Gathering Data for the Educational Lending Rights Survey

Author: *Corey Wallis, Systems Librarian,*
University of Adelaide Library

Last Update: *3/01/2007 3:37 PM*

Abstract: *The University of Adelaide Library was approached by the DCITA to participate in the Lending Rights Survey for 2006. To participate it was necessary to extract details of monographs, books, held by the Library. To extract this information a Perl script was developed and tested in conjunction with the DCITA.*

Problem

The DCITA required a report of all "books" held by the University of Adelaide Library. For the purposes of the report a "book" is defined as any item that has an ISBN and is a physical item held in the Library. This definition excludes such things as electronic books, journals, and books predating the introduction of the ISBN system.

The data extracted from the catalogue needed to contain the following three columns:

1. Book Title
2. ISBN
3. Number of copies held

Each column was to be separated by a "^" symbol and the data needed to be presented as a plain text file.

Solution

The solution was to develop a Perl script that would extract the required information from the Voyager database and create a text file in the appropriate format required by the DCITA. The following tables, in the Voyager Oracle database, are used to gather the required data:

1. bib_text
2. bib_mfhd
3. mfhd_master

To determine if an item is a “book” the following criteria are applied:

1. The *isbn* field in the *bib_text* table must not be *NULL*
2. The *bib_format* field in the *bib_text* table must equal *am*
3. The character at the 23rd position of the *008* field must be either a *space*, or an "*r*"
4. The title of the work can not contain the string: *[microform]*

Additional checks are performed against the *isbn* field in the *bib_text* table to ensure that only items with legitimate ISBNs are included in the file.

As well as a list of all holdings a log file is also created. This log file contains the *bib_id* of each record that had an ISBN that did not pass the validation checks.

Using the Script

1. Copy the source code from Appendix A and paste it into a new text file entitled *elr_survey.pl*
2. Copy the Perl script to the Voyager server
3. Ensure the VGER.pm Perl module is available to the script. As outlined in the document *A trivial Perl module improving Oracle access from Perl*
4. Ensure the script is executable
5. If you want to output a test file
    1. open the **elr_survey.pl** using vi
    2. Locate the following line in the file

       my $debug = 0;

    3. Ensure the value of the **$debug** variable is set to 1

4. Exit vi and save the file

6. If you want to output a complete export

    1. open the **elr_survey.pl** using vi

    2. Locate the following line in the file

       my $debug = 0;

    3. Ensure the value of the **$debug** variable is set to 0

    4. Exit vi and save the file

7. If you want to change the location of the holdings file

    1. open the **elr_survey.pl** using vi

    2. Locate the following line in the file

       my $output_file = "/tmp/elr_survey.txt";

    3. Ensure the value of the **$output_file** variable is set to the new location

    4. Exit vi and save the file

8. If you want to change the location of the log file

    1. open the **elr_survey.pl** using vi

    2. Locate the following line in the file

       my $log_file = "/tmp/elr_log.txt";

