

Importing a SNOMED CT Full Release into MySQL

Introduction

This document explains how to use an rf2_load SQL script to load a SNOMED CT International Release package into a MySQL database. The resulting database not only includes all the data from the release files but also provides a range of integrated and optimized snapshot views. These views include the ability to simultaneously access a current snapshot view and a user-specified previous snapshot view.

In addition to snapshot views of individual release table data, the resulting database provides direct access to several useful combined views. These include

- Preferred and/or acceptable terms in a specified language or dialect (including synonyms and/or fully specified names) for any selected set of concepts.
- The subtypes and supertypes of selected concepts (with the language/dialect specific synonym or full specified name for each concept);
- Defining attribute relationships (with the language/dialect specific synonym or full specified name for each attribute and value concept);
- The transitive closure of subtype relationships (current snapshot view only).
 - Allowing optimized retrieval of subtypes of selected concepts.

Prerequisites

MySQL Installation

The following sections of these notes assume that you have already installed MySQL and the MySQL Workbench on your system.

- MySQL and MySQL Workbench are available for as open source software for Windows, Mac and Unix like operating systems.

For download and installation details see [MySQL Community Edition](https://www.MySQL.com/products/community/)
(<https://www.MySQL.com/products/community/>)

SNOMED CT Release Package Download

The following sections of the notes assume that you have already downloaded the SNOMED CT Release Package. In order to do this you will first need to register for a SNOMED CT License.

- For details of how to register for a SNOMED CT license and download release packages please see the [SNOMED International Member Licencing and Distribution Service](https://mlds.ihtsdo.org/) (MLDS) (<https://mlds.ihtsdo.org/>).

Scripts to Process the Release Package

The following two scripts are required for the import process. They may be obtained in a zip download file or from a GitHub repository.

Script	SNOMED CT MySQL Loader Script
Name	<i>rf2_load_[package]_[YYYYMMDD]_relpath.sql</i>
Status	Essential
Notes	<p>The loader scripts provided by SNOMED International is an SQL script file. It is usually specific for a particular version of an International Edition release package. It is sometimes possible to adapt a loader script to work with a different version of the same package. However, packages or versions that contain additional reference set release files may not load completely. Similarly release packages or versions that do not include all the files in version for which the script is designed will cause the script to fail when it attempts to import the missing file.</p> <p>Before using this script copy it to file with a filename in which '_relpath' removed or replaced.</p> <p>Then open the file in a text editor and make the following replacements:</p> <ol style="list-style-type: none">1. In all cases you must replace every instance of \$RELPATH with the full path to the release files folder on your system (including the name of the folder). For example: /user/me/downloads/SnomedCT_InternationalRF2_PRODUCTION_20190131T120000Z2. If you are adapting the script to work with a release package with a different release date you must also replace all instances of the release date in the script with the actual release date (e.g. 20190131 → 20190731)3. If you are adapting the script to work with a release package with a different name you should also replace all instances of the release package name with the the name of the release package you are importing. The release package name is the name of the folder containing the release files (excluding the path name). For example: SnomedCT_InternationalRF2_PRODUCTION_20190131T120000Z<ul style="list-style-type: none">• Note that if the release package does not contain the same set of files as the release package for which the script was prepared, the import may fail.

Script	SNOMED CT Transitive Closure Script
Name	<i>transitiveClosureRf2SnapMulti.pl</i>
Status	Essential (see note)
Notes	<p>The loader scripts provided by SNOMED International expect to find a transitive closure file to import. This is not part of the release package itself but is generated from the release package by a Perl script. The Perl script file is provided with the loader script. Mac and Unix like systems usually have Perl installed.</p> <p>Windows users may install Strawberry Perl (freely available) so they are able to run the script.</p> <p><i>If you do not use the transitive closure script you will need to edit the SNOMED CT MySQL Loader Script file to remove the section that imports to and uses the transitive closure table. See notes in the next section.</i></p>

Importing the Release Files

Unzip the Release File Package

The release file package is usually delivered as a zip archive. You need to unzip (or expand) this package to allow the scripts to work. It is a good idea to move the zip folder to a convenient location with a relatively short path name.

