Wykorzystanie Pythona przy administrowaniu bazami danych Oracle

Wysyłanie mailem miesięcznych raportów i utworzenie wykresów wydajnościowych

Anna Kozub

Politechnika Krakowska, wydział Fizyki, Matematyki i Informatyki

Spis Treści

Wprowadzenie do problematyki projektu

Cel

Niniejszy projekt podzielony jest na dwie częsci. Celem pierwszej części jest zautomatyzowanie podstawowych czynności administratorskich, takich jak sprawdzenie czy backupy na bazie danych zosatły wykonane poprawnie, ile jest sesji w stanie 'inactive', a także zajętość FRA. Są to trzy podstawowe elementy, które są kluczowe dla zdrowia bazy.

Druga część to wizualne przedtawienie podstawowych problemów wydajnościowych, to jest blokujące sesje, które przebywają w tym stanie dłużej niż 1000 sekund, a także obciążenia bazy, czyli sprawdzenie ile sesji aktualnie podłączonych jest do bazy, itp.

Biblioteki niezbędne do współpracy z bazą Oracle

Biblioteka cx_Oracle

Cx_Oracle to moduł, który umożliwia dostęp do bazy danych Oracle. Został przetestowany w wersjach 11g.* i 12c.*. Jest on udostępniany na mocy open-sourcowej licencji BSD.

Aby móc z niej skorzystać na początku naszego kodu umieszczamy linię:

```
import cx_Oracle
```

Biblioteka os

Kolejną bardzo przydatną biblioteką jest biblioteka os. Pozwala ona na interakcję z systemem operacyjnym i będzie nam potrzebna do ustawienia zmiennych środowiskowych. Tak jak w powyższym przypadku, aby ją wykorzystać, dodajemy na początku naszego kodu linię:

```
import os
```

Dostęp do danych z bazy

Ustawienie zmiennych środowiskowych i podłączenie do bazy

Po tym gdy mamy już zaimportowane odpowiednie moduły, możemy przejść do podłączenia się do bazy. Pierwsze co ustawiamy ORACLE_HOME oraz ścieżkę do bibliotek Oracle:

```
os.putenv('ORACLE_HOME', 'ścieżka do oracle_home')
```

```
os.putenv('LD_LIBRARY_PATH', 'ścieżka do bibiotek Oracle')
```

Następnie podłączamy się do bazy:

```
db = cx_Oracle.connect('hr', 'hrpwd', 'localhost/xe')
```

Wykorzystanie poleceń SQLowych

Aby pobrać dane z bazy posługujemy się poleceniami SQLowymi. Najpierw tworzymy kursor, którego użyjemy do wykonania operacji na bazie. Następnie przygotowywujemy polecenie SQLowe, a w końcu przechwytujemy wynik zapytania i przypisujemy go do zmiennej o nazwie res. Otrzymany wynik zapytania to krotka. Na samym końcu zamykamy kursor.

```
cur = con.cursor()

cur.execute('select * from tabela')

res=cur.fetchall()

print res

cur.close()
```

Przesyłanie emailem outputu poleceń SQLowych

Wprowadzenie do problematyki

Po tym gdy zaimplementowaliśmy wykonanie poleceń SQLowych zwracających nam dane wydajnościowe, chcielibyśmy mieć do nich dostęp bez konieczności logowania się do serweru i pobierania ich stamtąd.

Jednym z rozwiązań tego problemu może być przesłanie naszych 'raportów' emailem. Umożliwia nam to zastosowanie sepcjalnych bibliotek i poleceń w naszym kodzie.

