

---

# **CityGML 3D City Database Utilities Package**

## **PostgreSQL Version**

### **Documentation**

Last update: 22 May 2018

## Active participants in development

Name	Institution	Email
Giorgio Agugiaro	AIT – Austrian Institute of Technology G.m.b.H. Center for Energy – Smart Cities & Regions	<a href="mailto:giorgio.agugiaro@ait.ac.at">giorgio.agugiaro@ait.ac.at</a>

## Acknowledgements

The 3D City Database Utilities Package (PostgreSQL version) was mostly developed in the framework of the following international project:

The JPI Urban Europe, ERANET Co-fund Smart Cities/Urban Futures 2015 “**IntegrCiTy**” project (FFG Project number 855078), funded (in Austria) by the Austrian Ministry for Transport, Innovation and Technology and with the support from the European Union’s Horizon 2020 research and innovation programme.



Homepage: <http://iese.heig-vd.ch/projets/integrcity>



# Table of Contents

<b>DISCLAIMER .....</b>	<b>7</b>
<b>INTRODUCTION .....</b>	<b>9</b>
<b>1    CONTENT .....</b>	<b>11</b>
1.1 <i>Insert stored procedures in schema citydb_pkg .....</i>	<i>11</i>
1.2 <i>Views .....</i>	<i>12</i>
1.3 <i>Additional stored procedures in schema citydb_view .....</i>	<i>14</i>
1.4 <i>Trigger functions .....</i>	<i>15</i>
1.5 <i>Lookup tables .....</i>	<i>15</i>
<b>2    INSTALLATION .....</b>	<b>17</b>
2.1 <i>System requirements .....</i>	<i>17</i>
2.2 <i>Automatic installation .....</i>	<i>17</i>
2.3 <i>(Alternative) manual installation .....</i>	<i>18</i>
<b>3    DOCUMENT REVISION NOTES .....</b>	<b>21</b>
<b>NOTES .....</b>	<b>23</b>



## Disclaimer

The *3D City Database Utilities Package*, developed by Austrian Institute of Technology, Center for Energy, Smart Cities and Regions Research Field (AIT), is free software and licensed under the Apache License, Version 2.0. See the file LICENSE file shipped together with the software for more details. You may obtain a copy of the license at <http://www.apache.org/licenses/LICENSE-2.0>.

THE SOFTWARE IS PROVIDED BY AIT "AS IS" AND "WITH ALL FAULTS." AIT MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE QUALITY, SAFETY OR SUITABILITY OF THE SOFTWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

AIT MAKES NO REPRESENTATIONS OR WARRANTIES AS TO THE TRUTH, ACCURACY OR COMPLETENESS OF ANY STATEMENTS, INFORMATION OR MATERIALS CONCERNING THE SOFTWARE THAT IS CONTAINED ON AND WITHIN ANY OF THE WEBSITES OWNED AND OPERATED BY AIT.

IN NO EVENT WILL AIT BE LIABLE FOR ANY INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES HOWEVER THEY MAY ARISE AND EVEN IF AIT HAVE BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.



## Introduction

The 3D City Database Utilities Package contains a number of additional stored procedures, views and trigger functions which extend the “vanilla” version of the 3DCityDB. The Utilities Package is intended to be installed optionally on top of a 3DCityDB database instance, and the overall goal is to offer additional features and functionalities which help in the data management tasks.

As a matter of fact, data handling within the 3DCityDB can be sometimes not immediately straightforward due to the complexity of the database schema (which indeed reflects the richness and complexity of the CityGML data model). For example, data belonging to a specific class can be stored in different linked tables in the database. Therefore CRUD<sup>1</sup> operations (i.e. select, insert, update and delete) may result in complex SQL statements. In order to partially overcome this complexity, the 3DCityDB is already shipped with a number of stored procedures, e.g. the “delete” ones. These are of particular help as the complexity of the ordered, hierarchical deletion process is carried out automatically and hidden from the user.

---

<sup>1</sup> CRUD represents an acronym for the database operations **C**reate, **R**ead, **U**ppdate, and **D**eleate.





# 1 Content

The Utility Package for the 3D City Database provides in the current release:

- Insert stored procedures for some of the “vanilla” 3DCityDB tables;
- Views offering a simplified access to data spread over multiple tables. All views are created in a separated database schema named *citydb\_view*;
- Additional “smart” CRUD stored procedures (for insert and delete operations) in the *citydb\_view* schema;
- Trigger functions built upon the aforementioned “smart” stored procedures which make most of the views updatable. As a consequence, the user can interact directly with the views as if they were normal tables and perform not only select operations, but also insert, update and delete.
- Lookup tables corresponding to the CityGML codelists

New and improved functionalities of the Utilities Package will be added in future releases as developments continues.

## 1.1 Insert stored procedures in schema *citydb\_pkg*

The insert stored procedures allow to insert data into the table they refer to. They are stored in the *citydb\_pkg* schema and are named using the table name prefixed with “insert\_”, e.g. `citydb_pkg.insert_building(...)` or `citydb_pkg.insert_cityobject(...)`.

All stored procedures (also called “functions”, in PostgreSQL) are written in PL/pgSQL and they all have parameters that are called using the named notation<sup>2</sup>. The named notation is especially useful for functions that have a large number of parameters, since it makes the associations between parameters and actual arguments more explicit and reliable. In the named notation, the arguments are matched to the function parameters by name and can be written in any order. Parameters that have default values given in the function declaration need not to be written at all in the call. This is particularly useful since any combination of parameters can be omitted. In order to use an insert function with the named notation, each argument's name is specified using `:=` to separate it from the argument expression. For example:

```
SELECT citydb_pkg.insert_building(  
id := 5094,  
building_root_id := 5094,  
class := 'Residential',  
storeys_above_ground := 5,  
storeys_below_ground := 2  
);
```

---

<sup>2</sup> The other approach is called *positional notation*. For more details, please refer to <https://www.postgresql.org/docs/9.1/static/sql-syntax-calling-funcs.html>

The insert stored procedures have the following characteristics:

- All parameters are defined as DEFAULT NULL, i.e. they are all optional, except for:
  - The OBJECTCLASS\_ID parameter, if the table has such attribute (e.g. table CITYOBJECT);
  - The ID parameter, if the table has a primary key which is *not* generated by a sequence (e.g. tables BUILDING);
  - Both parameters forming the primary key of an association table (e.g. GROUP\_TO\_CITYOBJECT). Such association tables can be easily recognised by the “\_TO\_” string in the table name;
- For tables having the ID generated by a sequence, there are two possibilities:
  - The user does not specify the ID parameter. In this case the next available value is retrieved from the corresponding sequence automatically and assigned to the new record to be inserted;
  - The user specifies the ID parameter. In this case, it is the responsibility of the user to choose the ID value properly, in order to comply with the UNIQUE constraint in the corresponding table. If the ID value is already used, an error will be raised and the insert operation aborted.
- If the GMLID parameter is not given by the user, then a universally unique identifier (UUID) is generated automatically and assigned to the new record to be inserted;
- Upon successful completion, the insert stored procedures return the ID of the inserted record. In case of an association table, the stored procedure returns null.
- In case of an EXCEPTION, a notice is raised with the error message.

## 1.2 Views

As mentioned before, a number of views is installed in the database schema *citydb\_view*. Regarding the **naming convention**, most of the views allow to access data for specific object classes. The name of the view coincides or approximates the name of the corresponding class in the OBJECTCLASS table. For example, view BUILDING presents data of Building objects, while view BUILDING\_PART presents data of BuildingPart objects. In addition, wherever appropriate, not only the OBJECTCLASS\_ID attribute, but also the **CLASSNAME** one (retrieved from the joined OBJECTCLASS table) are included in the view.

**Please note:** in order to access *any* database object in the *citydb\_view* schema, **qualified names**<sup>3</sup> consisting of the schema name and the object name separated by a dot **must be used**. This is a design decision, in order to avoid homonymy issues since some objects (e.g. views) have the same names of other objects in the *citydb* and *citydb\_pkg* schemas. As a matter of fact, the *search\_path* variable of the database instance remains unchanged to the default values (i.e. *search\_path* = *citydb*, *citydb\_pkg*, *public*).

Table 1 lists the views shipped with this version of the 3DCityDB Utilities Package, while Figure 1 presents an example of view BUILDING.

---

<sup>3</sup> For more details, see <https://www.postgresql.org/docs/9.1/static/ddl-schemas.html>

VIEW	UPDATABLE?
address_to_building_ext	
address_to_bridge_ext	
building	✓
building_furniture	
building_installation	
building_installation_exterior	
building_installation_interior	
building_lod0_footprint	
building_part	✓
citymodel	
cityobject_genericattrib	
cityobject_genericattrib_blob	
cityobject_genericattrib_date	
cityobject_genericattrib_geometry	
cityobject_genericattrib_group	
cityobject_genericattrib_integer	
cityobject_genericattrib_measure	
cityobject_genericattrib_real	
cityobject_genericattrib_string	
cityobject_genericattrib_surfgeom	
cityobject_genericattrib_uri	
cityobjectgroup	✓
generic_cityobject	
group_to_cityobject_ext	
land_use	
opening	
opening_door	
opening_window	
plant_cover	
room	
solitary_vegetat_object	
thematic_surface	
thematic_surface_ceiling	
thematic_surface_closure	
thematic_surface_floor	
thematic_surface_ground	
thematic_surface_interior_wall	
thematic_surface_outer_ceiling	
thematic_surface_outer_floor	
thematic_surface_roof	
thematic_surface_wall	
waterbody	

**Table 1: Views shipped with the current version of the Utilities Package. All views are stored in schema *citydb\_view*.**



## 1.4 Trigger functions

For some views (but the number will increase in the future released of the Utilities Package), a number of triggers and trigger functions are added, in order to make the views updatable. Therefore, if a view is also updatable, entries can be updated, inserted and deleted directly interacting with the view. Whether a view is updatable or not is shown in Table 1.

**Please note** that, by design decision, the underlying update trigger functions do *not* allow to change neither the ID nor the OBJECTCLASS\_ID attributes.

Although still limited at the time of writing, the number of database object shipped with the 3DCityDB Utilities Package will increase in the upcoming releases of the Utilities Package. It is nevertheless important to remark that the same concepts will apply for database objects shipped with ADEs (views, stored procedures, triggers, etc.), which will build upon some of the objects shipped with this package.

## 1.5 Lookup tables

A number of lookup tables is also provided. They implement the codelists defined in the CityGML standard. All tables are stored in the *citydb* schema and are named by adding the “lu\_” prefix to each table. Table 2 contains the list of the lookup tables shipped with the current implementation of the Utilities Package.

LOOKUP TABLE	CORRESPONDING CODELIST(S)
lu_appearance_text_mime_type	_Texture_mimeType
lu_aux_traffic_area_function	AuxiliaryTrafficArea_function
lu_aux_traffic_area_surf_material	AuxiliaryTrafficArea_surfaceMaterial
lu_bridge_class	_AbstractBridge_class
lu_bridge_function_usage	_AbstractBridge_function, _AbstractBridge_usage
lu_building_class	_AbstractBuilding_class
lu_building_function_usage	_AbstractBuilding_function, _AbstractBuilding_usage
lu_building_furniture_class	BuildingFurniture_class
lu_building_furniture_function_usage	BuildingFurniture_function, BuildingFurniture_usage
lu_building_installation_class	BuildingInstallation_class
lu_building_installation_function_usage	BuildingInstallation_function, BuildingInstallation_usage
lu_building_roof_type	_AbstractBuilding_roofType
lu_city_furniture_class	CityFurniture_class
lu_city_furniture_function_usage	CityFurniture_function, CityFurniture_usage
lu_cityobjectgroup_class	CityObjectGroup_class
lu_cityobjectgroup_function_usage	CityObjectGroup_function, CityObjectGroup_usage
lu_core_geom_mime_type	ImplicitGeometry_mimeType
lu_int_building_installation_class	IntBuildingInstallation_class

LOOKUP TABLE	CORRESPONDING CODELIST(S)
lu_int_building_installation_function_usage	IntBuildingInstallation_function, IntBuildingInstallation_usage
lu_landuse_class	LandUse_class
lu_landuse_function_usage	LandUse_function, LandUse_usage
lu_plant_cover_class	PlantCover_class
lu_plant_cover_function_usage	PlantCover_function, PlantCover_usage
lu_room_class	Room_class
lu_room_function_usage	Room_function, Room_usage
lu_sol_vegetationobject_class_function_usage	SolitaryVegetationObject_class, SolitaryVegetationObject_function, SolitaryVegetationObject_usage
lu_sol_vegetationobject_species	SolitaryVegetationObject_species
lu_traffic_area_function	TrafficArea_function
lu_traffic_area_surf_material	TrafficArea_surfaceMaterial
lu_traffic_area_usage	TrafficArea_usage
lu_transportation_complex_class	TransportationComplex_class
lu_transportation_complex_function_usage	TransportationComplex_function, TransportationComplex_usage
lu_tunnel_class	_AbstractTunnel_class
lu_tunnel_function_usage	_AbstractTunnel_function, _AbstractTunnel_usage
lu_water_surf_water_level	WaterSurface_waterLevel
lu_waterbody_class	WaterBody_class
lu_waterbody_function_usage	WaterBody_function, WaterBody_usage

**Table 2: Lookup tables shipped with this implementation of the Scenario ADE. All lookup tables are stored in schema *citydb* and prefixed with “*scn\_lu\_*”**

## 2 Installation

The source code and the documentation the 3DCityDB Utilities Package for PostgreSQL can be retrieved here: [https://github.com/gioagu/3dcitydb\\_ade/tree/master/00\\_utilities\\_package](https://github.com/gioagu/3dcitydb_ade/tree/master/00_utilities_package)

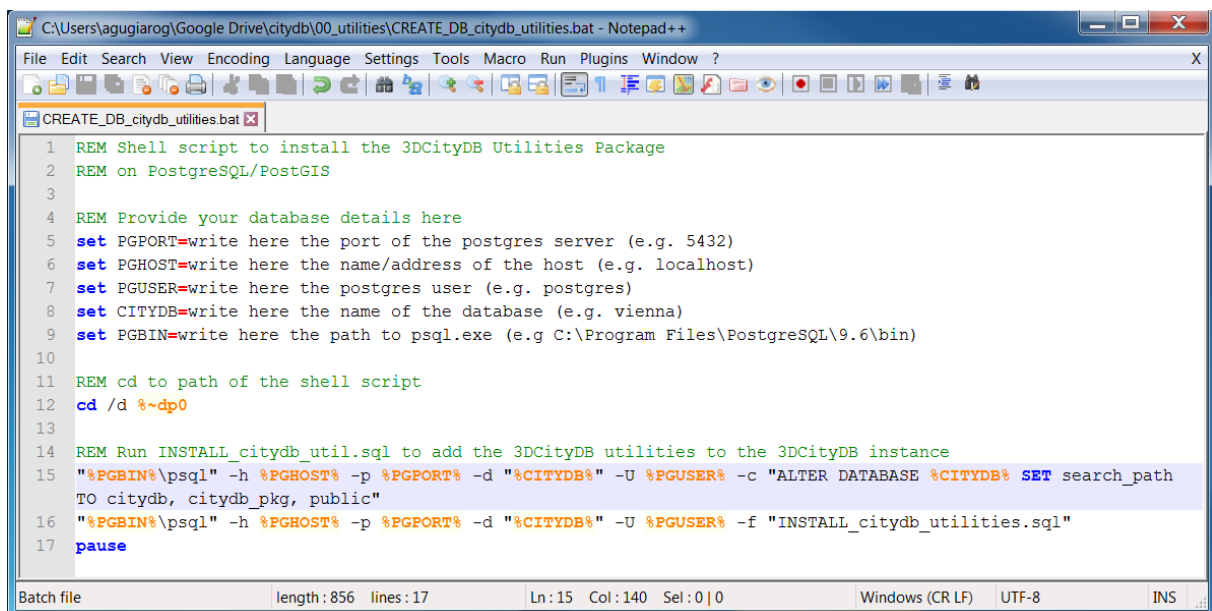
### 2.1 System requirements

In order to set up the 3DCityDB Utilities Package, the 3DCityDB v.3.3.2 needs to have been previously installed correctly. All system requirements of the 3DCityDB still apply. PostgreSQL is supported from v. 9.1. PostGIS 2.0 or higher is required.

### 2.2 Automatic installation

The (recommended) installation process can be carried out nearly completely automatically. The user simply needs to run from a Windows command shell (or simply by double-clicking it) the batch file CREATE\_DB\_citydb\_utilities.bat. However, the configuration parameters regarding the PostgreSQL server and the selected database must be first adapted. A simple text editor will suffice. Please note that the installation should conveniently be carried out using the same user that installed the *citydb* schema.

An example of the CREATE\_DB\_citydb\_utilities.bat file is shown in Figure 2. For Linux environments, the equivalent file CREATE\_DB\_citydb\_utilities.sh is also provided. Upon successful installation, the message shown in Figure 3 will be output.



```

1 REM Shell script to install the 3DCityDB Utilities Package
2 REM on PostgreSQL/PostGIS
3
4 REM Provide your database details here
5 set PGPORT=write here the port of the postgres server (e.g. 5432)
6 set PGHOST=write here the name/address of the host (e.g. localhost)
7 set PGUSER=write here the postgres user (e.g. postgres)
8 set CITYDB=write here the name of the database (e.g. vienna)
9 set PGBIN=write here the path to psql.exe (e.g C:\Program Files\PostgreSQL\9.6\bin)
10
11 REM cd to path of the shell script
12 cd /d %~dp0
13
14 REM Run INSTALL_citydb_util.sql to add the 3DCityDB utilities to the 3DCityDB instance
15 "%PGBIN%\psql" -h %PGHOST% -p %PGPORT% -d "%CITYDB%" -U %PGUSER% -c "ALTER DATABASE %CITYDB% SET search_path
    TO citydb, citydb_pkg, public"
16 "%PGBIN%\psql" -h %PGHOST% -p %PGPORT% -d "%CITYDB%" -U %PGUSER% -f "INSTALL_citydb_utilities.sql"
17 pause
  
```

Figure 2: Example of the CREATE\_DB\_citydb\_utilities.bat batch file



```

C:\WINDOWS\system32\cmd.exe
CREATE FUNCTION
DROP FUNCTION
CREATE FUNCTION
DROP TRIGGER
CREATE TRIGGER
DROP TRIGGER
CREATE TRIGGER
DROP TRIGGER
CREATE TRIGGER
DO
      installation_result
-----
3DCityDB view triggers installed correctly!

DROP TRIGGER
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Installation of the 3DcityDB Utilities Package complete!
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

C:\Users\agugiarog\Google Drive\citydb\00_utilities>pause
Press any key to continue . . .

```

Figure 3: Message upon successful installation of the 3DCityDB Utilities Package

## 2.3 (Alternative) manual installation

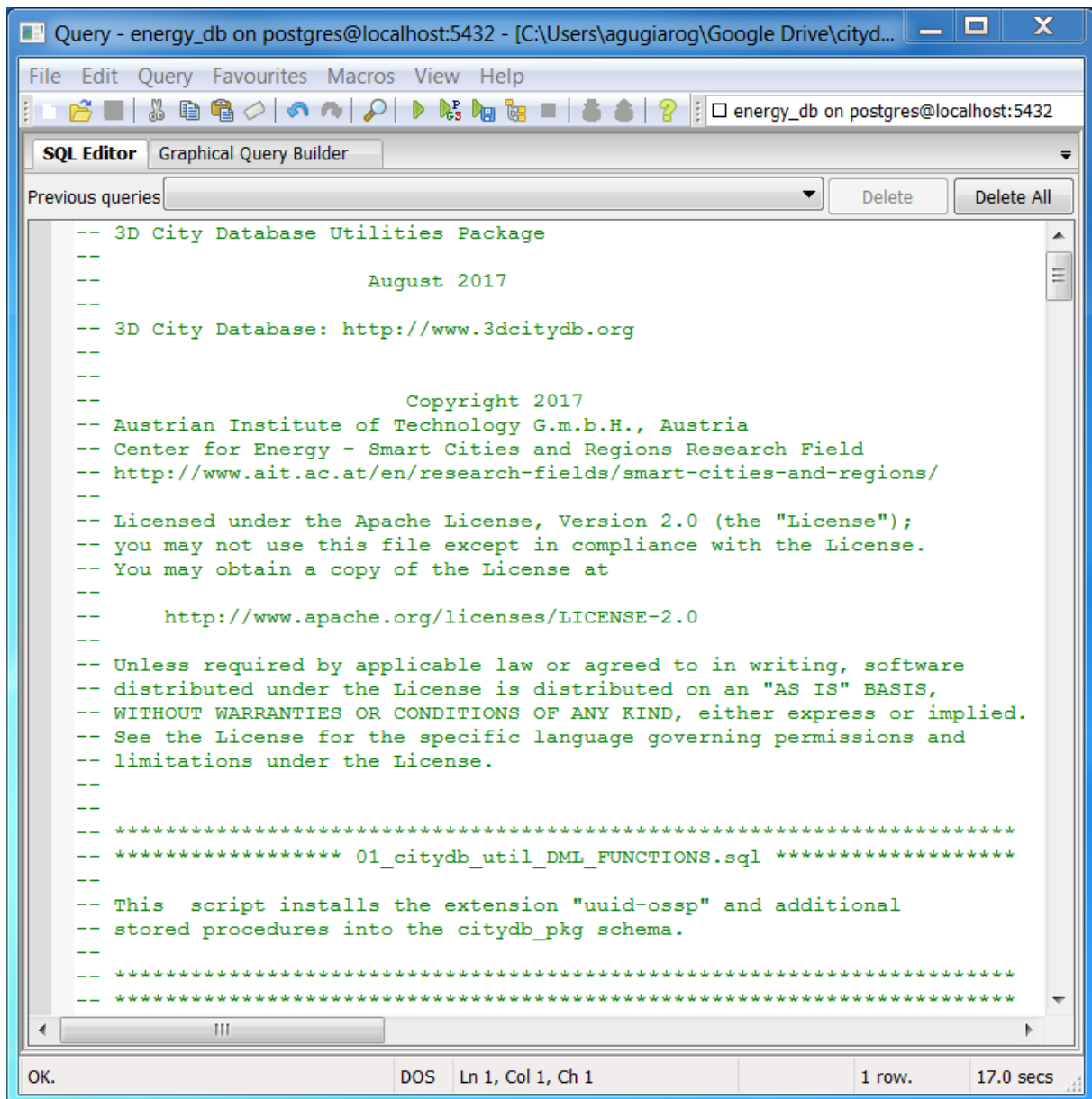
As an alternative, manual, installation process, the user must **sequentially** launch the following SQL scripts, using for example the PgAdmin GUI, which is generally installed together with any recent PostgreSQL installation package. See for example Figure 4, where PgAdmin III v. 1.22 is being used. Upon successful installation, messages like the one shown in Figure 5 **Error! Reference source not found.** will be output.

The SQL scripts can be found in the \00\_utilities\postgresql subfolder and are:

```

02_citydb_util_DML_FUNCTIONS.sql,
03_citydb_util_TABLES.sql,
05_citydb_util_TABLE_DATA.sql,
06_citydb_util_VIEWS.sql,
07_citydb_util_VIEW_FUNCTIONS.sql,
08_citydb_util_VIEW_TRIGGERS.sql.

```



**Figure 4: Installation of the Utilities Package using PgAdmin III**

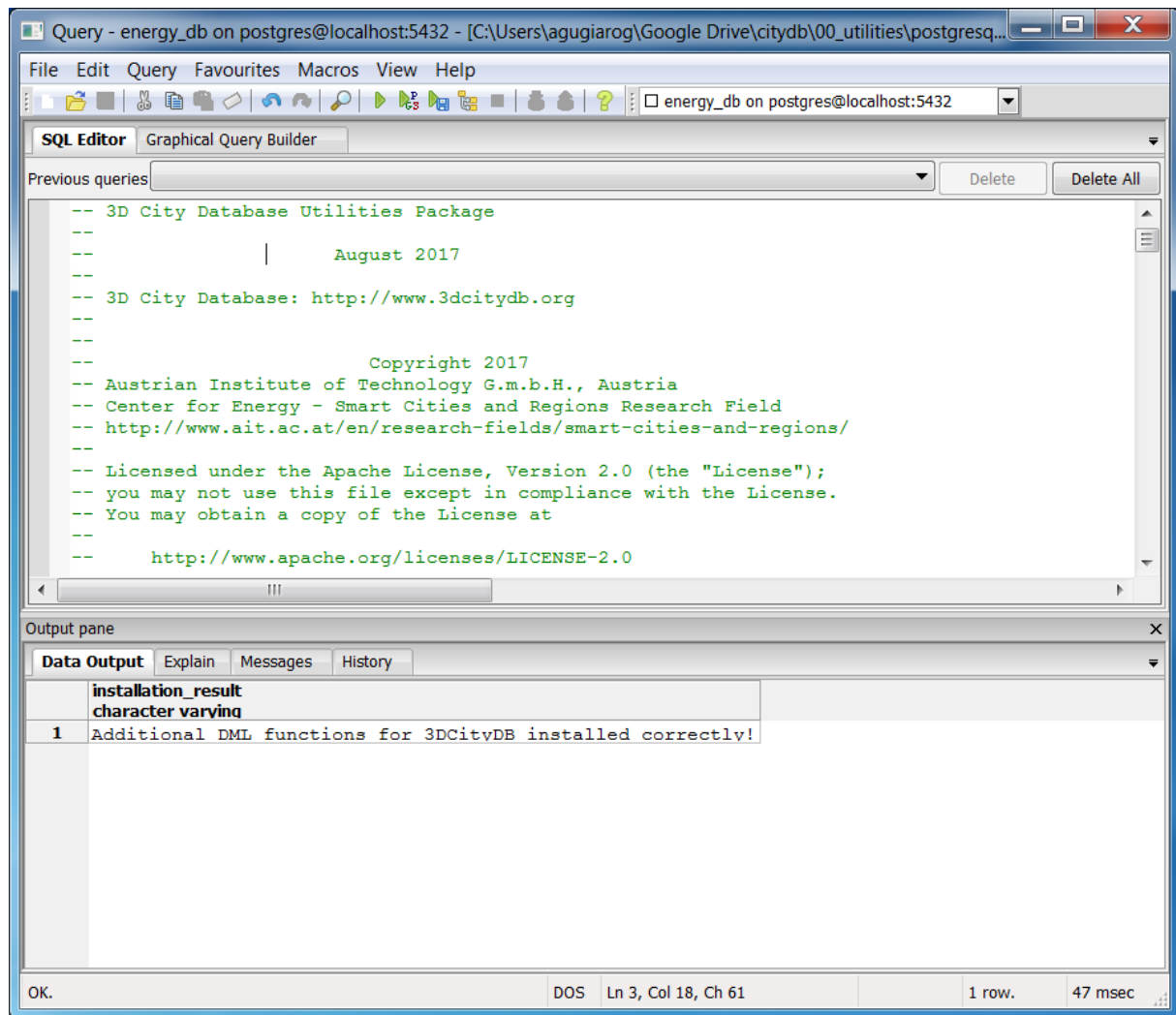


Figure 5: Upon successful installation, the resulting message can be read in the lower part of the window (Output pane)

### 3 Document revision notes

Date	Author	Notes
15 September 2017	Agugiaro Giorgio	First draft of the documentation
22 May 2018	Agugiaro Giorgio	Correction of typos. Improved text. Extended list of views. Added text about lookup tables. Added Chapter 3 “Document revision notes”



## Notes

## Notes