External Common Gateway Interface (CGI) Access to FindU

Thomas M. Schaefer, NY4I
Utah APRS User's Group
11678 Littler Rd
Sandy, UT 84092
nv4i&.irl..net

Abstract

This paper provides programmer documentation on external access to the FindU database. The Common Gateway Interface (CGI) is used as an example of methods for a script or program written on a web server to access that database.

Introduction

When Steve Dimse, K4HG, announced the FindU system at the 1999 APRS National Symposium [l] in Phoenix, little did the APRS community realize the impact this collection of APRS information would have on APRS. Fortunately, Steve has created a site that allows web access to perform certain queries about positions and other APRS information. Of course, being a busy person., Steve certainly cannot program every function the APRS world needs or wants into the system. Fortunately, Steve has allowed external access to the FindU database so inventive programmers can create their own access programs into the FindU system. This relies upon the fact that the FindU system itself is based on the very open, Linux operating system and the MySQL database management system. Using standard routines, it is possible for any programmer with access to the Internet to issue queries to the FindU system and return specific data to satisfy their individual need.

Database Structure

As previously stated, FindU relies upon a MySQL database system running on a Linux system. In order to successfully direct connect to the FindU database, the database structure must be known. Keeping in mind, that FindU is still an evolving system -- meaning the database is subject to change - this is not intended to be an all-inclusive list. It is a current working example at the time of this document's creation. At some point, it is expected that FindU itself will post the actual database format.

The main table for position information is the POSITION table. This table contains several fields that are relevant:

Call	Callsign of the posit
Lon	Long. Of the station
Lat	Lat of the station
Time rx	Time the spot was received
Speed	speed of the posit
Course	Course from the posit
Raw	Index to the RAW packet table (used to cross-reference the actual packet)

An example SQL query using this data is:

```
SELECT  p.call,p.lat,p.lon,p.time_rx,p.speed,p.course,raw.text
from position p, raw
WHERE p.call like 'ny4i%'
AND p.raw = raw.cnt
```

As written, this SQL statement returns all the records in the POSITION table for any callsign starting with “ny4i”. Additionally, it returns the actual raw packet from the raw table via an SQL join.

Database Usage

Now that a basic table structure is known, it requires SQL to put it to use. Depending upon your programming language, there are different ways to access that data. The website for MySQL (www.mvsql.corn) contains a section on programming interfaces to MySQL,. Interfaces exist for C++, ODBC (Windows), Java, and DBI (Perl). Since most CGI programs are written in Perl, the focus of this paper is the DBI method. The DBI library is a Per1 module that is loaded at ,the start of a script. As an example, the following code is the Per1 method to connect to the FindU database via DBI:

```
$driver = 'lmysql";
$database = riaprsH;
$host = "64.34.101.121";
$user = "guest'l;
$passwd = "";
$dsn = ~~~:$driver  :database=$database:host=$host";
$dbh = DBI->connect($dsn,$user,$passwd)  ;
$drh = DBI->install_driver("mysql");
```

With this code, a connection is now available to access the FindU database directly. Building on the previous example, the method to retrieve some position information using Per1 follows:

```
$statement = "SELECT p.call, p.lat, p.lon,  p.time_rx, p.speed,
p.course,  raw.text from position p/ raw WHERE p.call LIKE
\l$call%\l AND p.raw = raw.cnt AND p.time_rx :>= $startDate AND
p.time_rx c= $stopDate";

$sth = $dbh->prepare (Sstatement);
$sth->execute;

$numRows  = $sth->rows;
```

With this query, the database connection now contains any records available for the requested callsign. Retrieving the individual data fields requires code to loop through the returned database records and utilize the data. An example of this follows:

```
while (my $ref = $sth->fetchrow_arrayref)
   {
      $db call = $$ref[O];
      $db-lat = $$ref Cl];
      $db-lon = $$ref[2];
      $db-time rx = $$ref[3];
      $db-speed = $$ref[4];
      $db-course = $$ref[5];
      $db_raw = $$ref [6];

      print vkTR>\n^lf;
      print I^1  cTD VALIGN=\^11TOP\^15$urlc/TD>\n^11;
      print II  cTD VALIGN=\^11TOP\5$db latc/TD>\rF;
      print *I  cTD VALIGN=\~^1TOP\^f5$db~lon</TDs\n~^t;
      print II  cTD VALIGN=\I'TOP\%$topoURL</TD>\n^fl;
      print I^1  cTD VALIGN=\1^1TOP\5$db speedc/TD>\rF;
      print II  cTD VALIGN=\^1fTOP\5$db-course</TDz\n^11;
      print 'k/TR>\n'Q
   }
```

This code loops through each returned position and populates the "db_" variables. The variables are then used in an HTML table. It is important to keep in mind that while this application displays the data to a web browser, that is not the only use of it. It would be possible to simply search the records to determine maximum speed, or other things for strictly internal reporting.

SQL Cautions

Writing direct SQL queries, while generally easy to learn, does carry a responsibility. It is not terribly difficult to create a Linear Search where every record in the database table must be examined to find a match for your query. To maximize efficiency, indexes are normally used w‘hich provide a means to quickly search on a specific field. Similar excessive searches can occur if qu.eries are not dated. In the FindU case, the time rx field of the positions table should be used to limit the search to a specific data range, when known. By using common sense analysis of the data requested, everyone can successfully use the FindU database.

Windows Access to FindU

While Per1 and DBI have been illustrated here, there are other methods to ac.cess the FindU database. The author has created Microsoft Excel spreadsheets to access the FindU database via the Microsoft standard, Open Database Connectivity (ODBC). Using ODBC, programs like Word, Access, and Excel have total access to the FindU system. As with the Perl/DBI method, efficient usage is required to ensure responsive queries.

Conclusion

Using commonly available tools, external database access to FindU is available to integrate all APRS data into programs. Using Per1 and CGI, it is possible to return all positions of a station. Want to find out

where an APRS station has been in the last week, it is now possible. As compared to the original APRServe system, now many different programmers can create queries that suit their particular applications. With this power, does come a responsibility to be a responsible FindU “citizen”. But with the proper knowledge of SQL, well-crafted SQL queries can be created to return useful data for the author or the APRS community at large. Of course, without the pioneering efforts of Steve Dimse, K4HG, none of this would have been possible

References

PI Dimse, Steve, K4HG, “Internet and APRS,” [Online Document], 1999 May, Available HTTP: htttx//www.tanr.org/tapr/ra/dcc99.aprs.4.k4hg.ram,