

Workshop Spatial, Graph y Machine Learning en base de Datos Oracle

HOL3 Graph

Contenidos

| | |
|---|----------|
| WORKSHOP SPATIAL, GRAPH Y MACHINE LEARNING EN BASE DE DATOS ORACLE | 1 |
| HOL3 GRAPH | 1 |
| REQUERIMIENTOS INICIALES | 3 |
| ESCRITORIO REMOTO CON MICROSOFT WINDOWS | 3 |
| ESCRITORIO REMOTO CON MACOS | 4 |
| PROPERTY GRAPH (2 HORAS)..... | 7 |
| ANALÍTICA DE PROPERTY GRAPH | 7 |
| DESCRIPCIÓN Y OBJETIVO DEL TALLER | 7 |
| DESCRIPCIÓN DEL MODELO DE DATOS | 8 |
| DESCRIPCIÓN DEL ENTORNO DEL WORKSHOP | 10 |
| INSTRUCCIONES DEL WORKSHOP | 11 |
| CONTENIDO DE LOS NOTEBOOKS..... | 18 |
| 1.PGX.RingsAndFraud.ipynb | 18 |
| 2.PGX.RingsAndFraud.Community.ipynb | 22 |
| 3.PGX.Visualization.visjs.BankPGView.ipynb (opcional)..... | 27 |
| RESUMEN..... | 28 |
| <i>Más información</i> | 28 |



Requerimientos iniciales

Para la realización de este workshop se necesita un cliente de *Remote Desktop* de Windows.

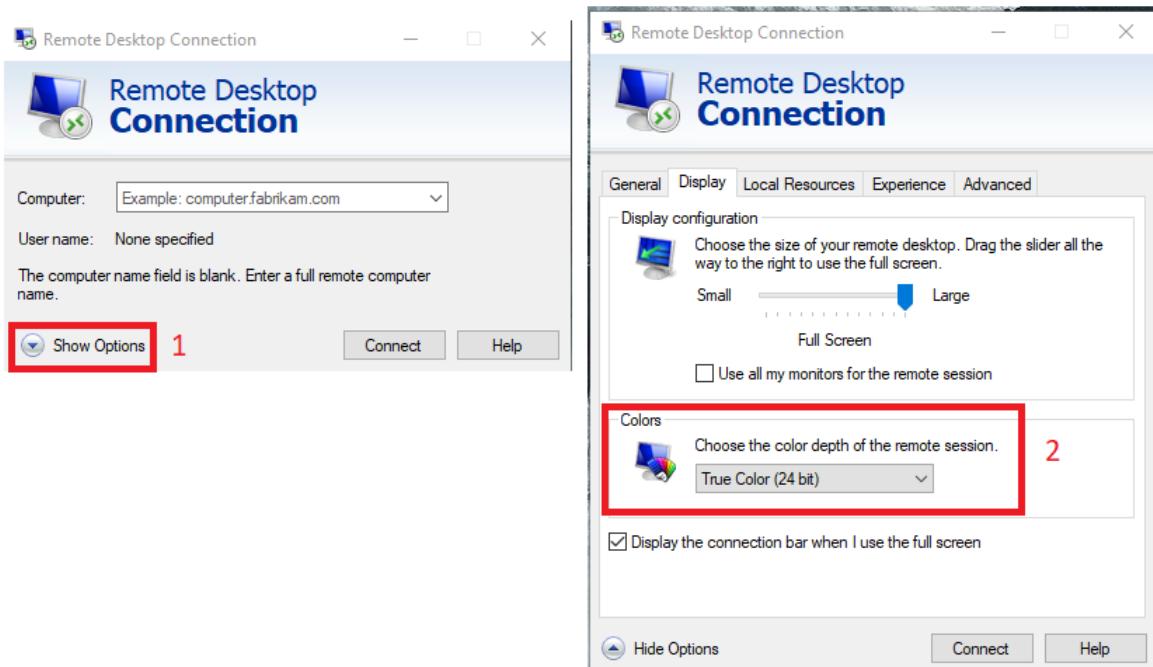
Este cliente está instalado por defecto en el sistema operativo Microsoft Windows, existiendo también clientes compatibles para MacOs y Linux.

La máquina del workshop se proporciona para uso individual de cada participante del workshop siendo para uso exclusivo del mismo.

Esta máquina está alojada en la nube pública **Oracle Cloud Infrastructure** (OCI) estando prohibida la reproducción o alteración de sus contenidos fuera de lo previsto en este manual de usuario.

Escrítorio Remoto con Microsoft Windows

En la configuración del cliente de Escritorio Remoto es importante especificar el uso de *True Color (24 bit)* para evitar problemas con algunas herramientas. Desde el botón *Show Options*, como se muestra a continuación:



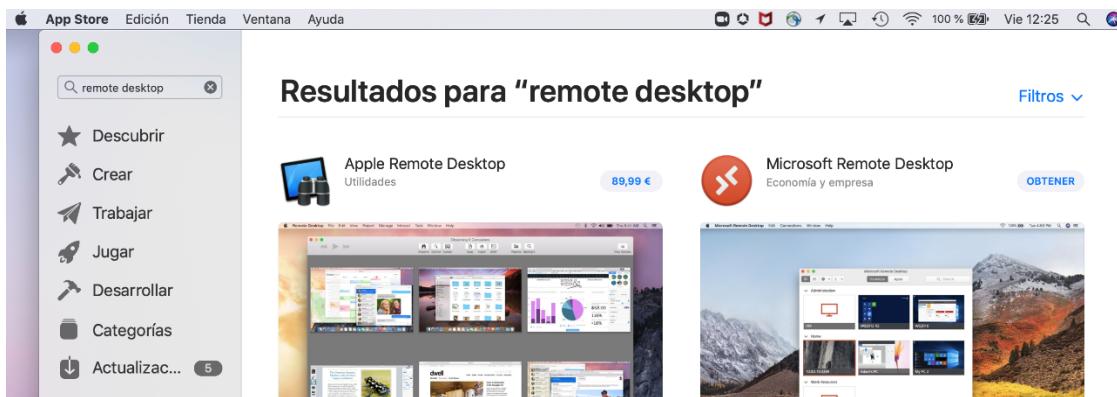
Como parte de la documentación del workshop se facilitarán los siguientes datos:

- Nombre de usuario y password
- IP pública de la maquina del workshop

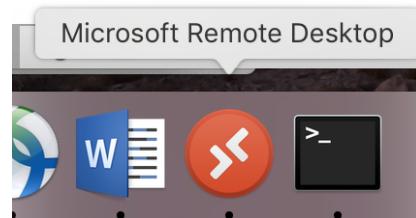
Escritorio Remoto con MacOS

En MacOs, la aplicación para conectar a escritorio remoto de Windows no viene instalada por defecto, pero está disponible en el App Store de manera gratuita.

Buscando “remote desktop” se encuentra como “*Microsoft Remote Desktop*” tal y como se muestra a en la siguiente captura:



Una vez instalada esta aplicación, aparecerá un ícono como el siguiente en la barra de aplicaciones para poder realizar las conexiones.

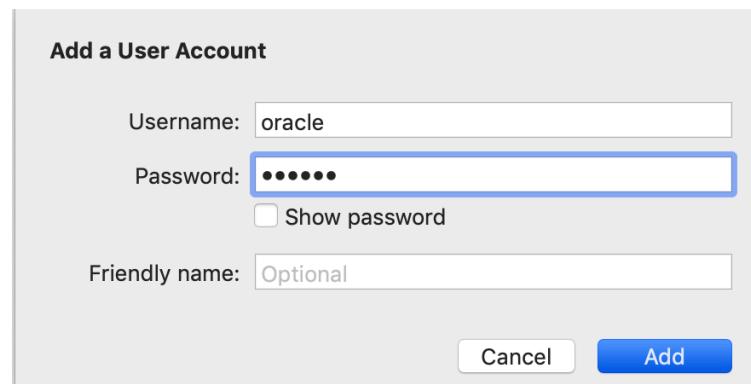
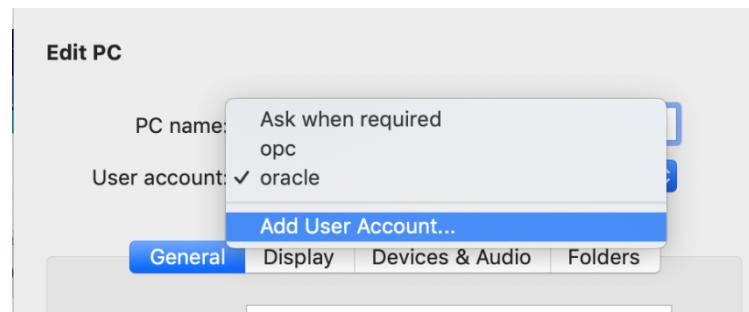


Como parte de la documentación del workshop se facilitarán los siguientes datos:

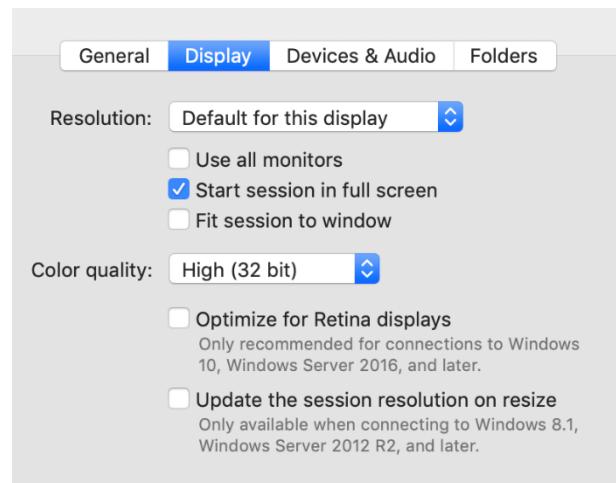
- Nombre de usuario y password
- IP pública de la maquina del workshop

Con los cuales se configura como se muestra a continuación.
Añadimos el usuario y la password proporcionados:



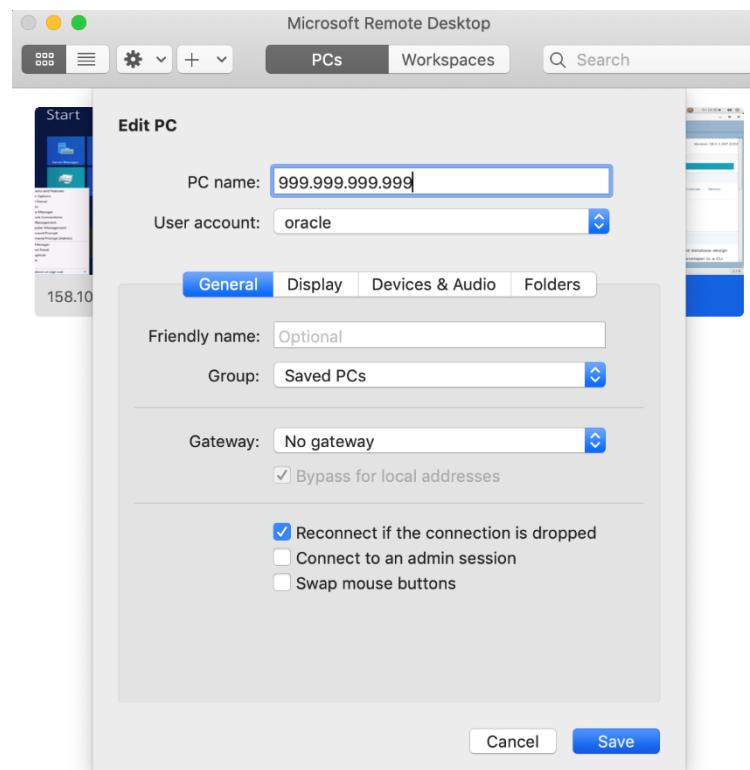


En la pestaña de Display se confirma que la calidad de color está seleccionada a 32 bits:



Se introduce la IP pública en ‘PC name’, se guarda el acceso con el botón Save y ya está preparado el acceso a la máquina virtual del workshop.



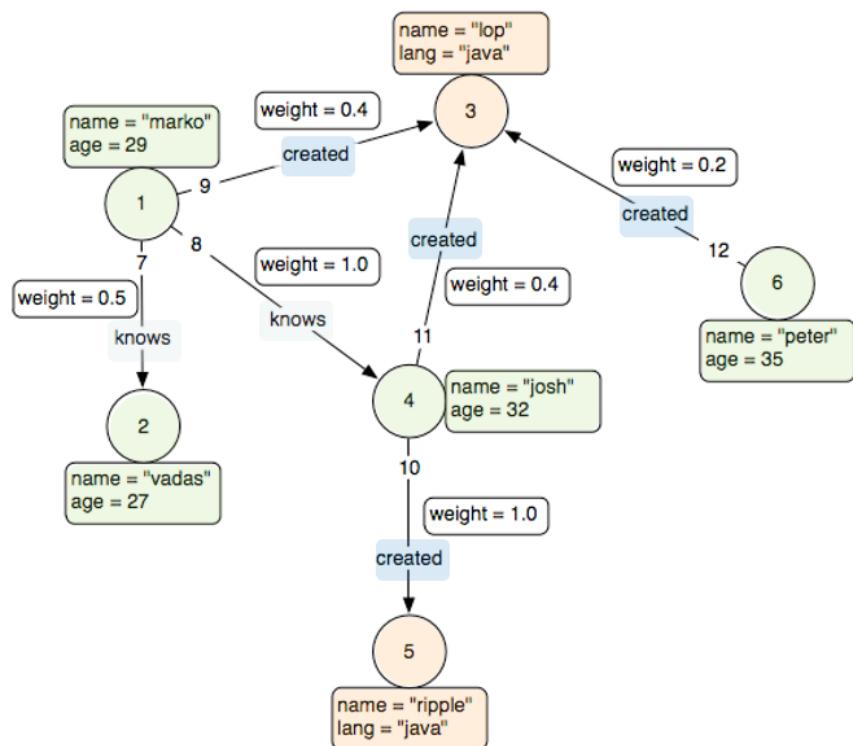


Property Graph (2 horas)

Analítica de Property Graph

La analítica de *property graph* permite encontrar información que está en las relaciones directas e indirectas entre los elementos de los datos.

En este tipo de analítica el modelo de datos representa las entidades como vértices y sus relaciones como aristas. Cada una de ellas puede opcionalmente tener varios atributos como muestra el siguiente ejemplo sencillo:



Descripción y objetivo del taller

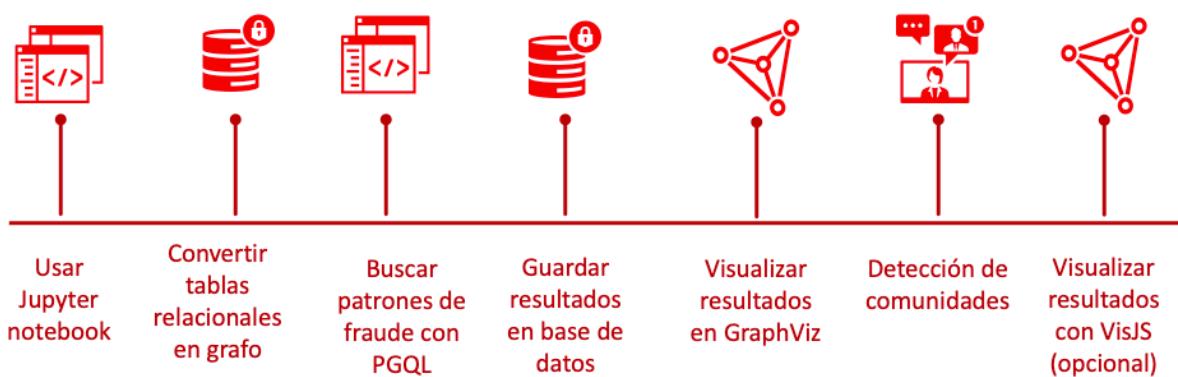
En este taller usted va a llevar a cabo un caso de uso de estudio y análisis de un juego de datos de clientes y transacciones con el objetivo de identificar una posible organización de fraude, así como sus ramificaciones que afectan a otros clientes que podrían estar participando sin saberlo en esta actividad sospechosa de fraude.

Para ello realizará las siguientes actividades:



- Transformación del juego de datos que se encuentra almacenado en tablas de base de datos Oracle en dos diferentes modelos de *property graph* (de ahora en adelante grafo).
- Búsqueda de patrones sospechosos de organización de fraude usando lenguaje de consulta de grafos. En las ocurrencias encontradas se estimará el impacto económico que podría tener el fraude.
- Visualización de los patrones encontrados en forma de tabla y de grafo.
- Detección de la comunidad que está participando con la organización de fraude encontrada en el primer paso.
- Visualización geográfica de la comunidad encontrada para estudiar su radio de acción y así poder tomar las medidas oportunas.

De manera resumida, estas serán las diferentes etapas del taller:



Descripción del modelo de datos

Todos los datos contenidos en estas tablas son completamente ficticios; cualquier coincidencia con la realidad es puramente casual.

