#### Python Custom Policy Checks Tutorial for Database Checks

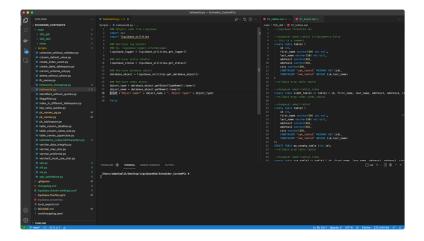
- Introduction
- Prerequisites
  - Private Repo
  - Public Repo
- Database Setup
- · Python Basic Code
- Disable All Checks
- · First Custom Policy Check
- Run Your First Python Custom Policy Check
- Task 1: Create a Check for Primary Key Name (xpk\_tableName)
- Task 2: Create a Check for Unique Constraint Name Must Start With "xak"
- Task 3: Create a Check for Index Name Must Start With "xie" + [0-9]{1,2} + (.\*)
- Task 4: Create a Check for Foreign Key Name Must Start With "xfk"
- Task 5: (database) Create a Check for Table Name Must Not Have Special Characters ".\*[\*#+-].\*"

#### Introduction

In this tutorial, we will explore authoring custom policy checks using Python.

We will write custom checks for database (i.e, --checks-scope=database) starting with an equivalent of a "hello world" example and then diving deeper.

All of the code editing in this tutorial was done using VS Code.



#### **Prerequisites**

- Install Liquibase 4.29.0+ ( ♠ Release Liquibase v4.29.2 · liquibase/liquibase )
- Install Policy Checks extension (maven repository)
- Install Python 3.10.14 or higher
- $\bullet \ \ \mbox{Set this property: } \ \mbox{checks-scripts-enabled=true}$

Verify that you have Python3 installed:

```
1 % python3 --version
2 Python 3.12.3
```

#### Private Repo

Clone @James Bennett's repository (git clone git@github.com:liquibase/CustomPCs.git) and work in that repo because there are tons of sample Python scripts available in Scripts directory

#### Public Repo

Clone @Adeel Malik 's repository (git@github.com:adeelmalik78/python\_policy\_checks.git) which contains examples shown in this tutorial.

Here is a starter liquibase.properties file:

```
changeLogFile: changelog.sql
url: jdbc:oracle:thin:@cs-oracledb.liquibase.net:1521/PP_DEV
username: liquibase_user
password: liquibase_user
liquibase.reports.enabled: false
checks-scripts-enabled=true
```

#### **Database Setup**

Setup database for what you want tested with Python custom policy checks.

The following changelog is compatible with both Oracle and Postgres. Name this file changelog.sq1. These few SQL commands create tables with columns, primary keys and unique constraints. We can design custom policy checks around these. Go ahead and deploy these changesets to your Oracle or Postgres database.

```
1 --liquibase formatted sql
2
3 --changeset adeel:table1
4 create table table1 (
      id int,
 6
     first_name varchar(50) not null,
7
     last_name varchar(50) not null,
8
     address1 varchar(50),
9
     address2 varchar(50),
10
     city varchar(30),
       CONSTRAINT "xpk_table1" PRIMARY KEY (id),
11
12
       CONSTRAINT "xak_table1" UNIQUE (id, last_name)
13 )
14 --rollback drop table table1
15
16 --changeset adeel:table1_index
17 create index xie01_table1 on table1 ( id, first_name, last_name, address1, address2, city );
18 --rollback drop index xie01_table1
19
20 -- changeset adeel: table2
21 create table table2 (
22
      id int,
23
       first_name varchar(50) not null,
```

```
last_name varchar(50) not null,
25
     address1 varchar(50),
26
     address2 varchar(50),
27
     city varchar(30),
28
       CONSTRAINT "xpk_table2" PRIMARY KEY (id),
       CONSTRAINT "xak_table2" UNIQUE (id, last_name)
30 )
31 --rollback drop table table2
32
33 --changeset adeel:table2_index
34 create index xie_table2 on table2 ( id, first_name, last_name, address1, address2, city );
35 --rollback drop index xie_table2
36
37 -- changeset adeel: person
38 CREATE TABLE person (
39
     id int,
40
     first_name varchar(50) NOT NULL,
41
     last_name varchar(50) NOT NULL,
     CONSTRAINT "pk_person" PRIMARY KEY (id),
      CONSTRAINT "ak_person" UNIQUE (id,last_name)
43
44 )
45 -- rollback drop table person
46
47 -- changeset adeel:person_index
48 create index xie03_person on person ( id, first_name, last_name );
49 --rollback drop index xie03_person
50
51 -- changeset adeel: fk_table1
52 alter table table1 add constraint fk_table1 foreign key (id) references table2 (id);
   --rollback alter table table1 drop constraint fk_table1;
54
55 -- changeset adeel:xfk_table2
56 alter table table2 add constraint xfk_table2 foreign key (id) references table1 (id);
57 --rollback alter table table2 drop constraint xfk_table2;
58
59 -- changeset adeel: employee
60 CREATE TABLE "employee+" (
61
     id int,
     first_name varchar(50) NOT NULL,
62
     last_name varchar(50) NOT NULL,
63
     CONSTRAINT "xpk_employee" PRIMARY KEY (id),
65
       CONSTRAINT "xak_employee" UNIQUE (id, last_name)
66 )
67 -- rollback drop table "employee+"
```

