**Лабораторно упражнение №8**

**Работа с SQL и бази от данни в Python**

Базата от данни е файл, който се организира за съхранение на данни. Повечето бази данни са организирани като речник - съпоставка на ключове и стойности. Най-голямата разлика е, че базата данни е на диск (или друго постоянно място за съхранение), така че тя се запазва и след приключване на програмата. Тъй като базата данни се съхранява в постоянна памет, тя може да съхранява много повече данни, отколкото речник, който е ограничен до размера на паметта в компютъра.

Има много различни системи за бази данни, които се използват за най-различни цели, включително: Oracle, MySQL, Microsoft SQL Server, PostgreSQL и SQLite.

Ще се фокусираме върху SQLite, защото тя е много често срещана база данни и е вградена в Python. SQLite е проектиран да бъде вграден в други приложения, за да осигури поддръжка на база данни в приложението. Например браузърът Firefox също използва вътрешно SQLite базата данни, както и много други продукти.

http://sqlite.org/

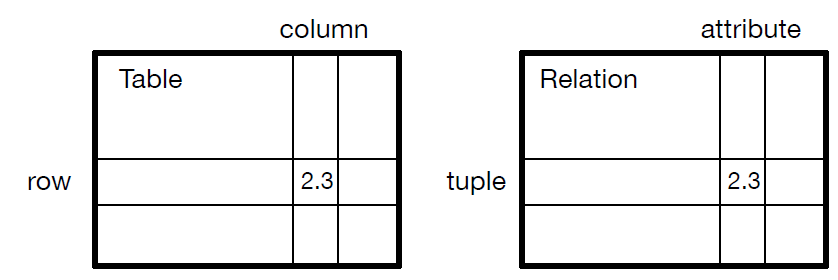
**Основни понятия**:

* атрибут - една от стойностите в един кортеж (tuple). По-често се нарича "колона" или "поле".
* ограничение - когато казваме на базата данни да наложи правило върху поле или ред в таблица. Общо ограничение е, че не може да има дублиращи се стойности в определено поле (т.е. всички стойности трябва да са уникални).
* cursor - курсорът позволява изпълнение на SQL команди в база данни и да извличане на данни от базата данни.
* браузър на базата данни - част от софтуер, който позволява директно свързване и управление на база данни, без да се налага да се пише програма.
* чужд ключ (foreign key) - цифров ключ, който сочи към първичния ключ на ред в друга таблица. Чуждите ключове установяват връзки между редовете, съхранявани в различни таблици.
* Индекс - допълнителни данни, които софтуерът на базата данни поддържа като редове и се вмъква в таблица, за да се правят търсенията много бързо.
* логически ключ (logical key) - ключ, който „външният свят“ използва за търсене на определен ред. Например в таблица с потребителски акаунти имейл адресът на човек може да бъде добър кандидат като логически ключ за данните на потребителя.
* нормализиране - проектиране на модел на данни, така че да не се репликират данните.
* първичен ключ (primary key) - числен ключ, присвоен на всеки ред, който се използва за препращане към един ред в таблица от друга таблица. Често базата данни е конфигурирана така, че автоматично да присвоява първични ключове, като се въвеждат редове.
* връзка (relation) - област в базата данни, която съдържа кортежи и атрибути.
* кортеж (tuple) - единичен запис в таблица на базата данни, който е набор от атрибути. По-често се нарича "ред".

**1. Концепции за база данни**

Базата данни, изглежда като електронна таблица с няколко листа.

Основните структури на данни в база данни са: таблици, редове и колони. В техническите описания на релационните бази данни понятията таблица, ред и колона по-официално се наричат съответно отношение, кортеж и атрибут (relation, tuple, and attribute).



**2. Браузър на база данни за SQLite**

Много операции могат да бъдат извършени с помощта на софтуер, наречен браузър на базата данни за SQLite, който е свободно достъпен от:

http://sqlitebrowser.org/

С помощта на браузъра може лесно да се създават таблици, да се вмъкват данни, да се редактират данни или да се стартират прости SQL в базата данни.

**3. Структура на заявката в SQLite**

Релационна база данни е съставена от таблици, редове и колони. Колоните обикновено имат тип като данни от текст, числови данни или дати.

**3.1. Създаване на таблица**

При създаване на таблица, се посочват имената и типовете на колоните:

***Пример* 1*:***

**CREATE TABLE** Tracks (title TEXT, plays INTEGER)

**3.2. Вмъкване на ред в таблица**

При вмъкване на ред в таблица, се използва командата SQL INSERT:

***Пример* 2*:***

**INSERT INTO** Tracks (title, plays) **VALUES** ('My Way', 15)

Операторът INSERT указва името на таблицата, след това списък на полетата / колоните, които ще се зададат в новия ред, след което ключовата дума VALUES и списък на съответните стойности за всяко от полетата.

