ÍNDICE

[1 Introducción 2](#_Toc36557122)

[2 Requisitos previos 2](#_Toc36557123)

[2.1 Requisito 1 2](#_Toc36557124)

[2.2 Requisito 2 4](#_Toc36557125)

[3 El código 5](#_Toc36557126)

[4 Cómo debo trabajar 5](#_Toc36557127)

[4.1 Asignar los valores de las variables del módulo db\_export\_parameters.py 5](#_Toc36557128)

[4.2 Dar los nombres de la variable del fichero pgdb.ini 5](#_Toc36557129)

[4.3 Asignar los valores de las variables en main.py 6](#_Toc36557130)

[4.4 Pasos recomendados en la ejecución del programa 6](#_Toc36557131)

[4.4.1 Ejecución con create\_db\_structure=True 6](#_Toc36557132)

[4.4.2 Ejecución con write\_sql=True 7](#_Toc36557133)

[4.4.3 Ejecución para insertar los datos en la db postgres 7](#_Toc36557134)

[4.4.4 Ejecución para convertir los datos de las columnas primary\_key a minúsculas 7](#_Toc36557135)

[5 Cambios de nombres de tablas y columnas en la base de datos postgres 4](#_Toc36557136)

# Introducción

El programa, o app, facilita la migración de una db Access a postgres a un usuario técnico con conocimientos de Access y postgres mediante la ejecución de una app escrita en python.

Importante: el programa utiliza el módulo pyodbc que necesita el driver ODBC de Microsoft que debe estar instalado antes de ejecutar el programa; si tienes instalado Office en tu ordenador ya lo tienes instalado. En el momento de escribir esta documentación, este driver no permite leer los contenidos de las claves externas (foreign keys); para solventar este problema debes seguir las instrucciones que se indican en 2 Requisitos previos.

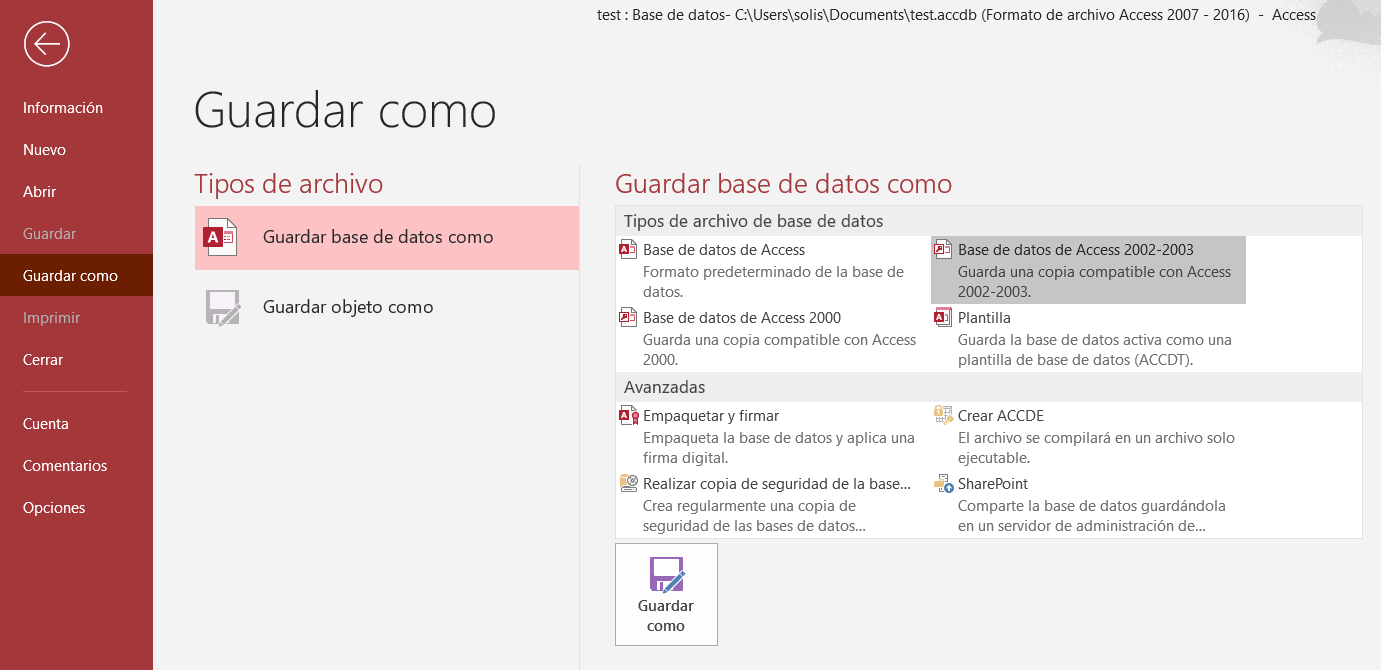
Para hacer la migración debes tener instalado un servidor Postgres y la db donde quieres migrar la db Access debe existir.