Deploy these changes to your database before configuring your custom policy checks.

# Python Basic Code

Create the following Python code and save it in Scripts directory. Name this script as helloworld.py. This script

```
### Helpers come from Liquibase
import sys
import liquibase_utilities

### Retrieve log handler
### Ex. liquibase_logger.info(message)
```

```
7 liquibase_logger = liquibase_utilities.get_logger()
8
9 ### Retrieve status handler
10 liquibase_status = liquibase_utilities.get_status()
11
12 ### Retrieve database object the liquibase policy check is examining
database_object = liquibase_utilities.get_database_object()
14
15 ### Retrieve object type and object name
object_type = database_object.getObjectTypeName().lower()
object_name = database_object.getName().lower()
18
19 ### Print object type and object name to console output
20 print ("Object name=" + object_name + ", Object type=" + object_type)
22 ### Default return code
23 False
```

Note: We are not triggering this check. We are always exiting this Python code with False . We will see in a later example how to trigger a check.

Also note: We haven't put any loop here.

Question: How many times will this Python script execute when we invoke this check?

We will answer this question later.

#### Disable All Checks

Let's create our liquibase.checks-settings.conf file:

```
1 % liquibase checks show
2 Press [1] to create a new checks settings file
3
4 % liquibase checks bulk-set --disable
5 Confirm: YES
6
7 % liquibase checks show --check-status=enabled
8 +-----+
9 | Short Name | Scope | Status | Severity | Customization | Description |
10 +-----+
```

## First Custom Policy Check

Let's use the CustomCheckTemplate to write our first Python-based custom quality check:

```
% liquibase checks customize --check-name=CustomCheckTemplate

Short name: HelloWorld

Severity: 0-4

Script description: List all objects

Script scope: database

Script message: This check will never trigger. This is a first custom policy check.
```

```
8 Script type: python
9 Script path:
            Scripts/helloworld.py
10 Script_Args: <empty>
11 Requires snapshot: false
12
13 % liquibase checks show --check-status=enabled
15 | Short Name | Scope | Status | Severity | Customization
                                            | Description
16 +-----
17 | HelloWorld | database | enabled | 4
                         | SCRIPT_DESCRIPTION = List all | Executes a custom check
 18
        | | objects
                                            | script.
 ш
19
                    | SCRIPT_SCOPE = database |
 П
20
         | SCRIPT_MESSAGE = This check |
                         | will never trigger. This is a |
21
                    22
                    | | first custom policy check.
 | SCRIPT_TYPE = PYTHON
23
               24
                    | SCRIPT_PATH =
25
                    | Scripts/helloworld.py
  | | SCRIPT_ARGS = null
26
                         | REQUIRES_SNAPSHOT = false
27
                   28 +--
    29
```

