**Codd’s 12 Rules for an RDBMS**

**Rule 1: The Information Rule**

*All information in the relational database is represented in exactly one and only one way—by values in tables.*

**SQL Code:**

SELECT PAT\_ID,

PAT\_FIRST\_NAME,

PAT\_LAST\_NAME,

PAT\_BIRTHDATE,

PAT\_ADDRESS,

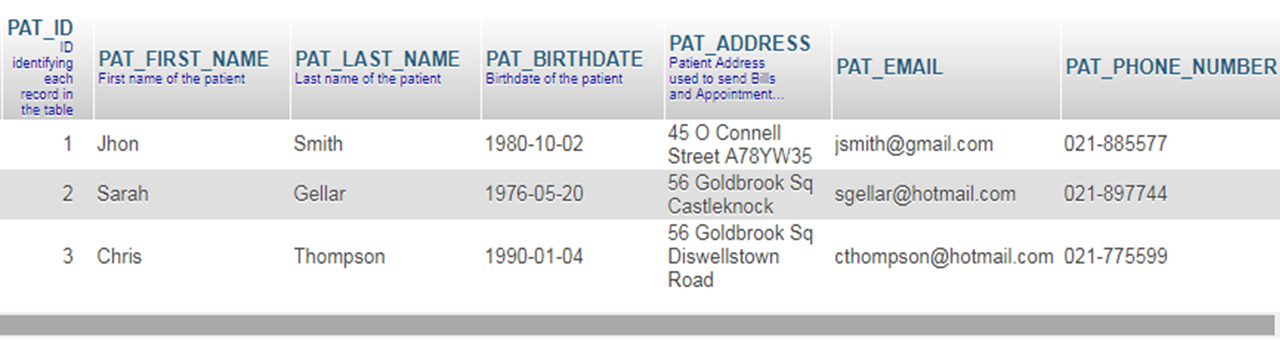
PAT\_EMAIL,

PAT\_PHONE\_NUMBER

FROM mydb.PATIENT;

**Explaining:** We can get all the information about the Patients of the Dentist’s Practice by selecting all columns from the PATIENT table.

Results obtained:



This rule is an informal definition of a relational database and indicates that every piece of data that we permanently store in a database is located in a table.

**Rule 2: Guaranteed Access Rule**

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

**SQL Code:**

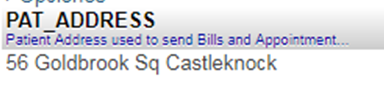
SELECT PAT\_ADDRESS

FROM mydb.PATIENT

WHERE PAT\_ID = 2;

**Explaining:** If we need to obtain the address of patient Sarah Gellar, we can access this data through the table name (PATIENT), the column name (ADDRESS) and the primary key value (PAT\_ID = 2).

Result obtained:



This rule stresses the importance of primary keys for locating data in the database. The table name locates the correct table, the column name finds the correct column, and the primary key value finds the row containing an individual data item of interest. In other words, each (atomic) piece of data is accessible by the combination of table name, primary key value, and column name.

