

# CODD’S RULES PROVED FOR THE SOUTH DUBLIN HOUSING COUNCIL REPAIRS DATABASE

## Contents

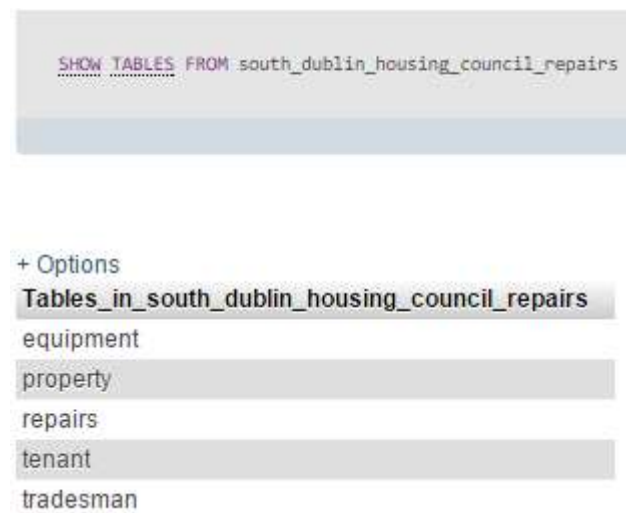
CODD’S RULES PROVED FOR THE SOUTH DUBLIN HOUSING COUNCIL REPAIRS DATABASE .....	1
RULE 1: THE INFORMATION RULE.....	2
RULE 2: THE GUARANTEED ACCESS RULE.....	5
RULE 3: SYSTEMATIC OF NULL VALUES.....	8
RULE 4: ACTIVE ONLINE CATALOG BASED ON THE RELATIONAL MODEL.....	11
RULE 5: THE COMPREHENSIVE DATA SUBLANGUAGE RULE.....	12
RULE 6: THE VIEW UPDATING RULE.....	15
RULE 7: HIGH-LEVEL INSERT, UPDATE, AND DELETE. ....	21
RULE 8: PHYSICAL DATA INDEPENDENCE .....	25
RULE 9: LOGICAL DATA INDEPENDENCE.....	29
RULE 10: INTEGRITY INDEPENDENCE.....	34
RULE 11: DISTRIBUTION INDEPENDENCE .....	36

## RULE 1: THE INFORMATION RULE

*“All information in a relational database is represented explicitly at the logical level in exactly one way – by values in tables”.*

This rule specifies that only two dimensional tables (relations) be the only data structures used in the relational database. A relation is the definition of a table with attributes (columns) and tuples (rows) in it. I typed SQL query's into phpMyadmin to prove that all information in the south\_dublin\_housing\_council\_repairs database is a relational database with tables in it which have attributes and tuples in it.

The SQL query *“SHOW TABLES FROM south\_dublin\_housing\_council\_repairs”* displays what tables are in my database.



To prove the information rule I typed the following SQL query into the console *“SELECT \* FROM ‘table’”*. This displays all the information from one of my table's when this SQL query is typed in the console. I typed this SQL query for each table to prove the information rule.

*SELECT \* FROM property;*

	POSTCODE	TENANT_ID	ADDRESS	REPAIR_ID
<input type="checkbox"/> Edit Copy Delete	A97 NX65	10	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	10
<input type="checkbox"/> Edit Copy Delete	D04 SN 807	9	54 Hazle Grove, Clondalkin	5
<input type="checkbox"/> Edit Copy Delete	D04 KN85	5	54 Sandymount Avenue	3
<input type="checkbox"/> Edit Copy Delete	D04 N2E9	2	89 Donnybrook Avenue	5
<input type="checkbox"/> Edit Copy Delete	D04 N824	16	THE SOUTH DOCK 18 BRIDGE STREET DUBLIN 4	7
<input type="checkbox"/> Edit Copy Delete	D06 P2N5	3	85 Rathmines Road Lower	4
<input type="checkbox"/> Edit Copy Delete	D06 W1D7	1	22 Belgrave Square South	1
<input type="checkbox"/> Edit Copy Delete	D08 F6W2	4	75 Lombard Street East	2
<input type="checkbox"/> Edit Copy Delete	D12 A2B9	12	75 RUTLAND GROVE, DUBLIN 12	6
<input type="checkbox"/> Edit Copy Delete	D24 WN33	21	32 MAPLEWOOD CLOSE DUBLIN 24	8

*SELECT \* FROM equipment;*

`SELECT * FROM equipment`

Options		EQUIPMENT_ID	TYPE	DESCRIPTION	VALUE	INSPECTION_DATE	STATE_OF_REPAIR
<input type="checkbox"/>	Edit Copy Delete	1	Drill	DEWALT pistol drill kit	16.00	2015-09-15	Good
<input type="checkbox"/>	Edit Copy Delete	2	Adjustable wrench	Stanley 300mm MaxSteel	14.00	2015-10-08	Excellent
<input type="checkbox"/>	Edit Copy Delete	3	Angle Grinder	Makita GA4530KD 720W	52.00	2015-10-12	Poor
<input type="checkbox"/>	Edit Copy Delete	4	Ladder	SKY528 Heavy duty foldable	120.00	2015-10-13	Good
<input type="checkbox"/>	Edit Copy Delete	5	Voltmeter	Voltmeter	10.00	2015-08-25	Excellent
<input type="checkbox"/>	Edit Copy Delete	6	Paint brush and roll	Stanley 8 piece decorating set	25.95	2015-12-14	Excellent
<input type="checkbox"/>	Edit Copy Delete	7	Pliers	Crescent 7" Tongue & Groove Pliers	17.50	2015-01-23	Poor
<input type="checkbox"/>	Edit Copy Delete	8	End cut Nippers	Channellock 7" End Cut Nippers	7.55	2015-11-23	Good
<input type="checkbox"/>	Edit Copy Delete	9	drill	Powerhouse 500	20.00	0000-00-00	Good
<input type="checkbox"/>	Edit Copy Delete	10	Shovel	Fiskars Heavy Duty Shovel	20.00	2015-12-01	Excellent
<input type="checkbox"/>	Edit Copy Delete	11	Hammer	Stanley claw hammer	8.49	2015-12-15	Good
<input type="checkbox"/>	Edit Copy Delete	12	Screwdriver	Stanley 10 piece screw driver set	20.00	2015-12-02	Good
<input type="checkbox"/>	Edit Copy Delete	13	wrench	Huskers 14 inch heavy duty pipe wrench	12.48	2015-12-12	Excellent
<input type="checkbox"/>	Edit Copy Delete	14	Saw	Stanley Heavy Duty 30 inch saw	15.50	2015-11-29	Excellent

`SELECT * FROM repairs;`

`SELECT * FROM repairs`

	REPAIR_ID	TYPE	URGENCY	DETAILS	DATE	TIME	TRADESMAN_ID
<input type="checkbox"/>	1	Plumbing	Urgent	Leak under sink unit	2015-11-17	14:45:00.000000	3
<input type="checkbox"/>	2	Electric	Emergency	Smoking fuse board	2015-11-18	15:00:00.000000	3
<input type="checkbox"/>	3	Plastering	Cyclical	Repairs to internal walls	2015-02-01	11:00:00.000000	5
<input type="checkbox"/>	4	Carpentry	Cyclical	Replace hall door	2015-05-05	10:00:00.000000	1
<input type="checkbox"/>	5	Plumbing	Routine	Replace toilet cistern	2015-09-10	12:00:00.000000	3
<input type="checkbox"/>	6	Woodwork	Cyclical	Replace damp skirting board	2015-12-14	11:00:00.000000	7
<input type="checkbox"/>	7	Painting	Cyclical	Painting of the outside wall of house	2016-10-15	10:00:00.000000	8
<input type="checkbox"/>	8	carpentry	Cyclical	construction of wooden fence in front garden	2016-06-05	10:30:00.000000	9
<input type="checkbox"/>	9	Flooded house	Emergency	HOUSE HAS FLOODED	2015-12-12	08:00:00.000000	10
<input type="checkbox"/>	10	TILING	Routine	putting tiles on the kitchen floor	2016-03-01	00:00:13.000000	2

`SELECT FROM tradesman;`

```
SELECT * FROM tradesman
```

	TRADESMAN_ID	FIRST_NAME	LAST_NAME	SKILL	CONTACT	EQUIPMENT_ID
Edit  Copy  Delete	1	Ronnie	Doyle	Carpenter	0878404260	1
Edit  Copy  Delete	2	Ray	Dunphy	Tiler	0852878541	3
Edit  Copy  Delete	3	Henry	Drew	Plumber	0894937393	2
Edit  Copy  Delete	4	Stella	McGregor	Electrician	0861876459	5
Edit  Copy  Delete	5	Henry	Dunne	Plasterer	0877362952	2
Edit  Copy  Delete	6	HORRID	HENRY	PLUMBING	0866153512	13
Edit  Copy  Delete	7	Stephen	Holt	Carpenter	0898362534	14
Edit  Copy  Delete	8	Gregory	Issaccs	Painter	0878954217	6
Edit  Copy  Delete	9	Larry	Quinn	Carpenter	085842846	15
Edit  Copy  Delete	10	FRANK	MC CANN	PLUMBER	089636464668	16