Bibloteki

By uzyskać działający kod, przede wszystkim musiemy uwzględnić bibliotekę:

import smtplib

Następnie korzystamy również z następujących modułów:

```
from email.message import Message
from email.encoders import encode_base64
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
```

Wysyłanie emaila z załącznikiem

Gdy już dołączyliśmy odpowiednie biblioteki, możemy przystąpić do wysłania emaila z naszego serwera SMTP:

```
msg = MIMEMultipart()
msg['From'] = 'Reports Service <reports@company.intranet>'
msg['To'] = 'receipients@company.intranet'
msg['Subject'] = 'Temat'

attachment.add_header('Content-Disposition', 'attachment;filename=nazwa)
msg.attach(attachment)

emailserver = smtplib.SMTP("localhost")
emailserver.sendmail(msg['From'], msg['To'], msg.as_string())
emailserver.quit()
```

Część pierwsza projektu - wyjaśnienie działania prgramu

Wstępne omówienie - część 1

Po tym gdy zapoznaliśmy się z podstawowymi założeniami dostępu do bazy i wysyłania maili w Pythonie, czas najwyższy omówić program.

Przede wszystkim pierwsza część programu zajmuje się wyciągbięciem z bazy takich informacji jak:

1)nieaktywne sesje (powstają one np wtedy gdy ktoś zapomni wyłączyć SqlDeveloper, zostawi włączony komputer i pójdzie do domu) - takie sesje, nic nie robią, jednak trzymają zasoby bazy, niepotrzebnie ją obciążając. Administator może chcieć zadecydować o zabiciu takich sesji.

2)rozmiar FRA- FRA to logiczna struktura na naszym dysku, w której przechowywane są backupy, archivelogi oraz flashbacklogi. Zapchanie się FRA może spowodować bardzo duże problemy jak zawiśnięcie bazy, nie wspominając o braku możliwości wykonania nowych backupów. Dlatego bardzo ważne jest śledzenie jego zużycia.

Wstępne omówienie - część 2

3) Backupy - każda baza danych musi mieć backupy, tak by w razie awarii można było odzyskać jak największą ilość danych. Dlatego bardzo ważne jest śledzenie backupów i ich poprawnego wykonania. Przydatną wiedzą dla administratora jest jednak nie tylko to czy ostatnie backupy są wykonane poprawnie, ale również to jak długo zajęło ich wykonywanie, rozmiar backupsetów (w danym okresie) czy też dane pozwalające zlokalizować awarię. Wszytskie te informacje pozwalają na lepsze zrozumienie specyfiki bazy i lepsze dostrojenie jej.

Działanie programu

W napisanym przeze mnie programie najpierw załączam odpowiednie biblioteki, przygotowuję odpowiednie polecenia SQLowe, następnie ich output wysyłam w załączniku za pomocą omówionych wcześniej poleceń.

Wykresy

Plotly

Jak wcześniej wspomniałam we wstępie, druga część projektu to przedstawienie problemów wydajnościowych za pomocą wykresów.

By utworzyć wykres należy dołączyć na początku swojego kodu linie:

import plotly.plotly as py

import plotly.graph_objs as go

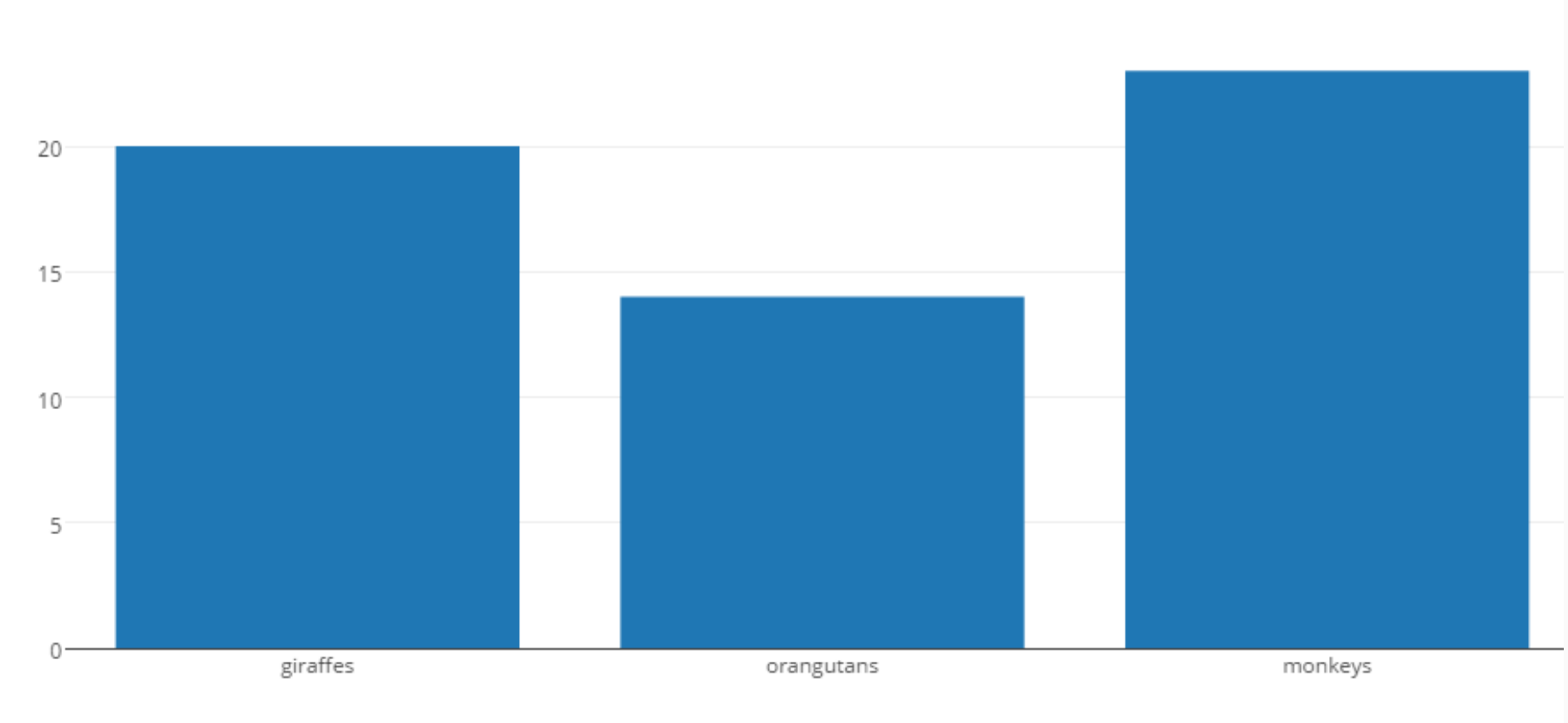
Następnie musimy zastanowić się nad stylem naszych wykresów (ja wybrałam słupkowe).

Jak utworzyć prosty wykres?

Najprostszym sposobem na utworzenie wykresu słupkowego jest zaimportowanie wcześniej wspomnainaych bibliotek oraz przypisanie danych do osi x oraz y.

```
import plotly.plotly as py

import plotly.graph_objs as go

data = [go.Bar(

x=['giraffes', 'orangutans', 'monkeys'],

y=[20, 14, 23]

)]

py.iplot(data, filename='basic-bar')
```

Otrzymany wykres



Część druga projektu - wyjaśnienie działania prgramu

Wstępne omówienie

Kolejnym ważnym problemem w utrzymaniu sprawnie działającej bazy jest sprawdzanie jej obciążenia.

Dwa podstawowe zgadnienia to:

1) Blocking sessions - sesje blokujące - dla niektórych środowisk blocking sessions mogą być mordercze, zwłaszcza te przedłużające się. Czas, podczas którego sesja jest zbalokowana, który staje się niebezpieczny dla bazy, jest sepcyficzny dla środowiska, ale zazwyczaj oscyluje koło 1000 sekund (tak więc przyjęłam w moim programie).

