

Acceso a BD con Python (Ejercicios con PostgreSQL)

Miguel R. Penabad

2021

1. Asegúrate de tener el entorno disponible:
 - Python 3.X.
 - Módulo `psycopg2`.
 - PostgreSQL: una versión reciente (13 o superior).
2. Crea un usuario en PostgreSQL identificado por una clave y con privilegio `createdb`. Crea una base de datos con ese usuario.

Asegúrate de que puedes conectarte con `psql`, bien usando una clave, o con autenticación `peer` de postgres.

Nota: Si creas un usuario con el nombre de tu usuario en tu equipo, y una base de datos con el mismo nombre, podrás conectarte con todas las opciones predeterminadas.
3. Conéctate con python a la base de datos. Comprueba los parámetros de la conexión viendo el valor del campo `dsn` o el método `get_dsn_parameters` de la conexión.
4. Descarga el fichero `exerdb.py`. Examina su contenido y ejecútalo.
5. Completa la función `connect_db()` para que se realice la conexión con la base de datos. Usa las credenciales y base de datos creadas anteriormente. Desactiva el modo `autocommit`.

La función debe devolver la conexión obtenida.
6. Completa la función `disconnect_db()` para desconectarte de forma adecuada de la base de datos.
7. Completa la función `create_table()` para crear la tabla `artigo` con 3 campos: `codart`, tipo `int`, clave primaria; `nomart`, tipo `varchar(30)` que no admite nulos, y `prezoart`, tipo `numeric(5,2)` que admite nulos y tiene una restricción que requiere que los precios deben ser positivos.

No debes hacer ningún control transaccional ni de errores.
8. Ejecuta el programa y crea la tabla `artigo`. Si sales del programa, abre una sesión con `psql` y mira si la tabla está creada (ejecuta `\d` en `psql` para listar las tablas del usuario).

Vuelve a seleccionar la opción de crear la tabla, y comprueba que pasa.
9. Vuelve a ejecutar el programa, selecciona la opción de crear la tabla, y luego sale del programa pulsando `q`. ¿Qué ocurre ahora?
10. Revisa la función `create_table()`. Incluye el control transaccional y el control de errores adecuado.
11. Crea una nueva opción de menú para eliminar (`drop`) la tabla, e implementa una función. A partir de ahora debes hacer siempre un control correcto de las transacciones y de las excepciones.
12. Crea una nueva función que pida por teclado un código, nombre y precio (debe ser posible omitir el precio) e inserte una fila en la tabla `artigo`.

Haz un control de errores de modo que la persona que use el programa no tenga que comprender conceptos como claves primarias o restricciones del modelo relacional.

Puedes usar la función `input(<prompt>)` para recopilar los datos como strings, y las funciones de conversión `int(<string>)` y `float(<string>)`.

No es necesario hacer un control exhaustivo de la entrada de datos.

13. Crea unha opción para borrar un artigo a partir do código. Indica se se borrou ou se o código de artigo non existía.
Non esquezas o control transaccional e de erros, pero recorda que se unha sentencia `delete` non borra ningunha fila porque non as atopa (por exemplo, ningunha satisfai o filtro do `where`), iso non é un erro.
14. Crea unha opción para borrar todos os artigos cuxo nome inclúa un determinado texto, que pedirás por teclado. Indica cantos artigos se borraron.
15. Crea unha opción que indique cantos artigos hai almacenados na base de datos
16. Crea unha opción que mostre o detalle dun artigo (todos os seus datos) a partir do seu código, que debes pedir por teclado. Ten en conta que o código introducido pode non existir.
17. Crea unha opción que mostre un listado completo de todos os artigos. Usa un bucle e a función `fetchone()` do cursor.
18. Crea unha opción que mostre un listado completo de todos os artigos que teñan un prezo maior que un prezo dado (que pedirás por teclado). Mostra ó final do listado cantos artigos hai nese listado (non uses un contador no bucle que procesa as filas!). Neste caso, usa a función `fetchall()` do cursor.
19. Modifica a función que mostra o detalle dun artigo para que:
 - Admita un parámetro adicional `control_tx` (opcional, predeterminado `True`) para especificar se queremos facer ou non control transaccional (se é `False` non fará `commit/rollback`)
 - Devolva o código do artigo, se este existe, e `None` noutro caso (ou ben non existe, ou se produciu un erro)
20. Crea unha opción para modificar artigo. Pide por teclado o código de artigo, mostra os datos actuais, solicita un novo nome e prezo, e realiza a modificación. Podes utilizar a nova versión da función que pide o código e mostra o detalle do artigo.
21. Crea unha opción para incrementar o prezo dun artigo. Pide por teclado o código de artigo, mostra a información sobre el, pide a porcentaxe de incremento de prezo e realiza a modificación.
O incremento debe facerse directamente na sentencia SQL. Non calcules o novo prezo no código Python.
22. Modifica a funcionalidade anterior, de xeito que poidas parar a execución do programa despois da sentencia `update` pero antes de confirmar a transacción (por exemplo, pedindo un valor por teclado).
Abre 2 sesións nas que executas o programa e executa a opción de incrementar o prezo do mesmo artigo ó mesmo tempo. Que ocorre?
23. Cambia o modo de aillamento para as transaccións nesta funcionalidade, poñéndoo `SERIALIZABLE`. Modifica o código para garantir un bo control de erros e unha boa usabilidade do programa.