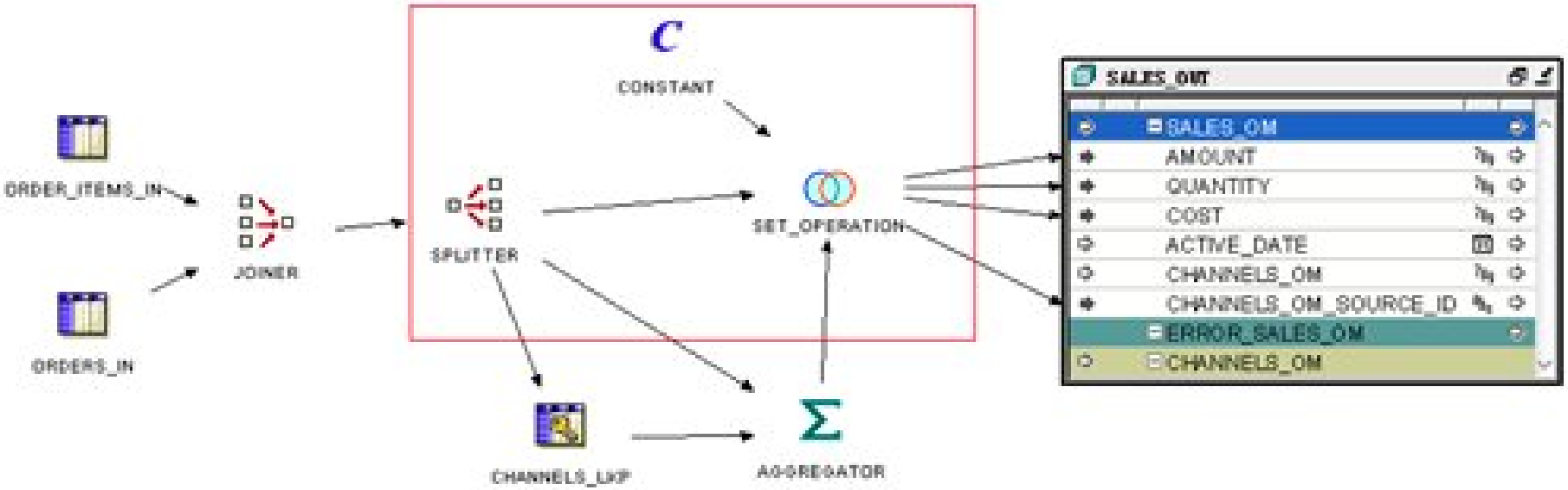
ETLp

A Simple ETL Framework

The Problem

- Customer Warehouse uses Oracle Warehouse Builder
- Hard to find **good** OWB resource at short notice
  - Without paying an arm and a leg
  - Same applies to any ETL tool
- Hampers the delivery of solutions to the business in a timely manner
- Hard to integrate with source control, release and build processes etc.
- Need to make a decision for the new warehouse

OWB provides graphical interface



Similar Tools

- Proprietary
  - Informatica
  - Datastage
- Open Source
  - Talend
  - Pentaho

The Alternative

- Hand-code the ETL
- Honourable history of Shell scripts, Perl and stored procedures used to provide ETL solutions
- Hand-coded solutions have problems:
  - Poor auditing
  - Poor metadata maintenance
  - Can lead to spaghetti code that is poor performing and hard to maintain
    - (As opposed to spaghetti ETL maps!)

Half-way house

- Add structure, error messaging and auditability to the hand-coded solutions (equivalent to OWB's audit browser)
- Where appropriate, use configuration and convention rather than hand-coding
- Supply routines to do the common day-to-day ETL processing (db-generic)
- Developers concentrate on business solutions (db-specific, problem domain-specific)

Implement a framework

- It should be light
  - We only have a short time to develop
- Once we're happy, we're done
  - Shouldn't require continual enhancements unless we introduce new technology (e.g. another type of DB to talk to)
- It should be unobtrusive
  - It shouldn't get in the way of the developers

Requirements

- Simple to use
- Provide the kind of auditing found in an ETL tool
- Can talk to any DB type
  - (once the DB-specific interface is written)
- Can validate and load data
  - Plan to call DB-specific loaders for large files
- Can link scheduled jobs to the processes they execute

Try to use repeatable ETL Patterns

- FTP Files
- Decrypt them
- Gunzip them
- Validate them
- Load data into staging tables
- Gzip the file
- Process into atomic data stores (3NF)
- Process into BI data marts (dimensional)