# Run Your First Python Custom Policy Check

Let's run our policy check and examine the output:

```
% liquibase checks run --checks-scope=database

Executing all database checks because a valid license key was found!

Object name=xpk_table1, Object type=index
Object name=first_name, Object type=column
Object name=city, Object type=column
Object name=address1, Object type=column
Object name=address2, Object type=column
Object name=id, Object type=column
Object name=table1, Object type=table
Object name=last_name, Object type=column
Object name=table2, Object type=column
Object name=first_name, Object type=column
```

```
15 Object name=xak_employee, Object type=uniqueconstraint
16 Object name=public, Object type=schema
17 Object name=xak_table2, Object type=index
18 Object name=first_name, Object type=column
19 Object name=last_name, Object type=column
20 Object name=xie_table2, Object type=index
21 Object name=id, Object type=column
22 Object name=xak_table2, Object type=uniqueconstraint
23 Object name=address1, Object type=column
24 Object name=first_name, Object type=column
25 Object name=postgres, Object type=catalog
26 Object name=address2, Object type=column
27 Object name=xieO1_table1, Object type=index
28 Object name=pk_person, Object type=index
29 Object name=xfk_table2, Object type=foreignkey
30 Object name=fk_table1, Object type=foreignkey
31 Object name=person, Object type=table
32 Object name=xpk_table1, Object type=primarykey
33 Object name=xpk_employee, Object type=index
34 Object name=ak_person, Object type=index
35 Object name=id, Object type=column
36 Object name=last_name, Object type=column
37 Object name=last_name, Object type=column
38 Object name=xpk_table2, Object type=index
39 Object name=xak_employee, Object type=index
40 Object name=id, Object type=column
41 Object name=employee+, Object type=table
42 Object name=xpk_table2, Object type=primarykey
43 Object name=xie03_person, Object type=index
44 Object name=pk_person, Object type=primarykey
45 Object name=xpk_employee, Object type=primarykey
46 Object name=city, Object type=column
47 Object name=xak_table1, Object type=uniqueconstraint
48 Object name=ak_person, Object type=uniqueconstraint
49 Object name=xak_table1, Object type=index
50 Checks-settings File: liquibase.checks-settings.conf
52 Database objects Validated:
     Catalog
53
      Column
                        : 18
54
     ForeignKey
55
                        : 2
       Index
56
                         : 11
57
       PrimaryKey
                        : 4
58
       Schema
                        : 1
59
       Table
                         : 4
60
        UniqueConstraint
                          : 4
61 To increase details set the --verbose property
62
63 Checks run against database jdbc:postgresql://localhost:5432/postgres:
64
     Custom Check Template (Short names: HelloWorld)
```

Did you get this result? Congratulations!!!!

So, how many times did the Python script execute?

Answer: As many times as there are objects in the database. We know this because our basic code printed out all object names that were created by our changelog.sql.

#### Task 1: Create a Check for Primary Key Name (xpk tableName)

Let's create a new python script in the Scripts directory and name it xpk\_tableName.py.

In this script, we will build on our earlier helloworld.py script.

- We add the code for finding the table name (lines 20-24).
- Find table's primary key (lines 26-31).
- Compare if the primary key name matches with xpk\_tableName naming convention (lines 32-35).
- If the primary key does not match naming convention then we trigger the check (line 36) and print out relevant message to the user (lines 38-40).
- Exit from the check (line 42).

```
1 ### Helpers come from Liquibase
 2 import sys
 3 import liquibase_utilities
 4
 5 ### Retrieve log handler
  6 ### Ex. liquibase_logger.info(message)
 7 liquibase_logger = liquibase_utilities.get_logger()
 8
 9 ### Retrieve status handler
10 liquibase_status = liquibase_utilities.get_status()
11
12 ### Retrieve database object
database_object = liquibase_utilities.get_database_object()
14
15 ### Retrieve database object type and object name
object_type = database_object.getObjectTypeName().lower()
object_name = database_object.getName().lower()
print ("Object name=" + object_name + ", Object type=" + object_type)
20 ### Are we executing this script on a table object?
21 if "table" in database_object.getObjectTypeName().lower():
22
23
                ### Get table name
                table_name = database_object.getName().lower()
24
25
26
            ### Get primary key for the table, if there is one ...
27
                pk_object = database_object.getPrimaryKey()
28
29
                ### If there is a primary key for this table, then ...
30
                if pk_object != None:
31
                        pk_name_current = pk_object.getName()
32
                         pk_name_expected = "xpk_" + f"{table_name}"
33
34
                         \#\#\# if the current primary key name not what we expected ...
35
                        if pk_name_expected not in pk_name_current:
                                  liquibase_status.fired = True # indicates the custom check has been triggered
36
37
                                  # set the message of the custom check, which liquibase will return
                                  status_message = "Found primary key name" + f"{pk_name_current}" + " which did not match what was status_message = "Found primary key name" + f"{pk_name_current}" + " which did not match what was status_message = "Found primary key name" + f"{pk_name_current}" + " which did not match what was status_message = "Found primary key name" + f"{pk_name_current}" + f" which did not match what was status_message = "Found primary key name" + f"{pk_name_current}" + f" which did not match what was status_message = "Found primary key name" + f"{pk_name_current}" + f" which did not match what was status_message = found primary key name for the first found primary key name for the
39
        expected: " + f"{pk_name_expected}"
40
                                  liquibase_status.message = status_message
41
42
                                  sys.exit(1)
                                                               # exit from the check
```

```
44 ### No primary key found in the table object.
45 ### For this table object we will not trigger this check
46 False
```