SELECT \* FROM tenant;

```
SELECT * FROM tenant
```

+ Options

	TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
Edit  Copy  Delete	0	2147483646	Badger	O Malley	757575	badger@gmail.com
Edit  Copy  Delete	1	27854	DUSTIN	DONNELLY	0896904260	ilovedubincoddle@gmail.com
Edit  Copy  Delete	2	39657	GEORGE	WALSH	0872402093	thundercats@yahoo.com
Edit  Copy  Delete	3	10550	MARY	FUREY	0862483815	iamlegend@eircom.ie
Edit  Copy  Delete	4	89386	AOIFE	DOYLE	0851278643	meow@outlook.com
Edit  Copy  Delete	5	42893	TOM	O GARA	08767672841	soundman@Hotmail.com
Edit  Copy  Delete	6	615865	Ann	Ward	08728978421	currychips@hotmail.com
Edit  Copy  Delete	7	61545648	Alan	Turing	08735635473	enigma@btinternet.com
Edit  Copy  Delete	8	2147483647	BONO	VOX	01234567	elevation@vertigo.ie
Edit  Copy  Delete	9	608488585	Michael	Walls	085 2789345	m.walls@yahoo.co.uk
Edit  Copy  Delete	10	9959595	Lawrence	Mullen	0879375639	arcobat@eircom.ie
Edit  Copy  Delete	11	5678915	george	Boole	08723478	george@gmail.com
Edit  Copy  Delete	12	214748364	Mary	Murray	0895487325	murray1975@gmail.com
Edit  Copy  Delete	13	289678	Phillip	Lynott	08735635473	emerald@eircom.ie
Edit  Copy  Delete	14	64476527	Jack	Hanna	0802424007	zuzannaefterosaga@outlook.com

## RULE 2: THE GUARANTEED ACCESS RULE

*“Each and every datum (atomic value) in a relational database is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.”*

All data in the SQL database can be accessed directly or put in simple terms, only three items are needed to locate any piece of data, the table name, the primary key of the row and the column name. The fundamental requirement for the guaranteed access rule is that each row in a table should have a unique primary key. To insert a primary key into a table you can type in the following SQL query:

```
“ALTER TABLE `tenant` ADD PRIMARY KEY (`TENANT_ID`);”
```

The primary key was already created in the database for all tables so this SQL query has not been demonstrated. There should be at least one primary key in each table of a database. It is important for each primary key to have a unique entry for each row in a table. If you try to insert a row with the same primary key as another row the action will not be carried out as you cannot put a primary key into the table with the same data as another primary.

This was demonstrated by trying to insert a row into my database by putting a TENANT ID (the primary key for the tenant table) with a value of '3' when this existed in the database already. The following SQL query gets an error because the same data cannot be entered for a primary key:

```
“INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL) VALUES ('3', '', 'Nadia', 'Baouche', '087348967', 'fortyacres@gmail.com');”
```



**Error**

SQL query:

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL)
VALUES ('3', '', 'Nadia', 'Baouche', '087348967', 'fortyacres@gmail.com')
```






















MySQL said: ?

#1062 - Duplicate entry '3' for key 'PRIMARY'

Without a unique primary key, you can retrieve some data, but maybe not what you really wanted. It is not necessary that each row primary key should be unique, however, if it is not, the guaranteed access rule will not hold true. A Database Management System without primary keys cannot comply with this rule, therefore it is important that a relational Database Management System supports the concept of primary keys.

This rule can be proved by “logically accessing the database by resorting to a combination of table name, primary key value and column name”. This rule was demonstrated simply by typing a SQL query to select one column from a table using table name, primary key, and column name. The SQL query typed was as follows: *“SELECT TENANT\_ID FROM tenant”*.



SELECT TENANT_ID FROM tenant				
				TENANT_ID
<input type="checkbox"/>		Edit	 Copy	 Delete 15
<input type="checkbox"/>		Edit	 Copy	 Delete 3
<input type="checkbox"/>		Edit	 Copy	 Delete 1
<input type="checkbox"/>		Edit	 Copy	 Delete 2
<input type="checkbox"/>		Edit	 Copy	 Delete 5
<input type="checkbox"/>		Edit	 Copy	 Delete 27
<input type="checkbox"/>		Edit	 Copy	 Delete 28

The same rule was then demonstrated by selecting information from two table using table names, column names and primary keys. The SQL query which was typed was as follows:

*"SELECT FIRST NAME, LAST NAME, ADDRESS, POSTCODE*

*FROM property, tenant*

WHERE property.TENANT\_ID = tenant.TENANT\_ID;"

```
SELECT FIRST_NAME, LAST_NAME, ADDRESS, POSTCODE FROM property, tenant WHERE property.TENANT_ID = tenant.TENANT_ID
```

☐ Show all | Number of rows: 25  | Filter rows:

+ Options			
FIRST_NAME	LAST_NAME	ADDRESS	POSTCODE
Lawrence	Mullen	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65
Michael	Walls	54 Hazle Grove, Clondalkin	D04 5N 807
TOM	O GARA	54 Sandymount Avenue	D04 KN85
GEORGE	WALSH	89 Donnybrook Avenue	D04 N2E9
Imelda	May	THE SOUTH DOCK 18 BRIDGE STREET DUBLIN 4	D04 N824
MARY	FUREY	85 Rathmines Road Lower	D06 P2N5
DUSTIN	DONNELLY	22 Belgrave Square South	D06 W1D7
AOIFE	DOYLE	75 Lombard Street East	D08 F6W2
Mary	Murray	75 RUTLAND GROVE, DUBLIN 12	D12 A2B9
Nadine	Burke	32 MAPLEWOOD CLOSE DUBLIN 24	D24 WN33
Sarah	McCoole	24 MAPLEWOOD LAWN DUBLIN 24	D24 WN42

Another way the rule was proved was when data was selected from multiple tables using table names, column names and primary keys. The SQL query typed was as follows:

```
SELECT tenant.TENANT_ID, FIRST_NAME, LAST_NAME, ADDRESS, POSTCODE, repairs.REPAIR_ID,
DETAILS
```

```
FROM property, tenant, repairs
```

```
WHERE property.TENANT_ID = tenant.TENANT_ID
```

```
AND property.REPAIR_ID = repairs.REPAIR_ID;"
```

```
SELECT tenant.TENANT_ID, FIRST_NAME, LAST_NAME, ADDRESS, POSTCODE, repairs.REPAIR_ID, DETAILS FROM
property, tenant, repairs WHERE property.TENANT_ID = tenant.TENANT_ID AND property.REPAIR_ID =
repairs.REPAIR_ID
```

TENANT_ID	FIRST_NAME	LAST_NAME	ADDRESS	POSTCODE	REPAIR_ID	DETAILS
10	Lawrence	Mullen	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	10	putting tiles on the kitchen floor
9	Michael	Walls	54 Hazle Grove, Clondalkin	D04 5N 807	5	Replace toilet cistern
5	TOM	O GARA	54 Sandymount Avenue	D04 KN85	3	Repairs to internal walls
2	GEORGE	WALSH	89 Donnybrook Avenue	D04 N2E9	5	Replace toilet cistern
16	Imelda	May	THE SOUTH DOCK 18 BRIDGE STREET DUBLIN 4	D04 N824	7	Painting of the outside wall of house
3	MARY	FUREY	85 Rathmines Road Lower	D06 P2N5	4	Replace hall door
1	DUSTIN	DONNELLY	22 Belgrave Square South	D06 W1D7	1	Leak under sink unit
4	AOIFE	DOYLE	75 Lombard Street East	D08 F6W2	2	Smoking fuse board
12	Mary	Murray	75 RUTLAND GROVE, DUBLIN 12	D12 A2B9	6	Replace damp skirting board
21	Nadine	Burke	32 MAPLEWOOD CLOSE DUBLIN 24	D24 WN33	8	construction of wooden fence in front garden

To prove that you need the primary key to carry out this task the primary keys were removed from the query and the wrong data came up in the tables. The following SQL query was typed into the console:

```
"SELECT tenant.TENANT_ID, FIRST_NAME, LAST_NAME, ADDRESS, POSTCODE, repairs.REPAIR_ID,
DETAILS
```

```
FROM property, tenant, repairs;"
```

```
SELECT tenant.TENANT_ID, FIRST_NAME, LAST_NAME, ADDRESS, POSTCODE,
repairs.REPAIR_ID, DETAILS FROM property, tenant, repairs
```

TENANT_ID	FIRST_NAME	LAST_NAME	ADDRESS	POSTCODE	REPAIR_ID	DETAILS
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	1	Leak under sink unit
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	2	Smoking fuse board
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	3	Repairs to internal walls
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	4	Replace hall door
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	5	Replace toilet cistern
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	6	Replace damp skirting board
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	7	Painting of the outside wall of house
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	8	construction of wooden fence in front garden
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	9	HOUSE HAS FLOODED
0	Badger	O Malley	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	A97 NX65	10	putting tiles on the kitchen floor
0	Badger	O Malley	54 Hazle Grove, Clondalkin	D04 5N 807	1	Leak under sink unit
0	Badger	O Malley	54 Hazle Grove, Clondalkin	D04 5N 807	2	Smoking fuse board
0	Badger	O Malley	54 Hazle Grove, Clondalkin	D04 5N 807	3	Repairs to internal walls
0	Badger	O Malley	54 Hazle Grove, Clondalkin	D04 5N 807	4	Replace hall door
0	Badger	O Malley	54 Hazle Grove, Clondalkin	D04 5N 807	5	Replace toilet cistern
0	Badger	O Malley	54 Hazle Grove, Clondalkin	D04 5N 807	6	Replace damp skirting board
0	Badger	O Malley	54 Hazle Grove, Clondalkin	D04 5N 807	7	Painting of the outside wall of house
0	Badger	O Malley	54 Hazle Grove, Clondalkin	D04 5N 807	8	construction of wooden fence in front garden

The wrong information came up as the primary key was removed from the SQL query.

### RULE 3: SYSTEMATIC OF NULL VALUES

*“Null values (distinct from empty character string or a string of blank characters or any other number) are supported in a fully relational DBMS for representing missing information in a systematic way, independent of data type.”*

A NULL value means the data in a particular field is unknown. The data may be missing or inapplicable, however, it is unacceptable to use a numerical zero or blank space character to represent missing data. Primary keys cannot be NULL. To demonstrate the way this should not be done two rows were inserted into a table with blank space characters in a column in each table. A record was inserted into the tenant table but left the register number blank and this display's as '0' in the database. Also a row was inserted into the tenant table but left the contact number blank. This is the way it should not be done and breaks the systematic of null values rule. The SQL queries entered were as follows:

*“INSERT INTO tenant (TENANT\_ID, REGISTER\_NUMBER, FIRST\_NAME, LAST NAME, CONTACT, EMAIL) VALUES ('31', ' ', 'Han', 'Solo', '089458765', 'milleniumfalcon@starwars.com');”*

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL) VALUES ('31', ' ', 'Han', 'Solo', '089458765', 'milleniumfalcon@starwars.com')
```

<input type="checkbox"/>	Edit  Copy  Delete	30	85454	Adam	Clayton	08778854858	lemon1993@yahoo.com
<input type="checkbox"/>	Edit  Copy  Delete	31	0	Han	Solo	089458765	milleniumfalcon@starwars.com

*“INSERT INTO tenant (TENANT\_ID, REGISTER\_NUMBER, FIRST\_NAME, LAST NAME, CONTACT, EMAIL) VALUES ('32','68785858','Luke','Skywalker','', 'deathstar@starwars.com');”*

✓ 1 row inserted. (Query took 0.0873 seconds.)

