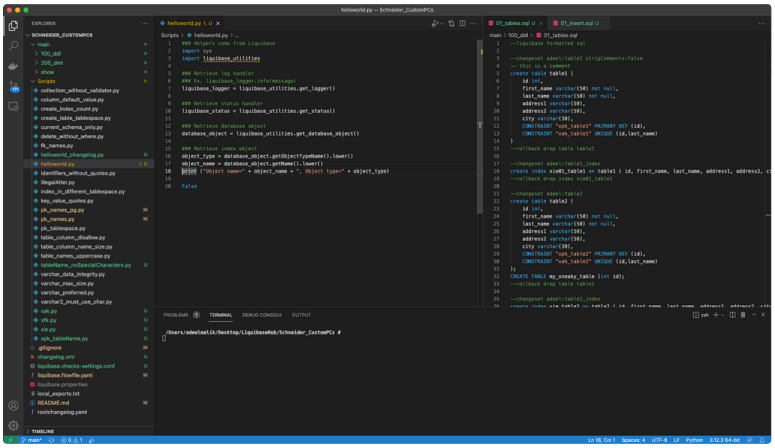
Python Custom Policy Checks Tutorial for Database Checks

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
- Set this property: `checks-scripts-enabled=true`
- James’ custom policy checks repo: https://github.com/liquibase/CustomPCs Connect your Github account

Verify that you have Python3 installed:

```
% python3 --version
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.sql` . 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.

```
--liquibase formatted sql

--changeset adeel:table1
create table table1 (
    id int,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    address1 varchar(50),
    address2 varchar(50),
    city varchar(30),
    CONSTRAINT "xpk_table1" PRIMARY KEY (id),
    CONSTRAINT "xak_table1" UNIQUE (id,last_name)
)
--rollback drop table table1

--changeset adeel:table1_index
create index xie01_table1 on table1 ( id, first_name, last_name, address1, address2, city );
--rollback drop index xie01_table1

--changeset adeel:table2
create table table2 (
    id int,
    first_name varchar(50) not null,
```

```
    last_name varchar(50) not null,
    address1 varchar(50),
    address2 varchar(50),
    city varchar(30),
    CONSTRAINT "xpk_table2" PRIMARY KEY (id),
    CONSTRAINT "xak_table2" UNIQUE (id,last_name)
)
--rollback drop table table2

--changeset adeel:table2_index
create index xie_table2 on table2 ( id, first_name, last_name, address1, address2, city );
--rollback drop index xie_table2

--changeset adeel:person
CREATE TABLE person (
    id int,
    first_name varchar(50) NOT NULL,
    last_name varchar(50) NOT NULL,
    CONSTRAINT "pk_person" PRIMARY KEY (id),
    CONSTRAINT "ak_person" UNIQUE (id,last_name)
)
--rollback drop table person

--changeset adeel:person_index
create index xie03_person on person ( id, first_name, last_name );
--rollback drop index xie03_person

--changeset adeel:fk_table1
alter table table1 add constraint fk_table1 foreign key (id) references table2 (id);
--rollback alter table table1 drop constraint fk_table1;

--changeset adeel:xfk_table2
alter table table2 add constraint xfk_table2 foreign key (id) references table1 (id);
--rollback alter table table2 drop constraint xfk_table2;

--changeset adeel:employee
CREATE TABLE "employee+" (
    id int,
    first_name varchar(50) NOT NULL,
    last_name varchar(50) NOT NULL,
    CONSTRAINT "xpk_employee" PRIMARY KEY (id),
    CONSTRAINT "xak_employee" UNIQUE (id,last_name)
)
--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)
```

```
liquibase_logger = liquibase_utilities.get_logger()

### Retrieve status handler
liquibase_status = liquibase_utilities.get_status()

### Retrieve database object the liquibase policy check is examining
database_object = liquibase_utilities.get_database_object()

### Retrieve object type and object name
object_type = database_object.getObjectTypeName().lower()
object_name = database_object.getName().lower()

### Print object type and object name to console output
print ("Object name=" + object_name + ", Object type=" + object_type)