La app da por supuesto que la db Access está almacenada en una codificación que representa correctamente caracteres latinos tipo Latin-1 y que la base de datos Postgres está codificada en UTF-8.

# Requisitos previos

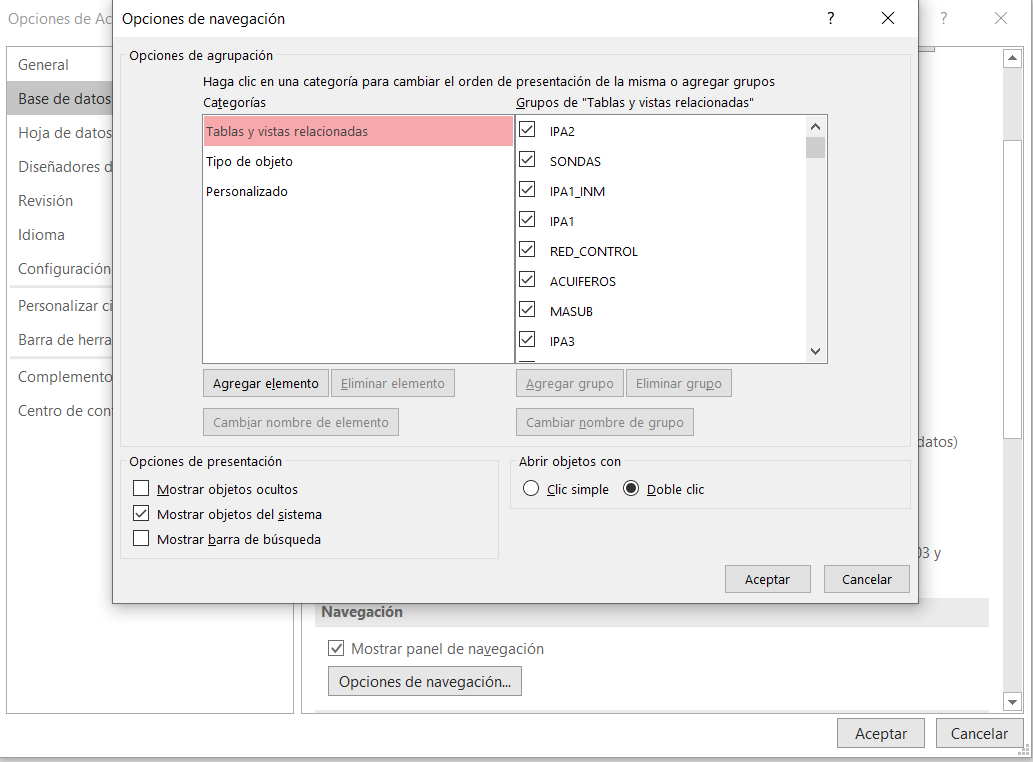
## Requisito 1

Para leer la estructura de la db Accesss y sus datos se utiliza el módulo pyodbc. Algunas funcionalidades del módulo no son utilizables debido a que no están soportados por el driver de Access que es suministrado por microsoft. Esta limitación puede resolverse leyendo una tabla del sistema de la db Access, que por defecto está oculta y protegida contra lectura (este es el único permiso necesario). Para poder cumplir estos pasos se necesita que la db sea Access 2003, pues las versiones siguientes de la db no permiten acceder a un menú de permisos y usuarios; Por tanto, si tienes una db de extensión accdb, conviértela a mdb (Archivo, Guardar como)

