Belluno Linux User Group

Il database tascabile!

Belluno, 27 ottobre 2007

Relatore: Ing. Mauro Barattin

Sommario

In questo talk parleremo di:

- Cos'è un database
- Caratteristiche di SQLite
- Quando utilizzare SQLite
- Quando non utilizzare SQLite
- Amministrazione tramite command line
- Tools di amministrazione con interfaccia grafica
- Programmazione con SQLite
- Uso di SQLite con OpenOffice
- Esempi applicativi
- Conclusioni

DB, DBMS e RDBMS

- Il termine Database (DB) indica un archivio di dati, riguardanti uno o più argomenti correlati tra loro, strutturato in modo tale da consentire la gestione dei dati stessi (l'inserimento, la ricerca, la cancellazione e l'aggiornamento) da parte di applicazioni software.
- Un Database Management System (DBMS) è un sistema software progettato per consentire la creazione e la manipolazione efficiente di database.
- Esistono diverse tipologie di database. In un Relational Database Management System (RDBMS) i dati sono organizzati in insiemi di "record" rappresentati come tabelle. Le relazioni tra le informazioni derivano dalla corrispondenza di alcuni campi di record appartenenti a tabelle diverse.

Caratteristiche di SQLite (1)

- La base di dati è un singolo file (max 2 Terabyte)
  - Formato indipendente dalla piattaforma hardware
  - Facilità di backup
- Facile da utilizzare
  - Semplice installazione (eseguibile sqlite3 o sqlite3.exe)
  - Nessuna configurazione
  - Semplice da amministrare
  - Non richiede privilegi di amministratore (root)
- Piccolo, veloce e affidabile
- E' multipiattaforma (Windows, Linux, *BSD,...)
  - Ports non ufficiali per WinCE, Nokia N800, PSP, ...)
- E' OpenSource (Public Domain, privo di qualsiasi licenza)

Caratteristiche di SQLite (2)

- E' una libreria C
  - Sviluppata a partire dal 2000 da Richard Hipp
  - Circa 250 Kb di libreria
- Fornisce le funzionalità di un DBMS relazionale
- Supporta Store Procedures, Triggers, Transazioni e Subquery
- Non implementa il paradigma client/server
- Non esiste il concetto di "account"
- Implementa il motore SQL (SQL-92)
  - Alcune caratteristiche non sono ancora implementate
    - ✔ Trigger con alcune limitazioni
    - ✔ Supporto ALTER TABLE non completo
    - ✔ No transazioni nidificate (solo una transazione attiva)
- Supporta la codifica dati internazionale UTF-8 e UTF-16

Quando usare SQLite?

- Quando non serve un database “enterprise”
  - Non è stato progettato per competere con Oracle...
- In applicazioni desktop che necessitano di una base dati
  - Semplicità di amministrazione e manutenzione
  - Semplice da utilizzare nei programmi
- Quando non serve la scalabilità multiutente
- In siti web a basso/medio traffico
  - Siti fino a 100000 hits/day (stima conservativa)
- Come alternativa ai DB “text based” (XML, Berkeley DB)
  - Supporto di SQL
- Come database temporaneo (“in-memory” database)
- In dispositivi Embedded (telefoni, PDA, appliances...)
- Per apprendere il linguaggio SQL

Quando non usare SQLite?

- In applicazioni client/server
  - Manca la gestione degli “account”
  - Scarse prestazioni di SQLite con network filesystem
- In siti web con molto traffico
  - Continui R/W sull'unico file
  - Database non scalabile
- Nel caso di grandi quantità di dati
  - SQLite usa circa 256 bytes di RAM for ogni Mbyte di dati
  - Limitazioni poste dal filesystem utilizzato (es. FAT)
- Con tante applicazioni concorrenti che accedono al database
  - SQLite usa dei lock reader/writer nell'intero database

Tipi di dati

- "La tipizzazione dei dati va al di fuori del compito di un database, che deve solo memorizzarli" (Richard Hipp)
- SQLIte 2.x era addirittura "typeless" (TEXT)
- SQLIte 3.x usa i seguenti tipi di dati
  - NULL
  - INTEGER (intero con segno da 1, 2, 3, 4, 6, o 8 bytes)
  - REAL (8-byte "IEEE floating point")
  - TEXT (stringa di testo UTF-8 o UTF-16)
  - BLOB ("blob" di dati, ad es. files binari)
- SQLite interpreta il tipo in base al valore del campo stesso e non in base al tipo dichiarato per il campo
- Per la compatibilità con gli altri motori di database, SQLite supporta il concetto di "type affinity" sui campi (tipo "consigliato" per i dati memorizzati in quel campo)

