

Database

講者：Isaac

Outline

- ▶ Database

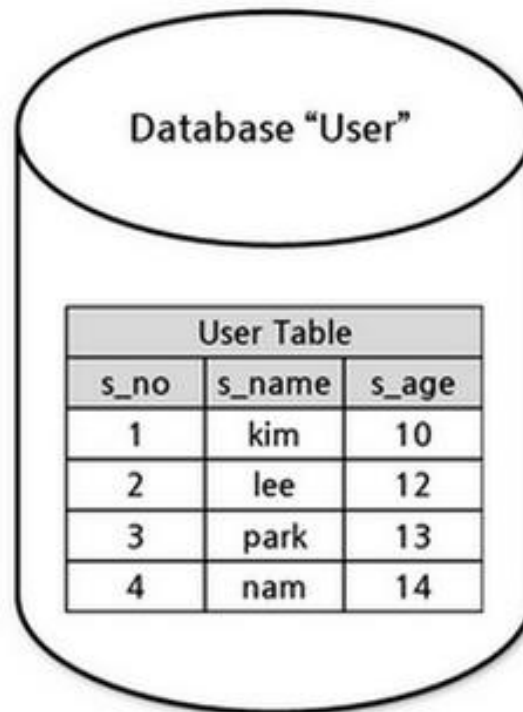


Database



Database

- ▶ Database stores many table
- ▶ Each table stores data



Database terminology

The diagram illustrates database terminology using a table and callouts. The table has five columns: Position Title, Education Requirements, Functional Area, Max Pay, and Min Pay. The rows represent records. Callouts identify the components: 'Column (field)' points to the Education Requirements column; 'Row (record)' points to the Recruiter row; 'Data Value' points to the 140,000 value in the Max Pay column; and 'Table (object)' points to the entire table structure.

Position Title	Education Requirements	Functional Area	Max Pay	Min Pay
Executive Assistant	Associate degree	Human Resources	60,000	40,000
Recruiter	Bachelor's degree	Human Resources	110,000	85,000
SW Engineer	Bachelor's degree	Engineering	140,000	110,000
SQA Engineer	Bachelor's degree	Engineering	140,000	110,000

Database

- ▶ **Database API in Python**

- ▶ DB-API(PEP 249) describe specification to provide Database access for Python application.
- ▶ Ex: MySQL, Oracle, SQLite...
- ▶ Interact diagram:



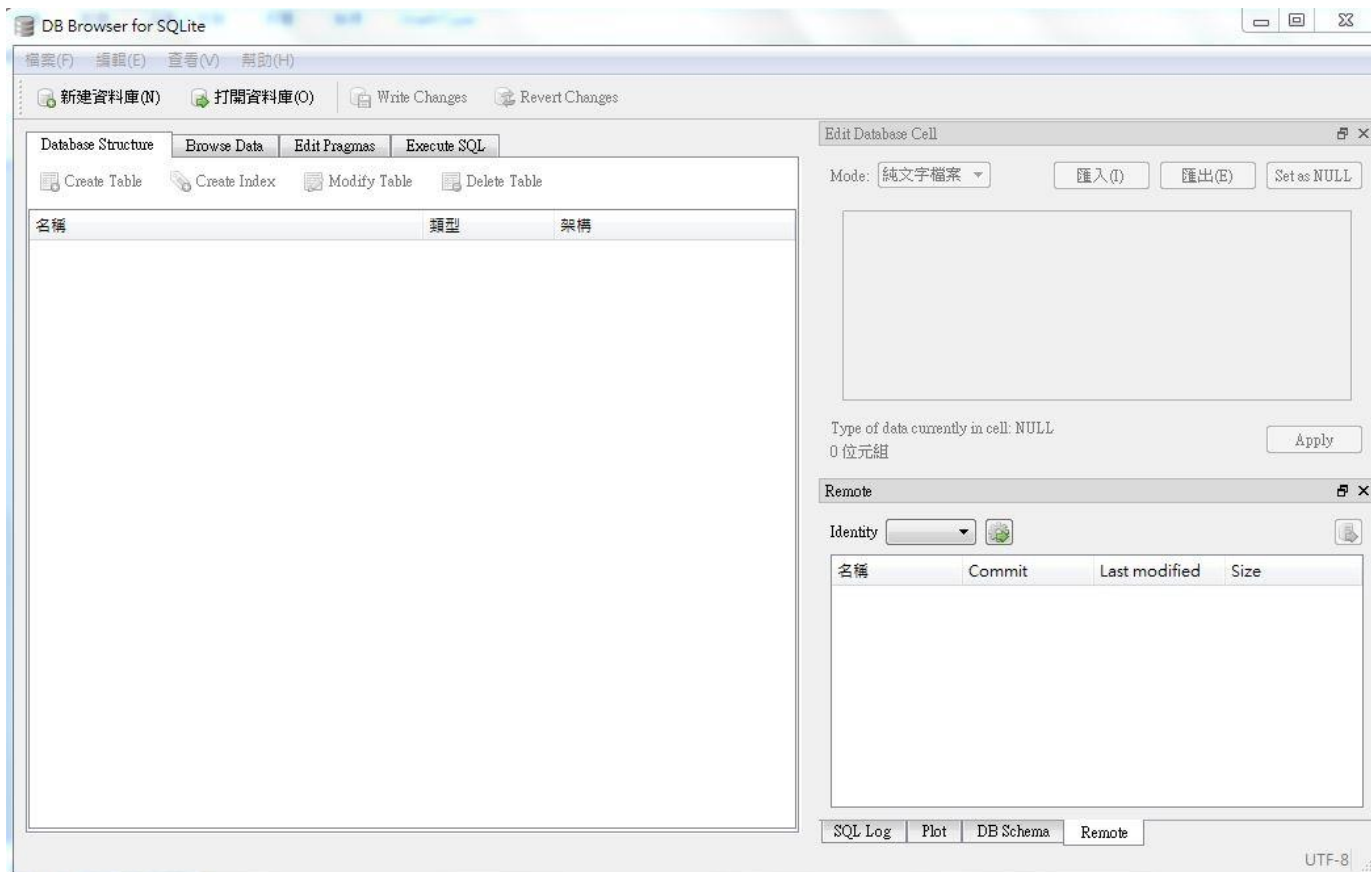
SQLite

- ▶ Relational database management system
- ▶ Embedded database software for local/client storage
- ▶ Use SQL to manipulate



