CS 631 : Implementation Techniques for Relational Database Systems

Functional Dependency Support in PostgreSQL

Pranshu Chourasia : 203050098
Swapnil Malviya: 203050067

15 December 2020

Contents

1 Abstract

In today's world Functional dependencies (FD's) are not supported by pretty much any database. From definition of Functional Dependency Given an FD :-

$$Determinant \rightarrow Determiner$$

you need to create an index on Determinant,Determiner) which stores a count of number of tuples with that (Determinant,Determiner) value, and flags an error if there is more than one Determiner value associated with a given Determinant value.

We try to provide support of FDs on the top of Postgres , such that any insertion into Database table that leads to violation of FD property is reported and not allowed to insert.
Initially we assume that FD supported by Database is already given .

As from definition of FD

$$A \rightarrow B$$

for some tuple t1 and t2 , if t1.A is equal to t2.A then t1.B must be equal to t2.B.

We can use triggers to report if different B values are inserted for the same A values.

As a part of further enhancement , instead of giving supported FD's directly we will try to fetch the supported FD's from the database itself.

2 Goals

1. Our goal is to extend the idea of Functional Dependencies to PostgreSQL.
2. We tried to incorporate multiple Functional Dependencies over single Postgres table using TRIGGERS.
3. Any insertion to the database table may lead to violation Functional Dependencies which is to be verified.
4. Any updation to the database table may lead to violation Functional Dependencies which is to be verified.
5. Deletion to the database table never leads to violation Functional Dependencies.
6. Our goal also includes to deal with dynamics FD's given at runtime by the user.
7. FD's provided by user can be either atomic or composite any form as :-

$$A \rightarrow B$$

$$A, B \rightarrow C$$

$$A \rightarrow B, C$$

$$A, B \rightarrow C, D$$

3 Implementation Details

To implement FDs on the top of postgres we have used python and PL/pgSQL to ensure that while inserting, updating or deleting certain tuples in a table it won't violate the set of Functional Dependency we have given.

Algorithm to check for violation due to insertion:

1. Create a function that will check for violation and create a trigger to call that function.
2. Create a temporary table with all the attributes of the given FD.
   Example: For FD

   $$A, B \rightarrow C$$

   create a temporary relation R(A,B,C)
3. **Before inserting** in main table , insert respective values in the temporary table.
4. Then compare the count of distinct Determinant(A,B) in temporary table and the count of distinct rows in the temporary table. If the counts are not equal that means the FD is violated and thus we will raise and exception and the whole system will be rolled back.
5. If the counts were equal i.e. the FDs are not violated abd we will return new and all the changes will be reflected in main table as well as in the temporary tables.

Algorithm to check for violation due to Updation:

1. Similar to insertion, create a function that will check for violation of FD and create a trigger to call that function.
2. Create a temporary table with all the attributes of the given FD.
   Similar to what we did during insertion.

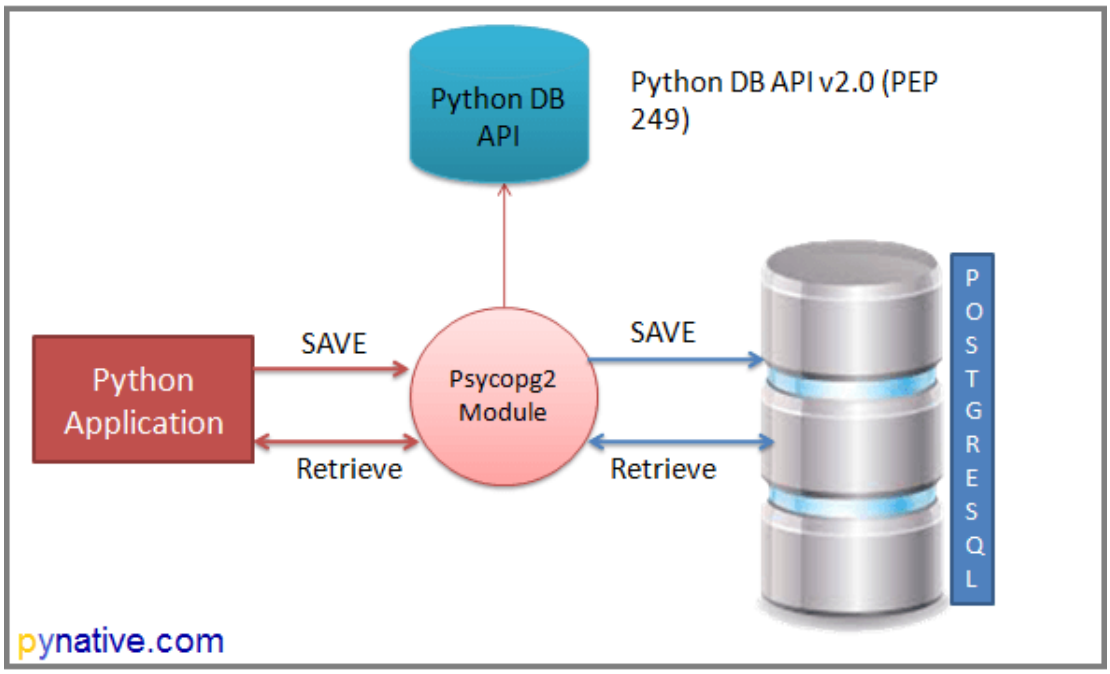
3. Unlike insertion, we are going to perform it **after updation** in main table.
4. We will drop the previous temporaray tables and will create the tables on the basis of the updated values of the Main table.
5. Similar to insertion we will check the count of distinct Determinant and the distinct rows in temporary table. If the count is equal then we will return new and the updates will be made permanent.
6. If the count is not equal then the update in temporary and main table will be rolled back to last consistent state.

