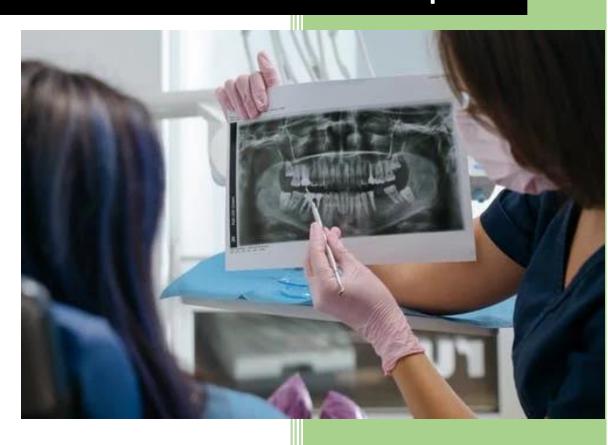
2021

Codd's Rules Examples



Máire Murphy G00375722 4/8/2021

Contents

C	odd's Rules	. 3
	Rule 1: The Information Rule	3
	Rule 2: The Guaranteed Access Rule	. 3
	Rule 3: Systematic Treatment of Null Values	. 3
	Rule 4: Dynamic Online Catalog based on the relational model	. 4
	Rule 5: The Comprehensive Data Sub Language Rule	. 4
	Rule 6: The View Updating Rule	. 6
	Rule 7: High Level Insert Update and Delete Rule	. 7
	Rule 8: Physical Data Independence	. 7
	Rule 9: Logical Data Independence	. 8
	Rule 10: Integrity Independence	. 8
	Rule 11: Distributed Independence	. 8
	Rule 12 Non Subversion Rule	q

Codd's Rules

Dr Edgar F Codd developed 12 rules that relational database Systems (RDBMS) should obey. Complying to these rules, result in correct data and relations in the database, making a perfect RDBMS.

Rule 1: The Information Rule

SQL	SELECT * FROM `treatment_guide` WHERE treatment_ID = "114"								
Result:	treatment_ID description price treatment_duration Approximate length of time in minutes to carry out 114 Routine Extraction 60 20								
Explanation	The Treatment_Guide table contains 13 different treatments. Each one is uniquely identified with a treatment_ID which is the primary key.								

Rule 2: The Guaranteed Access Rule

SQL	SELECT `name` FROM `patient` WHERE patient_ID = "2"
Result:	name te Kate Murphy
Explanation	The name of a patient (single value) can be retrieved using primary key and specifying the column name required.

Rule 3: Systematic Treatment of Null Values

SQL	show balance owed for each patient select name, bill_status, balance_owed from bill, patient whe re bill.patient_id = patient.patient_id									
Result:	name	bill_status	balance_owed							
	Mary Murphy	closed	NULL							
	Tim Collins	open	280							
	Kate Murphy	closed	NULL							
	Kate Murphy	closed	NULL							
	John Joe McGrath	closed	0							

Explanation	Several patients paid in full on the spot so their balance_owed is NULL. Other patients owe
	a balance or paid in instalments so their balance_owed has a value. Records are retrieved
	without issue.

Rule 4: Dynamic Online Catalog based on the relational model

Here is the meta data on the *treatment_guide* table and *patient_specialist_treatments* table taken from the data dictionary:

treatment_guide

Table comments: contains a list of treatment offerings by the Mulcahy dental practice

Column	Туре	Null	Default	Links to	Comments	Media (MIME) type
treatment_ID (Primary)	int(11)	No				
description	varchar(300)	No				
price	float	No				
treatment_duration	int(11)	No			Approximate length of time in minutes to carry out the procedure.	

Indexes

Keyname	Туре	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	treatment_ID	13	A	No	

patient_specialist_treatments

Table comments: Specialist treatments which have been referred by Mulcahy dentists are recorded in the patient's history.

Column	Туре	Null	Default	Links to	Comments	Media (MIME) type
sp_tr_ID (Primary)	int(11)	No				
patient_ID	int(11)	No		patient -> patient_ID		
dentist_ID	int(11)	No		dentists -> dentist_ID		
treatment_desc	varchar(300)	No				
referral_sent_date	date	No				
treatment_received	date	No				

Indexes

Keyname	Туре	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	sp_tr_ID	1	A	No	
fkIdx_96	BTREE	No	No	patient_ID	1	A	No	
fkIdx_99	BTREE	No	No	dentist_ID	1	A	No	

