Introdução ao SGBD SQLite

Wikipédia - http://pt.wikipedia.org/wiki/SQLite
SQLite é uma biblioteca C que implementa um banco de dados SQL embutido. Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo RDBMS separado.

SQLite não é uma biblioteca de cliente usada para conectar com um grande servidor de banco de dados. **SQLite é o servidor.** A biblioteca **SQLite lê e escreve diretamente para e do arquivo do banco de dados no disco.**

O uso do SQLite é recomendado onde a simplicidade da administração, implementação e manutenção são mais importantes que incontáveis recursos que SGBDs mais voltados para aplicações complexas possivelmente implementam. Entretando situações onde a simplicidade é a melhor escolha são muito mais freqüentes do que pode-se imaginar.

Exemplos de uso do SQLite não restrito a :

- sites com menos de cem mil requisições por dia,
- dispositivos e sistemas embarcados,
- aplicações desktop,
- ferramentas estatísticas e de análise,
- aprendizado de banco de dados,
- implementação de novas extensões à SQL.

Não se recomenda o uso do SQLite para sites com :

- muitos acessos,
- grande quantidades de dados (talvez maior que algumas duzias de gigabytes),
- sistemas com grande concorrência,
- aplicações cliente/servidor.

Algumas Características do SQLite:

- Software Livre/domínio público e Multiplataforma
- Mecanismo de armazenamento seguro com transações ACID
- Não necessita de instalação, configuração ou administração
- Implementa a maioria do SQL92
- O Banco de Dados é guardado em um único arquivo
- Suporta bases de dados acima de 2 terabytes
- Sem dependências externas

SQLite (http://www.sqlite.org) é uma pequena biblioteca C que implementa um SGBD SQL completo, embutido e sem necessitar de configurações.

Recursos:
Subselects,
Triggers,
Transactions,
Views
Mais rápido 2 a 3 vezes que MySQL ou PostgreSQL em muitas operações
Limite de 2TB

Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo RDBMS separado.

A biblioteca SQLite lê e escreve diretamente para e do arquivo de banco de dados no disco.

Características atuais:

- Transações são atômicas, consistentes, isoladas e duráveis (ACID) mesmo que o sistema trave ou a energia falhe.
- Configuração-zero - nenhuma instalação ou administração necessária.
- Implementação da maior parte do SQL92.
- Um banco de dados completo é armazenado em apenas um arquivo de sistema.
- Arquivos de banco de dados podem ser livremente compartilhados entre máquinas com diferentes ordens de byte.
- Suporta bases de dados de até 2 terabytes de tamanho.
- Tamanho de strings e BLOBs limitados apenas pela memória disponível.
- Mais rápido que populares bancos de dados cliente/servidor para a maioria das operações comuns.
- API simples e fácil de usar.
- TCL bindings inclusas. Bindings para a maioria das linguagens disponíveis separadamente.
- Código fonte bem comentado, com mais de 95% coberto por testes.
- Auto-contido: sem dependências externas.
- Fontes estão em domínio público. Use para qualquer propósito.

A distribuição SQLite vem com um programa de linha de comando (sqlite (http://sqlitebrasil.codigolivre.org.br/?pagina=doc/sqlite)) que pode ser usado para administrar um banco de dados SQLite e que serve como exemplo de como usar a biblioteca SQLite.

Além do programa em linha de comando, você pode utilizar alguns dos programas de terceiros com interface gráfica, como o SQLiteManager (http://sqlitemanager.sourceforge.net/) (web, no estilo PHPMyAdmin) ou o SQLiteBrowser (http://sqlitebrowser.sourceforge.net/) (QT).

O SQLite está embutido no PHP (http://br.php.net/sqlite) 5 e disponível como extensão no PHP 4. Assim, qualquer aplicação PHP pode utilizar um banco de dados sem necessitar de um SGDB.

