3D City Database extension

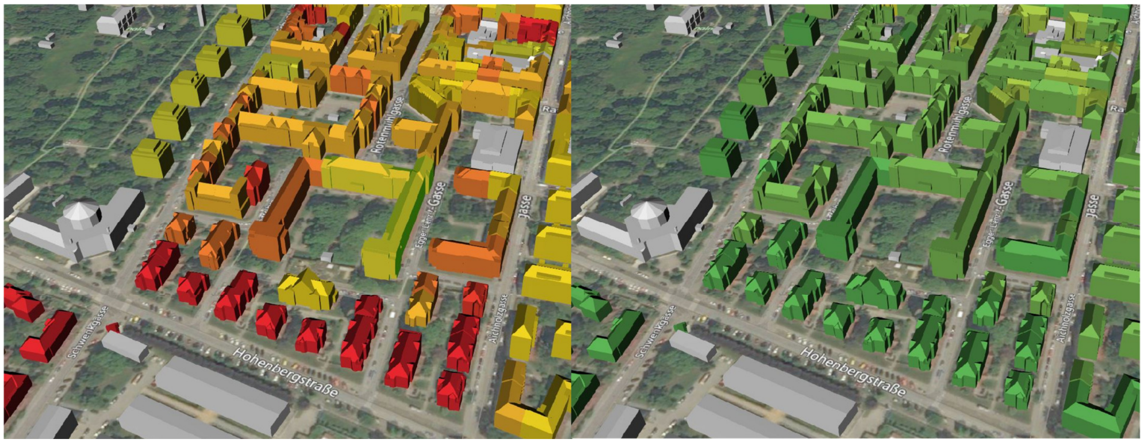
for the

CityGML Scenario ADE 0.2

PostgreSQL Version

Documentation

Last update: 14 September 2017



AUSTRIAN INSTITUTE
OF TECHNOLOGY
TOMORROW TODAY

Active participants in development

Name	Institution	Email
Giorgio Agugiaro	AIT – Austrian Institute of Technology G.m.b.H. Center for Energy – Smart Cities & Regions	giorgio.agugiaro@ait.ac.at

Acknowledgements

The 3D City Database extension for the CityGML Scenario ADE 0.1 (PostgreSQL version) was partially developed in the framework of the following international projects:

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://integrcity.epfl.ch

Finally, the author would like to thank Edmund Widl (AIT), Pablo Puerto (CREM) and Maximilian Sindram (TU München) for the suggestions and the fruitful discussions leading to the conception and implementation of the ADE.

Table of Contents

1 INTRODUCTION 9

1.1 System and design decisions 9

2 DATA MODELLING 11

2.1 Time series module 11

2.2 Core module 12

3 DATABASE DESIGN 15

3.1 Database schema of the Scenario ADE 15

3.1.1 Time series module 15
3.1.2 Core module 16
3.1.3 Lookup tables 17
3.1.4 Sequences 18
3.1.5 Stored procedures 18
3.1.6 Views 20

4 TEST DATA 23

5 INSTALLATION 24

5.1 System requirements 24

5.2 Automatic installation 24

5.3 Manual installation 26

5.4 Installation of the test data 28

NOTES 29

Disclaimer

The *3D City Database extension for the CityGML Scenario Application Domain Extension (ADE),* 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.

1 Introduction

The increasing diffusion and adoption of CityGML-based semantic 3D city models in the recent years has led to a growing number of applications being developed upon them. The application fields are numerous, ranging from urban planning, noise mapping, augmented reality, utility network management to energy simulation tools – just to name a few.

Due to the intrinsic advantages offered by the coherent and integrated information hub a CityGML-based semantic city model represents, a common functionality that many applications develop is the possibility to define scenarios, compute some results (e.g. by means of simulation tools) and Key Performance Indicators in order to assess qualitatively *and* quantitatively the performance or the effects of selected *measures* carried out either on a city model. The city model can represent the current situation (status quo) or be one of many different scenario-derived city models stemming from the status-quo one after a number of changes and modifications.

At the moment, however, CityGML does not provide a standardised way of dealing with scenarios, scenario-dependent results, or the description/documentation of the changes required to obtain a city model y from a city model x. On the other hand, thanks to the extensibility offered by CityGML in terms of Applications Domain Extensions, this current lack can be bridged.

The **Scenario ADE** presented in this document is a first proposal to define a data model to deals with scenario-dependent entities, and it comes also as a database schema implementation for the free and open-source 3D City Database.

The wish is that the availability of a common, shared database schema will contribute and foster adoption and further improvement of the Scenario ADE in a sort of virtuous circle. Hence, any feedback, constructive suggestions and help to correct, improve and test the current implementation are greatly welcome.

1.1 System and design decisions

Currently, the 3D City Database schema is provided for Oracle with the Spatial or Locator licensing option (10G R2 or higher) and PostgreSQL (9.1 or higher) with PostGIS extension (2.0 or higher). The 3DCityDB extension for the Scenario ADE requires the latest version of the 3DCityDB (v. 3.3.0) and is (currently) implemented for PostgreSQL only. As the overall goal of this extension is to facilitate direct connection and usage of the 3DCityDB relational database by users and applications programmers, a number of design and implementation decisions were taken according to the inspiring criteria briefly listed in the following:

a) Define a non-concurrent way of extending the 3DCityDB with other ADEs at the same time, for example adopting naming prefixes to avoid potential homonymy conflicts with other ADEs or standard 3DCityDB objects;

b) Build upon the existing objects of the 3DCityDB (relations, stored procedures, etc.), but keep the original ones untouched in order to guarantee the normal usage of the Importer/Exporter tool with “plain” CityGML instance documents;
c) Try to stay as close as possible to the original design “style” of the 3DCityDB when it comes to tables, constraints, naming conventions, data types, etc.;
d) Follow the v. 0.1 of the Scenario ADE as close as possible. This means also that tables and attribute names in the database are kept as close as possible to the original names of the Scenario ADE as indicated in the UML diagrams;
e) Keep the number of tables in check, i.e. grouping and merging multiple classes into one table – whenever reasonable –, although at the cost of some inefficiencies (e.g. in terms of disk space consumption) or some design “inelegances”;
f) Follow the conventions adopted for the 3DCityDB documentation when writing this document, in order to guarantee a certain layout and visual coherency.

For these reasons, several concepts will be only briefly mentioned in this document, while providing a reference to the existing documentation. The reader is therefore encouraged to keep at least a copy of the 3DCityDB and of the Scenario ADE documentation at hand. The **3DCityDB documentation** can be accessed on-line at this URL:

https://github.com/3dcitydb/3dcitydb/tree/master/Documentation

Finally, especially for those users who are completely new to the 3DCityDB, a very useful **hands-on tutorial**, where the most important steps to set up the database and use the Importer/Exporter are described, can be retrieved here:

https://github.com/3dcitydb/tutorials

2 Data Modelling

In the following pages the data model of the Scenario ADE is briefly presented and explained. UML diagrams are also depicted, in order to better show the Scenario ADE elements and where they connect to the existing CityGML classes.

