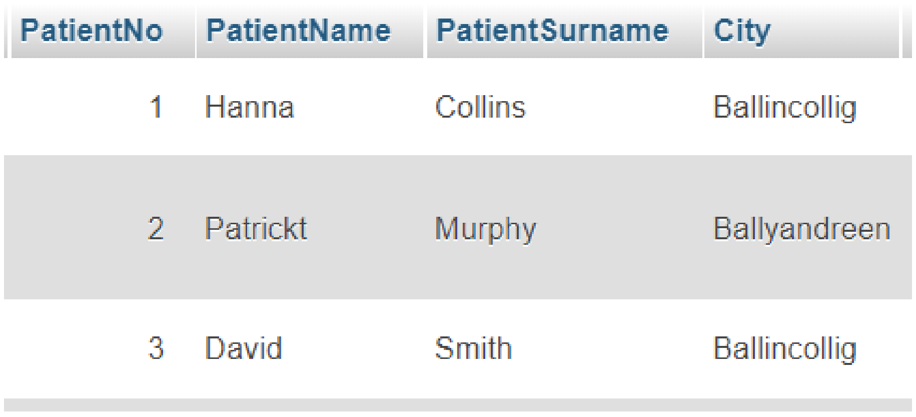
Example of DDL

```
-- create a new table "Appointment".

CREATE TABLE Appointment (
AppointmentNo        int(10) NOT NULL,
AppointmentDate      date default null,
AppointmentTime      time default null,
LateCnsltnFee        decimal(9,2) default null,
Reminder             bit(1) not null,
PatientNo            int(10) NOT NULL,

FOREIGN KEY (PatientNo) REFERENCES PatientChart(PatientNo),
primary key (AppointmentNo)
);
-- add a new column called 'gender'

ALTER TABLE `patientchart` ADD `gender` CHAR( 3) NULL AFTER `PatientSurname`
```



PatientNo	PatientName	PatientSurname	gender	City
1	Hanna	Collins	NULL	Ballincollig
2	Patrickt	Murphy	NULL	Ballyandreen
3	David	Smith	NULL	Ballincollig

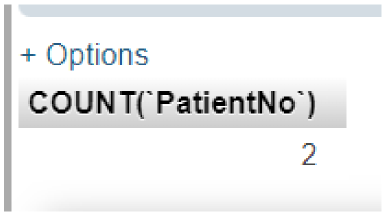
```
-- remove the column 'gender' from table

ALTER TABLE patientchart DROP gender
```



PatientNo	PatientName	PatientSurname	City
1	Hanna	Collins	Ballincollig
2	Patrickt	Murphy	Ballyandreen
3	David	Smith	Ballincollig

Example of DML

-- display number of patients who have outstanding payments
SELECT COUNT(`PatientNo`) FROM payment WHERE `DueUnpaidAmount` > 200