Amministrazione di SQLIte

- Command line
  - Inclusa nella distribuzione di SQLite
  - Simile alle command line di altri database (mysql, psql,...)
  - Amministrazione completa
  - Difficile da utilizzare
  - Integrabile in bash script come comando Unix
- Programmi con interfaccia grafica
  - SQLite Browser
  - SQLite Studio
  - SQLiteman
- Interfacce web-based
  - SQLite Manager

Il linguaggio SQL

- Nasce nel 1974 ad opera di Donald Chamberlin (IBM)
- Linguaggio per lavorare con database relazionali
- Inizialmente utilizzato da IBM per usi interni
- Nel tempo è diventato uno standard
- Ogni RDBMS moderno lo implementa
- Varie versioni (ISO) SQL/86, SQL/89, SQL/92 e SQL/2003
- Conoscerne la sintassi ci permette l'interrogazione di qualsiasi RDBMS che lo implementa

La command line (1)

mauro@jeeg:~

File Modifica Visualizza Terminale Schede Aiuto

```
mauro@jeeg:~$ sqlite3
SQLite version 3.3.13
Enter ".help" for instructions
sqlite> .help
.bail ON|OFF           Stop after hitting an error.  Default OFF
.databases             List names and files of attached databases
.dump ?TABLE? ...      Dump the database in an SQL 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
.import FILE TABLE     Import data from FILE into TABLE
.indices TABLE         Show names of all indices on TABLE
.mode MODE ?TABLE?     Set output mode where MODE is one of:
                         csv      Comma-separated values
                         column   Left-aligned columns.  (See .width)
                         html     HTML <table> code
                         insert   SQL insert statements for TABLE
                         line     One value per line
                         list     Values delimited by .separator string
                         tabs     Tab-separated values
                         tcl      TCL list elements
.nullvalue STRING      Print STRING in place of NULL values
.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 used by output mode and .import
.show                  Show the current values for various settings
.tables ?PATTERN?      List names of tables matching a LIKE pattern
.timeout MS            Try opening locked tables for MS milliseconds
.width NUM NUM ...     Set column widths for "column" mode
sqlite>
```

La command line (2)

```
$ sqlite3 sqlite.db
SQLite version 3.3.13
Enter ".help" for instructions
sqlite> CREATE TABLE Indirizzi (
   ...> ID              INTEGER PRIMARY KEY,
   ...> Nome            CHAR(20),
   ...> Cognome         CHAR(20),
   ...> Indirizzo       VARCHAR(200),
   ...> Telefono        VARCHAR(30));

sqlite> INSERT INTO Indirizzi VALUES (
   ...> 1,
   ...> 'Ayrton',
   ...> 'Senna',
   ...> 'Via Grand Prix, 1',
   ...> '367 123456');
```

La command line (3)

```
sqlite> .header on
sqlite> SELECT Cognome, Telefono FROM Indirizzi;
Cognome     Telefono
----------  ----------
Senna       367 123456
Schumacher  0437 98765
Del Piero   02 246810
Ghedina     0437 11223
Tomba       051 998877

sqlite> SELECT Cognome, Nome FROM Indirizzi ORDER BY Nome;
Cognome     Nome
----------  ----------
Tomba       Alberto
Del Piero   Alessandro
Senna       Ayrton
Ghedina     Christian
Schumacher  Micheal
```

