15.13: USING JOIN TO RETRIEVE DATA

Contributed by Chuck Severance
Clinical Associate Professor (School of Information) at University of Michigan

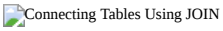
Now that we have followed the rules of database normalization and have data separated into two tables, linked together using primary and foreign keys, we need to be able to build a `SELECT` that reassembles the data across the tables.

SQL uses the `JOIN` clause to reconnect these tables. In the `JOIN` clause you specify the fields that are used to reconnect the rows between the tables.

The following is an example of a `SELECT` with a `JOIN` clause:

```
SELECT * FROM Follows JOIN People
    ON Follows.from_id = People.id WHERE People.id = 1
```

The `JOIN` clause indicates that the fields we are selecting cross both the `Follows` and `People` tables. The `ON` clause indicates how the two tables are to be joined: Take the rows from `Follows` and append the row from `People` where the field `from_id` in `Follows` is the same the `id` value in the `People` table.


Connecting Tables Using JOIN

Connecting Tables Using JOIN

The result of the JOIN is to create extra-long "metarows" which have both the fields from `People` and the matching fields from `Follows`. Where there is more than one match between the `id` field from `People` and the `from_id` from `People`, then JOIN creates a metarow for *each* of the matching pairs of rows, duplicating data as needed.

The following code demonstrates the data that we will have in the database after the multi-table Twitter spider program (above) has been run several times.

```
import sqlite3

conn = sqlite3.connect('friends.sqlite')
cur = conn.cursor()

cur.execute('SELECT * FROM People')
count = 0
print('People:')
for row in cur:
    if count < 5: print(row)
    count = count + 1
print(count, 'rows.')

cur.execute('SELECT * FROM Follows')
count = 0
print('Follows:')
for row in cur:
    if count < 5: print(row)
    count = count + 1
print(count, 'rows.')

cur.execute('''SELECT * FROM Follows JOIN People
            ON Follows.to_id = People.id
            WHERE Follows.from_id = 2''')
count = 0
print('Connections for id=2:')
for row in cur:
    if count < 5: print(row)
    count = count + 1
print(count, 'rows.')

cur.close()

# Code: http://www.py4e.com/code3/twjoin.py
```

In this program, we first dump out the `People` and `Follows` and then dump out a subset of the data in the tables joined together.

Here is the output of the program:

```
python twjoin.py
People:
(1, 'drchuck', 1)
(2, 'opencontent', 1)
(3, 'lhawthorn', 1)
(4, 'steve_coppin', 0)
(5, 'davidkocher', 0)
55 rows.
Follows:
(1, 2)
(1, 3)
(1, 4)
(1, 5)
(1, 6)
60 rows.
Connections for id=2:
(2, 1, 1, 'drchuck', 1)
(2, 28, 28, 'cnxorg', 0)
(2, 30, 30, 'kthanos', 0)
(2, 102, 102, 'SomethingGirl', 0)
(2, 103, 103, 'ja_Pac', 0)
20 rows.
```

You see the columns from the `People` and `Follows` tables and the last set of rows is the result of the `SELECT` with the `JOIN` clause.

In the last select, we are looking for accounts that are friends of "opencontent" (i.e., `People.id=2`).

In each of the "metarows" in the last select, the first two columns are from the `Follows` table followed by columns three through five from the `People` table. You can also see that the second column (`Follows.to_id`) matches the third column (`People.id`) in each of the joined-up "metarows".