

Functional Dependency in Postgresql

A project report

submitted by

Sai Teja Sreeram (22M0775)

Prem Prakash Hansda (22M0818)

Jawwad Pathan (22M0741)

Contents

1	Project Description	2
1.1	Abstract	2
1.2	Problem Description	2
2	Proposed Methodology	3
3	Working and Results	4
3.1	Assumptions	4
3.2	Results	4

Chapter 1

Project Description

1.1 Abstract

Functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table. We store functional dependencies in a separate table and refer this table for every insertion to check whether functional dependency relationship is maintained or not.

1.2 Problem Description

$$X \rightarrow Y$$

For a functional dependency from X (determinant) to Y (dependent) in a table, for each value of X in a table the corresponding Y values should have a one-to-one mapping i.e for two tuples with same X value their corresponding Y values must also be same. We need to restrict any insertion into table which does not satisfy the functional dependency relationship.

Chapter 2

Proposed Methodology

Functional dependencies are stored in a table `funct_dep` with columns (determinant, dependent, `table_name`). Each row in `funct_dep` represent a functional dependency from determinant to dependent column in a `table_name` table.

When any query is done, the program checks if the query is of INSERT type. If it is of INSERT type than preprocessing is done to extract table name. The INSERT type query can be of two types -

1. Type A - INSERT into TABLENAME (`comma_seperated_column_names`) values(`comma_seperated_values`);
2. Type B - INSERT into TABLENAME values(`comma_seperated_values`);

For type A queries we extract the list of column names and list of their corresponding values. For type B queries we extract just the list of values, as the type B queries don't specify column name it must have values for all the columns in the table. We then proceed to obtain the list of column names from `information_schema`.

Next we check if any entries are present in `funct_dep` for the given table name. If some entries are present then for each determinant we find the list of dependent values. We then check if all the dependent values are same or not. If same then the insertion is allowed in the table otherwise error is thrown.

Chapter 3

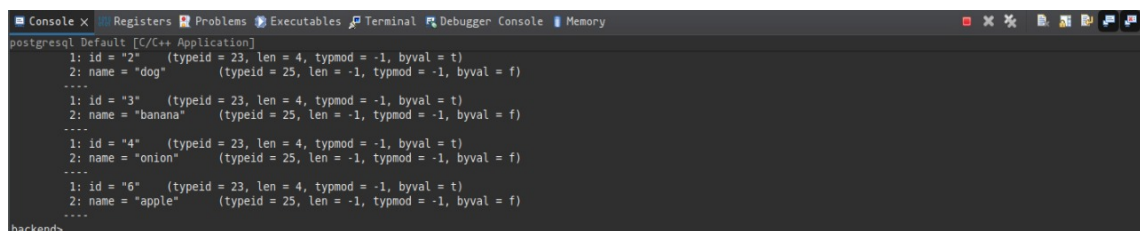
Working and Results

3.1 Assumptions

1. This implementation works only for INSERT type queries.
2. Functional dependency table must be populated before any insertions in related tables.
3. All functional dependencies involves exactly one determinant and one dependent column.

3.2 Results

Here table demo has functional dependencies from id to name.



```
postgresql Default [C/C++ Application]
1: id = "2"      (typeid = 23, len = 4, typmod = -1, byval = t)
2: name = "dog"  (typeid = 25, len = -1, typmod = -1, byval = f)
....
1: id = "3"      (typeid = 23, len = 4, typmod = -1, byval = t)
2: name = "banana" (typeid = 25, len = -1, typmod = -1, byval = f)
....
1: id = "4"      (typeid = 23, len = 4, typmod = -1, byval = t)
2: name = "onion" (typeid = 25, len = -1, typmod = -1, byval = f)
....
1: id = "6"      (typeid = 23, len = 4, typmod = -1, byval = t)
2: name = "apple" (typeid = 25, len = -1, typmod = -1, byval = f)
....
harkent@
```

Figure 3.1: Table containing values

Functional dependency table has entries for table name demo, determinant - leftter and dependant - rightter.

```
Console x Registers Problems Executables Terminal Debugger Console Memory
postgres Default [C/C++ Application]
1: id = "6" (typeid = 23, len = 4, typmod = -1, byval = t)
2: name = "apple" (typeid = 25, len = -1, typmod = -1, byval = f)
....
backend> select * from funct_dep;
1: lefter (typeid = 25, len = -1, typmod = -1, byval = f)
2: righter (typeid = 25, len = -1, typmod = -1, byval = f)
3: table_name (typeid = 25, len = -1, typmod = -1, byval = f)
....
1: lefter = "id" (typeid = 25, len = -1, typmod = -1, byval = f)
2: righter = "name" (typeid = 25, len = -1, typmod = -1, byval = f)
3: table_name = "demo" (typeid = 25, len = -1, typmod = -1, byval = f)
....
backend>
```

Figure 3.2: Functional dependency column

We can verify than any INSERTION violating the functional dependencies are not executed.

```
Console x Registers Problems Executables Terminal Debugger Console Memory
postgres Default [C/C++ Application]
2022-11-29 03:09:11.828 IST [233350] LOG: database system was interrupted; last known up at 2022-11-29 02:49:03 IST
2022-11-29 03:09:11.986 IST [233350] LOG: database system was not properly shut down; automatic recovery in progress
2022-11-29 03:09:11.987 IST [233350] LOG: redo starts at 0/4489C78
2022-11-29 03:09:11.988 IST [233350] LOG: invalid record length at 0/44A5EB8: wanted 24, got 0
2022-11-29 03:09:11.988 IST [233350] LOG: redo done at 0/44A5E90 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s

PostgreSQL stand-alone backend 14.5
backend> insert into demo values(2,'orange');
2022-11-29 03:09:34.439 IST [233350] ERROR: This insert could not be completed because id->name Functional Dependency is being violated'
2022-11-29 03:09:34.439 IST [233350] STATEMENT: insert into demo values(2,'orange');
backend>
```

Figure 3.3: Violation of FD