Rule 5: The Comprehensive Data Sub Language Rule

DDL	add a new column called 'notes'
Examples	ALTER TABLE `patient_specialist_treatments` ADD `notes` VARCHAR(
•	300) NULL AFTER `treatment received`;

```
-- remove the notes column from table
        ALTER TABLE `patient specialist treatments`
         DROP `notes`;
         --create a secretary table with 4 columns
        CREATE TABLE `dentist db`.`secretary`(
           `employee ID` INT(10) NOT NULL,
            `name` VARCHAR(100) NOT NULL,
            `address` DATE NOT NULL,
            `salary` FLOAT NOT NULL,
           PRIMARY KEY(`employee ID`)
        ) COMMENT = 'Temporary table for DDL demo';
             _____
        -- delete secretary table
        DROP TABLE `secretary`
        -- add a new office. Only the local office (Mulcahy's) has the -
DML
        - last four fields populated.
Examples
        INSERT INTO Office(
           `office ID`,
            `phone`,
            `address`,
            `practice name`,
            `secretary`,
            `bill amt threshold`,
            `overdue period`,
            `late_cancellation_fee`,
            `misc fee`
        VALUES (
           1,
            '(021) 461 4333',
           'Main st., Cobh, Co. Cork',
           'Mulcahy\'s Dental Practice',
            'Helen',
           400,
           30,
           10,
           NULL
        _____
        -- change a phone number
        UPDATE
            `patient phone number`
        SET
           `phone no` = '087 5778975'
```

```
patient_phone_number`.`eircode` = 'P24D123' AND
`patient phone number`.`phone no` = '087 5678975';
    _____
-- display number of patients who have outstanding bills
SELECT
   COUNT(`patient ID`)
FROM
   patient
WHERE
   `outstanding balance` > 0
-- select treatment history for patient with id=3 (Tim Collins)
-- given by MulCahy dentists.
SELECT
   t.description
FROM
   patient_chart details AS pcd,
   treatment guide AS t
WHERE
   pcd.patient_ID = "3" AND pcd.treatment_ID = t.treatment_ID
-- show complete history for a patient - local treatments and
-- any specialist treatments
SELECT
   t.description AS "Treatment History"
FROM
   patient chart details AS pcd,
   treatment guide AS t
WHERE
   pcd.patient ID = "2" AND pcd.treatment ID = t.treatment ID
UNION ALL
SELECT
   pst.treatment desc
FROM
   patient_specialist_treatments AS pst
WHERE
   pst.patient ID = "2"
```

Rule 6: The View Updating Rule

1. CREATE ALGORITHM = UNDEFINED VIEW `Treatments_Available_View` AS SELECT * FROM `treatment_guide`

2.	SELECT * FROM `treatments_available_view` WHERE `treatment_ID` = 116								
	treatment_ID	description	price	treatment_duration Approximate length of time in minutes to carry out					
	116	Crown	1200	30					
		l							
3.		_	_	v` SET `price` = '1250' WHERE `treatments_avail					
	able_view`.`treatment_ID` = 116;								
4.	SELECT * FROM	`treatment_g	guide` W	WHERE `treatment_ID` = 116					
	treatment_ID	description	price	treatment_duration Approximate length of time in minutes to carry out					
	116	Crown	1250	30					
Result	The price field is updated in the treatment_guide table after the view update.								

Rule 7: High Level Insert Update and Delete Rule:

```
--increase all treatment prices by 1 euro

UPDATE
    `treatment_guide`

SET
    price = (price + 1)

Result:

✓ 13 rows affected. (Query took 0.0293 seconds.)

Comment: DBMS allows multiple rows to be updated at once
```

Rule 8: Physical Data Independence

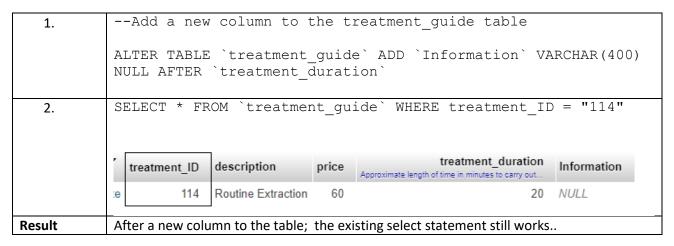
Case: Moving database to another machine that is faster and has more storage. The destination machine already has XAMPP with phpMyAdmin setup.

In source machine: In phpMyAdmin select the required database and in *Export* tab click 'go'. This will export the structure and data to a SQL file.

In destination machine: In phpMyAdmin, in *Import* tab, browse to the SQL file containing the structure and data and click 'go'.

The database should be created, and queries should work on new database (despite the destination machine having a different O.S. or hardware configuration).

Rule 9: Logical Data Independence



Rule 10: Integrity Independence

```
--display the total amount owed for a particular bill
           SELECT
               b.bill ID,
               SUM(tr.price)
           FROM
                `bill` AS b,
               bill items AS bi,
               treatment guide AS tr,
               patient chart details AS pcd
           WHERE
               b.bill ID = "2" AND b.bill id = bi.bill ID AND
           bi.chart item ID = pcd.chart item ID AND tr.treatment ID =
           pcd.treatment ID
              bill ID
                     SUM(tr.price)
                  2
                             280
           :te
Explanation
           This query relies on the use of foreign keys in other tables to retrieve the required
           information
```

Rule 11: Distributed Independence

End user experience of distributed database should feel and act like a non-distributed db. For example, large organizations may have data stored at local sites to enable faster access to certain information; but to the user it will appear as the organization database. MySQL\MariaDB support distributed DB functionality.

Rule 12 Non Subversion Rule

Example: If an employee's address was changed with an update query; the converted low-level language which updates the address record in the employee file also maintains the integrity of the data in the memory.