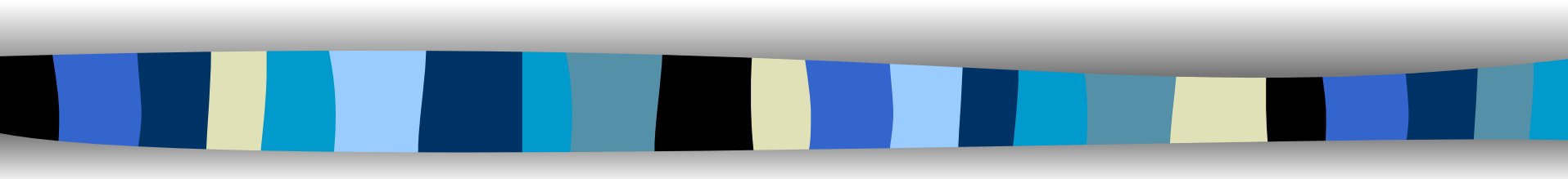
Perl/DBIによる次のステップ

～WWWログファイルだってSQLで操作する！？



川合孝典(Kansai.pm)

2002/2/15

こんなデータはどうします？

```
山田一郎　埼玉,東京(TAB)28
鈴木次郎　静岡,静岡(TAB)32
佐藤花子　神奈川,東京(TAB) 26
```

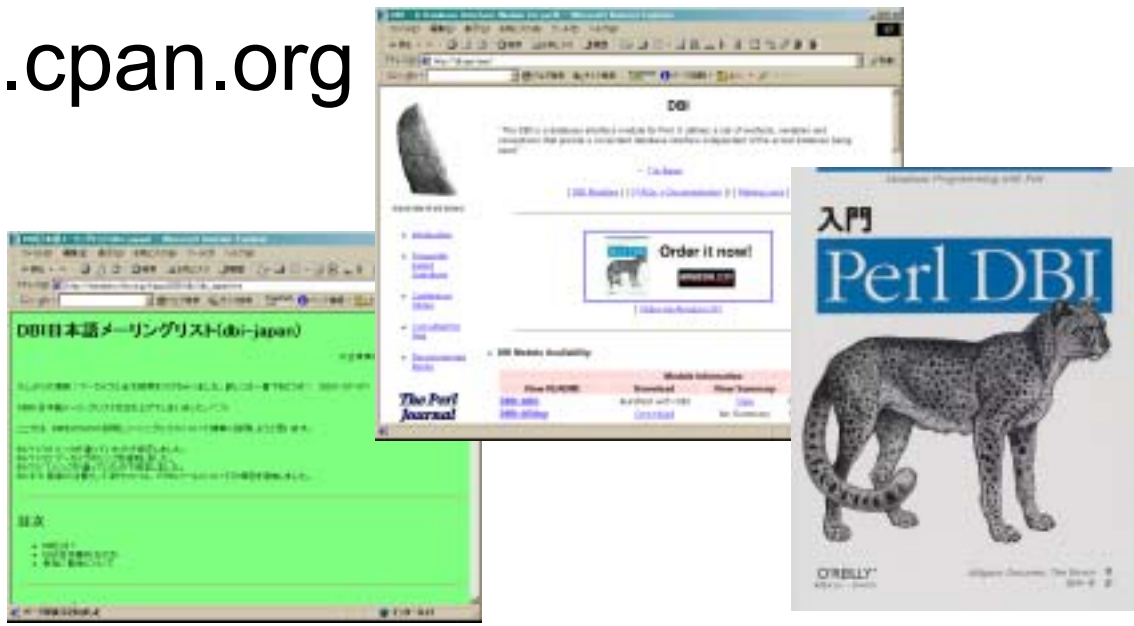
先頭項目(name)は10バイトの固定長
その後ろに現在の住所(addr)と出身地(birth)
がカンマ区切られて入っている。
そしてタブの後に年齢(age)が入っている

本日の流れ

- Perl/DBIとは？
  - DBIの構造、DBIの特徴
  - DBDの分類
- DBIのライバル？
  - DBIって速いの？
  - Perlって遅いの？
- Perlで簡単データベースを作る方法