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL) VALUES ('32','68785858','Luke','Skywalker','', 'deathstar@starwars.com')
```

<input type="checkbox"/>	Edit  Copy  Delete	30	85454	Adam	Clayton	08778854858	lemon1993@yahoo.com
<input type="checkbox"/>	Edit  Copy  Delete	31	0	Han	Solo	089458765	milleniumfalcon@starwars.com
<input type="checkbox"/>	Edit  Copy  Delete	32	68785858	Luke	Skywalker		deathstar@starwars.com


Instead of the results coming up as '0' or ' ' (blank character space) I want a record to come up as NULL if there is no value in it. To add the NULL values to a column you have to alter the table so it accepts NULL values. If you add a NULL value to a record where it has not altered the column to accept NULL values then phpadmin will not accept the NULL value in the query. To demonstrate this the following SQL query was typed into the console:

*“INSERT INTO property (POSTCODE, TENANT\_ID, ADDRESS, REPAIR\_ID) VALUES ('D6 WH49','6',NULL,'2');”*



But when this was typed into the console the following error came up:

```
INSERT INTO property (POSTCODE, TENANT_ID, ADDRESS, REPAIR_ID)
VALUES
('D6 WH49', '6', NULL, '2')
```

**MySQL said:** 

#1048 - Column 'ADDRESS' cannot be null

To alter the table so that a column data type can be changed to accept a NULL value the following was typed into the console:

*"ALTER TABLE `tenant` CHANGE `REGISTER\_NUMBER` `REGISTER\_NUMBER` INT(20) NULL;"*

*"ALTER TABLE `tenant` CHANGE `CONTACT` `CONTACT` VARCHAR(20) CHARACTER SET latin1 COLLATE latin1\_swedish\_ci NULL;"*

```
ALTER TABLE `tenant` CHANGE `REGISTER_NUMBER` `REGISTER_NUMBER` INT(20) NULL;
```

The two rows can now be updated and a NULL value can be added to the blank fields.

```
UPDATE `tenant` SET `REGISTER_NUMBER` = NULL WHERE `tenant`.`TENANT_ID` = 31
```

```
UPDATE `tenant` SET `CONTACT` = NULL WHERE `tenant`.`TENANT_ID` = 32;
```

	TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
e	31	NULL	Han	Solo	089458765	milleniumfalcon@starwars.com
e	32	68785858	Luke	Skywalker	NULL	deathstar@starwars.com

A new record can be inserted into the database with NULL values in it. This is demonstrated by typing in the following SQL query:

*"INSERT INTO tenant (TENANT\_ID, REGISTER\_NUMBER, FIRST\_NAME, LAST NAME, CONTACT, EMAIL) VALUES ('33', NULL, 'FRODO', 'BAGGINS', NULL, 'theshire@onering.com');"*

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL) VALUES ('33', NULL, 'FRODO', 'BAGGINS', NULL, 'theshire@onering.com' )
```

TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
31	NULL	Han	Solo	089458765	milleniumfalcon@starwars.com
32	68785858	Luke	Skywalker	NULL	deathstar@starwars.com
33	NULL	FRODO	BAGGINS	NULL	theshire@onering.com

A primary key cannot contain a NULL values, to prove this the primary key column to accept NULL values was changed to try and accept NULL values. To demonstrate this the TENANT\_ID column was changed to accept NULL values by typing the following SQL query:

```
ALTER TABLE 'tenant' CHANGE 'TENANT_ID' 'TENANT_ID' INT(20) NULL;
```

```
ALTER TABLE `tenant` CHANGE `TENANT_ID` `TENANT_ID` INT(20) NULL;
```

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	TENANT_ID 	int(20)			No	None	

The SQL query is accepted by phpadmin but in the Null column in the structure of the tenant table tells us that NULL has not been set.


To demonstrate that NULL value has not been set I typed in the following SQL query:

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL)
VALUES (NULL, '55885', 'GEORGE', 'LUCAS', '089458765', 'milleniumfalcon@starwars.com');
```

## Error

SQL query:

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL)
VALUES (NULL, '55885', 'GEORGE', 'LUCAS', '089458765', 'milleniumfalcon@starwars.com')
```

MySQL said: 

```
#1048 - Column 'TENANT_ID' cannot be null
```

An error came up because a NULL value cannot be used in a primary key.

## RULE 4: ACTIVE ONLINE CATALOG BASED ON THE RELATIONAL MODEL

“The data base description is represented at the logical level in the same way as ordinary data, so that authorised users can apply the same relational language to its interrogation as they apply to regular data.”

This rule tells us that we must be able to describe the structure of the database by looking at the metadata (data about the data). The metadata or structure of my database was accessed by typing the following SQL query or you can view it by viewing the data dictionary:

“SELECT \* FROM INNODB\_SYS\_TABLES;”

98	south_dublin_housing_council_repairs/equipment	1	9	84	Antelope	Compact	0
102	south_dublin_housing_council_repairs/property	1	7	88	Antelope	Compact	0
103	south_dublin_housing_council_repairs/repairs	1	10	89	Antelope	Compact	0
100	south_dublin_housing_council_repairs/tenant	1	9	86	Antelope	Compact	0
105	south_dublin_housing_council_repairs/tradesman	1	9	91	Antelope	Compact	0

The structure or metadata of the equipment table was accessed by typing the following SQL query:

“SELECT \* FROM INNODB\_SYS\_TABLES WHERE TABLE\_ID = ‘98’;”

The same SQL query can be typed into the console to access the other tables in the database, but you just have to change the TABLE\_ID value that is relevant to that table.



The screenshot shows a database management interface. At the top, a SQL query is entered: `SELECT * FROM `INNODB_SYS_TABLES` WHERE TABLE_ID='98'`. Below the query bar, there are controls for displaying the results: a checkbox for "Show all", a "Number of rows:" dropdown set to "25", and a "Filter rows:" search box. The results are displayed in a table with the following columns: TABLE\_ID, NAME, FLAG, N\_COLS, SPACE, FILE\_FORMAT, ROW\_FORMAT, and ZIP\_PAGE\_SIZE. The first row of data shows the metadata for the 'south\_dublin\_housing\_council\_repairs/equipment' table.

TABLE_ID	NAME	FLAG	N_COLS	SPACE	FILE_FORMAT	ROW_FORMAT	ZIP_PAGE_SIZE
98	south_dublin_housing_council_repairs/equipment	1	9	84	Antelope	Compact	0

The metadata of a table within a database can be changed by using the ALTER TABLE clause on a table to change what data has been entered in that table.

## RULE 5: THE COMPREHENSIVE DATA SUBLANGUAGE RULE

*“A relational system may support several languages and various modes of terminal use (for example, fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well defined syntax, as character strings and that is comprehensive in supporting all of the following items: data definition, view definition, data manipulation(interactive and by program), integrity constraints, transaction boundaries (begin, commit and rollback).”*

This rule defines a requirement for a language which can maintain database structural elements, modify and retrieve data. The main language that is used in the South Dublin Housing Council Repairs database is SQL (Structured Query Language) and this rule has been demonstrated through this language. I have also used PHP(PHP: Hypertext Preprocessor) to demonstrate this rule in the website which is linked to the database via PHP. The database can be updated from the website via the php insert statement and the information can be selected from the database to be displayed on the website via the php select statement.

Here is an example of php code used to insert data into the database:

```
“mysql_select_db("south_dublin_housing_council_repairs", $con);

$sql="INSERT INTO tradesman (TRADESMAN_ID, FIRST_NAME, LAST_NAME, SKILL, CONTACT,
EQUIPMENT_ID)
```

`VALUES`

```
('$_POST[TRADESMAN_ID]', '$_POST[FIRST_NAME]', '$_POST[LAST_NAME]', '$_POST[SKILL]',
'$_POST[CONTACT]', '$_POST[EQUIPMENT_ID]');"”
```

This rule has been demonstrated by using PHP to carryout data manipulation such as creating new records and updating them but also can be used in data retrieval transactions by the client. The view definition is being used as clients can use the PHP select statement to view certain data from the database.

The SQL or Structured Query Language is the main type of language used to carry out tasks in the database. The comprehensive data sublanguage rule can be proven to work by typing in the following SQL queries into the console:

Being able to create a table with SQL is one way of demonstrating this rule.

```
“CREATE TABLE housing_issue (
HOUSE_ID int,
ADDRESS varchar(255),
TYPE varchar(255),
COMPLAINT varchar(255);”
```

```
CREATE TABLE housing_issue ( HOUSE_ID int, ADDRESS varchar(255), TYPE
varchar(255), COMPLAINT varchar(255) )
```

Then alter the table by adding a primary key.