### Default return code
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:

```
% liquibase checks show
Press [1] to create a new checks settings file

% liquibase checks bulk-set --disable
Confirm: YES

% liquibase checks show --check-status=enabled
+------------+-------+--------+----------+---------------+-------------+
| Short Name | Scope | Status | Severity | Customization | Description |
+------------+-------+--------+----------+---------------+-------------+
```

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.
```

```
Script type:        python
Script path:        Scripts/helloworld.py
Script_Args:        <empty>
Requires snapshot:  false

% liquibase checks show --check-status=enabled
+------------+----------+---------+----------+--------------------------------+------------------------------
-+
| Short Name | Scope    | Status  | Severity | Customization                  | Description
|
+------------+----------+---------+----------+--------------------------------+------------------------------
-+
| HelloWorld | database | enabled | 4        | SCRIPT_DESCRIPTION = List all  | Executes a custom check
|
|            |          |         |          | objects                        | script.
|
|            |          |         |          | SCRIPT_SCOPE = database        |
|
|            |          |         |          | SCRIPT_MESSAGE = This check    |
|
|            |          |         |          | will never trigger. This is a  |
|
|            |          |         |          | first custom policy check.     |
|
|            |          |         |          | SCRIPT_TYPE = PYTHON           |
|
|            |          |         |          | SCRIPT_PATH =                  |
|
|            |          |         |          | Scripts/helloworld.py          |
|
|            |          |         |          | SCRIPT_ARGS = null             |
|
|            |          |         |          | REQUIRES_SNAPSHOT = false      |
|
+------------+----------+---------+----------+--------------------------------+------------------------------
-+

```

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=table
Object name=first_name, Object type=column
```

```
Object name=xak_employee, Object type=uniqueconstraint
Object name=public, Object type=schema
Object name=xak_table2, Object type=index
Object name=first_name, Object type=column
Object name=last_name, Object type=column
Object name=xie_table2, Object type=index
Object name=id, Object type=column
Object name=xak_table2, Object type=uniqueconstraint
Object name=address1, Object type=column
Object name=first_name, Object type=column
Object name=postgres, Object type=catalog
Object name=address2, Object type=column
Object name=xie01_table1, Object type=index
Object name=pk_person, Object type=index
Object name=xfk_table2, Object type=foreignkey
Object name=fk_table1, Object type=foreignkey
Object name=person, Object type=table
Object name=xpk_table1, Object type=primarykey
Object name=xpk_employee, Object type=index
Object name=ak_person, Object type=index
Object name=id, Object type=column
Object name=last_name, Object type=column
Object name=last_name, Object type=column
Object name=xpk_table2, Object type=index
Object name=xak_employee, Object type=index
Object name=id, Object type=column
Object name=employee+, Object type=table
Object name=xpk_table2, Object type=primarykey
Object name=xie03_person, Object type=index
Object name=pk_person, Object type=primarykey
Object name=xpk_employee, Object type=primarykey
Object name=city, Object type=column
Object name=xak_table1, Object type=uniqueconstraint
Object name=ak_person, Object type=uniqueconstraint
Object name=xak_table1, Object type=index
Checks-settings File:      liquibase.checks-settings.conf
================================================================================
Database objects Validated:
     Catalog             : 1
     Column              : 18
     ForeignKey          : 2
     Index               : 11
     PrimaryKey          : 4
     Schema              : 1
     Table               : 4
     UniqueConstraint    : 4
To increase details set the --verbose property

Checks run against database jdbc:postgresql://localhost:5432/postgres:
  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).

Copy/paste the following python code:

```
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
13 database_object = liquibase_utilities.get_database_object()
14
15 ### Retrieve database object type and object name
16 object_type = database_object.getObjectTypeName().lower()
17 object_name = database_object.getName().lower()
18 print ("Object name=" + object_name + ", Object type=" + object_type)
19
20 ### Are we executing this script on a table object?
21 if "table" in database_object.getObjectTypeName().lower():
22
23     ### Get table name
24     table_name = database_object.getName().lower()
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:
36             liquibase_status.fired = True       # indicates the custom check has been triggered
37
38             # set the message of the custom check, which liquibase will return
39             status_message = "Found primary key name " + f"{pk_name_current}" + " which did not match what was
   expected: " + f"{pk_name_expected}"
40             liquibase_status.message = status_message
41
42             sys.exit(1)     # exit from the check
```