For intuitive understanding, classes that will be merged to a single table in the relational schema are shown as orange blocks in the UML diagrams, while n:m relations, which only can be represented by additional tables, are represented as green blocks.

In the current implementation, the Scenario ADE consists of two parts:

- The time series module;
- The core module.

2.1 Time series module

This module introduces one main class, **_TimeSeries**, essential to model the time-depending inputs and results. This class is used in the core module of the Scenario ADE.

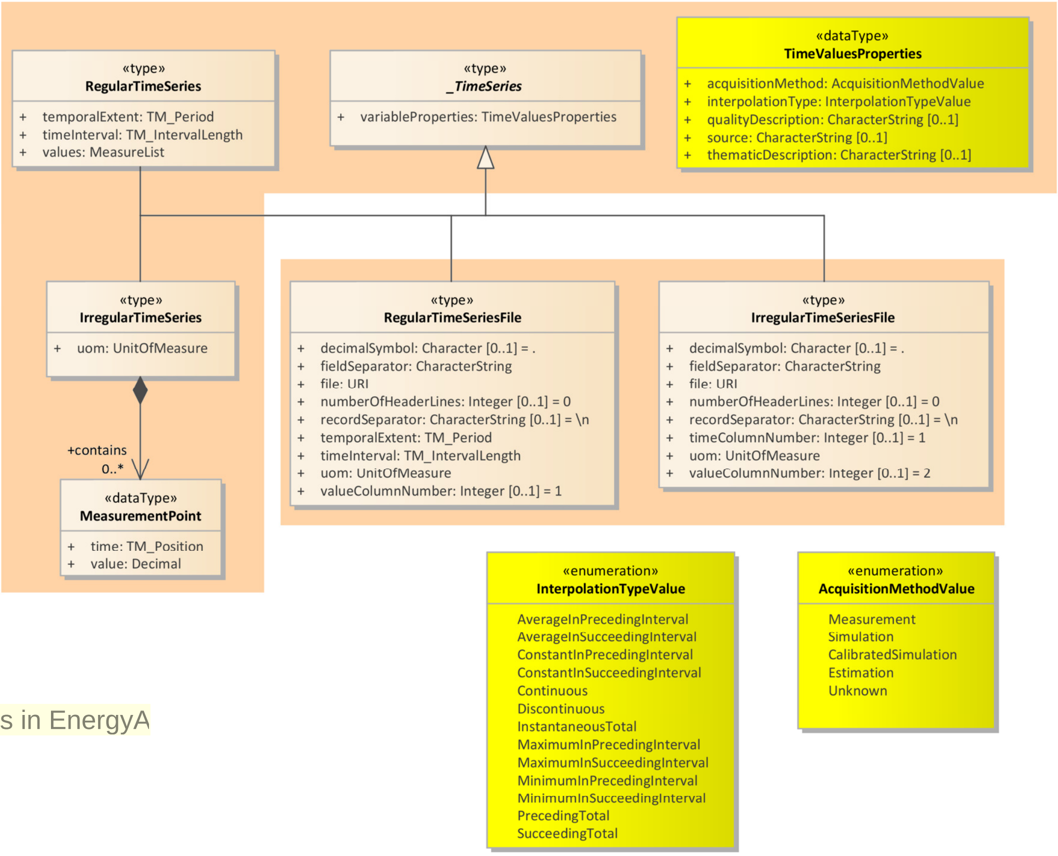
All _TimeSeries subclasses share some common properties, gathered in the TimeValuesProperties class, regarding for example the acquisition method, the type of interpolation, the source of the time series data, etc. Please refer to Figure 1 for the UML diagram.

Time series can be either regular or irregular. Regular time series contain values generated at regularly spaced interval of time, over a given temporal extent (i.e. start, end and duration time). In irregular time series, data follows a temporal sequence, but the measurement points may not happen at a regular time interval. Therefore, each value must be associated with its own timestamp.

Time series values may be stored in an external file (e.g. csv or text), both for regular (**RegularTimeSeriesFile** class) and irregular time series (**IrregularTimeSeriesFile** class). A number of attributes must be provided as metadata about the file, for example about the decimal symbol, the record and field separators, etc.

Please note: the time series class, and all its dependencies, correspond to those of the Energy ADE 0.8[1]. This is due to the decision to keep the data model of the time series in line between the two ADEs and simplify therefore the management of time-dependent data.

[1] https://github.com/gioagu/3dcitydb_ade/tree/master/02_energy_ade/diagrams


«type»
RegularTimeSeries
+ temporalExtent: TM_Period
+ timeInterval: TM_IntervalLength
+ values: MeasureList
«type»
_TimeSeries
+ variableProperties: TimeValuesProperties
«dataType»
TimeValuesProperties
+ acquisitionMethod: AcquisitionMethodValue
+ interpolationType: InterpolationTypeValue
+ qualityDescription: CharacterString [0..1]
+ source: CharacterString [0..1]
+ thematicDescription: CharacterString [0..1]
«type»
IrregularTimeSeries
+ uom: UnitOfMeasure
+contains
0..*
«dataType»
MeasurementPoint
+ time: TM_Position
+ value: Decimal
«type»
RegularTimeSeriesFile
+ decimalSymbol: Character [0..1] = .
+ fieldSeparator: CharacterString
+ file: URI
+ numberOfHeaderLines: Integer [0..1] = 0
+ recordSeparator: CharacterString [0..1] = \n
+ temporalExtent: TM_Period
+ timeInterval: TM_IntervalLength
+ uom: UnitOfMeasure
+ valueColumnNumber: Integer [0..1] = 1
«type»
IrregularTimeSeriesFile
+ decimalSymbol: Character [0..1] = .
+ fieldSeparator: CharacterString
+ file: URI
+ numberOfHeaderLines: Integer [0..1] = 0
+ recordSeparator: CharacterString [0..1] = \n
+ timeColumnNumber: Integer [0..1] = 1
+ uom: UnitOfMeasure
+ valueColumnNumber: Integer [0..1] = 2
«enumeration»
InterpolationTypeValue
AverageInPrecedingInterval
AverageInSucceedingInterval
ConstantInPrecedingInterval
ConstantInSucceedingInterval
Continuous
Discontinuous
InstantaneousTotal
MaximumInPrecedingInterval
MaximumInSucceedingInterval
MinimumInPrecedingInterval
MinimumInSucceedingInterval
PrecedingTotal
SucceedingTotal
«enumeration»
AcquisitionMethodValue
Measurement
Simulation
CalibratedSimulation
Estimation
Unknown

very same as in EnergyA

Figure 1: UML diagram of the Time series module of the Scenario ADE

2.2 Core module

The core module (Figure 2) is built around the main class **Scenario**. In the Scenario ADE, a scenario is intended as a *unique* composition of:

- a set of physical objects, be it a whole city model, a single city object, or a group of city objects (to be modelled by means of the CityObjectGroup class, which is itself derived from a _CityObject). For this purpose, the class Scenario is associated to both the CityGML **CityModel** and **_CityObject** classes with cardinality [0..1];
- a number of (optional) scenario parameters (modelled by the class **ScenarioParameter**). These parameters can be used to define some initial conditions of the scenario (i.e. as input parameters) and can contain information about possible constraints (e.g. the desired target CO2-emissions must be lower than a certain lever). Alternatively, the ScenarioParameter class is intended to model also any type of results associated to a scenario and obtained by data analyses or simulations (i.e. as output parameters). For this purpose, some attributes are defined to describe the origin of the data (e.g. from a simulation) and the type and temporal level of aggregation. It is possible to store data of any type: Boolean, text, integer, real, etc., and time series data, too;

- information about resources required by a specific scenario (e.g. in terms of money, energy, time, etc.) can optionally be added, too, by means of the abstract class _Resource and its derived classes.