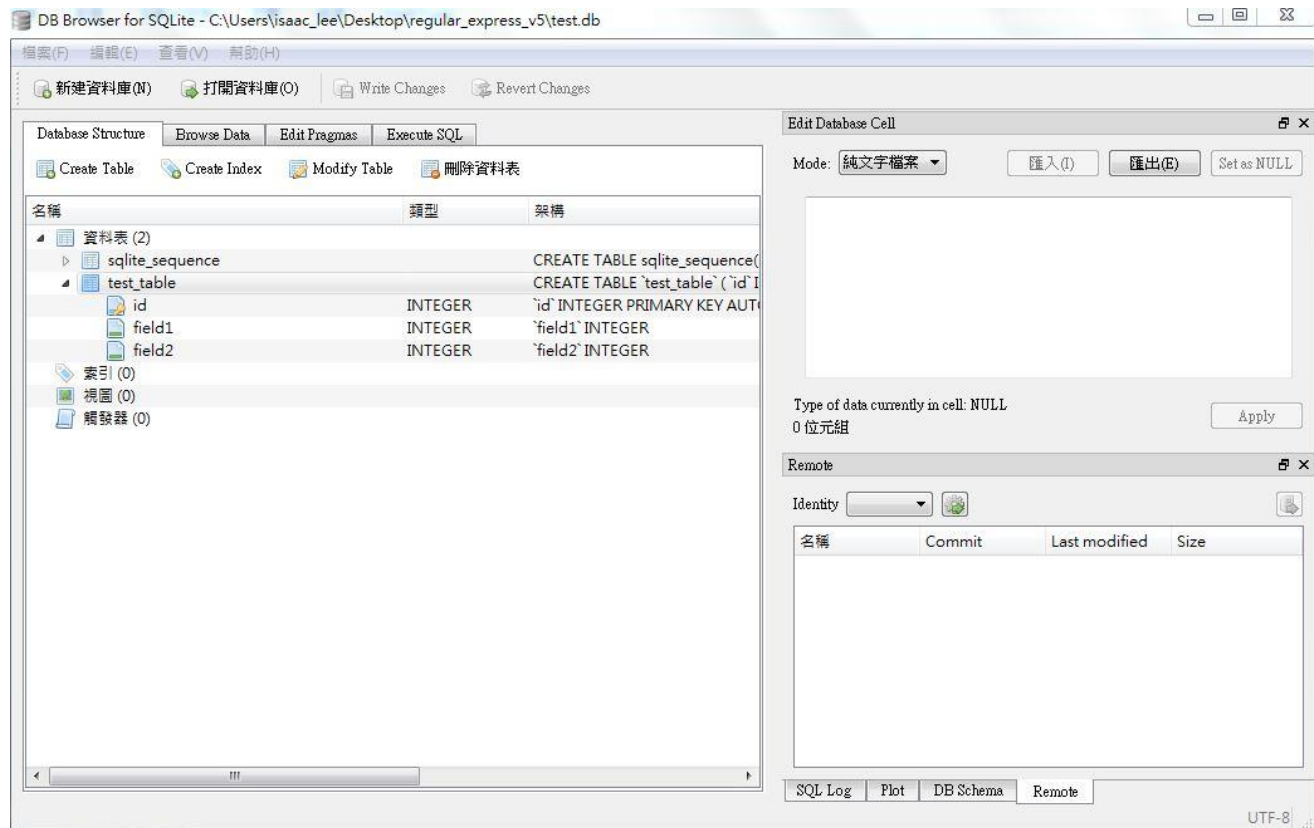
SQLite Installation

- ▶ Download and install sqlitebrowser from
 - ▶ <http://sqlitebrowser.org/>



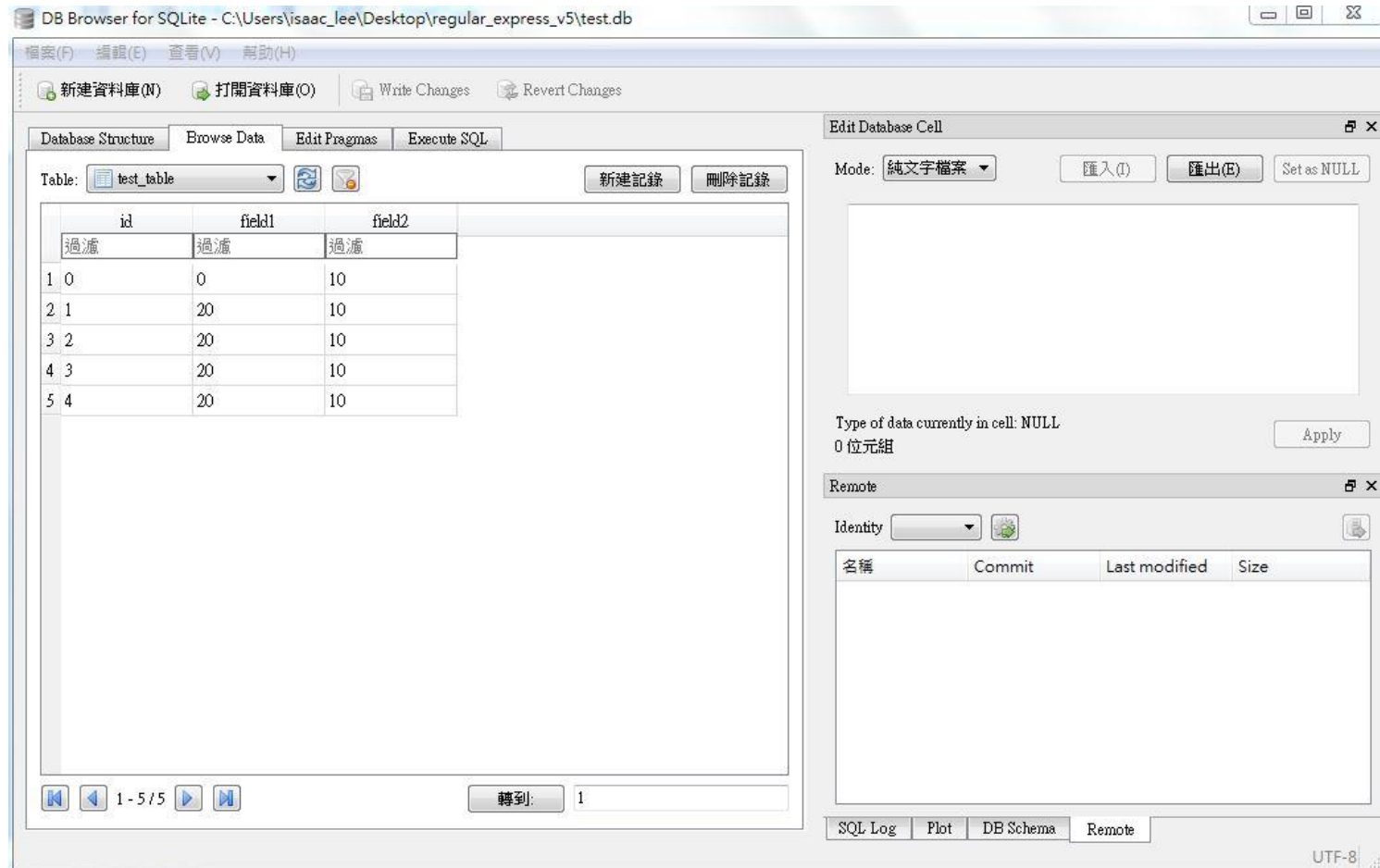
SQLite

- ▶ Create a database
- ▶ Create some table



SQLite

► Add values into the table



SQL

- ▶ SQL is a standard language for storing, manipulating and retrieving data in databases
 - ▶ Can be used in SQLite, MySQL, SQL Server, MS Access, etc.....
 - ▶ Common cmd: CREATE, SELECT, INSERT, UPDATE...



