Acceso a BD con Python (Exercicios con PostgreSQL)

Miguel R. Penabad

2021

1. Asegúrate de ter o entorno dispoñible:
   - Python 3.X.
   - Módulo `psycopg2`.
   - PostgreSQL: unha versión recente. As transparencias deste tema usan a versión 13.
2. Crea un usuario en PostgreSQL identificado por unha clave e co privilexio `createdb`. Crea unha base de datos con ese usuario.

   Asegúrate de que podes conectarte con `psql`, ben usando unha clave, ou coa autenticación `peer` de postgres.

   Nota: Se creas un usuario co nome do teu usuario no teu equipo, e unha base de datos co mesmo nome, poderás conectarte con todas as opcións predeterminadas.
3. Conéctate con python á base de datos. Comproba os parámetros da conexión vendo o valor do campo `dsn` ou o método `get_dsn_parameters` da conexión.
4. Descarga o ficheiro `exerbda.py`. Examina o seu contido e execútao.
5. Completa a función `connect_db()` para que se realice a conexión coa base de datos. Usa as credenciais e base de datos creadas anteriormente. Desactiva o modo `autocommit`.

   A función debe devolver a conexión obtida.
6. Completa a función `disconnect_db()` para desconectarte de forma axeitada da base de datos.
7. Completa a función `create_table()` para crear a táboa `artigo` con 3 campos: `codart`, tipo `int`, clave primaria; `nomart`, tipo `varchar(30)` que non admite nulos, e `prezoart`, tipo `numeric(5,2)` que admite nulos e ten unha restricción que require que os prezos deben ser positivos.

   Non fagas ningún control transaccional nin de erros.
8. Executa o programa e crea a táboa `artigo`. Sen saír do programa, abre unha sesión con `psql` e mira se a táboa está creada (executa `\d` en `psql` para listar as táboas do usuario).

   Volve a seleccionar a opción de crear a táboa, e comproba que pasa.
9. Volve a executar o programa, selecciona a opción de crear a táboa, e logo sae so programa pulsando `q`. Que ocorre agora?
10. Revisa a función `create_table()`. Inclúe o control transaccional e control de erros axeitado.
11. Crea unha nova opción de menú para eliminar (`drop`) a táboa, e impleméntaa nunha función. A partir de agora fai sempre un control correcto das transaccións e das excepcións.
12. Crea unha nova función que pida por teclado un código, nome e prezo (debe ser posible omitir o prezo) e inserte unha fila na táboa `artigo`.

    Fai un control de erros de xeito que a persoa que use o programa non teña que comprender conceptos como claves primarias ou restricións do modelo relacional.

    Podes usar a función `input(<prompt>)` para recoller os datos como strings, e as funcións de conversión `int(<string>)` e `float(<string>)`.

    Non é necesario facer un control exhaustivo da 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 introducio 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 `False` ou 0 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.