Appendice 5 – Uso di un database "embedded" con SQLite

La libreria SQLite (https://www.sqlite.org) consente di creare e gestire un database di tipo relazionale incorporato nella propria applicazione.

Il database è costituito da un singolo file che viene agevolmente inserito all'interno delle cartelle dell'applicazione e trasportato assieme alla stessa.

Per poter utilizzare il database non c'è bisogno di alcuna installazione o configurazione, ne' di avere a disposizione un apposito servizio software, aggiuntivo rispetto all'applicazione stessa.

Con SQLite si crea, tipicamente, un database pensato per un uso personale, tuttavia SQLite è anche in grado di

- gestire l'accesso concorrente di più utenti,
- applicare i vincoli di integrità del database,
- garantire transazioni con proprietà ACID (Atomicità, Consistenza, Isolamento, Persistenza).

Le prestazioni, in termini di numero di accessi simultanei al database, sono inferiori a quelle di un database gestito da un vero server, come MySQL, SQLServer, Oracle; tuttavia SQLite può venire impiegato, oltre che per una archiviazione di dati per uso personale, come quella inserita nei browser web, anche per realizzare un prototipo funzionante di una applicazione multiutente o per effettive applicazioni web di medio-piccole dimensioni, dove non si ha a disposizione un servizio di DBMS o non se ne vogliono sostenere i costi!

Si ha anche il vantaggio di poter spostare agevolmente l'applicazione con il suo database da un server web ad un altro.

Per una discussione sulla convenienza di uso di SQLite rispetto ad una soluzione client-server tradizionale si veda ad esempio "Usi appropriati di SQLite" (https://code-examples.net/it/docs/sqlite/whentouse).

Per la creazione di un database nel pc di sviluppo e l'inserimento preliminare di alcuni dati si può utilizzare il software opensource **SQLiteStudio**, https://sqlitestudio.pl/

Nell'esempio in esame, il database è costituito dal file negozio.db, che viene collocato nella cartella db, appositamente creata dentro la cartella application.

La cartella che contiene il database non risulta accessibile direttamente dal browser grazie alla presenza del file .htaccess nella cartella application.

Tentativo di accedere direttamente alla cartella contenente il database

Per la amministrazione del database tramite interfaccia web dopo che è stato caricato nel sito web, si può usare il programma **phpLiteAdmin** https://www.phpliteadmin.org .

Esso è costituito da un singolo file eseguibile di nome phpliteadmin.php e da un file di configurazione di nome phpliteadmin.config.php.

Nel caso in esame, il programma vien messo nella cartella phpLiteAdmin, che è stata creata direttamente all'interno della cartella radice del sito web.

Nel file di configurazione si devono specificare il nome e il percorso del database che si vuole gestire e la password per potervi accedere:

phpliteadmin.config.php

```
//password to gain access
$password = 'admin';

//directory relative to this file to search for databases (if false,
manually list databases in the $databases variable)
//$directory = '.';
$directory = '../application/db';

//whether or not to scan the subdirectories of the above directory
infinitely deep
$subdirectories = false;

//if the above $directory variable is set to false, you must specify
the databases manually in an array as the next variable
//if any of the databases do not exist as they are referenced by their
path, they will be created automatically
$databases = array(
      array(
            'path'=> 'negozio.db',
            'name'=> 'negozio'
      ),
      //array(
      //    'path'=> 'database2.sqlite',
      //    'name'=> 'Database 2'
      //),
);
```

Per poter usare questo database, **non occorre cambiare nulla nel codice della applicazione** che è stata sviluppata per MySQL: si deve soltanto impostare opportunamente il file di configurazione database.php della cartella config in modo che si vada ad utilizzare il database negozio.db di SQLite:

database.php

```
$db['default'] = array(
      'dsn' => 'sqlite:application/db/negozio.db', // Data Source Name
      'hostname' => '',
      'username' => '',
      'password' => '',
      'database' => '',
      'dbdriver' => 'pdo',  // driver universale PHP Data Objects
      'dbprefix' => '',
      'pconnect' => FALSE,
      'db_debug' => FALSE,  // (ENVIRONMENT !== 'production'),
```

```
	'cache_on' => FALSE,
	'cachedir' => '',
	'char_set' => 'utf8',
	'dbcollat' => 'utf8_general_ci',
	'swap_pre' => '',
	'encrypt' => FALSE,
	'compress' => FALSE,
	'stricton' => FALSE,
	'failover' => array(),
	'save_queries' => TRUE
);
```

Ci si deve accertare che il file php.ini della cartella php preveda il caricamento della dll corrispondente al driver per SQLite (extension library to load):

php.ini

```
extension=pdo_sqlite
```

Non deve esserci il punto e virgola all'inizio della riga!

La classe PDO (PHP Database Objects) è stata introdotta dalla versione 5 di php e consente di unificare l'accesso a diversi DBMS. In questo modo il programmatore può agevolmente cambiare il software di gestione del database (DBMS) senza dover modificare il codice applicativo scritto in php.

ATTENZIONE che SQLite richiede di **abilitare esplicitamente i controlli di integrità sulle chiavi esterne** perché essi sono disabilitati di default per questioni di compatibilità con le versioni di SQLite antecedenti al 2009.

Pertanto, ad ogni connessione si deve dare preliminarmente il comando:

```
PRAGMA foreign_keys = ON;
```

In sostanza, ciascun metodo che agisce per aggiornare i dati del database, dovrà contenere le istruzioni:

```
$this->db->query("PRAGMA foreign_keys = ON");
$this->db->query(" INSERT……..")  // oppure DELETE oppure UPDATE
```