```
43
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
4  Severity:           0-4
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
14 +--------------------+----------+---------+----------+--------------------------------+-----------------------
   ---------+
15 | Short Name         | Scope    | Status  | Severity | Customization                  | Description
   |
16 +--------------------+----------+---------+----------+--------------------------------+-----------------------
   ---------+
17 | PKNamingConvention | database | enabled | 4        | SCRIPT_DESCRIPTION = Primary   | Executes a custom
   check        |
18 |                    |          |         |          | Key name                       | script.
   |
19 |                    |          |         |          | SCRIPT_SCOPE = database        |
   |
20 |                    |          |         |          | SCRIPT_MESSAGE = The message   |
   |
21 |                    |          |         |          | to display when the check is   |
   |
22 |                    |          |         |          | triggered                      |
   |
23 |                    |          |         |          | SCRIPT_TYPE = PYTHON           |
   |
24 |                    |          |         |          | SCRIPT_PATH =                  |
   |
25 |                    |          |         |          | Scripts/xpk_tableName.py       |
   |
26 |                    |          |         |          | SCRIPT_ARGS = null             |
   |
27 |                    |          |         |          | REQUIRES_SNAPSHOT = false      |
   |
28 +--------------------+----------+---------+----------+--------------------------------+-----------------------
   ---------+
```

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
4 ---------------
5 Validation of the database snapshot found the following issues:
```

```
6
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)
12 Message:            Found primary key name pk_person which did not match what was expected: xpk_person
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               : 4
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: 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).

Copy/paste the following python code:

```
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
13 database_object = liquibase_utilities.get_database_object()
14
15 ### Retrieve database object type and object name
16 object_type = database_object.getObjectTypeName().lower()
17 object_name = database_object.getName().lower()
18 # print ("Object name=" + object_name + ", Object type=" + object_type)
19
```

```
### Are we executing this script on a uniqueconstraint object?
if "uniqueconstraint" in object_type:

    # ### If there is a unique constraint, then ...
    uc_name_current = object_name
    uc_name_expected = "xak_"

    ### if the current primary key name not what we expected ...
    if uc_name_expected not in uc_name_current:
        liquibase_status.fired = True       # indicates the custom check has been triggered

        # set the message of the custom check, which liquibase will return
        status_message = "Found unique constraint name " + f"{uc_name_current}" + " which did not start with "
+ f"{uc_name_expected}"
        liquibase_status.message = status_message