Introducing ETLp

- Configuration driven processing
- All processing is audited and is viewable in the audit browser
- Functionality is implemented with Plugins
- Housekeeping tasks like emailing of alerts is automatically handled
- Supports a number of databases (Oracle, PostgeSQL, MySQL)
- Open Source rewrite of original code.
  - Open Source version not used in anger
- Written in Perl – MooseX::Declare

ETLp Audit Browser

Jobs

ETLp: Jobs

Jobs | Items | File | User Admin | Edit Password | Logout

Config | Section | Status All | Min Start Date >= ... | Max Start Date < ... | Go

CONFIG	SECTION	STATUS	# ITEMS	SESSION ID	OS PROCESS ID	DATE CREATED	DATE UPDATED
bids	bids	running	3	1	11213	12/07/2010 21:36:06	12/07/2010 21:37:46
offers	offers	succeeded	4	1	11210	12/07/2010 21:35:49	12/07/2010 21:36:01
offers	generatorreserves	succeeded	12	1	11194	12/07/2010 21:35:26	12/07/2010 21:35:47
demo	five_min_prices	succeeded	8	1	11191	12/07/2010 21:35:19	12/07/2010 21:35:21
bids	bids	succeeded	8	1	10996	12/07/2010 21:29:34	12/07/2010 21:35:16
demo	five_min_prices	succeeded	32	1	10993	12/07/2010 21:29:20	12/07/2010 21:29:29
bids	bids	succeeded	12	1	10723	12/07/2010 21:20:38	12/07/2010 21:29:16
offers	generatorreserves	succeeded	12	1	10707	12/07/2010 21:20:12	12/07/2010 21:20:34
bids	bids	succeeded	12	1	10274	12/07/2010 21:11:20	12/07/2010 21:20:09
offers	offers	succeeded	12	1	10227	12/07/2010 21:10:44	12/07/2010 21:11:17
offers	ilreserves	succeeded	8	1	10211	12/07/2010 21:10:31	12/07/2010 21:10:39
bids	bids	succeeded	8	1	9857	12/07/2010 21:04:45	12/07/2010 21:10:26
offers	offers	succeeded	4	1	9827	12/07/2010 21:04:29	12/07/2010 21:04:41
bids	bids	succeeded	4	1	7703	12/07/2010 20:28:19	12/07/2010 20:31:23
offers	ilreserves	succeeded	4	1	6933	12/07/2010 20:14:11	12/07/2010 20:14:16
offers	ilreserves	warning	2	1	6826	12/07/2010 20:11:10	12/07/2010 20:11:11
offers	ilreserves	succeeded	0	1	6800	12/07/2010 20:10:48	12/07/2010 20:10:48
offers	generatorreserves	succeeded	4	1	6772	12/07/2010 20:10:09	12/07/2010 20:10:16

Done

Running a Pipeline Job

- Simply call the following from the scheduler or from the command line:

```
etlp <config_file> <section>
```

e.g.

```
etlp sales region_sales
```

Two kinds of Jobs

- Serial:
  - tasks are performed in order and the job completes
- Iterative
  - tasks are performed in order, once for each file
- A job can invoke another job upon completion

Bundled Iterative Plugins

- csv_loader: load CSV and other delimited files
- gunzip / gzip: uncompress / compress files
- sql_loader: Load data using Oracle SQL*Loader
- os: Call Operating System command
- perl: Call Perl subroutine
- plsql: Call Oracle stored procedure
- steady_state_check: check a file's steady state
- validate: validate file structure against definition

Bundled Serial Plugins

- os: Call Operating System command
- perl: Call Perl subroutine
- plsql: Call Oracle stored procedure
- watch: Watch for the appearance of files that match the specified pattern

Example…

```
<process_customers>
   type                    = iterative
   <config>
    filename_format     = (customer\d.csv)(?:\.gz)?$

    incoming_dir        = data/incoming
    archive_dir         = data/archive
    fail_dir            = data/fail    table_name
     = stg_customer
        controlfile_dir     = conf/control
        controlfile         = customer.ctl
    on_error            = die
   </config>
```

pre_process

```
<pre_process>
    <item>
          name      = decompress customer file
          type      = gunzip
    </item>
    <item>
          name         = validate customer file
          type         = validate
          file_type    = csv
          skip         = 1
    </item>
</pre_process>
```

process

```
<process>
    <item>
            name = load customer file
            type = csv_loader
            skip = 1
    </item>
</process>
```

post_process

```
  <post_process>
      <item>
            name = compress file
            type = gzip
      </item>
  </post_process>
</customer>
```