**3.3. Извличане на ред от таблица**

Командата SQL SELECT се използва за извличане на редове и колони от база данни. Операторът SELECT позволява да се укажат кои колони да се извлекат, а клауза WHERE се използва, за да се избере кои редове да се виждат. Възможно е да се използва и незадължителната клауза ORDER BY за контрол на сортирането на върнатите редове.

***Пример* 3*:***

**SELECT** \* **FROM** Tracks **WHERE** title = 'My Way'

Използването на \* означава, че базата от данни ще върне всички колони за всеки ред, който съответства на клаузата WHERE.

При Python, в клауза WHERE се използва един знак за равенство, а не двоен знак за равенство. Други логически операции, разрешени в клауза WHERE са <,>, <=,> =,! =, както и AND и OR и скоби за изграждане на логически изрази.

**3.4. Сортиране на таблица по определено поле**

***Пример* 4*:***

**SELECT** title,plays **FROM** Tracks **ORDER BY** title

**3.5. Изтриване на запис от таблица**

За да се премахне ред е необходимо да се използва оператор SQL DELETE и клауза WHERE. Клаузата WHERE определя кои редове да бъдат изтрити:

***Пример* 5*:***

**DELETE FROM** Tracks **WHERE** title = 'My Way'

**3.6. Актуализиране на колона/и в таблица**

Възможно е да се актуализира колона или колони в един или повече редове в таблицата, използвайки SQL UPDATE израза.

***Пример* 6*:***

**UPDATE** Tracks **SET** plays = 16 **WHERE** title = 'My Way'

Операцията UPDATE указва таблица и след това списък от полета и стойности, които да се променят след ключовата дума SET, а след това и незадължителна клауза WHERE за избор на редовете, които трябва да бъдат актуализирани. Един UPDATE оператор ще промени всички редове, които съответстват на клаузата WHERE. Ако клаузата WHERE не е посочена, тя изпълнява актуализацията на всички редове в таблицата.

Тези четири основни SQL команди (INSERT, SELECT, UPDATE и DELETE) осегуряват четирите основни операции, необходими за създаване и поддържане на данни.

**4. Създаване на таблица в база от данни**

При създаване на таблица в базата данни, трябва предварително да се укажат имената на всяка от колоните в таблицата и вида на данните, които ще се съхраняват във всяка колона. Когато софтуерът на базата данни знае вида на данните във всяка колона, той може да избере най-ефективния начин за съхранение и търсене на данните въз основа на типа данни.

Различните типове данни, поддържани от SQLite, могат да се разгледат на следния URL адрес:

<http://www.sqlite.org/datatypes.html>

***Пример* 7*:***

Създаване на файл на базата данни и таблица с име Tracks с две колони:

import sqlite3

conn = sqlite3.connect('music.sqlite')

cur = conn.cursor()

cur.execute('DROP TABLE IF EXISTS Tracks')

cur.execute('CREATE TABLE Tracks (title TEXT, plays INTEGER)')

conn.close()

Командата SQL INSERT посочва коя таблица ще се използва и след това дефинира нов ред, като изброява полетата, които ще се включат (title, play), последвани от VALUES, които искаме да поставим в новия ред. Определeните стойностите като въпросителни (?,?), показват, че действителните стойности се предават като кортеж („My Way“, 15) или като втори параметър на call ().

**import** sqlite3  
conn = sqlite3.connect(**'music.sqlite'**)  
cur = conn.cursor()  
cur.execute(**'INSERT INTO Tracks (title, plays) VALUES (?, ?)'**,(**'Thunderstruck'**, 20))  
cur.execute(**'INSERT INTO Tracks (title, plays) VALUES (?, ?)'**,(**'My Way'**, 15))  
conn.commit()  
print(**'Tracks:'**)  
cur.execute(**'SELECT title, plays FROM Tracks'**)  
**for** row **in** cur:  
 print(row)  
cur.execute(**'DELETE FROM Tracks WHERE plays < 100'**)  
conn.commit()  
cur.close()

***Резултат****:?*

Изходът от изпълнението на програмата е:

Tracks:

('Thunderstruck', 20)

('My Way', 15)

**Самостоятелна задача 1:**

Създайте Таблица Company, която съдържа информация за служители – номер, име, възраст, адрес, заплата. Добавете десет записа на служители. Извлечете и покажете записите. Променете заплатата на служител с номер 1 и адреса на служител 4. Изтрийте служител 2. Намерете минималната, максималната и средната заплата във фирмата. Изведете колко човека получават заплата над средната.