Esempio di Bash script

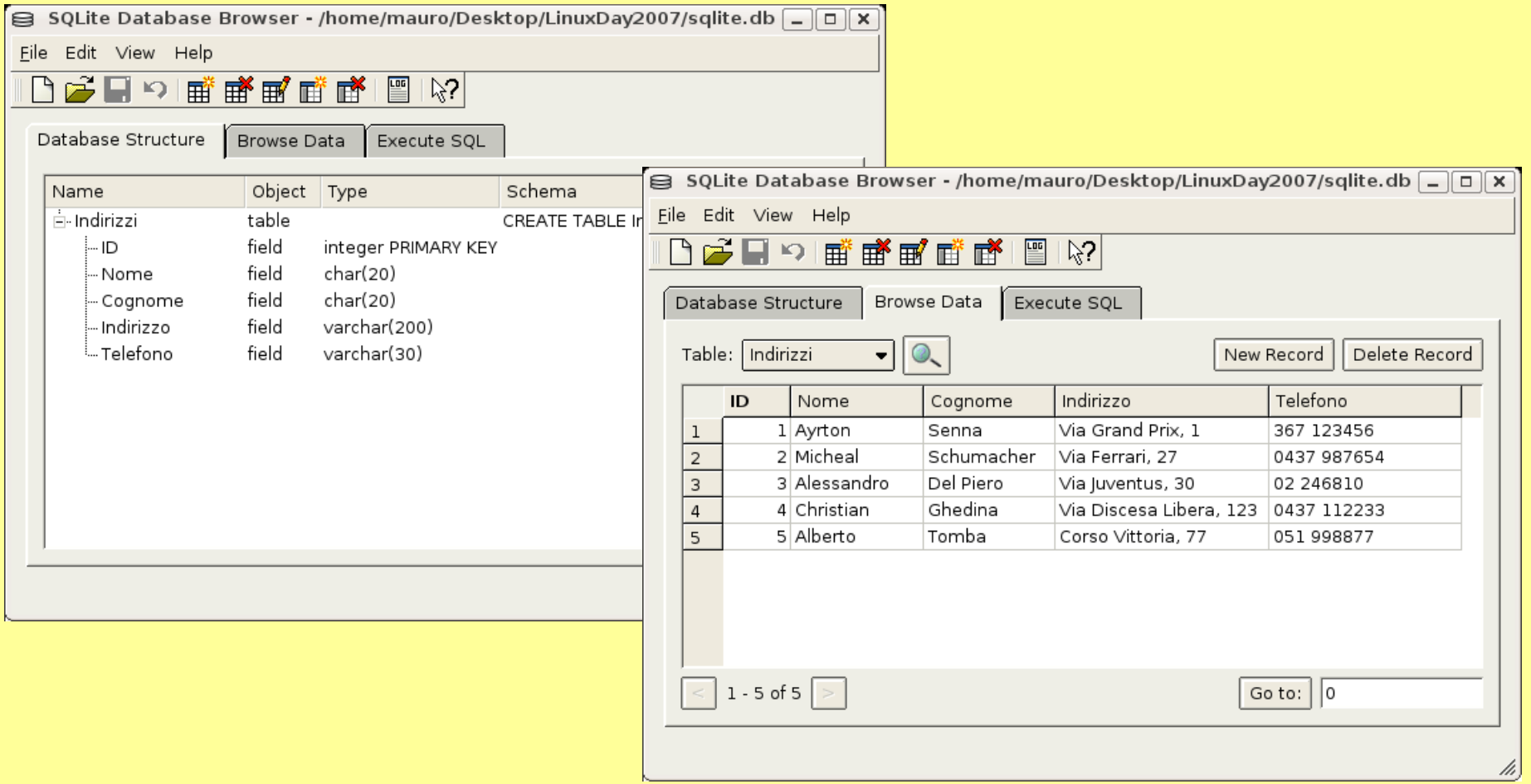
Segue un esempio di un Bash script:

```
#!/bin/sh
sqlite3 test.db <<EOF
CREATE TABLE Amici (ID INTEGER PRIMARY KEY, Nome CHAR(20));
INSERT INTO Amici VALUES (1, 'Pinco');
INSERT INTO Amici VALUES (2, 'Tizio');
INSERT INTO Amici VALUES (3, 'Caio');
EOF

$ sqlite3 test.db "SELECT * FROM Amici"
1|Pinco
2|Tizio
3|Caio
```

Programmi con GUI

Schermate di SQLite Browser:



Programmazione con SQLite

- E' una libreria C
  - Insieme di files .c e .h ANSI C
  - Utilizzabile su qualsiasi OS dotato di compilatore ANSI C
  - Non dipende da librerie esterne
  - Richieste hardware minime
  - Disponibile anche come unico file ANSI C "amalgamation"
    - ✔ Disponibile dalla versione 3.3.14
    - ✔ Facile integrazione in altri progetti
    - ✔ Maggiore velocità

Wrapper per SQLite

- Bindings TCL incluso nella distribuzione ufficiale
- Bindings per altri linguaggi disponibili separatamente
  - C/C++
  - Delphy
  - Fortran
  - Java (tramite driver JDBC)
  - Javascript
  - .NET Framework (incluso Mono)
  - Perl
  - PHP (di default a partire da PHP5)
  - Python (di default a partire da Python 2.5)
  - Ruby

... e molti altri!