Customize policy check using CustomCheckTemplate:

```
1 % liquibase checks customize --check-name=CustomCheckTemplate
2
3 Short name:
            PKNamingConvention
            0-4
4 Severity:
5 Script description: Primary Key name
6 Script scope: database
7 Script message: <empty>
8 Script type: python
9 Script path:
            Scripts/xpk_tableName.py
10 Script_Args: <empty>
11 Requires snapshot: false
12
13 % liquibase checks show --check-status=enabled
15 | Short Name | Scope | Status | Severity | Customization
 17 | PKNamingConvention | database | enabled | 4 | SCRIPT_DESCRIPTION = Primary | Executes a custom
 check |
                   | Key name
18
             | script.
 19
                    | SCRIPT_SCOPE = database
                     1
 20
                   | SCRIPT_MESSAGE = The message |
21
                   1
                               | to display when the check is |
                        22
                    | triggered
 П
23
                   | SCRIPT_TYPE = PYTHON
                               | SCRIPT_PATH =
24
                   П
25
                            | Scripts/xpk_tableName.py
                    26
                        | SCRIPT_ARGS = null
 П
27
                   | REQUIRES_SNAPSHOT = false
```

Let's run this check. If you ran against the database populated with the changelog.sql provided earlier, you would see a single check triggered:

```
liquibase checks run --checks-scope=database

DATABASE CHECKS

Validation of the database snapshot found the following issues:
```

```
7 Check Name:
                     Custom Check Template (PKNamingConvention)
8 Object Type:
                    table
9 Object Name:
                    person
10 Object Location: postgres.public.person
11 Check Severity: BLOCKER (Return code: 4)
                   Found primary key name pk_person which did not match what was expected: xpk_person
12 Message:
13
14 Database objects Validated:
     Catalog
15
16
      Column
                      : 2
17
     ForeignKey
18
      Index
                       : 11
       PrimaryKey
20
     Schema
                       : 1
      Table
                        : 4
       UniqueConstraint : 4
22
23 To increase details set the --verbose property
25 Checks run against database jdbc:postgresql://localhost:5432/postgres:
    Custom Check Template (Short names: PKNamingConvention)
```

### Task 2: Create a Check for Unique Constraint Name Must Start With "xak"

Let's create a new python script in the Scripts directory and name it xak.py.

In this script, we will build on our earlier xpk\_tableName.py script.

- We commented out the print statement in line 18.
- We updated the code for finding the unique constraint name (lines 20- 25).
- Compare if the unique constraint name contains xak (line 28).
- If the unique constraint name does not match naming convention then we trigger the check (line 29).
- Print out relevant message to the user (lines 31-33).
- Exit from the check (line 35).

```
1 ### Helpers come from Liquibase
2 import sys
3 import liquibase_utilities
5 ### Retrieve log handler
 6 ### Ex. liquibase_logger.info(message)
7 liquibase_logger = liquibase_utilities.get_logger()
8
9 ### Retrieve status handler
10 liquibase_status = liquibase_utilities.get_status()
11
12 ### Retrieve database object
database_object = liquibase_utilities.get_database_object()
14
15 ### Retrieve database object type and object name
object_type = database_object.getObjectTypeName().lower()
object_name = database_object.getName().lower()
# print ("Object name=" + object_name + ", Object type=" + object_type)
19
```

```
20 ### Are we executing this script on a uniqueconstraint object?
21 if "uniqueconstraint" in object_type:
22
23
       # ### If there is a unique constraint, then ...
24
       uc_name_current = object_name
     uc_name_expected = "xak_"
26
27
       ### if the current primary key name not what we expected ...
28
       if uc_name_expected not in uc_name_current:
29
           liquibase_status.fired = True  # indicates the custom check has been triggered
30
           # set the message of the custom check, which liquibase will return
31
           status_message = "Found unique constraint name" + f"{uc_name_current}" + " which did not start with "
32
   + f"{uc_name_expected}"
33
           liquibase_status.message = status_message
34
35
           sys.exit(1) # exit from the check
36
37 ### No unique constraints found in the database.
38 ### For this object we will not trigger this check
```

Customize another CustomCheckTemplate policy check for unique constraint naming convention:

```
1 % liquibase checks customize --check-name=CustomCheckTemplate
2
3 Short name:
               UCNamingConvention
4 Severity:
5 Script description: Unique Constraint name
6 Script scope: database
7 Script message: <empty>
8 Script type:
              python
9 Script path:
              Scripts/xak.py
10 Script_Args:
               <empty>
11 Requires snapshot: false
12
13 % liquibase checks show --check-status=enabled
| Scope | Status | Severity | Customization
15 | Short Name
                                                          | Description
16 +-----
17 | UCNamingConvention | database | enabled | 4 | SCRIPT_DESCRIPTION = Unique | Executes a custom
                      1
                                    | Constraint name
18
                                                           | script.
  19
                       | SCRIPT_SCOPE = database
  1
20
                                 | SCRIPT_MESSAGE = The message
  П
21
                       | to display when the check is |
22
                       1
                             | triggered
  П
23
                       | SCRIPT_TYPE = PYTHON
24
                            | SCRIPT_PATH =
```

