Databases and PERL

Erick Antezana
<erant@psb.ugent.be>
Motivation

- Data to be stored: text, strings,…
- Flat files
  - Don’t scale very well
  - Performance
  - Concurrency
  - …
- Solution: database
Database

• An organized collection of data
  – Quick and easy access to the data

• Relational database
  – Tables = relations
  – Normalization
    • Attributes are single valued and non-repeating
    • Functional dependency of non-PK attributes on PK
    • Non-PK attributes only dependant on the PK
    • http://en.wikipedia.org/wiki/Database_normalization
## Table

<table>
<thead>
<tr>
<th>Record 1 (tuple 1)</th>
<th>Attribute 1 (Column 1)</th>
<th>Attribute 2 (Column 2)</th>
<th>Attribute 3 (Column 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record 2 (tuple 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Keys

• Primary (PK)
  – Attribute(s) that uniquely identifies a record

• Foreign (FK)
  – Referential integrity, relations between two tables
  – PK in another table
SQL

- Structured Query Language
- Data manipulation
  - Table creation
  - Update
  - Deletion
- Queries
- DB administration
- RDBMS: MySQL, Oracle, …
PDFs

- http://backpan.cpan.org/authors/id/T/TI/TIMB/DBI_AdvancedTalk_200608.pdf
Why PERL?

- Rapid prototyping
- Fewer lines of code
- Text processing, handling processes
- Easy maintenance
- Swiss knife of programming languages
- “Must” for Bioinformaticians
- Lot of resources: books, websites, …
Perl and DBs

• Data processing
  – Parsing
  – Formatting
  – Filtering
  – ...

• Applications
  – Integration
  – Web
Simplified Typical Case

• I have a flat file with some data obtained in an experiment, data collected from some other DBs
• I need to parse/filter the relevant data
• OPT: Analysis, new data, …
• Make it publicly available thru the WWW
• OPT: Analysis, …
• Updates: new experiments, data from DBs
Perl modules

• What is a module?
  – self-contained set of reusable code
  – defined programming interface
  – use MyModule.pm;

• Is there a public library of modules?
  – Comprehensive Perl Archive Network
  – http://www.cpan.org
  – Go there first before implementing myself…
Interesting modules

- DBI.pm
- DBD...
- XML...
- CGI.pm
- HTML...
- ...
- ...
CGI

- Common Gateway Interface
- Method by which a web server can get/send data from/to db’s, documents, and other programs, and present that data to viewers via the web
- Widely used
- Simple interface
- [http://www.w3.org/CGI/](http://www.w3.org/CGI/)
- [http://search.cpan.org/dist/CGI.pm/](http://search.cpan.org/dist/CGI.pm/)
#!/usr/local/bin/perl -w

print "Content-type: text/html\n\n";
print "Hello, world";
CGI: ex2.pl

#!/usr/local/bin/perl

print "Content-type: text/html\n\n";

print "<html><head><title>CGI ex2.pl";
print "</title></head><body><h1>";
print "CGI Example # 2";
print "<table><tr><td>";
print "The date is: ", `date`
print "</table></body></html>\n";
<html>
<body>
<p>
<h1>This example shows how to interact with PERL from an HTML form</h1>
<form action="http://psbdev01/cgi-bin/ex3.pl" method="get">
Enter first number:
<input type=text name=number1>
<br>
Enter second number:
<input type=text name=number2>
Enter 3<sup>rd</sup> number:
<input type=text name=number3>
<br>
<input type=submit value="Submit">
</p>
</body>
</html>
This example shows how to interact with PERL from an HTML form

Enter first number: 
Enter second number: 
Submit
#!/usr/local/bin/perl

use CGI;

$query = new CGI;

print "Content-type: text/html\n\n";

print "<html><head><title>Perl CGI Example # 3</title></head><body><h1>Perl CGI Example # 2</h1><p>
$query->import_names("N");

$sum = $N::number1 - $N::number2;
$times = $N::number1 * $N::number2;

print "The sum of the two numbers is $sum";
print "<br>";
print "$N::number1 times $N::number2 is $times";
print "</p>";

print "</body></html>";
CGI: ex3 output

Perl CGI Example # 2

The sum of the two numbers is 12350
12345 times 5 is 61725
#!/usr/local/bin/perl -w

use strict;
use CGI;

my $q = new CGI;

print $q -> header( "text/html" ),
$q->start_html ( -title => "The TRANSISTOR project", -bgcolor => "#cceeaa" ),
$q->center(
$q->h1( "TRANSISTOR partners" ),
$q->p( "The table below shows some fake partners and the number of group members" ),
$q->table( {border => 1},
$q->Tr( {align => "center"},
[ 
   $q->th( [ 'Isaac Newton', 'Max Planck', 'Albert Einstein' ] ),
   $q->td( ['6','8','9'])
]
)
),
$q->end_html;
CGI output

