Building the PRONIR Conversion Clearinghouse.

Eric D. Schabell
ericS@cs.kun.nl

University of Nijmegen, Computing Science Institute, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

PUBLISHED AS:

E.D. Schabell. Building the pronir conversion clearinghouse. Technical Report NIII-R0317, Nijmegen Institute for Information and Computing Sciences, University of Nijmegen, Nijmegen, The Netherlands, EU, 2003.

Abstract

As a vital part of the larger architecture, our document conversion system will need to provide for the transformation of documents to a requested format. The first step on the way to realizing this system is to gather all the various conversion routines in a central location. To achieve this, the *Conversion Clearinghouse* has been implemented to provide a means for anyone to submit her favorite conversion routines. Our *Conversion Clearinghouse* also allows anyone to browse through and pick out any of the collected conversions. [1]

The following report describes the design and implementation of the various elements that make up our *Conversion Clearinghouse*. This work is part of the ongoing research project *Profile Based Retrieval Of Networked Information Resources (PRONIR)*.

1 Overview

Our design shows a split into *Phases*, which allow for a working evolution towards the final networked document conversion toolkit. For clarity, an overview of all four *Phases* will be provided first with a detailed explanation of the initial *Phase zero* being covered in the rest of this report.

1.1 Phase zero

A web based portal will be provide to offer access to a collection of existing one step conversion routines. It will also provide a mechanism for submitting conversion routines for addition to the collection. Finally, this web portal will provide a means for a browsing client to obtain needed conversion tools from the conversion clearinghouse collection.

[1] Conversion Clearinghouse: `http://www.pronir.nl/clearinghouse`.

1.2 Phase one

Phase one will add to the functionality of the conversion clearinghouse. Here we want to build a simple document conversion system that can make use of the one step conversion tools available in phase zero.

A Broker and Server interface will be created to provide the user with a seamless conversion experience. This can occur locally or invisibly via a networked conversion server.

The various existing conversions available in the conversion clearinghouse are the starting point for our offered conversion system. This will involve only single step conversions, ones that move a document from one type to another. For example, postscript to pdf format.

1.3 Phase two

This phase will expand on the previous one to include additional *recipes*. A conversion process should not only take a document from a to b, but also provide for possible a to b to ... to n. The client will be able to make use of these conversions without regard for quality of the conversion.

1.4 Phase three

At this stage, we will provide for a more complex management of the conversion process between a user and one or more servers in the conversion network.

The Broker will be expanded to provide for more functionality. The network will be able to host multiple Servers with each participating Server providing a listing of its own conversion services.The Broker will be able to negotiate for the client in a network of conversion servers, attempting to offer the quickest (distance to server) conversion. The Broker will offer the most efficient (quality comes into play) conversion possible.

Finally, conversion results should be cached. This caching will not be of the converted document (higher level component caching), but at a more elemental level (low level component caching) as described in [Sch02].

2 Introduction

Our search for a a generic information retrieval software architecture has been outlined previously in [Sch02]. Continuing work on this project has resulted in [GPB03b], [GS03] and [GPB03a].

The ultimate goal of this part of the overall software architecture is to provide a comprehensive document conversion system, one that works both locally and across networked environments. To meet this goal, the four previously defined *Phases* have been defined to allow for incremental evolution of the provided conversion functionality.

This technical report will deal only with the initial *Phase zero*, which details the setup of a central repository for the existing conversion tools available for a Linux machine.

2.1 Phase zero

Our first goal will be to create a central repository for the existing conversion tools available for a Linux machine. We call our repository *The PRONIR Conversion Clearinghouse* and it consists of the following elements:

- a single conversion will be *one-step* only, defined as converting a document from one type directly to another and no further.
- a collection of existing filters and wrappers that can make *one-step* conversions between document types (ps2pdf for example).
- provide a means for searching for a conversion based on feature type and representation type information.
- using a web interface to provide links for the client browsing to obtain the variou s files necessary to implement these conversions.

2.2 Implementation

A web based portal has been created which provides access to all of the above mentioned services. It provides access to our collection of existing one step conversion routines, provide a simple web tool for submitting conversions and provide a means for a browsing client to obtain needed conversion tools from the conversion clearinghouse collection.

3 Design

This section will outline the requirements, data model, relational database scheme, database scheme and other issues that were encountered during the design phase.

3.1 Requirements

Requirements were ascertained through the use of interviews and brainstorming sessions with part of the PRONIR project team. The following shows the functionality that our *Conversion Clearinghouse* will need to provide:

- a single conversion will be *one-step* only, defined as converting a document from one type directly to another and no further.
- a collection of existing filters and wrappers that can make *one-step* conversions between document types (ps2pdf for example).
- provide a means for searching for a conversion based on *feature type* and *representation type* information.[2]
- using a web interface to provide links for the client browsing to obtain the various files necessary to implement these conversions.

[2] *Feature types* and *representation types* are introduced in [GPB03a].

Based on the above listed needs the following functional requirements were the basis for further development:

1. user can interface with the clearinghouse via web browser.
2. user can submit a new conversion to the clearinghouse.
3. conversions should be browsable by user.
4. browsing queries should always provide as many results as possible.
5. user can obtain a desired conversion tool from the clearinghouse.

To support these a web based portal has been created. It provides access to our collection of existing one step conversion routines, provides a simple web tool for submitting conversions and provides a means for browsing clients to obtain conversion tools from the *Conversion Clearinghouse* collection. [3]

3.2 Data Model

To support the *Conversion Clearinghouse* in its retrieval and browsing functionality, a data analysis was conducted. The results were modeled in the entity-relationship (ER) diagram shown in Figure 1.

3.3 Relational Database Scheme

Mapping the ER model in Figure 1 to a relational database schema and then normalizing provides a structured data model as show in the following three tables [4]:

Package

name	`contact`	`orig-url`	`pronir-url`

The Package schema has a primary key consisting of the package name and is referenced by the *pkg-name* attribute from the Routine table.

Routine

name	***pkg-name***	`location`

The Routine schema has a double attribute key **name, pkg-name** and is referenced by the field *routine-name* from the Filter table.

