

Uniwersytet Morski w Gdyni

przedmiot:

## Narzędzia Informatyczne

### Ćw. 10 Bazy danych na przykładzie SQLite

#### 1. Cel ćwiczenia

Celem ćwiczenia jest przedstawienie podstaw relacyjnych baz danych i zasad obsługi Systemu Zarządzania Bazą Danych. Do ćwiczeń została wykorzystana baza SQLite.

#### 2. Wprowadzenie

Istnieje wiele sposobów przechowywania informacji cyfrowych. Bardzo popularną metodą jest umieszczanie danych w Bazach Danych. Z bazami danych wiąże się następujące definicje:

- Baza Danych – to uporządkowany logicznie zbiór danych o strukturze przyjętej z modelu danych.
- SZBZ (ang. DBMS) – System Zarządzania Bazą Danych (ang. Data Base Management System) to zbiór programów, narzędzi oraz bibliotek, które umożliwiają użytkownikowi interakcję z bazą danych.
- SQL – to strukturalny język zapytań używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych.

##### 2.1 SQLite

SQLite jest biblioteką wewnątrzprocesową, która implementuje niezależny, bezserwerowy, zero-konfiguracyjny, transakcyjny silnik bazy danych SQL. Jest to baza danych, której nie trzeba konfigurować w systemie.

SQLite jest świetnym programem pozwalającym na nauczenie się podstaw tworzenia baz danych oraz opanowanie składni języka SQL. Ogromną zaletą SQLite jest jego prostota – w przeciwieństwie do innych, bardziej popularnych, SZBZ tj. MySQL lub PostgreSQL nie wymaga on uruchomienia serwera i postawienia specjalnej usługi by działać. SQLite przechowuje bazy danych jako pliki i pozwala na prace poprzez konsolę lub za pomocą graficznego interfejsu korzystającego z jego API. Rozbudowane API pozwala także na używanie funkcjonalności SQLite w własnoręcznie napisanych programach (m.in. w C++) – umożliwia to na wprowadzenie do własnego programu obsługi i możliwości pracy na bazach danych.

Warto zapoznać się z dokumentacją projektu na stronie: <https://www.sqlite.org/docs.html>

Szczególnie warto przyjrzeć się typom danych na jakich pracuje SQLite, ponieważ typy danych uproszczono względem innych SZBZ. Można tego dokonać za pomocą tego linku : <https://www.sqlite.org/datatype3.html>

Warto także zapoznać się z podstawami języka SQL, lecz nie jest to wymagane do wykonania tego ćwiczenia laboratoryjnego. Godnym polecenia kursem jest kurs wideo ze strony:

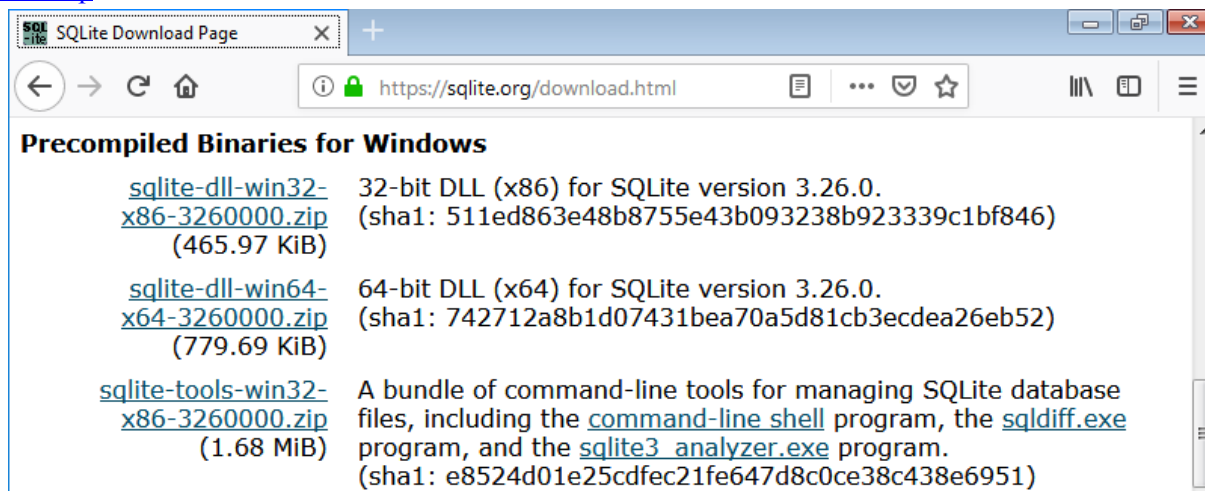
<https://pl.khanacademy.org/computing/computer-programming/sql/sql-basics>

#### 3. Przykłady

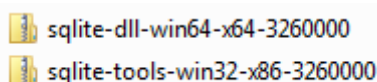
Pracę z programem SQLite należy zacząć od pobrania go i dokonania instalacji. Aby pobrać program należy przejść na stronę: <https://www.sqlite.org/download.html> , pobrać pliki [sqlite-dll-win64-x64-3260000.zip](https://www.sqlite.org/download.html) lub [sqlite-dll-win32-x86-3260000.zip](https://www.sqlite.org/download.html) (w zależności od architektury procesora) oraz [sqlite-tools-win32-x86-](https://www.sqlite.org/download.html)

*Projekt „SezAM wiedzy, kompetencji i umiejętności” jest współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego w ramach Programu Operacyjnego Wiedza Edukacja Rozwój*

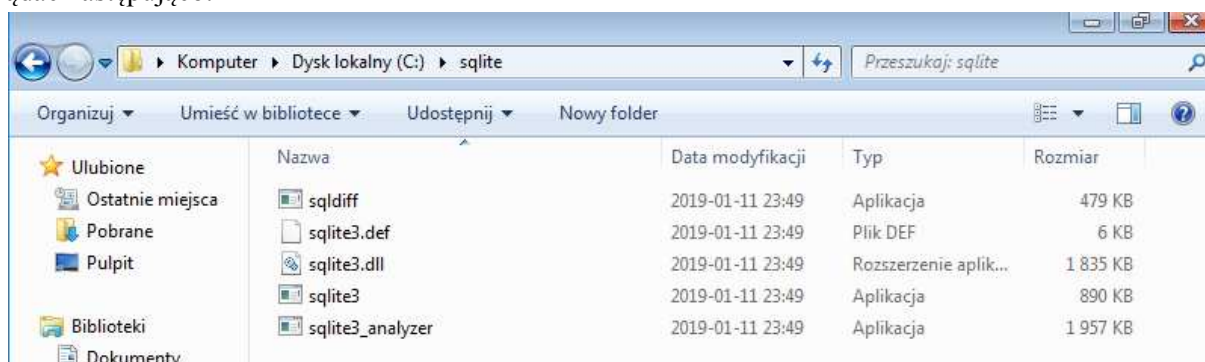
[3260000.zip](#)



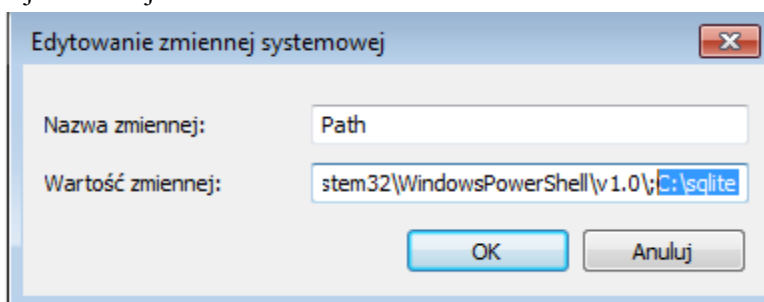
Powinniśmy otrzymać dwa archiwa.



Należy utworzyć folder `C:\sqlite` i wypakować do niego zawartość archiwów. Zawartość folderu *sqlite* powinna wyglądać następująco:



Warto dodać ten folder do zmiennej środowiskowej Path, aby móc posługiwać się zawartymi w nim programami jako polecenia z dowolnej lokalizacji.



Następnie należy wywołać z konsoli program *sqlite3*, aby sprawdzić czy poprawnie dokonano instalacji.



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



```
C:\Windows\system32\cmd.exe - sqlite3
Microsoft Windows [Wersja 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Win7UM1>sqlite3
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Za pomocą polecenia `.help` wyświetlimy wszystkie dostępne funkcje tego programu.