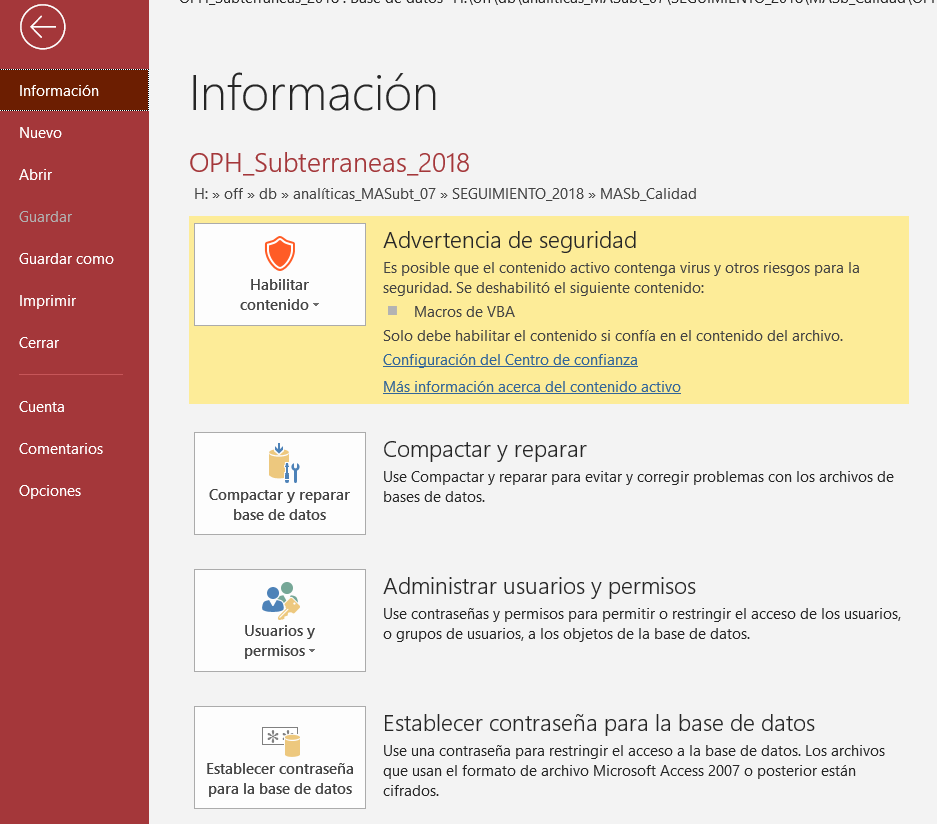


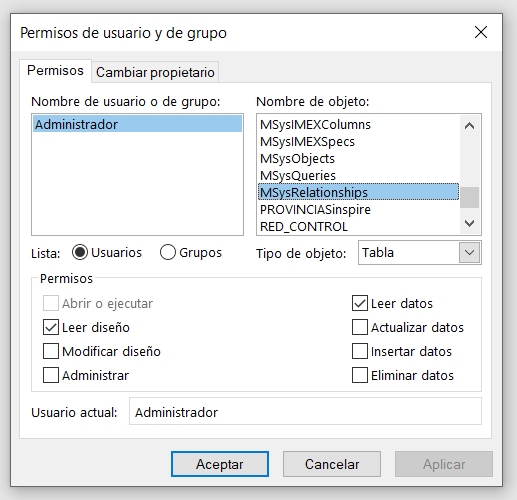
Los pasos necesarios para leer las foreign keys de la db Access mdb son:

1. Hacer visibles las tablas del sistema: Archivo -> Opciones -> Base de datos actual -> Navegación, Opciones de navegación, Opciones de presentación, y marcar Mostrar objetos del sistema



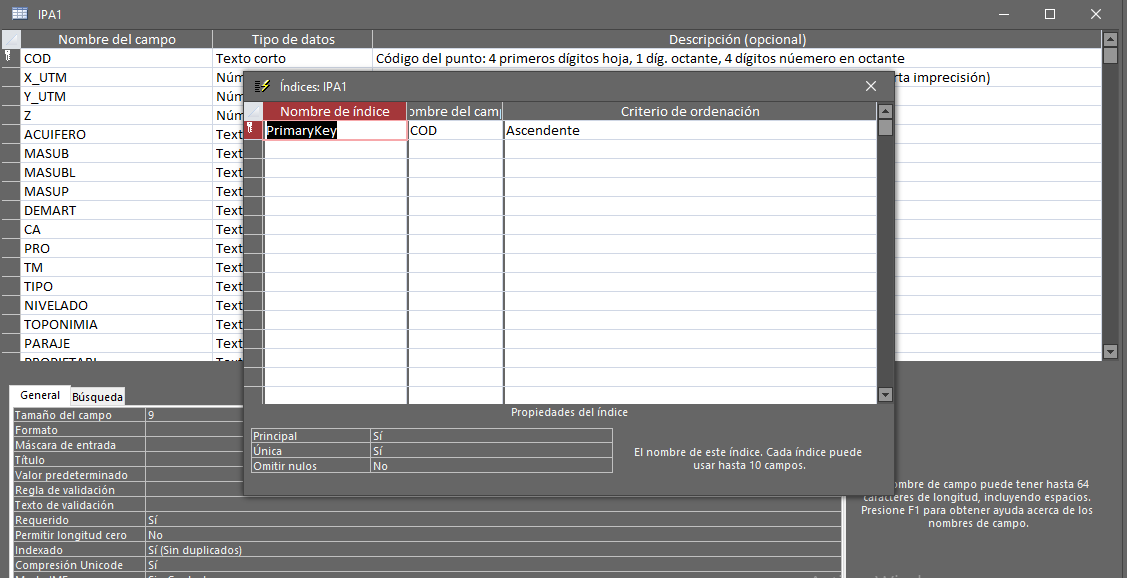
1. Permitir la lectura de la tabla MSysRelationShips: Archivo -> Administrar usuarios y permisos -> Permisos y usuarios de grupo, como Administrador, tipo de objeto tabla, nombre del objeto MysRelationShips, marcar leer diseño y leer datos.





## Requisito 2

En la db Access la clave primaria debe estar siempre referida como PrimaryKey; se ha comprobado que a veces toma otro nombre; en estos casos debe cambiarse el nombre a PrimaryKey



¿Cómo puede detectarse estos casos?

Además de inspeccionar las tablas de Access antes de iniciar la migración, cuando ya se ha creado la db sqlite con la estructura de la db Access, haz una select a la tabla tables. Observa las tablas que tienen vacía la columna primarykeys; si en realidad la tabla en Access sí tiene primary key, entonces se trata de un problema de nombre y opera como se ha indicado anteriormente.

# Cambios de nombres de tablas y columnas en la base de datos postgres

Postgres es más estricto que Access a la hora de dar nombres a las tablas y las columnas. El programa realiza un cambio automático de nombres en la db postgres. Se cambian:

* Espacios en blanco por ’\_’
* ‘-‘ por ’\_’
* Un dígito al principio por ‘d’ + nombre (con el dígito)
* Se convierten los caracteres a ascii; de esta manera desaparecen los acentos y las ñ
* Se pasa todo a minúscula

No se han encontrado nombres con ‘(‘ o ‘)’, si los hubiere seguramente habría que ampliar los caracteres de reemplazo y pasarlos a ‘\_’

# El código

La app está formada por 4 módulos:

* main.py, controla las opciones de la ejecución, instancia un objeto Migrate y realiza una ejecución con las opciones indicadas
* db\_export\_parameters: se definen los parámetros de las ejecuciones; una vez establecidos, sus valores no cambian en las sucesivas ejecuciones de la ap para un proceso concreto de migración
* db\_export: donde se define la clase Migrate y sus propiedades
* littleLogging.py: que controla los mensajes que se almacenan en el fichero app.log cuando termina la ejecución del programa

Requisitos de funcionamiento. Los siguientes módulos son requeridos y deben ser instalados en Python para que funciones la app:

* psycopg2, https://pypi.org/project/psycopg2/
* pyodbc, <https://github.com/mkleehammer/pyodbc/wiki>

# Cómo debo trabajar

Una vez completados los pasos anteriores, crea una carpeta para albergar los módulos de la app y copia los módulos de programa. Tienes que tener un editor de texto o una IDE para controlar algunos módulos de la app y controlar la ejecución del programa

## Asignar los valores de las variables del módulo db\_export\_parameters.py

Primero hay que abrir el módulo db\_export\_parameters.py para dar valor a las variables; el significado de las variables se explica en el módulo.

## Dar los nombres de la variable del fichero pgdb.ini

También debes dar los valores correctos al fichero pgdb.ini, que almacena los datos de la conexión a la db postgres donde se quiere migrar la db Access; la db postgres debe existir. El contenido de este fichero es -los valores de las variables son ejemplos-:

[h2ogeo]

host=localhost

database=h2ogeo

user=postgres

password=pg

siendo [h2ogeo] el nombre de la sección, que el calor que hay que en completar db\_export\_parameters.py

## Asignar los valores de las variables en main.py

Lo que se va a hacer en cada ejecución de la app se controla mediante una serie de variables definidas en main.py y que sólo pueden tener 2 valores, True o False. Las variables son:

* create\_db\_structure, si True se crea el fichero sqlite con la estructura de la db Access; si True la crea por primera vez o sobreescribe la existente.
* write\_sql, si True escribe la estructura de las tablas en 2 ficheros sql que deben ser ejecutados desde el programa cliente de postgres psql.
* write\_data\_to\_csv, si True graba los datos de cada tabla en un fichero csv.
* py\_upsert, si True ejecuta un upsert de los datos en la db access en la db postgres
* Para convertir los contenidos de las columnas implicadas en las claves primarias o ajenas en minúsculas hay 2 opciones:
* keys2lower\_py si True la conversón se hace directamente con python
* keys2lower\_sql si True se escribe un fichero sql.