Filter

cmd-string	***routine-name***
`from-feature-type`	`from-representation-type`
`to-feature-type`	`to-representation-type`

The Filter schema has a double attribute key **cmd-string, routine-name**.

[3] Conversion Clearinghouse: `http://www.pronir.nl/clearinghouse`.
[4] The **bold** fields denote primary keys, the *italics* denote foreign keys.

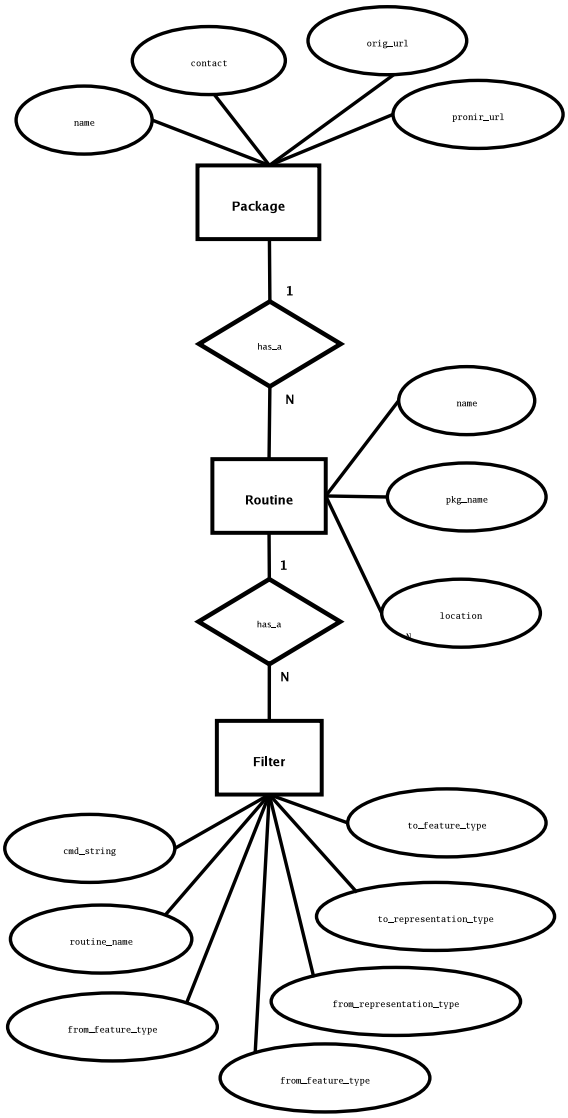

Figure 1: Conversion Clearinghouse *entity-relationship (ER)* data model

3.4 Database choices

When looking at our choices to implement our data model, we decided to keep it within the Free Software databases. Keeping in mind that it is a search for a *generic architecture* it was decided not to limit the implementation choices. Two options in the form of setup scripts are provided for the following databases:

1. Postgresql [Pos03]
2. MySQL [MyS03]

Here is an example of the table definitions for an implementation using MySQL as the database for a *Conversion Clearinghouse*:

MySQL table definitions

- CREATE TABLE Package (name VARCHAR(20) NOT NULL, contact VARCHAR(20) NOT NULL, orig-url VARCHAR(40), pronir-url VARCHAR(40), PRIMARY KEY (name)
- CREATE TABLE Routine (name VARCHAR(20) NOT NULL, pkg-name VARCHAR (30) NOT NULL REFERENCES Package(name), location VARCHAR(30) NOT NULL, PRIMARY KEY (name, pkg-name))
- CREATE TABLE Routine (name VARCHAR(20) NOT NULL, pkg-name VARCHAR (30) NOT NULL REFERENCES Package(name), location VARCHAR(30) NOT NULL, PRIMARY KEY (name, pkg-name))

3.5 Implementation

The web interface was developed through the use of the *Rapid Application Development (RAD)* methodology. The first interface was quickly prototyped in Perl and through the various iterations of user analysis it evolved into the site it is at:

`http://www.pronir.nl/clearinghouse`

An overview of the possible screens a user can encounter during a visit to the *Conversion Clearinghouse* is shown in Figure 2.

1. **Intro Screen -** displays three choices (listing of database, submit a conversion, retrieve a conversion) in the form of check boxes with a submit button for the user to indicate her choice.
2. **Listing -** if the user request to retrieve information about a specific conversion, then a new screen will be provided. This screen will provide pull-down menu buttons for the following fields: conversion name, from aspect, to aspect, from type, to type.
3. **Listing Results -** a new screen that displays the results of the user selected options in the previous Listing screen.

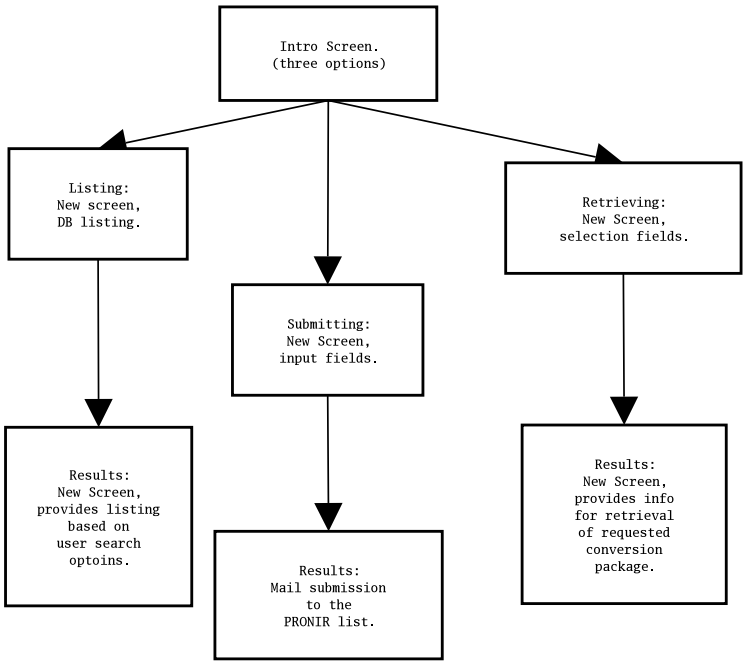

Figure 2: User screen flow experience

4. **Submitting -** if the user request to submit a conversion, then a new screen will be displayed with the following fields for user input: name of conversion, from aspect, to aspect, from type, to type, execution command, package name, URL to package source. Each field will have it's own explanation text to inform the user as to what type of input is required. This screen provides a submit button for the user to indicate submission of the provided fields.