Let's run this check. If you ran against the database populated with the changelog.sql provided earlier, you would see a single check triggered:

```
1 liquibase checks run --checks-scope=database
2
3 DATABASE CHECKS
5 Validation of the database snapshot found the following issues:
6
7 Check Name:
                    Custom Check Template (UCNamingConvention)
8 Object Type:
                    uniqueConstraint
9 Object Name:
                     ak_person
10 Object Location: postgres.public.person.id.ak_person
11 Check Severity: BLOCKER (Return code: 4)
12 Message:
                      Found unique constraint name ak_person which did not start with xak_
13
14 Database objects Validated:
15
     Catalog
16
      Column
                         : 18
      ForeignKey
17
                         : 2
18
       Index
19
       PrimaryKey
       Schema
        Table
21
22
        UniqueConstraint
                          : 4
23 To increase details set the --verbose property
24
25 Checks run against database jdbc:postgresql://localhost:5432/postgres:
     Custom Check Template (Short names: UCNamingConvention)
```

# Task 3: Create a Check for Index Name Must Start With "xie" + [0-9]{1,2} + (.\*)

Let's create a new python script in the Scripts directory and name it xie.py.

In this script, we will build on our earlier xak.py script.

- We import a package called re which can be used to work with Regular Expressions (line 4).
- We commented out the print statement in line 19.
- Print out all indexes found (line 24). This is needed because Liquibase identifies primary keys, unique constraints as indexes.
- We updated the code for finding the index name (lines 21- 28). The index naming pattern (regular expression) is passed as a string on line 28.
- Compare if the index name contains the index naming pattern (line 31).
- If the index name does not match naming convention then we trigger the check (line 33).
- Print out relevant message to the user (lines 35-37).
- Exit from the check (line 39).

Copy/paste the following python code:

```
1 ### Helpers come from Liquibase
2 import sys
3 import liquibase_utilities
4 import re
5
6 ### Retrieve log handler
7 ### Ex. liquibase_logger.info(message)
8 liquibase_logger = liquibase_utilities.get_logger()
9
10 ### Retrieve status handler
11 liquibase_status = liquibase_utilities.get_status()
12
13 ### Retrieve database object
14 database_object = liquibase_utilities.get_database_object()
15
16 ### Retrieve index object
17 object_type = database_object.getObjectTypeName().lower()
18 object_name = database_object.getName().lower()
# print ("Object name=" + object_name + ", Object type=" + object_type)
21 ### Are we executing this script on a index object?
22 if "index" in object_type:
23
       print ("Index name=" + object_name + ", Object type=" + object_type)
24
25
26
       # ### If there is a unique constraint, then ...
27
       index_name_current = object_name
28
       index_name_expected = "^(xie)[0-9]{1,2}(.*)"
29
       ### if the current primary key name not what we expected ...
     if not (re.search(index_name_expected, index_name_current)):
31
32
     # if index_name_expected not in index_name_current:
33
           liquibase_status.fired = True # indicates the custom check has been triggered
34
35
           # set the message of the custom check, which liquibase will return
           status_message = "Found index name " + f"{index_name_current}" + " which did not agree with INDEX
   naming convention " + f"{index_name_expected}"
37
           liquibase_status.message = status_message
38
39
           sys.exit(1)
                        # exit from the check
40
41 ### No unique constraints found in the database.
42 ### For this object we will not trigger this check
43 False
```

Customize another CustomCheckTemplate policy check for index naming convention:

```
1 % liquibase checks customize --check-name=CustomCheckTemplate
2
3 Short name:
                     IndexNamingConvention
4 Severity:
                     0-4
5 Script description: Index name
6 Script scope:
                   database
7 Script message:
                     <empty>
8 Script type:
                      python
9 Script path:
                     Scripts/xie.py
10 Script_Args:
                     <empty>
```

```
11 Requires snapshot: false
12
13 % liquibase checks show --check-status=enabled
14 +-----
15 | Short Name
                | Scope | Status | Severity | Customization
                                                        | Description
16 +-----
17 | IndexNamingConvention | database | enabled | 4 | SCRIPT_DESCRIPTION = Index | Executes a custom
  check
         18
                 | name
                                                        | script.
  П
19
                                   | SCRIPT_SCOPE = database
                        20
                        | SCRIPT_MESSAGE = The message
  21
                        | to display when the check is |
  ш
22
                              | triggered
  | SCRIPT_TYPE = PYTHON
23
                        24
                              | SCRIPT_PATH = Scripts/xie.py
  П
25
                                   | SCRIPT_ARGS = null
                        26
                             | REQUIRES_SNAPSHOT = false
```

Let's run this check. If you ran against the database populated with the changelog.sql provided earlier, you would see a single check triggered.

