

**DỰ ÁN PHÁT TRIỂN HỆ THỐNG THÔNG TIN NGÀNH LÂM NGHIỆP VIỆT NAM – FORMIS II**

**DATA CONVERTER ARCHITECUTRE**

**Hanoi, 10/2017**

**F O R M I S**

Development of A Management Information System for The Forestry Sector in Vietnam

**CHANGES**

\*A – Create, M – Update, D – Delete

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Position** | **A\***  **M, D** | **Orginal** | **Old version** | **Description** | **New version** |
| 29/11/2017 |  | A |  | 0.1 |  |  |
| 07/03/2018 |  | A, M |  | 0.2 | introduce LDB patching |  |

**TABLE OF CONTENT**

[1 OVERVIEW 5](#_Toc511131750)

[1.1 Purpose 5](#_Toc511131751)

[1.2 Scope 5](#_Toc511131752)

[1.3 Threats 6](#_Toc511131753)

[1.3.1 Upgrading LDB might corrupt datamodel but not data 6](#_Toc511131754)

[1.3.2 Upgrading LDB from v1.4.1 and v2.0.x to v3.0.x will create new UUID of data stored in LDB 6](#_Toc511131755)

[2 WORKFLOW 7](#_Toc511131756)

[2.1 Overall workflow 7](#_Toc511131757)

[2.2 Backup local database 8](#_Toc511131758)

[2.3 Backup local data workflow 9](#_Toc511131759)

[2.4 Connect to central database 10](#_Toc511131760)

[2.5 Determine database version 10](#_Toc511131761)

[2.6 Upgrade local database 10](#_Toc511131762)

[2.7 Workflow for clean up database 11](#_Toc511131763)

[2.8 Worflow for copy data from local to server 12](#_Toc511131764)

[2.9 Workflow to test data 13](#_Toc511131765)

[3 SOFTWARE PACKAGE 13](#_Toc511131766)

[4 Software maintenance 14](#_Toc511131767)

[4.1 Upgrade patches 14](#_Toc511131768)

[4.2 Testing patches 14](#_Toc511131769)

[4.3 Database versions 15](#_Toc511131770)

# OVERVIEW

## Purpose

This document is description of workflow for data convert tool. This tool was developed for some special cases to copy data from local to server as temporary solution to copy data from Local Database to Central Database. The two cases the tool is intended for is:

* In case standard synchronization of data fails
* In case if users didn’t synchronize their local data to CDB before a version upgrade

Synchronization might fail because of several reasons, like lack of stable network connectivity or corrupted data or software bugs of implementation. In these cases delivering the LDB to a PC with stable conncetion to CDB user can directly copy data by the tool to CDB practically overwriting it.

## Scope

The tool is capable to copy data between different versions of LDB and CDB in case CDB is newer than LDB. It upgrades LDB by various SQL conversion scripts to match the version of CDB and then copy data over to CDB.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Central Database version | | | | | | |
|  |  | 1.4.1 | 2.0.0 | 2.0.1 | 2.0.6 | 2.0.7 | 3.0.0 | 3.0.1 |
| Local Database version | 1.4.1 | ✔ | ✔↟ | ✔↟ | ✔↟ | ✔↟ | ✔↟ | ✔↟ |
| 2.0.0 |  | ✔ | ✔↟ | ✔↟ | ✔↟ | ✔↟ | ✔↟ |
| 2.0.1 |  |  | ✔ | ✔↟ | ✔↟ | ✔↟ | ✔↟ |
| 2.0.6 |  |  |  | ✔ | ✔↟ | ✔↟ | ✔↟ |
| 2.0.7 |  |  |  |  | ✔ | ✔↟ | ✔↟ |
| 3.0.0 |  |  |  |  |  | ✔ | ✔ |
| 3.0.1 |  |  |  |  |  |  | ✔ |

✔ Copy of data is possible  
↟ Upgrade of LDB is necessary

## Threats

### Upgrading LDB might corrupt datamodel but not data

The upgrade process of LDB happens for the sake of being able to copy data into CDB. The upgraded LDB after the process might not match properly the newer datamodel especially regarding to functions and triggers. Thus, create a file backup of the LDB before running the tool and restore it after copy is done. Don’t continue working with the updgraded LDB after data is copied from it to CDB.

The intended workflow requires that after upgrade and copy is done user discards the LDB and FRMS installation and start with a fresh installation of the newer version FRMS and executes an initial load.

