Artículo
Jose-Tomas Salvador · 6 oct, 2022 Lectura de 6 min

Open Exchange

# Cómo crear Procedimientos Almacenados usando Python Embebido

Python se ha convertido en el lenguaje de programación más utilizado del mundo (fuente: https://www.tiobe.com/tiobe-index/) y SQL sigue siendo el líder como lenguaje para las bases de datos. ¿No sería genial que Python y SQL trabajaran juntos para ofrecer nuevas funcionalidades que SQL por sí mismo no puede? Después de todo, Python tiene más de 380.000 librerías publicadas (fuente: https://pypi.org/) con funciones muy interesantes para ampliar las consultas SQL dentro de Python.

En este artículo detallo cómo crear nuevos Procedimientos Almacenados de SQL en la base de datos de InterSystems IRIS usando Embedded Python.

## Librerías de Python utilizadas como muestras

En este artículo utilizaré dos librerías muy útiles para cualquiera que trabaje con SQL en IRIS: Geopy y Chronyk.

**Geopy** es una librería utilizada para aplicar la geocodificación (asignación de coordenadas geográficas a direcciones) a los datos de direcciones. Con ella es posible, a partir del nombre de la calle, obtener el código postal y la dirección completa, en el formato de la oficina de correos. Es muy útil, ya que muchos registros tienen una dirección.

**Chronyk** se utiliza para procesar fechas y horas mediante el lenguaje humano. Esto es muy útil, porque internamente, tanto para IRIS como para Python, una fecha es un número que representa la cantidad de tiempo que ha transcurrido desde una fecha inicial. Para los humanos, una fecha es el 20 de julio, o ayer, o mañana, o hace dos horas. Chronyk acepta recibir la fecha así y luego la convierte al formato de fecha universal.

## Soporte de Python en InterSystems IRIS

Desde la versión 2021.2 es posible utilizar Python para crear métodos de clase, procedimientos almacenados, producciones de interoperabilidad y llamadas nativas entre Python e IRIS (ObjectScript) de forma bidireccional. No conozco ninguna otra plataforma de datos que trabaje tan estrechamente con Python. El requisito para que esto funcione es que Python se instale en la misma máquina física o virtual o contenedor que IRIS. Consulta más detalles en: https://docs.intersystems.com/iris20221/csp/docbook/DocBook.UI.Page.cls?...

Para instalar Python ejecuta:

```
# install libraries required for python and pip
RUN apt-get -y update \
  && apt-get -y install apt-utils \
  && apt-get install -y build-essential unzip pkg-config wget \
  && apt-get install -y python3-pip
```

## Soporte de librerías de Python en InterSystems IRIS

Para que InterSystems IRIS pueda utilizar una librería de Python es obligatorio que esté instalada dentro de <installdir>/mgr/python. installdir es la carpeta en la que está instalado IRIS. Para instalar nuevos paquetes ejecuta:

```
# use pip3 (the python zpm) to install geopy and chronyk packages
RUN pip3 install --upgrade pip setuptools wheel
RUN pip3 install --target /usr/irissys/mgr/python geopy chronyk
```

Pip3 es el administrador e instalador de paquetes más popular de Python, Pip.

# Cómo crear Procedimientos Almacenados en lenguaje de Python

Una de las posibilidades de utilizar Python en InterSystems IRIS es crear Procedimientos Almacenados usando Python.

Hay dos posibilidades:
1. Crear un Procedimiento Almacenado en Python utilizando la sentencia SQL de crear función o procedimiento
2. Crear un Método de clase dentro de la clase ObjectScript con las etiquetas sqlProc y language=Python.

## Cómo crear un Procedimiento Almacenado en Python utilizando la sentencia SQL de Crear Procedimiento

Según la documentación de InterSystems, también se puede escribir una función SQL o un Procedimiento Almacenado utilizando Python Embebido si se especifica el argumento LANGUAGE PYTHON en la sentencia CREATE, como se muestra a continuación (fuente: https://docs.intersystems.com/iris20221/csp/docbook/DocBook.UI.Page.cls?...):

```
CREATE FUNCTION tzconvert(dt TIMESTAMP, tzfrom VARCHAR, tzto VARCHAR)
    RETURNS TIMESTAMP
    LANGUAGE PYTHON
{
    from datetime import datetime
    from dateutil import parser, tz
    d = parser.parse(dt)
    if (tzfrom is not None):
        tzf = tz.gettz(tzfrom)
        d = d.replace(tzinfo = tzf)
    return d.astimezone(tz.gettz(tzto)).strftime("%Y-%m-%d %H:%M:%S")
}
```

