

2021

Codd's Rules Examples



Máire Murphy

G00375722

4/8/2021

Contents

Codd'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	9

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:	<table><tr><th>treatment_ID</th><th>description</th><th>price</th><th>treatment_duration</th></tr><tr><td>114</td><td>Routine Extraction</td><td>60</td><td>Approximate length of time in minutes to carry out... 20</td></tr></table>	treatment_ID	description	price	treatment_duration	114	Routine Extraction	60	Approximate length of time in minutes to carry out... 20
treatment_ID	description	price	treatment_duration						
114	Routine Extraction	60	Approximate length of time in minutes to carry out... 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:	<table><thead><tr><th>name</th></tr></thead><tbody><tr><td>Kate Murphy</td></tr></tbody></table>	name	Kate Murphy
name			
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	<pre>-- show balance owed for each patient select name, bill_status, balance_owed from bill, patient where re bill.patient_id = patient.patient_id</pre>																				
Result:	<table><tr><th>name</th><th>bill_status</th><th>balance_owed</th></tr><tr><td>Mary Murphy</td><td>closed</td><td>NULL</td></tr><tr><td>Tim Collins</td><td>open</td><td>280</td></tr><tr><td>Kate Murphy</td><td>closed</td><td>NULL</td></tr><tr><td>Kate Murphy</td><td>closed</td><td>NULL</td></tr><tr><td>John Joe McGrath</td><td>closed</td><td>0</td></tr></table>			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
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 <i>balance_owed</i> is NULL. Other patients owe a balance or paid in instalments so their <i>balance_owed</i> 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	Type	Null	Default	Links to	Comments	Media (MIME) type
treatment_ID (<i>Primary</i>)	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	Type	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	Type	Null	Default	Links to	Comments	Media (MIME) type
sp_tr_ID (<i>Primary</i>)	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	Type	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 Examples	-- add a new column called 'notes' ALTER TABLE `patient_specialist_treatments` ADD `notes` VARCHAR(300) NULL AFTER `treatment_received`;
---------------------	---

	<pre> ----- -- 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` </pre>
DML Examples	<pre> -- add a new office. Only the local office (Mulcahy's) has the - - last four fields populated. 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' WHERE </pre>

	<pre> `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" </pre>
--	---

Rule 6: The View Updating Rule

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

2.	<div>SELECT * FROM `treatments_available_view` WHERE `treatment_ID` = 116</div> <table><tr><th>treatment_ID</th><th>description</th><th>price</th><th>treatment_duration <small>Approximate length of time in minutes to carry out...</small></th></tr><tr><td>116</td><td>Crown</td><td>1200</td><td>30</td></tr></table>	treatment_ID	description	price	treatment_duration <small>Approximate length of time in minutes to carry out...</small>	116	Crown	1200	30
treatment_ID	description	price	treatment_duration <small>Approximate length of time in minutes to carry out...</small>						
116	Crown	1200	30						
3.	UPDATE `treatments_available_view` SET `price` = '1250' WHERE `treatments_available_view`.`treatment_ID` = 116;								
4.	<div>SELECT * FROM `treatment_guide` WHERE `treatment_ID` = 116</div> <table><tr><th>treatment_ID</th><th>description</th><th>price</th><th>treatment_duration <small>Approximate length of time in minutes to carry out...</small></th></tr><tr><td>116</td><td>Crown</td><td>1250</td><td>30</td></tr></table>	treatment_ID	description	price	treatment_duration <small>Approximate length of time in minutes to carry out...</small>	116	Crown	1250	30
treatment_ID	description	price	treatment_duration <small>Approximate length of time in minutes to carry out...</small>						
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:

	<pre>--increase all treatment prices by 1 euro UPDATE `treatment_guide` SET price =(price + 1)</pre>
Result:	
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

1.	--Add a new column to the treatment_guide table ALTER TABLE `treatment_guide` ADD `Information` VARCHAR(400) NULL AFTER `treatment_duration`
2.	SELECT * FROM `treatment_guide` WHERE treatment_ID = "114" 
Result	After a new column to the table; the existing select statement still works..

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 
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.