目次へ

SQLを使ってみる(sqlite3モジュール)

前回使ってみたshelveモジュールは便利なものでした。入っているデータを順番に全部眺めたりしなくても、キーさえ決まれば一発ですばやくデータを取り出せるという優れものです。しかも、データの値としてリストでも辞書でも入れることができるから、制約が少なくて間違える要素が少ないのもよかったですね。

今回のsqlite3モジュールも、ある種のデータベースをファイルに格納して再利用できるという意味では似たような機能のものだといえますが、もうちょっと込み入ったデータの操作ができます。その分、扱い方もまた手順がちょっとばかり込み入っていて堅苦しいんですけど。

- sqliteは、通常「えすきゅーらいと」と読むようです。なんで「3」がついてるのかといわれれば、sqliteってのはもともとpythonとは別個のものとして開発されていて、そのバージョン3が現在は広く使われているからです。そのくらいの理解でいいんじゃないかなあ。

pythonを初心者が学ぶ上でsqlite3モジュールが重要か、といわれれば、実はそうでもないかなと思います。あえてそんなものをここで紹介してみようと思うのは、SQLというものの存在を知ってもらいたいからに他なりません。都合のよいことに、最近のバージョンのPythonでは、SQLで遊べるこんなモジュールまでが標準配布に含まれているんですね。便利な世の中になったのう。

sqlite3は、リレーショナルデータベース(RDB)と呼ばれる機能を実現するためのプログラムのひとつです。リレーショナル云々ってのが詳しくはどういうものなのか、うまく説明する力量は筆者にはありません。テーブルってものがあって、フィールドってものがあって、そこに値を入れたり出したりするのにSQLっていう言語を使うやつが、RDBってのを表面的に説明したときの感じですよ。そのくらいの理解でも筆者はなんとかやっていけています。

RDBというものを実現するためのソフトウェアは他にも色々ありますよ。Oracle、SQLServer、DB2、PostgreSQL、MySQL、等々。あ、Accessもだな。いくつかは聞いたことがあるんじゃないでしょうか。使うのにお金がいるものや、オープンソースとして原則無料で使えるものなど、様々です。これらすべてのRDBソフトウェアもまた、SQLを使ってデータを操作できることが基本的な特徴です。ほとんどどれも、pythonからアクセスできますよ。標準配布じゃない追加ソフトウェアをインストールする必要はあるんですけどね。

sqlite3は、データベースをファイルの形で管理して、それを直接操作します。これはsqlite3の独特の点で、他の多くのRDBソフトウェアは、直接データファイルを管理するのは「サーバー」と呼ばれる中心部のソフトだけで、のこりの利用者（クライアント）はサーバーにネットワーク経由でSQLの実行依頼を出し、結果のデータが得られることだけを期待します。この方式だと、たくさんのコンピュータでひとつのデータベースを共有して仕事ができますから、こっちのほうが普通です。sqlite3は、この意味では「ひとり用」RDBって感じでしょうか。

SQLそのものの説明に入る前に、もうちょっとRDBの話をします。RDB用ソフトウェアの中にも色々と基本性能に差があって、有名どころではOracleなんかが「高級、高性能（、高価格）」というイメージをもたれているようですね。高性能ってのはどんなところかというと、扱うデータの量がものすごく大量になっても安定した性能を発揮できるとか、複雑なデータ検索をすばやく行えるとか、複数の利用者が寄ってたかってデータベースにアクセスしても大丈夫とか、複雑な権限管理ができるとか、滅多に壊れないとか、万一壊れかけてもすぐ直るとか、そんなのです。大きな会社の売上とか請求とかの伝票データを確実に管理したいとき、なかなか得体の知れないソフトウェアなんかを使う勇気は湧きません。データベースがコケたら、会社の運営がマヒしますからね。大枚をはたいてでも、広く実績のあるRDB（と、もちろん頑丈なコンピュータ）を使いたいものです。こういうのを「基幹データベース」と呼んだりします。止まることが許されない、極めて重要なデータベースです。

で、そんなデータベースも、通常は、やっぱりSQLを使ってデータを読み書きするものなのです。だから、sqlite3モジュールの「お砂場」でちょっと遊んでおくと、そんな重厚長大な基幹データベースにアクセスするときにも似たような方法でできるようになるんですね。もっとも、「お砂場」とはいえ、使い方によってはかなり実用的な仕事もできるもんですよ、sqlite3だけでも。

