Python and Ingres: Pyngres

Roy Hann

roy.hann@pyngres.com

Actian User Day
Birmingham, May 1st, 2025

Everyone is Using Python

TIOBE
the software quality company

Apr 2025	Apr 2024	Change	Programming Language	Ratings	Change
1	1		Python	23.08%	+6.67%
2	3	↑	C++	10.33%	+0.56%
3	2	↓	C	9.94%	-0.27%
4	4		Java	9.63%	+0.69%
5	5		C#	4.39%	-2.37%
6	6		JavaScript	3.71%	+0.82%
7	7		Go	3.02%	+1.17%
8	8		Visual Basic	2.94%	+1.24%
9	11	↑	Delphi/Object Pascal	2.53%	+1.06%
10	9	↓	SQL	2.19%	+0.57%
11	10	↓	Fortran	2.04%	+0.57%

PINGRES

Python Access to Ingres

- **PyODBC**
  - Open Database Connectivity
  - suited to DBMS-agnostic BI, reporting, and analytics
    - for other purposes ODBC is only better than nothing
  - no support for asynchronous operation
  - Ingres extensions not supported
    - in particular:
      - no support for repeated queries
      - no support for bulk data loading
      - no support for cost tracing (lock_trace, io_trace, etc.)
      - no provision for other tracing (set qep, trace point qe90, etc)
  - JDBC is cut from the same unsatisfactory cloth

PINGRES

Python Access to Ingres

- **`python-ingresdbi`**
  - implements the Python DB-API 2.0 standard
    - PEP 249
  - written in C, extends Python
  - uses ODBC under the covers
    - see previous slide
  - verrrrrry stale
    - 15 years since the last commit

PINGRES

Python Access to Ingres

- SQLAlchemy
  - useful for *ad hoc* querying and analytics
    - e.g. Jupyter
  - otherwise vile
    - PugSQL is not a sufficient deodorant
- Don’t get me started on ORMs
  - a crime against data
    - someone should be on trial in The Hague
- AnyIngres
  - I have heard of it
  - it uses ODBC/JDBC/.Net internally
  - I am not sure one can even get hold of it

PINGRES

So…Not Everyone is Using Python

- …not to build serious Ingres applications
- There is nothing very credible or attractive for building enterprise business applications using Python and Ingres
  - no matter how nice Python is

PINGRES

Introducing Pyngres

- Pyngres: a pure Python binding for the Actian Ingres OpenAPI
  - an alternative to using ODBC to access Ingres
    - offers some unique features and benefits
- Needs an Ingres client installation
  - uses **iigcn** and the OpenAPI library
  - can use **iigcc**
- Currently supported on:
  - Windows
  - Linux
  - MacOS/Darwin

PYNGRES

Pyngres Example

```
IIdemo_init(&envHandle);

/*
**  Connect with no connection parameters.
*/
printf( "apisconn: establishing connection\n" );

connParm.co_genParm.gp_callback = NULL;
connParm.co_genParm.gp_closure = NULL;
connParm.co_target =  argv[1];
connParm.co_connHandle = envHandle;
connParm.co_tranHandle = NULL;
connParm.co_type = IIAPI_CT_SQL;
connParm.co_username = NULL;
connParm.co_password = NULL;
connParm.co_timeout = -1;

IIapi_connect( &connParm );

while( connParm.co_genParm.gp_completed == FALSE )
   IIapi_wait( &waitParm );

connHandle = connParm.co_connHandle;
tranHandle = connParm.co_tranHandle;
```

C

```
envHandle = IIdemo_init()

##  connect with no connection parameters
print(f'{script}: establishing connection')
cop = IIAPI_CONNPARM()
cop.co_genParm.gp_callback = None
cop.co_genParm.gp_closure = None
cop.co_target = target
cop.co_connHandle = envHandle
cop.co_tranHandle = None
cop.co_type = IIAPI_CT_SQL
cop.co_username = None
cop.co_password = None
cop.co_timeout = -1
IIapi_connect(cop)
while not cop.co_genParm.gp_completed:
    IIapi_wait(wtp)
connHandle = cop.co_connHandle
tranHandle = cop.co_tranHandle
```

Python

Demonstration

A walk-through of a minimal "Hello World" application,
presented using Jupyter

`https://github.com/quelgeek/pyngres/blob/main/examples/pyngres-demo-1.ipynb`

PYNGRES

Getting the Feel of Pyngres

- There are 27 example C programs in **`$II_SYSTEM/ingres/demo/api`** showing how to use the OpenAPI
- I have re-implemented most of them in Python
  - `https://github.com/quelgeek/pyngres/tree/main/examples`
- Anyone at all familiar with the OpenAPI can read a C example alongside the Python example

PYNGRES

Ingres Data in Python

- The OpenAPI trafficks in binary data
  - even C programs need to format dates, money, decimal etc from Ingres for human consumption
  - Python programs need to convert all data
- I have provided the **`iitypes`** package to marshal data between binary and Python objects
  - supports all (currently) available Ingres types
    - including BLObs (LONG types)
    - excluding geospatial (for now) and OME
  - e.g. **`varchar`** is converted to/from Python **`str`**
  - e.g. **`ansidate`** is converted to/from Python **`datetime.date`**

