[Database installation 2](#_Toc342313177)

[Database folder structure 2](#_Toc342313178)

[Setting up the database using the .backup 3](#_Toc342313179)

[Creating an empty database 3](#_Toc342313180)

[Restoring from the network\_interdependency.backup (using pgAdmin) 3](#_Toc342313181)

[Restoring from the network\_interdependency.backup (using pgrestore) 4](#_Toc342313182)

[Setting up the database using the ni\_setup\_database.bat script 6](#_Toc342313183)

[Note: Handling database passwords 6](#_Toc342313184)

[Setting up the database using .bat and .sql scripts 7](#_Toc342313185)

## Database installation

### Database folder structure

The .sql files that make up the functions and tables of the database schema are stored within the folder structure defined below, alongside the documentation for the schema.

* **backup/** - contains the latest network\_interdependency.backup backup schema file
* **doc/** - documentation applicable to whole database schema e.g. installation and usage tutorial
  + **er\_examples/** – examples of Entity-Relationship diagrams for the network database schema, and examples of networks created within the schema
  + **functions/**
    - **generic/** - documentation for all generic functions
    - **processing/** - documentation for all processing functions
    - **trigger/** - documentation for all trigger functions
  + **tutorials/** - tutorials to guide on how to:
    - * install the database schema (this document)
      * import data from other network formats (***Tutorial – Importing networks from other formats.docx***)
      * export data to other network formats (***Tutorial – Exporting networks to other formats.docx***)
      * creating a national-scale rail network from Ordnance Survey Meridian 2 data (***Tutorial – Creating national-scale rail network from OS Meridian 2 data.docx***)
* **functions/**
  + **generic/**
    - \*.sql – all sql files contained in this folder define the generic functions of the database schema. These functions help to administer the database, but more can be found by reading the documents stored in the /FUNCTIONS/GENERIC/DOCS folder
    - \*.bat – batch files to aid installation of generic functions on Windows-based machines
  + **processing/**
    - \*.sql – all sql files contained in this folder define the processing functions of the database schema. These functions can be used to help pre-process data prior to network creation. More information can be found in the documents stored in the /FUNCTIONS/PROCESSING/DOCS folder
    - \*.bat – batch files to aid installation of processing functions on Windows-based machines
  + **trigger/**
    - \*.sql – all sql files contained in this folder define the trigger functions that operate on the different tables that make up the database schema itself. These functions try to help maintain the integrity of the topology of a network once created. More information can be found in the documents stored in the /FUNCTIONS/TRIGGER/DOCS folder.
    - \*.bat – batch files to aid installation of trigger functions on Windows-based machines
* **tables/**
  + \*.sql – all sql files contained in this folder define the tables that make up the database schema. Each file effectively defines another table.

### Setting up the database using the .backup

#### Creating an empty database

1. Open pgAdmin
2. Connect to your chosen database server
3. Right-click “Databases”
4. Select “New Database”. You should now see a window similar to Figure.1

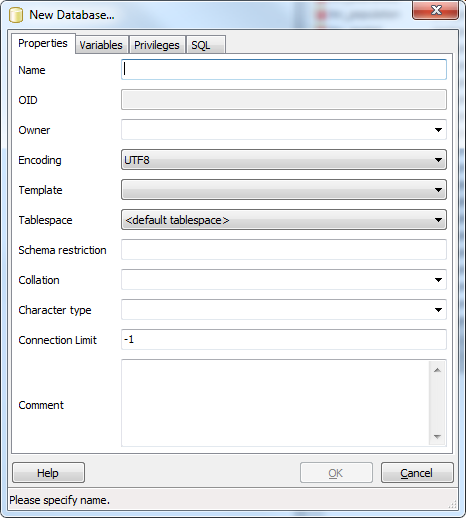


Figure - New database dialog in pgAdmin

1. Fill in the name for your database e.g. transport
2. Select “postgres” as the owner of your database, from the drop down list
3. Select the appropriate encoding for your database. Frequently UTF-8 is sufficient.
4. Select “template\_postgis” template from the drop down list of templates.
5. Add any comments into the “Comment” box that you wish e.g. “This database will contain raw transport data, alongside spatial / topological representations of transport networks, created via the use of the nx\_pg/nx\_pgnet Python modules, networkx and the custom network interdependency database schema”

#### Restoring from the network\_interdependency.backup (using pgAdmin)

1. Open the database by single left-clicking on it. This should turn the default database disconnected symbol  into the default database connected symbol .
2. Right-click on the connected symbol and select “Restore”. You should now see the restore dialog, as seen in Figure.2:

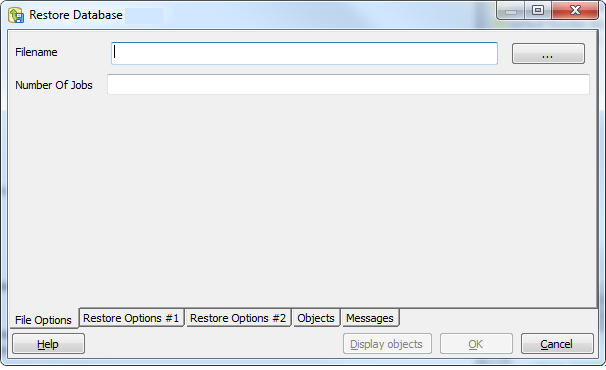


Figure - pgAdmin database restore dialog window

1. Click the browse button , and navigate to the folder where the network\_interdependency.backup file is located, within the backup folder.
2. Select the network\_interdependency.backup file. This should populate the Filename box at the top of the dialog window.
3. Leave all other remaining restore options as defaults, and click OK. Please be patient as this may take a few moments.
4. The result should be a database that contains the following tables (excluding the default spatial tables “spatial\_ref\_sys” and “geometry\_columns”):