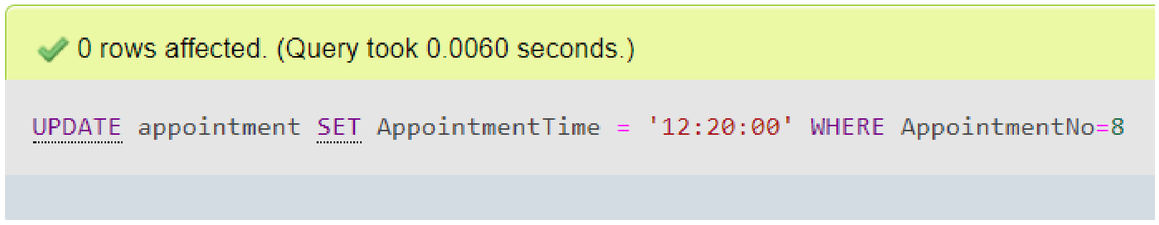


+ Options

COUNT(`PatientNo`)
2

-- Change appointment time

UPDATE appointment SET AppointmentTime = '12:20:00' WHERE AppointmentNo=8

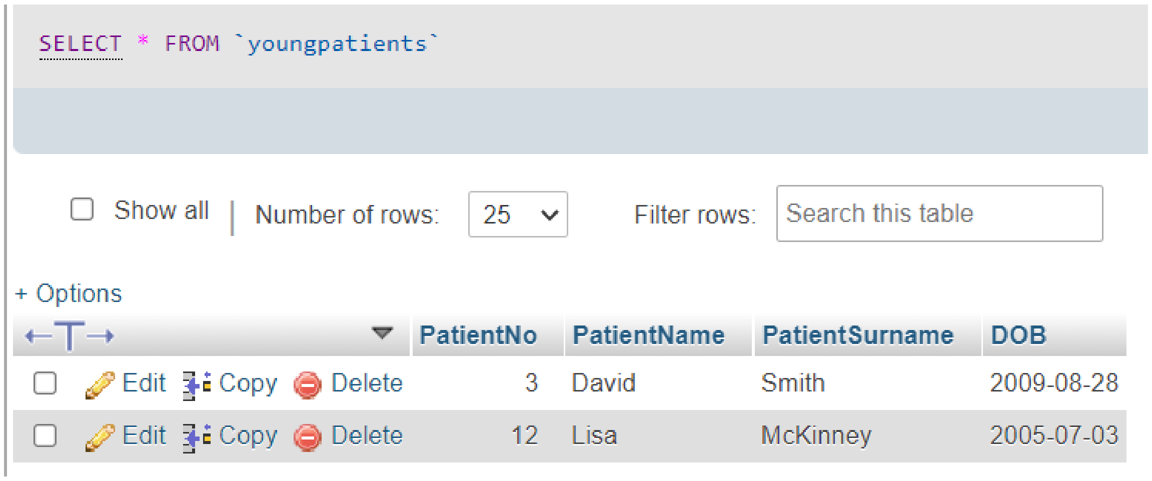


```
✓ 0 rows affected. (Query took 0.0060 seconds.)

UPDATE appointment SET AppointmentTime = '12:20:00' WHERE AppointmentNo=8
```

Rule 6: The View Updating Rule

Each view must support all data manipulation operations that relational tables support: insert, update, and delete operations.

CREATE view YoungPatients AS SELECT PatientNo, PatientName, PatientSurname, DOB fro
m patientchart WHERE DOB > '2000-01-01'



```
SELECT * FROM `youngpatients`
```

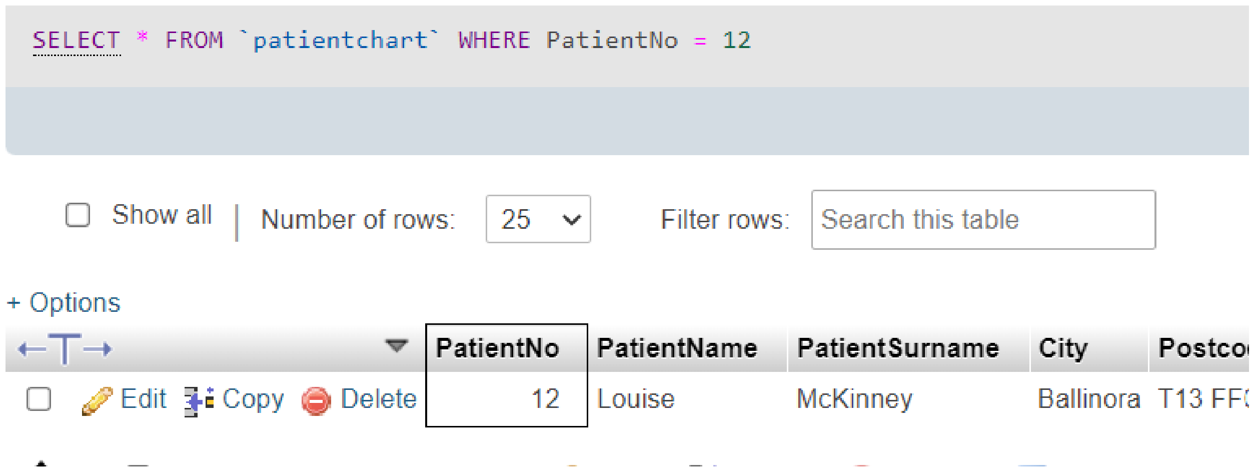
Show all | Number of rows: 25 Filter rows: Search this table

+ Options

PatientNo	PatientName	PatientSurname	DOB
3	David	Smith	2009-08-28
12	Lisa	McKinney	2005-07-03

UPDATE youngpatients SET PatientName = 'Louise' WHERE PatientNo = 12

SELECT * FROM `patientchart` WHERE PatientNo = 12



```
SELECT * FROM `patientchart` WHERE PatientNo = 12
```

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

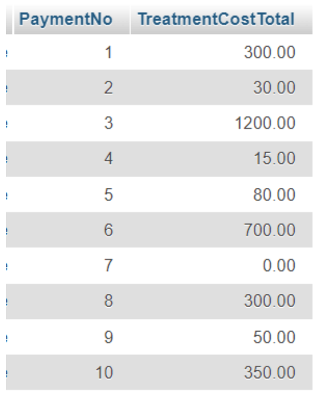
PatientNo	PatientName	PatientSurname	City	Postco
12	Louise	McKinney	Ballinora	T13 FF

The ‘PatientName’ field was updated in the ‘patientchart’ table after view updates

Rule 7: High Level Insert Update and Delete Rule

Insert, update, and delete operations must be supported not only on a single row of a relational table, but on any set of rows.