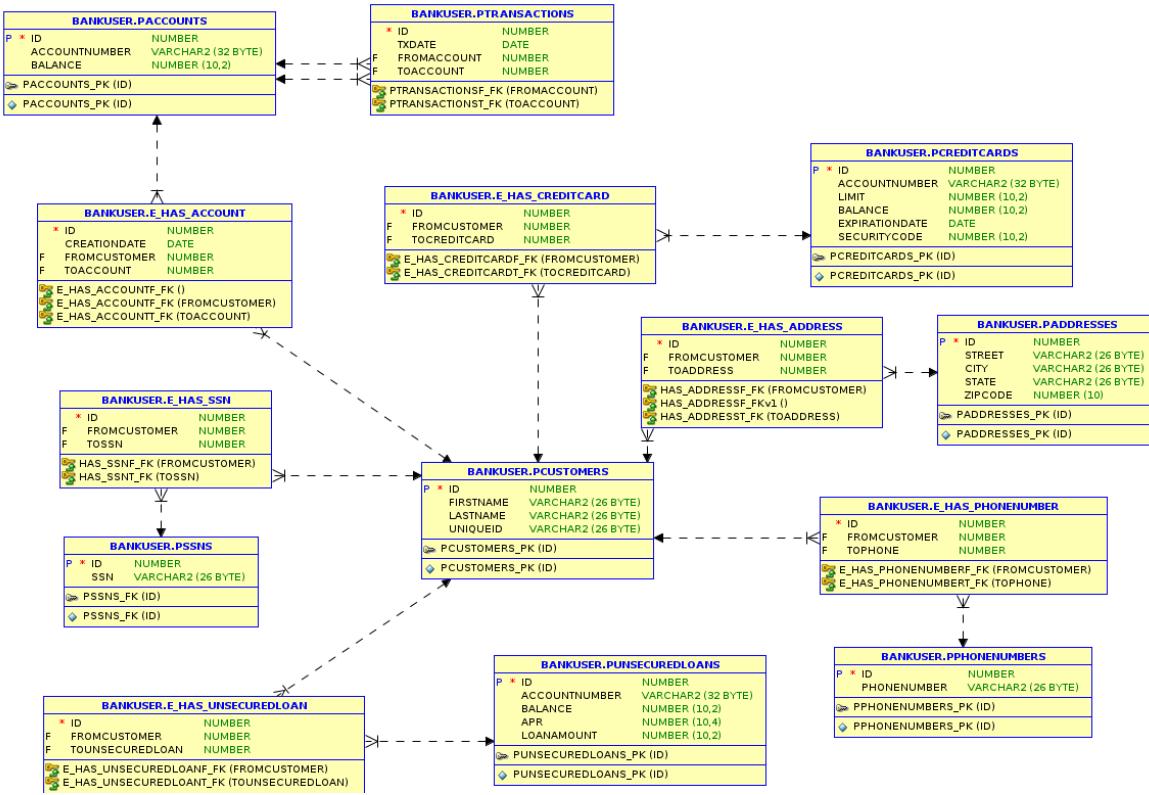
Los nombres, apellidos, números de teléfono, direcciones postales, etc han sido generados de forma artificial.

Las direcciones postales y geoposicionamientos están principalmente en la Comunidad Autónoma de Madrid.

El modelo de tablas de base de datos que se usará en el taller se detalla en el siguiente diagrama para su referencia.

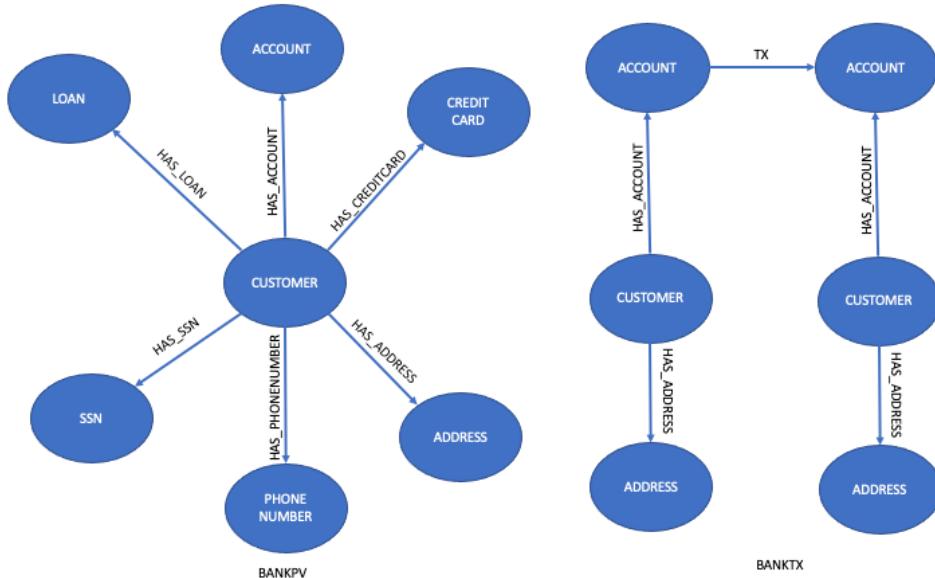
Se encuentra creado y precargado con los datos en su entorno del laboratorio.





Partiendo de este modelo relacional, se crearán dos grafos con diferentes elementos en función del tipo de análisis que se va a realizar, pero en ambos casos mapeando la información contenida en estas tablas.

La estructura resultante de estos grafos se puede ver a continuación:



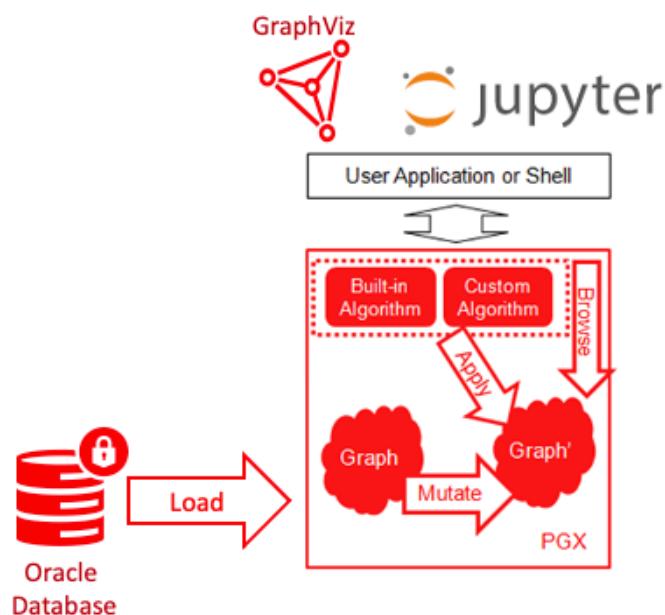
Descripción del entorno del workshop

Las actividades de este workshop se realizan con *Jupyter* como IDE cliente y con el lenguaje de programación Python.

Las distintas piezas implicadas en las actividades del workshop son:

- **In-memory PGX engine** o motor de analítica de *grafo*
- **Jupyter** como interfaz de usuario
- **GraphViz** como herramienta de visualización
- Fuentes de datos en tablas de **Oracle Database**

A continuación, se muestra un diagrama con estos elementos a modo de referencia:



Para interactuar con el motor analítico PGX, en este taller se usará el lenguaje Python.

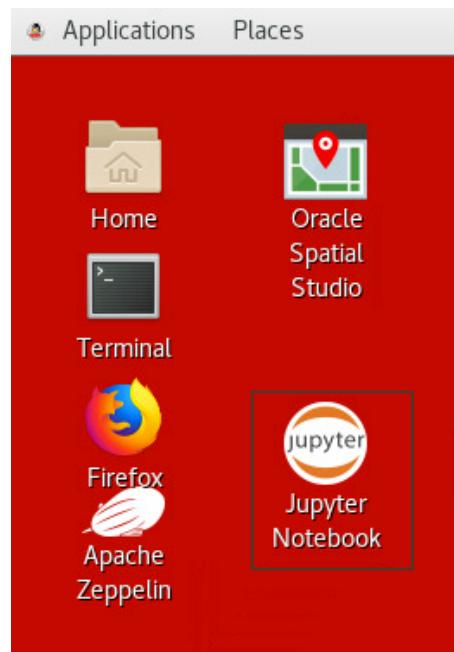
Su entorno de trabajo python dispone del paquete **pypgx** con la funcionalidad de grafos necesaria preinstalado, así como otras dependencias que se usarán para las actividades previstas en el taller.

Instrucciones del workshop

Siga las instrucciones proporcionadas para acceder al escritorio remoto. Se le proporcionará una dirección IP y un usuario/clave.

Una vez se haya accedido al servidor con el cliente de *Remote Desktop*, en el escritorio remoto encontrará un acceso directo a *Jupyter Notebook*.

