SQLite

Открытие базы данных

```
import sqlite3


conn = sqlite3.connect(sqlite_file)
c = conn.cursor()
…
conn.commit()
conn.close()
```

Создание таблицы

```
# Creating a new SQLite table with 1 column
c.execute('CREATE TABLE {tn} ({nf} {ft})'\
        .format(tn=table_name1, nf=new_field,
ft=field_type))


# Creating a second table with 1 column and set it as
PRIMARY KEY
# note that PRIMARY KEY column must consist of unique
values!
c.execute('CREATE TABLE {tn} ({nf} {ft} PRIMARY KEY)'\
        .format(tn=table_name2, nf=new_field,
ft=field_type))
```

Добавление столбцов

```
c.execute("ALTER TABLE {tn} ADD COLUMN '{cn}' {ct}"\
        .format(tn=table_name, cn=new_column1,
ct=column_type))


# B) Adding a new column with a default row value
c.execute("ALTER TABLE {tn} ADD COLUMN '{cn}' {ct}
DEFAULT '{df}'"\
        .format(tn=table_name, cn=new_column2,
ct=column_type, df=default_val))
```

Добавление и изменение записей

```
# A) Inserts an ID with a specific value in a second column
try:
    c.execute("INSERT INTO {tn} ({idf}, {cn}) VALUES (123456, 'test')".\
        format(tn=table_name, idf=id_column, cn=column_name))
except sqlite3.IntegrityError:
    print('ERROR: ID already exists in PRIMARY KEY column {}'.format(id_column))

# B) Tries to insert an ID (if it does not exist yet)
# with a specific value in a second column
c.execute("INSERT OR IGNORE INTO {tn} ({idf}, {cn}) VALUES (123456, 'test')".\
        format(tn=table_name, idf=id_column, cn=column_name))

# C) Updates the newly inserted or pre-existing entry
c.execute("UPDATE {tn} SET {cn}=('Hi World') WHERE {idf}=(123456)".\
        format(tn=table_name, cn=column_name, idf=id_column))
```

Создание индекса; уникальность

```
# Adding a new column and update some record
c.execute("ALTER TABLE {tn} ADD COLUMN '{cn}' {ct}"\
        .format(tn=table_name, cn=new_column, ct=column_type))
c.execute("UPDATE {tn} SET {cn}='sebastian_r' WHERE {idf}=123456".\
        format(tn=table_name, idf=id_column, cn=new_column))

# Creating an unique index
c.execute('CREATE INDEX {ix} on {tn}({cn})'\
        .format(ix=index_name, tn=table_name, cn=new_column))

# Dropping the unique index
# E.g., to avoid future conflicts with update/insert functions
c.execute('DROP INDEX {ix}'.format(ix=index_name))
```

Выбор записей

```
# 1) Contents of all columns for row that match a certain value in 1 column
c.execute('SELECT * FROM {tn} WHERE {cn}="Hi World"'.\
        format(tn=table_name, cn=column_2))
all_rows = c.fetchall()
print('1):', all_rows)

# 2) Value of a particular column for rows that match a certain value in column_1
c.execute('SELECT ({coi}) FROM {tn} WHERE {cn}="Hi World"'.\
        format(coi=column_2, tn=table_name, cn=column_2))
all_rows = c.fetchall()
print('2):', all_rows)
```

Выбор записей

```
# 3) Value of 2 particular columns for rows that match a certain value in 1 column
c.execute('SELECT {coi1},{coi2} FROM {tn} WHERE {coi1}="Hi World"'.\
        format(coi1=column_2, coi2=column_3, tn=table_name, cn=column_2))
all_rows = c.fetchall()
print('3):', all_rows)

# 4) Selecting only up to 10 rows that match a certain value in 1 column
c.execute('SELECT * FROM {tn} WHERE {cn}="Hi World" LIMIT 10'.\
        format(tn=table_name, cn=column_2))
ten_rows = c.fetchall()
print('4):', ten_rows)
```

Выбор записей

```
# 5) Check if a certain ID exists and print its
column contents
c.execute("SELECT * FROM {tn} WHERE {idf}={my_id}".\
        format(tn=table_name, cn=column_2,
idf=id_column, my_id=some_id))
id_exists = c.fetchone()
if id_exists:
    print('5): {}'.format(id_exists))
else:
    print('5): {} does not exist'.format(some_id))
```

SQL-инъекции

SQL-инъекции

```
# Неправильно:
c.execute("SELECT * FROM {tn} WHERE
{idf}={my_id}".\
        format(tn=table_name, cn=column_2,
idf=id_column, my_id=some_id))


# Правильно:
c.execute("SELECT * FROM {tn} WHERE {idf}=?".\
        format(tn=table_name, cn=column_2,
idf=id_column), (123456,))
```

Работа с датой и временем

```
# update row for the new current date and time column, e.g., 2014-03-06 16:26:37
c.execute("UPDATE {tn} SET {cn}=TIME('now') WHERE {idf}='some_id1'"\
          .format(tn=table_name, idf=id_field, cn=time_col))


# update row for the new current date and time column, e.g., 2014-03-06 16:26:37
c.execute("UPDATE {tn} SET {cn}=(CURRENT_TIMESTAMP) WHERE {idf}='some_id1'"\
          .format(tn=table_name, idf=id_field, cn=date_time_col))
```

Работа с датой и временем

```
# 4) Retrieve all IDs of entries between 2 date_times
c.execute("SELECT {idf} FROM {tn} WHERE {cn} BETWEEN '2013-03-06 10:10:10' AND '2015-03-06 10:10:10'".\
    format(idf=id_field, tn=table_name, cn=date_time_col))
all_date_times = c.fetchall()
print('4) all entries between ~2013 - 2015:', all_date_times)


# 5) Retrieve all IDs of entries between that are older than 1 day and 12 hrs
c.execute("SELECT {idf} FROM {tn} WHERE DATE('now') - {dc} >= 1 AND DATE('now') - {tc} >= 12".\
    format(idf=id_field, tn=table_name, dc=date_col, tc=time_col))
all_1day12hrs_entries = c.fetchall()
print('5) entries older than 1 day:', all_1day12hrs_entries)
```

Работа с датой и временем

```
# Разница времен в часах
SELECT (STRFTIME('%s','2014-03-14 14:51:00') -
STRFTIME('%s','2014-03-16 14:51:00'))
 / -3600

# Разница с текущим временем в часах
SELECT (STRFTIME('%s',DATETIME('now')) -
STRFTIME('%s','2014-03-15 14:51:00')) / 3600
```

Прочие драйверы баз данных

- mysqlclient;
- Cx_Oracle;
- И т. д.

Функции connect и execute работают так же, как в SQLite в 90% других баз данных.