The H.A.R.C. Database and Visualization Utilities

Joshua D. Katz
New Jersey Institute of Technology
William Engelke
University of Alabama
Dr. Nathaniel Frissell
New Jersey Institute of Technology

Jul 31, 2017

Abstract

HamSCI's goal is to construct a symbiotic relationship between the formal research community and the Amateur Radio community. To facilitate this transfer of knowledge HamSCI must pioneer technologies that allow scientists to easily obtain and understand Amateur Radio data. This task necessitates the creation of warehousing and visualization facilities that allow scientists to easily understand and make use of our data sets. We are currently testing a database and visualization toolkit designed to handle our existing 2 billion-record long QSO log. This data set represents a compiled version of data gathered by the Reverse Beacon Network, WSPRNet, and PSKReporter. Our goal is to build a robust, fast, and queryable front end to the massive, and currently underutilized, data sources created by Amateur Radio operators.

Keywords— Database, HamSCI, Contest Logs

1 Introduction

When investigating a phenomena it is often difficult to find easily accessible and understandable metrics and data sources that can be used for an analysis. There is also a large amount of testing and validation that must be done to any data source before it is acceptable to use. Because of this it is often difficult to convince researchers to consider using data sources that are not already characterized or understood. This presents an initial challenge to the adoption and use of data generated by Amateur Radio operators. We have taken steps to perform initial validation and support to reverse this trend and make Amateur Radio data more attractive to the research community. To do this we have developed a database to abstract the different storage schemas of various data sources, investigated visualization utilities, and worked towards a better understanding of the data source as a whole. This reduction of complexity and initial phase of verification makes Amateur Radio data much more attractive to the research community.

2 Implementation

The HARC, or the HamSCI Amateur Radio Communications, Database is an SQL schema for storing extremely large QSO logs. We are currently testing our schema within MariaDB which is a popular open source fork of MySQL that aims to improve documentation and performance while also adding important features. MariaDB was also chosen for its existing robust clustering facilities

that will allow us to deploy mirrored copies of the HARC database across multiple universities and institutions who wish to use the data source. Deploying in this fashion provides high availability, a deployment characteristic of extremely fault-tolerant systems, and quicker access times.

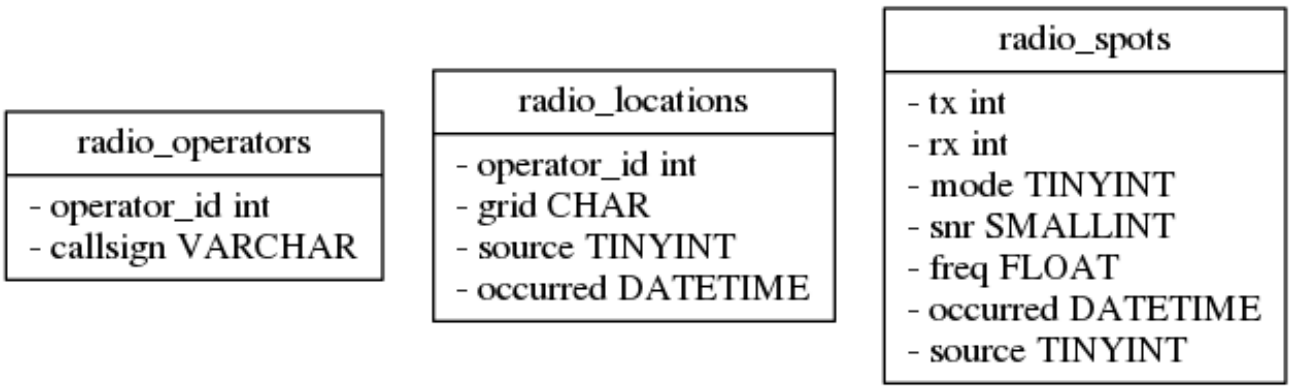

Figure 1: Demonstration of the HARC database storage and query scheme.

The storage schema for the tables used within the database are described in Figure 1. This schema was built to be as efficient and simple as possible. This system also allows us to still identify where each record comes from due to the "source" fields in both the "radio_locations" and "radio_spots" tables. Along with this analysis HamSCI has begun investigating data visualization techniques that can show scientifically interesting information in Amateur Radio systems. These systems consist of DX Display and other internal visualization tools.