Há um driver (http://dba.openoffice.org/drivers/sqlite/index.html)

(alpha) disponível para conectar bancos de dados SQLite com o OpenOffice.org. Tornando possível a criação de relatórios, formulários, etc.

Suas características o tornam ideal para desenvolver programas standalone, pequenos e médios sites, etc. Veja quando usar (e não usar) (http://localhost/sqlitewww/?pagina=doc/quando) o SQLite.

SQLite Brasil

A Comunidade SQLite Brasil (http://sqlitebrasil.codigolivre.org.br) tem o objetivo de difundir o uso do SQLite entre os desenvolvedores brasileiros, além de ser um canal de informação para quem já utliza o SQLite em suas aplicações.

Atualmente possuimos duas listas de discussão, uma para os usuários do SQLite no Brasil e outra dos desenvolvedores do projeto, um canal IRC e uma comunidade no Orkut.

Criando Banco

sqlite clientes (Com isso ele cria o banco clientes e já acessa a console deste banco)

Criando Tabela (A sintaxe não tem diferença dos grandes SGBDs, é puro SQL)

```
CREATE TABLE cliente (
    cpf VARCHAR(11) PRIMARY KEY,
    nome VARCHAR(45),
    fone VARCHAR(10)
);
```

Inserindo Registros (Também puro SQL)

```
INSERT INTO cliente (cpf, nome, fone) VALUES ('11111111111', 'João Abreu', '34543456');
```

Consultando registros

```
SELECT * FROM cliente;
```

Outros comandos do sqlite. Estando na console apenas digite ".help":

```
sqlite> .help
.databases            List names and files of attached databases
.dump ?TABLE? ...     Dump the database in a text format
.echo ON|OFF          Turn command echo on or off
.exit                 Exit this program
```

```
.explain ON|OFF        Turn output mode suitable for EXPLAIN on or off.
.header(s) ON|OFF      Turn display of headers on or off
.help                  Show this message
.indices TABLE         Show names of all indices on TABLE
.mode MODE             Set mode to one of "line(s)", "column(s)", "insert", "list", or "html"
.mode insert TABLE     Generate SQL insert statements for TABLE
.nullvalue STRING      Print STRING instead of nothing for NULL data
.output FILENAME       Send output to FILENAME
.output stdout         Send output to the screen
.prompt MAIN CONTINUE  Replace the standard prompts
.quit                  Exit this program
.read FILENAME         Execute SQL in FILENAME
.schema ?TABLE?        Show the CREATE statements
.separator STRING      Change separator string for "list" mode
.show                  Show the current values for various settings
.tables ?PATTERN?      List names of tables matching a pattern
.timeout MS            Try opening locked tables for MS milliseconds
.width NUM NUM ...     Set column widths for "column" mode
```

Escrevendo o Resultado em um Arquivo:

```
sqlite> .mode list
sqlite> .separator |
sqlite> .output test_file_1.txt
sqlite> select * from tbl1;
sqlite> .exit
$ cat test_file_1.txt
hello|10
goodbye|20
```

Usando SQLite com PHP

```
<?php

$db=sqlite_open("/home/ribafs/bancos/clientes.db");

$registros = sqlite_query($db, "SELECT * FROM cliente");

while ($i = sqlite_fetch_array($registros)) {
    print $i[0] . "__" . $i[1]. "__" . $i[2]. "<br>";
}

?>
```