SQLite & C

```
#include <stdio.h>
#include <sqlite3.h>

static int callback(void *NotUsed, int argc, char **argv, char **azColName){
  int i;
  for(i=0; i<argc; i++){
    printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
  }
  return 0;
}

int main(int argc, char **argv){
  sqlite3 *db;
  char *zErrMsg = 0;
  int rc;

  rc = sqlite3_open("test.db", &db);
  rc = sqlite3_exec(db, "SELECT * FROM Amici", callback, 0, &zErrMsg);
  sqlite3_close(db);
  return 0;
}
```

```
$ gcc sqlite.c -lsqlite3
$ ./a.out
ID = 1
Nome = Pinco
ID = 2
Nome = Tizio
ID = 3
Nome = Caio
```

SQLite & Python

```
from pysqlite2 import dbapi2 as sqlite
import os

def WriteDB(db):
    c = db.cursor()
    c.executescript("""CREATE TABLE Amici (ID INTEGER PRIMARY KEY, Nome CHAR(20));
                       INSERT INTO Amici VALUES (1, 'Pinco');
                       INSERT INTO Amici VALUES (2, 'Tizio');
                       INSERT INTO Amici VALUES (3, 'Caio'); """)
    c.close()

def ReadDB(db):
    c = db.cursor()
    c.execute("select * from Amici;")
    for record in c.fetchall():
        print record
    c.close()

if __name__ == "__main__":
    db = sqlite.connect("test.db", timeout = 10.0)
    WriteDB(db)
    ReadDB(db)
    db.close()
```

```
$ python sqlite.py
(1, u'Pinco')
(2, u'Tizio')
(3, u'Caio')
```

SQLite & Mono

```
using System;
using System.Data;
using Mono.Data.SqliteClient;

public class Test
{
    public static void Main(string[] args)
    {
      IDbConnection dbcon;
      dbcon = (IDbConnection) new SqliteConnection("URI=file:test.db,version=3");
      dbcon.Open();
      IDbCommand dbcmd = dbcon.CreateCommand();
      string sql = "SELECT * FROM Amici";
      dbcmd.CommandText = sql;
      IDataReader reader = dbcmd.ExecuteReader();
      while(reader.Read()) {
          string Nome = reader.GetString (1);
          Console.WriteLine(Nome);
      }
      reader.Close(); reader = null;  dbcmd.Dispose(); dbcmd = null; dbcon.Close();  dbcon = null;
    }
}
```

