# Codd's Rules

**Rule 1: The Information Rule.**
"All information in a relational database is represented explicitly at the logical level in exactly one way - by values in a table."

Statement to demonstrate:

SELECT firstName, surname FROM patients WHERE patientID = 5661;

Explanation:
All information in a relational database is to be represented as data stored in cells in a table. The only way we can interface with data and make changes in the database is by applying SQL to the logical structure. We are able to retrieve unique data sets based on the logical links. The above SELECT statement uses the primary key 'patientID' to return a unique data set: the patient's first name and surname.

**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."

Statement to demonstrate:

SELECT surname FROM specialist WHERE specialistID = 15;

Explanation:
This rule specifies how we access data using a language. Three items are required to locate any piece of data i.e. the table name, the primary key of the row and the column name. This rule reinforces that primary keys are of the utmost importance when locating data in the database. Assuming the primary key has been defined, it is guaranteed to be unique. We can perform a search for the primary key value using SQL and then once we find the row, the data can be accessed via the column name.
The above SELECT statement uses the table name 'specialist', the primary key 'specialistID', and the column name 'surname' to find the required data.

**Rule 3: 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.”

Statement to demonstrate:

SELECT reportReceived, reportPdf FROM referrals WHERE specialistID = 16;

Explanation:
A NULL placeholder must be supported by the database management system, regardless of data type. The database management system must handle NULL in a systematic way.
A NULL value is an unknown value and it is not acceptable to use a blank space or numerical zero to represent missing data, NULL must be used. This is a strength of the relational model as if we don’t have a complete data set we can use NULL and still query the database. It should be noted that a primary key cannot be NULL.
The above SELECT statement returns a NULL for both ‘reportReceived' and ‘reportPdf’ as the data is unknown for these fields. NULL is returned for both regardless of the fact they have two different data types i.e. ‘reportReceived' has a varchar data type and ‘reportPdf’ has a mediumblob data type.

**Rule 4:Dynamic Online Catalogue based on the 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.”

Statement to demonstrate:

SELECT table_name, table_type, engine FROM information_schema.tables WHERE table_schema = ‘mulcahydental’ ORDER BY table_name;

Explanation:
The relational database must be self describing i.e. contain information describing the structure of the database itself. The data we store about the data must be stored in the relational format e.g. data dictionaries are made up of a set of tables with properties identical to the tables used for data. This allows authorised users to use the same relational language to examine it as they use for regular data.
The above statement requests a list of all the tables in the database ‘mulcahydental’ and displays the table names, table types and engine information in order of table name.

**Rule 5: The 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 that is comprehensive in supporting all of the following items: Data Definition, View Definition, Data Manipulation (interactive and by program), Integrity Constraints, Transaction Boundaries (begin, commit and rollback).”

Statements to demonstrate:

1. CREATE TABLE lineitem(
   lineitemID int(11) NOT NULL,
   name varchar(20) NOT NULL,
   cost decimal(6,2),
   PRIMARY KEY(lineitemID));
   paymentID int(8) NOT NULL,
   billID smallint(5) NOT NULL,
   paymentDate datetime,
   amountPaid decimal(6,2),
   PRIMARY KEY(paymentID),
   FOREIGN KEY(billID) REFERENCES bill(billID));

2. INSERT INTO patients (firstName, surname) VALUES (‘John’, ‘Murphy’);
3. SELECT * FROM patients WHERE county = ‘Cork’;

Explanation:
There must be a relational database language in existence to manipulate data. This language must be capable of supporting the functions of a database management system e.g. creating a database, entering and retrieving data etc.
SQL is an extremely popular relational database language with a large user base. In examples 1 and 2 above, SQLlanguage is used to create the tables ‘lineitem’ and ‘payment’ in the database, to specify the names of the columns of the tables and the type of data the columns can hold, and to define primary and foreign keys.
In example 3 above SQL language is used to enter data into the table ‘patients’.
In example 4 above SQL language is used to retrieve all the data for the patients from Cork from the table ‘patients’.