`"ALTER TABLE `housing_issue` ADD PRIMARY KEY(`HOUSE_ID`);"`

```
ALTER TABLE `housing_issue` ADD PRIMARY KEY(`HOUSE_ID`);
```

#	Name	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	1	HOUSE_ID	int(11)		No	None	

Data is then inserted into the table.

`"INSERT INTO housing_issue (HOUSE_ID, ADDRESS, TYPE, COMPLAINT)`

`VALUES ('45','543 Spencer Street', 'leak', 'leak coming down through the ceiling');"`

```
INSERT INTO housing_issue (HOUSE_ID, ADDRESS, TYPE, COMPLAINT) VALUES ('45','543 Spencer Street','leak','leak coming down through the ceiling')
```

	HOUSE_ID	ADDRESS	TYPE	COMPLAINT
<input type="checkbox"/> Edit  Copy  Delete	45	543 Spencer Street	leak	leak coming down through the ceiling

Add a column to the table.

`"ALTER TABLE housing_issue`

`ADD COLUMN AREA VARCHAR(30);"`

```
ALTER TABLE housing_issue ADD COLUMN AREA VARCHAR(30)
```

HOUSE_ID	ADDRESS	TYPE	COMPLAINT	AREA
45	543 Spencer Street	leak	leak coming down through the ceiling	NULL

The table was updated by changing a row on the TYPE column to be changed to "leaking roof" and the AREA column to be "Dublin 4".

```
UPDATE housing_issue SET TYPE = 'leaking roof' WHERE HOUSE_ID = '45'
```

```
UPDATE housing_issue SET AREA = 'DUBLIN 4' WHERE HOUSE_ID = '45'
```

HOUSE_ID	ADDRESS	TYPE	COMPLAINT	AREA
45	543 Spencer Street	leaking roof	leak coming down through the ceiling	DUBLIN 4

Then a row was deleted using the DELETE clause, by entering the following SQL code into the console:

```
"DELETE FROM housing_issue WHERE AREA = 'DUBLIN 4';"
```

✓ 1 row affected. (Query took 0.0657 seconds.)

```
DELETE FROM housing_issue WHERE AREA = 'DUBLIN 4'
```

Then to get rid of the table I used the DROP clause

```
DROP TABLE housing_issue
```

❗ #1146 - Table  
'south\_dublin\_housing\_council\_repairs.housing\_issue'  
doesn't exist

## RULE 6: THE VIEW UPDATING RULE

*“All views that are theoretically updateable are also updateable by the system.”*

A view is a table that has been created by a SQL query. The view is like a virtual table which can take elements from multiple columns from one or more tables and display them in a separate view table. When the view table is created it is stored in the metadata of the database. A view table can also be updated, and the Data Base Management System should be able to handle the update, and if it is updated properly then the base table should be able to update also.

It was first demonstrated that a view table could be created in the database by typing in the following SQL code into the console:

First a view table was created from one table in the database and some of the columns in that table.

*“CREATE VIEW tenant\_details AS SELECT TENANT\_ID, FIRST\_NAME, LAST\_NAME FROM tenant;”*

The screenshot shows a database management tool interface. At the top, the title bar indicates the server is 127.0.0.1, the database is south\_dublin\_housing\_council\_repairs, and the current view is tenant\_details. Below the title bar is a toolbar with buttons for Browse, Structure, SQL, Search, Insert, Export, and Privileges. The main area displays a table with the following data:

	TENANT_ID	FIRST_NAME	LAST_NAME
<input type="checkbox"/> Edit Copy Delete	0	Badger	O Malley
<input type="checkbox"/> Edit Copy Delete	1	DUSTIN	DONNELLY
<input type="checkbox"/> Edit Copy Delete	2	GEORGE	WALSH
<input type="checkbox"/> Edit Copy Delete	3	MARY	FUREY
<input type="checkbox"/> Edit Copy Delete	4	AOIFE	DOYLE
<input type="checkbox"/> Edit Copy Delete	5	TOM	O GARA
<input type="checkbox"/> Edit Copy Delete	6	Ann	Ward
<input type="checkbox"/> Edit Copy Delete	7	Alan	Turing
<input type="checkbox"/> Edit Copy Delete	8	BONO	VOX
<input type="checkbox"/> Edit Copy Delete	9	Michael	Walls
<input type="checkbox"/> Edit Copy Delete	10	Lawrence	Mullen

A SELECT statement was then carried out on the view.

*“SELECT \* FROM ‘tenant\_details’*

*WHERE LAST\_NAME=‘DONNELLY’;”*

```
SELECT * FROM `tenant_details` WHERE LAST_NAME='DONNELLY'
```

☐ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

	TENANT_ID	FIRST_NAME	LAST_NAME
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	DUSTIN	DONNELLY

"SELECT \* FROM 'tenant\_details' WHERE TENANT\_ID = '13';"

```
SELECT * FROM `tenant_details` WHERE TENANT_ID = '13'
```

☐ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

	TENANT_ID	FIRST_NAME	LAST_NAME
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	13	Phil	Lynott

You can update your view and one base table in a view.

"UPDATE 'tenant\_details' SET FIRST\_NAME='PHIL' WHERE TENANT\_ID = '13';"

```
UPDATE `tenant_details` SET FIRST_NAME='PHIL' WHERE TENANT_ID = '13'
```

<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	13	PHIL	Lynott
---	----	------	--------

When the user inserts information into a view it also updates the main table. To insert information into the view input the following SQL query into the database:

```
INSERT INTO tenant_details (TENANT_ID, FIRST_NAME, LAST_NAME) VALUES ('60','Pat', 'LYONS');
```



✓ 1 row inserted. (Query took 0.0940 seconds.)

```
INSERT INTO tenant_details (TENANT_ID, FIRST_NAME, LAST_NAME) VALUES ('60', 'Pat', 'LYONS')
```

Database: south\_dublin\_housing\_council\_repairs » View: tei

	TENANT_ID	FIRST_NAME	LAST_NAME
Delete	50	OBI WAN	KENOBI
Delete	60	Pat	LYONS

Inserting information into the View also updates the tenant table.

127.0.0.1 » Database: south\_dublin\_housing\_council\_repairs » Table: tenant

TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
50	2858	OBI WAN	KENOBI	089464747	theforceawakens@deathstar.com
60	NULL	Pat	LYONS	NULL	NULL

Then try creating a view made out of multiple tables.

*“CREATE VIEW tenant\_details\_and\_repairs*

*AS SELECT tenant.TENANT\_ID, FIRST\_NAME, LAST\_NAME, ADDRESS, repairs.REPAIR\_ID, TYPE, DETAILS*

*FROM tenant, property, repairs*

*WHERE tenant.TENANT\_ID = property.TENANT\_ID AND property.REPAIR\_ID = repairs.REPAIR\_ID;”*

`SELECT * FROM 'tenant_details_and_repairs'`

[Edit inline][Edit][Explain SQL][Create PHP code][Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options		TENANT_ID	FIRST_NAME	LAST_NAME	ADDRESS	REPAIR_ID	TYPE	DETAILS
<input type="checkbox"/>	Edit Copy Delete	10	Lawrence	Mullen	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	10	TILING	putting tiles on the kitchen floor
<input type="checkbox"/>	Edit Copy Delete	9	Michael	Walls	54 Hazle Grove, Clondalkin	5	Plumbing	Replace toilet cistern
<input type="checkbox"/>	Edit Copy Delete	5	TOM	O GARA	54 Sandymount Avenue	3	Plastering	Repairs to internal walls
<input type="checkbox"/>	Edit Copy Delete	2	GEORGE	WALSH	89 Donnybrook Avenue	5	Plumbing	Replace toilet cistern
<input type="checkbox"/>	Edit Copy Delete	16	Imelda	May	THE SOUTH DOCK 18 BRIDGE STREET DUBLIN 4	7	Painting	Painting of the outside wall of house
<input type="checkbox"/>	Edit Copy Delete	3	MARY	FUREY	85 Rathmines Road Lower	4	Carpentry	Replace hall door
<input type="checkbox"/>	Edit Copy Delete	1	DUSTIN	DONNELLY	22 Belgrave Square South	1	Plumbing	Leak under sink unit
<input type="checkbox"/>	Edit Copy Delete	4	AOIFE	DOYLE	75 Lombard Street East	2	Electric	Smoking fuse board
<input type="checkbox"/>	Edit Copy Delete	12	Mary	Murray	75 RUTLAND GROVE, DUBLIN 12	6	Woodwork	Replace damp skirting board

Then try and update the view table made up out of multiple tables:

```
UPDATE `tenant_details_and_repairs` SET TYPE='carpentry'
```

```
WHERE TENANT_ID = '20';
```

```
UPDATE `tenant_details_and_repairs` SET TYPE='carpentry'
```

```
WHERE TENANT_ID = '5';
```

✓ 1 row affected. (Query took 0.0938 seconds.)

```
UPDATE `tenant_details_and_repairs` SET TYPE='carpentry' WHERE TENANT_ID = '5'
```

	TENANT_ID	FIRST_NAME	LAST_NAME	ADDRESS	REPAIR_ID	TYPE	DETAILS
<input type="checkbox"/> Edit <input type="plus"/> Copy <input type="minus"/> Delete	10	Lawrence	Mullen	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	10	TILING	putting tiles on the kitchen floor
<input type="checkbox"/> Edit <input type="plus"/> Copy <input type="minus"/> Delete	9	Michael	Walls	54 Hazle Grove, Clondalkin	5	Plumbing	Replace toilet cistern
<input type="checkbox"/> Edit <input type="plus"/> Copy <input type="minus"/> Delete	5	TOM	O GARA	54 Sandymount Avenue	3	carpentry	Repairs to internal walls

The view updates when a single column is selected from one table.

But the view does not update when multiple columns from multiple tables are updated by adding a row to a view. When the following SQL query was typed into the console it got an error which told that it cannot modify more than one base table through a join view:

```
INSERT INTO tenant_details_and_repairs (TENANT_ID, FIRST_NAME, LAST_NAME, ADDRESS,  
REPAIR_ID, TYPE, DETAILS)  
VALUES ('61','JACKIE', 'KENNY','802 Sunset Avenue','11', 'Plumbing','install heaters into sitting room');
```

## Error

### SQL query:

```
INSERT INTO tenant_details_and_repairs (TENANT_ID, FIRST_NAME, LAST_NAME, ADDRESS, REPAIR_ID, TYPE, D  
VALUES ('61','JACKIE', 'KENNY','802 Sunset Avenue','11', 'Plumbing','install heaters into sitting roo
```

### MySQL said:

```
#1393 - Can not modify more than one base table through a join view  
'south_dublin_housing_council_repairs.tenant_details_and_repairs'
```

The user can also create views of other view tables which are updateable which was proved by typing the following SQL code into the console:

```
CREATE VIEW tenant_details2
```

```
AS SELECT TENANT_ID, FIRST_NAME, LAST_NAME
```

```
FROM tenant_details;
```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0045 seconds.)

```
DROP VIEW tenant_details_and_repairs
```

	TENANT_ID	FIRST_NAME	LAST_NAME
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	DUSTIN	DONNELLY
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	GEORGE	WALSH
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	MARY	FUREY
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	AOIFE	DOYLE
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	TOM	O GARA

But a view of a view table should not be made if that view table is not updateable.

Also if a view does not have the primary keys in it the information will not come up correct. This was proved by tying in the following SQL code into the console:

```
CREATE VIEW tenant_details_and_repairs2
```

```
AS SELECT tenant.TENANT_ID, FIRST_NAME, LAST_NAME, ADDRESS, repairs.REPAIR_ID, TYPE, DETAILS
```

```
FROM tenant, property, repairs;
```

The query was carried out but the wrong information came up on the view table.

```
CREATE VIEW tenant_details_and_repairs2 AS SELECT tenant.TENANT_ID, FIRST_NAME, LAST_NAME, ADDRESS, repairs.REPAIR_ID, TYPE, DETAILS FROM tenant, property, repairs
```

Server: 127.0.0.1 » Database: south\_dublin\_housing\_council\_repairs » View: tenant\_details\_and\_repairs2

Browse

Structure

SQL

Search

Insert

Export

Privileges

Operations

Tracking

+ Options

	TENANT_ID	FIRST_NAME	LAST_NAME	ADDRESS	REPAIR_ID	TYPE	DETAILS
<div><div><div></div><div>Edit</div><div>Copy</div><div>Delete</div></div></div> <td>1</td> <td>DUSTIN</td> <td>DONNELLY</td> <td>ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN</td> <td>1</td> <td>Plumbing</td> <td>Leak under sink unit</td>	1	DUSTIN	DONNELLY	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	1	Plumbing	Leak under sink unit
<div><div><div></div><div>Edit</div><div>Copy</div><div>Delete</div></div></div> <td>1</td> <td>DUSTIN</td> <td>DONNELLY</td> <td>ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN</td> <td>2</td> <td>Electric</td> <td>Smoking fuse board</td>	1	DUSTIN	DONNELLY	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	2	Electric	Smoking fuse board
<div><div><div></div><div>Edit</div><div>Copy</div><div>Delete</div></div></div> <td>1</td> <td>DUSTIN</td> <td>DONNELLY</td> <td>ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN</td> <td>3</td> <td>carpentry</td> <td>Repairs to internal walls</td>	1	DUSTIN	DONNELLY	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	3	carpentry	Repairs to internal walls
<div><div><div></div><div>Edit</div><div>Copy</div><div>Delete</div></div></div> <td>1</td> <td>DUSTIN</td> <td>DONNELLY</td> <td>ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN</td> <td>4</td> <td>Carpentry</td> <td>Replace hall door</td>	1	DUSTIN	DONNELLY	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	4	Carpentry	Replace hall door
<div><div><div></div><div>Edit</div><div>Copy</div><div>Delete</div></div></div> <td>1</td> <td>DUSTIN</td> <td>DONNELLY</td> <td>ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN</td> <td>5</td> <td>Plumbing</td> <td>Replace toilet cistern</td>	1	DUSTIN	DONNELLY	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	5	Plumbing	Replace toilet cistern
<div><div><div></div><div>Edit</div><div>Copy</div><div>Delete</div></div></div> <td>1</td> <td>DUSTIN</td> <td>DONNELLY</td> <td>ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN</td> <td>6</td> <td>Woodwork</td> <td>Replace damp skirting board</td>	1	DUSTIN	DONNELLY	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	6	Woodwork	Replace damp skirting board
<div><div><div></div><div>Edit</div><div>Copy</div><div>Delete</div></div></div> <td>1</td> <td>DUSTIN</td> <td>DONNELLY</td> <td>ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN</td> <td>7</td> <td>Painting</td> <td>painting inside walls of the house</td>	1	DUSTIN	DONNELLY	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	7	Painting	painting inside walls of the house
<div><div><div></div><div>Edit</div><div>Copy</div><div>Delete</div></div></div> <td>1</td> <td>DUSTIN</td> <td>DONNELLY</td> <td>ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN</td> <td>8</td> <td>carpentry</td> <td>construction of wooden fence in front garden</td>	1	DUSTIN	DONNELLY	ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN	8	carpentry	construction of wooden fence in front garden

The user can then Drop the View to get rid of it with the following SQL code:

```
DROP VIEW tenant_details;
```

```
DROP VIEW tenant_details
```



#1146 - Table

'south\_dublin\_housing\_council\_repairs.tenant\_details'  
doesn't exist

*DROP VIEW tenant\_details\_and\_repairs;*

```
DROP VIEW tenant_details_and_repairs
```



#1356 - View

'south\_dublin\_housing\_council\_repairs.tenant\_details2'  
references invalid table(s) or column(s) or function(s) or  
definer/invoke of view lack rights to use them

*DROP VIEW tenant\_details\_and\_repairs2;*

```
DROP VIEW tenant_details_and_repairs2
```



#1146 - Table

'south\_dublin\_housing\_council\_repairs.tenant\_details\_and\_repairs2'  
doesn't exist

*DROP VIEW tenant\_details2;*



#1146 - Table

'south\_dublin\_housing\_council\_repairs.tenant\_details2'  
doesn't exist



## RULE 7: HIGH-LEVEL INSERT, UPDATE, AND DELETE.

*“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.”*

High level means that you can effect multiple rows from multiple tables with a single query. To prove this rule the user must be able to multi-insert data, multi-update data and to multi-delete data in multiple tables and rows in the database rather than just for a single row in a single table.

To prove the high level insert part of the rule I typed the following SQL query into the console:

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL)
VALUES ('35','937847', 'John', 'Grisham','08976543', 'ilovegeneaology@gmail.com'),
('36', '74573', 'Eve', 'Malone','087773789', 'meowtastic@hotmail.com'),
('37', '8764', 'Vera', 'Duckworth', '085337478', 'finalcountdown@europe.com'),
('38', '16274', 'Jack', 'Hackett', '0893646636', 'drinkfeckarse@feckarseindustries.ie');
```

✓ 4 rows inserted. (Query took 0.1321 seconds.)

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT,
EMAIL) VALUES ('35','937847', 'John', 'Grisham','08976543',
'ilovegeneaology@gmail.com'), ('36', '74573', 'Eve', 'Malone','087773789',
'meowtastic@hotmail.com'), ('37', '8764', 'Vera', 'Duckworth', '085337478',
'finalcountdown@europe.com'), ('38', '16274', 'Jack', 'Hackett', '0893646636',
'drinkfeckarse@feckarseindustries.ie')
```

	TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
e	35	937847	John	Grisham	08976543	ilovegeneaology@gmail.com
e	36	74573	Eve	Malone	087773789	meowtastic@hotmail.com
e	37	8764	Vera	Duckworth	085337478	finalcountdown@europe.com
e	38	16274	Jack	Hackett	0893646636	drinkfeckarse@feckarseindustri

To prove the high level update part of the rule I typed the following SQL query into the console:

```
UPDATE tenant
SET FIRST_NAME = 'JOHNNY'
WHERE FIRST_NAME = 'JOHN';
```

✓ 3 rows affected. (Query took 0.0773 seconds.)

```
UPDATE tenant SET FIRST_NAME = 'JOHNNY' WHERE FIRST_NAME = 'JOHN'
```

To prove the high level update part of the rule with multiple tables, more people with the first name JOHNNY will be multi-inserted into the tenant table and the tradesman table, then a multi-update will be used to change all the people with first name JOHNNY to the name JOHN.