        sys.exit(1)     # exit from the check

### No unique constraints found in the database.
### For this object we will not trigger this check
False
```

Customize another `CustomCheckTemplate` policy check for unique constraint naming convention:

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

Short name:         UCNamingConvention
Severity:           0-4
Script description: Unique Constraint name
Script scope:       database
Script message:     <empty>
Script type:        python
Script path:        Scripts/xak.py
Script_Args:        <empty>
Requires snapshot:  false

% liquibase checks show --check-status=enabled
+--------------------+----------+---------+----------+--------------------------------+-----------------------
---------+
| Short Name         | Scope    | Status  | Severity | Customization                  | Description
|
+--------------------+----------+---------+----------+--------------------------------+-----------------------
---------+
| UCNamingConvention | database | enabled | 4        | SCRIPT_DESCRIPTION = Unique    | Executes a custom
check        |
|                    |          |         |          | Constraint name                | script.
|
|                    |          |         |          | SCRIPT_SCOPE = database        |
|
|                    |          |         |          | SCRIPT_MESSAGE = The message   |
|
|                    |          |         |          | to display when the check is   |
|
|                    |          |         |          | triggered                      |
|
|                    |          |         |          | SCRIPT_TYPE = PYTHON           |
|
|                    |          |         |          | SCRIPT_PATH =                  |
|
```

```
|                    |          |         |          | Scripts/xak_tableName.py       |
|
|                    |          |         |          | SCRIPT_ARGS = null             |
|
|                    |          |         |          | 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:

Check Name:         Custom Check Template (UCNamingConvention)
Object Type:        uniqueConstraint
Object Name:        ak_person
Object Location:    postgres.public.person.id.ak_person
Check Severity:     BLOCKER (Return code: 4)
Message:            Found unique constraint name ak_person which did not start with xak_

Database objects Validated:
     Catalog             : 1
     Column              : 18
     ForeignKey          : 2
     Index               : 11
     PrimaryKey          : 4
     Schema              : 1
     Table               : 4
     UniqueConstraint    : 4
To increase details set the --verbose property

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:

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

### Retrieve log handler
### Ex. liquibase_logger.info(message)
liquibase_logger = liquibase_utilities.get_logger()

### Retrieve status handler
liquibase_status = liquibase_utilities.get_status()

### Retrieve database object
database_object = liquibase_utilities.get_database_object()

### Retrieve index object
object_type = database_object.getObjectTypeName().lower()
object_name = database_object.getName().lower()
# print ("Object name=" + object_name + ", Object type=" + object_type)

### Are we executing this script on a index object?
if "index" in object_type:

    print ("Index name=" + object_name + ", Object type=" + object_type)

    # ### If there is a unique constraint, then ...
    index_name_current = object_name
    index_name_expected = "^(xie)[0-9]{1,2}(.*)$"

    ### if the current primary key name not what we expected ...
    if not (re.search(index_name_expected, index_name_current)):
    # if index_name_expected not in index_name_current:
        liquibase_status.fired = True       # indicates the custom check has been triggered

        # 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}"
        liquibase_status.message = status_message

        sys.exit(1)     # exit from the check

### No unique constraints found in the database.
### For this object we will not trigger this check
False
```

Customize another `CustomCheckTemplate` policy check for index naming convention:

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

Short name:         IndexNamingConvention
Severity:           0-4
Script description: Index name
Script scope:       database
Script message:     <empty>
Script type:        python
Script path:        Scripts/xie.py
Script_Args:        <empty>
```

```
Requires snapshot:  false

% liquibase checks show --check-status=enabled
+----------------------+----------+---------+----------+--------------------------------+--------------------
------------+
| Short Name            | Scope    | Status  | Severity | Customization                  | Description
|
+----------------------+----------+---------+----------+--------------------------------+--------------------
------------+
| IndexNamingConvention | database | enabled | 4        | SCRIPT_DESCRIPTION = Index     | Executes a custom
check        |
|                       |          |         |          | name                           | script.
|
|                       |          |         |          | SCRIPT_SCOPE = database        |
|
|                       |          |         |          | SCRIPT_MESSAGE = The message   |
|
|                       |          |         |          | to display when the check is   |
|
|                       |          |         |          | triggered                      |
|
|                       |          |         |          | SCRIPT_TYPE = PYTHON           |
|
|                       |          |         |          | SCRIPT_PATH = Scripts/xie.py   |
|
|                       |          |         |          | SCRIPT_ARGS = null             |
|
|                       |          |         |          | 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:

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

Object name=xpk_table1, Object type=index
Object name=xie01_table1, Object type=index
Object name=xie02_table2, Object type=index
Object name=pk_person, Object type=index
Object name=xak_table2, Object type=index
Object name=ak_person, Object type=index
Object name=xpk_table2, Object type=index
Object name=xak_table1, Object type=index
Object name=xie03_person, Object type=index
Checks-settings File:      liquibase.checks-settings.conf
================================================================================

DATABASE CHECKS
---------------
Validation of the database snapshot found the following issues:

Check Name:         Custom Check Template (IndexNamingConvention)
Object Type:        index
Object Name:        xpk_table1
Object Location:    postgres.public.table1.xpk_table1
```

```
Check Severity:     BLOCKER (Return code: 4)
Message:            Found index name xpk_table1 which did not agree with INDEX naming convention ^(xie)[0-9]{1,2}(.*)$

