load_netblocks_data_to_mysql.py - a script to create and maintain a MySQL netblocks database

WhoisXML API, Inc.

January 22, 2020

Contents

1 About the script

This script is intended for subscribers of Whois XML's IP netblocks database download product, to help them with creating and maintaining a netblocks

database in MySQL.

1.1 Script availability

The actual version of the script is hosted on github under
`https://github.com/whois-api-llc/whois_database_download_support/tree/master/netblocks_csv_to_mysqldb`

1.2 Changelog

- 0.0.1: Initial version
- 0.0.2: Updated for modified data format including "remarks" and "as_type" fields

1.3 A usable demonstration script

The script is intended to be a programming example of how to accomplish this task and also to be useful as it is. For this reason the script is not perfectly robust, there are no checks of the validity of the input files and the exception handling is also not very detailed, to avoid a complex and less readable code. If used properly, however, it is can be used efficiently.

1.4 Specifications

Before using the script it is important to understand what it exactly does. Here we describe the cornerstones of its operation.

-The script is controlled by command-line parameters, a full list can be obtained with the –help option. A more detailed description is to be found later in this document.

-It assumes that the database is prepared according to the following documentation, the MySQL database and a user with the appropriate permissions exists.

-It creates a set of tables with hard_coded names if they do not exist. These names can be prefixed with a custom string (–table-prefix).

-If a gzipped csv file containing contacts information is provided with the –contacts-file option, it is processed first. If the file is not specified, -this step is omitted. This affects the table "contacts".

-If a gzipped csv file containing netblocks information is provided with the –contacts-file option, it is processed first. If the file is not specified, -this

step is omitted. This affects the table "netblocks" and a set of associative tables representing contact information.

-The processing of the input files is carried out in chunks of max. 500.000 records (this can be overridden with –chunksize, and the number of chunks to be processed can be limited by –nchunksmax for debug purposes.). The database transaction is committed at the end of each chunk.

-The most important specialty of loading WhoisXML netblocks csv data into a MySQL database is that since these data originate from a NoSQL database, there is no information on the maximum length of string records in it. The script solves this problem by extending the VARCHAR fields on the fly if a record would not fit into the table.

-The script creates the tables if they do not exist, and perform an update of the respective records regardless of the file type. (Except for records which have action="delete" are deleted along with the respective records of the associative tables.

-The script supports a field with R-tree spatial indices, an efficient approach to replace the less efficient "BETWEEN" clause of SQL. This idea comes from the following blog: `http://blog.jcole.us/2007/11/24/on-efficiently-geo-referencing-ips-with-maxmind-geoip-and-mysql-gis/` The creation of the respective column can be disabled with the –no-r-tree-index option. We describe the details in a dedicated Section.

-The progress is logged to STDERR, this can be suppressed with the –quiet option.

-Problematic records are reported to STDOUT, this can be suppressed with the –no-mysql-error-reports option

2 Downloading data, data specs

The data are available from

`https://ip-netblocks-whois-database.whoisxmlapi.com/datafeeds/`

(with basic password authentication) or on ftp, as described in the specification of the product:

`https://ip-netblocks-whois-database.whoisxmlapi.com/specifications`

This script uses gzipped csv data, e.g.

- full files like
  - ip_netblocks.2019-01-03.full.contacts.csv.gz (contacts)
  - ip_netblocks.2019-01-03.full.blocks.csv.gz (blocks)

- daily updates like
  - ip_netblocks.2019-01-04.daily.contacts.csv.gz (contacts)
  - ip_netblocks.2019-01-04.daily.blocks.csv.gz (blocks)

Please familiarize yourself with the basic specifications of data before using the script.

3 Prerequisites

Here we describe the installation of the software required to run the script.