Algorithm to avoid violation due to Deletion:

1. We will drop the previous temporary tables.
2. After deletion in main table , we will create new temporary files based on the updated main table.
3. Thus new temporary table will only have entries of all the valid entries in main table.

3.1 Psycopg2 Module

1. We can integrate Postgres with Python using the psycopg2 module.
2. Psycopg2 is a Postgres database adapter for Python. To use this module, you should first install it using :-
   pip3 install psycopg2
3. To connect to your database, you should first create a connection object representing the database.
4. Further we have to create a cursor object to help you in execution of your SQL statements.
5. At last we need to commit changes to database and close connection.



3.2 Functional Dependencies

1. Consider the Functional Dependency :-

   $$Determinant \rightarrow Determiner$$

   The left side of FD is known as a determinant, the right side of the production is known as a determiner.

2. Functional Dependency (FD) is a constraint that determines the relation of one attribute to another attribute in a Database Management System (DBMS). Functional Dependency helps to maintain the quality of data in the database.

3. As from definition of FD

   $$A \rightarrow B$$

   for some tuple t1 and t2 , if t1.A is equal to t2.A then t1.B must be equal to t2.B.

Rules of FD's

1. Reflexive rule – If A is a set of attributes and B is subset of A, then A holds a value of B.

2. Augmentation rule: When

   $$A \rightarrow B$$

   holds, and C is attribute set, then

   $$A, C \rightarrow B, C$$

   also holds. That is adding attributes which do not change the basic dependencies.

3. Transitivity rule: This rule is very much similar to the transitive rule in algebra if

   $$A \rightarrow B$$

   holds and

   $$B \rightarrow C$$

   holds, then

   $$A \rightarrow C$$

   also holds.

4 Future Work

1. We have implemented python file on the top of Postgres due to multiple FD support and dynamic nature of FD's, but in future we try to incorporate changes to postgres internals.

2. Generally TRIGGERS are not considered most efficient way for query's so we would like to explore more ways to check FD violation.

3. Using triggers we can also determine other constraints in the Database table like NULL constraints, Key Constraints etc, although these are provided by Postgres.

4. If we have Pre-Existing Database in the form of .csv file , we can directly import CSV file using Pandas,Numpy packages in python, and can check FD violation more efficiently as they work on Vectorized parallel data.

5. In the above implementation we have to produce multiple temporary tables , to optimize the query processing time which leads to excessive usage of storage, in future work we would like to optimize storage space as well.

6. In above implementation we expect the user to provide Minimal FD set for optimal query, but in future we would incorporate minimization of FD set.

5 References and Links

1. **https://youtu.be/ewuSDIKuq4Y**
2. **https://www.postgresql.org/docs/10/plpgsql.html**
3. **https://www.youtube.com/watch?v=uQhQp7nXUZs**
4. **https://stackabuse.com/working-with-postgresql-in-python**

1. **Github Link for Project**
   https://github.com/pranshu1202/Functional-Dependency

2. **Overleaf Report Link for Project**
   https://www.overleaf.com/read/gmyqqvcjjxhd