APPLICATION OF THE DEVICE DATABASE IN THE PYTHON PROGRAMMING

Tatsuro Nakamura
High Energy Accelerator Research Organization (KEK)
1-1 Oho, Tsukuba, Ibaraki, 305-0801, Japan

Abstract

The Device Database has been developed using the relational database in the KEKB accelerator control system. It contains many kinds of parameters of the devices, mainly magnets and magnet power supplies. The parameters consist of the wiring information, the address of the interfaces, the specification of the hardware, the calibration constants, the magnetic field excitation functions and the any other parameters for the device control. These parameters are necessary not only for constructing EPICS IOC database but also for providing information to the high-level application programs, most of which are written in the script languages such as SAD or Python. Particularly Python is often used to access the Device Database. For this purpose, the Python library module that is designed to handle tabular data of the relational database on memory has been developed. The overview of the library module is reported.

Pythonプログラムにおける機器データベースの利用

１．はじめに

KEKB加速器の制御システムでは電磁石、電源等の機器の情報をリレーショナルデータベースで管理している。特に配線情報、インターフェースアドレス、仕様や特性の各種定数など、制御に必要なパラメータを管理している。これらの情報は主にEPICS IOC databaseの定義ファイルを生成する際に用いられるが、それだけでなく上位計算機のアプリケーションプログラムからも参照されて利用される。これら上位のアプリケーションは主にSAD、Pythonといったスクリプト言語で作成される。特にリレーショナルデータベースの情報を読んで加工するのにPythonを用いる事が多い。そこでリレーショナルデータベースに特有な表形式のデータを扱うのに適したPythonライブラリを開発した。ここではこのライブラリの概要について述べる。

２．システムの概要

2.1 KEKB加速器制御システムとEPICS

KEKB加速器制御システム[1] [2] [3]は骨格となるソフトウェアにEPICS (Experimental Physics and Industrial Control System)[1]を採用している。EPICSは世界規模で多くの研究所が共同で開発・維持しているオープンソースのソフトウェアであり、加速器等の大型実験装置の計算機制御に向いた分散処理を特徴とする。EPICSではIOC (I/O Controller)と呼ばれる計算機にdatabaseを常駐させて制御を行なう。KEKB加速器ではIOCとして主にVMEのボード計算機を使用し、ローカル制御室に約100台を設置している。

[1] http://www.aps.anl.gov/epics/

IOC databaseはrecordと呼ばれる制御単位の集合体である。IOC databaseは定義ファイルを記述することによって作ることができ、標準的な機能であればほとんどプログラミングすること無しに制御システムを構築することも可能である。

2.2 機器データベース

KEKB加速器では2500台を超える電磁石電源があり、また扱う電磁石や電源の特性や使い方もさまざまである。それに伴い、配線情報、インターフェースアドレス、仕様や特性の各種定数、励磁曲線関数など、制御に必要なパラメータが多くある。これらの情報はリレーショナルデータベースに集め管理している。このデータベースを（IOC databaseと区別して）機器データベースと呼ぶ。KEKB加速器では機器データベースのためにOracleを使用している。

電磁石・電源のIOC databaseはこの機器データベースの情報から生成するようにしている。生成過程はいくつかの段階からなるが、主要な部分はOracleのスクリプト言語であるPL/SQLで記述されたプログラムによって行なっている。

機器データベースの最大の目的はこのようにIOC databaseの生成を自動化することであるが、上位層のアプリケーションプログラムからも参照されて利用されることも多い。このようなアプリケーションは特にPythonで書かれることが多い。

2.3 Python

Pythonは汎用のスクリプト言語で、文法が簡明で初心者にも扱いやすい。またリスト、タプル、辞書といった構造化に適したデータ型が用意されていて、複雑な処理も分かり易く記述できる。各種ライブラリも充実しており、効率的なプログラミングが可能である。KEKB加速器では制御アプリケーションの

作成にPythonがSADと共に広く使われている。

2.4 PythonとOracle

Pythonには、さまざまなリレーショナルデータベースにアクセスするためモジュールが各種用意されている。Oracleにアクセスするにはoracledbモジュールを使う。Pythonではデータベースの種類に依らず同じ方法でアクセスできるようライブラリの仕様が統一的に決められているため、プログラムをほとんど変更することなく異なるデータベースにアクセスが可能である。KEKB制御システムでは現在Oracleのほかに PostgreSQLに対しても同様にPythonからアクセスできるようになっている。

リレーショナルデータベースの基本的なデータ構造は表であり、表は行と列からなる2次元配列と見なせる。oracledbモジュールを使うと表にアクセスすることができる。表を読んで得られるデータでは、一つの行が一つのタプルで表され、表はこのタプルを並べたリストとして表されている。

3．機器データベース用ライブラリの導入

oracledbモジュールは多機能で柔軟な処理ができる反面、使い方がやや煩雑な面がある。そこで機器データベースの利用を念頭に置き、目的を絞った補助的なライブラリを導入することにした。制御アプリケーションプログラムから機器データベースを利用する場合は、次のような特徴がある。