**Rule 3: Systematic Treatment of NULL Values**

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

**SQL Code:**

SELECT ATRM\_ID,

ATRM\_APPOINTMENT\_ID,

ATRM\_TREATMENT\_ID,

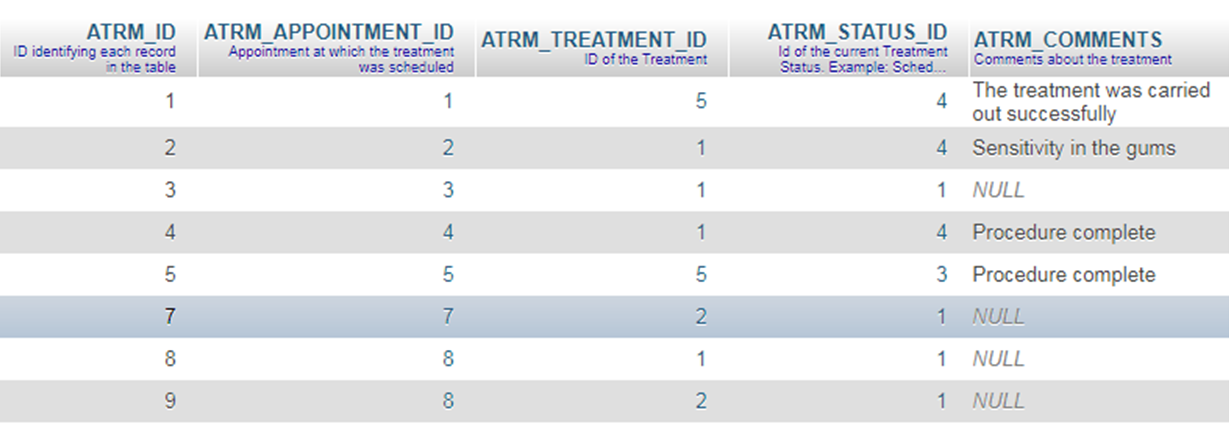
ATRM\_STATUS\_ID,

ATRM\_COMMENTS

FROM mydb.APPOINTMENT\_TREATMENT;

**Explaining:** The table APPOINTMENT\_TREATMENT, used to storage the treatments indicated to be performed during an appointment, can have NULL values in the column ATRM\_COMMENTS, as seen in the results obtained from the query.

Results obtained:



This rule requires that the RDBMS support a distinct NULL placeholder, regardless of data type. NULLs are distinct from an empty character string or any other number, and they are always to be considered as unknown values.

**Rule 4: Dynamic Online Catalog Based on the Relational Model**

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

**SQL Code:**

SELECT COLUMN\_NAME,

DATA\_TYPE,

CHARACTER\_MAXIMUM\_LENGTH,

IS\_NULLABLE,

COLUMN\_COMMENT

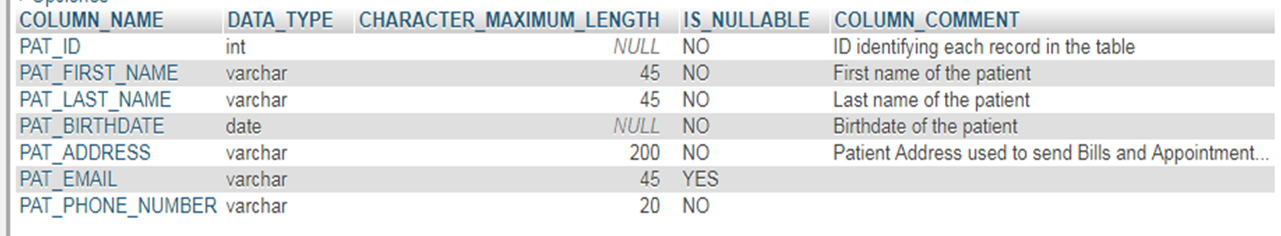
FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = 'PATIENT'

AND TABLE\_SCHEMA = 'mydb';

**Explaining:** The information of the tables is also storage in system tables such as INFORMATION\_SCHEMA.COLUMNS. In the example SQL code, the name, the type of data, the length, whether the columns of the Patient table can be NULL and comments are queried.

Results obtained:



This rule requires that a relational database be self-describing. In other words, the database must contain certain system tables whose columns describe the structure of the database itself, or alternatively, the database description is contained in user-accessible tables.

**Rule 5: Comprehensive Data Sublanguage Rule**

*A relational system may support several languages and various modes of terminal use. However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and whose ability to support all of the following is comprehensible: a. data definition b. view definition c. data manipulation (interactive and by program) d. integrity constraints e. authorization f. transaction boundaries (begin, commit, and rollback).*

**SQL Code:**

CREATE TABLE IF NOT EXISTS mydb.REFERRAL (

REF\_ID INT NOT NULL AUTO\_INCREMENT,

REF\_PATIENT\_ID INT NOT NULL,

REF\_SPECIALIST\_ID INT NOT NULL,

REF\_DATE DATE NOT NULL,

PRIMARY KEY (REF\_ID),

CONSTRAINT REFERRAL\_FK1 FOREIGN KEY (REF\_SPECIALIST\_ID ) REFERENCES mydb.SPECIALIST (SPC\_ID ),

CONSTRAINT REFERRAL\_FK2 FOREIGN KEY (REF\_PATIENT\_ID) REFERENCES mydb.PATIENT (PAT\_ID ));