Notice that in addition to triggering on indexes, this check is also triggered for primary keys and unique constraints in the database:

```
1 liquibase checks run --checks-scope=database
2
3 Object name=xpk_table1, Object type=index
4 Object name=xie01_table1, Object type=index
5 Object name=xie02_table2, Object type=index
6 Object name=pk_person, Object type=index
7 Object name=xak_table2, Object type=index
8 Object name=ak_person, Object type=index
9 Object name=xpk_table2, Object type=index
10 Object name=xak_table1, Object type=index
11 Object name=xie03_person, Object type=index
12 Checks-settings File: liquibase.checks-settings.conf
13 -----
14
15 DATABASE CHECKS
17 Validation of the database snapshot found the following issues:
18
                    Custom Check Template (IndexNamingConvention)
19 Check Name:
20 Object Type:
                    index
21 Object Name:
                    xpk_table1
22 Object Location:
                    postgres.public.table1.xpk_table1
```

```
23 Check Severity:
                       BLOCKER (Return code: 4)
24 Message:
                       Found index name xpk_table1 which did not agree with INDEX naming convention ^(xie)[0-9]
   {1,2}(.*)$
25
                       Custom Check Template (IndexNamingConvention)
26 Check Name:
27 Object Type:
                       index
28 Object Name:
                       pk_person
29 Object Location:
                       postgres.public.person.pk_person
30 Check Severity:
                       BLOCKER (Return code: 4)
                       Found index name pk_person which did not agree with INDEX naming convention ^(xie)[0-9]
31 Message:
   {1,2}(.*)$
32
33 Check Name:
                       Custom Check Template (IndexNamingConvention)
                       index
34 Object Type:
35 Object Name:
                       xak_table2
36 Object Location:
                       postgres.public.table2.xak_table2
37 Check Severity:
                       BLOCKER (Return code: 4)
38 Message:
                       Found index name xak_table2 which did not agree with INDEX naming convention ^(xie)[0-9]
   {1,2}(.*)$
39
                       Custom Check Template (IndexNamingConvention)
40 Check Name:
41 Object Type:
                       index
42 Object Name:
                       ak_person
43 Object Location:
                       postgres.public.person.ak_person
44 Check Severity:
                       BLOCKER (Return code: 4)
                       Found index name ak_person which did not agree with INDEX naming convention ^(xie)[0-9]
45 Message:
   {1,2}(.*)$
46
47 Check Name:
                       Custom Check Template (IndexNamingConvention)
48 Object Type:
                       index
49 Object Name:
                       xpk_table2
50 Object Location:
                       postgres.public.table2.xpk_table2
51 Check Severity:
                       BLOCKER (Return code: 4)
52 Message:
                       Found index name xpk_table2 which did not agree with INDEX naming convention ^(xie)[0-9]
   {1,2}(.*)$
53
                       Custom Check Template (IndexNamingConvention)
54 Check Name:
55 Object Type:
                      index
56 Object Name:
                       xak_table1
                       postgres.public.table1.xak_table1
57 Object Location:
58 Check Severity:
                      BLOCKER (Return code: 4)
59 Message:
                       Found index name xak_table1 which did not agree with INDEX naming convention ^(xie)[0-9]
   {1,2}(.*)$
60
61 Check Name:
                       Custom Check Template (IndexNamingConvention)
62 Object Type:
                       index
63 Object Name:
                       xie_table2
64 Object Location:
                       postgres.public.table2.xie_table2
65 Check Severity:
                       BLOCKER (Return code: 4)
66 Message:
                       Found index name xie_table2 which did not agree with INDEX naming convention ^(xie)[0-9]
   {1,2}(.*)$
67
68 Database objects Validated:
69
      Catalog
70
        Column
                           : 18
                           : 2
71
        ForeignKey
72
        Index
                           : 11
73
        PrimaryKey
                           : 4
```

```
74 Schema : 1
75 Table : 4
76 UniqueConstraint : 4
77 To increase details set the --verbose property
78
79 Checks run against database jdbc:postgresql://localhost:5432/postgres:
80 Custom Check Template (Short names: IndexNamingConvention)
```

For now, this check performs as it should. If we need to filter our primary keys and unique constraints, then we will need additional coding for that.

