Codd's Rules – by Conor Keating

Rule 1 – The Information Rule

All information should be stored in a uniquely referenced combination of columns and rows. Using the following statement will return a single data value referenced by the column name and a primary key.

```
SELECT PATIENTSNAME

FROM PATIENT

WHERE PATIENTNO=2;
```

Rule 2 – The Guaranteed Access Rule

This rule is very similar to rule 1 in that it specifies that every piece of data in the database should be accessible using statements such as:

```
SELECT TREATMENTNAME

FROM TREATMENT

WHERE TREATMENTNO=3;
```

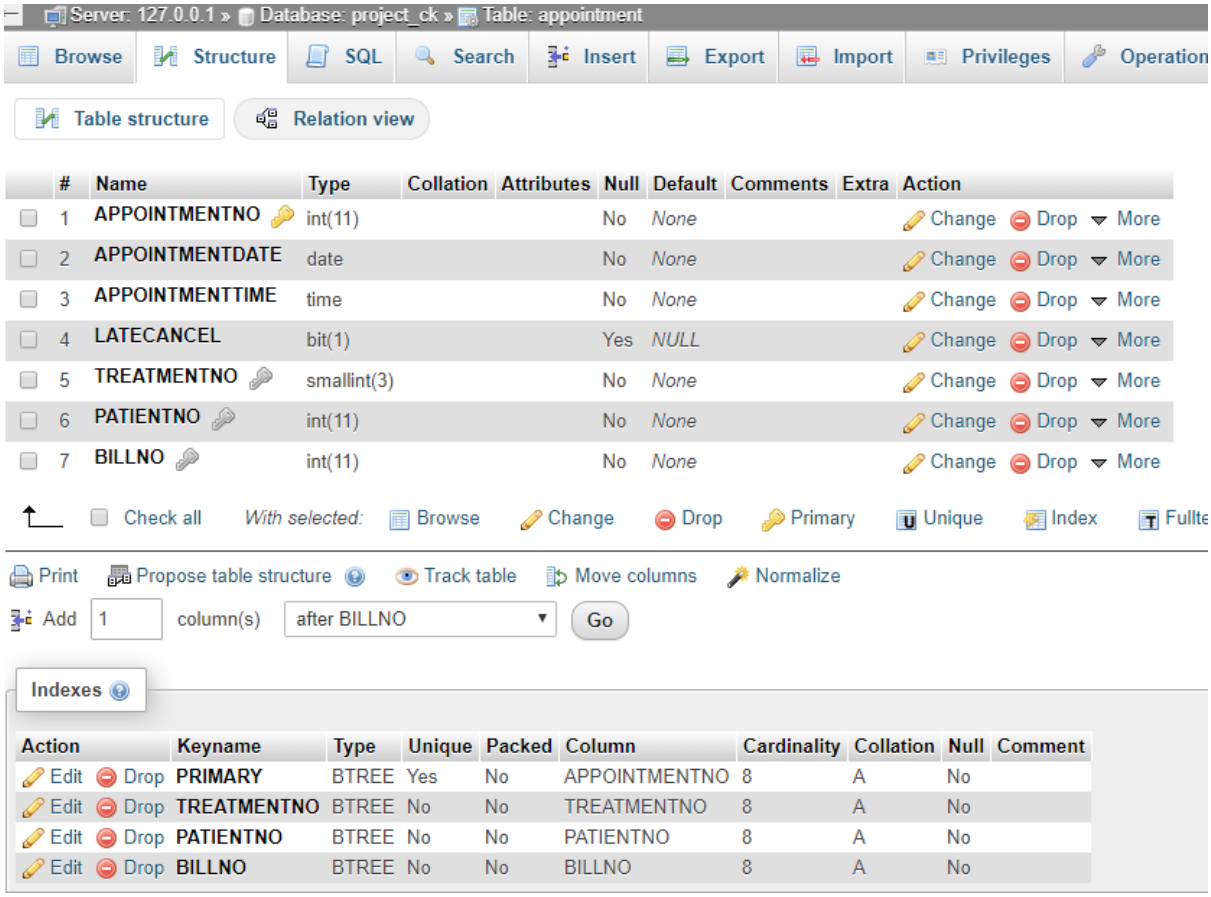
Rule 3 – Systematic Treatment of Null Values

