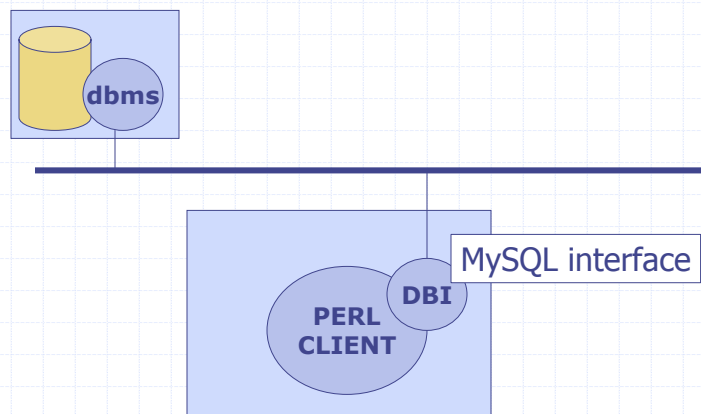


# Database Programming MySQL and Perl DBI

*Randy Julian*  
*Lilly Research Laboratories*

## Program Clients



## Perl DBI/DBD Interfaces

- ◆ DBI is the generic interface which relies on a database specific driver: DBD
- ◆ Must install the mysql DBD driver to use the DBI interface with MySQL

```
PPM>install DBD::mysql
```

## DBI: An Object Oriented Module...

- ◆ Uses the Perl Object-Oriented syntax for calls and access to return values:

```
my $dbh = DBI->connect( $dsn, $user_name, $password,  
                        { RaiseError => 1, PrintError => 0 } );
```

## Naming Conventions

### DBI Handle Variable Names

<b>Name</b>	<b>Meaning</b>
<code>\$dbh</code>	A handle to a database object
<code>\$sth</code>	A handle to a statement (query) object
<code>\$fh</code>	A handle to an open file
<code>\$h</code>	A generic handle - depends on context

### DBI Non-Handle Variable Names

<b>Name</b>	<b>Meaning</b>
<code>\$rc</code>	Return code from true/false operations
<code>\$rv</code>	Return code from "int" operations
<code>\$rows</code>	Return code from ops than return row count
<code>@ary</code>	Array (list) returned from a query

## Some DBI member functions

<code>-&gt;connect()</code>	connect to a database
<code>-&gt;prepare()</code>	setup a query
<code>-&gt;execute()</code>	perform a query that returns a result set
<code>-&gt;do()</code>	perform a query that returns row count
<code>-&gt;finish()</code>	complete a partial query
<code>-&gt;disconnect()</code>	disconnect from the database

## Simple Example: dump\_atom.pl

```
use strict;
use DBI;

my $dsn = "DBI:mysql:compound:localhost"; # data source
my $user_name = "chem";                 # user name
my $password = "chem";                  # password

# connect to database
my $dbh = DBI->connect( $dsn, $user_name, $password,
                       { RaiseError => 1, PrintError => 0 } );
```

here we have a handle to a database object: `$dbh`

## Perform a query on the database

```
# issue query
my $sth = $dbh->prepare(
    "SELECT * FROM atom ORDER BY atom_id" );
$sth->execute();
```

here we have a handle to a database query object: `$sth`  
and... we have the result set stored in the `$sth` object

## Get the results from the query

```
# read results of query, then clean up
while( my @ary = $sth->fetchrow_array() )
{
    print join("\t", @ary), "\n";
}
$sth->finish();
```

here we have an array holding the result set: @ary  
and... we have cleaned up the query object with finish()

## Disconnect and exit

```
$dbh->disconnect();
exit(0);
```

## Output

1	1	1	-0.3458	-2.9667	0	C
2	1	2	0.3667	-2.55	0	C
3	1	3	0.3621	-1.725	0	O
4	1	4	1.0834	-2.9585	0	C
5	2	1	-20	-15	0	C
6	2	2	-1	-15	0	C
7	2	3	20	-15	0	N
8	8	1	0.9754	-1.6212	0	C
9	8	2	0.9629	-4.0087	0	C
10	8	3	3.3545	-4.0212	0	C
11	8	4	3.367	-1.6337	0	C
12	8	5	1.8	-2.4458	0	C
13	8	6	1.8	-3.2708	0	C
14	8	7	2.625	-3.2708	0	C
15	8	8	2.625	-2.4458	0	C

## DBI Fetching Methods

### Method Name

fetchrow\_array()  
fetchrow\_arrayref()  
fetch()  
fetchrow\_hashref()

### Return Value

Array of row values  
Reference to array of row values  
same as fetchrow\_arrayref()  
Reference to a hash of row values  
- keyed by column name

### Method Name

fetchall\_arrayref()

### Return Value

Reference to array of row values  
- all the rows

## Quoting Issues

- ◆ SQL statements use quotes, so does Perl...
- ◆ Both Perl and SQL allow you to use *either* single *or* double quotes

```
$id = 7;  
$query = "INSERT INTO name VALUES(NULL, $id, \'acetone\');
```

```
INSERT INTO name VALUES(NULL, 7, \'acetone\')
```

```
$query = `INSERT INTO name VALUES(NULL, $id, \"acetone\`);
```

```
INSERT INTO name VALUES(NULL, $id, \'acetone\')
```

## Using qq{ }

```
$id=14;  
$name="acetonitrile";  
  
$query = qq{  
    INSERT INTO name VALUES(NULL, $id, \'$name\')  
};
```

## Using ->quote()

```
$name = "Triethylamine, 2,2',2"-trichloro-";  
$rows = $dbh->do(qq{ INSERT INTO name  
                      VALUES(NULL, 7, '$name') });
```

```
INSERT INTO name  
VALUES(NULL, 7, 'Triethylamine, 2,2',2"-trichloro-')
```

---

```
$name=$dbh->quote("Triethylamine, 2,2',2"-trichloro-");  
$rows = $dbh->do(qq{ INSERT INTO name  
                      VALUES(NULL, $id, '$name') });
```

```
INSERT INTO name  
VALUES(NULL, 7, 'Triethylamine, 2,2\,2\ \'-trichloro-')
```

## Example: Loading molfiles

- ◆ Objectives:
  - Read a .mol file
  - Parse out information
  - Store in database
  - Add compound name from filename
- ◆ Starting point:
  - mol2cml.pl - has mol parsing (almost)
- ◆ Result:
  - compound.sql
  - mol2sql.pl

## 1. Parse the command line

```
if( not defined($ARGV[0]) )
{
    die( "No file name supplied\n");
}
else
{
    $file = $ARGV[0];
}

if( not defined($ARGV[1]) )
{
    $mol_name = $file;
    print "molecule name set to filename ($file)\n";
}
else
{
    $mol_name = $ARGV[1];
}
```

## 2. Connect to the database

```
my $dsn = "DBI:mysql:compound:localhost"; # data source
my $user_name = "chem"; # user name
my $password = "chem"; # password

# connect to database
my $dbh = DBI->connect( $dsn, $user_name, $password,
    { RaiseError => 1, PrintError => 0 } );
```

### 3. Read entire file into an array

```
$INPUT_FILE = "@ARGV[0]" . ".mol";  
open(INPUT_FILE);  
@array = <INPUT_FILE>;  
close(INPUT_FILE);  
  
foreach (@array) {  
    $wholefile = $wholefile.$_;  
}
```

### 4. Parse atoms & bonds tables

acetone.mol

```
-ISIS- 04060323172D  
  
4 3 0 0 0 0 0 0 0 0999 V2000  
-0.3458 -2.9667 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0  
0.3667 -2.5500 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0  
0.3621 -1.7250 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0  
1.0834 -2.9585 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0  
2 3 2 0 0 0 0  
1 2 1 0 0 0 0  
2 4 1 0 0 0 0  
M END
```

