Medii vizuale de programare
Curs 4

Conf. dr.ing. GENGE Béla

Universitatea "Petru Maior", Departamentul de Informatică
Tîrgu Mureș, Romania
bela.genge@ing.upm.ro

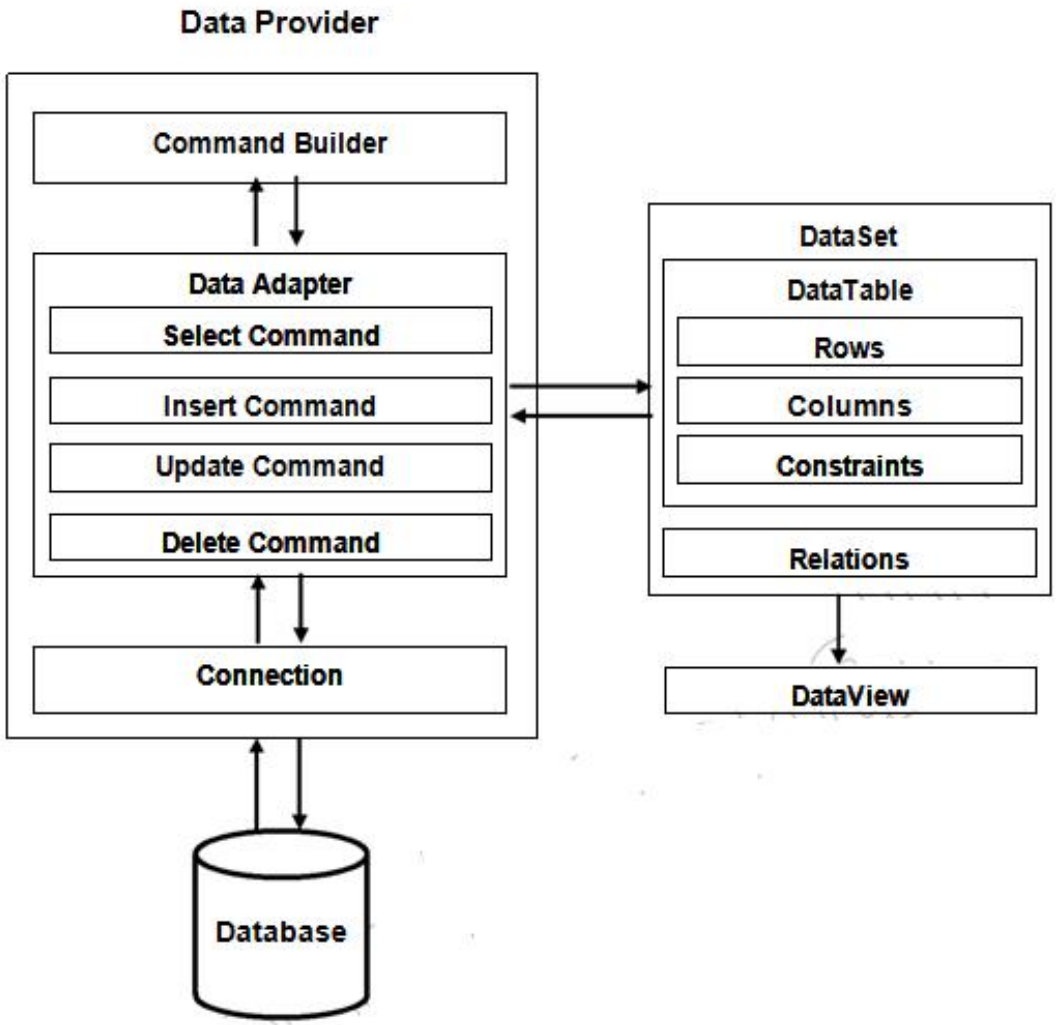
ADO.NET

- Un set de clase ce permite accesul la colecțiile de date
- Programatorului îi sunt oferite opțiuni bogate pentru conectarea la o bază de date
- Conexiunile pot fi permanente (scalabilitate redusă) sau temporare

ADO.NET - arhitectura

- Entitățile principale: `DataSet` și `DataProvider`
- `DataSet`:
  - Asigură o reprezentare în memorie a datelor
  - Implică realizarea unei copii locale a datelor asupra cărora se lucrează
- `DataProvider`:
  - Asigură legătura cu baza de date
  - Furnizori de date:
    - Microsoft Access Database (OleDb .NET)
    - Microsoft ODBC
    - Microsoft SQL Server
    - Oracle