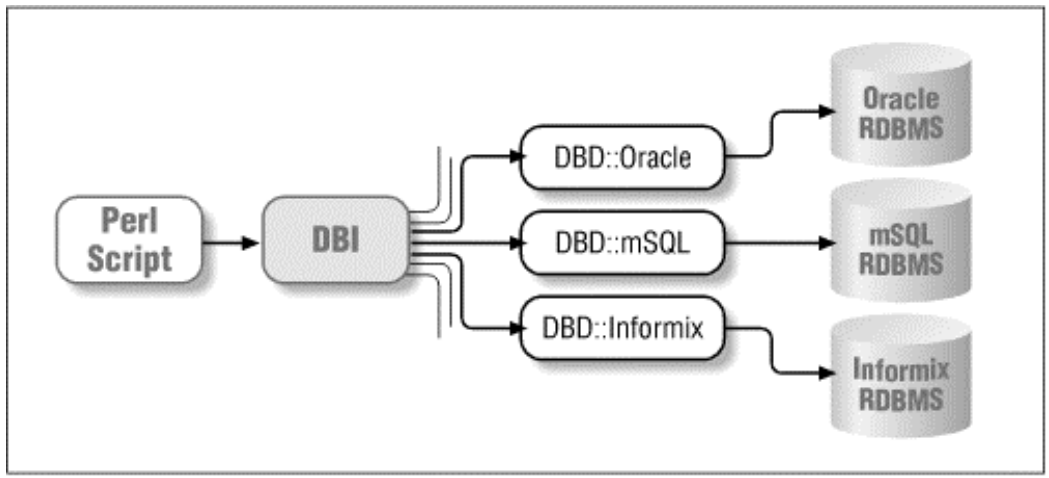
DBIはDB I/F

- Perlにおける代表的なデータベースI/F
- Tim Bunceが中心となり、作成
- http://dbi.cpan.org



DBI+DBD という構造

- DBI：DataBase Independent
- DBD:DataBase Depend



Programming Perl DBIからの引用
http://www.oreilly.com/catalog/perldbi/chapter/ch04.html

利用の流れ

- connect、prepare、execute、fetch

```
use strict;
use DBI;
my $hDb = DBI->connect('dbi:Pg:host=lins;dbname=test',
        'scott', 'tiger', {RaiseError => 1, AutoCommit=>0});
my $hSt = $hDb->prepare(q/SELECT * FROM TEST/);
$hSt->execute();
while(my $raD = $hSt->fetchrow_arrayref()) {
    print join(':',  @$raD), "¥n";
}
$hSt->finish;
$hDb->disconnect;
```


プレースホルダ、バインド値

- prepareしたSQLにexecuteで値を与える
  - 値だけが違うものを何回も実行するときなどに便利→パフォーマンスが上がる
  - クォート文字などのエスケープ自分でやる必要がない(特にバイナリデータなど)

```
my $hSt = $hDb->prepare('INSERT INTO ALBUM VALUES(?, ?, ?)');
$hSt->bind_param(3, undef, DBI::SQL_BINARY);
$hI->execute('A', '2001-11-25', $sStr);
```


ちょっとしたセキュリティの向上

```
SELECT * FROM TEST WHERE NO = '$sData'
```

- –$sDataが"AB'C"だったら？

```
SELECT * FROM TEST WHERE NO='AB'C'
```

- –$sData = "1' OR NO != '1"

```
SELECT * FROM TEST WHERE NO='1' OR NO != '1'
```

プレースホルダを使って入れば単なる値に

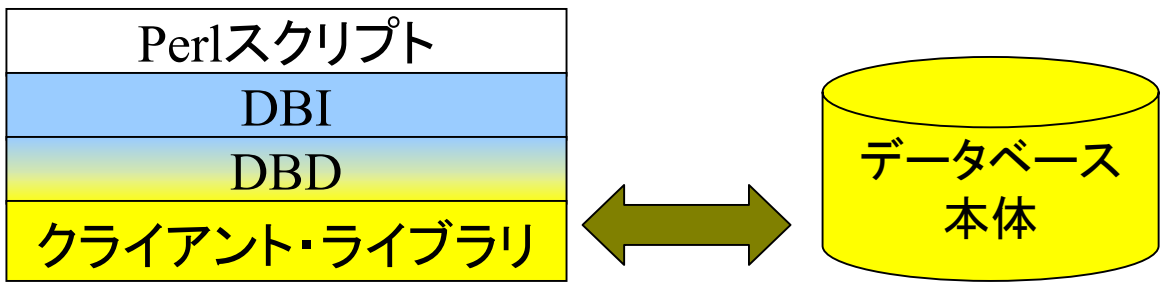
```
SELECT * FROM TEST WHERE NO='AB¥'C'
SELECT * FROM TEST WHERE NO='1¥' OR NO != ¥'1'
```


DBIの特徴

- 数多くのデータベースに対応
- 特定のデータベースに依存しない
- データベース特有の機能にも対応
- 多彩な取り出し系メソッド

DBDの分類

- 製品系：クライアントライブラリを利用する
  - DBD::Pg、DBD::mysqlなど主要なDBD



- 自作系：独自のデータベースを持っている
  - DBD::Sprite、DBD::CSVなど

DBDは40以上(1)