```
C:\Windows\system32\cmd.exe - sqlite3
sqlite> .help
.archive ...          Manage SQL archives
.auth ON|OFF          Show authorizer callbacks
.backup ?DB? FILE     Backup DB (default "main") to FILE
.bail on|off          Stop after hitting an error. Default OFF
.binary on|off        Turn binary output on or off. Default OFF
.cd DIRECTORY         Change the working directory to DIRECTORY
.changes on|off       Show number of rows changed by SQL
.check GLOB           Fail if output since .testcase does not match
.clone NEWDB          Clone data into NEWDB from the existing database
.databases            List names and files of attached databases
.dbconfig ?op? ?val? List or change sqlite3_db_config() options
.dbinfo ?DB?          Show status information about the database
.dump ?TABLE? ...     Render all database content as SQL
.echo on|off          Turn command echo on or off
.egg on|off|full       Enable or disable automatic EXPLAIN QUERY PLAN
.excel                Display the output of next command in a spreadsheet
.exit ?CODE?          Exit this program with return-code CODE
.expert              EXPERIMENTAL. Suggest indexes for specified queries
.fullschema ?--indent? Show schema and the content of sqlite_stat tables
.headers on|off       Turn display of headers on or off
.help ?-all? ?PATTERN? Show help text for PATTERN
.import FILE TABLE   Import data from FILE into TABLE
.imposter INDEX TABLE Create imposter table TABLE on index INDEX
.indexes ?TABLE?      Show names of indexes
.limit ?LIMIT? ?VAL? Display or change the value of an SQLITE_LIMIT
.lint OPTIONS         Report potential schema issues.
.load FILE ?ENTRY?    Load an extension library
.log FILE!off         Turn logging on or off. FILE can be stderr/stdout
.mode MODE ?TABLE?    Set output mode
.nullvalue STRING     Use STRING in place of NULL values
.once (-e|-x|FILE)    Output for the next SQL command only to FILE
.open ?OPTIONS? ?FILE? Close existing database and reopen FILE
.output ?FILE?        Send output to FILE or stdout if FILE is omitted
.print STRING...      Print literal STRING
.prompt MAIN CONTINUE Replace the standard prompts
.quit                Exit this program
.read FILE            Read input from FILE
.restore ?DB? FILE     Restore content of DB (default "main") from FILE
.save FILE            Write in-memory database into FILE
.scanstats on|off     Turn sqlite3_stmt_scanstatus() metrics on or off
.schema ?PATTERN?     Show the CREATE statements matching PATTERN
.selftest ?OPTIONS?   Run tests defined in the SELFTEST table
.separator COL ?ROW?  Change the column and row separators
.sha3sum ...          Compute a SHA3 hash of database content
.shell CMD ARGS...    Run CMD ARGS... in a system shell
.show                Show the current values for various settings
.stats ?on|off?       Show stats or turn stats on or off
.system CMD ARGS...   Run CMD ARGS... in a system shell
.tables ?TABLE?       List names of tables matching LIKE pattern TABLE
.testcase NAME        Begin redirecting output to 'testcase-out.txt'
.timeout MS           Try opening locked tables for MS milliseconds
.timer on|off         Turn SQL timer on or off
.trace FILE!off       Output each SQL statement as it is run
.vfsinfo ?AUX?        Information about the top-level VFS
.vfslist              List all available VFSes
.vfsname ?AUX?        Print the name of the VFS stack
.width NUM1 NUM2 ...  Set column widths for "column" mode
```

Skoro zapoznaliśmy się już z funkcjonalnością programu wywołajmy polecenie `.databases` aby wyświetlić bazy zarządzane aktualnie przez program (w tym przypadku nie istnieje żadna baza), a następnie utworzymy bazę `nowa_baza` za pomocą polecenia `.open nowa_baza.db` (otwarcie nieistniejącej bazy powoduje jej utworzenie).

```
sqlite> .databases
main:
sqlite> .open nowa_baza.db
sqlite> .databases
main: C:\Users\Win7UM1\nowa_baza.db
sqlite>
```

Stwórzmy przykładową tabelę



```
sqlite> CREATE TABLE tabela<
...> kolumna1 INTEGER PRIMARY KEY NOT NULL,
...> kolumna2 REAL,
...> kolumna3 TEXT
...> );
sqlite>
```

Wywołajmy polecenie `.tables`, aby sprawdzić czy tabela została utworzona poprawnie.

```
sqlite> .tables
tabela
sqlite>
```

Za pomocą polecenia `.schema` możemy odczytać polecenie tworzące jej strukturę. Wykonajmy więc polecenie `.schema tabela`

```
sqlite> .schema tabela
CREATE TABLE tabela<
kolumna1 INTEGER PRIMARY KEY NOT NULL,
kolumna2 REAL,
kolumna3 TEXT
>;
sqlite>
```

Skoro dokonaliśmy zmian w naszej bazie danych, warto wykonać jej kopię zapasową. Aby tego dokonać opuścimy program SQLite poleceniem `.exit`, a następnie wywołajmy polecenie `sqlite3 nowa_baza.db .dump > backup.sql`

```
sqlite> .exit
C:\Users\Win7UM1>sqlite3 nowa_baza.db .dump > backup.sql
C:\Users\Win7UM1>
```

Utworzony został plik `backup.sql`.

```
C:\Users\Win7UM1>dir
Wolumin w stacji C nie ma etykiety.
Numer seryjny woluminu: 8CDA-988F

Katalog: C:\Users\Win7UM1

2019-01-12  00:30    <DIR>          .
2019-01-12  00:30    <DIR>          ..
2019-01-12  00:30             152 backup.sql ←
2018-11-29  18:57    <DIR>          Contacts
2018-11-29  18:57    <DIR>          Desktop
2018-11-29  18:57    <DIR>          Documents
2019-01-11  23:50    <DIR>          Downloads
2018-11-29  18:57    <DIR>          Favorites
2018-11-29  18:57    <DIR>          Links
2018-11-29  18:57    <DIR>          Music
2019-01-12  00:24      8 192 nowa_baza.db
2018-11-29  18:57    <DIR>          Pictures
2018-11-29  18:57    <DIR>          Saved Games
2018-11-29  18:57    <DIR>          Searches
2018-11-29  18:57    <DIR>          Videos
                2 plik(ów)                8 344 bajtów
                13 katalog(ów)  22 499 766 272 bajtów wolnych

C:\Users\Win7UM1>
```

Otwórzmy ten plik za pomocą edytora tekstu, aby wyświetlić jego zawartość.



```

backup.sql — Notatnik
Plik Edycja Format Widok Pomoc
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE tabela(
  kolumna1 INTEGER PRIMARY KEY NOT NULL,
  kolumna2 REAL,
  kolumna3 TEXT
);
COMMIT;

C:\Users\Win7UM1>notepad backup.sql
C:\Users\Win7UM1>

```

Jak widzimy plik zawiera strukturę całej zawartości bazy danych (w tym przypadku jest to tylko i wyłącznie tabela o nazwie *tabela*).

Skoro posiadamy kopię naszej bazy danych, możemy zacząć dokonywać na niej operacji. Wpierw jednak usuńmy naszą tabelę oraz bazę danych a następnie przywróćmy ją z kopii zapasowej.

Aby usunąć tabelę otworzymy naszą bazę danych i posłużymy się poleceniem SQL: *DROP TABLE tabela*;

```

C:\Users\Win7UM1>sqlite3
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open nowa_baza.db
sqlite> .tables
tabela
sqlite> DROP TABLE tabela;
sqlite> .tables
sqlite>

```

Aby usunąć całą bazę danych opuścimy poleceniem *.exit* program *sqlite3* i wykonajmy polecenie *del /f nowa\_baza.db* (alternatywnie można usunąć plik bazy danych za pomocą *Ekploratora plików*).

```

sqlite> .exit
C:\Users\Win7UM1>del /f nowa_baza.db
C:\Users\Win7UM1>

```

Dokonajmy teraz przywrócenia naszej bazy danych wywołując polecenie *sqlite3 nowa\_baza.db < backup.sql*, a następnie sprawdzimy czy znajduje się tam utworzona przez nas wcześniej tabela.





```
C:\Users\Win7UM1>sqlite3 nowa_baza.db < backup.sql

C:\Users\Win7UM1>sqlite3
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open nowa_baza.db
sqlite> .tables
tabela
sqlite>
```

Program SQLite umożliwia także wykonanie kopii zapasowej wybranej tabeli zamiast całej bazy danych. Aby tego dokonać należy otworzyć bazę danych, ustawić za pomocą polecenie `.output` wyjście na plik w którym ma znajdować się kopia zapasowa tabeli (w tym przypadku utworzony zostanie plik `tabela.sql` w lokacji `C:\Users\Win7UM1`), a następnie za pomocą polecenia `.dump` z nazwą tabeli jako argumentem rzucić ją do pliku kopii zapasowej.

```
C:\Users\Win7UM1>sqlite3
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open nowa_baza.db
sqlite> .tables
tabela
sqlite> .output C:/Users/Win7UM1/tabela.sql
sqlite> .dump tabela
sqlite> .exit

C:\Users\Win7UM1>dir
Wolumin w stacji C nie ma etykiety.
Numer seryjny woluminu: 8CDA-988F

Katalog: C:\Users\Win7UM1

2019-01-12 00:55 <DIR> .
2019-01-12 00:55 <DIR> ..
2019-01-12 00:30 152 backup.sql
2018-11-29 18:57 <DIR> Contacts
2018-11-29 18:57 <DIR> Desktop
2018-11-29 18:57 <DIR> Documents
2019-01-11 23:50 <DIR> Downloads
2018-11-29 18:57 <DIR> Favorites
2018-11-29 18:57 <DIR> Links
2018-11-29 18:57 <DIR> Music
2019-01-12 00:44 8 192 nowa_baza.db
2018-11-29 18:57 <DIR> Pictures
2018-11-29 18:57 <DIR> Saved Games
2018-11-29 18:57 <DIR> Searches
2019-01-12 00:56 144 tabela.sql
2018-11-29 18:57 <DIR> Videos
4 plik(ów) 8 640 bajtów
13 katalog(ów) 22 500 044 800 bajtów wolnych

C:\Users\Win7UM1>
```

Warto otworzyć plik kopii zapasowej tabeli edytorem tekstu, aby zobaczyć jego strukturę.