There were two records multi-inserted into the tenant table with the first name JOHNNY with the following SQL query:

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL)
VALUES ('39','937', 'JOHNNY', 'LYONS','08976543', 'ilovegeneaology@gmail.com'),
      ('40', '743', 'JOHNNY', 'Malone','087773789', 'meowtastic@hotmail.com');
```

✓ 2 rows inserted. (Query took 0.1215 seconds.)

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL) VALUES
('39','937', 'JOHNNY', 'LYONS','08976543', 'ilovegeneaology@gmail.com'), ('40', '743', 'JOHNNY',
'Malone','087773789', 'meowtastic@hotmail.com')
```

TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
32	68785858	Luke	Skywalker	NULL	deathstar@starwars.com
33	NULL	FRODO	BAGGINS	NULL	theshire@onering.com
34	6346354	Badger	O Malley	087348967	badger@gmail.com
35	937847	JOHNNY	Grisham	08976543	ilovegeneaology@gmail.com
36	74573	Eve	Malone	087773789	meowtastic@hotmail.com
37	8764	JOHNNY	Duckworth	085337478	finalcountdown@europe.com
38	16274	Jack	Hackett	0893646636	drinkfeckarse@feckarseindustri
39	937	JOHNNY	LYONS	08976543	ilovegeneaology@gmail.com
40	743	JOHNNY	Malone	087773789	meowtastic@hotmail.com

There were two records multi-inserted into the tradesman table with the first name JOHNNY with the following SQL query:

```
INSERT INTO tradesman (TRADESMAN_ID, FIRST_NAME, LAST_NAME, SKILL, CONTACT,
EQUIPMENT_ID)
VALUES ('40', 'JOHNNY', 'LYONS','plumbing', '089575757', '10'),
      ('42', 'JOHNNY', 'Malone','woodwork', '087543653', '9');
```

```
INSERT INTO tradesman (TRADESMAN_ID, FIRST_NAME, LAST_NAME, SKILL, CONTACT, EQUIPMENT_ID) VALUES ('40',
'JOHNNY', 'LYONS','plumbing', '089575757', '10'), ('42', 'JOHNNY', 'Malone','woodwork', '087543653',
'9')
```

TRADESMAN_ID	FIRST_NAME	LAST_NAME	SKILL	CONTACT	EQUIPMENT_ID
9	Larry	Quinn	Carpenter	085842846	15
10	John	MC CANN	PLUMBER	089636464668	16
40	JOHNNY	LYONS	plumbing	089575757	10
42	JOHNNY	Malone	woodwork	087543653	9

To multi-update data on multiple rows on one table then type in the following SQL query into the console:

*UPDATE tenant*

*SET FIRST\_NAME = 'JOHN'*

*WHERE FIRST\_ID = 'JOHNNY';*

TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
35	937847	JOHN	Grisham	08976543	ilovegenealogy@
36	74573	Eve	Malone	087773789	meowtastic@hotm
37	8764	JOHN	Duckworth	085337478	finalcountdown@e
38	16274	Jack	Hackett	0893646636	drinkfeckarse@fec
39	937	JOHN	LYONS	08976543	ilovegenealogy@
40	743	JOHN	Malone	087773789	meowtastic@hotm

*UPDATE tradesman SET FIRST\_NAME = 'JOHN'*

*WHERE FIRST\_NAME = 'JOHNNY';*

✓ 2 rows affected. (Query took 0.0736 seconds.)

*UPDATE tradesman SET FIRST\_NAME = 'JOHN' WHERE FIRST\_NAME = 'JOHNNY'*

TRADESMAN_ID	FIRST_NAME	LAST_NAME	SKILL	CONTACT
9	Larry	Quinn	Carpenter	085842
10	John	MC CANN	PLUMBER	089636
40	JOHN	LYONS	plumbing	089575
42	JOHN	Malone	woodwork	087543

To prove the multi-delete part of the rule the following SQL query was typed in the console:

```
DELETE FROM tenant WHERE FIRST_NAME = 'JOHN';
```

✓ 5 rows affected. (Query took 0.0801 seconds.)

```
DELETE FROM tenant WHERE FIRST_NAME = 'JOHN'
```

TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
33	NULL	FRODO	BAGGINS	NULL	theshire@oneri
34	6346354	Badger	O Malley	087348967	badger@gmail.i
36	74573	Eve	Malone	087773789	meowtastic@hc
38	16274	Jack	Hackett	0893646636	drinkfeckarse@

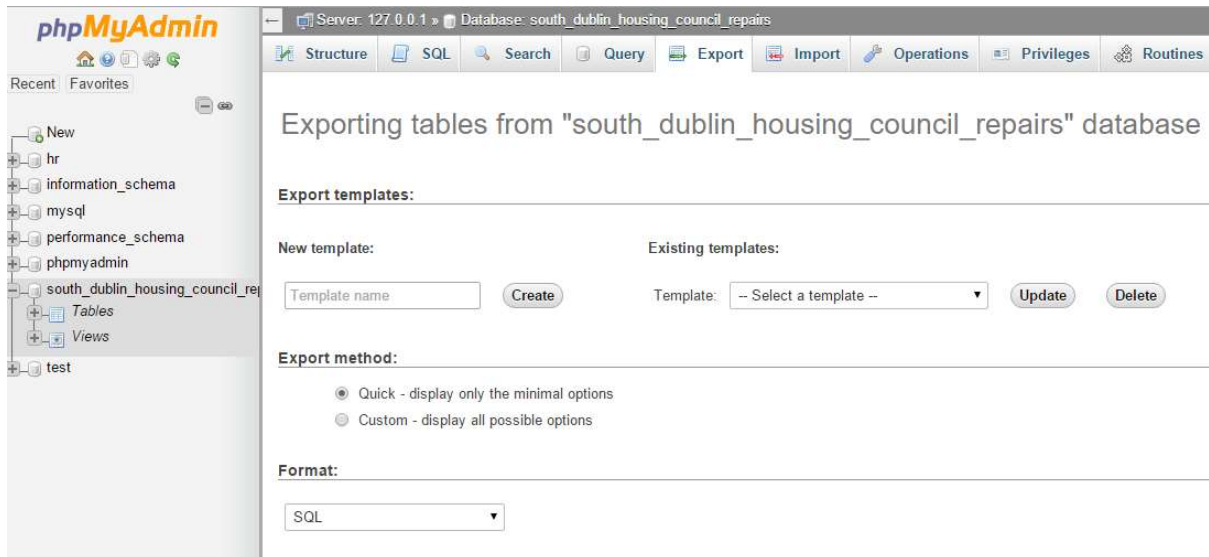


## RULE 8: PHYSICAL DATA INDEPENDENCE

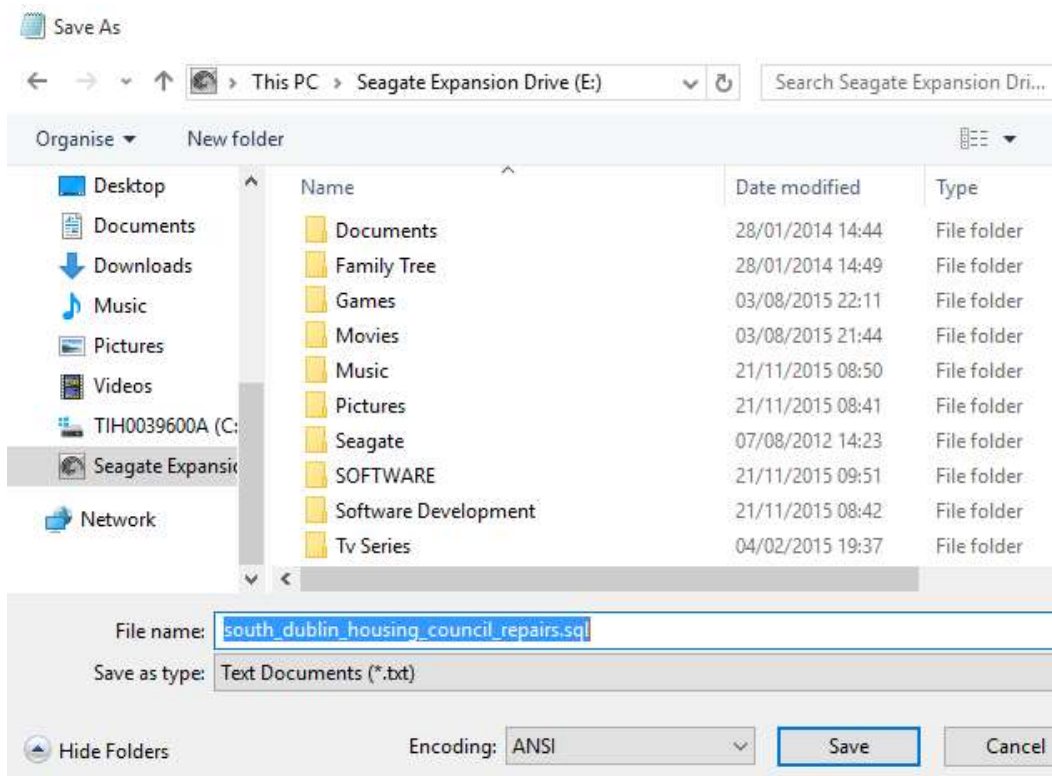
*"Applications programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods."*

The physical data independence rule is demonstrated by exporting your database to another drive or computer and that it works exactly the same as it did on the main user's computer.

This rule was demonstrated by exporting the database to another drive on the computer. The database exported from phpMyAdmin first.



A south\_dublin\_housing\_council\_repairs.sql file was created and saved to an external hard drive.



A new database called southdublin was created in phpmyadmin and the file name on the external harddrive was changed to south\_dublin.sql to avoid confusing the file with the original file. The south\_dublin.sql file was then imported into the southdublin database from the external hard drive.