Comandos com o PHP

```
<?php
    // Pequeno CRUD com PHP e SQLite
    // Abrir o banco de dados (deve criar, caso não exista)
    $db = sqlite_open("clientes.db");  // Lembrar que o diretório deve ter permissão de escrita

    // Criar tabela
    //$resultT = sqlite_query($db, "CREATE TABLE clientes(codigo int PRIMARY KEY, nome
VARCHAR(45), email VARCHAR(50));");

    // Inserir registro
    //$resultI = sqlite_query($db, "INSERT INTO clientes(codigo, nome, email) VALUES (1,
'Ribamar', 'ribafs@ribafs.net')");

    // Atualizar registro
    $resultU = sqlite_query($db, "UPDATE clientes set nome = 'Ribamar FS' WHERE codigo=1");
    if(!$resultU) print "Erro na atualização ".sqlite_last_error();

    // Consultar
    $resultS = sqlite_query($db, "SELECT * from clientes");

    // Captura cada registro com um array de campos. Primeiro campo $row[0], segundo $row[1], ...
    echo "<h1 align=center>CRUD em PHP com SQLite</h1>";
    echo "<center><table border=1>";
    while ($row = sqlite_fetch_array($resultS)) {
        echo "<tr><td>Código</td><td>" . $row[0] . "</td></tr><tr><td>Nome</td><td>" .
$row[1] . "</td></tr><tr><td>E-mail</td><td>".$row[2]."</td></tr>";
    }
    echo "</table><center>";
    // close the database
sqlite_close($db);
?>
```

```
<?php
// Default result type SQLITE_BOTH

$result = sqlite_query($db,
          "SELECT first, last from names");
$row = sqlite_fetch_array($result);

print_r($row);
?>

<?php
// column names only

$result = sqlite_query($db, "SELECT first, last from names");
$row = sqlite_fetch_array($result, SQLITE_ASSOC);

print_r($row);
?>

<?php
// column numbers only

$result = sqlite_query($db, "SELECT first, last from names");
$row = sqlite_fetch_array($result, SQLITE_NUM);

print_r($row);
?>

<?php
// Collecting all rows from a query

// Get the rows as an array of arrays of data
$rows = array();

$result = sqlite_query($db, "SELECT first, last from names");

// grab each row
while ($row = sqlite_fetch_array($result)) {
    $rows[] = $row;
}

// Now use the array; maybe you want to
// pass it to a Smarty template
$template->assign("names", $rows);
?>
```

```
<?php
// The same but with less typing and
// more speed

// Get the rows as an array of arrays of data
$rows = sqlite_array_query($db, "SELECT first, last from names");

// give it to Smarty
$template->assign("names", $rows);
?>


<?php

$count = sqlite_single_query($db, "SELECT count(first) from names");

echo "There are $count names";
?>


<?php

$first_names = sqlite_single_query($db, "SELECT first from names");

print_r($first_names);
?>


<?php
function md5_and_reverse($string) {
    return strrev(md5($string));
}

sqlite_create_function($db,
     'md5rev', 'md5_and_reverse');

$rows = sqlite_array_query($db,
          'SELECT md5rev(filename) from files');
?>


<?php

function max_len_step(&$context, $string) {
    if (strlen($string) > $context) {
        $context = strlen($string);
    }
}
```

```
function max_len_finalize(&$context) {
    return $context;
}

sqlite_create_aggregate($db,
    'max_len', 'max_len_step',
    'max_len_finalize');

$rows = sqlite_array_query($db,
    'SELECT max_len(a) from strings');

    print_r($rows);
?>
```

Erros:

int sqlite_last_error (resource db)
Returns last error code from database

string sqlite_error_string (int error_code)
Converts error code to a readable string

Ferramentas

Existe um gerenciador web para SQLite em PHP - http://phpsqliteadmin.sourceforge.net/
Download -http://downloads.sourceforge.net/project/phpsqliteadmin/phpSQLiteAdmin/phpSQLiteAdmin-0.3-Alpha1/phpSQLiteAdmin-0.3a1.tar.gz?use_mirror=ufpr

- Faça o download, descompacte no diretório web e renomeie para phpsqliteadmin
- Crie um diretório dentro dele chamado sqlite e dê permissão de escrita
- Ao abrir no navegador o usuário e senha default são "root" e "root".
- Também dê permissão de escrita no arquivo db/phpsla.sqlite

Para a criação de bancos use a linha de comando, com:

sqlite nomebanco

E também uma extensão para o Firefox -
http://code.google.com/p/sqlite-manager/