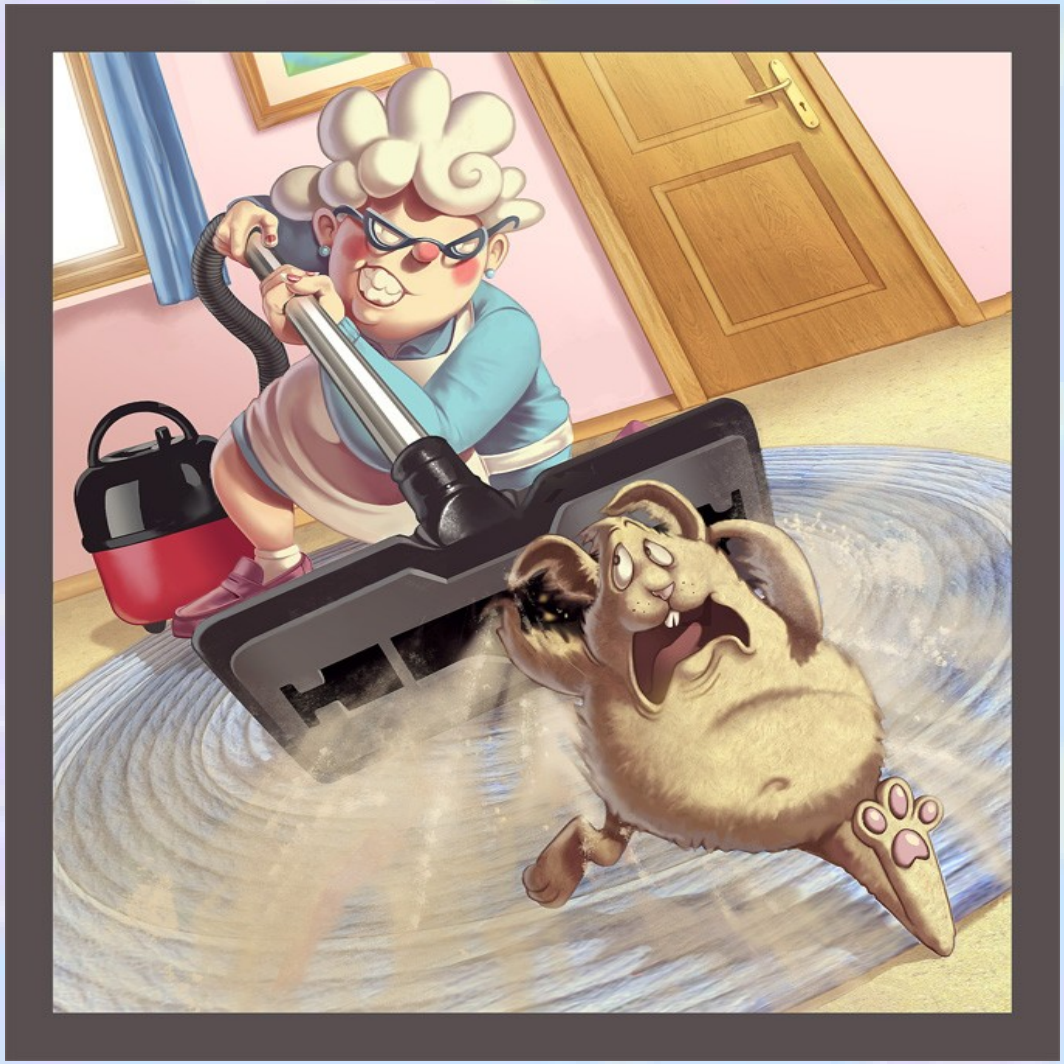
DBIx::QueryByName

or how to hide far and deep this ugly SQL code of yours



Mixing SQL and perl?

```
my $sth = $dbh->prepare('UPDATE departments
                         SET num_members = num_members + 1
                         WHERE id = ?')
    or die "Couldn't prepare statement: " . $dbh->errstr;
```

or?

```
my $sql = ''.
'UPDATE departments '.
'   SET num_members = num_members + 1'.
' WHERE id = ?';

my $sth = $dbh->prepare($sql)
    or die "Couldn't prepare statement: " . $dbh->errstr;
```

or?

```
my $sql = <<__SQL__;
UPDATE departments
   SET num_members = num_members + 1
 WHERE id = ?
__SQL__

my $sth = $dbh->prepare($sql)
    or die "Couldn't prepare statement: " . $dbh->errstr;
```

Building SQL in perl?

```
if (($bedrooms > 0) or ($price > 0)) {
    if ($in_clause != '') { $in_clause .= ','; }
    $in_clause .= '12';
}

# $where is '' at this point, no need for an and
if ($in_clause != '') {
    $where .= " fielddef_id in ($in_clause)";
}
if ($post != '') {
    if ($where != '') { $where .= " and"; }
    $where .= " value LIKE '%$post%'";
}
if ($bedrooms > 0) {
    if ($where != '') { $where .= " and"; }
    $where .= " value >= '$bedrooms'";
}
if ($price > 0) {
    if ($where != '') { $where .= " and"; }
    $where .= " value >= '$price'";
}
$query ="select distinct product_id from fieldvals";

if ($where != '') { $query .= " where $where"; }
```

Got enough?

SQL, be gone!

- SQL and perl are 2 different languages
- Mixing multiple languages in the same source is a bad idea:
  - Looks ugly
  - Harder to understand
  - Introduce bugs
- Separate perl and SQL code!!

How?

- Use an object relational mapper:
  - DBIx::Class
  - Rose
  - Jifty::DBI
  - Catalyst
  - etc.
- Perlishify SQL (DBIx::Perlish)

But...

- SQL still does a better job at querying database
- Some database specif c SQL tricks are only available via SQL
  - Ex: oracle hints
- So what to do when you need SQL but don't want it spoiling the source?

SQL in a file

```
<queries>

    <query name=”create_table” params=””>
CREATE TABLE jobs (
    INT id PRIMARY KEY,
    VARCHAR username,
    VARCHAR description,
    INT status,
)</query>

    <query name="add_job" params="id,username,description">INSERT
INTO jobs (id, username, description, status) VALUES
(?,?,?,0)</query>

    <query name="get_job_count" params="">SELECT COUNT(*) FROM
jobs</query>

</queries>
```

Still ugly, but at least it's not in your source...

Load and call as methods

```
use DBIx::QueryByName;

my $dbh = DBIx::QueryByName->new();

$dbh->connect("db1",
              "dbi:Pg:dbname=mydb;host=127.0.0.1;port=6666",
              $username, $password);

$dbh->load(session => 'db1', from_xml_file => $queries);

# insert a few rows in db1.jobs
$dbh->add_job( { id => 12,
                 username => "tom",
                 description => "catch mouse" } );

$dbh->add_job( { id => 13,
                 username => "jerry",
                 description => "run away from cat" } );
```

Bulk insert

```
# or insert 2 rows at once
$dbh->add_job( { id => 12,
                 username => "tom",
                 description => "catch mouse" },
               { id => 13,
                 username => "jerry",
                 description => "run away from cat" }
             );
```

Fetch results

```
my $sth = $dbh->get_job_count();

while (my @row = $sth->fetchrow_array()) {
    # do stuff
}


$sth = $dbh->get_user_jobs({ username => 'bob'});

while (my @row = $sth->fetchrow_array()) {
    # do stuff
}
```

That's it!!

More features...

- Automatic reconnect when session closes
- Re-execute queries when safe to do so
- Fork safe
- Log::Log4perl compatible

Questions? No? Thank you!