Control file

- Defines data file format
- Can also define validation rules for the "validation" plugin
- Only validates individual fields
  - can't aggregate rows
  - can't check one field against another

Example Control File

```
grid_point      N varchar(8)
trading_date    N date(%d/%m/%Y)
trading_period N integer;range(1,50)
market_time     N date(%H:%M)
price           N float
island          N varchar(2)
area            N varchar(2)
market_flag     N varchar(1)
runtime         N date(%d/%m/%Y %H:%M:%S)
```

Validation errors

```
Error processing
/
  home/dhorne/etl/data/incoming/5_minute_prices_WWD1103_20
  100609.csv:
5_minute_prices_WWD1103_20100609.csv failed validation:
Line number: 13
  field name:island
  field value:NNI
  error:Length must be less than or equal to 2 characters
Line number: 30
  field name:trading_date
  field value:09/13/2010
  error:Invalid date for pattern: %d/%m/%Y
```

Validation Rules

Rule	Description
varchar(*n*)	A variable number of characters, up to the value of *n*
integer	An integer value
float	A floating point number
date(<posix format>)	A date or date and time in the specified format
range(<lower>, <upper>)	The value is a numeric and must be between the upper and lower bounds. • range(1,12) • range(0,) • range(,40)
qr//	A regular expression

File Watcher

```
<fw_file>
    type = serial
    <config>
        directory = %app_root%/data/incoming
        call      = bill weekly
    </config>
    <process>
        <item>
            name          = File Name Match
            type          = watch
            duration      = 5h
            file_pattern  = bill.tar.gz
        </item>
    </process>
</fw_file>
```

Steady State Check

- Iterative plugin

```
<item>
   name = bill file check
   type = steady_state_check
   interval = 30
</item>
```

Placeholders

- Any application configuration parameter can be referenced in the items.
- Can use environment configuration parameters if allow_env_vars is true
- Simply use a placeholder:
  - %fail_dir%
- Framework maintains non configuration placeholders:
  - %app_root%
  - %filename%
  - %basename(filename)%

Writing Plugins

- Plugins sub-class ETLp::Plugin
- Tell ETLp the name of your plugin namespace in env.conf:

```
serial_plugin_ns = MyApp::Serial::Plugin
iterative_plugin_ns = MyApp::Iteratve::Plugin
```

Iterative Plugin Template

```
use MooseX::Declare;
class MyApp::Plugin::Iterative::<<Name>> extends
  ETLp::Plugin {

    sub type {
        return '<<type>>';
    }

    method run (Str $filename) {
        <<functionality here>>
        return $filename;
    }
}
```

Load XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<scores>
    <score>
        <id>1</id>
        <name>Smith</name>
        <value>50.5</value>
    </score>
    <score>
        <id>2</id>
        <name>Jones</name>
        <value>30.75</value>
    </score> ... etc...
</scores>
```

```
use MooseX::Declare;
class My::Plugin::Iterative::ScoreXML extends ETLp::Plugin {
    use XML::Simple;
    use File::Copy;
    use File::Basename;

    sub type {
        return 'score_xml';
    }

    method run (Str $filename) {
        my $aud_file_process = $self->audit->item->file_process;
        my $file_id = $aud_file_process->get_canonical_id;
        my $app_config = $self->config->{config};

        my $ref = XMLin($filename, KeyAttr => 'score');

        my $sth = $self->dbh->prepare(
            q{
                insert into scores (
                    id, name, score, file_id
                ) values (?, ?, ?, ?)
            }
        );
```

```
    foreach my $record (@{$ref->{score}}) {
         $sth->execute($record->{id}, $record->{name},
          $record->{value}, $file_id);
    }

    $self->dbh->commit;

     move($filename, $app_config->{archive_dir}) ||
        ETLpException->throw(error => "Unable to move $filename to " .
        $app_config->{archive_dir} . ": $!");

        return $app_config->{archive_dir} . '/' . basename($filename);
    }
}
```

Item uses new type

```
<process>
    <item>
        name = load score file
        type = score_xml
    </item>
</process>
```

Features to Add

- Scheduler – web-based interface for creating cron jobs
- Interface to MySQL and Infobright loaders
- Call MySQL stored procedures

In Closing

- Project code, bug db and documentation available at:
  - http://firefly.activestate.com/dhorne/etlp
- Keen to get users and feedback