

How to change DB Table Primary key using Connect SQL Rule

Step 1.

Before you drop and recreate a primary key, you should first identify the exact name of the primary key constraint, because PostgreSQL automatically generates constraint names if you didn't specify one during creation.

If you skip this step and just guess the name, your ALTER TABLE ... DROP CONSTRAINT ... will fail.

Here's how to find it:

Open Query Runner and run this query (Configure -> system -> database ->QueryRunner)

```
SELECT conname AS constraint_name  
FROM pg_constraint AS con  
INNER JOIN pg_class AS t ON con.conrelid = t.oid  
INNER JOIN pg_namespace AS n ON n.oid = t.relnamespace  
WHERE c.contype = 'p'  
AND t.relname = 'your_table_name'  
AND n.nspname = 'your_schema_name';
```

Both query's are ok ...

```
SELECT con.*  
FROM pg_catalog.pg_constraint con  
INNER JOIN pg_catalog.pg_class rel ON rel.oid = con.conrelid  
INNER JOIN pg_catalog.pg_namespace nsp ON nsp.oid = connamspace  
WHERE con.contype = 'p' AND rel.relname = '<table_Name>';
```

Explanation

- `contype = 'p'` → only primary key constraints
- `relname` → your table name
- `nspname` → your schema name (e.g., `public`)

You're working with PostgreSQL system catalog tables (`pg_catalog` schema).

- **`pg_constraint`** → stores info about constraints (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, etc.).
- **`pg_class`** → stores info about database objects like tables, indexes, sequences, etc.

`Pg_class` : [PostgreSQL: Documentation: 17: 51.11. pg_class](#)

The join logic

```
INNER JOIN pg_catalog.pg_class rel  
ON rel.oid = con.conrelid
```

Here's what's happening:

- `con.conrelid` in `pg_constraint` → the OID (object ID) of the table that the constraint belongs to.
- `rel.oid` in `pg_class` → the OID that identifies each table (or other database object) in PostgreSQL.

Why join them?

Because `pg_constraint` only stores the table ID (numeric OID), not the table name. To get the actual table name, you must join to `pg_class` (which has the `relname` column for table name).

1 The catalog tables involved

- `pg_namespace` → stores all schema names in the database (like `public`, `information_schema`, etc.).

- pg_constraint → has a column connamespace which stores the OID of the schema where the constraint is defined.

Pg_constraint : [PostgreSQL: Documentation: 17: 51.13. pg_constraint](#)

2 The join

```
INNER JOIN pg_catalog.pg_namespace nsp
  ON nsp.oid = con.connamespace
```

- con.connamespace → the schema ID where the constraint exists.
- nsp.oid → the actual ID of the schema in pg_namespace.

By joining, we can get nsp.nspname (schema name) instead of just seeing a numeric OID.

3 Why needed

pg_constraint doesn't store the schema name directly — it only keeps the ID (connamespace).

Joining to pg_namespace lets you display the schema name in a readable form.

Pg_namespace : [PostgreSQL: Documentation: 17: 51.32. pg_namespace](#)

- conname → this is the exact primary key

[Query Entry](#) History

Default query timeout is 60 seconds. It's not recommended to increase it! But if you absolutely must it's possible to increase database/QueryRunner/queryTimeout up to 300 seconds. Values above 300 seconds will be ignored. You can find more information on query runner documentation

Choose your database* CustomerData

SQL query

Use alias name if the column name contains special characters

Query to run*
 SELECT con.conname AS constraint_name, rel.relname AS tablename, con.contype AS cont_type, nsp.nspname AS nsp, rel.relname AS rel_name FROM pg_catalog.pg_constraint con INNER JOIN pg_catalog.pg_class rel ON rel.oid = con.conrelid INNER JOIN pg_catalog.pg_namespace nsp ON nsp.oid = connamespace WHERE con.contype='p' AND rel.relname='pr_zpt_crm_data_testnr';

[Run](#)

RESULT(S)

Limited results.

constraint_name	tablename	cont_type	nsp	rel_name
1 pr_zpt_crm_data_testnr_pkey	pr_zpt_crm_data_testnr	p	customerdata	pr_zpt_crm_data_testnr

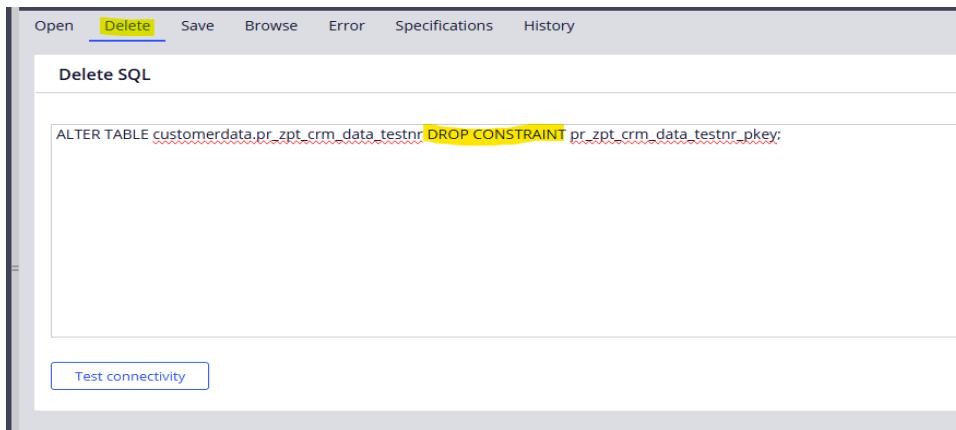
Export ▾

Keep exact constrain key.