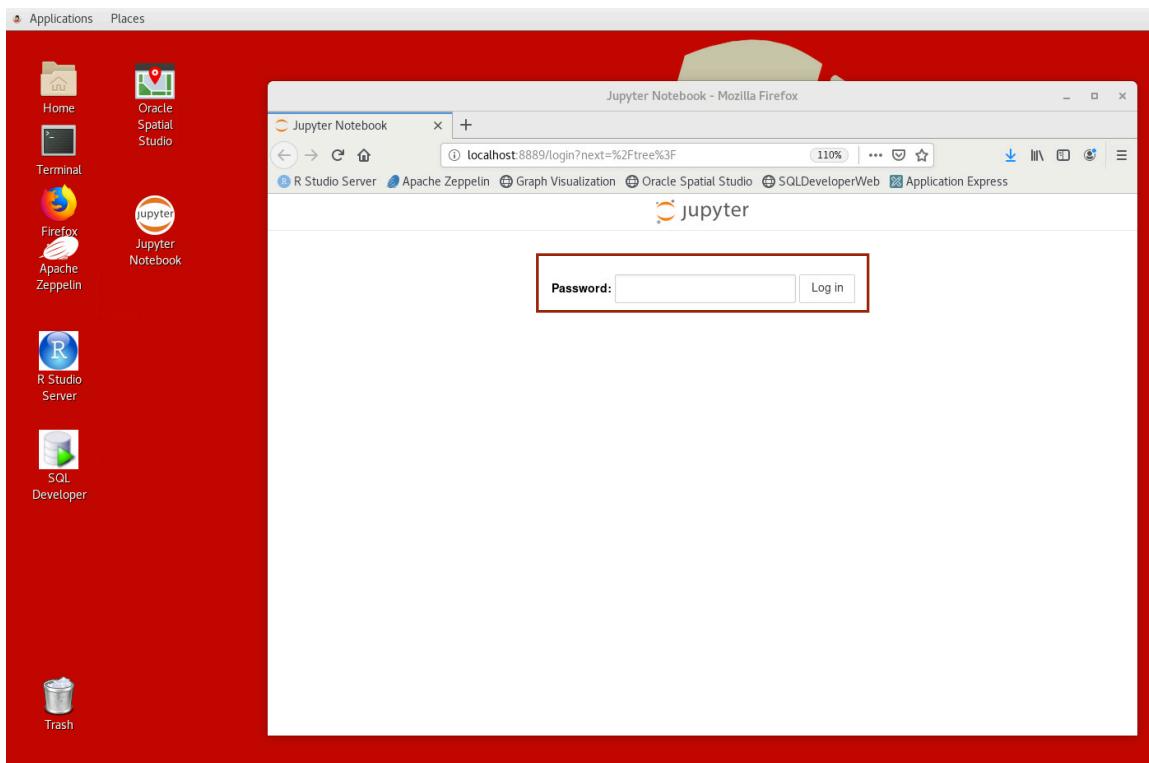
Haga doble click para abrir Jupyter en un navegador.



En la ventana del navegador aparecerá la pantalla de bienvenida a *Jupyter*, tal y como se muestra a continuación.

Introduzca la clave de acceso que le proporcione el instructor para acceder y pulse el botón *Login*.

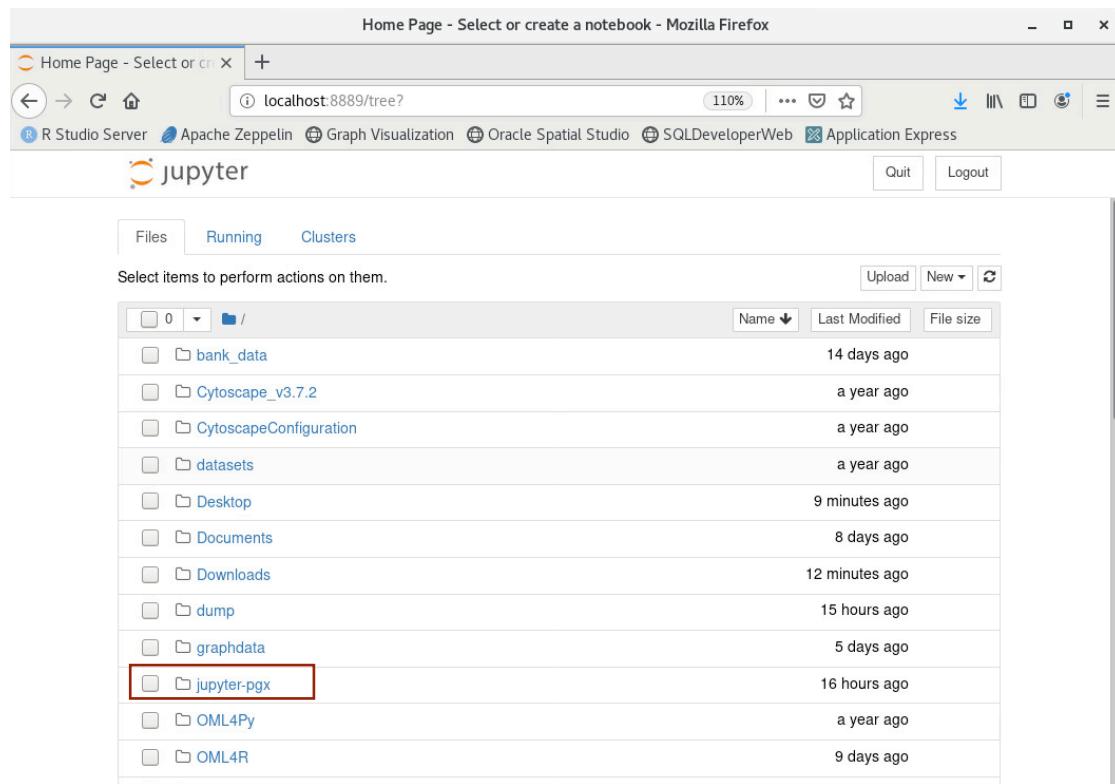




Una vez dentro de *Jupyter*, la primera pantalla muestra un navegador del sistema de ficheros en la carpeta home del usuario oracle.

Haga click en la carpeta **jupyter-pgx** donde se encuentran los notebooks necesarios para el taller.





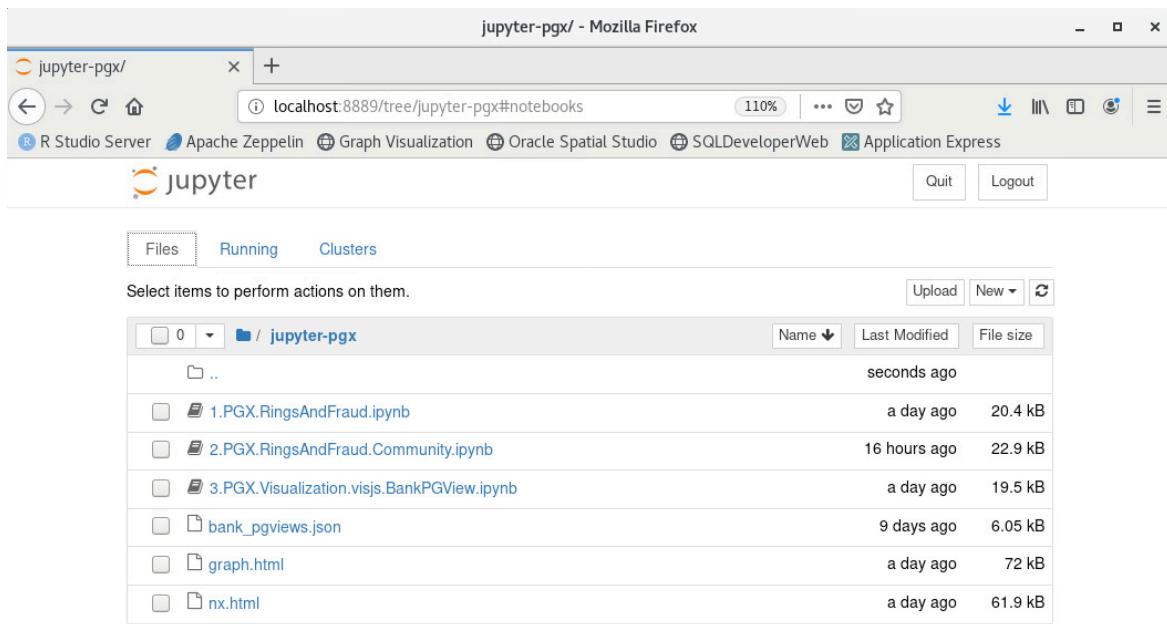
Verá un listado con tres notebooks:

- 1. PGX Rings and fraud
- 2. PGX Rings and fraud community
- 3. PGX Visualization visjs BankPGView (opcional)

El tercer notebook muestra cómo trabajar con la librería javascript de visualización Visjs, siendo opcional su ejecución.

Haciendo click sobre el nombre del notebook, *Jupyter* lo abre en otra pestaña del navegador quedando listo para su ejecución.





Se puede observar que el notebook está dividido en secciones o *párrafos* que pueden contener texto en formato *markdown* que se convierte en HTML o código Python que se va a ejecutar.

La captura siguiente, puede ver enmarcado en rojo un párrafo de *markdown* que contiene información acerca de lo que se va a hacer. Es importante leer la información de estos párrafos, que le indicarán lo que está ocurriendo en cada paso.

Enmarcado en verde, se muestra la barra de herramientas superior que permite realizar diferentes acciones sobre el párrafo que esté seleccionado en cada momento.

Esta barra de herramientas contiene el botón que permite ejecutar cada uno de los párrafos con independencia de si son *markdown* o Python.