A) 専らデータを参照するだけであり、データベースの更新などは行なわないことが多い。
B) アクセス対象のとなる表は比較的小さいものが多い。（大きいものでも数十万行程度）
C) 表をそのまま使うだけでなく複雑な演算処理を行なうことが多い。

演算処理はSQL文で書き表してデータベース・サーバ側で処理することも考えられるが、大多数のユーザはリレーショナルデータベースやSQLに不慣れであるため、Pythonで記述できるほうが望ましい。そこで表全体を一度に読み込み、Pythonプログラム内で演算や加工を行なうこととした。この目的でrdbtoolと名付けたライブラリモジュールを開発した。以下でこのモジュールを紹介する。

4．ライブラリの概要

4.1 表クラス・行クラス

前述したようにoracledbが返す表のデータ構造はタプルのリストになっている。このままでは扱いにくいので、新たなデータ型を定義した。Pythonはオブジェクト指向言語なので、この特徴を生かして表に対応するデータ型は表クラスとして定義した。表クラスのオブジェクトに対する各種の操作はメソッドとして定義している。また、一つの行を表すデータは行クラスとして定義した。

4.2 表の読み込み

データベースアクセスのためにrdbクラスを用意している。典型的な使い方は次のようになる。データベースへの接続まではoracledbモジュールを直接使って行ない、得られた接続オブジェクトを引数にしてrdbクラスのオブジェクトを生成する。rdbクラスではoracledbの機能の多くを引き継いでいるのでそれと同様の使い方もできるが、より簡単な使い方として表名を指定して一つの表全体を読みこむことができるようになっている。また、表名の代わりにSQLのselect文を与えて結果を読み込むこともできる。

4.3 表示メソッド

表クラスと行クラスにはshowメソッドがあり、自身の内容を整形されたテキストとしてプリントすることができる（図1a）。また、showdescメソッドを使うと列の属性情報をプリントすることができる（図1b）。

a) showメソッドの例

```
>>> mgSample.show()
NAME    HW_NAME ATTRIB MAGNET_PS MAGNET_TYPE
HM19P   HM19P   p      HM19P     BT_ZHP
VX03P   VX03P   p      VX03P     BT_ZVP
VX06P   VX06P   p      VX06P     BT_ZVP
VX09P   VX09P   p      VX09P     BT_ZVP
QXF2P_A QXF2P.A p      QXF2P     BT_QAP
QXD5P_K QXD5P.K p      QXD5P     BT_QNP
QWFP_2A QWFP.2A p      QWFP_1_4  BT_QAP
QWDP_1A QWDP.1A p      QWDP_1_3  BT_QAP
B1P     B1P     p      B1P       BT_B1P
B2P_2   B2P.2   p      B2P_1_2   BT_BH1P(TYPE3)
```

b) showdescメソッドの例

```
>>> mgSample.showdesc()
column_name  data_type    not_null disp_size  int_size
NAME         STRING(15)   NOT NULL        15        15
HW_NAME      STRING(15)   NOT NULL        15        15
ATTRIB       STRING(1)    NOT NULL         1         1
MAGNET_PS    STRING(20)                   20        20
MAGNET_TYPE  STRING(15)   NOT NULL        15        15
```

図１：表示メソッドの例

4.4 アクセスメソッド

表クラスはPythonのリストと同様の演算が可能である。例えば番号を指定して一つの行を取り出すことができるし、番号で範囲を指定して表の一部分（スライス）を得ることもできる（図2a,b）。リストと同様にfor文で各行を順に処理することも簡明に記述できる。またappend, extend, reverseといったリストが持つのと同じメソッドを使うこともできる。

行クラスはPythonのタプルと同様に扱うこともできるが、個々の列の要素を取り出すにはむしろ列名で指定できた方が便利である。そこで簡明な記法で特定の列の値を取り出すことができるようにした（図2c）。図に示すように記法には二種類ある。また、表クラスのオブジェクトに対しては同様の記法で特定の列の値をリストとして取り出すことができる（図2d）。こちらも二種類の記法がある。

このように表クラスでは基本的に行は番号（添え字）でアクセスし、列は名前（列名）でアクセスで

きるようになっている。
　行クラスでは指定したいくつかの列だけを含む行を作ることができる（図2e）。また、同様の記法は表クラスでも使うことができ、指定したいくつかの列だけを含む表を作ることができる（図2f）。
　このようにしていくつかの列だけに絞った表を得ることができるが、その中には全く同一の行が複数現れる場合がある。このような重複する行を省いた結果を得たい場合には代わりにdistinctメソッドを使うことで可能となる（図2g）。

4.5 フィルタリングとソート

　表クラスには特定の条件に合致する行のみを選び出すメソッドがある。pickupメソッドはある列が特定の値を取るか否かで選別を行なう（図2h）。より複雑で一般的な条件での選別にはfilterメソッドを使う。filterメソッドに条件判断関数を渡すことで任意の条件での選別を行なうことができる。
　表クラスにはsortメソッドがあり、列名を指定してその列の値の順に行を並べ替えることができる。列名を複数指定することもでき、最初に指定された列名で並べ替えた上で、その列が同じ値のもの同士は次に指定された列名で並べ替えていく（図2i）。