Step 2

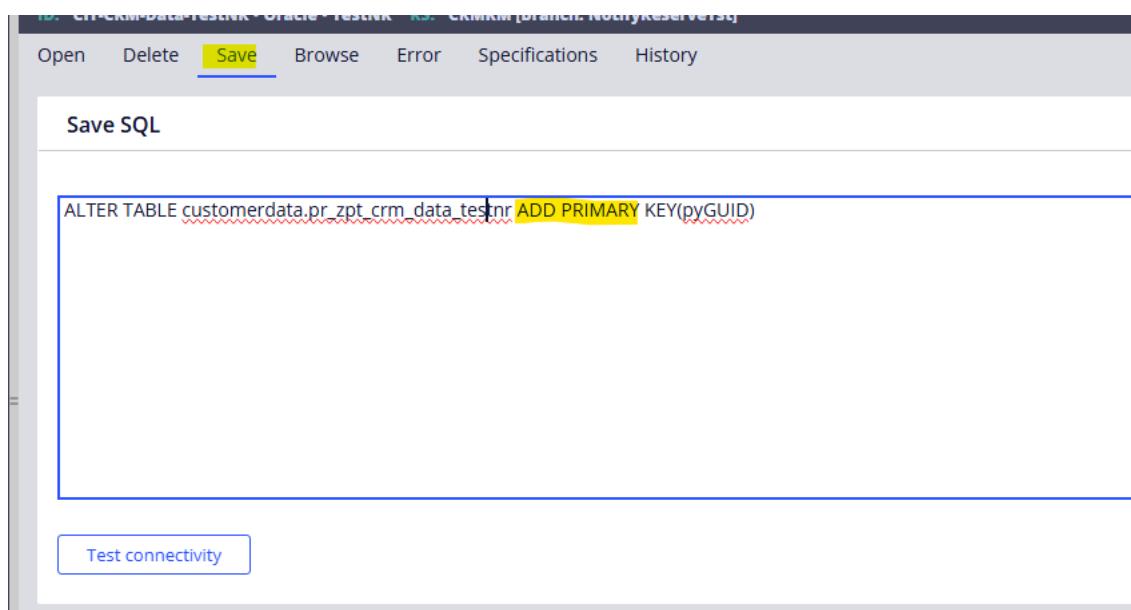
- Create new ConnectSQL rule and open delete tab.
- Put this Command on the delete Tab (remove existing primary key constraint)


```
ALTER TABLE <databaseName>.<tableName> DROP CONSTRAINT
<PrimaryKeyConstraint>
```



- Put this command for save Tab (add new primary key to table)


```
ALTER TABLE <databaseName>.<tableName> ADD PRIMARY (<Your primary key Name>)
```



Step 3

- Create activity and add **RDB-Delete** method for Drop pk constraint
- Add RDB-Save method for add new pk constraint
 - RequestType -->created ConnectSQL rule name

This record has 1 severe or moderate warning (including 1 unjustified) and 1 info war															
Steps	Parameters	Pages & Classes	Security	Test cases	Specifications										
1.	Label Loop When > RDB-Delete	Preferred contact method RDB-Delete	Step page NR	Description Drop Pk											
Method Parameters															
<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>*ClassName</td> <td>customerdata-TestNR</td> </tr> <tr> <td>*RequestType</td> <td>TestNR</td> </tr> <tr> <td>*Access</td> <td>Oracle</td> </tr> <tr> <td colspan="2">RunInParallel <input type="checkbox"/></td> </tr> </tbody> </table>						Name	Value	*ClassName	customerdata-TestNR	*RequestType	TestNR	*Access	Oracle	RunInParallel <input type="checkbox"/>	
Name	Value														
*ClassName	customerdata-TestNR														
*RequestType	TestNR														
*Access	Oracle														
RunInParallel <input type="checkbox"/>															
2.	Label Loop When > RDB-Save	Preferred contact method RDB-Save	Step page NR	Description Add new Pk											
+ Add a step Collapse all steps															

- Comment RDB-Save method and Run Activity, then our pk constraint will drop. You can verify by running bellow query on the query runner. you will see Empty result(s).
 - Comment RDB-Delete method and uncomment RDB-Save method , run activity. Then our new pk will added to table.

Open Database Table and click test connectivity button

The screenshot shows the Pega Platform interface for creating a new database table. The left sidebar has a dark theme with white text. The 'Database Table' option is highlighted with a blue background. The main panel has a light gray background. It displays a 'Database' configuration form with the following fields:

- Database: CustomerData
- Reports database: (empty)
- Catalog name: (empty)
- Schema name: (empty)
- Table name: pr_crm_data_testnr

At the bottom of the form is a blue button labeled 'Test connectivity'. A red arrow points to this button.

You will see your table pk has changed

```
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxCommitDateTime
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxCreateDateTime
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxCreateOperator
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxCreateOpName
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxCreateSystemID
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxSaveDateTime
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxUpdateDateTime
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxUpdateOperator
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxUpdateOpName
data_testnr in database CustomerData , but it contains a px, py, or pz property: pxUpdateSystemID
the primary key columns for external data table customerdata.pr_cit_crm_data_testnr in database CustomerData ; the class keys are [TestNR_ID] , but the table's primary keys are [pega.customerdata.pyguid]
```

..... Happy Learning