Check Name:         Custom Check Template (IndexNamingConvention)
Object Type:        index
Object Name:        pk_person
Object Location:    postgres.public.person.pk_person
Check Severity:     BLOCKER (Return code: 4)
Message:            Found index name pk_person which did not agree with INDEX naming convention ^(xie)[0-9]{1,2}(.*)$

Check Name:         Custom Check Template (IndexNamingConvention)
Object Type:        index
Object Name:        xak_table2
Object Location:    postgres.public.table2.xak_table2
Check Severity:     BLOCKER (Return code: 4)
Message:            Found index name xak_table2 which did not agree with INDEX naming convention ^(xie)[0-9]{1,2}(.*)$

Check Name:         Custom Check Template (IndexNamingConvention)
Object Type:        index
Object Name:        ak_person
Object Location:    postgres.public.person.ak_person
Check Severity:     BLOCKER (Return code: 4)
Message:            Found index name ak_person which did not agree with INDEX naming convention ^(xie)[0-9]{1,2}(.*)$

Check Name:         Custom Check Template (IndexNamingConvention)
Object Type:        index
Object Name:        xpk_table2
Object Location:    postgres.public.table2.xpk_table2
Check Severity:     BLOCKER (Return code: 4)
Message:            Found index name xpk_table2 which did not agree with INDEX naming convention ^(xie)[0-9]{1,2}(.*)$

Check Name:         Custom Check Template (IndexNamingConvention)
Object Type:        index
Object Name:        xak_table1
Object Location:    postgres.public.table1.xak_table1
Check Severity:     BLOCKER (Return code: 4)
Message:            Found index name xak_table1 which did not agree with INDEX naming convention ^(xie)[0-9]{1,2}(.*)$

Check Name:         Custom Check Template (IndexNamingConvention)
Object Type:        index
Object Name:        xie_table2
Object Location:    postgres.public.table2.xie_table2
Check Severity:     BLOCKER (Return code: 4)
Message:            Found index name xie_table2 which did not agree with INDEX naming convention ^(xie)[0-9]{1,2}(.*)$

Database objects Validated:
     Catalog             : 1
     Column              : 18
     ForeignKey          : 2
     Index               : 11
     PrimaryKey          : 4
```

```
     Schema              : 1
     Table               : 4
     UniqueConstraint    : 4
To increase details set the --verbose property

Checks run against database jdbc:postgresql://localhost:5432/postgres:
  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).

Copy/paste the following python code:

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

### Retrieve log handler
### Ex. liquibase_logger.info(message)
liquibase_logger = liquibase_utilities.get_logger()

### Retrieve status handler
liquibase_status = liquibase_utilities.get_status()

### Retrieve database object
database_object = liquibase_utilities.get_database_object()

### Retrieve index object
object_type = database_object.getObjectTypeName().lower()
object_name = database_object.getName().lower()
# print ("Object name=" + object_name + ", Object type=" + object_type)

### Are we executing this script on a uniqueconstraint object?
if "foreignkey" in object_type:

    print ("Foreign Key name=" + object_name + ", Object type=" + object_type)

    # ### If there is a foreign key, then ...
    fk_name_current = object_name
    fk_name_expected = "xfk"
```

```
28
29      ### if the current primary key name not what we expected ...
30      if fk_name_expected not in fk_name_current:
31          liquibase_status.fired = True       # indicates the custom check has been triggered
32
33          # set the message of the custom check, which liquibase will return
34          status_message = "Found foreign key name " + f"{fk_name_current}" + " which did not start with " + f"
    {fk_name_expected}"
35          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
2
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
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 | FKNamingConvention | database | enabled | 4        | SCRIPT_DESCRIPTION = Foreign   | Executes a custom
   check        |
18 |                    |          |         |          | Key name                       | script.
   |
19 |                    |          |         |          | SCRIPT_SCOPE = database        |
   |
20 |                    |          |         |          | SCRIPT_MESSAGE = The message   |
   |
21 |                    |          |         |          | to display when the check is   |
   |
22 |                    |          |         |          | triggered                      |
   |
23 |                    |          |         |          | SCRIPT_TYPE = PYTHON           |
   |
24 |                    |          |         |          | SCRIPT_PATH = Scripts/xfk.py   |
   |
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.