Cuando se ejecuta esta nueva función SQL:

```
SELECT tzconvert(now(), 'US/Eastern', 'UTC')
```

La función devuelve algo como:

```
2022-07-20 15:10:05
```

## Cómo crear el Método de clase dentro de la clase ObjectScript con las etiquetas sqlProc y language=Python

Confieso que este enfoque es mi favorito: crear un método de clase con las etiquetas sqlProc y language=Python.

En mi opinión es más fácil de mantener, está mejor documentado y con una mejor administración de las versiones del código fuente. Para este enfoque he publicado una aplicación de muestra: https://openexchange.intersystems.com/package/Python-IRIS-SQL-Procedures.... La utilizaré para mostrar este segundo enfoque con detalle.

## Instalación de la aplicación de muestra

Para instalar la aplicación de muestra, sigue estos pasos:

1. Clonar/git pull el repositorio en cualquier directorio local: $ git clone https://github.com/yurimarx/iris-sql-python-sample.git
2. Abrir un terminal de Docker en este directorio y ejecutar: $ docker-compose build
3. Ejecutar el contenedor de IRIS: $ docker-compose up -d

Otra posibilidad para instalar es utilizar el ZPM:

```
zpm "install iris-sql-python-sample"
```

## Ejemplos de Procedimientos Almacenamiento usando Python

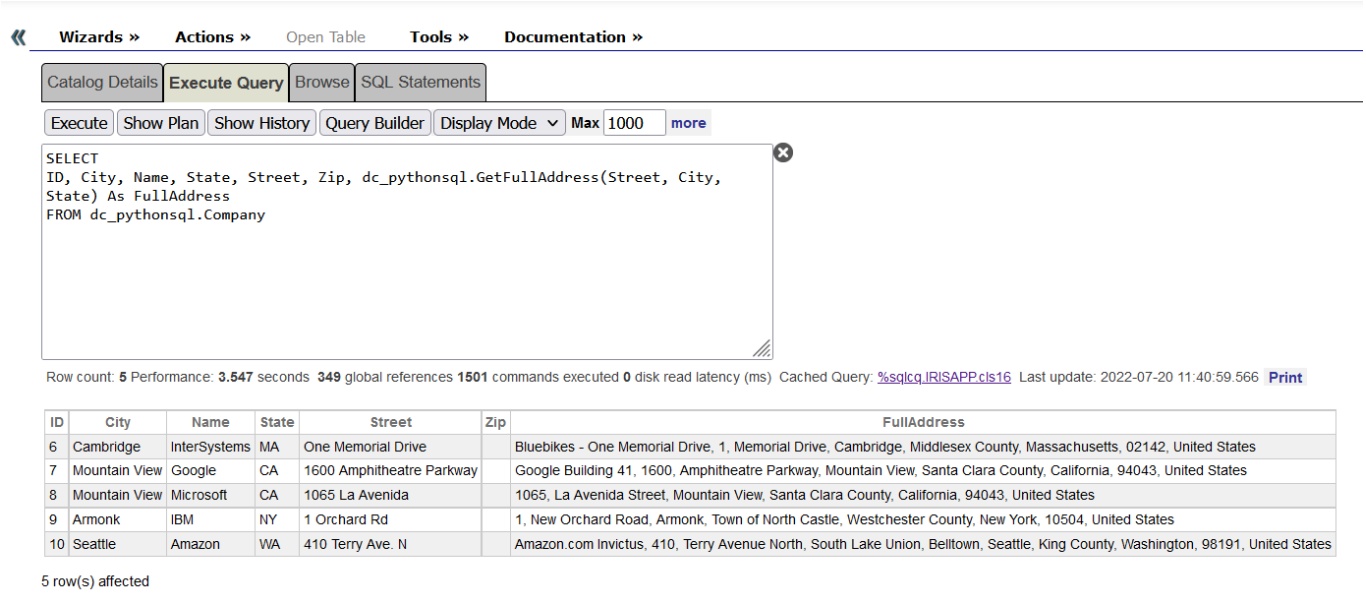
El primer ejemplo es un Procedimiento Almacenado para administrar la geocodificación de direcciones, consulta el código fuente:

```
  ClassMethod GetFullAddress(Street As %String, City As %String, State As %String)
As %String [ Language = python, SqlName = GetFullAddress, SqlProc ]
{
  import geopy.geocoders
  from geopy.geocoders import Nominatim
  geopy.geocoders.options.default_timeout = 7
  geolocator = Nominatim(user_agent="intersystemsiris"  )
  location = geolocator.geocode(Street + ", " + City + ", " + State, country_codes="US")
  return location.address
}
```