When the archive is expanded there should be a single folder with a name like

- SnomedCT_InternationalRF2_PRODUCTION_20190131T120000Z

Info:

In this case the parts of the name represent the package, status and version:

- SnomedCT : This is a SNOMED CT release file.
- International : It is the International Edition package
- PRODUCTION : It is a production release of this package
- 20190131 : It is the July 2018 release of the this package
- T120000Z : The national release time 12:00 UTC (note that updates may have a later time)

Run the Transitive Closure Script

In Mac or Unix environments use terminal

Change directory to the folder containing the transitiveClosureRf2SnapMulti.pl script.

Enter the following command replacing all instances of:

- **\$RELPATH** with the full path of the release package folder
- **\$RELDATE** with the date element of the release version YYYYMMDD (e.g. 20190131)

```
perl "transitiveClosureRf2SnapMulti.pl" "$RELPATH/Snapshot/Terminology/sct2_Relationship_Snap  
shot_INT_$RELDATE.txt" "$RELPATH/xder_transitiveClosure_Snapshot_INT_$RELDATE.txt"
```

Hint: Copy and paste the command line template into a text editor. Make the replacements and then copy and paste then copy and paste the result to the command line.

In Windows use the command line

Change directory to the folder containing the transitiveClosureRf2SnapMulti.pl script.

Enter the following command replacing all instances of:

- **\$RELPATH** with the full path of the release package folder
- **\$RELDATE** with the date element of the release version YYYYMMDD (e.g. 20190131)

```
perl "transitiveClosureRf2SnapMulti.pl" "$RELPATH\Snapshot\Terminology\sct2_Relationship_Sn  
apshot_INT_$RELDATE.txt" "$RELPATH\xder_transitiveClosure_Snapshot_INT_$RELDATE.txt"
```

Copy and Edit the Loader Script

Make a copy of the loader script.

Open the loader script in a text editor that supports plain text and search and replace.

The following notes are also present at the top of the loader script file and explain how to ensure the script will work correctly for you.

1. RELEASE PACKAGE AND VERSION

This template script is specific a specific version: yymmdd

of a specific release package: SnomedCT_InternationalRF2_PRODUCTION_yymmddThhmmssZ

If this is the release package version you are importing please skip to section 3 of these notes.

2. PACKAGE AND VERSION CONFIGURATION

If you are working with a different version of the same release package this script may be adapted to import that package.

- First replace all instances of:
SnomedCT_InternationalRF2_PRODUCTION_20190131T120000Z
with the release folder name for the release package version to be loaded.
- Then replace all instances of the date stamp: 20190131
with the date stamp for the version to be imported.

IMPORTANT NOTE

Different release packages or release package versions may contain different sets of files.

- If expected files are missing the script will fail.
- If files present in a imported release package are not in the release for which this script was developed those additional files will not be imported.

Updated versions of this script may be available for newer production releases of SNOMED CT International Edition packages.

3. FOLDER LOCATION CONFIGURATION

This templated version of the file contains placeholders \$RELPATH

Replace all instances of \$RELPATH with the fullpath of the folder:

SnomedCT_InternationalRF2_PRODUCTION_20190131T120000Z.

4. IF YOU ARE NOT IMPORTING A TRANSITIVE CLOSURE FILE

You are recommended to create a transitive closure file for import. However, if you do not want to have a transitive closure table or are unable to create the transitive closure file then find the line towards the end of this file that contains the text: #TRANSCLOSE#

Delete that line and all the lines that follow it up to the end of the file. If you do not do this, the script will complete to that point but will report an error when it completes.