```
studenci.sql — Notatnik
Plik Edycja Format Widok Pomoc
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE studenci(
  nr_albumu INTEGER PRIMARY KEY NOT NULL,
  imie TEXT NOT NULL,
  nazwisko TEXT NOT NULL,
  plec TEXT NOT NULL,
  kierunek TEXT,
  rok INTEGER
);
INSERT INTO studenci(nr_albumu,imie,nazwisko,plec,kierunek,rok)
VALUES (11111, 'Jan', 'Kowalski', 'Meczczyzna', 'Informatyka', 1);
INSERT INTO studenci(nr_albumu,imie,nazwisko,plec)
VALUES (11112, 'Jan', 'Nowak', 'Meczczyzna');
COMMIT;
```

Edytor tekstu wyświetla nam strukturę tabeli oraz dwie dane w niej zawarte. Kwerendy *INSERT INTO ... VALUES ...* służą do umieszczania danych w tabelach. W tym przypadku tabela posiada 6 kolumn, z czego wypełnienie 3 z nich przy tworzeniu nowego rekordu jest obowiązkowe (parametr *NOT NULL*), pozostałe 3 są opcjonalne. Tabela ta zawiera dwóch studentów: Jana Kowalskiego, w którego przypadku zdefiniowano wartość wszystkich 6 pól rekordu oraz Jana Nowaka, w którego przypadku zdefiniowano wartość tylko wymaganych 3 pól rekordu.

Dokonajmy importu zawartości pliku *studenci.sql* do naszej bazy danych.

```
C:\Users\Win7UM1>sqlite3 nowa_baza.db < studenci.sql

C:\Users\Win7UM1>sqlite3
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open nowa_baza.db
sqlite> .tables
studenci  tabela
sqlite> .schema studenci
CREATE TABLE studenci(
  nr_albumu INTEGER PRIMARY KEY NOT NULL,
  imie TEXT NOT NULL,
  nazwisko TEXT NOT NULL,
  plec TEXT NOT NULL,
  kierunek TEXT,
  rok INTEGER
);
sqlite> SELECT * FROM studenci;
11111|Jan|Kowalski|Meczczyzna|Informatyka|1
11112|Jan|Nowak|Meczczyzna|

sqlite> .header on
sqlite> .mode column
sqlite> SELECT * FROM studenci;
nr_albumu  imie      nazwisko   plec      kierunek   rok
-----
11111      Jan       Kowalski   Meczczyzna Informatyka 1
11112      Jan       Nowak      Meczczyzna
sqlite>
```

Dokonaliśmy importu tabeli do naszej bazy danych. Następnie za pomocą polecenia *.tables* sprawdziliśmy czy tabela została poprawnie umieszczona w naszej bazie. Kolejnym krokiem było wyświetlenie struktury tabeli poleceniem *.schema*. Następnie za pomocą kwerendy *SELECT* wyświetliśmy zawartość tabeli. Jako, że konfiguracja widoku zapytania *SELECT* programu *sqlite3* nie wyświetlała danych w sposób wystarczająco czytelny wykonaliśmy polecenie *.header on*, aby wyświetlać nazwy kolumn oraz *.mode column*, aby odzielić od siebie wartości pól poszczególnych kolumn tablicami. Następnie ponownie wywołano kwerendę *SELECT \* FROM studenci;*

Zawartość tabeli możemy uzupełniać o kolejnych studentów. Dane umieszczone możemy także edytować oraz usuwać.

Aby dodać nowego studenta posłużymy się kwerendą *INSERT* (dla ułatwienia kwerendę można skopiować z pliku *studenci.sql* za pomocą edytora tekstu, następnie wkleić do konsoli i zmienić dane).

```
sqlite> INSERT INTO studenci (nr_albumu, imie, nazwisko, plec, kierunek, rok) VALUES
(22222, 'Jacek', 'Kowal', 'Mezczyzna', 'Informatyka', 2)
...> ;
sqlite> SELECT * FROM studenci;
nr_albumu  imie      nazwisko   plec      kierunek   rok
-----
11111      Jan       Kowalski   Mezczyzna Informatyka 1
11112      Jan       Nowak      Mezczyzna Informatyka 1
22222      Jacek     Kowal      Mezczyzna Informatyka 2
sqlite>
```

Stosując kwerendę *SELECT* nie musimy wyświetlać wszystkich danych. Aby wyświetlić wszystkie dane stosujemy *\** (gwiazdkę), lecz możemy podać w tym miejscu tylko wybrane kolumny.

```
sqlite> SELECT imie, nazwisko FROM studenci;
imie      nazwisko
-----
Jan       Kowalski
Jan       Nowak
Jacek     Kowal
sqlite>
```

Jeśli spróbujemy wykonać kwerendę *INSERT* nie podając wartości dla pól obowiązkowych (ich kolumna zawiera parametr *NOT NULL*), wyświetlony zostanie błąd informujący nas, że polecenie jest niepoprawne.

```
sqlite> INSERT INTO studenci (nr_albumu, nazwisko) VALUES (1234, 'Kowalllll');
Error: NOT NULL constraint failed: studenci.imie
sqlite>
```

Aby zmienić wartość pola wybranego rekordu (wiersza) musimy posłużyć się kwerendą *UPDATE*. Kwerenda *UPDATE* wymaga jednak, aby podczas wykonywania wskazać jej rekord na którym ma dokonać zmiany – dokonuje się tego podając unikatową wartość, różną dla każdego rekordu (w tym przypadku jest to *nr\_albumu*). Należy więc wywołać kwerendę *UPDATE* zmieniającą wartość dowolnego pola rekordu, ale trzeba to zrobić wraz z poleceniem agregującym *WHERE* wskazującym na numer albumu konkretnego studenta.

```
sqlite> SELECT * FROM studenci
...> ;
nr_albumu  imie      nazwisko   plec      kierunek   rok
-----
11111      Jan       Kowalski   Mezczyzna Informatyka 1
11112      Jan       Nowak      Mezczyzna Informatyka 1
22222      Jacek     Kowal      Mezczyzna Informatyka 2
sqlite> UPDATE studenci SET rok = 1 WHERE nr_albumu == 22222;
sqlite> SELECT * FROM studenci;
nr_albumu  imie      nazwisko   plec      kierunek   rok
-----
11111      Jan       Kowalski   Mezczyzna Informatyka 1
11112      Jan       Nowak      Mezczyzna Informatyka 1
22222      Jacek     Kowal      Mezczyzna Informatyka 1
sqlite>
```

W tym przypadku dokonaliśmy zmiany roku Jacka Kowala, a jego rekord wskazaliśmy za pomocą jego *nr\_albumu*.

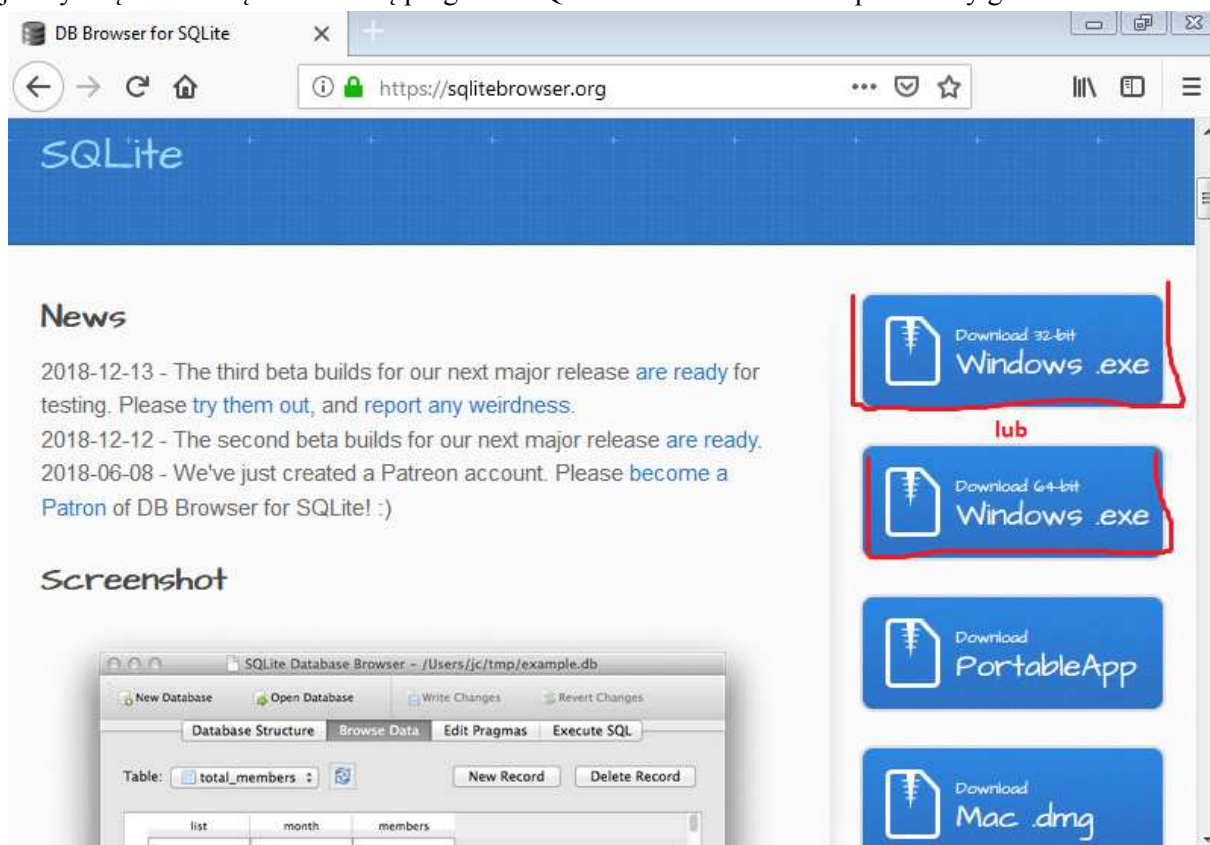
Możemy także usunąć rekord z bazy danych. Dokonamy tego za pomocą kwerendy *DELETE*. Analogicznie do kwerendy *UPDATE*, kwerenda ta wymaga wskazania jej rekordu na którym ma dokonać operacji.