Fíjate que se declaró un Método de clase (dentro de la clase dc.pythonsql.Company) con las etiquetas [ Language = python, SqlProc]. La etiqueta SqlName permite establecer un nombre para el nuevo Procedimiento Almacenado en sentencias SQL. Ve al Portal de administración, Sistema > SQL y ejecuta el siguiente código:

```
SELECT ID, City, Name, State, Street, Zip, dc_pythonsql.GetFullAddress(Street, City,
State) As FullAddress
FROM dc_pythonsql.Company
```

Verás estos resultados:




| ID | City | Name | State | Street | Zip | FullAddress |
|---|---|---|---|---|---|---|
| 6 | Cambridge | InterSystems | MA | One Memorial Drive | | Bluebikes - One Memorial Drive, 1, Memorial Drive, Cambridge, Middlesex County, Massachusetts, 02142, United States |
| 7 | Mountain View | Google | CA | 1600 Amphitheatre Parkway | | Google Building 41, 1600, Amphitheatre Parkway, Mountain View, Santa Clara County, California, 94043, United States |
| 8 | Mountain View | Microsoft | CA | 1065 La Avenida | | 1065, La Avenida Street, Mountain View, Santa Clara County, California, 94043, United States |
| 9 | Armonk | IBM | NY | 1 Orchard Rd | | 1, New Orchard Road, Armonk, Town of North Castle, Westchester County, New York, 10504, United States |
| 10 | Seattle | Amazon | WA | 410 Terry Ave. N | | Amazon.com Invictus, 410, Terry Avenue North, South Lake Union, Belltown, Seattle, King County, Washington, 98191, United States |

Ahora las direcciones incompletas regresan como direcciones "completas" (completas y calificadas).

*Nota: Si no devuelve nada, ejecuta #class(dc.pythonsql.Company).CreateFiveCompanies(). Creará cinco empresas para utilizar en las pruebas.*

Este paquete puede funcionar con los principales servicios de geocodificación abiertos y comerciales. En este ejemplo utilizamos el servicio abierto **Nominatim**, pero es posible utilizar Bing, Google, ArcGIS y otros. Consulta las posibilidades en https://geopy.readthedocs.io/en/stable/#module-geopy.geocoders.

El segundo ejemplo es un paquete de fecha y hora en formato humanizado, Chronyk. Permite enviar frases como "mañana", "ayer", "dentro de 4 horas", "4 de julio del 2022" y obtener el resultado en formato de fecha universal. Mira la creación del Procedimiento Almacenado:

```
ClassMethod GetHumanDate(Sentence As %String) As %String [ Language = python, SqlName =
GetHumanDate, SqlProc ]
{
  from chronyk import Chronyk
  t = Chronyk(Sentence)
  return t.ctime()
}
```

Realiza la siguiente llamada en el Portal de administración > Sistema > SQL:

```
SELECT ID, City, Name, State, Street, Zip, dc_pythonsql.GetHumanDate('yesterday') As
Datetime FROM dc_pythonsql.Company
```

Verás resultados como este:

Si quieres simplemente llamar al Procedimiento Almacenado, puedes usar esta sentencia SQL:

```
select dc_pythonsql.GetHumanDate('yesterday') as Datetime
```

Esta librería tiene varias posibilidades de fechas y horas humanizadas, consulta https://github.com/KoffeinFlummi/Chronyk.

Así que es fácil crear Procedimientos Almacenados en Python. ¡Disfrútalo!

#Embedded Python #Mejores prácticas #Python #SQL #InterSystems IRIS #InterSystems IRIS for Health
Ir a la aplicación en InterSystems Open Exchange

---

**URL de fuente:**https://es.community.intersystems.com/post/c%C3%B3mo-crear-procedimientos-almacenados-usando-python-embebido