2.1 Storage Size

In HARC there is no single table that contains all information about a contact. It would be much simpler to implement, understand, and test a database where every bit of information was stored in each record but unfortunately such a database would perform poorly and be difficult to maintain. The performance impact of storing repetitive data is not initially evident. Most people only look for the first order effects of large tables. As an exercise we will engage in a thought experiment discussing the implications of storing repetitive values in each record we store. For our example we will focus on storing the transmitter and receiver's callsigns with every QSO we write to disk. When an SQL server is instructed to store a record it will write what ever you tell it on to the disk. If you attempt to store 1 million rows with the same values the database will, by and large, not make any attempt to compress that data; it will write all 1 million copies of that data. Because of this storing callsigns in the communication logs causes your database size to balloon.

```
SELECT AVG(LENGTH(callsign)) FROM radio_operators;
+-----------------------------+
| AVG(LENGTH(callsign)) |
+-----------------------------+
|                      5.1047 |
+-----------------------------+
```

Finding average length of callsign.

From this query we can find that the average length of the callsings observed by the Reverse Beacon Network is approximately 5 characters. Each character is approximately one byte stored to the disk and there is a one byte length attached to each string.

```
SET @RBN_SPOTS_SOURCE = ...; /* Source ID for RBN */
SELECT COUNT(*) FROM radio_spots
WHERE source = @RBN_SPOTS_SOURCE;
```

```
+-------------+
|  COUNT(*)   |
+-------------+
| 577801090   |
+-------------+
```

Number of spots in the RBN from 2009 to 2016.

From this query we can see that within the RBN we have observed 577,801,090 spots. If we were to store both the transmitter and receiver callsigns for all of those spots we would need to store approximately 7 GBs of callsigns. This excludes any metadata or indexing that the SQL storage engine would append to this. We can improve this situation by storing only a unique number for each callsign and keeping a mapping of said unique number to callsign.

```
SELECT MAX(operator_id) FROM radio_operators;
+--------------------+
| MAX(operator_id)   |
+--------------------+
|            1146669 |
+--------------------+
```

Number of distinct transmitter callsigns observed by the RBN from 2009 to 2016.

From this we can see that there are only 1,146,669 distinct callsigns heard by the RBN. If we stored an UNSIGNED INT instead of the 5 character string from before we would only store approximately 3 GBs of data. This represents more than a 50% savings in storage space.

Some may say that this optimization is not needed because storage space is cheap. This is incorrect. Storage space is cheap but the width of your rows on disk does not only effect storage size. If you store a large row onto the disk you will eventually need to read or seek past that row at some point during a query execution. If your rows are overly bloated with unneeded information this will cause a performance impact that will either manifest itself as wasted CPU cycles, decreased disk transfer rates, or a low hit rate for the disk cache.

2.2 Indexing

Relational databases can only improve speeds for specific use cases by use of indexing. An index is a data structure that contains meta data about where on the disk specific data is stored. This is functionally equivalent to the alphabetization of a dictionary. Although dissimilar, the outcomes are the same. When you want to look up a word you do not need to read every word from the dictionary, in order, to find your word. You can flip to the section of the dictionary that you know should contain your word because of knowledge you already have about the organization, and location, of the data and your dictionary. Because of this looking up the definition of a word, or a set of words, is much faster. However if you need to look up the definition of every word you will not see any performance benefits from the ordering of the dictionary. This is the same with the database. Only queries that do not aggregate information about the entire database, and instead only look for a subset of records in the database, can be optimized with an index. Other types of queries will not see any speed improvement and will actually be slower if an index is applied.

2.3 Table “radio_operators”

The “radio_operators” table contains a mapping of every operator’s callsign to our internal unique operator ID. This internal ID is much smaller than a callsign when stored in the SQL databases. This will also allow us to make better use of different types of indexing, such as foreign keys, to provide much faster lookups for specific queries. This has not been implemented yet.

2.4 Table “radio_locations”

The “radio_locations” table contains the location of every operator ID we find. We also store a date when each location was obtained in the “occurred” column. This allows us to estimate the location of an operator at any given time.

```
SET @interesting_operator=... /* Fill in with Operator ID */
SET @interesting_datetime=... /* Fill in with a datetime  */

SELECT * FROM radio_locations
WHERE operator_id = @interesting_operator
```

SQL query obtaining an operator’s likely location at a specific date.

2.5 Table “radio_spots”

The “radio_spots” table contains status information for every communication observed by any network we have incorporated into our database. Any information that is repetitive, or will appear in multiple contacts, has been factored out into other tables. We have only included information which will change from contact to contact. Some examples of this information would be the signal to noise report (“snr”), the frequency (“freq”), when the contact was heard (“occurred”), what network this report is from (“source”) and the communication mode (“mode”). While not enough information is within this column to do all types of analysis we only store these columns because they are numeric. This will allow SQL engines to efficiently store and query this data. For the rest of the information you must query other tables. These extra tables are much smaller then the main dataset. These other tables can be joined to our main dataset within SQL to provide a more complete column set.