```
sqlite> SELECT * FROM studenci;
nr_albumu  imie      nazwisko  plec      kierunek  rok
-----
11111      Jan       Kowalski  Mężczyzna Informatyka 1
11112      Jan       Nowak     Mężczyzna
22222      Jacek     Kowal     Mężczyzna Informatyka 1
sqlite> DELETE FROM studenci WHERE nr_albumu == 22222;
sqlite> SELECT * FROM studenci;
nr_albumu  imie      nazwisko  plec      kierunek  rok
-----
11111      Jan       Kowalski  Mężczyzna Informatyka 1
11112      Jan       Nowak     Mężczyzna
sqlite>
```

W tym przypadku usunięto Jacka Kowala, a jego rekord wskazano za pomocą jego *nr\_albumu*.

Dokonywanie operacji na danych z poziomu konsoli nie jest optymalne. Konstruowanie ręcznie kwerend jest czasochłonne. Dużo wydajniejszą metodą edycji danych jest robienie tego za pomocą interfejsów graficznych. Wadą interfejsów graficznych jest jednak to, że nie rzadko kiedy implementują pełną funkcjonalność programu, na którym API działają. Zazwyczaj jednak oferują najczęściej wykorzystywaną funkcjonalność. Przykładem godnego polecenia interfejsu graficznego SQLite jest *SQLite Database Browser*, dostępnego do pobrania pod adresem: <https://sqlitebrowser.org/>

Przejdźmy więc na stronę internetową programu SQLite Database Browser i pobierzmy go.



Dokonajmy standardowej instalacji (według zaleceń instalatora).

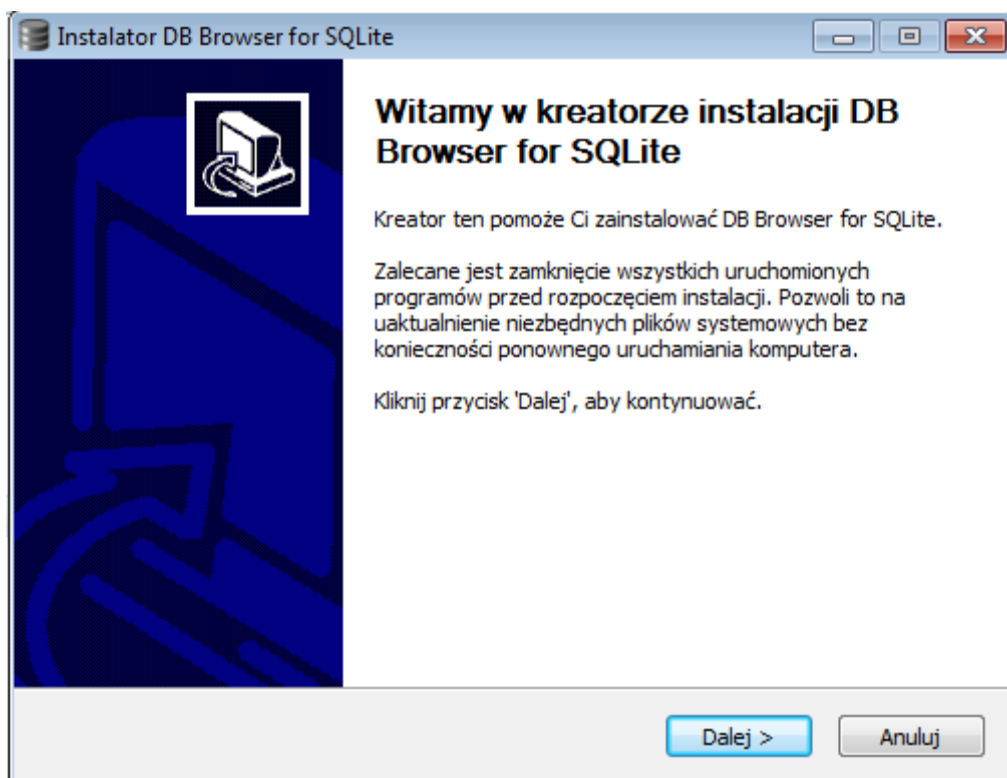


Fundusze Europejskie  
Wiedza Edukacja Rozwój

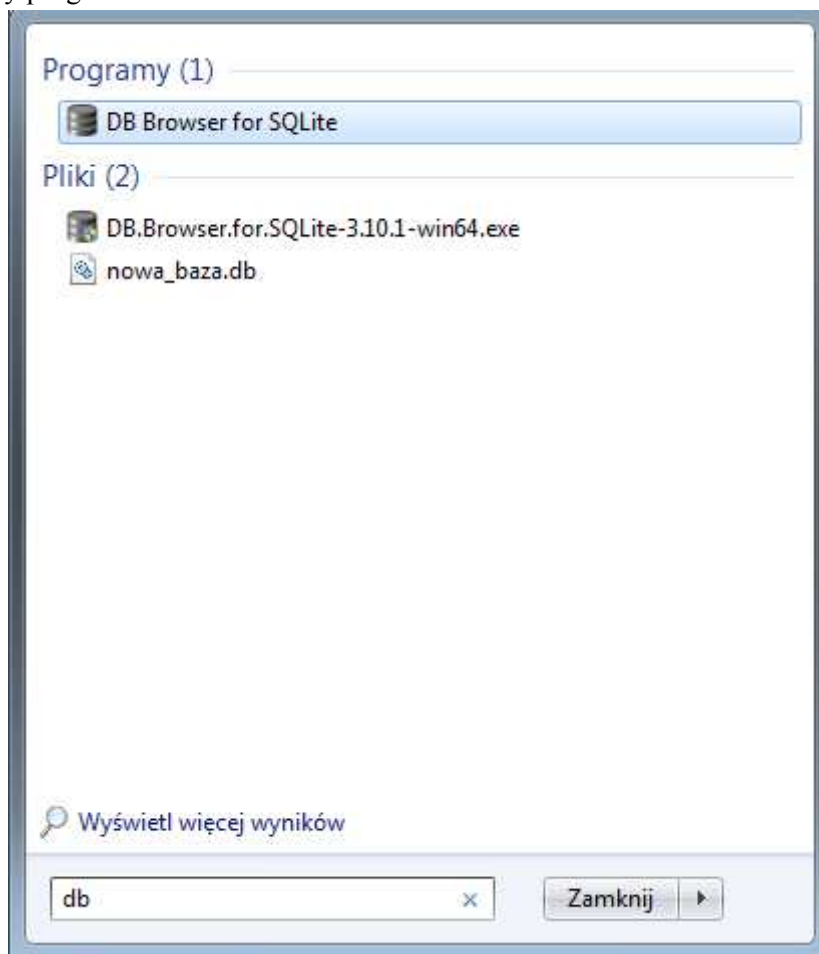


Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny

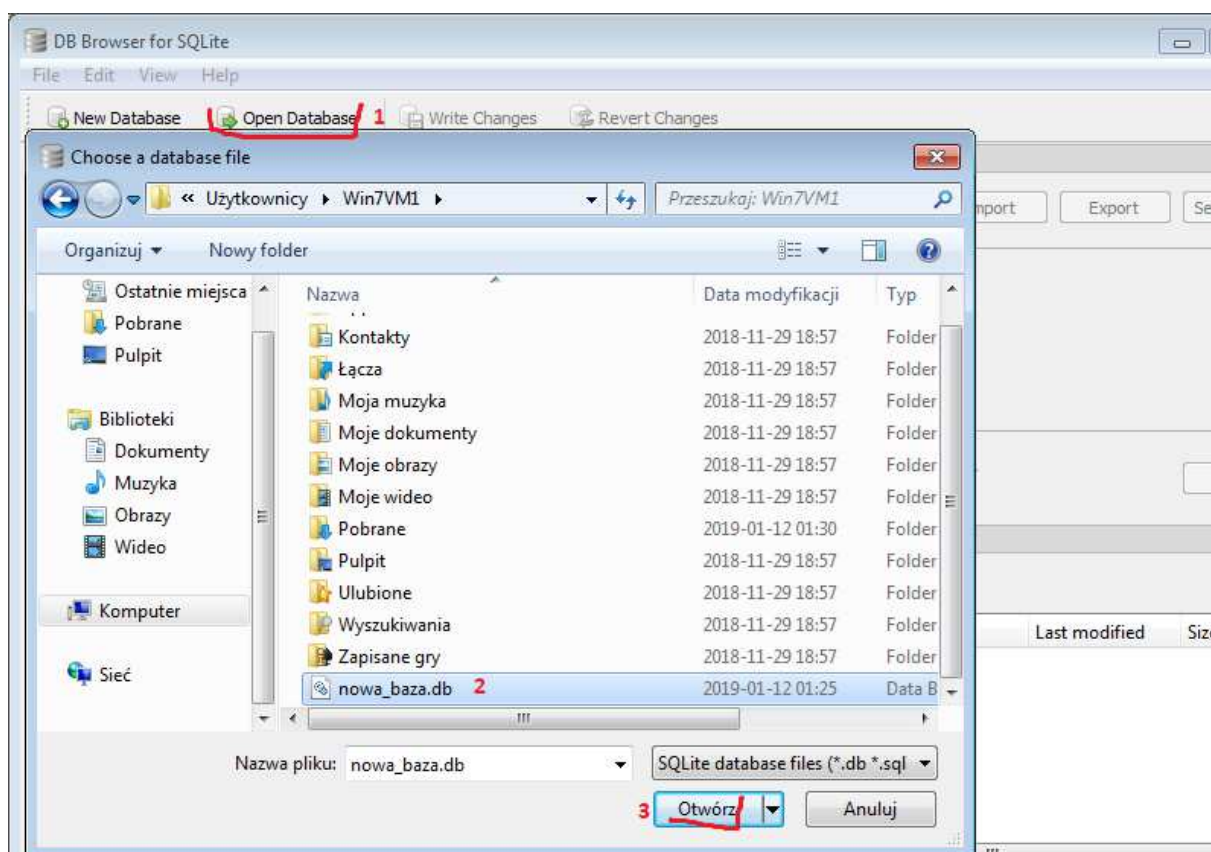


Następnie uruchommy program.

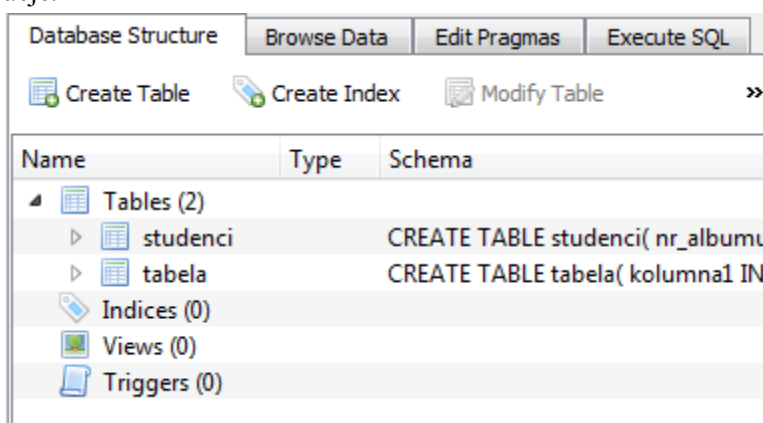


Otwórzmy utworzoną przedtem bazę danych

*Projekt „SezAM wiedzy, kompetencji i umiejętności” jest współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego w ramach Programu Operacyjnego Wiedza Edukacja Rozwój*

