Codd's Rules

Rule 1: The Information Rule

SELECT animalName FROM animal WHERE animalID = 6;

By using the primary key of the animal id, we can retrieve the name of a particular animal.

Rule 2: The guaranteed access Rule

SELECT medicationName FROM medicines_prescribed WHERE examinationNo = 100;

All data should be available unambiguously with a combination of table name, primary key and column name.

Rule 3: Systematic Treatment of Null Values

INSERT INTO medicines_prescribed (medicationName, medicationRegime, medicationCost, examinationNo) VALUES ('null', 'one injection given', '35.00', 210);

SELECT medicationName, medicationCost FROM medicines_prescribed WHERE examinationNo = 210;

The DBMS must support null values to represent missing or inapplicable information and be able to handle null regardless of the data type.

Rule 4: Dynamic Online Catalog based on the Relational model

SELECT * FROM INNODB_SYS_TABLES WHERE NAME = 'veterinary_clinic/animal';

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: The Comprehensive Data Sub Language Rule

SQL DDL query:
CREATE VIEW vterinarian_visits AS SELECT veterinarianID_fk, animalID_fk FROM animal_appoitment

WHERE veterinarianID_fk = 'J10_0203';

SQL DML query: SELECT * FROM vterinarian_visits;

Relational database can support multiple languages, however, there must be at least one language whose statements can be expressed according to a well-defined syntax.

Rule 6: The View Updating Rule

UPDATE medicines_used SET medicationRegime = '1 tablet per day for 2 weeks';

SELECT * FROM `medicines_used`;

A view is theoretically updatable if it consists of columns that directly correspond to the actual table columns.

Rule 7: High Level Insert Update and Delete Rule

INSERT INTO veterinarian (veterinarianID, veterinarianName, veterinarianSurname, position) VALUES ('A11_0918','Amanda', 'Curran', 'Veterinary Cleaner');

Requires rows to be treated as sets in insert, delete, and update operations. The rule is designed to prohibit implementations that only support row-by-time, navigational changes to the database.

Rule 8: Physical Data Independence
Any change at the physical level (file system) should not lead to a change at the logical level (tables, columns, rows, etc.).

Rule 9: Logical Data Independence

The term "logical independence of data" imposes the possibility of presenting stored information in diverse ways to different users. The way in which the data is stored and the way in which it is displayed to the individual user is independent. Different users perceive the same data differently.

For example, if the table medication_prescribed(medicationName, medicationRegime, medicationCost, reviewNo) a view with attributes (medicationName,medicationRegime) will not be affected if any other attribute of medication_prescribed changes.

Rule 10: Integrity Independence

CREATE TABLE `animal` (

`animalID` int(20) NOT NULL, -> ***primary key***

`animalName` varchar(50) NOT NULL,

`animalDOB` date NOT NULL,

`animalBread` varchar(50) NOT NULL,

`animalGender` varchar(10) NOT NULL,

`ownerID_fk` int(20) NOT NULL, -> ***foregin key*** references owner (ownerID)

`appoitmnentID_fk` int(20) NOT NUL**L -> *foregin key*** references appointments (appointmentID)

Rule 11: Distributed Independence

Distribution of parts of any database to different sites/locations should be invisible to end users. Also, existing applications can continue to run when the database is distributed or redistributed. This way of execution enables parallelism in the handling of transactions.

Rule 12: Non Subversion Rule

This rule dictates that there should be no way to violate the integrity constraints defined in the database in any form. Only the language/sublanguage that was used to define those constraints can redefine them such as SQL.

For example, violation can be done by disabling constraints and triggers options.
.