5. SAVE THE SCRIPT

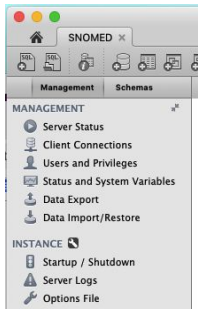
Save a copy of the SQL script with any modifications you have made.

Running the SQL Modified Script

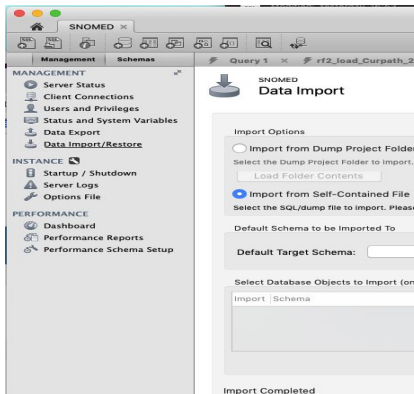
Option 1

The fastest way to import the data is to use the MySQL Workbench as follows:

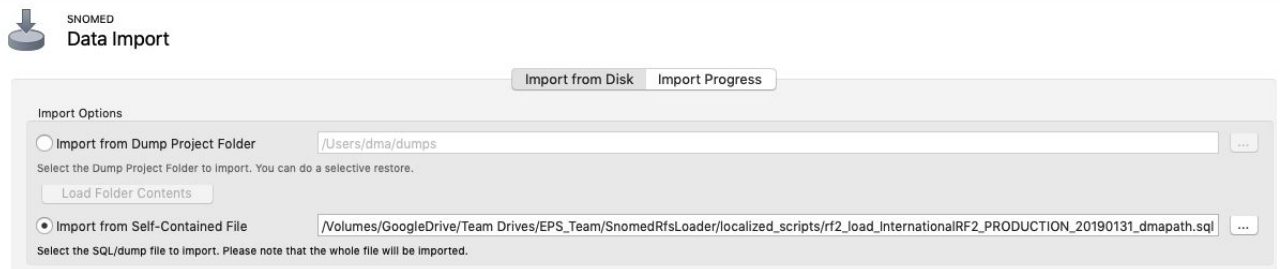
- a. Select the **Management** tab at the top of the left margin.



- b. Select the option **Data Import/Restore** from the options in the left margin.



- c. Choose the "Import from Self-Contained File option
- d. Then select the SQL script file to be imported (the file you edited to point to the location of the release pack on you system)





- e. Click the **Start Import Button**
- f. The import will start and you can review progress by selecting the **Import Progress** button near the top of the frame.
- g. You will then see a message appear in the log file with a message similar to the following appearing in the **Log** frame:

```
17:50:43 Restoring
/your-script-path/rf2_load_InternationalRF2_PRODUCTION_20190131.sql
Running: /Applications/MySQLWorkbench.app/Contents/MacOS/mysq
--defaults-file="/var/folders/15/bf9slzvd60n6lw919jbpwd5m0000gn/T/tmp32x8gI/extrap
arams.cnf" --protocol=tcp --host=localhost --user=root --port=3306
--default-character-set=utf8 --comments <
"/your-script-path/rf2_load_InternationalRF2_PRODUCTION_20190131.sql"
```

Using this approach the script will usually take between 15 and 40 minutes to run.

Note: If you are unable to access the Import functionality described here, you can also run the script to import the data in several other ways described on the following page.

Option 2

- Open MySQL Workbench and then open an SQL query using the  button.
- Select the SQL script file copy that you saved after any necessary edits (as described in the previous section).
- Click the lightning bolt icon  in the bar above the query window.

Note: The script may take between 30 minutes and 2 hours to run depending on the speed and memory available in the system. Do not assume it has failed because nothing seems to be happening as some steps in the process are much slower than others.

Option 3

Another way to run the script in the MySQL Workbench is to use the File / Run SQL Script option. Then select the script file and click the run button. The end result is the same but the processing is done with less feedback to than with Option 2.