## Pasos recomendados en la ejecución del programa

Te recomiendo que vayas paso a paso.

* La primera vez debes crear las tablas en postgres y ejecutar el fichero psql resultante. Recuerda que debes haber creado previamente Postgres.
* A continuación, debes insertar los datos
* Puedes pasar a minúscula los contenidos de las columnas implicados en claves primarias o foráneas
* Para finalizar crea las relaciones foráneas ejecutando un fichero sql

### Ejecución con create\_db\_structure=True

En todos los casos debes ejecutar primero create\_db\_structure; recomiendo que inspecciones las tablas creadas en sqlite porque todo lo que se haga después depende de su contenido.

El fichero sqlite tiene 3 tablas:

table tables

name TEXT, -- nombre de la tabla. Tiene las siguientes columnas:

* table\_type TEXT, -- tipo de tabla
* primary\_key TEXT, -- columna/s que actúa como clave primaria de la tabla

table columns – columnas de cada tabla. Tiene las siguientes columnas:

* table\_name TEXT, -- nombre de la tabla
* col\_name TEXT, -- nombre de la columna
* type\_i INTEGER, -- tipo de la columna (entero)
* type\_name TEXT, -- tipo de la columna (texto)
* column\_size INTEGER, -- tamaño d ela columna
* pg\_type\_name TEXT, -- tipo equivalente en postgres
* col\_number INTEGER, -- número de orden de la columna en la tabla Access

table relationships – foreign keys. Tiene las siguientes columnas:

* references\_table TEXT, table que referencia
* references\_cols TEXT, columna de la tabla que referencia
* referenced\_table TEXT, table referenciada
* referenced\_cols TEXT, columna de la tabla referenciada

En especial comprueba que:

* Todas las tablas con clave primaria tienen indicadas sus columnas (tabla tables)
* Todas las columnas de Access tienen un tipo equivalente correcto en Postgres (tabla columns) Insertar los datos con python

### Ejecución con write\_sql=True

Se crean 2 ficheros de extensión sql

* El primero tiene la estructura de las tablas y sus claves primarias, este fichero debe ser ejecutado con el programa cliente del servidor postgres psl antes de insertar los datos. El nombre de este fichero es *nombre\_db\_access*\_migrate01.sql
* También se crea otro fichero donde se crean las claves ajenas y opcionalmente se mueven las tablas a un esquema en la db, el esquema puede no existir, en cuyo caso lo crea; este fichero se recomienda que se ejecuta **DESPUÉS** de insertar los datos. El nombre de este fichero es *nombre\_db\_access*\_migrate04.sql

Antes de continuar es **IMPRESCINDIBLE** que ejecutes el fichero *nombre\_db\_access*\_migrate01.sql. Desde la línea de comandos ejecuta una línea similar a esta adaptada a tus nombres y direcciones:

psql -U postgres -f path2\*nombre\_db\_access*\_migrate01.sql -a postgresdb\_name > info.txt

### Ejecución para insertar los datos en la db postgres

Para insertar los datos tienes que haber ejecutado previamente el fichero *nombre\_db\_access*\_migrate01.sql como se ha comentado en el apartado anterior

A continuación, puedes insertar los datos de dos maneras: desde ficheros csv o directamente desde Access

Una vez más Postgres es más estricto que Access y se deben eliminar los espacios en blanco alrededor de los textos; si no se hace así las claves foráneas dan errores al comparar códigos sin y con espacios en blanco.

**Mediante ficheros csv**

Los ficheros csv se crean con la variable **write\_data\_to\_csv=True**. Una vez creados los ficheros se cargan en las tablas de Access mediante instrucciones psql. Desde una venta psql y una vez te encuentres en la db, por ej. test:

C:> psql -U user -d test

test=# \copy tabla1 from 'path1\datos\_tabla\_1.csv' with CSV header delimiter ',' encoding 'UTF-8'

**Directamente desde Access**

En este caso con la variable **py\_upsert=True**. Esta opción inserta o actualiza l información (se puede ejecutar con diferentes datos). Yo la encuentro más práctica que con ficheros csv.

### Ejecución para convertir los datos de las columnas primary\_key a minúsculas

Se puede hacer de 2 maneras.

* Se hace una ejecución con **keys2lower\_sql=True**; a continuación, se tiene que ejecutar el fichero sql con psql. Su nombre es *nombre\_db\_access*\_migrate03.sql
* Se ejecuta el programa con la variable **keys2lower\_py=True**, convierte directamente los datos de las columnas en claves primarias o ajenas en minúsculas