ADO.NET - arhitectura (sursa: programcall.com)



Limbajul SQL

- SQL: Structured Query Language.
- Limbaj standardizat pentru accesarea bazelor de date.
- SGBD: Sisteme de Gestiune a Bazelor de Date: set de programe care asigură interfața dintre o bază de date și utilizatorii acestuia.
- Exemple de SGBD-uri:
  - Oracle SQL server.
  - MySQL server.
  - MS SQL Server.
  - **SQLite**.

SQLite

- SQLite: bibliotecă software integrată ce implementează un motor SQL integrat.

About Sitemap Documentation Download License News Support

Welcome

SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed database engine in the world. The source code for SQLite is in the public domain. More...

Appropriate Uses For SQLite

SQLite is not directly comparable to client/server SQL database engines such as MySQL, Oracle, PostgreSQL, or SQL Server since SQLite is trying to solve a different problem.

Client/server SQL database engines strive to implement a shared repository of enterprise data. They emphasis scalability, concurrency, centralization, and control. SQLite strives to provide local data storage for individual applications and devices. SQLite emphasizes economy, efficiency, reliability, independence, and simplicity.

SQLite does not compete with client/server databases. SQLite competes with fopen().

SQL în SQLite

- Limbajul SQL în SQLite.

- aggregate functions
- ALTER TABLE
- ANALYZE
- ATTACH DATABASE
- BEGIN TRANSACTION
- comment
- COMMIT TRANSACTION
- core functions
- CREATE INDEX
- CREATE TABLE
- CREATE TRIGGER
- CREATE VIEW
- CREATE VIRTUAL TABLE
- date and time functions
- DELETE
- DETACH DATABASE
- DROP INDEX
- DROP TABLE
- DROP TRIGGER
- DROP VIEW
- END TRANSACTION
- EXPLAIN
- expression
- INDEXED BY
- INSERT
- keywords
- ON CONFLICT clause
- PRAGMA
- REINDEX
- RELEASE SAVEPOINT
- REPLACE
- ROLLBACK TRANSACTION
- SAVEPOINT
- SELECT
- UPDATE
- VACUUM
- WITH clause

SQL în SQLite

- Tipuri de date: INTEGER, REAL, TEXT, BLOB.
  - NULL: Valoare NULL.
  - INTEGER: Valoare întreagă (de la 1 la 8 octeți în funcție de dimensiunea datelor).
  - REAL: Valoare reală (8 octeți).
  - TEXT: Text (UTF).
  - BLOB: Date binare.

Consolă pentru testarea SQLite

- Descărcare shell SQLite: `http://www.sqlite.org/download.html`

Command Line Shell For SQLite

The SQLite project provides a simple command-line utility named **sqlite3** (or **sqlite3.exe** on windows) that allows the user to manually enter and execute SQL statements against an SQLite database. This document provides a brief introduction on how to use the **sqlite3** program.

Getting Started

To start the **sqlite3** program, just type "sqlite3" optionally followed by the name the file that holds the SQLite database. If the file does not exist, a new database file with the given name will be created automatically. If no database file is specified, a temporary database is created, then deleted when the "sqlite3" program exits.

When started, the **sqlite3** program will show a brief banner message then prompt you to enter SQL. Type in SQL statements (terminated by a semicolon), press "Enter" and the SQL will be executed.