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

DATABASE CHECKS
---------------
Validation of the database snapshot found the following issues:

Check Name:         Custom Check Template (FKNamingConvention)
Object Type:        foreignKey
Object Name:        fk_table1
Object Location:    postgres.public.table2.id.fk_table1
Check Severity:     BLOCKER (Return code: 4)
Message:            Found foreign key name fk_table1 which did not start with xfk

Database objects Validated:
     Catalog             : 1
     Column              : 18
     ForeignKey          : 2
     Index               : 11
     PrimaryKey          : 4
     Schema              : 1
     Table               : 4
     UniqueConstraint    : 4
To increase details set the --verbose property

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 ".*[*#+-].*"

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

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).

Copy/paste the following python code:

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

```

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

### Retrieve status handler
liquibase_status = liquibase_utilities.get_status()

### Retrieve database object
database_object = liquibase_utilities.get_database_object()

### Retrieve index object
object_type = database_object.getObjectTypeName().lower()
object_name = database_object.getName().lower()
# print ("Object name=" + object_name + ", Object type=" + object_type)

### Are we executing this script on a index object?
if "table" in object_type:

    print ("Table name=" + object_name + ", Object type=" + object_type)

    # ### If there is a table, then ...
    table_name_current = object_name
    table_name_expected = "[*#+-]"

    ### if the current table name not what we expected ...
    if re.search(table_name_expected, table_name_current):
    # if table_name_expected not in table_name_current:
        liquibase_status.fired = True       # indicates the custom check has been triggered

        # 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

        sys.exit(1)     # exit from the check

### No unique constraints found in the database.
### For this object we will not trigger this check
False
```

Customize another `CustomCheckTemplate` policy check for foreign key naming convention:

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

Short name:         TableNameNoSpecialCharacters
Severity:           0-4
Script description: Table name
Script scope:       database
Script message:     <empty>
Script type:        python
Script path:        Scripts/tableName_noSpecialCharacters.py
Script_Args:        <empty>
Requires snapshot:  false

% liquibase checks show --check-status=enabled
+------------------------------+----------+---------+----------+--------------------------------+-------------------+
| Short Name                   | Scope    | Status  | Severity | Customization                  | Description       |
```

```
+------------------------------+----------+---------+----------+--------------------------------+-------------
------------------+
| TableNameNoSpecialCharacters | database | enabled | 4        | SCRIPT_DESCRIPTION = Table     | Executes a
custom check        |
|                              |          |         |          | name                           | script.
|
|                              |          |         |          | SCRIPT_SCOPE = database        |
|
|                              |          |         |          | SCRIPT_MESSAGE = The message   |
|
|                              |          |         |          | to display when the check is   |
|
|                              |          |         |          | triggered                      |
|
|                              |          |         |          | SCRIPT_TYPE = PYTHON           |
|
|                              |          |         |          | SCRIPT_PATH =                  |
|
|                              |          |         |          | Scripts/tableName_NoSpecialCha |
|
|                              |          |         |          | racters.py                     |
|
|                              |          |         |          | SCRIPT_ARGS = null             |
|
|                              |          |         |          | 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:

Check Name:         Custom Check Template (TableNameNoSpecialCharacters)
Object Type:        table
Object Name:        employee+
Object Location:    postgres.public.employee+
Check Severity:     BLOCKER (Return code: 4)
Message:            Found table name employee+ contains one of these special characters [*#+-]

Database objects Validated:
     Catalog             : 1
     Column              : 18
     ForeignKey          : 2
     Index               : 11
     PrimaryKey          : 4
     Schema              : 1
     Table               : 4
     UniqueConstraint    : 4
To increase details set the --verbose property

Checks run against database jdbc:postgresql://localhost:5432/postgres:
  Custom Check Template (Short names: TableNameNoSpecialCharacters)
```