Database Project - Codds Rules

1. **Information Rule:** '*All information in a relational data base is represented explicitly at the logical level and in exactly one way — by values in tables.' (Webopedia.com, 2019)*

   All the informational data stored in my database (e.g. table names, column names and column data types) is represented as values within the tables as seen in the example below.

   Example:

```
CREATE TABLE `PATIENT_DETAILS` (
  `PATIENT_ID` int(10) NOT NULL,
  `NAME` varchar(50) DEFAULT NULL,
  `SURNAME` varchar(50) 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,
  `DATE OF BIRTH` DATE DEFAULT NULL
);
```

2. **Guaranteed Access Rule:** '*Each item of data in an RDBMS is guaranteed to be logically accessible by resorting to a combination of table name, primary key value, and column name.' (Webopedia.com, 2019)*

   Within the Database I created, I am able to uniquely identify and access each piece of data that is present within my database - as each unique piece of data has a primary key, a table name and an attribute.

   Examples:

```
CREATE TABLE `appointment_details` (
  `APP_ID` int(10) NOT NULL,
  `PATIENT_ID` int(10) NOT NULL,
  `DATE` date NOT NULL,
  `TIME` time NOT NULL,
  `TREAT_ID` int(10) NOT NULL,
  `CANCELLED` varchar(50) DEFAULT NULL
);

ALTER TABLE `appointment_details`
  ADD PRIMARY KEY (`APP_ID`),
  ADD KEY `PATIENT_ID` (`PATIENT_ID`),
  ADD KEY `TREAT_ID` (`TREAT_ID`);

ALTER TABLE `appointment_details`
  ADD CONSTRAINT `app_AB_1` FOREIGN KEY (`PATIENT_ID`) REFERENCES `patient_details` (`PATIENT_ID`),
  ADD CONSTRAINT `app_AB_2` FOREIGN KEY (`TREAT_ID`) REFERENCES `treatment_details` (`TREAT_ID`);
```

3. **Systematic Treatment of NULL values:** *'Null values (distinct from an empty character string or a string of blank characters and distinct from zero or any other number) are supported in a fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of the data type.' (Webopedia.com, 2019)*

   In my project I have made sure that any NULL values in the database I designed correspond to missing, unknown or not applicable values. All Primary Keys contain a code to ensure a NOT NULL value.

   Example:

```
CREATE TABLE `BILL_INVOICE` (
  `INVOICE_ID` varchar(15) NOT NULL,
  `PATIENT_ID` int(10) NOT NULL ,
  `TREAT_COST` decimal(5,2) DEFAULT NULL,
  `LATECANCEL_FEE` decimal(5,2) DEFAULT NULL,
  `TOTAL_COST` decimal(5,2) DEFAULT NULL
);
```

4. **Active Online Catalogue**: *'The database description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.' (Webopedia.com, 2019)*

   The make-up of the database should be stored in an online database. Those who are authorized to access my database should be able to apply the query language used (SQL) to the regular data and to any interrogational queries applied to the Database for data retrieval.

5. **Comprehensive Data Sub-language Rule:** '*A relational system may support several languages and various modes of terminal use (for example, the fill-in-blanks mode). 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, and transaction boundaries (begin, commit, and rollback).' (Webopedia.com, 2019)*

   My database provides a query language that allows the user to query and manipulate the created databases content. This includes Data Definition (use of the create, insert, update functions), View Definition, Data Manipulation (use of the alter, delete functions), Integrity Constraints (primary key, foreign key, null values, Authorization (use of the GRANT , REVOKE statements.) and Transaction boundaries (use of the begin, commit, rollback etc).

   Examples:

```
CREATE TABLE `SPECIALISTS` (
  `SPEC_CODE` varchar(15) NOT NULL,
  `TREAT_ID` int(10) NOT NULL ,
  `SPEC_NAME` varchar(50) DEFAULT NULL ,
  `SPEC_ADDRESS` varchar(50) DEFAULT NULL,
  `SPEC_NUMBER` varchar(13) DEFAULT NULL
);

SELECT PATIENT_ID , INVOICE_ID from bill_invoice where TOTAL_COST between 70.00 and 100.00;
```

```
INSERT INTO `APPOINTMENT_DETAILS` (100021, 17,'2019-03-29','11:00', 10001, 'NO');

UPDATE appointment_details
set TIME = '11:00'
where APP_ID = 100010 ;

DELETE from patient_details
where PATIENT_ID = 21;

ALTER TABLE `appointment_details`
  ADD PRIMARY KEY (`APP_ID`);

ALTER TABLE `appointment_details`
  ADD CONSTRAINT `app_AB_1` FOREIGN KEY (`PATIENT_ID`) REFERENCES `patient_details` (`PATIENT_ID`);

CREATE VIEW patApp AS
SELECT PATIENT_ID, TIME, TREAT_ID
FROM appointment_details
WHERE CANCELLED = 'NO';

DROP VIEW personnel ;
```