1．製品系

(1)データベース･システム

DBD-Adabas,DBD-ASAny,
DBD-DB2, DBD-DBMaker,
DBD-DtfSQLmac,
DBD-Empress, DBD-Illustra, DBD-Informix, DBD-Informix4,
DBD-Ingres, DBD-InterBase, DBD-Oracle, DBD-Ovrimos,
DBD-Pg, DBD-PgSPI, DBD-QBase, DBD-SearchServer,
DBD-Solid, DBD-Sybase, DBD-Teradata, DBD-Unify,
DBD-XBase, Msql-Mysql-modules

DBDは40以上(2)

(2)接続I/F

DBD-SQLRelay, DBD_SQLFLEX, DBD-FreeTDS, DBD-ADO, DBD-JDBC, DBD-ODBC

2. 自作系

DBD-AnyData, DBD-File, DBD-CSV, DBD-Excel, DBD-RAM, DBD-Sprite,

DBD-Proxy, DBD-LDAP, DBD-Chart, DBD-ExampleP, DBD-Sponge

データベースに依存しない

- データベースに依存しない構造、メソッド名

→移植が容易

  - 既存のシステムの移行など
    開発中は他のデータベースで作成し、本番環境は別のデータベースを利用する
    ex. DBD::Sprite → DBD::Oracle
  - 開発者の効率

特有の機能への対応

- SQL文は各データベース用のものを！
- DBD独自関数（funcメソッド）
  - DBD::Pgによるラージオブジェクトの対応など

```
my $oId = $hDb->func($hDb->{pg_INV_WRITE}, 'lo_creat');
my $oFd = $hDb->func($oId, $hDb->{pg_INV_WRITE}, 'lo_open')
```

- DBD独自プロパティ
  - DBD::mysqlのmysql_insertidプロパティなど

```
$insertId = $dbh->{‘mysql_insertid’}
#最後にオートインクリメントで挿入したIDの値
```


多彩な取り出し系

- 行単位に順次取り出し
  - fetchrow_arrayref、fetchrow_array、fetchrow_hashref
- まとめて
  - fetchall_arrayref : まとめて取り出し
  - selectrow_array：最初の行を配列で
  - selectall_arrayref、selectall_hashref
    全ての行を配列、ハッシュ・リファレンスで
  - selectcol_arrayref：全ての行の先頭カラム

fetch系メソッドで速いのは？

MS-Accessで5万件を取り出した時間

```
fetchrow_arrayref ： 18 秒
fetchrow_array    ： 23 秒
fetchrow_hashref  ： 36 秒
fetchall_arrayref ： 21 秒
```

- fetchrow_arrayrefが最速
- fetchrow_arrayは若干落ちる
- fetchrow_hashrefは1.5～2倍遅い

DBIは速いのか？

- プレースホルダが利用できる
  - 値をだけが違うSQLで威力を発揮
- 天下無敵のfetchrow_arrayref
  - Win32::ODBCには圧勝
  - Pgと比較してもDBI+DBD::Pgのほうが取り出しがほぼ互角か若干速いくらい。

Perlは本当に遅いのか？

- データベース関連の処理はPerlじゃない
  - XSを通じてクライアントライブラリを利用
  - データベースアクセスでのPerlのスクリプトの影響は相対的に小さい
- CGI=Perlではない
  - mod_phpと比較するのであればmod_perlのほうが正しい比較でしょ？
  - Apache::DBIという持続的コネクトへの工夫

ベンチマークをしよう

テスト方法
　Apacheのベンチマーク abを利用し、以下のコマンドで
　　./ab –c5 -n100 http://lins/perl/tpg.pl
　それぞれ5回実行、Requests per secondを記録。

CGI(1) 通常接続：Perl(2)、PHP(3) 持続接続:Perl(4)、PHP(5)

limit=10	CGI (1)	Perl (2)	PHP (3)	Perl (4)	PHP (5)
総平均	1.77	13.20	14.08	37.55	46.94
一回の時間(ms)	566.25	75.78	71.00	26.63	21.31

limit=50	(1)	(2)	(3)	(4)	(5)
総平均	1.73	11.47	11.23	26.52	25.02
一回の時間(ms)	578.03	87.18	89.05	37.70	39.97

KbWiki：PHPの方が軽くて速いは本当か？
http://www.hippo2000.net/cgi-bin/KbWiki/KbWiki.pl

DBDは作成簡単

- DBIがかなり肩代わりしてくれる
- 作成する必要があるもの
  - 3つのクラス(ドライバ、データベース、ステートメントの各ハンドルに対応)
  - 合計30個程度のメソッド／プロパティ
    - 必須のメソッドはもっと少ない
      prepare、executeなどの主要なメソッド、テーブル名、列名など
    - fetch系メソッドはfetchrow_arrayrefのみ