3.1 MySQL settings

The downloaded data files use Unicode encoding. Mysql uses a three-byte custom Unicode by default. In order to use this, include the following settings to your mysql configuration:

```
[mysql]
default-character-set=utf8mb4

[mysqld]
character-set-server=utf8mb4
collation-server=utf8mb4_unicode_ci
```

The first section typically goes to the mysql client config, whereas the second one goes to the config of mysql daemon. The organization of these can depend on your system, please consult its documentation. After the configuration change you need to restart your MySQL server. Note: according to our experience, on a Windows 10 platform running MySQL server 8.0.14, (our test environment 2), this configuration is not needed, the script creates an appropriate database with the default settings.

Ensure that there is sufficient disk space for the database: when writing this script, the size of the MySQL database directory is 5.6 gigabytes.

Having set-up your server properly, create the database for the data, and a user with the appropriate permission of the database.

3.2 Python libraries

The loader script is written in Series 3 Python; it was tested with Python 3.6.7. on Linux and Python 3.7.2 on Windows. It uses the following libraries:

Pandas (`https://pandas.pydata.org/`) , a data analysis library, in order to efficiently load chunks of csv files.

MySQL connector the library to access MySQL databases, provided by Oracle.

sqlescapy to help importing the multiline field "remarks" of the `netblocks` table. If not installed, a warning is raised at the beginning of the import, and all "remarks" fields will be set to `NULL`.

While Pandas can be simply installed with the package manager ("pip install pandas"), the vanilla MySQL connector is available from its download web-page (dev.mysql.com/downloads/connector/python/) . On some systems you may install both with the package manager of your OS (e.g. with "apt" on Debian-flavor Linuxes, including Ubuntu and Mint).

3.3 Environments in which the script has been tested in

So far we have tested the script in the following environment:

1. Ubuntu 18.04.1 LTS, mysqld Ver 5.7.24-0ubuntu0.18.04.1 for Linux on x86_64 (Ubuntu), Python 3.6.7, pandas 0.23.4, mysql.connector 2.1.6; on a with Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 4 gigabytes of RAM, running in a Virtualbox environment hosted on the same version of Linux, on a Dell Precision 3620 Mini Tower workstation.

2. Windows 10, 64 bit, Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 4 gigabytes of RAM, running in a Virtualbox environment hosted on Ubuntu 18.04.1 LTS. Python ver. 3.7.2 (64 bit), pandas v. 0.24.0, mysql.connector 8.0.14, MySQL server 8.0.14.

4 Examples of use

The script is a monolithic python3 code which can be run from its directory or anywhere else. (Since the file names are not globbed, if the files are not next to the script, use a full path to specify them.) Below there are the two most typical use cases:

4.1 Populate an empty database with full data of a day

This will create a netblocks database with the status on 2019-01-03. If the tables exist, they will be updated.

```
./load_netblocks_data_to_mysql.py \
--mysql-user whoisuser --mysql-password whoispassword --mysql-database whoisdatabase
--contacts-file ip_netblocks.2019-01-03.full.contacts.csv.gz \
--netblocks-file ip_netblocks.2019-01-03.full.blocks.csv.gz \
--full-netblocks-file
```

- IMPORTANT: omitting –full-netblocks-file will lead to an erroneous

operation.

- Run with the –help option to get a full list of options.
- Adding –r-tree-index will create the column ip_poly with r-spatial index to facilitate fast queries.
- Adding –no-inetnum-index will not create the index for the (inetnumFirst, inetnumLast) column pair. This saves some time but makes queries using the "BETWEEN" clause very slow, more than 20 seconds on our test environment 1. Do this only if you only use r-spatial index.
- Adding –quiet will suppress progress messages to STDERR
- Adding –no-mysql-error-reports option will suppress reports on any problematic records
- If any of the two files (contacts, blocks) is not given, it will be skipped. Loading netblocks without contacts, however, will cause constraint violations.
- In our test environment 1, the process took about an hour, in the test environment 2 it took more than 2 hours.