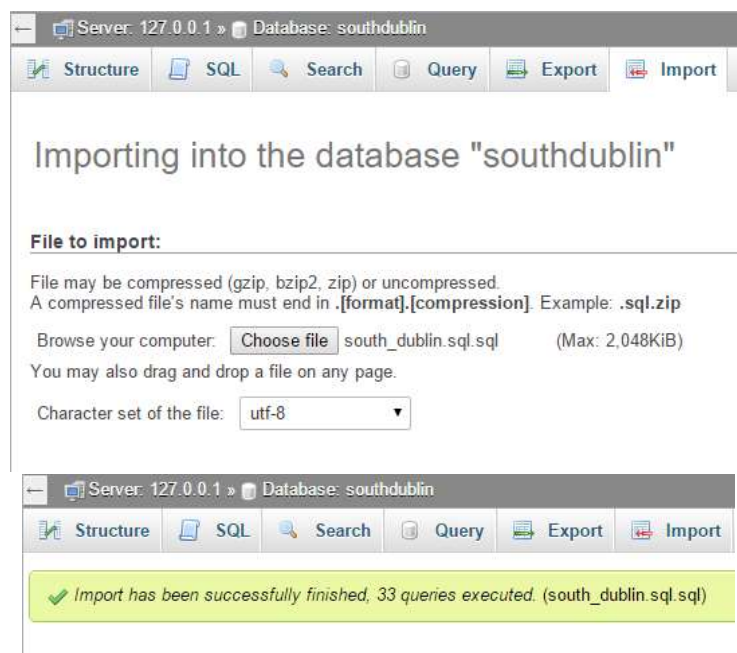
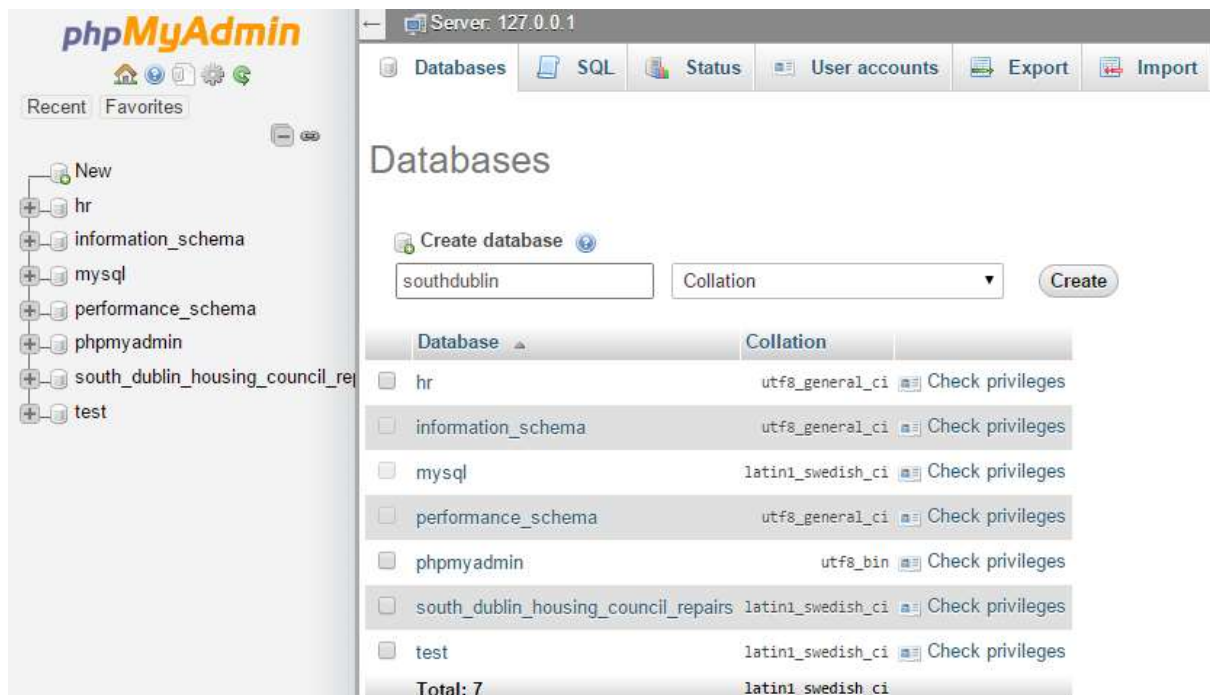
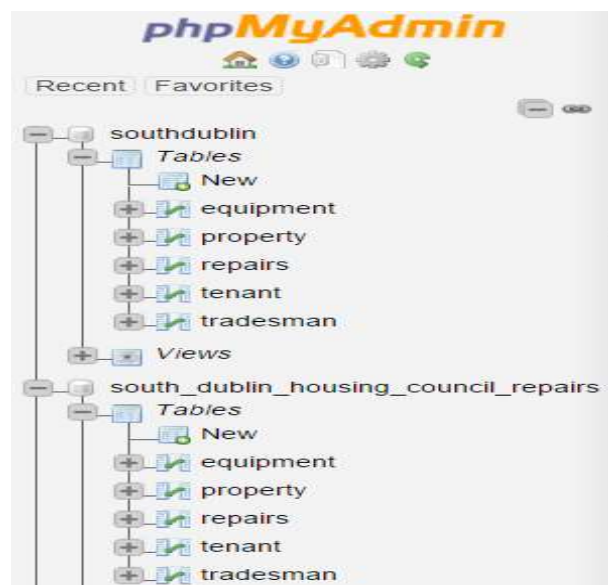


Table	Action	Rows	Type	Collation	Size	Overhead
equipment	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	16	InnoDB	latin1_swedish_ci	16 Kib	-
property	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	11	InnoDB	latin1_swedish_ci	48 Kib	-
repairs	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	10	InnoDB	latin1_swedish_ci	32 Kib	-
tenant	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	35	InnoDB	latin1_swedish_ci	32 Kib	-
tenant_details	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Drop</a> <a href="#">View</a>	~0				-
tenant_details_and_repairs	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Drop</a> <a href="#">View</a>	~0				-
tradesman	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	12	InnoDB	latin1_swedish_ci	32 Kib	-
<b>7 tables</b>	<b>Sum</b>	<b>~84</b>	<b>InnoDB</b>	<b>latin1_swedish_ci</b>	<b>160 Kib</b>	<b>0 B</b>

All the tables in the southdublin database are the same as the tables in the south\_dublin\_housing\_council\_repairs database but have been accessed through a sql file on the external harddrive in separate location to the original file but it still shows the same data.

	TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	27854	DUSTIN	DONNELLY	0896904260	ilovedublincoddle@gmail.com
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	2	39657	GEORGE	WALSH	0872402093	thundercats@yahoo.com
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	10550	MARY	FUREY	0862483815	iamlegend@eircom.ie
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	89386	AOIFE	DOYLE	0851278643	meow@outlook.com
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	5	42893	TOM	O GARA	08767672841	soundman@Hotmail.com
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	6	615865	Ann	Ward	08728978421	currychips@hotmail.com
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	7	61545648	Alan	Turing	08735635473	enigma@btinternet.com



The second way that rule 8 was proved was by exporting the south\_dublin\_housing\_council\_repairs database to another computer via usb. A new user was created to accept external connections on that computer. Then that computer connected remotely to the original computer via the console using the host option to override the default value of localhost (local machine).

The original computer connected to the other computer through the command prompt. The following line in the command prompt demonstrates that the database is working on the on the other computer and the original user can access that database online through the original computer where the database was originally created:

*"mysql -h10.12.22.210 -urepairs\_external -prepairs\_external south\_dublin\_housing\_council\_repairs".*

```
C:\Users\Stephen Fahy>mysql -h10.12.22.210 -urepairs_external -prepairs_external south_dublin_housing_council_repairs
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 72
Server version: 5.6.26 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [south_dublin_housing_council_repairs]> select * from property;
+-----+-----+-----+-----+
| POSTCODE | TENANT_ID | ADDRESS | REPAIR_ID |
+-----+-----+-----+-----+
| A97 NX65 | 10 | ELSINORE 15 COLIEMORE ROAD DALKEY CO. DUBLIN | 10 |
| D04 5N 807 | 9 | 54 Hazle Grove, Clondalkin | 5 |
| D04 KN85 | 5 | 54 Sandymount Avenue | 3 |
| D04 N2E9 | 2 | 89 Donnybrook Avenue | 5 |
| D04 N824 | 16 | THE SOUTH DOCK 18 BRIDGE STREET DUBLIN 4 | 7 |
| D06 P2N5 | 3 | 85 Rathmines Road Lower | 4 |
| D06 W1D7 | 1 | 22 Belgrave Square South | 1 |
| D08 F6W2 | 4 | 75 Lombard Street East | 2 |
| D12 A2B9 | 12 | 75 RUTLAND GROVE, DUBLIN 12 | 6 |
| D24 WN33 | 21 | 32 MAPLEWOOD CLOSE DUBLIN 24 | 8 |
| D24 WN42 | 20 | 24 MAPLEWOOD LAWN DUBLIN 24 | 8 |
+-----+-----+-----+-----+
11 rows in set (0.00 sec)

MySQL [south_dublin_housing_council_repairs]>
```

## RULE 9: LOGICAL DATA INDEPENDENCE

*“Applications programs and terminal activities remain logically unimpaired when Information Preserving changes of any that theoretically permit unimpairment are made to the base tables.*

If a change is made to the logical level (tables, rows, structure) or the base tables, such as adding a new column to one of the tables then data should still be able to be added to the table even though the structure has been changed. For example if I insert information from my website into my database even though the table structure has changed the user should still be able to enter information into the original columns.

I demonstrated this by first adding a column to the equipment table by typing the following SQL query into the console:

```
ALTER TABLE tenant ADD POSTCODE VARCHAR(10) NOT NULL;
```

ALTER TABLE tenant ADD POSTCODE VARCHAR(10) NOT NULL						
TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL	POSTCODE
1	27854	DUSTIN	DONNELLY	0896904260	ilovedubincoddle@gmail.com	
2	39657	GEORGE	WALSH	0872402093	thundercats@yahoo.com	
3	10550	MARY	FUREY	0862483815	iamlegend@eircom.ie	
4	89386	AOIFE	DOYLE	0851278643	meow@outlook.com	
5	42893	TOM	O GARA	08767672841	soundman@Hotmail.com	
6	615865	Ann	Ward	08728978421	currychips@hotmail.com	
7	61545648	Alan	Turing	08735635473	enigma@btinternet.com	
8	2147483647	BONO	VOX	01234567	elevation@vertigo.ie	
9	608488585	Michael	Walls	085 2789345	m.walls@yahoo.co.uk	

I should be able to insert a new row into the tenant table with no problems occurring. I inserted two new rows into the table by typing the following SQL query into the console:

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL, POSTCODE)
```

```
VALUES ('39','937', 'JOHNNY', 'LYONS','08976543', 'ilovegeneaology@gmail.com', ''),  
('40', '743', 'JOHNNY', 'VEGAS','087773789', 'meowtastic@hotmail.com', '');
```

✓ 2 rows inserted. (Query took 0.1002 seconds.)