５．まとめ

　KEKB加速器制御システムではリレーショナルデータベースを使った機器データベースを構築し、これよって電磁石・電源等の機器の情報を一元管理している。機器データベースに簡単にアクセスするためにPythonのライブラリモジュールrdbtoolを開発した。これは機器データベースの特徴を生かしon memoryでのデータ処理に特化したもので、SQLとは異なるアプローチでリレーショナルデータベースの基本的データ構造である表をPythonの中で扱うことができる。
　また、rdbtool自体はOracleに限らずリレーショナルデータベース一般に応用できる。KEKB加速器では少しの変更でPostgreSQLにも応用することができた。さらにはリレーショナルデータベースに限らず表の構造を持つデータ一般にも応用が可能であり、今後はこれを拡張・発展させたツールの開発も視野に入れたい。

参考文献

[1] N. Yamamoto et al., “KEKB control system: the present and the future”, Proceedings of the 1999 Particle Accelerator Conference, New York, 29 Mar.-2 Apr. 1999, pp. 343.345
[2] Nobumasa Akasaka et al., “KEKB accelerator control system”, Nucl. Instr. and Meth. A499 (2003) 138.166
[3] Tatsuro Nakamura, et al., “Status of KEKB Accelerators Control System in 2006”, Proceedings of the 3rd Annual Meeting of Particle Accelerator Society of Japan, Sendai, Aug. 2-4, 2006

a) 表の中の一行を取り出す。
```
>>> row = mgSample[0]
>>> row.show()
NAME  HW_NAME ATTRIB MAGNET_PS MAGNET_TYPE
HM19P HM19P   p      HM19P     BT_ZHP
```
b) 表の中のある範囲の行を取り出す。
```
>>> mg1 = mgSample[2:5]
>>> mg1.show()
NAME    HW_NAME ATTRIB MAGNET_PS MAGNET_TYPE
VXO6P   VXO6P   p      VXO6P     BT_ZVP
VXO9P   VXO9P   p      VXO9P     BT_ZVP
QXF2P_A QXF2P.A p      QXF2P     BT_QAP
```
c) 列名を指定し行の中のある列の値を得る。
```
>>> row['NAME']
'HM19P'
>>> row.NAME
'HM19P'
```
d) 表の中のある列の値をリストとして得る。
```
>>> mgSample['NAME']
['HM19P', 'VXO3P', 'VXO6P', 'VXO9P', 'QXF2P_A',
'QXD5P_K', 'QWFP_2A', 'QWDP_1A', 'B1P', 'B2P_2']
>>> mgSample.NAME
['HM19P', 'VXO3P', 'VXO6P', 'VXO9P', 'QXF2P_A',
'QXD5P_K', 'QWFP_2A', 'QWDP_1A', 'B1P', 'B2P_2']
```
e) 選択した列だけの行を作る。
```
>>> row2 = row['NAME','MAGNET_TYPE']
>>> row2.show()
NAME  MAGNET_TYPE
HM19P BT_ZHP
```
f) 選択した列だけの表を作る。
```
>>> mg2 = mg1['NAME','MAGNET_TYPE']
>>> mg2.show()
NAME    MAGNET_TYPE
VXO6P   BT_ZVP
VXO9P   BT_ZVP
QXF2P_A BT_QAP
```
g) 重複行を除き選択した列だけの表を作る。
```
>>> mgtyp = mgSample.distinct('MAGNET_TYPE')
>>> mgtyp.show()
MAGNET_TYPE
BT_ZHP
BT_ZVP
BT_QAP
BT_QNP
BT_B1P
BT_BH1P(TYPE3)
```
h) pickupメソッドによるフィルタリングの例
```
>>> mgqap = mgSample.pickup(MAGNET_TYPE='BT_QAP')
>>> mgqap.show()
NAME    HW_NAME ATTRIB MAGNET_PS MAGNET_TYPE
QXF2P_A QXF2P.A p      QXF2P     BT_QAP
QWFP_2A QWFP.2A p      QWFP_1_4  BT_QAP
QWDP_1A QWDP.1A p      QWDP_1_3  BT_QAP
```
i) sortメソッドを使った並べ替えの例
```
>>> mgSample.sort('MAGNET_TYPE','NAME')
>>> mgSample.show()
NAME    HW_NAME ATTRIB MAGNET_PS MAGNET_TYPE
B1P     B1P     p      B1P       BT_B1P
B2P_2   B2P.2   p      B2P_1_2   BT_BH1P(TYPE3)
QWDP_1A QWDP.1A p      QWDP_1_3  BT_QAP
QWFP_2A QWFP.2A p      QWFP_1_4  BT_QAP
QXF2P_A QXF2P.A p      QXF2P     BT_QAP
QXD5P_K QXD5P.K p      QXD5P     BT_QNP
HM19P   HM19P   p      HM19P     BT_ZHP
VXO3P   VXO3P   p      VXO3P     BT_ZVP
VXO6P   VXO6P   p      VXO6P     BT_ZVP
VXO9P   VXO9P   p      VXO9P     BT_ZVP
```
図２：表クラス、行クラスに対する操作