4.2 Update the database with data of a day

```
./load_netblocks_data_to_mysql.py \
--mysql-user whoisuser --mysql-password whoispassword --mysql-database whoisdatabase \
--contacts-file ip_netblocks.2019-01-04.daily.contacts.csv.gz \
--netblocks-file ip_netblocks.2019-01-04.daily.blocks.csv.gz
```

- As for options, see the previous example
- If you had an r-tree index column before, –r-tree-index is recommended here, too.

- It is much faster than initializing the db from scratch; on our test environment 1, it took about 2 minutes. On test environment 2 it was also slower.

5 Making queries in the database

5.1 Data structure

The relational database structure of the database is presented in the following

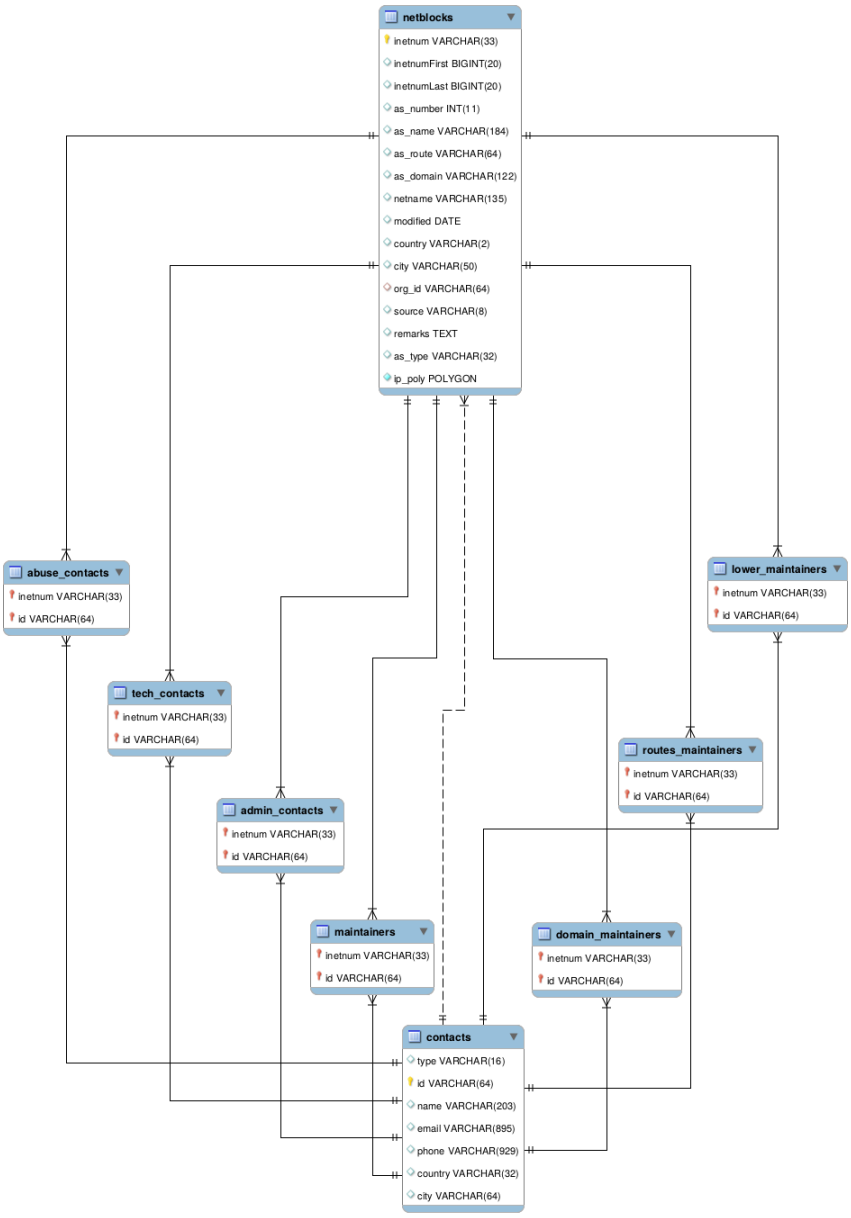

diagram:
Notes:

- The netblocks.ip_poly column is not there if –no-r-tree-index was set.
- The size of the VARCHAR fields can vary upon loading or updating.
- The main tables are the contacts and columns, in 1:n relation representing the organization the block belongs to.
- The 7 associational tables realize the n:m relations between the two main tables, representing other related organization and contact data to the block wherever available.
- As for the meaning of the fields, please consult the specification of the data at `https://ip-netblocks-whois-database.whoisxmlapi.com/specifications`
- This data structure contains all the data provided in the csvs, plus ip_poly, a generated column to facilitate an efficient search. If you find it redundant, modify the loader script to load less data.

5.2 Example queries

The maybe most typical query is to find the netblocks an IP address belongs to. Take 206.225.82.106, that is, our primary web server, whoisxmlapi.com as an example. The regular way of finding these netblocks would be

```
SELECT inetnum, netname, netblocks.country, netblocks.city,
       contacts.type, contacts.name, contacts.country, contacts.city
FROM netblocks LEFT JOIN contacts ON org_id=id
WHERE INET_ATON('206.225.82.106') BETWEEN inetnumFirst AND inetnumLast;
```

resulting in

```
+--------------------------------+-------------------------------------+---------+-----
| inetnum                        | netname                             | country | city
+--------------------------------+-------------------------------------+---------+-----
| 206.225.80.0 - 206.225.87.255  | CODERO2004A                         | US      | Over
| 206.195.64.0 - 206.252.223.255 | NON-RIPE-NCC-MANAGED-ADDRESS-BLOCK | EU      | NULL
| 206.0.0.0 - 206.255.255.255    | NET206                              | US      | Cent
| 0.0.0.0 - 255.255.255.255      | IANA-BLK                            | EU      | NULL
+--------------------------------+-------------------------------------+---------+-----
```

An equivalent one, based on the idea described in detail on `http://blog.jcole.us/2007/11/24/on-efficiently-geo-referencing-ips-with-maxmind-geoip-and-mysql-g`
is

```
SELECT inetnum, netname, netblocks.country, netblocks.city,
       contacts.type, contacts.name, contacts.country, contacts.city
FROM netblocks LEFT JOIN contacts ON org_id=id
WHERE MBRCONTAINS(ip_poly, ST_POINTFROMWKB(POINT(INET_ATON('206.225.82.106'), 0)));
```

resulting in the same results. On Windows systems we have found that this query is very slow for some reason; the query with "BETWEEN" is the recommended approach on that platform. The reason for this is under investigation.

Of course other contact information can be gained by using using the associative tables and querying them directly.

6 Submitting error reports or recommendations

If you find any problem with the operation of the script, or you have recommendations regarding the script, please contact us.

You can raise an issue on the github page of our support scripts, `https://github.com/whois-api-llc/whois_database_download_support` or send an e-mail to "support@whoisxmlapi.com".

In addition to your comment, please provide the following information:

- The version number of the script you are using. This can be obtained by invoking

  ```
  ./load_netblocks_data_to_mysql.py --version
  ```

  we need the the information in the first output line, e.g. load_netblocks_data_to_mysql.py ver. 0.0.1

- Information on the OS you are using (e.g. Ubuntu 18.04.1 LTS, or Windows 10),

- the version number of your Python (e.g. Python 3.6.7),

- the version of the mysql connector (e.g. '2.1.6', the value of "mysql.connector.__version__" after importing mysql connector) and pandas python libraries (e.g. '0.23.4', the value of "pandas.__version__" after importing pandas),

- and version information of your MySQL server (output of "mysqld –version", e.g. "mysqld Ver 5.7.24-0ubuntu0.18.04.1 for Linux on x86_64 (Ubuntu)" as well as settings you consider as relevant).