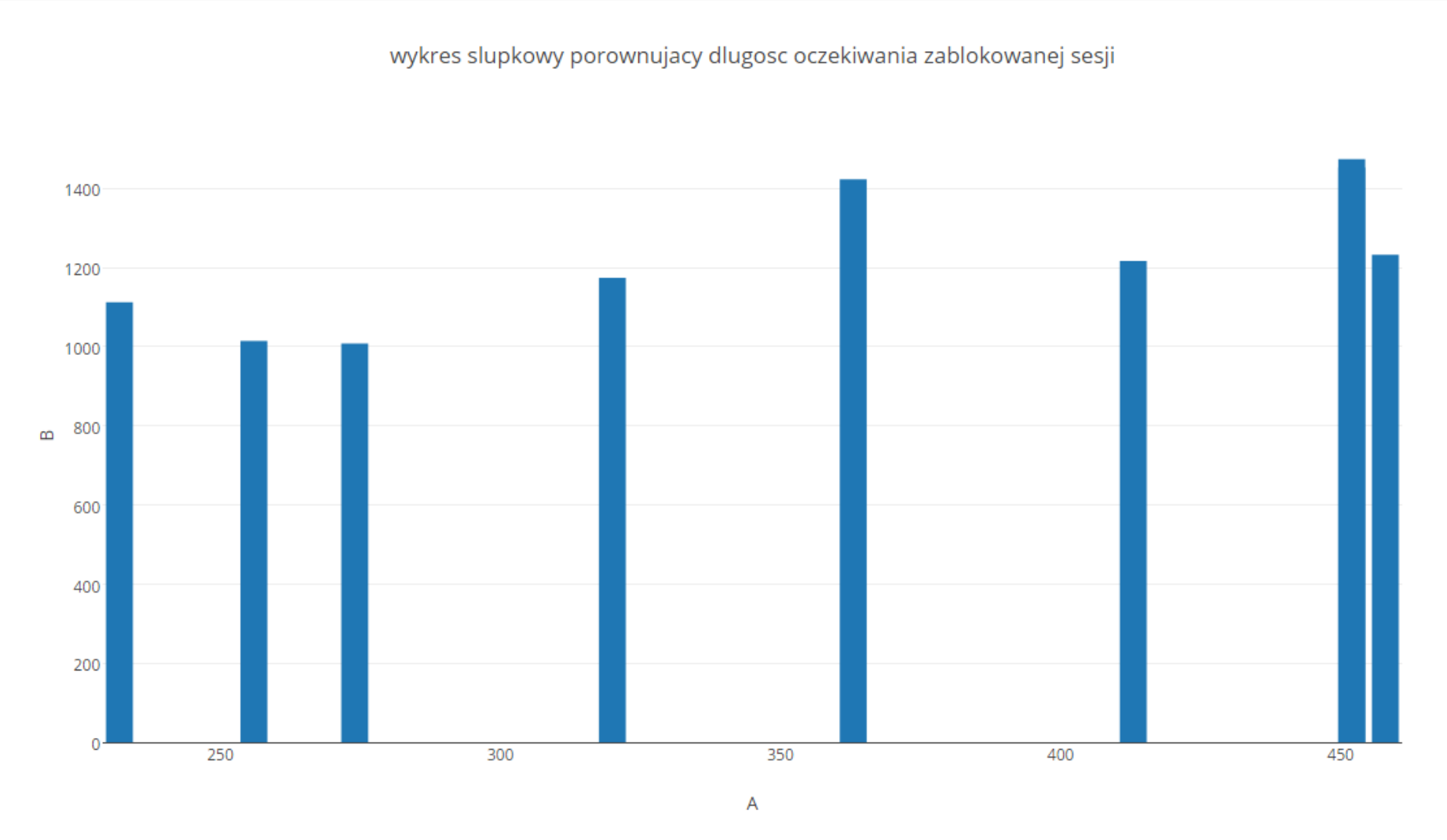
2) Następnie ważne jest sprawdzenie obciążenia bazy pod względem ilości sesji, procesów, locków itp. Ważne do odnotowanie jest, że gdy zostanie wykorzystana maksymalna pula sesji, nikt więcej nie może podłączyć się do bazy, co może spowodować poważne straty w biznesie.