### Upgrading LDB from v1.4.1 and v2.0.x to v3.0.x will create new UUID of data stored in LDB

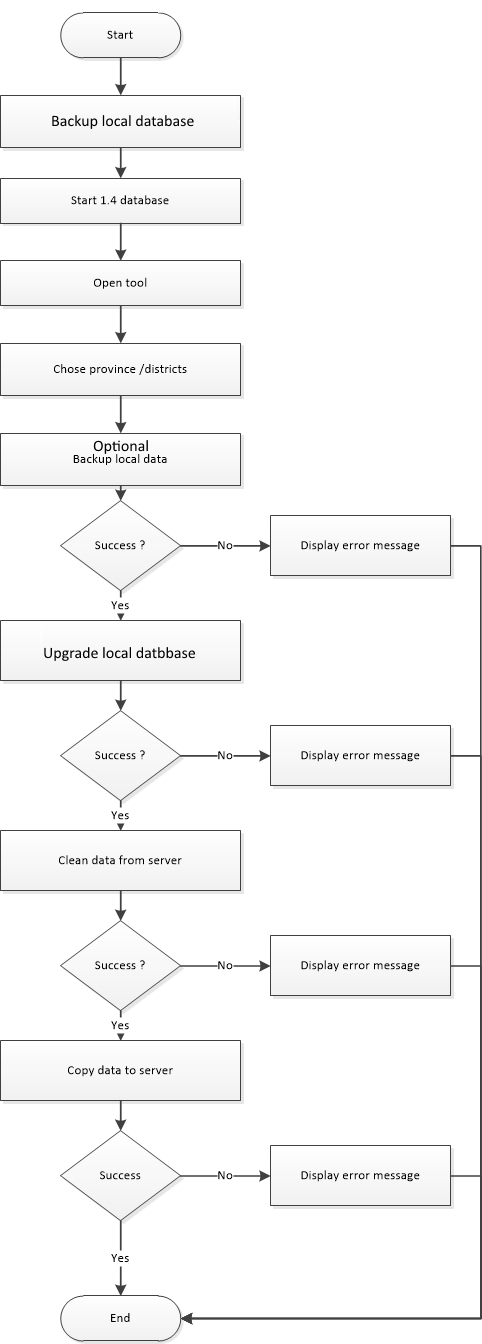
The upgrade scripts that bring LDB to version v3.0.x compatible will generate UUID primary keys for data in LDB and set up the relations according to these newly generated values. This doesn’t cause a confusion if user wants to replace content in CDB for the specific region with the content of LDB and there are no other users who work on the affected region using data in CDB.

If some users already checked out data from the affected region from CDB and the tool overwrites it from LDB the new data will be loaded to users by synchronization (redoing initial load is recommended) but their changes will be lost.

Pay attention to what region is overwritten and which users might be affected.

# WORKFLOW

## Overall workflow

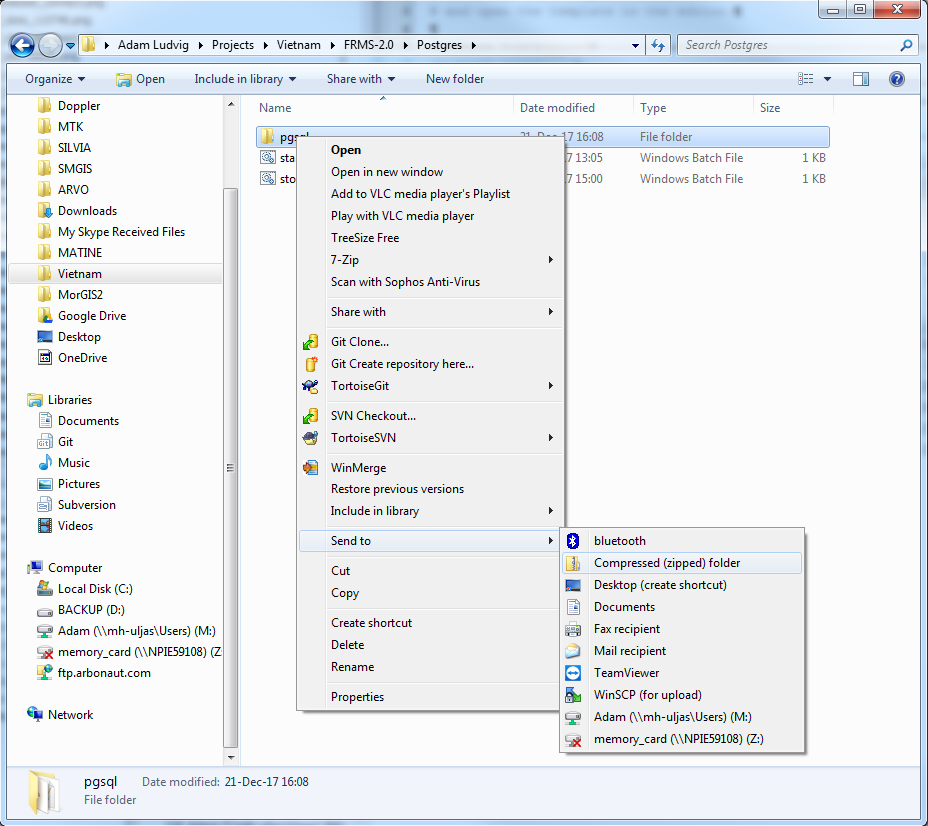


## Backup local database

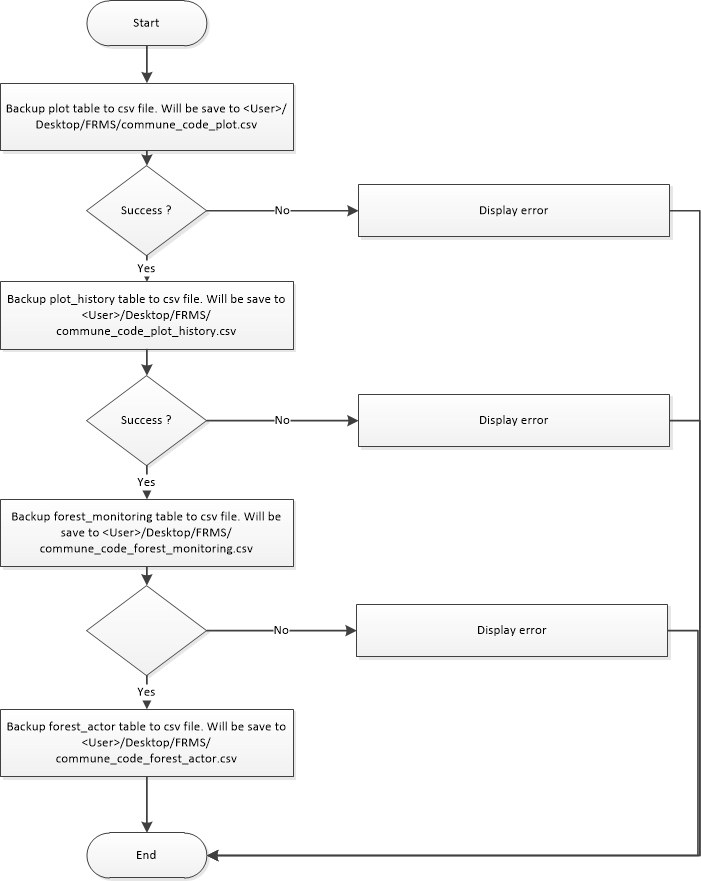
Manual step to have a backup of local database. Important, don’t skip this step as upgrading the schema of LDB might break it to be further used in FRMSClient, see the Threats above in the document.

Take a copy of FRMS-1.4.1\Postgres\pgsql folder or FRMS-2.0\Postgres\pgsql.

An easy solution is to create a zip of it and in case of restoring LDB just delete old psql folder and extract from zip archive.



## Backup local data workflow



## Connect to central database

The application uses the configuration file src/remote\_database.properties to connect. It has the following format. database.password.0 property is mandatory, other properties starting with database.password are optional.

database.host=vnforest.gov.vn

database.port=5434

database.name=data\_forest

database.classname=org.postgresql.Driver

database.type=postgresql

database.username=postgres

database.password.0=Friend5

database.password.1=Friend6

database.password.2=Friend8

You can specify several passwords for central database by using multiple lines by the pattern of database.password.%d, where %d should be substituted by 0 based ascending integer numbers. The application tries to use these passwords one by one when connecting the central database. Changing passwords is a way to prevent old tools connecting to newer central database that are not enchanced to use newer datamodel properly.

The source package contains remote\_database\_viet.properties and remote\_database\_arbo.properties files containing preconfigured settings for FRMS production environment and Arbonaut test environment. The application ignores these files, only remote\_database.properties is taken into account.