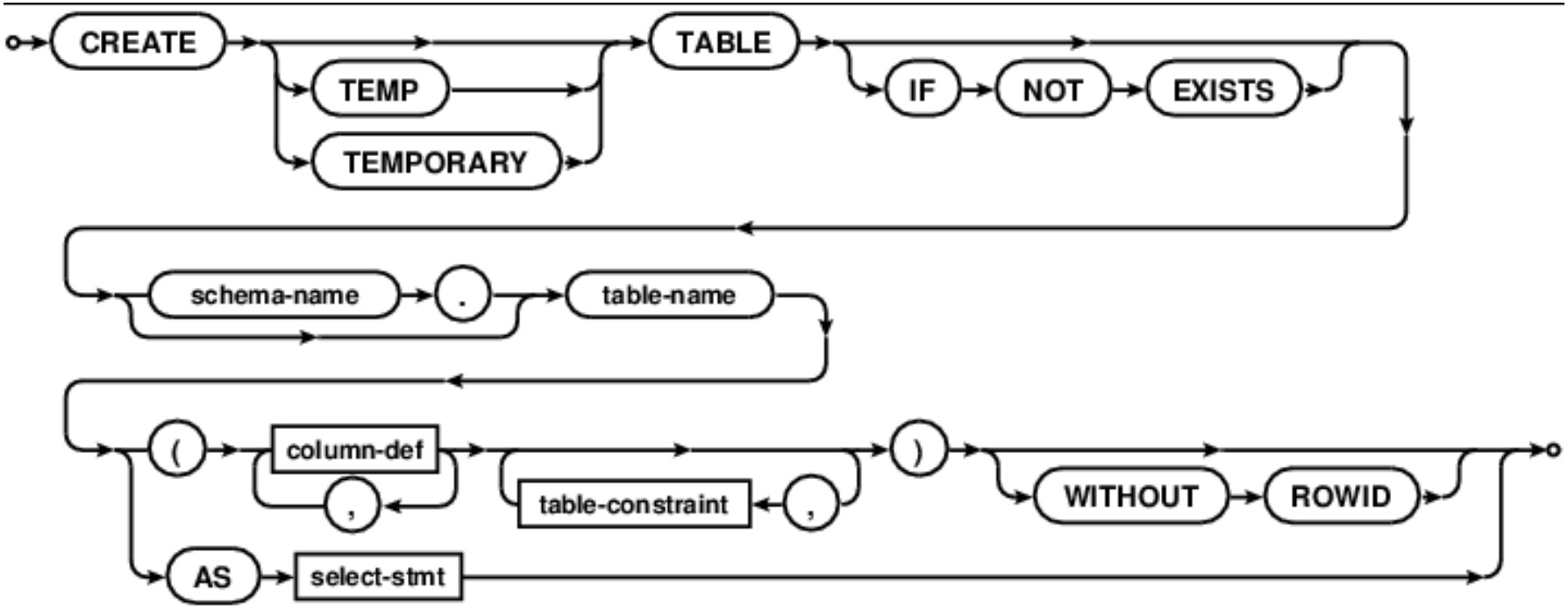
For example, to create a new SQLite database named "ex1" with a single table named "tbl1", you might do this:

```
$ sqlite3 ex1
SQLite version 3.8.5 2014-05-29 12:36:14
```

Comenzi consolă SQLite

- .help: lista comenzilor.
- .databases: lista bazelor de date (se creează automat “main” și “temp”).
- .schema: lista comenzilor folosite pentru crearea structurilor din baza de date.
- .tables: lista tabelelor.
- Atenție! Comenzile NU se termină cu “;”.
- Rulare: sqlite3.exe (fără argument va crea o bază de date în memorie care se va șterge la revenire din program).
- Rulare: sqlite3.exe NumeFisier (va crea/deschide fișierul).
- Utilizare (adăugarea la connector) mai multe baze de date: comanda ATTACH NumeFisier (DETACH NumeFisier).

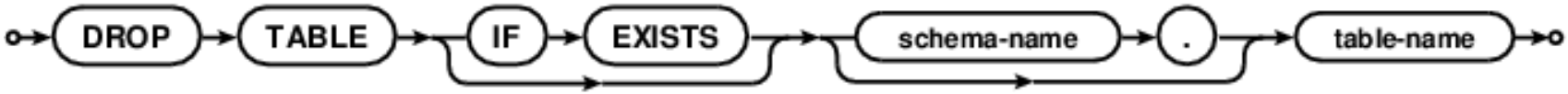
SQL în SQLite: CREATE TABLE



SQL Statement: creare tabel

```
CREATE TABLE Users(id INTEGER PRIMARY KEY AUTOINCREMENT,
Nume TEXT, Prenume TEXT, AnNastere INTEGER)
```