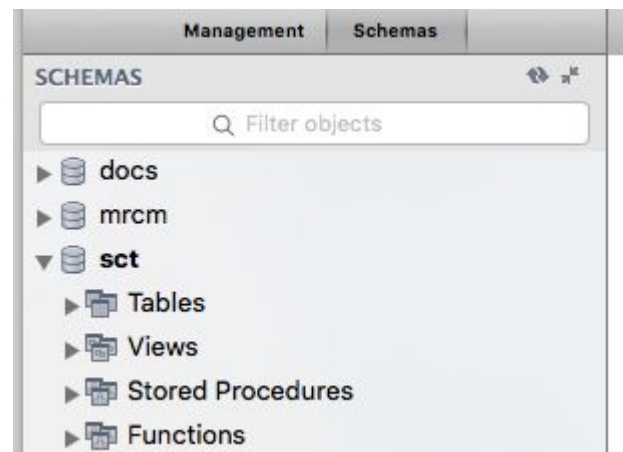
Note: In this option the user interface may report completion after reading and analyzing the script. After this it may appear to hang but the processing of the database is still continuing so leave it to finish. Actual completion will take between 20 minutes and 2 hours depending on system performance.

Option 4

Those familiar with MySQL command line may also run this script as input to the command line processor call rather than using the MySQL Workbench interface.

Successful Completion

In MySQL Workbench go to the schema tab in the left hand frame. If you do not see the **sct** schema right click and select **refresh all**. You should see a new database schema **sct**. Containing the tables, views, stored procedures and functions required to provide direct access to the SNOMED CT data and to optimize/simplify various common viewing requirements.



Information About the Database

The following notes summarize the naming conventions and usage of various components of the generated database.

Tables

A database table is created for each file in all subfolders of the Full release folder.

Table Names

The name of each table follows the following conventions based on but not identical to the names of the each release file.

- Prefix "sct_" (this prefix applied to all tables replacing the file prefixes "sct2_", "der2_", etc.
- Component tables are named by component type starting the first word with a lowercase letter (e.g. sct_concept, sct_description, sct_textDefinition).
- Refset tables are named "refset_" followed by the name of the refset (e.g. sct_refset_Language, sct_refset_ExtendedMap)
- The remaining elements of the release filename are omitted.

Column Names

The columns of the tables match those of the release file with one additional column (supersededTime). This additional column contains the date/time at which this row in the table was superseded by another version of the same component with a more recent effectiveTime. Thus the supersededTime of row that has been superseded is the effectiveTime of the immediately following version of that component or refset member. For rows that have not been superseded (i.e. rows that are current at the time of the most recent release) the supersededTime is set to a distant future date, indicating that the row has not been superseded. The supersededTime is generated during the load process and is used to optimize snapshot views.

Column Datatypes

- SNOMED CT identifiers are represented by the BIGINT datatype.
- UUID's are represented by the BINARY(16) datatype.
 - Note this datatype provides an efficient storage and keys, but does not display directly in queries. Where required it can be by the ShowUuid(id) function.
- Other integer values are represented by the INT datatype.
- Boolean values are represented by the TINYINT datatype.
- Short text strings are represented by the VARCHAR() datatype.
- Longer text strings are represented by the TEXT datatype.
- Dates and times are represented by the DATETIME datatype.

Additional Tables

There are four additional tables. Two of these are concerned with configuration (config_language and config_settings). The other two contain alternative views of subtype relationships. A snapshot view of the transitive closure of all subtype relationships (ss_transclose) and a snapshot view of the relationships between each concept and its proximal primitive supertypes (ss_proximal_primitives).

Snapshot Table Views

There are four snapshot views of each of the SNOMED CT tables. These have the same names as the tables to which they related but have a different prefix according to the view they represent.