ってことで、次は、RDBの基本概念である、テーブルとフィールドのことを説明します。なかなかpythonで実際のコードを書いてみるところまで到達せず恐縮ですが。

ひとつのデータベースは、通常、複数のテーブルで成り立っています。テーブルってのはこんな感じのもの。日本語なら「表」なわけだし、まあ直観的にわかるでしょうか。

[売上記録]テーブル

日付	担当者	商品コード	単価	数量
11月1日	Yama	A001	270	1
11月2日	Yama	B020	234	1
11月2日	Oka	A002	2100	1
11月6日	Yama	C100	80	10
11月7日	Hama	D200a	600	2
11月8日	Tera	A001	270	1

エクセルのシートみたいですね。でもエクセルよりは入力ルールに縛りがあります。例えば横方向の項目名（これがフィールドね）ごとに、ここの行には文字列が入るのか数字が入るのか日付が入るのか、文字列なら最高何文字まで入るのか、とかいったことをあらかじめ決める必要があって、それ以外の値を入れることが許されません。数字しか入らないと決めた場所に、'まだ未定で～す' とかそんな文字を入力することはできません。また、ある項目については、かぶった値を入れることができないという決まりを設けることができて、そのフィールドを「キー」と呼んだりします(ちょっと不正確な説明かな。まあいいや！)。キーって名前のあたりは、pythonの辞書とかshelveみたいですね。あと、縦方向のサイズ、つまり入力できるデータの件数は、通常、限度がありません。エクセルの昔のやつでは65000件ちょっとかいった限度がありましたっけ。

横一列分のデータを「レコード」と呼ぶことがあります。この例なら、11月何日に誰々が何を売りました、っていう一件分の伝票がレコードに相当します。

複数のテーブルでひとつのデータベースを表すことが多いのは、たとえば次のようなデータの持ち方をしたりするからです。

[商品]テーブル

商品コード	商品名
A001	何々チョコレート
B001	何々ガム
B020	鉢植え
…	

まず、商品の売上記録というテーブルがあったとする。当然そこには「どの商品が」ということを表す項目（フィールド）があるはずですが、ここに直接商品名を書くことにするとちょっとした表記ミスが起こりやすい。だから商品ごとに「商品コード」をつける決まりにしておいて、売上記録には商品コードだけ書いておくことにする。そのかわり、商品の詳細を別のテーブルとして作っておき、そちらでは商品コードをもとにその名前とか仕入先とか定価とかを記しておけばいい。商品コードは、別の商品とかぶらないように「キー」としておくのが普通でしょうね。こんな感じで、「商品コード」を橋渡しにしてふたつのテーブルが関係をもつようにしておくと、必要なデータは一箇所だけにまとまることになり、都合がいい場合が多いのです。

ある大きな仕事をどんなテーブル構造として表現するのかといった話は、とても奥が深くてこんな場所で語りつくすことはできません。というより、筆者にそんな力はありません。なのでこん

な話はそこそこにして終わっておきます。

さあ、SQLの話に戻ってきました。SQLはデータベース(RDB)を操作するためにpythonとは別に設計された、文字列で表現される呪文です。いくらでも複雑にできますが、基本的なところは簡単です。呪文には大きく種別があって、「新しくテーブルを作る呪文」「テーブルを抹消する呪文」「テーブルにレコードを追加する呪文」「レコードの一部を書き換える呪文」「レコードを削除する呪文」「データベースの利用権限を操作する呪文」など、これだけでもいろいろです。最初にやってみるのは、これらの中でも基本中の基本、「テーブルの中のデータを見る呪文」です。SELECTという文字列で始まるものですから、SELECT系とでも名づけましょうか。

サンプルデータベース

ということで、サンプルデータベースを作っておきました。下のリンクからダウンロードできるファイルを、いつもスクリプトを作るフォルダの中に格納してください。

sales.sqlite3

このファイルはsqlite3モジュールを使って中身を見ることができます。

そしたら、まずは、下のスクリプトをそのままマネするなりコピペするなりして実行し、表示される結果を確かめてみましょう。

```
import sqlite3
con = sqlite3.connect('sales.sqlite3')

sql = 'select * from uriage'
r = con.execute(sql)

for i in r:
  print i
```

こんな結果になりましたか。

```
(u'2010-08-01', u'Tsuru', u'A001', 30, 1)
(u'2010-08-01', u'Tsuru', u'A101', 90, 1)
(u'2010-08-01', u'Tsuru', u'A001', 110, 2)
(u'2010-08-01', u'Higa', u'A101', 70, 9)
(u'2010-08-01', u'Tsuru', u'B002', 180, 1)
(u'2010-08-02', u'Tsuru', u'A102', 150, 10)
(u'2010-08-03', u'Tsuru', u'B003', 120, 6)
(u'2010-08-03', u'Higa', u'A002', 50, 1)
...
```

