Nnenna Maduekwe
G00388219

Codds Rules

Codds rule are a set of rules proposed by Edgar F. Codd to define that is required from a database management system in order for it to be considered relational i.e a relational database management system (RDBMS).

1. **The Information Rule:** 'All information in the relational database is represented in exactly one and only one way—by values in tables.' *(https://link.springer.com, 2021)*

   All information in my database is represented in one and only one way, explicitly by values in column positions within rows of tables.

   Example:

```
CREATE TABLE `treatment` (
  `TREATMENT_ID` int(10) NOT NULL,
  `TREATMENT_TYPE` varchar(50) DEFAULT NULL,
  `TREATMENT_COST` decimal(5,2) DEFAULT NULL,
  `SPECIALIST_REQUIRED` varchar(10) DEFAULT NULL
);
ALTER TABLE `treatment`
  ADD PRIMARY KEY (`TREATMENT_ID`),
  ADD KEY `TREATMENT_COST` (`TREATMENT_COST`);
```

   In this simple SELECT statement, unique values for TREATMENT_TYPE and TREATMENT_COST will be returned using the primary key TREATMENT_ID.

```
SELECT TREATMENT_TYPE, TREATMENT_COST
FROM treatment
WHERE TREATMENT_ID = 10007;
```

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' (https://link.springer.com, 2021).*
   Each piece of data in my database can be uniquely identified by the combination of primary key value (finds the row containing an individual data item of interest), table name (locates the correct table), and attribute/ column name (locates the correct column)

   Examples:

```
CREATE TABLE `patient` (
  `PATIENT_ID` int(10) NOT NULL,
  `NAME` varchar(50) DEFAULT NULL,
  `SURNAME` varchar(50) DEFAULT NULL,
  `DATE OF BIRTH` DATE DEFAULT NULL,
  `ADDRESS LINE 1` varchar(50) DEFAULT NULL,
  `ADDRESS LINE 2` varchar(50) DEFAULT NULL,
  `PHONE` varchar(20) DEFAULT NULL,
  `EMAIL` varchar(100) DEFAULT NULL,
  `CASE_MANAGER` int(11) DEFAULT NULL
);
```

```
ALTER TABLE `patient`
  ADD PRIMARY KEY (`PATIENT_ID`),
  ADD KEY `CASE_MANAGER` (`CASE_MANAGER`);

ALTER TABLE `patient`
ADD CONSTRAINT `patients_DC_1` FOREIGN KEY (`CASE_MANAGER`)
REFERENCES `employees` (`employeeNumber`);
```

Using a simple SELECT statement, unique values can be accessed in a table via a combination of table name, primary key and column name.

```
SELECT NAME, SURNAME, PHONE
FROM patient
WHERE CASE_MANAGER is NULL
ORDER BY NAME;
```

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' (https://link.springer.com, 2021).*
   In my dental database, all Primary Keys contain a NOT NULL value. And if there is no data existing, NULL values are assigned it .

Example:

```
CREATE TABLE `BILL` (
  `INVOICE_ID` varchar(15) NOT NULL,
  `PATIENT_ID` int(10) NOT NULL ,
  `TREATMENT_COST` decimal(5,2) DEFAULT NULL,
  `CANCELLATION_FEE` decimal(5,2) DEFAULT NULL,
  `TOTAL_COST` decimal(5,2) DEFAULT NULL
);

SELECT PATIENT_ID, INVOICE_ID, TOTAL_COST
FROM BILL
WHERE CANCELLATION_FEE !='NULL';

SELECT PATIENT_ID, INVOICE_ID, TOTAL_COST
FROM BILL
WHERE CANCELLATION_FEE IS NULL
```

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.' (https://link.springer.com, 2021)*

   Those who are given access to my database structure can implement a similar query language used to access and retrieve data from the actual database.

5. **Comprehensive Data Sub-language 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: data definition, view definition, data manipulation (interactive*

and by program), integrity constraints, authorization and transaction boundaries (begin, commit, and rollback)' (https://link.springer.com, 2021).

Users of my database can use relational database language, such as SQL, to query and manipulate data. This language supports all the central functions of my DBMS; it includes Data Definition (create, alter, drop), View Definition, Data Manipulation (select, insert, update, delete), Integrity Constraints (primary key, foreign key, null values) Authorization (GRANT, REVOKE) and Transaction boundaries (begin, commit, rollback).