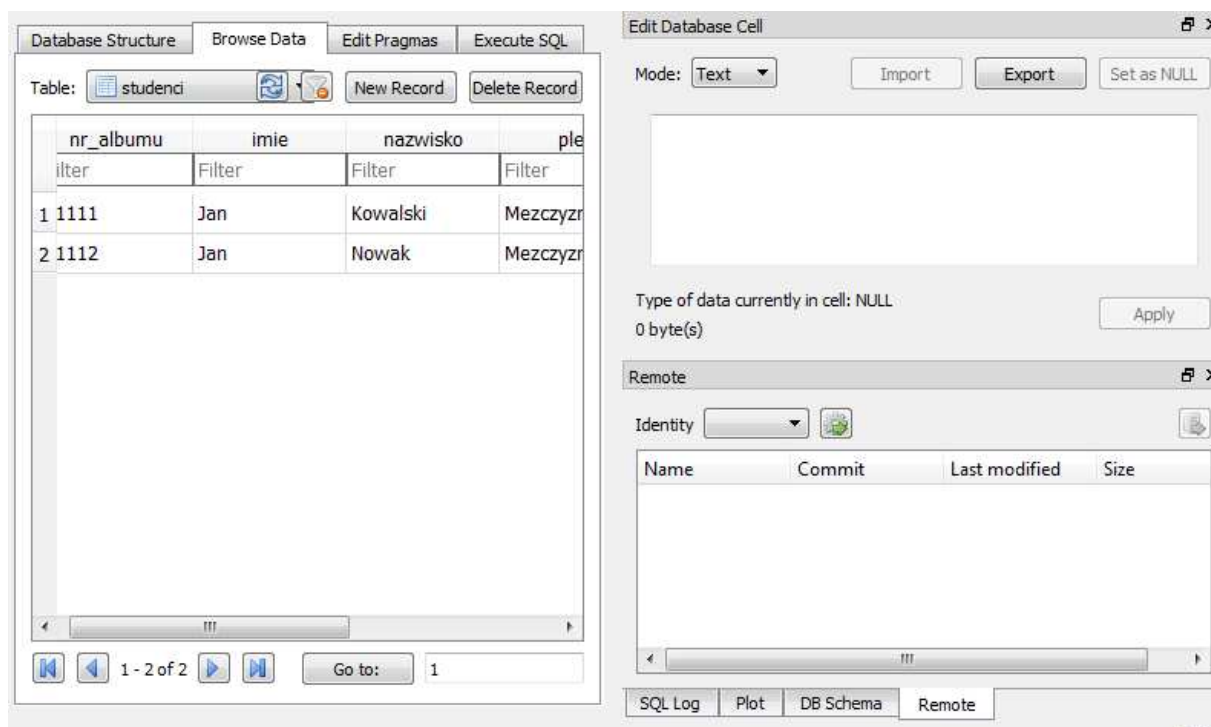


Zakładka *Database Structure* ukazuje strukturę bazy danych, czyli w tym przypadku znajdujące się w niej tabele oraz ich specyfikacje.



Zakładka *Browse Data* pozwala nam przeglądać dane zawarte w tabelach. Za pomocą listy jednokrotnego wyboru o etykiecie *Table* wybieramy tabelę, której chcemy przejrzeć zawartość (w tym przypadku tabela *studenci*).





Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: studenci

nr_albumu	imie	nazwisko	ple
1 1111	Jan	Kowalski	Mezczyzn
2 1112	Jan	Nowak	Mezczyzn

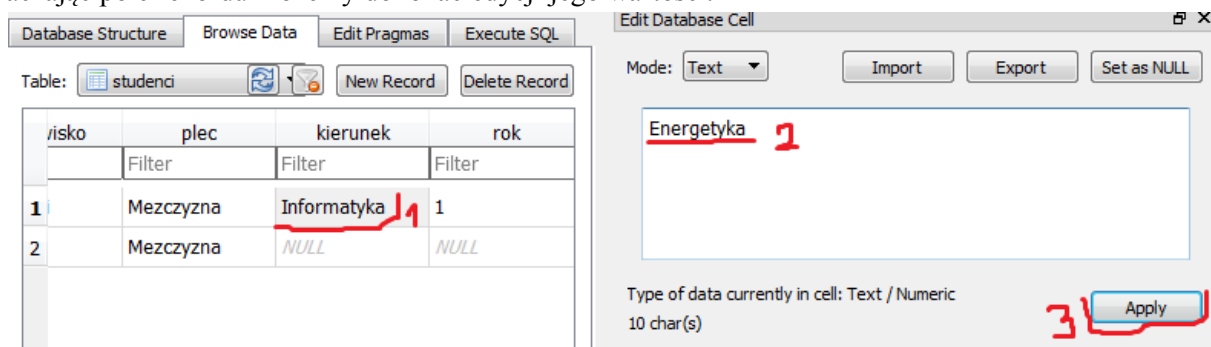
Mode: Text

Type of data currently in cell: NULL  
0 byte(s)

Remote

SQL Log | Plot | DB Schema | Remote

Zaznaczając pole rekordu możemy dokonać edycji jego wartości.



Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: studenci

nr_albumu	imie	nazwisko	plec
1	Mezczyzna	Informatyka	1
2	Mezczyzna	NULL	NULL

Mode: Text

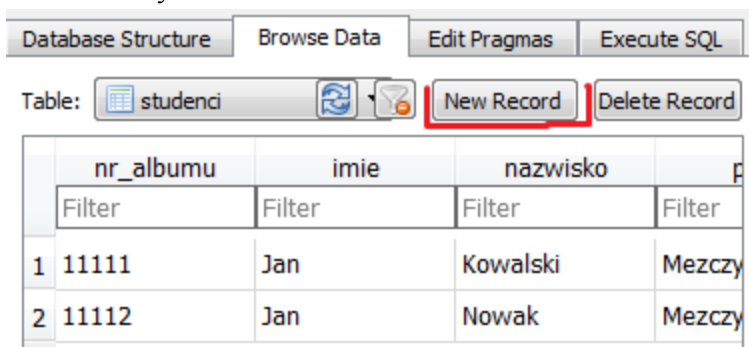
Type of data currently in cell: Text / Numeric  
10 char(s)

Apply

Wartość pola kierunek dla pierwszego rekordu zostanie zmieniona.

nr_albumu	imie	nazwisko	plec
1	Mezczyzna	Energetyka	1
2	Mezczyzna	NULL	NULL

Możemy także dodawać nowe rekordy do tabeli.



Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: studenci

New Record

nr_albumu	imie	nazwisko	plec
1 11111	Jan	Kowalski	Mezczyzn
2 11112	Jan	Nowak	Mezczyzn



Utworzony zostanie nowy rekord.

Database Structure Browse Data Edit Pragmas Execute SQL

Table: studenci New Record Delete Record

	nr_albumu	imie	nazwisko	p
	Filter	Filter	Filter	Filter
1	11111	Jan	Kowalski	Mezczy
2	11112	Jan	Nowak	Mezczy
3	11113			

W aktualnej formie rekord ten niezgodny jest ze schematem tabeli – pola, których kolumny posiadają parametru *NOT NULL* są puste. Należy te kolumny uzupełnić danymi.