El resto de los botones de esta barra no son necesario para la ejecución de las actividades previstas en este taller.



1.PGX.RingsAndFraud - Jupyter Notebook - Mozilla Firefox

jupyter-pgx/ 1.PGX.RingsAndFraud - Jupyter Notebook - Mozilla Firefox +

localhost:8889/notebooks/jupyter-pgx/1.PGX.RingsAndFraud 110% ... 🌐 ⚡

R Studio Server Apache Zeppelin Graph Visualization Oracle Spatial Studio SQLDeveloperWeb Application Express

jupyter 1.PGX.RingsAndFraud (autosaved) Logout Trusted Python 3

Connect to PGX

First Step: Get a session in PGX server.

PGX Server Design

- PGX is designed for a Server-Client usage model. That is, PGX is designed for the following situation: The PGX server is initiated on a server-class machine, while PGX clients (remotely) connect to the PGX server; multiple clients can connect to the same PGX server at the same time.
- Each PGX client sends out its requests to the PGX server, and the server returns responses for them.
- Internally, PGX maintains its own engine (i.e. thread pools) for running parallel graph algorithms.

Desplazándose por el notebook, verá los párrafos que contienen el código Python, que son los que debe ejecutar para realizar las acciones sobre el motor PGX, así como sobre la base de datos Oracle.

Es importante que sean ejecutados en el orden que aparecen en el notebook empezando por la parte superior, hacia la inferior. Aunque el código Python podría ser ejecutado por completo en un único paso, se ha dividido en diferentes secciones y se han intercalado párrafos de *markdown* con las explicaciones de lo que ocurre en cada uno de ellos para que resulte más cómoda su comprensión y ejecución.

Para completar el taller no es necesario escribir código nuevo o alterar el existente, aunque puede hacerlo si así lo desea o bien puede añadir párrafos nuevos donde probar su propio código.



The screenshot shows a Jupyter Notebook interface in Mozilla Firefox. The title bar reads "1.PGX.RingsAndFraud - Jupyter Notebook - Mozilla Firefox". The toolbar includes standard browser controls like back, forward, and search, along with specific Jupyter Notebook buttons for file operations and cell execution. A dropdown menu shows "Python 3" as the kernel. The main content area displays a code cell labeled "In []" containing Python code for generating an access token. The code uses the `json` and `ssl` modules to handle HTTPS requests. It defines a function `generateToken` that takes a base URL, username, and password. It constructs a JSON payload, sets headers, and sends a POST request to the `/auth/token` endpoint. It then decodes the response and extracts the 'access_token'. If an error occurs, it prints an appropriate message based on the HTTP status code. The code cell has a red box around the "Run" button in the toolbar.

```
1 import json
2 import ssl
3
4 ssl._create_default_https_context = ssl._create_unverified_context
5
6 def generateToken(base_url, username, password):
7     from urllib.request import Request, urlopen
8     from urllib.error import HTTPError
9
10    body = json.dumps({ 'username': username, 'password': password })
11    headers = { 'content-type': 'application/json' }
12    request = Request(base_url + '/auth/token', data=body, headers=headers)
13
14    try:
15        response = urlopen(request).read().decode('utf-8')
16        return json.loads(response).get('access_token')
17    except HTTPError as err:
18
19        if err.code == 400:
20            print('Authentication failed no username/password given')
21        elif err.code == 401:
22            print('Authentication failed invalid username/password')
```

Una vez se haya posicionado en un párrafo de código Python, haga click en el botón 'Run' de la barra de herramientas superior. Observará que a la izquierda de cada párrafo hay una línea IN[] que pasará a ser IN[#] con un número que indica el orden en que va ejecutando los párrafos:

The screenshot shows the same Jupyter Notebook interface after the code has been run. The title bar and toolbar are identical to the previous screenshot. The main content area now shows the output of the executed code. It includes a message about getting a token and creating a client session, followed by the generated access token. The code cell itself is still visible with its original number, indicating it was the first cell run.

```
In [1]: 1 import json
2 import ssl
3
4 ssl._create_default_https_context = ssl._create_unverified_context
```

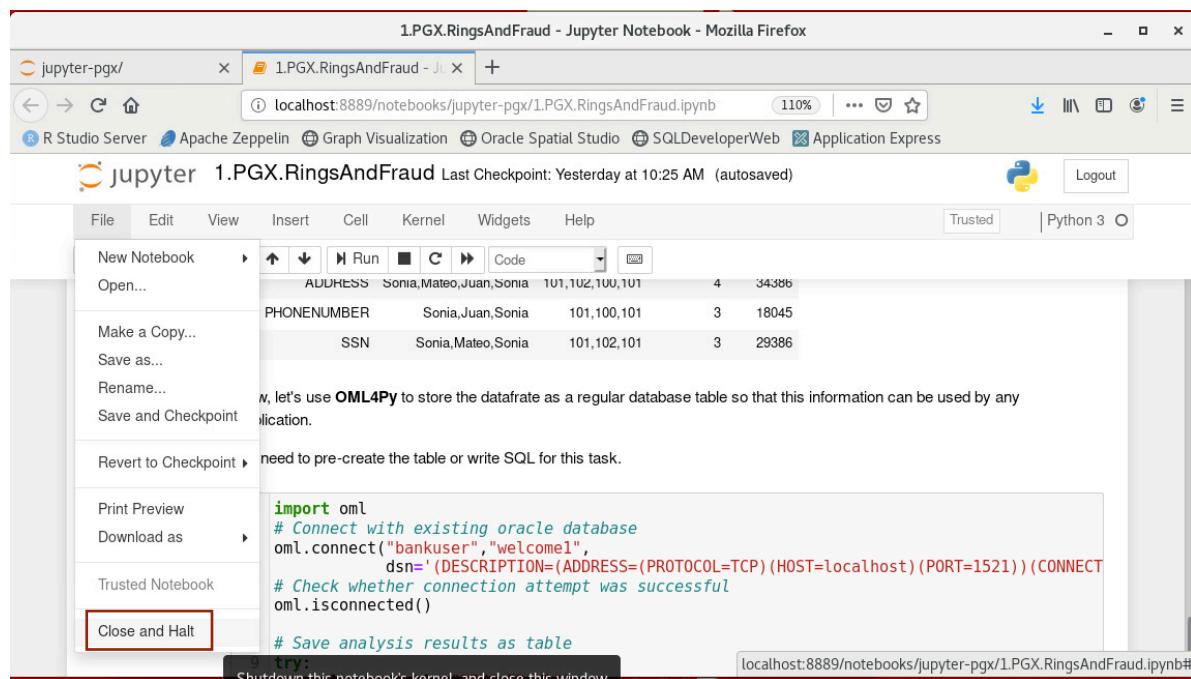


Puede avanzar por la ejecución del notebook haciendo sucesivos clicks en el botón ‘Run’. Observe la salida de cada párrafo y asegúrese de entender lo que está ejecutando en cada uno de los pasos.

En caso de producirse algún problema, aparecerá un mensaje de error justo debajo del párrafo. Consulte con su instructor para resolverlo.

El orden de ejecución de los párrafos es importante, ya que en cada párrafo se construyen objetos, variables y estructuras que son necesarios para los párrafos que van a continuación. Si los párrafos se ejecutan de manera desordenada se producirán errores.

Cuando llegue al final del notebook, en el menú superior abra **File -> Close and Halt** para cerrar el notebook y finalizar el proceso Python que está asociado a su sesión con el notebook. La pestaña del navegador se cerrará automáticamente.



The screenshot shows a Jupyter Notebook interface in a Mozilla Firefox browser window. The title bar reads "1.PGX.RingsAndFraud - Jupyter Notebook - Mozilla Firefox". The address bar shows "localhost:8889/notebooks/jupyter-pgx/1.PGX.RingsAndFraud.ipynb". The notebook content includes:

- A table with columns: ADDRESS, SONIA, MATEO, JUAN, SONIA, 101, 102, 100, 101, 4, 34386.
- A table with columns: PHONENUMBER, SONIA, JUAN, SONIA, 101, 100, 101, 3, 18045.
- A table with columns: SSN, SONIA, MATEO, SONIA, 101, 102, 101, 3, 29386.
- Text: "Now, let's use OML4Py to store the data from the database table so that this information can be used by any application." followed by "need to pre-create the table or write SQL for this task."
- Python code:

```
import oml
# Connect with existing oracle database
oml.connect("bankuser","welcome",
            dsn='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT
# Check whether connection attempt was successful
oml.isconnected()

# Save analysis results as table
```
- Message at the bottom: "Shutdown this notebook's kernel, and close this window"



Contenido de los notebooks

A continuación, se muestran los contenidos y objetivos de cada uno de los notebooks.