5. **Submit Results -** the screen is updated with a *sent* or *denied* message, depending on the results of the users attempt to submit a conversion.

6. **Retrieving -** if the user requests to retrieve information about a specific conversion, a new screen will be provided. This screen will provide a pull down menu listing all available conversion routines from which the user can make a selection.

7. **Retrieval Results -** a new screen, providing information from the conversion database as to the package needed for the conversion type selected. The user can download any listed packages from this screen by selecting one.

For a more detailed look at the code itself, the reader is referred to section 4.

3.6 System requirements

The running of the pronir clearinghouse phase zero will require at least the following components (this is the environment in which it was created and tested):

- Linux or Unix system that supports all of the following.
- Perl 5.8.0
- Perl modules used in cgi scripts (imported; CGI, DBI, Mail::Sendmail and depending on database choice; DBD-Pg for Postgresql or DBD-mysql for MySQL).
- Apache 2.0.44 or higher with CGI support.
- MySQL 4.0.12 (must have 4.0 or higher to use the UNION feature in our queries).
- Postgresql 7.3.3

Note that you have a choice to make between MySQL and Postgresql.

3.7 Developers setup

This section will detail the various locations of the source code tree and how to access it for development purposes.

First off the Perl cgi scripts are all located in an off-site *Code Version System (CVS)*. To check out the code you will need to have CVS installed on your system. You will also need an account on the CVS host machine, this can be arranged through Eric Schabell, *erics@cs.kun.nl* or Kees Leune, *kees@uvt.nl*. Once you have an account, you can check out the pronir module with the following command:

```
cvs-dmaximux.uvt.nl:/home/cvscheckoutpronir
```

The phase zero clearinghouse source code tree and documentation can be found in the following directories:

- pronir/prj/pronir-clearinghouse/src
- pronir/prj/pronir-clearinghouse/docs

4 Implementation

In this section the coded implementations of the previously shown design will be provided. Where these implementations are related to a certain screen as shown in Figure 2, it will be noted in the caption supplied at the top of each code frame.

All code examples have been cleaned of the integrated documentation formating tags. It should be noted that care was taken to document the code completely.

Listing 1: Clearinghouse Intro

```
#!/usr/bin/perl -Tw
#
# PROGRAM NAME - clearinghouse
#
# Author   : Eric Schabell
# Date     : 2002-12-23
# E-Mail   : erics@cs.kun.nl
#
# Changelog :
#
#   20030624 -  Release 1.0
#
use strict;
use vars '$VERSION';
use CGI qw(:all);

$VERSION = "1.0";

#
# Set these to reflect the server being used.
#
my $submit_url   = "http://localhost/cgi-bin/submit_conversion";
my $list_url     = "http://localhost/cgi-bin/listing_request";
my $obtain_url   = "http://localhost/cgi-bin/obtain_conversion";

my $list_choice    = "Browse available conversions";
my $submit_choice    = "Submit a conversion routine";
my $obtain_choice    = "Obtain a conversion routine";
my $submit_button    = "options";

# Apparently every cgi script runs itself again after submit button is pressed,
# so will make the checks for selection first and if this is the first run it
# will print the front page to allow for user input.  Otherwise it will react to
# the button selected and process the users request... <sigh>, web
# development ... wheeeee....
#
if (param($submit_button) =~ /^$submit_choice/) {

  print redirect($submit_url);

} elsif (param($submit_button) =~ /^$list_choice/) {

  print redirect($list_url);

} elsif (param($submit_button) =~ /^$obtain_choice/) {

  print redirect($obtain_url);

} else {

  print header,
    start_html(-title=>'Welcome to - The PRONIR Conversion
      Clearinghouse!',
    -author=>'erics@cs.kun.nl',
    -bgcolor=>'lightgrey'),
    "<CENTER>\n",
    h1('***** PRONIR Conversion Clearinghouse *****'),
    "</CENTER>\n";
  print_header();
  print hr,
    start_form,
    p,
    h2('Please select an option:'),
    radio_group(-name=>$submit_button,
    -values=>[$list_choice,
    $submit_choice,
    $obtain_choice],
    -default=>'-'),
    p,
    submit,
    end_form,
    hr,
```

```
    end_html;
}

sub print_header {

  # header and footer.
    #
  print "<!--  Download links at the top of the page -->",
    "<table width=\"100%\" border=\"2\">",
    "<th width=\"100%\" colspan=4 height=\"2%\" rowspan=1
      align=\"center\" nowrap>",
    "<a href=\"clearinghouse\"
      target=\"_new\">Conversion_Clearinghouse  -</a>",
    "<a href=\"http://www.pronir.nl\" target=\"_new\"> Pronir_Home  -
      </a>",
    "<a href=\"http://www.pronir.nl/pub/spws/download.html\"
      target=\"_new\"> Download  -</a>",
    "<a href=\"http://www.pronir.nl/pub/spws/docs/\"
      target=\"_new\"> Documentation  -</a>",
    "<a href=\"http://www.pronir.nl/pub/spws/call.html\"
      target=\"_new\"> Call for Interest </a>",
    "</th>",
    "</table>",
    "</CENTER>\n";
}

# end clearinghouse.
```

