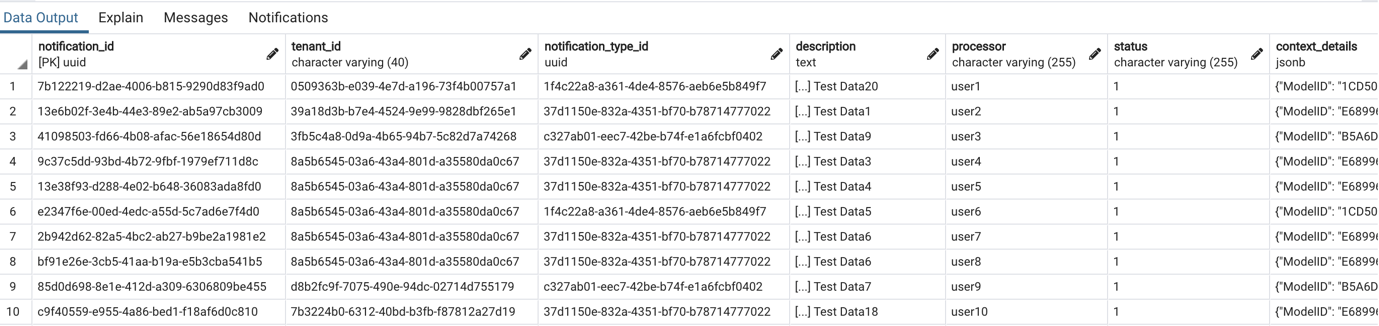
**Problem:**

Assume there is a table with very huge data and the size of the table grows eventually. When queried on this data, this may lead to performance issues.

For example, if there is table that stores the notifications in a data intense application, that is used to save the notifications of all the tenants. This table will increase gradually within no time. The data for tenants can grow drastically and over a period of time, it could result in performance issues. In the first place, it’s not a good design to have the data saved in single table.



Below is the DDL of the table

CREATE TABLE IF NOT EXISTS notification

(

notification\_id uuid NOT NULL DEFAULT uuid\_generate\_v4(),

tenant\_id character varying(40) NOT NULL,

notification\_type\_id uuid NOT NULL,

description text,

processor character varying(255),

status character varying(255),

context\_details jsonb,

context\_id character varying(40),

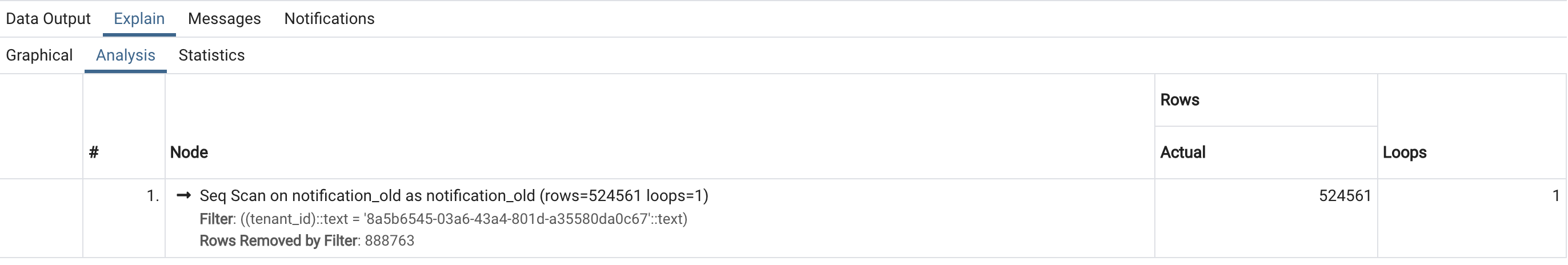
CONSTRAINT notification\_key PRIMARY KEY (notification\_id)

)

There is no index on tenant\_id column. Insert some million records in the table. Fetch the records based on tenant id.