**Самостоятелна задача 2:**

Създайте три таблици:

* Таблицата с контакти, която съхранява информация за контакт.
* Таблицата на групите, която съхранява информацията за групата.
* Таблицата contact\_groups, която съхранява връзката между контактите и групите.

Всеки контакт има следната информация:

-Име

-Фамилия

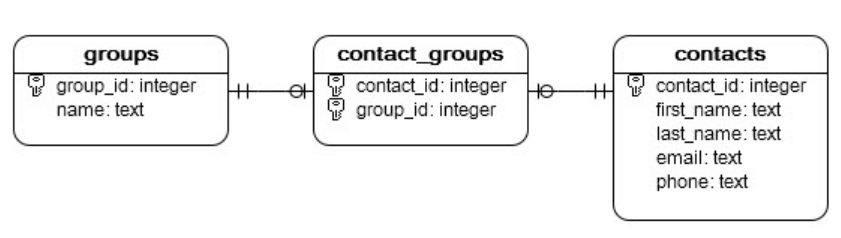
-Възраст

-Пол

-Електронна поща

-Телефон

E-mail адресът и телефонният номер трябва да са уникални. Всеки контакт може да принадлежи към една или много групи и всяка група може да има нула или много контакти.



Създайте 3 групи и 10 контакта. Изведете записите сортирани по пол. Намерете най-младия и най-възрастни участник в групата. Направете търсене на запис по телефонен номер. Променете телефонния номер на ред 5. Изтрийте запис 7.

import sqlite3

conn = sqlite3.connect("contact.sqlite")

cur = conn.cursor()

cur.execute("DROP TABLE IF EXISTS [Contacts]")

cur.execute("DROP TABLE IF EXISTS [Group]")

cur.execute("DROP TABLE IF EXISTS [Contact\_groups]")

cur.execute(

    "CREATE TABLE [Contacts] (contact\_id INTEGER PRIMARY KEY , name TEXT, family TEXT, age INTEGER,\

          gender TEXT,email TEXT, phone TEXT, UNIQUE(email,phone) )"

)

cur.execute("CREATE TABLE [Group] (group\_id INTEGER PRIMARY KEY , name TEXT )")

cur.execute(

    "CREATE TABLE [Contact\_groups] (contact\_id INTEGER , group\_id INTEGER ,\

        FOREIGN KEY (group\_id) REFERENCES [Group], \

        FOREIGN KEY (contact\_id) REFERENCES [Contacts] )"

)

for ii in range(1, 4):

    cur.execute(

        "INSERT INTO [Group] VALUES (?,?)",

        (ii, f"group\_name:{ii}"),

    )

conn.commit()

gender: str = "male"

for ii in range(1, 11):

    if gender == "male":

        gender = "female"

    else:

        gender = "male"

    cur.execute(

        "INSERT INTO [Contacts] (name, family, age, gender, email, phone ) VALUES (?,?,?,?,?,?)",

        (f"name:{ii}", f"familiy:{ii}", ii, gender, f"email:{ii}", ii),

    )

conn.commit()

print("SORT CONTACTS BY GENDER:")

cur.execute("SELECT \* FROM [Contacts] ORDER BY gender ASC ")

for row in cur:

    print(row)

print("get CONTACTS BY OLDEST/YOUNGEST:")

cur.execute("SELECT MAX(age) as 'max', MIN(age) as 'min' FROM [Contacts] ")

for row in cur:

    print(row)

print("SEARCH CONTACTS BY PHONE:")

cur.execute("SELECT \* FROM [Contacts] WHERE phone=5 ")

for row in cur:

    print(row)

cur.execute("UPDATE [Contacts] SET phone=0 WHERE contact\_id =1 ")

conn.commit()

print("UPDATE PHONE CONTACTS FROM ROW No:1:")

cur.execute("SELECT \* FROM [Contacts] WHERE contact\_id =1 ")

for row in cur:

    print(row)

cur.execute("DELETE FROM [Contacts] WHERE contact\_id =7")

conn.commit()

print("DELETED RECORDS WITHOUT ROW 7:")

cur.execute("SELECT \* FROM [Contacts] ")

for row in cur:

    print(row)

cur.close()

"""

for ii in range(1, 10):

    cur.execute(

        "INSERT INTO Company (number, name , age , address , payroll) VALUES (?, ?, ?, ?, ?)",