Examples:

```
CREATE TABLE `bill` (
  `INVOICE_ID` varchar(15) NOT NULL,
  `PATIENT_ID` int(10) NOT NULL ,
  `TREATMENT_COST` decimal(5,2) DEFAULT NULL,
  `CANCELLATION_FEE` decimal(5,2) DEFAULT NULL,
  `TOTAL_COST` decimal(5,2) DEFAULT NULL
);

SELECT Max(TOTAL_COST), NAME, SURNAME FROM patient, bill where patient.PATIENT_ID = bill.PATIENT_ID;

INSERT INTO `offices` (`officeCode`,`addressLine1`, `addressLine2`, `city`, `phone`, `postalCode`) VALUES ('1','100 Market Street', NULL, 'Baltimore, Cork', '091333487',  'IE94P80');

UPDATE patient
set PHONE = '0898873002'
where PATIENT_ID = 05;

DELETE from employees
where employeeNumber = 1077;

ALTER TABLE `payment`
  ADD PRIMARY KEY (`PAYMENT_ID`)

ALTER TABLE `payment`
  ADD CONSTRAINT `Pay_DC_1` FOREIGN KEY (`INVOICE_ID`) REFERENCES `bill` (`INVOICE_ID`);

CREATE VIEW pt_Appt  AS
SELECT PATIENT_ID, TIME, TREATMENT_ID
FROM appointment
WHERE STATUS = 'CONFIRMED';

DROP VIEW personnel ;
```

6. **View Updating Rule**: *'All views that are theoretically updateable are also updateable by the system' (https://link.springer.com, 2021).*

These views are virtual tables used to provide various users of my database different views of its structure.

Examples:

```
CREATE VIEW pt_Appt  AS
SELECT PATIENT_ID, TIME, TREATMENT_ID
FROM appointment
```

```
WHERE STATUS = 'CONFIRMED';

DROP VIEW pt_Appt;

SELECT * FROM pt_Appt;
```

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.' (https://link.springer.com, 2021)*

   Using a single query, users of my database can retrieve multiple rows and columns of data. They can also insert, delete and update multiple rows and columns of data through a single query.

   Examples:

```
INSERT INTO `offices` (`officeCode`,`addressLine1`, `addressLine2`, `city`, `phone`, `postalCode`) VALUES ('1','100 Market Street', NULL, 'Baltimore, Cork', '091333487', 'IE94P80');

UPDATE patient
set PHONE = '0898873002'
where PATIENT_ID = 05;

DELETE from employees
where employeeNumber = 1077;
```

8. **Physical data independence:** *'Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.' (https://link.springer.com, 2021)*

   Applications must still work using the same syntax. If data is updated or the physical structure of my database is changed, it will not interrupt the application that is accessing the database. Each data are not dependent on other data or an application.

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.' (https://link.springer.com, 2021)*

   If changes are made to my databases at a logical level (tables) it will have no affect on existing applications or queries. Existing queries will still work when new tables are created.

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.*' (https://link.springer.com, 2021)*

    To ensure an accurate and consistent data, my database is able to implement constraints to protect the data from invalid values, and that referential integrity is attained. By ensuring the accurate data types are used and no component of a primary key is allowed to have a NULL value (entity integrity).
    Example:

```
CREATE TABLE `bill` (
  `INVOICE_ID` varchar(15) NOT NULL,
  `PATIENT_ID` int(10) NOT NULL ,
  `TREATMENT_COST` decimal(5,2) DEFAULT NULL,
  `CANCELLATION_FEE` decimal(5,2) DEFAULT NULL,
  `TOTAL_COST` decimal(5,2) DEFAULT NULL
);


ALTER TABLE `bill`
  ADD PRIMARY KEY (`INVOICE_ID`),
  ADD KEY `TREATMENT_COST` (`TREATMENT_COST`,`TOTAL_COST`),
  ADD KEY `Bill_DC_2` (`PATIENT_ID`);

ALTER TABLE `bill`
  ADD CONSTRAINT `Bill_DC_1` FOREIGN KEY (`TREATMENT_COST`) REFERENCES `treatment` (`TREATMENT_COST`),
  ADD CONSTRAINT `Bill_DC_2` FOREIGN KEY (`PATIENT_ID`) REFERENCES `patient` (`PATIENT_ID`);

SELECT Max(TOTAL_COST), NAME, SURNAME FROM patient, bill where patient.PATIENT_ID = bill.PATIENT_ID;
```

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' (https://link.springer.com, 2021).*

    Queries should work the same way regardless of where the data is stored. If my database data is distributed across different locations, the user should not be able to find or view that the data is located in different locations.

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.' (https://link.springer.com, 2021)*

    There should be no way to modify the structure of my database or change data values other than through the multiple row database language (like SQL). Different methods of accessing the data are not able to bypass integrity constraints, which suggest that users can't violate the rules of the database in any way.

References

Codd's 12 Rules for an RDBMS. [online] Available at: https://link.springer.com/content/pdf/bbm%3A978-1-4302-0161-8%2F1.pdf [Accessed 1 April 2021].