自作派にはSQL::Statement

- DBD::CSV、DBD::AnyDataなどのベースとなるSQLパーザー／エンジン
- 2002年1月に機能が大幅にアップしたPurePerl版がリリース(1.001)
  - PurePerlだから実行環境が幅広い
  - ジョインや集計関数もサポート

これを使えばPerlでデータベースが作れる！

例：DBD::Excelの場合

- 特徴：
  - プラットホームに依存しないPurePerlなDBDモジュール（Spreadsheet::WriteExcel、ParseExcelを利用）
  - SQL::Statementを利用し、データはメモリに展開
- ステップ数：現在 900行弱
- 開発日数　：５日（調査、作成、テスト）

AnyDataという革命

- SQL::Statementを利用したDBDを簡単に作ることが出来る
- 独自のクラスを追加
  - テキストファイル：1行の文字列を列ごとに分解する、列のデータを行の出力文字列にする
  - バイナリ：データを行・列単位に分割する、与えられたデータを出力文字列に変換する

→ほんの数行、数十行で出来上がり

あんなデータもSQLで操作する

AnyData::Format::SomeFmtの定義

```
package AnyData::Format::SomeFmt;
use strict;
use AnyData::Format::Base;
use vars qw( @ISA $VERSION);
@ISA = qw( AnyData::Format::Base );
$VERSION = '0.01';
my $RecSep = ($^O eq 'MSWin32')?
   pack('c2', 0x0D, 0x0A): "¥n";
sub new {
   my ($class, $flg) = @_;
   $flg ||= {};
   $flg->{col_names} ||= 'name,addr,birth,age';
   my $self = AnyData::Format::Base->new($flg);
   return bless $self, $class;
}
```

```
sub read_fields($$) {
   my ($self, $sRec) = @_;
   my ($sCol1, $sCol2, $sCol3, $sCol4, $sCol2_3);
   if($sRec =~ /^(.{10})(.*)¥t(.*)$/) {
      ($sCol1, $sCol2_3, $sCol4) = ($1, $2, $3);
      $sCol1 =~ s/¥s+$//;
     ($sCol2, $sCol3) = split /,/, $sCol2_3;
   }
   return ($sCol1, $sCol2, $sCol3, $sCol4);
}
sub write_fields {
   my ($self, @aFlds) =@_;
   return (sprintf("%-10s%s,%s¥t%s", @aFlds) . $RecSep);
}
1;
```


実際の操作

```
use strict;
use DBI;
my $hDb = DBI->connect('dbi:AnyData:', undef, undef,
        {RaiseError=>1, AutoCommit=>1,});
$hDb->func( 'testfmt', 'SomeFmt', 'anyfmt.txt', 'ad_catalog' );
$hDb->do('DELETE FROM testfmt WHERE age = 32');
$hDb->do(q/UPDATE testfmt SET name = '山吹花子'
                    WHERE name ='佐藤花子' /);
my $hSt = $hDb->prepare(q/INSERT INTO testfmt VALUES(?, ?, ?, ?)/);
$hSt->execute('川合孝典',  '京都','静岡', 37);
$hSt->execute('川合美加子', '京都','三重', '');

my $hSt = $hDb->prepare('SELECT * FROM testfmt ORDER BY age');
$hSt->execute;
while(my $raD = $hSt->fetchrow_arrayref()) {
   print join(':', @$raD), "¥n";
}
$hSt->finish;
$hDb->disconnect;
```

＝実行前のanyfmt.txt＝
山田一郎　埼玉,東京　28
鈴木次郎　静岡,静岡　32
佐藤花子　神奈川,東京　26

＝実行時の表示＝
川合美加子:京都:三重:
山吹花子:神奈川:東京:26
山田一郎　:埼玉:東京:28
川合孝典:京都:静岡:37

＝実行後のanyfmt.txt＝
山田一郎　埼玉,東京　28
山吹花子　神奈川,東京　26
川合孝典　京都,静岡　37
川合美加子京都,三重

簡単DBDの展開

次期AnyDataでは関数定義のみでも可能

- 横への展開
  - 関数定義→RDBMSへ
- 縦への展開
  - 関数定義→AnyDataのモジュール
    →DBDへの昇格

今後の展開

- 新しいDBDへの拡張
  - 新しいデータベースなど
- DBIxなどの拡充
- SQL::Statementのパワーアップ
  - バグの改修
  - トランザクションなどの機能面の向上

次はあなたのモジュールかも？