

# Kurs języka Python

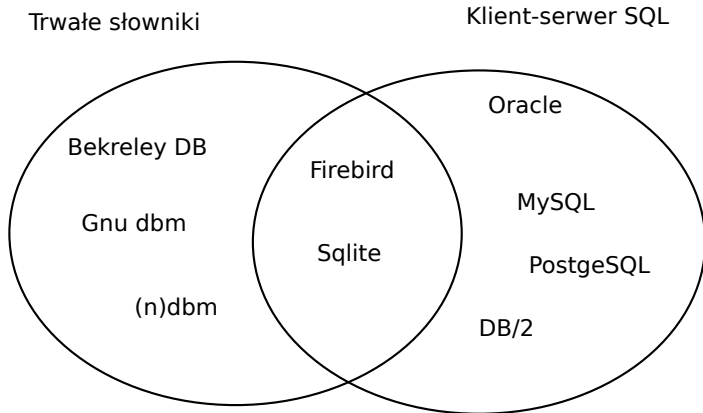
## Wykład 7.

Marcin Młotkowski

30 listopada 2009

- 1 Bazy danych DBM
- 2 Przechowywanie obiektów: shelve
- 3 Bazy danych SQL

# Rodzaje baz danych



# Bazy danych typu DBM

## Database manager

- Dane przechowywane są w pliku, który przypomina słownik
- Dostęp do danych jest po kluczu
- Wartość zwykle może być tylko napisem
- Implementacja: zwykle tablice haszujące i b-drzewa
- Brak odrębnego serwera, dane pamiętane są w lokalnym pliku
- Szybkie!!!

# Korzystanie z DBM

```
import <modul>  
db = <modul>.open('storage.dbm', 'c')
```

```
db['jeden'] = 'one'  
db['dwa'] = 'two'
```

```
if 'trzy' in db:  
    del db['trzy']
```

```
db.close()
```

# Korzystanie z DBM

```
import <modul>  
db = <modul>.open('storage.dbm', 'c')
```

```
db['jeden'] = 'one'  
db['dwa'] = 'two'
```

```
if 'trzy' in db:  
    del db['trzy']
```

```
db.close()
```

# Dostępne moduły

Python 2.*	Python 3.*	Opis
dumbdb	dbm.dumb	wolny, ale czysto Pythonowa implementacja
bsddb (dbhash)	—	implementacja Berkeley DB
dbm	dbm.ndbm	interfejs uniksowej (n)dbm
gdbm	dbm.gnu	rozszerzenie dbm

# Przetwarzanie całej kolekcji

```
for key in db.keys():  
    print db[key]
```

## Ostrzeżenie

Wymaga odczytu do pamięci całego pliku

# Przetwarzanie całej kolekcji

```
for key in db.keys():  
    print db[key]
```

## Ostrzeżenie

Wymaga odczytu do pamięci całego pliku

# Przetwarzanie kolekcji, dumbdbm

```
for key in db:  
    print 'db[', key, ']' =', db[key]
```

```
db.close()
```

# Przetwarzanie kolekcji

## dbhash

```
print db.first()
while True:
    try:
        el = db.next()
        print el
    except KeyError:
        break
```

## gdbm

```
k = db.firstkey()
while k != None:
    print k
    k = db.nextkey(k)
```

# Dostępne moduły

	<code>for-in</code>	<code>firstkey()</code> , <code>nextkey()</code>	<code>first()</code> , <code>next()</code>
<code>dumbdbm</code>	X	—	—
<code>(n)dbm</code>	—	—	—
<code>gdbm</code>	—	X	—
<code>dbhash</code>	—	—	X

# Jak sobie radzić w tym bałaganie

Używać generycznego pakietu anydbm

Używać pakietu whichdb:

```
>>> import anydbm  
>>> whichdb.whichdb('plik.db')
```

# Jak sobie radzić w tym bałaganie

Używać generycznego pakietu anydbm

Używać pakietu whichdb:

```
>>> import anydbm  
>>> whichdb.whichdb('plik.db')
```

# Berkeley DB

## Gdzie można spotkać Berkeley DB

OpenLDAP, Subversion, Spamassasin, KDevelop

### Właściwości:

- transakcje;
- replikacje;
- blokowanie rekordów;
- kluczami są liczby naturalne.

# A jak przechowywać obiekty

Pakiet `shelve`

pliki są słownikami przechowującymi pary (string, obiekt)

## Pakiet shelve

### przykład

```
import shelve
db = shelve.open('dane.db')
db['lista'] = [2,3,5,7,11]
db.sync()
del db['obiekt']
db.close()
```

# Serwery baz danych

- Oracle
- DB/2
- MySQL
- PostgreSQL
- MS SQL
- ...

# DB API

## Python Database API Specification

Zunifikowany interfejs (metody i pola) dostępu do różnych silników relacyjnych baz danych (Aktualna wersja: 2.0)

## Otwarcie połączenia z bazą danych

```
connect('parametry') # zwraca obiekt Connection
```

### MySQL

```
import MySQLdb  
db = MySQLdb.connect(host='localhost',  
                      db='testowa',  
                      user='user',  
                      passwd='123456')
```

## Otwarcie połączenia z bazą danych

```
connect('parametry') # zwraca obiekt Connection
```

### MySQL

```
import MySQLdb  
db = MySQLdb.connect(host='localhost',  
    db='testowa',  
    user='user',  
    passwd='123456')
```

## Zamknięcie połączenia z bazą danych

```
db.close()
```

## Komunikacja z bazą danych

### wysłanie zapytania

```
wynik = db.cursor()  
wynik.execute('SELECT * FROM Studenci')
```

### Pobranie wyniku

```
row = wynik.fetchone()  
while row:  
    print row  
    row = wynik.fetchone()
```

### Opcjonalnie

```
wynik.close()
```

## Komunikacja z bazą danych

### wysłanie zapytania

```
wynik = db.cursor()  
wynik.execute('SELECT * FROM Studenci')
```

### Pobranie wyniku

```
row = wynik.fetchone()  
while row:  
    print row  
    row = wynik.fetchone()
```

### Opcjonalnie

```
wynik.close()
```

# Wynik

Atrybuty wyniku (obiekt klasy Cursor):

- description: lista opisów poszczególnych kolumn odpowiedzi
- rowcount: liczba wierszy w odpowiedzi lub liczba przetworzonych wierszy (w przypadku INSERT czy UPDATE)
- ...

## DB API - dodatkowe informacje

Standaryzacja wyjątków:

Warning, DatabaseError, NotSupportedError, ...

# SQLite

- 'Plikowa' baza danych, nie wymaga zewnętrznego serwera ani kontaktu z adminem;
- moduł: `sqlite3`
- Implementuje DB API 2.0 z rozszerzeniami

## Korzystanie z Sqlite

### Założenie tabeli

```
import sqlite3
db = sqlite.connect('/tmp/temp.db')
kursor = db.cursor()
kursor.execute("""create table Ksiegozbior
                (Autor text, Tytul text, RokWyd int, cena real)""")
kursor.commit()
```

## Korzystanie z Sqlite

### Wstawienie wiersza do tabeli

```
cursor.execute("""insert into Ksiegozbiior values  
('Mickiewicz', 'Pan Tadeusz', 2003, 25.5)""")
```

## Korzystanie z Sqlite

### Odczyt danych (rozszerzenie DB API 2.0)

```
kursor.execute('SELECT * FROM Ksiegozbior')  
for row in kursor:  
    print row
```

# Sqlite

## Ciekawostka

```
db = sqlite.connect(':memory:')
```