Listing 2: Listing Request

```
#!/usr/bin/perl -Tw
#
# PROGRAM NAME - listing_request
#
# Author   : Eric Schabell
# Date     : 2003-01-08
# E-Mail   : erics@cs.kun.nl
#
# Changelog :
#
#   20030624 -  Release 1.0
#
use strict;
use vars '$VERSION';
use CGI qw(:all);
use DBI;

$VERSION = "1.0";
#
# Set these to reflect the server being used.
#
my $clearinghouse_db  = "clearinghouse";
my $db_driver      = "dbi:Pg:dbname";  # or "DBI:mysql:database" for MySQL.
#my $db_host       = "pgsql-clearinghouse.sci.kun.nl";
#my $db_user       = "clearinghouse-user";
#my $db_pw         = "ClearAtHouse";
my $db_host        = "localhost";
my $db_user        = "pronir";
my $db_pw        = "pron!r";

#
# Display the various conversion tables.
#
my $from_feature     = "From_Feature";
my $from_rep       = "From_Representation";
my $to_feature       = "To_Feature";
my $to_rep         = "To_Representation";
my (@from_feature_list,
  @from_representation_list,
  @to_feature_list,
  @to_representation_list) = ();
my $f_feature_select  = "SELECT DISTINCT from_feature_type
              FROM Filter
```

```
                ORDER BY from_feature_type";
my $f_representation_select = "SELECT DISTINCT from_representation_type
                FROM Filter
                ORDER BY from_representation_type";
my $t_feature_select   = "SELECT DISTINCT to_feature_type
                FROM Filter
                ORDER BY to_feature_type";
my $t_representation_select = "SELECT DISTINCT to_representation_type
                  FROM Filter
                  ORDER BY to_representation_type";

#
# Apparently every cgi script runs itself again after submit button is pressed,
# so will make the checks for selection first and if this is the first run it
# will print the front page to allow for user input.  Otherwise it will react to
# the button selected and process the users request... <sigh>, web
# development... wheeeee....
#
if (param()) {

  my $ff = param($from_feature);
  my $fr = param($from_rep);
  my $tf = param($to_feature);
  my $tr = param($to_rep);

  print
  redirect("http://localhost/cgi-bin/
      results_listing?from_f=$ff&from_r=$fr&to_f=$tf&to_r=$tr");

} else {

  my $dbh = DBI->connect("$db_driver=$clearinghouse_db;host=$db_host",
    $db_user,
    $db_pw,
    {'RaiseError' => 1});

  #
  # Display query page.
  #
  print header,
    start_html(-title=>'Welcome to - The PRONIR Conversion Clearinghouse!',
    -author=>'erics@cs.kun.nl',
    -bgcolor=>'lightgrey'),
    "<CENTER>\n",
    h1('***** Browsing the PRONIR Conversion Clearinghouse  *****');
  print_header();
  print hr;


  #
  # Setup lists for selection menu's.
  #

  # From Feature Listing.
  #
  my $sth_f_feature_list = $dbh->prepare($f_feature_select);
  $sth_f_feature_list->execute();
  while (my $ref = $sth_f_feature_list->fetchrow_hashref()) {
    @from_feature_list = (@from_feature_list, $ref->{'from_feature_type'});
  }
  $sth_f_feature_list->finish();

  # From Representation Listing.
  #
  my $sth_f_representation_list = $dbh->prepare($f_representation_select);
  $sth_f_representation_list->execute();
  while (my $ref = $sth_f_representation_list->fetchrow_hashref()) {
    @from_representation_list =
      (@from_representation_list, $ref->{'from_representation_type'});
  }
  $sth_f_representation_list->finish();

  # To Feature Listing.
  #
```

```
  my $sth_t_feature_list = $dbh->prepare($t_feature_select);
  $sth_t_feature_list->execute();
    while (my $ref = $sth_t_feature_list->fetchrow_hashref()) {
      @to_feature_list = (@to_feature_list, $ref->{'to_feature_type'});
    }
  $sth_t_feature_list->finish();

  # To Representation Listing.
  #
  my $sth_t_representation_list = $dbh->prepare($t_representation_select);
  $sth_t_representation_list->execute();
    while (my $ref = $sth_t_representation_list->fetchrow_hashref()) {
      @to_representation_list = (@to_representation_list,
                    $ref->{'to_representation_type'});
    }
  $sth_t_representation_list->finish();


  print start_form,
    h2('Choose the Feature and/or Representation types that
      represent your conversion wishes: '),
    br('Conversion ',
    b('FROM FEATURE'),
    ' type: '),
    popup_menu(-name=>$from_feature,
    -values=>\@from_feature_list,
    -default=>'*'),
    br('Conversion ',
    b('FROM REPRESENTATION'),
    ' type: '),
    popup_menu(-name=>$from_rep,
    -values=>\@from_representation_list,
    -default=>'*'),
    p,
    br('Conversion ',
    b('TO FEATURE'),
    ' type: '),
    popup_menu(-name=>$to_feature,
    -values=>\@to_feature_list, -default=>'*'),
    br('Conversion ',
    b('TO REPRESENTATION'),
    ' type: '),
    popup_menu(-name=>$to_rep,
    -values=>\@to_representation_list,
    -default=>'*'),
    p,
    submit,
    end_form,
    hr,
    end_html;

  $dbh->disconnect();
  print_help();
}


sub print_help {

  print "<Font=-5>\n",
    ('Definitions: '),
    br, em('Feature Types -> '),
    ('  related to a document type such as "full-content", "keywords"
      or a *  meaning that the conversion is able to handle any type.'),
    br,
    em('Representation Types -> '),
    (' related to the underlying aspect of a given document, such
      as "ps", "pdf", "cmd" such as the unix pipe, * for any or
      just plain text "ascii".'),
    hr;
    return;
}


sub print_header {
```

```
  # header and footer.
  #
  print "<!--  Download links at the top of the page -->",
    "<table width=\"100%\" border=\"2\">",
    "<th width=\"100%\" colspan=4 height=\"2%\" rowspan=1
      align=\"center\" nowrap>",
    "<a href=\"clearinghouse\" target=\"_new\">
      Conversion_Clearinghouse  -</a>",
    "<a href=\"http://www.pronir.nl\"
      target=\"_new\"> Pronir_Home  -</a>",
    "<a href=\"http://www.pronir.nl/pub/spws/download.html\"
      target=\"_new\"> Download  -</a>",
    "<a href=\"http://www.pronir.nl/pub/spws/docs/\"
      target=\"_new\"> Documentation  -</a>",
    "<a href=\"http://www.pronir.nl/pub/spws/call.html\"
      target=\"_new\"> Call for Interest </a>",
    "</th>",
    "</table>",
    "</CENTER>\n";
}

# end listing_request.
```