UPDATE mydb.APPOINTMENT\_TREATMENT

SET ATRM\_COMMENTS = 'The treatment was carried out successfully'

WHERE ATRM\_APPOINTMENT\_ID = 1

AND ATRM\_ID = 1;

**Explaining:** The Create command is an example of integrity constraints and the Update command is an example of data manipulation.

This rule mandates the existence of a relational database language, such as SQL, to manipulate data. The language must be able to support all the central functions of a DBMS: creating a database, retrieving and entering data, implementing database security, and so on.

**Rule 6: View Updating Rule**

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

**SQL Code:**

CREATE OR REPLACE VIEW mydb.UNPAID\_TREATMENT AS

SELECT ATRM\_ID,

APP\_DATE,

PAT\_FIRST\_NAME,

PAT\_LAST\_NAME,

TRM\_NAME,

ATRM\_COMMENTS

FROM mydb.APPOINTMENT\_TREATMENT,

mydb .APPOINTMENT,

mydb .TREATMENT,

mydb .PATIENT

WHERE TRM\_ID = ATRM\_TREATMENT\_ID

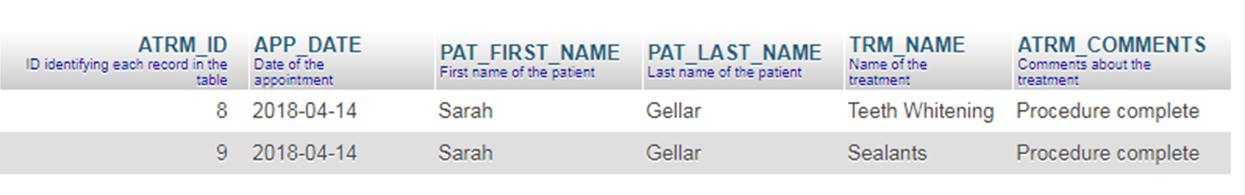
AND PAT\_ID = APP\_PATIENT\_ID

AND APP\_ID = ATRM\_APPOINTMENT\_ID

AND ATRM\_STATUS\_ID = 2;

**Explaining:** The example SQL code above creates the UNPAID\_TREATMENT view which contains the data of the treatments with PERFORMED status (value= 2), that is, they were performed and have not yet been billed. When a treatment is billed and its status changes, then the view will be updated and that treatment will be no longer in it.

The following figure shows the content of the view after the treatments of the 2018-04-14 day are performed on Patient Sarah Gellar.



This rule deals with views, which are virtual tables used to give various users of a database different views of its structure.

**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.*

**SQL Code:**

UPDATE mydb.APPOINTMENT\_TREATMENT

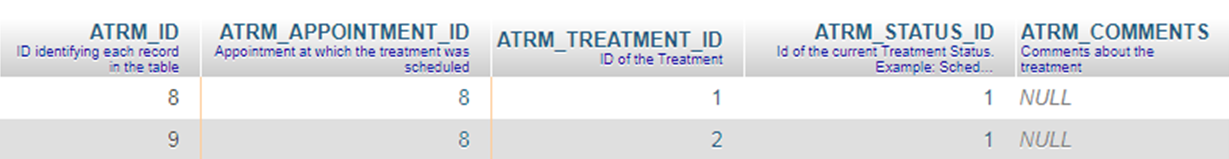
SET ATRM\_STATUS\_ID = 2,

ATRM\_COMMENTS = 'Procedure complete'

WHERE ATRM\_APPOINTMENT\_ID = 8;

**Explaining:** The SQL code updates the rows corresponding to the treatments of the appointment ID=8, setting status PERFORMED and indicating in the comments that the procedure was complete. In the next images the rows before and after the operation.