このスクリプトが何をしたのか、説明します。

最初の import sqlite3 は、今からsqlite3をいじるから、それ用の機能をひとそろい準備するという意味ですね。

で、次の con = … ってのは、ファイルをopenするときとか、shelveを使うときとかと同じノリです。sqlite3ファイルのファイル名を sqlite3.connect という関数で処理すると、それにアクセスするための目印が con 変数に入るのでした。一般に、この操作を、「データベースに接続する」って呼びます。connect だしね。con っていう変数名は、ここでは connection くらいの意味でつけました。

- sqlite3は、データベースに接続するときに、特に認証を要求しません。ちまたの色々なRDBは、こういうときにIDとパスワードとかいった情報も要求するもので、映画なんかでは、これを間違うと部屋中にアラームが鳴ったり、天井から牢屋が降ってきたりするものです。sqlite3でよかったね。

すると、その con の中身（オブジェクト）は、execute という機能（メソッド）を使うことができるようになります。ここにSQLを書いて、実行できるというわけ。簡単ですね。この例ではあらかじめ変数にSQL本文をしまっておいてから使いましたが、別に直接SQLをあらわす文字列をカッコ内に書き込んだって別に構いません。

で、この execute の結果もまた、変数に格納して使います。ここでは簡単に r という名前にしてみました。(execute した結果(result) が入る、くらいの意味合いです。）テキストファイルを開いて読み込むのと比べると、手間がひとつ多いですね。(1) データベースに接続する、(2) その接続に対して、SQLを実行する、(3) その実行結果の中身を見るとデータざくざく、といった感じで。あ、(3)は今から説明するんだ。

- pythonで一般的なデータベースアクセスの手順は、「コネクションを作って、そこからカーソルというものを作って、それにSQLを発行(execute)して…」という風になっていて、いま調べている例では「カーソル」とやらを作る段階を省略しちゃう方法を行っています。一応注記しておきますね。

で、(3)としては、結果の入った変数(ここではr)を、あたかもテキストファイルでも読み込むかのように　for を使って繰り返しながら一行ずつ取り出す、というあんばいです。一行ずつ取り出すと、最初からリストの形にバラけて取り出せるというのが特徴かな。（リストではなくて正確にはタプルだけど、タプルをまだ（！）説明してないね。）

結果一行分をもうちょっとよく見てみましょう。

```
(u'2010-08-01', u'Tsuru', u'A001', 30, 1)
```

u'.....' のついた文字列っぽいものは、ユニコード文字列です。sqlite3には文字列としてユニコード文字列を使うのが普通なので、今回はこういうのが入っています。あまりユニコードの扱いをスッキリと説明しきっていないので恐縮ですが、今のところは普通に文字列だと理解してもよいです。

で、リスト（タプル）の中で右ふたつは、数字です。最初からちゃんとした数字で、足し算なんかに使ってもエラーが起きません。RDBってのはフィールドごとに値のタイプが決まっているので、数字が入ると定められているところは最初から数字で結果が帰ってくるんです。

で、今回実行した 'select * from uriage' っていう文字列が、SQLの本体です。こいつはSELECT系の呪文の中でも一番簡単なやつで、「データベースの中に uriage っていうテーブルが入っているはずだから、とりあえず全部中身を見たい」という意味です。uriage テーブルは、あらかじめ下のようなフィールドで成り立っていると筆者が決めて作っておきました。

- 売上日付(yyyy-mm-dd形式) フィールド名は date
- 販売担当者 フィールド名は tahto

- 商品コード フィールド名は hin_id
- 販売単価 フィールド名は tanka。整数が入る
- 販売個数 フィールド名は su。整数が入る

なので、ここで決めた順にデータが入っていました。フィールド名はここでは表示に使っていませんが、出現順でわかるから、今のところはまあいいでしょ。

sqlを変数に入れている一行を下のようなものに変えて、同じスクリプトを実行してみましょう。

```
sql = 'select date, tahto, su from uriage'
```

```
（実行結果）
(u'2010-08-01', u'Tsuru', 1)
(u'2010-08-01', u'Tsuru', 1)
(u'2010-08-01', u'Tsuru', 2)
(u'2010-08-01', u'Higa', 9)
...
```

今回は、全部のフィールドを出すんじゃなくて、date, tahto, su の三つの項目だけを抜き出すというSQLの唱え方を示してみました。こんな書き方をしたいときに、フィールド名は役に立ちます。列挙する順番を変えれば、表示される結果の順番も変わりますよ。