```
INSERT INTO tenant (TENANT_ID, REGISTER_NUMBER, FIRST_NAME, LAST_NAME, CONTACT, EMAIL, POSTCODE) VALUES  
('39','937', 'JOHNNY', 'LYONS','08976543', 'ilovegeneaology@gmail.com', ''), ('40', '743', 'JOHNNY',  
'VEGAS','087773789', 'meowtastic@hotmail.com', '')
```



TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL	POSTCODE
36	74573	Eve	Malone	087773789	meowtastic@hotmail.com	
38	16274	Jack	Hackett	0893646636	drinkfeckarse@feckarseindustri	
39	937	JOHNNY	LYONS	08976543	ilovegenealogy@gmail.com	
40	743	JOHNNY	VEGAS	087773789	meowtastic@hotmail.com	

I was still able to input information into the database from my website even though I added a new column to the table and changed the structure of my database.

When a tenant sends in a request for repair the tenant is checked against a valid register number. If they are a registered tenant they may be entitled to a repair to their property. If they are a registered tenant then enter their details into the database.

Type the tenant details into the following fields to update our online database.

**TENANT ID:**

50

**REGISTER NUMBER:**

2858

**FIRST NAME:**

OBI WAN

**LAST NAME:**

KENOBI

**CONTACT:**

089464747

**EMAIL:**

theforceawakens@deathst

Submit

127.0.0.1 » Database: south_dublin_housing_council_repairs » Table: tenant						
Structure	SQL	Search	Insert	Export	Import	Privileges
TENANT_ID	REGISTER_NUMBER	FIRST_NAME	LAST_NAME	CONTACT	EMAIL	POSTCODE
e 40	743	JOHNNY	VEGAS	087773789	meowtastic@hotmail.com	
e 50	2858	OBI WAN	KENOBI	089464747	theforceawakens@deathstar.com	

If the user creates a view of a base table and then alters the base table by deleting a column or two. The user should still be able to update the remaining columns via a view. To demonstrate this rule the 'housing\_issue' table was created again, data was added to it and a view was made of it.

First a table was created through sql statements.

✓ 3 rows inserted. (Query took 0.0610 seconds.)

```
INSERT INTO housing_issue (HOUSE_ID, ADDRESS, TYPE, COMPLAINT) VALUES ('45','543 Spencer Street','leak','leak coming down through the ceiling'), ('46','544 Spencer Street','burstpipe','burst pipe in the kitchen'), ('47','545 Spencer Street','rodent infestation','there are rats in the house')
```

HOUSE_ID	ADDRESS	TYPE	COMPLAINT
45	543 Spencer Street	leak	leak coming down through the ceiling
46	544 Spencer Street	burstpipe	burst pipe in the kitchen
47	545 Spencer Street	rodent infestation	there are rats in the house

Then a view of this table was created by typing CREATE VIEW in SQL:

```
CREATE VIEW housing_issue_view AS SELECT HOUSE_ID, ADDRESS, TYPE, COMPLAINT FROM housing_issue;
```

```
CREATE VIEW housing_issue_view AS SELECT HOUSE_ID, ADDRESS, TYPE, COMPLAINT FROM housing_issue
```

base: south\_dublin\_housing\_council\_repairs » View: housing\_issue\_view

HOUSE_ID	ADDRESS	TYPE	COMPLAINT
45	543 Spencer Street	leak	leak coming down through the ceiling
46	544 Spencer Street	burstpipe	burst pipe in the kitchen
47	545 Spencer Street	rodent infestation	there are rats in the house

Then to demonstrate the rule the COMPLAINT column was deleted from the base table.

```
ALTER TABLE housing_issue
```

```
DROP COLUMN COMPLAINT;
```

```
ALTER TABLE housing_issue DROP COLUMN COMPLAINT
```

HOUSE_ID	ADDRESS	TYPE
45	543 Spencer Street	leak
46	544 Spencer Street	burstpipe
47	545 Spencer Street	rodent infestation

After the COMPLAINT column has been deleted from the base table housing\_issue try and update the remaining columns by updating the view.

```
INSERT INTO housing_issue_view (HOUSE_ID, ADDRESS, TYPE, COMPLAINT)
VALUES ('48','937 HAZEL GROVE', 'bad smell', 'bad smell coming from sewers'),
       ('49','24 CASTLE PARK','flooding','river is flooding apartment');
```

Inserting values into the view does not work as there is a COMPLAINT column in the view and not in the base table.

## Error

### SQL query:

```
INSERT INTO housing_issue_view (HOUSE_ID, ADDRESS, TYPE, COMPLAINT)
VALUES ('48','937 HAZEL GROVE', 'bad smell', 'bad smell coming from sewers'),
       ('49','24 CASTLE PARK','flooding','river is flooding apartment')
```

### MySQL said: ?

#1356 - View 'south\_dublin\_housing\_council\_repairs.housing\_issue\_view' references invalid table(s) or column(s) or function(s) or definer/invokee of view lack rights to use them

But the remaining columns will update if a view is created without the column table and information is inserted into it. The following code was typed into the console:

```
CREATE VIEW housing_issue_view2 AS SELECT HOUSE_ID, ADDRESS, TYPE FROM housing_issue;
```

```
CREATE VIEW housing_issue_view2 AS SELECT HOUSE_ID, ADDRESS, TYPE FROM housing_issue
```

HOUSE_ID	ADDRESS	TYPE
45	543 Spencer Street	leak
46	544 Spencer Street	burstpipe
47	545 Spencer Street	rodent infestation





The following sql query updates the view table and the base table even though a column has been deleted in the base table:





```
INSERT INTO housing_issue_view2 (HOUSE_ID, ADDRESS, TYPE)
VALUES ('48','937 HAZEL GROVE', 'bad smell'),
       ('49','24 CASTLE PARK', 'flooding');
```

✓ 2 rows inserted. (Query took 0.0864 seconds.)

```
INSERT INTO housing_issue_view2 (HOUSE_ID, ADDRESS, TYPE) VALUES ('48','937 HAZEL GROVE', 'bad smell'),
                           ('49','24 CASTLE PARK', 'flooding')
```

The view updated and the base table updated even though the structure had been changed.

Database: south_dublin_housing_council_repairs » View: housing_is			
 SQL	 Search	 Insert	 Export
▼ More			
HOUSE_ID	ADDRESS	TYPE	
45	543 Spencer Street	leak	
46	544 Spencer Street	burstpipe	
47	545 Spencer Street	rodent infestation	
48	937 HAZEL GROVE	bad smell	
49	24 CASTLE PARK	flooding	

Database: south_dublin_housing_council_repairs » Table: housing_is			
 SQL	 Search	 Insert	 Export
▼ More			
HOUSE_ID	ADDRESS	TYPE	
45	543 Spencer Street	leak	
46	544 Spencer Street	burstpipe	
47	545 Spencer Street	rodent infestation	
48	937 HAZEL GROVE	bad smell	
49	24 CASTLE PARK	flooding	

After proving this rule the housing\_issue table and all related views were dropped from the database.



## RULE 10: INTEGRITY INDEPENDENCE

*“Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and storable in the catalogue, not in the application programs.”*

This rule has already been demonstrated at the end of Rule 3 by trying change the primary key in the tenant table to a NULL value. A primary key cannot contain a NULL values, to prove this I tried changing a primary key column to accept NULL values. To demonstrate this I tried changing the TENANT\_ID column to accept NULL values by typing the following SQL:

```
ALTER TABLE `tenant` CHANGE `TENANT_ID` `TENANT_ID` INT(20) NULL;
```

```
ALTER TABLE `tenant` CHANGE `TENANT_ID` `TENANT_ID` INT(20) NULL;
```

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	TENANT_ID 	int(20)			No	None	


The SQL query is accepted by phpadmin but in the Null column in the structure of the tenant table tells us that NULL has not been set.

To demonstrate that NULL value has not been set I typed in the following SQL query:

**Error**

SQL query:

INSERT INTO tenant (TENANT\_ID, REGISTER\_NUMBER, FIRST\_NAME, LAST\_NAME, CONTACT, EMAIL)  
VALUES (NULL, '55885', 'GEORGE', 'LUCAS', '089458765', 'milleniumfalcon@starwars.com')

MySQL said: 

#1048 - Column 'TENANT\_ID' cannot be null




I get an error because a NULL value cannot be used in a primary key.

Then the other tables were tested to make sure that the primary key in each table cannot accept NULL values.

✔ Table equipment has been altered successfully.

ALTER TABLE `equipment` CHANGE `EQUIPMENT\_ID` `EQUIPMENT\_ID` INT(10) NULL;

[ Edit inline ] [ Edit ] [ Create PHP ]

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	EQUIPMENT_ID 	int(10)			No	None		 Change  Drop



## Error

---

### SQL query:

```
INSERT INTO equipment (EQUIPMENT_ID, TYPE, DESCRIPTION, VALUE, INSPECTION_DATE, STATUS, INSPECTOR)
VALUES (NULL, 'Nail Gun', 'Powergun 5000', '29.99', '2015/12/17', 'Poor', 'Yes')
```

### MySQL said:

#1048 - Column 'EQUIPMENT\_ID' cannot be null

### RULE 11: DISTRIBUTION INDEPENDENCE

*"A relational DBMS has distribution independence."*

The Database Management System should appear like one database and users and applications are not required to know where the data is stored. The end user of the Database Management System must not be able to see that the data is distributed over various locations. The user should always get the impression that the data is located at only one site.