Null values should be treated the same regardless of what data type they are governed by. The following statements place null values in a BIT field and a VARCHAR field and then query them.

```
UPDATE APPOINTMENT SET LATECANCEL=NULL WHERE APPOINTMENTNO=8;

UPDATE TREATMENT SET TREATMENTDESC=NULL WHERE TREATMENTNO=1;


SELECT APPOINTMENTNO

FROM APPOINTMENT

WHERE APPOINTMENT.LATECANCEL IS NULL;


SELECT TREATMENTNAME

FROM TREATMENT

WHERE TREATMENTDESC IS NULL;
```

Rule 4 – Dynamic Online Catalog Based on the Relational Model

The following snapshot displays the schema for the APPOINTMENT table. Primary and foreign keys are identified as well as the information about the data types used in each field.



Rule 5 – The Comprehensive Data Sub Language Rule

SQL is the language used to make changes to this database. The dentist table will be created, modified, queried and then deleted with the following commands.

```
CREATE TABLE DENTIST (
 DNO            VARCHAR(10) NOT NULL,
 FNAME                VARCHAR(10),
 SNAME                VARCHAR(20),
 REGNO                INT(10),
PRIMARY KEY (DNO));


INSERT INTO DENTIST VALUES
 (1, 'Tom', 'Ward', 452159),
 (2, 'John', 'Allen', 55625),
 (3, 'Jill', 'Burns', 998765);
```

UPDATE DENTIST

SET REGNO = 55471

WHERE FNAME = 'Jill' AND SNAME = 'Burns';

SELECT *

FROM DENTIST

WHERE DNO = 1 OR DNO = 2;

DROP TABLE DENTIST;

The following statement will result in an error because it does not follow the syntax of SQL.

SELECT * FROM PATIENT WHERE PATIENTNO==1;

Rule 6 – The View Updating Rule

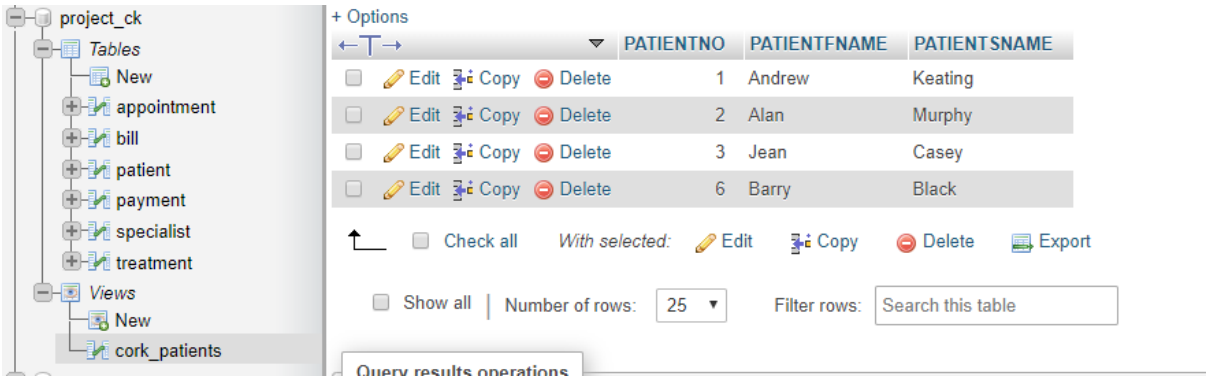
This rule states that updating of a view which is based on a single relation will translate into updating the value within that base relation. I will demonstrate by creating a view and allowing the view to edit the underlying base relation.