## 4a. RegEx matching: atoms

Find and load into an array, lines containing atoms

```
@no_of_atoms=$wholefile =~
```

```
m!(?=\n)\s+[-\d\.]+\s+[-\d\.]+\s+[-\d\.]+\s+[A-Za-z]+[\n]*(?=\n)!gmx;
```

```
-0.3458 -2.9667 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n
```

## 4b. RegEx matching: Bonds

```
4 3 0 0 0 0 0 0 0 0 0999 V2000
```

```
2 3 2 0 0 0 0 0
```

```
$wholefile =~ s/\s+([\d]+\s+)+V2000//;
```

this deletes the count line from the array

## 4b. RegEx matching: Bonds (cont)

```
@no_of_bonds=$wholefile =~  
m!(?=\n)\s+[\d]+\s+[\d]+\s+[\d]+[^\n]*(?=\n|$)!gmx;
```

2 3 2 0 0 0 0\n

## 5. Load molecule table

```
$no_of_atoms=@no_of_atoms;  
$no_of_bonds=@no_of_bonds;  
  
# issue query  
$rows = $dbh->do( qq{  
  INSERT INTO molecule (mol_id, num_atoms, num_bonds, CAS)  
  VALUES( NULL, $no_of_atoms, $no_of_bonds, 'new' ) } );
```

mol\_id is "auto increment"...

...but we don't know what it is, so mark it 'new'

## 5a. Get mol\_id

```
# issue query
my $sth = $dbh->prepare(
    "SELECT mol_id FROM molecule
    WHERE CAS='new'");
$sth->execute();

my @ary = $sth->fetchrow_array();
$mol_id=$ary[0];

$rows = $dbh->do( qq{
    UPDATE molecule SET CAS="
    WHERE cas='new'
});
```

find the record

get mol\_id

fix the record

## 6. Load atom Table

```
for ($i=1;$i<=$no_of_atoms;$i++) {
    if( $wholefile =~ m!(?=\n)\s+([-d\.]*)\s+([-d\.]*)\s+([-d\.]*)\s+
    ([A-Za-z]+)[^\n]*(?=\n)!gmx ) {
        my $x = $1;
        my $y = $2;
        my $z = $3;
        my $atom = $4;

        $sql= qq{
            INSERT INTO atom (atom_id, mol_id, sequence, x, y, z, atom_type)
            VALUES( NULL, $mol_id, $i, $x, $y, $z, '$atom' ) };
        $rows = $dbh->do( $sql );
    }
}
```

\$2

\$3

\$1

\$4

## 7. Load bond Table

```
for ($i=1;$i<=$no_of_bonds;$i++)
{
  if( $wholefile =~
m!(?=\n)\s+([\d]+)\s+([\d]+)\s+([\d]+)[^\n]*(?=(\n|$))!gmx )
  {
    my $atom_1 = $1;
    my $atom_2 = $2;
    my $type = $3;
    $rows = $dbh->do( qq{
      INSERT INTO bond (bond_id, mol_id, atom_1, atom_2, bond_type)
      VALUES( NULL, $mol_id, $atom_1, $atom_2, '$type' ) } );
  }
}
```

## 8. Load name Table & Exit

```
$rows = $dbh->do( qq{
  INSERT INTO name (name_id, mol_id, name )
  VALUES( NULL, $mol_id, '$mol_name' ) } );

$dbh->disconnect();
exit(0);
```

## CHM696D MySQL Server

- ◆ miner.chem.purdue.edu
- ◆ port 3306
- ◆ username: (first letter of first name)last name "rjulian"
- ◆ password: what you gave me
- ◆ database: username

## Mini-Lab:

- ◆ Go to computer lab
- ◆ Download SQLyog:  
<http://www.webyog.com/sqlyog/>
- ◆ Establish a connection to server
- ◆ Create tables for 'compound' in your database (you will need them for the final exam...)