Listing 3: Results Listing

```
#!/usr/bin/perl -Tw
#
# PROGRAM NAME - results_listing
#
# Author   : Eric Schabell
# Date     : 2003-02-07
# E-Mail   : erics@cs.kun.nl
#
# Changelog :
#
#   20030711 -  Bug fix in ORDER BY, nu 'name' was table column name.
#
use strict;
use vars '$VERSION';
use CGI qw(:all);
use DBI;

$VERSION = "1.0";

#
# set these url's to right locations!
#
my $GNU_URL        = "Check your local GNU mirror.";
my $PRONIR_URL       = "http://www.pronir.nl/pub/spws/mirror/";
my $clearinghouse_db   = "clearinghouse";
my $db_driver      = "dbi:Pg:dbname";  # or "DBI:mysql:database" for MySQL.
#my $db_host       = "pgsql-clearinghouse.sci.kun.nl";
#my $db_user         = "clearinghouse-user";   # user only select access to db.
#my $db_pw         = "ClearAtHouse";
my $db_host        = "localhost";
my $db_user        = "pronir";   # user only has select access to data tables.
my $db_pw        = "pron!r";

my $dbh = DBI->connect("$db_driver=$clearinghouse_db;host=$db_host",
            $db_user, $db_pw, {'RaiseError' => 1});

print header,
  "<CENTER>\n",
  start_html(-title=>'Welcome to - The PRONIR Conversion Clearinghouse!',
         -author=>'erics@cs.kun.nl',
         -bgcolor=>'lightgrey'),
  h1('***** Browsing the PRONIR Conversion Clearinghouse *****'),
  "</CENTER>\n";

print_header();
print hr;
process_query(param('from_f'), param('from_r'), param('to_f'), param('to_r'));
```

```
print_header();
print end_html;
$dbh->disconnect();


#
# Subroutine: process_query.
#
sub process_query {

  my $from_feature  = shift;
  my $from_rep      = shift;
  my $to_feature    = shift;
  my $to_rep        = shift;
  my ($ff, $fr, $tf, $tr);
  for ($from_feature) {

    if    (/\*/)        {$ff = "*"}
    elsif (/key-words/)   {$ff = "*"}
    elsif (/full-content/)  {$ff = "*"}
    else              {$ff = $from_feature}

  }

  for ($from_rep) {

    if    (/\*/)  {$fr = "*"}
    else      {$fr = $from_rep}
  }

  for ($to_feature) {

    if    (/\*/)        {$tf = "*"}
    elsif (/key-words/)   {$tf = "*"}
    elsif (/full-content/)  {$tf = "*"}
    else              {$tf = $to_feature}
  }

  for ($to_rep) {

    if    (/\*/)  {$tr = "*"}
    else      {$tr = $to_rep}
  }

  submit_query($ff, $fr, $tf, $tr);
}


#
# Subroutine: submit_query.
#
sub submit_query {

  my $ff = shift;
  my $fr = shift;
  my $tf = shift;
  my $tr = shift;
  my $i  = 0;     # counter
  my $select = "SELECT DISTINCT Routine.*, Filter.from_feature_type,
              Filter.from_representation_type,
              Filter.to_feature_type,
              Filter.to_representation_type,
              Package.pronir_url
         FROM Routine, Filter, Package
         WHERE ";
  my $query = $select;

    # work out the From Format first.
  if (!($ff =~ /\*/)) {

    # the feature type has been modified.
    if (!($fr =~ /\*/)) {

      # the rep type has also been modified.
```

```
    $query = $query."Filter.from_feature_type='$ff' AND ";
    $query = $query."Routine.name = Filter.routine_name AND
                Routine.pkg_name = Package.name ";
    $query = $query."UNION ";
    $query = $query.$select;
    $query = $query."Filter.from_representation_type='$fr' AND ";
    $query = $query."Routine.name = Filter.routine_name AND
                 Routine.pkg_name = Package.name ";
    $i++;

  }
  else {

    # only the feature type has been modified.
    $query = $query."Filter.from_feature_type='$ff' AND ";
    $query = $query."Routine.name = Filter.routine_name AND
                Routine.pkg_name = Package.name ";
    $i++;
  }
}
elsif (!($fr =~ /\*/)) {

  # only the rep type has been modified.
  $query = $query."Filter.from_representation_type='$fr' AND ";
  $query = $query."Routine.name = Filter.routine_name AND
                Routine.pkg_name = Package.name ";
  $i++;

}
else {

  # both types are '*', go to TO types.
}

  # now work our the To Format.
if (!($tf =~ /\*/)) {

  # the feature type has been modified.
  if (!($tr =~ /\*/)) {

    # the rep type has also been modified.
    if ($i > 0) {

      $query = $query." UNION ";
      $query = $query.$select;
    }
    $query = $query."Filter.to_feature_type='$tf' AND ";
    $query = $query."Routine.name = Filter.routine_name AND
                Routine.pkg_name = Package.name ";
    $query = $query."UNION ";
    $query = $query.$select;
    $query = $query."Filter.to_representation_type='$tr' AND ";
    $query = $query."Routine.name = Filter.routine_name AND
                Routine.pkg_name = Package.name ";
    $i++;

  }
  else {

    # only the feature type has been modified.
    if ($i > 0) {

      $query = $query." UNION ";
      $query = $query.$select;
    }
    $query = $query."Filter.to_feature_type='$tf' AND ";
    $query = $query."Routine.name = Filter.routine_name AND
                Routine.pkg_name = Package.name ";
    $i++;
  }
}
elsif (!($tr =~ /\*/)) {

  # only the rep type has been modified.
```

```
    if ($i > 0) {

      $query = $query." UNION ";
      $query = $query.$select;
    }
    $query = $query."Filter.to_representation_type='$tr' AND ";
    $query = $query."Routine.name = Filter.routine_name AND
                Routine.pkg_name = Package.name ";
    $i++;

  }
  else {

    # both types are '*', so do nothing.
  }

  if ($i == 0) {

    # Displaying complete listing of conversions.
    #
    $query = $query."Routine.name = Filter.routine_name AND
                Routine.pkg_name = Package.name ";
  }
  $query = $query."ORDER BY name";
  display($query);
}


#
# Subroutine: display.
#
sub display {

  my $received = shift;
  my $sth_query = $dbh->prepare($received);
  $sth_query->execute();

  print h2('These conversion routines might help you with
      your conversion: '),
    "<TABLE width=90% border=5>\n",
        "<TR>\n",
      "<td width=200><B>Name</td>\n",
            "<td width=200><B>Package Name</td>\n",
            "<td width=200><B>From Feature Type</td>\n",
            "<td width=200><B>From Representation Type</td>\n",
            "<td width=200><B>To Feature Type</td>\n",
            "<td width=200><B>To Representation Type</td>\n",
      "<td width=200><B>Pronir URL</td>\n",
        "<\TR>\n";

  while (my $ref = $sth_query->fetchrow_hashref()) {

    print "<TR>\n",
      "<td width=200>$ref->{'name'}</td>\n",
            "<td width=200>$ref->{'pkg_name'}</td>\n",
      "<td width=200>$ref->{'from_feature_type'}</td>\n",
      "<td width=200>$ref->{'from_representation_type'}</td>\n",
      "<td width=200>$ref->{'to_feature_type'}</td>\n",
      "<td width=200>$ref->{'to_representation_type'}</td>\n";

    if ($ref->{'pronir_url'} =~ /\$PRONIR/) {

      print "<td width=300>
          <a href=\"${PRONIR_URL}$ref->{'pkg_name'}.tar.bz2\"
              >Download pkg here</a></td>\n";
    } else {
          print "<td width=300><a href=\"$ref->
          {'pronir_url'}$ref->{'pkg_name'}.tar.bz2\"
          >Download pkg here</a></td>\n";
    }
    print "</TR>\n";
  }

  print "</TABLE>\n",
```

```
    br,
    em('*If this table is empty, there were no results
      matching your selections...');
  $sth_query->finish();
}

sub print_header {

  # header and footer.
    #
    print "<!--  Download links at the top of the page -->",
    "<table width=\"100%\" border=\"2\">",
        "<th width=\"100%\" colspan=4 height=\"2%\" rowspan=1
      align=\"center\" nowrap>",
        "<a href=\"clearinghouse\" target=\"_new\"
      >Conversion_Clearinghouse  -</a>",
        "<a href=\"http://www.pronir.nl\"
      target=\"_new\"> Pronir_Home  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/download.html\"
      target=\"_new\"> Download  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/docs/\"
      target=\"_new\"> Documentation  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/call.html\"
      target=\"_new\"> Call for Interest </a>",
        "</th>",
        "</table>",
        "</CENTER>\n";
}

# end results_listing.
```

