

# *AnyData – A Case Study*

---

Daniel J. Wright  
Lead Software Developer  
pairNIC.com

Download these slides with notes:

- <http://www.dwright.org>

The official AnyData project page:

- <http://www.vpservices.com/jeff/programs/AnyData>

# The Foundational Idea

- ◆ AnyData -
  - ◆ *“A suite of Perl modules dedicated to the proposition that all data is created equal”*
- ◆ One interface plus many data formats equal less work!



# *Supported Formats*

- ◆ Comma Separated Value
- ◆ Fix Width Tables
- ◆ HTML Tables
- ◆ INI Files
- ◆ MP3 Files
- ◆ Vertical Files
- ◆ XML Files
- ◆ Passwd Files
- ◆ Pipe Delimited Files
- ◆ Tab Delimited Files
- ◆ Web Common Log Format Files
- ◆ In-Memory Only Tables

# One Simple API!!!

- ◆ Find a user's home directory from a passwd file

```
use AnyData;
```

```
my $users = adTie( 'Passwd', '/etc/passwd' );
```

```
print $users->{jdoe}->{homedir};
```

- ◆ Retrieve a CSV file from an FTP server and print it as HTML

```
print adConvert( 'CSV',  
                'ftp://foo.edu/pub/bar.csv',  
                'HTMLtable' );
```

# *AnyData Can Be Extended*

- ◆ You can build your own format modules.
- ◆ Just make your own AnyData::Format::\_\_\_\_\_
- ◆ Inherit from AnyData::Format::Base and override what you need.

# Meet The Reaper

- ◆ “The Reaper” keeps logs that were only intended to be human parsable.
- ◆ Eventually, we needed to parse those logs automatically.



# *package AnyData::Format::ReaperLogs;*

```
use base qw[AnyData::Format::Base];

sub new {

    my $class = shift;

    my $self = { %{+shift},
        record_sep => "\n",
        key => 'datestamp',
        keep_first_line => 0
        colnames =>
        'string,date,reason,action,action_value,user,cmd' };

    return bless $self, $class;

}
```

# sub read\_fields

```
sub read_fields {
  my ($self, $str) = @_ ;
  return unless $str;
  my (@row) = $str =~ /
    # Column                               Delimiter
    ^(\w+\s\w+\s+\d+\s\d\d:\d\d:\d\d\s\w+\s\d{4}) \s::\s\(  

    ([^\])*) \)\s  

    (\w+) \s  

    ([^:]*?) : \s(?:PID\s\d+, \s)?USER\s  

    ([^,]+) , \sCMD\s'  

    (.*) '$/x;
  return @row ? ($str, @row) : $str;
}
```

# Accessing Log Data

- ◆ Now, we can use the AnyData API to access our logs:

```
my $reaperlogs = adTie( 'ReaperLogs', '/path/to/logs' );
```

- ◆ Print a list of all times Joe's processes were reaped:

```
my $reaps = $reaperlogs->{ {user => 'joe'} };  
print "$_->{date}\n" for @$reaps;
```

- ◆ Count per-user how many times we reaped PHP:

```
my %count;  
my $reaps = $reaperlogs->{ {cmd => qr/php/} };  
$count{$_->{user}}++ for @$reaps;
```

# ***Making Things Scale***

- ◆ You have a 300 row CSV file.
- ◆ It is going to grow to 300,000 rows.
- ◆ You want to move to SQL
- ◆ You don't have time to write all of the code right now.
- ◆ You want to write **some** SQL and have it work **before** you migrate your data to a DB server...

# ***DBD::AnyData***

- ◆ Gives you a DBI interface to text files without a daemon.
- ◆ Now you can use SQL on your CSV file.
- ◆ You can also access that CSV file the old way **at the same time!**
- ◆ Applications with many installations get a feature for free.

# *A DBD::AnyData Case Study*

- ◆ An example from pairNIC:
  - ◆ Single DB server, replication not available.
  - ◆ Query volume getting too high.
  - ◆ Data seldom updated.
  - ◆ Many client servers.

# *Our Solution*

- ◆ Convert the data in our tables to the in-memory format at compile time.
- ◆ Make the data available via `DBD::AnyData`
- ◆ Convert existing code to query new database handle.

# Importing Our Data

```
my $MEMORY_DBH;  
if ( CACHING_ENABLED ) {  
    $MEMORY_DBH =  
        DBI->connect( 'dbi:AnyData: (RaiseError=>1) ' );  
  
my $dbh_pg = get_pairnic_dbh();  
foreach my $table ( TABLE_A, TABLE_B ) {  
    my $query = sprintf( "select * from %s", $table );  
    $MEMORY_DBH->func( $table, 'DBI', $dbh_pg,  
        {sql=>$query}, 'ad_import' );  
}  
}
```

# Selecting The DB Handle

- ◆ `get_db_handle()` is the only thing that knows which handle we are using:

```
sub get_db_handle {  
    return $MEMORY_DBH if ( CACHING_ENABLED );  
    return get_pairnic_dbh()  
}
```

# *Analysis of DBD::AnyData*

## ♦ Advantages

- ♦ Fewer queries on DB servers.
- ♦ One code base.
- ♦ Improved query speed.
- ♦ Cheap.

## ♦ Disadvantages

- ♦ Required restarts on data change.
- ♦ Queries became slower.

# *How DBD::AnyData helped pairNIC grow*

- ◆ Initially, we had a single database server.
- ◆ We saw a future bottleneck.
- ◆ We wrote 17 lines of code.
- ◆ Then we didn't need to rush to find the long-term solution.



# Conclusions

- ◆ The ability to access multiple data stores with a single API means developers have freedom to change.
- ◆ The ability to intermix flat file access along with SQL access for the same data store means code can be released and tested rapidly during migration.
- ◆ We don't know always know which scripts are going to grow over time. But when we use tools like AnyData, we get a really convenient tool now, with large growth potential in the future.

# *For More Information*

- ◆ Presenter:

Daniel Wright <Dan@DWright.Org>

- ◆ AnyData author:

Jeff Zucker <jeff@vpservices.com>

- ◆ Download these slides:

<http://www.dwright.org>

- ◆ Or, just check out a copy of AnyData at your friendly local CPAN mirror...

