Tema: V.
Accés a BBDD des d'aplicacions de escriptori: model de capes.

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant

Contingut

Preliminars

- En aquest tema anem a veure com fer ús de BBDD relacionals des d'aplicacions d'escriptori.
- Farem una breu introduccion al model de capes que serà explicat en profunditat en la parteix web.
- Posarem en la pràctica l'anterior fent ús de sqlite3 . Para això veurem primer les característiques de *sqlite3* i posteriorment exemples d'ús des d'aplicacions escrites en Vala.

Presentació de Sqlite3 (I)

- Sqlite3 -o símplemente *Sqlite*- consisteix en una biblioteca programari que implementa un motor de BBDD relacional (SQL).
- Tal com indica al seu pàgina web , de manera molt resumida, les seves característiques principals són:
  - És autocontenida (*self-contained*).
  - No té un procés servidor (*serverless*).
  - No necessita cap configuració especial para començar a funcionar (*zero-configuration*).
  - És transaccional (*transactional*).

Presentació de Sqlite3 (II)

Algunes altres característiques no menys importants de sqlite són:

- Implementa gran part de `SQL92`.
- Una bb.dd. completa s'emmagatzema en un sol fitxer multiplataforma.
- Suporta bb.dd. de terabytes i cadenes/blobs de gigaoctets.
- Reduït grandària en memòria, per exemple, completament configurada pot ocupar uns 400KiB.
- Molt ràpida.
- `API` molt senzill, podent ser emprada des de diversos llenguatges de programació.
- Escrita en `ANSI-C` en un sol fitxer '`.c`' i el corresponent '`.h`'.
- Ve amb una aplicació en manera text -`CLI`- que fa les vegades d'administrador de bb.dd.: `sqlite3`.

Presentació de Sqlite3 (III)

Algunes aplicacions que usen sqlite:

- Adobe Photoshop Elements utilitza SQLite com a motor de base de dades en la seva última versió del producte (la 6.0) en substitució del Microsoft Access.
- Mozilla Firefox usa SQLite per emmagatzemar, entre uns altres, les cookies, els favorits, l'historial i les adreces de xarxa vàlides.
- Diverses aplicacions d'Apple utilitzen SQLite, incloent Apple Mail i el gestor de RSS que es distribueix amb Mac US X. El programari Aperture d'Apple guarda la informació de les imatges en una base de dades SQLite, utilitzant la API Core Data.
- El navegador web Opera usa SQLite per a la gestió de bases de dades WebSQL.
- Skype.

Sqlite i llenguatges de programació

Sqlite pot ser utilitzat des de diversos llenguatges de programació, alguns d'ells són:

- C, C++, Vala, Java
- Pascal, Delphi
- Python, Perl
- PHP
- Des de .NET es pot accedir usant el projecte de codi obert System.Data.SQLite

Presentació de Sqlite3 (IV)

Per completar aquesta introducció és interessant que consultis els següents enllaços:

- Projectes, aplicacions i empreses que usen sqlite .
- La sintáxis de SQL suportada .
- Documentació en general.
- Llibres sobre sqlite.

`Recomanable:` Com a programador d'aplicacions en general, també et pot ser molt útil consultar la manera en la qual es testea Sqlite: How SQLite Is Tested .

sqlite3 - L'intèrpret d'ordres de Sqlite. (I)

- Sqlite inclou un intèrpret d'ordres anomenat `sqlite3`.
- Permet introduir ordres SQL i executar-les directament contra una bb.dd. Sqlite.
- Per engegar-ho obrim un terminal en manera text i teclegem l'ordre: `sqlite3`.

Exemple d'ús de sqlite3

http://www.sqlite.org/sqlite.html

```
1  $ sqlite3
   SQLite version 3.7.15.2 2013-01-09 11:53:05
3  Enter ".help" for instructions
   Enter SQL statements terminated with a ";"
```

Per exemple, per crear una bb.dd. cridada '`test.db`' i que tingui una taula anomenada `tbl1` podríem fer el següent:

```
   $ sqlite3 test.db
2  SQLite version 3.6.11
   Enter ".help" for instructions
4  Enter SQL statements terminated with a ";"
   sqlite> create table tbl1(one varchar(10), two smallint);
6  sqlite> insert into tbl1 values('hello!',10);
   sqlite> insert into tbl1 values('goodbye', 20);
8  sqlite> select * from tbl1;
   hello!|10
10 goodbye|20
   sqlite>
```

Per sortir de l'intèrpret `sqlite3` usem el caràcter de finalització de fitxer: `Control-D` o l'ordre '`.exit`'.