Snapshot Prefixes

- `sva_` The current snapshot view (i.e. the most recent version of each component or refset member in the table)
- `svx_` The snapshot view at the date specified in the `config_settings` file.
- `soa_` An optimized view of the current snapshot (i.e. the most recent version of each component or refset member in the table)
- `sox_` An optimized view of the snapshot at the date specified in the `config_settings` file.

This date of the snapshot for the `svx` and `sox` views be set and changed using the `setSnapshotTime()` procedure.

For example: `Call setSnapshotTime("2017-07-31");`

The optimized (`soa` and `sox`) views return the same content as the (`sva` and `svx`) views. The only difference is how those views are generated. In some queries the optimized versions perform significantly better in others the different may be imperceptible. Occasionally, in particular types of queries that return the complete snapshot view of a table the optimized queries may be slower.

Additional Combined Views

In addition to the snapshot views of individual tables, a range of useful combined views are provided. In all cases these views use the same prefixes (`sva_`, `svx_`, `soa_` or `sox_`) to indicate the snapshot view on which they are based.

The names following each prefix indicate the combined content that will be returned.

In all cases, the descriptions and terms returned are those specified by the language setting in the `config_settings` table. This can be set or updated using the `setLanguage()` procedure.

For example: `Call setLanguage("en-GB");`

The following views can be used to view descriptions associated with a particular concept (specified by `WHERE conceptId=id`)

In all cases, the descriptions and terms returned are those specified by the language setting in the `config_settings` table. This can be set or updated using the `setLanguage()` procedure.

For example: `Call setLanguage("en-GB");`

- `s._fsn` - The FSN of a concept (specified `conceptId`)
- `s._pref` - The preferred synonym of a concept (specified `conceptId`)
- `s._syn` - The acceptable synonyms of a concept (specified `conceptId`)
- `s._synall` - The preferred and acceptable synonyms of a concept (specified `conceptId`)

Most of the other views have either a `_fsn` or `_pref` suffix indicating which term should be displayed for each referenced concept.

The following views return the relationships of a concept (specified by `WHERE src_id=id`) together with a display term for the source (`src_term`), type (`type_term`) and destination (`dest_term`):

- `s._rel_fsn` (or `rel_pref`) : All relationships
- `s._rel_def_fsn` (or `re_def_pref`) : Only defining attribute relationships not [is a] relationships.

The following views return the id and display term for the subtype children or supertype parents of a concept (specified by `WHERE conceptId=id`)

- `s._rel_child_fsn` (or `rel_child_pref`)
- `s._rel_parent_fsn` (or `rel_parent_pref`)

The following view returns the preferred terms for proximal primitive supertypes (specified by

WHERE subtypeId=**id**)

- s._proxprim_pref

Functions and Procedures

Functions

setLanguage(**languageCode**)

- Sets the default language for term display.
- Requires a valid language code (with the International Edition only en-US and en-GB are supported).
- The built in functionality will support other languages where the relevant descriptions and language refset members are included. Details of additional languages also need to be added to the config_language table.

setSnapshotTime(**datetime**)

- Sets the date/time on which the snapshot view with prefixes svx_ and sox_ are based.
- Can be any valid date / time string. Recommended format for the date is YYYY-MM-DD (e.g. "2018-01-31").
- The date does not have to coincide with a release date as the snapshot simply looks for the latest version of each component prior to the specified date.

Procedures

getLanguage()

- Returns the current Language as a refsetId (not the languageCode). This is because the primary use of the function is internal where it is used to select the language reference set.

getSnapshotTime()

- Returns the DATETIME value representing the most recently applied setting of the snapshot time. This is used internally when generating the svx_ and sox_ views.

showUid(**id**)

- Returns the standard UUID text string rendering for the id presented.
- This should be used if you want to display the id column from a refset.

For example:

```
SELECT ShowUid(`id`), `effectiveTime`, `active`, `moduleId`, `refsetId`,  
       `referencedComponentId` FROM `sox_refset_simple`;
```

Example Queries

A set of example queries that have been written to demonstrate some of the features of the database are also provided in the download package.