    3. Ensure the value of the **$log_file** variable is set to the new location

    4. Exit vi and save the file

9. Run the script by entering the following command
   (This will take some time)

   ./elr_survey.pl

10. The output of the script will be saved in the location specified by the **$output_file** variable

11. The log file of invalid ISBNs will be saved in the location specified by the **$log_file** variable

Appendix A: Source Code of the elr_survey.pl script

```
#!/m1/shared/bin/perl -w
# This is a Perl script to retrieve the data required for the
# Educational Lending Rights survey conducted by the DCITA

use lib "../lib";

use DBI;
use VGER;
use strict;

# Settings to control how the script behaves.
# These should not need to be edited
my $debug = 0;
my $output_file = "/tmp/elr_survey.txt";
my $log_file = "/tmp/elr_log.txt";
my $sep = "^";

# You should not need to edit anything below this line
#############################################################

# Open the output file for writing
# Specify UTF8 as the output encoding
if (! open OUTPUT, ">:utf8", $output_file) {
    die "Cannot open \"$output_file\"\n$!\n";
}

# Open the log file for writing
# Specify UTF8 as the output encoding
if (! open LOG, ">:utf8", $log_file) {
    die "Cannot open \"$log_file\"\n$!";
}

# Define other variables
my $counter = 0;
my $row_count = 0;
my @bad_isbns = [];
```

```
my $bib_id = 0;
my %bad_ids = ();
my $sql = "";

# Connect to the Oracle Database
my $dbh = DBI->connect("dbi:Oracle:host=$db_host;sid=$db_sid",
                                                    $db_username,
                                                    $db_password
   ) || die "Could not connect to DB: $DBI::errstr";

# Prepare the SQL statement
if ($debug == 1) {
    $sql = qq|
    SELECT *
    FROM (
        SELECT b.title, b.isbn, COUNT(b.isbn) as item_count, b.field_008
        FROM bib_text b, bib_mfhd bm, mfhd_master mm
        WHERE (
            b.isbn is not null
            and b.bib_format = 'am'
        )
        AND b.bib_id = bm.bib_id
        AND bm.mfhd_id = mm.mfhd_id
        GROUP BY b.title, b.isbn, b.field_008
    )
    WHERE ROWNUM <= 200|;
} else {
    $sql = qq|
    SELECT b.title, b.isbn, COUNT(b.isbn) as item_count, b.field_008
    FROM bib_text b, bib_mfhd bm, mfhd_master mm
    WHERE (
        b.isbn is not null
        and b.bib_format = 'am'
    )
    AND b.bib_id = bm.bib_id
    AND bm.mfhd_id = mm.mfhd_id
    GROUP BY b.title, b.isbn, b.field_008 |;
}

# Prepare the statement handler
```

```
my $sth = $dbh->prepare($sql) || die $dbh->errstr;

# Execute the script
$sth->execute || die $dbh->errstr;

# Process each row and output as necessary
while (my(@row) = $sth->fetchrow_array()) {
    # Check to ensure this is an actual book
    # Excludes things such as e-books and other works

    # The character at the 23rd position of the 008 tag
    # Should be either a space or the letter r
    my $book_check = "^.{23} |^.{23}r";

    # Some microform seem to pass this check
    # Do a simple check for the string [microform] in the title
    my $microform_check = '\[microform\]';

    if ($row[0] =~ /$microform_check/) {
        # skip to the next iteration of the loop
        next;
    }

    if ($row[3] =~ /($book_check)/) {
        # We have a book to process
        # We need to tidy up the ISBN
        # ISBNs should be 13 digits, 10 digits, 9 digits ending in X
        # Some old books have an SBN, which is 9 digits
        my $isbn_check = q /^\d{13} | ^\d{10} | ^\d{9}X | ^\d{9} |
        ^\d{8}X/;

        # Create a copy of the ISBN for error checking purposes
        my $isbn_copy = $row[1];

        # Remove any dashes in the ISBN
        $row[1] =~ s/-//g;

        # Remove any spaces in the ISBN
        $row[1] =~ s/ //g;
```

```
        # Remove any round brackets
        $row[1] =~ s/\(//g;
        $row[1] =~ s/\)//g;

        # Uppercase the string
        $row[1] = uc($row[1]);

        if ($row[1] =~ /($isbn_check)/x) {
            # We have a tidied ISBN
            $row[1] = $&;

            # Check if it is an SBN and add a leading 0
            if (length($row[1]) == 9) {
                # We have an SBN
                $row[1] = "0" . $row[1];
            }

            # Tidy up the title
            my $title_check = "^.*/";
            if ($row[0] =~ /($title_check)/) {
                my $tmp = $&;
                $tmp =~ s|\s+/||;
                $row[0] = $tmp;
            }

            # Output this record to the file
            print OUTPUT qq|$row[0]$sep$row[1]$sep$row[2]\n|;

            # Increment the counter
            $counter++;

        } else {
            # The ISBN didn't get tidied
            # Store the copy so we can get it later
            push (@bad_isbns, $isbn_copy);
        }
    }
}

# Close the output file
```

```
close OUTPUT;

# Gather number of records returned by query
$row_count = $sth->rows;

# Tidy up the statement handle
$sth->finish;

# Loop through the list of bad ISBNs and output the bib_id
# This can be used for checking later

# Prepare the SQL statement
$sql = qq|
    SELECT bib_id
    FROM bib_text
    WHERE isbn = ?
    |;

while (@bad_isbns > 0) {

    my $item = pop(@bad_isbns);

    if (@bad_isbns > 0) {
        # Prepare a cached query
        $sth = $dbh->prepare_cached($sql) || die $dbh->errstr;

        # Execute the statement
        $sth->execute($item) || die $dbh->errstr;

        # More than one record may be returned
        while ($bib_id = $sth->fetchrow_array()) {
            # Assign each ID to the hash
            # By using a hash we don't get any duplicates
            $bad_ids{$bib_id} = "";
        }
    }
}

# Get the list of bib_ids with bad ISBNs
# Reuse the @bad_isbns variable
```

```
@bad_isbns = keys(%bad_ids);
@bad_isbns = sort(@bad_isbns);

# Write to the output file
foreach my $item (@bad_isbns) {
    print LOG $item . "\n";
}

# Tidy up DB connection
$sth->finish;
$dbh->disconnect;

# Close the log file
close LOG;

# Print message to user
print "Process Completed.\n";
print "Number of records processed: $row_count\n";
print "Number of records output to file: $counter\n";
print "Number of records with errors: " . keys(%bad_ids) . "\n";
print "Output file: $output_file\n";
print "Log file: $log_file\n";
```

References

Lending Rights (DCITA)
http://www.dcita.gov.au/arts_culture/arts/lending_rights

A trivial Perl module improving Oracle access from Perl
http://hdl.handle.net/2440/14785