Finally, it is possible to add information about the operations carried out on a specific CityModel or _CityObject in order to achieve the desired configuration the scenario parameters refer to. It is here possible for example, to describe how, starting from a certain CityModel, particular GML features (e.g. city objects) were added, removed or changed. In this way, changes from one initial configuration (geometries or attributes, or both) can be recorded and documented, in order to facilitate the scenario generation process. For this purpose, the classes **AddFeature**, **RemoveFeature**, **ChangeFeatureAttribute** – derived from abstract class **_Operation** – can be used directly as simple, atomic operations (through **_SimpleOperation**) or combined by means of the **ComplexOperation** class. An operation can be associated to resources, too.

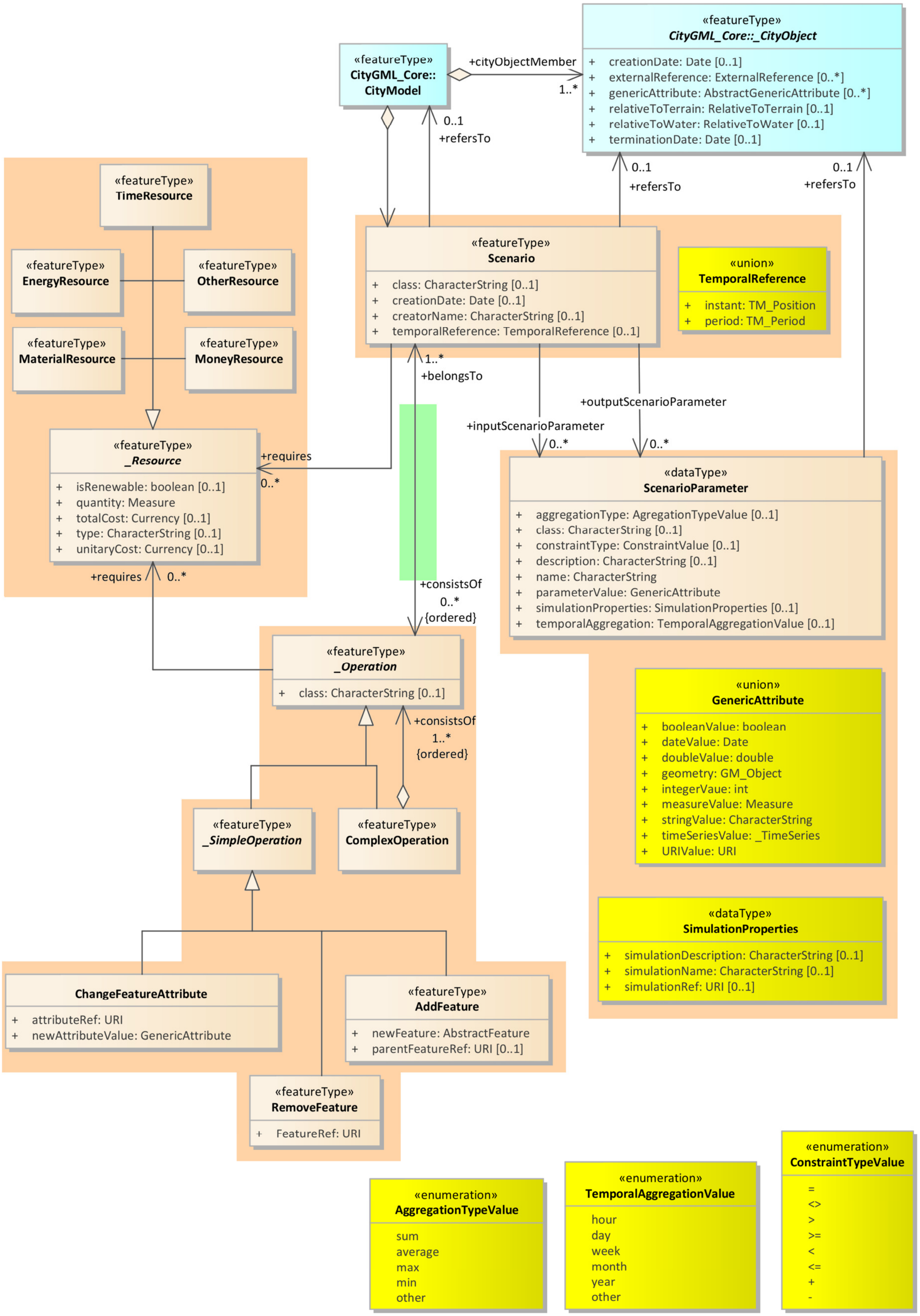

«featureType»
CityGML_Core::_CityObject
+ creationDate: Date [0..1]
+ externalReference: ExternalReference [0..*]
+ genericAttribute: AbstractGenericAttribute [0..*]
+ relativeToTerrain: RelativeToTerrain [0..1]
+ relativeToWater: RelativeToWater [0..1]
+ terminationDate: Date [0..1]
«featureType»
CityGML_Core::
CityModel
+cityObjectMember
1..*
0..1
+refersTo
0..1
+refersTo
0..1
+refersTo
«featureType»
TimeResource
«featureType»
EnergyResource
«featureType»
OtherResource
«featureType»
MaterialResource
«featureType»
MoneyResource
«featureType»
_Resource
+ isRenewable: boolean [0..1]
+ quantity: Measure
+ totalCost: Currency [0..1]
+ type: CharacterString [0..1]
+ unitaryCost: Currency [0..1]
+requires
0..*
+requires
0..*
«featureType»
Scenario
+ class: CharacterString [0..1]
+ creationDate: Date [0..1]
+ creatorName: CharacterString [0..1]
+ temporalReference: TemporalReference [0..1]
«union»
TemporalReference
+ instant: TM_Position
+ period: TM_Period
1..*
+belongsTo
+outputScenarioParameter
+inputScenarioParameter
0..*
0..*
«dataType»
ScenarioParameter
+ aggregationType: AgregationTypeValue [0..1]
+ class: CharacterString [0..1]
+ constraintType: ConstraintValue [0..1]
+ description: CharacterString [0..1]
+ name: CharacterString
+ parameterValue: GenericAttribute
+ simulationProperties: SimulationProperties [0..1]
+ temporalAggregation: TemporalAggregationValue [0..1]
+consistsOf
0..*
{ordered}
«featureType»
_Operation
+ class: CharacterString [0..1]
+consistsOf
1..*
{ordered}
«featureType»
_SimpleOperation
«featureType»
ComplexOperation
«union»
GenericAttribute
+ booleanValue: boolean
+ dateValue: Date
+ doubleValue: double
+ geometry: GM_Object
+ integerVaue: int
+ measureValue: Measure
+ stringValue: CharacterString
+ timeSeriesValue: _TimeSeries
+ URIValue: URI
«dataType»
SimulationProperties
+ simulationDescription: CharacterString [0..1]
+ simulationName: CharacterString [0..1]
+ simulationRef: URI [0..1]
ChangeFeatureAttribute
+ attributeRef: URI
+ newAttributeValue: GenericAttribute
«featureType»
AddFeature
+ newFeature: AbstractFeature
+ parentFeatureRef: URI [0..1]
«featureType»
RemoveFeature
+ FeatureRef: URI
«enumeration»
AggregationTypeValue
sum
average
max
min
other
«enumeration»
TemporalAggregationValue
hour
day
week
month
year
other
«enumeration»
ConstraintTypeValue
=
<>
>
>=
<
<=
+
-