Database Structure Browse Data Edit Pragmas Execute SQL

Table: studenci New Record Delete Record

	nr_albumu	imie	nazwisko	p
	Filter	Filter	Filter	Filter
1	11111	Jan	Kowalski	Mezczy
2	11112	Jan	Nowak	Mezczy
3	11113			

Edit Database Cell

Mode: Text Import Export Set as NULL

Wincent

Type of data currently in cell: Text / Numeric  
7 char(s)

Apply

Warto zauważyć, że pola, które muszą posiadać wartość są całkowicie puste, zaś pola opcjonalne posiadają wartość *NULL*.

	nazwisko	plec	kierunek	
	Filter	Filter	Filter	Filter
1	Kowalski	Mezczyzna	Energetyka	1
2	Nowak	Mezczyzna	NULL	NULL
3			NULL	NULL

Po uzupełnieniu rekordu danymi, jest on w pełni poprawny i zgodny ze schematem tabeli.

Database Structure Browse Data Edit Pragmas Execute SQL

Table: studenci New Record Delete Record

	nr_albumu	imie	nazwisko	p
	Filter	Filter	Filter	Filter
1	11111	Jan	Kowalski	Mezczy
2	11112	Jan	Nowak	Mezczy
3	11113	Wincent	Polak	Mezczy

Kolejną zakładką jest *Edit Pragmas*, która pozwala na zmianę konfiguracji działania SZBD.

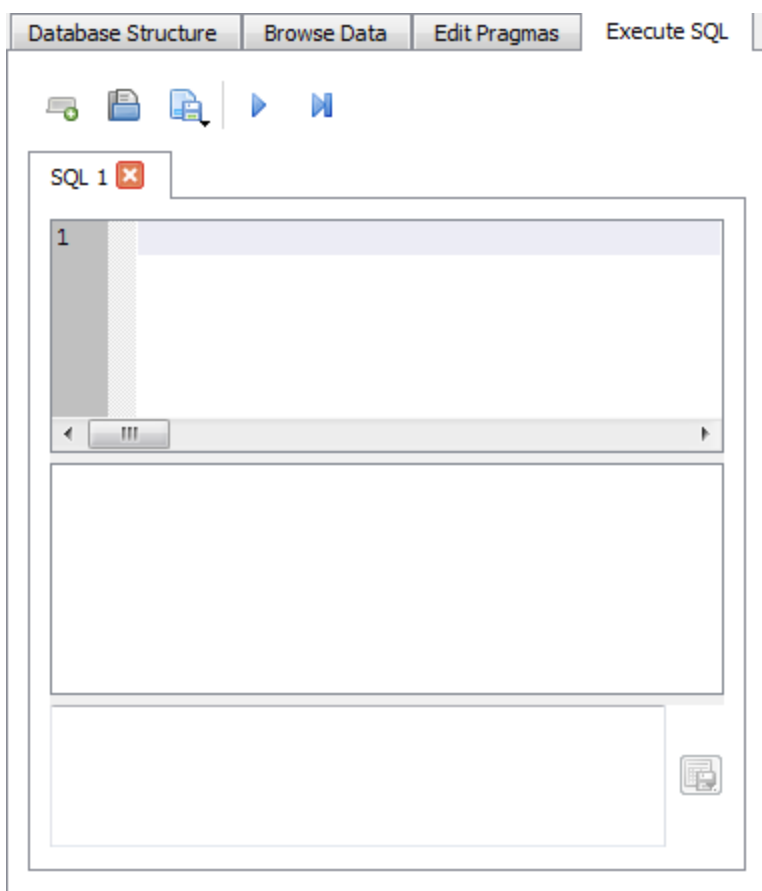


Database Structure Browse Data Edit Pragmas Execute SQL

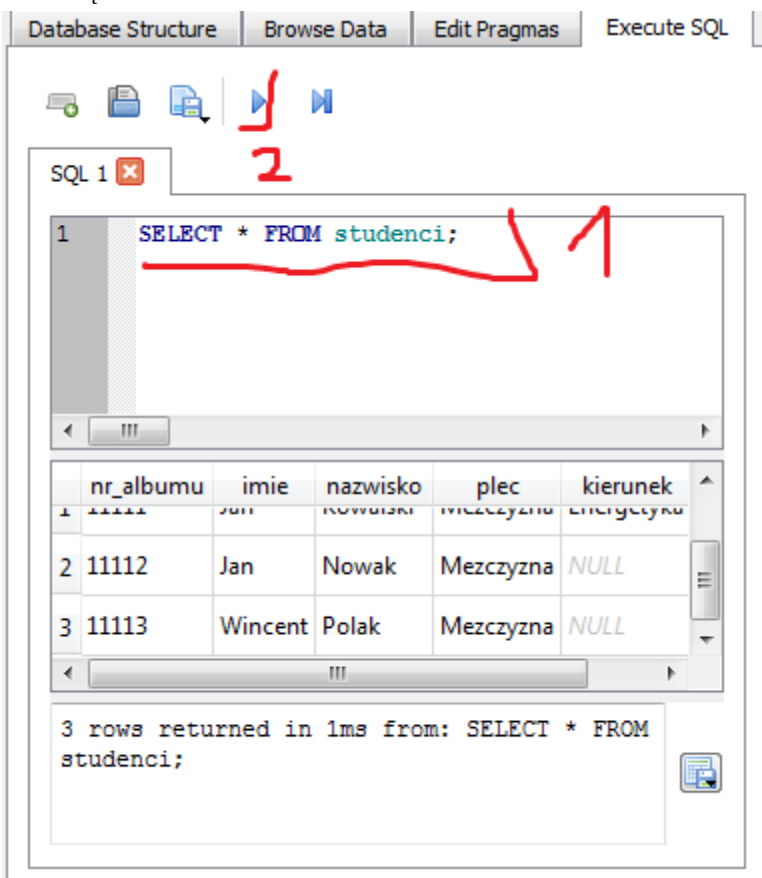
Auto Vacuum	None
Automatic Index	<input checked="" type="checkbox"/>
Checkpoint Full FSYNC	<input type="checkbox"/>
Foreign Keys	<input checked="" type="checkbox"/>
Full FSYNC	<input type="checkbox"/>
Ignore Check Constraints	<input type="checkbox"/>
Journal Mode	Delete
Journal Size Limit	-1
Locking Mode	Normal
Max Page Count	1073741823
Page Size	4096
Recursive Triggers	<input type="checkbox"/>
Secure Delete	<input type="checkbox"/>
Synchronous	Full
Temp Store	Default
User Version	0

Save Cancel

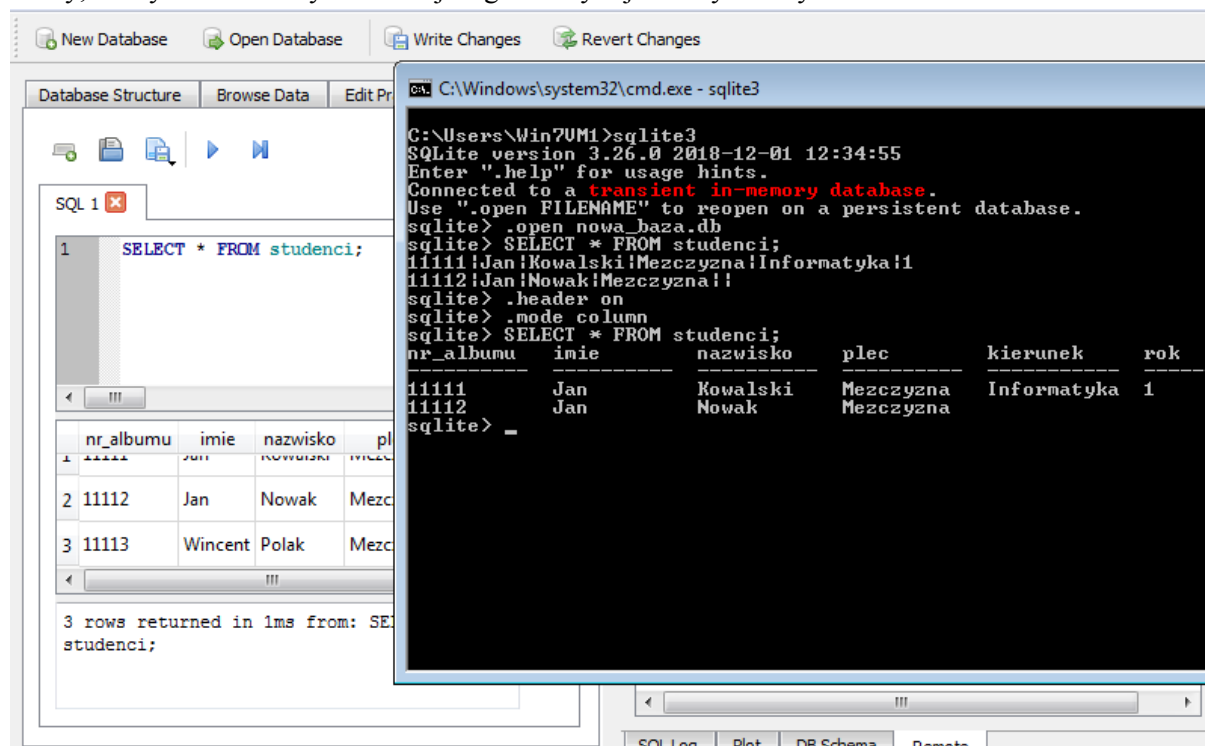