“Edge\_Geometry”

“Edges”

“Global\_Interdependency”

“Graphs”

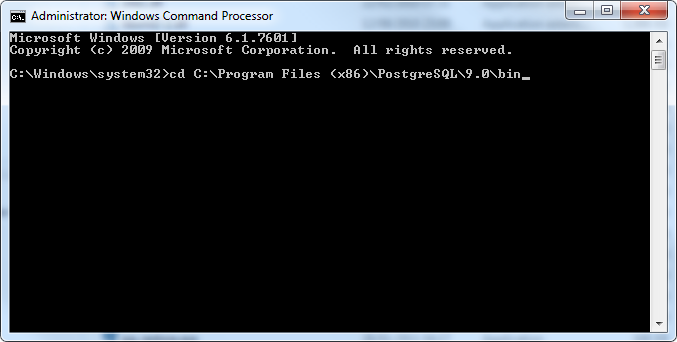
“Interdependency”

“Interdependency\_Edges”

“Nodes”

#### Restoring from the network\_interdependency.backup (using pgrestore from command line)

1. Open a command line window
2. Open Windows Explorer
3. Navigate to the bin folder of your PostgreSQL installation
4. Copy this path e.g. C:\Program Files (x86)\PostgreSQL\9.0\bin
5. In the command line window, type CD, and then paste the bin folder path, and press Enter e.g.



1. This will move the user in to the directory where the pg\_restore.exe is located. It is this tool that will be used to restore the database schema, and effectively load the database schema in to your chosen destination database
2. Type the following restore command, adjusting values to your local environment, where specified:

pg\_restore -U <insert\_postgresql\_username> -p <insert\_postgresql\_port\_number> -h <insert\_host\_name> -d <insert\_database\_name> <insert\_path\_to\_schema\_backup>

e.g. pg\_restore -U postgres -p 5432 -h localhost -d 'a\_spatial\_database' C:/SCHEMA/schema.backup

1. Press Enter when you have finished entering this information to install the database.

### Setting up the database using the ni\_setup\_database.bat script

If you are a user who possibly already has a spatial database with all of the necessary data that you wish to use stored within this database, and therefore does not want to overwrite this data by restoring from the .backup, then using the .bat scripts is one possible solution to delivering the custom schema in to your chosen database. The .bat script essentially points at a number of other .bat scripts (that too can also be executed independently with the same parameters as needed here), which install the processing functions, trigger functions, tables and generic functions that make up the schema. The installation must occur in this order to avoid dependency errors arising within the database e.g. a function calling a table that does not yet exist, or a trigger being referenced from a table where the trigger function has not been defined.

Having installed PostgreSQL and PostGIS, in order to install the schema using the .bat scripts will require the following information:

* Host name e.g. localhost
* Postgres user name e.g. postgres
* PostgreSQL bin folder name e.g. C:\Program Files (x86)\PostgreSQL\9.0\bin\
* Chosen database name e.g. “transport”

#### Note: Handling database passwords

In order to be able to execute the .bat script to install the database schema, the password for the chosen database and postgres user must be known. The most sensible way of storing this password for connection purposes, including executing the installer described here, is to use *pgpass.conf* files. More information on these files can be found [here](http://www.postgresql.org/docs/9.1/static/libpq-pgpass.html), and it is recommended that this is **read** and **configured** before proceeding.

**Note: DO NOT ADJUST THE FOLDER STRUCTURE OF THE DATABASE CODE BASE, OTHERWISE THIS INSTALLER WILL NOT EXECUTE CORRECTLY.**

1) Locate the ni\_setup\_database.bat file

2) Open a command line prompt window from the folder containing the ni\_setup\_database.bat file (hold the Space bar and right mouse click to select “Open Command Window here”)

3) Once the window has opened, type the following, substituting your values for those contained in the <> brackets:

ni\_setup\_database.bat <hostname> <postgres username> <PostgreSQL bin folder location> <database name>

4) Press return, and this should set of the script to install the database, by calling the necessary .sql files.

Alternatively, it is possible to edit the file **execute\_ni\_setup\_database.bat** using any simple text editor, and change the default values for hostname, postgres username, bin folder location and database name to your chosen values resulting in a .bat script that you can execute at your convenience and one that does not require you to enter multiple parameters each time, but only one i.e. the database name.

### Setting up the database using .bat and .sql scripts

The sql files that define the schema are stored within a defined folder structure, as described in the section ***Database folder structure***. It is possible to *install* the schema either via using the provided .backup, or by using the predefined ni\_setup\_database.bat script with the correct parameters defined.

The final alternative way to install the schema is to run the execute\_ni\_combine\_sql\_files.bat script. This script effectively combines all the sql files within each folder into a single .sql file for each folder, that can then individually being executed on the database in question. The result is:

* ni\_processing\_functions.sql – all processing function .sql files at /FUNCTIONS/PROCESSING/ being combined, ready for execution
* ni\_trigger\_functions.sql – all trigger function .sql files at /FUNCTIONS/TRIGGER/ being combined, ready for execution
* ni\_tables.sql – all table definition .sql files at /TABLES/ being combined, ready for execution
* ni\_generic\_functions.sql – all generic function .sql files at /FUNCTIONS/GENERIC/ being combined, ready for execution.

If a user decides to install the schema by using this method, they **MUST** ensure that the files are executed in the following order):

1. Processing Functions
2. Trigger Functions – the trigger functions must exist before the tables are defined, otherwise an error will occur where a table cannot be created because it cannot find the appropriate trigger function within the destination schema
3. Tables
4. Generic Functions