Before

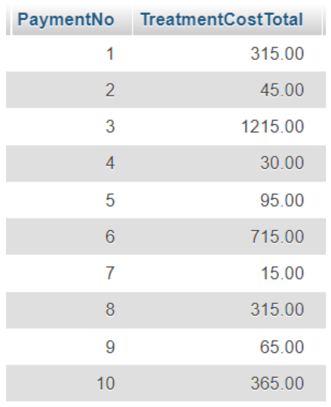


PaymentNo	TreatmentCostTotal
1	300.00
2	30.00
3	1200.00
4	15.00
5	80.00
6	700.00
7	0.00
8	300.00
9	50.00
10	350.00

After

--increase all treatment prices by 1 euro

```
UPDATE payment SET TreatmentCostTotal = (TreatmentCostTotal + 15)
```



PaymentNo	TreatmentCostTotal
1	315.00
2	45.00
3	1215.00
4	30.00
5	95.00
6	715.00
7	15.00
8	315.00
9	65.00
10	365.00

All multiple rows were updated at once

Rule 8: Physical Data Independence

Applications should not depend on the methods used to store data on media, on the hardware of computers on which the relational database is located.

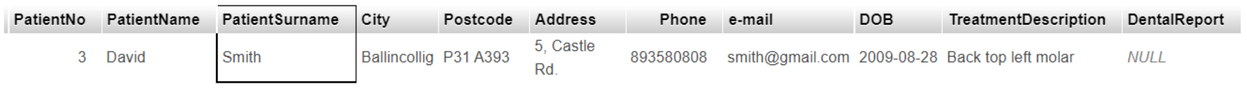
Using the Import and Export keys, one can transfer data from one computer to another. And even if these computers have a different version of the operating system installed, the database will be created and queries will work properly in the new database.

Rule 9: Logical Data Independence

The presentation of data in an application should not depend on the structure of relational tables. If normalization splits one relational table into two, the view must ensure that the data is merged so that changing the structure of the relational tables does not affect applications.

The addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

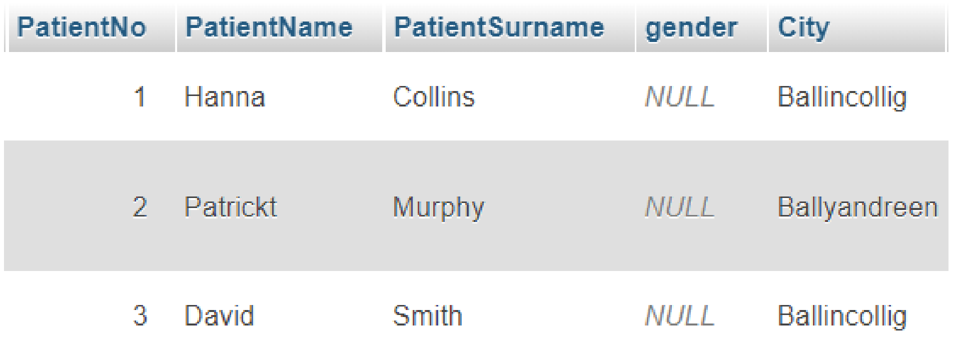
```
SELECT * FROM patientchart WHERE PatientSurname = 'Smith'
```



PatientNo	PatientName	PatientSurname	City	Postcode	Address	Phone	e-mail	DOB	TreatmentDescription	DentalReport
3	David	Smith	Ballincollig	P31 A393	5, Castle Rd.	893580808	smith@gmail.com	2009-08-28	Back top left molar	NULL

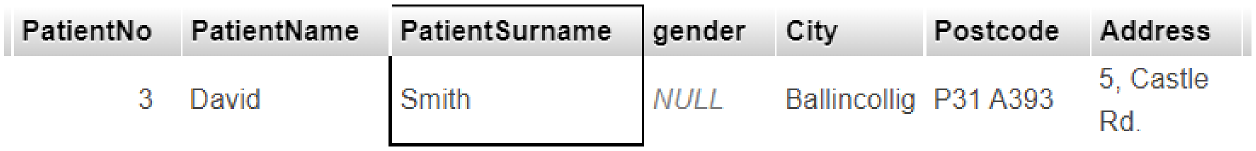
-- add a new column called 'gender' to the table 'patientchart'

```
ALTER TABLE `patientchart` ADD `gender` CHAR( 3) NULL AFTER `PatientSurname`
```



PatientNo	PatientName	PatientSurname	gender	City
1	Hanna	Collins	NULL	Ballincollig
2	Patrickt	Murphy	NULL	Ballyandreen
3	David	Smith	NULL	Ballincollig

```
SELECT * FROM patientchart WHERE PatientSurname = 'Smith'
```



PatientNo	PatientName	PatientSurname	gender	City	Postcode	Address
3	David	Smith	NULL	Ballincollig	P31 A393	5, Castle Rd.