Figure 2: UML diagram of the Core module of the Scenario ADE

3 Database design

This section presents and describes the implementation of the Scenario ADE as a relational database schema for PostgreSQL.

3.1 Database schema of the Scenario ADE

In the following paragraphs, the tables of the relational schema are described in detail and displayed graphically. The description is based on the remarks on UML diagrams in section 2. Focus is put on situations where the conversion into tables leads to changes in the ER model.

In general, the database relations correspond to the Scenario ADE v. 0.2. All database objects are prefixed with the "scn_", in order to comply with the requirements of the **Metadata module** for the 3DCityDB[2].

The figures are taken from PostgreSQL Maestro, a GUI-based admin tool for database management which offers a set of tools to edit and execute SQL scripts, build visual diagrams, etc. In future version of this document, the open-source tool pgModeler will be used to maintain the schema.

3.1.1 Time series module

For the _TimeSeries class (and derived/dependent classes), only two tables are created. Please refer to Figure 3 for the representation of the corresponding ER diagram.

In the **SCN_TIMESERIES** table most of the attributes of all _TimeSeries subclasses are merged. The table contains the attribute OBJECTCLASS_ID which references table OBJECTCLASS and allows to distinguish between the four type of times series. It also contains the GMLID and GMLID_CODESPACE attributes, plus the attributes from the TimeValuesProperties class. The next free ID value is provided by the database sequence SCN_TIMESERIES_ID_SEQ. Regular time series must be stored as an array of values in column VALUES_ARRAY. The units of measure are stored in the corresponding attribute VALUES_UNIT. For the irregular time series, the same approach still holds, but the vector of values is stored together with an array of ordered timestamps in the attribute TIMESTAMP_VALUES.

Please note that the adoption of an array type for the columns VALUES_ARRAY and TIME_ARRAY is PostgreSQL-specific and does not follow the 3DCityDB implementation rules which convert multiple occurrences of the same element into a string, where the values are separated by the '--/\--' string sequence (e.g. with attributes FUNCTION and USAGE). However, in this (only) case, this implementation decision was made in order to take advantage of the specific array functions already available in PostgreSQL.

Please note that, in order to input an array in PostgreSQL, two main methods exist. In the case of the VALUES_ARRAY and TIME_ARRAY columns, the values can be input as a string of

[2] https://github.com/gioagu/3dcitydb_ade/tree/master/01_metadata_module

comma-separated values between two curly braces (e.g. {1, 3, 5, 6, 7, 2}) or using the PostgreSQL array constructor ARRAY (e.g. ARRAY[1, 3, 5, 6, 7, 2])[3].

The remaining attributes for file-based time series are merged in the table **SCN_TIMESERIES_FILE**, which is linked to the SCN_TIMESERIES table by means of the foreign key ID. Additionally, the Boolean attribute IS_COMPRESSED is provided, although not included in the original UML diagrams.

3.1.2 Core module

Table **SCN_SCENARIO** corresponds to class Scenario. It contains the ID, GMLID, GMLID_CODESPACE, NAME, DESCRIPTION attributes. The attribute OBJECTCLASS_ID references the table OBJECTCLASS. The next free ID value is provided by the database sequence SCN_SCENARIO_ID_SEQ. The optional attributes the temporal reference (INSTANT, PERIOD_BEGIN and PERIOD_END) can be used to better characterise the temporal frame of a scenario (e.g. for 2030, 2050, etc.). The foreign keys CITYMODEL_ID and CITYOBJECT_ID refer to tables CITYMODEL and CITYOBJECT, respectively.

Table **SCN_SCENARIO_PARAMETER** corresponds to class ScenarioParameter. It contains the ID, NAME, DESCRIPTION attributes. The next free ID value is provided by the database sequence SCN_SCENARIO_PARAMETER_ID_SEQ. The attribute TYPE is used to distinguish between input and output scenario parameters. It is constrained to be *not null* and can contain values what are either "input" or "output". The foreign key TIME_SERIES_ID references table SCN_TIME_SERIES.

Table **SCN_RESOURCE** corresponds to class _Resource and merges all its subclasses (EnergyResource, MoneyResource, etc.). It contains the ID, GMLID, GMLID_CODESPACE, NAME, DESCRIPTION attributes. The attribute OBJECTCLASS_ID references the table OBJECTCLASS. The next free ID value is provided by the database sequence SCN_RESOURCE_ID_SEQ. The foreign keys SCENARIO_ID and OPERATION_ID refer to tables SCN_SCENARIO and SCN_OPERATION, respectively.

Table **SCN_OPERATION** corresponds to class _Operation and merges all its subclasses (AddFeature, RemoveFeature, ComplexOperation, etc.). It contains the ID, GMLID, GMLID_CODESPACE, NAME, DESCRIPTION attributes. The attribute OBJECTCLASS_ID references the table OBJECTCLASS. The next free ID value is provided by the database sequence SCN_OPERATION_ID_SEQ. The foreign key TIME_SERIES_ID references table SCN_TIME_SERIES.

By means of the table **SCN_SCENARIO_TO_OPERATION** the relation between the SCN_SCENARIO and the SCN_OPERATION tables is established, in that the foreign keys SCENARIO_ID and OPERATION_ID refer to the respective tables. Upon delete and update of the referenced tables, the corresponding record is deleted or updated automatically.