Figure 2: An automated propagation report system fed from multiple spotting networks. Built and run by William Engelke.

3 Data Visualization Tools: DX Display

DX Display is a real time service that automatically generates graphs that quantify Amateur Radio activity. This tool offers quick, and comprehensive, access to information from multiple Amateur Radio networks. By looking at the metrics displayed by DX Display the research community can easily pick out interesting phenomena, better understand how the data correlates to what is currently happening in the world, and have access to quick plotting utilities. DX Display is the first system to be affiliated with HamSCI which makes use of DX Cluster and WSPRNet spots in real time.

4 Data Visualization Tools: Python Data Analysis Environment

The Python data analysis environment is a thriving ecosystem of utilities and libraries. Many optimizations have been made to improve speed of complex numerical computations. The most common numerical analysis and scientific computing libraries available to scientists today in the Python software stack are SciPy, NumPy, Matplotlib, Seaborn, and Pandas. These libraries have built in facilities to handle column-based data sources. Pandas, a data loading and column processing framework, also supports directly grabbing data from SQL databases.

```
from sqlalchemy import create_engine
from pandas import read_sql

db = create_engine('mysql+mysqldb://USERNAME@HOSTNAME/DATABASE')

# Load last 15 minutes of spots.
spots = read_sql("""SELECT
    (SELECT callsign
      FROM radio_operators
      WHERE operator_id = rs.tx) as tx_call,
    tx_rl.grid,
    (SELECT callsign
      FROM radio_operators
      WHERE operator_id = rs.rx) as rx_call,
    rx_rl.grid,
    mode, snr, freq, occurred
  FROM radio_spots as rs
    INNER JOIN radio_locations as tx_rl
    ON tx_rl.operator_id = tx
    AND tx_rl.occurred = (SELECT occurred
      FROM radio_locations
      WHERE
        operator_id = tx_rl.operator_id AND
        occurred >= rs.occurred
        ORDER BY occurred DESC
        LIMIT 1)
    INNER JOIN radio_locations as rx_rl
    ON rx_rl.operator_id = rx
    AND rx_rl.occurred = (SELECT occurred
      FROM radio_locations
      WHERE
        operator_id = rx_rl.operator_id AND
        occurred >= rs.occurred
        ORDER BY occurred DESC
        LIMIT 1)
  WHERE
    occured >= (UTC_TIMESTAMP() - INTERVAL 15 MINUTE);""", con=db)
```

Loading data from HARC into Pandas Dataframe.

This example has been expanded and implemented in the most verbose and "correct" SQL query that represents this request. Many optimizations can be made to improve access times of the results of this query as you likely can avoid a JOIN in this situation. It is also possible to reduce the requested data from the database. You may not always need the TX and RX callsigns. You may not always need locations. When interfacing with this system it is recommended to only download data that you need and to provide as many filtering criteria for the data before obtaining it. For example if you need locations to be present for you analysis it is preferred to request the "INNER JOIN" of your two tables as this will automatically remove items that do not have any location data associated with them. If you would only like 10,000 spots "LIMIT" your query.

5 Benefit

Amateur Radio QSO logs are difficult to understand for the uninformed scientist. HARC brings a friendly abstraction to the table, specifically one which scientists already understand. We present QSO logs as instead being a list of radio transmissions that were heard in various places. We provide ways to graph these transmissions. We also have begun to define what is normal, what is interesting, what is obvious, and what is erroneous within the data sources. HARC removes the complexities and peculiarities of each individual data source by presenting a clean and uniform method for accessing the accumulated data of the Amateur Radio community.

6 Summary

To encourage researchers to use Amateur Radio data sources we have begun the long process of implementing, validating, and testing multiple systems that will enable scientists to more easily understand these data sources. We have done this by creating an abstract data acquisition layer, beginning analysis of the data, and building tools to help better understand the changes caused by ionospheric effects, space weather, and anthropogenic effects on Amateur Radio operations. We hope to deploy our data visualization tools and the HARC database to universities involved with the HamSCI investigation.

Biography

Joshua D. Katz, KD2JAO

Joshua D. Katz, KD2JAO, is an Undergraduate Associate Researcher at the New Jersey Institute of Technology Center for Solar-Terrestrial Research (NJIT-CSTR). He is currently working towards a major in Computer Science and the acting treasurer of K2MFF, the NJIT Amateur Radio Club.

Author E-Mail List

Name	Callsign	E-Mail
Joshua D. Katz	KD2JAO	jk369@njit.edu
Nathaniel Frissell	W2NAF	nathaniel.a.frissell@njit.edu
William Engelke	AB4EJ	bill.engelke@ua.edu