Działanie programu

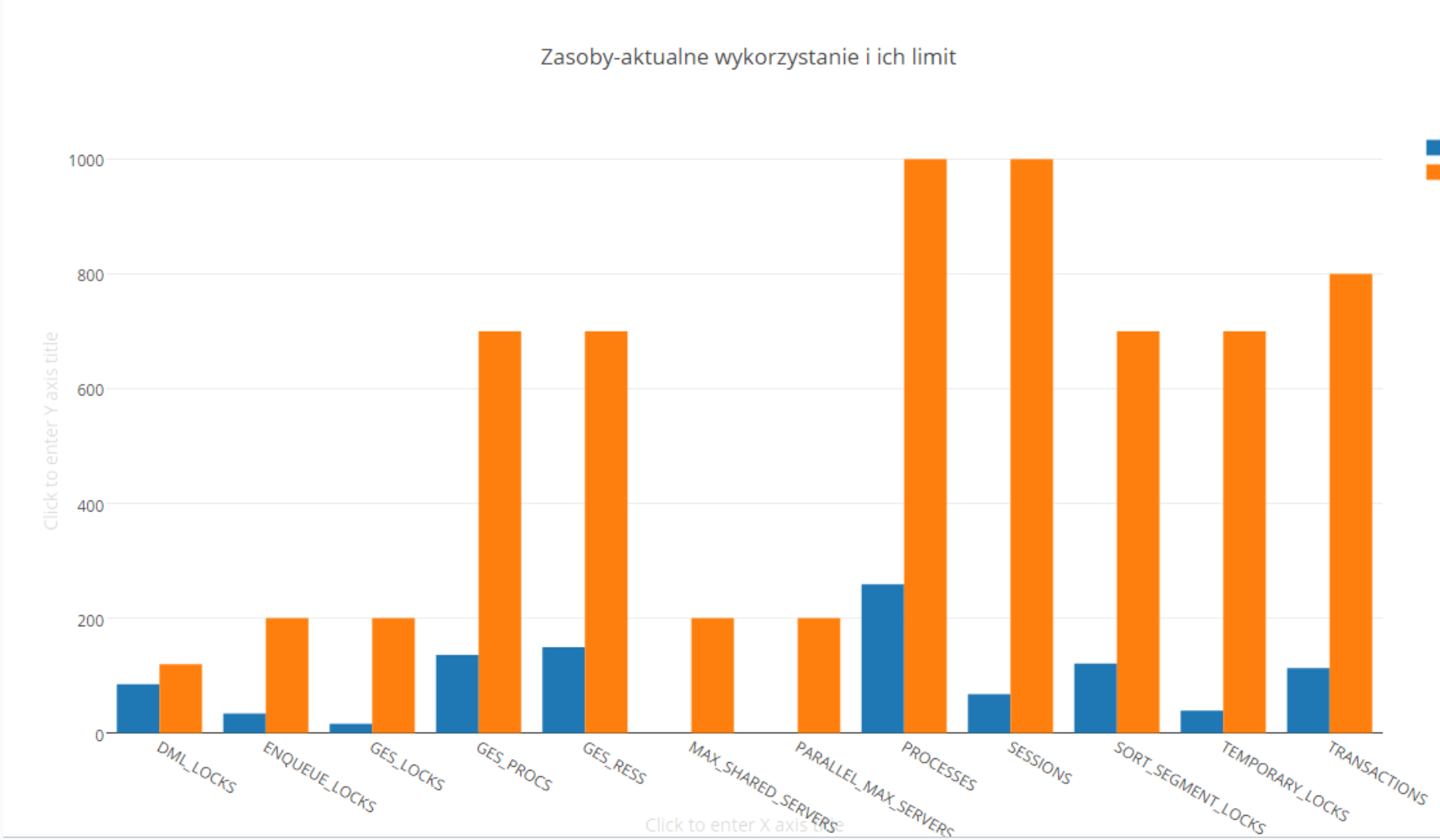
W napisanym przeze mnie programie najpierw załączam odpowiednie biblioteki, przygotowuję odpowiednie polecenia SQLowe, następnie otrzymane krotki konwertuję do listy, którą to z kolei wykorzystuję do uzyskania dwóch wykresów.

W dalszych dwóch slajdach załączam przykładowe wykresy uzyskane na bazie testowej.

Wykres blokujących sesji



Wykres obciążenia



Pytania?