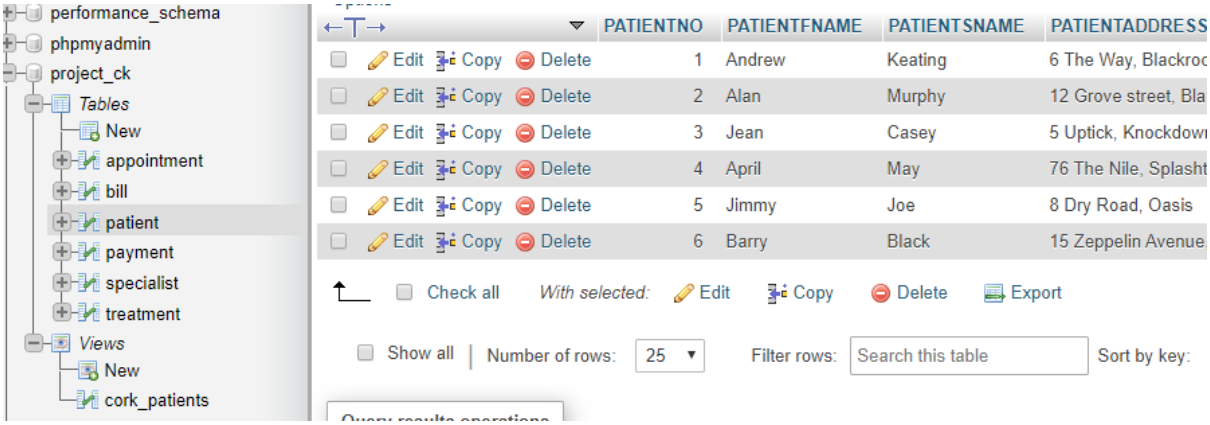
CREATE VIEW CORK_PATIENTS AS

SELECT PATIENTNO, PATIENTFNAME, PATIENTSNAME

FROM PATIENT

WHERE PATIENTCOUNTY='Cork';