## Task 4: Create a Check for Foreign Key Name Must Start With "xfk"

Let's create a new python script in the Scripts directory and name it xfk.py.

In this script, we will build on our earlier xak.py script.

- We commented out the print statement in line 18.
- Print out all foreign keys found (line 23).
- We updated the code for finding the foreign key name (lines 21-27).
- Compare if the foreign key name contain xfk (line 30).
- If the foreign key name does not match naming convention then we trigger the check (line 31).
- Print out relevant message to the user (lines 33-35).
- · Exit from the check (line 37).

```
1 ### Helpers come from Liquibase
2 import sys
3 import liquibase_utilities
4
5 ### Retrieve log handler
 6 ### Ex. liquibase_logger.info(message)
7 liquibase_logger = liquibase_utilities.get_logger()
8
9 ### Retrieve status handler
10 liquibase_status = liquibase_utilities.get_status()
11
12 ### Retrieve database object
database_object = liquibase_utilities.get_database_object()
14
15 ### Retrieve index object
object_type = database_object.getObjectTypeName().lower()
object_name = database_object.getName().lower()
# print ("Object name=" + object_name + ", Object type=" + object_type)
20 ### Are we executing this script on a uniqueconstraint object?
21 if "foreignkey" in object_type:
22
       print ("Foreign Key name=" + object_name + ", Object type=" + object_type)
23
24
25
       # ### If there is a foreign key, then ...
       fk_name_current = object_name
26
27
       fk_name_expected = "xfk"
```

```
28
29
       ### if the current primary key name not what we expected ...
       if fk_name_expected not in fk_name_current:
30
           liquibase_status.fired = True # indicates the custom check has been triggered
31
32
33
           # set the message of the custom check, which liquibase will return
           t=0 status_message = "Found foreign key name " + f"{fk_name_current}" + " which did not start with " + f"
34
   {fk_name_expected}"
           liquibase_status.message = status_message
36
37
           sys.exit(1) # exit from the check
38
39 ### No foreign keys found in the database.
40 ### For this object we will not trigger this check
41 False
```

Customize another CustomCheckTemplate policy check for foreign key naming convention:

```
1 % liquibase checks customize --check-name=CustomCheckTemplate
3 Short name:
              FKNamingConvention
4 Severity:
              0-4
5 Script description: Foreign Key name
6 Script scope: database
7 Script message: <empty>
8 Script type:
            python
9 Script path:
             Scripts/xfk.py
              <empty>
10 Script_Args:
11 Requires snapshot: false
12
13 % liquibase checks show --check-status=enabled
15 | Short Name | Scope | Status | Severity | Customization
                                                       | Description
17 | FKNamingConvention | database | enabled | 4 | SCRIPT_DESCRIPTION = Foreign | Executes a custom
 check
                     | Key name
18
                                                        | script.
  19
                     | SCRIPT_SCOPE = database
  П
                           | | SCRIPT_MESSAGE = The message |
20
                      | to display when the check is |
21
                     П
22
                     | triggered
  ш
23
                           | SCRIPT_TYPE = PYTHON
  П
24
                     | SCRIPT_PATH = Scripts/xfk.py |
                                  | SCRIPT_ARGS = null
25
                     П
26
                           | REQUIRES_SNAPSHOT = false |
```

```
27 +-----+
```

Let's run this check. If you ran against the database populated with the changelog.sql provided earlier, you would see a single check triggered.

```
1 liquibase checks run --checks-scope=database
3 DATABASE CHECKS
5 Validation of the database snapshot found the following issues:
6
7 Check Name:
                   Custom Check Template (FKNamingConvention)
8 Object Type:
                   foreignKey
                 fk_table1
9 Object Name:
10 Object Location: postgres.public.table2.id.fk_table1
11 Check Severity: BLOCKER (Return code: 4)
                   Found foreign key name fk_table1 which did not start with xfk
12 Message:
13
14 Database objects Validated:
     Catalog
15
                     : 18
: 2
    Column
ForeignKey
17
18
      Index
                       : 4
19
     PrimaryKey
      Schema
                        : 1
20
21
       Table
22
     UniqueConstraint : 4
23 To increase details set the --verbose property
24
25 Checks run against database jdbc:postgresql://localhost:5432/postgres:
   Custom Check Template (Short names: FKNamingConvention)
```

# Task 5: (database) Create a Check for Table Name Must Not Have Special Characters ".\*[\*#+-].\*"

 $\begin{tabular}{ll} Let's create a new python script in the {\tt Scripts} & directory and name it {\tt tableName\_noSpecialCharacters.py}. \\ \end{tabular}$ 

In this script, we will build on our earlier xie.py script.

