Sqlite3

Տվյալների բազաների մասին

Տվյալների բազան տվյալների հավաքածու է, որը պահվում է տվյալ սխեմայի համաձայն:

Դասընթացի ընթացքում դուք սովորաբար տվյալները պահում էիք տեքստային ֆայլում՝ դա կարող է լինել TXT ֆայլ՝ տողերի ցանկով, կամ JSON ֆայլ՝ ավելի բարդ կառուցվածքով: Նման ձևաչափերը լավ են պահեստավորման համար, բայց ոչ արդյունավետ օգտագործման համար: Օրինակ՝ կոնկրետ գրառում գտնելու համար, վատագույն դեպքում, պետք է ամբողջ ֆայլը կարդալ: Իսկ JSON-ի հետ աշխատելու համար հաճախ պետք է ամբողջությամբ փոխել ֆայլը՝ սկզբում կարդալ տեղեկատվությունը, հետո փոխել այն, հետո նորից գրել:

Արտադրողականությունը բարելավելու համար ստեղծեցին հատուկ կազմակերպված տվյալների բազաներ, որոնք թույլ են տալիս առավելագույն արդյունավետությամբ տեղադրել, ջնջել, խմբագրել և կարդալ գրառումները: Դրանք օգտագործվում են ամենուր, ուստի շատ հավանական է, որ հանդիպեք դրանց: Եվ որպեսզի իրական աշխատանքային առաջադրանքներ կատարելիս տվյալների բազայի հետ ծանոթանալը տեղի չունենա, մենք դա կանենք այսօր: Եկեք անցնենք SQLite-ին:

SQLite

SQLite-ը տվյալների բազայի կառավարման համակարգ է (DBMS), որը հիմնված է սուբյեկտների հարաբերությունների վրա: Նման տվյալների բազաները կոչվում են հարաբերական (անգլերեն realtion - հարաբերություն):

Անվան մեջ Lite-ը իսկապես նշանակում է օգտագործման հեշտություն: Եթե այնպիսի հսկաներ, ինչպիսիք են PostgreSQL-ը կամ MySQL-ը, ստեղծում են առանձին սերվեր տվյալների բազան պահպանելու համար, ապա SQLite-ը ստեղծում է միայն մեկ ֆայլ, որով այն աշխատում է: Այն նաև բավականին հեշտ է կարգավորել (իրականում դա նույնիսկ պարտադիր չէ), ուստի մենք կսկսենք դրա հետ տվյալների բազաներին ծանոթանալու մեր ճանապարհորդությունը:

SQL-ի հետ աշխատել sqlite3-ից

Նախքան սկսելը, համոզվեք, որ տեղադրել եք sqlite3: Բացեք հրամանի տողը և մուտքագրեք.

```
~ $sqlite3 -version
```

Եթե սխալ չառաջանա, և ծրագիրը ցուցադրի տարբերակը, ապա ամեն ինչ կարգին է: Հակառակ դեպքում, ներբեռնեք sqlite3-ը պաշտոնական կայքից կամ ձեր տեղադրված փաթեթների կառավարիչից:

Տվյալների բազայի ստեղծում

Այսպիսով, հրամանի տողում մուտքագրեք:

```
~ $sqlite3
```

Ծրագիրը պետք է արտադրի հետևյալ կոդ:

```
SQLite version 3.37.2 2022-01-06 13:25:41
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Մենք տեսնում ենք, որ հիմա մենք կապված չենք որևէ ֆայլի հետ: Եկեք ստեղծենք այն՝ օգտագործելով առաջարկվող հրամանը:

```
sqlite> .open database.db
```

Կամ կարող եք անմիջապես ստեղծել տվյալների բազա՝ գործարկման ժամանակ նշելով դրա անունը:

```
~ $sqlite3 database.db
```

.help գրելով՝ կարող եք տեսնել առկա հրամանները

```
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
```

Եկեք ստեղծենք առաջին աղյուսակը:

Սեղանի ստեղծում

Մեր աղյուսակը կկոչվի ուսանողներ, որտեղ մենք կպահենք ուսանողների մասին տեղեկությունները՝ նրանց id, անունն ու ազգանունը:

```
sqlite> CREATE TABLE students(
   ...> id INTEGER PRIMARY KEY AUTOINCREMENT,
   ...> name TEXT NOT NULL,
   ...> surname TEXT NOT NULL
   ...> );
```

INTEGER-ը տվյալների ամբողջ թիվ տեսակ է:

TEXT-ը տողային տվյալների տեսակ է:

NOT NULL նշանակում է, որ դաշտը չի կարող դատարկ լինել:

PRIMARY KEY-ն ասում է, որ այս բանալին կլինի առաջնային, դրա արժեքները եզակի կլինեն ամբողջ աղյուսակի համար:

AUTOINCREMENT նշանակում է, որ երբ նոր ուսանող ստեղծվի, նրա ID-ն կավելանա 1-ով վերջինի համեմատ:

Հիմա եկեք ավելացնենք ուսանողներին:

Ստեղծեք գրառումներ

Գրառում ստեղծելու համար մենք օգտագործում ենք INSERT հայտարարությունը:

```
sqlite> INSERT INTO students (name, surname) VALUES
('Gor', 'Voskanyan');
sqlite> INSERT INTO students (name, surname) VALUES
('Aram', 'Asatryan');
sqlite> INSERT INTO students (name, surname) VALUES
('Inesa', 'Karapetyan');
```

Առաջին փակագծերը ցույց են տալիս այն դաշտերը, որոնց մեջ ավելացնում ենք երկրորդ փակագծերում տեղադրվող արժեքներնը: Այստեղ մենք չենք նշում id - այն կհաշվարկվի ավտոմատ կերպով:

Ինչպե՞ս կարող եմ տեսնել ավելացված արժեքները:

Ընթերցանության գրառումներ

Գրառումները դիտելու համար մենք օգտագործում ենք SELECT հայտարարությունը:

```
sqlite> SELECT * FROM students;
1|Gor|Voskanyan
2|Aram|Asatryan
3|Inesa|Karapetyan
```

* Աստղանիշն ասում է, որ մենք պետք է ստանանք բոլոր դաշտերը ուսանողների աղյուսակից: Եկեք որոշ գեղեցկություն ավելացնենք արդյունքին՝ գործարկելով հետևյալ երկու հրամանները:

```
sqlite> .headers on
sqlite> .mode box
sqlite> SELECT * FROM students;
┌────┬────────┬─────────────┐
│ id │  name  │   surname   │
├────┼────────┼─────────────┤
│ 1  │ Gor    │ Voskanyan   │
│ 2  │ Aram   │ Asatryan    │
│ 3  │ Inesa  │ Karapetyan  │
└────┴────────┴─────────────┘
```

Եկեք հարցումը ավելի բարդացնենք՝ ընտրենք id > 1 ունեցող ուսանողներին և ստանանք միայն id և ազգանվան դաշտերը:

```
sqlite> SELECT id, surname FROM students WHERE id > 1;
┌────┬─────────────┐
│ id │   surname   │
├────┼─────────────┤
│ 2  │ Asatryan    │
│ 3  │ Karapetyan  │
└────┴─────────────┘
sqlite>
```

Եթե մենք ցանկանում ենք ավելի բարդ պայման, մենք կարող ենք օգտագործել AND կամ OR օպերատորները:

```
sqlite> SELECT id, surname FROM students WHERE id < 3 AND surname
like '%yan';
┌────┬────────────┐
│ id │  surname   │
├────┼────────────┤
│ 1  │ Voskanyan  │
│ 2  │ Asatryan   │
└────┴────────────┘
```

Այս կերպ մենք կստանանք բոլոր ուսանողների ազգանունները որոնք ավարտվում են yan-ով և id < 3-ից: Տոկոսային նշանը ցույց է տալիս, որ «yan» վերջավորությունից առաջ կան որոշ նշաններ:

Գրառումների խմբագրում

Ինեսան ամուսնացավ Արամի հետ և որոշեց վերցնել նրա ազգանունը: Եկեք սա շտկենք UPDATE հայտարարությամբ:

```
sqlite> UPDATE students SET surname = 'Asatryan' WHERE id = 3;
sqlite> SELECT * FROM students
   ...> ;
┌────┬──────┬────────────┐
│ id │ name │  surname   │
├────┼──────┼────────────┤
│ 1  │ Gor  │ Voskanyan  │
│ 2  │ Aram │ Asatryan   │
│ 3  │ Inesa│ Asatryan   │
└────┴──────┴────────────┘
```

WHERE-ում մենք նշում ենք այն չափանիշը, որում պետք է փոխվեն SET-ի դաշտերը: Ինեսան ուներ id = 3, ուստի հեշտ է գտնել նրան, օգտագործելով այն և փոխել ազգանունը:

Գրառումների ջնջում

Գոռ Ոսկանյանը խելացի տղա էր, բայց պիտի հեռանար երկրից, ուստի մենք նրան կհեռացնենք սեղանից:

```
sqlite> DELETE FROM students WHERE id = 1;
sqlite> SELECT * FROM students;
```

id	name	surname
2	Aram	Asatryan
3	Inesa	Asatryan

Սա ձեզ անհրաժեշտ SQL հայտարարությունների նվազագույն փաթեթն է: Հրամանի տողում աշխատելն իհարկե հետաքրքիր է, բայց եկեք օգտագործենք տվյալների բազան Python-ով գրված հավելվածում: Sqlite3-ից դուրս գալու համար հարկավոր է սեղմել Ctrl + D կամ գրել .quit:

Python-ից SQL-ի հետ աշխատելը

Այս բաժնում մենք կանդրադառնանք տվյալների բազայի հետ աշխատելու հիմնական մեթոդներին՝ օգտագործելով sqlite3 մոդուլը:

Միացում տվյալների բազայի հետ

Նախ, եկեք ներմուծենք sqlite3 մոդուլը:

database.py

```
1  import sqlite3
2
```

Տվյալների բազային միանալու համար օգտագործեք connect մեթոդը:

```
3  conn = sqlite3.connect('database.db')
4  ...
5  conn.close()
```

Սխալների դեպքում ճիշտ անջատման համար ավելի լավ է նախընտրել համատեքստի կառավարիչը(context manager):

```
with sqlite3.connect('database.db') as conn:
```

Հարցումների կատարում

SQL-ն օգտագործելու և հարցումների արդյունքներ ստանալու համար անհրաժեշտ է կուրսոր: Սա հատուկ օբյեկտ է, որը ստեղծվում է cursor() մեթոդով:

execute() մեթոդը կատարում է փոխանցված SQL հարցումը: Օգտագործելով fetchall() մենք կարող ենք ստանալ բոլոր գրառումները՝ որոնք այս հարցումը վերադարձրել է մեզ, որպես tuples-ների list:

database.py

```
import sqlite3

conn = sqlite3.connect('database.db')
...
conn.close()

with sqlite3.connect('database.db') as conn:
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM students;")
    print(cursor.fetchall())
    # [(2, 'Aram', 'Asatryan'), (3, 'Inesa', 'Asatryan')]
```

Կարևոր: Գրառումները ավելացնելուց, փոխելուց կամ ջնջելուց հետո դուք պետք է փոփոխություններ կատարեք՝ օգտագործելով Connection օբյեկտի commit մեթոդը.

conn.commit()

Հակառակ դեպքում դրանք չեն գրանցվի տվյալների բազայում: