

Python średnio zaawansowany

Dzień 8



Blok nr 3:

Przetwarzanie danych

AGENDA

- Wprowadzenie do relacyjnych baz danych
- Budowa bazy
- Język zapytań SQL
- ODBC

Relacyjne bazy danych



Teoria: relacyjne bazy danych

Relacyjna baza danych to opisany i zorganizowany zbiór tabel.

Ten sposób przechowywania informacji pozwala na uniknięcie redundancji (powtarzania się danych) oraz przeprowadzanie analiz na podstawie wielu tabel.

Każda tabela składa się z rekordów (krotek); tak nazywamy pojedyncze wiersze.

Poszczególne rekordy składają się z pól (atrybutów); każde pole przechowuje jedną porcję danych.

Praktyka: relacyjne bazy danych

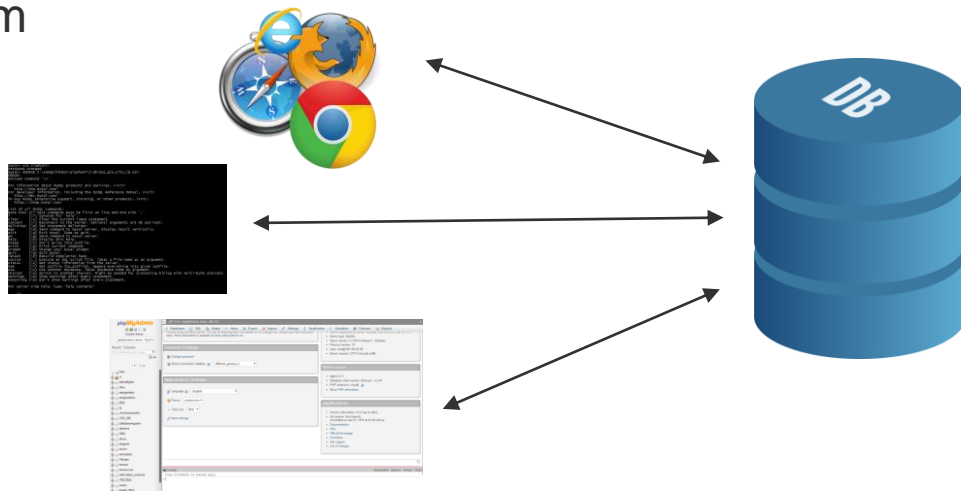
Dane cyfrowe gromadzone są zgodnie z zasadami przyjętymi dla programu komputerowego specjalizowanego do gromadzenia i przetwarzania tych danych (DBMS).



DBMS

Database Management System – system zarządzania bazą danych

Większość spotykanych systemów działa w oparciu o architekturę klient – serwer, gdzie dane przechowywane w bazie są udostępniane przez DBMS będący serwerem



DBMS

Popularne RDBMS (**Relational** Database Management Systems):

- MS SQL
- MS Access
- PostgreSQL
- Oracle Database Server
- MySQL
- **SQLite – wbudowany w środowisko Python**
- DB2

Lekki silnik: SQLite



- baza danych jednoplikowa lub trzymana wyłącznie w pamięci (in-memory),
- często używana na w systemach wbudowanych (GPS, telewizor, router SOHO) ale też w przeglądarkach czy telefonach komórkowych
- maksymalny rozmiar bazy to 140TB

Dla MS Windows:

<https://www.sqlite.org/download.html>

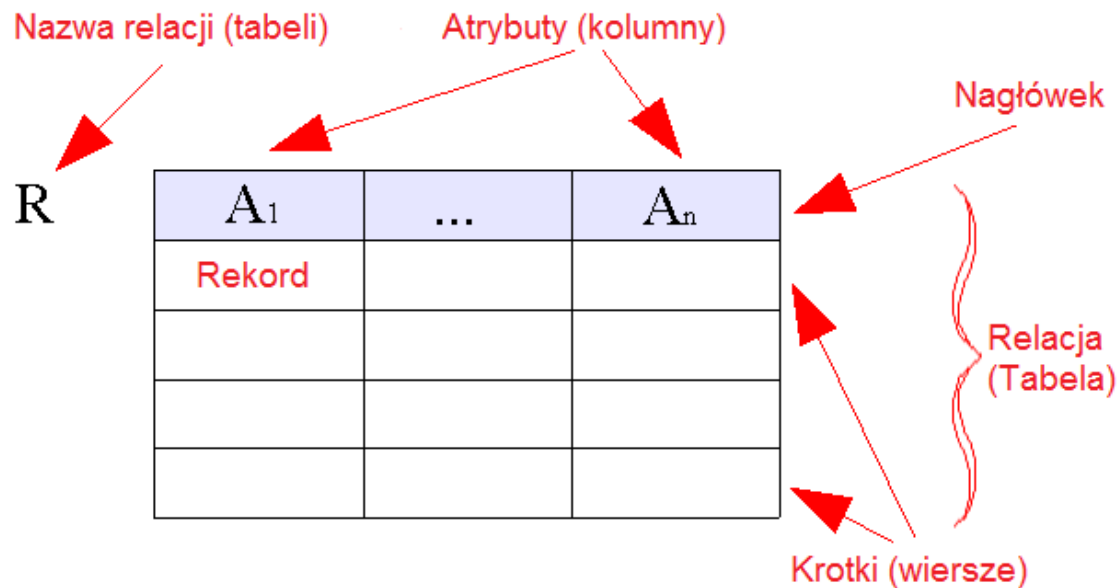
Uruchomienie:

> sqlite **plik_baz_danych.db**

Istnieje ,ożliwość podłączenia się przez PyCharm

Budowa bazy

Relacyjne bazy danych



Relacyjne bazy danych: klucze

Każda relacja (tabela) posiada klucz główny (primary key), który jest unikatowym identyfikatorem w relacji – ma on za zadanie jednoznacznie identyfikować każdy rekord (wpis) w tabeli.

Klucz główny może składać się z kilku kolumn, wówczas wartości w wyznaczonych kolumnach są jako zestaw niepowtarzalne w całej relacji.

Klucz obcy służy do wskazywania związku pomiędzy danymi składowanymi w różnych tabelach.

Klucz główny i klucz obcy

Klienci		
id (klucz główny)	Imię	Nazwisko
1	Jan	Kowalski

Jeden klient może posiadać
wiele adresów

Adresy			
id (klucz główny)	Klient_id (klucz obcy)	Alias	Adres
1	1	Dom	ul. Pythona 123 12-345 Pythonowo
2	1	Praca	ul. Pythona 123 12-345 Pythonowo

Język zapytań SQL

Język zapytań SQL

Sposób wydawania poleceń: SQL (Structured Query Language):

- DDL: Data Definition Language (CREATE, DROP, ALTER)
- DML: Data Manipulation Language (INSERT, UPDATE, DELETE)
- DQL: Data Query Language (SELECT)
- DCL: Data Control Language (GRANT, REVOKE, DENY)

Kiedyś: „**SELECT * FROM** " + tabela + „**WHERE** <warunek>„

Dziś: ORM (Object Relation Mapper), np. SQLAlchemy

Język zapytań SQL

Zalety:

- przydatny w procesie debugowania
- wydajny

Wady:

- implementacje zapytań mogą być podatne na atak SQL Injection
- implementacja zapytań zależna od rodzaju silnika
- zapytania wymagają aktualizacji podczas każdej modyfikacji struktury tabel

Język zapytań SQL

```
sqlite > select * from klienci;
```

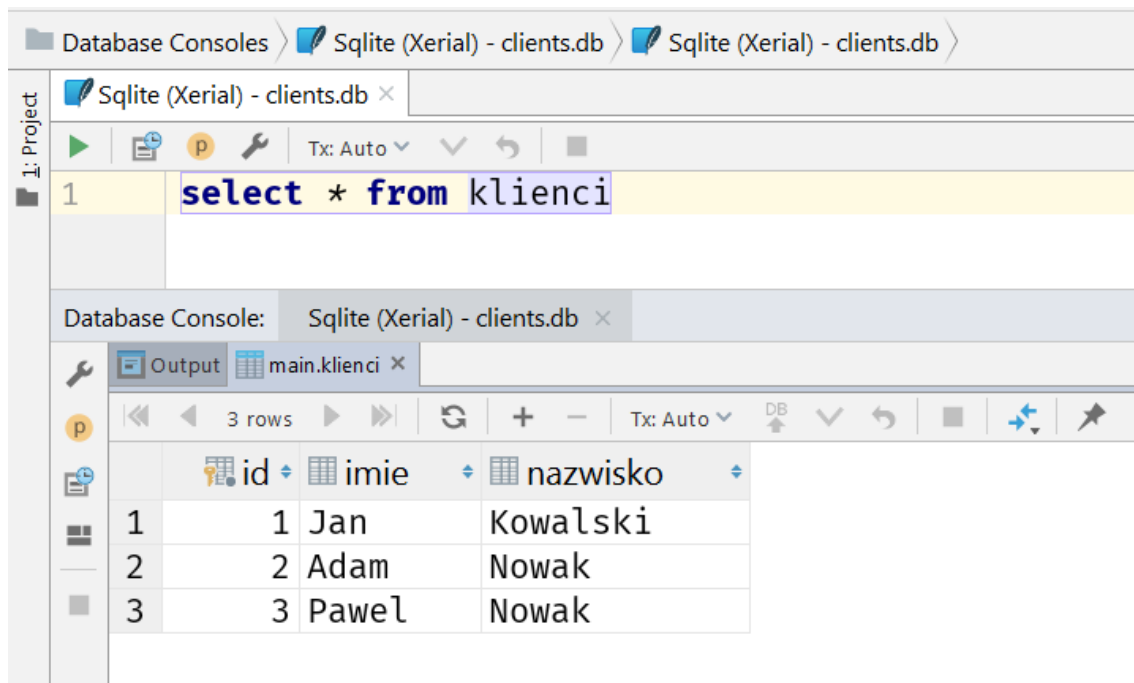
```
1 | Jan | Kowalski
```

```
2 | Adam | Nowak
```

```
3 | Pawel | Nowak
```

```
sqlite> select * from klienci;  
1|Jan|Kowalski  
2|Adam|Nowak  
3|Paweł|Nowak
```