1.PGX.RingsAndFraud.ipynb

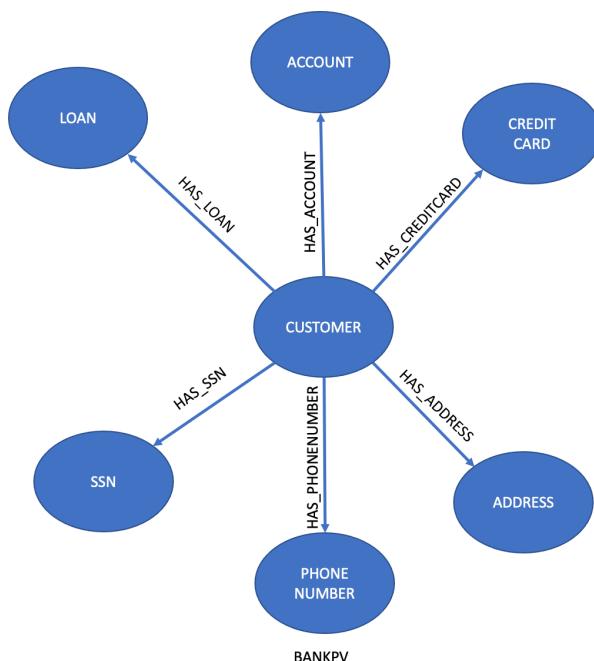
El objetivo de este primer notebook es modelar datos de clientes como un grafo para poder realizar una búsqueda de indicios de fraude. En la primera sección del notebook encontrará una explicación del tipo de escenario que se está buscando y el *modus-operandi* de este tipo de organizaciones fraudulentas.

Estos son los diferentes pasos que realizará en esta actividad:



Para lograrlo va a construir un *grafo* a partir de tablas de base de datos usando Jupyter como herramienta de trabajo y Python como lenguaje de programación.

En este notebook se construye un grafo llamado **bankpv** en PGX con la siguiente estructura:



Comprobará que el mapeo necesario para la conversión de las tablas de base de datos Oracle en este modelo de grafo se realiza mediante unos metadatos almacenados en forma de documento JSON.

```
1 # read Graph
2
3 graph = session.read_graph_with_properties("/home/oracle/graphdata/bank_pgviews.json")
4
```

Una vez construido el grafo, por medio del lenguaje PGQL se realiza una búsqueda de patrones usados habitualmente por organizaciones de fraude.

```
1 pgxResultSetNode = graph.query_pgql("""
2 select label(contact), listagg(acct.FIRSTNAME,','), count(*) as size
3 from match (acct)-[]>(contact)
4 where exists (
5   SELECT count(*) as ringsize, contact
6   from MATCH (account:CUSTOMER)-[]>(contact)
7   group by contact
8   having ringsize > 1
9   order by ringsize desc
10 )
11 group by label(contact)
12 """
13
14 for i in pgxResultSetNode:
15   print (i)

['ADDRESS', 'Juan,Sonia,Mateo', 3]
['PHONENUMBER', 'Juan,Sonia', 2]
['SSN', 'Sonia,Mateo', 2]
```

Determinados los clientes potencialmente fraudulentos, también se calcula el impacto financiero que pueden causar si consiguen completar su fraude en función del valor de los productos que tienen contratados.

```
1 pgxResultSetNode = graph.query_pgql("""
2 select label(contact), listagg(acct.FIRSTNAME,','), listagg(id(acct),','),
3       count(*) as size, round(sum(CASE
4                                     WHEN label(r)='HAS_CREDITCARDS' THEN unsecuredAccount.
5                                         WHEN label(r)='HAS_UNSECUREDLOANS' THEN unsecuredAccou
6                                         ELSE 0
7                                         END)) as FINANCIALRISK
8 from match (acct)-[]>(contact),
9         match (acct)-[r:HAS_CREDITCARDS|HAS_UNSECUREDLOANS]->(unsecuredAccount)
10 where exists (
11   SELECT count(*) as ringsize, contact
12   from MATCH (account:CUSTOMER)-[]>(contact)
13   group by contact
14   having ringsize > 1
15   order by ringsize desc
16 )
17 group by label(contact)
18 """
19
20 fin_risk = []
21
22 for i in pgxResultSetNode:
23   print (i)
24   fin_risk.append(i)
25

['ADDRESS', 'Sonia,Mateo,Juan,Sonia', '101,102,100,101', 4, 34386]
['PHONENUMBER', 'Sonia,Juan,Sonia', '101,100,101', 3, 18045]
['SSN', 'Sonia,Mateo,Sonia', '101,102,101', 3, 29386]
```



Observe que los identificadores de estos tres usuarios son: 100, 101 y 102. Estos identificadores se usarán en el siguiente notebook.

La información resultante del análisis se persiste en una tabla en la base de datos Oracle usando la funcionalidad de OML4Py.

```
1 import oml
2 # Connect with existing oracle database
3 oml.connect("bankuser","welcome1",
4             dsn='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))'
5 # Check whether connection attempt was successful
6 oml.isconnected()
7
8 # Save analysis results as table
9 try:
10     oml.drop(table="RING_FRAUD")
11 except:
12     pass
13 oml.create(df, table = 'RING_FRAUD')
14
15 # Close connection
16 oml.disconnect()
```

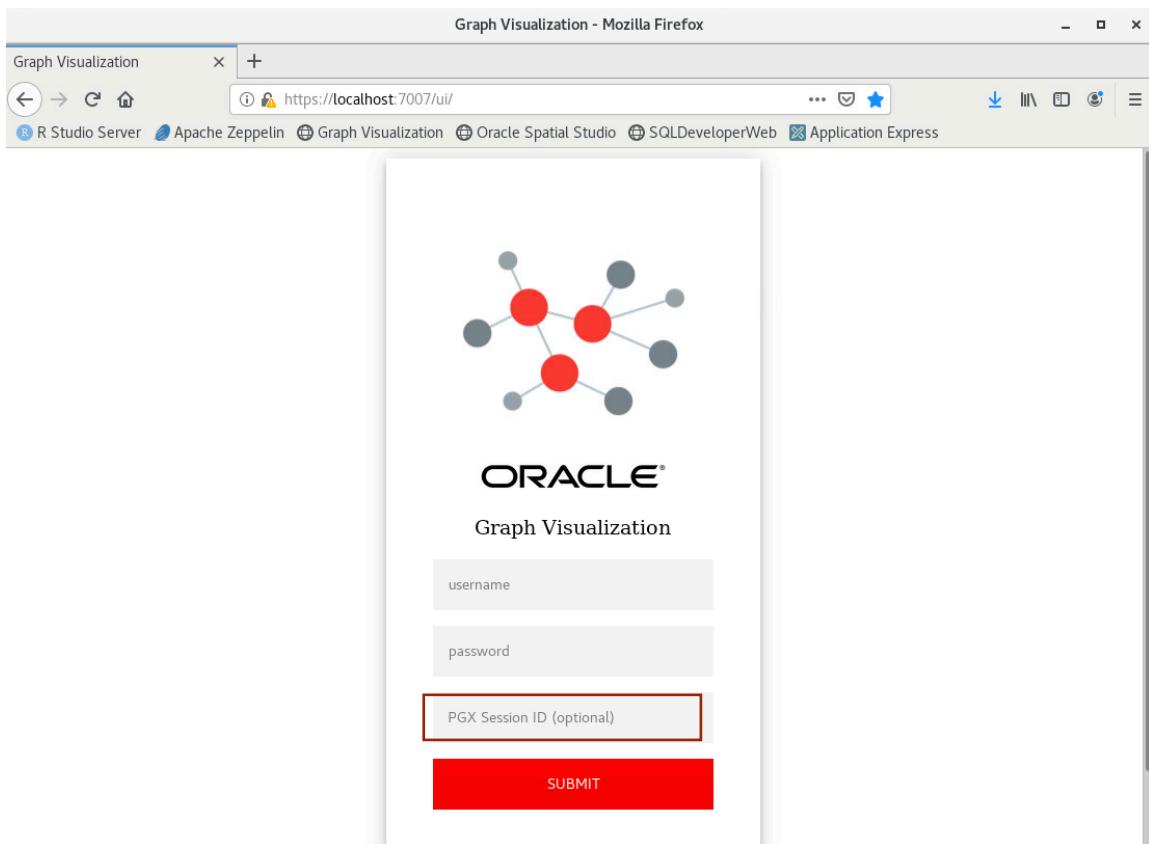
En último lugar se usa la herramienta de visualización incluida con el producto y llamada *Graph Visualization (GraphViz)*.

Cada sesión en PGX trabaja de forma aislada de las demás, de forma que los grafos y objetos no son visibles entre las sesiones, a menos que se hagan públicos de forma explícita.

Por este motivo, cuando se conecta a GraphViz, además de proporcionar las credenciales habituales de usuario y clave de acceso, se va a introducir el identificador de la sesión para tener disponible el grafo sobre el que se está trabajando y así poder visualizarlo. De no hacerlo así, se creará una sesión independiente que no tendrá visibilidad sobre el grafo que se está analizando.

La siguiente captura muestra la pantalla de acceso a GraphViz, con la casilla donde debe introducir el identificador de la sesión destacada.