Metadatos en sqlite

http://www.sqlite.org/sqlite.html

Les metadades o l'esquema d'una bb.dd. en sqlite s'emmagatzemen en una taula especial anomenada: `sqlite_master`. Aquesta taula es pot usar com qualsevol altra taula:

```
1  $ sqlite3 test.db
   SQLite version 3.6.11
3  Enter ".help" for instructions
   sqlite> select * from sqlite_master;
5     type = table
      name = tbl1
7 tbl_name = tbl1
  rootpage = 3
9      sql = create table tbl1(one varchar(10), two smallint)
   sqlite>
```


Algunes ordres útils en sqlite3 (I)

http://www.sqlite.org/sqlite.html

- Una vegada dins de l'intèrpret d'ordres de sqlite, aquest reconeix -a més de la sintáxis de SQL- una sèrie d'ordres directes per dur a terme certes accions.
- Aquestes ordres comencen per un caràcter '**.**'.
- Vegem algunes d'elles:

.help Mostra una breu ajuda de les ordres reconegudes.

```
   sqlite> .help
2  .backup ?DB? FILE      -- Backup DB (default "main")
       to FILE
   .bail ON|OFF           -- Stop after hitting an
       error.  Default OFF
4  ...
```

.databases Mostra les bb.dd. disponibles.

Algunes ordres útils en sqlite3 (II)

http://www.sqlite.org/sqlite.html

.mode list | line | column Canvia el format de la sortida produïda per exemple per sentències `select`.

.output fitxer-sortida.txt Redirigeix la sortida al fitxer `fichero-salida.txt`.

.tables Mostra les taules de la bb.dd.

.indices taula Mostra els índexs de la taula 'tabla'.

.schema Mostra les ordres 'CREATE TABLE' i 'CREATE INDEX' que es van usar per crear la bb.dd. actual. Si li passem com a argument el nom d'una taula, llavors ens mostra l'ordre 'CREATE' usada para crear aquesta taula i els seus índexs.

Algunes ordres útils en sqlite3 (III)

http://www.sqlite.org/sqlite.html

.dump taula Bolca el contingut de la bb.dd. d'aquesta taula en format SQL, per exemple:

```
sqlite> .output /tmp/test.sql
sqlite> .dump tabla
sqlite> .output stdout
```

.read fitxer.sql Llegeix i executa el codi SQL contingut en 'fichero.sql'.

.xou Mostra el valor de diversos ajustos:

```
sqlite> .show
     echo: off
  explain: off
  headers: on
     mode: column
nullvalue: ""
   output: stdout
separator: "|"
    width:
```


Algunes ordres útils en sqlite3 (IV)

http://www.sqlite.org/sqlite.html

.separator char Canvia el separador de camps al caràcter 'char':

```
sqlite> .separator ,
sqlite> .show

     echo: off
  explain: off
  headers: on
     mode: column
nullvalue: ""
   output: stdout
separator: ","
    width:
```

Això ens permet importar dades d'un fitxer 'CSV', p.i., si tenim un fitxer amb aquest contingut:

```
5,value5
6,value6
7,value7
```


Algunes ordres útils en sqlite3 (V)

http://www.sqlite.org/sqlite.html

.import fitxer.csv taula Importa les dades de l'arxiu 'fitxer.csv' en la taula 'taula' línia a línia:

```
sqlite> .import fichero.csv test
sqlite> select * from test;

ids         value
----------  ----------
1           value1
2           value2
3           value3
```

Uso no interactiu de Sqlite3 (I)

- Sqlite3 vaig poder ser cridat amb l'opció '`--help`' i saber que diferents formes de ser invocat té.
- Es pot emprar *sqlite3* com un intèrpret de SQL...de manera que puguem executar des de la línia de ordres:
  1. sentències individuals de SQL.
  2. una sèrie de sentències SQL guardades en un arxiu.

  Vegem uns exemples:

- Una sola sentència:

```
    user@host:~$ sqlite3 -header -column test.db '.schema'
2
    CREATE TABLE test (ids integer primary key, value text);
4   CREATE VIEW testview AS select * from test;
    CREATE INDEX testindex on test (value);
```


Uso no interactiu de Sqlite3 (II)

- Executant una sentència '`SELECT`':

```
1   user@host:~$ sqlite3 -header -column test.db 'select * from
        test'

3   ids         value
    ----------  ----------
5   1           value1
    2           value2
7   3           value3
```

- Exportar una bb.dd.:

```
1   user@host:~$ sqlite3 test.db '.dump' > dbbackup
```

- Executar sentències '`SQL`' guardades en un fitxer:

```
1   user@host:~$ sqlite3 test.db < statements.sql
```

  O també així:

```
1   user@host:~$ cat statements.sql | sqlite3 test.db
```


SQLiteBrowser

- Es tracta d'un interfície gràfic sobre sqlite.
- És senzill d'usar, a més de portable entre Windows/Mac/Linux.
- El pots trobar en el seu web

Ús de sqlite des d'un llenguatge de programació

- Hem vist com usar sqlite des de línia d'ordres i també amb una aplicació amb interfície gràfic com és `sqlitebrowser`.
- Anem a veure ara com podem fer ús de sqlite des d'una aplicació escrita en un llenguatge de programació.
- Veurem primer un exemple en 'C' atès que seria el llenguatge 'original' per treballar amb sqlite...
- I després veurem un exemple també molt senzill en Vala.

