DATA TECHNOLOGIES

Codd's Rules

Fionn Loftus - Student No.G00329882

1. Information Rule

All information in a relational database is represented in only one way, by values in tables.

We can demonstrate this by getting the database to present us data in the only way it can- through tables.

```
select * from guest
```

This command shows all the data from the table guest.

2. Guaranteed Access Rule

This rule refers to logical accessibility, in that we must be able to access every piece of data in the database.

This has to be done through a combination of table name, primary key and column name. The primary key is necessary to guarantee uniqueness.

```
select Guest_ID,Guest_num
from reservation
where Res_ID = 1005;
```

In this demonstration "reservation" is the table name "Guest_ID" and "Guest_num" are the required fields and Res_ID is the primary key.

3. Systematic treatment of Null values

Null values are different from empty string values and from zero and other numbers. It represents missing or not applicable data in the database.

In this database example the user "Valerie Johns" did not complete the booking form, leaving "contact_no" blank and the data missing.

It is therefore by default replaced with a null value.

```
select guest_ID, F_Name
from Guest
where contact_no is Null;
```

4. Dynamic online catalogue based on the relational model

This rule requires a database to contain system tables whose columns describe the structure of the database itself.

This can be demonstrated in the Hotel_reservation database the sql command below which describes the primary keys and foreign keys and table name which are contained within the metadata.

```
SELECT *
FROM information_schema.TABLE_CONSTRAINTS
where CONSTRAINT_SCHEMA = "hotel_reservation";
```

5. Comprehensive Data Sublanguage Rule

This rule requires the existence of an SQL(Structured Query Language)type language in order to manipulate data within the database.

This rule can be demonstrated by using SQL to perform a transaction within the database. In this example I use SQL to change the name of a customer whose guest_ID is "101"

```
update guest

set F_Name="John"

where Guest_ID= 101;
```

Here we used SQL to change the value of F_Name field

6. View Updating Rule

This rule refers to views which are virtual tables used to give various users of a database a different view of its structure. This rule is very hard to implement and few commercial products fully satisfies it today.

As shown in this product MySQL, a view is created using the first SQL command but when the second command is used to edit the view the command instead edits the view and the field in the primary table.

This demonstrates that the view updating Rule is not implemented in MySQL as if it were only the view would be changed.

```
CREATE VIEW booked rooms AS

SELECT Room.Room_ID, Room.Capacity, Room.Type_ID, Guest.F_Name, Guest.L_Name,
Reservation.Res_ID

from room, guest, reservation

where Reservation.Guest_ID=Guest.Guest_ID and Reservation.Room_ID=Room.Room_ID;
```

```
update booked_rooms

set F_Name="Micheal"

where Res_ID= 1001;
```

7. High-level insert, update, and delete

This rule states that every query language used by the database should have the ability to INSERT, DELETE and UPDATE files in the database.

This rule is very important for a database as without it the database is a static table unable to be altered.

This database is compliant with this rule as all three commands are usable.

```
insert into guest (Guest_ID, F_Name, L_Name, Email, Contact_No, Address, DOB)

Values (107, "Patrick", "Murphy", "PM@hotmail.com", "0857768246","Sligo", "2016-7-9")
```

```
INSERT into reservation (Res_ID, Guest_ID, Room_ID, Check_in, Check_out, Guest_num)

values (1007, 107, 14, "2016-01-20", "2016-01-25",2)
```

```
update guest

set F_Name="Liam", L_Name="Kelly"

where Guest_ID= 107;
```

```
delete from reservation

where guest_ID= 107;
```

In these commands we created a row with "insert", altered the data with update and then deleted the row in the reservation table.

8. Physical Data Independence

This refer to the condition that if there is any change in the physical storage of the data, it should not affect the data at the logical or external view.

This rule is demonstrated by finding the Hotel_reservation database in its current directory and moving it to another location and then observing that it works.

You can change the data directory of the databases by editing "my.ini" in MySQL's bin folder to change the directory of [datadir = "C:/xampp/mysql/data"] to elsewhere on the drive and the database will still work proving the rule.

9. Logical Data Independence

This is similar to Physical Data Independence, it differs in that if there are any changes to the logical view it should not be reflected in the user view.

This is can be demonstrated if we were run a query receive the results and then change the structure of a table by adding another column for example. If we then run the original query again it should output the same result proving that changes in the logical view has not resulted in changes to the user view.

If we were to run the query

```
Select F_Name, L_Name
From Guest
where address= "Galway"
```

The query output is "Micheal Smith"

```
ALTER TABLE Guest
ADD new_user varchar(24)
```

New column created "New user" the structure of the table is now changed.

Now if we run the SQL query again

```
Select F_Name, L_Name
From Guest
where address= "Galway"
```

The output is still "Micheal Smith" proving that there has been no change in the user view, proving the rule.

10. Integrity Independence

A minimum of the following two integrity constraints must be supported:

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 insists that the declaration of integrity constraints must be part of th language that is used to define the database structure.

To demonstrate this rule we will attempt to input a Null value as a primary key by using the following command.

```
INSERT into billing (Bill_ID)
VALUES (Null);
```

-- When executed this command is met with an error

-- Error

-- MySQL said

-- # 1048 - Column 'Bill_ID' cannot be null

The same is true for trying to delete a primary key in one table that is a foreign key in another table.

```
ALTER TABLE billing
DROP COLUMN Bill_ID
```

An error occurs and the transaction cannot be complete as a foreign key that is not linked to a primary will remain in the other table. It thus proves that the integrity of the database is maintained through this rule.

11. Distribution Independence

The distribution of the parts of any database to different sites/locations should be invisible to the end users.

That is like in the distributed database it should give a centralised effect to the end users who access it.

12 Non-Subversion Rule

This rule states that if a RDBMS has an interface that provides access to low level records, this interface must not be able to subvert the system and bypass security and integrity constraints.

Essentially there should be no way to modify the database structure other than through the multiple row database language like Structured Query Language (SQL)