## Determine database version

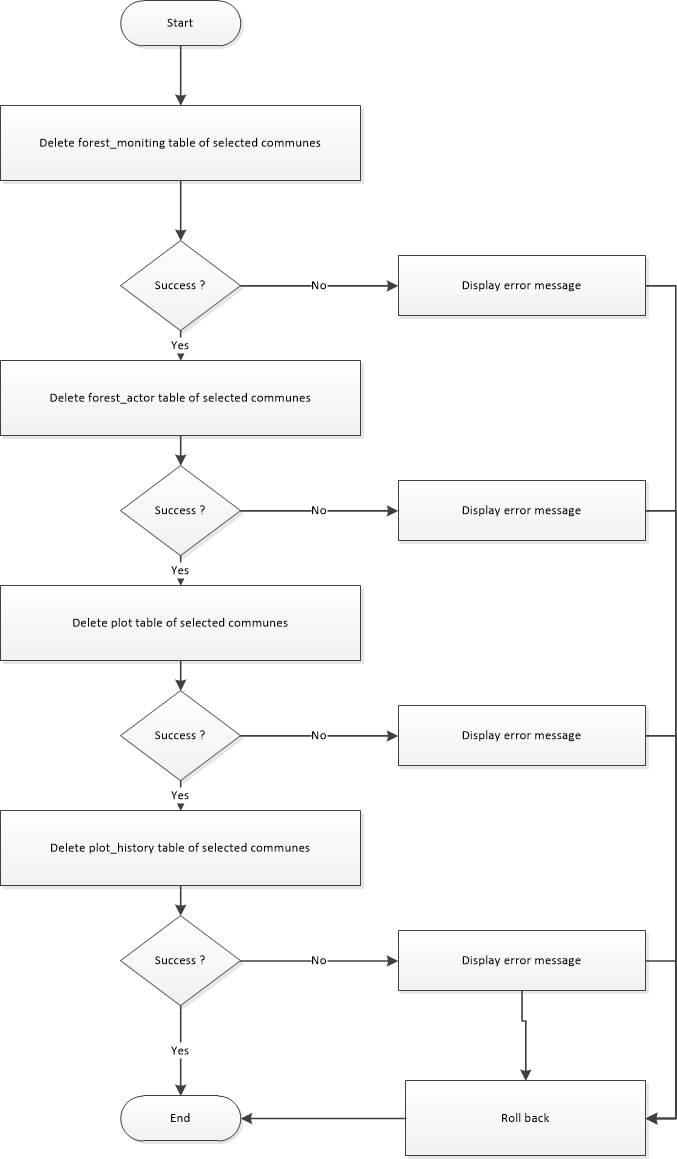
The application sources contain a series of database patches in src/patches folder and a series of tests based on these patches that can be executed against a database connection. When connected to a local or central database the application runs these tests to find out witch version of datamodel the database uses.

The application tries the tests in descending order and if a database satisfies all tests then the application assumes that the database has the determined version of datamodel.

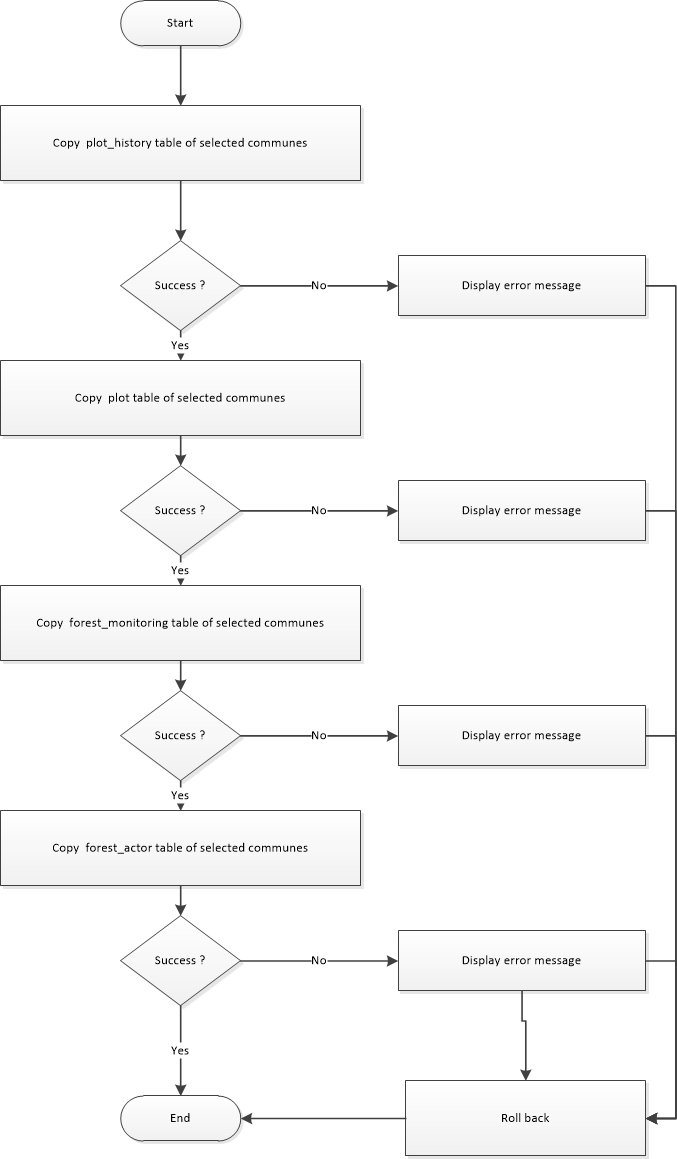
## Upgrade local database

The application determines the version of central database and local database and then executes a sequence of database patches on local database to reach the level of datamodel of central database.