- We commented out the print statement in line 19.
- Print out all tables found (line 24).
- We updated the code for finding the table name (lines 26-28).
- Compare if the table name contain [\*#+-] (line 31).
- If the table name does not match naming convention then we trigger the check (line 32).
- Print out relevant message to the user (lines 35-37).
- Exit from the check (line 39).

```
### Helpers come from Liquibase
import sys
import liquibase_utilities
import re
```

```
6 ### Retrieve log handler
7 ### Ex. liquibase_logger.info(message)
8 liquibase_logger = liquibase_utilities.get_logger()
9
10 ### Retrieve status handler
11 liquibase_status = liquibase_utilities.get_status()
12
13 ### Retrieve database object
14 database_object = liquibase_utilities.get_database_object()
15
16 ### Retrieve index object
17 object_type = database_object.getObjectTypeName().lower()
18 object_name = database_object.getName().lower()
19 # print ("Object name=" + object_name + ", Object type=" + object_type)
21 ### Are we executing this script on a index object?
22 if "table" in object_type:
23
24
       print ("Table name=" + object_name + ", Object type=" + object_type)
25
       # ### If there is a table, then ...
26
       table_name_current = object_name
27
28
       table_name_expected = "[*#+-]"
29
     ### if the current table name not what we expected ...
     if re.search(table_name_expected, table_name_current):
31
32
     # if table_name_expected not in table_name_current:
33
          liquibase_status.fired = True  # indicates the custom check has been triggered
34
35
           # set the message of the custom check, which liquibase will return
         status_message = "Found table name " + f"{table_name_current}" + " contains one of these special
  characters " + f"{table_name_expected}"
          liquibase_status.message = status_message
37
38
39
           sys.exit(1) # exit from the check
40
41 ### No unique constraints found in the database.
42 ### For this object we will not trigger this check
43 False
```

Customize another CustomCheckTemplate policy check for foreign key naming convention:

```
1 % liquibase checks customize --check-name=CustomCheckTemplate
3 Short name:
               TableNameNoSpecialCharacters
                0-4
4 Severity:
5 Script description: Table name
6 Script scope: database
7 Script message: <empty>
8 Script type: python
               Scripts/tableName_noSpecialCharacters.py
9 Script path:
10 Script_Args:
                <empty>
11 Requires snapshot: false
13 % liquibase checks show --check-status=enabled
15 | Short Name
                        | Scope | Status | Severity | Customization
                                                                         | Description
```

```
17 | TableNameNoSpecialCharacters | database | enabled | 4 | SCRIPT_DESCRIPTION = Table
                                                                      | Executes a
  custom check
               18
                                       1
                                                                      | script.
                                               | name
  1
19
                                               | SCRIPT_SCOPE = database
  П
20
                                               | SCRIPT_MESSAGE = The message
21
                                       | to display when the check is
22
                                | triggered
                                                                      Ī
23
                                               | SCRIPT_TYPE = PYTHON
  П
24
                                               | SCRIPT_PATH =
25
                                | Scripts/tableName_NoSpecialCha |
26
                                | racters.py
  27
                                               | SCRIPT_ARGS = null
                                               | REQUIRES_SNAPSHOT = false
```

Let's run this check. If you ran against the database populated with the <a href="changelog.sql">changelog.sql</a> provided earlier, you would see a single check triggered.

```
1 liquibase checks run --checks-scope=database
2
3 DATABASE CHECKS
5 Validation of the database snapshot found the following issues:
6
7 Check Name:
                      Custom Check Template (TableNameNoSpecialCharacters)
8 Object Type:
                      table
9 Object Name:
                    employee+
10 Object Location: postgres.public.employee+
11 Check Severity:
                      BLOCKER (Return code: 4)
                      Found table name employee+ contains one of these special characters [*#+-]
12 Message:
13
14 Database objects Validated:
15
        Catalog
                         : 1
16
        Column
                          : 18
17
        ForeignKey
                           : 2
18
        Index
                           : 11
19
        PrimaryKey
                           : 4
20
        Schema
                           : 1
21
        Table
22
        UniqueConstraint : 4
23 To increase details set the --verbose property
24
25 Checks run against database jdbc:postgresql://localhost:5432/postgres:
26
     Custom Check Template (Short names: TableNameNoSpecialCharacters)
```