Listing 4: Submitting Conversion

```
#!/usr/bin/perl -Tw
#
# PROGRAM NAME - submit_conversion
#
# Author   : Eric Schabell
# Date     : 2003-01-06
# E-Mail   : erics@cs.kun.nl
#
# Changelog :
#
#   20030624 -  Release 1.0
#
use strict;
use vars '$VERSION';
use CGI qw(:all);
use Mail::Sendmail;

$VERSION = "1.0";

#
# Please set a valid server here.
#
unshift @{$Mail::Sendmail::mailcfg{'smtp'}}, 'smtp-srv.cs.kun.nl'; #kun
#unshift @{$Mail::Sendmail::mailcfg{'smtp'}}, 'kubsus.kub.nl'; #uvt

#
# set urls for correct server environment.
#
#my $display_submit_url= "http://localhost/cgi-bin/display_submit";

#
# CGI html form below.
#
my @type_list          = ("", "ps", "ascii", "pdf");
my @aspect_list        = ("", "*", "full-content", "keywords");
my $type_from_menu_value  = "From Type";
my $aspect_from_menu_value  = "From Aspect";
my $type_to_menu_value     = "To Type";
my $aspect_to_menu_value  = "To Aspect";
```

```
#
# create submission form.
#
print header,
  start_html(-title=>'Welcome to - The PRONIR Conversion Clearinghouse!',
        -author=>'pronir-conversion@cs.kun.nl',
        -bgcolor=>'lightgrey'),
  ("<CENTER>\n"),
  h1('***** PRONIR Conversion Submit Form *****'),
  ("</CENTER>\n");

print_header();
print hr,
  ("To submit a conversion to the PRONIR Clearinghouse, please fill in
    the following form."),
  br("This form will be submitted to our administration, you will be
    contacted upon acceptance/denial."),
  p("Thank you for contributing to the PRONIR project!"),
  hr,
  start_form,
  h3('Conversion submission form: '),
  ('Please fill in the name of the conversion tool/application:   '),
  textfield(-name=>'name',
        -size=>37),
  p('Please enter the location of the source or website (URL):   ',
    textfield(-name=>'source site',
          -size=>37)),
  p('Please enter your contact e-mail, only valid entries will be sent:   ',
    textfield(-name=>'address',
          -size=>31)),
  p,
  submit,
  end_form,
  hr,
  end_html;

if (param()) {

  #
  # The only important security measure is to ensure that the email is
  # valid and not messing with strange characters (like *<>|).
  #
  my $mail_from = param('address');
  my $tool_name = param('name');
  my $location  = param('source site');

  if ((not $mail_from) or ($mail_from !~ /\@/)) {

    print h2('Please enter a valid e-mail address...');
  }
  elsif ($mail_from =~ tr/;<>?|*'&$!#[]%{}:'"//) {

    print h2('Please enter a valid e-mail address and quit being sneaky...');
  }
  elsif (not $tool_name) {

    print h2('Please enter the name of the conversion tool/application...');
  }
  elsif ((not $location) or ($location !~ /http:\/\/\w+\.\w+/)) {

    print h2('Please enter a valid URL as the sources location...');
  }
  else {

    my %mail = (
          To      => 'pronir-conversion@cs.kun.nl',
          From    => 'pronir-clearinghouse@cs.kun.nl',
      #Bcc     => 'Someone <him@there.com>, Someone else her@there.com',
          Subject => "A PRONIR Clearinghouse submission from $mail_from",
          'X-Mailer' => "Pronir Clearinghouse Mail::Sendmail submission",
      Message => "conversion tool: $tool_name
      source location: $location
      submit from    : $mail_from"
```

```
      );

      if (sendmail %mail) {

        print h2('Mail sent to PRONIR Clearinghouse!');
        #
        # can turn this on for logging by writing to file.
        #
        #print "\n\$Mail::Sendmail::log says:\n", $Mail::Sendmail::log;

      }
      else {

        print h2("Error sending mail: $Mail::Sendmail::error");
      }
  }
}

sub print_header {

  # header and footer.
    #
    print "<!--  Download links at the top of the page -->",
      "<table width=\"100%\" border=\"2\">",
        "<th width=\"100%\" colspan=4 height=\"2%\" rowspan=1
      align=\"center\" nowrap>",
        "<a href=\"clearinghouse\"
      target=\"_new\">Conversion_Clearinghouse  -</a>",
        "<a href=\"http://www.pronir.nl\"
      target=\"_new\"> Pronir_Home  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/download.html\"
      target=\"_new\"> Download  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/docs/\"
      target=\"_new\"> Documentation  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/call.html\"
      target=\"_new\"> Call for Interest </a>",
        "</th>",
        "</table>",
        "</CENTER>\n";
}

# end submit_converion.
```