**QUERY** : *select \* from notification where tenant\_id = '8a5b6545-03a6-43a4-801d-a35580da0c67'*

**Execution Plan:**



**Output**: *Successfully run. Total query runtime: 3 min 25 secs.*

*524561 rows affected.*

Adding an Index:

Check if indexing alone helps. Now add index to the tenant\_id column of the table.

*CREATE INDEX tenant\_indx on notification(tenant\_id);*

**QUERY** : select \* from notification where tenant\_id = '8a5b6545-03a6-43a4-801d-a35580da0c67'

**Output**: *Total query runtime: 3 min 13 secs.*

*524561 rows affected.*

Addition of Partition:

Now the table can be partitioned based on the table. First we will have to create a partition based on List. Then for the unique values of tenant, we create a new partition by suffixing the tenant\_id to the partition name.

Below script will

1. Rename the old table
2. Create a new table with partition
3. Reads the unique tenant values from old table
4. Creates new partition tables considering tenant as the partition value. Also the partition table name will be suffixed with the tenant Id.

*do*

*$$*

*declare*

*tenant text;*

*table\_tenant text;*

*rec record;*

*begin*

*EXECUTE 'ALTER TABLE notification rename to notification\_old'*

*EXECUTE 'CREATE TABLE IF NOT EXISTS notification*

*(*

*notification\_id uuid NOT NULL DEFAULT uuid\_generate\_v4(),*

*tenant\_id character varying(40) NOT NULL,*

*notification\_type\_id uuid NOT NULL,*

*description text,*

*processor character varying(255),*

*status character varying(255),*

*context\_details jsonb,*

*context\_id character varying(40),*

*CONSTRAINT notification\_key PRIMARY KEY (notification\_id)*

*) PARTITION BY LIST(tenant\_id)'*

*FOR rec IN*

*select DISTINCT tenant\_id from notification\_old*

*loop*

*tenant := rec.tenant\_id;*

*table\_tenant := 'notification\_'|| REPLACE (rec.tenant\_id, '-', '\_');*

*execute 'CREATE TABLE IF NOT EXISTS ' || table\_tenant || ' PARTITION OF notification FOR VALUES IN ('''|| tenant ||''')';*

*END LOOP;*

*end;*

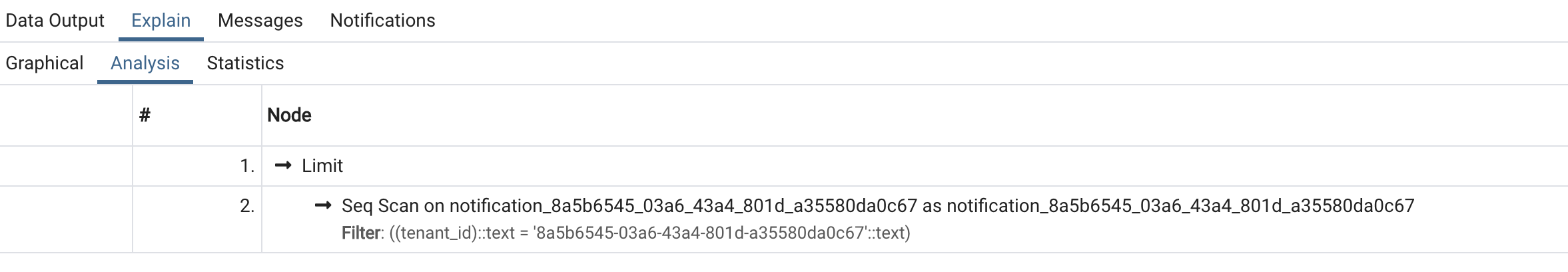
*$$ language plpgsql*

Then to copy the data from older table to newer table,

**Query**: *insert into notification*

*select \* from notification\_old*

**Execution Plan:**



The query executes directly on the partitioned value with given tenant\_id

**Output** : *Successfully run. Total query runtime: 2 min 6 secs.*

*524561 rows affected.*

**Conclusion**: Applying partitions on tables will have very good performance improvements when data is filtered on partition value. However, if entire data needs to be fetched from the parent table without any filter, the query has to union all the partition tables.