En el notebook encontrará un párrafo que le proporciona este identificador de su sesión como el que puede ver a continuación (use el que tenga en su notebook, no el de la captura):

Let's now visualize the graph using the built-in tool: GraphViz.

Open <https://localhost:7007/ui> and log in with the same credentials used in the top of this notebook.

Also, the session ID is needed because graphs created in this session, are only visible to the current session. Session ID can be found by inspecting the session object as shown in the paragraph below.

It's a string that looks like this:

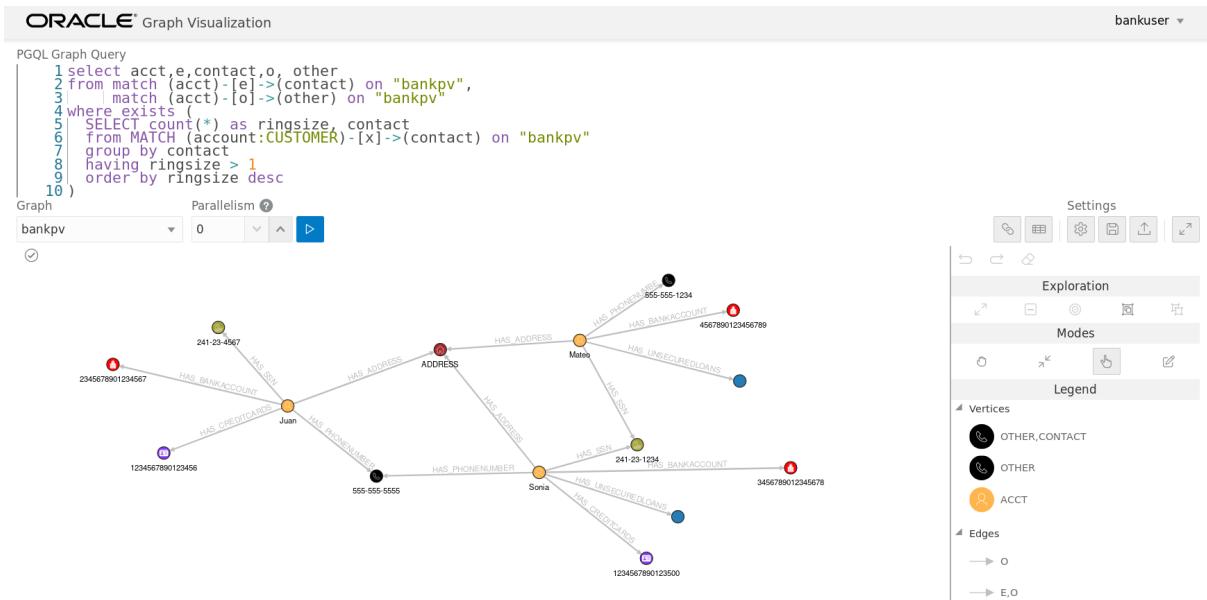
```
xxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx
```

```
1 print(session)  
PgxsSession(id: 3ce5daa5-bb0a-4937-83ff-f613c8e2c64f, name: pypgx)
```

El objetivo final es conseguir la siguiente visualización. En ella se pueden ver los miembros de la organización de fraude, con sus datos personales comunes, así como los diferentes productos que tienen contratados.

Siga las instrucciones proporcionadas en el notebook hasta conseguirlo.





Se pueden posicionar los nodos (círculos de colores) de la manera que se desee arrastrándolos con el ratón.

Se puede inspeccionar el valor de las propiedades pulsando botón derecho del ratón sobre los vértices o las aristas.

2.PGX.RingsAndFraud.Community.ipynb

En este segundo notebook se van a modelar datos de transacciones entre clientes como un grafo para poder realizar una detección de comunidades, usando algoritmos.

El objetivo es identificar si los miembros sospechosos de organización fraudulenta encontrados en el ejercicio anterior están participando en alguna comunidad que puedan estar usando para aumentar el tamaño del fraude. Es posible que otros miembros de esa comunidad estén participando en la red de fraude sin ser conscientes de ello. Esta información puede ser útil para tomar medidas preventivas que impidan que aumente el tamaño de la comunidad.

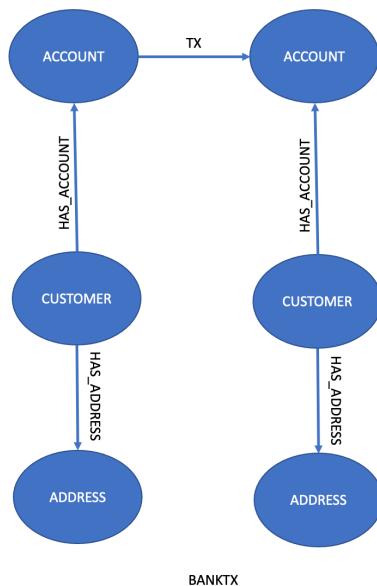


Estos son los diferentes pasos que realizará en esta actividad:



Para lograrlo va a construir un *grafo* a partir de tablas de base de datos usando Jupyter como herramienta de trabajo y Python como lenguaje de programación.

En este notebook se construye un grafo llamado **banktx** en PGX con la siguiente estructura:



De nuevo el mapeo entre las tablas de base de datos Oracle y este modelo de grafo se realiza mediante metadatos almacenados en forma de documento JSON.

En este caso el número de tablas implicadas es inferior y diferente al usado en el caso anterior. Esto es así puesto que el tipo de análisis es diferente.

```
1 # read Graph
2
3 graph = session.read_graph_with_properties("/home/oracle/graphdata/bank_tx.json")
4
```

Una vez construido el grafo, por medio del objeto **analyst**, se aplica un algoritmo de detección de comunidades denominado *label propagation*. Recuerde que este objeto se construye en los primeros párrafos, cuando se crea una sesión en PGX.

```
1 try:
2     import pypgx as pgx
3
4     session = pgx.get_session(base_url=baseUrl, token=token)
5     instance = session.server.instance
6     analyst = session.create_analyst()
7
8 except SystemExit:
9     os._exit(0)
10
```

La forma de aplicarlo sobre el grafo es tan sencilla como se puede ver a continuación.

```
1 analyst.communities_label_propagation(graph, 30)
```

El resultado es una nueva propiedad en los vértices con el identificador de la comunidad:

```
1 graph.get_vertex_properties()
[VertexProperty(name: ACCOUNTNUMBER, type: string, graph: banktx),
 VertexProperty(name: BALANCE, type: long, graph: banktx),
 VertexProperty(name: CITY, type: string, graph: banktx),
 VertexProperty(name: FIRSTNAME, type: string, graph: banktx),
 VertexProperty(name: LASTNAME, type: string, graph: banktx),
 VertexProperty(name: LATITUDE, type: string, graph: banktx),
 VertexProperty(name: LONGITUDE, type: string, graph: banktx),
 VertexProperty(name: STATE, type: string, graph: banktx),
 VertexProperty(name: STREET, type: string, graph: banktx),
 VertexProperty(name: ZIPCODE, type: long, graph: banktx),
 VertexProperty(name: label_propagation, type: long, graph: banktx)]
```

En la lista de comunidades resultante, destaca una con un tamaño superior al resto:



| COMMUNITY_ID | SIZE |
|--------------|------|
| 0 | 25 |
| 1 | 9 |
| 2 | 8 |
| 3 | 8 |
| 4 | 8 |
| 5 | 8 |
| 6 | 8 |
| 7 | 6 |
| 8 | 4 |
| 9 | 2 |

A continuación, se comprueba a qué comunidad pertenecen los clientes sospechosos de fraude (identificadores 100, 101 y 102), que resultan estar en la comunidad de mayor tamaño:

| CUSTOMER_ID | COMMUNITY_ID |
|-------------|--------------|
| 0 | 100 |
| 1 | 101 |
| 2 | 102 |

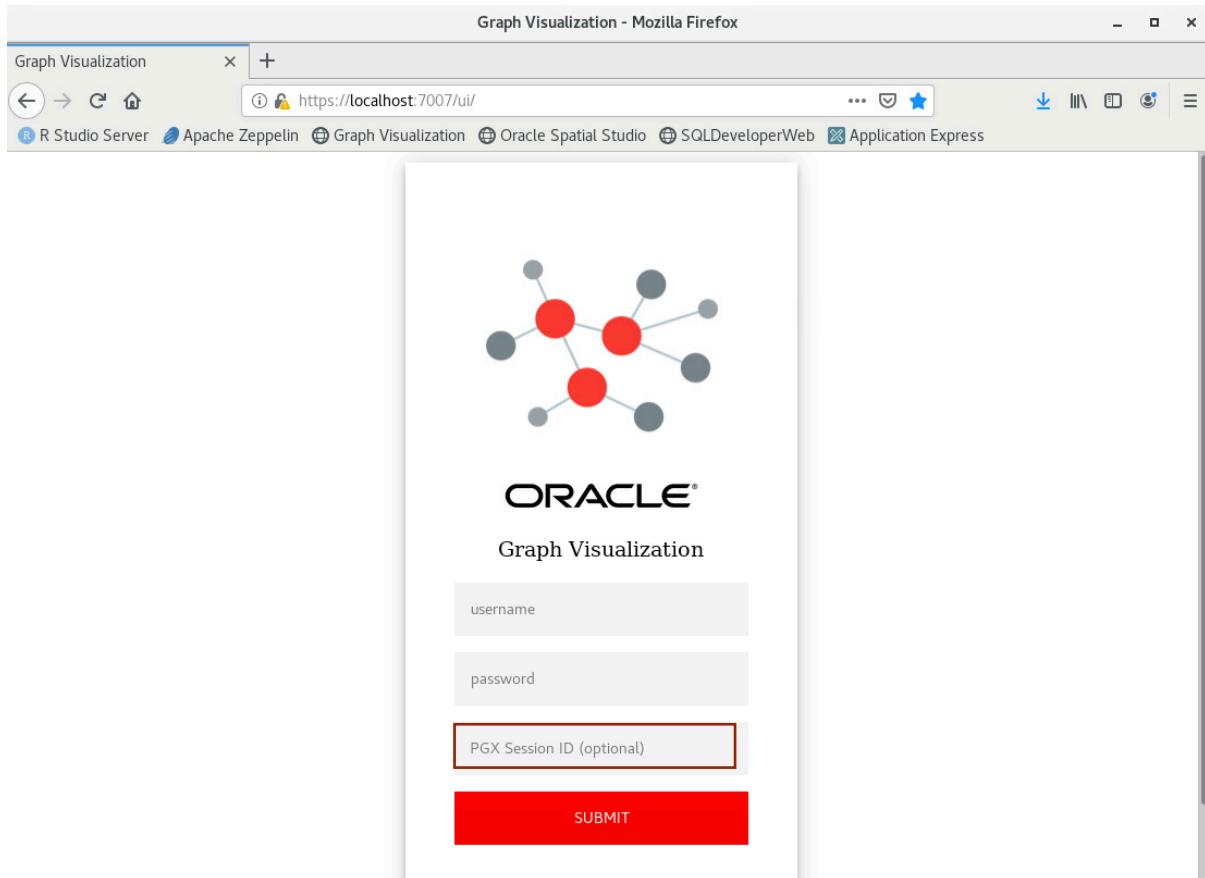
Aprovechando que la tabla PADDRESSES, que se mapea a vértices de tipo ADDRESS contiene la geolocalización de los clientes (fue generada con **Oracle Spatial Studio** de la misma manera que se hizo en el primer día del taller), se va a utilizar la funcionalidad de representación geográfica de GraphViz para representarlo en un mapa.

| Welcome Page × PADDRESSES × | | | | | | |
|--|--------------|-------------------|----------|--------------------------------------|-----------|----------|
| Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL Actions... | | | | | | |
| | COLUMN_NAME | DATA_TYPE | NULLABLE | DATA_DEFAULT | COLUMN_ID | COMMENTS |
| 1 | ID | NUMBER | No | "BANKUSER"."ISEQ\$\$_587473".nextval | 1 | (null) |
| 2 | STREET | VARCHAR2(64 BYTE) | Yes | (null) | 2 | (null) |
| 3 | CITY | VARCHAR2(26 BYTE) | Yes | (null) | 3 | (null) |
| 4 | STATE | VARCHAR2(26 BYTE) | Yes | (null) | 4 | (null) |
| 5 | ZIPCODE | NUMBER(10, 0) | Yes | (null) | 5 | (null) |
| 6 | GC_GEOGRAPHY | SDO_Geometry | Yes | (null) | 6 | (null) |
| 7 | LATITUDE | VARCHAR2(26 BYTE) | Yes | (null) | 7 | (null) |
| 8 | LONGITUDE | VARCHAR2(26 BYTE) | Yes | (null) | 8 | (null) |

El juego de datos usado para el taller contiene clientes localizados principalmente en la Comunidad Autónoma de Madrid.



De nuevo se va a usar *GraphViz* y debe usarse el identificador de esta sesión para poder usar el grafo **banktx**, si aún tiene una sesión en Graphviz del notebook anterior, debe cerrarla y abrir una con el nuevo identificador de sesión.

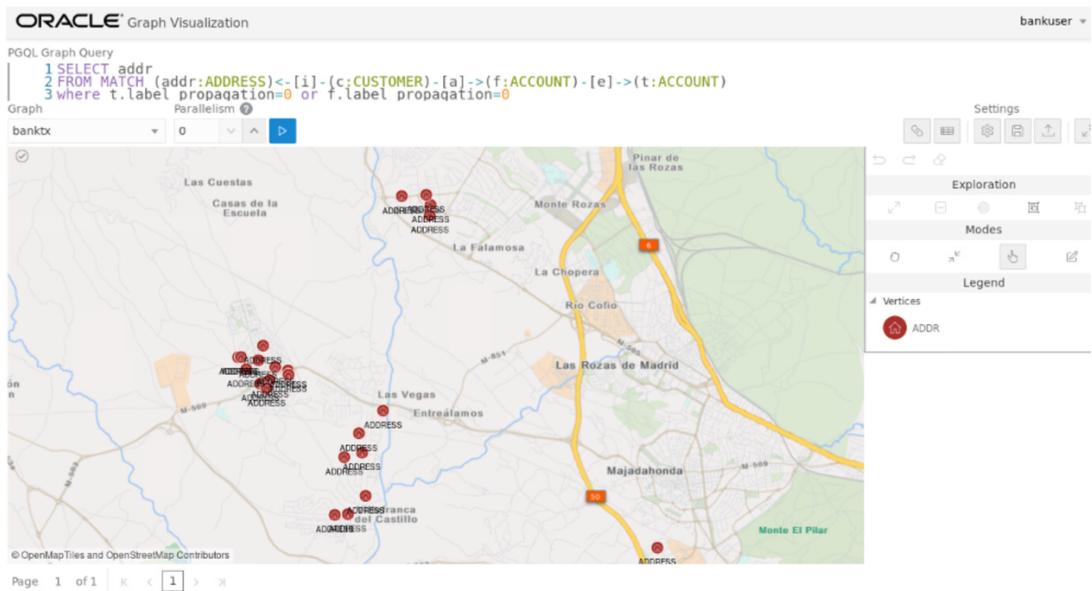


Use el nuevo identificador de su sesión.

```
1 print(session)  
PgxSession(id: 3ce5daa5-bb0a-4937-83ff-f613c8e2c64f, name: pypgx)
```

El objetivo es conseguir una salida similar a la que puede ver en esta captura. Siga las instrucciones proporcionadas en el notebook.



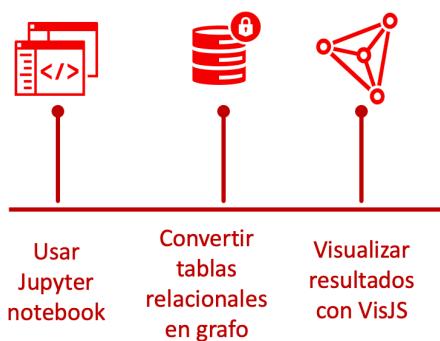


Observará que la comunidad de la organización de fraude está localizada en un área bastante concreta de la provincia de Madrid, mientras que el resto de comunidades están dispersas por toda la provincia. Esto podría ser indicativo de que esta organización está llevando a cabo algún tipo de actividad de captación en esa área.

3.PGX.Visualization.visjs.BankPGView.ipynb (opcional)

Este es un notebook opcional donde se demuestra cómo realizar visualizaciones de un grafo usando la librería Visjs.

Estos son los diferentes pasos que realizará en esta actividad:



Esta actividad es opcional. Muestra cómo integrar otros *framework* de visualización diferentes a *GraphViz*, que es el que viene incluido con Oracle Graph.



Resumen

¡Enhorabuena! Ha conseguido llegar al final de las actividades de este taller.

En este taller usted ha conseguido los siguientes retos:

- familiarizarse con Jupyter como interfaz de usuario para trabajar con el motor analítico PGX
- conocer los elementos de trabajo de PGX: session, graph, PGQL y analyst.
- transformar un modelo de datos relacional basado en tablas de Base de Datos Oracle en un modelo de grafo basado en vértices y aristas.
- realizar consultas usando lenguaje PGQL.
- realizar analítica con los algoritmos disponibles en el analyst.
- realizar visualizaciones con la herramienta GraphViz incluida.
- (opcional) realizar visualizaciones con VisJS

Más información

Descarga del producto:

<https://www.oracle.com/database/technologies/spatialandgraph/property-graph-features/graph-server-and-client/graph-server-and-client-downloads.html>

Puede consultar la documentación oficial en:

<https://docs.oracle.com/en/database/oracle/oracle-database/21/graph.html>

Puede encontrar más ejemplos y casos de uso en el siguiente repositorio de github:

<https://github.com/oracle/pgx-samples>