Before:



After:



This rule stresses the set-oriented nature of a relational database. It requires that rows be treated assets in insert, delete, and update operations. The rule is designed to prohibit implementations that support only row-at-a-time, navigational modification of the database. The SQL language covers this via the INSERT, UPDATE and DELETE statements.

**Rule 8: Physical Data Independence**

*Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.*

**SQL Code:**

ALTER TABLE mydb.PATIENT

MODIFY COLUMN PAT\_LAST\_NAME VARCHAR (50);

**Explaining:** The SQL code increases the length of the PAT\_LAST\_NAME column from 45 to 50 characters. This change should not affect the logic of the applications that work with the database.

Applications must still work using the same syntax, even when changes are made to the way in which the database internally implements data storage and access methods. This rule implies that the way the data is stored physically must be independent of the logical manner in which it’s accessed.

**Rule 9: Logical Data Independence**

*Application programs and terminal activities remain logically unimpaired when information preserving changes of any kind that theoretically permit unimpairment are made to the base tables.*

**SQL Code:**

ALTER TABLE mydb.SPECIALIST

ADD SPC\_EMAIL VARCHAR (45);

**Explaining:** The Alter statement adds a new column to the SPECIALIST table. After this change the application programs that work with database should be unimpaired.

Along with rule 8, this rule insulates the user or application program from the low-level implementation of the database. Together, they specify that specific access or storage techniques used by the RDBMS—and even changes to the structure of the tables in the database—shouldn’t affect the user’s ability to work with the data.

**Rule 10: Integrity Independence**

*Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.*

**SQL Code:**

CREATE TABLE IF NOT EXISTS mydb.BILL\_STATUS (

BST\_ID INT NOT NULL AUTO\_INCREMENT,

BST\_NAME VARCHAR(50) NOT NULL,

BST\_DESCRIPTION VARCHAR(100) NOT NULL,

PRIMARY KEY (BST\_ID) );

CREATE TABLE IF NOT EXISTS mydb.BILL (

BILL\_NUMBER INT NOT NULL AUTO\_INCREMENT,

BILL\_PATIENT\_ID INT NOT NULL,

BILL\_DATE DATE NOT NULL,

BILL\_STATUS\_ID INT NOT NULL,

BILL\_AGREED\_PAYMENTS INT NOT NULL DEFAULT 1,

PRIMARY KEY (BILL\_NUMBER),

CONSTRAINT BILL\_FK1 FOREIGN KEY (BILL\_PATIENT\_ID ) REFERENCES mydb.PATIENT (PAT\_ID ),

CONSTRAINT BILL\_FK2 FOREIGN KEY (BILL\_STATUS\_ID ) REFERENCES mydb.BILL\_STATUS (BST\_ID ));

**Explaining:** The SQL example code creates two tables, BILL\_STATUS and BILL. Both tables have their respective primary key, which cannot be NULL, and in addition table BILL has a mandatory foreign key that refers to the primary key of BILL\_STATUS. With these characteristics, rule number 10 is demonstrated.

The database must support a minimum of the following two integrity constraints:

• *Entity integrity*: No component of a primary key is allowed to have a NULL value.

• *Referential integrity*: For each distinct non-NULL foreign key value in a relational database, there must exist a matching primary key value from the same domain.

This rule says that the database language should support integrity constraints that restrict the data that can be entered into the database and the database modifications that can be made. In other words, the RDBMS must internally support the definition and enforcement of entity integrity (primary keys) and referential integrity (foreign keys).

**Rule 11: Distribution Independence**

*The data manipulation sublanguage of a relational DBMS must enable application programs and terminal activities to remain logically unimpaired whether and whenever data are physically centralized or distributed.*

**Explaining:** This rule says that the database language must be able to manipulate data located on other computer systems. In essence, we should be able to split the data on the RDBMS out onto multiple physical systems without the user realizing it.

**Rule 12: Non-Subversion Rule**

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

**Explaining:** This rule requires that alternate methods of accessing the data are not able to bypass integrity constraints, which means that users can’t violate the rules of the database in any way.