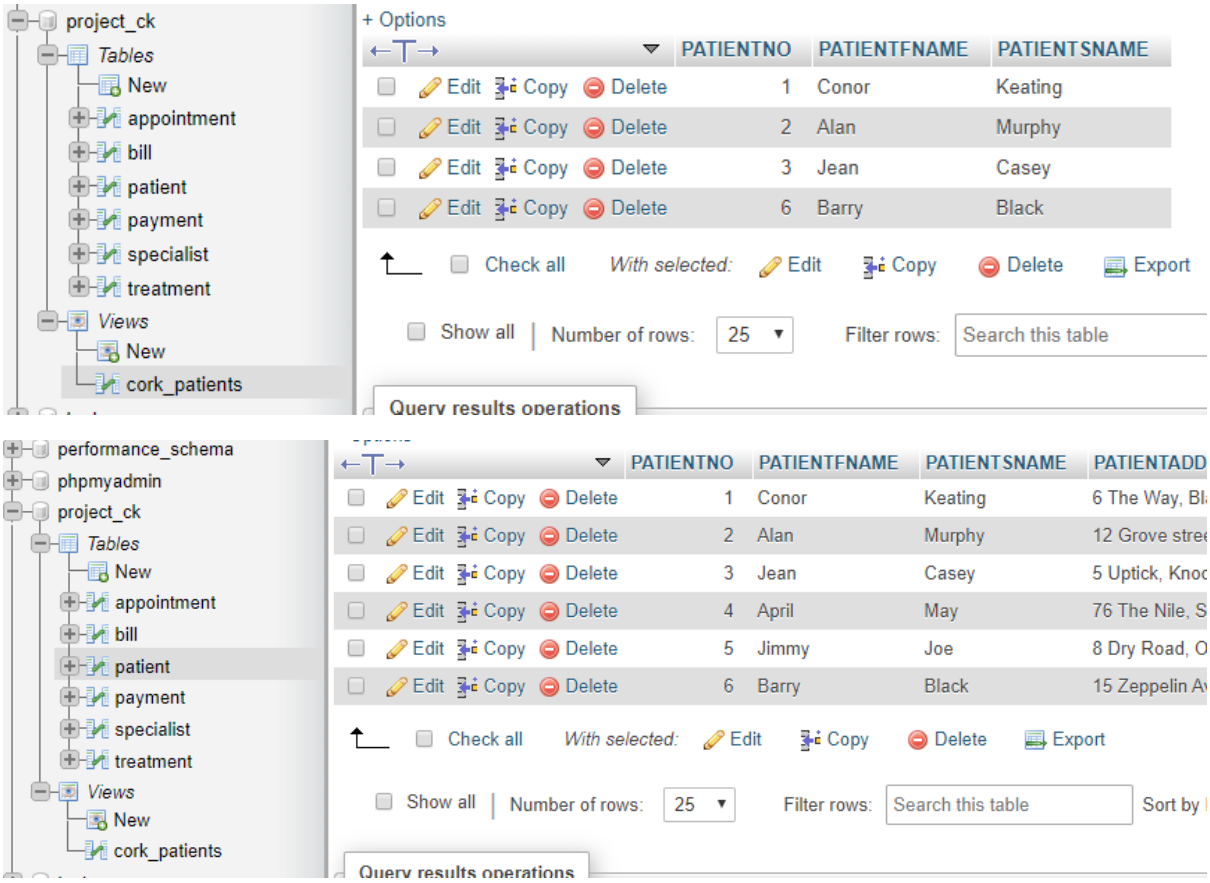




UPDATE CORK_PATIENTS

SET PATIENTFNAME = 'Conor'

WHERE PATIENTFNAME = 'Andrew';



Rule 7 – High Level Insert, Update and Delete Rule

This rule is demonstrated by running a query which will affect multiple rows. The following query achieves this.

UPDATE PATIENT

SET PATIENTCOUNTY='Rebelland'

WHERE PATIENTCOUNTY='Cork';

Rule 8 – Physical Data Independence

This rule implies that moving any data from one physical storage disk to another will not affect the functioning of the database if the relationships between that data is maintained. Moving the database from a primary partition to a secondary partition for example, will not affect functionality.

Rule 9 – Logical Data Independence

Logical data independence is achieved when changes to the logical level of the database do not affect the functionality of the database. For example, to include an extra table to record each dentist at the clinic will not affect any previous queries which may have ran against the database.

```
CREATE TABLE DENTIST (
 DNO            VARCHAR(10) NOT NULL,
 FNAME                VARCHAR(10),
 SNAME                VARCHAR(20),
 REGNO                INT(10),
PRIMARY KEY (DNO));
```

```
SELECT * FROM TREATMENT;
```

The select statement will result in the same output regardless of whether the new table is created or not.

Rule 10 – Integrity Independence

Integrity independence means that any application accessing data from the database should not be affected by certain changes to the primary or foreign keys. If the logic applied to the keys works, the application should work correctly.

```
SELECT PATIENTFNAME, PATIENTSNAME
FROM PATIENT, APPOINTMENT
WHERE APPOINTMENTNO=7 AND
APPOINTMENT.PATIENTNO=PATIENT.PATIENTNO;
```

The output is correctly achieved by using the foreign key in the APPOINTMENT table (PATIENTNO) to refer to the primary key in the PATIENT table (PATIENTNO).

Rule 11 – Distributed Independence

Data within a database may be spread across several different locations. For example, half of a databases tables may be stored locally on a hard drive in Dublin, while the other half may be stored in Cork. Thus, they are distributed across a network. To the end user or application, they should not be able to determine that this is the case. The functionality of the database should appear as if all tables are stored together.

Rule 12 – Non-Subversion Rule

Access to the data should be maintained only through the DBMS. In the case of operating systems which have access to the physical drives where the database is stored, that operating system should not be able to bypass the security and integrity constraints of the database which would allow them to make changes to the data.