Język zapytań SQL



The screenshot shows a database console interface. The top pane displays the SQL query: `select * from klienci`. The bottom pane shows the results of the query, which are 3 rows of data from the `main.klienci` table. The results are displayed in a table with columns `id`, `imie`, and `nazwisko`.

	id	imie	nazwisko
1	1	Jan	Kowalski
2	2	Adam	Nowak
3	3	Pawel	Nowak

Język zapytań SQL

```
sqlite > select nazwisko from klienci;
```

```
Kowalski
```

```
Nowak
```

```
Nowak
```

```
sqlite> select nazwisko from klienci;  
Kowalski  
Nowak  
Nowak
```

Język zapytań SQL

```
sqlite > select * from klienci where imie = "Adam";
```

```
2 | Adam | Nowak
```

```
sqlite> select * from klienci where imie = "Adam";  
2|Adam|Nowak
```

Język zapytań SQL

```
sqlite > SELECT * FROM adresy;
```

```
1 | Dom | ul.Pythona 15 12 - 345 Pythonowo | 1
```

```
2 | Praca | ul.Pythona 123 12 - 345 Pythonowo | 1
```

```
3 | Dom | ul.Pythona 23 12 - 345 Pythonowo | 2
```

```
sqlite> SELECT * FROM adresy;  
1|Dom|ul. Pythona 15 12-345 Pythonowo|1  
2|Praca|ul. Pythona 123 12-345 Pythonowo|1  
3|Dom|ul. Pythona 23 12-345 Pythonowo|2  
sqlite>
```

Język zapytań SQL - klucz obcy

```
sqlite > SELECT imie, nazwisko, adres, nazwa FROM klienci LEFT JOIN adresy  
ON klienci.id = adresy.klient_id;
```

```
Jan | Kowalski | ul.Pythona 15 12 - 345 Pythonowo | Dom
```

```
Jan | Kowalski | ul.Pythona 123 12 - 345 Pythonowo | Praca
```

```
Adam | Nowak | ul.Pythona 23 12 - 345 Pythonowo | Dom
```

```
Pawel | Nowak | |
```

```
sqlite> SELECT imie, nazwisko, adres, nazwa  
...> FROM klienci LEFT JOIN adresy ON klienci.id = adresy.klient_id;  
Jan|Kowalski|ul. Pythona 15 12-345 Pythonowo|Dom  
Jan|Kowalski|ul. Pythona 123 12-345 Pythonowo|Praca  
Adam|Nowak|ul. Pythona 23 12-345 Pythonowo|Dom  
Pawel|Nowak||  
sqlite>
```

| ODBC

ODBC

ODBC (ang. Open DataBase Connectivity) - interfejs pozwalający programom łączyć się z DBMS.

To API niezależne od języka programowania, systemu operacyjnego i bazy danych. W skład ODBC wchodzi wywołania wbudowane w aplikacje oraz sterowniki ODBC.

Implementacje ODBC dostępne są w MS Windows, Linux/Unix, Mac OS X.

W systemach bazodanowych typu klient-serwer (np. Oracle lub PostgreSQL) sterowniki dają dostęp do silnika baz danych, natomiast w programach dla komputerów osobistych sterowniki sięgają bezpośrednio do danych.

Jedna z implementacji dla Pythona: **pyodbc** (umożliwia skorzystanie z dowolnego sterownika zainstalowanego w systemie operacyjnym).

Dzięki!