- ところで、販売担当者は tanto じゃないの？ なんで tahto になってるんだろう、とお考えでしょうか。いやあ、最初に筆者がサンプルデータベースを作るときに入力ミスしちゃって、そのまま引きずっているだけです…

じゃー次はこんなSQLでは。

```
sql = "select * from uriage where date = '2010-11-10'"
```

```
（実行結果）
(u'2010-11-10', u'Sawa', u'B100', 60, 1)
(u'2010-11-10', u'Yama', u'B001', 20, 2)
(u'2010-11-10', u'Tsuru', u'A101', 150, 7)
(u'2010-11-10', u'Sawa', u'B002', 130, 5)
```

特定の条件を満たすレコードだけを抜き出すための、where という指定を加えたやり方です。特に説明しないでも、このくらいなら何となく「ああ、そうやって書くの」と分かるでしょうし、そもそもここではSQLの詳細は説明しません。でも、SQLの「呪文」の唱え方次第で、ちょっと面白いことができるらしい、という予感を覚えたりはしませんか。しませんか、そうですか。

ところで、この売上記録には、商品コードまでしか入っていません。B100とかA101とかって具体的に何よ、とお考えでしょう。このデータベースには商品テーブル(shohin)も一緒に入れておいたので、今度はそいつを見ましょう。

```
sql = "select * from shohin"
```

```
（実行結果）
(u'A001', u'¥u30ad¥u30e3¥u30d9¥u30c4')
(u'A002', u'¥u82f1¥u548c¥u8f9e¥u5178')
(u'A101', u'¥u540d¥u523a¥u30db¥u30eb¥u30c0¥u30fc')
(u'A102', u'¥u8f2a¥u30b4¥u30e0')
(u'B001', u'¥u30d4¥u30f3¥u30dd¥u30f3¥u53f0')
(u'B002', u'¥u306d¥u3058')
(u'B003', u'¥u826f¥u5bdb¥u5168¥u96c6')
(u'B100', u'¥u30a8¥u30a2¥u30af¥u30ea¥u30fc¥u30ca¥u30fc')
```

おっと、日本語が入っていると、リスト（タプルだけどさ）の表示はちょっと期待どおりにいかなくなるんでした。ちょっとスクリプトを書き直して試しなおしましょう。

```
for i in r:
  print i
```

を、

```
for i in r:
  print i[0], i[1]
```

に直せばちゃんと出ます。

```
 （実行結果）
A001 キャベツ
A002 英和辞典
A101 名刺ホルダー
A102 輪ゴム
B001 ピンポン台
B002 ねじ
B003 良寛全集
B100 エアクリーナー
```

品揃えも金額設定もこの例ではなんだかめちゃくちゃですが、まあ、例ということで。ひとつのデータベースファイルの中にいくつもテーブルが入っているんだなあ、というくらいの実感は感じていただけますでしょうか。

- 商品テーブルのほうに単価を入れておかないの？　という疑問を持つかたがいるでしょうが、ここではたまたまそうしていません。データベースの設計次第ですね。きっとこの店は、商品の単価なんかあってないようなものであり、客を見ながら値段を決めるような商売してるんでしょう…

SQLの説明をし始めたらキリがない

この回はとりあえずここまでです。

SQLはSELECT系だけでなく、新しいデータを格納したり、既存のデータを書き直したりといった操作もできます。色々なSQLをマスターするにつれ、データベースの操作は「なんでも」書けるようになっていきます。テーブルの作成そのものだって、SQLで書いたりするのです。SQLの世界のすごい人たちは、驚くべき複雑さのSQL呪文をクールに唱えたりして、まるで灰色のガンダルフか何かのようです。

だから、SQLの書き方をちゃんと勉強しようとする方（または、せざるをえなくなった方）は、参考書を探すと、SQL入門とかSQLウィザードとかそんなのがたぶん見つかるでしょう。ここからはpythonとは別世界の話です。地の底にでも峰の上にでも逝ってらっしゃい。

でも、今回SELECTの初歩の初歩だけやってみて、いわゆるデータベースというものからデータを拝借してきて使うくらいなら結構簡単なんじゃないかと思ってくれると嬉しいです。今回は練習問題とかにしませんが、たとえば会社の「基幹データベース」から必要なデータをもらって、そいつをもとに、例えば今までやってきたような集計とかHTML生成とかのネタに使わせてもら

う、といったような仕事がイメージできるようになってきませんか。データさえもらってこれば、あとは我々の土俵ですからね。スクリプトを書くのがうまくなってきたら、色々できますよ。