        (ii, f"name:{ii}", ii, f"address:{ii}", ii),

    )

conn.commit()

print("INSERT RECORDS:")

cur.execute("SELECT \* FROM Company")

for row in cur:

    print(row)

cur.execute("UPDATE Company SET payroll = 100 WHERE number = 1")

cur.execute("UPDATE Company SET address = 'changed address' WHERE number = 4")

conn.commit()

print("UPDATE RECORDS:")

cur.execute("SELECT \* FROM Company")

for row in cur:

    print(row)

cur.execute("DELETE FROM Company WHERE number = 2")

conn.commit()

print("DELETED RECORDS:")

cur.execute("SELECT \* FROM Company")

for row in cur:

    print(row)

print("MAX , MIN , AVG RECORDS:")

cur.execute(

    "SELECT max(payroll) as 'Max payroll', min(payroll) as 'Min payroll', AVG(payroll) FROM Company"

)

for row in cur:

    print(row)

print("PEOPLE ABOVE AVG RECORDS:")

cur.execute(

    "SELECT name FROM Company WHERE payroll > (select AVG(payroll) from Company)"

)

for row in cur:

    print(row)

cur.close()

conn = sqlite3.connect('music.sqlite')

cur = conn.cursor()

cur.execute('DROP TABLE IF EXISTS Tracks')

cur.execute('CREATE TABLE Tracks (title TEXT, plays INTEGER)')

conn.close()

cur.execute("INSERT INTO Tracks (title, plays) VALUES (?, ?)", ("Thunderstruck", 20))

cur.execute("INSERT INTO Tracks (title, plays) VALUES (?, ?)", ("My Way", 15))

conn.commit()

print("Tracks:")

cur.execute("SELECT title, plays FROM Tracks")

for row in cur:

    print(row)

cur.execute("DELETE FROM Tracks WHERE plays < 100")

conn.commit()

cur.close()

conn = sqlite3.connect("music.sqlite")

cur = conn.cursor()

cur.execute("DROP TABLE IF EXISTS Tracks")

cur.execute("CREATE TABLE Tracks (title TEXT, plays INTEGER)")

conn.close()

"""

SORT CONTACTS BY GENDER:

(1, 'name:1', 'familiy:1', 1, 'female', 'email:1', '1')

(3, 'name:3', 'familiy:3', 3, 'female', 'email:3', '3')

(5, 'name:5', 'familiy:5', 5, 'female', 'email:5', '5')

(7, 'name:7', 'familiy:7', 7, 'female', 'email:7', '7')

(9, 'name:9', 'familiy:9', 9, 'female', 'email:9', '9')

(2, 'name:2', 'familiy:2', 2, 'male', 'email:2', '2')

(4, 'name:4', 'familiy:4', 4, 'male', 'email:4', '4')

(6, 'name:6', 'familiy:6', 6, 'male', 'email:6', '6')

(8, 'name:8', 'familiy:8', 8, 'male', 'email:8', '8')

(10, 'name:10', 'familiy:10', 10, 'male', 'email:10', '10')

get CONTACTS BY OLDEST/YOUNGEST:

(10, 1)

SEARCH CONTACTS BY PHONE:

(5, 'name:5', 'familiy:5', 5, 'female', 'email:5', '5')

UPDATE PHONE CONTACTS FROM ROW No:1:

(1, 'name:1', 'familiy:1', 1, 'female', 'email:1', '0')

DELETED RECORDS WITHOUT ROW 7:

(1, 'name:1', 'familiy:1', 1, 'female', 'email:1', '0')

(2, 'name:2', 'familiy:2', 2, 'male', 'email:2', '2')

(3, 'name:3', 'familiy:3', 3, 'female', 'email:3', '3')

(4, 'name:4', 'familiy:4', 4, 'male', 'email:4', '4')

(5, 'name:5', 'familiy:5', 5, 'female', 'email:5', '5')

(6, 'name:6', 'familiy:6', 6, 'male', 'email:6', '6')

(8, 'name:8', 'familiy:8', 8, 'male', 'email:8', '8')

(9, 'name:9', 'familiy:9', 9, 'female', 'email:9', '9')

(10, 'name:10', 'familiy:10', 10, 'male', 'email:10', '10')

**Допълнителни задачи:**

1. Напишете програма на Python, която създава таблица и записва 10 реда в нея. Програмата да извежда тези редове на екрана.
2. Напишете програма на Python, която записва в SQLite таблица 10 записа от списък (list).
3. Напишете програма на Python, която въвежда в таблица 10 записа, данните в които се въвеждат от потребител.
4. Напишете програма на Python, която извежда броя на редовете в SQLite таблица.
5. Напишете програма на Python, която редактира една стойност от дадена колона в таблица. Програмата да извежда всички редове преди и след редактирането.
6. Напишете програма на Python, която редактира всички стойности от дадена колона в таблица
7. Напишете програма на Python, която изтрива даден ред от таблица.
8. Напишете програма на Python, която създава архив (backup) на дадена SQLite база.