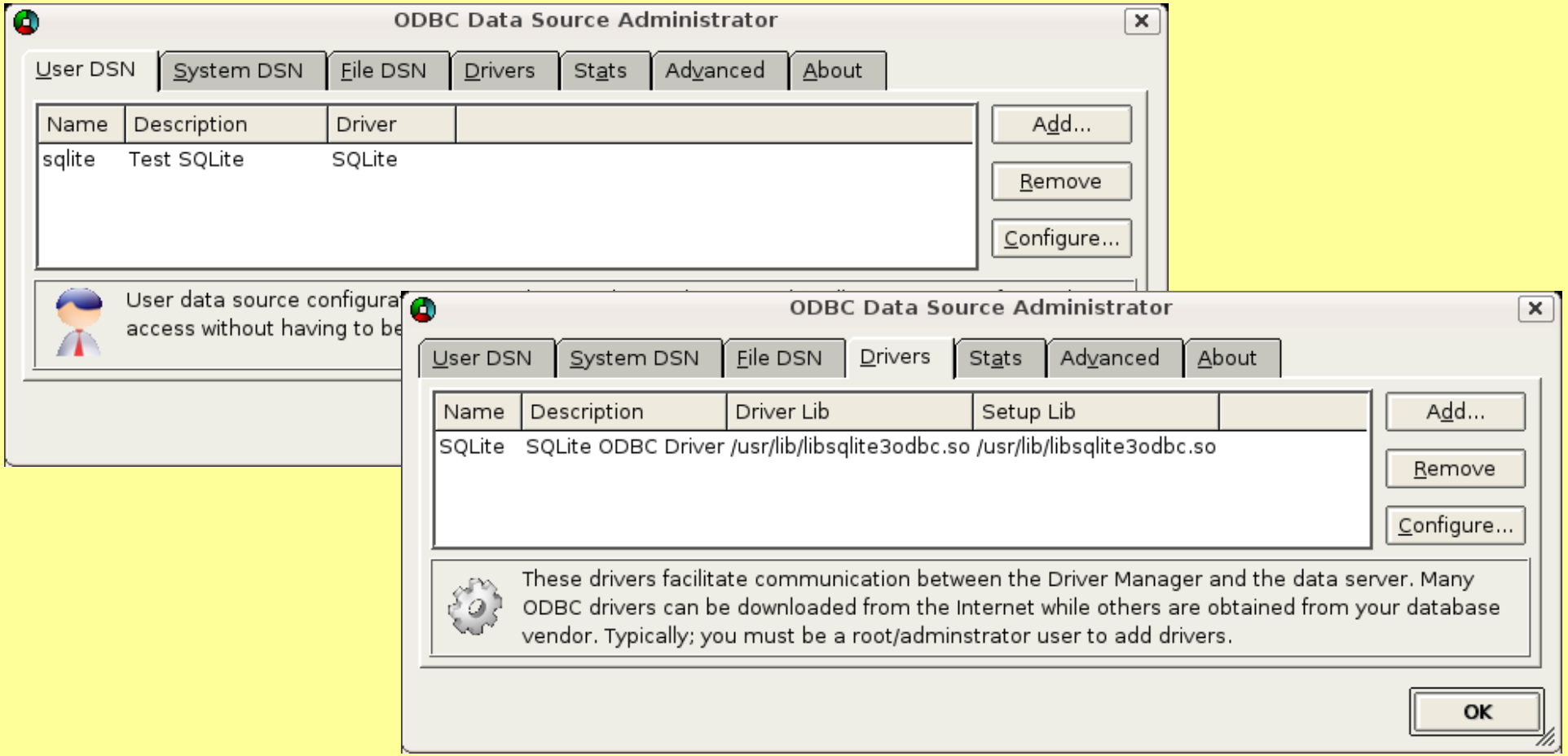
```
$ mcs sqlite.cs -r:System.Data.dll \
                -r:Mono.Data.SqliteClient
$ mono sqlite.exe
Pinco
Tizio
Caio
```

ODBC (1)

- Acronimo di “Open DataBase Connectivity”
- Interfaccia standard per la connessione ai DBMS
- Le applicazioni comunicano (sempre allo stesso modo) con il servizio ODBC, il quale comunica con i DBMS sottostanti, preoccupandosi di adattarsi alle loro particolarità.
- unixODBC + SQLite ODBC Driver
- Configurazione della sorgente dati ODBC
  - Driver ODBC per SQLite (/usr/lib/libsqlite3odbc.so)
    - ✔ Occorre essere root
    - ✔ Il file di configurazione è /etc/odbcinst.ini
  - Sorgente dati (User DSN, Database Source Name)
    - ✔ Qualunque utente può creare una sorgente dati
    - ✔ Il file di configurazione è ~/.odbc.ini
  - Interfaccia grafica ODBCConfig

ODBC (2)

Setup visuale della sorgente dati con ODBCConfig:

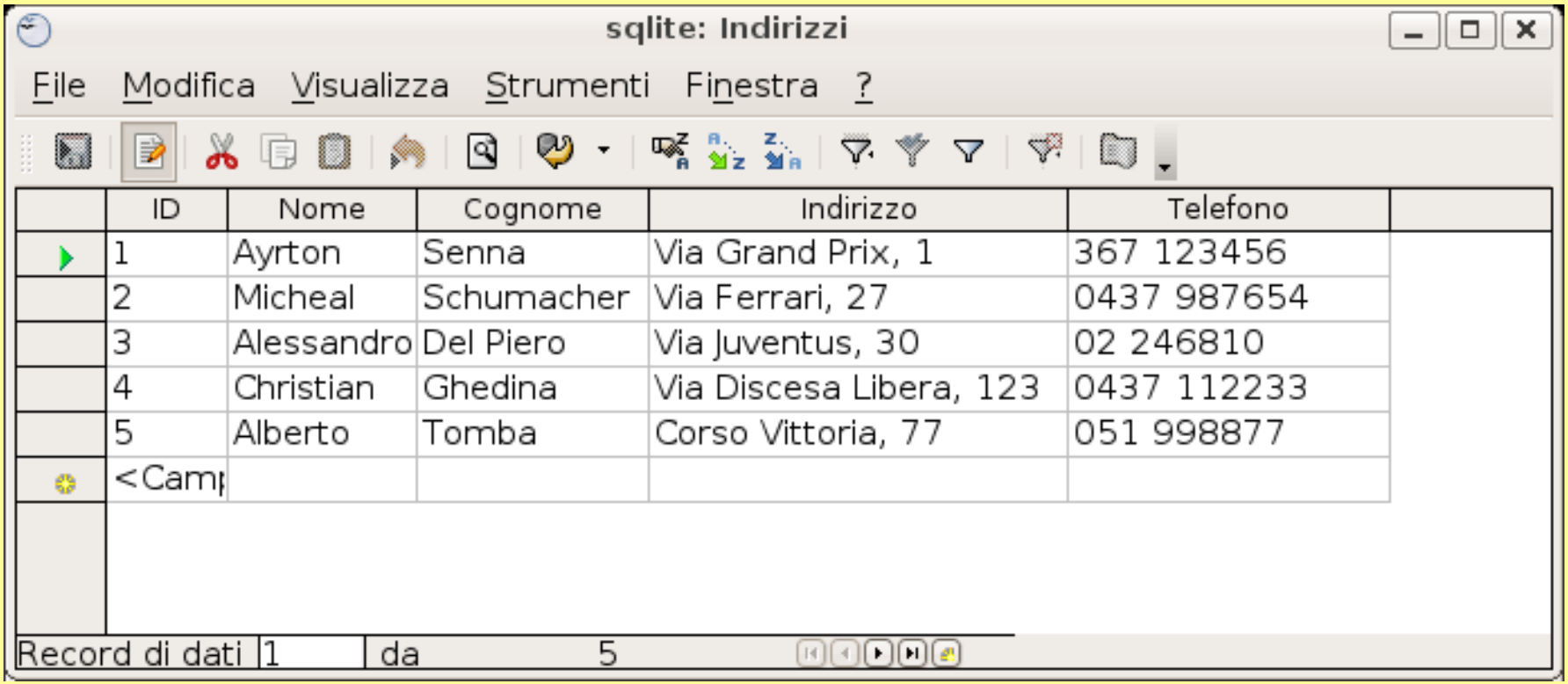


OpenOffice (1)

- OpenOffice Base: pproccio “user friendly” ai database
  - Interfaccia grafica sofisticata, semplice ed intuitiva
  - Gestione tabelle
  - Ricerche (query)
  - Formulari ovvero inserimento dati (forms)
  - Rapporti
- OpenOffice 2.x ha un suo motore di database (HSQLDB)
- In sviluppo un driver SDBC (nativo di OpenOffice) per SQLite
- Ampia disponibilità di drivers ODBC
- E' necessario prima configurare la sorgente dati ODBC
- Il database iniziale deve essere creato con un altro tool, ad esempio con la command line.
- I dati sono accessibili anche da Writer e Calc

OpenOffice (2)

Visualizzazione di una tabella:



Chi usa SQLite?

Software basati su SQLite

- AmaROK (amarok.kde.org)
- Bacula (www.bacula.org)
- Bogofilter (http://bogofilter.sourceforge.net)
- Dovecot (http://dovecot.org)
- Roundup (http://roundup.sf.net)

...e molti altri!

Conclusioni

- Ottimo strumento per una vasta gamma di applicazioni
- Valida alternativa ad altri DBMS (es. MS Access)
- Semplicità di utilizzo
- Disponibilità di wrapper per i principali linguaggi
- Facile collegamento ad OpenOffice
- Utilizzato in numerosi progetti
- Multipiattaforma
- Non ci sono licenze che ne restringano l'utilizzo

Bibliografia

- SQLite (http://www.sqlite.org)
- Wikipedia (http://en.wikipedia.org/wiki/SQLite)
- A. Carichini - “SQLIte: il database tascabile”
- M. Scabarrà - “SQLlite, la risposta Open Source ad Access”
- F. Marchesi, G. Tufano, A. Babini - “SQLite”
- PySQLite (http://www.initd.org/tracker/pysqlite/wiki/pysqlite)
- SQLite ODBC Driver (http://www.ch-werner.de/sqliteodbc/)
- unixODBC (http://www.unixodbc.org)
- SQLiteJDBC (http://www.zentus.com/sqlitejdbc/)
- SQLite Browser (sqlitebrowser.sourceforge.net)