[3] https://www.postgresql.org/docs/9.1/static/arrays.html

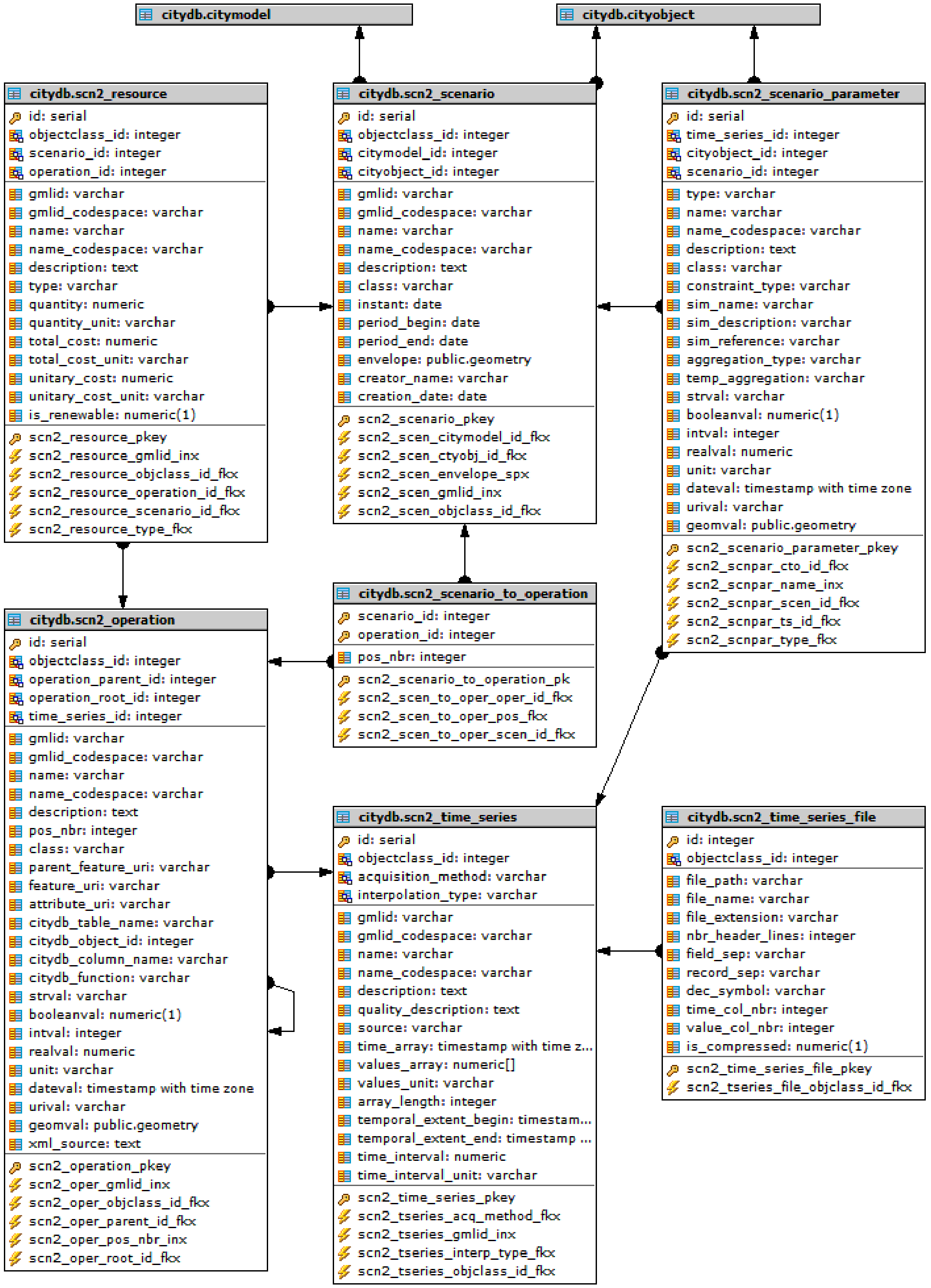


Figure 3: ER diagram of the Scenario ADE

3.1.3 Lookup tables

A number of lookup tables is also provided. They implement enumerations (or codelists) defined in the Scenario ADE. All tables are stored in the *citydb* schema and are named by adding the “scn_lu_” prefix to each table. Table 1 contains the list of the lookup tables shipped with the current implementation of the Scenario ADE and the class they correspond to.

LOOKUP TABLE	SCENARIO ADE CLASS	REFERENCING TABLE(S)
scn_lu_acquisition_method	AcquisitionMethodValue	scn_time_series
scn_lu_aggreagation_type	AggregationTypeValue	scn_scenario_parameter
snc_lu_constraint_type	ConstraintTypeValue	scn_scenario_parameter
scn_lu_interpolation	InterpolationTypeValue	scn_time_series
scn_lu_temporal_aggregation	TemporalAggregationValue	scn_scenario_parameter

Table 1: Lookup tables shipped with this implementation of the Scenario ADE. All lookup tables are stored in schema *citydb* and prefixed with “*scn_lu_*”

3.1.4 Sequences

Upon installation, a number of database sequences is also generated. They can be identified by means of the db_prefix (e.g. “scn_”). Analogously to other 3DCityDB tables, it is highly recommended to generate ID values for the respective Scenario ADE tables by using the predefined sequences only.

Please note that, unlike in the 3DCityDB SQL scripts, all new sequences are generated directly within the CREATE TABLE statement as follows:

```
CREATE TABLE citydb.scn_scenario (
id serial PRIMARY KEY,
name varchar,
…);
```

As a matter of fact, the automatically created sequence corresponds to the statement

```
CREATE SEQUENCE scn_scenario_id_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 9223372036854775807
  START 1
  CACHE 1;
ALTER TABLE scn_scenario_id_seq OWNER TO postgres;
```

Although theoretically equivalent, a practical difference has been found so far. Safe Software’s FME PostGIS and PostgreSQL writers can profit of the auto-increment property of serial types in INSERT statements only if a sequence is created with a CREATE TABLE statement. Please read the last answer in https://knowledge.safe.com/questions/1703/serial-data-types-and-insert-into-postgresql.html

3.1.5 Stored procedures

The 3DCityDB extension for the Scenario ADE is shipped with a set of stored procedures (or “functions”, in the PostgreSQL jargon). They are automatically installed during the setup. In the PostgreSQL version, stored procedures are written in PL/pgSQL and stored in the database schema *citydb_pkg*. Coherently with the naming rules, all stored procedures are prefixed with the db_prefix (e.g. "scn"). Please note that, as already mentioned, the input parameters are omitted and indicated by a simple "…" in the following text for better readability.

At the moment, two main families of stored procedures are shipped. They can be generally grouped into the "delete" and "get_envelope" stored procedures.

A "delete" stored procedure allows to delete an Scenario ADE object and all its dependences at once. All stored procedures of this kind are named `scn_delete_*(…)`, where * is generally the corresponding name of the object to be deleted from the database. So, `scn_delete_scenario(scenario_id)` will delete all data related to the scenario having as ID the `scenario_id` (e.g. 22456). Conceptually, these stored procedures reflect the analogous ones shipped with the "vanilla" 3DCityDB.