PINGRES

Ingres Data in Python

```
import iitypes as ii
...
##  set up the tuple buffer list
tuple = {}
columnData = (py.IIAPI_DATAVALUE * gdp.gd_descriptorCount)()
for column_index in range(gdp.gd_descriptorCount):
    descriptor = gdp.gd_descriptor[column_index]
    clone = type(descriptor).from_buffer_copy(descriptor)
    descriptor = clone
    buffer_allocator = ii.allocator_for_type(descriptor)
    buffer = buffer_allocator(descriptor=descriptor)
    columnData[column_index] = buffer.datavalue
    columnName = descriptor.ds_columnName.decode()
    tuple[columnName] = buffer
```

Yeah, that's pretty dense, but it will handle all cases. After you bury it in a method you'll never have to think about it again

PINGRES

Synchronous versus Asynchronous

- The OpenAPI is intrinsically asynchronous
- To use it synchronously you have to follow each call with a busy-wait loop

```
py.IIapi_connect(cop)
while not cop.co_genParm.gp_completed:
    py.IIapi_wait(wtp)
```

- You can't even check for errors until the **gp_completed** flag is set
- Sure, the OpenAPI is asynchronous, but it is hard to get the benefit of that using C

PINGRES

Python Makes it Easy

- The OpenAPI is a very natural fit with the **asyncio** features of Python
  - no multi-threading required

Instead of this:

```
py.IIapi_connect(cop)
while not cop.co_genParm.gp_completed:
    py.IIapi_wait(wtp)
```

Do this:

```
await py.IIapi_connect(cop)
```

- Using the **await** syntax tells Python to switch to some other runnable task
  - control will return automatically when the OpenAPI call is completed

PINGRES

Pyngres Three Ways

- After installing **pyngres** in your environment:
  - **`import pyngres as py`**
    - straightforward binding
    - good for prototyping an eventual C implementation
  - **`import pyngres.blocking as py`**
    - as above but without explicit busy-wait loops
    - good for writing simple Python applications
  - **`import pyngres.asyncio as py`**
    - fully asynchronous using **`asyncio`**
    - good for implementing Python-based servers and GUIs—anything that needs to cooperate with an event loop

PYNGRES

Demonstration

10 concurrent Ingres sessions running in a single thread, executing TPC-C workload, using asyncio

`https://github.com/quelgeek/tpyc_c`

PYNGRES

Reactive

- Python ain't C
  - it has other virtues
- **`pyngres.asyncio`** is not notably fast
  - but it is very reactive
  - facilitates a good user-experience
    - without the complications of multi-threading
    - e.g. use **`PySide6.QtAsyncio`** to run Qt and Pyngres on the same event loop
    - rich, responsive GUI applications interacting with Ingres
- Pyngres is *plenty* fast enough for:
  - user-facing GUIs
  - data-acquisition devices and IoT applications

PYNGRES

Secure

- Pyngres is naturally resistant to most SQL-injection exploits
  - because Ingres is naturally resistant
- Middleware can introduce attack surfaces even if your DBMS is as resistant as Ingres
  - avoiding middleware is avoiding vulnerability

PYNGRES

What's Missing?

- Not much
  - ~~I lost interest in **IIapi_getCopyMap()**~~
    - ~~it probably works but I haven't tested it~~
    - ~~...I suppose I probably should~~
  - Geospatial types—but just say the word!
  - I didn't bother with Python implementations of the **apiasvr.c** and **apiaclnt.c** examples
    - using **pyngres.asyncio** is so slick I'd be doing a disservice by encouraging multithreading
- That really is all that's missing
  - I even got asynchronous callbacks to Python functions to work

It's tested now, and it works—woo-hoo!

PYNGRES

What Next?

- There is a pure Python driver coming
  - not sure when but not today
- Pyngres is potentially useful until then
- Bare OpenAPI is (very) hard work
  - Pyngres is only enabling, it's not The Answer™
  - I have plenty of ideas
    - someone could create **asyncdb-ingres**
    - someone could re-do **python-ingresdbi**
    - or someone could do something quite slick
  - at the very least, I'll add a bunch of convenience functions

PYNGRES

Special Mention

- I don’t know who wrote the *OpenAPI User Guide* but it has proven to be as accurate and complete as it is laconic
  - what it doesn’t say is sometimes significant
- Now that I know how to use the OpenAPI I realize everything I needed to know was in the guide all along!
  - except what I didn’t, which wasn’t...

PINGRES

Questions?

PINGRES

Resources

- Install Pyngres from PyPI
  - **`pip install pyngres`**
  - **`pip install iitypes`**
- Clone **`tpyc_c`** and **`tpyc_c_db`** from GitHub

`tpyc_c`

`tpyc_c_db`

- Contact me via `roy.hann@pyngres.com`

PYNGRES