**Rule 6:The View Updating Rule.**
“All views that are theoretically updateable are also updateable by the system”.

Statements to demonstrate:

1. CREATE VIEW lineitemName AS SELECT lineitemID, name FROM lineitem;
2. UPDATE lineitemName SET name = ‘BLEACHING’ WHERE lineitemID = 9008;
3. SELECT * FROM lineitem;

Explanation:
Views are virtual tables and allow different users of a database to have different views of its structure. This can be useful to summarise information or restrict access to sensitive data. If a view is composed of columns that directly correspond to real table columns, then it is theoretically updateable. A database management system should have the capability of handling the update and communicating the updates back to the base tables. However this rule can be difficult to implement properly and views should be used carefully.
The above statements demonstrate the Updating Rule:
1. The view ‘lineitemName’ is built and includes the lineitemID and name columns from the ‘lineitem’ table.
2. The ‘lineitemName’ view is updated and where the lineitemID is 9008, the ‘name’ should be updated to ‘BLEACHING’.
3. A SELECT statement is executed to display all the information in the base ‘lineitem’ table and where the lineitemID is 9008; the ‘name’ should have been updated to ‘BLEACHING’.

**Rule 7: High Level Insert Update and Delete Rule.**
“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.”

Statement to demonstrate:

DELETE FROM referrals WHERE specialistID = 15;

Explanation:
For INSERT, DELETE AND UPDATE operations, this rule requires that rows be treated as sets. These operations must be able to be performed on more than one row with a single command. Without this it would be very tedious to carry out multiple updates.
The above statement specifies logical criteria to identify the rows to be updated i.e. wherever the specialistID is 15, the row should be deleted from ‘referrals’. The database management system finds the rows and performs the operation.

**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.”

Statement to demonstrate:

1. ALTER DATABASE mulcahydental SET OFFLINE;
2. ALTER DATABASE mulcahydental
   MODIFY FILE (NAME = mulcahydental_Data,
   FILENAME = ‘E:\newsqlfiles\mulcahydental_Data.mdf’);
3. ALTER DATABASE mulcahydental
   MODIFY FILE (NAME = mulcahydental_Log,
   FILENAME = ‘E:\newsqlfiles\mulcahydental_Log.ldf’);
4. ALTER DATABASE mulcahydental SET ONLINE;

Explanation:
This rule implements the ANSI/SPARC model. The physical storage of data must be independent of the logical way it is accessed and abstraction is used to shield applications and users from changes to the physical storage of the database. The database management system maps the conceptual schema to the host system and this allows for the rule to be fulfilled. Data can be accessed using the same queries even if the physical location of the files on the file system is moved.
The above statements are used to move SQL database files to a new location. Note: if a database is being used by any resources, these must be stopped and existing database connections must be closed. The location/names of the MDF(main database file) and LDF(log file) should be found and a folder created on the drive you want to move the files to (newsqlfiles on the E drive in this case) before beginning.

1. This procedure usually requires some downtime, it can be done with the database live however it is better to take it offline and this statement will take ‘mulcahydental’ offline.
2. and 3. These statements alter the database and change the file names to move the database to the new location.
4. This statement will take ‘mulcahydental’ back online.

This change of location of files should have no impact on the operation of the database.

**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.”

Statement to demonstrate:

```
ALTER TABLE patients ADD telNumber varchar(50);
```

Explanation:
This shields the user or applications from the low-level implementation of the database e.g. changes to the structure of the tables in the database at a logical level will not impact the user’s ability to work with the data and will have no impact on existing applications.
In the statement above a column ‘telNumber’ is added to the table ‘patients’, this will not impact existing applications.

**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.”

Statement to demonstrate:

```
DELETE FROM specialist WHERE specialistID = 18;
```

Explanation:
The database language must support integrity constraints that restrict the data that can be entered into the database and the permitted modifications. This means the database must support entity integrity and referential integrity. Entity integrity relates to the fact that a primary key cannot have a NULL value. Referential integrity relates to the fact that for each non-NULL foreign key value, a matching primary key value must exist in the same domain.
When the above statement was run in the mulcahydental database an error occurred. The error message stated “cannot delete or update a parent row: a foreign key constraint fails.” This occurred because the specialistID 18 was also in the referrals table.

**Rule 11:Distributed Independence.**
“A relational DBMS has distribution independence.”

Explanation:
This rule states that we should be able to split the data and put it onto different physical systems without the user realising it.

Theory example:
In the ‘mulcahydental’ database the patients table could be split by county and queries should still work.

**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.”

Explanation:
This rule states that alternative methods of accessing data should not be in a position to bypass any integrity constraints.

Theory example:
It is possible to disable constraints. If, for example, the NOT NULL constraint on the patientID primary key was disabled in the ‘mulcahydental’ database, this could cause huge issues for data integrity and the dental practice as each patient would not have a unique idenifier.