SQL

SQLite Operation in Python

- ▶ **Install SQLite module:**
 - ▶ `pip install sqlite3`
- ▶ **Import SQLite module in Python**
 - ▶ `import sqlite3`
- ▶ **Connect database**
 - ▶ `sqlite3.connect('<DatabaseName>.db')`
- ▶ **Set cursor**
 - ▶ `cursor()`

SQLite Operation in Python

- ▶ **Execute SQL cmd**
 - ▶ `execute('<SQL cmd>')`
- ▶ **Put modifications into the file**
 - ▶ `commit()`
- ▶ **Close the connection to database**
 - ▶ `close()`

SQLite

► Example

- Standard operation programming flow.

```
import sqlite3  
import os
```

Import SQLite module

```
folder_path = os.getcwd()
```

```
conn = sqlite3.connect(folder_path + '/testDB.db')
```

Build connection to db

```
c = conn.cursor()
```

Set cursor

```
c.execute("DROP TABLE IF EXISTS example")  
c.execute("CREATE TABLE example (date text, name text, score text)")
```

SQL cmd

```
conn.commit()
```

Confirm SQL cmd

```
conn.close()
```

Close connection to db

SQLite operation

▶ SQL cmd

▶ DROP + EXIST

- ▶ delete table name example if it exists

```
c.execute("DROP TABLE IF EXISTS example")
```

▶ CREATE

- ▶ create table with the following elements

```
c.execute("CREATE TABLE example (date text, name text, score text)")
```

▶ INSERT

- ▶ Put values into the table “example”

```
c.execute("INSERT INTO example VALUES ('2020-04-01', 'Jason', '78')")  
c.execute("INSERT INTO example VALUES ('2018-04-02', 'Mary', '59')")  
c.execute("INSERT INTO example VALUES ('2018-04-03', 'Celine', '99')")
```

SQLite operation

▶ SQL cmd

▶ SELECT

- ▶ Select all data from table name example

```
1 for row in c.execute('SELECT * FROM example'):  
2     print(row)
```

```
('2020-04-01', 'Jason', '78')  
( '2018-04-02', 'Mary', '59')  
( '2018-04-03', 'Celine', '99')
```

▶ SELECT + ORDER BY

- ▶ Select data from table name example in particular order

```
1 for row in c.execute('SELECT * FROM example ORDER BY score'):  
2     print(row)
```

```
('2018-04-02', 'Mary', '59')  
( '2020-04-01', 'Jason', '78')  
( '2018-04-03', 'Celine', '99')
```


SQLite operation

▶ SQL cmd

▶ SELECT + WHERE

- ▶ Select all data from table name example with condition

```
1 for row in c.execute('SELECT * FROM example WHERE score > 60'):  
2     print(row)
```

```
('2020-04-01', 'Jason', '78')  
( '2018-04-03', 'Celine', '99')
```

▶ SELECT + DISTINCT

- ▶ Find the different date data from table name example

```
1 for row in c.execute('SELECT DISTINCT date FROM example'):  
2     print(row)
```

```
('2020-04-01',)  
( '2018-04-02',)  
( '2018-04-03',)
```

SQLite operation

▶ SQL cmd

▶ SELECT + WHERE + AND

- ▶ Select all data from table name example with condition

```
1 for row in c.execute('SELECT * FROM example WHERE score > 60 AND score > 90'):  
2     print(row)
```

```
('2018-04-03', 'Celine', '99')
```

▶ SELECT + WHERE + OR

- ▶ Find the different date data from table name example

```
1 for row in c.execute('SELECT * FROM example WHERE score < 60 OR score > 90'):  
2     print(row)
```

```
('2018-04-02', 'Mary', '59')  
( '2018-04-03', 'Celine', '99')
```

SQLite operation

▶ SQL cmd

▶ UPDATE + SET + WHERE

- ▶ Update specific data from table name example with condition

```
1 c.execute("UPDATE example SET score = 60 WHERE name = 'Mary' ")
2 conn.commit()
3
4 for row in c.execute('SELECT * FROM example'):
5     print(row)
6
```

```
('2020-04-01', 'Jason', '78')
('2018-04-02', 'Mary', '60')
('2018-04-03', 'Celine', '99')
```