Please note: the "vanilla" 3DCityDB also provides stored procedures to delete multiple objects at a time by means of the lineage column in the CITYOBJECT table. Such stored procedures are named with plural names, e.g. `delete_buildings(…)` or `delete_cityobjectgroups(…)` (please note the **s** at the end of the names).

The current implementation of the Scenario ADE still lacks such lineage-based stored procedures, they will be added in future releases.

Finally, a number of insert_*() stored procedures is also shipped and installed in the schema *citydb_pkg*. These insert_*() stored procedures comply with the criteria defined in the **3DCityDB Utilities Package**[4].

In Table 2 all stored procedures shipped with this implementation of the Scenario ADE are listed.

DELETE STORED PROCEDURES
scn_delete_cityobject(…)
scn_delete_operation(…)
scn_delete_resource(…)
scn_delete_scenario_parameter(…)
scn_delete_scenario(…)
scn_delete_time_series(…)
scn_intern_delete_cityobject(…)
GET ENVELOPE STORED PROCEDURES
N/A
INSERT STORED PROCEDURES
scn_insert_operation(…)
scn_insert_resource(…)
scn_insert_scenario(…)
scn_insert_scenario_parameter(…)
scn_insert_scenario_to_operation(…)

[4] https://github.com/gioagu/3dcitydb_ade/tree/master/00_utilities_package/manual

scn_insert_time_series(…)
scn_insert_time_series_file(…)
MISCELLANEAOUS STORED PROCEDURES
scn_cleanup_schema(…)
scn_set_ade_columns_srid(…)

Table 2: Stored procedures shipped with this implementation of the Scenario ADE. All stored procedures are stored in schema *citydb_pkg*.

3.1.6 Views

As data belonging to a specific class can be stored in different tables in the database, the database structure can be sometimes complex in terms of CRUD[5] operations (i.e. select, insert, update and delete), especially when dealing with the tables added by an ADE (and as long the Importer/Exporter does not support ADEs natively).

For this reason, and in order to facilitate CRUD operations at database level, some views are also provided. Views represent a facilitated way to access data, ideally providing one single table that "hides" the complex structure of the actual database. Moreover, if a view is also updatable, entries can be updated, inserted and deleted directly interacting with the view. A series of triggers and trigger functions must be implemented therefore for each view.

All view names are prefixed with the *db_prefix*. All views are installed in a specific, newly added, database schema named *citydb_view*.

Please note: in order to access *any* database object in the *citydb_view* schema, **qualified names**[6] 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 3 contains all views shipped with this version of the 3DCityDB extension for the Scenario ADE.

VIEW	UPDATABLE?
scn_operation_complex	
scn_operation_simple	
scn_operation_simple_add	
scn_operation_simple_change	
scn_operation_simple_change_ts	
scn_operation_simple_remove	
scn_resource	
scn_resource_energy	✓
scn_resource_material	✓
scn_resource_money	✓
scn_resource_other	✓

[5] CRUD represents an acronym for the database operations **C**reate, **R**ead, **U**pdate, and **D**elete.
[6] For more details, see https://www.postgresql.org/docs/9.1/static/ddl-schemas.html

VIEW	UPDATABLE?
scn_resource_time	✓
scn_scenario	✓
scn_scenario_parameter	✓
scn_scenario_parameter_ts	✓
scn_scenario_to_operation	
scn_time_series	
scn_time_series_irregular	✓
scn_time_series_irregular_file	✓
scn_time_series_regular	✓
scn_time_series_regular_file	✓

Table 3: Views shipped with this implementation of the Scenario ADE. All views are stored in schema *citydb_view*

In general, all views follow the same rules of those shipped with the **3DCityDB Utilities Package**. In addition to the views, a number of stored procedures, triggers and trigger functions will be added in future releases and installed both in schema *citydb_pkg* and in schema *citydb_view*, in order to make the views updatable. For more information, please refer to the documentation of the **3DCityDB Utilities Package** which can be downloaded at

https://github.com/gioagu/3dcitydb_ade/tree/master/00_utilities_package/manual

4 Test data

A small exemplary data set to be installed as test data is provided. It is intended to populate the tables with a simple yet consistent dataset and to simplify the understanding of the data model and its manipulation (read/write).

5 Installation

The 3D City Database extension for the Scenario ADE comes with a number of SQL scripts for setting up the relational schema in a PostgreSQL/PostGIS database upon which an instance of the 3DCityDB was previously installed. Detailed instruction on how to set up the 3DCityDB are contained in the **3DCityDB documentation**, accessible on-line at this URL:

https://github.com/3dcitydb/3dcitydb/tree/master/Documentation

Moreover, a very useful **hands-on tutorial**, where the most important steps to set up the database and use the Importer/Exporter are described, can be retrieved here:

https://github.com/3dcitydb/tutorials

The source code and documentation of the **3DCityDB extension of the Scenario ADE** for PostgreSQL can be retrieved here:

https://github.com/gioagu/3dcitydb_ade/tree/master/04_scenario_ade

Please follow the instructions on the next pages in order to complete a proper installation.

5.1 System requirements

In order to set up the 3DCityDB extension for the Scenario ADE, the 3DCityDB v.3.3.0 needs to be previously installed correctly. All system requirements of the 3DCityDB still apply. PostgreSQL is supported from v. 9.1, although v. 9.3 or higher is recommended. PostGIS 2.0 or higher is required.

Additionally, the **Metadata module** for the 3DCityDB and (optionally) and **3DCityDB Utilities Package** are required and need to be installed previously. Both can be downloaded from:

https://github.com/gioagu/3dcitydb_ade

5.2 Automatic installation

The Scenario ADE comes with a “base” and an “optional” installation part. With regards to the SQL scripts listed in the following, the “optional” part consists of the views and all related stored procedures to make them updatable (i.e. from script 06_Scenario_ADE_VIEWS.sql onward). If the user wants or needs also the “optional” part, then the 3DCityDB Utilities Package needs to be installed first.

In any case, in order to install the 3DCityDB extension for the Scenario ADE, the user needs to perform the steps described in the following.

Step 1 – Identify the owner of the citydb schema

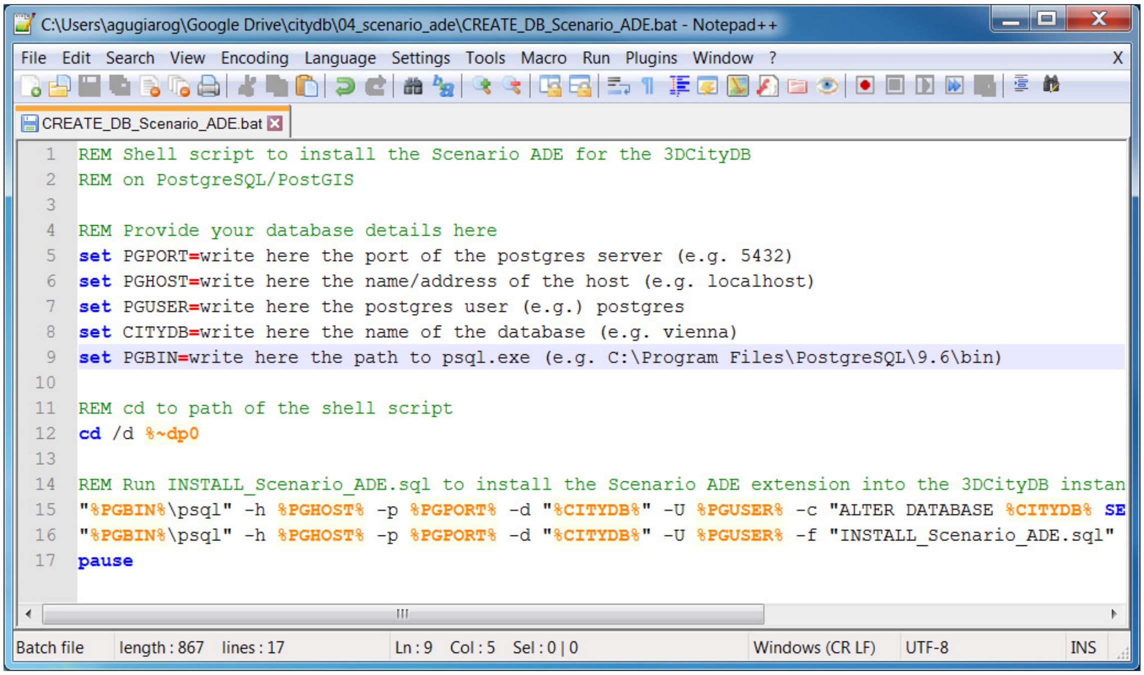
The installation of the 3DCityDB extension should conveniently be carried out using the same user that installed the *citydb* schema and all included relations.

Step 2 – Execute the batch file CREATE_DB_Scenario_ADE.bat

The batch file CREATE_DB_Scenario_ADE.bat can be run from a Windows command shell (or simply by double-clicking it). However, also in this case the configuration parameters regarding the PostgreSQL server and the selected database must be first adapted accordingly. An example is given in Figure 4. For Linux environments, the equivalent file CREATE_DB_Scenario_ADE.sh is also provided.

Please note: The batch file installs both the "base" and the "optional" parts. The user must therefore be sure to have previously installed the 3DCityDB Utilities Package.

Upon successful installation, the message shown in Figure 5 will be output.



C:\Users\agugiarog\Google Drive\citydb\04_scenario_ade\CREATE_DB_Scenario_ADE.bat - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
CREATE_DB_Scenario_ADE.bat
Batch file length : 867 lines : 17 Ln : 9 Col : 5 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

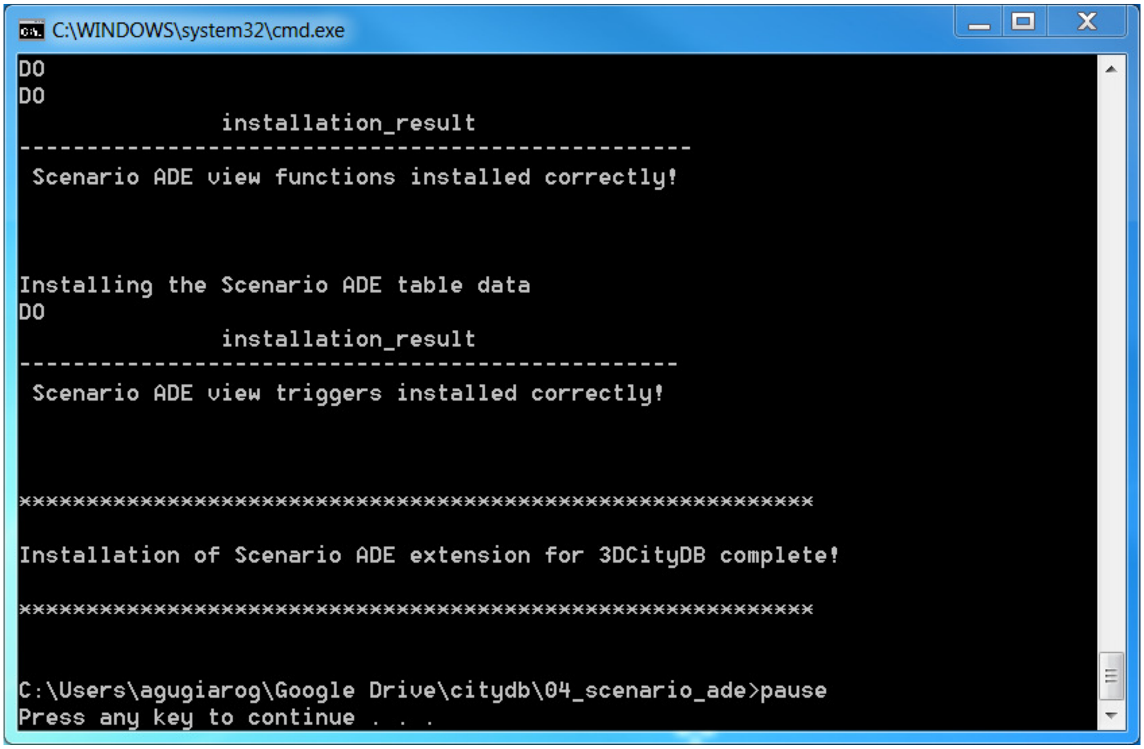
```
REM Shell script to install the Scenario ADE for the 3DCityDB
REM on PostgreSQL/PostGIS

REM Provide your database details here
set PGPORT=write here the port of the postgres server (e.g. 5432)
set PGHOST=write here the name/address of the host (e.g. localhost)
set PGUSER=write here the postgres user (e.g.) postgres
set CITYDB=write here the name of the database (e.g. vienna)
set PGBIN=write here the path to psql.exe (e.g. C:\Program Files\PostgreSQL\9.6\bin)

REM cd to path of the shell script
cd /d %~dp0

REM Run INSTALL_Scenario_ADE.sql to install the Scenario ADE extension into the 3DCityDB instan
"%PGBIN%\psql" -h %PGHOST% -p %PGPORT% -d "%CITYDB%" -U %PGUSER% -c "ALTER DATABASE %CITYDB% SE
"%PGBIN%\psql" -h %PGHOST% -p %PGPORT% -d "%CITYDB%" -U %PGUSER% -f "INSTALL_Scenario_ADE.sql"
pause
```

Figure 4: Example of the CREATE_DB_Scenario_ADE.bat batch file



```
DO
DO
               installation_result
--------------------------------------------------
 Scenario ADE view functions installed correctly!


Installing the Scenario ADE table data
DO
               installation_result
-------------------------------------------------
 Scenario ADE view triggers installed correctly!



***********************************************************

Installation of Scenario ADE extension for 3DCityDB complete!

***********************************************************


C:\Users\agugiarog\Google Drive\citydb\04_scenario_ade>pause
Press any key to continue . . .
```

Figure 5: Message upon successful installation of the Scenario ADE extension for the 3DCityDB

5.3 Manual installation

In order to install the 3DCityDB extension for the Scenario ADE, the user needs to perform the steps described in the following.

Step 1 – Identify the owner of the citydb schema

As seen before, the installation of the 3DCityDB extension should conveniently be carried out using the same user that installed the *citydb* schema and all included relations.

Step 2 – Execute the SQL scripts in folder *\04_scenario_ade\postgresql*

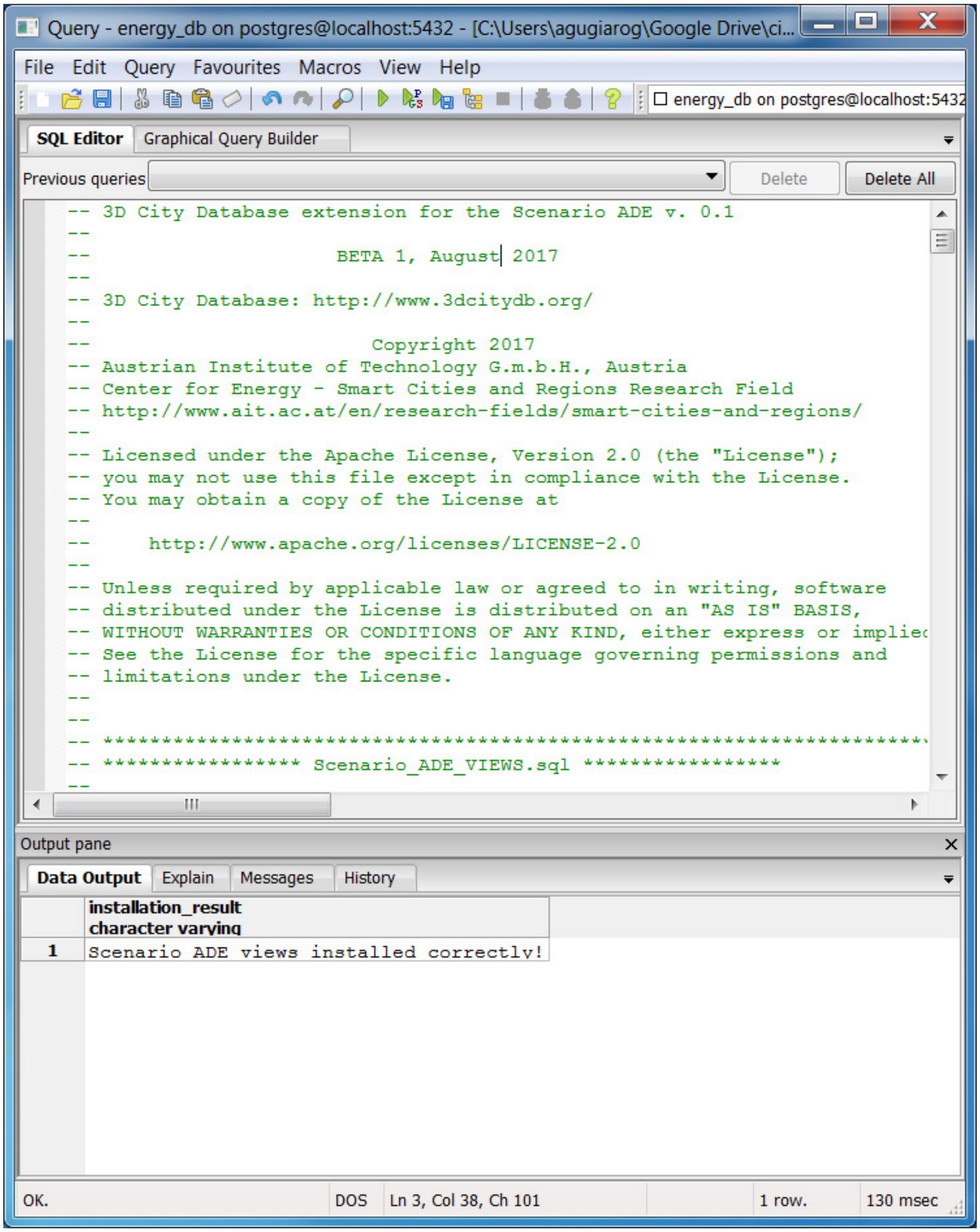
The SQL scripts in the folder ***\04_scenario_ade\postgresql***

01_Scenario_ADE_FUNCTIONS.sql,
02_Scenario_ADE_DML_FUNCTIONS.sql,
03_Scenario_ADE_TABLES.sql,
04_Scenario_ADE_TRIGGERS.sql,
05_Scenario_ADE_TABLE_DATA.sql,
06_Scenario_ADE_VIEWS.sql, (requires 3DCityDB Utilities Package)
07_Scenario_ADE_VIEW_FUNCTION.sql, (requires 3DCityDB Utilities Package)
08_Scenario_ADE_VIEW_TRIGGERS.sql, (requires 3DCityDB Utilities Package)

must simply run **sequentially** as seen before.

Upon installation, the corresponding Scenario ADE entry will be added to ADE table. All new tables will be added to the *citydb* schema, all new stored procedures will be installed in the *citydb_pkg* schema, and (optionally) all updatable views will be installed in the newly added *citydb_view* schema. All objects have names prefixed with the db_prefix value.

Upon successful installation, the message shown in Figure 6 will be output.



```
-- 3D City Database extension for the Scenario ADE v. 0.1
--
--                        BETA 1, August 2017
--
-- 3D City Database: http://www.3dcitydb.org/
--
--                           Copyright 2017
-- Austrian Institute of Technology G.m.b.H., Austria
-- Center for Energy - Smart Cities and Regions Research Field
-- http://www.ait.ac.at/en/research-fields/smart-cities-and-regions/
--
-- Licensed under the Apache License, Version 2.0 (the "License");
-- you may not use this file except in compliance with the License.
-- You may obtain a copy of the License at
--
--     http://www.apache.org/licenses/LICENSE-2.0
--
-- Unless required by applicable law or agreed to in writing, software
-- distributed under the License is distributed on an "AS IS" BASIS,
-- WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implie
-- See the License for the specific language governing permissions and
-- limitations under the License.
--
--
-- ***********************************************************************
-- ***************** Scenario_ADE_VIEWS.sql *****************
--
```

	installation_result character varying
1	Scenario ADE views installed correctly!

Figure 6: Message upon successful installation of the Scenario ADE extension for the 3DCityDB

5.4 Installation of the test data

In order to install the test data, an instance of the 3DCityDB using the **EPSG code 31256** is needed. In addition, both the Metadata module and the 3DCityDB extension for the Scenario ADE need to be installed, as described in the previous sections.

All test data are contained within on single SQL script file, named Scenario_ADE_TEST_DATA.sql and located in the \04_scenario_ade\test_data folder.

The installation process is the same as before: the script can be run from within an SQL command window of PgAdmin.

Notes

Notes