6. **View Updating Rule:** '*All views of the data which are theoretically updatable must be updatable in practice by the DBMS.' (Webopedia.com, 2019)*

   If I create a view in my project (i.e. a temporarily table that is derived from a few base tables), the database system should allow the view to automatically update its associated base tables if any alteration is made to the view by the user.

   Examples:

```
CREATE VIEW patApp AS
SELECT PATIENT_ID, TIME, TREAT_ID
FROM appointment_details
WHERE CANCELLED = 'NO';

DROP VIEW patApp ;

SELECT * FROM patapp;
```

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.' (Webopedia.com, 2019)*

   I can query, insert, delete or update sets of values in multiple rows/tables in a single command in my database. The project database should also support the ability to carry out a number of these commands in one operation.

   Examples:

```
SELECT PATIENT_ID , INVOICE_ID from bill_invoice where TOTAL_COST between 70.00 and 100.00;

INSERT INTO `APPOINTMENT_DETAILS` (100021, 17,'2019-03-29','11:00', 10001, 'NO');

UPDATE appointment_details
set TIME = '11:00'
where APP_ID = 100010 ;
```

```
DELETE from patient_details
where PATIENT_ID = 21;
```

8. **Physical data independence**: *'Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.' (Webopedia.com, 2019)*

   Any alterations I make to the physical level of my database i.e. the physical schema (e.g. addition or removal of new entities, attributes, or relationships) should not be able to modify the application (i.e. There is no need to rewrite the application if physical data level is altered.).

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.' (Webopedia.com, 2019)*

   Any changes to the database schema that I make to the logical level (i.e. the conceptual model) of my created database should not affect the application or user accessing it.

10. **Integrity Independence:** '*Integrity constraints specific to a particular relational data base must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.' (Webopedia.com, 2019)*

    For the lifespan of my database created for this project I need to ensure that all the data is correct and consistent. I do this by ensuring the correct data types are used and that the constraints are correctly used (e.g. unique primary and secondary keys). This is for the prevention of errors in the application caused by invalid data.

    Example:

```
CREATE TABLE `APPOINTMENT_DETAILS`(
  `APP_ID` int(10) NOT NULL,
  `PATIENT_ID` int(10) NOT NULL ,
  `DATE` DATE NOT NULL,
  `TIME` TIME NOT NULL,
  `TREAT_ID` int(10) NOT NULL ,
  `CANCELLED` varchar(50) DEFAULT NULL
);
ALTER TABLE `appointment_details`
  ADD PRIMARY KEY (`APP_ID`);

ALTER TABLE `appointment_details`
  ADD CONSTRAINT `app_AB_1` FOREIGN KEY (`PATIENT_ID`) REFERENCES `patient_details` (`PATIENT_ID`);
```

11. **Distribution Independence**: *'An RDBMS has distribution independence. Distribution independence implies that users should not have to be aware of whether a database is distributed.' (Webopedia.com, 2019)*

    If my database data is distribution over various locations, it should appear as if it is one database (The distribution of my data should not be visible to end-users i.e. the users shouldn't know whether its distributed/ not).

12. **Non-Subversion Rule:** *'If the database has any means of handling a single record at a time, that low-level language must not be able to subvert or avoid the integrity rules which are expressed in a higher-level language that handles multiple records at a time.' (Webopedia.com, 2019)*

    If a low-level interface was used in my database for challenging transactions. This interface should not be able to modify the database structure by bypassing integrity constraints to change the data in my database. This should only be possible by using the applied query language (SQL).

References

Webopedia.com. (2019). *What is Codd's Law (Codd's Rules)*. [online] Available at: https://www.webopedia.com/TERM/C/Codds_Rules.html [Accessed 1 May 2019].