Następną zakładką jest *Execute SQL*, która pozwala na wykonywanie kwerend. Pozwala to użytkownikowi wykonać kwerendę w otwartej bazie danych.



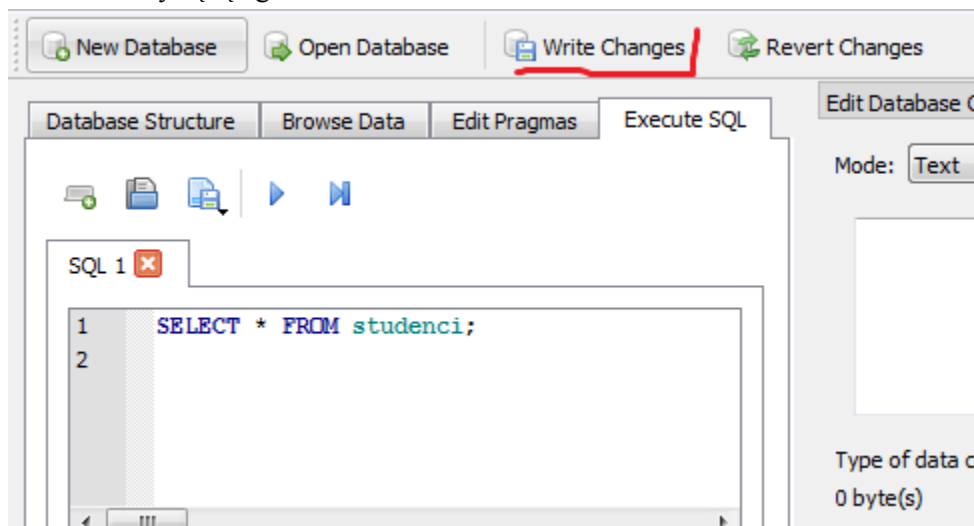
Wykonajmy dowolną kwerendę.



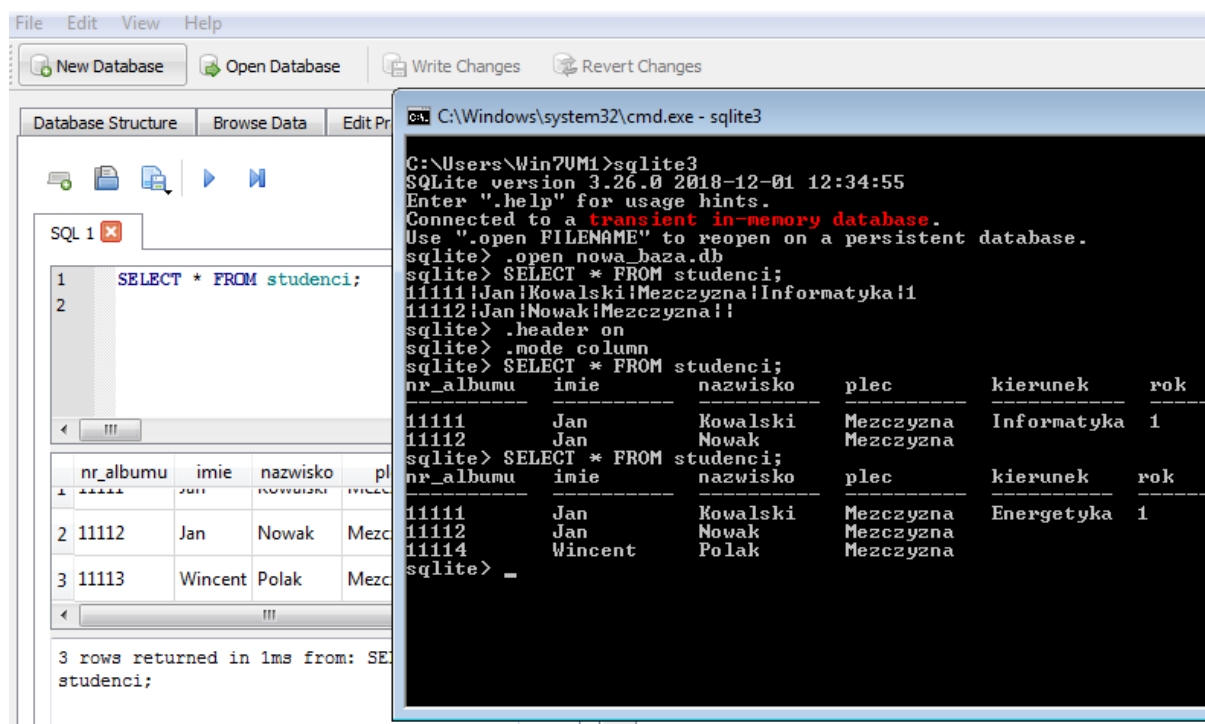
Zauważmy, że wynik kwerendy w interfejsie graficznym jest inny niż wynik w konsoli.



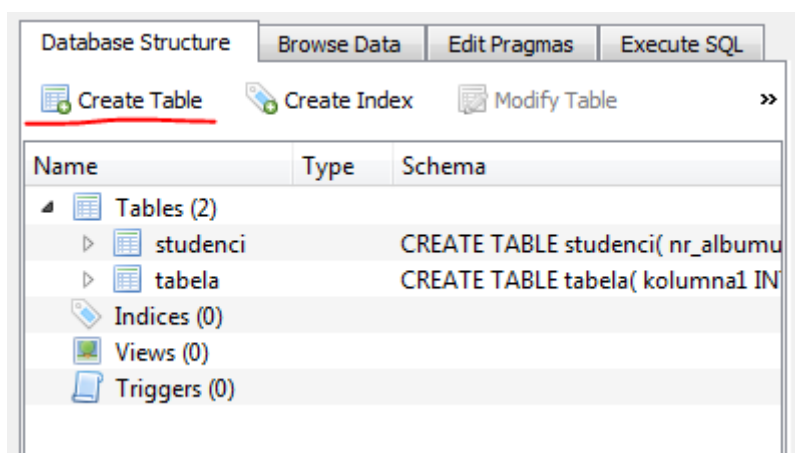
Jest tak dlatego, że program SQLite Database Browser pracuje na kopii tymczasowej, a zmiany do bazy danych wprowadza dopiero po naciśnięciu przycisku *Write Changes*. Zmiany zostaną wprowadzone dopiero wtedy, jeśli wszystkie utworzone rekordy będą zgodne ze schematem tabeli itd.



Po zapisaniu zmian wynik zarówno w interfejsie graficznym jak i konsoli jest identyczny.



Interfejs graficzny pozwala nam nie tylko na pracowanie na utworzonych już tabelach. Tabele możemy tworzyć oraz modyfikować. Aby utworzyć tabelę przejdźmy do zakładki *Database Structure* oraz naciśnijmy przycisk *Create Table*.



Uruchomiony zostanie kreator tabeli, w tym przypadku nadamy jej nazwę przedmioty.

Edit table definition

Table

przedmioty

Advanced

Fields

Add field
 Remove field
 Move field up
 Move field down

Name	Type	Not	PK	AI	U	Default	Check
<div>Check constraint</div>							

1

2

3

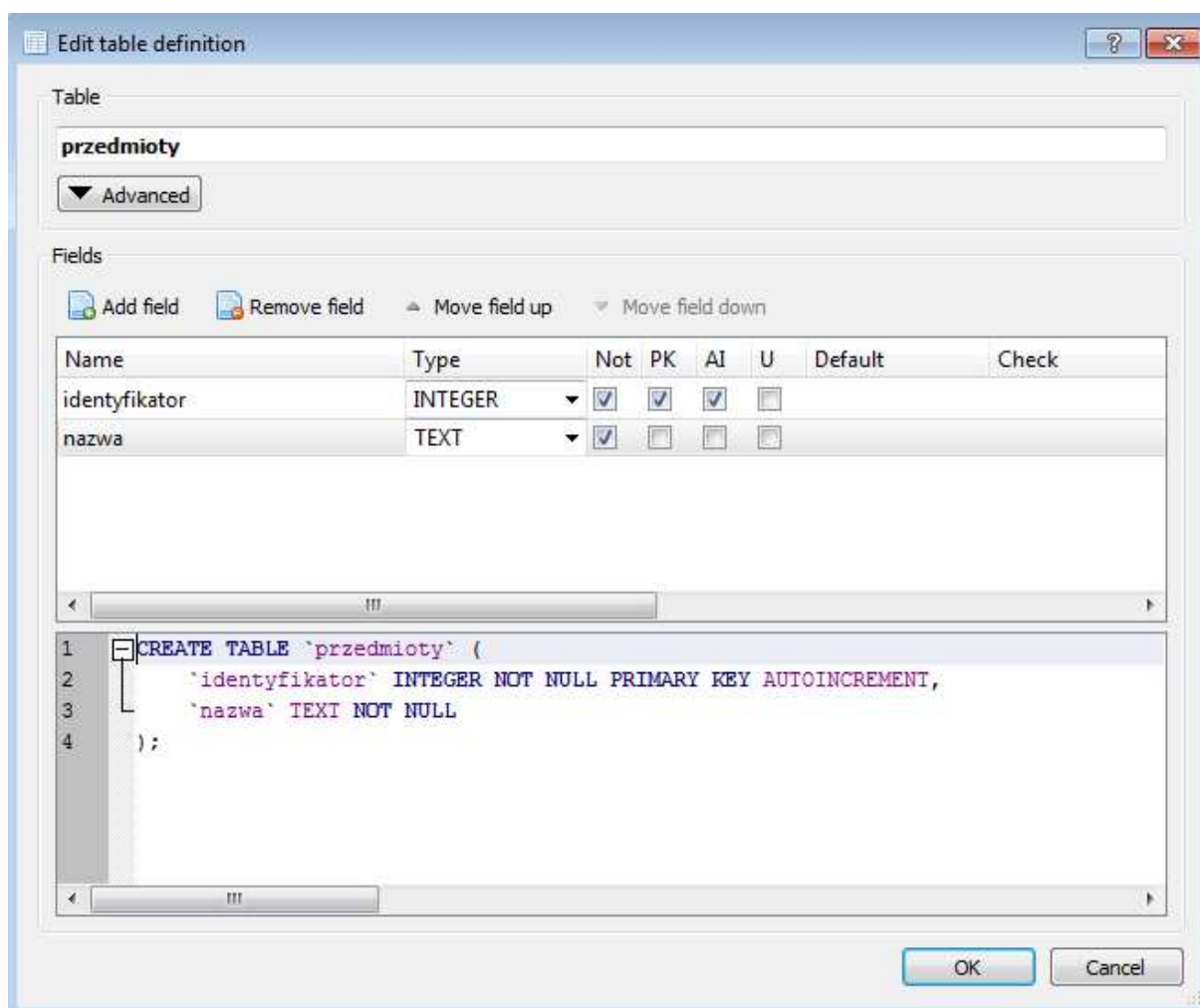
```
CREATE TABLE `przedmioty` (

```

);

Za pomocą przycisku *Add field* możemy dodać kolejne kolumny oraz ustalać typ danych oraz właściwości. Dodajmy kolumnę *identyfikator* oraz *nazwa*.





**Edit table definition**

Table: **przedmioty**

Advanced

Fields

Add field Remove field Move field up Move field down

Name	Type	Not	PK	AI	U	Default	Check
identyfikator	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
nazwa	TEXT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

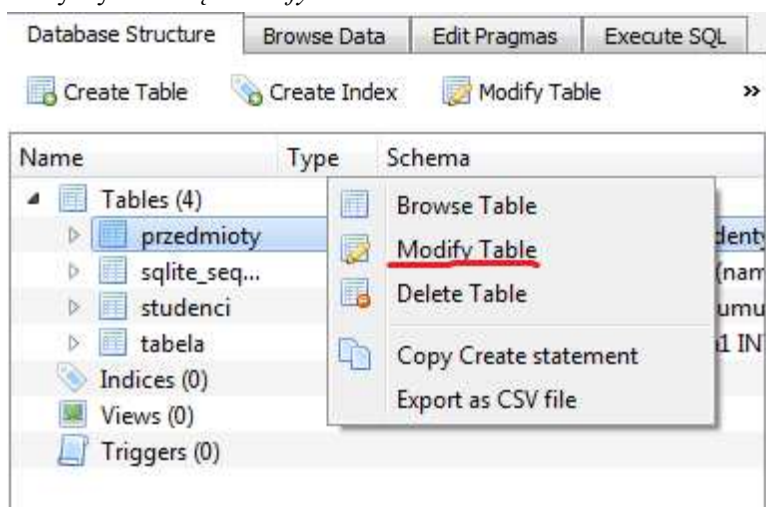
```