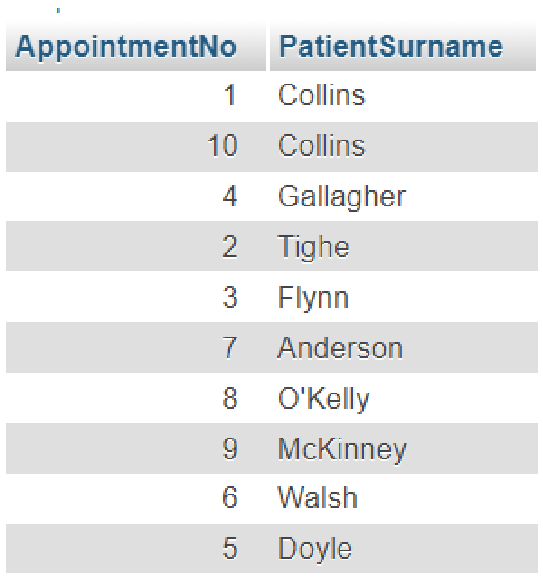
When the new column 'gender' was added to the table 'patientchart', the existing select statement still works.

Rule 10: Integrity Independence

All information necessary to maintain integrity must be in the data dictionary. A data language should perform input validation and automatically maintain data integrity.

The tables ‘Appointment’ and ‘PatientChart’ are linked via foreign key restrictions.

```
CREATE TABLE Appointment (

AppointmentNo        int(10) NOT NULL,
AppointmentDate      date default null,
AppointmentTime      time default null,
LateCnsltnFee        decimal(9,2) default null,
Reminder             bit(1) not null,
PatientNo            int(10) NOT NULL,

FOREIGN KEY (PatientNo) REFERENCES PatientChart(PatientNo),
primary key (AppointmentNo)
);


CREATE TABLE `PatientChart` (

  `PatientNo`          int(10) NOT NULL,
  `PatientName`        varchar(50) default NULL,
  `PatientSurname`     varchar(50) default NULL,
  `City`       varchar(50) default NULL,
  `Postcode`   varchar(50) default NULL,
  `Address`    varchar(200) default NULL,
  `Phone`      int(15) default NULL,
  `e-mail`     varchar(100) default NULL,
  `DOB`        date default null,
  `TreatmentDescription`       varchar(300) default NULL,
  `DentalReport`       varchar(300) default NULL,


  PRIMARY KEY (`PatientNo`)
);


SELECT appointment.AppointmentNo, patientchart.PatientSurname FROM appointment inner J
OIN patientchart ON appointment.PatientNo = patientchart.PatientNo
```



AppointmentNo	PatientSurname
1	Collins
10	Collins
4	Gallagher
2	Tighe
3	Flynn
7	Anderson
8	O'Kelly
9	McKinney
6	Walsh
5	Doyle

The inner join keyword selects records that have matching values in both tables. This query is based on the use of foreign keys in other tables to obtain the necessary information.

Rule 11: Distributed Independence

The database can be distributed, can be located on several computers, and this should not affect applications. Transferring a database to another computer should not affect applications.

This rule is an extension of rule 8 in that the database must be portable not only within a system (locally portable) but also across a network of multiple systems (remotely portable).

The end user should not see that the data is distributed in different places. Users should always be given the impression that the data is only on one site. This rule is considered the basis of distributed database systems.

Large organizations may store data on local sites to provide faster access to certain information; but to the user, it will appear as an organization database.

Rule 12 Non Subversion Rule

If a low-level data access language is used, it must not override the security and integrity rules that are supported by the higher-level language.

This rule prevents the use of database features other than the database language, as this could compromise the integrity of the database.

If some data has been changed in the file with the update request; the converted low-level language that updates the data entry in this file also maintains data integrity in memory.

References:

1) Lectures on the subject of database design and development, Dr. O.Foley

2) https://www.exploredatabase.com/2015/02/codds-twelve-rules.html