Listing 5: Retrieving Conversions

```
#!/usr/bin/perl -Tw
#
# PROGRAM NAME - obtain_conversion
#
# Author   : Eric Schabell
# Date     : 2003-01-08
# E-Mail   : erics@cs.kun.nl
#
# Changelog :
#
#   20030624 -  Release 1.0
#
use strict;
use vars '$VERSION';
use CGI qw(:all);
use DBI;

$VERSION = "1.0";

my @conversion_list   = ();
my $conversion_names  = "Conversions";
my $clearinghouse_db    = "clearinghouse";
my $db_driver     = "dbi:Pg:dbname";  #or "DBI:mysql:database" for MySQL.
#my $db_host          = "pgsql-clearinghouse.sci.kun.nl";
#my $db_user            = "clearinghouse-user";
#my $db_pw              = "ClearAtHouse";
my $db_host         = "localhost";
my $db_user           = "pronir";
```

```
my $db_pw                = "pron!r";

my $list_select          = "SELECT DISTINCT name FROM Routine";

#
# Apparently every cgi script runs itself again after submit button is
# pressed, so will make the checks for selection first and if this is
# the first run it will print the front page to allow for user input.
# Otherwise it will react to the button selected and process the users
# request... <sigh>, web development... wheeeee....
#
if (param()) {

        my $cl = param($conversion_names);
        print redirect("http://localhost/cgi-bin/obtaining_results?conversion=$cl");

}
else {

  #
  # Setup list for selection menu.
  #
  my $dbh = DBI->connect("$db_driver=$clearinghouse_db;host=$db_host",
              $db_user, $db_pw, {'RaiseError' => 1});

  my $sth_list = $dbh->prepare($list_select);
  $sth_list->execute();
  while (my $ref = $sth_list->fetchrow_hashref()) {
    @conversion_list = (@conversion_list, $ref->{'name'});
  }
  $sth_list->finish();

  print header,
    start_html(-title=>'Welcome to - The PRONIR Conversion Clearinghouse!',
           -author=>'erics@cs.kun.nl',
           -bgcolor=>'lightgrey'),
    "<CENTER>\n",
    h1('***** Obtaining a conversion from the PRONIR Conversion
      Clearinghouse *****'),
    "</CENTER>\n";

  print_header();
  print hr,
    start_form,
    h2('To obtain a conversion, please select from the list below: '),
    p,
    br('Clearinghouse conversions: '),
    popup_menu(-name=>$conversion_names,
           -values=>\@conversion_list,
             -default=>'-'),
    p,
    submit,
    end_form,
    hr,
    end_html;

  $dbh->disconnect();
}


sub print_header {

  # header and footer.
    #
    print "<!--  Download links at the top of the page -->",
      "<table width=\"100%\" border=\"2\">",
        "<th width=\"100%\" colspan=4 height=\"2%\" rowspan=1
      align=\"center\" nowrap>",
        "<a href=\"clearinghouse\"
      target=\"_new\">Conversion_Clearinghouse  -</a>",
        "<a href=\"http://www.pronir.nl\"
      target=\"_new\"> Pronir_Home  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/download.html\"
      target=\"_new\"> Download  -</a>",
```

```
        "<a href=\"http://www.pronir.nl/pub/spws/docs/\"
      target=\"_new\"> Documentation  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/call.html\"
      target=\"_new\"> Call for Interest </a>",
        "</th>",
        "</table>",
        "</CENTER>\n";
}

# end obtain_conversion.

=head1 Obtain Conversion

F<obtain_conversion> - CGI script that allows a user to select a conversion
routine from the PRONIR Conversion Clearinghouse.

=head1 Version

This document refers to version <1.0> of F<obtain_conversion>,
released on 24.06.2003.

=head1 Usage

Can be reached via a submit from the the F<clearinghouse> file via
URL/cgi-bin/clearinghouse and selecting the obtain a conversion routine
option.

=head1 Description

F<obtain_conversion> gives the user a pull-down menu from which to select
an available conversion routine from the conversions available.

=head2 Overview

This script allows for conversion routine selection.

=head1 Environment

Not reliant on any environment variables.

=head1 Bugs

Description of known bugs (and any workarounds).
Please send any information about bugs to <erics@cs.kun.nl>.

=head1 Files

This file is reached from the F<clearinghouse> file and will lead to the
F<obtaining_results> file.

=head1 See Also

Documentation for F<clearinghouse> and F<obtaining_results>.

=head1 Author

Eric D. Schabell
  erics@cs.kun.nl

=head1 Copyright

Copyright (c) 2003, Eric D. Schabell. All Rights Reserved.
This module is free software. It may be used, redistributed
and/or modified under the same terms as Perl itself.

=cut
```