Sqlite en C (I)

- El codi de les dues següents transparències es guarda en un arxiu anomenat '`sqlite-example.c`'.
- Es compila amb l'ordre: '*gcc sqlite-example.c -o sqlite-example -lsqlite3*'

```
1  #include <stdio.h>
   #include <sqlite3.h>
3
   static int callback_fn(void *NotUsed,
5                         int argc, char **argv,
                          char **azColName){
7    int i;

9    for(i=0; i<argc; i++) {
       printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
11   }
     printf("\n");
13
     return 0;
15 }
```


21 / 27

Sqlite en C (II)

```
1  int main(int argc, char **argv) {
     sqlite3 *db;
3    char *zErrMsg = 0;
     int rc;
5
     if( argc!=3 ) {
7      fprintf(stderr, "Usage: %s DATABASE SQL-STATEMENT\n", argv[0]);
       return(1);
9    }

11   rc = sqlite3_open(argv[1], &db);
     if( rc ) {
13     fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
       sqlite3_close(db);
15     return(1);
     }
17
     rc = sqlite3_exec(db, argv[2], callback_fn, 0, &zErrMsg);
19   if( rc!=SQLITE_OK ) {
       fprintf(stderr, "SQL error: %s\n", zErrMsg);
21     sqlite3_free(zErrMsg);
     }
23
     sqlite3_close(db);
25   return 0;
   }
```


Sqlite en C (III)

- Aquesta aplicació té dos arguments, el primer és la bb.dd. amb la qual treballar i el segon, l'ordre SQL que li donem.
- Encara que no ho sembli...hem creat una versió senzilla de la aplicació de línia d'ordres *sqlite3*.
- Podem executar coses com aquestes:

```
     sqlite-example test.db "create table test (ids integer primary
           key , value text )";
2    sqlite-example test.db "insert into test values('hola', 10);"
     sqlite-example test.db "insert into test values('adios',20);"
4    sqlite-example test.db "select * from test;"
     ids = hello
6    value = 10

8    ids = goodbye
     value = 20
```

Sqlite en Vala (I)

- Vegem exactament el mateix exemple en Vala.
- Ho compilarem amb la següent ordre: '*valac –pkg sqlite3 sqlitesample.vala*'.

```
1  using Sqlite;

3  public class SqliteSample : GLib.Object {
     public static int callback (int n_columns,
5                                string[] values,
                                 string[] column_names)
7    {
       for (int i = 0; i < n_columns; i++) {
9        stdout.printf ("%s = %s\n", column_names[i], values[i]);
       }
11
       stdout.printf ("\n");
13     return 0;
     }
15 }
```


Sqlite en Vala (II)

```
1  public class SqliteSample : GLib.Object {
     public static int main (string[] args) {
3      Database db;
       int rc;
5
       if (args.length != 3) {
7        stderr.printf ("Usage: %s DATABASE SQL-STATEMENT\n", args[0]);
         return 1;
9      }

11     if (!FileUtils.test (args[1], FileTest.IS_REGULAR)) {
         stderr.printf ("Database %s does not exist or is directory\n",
              args[1]);
13       return 1;
       }
15
       rc = Database.open (args[1], out db);
17     if (rc != Sqlite.OK) {
         stderr.printf ("Can't open database: %d, %s\n", rc, db.errmsg ()
              );
19       return 1;
       }
21
       rc = db.exec (args[2], callback, null);
23     if (rc != Sqlite.OK) {
         stderr.printf ("SQL error: %d, %s\n", rc, db.errmsg ());
25       return 1;
       }
27
       return 0;
29   }
   }
```


Arquitectura de capes (I)

Entitat de negoci (EN), Componente d'acces a dades (CAD).

- Amb la finalitat d'obtindre un codi més llegible y fàcilment mantenible per aquest tipus d'aplicacions anem a veure com estructurar el seu codi font.
- Proposem seguir un patró de capes[1] per dividir el codi de l'aplicació segons '*divisions*' lógiques...similar a como vam veure amb *MVC*.
- Cadascuna d'aquestes '*divisions*' es desenvolupa y manté per separat.

[1] Serà ampliat posteriorment en la part web.

Arquitectura de capes (II)

Entitat de negoci (EN), Componente d'acces a dades (CAD).

Dividirem el codi en tres capes o components:

- Capa de interficie de usuari.
- Capa de lògica de negoci o *Entitat de Negoci* (**EN**).
  - Sería l'equivalent a lo que en *MVC* coneixem com la Capa del *Model*.
  - Se li asocia un *CAD* mitjançant el qual pot almacenarse/modificarse/recuperarse... en la bb.dd. amb la que treballarem.
- Capa de persistència o *Component d'Acces a Dades* (**CAD**).
  - Els *CAD* implementen la lògica de comunicació amb la bb.dd. la cual es bidireccional entre las *EN* y la bb.dd.
  - Les operacions habituals que proporciona un *CAD* son les de **creació**, **lectura**, **actualització** y **esborrat** de registres de la bb.dd.