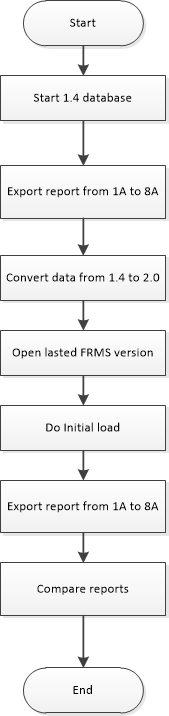
## Workflow for clean up database



## Worflow for copy data from local to server

Datamodel versions are determined and in case version of local and central datamodel differ the copy workflow is interrupted, and an error message is displayed about specific versions of datamodels. 

## Workflow to test data



# SOFTWARE PACKAGE

* **vnforest.formis.common**: contains common java classes
* vnforest.formis.db.model: contains java classes to populate data from server
* **vnforest.formis.db.processor**: contains java classes (extends SwingWorker) to convert process data
* **vnforest.formis.ui**: contains java classes for user interface
* **vnforest.formis.ui.datamodel**: contains java classes to bind data to UI elements
* **vnforest.formis.utils**: contains utilities java classes

# Software maintenance

## Upgrade patches

The source directory patches contain sets of SQL scripts used to upgrade LDB and CDB. It also includes some CDB patches to overcome previous version upgrades when users were required to return their data, upgrade it in CDB and then start with a clean copy of FRMSClient installation. The data converter tools upgrade process is implemented to substitute these version upgrades in FRMSClient’s history.

The patches directory has subdirectories following pattern 00dd where dd is the incremental version number of the datamodel containing a set of SQL scripts to execute agains LDB.

SQL patch files needs to be implemented idempotentially, the implementer must be certain that executing the patch can be executed several times without causing harm or failure. An example could be 0002-C-001\_update\_datamodel.sql from directory 0002 where each ALTER TABLE ADD COLUMN command is covered by a DO code block and proper exception handling. This behavior is necessary in case an SQL script has more than one change to data or datamodel and in case a later change in the script fails the tool reexecutes the whole script. If this requirement is too difficult to satisfy developer might separate complex sql queries into simple, trivial ones and support them with proper tests so the tool doesn’t try to reexecute the script if the LDB already satisfies the tests.

DatabasePatch.java source file in package vnforest.formis.utils has to be aware of new patches. Developer must specify the path of SQL upgrade script and can specify tests to check LDB whether it is necessary to execute the upgrade script. If developer doesn’t specify test then the tool assumes executing the script doesn’t to any harm.

## Testing patches

The developer might specify test against LDB for each upgrade SQL script in DatabasePatch.java source file in package vnforest.formis.utils. The following classes are implemented to help quickly implement simple test on data or metadata.

* HasTablesTest(String… tableName)
* HasTableColumnsTest(String tableName, String… columnName)
* SqlTest(String query)

With these tools developer can check whether some tables exists, what columns of a table has or execute arbitrary sql query returning a Boolean value. If these tools do not satisfy the developer’s demand custom tests can be implemented using the interface DatabaseTest.java.

Predefined test are implemented as static fields of DatabaseTest.java for reusability and to easily match them with new DatabasePatch objects.

## Database versions

Developer can define a DatabaseVersion in DatabaseVersion.java in package vnforest.formis.utils by selecting a set of DatabasePatches. The tests of the selected DatabasePatches will be used to determine CDB and LDB version level and the set of patches to be executed to updgare LDB to the same version as CDB is.

**F O R M I S | Development of Management Information System for Forestry Sector in Vietnam**

**Supported by:**



**Implementing Agency: Lead consultant: In consortium with:**



FORMIS II Office: For further information,

**Mr. Truong Le Hieu**

FORMIS National Coordinator

Email: nguyenba73@yahoo.com

Tel. +84 91 303 7950

**Mr. Tapio Leppänen**

FORMIS Chief Technical Adviser (CTA)

Email: tapio.leppanen1@gmail.com

Tel. +84 123 723 6298

31 Hoang Hoa Tham Str.

Ba Dinh distr., Hanoi, Vietnam

Tel. (+84 4) 3722 8733/8734