Listing 6: Retrieval Results

```
#!/usr/bin/perl -Tw
#
# PROGRAM NAME - obtaining_results
#
```

```
# Author   : Eric Schabell
# Date     : 3003-02-14
# E-Mail   : erics@cs.kun.nl
#
# Changelog :
#
#   20030624 -  Release 1.0
#
use strict;
use vars '$VERSION';
use CGI qw(:all);
use DBI;

$VERSION = "1.0";

my $clearinghouse_db     = "clearinghouse";
my $db_driver      = "dbi:Pg:dbname";  #or "DBI:mysql:database" for MySQL.
#my $db_host             = "pgsql-clearinghouse.sci.kun.nl";
#my $db_user             = "clearinghouse-user";
#my $db_pw               = "ClearAtHouse";
my $db_host            = "localhost";
my $db_user            = "pronir";
my $db_pw                = "pron!r";
my $PRONIR_URL           = "http://www.pronir.nl/pub/spws/mirror/";

my $conversion_name = param('conversion');

my $dbh = DBI->connect("$db_driver=$clearinghouse_db;host=$db_host",
            $db_user, $db_pw, {'RaiseError' => 1});


my $query = "SELECT DISTINCT Routine.*, Filter.from_representation_type,
        Filter.to_representation_type, Package.pronir_url
        FROM Filter, Routine, Package
          WHERE
            Routine.name='$conversion_name' AND
            Routine.pkg_name = Package.name AND
            Routine.name = Filter.routine_name";

my $sth_query = $dbh->prepare($query);
$sth_query->execute();

print header,
  start_html(-title=>'Welcome to - The PRONIR Conversion Clearinghouse!',
         -author=>'erics@cs.kun.nl',
         -bgcolor=>'lightgrey'),
  "<CENTER>\n",
  h1('***** Obtaining a conversion from the PRONIR Conversion
    Clearinghouse *****'),
  "</CENTER>\n";

print_header();
print hr,
  h2('To obtain the selected conversion, see information below: '),
  "<FONT=+5>";

while (my $ref = $sth_query->fetchrow_hashref()) {

  print "<FONT=+5>",
    "<TABLE width=70% border=0>\n",
      "<TR>\n",
        "<TD width=300>",
          em('Conversion name -> '),
        "</TD>\n",
        "<TD width=300>",
          b("$ref->{'name'}"),
        "</TD>\n",
        "<TD>\n",
        "</TD>\n",
      "</TR>\n",
      "<TR>\n",
        "<TD width=300>",
          em('Package containing conversion -> '),
        "</TD>\n",
```

```
        "<TD width=300>",
          b("$ref->{'pkg_name'}"),
        "</TD>\n",
        "<TD>\n",
        "</TD>\n",
      "</TR>\n",
      "<TR>\n",
        "<TD width=300>",
          em('Converts from -> '),
        "</TD>\n",
        "<TD width=300>",
          b("$ref->{'from_representation_type'}"),
        "</TD>\n",
        "<TD>\n",
        "</TD>\n",
      "</TR>\n",
      "<TR>\n",
        "<TD width=300>",
          em('Converts to -> '),
        "</TD>\n",
        "<TD width=300>",
          b("$ref->{'to_representation_type'}"),
        "</TD>\n",
        "<TD>\n",
        "</TD>\n",
      "</TR>\n";


  for ($ref->{'pronir_url'}) {

    if (/\$PRONIR/) {print "<TR>\n",
                "<TD width=300>",
                  em('Clearinghouse package location -> '),
                "</TD>\n",
                "<TD width=300>",
                  ('<a href='),
                  ("${PRONIR_URL}$ref->{'pkg_name'}.tar.bz2"),
                  ('>Clearinghouse Mirror Download.</a>'),
                  ("\n"),
                "</TD>\n",
                "<TD>\n",
                "</TD>\n",
                "</TR>\n"}
    else       {print "<TR>\n",
                  "<TD width=300>",
                  em('Clearinghouse package location -> '),
                  "</TD>\n",
                  "<TD width=300>",
                    ('<a href='),
                    ("$ref->{'orig_url'}$ref->{'pkg_name'}.tar.bz2"),
                    ('>Clearinghouse Mirror Download.</a>'),
                    ("\n"),
                  "</TD>\n",
                  "<TD>\n",
                  "</TD>\n",
                  "</TR>\n"}
  }
}

$sth_query->finish();
print end_html;

$dbh->disconnect();
# end obtaining_results.

sub print_header {

  # header and footer.
    #
    print "<!--  Download links at the top of the page -->",
      "<table width=\"100%\" border=\"2\">",
        "<th width=\"100%\" colspan=4 height=\"2%\" rowspan=1
      align=\"center\" nowrap>",
        "<a href=\"clearinghouse\"
```

```
      target=\"_new\">Conversion_Clearinghouse  -</a>",
        "<a href=\"http://www.pronir.nl\"
      target=\"_new\"> Pronir_Home  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/download.html\"
      target=\"_new\"> Download  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/docs/\"
      target=\"_new\"> Documentation  -</a>",
        "<a href=\"http://www.pronir.nl/pub/spws/call.html\"
      target=\"_new\"> Call for Interest </a>",
        "</th>",
        "</table>",
        "</CENTER>\n";
}

# end obtaining_results.
```

References

[GPB03a] B. van Gils, H.A. Proper, and P. van Bommel. A conceptual model for information suppy. Technical Report NIII-R0313, Nijmegen Institute for Information and Computing Sciences, University of Nijmegen, Nijmegen, The Netherlands, EU, 2003. Accepted for publication in Data & Knowledge Engineering.

[GPB03b] B. van Gils, H.A. Proper, and P. van Bommel. Towards a general theory for information supply. In C. Stephanidis, editor, *Proceedings of the 10th International Conference on Human-Computer Interaction*, pages 720–724, Crete, Greece, EU, 2003. ISBN 0805849300

[GS03] B. van Gils and E.D. Schabell. User-profiles for information retrieval. In *Proceedings of the 15th Belgian-Dutch Conference on Artificial Intelligence (BNAIC'03)*, Nijmegen, The Netherlands, October 2003.

[MyS03] *MySQL Website*, 2003.
`http://www.mysql.com`

[Pos03] *Postgres SQL Website*, 2003.
`http://www.postgresql.org`

[Sch02] Eric D. Schabell. Resource access in generic information retrieval systems. Master's thesis, Vrije Universiteit, Amsterdam, Netherlands, 2002.