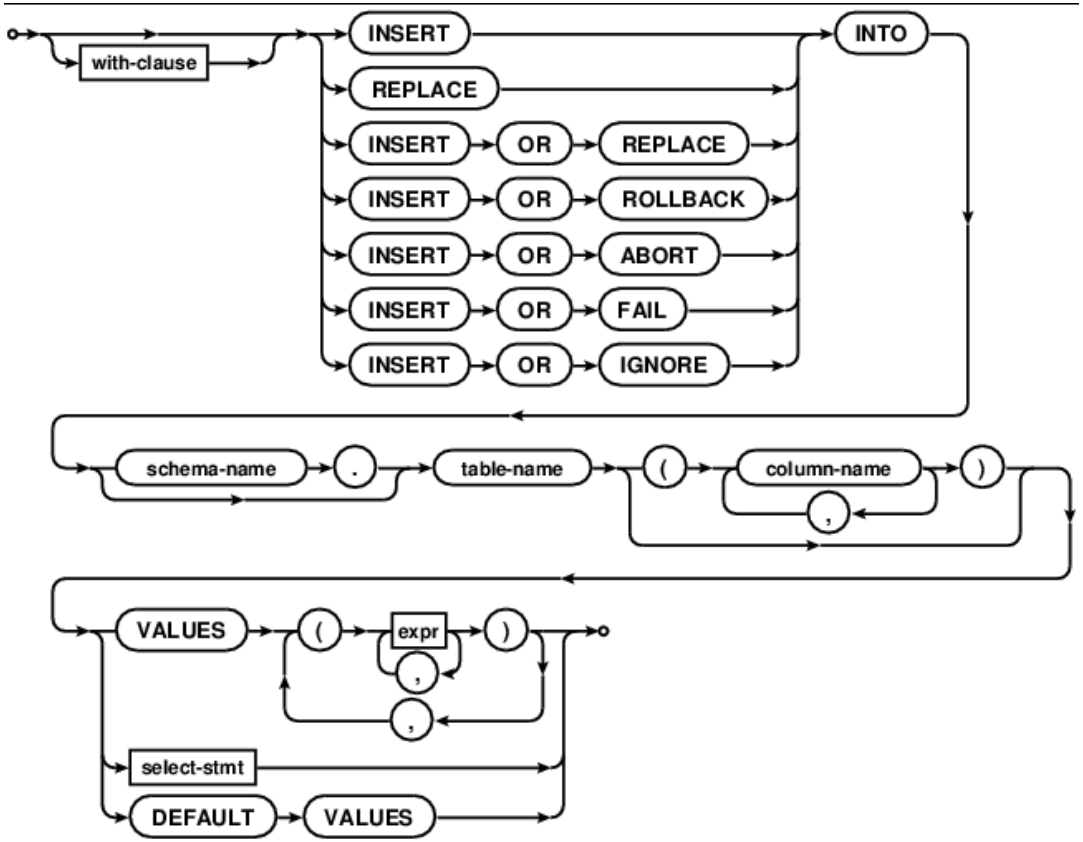
SQL în SQLite: DROP TABLE



SQL Statement: ștergere tabel

```
DROP TABLE Users
```

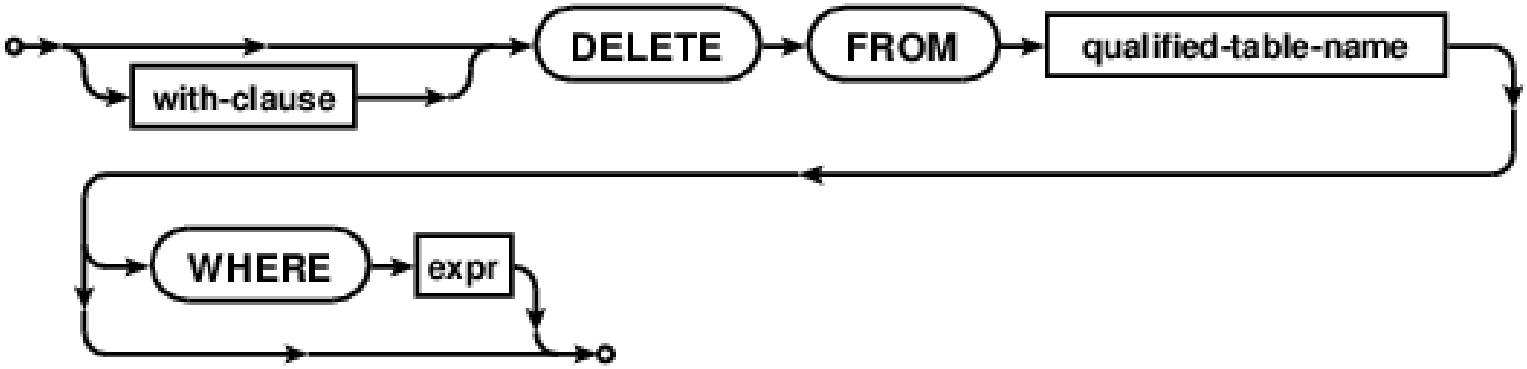
SQL în SQLite: INSERT INTO



SQL Statement: inserare înregistrare

```
INSERT INTO Users(Nume, Prenume, AnNastere)
VALUES('Popescu', 'Adrian', '1980')
```

SQL în SQLite: DELETE FROM



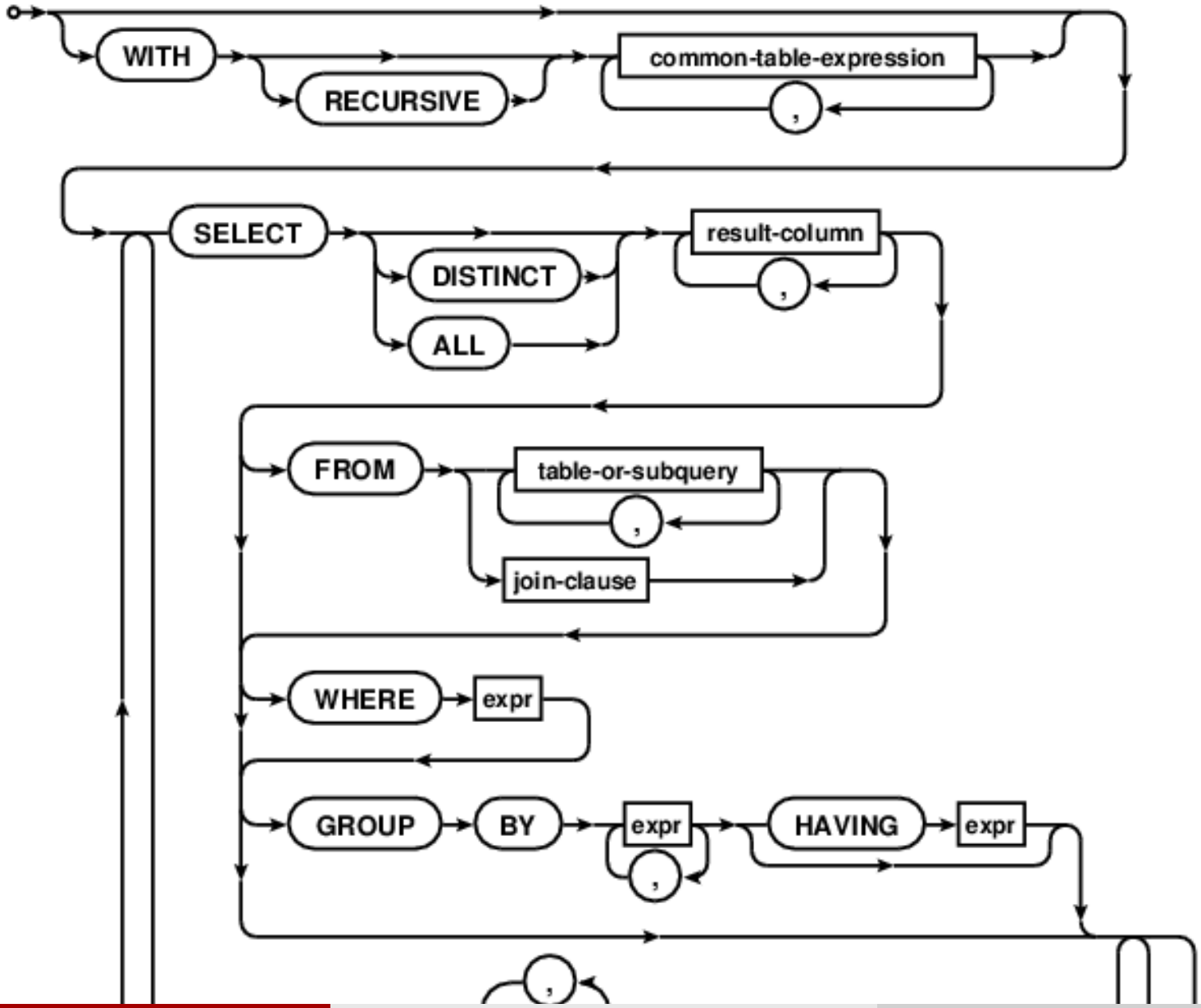
SQL Statement: ștergere înregistrare

```
DELETE FROM Users WHERE Nume='Popescu'
```

SQL Statement: ștergerea tuturor înregistrărilor

```
DELETE FROM Users WHERE Nume='Popescu'
```

SQL în SQLite: SELECT



SQL în SQLite: SELECT

SQL Statement: selectare toate înregistrări

```
SELECT * FROM Users
```

SQL Statement: selecție condiționată

```
SELECT * FROM Users WHERE Nume='Popescu'
```

SQL Statement: selecție coloane

```
SELECT id,Prenume FROM Users WHERE Nume='Popescu'
```

SQLite în .NET

- Instalare:
  - Binare și arhive: `https://system.data.sqlite.org/index.html/doc/trunk/www/downloads.wiki`
  - Setup (.exe) și configurare automată.
  - Descărcare arhivă ZIP, copiere DLL-uri în directorul aplicației, adăugare referință către `System.Data.SQLite.DLL`.
- Atenție la opțiuni (trebuie să coincidă cu configurația de compilare și cu configurația sistemului pe care va rula aplicația):
  - Platformă: 32-biți sau 64-biți.
  - .NET Framework: 2.0, 3.5, 4.0, 4.5, ...
  - Mixed-mode assembly: conține atât cod managed cât și cod unmanaged (util dacă se dorește accesarea din Visual C++).
  - Binaries: doar cod managed.

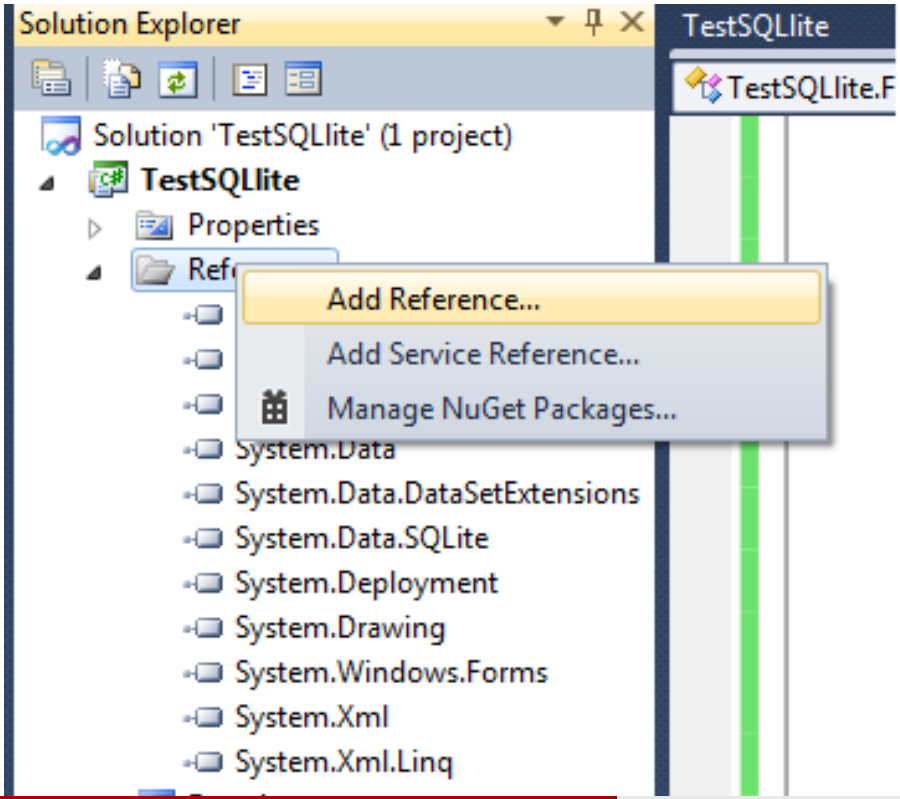
Precompiled Binaries for 64-bit Windows (.NET Framework 3.5 SP1)

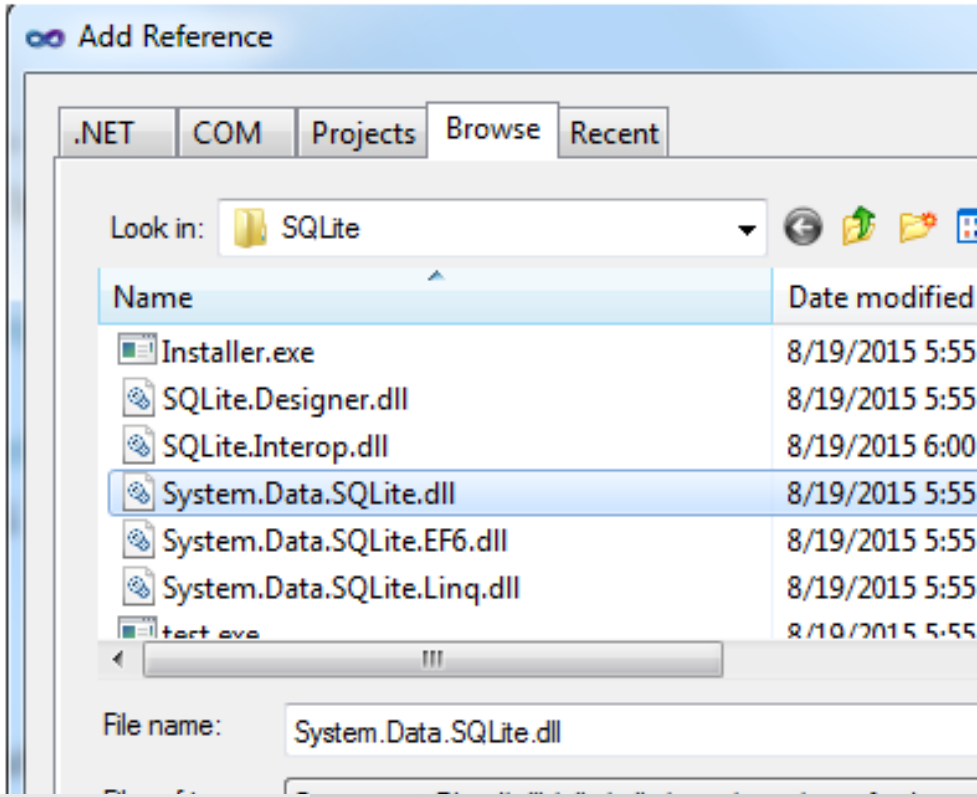
sqlite-netFx35-binary-bundle-x64-2008-1.0.98.0.zip (2.19 MiB)

This binary package features the mixed (3.8.11.1) package. The Visual C++ 20 (sha1: 8be5e8a3a097e8ac56bf548542

SQLite în .NET

- Adăugare la aplicație:
  - Adăugare referință către `System.Data.SQLite.DLL`.
  - Copiere `System.Data.SQLite.DLL` și `SQLite.Interop.DLL` în `bin -> Debug`.
  - Adăugare: `using System.Data.SQLite;`





Conectare la baza de date

- Adăugare la aplicație:
  - Adăugare referință către `System.Data.SQLite.DLL`.
  - Copiere `System.Data.SQLite.DLL` și `SQLite.Interop.DLL` în `bin -> Debug`.
  - Adăugare: `using System.Data.SQLite;`

Creare conexiune

```
if (!File.Exists("MyDatabase.sqlite")){
SQLiteConnection.CreateFile("MyDatabase.sqlite");
}
try{
_conn = new SQLiteConnection("Data Source=MyDatabase.sqlite;
Version=3;");
_conn.Open();
}
```

Închidere conexiune la baza de date

Închidere conexiune

```
if (null != _conn){
_conn.Close();
_conn.Dispose();
_conn = null;
}
```

Comenzi SQL

- Toate comenzile SQL se construiesc sub forma unor șiruri de caractere (string).
- Execuția comenzilor cu `SQLiteCommand`.

Creare comandă SQL

```
string stmt = "...";
SQLiteCommand command = new SQLiteCommand(stmt, _conn);
command.ExecuteNonQuery();
//sau
command.ExecuteScalar();
//sau
command.ExecuteReader();
```

Metode apelate din SQLiteCommand

- ExecuteNonQuery: comenzi care nu returnează înregistrări, e.g., INSERT, UPDATE, DELETE.
- ExecuteScalar: comenzi care returnează o singură valoare, e.g., count.
- ExecuteReader: comenzi care returnează un set de date, e.g., SELECT, în DataReader.

SQLite - creare tabel

- Construire comandă.
- Apel `ExecuteNonQuery`.
- Tratare eroare (excepție).

SQL Statement: creare tabel

```
string stmt="CREATE TABLE Users(id INTEGER PRIMARY KEY
AUTOINCREMENT, Nume TEXT, Prenume TEXT, AnNastere INTEGER)";
SQLiteCommand cmd = new SQLiteCommand(stmt, _conn);
try{
cmd.ExecuteNonQuery();
}
catch (Exception ex){
MessageBox.Show("Unable to create table! ERROR:" +
ex.ToString());
return;
}
```

SQLite - creare tabel - dacă nu există deja

- Construire comanda.
- Apel `ExecuteNonQuery`.
- Tratare eroare (excepție).

SQL Statement: creare tabel

```
string stmt="CREATE TABLE IF NOT EXISTS Users(id INTEGER
PRIMARY KEY AUTOINCREMENT, Nume TEXT, Prenume TEXT,
AnNastere INTEGER)";
SQLiteCommand cmd = new SQLiteCommand(stmt, _conn);
try{
cmd.ExecuteNonQuery();
}
catch (Exception ex){
MessageBox.Show("Unable to create table! ERROR:" +
ex.ToString());
return;
}
```

SQLite - inserare înregistrare

- Construire comandă.
- Apel `ExecuteNonQuery`.
- Tratare eroare (excepție).

SQL Statement: inserare

```
string stmt="INSERT INTO Users(Nume, Prenume, AnNastere)
VALUES('Popescu', 'Adrian', 1980)";
SQLiteCommand cmd = new SQLiteCommand(stmt, _conn);
try{
cmd.ExecuteNonQuery();
}
catch (Exception ex){
MessageBox.Show("Unable to insert into table! ERROR:" +
ex.ToString());
return;
}
```

SQLite - ștergere înregistrare

- Construire comandă.
- Apel `ExecuteNonQuery`.
- Tratare eroare (excepție).

SQL Statement: inserare

```
string stmt="DELETE FROM Users WHERE Nume='Popescu'";
SQLiteCommand cmd = new SQLiteCommand(stmt, _conn);
try{
cmd.ExecuteNonQuery();
}
catch (Exception ex){
MessageBox.Show("Unable to delete from table! ERROR:" +
ex.ToString());
return;
}
```

SQLite - citire înregistrări

- Construire comandă.
- Apel `ExecuteReader` - datele accesibile prin `SQLiteDataReader`.
- Tratare eroare (excepție).
- Metode/proprietăți `SQLiteDataReader`:
  - `Read()`: returnează `true` cât timp mai sunt înregistrări.
  - `FieldCount`: numărul de câmpuri.
  - `GetName(i)`: denumirea câmpului i.
  - `GetInt32(i)`: valoarea câmpului i pentru câmpuri de tipul INTEGER.
  - `GetString(i)`: valoarea câmpului i pentru câmpuri de tipul TEXT.
  - `Close()`: eliberează resursele alocate citirii - OBLIGATORIU trebuie apelat.
  - IMPORTANT! Alegerea apelului `GetInt32(i)` sau `GetString(i)` se face în cod în funcție de tipul coloanei.

SQLite - citire înregistrări

SQL Statement: citire înregistrări

```
string stmt="SELECT * FROM Users";
SQLiteCommand cmd = new SQLiteCommand(stmt, _conn);
SQLiteDataReader reader = null;
try{reader = cmd.ExecuteReader();}catch(...){...}
string data = "Fields:";
for (int i = 0; i < reader.FieldCount; ++i)
  data += reader.GetName(i) + "\n";
data += "DATA: \n";
while (reader.Read()){
  data += reader.GetInt32(0).ToString() + " ";
  data += reader.GetString(1) + " ";
  data += reader.GetString(2) + " ";
  data += reader.GetInt32(3).ToString() + " ";
  data += "\n";}
reader.Close();
```