TRANSISTOR partners

The table below shows some fake partners and the number of group members:

<table>
<thead>
<tr>
<th>Isaac Newton</th>
<th>Max Planck</th>
<th>Albert Einstein</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
PERL DBI

• Standard interface
• A PERL module
• Specification: set of constructs
• Independent of the actual DB
• See: http://dbi.perl.com
• http://search.cpan.org/~timb/DBI/DBI.pm
Example

use DBI; # Database Interface

$db = DBI->connect('dbi:ODBC:ONTO', 'user', 'password',
       { RaiseError => 1 });
$update = $db->prepare('UPDATE genes SET synonym=?
                     WHERE gene_ac=?');
$insert = $db->prepare('INSERT INTO genes(gene_ac,synonym)
                     VALUES (?,?)');

while ($line = <SYNONYMS>) {
    . . .
    $rows = $update->execute($synonym, $gene_ac);
    $insert->execute($gene_ac, $synonym) if ($rows == 0);
    . . .
}
$db->commit;
Example (Cont…)

```php
$synonyms = $db->prepare('SELECT gene_ac,synonym FROM synonyms');
$synonyms->execute;
while ( ($gene_ac, $cost) = $synonyms->fetchrow_array ) {
    . . .
}
```
DBI

- Simple architecture
- Flexible
- Free
- Interface for several DBs
  - Oracle,
  - MySQL,
  - Postgres,
  - Informix, and so forth…
DBI features

- Defines and implements an interface
- Provides methods, functions and tools for drivers
- Tracing, debugging
- Efficient implementations
- http://dbi.perl.org
DBI

Perl script

DBI

DBD::Oracle
DBD::MySQL
DBD::Postgresql

Oracle
MySQL
Postgresql
#!/usr/bin/perl -w

use strict;
use DBI;

my %attr = (  
    PrintError => 0,  
    RaiseError => 1,  
    AutoCommit => 1,  
    AutoCommit => 0  
);

my $dbh = DBI->connect("dbi:mysql:db_transistor:host=psbora01", "transistor", "trans", \%attr) or die ($DBI::errstr, "\n");

my $sth = $dbh->prepare("SELECT name, family_name FROM Person" );

$sth->execute;
$sth->dump_results;
$sth->disconnect;
$dbh->disconnect;
#!/usr/bin/perl -w

use strict;
use DBI;

my %attr = (
    PrintError => 0,
    RaiseError => 1,
    AutoCommit => 0
);

my $dbh = DBI->connect("dbi:mysql:db_transistor:host=psbora01", "transistor", "trans", \%attr) or die ( $DBI::errstr, "\n");

my $sth = $dbh->prepare("SELECT name, family_name FROM Person");

my ( $name, $family_name);

$sth->execute;

while ( ( $name, $family_name ) = $sth->fetchrow_array ) {
    print "$name $family_name\n";
}

$dbh->disconnect;
#!/usr/bin/perl -w

use strict;
use DBI;

my %attr = (  
    PrintError => 0,  
    RaiseError => 1,  
    AutoCommit => 0  
);

my $dbh = DBI->connect("dbi:mysql:db_transistor:host=psbora01",  
    "transistor", "trans", %attr) or die ($DBI::errstr, "\n");

my @results = $dbh->selectrow_array("SELECT * FROM Person WHERE  
    nationality='Spain'  ");

foreach my $field (@results) {
    $field = "" unless $field;
    print "$field\n";
}

$dbh->disconnect;
Database handles

- Create database handles
  - `$dbh=DBI->connect('DBI:mysql:db_transistor');`

- Database methods
  - `$dbh->do('DELETE FROM Team');`

- Transaction control
  - `$dbh->rollback;`
  - `$dbh->commit;`

- Disconnect
  - `$dbh->disconnect;`
Statement handles

• Create (prepare) an statement handle
  - $sth = $dbh->prepare("DELETE FROM Team
    WHERE name LIKE '%$name%'");

• Statement handles let us execute the prepared SQL code
  - $sth->execute();

• Statement handles may be released
  – Implicitly
    • Once it is out of scope
  – Explicitly
    • undef $sth;
Commit

• On by default
• Off:
  – $dbh->{AutoCommit} = 0;
• Off:
  – $dbh = DBI->
    connect("DBI:mysql:db_transistor",
    "transistor", "trans", { AutoCommit => 0 });
<html>
  <head>
    <title> Getting connected to a mysql database</title>
  </head>
  <body bgcolor="#afeeee">
    <center>
      <form action="ex5.pl" method="post">
        <h1> Getting connected to a mysql database</h1>
        <b>Database name:</b> <input type="text" name="nom" size="24"> <BR>
        <b>Password:</b> <input type="password" name="motdepasse" size="24"> <BR>
        <input type="submit" name="submitButtonName" value="Connect to database">
        <input type="reset">
      </form>
    </center>
  </body>
</html>
#!/usr/local/bin/perl

use CGI;
use DBI;

# get the params from the form:
$cgi = CGI -> new();
my $nom = $cgi -> param('nom');
my $user = $cgi -> param('user');
my $pass = $cgi -> param('motdepasse');

## Connect to the database.
my $dbh = DBI->connect("DBI:mysql:$nom:host=psbora01", "$user", "$pass");

## extract a list of the database's tables
$n = @tables = $dbh->tables();

## Disconnect from the database.
$dbh -> disconnect();

## start HTML
print "Content-type: text/html \n\n";
print "<HTML><HEAD></HEAD><BODY bgcolor='#afeeee'>\n";
print "<h1>PERL CGI Example # 5</h1>
print "<table border=2>
print "<tr>
print "<th>There are $n tables in this database</th>
print "</tr>

## print data
foreach $table (@tables) {
    print "<tr><td>$table</td></tr>
}
print "</table>
print "<hr>
print "<h1>Simple query using the table Person:</h1>

## form for query
print "<form action="ex6.pl" method="post">
print "<input type="hidden" name="nom" value="$nom">
print "<input type="hidden" name="user" value="$user">
print "<input type="hidden" name="motdepasse" value="$pass">
print "<input type="text" name="query" size="24">
print "<input type="submit" name="search"
print "<input type="reset">
print "</form>
print "</body>
print "</html>";
PERL CGI Example # 5

There are 6 tables in this database:
- 'Benz/
- 'Presse/
- 'Tom/
- 'Tit/
- 'Tips_Benz/
- 'student/

Simple query using the table Person:

Country name:  Query the table  Reset
#!/usr/local/bin/perl

use CGI;
use DBI;

$sgi = CGI -> new();
my $nom = $cgi -> param('nom');
my $user = $cgi -> param('user');
my $pass = $cgi -> param('motdepasse');
my $query = $cgi -> param('query');

my $dbh = DBI->connect("DBI:mysql:$nom:host=psbora01", "$user", "$pass") or die ($DBI::errstr, "\n");

print "Content-type: text/html \n\n";
print "<HTML><HEAD></HEAD><BODY bgcolor='#afeeee'>\n";

$requete="SELECT name, family_name, country from Person where country like \"\%$query\%\";";
print "<h1>$requete</h1>\n";
my $sth = $dbh->prepare($requete);
$sth->execute();
$numRows = $sth->rows;
print "<h1>There are $numRows entries:</h1>\n";
$numFields = $sth->{'NUM_OF_FIELDS'};
$colNames = $sth->{'NAME'};

print "$numRows rows;\nprint "<h1>There are $numRows entries:</h1>\n";
print "<table border=2>\n";
print "<tr>";
for ($i = 0; $i < $numFields; $i++) {
    print "<th>$$colNames[$i]</th>"
}
print "</tr>\n";

while (my $ref = $sth->fetchrow_arrayref) {
    print "<tr>";
    for ($i = 0; $i < $numFields; $i++) {
        print "<td>$$ref[$i]</td>"
    }
    print "</tr>\n";
}
print "</table>\n";

$sth->finish();
$dbh->disconnect();

print "</body>";
print "</html>";
SELECT name, family_name, country from Person where country like 'Belgium';

There are 2 entries:

<table>
<thead>
<tr>
<th>name</th>
<th>family_name</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ward</td>
<td>Richard</td>
<td>Belgium</td>
</tr>
<tr>
<td>Dave</td>
<td>Dr. White</td>
<td>Belgium</td>
</tr>
</tbody>
</table>
More doc

- Resources:
  - Books:
    - Learning Perl, Schwartz/Christiansen
    - Programming Perl, Wall/Christiansen/Schwartz
  - http://www.perl.com
  - http://learn.perl.org
  - “Programming the Perl DBI”, ISBN 1-56592-699-4