1 CREATE TABLE `przedmioty` (
2   `identyfikator` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
3   `nazwa` TEXT NOT NULL
4 );
  
```

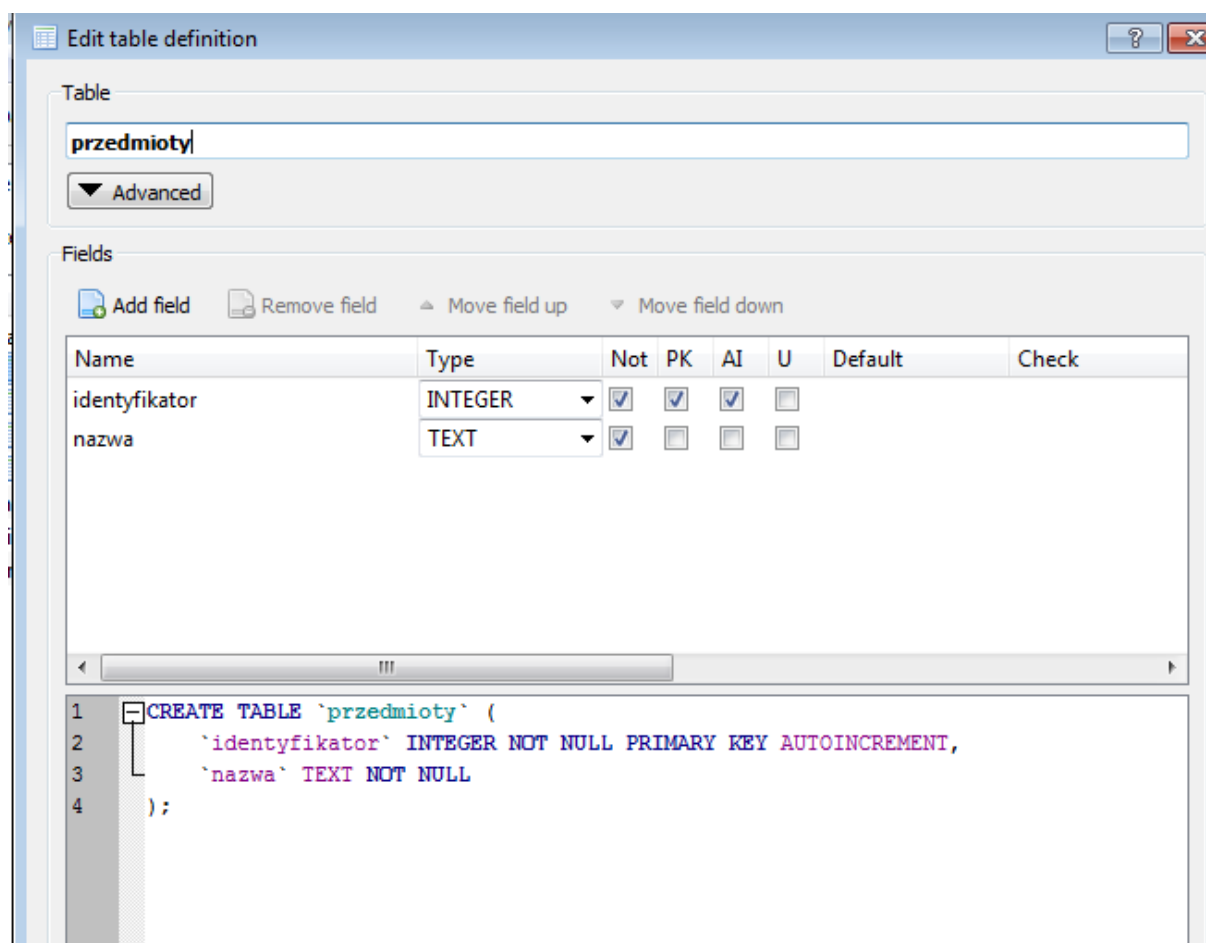
OK Cancel

Kolumna *identyfikator* posiada typ danych *INTEGER*, nie może być pusta, jest *kluczem podstawowym* (*primary key*) oraz wartość tego pola jest inkrementowana dla każdego nowego rekordu. Kolumna *nazwa* posiada typ danych *TEXT* oraz nie może być pusta.

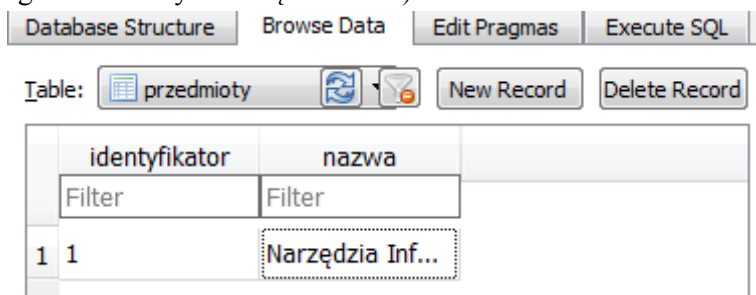
Po kliknięciu przycisku *OK*, tabela zostaje utworzona, aby dokonać zmian w jej strukturze możemy nacisnąć na niej prawym przyciskiem myszy i kliknąć *Modify Table*.



Wywołany zostanie ponownie edytor struktury tabeli.



W zakładce po dodaniu nowego rekordu, wartość pola identyfikator automatycznie przybierze wartość 1 (wartość pola dla kolejnego rekordu wynosić będzie 2 itd.)



Zapiszmy zmiany w bazie danych klikając *Write Changes*, jeśli nie zostanie wyświetlony żadnej błąd, oznacza to, że poprawnie stworzyliśmy nową tabelę oraz wprowadziliśmy do niej rekord.

## 4. Zadania

Na każdym etapie zadania twórz screenshot'y (w nazwach powinna znajdować się numeracja wskazująca wykonywaną kolejność kroków), a następnie spakuj do archiwum zip o nazwie *WdZOW\_LAB\_NrAlbumu* oraz wyślij na adres e-mail podany przez prowadzącego zajęcia laboratoryjne.

### 4.1 Skonfiguruj maszynę wirtualną z dostępem do Internetu

### 4.2 Dokonaj instalacji oprogramowania

- Zainstaluj SQLite

- Dodaj lokalizację SQLite do zmiennej środowiskowej Path
- Zainstaluj SQL Database Browser
- Sporządź zrzuty ekranu ilustrujące wykonanie powyższych punktów

#### **4.3** Dokonaj operacji na bazach danych posługując się Sqlite3

- Stwórz bazę danych o nazwie .....
- Stwórz tabelę .....

#### **4.4** Dokonaj operacji na bazach danych posługując się SQLite Database Browser

- Otwórz utworzoną w poprzednim zadaniu bazę danych
- Dodaj nową .....

### **Literatura**

1. A. Kisielewicz, Wprowadzenie do informatyki, Helion, Gliwice 2002
2. Scott H. A. Clark, W sercu PC – wg Petera Nortona, Helion, Gliwice 2002
3. J. Shim, J. Siegel, R. Chi, Technologia Informacyjna, Dom Wydawniczy ABC, Warszawa, 1999
4. A. Silberschatz, P.B. Galvin, G. Gagne, Podstawy systemów operacyjnych, WNT, Warszawa 2006
5. A. S. Twnenbaum, Systemy operacyjne, Helion, Gliwice 2010
6. P. Beynon-Davies, Systemy baz danych, WNT, Warszawa 2000
7. W. Stallings, Systemy operacyjne, Struktura i zasady budowy, PWN, Warszawa 2006
8. A. Jakubowski, Podstawy SQL. Ćwiczenia praktyczne, Helion, Gliwice 2004